The markovchain Package: A Package for Easily Handling Discrete Markov Chains in R

Giorgio Alfredo Spedicato **Mirko Signorelli**

Abstract

markovchain aims to fill a gap within R packages providing S4 classes and methods to easily handling discrete markov chains, both homogeneous and inhomogeneous. The S4 class structure will be presented as well implemented classes and methods. Applied examples will follow

Keywords: markov chain, transition probabilities.

1. Introduction

Markov chains represent a class of stochastic processes of great interest for the wide spectrum of practical applications. In particular, discrete time Markov chains (DTMC) permit to model the transition probabilities between discrete states by the aid of matrices. Various R packages deals with models that are based on Markov chains: **msm** (Jackson 2011) handle Multi-State Models for panel data, **mcmcR** (Geyer and Johnson 2013) is one among the many package that implements Monte Carlo Markov Chain approach that is widely used to estimate parameters of parametric statistical models. Hidden Markov models with covariates are fit with package **hmm**, whilst **mstate** fits Multi-State Models based on Markov chains for survival analysis (**?**). The R statistical environment (R Core Team 2013) nevertheless seems to lack a simple package that coherently defines S4 classes for discrete Markov chains and that allows to perform probabilistic analysis, statistical inferences and applications. For the sake of completeness, **markovchain** is not the first package specifically dedicated to DTMC analysis, being **DTMCPack** the first one, Nicholson (2013). Nevertheless **markovchain** package (Spedicato 2013) aims to offer greater flexibility in handling discrete time Markov chains than existing solutions, providing S4 classes for both homogeneous and non-homogeneous Markov chains as well as methods suited to perform statistical and probabilistic analysis. By the way, other scientifical softwares provides functions specifically designed to analyze Markov chains, as Mathematica 9 for example (Wolfram Research 2013).

The paper is structured as follows: Section 2 briefly reviews mathematic and definitions regarding discrete Markov chains, Section 3 discusses on how to handle and manage Markov chains objects within the package, Section 4 and Section 5 show how to perform probabilistic and statistical modelling whilst Section 6 presents applied examples of discrete Markov chains in various fields.

Finally, the **markovchain** package depends by following R packages: **expm** (Goulet, Dutang, Maechler, Firth, Shapira, Stadelmann, and expm-developers@lists.R-forge.R-project.org 2013)

to perform efficient matrices powers; **igraph** (Csardi and Nepusz 2006) to perform pretty plotting of `markovchain` objects and **matlab** (Roebuck 2011) that contains functions for matrix management and calculations that emulate those within Matlab environment.

2. Markov chains mathematic revies

2.1. General Definitions

A discrete-time Markow chain is a sequence of random variables $X_1, X_2, X_3, \ldots$ characterized by memorylessness property (also known as Markov property, see Equation 1), that is that the distribution of forthcoming state (of X_{n+1}) depends only by the current state (X_n) and not by previous ones $X_{n-1}, X_{n-2}, \ldots, X_1$.

$$Pr\left(X_{n+1} = x_{n+1} \,|X_1 = x_1, X_2 = x_{2,}..., X_n = x_n\right) = Pr\left(X_{n+1} = x_{n+1} \,|X_n = x_n\right). \tag{1}$$

The set of possible states $S = \{s_1, s_2, ..., s_r\}$ of X_j is named the state space of the chain. In discrete-time Markov chain, S is finite or countable.

A Markow chain is time-homogeneous the property shown in Equation 2 holds. It implies no change in the underlying transition probabilities as time goes on.

$$Pr\left(X_{n+1} = x \,|X_n = y\right) = Pr\left(X_n = x \,|X_{n-1} = y\right), \tag{2}$$

The chain successively moves from one state to another (this change is named either 'transition' or 'step') and the probability p_{ij} to move from state s_i to state s_j is named transition probability (see Equation 3).

$$p_{ij} = Pr\left(X_1 = s_j \,|X_0 = s_i\right). \tag{3}$$

The probability of going from state i to j in n steps is $p_{ij}^{(n)} = Pr\left(X_n = s_j \,|X_0 = s_i\right)$.

If the Markov chain is stationary $p_{ij} = Pr\left(X_{k+1} = s_j \,|X_k = s_i\right)$ and $p_{ij}^{(n)} = Pr\left(X_{n+k} = s_j \,|X_k = s_i\right)$, where $k > 0$.

The probability distributions of transitions from one state to another can be represented into a transition matrix P, where each element of position (i, j) represents the probability p_{ij}. For example, if $r = 3$ the transition matrix P is shown in Equation 4

$$P = \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix}. \tag{4}$$

The distribution over the states can be written as a stocastic row vector x: if the current state of x is s_2, $x = (0\,1\,0)$. As a consequence, the relation between $x^{(1)}$ and $x^{(0)}$ is $x^{(1)} = x^{(0)}P$ and, recursively, $x^{(2)} = x^{(0)}P^2$, $x^{(n)} = x^{(0)}P^n$, $n > 0$.

Discrete Markov chains are explained in most stochastic processes theory books, see for example Ching and Ng (2006). Valuable references freely available online are: **?**, Snell (1999), Wikipedia (2013) and Bard (2000).

2.2. Properties and classification of states

A state s_j is said "reacheable" from state s_i (written $s_i \rightarrow s_j$) if a system started in state s_i has a positive probability of transitioning into state s_j at a certain point. If both $s_i \rightarrow s_j$ and $s_j \rightarrow s_i$ the states s_i and s_j are said to communicate.

A communicating class is a set of states that communicate with each other. A Markov chain is composed by one or more communicating classes. A communicating class is said to be "closed" if no states outside of the class can be reached from any state inside it. If the Markov chain is composed by only one communicating class (if all states in the chain communicate to each other), it is said "irreducible".

A state is said "periodic" if it can only return to itself after a fixed number of transitions greater than 1 (or multiple of a fixed number), else it is called "aperiodic". A state s_i is said to be "transient" if, given that we start in state s_i, there is a positive probability that we will never return to s_i; instead, when $p_{ii} = 1$, s_i is defined an "absorbing state", i.e. a closed communicating class composed by only one state. The Markov chain is absorbing if there is at least one recurrent state; otherwise, the chain is said to be ergodic (or irreducible) and it is possible to get to any state from any state.
A Markov chain is said in "canonic form" if the transition matrix is shown in a block form being the closed comminicating classes shown at the beginning of the matrix diagonal.

A state s_i has a period k if any return to state s_i must occur in multiplies of k steps, that is $k = gcd\{n : Pr(X_n = s_i | X_0 = s_i) > 0\}$,where 'gcd' is the greatest common divisor. If $k = 1$ the state is said to be aperiodic, if $k > 1$ the state is periodic with period k.

Given a time homogeneous Markov chain with transition matrix P, a stationary vector v is a vector satisfying $0 \leq v_j \leq 1 \, \forall j$, $\sum_{j \in S} v_j = 1$ and $v_j = \sum_{i \in S} v_i p_{ij}$.
A Markov chain is said to be regular if some power of the transition matrix has positive elements only. Regular Markov chains form a subset of ergodic chains.

An interesting property of regular Markov chains is that, if P is the $k \times k$ transition matrix and $z = (z_1, ..., z_k)$ is the eigenvector of P having $\sum_{i=1}^{k} z_i = 1$ then Equation 5 holds.

$$\lim_{n \to \infty} P^n = Z, \tag{5}$$

where Z is the matrix having all rows equal to z.

It is possible to analyze timing of a state being reached. The first passage time from state i to state j is the number T_{ij} of steps taken by the chain until it arrives for the first time at state j given that $X_0 = i$. Its probability distribution is defined by Equation 7

$$h_{ij}{}^{(n)} = P(T_{ij} = n) = P(X_n = j, X_{n-1} \neq j, \ldots, X_1 \neq j | X_0 = i) \tag{6}$$

and it can be found recursively using Equation **??**, knowing that ${h_{ij}}^{(n)} = p_{ij}$.

$$ {h_{ij}}^{(n)} = \sum_{k \in S - \{j\}} p_{ik} {h_{kj}}^{(n-1)} \tag{7} $$

2.3. A short example

Consider the following numerical example. Suppose we have a Markov chain with a set of 3 possible states s_1, s_2 and s_3. Let the transition matrix be defined in Equation 8

$$ P = \begin{bmatrix} 0.5 & 0.2 & 0.3 \\ 0.15 & 0.45 & 0.4 \\ 0.25 & 0.35 & 0.4 \end{bmatrix}. \tag{8} $$

In P, $p_{11} = 0.5$ is the probability that $X_1 = s_1$ given that we observed $X_0 = s_1$ is 0.5, and so on. If the current state is $X_0 = s_2$, then Equation 9 and Equation 10 hold.

$$ x^{(1)} = (0\,1\,0) \begin{bmatrix} 0.5 & 0.2 & 0.3 \\ 0.15 & 0.45 & 0.4 \\ 0.25 & 0.35 & 0.4 \end{bmatrix} = (0.15\,0.45\,0.4)\,, \tag{9} $$

$$ x^{(2)} = x^{(n+1)} P = (0.15\,0.45\,0.4) \begin{bmatrix} 0.5 & 0.2 & 0.3 \\ 0.15 & 0.45 & 0.4 \\ 0.25 & 0.35 & 0.4 \end{bmatrix} = (0.2425\,0.3725\,0.385) \tag{10} $$

and so on. The last result means that $Pr\,(X_2 = s_1\,|X_0 = s_2\,) = 0.2425$, $Pr\,(X_2 = s_2\,|X_0 = s_2\,) = 0.3725$ and $Pr\,(X_2 = s_3\,|X_0 = s_2\,) = 0.385$.

3. The structure of the package

3.1. Creating markovchain objects

The package **markovchain** contains classes and methods that handle markov chain in a convenient manner.

The package is loaded within the R command line as follows:

```
R> #library(markovchain)
```

The `markovchain` and `markovchainList` S4 classes (Chambers 2008) is defined within the **markovchain** package as displayed:

```
Class "markovchain" [in ".GlobalEnv"]

Slots:
```

```
Name:            states            byrow transitionMatrix
Class:        character          logical            matrix

Name:              name
Class:        character

Class "markovchainList" [in ".GlobalEnv"]

Slots:

Name:  markovchains         name
Class:         list    character
```

The first class has been designed to handle homogeneous Markov chain processes, whilst the latter (that is itself a list of `markovchain` objects) has been designed to handle non-homogeneous Markov chains processes.

Any element of `markovchain` class is comprised by following slots:

1. `states`: a character vector, listing the states for which transition probabilities are defined.
2. `byrow`: a logical element, indicating whether transition probabilities are shown by row or by column.
3. `transitionMatrix`: the probabilities of transition matrix.
4. `name`: optional character element to name the Markov chain.

`markovchainList` objects are defined by following slots:

1. `markovchains`: a list of `markovchain` objects.
2. `name`: optional optional character element to name the Markov chain.

`markovchain` objects can be created either in a long way, as the following code shows,

```
R> weatherStates<-c("sunny", "cloudy", "rain")
R> byRow<-TRUE
R> weatherMatrix<-matrix(data=c(0.70, 0.2,0.1,
+                          0.3,0.4, 0.3,
+                          0.2,0.45,0.35),byrow=byRow, nrow=3,
+                        dimnames=list(weatherStates, weatherStates))
R> mcWeather<-new("markovchain",states=weatherStates, byrow=byRow,
+                 transitionMatrix=weatherMatrix, name="Weather")
```

or in a shorter way, displayed below.

```
R> mcWeather<-new("markovchain", states=c("sunny", "cloudy", "rain"),
+      transitionMatrix=matrix(data=c(0.70, 0.2,0.1,
+                             0.3,0.4, 0.3,
+                             0.2,0.45,0.35),byrow=byRow, nrow=3),
+             name="Weather")
```

When `new("markovchain")` is called alone a defaut Markov chain is created.

```
R> defaultMc<-new("markovchain")
```

The quicker form of object creation is made possible thanks to the implemented `initialize` S4 method that assures:

- the `transitionMatrix` to be a transition matrix, i.e., all entries to be probabilities and either all rows or all columns to sum up to one, according to the value of `byrow` slot.
- the columns and rows nams of `transitionMatrix` to be defined and to coincide with `states` vector slot.

`markovchain` objects can be collected in a list within `markovchainList` S4 objects as following example shows.

```
R> mcList<-new("markovchainList",markovchains=list(mcWeather, defaultMc),
+                  name="A list of Markov chains")
```

3.2. Handling markovchain objects

markovchain contains two classes, `markovchain` and `markovchainList`. `markovchain` objects handle discrete Markov chains, whilst `markovchainList` objects consists in list of `markovchain` that can be useful to model non - homogeneous Markov chain processess.

Table 1 lists which of implemented methods handle and manipulate `markovchain` objects.

Operations on the markovchains objects can be easily performed. Using the previously defined matrix we can find what is the probability distribution of expected weather states two and seven days after, given actual state to be cloudy.

```
R> initialState<-c(0,1,0)
R> after2Days<-initialState*(mcWeather*mcWeather)
R> after7Days<-initialState*(mcWeather^7)
R> after2Days
```

```
     sunny cloudy  rain
[1,]  0.39  0.355 0.255
```

```
R> after7Days
```

Method	Purpose
*	Direct multiplication for transition matrices.
[	Direct access to the elements of the transition matrix.
==	Equality operator between two transition matrices.
as	Operator to convert from markovchain objects toward data.frame and table object.
dim	Dimenion of the transition matrix.
plot	plot method for markovchain objects.
print	print method for markovchain objects.
show	show method for markovchain objects.
states	name of the transition states.
t	Transposition operator (it switches byrow slot value and modifies the transition matrix coherent

Table 1: **markovchain** methods: matrix handling.

```
         sunny    cloudy      rain
[1,] 0.4622776 0.3188612 0.2188612
```

A similar answer could have been obtained if the probabilities were defined by column. A column - defined probability matrix could be set up either creating a new matrix or transposing an existing markovchain object thanks to the t vector.

```
R> initialState<-c(0,1,0)
R> mcWeatherTransposed<-t(mcWeather)
R> after2Days<-(mcWeatherTransposed*mcWeatherTransposed)*initialState
R> after7Days<-(mcWeather^7)*initialState
R> after2Days

        [,1]
sunny  0.390
cloudy 0.355
rain   0.255

R> after7Days

           [,1]
sunny  0.3172005
cloudy 0.3188612
rain   0.3192764
```

Basing informational methods have been defined for markovchain objects to quickly get states and dimension.

```
R> states(mcWeather)

[1] "sunny"  "cloudy" "rain"
```

```
R> dim(mcWeather)

[1] 3
```

A direct access to transition probabilities is provided both by `transitionProbability` method and "[" method.

```
R> transitionProbability(mcWeather, "cloudy","rain")

[1] 0.3

R> mcWeather[2,3]

[1] 0.3
```

The `markovchain` object's underlying transition Matrix can be displayed using `print`, `show` methods (the latter being less laconic). Similarly, the underlying transition probability diagram can be plot by the use of `plot` method (as shown in Figure 1) that was based on **igraph** package (Csardi and Nepusz 2006) used to manage and analyze networks data. The **igraph** package (Csardi and Nepusz 2006) is used for plotting, being `plot` method a wrapper of `plot.igraph` for `igraph` S4 objects defined within the **igraph** package. Additional parameters can be passed by means of `...` to control the network graph layout as shown.

```
R> print(mcWeather)

       sunny cloudy rain
sunny    0.7   0.20 0.10
cloudy   0.3   0.40 0.30
rain     0.2   0.45 0.35

R> show(mcWeather)

Weather
 A  3 - dimensional discrete Markov Chain with following states
 sunny cloudy rain
 The transition matrix   (by rows)  is defined as follows
       sunny cloudy rain
sunny    0.7   0.20 0.10
cloudy   0.3   0.40 0.30
rain     0.2   0.45 0.35
```

Exporting to `data.frame` is possible and similarly it is possible to import from.

```
R> mcDf<-as(mcWeather, "data.frame")
R> mcNew<-as(mcDf, "markovchain")
```

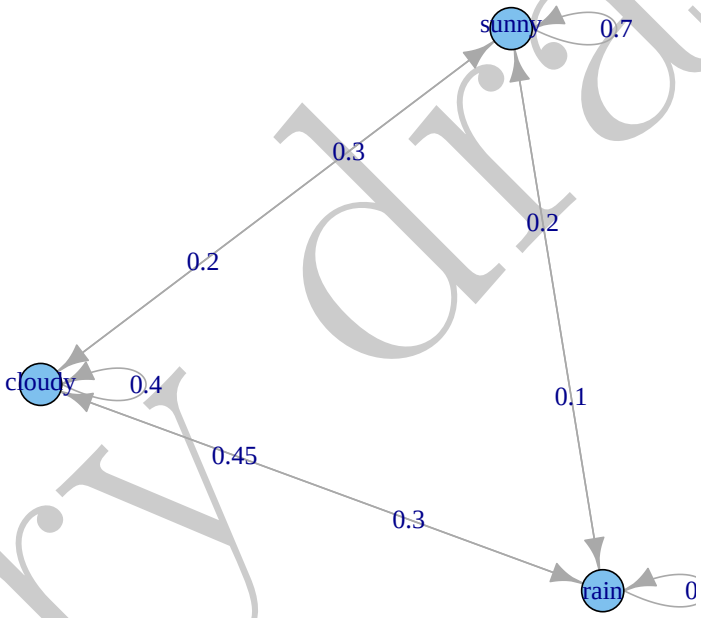

Figure 1: Weather example Markov chain plot

Similarly it is possible to create a `markovchain` object from a suitable two-way contingency table using `as(table, "markovchain")` code (see Section 6 for few examples).

Non-homogeneous markov chains can be created with the aid of `markovchainList` object. The example that follows arises from Health Insurance, where the costs associated to patients in a Continuous Care Health Community (CCHC) are modelled by a non-homogeneous Markov Chain, since the transition probabilities change by year. Methods explicitely written for `markovchainList` objects are: `print`, `show`, `dim` and `[`.

```
Continuous Care Health Community  list of Markov chain(s)
Markovchain  1
state t0
 A  3 - dimensional discrete Markov Chain with following states
 H I D
 The transition matrix   (by rows)  is defined as follows
    H   I   D
H 0.7 0.2 0.1
I 0.1 0.6 0.3
D 0.0 0.0 1.0

Markovchain  2
state t1
 A  3 - dimensional discrete Markov Chain with following states
 H I D
 The transition matrix   (by rows)  is defined as follows
    H   I   D
H 0.5 0.3 0.2
I 0.0 0.4 0.6
D 0.0 0.0 1.0

Markovchain  3
state t2
 A  3 - dimensional discrete Markov Chain with following states
 H I D
 The transition matrix   (by rows)  is defined as follows
    H   I   D
H 0.3 0.2 0.5
I 0.0 0.2 0.8
D 0.0 0.0 1.0

Markovchain  4
state t3
 A  3 - dimensional discrete Markov Chain with following states
 H I D
 The transition matrix   (by rows)  is defined as follows
  H I D
H 0 0 1
```

```
I 0 0 1
D 0 0 1
```

It is possible to perform direct access to `markovchainList` elements as well as determining the number of `markovchain` objects a `markovchainList` object is composed by.

```
R> mcCCRC[[1]]

state t0
 A  3 - dimensional discrete Markov Chain with following states
 H I D
 The transition matrix   (by rows)  is defined as follows
    H   I   D
H 0.7 0.2 0.1
I 0.1 0.6 0.3
D 0.0 0.0 1.0

R> dim(mcCCRC)

[1] 4
```

`markovchain` package contains some data found in literature on which discrete Markov chain models have been applied (as briefly exemplified in Section 6) Table 2 lists data set and tables bundled within the current release of the package.

Dataset	Description
`preproglucacon`	Preproglucacon gene DNA basis, Peter J. Avery and Daniel A. Henderson (1999).
`rain`	Alofi Island rains, Peter J. Avery and Daniel A. Henderson (1999).
`craigsendi`	CD4 cells count table at zero and six month, Bruce A. Craig and Arthur A. Sendi (200
`blanden`	mobility across quartiles of income in UK for 1970 cohort, Jo Blanden and Machin (20

Table 2: **markovchain** `data.frame` and `table`.

Finally, Table 3 lists the demos bundled in the package's demo directory and their description.

Dataset	Description
`examples.R`	Notable Markov chains, e.g. The Gambler Ruin chain.
`quickStart.R`	Generic examples.
`bard.R`	Structural analysis of Markov chains from Bard PPT.

Table 3: **markovchain** demos

4. Probability with markovchain objects

4.1. Structural proprieties of a Markov chain

markovchain contains functions to analyze discrete Markov chains from a probabilistic perspective. For example, methods are provided for finding stationary distributions, absorbing

and transient states. In addition Matlab listings (Feres 2007) have been translated that provide methods to find communicating classes and transient states.

Table 4 shows methods appliable on `markovchain` objects to perform probabilistic analysis.

Method	Purpose
`conditionalDistribution`	it returns the conditional distribution of the subsequent state s_j, given actua
`absorbingStates`	it returns the absorbing states of the transition matrix, if any.
`steadyStates`	it returns the vector(s) of steady state(s) in matricial form.
`transientStates`	it returns the transient states of the transition matrix, if any.
`summary`	it summarizes the statistical probabilities of a Markov chain.

Table 4: **markovchain** methods: statistical operations.

The conditional distribution of the weather states, given current day's weather is sunny, is given by following code.

```
R> conditionalDistribution(mcWeather, "sunny")

 sunny cloudy   rain
   0.7    0.2    0.1
```

The steady state(s), also known as stationary distribution(s), of the Markov chains are identified by the such described algorithm:

1. decompose the transition matrix in eigenvalues and eigenvectors.
2. consider only eigenvectors corresponding to eigenvalues equal to one.
3. normalize such eigenvalues so the sum of their components to total one.

The result is returned in matricial form.

```
R> steadyStates(mcWeather)

          sunny    cloudy      rain
[1,] 0.4636364 0.3181818 0.2181818
```

It is possible a Markov chain to have more than one stationary distribuition, as the gambler ruin example shows.

```
R> gamblerRuinMarkovChain<-function(moneyMax, prob=0.5) {
+    require(matlab)
+    matr<-zeros(moneyMax+1)
+    states<-as.character(seq(from=0, to=moneyMax, by=1))
+    rownames(matr)=states; colnames(matr)=states
+    matr[1,1]=1;matr[moneyMax+1,moneyMax+1]=1
+    for(i in 2:moneyMax)
+    {
```

```
+      matr[i,i-1]=1-prob;matr[i,i+1]=prob
+    }
+    out<-new("markovchain",
+             transitionMatrix=matr,
+             name=paste("Gambler ruin",moneyMax,"dim",sep=" ")
+             )
+    return(out)
+  }
R> mcGR4<-gamblerRuinMarkovChain(moneyMax=4, prob=0.5)
R> steadyStates(mcGR4)

     0 1 2 3 4
[1,] 1 0 0 0 0
[2,] 0 0 0 0 1
```

Any absorbing state is determined by the inspection of results returned by `steadyStates` method.

```
R> absorbingStates(mcGR4)

[1] "0" "4"

R> absorbingStates(mcWeather)

character(0)
```

Code to perform many probabilistic analyses has been developed translating Matlab listings found in Feres (2007) and `?` on which the interested reader is remaineded for full descritption of theory and algoritms.

The key function used within Feres (2007) (and **markovchains** derived functions) is `.commclassKernel`, listed below.

`.commclassKernel` function gets a transition matrix of dimension n and return a list of two items:

1. `C` an adjacency matrix showing for each state j (in the row) which states lie in the same communicating class of j (flagged with 1).
2. `v` a vector indicating whether the state j is transient (0) or not (1).

These functions are used by two other internal functions on which the `summary` method for `markovchain` objects works.

The example matrix used in the Feres (2007) paper well exemplifies the function purpose. The adjacency matrix shows which class each

```
R> P<-matlab::zeros(10)
R> P[1,c(1, 3)]=1/2;
R> P[2,2]=1/3; P[2,7]=2/3;
R> P[3,1]=1;
R> P[4,5]=1;
R> P[5,c(4, 5, 9)]=1/3;
R> P[6,6]=1;
R> P[7,7]=1/4; P[7,9]=3/4;
R> P[8,c(3, 4, 8, 10)]=1/4;
R> P[9,2]=1;
R> P[10,c(2, 5, 10)]=1/3;
R> rownames(P) <- letters[1:10]
R> colnames(P) <- letters[1:10]
R> probMc<-new("markovchain", transitionMatrix=P, name="Probability MC")
R> .commclassesKernel(P)

$C
      a     b     c     d     e     f     g     h     i     j
a  TRUE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
b FALSE  TRUE FALSE FALSE FALSE FALSE  TRUE FALSE  TRUE FALSE
c  TRUE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
d FALSE FALSE FALSE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE
e FALSE FALSE FALSE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE
f FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE
g FALSE  TRUE FALSE FALSE FALSE FALSE  TRUE FALSE  TRUE FALSE
h FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE
i FALSE  TRUE FALSE FALSE FALSE FALSE  TRUE FALSE  TRUE FALSE
j FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE

$v
    a     b     c     d     e     f     g     h     i     j
 TRUE  TRUE  TRUE FALSE FALSE  TRUE  TRUE FALSE  TRUE FALSE

R> summary(probMc)

Probability MC  Markov chain that is comprised by:
Closed classes:
a c
b g i
f
Transient classes:
d e
h
j
The Markov chain is not irreducible
The absorbing states are: f
```

All states that pertain to a transient class are defined transient and a specific method has been written to elicit them.

```
R> transientStates(probMc)

[1] "d" "e" "h" "j"
```

Listings from Feres (2007) have been adapted into `canonicForm` method that turns a Markov chain into canonic form.

```
R> probMcCanonic<-canonicForm(probMc)
R> probMc

Probability MC
 A  10 - dimensional discrete Markov Chain with following states
 a b c d e f g h i j
 The transition matrix   (by rows)  is defined as follows
    a         b    c         d         e f         g    h         i
a 0.5 0.0000000 0.50 0.0000000 0.0000000 0 0.0000000 0.00 0.0000000
b 0.0 0.3333333 0.00 0.0000000 0.0000000 0 0.6666667 0.00 0.0000000
c 1.0 0.0000000 0.00 0.0000000 0.0000000 0 0.0000000 0.00 0.0000000
d 0.0 0.0000000 0.00 0.0000000 1.0000000 0 0.0000000 0.00 0.0000000
e 0.0 0.0000000 0.00 0.3333333 0.3333333 0 0.0000000 0.00 0.3333333
f 0.0 0.0000000 0.00 0.0000000 0.0000000 1 0.0000000 0.00 0.0000000
g 0.0 0.0000000 0.00 0.0000000 0.0000000 0 0.2500000 0.00 0.7500000
h 0.0 0.0000000 0.25 0.2500000 0.0000000 0 0.0000000 0.25 0.0000000
i 0.0 1.0000000 0.00 0.0000000 0.0000000 0 0.0000000 0.00 0.0000000
j 0.0 0.3333333 0.00 0.0000000 0.3333333 0 0.0000000 0.00 0.0000000
          j
a 0.0000000
b 0.0000000
c 0.0000000
d 0.0000000
e 0.0000000
f 0.0000000
g 0.0000000
h 0.2500000
i 0.0000000
j 0.3333333

R> probMcCanonic

Probability MC
 A  10 - dimensional discrete Markov Chain with following states
 a c b g i f d e h j
 The transition matrix   (by rows)  is defined as follows
```

```
    a    c         b         g         i f         d         e    h
a 0.5 0.50 0.0000000 0.0000000 0.0000000 0 0.0000000 0.0000000 0.00
c 1.0 0.00 0.0000000 0.0000000 0.0000000 0 0.0000000 0.0000000 0.00
b 0.0 0.00 0.3333333 0.6666667 0.0000000 0 0.0000000 0.0000000 0.00
g 0.0 0.00 0.0000000 0.2500000 0.7500000 0 0.0000000 0.0000000 0.00
i 0.0 0.00 1.0000000 0.0000000 0.0000000 0 0.0000000 0.0000000 0.00
f 0.0 0.00 0.0000000 0.0000000 0.0000000 1 0.0000000 0.0000000 0.00
d 0.0 0.00 0.0000000 0.0000000 0.0000000 0 0.0000000 1.0000000 0.00
e 0.0 0.00 0.0000000 0.0000000 0.3333333 0 0.3333333 0.3333333 0.00
h 0.0 0.25 0.0000000 0.0000000 0.0000000 0 0.2500000 0.0000000 0.25
j 0.0 0.00 0.3333333 0.0000000 0.0000000 0 0.0000000 0.3333333 0.00
          j
a 0.0000000
c 0.0000000
b 0.0000000
g 0.0000000
i 0.0000000
f 0.0000000
d 0.0000000
e 0.0000000
h 0.2500000
j 0.3333333
```

The function `is.accessible` permits to investigate whether a state j is accessible from state i, that is whether the probability to eventually reach j starting from i is greater than zero.

```
R> is.accessible(object=probMc, from="a",to="c")

[1] TRUE

R> is.accessible(object=probMc, from="g",to="c")

[1] FALSE
```

The example Markov chain found in mathematica web site Wolfram Research has been used, that Figure 2 plots.

```
R> require(matlab)
R> mathematicaMatr<-zeros(5)
R> mathematicaMatr[1,]<-c(0, 1/3, 0, 2/3, 0)
R> mathematicaMatr[2,]<-c(1/2, 0, 0, 0, 1/2)
R> mathematicaMatr[3,]<-c(0, 0, 1/2, 1/2, 0)
R> mathematicaMatr[4,]<-c(0, 0, 1/2, 1/2, 0)
R> mathematicaMatr[5,]<-c(0, 0, 0, 0, 1)
R> statesNames<-letters[1:5]
R> mathematicaMc<-new("markovchain",transitionMatrix=mathematicaMatr,name="Mathematica MC"
```

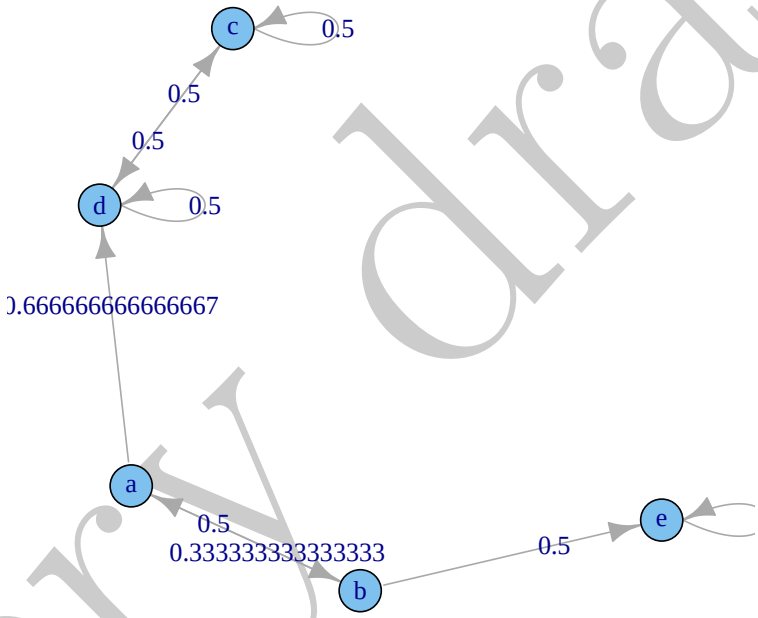

Figure 2: Mathematica 9 Markov chain illustrative example

```
Mathematica MC  Markov chain that is comprised by:
Closed classes:
c d
e
Transient classes:
a b
The Markov chain is not irreducible
The absorbing states are: e
```

4.2. First passage time

Feres (2007) provides code to compute first passage time (within $1, 2, \ldots, n$ steps) given the initial state to be i. The Matlab listings as ported in R on which the `firstPassage` function is based are

```
R> .firstpassageKernel<-function(P,i,n){
+    G<-P
+    H<-P[i,]
+    E<-1-diag(size(P)[2])
+    for (m in 2:n) {
+      G<-P%*%(G*E)
+      H<-rbind(H,G[i,])
+    }
+    return(H)
+  }
```

It is therefore easy to compute the probability that the first rainy day to be the third, given that the current state is sunny.

```
R> firstPassagePdF<-firstPassage(object=mcWeather,state="sunny",n=10)
R> firstPassagePdF[3,3]

[1] 0.121
```

5. Statistical analysis

Table 5 lists functions and methods as implemented within the package that helps to fit, simulate and predict Markov chains in the discrete time.

5.1. Simulation

Simulating a random sequence from an underlying Markov chain is quite easy thanks to the function `rmarkovchain`. The following code generates a "year" of weather states according to **?** underlying markovian stochastic process.

Function	Purpose
`markovchainFit`	function to return fitten markov chain for a given sequence.
`rmarkovchain`	function to sample from `markovchain` or `markovchainList` objects.
`predict`	method to calculate predictions from `markovchain` or `markovchainList` objects

Table 5: **markovchain** statistical functions.

```
R> weathersOfDays<-rmarkovchain(n=365,object=mcWeather,t0="sunny")
R> weathersOfDays[1:30]

 [1] "sunny"  "sunny"  "sunny"  "rain"   "sunny"  "sunny"  "sunny"
 [8] "rain"   "rain"   "rain"   "rain"   "rain"   "cloudy" "cloudy"
[15] "sunny"  "rain"   "rain"   "rain"   "cloudy" "rain"   "rain"
[22] "cloudy" "cloudy" "cloudy" "cloudy" "rain"   "cloudy" "cloudy"
[29] "cloudy" "cloudy"
```

Similarly, it is possible to simulate one o more sequence from a non-homogeneous Markov chain, as the following code (applied on CCHC example) exemplifies.

```
R> patientStates<-rmarkovchain(n=5, object=mcCCRC,t0="H",include.t0=TRUE)
R> patientStates[1:10,]

   iteration values
1          1      H
2          1      I
3          1      D
4          1      D
5          1      D
6          2      H
7          2      D
8          2      D
9          2      D
10         2      D
```

5.2. Estimation

A time homogeneous Markov chain can be fit can be fit from given data. Three methods have been implemented within current version of **markovchain** package: maximum likelihood, maximum likelihood with Laplace smoothing, Bootstrap approach.

Equation 11 shows the maximum likelihood estimate (MLE) of the p_{ij} entry, where the n_{ij} element consists in the number sequences $(X_t = i, X_{t+1} = j)$ found in the sample

$$\hat{p}_{ij}^{MLE} = \frac{n_{ij}}{\sum_{u=1}^{k} n_{iu}} \tag{11}$$

```
R> weatherFittedMLE<-markovchainFit(data=weathersOfDays, method="mle",name="Weather MLE")
R> weatherFittedMLE$estimate

Weather MLE
 A  3 - dimensional discrete Markov Chain with following states
 cloudy rain sunny
 The transition matrix   (by rows)  is defined as follows
          cloudy      rain     sunny
cloudy 0.4736842 0.3007519 0.2255639
rain   0.4791667 0.3645833 0.1562500
sunny  0.1777778 0.1555556 0.6666667
```

The Laplace smoothing approach is a variation of the MLE one where the n_{ij} is substituted by $n_{ij} + \alpha$ as Equation 12 shows, being α a positive stabilizing parameter judgmentally selected.

$$\hat{p}_{ij}^{LS} = \frac{n_{ij} + \alpha}{\sum_{u=1}^{k} (n_{iu} + \alpha)} \tag{12}$$

```
R> weatherFittedLAPLACE<-markovchainFit(data=weathersOfDays, method="laplace",laplacian=0.
R> weatherFittedLAPLACE$estimate

Weather LAPLACE
 A  3 - dimensional discrete Markov Chain with following states
 cloudy rain sunny
 The transition matrix   (by rows)  is defined as follows
          cloudy      rain     sunny
cloudy 0.4736526 0.3007592 0.2255882
rain   0.4791211 0.3645736 0.1563053
sunny  0.1778123 0.1555951 0.6665926
```

Both MLE and Laplace approach are based on the `createSequenceMatrix` functions that converts a data (character) sequence into a contingency table showing the $(X_t = i, X_{t+1} = j)$ distribution within the sample, as code below shows.

```
R> createSequenceMatrix(stringchar = weathersOfDays)

       cloudy rain sunny
cloudy     63   40    30
rain       46   35    15
sunny      24   21    90
```

An issue occurs when the sample cointain only one realization of a state (say X_β) that is located at the end of the data sequence (thanks Michael Cole for having noticed it), since it yields to a row of zero (no sample to estimate the conditional distribution of the transition). In this case the estimated transition matrix is "sanitized" assuming $p_{\beta,j} = 1/k$ being k the possible states.

A bootstrap estimation approach has been developed within the package in order to provide an indication of the variability of $\hat{p}_{ij}$ estimates. The bootstrap approach implemented within the **markovchain** package follows these steps:

1. bootstrap the data sequences following the conditional distributions of states estimated from the original one. The default bootstrap samples is 10, as specified in `nboot` parameter of `markovchainFit` function.
2. apply MLE estimation on bootstrapped data sequences that are saved in `bootStrapSamples` slot of the returned list.
3. the ${p^{BOOTSTRAP}}_{ij}$ is the average of all ${p^{MLE}}_{ij}$ across the `bootStrapSamples` list, row normalized. A `standardError` of ${p^{\hat{MLE}}}_{ij}$ estimate is provided as well.

```
R> weatherFittedBOOT<-markovchainFit(data=weathersOfDays, method="bootstrap",nboot=100)
R> weatherFittedBOOT$estimate

BootStrap Estimate
 A  3 - dimensional discrete Markov Chain with following states
 1 2 3
 The transition matrix   (by rows)  is defined as follows
          1         2         3
1 0.4743254 0.2987720 0.2269026
2 0.4897790 0.3650753 0.1451457
3 0.1780357 0.1566469 0.6653174

R> weatherFittedBOOT$standardError

           [,1]       [,2]       [,3]
[1,] 0.04138949 0.04098283 0.03968799
[2,] 0.05036841 0.04419844 0.03778620
[3,] 0.03629605 0.03132365 0.04545000
```

5.3. Prediction

n-step predictions can be obtained using the predict methods explicitely written for `markovchain` and `markovchainList` objects. The prediction is the mode of the conditional distribution of X_{t+1} given $X_t = s$, where s is the last realization of the Markov chains (homogeneous or non-homogeneous).

Predicting from a markovchain object

3-days forward predictions from `markovchain` object can be generated as it follows, assuming last two days were respectively "cloudy" and "sunny".

```
R> predict(object=weatherFittedMLE$estimate,newdata=c("cloudy","sunny"),n.ahead=3)
```

```
[1] "sunny" "sunny" "sunny"
```

Predicting from a markovchainList object

Given an initial two year (H)ealty status, the 5-year ahead prediction of any CCRC guest is

```
R> predict(mcCCRC,newdata=c("H","H"),n.ahead=5)
```

```
[1] "H" "D" "D"
```

The prediction has been stopped at time sequence since the underlying non-homogeneous Markov chain has a length of four. In order to continue five years ahead, a small change to the code has to be set.

```
R> predict(mcCCRC,newdata=c("H","H"),n.ahead=5, continue=TRUE)
```

```
[1] "H" "D" "D" "D" "D"
```

6. Applications

6.1. Actuarial examples

Markov chains are widely applied in the fields of actuarial science. Two classical applications are: bonus-malus class distribution in a Motor Third Party Liability (MTPL) portfolio (see Section **??**) and Health Insurance pricing and reserving (see Section 6.1.2)

MPTL Bonus Malus

Bonus Malus (BM) contracts grant the policyholder a discount (enworsen) as a function of the number of claims in the experience period. The discount (enworsen) is applied on a premium that already allows for known (a priori) policyholder characteristics (**?**). It tipically depends by vehicle, territory, the demographic profile of the policyholder, and policy coverages dept (deductible and policy limits).
Since the proposed BM level depends by the claim on the previous period, it can be modelled by a discrete Markov chain. A very simplified example follows. Assumed a BM scale from 1 to 5, being 4 the starting level. The evolution rules are shown by Equation 13.

$$bm_{t+1} = \max\left(1, bm_t - 1\right) * \left(\tilde{N} = 0\right) + \min\left(5, bm_t + 2 * \tilde{N}\right) * \left(\tilde{N} \geq 1\right) \tag{13}$$

Clearly $\tilde{N}$, the number of claim, is a random - variable that is assumed to be Poisson distributed.

```
R> getBonusMalusMarkovChain<-function(lambda)
+  {
```

```
+           bmMatr<-zeros(5)
+           bmMatr[1,1]<-dpois(x=0,lambda)
+           bmMatr[1,3]<-dpois(x=1,lambda)
+           bmMatr[1,5]<-1-ppois(q=1,lambda)
+
+           bmMatr[2,1]<-dpois(x=0,lambda)
+           bmMatr[2,4]<-dpois(x=1,lambda)
+           bmMatr[2,5]<-1-ppois(q=1,lambda)
+
+           bmMatr[3,2]<-dpois(x=0,lambda)
+           bmMatr[3,5]<-1-dpois(x=0,lambda)
+
+           bmMatr[4,3]<-dpois(x=0,lambda)
+           bmMatr[4,5]<-1-dpois(x=0,lambda)
+           bmMatr[5,4]<-dpois(x=0,lambda)
+           bmMatr[5,5]<-1-dpois(x=0,lambda)
+           stateNames<-as.character(1:5)
+           out<-new("markovchain",transitionMatrix=bmMatr, states=stateNames, name="BM Mat
+           return(out)
+  }
R>
```

Assuming that the a-priori claim frequency per car-year to be 0.05 in the class (being the class the group of policyholders that share the same common characteristics) the underlying BM transition matrix and its underlying steady state

```
R> bmMc<-getBonusMalusMarkovChain(0.05)
R> as.numeric(steadyStates(bmMc))

[1] 0.895836079 0.045930498 0.048285405 0.005969247 0.003978772
```

If the underlying BM coefficient of the class are 0.5, 0.7, 0.9,1.0,1.25 this means the average BM coefficient applied on the long run to the class to be.

```
R> sum(as.numeric(steadyStates(bmMc))*c(0.5,0.7,0.9,1,1.25))

[1] 0.534469
```

This means that the average premium almost halves in the long run.

Health insurance example

Actuaries quantify the risk inherent in insurance contracts evaluating the premium of insurance contract to be sold (therefore covering future risk) and evaluating the actuarial reseves of existing portfolios (the liabilities in terms of benefits or claims payments due to policyholder arising from previously sold contracts).
Key quantities of actuarial interest are: the expected present value of future benefits, $PVFB$,

the (periodic) benefit premium, P, and the present value of future premium $PVFP$. A level benefit premium could be set equating at the beginning of the contract $PVFB = PVFP$. After the beginning of the contract the benefit reserve is the differenbe between $PVFB$ and $PVFP$. The first example shows the pricing and reserving of a (simple) health insurance contract. The second example analyze the evolution of a MTPL portfolio characterized by Bonus Malus experience rating feature. The example comes from Deshmukh (2012). The interest rate is 5%, benefits are payable upon death (1000) and disability (500). Premiums are payable at the beginning of period only if policyholder is active. The contract term is three years

```
R> mcHI=new("markovchain", states=c("active", "disable", "withdrawn", "death"),
+           transitionMatrix=matrix(c(0.5,.25,.15,.1,
+                                     0.4,0.4,0.0,.2,
+                                     0,0,1,0,
+                                     0,0,0,1), byrow=TRUE, nrow=4))
R> benefitVector=as.matrix(c(0,0,500,1000))
R>
```

The policyholders is active at T_0. Therefore the expected states at $T_1, \dots T_3$ are calculated as shown.

```
R> T0=t(as.matrix(c(1,0,0,0)))
R> T1=T0*mcHI
R> T2=T1*mcHI
R> T3=T2*mcHI
```

Therefore the present value of future benefit at T0 is

```
R> PVFB=T0%*%benefitVector*1.05^-0+T1%*%benefitVector*1.05^-1+
+                 T2%*%benefitVector*1.05^-2+T3%*%benefitVector*1.05^-3
```

and the yearly premium payable whether the insured is alive is

```
R> P=PVFB/(T0[1]*1.05^-0+T1[1]*1.05^-1+T2[1]*1.05^-2)
```

The reserve at the beginning of year two, in case of the insured being alive, is

```
R> PVFB=(T2%*%benefitVector*1.05^-1+T3%*%benefitVector*1.05^-2)
R> PVFP=P*(T1[1]*1.05^-0+T2[1]*1.05^-1)
R> V=PVFB-PVFP
R> V

         [,1]
[1,] 300.2528
```

6.2. Weather forecasting

A traditional application of Markov chains lies in weather forecasting. Markov chains provide a simple model to predict the nexth day's weather given the current meteorological condition. Two example will be shown: the "Land of Oz"
The first application herewith shown is the "Land of Oz" in Section 6.2.1, taken from J. G. Kemeny, J. L.Snell, and G. L. Thompson (1974) and "Alofi Island Rainfall" in Section 6.2.2, taken from Peter J. Avery and Daniel A. Henderson (1999).

Land of Oz

According to the example, the Land of Oz is acknowledged not to have ideal weather conditions at all: the weather is snowy or rainy very often and, once more, there are never two nice days in a row. Consider three weather states: rainy, nice and snowy. Let the transition matrix be

```
R> mcWP=new("markovchain", states=c("rainy", "nice", "snowy"),
+           transitionMatrix=matrix(c(0.5, 0.25, 0.25,
+                                     0.5, 0, 0.5,
+                                     0.25,0.25,0.5), byrow=TRUE, nrow=3))
```

Given that today it's a nice day, the corresponding stochastic row vector is $w_0 = (0\,1\,0)$ and the forecast after 1, 2 and 3 days are

```
R> W0=t(as.matrix(c(0,1,0)))
R> W1=W0*mcWP
R> W1

     rainy nice snowy
[1,]   0.5    0   0.5

R> W2=W0*(mcWP^2)
R> W2

     rainy nice snowy
[1,] 0.375 0.25 0.375

R> W3=W0*(mcWP^3)
R> W3

       rainy   nice   snowy
[1,] 0.40625 0.1875 0.40625
```

As can be seen from w_1, in the Land of Oz if today is a nice day tomorrow it will rain or snow. One week later, furtherly, the prediction is

```
R> W7<-W0*(mcWP^7)
R> W7
```

```
         rainy      nice     snowy
[1,] 0.4000244 0.1999512 0.4000244
```

The steady state of the chain can be computed as

```
R> q<-steadyStates(mcWP)
R> q
```

```
     rainy nice snowy
[1,]   0.4  0.2   0.4
```

Note that from the seventh day on, the predicted probabilities are substantially equals to the steady state of the chain and don't depend from the starting point. In fact, if we start from a rainy or a snowy day we equally get

```
R> R0<-t(as.matrix(c(1,0,0)))
R> R7<-W0*(mcWP^7)
R> R7
```

```
         rainy      nice     snowy
[1,] 0.4000244 0.1999512 0.4000244
```

```
R> S0<-t(as.matrix(c(0,0,1)))
R> R7<-W0*(mcWP^7)
R> R7
```

```
         rainy      nice     snowy
[1,] 0.4000244 0.1999512 0.4000244
```

Alofi Island Rainfall

The example is taken from Peter J. Avery and Daniel A. Henderson (1999). Alofi Island daily rainfall data were recorded from January 1st, 1987 until December 31st, 1989 and classified into three states: "0", no rain, "1-5", from non zero until 5 mm, "6+" over than 5mm. Corresponding dataset is provided within the **markovchain** package.

```
R> data(rain, package="markovchain")
R> table(rain$rain)
```

```
  0 1-5  6+
548 295 253
```

The underlying transition matrix is estimated as it follows

```
R> mcAlofi<-markovchainFit(data=rain$rain, name="Alofi MC")$estimate
R> mcAlofi
```

```
Alofi MC
 A  3 - dimensional discrete Markov Chain with following states
 0 1-5 6+
 The transition matrix   (by rows)  is defined as follows
           0       1-5        6+
0   0.6605839 0.2299270 0.1094891
1-5 0.4625850 0.3061224 0.2312925
6+  0.1976285 0.3122530 0.4901186
```

from which the long term daily rainfall distribution can be obtained

```
R> steadyStates(mcAlofi)

             0       1-5        6+
[1,] 0.5008871 0.2693656 0.2297473
```

6.3. Finance, Economics and Social Science

6.4. Finance

Credit ratings transitions have been successfully modelled with discrete time Markov chains. Some rating agencies publish transition matrices that show the empirical transition probabilities across credit ratings. The example that follows is taken from **CreditMetrics** R package (**?**), carring a & Poors published data.

```
R> rc <- c("AAA", "AA", "A", "BBB", "BB", "B", "CCC", "D")
R> creditMatrix <- matrix(c(90.81, 8.33, 0.68, 0.06, 0.08, 0.02, 0.01, 0.01,
+  0.70, 90.65, 7.79, 0.64, 0.06, 0.13, 0.02, 0.01,
+  0.09, 2.27, 91.05, 5.52, 0.74, 0.26, 0.01, 0.06,
+  0.02, 0.33, 5.95, 85.93, 5.30, 1.17, 1.12, 0.18,
+  0.03, 0.14, 0.67, 7.73, 80.53, 8.84, 1.00, 1.06,
+  0.01, 0.11, 0.24, 0.43, 6.48, 83.46, 4.07, 5.20,
+  0.21, 0, 0.22, 1.30, 2.38, 11.24, 64.86, 19.79,
+  0, 0, 0, 0, 0, 0, 0, 100
+  )/100, 8, 8, dimnames = list(rc, rc), byrow = TRUE)
```

It is therefore easy to convert such matrices in `markovchain` objects and to perform some analyses

```
R> creditMc<-new("markovchain",transitionMatrix=creditMatrix, name="S&P Matrix")
R> absorbingStates(creditMc)

[1] "D"
```

Economics

A dynamic system generates two kinds of economic effects (Bard 2000):

1. those incurred when the system is in a specified state, and
2. those incurred when the system makes a transition from one state to another.

Let the monetary amount of being in a particular state to be represented as a m-dimensional column vector c_s, whilst let the monetary amount of a transition to be embodied in a C^R matrix where each component specifies the monetary amount of going from state i to state j in a single step. Henceforth Equation 14 represents the monetary of being in state i.

$$c_i = c_i^{\mathrm{S}} + \sum_{j=1}^{m} C_{ij}^{\mathrm{R}} p_{ij} \tag{14}$$

Let $\bar{C} = [c_i]$ and let e_i the vector valued 1 in the initial state and 0 in all other, then, if f_n is the random variable representing the economic return associated with the stochastic process at time n is expressed by Equation 15.

$$E\left[f_n\left(X_n\right)|X_0 = i\right] = e_i P^n \bar{c} \tag{15}$$

The following example assumes a telephone company to model transition probabilities between customer / non customer status by matrix P and the cost associated to states to be modelled by matrix M

If the average revenue for existing customer is +100, the cost per state is:

For an existing customer, the expected gain (loss) at the fifth year is:

```
R> as.numeric((c(1,0)*mcP^5)%*%(as.vector(c(c1,c2))))

[1] 48.96009
```

Social Science

Markov chains have been actively used to model progression and regressions between social classes. The first study is Glass and Hall (1954), whilst a more recent application can be found in Jo Blanden and Machin (2005). The table that follows shows the income quartile of the father when the son was 16 (in 1984) and the income quartile of the son when aged 30 (in 2000), for the 1970 cohort.

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

Unnamed Markov chain
 A  4 - dimensional discrete Markov Chain with following states
 Bottom 2nd 3rd Top
 The transition matrix   (by rows)  is defined as follows
            2nd       3rd    Bottom       Top
Bottom 0.2900000 0.2200000 0.3800000 0.1100000
```

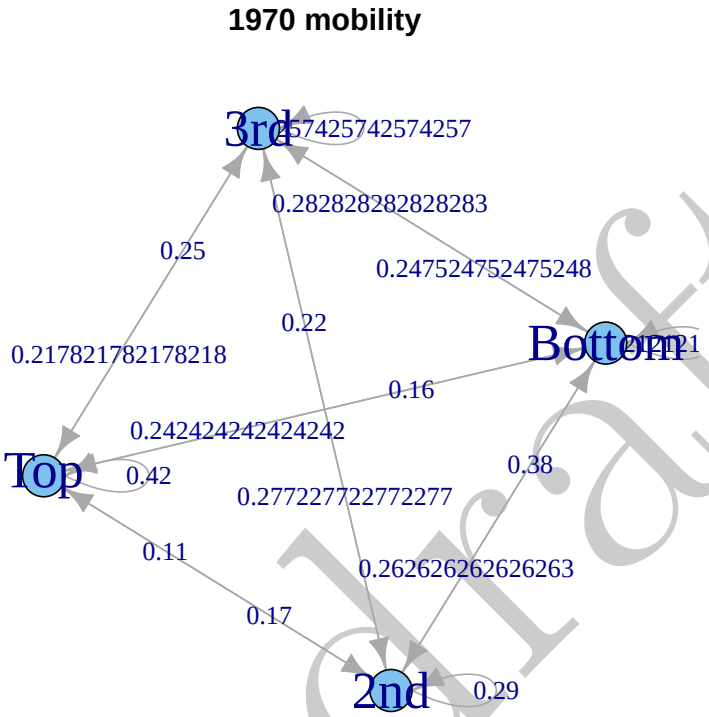

Figure 3: 1970 UK cohort mobility data

```
2nd    0.2772277 0.2574257 0.2475248 0.2178218
3rd    0.2626263 0.2828283 0.2121212 0.2424242
Top    0.1700000 0.2500000 0.1600000 0.4200000
```

The underlying transition graph is plot in Figure 3.

The steady state distribution is computed as follows. Since transition across quartiles are represented, the probability function is evenly 0.25.

```
R> round(steadyStates(mobilityMc),2)
```

```
     Bottom  2nd  3rd  Top
[1,]   0.25 0.25 0.25 0.25
```

6.5. Genetics and Medicine

This section contains two examples: the first shows the use of Markov chain models in genetics (Section 6.5.1), the second shows an application of Markov chains in modelling diseases dynamics (Section 6.5.2)

Genetics

Peter J. Avery and Daniel A. Henderson (1999) discusses the use of Markov chains in model Preprogucacon gene protein bases sequence. `preproglucacon` dataset in **markovchain** contains the dataset shown in the package.

```
R> data(preproglucacon, package="markovchain")
```

Therefore it is possible to model the transition probabilities between bases

```
R> mcProtein<-markovchainFit(preproglucacon$preproglucacon, name="Preproglucacon MC")$esti
```

Medicine

Discrete-time Markov chains are also employed to study the progression of chronic diseases. The following example is taken from Bruce A. Craig and Arthur A. Sendi (2002), in which the estimation of the monthly transition matrix is obtained in order to describe the monthly progression of CD4-cell counts of HIV infected subjects starting from six month follow-up data.
Code below shows the original data taken from the Bruce A. Craig and Arthur A. Sendi (2002) paper, from which the computation of the maximum likelihood estimate of the six month transition matrix M_6 is performed:

```
R> craigSendiMatr<-matrix(c(682,33,25,
+                 154,64,47,
+                 19,19,43), byrow=T,nrow=3)
R> hivStates<-c("0-49", "50-74", "75-UP")
R> rownames(craigSendiMatr)<-hivStates
R> colnames(craigSendiMatr)<-hivStates
R> craigSendiTable<-as.table(craigSendiMatr)
R> mcM6<-as(craigSendiTable,"markovchain")
R> mcM6@name="Zero-Six month CD4 cells transition"
R> mcM6
```

```
Zero-Six month CD4 cells transition
 A  3 - dimensional discrete Markov Chain with following states
 0-49 50-74 75-UP
 The transition matrix   (by rows)  is defined as follows
           0-49      50-74      75-UP
0-49  0.9216216 0.04459459 0.03378378
50-74 0.5811321 0.24150943 0.17735849
75-UP 0.2345679 0.23456790 0.53086420
```

As shown in the paper, the second passage consists in the decomposition of $M_6 = V * D * V^-1$ and to obtain M_1 as $M_1 = V * D^{1/6} * V^-1$

```
R> autov=eigen(mcM6@transitionMatrix)
R> D=diag(autov$values)

R> P=autov$vectors
R> P%*%D%*%solve(P)

          [,1]       [,2]       [,3]
[1,] 0.9216216 0.04459459 0.03378378
[2,] 0.5811321 0.24150943 0.17735849
[3,] 0.2345679 0.23456790 0.53086420

R> d=D^(1/6)
R> M=P%*%d%*%solve(P)
R> mcM1<-new("markovchain",transitionMatrix=M,states=hivStates)
```

7. Discussion, Issues and Future Plans

The **markovchain** has been designed in order to provide easily handling of DTMC and to be enough flexible to communicate with other packages that implement models based on DTMC.

Future versions of the package are expected to improve the code by terms of numerical accuracy and rapidity. More deep internal function profiling and integration of C++ code by means of **Rcpp** package (**?**) will be explored.

8. Aknowledgments

*

I wish to thank Michael Cole and Tobi Gutman for their suggestions and bug checkings.

References

Bard JF (2000). "Lecture 12.5 - Additional Issues Concerning Discrete-Time Markov Chains." URL http://www.me.utexas.edu/~jensen%20/ORMM/instruction/powerpoint/or_models_09/12.5_dtmc2.ppt.

Bruce A Craig, Arthur A Sendi (2002). "Estimation of the Transition Matrix of a Discrete-Time Markov Chain." *Health Economics*, **11**, 33–42.

Chambers J (2008). *Software for Data Analysis: Programming with R*. Statistics and computing. Springer-Verlag. ISBN 9780387759357.

Ching W, Ng M (2006). *Markov Chains: Models, Algorithms and Applications*. International Series in Operations Research & Management Science. Springer. ISBN 9780387293356.

Csardi G, Nepusz T (2006). "The igraph Software Package for Complex Network Research." *InterJournal*, **Complex Systems**, 1695. URL http://igraph.sf.net.

Deshmukh S (2012). *Multiple Decrement Models in Insurance: An Introduction Using R*. SpringerLink : Bücher. Springer. ISBN 9788132206590. URL http://books.google.it/books?id=L3fpKq9zT5QC.

Feres R (2007). "Notes for Math 450 Matlab Listings for Markov chains." URL http://www.math.wustl.edu/~feres/Math450Lect04.pdf.

Geyer CJ, Johnson LT (2013). *mcmc: Markov Chain Monte Carlo*. R package version 0.9-2, URL http://CRAN.R-project.org/package=mcmc.

Glass D, Hall JR (1954). "Social Mobility in Great Britain: A Study in Intergenerational Change in Status." In *Social Mobility in Great Britain*. Routledge and Kegan Paul.

Goulet V, Dutang C, Maechler M, Firth D, Shapira M, Stadelmann M, expm-developers@listsR-forgeR-projectorg (2013). *expm: Matrix exponential*. R package version 0.99-1, URL http://CRAN.R-project.org/package=expm.

J G Kemeny, J LSnell, G L Thompson (1974). *Introduction to Finite Mathematics*. Prentice Hall.

Jackson CH (2011). "Multi-State Models for Panel Data: The msm Package for R." *Journal of Statistical Software*, **38**(8), 1–29. URL http://www.jstatsoft.org/v38/i08/.

Jo Blanden PG, Machin S (2005). "Intergenerational Mobility in Europe and North America." *Technical report*, Center for Economic Performances. URL http://cep.lse.ac.uk/about/news/IntergenerationalMobility.pdf.

Nicholson W (2013). *DTMCPack: Suite of functions related to discrete-time discrete-state Markov Chains*. R package version 0.1-2, URL http://CRAN.R-project.org/package=DTMCPack.

Peter J Avery, Daniel A Henderson (1999). "Fitting Markov Chain Models to Discrete State Series." *Applied Statistics*, **48**(1), 53–61.

R Core Team (2013). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org/.

Roebuck P (2011). *matlab: MATLAB emulation package*. R package version 0.8.9, URL http://CRAN.R-project.org/package=matlab.

Snell L (1999). "Probability Book: chapter 11." URL http://www.dartmouth.edu/~chance/teaching_aids/books_articles/probability_book/Chapter11.pdf.

Spedicato GA (2013). *markovchain: an R package to easily handle discrete markov chain*. R package version 0.0.1.

Wikipedia (2013). "Markov chain — Wikipedia, The Free Encyclopedia." [Online; accessed 23-August-2013], URL http://en.wikipedia.org/w/index.php?title=Markov_chain&oldid=568910294.

Wolfram Research I (????). URL http://www.wolfram.com/mathematica/new-in-9/markov-chains-and-queues/structural-properties-of-finite-markov-processes.html.

Wolfram Research I (2013). *Mathematica*. Wolfram Research, Inc., ninth edition.

Affiliation:

Giorgio Alfredo Spedicato
StatisticalAdvisor
Via Firenze 11 20037 Italy
Telephone: +39/334/6634384
E-mail: spedygiorgio@gmail.com
URL: www.statisticaladvisor.com
Mirko Signorelli
E-mail:m.signorelli6@campus.unimib.it