Package 'OBRE'

January 20, 2025

Title Optimal B-Robust Estimator Tools

Date 2023-06-07

Version 0.2-0

Author Andrea Riboldi [aut],
Ivan Luciano Danesi [aut],
Fabio Piacenza [aut, cre],
Oliver Kuehnle [aut],
Davide Di Vincenzo [aut],
Ruben Ciaponi [ctb],
Stephen Allen [ctb],
Novella Saccenti [ctb],
Annarita Filippi [ctb]

Maintainer Fabio Piacenza <fabio.piacenza@unicredit.eu>

Description An implementation for computing Optimal B-Robust Estimators of two-parameter distribution. The procedure is composed of some equations that are evaluated alternatively until the solution is reached. Some tools for analyzing the estimates are included. The most relevant is covariance matrix computation using a closed formula.

Depends R (>= 4.0.0), pracma(>= 1.7.3)

License GPL (>= 3)

Encoding UTF-8

ByteCompile true

RoxygenNote 7.2.3

Imports methods

NeedsCompilation no

Repository CRAN

Date/Publication 2023-06-09 16:20:13 UTC

Contents

`densityExpressions` *Distributions formulas for OBRE*

Description

Function containing expressions of density and cumulative functions, plus the first and second derivatives.

Usage

```
densityExpressions(strDistribution = "normal", eDensityFun = NA)
```

Arguments

strDistribution	Distribution input between "normal" (Normal distribution), "logNormal" (logNormal distribution), "weibull" (Weibull distribution), "logLogistic" (logLogistic distribution), "gpd2" (Generalized Pareto Distribution with two parameters) or "custom" if the distribution is written by the user.
eDensityFun	The density of a two parameters distribution. This should be an expression object, the two parameters should be called "nTheta1" and "nTheta2", the data "nvData" and its formula should be derivable

Value

Returns list containing all the symbolic functions.

Examples

```
# Generates the Normal distribution input for OBRE
distrForOBRE <- densityExpressions(strDistribution = "normal")
# The same result can be generated by inserting manually the formula
distrForOBRE <- densityExpressions(strDistribution = "custom",
eDensityFun = expression((exp( -((nvData - nTheta1)^2) / (2 * nTheta2^2)) /
(sqrt(2 * pi) * nTheta2))))
```

---

fisherEl11Part1 *Part 1 of element [1, 1] for Fisher Information matrix*

Description

Function computing part 1 of element [1, 1] for Fisher Information matrix computation. The Fisher Information matrix is splitted in the four elements ([1, 1], [1, 2], [2, 1], [2, 2]). Each element is split in part 1 and part 2

Usage

```
fisherEl11Part1(nvData, nTheta1, nTheta2, lDensityExpr)
```

Arguments

nvData	The vector of data.
nTheta1	The first parameter.
nTheta2	The second parameter.
lDensityExpr	List of symbolic expressions of density, cumulative and derivatives.

fisherEl11Part2 *Part 2 of element [1, 1] for Fisher Information matrix*

Description

Function computing part 2 of element [1, 1] for Fisher Information matrix computation. The Fisher Information matrix is splitted in the four elements ([1, 1], [1, 2], [2, 1], [2, 2]). Each element is split in part 1 and part 2

Usage

```
fisherEl11Part2(nvData, nTheta1, nTheta2, lDensityExpr)
```

Arguments

nvData	The vector of data.
nTheta1	The first parameter.
nTheta2	The second parameter.
lDensityExpr	List of symbolic expressions of density, cumulative and derivatives.

fisherEl12Part1 *Part 1 of element [1, 2] for Fisher Information matrix*

Description

Function computing part 1 of element [1, 2] for Fisher Information matrix computation. The Fisher Information matrix is splitted in the four elements ([1, 1], [1, 2], [2, 1], [2, 2]). Each element is split in part 1 and part 2

Usage

```
fisherEl12Part1(nvData, nTheta1, nTheta2, lDensityExpr)
```

Arguments

nvData	The vector of data.
nTheta1	The first parameter.
nTheta2	The second parameter.
lDensityExpr	List of symbolic expressions of density, cumulative and derivatives.

`fisherEl12Part2` *Part 2 of element [1, 1] for Fisher Information matrix*

Description

Function computing part 2 of element [1, 1] for Fisher Information matrix computation. The Fisher Information matrix is splitted in the four elements ([1, 1], [1, 2], [2, 1], [2, 2]). Each element is split in part 1 and part 2

Usage

```
fisherEl12Part2(nvData, nTheta1, nTheta2, lDensityExpr)
```

Arguments

`nvData`	The vector of data.
`nTheta1`	The first parameter.
`nTheta2`	The second parameter.
`lDensityExpr`	List of symbolic expressions of density, cumulative and derivatives.

`fisherEl22Part1` *Part 1 of element [2, 2] for Fisher Information matrix*

Description

Function computing part 1 of element [2, 2] for Fisher Information matrix computation. The Fisher Information matrix is splitted in the four elements ([1, 1], [1, 2], [2, 1], [2, 2]). Each element is split in part 1 and part 2

Usage

```
fisherEl22Part1(nvData, nTheta1, nTheta2, lDensityExpr)
```

Arguments

`nvData`	The vector of data.
`nTheta1`	The first parameter.
`nTheta2`	The second parameter.
`lDensityExpr`	List of symbolic expressions of density, cumulative and derivatives.

fisherEl22Part2 *Part 2 of element [2, 2] for Fisher Information matrix*

Description

Function computing part 2 of element [2, 2] for Fisher Information matrix computation. The Fisher Information matrix is splitted in the four elements ([1, 1], [1, 2], [2, 1], [2, 2]). Each element is split in part 1 and part 2

Usage

```
fisherEl22Part2(nvData, nTheta1, nTheta2, lDensityExpr)
```

Arguments

nvData	The vector of data.
nTheta1	The first parameter.
nTheta2	The second parameter.
lDensityExpr	List of symbolic expressions of density, cumulative and derivatives.

matFisherComputation *Fisher information matrix*

Description

Function calculating the Fisher information matrix.

Usage

```
matFisherComputation(nTheta1, nTheta2, lDensityExpr)
```

Arguments

nTheta1	First parameter.
nTheta2	Second parameter.
lDensityExpr	List of symbolic expressions of density, cumulative and derivatives.

Value

The Fisher information matrix.

MLE *Numerical Maximum Likelihood Estimator*

Description

The parameters Maximum Likelihood Estimation is obtained by numerical optimization.

Usage

```
MLE(nvData, strDistribution, lDensityExpr)
```

Arguments

nvData	The vector of the data.
strDistribution	The distribution name.
lDensityExpr	The distribution expression,

Value

A list with distribution name, distribution parameters, value of the objective function corresponding to the parameters, additional information returned by the optimizer, convergence of the algorithm.

NlLike *Negative Log-Likelihood*

Description

The function compute the Negative Log-Likelihood value that has to be used for optimization in MLE function.

Usage

```
NlLike(nvTheta, nvData, lDensityExpr)
```

Arguments

nvTheta	Parameters of the distribution.
nvData	The vector of the data.
lDensityExpr	The distribution density espressions.

Value

Negative log likelihood value.

OBRE *Optimal B-Robust Estimator*

Description

Function for obtaining the Optimal B-Robust Estimates starting by a vector of data and a two parameters distribution.

Usage

```
OBRE(
  nvData,
  strDistribution,
  nCParOBRE,
 dfParOBRE = data.frame(nEta = 1e-06, nMaxIterLoopWc = 10, nMaxIterLoopA = 10, nRelTol =
    0.001, nAbsTol = 0.5, stringsAsFactors = FALSE),
  nTheta1Init = NA,
  nTheta2Init = NA,
  eDensityFun = NA
)
```

Arguments

nvData	The vector of data.
strDistribution	The distribution name between "normal" (Normal distribution), "logNormal" (logNormal distribution), "weibull" (Weibull distribution), "logLogistic" (logLogistic distribution), "gpd2" (Generalized Pareto Distribution with two parameters) or "custom" if the distribution is written by the user as an input of "eDensityFun" parameter. Alternatively, the input of "strDistribution" can be an object of class "OBREdist", obtained using function densityExpressions.
nCParOBRE	OBRE robustness parameter.
dfParOBRE	A data frame containing oprimization parameters, i.e. nEta, the precision between two parameters optimization, nMaxIterLoopWc and nMaxIterLoopA, the number of iterations in the optimization proceture, nRelTol and nAbsTol, the relative and absolute tolerances.
nTheta1Init	First parameter for the beginning of the computation.
nTheta2Init	Second parameter for the beginning of the computation.
eDensityFun	The density of a two parameters distribution. To be inserted if in strDistribution the "custom" option is chosen. This should be an expression object, the two parameters should be called "nTheta1" and "nTheta2", the data "nvData" and its formula should be derivable

Value

A list with the vector containing the final parameters, the exit OBRE message, the values of vector a and matrix A, the OBRE tuning parameter c, the initial values of the parameters (if unspecified by the user, the values of MLE are reported), the vector of data, the density expression.

References

Bellio, R. (2007). Algorithms for bounded-influence estimation. Comput. Stat. Data Anal. 51, 2531-2541.

Hampel F (1968). Contributions to the theory of robust estimation. University of California.

Hampel, F., Ronchetti, E., Rousseeuw, P. & Stahel, W. (1985). Robust Statistics. The approach based on influence function. John Wiley and Sons Ltd., Chichester, UK.

Victoria-Feser, M.P. & Ronchetti, E. (1994). Robust methods for personal-income distribution models. Canadian Journal of Statistics 22, 247-258.

Examples

```
# Using the densityExpressions function for initialize the distribution
distrForOBRE <- densityExpressions(strDistribution = "normal")
simData = c(rnorm(1000, 12, 2),200,150)
try({estOBRE <- OBRE(nvData = simData, strDistribution = distrForOBRE, nCParOBRE = 3)
# Launching the generation of the density expression directly from OBRE
simData = c(rnorm(1000, 12, 2),200,150)
estOBRE <- OBRE(nvData = simData, strDistribution = "normal", nCParOBRE = 3)
# Using the "custom" option and using the normal distribution
simData = c(rnorm(1000, 12, 2),200,150)
estOBRE <- OBRE(nvData = simData, strDistribution = "custom", nCParOBRE = 3,
eDensityFun = expression((exp( -((nvData - nTheta1)^2) / (2 * nTheta2^2)) /
(sqrt(2 * pi) * nTheta2))))})
```

`OBRECheckTolParameters` *Check if OBRE matrix A and vector a are final.*

Description

The function compute the relative distance from the past to the current iteration of matrix A, with respect to the relative tolerance if at the current iteration matrix A is not null. Otherwise the absolute error is checked. Then the vector a is checked in the same way.

Usage

```
OBRECheckTolParameters(matANew, matAOld, nvANew, nvAOld, nRelTol, nAbsTol)
```

Arguments

matANew	Matrix A at the current iteration.
matAOld	Matrix A at the past iteration.
nvANew	Vector a at the current iteration.
nvAOld	Vector a at the past iteration.
nRelTol	Relative tolerance.
nAbsTol	Absolute tolerance.

Value

A flag indicating if condition on matrix A and vector a are both satisfied.

OBRECovarianceMatrix *Function that computes the OBRE covariance matrix.*

Description

The function computes matrices M (Jacobian) and Q (Variability) and uses them to evaluate the covariance matrix V.

Usage

```
OBRECovarianceMatrix(lOBRE)
```

Arguments

lOBRE	List of all the variables resulting from the OBRE computation.

Value

A list containing Jacobian of the estimate function, variability and asymptotic covariance matrices, as well as the relative efficiency with respect to Maximum Likelihood Estimator

References

Hampel, F., Ronchetti, E., Rousseeuw, P. & Stahel, W. (1985). Robust Statistics. The approach based on influence function. John Wiley and Sons Ltd., Chichester, UK.

Heritier S, Cantoni E, Copt S, Victoria-Feser M (2011). Robust Methods in Biostatistics. John Wiley and Sons Ltd., Chichester, UK.

Examples

```
try({distrForOBRE <- densityExpressions(strDistribution = "normal")
simData = c(rnorm(1000, 12, 2),200,150)
estOBRE <- OBRE(nvData = simData, strDistribution = distrForOBRE, nCParOBRE = 3)
lOBRECov = OBRECovarianceMatrix(estOBRE)})
```

OBREmatMArgumentA *Argument A for OBRE matrix M integrals.*

Description

Function computing argument A for OBRE matrix M integrals.

Usage

```
OBREmatMArgumentA(
  nvData,
  nTheta1,
  nTheta2,
  lDensityExpr,
  nCParOBRE,
  matA,
  nvA,
  nK
)
```

Arguments

nvData	The vector of data.
nTheta1	The first parameter.
nTheta2	The second parameter.
lDensityExpr	List of symbolic expressions of density, cumulative and derivatives.
nCParOBRE	OBRE c parameter.
matA	Matrix A.
nvA	Vector a.
nK	Exponent which differentiate M_1 from M_2.

OBREmatMArgumentB *Argument B for OBRE matrix M integrals.*

Description

Function computing argument B for OBRE matrix M integrals.

Usage

```
OBREmatMArgumentB(
  nvData,
  nTheta1,
  nTheta2,
  lDensityExpr,
  nCParOBRE,
  matA,
  nvA,
  nK
)
```

Arguments

nvData	The vector of data.
nTheta1	The first parameter.
nTheta2	The second parameter.
lDensityExpr	List of symbolic expressions of density, cumulative and derivatives.
nCParOBRE	OBRE c parameter.
matA	Matrix A.
nvA	Vector a.
nK	Exponent which differentiate M_1 from M_2.

OBREmatMArgumentC *Argument C for OBRE matrix M integrals.*

Description

Function computing argument C for OBRE matrix M integrals.

Usage

```
OBREmatMArgumentC(
  nvData,
  nTheta1,
  nTheta2,
  lDensityExpr,
  nCParOBRE,
  matA,
  nvA,
  nK
)
```

Arguments

nvData	The vector of data.
nTheta1	The first parameter.
nTheta2	The second parameter.
lDensityExpr	List of symbolic expressions of density, cumulative and derivatives.
nCParOBRE	OBRE c parameter.
matA	Matrix A.
nvA	Vector a.
nK	Exponent which differentiate M_1 from M_2.

OBREMatMComputation *Function computing the OBRE matrix M.*

Description

The function evaluates integrals used to compute the M_1 and M_2 OBRE matrices. Element (1,1) uses argument (A,B,F); element (1,2) uses argument (B,D,E,F); elements (2,2) uses arguments (C,D,F).

Usage

```
OBREMatMComputation(
  nvData,
  nTheta1,
  nTheta2,
  lDensityExpr,
  nCParOBRE,
  matA,
  nvA,
  nK
)
```

Arguments

nvData	The vector of data.
nTheta1	The first parameter.
nTheta2	The second parameter.
lDensityExpr	List of symbolic expressions of density, cumulative and derivatives.
nCParOBRE	OBRE c parameter.
matA	Matrix A.
nvA	Vector a.
nK	Exponent which differentiate M_1 from M_2.

Value

OBRE M matrix (M_1 if nK = 1; M_2 if nK = 2).

`OBREMatVMatMEl11` *Element [1, 1] of matrix M.*

Description

Function computing element [1, 1] of matrix M, for the computation of asymptotic covariance matrix V.

Usage

```
OBREMatVMatMEl11(nvData, nTheta1, nTheta2, lDensityExpr, nCParOBRE, matA, nvA)
```

Arguments

`nvData`	The vector of data.
`nTheta1`	The first parameter.
`nTheta2`	The second parameter.
`lDensityExpr`	List of symbolic expressions of density, cumulative and derivatives.
`nCParOBRE`	OBRE c parameter.
`matA`	Matrix A.
`nvA`	Vector a.

`OBREMatVMatMEl12` *Element [1, 2] of matrix M.*

Description

Function computing element [1, 2] of matrix M, for the computation of asymptotic covariance matrix V.

Usage

```
OBREMatVMatMEl12(nvData, nTheta1, nTheta2, lDensityExpr, nCParOBRE, matA, nvA)
```

Arguments

`nvData`	The vector of data.
`nTheta1`	The first parameter.
`nTheta2`	The second parameter.
`lDensityExpr`	List of symbolic expressions of density, cumulative and derivatives.
`nCParOBRE`	OBRE c parameter.
`matA`	Matrix A.
`nvA`	Vector a.

OBREMatVMatMEl21 *Element [2, 1] of matrix M.*

Description

Function computing element [2, 1] of matrix M, for the computation of asymptotic covariance matrix V.

Usage

```
OBREMatVMatMEl21(nvData, nTheta1, nTheta2, lDensityExpr, nCParOBRE, matA, nvA)
```

Arguments

nvData	The vector of data.
nTheta1	The first parameter.
nTheta2	The second parameter.
lDensityExpr	List of symbolic expressions of density, cumulative and derivatives.
nCParOBRE	OBRE c parameter.
matA	Matrix A.
nvA	Vector a.

OBREMatVMatMEl22 *Element [2, 2] of matrix M.*

Description

Function computing element [2, 2] of matrix M, for the computation of asymptotic covariance matrix V.

Usage

```
OBREMatVMatMEl22(nvData, nTheta1, nTheta2, lDensityExpr, nCParOBRE, matA, nvA)
```

Arguments

nvData	The vector of data.
nTheta1	The first parameter.
nTheta2	The second parameter.
lDensityExpr	List of symbolic expressions of density, cumulative and derivatives.
nCParOBRE	OBRE c parameter.
matA	Matrix A.
nvA	Vector a.

OBREMatVMatQEl11 *Element [1, 1] of matrix Q.*

Description

Function computing argument element [1, 1] of matrix Q of asymptotic covariance matrix V.

Usage

```
OBREMatVMatQEl11(nvData, nTheta1, nTheta2, lDensityExpr, nCParOBRE, matA, nvA)
```

Arguments

nvData	The vector of data.
nTheta1	The first parameter.
nTheta2	The second parameter.
lDensityExpr	List of symbolic expressions of density, cumulative and derivatives.
nCParOBRE	OBRE c parameter.
matA	Matrix A.
nvA	Vector a.

OBREMatVMatQEl12 *Element [1, 2] of matrix Q.*

Description

Function computing argument element [1, 2] of matrix Q of asymptotic covariance matrix V.

Usage

```
OBREMatVMatQEl12(nvData, nTheta1, nTheta2, lDensityExpr, nCParOBRE, matA, nvA)
```

Arguments

nvData	The vector of data.
nTheta1	The first parameter.
nTheta2	The second parameter.
lDensityExpr	List of symbolic expressions of density, cumulative and derivatives.
nCParOBRE	OBRE c parameter.
matA	Matrix A.
nvA	Vector a.

OBREMatVMatQEl22 *Element [2, 2] of matrix Q.*

Description

Function computing argument element [2, 2] of matrix Q of asymptotic covariance matrix V.

Usage

```
OBREMatVMatQEl22(nvData, nTheta1, nTheta2, lDensityExpr, nCParOBRE, matA, nvA)
```

Arguments

nvData	The vector of data.
nTheta1	The first parameter.
nTheta2	The second parameter.
lDensityExpr	List of symbolic expressions of density, cumulative and derivatives.
nCParOBRE	OBRE c parameter.
matA	Matrix A.
nvA	Vector a.

OBREnvAComputation *OBRE vector a.*

Description

The function evaluates integrals used to compute the components of OBRE a vector.

Usage

```
OBREnvAComputation(
  nvData,
  nTheta1,
  nTheta2,
  lDensityExpr,
  nCParOBRE,
  matA,
  nvA
)
```

Arguments

nvData	The vector of data.
nTheta1	The first parameter.
nTheta2	The second parameter.
lDensityExpr	The list of symbolic expressions of density, cumulative and derivatives.
nCParOBRE	OBRE c parameter.
matA	OBRE matrix A.
nvA	OBRE vector a.

Value

The OBRE a vector.

OBREnvADen *Denominator for nvA*

Description

Function computing denominator for OBRE numeric vector nvA evaluation.

Usage

```
OBREnvADen(nvData, nTheta1, nTheta2, lDensityExpr, nCParOBRE, matA, nvA)
```

Arguments

nvData	The vector of data.
nTheta1	The first parameter.
nTheta2	The second parameter.
lDensityExpr	List of symbolic expressions of density, cumulative and derivatives.
nCParOBRE	OBRE c parameter.
matA	Matrix A.
nvA	Vector a.

OBREnvANum1 *First part numerator for nvA*

Description

Function computing first part numerator for OBRE numeric vector nvA evaluation.

Usage

```
OBREnvANum1(nvData, nTheta1, nTheta2, lDensityExpr, nCParOBRE, matA, nvA)
```

Arguments

nvData	The vector of data.
nTheta1	The first parameter.
nTheta2	The second parameter.
lDensityExpr	List of symbolic expressions of density, cumulative and derivatives.
nCParOBRE	OBRE c parameter.
matA	Matrix A.
nvA	Vector a.

OBREnvANum2 *Second part numerator for nvA*

Description

Function computing second part numerator for OBRE numeric vector nvA evaluation.

Usage

```
OBREnvANum2(nvData, nTheta1, nTheta2, lDensityExpr, nCParOBRE, matA, nvA)
```

Arguments

nvData	The vector of data.
nTheta1	The first parameter.
nTheta2	The second parameter.
lDensityExpr	List of symbolic expressions of density, cumulative and derivatives.
nCParOBRE	OBRE c parameter.
matA	Matrix A.
nvA	Vector a.

OBREWeightsFun *OBRE weights.*

Description

Function for computing OBRE weights. The function computes the score function for both parameters and build the score matrix. The score matrix is then modified using OBRE parameters A matrix and a vector and an euclidean norm is derived. The weights are finally found as the minimum between the normalized nCParOBRE and 1.

Usage

```
OBREWeightsFun(nvData, nTheta1, nTheta2, lDensityExpr, nCParOBRE, matA, nvA)
```

Arguments

nvData	The vector of data.
nTheta1	The first parameter.
nTheta2	The second parameter.
lDensityExpr	The list of symbolic expressions of density, cumulative and derivatives.
nCParOBRE	OBRE c parameter.
matA	OBRE matrix A.
nvA	OBRE vector a.

Value

A numeric vector containing OBRE weights.

plot.OBREresult *Function that plot an OBREresult object.*

Description

The function computes the plot of the OBRE computation

Usage

```
## S3 method for class 'OBREresult'
plot(x, ...)
```

Arguments

x	The OBREresult object (output of OBRE function) that has to be plotted.
...	Added argument for consistency with the plot generic function.

Value

A graphical representation of an OBREresult obect. The plot is composed by four plots: the value of input data in logaritmic scale, the values of score function evaluated in the input data, the OBRE weights, the values of OBRE components.

Examples

```
try({# Generates the Normal distribution input for OBRE
distrForOBRE <- densityExpressions(strDistribution = "normal")
# Generates input data
simData = c(rnorm(100, 12, 1), rnorm(10, 10, 10))
# Estimates OBREresult object
estOBRE = OBRE(nvData = simData, strDistribution = "normal", nCParOBRE = 3)
plot(estOBRE)})
```

---

scoreComponent *First component of the score function.*

Description

The function evaluates the formula used to compute the first component of the score function. The missing elements are imputed with 0.

Usage

```
scoreComponent(nvData, nTheta1, nTheta2, lDensityExpr, nParIndex)
```

Arguments

nvData	The vector of data.
nTheta1	The first parameter.
nTheta2	The second parameter.
lDensityExpr	The list of symbolic expressions of density, cumulative and derivatives.
nParIndex	Which component parameter needs to be calculated.

Value

The first component of the score function.

summary *Generic summary method*

Description

Generic summary method

Usage

```
summary(object)
```

Arguments

object	...

summary.OBREresult *Function that summarize the results contained in an OBREresult object.*

Description

The function shows the estimated parameters, the OBRE tuning parameter, the proportion of data weighted and the relative efficiency with respect to MLE of an OBREresult object.

Usage

```
## S3 method for class 'OBREresult'
summary(object)
```

Arguments

object	The OBREresult object (output of OBRE function) that has to be plotted.

Value

The summary an OBREresult obect with the estimated parameters, the OBRE tuning parameter, the proportion of data weighted and the relative efficiency with respect to MLE.

Examples

```
try({# Generates the Normal distribution input for OBRE
distrForOBRE <- densityExpressions(strDistribution = "normal")
# Generates input data
simData = c(rnorm(100, 12, 1), rnorm(10, 10, 10))
# Estimates OBREresult object
estOBRE <- OBRE(nvData = simData, strDistribution = distrForOBRE, nCParOBRE = 3)
# Summary of the results
summary(estOBRE)})
```

Index