Computer Lab 1

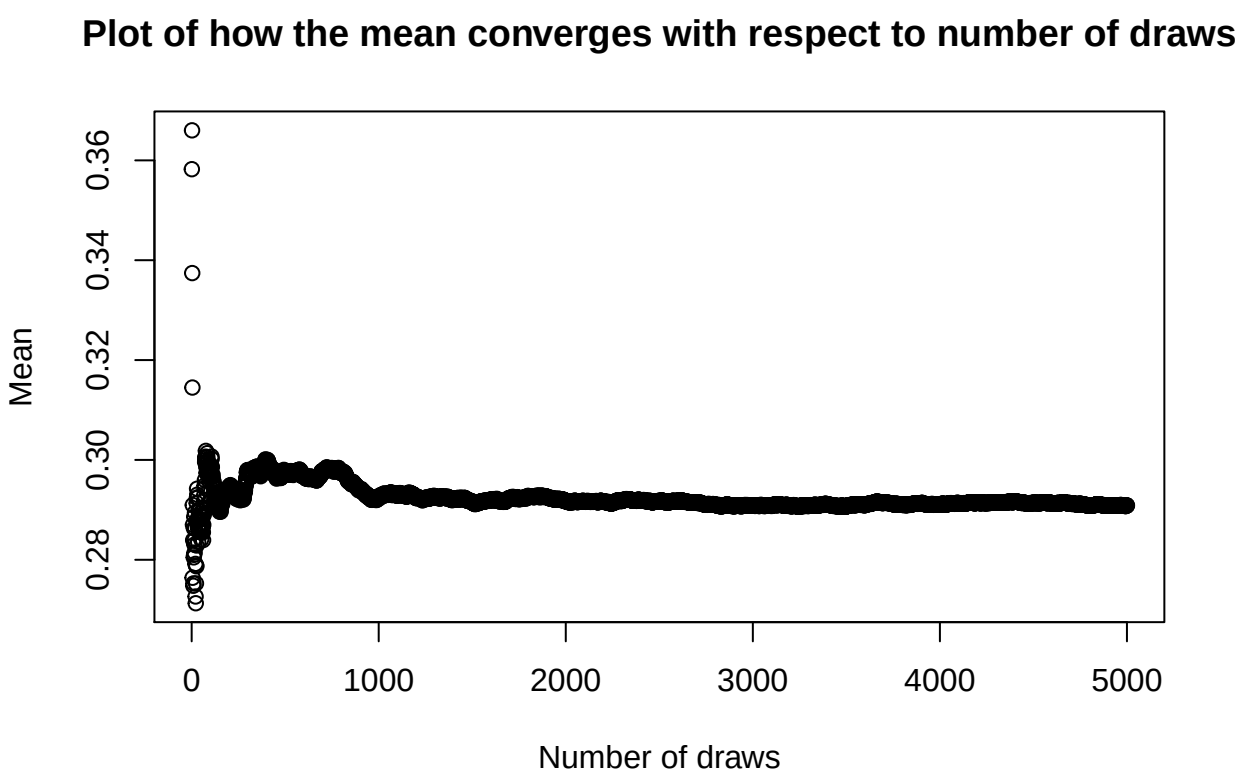
Christian von Koch, William Anzén

2020-04-17

Assignment 1

Task a)

```
set.seed(12345)
alpha0=2
beta0=2
s=5
f=15
n=20

# Function for calculating the mean of a beta-distribution
calcMeanBeta = function(alpha, beta) {
  return(alpha/(alpha+beta))
}

# Function for calculating the standard deviation of a beta-distribution
calcStdDevBeta = function(alpha, beta) {
  return(sqrt(alpha*beta/((alpha+beta)^2*(alpha+beta+1))))
}

# Function for calculating the mean squared error of drawed data
calcMSE = function(n, mean, data){
  return(sqrt(1/(n-1)*sum((data-mean)^2)))
}

# Function for drawing random values from the betadistribution
drawBetaValues = function(n, alpha, beta) {
  return(rbeta(n, alpha, beta))
}

MeanOfPosterior = calcMeanBeta(alpha0+s, beta0+f)
StdOfPosterior = calcStdDevBeta(alpha0+s, beta0+f)
nVector = seq(1, 5000, 1)
meanVector=c()
stdVector=c()
for (i in nVector) {
  set.seed(12345)
  betaValues= drawBetaValues(i, alpha0+s, beta0+f)
  meanVector=c(meanVector, mean(betaValues))
  stdVector=c(stdVector, calcMSE(i, mean(betaValues), betaValues))
}
plot(nVector, meanVector,
     main="Plot of how the mean converges with respect to number of draws",
     xlab="Number of draws", ylab="Mean")
```

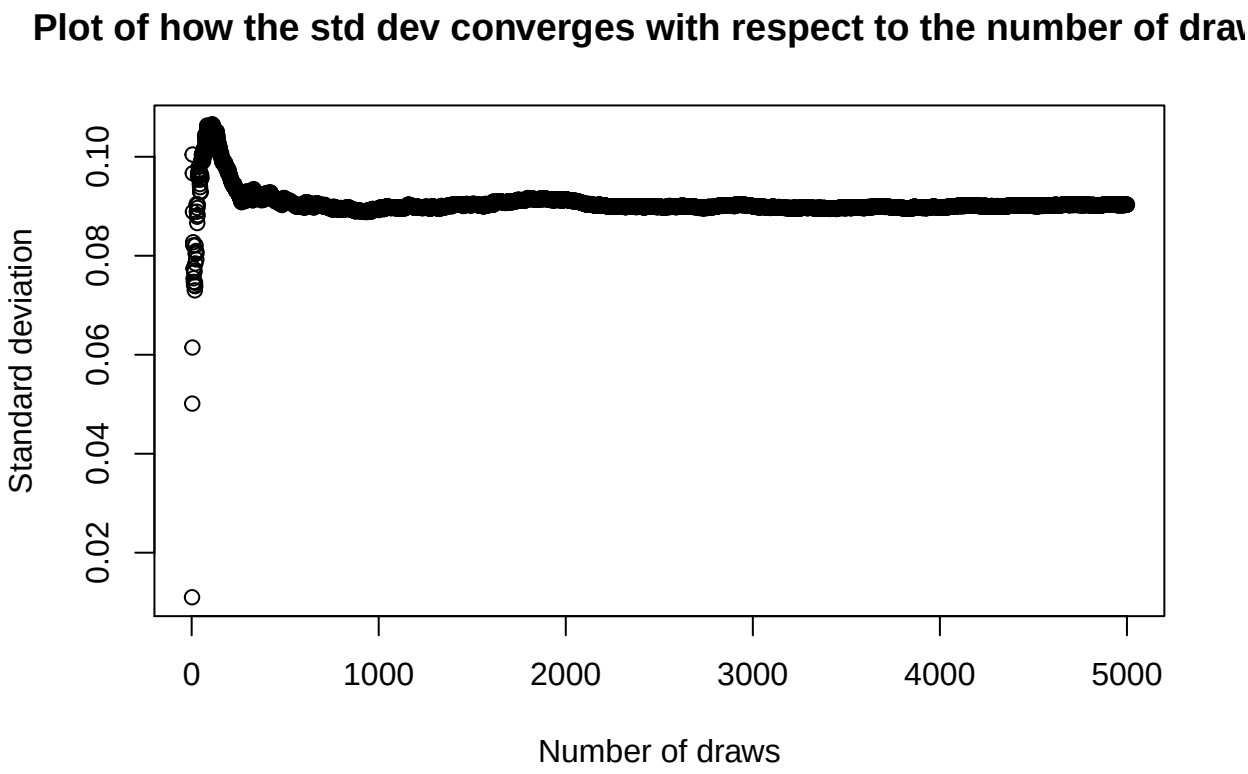



```
print(paste("The true value of the mean of the distribution is", MeanOfPosterior))
```

```
## [1] "The true value of the mean of the distribution is 0.291666666666667"
```

As seen in the plot the mean converges to its true value of approximately 0.29 as the number of draws grows large.

```
plot(nVector, stdVector,
     main="Plot of how the std dev converges with respect to the number of draws",
     xlab="Number of draws", ylab="Standard deviation")
```




```
print(paste("The true value of the standard deviation of the distribution is", StdOfPosterior))
```

```
## [1] "The true value of the standard deviation of the distribution is 0.0909059342886309"
```

As seen in the plot the standard deviation converges to its true value of approximately 0.09 as the number of draws grows large.

Task b)

```
trueProb=1-pbeta(0.3, alpha0+s, beta0+f)
set.seed(12345)
draw10000=rbeta(10000, alpha0+s, beta0+f)
probHat=sum(draw10000>0.3)/10000
print(paste("The approximated probability of theta being larger than 0.3 is calculated to be", probHat))
```

```
## [1] "The approximated probability of theta being larger than 0.3 is calculated to be 0.4392"
```

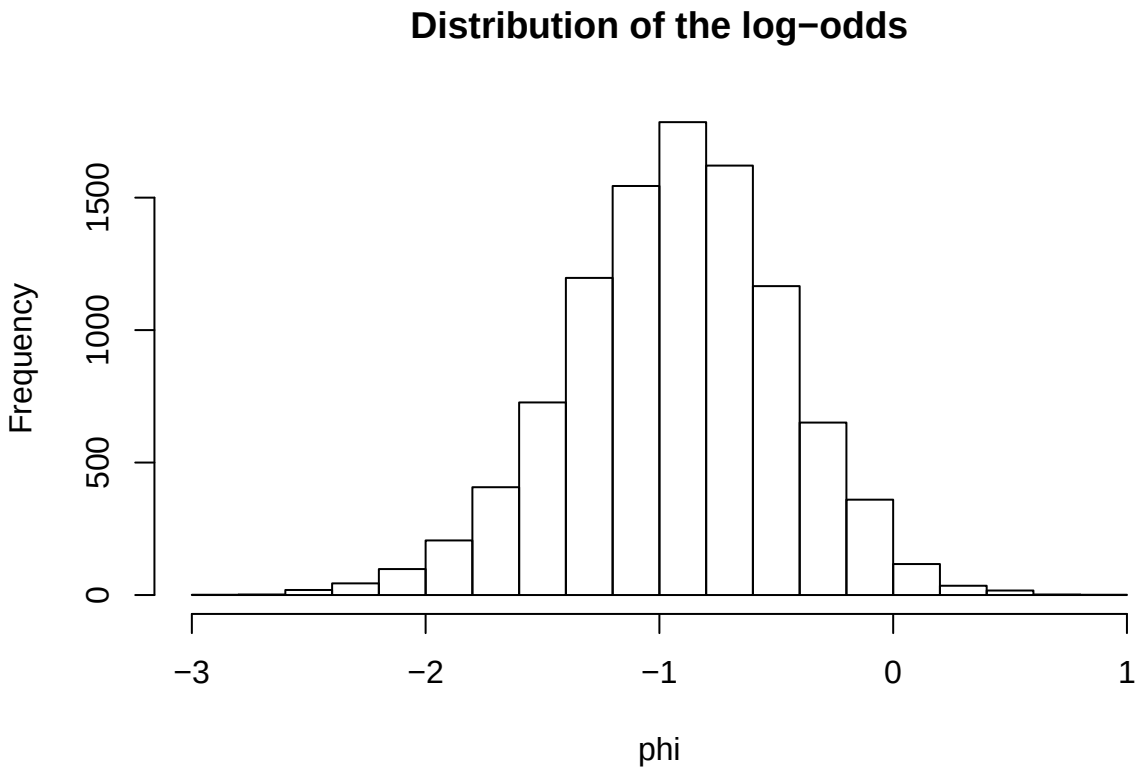
```
print(paste("The true probability of theta being larger than 0.3 for the posterior distribution is",
            trueProb))
```

```
## [1] "The true probability of theta being larger than 0.3 for the posterior distribution is 0.4399472
```

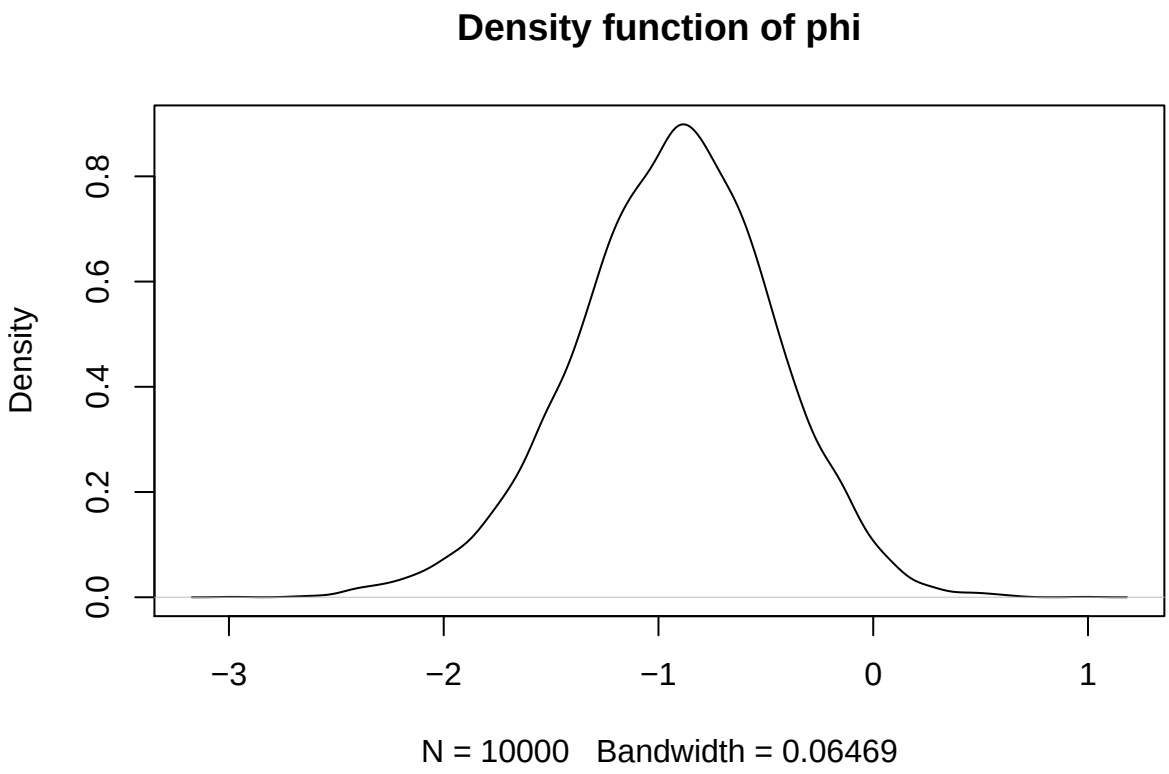
As seen in the results from both calculations the probHat is very close to the true probability from the beta distribution. As the number of draws increases the approximated probability will converge towards the true value.

Task c)

```
# Reusing the 10000 draws made earlier
phi=log(draw10000/(1-draw10000))
hist(phi, breaks=20, main="Distribution of the log-odds")
```




```
plot(density(phi), main="Density function of phi")
```



The posterior distribution of the simulated log-odds-function is presented above through a histogram and a density approximation.

Assignment 2

Task a)

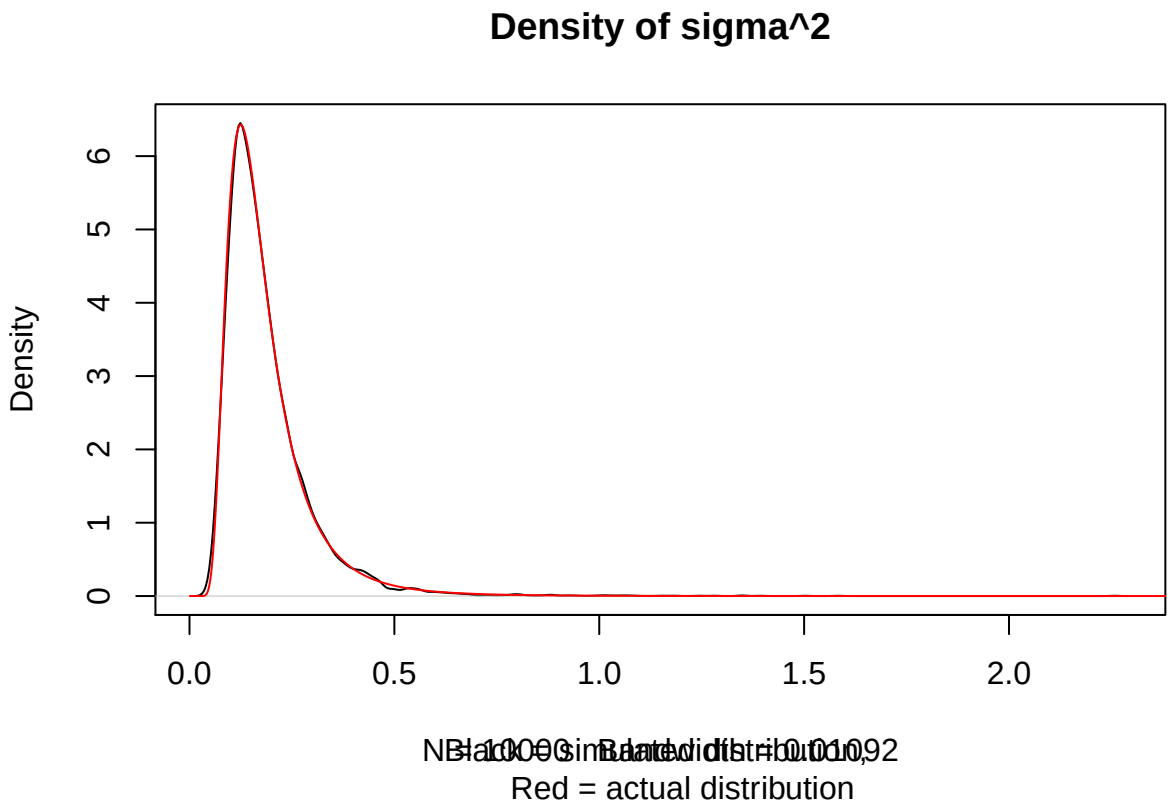
```
library(geoR)
```

```
## Warning: package 'geoR' was built under R version 3.6.3
```

```
## ----------------------------------------------------------
##  Analysis of Geostatistical Data
##  For an Introduction to geoR go to http://www.leg.ufpr.br/geoR
##  geoR version 1.8-1 (built on 2020-02-08) is now loaded
## ----------------------------------------------------------
```

```
x=c(44, 25, 45, 52, 30, 63, 19, 50, 34, 67)
n=length(x)
my=3.7

#Function for calculating thao^2
calcThao = function(data, my, n) {
  return(sum((log(data)-my)^2)/n)
}

thaosq=calcThao(x, my, n)
```

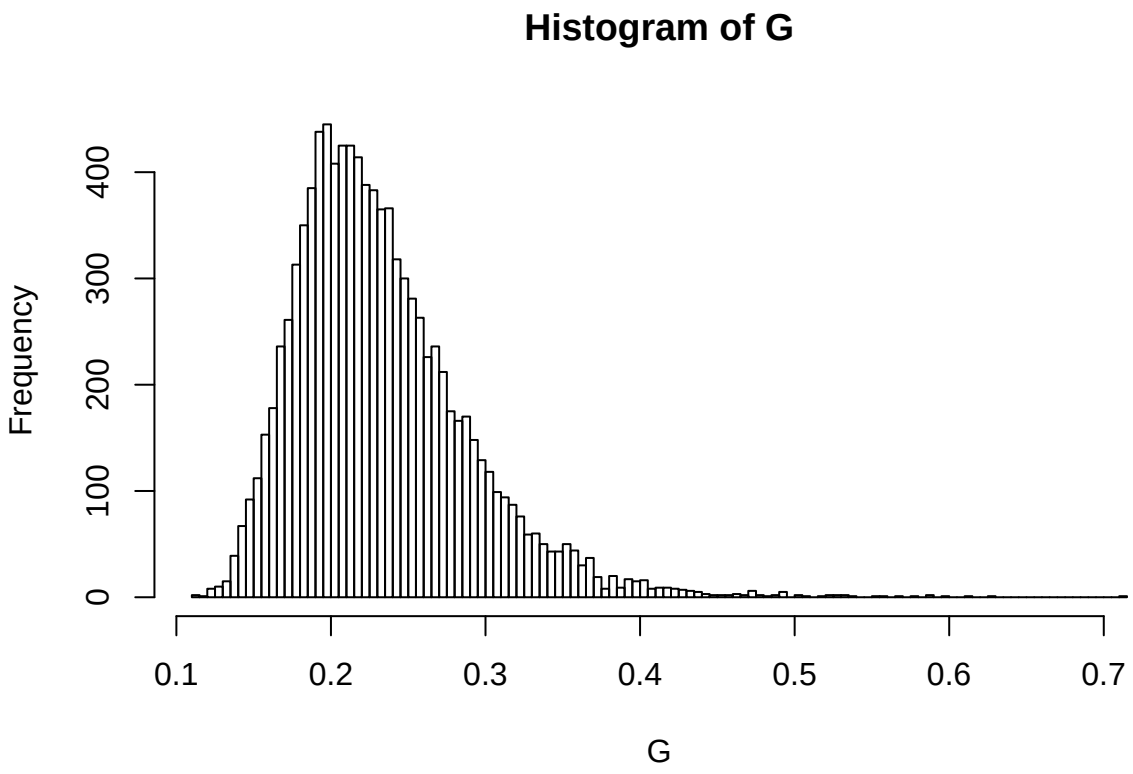
```r
set.seed(12345)
drawX=rchisq(10000, n)
sigmasq=(n)*thaosq/drawX
xvals=seq(0.001, 3, 0.001)
plot(density(sigmasq), main="Density of sigma^2", sub="Black = simulated distribution,
     Red = actual distribution")
lines(xvals,dinvchisq(xvals, n, thaosq), col="red")
```



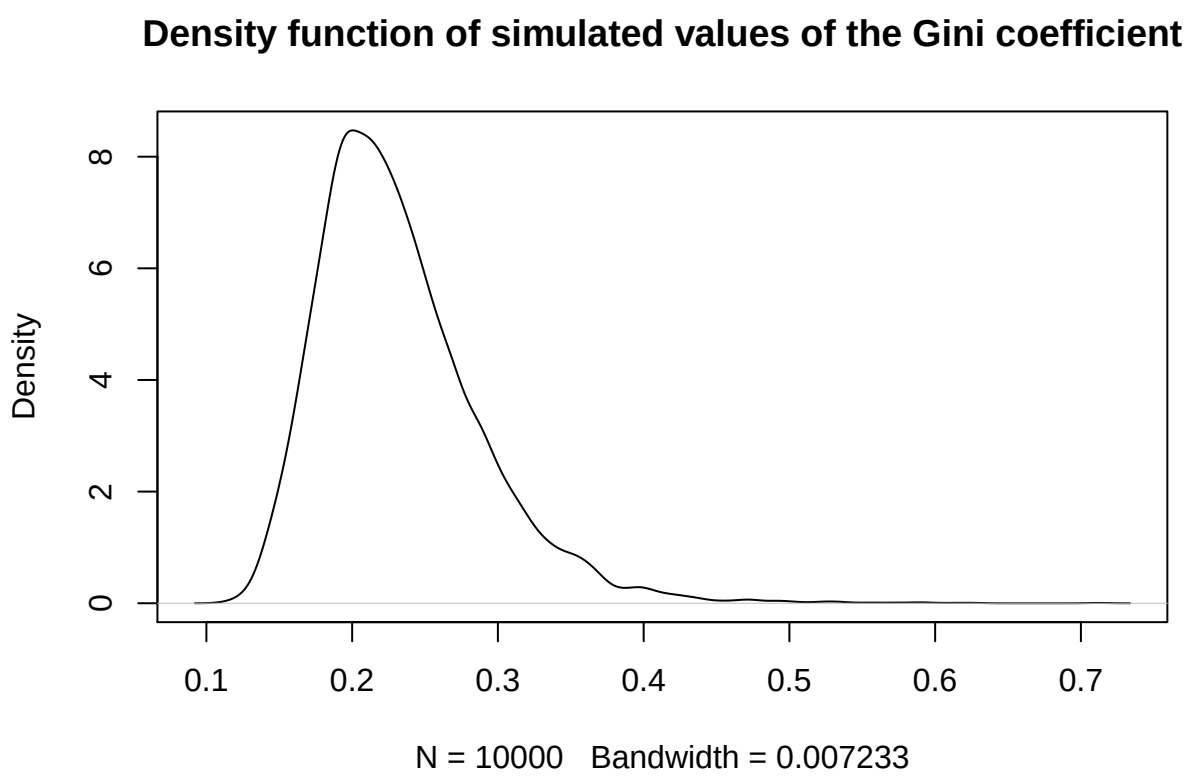
As seen in the plot above the black plot resembles the red plot well with small deviations due to the fact that it has been simulated with a large number of draws. As the number of draws increases the resembleness of the two plots will increase further.

Task b)

```r
G=2*pnorm(sqrt(sigmasq/2), mean=0, sd=1)-1
hist(G, breaks=100)
```




```
plot(density(G), main="Density function of simulated values of the Gini coefficient")
```



As seen in the plot the density function of the gini coefficient has its highest density between values between 0.15 and 0.3 approximately with a peak at around 0.2. This seems quite reasonable since the plot looks similar to the log normal distribution which it is supposed to do.

Task c)

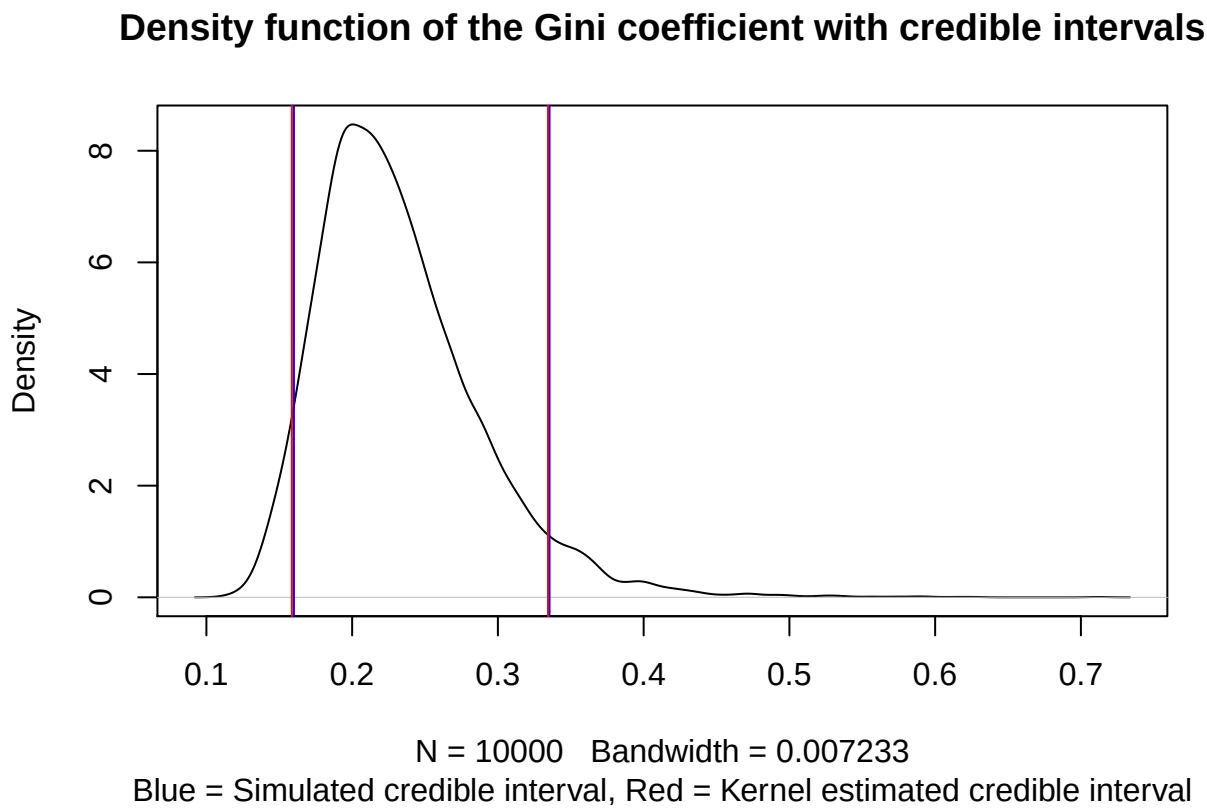
```
GSorted=sort(G)[(0.05*length(G)+1):(0.95*length(G))]
# 90 % credible interval for G through the simulated draws
G_CredInterval=c(min(GSorted),max(GSorted))
print(G_CredInterval)
```

```
## [1] 0.1600238 0.3354042
```

```
plot(density(G), main="Density function of the Gini coefficient with credible intervals")
abline(v = G_CredInterval[1], col="blue")
abline(v = G_CredInterval[2], col="blue")

GDensity=density(G)
GDensity.df=data.frame(x=GDensity$x, y=GDensity$y)
GDensity.df$y=cumsum(GDensity$y)/sum(GDensity$y)
GDensity_CredInterval_Vals=GDensity.df[GDensity.df$y>0.05 & GDensity.df$y<0.95,]
GDensity_CredInterval=c(min(GDensity_CredInterval_Vals$x), max(GDensity_CredInterval_Vals$x))
print(GDensity_CredInterval)
```

```
## [1] 0.1586145 0.3343917
```

```r
abline(v = GDensity_CredInterval[1], col="red")
abline(v = GDensity_CredInterval[2], col="red")
title(sub="Blue = Simulated credible interval, Red = Kernel estimated credible interval")
```



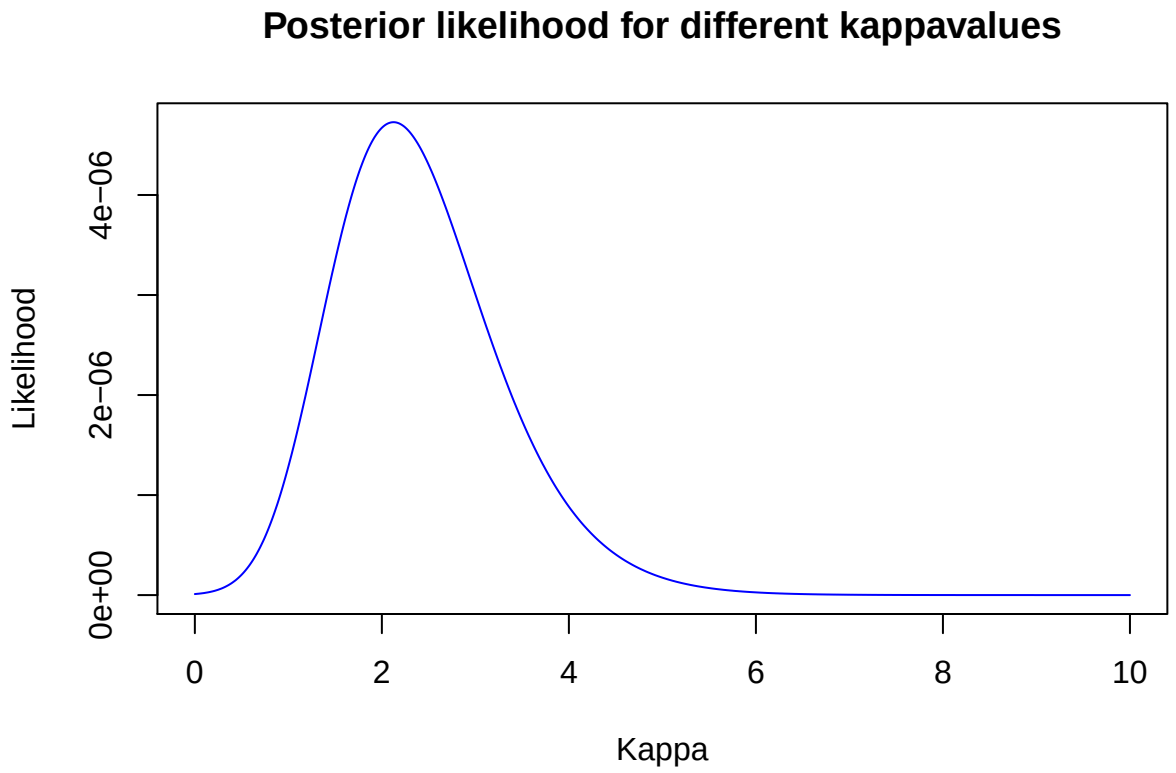
As seen in the plot the credible intervals are quite similar with small deviations. This is due to that the function density(G) creates an estimate of a function based on the values of vector G.

Assignment 3

Task a)

```r
data_radian=c(-2.44,2.14,2.54,1.83,2.02,2.33,-2.79,2.23,2.07,2.02)
my=2.39
lambda=1

# Function for computing the vonMisesDistrib for a given dataset
vonMisesDistrib = function(kappa, data, my){
  likelihood=1
  for (i in data) {
    likelihood=likelihood*exp(kappa*cos(i-my))/(2*pi*besselI(kappa, 0))
  }
  return(likelihood)
}

# Function for computing the exponential distribution
exponDistrib = function(data, lambda) {
```

```
  return(1/lambda*exp(-1/lambda*data))
}

kappa_values=seq(0,10,0.01)

# Function for computing the posterior distribution
posteriorDistrib = function(kappa, lambda, data, my) {
  likelihood=vonMisesDistrib(kappa, data, my)
  prior=exponDistrib(kappa, lambda)
  return(likelihood*prior)
}

posteriorLikelihood=posteriorDistrib(kappa_values, lambda, data_radian, my)
plot(kappa_values, posteriorLikelihood, xlab="Kappa", ylab="Likelihood",
     main="Posterior likelihood for different kappavalues", type="l", col="blue")
```



As seen in the plot the likelihood of the posterior peaks between 2 and 4 and then dies off for larger kappa-values.

```
posterior.df=data.frame(kappa=kappa_values, likelihood=posteriorLikelihood)
posteriorMode=subset(posterior.df, likelihood==max(likelihood), kappa)
print(posteriorMode$kappa)
```

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

The approximated posterior mode is found to be 2.12.

Appendix

```
### Assignment 1

## Assignment 1: Let y1,...,yn be Bernoulli distributed with parameter theta. Assume that
## you have obtained sample with s=5 successes in n=20 trials. Assume a Beta(alpha0, beta0)
## prior for theta and let alpha0=beta0=2.

## a) Draw random numbers from the posterior theta given y ~ Beta(alpha0+s, beta0+f) and
## verify graphically that the posterior mean and standard deviation converges to the true
## values as the number of random draws grows large.

set.seed(12345)
alpha0=2
beta0=2
s=5
f=15
n=20

# Function for calculating the mean of a beta-distribution
calcMeanBeta = function(alpha, beta) {
  return(alpha/(alpha+beta))
}

# Function for calculating the standard deviation of a beta-distribution
calcStdDevBeta = function(alpha, beta) {
  return(sqrt(alpha*beta/((alpha+beta)^2*(alpha+beta+1))))
}

# Function for calculating the mean squared error of drawed data
calcMSE = function(n, mean, data){
  return(sqrt(1/(n-1)*sum((data-mean)^2)))
}

# Function for drawing random values from the betadistribution
drawBetaValues = function(n, alpha, beta) {
  return(rbeta(n, alpha, beta))
}

MeanOfPosterior = calcMeanBeta(alpha0+s, beta0+f)
StdOfPosterior = calcStdDevBeta(alpha0+s, beta0+f)
nVector = seq(1, 5000, 1)
meanVector=c()
stdVector=c()
for (i in nVector) {
  set.seed(12345)
  betaValues= drawBetaValues(i, alpha0+s, beta0+f)
  meanVector=c(meanVector, mean(betaValues))
  stdVector=c(stdVector, calcMSE(i, mean(betaValues), betaValues))
}
plot(nVector, meanVector,
     main="Plot of how the mean converges with respect to number of draws",
     xlab="Number of draws", ylab="Mean")
print(paste("The true value of the mean of the distribution is", MeanOfPosterior))
```

```r
plot(nVector, stdVector,
     main="Plot of how the std dev converges with respect to the number of draws",
     xlab="Number of draws", ylab="Standard deviation")
print(paste("The true value of the standard deviation of the distribution is", StdOfPosterior))

## b) Use simulation (nDraws=10000) to compute the posterior probability Pr(theta>0.3 given y)
## and compare with with the exact value

trueProb=1-pbeta(0.3, alpha0+s, beta0+f)
set.seed(12345)
draw10000=rbeta(10000, alpha0+s, beta0+f)
probHat=sum(draw10000>0.3)/10000
print(paste("The approximated probability of theta being larger than 0.3 is calculated to be", probHat))
print(paste("The true probability of theta being larger than 0.3 for the posterior distribution is",
            trueProb))

## c) Compute the posterior distribution of the log-odds phi= log(theta/(1-theta)) by
# simulation (nDraws=10000)

# Reusing the 10000 draws made earlier
phi=log(draw10000/(1-draw10000))
hist(phi, breaks=20, main="Distribution of the log-odds")
plot(density(phi), main="Density function of phi")

### Assignment 2

## Assignment 2: Assume that you have asked 10 randomly selected persons about their monthly
## income(inthousandsSwedishKrona)andobtainedthefollowingtenobservations:
## 44, 25, 45, 52, 30, 63, 19, 50, 34 and 67. A common model for non-negative continuous
## variables is the log-normal distribution. The log-normal distribution log(N(my, sigma^2))
## has density function ... for y > 0, my > 0 and sigma > 0. The log-normal distribution is
## related to the normal distribution as follows: if y ~ log N(my, sigma^2) then
## log y ~ N(my, sigma^2). Let y1,...,yn given my and simga^2 ~ log N(my, sigma^2), where my=3.7
## is assumed to be known but sigma^2 is unknown with noninformative prior p(sigma^2) is
## proportional to 1/sigma^2. The posterior for sigma^2 is the Inv - chitwo distribution with
## X(n, thao^2) distribution, where thao^2=sum((log(yi)-my)^2)/n

library(geoR)
x=c(44, 25, 45, 52, 30, 63, 19, 50, 34, 67)
n=length(x)
my=3.7

#Function for calculating thao^2
calcThao = function(data, my, n) {
  return(sum((log(data)-my)^2)/n)
}

thaosq=calcThao(x, my, n)
set.seed(12345)
drawX=rchisq(10000, n)
sigmasq=(n)*thaosq/drawX
xvals=seq(0.001, 3, 0.001)
plot(density(sigmasq), main="Density of sigma^2", sub="Black = simulated distribution,
```

```
     Red = actual distribution")
lines(xvals,dinvchisq(xvals, n, thaosq), col="red")

## b) The most common measure of income inequality is the Gini coefficient, G, where
## 0<=G<=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*CDF-normal(sigma/sqrt(2))-1 when income follow a log N(my, sigma^2) distribution.
## Use the posterior draws in a) to compute the posterior distribution of the Gini coefficient
## G for the current data set.

G=2*pnorm(sqrt(sigmasq/2), mean=0, sd=1)-1
hist(G, breaks=100)
plot(density(G), main="Density function of simulated values of the Gini coefficient")

## c) Use the posterior draws from b) to compute a 90% equal tail credible interval for G.
## A 90% equal tail interval (a,b) cuts off 5% percent of the posterior probability mass to
## the left of a, and 5% to the right of b. Also, do a kernel density estimate of the posterior
## of G using the density function in R with defaultsettings,  and use that kernel density
## estimate to compute a 90% Highest Posterior Density interval for G. Compare the  two intervals.

GSorted=sort(G)[(0.05*length(G)+1):(0.95*length(G))]
# 90 % credible interval for G through the simulated draws
G_CredInterval=c(min(GSorted),max(GSorted))
print(G_CredInterval)
plot(density(G), main="Density function of the Gini coefficient with credible intervals")
abline(v = G_CredInterval[1], col="blue")
abline(v = G_CredInterval[2], col="blue")

GDensity=density(G)
GDensity.df=data.frame(x=GDensity$x, y=GDensity$y)
GDensity.df$y=cumsum(GDensity$y)/sum(GDensity$y)
GDensity_CredInterval_Vals=GDensity.df[GDensity.df$y>0.05 & GDensity.df$y<0.95,]
GDensity_CredInterval=c(min(GDensity_CredInterval_Vals$x), max(GDensity_CredInterval_Vals$x))
print(GDensity_CredInterval)
abline(v = GDensity_CredInterval[1], col="red")
abline(v = GDensity_CredInterval[2], col="red")
title(sub="Blue = Simulated credible interval, Red = Kernel estimated credible interval")

### Assignment 3

## 3.a) 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:
## (40, 303, 326, 285, 296, 314, 20, 308, 299, 296) where North is located at zero degrees
## (see Figure 1 on the next page, where the angles are measured clockwise). To fit with
## Wikipedias description of probability distributions for circular data we convert the
## data into radians -pi<=y<=pi. The 10 observations in radians are
## (-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 given my,k) = exp(k*cos(y-u))/(2*pi*I0(k)), -pi<=y<=pi, where I0(k) is the modified
## Bessel function of the first kind of order zero (see ?besselI in R). The parameter my
```

```r
## (-pi<=my<=pi) is the mean direction and k>0 is called the concentration parameter. Large
## k gives a small variance around my, and vice versa. Assume that my is known to be 2.39.
## Let K ~ Exponential(Lambda=1) a priori, where lambda is the rate parameter of the
## exponential distribution (so that the mean is 1/lambda).

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

data_radian=c(-2.44,2.14,2.54,1.83,2.02,2.33,-2.79,2.23,2.07,2.02)
my=2.39
lambda=1

# Function for computing the vonMisesDistrib for a given dataset
vonMisesDistrib = function(kappa, data, my){
  likelihood=1
  for (i in data) {
    likelihood=likelihood*exp(kappa*cos(i-my))/(2*pi*besselI(kappa, 0))
  }
  return(likelihood)
}

# Function for computing the exponential distribution
exponDistrib = function(data, lambda) {
  return(1/lambda*exp(-1/lambda*data))
}

kappa_values=seq(0,10,0.01)

# Function for computing the posterior distribution
posteriorDistrib = function(kappa, lambda, data, my) {
  likelihood=vonMisesDistrib(kappa, data, my)
  prior=exponDistrib(kappa, lambda)
  return(likelihood*prior)
}

posteriorLikelihood=posteriorDistrib(kappa_values, lambda, data_radian, my)
plot(kappa_values, posteriorLikelihood, xlab="Kappa", ylab="Likelihood",
     main="Posterior likelihood for different kappavalues", type="l", col="blue")

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

# Puts likelihood values with corresponding kappa-values to be able to retrieve the
# kappa-value corresponding to the highest likelihood (mode)
posterior.df=data.frame(kappa=kappa_values, likelihood=posteriorLikelihood)
posteriorMode=subset(posterior.df, likelihood==max(likelihood), kappa)
print(posteriorMode$kappa)
```