Package ‘fidcMC’

June 27, 2016

Title Finite and Infinite, Discrete and Continuous markov chain toolkit

Description
This package includes toolkits for analyzing DTMC (discrete-time Markov chain) and CTMC (continuous-time Markov chain) in both finite and infinite stage settings.

Version 0.2

License MIT + file LICENSE

Author Bingxi Li, Qiwei Li, Jiaping Zhang

Maintainer Qiwei Li `<qwli@ucdavis.edu>`

Imports markovchain, lsa

R topics documented:

`check.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 exists a path that leads to state j leaving from state i with positive probability

Usage

```
check.accessible(mc, from, to)
```

Arguments

`mc`	mc class object
`from`	character of the state name
`to`	character of the state name

Value

a boolean value

References

markovchain CRAN package

Examples

```
statesNames=c("a","b","c")
markovB<-mc.create(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)), discrete = TRUE, infinite = FALSE)
check.accessible(markovB,"a","c")
```

`check.irreducible` *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

```
check.irreducible(mc)
```

Arguments

`mc`	mc class object

Value

A boolean value

References

markovchain CRAN package

Examples

```
statesNames<-c("a","b","c")
mat = 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)
)
mc.test = mc.create(mat,discrete=TRUE, infinite=FALSE)
check.irreducible(mc.test)
```

`fitDiscreteMarkovchain` *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.

Usage

```
fitDiscreteMarkovchain(data_seq, method = "mle", byrow = TRUE, nboot = 10L,
                    laplacian = 0, name = "", parallel = FALSE,
                    confidencelevel = 0.95, hyperparam = matrix())
```

Arguments

`data_seq`	A character 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.
`parallel`	Boolean. Whether to use parallel computing
`name`	Optional character for name slot.
`confidencelevel`	level for conficence intervals width. Used only when method equal to "mle".
`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 kxk where k is the number of states in the chain and the values should typically be non-negative integers.

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.

References

markovchain CRAN project

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

Examples

```
sequence<-c("a", "b", "a", "a", "a", "a", "b", "a", "b", "a", "b", "a", "a",
    "b", "b", "b", "a")
mcFitMLE<-fitDiscreteMarkovchain(data_seq=sequence)
mcFitBSP<-fitDiscreteMarkovchain(data_seq=sequence,method="bootstrap",
nboot=5, name="Bootstrap Mc")
```

---

generateMarkovchain *Generate a sequence of states from a Markov chain.*

Description

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

Usage

```
generateMarkovchain(n,mc)
```

Arguments

n	integer
mc	mc class

Value

A vector of states names

References

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

markovchain CRAN package

Examples

```
## The function is currently defined as
#define the Markov chain
statesNames=c("a","b","c")
mcB<-mc.create(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)), discrete = TRUE,infinite = FALSE)
outs<-generateMarkovchain(n=20,mcB)
```

getAbsorbingStates *Get Absorbing States of Markov chain*

Description

The function return absorbing states of the markovchain object.

Usage

```
getAbsorbingStates(mc)
```

Arguments

mc	mc class object

Value

vector of characters for state names

References

markovchain CRAN package

Examples

```
## The function is currently defined as
statesNames=c("a","b","c")
markovB<-mc.create(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)), discrete=TRUE, infinite=FALSE)
getAbsorbingStates(markovB)
```

getCommunicatingClasses *Get Communicating Classes of Markov chain*

Description

The function returns communicating classes of the markovchain object.

Usage

```
getCommunicatingClasses(mc)
```

Arguments

mc	mc class object

Value

list of separate communicating classes - state names

References

markovchain CRAN package

Examples

```
statesNames=c("a","b","c")
markovB<-mc.create(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)), discrete=TRUE, infinite=FALSE)
getCommunicatingClasses(markovB)
```

---

getConditionalDistribution

Extracts the conditional distribution from Markov Chain

Description

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

Usage

```
getConditionalDistribution(mc, state)
```

Arguments

`mc`	mc class.
`state`	char.

Value

The conditional distribution for each subsequent state.

References

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

markovchain CRAN package

Examples

```
statesNames=c("a","b","c")
markovB<- mc.create(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)),discrete = TRUE ,infinite = FALSE)
getConditionalDistribution(markovB,"b")
```

getHittingTime *getHittingTime*

Description

Calculate the hitting time matrix of a markov chain if it exists

Usage

```
getHittingTime(i=NULL, j=NULL, mc.obj)
```

Arguments

i	row or rows to be returned in the hitting time matrix. Default to NULL and to return entire matrix.
j	column or columns to be returned in the hitting time matrix. Default to NULL and to return entire matrix.
mc.obj	Markov chain object created by the 'mc.create' function

Value

returns the submatrix of the hitting time matrix

Examples

```
singleServer = function(i,j){
  if(i==1 && j==2)
    return(1)
  p = 0.3
  q = 0.7
  r = 0
  if(j == i+1)
    return(p)
  if(j == i-1)
    return(q)
  if(j==i)
    return(r)
  return(0)
}
ex = mc.create(pijdef=singleServer, discrete=TRUE, infinite=TRUE)
ans = getHittingTime(mc.obj = ex)
```

getRecurrentClasses *Get Recurrent Classes of Markov chain*

Description

The function returns recurrent classes of the markovchain object.

Usage

```
getRecurrentClasses(mc)
```

Arguments

mc	mc class object

Value

list of separate communicating classes - state names

References

markovchain CRAN package

Examples

```
statesNames=c("a","b","c")
markovB<-mc.create(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)), discrete=TRUE, infinite=FALSE)
getRecurrentClasses(markovB)
```

---

getStationaryDistribution

getStationaryDistribution

Description

Calculate the stationary distribution of a markov chain if it exists

Usage

```
getStationaryDistribution(mc.obj, epsilon=0.01, iteration=30, totalProb=0.9)
```

Arguments

mc.obj	Markov chain object created by the 'mc.create' function
epsilon	The threshold for the converging (in cosine). Needed in infinite markov chain. Default to 0.01.
iteration	The max iteration for the approximation algorithm. Needed in infinite markov chain. Default to 30
totalProb	The proportion of information that the approximate distribution contains. Default to 0.9

Value

In the case of finite, returns the stationary distribution. In the case of infinite, returns a list containing an success indicator, converge state count, approximated stationary distribution, and converge transition matrix.

Examples

```
singleServer = function(i,j){
  if(i==1 && j==2)
    return(1)
  p = 0.3
  q = 0.7
  r = 0
  if(j == i+1)
    return(p)
  if(j == i-1)
    return(q)
  if(j==i)
    return(r)
  return(0)
}
ex = mc.create(pijdef=singleServer, discrete=TRUE, infinite=TRUE)
ans = getStationaryDistribution(ex)
```

`getTransientStates` *Get Transient States of Markov chain*

Description

The function returns transient states of the markovchain object.

Usage

```
getTransientStates(mc)
```

Arguments

`mc`	mc class object

Value

vector of characters for state names

References

markovchain CRAN package

Examples

```
## The function is currently defined as
statesNames=c("a","b","c")
markovB<-mc.create(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)), discrete=TRUE, infinite=FALSE)
getTransientStates(markovB)
```

mc.create *mc.create*

Description

Create an appropriate markov chain object

Usage

```
mc.create(pijdef, stateNames=NULL, chainName=NULL, qidef=NULL, discrete, infinite, use_ratematrix
```

Arguments

pijdef	Transition matrix definition. Can be a matrix or a function with input(i,j) and output P(i to j).
stateNames	Names of states. Default to NULL, which forces the names to be 1 to n.
chainName	Name of the markov chain. Default to NULL.
qidef	A vector of holding time distribution parameters. Must have for continuous markov chain.
discrete	A logical value indicating if the markov chain is discrete (FALSE means continuous).
infinite	A logical value indicating if the markov chain is infinite (FALSE means finite).
use_ratematrix	If you wish to solve the problem with a matrix.

Value

returns an appropriate markov chain object

Examples

```
# discrete finite
threeHeads = function(){
  p = matrix(0, nrow=3, ncol=3)
  p[1, c(1,2)] = 0.5
  p[2, c(1,3)] = 0.5
  p[3, 1] = 1
  return(p)
}
ex = mc.create(pijdef = threeHeads(), discrete = TRUE, infinite = FALSE)

# discrete infinite
singleServer = function(i,j){
  p = 0.3
  q = 0.7
  r = 0
  if(j == i+1)
    return(p)
  if(j == i-1)
    return(q)
  if(j==i)
    return(r)
```

```
  return(0)
}
ex = mc.create(pijdef=singleServer, discrete=TRUE, infinite=TRUE)
```

Index