Package ‘Rfast2’

March 23, 2022

Type  Package
Title  A Collection of Efficient and Extremely Fast R Functions II
Version  0.1.3
Date  2022-03-22
Author  Manos Papadakis, Michail Tsagris, Stefanos Fafalios and Marios Dimitriadis.
Maintainer  Manos Papadakis <rfastofficial@gmail.com>
Depends  R (>= 3.5.0), Rcpp (>= 0.12.3)
LinkingTo  Rcpp (>= 0.12.3), RcppArmadillo
Imports  Rfast, RANN
SystemRequirements  C++11

BugReports  https://github.com/RfastOfficial/Rfast2/issues
URL  https://github.com/RfastOfficial/Rfast2
Description  A collection of fast statistical and utility functions for data analysis. Functions for regression, maximum likelihood, column-wise statistics and many more have been included. C++ has been utilized to speed up the functions.
License  GPL (>= 2.0)
NeedsCompilation  yes
Repository  CRAN
Date/Publication  2022-03-23 07:50:24 UTC

R topics documented:

Rfast2-package ................................................................. 4
Add many single terms to a model ..................................... 5
Angular Gaussian random values simulation ......................... 6
Anova for circular data ..................................................... 7
Backward selection with the F test or the partial correlation coefficient ..................................................... 9
Benchmark - Measure time ............................................... 10
BIC of many simple univariate regressions ........................... 11
Binomial regression .......................................................... 12
Bootstrap James and Hotelling test for 2 independent sample mean vectors  
Bootstrap Student’s t-test for 2 independent samples  
Censored Weibull regression model  
Check if a matrix is Lower or Upper triangular  
Check whether a square matrix is skew-symmetric  
Circular correlations between two circular variables  
Column and row-wise jackknife sample means  
Column-wise means and variances  
Column-wise MLE of some univariate distributions  
Column-wise MLE of the angular Gaussian distribution  
Column-wise pooled variances across groups  
Column-wise summary statistics with grouping variables  
Conditional least-squares estimate for Poisson INAR(1) models  
Constrained least squares  
Contour plots of some bivariate distributions  
Correlation significance testing using Fisher’s z-transformation  
Covariance between a variable and a matrix of variables  
Cross-validation for the k-NN algorithm for really large scale data  
Cross-validation for the multinomial regression  
Cross-validation for the naive Bayes classifiers  
Cross-validation for the regularised maximum likelihood linear discriminant analysis  
Diagonal values of the Hat matrix  
Distance between two covariance matrices  
Distance correlation matrix  
Empirical entropy  
Fisher’s linear discriminant analysis  
Fixed effects regression  
Fixed intercepts Poisson regression  
Forward Backward Early Dropping selection regression  
Fractional polynomial regression with one independent variable  
Gamma regression with a log-link  
GEE Gaussian regression  
Gumbel regression  
Hellinger distance based regression for count data  
Hellinger distance based univariate regression for proportions  
Heteroscedastic linear models for large scale data  
Hurdle-Poisson regression  
Hypothesis tests for equality of a covariance matrix  
Hypothesis tests for equality of multiple covariance matrices  
Intersect  
Item difficulty and discrimination  
Jackknife sample mean  
Kaplan-Meier estimate of a survival function  
Linear regression with clustered data  
Mahalanobis depth  
Many 2 sample student’s t-tests  
Many approximate simple logistic regressions  
Many binary classification metrics
Score test for overdispersion in Poisson regression ........................................ 123
Single terms deletion hypothesis testing in a linear regression model .................. 124
Skeleton of the FEDHC algorithm ....................................................................... 126
Skeleton of the MMHC algorithm ........................................................................ 127
Split the matrix in lower, upper triangular and diagonal ...................................... 129
The k-NN algorithm for really large scale data ..................................................... 130
Tobit regression ...................................................................................................... 131
Trimmed mean ....................................................................................................... 133
Variable selection using the PC-simple algorithm ............................................... 134
Wald confidence interval for the ratio of two Poisson variables .............................. 135
Walter’s confidence interval for the ratio of two binomial variables (and the relative risk) 136
Zero inflated Gamma regression .......................................................................... 137
Zero truncated Poisson regression ....................................................................... 139

Index ....................................................................................................................... 141

Rfast2-package

Description

A collection of Rfast2 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

Package: Rfast2
Type: Package
Version: 0.1.3
Date: 2022-03-22
License: GPL-2

Maintainers

Manos Papadakis <rfastofficial@gmail.com>

Author(s)

Manos Papadakis <papadakm95@gmail.com>, Michail Tsagris <mtsagris@uoc.gr>, Stefanos Fafalios <stefanosfafalios@gmail.com>, Marios Dimitriadis <kmdimitriadis@gmail.com>.
Add many single terms to a model

Description
Add many single terms to a model.

Usage
add.term(y, xinc, xout, devi_0, type = "logistic", logged = FALSE, to1 = 1e-07, maxiters = 100, parallel = FALSE)

Arguments
y The response variable. It must be a numerical vector.
xinc The already included independent variable(s).
xout The independent variables whose conditional association with the response is to be calculated.
devi_0 The deviance for Poisson, logistic, qpoisson, qlogistic and normlog regression or the log-likelihood for the Weibull, spml and multinomial regressions. See the example to understand better.
type The type of regression, "poisson", "logistic", "qpoisson" (quasi Poisson), "qlogistic" (quasi logistic) "normlog" (Gaussian regression with log-link) "weibull", "spml" and "multinom".
logged Should the logarithm of the p-value be returned? TRUE or FALSE.
tol The tolerance value to terminate the Newton-Raphson algorithm when fitting the regression models.
maxiters The maximum number of iterations the Newton-Raphson algorithm will perform.
parallel Should the computations take place in parallel? TRUE or FALSE.

Details
The function is similar to the built-in function add1. You have already fitted a regression model with some independent variables (xinc). You then add each of the xout variables and test their significance.

Value
A matrix with two columns. The test statistic and its associated (logged) p-value.

Author(s)
Stefanos Fafalios.
R implementation and documentation: Stefanos Fafalios <stefanosfafalios@gmail.com>.
Angular Gaussian random values simulation

References


See Also

bic.regs, logiquant.regs, sp.logiregs

Examples

```r
x <- matrix( rnorm(200 * 10), ncol = 10)
y <- rpois(200, 10)
devi_0 <- deviance( glm(y ~ x[, 1:2], poisson) )
a <- add.term(y, xinc = x[,1:2], xout = x[, 3:10], devi_0 = devi_0, type= "poisson")

y <- rbinom(200, 1, 0.5)
devi_0 <- deviance( glm(y ~ x[, 1:2], binomial) )
a <- add.term(y, xinc = x[,1:2], xout = x[, 3:10], devi_0 = devi_0, type= "logistic")

y <- rbinom(200, 2, 0.5)
devi_0 <- Rfast::multinom.reg(y, x[, 1:2])$loglik
a <- add.term(y, xinc = x[,1:2], xout = x[, 3:10], devi_0 = devi_0, type= "multinom")

y <- rgamma(200, 3, 1)
devi_0 <- Rfast::weib.reg(y, x[, 1:2])$loglik
a <- add.term(y, xinc = x[,1:2], xout = x[, 3:10], devi_0 = devi_0, type= "weibull")
```

Description

Angular Gaussian random values simulation.

Usage

```r
riag(n, mu)
```

Arguments

- **n**: The sample size, a numerical value.
- **mu**: The mean vector in \( \mathbb{R}^d \).
Details

The algorithm uses univariate normal random values and with some mean. The vectors are then scaled to have unit length.

Value

A matrix with the simulated data.

Author(s)

Michail Tsagris.

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

References


See Also

colspml.mle,circ.cor1,circ.cors1

Examples

x <- riag(20, rnorm(4, 3, 1))
Anova for circular data

Arguments

u
A numeric vector containing the data that are expressed in rads.

ina
A numerical or factor variable indicating the group of each value.

Details

The high concentration (hcf.circaov), log-likelihood ratio (lr.circaov), embedding approach (embed.circaov) or the non equal concentration parameters approach (het.circaov) is used.

Value

A vector including:

- test
  The value of the test statistic.

- p-value
  The p-value of the test.

- kappa
  The concentration parameter based on all the data. If the het.circaov is used this argument is not returned.

Author(s)

Michail Tsagris.

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

References


See Also

multivm.mle,vm.nb

Examples

```R
x <- rnorm(60, 2.3, 0.3)
ina <- rep(1:3,each = 20)
hcf.circaov(x, ina)
lr.circaov(x, ina)
het.circaov(x, ina)
embed.circaov(x, ina)
```
Description

backward selection with the F test or the partial correlation coefficient.

Usage

```
lm.bsreg(y, x, alpha = 0.05, type = "F")
```

Arguments

- **y**: The dependent variable, a numerical vector with numbers.
- **x**: A numerical matrix with the independent variables. We add, internally, the first column of ones.
- **alpha**: If you want to perform the usual F (or t) test set this equal to "F". For the test based on the partial correlation set this equal to "cor".
- **type**: The type of backward selection to be used, "F" stands for F-test, where "cor" stands for partial correlation.

Details

It performs backward selection with the F test or the partial correlation coefficient. For the linear regression model, the Wald test is equivalent to the partial F test. So, instead of performing many regression models with single term deletions we perform one regression model with all variables and compute their Wald test effectively. Note, that this is true, only if the design matrix "x" contains the vectors of ones and in our case this must be, strictly, the first column. The second option is to compute the p-value of the partial correlation.

Value

A matrix with two columns. The removed variables and their associated p-value.

Author(s)

Michail Tsagris.

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

References

See Also

`lm.drop1, mmpc2, gee.reg, pc.sel`

Examples

```r
y <- rnorm(150)
x <- as.matrix(iris[, 1:4])
a <- lm(y ~., data.frame(x))
lm.bsreg(y, x)
```

Description

Benchmark - Measure time.

Usage

```r
benchmark(..., times, envir=parent.frame(), order=NULL)
## S3 method for class 'benchmark'
print(x, ...)
```

Arguments

- `...` Expressions to the benchmark function.
- `x` Object of class "benchmark" to print.
- `times` Number of time to measure execution time of the expression.
- `envir` Environment to evaluate the expressions.
- `order` An integer vector to execute the expressions with this order, otherwise the execution order is random.

Details

For measuring time we have used C++'s new library "chrono".

Value

The execution time for each expression.

Author(s)

Manos Papadakis

R implementation and documentation: Manos Papadakis <papadakm95@gmail.com>.
BIC of many simple univariate regressions

See Also

Quantile, trim.mean

Examples

bench(x <- matrix(runif(10*10),10,10),times=10)

bic.regs(y, x, family = "normal")

Arguments

y The dependent variable, a numerical vector.

x A matrix with the independent variables.

family The family of the regression models. "normal", "binomial", "poisson", "multinomial", "normlog" (Gaussian regression with log link), "spmpl" (SPML regression) or "weibull" for Weibull regression.

Details

Many simple univariate regressions are fitted and the BIC of every model is computed.

Value

A vector with the BIC of each regression model.

Author(s)

Michail Tsagris.

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

See Also

logistic_only, poisson_only
Examples

```r
y <- rbinom(100, 1, 0.6)
x <- matrix(rnorm(100 * 50), ncol = 50)
bic.regs(y, x, "binomial")
```

Description

Binomial regression.

Usage

```r
binom.reg(y, ni, x, full = FALSE, tol = 1e-07, maxiters = 100)
```

Arguments

- **y**: The dependent variable; a numerical vector with integer values, 0, 1, 2,... The successes.
- **ni**: A vector with integer values, greater than or equal to y. The trials.
- **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).
- **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 difference from logistic regression is that in the binomial regression the binomial distribution is used and not the Bernoulli.

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.
Bootstrap James and Hotelling test for 2 independent sample mean vectors

Author(s)

Michail Tsagris.

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

References


See Also

negbin.reg, hp.reg, ztp.reg

Examples

x <- matrix(rnorm(100 * 2), ncol = 2)
y <- rbinom(100, 20, 0.5)  # binary logistic regression
ni <- rep(20, 100)
a <- binom.reg(y, ni, x, full = TRUE)
x <- NULL

Description

Bootstrap James and Hotelling test for 2 independent sample mean vectors.

Usage

boot.james(y1, y2, R = 999)
boot.hotel2(y1, y2, R = 999)

Arguments

y1  A numerical matrix with the data of the one sample.
y2  A numerical matrix with the data of the other sample.
R   The number of bootstrap samples to use.
Bootstrap Student’s t-test for 2 independent samples

Details

We bootstrap the 2-samples James (does not assume equal covariance matrices) and Hotelling test (assumes equal covariance matrices). The difference is that the Hotelling test statistic assumes equality of the covariance matrices, which if violated leads to inflated type I errors. Bootstrap calibration though takes care of this issue. As for the bootstrap calibration, 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 (Chatzipantsiou et al., 2019).

Value

The bootstrap p-value.

Author(s)

Michail Tsagris.

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

References


See Also

welch.tests,trim.mean

Examples

```
boot.james( as.matrix(iris[1:25, 1:4]), as.matrix(iris[26:50, 1:4]) )
```
Bootstrap Student’s t-test for 2 independent samples

Usage

boot.student2(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

We bootstrap Student’s (Gosset’s) t-test statistic and not the Welch t-test statistic. For the latter case see the "boot.ttest2" function in Rfast. The difference is that Gosset’s test statistic assumes equality of the variances, which if violated leads to inflated type I errors. Bootstrap calibration though takes care of this issue. As for the bootstrap calibration, 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 (Chatzipantsiou et al., 2019).

Value

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

Author(s)

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

References


See Also

welch.tests, trim.mean

Examples

```r
x <- rexp(40, 4)
y <- rbeta(50, 2.5, 7.5)
system.time(t.test(x, y, var.equal = TRUE) )
system.time( a <- boot.student2(x, y, 9999) )
a
```
Censored Weibull regression model

Description

Censored Weibull regression model.

Usage

censweib.reg(y, x, di, 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).

di
A vector with 1s and 0s indicating the censored value. The value of 1 means uncensored value, whereas the value of 0 means censored value.

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.

Author(s)

Michail Tsagris and Stefanos Fafalios.

R implementation and documentation: Michail Tsagris <mtsagris@uoc.gr> and Stefanos Fafalios <stefanosfafalios@gmail.com>.
Check if a matrix is Lower or Upper triangular

References

See Also
censweibull.mle, km, gumbel.reg

Examples
## Not run:
x <- matrix(rnorm(100 * 2), ncol = 2)
y <- rexp(100, 1)
di <- rbinom(100, 1, 0.8)
mod <- censweib.reg(y, x, di)
x <- NULL
## End(Not run)

Description
Lower/upper triangular matrix.

Usage
is.lower.tri(x, diag = FALSE)
is.upper.tri(x, diag = FALSE)

Arguments
x A matrix with data.
diag A logical value include the diagonal to the result.

Value
Check if a matrix is lower or upper triangular. You can also include diagonal to the check.

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

Intersect

Examples

```r
x <- matrix(runif(10*10),10,10)
is.lower.tri(x)
is.lower.tri(x,TRUE)

is.upper.tri(x)
is.upper.tri(x,TRUE)
```

Description

Check whether a square matrix is skew-symmetric.

Usage

```r
is.skew.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

```r
x <- matrix(rnorm(100 * 400), ncol = 400)
s1 <- cor(x)
is.skew.symmetric(s1)
x <- x[1:100, ]
is.skew.symmetric(x)
```

Description

Circular correlations between two circular variables.

Usage

- `circ.cor1(theta, phi, pvalue = FALSE)`
- `circ.cors1(theta, phi, pvalue = FALSE)`

Arguments

- `theta`  
  The first circular variable expressed in radians, not degrees.
- `phi`  
  The other circular variable. In the case of `circ.cors1` this is a matrix with many circular variables. In either case, the values must be in radians, not degrees.
- `pvalue`  
  If you want the p-value of the zero correlation hypothesis testing set this to TRUE, otherwise leave it FALSE.

Details

Correlation for circular variables using the cosine and sine formula of Jammaladaka and Sen-Gupta (1988).

Value

If you set `pvalue` = TRUE, then for the "circ.cor1" a vector with two values, the correlation and its associated p-value, otherwise the correlation only. For the "circ.cors1", either a vector with the correlations only or a matrix with two columns, the correlation and the p-values.

Author(s)

Michail Tsagris.

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


See Also

spml.reg

Examples

```r
y <- runif(50, 0, 2 * pi)
x <- runif(50, 0, 2 * pi)
circ.cor!(y, x, TRUE)
x <- matrix(runif(50 * 10, 0, 2 * pi), ncol = 10)
circ.cors!(y, x, TRUE)
```

Description

Column and row-wise jackknife sample means.

Usage

```r
coljack.means(x)
rowjack.means(x)
```

Arguments

- `x`: A numerical matrix with data.

Details

An efficient implementation of the jackknife mean is provided.

Value

A vector with the jackknife sample means.

Author(s)

Michail Tsagris.
R implementation and documentation: Michail Tsagris <mtsagris@uoc.gr>.
Column-wise means and variances

References


See Also

welch.tests, trim.mean

Examples

```r
x <- as.matrix(iris[1:50, 1:4])
coljack.means(x)
```

Description

Column-wise means and variances of a matrix.

Usage

```r
colmeansvars(x, std = FALSE, parallel = FALSE)
```

Arguments

- `x`: A matrix with the data.
- `std`: A boolean variable specifying whether you want the variances (FALSE) or the standard deviations (TRUE) of each column.
- `parallel`: A boolean value for parallel version.

Details

This function calculates the column-wise means and variances (or standard deviations).

Value

A matrix with two rows. The first contains the means and the second contains the variances (or standard deviations).

Author(s)

Michail Tsagris.

R implementation and documentation: Michail Tsagris <mtsagris@uoc.gr> and Manos Papadakis <papadakm95@gmail.com>.
Column-wise MLE of some univariate distributions

See Also

pooled.colVars

Examples

colmeansvars(as.matrix(iris[,1:4]))

Description

Column-wise MLE of some univariate distributions.

Usage

collognorm.mle(x)
collogitnorm.mle(x)
colborel.mle(x)
colhalfnorm.mle(x)
colordinal.mle(x, link = "logit")
colcauchy.mle(x, tol = 1e-07, maxiters = 100, parallel = FALSE)
colbeta.mle(x, tol = 1e-07, maxiters = 100, parallel = FALSE)
colunitweibull.mle(x, tol = 1e-07, maxiters = 100, parallel = FALSE)

Arguments

x A numerical matrix with data. Each column refers to a different vector of observations of the same distribution. The values of for Lognormal must be greater than zero, for the logitnormal they must by percentages, excluding 0 and 1, whereas for the Borel distribution the x must contain integer values greater than 1. For the halfnormal the numbers must be strictly positive, while for the ordinal this can be a numerical matrix with values 1, 2, 3,..., not zeros.

link This can either be "logit" or "probit". It is the link function to be used.

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 parameters and log-likelihood are computed.
Value
A matrix with two or three columns. The first one or the first two contain the parameter(s) of the distribution and the second or third column the relevant log-likelihood. For the ordinal a list including:

- `param`: A matrix with the intercepts (threshold coefficients) of the model applied to each column (or variable).
- `loglik`: 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
censpois.mle, gammapois.mle

Examples
```r
x <- matrix( exp( rnorm(1000 * 50) ), ncol = 50)
a <- colspml.mle(x)
x <- NULL
```

Description
Column-wise MLE of the angular Gaussian distribution.

Usage
```
colspml.mle(x, tol = 1e-07, maxiters = 100, parallel = FALSE)
```
Arguments

x A numerical matrix with data. Each column refers to a different vector of observations of the same distribution. The values of for Lognormal must be greater than zero, for the logitnormal they must by percentages, excluding 0 and 1, whereas for the Borel distribution the x must contain integer values greater than 1.

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

maxiters The maximum number of iterations that can take place in each regression.

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.

Details

For each column, spml.mle function is applied that fits the angular Gaussian distribution estimates its parameters and computes the maximum log-likelihood.

Value

A matrix with four columns. The first two are the mean vector, then the $\gamma$ parameter, and the fourth column contains maximum log-likelihood.

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

collognorm.mle, gammapois.mle

Examples

x <- matrix( runif(100 * 10), ncol = 10)
a <- colspml.mle(x)
x <- NULL
Description

Column-wise pooled variances across groups.

Usage

pooled.colVars(x, ina, std = FALSE)

Arguments

x A matrix with the data.
ina A numerical vector specifying the groups. If you have numerical values, do not put zeros, but 1, 2, 3 and so on.
std A boolean variable specifying whether you want the variances (FALSE) or the standard deviations (TRUE) of each column.

Details

This function calculates the pooled variance (or standard deviation) for a range of groups for each column.

Value

A vector with the pooled column variances or standard deviations.

Author(s)

Michail Tsagris.

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

See Also

colmeansvars

Examples

pooled.colVars( as.matrix(iris[, 1:4]), as.numeric(iris[, 5]) )
Description

Column-wise summary statistics with grouping variables.

Usage

```r
colGroup(x, ina, method = "sum", names = TRUE, std = FALSE)
```

Arguments

- `x`: A matrix with data.
- `ina`: A numerical vector specifying the groups. If you have numerical values, do not put zeros, but 1, 2, 3 and so on. **The numbers must be consecutive**, like 1, 2, 3,.. Do not put 1, 3, 4 as this will cause C++ to crash.
- `method`: One of the: "sum", "min", "max", "median", "var".
- `names`: Set the name of the result vector with the unique numbers of group variable.
- `std`: A boolean variable specifying whether you want the variances (FALSE) or the standard deviations (TRUE) of each column. This is taken into account only when `method = "var"`.

Value

Column wise of grouping variables. You can also include diagonal to the check.

Author(s)

Manos Papadakis.

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

See Also

`Quantile, colQuantile, rowQuantile`

Examples

```r
x <- matrix(runif(100 * 5), 100, 5)
id <- sample(1:3, 100, TRUE)

all.equal( colGroup(x, id), rowsum(x, id) )
```
Description
Conditional least-squares estimate for Poisson INAR(1) models.

Usage

\begin{verbatim}
  pinar1(x, unbiased = FALSE)
colpinar1(x, unbiased = FALSE)
\end{verbatim}

Arguments

\begin{itemize}
  \item \texttt{x} \\
  \hspace{1cm} Either a numerical vector or a matrix, depending on the function.
  \item \texttt{unbiased} \\
  \hspace{1cm} If you want the unbiased estimation select \texttt{TRUE}.
\end{itemize}

Details

The function computes the constant and slope coefficients of the Poisson Integer Autoregressive of order 1 (Poisson INAR(1)) model using the conditional least-squares method.

Value

For \texttt{pinar1()} a vector with two values, the \(\lambda\) coefficient (constant) and the \(\alpha\) coefficient (slope). See references for more information.

For the \texttt{colpinar1()} a matrix with two columns, the \(\lambda\) coefficient (constant) and the \(\alpha\) coefficient (slope) for each variable (column of \texttt{x}).

Author(s)

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

References


See Also

\begin{verbatim}
  fipois.reg, hp.reg
\end{verbatim}

Examples

\begin{verbatim}
  x <- rpois(200, 10)
pinar1(x)
\end{verbatim}
Constrained least squares

Description

Constrained least squares.

Usage

\texttt{cls(y, x, R, ca)}

Arguments

\begin{itemize}
  \item \texttt{y} \hspace{1cm} The response variables, a numerical vector with observations.
  \item \texttt{x} \hspace{1cm} A matrix with independent variables, the design matrix.
  \item \texttt{R} \hspace{1cm} The R vector that contains the values that will multiply the beta coefficients. See details and examples.
  \item \texttt{ca} \hspace{1cm} The value of the constraint, $R^T \beta = c$. See details and examples.
\end{itemize}

Details

This is described in Chapter 8.2 of Hansen (2019). The idea is to inimise the sum of squares of the residuals under the constraint $R^T \beta = c$. As mentioned above, be careful with the input you give in the x matrix and the R vector.

Value

A list including:

\begin{itemize}
  \item \texttt{bols} \hspace{1cm} The OLS (Ordinary Least Squares) beta coefficients.
  \item \texttt{bcls} \hspace{1cm} The CLS (Constrained Least Squares) beta coefficients.
\end{itemize}

Author(s)

Michail Tsagris.

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

References


See Also

gee.reg, bic.reg, ztp.reg
Examples

```r
x <- as.matrix(iris[1:50, 1:4])
y <- rnorm(50)
R <- c(1, 1, 1, 1)
cls(y, x, R, 1)
```

Description

Contour plots of some bivariate distributions.

Usage

```r
den.contours(x, type = "normal", v = 5)
```

Arguments

- `x`: A matrix with two columns containing the data.
- `type`: The distribution whose contours will appear. This can be "normal", "t" or "ml-norm", standing for the bivariate normal, t and bivariate log-normal.
- `v`: The degrees of freedom of the bivariate t distribtuion.

Value

The contour plot.

Author(s)

Michail Tsagris.

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

See Also

`collognorm.mle`, `halfcauchy.mle`

Examples

```r
x <- as.matrix(iris[, 1:2])
den.contours(x)
```
Correlation significance testing using Fisher's z-transformation

Description

Correlation significance testing using Fisher's z-transformation.

Usage

\texttt{cor\_test(y, x, type = "pearson", rho = 0, a = 0.05 )}

Arguments

- \texttt{y} A numerical vector.
- \texttt{x} A numerical vector.
- \texttt{type} The type of correlation you want. "pearson" and "spearman" are the two supported types because their standard error is easily calculated.
- \texttt{rho} The value of the hypothesised correlation to be used in the hypothesis testing.
- \texttt{a} The significance level used for the confidence intervals.

Details

The function uses the built-in function "cor" which is very fast, then computes a confidence interval and produces a p-value for the hypothesis test.

Value

A vector with 5 numbers; 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.

Author(s)

Michail Tsagris.

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

See Also

allbetas, univglm

Examples

\begin{verbatim}
  x <- rcauchy(60)
  y <- rnorm(60)
  cor_test(y, x)
\end{verbatim}
Covariance between a variable and a matrix of variables

Description
Covariance between a variable and a matrix of variables.

Usage
\texttt{covar(y, x)}

Arguments
\begin{itemize}
\item \texttt{y} \hspace{1cm} A numerical vector.
\item \texttt{x} \hspace{1cm} A numerical matrix.
\end{itemize}

Details
The function calculates the covariance between a variable and many others.

Value
A vector with the covariances.

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

See Also
\texttt{circ.cors1, bic.regs}

Examples
\begin{verbatim}
y <- rnorm(40)
x <- matrix( rnorm(40 * 10), ncol = 10 )
covar(y, x)
cov(y, x)
\end{verbatim}
Cross-validation for the k-NN algorithm for really large scale data

Description

Cross-validation for the k-NN algorithm for really large scale data.

Usage

```r
bigknn.cv(y, x, k = 5:10, type = "C", folds = NULL, nfolds = 10,
          stratified = TRUE, seed = 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.
- `type`: If your response variable `y` is numerical data, then this should be "R" (regression). If `y` is in general categorical set this argument to "C" (classification).
- `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.
- `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.

This function allows for the Euclidean distance only.
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>.

References


See Also

`big.knn,regmlelda.cv,multinomreg.cv`

Examples

```r
x <- as.matrix(iris[, 1:4])
mod <- bigknn.cv(y = iris[, 5], x = x, k = c(3, 4) )
```

Description

Cross-validation for the multinomial regression.

Usage

```r
multinomreg.cv(y, x, folds = NULL, nfolds = 10, stratified = TRUE, seed = FALSE, pred.ret = FALSE)
```
Arguments

- **y**: The response variable. A numerical or a factor type vector.
- **x**: A matrix or a data.frame with the predictor variables.
- **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.
- **pred.ret**: If you want the predicted values returned set this to TRUE.

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.

Author(s)

Michail Tsagris.

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

References


See Also

- bigknn.cv, mle.lda, reg.mle.lda

Examples

```r
x <- as.matrix(iris[, 1:2])
mod <- multinomreg(cv(iris[, 5], x)
```
Cross-validation for the naive Bayes classifiers

Description

Cross-validation for the naive Bayes classifiers.

Usage

```r
nb.cv(x, ina, type = "gaussian", folds = NULL, nfolds = 10,
      stratified = TRUE, seed = FALSE, pred.ret = FALSE)
```

Arguments

- `x`: A matrix with the available data, the predictor variables.
- `ina`: A vector of data. The response variable, which is categorical (factor is acceptable).
- `type`: The type of naive Bayes, "gaussian", "gamma", "weibull", "normlog", "laplace", "cauchy", "logitnorm", "beta", "vm" or "spml", "poisson", "multinom", "geom" or "bernoulli".
- `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.
- `seed`: If you set this to TRUE, the same folds will be created every time.
- `pred.ret`: If you want the predicted values returned set this to TRUE.

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.

Author(s)

Michail Tsagris.

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


See Also

weibullnb.pred, weibull.nb, vm.nb, vmnb.pred, mle.lda, reg.mle.lda, multinom.reg

Examples

x <- as.matrix(iris[, 1:4])
mod <- nb.cv(ina = iris[, 5], x = x )

Description

Cross-validation for the regularised maximum likelihood linear discriminant analysis.

Usage

regmlelda.cv(x, ina, lambda = seq(0, 1, by = 0.1), folds = NULL, nfolds = 10,
            stratified = TRUE, seed = FALSE, pred.ret = FALSE)

Arguments

x A matrix with numerical data.
ina A numerical vector or factor with consecutive numbers indicating the group to
which each observation belongs to.
lambda A vector of regularization values \( \lambda \) such as \( (0, 0.1, 0.2,\ldots) \).
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.
pred.ret If you want the predicted values returned set this to TRUE.
Details

Cross-validation for the regularised maximum likelihood linear discriminant analysis is performed. The function is not extremely fast, yet is pretty fast.

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>`.

References


See Also

- `reg.mle.lda`, `bigknn.cv`, `mle.lda`, `big.knn`, `weibull.nb`

Examples

```r
x <- as.matrix(iris[, 1:4])
mod <- regmlelda.cv(x, iris[, 5])
```

---

Diagonal values of the Hat matrix

Description

Diagonal values of the Hat matrix.

Usage

`leverage(x)`
Distance between two covariance matrices

Arguments

x A matrix with independent variables, the design matrix.

Details

The function returns the diagonal values of the Hat matrix used in linear regression. We did not call it "hatvalues" as R contains a built-in function with such a name.

Value

A vector with the diagonal Hat matrix values, the leverage of each observation.

Author(s)

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

References


See Also

ggee.reg, bic.regs, ztp.reg

Examples

x <- as.matrix( iris[1:50, 1:4] )
a <- leverage(x)

distance between two covariance matrices

distance between two covariance matrices

Description

Distance between two covariance matrices.

Usage

covdist(s1, s2)

Arguments

s1 The first covariance matrix.
s2 The second covariance matrix.
Details

A metric for covariance matrices is the title of a paper by Forstner and Moonen (2003). The metric is computed for two non-singular covariance matrices.

Value

The distance between the two covariance matrices.

Author(s)

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

References


See Also
covlikel, covequal, covar, cor_test

Examples

s1 <- cov(iris[1:50, 1:4])
s2 <- cov(iris[51:100, 1:4])
covdist(s1, s2)

dcov <- dcora(s1)
dcov

distance correlation matrix

distance correlation matrix

Description

Distance correlation matrix.

Usage
dcora(x)

Arguments

x A numerical matrix.

Details

The distance correlation matrix is computed.
Empirical entropy

Value
A matrix with the pairwise distance correlations between all variables in x.

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

References

See Also
cor_test, covar

Examples
x <- as.matrix( iris[1:50, 1:4] )
res <- dcora(x)

Description
Empirical entropy.

Usage
empirical.entropy(x, k = NULL, pretty = FALSE)

Arguments
x A numerical vector with continuous values.
k If you want to cut the data into a specific range plug it here, otherwise this decide based upon the Freedman-Diaconis’ rule.
pretty Should the breaks be equally space upon the range of x? If yes, let this FALSE. If this is TRUE, the breaks are decided using the base command pretty.

Details
The function computes the empirical entropy.

Value
The estimated empirical entropy.
Fisher’s linear discriminant analysis

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

References
https://en.wikipedia.org/wiki/Histogram

See Also
Quantile, pretty

Examples

```r
x <- rnorm(100)
empirical.entropy(x)
empirical.entropy(x, pretty = TRUE)
```

Description
Fisher’s linear discriminant analysis.

Usage

```r
fisher.da(xnew, x, ina)
```

Arguments

- `xnew`: A numerical vector or a matrix with the new observations, continuous data.
- `x`: A matrix with numerical data.
- `ina`: A numerical vector or factor with consecutive numbers indicating the group to which each observation belongs to.

Details
Maximum likelihood linear discriminant analysis is performed.

Value
A vector with the predicted group of each observation in "xnew".
Author(s)

Michail Tsagris.

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

References


See Also

mle.lda, reg.mle.lda, big.knn, weibull.nb

Examples

x <- as.matrix(iris[, 1:4])
ina <- iris[, 5]
a <- fisher.da(x, x, ina)

---

Description

Fixed effects regression.

Usage

fe.lmfit(y, x, id)

Arguments

y A numerical vector or a numerical matrix.
x A numerical matrix with the predictor variables.
id A vector with the subject ids. This can be factor or a numerical.

Details

The function performs fixed effects regression (within estimator) for panel (longitudinal) data. It can also handle unbalanced designs. A main difference from the package "plm" is that it returns much fewer information, but much faster.
Fixed intercepts Poisson regression

Value

A list including:

- **be**: The beta coefficients.
- **fe**: The fixed effect deviations.
- **residuals**: The residuals of the linear model(s).

Author(s)

Michail Tsagris.

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

References

https://www.econometrics-with-r.org/10-rwpd.html

See Also

- `cluster.lm`, `gee.reg`, `fipois.reg`

Examples

```r
y <- rnorm(100)
x <- rnorm(100)
id <- rep(1:10, 10)
mod <- fe.lmfit(y, x, id)
```

Description

Fixed intercepts Poisson regression.

Usage

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

Arguments

- **y**: The dependent variable, a numerical vector with integer, non negative valued data.
- **x**: A matrix with 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 $10^{-7}$ by default.
- **maxiters**: The maximum number of iterations that can take place during the fitting.
Fixed intercepts Poisson regression

Details

Fixed intercepts Poisson regression for clustered count data is fitted. According to Demidenko (2013), when the number of clusters (N) is small and the number of observations per cluster ($n_i$) is relatively large, say $\text{min}(n_i) > N$, one may assume that the intercept $\alpha_i = \beta + u_i$ is fixed and unknown ($i = 1, ..., N$).

Value

A list including:

- **be** The regression coefficients.
- **seb** The standard errors of the regression coefficients.
- **ai** The estimated fixed intercepts for each cluster of observations.
- **covbeta** The covariance matrix of the regression coefficients.
- **loglik** The maximised log-likelihood value.
- **iters** The number of iteration the Newton-Raphson required.

Author(s)

Michail Tsagris.

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

References


See Also

cluster.lm, fe.lmfit, gee.reg, covar, welch.tests

Examples

```r
y <- rpois(200, 10)
id <- sample(1:10, 200, replace = TRUE)
x <- rpois(200, 10)
fipois.reg(y, x, id)
```
Description

Forward Backward Early Dropping selection regression.

Usage

```r
fbed.reg(y, x, alpha = 0.05, type = "logistic", K = 0, backward = FALSE,
        parallel = FALSE, tol = 1e-07, maxiters = 100)
```

Arguments

- `y`: The response variable, a numeric vector.
- `x`: A matrix with continuous variables.
- `alpha`: The significance threshold value for assessing p-values. Default value is 0.05.
- `type`: The available types are: "logistic" (binary logistic regression), "qlogistic" (quasi logistic regression, for binary value or proportions including 0 and 1), "poisson" (Poisson regression), "qpoisson" (quasi Poisson regression), "weibull" (Weibull regression) and "spml" (SPML regression).
- `K`: How many times should the process be repeated? The default value is 0.
- `backward`: After the Forward Early Dropping phase, the algorithm proceeds with the usual Backward Selection phase. The default value is set to TRUE. It is advised to perform this step as maybe some variables are false positives, they were wrongly selected. This is rather experimental now and there could be some mistakes in the indices of the selected variables. Do not use it for now.
- `parallel`: If you want the algorithm to run in parallel set this TRUE.
- `tol`: The tolerance value to terminate the Newton-Raphson algorithm.
- `maxiters`: The maximum number of iterations Newton-Raphson will perform.

Details

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

The "qlogistic" and "qpoisson" proceed with the Wald test and no backward is performed, while for all the other regression types, the log-likelihood ratio test is used and backward phase is available.
Fractional polynomial regression with one independent variable

Value

If \( K \) is a single number a list including:

Note, that the "gam" argument must be the same though.

- **res**: A matrix with the selected variables and their test statistic.
- **info**: A matrix with the number of variables and the number of tests performed (or models fitted) at each round (value of \( K \)). This refers to the forward phase only.
- **runtime**: The runtime required.

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

logiquant.regs, bic.regs, gee.reg

Examples

```r
# simulate a dataset with continuous data
x <- matrix( runif(100 * 50, 1, 100), ncol = 50 )
y <- rnbinom(100, 10, 0.5)
a <- fbed.reg(y, x, type = "poisson")
```

---

Fractional polynomial regression with one independent variable

Fractional polynomial regression with one independent variable.

Description

Fractional polynomial regression with one independent variable.

Usage

```r
fp(y, x, aa, di = NULL, type = "normal", full = FALSE, seb = FALSE,
tol = 1e-07, maxiters = 100)
```
Arguments

- **y**: The dependent variable, a numerical vector.
- **x**: A vector, the independent variable.
- **aa**: A vector with two values indicating the range of the optimal value of $\alpha$ to search within.
- **di**: This is valid only for the Weibull regression. A vector with 1s and 0s indicating the censored value. The value of 1 means uncensored value, whereas the value of 0 means censored value.
- **type**: The type of regression model: "normal", "logistic", "poisson", "spml" (SPML regression), "gamma", "normlog", "weibull", "negbin".
- **full**: If this is FALSE, the coefficients and the deviance will be returned only. If this is TRUE, more information is returned.
- **seb**: Do you want the standard error of the estimates to be returned? TRUE or FALSE.
- **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 independent variable is power transformed and this function searches for the optimal power.

Value

A list including:

- **a**: The power that yields the optimal fit.
- **mod**: The model with the independent variable power transformed.

Author(s)

Michail Tsagris.

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

References


See Also

logistic_only, poisson_only

Examples

```r
y <- rnorm(100)
x <- abs( rnorm(100) )
mod <- fp(y, x, c(-2, 2) )
```
Gamma regression with a log-link

Description

Gamma regression with a log-link.

Usage

gammareg(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

The gammareg fits a Gamma regression with a log-link. The gamma.con fits a Gamma regression with a log link with the intercept only (glm(y ~ 1, Gamma(log)))

Value

A list including:

- iters: The number of iterations required by the newton-Raphson.
- 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).

Author(s)

Stefanos Fafalios and Michail Tsagris.

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

References

GEE Gaussian regression

See Also

gammaregs, zigamma.mle

Examples

## Not run:
y <- rgamma(100, 3, 4)
x <- matrix(rnorm(100 * 2), ncol = 2)
m1 <- glm(y ~ x, family = Gamma(log))
m2 <- gammareg(y, x)

## End(Not run)

Description

GEE Gaussian regression.

Usage

ggee.reg(y, x, id, tol = 1e-07, maxiters = 100)

Arguments

y The dependent variable, a numerical vector.
x A matrix with 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 $10^{-7}$ by default.
maxiters The maximum number of iterations that can take place during the fitting.

Details

Gaussin GEE regression is fitted.

Value

A list including:

be The regression coefficients.
seb The standard errors of the regression coefficients.
phi The $\phi$ parameter.
Gumbel regression

a  The $\alpha$ parameter.
covbeta  The covariance matrix of the regression coefficients.
ites  The number of iteration the Newton-Raphson required.

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

References

See Also
ccluster.lm,fe.lmfit,fipois.reg,covar,welch.tests

Examples

```r
y <- rnorm(200)
id <- sample(1:20, 200, replace = TRUE)
x <- rnorm(200, 3)
gee.reg(y, x, id)
```

Gumbel regression

Description
Gumbel regression.

Usage
gumbel.reg(y, x, tol = 1e-07, maxiters = 100)

Arguments

- **y**: The dependent variable, a numerical vector with real valued numbers.
- **x**: A matrix or a data.frame with the indendent variables.
- **tol**: The tolerance value required by the Newton-Raphson to stop.
- **maxiters**: The maximum iterations allowed.
Hellinger distance based regression for count data

Details

A Gumbel regression model is fitted. The standard errors of the regressions are not returned as we do not compute the full Hessian matrix at each step of the Newton-Raphson.

Value

A list including:

- be: The regression coefficients.
- sigma: The scale parameter.
- loglik: The loglikelihood of the regression model.
- iters: The iterations required by the Newton-Raphson.

Author(s)

Michail Tsagris.

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

See Also

negbin.reg, ztp.reg

Examples

```r
y <- rnorm(100)
x <- matrix(rnorm(100 * 3), ncol = 3)
mod <- gumbel.reg(y, x)
```

---

Hellinger distance based regression for count data

---

Description

Hellinger distance based regression for count data.

Usage

```r
hellinger.countreg(y, x, tol = 1e-07, maxiters = 100)
```
Arguments

\( y \)

The dependent variable, a numerical vector with integer valued data, counts.

\( x \)

A numerical matrix with the independent variables. We add, internally, the first column of ones.

\( \text{tol} \)

The tolerance value to terminate the Newton-Raphson algorithm.

\( \text{maxiters} \)

The max number of iterations that can take place in each regression.

Details

We minimise the Hellinger distance instead of the ordinarily used divergence, the Kullback-Leibler. Both of them fall under the \( \phi \)-divergence class models and hence this one produces asymptotically normal regression coefficients as well.

Value

A list including:

\( \text{be} \)

The regression coefficients.

\( \text{seb} \)

The sandwich standard errors of the coefficients.

\( \text{covbe} \)

The sandwich covariance matrix of the regression coefficients.

\( H \)

The final Hellinger distance.

\( \text{iters} \)

The number of iterations required by Newton-Raphson.

Author(s)

Michail Tsagris.

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

See Also

\texttt{negbin.reg, gee.reg}

Examples

\begin{verbatim}
  y <- rpois(100, 10)
  x <- iris[1:100, 1]
  a <- hellinger.countreg(y, x)
\end{verbatim}
Hellinger distance based univariate regression for proportions

Description

Hellinger distance based univariate regression for proportions.

Usage

prophelling.reg(y, x, cov = FALSE, tol = 1e-07, maxiters = 100)

Arguments

y
The dependent variable, a numerical vector with percentages.

x
A numerical matrix with the independent variables. We add, internally, the first column of ones.

cov
Should the sandwich covariance matrix and the standard errors be returned? If yes, set this equal to TRUE.

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

maxiters
The max number of iterations that can take place in each regression.

Details

We minimise the Jensen-Shannon divergence instead of the ordinarily used divergence, the Kullback-Leibler. Both of them fall under the $\phi$-divergence class models and hence this one produces asymptotically normal regression coefficients as well.

Value

A list including:

be
The regression coefficients.

seb
The sandwich standard errors of the beta coefficients, if the input argument argument was set to TRUE.

covb
The sandwich covariance matrix of the beta coefficients, if the input argument argument was set to TRUE.

js
The final Jensen-Shannon divergence.

H
The final Hellinger distance.

iters
The number of iterations required by Newton-Raphson.

Author(s)

Michail Tsagris.

R implementation and documentation: Michail Tsagris <mtsagris@uoc.gr>.
Heteroscedastic linear models for large scale data

References


See Also

propols.reg, simplex.mle, kumar.mle

Examples

```r
y <- rbeta(150, 3, 4)
x <- iris
a <- prophelling.reg(y, x)
```

Heteroscedastic linear models for large scale data

Description

Heteroscedastic linear models for large scale data.

Usage

```r
het.lmfit(x, y, type = 1)
```

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.

- `type`: The type of regression to be fit in order to find the weights. The type 1 is described in Wooldridge (2012, page 287), whereas type 2 is described in page Wooldridge (2012, page 287).

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 addition is that we allow for estimation of the regression coefficients when heteroscedasticity is present.
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>.

References


See Also

- `cls`, `cluster.lm`, `lm.parboot`, `cor_test`, `lm.drop1`

Examples

```r
x <- cbind(1, matrix( rnorm( 100 * 4), ncol = 4 ) )
y <- rnorm(100)
a <- het.lmfit(x, y)
x <- NULL
```

Description

Hurdle-Poisson regression.

Usage

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

Arguments

- **y**: The dependent variable, a numerical vector with numbers.
- **x**: A numerical matrix with the independent variables. We add, internally, the first column of ones.
- **full**: If this is FALSE, the coefficients and the log-likelihood 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.
Hypothesis tests for equality of a covariance matrix

Details

Two regression models are fitted, a binary logistic regression and a zero truncated Poisson regression model.

Value

Depending on whether "full" is TRUE or not different outputs are returned. In general, the regression coefficients, the iterations required by Newton-Raphson and the deviances are returned. If full is TRUE, a matrix with their standard errors and the Wald test statistics is returned as well.

Author(s)

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

References


See Also

degbin.reg, ztp.reg

Examples

```r
y <- rpois(100, 4)
x <- iris[1:100, 1]
a <- hp.reg(y, x)
```

Hypothesis tests for equality of a covariance matrix

Hypothesis tests for equality of a covariance matrix

Description

Hypothesis tests for equality of a covariance matrix.

Usage

```r
covequal(x, sigma, a = 0.05)
```

Arguments

- `x` A numerical matrix with the data whose covariance matrix will be tested for equality.
- `sigma` The covariance matrix that is to be tested for equality.
- `a` The level of significance, default value is equal to 0.05.
Details

The likelihood-ratio test is used to test whether the sample covariance matrix from some data is equal to some pre-specified covariance matrix.

Value

A vector with the test statistic, its p-value, the degrees of freedom and the critical value of the test.

Author(s)

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

References


See Also

covlikel,covdist,covar,cor_test

Examples

x <- as.matrix(iris[1:50, 1:4])
sigma <- cov(iris[, 1:4])
covequal(x, sigma)

Description

Hypothesis tests for equality of multiple covariance matrices.

Usage

covlikel(x, ina, a = 0.05)
covmtest(x, ina, a = 0.05)

Arguments

x A numerical matrix with the data whose covariance matrices will be tested for equality.
ina A vector with the grouping variable that defines the groups.
a The level of significance, default value is equal to 0.05.
Details

The likelihood-ratio test and the Box’s M-test for testing equality of multiple covariance matrices. The log-likelihood ratio test is the multivariate generalization of Bartlett’s test of homogeneity of variances. According to Mardia (1979, pg. 140), it may be argued that if $n_i$ is small, then the log-likelihood ratio test gives too much weight to the contribution of $S$. This consideration led Box (1949) to propose his test statistic.

Value

A vector with the test statistic, its p-value, the degrees of freedom and the critical value of the test.

Author(s)

Michail Tsagris.

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

References


See Also

covequal, covdist, covar, cor_test

Examples

```r
x <- as.matrix(iris[, 1:4])
ina <- iris[, 5]
covlikel(x, ina)
```

---

**Intersect**

**Intersect Operation**

Description

Performs intersection in the same manner as R’s base package intersect works.

Usage

```r
Intersect(x, y)
```

Arguments

- `x, y` vectors containing a sequence of items, ideally of the same mode
Item difficulty and discrimination

Details
The function will discard any duplicated values in the arguments.

Value
The function will return a vector of the same mode as the arguments given. NAs will be removed.

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

See Also
intersect

Examples
x <- c(sort(sample(1:20, 9)))
y <- c(sort(sample(3, 23, 7)))
Intersect(x, y)

---

Item difficulty and discrimination

Item difficulty and discrimination

Description
Item difficulty and discrimination.

Usage
diffic(x)

discrim(x, frac = 1/3)

Arguments
x A numerical matrix with 0s (wrong answer) and 1s (correct answer).
frac A number between 0 and 1 used to calculate the difficulty of each question.

Details
The difficulty and the discrimination of each question (item) are calculated.

Value
A vector with the item difficulties or item discriminations.
**Author(s)**

Michail Tsagris.

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

**References**


**See Also**

Quantile, colmeansvars

**Examples**

```r
x <- matrix(rbinom(100 * 10, 1, 0.7), ncol = 10)
diffic(x)
discrim(x)
```

---

**Description**

Jackknife sample mean.

**Usage**

```r
jack.mean(x)
```

**Arguments**

- `x`: A numerical vector with data.

**Details**

An efficient implementation of the jackknife mean is provided.

**Value**

The jackknife sample mean.

**Author(s)**

Michail Tsagris.

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

---

**Jackknife sample mean**
Kaplan-Meier estimate of a survival function

References


See Also

welch.tests, trim.mean

Examples

x <- rnorm(50)
jack.mean(x)

Description

Kaplan-Meier estimate of a survival function.

Usage

km(ti, di)

Arguments

  ti                A numerical vector with the survival times.
  di                A numerical vector indicating the censorings. 0 = censored, 1 = not censored.

Details

  The Kaplan-Meier estimate of the survival function takes place.

Value

  A matrix with 4 columns. The non censored times, the number of subjects at risk, the number of events at each time and the estimated survival

Author(s)

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

References


See Also

sp.logiregs

Examples

```r
y <- rgamma(40, 10, 1)
di <- rbinom(40, 1, 0.6)
a <- km(y, di)
```

Description

Linear regression with clustered data.

Usage

```r
cluster.lm(y, x, id)
```

Arguments

- **y**: The dependent variable, a numerical vector with numbers.
- **x**: A matrix or a data.frame with the independent variables.
- **id**: A numerical variable with 1, 2, ... indicating the subject. Unbalanced design is of course welcome.

Details

A linear regression model for clustered data is fitted. For more information see Chapter 4.21 of Hansen (2019).

Value

A list including:

- **be**: The (beta) regression coefficients.
- **becov**: Robust covariance matrix of the regression coefficients.
- **seb**: Robust standard errors of the regression coefficients.

Author(s)

Michail Tsagris.

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

References


See Also

gge.reg,fe.lmfit

Examples

y <- rnorm(200)
id <- sample(1:20, 200, replace = TRUE)
x <- rnorm(200, 3)
ccluster.lm(y, x, id)

Description

Mahalanobis depth.

Usage

depth.mahala(x, data)

Arguments

x
  A numerical vector or matrix whose depth you want to compute.

data
  A numerical matrix used to compute the depth of x.

Details

This function computes the Mahalanobis depth of x with respect to data.

Value

A numerical vector with the Mahalanobis depth for each value of x.

Author(s)

Michail Tsagris.
R implementation and documentation: Michail Tsagris <mtsagris@uoc.gr>.
Many 2 sample student’s t-tests

References


See Also

welch.tests,trim.mean

Examples

```r
x <- as.matrix(iris[1:50, 1:4])
depth.mahala(x, x)
```

Description

It performs very many 2 sample student’s t-tests.

Usage

```r
stud.ttests(x, y = NULL, ina, logged = FALSE, parallel = 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".
- `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 t-tests, the student’s t-test (that assumes equal variances) is performed.
Many approximate simple logistic regressions

Value

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

Author(s)

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

References


See Also

boot.student2, boot.ttest1

Examples

```r
## 1000 variables, hence 20 t-tests will be performed
x = matrix( rnorm(100 * 20), ncol = 20)
## 100 observations in total
ina = rbinom(100, 1, 0.6) + 1  ## independent samples t-test
system.time( stud.ttests(x, ina = ina))
x1 = x[ina == 1,]
x2 = x[ina == 2,]
system.time( stud.ttests(x1, x2) )
x <- NULL
```

Description

Many approximate simple logistic regressions.

Usage

```r
sp.logiregs(y, x, logged = FALSE)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>y</td>
<td>The dependent variable, a numerical vector with 0s or 1s.</td>
</tr>
<tr>
<td>x</td>
<td>A matrix with the independent variables.</td>
</tr>
<tr>
<td>logged</td>
<td>Should the p-values be returned (FALSE) or their logarithm (TRUE)?</td>
</tr>
</tbody>
</table>
Details

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

Value

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

Author(s)

Initial author Karolina Sikorska. Modifications by Michail Tsagris.

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

References


https://bmcbioinformatics.biomedcentral.com/track/pdf/10.1186/1471-2105-14-166

See Also

`logiquant.regs`, `bic.regs`

Examples

```r
y <- rbinom(200, 1, 0.5)
x <- matrix(rnorm(200 * 50), ncol = 50 )
a <- sp.logiregs(y, x)
```

Many binary classification metrics

`Many binary classification metrics`
Many binary classification metrics

Usage

colaccs(group, preds)
colsens(group, preds)
colspecs(group, preds)
colprecis(group, preds)
colfscores(group, preds)
colfbscores(group, preds, b)
colfmis(group, preds)

Arguments

group A numerical vector with two values, 0 and 1.
preds A numerical matrix with scores, probabilities or any other measure.
b The $\beta$ parameter in the $F_\beta$-score.

Details

The accuracies, sensitivities, specificities, precisions, F-scores, $F_\beta$-scores and the Fowlkes-Mallows index are calculated column-wise. The colaccs is the only metric that can be used with a multinomial response as well.

Value

A vector with length equal to the number of columns of the "preds" argument containing the relevant values computed for each column.

Author(s)

Michail Tsagris.

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

References

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

See Also

colmses, bernoulli.nb, bigknn.cv

Examples

## 20 variables, hence 20 accuracies will be calculated
ina <- rbinom(100, 1, 0.6)
x <- matrix( rnorm(100 * 20), ncol = 20 )
a <- colaccs(ina, x)
Many Gamma regressions

Description

Many Gamma regressions.

Usage

gammaregs(y, x, tol = 1e-07, logged = FALSE, parallel = 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 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 Michail Tsagris.

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

References


Many metrics for a continuous response variable

See Also

bic.regs, gammareg

Examples

## Not run:
y <- rgamma(100, 3, 10)
x <- matrix(rnorm(100 * 10), ncol = 10)
b <- glm(y ~ x[, 1], family = Gamma(log))
anova(b, test = "F")
a <- gammaregs(y, x)
x <- NULL
## End(Not run)

---

Many metrics for a continuous response variable

*any metrics for a continuous response variable*

---

**Description**

any metrics for a continuous response variable.

**Usage**

```r
colmses(y, yhat, parallel = FALSE)
colmaes(y, yhat, parallel = FALSE)
colpkl(y, yhat, parallel = FALSE)
colukl(y, yhat, parallel = FALSE)
```

**Arguments**

- `y`: A numerical vector.
- `yhat`: A numerical matrix with with the predictions.
- `parallel`: If you want parallel computations set this equal to TRUE.

**Details**

The mean squared errors, mean absolute errors, and Kullback-Leibler divergence for percentages (colpkl) and non-negative values or discrete values (colukl) are computed.

**Value**

A vector with length equal to the number of columns of the "yhat" argument containing the relevant values computed for each column.
Many negative binomial regressions

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

See Also
colaccs, bigknn.cv, mmpc, pc.sel

Examples
```r
## 20 variables, hence 20 MSEs will be calculated
y <- rnorm(100, 1, 0.6)
yhat <- matrix( rnorm(100 * 20), ncol = 20 )
a <- colmses(y, yhat)
```

Description
Many simple negative binomial regressions.

Usage
```r
negbin.regs(y, x, type = 1, tol = 1e-07, logged = FALSE, parallel = FALSE, maxiters = 100)
```

Arguments
- `y`: The dependent variable, a numerical variable with non negative numbers.
- `x`: A matrix with the independent variables.
- `type`: You can choose which way you prefer. Type 1 is for small sample sizes, whereas type 2 is for larger ones as is faster.
- `tol`: The tolerance value to terminate the Newton-Raphson algorithm.
- `logged`: If you want the logarithm of the p-values set this equal 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 negative binomial regressions with a log-link are fitted.

Value
A matrix with the test statistic values and their relevant (logged) p-values.
Many score based regressions with multiple response variables and a single predictor variable

Author(s)

Stefanos Fafalios and Michail Tsagris.

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

References


See Also

bic.regs, negbin.reg, score.zipregs, ztp.reg

Examples

```r
## Not run:
y <- rnbinom(100, 10, 0.6)
x <- matrix( rnorm( 100 * 200), ncol = 200 )
a <- negbin.regs(y, x)
x <- NULL
## End(Not run)
```

Description

Many score based regressions with multiple response variables and a single predictor variable.

Usage

```r
mv.score.glms(y, x, oiko = NULL, logged = FALSE)
mv.score.weibregs(y, x, logged = FALSE)
mv.score.betaregs(y, x, logged = FALSE)
mv.score.gammaregs(y, x, logged = FALSE)
mv.score.expregs(y, x, logged = FALSE)
mv.score.invgaussregs(y, x, logged = FALSE)
```
Arguments

$y$ A matrix with either discrete or binary data for the Poisson or binary logistic regression respectively. For the Weibull, gamma, inverse Gaussian and exponential regressions the values of $y$ must be strictly positive data, lifetimes or durations for example. For the beta regression they must be numbers between 0 and 1.

$x$ A vector with continuous data, the predictor variable.

$oiko$ This can be either "poisson" or "binomial".

$logged$ A boolean variable; it will return the logarithm of the p-value if set to TRUE.

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 least 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 then 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 all other regression models via the $\chi^2$ distribution.

Author(s)

Michail Tsagris.

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

References


Many score based zero inflated Poisson regressions

See Also

score.zipregs, gammaregs, weib.regs

Examples

```r
y <- matrix(rbeta(100 * 10, 2, 3), ncol = 10)
x <- rnorm(100)
a <- mv.score.betaregs(y, x)
y <- NULL
```

Many score based zero inflated Poisson regressions

Description

Many score based zero inflated Poisson regressions.

Usage

```r
score.zipregs(y, x, logged = FALSE )
```

Arguments

- `y`: A vector with discrete data, counts.
- `x`: A matrix with data, the predictor variables.
- `logged`: A boolean variable; it will return the logarithm of the p-value if set to TRUE.

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
need to be fitted. The first derivative of the log-likelihood is known in closed form and under the
null hypothesis the fitted values are all equal to the mean of the response variable \( y \). 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. It is also much
faster then the classical likelihood ratio test.

Value

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

Author(s)

Michail Tsagris.

R implementation and documentation: Michail Tsagris <mtsagris@uoc.gr>.
Many simple quantile regressions using logistic regressions

References


See Also

ztp.reg, censpois.mle

Examples

```r
x <- matrix(rnorm(1000 * 100), ncol = 100)
y <- rpois(1000, 10)
y[1:150] <- 0
a <- score.zipregs(y, x)
x <- NULL
mean(a < 0.05) # # estimated type I error
```

Description

Many simple quantile regressions using logistic regressions.

Usage

`logiquant.regs(y, x, logged = FALSE)`

Arguments

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

Details

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

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

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

References

See Also
bic.regs, sp.logiregs

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

Description
Many simple Weibull regressions.

Usage
weib.regs(y, x, tol = 1e-07, logged = FALSE, parallel = FALSE, maxiters = 100)

Arguments

y
The dependent variable, either a numerical variable with numbers greater than zero.

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++, 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.
Many Welch tests

Details
Many simple weibull regressions are fitted.

Value
A matrix with the test statistic values and their associated (logged) p-values.

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

See Also
bic.regs

Examples

```r
y <- rgamma(100, 3, 4)
x <- matrix(rnorm(100 * 30), ncol = 30)
a <- weib.regs(y, x)
x <- NULL
```

Description
Many Welch tests.

Usage
welch.tests(y, x, logged = FALSE, parallel = FALSE)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>y</td>
<td>The dependent variable, a numerical vector.</td>
</tr>
<tr>
<td>x</td>
<td>A matrix with the independent variables. They must be integer valued data starting from 1, not 0 and be consecutive numbers. Instead of a data.frame with factor variables, the user must use a matrix with integers.</td>
</tr>
<tr>
<td>logged</td>
<td>Should the p-values be returned (FALSE) or their logarithm (TRUE)?</td>
</tr>
<tr>
<td>parallel</td>
<td>If you want to run the function in parallel set this equal to TRUE.</td>
</tr>
</tbody>
</table>

Details
For each categorical predictor variable, a Welch test is performed. This is useful in feature selection algorithms, to determine for which variable, the means of the dependent variable differ across the different values.
Max-Min Parents and Children variable selection algorithm for continuous responses

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

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

References

See Also
sp.logiregs, pc.sel

Examples
y <- rnorm(200)
x <- matrix(rbinom(200 * 50, 2, 0.5), ncol = 50) + 1
a <- welch.tests(y, x)

Description
Max-Min Parents and Children variable selection algorithm for continuous responses.

Usage
mmpc(y, x, max_k = 3, alpha = 0.05, method = "pearson",
ini = NULL, hash = FALSE, hashobject = NULL, backward = FALSE)

Arguments
y The class variable. Provide a numeric vector.
x The main dataset. Provide a numeric matrix.
max_k The maximum conditioning set to use in the conditional independence test. Provide an integer.
The default value set is 3.
Max-Min Parents and Children variable selection algorithm for continuous responses

alpha  
Threshold for assessing p-values’ significance. Provide a double value, between 0.0 and 1.0.  
The default value set is 0.05.

method  
Currently only "pearson" is supported.

ini  
This argument is used for the avoidance of the univariate associations re-calculations, in the case of them being present. Provide it in the form of a list.

hash  
Boolean value for the activation of the statistics storage in a hash type object.  
The default value is false.

hashobject  
This argument is used for the avoidance of the hash re-calculation, in the case of them being present, similarly to ini argument. Provide it in the form of a hash. Please note that the generated hash object should be used only when the same dataset is re-analyzed, possibly with different values of max_k and alpha.

backward  
Boolean value for the activation of the backward/symmetry correction phase. This option removes and falsely included variables in the parents and children set of the target variable. It calls the link{mmpc_bp} for this purpose. The backward option seems dubious. Please do not use at the moment.

Details

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

Value

The output of the algorithm is an list including:

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

hashobject  
The hash object containing the statistics calculated in the current run.

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

stats  
The statistics corresponding to the aforementioned pvalues (higher values indicate higher association).

univ  
This is a list with the univariate associations; the test statistics and their corresponding logged p-values.

max_k  
The max_k value used in the current execution.

alpha  
The alpha value used in the current execution.

n.tests  
If hash = TRUE, the number of tests performed will be returned. If hash != TRUE, the number of univariate associations will be returned.

runtime  
The time (in seconds) that was needed for the execution of algorithm.
Max-Min Parents and Children variable selection algorithm for non continuous responses

Author(s)

Marios Dimitriadis.

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

References


See Also

mmpc

Examples

set.seed(123)

# Dataset with continuous data
ds <- matrix(runif(100 * 500, 1, 100), ncol = 500)

# Class variable
tar <- 3 * ds[, 10] + 2 * ds[, 100] + 3 * ds[, 20] + rnorm(100, 0, 5)

mmpc(tar, ds, max_k = 3, alpha = 0.05, method = "pearson")

Max-Min Parents and Children variable selection algorithm for non continuous responses

Description

Max-Min Parents and Children variable selection algorithm for non continuous responses.
Usage

\begin{verbatim}
mmpc2(y, x, max_k = 3, threshold = 0.05, test = "logistic", init = NULL, 
tol = 1e-07, backward = FALSE, maxiters = 100, parallel = FALSE)
\end{verbatim}

Arguments

- **y**: The response variable, a numeric vector with either count data or binary data.
- **x**: A numerical matrix with the independent (predictor) variables.
- **max_k**: The maximum conditioning set to use in the conditional independence test (see Details). Integer, default value is 3.
- **threshold**: Threshold (suitable values in (0, 1)) for assessing p-values significance. Default value is 0.05.
- **test**: One of the following: "logistic", "poisson", "qpoisson".
- **init**: A numeric vector with the logged p-values of the univariate associations. **Do not use this at the moment.**
- **tol**: The tolerance value to stop the Newtn-Raphson algorithm inside a regression model.
- **backward**: If TRUE, the backward (or symmetry correction) phase will be implemented. This removes any falsely included variables in the parents and children set of the target variable. It calls the link\{mmpcbackphase\} for this purpose.
- **maxiters**: The maximum number of iterations a Newtn-Raphson algorithm will perform inside a regression model.
- **parallel**: Do you want the computations to take part in parallel? Set TRUE if yes.

Details

MMPC tests each feature for inclusion (selection). It is a variant of the forward selection procedure. a) at every step it removes the non significant variables and does not check them again. b) Instead of testing a candidate variable conditioning on all previously selected variables, it uses subsets of the previously selected variables. All possible subsets of maximum size equal to max_k. With the appropriate pre-computations, at every step, it performs only the tests that were not executed before, so it is not that time consuming.

Value

The output of the algorithm is an S3 object including:

- **selectedVars**: The selected variables, i.e., the signature of the target variable.
- **pvalues**: For each feature included in the dataset, this vector reports the strength of its association with the target in the context of all other variable. Particularly, this vector reports the max p-values found when the association of each variable with the target is tested against different conditional sets. Lower values indicate higher association.
Max-Min Parents and Children variable selection algorithm for non continuous responses

univ A vector with the logged p-values of the univariate associations. This vector is very important for subsequent runs of MMPC with different hyper-parameters. After running SES with some hyper-parameters you might want to run MMPC again with different hyper-parameters. To avoid calculating the univariate associations (first step) again, you can take this list from the first run of SES and plug it in the argument "ini" in the next run(s) of MMPC. This can speed up the second run (and subsequent runs of course) by 50%. See the argument "univ" in the output values.

kapa_pval A list with the same number of elements as the max_k. Every element in the list is a matrix. The first column is the logged p-values, the second column is the variable whose conditional association with the response variable was tested and the other columns are the conditioning variables.

max_k The max_k option used in the current run.

threshold The threshold option used in the current run.

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

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

Author(s)

Manos Papadakis.

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

References


See Also

mmpc, pc.sel, fbed.reg

Examples

```r
y <- rbinom(100, 1, 0.5)
x <- matrix( rnorm(100 * 500), ncol = 500 )
ml <- mmpc2(y, x, max_k = 3, threshold = 0.05, test = "logistic")
```
Maximum likelihood linear discriminant analysis

m2 <- fbed.reg(y, x, type = "logistic")

Maximum likelihood linear discriminant analysis

Description

Maximum likelihood linear discriminant analysis.

Usage

mle.lda(xnew, x, ina)

Arguments

xnew A numerical vector or a matrix with the new observations, continuous data.
x A matrix with numerical data.
ina A numerical vector or factor with consecutive numbers indicating the group to which each observation belongs to.

Details

Maximum likelihood linear discriminant analysis is performed.

Value

A vector with the predicted group of each observation in "xnew".

Author(s)

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

References


See Also

fisher.da, reg.mle.lda, big.knn, weibull.nb

Examples

x <- as.matrix(iris[, 1:4])
ina <- iris[, 5]
a <- mle.lda(x, x, ina)
Description

Merge 2 sorted vectors in 1 sorted vector.

Usage

Merge(x, y)

Arguments

x  A sorted vector with data.
y  A sorted vector with data.

Value

A sorted vector of the 2 arguments.

Author(s)

Manos Papadakis.

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

See Also

is.lower.tri, is.upper.tri

Examples

x <- 1:10
y <- 1:20

Merge(x, y)

x <- y <- NULL
Description

MLE of continuous univariate distributions defined on the positive line.

Usage

halfcauchy.mle(x, tol = 1e-07)
powerlaw.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.

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 power law we assume that the minimum value of x is above zero in order to perform the maximum likelihood estimation in the usual way.

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.
scale The scale parameter of the half Cauchy distribution.
alpha The value of the power parameter for the power law distribution.

Author(s)

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

References

You can also check the relevant wikipedia pages for these distributions.
**MLE of distributions defined for proportions**

**See Also**

`zigamma.mle, censweibull.mle`

**Examples**

```r
x <- abs(rcauchy(1000, 0, 2))
halfcauchy.mle(x)
```

**Description**

MLE of distributions defined for proportions.

**Usage**

```r
kumar.mle(x, tol = 1e-07, maxiters = 50)
simplex.mle(x, tol = 1e-07)
zil.mle(x)
unitweibull.mle(x, tol = 1e-07, maxiters = 100)
cbern.mle(x, tol = 1e-6)
```

**Arguments**

- `x` A vector with proportions or percentages. Zeros are allowed only for the zero inflated logistic normal distribution (`zil.mle`).
- `tol` The tolerance level up to which the maximisation stops.
- `maxiters` The maximum number of iterations the Newton-Raphson will perform.

**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. The distributions are Kumaraswamy, zero inflated logistic normal, simplex, unit Weibull and continuous Bernoulli.

**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.
- `param` The two parameters (shape and scale) of the Kumaraswamy distribution or the means and sigma of the simpled distribution. For the zero inflated logistic normal, the probability of non zeros, the mean and the unbiased variance.
- `loglik` The value of the maximised log-likelihood.
Author(s)
Michail Tsagris.
R implementation and documentation: Michail Tsagris <mtsagris@uoc.gr>.

References


You can also check the relevant wikipedia pages.

See Also
zigamma.mle, censweibull.mle

Examples

```r
u <- runif(1000)
 a <- 0.4 ; b <- 1
x <- ( 1 - (1 - u)^((1/b) )*(1/a)
kumar.mle(x)
```

Description
MLE of some circular distributions with multiple samples.

Usage

```r
multivm.mle(x, ina, tol = 1e-07, ell = FALSE)
multispml.mle(x, ina, tol = 1e-07, ell = FALSE)
```
MLE of some circular distributions with multiple samples

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.

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.

tol The tolerance level to stop the iterative process of finding the MLEs.

e11 Do you want the log-likelihood returned? The default value is FALSE.

Details

The parameters of the von Mises and of the bivariate angular Gaussian distributions are estimated for multiple samples.

Value

A list including:

iters The iterations required until convergence. This is returned in the wrapped Cauchy distribution only.

loglik A vector with the value of the maximised log-likelihood for each sample.

mi For the von Mises, this is a vector with the means of each sample. For the angular Gaussian (spml), a matrix with the mean vector of each sample

ki A vector with the concentration parameter of the von Mises distribution at each sample.

gi A vector with the norm of the mean vector of the angular Gaussian distribution at each sample.

Author(s)

Michail Tsagris.

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

References


See Also

colspml.mle, purka.mle
MLE of some truncated distributions

Examples

y <- rcauchy(100, 3, 1)
x <- y
ina <- rep(1:2, 50)
multivm.mle(x, ina)
multispml.mle(x, ina)

Description

MLE of some truncated distributions.

Usage

trunccauchy.mle(x, a, b, tol = 1e-07)
truncexpmle(x, b, tol = 1e-07)

Arguments

x A numerical vector with continuous data. For the Cauchy distribution, they can be anywhere on the real line. For the exponential distribution they must be strictly positive.
a The lower value at which the Cauchy distribution is truncated.
b The upper value at which the Cauchy or the exponential distribution is truncated. For the exponential this must be greater than zero.
tol The tolerance value to terminate the fitting algorithm.

Details

Maximum likelihood of some truncated distributions is performed.

Value

A list including:

- iters The number of iterations required by the Newton-Raphson algorithm.
- loglik The log-likelihood.
- lambda The \( \lambda \) parameter in the exponential distribution.
- param The location and scale parameters in the Cauchy distribution.

Author(s)

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

http://lagrange.math.siu.edu/Olive/ol-bookp.htm

See Also

purka.mle

Examples

```r
x <- rnorm(500)
trunccauchy.mle(x, -1, 1)
```

Description

MLE of the Cauchy and generalised normal distributions with zero location.

Usage

```r
cauchy0.mle(x, tol = 1e-07)
gnormal0.mle(x, tol = 1e-06)
```

Arguments

- `x`: A numerical vector with positive real numbers.
- `tol`: The tolerance level up to which the maximisation stops set to 1e-07 by default.

Details

The Cauchy is the t distribution with 1 degree of freedom. The cauchy0.mle estimates the usual Cauchy distribution, over the real line, but assumes a zero location. The gnormal0.mle estimates the generalised normal distribution assuming a zero location. The generalised normal distribution is also known as the exponential power distribution or the generalized error distribution.

Value

A list including:

- `iters`: The number of iterations required by the Newton-Raphson algorithm.
- `loglik`: The value of the maximised log-likelihood.
- `scale`: The estimated scale parameter of the Cauchy distribution.
- `param`: The estimated scale and shape parameters of the generalised normal distribution.
MLE of the censored Weibull distribution

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

References

See Also
   censweibull.mle

Examples
   x <- rcauchy(150, 0, 2)
   cauchy0.mle(x)
   x <- rnorm(200)
   gnormal0.mle(x)

MLE of the censored Weibull distribution

Description
MLE of the censored Weibull distribution.

Usage
   censweibull.mle(x, di, tol = 1e-07)

Arguments
   x A vector with positive valued data (zeros are not allowed).
   di A vector of 0s (censored) and 1s (not censored) values.
   tol The tolerance level up to which the maximisation stops; set to 1e-07 by default.

Details
Instead of maximising the log-likelihood via a numerical optimiser we have used a Newton-Raphson algorithm which is faster.
MLE of the gamma-Poisson distribution

Value
A list 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>.

References

See Also
km, censpois.mle

Examples
x <- rweibull(300, 3, 6)
censweibull.mle(x, di = rep(1, 300))
di <- rbinom(300, 1, 0.9)
censweibull.mle(x, di)

MLE of the gamma-Poisson distribution

MLE of the gamma-Poisson distribution

Description
MLE of the gamma-Poisson distribution.

Usage
gammapois.mle(x, tol = 1e-07)

Arguments
x: A numerical vector with positive data and zeros.
tol: The tolerance value to terminate the Newton-Raphson algorithm.
MLE of the left censored Poisson distribution

Details

MLE of the gamma-Poisson distribution is fitted. When the rate in the Poisson follows a gamma distribution with shape = r and scale $\theta$, the resulting distribution is the gamm-Poisson. If the shape $r$ is integer, the distribution is called negative binomial distribution.

Value

A list including:

- `iters` The iterations required by the Newton-Raphson to estimate the parameters of the distribution for the non zero data.
- `loglik` The full log-likelihood of the model.
- `param` The parameters of the model.

Author(s)

Michail Tsagris.

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

References


See Also

`zigamma.mle`

Examples

```r
x <- rnbinom(200, 20, 0.7)
gammapois.mle(x)
```

---

MLE of the left censored Poisson distribution

`MLE of the left censored Poisson distribution`

Description

MLE of the left censored Poisson distribution.

Usage

```r
censpois.mle(x, tol = 1e-07)
```
MLE of the Purkayastha distribution

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-07 by default.

Details

Instead of maximising the log-likelihood via a numerical optimiser we have used a Newton-Raphson algorithm which is faster. The lowest value in x is taken as the censored point. Values below are censored.

Value

A list including:

- iters The number of iterations required for the Newton-Raphson to converge.
- loglik The value of the maximised log-likelihood.
- lambda The estimated $\lambda$ parameter.

Author(s)

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

See Also

km, censweibull.mle

Examples

```r
x1 <- rpois(10000, 15)
x <- x1
x[x <= 10] = 10
mean(x)
censpois.mle(x)$lambda
```

---

MLE of the Purkayastha distribution

MLE of the Purkayastha distribution

Description

MLE of the Purkayastha distribution.

Usage

purka.mle(x, tol = 1e-07)
MLE of the Purkayastha distribution

Arguments

x
A numerical vector with data expressed in radians or a matrix with spherical data.

tol
The tolerance value to terminate the Brent algorithm.

Details

MLE of the Purkayastha distribution is performed.

Value

A list including:

theta
The median direction.

alpha
The concentration parameter.

loglik
The log-likelihood.

alpha.sd
The standard error of the concentration parameter.

Author(s)

Michail Tsagris.

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

References


See Also

circ.cor1

Examples

```r
x <- cbind( rnorm(100,1,1), rnorm(100, 2, 1) )
x <- x / sqrt(rowSums(x^2))
purka.mle(x)
```
MLE of the zero inflated Gamma and Weibull distributions

Description

MLE of the zero inflated Gamma and Weibull distributions.

Usage

`zigamma.mle(x, tol = 1e-07)`
`ziweibull.mle(x, tol = 1e-07)`

Arguments

- `x` A numerical vector with positive data and zeros.
- `tol` The tolerance value to terminate the Newton-Raphson algorithm.

Details

MLE of some zero inflated models is performed.

Value

A list including:

- `iters` The iterations required by the Newton-Raphson to estimate the parameters of the distribution for the non zero data.
- `loglik` The full log-likelihood of the model.
- `param` The parameters of the model.

Author(s)

Michail Tsagris.

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

References


See Also

`zigamma.reg`, `gammapois.mle`
Monte Carlo integration with a normal distribution

Examples

\begin{verbatim}
x <- rgamma(200, 4, 1)
x[sample(1:200, 20)] <- 0
zigamma.mle(x)
\end{verbatim}

Description

Monte Carlo Integration with a normal distribution.

Usage

\begin{verbatim}
mci(fun, R = 10^6)
\end{verbatim}

Arguments

- **fun** A function denoting the inside part of the expectation to be computed.
- **R** The number of draws from the normal distribution.

Value

The result of the integral.

Author(s)

Michail Tsagris.

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

References


See Also

- riag, rbeta1

Examples

\begin{verbatim}
## compute the expectation of abs(x)
fun <- function(x) abs(x)
mci(fun, R = 10^5)
a <- function(x) abs(x) * dnorm(x)
integrate(a, -Inf, Inf)
\end{verbatim}
Moran's I measure of spatial autocorrelation

Description

Moran’s I measure of spatial autocorrelation.

Usage

moranI(x, w, scaled = FALSE, R = 999)

Arguments

x
A numerical vector with observations.

w
The inverse of a (symmetric) distance matrix. After computing the distance
matrix, you invert all its elements and the elements which are zero (diagonal)
and have become Inf. set them to 0. This is the w matrix the functions requires.
If you want an extra step, you can row standardise this matrix by dividing each
row by its total. This will makw the rowsums equal to 1.

scaled
If the matrix is row-standardised (all rowsums are equal to 1) then this is TRUE
and FALSE otherwise.

R
The number of permutations to use in order to obtain the permutation based-
pvalue. If R is 1 or less no permutation p-value is returned.

Details

Moran’ I index is a measure of spatial autocorrelation. that was proposed in 1950. Instead of
computing an asymptotic p-value we compute a permutation based p-value utilizing the fast method
of Chatzipantsiou et al. (2019).

Value

A vector of two values, the Moran’s I index and its permutation based p-value. If R is 1 or less no
permutation p-value is returned, and the second element is "NA".

Author(s)

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

References

Chatzipantsiou C., Dimitriadis M., Papadakis M. and Tsagris M. (2019). Extremely efficient per-
mutation and bootstrap hypothesis tests using R. Journal of Modern Applied Statistical Methods
Multinomial regression

See Also
censpois.mle, gammapois.mle

Examples

x <- rnorm(50)
w <- as.matrix( dist(iris[1:50, 1:4]) )
w <- 1/w
diag(w) <- 0
moranI(x, w, scaled = FALSE)

Multinomial regression

Multinomial regression

Description
Multinomial regression.

Usage

multinom.reg(y, x, tol = 1e-07, maxiters = 100)

Arguments

y The response variable. A numerical or a factor type vector.
x A matrix or a data.frame with the predictor variables.
tol The 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 and Stefanos Fafalios.

R implementation and documentation: Michail Tsagris <mtsagris@uoc.gr> and Stefanos Fafalios <stefanosfafalios@gmail.com>.
Naive Bayes classifier for binary (Bernoulli) data

References


See Also

logiquant.regs, fbed.reg

Examples

```r
y <- iris[, 5]
x <- matrix( rnorm(150 * 2), ncol = 2 )
mod <- multinom.reg(y, x)
```

Naive Bayes classifier for binary (Bernoulli) data

```r
Naive Bayes classifier for binary Bernoulli data
```

Description

Naive Bayes classifier for binary (Bernoulli) data.

Usage

```r
bernoulli.nb(xnew = NULL, x, ina)
```

Arguments

- `xnew`: A numerical matrix with new predictor variables whose group is to be predicted. Each column contains binary (0 or 1) data.
- `x`: A numerical matrix with observed predictor variables. Each column contains binary (0 or 1) data.
- `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.

Details

Each column is supposed to contain binary data. Thus, for each column a Bernoulli distributions is fitted. The product of the densities is the joint multivariate distribution.
Value

A list including:

- **pi**: A matrix with the estimated probabilities of each group and variable.
- **ni**: The sample size of each group in the dataset.
- **est**: The estimated group of the \( x_{\text{new}} \) 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(ina)\' 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>.

See Also

- [bernoullinb.pred](#), [nb.cv](#)

Examples

```r
x <- matrix(rbinom(50 * 4, 1, 0.5), ncol = 4)
ina <- rbinom(50, 1, 0.5) + 1
a <- bernoulli.nb(x, x, ina)
```

Description

Naive Bayes classifiers.

Usage

```r
weibull.nb(xnew = NULL, x, ina, tol = 1e-07)
normlog.nb(xnew = NULL, x, ina)
laplace.nb(xnew = NULL, x, ina)
logitnorm.nb(xnew = NULL, x, ina)
beta.nb(xnew = NULL, x, ina)
cauchy.nb(xnew = NULL, x, ina)
```
Arguments

\( \text{xnew} \) A numerical matrix with new predictor variables whose group is to be predicted. This is set to NUUL, as you might want just the model and not to predict the membership of new observations. For the normlog this contains positive numbers (greater than or equal to zero), but for the multinomial and Poisson cases, the matrix must contain integer valued numbers only. For the logistic normal (logitnorm.nb) and beta (beta.nb) the data must be strictly between 0 and 1.

\( x \) A numerical matrix with the observed predictor variable values. For the Gaussian case (normlognb.nb) this contains positive numbers (greater than or equal to zero), but for the multinomial and Poisson cases, the matrix must contain integer valued numbers only. For the logistic normal (logitnorm.nb) and beta (beta.nb) the data must be strictly between 0 and 1.

\( \text{ina} \) 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.

\( \text{tol} \) The tolerance value to terminate the Newton-Raphson algorithm in the Weibull distribution.

Value

Depending on the classifier a list including (the \( \text{ni} \) and \( \text{est} \) are common for all classifiers):

- \( \text{shape} \) A matrix with the shape parameters.
- \( \text{scale} \) A matrix with the scale parameters.
- \( \text{expmu} \) A matrix with the mean parameters.
- \( \text{sigma} \) A matrix with the (MLE, hence biased) variance parameters.
- \( \text{location} \) A matrix with the location parameters (medians).
- \( \text{scale} \) A matrix with the scale parameters.
- \( \text{mean} \) A matrix with the scale parameters.
- \( \text{var} \) A matrix with the variance parameters.
- \( \text{a} \) A matrix with the "alpha" parameters.
- \( \text{b} \) A matrix with the "beta" parameters.
- \( \text{ni} \) The sample size of each group in the dataset.
- \( \text{est} \) 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(ina)\` 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>.

See Also

weibullnb.pred, vm.nb, nb.cv, mle.lda, big.knn
Examples

```r
x <- matrix(rweibull(100, 3, 4), ncol = 2)
ina <- rbinom(50, 1, 0.5) + 1
a <- weibull.nb(x, x, ina)
```

Description

Naive Bayes classifiers for directional data.

Usage

```r
vm.nb(xnew = NULL, x, ina, tol = 1e-07)
spml.nb(xnew = NULL, x, ina, tol = 1e-07)
```

Arguments

- `xnew`: A numerical matrix with new predictor variables whose group is to be predicted. Each column refers to an angular variable.
- `x`: A numerical matrix with observed predictor variables. Each column refers to an angular variable.
- `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.
- `tol`: The tolerance value to terminate the Newton-Raphson algorithm.

Details

Each column is supposed to contain angular measurements. Thus, for each column a von Mises distribution or a circular angular Gaussian distribution is fitted. The product of the densities is the joint multivariate distribution.

Value

A list including:

- `mu`: A matrix with the mean vectors expressed in radians.
- `mu1`: A matrix with the first set of mean vectors.
- `mu2`: A matrix with the second set of mean vectors.
- `kappa`: A matrix with the kappa parameters for the vonMises distribution or with the norm of the mean vectors for the circular angular Gaussian distribution.
- `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(ina)` in order to see what is the predicted class of the new data.
Negative binomial regression

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

See Also

vmnb.pred, weibull.nb, nb.cv

Examples

x <- matrix( runif(100, 0, 1), ncol = 2 )
ina <- rbinom(50, 1, 0.5) + 1
a <- vm.nb(x, x, ina)

Description

Negative binomial regression.

Usage

negbin.reg(y, x, tol = 1e-07, maxiters = 100)

Arguments

y 
The dependent variable, a numerical vector with integer valued numbers.

x 
A matrix or a data.frame with the indendent variables.

tol 
The tolerance value required by the Newton-Raphson to stop.

maxiters 
The maximum iterations allowed.

Details

A negative binomial regression model is fitted. The standard errors of the regressions are not returned as we do not compute the full Hessian matrix at each step of the Newton-Raphson.

Value

A list including:

be 
The regression coefficients.

loglik 
The loglikelihood of the regression model.

iters 
The iterations required by the Newton-Raphson.
Non linear least squares regression for percentages or proportions

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

See Also
ztp.reg, binom.reg

Examples

```r
y <- rnbinom(100, 10, 0.7)
x <- matrix(rnorm(100 * 3), ncol = 3)
mod <- negbin.reg(y, x)
```

---

Non linear least squares regression for percentages or proportions

Description
Non linear least squares regression for percentages or proportions.

Usage

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

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>y</td>
<td>The dependent variable, a numerical vector with percentages or proportions, including 0s and or 1s.</td>
</tr>
<tr>
<td>x</td>
<td>A matrix with the independent variables.</td>
</tr>
<tr>
<td>cov</td>
<td>Should the sandwich covariance matrix and the standard errors be returned? If yes, set this equal to TRUE.</td>
</tr>
<tr>
<td>tol</td>
<td>The tolerance value to terminate the Newton-Raphson algorithm. This is set to $10^{-7}$ by default.</td>
</tr>
<tr>
<td>maxiters</td>
<td>The maximum number of iterations that can take place during the fitting.</td>
</tr>
</tbody>
</table>

Details
The ordinary least squares between the observed and the fitted percentages is adopted as the objective function. This involves numerical optimization since the relationship is non-linear. There is no log-likelihood. This is the univariate version of the OLS regression for compositional data mentioned in Murteira and Ramalho (2016).
One sample bootstrap t-test for a vector

Value

A list including:

*sse*  
The sum of squares of the raw residuals.

*be*  
The beta coefficients.

*seb*  
The sandwich standard errors of the beta coefficients, if the input argument argument was set to TRUE.

*covb*  
The sandwich covariance matrix of the beta coefficients, if the input argument argument was set to TRUE.

*iters*  
The number of iterations required by the Newton-Raphson algorithm.

Author(s)

Michail Tsagris.

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

References


See Also

`prophelling.reg`, `simplex.mle`, `kumar.mle`

Examples

```r
y <- rbeta(150, 3, 4)
x <- iris
a <- propols.reg(y, x)
```

Description

One sample bootstrap t-test for a vector.

Usage

`boot.ttest1(x, m, R = 999)`
Orthogonal matching pursuit variable selection

Arguments

x A numerical vector with the data.
\( m \) The assumed mean value.
\( R \) The number of bootstrap resamples to draw.

Details

The usual one sample bootstrap t-test is implemented, only faster.

Value

res A two valued vector with the test statistic and its p-value.

Author(s)

Michail Tsagris.

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

See Also

boot.student2, perm.ttest2, welch.tests, jack.mean

Examples

```r
x <- rexp(30)
a <- t.test(x, mu = 0)
b <- boot.ttest1(x, 0)
```

Description

Orthogonal matching variable selection.

Usage

```r
omp2(y, x, xstand = TRUE, tol = qchisq(0.95, 1), type = "gamma")
```
Orthogonal matching pursuit variable selection

Arguments

y  The response variable, a numeric vector. For "omp" this can be either a vector with discrete (count) data, 0 and 1, non negative values, strictly positive or a factor (categorical) variable.

x  A matrix with the data, where the rows denote the observations and the columns are the variables.

xstand  If this is TRUE the independent variables are standardised.

tol  The tolerance value to terminate the algorithm. This is the change in the criterion value between two successive steps. For "ompr" the default value is 2 because the default method is "BIC". The default value is the 95% quantile of the $\chi^2$ distribution.

type  This denotes the parametric model to be used each time. It depends upon the nature of y. The possible values are "gamma", "negbin", or "multinomial".

Details

This is the continuation of the "omp" function of the Rfast. We added some more regression models. The "gamma" and the "multinomial" models have now been implemented in C++.

Value

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.

Author(s)

Michail Tsagris.

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

References


See Also

mmpc2, pc.sel

Examples

```r
x <- matrix(rnorm(100 * 50), ncol = 50)
y <- rgamma(100, 4, 1)
a <- omp2(y, x)
a
x <- NULL
```

Description

Parametric bootstrap for linear regression model.

Usage

```r
lm.parboot(x, y, R = 1000)
```

Arguments

- `x`: The predictor variables, a vector or a matrix or a data frame.
- `y`: The response variable, a numerical vector with data.
- `R`: The number of parametric bootstrap replications to perform.

Details

An efficient implementation of the parametric bootstrap for linear models is provided.

Value

A matrix with \( R \) columns and rows equal to the number of the regression parameters. Each column contains the set of a bootstrap beta regression coefficients.

Author(s)

Michail Tsagris.

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

References

Permutation t-test for 2 independent samples

See Also

lm.drop1, leverage, pc.sel, mmpc

Examples

```r
y <- rnorm(50)
x <- matrix( rnorm( 50 * 2 ), ncol = 2 )
a <- lm.parboot(x, y, 500)
```

---

Permutation t-test for 2 independent samples

### Description

Permutation t-test for 2 independent samples.

### Usage

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

### Arguments

- `x`: A numerical vector with the data.
- `y`: A numerical vector with the data.
- `B`: The number of permutations to perform.

### Details

The usual permutation based p-value is computed.

### Value

A vector with the test statistic and the permutation based p-value.

### Author(s)

Michail Tsagris.

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

### References

Prediction with naive Bayes classifier for binary (Bernoulli) data

See Also

jack.mean, trim.mean, moranI

Examples

```r
x <- rexp(30, 4)
y <- rbeta(30, 2.5, 7.5)
perm.ttest2(x, y, 999)
```

Description

Prediction with naive Bayes classifier for binary (Bernoulli) data.

Usage

```r
bernoullinb.pred(xnew, pi, ni)
```

Arguments

- `xnew`: A numerical matrix with new predictor variables whose group is to be predicted. Each column refers to an angular variable.
- `pi`: A matrix with the estimated probabilities of each group.
- `ni`: The sample size of each group in the dataset.

Details

Each column is supposed to contain binary data. Thus, for each column a Bernoulli distribution is fitted. The product of the densities is the joint multivariate distribution.

Value

A numerical vector with 1, 2, ... denoting the predicted group.

Author(s)

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

See Also

bernoulli.nb, nb.cv
**Examples**

```r
x <- matrix(rbinom(50 * 4, 1, 0.5), ncol = 4)
ina <- rbinom(50, 1, 0.5) + 1
a <- bernoulli.nb(x, x, ina)
```

**Description**

Prediction with some naive Bayes classifiers.

**Usage**

- `weibullnb.pred(xnew, shape, scale, ni)`
- `normlognb.pred(xnew, expmu, sigma, ni)`
- `laplacenb.pred(xnew, location, scale, ni)`
- `logitnormnb.pred(xnew, m, s, ni)`
- `betanb.pred(xnew, a, b, ni)`
- `cauchynb.pred(xnew, location, scale, ni)`

**Arguments**

- `xnew` : A numerical matrix with new predictor variables whose group is to be predicted. For the Gaussian naive Bayes, this is set to NUUL, as you might want just the model and not to predict the membership of new observations. For the Gaussian case this contains positive numbers (greater than or equal to zero), but for the multinomial and Poisson cases, the matrix must contain integer valued numbers only. For the logistic normal (logitnormnb.pred) the data must be percentages strictly between 0 and 1.
- `shape` : A matrix with the group shape parameters. Each row corresponds to a group.
- `scale` : A matrix with the group scale parameters of the Laplace or the Cauchy distribution. Each row corresponds to a group.
- `expmu` : A matrix with the group mean parameters. Each row corresponds to a group.
- `m` : A matrix with the group mean parameters. Each row corresponds to a group.
- `sigma` : A matrix with the group (MLE, hence biased) variance parameters. Each row corresponds to a group.
- `s` : A matrix with the group MLE variance parameters. Each row corresponds to a group.
- `location` : A matrix with the group location parameters of the Laplace or of the Cauchy distribution. Each row corresponds to a group.
- `a` : A matrix with the group "alpha" parameters of the beta distribution. Each row corresponds to a group.
Prediction with some naive Bayes classifiers for circular data

b  A matrix with the group "beta" parameters of the beta distribution. Each row
corresponds to a group.

ni  A vector with the frequencies 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>.

See Also

weibull.nb,vmnb.pred,nb,cv

Examples

```r
x <- matrix( rweibull( 100, 3, 4 ), ncol = 2 )
ina <- rbinom(50, 1, 0.5) + 1
a <- weibull.nb(x, x, ina)
est <- weibullnb.pred(x, a$shape, a$scale, a$ni)
table(ina, est)
```

Description

Prediction with some naive Bayes classifiers for circular data.

Usage

```r
vmnb.pred(xnew, mu, kappa, ni)
spmlnb.pred(xnew, mu1, mu2, ni)
```

Arguments

```r
xnew  A numerical matrix with new predictor variables whose group is to be predicted.
       Each column refers to an angular variable.

mu    A matrix with the mean vectors expressed in radians.

mu1   A matrix with the first set of mean vectors.

mu2   A matrix with the second set of mean vectors.

kappa A matrix with the kappa parameters for the vonMises distribution or with the
       norm of the mean vectors for the circular angular Gaussian distribution.

ni    The sample size of each group in the dataset.
```
Principal component analysis

Details
Each column is supposed to contain angular measurements. Thus, for each column a von Mises distribution or an circular angular Gaussian distribution is fitted. The product of the densities is the joint multivariate distribution.

Value
A numerical vector with 1, 2, ... denoting the predicted group.

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

See Also
vm.nb, weibullnb.pred, nb.cv

Examples
x <- matrix( runif( 100, 0, 1 ), ncol = 2 )
ina <- rbinom(50, 1, 0.5) + 1
a <- vm.nb(x, x, ina)
a2 <- vmnb.pred(x, a$mu, a$kappa, a$ni)

Description
Principal component analysis.

Usage
pca(x, center = TRUE, scale = TRUE, k = NULL, vectors = FALSE)

Arguments
x A numerical n × 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.
Details

The function is a faster version of R’s prcomp.

Value

A list including:

values  The eigenvalues.
vectors  The eigenvectors.

Author(s)

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

See Also

reg.mle.lda

Examples

x <- matrix( rnorm(1000 * 20), ncol = 20)
a <- pca(x)
x <- NULL

Description

Principal components regression.

Usage

pcr(y, x, k = 1, xnew = NULL)

Arguments

y  A real values vector.
x  A matrix with the predictor variable(s), they have to be continuous.
k  The number of principal components to use. This can be a single number or a vector starting from 1. In the second case you get results for the sequence of principal components.
xnew  If you have new data use it, otherwise leave it NULL.
Details

The principal components of the cross product of the independent variables are obtained and classical regression is performed.

Value

A list including:

- **be**: The beta coefficients of the predictor variables computed via the principal components.
- **per**: The percentage of variance of the cross product of the independent variables explained by the k components.
- **vec**: The principal components, the loadings.
- **est**: The fitted or the predicted values (if xnew is not NULL).

Author(s)

Michail Tsagris.

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

References


See Also

pca

Examples

```r
y <- as.vector(iris[, 1])
x <- as.matrix(iris[, 2:4])
mod1 <- pcr(y, x, 1)
mod2 <- pcr(y, x, 2)
mod <- pcr(y, x, k = 1:3) ## all results at once
```

Usage

`refmeta(yi, vi, tol = 1e-07)`
Arguments

- **yi**: The observations.
- **vi**: The variances of the observations.
- **tol**: The tolerance value to terminate Brent's algorithm.

Details

Random effects estimation, via restricted maximum likelihood estimation (REML), of the common mean.

Value

A vector with many elements. The fixed effects mean estimate, the \( \bar{v} \) estimate, the \( I^2 \), the \( H^2 \), the Q test statistic and its p-value, the \( \tau^2 \) estimate and the random effects mean estimate.

Author(s)

Michail Tsagris.

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

References


See Also

- **bic.regs**

Examples

```r
y <- rnorm(30)
vi <- rexp(30, 3)
refmeta(y, vi)
```

Random integer values simulation

Random integer values simulation.
Random values generation from a Be(a, 1) distribution

Usage

Sample.int(n, size = n, replace = FALSE)
Sample(x, size, replace = FALSE)

Arguments

x
A numeric vector for sampling.

n
This must be an integer value. The function will then draw random integer values from 1:n.

size
The number of integer values to sample.

replace
Do you want to sample with replacement? If yes, set this equal to TRUE.

Details

The function does the same job, up to some level, with R’s built-in function `sample.int`.

Value

A vector with integer values.

Author(s)

Manos Papadakis.

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

See Also

Runif,rbeta1,riag

Examples

```r
x <- Sample.int(10, 1000, replace = TRUE)
Sample(x,length(x))
```

---

Random values generation from a Be(a, 1) distribution

```
  Random values generation from a Be(a, 1) distribution
```

Description

Random values generation from a Be(a, 1) distribution.

Usage

```r
rbeta1(n, a)
```
Random values simulation from the uniform distribution

Arguments

- n: The sample size, a numerical value.
- a: The shape parameter of the beta distribution.

Details

The function generates random values from a $\text{Be}(a, 1)$ distribution.

Value

A vector with the simulated data.

Author(s)

Michail Tsagris.

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

See Also

- kumar.mle
- simplex.mle
- collogitnorm.mle
- propols.reg

Examples

```r
x <- rbeta1(100, 3)
```

Description

Random values simulation from the uniform distribution.

Usage

`Runif(n, min = 0, max = 1)`

Arguments

- n: The number of values to generate.
- min: The lower value of the uniform distribution.
- max: The upper value of the uniform distribution.

Details

This function does the same job as R’s built-in function `runif`. 
Regularised maximum likelihood linear discriminant analysis

Value
A vector with simulated values.

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

See Also
Sample.int, rbeta1, riag

Examples
x <- Runif(1000, 0, 1)

Description
Regularised maximum likelihood linear discriminant analysis.

Usage
reg.mle.lda(xnew, x, ina, lambda)

Arguments
xnew  A numerical vector or a matrix with the new observations, continuous data.
x     A matrix with numerical data.
ina   A numerical vector or factor with consecutive numbers indicating the group to which each observation belongs to.
lambda A vector of regularization values $\lambda$ such as (0, 0.1, 0.2,...).

Details
Regularised maximum likelihood linear discriminant analysis is performed. The function is not extremely fast, yet is pretty fast.

Value
A matrix with the predicted group of each observation in "xnew". Every column corresponds to a $\lambda$ value. If you have just on value of $\lambda$, then you will have one column only.
Repeated measures ANOVA (univariate data) using Hotelling’s T2 test

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

See Also
regmlelda.cv mle.lda, fisher.da, big.knn, weibull.nb

Examples
x <- as.matrix(iris[, 1:4])
ina <- iris[, 5]
a <- reg.mle.lda(x, x, ina, lambda = seq(0, 1, by = 0.1) )

Description
Repeated measures ANOVA (univariate data) using Hotelling’s T_2 test.

Usage
rm.hotel(x, a = 0.05)

Arguments
x A numerical matrix with the repeated measurements. Each column contains the values of the repeated measurements.
a The level of significance, default value is equal to 0.05.

Details
This is a multivariate test for the equality of means of repeated measurements.

Value
A list including:
m The mean vector.
result A vector with the test statistic value, it’s associated p-value, the numerator and denominator degrees of freedom and the critical value.
Sample quantiles and col/row wise quantiles

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

See Also
sp.logiregs, pc.sel

Examples
x <- as.matrix(iris[, 1:4]) ## assume they are repeated measurements
rm.hotel(x)

Description
Sample quantiles and col/row wise quantiles.

Usage
colQuantile(x, probs, parallel = FALSE)
colQuantile(x, probs, parallel = FALSE)
colQuantile(x, probs, parallel = FALSE)
colQuantile(x, probs, parallel = FALSE)
rowQuantile(x, probs, parallel = FALSE)
Quantile(x, probs)

Arguments
x Numeric vector whose sample quantiles are wanted. NA and NaN values are not allowed in numeric vectors. For the col/row versions a numerical matrix.
probs Numeric vector of probabilities with values in [0,1], not missing values. Values up to 2e-14 outside that range are accepted and automatically moved to the nearby endpoint by C++.
parallel Do you want to do it in parallel, for column - row major, in C++? TRUE or FALSE.

Details
This is the same function as R’s built in "quantile" with its default option, type = 7. We have also implemented it in a col/row-wise fashion.
Scaled logistic regression

Value
The function will return a vector of the same mode as the arguments given. NAs will be removed.

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

See Also
trim.mean

Examples

```r
x<-rnorm(1000)
probs<-runif(10)
sum( quantile(x, probs = probs) - Quantile(x, probs) )
```

Description
Scaled logistic regression.

Usage

```r
sclr(y, x, full = FALSE, tol = 1e-07, maxiters = 100)
```

Arguments

- `y` The dependent variable; a numerical vector with two values (0 and 1).
- `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).
- `full` If this is FALSE, the coefficients and the log-likelihood 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.
Score test for overdispersion in Poisson regression

Value

When full is FALSE a list including:

- `theta`: The estimated $\theta$ parameter.
- `be`: The estimated regression coefficients.
- `loglik`: The log-likelihood of the model.
- `iters`: The number of iterations required by Newton-Raphson.

When full is TRUE a list including:

- `info`: The estimated $\theta$, regression coefficients, their standard error, their Wald test statistic and their p-value.
- `loglik`: The log-likelihood.
- `iters`: The number of iterations required by Newton-Raphson.

Author(s)

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

References


See Also

- `propols.reg`

Examples

```r
x <- matrix(rnorm(100 * 2), ncol = 2)
y <- rbinom(100, 1, 0.6)  ## binary logistic regression
a <- sclr(y, x)
```

Score test for overdispersion in Poisson regression

Description

Score test for overdispersion in Poisson regression.

Usage

`overdispreg.test(y, x)`
Arguments
y A vector with count data.
x A numerical matrix with predictor variables.

Details
A score test for overdispersion in Poisson regression is implemented.

Value
A vector with two values. The test statistic and its associated p-value.

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

References

See Also
ztp.reg, censpois.mle, wald.poisrat

Examples
y <- rnbinom(100, 10, 0.4)
x <- rnorm(100)
overdisreg.test(y, x)
Arguments

y  The dependent variable, a numerical vector with numbers.

x  A numerical matrix with the independent variables. We add, internally, the first column of ones.

type  If you want to perform the usual F (or t) test set this equal to "F". For the test based on the partial correlation set this equal to "cor".

Details

This is the same function as R's built in drop1 that it works with the F test or the partial correlation coefficient. For the linear regression model, the Wald test is equivalent to the partial F test. So, instead of performing many regression models with single term deletions we perform one regression model with all variables and compute their Wald test effectively. Note, that this is true, only if the design matrix "x" contains the vectors of ones and in our case this must be, strictly, the first column. The second option is to compute the p-value of the partial correlation.

Value

A matrix with two columns. The test statistic and its associated p-value for each independent variable.

Author(s)

Michail Tsagris.

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

References


See Also

lm.bsreg

Examples

```r
y <- rnorm(150)
x <- as.matrix(iris[, 1:4])
a <- lm(y~., data.frame(x))
drop1(a, test = "F")
lm.drop1(y, x)
```
The skeleton of a Bayesian network produced by the FEDHC algorithm.

**Description**

The skeleton of a Bayesian network produced by the FEDHC algorithm.

**Usage**

```r
fedhc.skel(x, method = "pearson", alpha = 0.05,
          ini.stat = NULL, R = NULL, parallel = FALSE)
```

**Arguments**

- `x`: 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` from the R package Rfast. Note, that for the categorical case data, the numbers must start from 0. No missing data are allowed.

- `method`: If you have continuous data, this "pearson". If you have categorical data though, this must be "cat". In this case, make sure the minimum value of each variable is zero. The function "g2Test" in the R package Rfast and the relevant functions work that way.

- `alpha`: The significance level (suitable values in (0, 1)) for assessing the p-values. Default value is 0.05.

- `ini.stat`: If the initial test statistics (univariate associations) are available, pass them through this parameter.

- `R`: If the correlation matrix is available, pass it here.

- `parallel`: Set this to TRUE for parallel computations.

**Details**

Similar to MMHC and PCHC the first phase consists of a variable selection procedure, the FBED algorithm (Borboudakis and Tsamardinos, 2019).

**Value**

A list including:

- `ini.stat`: The test statistics of the univariate associations.
- `ini.pvalue`: The initial p-values univariate associations.
- `pvalue`: A matrix with the logarithm of the p-values of the updated associations. This final p-value is the maximum p-value among the two p-values in the end.
- `runtime`: The duration of the algorithm.
ntests

The number of tests conducted during each k.

G

The adjacency matrix. A value of 1 in G[i, j] appears in G[j, i] also, indicating that i and j have an edge between them.

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

mmhc.skel, mmpci, mmpci2

Examples

# simulate a dataset with continuous data
x <- matrix( rnorm(200 * 50, 1, 10), nrow = 200 )
a <- fedhc.skel(x)

Skeleton of the MMHC algorithm

The skeleton of a Bayesian network learned with the MMHC algorithm

Description

The skeleton of a Bayesian network learned with the MMHC algorithm.

Usage

mmhc.skel(x, method = "pearson", max_k = 3, alpha = 0.05, ini.stat = NULL, R = NULL, parallel = FALSE)
Arguments

- **x**: 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` from the R package Rfast. Note, that for the categorical case data, the numbers must start from 0. No missing data are allowed.

- **method**: If you have continuous data, this is "pearson". If you have categorical data though, this must be "cat". In this case, make sure the minimum value of each variable is zero. The function "g2Test" in the R package Rfast and the relevant functions work that way.

- **max_k**: The maximum conditioning set to use in the conditional independence test (see Details). Integer, default value is 3.

- **alpha**: The significance level (suitable values in (0, 1)) for assessing the p-values. Default value is 0.05.

- **ini.stat**: If the initial test statistics (univariate associations) are available, pass them through this parameter.

- **R**: If the correlation matrix is available, pass it here.

- **parallel**: Set this to TRUE for parallel computations.

Details

The **max_k** option: the maximum size of the conditioning set to use in the conditioning independence test. Larger values provide more accurate results, at the cost of higher computational times. When the sample size is small (e.g., < 50 observations) the **max_k** parameter should be 3 for example, otherwise the conditional independence test may not be able to provide reliable results.

Value

A list including:

- **ini.stat**: The test statistics of the univariate associations.

- **ini.pvalue**: The initial p-values univariate associations.

- **pvalue**: A matrix with the logarithm of the p-values of the updated associations. This final p-value is the maximum p-value among the two p-values in the end.

- **runtime**: The duration of the algorithm.

- **ntests**: 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.

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

fedhc.skel, mmpc, mmpc2

Examples

# simulate a dataset with continuous data
x <- matrix( rnorm(300 * 30, 1, 100), nrow = 300 )
a <- mmhc.skel(x)

Description

Split the matrix in lower, upper triangular and diagonal.

Usage

lud(x)

Arguments

x A matrix with data.

Value

A list with 3 fields:

lower The lower triangular of argument "x".
upper The upper triangular of argument "x".
diagonal The diagonal elements.

Author(s)

Manos Papadakis.

R implementation and documentation: Manos Papadakis <papadakm95@gmail.com>.
The k-NN algorithm for really large scale data

See Also
Intersect

Examples

```r
x <- matrix(runif(10*10),10,10)
b<-lud(x)
```

Description
The k-NN algorithm for really large scale data.

Usage
```r
big.knn(xnew, y, x, k = 2:100, type = "C")
```

Arguments

- `xnew` A matrix with new data, new predictor variables whose response variable must be predicted.
- `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.
- `type` If your response variable `y` is numerical data, then this should be "R" (regression). If `y` is in general categorical set this argument to "C" (classification).

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 allows for the Euclidean distance only.
Tobit regression

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

References

See Also
bigknn.cv, reg.mle.lda, multinom.reg

Examples
```r
x <- as.matrix(iris[, 1:4])
mod <- big.knn(xnew = x, y = iris[, 5], x = x, k = c(6, 7) )
```

Description
Tobit regression.

Usage
```r
tobit.reg(y, x, ylow = 0, full = FALSE, tol = 1e-07, maxiters = 100)
```

Arguments
- **y**: The dependent variable; a numerical vector with values.
- **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).
- **ylow**: The lowest value below which nothing is observed. The cut-off value.
- **full**: If this is FALSE, the coefficients and the log-likelihood 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 tobit regression model is fitted.

Value

When full is FALSE a list including:

- `be`  The estimated regression coefficients.
- `s`    The estimated scale parameter.
- `loglik` The log-likelihood of the model.
- `iters` The number of iterations required by Newton-Raphson.

When full is TRUE a list including:

- `info` The estimated `theta`, regression coefficients, their standard error, their Wald test statistic and their p-value.
- `loglik` The log-likelihood.
- `iters` The number of iterations required by Newton-Raphson.

Author(s)

Michail Tsagris.

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

References


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

See Also

`hp.reg, ztp.reg, censweibull.mle, censpois.mle`

Examples

```r
x <- rnorm(100)
y <- rnorm(100)
y[y < 0] <- 0
a <- tobit.reg(y, x, full = TRUE)
```
Trimmed mean

Description

Trimmed mean.

Usage

trim.mean(x, a = 0.05)
colTrimMean(x, a = 0.05, parallel=FALSE)
## S3 method for class 'matrix'
colTrimMean(x,a = 0.05,parallel=FALSE)
## S3 method for class 'data.frame'
colTrimMean(x,a = 0.05,parallel=FALSE)
rowTrimMean(x, a = 0.05, parallel=FALSE)

Arguments

x      A numerical vector or a numerical matrix.

a      A number in (0, 1), the proportion of data to trim.

parallel Run the algorithm parallel in C++.

Details

The trimmed mean is computed. The lower and upper a% of the data are removed and the mean is calculated using the rest of the data.

Value

The trimmed mean.

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

Quantile
Variable selection using the PC-simple algorithm

Examples

\[
x <- \text{rnorm}(100, 1, 1)
ALL.EQUAL(\text{trim.mean}(x, 0.05), \text{mean}(x, 0.05))
\]

\[
x<-\text{matrix}(x,10,10)
\]

\[
col\text{TrimMean}(x,0.05)
\]

\[
row\text{TrimMean}(x,0.05)
\]

Description

Variable selection using the PC-simple algorithm.

Usage

\[
\text{pc.sel}(y, x, \text{ystand} = \text{TRUE}, \text{xstand} = \text{TRUE}, \text{alpha} = 0.05)
\]

Arguments

- \text{y} 
  A numerical vector with continuous data.
- \text{x} 
  A matrix with numerical data; the independent variables, of which some will probably be selected.
- \text{ystand} 
  If this is \text{TRUE} the response variable is centered. The mean is subtracted from every value.
- \text{xstand} 
  If this is \text{TRUE} the independent variables are standardised.
- \text{alpha} 
  The significance level.

Details

Variable selection for continuous data only is performed using the PC-simple algorithm (Buhlmann, Kalisch and Maathuis, 2010). The PC algorithm used to infer the skeleton of a Bayesian Network has been adopted in the context of variable selection. In other words, the PC algorithm is used for a single node.

Value

A list including:

- \text{vars} 
  A vector with the selected variables.
- \text{n.tests} 
  The number of tests performed.
- \text{runtime} 
  The runtime of the algorithm.
Wald confidence interval for the ratio of two Poisson variables

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

References

See Also
pc.skel, omp

Examples
y <- rnorm(100)
x <- matrix( rnorm(100 * 50), ncol = 50)
a <- pc.sel(y, x)

Description
Wald confidence interval for the ratio of two Poisson variables.

Usage
wald.poisrat(x, y, alpha = 0.05)
col.waldpoisrat(x, y, alpha = 0.05)

Arguments
x A numeric vector or a matrix with count data.
y A numeric vector or a matrix with count data.
alpha The 1 - confidence level. The default value is 0.05.

Details
wald confidence interval for the ratio of two Poisson means is/are calculated.

Value
For the wald.poisrat a vector with three elements, the ratio and the lower and upper confidence interval limits. For the col.waldpoisrat a matrix with three columns, the ratio and the lower and upper confidence interval limits.
Walter’s confidence interval for the ratio of two binomial variables (and the relative risk)

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

References

See Also
censpois.mle,

Examples
x <- rpois(100, 10)
y <- rpois(100, 10)
wald.poisrat(x, y)

Walter’s confidence interval for the ratio of two binomial variables (and the relative risk)

Description
Walter’s confidence interval for the ratio of two binomial variables (and the relative risk).

Usage
walter.ci(x1, x2, n1, n2, a = 0.05)

Arguments
x1 An integer number, greater than or equal to zero.
x2 A second integer number, greater than or equal to zero.
n1 An integer number, greater than or equal to x1.
n2 A second integer number, greater than or equal to x2.
a The significance level. The produced confidence interval has a confidence level equal to 1-a.

Details
This calculates a (1-a)% confidence interval for the ratio of two binomial variables (and hence for the relative risk) using Walter’s suggestion (Walter, 1975). That is, to add 0.5 in each number. This not only overcomes the problem of zero values, but produces intervals that are more accurate than the classical asymptotic confidence interval (Alharbi and Tsagris, 2018).
Value

A list including:

\textit{rat} \quad \text{The ratio of the two binomial distributions.}

\textit{ci} \quad \text{Walter's confidence interval.}

Author(s)

Michail Tsagris.

R implementation and documentation: Michail Tsagris \texttt{<mtsagris@uoc.gr>}.

References


See Also

\texttt{mle.lda,welch.tests}

Examples

\begin{verbatim}
x1 <- rbinom(1, 20, 0.7)
x2 <- rbinom(1, 30, 0.6)
n1 <- 20
n2 <- 30
walter.ci(x1,x2,n1,n2)
\end{verbatim}
Zero inflated Gamma regression

Arguments

- **y**: The dependent variable, a numerical vector with numbers, zeros and higher.
- **x**: A numerical matrix with the independent variables. We add, internally, the first column of ones.
- **full**: If this is FALSE, the coefficients and the log-likelihood will be returned only. If this is TRUE, more information is returned.
- **tol**: The tolerance value to terminate the Newton-Raphson algorithm.
- **maxiters**: The maximum number of iterations that can take place in each regression.

Details

Two regression models are fitted, a binary logistic regression and a Gamma regression model to the non-zero responses.

Value

Depending on whether "full" is TRUE or not different outputs are returned. In general, the regression coefficients, the iterations required by Newton-Raphson and the deviances are returned. If full is TRUE, a matrix with their standard errors and the Wald test statistics is returned as well.

Author(s)

Michail Tsagris.

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

References


See Also

zigamma.mle, ztp.reg

Examples

```r
y <- rgamma(100, 4, 1)
y[sample(100, 10)] <- 0
x <- rnorm(100)
a <- zigamma.reg(y, x)
```
Zero truncated Poisson regression

Description

Zero truncated Poisson regression.

Usage

ztp.reg(y, x, full = FALSE, tol = 1e-07, maxiters = 100)

Arguments

y The dependent variable, a numerical vector with integer valued numbers.
x A matrix or a data.frame with the independent variables.
full If you want full information (standard errors, Walt test statistics and p-values of the regression coefficients) set this equal to TRUE.
tol The tolerance value required by the Newton-Raphson to stop.
maxiters The maximum iterations allowed.

Details

A zero truncated poisson regression model is fitted.

Value

A list including:

be The regression coefficients if "full" was set to FALSE.
info This is returned only if "full" was set to TRUE. It is a matrix with the regression coefficients, their standard errors, Walt test statistics and p-values.
loglik The loglikelihood of the regression model.
iter The iterations required by the Newton-Raphson.

Author(s)

Michail Tsagris.

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

See Also

bic.regs
Examples

```r
y <- rpois(100, 5)
y[y == 0] <- 1
x <- matrix(rnorm(100 * 5), ncol = 5)
mod <- ztp.reg(y, x)
```
Index

* Benchmark - Measure time
  Benchmark - Measure time, 10
* Check if a matrix is Lower or Upper triangular
  Check if a matrix is Lower or Upper triangular, 17
* Chrono Library
  Benchmark - Measure time, 10
* Column wise of grouping variables
  Column-wise summary statistics with grouping variables, 26
* Correlation
  Correlation significance testing using Fisher's z-transformation, 30
* Feature Selection
  Max-Min Parents and Children variable selection algorithm for continuous responses, 77
* Intersect
  Intersect, 58
* Merge 2 sorted vectors in 1 sorted vector
  Merge 2 sorted vectors in 1 sorted vector, 83
* Multinomial distribution
  Multinomial regression, 98
* Multiple Feature Signatures
  Max-Min Parents and Children variable selection algorithm for continuous responses, 77
* Sample Quantiles and col - row wise Quantiles
  Sample quantiles and col/row wise quantiles, 121
* Symmetric matrix
  Check whether a square matrix is skew-symmetric, 18
* Variable Selection
  Max-Min Parents and Children variable selection algorithm for continuous responses, 77

* regression
  Multinomial regression, 98
  .lm.fit, 54

Add many single terms to a model, 5
add.term(Add many single terms to a model), 5
allbetas, 30
Angular Gaussian random values simulation, 6
Anova for circular data, 7
Backward selection with the F test or the partial correlation coefficient, 9
benchmark(Benchmark - Measure time), 10
benchmark (Benchmark - Measure time), 10
beroulli.nb, 67, 110
beroulli.nb (Naive Bayes classifier for binary (Bernoulli) data), 99
beroullinb.pred, 100
beroullinb.pred (Prediction with naive Bayes classifier for binary (Bernoulli) data), 110
beta.nb (Naive Bayes classifiers), 100
betanb.pred (Prediction with some naive Bayes classifiers), 111
BIC of many simple univariate regressions, 11
bic.regs, 6, 28, 31, 38, 46, 66, 69, 71, 75, 76, 116, 139
bic.regs (BIC of many simple univariate regressions), 11
big.knn, 33, 37, 42, 82, 101, 120
big.knn (The k-NN algorithm for really large scale data), 130
bigknn.cv, 34, 37, 67, 70, 131
bigknn.cv (Cross-validation for the k-NN algorithm for really large scale data), 32
binom.reg, 104
binom.reg (Binomial regression), 12
Binomial regression, 12
boot.hotel2 (Bootstrap James and Hotelling test for 2 independent sample mean vectors), 13
boot.james (Bootstrap James and Hotelling test for 2 independent sample mean vectors), 13
boot.student2, 65, 106
boot.student2 (Bootstrap Student’s t-test for 2 independent samples), 14
boot.ttest1, 65
boot.ttest1 (One sample bootstrap t-test for a vector), 105
Bootstrap James and Hotelling test for 2 independent sample mean vectors, 13
Bootstrap Student’s t-test for 2 independent samples, 14
cauchy.nb (Naive Bayes classifiers), 100
cauchy0.mle (MLE of the Cauchy and generalised normal distributions with zero location), 89
cauhvnb.pred (Prediction with some naive Bayes classifiers), 111
cbern.mle (MLE of distributions defined for proportions), 85
Censored Weibull regression model, 16
censpois.mle, 23, 74, 91, 98, 124, 132, 136
censpois.mle (MLE of the left censored Poisson distribution), 92
censweib.reg (Censored Weibull regression model), 16
censweibull.mle, 17, 85, 86, 90, 93, 132
censweibull.mle (MLE of the censored Weibull distribution), 90
Check if a matrix is Lower or Upper triangular, 17
Check whether a square matrix is skew-symmetric, 18
cholesky, 18
circ.cor1, 7, 94
circ.cor1 (Circular correlations between two circular variables), 19
circ.cors1, 7, 31
circ.cors1 (Circular correlations between two circular variables), 19
Circular correlations between two circular variables, 19
cls, 55
cls (Constrained least squares), 28
cluster_lm, 43, 44, 50, 55
cluster_lm (Linear regression with clustered data), 62
col.waldpoisrat (Wald confidence interval for the ratio of two Poisson variables), 135
colaccs, 70
colaccs (Many binary classification metrics), 66
colbeta.mle (Column-wise MLE of some univariate distributions), 22
colborel.mle (Column-wise MLE of some univariate distributions), 22
colcauchy.mle (Column-wise MLE of some univariate distributions), 22
colfbscores (Many binary classification metrics), 66
colfmis (Many binary classification metrics), 66
colfscores (Many binary classification metrics), 66
colGroup (Column-wise summary statistics with grouping variables), 26
colhalfnorm.mle (Column-wise MLE of some univariate distributions), 22
coljack.means (Column and row-wise jackknife sample means), 20
collogitnorm.mle, 118
collogitnorm.mle (Column-wise MLE of some univariate distributions), 22
collognorm.mle, 24, 29
collognorm.mle (Column-wise MLE of some
univariate distributions), 22
colmaes (Many metrics for a continuous response variable), 69
colmeansvars, 25, 60
colmeansvars (Column-wise means and variances), 21
colmses, 67
colmses (Many metrics for a continuous response variable), 69
colordinal.mle (Column-wise MLE of some univariate distributions), 22
colpinal1 (Conditional least-squares estimate for Poisson INAR(1) models), 27
colpkl (Many metrics for a continuous response variable), 69
colprecs (Many binary classification metrics), 66
colQuantile, 26
colQuantile (Sample quantiles and col/row wise quantiles), 121
colsens (Many binary classification metrics), 66
colspecs (Many binary classification metrics), 66
colspml.mle, 7, 87
colspml.mle (Column-wise MLE of the angular Gaussian distribution), 23
colTrimMean (Trimmed mean), 133
colukl (Many metrics for a continuous response variable), 69
Column and row-wise jackknife sample means, 20
Column-wise means and variances, 21
Column-wise MLE of some univariate distributions, 22
Column-wise MLE of the angular Gaussian distribution, 23
Column-wise pooled variances across groups, 25
Column-wise summary statistics with grouping variables, 26
colunitweibull.mle (Column-wise MLE of some univariate distributions), 22
Constrained least squares, 28
Contour plots of some bivariate distributions, 29
cor_test, 39, 40, 55, 57, 58
cor_test (Correlation significance testing using Fisher's z-transformation), 30
cora, 18
Correlation significance testing using Fisher's z-transformation, 30
cova, 18
covar, 39, 40, 44, 50, 57, 58
covar (Covariance between a variable and a matrix of variables), 31
Covariance between a variable and a matrix of variables, 31
covdist, 57, 58
covdist (Distance between two covariance matrices), 38
covequal, 39, 58
covequal (Hypothesis tests for equality of a covariance matrix), 56
covlikel, 39, 57
covlikel (Hypothesis tests for equality of multiple covariance matrices), 57
covmtest (Hypothesis tests for equality of multiple covariance matrices), 57
Cross-validation for the k-NN algorithm for really large scale data, 32
Cross-validation for the multinomial regression, 33
Cross-validation for the naive Bayes classifiers, 35
Cross-validation for the regularised maximum likelihood linear discriminant analysis, 36
data.frame.to_matrix, 126, 128
dcora (Distance correlation matrix), 39
den.contours (Contour plots of some bivariate distributions), 29
depth.mahala (Mahalanobis depth), 63
Diagonal values of the Hat matrix, 37
diffic (Item difficulty and discrimination), 59
discrim (Item difficulty and discrimination), 59
Distance between two covariance matrices, 38
Distance correlation matrix, 39
dropl, 125

embed.circaov (Anova for circular data), 7
Empirical entropy, 40
empirical.entropy (Empirical entropy), 40

fbed.reg, 81, 99
fbed.reg (Forward Backward Early Dropping selection regression), 45
fe.lmfit, 44, 50, 63
fe.lmfit (Fixed effects regression), 42
fedhc.skel (Skeleton of the FEDHC algorithm), 126
fipois.reg, 27, 43, 50
fipois.reg (Fixed intercepts Poisson regression), 43
Fisher’s linear discriminant analysis, 41
fisher.da, 82, 120
fisher.da (Fisher’s linear discriminant analysis), 41
Fixed effects regression, 42
Fixed intercepts Poisson regression, 43
Forward Backward Early Dropping selection regression, 45
fp (Fractional polynomial regression with one independent variable), 46
Fractional polynomial regression with one independent variable, 46

Gamma regression with a log-link, 48
gammapois.mle, 23, 24, 95, 98
gammapois.mle (MLE of the gamma-Poisson distribution), 91
gammareg, 69
gammareg (Gamma regression with a log-link), 48
gammaregs, 49, 73
gammaregs (Many Gamma regressions), 68

GEE Gaussian regression, 49
gee.reg, 10, 28, 38, 43, 44, 46, 52, 63
gee.reg (GEE Gaussian regression), 49
gnormal0.mle (MLE of the Cauchy and generalised normal distributions with zero location), 89
Gumbel regression, 50
gumbel.reg, 17
gumbel.reg (Gumbel regression), 50

halfcauchy.mle, 29
halfcauchy.mle (MLE of continuous univariate distributions defined on the positive line), 84
hcf.circaov (Anova for circular data), 7
Hellinger distance based regression for count data, 51
Hellinger distance based univariate regression for proportions, 53
hellinger.countreg (Hellinger distance based regression for count data), 51
het.circaov (Anova for circular data), 7
het.lmfit (Heteroscedastic linear models for large scale data), 54
Heteroscedastic linear models for large scale data, 54
hp.reg, 13, 27, 132
hp.reg (Hurdle-Poisson regression), 55
Hurdle-Poisson regression, 55
Hypothesis tests for equality of a covariance matrix, 56
Hypothesis tests for equality of multiple covariance matrices, 57

Intersect, 18, 58, 130
intersect, 59
is.lower.tri, 83
is.lower.tri (Check if a matrix is Lower or Upper triangular), 17
is.skew.symmetric (Check whether a square matrix is skew-symmetric), 18
is.upper.tri, 83
is.upper.tri (Check if a matrix is Lower or Upper triangular), 17
Item difficulty and discrimination, 59

jack.mean, 106, 110
jack.mean (Jackknife sample mean), 60
Jackknife sample mean, 60

Kaplan-Meier estimate of a survival function, 61
km, 17, 91, 93
km (Kaplan-Meier estimate of a survival function), 61
kumar.mle, 54, 105, 118
kumar.mle (MLE of distributions defined for proportions), 85

laplace.nb (Naive Bayes classifiers), 100
laplacenb.pred (Prediction with some naive Bayes classifiers), 111

leverage, 109
leverage (Diagonal values of the Hat matrix), 37
Linear regression with clustered data, 62

lm, 54
lm.bsreg, 125
lm.bsreg (Backward selection with the F test or the partial correlation coefficient), 9
lm.drop1, 10, 55, 109
lm.drop1 (Single terms deletion hypothesis testing in a linear regression model), 124

lm.fit, 54
lm.parboot, 55
lm.parboot (Parametric bootstrap for linear regression model), 108

logiquant.regs, 6, 46, 66, 99
logiquant.regs (Many simple quantile regressions using logistic regressions), 74

logistic_only, 11, 47
logitnorm.nb (Naive Bayes classifiers), 100
logitnormnb.pred (Prediction with some naive Bayes classifiers), 111

lr.circaov (Anova for circular data), 7

lud (Split the matrix in lower, upper triangular and diagonal), 129

Mahalanobis depth, 63
Many 2 sample student's t-tests, 64
Many approximate simple logistic regressions, 65
Many binary classification metrics, 66
Many Gamma regressions, 68
Many metrics for a continuous response variable, 69
Many negative binomial regressions, 70
Many score based regressions with multiple response variables and a single predictor variable, 71
Many score based zero inflated Poisson regressions, 73
Many simple quantile regressions using logistic regressions, 74
Many simple Weibull regressions, 75
Many Welch tests, 76
Max-Min Parents and Children variable selection algorithm for continuous responses, 77
Max-Min Parents and Children variable selection algorithm for non continuous responses, 79
Maximum likelihood linear discriminant analysis, 82
mci (Monte Carlo integration with a normal distribution), 96
Merge (Merge 2 sorted vectors in 1 sorted vector), 83

MLE of continuous univariate distributions defined on the positive line, 84
MLE of distributions defined for proportions, 85
MLE of some circular distributions with multiple samples, 86
MLE of some truncated distributions, 88
MLE of the Cauchy and generalised normal distributions with zero location, 89
MLE of the censored Weibull distribution, 90
MLE of the gamma-Poisson distribution, 91
MLE of the left censored Poisson distribution, 92
MLE of the Purkayastha distribution, 93
MLE of the zero inflated Gamma and Weibull distributions, 95
mle.lda, 34, 36, 37, 42, 101, 120, 137
mle.lda (Maximum likelihood linear discriminant analysis), 82
mmhc.skel, 127
mmhc.skel (Skeleton of the MMHC algorithm), 127
mmpc, 70, 79, 81, 109, 127, 129
mmpc (Max–Min Parents and Children variable selection algorithm for continuous responses), 77
mmpc2, 10, 108, 127, 129
mmpc2 (Max–Min Parents and Children variable selection algorithm for non continuous responses), 79
Monte Carlo integration with a normal distribution, 96
Moran's I measure of spatial autocorrelation, 97
moranI, 110
moranI (Moran's I measure of spatial autocorrelation), 97
multinom.reg, 36, 131
multinom.reg (Multinomial regression), 98
Multinomial regression, 98
multinomreg.cv, 33
multinomreg.cv (Cross-validation for the multinomial regression), 33
multisplm.mle (MLE of some circular distributions with multiple samples), 86
multivm.mle, 8
multivm.mle (MLE of some circular distributions with multiple samples), 86
mv.score.betaregs (Many score based regressions with multiple response variables and a single predictor variable), 71
mv.score.glms (Many score based regressions with multiple response variables and a single predictor variable), 71
mv.score.invgaussregs (Many score based regressions with multiple response variables and a single predictor variable), 71
mv.score.weibregs (Many score based regressions with multiple response variables and a single predictor variable), 71
Naive Bayes classifier for binary (Bernoulli) data, 99
Naive Bayes classifiers, 100
Naive Bayes classifiers for circular data, 102
nb.cv, 100, 101, 103, 110, 112, 113
nb.cv (Cross-validation for the naive Bayes classifiers), 35
Negative binomial regression, 103
negbin.reg, 13, 51, 52, 56, 71
negbin.reg (Negative binomial regression), 103
negbin.regs (Many negative binomial regressions), 70
Non linear least squares regression for percentages or proportions, 104
normlog.nb (Naive Bayes classifiers), 100
normlognb.pred (Prediction with some naive Bayes classifiers), 111
omp, 135
omp2 (Orthogonal matching pursuit variable selection), 106
One sample bootstrap t-test for a vector, 105
Orthogonal matching pursuit variable selection, 106
overdispreg.test (Score test for overdispersion in Poisson regression), 123

Parametric bootstrap for linear regression model, 108
pc.sel, 10, 70, 77, 81, 108, 109, 121
pc.sel (Variable selection using the PC-simple algorithm), 134
pc.skel, 135
pca, 115
pca (Principal component analysis), 113
pca (Principal components regression), 114
perm.ttest2, 106
perm.ttest2 (Permutation t-test for 2 independent samples), 109
Permutation t-test for 2 independent samples, 109
pinar1 (Conditional least-squares estimate for Poisson INAR(1) models), 27
poisson_only, 11, 47
pooled.colVars, 22
pooled.colVars (Column-wise pooled variances across groups), 25
powerlaw.mle (MLE of continuous univariate distributions defined on the positive line), 84
Prediction with naive Bayes classifier for binary (Bernoulli) data, 110
Prediction with some naive Bayes classifiers, 111
Prediction with some naive Bayes classifiers for circular data, 112
pretty, 41
Principal component analysis, 113
Principal components regression, 114
print.benchmark (Benchmark - Measure time), 10
prophelling.reg, 105
prophelling.reg (Hellinger distance based univariate regression for proportions), 53
propols.reg, 54, 118, 123
propols.reg (Non linear least squares regression for percentages or proportions), 104
purka.mle, 87, 89
purka.mle (MLE of the Purkayastha distribution), 93
Quantile, 11, 26, 41, 60, 133
Quantile (Sample quantiles and col/row wise quantiles), 121
Random effects meta analysis, 115
Random integer values simulation, 116
Random values generation from a Be(a, 1) distribution, 117
Random values simulation from the uniform distribution, 118
rbeta, 96, 117, 119
rbeta (Random values generation from a Be(a, 1) distribution), 117
refmeta (Random effects meta analysis), 115
reg.mle.lda, 34, 36, 37, 42, 82, 114, 131
reg.mle.lda (Regularised maximum likelihood linear discriminant analysis), 119
regmlelda.cv, 33, 120
regmlelda.cv (Cross-validation for the regularised maximum likelihood linear discriminant analysis), 36
Repeated measures ANOVA (univariate data) using Hotelling’s T2 test, 120
Rfast2-package, 4
riag, 96, 117, 119
riag (Angular Gaussian random values simulation), 6
rm.hotel (Repeated measures ANOVA (univariate data) using Hotelling’s T2 test), 120
rowjack.means (Column and row-wise jackknife sample means), 20
rowQuantile, 26
rowQuantile (Sample quantiles and col/row wise quantiles), 121
rowTrimMean (Trimmed mean), 133
Runif, 117
Runif (Random values simulation from the uniform distribution), 118
runif, 118
Sample (Random integer values simulation), 116
Sample quantiles and col/row wise quantiles, 121
Sample.int, 119
sample.int, 117
Scaled logistic regression, 122
sclr (Scaled logistic regression), 122
Score test for overdispersion in Poisson regression, 123
score.zipregs, 71, 73
score.zipregs (Many score based zero inflated Poisson regressions), 73
simplex.mle, 54, 105, 118
simplex.mle (MLE of distributions defined for proportions), 85
Single terms deletion hypothesis testing in a linear regression model, 124
Skeleton of the FEDHC algorithm, 126
Skeleton of the MMHC algorithm, 127
sp.logiregs, 6, 62, 75, 77, 121
sp.logiregs (Many approximate simple logistic regressions), 65
Split the matrix in lower, upper triangular and diagonal, 129
spml.nb (Naive Bayes classifiers for circular data), 102
spml.reg, 20
spmlnb.pred (Prediction with some naive Bayes classifiers for circular data), 112
stud.ttests (Many 2 sample student’s t-tests), 64

The k-NN algorithm for really large scale data, 130
Tobit regression, 131
tobit.reg (Tobit regression), 131
trim.mean, 11, 14, 15, 21, 61, 64, 110, 122
trim.mean (Trimmed mean), 133
Trimmed mean, 133

trunccauchy.mle (MLE of some truncated distributions), 88
truncexpmle (MLE of some truncated distributions), 88
unitweibull.mle (MLE of distributions defined for proportions), 85
univglm, 30
Variable selection using the PC-simple algorithm, 134
vm.nb, 8, 36, 101, 113
vm.nb (Naive Bayes classifiers for circular data), 102
vmnb.pred, 36, 103, 112
vmnb.pred (Prediction with some naive Bayes classifiers for circular data), 112
Wald confidence interval for the ratio of two Poisson variables, 135
wald.poisrat, 124
wald.poisrat (Wald confidence interval for the ratio of two Poisson variables), 135
Walter’s confidence interval for the ratio of two binomial variables (and the relative risk), 136
walter.ci (Walter’s confidence interval for the ratio of two binomial variables (and the relative risk)), 136
weib.reg, 73
weib.reg (Many simple Weibull regressions), 75
weibull.nb, 36, 37, 42, 82, 103, 112, 120
weibull.nb (Naive Bayes classifiers), 100
weibullnb.pred, 36, 101, 113
weibullnb.pred (Prediction with some naive Bayes classifiers), 111
welch.tests, 14, 15, 21, 44, 50, 61, 64, 106, 137
welch.tests (Many Welch tests), 76
Zero inflated Gamma regression, 137
Zero truncated Poisson regression, 139
zigamma.mle, 49, 85, 86, 92, 138
zigamma.mle (MLE of the zero inflated Gamma and Weibull distributions), 95
zigamma.reg, 95
zigamma.reg (Zero inflated Gamma regression), 137
zil.mle (MLE of distributions defined for proportions), 85
ziweibull.mle (MLE of the zero inflated Gamma and Weibull distributions), 95
ztp.reg, 13, 28, 38, 51, 56, 71, 74, 104, 124, 132, 138
ztp.reg (Zero truncated Poisson regression), 139