Markov Chain Monte Carlo and Bayesian Inference

Benedetto Jacopo Buratti

February 1, 2017

Abstract

This second homework covers the main concepts behind Monte Carlo Markov Chain simulation (MCMC).

Contents

Glossary

1. p for ”probability”, the cumulative distribution function (c. d. f.)
2. q for ”quantile”, the inverse c. d. f.
3. d for ”density”, the density function (p. f. or p. d. f.)
4. r for ”random”, a random variable having the specified distribution

Markov Chains Essentials

Markov Chain Definition

Markov Process are sthocastics process which are usually defined as a collection of random variables. Markov Chains are useful to model random process which has a short memory dependence.

We can have four types of Markov Chains:

Types of Markov Chains		
Time/State-Space	Countable State Space	General State Space
Discrete-Time	MC on a finite state space	MC on a general state space
Continuos-Time	Markov Process	Stochastic Process w/ Markov Property

In this case we are instrested in defining the probability law of Markov Chains on a generals state space. Let $t \in \{0, 1, 2, ...\}$ be the index of the process and $S \subset R^k$ the general state space of states:

1. $\mu \rightarrow$ initial distribution at time t = 0
2. Transition Kernel $K_t(x, A) = Pr\{X_{t+1} \in A | X_t = x\}$ for each $t = \{1, 2, ...\}$

Where the transition kernel is a function $K(\cdot, \cdot) : S \times \mathcal{B}(\mathcal{S}) \rightarrow [0, 1]$

- $\forall x \in \mathcal{S} : K(x, \cdot)$ is a probability measure
- $\forall A \in \mathcal{B}(S) : K(\cdot, A)$ is measurable

Markov Chain as an approximation tool

Markov Chains have several properties among which **Invariant Measure** and **Stationarity** at steady-state. More formally given a finite Markov Chain $X_t, t \in \mathcal{T}$ which is irreducible and poistive recurrent than after t stpes I get a random value: $\theta_t \sim P_y^t(\cdot) = K^t(y, \cdot)$. This is true thanks to the ergodic theorem, which holds in under the previous properties.

In the end we obtain $P_y^t(\cdot) \rightarrow P_y^\infty = \pi(\cdot)$ or more specifically,

$$\hat{I} = \frac{1}{t} \sum_{i=T_0}^{T_0+t} h(\theta_i) \rightarrow E_\pi[h(\theta)] = I \quad for\ t \rightarrow \infty \tag{1}$$

Given a suitable Markov Chain defined by a Markov Kernel $K(x, \cdot)$ depends on the possibility of finding a suitable π such that $X_n \sim \pi \Rightarrow X_{n+1} \sim \pi$

Thus the subsequent issue that we have to analyze how is possible to generate a stationary distribution from a Markov Chain. This requires a useful property defined as **Detailed Balance Condition** which than implies that the Markov Chain is reversible $Pr_\pi\{X_t \in A, X_{t+1} \in B\} = Pr\{X_{t+1} \in A, X_t \in B\}$ which is equivalent to $\pi(x)q(x, y) = \pi(y)q(y, x)$ given this backword transition.

At this point we should be clear the working principle of Markov Chain Monte Carlo is pretty much the same as the Vanilla Monte Carlo, with the only difference that in this case we are not generating our samples from a distribution from a closed form but from a markov chaing.

Essetially given a target density f, we build a markov kernel K with a stationary distirbution f and then generate a Markov Chain $X^{(t)}$ using this kernel so that the limiting distirbution is f and the integrals can be approximated according to the Egodic Theorem.

The issue with this approach that is we need to be able to build a kernel K that is associated with an arbitray density f. To cope with this problem there are several approaches, among which the most notable are:

- Metropolis-Hasting
- Gibbs Algorithm

Those two algorithms are crucial in order to build a kernel that is able to approximate a target function. The two algorithms are summarized in the following tables, where pros and cons are compared.

Markov Chain Generation Algorithms		
Feature/Algorithm	Metropolis-Hastings	Gibbs Sampling
Description	MC on a finite state space	MC on a general state space
Pros	Markov Process	Stochastic Process w/ Markov Property
Cons	Markov Process	Stochastic Process w/ Markov Property
Notes	Markov Process	Stochastic Process w/ Markov Property

Monte Carlo Markov Chain Essentials

Expectation Estimation via MCMC

MCMC Error Control

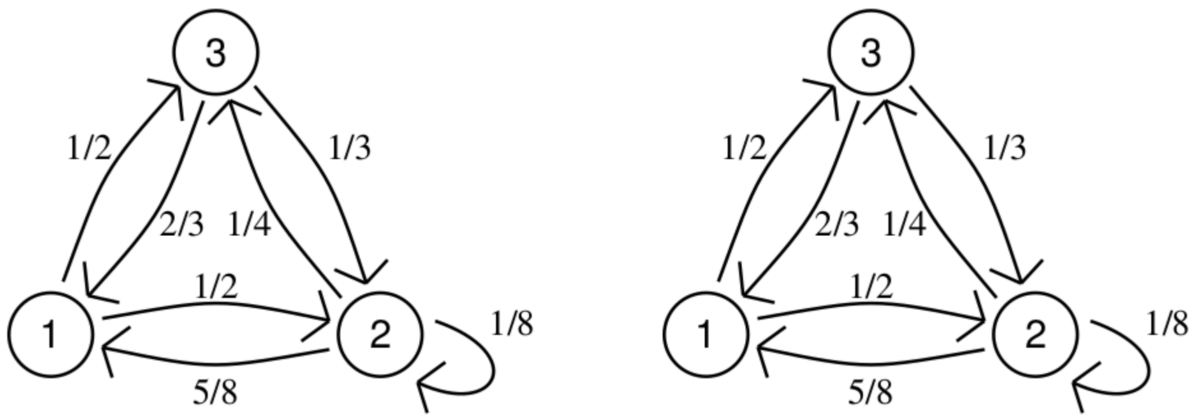


Figure 1: On the left the graphical reppresentaion of our Markov Chain and on the right its transition matrix

Puppet Markov Chain

First of all let's define the Markov Chain that we want to analyze:

First of all we define a s4 object constructor for Markov Chains and than delcare an puppet markov chain:

```
#Markov Chains Constructor

setClass("markov_chain",
         slots = list(state_space="numeric",
                      initial_state="numeric",
                      chain_size="numeric",
                      transition_matrix="matrix"))

setGeneric("MCSimulator", function(object) {
  standardGeneric("MCSimulator")
})
```

```
## [1] "MCSimulator"
```

```
setGeneric("set_initial_state", function(object, init_state ) {
  standardGeneric("set_initial_state")
})
```

```
## [1] "set_initial_state"
```

Markov Chain Simulation

We now define a function that would allows us to simulate our chain

```
#Markov Chain Simulator

setMethod("MCSimulator", signature("markov_chain"), function(object){
  markov_chain <- c(vector(), 1:(object@chain_size-1))
  markov_chain[1] = object@initial_state

  for(t in 1:(object@chain_size)){
    markov_chain[t+1] <- sample(object@state_space, size = (object@chain_size), prob=object@transition_matrix
  }
  return(markov_chain)
})
```

```
## [1] "MCSimulator"
```

```r
setMethod("set_initial_state", signature("markov_chain", "numeric"), function(object, init_state){
  object@initial_state <- init_state
  return(object)})
```

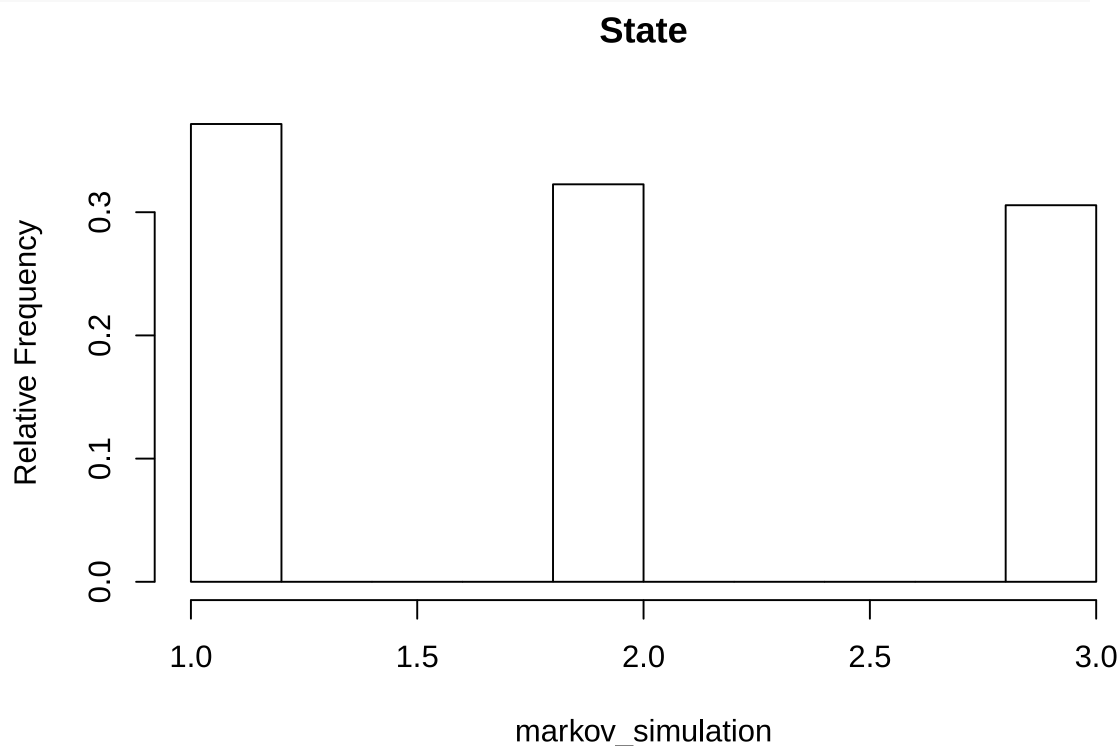
```
## [1] "set_initial_state"
```

Now that the we have all the necessary elements we can simulate our markov chain:

States Empirical Relative Frequency

At this point is pretty straight forward to printthe empirical relative frequency of the three states

```r
library(HistogramTools)
PlotRelativeFrequency(hist(markov_simulation, plot = F), main="State")
```



Simulation Repeatition

At this point we compute the simulation by calling

Invariant Distribution π (closed form)

We can compute the invariant distribution by definition, hene:

$$\pi = \begin{pmatrix} \pi_1 \\ \pi_2 \\ \pi_3 \end{pmatrix} = (\pi_1 \pi_2 \pi_3) \cdot \begin{pmatrix} 0 & \frac{1}{2} & \frac{1}{2} \\ \frac{5}{8} & \frac{1}{8} & \frac{1}{4} \;) \\ \frac{2}{3} & \frac{1}{3} & 0 \end{pmatrix} = \pi P \tag{2}$$

Which is subject to the costraint of a probability measure, indeed $\pi_1 + \pi_2 + \pi_3 = 1$ which

at this point the only thing that we have to do is to solve this system by substituting the second equation with the constraint since is linear combination of the other equations:

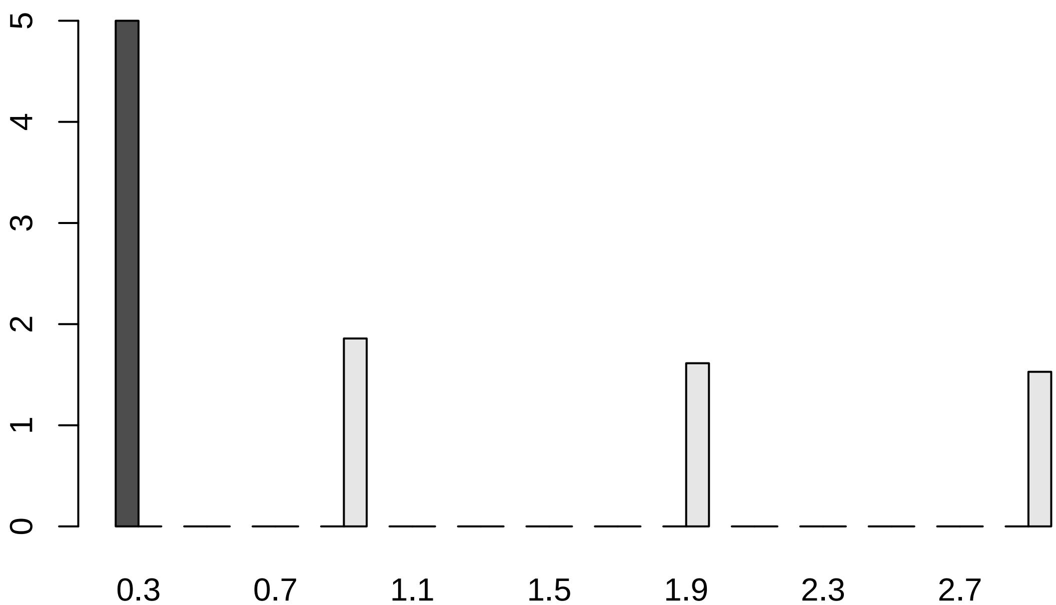
```r
library(limSolve)
A = matrix(c(-1, 1/2, 1/2, 1, 5/8, -7/8, 1/4, 1, 2/3, 1/3, -1,1), nrow = 4, ncol = 3)
b = c(0,0,0,1)
analytical_invariant_distribution <- Solve(A,b)
analytical_invariant_distribution
```

```
## [1] 0.3917526 0.3298969 0.2783505
```

Evaluation

Now that we have the true analytical solution of our Markov Chain we want to check if our simulation is indeed a good approximation of this value. In order to compare those values we compare the two gen

```r
genziana <- as.vector(table(markov_simulation)/sum(markov_simulation))
library(plotrix)
l <- list(analytical_invariant_distribution, markov_simulation)
multhist(l, beside = TRUE, freq = FALSE)
```



Initial State Independecy

If we change the initial state, since the Markov chain is irreducible and positive recurrent, we will not have any sensible difference in the final outcome for the ergodic theorem. This is can be experimentally proven by simulating the markov chain with different starting point for 500 times with different values in X_{1000}

Once again at this point we declare a new object Markov Chain:

```r
puppet_markov_chain = set_initial_state(puppet_markov_chain, 2)
x_1000 <- c()
for(i in 1:500){
  mc <- MCSimulator(puppet_markov_chain)
  x_1000[i] <- mc[1000]
}

l <- list(analytical_invariant_distribution, markov_simulation)
multhist(l, beside = TRUE, freq = FALSE)
```

Coal Mining Disaster

The Dugong Strikes back