

# Assignment Covering Chapters 2 & 3 of *Bayesian Computation with R*

Chris Hayduk

March 25, 2019

1. **Chapter 2: Introduction to Bayesian Thinking** Write your own code for the following:
  - (a) Simulating a sample from a posterior distribution when you have a histogram prior.
  - (b) Prior predictive density function, `pdispc()`.
  - (c) Prior predictive density function, `pbetap()`.
  - (d) Discrete credible interval, `discint`.

## Solution:

- (a) Let's start by defining a function to generate the values for the histogram prior:

```
#Function to generate prior values for each value in x
get_histprior_value <- function(x, histprior){
  vec <- rep(NA, length=length(x))
  for(i in 1:length(vec)){
    new_vec <- histprior[histprior[,1]>=x[i],]

    #Check if new_vec is matrix or vector and index accordingly
    if(!is.null(ncol(new_vec))){
      vec[i] <- new_vec[1,2]
    } else{
      vec[i] <- new_vec[2]
    }
  }
  return(vec)
}
```

Now let's prepare the data for input into our new `get_histprior_value` function:

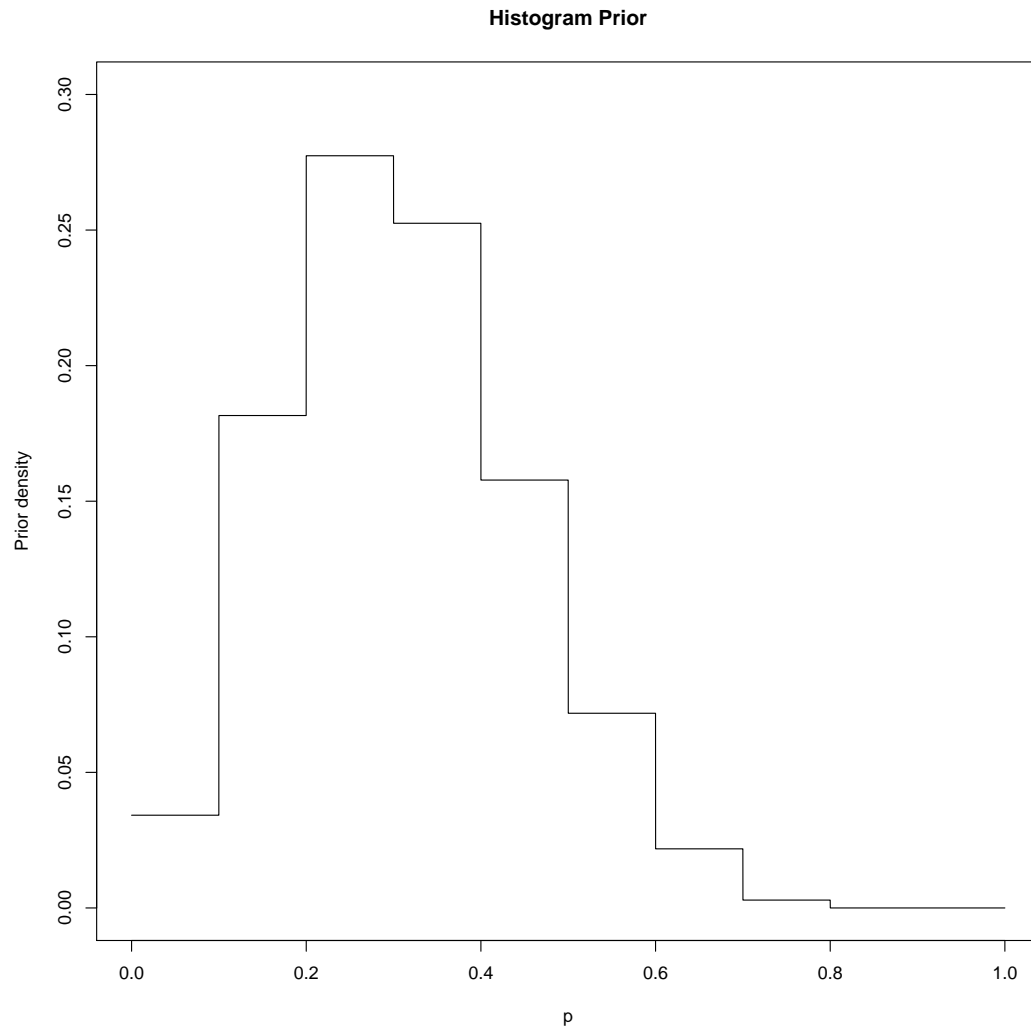
```
#Generate points for interval
interval <- seq(0.1, 1, by = 0.1)

#Prior probability
prior <- c(1, 5.2, 8, 7.2, 4.6, 2.1, 0.7, 0.1, 0, 0)
prior <- prior/sum(prior)

#Create the histogram prior
histprior <- sample(interval, 10000, replace=TRUE, prob = prior)
histprior <- table(histprior)/sum(table(histprior))
histprior <- as.matrix(histprior)
names <- rownames(histprior)
rownames(histprior) <- NULL
histprior <- cbind(as.numeric(names), as.numeric(histprior))
histprior <- rbind(histprior, c(0.9, 0))
histprior <- rbind(histprior, c(1.0, 0))
```

Now let's plot the prior:

```
#Output histogram of prior  
curve(get_histprior_value(x, histprior), from=0, to = 1,  
      ylim = c(0, 0.3), n = 10000,  
      xlab="p", ylab = "Prior density", main = "Histogram Prior")
```



Finally, let's generate and plot the posterior:

```
s <- 11
f <- 16

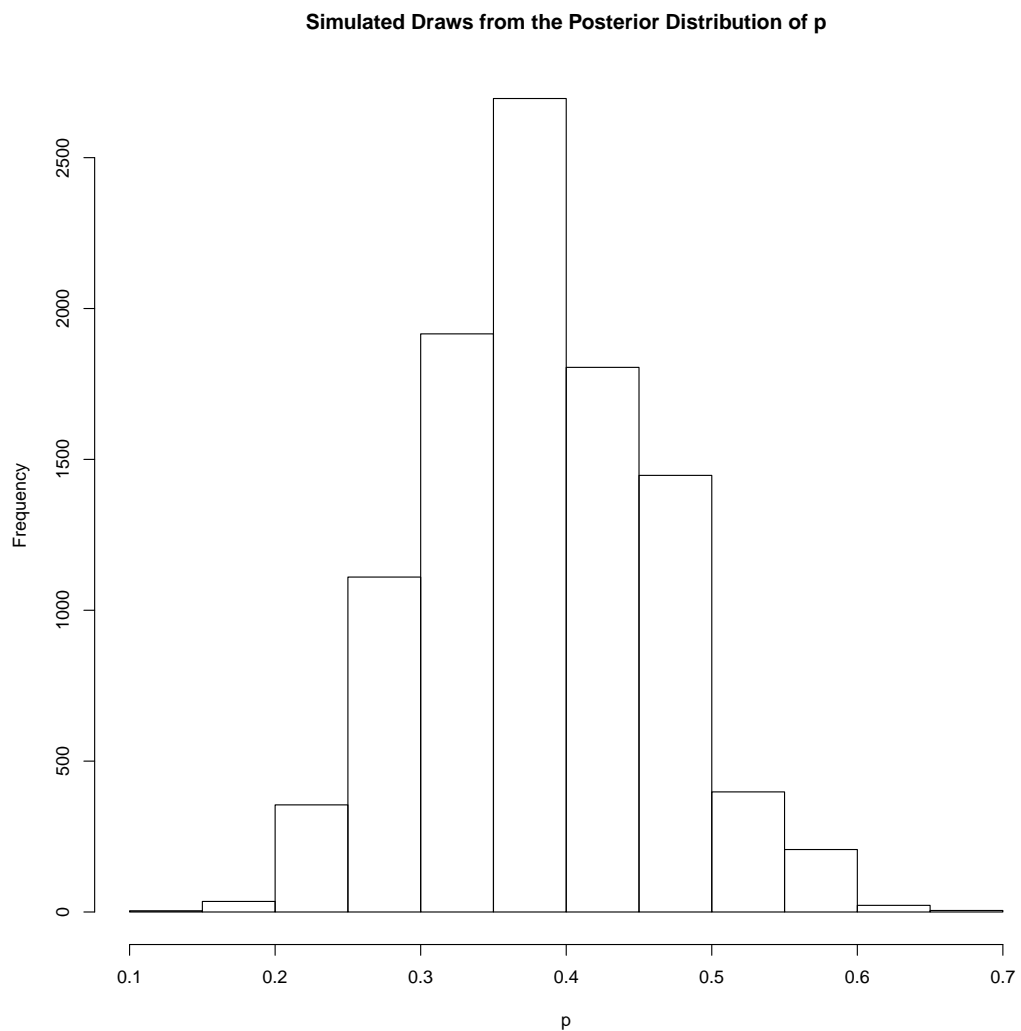
p <- seq(0, 1, length = 10000)

post <- get_histprior_value(p, histprior) * dbeta(p, s+1, f+1)

post <- post/sum(post)

ps <- sample(p, replace = TRUE, prob = post)

hist(ps, xlab="p", main="Simulated Draws from the Posterior Distribution of p")
```



(b) Let's start with `pdiscp()`:

```
pdiscp <- function(p, prior, m, ys){  
  pred <- rep(NA, length = length(ys))  
  
  #Loop through possible y values  
  for(i in 1:length(ys)){  
    val <- 0  
    #Loop through possible proportions in discrete prior  
    for(j in 1:length(p)){  
      f <- choose(m, ys[i]) * (p[j])^(ys[i]) * (1-p[j])^(m-ys[i])  
      g <- prior[j]  
      val <- val + (f*g)  
    }  
    pred[i] <- val  
  }  
  
  #Return vector of probabilities for each y value  
  return(pred)  
}
```

```

#Test run
p <- seq(0.05, 0.95, by=0.1)
prior <- c(1, 5.2, 8, 7.2, 4.6, 2.1, 0.7, 0.1, 0, 0)
prior <- prior/sum(prior)
m <- 20
ys <- 0:20
pred <- pdiscp(p, prior, m, ys)

round(cbind(0:20, pred), 3)

##          pred
## [1,] 0 0.020
## [2,] 1 0.044
## [3,] 2 0.069
## [4,] 3 0.092
## [5,] 4 0.106
## [6,] 5 0.112
## [7,] 6 0.110
## [8,] 7 0.102
## [9,] 8 0.089
## [10,] 9 0.074
## [11,] 10 0.059
## [12,] 11 0.044
## [13,] 12 0.031
## [14,] 13 0.021
## [15,] 14 0.013
## [16,] 15 0.007
## [17,] 16 0.004
## [18,] 17 0.002
## [19,] 18 0.001
## [20,] 19 0.000
## [21,] 20 0.000

```

(c) Now let's try `pbetap()`:

```
pbetap <- function(ab, m, ys){
  pred <- rep(NA, length = length(ys))

  for(i in 1:length(ys)){
    choose <- choose(m, ys[i])

    beta_num <- beta(ab[1] + ys[i], ab[2] + m - ys[i])

    beta_denom <- beta(ab[1], ab[2])

    pred[i] <- choose*(beta_num/beta_denom)
  }

  #Return vector of probabilities for each y value
  return(pred)
}
```

```
#Test run
ab <- c(3.26, 7.19)
m <- 20
ys <- 0:20
pred <- pbetap(ab, m, ys)
round(cbind(0:20, pred), 3)
```

```
##          pred
## [1,] 0 0.018
## [2,] 1 0.045
## [3,] 2 0.072
## [4,] 3 0.095
## [5,] 4 0.108
## [6,] 5 0.114
## [7,] 6 0.111
## [8,] 7 0.102
## [9,] 8 0.088
## [10,] 9 0.073
## [11,] 10 0.057
## [12,] 11 0.043
## [13,] 12 0.030
## [14,] 13 0.020
## [15,] 14 0.012
## [16,] 15 0.007
## [17,] 16 0.004
## [18,] 17 0.002
## [19,] 18 0.001
## [20,] 19 0.000
## [21,] 20 0.000
```

(d) Finally, here's discint:

```
discint <- function(dist, covprob){
  total_prob <- 0
  y_list <- c()

  #Find max probability value
  #Centers credible interval around y value with max probability
  i <- which.is.max(dist[,2])
  total_prob <- total_prob + dist[i,2]
  y_list <- c(y_list, dist[i,1])

  i_prev <- i-1
  i_next <- i+1

  while(total_prob < covprob){
    #Check if previous row is within range of matrix
    if(i_prev >= 1 & i_prev <= nrow(dist)){
      prev_val <- dist[i_prev,2]
    } else{
      prev_val <- -1
    }

    #Check if next row is within range of matrix
    if(i_next >= 1 & i_next <= nrow(dist)){
      next_val <- dist[i_next,2]
    } else{
      next_val <- -1
    }

    #Compare values for i_next and i_prev
    if(dist[i_next,2] >= dist[i_prev,2]){
      total_prob <- total_prob + dist[i_next, 2]
      y_list <- c(y_list, dist[i_next, 1])
      i_next <- i_next + 1
    } else{
      total_prob <- total_prob + dist[i_prev, 2]
      y_list <- c(y_list, dist[i_prev, 1])
      i_prev <- i_prev - 1
    }
  }

  output <- list(prob = total_prob, set = sort(y_list))

  return(output)
}
```



```

#Test run
p <- rbeta(1000, 3.26, 7.19)
y <- rbinom(1000, 20, p)
freq <- table(y)
ys <- as.integer(names(freq))
predprob <- freq/sum(freq)

dist <- cbind(ys, predprob)
covprob <- 0.9

discint(dist, covprob)

## $prob
## [1] 0.903
##
## $set
## [1] 1 2 3 4 5 6 7 8 9 10 11

```

2. **Chapter 2: Introduction to Bayesian Thinking** Show how the author derived the predictive density formula below with a  $\text{beta}(a, b)$  prior (pg. 31 of book):

$$\begin{aligned}
 f(\tilde{y}) &= \int f_B(\tilde{y}|m, p)g(p)dp \\
 &= \binom{m}{\tilde{y}} \frac{B(a + \tilde{y}, b + m - \tilde{y})}{B(a, b)}
 \end{aligned}$$

where  $\tilde{y} = 0, \dots, m$ .

**Solution:**

$$\begin{aligned}
 f(\tilde{y}) &= \int f_B(\tilde{y}|m, p)g(p)dp \\
 &= \int_0^1 \left( \binom{m}{\tilde{y}} p^{\tilde{y}}(1-p)^{m-\tilde{y}} \left( \int_0^1 p^{a-1}(1-p)^{b-1}dp \right) \right) dp \\
 &= \binom{m}{\tilde{y}} \int_0^1 \left( p^{\tilde{y}}(1-p)^{m-\tilde{y}} \left( \int_0^1 p^{a-1}(1-p)^{b-1}dp \right) \right) dp \\
 &= \binom{m}{\tilde{y}} \int_0^1 (p^{\tilde{y}}(1-p)^{m-\tilde{y}} (B(a, b))) dp \\
 &= \binom{m}{\tilde{y}} B(a, b) \int_0^1 p^{\tilde{y}}(1-p)^{m-\tilde{y}} dp \\
 &= \binom{m}{\tilde{y}} B(a, b) B(\tilde{y} + 1, m - \tilde{y} + 1)
 \end{aligned}$$

Not sure where to go from here. **Come back to this one later.**

### 3. Chapter 2: Introduction to Bayesian Thinking Exercise 5.

#### Solution:

```
mu <- c(20, 30, 40, 50, 60, 70)
prior <- c(.1, .15, .25, .25, .15, .1)
prior <- prior/sum(prior)

y <- c(38.6, 42.4, 57.5, 40.5, 51.7, 67.1, 33.4, 60.9, 64.1, 40.1, 40.7, 6.4)
ybar <- mean(y)

n <- length(y)
sigma_squared <- var(y)

like <- exp(-(n/(2*sigma_squared))*(mu - ybar)^2)
post <- prior*like/sum(prior*like)

dist <- cbind(mu, post)
discint(dist, 0.8)

## $prob
##      post
## 0.9921244
##
## $set
## mu mu
## 40 50
```

#### 4. Chapter 2: Introduction to Bayesian Thinking Exercise 6.

##### Solution:

```
#a)
lambda <- c(0.5, 1, 1.5, 2, 2.5, 3)
prior <- c(0.1, 0.2, 0.3, 0.2, 0.15, 0.05)
prior <- prior/sum(prior)

pred <- rep(NA, length = length(lambda))
y <- 12

for(i in 1:length(lambda)){
  pred[i] <- prior[i]*exp(-6*lambda[i])*(6*lambda[i])^12
}
pred <- pred/sum(pred)
pred <- cbind(lambda, pred)
print(round(pred,5))

##      lambda    pred
## [1,]    0.5 0.00009
## [2,]    1.0 0.03679
## [3,]    1.5 0.35652
## [4,]    2.0 0.37357
## [5,]    2.5 0.20299
## [6,]    3.0 0.03004

#b)
predictive_prob <- rep(NA, length = length(lambda))
for(i in 1:length(lambda)){
  predictive_prob[i] <- exp(-7*lambda[i])*pred[i]
}

sum(predictive_prob)

## [1] 0.01605361
```

5. **Chapter 3: Single-Parameter Models** Write your own versions of the functions `binomial.beta.mix()` and `pbetat()`.

**Solution:**

```
binomial.beta.mix <- function(probs, betapar, data){
  m = data[1] + data[2]

  lambda_data <- (probs[1] * pbetap(betapar[1,], m, data[1]))/
    (probs[1] * pbetap(betapar[1,], m, data[1]) +
     probs[2] * pbetap(betapar[2,], m, data[1]))

  posterior <- c()
  posterior$probs <- matrix(c(lambda_data, (1-lambda_data)), nrow=1, ncol=2)
  colnames(posterior$probs) <- c("beta.par1", "beta.par2")

  posterior$betapar <- rbind(c(betapar[1,1] + data[1], betapar[1,2] + data[2]),
                           c(betapar[2,1] + data[1], betapar[2,2] + data[2]))
  rownames(posterior$betapar) <- c("beta.par1", "beta.par2")
  colnames(posterior$betapar) <- c("", "")

  return(posterior)
}
```

Here is an example run using the values from the book (p. 51-52):

```
#Example run
probs <- c(.5, .5)
beta.par1 <- c(6, 14)
beta.par2 <- c(14, 6)
betapar <- rbind(beta.par1, beta.par2)
data <- c(7,3)

post <- binomial.beta.mix(probs, betapar, data)

post$probs

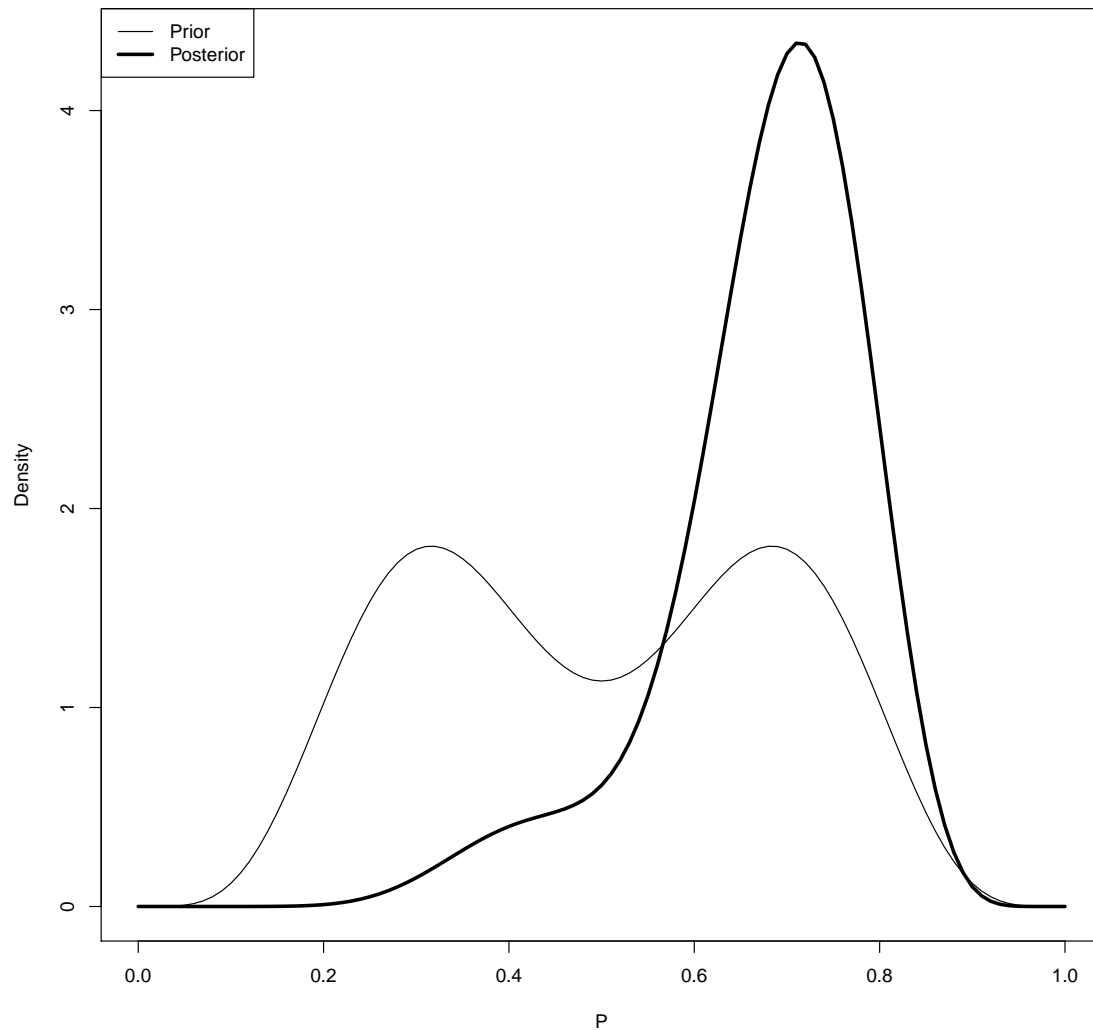
##          beta.par1 beta.par2
## [1,] 0.09269663 0.9073034

post$betapar

##
## beta.par1 13 17
## beta.par2 21  9
```

Now let's plot the posterior and the prior:

```
curve(post$probs[1]*dbeta(x,13,17) + post$probs[2]*dbeta(x,21,9),  
      from = 0, to = 1, lwd=3, xlab = "P", ylab = "Density")  
curve(0.5*dbeta(x,6,12)+0.5*dbeta(x,12,6),0,1,add=TRUE)  
legend("topleft", legend=c("Prior", "Posterior"), lwd=c(1,3))
```



6. **Chapter 3: Single-Parameter Models** On pg. 55, Albert writes, “One can show that the posterior probability that the coin is fair is given by:”

$$\lambda(y) = \frac{0.5P_0(Y \leq 5)}{0.5P_0(Y \leq 5) + 0.5P_1(Y \leq 5)}.$$

Show this.

**Solution:** Come back to this one later.

7. **Chapter 3: Single-Parameter Models** Do all Exercises at the end of the chapter.

**Solution:**

```
1. data <- c(0, 10, 9, 8, 11, 3, 3, 8, 8, 11)

theta <- seq(from = -2, to = 12, by = 0.1)

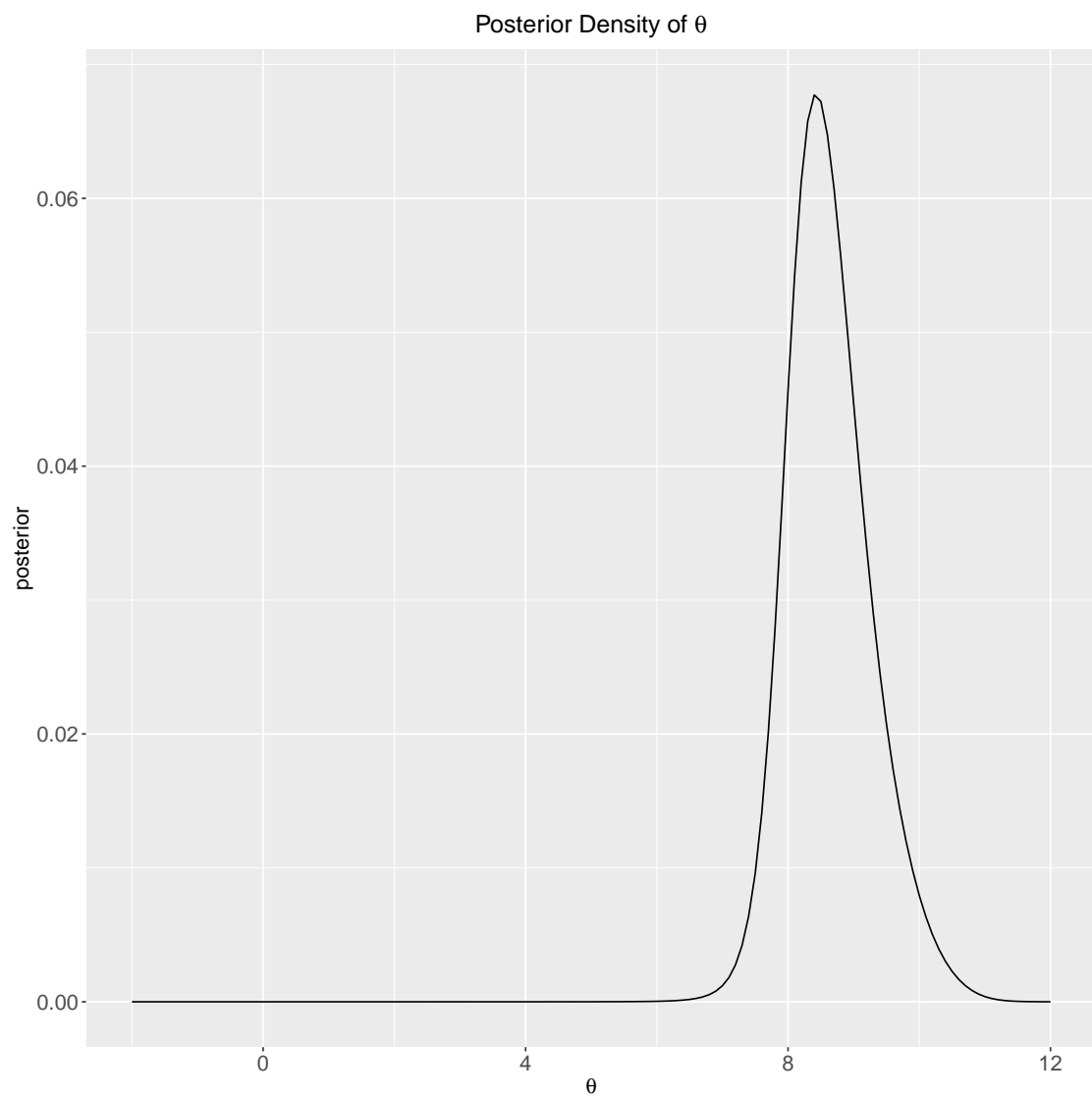
posterior_density <- rep(1, length(theta))

for(i in 1:length(theta)){
  for(j in 1:length(data)){
    posterior_density[i] <- posterior_density[i] * (1/(1+((data[j] - theta[i])^2)))
  }
}

posterior_density <- posterior_density/sum(posterior_density)

df <- data.frame(theta = theta, posterior = posterior_density)

ggplot(df, aes(x = theta, y = posterior)) +
  geom_line() +
  xlab(TeX('$\\theta$')) +
  ggtitle(TeX('Posterior Density of $\\theta$')) +
  theme.info
```



```
samples <- sample(theta, 100000, replace = TRUE, prob = posterior_density)

#Posterior mean
mean(samples)

## [1] 8.632004

#Posterior standard deviation
sd(samples)

## [1] 0.6475669
```

2. Let  $\theta = 1/\lambda = \lambda^{-1}$ . Then we have  $\lambda = \theta^{-1}$  and:

$$\begin{aligned}\lambda^{-n-1}e^{-s/\lambda} &= (\theta^{-1})^{-n-1}e^{-s/(\theta^{-1})} \\ &= \theta^{-1(-n-1)}e^{-s\theta} \\ &= \theta^{n+1}e^{-s\theta}\end{aligned}$$

So,  $\theta$  has a gamma density with shape parameter  $n + 2$  and rate parameters  $s$ .

```
burn_times <- c(751, 594, 1213, 1126, 819)

n <- length(burn_times)
s <- sum(burn_times)

theta_draws <- rgamma(1000, shape = n+2, rate = s)

lambda_draws <- 1/theta_draws

greater_than_1000 <- sum(lambda_draws > 1000)

posterior_prob_greater_than_1000 <- greater_than_1000/length(lambda_draws)

print(posterior_prob_greater_than_1000)

## [1] 0.184
```

3. We know  $100 \leq N \leq 200$ . So,

```
N <- seq(from = 100, to = 200, by = 1)
post <- rep(0, length = length(N))

n <- length(y)

for(i in 1:length(post)){
  post[i] <- 1/(N[i]^n)
}

samples <- sample(N, 100000, replace = TRUE, prob = post)

#Posterior mean
mean(samples)

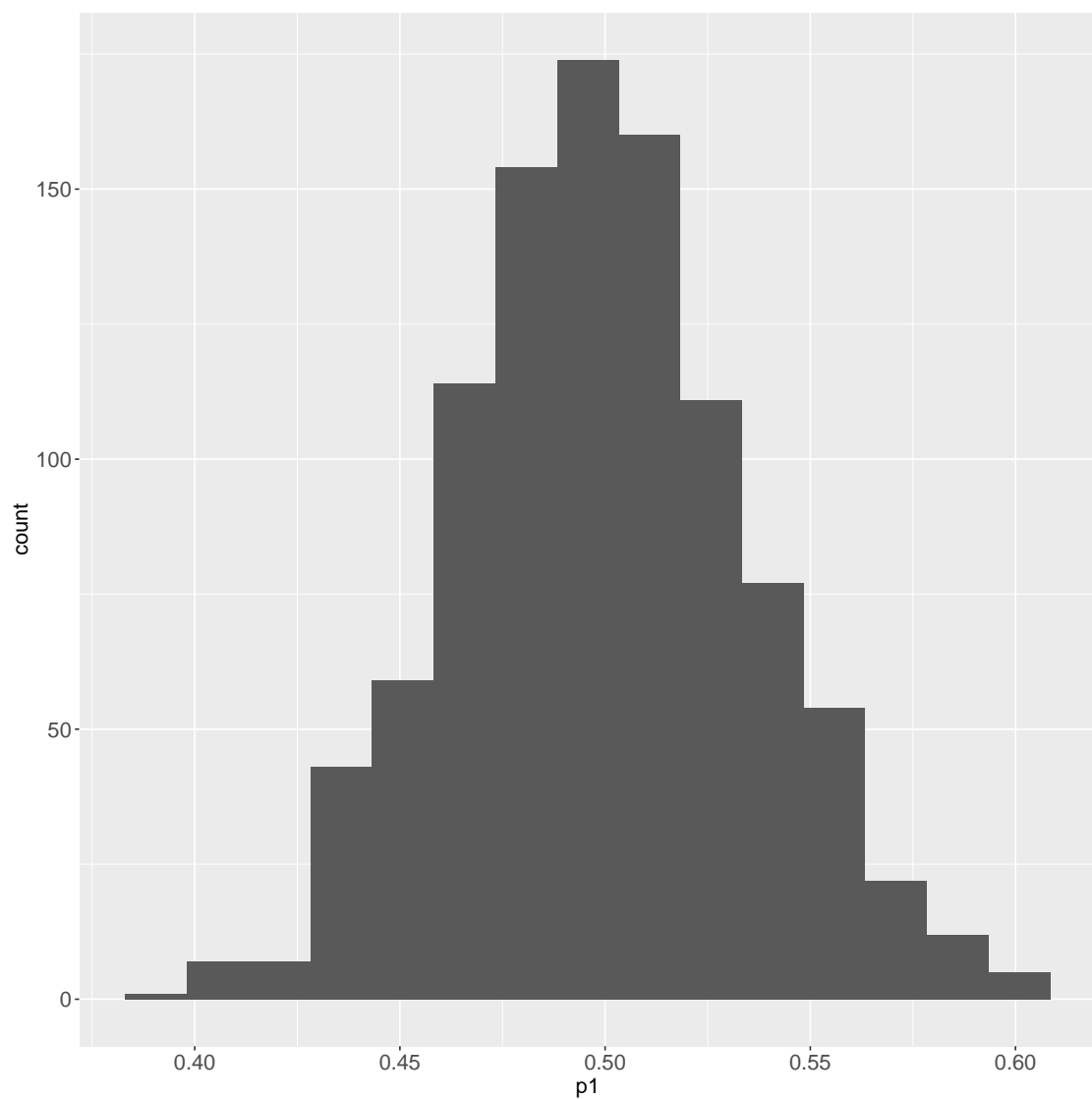
## [1] 144.1586

#Posterior standard deviation
sd(samples)

## [1] 29.04718
```



```
4. p1 <- rbeta(1000, shape1 = 100, shape2 = 100)
df1 <- data.frame(p1 = p1)
ggplot(data = df1, aes(x=p1)) + geom_histogram(bins = 15) + theme.info
```



```
#Mean of P1. Target is 0.5
mean(p1)

## [1] 0.4992613

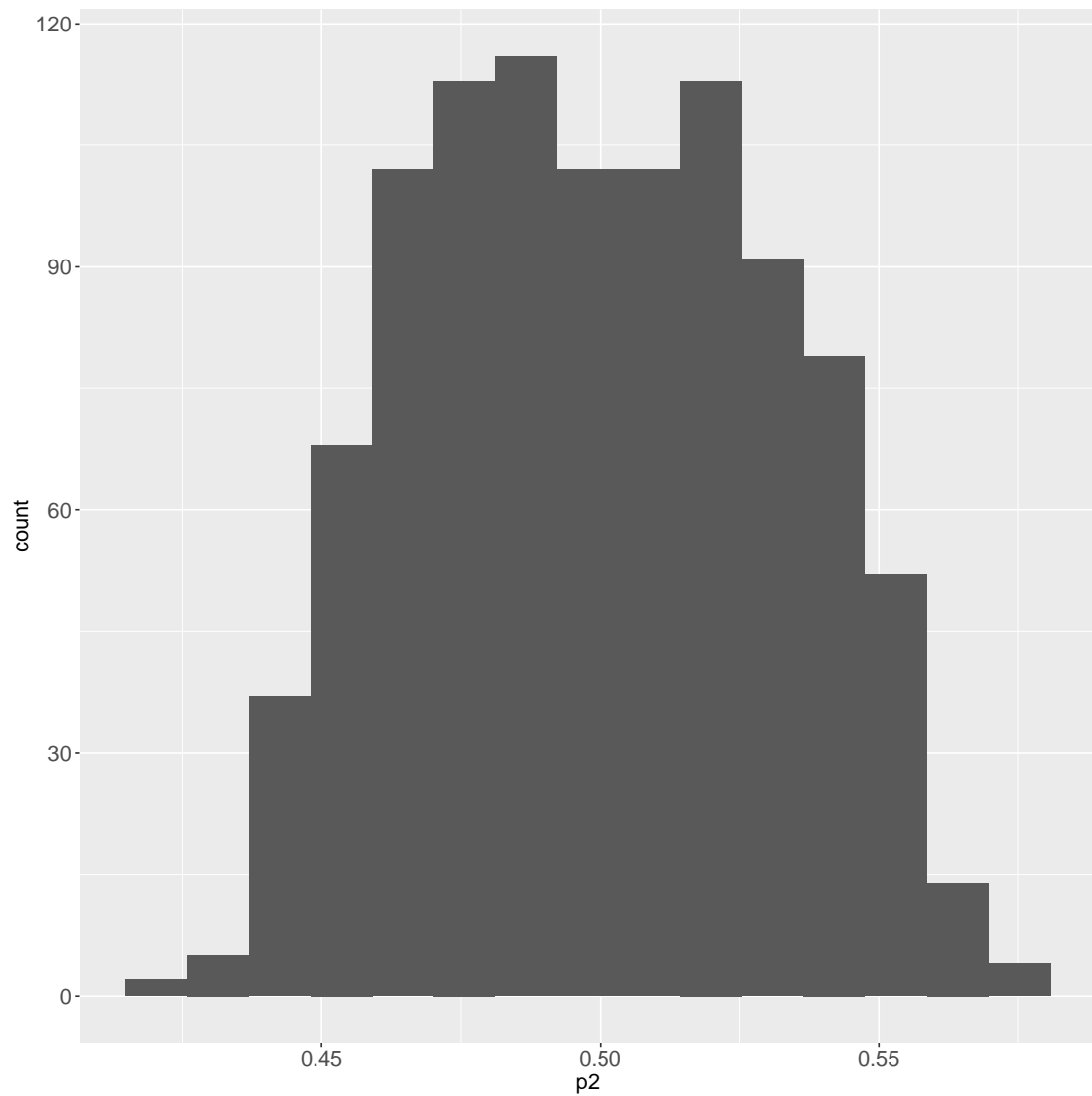
#Probability that 0.44 < p < 0.56. Target is a probability of 0.9
(sum(p1 < 0.56) - sum(p1 < 0.44))/length(p1)

## [1] 0.902

p2 <- 0.9*rbeta(1000, 500, 500) + 0.1*rbeta(1000, 1, 1)

df2 <- data.frame(p2 = p2)

ggplot(data = df2, aes(x=p2)) + geom_histogram(bins = 15) + theme.info
```



```
#Mean of P2. Target is 0.5  
mean(p2)
```

```
## [1] 0.4994086
```

```
#Probability that 0.44 < p < 0.56. Target is a probability of 0.9  
(sum(p2 < 0.56) - sum(p2 < 0.44))/length(p2)
```

```
## [1] 0.974
```

```
#Continue this one later
```

5.