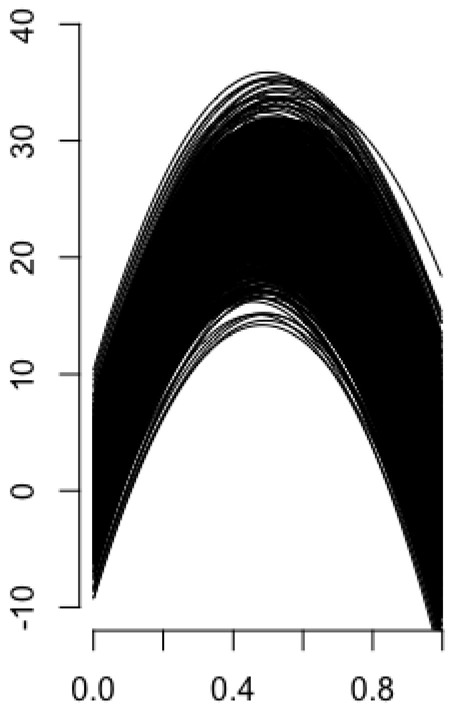
**Assignment 1:**

**1a)**



The plot above shows the predictions from the prior draws when we use v0 = 100. We decided to increase the value for degrees of freedom as v0=1 didn’t give good results. The plot above gives a reasonable output where temperatures are expected to peak in the middle of the year. The predictions follow our prior beliefs regarding temperatures over the year.

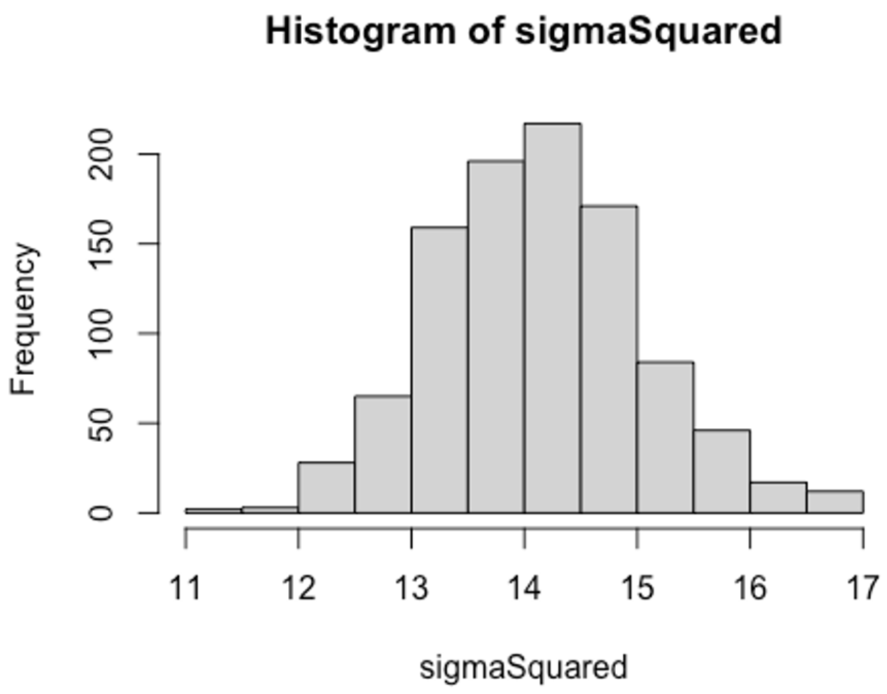
**Code:**

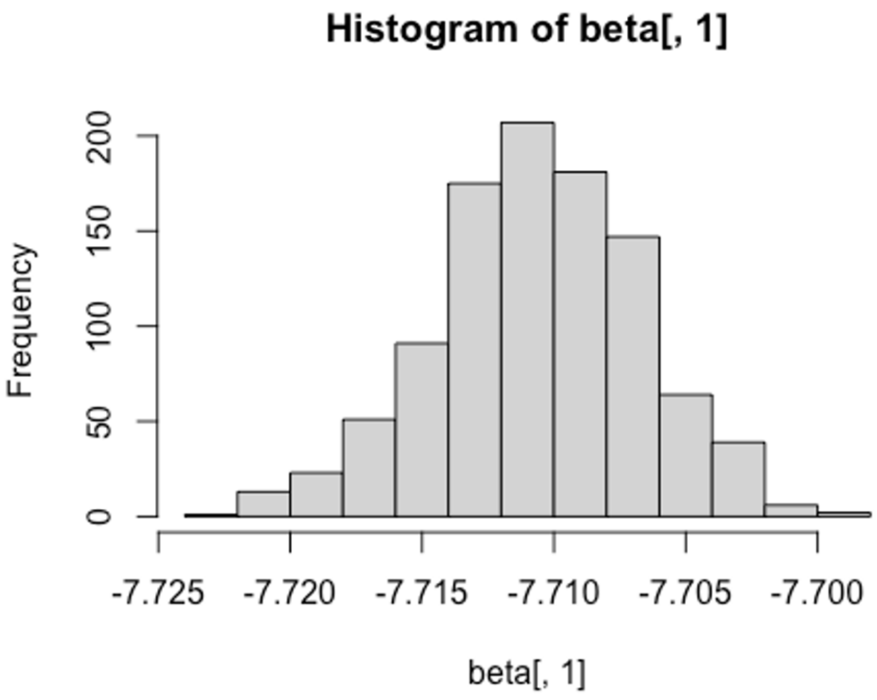
```
#The dataset Linkoping2022.xlsx contains daily average temperatures (in degree Celcius)
#in Linköping over the course of the year 2022. Use the function read_xlsx(), which is
included
#in the R package readxl (install.packages("readxl")), to import the dataset in R. The
response variable is
#temp and the covariate time that you need to create yourself is de□ned by

#time =(#days since beginning of year)/365

#A Bayesian analysis of the following quadratic regression model is to be performed:
#temp=β0+β1·time+β2·time +ε,ε~N(0,σ).
```

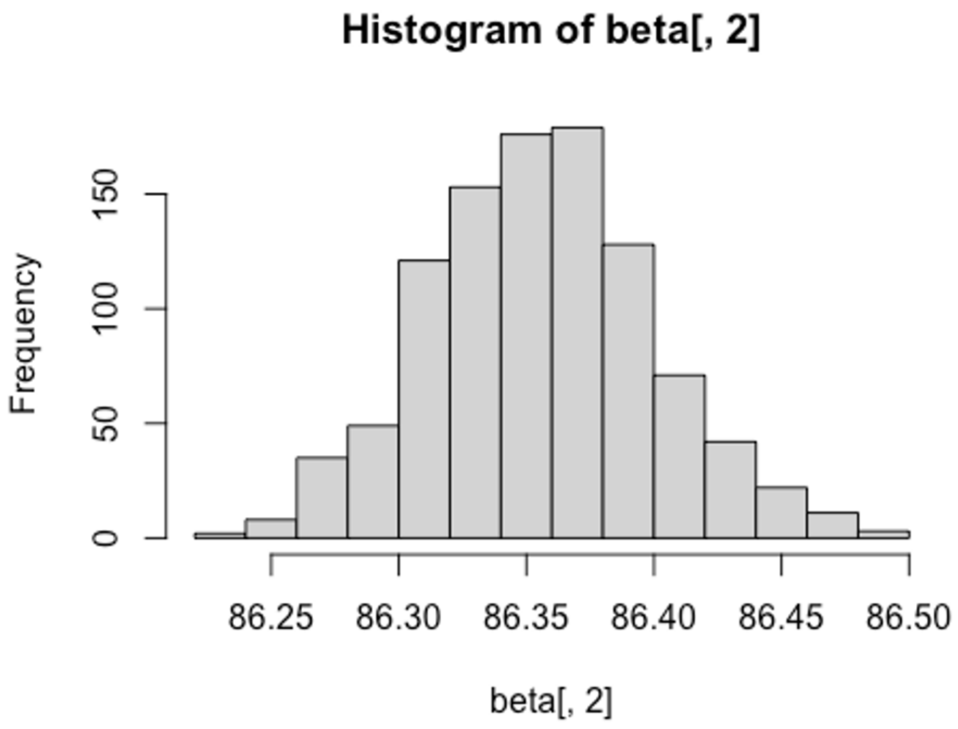
```
#Read data and import library
library(readxl)
data = read_xlsx("Linkoping2022.xlsx")

#Calculate variable time
data$time = as.integer(difftime(data$datetime,"2022-01-01", units = "day"))/365

#a)
# Use the conjugate prior for the linear regression model. The prior hyper- parameters μ0,
Ω0, v0 and σ02
#shall be set to sensible values. Start with μ0 = (0,100,−100)T, Ω0 = 0.01 · I3, v0 = 1 and
σ02 = 1.
#Check if this prior agrees with your prior opinions by simulating draws from the joint prior of
all parameters
#and for every draw compute the regression curve. This gives a collection of regression
curves; one for each draw
#from the prior. Does the collection of curves look reasonable? If not, change the prior
hyperparame- ters until the
#collection of prior regression curves agrees with your prior beliefs about the regression
curve.
#[Hint: R package mvtnorm can be used and your Inv-χ2 simulator of random draws from
Lab 1.]

library(mvtnorm)

### LOAD DATA ###
my_zero = c(0,100,-100)
omega_zero = 0.1*diag(3)
v_zero = 100 #Increased from 1 since we got bad regression curves when using low values
sigma2_zero = 1

#################



###Simulate draws from prior for sigma squared
Ndraw = 1000 #Integer for how many draws we want to make
set.seed(12345)
chi_vals = rchisq(Ndraw,v_zero) #Drawing 10000 random samples from chi-squared
distribution
sigmaSquared = v_zero*sigma2_zero/chi_vals #Calculating prior sigma squared

###Simulate draws from prior beta
beta=matrix(0,length(sigmaSquared),3) #Create matrix to store beta draws
for (i in 1:length(sigmaSquared)) {
beta[i,] = rmvnorm(1, my_zero, sigmaSquared[i]*solve(omega_zero)) #Drawing beta values
from prior
}
```

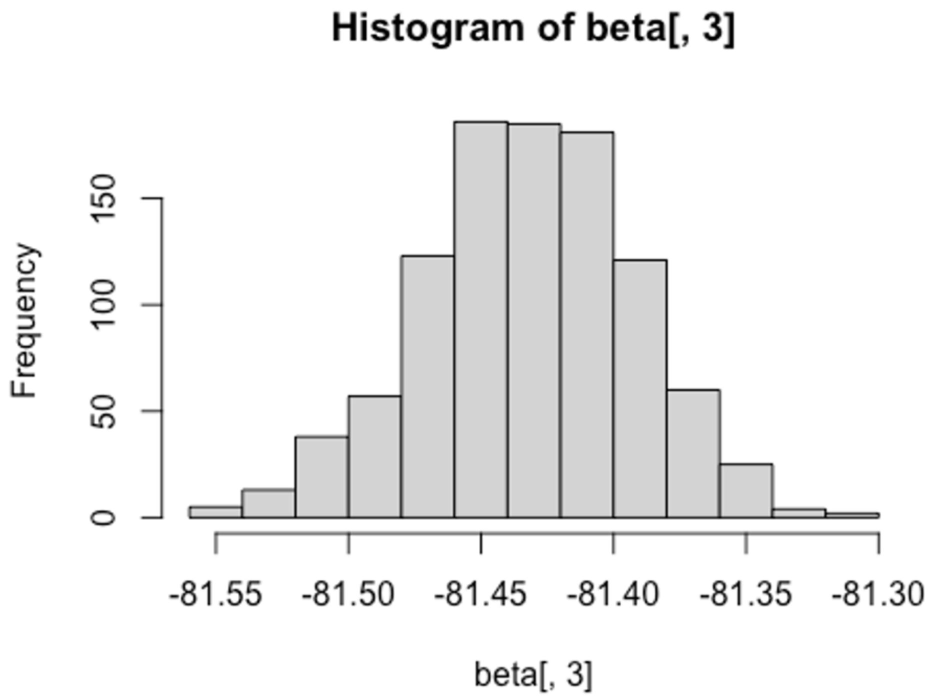
```
reg = matrix(0,365,Ndraw) #Creatibg matrix to store tge set of regressions
for (i in 1:Ndraw) {
reg[,i]=beta[i,1]+data$time*beta[i,2]+(data$time**2)*beta[i,3] #Calculating regressions
}
###Creating plot where all regressions can be plotted
plot.new()
plot.window(xlim=c(0,1), ylim=c(-10,40))
axis(side = 1)
axis(side = 2)
for (i in 1:Ndraw) {
  lines(data$time, reg[,i], type = "l") #Plottnig regression
}

#Conclusion: By increasing the prior degrees of freedom, we got better regression curves.
```

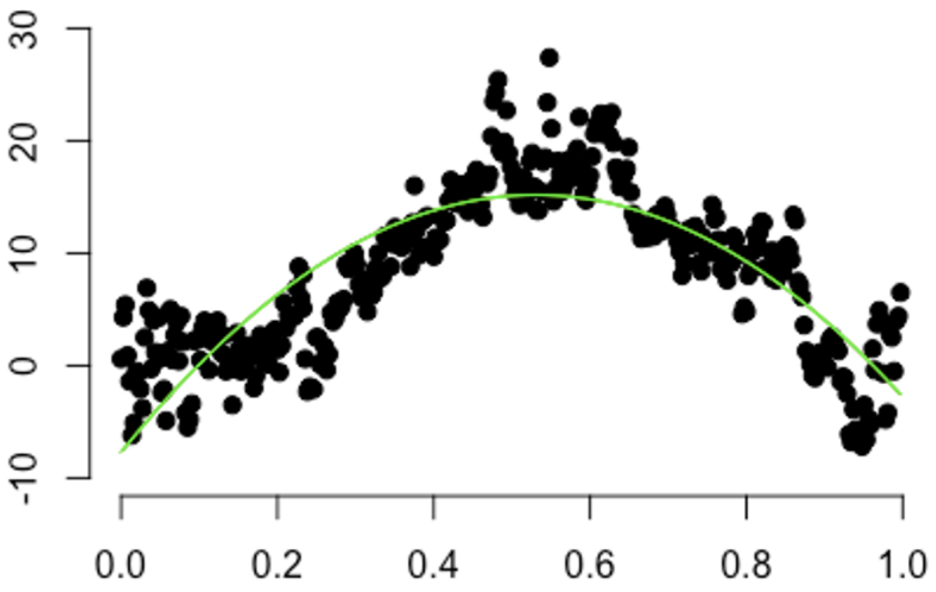
**1b)**




Histogram of beta[, 1]
Frequency
0
50
100
150
200
-7.725
-7.720
-7.715
-7.710
-7.705
-7.700
beta[, 1]


Histogram of beta[, 2]
Frequency
0
50
100
150
86.25
86.30
86.35
86.40
86.45
86.50
beta[, 2]





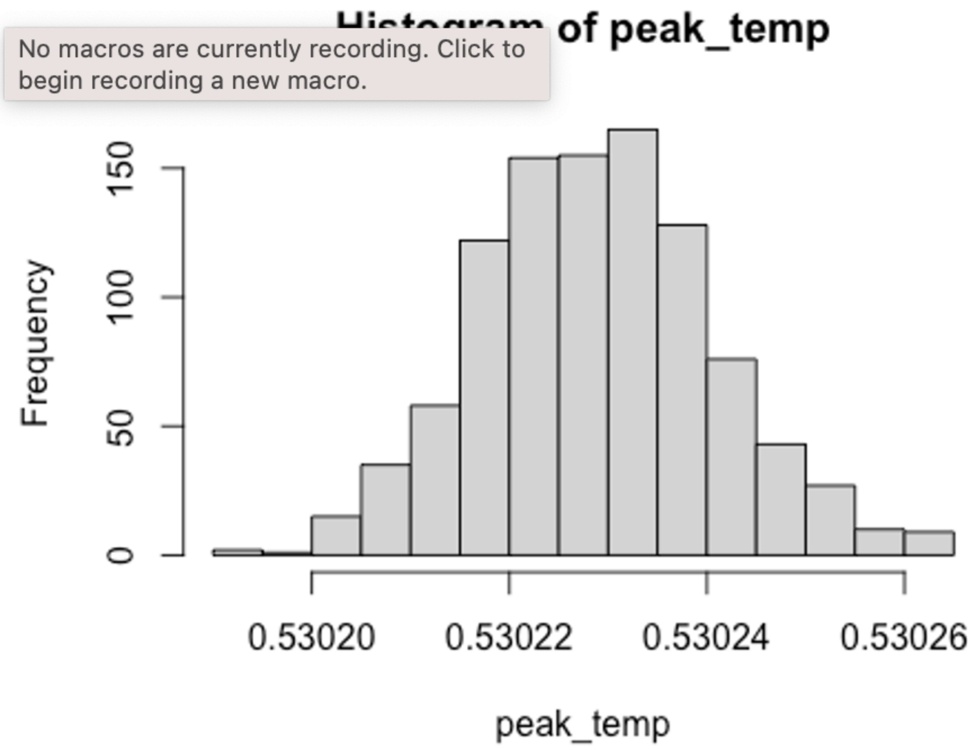
In the scatter plot above, we can see the actual temperatures (indicated with black dots). Furthermore we can see two green lines (indicating under and upper limit) covering the median (which is indicated by a red line). The under and upper limit are not the same, however they are very close to each other which is reasonable as the histograms above suggest that all beta values are in a really slim interval, giving a slim distribution in the posterior. We did not expect the equal tail interval to capture the points as the interval is based on posterior predictions, not the true data.

**Code:**

```
#b)
#Write a function that simulate draws from the joint posterior distribution of β0, β1,β2 and σ2.
#i) Plot a histogram for each marginal posterior of the parameters.
#ii) Make a scatter plot of the temperature data and overlay a curve for the posterior median of the
#regression function f(time) = E[temp|time] = β0 + β1 · time + β2 · time2, i.e. the median of f (time)
#is computed for every value of time. In addition, overlay curves for the 90% equal tail posterior probability
#intervals of f(time), i.e. the 5 and 95 posterior per- centiles of f (time) is computed for every value of time.
#Does the posterior probability intervals contain most of the data points? Should they?


###DATA###
n = length(data$temp)
v_n = v_zero + n
X = cbind(1, data$time, data$time**2)
beta_hat=solve(t(X)%*%X)%*%t(X)%*%data$temp
my_n=solve(t(X)%*%X+omega_zero)%*%(t(X)%*%X%*%beta_hat+omega_zero%*%my_zero)
omega_n = t(X)%*%X+omega_zero
sigma2_n = as.numeric((sigma2_zero*v_zero +
(t(data$temp)%*%data$temp+t(my_zero)%*%omega_zero%*%my_zero-t(my_n)%*%omega_n%*%my_n))/v_n)
###########

###Draw posterior values for sigma squared
Ndraw = 1000 #Integer for how many draws we want to make
set.seed(12345)
chi_vals = rchisq(Ndraw,v_n) #Drawing 10000 random samples from chi-squared distribution
sigmaSquared = v_n*sigma2_n/chi_vals #Calculating prior sigma squared

###Simulate draws for posterior beta
beta=matrix(0,length(sigmaSquared),3) #Create matrix to store beta simulations
for (i in 1:length(sigmaSquared)) {
  beta[i,] = rmvnorm(1, my_n, sigmaSquared[i]*solve(omega_n)) #Drawing beta values from posterior
}

###Plotting histogram for posterior draws
hist(sigmaSquared) #Histogram for sigma_squared
```

```r
hist(beta[,1]) #Histogram for beta_zero

hist(beta[,2]) #Histogram for beta_one

hist(beta[,3]) #Histogram for beta_two

###Scatter plot
plot.new()
plot.window(xlim=c(0,1), ylim=c(-10,30))
axis(side = 1)
axis(side = 2)
points(data$time, data$temp, pch=19)

reg = matrix(0,365,Ndraw) #Creating matrix to store the set of regressions
for (i in 1:Ndraw) {
  reg[,i]=X%*%beta[i,] #Calculating regressions
}

temp_medians = rep(0,dim(reg)[1]) #Creating list to store median values of temps each day
for(i in 1:dim(reg)[1]) {
 temp_medians[i] = median(reg[i,])
}

#Plotting median values
lines(data$time, temp_medians, pch=19, col="red")

#Creating 90% crediible interval
min = 0.05*dim(reg)[2]+1
max = 0.95*dim(reg)[2]

under_limit = rep(0,dim(reg)[1])
upper_limit = rep(0,dim(reg)[1])
for(i in 1:dim(reg)[1]) {
  sorted = sort(reg[i,])
  under_limit[i]=quantile(sorted,probs=0.05)
  upper_limit[i]=quantile(sorted,probs=0.95)
}

lines(data$time, under_limit, col="green")
lines(data$time, upper_limit, col="green")

###Conclusion:The equal tail intervals are very narrow as we can see in the plot. As we saw in the previously
### presented histograms, the posterior draws are narrow which yields a very slim upper and under limit
### (they are very similar, but not the same however)
```

**1c)**



The histogram above suggests a slim interval for when the temperature is expected to peak. It is a reasonable prediction as this is in the middle of July.

**Code:**

```
#c)
#It is of interest to locate the time with the highest expected temperature (i.e. the time where f(time) is maximal).
#Let's call this value x˜. Use the simulated draws in (b) to simulate from the posterior distribution of x˜.
#[Hint: the regression curve is a quadratic polynomial. Given each posterior draw of β0, β1 and β2,
#you can nd a simple formula for x˜.]


#As it is a polynomial function with a maximum point, the time for the highest temp
#should be located where the derivative is 0

###Simulating the expected time where the temperature is at its peak
peak_temp = rep(0,Ndraw)
for (i in 1:Ndraw) {
  peak_temp[i]=(-1)*(beta[i,2]/(2*beta[i,3])) #Estimating time when temperature reaches its peak
}
```

```
hist(peak_temp) #Plotting histogram of when the temperature is expected to peak

###Conclusion: Once again, the posterior distribution is very narrow.
###Furthermore, posterior distribution that temperature is likely to peak in mid July
### which is reasonable
```

**1d)**

The prior my_0 should be kept close to 0 since we want most coefficients to be close to 0 to obtain shrinkage as terms increase in order. The omea_0 should be X*I (X times identity matrix), where X should be set to a large value to obtain a smaller spread to make beta values more likely to stay close to 0. However, you could also argue that X should be set to a low value as this would give the data less impact, making it less likely to overfit to the data.

**Code:**

```
#d)
#Say now that you want to estimate a polynomial regression of order 10, but you suspect that higher
#order terms may not be needed, and you worry about over tting the data. Suggest a suitable prior that mitigates
#this potential problem. You do not need to compute the posterior.
#Just write down your prior. [Hint: the task is to specify μ0 and Ω0 in a suitable way.]


#Answer:
#The prior my_zero should be 0 (or close to this) and the prior omega_zero should be X*I (X times identity matrix).
#We want a low value for my_zero because we want most coefficients to be close to 0 to obtain shrinkage as
#terms increases in order.
#Furthermore, the X value for the omega_zero should be set to a large value in order to obtain
# a smaller spread for the beta values, making them more likely to stay close to zero.
#On the other hand, you could also argue that the X value should be low as this would give the data
#less impact and therefore decreas the risk of overfitting to the data.
```

**Assingment 2:**

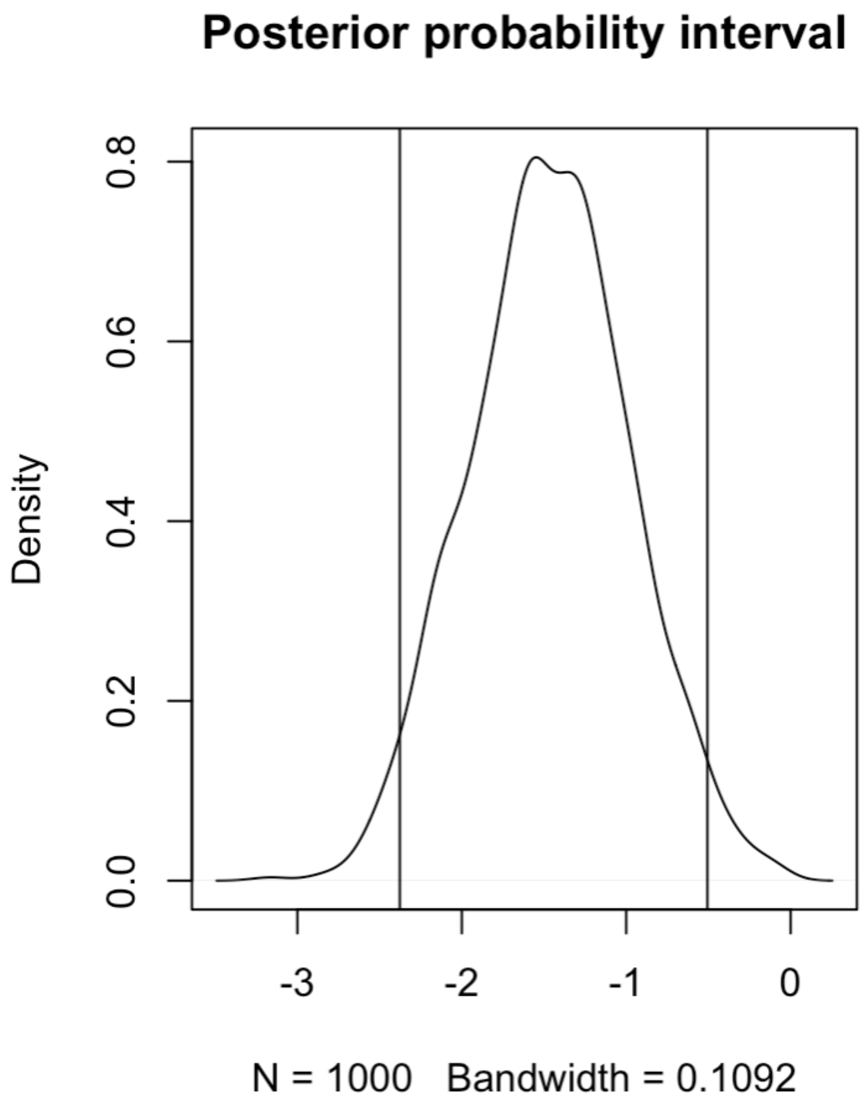
**2a)**

Mean values for Beta

```
[1,] -0.04036943 "Constant"
[2,] -0.03730689 "HusbandInc"
[3,]  0.17868950 "EducYears"
[4,]  0.12073637 "ExpYears"
[5,] -0.04618995 "Age"
[6,] -1.47248930 "NSmallChild"
[7,] -0.02014458 "NBigChild"
```

<u>Approximate std.dev for Beta</u>

```
Constant     HusbandInc   EducYears    ExpYears       Age        NSmallChild   NBigChild
1.38198487  0.02198474  0.08920960  0.03335982  0.02747315  0.47746764  0.16401959
```



Higher bound is -0.5, we can reject the null hypothesis. It seems a higher number of small children will make a woman less likely to work.

**Code:**

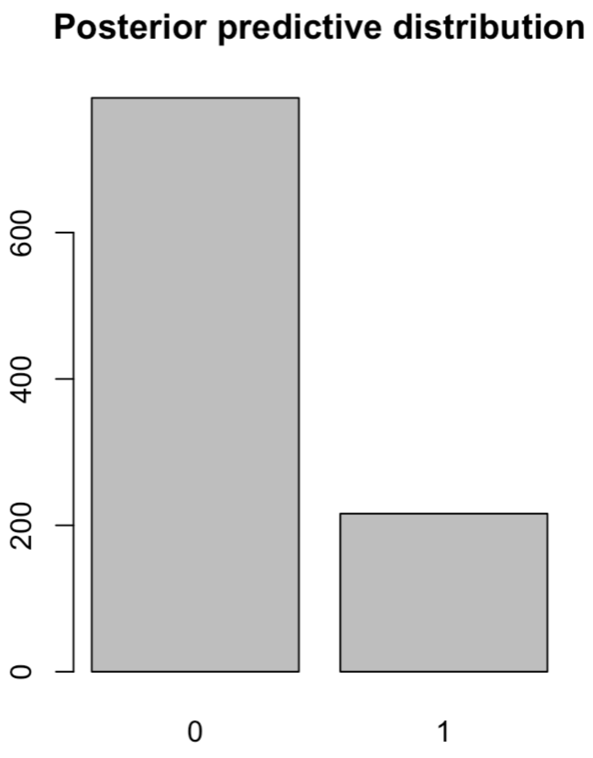
```
# Assignment 2 Posterior approximation for classification with logistic regression
#The dataset WomenAtWork.dat contains n = 132 observations on the following eight
variables related to women:

#2.a)
#Consider the logistic regression model: Pr(y = 1|x, β) =  exp(xTβ) / (1 + exp (xT β))
# where y equals 1 if the woman works and 0 if she does not.
#x is a 7-dimensional vector containing the seven features (including a 1 to model the
intercept).
#The goal is to approximate the posterior distribution of the parameter vector β with a
multivariate normal distribution
#  β|y, x ~ N(B_hat, J^-1(B_hat)) where B_hat is the posterior mode and J(B_hat) = −∂2
lnp(β|y)|
#  Note that   ∂ β ∂ β T β = B_hat is the negative of ∂2 ln p(β|y) the observed Hessian
evaluated at the posterior mode.
#Note that ∂β∂βT is a 7 × 7 matrix with second derivatives on the diagonal and
cross-derivatives ∂2 ln p(β|y) ∂βi∂βj on the off-diagonal.
#You can compute this derivative by hand, but we will let the computer do it numerically for
you.
#Calculate both B_hat and J(B_hat) by using the optim function in R.
#[Hint: You may use code snippets from my demo of logistic regression in Lecture 6.] Use
the prior β ~ N(0,τ2I), where τ = 2.
#  Present the numerical values of β˜ and J−1(β˜) for the WomenAtWork dat
#a. Compute an approximate 95% equal tail posterior probability interval for the regression
coefficient to the variable NSmallChild.
#Would you say that this feature is of importance for the probability that a woman works?
#[Hint: You can verify that your estimation results are reasonable by comparing the posterior
means
# to the maximum likelihood estimates, given by: glmModel <- glm(Work ~ 0 + ., data =
WomenAtWork, family = binomial).]


#Import packages
```

```
library(mvtnorm)

#Read the data
data <-  (read.table("WomenAtWork.dat", header=TRUE))

Xnames <- names(data[,2:ncol(data)]) #Names of columns
head(data)

##   Work Constant HusbandInc EducYears ExpYears Age NSmallChild NBigChild
## 1    1        1  22.394940        12        7  43           0         3
## 2    0        1   7.232000         8       10  34           0         7
## 3    1        1  18.271990        12        4  41           1         5
## 4    0        1  28.069000        14        2  43           0         2
## 5    1        1   7.799889        12       10  31           0         1
## 6    0        1  28.630000        16        6  37           0         3

n_rows <- dim(data)[1]  #number of observations


LogPostLogistic <- function(betas,y,X,mu,Sigma){
  linPred <- X%*%betas;
  logLik <- sum( linPred*y - log(1 + exp(linPred)) );
  #if (abs(logLik) == Inf) logLik = -20000; # Likelihood is not finite, stear the optimizer away 
from here!
  logPrior <- dmvnorm(betas, mu, Sigma, log=TRUE);

  return(logLik + logPrior)
}

#initialize variables
set.seed(123)
y <- as.numeric(data[,1]) #target output
X <- as.matrix(data[,-1]) #training data
X <- apply(X, c(1, 2), as.numeric) #convert from characters to doubles
n_col <- dim(X)[2] #number of columns in training data
tao = 2 #prior std.dev for Beta
initVal <- matrix(0,n_col,1) #initial values for Beta
mu <- as.matrix(rep(0,n_col)) # Prior mean vector for Beta
Sigma <- tao^2*diag(n_col) # Prior covariance matrix for Beta
logPost <- LogPostLogistic; #function to be minimized

OptimRes <- 
optim(initVal,logPost,gr=NULL,y,X,mu,Sigma,method=c("BFGS"),control=list(fnscale=-1),hes
sian=TRUE)

# Printing the results of mean and variance of normally distributed Beta
OptimBeta <- OptimRes$par
names(OptimBeta) <- Xnames # Naming the coefficient by covariates
```

```
PostCov <- solve(-OptimRes$hessian) #get covariance matrix by inverting the negative
hessian
approxPostStd <- sqrt(diag(PostCov)) # Computing approximate standard deviations.
names(approxPostStd) <- Xnames # Naming the coefficient by covariates
print('The posterior mode is:')

## [1] "The posterior mode is:"

print(OptimBeta)

##              [,1]
## [1,] -0.04036943
## [2,] -0.03730689
## [3,]  0.17868950
## [4,]  0.12073637
## [5,] -0.04618995
## [6,] -1.47248930
## [7,] -0.02014458
## attr(,"names")
## [1] "Constant"     "HusbandInc"  "EducYears"   "ExpYears"     "Age"
## [6] "NSmallChild" "NBigChild"

print('The approximate posterior standard deviation is:')

## [1] "The approximate posterior standard deviation is:"

print(approxPostStd)

##      Constant  HusbandInc   EducYears   ExpYears        Age NSmallChild
##  1.38198487  0.02198474  0.08920960  0.03335982  0.02747315  0.47746764
##   NBigChild
##  0.16401959

#Draw samples from distribution of NSmallChild
my <- OptimBeta[6] #mean of Beta[NSmallChild]
sigma <-approxPostStd[6] #std.dev of Beta[NSmallChild]
samples <- rnorm(1000, my, sigma)
quantiles <-  quantile(samples, c(0.025, 0.975))
plot(density(samples), main="Posterior probability interval")
abline(v=quantiles)

print("Upper and lower bounds are:")

## [1] "Upper and lower bounds are:"

print(quantiles)

##       2.5%     97.5%
## -2.3995187 -0.4994643
```

```
#Check that estimation results are reasonable
glmModel <- glm(Work ~ 0 + ., data = data, family = binomial)
print(OptimBeta)

##                  [,1]
## [1,] -0.04036943
## [2,] -0.03730689
## [3,]  0.17868950
## [4,]  0.12073637
## [5,] -0.04618995
## [6,] -1.47248930
## [7,] -0.02014458
## attr(,"names")
## [1] "Constant"       "HusbandInc"  "EducYears"   "ExpYears"      "Age"
## [6] "NSmallChild" "NBigChild"

print(glmModel$coefficients)

##       Constant  HusbandInc   EducYears   ExpYears        Age NSmallChild
##  0.02262929 -0.03796308  0.18447411  0.12131763 -0.04858167 -1.56485140
##   NBigChild
## -0.02526059

#values are similar, estimation seems reasonable. Since the boundary is negative, it seems like a higher number
#of small children makes a woman less likely to work.
```

**2b)**

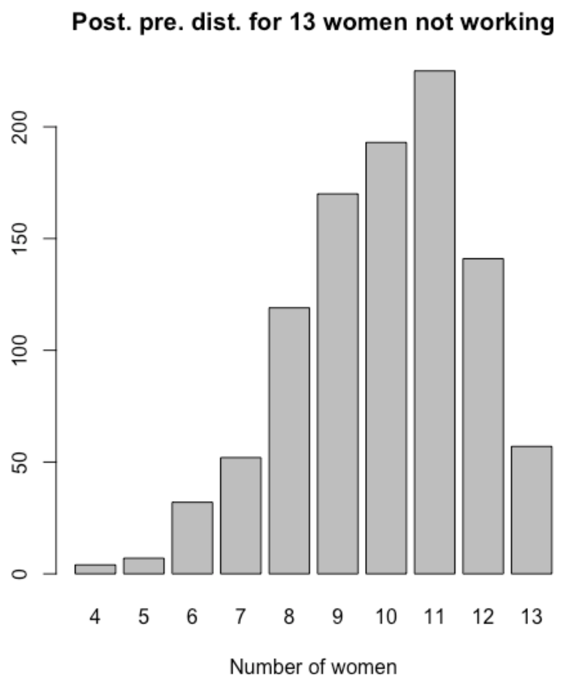


Samples drawn from the probabilities show that there is a high chance that this woman does not work, since roughly 80% of samples predict 0.

**Code:**

```
#b) Use your normal approximation to the posterior from (a).
#Write a function that simulate draws from the posterior predictive distribution of Pr(y = 0|x),
#where the values of x corresponds to a 40-year-old woman, with two children (4 and 7
years old),
#11 years of education, 7 years of experience, and a husband with an income of 18.
#Plot the posterior predictive distribution of Pr(y = 0|x) for this woman.
#[Hints: The R package mvtnorm will be useful. Remember that Pr(y = 0|x) can be calculated
for each posterior draw of β.]

#sigmoid function
sigmoid <- function(x, beta) {
  s <- beta %*% x
  return (exp(s) / (1+exp(s)))
}

set.seed(123)
PostStd <- c(approxPostStd) #convert to vector
beta_samples <- rmvnorm(n=1000, mean=OptimBeta, sigma=PostCov) #draw samples of
Beta from the distribution
x <- c(1, 18, 11, 7, 40, 1, 1) #values for a single sample of x

y <- rep(0,1000)
for(i in 1:1000) {
  y[i] = sigmoid(x, beta_samples[i,])
}

#draw random binomial samples given the probabilities
predictions <- rep(0,1000)
for(i in seq_along(y)) {
  p = y[i] #probability
  predictions[i] <- rbinom(1,1,p)
}
barplot(table(predictions), main="Posterior predictive distribution")
#We can see that the prediction is 0 for approximately 80% of the samples, which means it
is
#likely that this woman does not work.
```

**2c)**



The conclusion is that according to the posterior predictive distribution, it seems that 11 women most likely does not work, with high numbers for 8-12 women not working out of 1000 samples from the distribution of Beta.

**Code:**

```
#c) Now, consider 13 women which all have the same features as the woman in (b).
#Rewrite your function and plot the posterior predictive distribution for the number of women,
#out of these 13, that are not working.
#[Hint: Simulate from the binomial distribution, which is the distribution for a sum of Bernoulli
random variables.]
set.seed(123)
n_women <- 13
y <- rep(0,1000)
predictions <- c()

#samples from Beta distribution 1000 times and make predictions. Simulate from binomial
distribution for the 13 women.
for (i in 1:1000) {
  beta_samples <- rmvnorm(n=1000, mean=OptimBeta, sigma=PostCov) #draw samples of
Beta from the distribution
  y[i] <- sigmoid(x, beta_samples[i,]) #probabilities
  predictions <- c(predictions, n_women-rbinom(n=1, size=n_women, y[i])) #simulate from
binomial distribution where each result is prediction of number of women working
}

barplot(table(predictions), main=paste("Post. pre. dist. for", n_women, "women not
working"), xlab="Number of women")
```

```
#The conclusion is that according to the posterior predictive distribution, it seems that 11 women most likely does not work, with high numbers
#for 8-12 women not working.
```

**Appendix:**

**Code Assignment 1:**

```
#The dataset Linkoping2022.xlsx contains daily average temperatures (in degree Celcius)
#in Linköping over the course of the year 2022. Use the function read_xlsx(), which is
included
#in the R package readxl (install.packages("readxl")), to import the dataset in R. The
response variable is
#temp and the covariate time that you need to create yourself is de□ned by

#time =(#days since beginning of year)/365

#A Bayesian analysis of the following quadratic regression model is to be performed:
#temp=β0+β1·time+β2·time +ε,ε~N(0,σ).

#Read data and import library
library(readxl)
data = read_xlsx("Linkoping2022.xlsx")

#Calculate variable time
data$time = as.integer(difftime(data$datetime,"2022-01-01", units = "day"))/365

#a)
# Use the conjugate prior for the linear regression model. The prior hyper- parameters μ0,
Ω0, v0 and σ02
#shall be set to sensible values. Start with μ0 = (0,100,−100)T, Ω0 = 0.01 · I3, v0 = 1 and
σ02 = 1.
#Check if this prior agrees with your prior opinions by simulating draws from the joint prior of
all parameters
#and for every draw compute the regression curve. This gives a collection of regression
curves; one for each draw
#from the prior. Does the collection of curves look reasonable? If not, change the prior
hyperparame- ters until the
#collection of prior regression curves agrees with your prior beliefs about the regression
curve.
#[Hint: R package mvtnorm can be used and your Inv-χ2 simulator of random draws from
Lab 1.]

library(mvtnorm)

### LOAD DATA ###
my_zero = c(0,100,-100)
omega_zero = 0.1*diag(3)
v_zero = 100 #Increased from 1 since we got bad regression curves when using low values
sigma2_zero = 1

#################
```

```
###Simulate draws from prior for sigma squared
Ndraw = 1000 #Integer for how many draws we want to make
set.seed(12345)
chi_vals = rchisq(Ndraw,v_zero) #Drawing 10000 random samples from chi-squared
distribution
sigmaSquared = v_zero*sigma2_zero/chi_vals #Calculating prior sigma squared

###Simulate draws from prior beta
beta=matrix(0,length(sigmaSquared),3) #Create matrix to store beta draws
for (i in 1:length(sigmaSquared)) {
set.seed(12345)
beta[i,] = rmvnorm(1, my_zero, sigmaSquared[i]*solve(omega_zero)) #Drawing beta values
from prior
}

reg = matrix(0,365,Ndraw) #Creatibg matrix to store tge set of regressions
for (i in 1:Ndraw) {
reg[,i]=beta[i,1]+data$time*beta[i,2]+(data$time**2)*beta[i,3] #Calculating regressions
}
###Creating plot where all regressions can be plotted
plot.new()
plot.window(xlim=c(0,1), ylim=c(-10,40))
axis(side = 1)
axis(side = 2)
for (i in 1:Ndraw) {
  lines(data$time, reg[,i], type = "l") #Plottnig regression
}

#Conclusion: By increasing the prior degrees of freedom, we got better regression curves.

#b)
#Write a function that simulate draws from the joint posterior distribu- tion of β0, β1,β2 and
σ2.
#i) Plot a histogram for each marginal posterior of the parameters.
#ii) Make a scatter plot of the temperature data and overlay a curve for the posterior median
of the
#regression function f(time) = E[temp|time] = β0 + β1 · time + β2 · time2, i.e. the median of f
(time)
#is computed for every value of time. In addition, overlay curves for the 90% equal tail
posterior probability
#intervals of f(time), i.e. the 5 and 95 posterior per- centiles of f (time) is computed for every
value of time.
#Does the posterior probability intervals contain most of the data points? Should they?


###DATA###
```

```r
n = length(data$temp)
v_n = v_zero + n
X = cbind(1, data$time, data$time**2)
beta_hat=solve(t(X)%*%X)%*%t(X)%*%data$temp
my_n=solve(t(X)%*%X+omega_zero)%*%(t(X)%*%X%*%beta_hat+omega_zero%*%my_ze
ro)
omega_n = t(X)%*%X+omega_zero
sigma2_n = as.numeric((sigma2_zero*v_zero +
(t(data$temp)%*%data$temp+t(my_zero)%*%omega_zero%*%my_zero-t(my_n)%*%omega
_n%*%my_n))/v_n)
###########

###Draw posterior values for sigma squared
Ndraw = 1000 #Integer for how many draws we want to make
set.seed(12345)
chi_vals = rchisq(Ndraw,v_n) #Drawing 10000 random samples from chi-squared distribution
sigmaSquared = v_n*sigma2_n/chi_vals #Calculating prior sigma squared

###Simulate draws for posterior beta
beta=matrix(0,length(sigmaSquared),3) #Create matrix to store beta simulations
for (i in 1:length(sigmaSquared)) {
  beta[i,] = rmvnorm(1, my_n, sigmaSquared[i]*solve(omega_n)) #Drawing beta values from
posterior
}

###Plotting histogram for posterior draws
hist(sigmaSquared) #Histogram for sigma_squared

hist(beta[,1]) #Histogram for beta_zero

hist(beta[,2]) #Histogram for beta_one

hist(beta[,3]) #Histogram for beta_two

###Scatter plot
plot.new()
plot.window(xlim=c(0,1), ylim=c(-10,30))
axis(side = 1)
axis(side = 2)
points(data$time, data$temp, pch=19)

reg = matrix(0,365,Ndraw) #Creating matrix to store the set of regressions
for (i in 1:Ndraw) {
  reg[,i]=X%*%beta[i,] #Calculating regressions
}

temp_medians = rep(0,dim(reg)[1]) #Creating list to store median values of temps each day
for(i in 1:dim(reg)[1]) {
 temp_medians[i] = median(reg[i,])
}

#Plotting median values
lines(data$time, temp_medians, pch=19, col="red")

#Creating 90% crediible interval
min = 0.05*dim(reg)[2]+1
max = 0.95*dim(reg)[2]

under_limit = rep(0,dim(reg)[1])
upper_limit = rep(0,dim(reg)[1])
for(i in 1:dim(reg)[1]) {
  sorted = sort(reg[i,])
  under_limit[i]=quantile(sorted,probs=0.05)
  upper_limit[i]=quantile(sorted,probs=0.95)
}

lines(data$time, under_limit, col="green")
lines(data$time, upper_limit, col="green")

###Conclusion:The equal tail intervals are very narrow as we can see in the plot. As we saw in the previously
### presented histograms, the posterior draws are narrow which yields a very slim upper and under limit
### (they are very similar, but not the same however)

#c)
#It is of interest to locate the time with the highest expected temperature (i.e. the time where f(time) is maximal).
#Let's call this value x˜. Use the simulated draws in (b) to simulate from the posterior distribution of x˜.
#[Hint: the regression curve is a quadratic polynomial. Given each posterior draw of β0, β1 and β2,
#you can nd a simple formula for x˜.]


#As it is a polynomial function with a maximum point, the time for the highest temp
#should be located where the derivative is 0

###Simulating the expected time where the temperature is at its peak
peak_temp = rep(0,Ndraw)
for (i in 1:Ndraw) {
  peak_temp[i]=(-1)*(beta[i,2]/(2*beta[i,3])) #Estimating time when temperature reaches its peak
}

hist(peak_temp) #Plotting histogram of when the temperature is expected to peak
```

```
###Conclusion: Once again, the posterior distribution is very narrow.
###Furthermore, posterior distribution that temperature is likely to peak in mid July
### which is reasonable

#d)
#Say now that you want to estimate a polynomial regression of order 10, but you suspect that higher
#order terms may not be needed, and you worry about over□tting the data. Suggest a suitable prior that mitigates
#this potential problem. You do not need to compute the posterior.
#Just write down your prior. [Hint: the task is to specify μ0 and Ω0 in a suitable way.]


#Answer:
#The prior my_zero should be 0 (or close to this) and the prior omega_zero should be X*I (X times identity matrix).
#We want a low value for my_zero because we want most coefficients to be close to 0 to obtain shrinkage as
#terms increases in order.
#Furthermore, the X value for the omega_zero should be set to a large value in order to obtain
# a smaller spread for the beta values, making them more likely to stay close to zero.
#On the other hand, you could also argue that the X value should be low as this would give the data
#less impact and therefore decreas the risk of overfitting to the data.
```

**Code Assignment 2:**

# Assignment2.R

**snallfot**

**2023-04-26**

```
# Assignment 2 Posterior approximation for classification with logistic regression
#The dataset WomenAtWork.dat contains n = 132 observations on the following eight variables related to women:

#2.a)
#Consider the logistic regression model: Pr(y = 1|x, β) =  exp(xTβ) / (1 + exp (xT β))
# where y equals 1 if the woman works and 0 if she does not.
#x is a 7-dimensional vector containing the seven features (including a 1 to model the intercept).
#The goal is to approximate the posterior distribution of the parameter vector β with a multivariate normal distribution
```

```
#  β|y, x ~ N(B_hat, J^-1(B_hat)) where B_hat is the posterior mode and J(B_hat) = −∂2
lnp(β|y)|
#  Note that   ∂ β ∂ β T β = B_hat is the negative of ∂2 ln p(β|y) the observed Hessian
evaluated at the posterior mode.
#Note that ∂β∂βT is a 7 × 7 matrix with second derivatives on the diagonal and
cross-derivatives ∂2 ln p(β|y) ∂βi∂βj on the off-diagonal.
#You can compute this derivative by hand, but we will let the computer do it
numerically for you.
#Calculate both B_hat and J(B_hat) by using the optim function in R.
#[Hint: You may use code snippets from my demo of logistic regression in Lecture 6.]
Use the prior β ~ N(0,τ2I), where τ = 2.
#  Present the numerical values of β˜ and J−1(β˜) for the WomenAtWork dat
#a. Compute an approximate 95% equal tail posterior probability interval for the
regression coefficient to the variable NSmallChild.
#Would you say that this feature is of importance for the probability that a woman
works?
#[Hint: You can verify that your estimation results are reasonable by comparing the
posterior means
# to the maximum likelihood estimates, given by: glmModel <- glm(Work ~ 0 + ., data =
WomenAtWork, family = binomial).]


#Import packages
library(mvtnorm)

#Read the data
data <-  (read.table("WomenAtWork.dat", header=TRUE))

Xnames <- names(data[,2:ncol(data)]) #Names of columns
head(data)

##   Work Constant HusbandInc EducYears ExpYears Age NSmallChild NBigChild
## 1    1        1  22.394940        12        7  43           0         3
## 2    0        1   7.232000         8       10  34           0         7
## 3    1        1  18.271990        12        4  41           1         5
## 4    0        1  28.069000        14        2  43           0         2
## 5    1        1   7.799889        12       10  31           0         1
## 6    0        1  28.630000        16        6  37           0         3

n_rows <- dim(data)[1]  #number of observations


LogPostLogistic <- function(betas,y,X,mu,Sigma){
  linPred <- X%*%betas;
  logLik <- sum( linPred*y - log(1 + exp(linPred)) );
  #if (abs(logLik) == Inf) logLik = -20000; # Likelihood is not finite, stear the optimizer
away from here!
  logPrior <- dmvnorm(betas, mu, Sigma, log=TRUE);
  return(logLik + logPrior)
}

#initialize variables
set.seed(123)
y <- as.numeric(data[,1]) #target output
X <- as.matrix(data[,-1]) #training data
X <- apply(X, c(1, 2), as.numeric) #convert from characters to doubles
n_col <- dim(X)[2] #number of columns in training data
tao = 2 #prior std.dev for Beta
initVal <- matrix(0,n_col,1) #initial values for Beta
mu <- as.matrix(rep(0,n_col)) # Prior mean vector for Beta
Sigma <- tao^2*diag(n_col) # Prior covariance matrix for Beta
logPost <- LogPostLogistic; #function to be minimized

OptimRes <-
optim(initVal,logPost,gr=NULL,y,X,mu,Sigma,method=c("BFGS"),control=list(fnscale=-
1),hessian=TRUE)

# Printing the results of mean and variance of normally distributed Beta
OptimBeta <- OptimRes$par
names(OptimBeta) <- Xnames # Naming the coefficient by covariates
PostCov <- solve(-OptimRes$hessian) #get covariance matrix by inverting the
negative hessian
approxPostStd <- sqrt(diag(PostCov)) # Computing approximate standard deviations.
names(approxPostStd) <- Xnames # Naming the coefficient by covariates
print('The posterior mode is:')

## [1] "The posterior mode is:"

print(OptimBeta)

##                  [,1]
## [1,] -0.04036943
## [2,] -0.03730689
## [3,]  0.17868950
## [4,]  0.12073637
## [5,] -0.04618995
## [6,] -1.47248930
## [7,] -0.02014458
## attr(,"names")
## [1] "Constant"     "HusbandInc"  "EducYears"   "ExpYears""Age"
## [6] "NSmallChild" "NBigChild"

print('The approximate posterior standard deviation is:')

## [1] "The approximate posterior standard deviation is:"
```

```
print(approxPostStd)

##     Constant  HusbandInc   EducYears     ExpYears      Age NSmallChild
##  1.38198487  0.02198474  0.08920960  0.03335982  0.02747315  0.47746764
##   NBigChild
##  0.16401959

#Draw samples from distribution of NSmallChild
my <- OptimBeta[6] #mean of Beta[NSmallChild]
sigma <-approxPostStd[6] #std.dev of Beta[NSmallChild]
samples <- rnorm(1000, my, sigma)
quantiles <-  quantile(samples, c(0.025, 0.975))
plot(density(samples), main="Posterior probability interval")
abline(v=quantiles)

print("Upper and lower bounds are:")

## [1] "Upper and lower bounds are:"

print(quantiles)

##       2.5%     97.5%
## -2.3995187 -0.4994643

#Check that estimation results are reasonable
glmModel <- glm(Work ~ 0 + ., data = data, family = binomial)
print(OptimBeta)

##                [,1]
## [1,] -0.04036943
## [2,] -0.03730689
## [3,]  0.17868950
## [4,]  0.12073637
## [5,] -0.04618995
## [6,] -1.47248930
## [7,] -0.02014458
## attr(,"names")
## [1] "Constant"    "HusbandInc"  "EducYears"   "ExpYears"  "Age"
## [6] "NSmallChild" "NBigChild"

print(glmModel$coefficients)

##     Constant  HusbandInc   EducYears     ExpYears      Age NSmallChild
##  0.02262929 -0.03796308  0.18447411  0.12131763 -0.04858167 -1.56485140
##   NBigChild
## -0.02526059

#values are similar, estimation seems reasonable. Since the boundary is negative, it seems like a higher number
#of small children makes a woman less likely to work.
```

```
#b) Use your normal approximation to the posterior from (a).
#Write a function that simulate draws from the posterior predictive distribution of Pr(y
= 0|x),
#where the values of x corresponds to a 40-year-old woman, with two children (4 and
7 years old),
#11 years of education, 7 years of experience, and a husband with an income of 18.
#Plot the posterior predictive distribution of Pr(y = 0|x) for this woman.
#[Hints: The R package mvtnorm will be useful. Remember that Pr(y = 0|x) can be
calculated for each posterior draw of β.]

#sigmoid function
sigmoid <- function(x, beta) {
  s <- beta %*% x
  return (exp(s) / (1+exp(s)))
}

set.seed(123)
PostStd <- c(approxPostStd) #convert to vector
beta_samples <- rmvnorm(n=1000, mean=OptimBeta, sigma=PostCov) #draw samples
of Beta from the distribution
x <- c(1, 18, 11, 7, 40, 1, 1) #values for a single sample of x

y <- rep(0,1000)
for(i in 1:1000) {
  y[i] = sigmoid(x, beta_samples[i,])
}

#draw random binomial samples given the probabilities
predictions <- rep(0,1000)
for(i in seq_along(y)) {
  p = y[i] #probability
  predictions[i] <- rbinom(1,1,p)
}
barplot(table(predictions), main="Posterior predictive distribution")

#We can see that the prediction is 0 for approximately 80% of the samples, which
means it is
#likely that this woman does not work.

#c) Now, consider 13 women which all have the same features as the woman in (b).
#Rewrite your function and plot the posterior predictive distribution for the number of
women,
#out of these 13, that are not working.
#[Hint: Simulate from the binomial distribution, which is the distribution for a sum of
Bernoulli random variables.]
set.seed(123)
n_women <- 13
y <- rep(0,1000)
```

```r
predictions <- c()

#samples from Beta distribution 1000 times and make predictions. Simulate from
binomial distribution for the 13 women.
for (i in 1:1000) {
  beta_samples <- rmvnorm(n=1000, mean=OptimBeta, sigma=PostCov) #draw
samples of Beta from the distribution
  y[i] <- sigmoid(x, beta_samples[i,]) #probabilities
  predictions <- c(predictions, n_women-rbinom(n=1, size=n_women, y[i])) #simulate
from binomial distribution where each result is prediction of number of women
working
}

barplot(table(predictions), main=paste("Post. pre. dist. for", n_women, "women not
working"), xlab="Number of women")

#The conclusion is that according to the posterior predictive distribution, it seems
that 11 women most likely does not work, with high numbers
#for 8-12 women not working.
```