marho558
erisn497

Assignment 1

a)

True mean for beta distribution: **0.3488**

True standard deviation for beta distribution: **0.0511**

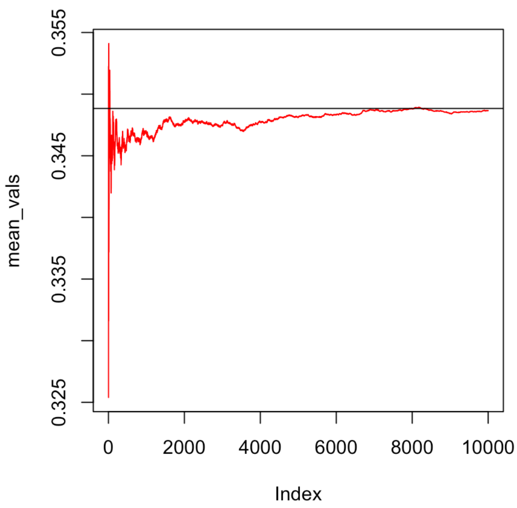

Figure 1.1: Mean per draw with true mean as line.

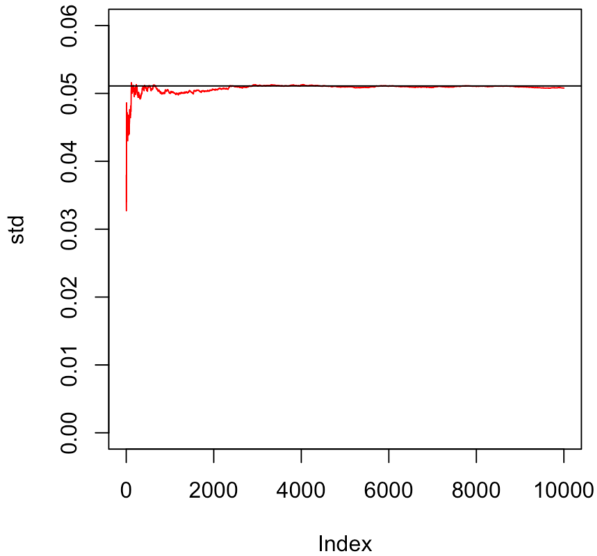

Figure 1.2: Standard deviation per draw with true standard deviation as line.

As we can see in figure 1.1 and figure 1.2, the mean and standard deviation converges towards the true values for large n:s.

Code:
```
#Assignment 1

#Let y1, ..., yn|ϑ ~ Bern(ϑ), and assume that you have obtained a sample
#with s = 22 successes in n = 70 trials. Assume a Beta(α0, β0) prior for ϑ
and let α0 = β0 = 8.


###Starting assignment###


###Variables###
s=22
n=70
a0=8
b0=8
###############

#Assignment 1
meanBeta <- function(a,b){
  return(a/(a+b))
```

```
}

varBeta <-function(a,b) {
  return(a*b/(((a+b)**2)*(a+b+1)))
}

BetaPlot <- function(a,b){
  xGrid <- seq(0.001, 0.999, by=0.001)
  prior = dbeta(xGrid, a, b)
  maxDensity <- max(prior) # Use to make the y-axis high enough
  plot(xGrid, prior, type = 'l', lwd = 3, col = "blue", xlim <- c(0,1), yl
im <- c(0, maxDensity), xlab = "theta",
       ylab = 'Density', main = 'Beta(a,b) density')
}

#a) Draw 10000 random values (nDraws = 10000) from the posterior ϑ|y ∼ Bet
a(α0+ s, β0 + f), where y = (y1, . . . , yn),
#and verify graphically that the posterior mean E [ϑ|y] and standard devia
tion SD [ϑ|y] converges to the true values
#as the number of random draws grows large. [Hint: use rbeta() to draw ran
dom values and make graphs of the sample means
#and standard deviations of ϑ as a function of the accumulating number of
drawn values].


BetaPlot(a0,b0) #Pdf of prior
```

```
BetaPlot(a0+s,b0+n-s) #Pdf of posterior
```

```
meanPost = meanBeta(a0+s,b0+n-s) #Calculate the actual mean of the posteri
or
meanPost #actual is 0.3488

## [1] 0.3488372

varPost = varBeta(a0+s,b0+n-s) #Calculate the actual variance of the poste
rior
varPost #actual variance is 0.00261

## [1] 0.002610917

stdPost = sqrt(varPost) #Standard deviation of actual
stdPost #actual standard deviation is 0.0511

## [1] 0.05109714

Ndraws = 10000 #Number of samples to draw
samples = rbeta(Ndraws, shape1=a0+s, shape2=b0+n-s) #Draws 10000 random sa
mples from the posterior

mean_vals = rep(0,10000) #List to store mean
std = rep(0,10000) #List to store standard deviation
```

```
for (i in 1:10000) {
  mean = mean(samples[1:i])
  mean_vals[i] = mean
  std[i] = sd(samples[1:i])
}

plot(mean_vals, type ="l", col="red") #Plotting accum. mean vals based on
increased Ndraws
abline(meanPost,0) #plotting tangent line of actual mean
```

```
plot(std, type="l", col="red", ylim=c(0,0.06)) #Plotting accum. standard d
eviation based on increased Ndraws
abline(stdPost,0) #plotting tangent line of actual std
```

```
#As seen in the plotted graphs, the mean and std dev converges towards the
true values for big n:s
```

b)

True probability: **0.83**

Calculated probability: **0.8286**

We can see that the calculated and true probabilities are roughly the same, meaning that large data sets give good approximations.

Code:

```
#b) Draw 10000 random values from the posterior to compute the posterior p
rob- ability
#Pr(θ > 0.3|y) and compare with the exact value from the Beta posterior. [
Hint: use pbeta()].


threshold = 0.3
true_probability = 1 - pbeta(threshold,shape1=a0+s, shape2=b0+n-s) #Calcul
ating the true probability of the posterior
true_probability #True value is 0.83

## [1] 0.8285936

samples = rbeta(Ndraws, shape1=a0+s, shape2=b0+n-s) #Drawing 10000 samples
from the beta distribution
p_greater = mean(samples > threshold) #Calculating the mean of all random
samples greater than 0.3
p_greater #Calculated value is 0.8286

## [1] 0.8349
```

```
#We can see that the probability is ~83% for both, indicating we get good
#approximations for large data sets of random samples
```

c)

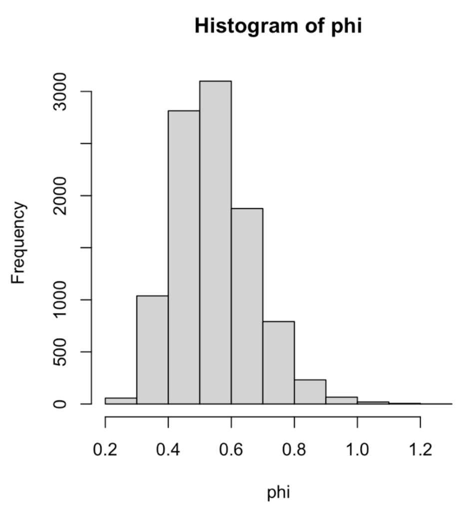

Figure 1.3: Histogram of phi values

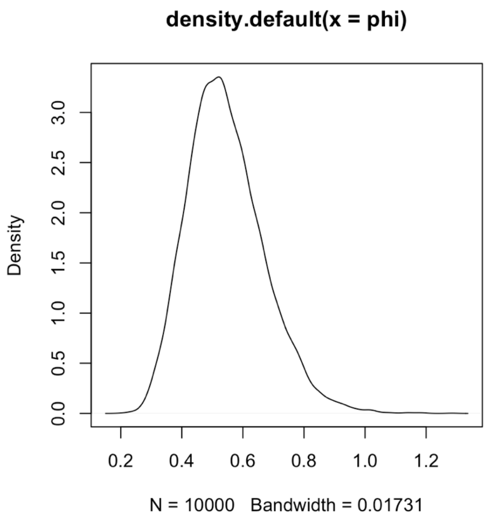

Figure 1.4: Plot of density for phi values

By comparing the graphs, we can see that the histogram and the density plot follows the same shape which was expected.

Code:

```
#c) Draw 10000 random values from the posterior of the odds φ = ϑ by using 
1−ϑ
#the previous random draws from the Beta posterior for ϑ and plot the post
erior distribution of φ.
#[Hint: hist() and density() can be utilized].


samples = rbeta(Ndraws, shape1=a0+s, shape2=b0+n-s) #Sample 10000 random s
amples
phi = samples/(1-samples) #Calculate phi
hist(phi) #Plot histogram
```

```
plot(density(phi)) #Plot density function
```

Assignment 2

a)

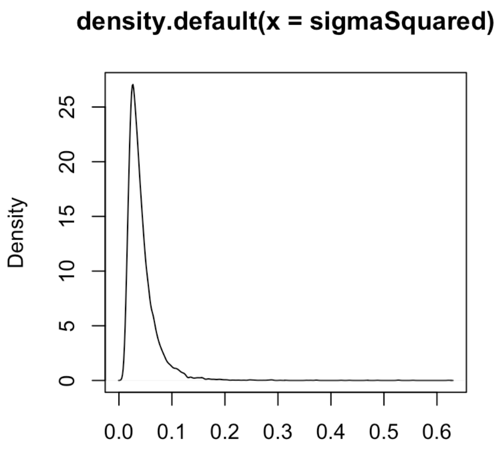

Figure 2.1: Density plot of the posterior for sigma squared.

The posterior of the random draws of sigma squared is heavily cantered around 0.05 which is indicated in the density plot in figure 2.1.

Code:

```
###Starting assignment###

###Variables###
income = c(33,24,48,32,55,74,23,17)
my = 3.6
Ndraw = 10000
###############

#Calculating tao
taoFunc = function(y) {
  return((sum(log(y)-my)**2)/length(y))
}

#Function calculate sigma squared
sigmaSquaredFunc = function(chi_vals,tao,n) {
 return(((n)*tao)/chi_vals)
}


#a) Draw 10000 random values from the posterior of σ2 by assuming μ = 3.6
and plot the posterior distribution.


tao = taoFunc(income) #Calculating tao value
n = length(income) #Defining number of data points
```

Commented [ES1]: Ska vara n-1 här?

```
set.seed(12345)
chi_vals = rchisq(Ndraw,n) #Drawing 10000 random samples from chi-squared distribution
sigmaSquared = sigmaSquaredFunc(chi_vals, tao, n) #Calculating apporximation for sigmaSquared

#Plotting the posterior distribution of simgasquared
plot(density(sigmaSquared))
```

Commented [ES2]: N-1 här också?

b)

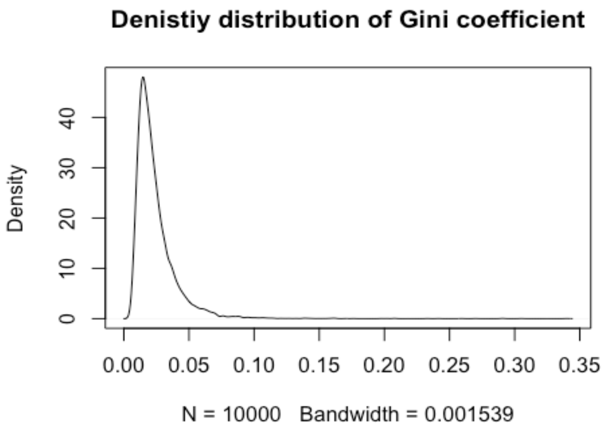

Figure 2.2: Density plot of gini coefficient.

The density plot of the gini coefficient is weighted towards values in between 0.00 and 0.05, indicating that the random draws of income are almost completely equal. This was expected as figure 2.1 indicated that most incomes landed in the same interval.

Commented [ES3]: Vi menar att std är liten och därför är det equal?

Code:

```
#b) The most common measure of income inequality is the Gini coe�cient, G,
#where 0 ≤ G ≤ 1. G = 0 means a completely equal income distribution, whereas G = 1 means complete
#income inequality (see e.g. Wikipedia for more information about the Gini coe�cient).
#It can be shown that G = 2Φ �σ/√2�−1 when incomes follow a log N (μ, σ2) 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 coe�cient G
#for the current data set.
```

```
G = 2*pnorm(sigmaSquared/sqrt(2))-1 #Calculating gini values
plot(density(G), main="Denistiy distribution of Gini coefficient") #Densit
y distribution of Gini values
```

c)

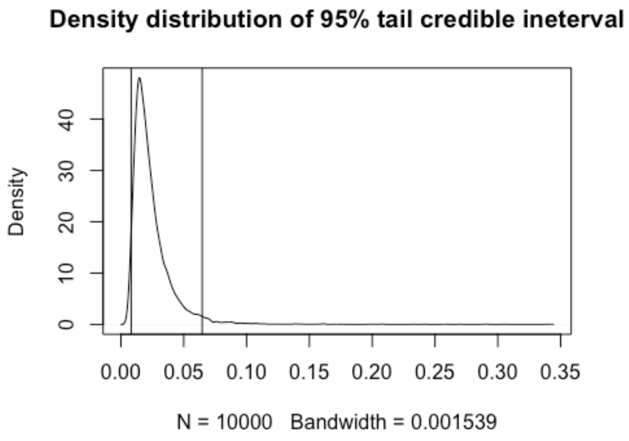

Figure 2.3: Equal tail distribution indicated with vertical line.

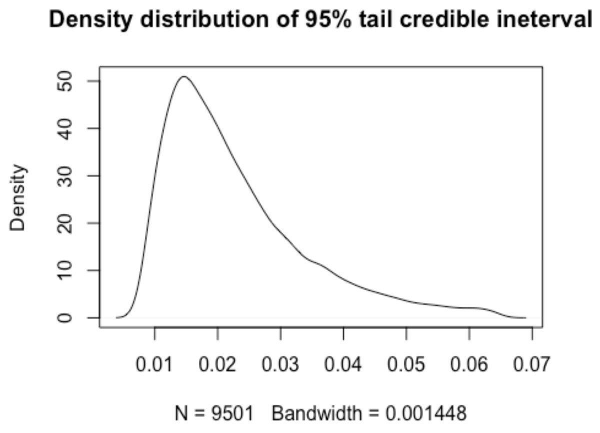

Figure 2.4: Density plot with removed tails.

The plot in figure 2.3 indicates that by using an equal tail interval, we risk removing some of the parts where we have the most density (since the data are heavily weighted on the left side).

Code:

```
#c) Use the posterior draws from b) to compute a 95% equal tail credible i
nterval for G.
#A 95% equal tail credible interval (a,b) cuts o� 2.5% percent of the post
erior probability mass to the
#left of a, and 2.5% to the right of b.


sorted_G = sort(G) #Sorting values in G
min = length(G)*0.025 #Calculating lower bound
max = length(G)*0.975 #Calculating upper bound
filt_G = sorted_G[min:max] #Filtering out tails
plot(density(G), main="Density distribution of 95% tail credible ineterval
")
abline(v=sorted_G[min]) #Lower end of credible interval
abline(v=sorted_G[max]) #Upper end of credible interval

plot(density(filt_G), main="Density distribution of 95% tail credible inet
erval") #Plot with only values in credible interval
```

d)

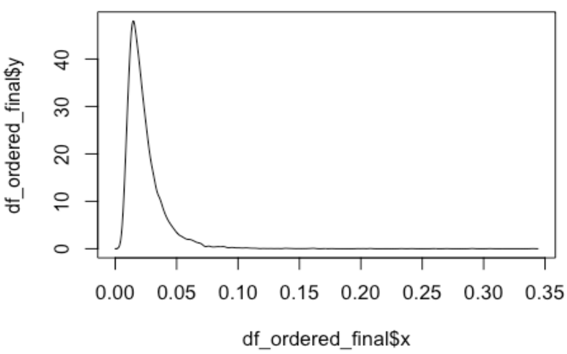

Figure 2.5: HPDI plot of gini data

Since we remove the 5% with the lowest density, we retain the shape which was expected. The difference between figure 2.3 and figure 2.5 clearly illustrates the effects of equal tail interval compared to HPDI.

Commented [ES4]: Vad menas

Commented [ES5R4]: Varför blir resultatet annorlunda?

Code:

```
#d) Use the posterior draws from b) to compute a 95% Highest Posterior Den
sity Interval (HPDI) for G.
#Compare the two intervals in (c) and (d). [Hint: do a kernel density esti
mate of the posterior of G using the
#density function in R with default settings, and use that kernel density
estimate to compute the HPDI.
#Note that you need to order/sort the estimated density values to obtain t
he HPDI.].


density_vals = density(G) #Using density function to load info for Gini co
efficients density
df = data.frame(x=density_vals$x, y=density_vals$y) #Storing coordinates a
nd indexes in a data frame
df_ordered=df[order(df$y),] #Sorting the data frame in ascending order
df_ordered$index=seq(1,length(density_vals$x),1) #Creating index to keep t
rack of x & y coordinates from G:s density
ind_to_remove = round(length(df$x)*0.05) #Calculating how many indexes to
remove to get the 95% HPDI


library(dplyr)

df_ordered_filt = df_ordered %>% slice(-c(1:ind_to_remove)) #Removing lowe
st y values
df_ordered_final = df_ordered_filt[order(df_ordered_filt$x),] #Filtering i
n order of x values
plot(df_ordered_final$x, df_ordered_final$y, type="l") #Plotting result
```

Assignment 3

a)

F(kappa) was derived by starting in p(kappa|y,my), which is proportional to p(y|kappa,my)*p(kappa). This can in turn be calculated with the given information in the assignment.

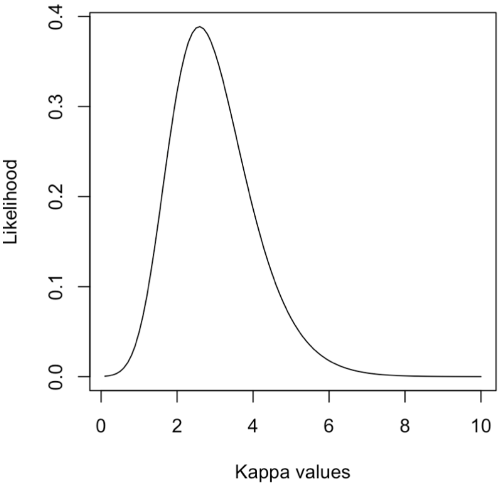

Figure 3.1: Likelihood function for posterior of kappa plotted for kappa values.

Code:

```
#Assignment 3

#This exercise is concerned with directional data. The point is to show yo
u that the posterior distribution
#for somewhat weird models can be obtained by plotting it over a grid of v
alues.
#The data points are observed wind directions at a given location on ten d
iﬀerent days.
#The data are recorded in degrees: (20, 314, 285, 40, 308, 314, 299, 296,
303, 326),
#where North is located at zero degrees (see Figure 1 on the next page, wh
ere the angles are measured clockwise).
#To ﬁt with Wikipedia's description of probability distributions for circu
lar data we convert
#the data into radians -π ≤ y ≤ π . The 10 observations in
#radians are (-2.79, 2.33, 1.83, -2.44, 2.23, 2.33, 2.07, 2.02, 2.14, 2.54
).
#Assume that these data points conditional on (μ,κ) are
#independent observations from the following von Mises distribution:

#where I0(κ) is the modiﬁed Bessel function of the ﬁrst kind of order zero
[see ?besselI in R].
#The parameter μ (-π ≤ μ ≤ π) is the mean direction and κ > 0 is called th
e concentration parameter.
#Large κ gives a small variance around μ, and vice versa. Assume that μ is
```

```
known to be 2.4. Let κ ~ Exponential(λ = 0.5)
#a priori, where λ is the rate parameter of the exponential distribution (
so that the mean is 1/λ).


###Starting assignment###

#a) Derive the expression for what the posterior p(κ|y, μ) is proportional
to. Hence, derive the function f (κ) such
#that p(κ|y, μ) ∝ f (κ). Then, plot the posterior distribution of κ for th
e wind direction data over a fine grid of κ
#values. [Hint: you need to normalize the posterior distribution of κ so t
hat it integrates to one.]


#Derive the posterior p(k|y,my) is proportional to p(y|k,my)*p(k|my) is pr
oportional to p(y|k,my)*p(k)
#p(k|my) is proportional to p(k) since k is independent of my

#Data and know parameters
degrees = c(-2.79,2.33,1.83,-2.44,2.23,2.33,2.07,2.02,2.14,2.54) #Directio
ns in radians
my = 2.4 #Given my value
lambda = 0.5 #Given lambda value

#Function for von Mises distribution
vonMises = function(kappa, y, my) {
  likelihood = 1
  for (data in y) {
  likelihood = likelihood*exp(kappa*cos(data-my))/(2*pi*besselI(kappa,0))
  }
  return(likelihood)
}

#Function for exponential distribution
exponential = function(tetha, data) {
  return(tetha*exp(-tetha*data))
}

kappa = seq(0.1,10,0.1) #Kappa values to be used in distribution
VMProbs = rep(0, length(kappa)) #List to store probs from von Mis.
expProbs = rep(0, length(kappa)) #List to store probs from exp. distr

#Loop to generate all probs from given distribution
for(i in 1:length(kappa)) {
VMProbs[i] = vonMises(kappa[i],degrees,my)
expProbs[i] = exponential(lambda,kappa[i])
}

posterior = expProbs*VMProbs #Calculating posterior
postIntegral = sum(posterior*0.1) #Calculating the integral for the poster
ior
normPosterior = posterior/postIntegral #Normalizing the posterior to integ
rate to 1
```

```
plot(kappa,normPosterior, type="l", xlab="Kappa values", ylab="Likelihood"
) #Plotting the normalized posterior
```

b)

In figure 3.1 we can see that the approximate posterior mode of kappa is when kappa is between 2 & 4. A good graphical estimation of the mode of kappa would be kappa ~2.8.

Appendix

Code assignment 1:

```
#Assignment 1

#Let y1, ..., yn|θ ~ Bern(θ), and assume that you have obtained a sample
#with s = 22 successes in n = 70 trials. Assume a Beta(α0, β0) prior for θ
and let α0 = β0 = 8.


###Starting assignment###


###Variables###
s=22
n=70
a0=8
b0=8
###############

#Assignment 1
meanBeta <- function(a,b){
  return(a/(a+b))
}

varBeta <-function(a,b) {
  return(a*b/(((a+b)**2)*(a+b+1)))
}

BetaPlot <- function(a,b){
  xGrid <- seq(0.001, 0.999, by=0.001)
  prior = dbeta(xGrid, a, b)
  maxDensity <- max(prior) # Use to make the y-axis high enough
  plot(xGrid, prior, type = 'l', lwd = 3, col = "blue", xlim <- c(0,1), yl
im <- c(0, maxDensity), xlab = "theta",
       ylab = 'Density', main = 'Beta(a,b) density')
}

#a) Draw 10000 random values (nDraws = 10000) from the posterior θ|y ~ Bet
a(α0+ s, β0 + f), where y = (y1, . . . , yn),
```

```
#and verify graphically that the posterior mean E [ϑ|y] and standard devia
tion SD [ϑ|y] converges to the true values
#as the number of random draws grows large. [Hint: use rbeta() to draw ran
dom values and make graphs of the sample means
#and standard deviations of ϑ as a function of the accumulating number of
drawn values].


BetaPlot(a0,b0) #Pdf of prior
```

```
BetaPlot(a0+s,b0+n-s) #Pdf of posterior
```

```
meanPost = meanBeta(a0+s,b0+n-s) #Calculate the actual mean of the posteri
or
meanPost #actual is 0.3488

## [1] 0.3488372

varPost = varBeta(a0+s,b0+n-s) #Calculate the actual variance of the poste
rior
varPost #actual variance is 0.00261

## [1] 0.002610917

stdPost = sqrt(varPost) #Standard deviation of actual
stdPost #actual standard deviation is 0.0511

## [1] 0.05109714

Ndraws = 10000 #Number of samples to draw
samples = rbeta(Ndraws, shape1=a0+s, shape2=b0+n-s) #Draws 10000 random sa
mples from the posterior

mean_vals = rep(0,10000) #List to store mean
std = rep(0,10000) #List to store standard deviation

for (i in 1:10000) {
  mean = mean(samples[1:i])
  mean_vals[i] = mean
  std[i] = sd(samples[1:i])
}

plot(mean_vals, type ="l", col="red") #Plotting accum. mean vals based on
increased Ndraws
abline(meanPost,0) #plotting tangent line of actual mean
```

```
plot(std, type="l", col="red", ylim=c(0,0.06)) #Plotting accum. standard d
eviation based on increased Ndraws
abline(stdPost,0) #plotting tangent line of actual std
```

```
#As seen in the plotted graphs, the mean and std dev converges towards the
true values for big n:s

#b) Draw 10000 random values from the posterior to compute the posterior p
rob- ability
#Pr(ϑ > 0.3|y) and compare with the exact value from the Beta posterior. [
Hint: use pbeta()].


threshold = 0.3
true_probability = 1 - pbeta(threshold,shape1=a0+s, shape2=b0+n-s) #Calcul
ating the true probability of the posterior
true_probability #True value is 0.83

## [1] 0.8285936

samples = rbeta(Ndraws, shape1=a0+s, shape2=b0+n-s) #Drawing 10000 samples
from the beta distribution
p_greater = mean(samples > threshold) #Calculating the mean of all random
samples greater than 0.3
p_greater #Calculated value is 0.8286

## [1] 0.8349

#We can see that the probability is ~83% for both, indicating we get good
#approximations for large data sets of random samples

#c) Draw 10000 random values from the posterior of the odds φ = ϑ by using
1−ϑ
#the previous random draws from the Beta posterior for ϑ and plot the post
erior distribution of φ.
#[Hint: hist() and density() can be utilized].


samples = rbeta(Ndraws, shape1=a0+s, shape2=b0+n-s) #Sample 10000 random s
amples
phi = samples/(1-samples) #Calculate phi
hist(phi) #Plot histogram
```

```
plot(density(phi)) #Plot density function
```

Code assignment 2:

```
#Assignment 2

#Assume that you have asked 8 randomly selected persons about their monthl
y income
#(in thousands Swedish Krona) and obtained the following eight observation
```

```
s: 33, 24, 48, 32, 55, 74, 23, and 17.
#A common model for non-negative continuous variables is the log-normal di
stribution.
#The log-normal distribution logN(μ,σ2) has density function

#where y > 0, -∞ < μ < ∞ and σ2 > 0. The log-normal distribution is relate
d
#to the normal distribution as follows: if y ∼ logN(μ,σ2) then logy ∼ N(μ,
σ2).
#2iid 2 2 Let y1,...,yn|μ,σ ∼ logN(μ,σ ), where μ = 3.6 is assumed to be k
nown but σ
#is unknown with non-informative prior p(σ2) ∝ 1/σ2. The posterior for σ2
is the Inv−χ2(n,τ2) distribution, where

###Starting assignment###

###Variables###
income = c(33,24,48,32,55,74,23,17)
my = 3.6
Ndraw = 10000
###############

#Calculating tao
taoFunc = function(y) {
  return((sum(log(y)-my)**2)/length(y))
}

#Function calculate sigma squared
sigmaSquaredFunc = function(chi_vals,tao,n) {
 return(((n)*tao)/chi_vals)
}


#a) Draw 10000 random values from the posterior of σ2 by assuming μ = 3.6
and plot the posterior distribution.


tao = taoFunc(income) #Calculating tao value
n = length(income) #Defining number of data points
set.seed(12345)
chi_vals = rchisq(Ndraw,n) #Drawing 10000 random samples from chi-squared
distribution
sigmaSquared = sigmaSquaredFunc(chi_vals, tao, n) #Calculating apporximati
on for sigmaSquared

#Plotting the posterior distribution of simgasquared
plot(density(sigmaSquared))
```

```
#b) The most common measure of income inequality is the Gini coe�cient, G,
#where 0 ≤ G ≤ 1. G = 0 means a completely equal income distribution, wher
eas G = 1 means complete
#income inequality (see e.g. Wikipedia for more information about the Gini
```

```
coe￿cient).
#It can be shown that G = 2Φ ￿σ/√2￿−1 when incomes follow a log N (μ, σ2)
distribution.
#Φ(z) is the cumulative distribution function (CDF) for the standard norma
l distribution with mean zero
#and unit variance. Use the posterior draws in a) to compute the posterior
distribution of the Gini coe￿cient G
#for the current data set.


G = 2*pnorm(sigmaSquared/sqrt(2))-1 #Calculating gini values
plot(density(G), main="Denistiy distribution of Gini coefficient") #Densit
y distribution of Gini values
```

```
#Centered around low values (~0.03) which shows equal distribution?

#c) Use the posterior draws from b) to compute a 95% equal tail credible i
nterval for G.
#A 95% equal tail credible interval (a,b) cuts o￿ 2.5% percent of the post
erior probability mass to the
#left of a, and 2.5% to the right of b.


sorted_G = sort(G) #Sorting values in G
min = length(G)*0.025 #Calculating lower bound
max = length(G)*0.975 #Calculating upper bound
filt_G = sorted_G[min:max] #Filtering out tails
plot(density(G), main="Density distribution of 95% tail credible ineterval
")
abline(v=sorted_G[min]) #Lower end of credible interval
abline(v=sorted_G[max]) #Upper end of credible interval

plot(density(filt_G), main="Density distribution of 95% tail credible inet
erval") #Plot with only values in credible interval
```

```
#d) Use the posterior draws from b) to compute a 95% Highest Posterior Den
sity Interval (HPDI) for G.
#Compare the two intervals in (c) and (d). [Hint: do a kernel density esti
mate of the posterior of G using the
#density function in R with default settings, and use that kernel density
estimate to compute the HPDI.
#Note that you need to order/sort the estimated density values to obtain t
he HPDI.].


density_vals = density(G) #Using density function to load info for Gini co
efficients density
df = data.frame(x=density_vals$x, y=density_vals$y) #Storing coordinates a
nd indexes in a data frame
df_ordered=df[order(df$y),] #Sorting the data frame in ascending order
df_ordered$index=seq(1,length(density_vals$x),1) #Creating index to keep t
```

```
rack of x & y coordinates from G:s density
ind_to_remove = round(length(df$x)*0.05) #Calculating how many indexes to
remove to get the 95% HPDI


library(dplyr)

df_ordered_filt = df_ordered %>% slice(-c(1:ind_to_remove)) #Removing lowe
st y values
df_ordered_final = df_ordered_filt[order(df_ordered_filt$x),] #Filtering i
n order of x values
plot(df_ordered_final$x, df_ordered_final$y, type="l") #Plotting result
```

Code assignment 3:

```
#Assignment 3

#This exercise is concerned with directional data. The point is to show yo
u that the posterior distribution
#for somewhat weird models can be obtained by plotting it over a grid of v
alues.
#The data points are observed wind directions at a given location on ten d
i�erent days.
#The data are recorded in degrees: (20, 314, 285, 40, 308, 314, 299, 296,
303, 326),
#where North is located at zero degrees (see Figure 1 on the next page, wh
ere the angles are measured clockwise).
#To �t with Wikipedia's description of probability distributions for circu
lar data we convert
#the data into radians −π ≤ y ≤ π . The 10 observations in
#radians are (−2.79, 2.33, 1.83, −2.44, 2.23, 2.33, 2.07, 2.02, 2.14, 2.54
).
#Assume that these data points conditional on (μ,κ) are
#independent observations from the following von Mises distribution:

#where I0(κ) is the modi�ed Bessel function of the �rst kind of order zero
[see ?besselI in R].
#The parameter μ (−π ≤ μ ≤ π) is the mean direction and κ > 0 is called th
e concentration parameter.
#Large κ gives a small variance around μ, and vice versa. Assume that μ is
known to be 2.4. Let κ ∼ Exponential(λ = 0.5)
#a priori, where λ is the rate parameter of the exponential distribution (
so that the mean is 1/λ).


###Starting assignment###

#a) Derive the expression for what the posterior p(κ|y, μ) is proportional
to. Hence, derive the function f (κ) such
```

```
#that p(κ|y, μ) ∝ f (κ). Then, plot the posterior distribution of κ for th
e wind direction data over a ﬁne grid of κ
#values. [Hint: you need to normalize the posterior distribution of κ so t
hat it integrates to one.]


#Derive the posterior p(k|y,my) is proportional to p(y|k,my)*p(k|my) is pr
oportional to p(y|k,my)*p(k)
#p(k|my) is proportional to p(k) since k is independent of my

#Data and know parameters
degrees = c(-2.79,2.33,1.83,-2.44,2.23,2.33,2.07,2.02,2.14,2.54) #Directio
ns in radians
my = 2.4 #Given my value
lambda = 0.5 #Given lambda value

#Function for von Mises distribution
vonMises = function(kappa, y, my) {
  likelihood = 1
  for (data in y) {
  likelihood = likelihood*exp(kappa*cos(data-my))/(2*pi*besselI(kappa,0))
  }
  return(likelihood)
}

#Function for exponential distribution
exponential = function(tetha, data) {
  return(tetha*exp(-tetha*data))
}

kappa = seq(0.1,10,0.1) #Kappa values to be used in distribution
VMProbs = rep(0, length(kappa)) #List to store probs from von Mis.
expProbs = rep(0, length(kappa)) #List to store probs from exp. distr

#Loop to generate all probs from given distribution
for(i in 1:length(kappa)) {
VMProbs[i] = vonMises(kappa[i],degrees,my)
expProbs[i] = exponential(lambda,kappa[i])
}

posterior = expProbs*VMProbs #Calculating posterior
postIntegral = sum(posterior*0.1) #Calculating the integral for the poster
ior
normPosterior = posterior/postIntegral #Normalizing the posterior to integ
rate to 1
plot(kappa,normPosterior, type="l", xlab="Kappa values", ylab="Likelihood"
) #Plotting the normalized posterior
```

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

```
#As seen in the plot, the approximate posterior mode for kappa is when kap
pa is between 2 & 4
#A good estimation judging by the graph seems to be around ~2.8
```