Package 'PaolaR6Nuevo'

November 2, 2023

Type Package

Title Confusion Matrix

Version 0.1.0

Author Paola

Maintainer Paola Barba Ceballos <pbarba@ujaen.es>

Description This package provides useful functions for the analysis of confusion matrices in classification problems. Includes methods to calculate overall accuracy, user accuracy, and map creator accuracy.

Imports R6

License GPL

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

R topics documented:

`MatCon` *Confusion matrix*

Description

Using the confusion matrix, various indices are calculated.

Value

Object of class MatCon #y mas

Methods

Public methods:

- MatCon$new()
- MatCon$oa()
- MatCon$dec()
- MatCon$ua()
- MatCon$uai()
- MatCon$pai()
- MatCon$pa()
- MatCon$aup()
- MatCon$ICSI()
- MatCon$mah()
- MatCon$mas()
- MatCon$cku()
- MatCon$ckp()
- MatCon$mcku()
- MatCon$mckp()
- MatCon$ecnu()
- MatCon$ecnp()
- MatCon$aau()
- MatCon$Normalize()
- MatCon$aap()
- MatCon$daup()
- MatCon$CSI()
- MatCon$amah()
- MatCon$amas()
- MatCon$cau()
- MatCon$cap()
- MatCon$caup()
- MatCon$KappaValue()
- MatCon$mkp()
- MatCon$ami()
- MatCon$nmiu()
- MatCon$nmip()
- MatCon$nmiam()
- MatCon$nmigm()
- MatCon$nmimx()
- MatCon$MPseudozeroes()
- MatCon$MTypify()
- MatCon$MAllParameters()
- MatCon$CAccuracies()
- MatCon$CAccuraciesW()
- MatCon$DetailedKappa()
- MatCon$DetailedCKappa()
- MatCon$DetailedWKappa()

- MatCon$DetailedTau()
- MatCon$QES()
- MatCon$clone()

Method `new()`**:**

Usage:

```
MatCon$new(values, ID = NULL, Date = NULL)
```

Arguments:

`values` Confusion matrix

`ID` Identifier (optional)

`Date` System or user-provided date (optional)

Method `oa()`**:** Overall accuracy for a particular classified image/map is then calculated by dividing the sum of the entries that form the major diagonal (i.e., the number of correct classifications) by the total number of samples taken.

The mathematical expression is:

$$oa = \frac{\sum_{i=1}^{n} x_{ii}}{\sum_{i,j=1}^{n} x_{ij}}$$

Where:

1. 'oa': overall accuracy.
2. 'x_ii': diagonal element of the matrix.
3. 'x_ij': element of the matrix.

This represents a mathematical expression with a fraction.

Usage:

```
MatCon$oa(...)
```

Arguments:

`...` (ignored).

Returns: Overall accuracy and variance.

Examples:

```
A<-matrix(c(36,1,0,0,2,0,0,1,20),nrow=3,ncol=3)
p<-MatCon$new(A)
p$oa()
```

Method `dec()`**:** Determines whether a value is decimal or not.

Usage:

```
MatCon$dec(...)
```

Arguments:

`...` (ignored).

Returns: TRUE or FALSE

Examples:

```
A<-matrix(c(36,1,0,0,2,0,0,1,20),nrow=3,ncol=3)
p<-MatCon$new(A)
p$dec()
```

Method `ua()`**:** The accuracy from the point of view of a map user, not the map maker.
The mathematical expression is:

$$ua = \frac{x_{ii}}{\sum_{j=1}^{n} x_{ij}}$$

where:

1. 'ua': user accuracy.
2. 'x_ii': diagonal element of the matrix.
3. 'x_ij': element of the matrix.

Usage:

```
MatCon$ua(...)
```

Arguments:

`...` (ignored).

Returns: vector of values with the user's accuracy indexes of all classes and their variances.

Examples:

```
A<-matrix(c(36,1,0,0,2,0,0,1,20),nrow=3,ncol=3)
p<-MatCon$new(A,ID=1,Date="30/10/2023")
p$ua()
```

Method `uai()`**:** The accuracy from the point of view of a map user, not the map maker.

$$ua_i = \frac{x_{ii}}{\sum_{j=1}^{n} x_{ij}}$$

where:

1. 'ua_i': user accuracy.
2. 'x_ii': diagonal element of the matrix.
3. 'x_ij': element of the matrix.

Usage:

```
MatCon$uai(i)
```

Arguments:

`i` User class to evaluate

Returns: Class i user accuracy index and their variance.

Examples:

```
A<-matrix(c(36,1,0,0,2,0,0,1,20),nrow=3,ncol=3)
p<-MatCon$new(A)
p$uai(2)
```

Method `pai()`**:** The map accuracy from the point of view of the map maker (the producer).

$$pa_i = \frac{x_{jj}}{\sum_{j=1}^{n} x_{ij}}$$

where:

1. 'pa_i': producer accuracy.
2. 'x_jj': diagonal element of the matrix.
3. 'x_ij': element of the matrix.

Usage:

```
MatCon$pai(i)
```

Arguments:

`i` Producer class to evaluate

Returns: Class i producer accuracy index and their variance.

Examples:

```
A<-matrix(c(36,1,0,0,2,0,0,1,20),nrow=3,ncol=3)
p<-MatCon$new(A)
p$pai(1)
```

Method `pa()`**:** The map accuracy from the point of view of the map maker (the producer).

$$pa = \frac{x_{jj}}{\sum_{j=1}^{n} x_{ij}}$$

where:

1. 'pa': producer accuracy.
2. 'x_jj': diagonal element of the matrix.
3. 'x_ij': element of the matrix.

Usage:

```
MatCon$pa(...)
```

Arguments:

`...` (ignored).

Returns: Vector of values with the producer's accuracy indexes of all classes

Examples:

```
A<-matrix(c(36,1,0,0,2,0,0,1,20),nrow=3,ncol=3)
p<-MatCon$new(A)
p$pa()
```

Method `aup()`**:** Average of the accuracy from the point of view of a map user, not the map maker and the map accuracy from the point of view of the map maker (the producer).

$$aup = \frac{ua_i + pa_i}{2}$$

where:

1. 'aup': Average of user's and producer's accuracy.
2. 'ua_i': user accuracy
3. 'pa_i': producer accuracy.

Usage:

```
MatCon$aup(i)
```

Arguments:

`i` Class to evaluate.

Returns: Average of user's and producer's accuracy and its variance.

Examples:

```
A<-matrix(c(36,1,0,0,2,0,0,1,20),nrow=3,ncol=3)
p<-MatCon$new(A)
p$aup(2)
```

Method `ICSI()`**:** The Individual Classification Success Index (ICSI) applies to the classification effectiveness for one particular class of interest.

$$ICSI = ua_i + pa_i - 1$$

where:

1. 'ICSI': Individual classification success index.
2. 'ua_i': user accuracy.
3. 'pa_i': producer accuracy.

Usage:

```
MatCon$ICSI(i)
```

Arguments:

`i` Class to evaluate.

Returns: Individual Classification Success Index and its variance.

Examples:

```
A<-matrix(c(36,1,0,0,2,0,0,1,20),nrow=3,ncol=3)
p<-MatCon$new(A)
p$ICSI(2)
```

Method `mah()`**:** The probability that a randomly chosen point of a specific class on the map has a correspondence of the same class in the same position in the field and that a randomly chosen point in the field of the same class has a correspondence of the same class in the same position on the map.

$$mah = \frac{2}{\frac{1}{ua_i} + \frac{1}{pa_i}}$$

where:

1. 'mah': Hellden's mean accuracy.
2. 'ua_i': user accuracy.
3. 'pa_i': producer accuracy.

Usage:

```
MatCon$mah(i)
```

Arguments:

`i` Class to evaluate.

Examples:

```
A<-matrix(c(36,1,0,0,2,0,0,1,20),nrow=3,ncol=3)
p<-MatCon$new(A)
p$mah(2)
```

Method `mas()`**:** Mapping accuracy for each class is stated as the number of correctly classified pixels (equal to the total in the correctly classified area) in terms of all pixels affected by its classification (equal to this total in the displayed area as well as the pixels involved in errors of commission and omission).

$$mas = \frac{x_{ii}}{\sum_{j=1}^{n} x_{.j} + \sum_{i=1}^{n} x_{i.} - x_{ii}}$$

where:

1. 'mas': Short's mapping accuracy
2. 'x_ii': diagonal element of the matrix.
3. 'x_.j': sum with respect to j (rows).
4. 'x_i.': sum with respect to i (columns).

Usage:

```
MatCon$mas(i)
```

Arguments:

`i` Class to evaluate.

Returns: Short's mapping accuracy and its variance.

Examples:

```
A<-matrix(c(36,1,0,0,2,0,0,1,20),nrow=3,ncol=3)
p<-MatCon$new(A)
p$mas(2)
```

Method `cku()`**:** Conditional Kappa will identify the degree of agreement between the two raters for each possible category.

$$cku = \frac{ua_i - \frac{\sum_{i=1}^{n} x_{i\cdot}}{\sum_{i=1}^{n}\sum_{j=1}^{n} x_{ij}}}{1 - \frac{\sum_{i=1}^{n} x_{i\cdot}}{\sum_{i=1}^{n}\sum_{j=1}^{n} x_{ij}}}$$

where:

1. 'cku': Conditional kappa (user's).
2. 'ua_i': user accuracy.
3. 'x_ii': diagonal element of the matrix.
4. 'x_.j': sum with respect to j (rows).
5. 'x_i.': sum with respect to i (columns).

Usage:

```
MatCon$cku(i)
```

Arguments:

`i` Class to evaluate.

Returns: Conditional kappa (user's) and its variance.

Examples:

```
A<-matrix(c(36,1,0,0,2,0,0,1,20),nrow=3,ncol=3)
p<-MatCon$new(A)
p$cku(2)
```

Method `ckp()`**:** Conditional Kappa will identify the degree of agreement between the two raters for each possible category.

$$ckp = \frac{pa_i - \frac{\sum_{j=1}^{n} x_{\cdot j}}{\sum_{i=1}^{n}\sum_{j=1}^{n} x_{ij}}}{1 - \frac{\sum_{j=1}^{n} x_{\cdot j}}{\sum_{i=1}^{n}\sum_{j=1}^{n} x_{ij}}}$$

where:

1. 'ckp': Conditional kappa (producer's).
2. 'pa_i': producer accuracy.

3. 'x_ii': diagonal element of the matrix.
4. 'x_.j': sum with respect to j (rows).
5. 'x_i.': sum with respect to i (columns).

Usage:

```
MatCon$ckp(i)
```

Arguments:

`i` Class to evaluate.

Returns: Conditional kappa (producer's) and its variance.

Examples:

```
A<-matrix(c(36,1,0,0,2,0,0,1,20),nrow=3,ncol=3)
p<-MatCon$new(A)
p$ckp(2)
```

Method `mcku()`**:** Modified kappa index for the user

$$mcku = \frac{ua_i - \frac{1}{\sqrt{card(p)}}}{1 - \frac{1}{\sqrt{card(p)}}}$$

where:

1. 'mcku': Modified conditional kappa (user's).
2. 'ua_i': user accuracy.
3. 'card(p)': number of elements of the matrix, cardinal of the matrix.

Usage:

```
MatCon$mcku(i)
```

Arguments:

`i` Class to evaluate.

Returns: Modified conditional kappa (user's) and its variance.

Examples:

```
A<-matrix(c(36,1,0,0,2,0,0,1,20),nrow=3,ncol=3)
p<-MatCon$new(A)
p$mcku(2)
```

Method `mckp()`**:** Modified kappa index for the producer

$$mckp = \frac{pa_i - \frac{1}{\sqrt{card(p)}}}{1 - \frac{1}{\sqrt{card(p)}}}$$

where:

1. 'mckp': Modified conditional kappa (producer's).
2. 'pa_i': producer accuracy.
3. 'card(p)': number of elements of the matrix, cardinal of the matrix.

Usage:

```
MatCon$mckp(i)
```

Arguments:

`i` Class to evaluate.

Returns: Modified conditional kappa (producer's) and variance.

Examples:

```
A<-matrix(c(36,1,0,0,2,0,0,1,20),nrow=3,ncol=3)
p<-MatCon$new(A)
p$mckp(2)
```

Method `ecnu()`**:** Relative entropy is a quantity that measures the difference between two maps.

$$H(A) = -\sum_{j=1}^{n}((\frac{\sum_{i=1}^{n} x_{i\cdot}}{\sum_{i,j=1}^{n} x_{ij}}) \cdot \log(\frac{\sum_{i=1}^{n} x_{i\cdot}}{\sum_{i,j=1}^{n} x_{ij}}))$$

$$H(A|b_i) = -\sum_{j=1}^{n}((\frac{x_{ij}}{\sum_{j=1}^{n} x_{\cdot j}}) \cdot \log(\frac{x_{ij}}{\sum_{j=1}^{n} x_{\cdot j}}))$$

$$ecnu = \frac{H(A) - H(A|b_i)}{H(A)}$$

where:

1. 'ecnu': Relative change of entropy given a category on map.
2. 'H(A)': the entropy of the map.
3. 'x_.j': sum with respect to j (rows).
4. 'x_i.': sum with respect to i (columns).
5. 'H(A|b_i)': Entropy of map A knowing that the location corresponding to map B is in class b_i.

Usage:

```
MatCon$ecnu(i)
```

Arguments:

`i` Class to evaluate.

Returns: Relative change of entropy given a category on map and its variance.

Examples:

```
A<-matrix(c(36,1,0,0,2,0,0,1,20),nrow=3,ncol=3)
p<-MatCon$new(A)
p$ecnu(2)
```

Method `ecnp()`**:** Relative entropy is a quantity that measures the difference between two ground truthing.

$$H(B) = -\sum_{i=1}^{n}((\frac{\sum_{j=1}^{n} x_{\cdot j}}{\sum_{i,j=1}^{n} x_{ij}}) \cdot \log(\frac{\sum_{j=1}^{n} x_{\cdot j}}{\sum_{i,j=1}^{n} x_{ij}}))$$

$$H(B|a_j) = -\sum_{j=1}^{n}((\frac{x_{ij}}{\sum_{i=1}^{n} x_{i\cdot}}) \cdot \log(\frac{x_{ij}}{\sum_{i=1}^{n} x_{i\cdot}}))$$

$$ecnp = \frac{H(B) - H(B|a_j)}{H(B)}$$

where:

1. 'ecnp': Relative change of entropy given a category on ground truthing.
2. 'H(B)': the entropy of the map.

3. 'x_.j': sum with respect to j (rows).
4. 'x_i.': sum with respect to i (columns).
5. 'H(B|a_j)': Entropy of map B knowing that the location corresponding to map A is in class a_j.

Usage:

```
MatCon$ecnp(i)
```

Arguments:

`i` Class to evaluate

Returns: Relative change of entropy given a category on ground truthing and its variance.

Examples:

```
A<-matrix(c(36,5,4,7,2,2,6,1,20),nrow=3,ncol=3)
p<-MatCon$new(A)
p$ecnp(2)
```

Method `aau()`**:** The average accuracy is an average of the accuracy of individual categories. Because the individual categories can be the user's or the producer's accuracy, it can be computed in both ways accordingly.

$$aau = \frac{1}{\sqrt{card(p)}} \sum_{i=1}^{n} \frac{x_{ii}}{\sum_{j=1}^{n} x_{ij}}$$

where:

1. 'aau': Average accuracy from user's perspective.
2. 'card(p)': number of elements of the matrix, cardinal of the matrix.
3. 'x_.j': sum with respect to j (rows).
4. 'x_ii': diagonal element of the matrix.

Usage:

```
MatCon$aau()
```

Returns: Average accuracy from user's perspective and its variance.

Examples:

```
A<-matrix(c(36,1,0,0,2,0,0,1,20),nrow=3,ncol=3)
p<-MatCon$new(A)
p$aau()
```

Method `Normalize()`**:** An iterative process is carried out where each element is divided by the total of the sum of its row, thus obtaining new values. In the next iteration, all the elements are added by columns and each element is divided by the total of its column and they obtain new values, and so on.

Usage:

```
MatCon$Normalize(n = NULL)
```

Arguments:

`n` Iteration

Returns: Normalized matrix (Class MatCon) and its variance.

Examples:

```
A<-matrix(c(36,1,0,0,2,0,0,1,20),nrow=3,ncol=3)
p<-MatCon$new(A)
p$Normalize()
p$Normalize(1000)
```

Method `aap()`**:** The average accuracy is an average of the accuracy of individual categories. Because the individual categories can be the user's or the producer's accuracy, it can be computed in both ways accordingly.

$$aap = \frac{1}{\sqrt{card(p)}} \sum_{i=1}^{n} \frac{x_{ii}}{\sum_{j=1}^{n} x_{ji}}$$

where:

1. 'aap': Average accuracy from producer's perspective.
2. 'card(p)': number of elements of the matrix, cardinal of the matrix.
3. 'x_.j': sum with respect to j (rows).
4. 'x_ii': diagonal element of the matrix.

Usage:

```
MatCon$aap()
```

Returns: Average accuracy from producer's perspective and its variance.

Examples:

```
A<-matrix(c(36,1,0,0,2,0,0,1,20),nrow=3,ncol=3)
p<-MatCon$new(A)
p$aap()
```

Method `daup()`**:** It is the average of the average accuracy from user's and producer's perspective.

$$daup = \frac{aau + aap}{2}$$

where:

1. 'daup': Double average of user's and producer's perspective.
2. 'aau': Average accuracy from user's perspective.
3. 'aap': Average accuracy from producer's perspective.

Usage:

```
MatCon$daup()
```

Returns: Double average of user's and producer's perspective and its variance.

Examples:

```
A<-matrix(c(36,1,0,0,2,0,0,1,20),nrow=3,ncol=3)
p<-MatCon$new(A)
p$daup()
```

Method `CSI()`**:** The Classification Success Index (CSI) applies to all classes and gives an overall estimation of classification effectiveness.

$$CSI = aau + aap - 1$$

where:

1. 'CSI': Classification succes index.
2. 'aau': Average accuracy from user's perspective.
3. 'aap': Average accuracy from producer's perspective.

Usage:

```
MatCon$CSI()
```

Returns: Classification sucess index and its variance.

Examples:

```
A<-matrix(c(36,1,0,0,2,0,0,1,20),nrow=3,ncol=3)
p<-MatCon$new(A)
p$CSI()
```

Method `amah()`**:** This function provides the average value of the Hellden mean precision index

$$amah = \frac{1}{\sqrt{card(p)}} \sum_{i=1}^{n} \frac{2}{\frac{1}{ua_i} + \frac{1}{pa_i}}$$

where:

1. 'amah': Average of Hellden's mean accuracy index.
2. 'ua_i': user accuracy.
3. 'pa_i': producer accuracy.
4. 'card(p)': number of elements of the matrix, cardinal of the matrix.

Usage:

```
MatCon$amah()
```

Returns: Average of Hellden's mean accuracy index and its variance.

Examples:

```
A<-matrix(c(36,1,0,0,2,0,0,1,20),nrow=3,ncol=3)
p<-MatCon$new(A)
p$amah()
```

Method `amas()`**:** This function provides the average of Short's mapping accuracy index.

$$amas = \frac{1}{\sqrt{card(p)}} \frac{\frac{\sum_{i=1}^{n} x_{ii}}{\sum_{i,j=1}^{n} x_{ij}}}{\sum_{j=1}^{n} x_{\cdot j} + \sum_{i=1}^{n} x_{i\cdot} - x_{ii}}$$

where:

1. 'amas': Average of Short's mapping accuracy index.
2. 'x_ii': diagonal element of the matrix.
3. 'x_.j': sum with respect to j (rows).
4. 'x_i.': sum with respect to i (columns).
5. 'card(p)': number of elements of the matrix, cardinal of the matrix.

Usage:

```
MatCon$amas()
```

Returns: Average of Short's mapping accuracy index and its variance.

Examples:

```
A<-matrix(c(36,1,0,0,2,0,0,1,20),nrow=3,ncol=3)
p<-MatCon$new(A)
p$amas()
```

Method `cau()`**:** The combined accuracy is the average of the overall accuracy and average accuracy.

$$cau = \frac{oa + aau}{2}$$

where:

1. 'cau': Combined accuracy from user's perspective.
2. 'oa': Overall accuracy.
3. 'aau': Average accuracy from user's perspective.

Usage:

```
MatCon$cau()
```

Returns: Combined accuracy from user's perspective and its variance.

Examples:

```
A<-matrix(c(36,1,0,0,2,0,0,1,20),nrow=3,ncol=3)
p<-MatCon$new(A)
p$cau()
```

Method `cap()`**:** The combined accuracy is the average of the overall accuracy and average accuracy.

$$cap = \frac{oa + aap}{2}$$

where:

1. 'cap': Combined accuracy from producer's perspective.
2. 'oa': overall accuracy.
3. 'aap': Average accuracy from producer's perspective.

Usage:

```
MatCon$cap()
```

Returns: Combined accuracy from producer's perspective and its variance.

Examples:

```
A<-matrix(c(36,1,0,0,2,0,0,1,20),nrow=3,ncol=3)
p<-MatCon$new(A)
p$cap()
```

Method `caup()`**:** The combined accuracy is the average of the overall accuracy and average accuracy.

$$caup = \frac{oa + amah}{2}$$

where:

1. 'caup': Combined accuracy from both user's and producer's perspectives.
2. 'oa': Overall accuracy.
3. 'amah': Average of Hellden's mean accuracy index.

Usage:

```
MatCon$caup()
```

Returns: Combined accuracy from both user's and producer's perspectives and its variance.

Examples:

```
A<-matrix(c(36,1,0,0,2,0,0,1,20),nrow=3,ncol=3)
p<-MatCon$new(A)
p$caup()
```

Method `KappaValue()`**:** It measures the relationship of beyond chance agreement to expected disagreement.

$$ea = \sum_{i=1}^{n}(\frac{x_{.i}}{\sum_{j=1}^{n} x_{ij}} \cdot \frac{x_{i.}}{\sum_{j=1}^{n} x_{ij}}) KappaValue = \frac{oa - ea}{1 - ea}$$

where:

1. 'KappaValue': Kappa coefficient.
2. 'oa': Overall accuracy.
3. 'ea': Expected accuracy of agreement if agreement were purely random.

Usage:

```
MatCon$KappaValue()
```

Returns: Kappa coefficient and its variance.

Examples:

```
A<-matrix(c(36,1,0,0,2,0,0,1,20),nrow=3,ncol=3)
p<-MatCon$new(A)
p$KappaValue()
```

Method `mkp()`**:** It is the proportion of agreement after chance agreement is removed from consideration.

$$mkp = \frac{oa - \frac{1}{\sqrt{card(p)}}}{1 - \frac{1}{\sqrt{card(p)}}}$$

where:

1. 'mkp': Modified kappa
2. 'oa': Overall accuracy.
3. 'card(p)': number of elements of the matrix, cardinal of the matrix.

Usage:

```
MatCon$mkp()
```

Returns: Modified kappa and its variance.

Examples:

```
A<-matrix(c(36,1,0,0,2,0,0,1,20),nrow=3,ncol=3)
p<-MatCon$new(A)
p$mkp()
```

Method `ami()`**:** Average mutual information (AMI), is applied to the comparison of thematic maps.

$$ami = \sum_{i,j=1}^{n}(\frac{x_{ij}}{\sum_{i,j=1}^{n} x_{ij}} \cdot \log_{10}(\frac{x_{ij}}{\frac{\sum_{i=1}^{n} x_{i.} \cdot \sum_{j=1}^{n} x_{.j}}{\sum_{i,j=1}^{n} x_{ij}}}))$$

where:

1. 'ami': Average mutual information.
2. 'x_.j': sum with respect to j (rows).
3. 'x_i.': sum with respect to i (columns).

Usage:

```
MatCon$ami()
```

Returns: Average mutual information and its variance.

Examples:

```
A<-matrix(c(36,5,4,7,2,2,6,1,20),nrow=3,ncol=3)
p<-MatCon$new(A)
p$ami()
```

Method `nmiu()`**:**

$$H(B) = -\sum_{i=1}^{n}((\frac{\sum_{j=1}^{n} x_{\cdot j}}{\sum_{i,j=1}^{n} x_{ij}}) \cdot \log(\frac{\sum_{j=1}^{n} x_{\cdot j}}{\sum_{i,j=1}^{n} x_{ij}}))$$

$$nmiu = \frac{ami}{H(B)}$$

where:

1. 'nmiu': Normalized mutual information using the entropy on map.
2. 'H(B)': The entropy of the map.
3. 'x_.j': sum with respect to j (rows).
4. 'x_i.': sum with respect to i (columns).
5. 'ami': Average mutual information.

Usage:

```
MatCon$nmiu()
```

Returns: Normalized mutual information using the entropy on map and its variance.

Examples:

```
A<-matrix(c(36,5,4,7,2,2,6,1,20),nrow=3,ncol=3)
p<-MatCon$new(A)
p$nmiu()
```

Method `nmip()`**:**

$$H(A) = -\sum_{j=1}^{n}((\frac{\sum_{i=1}^{n} x_{i\cdot}}{\sum_{i,j=1}^{n} x_{ij}}) \cdot \log(\frac{\sum_{i=1}^{n} x_{i\cdot}}{\sum_{i,j=1}^{n} x_{ij}}))$$

$$nmip = \frac{ami}{H(A)}$$

where:

1. 'nmip': Normalized mutual information using the entropy on ground truthing.
2. 'H(A)': the entropy of the map.
3. 'x_.j': sum with respect to j (rows).
4. 'x_i.': sum with respect to i (columns).
5. 'ami': Average mutual information.

Usage:

```
MatCon$nmip()
```

Returns: Normalized mutual information using the entropy on ground truthing and its variance.

Examples:

```
A<-matrix(c(36,5,4,7,2,2,6,1,20),nrow=3,ncol=3)
p<-MatCon$new(A)
p$nmip()
```

Method `nmiam()`**:**

$$H(A) = -\sum_{j=1}^{n}((\frac{\sum_{i=1}^{n} x_{i\cdot}}{\sum_{i,j=1}^{n} x_{ij}}) \cdot \log(\frac{\sum_{i=1}^{n} x_{i\cdot}}{\sum_{i,j=1}^{n} x_{ij}}))$$

$$H(B) = -\sum_{i=1}^{n}((\frac{\sum_{j=1}^{n} x_{\cdot j}}{\sum_{i,j=1}^{n} x_{ij}}) \cdot \log(\frac{\sum_{j=1}^{n} x_{\cdot j}}{\sum_{i,j=1}^{n} x_{ij}}))$$

$$nmiam = \frac{2ami}{HA + HB}$$

where:

1. 'nmiam': Normalized mutual information using the arithmetic mean of the entropies on map and on ground truthing.
2. 'H(A)': the entropy of the map.
3. 'H(B)': The entropy of the map.
4. 'x_.j': sum with respect to j (rows).
5. 'x_i.': sum with respect to i (columns).
6. 'ami': Average mutual information.

Usage:

```
MatCon$nmiam()
```

Returns: Normalized mutual information using the arithmetic mean of the entropies on map and on ground truthing and its variance.

Examples:

```
A<-matrix(c(36,5,4,7,2,2,6,1,20),nrow=3,ncol=3)
p<-MatCon$new(A)
p$nmiam()
```

Method `nmigm()`**:**

$$H(A) = -\sum_{j=1}^{n}((\frac{\sum_{i=1}^{n} x_{i\cdot}}{\sum_{i,j=1}^{n} x_{ij}}) \cdot \log(\frac{\sum_{i=1}^{n} x_{i\cdot}}{\sum_{i,j=1}^{n} x_{ij}}))$$

$$H(B) = -\sum_{i=1}^{n}((\frac{\sum_{j=1}^{n} x_{\cdot j}}{\sum_{i,j=1}^{n} x_{ij}}) \cdot \log(\frac{\sum_{j=1}^{n} x_{\cdot j}}{\sum_{i,j=1}^{n} x_{ij}}))$$

$$nmigm = \frac{ami}{\sqrt{H(A) \cdot H(B)}}$$

where:

1. 'nmigm': Normalized mutual information using the geometric mean of the entropies on map and on ground truthing.
2. 'H(A)': the entropy of the map.
3. 'H(B)': The entropy of the map.
4. 'x_.j': sum with respect to j (rows).
5. 'x_i.': sum with respect to i (columns).
6. 'ami': Average mutual information.

Usage:

```
MatCon$nmigm()
```

Returns: Normalized mutual information using the geometric mean of the entropies on map and on ground truthing and its variance.

Examples:

```
A<-matrix(c(36,5,4,7,2,2,6,1,20),nrow=3,ncol=3)
p<-MatCon$new(A)
p$nmigm()
```

Method `nmimx()`**:**

$$H(A) = -\sum_{j=1}^{n}((\frac{\sum_{i=1}^{n} x_{i\cdot}}{\sum_{i,j=1}^{n} x_{ij}}) \cdot \log(\frac{\sum_{i=1}^{n} x_{i\cdot}}{\sum_{i,j=1}^{n} x_{ij}}))$$

$$H(B) = -\sum_{i=1}^{n}((\frac{\sum_{j=1}^{n} x_{\cdot j}}{\sum_{i,j=1}^{n} x_{ij}}) \cdot \log(\frac{\sum_{j=1}^{n} x_{\cdot j}}{\sum_{i,j=1}^{n} x_{ij}}))$$

$$nmimx = \frac{2ami}{max(H(A)) + max(H(B))} = \frac{ami}{\log\sqrt{card(p)}}$$

where:

1. 'nmimx': Normalized mutual information using the arithmetic mean of the maximum entropies on map and on ground truthing
2. 'H(A)': the entropy of the map.
3. 'H(B)': The entropy of the map.
4. 'x_.j': sum with respect to j (rows).
5. 'x_i.': sum with respect to i (columns).
6. 'ami': Average mutual information.

Usage:

```
MatCon$nmimx()
```

Returns: Normalized mutual information using the arithmetic mean of the maximum entropies on map and on ground truthing and its variance.

Examples:

```
A<-matrix(c(36,5,4,7,2,2,6,1,20),nrow=3,ncol=3)
p<-MatCon$new(A)
p$nmimx()
```

Method `MPseudozeroes()`**:** Small values are calculated for empty cells of the matrix. All non-empty cells of the matrix change their values. This function will not be applied if all the elements of the matrix are different from 0.

Usage:

```
MatCon$MPseudozeroes()
```

Returns: An object of class MatCon with the matrix of Pseudoceros.

Examples:

```
A <- t(matrix(c(35, 14,11,1,4,11,3,0,12,9,38,4,2,5,12,2), nrow = 4, ncol=4))
p<-MatCon$new(A)
p$MPseudozeroes()
```

Method `MTypify()`**:** Cell values are typified. The total sum of the original matrix is used for the typification. Resulting values can be presented as real (parameter RaR=1) or as percentage (parameter RaR !=1)

$$MTypify = \frac{x_{ij}}{\sum_{i,j=1}^{n} x_{ij}}$$

where:

1. 'MTyipify': typified matrix.
2. 'x_ij': matrix element.
3. 'sumx_ij': sum of all the elements of the matrix.

Usage:

```
MatCon$MTypify(RaR = NULL)
```

Arguments:

`RaR` "1" indicates result as real, other values mean percentage as integer. By default RaR=1

Returns: Typified matrix

Examples:

```
x <- t(matrix(c(35, 14,11,1,4,11,3,0,12,9,38,4,2,5,12,2), nrow = 4, ncol=4))
p<-MatCon$new(x)
p$MTypify(RaR=5)
```

Method `MAllParameters()`**:** Several parameters are calculated for the given Confusion Matrix. The.

Usage:

```
MatCon$MAllParameters()
```

Returns: Confusion Matrix, Dimension, Total sum of cell values, Overall Accuracy, Variance overall accuracy, Kappa index of global accuracy, Simplified variance of the global Kappa, per-clas producer's accuracy, per-class user's accuracy, k value for the calculation of pseudozeroes, Pseudozeroes Matrix, L matrix for the calculation of pseudozeroes.

Examples:

```
x <- t(matrix(c(35, 14,11,1,4,11,3,0,12,9,38,4,2,5,12,2), nrow = 4, ncol=4))
p<-MatCon$new(x)
p$MAllParameters()
```

Method `CAccuracies()`**:** User's and producer's accuracies and standard deviations are computed

Usage:

```
MatCon$CAccuracies()
```

Returns: A list with the producer's accuracy, its standard deviation, the user's accuracy, and its standard deviation

Examples:

```
x <- t(matrix(c(35, 14,11,1,4,11,3,0,12,9,38,4,2,5,12,2), nrow = 4, ncol=4))
p<-MatCon$new(x)
p$CAccuracies()
```

Method `CAccuraciesW()`**:** User's and producer's weighted accuracies and standard deviations are computed.

Usage:

```
MatCon$CAccuraciesW(MP)
```

Arguments:

`MP` matrix of weights

Returns: Matrix formed with its original elements and their corresponding weights, general accuracy of the weight matrix obtained, accuracy of the producer and user and their standard deviations,

Examples:

```
x <- t(matrix(c(35, 14,11,1,4,11,3,0,12,9,38,4,2,5,12,2), nrow = 4, ncol=4))
p<-MatCon$new(x)
MP<- t(matrix(c(1,0,0.67,1,0,1,0,0,1,0,1,1,0.91,0,0.61,1), nrow = 4, ncol=4))
p$CAccuraciesW(MP)
```

Method `DetailedKappa()`**:** Overall Kappa agreement index and variance elements are computed

Usage:

```
MatCon$DetailedKappa()
```

Returns: Overall accuracy, Expected accuracy of agreement if agreement were purely random,,, coefficient kappa, standar desviation kappa,

Examples:

```
x <- t(matrix(c(35, 14,11,1,4,11,3,0,12,9,38,4,2,5,12,2), nrow = 4, ncol=4))
p$DetailedKappa()
```

Method `DetailedCKappa()`**:** Class Kappa agreement index (conditional Kappa) and its variance are computed

Usage:

```
MatCon$DetailedCKappa()
```

Examples:

```
x <- t(matrix(c(35, 14,11,1,4,11,3,0,12,9,38,4,2,5,12,2), nrow = 4, ncol=4))
p$DetailedCKappa ()
```

Method `DetailedWKappa()`**:** Overall Kappa agreement index (Weighted) and its variance are computed

Usage:

```
MatCon$DetailedWKappa(MW)
```

Arguments:

`MW` matrix of weights

Examples:

```
x <- t(matrix(c(35, 14,11,1,4,11,3,0,12,9,38,4,2,5,12,2), nrow = 4, ncol=4))
MW<- t(matrix(c(1,0,0.67,1,0,1,0,0,1,0,1,1,0.91,0,0.61,1), nrow = 4, ncol=4))
p<-MatCon$new(x)
p$DetailedWKappa(MW)
```

Method `DetailedTau()`**:** Overall Tau agreement index and variance elements are computed

Usage:

```
MatCon$DetailedTau(VP)
```

Arguments:

`VP` vector of proportions (as matrix)

Examples:

```
x <- t(matrix(c(35, 14,11,1,4,11,3,0,12,9,38,4,2,5,12,2), nrow = 4, ncol=4))
VP <-matrix(c(0.4, 0.1, 0.4, 0.1), ncol=4)
p<-MatCon$new(x)
p$DetailedTau(VP)
```

Method `QES()`**:** Quantity, Exchange and Shift values are computed

Usage:

```
MatCon$QES(TI = 1, SF = 1)
```

Arguments:

`TI` time interval (default value = 1)

`SF` Scale factor for results (default value = 1)

Examples:

```
x <- t(matrix(c(0,1,1,2,0,0,0,2,0), nrow =3, ncol=3))
p<-MatCon$new(x)
p$QES(TI=1, SF=6)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
MatCon$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

References

Story, M., & Congalton, R. G. (1986). Accuracy assessment: a user's perspective. Photogrammetric Engineering and remote sensing, 52(3), 397-399.

Koukoulas, S., & Blackburn, G. A. (2001). Introducing new indices for accuracy evaluation of classified images representing semi-natural woodland environments. Photogrammetric Engineering and Remote Sensing, 67(4), 499-510.

Turk, G. (2002). Map evaluation and" chance correction". Photogrammetric Engineering and Remote Sensing, 68(2), 123-129.

Helldén, U. (1980). A test of landsat-2 imagery and digital data for thematic mapping illustrated by an environmental study in northern Kenya, Lund University. Natural Geography Institute Report No. 47.

Rosenfield, G. H., & Fitzpatrick-Lins, K. (1986). A coefficient of agreement as a measure of thematic classification accuracy. Photogrammetric engineering and remote sensing, 52(2), 223-227.

Short, N. M. (1982). The Landsat tutorial workbook: Basics of satellite remote sensing (Vol. 1078). National Aeronautics and Space Administration, Scientific and Technical Information Branch.

Stehman, S. V. (1997). Selecting and interpreting measures of thematic classification accuracy. Remote sensing of Environment, 62(1), 77-89.

Finn, J. T. (1993). Use of the average mutual information index in evaluating classification error and consistency. International Journal of Geographical Information Science, 7(4), 349-366.

Tung, F., & LeDrew, E. (1988). The determination of optimal threshold levels for change detection using various accuracy indexes. Photogrammetric Engineering and Remote Sensing, 54(10), 1449-1454.

Fienberg, S. E. (1970). An iterative procedure for estimation in contingency tables. The Annals of Mathematical Statistics, 41(3), 907-917.

Liu, C., Frazier, P., & Kumar, L. (2007). Comparative assessment of the measures of thematic classification accuracy. Remote sensing of environment, 107(4), 606-616.

Cohen, J. (1960). A coefficient of agreement for nominal scales. Educational and psychological measurement, 20(1), 37-46.

Ghosh, J., Strehl, A., & Merugu, S. (2002, November). A consensus framework for integrating distributed clusterings under limited knowledge sharing. In Proc. NSF Workshop on Next Generation Data Mining (pp. 99-108).

Strehl, A. (2002). Relationship-based clustering and cluster ensembles for high-dimensional data mining. The University of Texas at Austin.

Muñoz, J. M. S. (2016). Análisis de Calidad Cartográfica mediante el estudio de la Matriz de Confusión. Pensamiento matemático, 6(2), 9-26.

Examples

```
C = matrix( c(5, 0, 1, 0,4,0,0,0,3), nrow=3,  ncol=3)
mc2 <- MatCon$new (C,ID=5,Date="27-10-2023")

## ------------------------------------------------
## Method `MatCon$oa`
## ------------------------------------------------

A<-matrix(c(36,1,0,0,2,0,0,1,20),nrow=3,ncol=3)
p<-MatCon$new(A)
p$oa()

## ------------------------------------------------
## Method `MatCon$dec`
## ------------------------------------------------

A<-matrix(c(36,1,0,0,2,0,0,1,20),nrow=3,ncol=3)
p<-MatCon$new(A)
p$dec()

## ------------------------------------------------
## Method `MatCon$ua`
## ------------------------------------------------

A<-matrix(c(36,1,0,0,2,0,0,1,20),nrow=3,ncol=3)
p<-MatCon$new(A,ID=1,Date="30/10/2023")
p$ua()

## ------------------------------------------------
## Method `MatCon$uai`
## ------------------------------------------------

A<-matrix(c(36,1,0,0,2,0,0,1,20),nrow=3,ncol=3)
p<-MatCon$new(A)
p$uai(2)

## ------------------------------------------------
## Method `MatCon$pai`
## ------------------------------------------------

A<-matrix(c(36,1,0,0,2,0,0,1,20),nrow=3,ncol=3)
p<-MatCon$new(A)
p$pai(1)

## ------------------------------------------------
## Method `MatCon$pa`
## ------------------------------------------------
```

```
A<-matrix(c(36,1,0,0,2,0,0,1,20),nrow=3,ncol=3)
p<-MatCon$new(A)
p$pa()

## ------------------------------------------------
## Method `MatCon$aup`
## ------------------------------------------------

A<-matrix(c(36,1,0,0,2,0,0,1,20),nrow=3,ncol=3)
p<-MatCon$new(A)
p$aup(2)

## ------------------------------------------------
## Method `MatCon$ICSI`
## ------------------------------------------------

A<-matrix(c(36,1,0,0,2,0,0,1,20),nrow=3,ncol=3)
p<-MatCon$new(A)
p$ICSI(2)

## ------------------------------------------------
## Method `MatCon$mah`
## ------------------------------------------------

A<-matrix(c(36,1,0,0,2,0,0,1,20),nrow=3,ncol=3)
p<-MatCon$new(A)
p$mah(2)

## ------------------------------------------------
## Method `MatCon$mas`
## ------------------------------------------------

A<-matrix(c(36,1,0,0,2,0,0,1,20),nrow=3,ncol=3)
p<-MatCon$new(A)
p$mas(2)

## ------------------------------------------------
## Method `MatCon$cku`
## ------------------------------------------------

A<-matrix(c(36,1,0,0,2,0,0,1,20),nrow=3,ncol=3)
p<-MatCon$new(A)
p$cku(2)

## ------------------------------------------------
## Method `MatCon$ckp`
## ------------------------------------------------

A<-matrix(c(36,1,0,0,2,0,0,1,20),nrow=3,ncol=3)
p<-MatCon$new(A)
p$ckp(2)

## ------------------------------------------------
## Method `MatCon$mcku`
## ------------------------------------------------
```

```
A<-matrix(c(36,1,0,0,2,0,0,1,20),nrow=3,ncol=3)
p<-MatCon$new(A)
p$mcku(2)

## ------------------------------------------------
## Method `MatCon$mckp`
## ------------------------------------------------

A<-matrix(c(36,1,0,0,2,0,0,1,20),nrow=3,ncol=3)
p<-MatCon$new(A)
p$mckp(2)

## ------------------------------------------------
## Method `MatCon$ecnu`
## ------------------------------------------------

A<-matrix(c(36,1,0,0,2,0,0,1,20),nrow=3,ncol=3)
p<-MatCon$new(A)
p$ecnu(2)

## ------------------------------------------------
## Method `MatCon$ecnp`
## ------------------------------------------------

A<-matrix(c(36,5,4,7,2,2,6,1,20),nrow=3,ncol=3)
p<-MatCon$new(A)
p$ecnp(2)

## ------------------------------------------------
## Method `MatCon$aau`
## ------------------------------------------------

A<-matrix(c(36,1,0,0,2,0,0,1,20),nrow=3,ncol=3)
p<-MatCon$new(A)
p$aau()

## ------------------------------------------------
## Method `MatCon$Normalize`
## ------------------------------------------------

A<-matrix(c(36,1,0,0,2,0,0,1,20),nrow=3,ncol=3)
p<-MatCon$new(A)
p$Normalize()
p$Normalize(1000)

## ------------------------------------------------
## Method `MatCon$aap`
## ------------------------------------------------

A<-matrix(c(36,1,0,0,2,0,0,1,20),nrow=3,ncol=3)
p<-MatCon$new(A)
p$aap()

## ------------------------------------------------
## Method `MatCon$daup`
## ------------------------------------------------
```

```
A<-matrix(c(36,1,0,0,2,0,0,1,20),nrow=3,ncol=3)
p<-MatCon$new(A)
p$daup()

## ------------------------------------------------
## Method `MatCon$CSI`
## ------------------------------------------------

A<-matrix(c(36,1,0,0,2,0,0,1,20),nrow=3,ncol=3)
p<-MatCon$new(A)
p$CSI()

## ------------------------------------------------
## Method `MatCon$amah`
## ------------------------------------------------

A<-matrix(c(36,1,0,0,2,0,0,1,20),nrow=3,ncol=3)
p<-MatCon$new(A)
p$amah()

## ------------------------------------------------
## Method `MatCon$amas`
## ------------------------------------------------

A<-matrix(c(36,1,0,0,2,0,0,1,20),nrow=3,ncol=3)
p<-MatCon$new(A)
p$amas()

## ------------------------------------------------
## Method `MatCon$cau`
## ------------------------------------------------

A<-matrix(c(36,1,0,0,2,0,0,1,20),nrow=3,ncol=3)
p<-MatCon$new(A)
p$cau()

## ------------------------------------------------
## Method `MatCon$cap`
## ------------------------------------------------

A<-matrix(c(36,1,0,0,2,0,0,1,20),nrow=3,ncol=3)
p<-MatCon$new(A)
p$cap()

## ------------------------------------------------
## Method `MatCon$caup`
## ------------------------------------------------

A<-matrix(c(36,1,0,0,2,0,0,1,20),nrow=3,ncol=3)
p<-MatCon$new(A)
p$caup()

## ------------------------------------------------
## Method `MatCon$KappaValue`
## ------------------------------------------------

A<-matrix(c(36,1,0,0,2,0,0,1,20),nrow=3,ncol=3)
```

```
p<-MatCon$new(A)
p$KappaValue()

## ------------------------------------------------
## Method `MatCon$mkp`
## ------------------------------------------------

A<-matrix(c(36,1,0,0,2,0,0,1,20),nrow=3,ncol=3)
p<-MatCon$new(A)
p$mkp()

## ------------------------------------------------
## Method `MatCon$ami`
## ------------------------------------------------

A<-matrix(c(36,5,4,7,2,2,6,1,20),nrow=3,ncol=3)
p<-MatCon$new(A)
p$ami()

## ------------------------------------------------
## Method `MatCon$nmiu`
## ------------------------------------------------

A<-matrix(c(36,5,4,7,2,2,6,1,20),nrow=3,ncol=3)
p<-MatCon$new(A)
p$nmiu()

## ------------------------------------------------
## Method `MatCon$nmip`
## ------------------------------------------------

A<-matrix(c(36,5,4,7,2,2,6,1,20),nrow=3,ncol=3)
p<-MatCon$new(A)
p$nmip()

## ------------------------------------------------
## Method `MatCon$nmiam`
## ------------------------------------------------

A<-matrix(c(36,5,4,7,2,2,6,1,20),nrow=3,ncol=3)
p<-MatCon$new(A)
p$nmiam()

## ------------------------------------------------
## Method `MatCon$nmigm`
## ------------------------------------------------

A<-matrix(c(36,5,4,7,2,2,6,1,20),nrow=3,ncol=3)
p<-MatCon$new(A)
p$nmigm()

## ------------------------------------------------
## Method `MatCon$nmimx`
## ------------------------------------------------

A<-matrix(c(36,5,4,7,2,2,6,1,20),nrow=3,ncol=3)
p<-MatCon$new(A)
```

```
p$nmimx()

## ------------------------------------------------
## Method `MatCon$MPseudozeroes`
## ------------------------------------------------

A <- t(matrix(c(35, 14,11,1,4,11,3,0,12,9,38,4,2,5,12,2), nrow = 4, ncol=4))
p<-MatCon$new(A)
p$MPseudozeroes()

## ------------------------------------------------
## Method `MatCon$MTypify`
## ------------------------------------------------

x <- t(matrix(c(35, 14,11,1,4,11,3,0,12,9,38,4,2,5,12,2), nrow = 4, ncol=4))
p<-MatCon$new(x)
p$MTypify(RaR=5)

## ------------------------------------------------
## Method `MatCon$MAllParameters`
## ------------------------------------------------

x <- t(matrix(c(35, 14,11,1,4,11,3,0,12,9,38,4,2,5,12,2), nrow = 4, ncol=4))
p<-MatCon$new(x)
p$MAllParameters()

## ------------------------------------------------
## Method `MatCon$CAccuracies`
## ------------------------------------------------

x <- t(matrix(c(35, 14,11,1,4,11,3,0,12,9,38,4,2,5,12,2), nrow = 4, ncol=4))
p<-MatCon$new(x)
p$CAccuracies()

## ------------------------------------------------
## Method `MatCon$CAccuraciesW`
## ------------------------------------------------

x <- t(matrix(c(35, 14,11,1,4,11,3,0,12,9,38,4,2,5,12,2), nrow = 4, ncol=4))
p<-MatCon$new(x)
MP<- t(matrix(c(1,0,0.67,1,0,1,0,0,1,0,1,1,0.91,0,0.61,1), nrow = 4, ncol=4))
p$CAccuraciesW(MP)

## ------------------------------------------------
## Method `MatCon$DetailedKappa`
## ------------------------------------------------

x <- t(matrix(c(35, 14,11,1,4,11,3,0,12,9,38,4,2,5,12,2), nrow = 4, ncol=4))
p$DetailedKappa()

## ------------------------------------------------
## Method `MatCon$DetailedCKappa`
## ------------------------------------------------

x <- t(matrix(c(35, 14,11,1,4,11,3,0,12,9,38,4,2,5,12,2), nrow = 4, ncol=4))
p$DetailedCKappa ()
```

```
## ------------------------------------------------
## Method `MatCon$DetailedWKappa`
## ------------------------------------------------

x <- t(matrix(c(35, 14,11,1,4,11,3,0,12,9,38,4,2,5,12,2), nrow = 4, ncol=4))
MW<- t(matrix(c(1,0,0.67,1,0,1,0,0,1,0,1,1,0.91,0,0.61,1), nrow = 4, ncol=4))
p<-MatCon$new(x)
p$DetailedWKappa(MW)

## ------------------------------------------------
## Method `MatCon$DetailedTau`
## ------------------------------------------------

x <- t(matrix(c(35, 14,11,1,4,11,3,0,12,9,38,4,2,5,12,2), nrow = 4, ncol=4))
VP <-matrix(c(0.4, 0.1, 0.4, 0.1), ncol=4)
p<-MatCon$new(x)
p$DetailedTau(VP)

## ------------------------------------------------
## Method `MatCon$QES`
## ------------------------------------------------

x <- t(matrix(c(0,1,1,2,0,0,0,2,0), nrow =3, ncol=3))
p<-MatCon$new(x)
p$QES(TI=1, SF=6)
```

Index