Package ‘SCCI’

June 4, 2019

Type  Package
Title  Stochastic Complexity-Based Conditional Independence Test for Discrete Data
Version 1.2
Date  2019-06-04
Author Alexander Marx [aut,cre] Jilles Vreeken [aut]
Maintainer Alexander Marx <amarx@mpi-inf.mpg.de>
Description
An efficient implementation of SCCI using 'Rcpp'. SCCI is short for the Stochastic Complexity-based Conditional Independence criterium (Marx and Vreeken, 2019). SCCI is an asymptotically unbiased and L2 consistent estimator of (conditional) mutual information for discrete data.
License GPL (>= 2)
Imports  Rcpp (>= 0.12.13)
LinkingTo Rcpp
Suggests  pcalg, Rgraphviz
Encoding UTF-8
NeedsCompilation yes
Repository CRAN
Date/Publication 2019-06-04 15:00:06 UTC

R topics documented:

conditionalShannonEntropy ........................................... 2
conditionalStochasticComplexity ..................................... 2
pSCCI ................................................................. 3
regret .......................................................................... 4
SCCI ........................................................................... 5
shannonEntropy ............................................................. 6
stochasticComplexity ...................................................... 6

Index 8
conditionalShannonEntropy

*Conditional Shannon Entropy*

**Description**
Calculates the Shannon entropy of a discrete random variable \(X\) conditioned on a discrete (possibly multivariate) random variable \(Y\).

**Usage**
```r
conditionalShannonEntropy(x, y)
```

**Arguments**
- `x`: A discrete vector.
- `y`: A data frame.

**Examples**
```r
set.seed(1)
x = round(runif(1000, min=0, max=5))
y = data.frame(round(runif(1000, min=0, max=5)), round(runif(1000, min=0, max=5)))
conditionalShannonEntropy(x=x, y=y) # # 2.41972
```

conditionalStochasticComplexity

*Conditional Stochastic Complexity for Multinomials*

**Description**
Calculates the Stochastic Complexity of a discrete random variable \(X\) conditioned on a discrete (possibly multivariate) random variable \(Y\). Variants for both factorized NML (fNML, Silander et al. 2008) and quotient NML (qNML, Silander et al. 2018) are included.

**Usage**
```r
conditionalStochasticComplexity(x, y, score="fNML")
```

**Arguments**
- `x`: A discrete vector.
- `y`: A discrete vector or a data frame containing multiple discrete vectors to condition \(X\) on.
- `score`: Default: fNML, optionally qNML can be passed.
References

Tomi Silander, Janne Leppä-aho, Elias Jääsaari, Teemu Roos; Quotient normalized maximum likelihood criterion for learning bayesian network structures, Proceedings of the 21nd International Conference on Artificial Intelligence and Statistics (AISTATS), PMLR, 2018


Examples

```r
set.seed(1)
x = round((runif(1000, min=0, max=5)))
y = data.frame(round((runif(1000, min=0, max=5))), round((runif(1000, min=0, max=5))))
conditionalStochasticComplexity(x=x, y=Y, score="fNML") # # 2779.477
```

---

pSCCI

Stochastic Complexity-based Conditional Independence Criterium (p-value)

Description

This is an adapted version of SCCI for which the output can be interpreted as a p-value. For this, we adapted SCCI such that if SCCI = 0 (X is independent of Y given Z) it gives a p-value greater than 0.01 and for SCCI > 0 (X is not independent of Y given Z) gives a p-value smaller or equal to 0.01. Note that we just transformed the output of SCCI and do not obtain a real p-value. In essence, we define the artificial p-value as follows. Let v the output of SCCI divided by the number of samples n.  
\[ p = 2^{-\left(6.643855 - v \right)} \]

which is equal to 0.0100001 if v = 0. Further, \( p \leq 0.01 \) for SCCI \( \geq 0.000001 \). We restrict the p-values to be between 0 and 1.

Unlike SCCI, pSCCI is currently only instantiated with fNML.

pSCCI can be used directly in the PC algorithm developed by Spirtes et al. (2000), which was implemented in the 'pcalg' R-package by Kalisch et al. (2012), as shown in the example.

Usage

```r
pSCCI(x, y, S, suffStat)
```

Arguments

- `x` Position of x variabe (integer).
- `y` Position of y variabe (integer).
- `S` Vector of the position of zero or more conditioning variables (integer).
- `suffStat` This format was adapted such that it can be used in the PC algorithm and other algorithms from the 'pcalg' package. SCCI only need the filed "dm" that contains the data matrix.
References


Alexander Marx and Jilles Vreeken; Testing Conditional Independence on Discrete Data using Stochastic Complexity. Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics (AISTATS), PMLR, 2019

Peter Spirtes, Clark N. Glymour, Richard Scheines, David Heckerman, Christopher Meek, Gregory Cooper and Thomas Richardson; Causation, Prediction, and Search, MIT press, 2000

Examples

```r
set.seed(1)
x = round(rnorm(1000, min=0, max=5))
y = round(rnorm(1000, min=0, max=5))
Z = data.frame(round(rnorm(1000, min=0, max=5)), round(rnorm(1000, min=0, max=5)))
## create data matrix
data_matrix = as.matrix(data.frame(x,y,2))
suffstat = list(dm=data_matrix)
pSCCI(x,y=2,z=c(3,4),suffStat=suffStat) # 0.01000001

### Using SCI within the PC algorithm
if(require(pcalg)){
  ## Load data
data(gmd)
  V <- colnames(gmd$x)
  ## define sufficient statistics
  suffStat <- list(dm = gmd$x, nlev = c(3,2,3,4,2), adaptDF = FALSE)
  ## estimate CPDAG
  pc.D <- pc(suffStat,
             ## independence test: SCCI using fNML
             indepTest = pSCCI, alpha = 0.01, labels = V, verbose = TRUE)
}
if (require(pcalg) & require(Rgraphviz)) {
  ## show estimated CPDAG
  par(mfrow = c(1,2))
  plot(pc.D, main = "Estimated CPDAG")
  plot(gmd$g, main = "True DAG")
}
```

regret

Multinomial Regret Term

Description

Calculates the multinomial regret term for for a discrete random variable with domain size \( k \) and sample size \( n \) (see Silander et al. 2018). Note that we use the logarithm to basis 2 to calculate the result. To compare the results to Silander et al. (2018), we need to multiply the result with \( \log(2) \) to compare the results.
Usage

```r
regret(n, k)
```

Arguments

- **n**: Integer (sample size)
- **k**: Integer (domain size)

References

Tomi Silander, Janne Leppä-aho, Elias Jääsaari, Teemu Roos; Quotient normalized maximum likelihood criterion for learning bayesian network structures, Proceedings of the 21nd International Conference on Artificial Intelligence and Statistics (AISTATS), PMLR, 2018

Examples

```r
regret(50, 10)  ## 19.1
regret(50, 10) * log(2)  ## 13.24 (see Silander et al. 2018)
```

---

**SCCI**

*Stochastic Complexity-based Conditional Independence Criterium*

**Description**

Calculates whether two random variables \(X\) and \(Y\) are independent given a set of variables \(Z\) using SCCI. A score of 0 denotes that independence holds and values greater than 0 mean that \(X\) is not independent of \(Y\) given \(Z\). For details on SCCI, we refer to Marx and Vreeken (AISTATS, 2019). If you use SCCI in your work, please cite Marx and Vreeken (AISTATS, 19).

The output of SCCI(\(\cdot\)) is the difference in number of bits between conditioning \(X\) only on \(Z\) and conditioning on \(Z\) and \(Y\). For the variant of SCCI that gives outputs that can be interpreted as p-values, please refer to pSCCI.

**Usage**

```r
SCCI(x, y, z, score="fNML", sym=FALSE)
```

**Arguments**

- **x**: A discrete vector.
- **y**: A discrete vector.
- **z**: A data frame consisting of zero or more columns of discrete vectors.
- **score**: Default: fNML, optionally qNML can be passed.
- **sym**: sym can be true or false
References
Alexander Marx and Jilles Vreeken; Testing Conditional Independence on Discrete Data using Stochastic Complexity, Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics (AISTATS), PMLR, 2019

Examples
set.seed(1)
x = round(runif(1000, min=0, max=5))
y = round(runif(1000, min=0, max=5))
Z = data.frame(round(runif(1000, min=0, max=5)), round(runif(1000, min=0, max=5)))
SCCI(x=x,y=y,Z,Z,score="FNML",sym=FALSE) ## 0

shannonEntropy  Shannon Entropy

Description
Calculates the Shannon entropy over data of a discrete random variable \(X\).

Usage
shannonEntropy(x)

Arguments
x  A discrete vector.

Examples
set.seed(1)
x = round(runif(1000, min=0, max=5))
shannonEntropy(x=x) ## 2.522265

stochasticComplexity  Stochastic Complexity for Multinomials

Description
Efficient implementation of the exact computation of stochastic complexity for multinomials (Sillander et al. 2008) for data over a discrete random variable \(X\).

Usage
stochasticComplexity(x)
**stochasticComplexity**

**Arguments**

- **x**  
  A discrete vector.

**References**


**Examples**

```r
set.seed(1)
x = round(runif(1000, min=0, max=5))
stochasticComplexity(x=x) # # 2544.698
```
Index

conditionalShannonEntropy, 2
conditionalStochasticComplexity, 2

pSCCI, 3

regret, 4

SCCI, 5
shannonEntropy, 6
stochasticComplexity, 6