Package 'Binarize'

January 20, 2025

Type Package

Title Binarization of One-Dimensional Data

Version 1.3.1

Date 2023-04-02

Author Stefan Mundus, Christoph Müssel, Florian Schmid, Ludwig Lausser, Tamara J. Blätte, Martin Hopfensitz, Hans A. Kestler

Maintainer Hans Kestler <hans.kestler@uni-ulm.de>

Description
Provides methods for the binarization of one-dimensional data and some visualization functions.

License Artistic-2.0

LazyLoad yes

Imports graphics, stats

Depends methods, diptest

Encoding UTF-8

NeedsCompilation yes

Repository CRAN

Date/Publication 2023-04-04 15:30:16 UTC

Contents

`BASCResult-class` *Class "BASCResult"*

Description

A specialized class storing the results of a call to `binarize.BASC`.

Objects of this class

Objects of this class shouldn't be created directly. They are created implicitly by a call to `binarize.BASC`.

Slots

`p.value`**:** The p-value of the statistical test for reliability of the binarization.

`intermediateSteps`**:** A matrix specifying the optimal step functions from which the binarization was calculated. The number of rows corresponds to the number of step functions, and the number of columns is determined by the length of the input vector minus 2 (that is, the length of the step function corresponding to the input vector). From the first to the last row, the number of steps increases. The non-zero entries of the matrix represent the locations of the steps. Step functions with fewer steps than the input step function have entries set to zero.

`intermediateHeights`**:** A matrix giving the jump heights of the steps supplied in `intermediateSteps`.

`intermediateStrongestSteps`**:** A vector with one entry for each step function (row) in `intermediateSteps`. The entries specify the location of the strongest step for each of the functions.

`originalMeasurements`**:** A numeric vector storing the input measurements.

`binarizedMeasurements`**:** An integer vector of binarized values (0 or 1) corresponding to the original measurements.

`threshold`**:** The threshold that separates 0 and 1.

`method`**:** A string describing the binarization method that yielded the result.

Extends

Class `"BinarizationResult"`, directly.

Methods

plotStepFunctions `signature(x = "BASCResult")`: Plot the intermediate optimal step functions used to determine the threshold.

print `signature(x = "BASCResult")`: Print a summary of the binarization.

show `signature(object = "BASCResult")`: ...

See Also

binarize.BASC, BinarizationResult

`binarizationExample` *An artificial data set consisting of ten artificial feature vectors.*

Description

An artificial data set consisting of ten artificial feature vectors that are used to illustrate the binarization methods in the package vignette. Each row of the matrix `binarizationExample` corresponds to one feature vector, of which 10 measurements are drawn from a normal distribution N(0,1). The remaining 10 measurements are drawn from a normal distribution N(m,1), with `m=10:1` decreasing from the first to the last row.

Usage

```
data(binarizationExample)
```

Format

The data is a matrix with 20 columns and 10 rows.

`BinarizationResult-class`
Class "BinarizationResult"

Description

This is the base class for objects that store the results of a binarization algorithm. It defines the slots and methods that the results of all algorithms share.

Objects of this class

Objects of this class shouldn't be created directly. They are created implicitly by a call to one of the binarizeation algorithms.

Slots

`originalMeasurements`**:** A numeric vector storing the input measurements.

`binarizedMeasurements`**:** An integer vector of binarized values (0 or 1) corresponding to the original measurements.

`threshold`**:** The threshold that separates 0 and 1.

`method`**:** A string describing the binarization method that yielded the result.

`p.value`**:** The p-value obtained by a test for validity of the binarization (e.g. BASC bootstrap test, Hartigan's dip test for k-means binarization, scan statistic p-value for best window. If no test was performed, this is `NA`.

Methods

plot `signature(x = "BinarizationResult")`: Plot the binarization and the threshold.

print `signature(x = "BinarizationResult")`: Print a summary of the binarization.

show `signature(object = "BinarizationResult")`: ...

See Also

`binarize.BASC`, `binarize.kMeans`, `BASCResult`,

---

binarize.BASC *Binarization Across Multiple Scales*

Description

Binarizes real-valued data using the multiscale BASC methods.

Usage

```
binarize.BASC(vect,
              method = c("A","B"),
              tau = 0.01,
              numberOfSamples = 999,
              sigma = seq(0.1, 20, by=.1),
              na.rm=FALSE)
```

Arguments

`method`	Chooses the BASC method to use (see details), i.e. either "A" or "B".
`vect`	A real-valued vector of data to binarize.
`tau`	This parameter adjusts the sensitivity and the specificity of the statistical testing procedure that rates the quality of the binarization. Defaults to 0.01.
`numberOfSamples`	The number of samples for the bootstrap test. Defaults to 999.
`sigma`	If `method="B"`, this specifies a vector of different sigma values for the convolutions with the Bessel function. Ignored for `method="A"`.
`na.rm`	If set to `TRUE`, `NA` values are removed from the input. Otherwise, binarization will fail in the presence of `NA` values.

Details

The two BASC methods can be subdivided into three steps:

Compute a series of step functions: An initial step function is obtained by rearranging the original time series measurements in increasing order. Then, step functions with fewer discontinuities are calculated. BASC A calculates these step functions in such a way that each minimizes the Euclidean distance to the initial step function. BASC B obtains step functions from smoothened versions of the input function in a scale-space manner.

Find strongest discontinuity in each step function: A strong discontinuity is a high jump size (derivative) in combination with a low approximation error.

Estimate location and variation of the strongest discontinuities: Based on these estimates, data values can be excluded from further analyses.

Value

Returns an object of class BASCResult.

References

M. Hopfensitz, C. Müssel, C. Wawra, M. Maucher, M. Kuehl, H. Neumann, and H. A. Kestler. Multiscale Binarization of Gene Expression Data for Reconstructing Boolean Networks. IEEE/ACM Transactions on Computational Biology and Bioinformatics 9(2):487-498, 2012.).

See Also

BinarizationResult, BASCResult

Examples

```
par(mfrow=c(2,1))
result <- binarize.BASC(iris[,"Petal.Length"], method="A", tau=0.15)
print(result)
plot(result)

result <- binarize.BASC(iris[,"Petal.Length"], method="B", tau=0.15)
print(result)
plot(result)
```

---

binarize.kMeans *k-means Binarization*

Description

Binarizes a vector of real-valued data using the k-means clustering algorithm. The data is first split into 2 clusters.The values belonging to the cluster with the smaller centroid are set to 0, and the values belonging to the greater centroid are set to 1.

Usage

```
binarize.kMeans(vect,
                nstart=1,
                iter.max=10,
                dip.test=TRUE,
                na.rm=FALSE)
```

Arguments

vect	A real-valued vector to be binarized (at least 3 measurements).
nstart	The number of restarts for k-means. See kmeans for details.
iter.max	The maximum number of iterations for k-means. See kmeans for details.
dip.test	If set to TRUE, Hartigan's dip test for unimodality is performed on vect, and its p-value is returned in the pvalue slot of the result. An insignificant test indicates that the data may not be binarizeable.
na.rm	If set to TRUE, NA values are removed from the input. Otherwise, binarization will fail in the presence of NA values.

Value

Returns an object of class BinarizationResult.

See Also

kmeans, BinarizationResult,

Examples

```
result <- binarize.kMeans(iris[,"Petal.Length"])

print(result)
plot(result, twoDimensional=TRUE)
```

---

binarizeMatrix *Utility function to binarize a matrix of measurements*

Description

Binarizes a matrix of measurements all at once, and returns the binarized vectors as well as the binarization thresholds and the p-values.

Usage

```
binarizeMatrix(mat,
               method = c("BASCA", "BASCB", "kMeans"),
               adjustment = "none",
               ...)
```

Arguments

mat	A n x m matrix comprising m raw measurements of n features.
method	The binarization algorithm to be used. method="BASCA" calls binarize.BASC with method="A". method="BASCB" calls binarize.BASC with method="B". method="kMeans" calls binarize.kMeans.
adjustment	Specifies an optional adjustment for multiple testing that is applied to the p-values (see p.adjust for possible values). By default, no adjustment is applied.
...	Further parameters that are passed to the respective binarization methods (binarize.BASC or method="kMeans").

Value

A n x (m+2) matrix of binarized measurements. Here, the first m columns correspond to the binarized measurements. The m+1-st column comprises the binarization thresholds for the features, and the m+2-nd column contains the p-values.

See Also

binarize.BASC, binarize.kMeans, p.adjust

Examples

```
bin <- binarizeMatrix(t(iris[,1:4]))
print(bin)
```

---

plot,BinarizationResult-method

Visualization of binarization results.

Description

Visualizes a binarization as a ray or a two-dimensional plot.

Usage

```
## S4 method for signature 'BinarizationResult,ANY'
plot(x,
     twoDimensional=FALSE,
     showLegend=TRUE,
     showThreshold=TRUE,
     ...)
## S4 method for signature 'numeric,BinarizationResult'
plot(x,
                                          y,
                                          showLegend=TRUE,
                                          showThreshold=TRUE,
                                          ...)
```

Arguments

x	If y is supplied, this is a vector of x coordinates for the binarization values iny, which are plotted on the y axis. If y is not supplied, this is object of class BinarizationResult containing the binarized values to visualize.
y	If x is a vector of x coordinates, this is object of class BinarizationResult containing the binarized values to visualize.
twoDimensional	Specifies whether the binarization is depicted as a ray or as a two-dimensional curve (see details).
showLegend	If set to true, a legend is included in the plot.
showThreshold	If set to true, the binarization threshold is depicted as a horizontal or vertical line (depending on twoDimensional).
...	Further graphical parameters to be passed to plot. The parameters col and pch can be supplied in different ways: If supplied as vectors of size 2, the first value corresponds to a 0 in the binarization, and the second value corresponds to a 1 in the binarization. col can also have length 3, in which case the third entry is the color of the threshold line. If col or pch have the size of the input vector, the corresponding colors and symbols are assigned to the data points.

Details

The function comprises two different plots: If twoDimensional = TRUE, the positions in the input vector are aligned with the x axis, and the y axis corresponds to the values. The binarization threshold is shown as a horizontal line, and the binarization is indicated by two different symbols.

If twoDimensional = FALSE, the binarized values are aligned with a one-dimensional ray, and the separating threshold is depicted as a vertical line.

See Also

plot, BinarizationResult

Examples

```
# plot a binarization in one and two dimensions
res <- binarize.BASC(iris[,"Petal.Length"], method="A")
plot(res)
plot(res, twoDimensional = TRUE)
plot(res, twoDimensional = TRUE,
     pch = c("x", "+"),
     col = c("red", "black", "royalblue"),
     lty = 4, lwd = 2)
```

plot,TrinarizationResult-method
Visualization of trinarization results.

Description

Visualizes a trinarization as a ray or a two-dimensional plot.

Usage

```
## S4 method for signature 'TrinarizationResult,ANY'
plot(x,
twoDimensional=FALSE,
showLegend=TRUE,
showThreshold=TRUE,
...)
## S4 method for signature 'numeric,TrinarizationResult'
plot(x,
y,
showLegend=TRUE,
showThreshold=TRUE,
...)
```

Arguments

x	If y is supplied, this is a vector of x coordinates for the trinarization values iny, which are plotted on the y axis. If y is not supplied, this is object of class TrinarizationResult containing the trinarized values to visualize.
y	If x is a vector of x coordinates, this is object of class TrinarizationResult containing the trinarized values to visualize.
twoDimensional	Specifies whether the trinarization is depicted as a ray or as a two-dimensional curve (see details).
showLegend	If set to true, a legend is included in the plot.
showThreshold	If set to true, the trinarization thresholds are depicted as a horizontal or vertical lines (depending on twoDimensional).
...	Further graphical parameters to be passed to plot. The parameters col and pch can be supplied in different ways: If supplied as vectors of size 3, the first value corresponds to a 0 in the trinarization, the second value corresponds to a 1, and the third value corresponds to a 2. col can also have length 4, in which case the fourth entry is the color of the threshold line. If col or pch have the size of the input vector, the corresponding colors and symbols are assigned to the data points.

Details

The function comprises two different plots: If `twoDimensional = TRUE`, the positions in the input vector are aligned with the x axis, and the y axis corresponds to the values. The trinarization thresholds are shown as a horizontal lines, and the trinarization is indicated by three different symbols.

If `twoDimensional = FALSE`, the trinarized values are aligned with a one-dimensional ray, and the separating thresholds are depicted as a vertical lines.

See Also

`plot`, `TrinarizationResult`

Examples

```
# plot a binarization in one and two dimensions
res <- TASC(iris[,"Petal.Length"])
plot(res)
plot(res, twoDimensional = TRUE)
plot(res, twoDimensional = TRUE,
 pch = c("x", "+"),
 col = c("red", "black", "royalblue", "green"),
 lty = 4, lwd = 2)
```

---

plotStepFunctions *Plot all step functions for BASC or TASC*

Description

A specialized visualization that plots all the optimal step functions computed by the BASC algorithms or TASC.

Usage

```
plotStepFunctions(x,
                  showLegend=TRUE,
                  connected=FALSE,
                  withOriginal=TRUE,
                  ...)
```

Arguments

`x`	A binarization (or trinarisation) result object of class `BASCResult` (`TASCResult`).
`showLegend`	If `TRUE`, a legend is included in the plot.
`connected`	If `TRUE`, the single steps of the step functions are connected by lines.
`withOriginal`	If set to `TRUE`, the original step function (i.e. the sorted input vector) is included in the plot.
`...`	Additional graphical parameters to be passed to `plot`.

See Also

BASCResult, binarize.BASC, TASCResult, TASC

Examples

```
result <- binarize.BASC(iris[,"Petal.Width"],
                        method="B")
plotStepFunctions(result)

result <- TASC(iris[,"Petal.Width"])
plotStepFunctions(result)
```

TASC *Trinarization Across Multiple Scales*

Description

Trinarizes real-valued data using the multiscale TASC method.

Usage

```
TASC(vect,
method = c("A","B"),
tau = 0.01,
numberOfSamples = 999,
sigma = seq(0.1, 20, by=.1),
na.rm=FALSE,
error = c("mean", "min"))
```

Arguments

method	Chooses the TASC method to use (see details), i.e. either "A" or "B".
vect	A real-valued vector of data to trinarize.
tau	This parameter adjusts the sensitivity and the specificity of the statistical testing procedure that rates the quality of the trinarization. Defaults to 0.01.
numberOfSamples	The number of samples for the bootstrap test. Defaults to 999.
sigma	If method="B", this specifies a vector of different sigma values for the convolutions with the Bessel function. Ignored for method="A".
na.rm	If set to TRUE, NA values are removed from the input. Otherwise, trinarization will fail in the presence of NA values.
error	Determines which error should be used for the data points between two thresholds, the "mean" error (default) to the thresholds or the "min" error.

Details

The two TASC methods can be subdivided into three steps:

Compute a series of step functions: An initial step function is obtained by rearranging the original time series measurements in increasing order. Then, step functions with fewer discontinuities are calculated. TASC A calculates these step functions in such a way that each minimizes the Euclidean distance to the initial step function. TASC B obtains step functions from smoothened versions of the input function in a scale-space manner.

Find strongest discontinuities in each step function: A strong discontinuity is a high jump size (derivative) in combination with a low approximation error. For TASC a pair of strongest discontinuities is determined.

Estimate location and variation of the strongest discontinuities: Based on these estimates, data values can be excluded from further analyses.

Value

Returns an object of class TASCResult.

See Also

TrinarizationResult, TASCResult

Examples

```
par(mfrow=c(2,1))
result <- TASC(iris[,"Petal.Width"], method="A", tau=0.15)
print(result)
plot(result)

result <- TASC(iris[,"Petal.Width"], method="B", tau=0.15)
print(result)
plot(result)
```

TASCResult-class *Class "TASCResult"*

Description

A specialized class storing the results of a call to TASC.

Objects of this class

Objects of this class shouldn't be created directly. They are created implicitly by a call to TASC.

Slots

`p.value`: The p-value of the statistical test for reliability of the trinarization.

`intermediateSteps`: A matrix specifying the optimal step functions from which the trinarization was calculated. The number of rows corresponds to the number of step functions, and the number of columns is determined by the length of the input vector minus 2 (that is, the length of the step function corresponding to the input vector). From the first to the last row, the number of steps increases. The non-zero entries of the matrix represent the locations of the steps. Step functions with fewer steps than the input step function have entries set to zero.

`intermediateHeights1`: A matrix giving the jump heights of the steps supplied in `intermediateSteps` for the first threshold.

`intermediateHeights2`: A matrix giving the jump heights of the steps supplied in `intermediateSteps` for the second threshold.

`intermediateStrongestSteps`: A matrix with one row for each step function (row) in `intermediateSteps`. The entries specify the location of the two strongest steps for each of the functions.

`originalMeasurements`: A numeric vector storing the input measurements.

`trinarizedMeasurements`: An integer vector of trinarized values (0, 1 or 2) corresponding to the original measurements.

`threshold1`: The threshold that separates 0 from 1.

`threshold2`: The threshold that separates 1 from 2.

`method`: A string describing the trinarization method that yielded the result.

Extends

Class "TrinarizationResult", directly.

Methods

plotStepFunctions `signature(x = "TASCResult")`: Plot the intermediate optimal step functions used to determine the thresholds.

print `signature(x = "TASCResult")`: Print a summary of the trinarization.

show `signature(object = "TASCResult")`: ...

See Also

TASC, TrinarizationResult

trinarizationExample *An artificial data set consisting of ten artificial feature vectors.*

Description

An artificial data set consisting of 100 artificial feature vectors that are used to illustrate the trinarization methods in the package vignette. Each row of the matrix `trinarizationExample` corresponds to one feature vector, of which 5 measurements are drawn from a normal distribution N(0,1). The remaining 10 measurements are drawn from two normal distributions N(m,1), with `m=10:1` and `m=seq(20,2,by=-2)` (5 meansurements per distribution).

Usage

```
data(trinarizationExample)
```

Format

The data is a matrix with 15 columns and 100 rows.

TrinarizationResult-class *Class "TrinarizationResult"*

Description

This is the base class for objects that store the results of a trinarization algorithm. It defines the slots and methods that the results of all algorithms share.

Objects of this class

Objects of this class shouldn't be created directly. They are created implicitly by a call to one of the trinarizeation algorithms.

Slots

`originalMeasurements`**:** A numeric vector storing the input measurements.

`trinarizedMeasurements`**:** An integer vector of trinarized values (0 or 1 or 2) corresponding to the original measurements.

`threshold1`**:** The threshold that separates 0 and 1.

`threshold2`**:** The threshold that separates 1 and 2.

`method`**:** A string describing the trinarization method that yielded the result.

`p.value`**:** The p-value obtained by a test for validity of the trinarization (e.g. TASC bootstrap test). If no test was performed, this is `NA`.

Methods

plot `signature(x = "TrinarizationResult")`: Plot the trinarization and the thresholds.

print `signature(x = "TrinarizationResult")`: Print a summary of the trinarization.

show `signature(object = "TrinarizationResult")`: ...

See Also

TASC, TASCResult,

`trinarizeMatrix` *Utility function to trinarize a matrix of measurements*

Description

Trinarizes a matrix of measurements all at once, and returns the trinarized vectors as well as the trinarization thresholds and the p-values.

Usage

```
trinarizeMatrix(mat,
                method = c("TASCA", "TASCB"),
                adjustment = "none",
                ...)
```

Arguments

`mat`	A n x m matrix comprising m raw measurements of n features.
`method`	The trinarization algorithm to be used. `method="TASCA"` calls TASC with `method="A"`. `method="TASCB"` calls TASC with `method="B"`.
`adjustment`	Specifies an optional adjustment for multiple testing that is applied to the p-values (see `p.adjust` for possible values). By default, no adjustment is applied.
`...`	Further parameters that are passed to the respective trinarization methods (TASC).

Value

A n x (m+3) matrix of trinarized measurements. Here, the first m columns correspond to the trinarized measurements. The m+1-st and the m+2-st column comprises the trinarization thresholds for the features, and the m+3-nd column contains the p-values.

See Also

TASC, `p.adjust`

Examples

```
tri <- trinarizeMatrix(t(iris[,1:4]))
print(tri)
```

Index