Bayesian Learning (732A91) Lab1

Anubhav Dikshit(anudi287) and Lennart Schilling (lensc874)

03 April, 2019

Contents

Question 1: Bernoulli ... again. **2**

a) Draw random numbers from the posterior θ|y ~ Beta(α_0 + s; β_0 + f), y = (y1,y2,y3....yn) and verify graphically that the posterior mean and standard deviation converges to the true values as the number of random draws grows large. 2

b) Use simulation (nDraws = 10000) to compute the posterior probability Pr($\theta < 0.4$|y) and compare with the exact value [Hint: pbeta()]. 5

c) Compute the posterior distribution of the log-odds, $/phi = log\frac{\theta}{1-\theta}$ by simulation (nDraws = 10000). 6

Question 2: Log-normal distribution and the Gini coefficient **6**

b) The most common measure of income inequality is the Gini coefficient, G, where $0 \leq G \leq 1$. G = 0 means a completely equal income distribution, whereas G = 1 means complete income inequality. See Wikipedia for more information. It can be shown that $G = 2\phi(\frac{\sigma}{\sqrt{2}})$ when incomes follow a logN(μ, σ^2) distribution. ϕ(z) is the cumulative distribution function (CDF) for the standard normal distribution with mean zero and unit variance. Use the posterior draws in a) to compute the posterior distribution of the Gini coefficient G for the current data set. 9

c)Use the posterior draws from b) to compute a 95% equal tail credible interval for G. An 95% equal tail interval (a,b) cuts off 2.5% percent of the posterior probability mass to the left of a, and 97.5% to the right of b. Also, do a kernel density estimate of the posterior of G using the density function in R with default settings, and use that kernel density estimate to compute a 95% Highest Posterior Density interval for G. Compare the two intervals. 10

Question 3: Bayesian Inference **12**

a)Plot the posterior distribution of κ for the wind direction data over a fine grid of κ values. . . . 13

b)Find the (approximate) posterior mode of κ from the information in a). 14

Appendix **15**

Question 1: Bernoulli . . . again.

Let y1,y2,y3|θ ~ Bern(θ), and assume that you have obtained a sample with s = 14 successes in n = 20 trials. Assume a Beta(α_0,β_0) prior for θ and let $\alpha_0 = \beta_0 = 2$

a) Draw random numbers from the posterior θ|y ~ Beta(α_0 + s; β_0 + f), y = (y1,y2,y3. . . .yn) and verify graphically that the posterior mean and standard deviation converges to the true values as the number of random draws grows large.

Let us verify the result for better understanding

- PDF of Bernoulli distribution (unordered) is given by:

$$p(s|\theta) = \binom{N}{s}\theta^s \cdot (1-\theta)^{n-s} \quad where\ s\ is\ the\ number\ of\ success$$

- PDF of Beta distribution is given by:

$$f(\theta|\alpha,\beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\cdot\Gamma(\beta)}\theta^{\alpha-1}\cdot(1-\theta)^{\beta-1}$$
$$Where\ \Gamma(n+1) = (n)!$$

$$Mean: \frac{\alpha}{\alpha+\beta}$$
$$Standard\ Deviation: \sqrt{\frac{\alpha\cdot\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}}$$

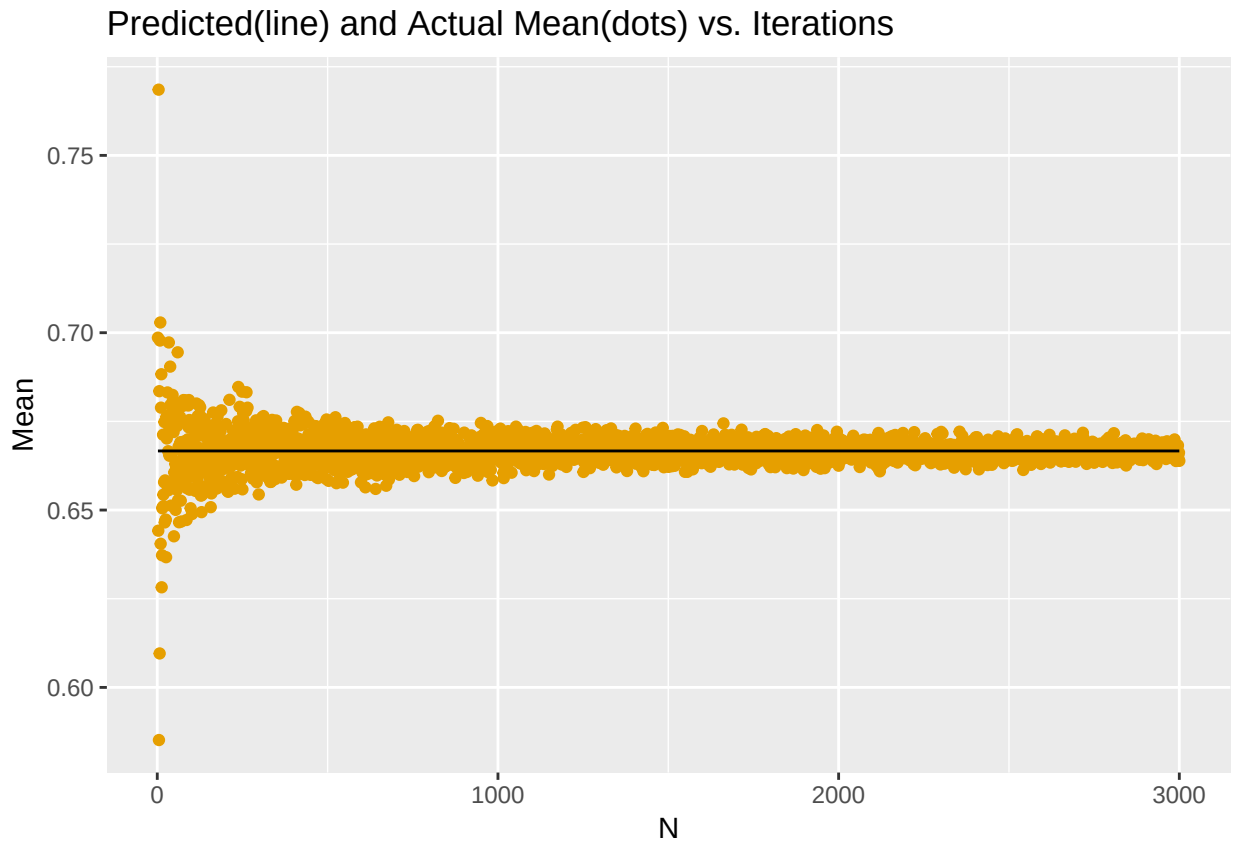
* Verification:

$$Likelihood = \binom{N}{s}\theta^s \cdot (1-\theta)^{n-s}$$
$$Prior = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\cdot\Gamma(\beta)}\theta^{\alpha-1}\cdot(1-\theta)^{\beta-1}$$
$$Prior \propto \theta^{\alpha-1}\cdot(1-\theta)^{\beta-1}$$

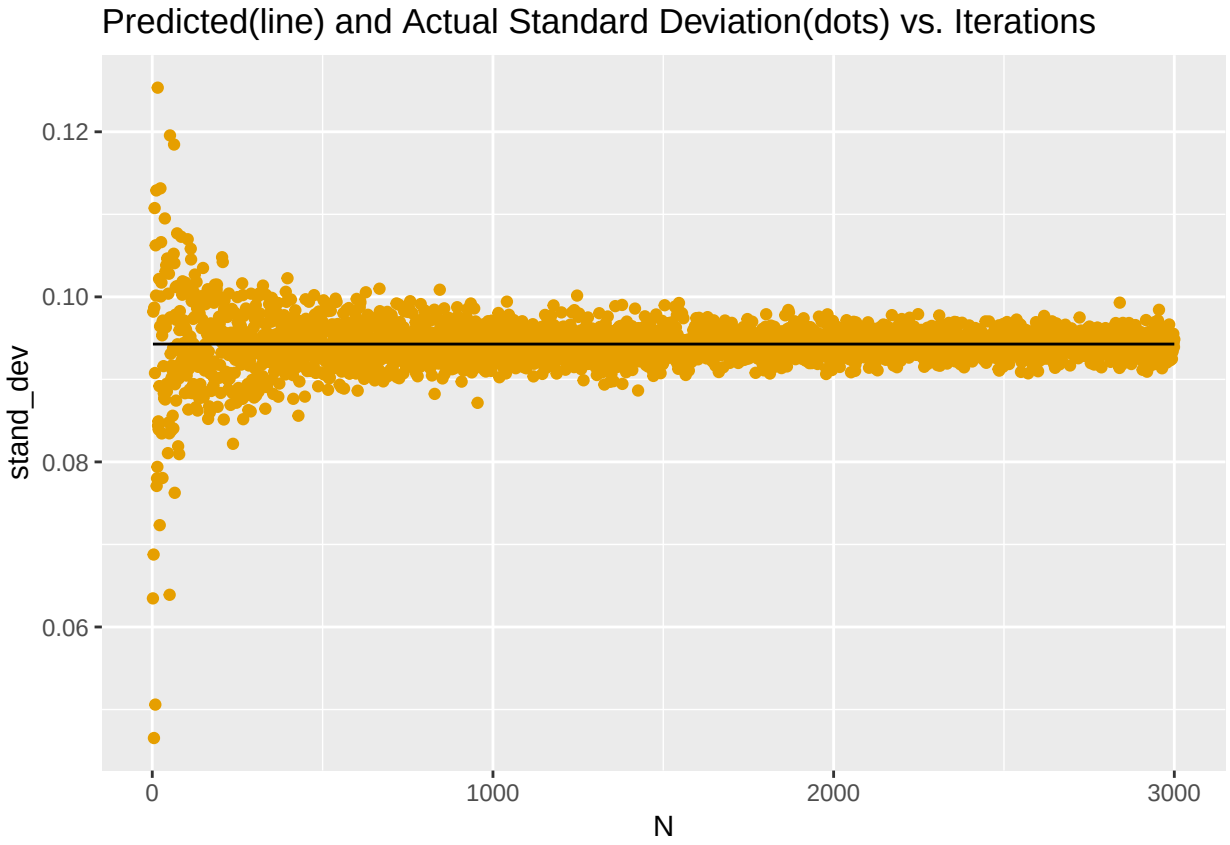
$$Posterior \propto Likelihood \times Prior$$
$$Posterior \propto Beta(\alpha_0+s, \beta_0+f)$$

```
#' Title A beta distribution generator to generate N values
#'
#' @param N Number of iterations
#' @param alpha Alpha value for beta distribution
#' @param beta  Beta value for beta distribution
#'
#' @return a list containing the mean and std dev of the generated vector
#' @export
#'
#' @examples
```

```r
gen_beta_sample <- function(N, alpha, beta){
  temp <- rbeta(N, shape1=alpha, shape2=beta)
  mean_value <- mean(temp)
  std_dev <- sd(temp)
  answer <- list(mean_value=mean_value, std_dev=std_dev)
  return(answer)
}

# required input
N = 3000
alpha = 16
beta = 8

# Running the loop inside a function to get the values for varying N
final <- NULL
for(i in 2:N){
  temp <- gen_beta_sample(N=i, alpha = alpha, beta=beta)
  df <- cbind(N = i, Mean = temp$mean_value, stand_dev = temp$std_dev)
  final <- rbind(df, final)
}

# Making final as a dataframe to make it easy to plot
final <- final %>% data.frame()
final$true_mean <- ((alpha)/(alpha+beta))
final$true_std_dev <- sqrt((alpha*beta)/((alpha+beta)^2 *(alpha+beta+1)))

#plots
ggplot(data=final, aes(x=N, y=Mean)) +
  geom_point(color= "#E69F00") +
  geom_line(aes(y=true_mean), color = "#000000") +
  ggtitle("Predicted(line) and Actual Mean(dots) vs. Iterations")
```




```
ggplot(data=final, aes(x=N, y=stand_dev)) +
  geom_point(color= "#E69F00") +
  geom_line(aes(y=true_std_dev), color = "#000000") +
  ggtitle("Predicted(line) and Actual Standard Deviation(dots) vs. Iterations")
```



b) Use simulation (nDraws = 10000) to compute the posterior probability Pr(θ < 0.4|y) and compare with the exact value [Hint: pbeta()].

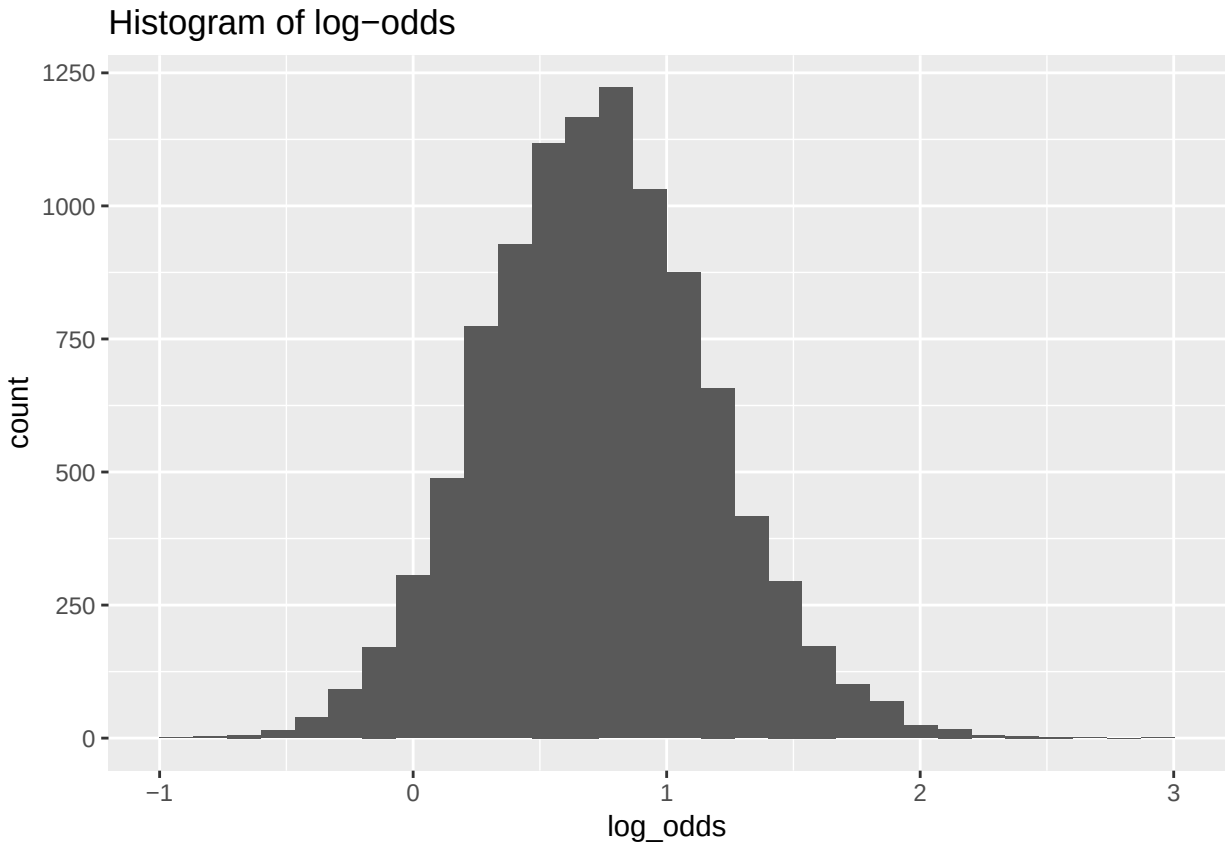
```
#' Title A beta distribution generator to generate N values without mean
#'
#' @param N Number of iterations
#' @param alpha Alpha value for beta distribution
#' @param beta  Beta value for beta distribution
#'
#' @return a list containing the mean and std dev of the generated vector
#' @export
#'
#' @examples
gen_beta_sample_2 <- function(N, alpha, beta){
  answer <- rbeta(N, shape1=alpha, shape2=beta)
  return(answer)
}

# required input
N = 10000

# Running the loop inside a function to get the values for varying N
beta_values <- gen_beta_sample_2(N=N, alpha = alpha, beta = beta)
actual_prop <- sum(beta_values < 0.4)/N
```

```r
true_prop <- pbeta(q=0.4, shape1=alpha, shape2=beta, lower.tail = TRUE)

cat("The true value of the probability is: ", true_prop, ",
    while the calculated value is: ", actual_prop)
```

```
## The true value of the probability is:  0.003972681 ,
##     while the calculated value is:  0.0044
```

c) Compute the posterior distribution of the log-odds, $/phi = log\frac{\theta}{1-\theta}$ by simulation (nDraws = 10000).

```r
log_odds = log(beta_values/(1-beta_values))
log_odds <- as.data.frame(log_odds)

ggplot(data=log_odds, aes(x=log_odds)) +
  geom_histogram(bins = 30) +
  ggtitle("Histogram of log-odds")
```



Question 2: Log-normal distribution and the Gini coefficient

Assume that you have asked 10 randomly selected persons about their monthly income (in thousands Swedish Krona) and obtained the following ten observations: 14, 25, 45, 25, 30, 33, 19, 50, 34 and 67. A common

model for non-negative continuous variables is the log-normal distribution. The log-normal distribution $\log N(\mu, \sigma^2)$ has density function

$$p(y|\mu, \sigma^2) = \frac{1}{y \cdot \sqrt{2\pi\sigma^2}} e[-\frac{(logy - \mu)^2}{2\sigma^2}]$$

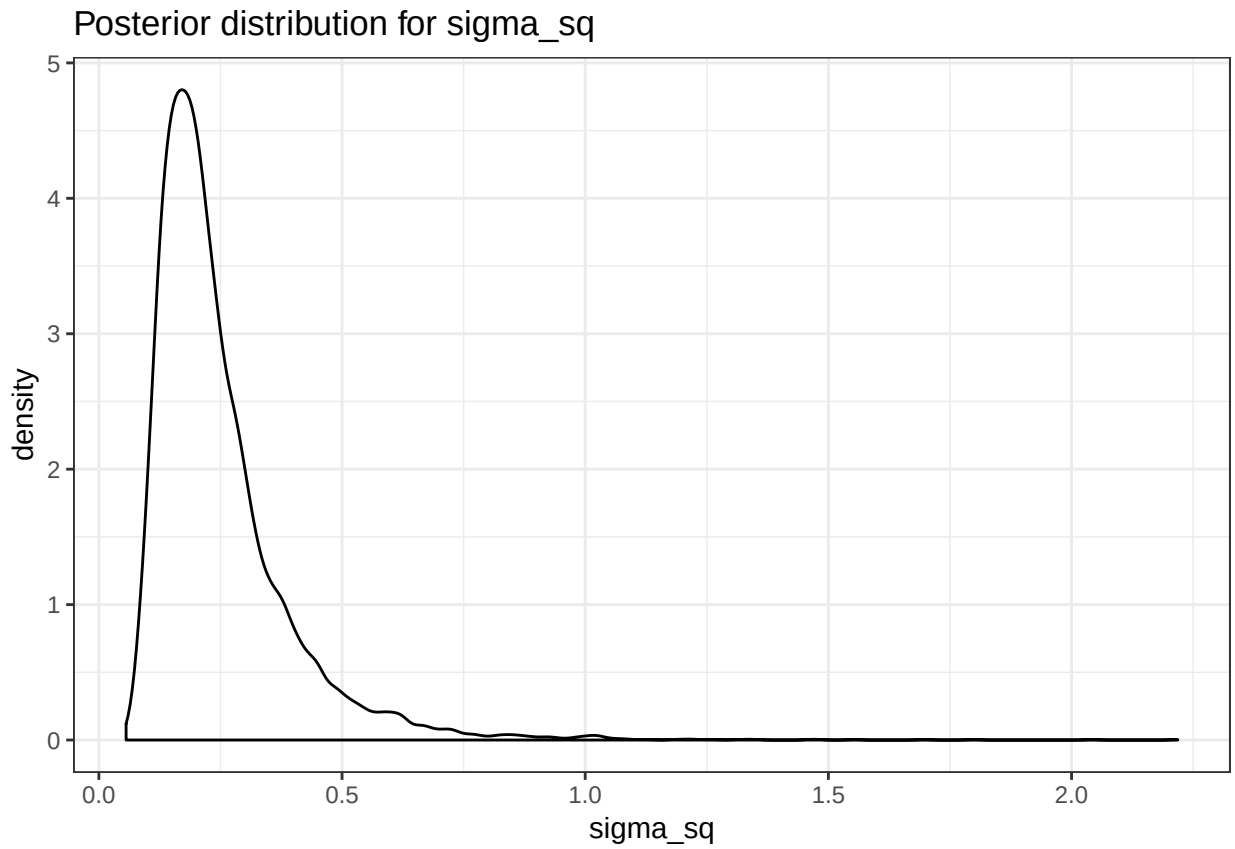
for y > 0, $\mu > 0$ and $\sigma^2 > 0$. The log-normal distribution is related to the normal distribution as follows: if y ~ logN(μ, σ^2) then log y ~ N(μ; σ^2). Let y1, y2, y3. . . .yn |μ, σ^2 iid ~ logN(μ, σ^2), where $\mu = 3.5$ is assumed to be known but σ^2 is unknown with non-informative prior p(σ^2) / 1=σ^2. The posterior for σ^2 is the Inv χ^2 (n, τ^2) distribution, where

$$\tau^2 = \frac{\sum_{i=1}^{n}(logy_i - \mu)^2}{n}$$

a) Simulate 10,000 draws from the posterior of σ^2 (assuming $\mu = 3.5$) and compare with the theoretical Inv - $\chi^2(n, \tau^2)$ posterior distribution.

In the following, τ^2 can be used to simulate from the posterior $\sigma^2|(y_1, ..., y_n); \mu \sim Inv - \chi^2(n, \tau^2)$. This will be done by first drawing $X \sim \chi^2(n)$. This drawed value will then be used within the formula $\sigma^2 = \frac{n\tau^2}{X}$ which is a draw from $Inv - \chi^2(n, \tau^2)$. This process will be repeated 10000 times. The obtained values for σ^2 will be stored.

```
y = c(14, 25, 45, 25, 30, 33, 19, 50, 34, 67)
n = length(y)
mu = 3.5
tau_sq = sum((log(y)-mu)^2)/n

sigma_sq = c()
for (i in 1:10000) {
  # Drawing x.
  x = rchisq(n = 1, df = n)
  # Calculating and storing sigma_sq.
  sigma_sq[i] = (n*tau_sq)/x
}
# Plotting simulated posterior distribution.
ggplot() +
  geom_density(aes(x = sigma_sq)) +
  labs(title = "Posterior distribution for sigma_sq") +
  theme_bw()
```



After the simulated posterior distribution has been plotted, it will be compared to the theoretical distribution. This will be done by a comparison of the mean and the standard deviation. The theoretical mean and standard deviation are obtained with $\frac{n\tau^2}{n-2}$ for $n > 2$ and $\sqrt{\frac{2n^2\tau^4}{(n-2)^2(n-4)}}$ for $n > 4$, respectively.

```
# Printing statistics of simulated distribution compared to theoretical values.
knitr::kable(
  as.data.frame(
    rbind(
      cbind(Posterior = "Simulation", Mean = mean(sigma_sq), Sd = sd(sigma_sq)),
      cbind(Posterior = "Theory", Mean = n*tau_sq/(n-2), Sd = sqrt((2*n^2*tau_sq^2)/(((n-2)^2)*(n-4))))
    )
  )
)
```

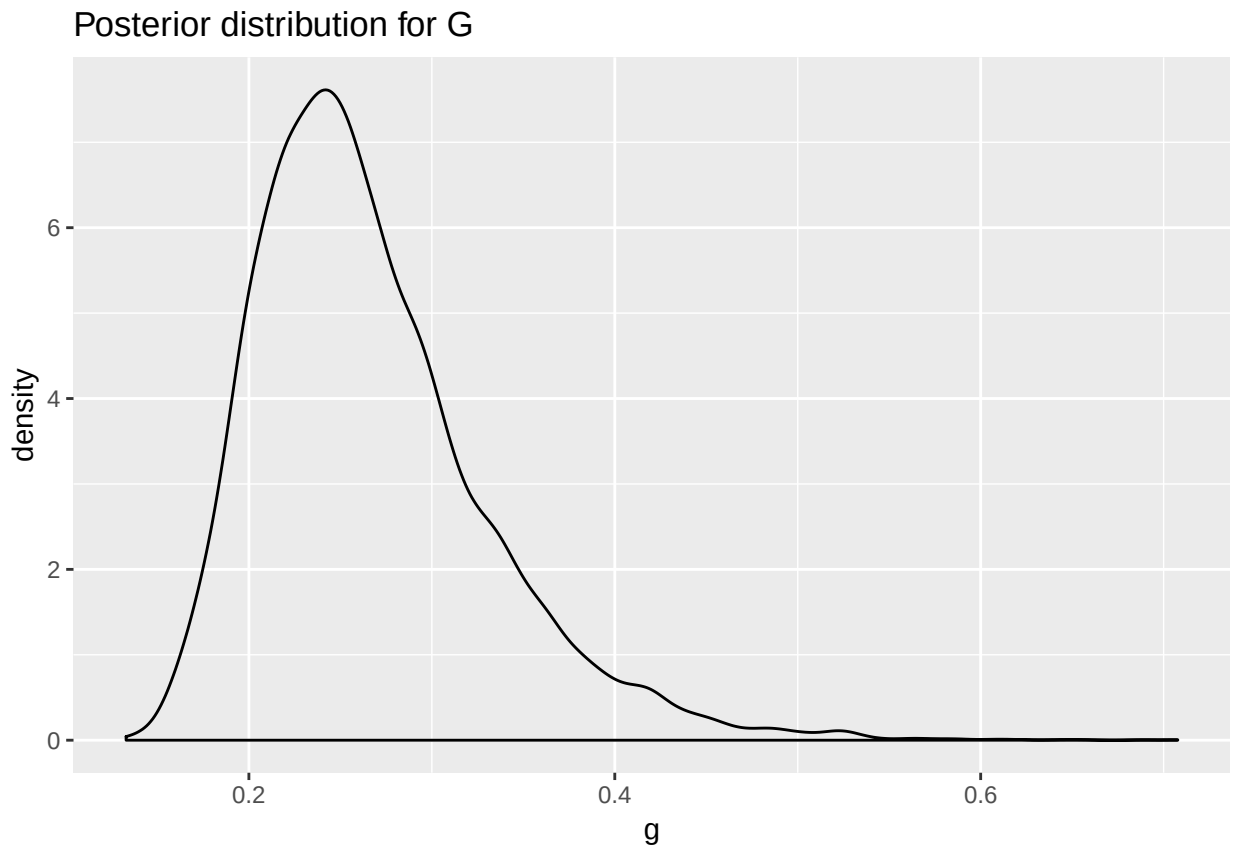
Posterior	Mean	Sd
Simulation	0.247548157149189	0.141310700422757
Theory	0.247349686559943	0.142807408119353

It can be seen that the statistics obtained with the simulation are very close to the theoretical values. Thus, we assume that the simulation of the posterior distribution for σ^2 has been successful.

b) The most common measure of income inequality is the Gini coefficient, G, where $0 \leq G \leq 1$. $G = 0$ means a completely equal income distribution, whereas $G = 1$ means complete income inequality. See Wikipedia for more information. It can be shown that $G = 2\phi(\frac{\sigma}{\sqrt{2}})$ when incomes follow a $\log N(\mu, \sigma^2)$ distribution. $\phi(z)$ is the cumulative distribution function (CDF) for the standard normal distribution with mean zero and unit variance. Use the posterior draws in a) to compute the posterior distribution of the Gini coefficient G for the current data set.

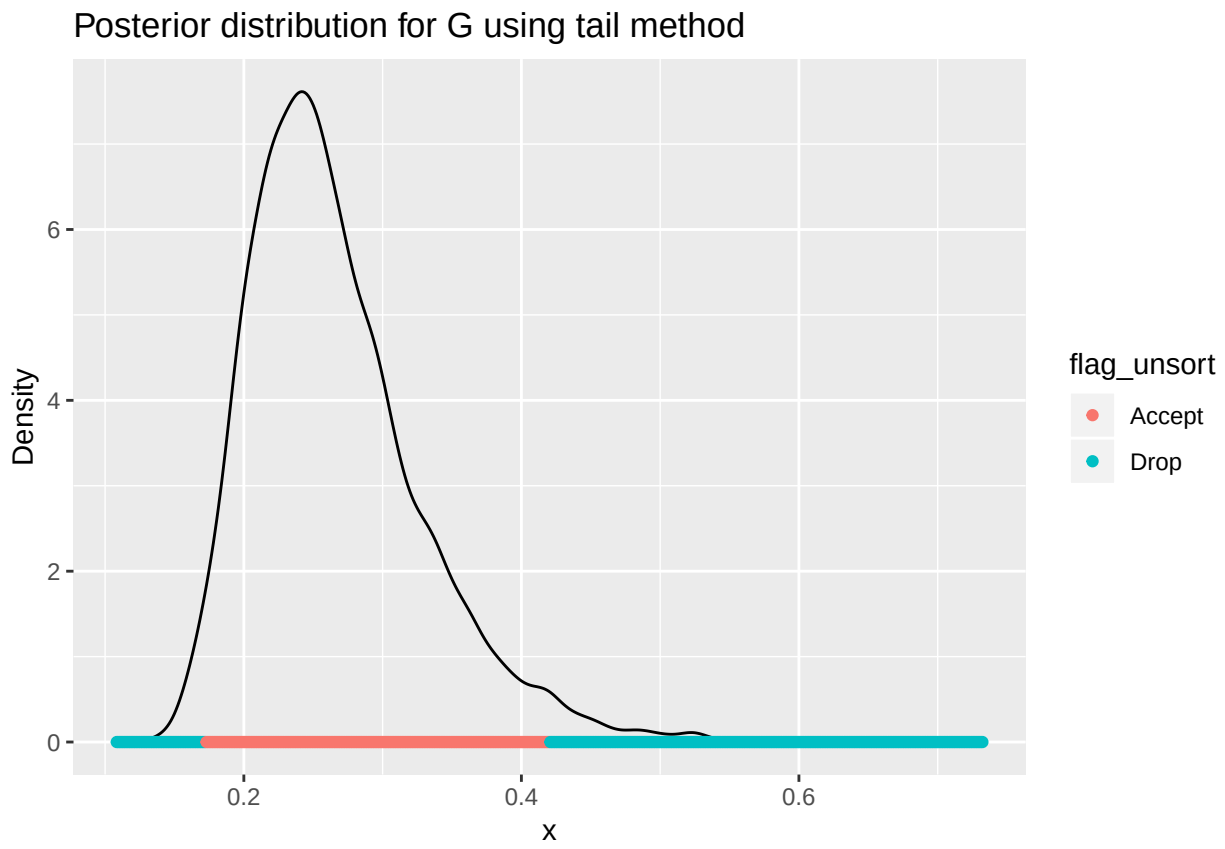
In *2a)*, we simulated the posterior distribution for σ^2 using the current data set. These obtained values for σ^2 now can be used within $G = 2\phi(\sigma/\sqrt{2}) - 1$ to compute the posterior distribution of the Gini coefficient G. Since $\phi(z)$ refers to the CDF of the standard normal distribution, we can make use of the `pnorm(q, mean = 0, sd = 1)`-function where `q` equals to the different values of $\sigma/\sqrt{2}$. The computed distribution will be plotted.

```
# Computing g values.
g = 2 * pnorm(q = sqrt(sigma_sq)/sqrt(2), mean = 0, sd = 1) - 1
# Plotting distribution.
ggplot() +
  geom_density(aes(x = g)) +
  ggtitle("Posterior distribution for G")
```

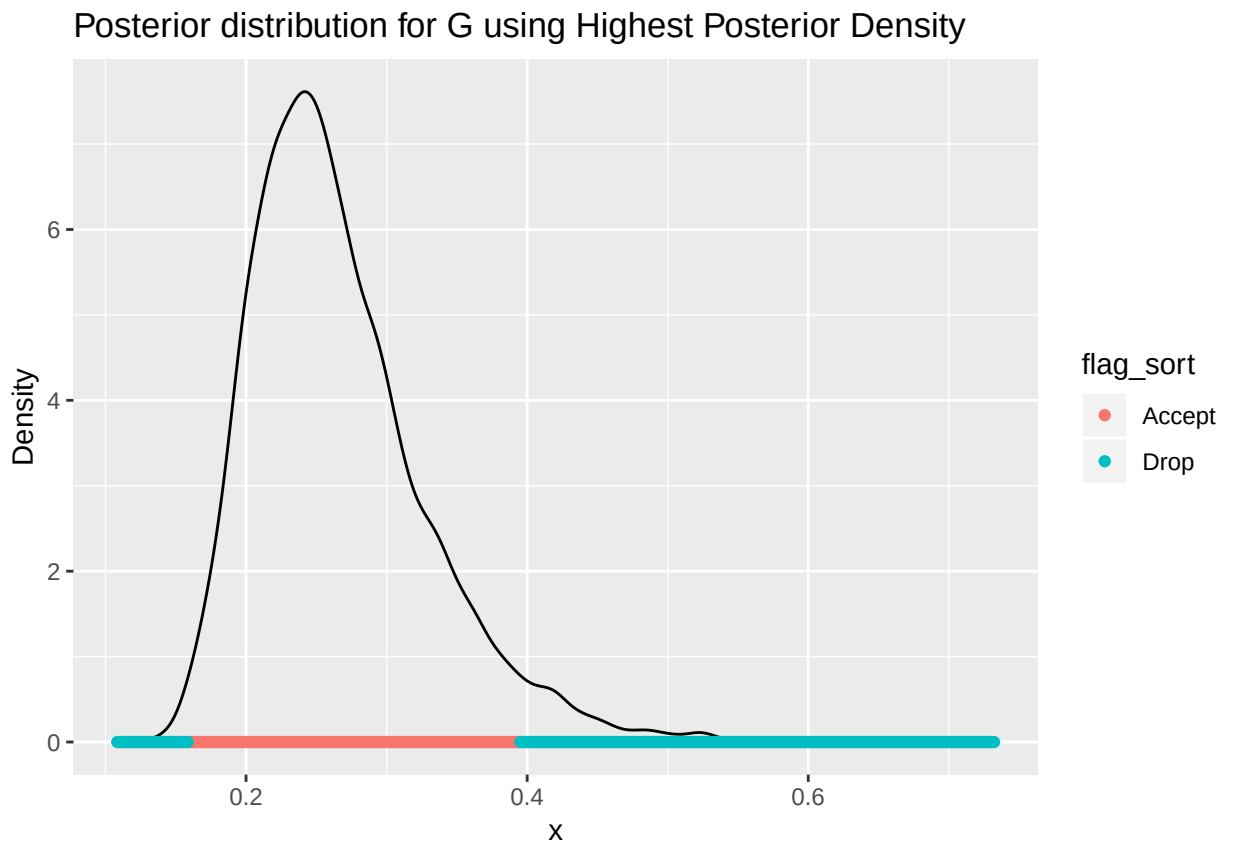


c)Use the posterior draws from b) to compute a 95% equal tail credible interval for G. An 95% equal tail interval (a,b) cuts off 2.5% percent of the posterior probability mass to the left of a, and 97.5% to the right of b. Also, do a kernel density estimate of the posterior of G using the density function in R with default settings, and use that kernel density estimate to compute a 95% Highest Posterior Density interval for G. Compare the two intervals.

```
temp <- density(g)
values_df <- data.frame(x = temp$x, Density = temp$y)
total <- sum(values_df$Density)

values_df_unsorted <- values_df %>%
  arrange(x) %>%
  mutate(running_per_unsort = 100 * cumsum(Density)/total) %>%
  mutate(flag_unsort = ifelse(running_per_unsort < 2.50, "Drop",
                       (ifelse(running_per_unsort < 97.50, "Accept", "Drop"))))

values_df_sorted <- values_df %>%
  arrange(desc(Density)) %>%
  mutate(running_per_sort = 100 * cumsum(Density)/total) %>%
  mutate(flag_sort = ifelse(running_per_sort < 95.00, "Accept", "Drop"))

ggplot(data=values_df_unsorted, aes(x=x, y=Density)) +
  geom_line() +
  geom_point(aes(x=x,y=0,color=flag_unsort)) +
  ggtitle("Posterior distribution for G using tail method")
```




```
ggplot(data=values_df_sorted, aes(x=x, y=Density)) +
  geom_line() +
  geom_point(aes(x=x,y=0,color=flag_sort)) +
  ggtitle("Posterior distribution for G using Highest Posterior Density")
```



Analysis: We see that Highest Posterior Density works better and covers the points where the density is higher, while the usuall symmetric tail method clips even high density regions.

Question 3: Bayesian Inference

Bayesian inference for the concentration parameter in the von Mises distribution. This exercise is concerned with directional data. The point is to show you that the posterior distribution for somewhat weird models can be obtained by plotting it over a grid of values. The data points are observed wind directions at a given location on ten different days. The data are recorded in degrees and converted into radians: (2.44, 2.14, 2.54, 1.83, 2.02, 2.33, 2.79, 2.23, 2.07, 2.02)

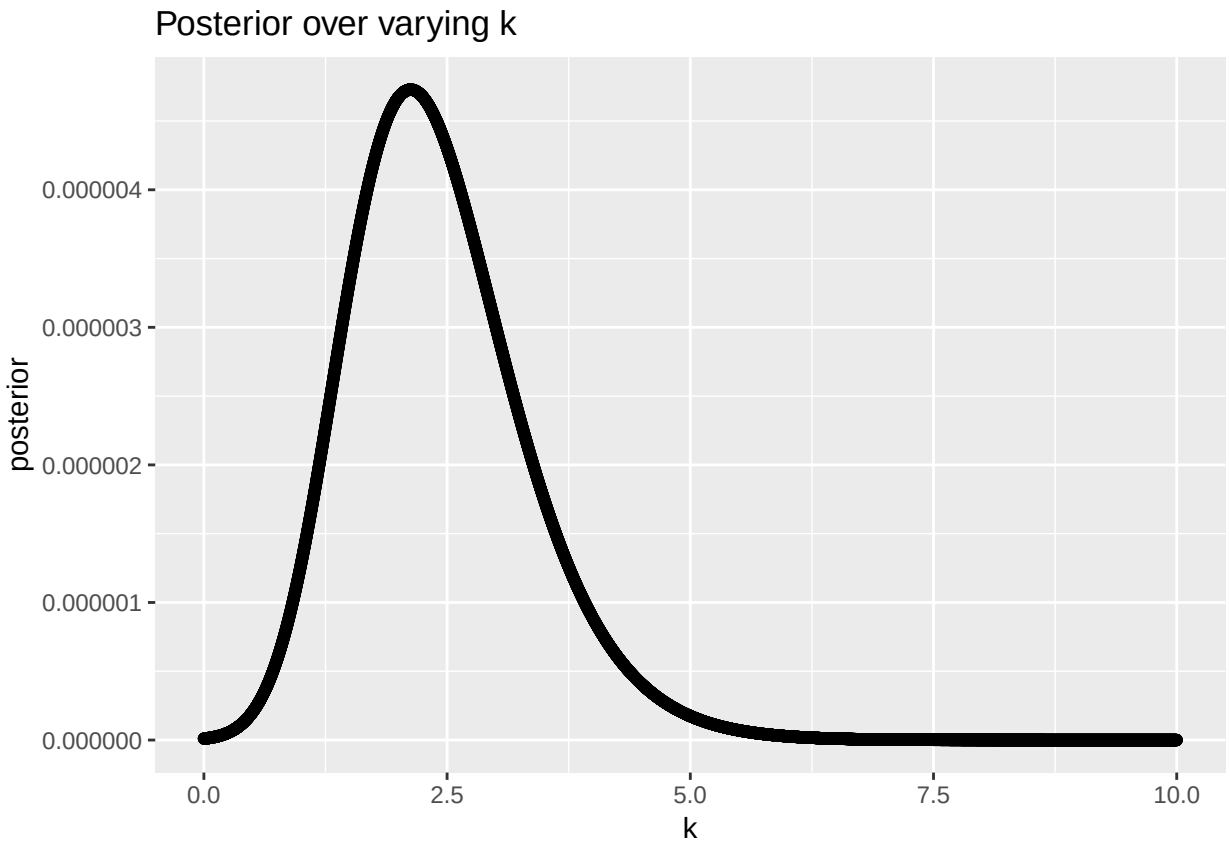
Assume that these data points are independent observations following the von Mises distribution:

$$p(y|\mu,\kappa) = \frac{e[\kappa \cdot cos(y-\mu)]}{2\pi I_0(\kappa)}, \quad -\pi \leq y \leq \pi$$

where I0(k) is the modified Bessel function of the first kind of order zero [see ?besselI in R]. The parameter μ ($-\pi \leq y \leq \pi$) is the mean direction and k > 0 is called the concentration parameter. Large k gives a small variance around μ, and vice versa. Assume that μ is known to be 2:39. Let k ~ Exponential($\lambda = 1$) a priori, where λ is the rate parameter of the exponential distribution (so that the mean is $1/\lambda$).

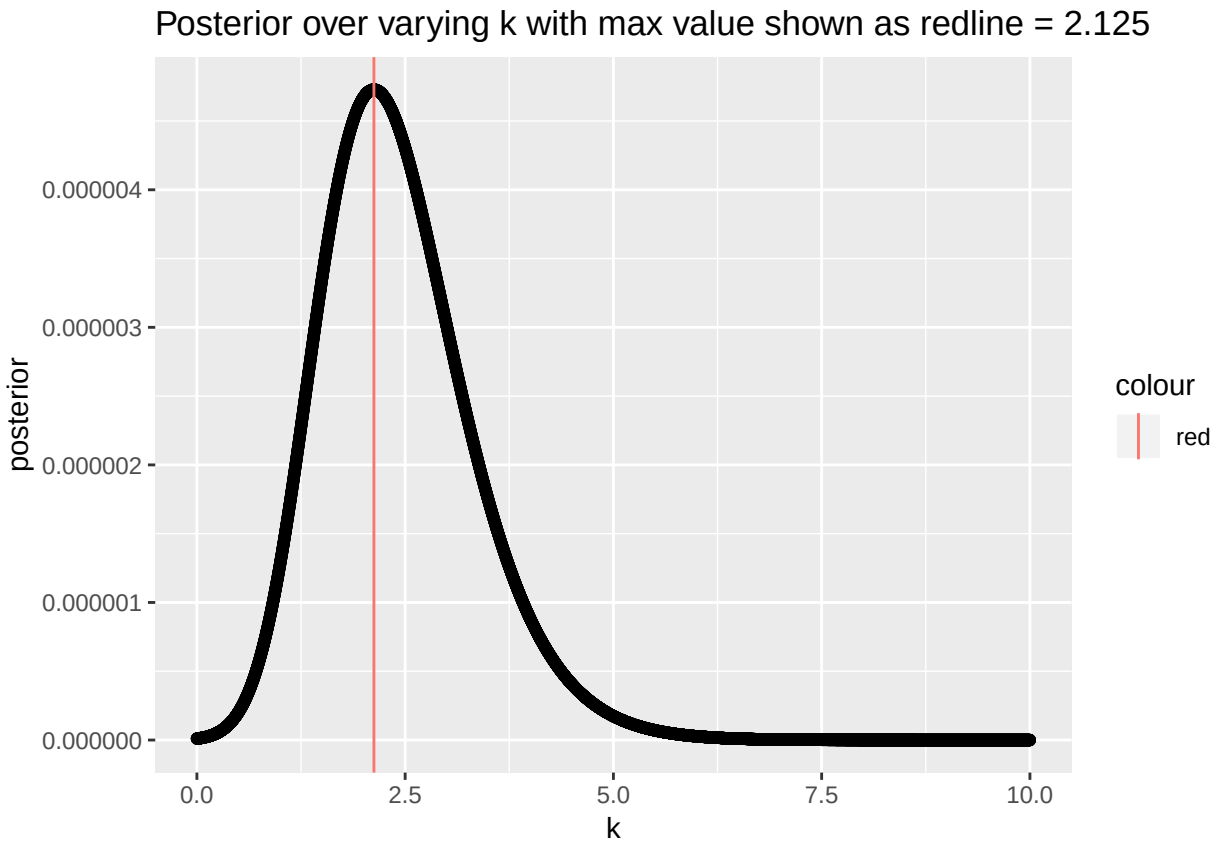
a)Plot the posterior distribution of κ for the wind direction data over a fine grid of κ values.

```
y <- c(-2.44, 2.14, 2.54, 1.83, 2.02, 2.33, -2.79, 2.23, 2.07, 2.02)
mean_y <- 2.39
k <- seq(0,10,0.001)

calc_prob = function(k, y){
  prob = exp(k * cos(y - mean_y)) / (2*pi*besselI(k, 0))
  return (prob)
}

calc_post = function(k){
  probabilities = sapply(y, calc_prob, k=k)
  prior = dexp(k)
  posterior = prod(probabilities) * prior
  return (posterior)
}

posterior = sapply(k, calc_post)

result_df <- data.frame(k = k, posterior = posterior)

ggplot(data=result_df, aes(x=k, y=posterior)) +
  geom_point() +
  ggtitle("Posterior over varying k")
```



b)Find the (approximate) posterior mode of κ from the information in a).

```
# Find the k value which maximizes posterior
mode = result_df$k[which.max(result_df$posterior)]

ggplot(data=result_df, aes(x=k, y=posterior)) +
  geom_point() +
  geom_vline(aes(xintercept = mode, color = "red")) +
  ggtitle(paste0("Posterior over varying k with max value shown as redline = ",mode))
```



Analysis: The max value of posterior is termed as the mode of kappa

Appendix

```
knitr::opts_chunk$set(echo = TRUE)
options(scipen=999)
library("ggplot2")
library("dplyr")
library("LaplacesDemon")

# The palette with black:
cbbPalette <- c("#000000", "#E69F00", "#56B4E9", "#009E73",
                "#F0E442", "#0072B2", "#D55E00", "#CC79A7")

#' Title A beta distribution generator to generate N values
#'
#' @param N Number of iterations
#' @param alpha Alpha value for beta distribution
#' @param beta  Beta value for beta distribution
#'
#' @return a list containing the mean and std dev of the generated vector
#' @export
#'
#' @examples
```

```r
gen_beta_sample <- function(N, alpha, beta){
  temp <- rbeta(N, shape1=alpha, shape2=beta)
  mean_value <- mean(temp)
  std_dev <- sd(temp)
  answer <- list(mean_value=mean_value, std_dev=std_dev)
  return(answer)
}

# required input
N = 3000
alpha = 16
beta = 8

# Running the loop inside a function to get the values for varying N
final <- NULL
for(i in 2:N){
  temp <- gen_beta_sample(N=i, alpha = alpha, beta=beta)
  df <- cbind(N = i, Mean = temp$mean_value, stand_dev = temp$std_dev)
  final <- rbind(df, final)
}

# Making final as a dataframe to make it easy to plot
final <- final %>% data.frame()
final$true_mean <- ((alpha)/(alpha+beta))
final$true_std_dev <- sqrt((alpha*beta)/((alpha+beta)^2 *(alpha+beta+1)))

#plots
ggplot(data=final, aes(x=N, y=Mean)) +
  geom_point(color= "#E69F00") +
  geom_line(aes(y=true_mean), color = "#000000") +
  ggtitle("Predicted(line) and Actual Mean(dots) vs. Iterations")


ggplot(data=final, aes(x=N, y=stand_dev)) +
  geom_point(color= "#E69F00") +
  geom_line(aes(y=true_std_dev), color = "#000000") +
  ggtitle("Predicted(line) and Actual Standard Deviation(dots) vs. Iterations")




#' Title A beta distribution generator to generate N values without mean
#'
#' @param N Number of iterations
#' @param alpha Alpha value for beta distribution
#' @param beta  Beta value for beta distribution
#'
#' @return a list containing the mean and std dev of the generated vector
#' @export
#'
#' @examples
gen_beta_sample_2 <- function(N, alpha, beta){
  answer <- rbeta(N, shape1=alpha, shape2=beta)
```

```r
  return(answer)
}

# required input
N = 10000

# Running the loop inside a function to get the values for varying N
beta_values <- gen_beta_sample_2(N=N, alpha = alpha, beta = beta)
actual_prop <- sum(beta_values < 0.4)/N
true_prop <- pbeta(q=0.4, shape1=alpha, shape2=beta, lower.tail = TRUE)

cat("The true value of the probability is: ", true_prop, ",
    while the calculated value is: ", actual_prop)

log_odds = log(beta_values/(1-beta_values))
log_odds <- as.data.frame(log_odds)

ggplot(data=log_odds, aes(x=log_odds)) +
  geom_histogram(bins = 30) +
  ggtitle("Histogram of log-odds")

y = c(14, 25, 45, 25, 30, 33, 19, 50, 34, 67)
n = length(y)
mu = 3.5
tau_sq = sum((log(y)-mu)^2)/n

sigma_sq = c()
for (i in 1:10000) {
  # Drawing x.
  x = rchisq(n = 1, df = n)
  # Calculating and storing sigma_sq.
  sigma_sq[i] = (n*tau_sq)/x
}
# Plotting simulated posterior distribution.
ggplot() +
  geom_density(aes(x = sigma_sq)) +
  labs(title = "Posterior distribution for sigma_sq") +
  theme_bw()
# Printing statistics of simulated distribution compared to theoretical values.
knitr::kable(
  as.data.frame(
    rbind(
      cbind(Posterior = "Simulation", Mean = mean(sigma_sq), Sd = sd(sigma_sq)),
      cbind(Posterior = "Theory", Mean = n*tau_sq/(n-2), Sd = sqrt((2*n^2*tau_sq^2)/(((n-2)^2)*(n-4))))
    )
  )
)
# Computing g values.
g = 2 * pnorm(q = sqrt(sigma_sq)/sqrt(2), mean = 0, sd = 1) - 1
# Plotting distribution.
ggplot() +
  geom_density(aes(x = g)) +
  ggtitle("Posterior distribution for G")
```

```r
temp <- density(g)
values_df <- data.frame(x = temp$x, Density = temp$y)
total <- sum(values_df$Density)

values_df_unsorted <- values_df %>%
  arrange(x) %>%
  mutate(running_per_unsort = 100 * cumsum(Density)/total) %>%
  mutate(flag_unsort = ifelse(running_per_unsort < 2.50, "Drop",
                       (ifelse(running_per_unsort < 97.50, "Accept", "Drop"))))

values_df_sorted <- values_df %>%
  arrange(desc(Density)) %>%
  mutate(running_per_sort = 100 * cumsum(Density)/total) %>%
  mutate(flag_sort = ifelse(running_per_sort < 95.00, "Accept", "Drop"))

ggplot(data=values_df_unsorted, aes(x=x, y=Density)) +
  geom_line() +
  geom_point(aes(x=x,y=0,color=flag_unsort)) +
  ggtitle("Posterior distribution for G using tail method")

ggplot(data=values_df_sorted, aes(x=x, y=Density)) +
  geom_line() +
  geom_point(aes(x=x,y=0,color=flag_sort)) +
  ggtitle("Posterior distribution for G using Highest Posterior Density")

y <- c(-2.44, 2.14, 2.54, 1.83, 2.02, 2.33, -2.79, 2.23, 2.07, 2.02)
mean_y <- 2.39
k <- seq(0,10,0.001)

calc_prob = function(k, y){
  prob = exp(k * cos(y - mean_y)) / (2*pi*besselI(k, 0))
  return (prob)
}

calc_post = function(k){
  probabilities = sapply(y, calc_prob, k=k)
  prior = dexp(k)
  posterior = prod(probabilities) * prior
  return (posterior)
}

posterior = sapply(k, calc_post)

result_df <- data.frame(k = k, posterior = posterior)

ggplot(data=result_df, aes(x=k, y=posterior)) +
  geom_point() +
  ggtitle("Posterior over varying k")


# Find the k value which maximizes posterior
mode = result_df$k[which.max(result_df$posterior)]
```

```r
ggplot(data=result_df, aes(x=k, y=posterior)) +
  geom_point() +
  geom_vline(aes(xintercept = mode, color = "red")) +
  ggtitle(paste0("Posterior over varying k with max value shown as redline = ",mode))
```