# Bayesian Learning lab1

Jaskirat S Marar (jasma356), Dinuke Jayaweera (dinja628)

4/11/2022

## Question 1 : Daniel Bernoulli

Let $y_1, ..., y_n|\theta \sim \text{Bern}(\theta)$, and assume that you have obtained a sample with s = 13 successes in n = 50 trials. Assume a Beta($\alpha_0$, $\beta_0$) prior for $\theta$ and let $\alpha_0 = \beta_0 = 5$.

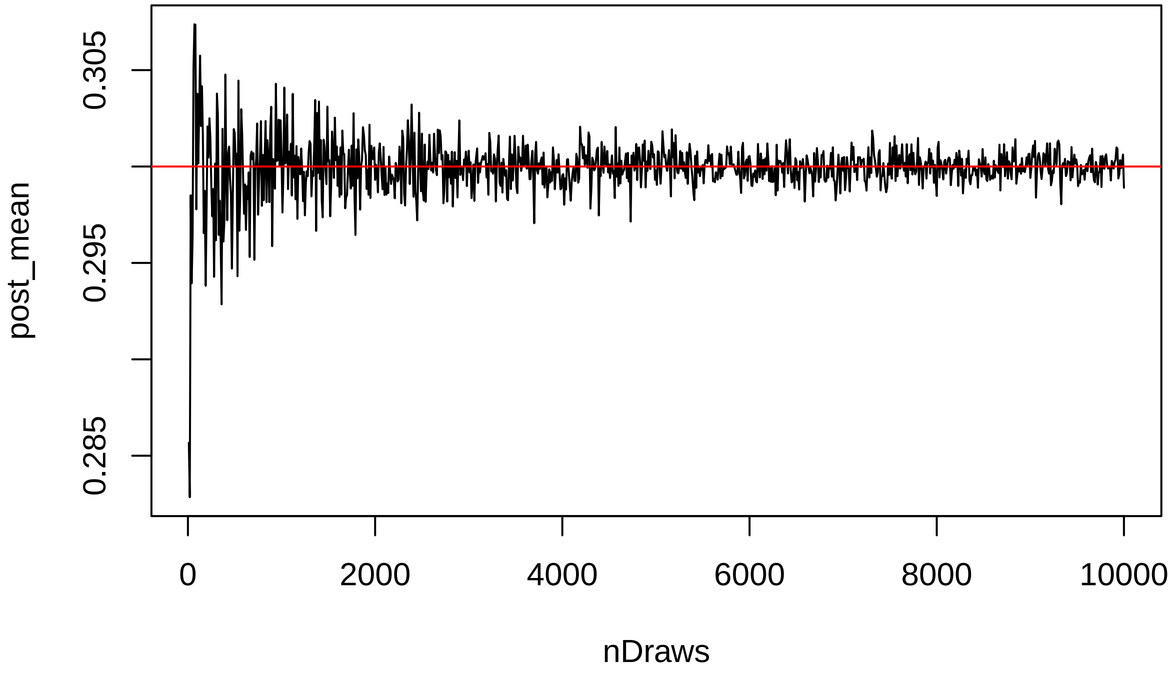
```
#QUESTION 1

rm(list = ls())

#Alpha and Beta values for prior
a0 = 5
b0 = 5

#sample data
n = 50
s = 13

#Alpha and beta values for posterior
a1 <- a0 + s
b1 <- b0 + n - s
```

a. Draw random numbers from the posterior $\theta|\text{y} \sim \text{Beta}(\alpha_0 + s, \beta_0 + f)$, where y = $(y_1, ..., y_n)$, and verify graphically that the posterior mean E $[\theta|\text{y}]$ and standard deviation SD$[\theta|\text{y}]$ converges to the true values as the number of random draws grows large. [Hint: use rbeta()].

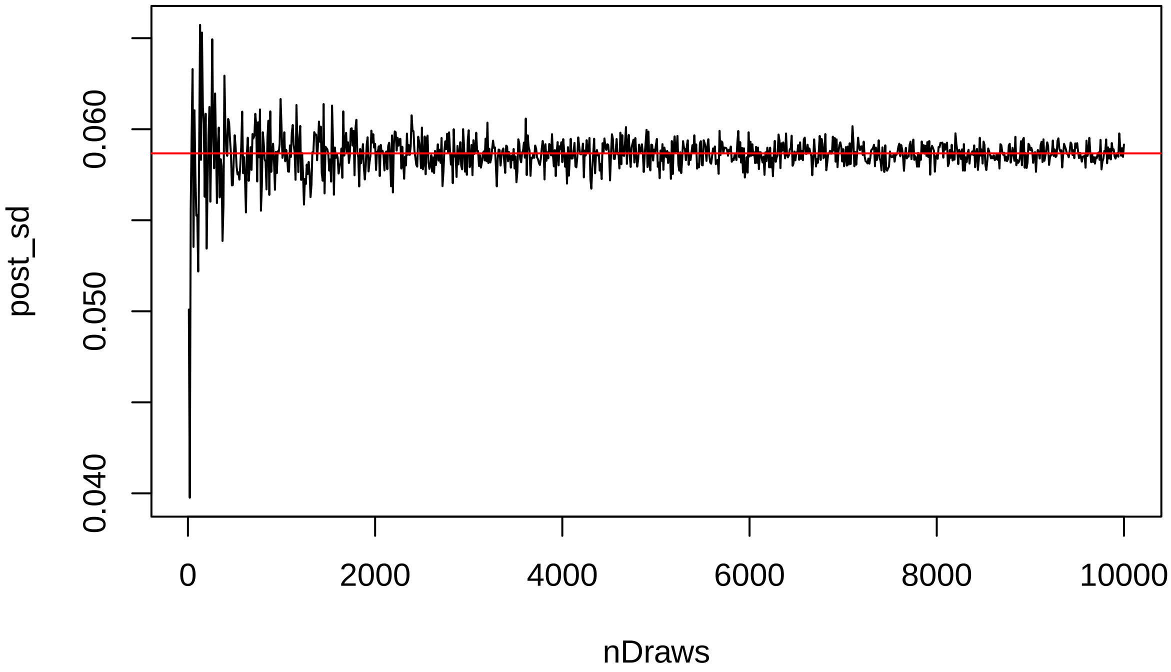
For solving this problem, we first stored the true mean and sd by using the known expressions for any given beta distribution. To visualize the mean and sd for different values of different sample draws, we loop over a vector of sample draw sizes and store each mean and sd and plot them later to show convergence to true mean and sd.

```
#PART A

#true mean and sd
true_mean <- a1/(a1+b1)
true_sd <- sqrt(a1*b1/((a1+b1)^2 * (a1+b1+1)))

#initialize vectors for posterior draws, mean and sd
post_mean <- c()
post_sd <- c()
posterior <- c()

#initialize vector for simulating number of draws
nDraws <- seq(10, 10000, by = 10)
```

```
#loop to draw random numbers and calculate each mean and sd
for(i in 1:length(nDraws)){
  posterior <- rbeta(nDraws[i], a1, b1)
  post_mean[i] <- mean(posterior)
  post_sd[i] <- sd(posterior)
}

#plot the means
plot(x = nDraws, y = post_mean, type = "l")
abline(h = true_mean, col = "red")
```



```
#plot the sd
plot(x = nDraws, y = post_sd, type = "l")
abline(h = true_sd, col = "red")
```



b. Use simulation (nDraws = 10000) to compute the posterior probability $\Pr(\theta < 0.3|y)$ and compare with the exact value from the Beta posterior. [Hint: use pbeta()].

Here we first calculate the exact value of the posterior using the pbeta() and giving the input cutoff at <0.3

To find the posterior probability of theta < 0.3, we simulate a draw of 10000 samples and find the count of samples that are < 0.3 and divide by the total number of draws i.e. 10000

```
#PART B

exact_posterior <- pbeta(0.3, a1, b1)

compute_posterior <- rbeta(10000, a1, b1)

compute_est_posterior <- length(compute_posterior[which(compute_posterior < 0.3)])/10000
```

The comparison of estimated and exact values of posterior probability is as follows:

```
cat("\n", "Estimated Posterior Probability: ", round(compute_est_posterior,4))
```

```
##
##  Estimated Posterior Probability:  0.5186
```

```
cat("\n", "Exact Posterior Probability: ", round(exact_posterior,4))
```

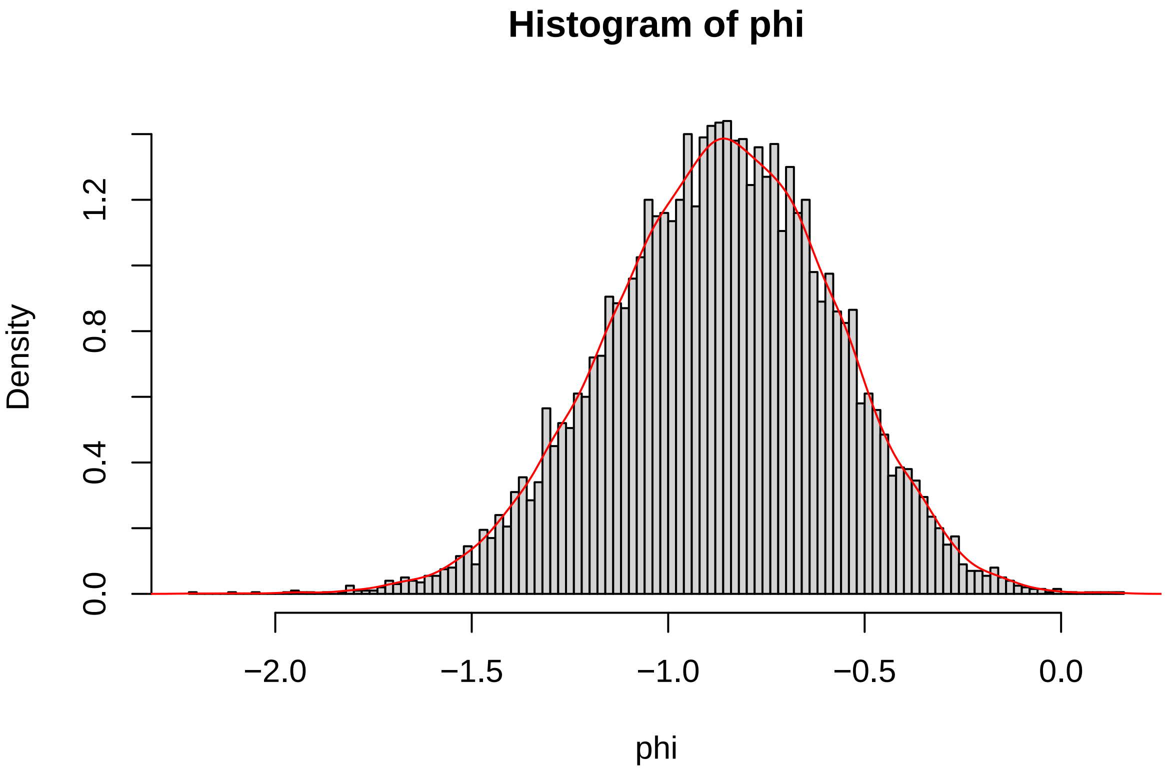
```
##
##  Exact Posterior Probability:  0.515
```

c. Simulate draws from the posterior distribution of the log-odds $\phi = log\frac{\phi}{1/\phi}$ by using simulated draws from the Beta posterior for $\theta$ and plot the posterior distribution of $\phi$ (nDraws = 10000). [Hint: hist()

and density() can be utilized].

Here we compute log-odds $\phi$ using the given expression by using the draws from the previous problem. Plot & histogram are shown below.

```
phi <- log(compute_posterior/(1 - compute_posterior))

hist(phi, probability = TRUE, breaks = 100)

lines(density(phi), col="red", ylim = 1.5)
```



## Question 2: Log-normal Distribution and Gini Coefficient

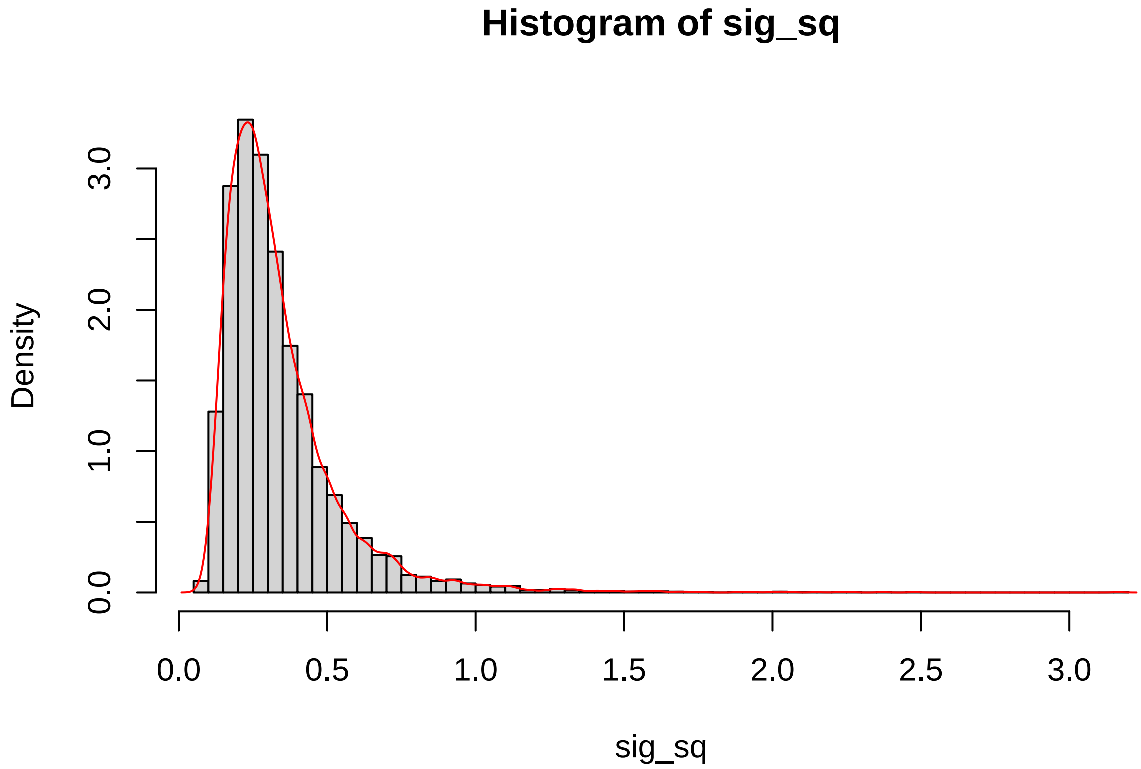
a. Simulate 10000 draws from the posterior of $\sigma^2$ by assuming $\mu = 3.5$ and plot the posterior distribution.

Since the posterior is an $Inv - \chi^2$ distribution, we need to first draw samples from a $\chi^2$ distribution with 'n' degrees of freedom i.e. the sample size. The resulting samples are then used to compute values of $\sigma^2$ as er the following expression:

$$\sigma^2 = \frac{n \cdot \tau^2}{X} \text{ ; } where\ X \sim \chi^2(n, \tau^2)$$

The computed samples will be $Inv - \chi^2(n, \tau^2)$ distributed. The resulting samples are plotted below:

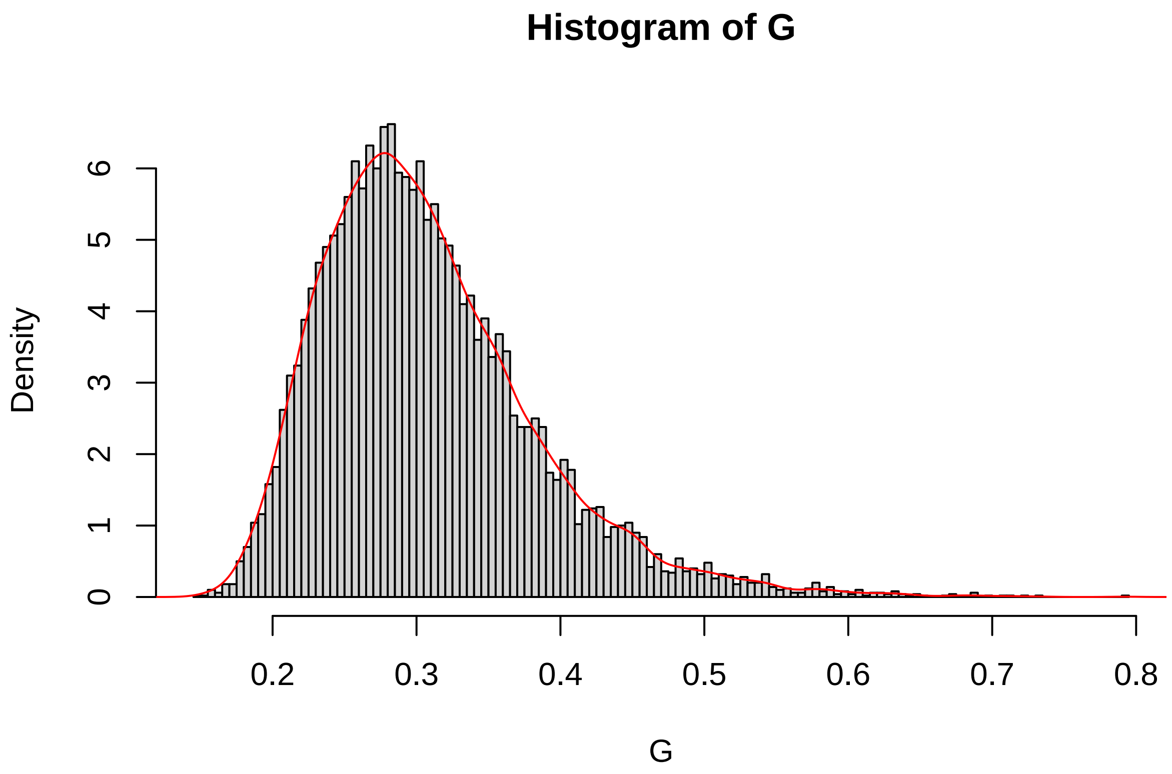
```
#QUESTION 2

rm(list = ls())

y_i <- c(33, 24, 48, 32, 55, 74, 23, 76, 17)
mu = 3.5

#PART A
nDraws = 10000
tau_sq <- sum((log(y_i) - mu)^2)/length(y_i) #given expression for tau^2
X <- rchisq(nDraws, length(y_i)) #draw x from chi-sq with n df
sig_sq <- length(y_i) * tau_sq/X #compute for inv-chi-sq draws

hist(sig_sq, probability = TRUE, breaks = 100)
lines(density(sig_sq), col = "red", ylim = 10)
```



b. Use the posterior draws in a) to compute the posterior distribution of the Gini coefficient G for the current data set.

We use the expression for calculating the Gini index. $\phi$ is calculated as the standard normal CDF of the $\sigma^2$ values drawn in the previous example by using the pnorm(). The resulting histogram and density curve is shown below:

```
#PART B

#CDF of std normal with mean = 0, sd = 1, data = sigma^2
phi <- pnorm(sqrt(sig_sq/2))

#Gini Coefficient using given expression
G <- 2*phi - 1

hist(G, breaks = 100, probability = TRUE)
lines(density(G), col = "red")
```

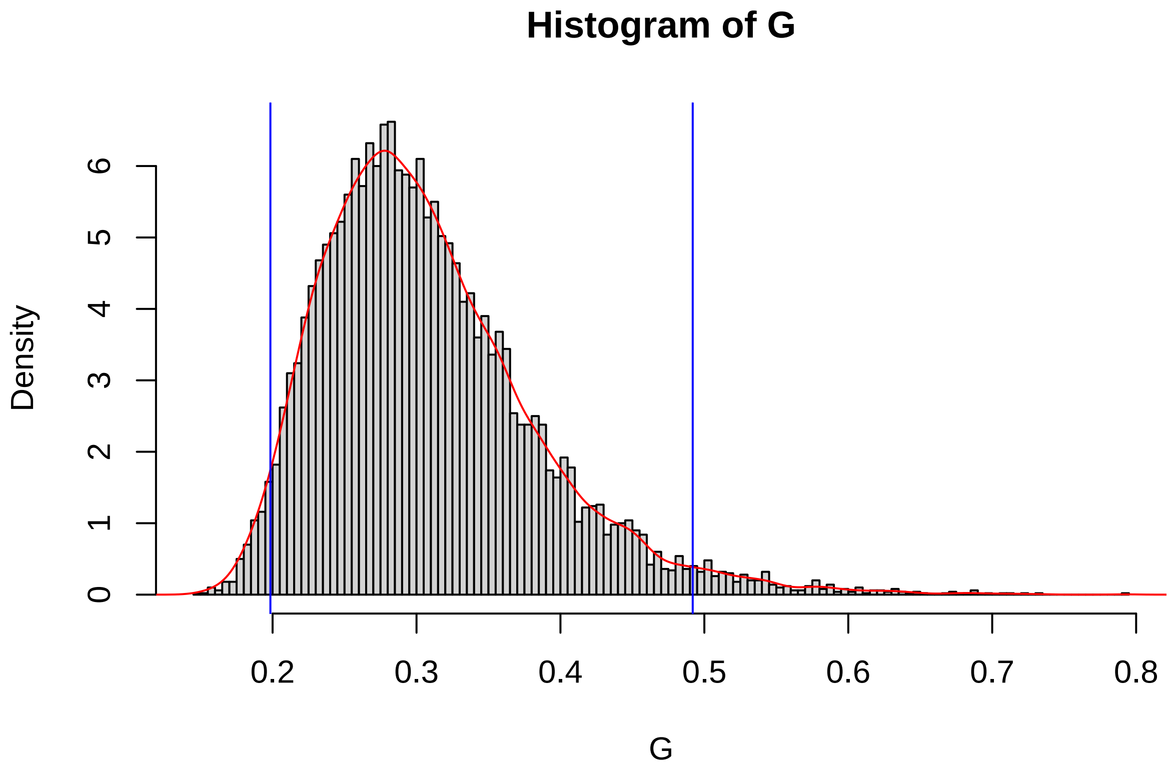


c. Use the posterior draws from b) to compute a 95% equal tail credible interval for G.

To compute the equal tail CI, we employ the quantile() over the posterior draws to get both the lower and upper CI and overlay on the earlier plot.

```
#PART C

#ETI cutoffs for 2.5% and 97.5% cutoffs
margin <- quantile(G, probs = c(0.025, 0.975))

hist(G, breaks = 100, probability = TRUE)
```

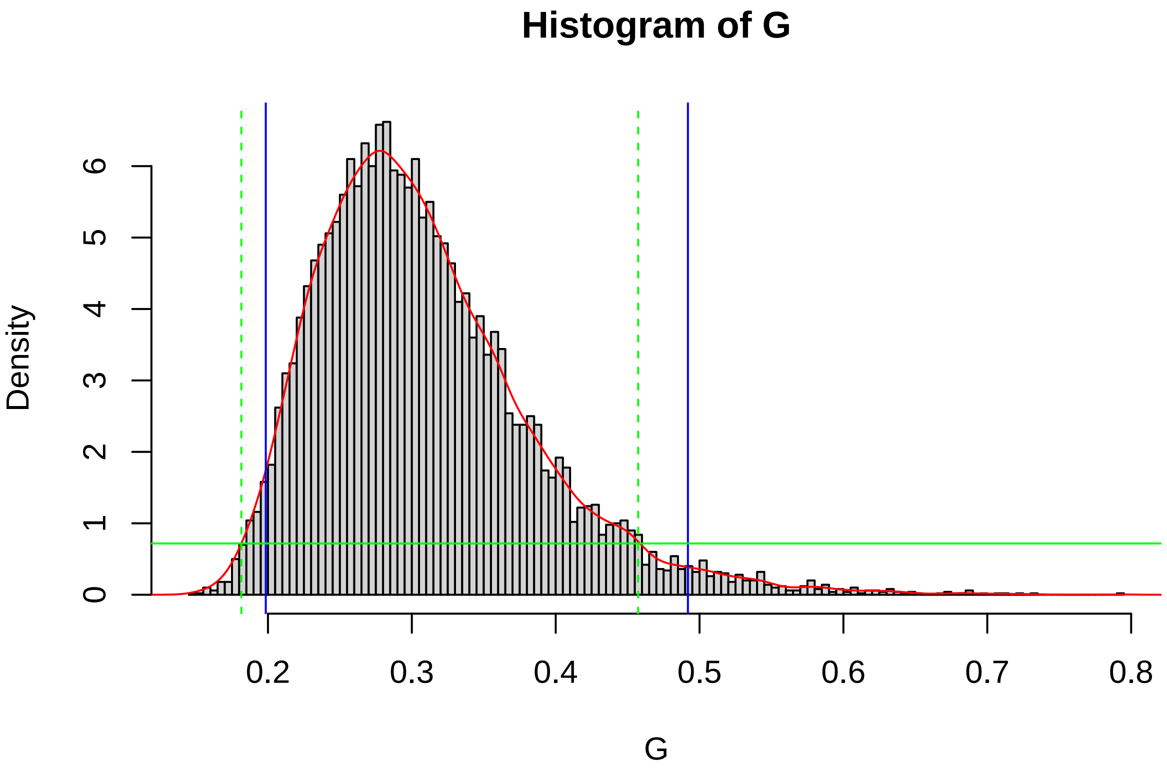
```
lines(density(G), col = "red")
abline(v = margin, col = "blue")
```



d. Use the posterior draws from b) to compute a 95% Highest Posterior Density Interval (HPDI) for G. Compare the two intervals in (c) and (d).

In order to compute the HPDI for G, we need to first find the density of the samples using density(). We then order the density data and find the cut-off density where the cumulative density >95% of the the total density. Since, the data is ordered and we have seen that the data just has one peak, the cut off will only intersect the curve at 2 points. Once we isolate the density points which are outside the 95% HDPI, we can match the cut off density in the isolated samples to find the HPDI cut offs.

```
#PART D
G_density <- density(G) #convert G into density
G_df <- data.frame(x = G_density$x, y = G_density$y) #create df of density
G_df <- G_df[order(G_df$y, decreasing = TRUE),] #order df to find cutoff
G_df$cum_y <- cumsum(G_df$y) #add column of cum density in ordered df
G_df_hdi <- G_df[which(G_df$cum_y < 0.95*sum(G_df$y)),] #subset density samples outside HPDI
hdi_low <- min(G_df_hdi$x) #lower HPDI cutoff
hdi_max <- max(G_df_hdi$x) #upper HPDI cutoff

hist(G, breaks = 100, probability = TRUE)
lines(density(G), col = "red")
abline(v = margin, col = "blue")
abline(h = G_df_hdi[G_df_hdi$x == hdi_low,2], col = "green")
abline(v = hdi_low, col = "green", lty  = 2)
abline(v = hdi_max, col = "green", lty  = 2)
```



*Comparison:*

In c) we are computing the equal tail CI which essentially finds the cut offs where 2.5% of data is excluded from both the tails equally. Whereas in d) the HPDI corresponds to the interval where are choosing the cut offs with the highest density. Hence, the chosen samples are more credible than if we employ ETI. For example, in the lower tail, the samples that are excluded from the ETI, actually have a higher density than the samples that were 'compensated' with in the upper tail. Hence, HPDI can be used as a better decision criteria than ETI.

## Question 3: Bayesian inference for the concentration parameter in the von Mises distribution

a. Derive the expression for what the posterior p($\kappa|y,\mu$) is proportional to. Hence, derive the function f ($\kappa$) such that $p(\kappa|y,\mu) \propto f(\kappa)$. Then, plot the posterior distribution of $\kappa$ for the wind direction data over a fine grid of $\kappa$ values.

$$P(\kappa|y,\mu) \propto P(y|\kappa,\mu) * P(\kappa|\mu)$$

Likelihood - Von-Mises distribution

$$P(y|\kappa,\mu) = \prod_{i=1}^{n} \frac{1}{2\pi I_o(\kappa)} exp[\kappa \cdot cos(y_i - \mu)]$$

$$= \Big(\frac{1}{2\pi I_o(\kappa)}\Big)^n exp[\kappa \cdot \sum_{i=1}^{n} cos(y_i - \mu)]$$

$$= \frac{1}{(2\pi)^n I_o(\kappa)^n} exp[\kappa \cdot \sum_{i=1}^{n} cos(y_i - \mu)]$$

Remove the constant as it does not depend on $\kappa$: $(2\pi)^n$.

$$P(y|\kappa,\mu) \propto I_o(\kappa)^{-n} exp[\kappa \cdot \sum_{i=1}^{n} cos(y_i - \mu)]$$

Prior - Exponential distribution

$$P(\kappa) = \lambda \cdot exp[-\lambda\kappa],\ where\ \lambda = 1 = exp[-\kappa]$$

$$Posterior \propto Likelihood \cdot\ Prior$$

$$\propto \frac{1}{I_o(\kappa)^n} exp[\kappa \cdot \sum_{i=1}^{n} cos(y_i - \mu) - \kappa]$$

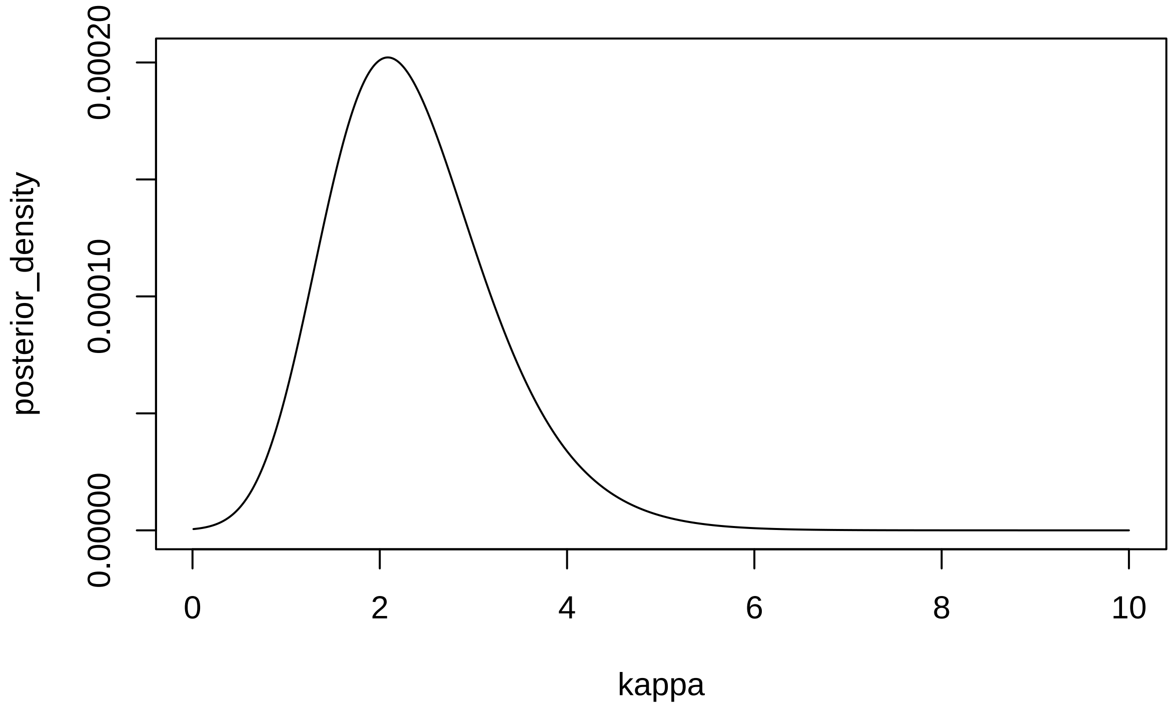
$$\propto \frac{1}{I_o(\kappa)^n} exp[\kappa \cdot (\sum_{i=1}^{n} cos(y_i - \mu) - 1)]$$

Normalizing Constant:

The integral for the normalizing constant is very complex to find, so we will employ a trick that approximates the area under the curve of the posterior pdf obtained above and divide the above expression by it to normalize it. The way we do this is by using the fine grid of $\kappa$ values separated infinitesimally and taking the sum of areas of rectangles with height = the y-value on the curve for each kappa grid value. This would give us a fair approximation of the area under the curve.

In order to best estimate the area under the curve, we choose a very fine grid of $\kappa$ values. We create a separate function for the posterior using our derivation above

```
#QUESTION 3

rm(list = ls())

y <- c(1.83, 2.02, 2.33, -2.79, 2.07, 2.02, -2.44, 2.14, 2.54, 2.23)
mu = 2.51

kappa <- seq(0.01,10, by = 0.001)

posterior <- function(k, y, mu){
  post_pdf <- exp(k * (sum(cos(y - mu)) - 1)) / (besselI(k, nu = 0)^length(y))
  return(post_pdf)
}

#draws from the posterior
posterior_test <- posterior(kappa, y, mu)

#normalize the posterior
posterior_density <- posterior_test/sum(kappa*posterior_test)

plot(kappa, posterior_density, type = 'l')
```



b. Find the (approximate) posterior mode of $\kappa$ from the information in a)

```
#find the kappa value of the posterior mode
k_mode = kappa[which.max(posterior_density)]
```

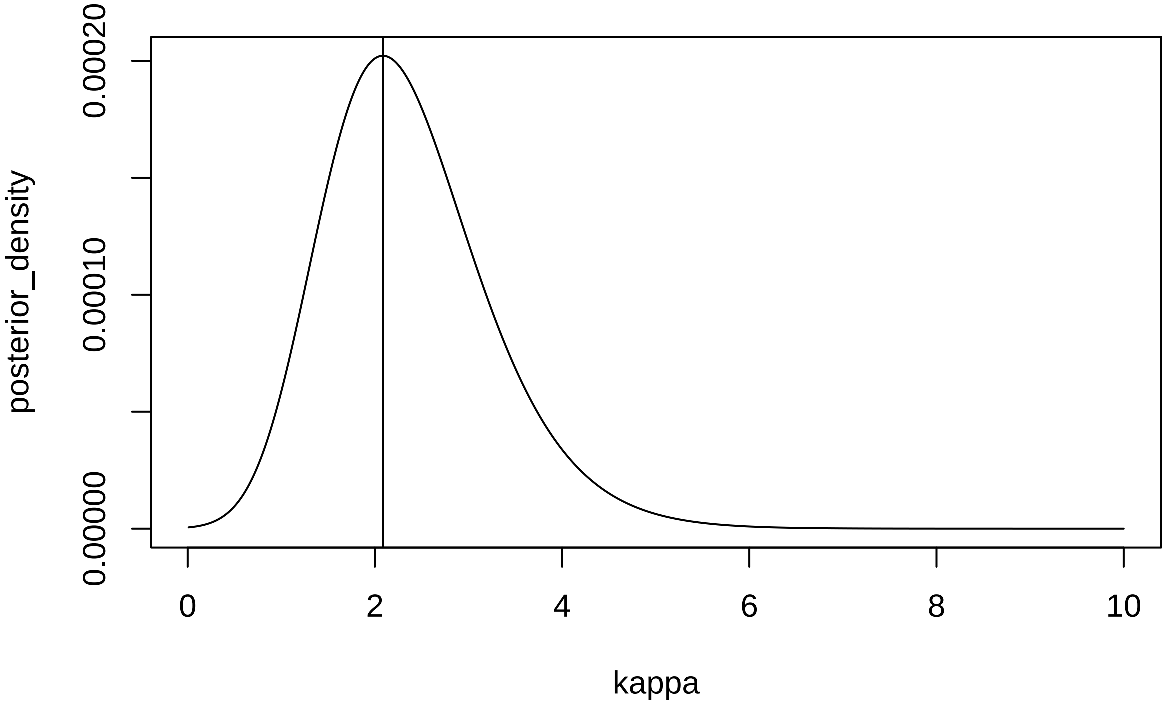
```
#find the posterior mode
posterior_k_mode <- posterior_density[which.max(posterior_density)]

cat("Approximate Posterior Mode of kappa: ", round(posterior_k_mode,6), "\n")
```

```
## Approximate Posterior Mode of kappa:  0.000202
```

```
cat("Kappa value for posterior mode:", round(k_mode,3), "\n")
```

```
## Kappa value for posterior mode: 2.086
```

```
plot(kappa, posterior_density, type = 'l')
abline(v = k_mode)
```



#Appendix

```r
knitr::opts_chunk$set(echo = TRUE)
#QUESTION 1

rm(list = ls())

#Alpha and Beta values for prior
a0 = 5
b0 = 5

#sample data
n = 50
s = 13

#Alpha and beta values for posterior
a1 <- a0 + s
b1 <- b0 + n - s


#PART A

#true mean and sd
true_mean <- a1/(a1+b1)
true_sd <- sqrt(a1*b1/((a1+b1)^2 * (a1+b1+1)))

#initialize vectors for posterior draws, mean and sd
post_mean <- c()
post_sd <- c()
posterior <- c()

#initialize vector for simulating number of draws
nDraws <- seq(10, 10000, by = 10)

#loop to draw random numbers and calculate each mean and sd
for(i in 1:length(nDraws)){
  posterior <- rbeta(nDraws[i], a1, b1)
  post_mean[i] <- mean(posterior)
  post_sd[i] <- sd(posterior)
}

#plot the means
plot(x = nDraws, y = post_mean, type = "l")
abline(h = true_mean, col = "red")

#plot the sd
plot(x = nDraws, y = post_sd, type = "l")
abline(h = true_sd, col = "red")


#PART B

exact_posterior <- pbeta(0.3, a1, b1)
```

```r
compute_posterior <- rbeta(10000, a1, b1)

compute_est_posterior <- length(compute_posterior[which(compute_posterior < 0.3)])/10000

cat("\n", "Estimated Posterior Probability: ", round(compute_est_posterior,4))

cat("\n", "Exact Posterior Probability: ", round(exact_posterior,4))


phi <- log(compute_posterior/(1 - compute_posterior))

hist(phi, probability = TRUE, breaks = 100)

lines(density(phi), col="red", ylim = 1.5)

#QUESTION 2

rm(list = ls())

y_i <- c(33, 24, 48, 32, 55, 74, 23, 76, 17)
mu = 3.5

#PART A
nDraws = 10000
tau_sq <- sum((log(y_i) - mu)^2)/length(y_i) #given expression for tau^2
X <- rchisq(nDraws, length(y_i)) #draw x from chi-sq with n df
sig_sq <- length(y_i) * tau_sq/X #compute for inv-chi-sq draws

hist(sig_sq, probability = TRUE, breaks = 100)
lines(density(sig_sq), col = "red", ylim = 10)


#PART B

#CDF of std normal with mean = 0, sd = 1, data = sigma^2
phi <- pnorm(sqrt(sig_sq/2))

#Gini Coefficient using given expression
G <- 2*phi - 1

hist(G, breaks = 100, probability = TRUE)
lines(density(G), col = "red")


#PART C

#ETI cutoffs for 2.5% and 97.5% cutoffs
margin <- quantile(G, probs = c(0.025, 0.975))

hist(G, breaks = 100, probability = TRUE)
lines(density(G), col = "red")
abline(v = margin, col = "blue")
```

```r
#PART D
G_density <- density(G) #convert G into density
G_df <- data.frame(x = G_density$x, y = G_density$y) #create df of density
G_df <- G_df[order(G_df$y, decreasing = TRUE),] #order df to find cutoff
G_df$cum_y <- cumsum(G_df$y) #add column of cum density in ordered df
G_df_hdi <- G_df[which(G_df$cum_y < 0.95*sum(G_df$y)),] #subset density samples outside HPDI
hdi_low <- min(G_df_hdi$x) #lower HPDI cutoff
hdi_max <- max(G_df_hdi$x) #upper HPDI cutoff

hist(G, breaks = 100, probability = TRUE)
lines(density(G), col = "red")
abline(v = margin, col = "blue")
abline(h = G_df_hdi[G_df_hdi$x == hdi_low,2], col = "green")
abline(v = hdi_low, col = "green", lty  = 2)
abline(v = hdi_max, col = "green", lty  = 2)


#QUESTION 3

rm(list = ls())

y <- c(1.83, 2.02, 2.33, -2.79, 2.07, 2.02, -2.44, 2.14, 2.54, 2.23)
mu = 2.51

kappa <- seq(0.01,10, by = 0.001)

posterior <- function(k, y, mu){
  post_pdf <- exp(k * (sum(cos(y - mu)) - 1)) / (besselI(k, nu = 0)^length(y))
  return(post_pdf)
}

#draws from the posterior
posterior_test <- posterior(kappa, y, mu)

#normalize the posterior
posterior_density <- posterior_test/sum(kappa*posterior_test)

plot(kappa, posterior_density, type = 'l')


#find the kappa value of the posterior mode
k_mode = kappa[which.max(posterior_density)]

#find the posterior mode
posterior_k_mode <- posterior_density[which.max(posterior_density)]

cat("Approximate Posterior Mode of kappa: ", round(posterior_k_mode,6), "\n")
cat("Kappa value for posterior mode:", round(k_mode,3), "\n")

plot(kappa, posterior_density, type = 'l')
abline(v = k_mode)
```