Package: Irescale (via r-universe)

October 30, 2024

Type Package

Title Calculate and Rectify Moran's I

Version 2.3.0

Author Ivan Fuentes, Thomas DeWitt, Thomas Ioerger, Michael Bishop

Maintainer Ivan Fuentes <jivfur@tamu.edu>

Description Provides a scaling method to obtain a standardized Moran's I measure. Moran's I is a measure for the spatial autocorrelation of a data set, it gives a measure of similarity between data and its surrounding. The range of this value must be [-1,1], but this does not happen in practice. This package scale the Moran's I value and map it into the theoretical range of [-1,1]. Once the Moran's I value is rescaled, it facilitates the comparison between projects, for instance, a researcher can calculate Moran's I in a city in China, with a sample size of n1 and area of interest a1. Another researcher runs a similar experiment in a city in Mexico with different sample size, n2, and an area of interest a2. Due to the differences between the conditions, it is not possible to compare Moran's I in a straightforward way. In this version of the package, the spatial autocorrelation Moran's I is calculated as proposed in Chen(2013) <arXiv:1606.03658>.

License GPL (>= 2)

URL https://github.tamu.edu/jivfur/rectifiedI

Encoding UTF-8

LazyData true

Imports ggplot2, sp, e1071, graphics, grDevices, stats, utils,Rdpack, fBasics, imager, reshape2

RoxygenNote 6.1.1

RdMacros Rdpack

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Date/Publication 2019-11-21 17:40:02 UTC

Repository https://jivfur.r-universe.dev

RemoteUrl https://github.com/cran/Irescale

RemoteRef HEAD

RemoteSha 796d127996edf301990549753d698aa0da4dd6d1

Contents

`buildStabilityTable` *Finds how many iterations are necessary to achieve stability in resampling method.*

Description

`buildStabilityTable` finds how many iterations are necessary to achieve stability in resampling method, plotting in a log scale.

Usage

```
buildStabilityTable(data, times = 10, samples = 100, plots = TRUE,
  scalingUpTo = "Quantile")
```

Arguments

`data`	data structure after loading the file using `loadFile` function
`times`	the number of times `rescaleI` will be executed. The default value is 100.
`samples`	size of the resampling method. The default value is 1000
`plots`	to draw the significance plot
`scalingUpTo`	the rescaling could be done up to the 0.01% and 99.99% quantile or max and min values. The two possible options are: "MaxMin", or "Quantile". The default value for this parameter is "Quantile"

Value

A vector with the average $\log(samples)$ averages I

Examples

```
fileInput <- system.file("testdata", "chen.csv", package="Irescale")
data <- loadFile(fileInput)
resultsChen<-buildStabilityTable(data=data,times=10,samples=100,plots=TRUE,scalingUpTo="Quantile")
```

`buildStabilityTableForCorrelation`

Finds how many iterations are necessary to achieve stability in resampling method for rectifying I through pearson corrrelation.

Description

`buildStabilityTableForCorrelation` finds how many iterations are necessary to achieve stability in resampling method, plotting in a log scale.

Usage

```
buildStabilityTableForCorrelation(data, times = 10, samples = 100,
  plots = TRUE)
```

Arguments

data	data structure after loading the file using loadFile function
times	the number of times rescaleI will be executed. The default value is 100.
samples	size of the resampling method. The default value is 1000
plots	to draw the significance plot

Value

A vector with the average $\log(samples)$ averages I

Examples

```
fileInput <- system.file("testdata", "chen.csv", package="Irescale")
data <- loadFile(fileInput)
resultsChen<-buildStabilityTableForCorrelation(data=data,times=10,samples=100,plots=TRUE)
```

---

calculateDistMatrixFromBoard

Calculates the distance in a chessboard-alike structure.

Description

calculateDistMatrixFromBoard calculates the distance matrix when the field is divided in a matrix shape (rows and columns). This board could have different number of columns for each row. For example:

1	1	1	1	1	1
2	2	2	2	2	
3	3	3	3	3	
4	4	4	4	4	

The dimension of obtained squared matrix is given by the square of the maximumn dimension of the original matrix. In the previous example, the matrix will have a size of (36,36).

Usage

```
calculateDistMatrixFromBoard(data)
```

Arguments

data	is a 2D data structure.

Value

distM the distance between each cell.

Examples

```
fileInput <- system.file("testdata", "chessboard.csv", package="Irescale")
data<-loadChessBoard(fileInput)
distM<-calculateEuclideanDistance(data$data)
```

`calculateEuclideanDistance` *Given a 2D data structure, it calculates the euclidean distance among all the points.*

Description

`calculateEuclideanDistance` Computes the euclidean distance betwen all pairs of nodes provided in the input vector.

Usage

```
calculateEuclideanDistance(data)
```

Arguments

`data`	2D data structure for latitute and longitute respectively.

Details

Computes the euclidean distance, $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$, matrix between each pair of points.

Value

Matrix, of size $nrow(data) \times nrow(data)$, with the distance between all the pair of points.

Examples

```
fileInput <- system.file("testdata", "chen.csv", package="Irescale")
data<-loadFile(fileInput)
distM<-calculateEuclideanDistance(data$data)
```

calculateLocalI *Computing the Local Moran's I*

Description

calculateLocalI calculates the local Moran's I without rescaling

Usage

```
calculateLocalI(z, distM, scaling = TRUE)
```

Arguments

z	vector with the var of interest
distM	distance matrix
scaling	to scale the variable of interest. The default value is set to TRUE

Value

a vector with the local Moran's I

Examples

```
fileInput <- system.file("testdata", "chen.csv", package="Irescale")
input <- loadFile(fileInput)
distM<-calculateEuclideanDistance(input$data)
localI<-calculateLocalI(input$varOfInterest,distM)
```

calculateManhattanDistance *Calculates the manhattan distance.*

Description

calculateManhattanDistance Calculates the manhattan distance between each pair of nodes.

Usage

```
calculateManhattanDistance(data)
```

Arguments

data	2D structure with n rows and 2 colums that represents the coordinate in a plane.

Value

Matrix, of size $nrow(data) \times nrow(data)$, with the distance between each the pair of points.

Examples

```
fileInput <- system.file("testdata", "chessboard.csv", package="Irescale")
data<-loadChessBoard(fileInput)
distM<-calculateManhattanDistance(data$data)
```

`calculateMoranI` *Calculates the Moran's I using the algorithm proposed by Chen (Chen 2013).*

Description

`calculateMoranI` Moran's I computing method.

$$I = varOfInterest^t \times weightedM \times varOfInterest$$

Usage

```
calculateMoranI(distM, varOfInterest, scaling = TRUE)
```

Arguments

`distM`	the distance matrix. Altough the equation asks for weighted distant matrix, the paramenter that is required is only the distance matrix because this procedure calculate calculates the weighted distance mantrix by itself.
`varOfInterest`	the variable of interest to calculate Moran's I.
`scaling`	if the values are previously scaled, set this parameter to False. The default value is TRUE.

Value

Moran's I

References

Chen Y (2013). "New approaches for calculating Moran's index of spatial autocorrelation." *PloS one*, **8**(7), e68336.

Examples

```
inputFileName<-system.file("testdata", "chen.csv", package="Irescale")
input<-loadFile(inputFileName)
distM<-calculateEuclideanDistance(input$data)
I<-calculateMoranI(distM = distM,varOfInterest = input$varOfInterest)
```

calculatePvalue *p-value calculation.*

Description

calculatePvalue calculates a p-value for the null hypothesis.

Usage

```
calculatePvalue(sample, value, mean)
```

Arguments

sample	the vector that will be used as reference.
value	the value of interest.
mean	the mean of interest.

Examples

```
fileInput<-system.file("testdata", "chen.csv", package="Irescale")
input<-loadFile(fileInput)
distM<-calculateEuclideanDistance(input$data)
I<-calculateMoranI(distM = distM,varOfInterest = input$varOfInterest)
vI<-resamplingI(distM, input$varOfInterest, n=1000) # This is the permutation
statsVI<-summaryVector(vI)
corrections<-iCorrection(I,vI)
pv<-calculatePvalue(corrections$scaledData,corrections$newI,corrections$summaryScaledD$mean)
```

calculateWeightedDistMatrix
Calculates a weighted representation of the distance matrix.

Description

calculateWeightedDistMatrix The weighted matrix is used as a standardized version of the distance matrix.

Usage

```
calculateWeightedDistMatrix(distM)
```

Arguments

distM	2D matrix with the distance among all pair of coordinates.

Details

Computes the similarity matrix of the distance by taking the reciprocal of the distance $\frac{1}{d}$. A value of Zero is assigned when this value can not be calculated. The whole reciprocal matrix is scaled by dividing each value by the sum of all the elements of the matrix.

Value

weighted distance matrix. The sum of this matrix is 1.

Examples

```
fileInput <- system.file("testdata", "chen.csv", package="Irescale")
data<-loadFile(fileInput)
distM<-calculateEuclideanDistance(data$data)
distW<-calculateWeightedDistMatrix(distM)
```

---

convexHull — *Plots the convexhull polygon from the data (latitude, longitude), and calculates the center of the convexhull and its area.*

Description

`convexHull` Computes the area and centroid of the convex hull from the (latitute, longitude) vector. It provides a plot of how the points are dispersed in the field of interest.

Usage

```
convexHull(X, varOfInterest)
```

Arguments

`X`	dataframe with two colums, latitute and longitude respectively.
`varOfInterest`	variable of interest to plot. This variable is needed to color the points on the convexhull.

Details

Consideration for this function:

- It makes usage of chull from rgeos and Polygon from graphics.
- The centroid of the polygon is calculated by averaging the vertices of it.
- The shown plot uses the basic `plot` command.

Value

A vector with two elements, the first element is the area and the second one is the centroid. The centroid is a list of two elements, latitude and longitude that represents the centroid. To have a visual idea of the returned object, it has the following shape $[area, [latitude, longitude], plotObject]$.

Examples

```
fileInput <- system.file("testdata", "chen.csv", package="Irescale")
data<-loadFile(fileInput)
area_centroid<-convexHull(data$data,data$varOfInterest)
```

`coor` *Transforms a x,y position in a cartesian plane into a position in a 1D array.*

Description

`coor` Transforms a x,y position in a cartesian plane into a position in a 1D array.

Usage

```
coor(i, j, size)
```

Arguments

`i`	the value of the row.
`j`	the value of the column.
`size`	the maximum between row and columns of the matrix.

Value

an integer value that represents the position in the array.

Examples

```
pos<-coor(1,1,10)
```

`expectedValueI` *Calculates the expected value for local I*

Description

`expectedValueI` Calculates the expected value for local I

Usage

```
expectedValueI(W)
```

Arguments

`W`	Weighted Distance Matrix.

Value

Expected Value

Examples

```
W<-matrix(1:100,nrow=10,ncol=10)
evI<-expectedValueI(W)
```

`iCorrection` *Scaling process for Moran's I.*

Description

`iCorrection` . consists in centering the I value (I-median) and scaling by the difference between the median and 1st or 99th quantile. The correction is according to the following equation:

$$I = \left\{ \begin{array}{ll} \frac{(I-median)}{(median-Q1)} & I < median \\ \frac{(I-median)}{(Q99-median)} & I > median \end{array} \right\}$$

Usage

```
iCorrection(I, vI, statsVI, scalingUpTo = "Quantile", sd = 1)
```

Arguments

`I`	Moran's I, It could be computed using calculateMoranI function.
`vI`	the vector obtained by resamplingI.
`statsVI`	the statistic vector obtained from summaryVector.
`scalingUpTo`	the rescaling could be done up to the 0.01% and 99.9% quantile or max and min values. The two possible options are: "MaxMin", or "Quantile". The default value for this parameter is Quantile.
`sd`	this represents upto which standard deviation you want to scale I

Value

rescaled I

Examples

```
inputFileName<-system.file("testdata", "chen.csv", package="Irescale")
input<-loadFile(inputFileName)
distM<-calculateEuclideanDistance(input$data)
I<-calculateMoranI(distM = distM,varOfInterest = input$varOfInterest)
vI<-resamplingI(distM, input$varOfInterest)
statsVI<-summaryVector(vI)
corrections<-iCorrection(I,vI,scalingUpTo="Quantile")
```

ItoPearsonCorrelation *Calculate the equivalence r from the I percentile in the I-Null Distribution.*

Description

`ItoPearsonCorrelation` It calculates the Null distribution of I and determine what is the percentile of the real value of I, then It calculates the inverse of the Normal Distribution(qnorm) to obtain the value of R to which this percentile belongs to.

Usage

```
ItoPearsonCorrelation(vI, n, medianCenter = TRUE)
```

Arguments

`vI`	the vector obtained by resamplingI.
`n`	sample size
`medianCenter`	to center all the values to the median. The defaul value is TRUE

Value

a list with r correlation equivalence and the rectified vector

Examples

```
fileInput <- system.file("testdata", "chen.csv", package="Irescale")
data <- loadFile(fileInput)
distM<-calculateEuclideanDistance(data$data)
vI<-resamplingI(distM,data$varOfInterest,n = 1000)
rectifiedI<- ItoPearsonCorrelation(vI, length(data))
```

loadChessBoard *Loads a chessboard or matrix alike input file.*

Description

`loadChessBoard` is used when the input file has a 2D shape, this is a board shape, and it is only one variable of interest. For example:

1	1	1	1	1	1
2	2	2	2	2	
3	3	3	3	3	
4	4	4	4	4	

Usage

```
loadChessBoard(fileName)
```

Arguments

fileName	the path and file's name to load.

Value

data frame with two variables, the first variable is a vector with coordinate x (latitude) and y (longitude), the second variable contains the values of the variable of interest.

Examples

```
fileInput <- system.file("testdata", "chessboard.csv", package="Irescale")
data<-loadChessBoard(fileInput)
```

---

loadDistanceMatrix *Loads a distance matrix. Instead of computing the distance from latitute and longitude* LoadDistanceMatrix *Loads the distance matrix, avoiding computing it from latitude and longitude.*

Description

Loads a distance matrix. Instead of computing the distance from latitute and longitude LoadDistanceMatrix Loads the distance matrix, avoiding computing it from latitude and longitude.

Usage

```
loadDistanceMatrix(fileName, colnames = TRUE, rownames = TRUE)
```

Arguments

fileName	file's name and path to the file
colnames	If the first row of the file is the names for the columns. The default value is TRUE
rownames	If the first column is the the row names. The default value is TRUE

Value

The distance matrix

Examples

```
fileInput <- system.file("testdata", "chenDistance.csv", package="Irescale")
distM<-loadDistanceMatrix(fileInput)
```

loadFile *Loads a file with latitude, longitude and variable of interest*

Description

loadFile loads the input file with the following format:

- Column 1 represents the sample Id. It has to be Unique.
- Column 2,3 Lat/Long respectively.
- Column 4 and beyond the variables of interest.

Usage

```
loadFile(fileName)
```

Arguments

fileName	the file's name and path.

Value

it returns a data frame with two variables $data$ and $varOfInterest$. The variable $data$ is a 2D list with the latitude and longitude respectly, while the variable $varOfInterest$ is a matrix with all the variables to calculate and rescale Moran's I.

Examples

```
fileInput <- system.file("testdata", "chessboard.csv", package="Irescale")
data<-loadFile(fileInput)
```

loadSatelliteImage *Loads a Satellite image in PNG format*

Description

loadSatelliteImage Loads a Satellite image in PNG format. It does not matter the number of chanells it will return it in grayscale (One channel)

Usage

```
loadSatelliteImage(fileName)
```

Arguments

fileName	file's name and path to the file

Value

An cimg object in gray scale.

Examples

```
fileInput <- system.file("testdata", "imageGray.png", package="Irescale")
img<-loadSatelliteImage(fileInput)
```

`localICorrection` *Scaling process for Local Moran's I.*

Description

`localICorrection` . consists in centering the local I value (I-median) and scaling by the difference between the median and 1st or 99th quantile. The correction is according to the following equation:

$$I = \left\{ \begin{array}{ll} \frac{(I-median)}{(median-Q1)} & I < median \\ \frac{(I-median)}{(Q99-median)} & I > median \end{array} \right\}$$

Usage

```
localICorrection(localI, vI, statsVI, scalingUpTo = "Quantile")
```

Arguments

`localI`	Local Moran's I, It could be computed using calculateLocalMoranI function.
`vI`	the vector obtained by resamplingI.
`statsVI`	the statistic vector obtained from summaryLocalIVector.
`scalingUpTo`	the rescaling could be done up to the 0.01% and 99.9% quantile or max and min values. The two possible options are: "MaxMin", or "Quantile". The default value for this parameter is Quantile.

Value

rescaled local I vector

Examples

```
inputFileName<-system.file("testdata", "chen.csv", package="Irescale")
input<-loadFile(inputFileName)
distM<-calculateEuclideanDistance(input$data)
localI <- calculateLocalI(input$varOfInterest,distM)
vI<-resamplingLocalI(input$varOfInterest,distM)
statsVI<-summaryLocalIVector(vI)
corrections<-localICorrection(localI,vI,scalingUpTo="Quantile")
```

`nullDristribution` *Calculate a distribution of how the var of interest is correlated to a*

Description

`nullDristribution` Calculate a linear regression between variable of interest and latitude, longitude and latitude*longitude. The residuals of this data set is calculated The variable of interest is shuffle by numReplicates times and each time the linear regression and residuals are calculated. At each interation the correlation between the original residuals and the shuffle residuals is calculated This vector os correlations is returned and plot it as histogram.

Usage

```
nullDristribution(data, numReplicates)
```

Arguments

`data`	the distance matrix. Altough the equation asks for weighted distant matrix, the paramenter that is required is only the distance matrix because this procedure calculate calculates the weighted distance mantrix by itself.
`numReplicates`	the variable of interest to calculate Moran's I.

Value

Histogram and the vector of correlations between residuals

Examples

```
inputFileName<-system.file("testdata", "chen.csv", package="Irescale")
input<-loadFile(inputFileName)
c<-nullDristribution(input,1000)
```

`plotHistogramOverlayCorrelation`

Creates an overlay of the histogram of the data and the theorical normal distribution.

Description

`plotHistogramOverlayCorrelation` Overlays the histogram and the theorical normal distribution.

Usage

```
plotHistogramOverlayCorrelation(originalVec, vec, I, n, bins = 50,
  main = "Histogram")
```

Arguments

originalVec	The original vector of I, it should be sorted.
vec	the vector to plot.
I	the value of I to plot
n	number of observations in the sample.
bins	the number of bins for the histogram, The default value is 30.
main	the title of the histogram, The default value is "Histogram".

Examples

```
inputFileName<-system.file("testdata", "chen.csv", package="Irescale")
input<-loadFile(inputFileName)
distM<-calculateEuclideanDistance(input$data)
I<-calculateMoranI(distM = distM,varOfInterest = input$varOfInterest)
originalI<-resamplingI(distM, input$varOfInterest)
correlationI<-ItoPearsonCorrelation(originalI,length(input$varOfInterest))
plotHistogramOverlayCorrelation(originalI,correlationI,I,length(input$varOfInterest))
```

---

plotHistogramOverlayNormal

Creates an overlay of the histogram of the data and the theorical normal distribution.

Description

plotHistogramOverlayNormal Overlays the histogram and the theorical normal distribution.

Usage

```
plotHistogramOverlayNormal(vec, stats, bins = 50, main = "Histogram")
```

Arguments

vec	the vector to plot.
stats	the stats obtained from summaryVector.
bins	the number of bins for the histogram, The default value is 30.
main	the title of the histogram, The default value is "Histogram".

Examples

```
inputFileName<-system.file("testdata", "chen.csv", package="Irescale")
input<-loadFile(inputFileName)
distM<-calculateEuclideanDistance(input$data)
I<-calculateMoranI(distM = distM,varOfInterest = input$varOfInterest)
vI<-resamplingI(distM, input$varOfInterest)
statsVI<-summaryVector(vI)
plotHistogramOverlayNormal(vI,statsVI)
```

`procrustes` *Procrustes distance between two surfaces*

Description

`procrustes` Procrustes distance between two surfaces. The Procrustes distance is used to quantify the similarity or dissimilarity of (3-dimensional) shapes, and extensively used in biological morphometrics.

Usage

```
procrustes(U, V)
```

Arguments

`U`	Vector of the first surface.
`V`	Vector of the second surface.

Value

Procrustes distance

`rectifyIrho` *Rectify I using a correlation method for all the variables in an input file.*

Description

`rescaleI` It executes the whole rectifying using theorical R distribution for all the measurements in the csv file.

- It plots the histogram with the theorical distribution.
- It plots the convexHull for each variable.
- It calcualtes the area and centroid of the convex hull for each variable.
- It calculates the I and rescale it for every variable.
- It returns an object with the computations.

Usage

```
rectifyIrho(data, samples = 10000)
```

Arguments

`data`	the data frame obtained from `loadFile`.
`samples`	number of permutations for the resampling method.

Value

An object with I, rescaleI and statistic summary for the inputs without scaling, the same statistics after scaling them, the p-value and the convexhull information

Examples

```
fileInput <- system.file("testdata", "chen.csv", package="Irescale")
data <- loadFile(fileInput)
rectifiedI<-rectifyIrho(data,100)
```

---

resamplingI — *Calculates n permutations of the variable of interest to calculate n different I in order to create the $Null$ distribution.*

Description

`resamplingI` Permute n-1 times the values of the variable of interest to calculate a Null distribution for I. It is done n-1, because one order is the original one, to make sure it is included.

Usage

```
resamplingI(distM, varOfInterest, n = 1000, scaling = TRUE)
```

Arguments

`distM`	the distance matrix. Although the equation requires a weighted distant matrix, the only parameter that will be needed is the distance matrix. This procedure is able to calculate the weighted distance matrix by itself.
`varOfInterest`	the variable name or position of the variable we are interested to calculate the spatial autocorrelation.
`n`	number of permutations. The default value is 1000
`scaling`	The default value is TRUE. However, if the values are previously scaled, this parameter must be set to FALSE..

Value

A vector with the n calculated Moran's I.

Examples

```
inputFileName<-system.file("testdata", "chen.csv", package="Irescale")
input<-loadFile(inputFileName)
distM<-calculateEuclideanDistance(input$data)
I<-calculateMoranI(distM = distM,varOfInterest = input$varOfInterest)
vI<-resamplingI(distM, input$varOfInterest)
```

`resamplingLocalI` *Calculates n permutations of the variable of interest to calculate n different I in order to create the $Null$ distribution.*

Description

`resamplingLocalI` Permute n-1 times the values of the variable of interest to calculate a Null distribution for I. It is done n-1, because one order is the original one, to make sure it is included.

Usage

```
resamplingLocalI(varOfInterest, distM, n = 1000, scaling = TRUE)
```

Arguments

`varOfInterest`	the variable name or position of the variable we are interested to calculate the spatial autocorrelation.
`distM`	the distance matrix. Although the equation requires a weighted distant matrix, the only parameter that will be needed is the distance matrix. This procedure is able to calculate the weighted distance matrix by itself.
`n`	number of permutations.
`scaling`	The default value is TRUE. However, if the values are previously scaled, this parameter must be set to FALSE..

Value

A vector with the n calculated Local Moran's I.

Examples

```
inputFileName<-system.file("testdata", "chen.csv", package="Irescale")
input<-loadFile(inputFileName)
distM<-calculateEuclideanDistance(input$data)
vI<-resamplingLocalI(input$varOfInterest,distM,n=100)
```

`rescaleI` *Performs the rescale for all the variables in an input file.*

Description

`rescaleI` It executes the whole analysis for all the measurements in the field.

- It plots the histogram with the theorical distribution.
- It plots the convexHull for each variable.
- It calcualtes the area and centroid of the convex hull for each variable.
- It calculates the I and rescale it for every variable.
- It returns an object with the computations.

Usage

```
rescaleI(data, samples = 10000, scalingUpTo = "Quantile", sd = 1)
```

Arguments

data	the data frame obtained from loadFile.
samples	number of permutations for the resampling method.
scalingUpTo	the rescaling could be done up to the 0.01% and 99.99% quantile or max and min values. The two possible options are: "MaxMin", or "Quantile". The default value for this parameter is "Quantile"
sd	this represents upto which standard deviation you want to scale I

Value

An object with I, rescaleI and statistic summary for the inputs without scaling, the same statistics after scaling them, the p-value and the convexhull information

Examples

```
fileInput <- system.file("testdata", "chen.csv", package="Irescale")
data <- loadFile(fileInput)
scaledI<-rescaleI(data,100)
```

---

saveFile *Saves a report with important statistics to describe the sample.*

Description

saveFile Saves a csv report with the following columns: Convex Hull Area, Convex Hull Centroid X, Convex Hull Centroid Y, Sample Size, Ichen, Iscaled, pvalue , Mean, MeanScaled, STD DEV, SDScaled, Q_1%, Q_1%Scaled, $Q_99%, Q_99%Scaled, Max, Max_Scaled, Min, Min_Scaled, Skew,Skew_Scaled, Kutorsis,Kutorsis_Scaled.

Usage

```
saveFile(fileName, results)
```

Arguments

fileName	the name of the file with the path where the CSV file will be saved.
results	is the vector obtained from running the rescaling process over all the variables of interest.

Examples

```
fileInput <- system.file("testdata", "chen.csv", package="Irescale")
data <- loadFile(fileInput)
scaledI<-rescaleI(data,1000)
fn = file.path(tempdir(),"output.csv",fsep = .Platform$file.sep)
saveFile(fn,scaledI)
if (file.exists(fn)){
    file.remove(fn)
}
```

`standardize` *Standardize the input vector*

Description

`standardize` Calculates the z-values of the input vector. #'

$$z = \frac{vectorI - meanI}{\sqrt{varI}}$$

Usage

```
standardize(vectorI, W)
```

Arguments

`vectorI`	vector to be standardized.
`W`	weighed distance matrix

Value

z values

Examples

```
W<-matrix(runif(100, min=0, max=1),nrow=10,ncol=10)
vectorI<-runif(10, min=0, max=1)
standardize(vectorI,W)
```

standardizedByColumn *Scales a matrix by column.*

Description

standardizedByColumn It considers each column independently to scale them.

Usage

```
standardizedByColumn(M)
```

Arguments

M	Matrix to be scaled by column.

Value

a matrix scaled by column.

summaryLocalIVector *Calculates statistic for the received Matrix.*

Description

summaryLocalIVector. Calculates basic statistic of the received Matrix, like mean, standard deviation, maximum, minimum, 0.1% and 99.9% quantile and median.

Usage

```
summaryLocalIVector(vec)
```

Arguments

vec	the vector to calculate the summary.

Value

a list with mean, standard deviation, maximum, minimum, 0.1% and 99.9% quantile and median of the received vector.

Examples

```
inputFileName<-system.file("testdata", "chen.csv", package="Irescale")
input<-loadFile(inputFileName)
distM<-calculateEuclideanDistance(input$data)
vI<-resamplingLocalI(input$varOfInterest,distM)
statsVI<-summaryLocalIVector(vI)
```

summaryVector *Calculates statistic for the received vector.*

Description

summaryVector. Calculates basic statistic of the received vector, like mean, standard deviation, maximum, minimum, 0.1% and 99.9% quantile and median.

Usage

```
summaryVector(vec)
```

Arguments

vec	the vector to calculate the summary.

Value

a list with mean, standard deviation, maximum, minimum, 0.1% and 99.9% quantile and median of the received vector.

Examples

```
inputFileName<-system.file("testdata", "chen.csv", package="Irescale")
input<-loadFile(inputFileName)
distM<-calculateEuclideanDistance(input$data)
I<-calculateMoranI(distM = distM,varOfInterest = input$varOfInterest)
vI<-resamplingI(distM, input$varOfInterest)
statsVI<-summaryVector(vI)
```

transformImageToList *Transforms the image in the object need it to run the analysis.*

Description

transformImageToList transforms the image into a list with two variables, data and varOfInterest, which are the identificators needed to execute the rectification.

Usage

```
transformImageToList(im)
```

Arguments

im	cimg object.

Examples

```
fileInput <- system.file("testdata", "imageGray.png", package="Irescale")
img<-loadSatelliteImage(fileInput)
data<-transformImageToList(img)
```

`transformImageToMatrix` *Transforms the image to a matrix.*

Description

`transformImageToMatrix` transforms the image into a 2D matrix.

Usage

```
transformImageToMatrix(im)
```

Arguments

`im`	cimg object.

Examples

```
fileInput <- system.file("testdata", "imageGray.png", package="Irescale")
img<-loadSatelliteImage(fileInput)
data<-transformImageToList(img)
```

Index