# Package ‘MPkn’

May 7, 2018

<table>
<thead>
<tr>
<th><strong>Type</strong></th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Title</strong></td>
<td>Calculations of One Discrete Model in Several Time Steps</td>
</tr>
<tr>
<td><strong>Version</strong></td>
<td>0.1.0</td>
</tr>
<tr>
<td><strong>Date</strong></td>
<td>2018-05-03</td>
</tr>
<tr>
<td><strong>Author</strong></td>
<td>Josef Brejcha</td>
</tr>
<tr>
<td><strong>Maintainer</strong></td>
<td>Josef Brejcha &lt;<a href="mailto:brchjo@gmail.com">brchjo@gmail.com</a>&gt;</td>
</tr>
<tr>
<td><strong>Suggests</strong></td>
<td>knitr, rmarkdown, matrixcalc, markovchain, matlib</td>
</tr>
<tr>
<td><strong>VignetteBuilder</strong></td>
<td>knitr</td>
</tr>
</tbody>
</table>
| **Description** | A matrix discrete model having the form 

\[ M[i+1] = (I + Q) \times M[i]. \]

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 MUPlK() may be used to calculate the steady-state distribution \( p \) for

\[ p = Q \times p. \]

Matrix power of non integer (matrix.powerni()) gives the same results as a mpower() from package 'matlib'.

**References:**


<table>
<thead>
<tr>
<th><strong>License</strong></th>
<th>GPL (&gt;= 3)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Encoding</strong></td>
<td>UTF-8</td>
</tr>
<tr>
<td><strong>LazyData</strong></td>
<td>true</td>
</tr>
<tr>
<td><strong>RoxygenNote</strong></td>
<td>6.0.1</td>
</tr>
<tr>
<td><strong>NeedsCompilation</strong></td>
<td>no</td>
</tr>
</tbody>
</table>
Calculations of One Discrete Model in Several Time Steps

Description

A matrix discrete model having the form $M[i+1] = (I + Q)\times M[i]$. 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 $p = Q \times p$. See example bottom.

Matrix power of non integer (`matrix.powerni`) gives the same results as a `mpower` from package `matlib`.

Details

<table>
<thead>
<tr>
<th>Package:</th>
<th>MPkn</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type:</td>
<td>Package</td>
</tr>
<tr>
<td>Version:</td>
<td>0.1.0</td>
</tr>
<tr>
<td>Date:</td>
<td>2018-05-03</td>
</tr>
<tr>
<td>License:</td>
<td>GPL (&gt;= 3)</td>
</tr>
</tbody>
</table>

Author(s)

Josef Brejcha

Maintainer: Josef Brejcha <brchjo@gmail.com>
References


Donald R. Burleson, Ph.D. "ON NON-INTEGER POWERS OF A SQUARE MATRIX", (2005), [http://www.blackmesapress.com/Eigenvalues.htm](http://www.blackmesapress.com/Eigenvalues.htm)

Examples

```r
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[1,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-INTEGRAL POWERS OF A SQUARE MATRIX", http://www.blackmesapress.com/Eigenvalues.htm

Examples

```r
require(MPKl)
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.powern(matmult(M4, matrix.inverse(M2)) - I, 1/2)
Q[[4]] = matrix.powern(matmult(M8, matrix.inverse(M4)) - I, 1/4)
Q[[5]] = matrix.powern(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 MPKlLo
**Description**

Specified row of output \texttt{MUPkLo} is a number step of process which computes \texttt{MUPkLo} function.

**Usage**

\texttt{MPKLMatrix(Mx, step, nc, sta)}

**Arguments**

\begin{itemize}
\item \texttt{Mx} \hspace{1cm} output matrix of \texttt{MUPkLo}
\item \texttt{step} \hspace{1cm} row name of matrix \texttt{Mx}
\item \texttt{nc} \hspace{1cm} number of columns of matrix \texttt{Mx}
\item \texttt{sta} \hspace{1cm} vector with column indices of input matrices into \texttt{MUPkLo}
\end{itemize}

**Value**

The matrix with \texttt{nc} rows and columns.

**Author(s)**

Josef Brejcha

**Examples**

\begin{verbatim}
A <- array(c(0.9, 0.6, 0.8, 0.05, 0.2, 0.05, 0.2, 0.05, 0.2, 0.15), c(3, 3))
P <- array(c(0.9, 0.6, 0.8, 0.05, 0.2, 0.05, 0.2, 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))
\end{verbatim}

---

**MUPkL**

Calculations of one discrete model in several time steps

---

**Description**

\texttt{M[i + 1]} = (\texttt{I} + \texttt{Q}) \times \texttt{M[i]} process in several selected steps.

\texttt{Q} = \texttt{P} \times \texttt{U}, matrix multiplication.

Computation process only in the following steps \texttt{i}:

\begin{verbatim}
c(1 : k, k \times 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.
\end{verbatim}
\[ M[2 * i] = (I + Q^i) * M[i] \] for \( k = 0 \).

**Usage**

\[
\text{MUPkL}(A, P, U, n, k, \text{sta})
\]

**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**
  - \( 1/\text{Navg} \)
- **x**
  - steps vector

**Author(s)**

Josef Brejcha

**Examples**

```r
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
```
Calculations of one discrete model in several time steps

Description

$M[i+1] = (I + Q) \times M[i]$ process in several selected steps.

$Q = P \times U$, 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:

\[ k[i] = 1 \quad \ldots M[n] = \text{sum}(i = 0, n - 1)(Q^i) \times A \quad \text{for } n = 0, 1, 2, \ldots \]
\[ k[i] = 0 \quad \ldots M[2n] = (I + Q^n) \times M[n] \quad \text{for } n = r \times 2^i, i = 1, 2, 3, \ldots \]

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 $m_{ij}$ from the state of the $i$ to the state $j$. 

```r
M11 <- MUPkLo(A, P, U, n, k, sta)
print(M11$N)
k <- 0
n <- 6
M00 <- MUPkLo(A, P, U, n, k, sta)
print(M00$N)
```
Details

Relationship between $k$ and $n$:

\[ \text{length}(k) = \text{length}(n). \]

It is recommended to determine the value of well vectors $n$ and $k$.

Value

An array $(r \times \text{slp} \times \text{sta})$ where

\[
\begin{align*}
\text{r} & \quad r = n[\text{length}(n)] \\
\text{slp} & \quad \text{Vector of column indices of the matrix P} \\
\text{sta} & \quad \text{Vector of column indices of the matrix M}
\end{align*}
\]

Row of the output matrix (array) is the column in the matrix $M$ and whose number is specified in the $\text{sta}$. The matrix $M$ contains the cumulative number of inputs $m_{ij}$ from the state of the $i$ to the state $j$.

Author(s)

Josef Brejcha

Examples

\[
\begin{align*}
\text{A} & \equiv \text{array}(c(-2, -3, 1, 4, -2, 1, 3, -1, -2), c(3, 3)) \\
\text{P} & \equiv \text{array}(c(0.9, 0.5, 0.8, 0.05, 0.2, 0.05, 0.2, 0.05, 0.15), c(3, 3)) \\
\text{U} & \equiv \text{array}(c(0.8, 0.8, 0.7, 0.06, 0.02, 0.2, 0.14, 0.18, 0.1), c(3, 3)) \\
\text{sta} & \equiv 3 \\
\text{Ao} & \equiv \text{A} \\
\text{k} & \equiv c(1, 0, 1, 0) \\
\text{n} & \equiv c(5, 7, 12, 17) \\
\# \text{Steps, in which will compute the value of the Mx:} \\
\# & \equiv 1, 2, 3, 4, 5, 10, 20, 21, 22, 23, 24, 25, 50, 100, 200, 400, 800 \\
\text{Mx} & \equiv \text{MUPkLo}(\text{A}, \text{P}, \text{U}, \text{n}, \text{k}, \text{sta}) \\
\text{print(Mx)} \\
\text{A} & \equiv \text{Ao} \\
\text{Mb} & \equiv \text{MUPkLo}(\text{A}, \text{P}, \text{U}, \text{n} = 100, \text{k} = 1, \text{sta}) \\
\text{Mb}[100,,] 
\end{align*}
\]

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

*Topic MPKIMatrix
   MPKIMatrix, 4
*Topic MUPkLo
   MUPkLo, 7
*Topic MUPkL
   MUPkL, 5
*Topic matrix.powerni
   matrix.powerni, 3
*Topic radekW
   radekW, 8

matrix.powerni, 3
MPKIMatrix, 4
MPkn-package, 2
MUPkL, 5
MUPkLo, 4, 5, 7

radekW, 8