

# 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

<b>Glossary</b>	<b>1</b>
<b>Markov Chains Essentials</b>	<b>2</b>
Markov Chain Definition . . . . .	2
Markov Chain as an approximation tool . . . . .	2
<b>Monte Carlo Markov Chain Essentials</b>	<b>4</b>
Expectation Estimation via MCMC . . . . .	4
MCMC Error Control . . . . .	4
<b>Puppet Markov Chain</b>	<b>5</b>
Markov Chain Simulation . . . . .	5
States Empirical Relative Frequency . . . . .	5
Simulation Repeatition . . . . .	6
Invariant Distribution $\pi$ (closed form) . . . . .	6
Evaluation . . . . .	6
Initial State Independency . . . . .	6
<b>Coal Mining Disaster</b>	<b>7</b>
<b>The Dugongs Strike back</b>	<b>8</b>

## 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 stochastics 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
Continuous-Time	Markov Process	Stochastic Process w/ Markov Property

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

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

- $\forall x \in S : K(x, \cdot)$  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 positive recurrent then after  $t$  steps 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,

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

Given a suitable Markov Chain defined by a Markov Kernel  $K(x, \cdot)$  depends on the possibility of finding a suitable  $\pi$  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 then 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 backward 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 chain.

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

The issue with this approach that is we need to be able to build a kernel  $K$  that is associated with an arbitrary 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

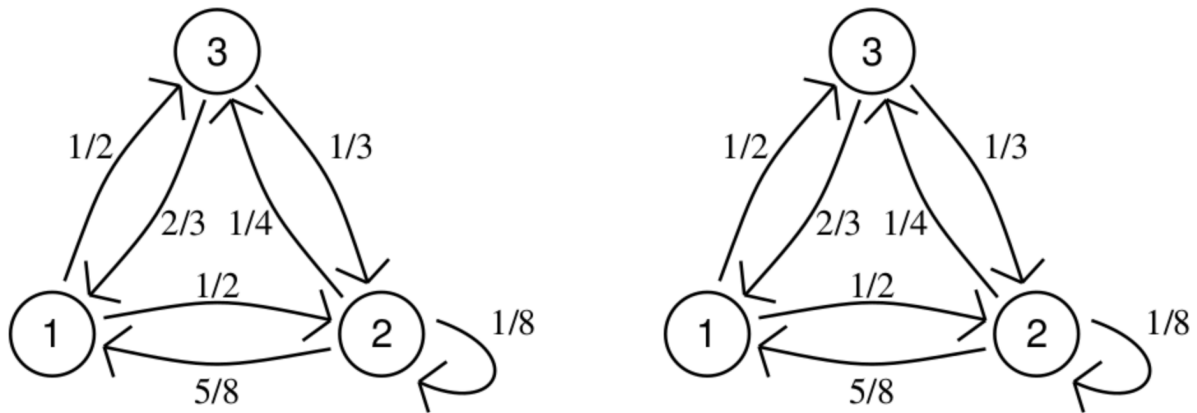


Figure 1: On the left the graphical representation of our Markov Chain and on the right its transition matrix

## 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 then declare a puppet markov chain:

```
#Markov Chains Constructor

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

## Markov Chain Simulation

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

```
#Markov Chain Simulator

setMethod("simulate", 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)
})

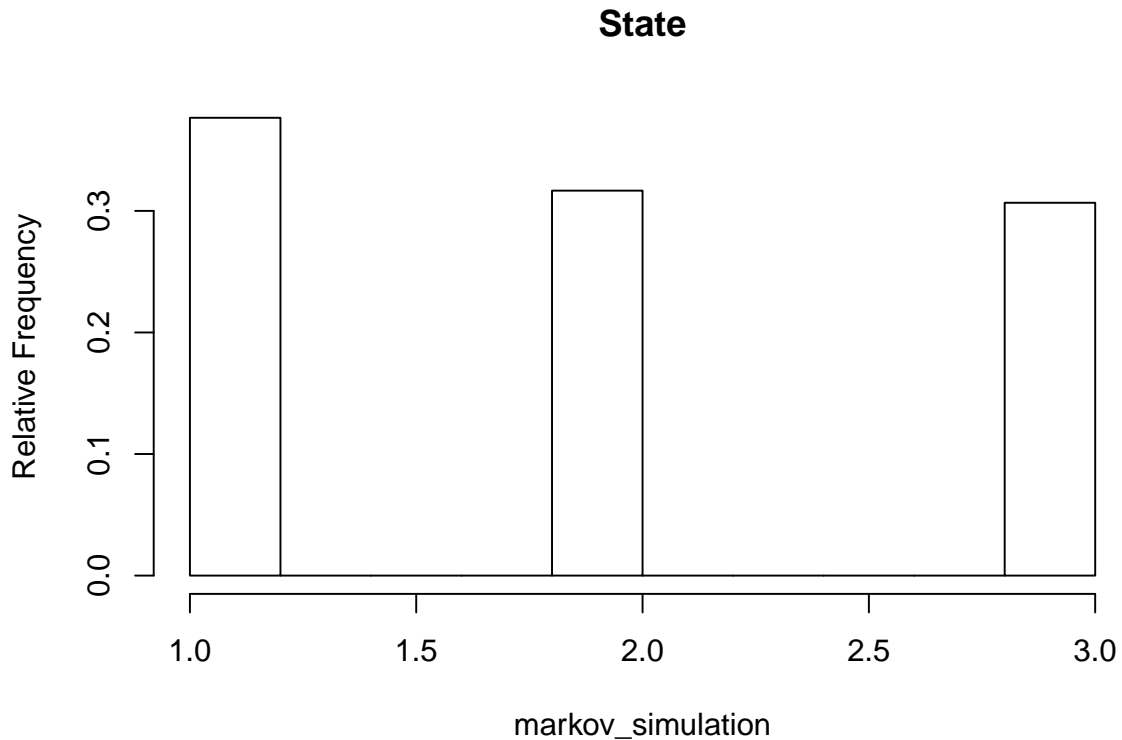
## [1] "simulate"
```

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

## States Empirical Relative Frequency

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

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



## Simulation Repeation

At this point we compute the simulation by calling

## Invariant Distribution $\pi$ (closed form)

We can compute the invariant distribution by definition, here:

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

Which is subject to the constraint 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:

```
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)
print(Solve(A,b))
```

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

## Evaluation

### Initial State Independecy

## Coal Mining Disaster

## The Dugongs Strike back