Assignment_5

Abbas Zal_ 2072054

```
library(ggplot2)
library(coda)
library(rjags)

## Linked to JAGS 4.3.1

## Loaded modules: basemod,bugs
```

Exercise 1

Ladislaus Josephovich Bortkiewicz was a Russian economist and statistician. He noted that the Poisson distribution can be very useful in applied statistics when describing low-frequency events in a large population. In a famous example he showed that the number of deaths by horse kick among the Prussian army follows the Poisson distribution.

Considering the following to sets of observations taken over a fixed large time interval in two different corps:

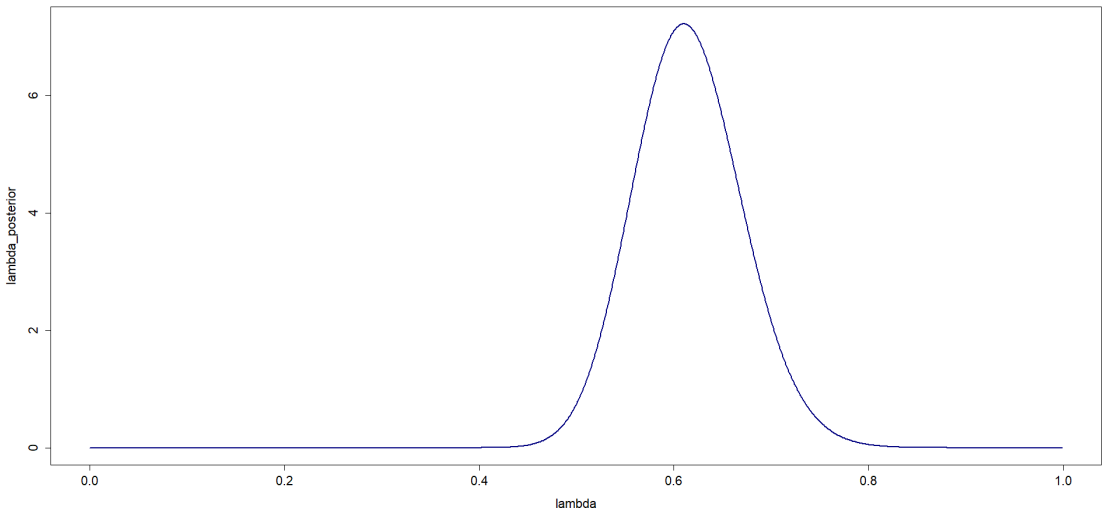
y death soldiers	0	1	2	3	4	≥ 5
n_1 observations	109	65	22	3	1	0
n_2 observations	144	91	32	11	2	0

A - assuming a uniform prior, compute and plot the posterior distribution for λ, the death rate over the measurement time. Determine the posterior mean, median and variance, and compute the 95% credibility interval.

For n1 observations:

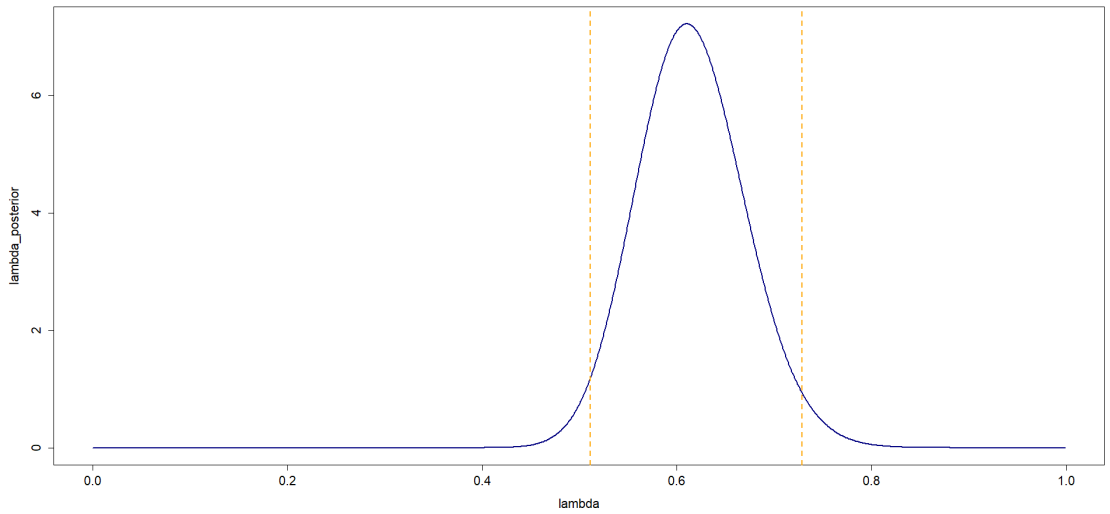
```
# Example usage
death <- c(0, 1, 2, 3, 4, 0)
obs_1 <- c(109, 65, 22, 3, 1, 0)
obs_2 <- c(144, 91, 32, 11, 2, 0)

calculate_sigma <- function(death, obs) {
  sigma_y <- 0
  for (i in 1:6) {
    sigma_y <- sigma_y + (obs[i] * death[i])
  }
  return(sigma_y)
```

```
}

sigma_y_1 <- calculate_sigma(death, obs_1)
sigma_y_2 <- calculate_sigma(death, obs_2)

n_1 <- sum(obs_1)
n_2 <- sum(obs_2)


n.sample <- 1000
delta.lambda <- 1/n.sample

alpha <- 1 + sigma_y_1
mu <- n_1
lambda <- seq(from = 0 , by = delta.lambda , length.out = n.sample )

lambda_posterior <- dgamma(lambda , alpha , mu)


mean <- delta.lambda * sum(lambda * lambda_posterior)
variance <- delta.lambda * sum(lambda ^ 2 * lambda_posterior) - mean ^ 2
median <- qgamma(0.5 , alpha , mu )
ci <- qgamma(c(0.025 , 0.975) , alpha , mu)



plot(lambda , lambda_posterior  , type = 'l' ,lwd = 2.5 ,  col = 'navyblue',
cex.axis = 1.40, cex.lab = 1.40)
```

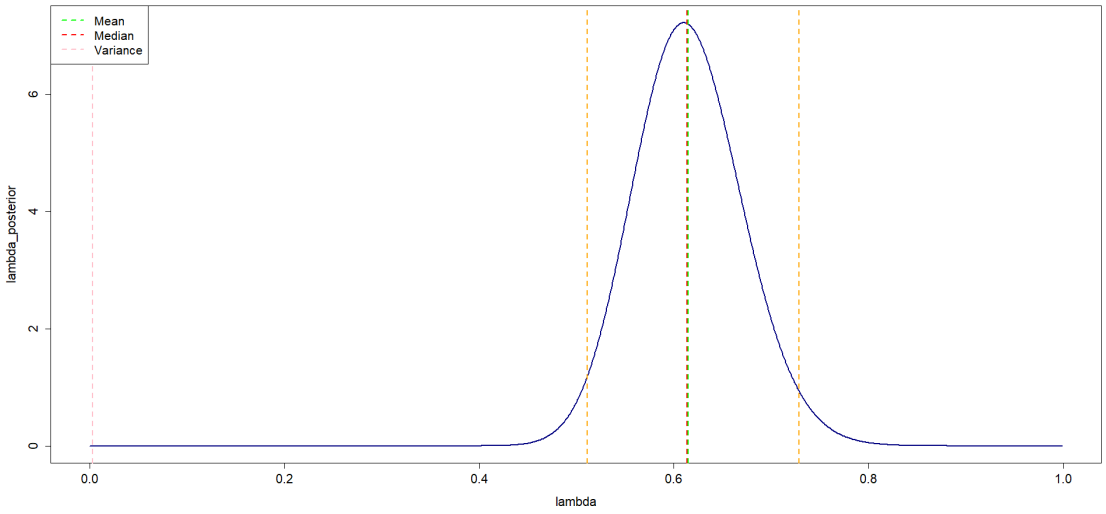



```
#print the results
cat("mean: ", round(mean, 5), "\n")

## mean:  0.615
```

```r
cat("variance: ", round(variance, 5), "\n")

## variance:  0.00308

cat("median: ", round(median, 5), "\n")

## median:  0.61333

plot(lambda , lambda_posterior  , type = 'l' ,lwd = 2.5 ,  col = 'navyblue',
cex.axis = 1.40, cex.lab = 1.40)
abline(v = ci[1] , lty = 'dashed' , lwd = 2.4, col = 'orange')
abline(v = ci[2] , lty = 'dashed' , lwd = 2.4, col = 'orange')
```

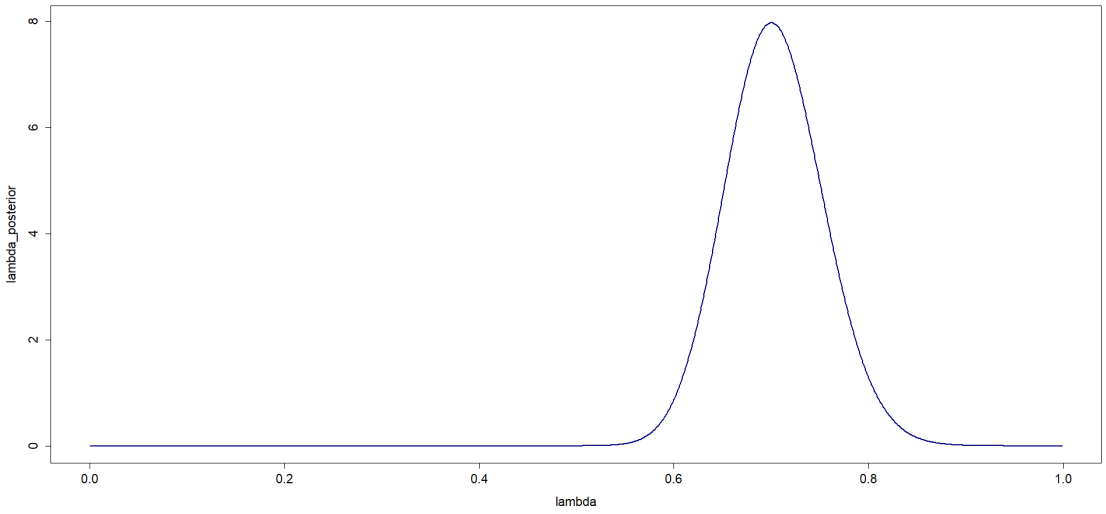



```r
plot(lambda , lambda_posterior  , type = 'l' ,lwd = 2.5 ,  col = 'navyblue',
cex.axis = 1.40, cex.lab = 1.40)
abline(v = mean , lty = 'dashed' , lwd = 2.4, col = 'green')
abline(v = median , lty = 'dashed' , lwd = 2.4, col = 'red')
abline(v = variance , lty = 'dashed' , lwd = 2.4, col = 'pink')
abline(v = ci[1] , lty = 'dashed' , lwd = 2.4, col = 'orange')
abline(v = ci[2] , lty = 'dashed' , lwd = 2.4, col = 'orange')

# Legend
legend("topleft", legend = c( "Mean", "Median", "Variance"),
       lty = c("dashed", "dashed", "dashed"),
       lwd = c(2,2,2),
       col = c( "green", "red", "pink" ),
       cex= 1.3)
```

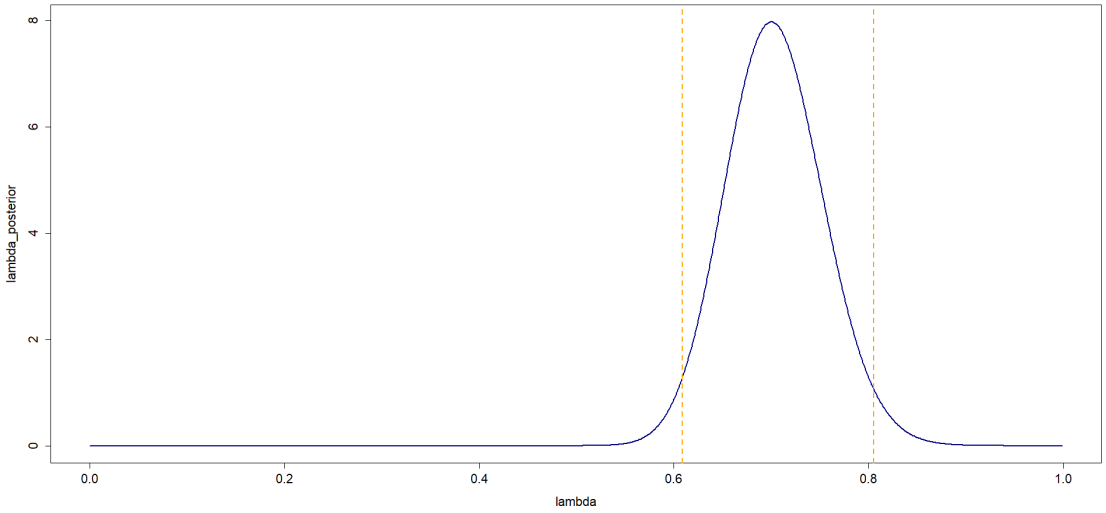


For n2 observations:

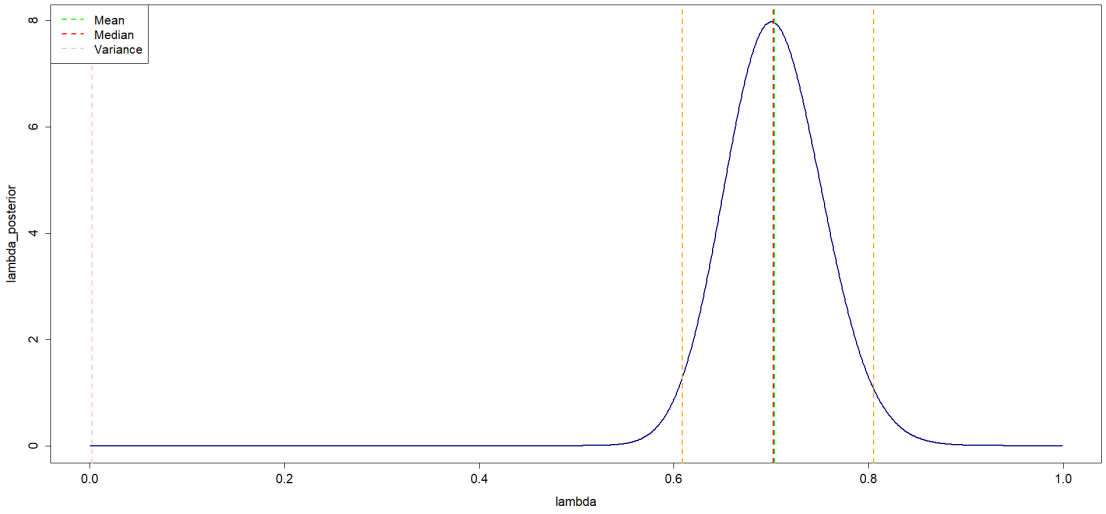
```
alpha <- 1 + sigma_y_2
mu <- n_2
lambda <- seq(from = 0 , by = delta.lambda , length.out = n.sample )

lambda_posterior <- dgamma(lambda , alpha , mu)


mean <- delta.lambda * sum(lambda * lambda_posterior)
variance <- delta.lambda * sum(lambda ^ 2 * lambda_posterior) - mean ^ 2
median <- qgamma(0.5 , alpha , mu )
ci <- qgamma(c(0.025 , 0.975) , alpha , mu)



plot(lambda , lambda_posterior  , type = 'l' ,lwd = 2.5 ,  col = 'navyblue',
cex.axis = 1.40, cex.lab = 1.40)
```




```
#print the results
cat("mean: ", round(mean, 5), "\n")

## mean:  0.70357

cat("variance: ", round(variance, 5), "\n")

## variance:  0.00251

cat("median: ", round(median, 5), "\n")

## median:  0.70238

plot(lambda , lambda_posterior  , type = 'l' ,lwd = 2.5 ,  col = 'navyblue',
cex.axis = 1.40, cex.lab = 1.40)
abline(v = ci[1] , lty = 'dashed' , lwd = 2.4, col = 'orange')
abline(v = ci[2] , lty = 'dashed' , lwd = 2.4, col = 'orange')
```



```r
plot(lambda, lambda_posterior, type = 'l', lwd = 2.5, col = 'navyblue',
cex.axis = 1.40, cex.lab = 1.40)
abline(v = mean, lty = 'dashed', lwd = 2.4, col = 'green')
abline(v = median, lty = 'dashed', lwd = 2.4, col = 'red')
abline(v = variance , lty = 'dashed' , lwd = 2.4 , col = 'pink')
abline(v = ci[1], lty = 'dashed', lwd = 2.4, col = 'orange')
abline(v = ci[2], lty = 'dashed', lwd = 2.4, col = 'orange')

# Legend
legend("topleft", legend = c( "Mean", "Median", "Variance"),
       lty = c("dashed", "dashed", "dashed"),
       lwd = c(2,2,2),
       col = c( "green", "red", "pink" ),
       cex = 1.3)
```



B - assuming now a Jeffreys' prior,

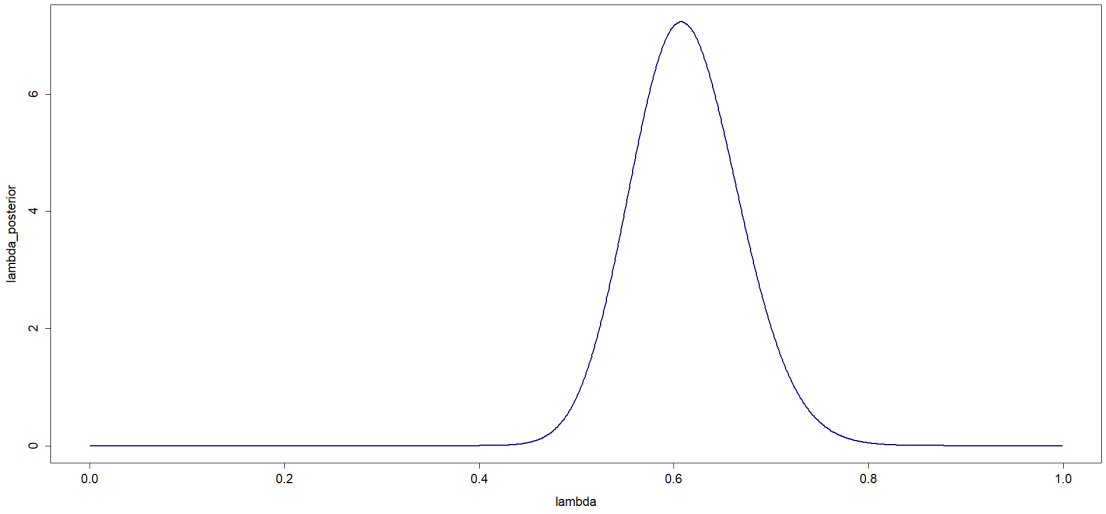
$g(\lambda) \propto 1 / \sqrt{\lambda}$, with $\lambda > 0$

In this section we want to compute and plot the posterior distribution for λ, the death rate over the measurement time. Determine the posterior mean, median and variance, and compute the 95% credibility interval.

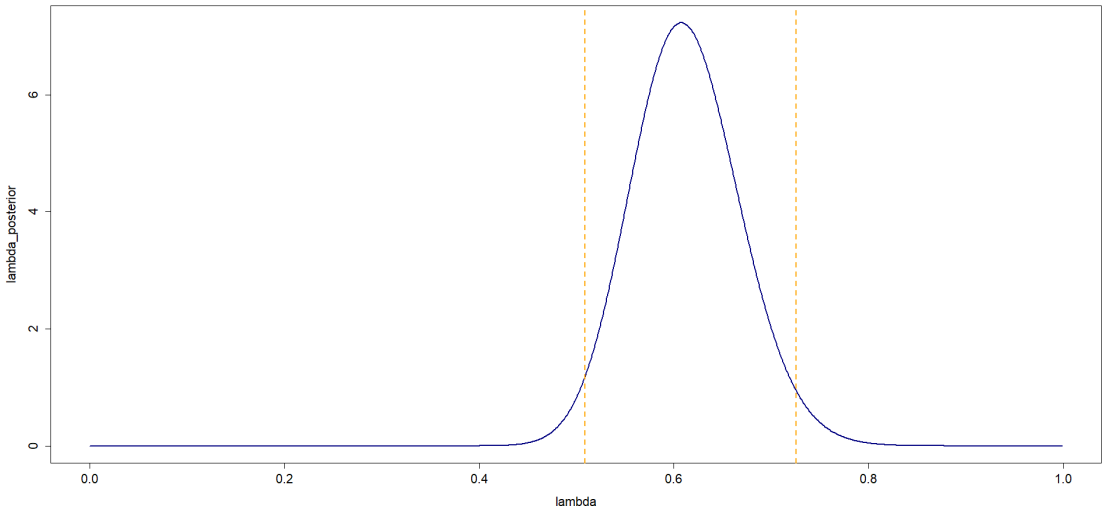
For n1 observation:

```r
alpha <- 0.5 + sigma_y_1
mu <- n_1


lambda_posterior <- dgamma(lambda , alpha , mu)


mean <- delta.lambda * sum(lambda * lambda_posterior)
```

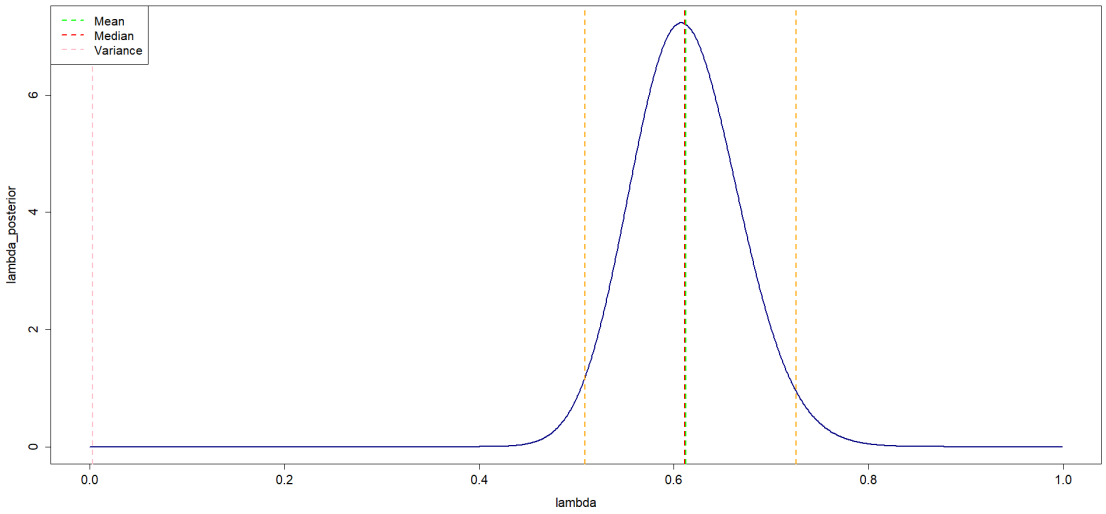
```r
variance <- delta.lambda * sum(lambda ^ 2 * lambda_posterior) - mean ^ 2
median <- qgamma(0.5 , alpha , mu )
ci <- qgamma(c(0.025 , 0.975) , alpha , mu)

plot(lambda , lambda_posterior  , type = 'l' ,lwd = 2.5 ,  col = 'navyblue',
cex.axis = 1.40, cex.lab = 1.40)
```




```r
#print the results
cat("mean: ", round(mean, 5), "\n")

## mean:  0.6125

cat("variance: ", round(variance, 5), "\n")

## variance:  0.00306

cat("median: ", round(median, 5), "\n")

## median:  0.61083

plot(lambda , lambda_posterior  , type = 'l' ,lwd = 2.5 ,  col = 'navyblue',
cex.axis = 1.40, cex.lab = 1.40)
abline(v = ci[1] , lty = 'dashed' , lwd = 2.4, col = 'orange')
abline(v = ci[2] , lty = 'dashed' , lwd = 2.4, col = 'orange')
```

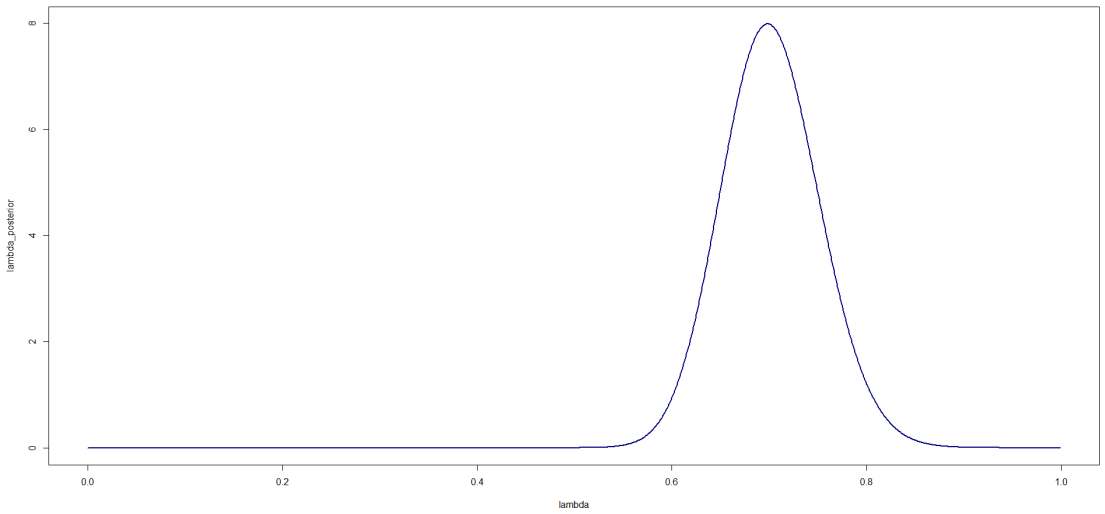



```
plot(lambda, lambda_posterior, type = 'l', lwd = 2.5, col = 'navyblue',
cex.axis = 1.40, cex.lab = 1.40)
abline(v = mean, lty = 'dashed', lwd = 2.4, col = 'green')
abline(v = median, lty = 'dashed', lwd = 2.4, col = 'red')
abline(v = variance , lty = 'dashed' , lwd = 2.4 , col = 'pink')
abline(v = ci[1], lty = 'dashed', lwd = 2.4, col = 'orange')
abline(v = ci[2], lty = 'dashed', lwd = 2.4, col = 'orange')

# Legend
legend("topleft", legend = c( "Mean", "Median", "Variance"),
       lty = c("dashed", "dashed", "dashed"),
       lwd = c(2,2,2),
       col = c( "green", "red", "pink" ),
       cex = 1.3)
```

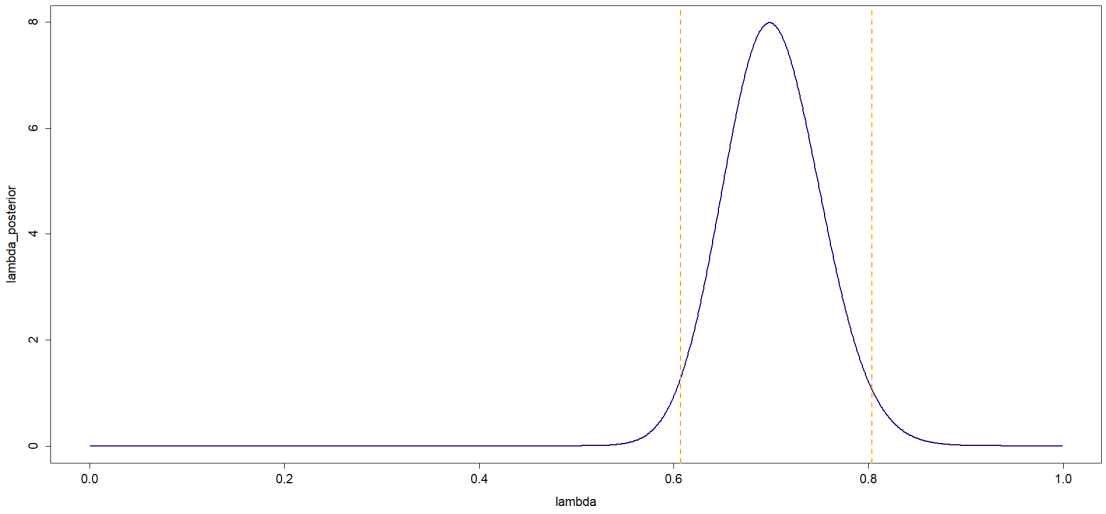


For n2 observation:

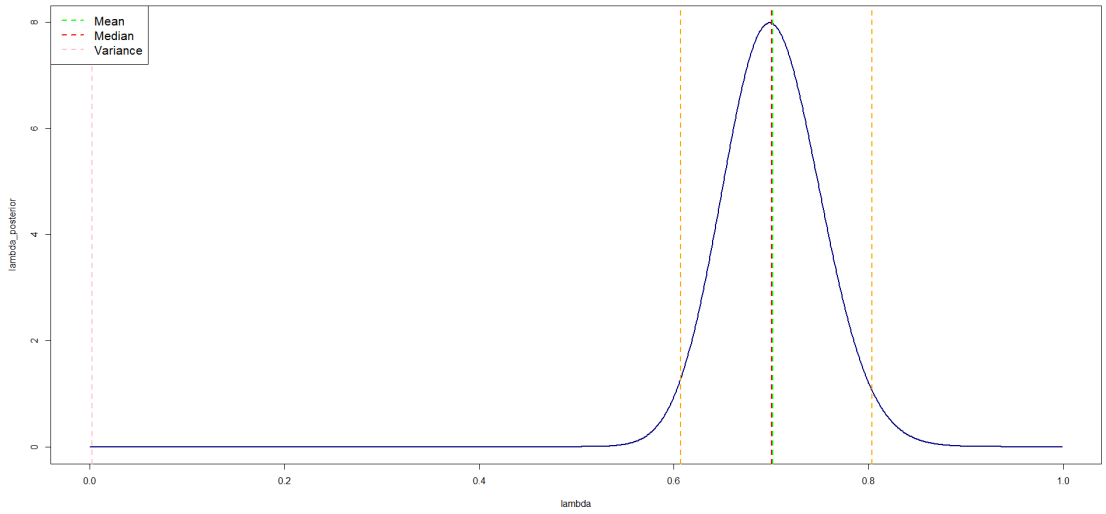
```r
alpha <- 0.5 + sigma_y_2
mu <- n_2


lambda_posterior <- dgamma(lambda , alpha , mu)


mean <- delta.lambda * sum(lambda * lambda_posterior)
variance <- delta.lambda * sum(lambda ^ 2 * lambda_posterior) - mean ^ 2
median <- qgamma(0.5 , alpha , mu )
ci <- qgamma(c(0.025 , 0.975) , alpha , mu)

plot(lambda , lambda_posterior  , type = 'l' ,lwd = 2.5 ,  col = 'navyblue')
```




```r
#print the results
cat("mean: ", round(mean, 5), "\n")

## mean:  0.70179

cat("variance: ", round(variance, 5), "\n")

## variance:  0.00251

cat("median: ", round(median, 5), "\n")

## median:  0.7006

plot(lambda , lambda_posterior  , type = 'l' ,lwd = 2.5 ,  col = 'navyblue',
cex.axis = 1.40, cex.lab = 1.40)
abline(v = ci[1] , lty = 'dashed' , lwd = 2.4, col = 'orange')
abline(v = ci[2] , lty = 'dashed' , lwd = 2.4, col = 'orange')
```




```r
plot(lambda, lambda_posterior, type = 'l', lwd = 2.5, col = 'navyblue')
abline(v = mean, lty = 'dashed', lwd = 2.4, col = 'green')
abline(v = median, lty = 'dashed', lwd = 2.4, col = 'red')
abline(v = variance , lty = 'dashed' , lwd = 2.4 , col = 'pink')
abline(v = ci[1], lty = 'dashed', lwd = 2.4, col = 'orange')
abline(v = ci[2], lty = 'dashed', lwd = 2.4, col = 'orange')

# Legend
legend("topleft", legend = c( "Mean", "Median", "Variance"),
       lty = c("dashed", "dashed", "dashed"),
       lwd = c(2,2,2),
       col = c( "green", "red", "pink" ),
       cex = 1.3)
```

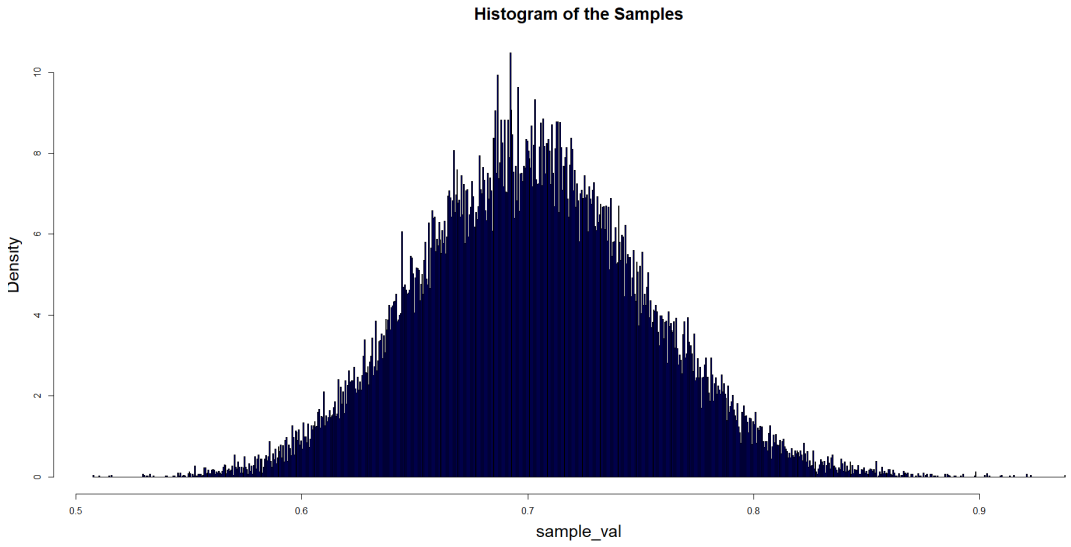


Exercise 2

solve Exercise 1 with a Markov Chain Monte Carlo. Build your own MCMC, we are using the functions that were introduced during lectures.

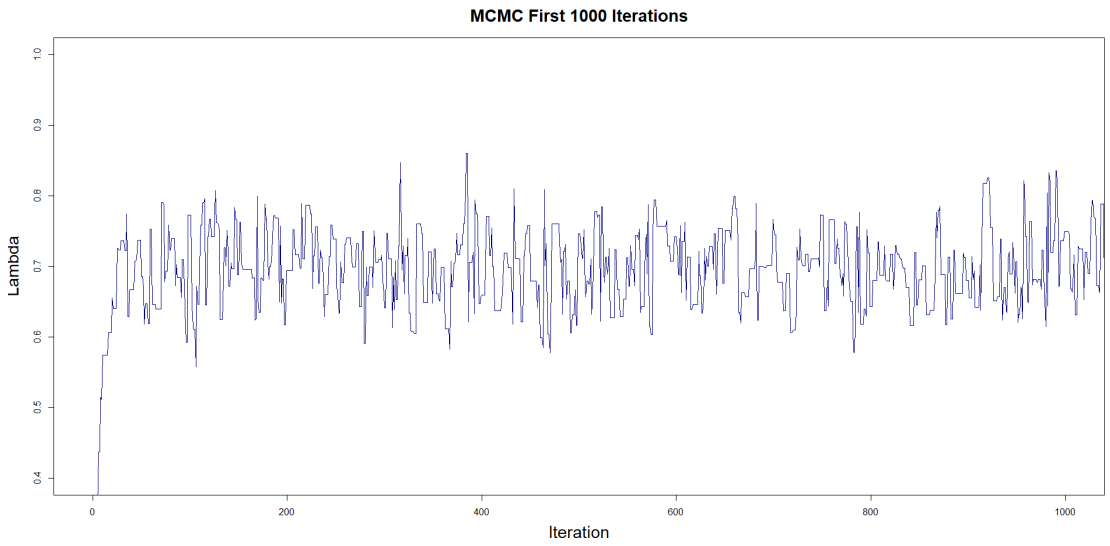
```
# Parameters:
# func : a function whose first argument is a real vector of parameters
# func returns a log10 of the likelihood function
# lambda.init : the initial value of the Markov Chain (and of func)
# n.sample: number of required samples
# sigma : standar deviation of the gaussian MCMC sampling pdf

metropolis.1dim <- function(func , lambda.init , n.sample , sigma) {
theta.cur <- lambda.init
func.Cur <- func(theta.cur)
func.Samp <- matrix(data=NA, nrow=n.sample , ncol=2+1)
n.accept <- 0
rate.accept <- 0.0

for (n in 1:n.sample) {
theta.prop <- rnorm(n=1, mean = theta.cur, sigma)
func.Prop <- func(theta.prop)
logMR <- func.Prop - func.Cur # Log10 of the Metropolis ratio
if ( logMR >=0 || logMR >log10(runif(1)) ) {
theta.cur <- theta.prop
func.Cur <- func.Prop
n.accept <- n.accept + 1
}
func.Samp[n, 1] <- func.Cur
func.Samp[n, 2] <- theta.cur
func.Samp[n, 3] <- n
}
return(func.Samp)
}



#
# Our test function
#
testfunc <- function(lambda) {
return(dgamma(lambda , shape = alpha , rate = mu))
}
#
# - interface for the metropolis function , gets the log10 of test function
testfunc.metropolis <- function(lambda) {
return(log10(testfunc(lambda )))
}

lambda.init <- 0.1
sample.sig <- 0.1
```

```r
n.sample <- 100000
demo <- TRUE

set.seed(20190513)
chain <- metropolis.1dim(func=testfunc.metropolis ,
                         lambda.init = lambda.init ,
                         n.sample = n.sample ,
                         sigma = sample.sig)

n <- nrow(chain)
mcmc.data <- chain[1000:n , 2]
str(mcmc.data)

##  num [1:99001] 0.749 0.749 0.749 0.749 0.747 ...

hist(mcmc.data , breaks = 1000 , freq = F , main = 'Histogram of the Samples'
, xlab = 'sample_val', col="navyblue",cex.main=1.8,cex.lab=1.8)
```



In this context, chain is a matrix where each row represents a sample and each column represents a different variable or parameter. The third column (chain[, 3]) contains the iteration numbers, while the second column (chain[, 2]) contains the corresponding lambda values.

```r
plot(chain[, 3], chain[, 2], type = 'l', ylim = c(0.4, 1), xlab =
'Iteration', ylab = 'Lambda', xlim = c(0, 1000), main = 'MCMC First 1000
Iterations',lwd=1.5, col="navyblue",cex.main=1.8,cex.lab=1.8)
```




```
mean_<- mean(mcmc.data)
variance_<- var(mcmc.data)
median_<- median(mcmc.data)

#print the results
cat("mean: ", round(mean_, 5), "\n")

## mean:  0.70182

cat("variance: ", round(variance_, 5), "\n")

## variance:  0.00251

cat("median: ", round(median_, 5), "\n")

## median:  0.70092
```

Exercise 3

A study on water quality of streams, a high level of bacter X was defined as a level greater than 100 per 100 ml of stream water. n = 116 samples were taken from streams having a high environmental impact on pandas. Out of these, y = 11 had a high bacter X level.

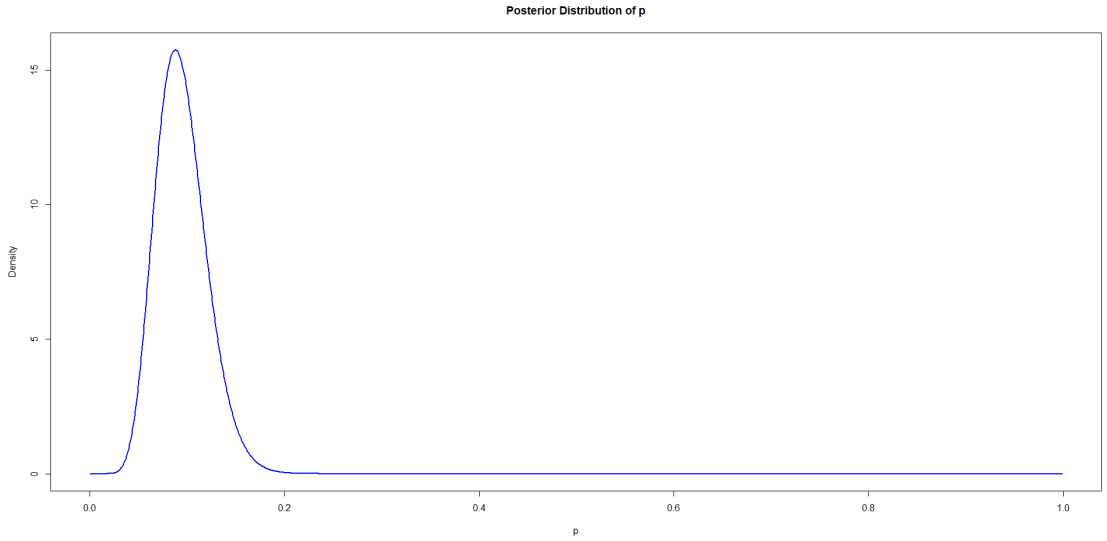
indicating with p the probability that a sample of water taken from the stream has a high bacter X level,

(a) find the frequentist estimator for p:

```
y <- 11
n <- 116
frequentist_estimator <- y/n
frequentist_estimator

## [1] 0.09482759
```

(b) using a Beta(1, 10) prior for p, calculate and posterior distribution P(p | y)

```
alpha_prior <- 1
beta_prior <- 10

alpha_posterior <- alpha_prior + y
beta_posterior <- beta_prior + (n - y)
p = seq(from = 0 , by = 0.001 , length.out = 1000)

posterior_samples <- dbeta(p, alpha_posterior, beta_posterior)

# Plotting the posterior distribution
plot(p, posterior_samples, type = 'n', xlab = "p", ylab = "Density", main =
"Posterior Distribution of p")
polygon(c(p, rev(p)), c(posterior_samples, rep(0, length(p))), col = "white",
border = NA)
lines(p, posterior_samples,lwd=2, col = "blue",cex.main=1.8,cex.lab=1.8)
```



(c) find the bayesian estimator for p, the posterior mean and variance, and a 95% credible interval:

```
n <- nrow(chain)

sum <- sum( posterior_samples * p)
mean <- sum / n

variance = 0.001 * sum(p ^ 2 * posterior_samples) - mean ^ 2
sigma = sqrt(variance)

cat("mean: ", round(mean, 5), "\n")

## mean:  0.00094

cat("variance: ", round(variance, 5), "\n")
```

```
## variance:  0.0096

print(c(mean - 2 * sigma , mean + 2 * sigma))

## [1] -0.1949690  0.1968588
```

(d) test the hypotesis:

$$H_\circ : p = 0.1 \text{ versus } H_1 : p \neq 0.1$$

at 5% level of significance with both the frequentist and bayesian approach :

```
p_value <- pbinom(11 , n , 0.1 , lower.tail = T) + pbinom(105 , n , 0.1 ,
lower.tail = F)
p_null <- 0.1

# Compare the p-value with the significance level
alpha <- 0.05
if (p_value < alpha) {
  cat("Reject the null hypothesis H₀: p =", p_null, "\n")
} else {
  cat("Fail to reject the null hypothesis H₀: p =", p_null, "\n")
}

## Fail to reject the null hypothesis H₀: p = 0.1

cat("p-value:", p_value, "\n")

## p-value: 1
```

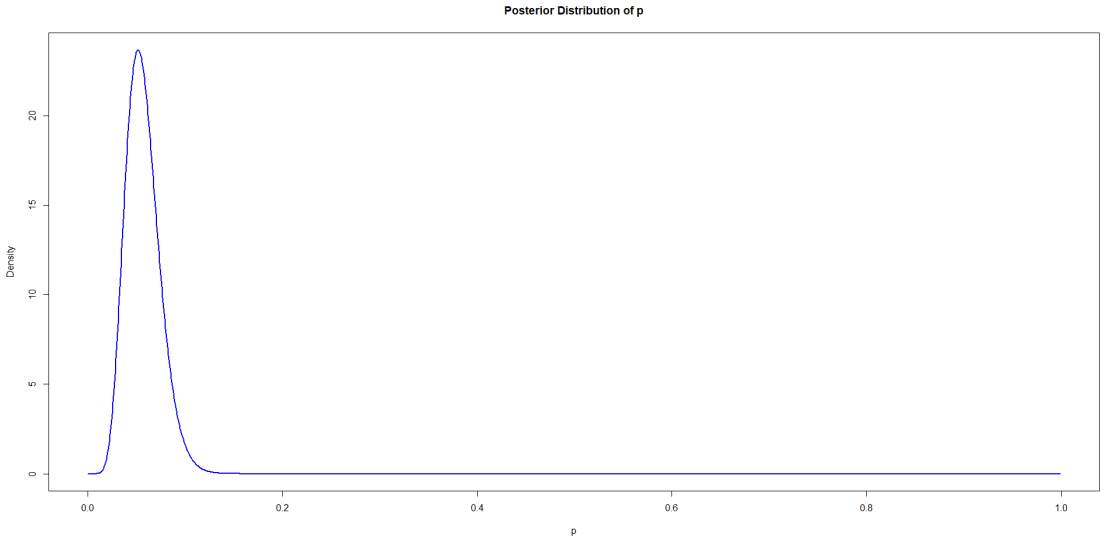
(e) find the frequentist estimator for p:

```
y <- 9
n <- 165
frequentist_estimator <- y/n
frequentist_estimator

## [1] 0.05454545
```

(f)) find a bayesian estimator for p, assuming both a Beta(1, 10) prior for p, and assuming the posterior probability of the older measurement as the prior for the new one.

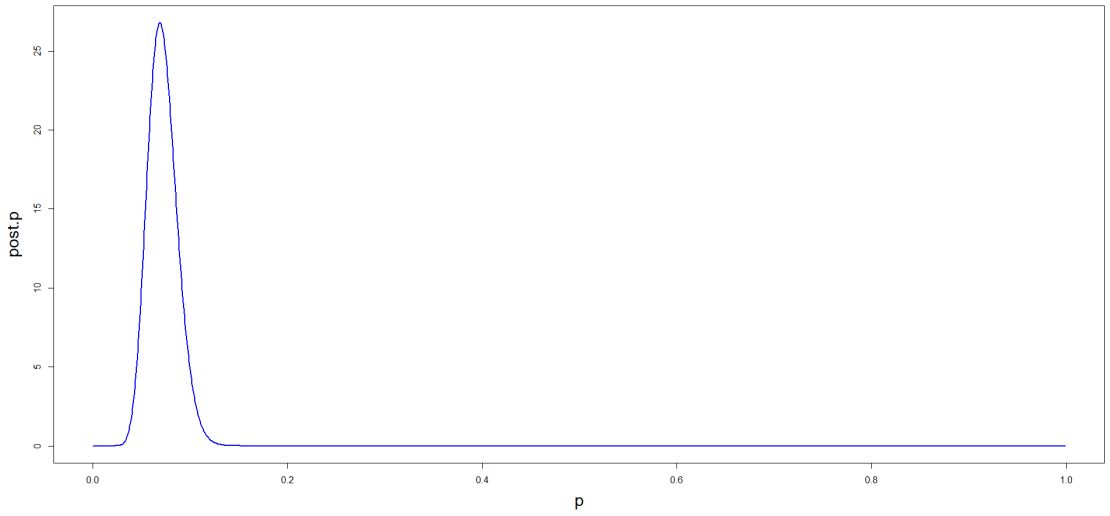
```
alpha_prior <- 1
beta_prior <- 10

alpha_posterior <- alpha_prior + y
beta_posterior <- beta_prior + (n - y)
p = seq(from = 0 , by = 0.001 , length.out = 1000)

posterior_samples <- dbeta(p, alpha_posterior, beta_posterior)
```

```r
# Plotting the posterior distribution
plot(p, posterior_samples, type = 'n', xlab = "p", ylab = "Density", main =
"Posterior Distribution of p")
polygon(c(p, rev(p)), c(posterior_samples, rep(0, length(p))), col = "white",
border = NA)
lines(p, posterior_samples,lwd=2, col = "blue",cex.main=1.8,cex.lab=1.8)
```



Now we want to assume the posterior probability of the older measurement as the prior for the new one.

```r
#The shape parameters are 1 + 11 + 9, representing the sum of prior
successes, current successes, and future successes. The rate parameters are
165 - 9 + 116 - 11 + 10, representing the sum of prior failures, current
failures, and future failures.

post.p = dbeta(p, 1 + 11 + 9 ,165 - 9 +116 -11 + 10)

plot(p , post.p , type = 'l', lwd = 2 ,  col
="blue",cex.main=1.8,cex.lab=1.8)
```



(g) find the bayesian estimator for p, the posterior mean and variance, and a 95% credible interval:

```
n <- nrow(chain)
sum <- sum( posterior_samples * p)
mean <- sum / n

variance = 0.001 * sum(p ^ 2 * posterior_samples) - mean ^ 2
sigma = sqrt(variance)

cat("mean: ", round(mean, 5), "\n")

## mean:  0.00057

cat("variance: ", round(variance, 5), "\n")

## variance:  0.00353

print(c(mean - 2 * sigma , mean + 2 * sigma))

## [1] -0.1182721  0.1194084
```

(h) test the hypotesis

$$H_\circ : p = 0.1 \text{ versus } H_1 : p \neq 0.1$$

```
p_value <- pbinom(9 , n , 0.1 , lower.tail = T) + pbinom(107 , n , 0.1 ,
lower.tail = F)
p_null <- 0.1

# Compare the p-value with the significance level
alpha <- 0.05
if (p_value < alpha) {
```

```
  cat("Reject the null hypothesis H₀: p =", p_null, "\n")
} else {
  cat("Fail to reject the null hypothesis H₀: p =", p_null, "\n")
}

## Fail to reject the null hypothesis H₀: p = 0.1

cat("p-value:", p_value, "\n")

## p-value: 1
```

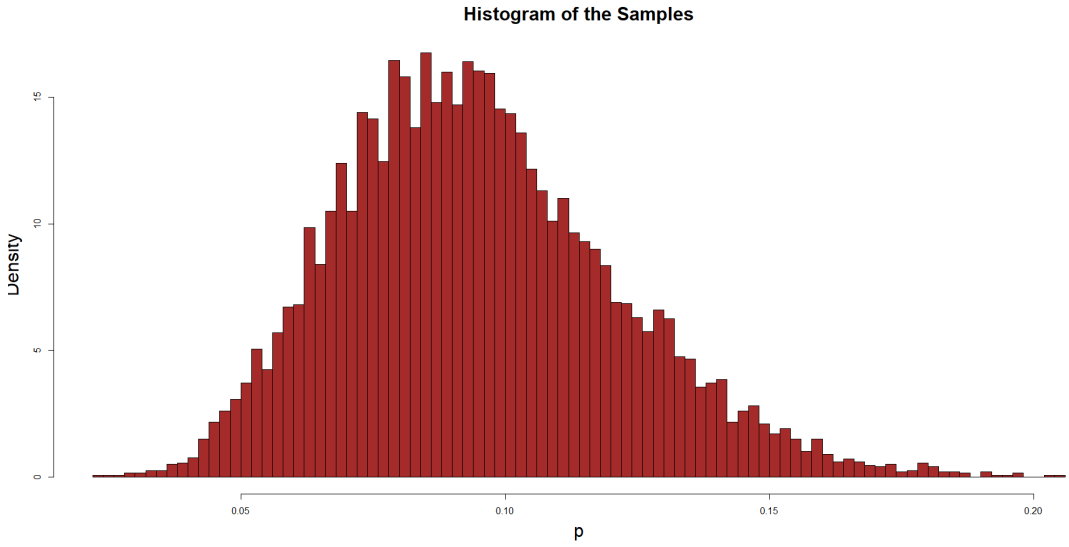
at 5% level of significance with both the frequentist and bayesian approach

Exercise 4

we want to analyze the data of Exercise 3 and solve points (b) and (c) using a MCMC with JAGS:

```
# First, we load the required packages for performing Bayesian inference with
JAGS.
#library(rjags)  # For running JAGS models
#library(coda)   # For analyzing and visualizing MCMC samples

# Next, we specify the data that will be used in the analysis.
# In this case, we have 'n' observations and 'y' successes.
data <- list()
data$n <- 116
data$y <- 11

# We create the JAGS model by calling the 'jags.model()' function.
# The model specification is provided in a separate file called "model.txt".
model <- jags.model("model.txt", data = data)

## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 1
##    Unobserved stochastic nodes: 1
##    Total graph size: 5
##
## Initializing model

#model {
# Prior distribution for p
#p ~ dbeta(1, 10)

# Likelihood
#y ~ dbin(p, n)
#}
```

```r
# We generate samples from the model using the 'coda.samples()' function.
# The 'variable.names' argument specifies the parameters of interest, in this case, 'p'.
# The 'n.iter' argument determines the number of iterations to run the sampler.
samples <- coda.samples(model, variable.names = "p", n.iter = 10000)

# We can then summarize the samples using the 'summary()' function to obtain
# various statistics such as mean, median, standard deviation, and quantiles.

# Next, we convert the samples to a data frame using the 'as.data.frame()' function
# from the 'coda' package. This allows for easier analysis and visualization.
p.df <- as.data.frame(as.mcmc(samples))

# We can plot a histogram of the samples using the 'hist()' function.
# The histogram provides an estimate of the probability distribution of the parameter 'p'.
hist(p.df$p, breaks = 100, freq = FALSE, main = "Histogram of the Samples",
     xlab = "p", col = "brown", cex.main = 2, cex.lab = 2)
```




```r
# Calculate mean, standard deviation, and 95% credible interval
mean <- mean(p.df$p)
std <- sd(p.df$p)
ci <- c(mean - 2 * std, mean + 2 * std)

cat("Mean: ", mean, "\n")

## Mean:  0.09480804

cat("Standard Deviation: ", std, "\n")

## Standard Deviation:  0.02609114
```

```r
cat("95% Credible Interval: [", ci[1], ", ", ci[2], "]\n")
```

```
## 95% Credible Interval: [ 0.04262575 ,  0.1469903 ]
```