Simon Boman, simbo160
Frida Andersson, frian632
2022-04-16

Bayesian Learning Computer Lab 1

- Include all solutions and plots to the stated problems with necessary comments
- Submit report with code attached to the solution of each sub-prblem in ode PDF document
- Submit a separate file containing all code

1. Daniel Bernoulli

Given: *$s = 13$, $n = 50$, Beta(α_0,β_0) prior for θ where $\alpha_0 = \beta_0 = 5$.*

A. *Draw random numbers from posterior and verify graphically that the posterior mean and standard deviation converges to the true values as the number of draws grows large.*

```
##############################1a#################################
options(scipen=999) #for removing scientific notation on the plots

n = 50
s = 13
#alpha and beta in prior Beta distribution
alpha0 = 5
beta0 = 5
#alpha and beta in posterior Beta distribution
alpha = alpha0 + s
beta = beta0 + (n - s)

#true mean and standard deviation of the posterior Beta distribution (formulas from wikipedia for Beta distribution)
true_mean = alpha / (alpha + beta)
true_sd = sqrt(alpha * beta / ((alpha + beta)^2 * (alpha+beta+1)))


draws = seq(5000, 500000, 5000) #different number of draws, i.e. sample sizes to use
means = c(1:length(draws)) #vector to store the means of the different sample sizes
sds = c(1:length(draws)) #vector to store the sd's of the different sample sizes

#for each of the different number of draws, compute and store the estimated mean and sd.
#storing the absolute value for easier comparison.
for (i in 1:length(draws)) {
  set.seed(12345)
  data = rbeta(draws[i], alpha, beta)
  mean = mean(data)
  sd = sd(data)
  means[i] = abs(true_mean - mean)
  sds[i] = abs(true_sd - sd)
}

#As we see in the plot, the difference between the true mean/sd and the estimated mean/sd from the simulations
#go to 0 as we increase the number of draws, i.e. the posterior mean and sd converges to the true values of 0.3 and 0.05867387.
max_y = max(means, sds)
plot(draws, means, type="l", lwd = 2, col="blue", ylim=c(0,max_y), main = "Deviance from true mean and sd for different number of draws",
     xlab="Number of draws", ylab="Deviance between true mean/sd and simulated mean/sd" )
points(draws, sds, type="l", lwd = 2, col="green")
legend(x = max(draws)*0.8, y = max_y*0.95, legend = c("mean", "sd"), col = c("blue","green"), lwd = 3)
```

The given alpha and beta in the prior distribution can be used to calculate the true mean and standard deviation as well as the alpha and beta for the posterior distribution. We get that the true mean is 0.300 and the true standard deviation is 0.0586. By simulating draws from the posterior distribution using different sample sizes starting with 1000 draws and incrementing this by 1000 until we reach 100000 draws, we can store the sample size, absolute deviance in mean and absolute deviance in standard deviation for each sample size.

Figure 1 below shows graphically how the posterior mean and standard deviation converges to the true values as the number of draws grows larger. We would of course need a very large sample size for the deviances to reach zero, but as can be seen in the graph the deviances for the mean and standard deviation are around 0.0001 already at around 100000 draws.

Simon Boman, simbo160
Frida Andersson, frian632
2022-04-16

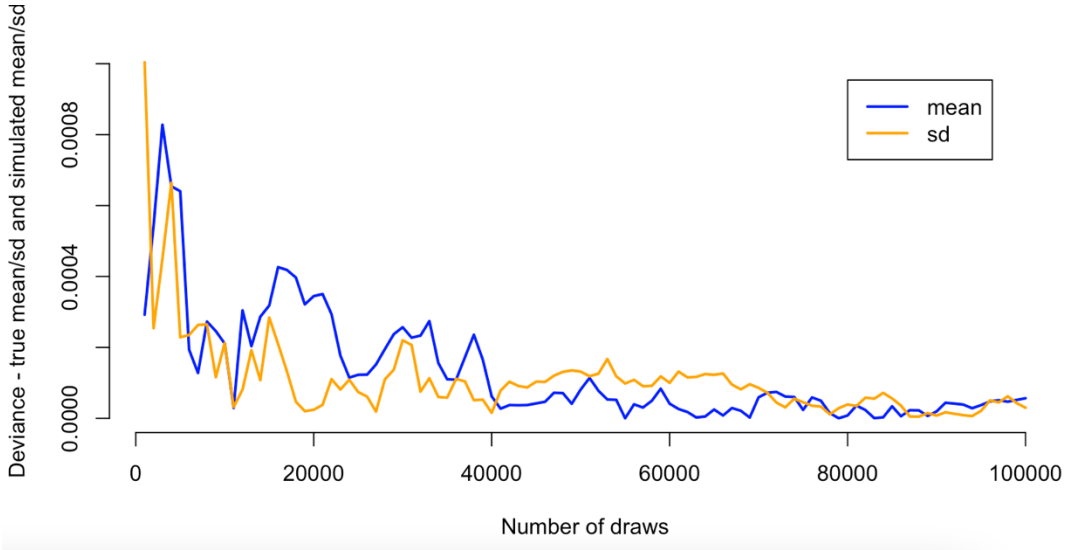

Figure 1 - Calculated mean and standard deviation converge to true mean and standard deviation

B. *Compute the posterior probability Pr(θ < 0.3 | y). Compare with exact value from the Beta posterior.*

```
#################################1b#####################################
#pbeta gives the distribution function for the Beta distribution,
#i.e. the accumulated probability mass up to some value (area under curve less or equal to the value)
true_prob = pbeta(0.3, alpha, beta)
true_prob

nDraws = 10000
set.seed(12345)
data = rbeta(nDraws, alpha, beta)
#using our 10000 draws from the posterior,
#we can calculate the probability by taking number of samples < 0.3 divided by total number of samples
sim_prob = length(data[data<0.3]) / nDraws
sim_prob

#The probability from simulation with 10000 draws, 0.5156, is very close to the exact probability of 0.5150.

#Plot of density estimates of our data, a value of around 0.515 looks reasonable.
plot(density(data))
abline(v=0.3, col="red")
```

Figure 2 below shows the posterior probability for 10000 random draws where the filled area shows Pr(θ < 0.3 | y). This probability is 0.5156, which we can calculate by taking number of samples < 0.3 divided by total number of samples, which gives us 0.5156. This is close to the exact value from the Beta posterior which we calculate using pbeta() (0.5150226).

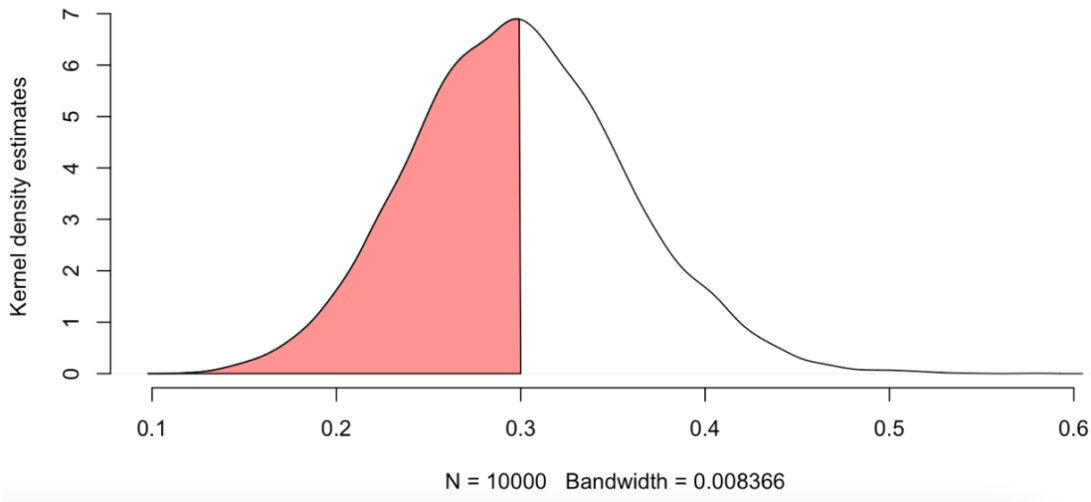

Figure 2 - Filled area shows Pr(θ < 0.3 | y)

C. *Simulate draws from the posterior distribution of the log-odds and plot the posterior distribution of the log-odds.*

```
################################1c#####################################
nDraws = 10000
set.seed(12345)
data = rbeta(nDraws, alpha, beta)

#posterior distribution
hist(data, breaks=25)
plot(density(data), main = "Density estimation of posterior distribution")

#simulating draws from the posterior distribution of the log-odds
#(logit maps the probability values from (0,1) to (-inf, inf))
log_odds = log(data / (1-data))

#posterior distribution of the log-odds
hist(log_odds, breaks=25)
plot(density(log_odds), main = "Density estimation of posterior distribution of log_odds")
```

Figure 3 below shows the posterior distribution of log-odds which have been estimated using simulated draws from the Beta posterior. By simulating draws from the posterior distribution and transforming these draws to log-odds, i.e., by applying the logit function, we map the probability values from (0,1) to (-∞, ∞).

Simon Boman, simbo160
Frida Andersson, frian632
2022-04-16

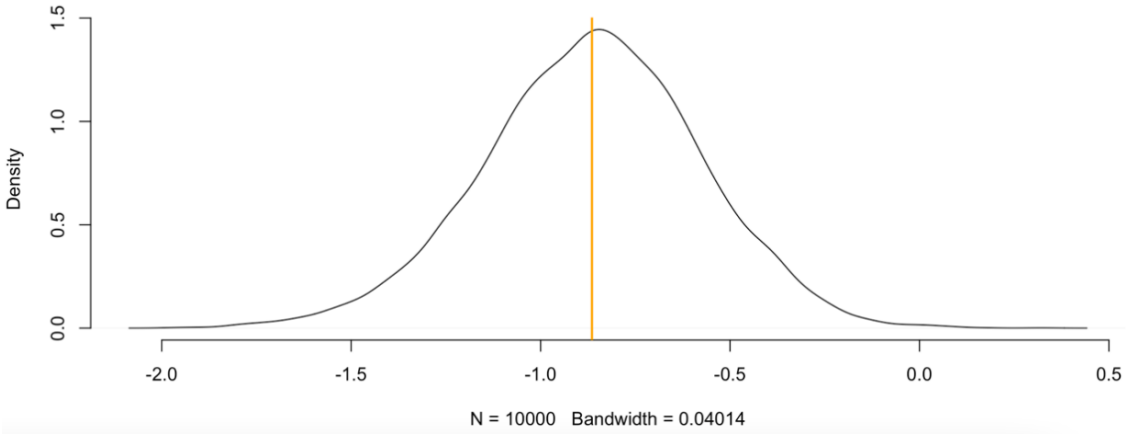

Figure 3 - Posterior distribution of log-odds where orange line shows μ

2. Log-normal distribution and the Gini coefficient

Given: *n = 9, y = [33, 24, 48, 32, 55, 74, 23, 76, 17], μ = 3.5, σ^2 is unknown.*

A. *Simulate 10,000 draws from the posterior of σ^2 and plot the posterior distribution.*

```
##############################2a#####################################
data = c(33, 24, 48, 32, 55, 74, 23, 76, 17)
n <- length(data)

mju = 3.5
tao2 = sum((log(data)-mju)^2)/n

NDraws = 10000
PostDraws = c(1:NDraws)
set.seed(123465)
#drawing 10000 sigma^2 from the inv-chi^2 distribution
PostDraws = ((n)*tao2)/rchisq(NDraws,n)

plot(density(PostDraws), xlim=c(0,2), main = expression(paste('Posterior distribution of ',sigma^2,)))
```

If we assume that the mean is 3.5 we can simulate draws from the posterior distribution of σ^2 by simulating draws from the given Inverse-chi-squared distribution. Simulating 10,000 draws and plotting the posterior distribution of σ^2 can be seen in figure 4 below.

Simon Boman, simbo160
Frida Andersson, frian632
2022-04-16

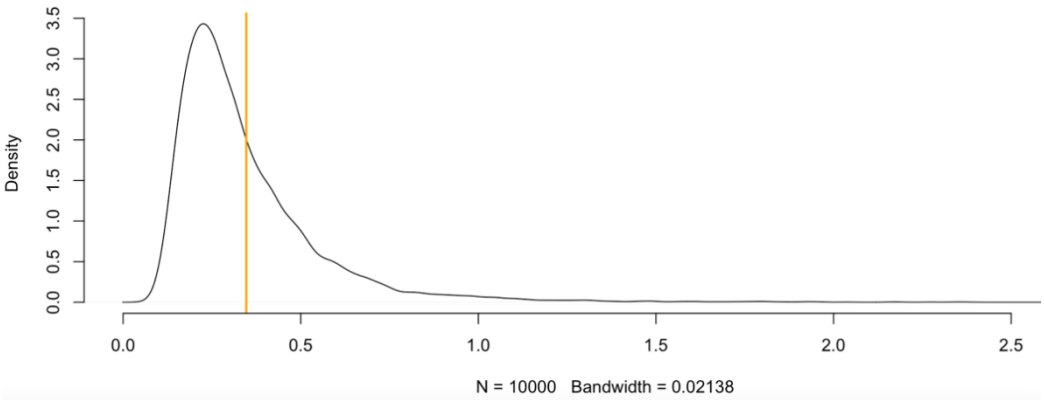

Figure 4 - Posterior distribution of σ2 where orange line shows μ

B. *Use the posterior draws in A. to compute the posterior distribution of the Gini coefficient G for the current data set.*

```
##############################2b#####################################
#phi(z) is the cumulative distribution function (CDF) for the standard normal distribution with mean 0 and variance 1.
#pnorm gives the distribution function for the Normal distribution with default mean=1 and sd=1 (the CDF),
#i.e. the accumulated probability mass up to some value (area under curve less or equal to the value)
G = 2*pnorm(sqrt(PostDraws/2)) - 1
plot(density(G), main = "Posterior distribution of the Gini coefficient G for the current data set ")
```

Since the posterior draws in A. follow a log N(μ, σ^2) distribution, we can use G = 2 Φ (σ /$\sqrt{2}$) -1 to calculate our Gini coefficient. Figure 5 below shows the posterior distribution for the calculated Gini coefficient and μ in orange.

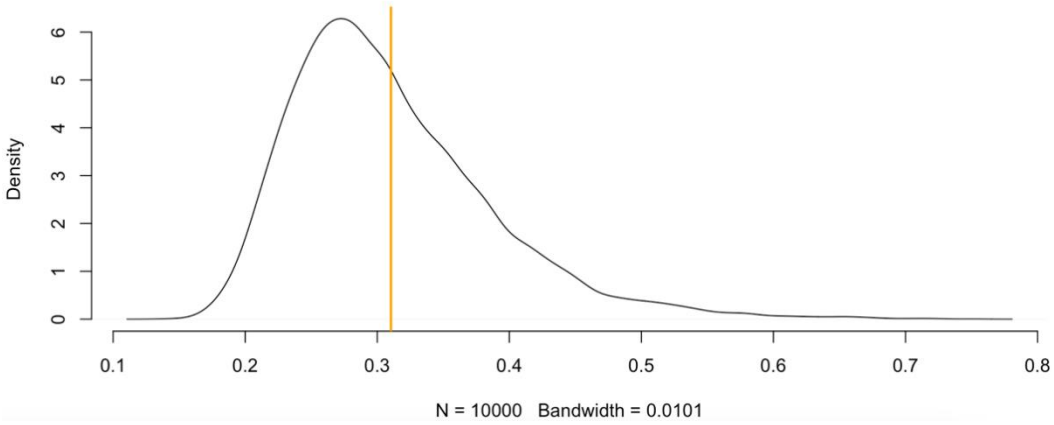

Figure 5 - Posterior distribution of the Gini coefficient where orange line shows μ

Simon Boman, simbo160
Frida Andersson, frian632
2022-04-16

C. Use the posterior draws from B. to compute 95 % equal tail credible interval for G.

```
##############################2c#####################################
#estimation of the the 95% equal tail credible interval using mean and sd
mean = mean(G)
sd = sd(G)
lowerBound = mean - 1.96*sd
upperBound = mean + 1.96*sd
lowerBound
upperBound

#more exact values using qnorm
lowerBound = qnorm(0.025, mean, sd)
upperBound = qnorm(0.975, mean, sd)
lowerBound
upperBound

plot(density(G), main = "Posterior distribution of the Gini coefficient G for the current data set,
abline(v=lowerBound, col="red")
abline(v=upperBound, col="red")
```

By calculating mean and standard deviation from our posterior distribution of the Gini coefficient, we can calculate our lower and upper bound for the ETI. K is set to 1.96 as we want our ETI to be 95 %. The interval can also be calculated using the built in qnorm()-function. Figure 6 shows our posterior distribution for the Gini coefficient where the filled area shows our 95 % equal tail credible interval, which is [0.1583, 0.46071].

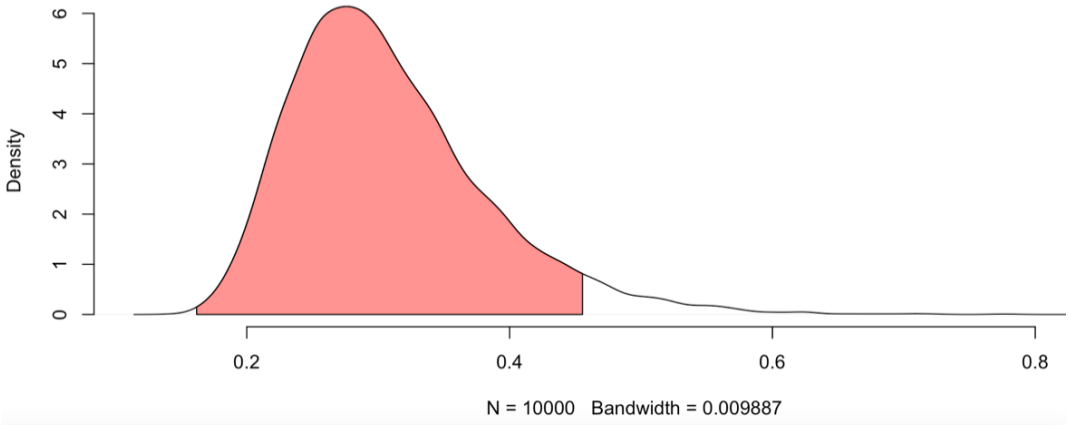

Figure 6 - Posterior distribution for the Gini coefficient where filled are shows 95 % equal tail credible interval

Simon Boman, simbo160
Frida Andersson, frian632
2022-04-16

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.

```
##############################2d#####################################
#extract x- and y-values from the density estimate and sort these on y-value
#density = density(G, n=10000)
density = density(G)
x = density$x
y = density$y
xy = cbind(x,y)
xy_sorted = xy[order(xy[,2],decreasing=TRUE),]

#calculate what value 95% of the accumulated y-values corresponds to, i.e. the accumulated value to stop at
#when deciding which (x,y)-pairs to look at
stop_value = sum(xy_sorted[,2])*0.95

#accumulated sum of the sorted y-values
#find out at which index we reach 95% mass (i.e. the stop_value),
#then index 1 to stop_index corresponds to 95% mass
acumulatedMass <- function(stop_val) {
  for (i in 1:length(xy_sorted)) {
    if (sum(xy_sorted[1:i,2]) > stop_val) {
      return (i-1)
    }
  }
}
stop_index = acumulatedMass(stop_value)
```

```
#only keep the 95% of the mass corresponding to highest densities
xy_sorted_95 = xy_sorted[1:stop_index,]

#find the indexes corresponding to lower and upper bound
lower_index = which.min(xy_sorted_95[,1])
upper_index = which.max(xy_sorted_95[,1])
lower_index
upper_index

#extract actual lower and upper bounds
HPDIlower = xy_sorted_95[lower_index,1]
HPDIupper = xy_sorted_95[upper_index,1]
HPDIlower
HPDIupper

plot(density(G), main = "Posterior distribution of the Gini coefficient G for the current data set, \n with
legend(x = 0.6, y = 5, legend = c("Equal Tail Credible Interval", "HPDI"), col = c("red","blue"), lwd = 3)
abline(v=lowerBound, col="red")
abline(v=upperBound, col="red")
abline(v=HPDIlower, col="blue")
abline(v=HPDIupper, col="blue")
```

To compute the 95% HPDI, we begin by extracting x- and y-values from the density estimate from the density()-function and sorting these on decreasing y-value. We can then calculate what value 95% of the accumulated y-values corresponds to, i.e. the accumulated value to stop at when deciding which (x,y)-pairs to look at, and find which indexes corresponds to the 95% HPDI . We can then find the lower and upper bound for the HPDI.

Figure 7 shows the posterior distribution for the Gini coefficient where the filled area is the 95 % HPDI compared to the orange lines representing 95 % ETI. A comparison of HPDI and ETI shows that the HPDI gives a better summary for the posterior distribution for the Gini coefficient, due the asymmetry of the data with one tail being significantly fatter and longer than the other.

Simon Boman, simbo160
Frida Andersson, frian632
2022-04-16

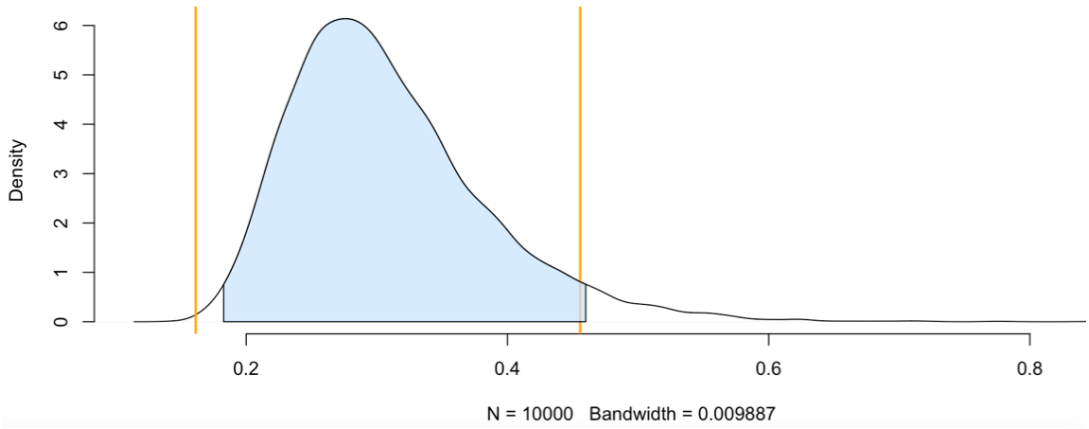

Figure 7 - Posterior distribution for the Gini coefficient where filled area shows 95 % HPDI and orange lines 95 % ETI calculated in C.

3. Bayesian inference for the concentration parameter in the von Mises distribution

Given: *n = 10, y = [1.83, 2.02, 2.33, -2.79, 2.07, 2.02, -2.44, 2.14, 2.54, 2.23] in radians, μ = 2.51.*

A. *Derive the expression for what the posterior p(κ | y, μ) is proportional to and plot the posterior distribution of κ for the wind direction data over a fine grid of κ values.*

```r
################################3a#####################################
degrees = c(285, 296, 314, 20, 299, 296, 40, 303, 326, 308)
radians = c(1.83, 2.02, 2.33, -2.79, 2.07, 2.02, -2.44, 2.14, 2.54, 2.23)

mju = 2.51
lambda = 1
besselDegree = 0

#posterior proportional to likelihood * prior
#likelihood proportional to exp(k*cos(y-mju)) / I0(k)
#K~Exp(lambda=1) which gives the prior: lambda*exp(-lambda*k)
posterior = function(k, y, mju, degree, lambda) {
  #posterior proportional to:
  posterior = prod( exp(k*cos(y-mju))/besselI(k, degree) ) * lambda*exp(-lambda*k)
  #posterior = prod( exp(k*cos(y-mju))/besselI(k, degree) ) * dexp(k, rate=lambda)
}

#grid of k-values to plot over
gridstep = 0.005
k = seq(0, 7, gridstep)

#calculate and store the value for each of the k-values
post = c(1:length(k))
for(i in 1:length(k)) {
  post[i] = posterior(k[i], radians, mju, besselDegree, lambda)
}

#normalizing the posterior distirbution of k so that it integrates to 1
post = post/(sum(post)*gridstep)
plot(k, post, type="l", main="Posterior distribution of k", col="blue")
```

The expression for what the posterior distribution of κ is proportional to can be derived by taking the given likelihood times the prior, which both are given in the lab description. By doing so, and calculating the posterior for different values, we get the following plot, see figure 8.

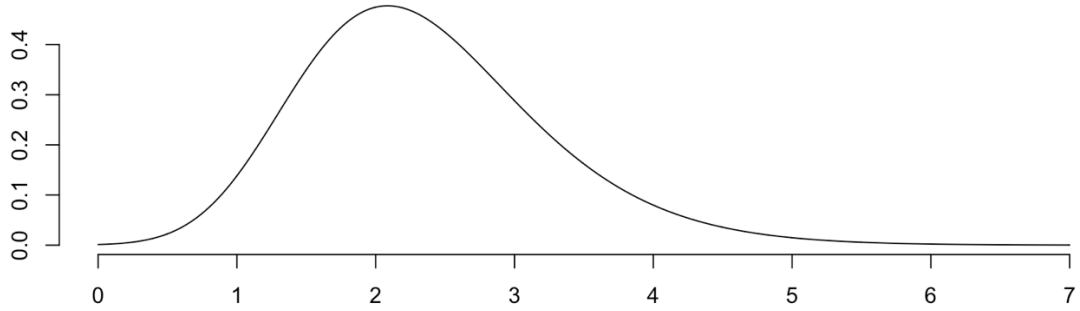

Figure 8 - Posterior distribution of κ

B. *Find the approximate posterior mode of κ from the information in A.*

```
################################3b#####################################
#the approximate posterior mode of k is the largest value in the posterior
approx_post_K = k[which.max(post)]
approx_post_K
plot(k, post, type="l", main="Posterior distribution of k", col="blue")
abline(v=approx_post_K, col="red")
```

The approximate posterior mode of κ using the information from A., can be found by taking the maximum value in the posterior, which is 2.085. See figure 9 for a visualization of the mode.

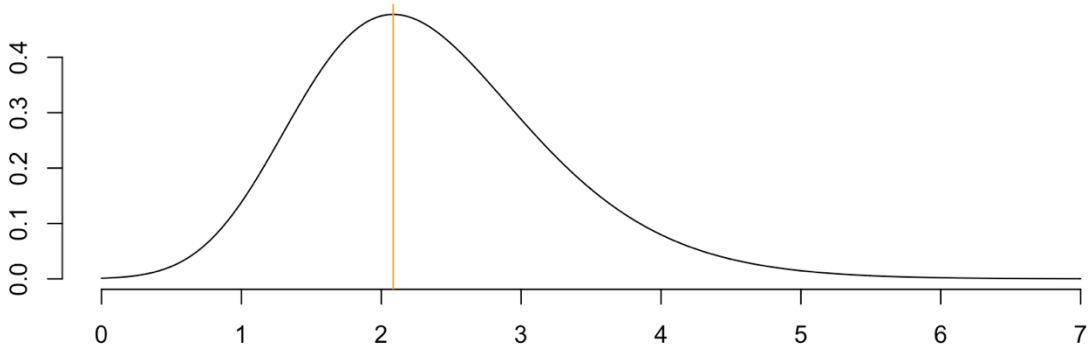

Figure 9 - Posterior mode of k