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 Markov chains permit to model the transition probabilities between discrete states by the aid of matrices. Various R packages deals with Markov chains processes and their applications: **msm** (Jackson 2011) handle Multi-State Models for Panel Data, **mcmcR** (Geyer and Johnson 2013) is only one of the many package that implements Monte Carlo Markov Chain approach for estimating models' parameters, **hmm** fits hidden markov models taking into account covariates. The R statistical environment (R Core Team 2013) 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. **markovchain** package (Spedicato 2013) aims to offer greater flexibility in handling discrete time Markov chains than existing solutions. 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.

2. Markov chains mathematic revies

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 next state of X_{n+1} depends only by the current state of X_n and not by the events that preceded it.

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

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, that implies no change in the underlying transition probabilities as time goes on.

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

.

The chain moves successively 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 shown in Equation 3 is called transition probability.

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

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

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

$$P=\left[\begin{array}{ccc} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{array}\right]. \quad (4)$$

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, $x^{(2)}=x^{(0)}P^2$, $x^{(n)}=x^{(0)}P^n$, $n>0$.

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 5

$$P=\left[\begin{array}{ccc} 0.5 & 0.2 & 0.3 \\ 0.15 & 0.45 & 0.4 \\ 0.25 & 0.35 & 0.4 \end{array}\right]. \quad (5)$$

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 6 and Equation 7 hold.

$$x^{(1)}=(0\,1\,0)\left[\begin{array}{ccc} 0.5 & 0.2 & 0.3 \\ 0.15 & 0.45 & 0.4 \\ 0.25 & 0.35 & 0.4 \end{array}\right]=(0.15\,0.45\,0.4), \quad (6)$$

$$x^{(2)} = x^{(n+1)}P = (0.15\,0.45\,0.4)\begin{bmatrix} 0.5 & 0.2 & 0.3 \\ 0.15 & 0.45 & 0.4 \\ 0.25 & 0.35 & 0.4 \end{bmatrix} = (0.2425\,0.3725\,0.385) \tag{7}$$

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

Properties and classification of states

A state s_j is said to be accessible from a state s_i (written $s_i \to s_j$) if a system started in state s_i has a positive probability of transitioning into state s_j at a certain point. If both $s_i \to s_j$ and $s_j \to s_i$ the states s_i and s_j are said to communicate. A group of one or more communicating states generates a communicating class. 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.
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 $k = gcd\left\{n : Pr\left(X_n = s_i \left| X_0 = s_i \right.\right) > 0\right\}$,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 8 holds.

$$\lim_{n \to \infty} P^n = Z, \tag{8}$$

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

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") #quando viene pubblicato
R> #per ora fare il source
R> workDirGiorgio='D:/Universita/Ricerca/markovchain/'
R> workDirGiorgio2='F:\\giorgio lavoro\\universita\\markovChain'
R> #setwd(workDirGiorgio2)
R>
R> #workDirMirko='C:/Users/Mirko/Desktop/markovchain/'
R> #workDirGiorgioDropBox='D:\\Dropbox\\Dropbox\\markovchain'
R> setwd(workDirGiorgio)
R> library(expm)
R> library(igraph)
R> library(matlab)
R> source('./R Code/classesAndMethods.R')
R> source('./R Code/functions4Fitting.R')
R> source('./R Code/probabilistic.R')
```

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.

Method	Purpose
`*`	Algebraic operators on the transition matrix.
`[`	Direct access to transition matrix elements.
`==`	Equality operator on the transition matrix.
`dim`	Dimenion of the transition matrix.
`states`	Defined transition states.
`t`	Transposition operator (it switches byrow slot value and modifies the transition matrix coherent
`as`	Operator con switch from `markovchain` objects to `data.frame` objects and vice - versa.

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

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

         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
```

A 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 `plotMc` method that was based on **igraph** package (Csardi and Nepusz 2006) as Figure 1 displays.

```
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)
```

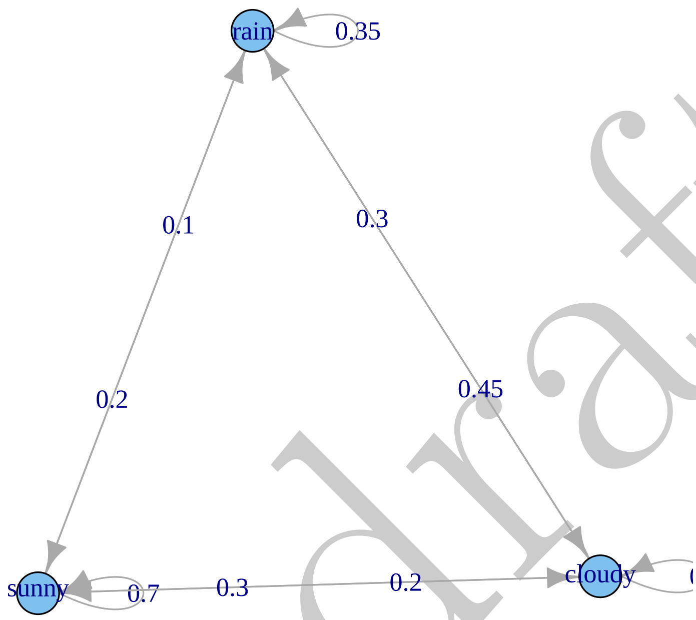

Figure 1: Weather example Markov chain plot

```
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
```

The **igraph** package (Csardi and Nepusz 2006) is used for plotting. `...` additional parameters are passed to `graph.adjacency` function to control the graph layout.

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

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

Similarly it is possible to export a `markovchain` class toward an adjacency matrix.

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 can 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 underlying `markovchain` objects contained therin in advance.

```
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
```

Finally, the `markovchain` package contains some data sets found in literature on which discrete Markov chain models have been applied. Table 2 lists data set 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).

Table 2: **markovchain** datasets.

4. Probability with markovchain objects

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 3 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.

Table 3: **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 following steps:

1. decompose the Markov chain 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)
```

The algorithm to identify transient state has been converted from Matlab listing found in Feres (2007).

```
R> transientStates(mcWeather)
```

```
character(0)
```

```
R> transientStates(mcGR4)
```

```
[1] "1" "2" "3"
```

5. Statistical analysis

Table 4 lists functions and methods as implemented within the package that helps to fit, simulate and predict Markov chains in the discrete time.

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 4: **markovchain** statistical functions.

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.

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

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

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      H
3          1      H
4          1      H
5          1      D
6          2      H
7          2      H
8          2      H
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 9 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

$$\hat{p}_{ij}^{MLE} = \frac{n_{ij}}{\sum\limits_{u=1}^{k} n_{iu}} \tag{9}$$

```
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 0.4396552 0.27586207 0.2844828
rain   0.4050633 0.41772152 0.1772152
sunny  0.2011834 0.08284024 0.7159763
```

The Laplace smoothing approach is a variation of the MLE one where the n_{ij} is substituted by $n_{ij} + \alpha$ as Equation 10 shows, being α a positive stabilizing parameter judgmentally selected.

$$\hat{p}_{ij}^{LS} = \frac{n_{ij} + \alpha}{\sum_{u=1}^{k} (n_{iu} + \alpha)} \tag{10}$$

```
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 0.4396277 0.2758769 0.2844954
rain   0.4050361 0.4176895 0.1772745
sunny  0.2012069 0.0828847 0.7159084
```

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     51   32    33
rain       32   33    14
sunny      34   14   121
```

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 to having signaled 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 0.4447934 0.27195630 0.2832503
2 0.4076541 0.41343302 0.1789129
3 0.1997245 0.08384629 0.7164292

R> weatherFittedBOOT$standardError

           [,1]       [,2]       [,3]
[1,] 0.04467520 0.04051937 0.04214391
[2,] 0.06320081 0.06234972 0.04543407
[3,] 0.02826603 0.02311234 0.03240090
```

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 11.

$$bm_{t+1} = \max\left(1, bm_t - 1\right) * \left(\tilde{N} = 0\right) + \min\left(5, bm_t + 2 * \tilde{N}\right) * \left(\tilde{N} \geq 1\right) \tag{11}$$

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. Genetics and Medicine

This section contains two examples: the first shows the use of Markov chain models in genetics (Section 6.3.1), the second shows an application of Markov chains in modelling diseases dynamics (Section 6.3.2)

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. Aknowledgments

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

References

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.

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."

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

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/`.

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/.

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

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