Package ‘PaolaR6Nuevo’

November 15, 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 or or an error if a matrix is not entered.

- `Error type 1`: Non-square matrix.
- `Error type 2`: Single element matrix.
- `Error type 3`: negative values.
- `Error type 4`: Sum of elements 0.
- `Error type 5`: Sum of rows 0.
- `Error type 6`: Sum of columns 0.
- `Error type 7`: It is not a matrix.

Methods

Public methods:

- `MatCon$new()`
- `MatCon$oa()`
- `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$aau()`
- `MatCon$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$nmimx()`
- `MatCon$BootStrap()`
- `MatCon$Normalize()`
- `MatCon$MPseudozeroes()`
- `MatCon$MTypify()`
- `MatCon$MAllParameters()`
- `MatCon$CAccuracies()`
- `MatCon$CAccuraciesW()`
- `MatCon$DetailedKappa()`
- `MatCon$DetailedCKappa()`
- `MatCon$DetailedWKappa()`
- `MatCon$Tau()`
- `MatCon$DetailedTau()`
- `MatCon$QES()`
- `MatCon$clone()`

Method `new()`**:** Creates a new instance of this [R6][R6::R6Class] class.

Usage:

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

Arguments:

`values` Confusion matrix

`ID` Identifier. By default ID is a random number between 1 and 1000.

`Date` Date provided by the user. By default the date provided by the system will be taken.

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. See reference [1].
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()
```

Returns: Overall accuracy and variance.

Examples:

```
A<-matrix(c(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
p<-MatCon$new(A)
p$oa()
```

Method `ua()`**:** The accuracy from the point of view of a map user, not the map maker. See reference [1].
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()
```

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

Examples:

```
A<-matrix(c(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
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. See reference [1].

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

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(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
p<-MatCon$new(A)
p$uai(2)
```

Method `pai()`**:** The map accuracy from the point of view of the map maker (the producer). See reference [1].

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

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(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
p<-MatCon$new(A)
p$pai(1)
```

Method `pa()`**:** The map accuracy from the point of view of the map maker (the producer). See reference [1].

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

where:

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

Usage:

```
MatCon$pa()
```

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

Examples:

```
A<-matrix(c(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
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). See reference [2].

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

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(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
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. See reference [3,4].

$$ICSI = ua_i + pa_i - 1$$

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(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
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. See references [5,6].

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

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.

Returns: Hellden’s mean accuracy.

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$new(A)
p$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). See references [6,7].

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

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(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
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. See reference [6].

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

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(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
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. See reference [6].

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

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(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
p<-MatCon$new(A)
p$ckp(2)
```

Method `mcku()`**:** Modified kappa index for the user. See reference [8]

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

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(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
p<-MatCon$new(A)
p$mcku(2)
```

Method `mckp()`**:** Modified kappa index for the producer. See reference [8].

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

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(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
p<-MatCon$new(A)
p$mckp(4)
```

Method `ecnu()`**:** Relative entropy is a quantity that measures the difference between two maps. See reference [9].

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

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

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

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(Alb_i)‘: Entropy of map A knowing that the location corresponding to map B is in class b_i.

Usage:

```
MatCon$ecnu(i, v = NULL)
```

Arguments:

`i` Class to evaluate (row).

`v` Base of the logarithm. By default v=10. This value is used for the entropy units, v=10(Hartleys), v=2(bits), v=e(nats).

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

Examples:

```
A<-matrix(c(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
p<-MatCon$new(A)
p$ecnu(2)
```

Method `ecnp()`**:** Relative entropy is a quantity that measures the difference between two ground truthing. See reference [8].

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

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

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

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, v = NULL)
```

Arguments:

`i` Class to evaluate

`v` Base of the logarithm. By default v=10. This value is used for the entropy units, v=10(Hartleys), v=2(bits), v=e(nats).

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

Examples:

```
A<-matrix(c(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
p<-MatCon$new(A)
p$ecnp(4)
```

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. See reference [10].

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

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(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
p<-MatCon$new(A)
p$aau()
```

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. See reference [10].

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

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 <- 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$new(A)
p$aap()
```

Method `daup()`**:** It is the average of the average accuracy from user's and producer's perspective. See reference [2].

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

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(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
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. See reference [3,4].

$$CSI = aau + aap - 1$$

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(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
p<-MatCon$new(A)
p$CSI()
```

Method `amah()`**:** This function provides the average value of the Hellden mean precision index. See reference [2].

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

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(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
p<-MatCon$new(A)
p$amah()
```

Method `amas()`**:** This function provides the average of Short’s mapping accuracy index. See reference [2].

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

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(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
p<-MatCon$new(A)
p$amas()
```

Method `cau()`**:** The combined accuracy is the average of the overall accuracy and average accuracy. See reference [10].

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

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(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
p<-MatCon$new(A)
p$cau()
```

Method `cap()`**:** The combined accuracy is the average of the overall accuracy and average accuracy. See reference [10].

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

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(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
p<-MatCon$new(A)
p$cap()
```

Method `caup()`**:** The combined accuracy is the average of the overall accuracy and average accuracy. See reference [2].

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

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(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
p<-MatCon$new(A)
p$caup()
```

Method `KappaValue()`**:** It measures the relationship of beyond chance agreement to expected disagreement. See reference [11].

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

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(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
p<-MatCon$new(A)
p$KappaValue()
```

Method `mkp()`**:** It is the proportion of agreement after chance agreement is removed from consideration. See reference [2].

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

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(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
p<-MatCon$new(A)
p$mkp()
```

Method `ami()`**:** Average mutual information (AMI), is applied to the comparison of thematic maps. See reference [9].

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

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(v = NULL)
```

Arguments:

`v` Base of the logarithm. By default v=10. This value is used for the entropy units, v=10(Hartleys), v=2(bits), v=e(nats).

Returns: Average mutual information and its variance.

Examples:

```
A<-matrix(c(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
p<-MatCon$new(A)
p$ami()
```

Method `nmiu()`**:** See reference [9].

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

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

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(v = NULL)
```

Arguments:

v Base of the logarithm. By default v=10. This value is used for the entropy units, v=10(Hartleys), v=2(bits), v=e(nats).

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

Examples:

```
A<-matrix(c(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
p<-MatCon$new(A)
p$nmiu()
```

Method `nmip()`**:** See reference [9].

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

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

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(v = NULL)
```

Arguments:

v Base of the logarithm. By default v=10. This value is used for the entropy units, v=10(Hartleys), v=2(bits), v=e(nats).

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

Examples:

```
A<-matrix(c(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
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(v = NULL)
```

Arguments:

v Base of the logarithm. By default v=10. This value is used for the entropy units, v=10(Hartleys), v=2(bits), v=e(nats).

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

Examples:

```
A<-matrix(c(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
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(v = NULL)
```

Arguments:

v Base of the logarithm. By default v=10. This value is used for the entropy units, v=10(Hartleys), v=2(bits), v=e(nats).

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

Examples:

```
A<-matrix(c(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
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(v = NULL)
```

Arguments:

`v` Base of the logarithm. By default v=10. This value is used for the entropy units, v=10(Hartleys), v=2(bits), v=e(nats).

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(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
p<-MatCon$new(A)
p$nmimx()
```

Method `BootStrap()`: N resamples of the confusion matrix are performed. See reference [14].

Usage:

```
MatCon$BootStrap(n)
```

Arguments:

`n` Number of resamples.

Returns: n simulated matrices, from the confusion matrix, applying the multinomial distribution

Examples:

```
A<-matrix(c(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
p<-MatCon$new(A)
p$BootStrap(2)
```

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. See reference [15,16].

Usage:

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

Arguments:

`n` Iteration. By default n=100.

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

Examples:

```
A<-matrix(c(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
p<-MatCon$new(A)
p$Normalize()$values
```

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. See reference [16].

Usage:

```
MatCon$MPseudozeroes()
```

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

Examples:

```
A<-matrix(c(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
p<-MatCon$new(A)
p$MPseudozeroes()$values
```

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:

```
A<-matrix(c(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
p<-MatCon$new(A)
p$MTypify(RaR=5)
```

Method `MAllParameters()`**:** Several parameters are calculated for the given Confusion Matrix. See references [1,11,16].

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:

```
A<-matrix(c(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
p<-MatCon$new(A)
p$MAllParameters()
```

Method `CAccuracies()`**:** User's and producer's accuracies and standard deviations are computed. See reference [1].

Usage:

```
MatCon$CAccuracies()
```

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

Examples:

```
A<-matrix(c(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
p<-MatCon$new(A)
p$CAccuracies()
```

Method `CAccuraciesW()`**:** User's and producer's weighted accuracies and standard deviations are computed. See reference [1].

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:

```
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$new(A)
MP<- t(matrix(c(1,0,0.67,1,0,1,0,0,1,0,1,1,0.91,0,0.61,1), nrow = 4, ncol=4))
p$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:

```
A<-matrix(c(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
p<-MatCon$new(A)
p$DetailedKappa()
```

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

Usage:

`MatCon$DetailedCKappa()`

Returns: Kappa index and its standard deviation.

Examples:

```
A<-matrix(c(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
p<-MatCon$new(A)
p$DetailedCKappa ()
```

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

Usage:

`MatCon$DetailedWKappa(MW)`

Arguments:

`MW` Matrix of weights.

Returns: Kappa index and its variance.

Examples:

```
A <- A<-matrix(c(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),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$new(A)
p$DetailedWKappa(MW)
```

Method `Tau()`**:** Calculate the tau index and its variance. Its value indicates how much the classification has improved compared to a random classification of the N elements into M groups. See reference [17].

Usage:

`MatCon$Tau()`

Returns: Tau index and its variance.

Examples:

```
A<-matrix(c(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
p<-MatCon$new(A)
p$Tau()
```

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

Usage:

`MatCon$DetailedTau(VP)`

Arguments:

`VP` Vector of proportions (as matrix)

Returns: NO LO VEO CLARO. Overall accuracy index, producer accurancy index, O3,O4, Tau index?(mirar definicion en funcion) y its standard desviation.

Examples:

```
A<-matrix(c(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
VP <-matrix(c(0.4, 0.1, 0.4, 0.1), ncol=4)
p<-MatCon$new(A)
p$DetailedTau(VP)
```

Method `QES()`**:** Quantity, Exchange and Shift values are computed. See reference [18].

Usage:

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

Arguments:

`TI` Time interval (default value = 1)

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

Returns: NO VEO MUY CLARO QUE HACE

Examples:

```
A<-matrix(c(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
p<-MatCon$new(A)
p$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

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

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

[3] 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.

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

[5] 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.

[6] 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.

[7] 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.

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

[9] 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.

[10] 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.

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

[11] Strehl, A., & Ghosh, J. (2002). Cluster ensembles—a knowledge reuse framework for combining multiple partitions. Journal of machine learning research, 3(Dec), 583-617.

[12] 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).

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

[14] Ariza, F. J., Pinilla, C., & Garcia, J. L. (2011). Comparación de matrices de confusión celda a celda mediante bootstraping.

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

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

[17] Ariza-López, F. J. (2013). Fundamentos de evaluación de la calidad de la información geográfica. Universidad de Jaén. Servicio de Publicaciones.

[18] Pontius Jr, R. G., & Santacruz, A. (2014). Quantity, exchange, and shift components of difference in a square contingency table. International Journal of Remote Sensing, 35(21), 7543-7554.

Examples

```
A<-matrix(c(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
mc <- MatCon$new (A,ID=5,Date="27-10-2023")


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

A<-matrix(c(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
p<-MatCon$new(A)
p$oa()


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

A<-matrix(c(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
p<-MatCon$new(A,ID=1,Date="30/10/2023")
p$ua()
```

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

A<-matrix(c(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
p<-MatCon$new(A)
p$uai(2)


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

A<-matrix(c(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
p<-MatCon$new(A)
p$pai(1)


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

A<-matrix(c(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
p<-MatCon$new(A)
p$pa()


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

A<-matrix(c(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
p<-MatCon$new(A)
p$aup(2)


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

A<-matrix(c(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
p<-MatCon$new(A)
p$ICSI(2)


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

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$new(A)
p$mah(2)


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

```
A<-matrix(c(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
p<-MatCon$new(A)
p$mas(2)


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

A<-matrix(c(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
p<-MatCon$new(A)
p$cku(2)


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

A<-matrix(c(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
p<-MatCon$new(A)
p$ckp(2)


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

A<-matrix(c(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
p<-MatCon$new(A)
p$mcku(2)


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

A<-matrix(c(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
p<-MatCon$new(A)
p$mckp(4)


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

A<-matrix(c(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
p<-MatCon$new(A)
p$ecnu(2)


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

A<-matrix(c(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
p<-MatCon$new(A)
```

```
p$ecnp(4)


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

A<-matrix(c(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
p<-MatCon$new(A)
p$aau()


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

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$new(A)
p$aap()


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

A<-matrix(c(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
p<-MatCon$new(A)
p$daup()


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

A<-matrix(c(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
p<-MatCon$new(A)
p$CSI()


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

A<-matrix(c(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
p<-MatCon$new(A)
p$amah()


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

A<-matrix(c(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
p<-MatCon$new(A)
p$amas()
```

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

A<-matrix(c(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
p<-MatCon$new(A)
p$cau()


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

A<-matrix(c(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
p<-MatCon$new(A)
p$cap()


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

A<-matrix(c(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
p<-MatCon$new(A)
p$caup()


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

A<-matrix(c(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
p<-MatCon$new(A)
p$KappaValue()


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

A<-matrix(c(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
p<-MatCon$new(A)
p$mkp()


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

A<-matrix(c(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
p<-MatCon$new(A)
p$ami()


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

```
A<-matrix(c(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
p<-MatCon$new(A)
p$nmiu()


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

A<-matrix(c(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
p<-MatCon$new(A)
p$nmip()


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

A<-matrix(c(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
p<-MatCon$new(A)
p$nmiam()


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

A<-matrix(c(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
p<-MatCon$new(A)
p$nmigm()


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

A<-matrix(c(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
p<-MatCon$new(A)
p$nmimx()


## ------------------------------------------------
## Method `MatCon$BootStrap`
## ------------------------------------------------

A<-matrix(c(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
p<-MatCon$new(A)
p$BootStrap(2)


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

A<-matrix(c(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
p<-MatCon$new(A)
```

```
p$Normalize()$values


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

A<-matrix(c(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
p<-MatCon$new(A)
p$MPseudozeroes()$values


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

A<-matrix(c(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
p<-MatCon$new(A)
p$MTypify(RaR=5)


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

A<-matrix(c(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
p<-MatCon$new(A)
p$MAllParameters()


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

A<-matrix(c(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
p<-MatCon$new(A)
p$CAccuracies()


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

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$new(A)
MP<- t(matrix(c(1,0,0.67,1,0,1,0,0,1,0,1,1,0.91,0,0.61,1), nrow = 4, ncol=4))
p$CAccuraciesW(MP)


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

A<-matrix(c(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
p<-MatCon$new(A)
p$DetailedKappa()
```

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

A<-matrix(c(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
p<-MatCon$new(A)
p$DetailedCKappa ()


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

A <- A<-matrix(c(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),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$new(A)
p$DetailedWKappa(MW)


## ------------------------------------------------
## Method `MatCon$Tau`
## ------------------------------------------------

A<-matrix(c(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
p<-MatCon$new(A)
p$Tau()


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

A<-matrix(c(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
VP <-matrix(c(0.4, 0.1, 0.4, 0.1), ncol=4)
p<-MatCon$new(A)
p$DetailedTau(VP)


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

A<-matrix(c(65,6,0,4,4,81,11,7,22,5,85,3,24,8,19,90),nrow=4,ncol=4)
p<-MatCon$new(A)
p$QES(TI=1, SF=6)
```

QCCS *Quality Control Columns Set*

Description

The p value is calculated using the multinomial distribution from vectors and their corresponding probabilities.

Value

Object of class QCCS

Methods

Public methods:

- `QCCS$new()`
- `QCCS$QCCS()`
- `QCCS$clone()`

Method `new()`: Creates a new instance of this [R6][R6::R6Class] class.

Usage:

```
QCCS$new(vectors, prob)
```

Arguments:

`vectors` vector list.

`p` probabilities list.

Method `QCCS()`: Using a list of vectors and their corresponding probabilities, through a multinomial distribution, the p value is calculated using each of the vectors. See reference [1].

Usage:

```
QCCS$QCCS()
```

Returns: The p value is obtained for each vector, and using the Bonferroni criterion it is decided whether the elements are well classified or not.

Examples:

```
vectors<-list(c(47,4,0),c(40,5,3),c(45,6,2),c(48,0))
prob<-list(c(0.95,0.04,0.01),c(0.88,0.1,0.02),c(0.9,0.08,0.02),c(0.99,0.01))
A <- QCCS$new(vectors,prob)
A$QCCS()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

References

[1] Ariza-López, F. J., Rodríguez-Avi, J., Alba-Fernández, M. V., & García-Balboa, J. L. (2019). Thematic accuracy quality control by means of a set of multinomials. Applied Sciences, 9(20), 4240.

Examples

```
vectors<-list(c(47,4,0),c(40,5,3),c(45,6,2),c(48,0))
prob<-list(c(0.95,0.04,0.01),c(0.88,0.1,0.02),c(0.9,0.08,0.02),c(0.99,0.01))
A <- QCCS$new(vectors,prob)
```

```
## ------------------------------------------------
## Method `QCCS$QCCS`
## ------------------------------------------------

vectors<-list(c(47,4,0),c(40,5,3),c(45,6,2),c(48,0))
prob<-list(c(0.95,0.04,0.01),c(0.88,0.1,0.02),c(0.9,0.08,0.02),c(0.99,0.01))
A <- QCCS$new(vectors,prob)
A$QCCS()
```

Index