Package ‘markovchain’

March 7, 2020

Type Package
Title Easy Handling Discrete Time Markov Chains
Version 0.8.3
Date 2020-03-10
Maintainer Giorgio Alfredo Spedicato <spedicato_giorgio@yahoo.it>
Description Functions and S4 methods to create and manage discrete time Markov chains more easily. In addition functions to perform statistical (fitting and drawing random variates) and probabilistic (analysis of their structural proprieties) analysis are provided.
License GPL-2
Depends R (>= 3.6.0), methods
Imports igraph, Matrix, matlab, expm, stats4, parallel, Rcpp (>= 1.0.2), RcppParallel, utils, stats, grDevices
Suggests knitr, testthat, diagram, DiagrammeR, msm, etm, Rsolnp, knitcitations, rmarkdown, ctmcd, bookdown, rticles
VignetteBuilder utils, knitr
LinkingTo Rcpp, RcppParallel, RcppArmadillo (>= 0.9.600.4.0)
SystemRequirements GNU make
LazyLoad yes
ByteCompile yes
Encoding UTF-8
BugReports http://github.com/spedygiorgio/markovchain/issues
URL http://github.com/spedygiorgio/markovchain/
RoxygenNote 7.0.2
NeedsCompilation yes
Author Giorgio Alfredo Spedicato [aut, cre]
  (<https://orcid.org/0000-0002-0315-8888>),
  Tae Seung Kang [aut],
  Sai Bhargav Yalamanchi [aut],
  1
Mildenberger Thoralf [ctb] (<https://orcid.org/0000-0001-7242-1873>),
Deepak Yadav [aut],
Ignacio Cordón [aut] (<https://orcid.org/0000-0002-3152-0231>),
Vandit Jain [ctb],
Toni Giorgino [ctb] (<https://orcid.org/0000-0001-6449-0596>)

**Repository**  CRAN

**Date/Publication**  2020-03-07 06:00:08 UTC

### R topics documented:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>markovchain-package</td>
<td>3</td>
</tr>
<tr>
<td>absorptionProbabilities</td>
<td>4</td>
</tr>
<tr>
<td>blanden</td>
<td>5</td>
</tr>
<tr>
<td>committorAB</td>
<td>6</td>
</tr>
<tr>
<td>conditionalDistribution</td>
<td>7</td>
</tr>
<tr>
<td>craigsendi</td>
<td>8</td>
</tr>
<tr>
<td>createSequenceMatrix</td>
<td>8</td>
</tr>
<tr>
<td>ctmc-class</td>
<td>11</td>
</tr>
<tr>
<td>ctmcFit</td>
<td>12</td>
</tr>
<tr>
<td>expectedRewards</td>
<td>13</td>
</tr>
<tr>
<td>expectedRewardsBeforeHittingA</td>
<td>14</td>
</tr>
<tr>
<td>ExpectedTime</td>
<td>15</td>
</tr>
<tr>
<td>firstPassage</td>
<td>16</td>
</tr>
<tr>
<td>firstPassageMultiple</td>
<td>17</td>
</tr>
<tr>
<td>fitHigherOrder</td>
<td>18</td>
</tr>
<tr>
<td>fitHighOrderMultivarMC</td>
<td>19</td>
</tr>
<tr>
<td>freq2Generator</td>
<td>20</td>
</tr>
<tr>
<td>generatorToTransitionMatrix</td>
<td>21</td>
</tr>
<tr>
<td>HigherOrderMarkovChain-class</td>
<td>22</td>
</tr>
<tr>
<td>hittingProbabilities</td>
<td>22</td>
</tr>
<tr>
<td>holson</td>
<td>23</td>
</tr>
<tr>
<td>hommc-class</td>
<td>24</td>
</tr>
<tr>
<td>ictmc-class</td>
<td>24</td>
</tr>
<tr>
<td>impreciseProbabilityatT</td>
<td>25</td>
</tr>
<tr>
<td>inferHyperparam</td>
<td>26</td>
</tr>
<tr>
<td>is.accessible</td>
<td>27</td>
</tr>
<tr>
<td>is.CTMCirreducible</td>
<td>28</td>
</tr>
<tr>
<td>is.irreducible</td>
<td>29</td>
</tr>
<tr>
<td>is.regular</td>
<td>30</td>
</tr>
<tr>
<td>is.TimeReversible</td>
<td>31</td>
</tr>
<tr>
<td>kullback</td>
<td>32</td>
</tr>
<tr>
<td>markovchain-class</td>
<td>32</td>
</tr>
<tr>
<td>markovchainList-class</td>
<td>35</td>
</tr>
<tr>
<td>markovchainListFit</td>
<td>36</td>
</tr>
<tr>
<td>markovchainSequence</td>
<td>37</td>
</tr>
<tr>
<td>meanAbsorptionTime</td>
<td>38</td>
</tr>
<tr>
<td>meanFirstPassageTime</td>
<td>39</td>
</tr>
</tbody>
</table>
markovchain-package

Easy Handling Discrete Time Markov Chains

Description

The package contains classes and method to create and manage (plot, print, export for example) discrete time Markov chains (DTMC). In addition it provide functions to perform statistical (fitting and drawing random variates) and probabilistic (analysis of DTMC proprieties) analysis

Details

Package: markovchain
Type: Package
Version: 0.8.2
Date: 2020-01-5
License: GPL-2
Depends: R (>= 3.6.0), methods, expm, matlab, igraph, Matrix
Author(s)
Giorgio Alfredo Spedicato Maintainer: Giorgio Alfredo Spedicato <spedicato_giorgio@yahoo.it>

References
Discrete-Time Markov Models, Bremaud, Springer 1999

Examples

```r
# create some markov chains
statesNames=c("a","b")
mcA<-new("markovchain", transitionMatrix=matrix(c(0.7,0.3,0.1,0.9),byrow=TRUE, nrow=2, dimnames=list(statesNames,statesNames)))

statesNames=c("a","b","c")
mcB<-new("markovchain", states=statesNames, transitionMatrix= matrix(c(0.2,0.5,0.3,0.1,0.01,0.8,0.1), nrow=3, byrow=TRUE, dimnames=list(statesNames, statesNames)))

statesNames=c("a","b","c","d")
matrice<-matrix(c(0.25,0.75,0,0,0.4,0.6,0,0,0.04,0.06,0,0,0.1,0.9,0,0,0.7,0.3), nrow=4, byrow=TRUE)
mcC<-new("markovchain", states=statesNames, transitionMatrix=matrice)
mcD<-new("markovchain", transitionMatrix=matrix(c(0,1,0,1), nrow=2,byrow=TRUE))

#operations with S4 methods
mcA^2
steadyStates(mcB)
absorbingStates(mcB)
markovchainSequence(n=20, markovchain=mcC, include=TRUE)
```

---

### absorptionProbabilities

**Absorption probabilities**

**Description**
Computes the absorption probability from each transient state to each recurrent one (i.e. the (i, j) entry or (j, i), in a stochastic matrix by columns, represents the probability that the first not transient state we can go from the transient state i is j (and therefore we are going to be absorbed in the communicating recurrent class of j)

**Usage**

```r
absorptionProbabilities(object)
```

**Arguments**

- **object** the markovchain object
Value

A named vector with the expected number of steps to go from a transient state to any of the recurrent ones

Author(s)

Ignacio Cordón

References


Examples

```r
m <- matrix(c(1/2, 1/2, 0,
              1/2, 1/2, 0,
              0, 1/2, 1/2), ncol = 3, byrow = TRUE)
mc <- new("markovchain", states = letters[1:3], transitionMatrix = m)
absorptionProbabilities(mc)
```

blanden

<table>
<thead>
<tr>
<th>Mobility between income quartiles</th>
</tr>
</thead>
</table>

Description

This table show mobility between income quartiles for father and sons for the 1970 cohort born

Usage

data(blanden)

Format

An object of class table with 4 rows and 4 columns.

Details

The rows represent fathers’ income quartile when the son is aged 16, whilst the columns represent sons’ income quartiles when he is aged 30 (in 2000).

Source

Personal reworking

References

Jo Blanden, Paul Gregg and Stephen Machin, Intergenerational Mobility in Europe and North America, Center for Economic Performances (2005)
Examples

```r
data(blanden)
mobilityMc<-as(blanden, "markovchain")
```

---

**committorAB**

*Calculates committor of a markovchain object with respect to set A, B*

---

**Description**

Returns the probability of hitting states from set A before set B with different initial states

**Usage**

```r
committorAB(object,A,B,p)
```

**Arguments**

- `object` a markovchain class object
- `A` a set of states
- `B` a set of states
- `p` initial state (default value : 1)

**Details**

The function solves a system of linear equations to calculate probability that the process hits a state from set A before any state from set B

**Value**

Return a vector of probabilities in case initial state is not provided else returns a number

**Examples**

```r
transMatr <- matrix(c(0,0,0,1,0.5,
                      0.5,0,0,0,0,
                      0.5,0,0,0,0,
                      0.2,0.4,0,0,0,
                      0,0.8,0.6,0,0.5),
                     nrow = 5)
object <- new("markovchain", states=c("a","b","c","d","e"),transitionMatrix=transMatr)
committorAB(object,c(5),c(3))
```
Description

It extracts the conditional distribution of the subsequent state, given current state.

Usage

conditionalDistribution(object, state)

Arguments

object A markovchain object.
state Subsequent state.

Value

A named probability vector

Author(s)

Giorgio Spedicato, Deepak Yadav

References

A First Course in Probability (8th Edition), Sheldon Ross, Prentice Hall 2010

See Also

markovchain

Examples

# define a markov chain
statesNames <- c("a", "b", "c")
markovB <- new("markovchain", states = statesNames, transitionMatrix =
    matrix(c(0.2, 0.5, 0.3, 0, 1, 0, 0.1, 0.8, 0.1), nrow = 3,
    byrow = TRUE, dimnames = list(statesNames, statesNames)))
conditionalDistribution(markovB, "b")
createSequenceMatrix

Description

Given a sequence of states arising from a stationary state, it fits the underlying Markov chain distribution using either MLE (also using a Laplacian smoother), bootstrap or by MAP (Bayesian) inference.

craigsendi

CD4 cells counts on HIV Infects between zero and six month

Description

This is the table shown in Craig and Sendi paper showing zero and six month CD4 cells count in six brackets

Usage

data(craigsendi)

Format

The format is: table [1:3, 1:3] 682 154 19 33 64 19 25 47 43 - attr(*, "dimnames")=List of 2 ..$ : chr [1:3] "0-49" "50-74" "75-UP" ..$ : chr [1:3] "0-49" "50-74" "75-UP"

Details

Rows represent counts at the beginning, cols represent counts after six months.

Source


References

see source

Examples

data(craigsendi)
csMc<-as(craigsendi, "markovchain")steadyStates(csMc)
createSequenceMatrix

Usage

createSequenceMatrix(
  stringchar,
  toRowProbs = FALSE,
  sanitize = FALSE,
  possibleStates = character()
)

markovchainFit(
  data,
  method = "mle",
  byrow = TRUE,
  nboot = 10L,
  laplacian = 0,
  name = "",
  parallel = FALSE,
  confidencelevel = 0.95,
  confint = TRUE,
  hyperparam = matrix(),
  sanitize = FALSE,
  possibleStates = character()
)

Arguments

stringchar  It can be a n x n matrix or a character vector or a list

toRowProbs  converts a sequence matrix into a probability matrix

sanitize  put 1 in all rows having rowSum equal to zero

possibleStates  Possible states which are not present in the given sequence

data  It can be a character vector or a n x n matrix or a n x n data frame or a list

method  Method used to estimate the Markov chain. Either "mle", "map", "bootstrap" or "laplace"

byrow  it tells whether the output Markov chain should show the transition probabilities by row.

nboot  Number of bootstrap replicates in case "bootstrap" is used.

laplacian  Laplacian smoothing parameter, default zero. It is only used when "laplace" method is chosen.

name  Optional character for name slot.

parallel  Use parallel processing when performing Bootstrap estimates.

confidencelevel  \( \alpha \)

level for confidence intervals width. Used only when method equal to "mle".
confint  a boolean to decide whether to compute Confidence Interval or not.
hypermparam  Hyperparameter matrix for the a priori distribution. If none is provided, default value of 1 is assigned to each parameter. This must be of size k x k where k is the number of states in the chain and the values should typically be non-negative integers.

Details
Disabling confint would lower the computation time on large datasets. If data or stringchar contain NAs, the related NA containing transitions will be ignored.

Value
A list containing an estimate, log-likelihood, and, when "bootstrap" method is used, a matrix of standards deviations and the bootstrap samples. When the "mle", "bootstrap" or "map" method is used, the lower and upper confidence bounds are returned along with the standard error. The "map" method also returns the expected value of the parameters with respect to the posterior distribution.

Note
This function has been rewritten in Rcpp. Bootstrap algorithm has been defined "heuristically". In addition, parallel facility is not complete, involving only a part of the bootstrap process. When data is either a data.frame or a matrix object, only MLE fit is currently available.

Author(s)
Giorgio Spedicato, Tae Seung Kang, Sai Bhargav Yalamanchi

References
A First Course in Probability (8th Edition), Sheldon Ross, Prentice Hall 2010
Inferring Markov Chains: Bayesian Estimation, Model Comparison, Entropy Rate, and Out-of-Class Modeling, Christopher C. Strelioff, James P. Crutchfield, Alfred Hubler, Santa Fe Institute
Yalamanchi SB, Spedicato GA (2015). Bayesian Inference of First Order Markov Chains. R package version 0.2.5

See Also
markovchainSequence, markovchainListFit

Examples
sequence <- c("a", "b", "a", "a", "a", "a", "b", "a", "b", "a", "a", "a", "b", "b", "a", "a", "b", "b", "a")
sequenceMatr <- createSequenceMatrix(sequence, sanitize = FALSE)
mcFitMLE <- markovchainFit(data = sequence)
mcFitBSP <- markovchainFit(data = sequence, method = "bootstrap", nboot = 5, name = "Bootstrap Mc")
na.sequence <- c("a", NA, "a", "b")
# There will be only a (a,b) transition
ctmc-class

na.sequenceMatr <- createSequenceMatrix(na.sequence, sanitize = FALSE)
mcFitMLE <- markovchainFit(data = na.sequence)

# data can be a list of character vectors
sequences <- list(x = c("a", "b", "a"), y = c("b", "a", "b", "a", "c"))
mcFitMap <- markovchainFit(sequences, method = "map")
mcFitMle <- markovchainFit(sequences, method = "mle")

ctmc-class

Continuous time Markov Chains class

Description

The S4 class that describes ctmc (continuous time Markov chain) objects.

Arguments

states Name of the states. Must be the same of colnames and rownames of the generator matrix
byrow TRUE or FALSE. Indicates whether the given matrix is stochastic by rows or by columns
generator Square generator matrix
name Optional character name of the Markov chain

Methods

dim signature(x = "ctmc"): method to get the size
initialize signature(.Object = "ctmc"): initialize method
states signature(object = "ctmc"): states method.
steadyStates signature(object = "ctmc"): method to get the steady state vector.
plot signature(x = "ctmc", y = "missing"): plot method for ctmc objects

Note

1. ctmc classes are written using S4 classes
2. Validation method is used to assess whether either columns or rows totals to zero. Rounding is used up to 5th decimal. If state names are not properly defined for a generator matrix, coercing to ctmc object leads to overriding states name with artificial "s1", "s2", ... sequence

References

Introduction to Stochastic Processes with Applications in the Biosciences (2013), David F. Anderson, University of Wisconsin at Madison. Sai Bhargav Yalamanchi, Giorgio Spedicato

See Also
generatorToTransitionMatrix,rctmc
Examples

```r
energyStates <- c("sigma", "sigma_star")
byRow <- TRUE
gen <- matrix(data = c(-3, 3,
                       1, -1), nrow = 2,
                       byrow = byRow, dimnames = list(energyStates, energyStates))
molecularCTMC <- new("ctmc", states = energyStates,
                     byrow = byRow, generator = gen,
                     name = "Molecular Transition Model")
steadyStates(molecularCTMC)
```

---

**ctmcFit**

*Function to fit a CTMC*

**Description**

This function fits the underlying CTMC given the state transition data and the transition times using the maximum likelihood method (MLE).

**Usage**

```r
c tm cF it(data, byrow = TRUE, name = "", confidencelevel = 0.95)
```

**Arguments**

- `data`: It is a list of two elements. The first element is a character vector denoting the states. The second is a numeric vector denoting the corresponding transition times.
- `byrow`: Determines if the output transition probabilities of the underlying embedded DTMC are by row.
- `name`: Optional name for the CTMC.
- `confidencelevel`: Confidence level for the confidence interval construction.

**Details**

Note that in `data`, there must exist an element wise corresponding between the two elements of the list and that `data[2][1]` is always 0.

**Value**

It returns a list containing the CTMC object and the confidence intervals.

**Author(s)**

Sai Bhargav Yalamanchi
expectedRewards

References
Continuous Time Markov Chains (vignette), Sai Bhargav Yalamanchi, Giorgio Alfredo Spedicato 2015

See Also
rctmc

Examples

```r
data <- list(c("a", "b", "c", "a", "b", "a", "c", "b", "c"), c(0, 0.8, 2.1, 2.4, 4, 5, 5.9, 8.2, 9))
ctmcFit(data)
```

---

expectedRewards | Expected Rewards for a markovchain

Description
Given a markovchain object and reward values for every state, function calculates expected reward value after n steps.

Usage

```r
expectedRewards(markovchain, n, rewards)
```

Arguments

- `markovchain` the markovchain-class object
- `n` no of steps of the process
- `rewards` vector depicting rewards corresponding to states

Details
the function uses a dynamic programming approach to solve a recursive equation described in reference.

Value
returns a vector of expected rewards for different initial states

Author(s)
Vandit Jain

References
**Examples**

```r
transMatr<-matrix(c(0.99,0.01,0.01,0.99),nrow=2,byrow=TRUE)
simpleMc<-new("markovchain", states=c("a","b"),
  transitionMatrix=transMatr)
expectedRewards(simpleMc,1,c(0,1))
```

**expectedRewardsBeforeHittingA**

*Expected first passage Rewards for a set of states in a markovchain*

**Description**

Given a markovchain object and reward values for every state, function calculates expected reward value for a set A of states after n steps.

**Usage**

```r
expectedRewardsBeforeHittingA(markovchain, A, state, rewards, n)
```

**Arguments**

- `markovchain` the markovchain-class object
- `A` set of states for first passage expected reward
- `state` initial state
- `rewards` vector depicting rewards corresponding to states
- `n` no of steps of the process

**Details**

The function returns the value of expected first passage rewards given rewards corresponding to every state, an initial state and number of steps.

**Value**

returns a expected reward (numerical value) as described above

**Author(s)**

Sai Bhargav Yalamanchi, Vandit Jain
**ExpectedTime**

*Returns expected hitting time from state i to state j*

**Description**

Returns expected hitting time from state i to state j

**Usage**

```
ExpectedTime(C,i,j,useRCpp)
```

**Arguments**

- **C**: A CTMC S4 object
- **i**: Initial state i
- **j**: Final state j
- **useRCpp**: logical whether to use Rcpp

**Details**

According to the theorem, holding times for all states except j should be greater than 0.

**Value**

A numerical value that returns expected hitting times from i to j

**Author(s)**

Vandit Jain

**References**

*Markovchains, J. R. Norris, Cambridge University Press*

**Examples**

```r
states <- c("a","b","c","d")
byRow <- TRUE
gen <- matrix(data = c(-1, 1/2, 1/2, 0, 1/4, -1/2, 0, 1/4, 1/6, 0, -1/3, 1/6, 0, 0, 0, 0),
nrow = 4,byrow = byRow, dimnames = list(states,states))
ctmc <- new("ctmc",states = states, byrow = byRow, generator = gen, name = "testctmc")
ExpectedTime(ctmc,1,4,TRUE)
```
firstPassage  

First passage across states

Description

This function compute the first passage probability in states

Usage

```
firstPassage(object, state, n)
```

Arguments

- **object**: A `markovchain` object
- **state**: Initial state
- **n**: Number of rows on which compute the distribution

Details

Based on Feres' Matlab listings

Value

A matrix of size 1:n x number of states showing the probability of the first time of passage in states to be exactly the number in the row.

Author(s)

Giorgio Spedicato

References

Renaldo Feres, Notes for Math 450 Matlab listings for Markov chains

See Also

- `conditionalDistribution`

Examples

```
simpleMc <- new("markovchain", states = c("a", "b"),
  transitionMatrix = matrix(c(0.4, 0.6, .3, .7),
    nrow = 2, byrow = TRUE))
firstPassage(simpleMc, "b", 20)
```
**firstPassageMultiple**  
*function to calculate first passage probabilities*

**Description**

The function calculates first passage probability for a subset of states given an initial state.

**Usage**

```r
firstPassageMultiple(object, state, set, n)
```

**Arguments**

- `object`: a markovchain-class object
- `state`: initial state of the process (character vector)
- `set`: set of states A, first passage of which is to be calculated
- `n`: Number of rows on which compute the distribution

**Value**

A vector of size n showing the first time probabilities

**Author(s)**

Vandit Jain

**References**

Renaldo Feres, Notes for Math 450 Matlab listings for Markov chains; MIT OCW, course - 6.262, Discrete Stochastic Processes, course-notes, chap -05

**See Also**

`firstPassage`

**Examples**

```r
statesNames <- c("a", "b", "c")
markovB <- new("markovchain", states = statesNames, transitionMatrix =
matrix(c(0.2, 0.5, 0.3,
       0, 1, 0,
       0.1, 0.8, 0.1), nrow = 3, byrow = TRUE,
       dimnames = list(statesNames, statesNames))
)
firstPassageMultiple(markovB,"a",c("b","c"),4)
```
fitHigherOrder

Functions to fit a higher order Markov chain

Description
Given a sequence of states arising from a stationary state, it fits the underlying Markov chain distribution with higher order.

Usage
```r
fitHigherOrder(sequence, order = 2)
seq2freqProb(sequence)
seq2matHigh(sequence, order)
```

Arguments
- `sequence`: A character list.
- `order`: Markov chain order

Value
A list containing lambda, Q, and X.

Note
This function is written in Rcpp.

Author(s)
Giorgio Spedicato, Tae Seung Kang

References

Examples
```r
sequence <- c("a", "a", "b", "b", "a", "c", "b", "a", "b", "c", "a", "b",
               "c", "a", "b", "c", "a", "b", "a", "b")
fitHigherOrder(sequence)
```
fitHighOrderMultivarMC

Function to fit Higher Order Multivariate Markov chain

Description

Given a matrix of categorical sequences it fits Higher Order Multivariate Markov chain.

Usage

fitHighOrderMultivarMC(seqMat, order = 2, Norm = 2)

Arguments

seqMat a matrix or a data frame where each column is a categorical sequence
order Multivariate Markov chain order. Default is 2.
Norm Norm to be used. Default is 2.

Value

an hommc object

Author(s)

Giorgio Spedicato, Deepak Yadav

References

W.-K. Ching et al. / Linear Algebra and its Applications

Examples

data <- matrix(c('2', '1', '3', '3', '4', '3', '2', '1', '3', '3', '2', '1',
                '2', '4', '4', '4', '4', '2', '3', '3', '1', '4', '3', '3'),
               ncol = 2, byrow = FALSE)

fitHighOrderMultivarMC(data, order = 2, Norm = 2)
freq2Generator

Returns a generator matrix corresponding to frequency matrix

Description

The function provides interface to calculate generator matrix corresponding to a frequency matrix and time taken.

Usage

freq2Generator(P, t = 1, method = "QO", logmethod = "Eigen")

Arguments

P relative frequency matrix

(t) (default value = 1)

method one among "QO" (Quasi optimaisation), "WA" (weighted adjustment), "DA" (diagonal adjustment)

logmethod method for computation of matrix algorithm (by default : Eigen)

Value

returns a generator matrix with same dimnames

References


Examples

```r
call <- matrix(c(150,2,1,1,200,2,1,2,1,175,1,1,1,1,150),nrow = 4,byrow = TRUE)
sample_rel = rbind((call/rowSums(call))[1:dim(call)[1]-1,],c(rep(0,dim(call)[1]-1),1))
freq2Generator(sample_rel,1)

data(tm_abs)
tm_rel=rbind((tm_abs/rowSums(tm_abs))[1:7,],c(rep(0,7),1))
## Derive quasi optimization generator matrix estimate
freq2Generator(tm_rel,1)
```
**generatorToTransitionMatrix**

*Function to obtain the transition matrix from the generator*

**Description**

The transition matrix of the embedded DTMC is inferred from the CTMC's generator

**Usage**

```
generatorToTransitionMatrix(gen, byrow = TRUE)
```

**Arguments**

- `gen`: The generator matrix
- `byrow`: Flag to determine if rows (columns) sum to 0

**Value**

Returns the transition matrix.

**Author(s)**

Sai Bhargav Yalamanchi

**References**

*Introduction to Stochastic Processes with Applications in the Biosciences* (2013), David F. Anderson, University of Wisconsin at Madison

**See Also**

- `rctmc`
- `ctmc-class`

**Examples**

```r
enenergyStates <- c("sigma", "sigma_star")
byRow <- TRUE

gen <- matrix(data = c(-3, 3, 1, -1), nrow = 2,
byrow = byRow, dimnames = list(energyStates, energyStates))
generatorToTransitionMatrix(gen)
```
HigherOrderMarkovChain-class

*Higher order Markov Chains class*

**Description**

The S4 class that describes HigherOrderMarkovChain objects.

**hittingProbabilities**  *Hitting probabilities for markovchain*

**Description**

Given a markovchain object, this function calculates the probability of ever arriving from state i to j

**Usage**

hittingProbabilities(object)

**Arguments**

- **object**  the markovchain-class object

**Value**

a matrix of hitting probabilities

**Author(s)**

Ignacio Cordón

**References**

R. Vélez, T. Prieto, Procesos Estocásticos, Librería UNED, 2013

**Examples**

```r
M <- matlab::zeros(5, 5)
M[1,1] <- M[5,5] <- 1

mc <- new("markovchain", transitionMatrix = M)
hittingProbabilities(mc)
```
holson

| holson | Holson data set |

Description

A data set containing 1000 life histories trajectories and a categorical status (1,2,3) observed on eleven evenly spaced steps.

Usage

data(holson)

Format

A data frame with 1000 observations on the following 12 variables.

- id  unique id
- time1 observed status at i-th time
- time2 observed status at i-th time
- time3 observed status at i-th time
- time4 observed status at i-th time
- time5 observed status at i-th time
- time6 observed status at i-th time
- time7 observed status at i-th time
- time8 observed status at i-th time
- time9 observed status at i-th time
- time10 observed status at i-th time
- time11 observed status at i-th time

Details

The example can be used to fit a markovchain or a markovchainList object.

Source

Private communications

References

Private communications

Examples

data(holson)
head(holson)
hommc-class

An S4 class for representing High Order Multivariate Markovchain (HOMMC)

Description
An S4 class for representing High Order Multivariate Markovchain (HOMMC)

Slots
order an integer equal to order of Multivariate Markovchain
states a vector of states present in the HOMMC model
P array of transition matrices
Lambda a vector which stores the weightage of each transition matrices in P
byrow if FALSE each column sum of transition matrix is 1 else row sum = 1
name a name given to hommc

Author(s)
Giorgio Spedicato, Deepak Yadav

Examples

```r
statesName <- c("a", "b")
P <- array(0, dim = c(2, 2, 4), dimnames = list(statesName, statesName))
P[,1] <- matrix(c(0, 1, 1/3, 2/3), byrow = FALSE, nrow = 2)
P[,2] <- matrix(c(1/4, 3/4, 0, 1), byrow = FALSE, nrow = 2)
P[,3] <- matrix(c(1, 0, 1/3, 2/3), byrow = FALSE, nrow = 2)
P[,4] <- matrix(c(3/4, 1/4, 0, 1), byrow = FALSE, nrow = 2)
Lambda <- c(0.8, 0.2, 0.3, 0.7)

ob <- new("hommc", order = 1, states = statesName, P = P,
            Lambda = Lambda, byrow = FALSE, name = "FOMMC")
```

ictmc-class

An S4 class for representing Imprecise Continuous Time Markovchains

Description
An S4 class for representing Imprecise Continuous Time Markovchains
impreciseProbabilityatT

Slots

- states: a vector of states present in the ICTMC model
- Q: matrix representing the generator demonstrated in the form of variables
- range: a matrix that stores values of range of variables
- name: name given to ICTMC

impreciseProbabilityatT

Calculating full conditional probability using lower rate transition matrix

Description

This function calculates full conditional probability at given time \( s \) using lower rate transition matrix.

Usage

impreciseProbabilityatT(C, i, t, s, error, useRCpp)

Arguments

- \( C \): a ictmc class object
- \( i \): initial state at time \( t \)
- \( t \): initial time \( t \). Default value = 0
- \( s \): final time
- \( error \): error rate. Default value = 0.001
- \( useRCpp \): logical whether to use RC\text{pp} implementation; by default TRUE

Author(s)

Vandit Jain

References

Imprecise Continuous-Time Markov Chains, Thomas Krak et al., 2016

Examples

```
states <- c("n","y")
Q <- matrix(c(-1,1,1,-1),nrow = 2,byrow = TRUE,dimnames = list(states,states))
range <- matrix(c(1/52,3/52,1/2,2),nrow = 2,byrow = 2)
name <- "testictmc"
ictmc <- new("ictmc",states = states,Q = Q,range = range,name = name)
impreciseProbabilityatT(ictmc,2,0,1,10^-3,TRUE)
```
Function to infer the hyperparameters for Bayesian inference from an a priori matrix or a data set

Description

Since the Bayesian inference approach implemented in the package is based on conjugate priors, hyperparameters must be provided to model the prior probability distribution of the chain parameters. The hyperparameters are inferred from a given a priori matrix under the assumption that the matrix provided corresponds to the mean (expected) values of the chain parameters. A scaling factor vector must be provided too. Alternatively, the hyperparameters can be inferred from a data set.

Usage

inferHyperparam(transMatr = matrix(), scale = numeric(), data = character())

Arguments

- transMatr: A valid transition matrix, with dimension names.
- scale: A vector of scaling factors, each element corresponds to the row names of the provided transition matrix transMatr, in the same order.
- data: A data set from which the hyperparameters are inferred.

Details

transMatr and scale need not be provided if data is provided.

Value

Returns the hyperparameter matrix in a list.

Note

The hyperparameter matrix returned is such that the row and column names are sorted alphanumerically, and the elements in the matrix are correspondingly permuted.

Author(s)

Sai Bhargav Yalamanchi, Giorgio Spedicato

References

Yalamanchi SB, Spedicato GA (2015). Bayesian Inference of First Order Markov Chains. R package version 0.2.5
is.accessible

See Also
markovchainFit, predictiveDistribution

Examples

data(rain, package = "markovchain")
inferHyperparam(data = rain$rain)

weatherStates <- c("sunny", "cloudy", "rain")
weatherMatrix <- matrix(data = c(0.7, 0.2, 0.1,
                               0.3, 0.4, 0.3,
                               0.2, 0.4, 0.4),
                          byrow = TRUE, nrow = 3,
                          dimnames = list(weatherStates, weatherStates))
inferHyperparam(transMatr = weatherMatrix, scale = c(10, 10, 10))

is.accessible Verify if a state \( j \) is reachable from state \( i \).

Description
This function verifies if a state is reachable from another, i.e., if there exists a path that leads to state \( j \) leaving from state \( i \) with positive probability.

Usage
is.accessible(object, from, to)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>A markovchain object.</td>
</tr>
<tr>
<td>from</td>
<td>The name of state &quot;i&quot; (beginning state).</td>
</tr>
<tr>
<td>to</td>
<td>The name of state &quot;j&quot; (ending state).</td>
</tr>
</tbody>
</table>

Details
It wraps an internal function named reachabilityMatrix.

Value
A boolean value.

Author(s)
Giorgio Spedicato, Ignacio Cordón
References
James Montgomery, University of Madison

See Also
is.irreducible

Examples
statesNames <- c("a", "b", "c")
markovB <- new("markovchain", states = statesNames,
               transitionMatrix = matrix(c(0.2, 0.5, 0.3,
                                         0, 1, 0,
                                         0.1, 0.8, 0.1), nrow = 3, byrow = TRUE,
                                         dimnames = list(statesNames, statesNames)
               )
)
is.accessible(markovB, "a", "c")

is.CTMCirreducible   Check if CTMC is irreducible

Description
This function verifies whether a CTMC object is irreducible

Usage
is.CTMCirreducible(ctmc)

Arguments
cctmc       a ctmc-class object

Value
a boolean value as described above.

Author(s)
Vandit Jain

References
Continuous-Time Markov Chains, Karl Sigman, Columbia University
Examples

```r
energyStates <- c("sigma", "sigma_star")
byRow <- TRUE
gen <- matrix(data = c(-3, 3,
                        1, -1), nrow = 2,
                        byrow = byRow, dimnames = list(energyStates, energyStates))
molecularCTMC <- new("ctmc", states = energyStates,
                      byrow = byRow, generator = gen,
                      name = "Molecular Transition Model")
is.CTMCirreducible(molecularCTMC)
```

is.irreducible  Function to check if a Markov chain is irreducible (i.e. ergodic)

Description

This function verifies whether a markovchain object transition matrix is composed by only one communicating class.

Usage

```r
is.irreducible(object)
```

Arguments

- `object` A markovchain object

Details

It is based on .communicatingClasses internal function.

Value

A boolean values.

Author(s)

Giorgio Spedicato

References

Feres, Matlab listings for Markov Chains.

See Also

summary
Examples

```r
statesNames <- c("a", "b")
mcA <- new("markovchain", transitionMatrix = matrix(c(0.7,0.3,0.1,0.9),
byrow = TRUE, nrow = 2,
dimnames = list(statesNames, statesNames))
)
is.irreducible(mcA)
```

---

is.regular Check if a DTMC is regular

Description

Function to check whether a DTMC is regular

Usage

```r
is.regular(object)
```

Arguments

- `object` a markovchain object

Details

A Markov chain is regular if some of the powers of its matrix has all elements strictly positive

Value

A boolean value

Author(s)

Ignacio Cordón

References

Matrix Analysis. Roger A.Horn, Charles R.Johnson. 2nd edition. Corollary 8.5.8, Theorem 8.5.9

See Also

- `is.irreducible`
is.TimeReversible

Examples

\[
P \leftarrow \begin{pmatrix}
0.5 & 0.25 & 0.25 \\
0.5 & 0 & 0.5 \\
0.25 & 0.25 & 0.5
\end{pmatrix}, \text{nrow} = 3
\]

colnames(P) <- rownames(P) <- c("R","N","S")
ciao <- as(P, "markovchain")
is.regular(ciao)

is.TimeReversible checks if ctmc object is time reversible

Description

The function returns checks if provided function is time reversible

Usage

is.TimeReversible(ctmc)

Arguments

cutm\text{c} \hspace{1cm} \text{a ctmc-class object}

Value

Returns a boolean value stating whether ctmc object is time reversible

a boolean value as described above

Author(s)

Vandit Jain

References

INTRODUCTION TO STOCHASTIC PROCESSES WITH R, ROBERT P. DOBROW, Wiley

Examples

generStates <- c("sigma", "sigma_star")
byRow <- TRUE
gen <- matrix(data = c(-3, 3,
1, -1), nrow = 2,
byrow = byRow, dimnames = list(generStates, generStates))
molecularCTMC <- new("ctmc", states = generStates,
byrow = byRow, generator = gen,
name = "Molecular Transition Model")
is.TimeReversible(molecularCTMC)
Example from Kullback and Kupperman Tests for Contingency Tables

**Description**

A list of two matrices representing raw transitions between two states

**Usage**

data(kullback)

**Format**

A list containing two 6x6 non-negative integer matrices

---

**markovchain-class**

*Markov Chain class*

**Description**

The S4 class that describes `markovchain` objects.

**Arguments**

- **states**: Name of the states. Must be the same of colnames and rownames of the transition matrix
- **byrow**: TRUE or FALSE indicating whether the supplied matrix is either stochastic by rows or by columns
- **transitionMatrix**: Square transition matrix
- **name**: Optional character name of the Markov chain

**Creation of objects**

Objects can be created by calls of the form `new("markovchain", states, byrow, transitionMatrix, ...).

**Methods**

- `signature(e1 = "markovchain", e2 = "markovchain")`: multiply two `markovchain` objects
- `signature(e1 = "markovchain", e2 = "matrix")`: `markovchain` by matrix multiplication
- `signature(e1 = "markovchain", e2 = "numeric")`: `markovchain` by numeric vector multiplication
- `signature(e1 = "matrix", e2 = "markovchain")`: matrix by `markovchain`
* signature(e1 = "numeric", e2 = "markovchain"): numeric vector by markovchain multiplication
[ signature(x = "markovchain", i = "ANY", j = "ANY", drop = "ANY"): ...
^ signature(e1 = "markovchain", e2 = "numeric"): power of a markovchain object
== signature(e1 = "markovchain", e2 = "markovchain"): equality of two markovchain object
!= signature(e1 = "markovchain", e2 = "markovchain"): non-equality of two markovchain object

**absorbingStates** signature(object = "markovchain"): method to get absorbing states

**canonicForm** signature(object = "markovchain"): return a markovchain object into canonic form

**coerce** signature(from = "markovchain", to = "data.frame"): coerce method from markovchain to data.frame

**conditionalDistribution** signature(object = "markovchain"): returns the conditional probability of subsequent states given a state

**coerce** signature(from = "data.frame", to = "markovchain"): coerce method from data.frame to markovchain

**coerce** signature(from = "table", to = "markovchain"): coerce method from table to markovchain

**coerce** signature(from = "msm", to = "markovchain"): coerce method from msm to markovchain

**coerce** signature(from = "msm.est", to = "markovchain"): coerce method from msm.est (but only from a Probability Matrix) to markovchain

**coerce** signature(from = "etm", to = "markovchain"): coerce method from etm to markovchain

**coerce** signature(from = "sparseMatrix", to = "markovchain"): coerce method from sparseMatrix to markovchain

**coerce** signature(from = "markovchain", to = "igraph"): coercing to igraph objects

**coerce** signature(from = "markovchain", to = "matrix"): coercing to matrix objects

**coerce** signature(from = "markovchain", to = "sparseMatrix"): coercing to sparseMatrix objects

**coerce** signature(from = "matrix", to = "markovchain"): coercing to markovchain objects from matrix one

**dim** signature(x = "markovchain"): method to get the size

**names** signature(x = "markovchain"): method to get the names of states

**names<-** signature(x = "markovchain", value = "character"): method to set the names of states

**initialize** signature(.Object = "markovchain"): initialize method

**plot** signature(x = "markovchain", y = "missing"): plot method for markovchain objects

**predict** signature(object = "markovchain"): predict method

**print** signature(x = "markovchain"): print method.

**show** signature(object = "markovchain"): show method.

**sort** signature(x = "markovchain", decreasing = FALSE): sorting the transition matrix.

**states** signature(object = "markovchain"): returns the names of states (as names.
steadyStates signature(object = "markovchain"): method to get the steady vector.

summary signature(object = "markovchain"): method to summarize structure of the markov chain

transientStates signature(object = "markovchain"): method to get the transient states.

t signature(x = "markovchain"): method to get the transient states.

transitionProbability signature(object = "markovchain"): transition probability

Note

1. markovchain object are backed by S4 Classes.

2. Validation method is used to assess whether either columns or rows totals to one. Rounding is used up to .Machine$double.eps * 100. If state names are not properly defined for a probability matrix, coercing to markovchain object leads to overriding states name with artificial "s1", "s2", ... sequence. In addition, operator overloading has been applied for +, *, ==, != operators.

Author(s)

Giorgio Spedicato

References

A First Course in Probability (8th Edition), Sheldon Ross, Prentice Hall 2010

See Also

markovchainSequence, markovchainFit

Examples

#show markovchain definition
showClass("markovchain")
#create a simple Markov chain
transMatr<-matrix(c(0.4,0.6,.3,.7),nrow=2,byrow=TRUE)
simpleMc<-new("markovchain", states=c("a","b"),
    transitionMatrix=transMatr,
    name="simpleMc")

#power
simpleMc^4
#some methods
steadyStates(simpleMc)
asorbingStates(simpleMc)
simpleMc[2,1]
t(simpleMc)
is.irreducible(simpleMc)
#conditional distributions
conditionalDistribution(simpleMc, "b")
#example for predict method
sequence<-c("a", "b", "a", "a", "a", "b", "a", "b", "a", "b", "a", "a", "b", "b", "b", "b", "a")
mcFit<-markovchainFit(data=sequence)
predict(mcFit$estimate, newdata="b",n.ahead=3)
#direct conversion
myMc<-as(transMatr, "markovchain")

#example of summary
summary(simpleMc)
## Not run: plot(simpleMc)

markovchainList-class  Non homogeneous discrete time Markov Chains class

Description
A class to handle non homogeneous discrete Markov chains

Arguments
markovchains Object of class "list": a list of markovchains
name Object of class "character": optional name of the class

Objects from the Class
A markovchainlist is a list of markovchain objects. They can be used to model non homogeneous discrete time Markov Chains, when transition probabilities (and possible states) change by time.

Methods
[[ signature(x = "markovchainList"): extract the i-th markovchain
dim signature(x = "markovchainList"): number of markovchain underlying the matrix
predict signature(object = "markovchainList"): predict from a markovchainList
print signature(x = "markovchainList"): prints the list of markovchains
show signature(object = "markovchainList"): same as print

Note
The class consists in a list of markovchain objects. It is aimed at working with non homogeneous Markov chains.

Author(s)
Giorgio Spedicato

References
A First Course in Probability (8th Edition), Sheldon Ross, Prentice Hall 2010
See Also

markovchain

Examples

```r
showClass("markovchainList")
#define a markovchainList
statesNames=c("a","b")

mcA<-new("markovchain",name="MCA",
  transitionMatrix=matrix(c(0.7,0.3,0.1,0.9),
    byrow=TRUE, nrow=2,
    dimnames=list(statesNames,statesNames))
)

mcB<-new("markovchain", states=c("a","b","c"), name="MCB",
  transitionMatrix=matrix(c(0.2,0.5,0.3,0,1,0,0.1,0.8,0.1),
    nrow=3, byrow=TRUE))

mcC<-new("markovchain", states=c("a","b","c","d"), name="MCC",
  transitionMatrix=matrix(c(0.25,0.75,0,0,0.4,0.6,
    0,0,0,0.1,0.9,0,0,0.7,0.3),
    nrow=4, byrow=TRUE))

mcList<-new("markovchainList",markovchains=list(mcA, mcB, mcC),
  name="Non - homogeneous Markov Chain")
```

markovchainListFit  markovchainListFit

Description

Given a data frame or a matrix (rows are observations, by cols the temporal sequence), it fits a non-homogeneous discrete time markov chain process (storing row). In particular a markovchainList of size = ncol - 1 is obtained estimating transitions from the n samples given by consecutive column pairs.

Usage

markovchainListFit(data, byrow = TRUE, laplacian = 0, name)

Arguments

data  Either a matrix or a data.frame or a list object.
byrow Indicates whether distinc stochastic processes trajectories are shown in distinct rows.
laplacian Laplacian correction (default 0).
name Optional name.
**markovchainSequence**

**Details**

If data contains NAs then the transitions containing NA will be ignored.

**Value**

A list containing two slots: estimate (the estimate) name

**Examples**

```r
# using holson dataset
data(holson)
# fitting a single markovchain
singleMc <- markovchainFit(data = holson[,2:12])
# fitting a markovchainList
mclistFit <- markovchainListFit(data = holson[, 2:12], name = "holsonMcList")
```

---

**Description**

Provided any markovchain object, it returns a sequence of states coming from the underlying stationary distribution.

**Usage**

```r
markovchainSequence(
  n,
  markovchain,
  t0 = sample(markovchain@states, 1),
  include.t0 = FALSE,
  useRCpp = TRUE
)
```

**Arguments**

- `n` Sample size
- `markovchain` markovchain object
- `t0` The initial state
- `include.t0` Specify if the initial state shall be used
- `useRCpp` Boolean. Should RCpp fast implementation being used? Default is yes.

**Details**

A sequence of size n is sampled.
meanAbsorptionTime

Value

A Character Vector

Author(s)

Giorgio Spedicato

References

A First Course in Probability (8th Edition), Sheldon Ross, Prentice Hall 2010

See Also

markovchainFit

Examples

# define the markovchain object
statesNames <- c("a", "b", "c")
mcB <- new("markovchain", states = statesNames,
          transitionMatrix = matrix(c(0.2, 0.5, 0.3, 0, 0.2, 0.8, 0.1, 0.8, 0.1),
                                   nrow = 3, byrow = TRUE, dimnames = list(statesNames, statesNames)))

# show the sequence
outs <- markovchainSequence(n = 100, markovchain = mcB, t0 = "a")

meanAbsorptionTime

Mean absorption time

Description

Computes the expected number of steps to go from any of the transient states to any of the recurrent states. The Markov chain should have at least one transient state for this method to work

Usage

meanAbsorptionTime(object)

Arguments

object the markovchain object

Value

A named vector with the expected number of steps to go from a transient state to any of the recurrent ones
**meanFirstPassageTime**

**Author(s)**

Ignacio Cordón

**References**


**Examples**

```r
m <- matrix(c(1/2, 1/2, 0,
              1/2, 1/2, 0,
              0, 1/2, 1/2), ncol = 3, byrow = TRUE)
m <- new("markovchain", states = letters[1:3], transitionMatrix = m)
times <- meanAbsorptionTime(m)
```

---

**meanFirstPassageTime**  
*Mean First Passage Time for irreducible Markov chains*

**Description**

Given an irreducible (ergodic) markovchain object, this function calculates the expected number of steps to reach other states

**Usage**

```r
meanFirstPassageTime(object, destination)
```

**Arguments**

- `object`  
  the markovchain object
- `destination`  
  a character vector representing the states respect to which we want to compute the mean first passage time. Empty by default

**Details**

For an ergodic Markov chain it computes:

- If destination is empty, the average first time (in steps) that takes the Markov chain to go from initial state i to j. (i, j) represents that value in case the Markov chain is given row-wise, (j, i) in case it is given col-wise.
- If destination is not empty, the average time it takes us from the remaining states to reach the states in destination

**Value**

- a Matrix of the same size with the average first passage times if destination is empty, a vector if destination is not
meanNumVisits

Author(s)
Toni Giorgino, Ignacio Cordón

References

Examples
```r
m <- matrix(1 / 10 * c(6,3,1,
  2,3,5,
  4,1,5), ncol = 3, byrow = TRUE)
mc <- new("markovchain", states = c("s","c","r"), transitionMatrix = m)
meanFirstPassageTime(mc, "r")

# Grinstead and Snell's "Oz weather" worked out example
mOz <- matrix(c(2,1,1,
  2,0,2,
  1,1,2)/4, ncol = 3, byrow = TRUE)
mcOz <- new("markovchain", states = c("s","c","r"), transitionMatrix = mOz)
meanFirstPassageTime(mcOz)
```

---

meanNumVisits

Mean num of visits for markovchain, starting at each state

Description
Given a markovchain object, this function calculates a matrix where the element (i, j) represents the expect number of visits to the state j if the chain starts at i (in a Markov chain by columns it would be the element (j, i) instead)

Usage
```r
meanNumVisits(object)
```

Arguments
- object: the markovchain-class object

Value
- a matrix with the expect number of visits to each state

Author(s)
Ignacio Cordón
**meanRecurrenceTime**

**References**

R. Vélez, T. Prieto, Procesos Estocásticos, Librería UNED, 2013

**Examples**

```r
M <- matlab::zeros(5, 5)
M[1,1] <- M[5,5] <- 1

mc <- new("markovchain", transitionMatrix = M)
meanNumVisits(mc)
```

---

<table>
<thead>
<tr>
<th>meanRecurrenceTime</th>
<th>Mean recurrence time</th>
</tr>
</thead>
</table>

**Description**

Computes the expected time to return to a recurrent state in case the Markov chain starts there.

**Usage**

```r
meanRecurrenceTime(object)
```

**Arguments**

- `object` the markovchain object

**Value**

For a Markov chain it outputs a named vector with the expected time to first return to a state when the chain starts there. States present in the vector are only the recurrent ones. If the matrix is ergodic (i.e. irreducible), then all states are present in the output and order is the same as states order for the Markov chain.

**Author(s)**

Ignacio Cordón

**References**

Examples

```r
m <- matrix(1 / 10 * c(6, 3, 1,
                        2, 3, 5,
                        4, 1, 5), ncol = 3, byrow = TRUE)
mc <- new("markovchain", states = c("s","c","r"), transitionMatrix = m)
meanRecurrenceTime(mc)
```

---

**multinomialConfidenceIntervals**

_A function to compute multinomial confidence intervals of DTMC_

---

Description

Return estimated transition matrix assuming a Multinomial Distribution

Usage

```r
multinomialConfidenceIntervals(
  transitionMatrix,
  countsTransitionMatrix,
  confidencelevel = 0.95
)
```

Arguments

- `transitionMatrix`  
  An estimated transition matrix.
- `countsTransitionMatrix`  
  Empirical (conts) transition matrix, on which the `transitionMatrix` was performed.
- `confidencelevel`  
  confidence interval level.

Value

Two matrices containing the confidence intervals.

References

Constructing two-sided simultaneous confidence intervals for multinomial proportions for small counts in a large number of cells. Journal of Statistical Software 5(6) (2000)

See Also

- `markovchainFit`
name

Examples

seq <- c("a", "b", "a", "a", "a", "b", "a", "b", "a", "b", "a", "b", "b", "b", "a")
mcfit <- markovchainFit(data = seq, byrow = TRUE)
seqmat <- createSequenceMatrix(seq)
multipomialConfidenceIntervals(mcfit$estimate@transitionMatrix, seqmat, 0.95)

name

Method to retrieve name of markovchain object

Description

This method returns the name of a markovchain object

Usage

name(object)

## S4 method for signature 'markovchain'
name(object)

Arguments

object A markovchain object

Author(s)

Giorgio Spedicato, Deepak Yadav

Examples

statesNames <- c("a", "b", "c")
markovB <- new("markovchain", states = statesNames, transitionMatrix =
  matrix(c(0.2, 0.5, 0.3, 0, 1, 0, 0.1, 0.8, 0.1), nrow = 3,
  byrow = TRUE, dimnames=list(statesNames,statesNames)),
  name = "A markovchain Object"
)
name(markovB)
**Method to set name of markovchain object**

**Description**
This method modifies the existing name of markovchain object

**Usage**

```
name(object) <- value
```

```
## S4 replacement method for signature 'markovchain'
name(object) <- value
```

**Arguments**

- `object`: A markovchain object
- `value`: New name of markovchain object

**Author(s)**
Giorgio Spedicato, Deepak Yadav

**Examples**

```r
statesNames <- c("a", "b", "c")
markovB <- new("markovchain", states = statesNames, transitionMatrix =
  matrix(c(0.2, 0.5, 0.3, 0, 1, 0, 0.1, 0.8, 0.1), nrow = 3,
  byrow = TRUE, dimnames=list(statesNames,statesNames)),
  name = "A markovchain Object"
)
name(markovB) <- "dangerous mc"
```

**Returns the states for a Markov chain object**

**Description**
Returns the states for a Markov chain object

**Usage**

```
## S4 method for signature 'markovchain'
names(x)
```
Arguments

x object we want to return states for

\[ \text{noofVisitsDist} \quad \text{return a joint pdf of the number of visits to the various states of the DTMC} \]

Description

This function would return a joint pdf of the number of visits to the various states of the DTMC during the first \( N \) steps.

Usage

\[ \text{noofVisitsDist}(\text{markovchain}, N, \text{state}) \]

Arguments

markovchain a markovchain-class object
N no of steps
state the initial state

Details

This function would return a joint pdf of the number of visits to the various states of the DTMC during the first \( N \) steps.

Value

a numeric vector depicting the above described probability density function.

Author(s)

Vandit Jain

Examples

transMatr<-matrix(c(0.4,0.6,.3,.7),nrow=2,byrow=TRUE)
simpleMc<-new("markovchain", states=c("a","b"),
transitionMatrix=transMatr,
name="simpleMc")
noofVisitsDist(simpleMc,5,"a")
Various functions to perform structural analysis of DTMC

Description

These functions return absorbing and transient states of the markovchain objects.

Usage

```r
period(object)
communicatingClasses(object)
recurrentClasses(object)
transientClasses(object)
transientStates(object)
recurrentStates(object)
absorbingStates(object)
canonicForm(object)
```

Arguments

- `object` A markovchain object.

Value

- `period` returns an integer number corresponding to the periodicity of the Markov chain (if it is irreducible).
- `absorbingStates` returns a character vector with the names of the absorbing states in the Markov chain.
- `communicatingClasses` returns a list in which each slot contains the names of the states that are in that communicating class.
- `recurrentClasses` analogously to `communicatingClasses`, but with recurrent classes.
- `transientClasses` analogously to `communicatingClasses`, but with transient classes.
- `transientStates` returns a character vector with all the transient states for the Markov chain.
- `recurrentStates` returns a character vector with all the recurrent states for the Markov chain.
- `canonicForm` returns the Markov chain reordered by a permutation of states so that we have blocks submatrices for each of the recurrent classes and a collection of rows in the end for the transient states.
Author(s)

Giorgio Alfredo Spedicato, Ignacio Cordón

References

Feres, Matlab listing for markov chain.

See Also

markovchain

Examples

statesNames <- c("a", "b", "c")
mc <- new("markovchain", states = statesNames, transitionMatrix =
    matrix(c(0.2, 0.5, 0.3,
             0, 1, 0, 
             0.1, 0.8, 0.1), nrow = 3, byrow = TRUE,
             dimnames = list(statesNames, statesNames))
)

communicatingClasses(mc)
 recurrentClasses(mc)
 recurrentClasses(mc)
 absorbingStates(mc)
 transientStates(mc)
 canonicForm(mc)

# periodicity analysis
A <- matrix(c(0, 1, 0, 0.5, 0, 0.5, 0, 0.5, 0, 0, 1, 0),
            nrow = 4, ncol = 4, byrow = TRUE)
mcA <- new("markovchain", states = c("a", "b", "c", "d"),
            transitionMatrix = A,
            name = "A")

is.irreducible(mcA) #true
period(mcA) #2

# periodicity analysis
B <- matrix(c(0, 0, 1/2, 1/4, 1/4, 0, 0,
             0, 0, 1/3, 0, 2/3, 0, 0,
             0, 0, 0, 0, 1/3, 2/3,
             0, 0, 0, 0, 1/2, 1/2,
             0, 0, 0, 0, 3/4, 1/4,
             1/2, 1/2, 0, 0, 0, 0,
             1/4, 3/4, 0, 0, 0, 0), byrow = TRUE, ncol = 7)
mcB <- new("markovchain", transitionMatrix = B)
period(mcB)
predictHommc

Simulate a higher order multivariate markovchain

Description

This function provides a prediction of states for a higher order multivariate markovchain object.

Usage

predictHommc(hommc, t, init)

Arguments

- hommc: a hommc-class object
- t: no of iterations to predict
- init: matrix of previous states size of which depends on hommc

Details

The user is required to provide a matrix of giving n previous corresponding every categorical sequence. Dimensions of the init are s X n, where s is number of categorical sequences and n is order of the homc.

Value

The function returns a matrix of size s X t displaying t predicted states in each row corresponding to every categorical sequence.

Author(s)

Vandit Jain

predictiveDistribution

Description

The function computes the probability of observing a new data set, given a data set.

Usage

predictiveDistribution(stringchar, newData, hyperparam = matrix())
Arguments

stringchar This is the data using which the Bayesian inference is performed.

newData This is the data whose predictive probability is computed.

hyperparam This determines the shape of the prior distribution of the parameters. If none is provided, default value of 1 is assigned to each parameter. This must be of size kxk where k is the number of states in the chain and the values should typically be non-negative integers.

Details

The underlying method is Bayesian inference. The probability is computed by averaging the likelihood of the new data with respect to the posterior. Since the method assumes conjugate priors, the result can be represented in a closed form (see the vignette for more details), which is what is returned.

Value

The log of the probability is returned.

Author(s)

Sai Bhargav Yalamanchi

References

Inferring Markov Chains: Bayesian Estimation, Model Comparison, Entropy Rate, and Out-of-Class Modeling, Christopher C. Strelioff, James P. Crutchfield, Alfred Hubler, Santa Fe Institute

Yalamanchi SB, Spedicato GA (2015). Bayesian Inference of First Order Markov Chains. R package version 0.2.5

See Also

markovchainFit

Examples

```r
sequence<- c("a", "b", "a", "a", "a", "b", "a", "b", "a", "a", "a", "b", "b", "b", "a")
hyperMatrix<-matrix(c(1, 2, 1, 4), nrow = 2,dimnames=list(c("a","b"),c("a","b")))
predProb <- predictiveDistribution(sequence[1:10], sequence[11:17], hyperparam =hyperMatrix )
hyperMatrix2<-hyperMatrix[c(2,1),c(2,1)]
predProb2 <- predictiveDistribution(sequence[1:10], sequence[11:17], hyperparam =hyperMatrix2 )
predProb2==predProb
```
**preproglucacon**  
*Preproglucacon DNA protein bases sequences*

**Description**

Sequence of bases for preproglucacon DNA protein

**Usage**

data(preproglucacon)

**Format**

A data frame with 1572 observations on the following 2 variables.

<table>
<thead>
<tr>
<th>V1</th>
<th>V2</th>
</tr>
</thead>
<tbody>
<tr>
<td>a numeric vector, showing original coding</td>
<td>a character vector, showing initial of DNA bases (Adenine, Cytosine, Guanine, Thymine)</td>
</tr>
</tbody>
</table>

**Source**

Avery Henderson

**References**

Avery Henderson, Fitting markov chain models on discrete time series such as DNA sequences

**Examples**

data(preproglucacon)  
preproglucaconMc<-markovchainFit(data=preproglucacon$preproglucacon)

---

**priorDistribution**

**Description**

Function to evaluate the prior probability of a transition matrix. It is based on conjugate priors and therefore a Dirichlet distribution is used to model the transitions of each state.

**Usage**

priorDistribution(transMatr, hyperparam = matrix())
priorDistribution

Arguments

- `transMatr` The transition matrix whose probability is the parameter of interest.
- `hyperparam` The hyperparam matrix (optional). If not provided, a default value of 1 is assumed for each and therefore the resulting probability distribution is uniform.

Details

The states (dimnames) of the transition matrix and the hyperparam may be in any order.

Value

The log of the probabilities for each state is returned in a numeric vector. Each number in the vector represents the probability (log) of having a probability transition vector as specified in corresponding the row of the transition matrix.

Note

This function can be used in conjunction with inferHyperparam. For example, if the user has a prior data set and a prior transition matrix, he can infer the hyperparameters using inferHyperparam and then compute the probability of their prior matrix using the inferred hyperparameters with priorDistribution.

Author(s)

Sai Bhargav Yalamanchi, Giorgio Spedicato

References

Yalamanchi SB, Spedicato GA (2015). Bayesian Inference of First Order Markov Chains. R package version 0.2.5

See Also

`predictiveDistribution`, `inferHyperparam`

Examples

```r
priorDistribution(matrix(c(0.5, 0.5, 0.5, 0.5),
nrow = 2,
dimnames = list(c("a", "b"), c("a", "b"))),
matrix(c(2, 2, 2, 2),
nrow = 2,
dimnames = list(c("a", "b"), c("a", "b")))
```
Probability at T

Calculating probability from a ctmc object

Description

This function returns the probability of every state at time t under different conditions.

Usage

`probabilityatT(C, t, x0, useRCpp)`

Arguments

- `C`: A CTMC S4 object
- `t`: final time t
- `x0`: initial state
- `useRCpp`: logical whether to use RCpp implementation

Details

The initial state is not mandatory. In case it is not provided, function returns a matrix of transition function at time t else it returns vector of probabilities of transition to different states if initial state was `x0`.

Value

returns a vector or a matrix in case `x0` is provided or not respectively.

Author(s)

Vandit Jain

References

INTRODUCTION TO STOCHASTIC PROCESSES WITH R, ROBERT P. DOBROW, Wiley

Examples

```r
states <- c(“a”, “b”, “c”, “d”)
byRow <- TRUE
gen <- matrix(data = c(-1, 1/2, 1/2, 0, 1/4, -1/2, 0, 1/4, 1/6, 0, -1/3, 1/6, 0, 0, 0, 0),
nrow = 4, byrow = byRow, dimnames = list(states, states))
ctmc <- new(“ctmc”, states = states, byrow = byRow, generator = gen, name = “testctmc”)
probabilityatT(ctmc, 1, useRCpp = TRUE)
```
Description

Rainfall measured in Alofi Island

Usage

data(rain)

Format

A data frame with 1096 observations on the following 2 variables.

V1 a numeric vector, showing original coding
rain a character vector, showing daily rainfall millilitres brackets

Source

Avery Henderson

References

Avery Henderson, Fitting markov chain models on discrete time series such as DNA sequences

Examples

data(rain)
rainMc<-markovchainFit(data=rain$rain)

Description

The function generates random CTMC transitions as per the provided generator matrix.

Usage

rctmc(n, ctmc, initDist = numeric(), T = 0, include.T0 = TRUE, out.type = "list")
Arguments

- `n`: The number of samples to generate.
- `ctmc`: The CTMC S4 object.
- `initDist`: The initial distribution of states.
- `T`: The time up to which the simulation runs (all transitions after time T are not returned).
- `include.T0`: Flag to determine if start state is to be included.
- `out.type`: "list" or "df"

Details

In order to use the T0 argument, set n to Inf.

Value

Based on out.type, a list or a data frame is returned. The returned list has two elements - a character vector (states) and a numeric vector (indicating time of transitions). The data frame is similarly structured.

Author(s)

Sai Bhargav Yalamanchi

References

Introduction to Stochastic Processes with Applications in the Biosciences (2013), David F. Anderson, University of Wisconsin at Madison

See Also

generatorToTransitionMatrix, ctmc-class

Examples

```r
energyStates <- c("sigma", "sigma_star")
byRow <- TRUE
gen <- matrix(data = c(-3, 3, 1, -1), nrow = 2,
byrow = byRow, dimnames = list(energyStates, energyStates))
molecularCTMC <- new("ctmc", states = energyStates,
byrow = byRow, generator = gen,
name = "Molecular Transition Model")

statesDist <- c(0.8, 0.2)
rctmc(n = Inf, ctmc = molecularCTMC, T = 1)
rctmc(n = 5, ctmc = molecularCTMC, initDist = statesDist, include.T0 = FALSE)
```
Function to generate a sequence of states from homogeneous or non-homogeneous Markov chains.

Description

Provided any markovchain or markovchainList objects, it returns a sequence of states coming from the underlying stationary distribution.

Usage

```r
rmarkovchain(
  n,
  object,
  what = "data.frame",
  useRCpp = TRUE,
  parallel = FALSE,
  num.cores = NULL,
  ...
)
```

Arguments

- `n` Sample size
- `object` Either a markovchain or a markovchainList object
- `what` It specifies whether either a data.frame or a matrix (each rows represent a simulation) or a list is returned.
- `useRCpp` Boolean. Should RCpp fast implementation being used? Default is yes.
- `parallel` Boolean. Should parallel implementation being used? Default is yes.
- `num.cores` Number of Cores to be used
- `...` additional parameters passed to the internal sampler

Details

When a homogeneous process is assumed (markovchain object) a sequence is sampled of size `n`. When a non-homogeneous process is assumed, `n` samples are taken but the process is assumed to last from the begin to the end of the non-homogeneous markov process.

Value

Character Vector, data.frame, list or matrix

Note

Check the type of input
Sales Demand Sequences

Description
Sales demand sequences of five products (A, B, C, D, E). Each row corresponds to a sequence. First row corresponds to Sequence A, Second row to Sequence B and so on.
Usage

data("sales")

Format

An object of class matrix with 269 rows and 5 columns.

Details

The example can be used to fit High order multivariate markov chain.

Examples

data("sales")
# fitHighOrderMultivarMC(seqMat = sales, order = 2, Norm = 2)

show.hommc-method

Function to display the details of hommc object

Description

This is a convenience function to display the slots of hommc object in proper format

Usage

## S4 method for signature 'hommc'
show(object)

Arguments

object An object of class hommc

states Defined states of a transition matrix

Description

This method returns the states of a transition matrix.

Usage

states(object)

## S4 method for signature 'markovchain'
states(object)
**steadyStates**

**Arguments**

object  
A discrete markovchain object

**Value**

The character vector corresponding to states slot.

**Author(s)**

Giorgio Spedicato

**References**

A First Course in Probability (8th Edition), Sheldon Ross, Prentice Hall 2010

**See Also**

markovchain

**Examples**

```r
statesNames <- c("a", "b", "c")
markovB <- new("markovchain", states = statesNames, transitionMatrix = matrix(c(0.2, 0.5, 0.3, 0, 1, 0, 0.1, 0.8, 0.1), nrow = 3, byrow = TRUE, dimnames=list(statesNames,statesNames)),
    name = "A markovchain Object"
)

states(markovB)
names(markovB)
```

---

**steadyStates**

Stationary states of a markovchain object

**Description**

This method returns the stationary vector in matricial form of a markovchain object.

**Usage**

steadyStates(object)

**Arguments**

object  
A discrete markovchain object

**Value**

A matrix corresponding to the stationary states
Note

The steady states are identified starting from which eigenvectors correspond to identity eigenvalues and then normalizing them to sum up to unity. When negative values are found in the matrix, the eigenvalues extraction is performed on the recurrent classes submatrix.

Author(s)

Giorgio Spedicato

References

A First Course in Probability (8th Edition), Sheldon Ross, Prentice Hall 2010

See Also

markovchain

Examples

```r
statesNames <- c("a", "b", "c")
markovB <- new("markovchain", states = statesNames, transitionMatrix =
  matrix(c(0.2, 0.5, 0.3, 0, 1, 0, 0.1, 0.8, 0.1), nrow = 3,
        byrow = TRUE, dimnames=list(statesNames,statesNames)),
        name = "A markovchain Object"
)
steadyStates(markovB)
```

---

tm_abs  

Single Year Corporate Credit Rating Transitions

Description

Matrix of Standard and Poor’s Global Corporate Rating Transition Frequencies 2000 (NR Removed)

Usage

data(tm_abs)

Format

The format is: num [1:8, 1:8] 17 2 0 0 0 0 1 455 ... - attr(*, "dimnames")=List of 2 ..$ : chr [1:8] "AAA" "AA" "A" "BBB" ... ..$ : chr [1:8] "AAA" "AA" "A" "BBB" ...

References

transition2Generator

Return the generator matrix for a corresponding transition matrix

Examples

data(tm_abs)

```
mymatr <- matrix(c(.4, .6, .1, .9), nrow = 2, byrow = TRUE)
Q <- transition2Generator(P = mymatr)
expm::expm(Q)
```
transitionProbability  Function to get the transition probabilities from initial to subsequent states.

Description
This is a convenience function to get transition probabilities.

Usage
transitionProbability(object, t0, t1)

## S4 method for signature 'markovchain'
transitionProbability(object, t0, t1)

Arguments

object  A markovchain object.
t0      Initial state.
t1      Subsequent state.

Value

Numeric Vector

Author(s)
Giorgio Spedicato

References
A First Course in Probability (8th Edition), Sheldon Ross, Prentice Hall 2010

See Also
markovchain

Examples

```
statesNames <- c("a", "b", "c")
markovB <- new("markovchain", states = statesNames, transitionMatrix =
    matrix(c(0.2, 0.5, 0.3, 0, 1, 0, 0.1, 0.8, 0.1), nrow = 3,
    byrow = TRUE, dimnames=list(statesNames,statesNames)),
    name = "A markovchain Object")
transitionProbability(markovB,"b", "c")
```
verifyMarkovProperty

Various functions to perform statistical inference of DTMC

Description

These functions verify the Markov property, assess the order and stationarity of the Markov chain. This function tests whether an empirical transition matrix is statistically compatible with a theoretical one. It is a chi-square based test.

Verifies that the s elements in the input list belongs to the same DTMC.

Usage

verifyMarkovProperty(sequence, verbose = TRUE)

assessOrder(sequence, verbose = TRUE)

assessStationarity(sequence, nblocs, verbose = TRUE)

verifyEmpiricalToTheoretical(data, object, verbose = TRUE)

verifyHomogeneity(inputList, verbose = TRUE)

Arguments

sequence: An empirical sequence.
verbose: Should test results be printed out?
nblocks: Number of blocks.
data: matrix, character or list to be converted in a raw transition matrix
object: a markovchain object
inputList: A list of items that can coerced to transition matrices

Value

Verification result

a list with following slots: statistic (the chi - square statistic), dof (degrees of freedom), and corresponding p-value

a list of transition matrices?

Author(s)

Tae Seung Kang, Giorgio Alfredo Spedicato

References

Anderson and Goodman.
See Also

markovchain

Examples

```r
sequence <- c("a", "b", "a", "a", "a", "b", "a", "b", "a", "b", "a")
mcFit <- markovchainFit(data = sequence, byrow = FALSE)
verifyMarkovProperty(sequence)
assessOrder(sequence)
assessStationarity(sequence, 1)
```

#Example taken from Kullback Kupperman Tests for Contingency Tables and Markov Chains

```r
sequence<-c(0,1,2,2,1,0,0,0,0,0,1,2,2,1,0,1,0,0,0,0,0,1,1,1,1,1,1,0,0,0,2,1,0,
2,1,0,0,0,0,0,1,1,1,2,2,2,0,0,2,1,1,1,2,1,1,1,1,1,1,1,1,1,0,2,
0,1,1,0,0,1,2,2,0,0,0,0,0,2,2,2,1,1,1,0,1,1,1,1,0,0,2,1,1,
0,0,0,0,0,2,2,1,1,1,1,1,2,1,2,0,0,0,1,2,2,2,0,0,0,1,1)
mc=matrix(c(5/8,1/4,1/8,1/4,1/2,1/4,1/4,3/8,3/8),byrow=TRUE, nrow=3)
rownames(mc)<-colnames(mc)<-0:2; theoreticalMc<-as(mc, "markovchain")
verifyEmpiricalToTheoretical(data=sequence,object=theoreticalMc)
```

```r
data(kullback)
verifyHomogeneity(inputList=kullback,verbose=TRUE)
```

```r
data(kullback)
verifyHomogeneity(inputList=kullback,verbose=TRUE)
```
Index

!=, markovchain, markovchain-method (markovchain-class), 32

* Topic classes
  ctmc-class, 11
  markovchain-class, 32
  markovchainList-class, 35

* Topic datasets
  bladen, 5
  craigsendi, 8
  holson, 23
  kullback, 32
  preproglucacoa, 50
  rain, 53
  sales, 56
  tm_abs, 59

* Topic package
  markovchain-package, 3
  *, markovchain, markovchain-method (markovchain-class), 32
  *, markovchain, matrix-method (markovchain-class), 32
  *, markovchain, numeric-method (markovchain-class), 32
  *, matrix, markovchain-method (markovchain-class), 32
  *, numeric, markovchain-method (markovchain-class), 32
  ==, markovchain, markovchain-method (markovchain-class), 32
  [], markovchain, ANY, ANY, ANY-method (markovchain-class), 32
  [[], markovchainList-method (markovchainList-class), 35
  ^, markovchain, numeric-method (markovchain-class), 32

  absorbingStates (period), 46
  absorbingStates, markovchain-method (markovchain-class), 32
  absorptionProbabilities, markovchain-method (markovchain-class), 32
  assessOrder (verifyMarkovProperty), 62
  assessStationarity (verifyMarkovProperty), 62
  bladen, 5
  canonicForm (period), 46
  canonicForm, markovchain-method (markovchain-class), 32
  coerce, data.frame, markovchain-method (markovchain-class), 32
  coerce, etm, markovchain-method (markovchain-class), 32
  coerce, markovchain, data.frame-method (markovchain-class), 32
  coerce, markovchain, igraph-method (markovchain-class), 32
  coerce, markovchain, matrix-method (markovchain-class), 32
  coerce, markovchain, sparseMatrix-method (markovchain-class), 32
  coerce, matrix, markovchain-method (markovchain-class), 32
  coerce, msm, markovchain-method (markovchain-class), 32
  coerce, msm.est, markovchain-method (markovchain-class), 32
  committorAB, 6
  communicatingClasses (period), 46
  communicatingClasses, markovchain-method (markovchain-class), 32
  conditionalDistribution, 7, 16
  conditionalDistribution, markovchain-method (markovchain-class), 32
Index

print, markovchain-method (markovchain-class), 32
print, markovchainList-method (markovchainList-class), 35
priorDistribution, 50
probabilityatT, 52
rain, 53
rctmc, 11, 13, 21, 53, 60
recurrentClasses (period), 46
recurrentClasses, markovchain-method (markovchain-class), 32
recurrentStates (period), 46
recurrentStates, markovchain-method (markovchain-class), 32
rmarkovchain, 55
sales, 56
seq2freqProb (fitHigherOrder), 18
seq2matHigh (fitHigherOrder), 18
show, hommc-method, 57
show, markovchain-method (markovchain-class), 32
show, markovchainList-method (markovchainList-class), 35
sort, markovchain-method (markovchain-class), 32
states, 57
states, ctmc-method (ctmc-class), 11
states, markovchain-method (states), 57
steadyStates, 58
steadyStates, ctmc-method (ctmc-class), 11
steadyStates, markovchain-method (markovchain-class), 32
summary, 29
summary, markovchain-method (markovchain-class), 32
t, markovchain-method (markovchain-class), 32
tm_abs, 59
transientClasses (period), 46
transientClasses, markovchain-method (markovchain-class), 32
transientStates (period), 46
transientStates, markovchain-method (markovchain-class), 32
transition2Generator, 60
transitionProbability, 61
transitionProbability, markovchain-method (transitionProbability), 61
verifyEmpiricalToTheoretical (verifyMarkovProperty), 62
verifyHomogeneity (verifyMarkovProperty), 62
verifyMarkovProperty, 62