Exercise 9 - MCMC - Gibbs Sampling, Metropolis-Hastings Sampling

Konstantinos Vakalopoulos 12223236

2024-01-10

Contents

Task 1

Consider the standard normal bivariate distribution with parameters

$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \text{and} \quad \Sigma = \begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix}$$

with p.d.f. given by:

$$p(\theta_1, \theta_2) = \frac{1}{2\pi\sqrt{1-\rho^2}} e^{-\frac{1}{2(1-\rho^2)}(\theta_1^2 - 2\rho\theta_1\theta_2 + \theta_2^2)}$$

$$\text{with } \theta_1, \theta_2 \in \mathbb{R} \text{ and } \rho \in [0, 1].$$

where we also know that the marginal distributions are given by

$p(\theta_i) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}\theta_i^2}, \quad i = 1, 2.$

In what follows, consider $\rho = 0.5$, chain size $M = 30000$, and choose a non-informative prior-setting to initialize the algorithm.

Task 1.1

Implement a Gibbs sampler to sample from this distribution.

According to the instructions of the task, we have a bivariate normal distribution. Also, it is known that the marginal distribution for each parameter θ is a normal distribution with mean zero and standard deviation 1. This is proven from the probability density functions, given by task description. Furthermore, the chain size is 30000, which means that 30000 iterations are going to be performed, and ρ is equal to 0.5.

Regarding the Gibbs sampler, the algorithm for slide 31 from the lecture notes is going to be performed. However, the thing that is missing from the algorithm is the full conditional distributions. The answer to that problem, will be given from the normal distribution of each parameter θ_i. Therefore, based on 7.3.1 chapter (https://bookdown.org/rdpeng/advstatcomp/gibbs-sampler.html):

$$x_{n+1} \sim \mathcal{N}\left(\mu_1 + \rho\frac{\sigma_1}{\sigma_2}(y_n - \mu_2), \sigma_1^2(1-\rho^2)\right)$$
$$y_{n+1} \sim \mathcal{N}\left(\mu_2 + \rho\frac{\sigma_2}{\sigma_1}(x_{n+1} - \mu_1), \sigma_2^2(1-\rho^2)\right)$$
$$\text{where } \mu_1 = \mu_2 = 0, \rho = 0.5 \text{ and } \sigma_1^2 = \sigma_2^2 = 1$$

In R the Gibbs sampler will be:

```
# Gibbs Sampling
sample_theta1 <- function(theta2, rho) {
  mean_theta1 <- rho * theta2
  sd_theta1 <- sqrt(1 - rho^2)
  return(rnorm(1, mean = mean_theta1, sd = sd_theta1))
}

sample_theta2 <- function(theta1, rho) {
  mean_theta2 <- rho * theta1
  sd_theta2 <- sqrt(1 - rho^2)
  return(rnorm(1, mean = mean_theta2, sd = sd_theta2))
```

```
}
# Gibbs sampler
gibbs_sampler <- function(iterations, rho) {
  # Initialize theta and a matrix where we are going to store the samples
  theta <- c(0, 0)  # Initial values for the parameters
  samples <- matrix(0, nrow = iterations, ncol = 2)

  # Gibbs sampler
  for (i in 1:iterations) {
    # Sample Theta 1 given Theta 2
    theta[1] <- sample_theta1(theta[2], rho)

    # Sample Theta 2 given the new value of Theta 1
    theta[2] <- sample_theta2(theta[1], rho)

    # Store the sample to the matrix initialized at the beginning
    samples[i, ] <- theta
  }
  return(samples)
}
```

The function gibbs_sampler is responsible for the Gibbs sampling. It takes as input the number of iterations, which is 15000 (30000/2) in order to get an output of 30000 samples for each θ and the ρ and returns the samples for the two parameters, θ_1 and θ_2. For each iteration, the functions sample_theta1 and sample_theta2 are called and these functions are responsible for the implementation of the two formulas, which are written above. Therefore, in case the functions will be called, a seed and the proper parameters have to be set.

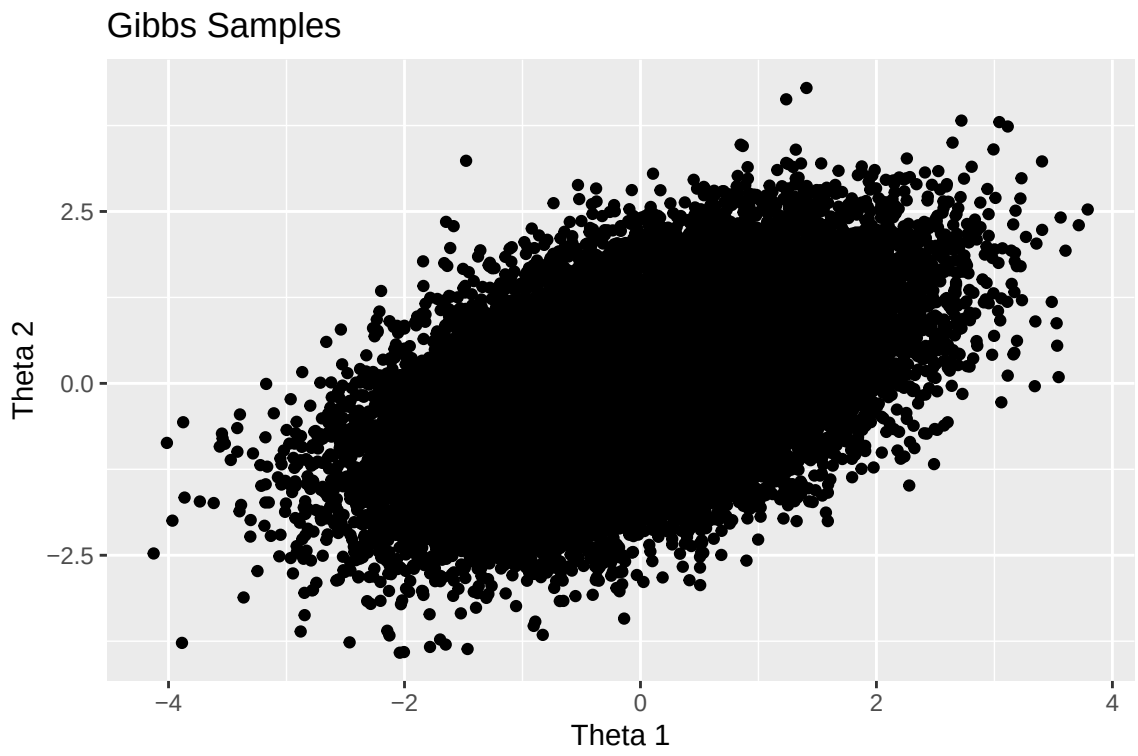
```
# Set parameters
rho <- 0.5
chain_size <- 30000

# Run Gibbs sampler
set.seed(12223236)
gibbs_samples <- gibbs_sampler(chain_size, rho)
```

The scatterplot below, shows the distribution of the generated Gibbs samples.

```
library(ggplot2)
gibbs_samples_frame <- data.frame(gibbs_samples)
```

```
ggplot(gibbs_samples_frame, aes(x = X1, y = X2)) +
  geom_point() +
  labs(title = "Gibbs Samples", x = "Theta 1", y = "Theta 2")
```



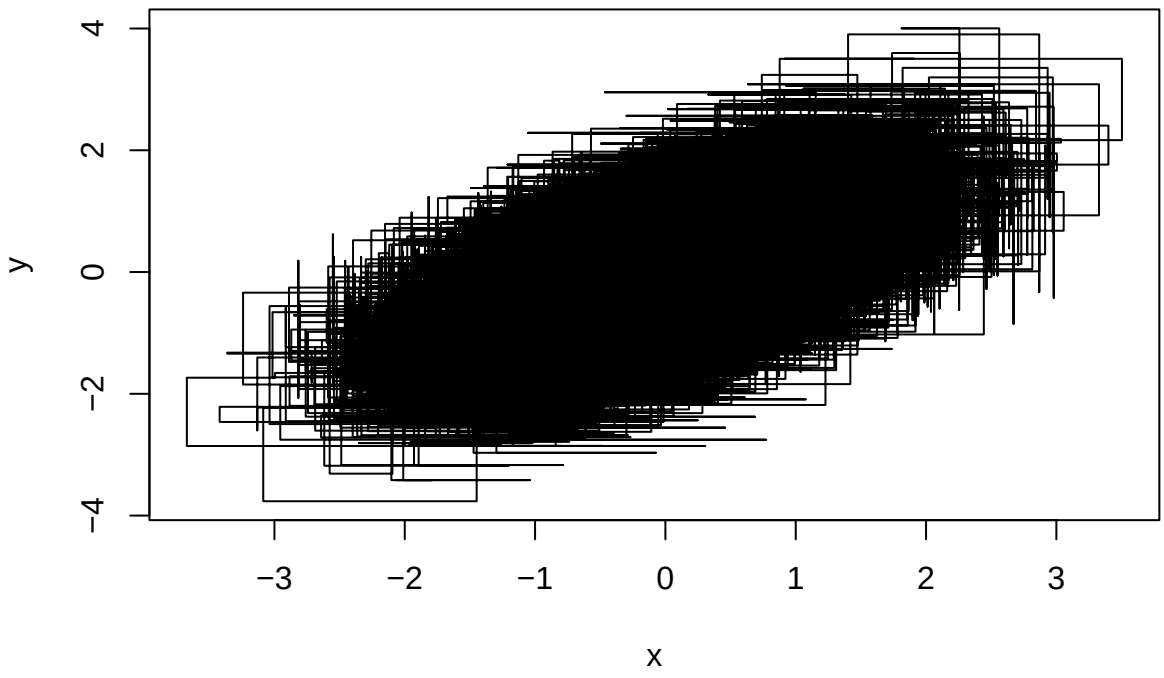
Task 1.2

Use the Metropolis–Hastings algorithm with block-wise update to simulate from this distribution.

Regarding the Metropolis–Hastings algorithm with block-wise update, two implementations are performed. In the first implementation, the function bivnormMH from the library Bolstad2 is used, which has as an input the type "block". The second implementation is based on the algorithm in slide 24 from the lecture notes.

As for the first implementation, the library Bolstad2 was installed and introduced and then the function bivnormMH was used.

```
# install.packages("Bolstad2")
library(Bolstad2)
metropolis_hastings_samples <- bivnormMH(rho=0.5,
                                         sigma = c(1,1),
                                         steps=15000,
                                         type="block")$targetSample
```

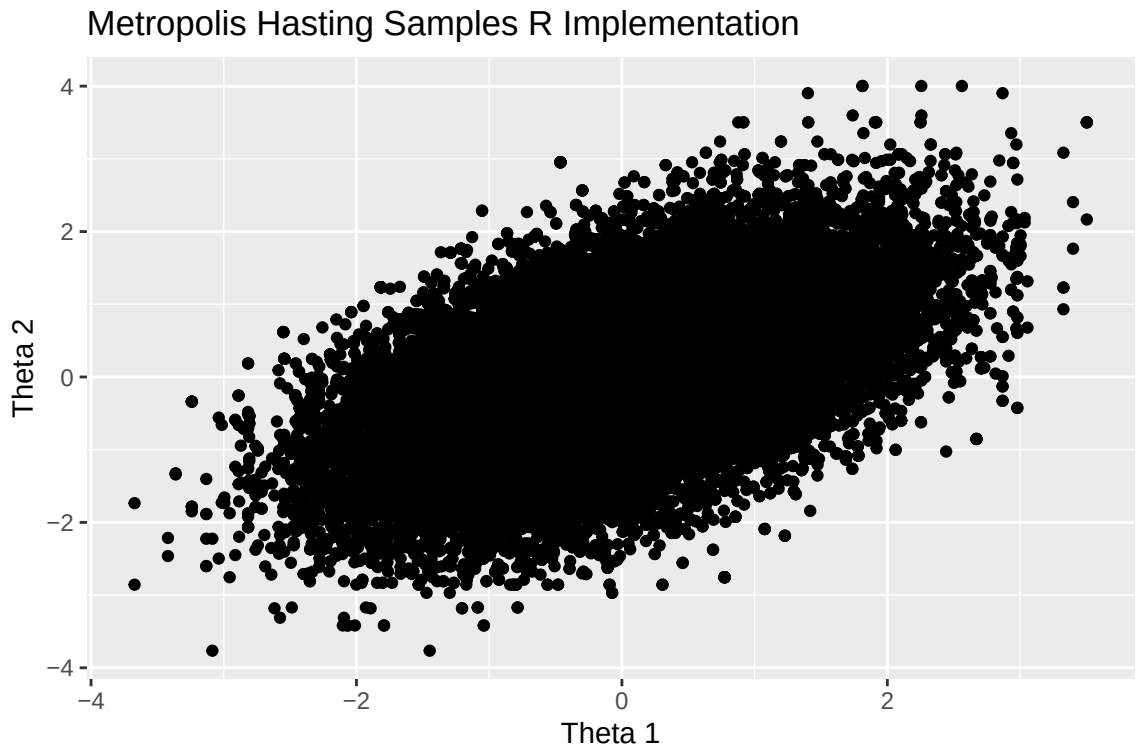



```
metropolis_hastings_samples <- data.frame(metropolis_hastings_samples)
```

As it can be seen from the function bivnormMH() the ρ is equal to 0.5, the standard deviation for θ_1 and θ_2 is 1, the steps/iterations are 30000 and the type is block, indicating the block-wise update. Also, the targetSample, in the bivnormMH function, is used in order to get the samples for the θ_1 and θ_2. Finally, the plot above is generated automatically from the function and shows the transition from one state to another due to the Markov Chain.

The scatterplot below, shows the distribution of the generated Metropolis hasting samples using the function bivnormMH.

```
ggplot(metropolis_hastings_samples, aes(x = x, y = y)) +
  geom_point() +
  labs(title = "Metropolis Hasting Samples R Implementation",
       x = "Theta 1",
       y = "Theta 2")
```



Regarding the second implementation (own implementation), the algorithm will be:

```r
# Function to calculate the joint probability density function
joint_pdf <- function(theta, rho) {
  numerator <- exp(-0.5 * (1 - rho^2) * (theta[1]^2 - 2 *
                                           rho * theta[1] *
                                           theta[2] + theta[2]^2))
  denominator <- 2 * pi * sqrt(1 - rho^2)
  return(numerator / denominator)
}

# Metropolis-Hastings algorithm with block-wise update
metropolis_hastings <- function(iterations, rho) {
  # Initialize
  theta <- c(0, 0)  # Initial values for the parameters
  samples <- matrix(0, nrow = iterations, ncol = 2)

  # Metropolis-Hastings algorithm
  for (i in 1:iterations) {
    # Block-wise update: propose new values for both parameters simultaneously
    # You can adjust the proposal standard deviation
    proposal <- rnorm(2, mean = theta, sd = c(1, 1))

    # Calculate acceptance ratio
    alpha <- min(1, joint_pdf(proposal, rho) / joint_pdf(theta, rho))

    # Accept or reject the proposal
    if (runif(1) < alpha) {
      theta <- proposal
```

```
    }

    # Store the sample
    samples[i, ] <- theta
  }

  return(samples)
}
```

The above code block is based on the slide 24 of the lecture notes. It consists of two function. In the first function (joint_pdf), the ρ and the values of θ are given and the joint probability, of the two θ, is calculated based on the description of the exercise.

The main metropolis hasting sampling is done in the second function (metropolis_hastings). First, the θ values are initialized. Afterwards, for 30000 iterations, 2 values are proposed from 2 normal distributions with means equal to the old theta values and standard deviations equals to 1. Then, the accept ratio (α) is calculated by taking the min between 1 and the fraction of the joint probability of the proposed thetas divided by the joint probability of the thetas from the previous step. For that specific accept ratio α, the α is compared with a generated value from a uniform distribution $U(0, 1)$. If α is greater than the generated value, the new values of thetas are assigned to the proposal values. Otherwise, the new thetas receives the values of thetas from the previous step. Finally, the thetas are stored in the matrix, which was initialized at the beginning.

It is worth mentioning that the acceptance or the rejection of the thetas is based on the **acceptance-rejection method**. Furthermore, according to the algorithm:

$$\alpha(x, y) = \min\left\{\frac{\xi(y)}{\xi(x)}\frac{q(x|y)}{q(y|x)}, 1\right\}$$

Due to the fact that we have normal distributions, which are symmetric, for both parameters θ, it applies that:

$$q(x|y) = q(y|x)$$

Therefore, it cancels out in the acceptance probability and therefore they were removed from the calculation of α.

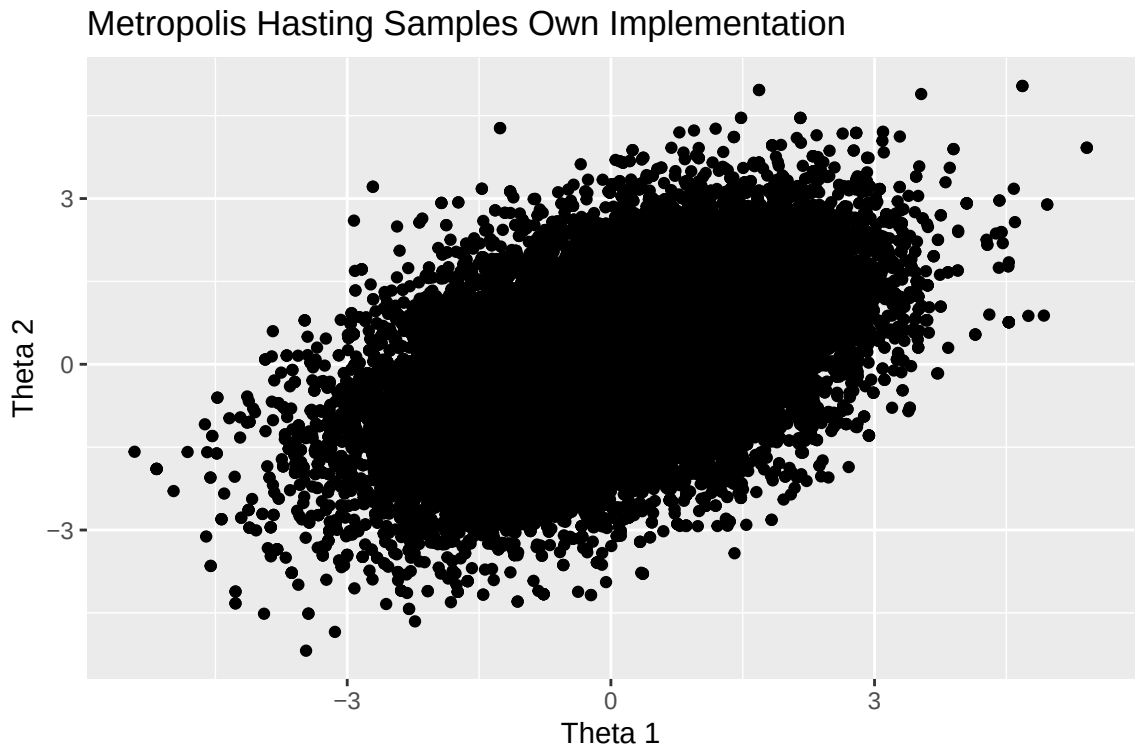
Afterwards, the algorithm above is run.

```
# Set parameters
rho <- 0.5
chain_size <- 30000

# Run Metropolis-Hastings algorithm
set.seed(12223236)
metropolis_samples_own <- metropolis_hastings(chain_size, rho)
metropolis_samples_own_frame <- data.frame(metropolis_samples_own)
```

In a repetitive manner, the scatterplot will be:

```
ggplot(metropolis_samples_own_frame, aes(x = X1, y = X2)) +
  geom_point() +
  labs(title = "Metropolis Hasting Samples Own Implementation", x = "Theta 1", y = "Theta 2")
```



Task 1.3

Perform chain diagnostics on the resulting chain.

We are going to perform chain diagnostics for the 3 cases:

1. Gibbs samples
2. Metropolis Hasting samples (R implementation)
3. Metropolis Hasting samples (own implementation)

The chain diagnostics, for each parameter theta, consist of 4 plots.

1. Trace plot, which a graphical representation that displays the iterations vs the sampled values for each parameter in the chain. It helps us understand the convergence, the different properties and the stationarity of the chain.
2. Autocorrelation plot, which is the correlation between the given parameter value in the Markov chain separated by h iterations. The term $h > 1$ is usually referred to as lag ($\rho_j(h) = Cor(\theta_j^{(t)}, \theta_j^{(t-h)})$). It is worth mentioning that for $h = 0$ the $\rho_j(h) = 1$ because it is the correlation of $\theta_j^{(t)}$ with itself and this always indicates perfect correlation. Furthermore, according to the lecture notes, the faster the decrease in the value of the autocorrelation function, while the lag increases, the more efficient the Markov chains. Therefore, based on the decreasing rate of the autocorrelation function, there are 3 cases:
    1. Ideal mixing, where the autocorrelation decreases instantly
    2. Typical good mixing, where the autocorrelation decreases steadily but fast
    3. Poor mixing, where the autocorrelation decreases at a very low rate and never becomes zero.

3. Mean plot, which is responsible for representing, for each parameter and for each iteration, the mean of the sampled values. It is a different way to help us assess the convergence and the stationarity of the chain.

4. Distribution plot, including the density, which shows how the sampled values are distributed.

Regarding the chain diagnostics, the code from slide 44 from the lecture notes. Therefore, few modifications are done:

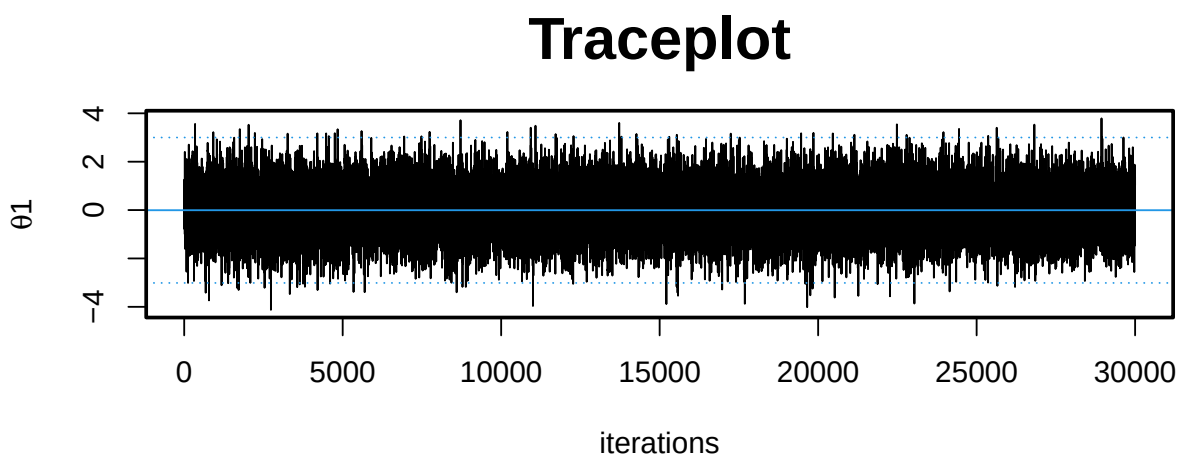
```
# install.packages("mcmcplots") # uncomment for installation
library(mcmcplots)
M <- chain_size # chain_size is 300000
```

Gibbs samples

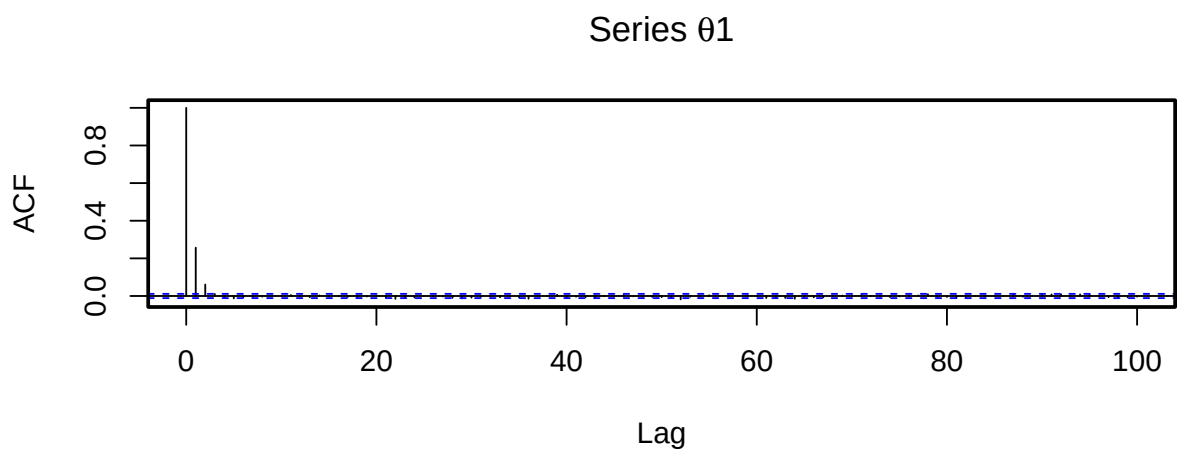
The chain diagnostics for Metropolis Hasting samples in R implementation, are:

```
t1 <- gibbs_samples[,1]
t2 <- gibbs_samples[,2]
```

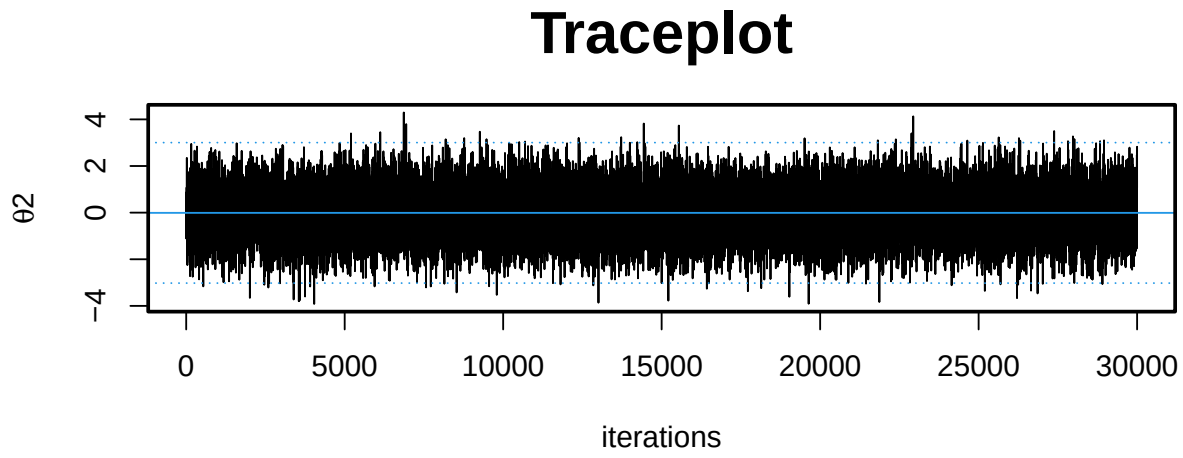
```
# trace plots
# par(mfrow=c(2,1),mar=c(5,5,4,2))
plot(1:M,t1,type="l",ylim=c(min(t1,mean(t1)-3*sd(t1)),
max(t1,mean(t1)+3*sd(t1))),xlab="iterations",
ylab=expression(paste(theta,"1")),cex.main=2,main="Traceplot")
abline(h=mean(t1),lty=1,col=4);
abline(h=mean(t1)+3*sd(t1),lty=3,col=4)
abline(h=mean(t1)-3*sd(t1),lty=3,col=4); box(lwd=2)
```



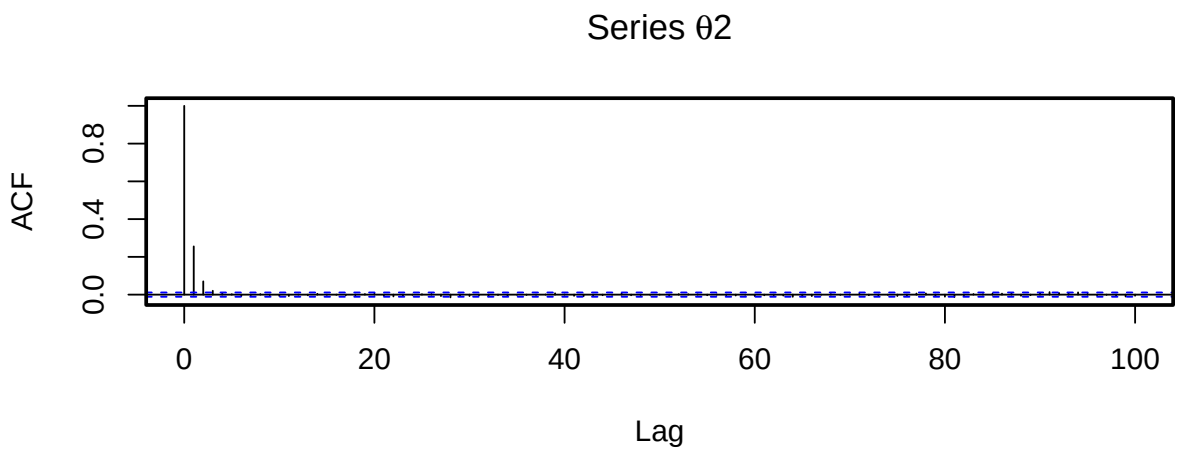

```
# ACF plots with run mean
acf(t1,lag.max=100,main=expression(paste("Series ",theta,"1")),cex.main=2); box(lwd=2)
```



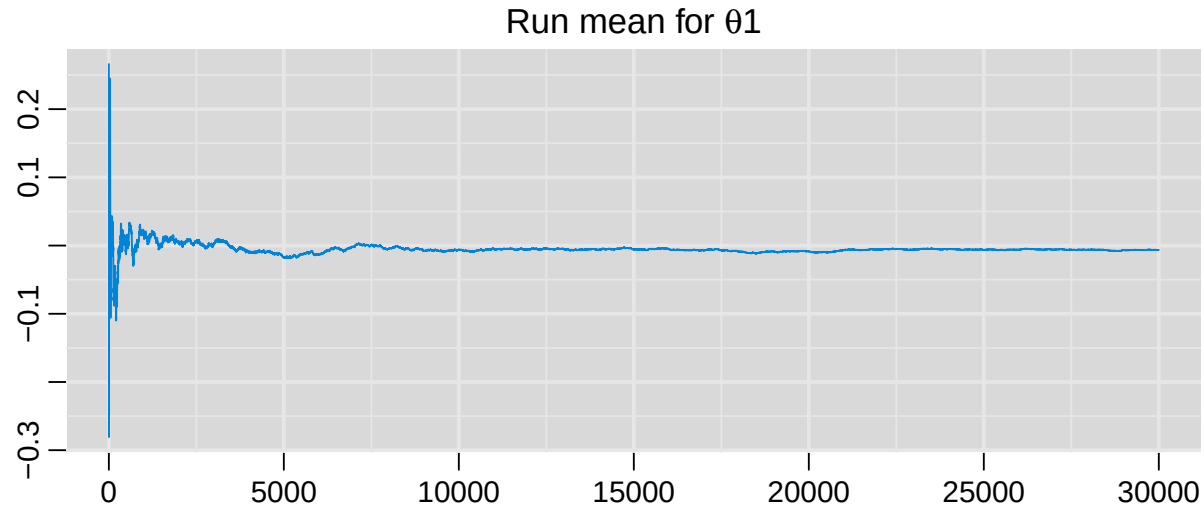

```
# par(mfrow=c(2,1),mar=c(5,5,4,2))
plot(1:M,t2,type="l",ylim=c(min(t2,mean(t2[-(1:200)])-3*sd(t2[-(1:200)])),
max(t2,mean(t2[-(1:200)])+3*sd(t2[-(1:200)]))),xlab="iterations",
ylab=expression(paste(theta,"2")),cex.main=2,main="Traceplot")
abline(h=mean(t2[-(1:200)]),lty=1,col=4)
abline(h=mean(t2[-(1:200)])+3*sd(t2[-(1:200)]),lty=3,col=4)
abline(h=mean(t2[-(1:200)])-3*sd(t2[-(1:200)]),lty=3,col=4); box(lwd=2)
```



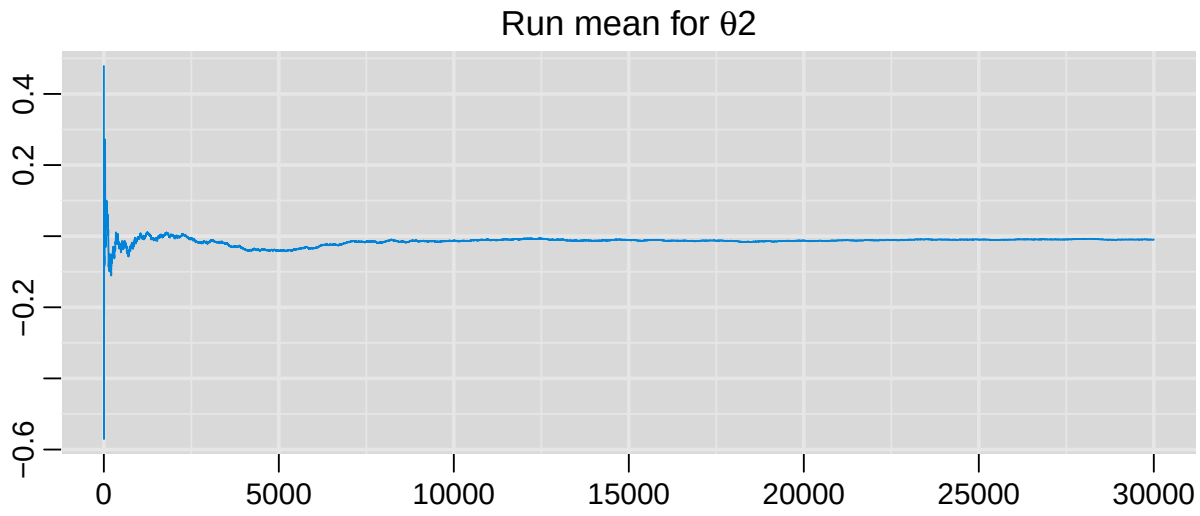

```
acf(t2,lag.max=100,main=expression(paste("Series ",theta,"2")),cex.main=2); box(lwd=2)
```



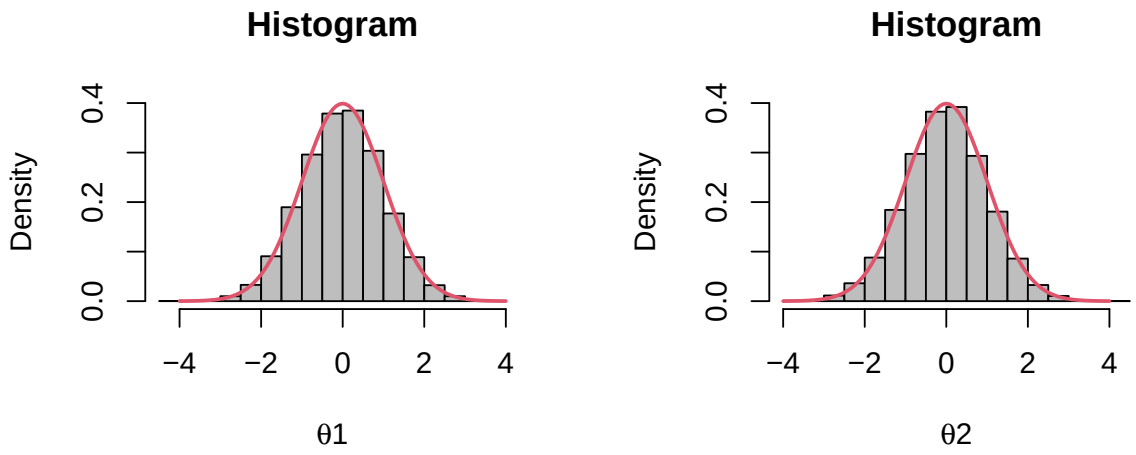

```
# histrograms of marginal posteriors
rmeanplot(t1,lwd=2,main=expression(paste("Run mean for ",theta,"1")),mar=c(2,2,1.5,1)+ 0.1)
```




```
rmeanplot(t2,lwd=2,main=expression(paste("Run mean for ",theta,"2")),mar=c(2,2,1.5,1)+ 0.1)
```




```
par(mfrow=c(1,2),mar=c(5,5,4,2))
hist(t1,freq=F,col="grey",main="Histogram",xlab=expression(paste(theta,"1")),ylim = c(0,0.4))
curve(dnorm(x,0,1),from=-4,4,add=T,lwd=2,col=2)
hist(t2,freq=F,col="grey",main="Histogram",xlab=expression(paste(theta,"2")), ylim = c(0,0.4))
curve(dnorm(x,0,1),from=-4,4,add=T,lwd=2,col=2)
```



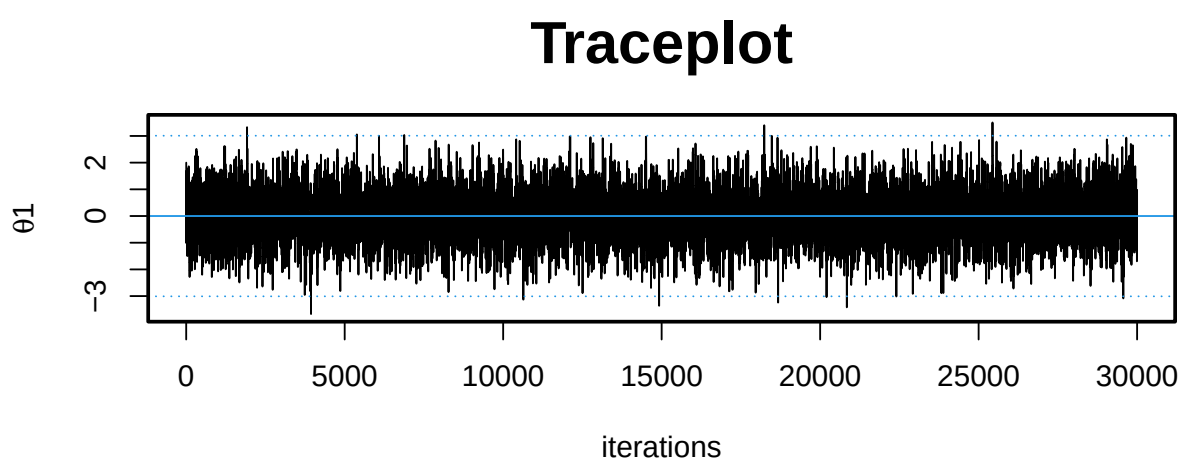
Metropolis Hasting samples (R implementation)

The chain diagnostics for Metropolis Hasting samples in R are:

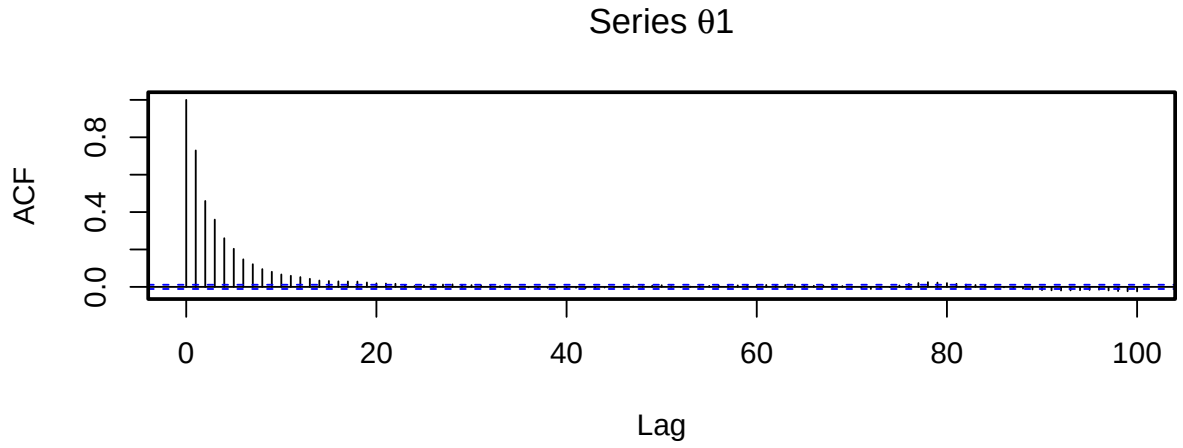
```
t1 <- metropolis_hastings_samples$x
t2 <- metropolis_hastings_samples$y
```

```
# trace plots
# par(mfrow=c(2,1),mar=c(5,5,4,2))
plot(1:M,t1,type="l",ylim=c(min(t1,mean(t1)-3*sd(t1)),
max(t1,mean(t1)+3*sd(t1))),xlab="iterations",
ylab=expression(paste(theta,"1")),cex.main=2,main="Traceplot")
```

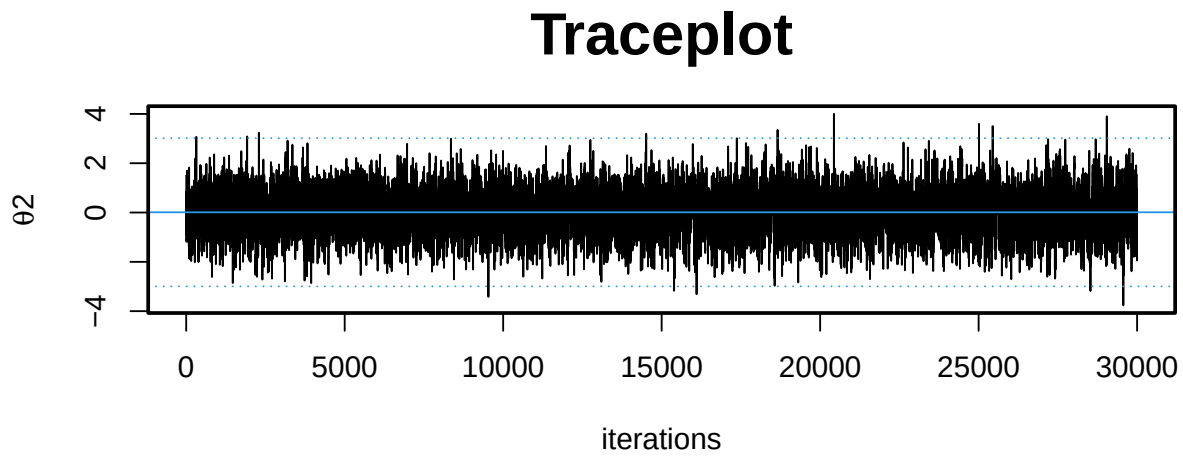
```
abline(h=mean(t1),lty=1,col=4);
abline(h=mean(t1)+3*sd(t1),lty=3,col=4)
abline(h=mean(t1)-3*sd(t1),lty=3,col=4); box(lwd=2)
```



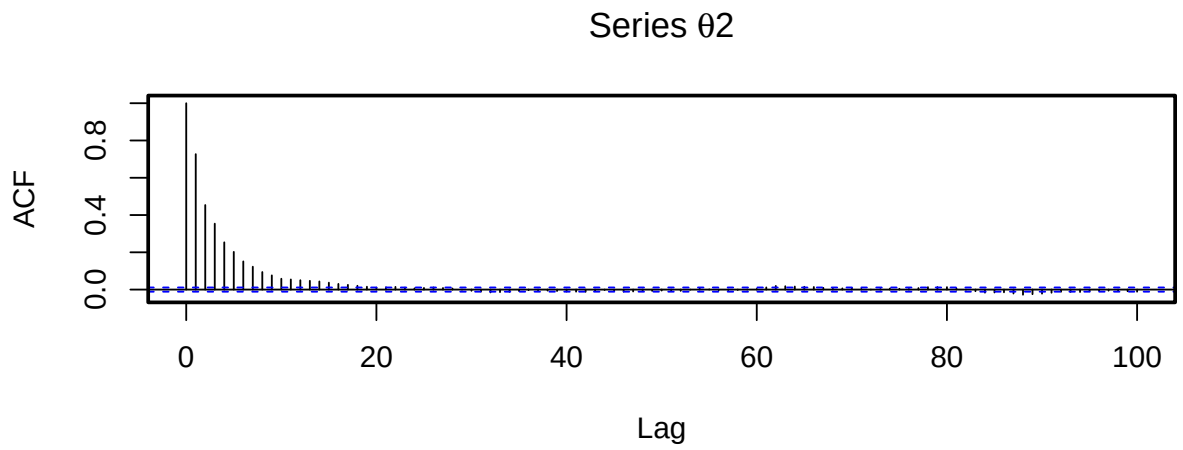

```
# ACF plots with run mean
acf(t1,lag.max=100,main=expression(paste("Series ",theta,"1")),cex.main=2); box(lwd=2)
```



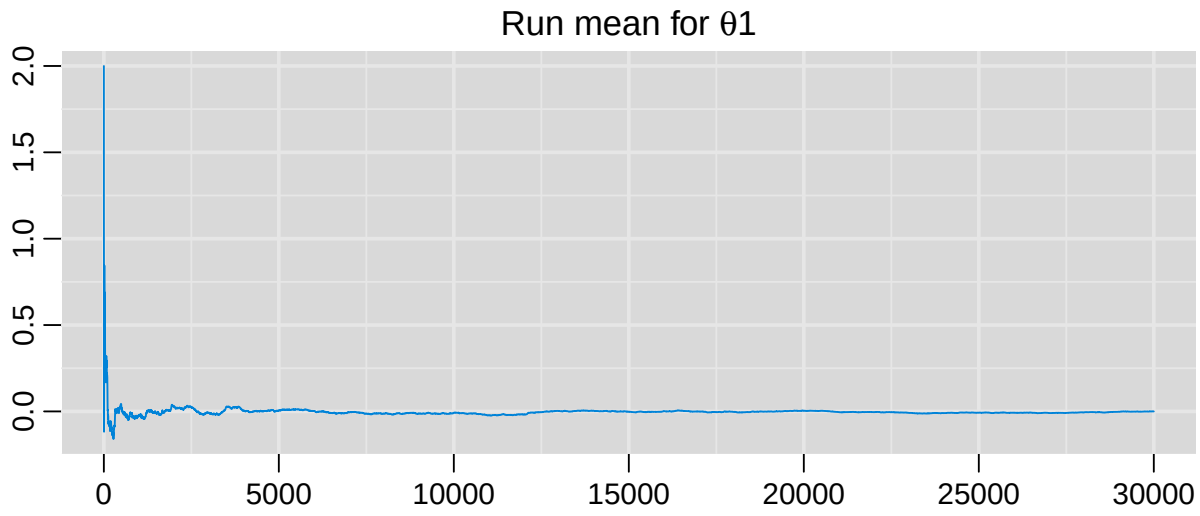

```
# par(mfrow=c(2,1),mar=c(5,5,4,2))
plot(1:M,t2,type="l",ylim=c(min(t2,mean(t2[-(1:200)])-3*sd(t2[-(1:200)])),
max(t2,mean(t2[-(1:200)])+3*sd(t2[-(1:200)]))),xlab="iterations",
ylab=expression(paste(theta,"2")),cex.main=2,main="Traceplot")
abline(h=mean(t2[-(1:200)]),lty=1,col=4)
abline(h=mean(t2[-(1:200)])+3*sd(t2[-(1:200)]),lty=3,col=4)
abline(h=mean(t2[-(1:200)])-3*sd(t2[-(1:200)]),lty=3,col=4); box(lwd=2)
```



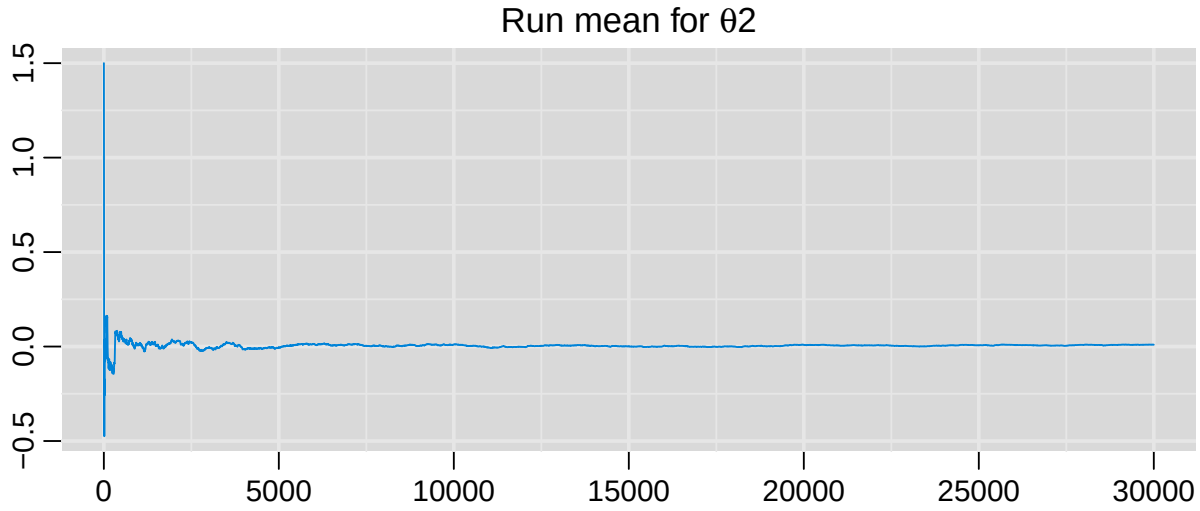

```
acf(t2,lag.max=100,main=expression(paste("Series ",theta,"2")),cex.main=2); box(lwd=2)
```



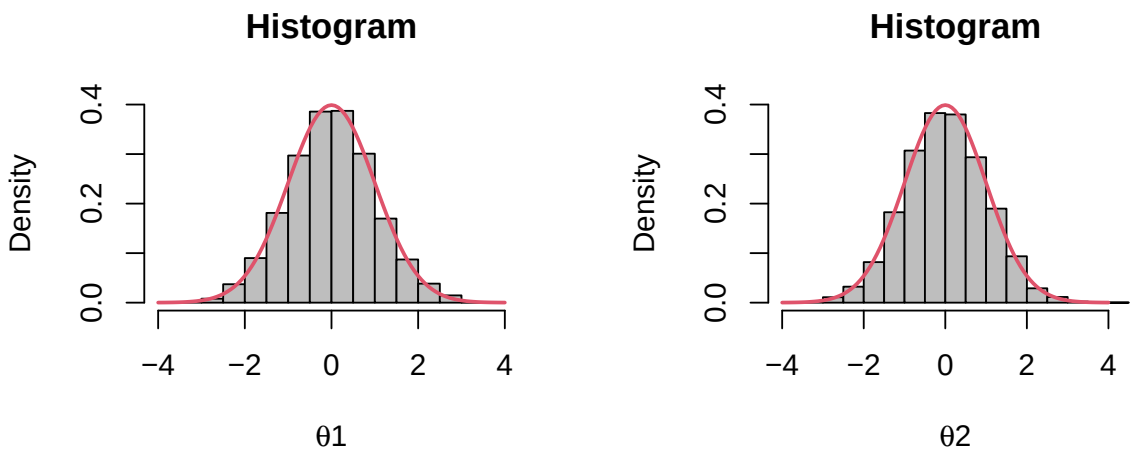

```
# histrograms of marginal posteriors
rmeanplot(t1,lwd=2,main=expression(paste("Run mean for ",theta,"1")),mar=c(2,2,1.5,1)+ 0.1)
```




```
rmeanplot(t2,lwd=2,main=expression(paste("Run mean for ",theta,"2")),mar=c(2,2,1.5,1)+ 0.1)
```




```
par(mfrow=c(1,2),mar=c(5,5,4,2))
hist(t1,freq=F,col="grey",main="Histogram",xlab=expression(paste(theta,"1")),ylim = c(0,0.4))
curve(dnorm(x,0,1),from=-4,4,add=T,lwd=2,col=2)
hist(t2,freq=F,col="grey",main="Histogram",xlab=expression(paste(theta,"2")), ylim = c(0,0.4))
curve(dnorm(x,0,1),from=-4,4,add=T,lwd=2,col=2)
```

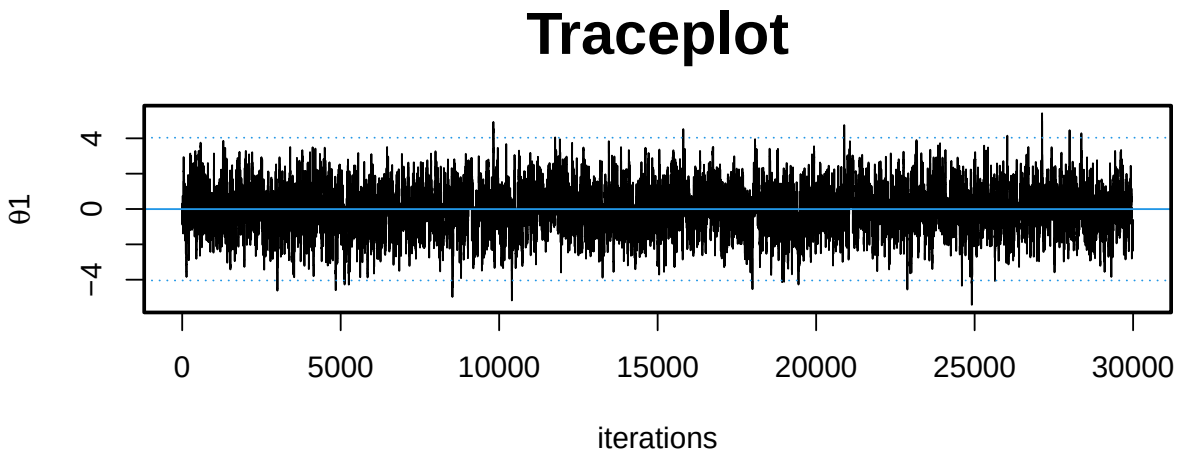


Metropolis Hasting samples (own implementation)

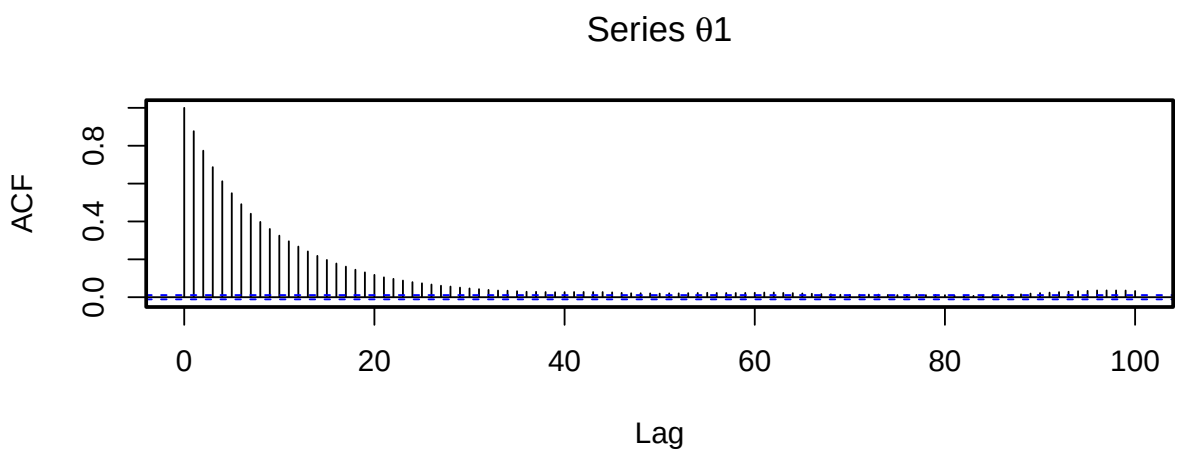
The chain diagnostics for Metropolis Hasting samples in our own implementation are:

```
t1 <- metropolis_samples_own[,1]
t2 <- metropolis_samples_own[,2]
```

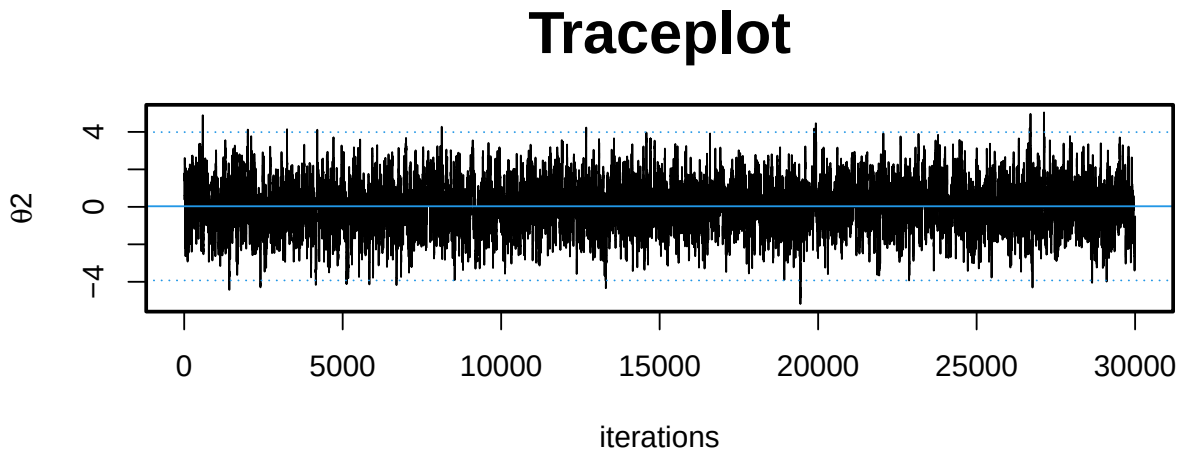
```
# trace plots
# par(mfrow=c(2,1),mar=c(5,5,4,2))
plot(1:M,t1,type="l",ylim=c(min(t1,mean(t1)-3*sd(t1)),
max(t1,mean(t1)+3*sd(t1))),xlab="iterations",
ylab=expression(paste(theta,"1")),cex.main=2,main="Traceplot")
abline(h=mean(t1),lty=1,col=4);
abline(h=mean(t1)+3*sd(t1),lty=3,col=4)
abline(h=mean(t1)-3*sd(t1),lty=3,col=4); box(lwd=2)
```



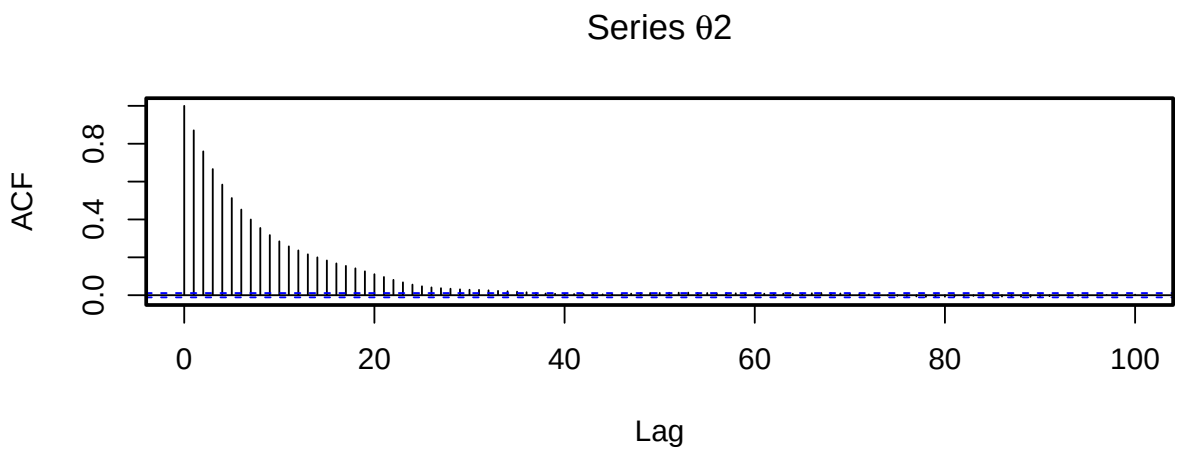

```
# ACF plots with run mean
acf(t1,lag.max=100,main=expression(paste("Series ",theta,"1")),cex.main=2); box(lwd=2)
```



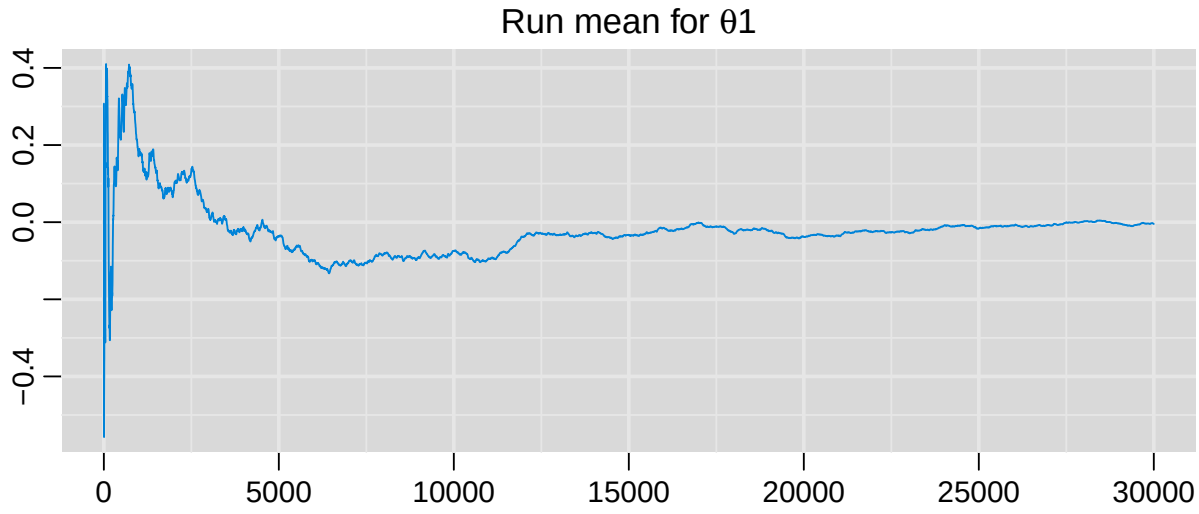

```
# par(mfrow=c(2,1),mar=c(5,5,4,2))
plot(1:M,t2,type="l",ylim=c(min(t2,mean(t2[-(1:200)])-3*sd(t2[-(1:200)])),
max(t2,mean(t2[-(1:200)])+3*sd(t2[-(1:200)]))),xlab="iterations",
ylab=expression(paste(theta,"2")),cex.main=2,main="Traceplot")
abline(h=mean(t2[-(1:200)]),lty=1,col=4)
abline(h=mean(t2[-(1:200)])+3*sd(t2[-(1:200)]),lty=3,col=4)
abline(h=mean(t2[-(1:200)])-3*sd(t2[-(1:200)]),lty=3,col=4); box(lwd=2)
```



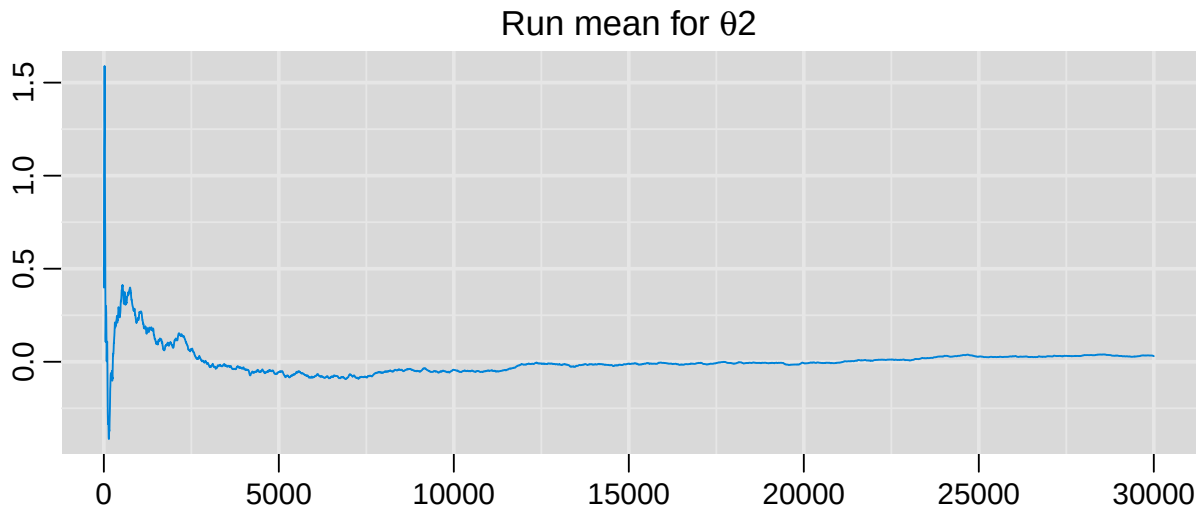

```
acf(t2,lag.max=100,main=expression(paste("Series ",theta,"2")),cex.main=2); box(lwd=2)
```



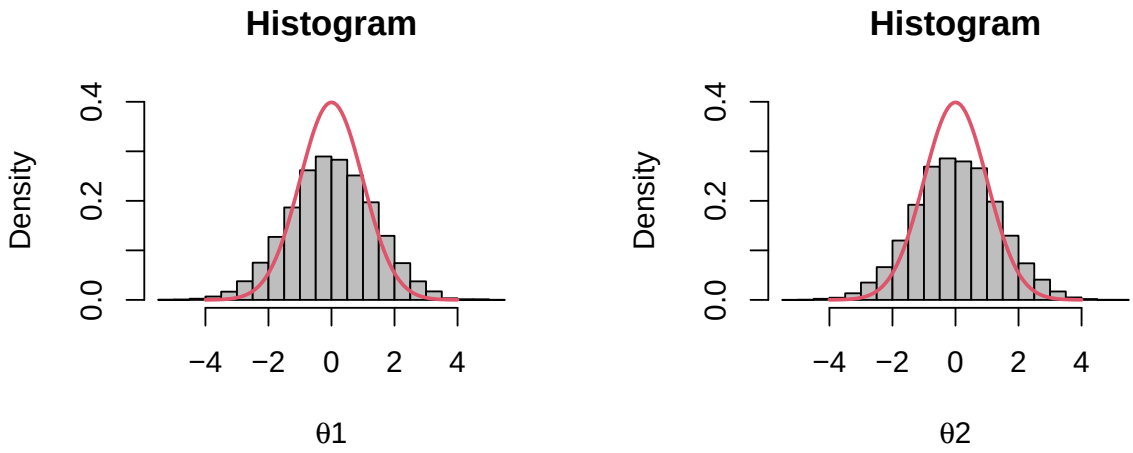

```
# histrograms of marginal posteriors
rmeanplot(t1,lwd=2,main=expression(paste("Run mean for ",theta,"1")),mar=c(2,2,1.5,1)+ 0.1)
```




```
rmeanplot(t2,lwd=2,main=expression(paste("Run mean for ",theta,"2")),mar=c(2,2,1.5,1)+ 0.1)
```




```
par(mfrow=c(1,2),mar=c(5,5,4,2))
hist(t1,freq=F,col="grey",main="Histogram",xlab=expression(paste(theta,"1")),ylim = c(0,0.4))
curve(dnorm(x,0,1),from=-4,4,add=T,lwd=2,col=2)
hist(t2,freq=F,col="grey",main="Histogram",xlab=expression(paste(theta,"2")), ylim = c(0,0.4))
curve(dnorm(x,0,1),from=-4,4,add=T,lwd=2,col=2)
```



Results interpretation

Gibbs samples

Regarding the trace plot for each parameter theta, the shape is good because all chains are stationary. There are no extreme and frequent spikes that exceed the thresholds, which are the $mean \pm 3sd$.

Regarding the autocorrelation functions, as mentioned before, it is basically the ideal mixing, for both parameters, because the correlation decreases instantly.

Regarding the mean plots, in the first 1500 iterations there are few spikes. However, afterwards the means converge to zero. The spikes in the beginning of the plot indicate the burn-in period which is the period until the chain converges. All those iterates/samples should be discarded.

Regarding the distribution of each parameter, the sampled values are normally distributed which is reasonable because it was known from the description of the assignment.

Metropolis Hasting samples (R implementation)

Regarding the trace plot for each parameter theta, the shape is, also, good because all chains are stationary.

Regarding the autocorrelation functions, they are slightly different compared to the Gibbs implementation. The mixing is typically good because the correlation decreases steadily but fast.

Regarding the mean plots, the means converge almost immediately to zero after 100-200 iterations. Therefore, there are quite a few samples that need to be discarded.

Regarding the distribution of each parameter, the distributions are normal as they were in the Gibbs implementation.

Metropolis Hasting samples (own implementation)

In this particular implementation, the most differences are observed.

Regarding the trace plots, The periodicity in the sample values is evident. This implies the difficulty of the chains to converge to a stationary state. This is also apparent from the mean plots.

Regarding the autocorrelation functions, we could say that the mixing is poor duo to the fact that the autocorrelation decreases at a very low rate. It becomes zero after the lag is greater than 25.

Regarding the distribution of each parameter, the distributions are normal as they were in the Gibbs and Metropolis hasting in R implementation but the peaks are slightly lower compared to the others.