Package ‘Informeasure’

December 6, 2025

Type Package

Title R implementation of information measures

Version 1.21.0

Description This package consolidates a comprehensive set of information measurements, encompassing mutual information, conditional mutual information, interaction information, partial information decomposition, and part mutual information.

License Artistic-2.0

Depends R (>= 4.0)

Encoding UTF-8

LazyData true

RoxygenNote 7.3.1

biocViews GeneExpression, NetworkInference, Network, Software

Imports entropy

Suggests knitr, BiocStyle, rmarkdown, testthat (>= 3.0.0), SummarizedExperiment

VignetteBuilder knitr

URL https://github.com/chupan1218/Informeasure

BugReports https://github.com/chupan1218/Informeasure/issues

Config/testthat/edition 3

git_url https://git.bioconductor.org/packages/Informeasure

git_branch devel

git_last_commit efa47cf

git_last_commit_date 2025-10-29

Repository Bioconductor 3.23

Date/Publication 2025-12-05

Author Chu Pan [aut, cre]

Maintainer Chu Pan <chu.pan@hnu.edu.cn>

Contents

`CMI.measure` *A comprehensive function for estimating conditional mutual information*

Description

The CMI.measure function is used to calculate the expected mutual information between two random variables conditioned on the third one from the joint count table.

Usage

```
CMI.measure(
  XYZ,
  method = c("ML", "Jeffreys", "Laplace", "SG", "minimax", "shrink"),
  lambda.probs,
  unit = c("log", "log2", "log10"),
  verbose = TRUE
)
```

Arguments

`XYZ`	a joint count distribution table of three random variables.
`method`	six probability estimation algorithms are available, "ML" is the default.
`lambda.probs`	the shrinkage intensity, only called when the probability estimator is "shrink".
`unit`	the base of the logarithm. The default is natural logarithm, which is "log". For evaluating entropy in bits, it is suggested to set the unit to "log2".
`verbose`	a logic variable. if verbose is true, report the shrinkage intensity.

Details

Six probability estimation methods are available to evaluate the underlying bin probability from observed counts:
method = "ML": maximum likelihood estimator, also referred to empirical probability,
method = "Jeffreys": Dirichlet distribution estimator with prior a = 0.5,
method = "Laplace": Dirichlet distribution estimator with prior a = 1,
method = "SG": Dirichlet distribution estimator with prior a = 1/length(XY),
method = "minimax": Dirichlet distribution estimator with prior a = sqrt(sum(XY))/length(XY),
method = "shrink": shrinkage estimator.

Value

CMI.measure returns the conditional mutual information.

References

#' Hausser, J., & Strimmer, K. (2009). Entropy Inference and the James-Stein Estimator, with Application to Nonlinear Gene Association Networks. Journal of Machine Learning Research, 1469-1484.

Examples

```
# three numeric vectors corresponding to three continuous random variables
x <- c(0.0, 0.2, 0.2, 0.7, 0.9, 0.9, 0.9, 0.9, 1.0)
y <- c(1.0, 2.0,  12, 8.0, 1.0, 9.0, 0.0, 3.0, 9.0)
z <- c(3.0, 7.0, 2.0,  11,  10,  10,  14, 2.0,  11)

# corresponding joint count table estimated by "uniform width" algorithm
XYZ <- discretize3D(x, y, z, "uniform_width")

# corresponding conditional mutual information
CMI.measure(XYZ)
```

CMI.plugin *A plug-in calculator for evaluating conditional mutual information*

Description

CMI.plugin measures the expected mutual information between two random variables conditioned on the third one from the joint probability distribution table.

Usage

```
CMI.plugin(probs, unit = c("log", "log2", "log10"))
```

Arguments

`probs`	the joint probability distribution table of three random variables.
`unit`	the base of the logarithm. The default is natural logarithm, which is "log". For evaluating entropy in bits, it is suggested to set the unit to "log2".

Value

CMI.plugin returns the conditional mutual information.

References

Wyner, A. D. (1978). A definition of conditional mutual information for arbitrary ensembles. Information & Computation, 38(1), 51-59.

Examples

```
# three numeric vectors corresponding to three continuous random variables
x <- c(0.0, 0.2, 0.2, 0.7, 0.9, 0.9, 0.9, 0.9, 1.0)
y <- c(1.0, 2.0,  12, 8.0, 1.0, 9.0, 0.0, 3.0, 9.0)
z <- c(3.0, 7.0, 2.0,  11,  10,  10,  14, 2.0,  11)

# corresponding joint count table estimated by "uniform width" algorithm
count_xyz <- discretize3D(x, y, z, "uniform_width")

# the joint probability distribution table of the count data
library("entropy")
probs_xyz <- freqs.empirical(count_xyz)

# corresponding conditional mutual information
CMI.plugin(probs_xyz)
```

`discretize1D` *Discretize one-dimensional continuous data into bins*

Description

The function of discretize1D is used to assign the observations of a set of continuous random variables to bins, and returns a corresponding one-dimensional count table. Two of the most common discretization methods are available: "uniform width" and "uniform frequency".

Usage

```
discretize1D(x, algorithm = c("uniform_width", "uniform_frequency"))
```

Arguments

`x`	a numeric vector of the random variable x.
`algorithm`	two discretization algorithms are available, "uniform_width" is the default.

Details

Uniform width-based method ("uniform_width") divides the continuous data into N bins with equal width, while Uniform frequency-based method ("uniform_frequency") divides the continuous data into N bins with (approximate) equal count number. By default in both methods, the number of bins N is initialized into a round-off value according to the square root of the data size.

Value

discretize1D returns a one-dimensional count table.

Examples

```
# a numeric vector corresponding to a continuous random variable
x <- c(0.0, 0.2, 0.2, 0.7, 0.9, 0.9, 0.9, 0.9, 1.0)

# corresponding count table estimated by "uniform width" algorithm
discretize1D(x, "uniform_width")

# corresponding count table estimated by "uniform frequency" algorithm
discretize1D(x, "uniform_frequency")
```

discretize1d.uniform_frequency

Discretize a set of continuous data into 1-dimensional bins by uniform frequency

Description

discretize1d.uniform_frequency assigns the observations of a continuous random variables to bins according to the "uniform frequency" method, and returns a corresponding count table.

Usage

```
discretize1d.uniform_frequency(x)
```

Arguments

`x`	a numeric vector of a random variable.

Details

Uniform frequency-based method ("uniform_frequency") divides the continuous data into N bins with (approximate) equal count number. The number of bins N is initialized into a round-off value according to the square root of the data size.

Value

discretize1d.uniform_frequency returns a one-dimensional count table.

Examples

```
# a numeric vector corresponding to a continuous random variable
x <- c(0.0, 0.2, 0.2, 0.7, 0.9, 0.9, 0.9, 0.9, 1.0)

# corresponding count table estimated by "uniform frequency" algorithm
discretize1d.uniform_frequency(x)
```

discretize1d.uniform_width

Discretize a set of continuous data into 1-dimensional bins by "uniform width" method

Description

discretize1d.uniform_width assigns the observations of continuous random variables to bins according to the "uniform width" method, and returns a corresponding count table.

Usage

```
discretize1d.uniform_width(x)
```

Arguments

`x`	a numeric vector of a random variable.

Details

Uniform width-based method ("uniform_width") divides the continuous data into N bins with equal width. The number of bins N is initialized into a round-off value according to the square root of the data size.

Value

discretize1d.uniform_width returns a count table.

Examples

```
# a numeric vector corresponding to a continuous random variable
x <- c(0.0, 0.2, 0.2, 0.7, 0.9, 0.9, 0.9, 0.9, 1.0)

# corresponding count table estimated by "uniform width" algorithm
discretize1d.uniform_width(x)
```

`discretize2D` *Discretize 2-dimensional continuous data into bins*

Description

The function of discretize2D is used to assign the observations of two sets of continuous random variables to bins, and returns a corresponding two-dimensional count table. Two of the most common discretization methods are available: "uniform width" and "uniform frequency".

Usage

```
discretize2D(x, y, algorithm = c("uniform_width", "uniform_frequency"))
```

Arguments

`x`	a numeric vector of the random variable x.
`y`	a numeric vector of the random variable y.
`algorithm`	two discretization algorithms are available, "uniform_width" is the default.

Details

Uniform width-based method ("uniform_width") divides the continuous data into N bins with equal width, while Uniform frequency-based method ("uniform_frequency") divides the continuous data into N bins with (approximate) equal count number. By default in both methods, the number of bins N is initialized into a round-off value according to the square root of the data size.

Value

discretize2D returns a 2-dimensional count table.

Examples

```
# two numeric vectors that correspond to two continuous random variables
x <- c(0.0, 0.2, 0.2, 0.7, 0.9, 0.9, 0.9, 0.9, 1.0)
y <- c(1.0, 2.0,  12, 8.0, 1.0, 9.0, 0.0, 3.0, 9.0)

# corresponding count table estimated by "uniform width" algorithm
discretize2D(x,y, "uniform_width")

# corresponding count table estimated by "uniform frequency" algorithm
discretize2D(x,y, "uniform_frequency")
```

---

discretize2d.uniform_frequency

Discretize two sets of continuous data into 2-dimensional bins by uniform frequency

---

Description

discretize2d.uniform_frequency assigns the observations of two continuous random variables to bins according to the "uniform frequency" method, and returns a corresponding 2-dimensional count table.

Usage

```
discretize2d.uniform_frequency(x, y)
```

Arguments

x	a numeric vector of the first random variable.
y	a numeric vector of the second random variable.

Details

Uniform frequency-based method ("uniform_frequency") divides the continuous data into N bins with (approximate) equal count number. The number of bins N is initialized into a round-off value according to the square root of the data size.

Value

discretize2d.uniform_frequency returns a 2-dimensional count table.

Examples

```
# two numeric vectors corresponding to two continuous random variables
x <- c(0.0, 0.2, 0.2, 0.7, 0.9, 0.9, 0.9, 0.9, 1.0)
y <- c(1.0, 2.0,  12, 8.0, 1.0, 9.0, 0.0, 3.0, 9.0)

# corresponding joint count table estimated by "uniform frequency" algorithm
discretize2d.uniform_frequency(x,y)
```

---

discretize2d.uniform_width

Discretize two sets of continuous data into 2-dimensional bins by "uniform width" method

---

Description

discretize2d.uniform_width assigns the observations of two continuous random variables to bins according to the "uniform width" method, and returns a corresponding 2-dimensional count table.

Usage

```
discretize2d.uniform_width(x, y)
```

Arguments

x	a numeric vector of the first random variable.
y	a numeric vector of the second random variable.

Details

Uniform width-based method ("uniform_width") divides the continuous data into N bins with equal width. The number of bins N is initialized into a round-off value according to the square root of the data size.

Value

discretize2d.uniform_width returns a 2-dimensional count table.

Examples

```
# two numeric vectors corresponding to two continuous random variables
x <- c(0.0, 0.2, 0.2, 0.7, 0.9, 0.9, 0.9, 0.9, 1.0)
y <- c(1.0, 2.0,  12, 8.0, 1.0, 9.0, 0.0, 3.0, 9.0)

# corresponding joint count table estimated by "uniform width" algorithm
discretize2d.uniform_width(x,y)
```

`discretize3D` *Discretize 3-dimensional continuous data into bins*

Description

The function of discretize3D is used to assign the observations of three sets of continuous random variables to bins, and returns a corresponding three-dimensional count table. Two of the most common discretization methods are available: "uniform width" and "uniform frequency".

Usage

```
discretize3D(x, y, z, algorithm = c("uniform_width", "uniform_frequency"))
```

Arguments

`x`	a numeric vector of the random variable x.
`y`	a numeric vector of the random variable y.
`z`	a numeric vector of the random variable z.
`algorithm`	two discretization algorithms are available, "uniform_width" is the default.

Details

Uniform width-based method ("uniform_width") divides the continuous data into N bins with equal width, while Uniform frequency-based method ("uniform_frequency") divides the continuous data into N bins with (approximate) equal count number. By default in both methods, the number of bins N is initialized into a round-off value according to the square root of the data size.

Value

discretize3D returns a 3-dimensional count table.

Examples

```
# three vectors that correspond to three continuous random variables
x <- c(0.0, 0.2, 0.2, 0.7, 0.9, 0.9, 0.9, 0.9, 1.0)
y <- c(1.0, 2.0,  12, 8.0, 1.0, 9.0, 0.0, 3.0, 9.0)
z <- c(3.0, 7.0, 2.0,  11,  10,  10,  14, 2.0,  11)

# corresponding count table estimated by "uniform width" algorithm
discretize3D(x,y,z, "uniform_width")

# corresponding count table estimated by "uniform frequency" algorithm
discretize3D(x,y,z, "uniform_frequency")
```

`discretize3d.uniform_frequency`

Discretize three sets of continuous data into 3-dimensional bins by uniform frequency

Description

discretize3d.uniform_frequency assigns the observations of three continuous random variables to bins according to the "uniform frequency" method, and returns a corresponding 3-dimensional count table.

Usage

```
discretize3d.uniform_frequency(x, y, z)
```

Arguments

`x`	a numeric vector of the first random variable.
`y`	a numeric vector of the second random variable.
`z`	a numeric vector of the third random variable.

Details

Uniform frequency-based method ("uniform_frequency") divides the continuous data into N bins with (approximate) equal count number. The number of bins N is initialized into a round-off value according to the square root of the data size.

Value

discretize3d.uniform_frequency returns a 3-dimensional count table.

Examples

```
# three numeric vectors corresponding to three continuous random variables
x <- c(0.0, 0.2, 0.2, 0.7, 0.9, 0.9, 0.9, 0.9, 1.0)
y <- c(1.0, 2.0,  12, 8.0, 1.0, 9.0, 0.0, 3.0, 9.0)
z <- c(3.0, 7.0, 2.0,  11,  10,  10,  14, 2.0,  11)

# corresponding joint count table estimated by "uniform frequency" algorithm
discretize3d.uniform_frequency(x,y,z)
```

`discretize3d.uniform_width`

Discretize three sets of continuous data into 3-dimensional bins by "uniform width" method

---

Description

discretize3d.uniform_width assigns the observations of three continuous random variables to bins according to the "uniform width" method, and returns a corresponding 3-dimensional count table.

Usage

```
discretize3d.uniform_width(x, y, z)
```

Arguments

x	a numeric vector of the first random variable.
y	a numeric vector of the second random variable.
z	a numeric vector of the third random variable.

Details

The uniform width-based method ("uniform_width") that divides the continuous data into N bins with equal width. The number of bins is initialized into a round-off value according to the square root of the data size.

Value

discretize3d.uniform_width returns a 3-dimensional count table.

Examples

```
# three numeric vectors corresponding to three continuous random variables
x <- c(0.0, 0.2, 0.2, 0.7, 0.9, 0.9, 0.9, 0.9, 1.0)
y <- c(1.0, 2.0,  12, 8.0, 1.0, 9.0, 0.0, 3.0, 9.0)
z <- c(3.0, 7.0, 2.0,  11,  10,  10,  14, 2.0,  11)

# corresponding joint count table estimated by "uniform width" algorithm
discretize3d.uniform_width(x,y,z)
```

`II.measure` *A comprehensive function for evaluating interaction information*

Description

The II.measure function is used to calculate the amount information contained in a set of variables from the joint count table. The number of variables here is limited to three.

Usage

```
II.measure(
  XYZ,
  method = c("ML", "Jeffreys", "Laplace", "SG", "minimax", "shrink"),
  lambda.probs,
  unit = c("log", "log2", "log10"),
  verbose = TRUE
)
```

Arguments

`XYZ`	a joint count distribution table of three random variables.
`method`	six probability estimation algorithms are available, "ML" is the default.
`lambda.probs`	the shrinkage intensity, only called when the probability estimator is "shrink".
`unit`	the base of the logarithm. The default is natural logarithm, which is "log". For evaluating entropy in bits, it is suggested to set the unit to "log2".
`verbose`	a logic variable. if verbose is true, report the shrinkage intensity.

Details

Six probability estimation methods are available to evaluate the underlying bin probability from observed counts:
method = "ML": maximum likelihood estimator, also referred to empirical probability,
method = "Jeffreys": Dirichlet distribution estimator with prior a = 0.5,
method = "Laplace": Dirichlet distribution estimator with prior a = 1,
method = "SG": Dirichlet distribution estimator with prior a = 1/length(XY),
method = "minimax": Dirichlet distribution estimator with prior a = sqrt(sum(XY))/length(XY),
method = "shrink": shrinkage estimator.

Value

II.measure returns the interaction information.

References

Hausser, J., & Strimmer, K. (2009). Entropy Inference and the James-Stein Estimator, with Application to Nonlinear Gene Association Networks. Journal of Machine Learning Research, 1469-1484.

Mcgill, W. J. (1954). Multivariate information transmission. Psychometrika, 19(2), 97-116.

Examples

```
# three numeric vectors corresponding to three continuous random variables
x <- c(0.0, 0.2, 0.2, 0.7, 0.9, 0.9, 0.9, 0.9, 1.0)
y <- c(1.0, 2.0,  12, 8.0, 1.0, 9.0, 0.0, 3.0, 9.0)
z <- c(3.0, 7.0, 2.0,  11,  10,  10,  14, 2.0,  11)

# corresponding joint count table estimated by "uniform width" algorithm
XYZ <- discretize3D(x, y, z, "uniform_width")

# corresponding interaction information
II.measure(XYZ)
```

`II.plugin` *A plug-in calculator for evaluating the interaction information*

Description

II.plugin measures the amount information contained in a set of variables from the joint probability distribution table. The number of variables here is limited to three.

Usage

```
II.plugin(probs, unit = c("log", "log2", "log10"))
```

Arguments

`probs`	the joint probability distribution table of three random variables.
`unit`	the base of the logarithm. The default is natural logarithm, which is "log". For evaluating entropy in bits, it is suggested to set the unit to "log2".

Value

II.plugin returns the interaction information.

References

Mcgill, W. J. (1954). Multivariate information transmission. Psychometrika, 19(2), 97-116.

Examples

```
# three numeric vectors corresponding to three continuous random variables
x <- c(0.0, 0.2, 0.2, 0.7, 0.9, 0.9, 0.9, 0.9, 1.0)
y <- c(1.0, 2.0,  12, 8.0, 1.0, 9.0, 0.0, 3.0, 9.0)
z <- c(3.0, 7.0, 2.0,  11,  10,  10,  14, 2.0,  11)

# corresponding joint count table estimated by "uniform width" algorithm
count_xyz <- discretize3D(x, y, z, "uniform_width")
```

```
# the joint probability distribution table of the count data
library("entropy")
probs_xyz <- freqs.empirical(count_xyz)

# corresponding interaction information
II.plugin(probs_xyz)
```

MI.measure *A comprehensive function for evaluating mutual information*

Description

The MI.measure function is used to calculate the mutual information between two random variables from the joint count table.

Usage

```
MI.measure(
  XY,
  method = c("ML", "Jeffreys", "Laplace", "SG", "minimax", "shrink"),
  lambda.probs,
  unit = c("log", "log2", "log10"),
  verbose = TRUE
)
```

Arguments

XY	a joint count distribution table of two random variables.
method	six probability estimation algorithms are available, "ML" is the default.
lambda.probs	the shrinkage intensity, only called when the probability estimator is "shrink".
unit	the base of the logarithm. The default is natural logarithm, which is "log". For evaluating entropy in bits, it is suggested to set the unit to "log2".
verbose	a logic variable. if verbose is true, report the shrinkage intensity.

Details

Six probability estimation methods are available to evaluate the underlying bin probability from observed counts:
method = "ML": maximum likelihood estimator, also referred to empirical probability,
method = "Jeffreys": Dirichlet distribution estimator with prior a = 0.5,
method = "Laplace": Dirichlet distribution estimator with prior a = 1,
method = "SG": Dirichlet distribution estimator with prior a = 1/length(XY),
method = "minimax": Dirichlet distribution estimator with prior a = sqrt(sum(XY))/length(XY),
method = "shrink": shrinkage estimator.

Value

MI.measure returns the mutual information.

References

Hausser, J., & Strimmer, K. (2009). Entropy Inference and the James-Stein Estimator, with Application to Nonlinear Gene Association Networks. Journal of Machine Learning Research, 1469-1484.

Wyner, A. D. (1978). A definition of conditional mutual information for arbitrary ensembles. Information & Computation, 38(1), 51-59.

Examples

```
# two numeric vectors corresponding to two continuous random variables
x <- c(0.0, 0.2, 0.2, 0.7, 0.9, 0.9, 0.9, 0.9, 1.0)
y <- c(1.0, 2.0,  12, 8.0, 1.0, 9.0, 0.0, 3.0, 9.0)

# corresponding joint count table estimated by "uniform width" algorithm
XY <- discretize2D(x, y, "uniform_width")

# corresponding mutual information
MI.measure(XY)
```

MI.plugin *A plug-in calculator for evaluating mutual information*

Description

MI.plugin measures the mutual information between two random variables from the joint probability distribution table.

Usage

```
MI.plugin(probs, unit = c("log", "log2", "log10"))
```

Arguments

probs	the joint probability distribution table of two random variables.
unit	the base of the logarithm. The default is natural logarithm, which is "log". For evaluating entropy in bits, it is suggested to set the unit to "log2".

Value

MI.plugin returns the mutual information.

Examples

```
# two numeric vectors corresponding to two continuous random variables
x <- c(0.0, 0.2, 0.2, 0.7, 0.9, 0.9, 0.9, 0.9, 1.0)
y <- c(1.0, 2.0,  12, 8.0, 1.0, 9.0, 0.0, 3.0, 9.0)

# corresponding joint count table estimated by "uniform width" algorithm
count_xy <- discretize2D(x, y, "uniform_width")
```

```
# the joint probability distribution table of the count data
library("entropy")
probs_xy <- freqs.empirical(count_xy)

# corresponding mutual information
MI.plugin(probs_xy)
```

PID.measure *A comprehensive function for evaluating the partial information decomposition*

Description

The PID.measure function is used to decompose two source information acting on the common target into four parts: joint information (synergy), unique information from source x, unique information from source y and shared information (redundancy). The input of the PID.measure is the joint count table.

Usage

```
PID.measure(
  XYZ,
  method = c("ML", "Jeffreys", "Laplace", "SG", "minimax", "shrink"),
  lambda.probs,
  unit = c("log", "log2", "log10"),
  verbose = TRUE
)
```

Arguments

`XYZ`	a joint count distribution table of three random variables
`method`	six probability estimation algorithms are available, "ML" is the default.
`lambda.probs`	the shrinkage intensity, only called when the probability estimator is "shrink".
`unit`	the base of the logarithm. The default is natural logarithm, which is "log". For evaluating entropy in bits, it is suggested to set the unit to "log2".
`verbose`	a logic variable. if verbose is true, report the shrinkage intensity.

Details

Six probability estimation methods are available to evaluate the underlying bin probability from observed counts:
method = "ML": maximum likelihood estimator, also referred to empirical probability,
method = "Jeffreys": Dirichlet distribution estimator with prior a = 0.5,
method = "Laplace": Dirichlet distribution estimator with prior a = 1,
method = "SG": Dirichlet distribution estimator with prior a = 1/length(XY),
method = "minimax": Dirichlet distribution estimator with prior a = sqrt(sum(XY))/length(XY),
method = "shrink": shrinkage estimator.

Value

PID.measure returns a list that includes synergistic information, unique information from x, unique information from y, redundant information and the sum of the four parts of information.

References

Hausser, J., & Strimmer, K. (2009). Entropy Inference and the James-Stein Estimator, with Application to Nonlinear Gene Association Networks. Journal of Machine Learning Research, 1469-1484.

Williams, P. L., & Beer, R. D. (2010). Nonnegative Decomposition of Multivariate Information. arXiv: Information Theory.

Chan, T. E., Stumpf, M. P., & Babtie, A. C. (2017). Gene Regulatory Network Inference from Single-Cell Data Using Multivariate Information Measures. Cell Systems, 5(3).

Examples

```
# three numeric vectors corresponding to three continuous random variables
x <- c(0.0, 0.2, 0.2, 0.7, 0.9, 0.9, 0.9, 0.9, 1.0)
y <- c(1.0, 2.0,  12, 8.0, 1.0, 9.0, 0.0, 3.0, 9.0)
z <- c(3.0, 7.0, 2.0,  11,  10,  10,  14, 2.0,  11)

# corresponding joint count table estimated by "uniform width" algorithm
XYZ <- discretize3D(x, y, z, "uniform_width")

# corresponding partial information decomposition
PID.measure(XYZ)

# corresponding count table estimated by "uniform frequency" algorithm
XYZ <- discretize3D(x, y, z, "uniform_frequency")

# corresponding partial information decomposition
PID.measure(XYZ)
```

PID.plugin — *A plug-in calculator for evaluating partial information decomposition*

Description

PID.plugin decomposes two source information acting on the common target into four parts: joint information (synergy), unique information from source x, unique information from source y and shared information (redundancy). The input of PMI.plug is the joint probability distribution table.

Usage

```
PID.plugin(probs, unit = c("log", "log2", "log10"))
```

Arguments

probs	the joint probability distribution of three random variables.
unit	the base of the logarithm. The default is natural logarithm, which is "log". For evaluating entropy in bits, it is suggested to set the unit to "log2".

Value

PID.plugin returns a list that includes synergistic information, unique information from source x, unique information from source y, redundant information and the sum of the four parts of information.

References

Williams, P. L., & Beer, R. D. (2010). Nonnegative Decomposition of Multivariate Information. arXiv: Information Theory.

Chan, T. E., Stumpf, M. P., & Babtie, A. C. (2017). Gene Regulatory Network Inference from Single-Cell Data Using Multivariate Information Measures. Cell systems, 5(3).

Examples

```
# three numeric vectors corresponding to three continuous random variables
x <- c(0.0, 0.2, 0.2, 0.7, 0.9, 0.9, 0.9, 0.9, 1.0)
y <- c(1.0, 2.0,  12, 8.0, 1.0, 9.0, 0.0, 3.0, 9.0)
z <- c(3.0, 7.0, 2.0,  11,  10,  10,  14, 2.0,  11)

# corresponding joint count table estimated by "uniform width" algorithm
count_xyz <- discretize3D(x, y, z, "uniform_width")

# the joint probability distribution table of the count data
library("entropy")
probs_xyz <- freqs.empirical(count_xyz)

# corresponding partial information decomposition
PID.plugin(probs_xyz)
```

PMI.measure *A comprehensive function for evaluating part mutual information*

Description

The PMI.measure function is used to calculate the non-linearly direct dependencies between two variables conditioned on the third one form the joint count table.

Usage

```
PMI.measure(
  XYZ,
  method = c("ML", "Jeffreys", "Laplace", "SG", "minimax", "shrink"),
  lambda.probs,
  unit = c("log", "log2", "log10"),
  verbose = TRUE
)
```

Arguments

`XYZ`	a joint count distribution table of three random variables.
`method`	six probability estimation algorithms are available, "ML" is the default.
`lambda.probs`	the shrinkage intensity, only called when the probability estimator is "shrink".
`unit`	the base of the logarithm. The default is natural logarithm, which is "log". For evaluating entropy in bits, it is suggested to set the unit to "log2".
`verbose`	a logic variable. if verbose is true, report the shrinkage intensity.

Details

Six probability estimation methods are available to evaluate the underlying bin probability from observed counts:
method = "ML": maximum likelihood estimator, also referred to empirical probability,
method = "Jeffreys": Dirichlet distribution estimator with prior a = 0.5,
method = "Laplace": Dirichlet distribution estimator with prior a = 1,
method = "SG": Dirichlet distribution estimator with prior a = 1/length(XY),
method = "minimax": Dirichlet distribution estimator with prior a = sqrt(sum(XY))/length(XY),
method = "shrink": shrinkage estimator.

Value

PMI.measure returns the part mutual information.

References

Hausser, J., & Strimmer, K. (2009). Entropy Inference and the James-Stein Estimator, with Application to Nonlinear Gene Association Networks. Journal of Machine Learning Research, 1469-1484.

Zhao, J., Zhou, Y., Zhang, X., & Chen, L. (2016). Part mutual information for quantifying direct associations in networks. Proceedings of the National Academy of Sciences of the United States of America, 113(18), 5130-5135.

Examples

```
# three numeric vectors corresponding to three continuous random variables
x <- c(0.0, 0.2, 0.2, 0.7, 0.9, 0.9, 0.9, 0.9, 1.0)
y <- c(1.0, 2.0,  12, 8.0, 1.0, 9.0, 0.0, 3.0, 9.0)
z <- c(3.0, 7.0, 2.0,  11,  10,  10,  14, 2.0,  11)
```

```
# corresponding joint count table estimated by "uniform width" algorithm
XYZ <- discretize3D(x, y, z, "uniform_width")

# corresponding part mutual information
PMI.measure(XYZ)
```

`PMI.plugin` *A plug-in calculator for evaluating the part mutual information*

Description

PMI.plug measures the non-linearly direct dependencies between two variables conditioned on the third one form the joint probability distribution table.

Usage

```
PMI.plugin(probs, unit = c("log", "log2", "log10"))
```

Arguments

probs	the joint probability distribution table of three random variables.
unit	the base of the logarithm. The default is natural logarithm, which is "log". For evaluating entropy in bits, it is suggested to set the unit to "log2".

Value

PMI.plugin returns the part mutual information.

References

Zhao, J., Zhou, Y., Zhang, X., & Chen, L. (2016). Part mutual information for quantifying direct associations in networks. Proceedings of the National Academy of Sciences of the United States of America, 113(18), 5130-5135.

Examples

```
# three numeric vectors corresponding to three continuous random variables
x <- c(0.0, 0.2, 0.2, 0.7, 0.9, 0.9, 0.9, 0.9, 1.0)
y <- c(1.0, 2.0,  12, 8.0, 1.0, 9.0, 0.0, 3.0, 9.0)
z <- c(3.0, 7.0, 2.0,  11,  10,  10,  14, 2.0,  11)

# corresponding joint count table estimated by "uniform width" algorithm
count_xyz <- discretize3D(x, y, z, "uniform_width")

# the joint probability distribution table of the count data
library("entropy")
probs_xyz <- freqs.empirical(count_xyz)

# corresponding part mutual information
PMI.plugin(probs_xyz)
```

Index