Bayesian Lab1

Mohamed Ali{mohal954} & Aderinto Adesijibomi{adead268}

4/5/2022

1.Daniel Bernoulli

1a)

```
library(ggplot2)
# Initial parameters
alpha0 = 5
beta0 = 5
n = 50
s = 13
f = n - s
#Posterior distribution
new_alpha = alpha0 + s
new_beta = beta0 + f
```

We calculate the true mean and true standard deviation of the beta distribution as

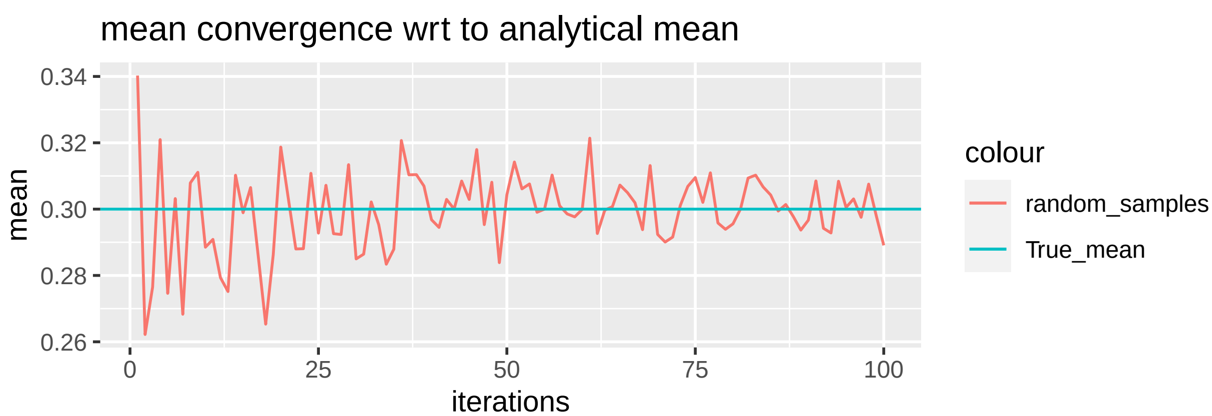
$$E(\theta) = \frac{\alpha}{\alpha + \beta}$$

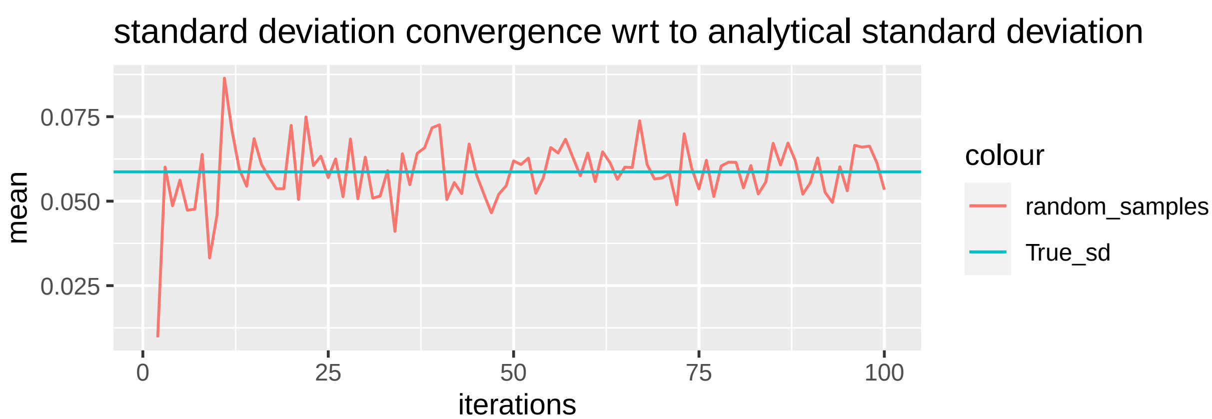
\

$$Var(\theta) = \frac{\alpha\beta}{(\alpha + \beta)^2(\alpha + \beta + 1)}$$

```
true_mean = new_alpha/(new_alpha + new_beta)
true_sd = sqrt((new_alpha*new_beta) / ((new_alpha + new_beta)^2 * (new_alpha + new_beta +1)))

mres = c()
for(i in 1:100){
  mres[i] = mean(rbeta(n = i, shape1 = new_alpha, shape2 = new_beta))
}

sres = c()
for(i in 1:100){
  sres[i] = sd(rbeta(n = i, shape1 = new_alpha, shape2 = new_beta))
}

plot1a = data.frame(x = 1:100,true_mean = true_mean,true_sd = true_sd)
```

```r
mean <- ggplot(data = plot1a, aes(x = plot1a$x)) +
  geom_line(aes(x = x, y = mres, colour = "random_samples")) +
  geom_hline(aes(yintercept = true_mean, colour = "True_mean")) +
  ggtitle("mean convergence wrt to analytical mean") + xlab("iterations") + ylab("mean")

sd <- ggplot(data = plot1a, aes(x = plot1a$x)) +
  geom_line(aes(x = x, y = sres, colour = "random_samples"))+
  geom_hline(aes(yintercept = true_sd,colour = "True_sd")) +
  ggtitle("standard deviation convergence wrt to analytical standard deviation") + xlab("iterations") +

library(gridExtra)
grid.arrange(mean,sd,nrow=2)
```





1b) n = (ndraws=10000) to compute posterior probability = $\Pr(\theta < 0.3|y)$

```r
library(gridExtra)
ndraws <- 10000
sample_draws1b <- rbeta(n =ndraws, shape1 = new_alpha, shape2 = new_beta)
sample_condition = ifelse(sample_draws1b < 0.3, 1, 0)
prob_sample_draws = sum(sample_condition)/ndraws

Beta_post = round(pbeta(q = 0.3, shape1 = new_alpha, shape2 = new_beta),3)

result <- data.frame("Expected value" = Beta_post, "Simulated vaule" = prob_sample_draws)
knitr::kable(result)
```

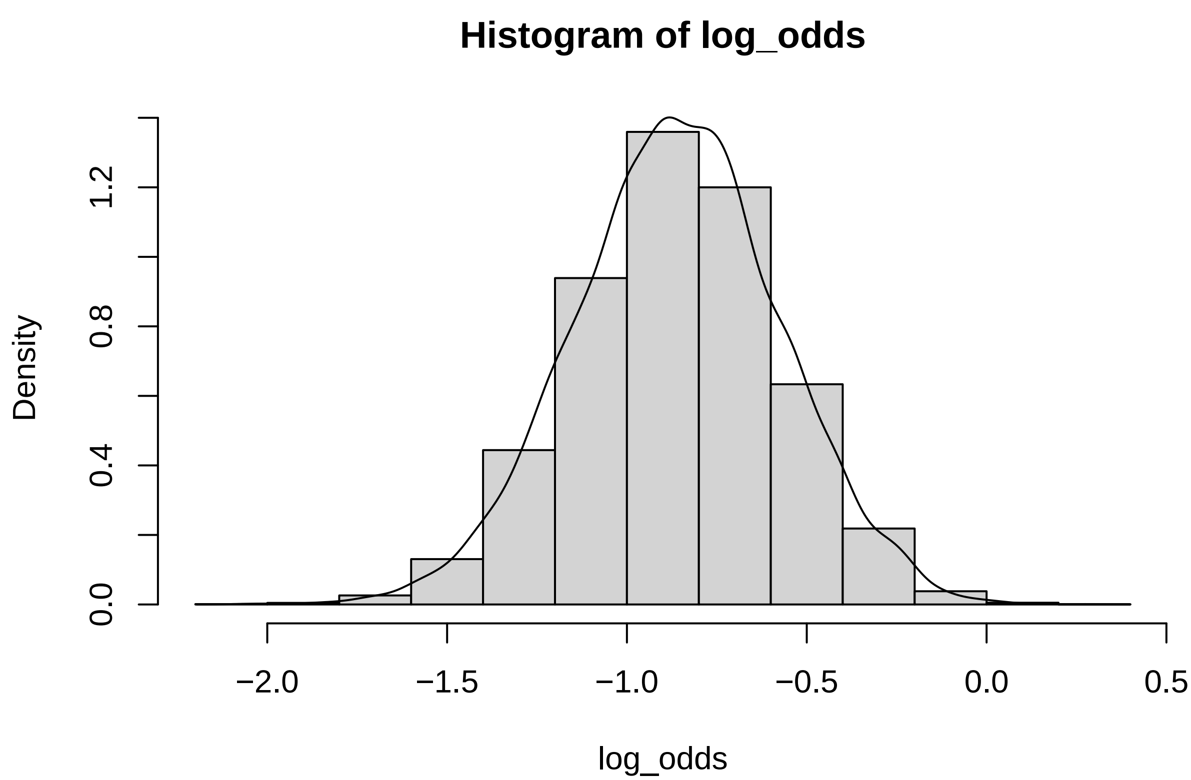
Expected.value	Simulated.vaule
0.515	0.5177

comments Posterior probabilities values and Analytical values are similar

c)

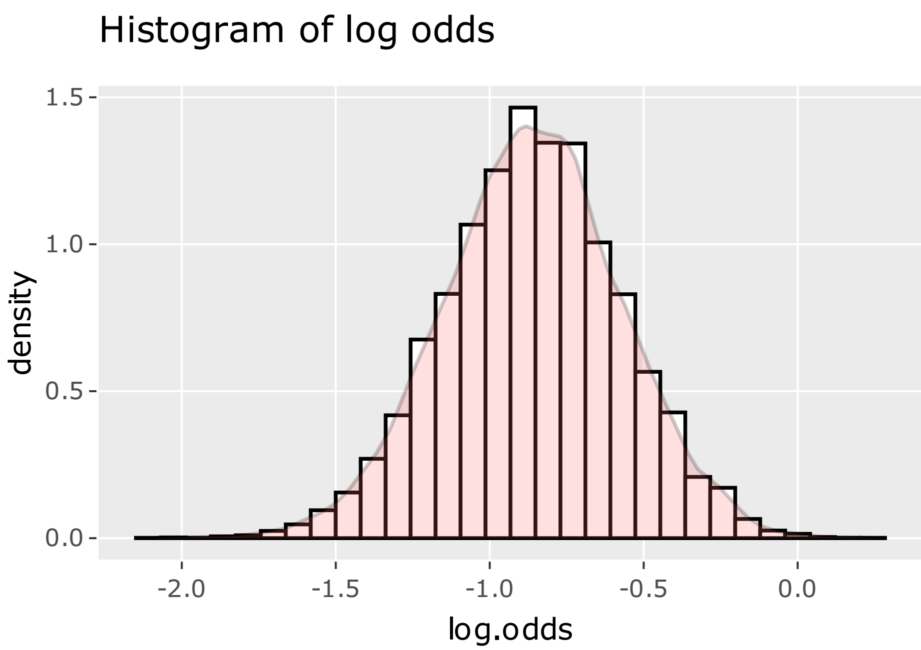
.

```
library(plotly)
ndraws = 10000
sample_draws1c = rbeta(n = ndraws, shape1 = new_alpha, shape2 = new_beta)
log_odds = log(sample_draws1c/(1-sample_draws1c))

hist(log_odds, probability = TRUE)
lines(density(log_odds))
```




```
plot1c = data.frame("Draw" = 1:ndraws,"log-odds" = log_odds)

ggplotly(ggplot(data = plot1c, aes(x=log.odds)) +
 geom_histogram(aes(y=..density..),
                colour="black",
```

```
                 fill="white",
                 bins=30)+
  geom_density(alpha=.2, fill="#FF6666") +
  ggtitle("Histogram of log odds"))
```



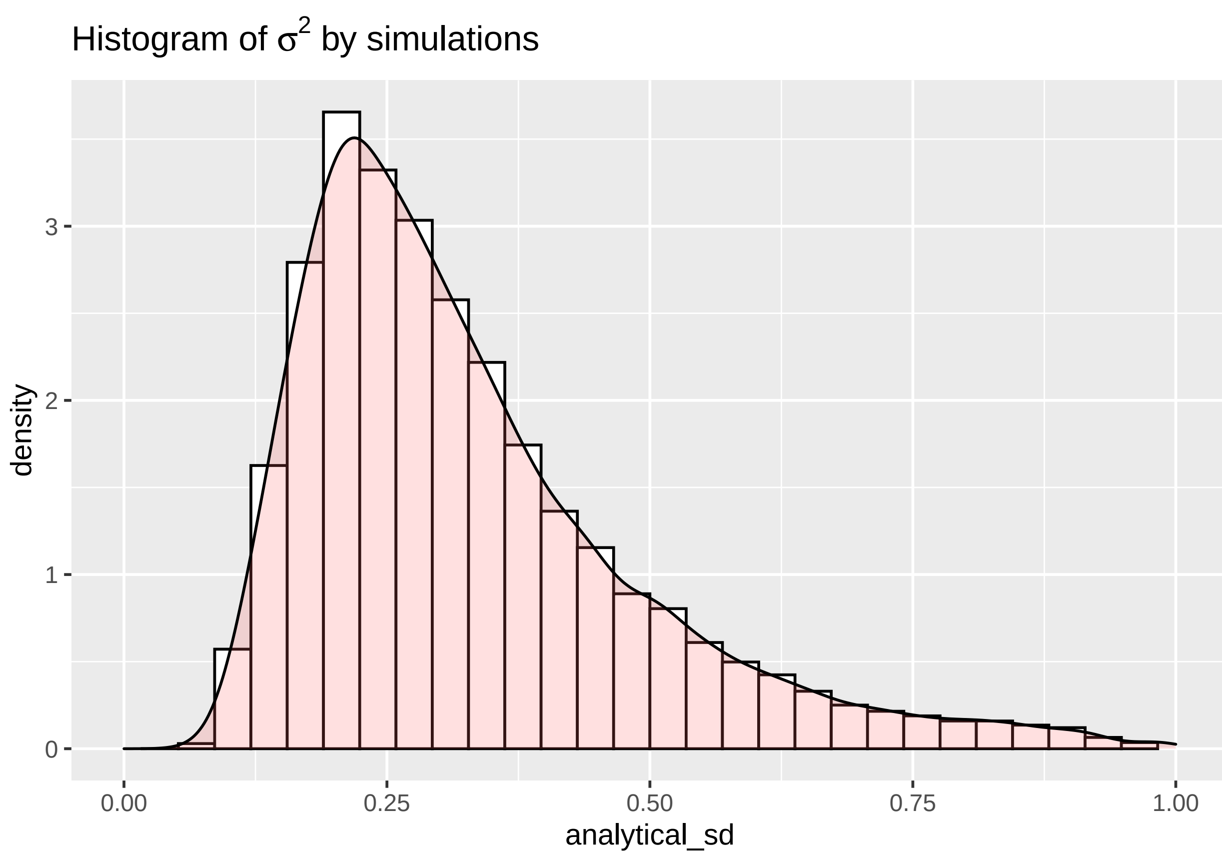
2. Log-normal distribution and the Gini coefficient.

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

2a).

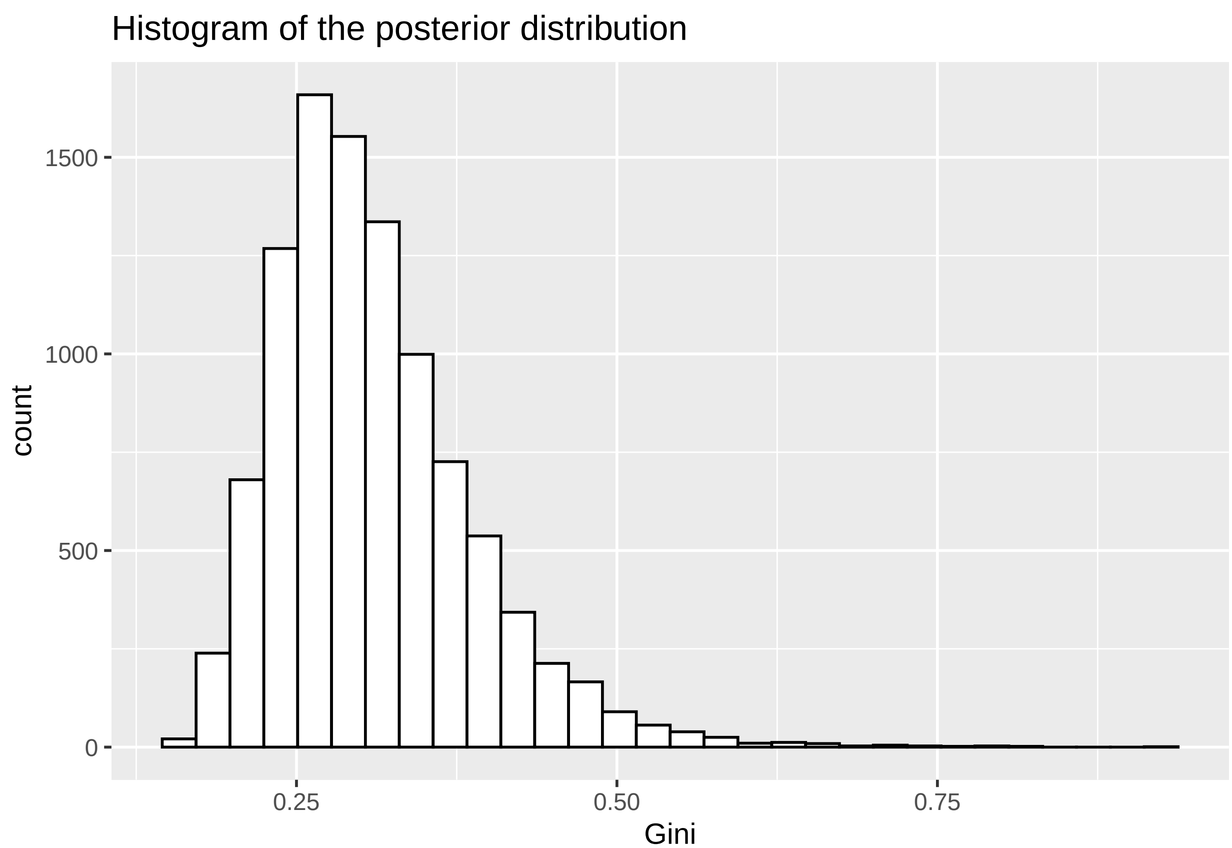
```
library(latex2exp)
library(LaplacesDemon)
set.seed(12345)
ndraws = 10000

observed_values <- c(33,24,48,32,55,74,23,76, 17)
n_values <- length(observed_values)
m = 3.5


tau <- function(y,m=3.5){
  n <- length(y)
  result <- sum((log(y)-m)^2)/n
```

```r
  return(result)
}

llk <- tau(observed_values)

#### from slide 5 lecture 3
analytical_sd <- c()
for (i in 1:ndraws) {
  X <- rchisq(1, n_values)
  analytical_sd[i] <-  (n_values) * llk / X
}

data2a <- data.frame(analytical_sd)

plot2a <- ggplot(data2a,aes(x=analytical_sd))+
 geom_histogram(aes(y=..density..),
                 colour="black",
                 fill="white",
                bins = 30)+
  geom_density(alpha=.2, fill="#FF6666") +
  ggtitle(TeX("Histogram of $\\sigma^2$ by simulations"))+
  scale_x_continuous(limits = c(0,1))
plot2a
```



2b).

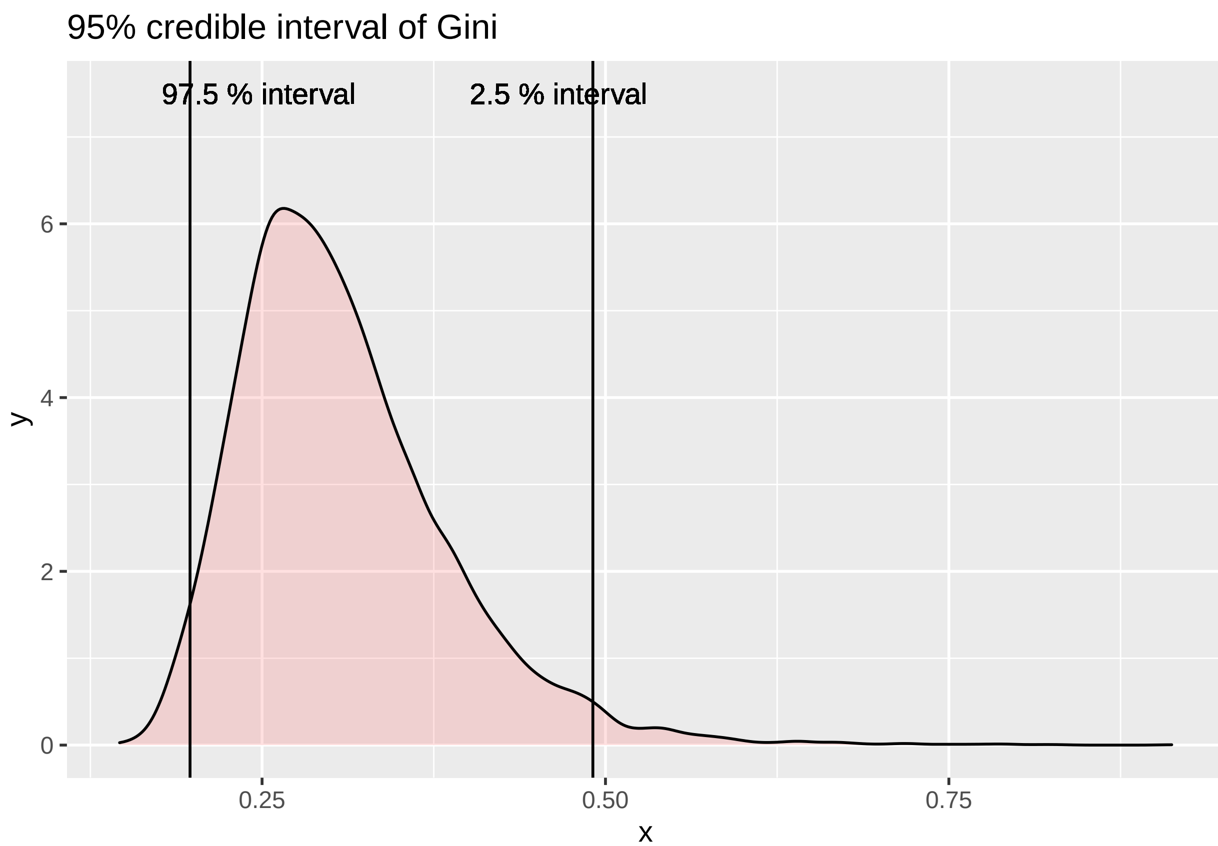
```
Gini <- c()
Gini <- 2 * pnorm(sqrt(analytical_sd)/sqrt(2), mean = 0, sd = 1) -1

plot2b = data.frame("G" = Gini,"nr" = 1:length(Gini))

ggplot(data = plot2b,aes(x=Gini)) +
  ggtitle("Histogram of the posterior distribution") +
   geom_histogram(aes(x=Gini),
                 colour="black",
                 fill="white",
                 bins=30)
```



2(c) tail credible interval for G

```
a =0.025
b = 0.975
CI_a = quantile(Gini, probs = a)
CI_b = quantile(Gini, probs = b)
```

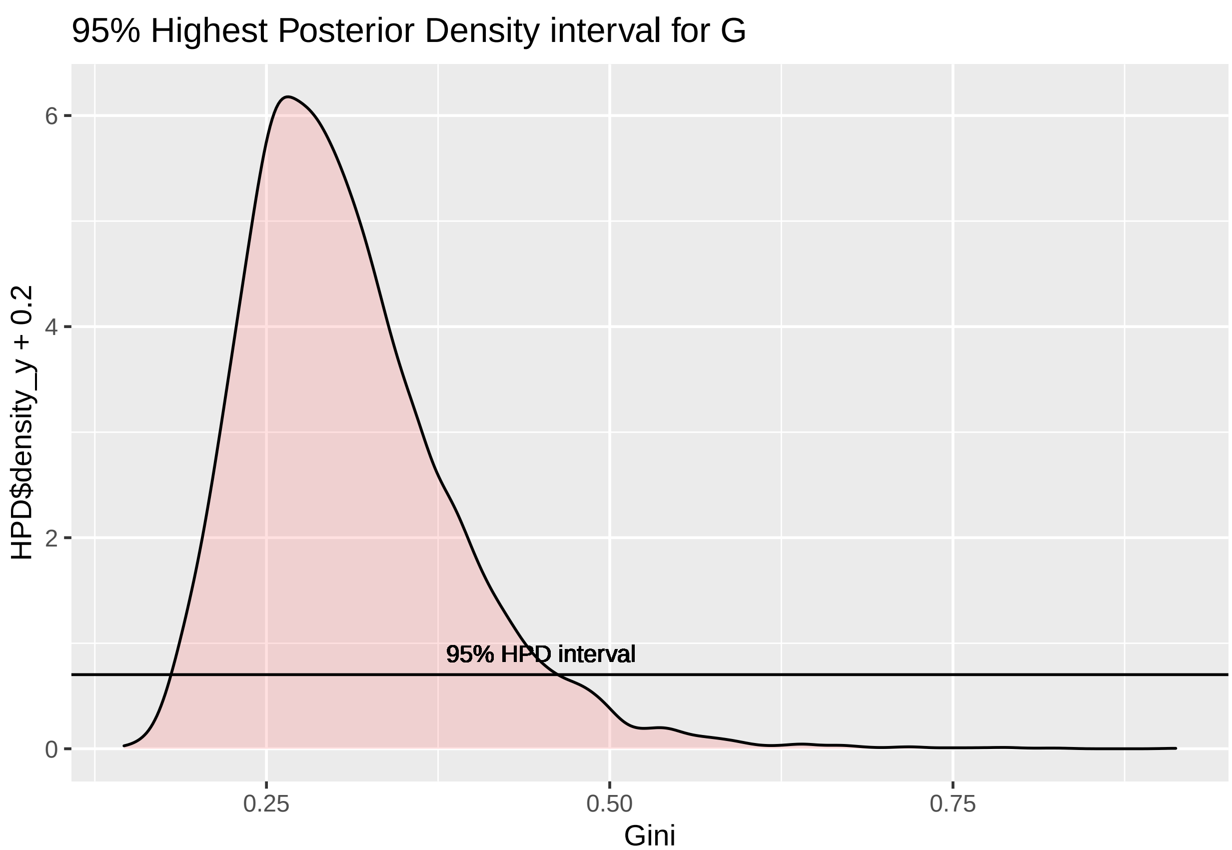
```r
ggplot(data = plot2b, aes(x = Gini)) +
  ggtitle("95% credible interval of Gini") +
  geom_density(alpha=.2, fill="#FF6666") +
  geom_vline(aes(xintercept = CI_a)) +
  geom_vline(aes(xintercept = CI_b))+
  geom_text(aes(label ="97.5 % interval", x =  CI_a + 0.05, y = 7.5),size = 4 )+
  geom_text(aes(label ="2.5 % interval", x =  CI_b - 0.025, y = 7.5),size = 4 ) +
  xlab("x")
```



2d)

```r
density_Gini <- density(Gini)
density_y <- density_Gini$y
density_x <- density_Gini$x
density_df <- data.frame(nr = 1:length(density_x),density_x <- density_x,density_y <- density_y)
density_df_order <- density_df[order(density_y, decreasing = TRUE),]
density_df_order$cumsum_y <- cumsum(density_df_order$density_y)
density_df_order$cumsum_y_proportional_percent = density_df_order$cumsum_y/sum(density_y)*100
density_df_order$in_ci <- density_df_order$cumsum_y_proportional_percent <= 95
#data frame with just true values
density_df_order_trueci = density_df_order[(density_df_order$in_ci == TRUE),]
HPD = density_df_order_trueci[nrow(density_df_order_trueci),]
```

```r
ggplot(data = plot2b, aes(x = Gini)) +
  ggtitle("95% Highest Posterior Density interval for G") +
  geom_density(alpha=.2, fill="#FF6666") +
  geom_hline(aes(yintercept = HPD$density_y)) +
   geom_text(aes(label ="95% HPD interval", x =  0.45, y = HPD$density_y+0.2),size = 3 )
```



3. Bayesian inference

3a)

```r
# given values
radians = c(1.83, 2.02, 2.33,-2.79, 2.07, 2.02,-2.44, 2.14, 2.54, 2.2)
n_radians = length(radians)
mu = 2.51
# grid of k values
k = seq(from = 0.01,to = 7, by = 0.01) # k>0
```

equation for posterior is given by

$$posterior \propto likelihood \times prior$$

Likelihood $f(y_1, y_2, ., ., y_n)$ taking the sum of the product of the pdf

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

equals

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

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

Exponential equation for prior

$$prior \sim \begin{cases} \lambda \cdot exp^{-\lambda x} \quad where \quad \lambda = 1 \quad \kappa = x \\ \exp^{-\kappa} \end{cases}$$

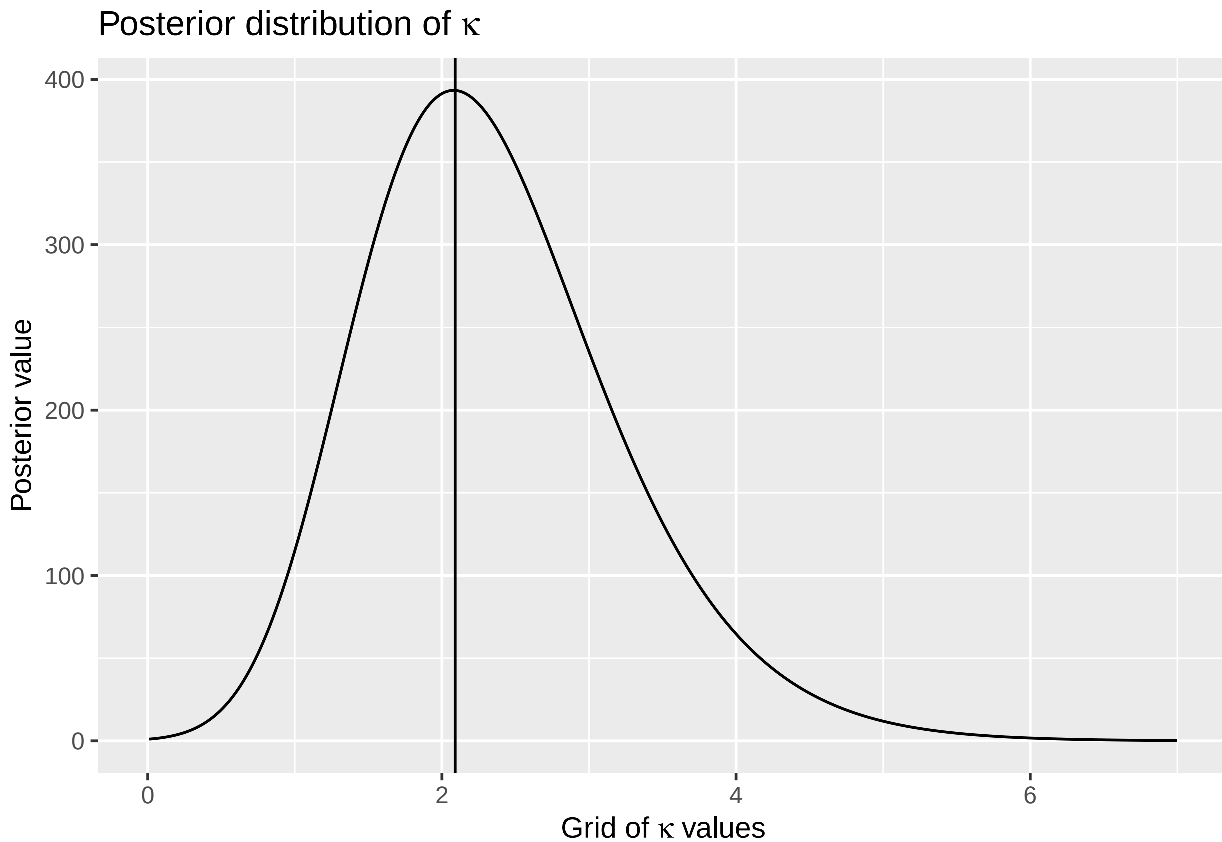
$$posterior \propto \frac{1}{I_o(\kappa)^2} exp[\kappa \cdot cos(y_i - \mu) - \kappa]$$

$$\propto \frac{1}{I_o(\kappa)^2} exp[\kappa(cos(y_i - \mu) - 1)]$$

```r
res <- exp(k*sum(cos(radians-mu))-k)/besselI(x = k, nu=0)^n_radians
plot3a = data.frame("k" = k,"posterior" = res)
plot3a = ggplot(data = plot3a, aes(x=k, y=res)) +
  geom_line() +
  ylab("Posterior value") + ggtitle(TeX("Posterior distribution of $\\kappa$"))+
  xlab(TeX("Grid of $\\kappa$ values"))
k_max_index = which.max(res)
k_max = res[k_max_index]
k_max
```

```
## [1] 393.346
```

```r
plot3a + geom_vline(aes(xintercept = (k_max_index*0.01 + 0.01)))
```



Appendix

```
knitr::opts_chunk$set(echo = TRUE)
library(ggplot2)
# Initial parameters
alpha0 = 5
beta0 = 5
n = 50
s = 13
f = n - s
#Posterior distribution
new_alpha = alpha0 + s
new_beta = beta0 + f

true_mean = new_alpha/(new_alpha + new_beta)
true_sd = sqrt((new_alpha*new_beta) / ((new_alpha + new_beta)^2 * (new_alpha + new_beta +1)))

mres = c()
for(i in 1:100){
  mres[i] = mean(rbeta(n = i, shape1 = new_alpha, shape2 = new_beta))
}

sres = c()
for(i in 1:100){
```

```r
  sres[i] = sd(rbeta(n = i, shape1 = new_alpha, shape2 = new_beta))
}

plot1a = data.frame(x = 1:100,true_mean = true_mean,true_sd = true_sd)


mean <- ggplot(data = plot1a, aes(x = plot1a$x)) +
  geom_line(aes(x = x, y = mres, colour = "random_samples")) +
  geom_hline(aes(yintercept = true_mean, colour = "True_mean")) +
  ggtitle("mean convergence wrt to analytical mean") + xlab("iterations") + ylab("mean")

sd <- ggplot(data = plot1a, aes(x = plot1a$x)) +
  geom_line(aes(x = x, y = sres, colour = "random_samples"))+
  geom_hline(aes(yintercept = true_sd,colour = "True_sd")) +
  ggtitle("standard deviation convergence wrt to analytical standard deviation") + xlab("iterations") +

library(gridExtra)
grid.arrange(mean,sd,nrow=2)

library(gridExtra)
ndraws <- 10000
sample_draws1b <- rbeta(n =ndraws, shape1 = new_alpha, shape2 = new_beta)
sample_condition = ifelse(sample_draws1b < 0.3, 1, 0)
prob_sample_draws = sum(sample_condition)/ndraws

Beta_post = round(pbeta(q = 0.3, shape1 = new_alpha, shape2 = new_beta),3)

result <- data.frame("Expected value" = Beta_post, "Simulated vaule" = prob_sample_draws)
knitr::kable(result)
library(plotly)
ndraws = 10000
sample_draws1c = rbeta(n = ndraws, shape1 = new_alpha, shape2 = new_beta)
log_odds = log(sample_draws1c/(1-sample_draws1c))

hist(log_odds, probability = TRUE)
lines(density(log_odds))

plot1c = data.frame("Draw" = 1:ndraws,"log-odds" = log_odds)

ggplotly(ggplot(data = plot1c, aes(x=log.odds)) +
 geom_histogram(aes(y=..density..),
                colour="black",
                fill="white",
                bins=30)+
  geom_density(alpha=.2, fill="#FF6666") +
  ggtitle("Histogram of log odds"))

library(latex2exp)
library(LaplacesDemon)
set.seed(12345)
ndraws = 10000

observed_values <- c(33,24,48,32,55,74,23,76, 17)
```

```r
n_values <- length(observed_values)
m = 3.5


tau <- function(y,m=3.5){
  n <- length(y)
  result <- sum((log(y)-m)^2)/n
  return(result)
}

llk <- tau(observed_values)

#### from slide 5 lecture 3
analytical_sd <- c()
for (i in 1:ndraws) {
  X <- rchisq(1, n_values)
  analytical_sd[i] <-  (n_values) * llk / X
}

data2a <- data.frame(analytical_sd)

plot2a <- ggplot(data2a,aes(x=analytical_sd))+
 geom_histogram(aes(y=..density..),
                 colour="black",
                 fill="white",
                bins = 30)+
  geom_density(alpha=.2, fill="#FF6666") +
  ggtitle(TeX("Histogram of $\\sigma^2$ by simulations"))+
  scale_x_continuous(limits = c(0,1))
plot2a

Gini <- c()
Gini <- 2 * pnorm(sqrt(analytical_sd)/sqrt(2), mean = 0, sd = 1) -1

plot2b = data.frame("G" = Gini,"nr" = 1:length(Gini))

ggplot(data = plot2b,aes(x=Gini)) +
  ggtitle("Histogram of the posterior distribution") +
   geom_histogram(aes(x=Gini),
                 colour="black",
                 fill="white",
                 bins=30)

a =0.025
b = 0.975
CI_a = quantile(Gini, probs = a)
CI_b = quantile(Gini, probs = b)

ggplot(data = plot2b, aes(x = Gini)) +
  ggtitle("95% credible interval of Gini") +
  geom_density(alpha=.2, fill="#FF6666") +
  geom_vline(aes(xintercept = CI_a)) +
  geom_vline(aes(xintercept = CI_b))+
```

```r
  geom_text(aes(label ="97.5 % interval", x =  CI_a + 0.05, y = 7.5),size = 4 )+
  geom_text(aes(label ="2.5 % interval", x =  CI_b - 0.025, y = 7.5),size = 4 ) +
  xlab("x")
density_Gini <- density(Gini)
density_y <- density_Gini$y
density_x <- density_Gini$x
density_df <- data.frame(nr = 1:length(density_x),density_x <- density_x,density_y <- density_y)
density_df_order <- density_df[order(density_y, decreasing = TRUE),]
density_df_order$cumsum_y <- cumsum(density_df_order$density_y)
density_df_order$cumsum_y_proportional_percent = density_df_order$cumsum_y/sum(density_y)*100
density_df_order$in_ci <- density_df_order$cumsum_y_proportional_percent <= 95
#data frame with just true values
density_df_order_trueci = density_df_order[(density_df_order$in_ci == TRUE),]
HPD = density_df_order_trueci[nrow(density_df_order_trueci),]
ggplot(data = plot2b, aes(x = Gini)) +
  ggtitle("95% Highest Posterior Density interval for G") +
  geom_density(alpha=.2, fill="#FF6666") +
  geom_hline(aes(yintercept = HPD$density_y)) +
   geom_text(aes(label ="95% HPD interval", x =  0.45, y = HPD$density_y+0.2),size = 3 )
# given values
radians = c(1.83, 2.02, 2.33,-2.79, 2.07, 2.02,-2.44, 2.14, 2.54, 2.2)
n_radians = length(radians)
mu = 2.51
# grid of k values
k = seq(from = 0.01,to = 7, by = 0.01) # k>0
res <- exp(k*sum(cos(radians-mu))-k)/besselI(x = k, nu=0)^n_radians
plot3a = data.frame("k" = k,"posterior" = res)
plot3a = ggplot(data = plot3a, aes(x=k, y=res)) +
  geom_line() +
  ylab("Posterior value") + ggtitle(TeX("Posterior distribution of $\\kappa$"))+
  xlab(TeX("Grid of $\\kappa$ values"))
k_max_index = which.max(res)
k_max = res[k_max_index]
k_max
plot3a + geom_vline(aes(xintercept = (k_max_index*0.01 + 0.01)))
```