Package 'MPkn'

May 7, 2018

Type Package

Title Calculations of One Discrete Model in Several Time Steps

Version 0.1.0

Date 2018-05-03

Author Josef Brejcha

Maintainer Josef Brejcha <brchjo@gmail.com>

Suggests knitr, rmarkdown, matrixcalc, markovchain, matlib

VignetteBuilder knitr

Description A matrix discrete model having the form
'M[i+1] = (I + Q)*M[i]'.
The calculation of the values of 'M[i]' only for
pre-selected values of 'i'. The method of calculation
is presented in the vignette 'Fundament' ('Base'). Maybe it`s own
idea of the author of the package. A weakness is that the method
gives information only in selected steps of the process.
It mainly refers to cases with matrices that are not Markov chain.
If 'Q' is Markov transition matrix, then MUPkL() may be
used to calculate the steady-state distribution 'p' for
'p = Q*p'.
Matrix power of non integer (matrix.powerni()) gives
the same results as a mpower() from package 'matlib'.
References:
``Markov chains'',
(<https://en.wikipedia.org/wiki/Markov_chain#Expected_number_of_visits>).
Donald R. Burleson, Ph.D. (2005),
``ON NON-INTEGER POWERS OF A SQUARE MATRIX'',
(<http://www.blackmesapress.com/Eigenvalues.htm>).

License GPL (>= 3)

Encoding UTF-8

LazyData true

RoxygenNote 6.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2018-05-07 13:05:21 UTC

R topics documented:

`MPkn-package` *Calculations of One Discrete Model in Several Time Steps*

Description

A matrix discrete model having the form `M[i+1] = (I + Q)*M[i]`. The calculation of the values of `M[i]` only for pre-selected values of `i`. The method of calculation is presented in the vignette 'Fundament' ('Base'). Maybe it's own idea of the author of the package. A weakness is that the method gives information only in selected steps of the process. It mainly refers to cases with matrices that are not Markov chain.
If `Q` is markov transition matrix, then `MUPkL` may be used to calculate the steady-state distribution `p` for $p = Q * p$. See example bottom.
Matrix power of non integer (`matrix.powerni`) gives the same results as a `mpower` from package `matlib`.

Details

Package:	MPkn
Type:	Package
Version:	0.1.0
Date:	2018-05-03
License:	GPL (>= 3)

Author(s)

Josef Brejcha

Maintainer: Josef Brejcha <brchjo@gmail.com>

References

Ton van den Boom, "Discrete-time systems analysis" (2006), Additional Lecture Notes for the course SC4090, www.dcsc.tudelft.nl/~sc4060/transp/discreteNOTES.pdf
Richard Weber, "Markov Chains" (2011), http://www.statslab.cam.ac.uk/~rrw1/markov/M.pdf
"Examples of Markov chains", https://en.wikipedia.org/wiki/Examples_of_Markov_chains
"Markov chains", https://en.wikipedia.org/wiki/Markov_chain#Expected_number_of_visits
Donald R. Burleson, Ph.D. "ON NON-INTEGER POWERS OF A SQUARE MATRIX", (2005), http://www.blackmesapress.com/Eigenvalues.htm

Examples

```
require(MPkn)
require(markovchain)
options(digits = 14)
n = 12
k = 2
rz = 11
P = array(0, c(rz, rz))
for (i in 1:rz){
  po = runif(rz)
  P[i, ] = po/sum(po)
}
I = diag(1, rz, rz)
Myy = MUPkL(P, P, I, n, k, c(1:rz))
StSy = NULL
for (i in 1:rz) StSy = c(StSy, Myy$Navg[,,i][n])
mrkv = new("markovchain", transitionMatrix = P)
StSx = steadyStates(mrkv)
print("MPkn");  print(StSy)
print("markovchain"); print(StSx)
```

---

matrix.powerni   *Matrix Power of Non Integer*

Description

Square matrix power of non integer.

Usage

```
matrix.powerni(A, p)
```

Arguments

A	square matrix
p	non integer (real) number

Value

square matrix

Author(s)

Josef Brejcha

References

Donald R. Burleson, Ph.D., "ON NON-INTEGER POWERS OF A SQUARE MATRIX", http://www.blackmesapress.com/Eigenvalues.htm

Examples

```
require(MPkn)
require(matrixcalc)
matmult <- function(A, B){
C = matrix(numeric(4), 2, 2)
for (i in 1:2){
for (j in 1:2){ C[i, j] = sum(A[i, ]*B[, j])}
}
return(C)
}
I = diag(1, 2, 2)
P = matrix(c(0.9, 0.3, 0.1, 0.7), 2, 2)
M1 = P
M2 = matmult((I + P), M1)
M4 = matmult((I + t(matrix.power(P, 2))), M2)
M8 = matmult((I + t(matrix.power(P, 4))), M4)
M16 = matmult((I + t(matrix.power(P, 8))), M8)
## =====================
Q = list()
Q[[1]] = M1
Q[[2]] = matmult(M2, matrix.inverse(M1)) - I
Q[[3]] = matrix.powerni(matmult(M4, matrix.inverse(M2)) - I, 1/2)
Q[[4]] = matrix.powerni(matmult(M8, matrix.inverse(M4)) - I, 1/4)
Q[[5]] = matrix.powerni(matmult(M16, matrix.inverse(M8)) - I, 1/8)
print("Q"); print(Q)
S = as.matrix(Q[[1]], 2, 2)
for (i in 2:5){
  S = S + as.matrix(Q[[i]], 2, 2)
}
Qs = S/5
print("Qs"); print(Qs)
```

MPKlMatrix *Creates a matrix of specified row of output* MUPkLo

---

Description

Specified row of output MUPkLo is a number step of process which computes MUPkLo function.

Usage

```
MPKlMatrix(Mx, step, nc, sta)
```

Arguments

Mx	output matrix of MUPkLo
step	row name of matrix Mx
nc	number of columns of matrix Mx
sta	vector with column indices of input matrices into MUPkLo

Value

The matrix with nc rows and columnes.

Author(s)

Josef Brejcha

Examples

```
A <- array(c(0.9, 0.6, 0.8, 0.05, 0.2, 0.05, 0.05, 0.2, 0.15), c(3, 3))
P <- array(c(0.9, 0.6, 0.8, 0.05, 0.2, 0.05, 0.05, 0.2, 0.15), c(3, 3))
U <- array(c(0.8, 0.8, 0.7, 0.06, 0.02, 0.2, 0.14, 0.18, 0.1), c(3, 3))
sta <- c(1, 2, 3)
k <- c(1, 0, 1, 0)
n <- c(5, 7, 12, 17)
Mx <- MUPkLo(A, P, U, n, k, sta)
M100 = MPKlMatrix(Mx, step = 100, nc = 3, sta = c(1, 2, 3))
```

MUPkL *Calculations of one discrete model in several time steps*

Description

M[i + 1] = (I + Q) * M[i] process in several selected steps.
$Q = P * U$, matrix multiplication.
Computation process only in the following steps i:

$$c(1 : k, k * 2^(1 : (n - k))) \text{ where k > 1;}$$
$$c(2^((1 : n) - 1)) \text{ for k == 0;}$$
$$seq(1, n, 1) \text{ for k == 1.}$$

$M[2*i] = (I + Q^i) * M[i]$ for `k == 0`.

Usage

```
MUPkL(A, P, U, n, k, sta)
```

Arguments

A	starting square matrix a process at time 0
P	basic transition matrix chain
U	correction matrix chain
n	The number of steps. The length of the steps depends on the value of k.
k	`k == 0` ... step length `i` is equal to $2^(i-1), i = 1, 2, .., n$. `k == 1` ... step length `i` is equal to 1. `k > 1` ... The first `n` steps has a length equal to 1. Other then have a length of twice the previous step.
sta	Vector whose values are the indices of the columns of the `A` matrix.

Details

Both `n` and `k` are single positive integers.

Value

A list with following components:

N	sum values of entries into state
Navg	average N in interval `(i - 1, i]`
Tavg	$1/Navg$
x	steps vector

Author(s)

Josef Brejcha

Examples

```
A <- array(c(2, 3, 1, 4, 2, 1, 3, 1, 2), c(3, 3))
P <- array(c(0.9, 0.6, 0.8, 0.05, 0.2, 0.05, 0.05, 0.2, 0.15),
 c(3, 3))
U <- array(c(0.8, 0.8, 0.7, 0.06, 0.02, 0.2, 0.14, 0.18, 0.1),
 c(3, 3))
sta <- c(1, 3)
k <- 3
n <- 8
M33 <- MUPkL(A, P, U, n, k, sta)
print(M33$N)
k <- 1
```

```
n <- 24
M11 <- MUPkL(A, P, U, n, k, sta)
print(M11$N)
k <- 0
n <- 6
M00 <- MUPkL(A, P, U, n, k, sta)
print(M00$N)
```

MUPkLo *Calculations of one discrete model in several time steps*

Description

$M[i+1] = (I+Q)*M[i]$ process in several selected steps.
$Q = P*U$, matrix multiplication.
The calculation is performed in steps determined by integer vectors `k` and `n`. The sections defined by integers `k` and `n` are applied as follows:

`k[i] == 1`	$\ldots M[n] = sum(i=0, n-1)(Q^i)*A$	, for $n = 0, 1, 2, \ldots$
`k[i] == 0`	$\ldots M[2n] = (I+Q^n)*M[n]$	, for $n = r*2^i, i = 1, 2, 3, ..$

where `r` is the last step before section with `k[i] == 0`

Usage

```
MUPkLo(A, P, U, n, k, sta)
```

Arguments

`A`	an initial square matrix a process at time 0
`P`	a basic transition matrix chain
`U`	a correction matrix chain
`n`	An integer vector cumulative number of individual process steps. `n[1] > 0`, `n[i] > n[i-1]`.
`k`	A vector of 0 and 1 identifying the mode of calculation in the stretch step. `k[i] = 1` for `rn[j] = rn[j-1]+1`, `k[i] = 0` for `rn[j] = 2*rn[j-1]`, where `rn[j]` is the j-th row name of the output value matrix.
`sta`	Vector of indices of the columns of the matrix `M`. The matrix `M` contains the cumulative number of inputs `mij` from the state of the `i` to the state `j`.

Details

Relationship between `k` and `n`:
`length(k) == lentgh(n)`.
It is recommended to determine the value of well vectors `n` and `k`.

Value

An array (r x slp x sta) where

`r`	$r = n[length(n)]$
`slp`	Vector of column indices of the matrix P
`sta`	Vector of column indices of the matrix M

Row of the output matrix (array) is the column in the matrix `M` and whose number is specified in the `sta`. The matrix `M` contains the cumulative number of inputs `mij` from the state of the `i` to the state `j`.

Author(s)

Josef Brejcha

Examples

```
A = array(c(-2, -3, 1, 4, -2, 1, 3, -1, -2), c(3, 3))
P <- array(c(0.9, 0.6, 0.8, 0.05, 0.2, 0.05, 0.05, 0.2, 0.15), c(3, 3))
U <- array(c(0.8, 0.8, 0.7, 0.06, 0.02, 0.2, 0.14, 0.18, 0.1), c(3, 3))
sta <- 3
Ao <- A
k <- c(1, 0, 1, 0)
n <- c(5, 7, 12, 17)
# Steps, in which will compute the value of the Mx:
# 1, 2, 3, 4, 5, 10, 20, 21,  22, 23, 24, 25, 50, 100, 200, 400, 800
Mx <- MUPkLo(A, P, U, n, k, sta)
print(Mx)
A <- Ao
Mb <- MUPkLo(A, P, U, n = 100, k = 1, sta)
Mb[100,,]
```

---

radekW *The Numbers of Rows of the Output Matrix*

Description

The numbers of rows of the output matrix. These numbers are determined by the vectors of `n` and `k`.

Usage

```
radekW(n, k)
```

Arguments

- `n` An integer vector cumulative number of individual process steps.
  `n[1] > 0`, `n[i] > n[i-1]`.
- `k` A vector of 0 and 1 identifying the mode of calculation in the stretch step.
  `k[i] = 1` for `rn[j] = rn[j-1]+1`,
  `k[i] = 0` for `rn[j] = 2*rn[j-1]`,
  where `rn[j]` is the j-th row name of the output value matrix.

Value

Matrix size `n[length(n)] x 1`.
The values of the rows of the matrix are the numbers of steps of the chain.

Author(s)

Josef Brejcha

Examples

```
radekW(n = c(3, 5, 8, 9, 11), k = c(1, 0, 1, 0, 0))
```

Index