Package 'Delta'

July 27, 2017

Title Measure of Agreement Between Two Raters

Version 0.1.1.11

Description Measure of agreement Delta was originally by Martín & Femia (2004) <DOI:10.1348/000711004849268>. Since then has been considered as agreement measure for different fields, since their behavior is usually better than the usual Kappa index by Cohen (1960) <DOI:10.1177/001316446002000104>. The main issue with Delta is that can not be computed by hand contrary to Kappa.

Depends R (>= 3.2.0)

Imports stats

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 6.0.1.9000

NeedsCompilation no

Author Antonio Rodriguez [aut, cre],
Pedro Femia [aut, cph],
Antonio Martín [aut, cph]

Maintainer Antonio Rodriguez <tonirodriguezcontesti@gmail.com>

Repository CRAN

Date/Publication 2017-07-27 20:52:04 UTC

R topics documented:

`CheckInput` *Main Check Input function*

Description

This function perform multiple tasks. First of all, check the parameters specified by the user. Also asign default values to some parameters not defined by the user. Finally it generates error messages and halt the execution in case it is needed.

Usage

```
CheckInput(datatable, fixedrows = FALSE, gstandard = "No", maxits = 1000,
  tol = 1e-12, dplaces = 4, showall = FALSE)
```

Arguments

`datatable`	Matrix. Expected to be square matrix with at least 2 rows (columns), non negative values and at least an element different of zero.
`fixedrows`	Boolean. Indicate if sample rows are fixed beforehand. Default is TRUE.
`gstandard`	Text. Indicate if there are a Gold Standard by Rows or columns. Only first letter matter without Case sensitivity. Options are: "N" for None, "R" for in Rows and "C" for in Columns. Default is "N".
`maxits`	Whole number. Indicate the maximum number of iterations of the numeric method to calculate B. Expected to be 100 <= maxits <= 5000. Default is 1000.
`tol`	Double number. Indicate the precision of the numeric method to calculate B. Expected to be 1e-6 <= tol <= 1e-15.Default is 1e-12.
`dplaces`	Whole number. Decimal placed to be shown in the result. Expected to be 1 <= dplaces <0 6. Default 4.
`showall`	Boolean. Indicate if all output should be shown. If TRUE also shown hidden results. If FALSE shown only main output. By default is FALSE.

Examples

```
CheckInput(matrix(c(1,2,3,4),2,2),fixedrows=FALSE,gstandard="No",maxits=100,tol=1e-12,dplaces=4)
```

CheckInputData *Check Input Matrix function*

Description

This function checks that matrix introduced is as expected. Should be a matrix, squared, with a dimension greater or equal to two, without negative entries and at least an entry different of 0.

Usage

```
CheckInputData(datatable)
```

Arguments

datatable	Matrix. Expected to be square matrix with at least 2 rows (columns), non negative values and at least an element different of zero.

Examples

```
CheckInputData(matrix(c(1,2,3,4),2,2))
```

CheckSampling *Check Sampling type function*

Description

This function checks that fixedrows and gstandard parameters are correct. Also return a value indicating if it is or not valid and what kind of output should be shown at the end of the execution.

Usage

```
CheckSampling(fixedrows, gstandard)
```

Arguments

fixedrows	Boolean. Indicate if sample rows are fixed beforehand. Default is TRUE.
gstandard	Text. Indicate if there are a Gold Standard by Rows or columns. Only first letter matter without Case sensitivity. Options are: "N" for None, "R" for in Rows and "C" for in Columns. Default is "N".

Examples

```
CheckSampling(TRUE,"rows")
CheckSampling(TRUE,"Columns")
```

Delta *Delta coefficient function*

Description

This function provides an analysis of the matrix provided, returning all all the parameters estimations and SE calculations that have sense with the fixedrows and gstandard provided.

Usage

```
Delta(datatable, fixedrows = FALSE, gstandard = "No", maxits = 1000,
  tol = 1e-12, dplaces = 4, showall = FALSE)

## S3 method for class 'Delta'
print(x, ...)

## S3 method for class 'Delta'
summary(object, ...)
```

Arguments

datatable	Matrix. Expected to be square matrix with at least 2 rows (columns), non negative values and at least an element different of zero.
fixedrows	Boolean. Indicate if sample rows are fixed beforehand. Default is TRUE.
gstandard	Text. Indicate if there are a Gold Standard by Rows or columns. Only first letter matter without Case sensitivity. Options are: "N" for None, "R" for in Rows and "C" for in Columns. Default is "N".
maxits	Whole number. Indicate the maximum number of iterations of the numeric method to calculate B. Expected to be 100 <= maxits <= 5000. Default is 1000.
tol	Double number. Indicate the precision of the numeric method to calculate B. Expected to be 1e-6 <= tol <= 1e-15.Default is 1e-12.
dplaces	Whole number. Decimal placed to be shown in the result. Expected to be 1 <= dplaces <0 6. Default 4.
showall	Boolean. Indicate if all output should be shown. If TRUE also shown hidden results. If FALSE shown only main output. By default is FALSE.
x	List produced by Delta
...	Other print options
object	List produced by Delta

Details

This function study the matrix provided by the user. This function modify the matrix deleting missing rows and columns and if it is needed for the estimation, adding 0.5 to each cell.

Also calculate Cohen's Kappa coefficient and the goodness of fit for the Delta model.

Value

NULL

NULL

Examples

```
Delta(matrix(c(1,2,3,4),2,2))
Delta(matrix(c(65,5,10,20),2,2),fixedrows=TRUE,gstandard="Row")
```

GetAsinDeltaParams *Calculate Asintotic Delta related parameters function*

Description

This function perform all needed calculations to get all Delta related parameters, for a 2x2 matrix. All calculations are asintotics.

Usage

```
GetAsinDeltaParams(mx, fixedrows = TRUE)

## S3 method for class 'GetAsinDeltaParams'
print(x, ...)
```

Arguments

mx	Matrix. Agreement contingency table to perform calculations
fixedrows	Boolean. Indicate if sample rows are fixed beforehand.
x	List produced by GetAsinDeltaParams
...	Other print options

Value

NULL

Examples

```
GetAsinDeltaParams(matrix(c(60,10,10,20),2,2),TRUE)
```

GetB *Calculate B function*

Description

This function solve numericaly the non lineal inequation of the Delta system. Also return the s(i) values of the equation.

Usage

```
GetB(mx, tol = 1e-12, maxits = 1000)
```

Arguments

mx	Matrix. Modified matrix to have a solution Usually GetMx$M2.
tol	Double number. Indicate the precision of the numeric method to calculate B. Expected to be 1e-6 <= tol <= 1e-15.Default is 1e-12.
maxits	Whole number. Indicate the maximum number of iterations of the numeric method to calculate B. Expected to be 100 <= maxits <= 5000. Default is 1000.

Examples

```
GetB(mx = matrix(c(1,0,0,0,2,0,0,0,3),3,3),tol = 1e-12, maxits = 1000)
GetB(mx = matrix(c(1,2,0,3,4,0,0,0,1),3,3),tol = 1e-12, maxits = 1000)
```

GetCovariance *Calculate Covariance function*

Description

This function calculate covariance for combinations Cov(Delta,Delta), Cov(Delta,Pi) and Cov(Pi,Pi).

Usage

```
GetCovariance(mx, Delta, Pi, B)
```

Arguments

mx	Matrix. Modified matrix to have a solution Usually GetMx$M2.
Delta	Vector. Each element indicate the probability of recognize an element i.
Pi	Vector. Each element indicate the probability of classify at random an element in category i.
B	Double. Numerical solution to the equation given by the model.

Examples

```
GetCovariance(mx = matrix(c(1.5,0.5,0.5,0.5,2.5,0.5,0.5,0.5,3.5),3,3),
 Delta = c(0.4,0.5714286,0.666667),
             Pi = c(0.3333,0.333333,0.33333),B = 4.5)
GetCovariance(mx = matrix(c(60,0,3,2,50,1,3,2,79),3,3),
Delta = c( 0.8945724, 0.9522836, 0.8962094),
             Pi = c( 0.2703707, 0.1939561, 0.5356732), B = 17.94867)
```

GetDeltaParams *Calculate Delta related parameters function*

Description

This function perform all needed calculations to get all Delta related parameters. For do the exact calculations some variables previously calculated are needed.

Usage

```
GetDeltaParams(mx, Delta, Pi, k)
```

Arguments

mx	Matrix. Agreement contingency table to perform calculations
Delta	Vector. Each element indicate the probability of recognize an element i.
Pi	Vector. Each element indicate the probability of classify at random an element in category i.
k	Integer. Dimension of the problem.

Examples

```
GetDeltaParams(mx = matrix(c(60,0,3,2,50,1,3,2,79),3,3),
Delta = c( 0.8945724, 0.9522836, 0.8962094),
             Pi = c( 0.2703707, 0.1939561, 0.5356732), k = 3)
```

GetDeltaParamsVar *Calculate Delta related parameters variance function*

Description

This function perform all needed calculations to get all Delta related parameters variance. For do the exact calculations some variables previously calculated are needed.

Usage

```
GetDeltaParamsVar(mx, fixedrows = FALSE, Delta, Pi, k, Cov, E)
```

Arguments

mx	Matrix. Agreement contingency table to perform calculations
fixedrows	Boolean. Indicate if sample rows are fixed beforehand.
Delta	Vector. Each element indicate the probability of recognize an element i.
Pi	Vector. Each element indicate the probability of classify at random an element in category i.
k	Integer. Dimension of the problem.
Cov	Matrix. Covariance matrix of Delta.
E	Double. Value calculated for Cov matrix derivation.

Examples

```
GetDeltaParamsVar(mx = matrix(c(60,0,3,2,50,1,3,2,79),3,3),
fixedrows = FALSE,Delta = c( 0.8945724, 0.9522836, 0.8962094),
     Pi = c( 0.2703707, 0.1939561, 0.5356732), k = 3,
     Cov = matrix(c(0.002736490,  0.000004188, -0.001074704,
                0.000004188, 0.001141059, -0.000181746,
               -0.001074704, -0.000181746,  0.004912131),3,3),
E = c(0.03159824,  0.01304313, -0.88650011))
```

GetDeltaPi *Calculate Delta and Pi parameters function*

Description

This function provide an estimation of Pi and Delta for each category. To do so, it is needed to solve the non-linear equation of B, given by the function GetB.

Usage

```
GetDeltaPi(mx, tp, tol = 1e-12, maxits = 1000)
```

Arguments

mx	Matrix. Modified matrix to have a solution Usually GetMx$M2.
tp	String. Type of problem.
tol	Double number. Indicate the precision of the numeric method to calculate B. Expected to be 1e-6 <= tol <= 1e-15.Default is 1e-12.
maxits	Whole number. Indicate the maximum number of iterations of the numeric method to calculate B. Expected to be 100 <= maxits <= 5000. Default is 1000.

Details

In some type of problems, such as "2.1" and "3.1", problem has a unique solution for Delta and infinite for Pi. For this reason we calculate the solution for Delta and get Pi as NULL.

Examples

```
GetDeltaPi(mx = matrix(c(1,0,0,0,2,0,0,0,3),3,3), tp = "3.1", tol = 1e-12, maxits = 1000)
GetDeltaPi(mx = matrix(c(1.5,2.5,0.5,3.5,4.5,0.5,0.5,0.5,1.5),3,3), tp = "3.2",
           tol = 1e-12, maxits = 1000)
```

`GetGoodness` *Calculate Goodness of fit function*

Description

This function provide an Chi-square test for the given matrix, Delta and Pi provided.

Usage

```
GetGoodness(mx, Pi, Delta)

## S3 method for class 'GetGoodness'
print(x, ...)
```

Arguments

`mx`	Matrix. Modified matrix to have a solution Usually GetMx$M2.
`Pi`	Vector. Each element indicate the probability of classify at random an element in category i.
`Delta`	Vector. Each element indicate the probability of recognize an element i.
`x`	List produced by GetGoodness
`...`	Other print options

Value

`NULL`

Examples

```
GetGoodness(mx = matrix(c(1,0,0,0,2,0,0,0,3),3,3), Delta = c(1,1,1), Pi = NULL)
GetGoodness(mx = matrix(c(1.5,2.5,0.5,3.5,4.5,0.5,0.5,0.5,1.5),3,3),
            Delta = c(-0.2662395,  0.2047577,  0.5664672),
            Pi = c(0.42564365, 0.49700867, 0.07734769))
GetGoodness(mx = matrix(c(60,0,3,2,50,1,3,2,79),3,3),
            Delta = c( 0.8945724, 0.9522836, 0.8962094),
            Pi = c( 0.2703707, 0.1939561, 0.5356732))
```

GetKappa *Calculate Cohen's Kappa coefficient function*

Description

This function perform Cohen's Kappa coefficient calculations. The function provide the Kappa coefficient and SE.

Usage

```
GetKappa(mx)

## S3 method for class 'GetKappa'
print(x, ...)
```

Arguments

mx	Matrix. Agreement contingency table to perform calculations
x	List produced by GetKappa
...	Other print options

Value

NULL

Examples

```
GetKappa(matrix(c(50,10,10,20),2,2))
```

GetM1 *Get reduced matrix (M1) function*

Description

This function reduce matrix provided by the user deleting missing categories, those j where sum(datatable[j,]) = sum(datatable[,j]) = 0. Also provide a list of the categories deleted and provides the new size of the problem

Usage

```
GetM1(datatable)

## S3 method for class 'GetM1'
print(x, ...)
```

Arguments

datatable	Matrix. Expected to be square matrix with at least 2 rows (columns), non negative values and at least an element different of zero.
x	List produced by GetM1
...	Other print options

Value

NULL

Examples

```
GetM1(matrix(c(1,2,0,3,4,0,0,0,0),3,3))
```

GetMx *Get matrix of the problem (Mx) function*

Description

This function produce 4 new auxiliar matrix. The definition of the matrix depends on the type of problem (tp). For example, if tp = 2.X, the program will create M2 with dimension 3x3 instead 2x2, and will add to each cell, this way we can use this auxiliars macros to solve the problem and avoid issues with solutions in the boundary or not completly defined. See complete list in Detail section.

Usage

```
GetMx(tp, M1)
```

Arguments

tp	String. Type of problem. Generated by GetProblemType function.
M1	Matrix. Initial matrix without missing categories.

Details

———-+———+————————+————————+————————+————————+ | Case | M2 | M3 | M4 | M5 | ———-+———+————————+————————+————————+————————+ Purpose| | Delta + SE | Delta + SE | Delta asintotic | Delta asintotic | | | | M1 diagonal or| Kappa + SE | extra | | | | frontier sol. | | | ———-+———+————————+————————+————————+————————+ | 2.0 | M1* +0.5 | — | M1 + 0.5 | M1 + 1 | ———-+———+————————+————————+————————+————————+ | 2.1 | M1* +0.5 | — | M1 + 0.5 | M1 + 1 | ———-+———+————————+————————+————————+————————+ |

2.2 | M1* +0.5 | — | M1 | M1 + 1 | ——-+——+————+————+————+———
———+ | 3.0 | M1 +0.5 | — | M1 + 0.5 | — | ——-+——+————+————+———————
+————+ | 3.1 | M1 | M1 + 0.5 | M1 + 0.5 | — | ——-+——+————+————+——
————+————+ | 3.2 | M1 +0.5 | — | M1 | — | ——-+——+————+————+—
————+————+ | 3.3 | M1 +0.5 | — | M1 | — | ——-+——+————+————————
+————+————+ | 3.4 | M1 | M1 + 0.5 | M1 | — | ——-+——+————+———
——+————+————+ | 3.5 | M1 | — | M1 | — | ——-+——+————+————
—+————+————+

Examples

```
GetMx("3.2",matrix(c(1,2,0,3,4,0,0,0,1),3,3))
```

`GetProblemType` *Get problem type function*

Description

This function apply TestType function multiple times, to be able of determine the type of problems between following 1.0 = Matrix size 0 1.1 = Matrix size 1 2.0 = Matrix size 2 where a marginal row or column equal 0 2.1 = Matrix size 2 and diagonal 2.2 = Matrix size 2 not diagonal 3.0 = Matrix size greater than 2 with a marginal row or column equal 0 3.1 = Matrix size greater than 2 and diagonal 3.2 = Matrix size greater than 2 with k-1 row or columns with marginal equal to a_ii or k-2 where marginals of columns and rows equals to a_ii 3.3 = Matrix size greater than 2 with h where Xt - sum(diag(M1)) = Xr[h] + Xc[h] - 2 * M1[h,h] 3.4 = Matrix size greater than 2 with i where a_ii = 0 or equl to marginals by columns or row 3.5 = Matrix size greater than 2 without a_ii = 0 or without marginals by columns or row equals to zero

Usage

```
GetProblemType(M1, k)
```

Arguments

`M1`	Matrix. Matrix reduced.
`k`	Integer. Size of the problem.

Examples

```
GetProblemType(matrix(c(1,2,0,3,4,0,0,0,1),3,3), 3)
GetProblemType(matrix(c(1,0,0,0,2,0,0,0,3),3,3), 3)
```

`TestType` *Test problem type function*

Description

This function calculate for the parameters given, the type of problem. Problems could be 1.0 = Matrix size 0 1.1 = Matrix size 1 2.0 = Matrix size 2 where a marginal row or column equal 0 2.1 = Matrix size 2 and diagonal 2.2 = Matrix size 2 not diagonal 3.0 = Matrix size greater than 2 with a marginal row or column equal 0 3.1 = Matrix size greater than 2 and diagonal 3.2 = Matrix size greater than 2 with k-1 row or columns with marginal equal to a_ii or k-2 where marginals of columns and rows equals to a_ii 3.3 = Matrix size greater than 2 with h where Xt - sum(diag(M1)) = Xr[h] + Xc[h] - 2 * M1[h,h] 3.4 = Matrix size greater than 2 with i where a_ii = 0 or equal to marginals by columns or row 3.5 = Matrix size greater than 2 without a_ii = 0 or without marginals by columns or row equals to zero

Usage

```
TestType(tp, M1, k, Xr, Xc, Xt)
```

Arguments

`tp`	String. Type of problem to check of the previous list.
`M1`	Matrix. Matrix reduced.
`k`	Integer. Size of the problem.
`Xr`	Vector. Marginals by row.
`Xc`	Vector. Marginals by column.
`Xt`	Double. Sum of all elements in the matrix.

Examples

```
TestType("1.0", matrix(c(1,2,0,3,4,0,0,0,1),3,3), 3, c(4,5,1), c(3,7,1), 11)
```

Index