Bayesian Learning

Lab 1

Emil K Svensson and Rasmus Holm

2017-04-11

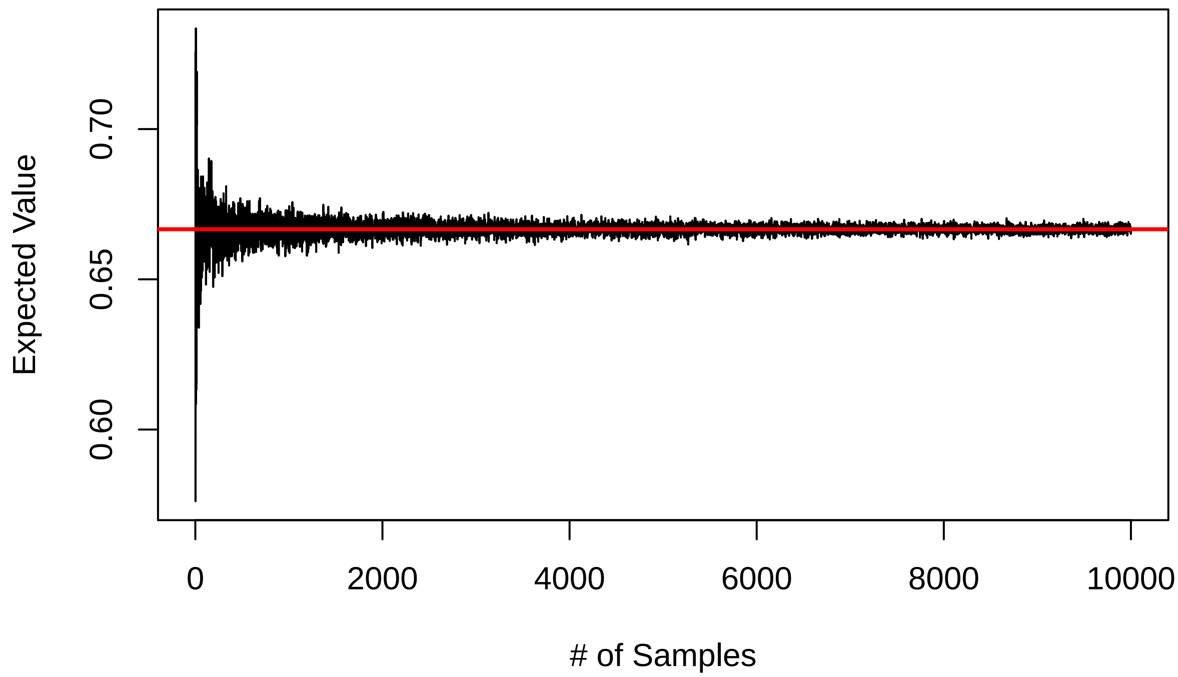
Question 1

```
n <- 20
s <- 14
f <- n - s

alpha_prior <- 2
beta_prior <- 2
```

a

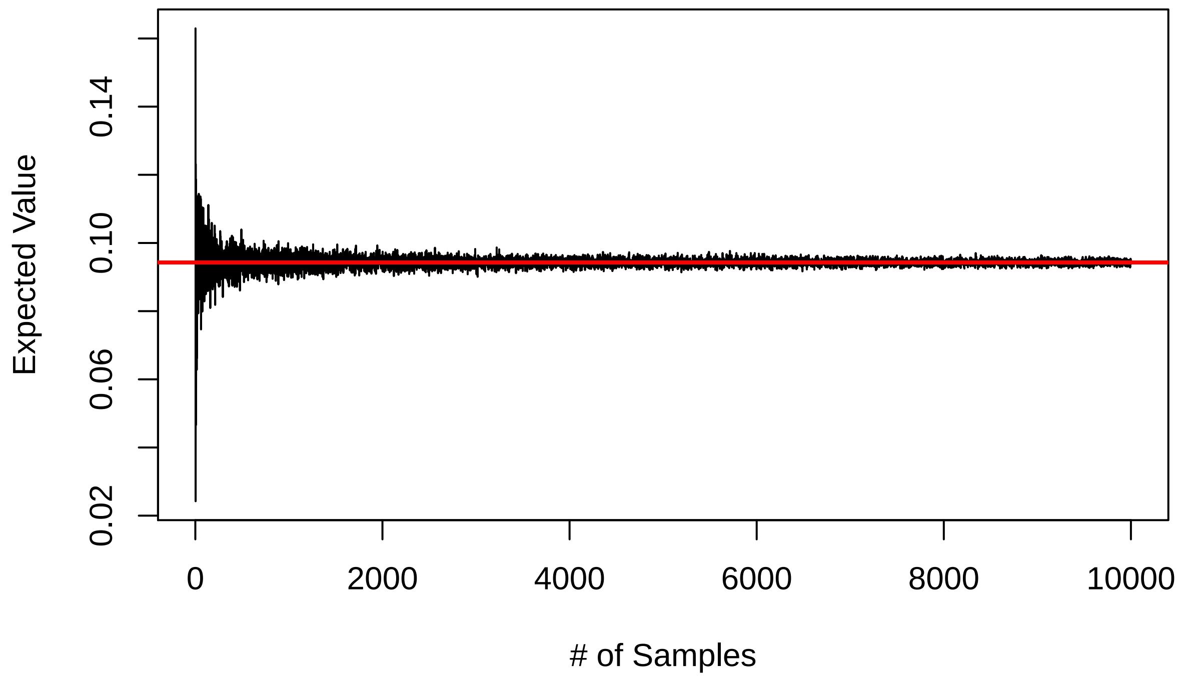
```
drawSamples <- function(nDraws, alpha, beta) {
    rbeta(nDraws, shape1=alpha, shape2=beta)
}

nDraws <- 10000
alpha <- alpha_prior + s
beta <- beta_prior + f

set.seed(123)
samples <- drawSamples(nDraws, alpha, beta)

true_mean <- alpha / (alpha + beta)
estimated_mean <- mean(samples)

true_sd <- sqrt((alpha * beta) / ((alpha + beta)^2 * (alpha + beta + 1)))
estimated_sd <- sd(samples)

set.seed(123)
betamean <- sapply(1:nDraws, FUN = function(X) {
    mean(drawSamples(nDraws = X, alpha = alpha , beta = beta))
})

set.seed(123)
betasd <- sapply(1:nDraws, FUN = function(X) {
    sd(drawSamples(nDraws = X, alpha = alpha , beta = beta))
})
```

```
plot(x = 1:nDraws, y = betamean, type = "l",
     xlab="# of Samples", ylab="Expected Value")
abline(h = true_mean, col = "red", lwd = 2)
```



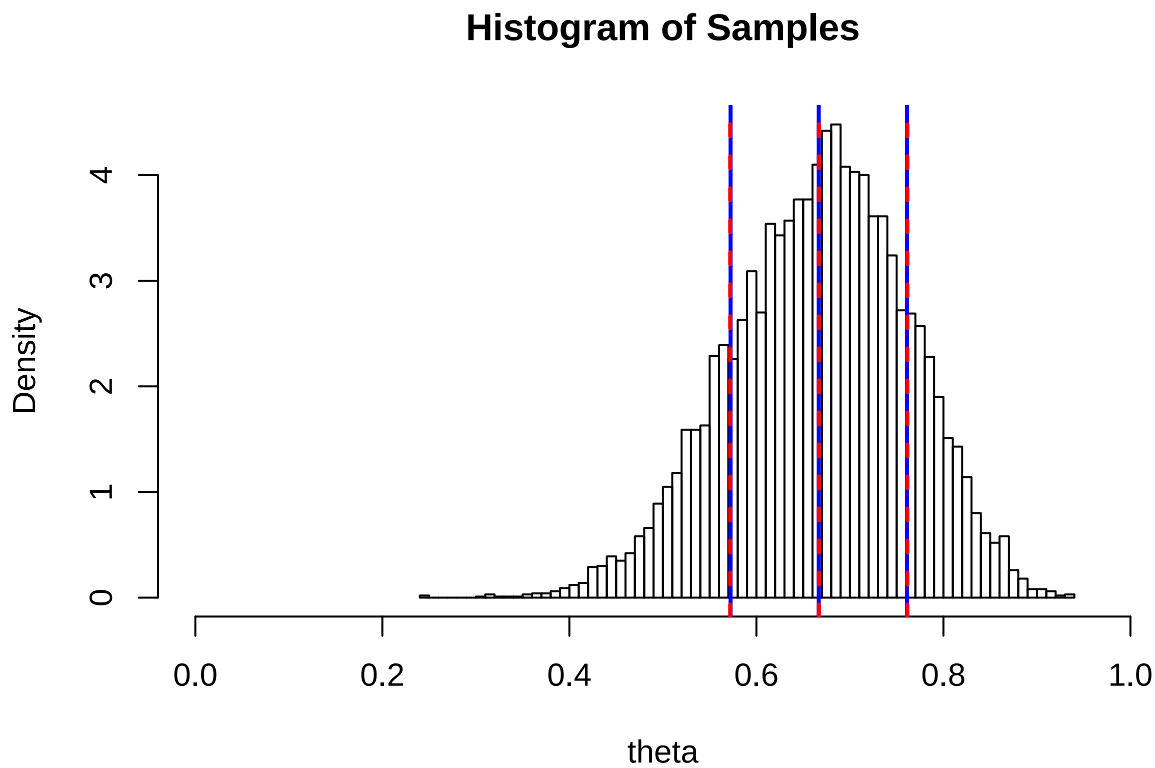
The sample mean for the beta distribution seem to be converging at the beta distributions expected mean here represented by the blue line.

```
plot(x = 1:nDraws, y = betasd, type = "l",
     xlab="# of Samples", ylab="Expected Value")
abline(h = true_sd, col = "red", lwd = 2)
```



The sample standard deviation also converges at the expected standard deviation as expected.

```
hist(samples, freq=FALSE, breaks=50,
     xlab="theta", main="Histogram of Samples", xlim=c(0, 1))

abline(v=true_mean, col="blue", lwd=2)
abline(v=true_mean + true_sd, col="blue", lwd=2)
abline(v=true_mean - true_sd, col="blue", lwd=2)

abline(v=estimated_mean, col="red", lwd=2, lty=2)
abline(v=estimated_mean + estimated_sd, col="red", lwd=2, lty=2)
abline(v=estimated_mean - estimated_sd, col="red", lwd=2, lty=2)
```



Here we can see the expected mean and standard deviations for 10000 samples and they seem to be very close to each other.

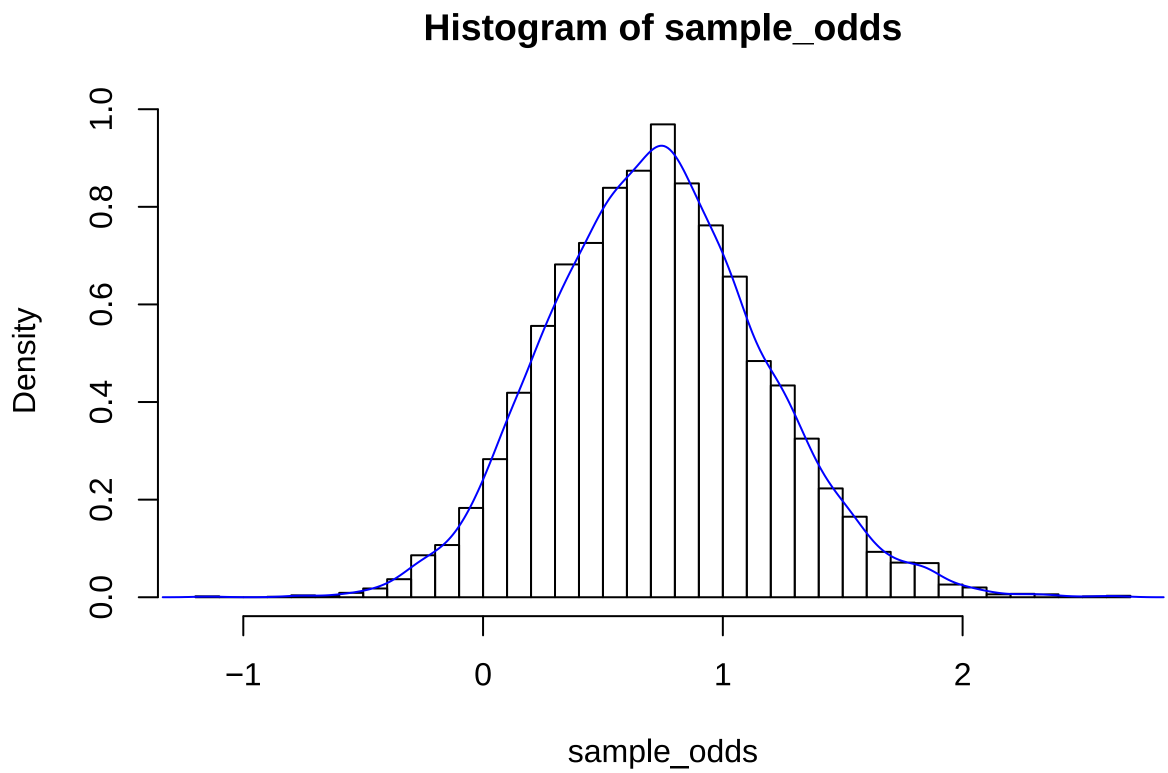
b

```
est_04 <- sum(samples < 0.4) / length(samples)
true_04 <- pbeta(0.4, shape1 = alpha, shape2 = beta)
```

$Pr(\theta < 0.4|y) \approx 0.0035$ The true value being $Pr(\theta < 0.4|y) = 0.0039727$

c

```
log_odds <- function(theta){
  log(theta / (1 - theta))
}

sample_odds <- log_odds(samples)

hist(sample_odds, breaks=50, freq = FALSE)
lines(density(sample_odds), col = "blue")
```



A thing of beauty.

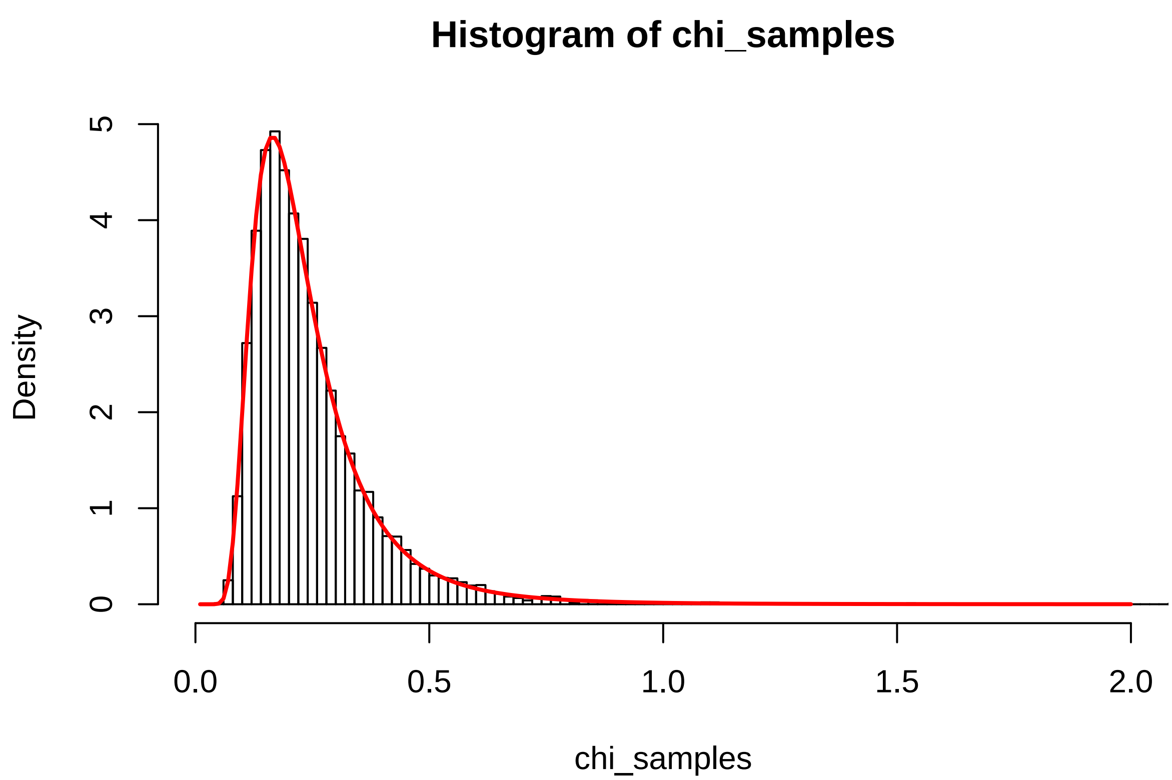
Question 2

a

```r
library(geoR)
sek <- c(14, 25, 45, 25, 30, 33, 19, 50, 34 ,67)

n <- length(sek)
mu <- 3.5
tausq <- sum((log(sek)- mu)^2) / n

set.seed(123)
chi_samples <- rinvchisq(nDraws, df = n, scale = tausq)
```

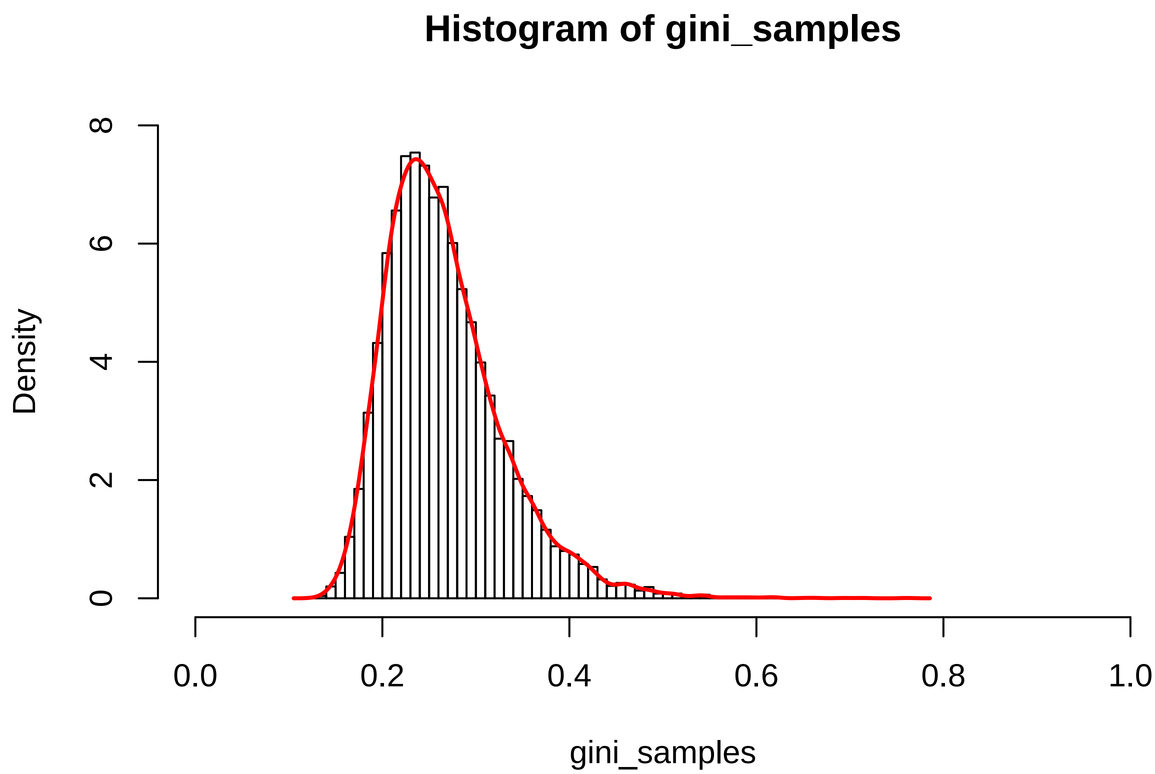
```r
invchisq_pdf <- function(x, df, scale) {
    factor1 <- (scale * df / 2)^(df / 2) / gamma(df / 2)
    factor2 <- exp(-(df * scale) / (2 * x)) / x^(1 + df / 2)
    factor1 * factor2
}

xs <- seq(0.01, 2, 0.01)
ys <- invchisq_pdf(xs, 10, tausq)
```

```r
hist(chi_samples, breaks = 100, freq=FALSE, xlim=c(0, 2))
lines(xs, ys, col="red", lwd=2)
```



b

```
gini <- function(sigma){
    2 * pnorm(sigma / sqrt(2), mean = 0 , sd = 1) - 1
}

gini_samples <- gini(sqrt(chi_samples))

hist(gini_samples, breaks = 50, freq = FALSE,
     ylim = c(0,8), xlim=c(0, 1))
lines(density(gini_samples), col = "red", lwd = 2)
```



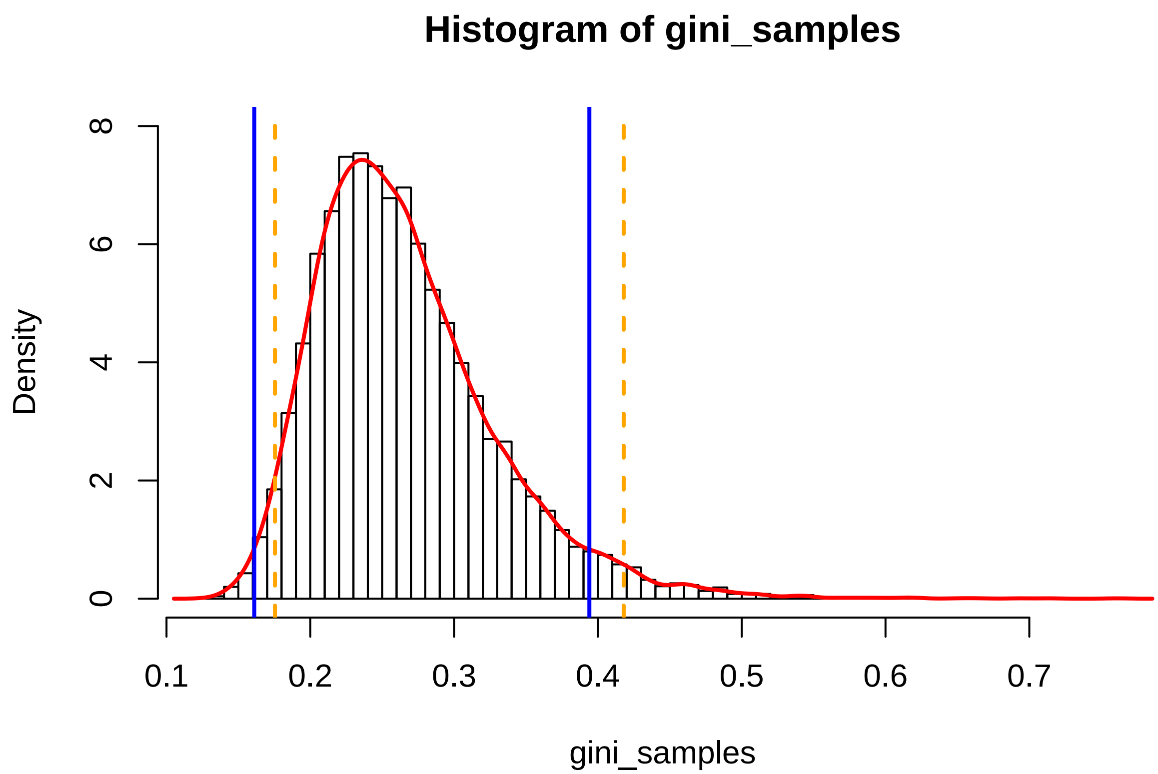
c

```
gini_cred <- quantile(gini_samples, probs = c(0.025, 1 - 0.025))
```

```
dense_gini <- density(gini_samples)
dense_gini_cdf <- dense_gini$y/sum(dense_gini$y)
index_gini <- order(dense_gini_cdf)
dense_gini_cdf <- dense_gini_cdf[index_gini]

while(sum(dense_gini_cdf) > 0.95){
    dense_gini_cdf <- dense_gini_cdf[-1]
    index_gini <- index_gini[-1]
```

```
}

lower <- min(dense_gini$x[index_gini])
upper <- max(dense_gini$x[index_gini])

gini_HPD <- c(lower, upper)
```

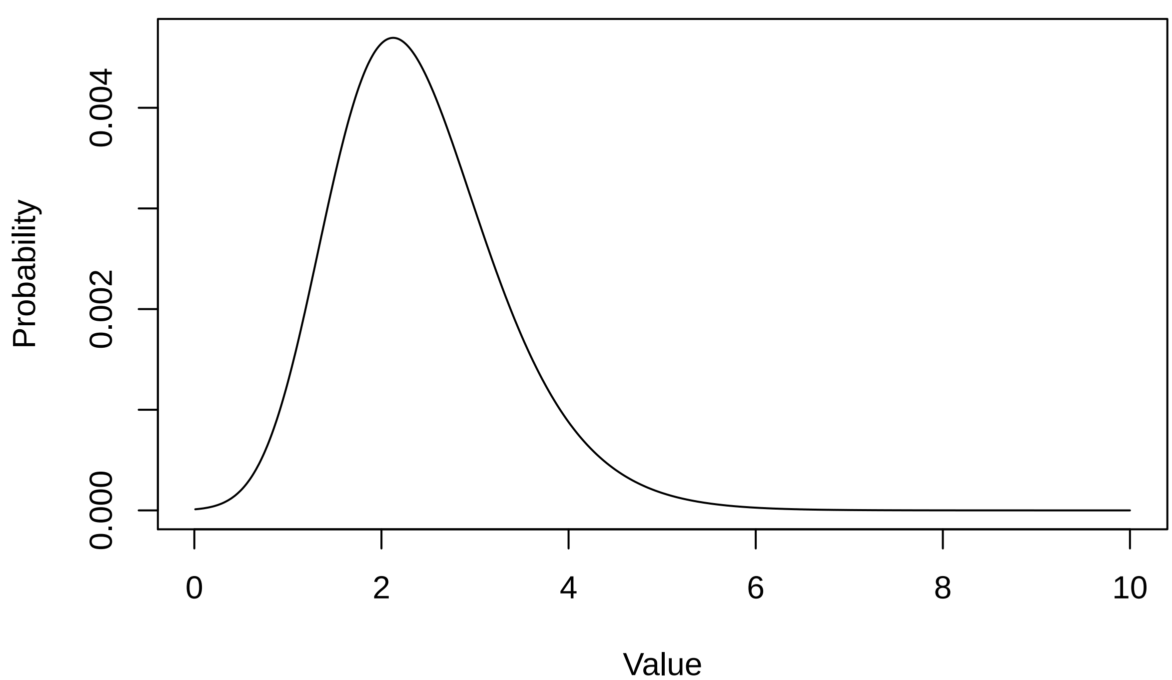
```
hist(gini_samples, breaks = 50, freq = FALSE, ylim = c(0, 8))
lines(density(gini_samples), col = "red", lwd = 2)
abline(v = gini_HPD, col = "blue", lwd = 2)
abline(v = gini_cred, col = "orange", lwd = 2, lty = 2)
```



Question 3

a

```r
vonMises <- function(y,K,mu){
    exp(K * cos(y - mu)) /(2 * pi * besselI(K, nu = 0))
}

sample_exp <- seq(0.01, 10, 0.01)

prior_exp <- dexp(sample_exp, rate = 1)
y <- c(-2.44, 2.14, 2.54, 1.83, 2.02, 2.33, -2.79, 2.23, 2.07, 2.02)

sample_mises <- t(sapply(y, function(X) vonMises(y = X, K = sample_exp, mu = 2.39)))

mises_like <- apply(sample_mises, MARGIN = 2, FUN = prod)

post_mises <- mises_like * prior_exp
post_mises <- post_mises / sum(post_mises)

plot(sample_exp, post_mises, type="l", ylab="Probability", xlab="Value")
```



b

```
sample_exp[which.max(post_mises)]
```

```
## [1] 2.12
```