Package ‘markovchain’

July 22, 2019

Type Package

Title Easy Handling Discrete Time Markov Chains

Version 0.6.9.16

Date 2019-07-21

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 (>= 3.5.0), methods

Imports igraph, Matrix, matlab, expm, stats4, parallel, Rcpp (>= 0.12.18), RcppParallel, utils, stats, grDevices

Suggests knitr, testthat, diagram, DiagrammeR, msm, etm, Rsolnp, knitr, markdown, ctmcR, MmgraphR, bookdown, rtables

VignetteBuilder utils, knitr

LinkingTo Rcpp, RcppParallel, RcppArmadillo

SystemRequirements GNU make

LazyLoad yes

ByteCompile yes

Encoding UTF-8

BugReports http://github.com/spedygiorgio/markovchain/issues

URL http://github.com/spedygiorgio/markovchain/

RoxygenNote 6.1.1

NeedsCompilation yes

Author Giorgio Alfredo Spedicato [aut, cre]

(<https://orcid.org/0000-0002-0315-8888>),

Tae Seung Kang [aut],

Sai Bhargav Yalamanchi [aut],
R topics documented:

Mildenberger Thoralf [ctb] (<https://orcid.org/0000-0001-7242-1873>),
Deepak Yadav [aut],
Ignacio Cordón [ctb],
Vandit Jain [ctb],
Toni Giorgino [ctb] (<https://orcid.org/0000-0001-6449-0596>)

Repository CRAN
Date/Publication 2019-07-22 21:20:02 UTC

R topics documented:

markovchain-package ........................................... 2
absorbingStates ............................................. 3
blanden ......................................................... 5
committorAB ................................................. 6
conditionalDistribution ..................................... 7
craigsendi ..................................................... 8
createSequenceMatrix ....................................... 8
ctmc-class .................................................... 11
ctmcFit ......................................................... 12
expectedRewards ............................................. 13
expectedRewardsBeforeHittingA ................................ 14
ExpectedTime ................................................ 15
firstPassage .................................................. 16
firstPassageMultiple ....................................... 17
fitHigherOrder ............................................. 18
fitHighOrderMultivarMC .................................... 19
freq2Generator ............................................... 20
generatorToTransitionMatrix ................................ 21
HigherOrderMarkovChain-class ............................. 22
hittingProbabilities ....................................... 22
holson ......................................................... 23
hommc-class .................................................. 24
ictmc-class .................................................... 24
impreciseProbabilityatT .................................. 25
inferHyperparam ............................................. 26
is.accessible ................................................ 27
is.CTMCirreducible .......................................... 28
is.irreducible ................................................ 29
is.TimeReversible .......................................... 30
kullback ....................................................... 31
markovchain-class .......................................... 31
markovchainList-class .................................... 34
markovchainListFit .......................................... 35
markovchainSequence ....................................... 36
meanFirstPassageTime ..................................... 37
multinomialConfidenceIntervals .......................... 38
name .......................................................... 39
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: 0.6.9.10
Date: 2018-05-30
License: GPL-2
Depends: R (>= 3.4.0), methods, expm, matlab, igraph, Matrix

Author(s)

Giorgio Alfredo Spedicato
Maintainer: Giorgio Alfredo Spedicato <spedicato_giorgio@yahoo.it>
Examples

# 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,
                  0.1,0.8,0.1),nrow=3, byrow=TRUE, dimnames=list(statesNames, statesNames)))

statesNames=c("a","b","c","d")
matrice<-matrix(c(0.25,0.75,0,0,0.4,0.6,0,0,0,0,0.1,0.9,0,0,0.7,0.3), 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)

absorbingStates Various function to perform structural analysis of DTMC

Description

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

Usage

absorbingStates(object)

transientStates(object)

## S4 method for signature 'markovchain'
transientStates(object)
canonicForm(object)
communicatingClasses(object)
recurrentClasses(object)
period(object)

Arguments

object A markovchain object.

Value

vector, matrix or list

Author(s)

Giorgio Alfredo Spedicato

References

Feres, Matlab listing for markov chain.

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))

communicatingClasses(markovB)
recurrentClasses(markovB)
absorbingStates(markovB)
transientStates(markovB)

canonicForm(markovB)

# periodicity analysis : 1
E <- matrix(c(0, 1, 0, 0, 0.5, 0, 0.5, 0, 0.5, 0, 0.5, 0, 0, 1, 0),
            nrow = 4, ncol = 4, byrow = TRUE)
mcE <- new("markovchain", states = c("a", "b", "c", "d"),
           transitionMatrix = E,
           name = "E")
is.irreducible(mcE) #true
period(mcE) #2

# periodicity analysis : 2
myMtr <- 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, 0, 1/2, 1/2, 0,
                  0, 0, 0, 0, 3/4, 1/4,
                  1/2, 1/2, 0, 0, 0, 0), byrow = TRUE, ncol = 7)

myMc <- new("markovchain", transitionMatrix = myMtr)
period(myMc)

---

blanden  Mobility between income quartiles

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
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 rom set A before set B with different initial states

Usage
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
transMatr <- matrix(c(0,0,0,1,0.5, 0.5,0,0,0,0, 0.5,0,0,0,0, 0,0.2,0.4,0,0, 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))
conditionalDistribution

conditionalDistribution of a Markov Chain

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")
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
Function to fit a discrete Markov chain

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.
createSequenceMatrix

declareSequenceMatrix(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  
The 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.

hyperparam  
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. Streloff, 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**

```r
sequence <- c("a", "b", "a", "a", "a", "b", "a", "b", "a", "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
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  

Class "ctmc"

Description

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

Objects from the Class

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

Slots

- **states**: Name of the states. Must be the same of `colnames` and `rownames` of the generator matrix.
- **byrow**: Binary flag.
- **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

Warning

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

Note

ctmc objects are written using S4 Classes.

Author(s)

Sai Bhargav Yalamanchi, Giorgio Spedicato

References

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

ctmcFit(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.
expectedRewards

Author(s)
Sai Bhargav Yalamanchi

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))
tctmcFit(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
**expectedRewardsBeforeHittingA**

**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

Expecting time

## 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

Markov chains, J. R. Norris, Cambridge University Press

## Examples

```
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 computes the first passage probability in states.

**Usage**

```r
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**

```r
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
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
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**

**package version 0.2.5**

**See Also**
markovchain

**Examples**

```r
sequence <- c("a", "a", "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',
                c('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

c <- matrix(c(150,2,1,1,1200,2,1,2,1,175,1,1,1,1,150),nrow = 4,byrow = TRUE)
sample_rel = cbind((c(sample/rowSums(sample)))[1:dim(sample)[1]-1,:],c(rep(0,dim(sample)[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

energyStates <- 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

Class "HigherOrderMarkovChain"

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(markovchain)

Arguments
markovchain 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
M <- matlab::zeros(5, 5)
M[1,1] <- M[5,5] <- 1

markovChain <- new("markovchain", transitionMatrix = M)
hittingProbabilities(markovChain)
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

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 RCpp 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)
inferHyperparam

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 alphabetically, 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

object A markovchain object.
from The name of state "i" (beginning state).
to The name of state "j" (ending state).

Details

It wraps an internal function named .commStatesFinder.

Value

A boolean value.

Author(s)

Giorgio Spedicato
is.CTMCirreducible

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

c EMC 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

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

Description

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

Usage

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)
```

### Description

The function returns checks if provided function is time reversible.

### Usage

```r
is.TimeReversible(ctmc)
```

### Arguments

- **ctmc**: 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

```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.TimeReversible(molecularCTMC)
```
kullback

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

Class "markovchain"

Description

The S4 class that describes markovchain objects.

Objects from the Class

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

Slots

states: Name of the states. Must be the same of colnames and rownames of the transition matrix
byrow: Binary flag.
transitionMatrix: Square transition matrix
name: Optional character name of the Markov chain

Methods

* signature(el = "markovchain", e2 = "markovchain"): multiply two markovchain objects
* signature(el = "markovchain", e2 = "matrix"): markovchain by matrix multiplication
* signature(el = "markovchain", e2 = "numeric"): markovchain by numeric vector multiplication
* signature(el = "matrix", e2 = "markovchain"): matrix by markov chain
Markov chain class

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"): transpose matrix

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

Warning

Validation method is used to assess whether either columns or rows totals to one. Rounding is used up to 5th decimal. 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.

Note

markovchain object are written in S4 Classes.

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)
absorbingStates(simpleMc)
simpleMc[2,1]
t(simpleMc)
is.irreducible(simpleMc)
conditionalDistribution(simpleMc, "b")

#example for predict method
sequence<-c("a", "b", "a", "a", "a", "b", "a", "b", "a", "a", "b", "b", "b", "b", "b")
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

Class "markovchainList"

Description

A class to handle non-homogeneous Markov chains

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.

Slots

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

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 can help to deal 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(0.2,0.5,0.3,0,1,0.1,0.8,0.1), nrow=3, byrow=TRUE))

mcC<-new("markovchain", states=c("a","b","c","d"), name="MCC", transitionMatrix=matrix(c(0.25,0.75,0,0.4,0.6, 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")
```

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

```r
markovchainListFit(data, byrow = TRUE, laplacian = 0, name)
```
**markovchainSequence**

Function to generate a sequence of states from homogeneous Markov chains.

**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.

**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")
```
Details
A sequence of size n is sampled.

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
```r
# 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")
```

---

**meanFirstPassageTime**

*Mean First Passage Time for markovchain*

Description
Given a markovchain object, this function calculates the expected steps to go from state i to j

Usage
```r
meanFirstPassageTime(markovchain, destination_set)
```

Arguments
- `markovchain` the markovchain-class object
- `destination_set` the set of destination states or NULL (all states)
multinomialConfidenceIntervals

Details

if destination_set is one or more states, the mean first passage time from each remaining state to the
given set is computed. If NULL, the full MFPT matrix is computed (with a different algorithm).

Value

a vector (if destination_set given) or a matrix (otherwise) of mean passage times

Author(s)

Toni Giorgino

References


Examples

Pmat <- matrix(c(6,3,1, 2,3,5, 4,1,5)/10, ncol=3, byrow=TRUE)
P <- new("markovchain", states=c("s","c","r"), transitionMatrix=Pmat)
meanFirstPassageTime(P, "r")
meanFirstPassageTime(P)

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

multinomialConfidenceIntervals

A function to compute multinomial confidence intervals of DTMC

Description

Return estimated transition matrix assuming a Multinomial Distribution

Usage

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
markovchain

Examples

```r
seq<-c("a", "b", "a", "a", "a", "b", "a", "b", "a", "a", "b", "b", "b", "b", "a")
mcfit<-markovchainFit(data=seq,byrow=TRUE)
seqmat<-createSequenceMatrix(seq)
multinomialConfidenceIntervals(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

```r
name(object)
```

### S4 method for signature 'markovchain'

```r
name(object)
```

Arguments

- **object**: A markovchain object

Author(s)

Giorgio Spedicato, Deepak Yadav
name<-  

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)  
```  

name<-  

Method to set name of markovchain object  

Description  

This method modifies the existing name of markovchain object  

Usage  

```r  
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"
```
names,markovchain-method

*Returns the states for a Markov chain object*

**Description**

Returns the states for a Markov chain object

**Usage**

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

**Arguments**

- `x`: object we want to return states for

**noofVisitsDist**  
*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**

```r
noofVisitsDist(markovchain,N,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.
**predictHommc**

**Author(s)**

Vandit Jain

**Examples**

```r
transMatr <- matrix(c(0.4, 0.6, 0.3, 0.7), nrow = 2, byrow = TRUE)
simpleMc <- new("markovchain", states = c("a", "b"),
               transitionMatrix = transMatr,
               name = "simpleMc")
nofVisitsDist(simpleMc, 5, "a")
```

**predictHommc**

*Simulate a higher order multivariate markovchain*

**Description**

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

**Usage**

```r
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

Function to compute the probability of observing a new data set, given a data set

Description

The function computes the probability of observing a new data set using information from a given data set. Additionally, hyperparameters can be provided.

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 k x k 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", "b", "a", "b", "a", "b", "a", "b", "a", "a", "b", "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

```r
data(preproglucacon)
```

Format

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

- **V1**  a numeric vector, showing original coding
- **preproglucacon**  a character vector, showing initial of DNA bases (Adenine, Cytosine, Guanine, Thymine)

Source

Avery Henderson

References

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

Examples

```r
data(preproglucacon)
preproglucaconMc <- markovchainFit(data = preproglucacon$preproglucacon)
```
priorDistribution  

Prior Distribution

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())

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"))))
```

Description

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

Usage

```r
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))
tcmc <- new("ctmc", states = states, byrow = byRow, generator = gen, name = "testctmc")
probabilityatT(tcmc,1,useRCpp = TRUE)
```

---

### rain

**Alofi island daily rainfall**

---

**Description**

Rainfall measured in Alofi Island

**Usage**

```r
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**

```r
data(rain)

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

---
**rctmc**

*Function to generate a sequence of random CTMC transitions.*

**Description**

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

**Usage**

```r
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)
```

rmarkovchain

*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.
**rmarkovchain**

**Value**

Character Vector, data.frame, list or matrix

**Note**

Check the type of input

**Author(s)**

Giorgio Spedicato

**References**

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

**See Also**

markovchainFit, markovchainSequence

**Examples**

```r
# 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 <- rmarkovchain(n = 100, object = mcB, what = "list")

# define markovchainList object
statesNames <- c("a", "b", "c")
mcA <- 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)))
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)))
mcC <- 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)))
mclist <- new("markovchainList", markovchains = list(mcA, mcB, mcC))

# show the list of sequence
rmarkovchain(100, mclist, "list")
```
sales

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)

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

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  \textit{Stationary states of a markovchain object}

\section*{Description}
This method returns the stationary vector in matricial form of a markovchain object.

\section*{Usage}
\begin{verbatim}
steadyStates(object)
\end{verbatim}

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

\section*{Arguments}
\begin{itemize}
  \item \texttt{object} \hspace{1cm} \textit{A discrete markovchain object}
\end{itemize}

\section*{Value}
A matrix corresponding to the stationary states

\section*{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.

\section*{Author(s)}
Giorgio Spedicato

\section*{References}
A First Course in Probability (8th Edition), Sheldon Ross, Prentice Hall 2010

\section*{See Also}
markovchain

\section*{Examples}
\begin{verbatim}
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)
\end{verbatim}
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 0 1 455 ... - attr(*, "dimnames")=List of 2 ..$ : chr [1:8] "AAA" "AA" "A" "BBB" "CCC" "DDD" "EEE" "FFF"

References

Examples
data(tm_abs)

transition2Generator
Return the generator matrix for a corresponding transition matrix

Description
Calculate the generator matrix for a corresponding transition matrix

Usage
transition2Generator(P, t = 1, method = "logarithm")

Arguments

P transition matrix between time 0 and t
t time of observation
method "logarithm" returns the Matrix logarithm of the transition matrix

Value
A matrix that represent the generator of P
transitionProbability

Transition 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

```r
mymatr <- matrix(c(.4, .6, .1, .9), nrow = 2, byrow = TRUE)
Q <- transition2Generator(P = mymatr)
expm::expm(Q)
```
**Examples**

```r
c(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")
```

**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**

```r
verifyMarkovProperty(sequence, verbose = TRUE)
assessOrder(sequence, verbose = TRUE)
assessStationarity(sequence, nblocks, 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", "a", "b", "b", "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(0,1,2,1,0,0,0,0,1,2,2,1,0,0,1,0,0,0,0,0,1,1,1,1,1,1,0,0,0,0,0,2,1,0,
            0,2,1,0,0,0,0,0,1,1,1,2,2,2,0,0,2,1,1,1,1,1,1,0,1,1,1,0,2,
            0,1,1,0,0,1,2,2,0,0,0,0,0,0,0,0,2,2,2,1,1,1,0,1,1,1,0,0,2,1,1,
            0,0,0,0,0,2,2,1,1,1,1,1,2,1,2,0,0,0,1,2,2,2,0,0,0,0,1,1)
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)
```