Package 'maRkov'

May 10, 2016

Type Package

Title Goodness-of-fit Tests for Binary Markov Chains

Version 0.1

Date 2015-08-04

Author c(person(given = ``Conrad'', family = ``Cartmell'', role = c(``aut'', ``cre''), email = ``cwcartmell@gmail.com''), person(given = ``Debashis'', family = ``Mondal'', role = ``ctb'', email = ``debashis.stat@gmail.com''))

Maintainer Conrad Cartmell <cwcartmell@gmail.com>

Description Preforming goodness-of-fit tests on single binary Markov chains, as well as sets of multiple binary Markov chains. Provides LRT, chi square, and run test statistics, and generates data that can be used to preform user-created tests as well.

License GPL

Imports Rcpp

LinkingTo Rcpp

LazyData TRUE

RoxygenNote 5.0.1

Depends R (>= 2.10)

Suggests knitr, rmarkdown

VignetteBuilder knitr

R topics documented:

`alter.to.true.binary` *Take a one dimensional vector with two unique values and change those values to integers 0 and 1.*

Description

`alter.to.true.binary` takes a one dimensional vector that contains two unique values and switches all of the values with either integer values 0 or 1, so that the resulting vector has a one-to-one correspondence with the original.

Usage

```
alter.to.true.binary(bin.chain)
```

Arguments

bin.chain	A one dimensional vector with two unique elements.

Details

For this function to work properly, its argument should be checked first using check.false.binary, and only used in alter.to.true.binary if check.false.binary returns TRUE.

Examples

```
alter.to.true.binary(c("A","B","B","B","A","B","A","A"))
alter.to.true.binary(c(TRUE,TRUE,TRUE,FALSE,FALSE,TRUE,FALSE,FALSE,TRUE,TRUE,
FALSE))
```

---

alter.to.true.binary.multiple

Take a two dimensional matrix with two unique values and change those values to integers 0 and 1.

Description

alter.to.true.binary.multiple takes a two dimensional matrix that contains two unique values and switches all of the values with either integer values 0 or 1, so that the resulting matrix has a one-to-one correspondence with the original.

Usage

```
alter.to.true.binary.multiple(bin.chains)
```

Arguments

bin.chains	A two dimensional vector with two unique elements.

Details

For this function to work properly, its argument should be checked first using check.false.binary.multiple and only used in alter.to.true.binary if check.false.binary.multiple returns TRUE.

Examples

```
alter.to.true.binary.multiple(matrix(data = c("A","B","A","B","A","B","B",
"A","A"), ncol = 3))
alter.to.true.binary.multiple(matrix(data = c(TRUE,TRUE,TRUE,FALSE,FALSE,
TRUE,FALSE,FALSE,TRUE,TRUE,FALSE), ncol = 3))
```

check.false.binary *Check if a vector has two unique elements.*

Description

check.false.binary returns TRUE if there are two unique elements in its argument, and returns FALSE if there are not two unique elements in its argument.

Usage

```
check.false.binary(bin.chain)
```

Arguments

bin.chain	A one dimension vector.

Details

This function is not designed to be used outside of this package. It is sufficiently simple as to be practically redundant to the user.

Examples

```
check.false.binary(c(1,0,0,1,0,0,0,1))
check.false.binary(c("A","B","B","B","A","B","A","A"))
check.false.binary(c(TRUE,TRUE,TRUE,FALSE,FALSE,TRUE,FALSE,FALSE,TRUE,TRUE,
FALSE))
```

check.false.binary.multiple *Check if a two dimensional matrix has two unique elements.*

Description

check.false.binary.multiple returns TRUE is there are two unique elements in its argument, and returns FALSE if there are not two unique elements in its argument.

Usage

```
check.false.binary.multiple(bin.chains)
```

Arguments

bin.chains	A two dimensional matrix

Details

This function checks every row and column element to see if they all share the same two values.

Examples

```
check.false.binary.multiple(matrix(data = c(1,0,1,0,1,0,0,1,1), ncol = 3))
check.false.binary.multiple(matrix(data = c("A","B","A","B","A","B","B","A",
"A"), ncol = 3))
check.false.binary.multiple(matrix(data = c(TRUE,TRUE,TRUE,FALSE,FALSE,TRUE,
FALSE,FALSE,TRUE,TRUE,FALSE), ncol = 3))
```

---

check.true.binary *Check if a one dimensional vector has only integer elements 0 and 1.*

Description

check.true.binary returns TRUE if all of the elements in the argument are either integers of value 0 or 1. check.true.binary returns FALSE if not all of the elements in the argument are integers of value 0 or 1.

Usage

```
check.true.binary(bin.chain)
```

Arguments

bin.chain	A one dimensional vector.

Details

This function checks every element of its argument to see if they contain either the integer values 0 or 1.

Examples

```
check.true.binary(c(1,0,0,1,0,0,0,1))
check.true.binary(c("A","B","B","B","A","B","A","A"))
check.true.binary(c(TRUE,TRUE,TRUE,FALSE,FALSE,TRUE,FALSE,FALSE,TRUE,TRUE,
FALSE))
```

---

check.true.binary.multiple *Check if a two dimensional matrix has only integer elements 0 and 1.*

Description

check.true.binary.multiple returns TRUE if all of the elements in its argument are integers 0 or 1, and FALSE if not all of the values in its argument are integers 0 or 1.

Usage

```
check.true.binary.multiple(bin.chains)
```

Arguments

bin.chains	A two dimensional matrix.

Details

This function checks every row and column element of its argument to see if they contain either the integer values 0 or 1.

Examples

```
check.true.binary.multiple(matrix(data = c(1,0,1,0,1,0,0,1,1), ncol = 3))
check.true.binary.multiple(matrix(data = c("A","B","A","B","A","B","B","A",
"A"), ncol = 3))
check.true.binary.multiple(matrix(data = c(TRUE,TRUE,TRUE,FALSE,FALSE,TRUE,
FALSE,FALSE,TRUE,TRUE,FALSE), ncol = 3))
```

---

chiSqTestStat — *Calculates the Pearson's chi square test statistic for a single binary chain .*

Description

chiSqTestStat takes a binary chain of data and calculates the Pearson's chi square test statistic associated with it.

Usage

```
chiSqTestStat(binChain, nChainUniques)
```

Arguments

binChain	A single binary chain of data in the form of an integer vector.
nChainUniques	A integer value representing the number of unique values in binChain.

---

chiSqTestStatArray — *Calculate the chi square test statistics for many single binary chains.*

Description

chiSqTestStatArray takes a two dimensional matrix of many binary chains of data and returns a numeric vector filled with a chi square test statistic for each of them.

Usage

```
chiSqTestStatArray(binChains, nChainUniques)
```

Arguments

binChains	A integer matrix of binary chains of data, with each row being a different chain.
nChainUniques	A integer value representing the number of unique values in binChains.

`iDimSum` *Sum all the values in the first dimension of a three dimensional integer vector.*

Description

`iDimSum` takes a three dimensional integer vector, fixes the values of the second and third dimensions, and then sums the values of all the entries in the vector with those two values for its second and third dimensions.

Usage

```
iDimSum(n, j, k)
```

Arguments

`n`	A three dimensional integer vector.
`j`	An integer that is a valid index of the second dimension of `n`.
`k`	An integer that is a valid index of the third dimension of `n`.

`ikDimSum` *Sum all the values in the first and third dimensions of a three dimensional integer vector.*

Description

`ikDimSum` takes a three dimensional integer vector, fixes the value of the second dimension, and then sums the values of all the entries in the vector with that value for its second dimension.

Usage

```
ikDimSum(n, j)
```

Arguments

`n`	A three dimensional integer vector.
`j`	An integer that is a valid index of the second dimension of `n`.

indicateRun — *Indicate whether or not there is a run at a point in a integer vector.*

Description

indicateRun takes an integer vector binChain, and two integers, p and i, and tells the user if a run of length p starting and index i in the form of a Boolean value.

Usage

```
indicateRun(binChain, p, i)
```

Arguments

binChain	A binary chain of data in the form of an integer vector.
p	A integer value representing the length of the run to test for.
i	A integer value representing the location in binChain to test for a run starting at.

jDimSum — *Sum all the values in the second dimension of a three dimensional integer vector.*

Description

jDimSum takes a three dimensional integer vector, fixes the values of the first and third dimensions, and then sums the values of all the entries in the vector with those two values for its first and third dimensions.

Usage

```
jDimSum(n, i, k)
```

Arguments

n	A three dimensional integer vector.
i	An integer that is a valid index of the first dimension of n.
k	An integer that is a valid index of the third dimension of n.

jkDimSum *Sum all the values in the second and third dimensions of a three dimensional integer vector.*

Description

ikDimSum takes a three dimensional integer vector, fixes the value of the first dimension, and then sums the values of all the entries in the vector with that value for its first dimension.

Usage

```
jkDimSum(n, i)
```

Arguments

n	A three dimensional integer vector.
i	An integer that is a valid index of the first dimension of n.

kDimSum *Sum all the values in the third dimension of a three dimensional integer vector.*

Description

kDimSum takes a three dimensional integer vector, fixes the values of the first and second dimensions, and then sums the values of all the entries in the vector with those two values for its first and second dimensions.

Usage

```
kDimSum(n, i, j)
```

Arguments

n	A three dimensional integer vector.
i	An integer that is a valid index of the first dimension of n.
j	An integer that is a valid index of the second dimension of n.

madras *Data from the Madras schizophrenia study*

Description

A data set documenting the presence of thought disorders across 12 months since hospitalization. Each column represents a single patient, and each row represents a separate month. Months with thought disorders are encoded as 1 and those without are encoded as 0. Data is sourced from "Analysis of Longitudinal Data" by Peter J. Diggle, Patrick J. Heagerty, Kung-Yee Liang, and Scott L. Zeger. It was retrieved from http://faculty.washington.edu/heagerty/Books/AnalysisLongitudinal/madras.data . Individuals who did not have 12 months of post hospitalization data were removed to make the data rectangular.

Usage

```
madras
```

Format

A data frame with 69 rows and 12 columns.

metropolis *Generate independent data from a single binary chain.*

Description

`metropolis` takes a single binary chain of data in the form of an integer vector and generates `b` new independent chains of data, placing all of them in an integer matrix with the original data in the first row.

Usage

```
metropolis(binChain, m, b)
```

Arguments

`binChain`	A single binary chain of data represented by an integer vector.
`m`	An integer representing the number of swaps to be attempted.
`b`	An integer representing the number of new chains of data to be generated.

Details

`metropolis` works by taking the supplied `binChain`, and attempting `m` swaps on it, only preforming a swap of elements if doing so maintains the number of transitions between states in the resulting chain. `metropolis` then takes the resulting chain, and attempts `m` swaps on it again, then saving the resulting vector in a new row of an output matrix. `metropolis` does this `b` times, each time saving the resulting vector. Once all of the new data has been generated, `metropolis` returns the newly built integer matrix, of which the first row is the original chain of data `binChain`.

`multiple.binary.test` *Perform goodness-of-fit tests on multiple binary chains.*

Description

`multiple.binary.test` is used to preform goodness-of-fit tests on multiple binary chains of data of the same length to see if a Markov chain model is appropriate.

Usage

```
multiple.binary.test(binary.chains, swaps = 1000, n = 1000, run = 4,
  bins = 30)
```

Arguments

`binary.chains`	A two dimensional matrix, in which there are two unique values.
`swaps`	A positive nonzero integer value for the number of swaps to be attempted on the chain. Larger values will tend to yield "more independent" data. Generally, the number of swaps should be much larger then the number of elements in the matrix `binary.chains`.
`n`	A positive nonzero integer representing the number of new sets of chains to be generated.
`run`	The length of the run to test for if one is interested in run test statistics.
`bins`	The number of bins to be displayed in histograms of test statistics when one plots objects generated by `multiple.binary.test`.

Details

`multiple.binary.test` works by taking the supplied `binary.chains` parameter, counting the transitions between different elements, and then generating `n` new sets of chains with the same number of transitions. It generates these new sets of chains by attempting to swap random elements of the chains `swaps` times, only doing so if the attempted swap preserves the number of transitions between the two unique elements of the chains. `multiple.binary.test` then saves the chain generated by this process, then preforms a number of swaps equivalent to the value of `swaps` on that chain again, then recording the result in a new entry in a list of data. `multiple.binary.test` does this `n` times to generate the `n` new sets of chains. These new sets of chains are effectively independent of the original one.

Once `multiple.binary.test` has generated new data, it preforms various tests of that data. included in the function are the likelihood ratio test, the Pearson's chi square test, and a run test for a run of length specified by the argument `run`.

Value

`multiple.binary.test` returns a list of class `"multiple.binary.test"` with the following elements:

`data`, a list of matrices, the first of which is binary.chains, and the rest of which are the generated data.

`test.stats.lrt`, vector of likelihood ratio test statistics for each element of list `data`.

`test.stats.chisq`, a vector of Pearson's chi square test statistics for each element of list `data`.

test.stats.run, a vector of run test statistics for a run of length run for each element of list data.

p.value.lrt, the p-value of binary.chain, calculated exactly from the distribution of test.stats.lrt.

p.value.chisq, the p-value of binary.chain, calculated exactly from the distribution of test.stats.chisq.

p.value.run, the p-value of binary.chain, calculated exactly from the distribution of test.stats.run.

call, the function call.

bins, the number of bins specified in the function call.

run, the length of run specified in the function call.

Examples

```
data <- as.matrix(maRkov::snoqualmie)
foo <- multiple.binary.test(binary.chains = data, swaps = 10000, n = 10000,
                            run = 3, bins = 50)
```

---

multipleChiSqTestStat *Calculate the Pearson's chi square test statistic for a set of binary chains of data.*

Description

multipleChiSqTestStat takes a two dimensional integer vector binChains in which each row represents a single binary chain of data, and calculates a Pearson's chi square test statistic for the entire set.

Usage

```
multipleChiSqTestStat(binChains, nChainUniques)
```

Arguments

binChains	A two dimensional integer vector where each row is a separate binary chain of data.
nChainUniques	An integer value representing the number of unique elements in the set of chains binChains.

---

multipleChiSqTestStatArray *Calculate Pearson's chi square test statistics for many sets of binary chains of data.*

Description

multipleChiSqTestStatArray takes a three dimensional vector containing multiple sets of binary chains of data, and returns a numeric vector with entries corresponding to the Pearson's chi square test statistics of each set of binary chains of data.

Usage

```
multipleChiSqTestStatArray(binChains, nChainUniques)
```

Arguments

binChains	A three dimensional vector containing sets of chains of binary data.
nChainUniques	An integer value representing the number of unique elements in the set of chains binChains.

---

multipleIndicateRun *Indicate whether or not a run of a certain length exists starting at a certain point.*

Description

multipleIndicateRun takes a single binary chain binChain, a valid index of that chain i, and a length of run p and tests whether or not a run of that length starts at index i.

Usage

```
multipleIndicateRun(binChain, p, i)
```

Arguments

binChain	A one dimensional integer vector representing a binary chain of data.
p	An integer representing the length of run to test for.
i	An integer representing a valid index of binChain.

---

multipleRunTestStat *Calculate the run test statistic for a set of binary chains of data and a run of a certain length.*

Description

multipleRunTestStat takes a two dimensional integer vector binChains in which each row represents a single binary chain of data, and calculates a run test statistic for a run of length p for the entire set.

Usage

```
multipleRunTestStat(binChains, p)
```

Arguments

binChains	A two dimensional integer vector where each row is a separate binary chain of data.
p	An integer value representing the length of run to test for.

`multipleRunTestStatArray` *Calculate run test statistics for many sets of chains of binary data.*

Description

`multipleRunTestStatArray` takes a three dimensional integer vector containing multiple sets of binary chains of data, and returns a numeric vector with entries corresponding to the run test statistics for runs of length p for each set of binary chains of data.

Usage

```
multipleRunTestStatArray(binChains, p)
```

Arguments

`binChains`	A three dimensional integer vector containing sets of chains of binary data.
`p`	An integer representing the length of run to test for.

`nCounts` *Second order transition counts for a single binary chain.*

Description

`nCounts` counts the number of second order transitions in a binary chain of data, then returns a three dimensional vector whose indices represent the type of transition, and whose values represent the number of times that each transition occurs in the chain.

Usage

```
nCounts(binChain, nChainUniques)
```

Arguments

`binChain`	An integer vector representing a chain of data.
`nChainUniques`	The number or unique values in the chain `binChains`, represented as an integer value.

nCountsMultiple *Second order transition counts for multiple binary chains.*

Description

nCountsMultiple counts the number of second order transitions in a integer matrix whose rows represent individual binary chains. It returns a three dimensional vector whose indices represent the type of transition, and whose values represent the number of times that each transition occurs in the set of chains.

Usage

```
nCountsMultiple(binChains, nChainUniques)
```

Arguments

binChains	A two dimensional integer vector, each of whose rows represents a single binary chain of data.
nChainUniques	The number of unique values in the set of chains binChains, represented as an integer value.

ok_tornado *Tornado data for Oklahoma, 1950 - 2015*

Description

A data set documenting the presence of tornadic activity in Oklahoma. Each row represents a year, starting with 1950, and ending in 2015, and each column is a day of that year. Each day on which a tornado occurred is marked encoded as a 1 and each day without tornadic activity is encoded as a 0. The data is inspired by, but not drawn from, the paper "A Markov Chain Model of Tornadic Activity" by Mathias Drton, Caren Marzban, Peter Guttorp, and Joseph T. Schaefer. As mentioned in their paper, "day 366 of a non-leap year is coded as a non-tornadic day. This is not expected to adversely affect the results." Data is sourced from the National Weather Service Weather Forecast Office in Norman, Oklahoma, and can be found at http://www.srh.noaa.gov/oun/?n=tornadodata-ok-monthlyannual .

Usage

```
ok_tornado
```

Format

A data frame with 66 rows and 366 columns.

`plot.multiple.binary.test`
Produce plots from objects of class "multiple.binary.test".

Description

Produces some interesting plots of data and test statistics of objects of class `multiple.binary.test`.

Usage

```
## S3 method for class 'multiple.binary.test'
plot(x, ...)
```

Arguments

`x`	An object of `class` `multiple.binary.test`.
`...`	Further arguments passed to or from other methods.

`plot.single.binary.test`
Produce plots from objects of class "single.binary.test".

Description

Produces some interesting plots of data and test statistics of objects of class `single.binary.test`.

Usage

```
## S3 method for class 'single.binary.test'
plot(x, ...)
```

Arguments

`x`	An object of `class` `single.binary.test`.
`...`	Further arguments passed to or from other methods.

`print.multiple.binary.test` *Print instructions for examining objects of class "multiple.binary.test".*

Description

Instruct user to use `summary` and `print` to examine objects of class `multiple.binary.test`.

Usage

```
## S3 method for class 'multiple.binary.test'
print(x, ...)
```

Arguments

`x`	An object of `class` `multiple.binary.test`.
`...`	Further arguments passed to or from other methods.

`print.single.binary.test` *Print instructions for examining objects of class "single.binary.test".*

Description

Instruct user to use `summary` and `print` to examine objects of class `single.binary.test`.

Usage

```
## S3 method for class 'single.binary.test'
print(x, ...)
```

Arguments

`x`	An object of `class` `single.binary.test`.
`...`	Further arguments passed to or from other methods.

runTestStat *Calculate the run test statistic for a single binary chain.*

Description

runTestStat takes an integer vector binChain of a chain of binary data, and a integer p representing the length of run to test for. It returns the run test stat for that chain of data.

Usage

```
runTestStat(binChain, p)
```

Arguments

binChain	A binary chain of data in the form of an integer vector.
p	An integer greater than one representing the length of run to test for.

runTestStatArray *Calculate run test statistics for many binary chains.*

Description

runTestStatArray takes an integer matrix with each row denoting a binary chain of data and returns an integer vector with run test statistics for runs of length p corresponding to each binary chain.

Usage

```
runTestStatArray(binChains, p)
```

Arguments

binChains	A two dimensional integer matrix with each row denoting a individual binary chain of data.
p	An integer value representing the length of run to test for.

`single.binary.test` *Perform goodness-of-fit tests on a single binary chain.*

Description

`single.binary.test` is used to preform goodness-of-fit tests on single binary chains of data to see if a Markov chain model is appropriate.

Usage

```
single.binary.test(binary.chain, swaps = 1000, n = 1000, run = 4,
  tiles = 30, bins = 30)
```

Arguments

`binary.chain`	A one dimensional vector with two unique values.
`swaps`	A positive nonzero integer value for the number of swaps to be attempted on the chain. Larger numbers will tend to yield "more independent" data. Generally, the number of swaps should be far greater than the length of `binary.chain`.
`n`	A positive nonzero integer value representing the number of new chains to be generated.
`run`	The length of run to test for if one is interested in run test statistics.
`tiles`	The number of chains to be represented in the tile plot when one plots objects generated by `single.binary.test`
`bins`	The number of bins to be displayed in histograms of test statistics when one plots objects generated by `single.binary.test`.

Details

`single.binary.test` works by taking the supplied `binary.chain` parameter, counting the transitions between different elements, and then generating `n` new chains with the same number of transitions. It generates these new chains by attempting to swap random elements of the chain `swaps` times, only doing so if the attempted swap preserves the number of transitions between the two unique elements of the chain. `single.binary.test` then saves the chain generated by this process, then preforms a number of swaps equivalent to the value of `swaps` on that chain again, then recording the result in a matrix of new data. `single.binary.test` does this `n` times to generate the `n` new chains. These new chains are effectively independent of the original one.

Once `single.binary.test` has generated new data, it preforms various tests on that data. Included in the function are the likelihood ratio test, the Pearson's chi square test, and a run test for a run of length specified by the argument `run`.

Value

`single.binary.test` returns a list of class `"single.binary.test"` with the following elements:

`data`, a matrix of data with `binary.chain` in the first row, and the generated `n` rows of data in the following columns.

`test.stats.lrt`, a vector of likelihood ratio test statistics for each row of data in `data`.

`test.stats.chisq`, a vector of Pearson's chi square test statistics for each row of data in `data`.

`test.stats.run`, a vector of run test statistics for a run of length `run` for each row of data in `data`.

`p.value.lrt`, the p-value of `binary.chain` calculated exactly from the distribution of `test.stats.lrt`

`p.value.chisq`, the p-value of `binary.chain` calculated exactly from the distribution of `test.stats.chisq`

`p.value.run`, the p-value of `binary.chain` calculated exactly from the distribution of `test.stats.run`

`call`, the function call.

`run`, the length of run specified in the function call.

`tiles`, the number of tiles specified in the function call.

`bins`, the number of bins specified in the function call.

Examples

```
data <- c(1,1,1,1,1,1,1,1,1,1,1,1,0,1,1,1,0,1,1,1,1,1,1,1,1,1,0,0,0)
foo <- single.binary.test(binary.chain = data, swaps = 10000, n = 10000,
run = 4, tiles = 32, bins = 100)
```

---

snoqualmie *Rainfall data for the month of January in Snoqualmie Falls, 1948 - 1983.*

Description

A data set documenting the presence of precipitation in Snoqualmie Falls Washington on every day of every January from 1948 to 1983. 1 indicates a day with >=0.01 inches of precipitation, 0 indicates a day with <0.01 inches of precipitation. Data sourced from Peter Guttorp's Stochastic Modeling of Scientific Data, 1995.

Usage

```
snoqualmie
```

Format

A data frame with 36 rows and 31 columns.

---

summary.multiple.binary.test *Produce a summary of objects of class "multiple.binary.test".*

Description

Prints test statistics and p-values for objects of class `multiple.binary.test`.

Usage

```
## S3 method for class 'multiple.binary.test'
summary(object, ...)
```

Arguments

object	An object of class multiple.binary.test.
...	Further arguments passed to or from other methods.

---

summary.single.binary.test *Produce a summary of objects of class "single.binary.test".*

Description

Prints test statistics, p-values, and provides sample data for objects of class single.binary.test.

Usage

```
## S3 method for class 'single.binary.test'
summary(object, ...)
```

Arguments

object	An object of class single.binary.test.
...	Further arguments passed to or from other methods.

---

swap *Swap elements of single binary chains*

Description

swap is used to swap elements of a single binary chain if doing so maintains the same number of transitions between the two states of that chain.

Usage

```
swap(binChain, m)
```

Arguments

binChain	A binary one dimensional integer vector.
m	A integer value representing the number of swaps to attempt.

Details

swap takes a one dimensional vector of integers binChain and an integer m. It attempts to swap elements of binChain m times, each time only completing the swap if it does not affect the number of transitions between states in the sequence.

swapMult *Swap elements of multiple binary chains*

Description

swapMult is used to swap elements of multiple binary chains if doing so maintains the same number of transitions between the two states of those chains.

Usage

```
swapMult(binChains, m)
```

Arguments

binChains	A two dimensional integer vector with binary values.
m	A positive nonzero integer value for the attempted number of swaps to attempt on binChains.

Details

swapMult works by taking a two dimensional integer vector binChains and m, a number of times to attempt swaps. It generates random integers which are valid indices of the two dimensional vector binChains and tries to swap the elements of the vector at the indices that it generates, only doing so if this preserves the total number of transitions between states. After attempting m swaps, swapMult returns the new, freshly swapped two dimensional vector of binary chains.

u1TestStat *Calculates the likelihood ratio test statistic for a single binary chain.*

Description

u1TestStat takes a binary chain of data and calculates the likelihood ratio test statistic associated with it.

Usage

```
u1TestStat(binChain, nChainUniques)
```

Arguments

binChain	A binary chain of data in the form of a one dimensional integer vector.
nChainUniques	A integer value representing the number of unique values in binChain.

u1TestStatArray — *Calculate likelihood ratio test statistics for many binary chains.*

Description

u1TestStatArray takes an integer matrix with each row denoting a binary chain of data and returns an integer vector with likelihood ratio test statistics corresponding to each binary chain.

Usage

```
u1TestStatArray(binChains, nChainUniques)
```

Arguments

binChains	A two dimensional integer matrix with each row denoting a individual binary chain of data.
nChainUniques	An integer value representing the number of unique values in the binary chains found in binChains.

u6Metropolis — *Generate independent data from a set of binary chains.*

Description

u6Metropolis takes a set of binary chains of data in the form of an integer matrix and returns a three dimensional integer vector with with the first entry of the first dimension filled with the original set of binary chains and the rest filled with independent chains generated by u6Metropolis.

Usage

```
u6Metropolis(binChains, m, b)
```

Arguments

binChains	An integer matrix whose rows represent separate binary chains of data.
m	An integer value representing the number of swaps to be attempted.
b	An integer value representing the number of new sets of data to be generated.

Details

u6Metropolis works by taking a supplied set of binary chains binChains and attempting a number m swaps on entries of those chains, only swapping if doing so maintains the number of transitions between states that existed in the initial set of chains binChains. After it does this, it repeats the process on the newly generated set of binary chains of data b times, each time saving the new set of chains in a three dimensional vector of data. The first entry of the first dimension of this vector is used to store the original set of binary chains binChains.

`u6TestStat` *Calculate the likelihood ratio test statistic for a set of binary chains of data.*

Description

`u6TestStat` takes a two dimensional integer vector `binChains` in which each row represents a single binary chain of data, and calculates a likelihood ratio test statistic for the entire set.

Usage

```
u6TestStat(binChains, nChainUniques)
```

Arguments

`binChains`	A two dimensional integer vector where each row is a separate binary chain of data.
`nChainUniques`	An integer value representing the number of unique elements in the set of chains `binChains`.

`u6TestStatArray` *Calculate likelihood ratio test statistics for many sets of binary chains of data.*

Description

`u6TestStatArray` takes a three dimensional vector containing multiple sets of binary chains of data, and returns a numeric vector with entries corresponding to the likelihood ratio test statistics of each set of binary chains of data.

Usage

```
u6TestStatArray(binChains, nChainUniques)
```

Arguments

`binChains`	A three dimensional vector containing sets of chains of binary data.
`nChainUniques`	An integer value representing the number of unique elements in the set of chains `binChains`.

`vecGreaterThan` *Find the number of entries in a vector greater or equal to the value of the first entry.*

Description

`vecGreaterThan` counts the number of entries in a numeric vector `testStats` whose values are greater than that of the value of the first element of the vector.

Usage

```
vecGreaterThan(testStats)
```

Arguments

`testStats`	A one dimensional numeric vector.

Index