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

Giorgio Alfredo Spedicato, Ph.D C.Stat ACAS

Abstract

markovchain aims to fill a gap within R packages providing S4 classes and methods to easily handling discrete markov chains. 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 possible discrete states by the aid of matrices. Various R packages deals with Markov chains processes and their applications: **msm** (Jackson 2011) works with 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. R statistical environments seems to lack a simple R package that coherently defines S4 classes for discrete Markov chains and that allows the statistical analyst to perform probabilistic analysis and statistical infrence. **markovchain** (Spedicato 2013) aims to offer greater flexibility in handling discrete time Markov chains. The paper is structured as it follows: Section 2 briefly revies mathematic and definitions on discrete Markov chains, Section 4 shows applied example of discrete Markov chains in various fields.

2. Markov chains mathematic revies

A general overview of Discrete Markov chains can be found in various web sites. See for example Wikipedia (2013) and **?**.

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:

```
> #library("markovchain")
> rm(list=ls())
> workDir='D:\\Dropbox\\Dropbox\\markovchain'
> setwd(workDir)
> source('./R code/classesAndMethods.R')
> source('./R code/variousFunctions.R')
>
>
```

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

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

Slots:

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

Class "markovchainList" [in ".GlobalEnv"]

Slots:

Name:  markovchains         name
Class:         list    character
```

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

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

```
> weatherStates<-c("sunny", "cloudy", "rain")
> byRow<-TRUE
> 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))
> mcWeather<-new("markovchain",states=weatherStates, byrow=byRow,
+                transitionMatrix=weatherMatrix, name="Weather")
```

or in a shorter way, displayed below.

```
> mcWeather<-new("markovchain", states=c("sunny", "cloudy", "rain"), transitionMatrix=matr
+                        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.

```
> 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.

```
> mcList<-new("markovchainList",markovchains=list(mcWeather, defaultMc), name="A list of M
```

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.

Following methods have been implemented within the package for `markovchain` and `markovchainLists` respectively:

```
Function: * (package base)
e1="markovchain", e2="markovchain"
e1="markovchain", e2="matrix"
e1="markovchain", e2="numeric"
e1="matrix", e2="markovchain"
e1="numeric", e2="markovchain"
```

```
Function: ^ (package base)
e1="markovchain", e2="numeric"

Function: == (package base)
e1="markovchain", e2="markovchain"

Function: absorbingStates (package .GlobalEnv)
object="markovchain"

Function: coerce (package methods)
from="data.frame", to="markovchain"
from="markovchain", to="data.frame"

Function: dim (package base)
x="markovchain"

Function: initialize (package methods)
.Object="markovchain"


Function "isDiagonal":
 <not an S4 generic function>

Function "isTriangular":
 <not an S4 generic function>
Function: length (package base)

Function: plotMc (package .GlobalEnv)
object="markovchain"

Function: print (package base)
x="markovchain"

Function: show (package methods)
object="markovchain"

Function: states (package .GlobalEnv)
object="markovchain"

Function: steadyStates (package .GlobalEnv)
object="markovchain"

Function: t (package base)
x="markovchain"

Function: transitionProbability (package .GlobalEnv)
object="markovchain"
```

```
Function: initialize (package methods)
.Object="markovchainList"
    (inherited from: .Object="ANY")


Function "isDiagonal":
 <not an S4 generic function>

Function "isTriangular":
 <not an S4 generic function>
Function: length (package base)
```

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

Method	Purpose
*	Algebraic operators on the transition matrix.
==	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.

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

     sunny cloudy  rain
[1,]  0.39  0.355 0.255

> 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.

```
> initialState<-c(0,1,0)
> mcWeatherTransposed<-t(mcWeather)
```

```
> after2Days<-(mcWeatherTransposed*mcWeatherTransposed)*initialState
> after7Days<-(mcWeather^7)*initialState
> after2Days

        [,1]
sunny  0.390
cloudy 0.355
rain   0.255

> 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.

```
> states(mcWeather)

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

> dim(mcWeather)

[1] 3
```

A direct access to transition probabilities is provided by `transitionProbability` method.

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

[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.

```
> 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

> show(mcWeather)
```

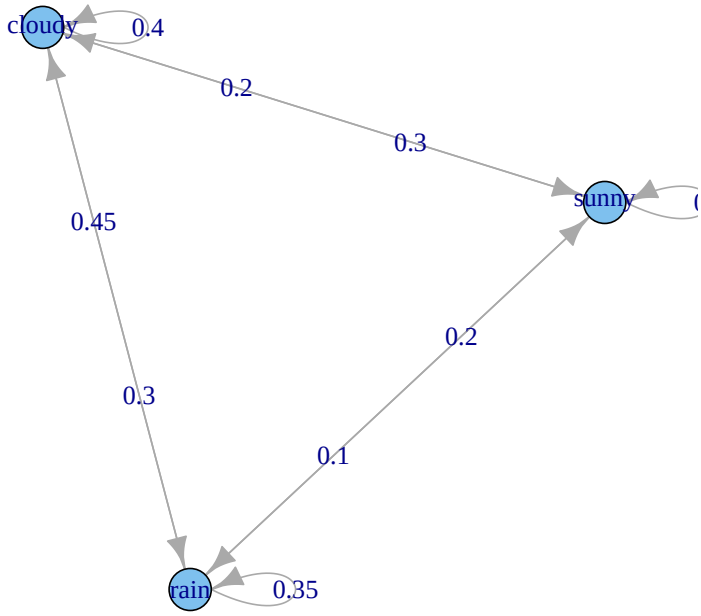

Figure 1: Weather example Markov chain plot

```
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.

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

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

3.3. Statistics with markovchain objects

Table 2 shows methods appliable on `markovchain` objects to perform probabilistic analysis.

Method	Purpose
`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.

Table 2: **markovchain** methods: statistical operations.

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

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.

```
> 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.

```
> 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)
+ }
> mcGR4<-gamblerRuinMarkovChain(moneyMax=4, prob=0.5)
> 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.

```
> absorbingStates(mcGR4)

[1] "0" "4"

> absorbingStates(mcWeather)

character(0)
```

Table 3 lists functions (and their purpose) as implemented within the package that helps to fit and simulate discrete time Markov chains.

Function	Purpose
`markovchainFit`	function to return fitten markov chain for a given sequence.
`markovchainSequence`	function to obtain a sample of the stationary process underlying the markov chai

Table 3: **markovchain** statistical functions.

Simulating a random sequence from an underlying Markov chain is quite easy thanks to the function `markovchainSequence`.

```
> weathersOfDays<-markovchainSequence(n=365,markovchain=mcWeather,t0="sunny")
> weathersOfDays

  [1] "rain"   "sunny"  "rain"   "cloudy" "rain"   "rain"   "rain"   "cloudy" "rain"
 [10] "rain"   "cloudy" "cloudy" "rain"   "rain"   "cloudy" "sunny"  "sunny"  "sunny"
 [19] "sunny"  "sunny"  "sunny"  "sunny"  "sunny"  "sunny"  "sunny"  "sunny"  "sunny"
 [28] "sunny"  "sunny"  "sunny"  "sunny"  "sunny"  "cloudy" "sunny"  "cloudy" "rain"
 [37] "rain"   "sunny"  "sunny"  "sunny"  "sunny"  "cloudy" "sunny"  "cloudy" "rain"
 [46] "cloudy" "cloudy" "rain"   "rain"   "rain"   "rain"   "cloudy" "rain"   "sunny"
 [55] "sunny"  "sunny"  "cloudy" "cloudy" "cloudy" "sunny"  "rain"   "rain"   "cloudy"
 [64] "sunny"  "sunny"  "rain"   "cloudy" "rain"   "rain"   "rain"   "rain"   "cloudy"
 [73] "cloudy" "cloudy" "cloudy" "rain"   "cloudy" "cloudy" "rain"   "sunny"  "sunny"
 [82] "sunny"  "sunny"  "cloudy" "sunny"  "cloudy" "cloudy" "cloudy" "cloudy" "cloudy"
 [91] "rain"   "cloudy" "rain"   "rain"   "cloudy" "rain"   "cloudy" "cloudy" "rain"
[100] "cloudy" "cloudy" "cloudy" "sunny"  "sunny"  "cloudy" "sunny"  "sunny"  "rain"
[109] "cloudy" "sunny"  "sunny"  "sunny"  "sunny"  "sunny"  "sunny"  "sunny"  "sunny"
[118] "sunny"  "sunny"  "sunny"  "rain"   "rain"   "cloudy" "cloudy" "sunny"  "sunny"
[127] "sunny"  "rain"   "cloudy" "sunny"  "sunny"  "sunny"  "rain"   "cloudy" "cloudy"
[136] "sunny"  "sunny"  "sunny"  "cloudy" "cloudy" "cloudy" "cloudy" "cloudy" "cloudy"
[145] "sunny"  "sunny"  "sunny"  "sunny"  "sunny"  "sunny"  "sunny"  "sunny"  "sunny"
```

```
[154] "sunny"  "sunny"  "sunny"  "sunny"  "sunny"  "sunny"  "cloudy" "cloudy" "cloudy"
[163] "cloudy" "rain"   "cloudy" "sunny"  "cloudy" "cloudy" "cloudy" "sunny"  "cloudy"
[172] "sunny"  "cloudy" "sunny"  "sunny"  "rain"   "rain"   "rain"   "sunny"  "sunny"
[181] "sunny"  "sunny"  "sunny"  "sunny"  "cloudy" "rain"   "cloudy" "sunny"  "sunny"
[190] "sunny"  "cloudy" "sunny"  "sunny"  "cloudy" "rain"   "rain"   "cloudy" "rain"
[199] "cloudy" "sunny"  "sunny"  "sunny"  "sunny"  "rain"   "cloudy" "rain"   "rain"
[208] "cloudy" "cloudy" "sunny"  "sunny"  "cloudy" "sunny"  "cloudy" "cloudy" "cloudy"
[217] "sunny"  "sunny"  "sunny"  "sunny"  "cloudy" "cloudy" "rain"   "sunny"  "sunny"
[226] "sunny"  "sunny"  "sunny"  "cloudy" "cloudy" "rain"   "sunny"  "rain"   "cloudy"
[235] "rain"   "cloudy" "rain"   "sunny"  "sunny"  "sunny"  "cloudy" "rain"   "rain"
[244] "rain"   "rain"   "rain"   "cloudy" "rain"   "sunny"  "sunny"  "sunny"  "rain"
[253] "cloudy" "sunny"  "sunny"  "sunny"  "sunny"  "sunny"  "sunny"  "sunny"  "sunny"
[262] "sunny"  "sunny"  "rain"   "sunny"  "sunny"  "sunny"  "sunny"  "sunny"  "sunny"
[271] "sunny"  "sunny"  "sunny"  "sunny"  "sunny"  "sunny"  "sunny"  "sunny"  "sunny"
[280] "sunny"  "sunny"  "sunny"  "sunny"  "sunny"  "sunny"  "sunny"  "sunny"  "sunny"
[289] "sunny"  "cloudy" "cloudy" "sunny"  "sunny"  "rain"   "cloudy" "sunny"  "sunny"
[298] "cloudy" "sunny"  "cloudy" "sunny"  "sunny"  "rain"   "cloudy" "sunny"  "sunny"
[307] "sunny"  "sunny"  "cloudy" "rain"   "sunny"  "sunny"  "sunny"  "sunny"  "sunny"
[316] "sunny"  "sunny"  "sunny"  "sunny"  "rain"   "rain"   "sunny"  "sunny"  "rain"
[325] "rain"   "cloudy" "rain"   "rain"   "cloudy" "sunny"  "rain"   "rain"   "cloudy"
[334] "rain"   "sunny"  "cloudy" "sunny"  "rain"   "cloudy" "rain"   "rain"   "rain"
[343] "cloudy" "cloudy" "rain"   "sunny"  "sunny"  "sunny"  "sunny"  "sunny"  "cloudy"
[352] "cloudy" "cloudy" "cloudy" "cloudy" "sunny"  "sunny"  "sunny"  "sunny"  "sunny"
[361] "sunny"  "sunny"  "sunny"  "sunny"  "sunny"
```

Similarly, a `markovchain` object can be fit from given data

```
> mcFitted<-markovchainFit(data=weathersOfDays, method="mle")
>
```

4. Applied examples

4.1. Actuarial examples

Markov chains are widely applied in the fields of actuarial science. 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 betwwen PVFBandPVFP.
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.

Health insurance example

The example comes from **?**. 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

```
> 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))
> benefitVector=as.matrix(c(0,0,500,1000))
>
```

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

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

Therefore the present value of future benefit at T0 is

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

and the yearly premium payable whether the insured is alive is

```
> 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

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

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

5. Aknowledgments

References

Csardi G, Nepusz T (2006). "The igraph software package for complex network research." *InterJournal*, **Complex Systems**, 1695. URL http://igraph.sf.net.

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

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/.

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.

Affiliation:

Giorgio Alfredo Spedicato
StatisticalAdvisor
Via Firenze 11 20037 Italy
Telephone: +39/334/6634384
E-mail: spedygiorgio@gmail.com
URL: www.statisticaladvisor.com