Computer Lab 2

Christian von Koch, William Anzén

2020-04-27

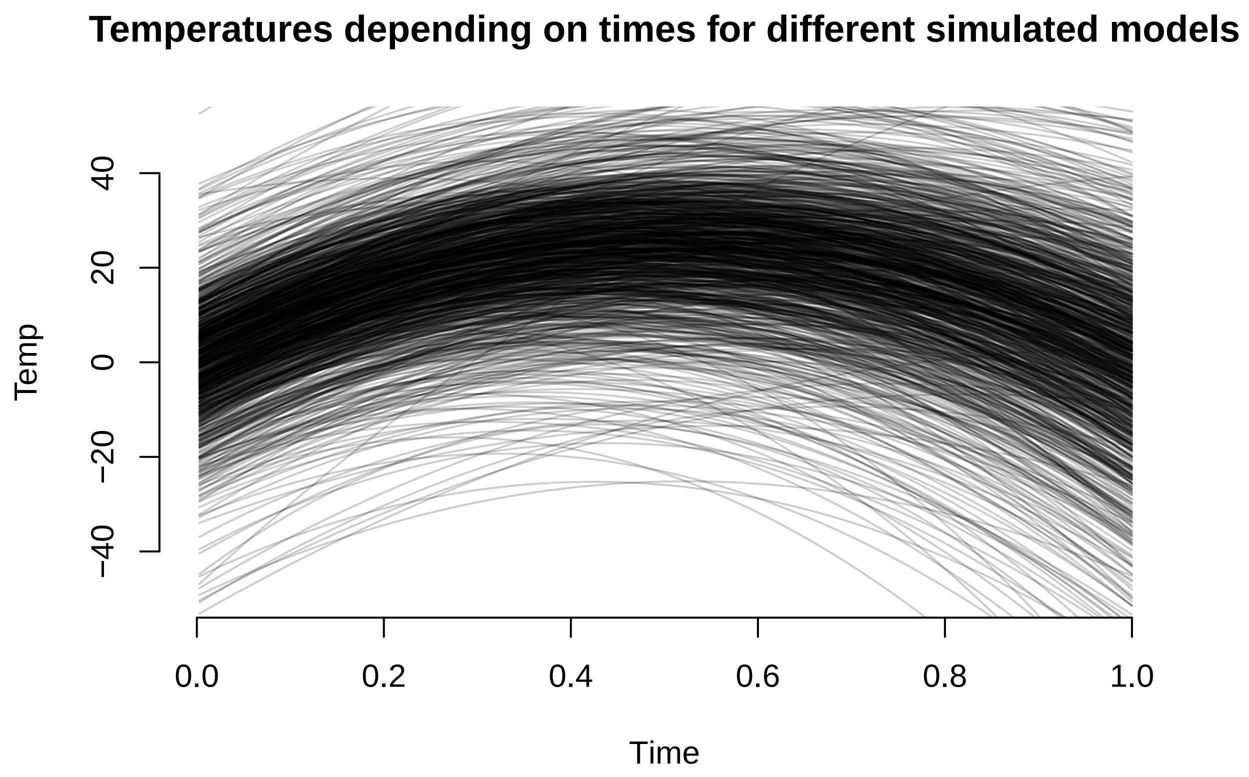
Assignment 1

Task a)

```r
# Read file
temp = read.table("TempLinkoping.txt", header=TRUE)

## install.packages("mvtnorm")
library(mvtnorm)
```

```
## Warning: package 'mvtnorm' was built under R version 3.6.2
```

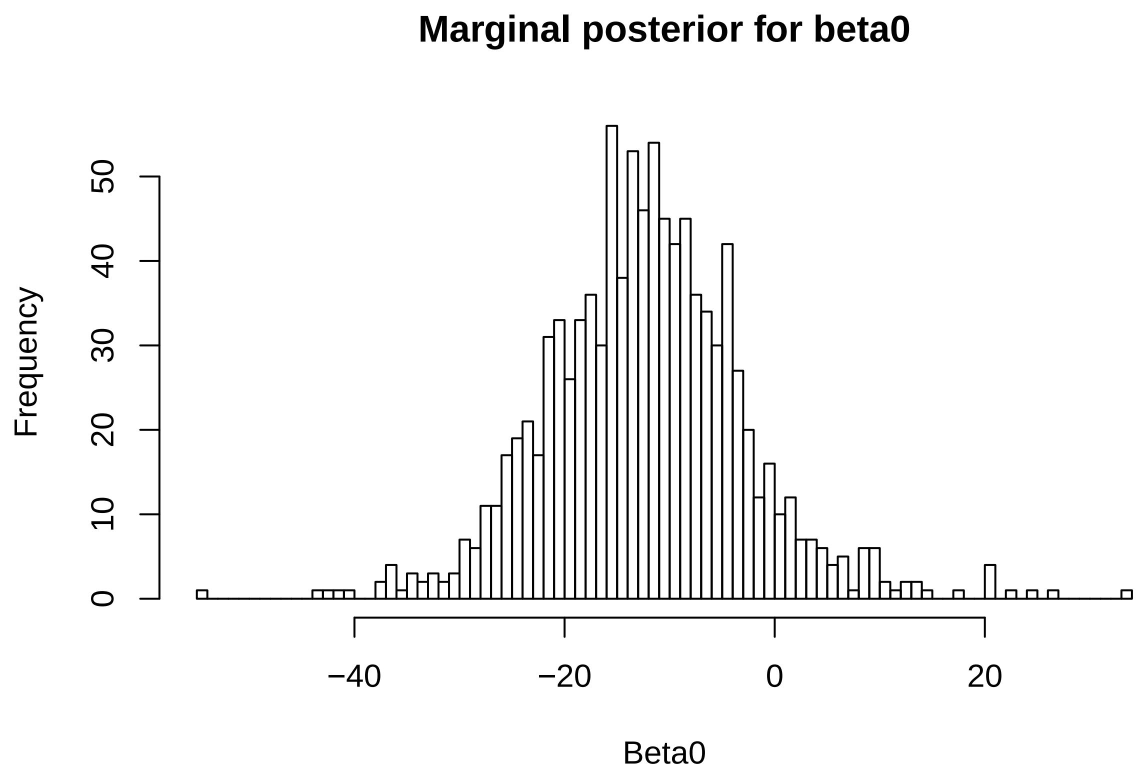
```r
# Defining the parameters for the prior distribution
# Switched to beta0=0 since it seems more reasonable and -10 seems too low.
my0=c(0,100,-100)
omega0=0.01*diag(3)
v0=4
sigma0_sq=1
omega0Inv=solve(omega0)

# Function for returning the response variable
calcRegr = function(betaMatrix, row, x) {
  return(betaMatrix[row,1]+betaMatrix[row,2]*x+betaMatrix[row,3]*x^2)
}

# Function for drawing simulated betavalues
drawBeta = function(my, sigma_sq, omegaInv) {
  return(rmvnorm(1, mean=my, sigma=sigma_sq*omegaInv))
}

nDraws=1000
set.seed(12345)
drawX=rchisq(nDraws, v0)
sigma_sq=(v0)*sigma0_sq/drawX
betaMatrix=matrix(0,nDraws,3)
# Create new plot with specific settings so that the loop can overlay plots
plot.new()
plot.window(xlim=c(0,1), ylim=c(-50,50))
axis(side=1)
axis(side=2)
set.seed(12345)
for (i in 1:nDraws) {
  betaMatrix[i,]=drawBeta(my0, sigma_sq[i], omega0Inv)
  lines(temp$time, calcRegr(betaMatrix, i, temp$time), col=rgb(0,0,0,0.2))
}
title(main="Temperatures depending on times for different simulated models", xlab="Time",
      ylab="Temp")
```



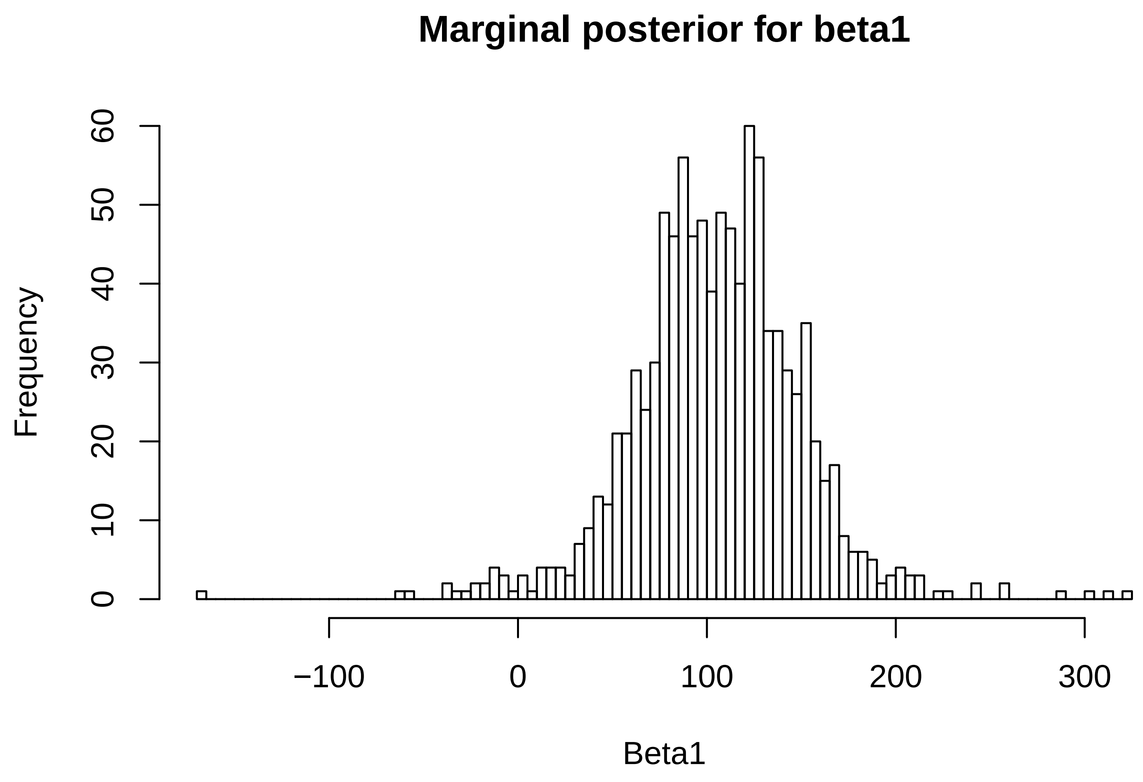
The collection of curves look reasonable and in line with our prior beliefs. The temperature rises during the summer months and stays low in the beginning and the end of the year respectively. However, the value of -10 were switched to 0 since it seems more reasonable with a measurement of the temperature 0 on the 1st of January (close to the intercept) than a measurement of -10.

Task b)

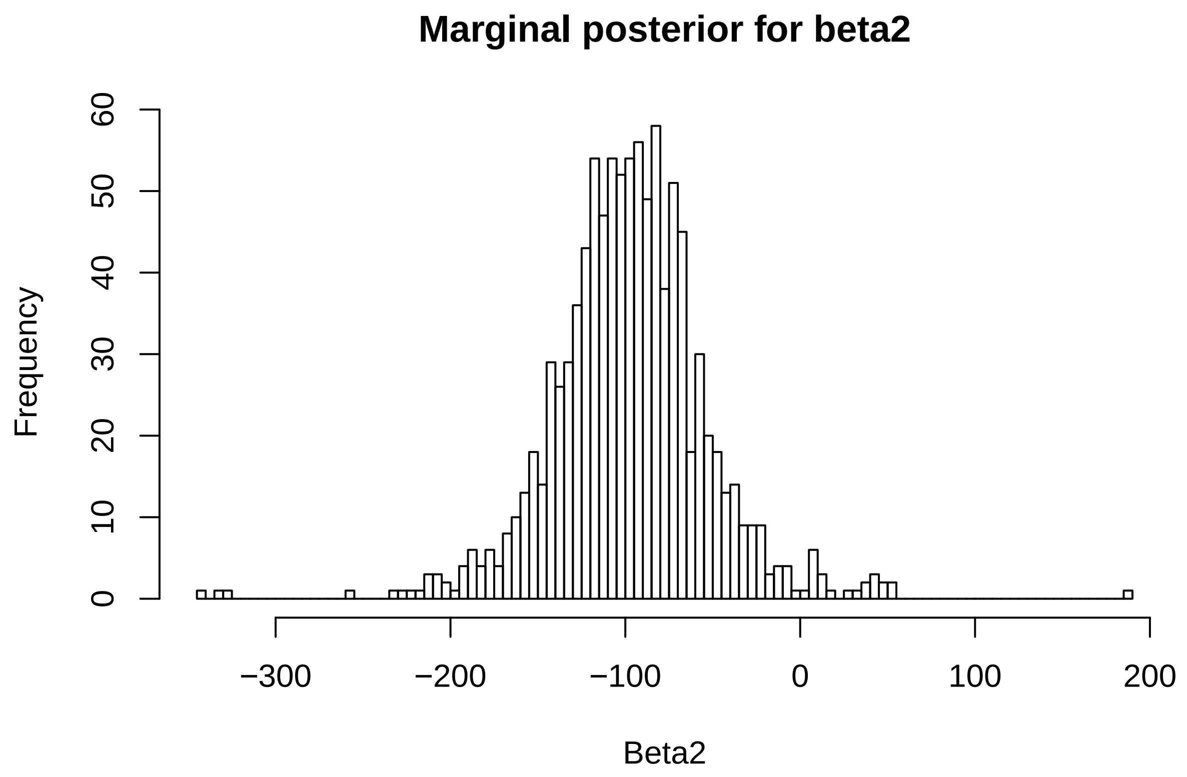
```
# Calculating the parameters for the posterior distribution
v_n=v0+length(temp$temp)
X=cbind(1, temp$time, temp$time^2)
Y=temp$temp
beta_hat=solve(t(X)%*%X)%*%t(X)%*%Y
my_n=solve(t(X)%*%X+omega0)%*%(t(X)%*%X%*%beta_hat+omega0%*%my0)
omega_n=t(X)%*%X+omega0
omega_n_Inv=solve(omega_n)
sigma_sq_n=(v0*sigma0_sq+(t(Y)%*%Y+t(my0)%*%omega0%*%my0-t(my_n)%*%omega_n%*%my_n))/v_n

# Simulate the joint posterior
sigma_sq_post=(v_n)*c(sigma_sq_n)/drawX
betaMatrix_post=matrix(0,nDraws,3)
response_post_temp=matrix(0,nDraws,length(temp$time))
for (i in 1:nDraws) {
  betaMatrix_post[i,]=drawBeta(my_n, sigma_sq_post[i], omega_n_Inv)
}
# Plots the marginal distributions for the different beta-values
hist(betaMatrix_post[,1], breaks=100, main="Marginal posterior for beta0", xlab="Beta0")
```



```r
hist(betaMatrix_post[,2], breaks=100, main="Marginal posterior for beta1", xlab="Beta1")
```

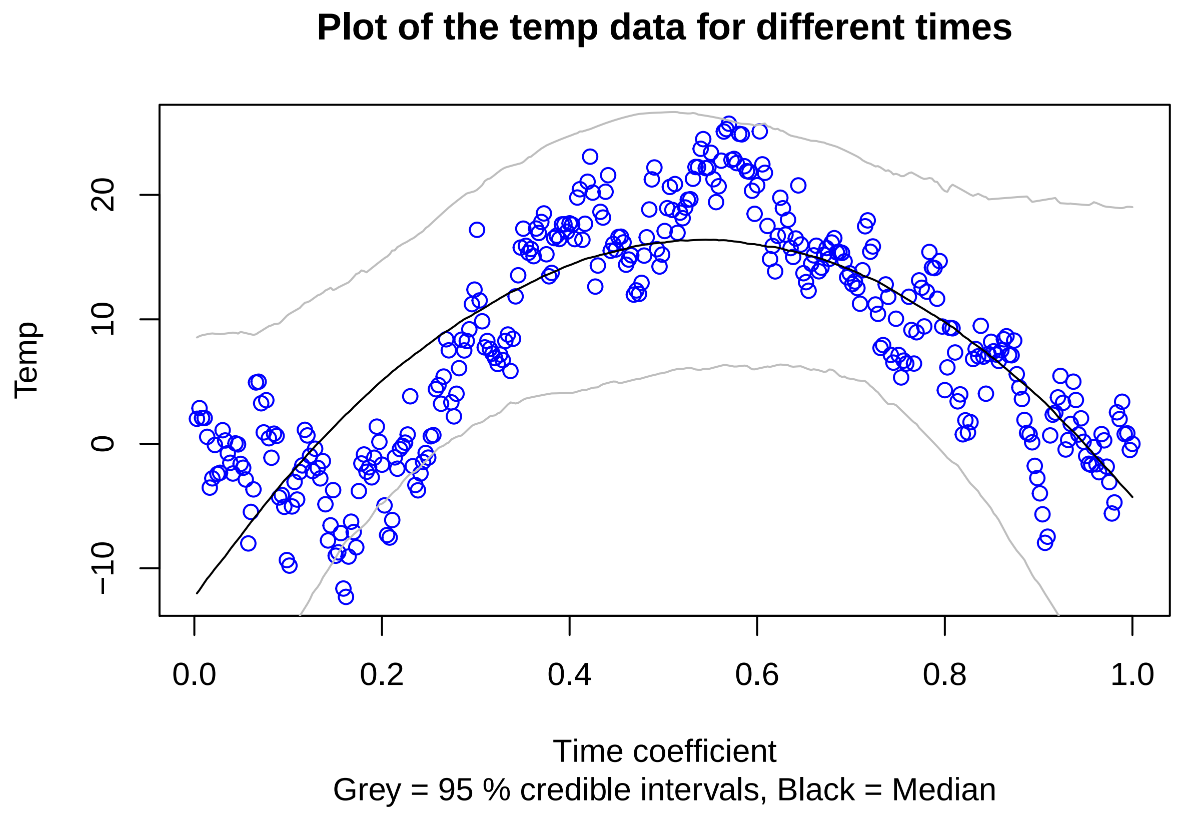


```r
hist(betaMatrix_post[,3], breaks=100, main="Marginal posterior for beta2", xlab="Beta2")
```



As seen in the plots for the marginal posterior of β_1 and β_2 the maximum density of the distributions are located close to the prior beliefs for these parameters. However, for β_0 (the parameter value which were changed in the prior to 0 instead of -10) we can see that after using the data available the distribution has moved towards -10 which indicates that the data suggests that this value for the parameter is more reasonable than 0.

```
plot(temp$time, Y, main="Plot of the temp data for different times", col="blue",
     xlab="Time coefficient", ylab="Temp")
# Applies function calcRegr to the time-values for each of the drawn betas and stores the
# results in matrix
for (i in 1:nDraws) {
  betaTemp=sapply(temp$time, calcRegr, betaMatrix=betaMatrix_post, row=i)
  response_post_temp[i,]=betaTemp
}

response_post=c()
credInterval=matrix(0, length(temp$time), 2)
# Retrieves the median of the response values as well as obtaining the upper and lower
# bound of credInterval
for (i in 1:length(temp$time)) {
  sortedTemp=sort(response_post_temp[,i])
  response_post=c(response_post, (sortedTemp[500]+sortedTemp[501])/2)
  credInterval[i,]=c(sortedTemp[(0.025*length(sortedTemp)+1)],
                     sortedTemp[(0.975*length(sortedTemp))])
}

lines(temp$time, response_post)
lines(temp$time, credInterval[,1], col="gray")
lines(temp$time, credInterval[,2], col="gray")
title(sub="Grey = 95 % credible intervals, Black = Median")
```

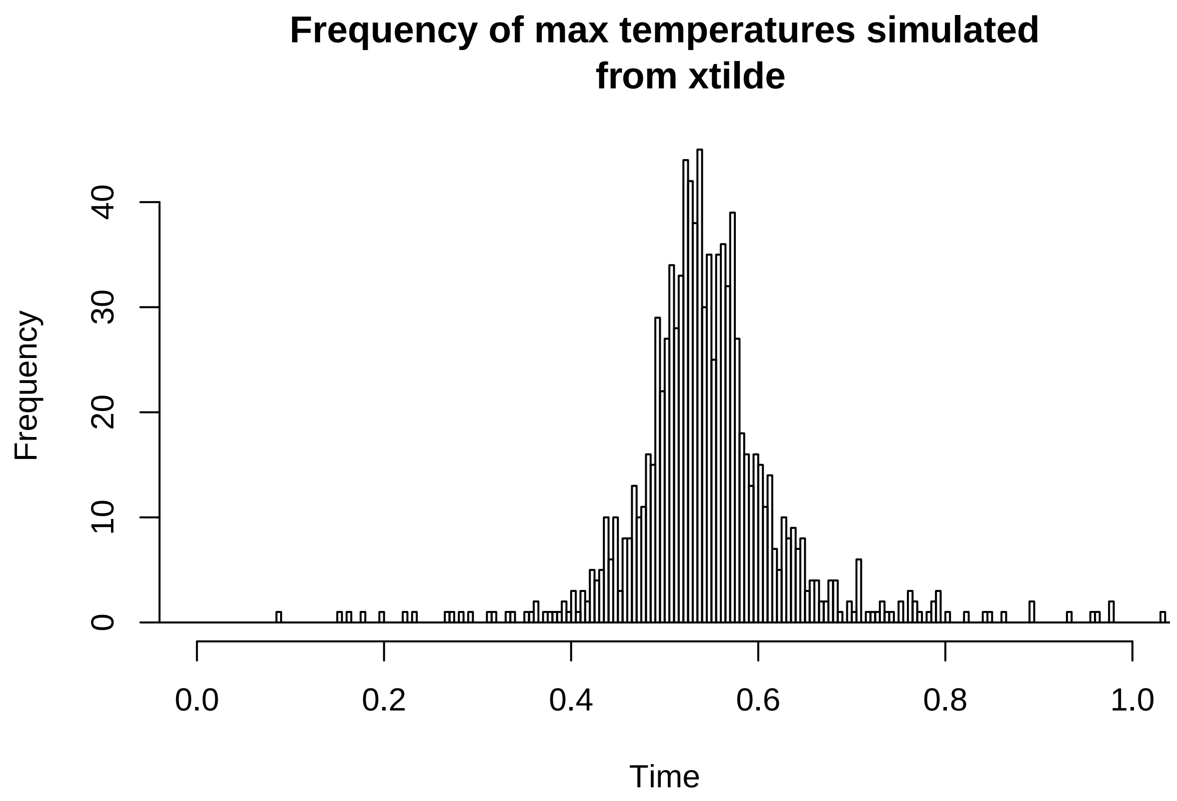


As seen in the plot the black line changes in a way similar to the trend shown by the blue points. This is reasonable since the black line represents the median value of the different simulated outcomes from the models for all times. The gray dashed lines represent the upper and lower bounds of the 95 % credible interval for all of the different time-values. The interval bands contain most of the data points as seen in the plot. They should contain most of the data points if the model is accurate in terms of describing the reality. In this case, it seems like the model has captured most of the data points which means that the model describes the reality fairly well. You would also want the confidence intervals to be as tight as possible to receive the most accurate and informative answer to a question (for example what kind of temperatures might we expect in March). In this case the margin between the outer most points and the confidence intervals are quite tight which suggests that the confidence intervals are quite informative.

Task c)

```
# Function for calculating the time-value which yields the maximum response (the derivative
# of response function)
calcMaxTemp = function(betaMatrix, row) {
  return(-betaMatrix[row,2]/(2*betaMatrix[row,3]))
}

# For each of the draws the time-value which yields the maximum temperature is stored in
# a vector
time_max_temp=c()
```

```r
for (i in 1:nDraws) {
  time_max_temp=c(time_max_temp, calcMaxTemp(betaMatrix_post, i))
}

hist(time_max_temp, breaks=1000, xlim=c(0,1), main="Frequency of max temperatures simulated
     from xtilde", xlab="Time")
```



As seen in the plot the maximum density can be found approximately from 0.55 to 0.6 which corresponds to late June. This seems like a reasonable time of year where we can expect the highest temperatures in Malmslätt, Linköping.

Task d)

A suitable prior for this task would be to set μ_0 to 0 since you want most of the coefficients close to zero to obtain increased shrinkage. You would also want to set Ω_0 to $\lambda * I$. This would mean that for larger values of λ more and more of the β-values would be close to 0 since the spread of the distribution of the β-values would decrease. In this case, where there is a worry about overfitting, it might be a good idea to choose a large value of λ to decrease the spread of the β-values and increase the probability that most of the β-values are around 0.

Assignment 2

Task a)

```r
# Use of libraries
library(mvtnorm)
```

```r
# Read data
WomenWork = read.table("WomenWork.dat", header=TRUE)

# User input
tau = 10

# Defining vectors X and Y
X = as.matrix(WomenWork[,2:ncol(WomenWork)])
Y = WomenWork[,1]
nFeatures = dim(X)[2]
covNames=names(WomenWork[,2:ncol(WomenWork)])

# Constructing prior
mu_prior = rep(0,nFeatures)
sigma_prior = tau^2*diag(nFeatures)

# Defining function for returning the log posterior

logPostLogistic = function(beta, Y, X, mu, sigma) {
  nFeat = length(beta)
  XBeta=X%*%beta
  # Defining loglikelihood
  logLike = sum(Y*XBeta-log(1+exp(XBeta)))
  # Defining prior
  prior = dmvnorm(beta, mean=mu, sigma=sigma, log=TRUE)
  # Adding loglikelihood and logprior together. Since it is log both of them are added
  # instead of multiplied
  return(logLike + prior)
}

# Defining initial values to be passed on to the optimizer
set.seed(12345)
initVals = rnorm(dim(X)[2])

# Finding the optimized betavector
optimResult = optim(initVals, logPostLogistic, Y=Y, X=X, mu=mu_prior, sigma=sigma_prior,
                    method=c("BFGS"), control=list(fnscale=-1), hessian=TRUE)

# Defining the values of interest
postMode = optimResult$par
postCov = -solve(optimResult$hessian)
names(postMode) = covNames
approx_PostStd = sqrt(diag(postCov))
names(approx_PostStd) = covNames
print("The posterior mode is:")
```

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

```r
print(postMode)
```

```
##    Constant  HusbandInc   EducYears    ExpYears   ExpYears2         Age
##  0.62677654 -0.01979171  0.18022304  0.16756764 -0.14460934 -0.08206690
## NSmallChild   NBigChild
## -1.35915450 -0.02469156
```

```
print("The approximated standard deviations are:")
```

```
## [1] "The approximated standard deviations are:"
```

```
print(approx_PostStd)
```

```
##    Constant  HusbandInc   EducYears    ExpYears   ExpYears2         Age
##  1.50533202  0.01589982  0.07885554  0.06596694  0.23574723  0.02680419
## NSmallChild   NBigChild
##  0.38892571  0.14132336
```

Above the approximated posterior mode as well as the approximated standard deviations for the different covariates are presented.

```
# Compute marginal distribution for nSmallChild
NSmallChild_mode = as.numeric(postMode["NSmallChild"])
NSmallChild_std = as.numeric(approx_PostStd["NSmallChild"])
set.seed(12345)
NSmallChild_distrib = rnorm(10000, mean=NSmallChild_mode, sd=NSmallChild_std)
sorted_NSmallChild=sort(NSmallChild_distrib)
credInterval_NSmallChild = c(sorted_NSmallChild[(0.025*length(sorted_NSmallChild)+1)],
                             sorted_NSmallChild[(0.975*length(sorted_NSmallChild))])
print(paste("The lower bound of the 95 % credible interval for the feature NSmallChild is",
            round(credInterval_NSmallChild[1], 6)))
```

```
## [1] "The lower bound of the 95 % credible interval for the feature NSmallChild is -2.132527"
```

```
print(paste( "The upper bound of the 95 % credible interval for the feature NSmallChild is",
            round(credInterval_NSmallChild[2], 6)))
```

```
## [1] "The upper bound of the 95 % credible interval for the feature NSmallChild is -0.596241"
```

```
# Verify that the calculations have been made correctly
glmModel = glm(Work ~ 0+., data=WomenWork, family=binomial)
print("The coefficients for the glmmodel using maximum likelihood are:")
```

```
## [1] "The coefficients for the glmmodel using maximum likelihood are:"
```

```
print(glmModel$coefficients)
```

```
##    Constant  HusbandInc   EducYears    ExpYears   ExpYears2         Age
##  0.64430363 -0.01977457  0.17988062  0.16751274 -0.14435946 -0.08234033
## NSmallChild   NBigChild
## -1.36250239 -0.02542986
```

```
print("The coefficients calculated by the optimizer function with the approximated distribution are:")
```

```
## [1] "The coefficients calculated by the optimizer function with the approximated distribution are:"
```

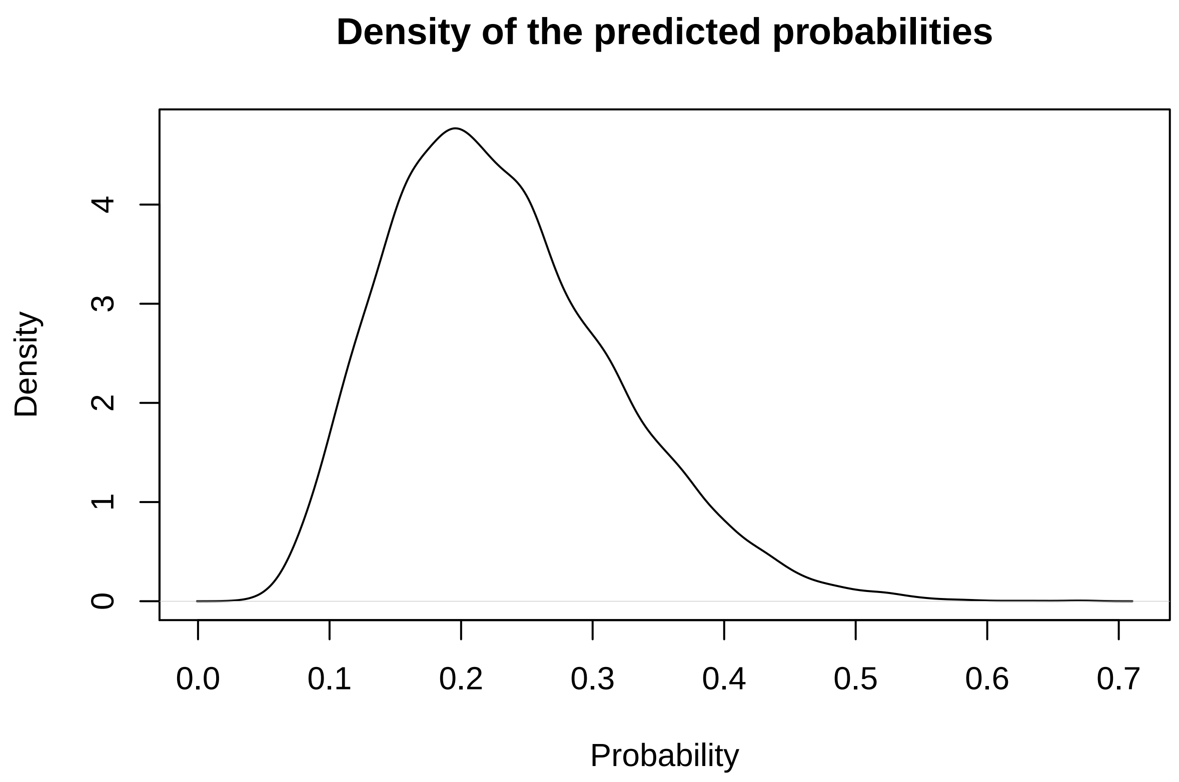
```
print(postMode)
```

```
##    Constant  HusbandInc   EducYears    ExpYears   ExpYears2         Age
##  0.62677654 -0.01979171  0.18022304  0.16756764 -0.14460934 -0.08206690
## NSmallChild   NBigChild
## -1.35915450 -0.02469156
```

Since the values for the credible interval for NSmallChild are quite large in the negative direction it is reasonable to conclude that the feature NSmallChild affects the response variable farily much towards the response 0 which means that the woman doesn't work. This seems like a reasonable conclusion in terms of how it is in reality as well (since when a family has small children the women are usually home taking care of the children). When checking if the results are reasonable, a comparison was made with an estimation using the maximum likelihood method through the glmmodel. The results was very similar which strongly suggests that the results obtained from the code are reasonable.

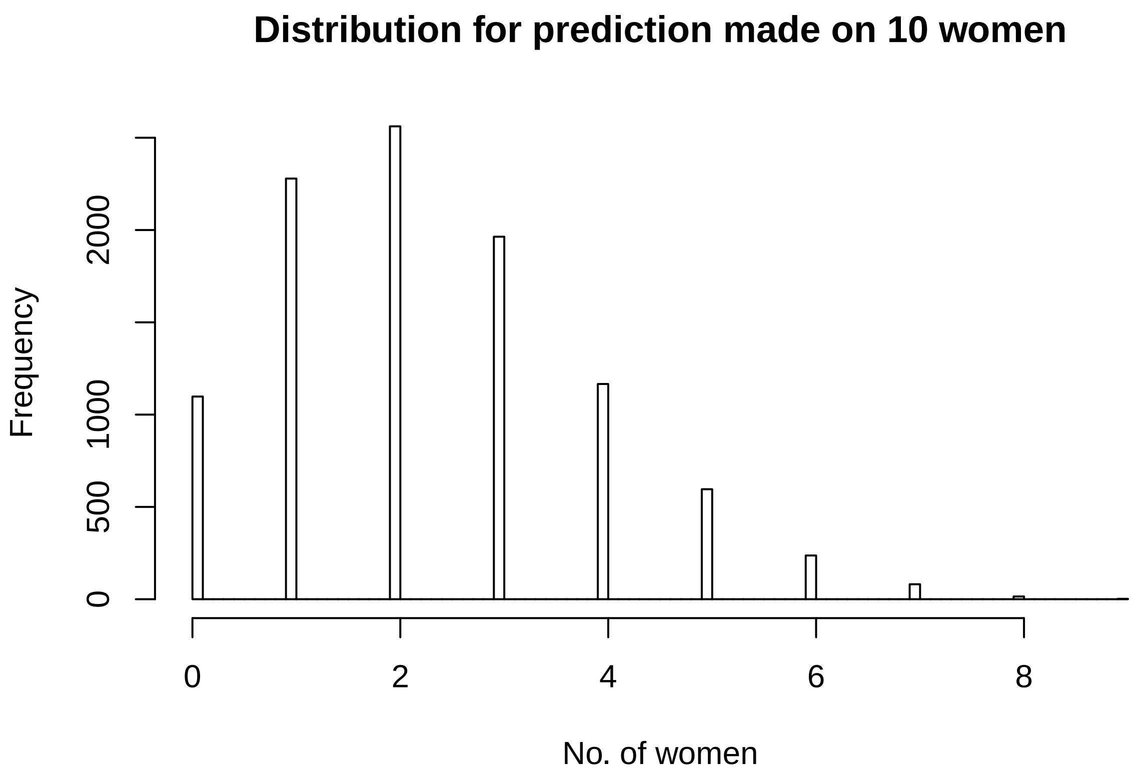
Task b)

```r
sigmoid = function(value) {
  return (exp(value)/(1+exp(value)))
}

makePredLogReg = function(data, mean, sigma, nDraws) {
  betaPred = rmvnorm(nDraws, mean=mean, sigma=sigma)
  linearPred = betaPred %*% data
  logPred = sigmoid(linearPred)
  return(logPred)
}

nDraws=10000
woman=c(1, 10, 8, 10, (10/10)^2, 40, 1, 1)
set.seed(12345)
womanWorkPred=makePredLogReg(woman, postMode, postCov, nDraws)
plot(density(womanWorkPred), main="Density of the predicted probabilities",
     xlab="Probability", ylab="Density")
```



As seen in the plots the calculated probabilities of the woman in question working is fairly low. The highest density is seen in the range between 0.2 and 0.3 approximately. If the classification of the response variable results in "working" if the predicted probabilities is above 0.5 and "not working" otherwise, it is clear from the distribution that the classification of a woman working, with the parameters inputted, is very unlikely to happen according to the predictive distribution.

```
makePredLogRegMultiple = function(data, mean, sigma, nDraws, n) {
  multiplePred=c()
  for (i in 1:nDraws) {
    betaDraw = makePredLogReg(data, mean, sigma, 1)
    multiplePred=c(multiplePred, rbinom(1, n, betaDraw))
  }
  hist(multiplePred, breaks=100, main=paste("Distribution for prediction made on", n, "women"),
       xlab="No. of women")
}

makePredLogRegMultiple(woman, postMode, postCov, 10000, 10)
```



As seen in the histogram the binomial case resembles the density of predicted probabilities with the highest density found at 2 women. This result seems reasonable since when the number of draws taken from the binomial distribution goes towards infinity the shape of the corresponding distribution will resemble the shape of the distribution for the probability p in the Bernoulli case, more and more.

Appendix

```r
## Assignment 1: The dataset TempLinkoping.txt contains daily average tamperatures
## (in Celcius degrees) at Malmslatt, Linkoping over the course of the year 2018.
## The response variable is temp and the covariate istime=(the number of days since
## beginning of year)/365 You're task is to perform a Bayesian analysis of a
## quadratic regression temp=beta0+beta1*time+beta2*time^2+epsilon, epsilon~N(0,sigma^2)

## a) Determining the prior distribution of the model parameters. Use the conjugate
## prior for the linear  regression model. Your task is to set the prior hyperparameters
## my0, omega0, v0 and sigma0^2 to sensible  values. Start with
## my0=(-10,100,-100)T, omega0=0.01*I3, v0=4 and sigma0^2=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. Do the collection of curves
## look reasonable? If not, change the prior hyperparameters until the collection
## of prior regression curves agrees with your prior beliefs about the regression curve.
## [Hint: the R package mvtnorm will be handy. And use your Inv-chisquared simulator
## from Lab1.

# Read file
temp = read.table("TempLinkoping.txt", header=TRUE)

## install.packages("mvtnorm")
library(mvtnorm)
# Defining the parameters for the prior distribution
# Switched to beta0=0 since it seems more reasonable and -10 seems too low.
my0=c(0,100,-100)
omega0=0.01*diag(3)
v0=4
sigma0_sq=1
omega0Inv=solve(omega0)

# Function for returning the response variable
calcRegr = function(betaMatrix, row, x) {
  return(betaMatrix[row,1]+betaMatrix[row,2]*x+betaMatrix[row,3]*x^2)
}

# Function for drawing simulated betavalues
drawBeta = function(my, sigma_sq, omegaInv) {
  return(rmvnorm(1, mean=my, sigma=sigma_sq*omegaInv))
}

nDraws=1000
set.seed(12345)
drawX=rchisq(nDraws, v0)
sigma_sq=(v0)*sigma0_sq/drawX
betaMatrix=matrix(0,nDraws,3)
# Create new plot with specific settings so that the loop can overlay plots
plot.new()
plot.window(xlim=c(0,1), ylim=c(-50,50))
axis(side=1)
axis(side=2)
set.seed(12345)
```

```r
for (i in 1:nDraws) {
  betaMatrix[i,]=drawBeta(my0, sigma_sq[i], omega0Inv)
  lines(temp$time, calcRegr(betaMatrix, i, temp$time), col=rgb(0,0,0,0.2))
}
title(main="Temps depending on different times for different simulated models",
      xlab="Time", ylab="Temp")

## b) Write a program that simulates from the joint posterior distribution of beta0, beta1,
## beta2 and sigma^2. Plot the marginal posteriors of each parameter as a histogram. Also
## produce another figure with a scatter plot of the temperature data and overlay a curve
## for the posterior median of the regression function
## f(time)=beta0+beta1*time+beta2*time^2, computed for every value of time. Also overlay
## curves for the lower 2.5% and upper 97.5% posterior credible interval for f(time). That
## is, compute the 95% equal tail posterior probability intervals for every value of time
## and then connect the lower and upper limits of the interval by curves. Does the interval
## bands contain most of the data points? Should they?

# Calculating the parameters for the posterior distribution
v_n=v0+length(temp$temp)
X=cbind(1, temp$time, temp$time^2)
Y=temp$temp
beta_hat=solve(t(X)%*%X)%*%t(X)%*%Y
my_n=solve(t(X)%*%X+omega0)%*%(t(X)%*%X%*%beta_hat+omega0%*%my0)
omega_n=t(X)%*%X+omega0
omega_n_Inv=solve(omega_n)
sigma_sq_n=(v0*sigma0_sq+(t(Y)%*%Y+t(my0)%*%omega0%*%my0-t(my_n)%*%omega_n%*%my_n))/v_n

# Simulate the joint posterior
sigma_sq_post=(v_n)*c(sigma_sq_n)/drawX
betaMatrix_post=matrix(0,nDraws,3)
response_post_temp=matrix(0,nDraws,length(temp$time))
for (i in 1:nDraws) {
  betaMatrix_post[i,]=drawBeta(my_n, sigma_sq_post[i], omega_n_Inv)
}
# Plots the marginal distributions for the different beta-values
hist(betaMatrix_post[,1], breaks=100, main="Marginal posterior for beta0")
hist(betaMatrix_post[,2], breaks=100, main="Marginal posterior for beta1")
hist(betaMatrix_post[,3], breaks=100, main="Marginal posterior for beta2")

plot(temp$time, Y, main="Plot of the temp data for different times", col="blue",
     xlab="Time coefficient", ylab="Temp")
# Applies function calcRegr to the time-values for each of the drawn betas and stores the
# results in matrix
for (i in 1:nDraws) {
  betaTemp=sapply(temp$time, calcRegr, betaMatrix=betaMatrix_post, row=i)
  response_post_temp[i,]=betaTemp
}

response_post=c()
credInterval=matrix(0, length(temp$time), 2)
# Retrieves the median of the response values as well as obtaining the upper and lower
# bound of credInterval
for (i in 1:length(temp$time)) {
  sortedTemp=sort(response_post_temp[,i])
  response_post=c(response_post, (sortedTemp[500]+sortedTemp[501])/2)
  credInterval[i,]=c(sortedTemp[(0.025*length(sortedTemp)+1)], sortedTemp[(0.975*length(sortedTemp))])
}

lines(temp$time, response_post)
lines(temp$time, credInterval[,1], lty=21, col="gray")
lines(temp$time, credInterval[,2], lty=21, col="gray")
title(sub="Grey = 95 % credible intervals, Black = Median")

## c) It is of interest to locate the time with the highest expected temperature
## (that is, the time where f(time) is maximal). Let's call this value xtilde. Use
## the simulations in b) to simulate from posterior distribution of xtilde.
## [Hint: The regression curve is quadratic. You can find a simple formula for xtilde
## given beta0, beta1 and beta2]

# Function for calculating the time-value which yields the maximum response (the derivative
# of response function)
calcMaxTemp = function(betaMatrix, row) {
  return(-betaMatrix[row,2]/(2*betaMatrix[row,3]))
}

# For each of the draws the time-value which yields the maximum temperature is stored in a vector
time_max_temp=c()
for (i in 1:nDraws) {
  time_max_temp=c(time_max_temp, calcMaxTemp(betaMatrix_post, i))
}

hist(time_max_temp, breaks=1000, xlim=c(0,1),
     main="Frequency of max temperatures simulated from xtilde", xlab="Temperature")

## d) Say now that you want to estimate a polynomial model of order 7, but you suspect
## that higher order terms may not be needed, and you worry about overfitting. 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 my0 and
## omega0 in a smart way.]

## Assignment 2:  Consider the logistic regression Pr(y=1 given x)=exp(xT*Beta)/(1+exp(xT*Beta))
## , where y is the binary variable with y = 1 if the woman works and y = 0 if she does not.
## x is a 8-dimensional vector containing the eight features (including a one for the
## constant term that models the intercept). The goal is to approximate the posterior
## distribution of the 8-dim parameter vector beta with multivariate normal distribution.
## Beta given y and x ~ N(~Beta, JY(~Beta)^(-1)), where ~Beta is the posterior mode and
## J(~Beta) is the second derivative, the observed Hessian evaluated at the posterior mode.
## It is actually not hard to compute this derivative by hand, but don't worry, we will
## let the computer do it numerically for you. Now, both ~Beta and J(~Beta) are computed
## by the optim function in R. I want you to implement an own version of my example code
## at the website. You can use my code as a template, but I want you to write your own file
## so that you understand every line of your code. Don't just copy my code. Use the prior
## Beta ~ N(0, thao^2*I) with thao=10. Your report should include your code as well as
## numerical values for ~Beta and JY(~Beta)^(-1) for the WomenWork data. Compute an
## approximate 95% credible interval for the variable NSmallChild. Would you say that this
## feature is an important determinant of the probability that a women works?
## [Hint: To verify that your results are reasonable, you can compare to you get by estimating
## the parameters using maximum likelihood.
## glmmodel = glm(Work~0+., data=WomenWork, family=binomial)

# Use of libraries
library(mvtnorm)

# Read data
WomenWork = read.table("WomenWork.dat", header=TRUE)

# User input
tau = 10

# Defining vectors X and Y
X = as.matrix(WomenWork[,2:ncol(WomenWork)])
Y = WomenWork[,1]
nFeatures = dim(X)[2]
covNames=names(WomenWork[,2:ncol(WomenWork)])

# Constructing prior
mu_prior = rep(0,nFeatures)
sigma_prior = tau^2*diag(nFeatures)

# Defining function for returning the log posterior

logPostLogistic = function(beta, Y, X, mu, sigma) {
  nFeat = length(beta)
  XBeta=X%*%beta
  # Defining loglikelihood
  logLike = sum(Y*XBeta-log(1+exp(XBeta)))
  # Defining prior
  prior = dmvnorm(beta, mean=mu, sigma=sigma, log=TRUE)
  # Adding loglikelihood and logprior together. Since it is log both of them are added
  # instead of multiplied
  return(logLike + prior)
}

# Defining initial values to be passed on to the optimizer
set.seed(12345)
initVals = rnorm(dim(X)[2])

# Finding the optimized betavector
optimResult = optim(initVals, logPostLogistic, Y=Y, X=X, mu=mu_prior, sigma=sigma_prior,
                    method=c("BFGS"), control=list(fnscale=-1), hessian=TRUE)

# Defining the values of interest
postMode = optimResult$par
postCov = -solve(optimResult$hessian)
names(postMode) = covNames
approx_PostStd = sqrt(diag(postCov))
names(approx_PostStd) = covNames
print("The posterior mode is:")
print(postMode)
print("The approximated standard deviations are:")
print(approx_PostStd)

# Compute marginal distribution for nSmallChild
NSmallChild_mode = as.numeric(postMode["NSmallChild"])
NSmallChild_std = as.numeric(approx_PostStd["NSmallChild"])
set.seed(12345)
NSmallChild_distrib = rnorm(10000, mean=NSmallChild_mode, sd=NSmallChild_std)
sorted_NSmallChild=sort(NSmallChild_distrib)
credInterval_NSmallChild = c(sorted_NSmallChild[(0.025*length(sorted_NSmallChild)+1)],
                             sorted_NSmallChild[(0.975*length(sorted_NSmallChild))])
print(paste("The lower bound of the 95 % credible interval for the feature NSmallChild is",
            round(credInterval_NSmallChild[1], 6), "and the upper bound is",
            round(credInterval_NSmallChild[2], 6)))

# Control that the calculations have been made correctly
glmModel = glm(Work ~ 0+., data=WomenWork, family=binomial)
print(glmModel$coefficients)
print(postMode)

## b) Write a function that simulates from the predictive distribution of the
## response variable in a logistic regression. Use your normal approximation from
## 2(a). Use that function to simulate and plot the predictive distribution for
## the Work variable for a 40 year old woman, with two children (3 and 9 years old)
## , 8 years of education, 10 years of experience. and a husband with an income of 10.
## [Hints: The R package mvtnorm will gain be handy. Remember my discussion on how
## Bayesian prediction can be done by simulation.]

sigmoid = function(value) {
  return (exp(value)/(1+exp(value)))
}

makePredLogReg = function(data, mean, sigma, nDraws) {
  betaPred = rmvnorm(nDraws, mean=mean, sigma=sigma)
  linearPred = betaPred %*% data
  logPred = sigmoid(linearPred)
  return(logPred)
}

nDraws=10000
woman=c(1, 10, 8, 10, (10/10)^2, 40, 1, 1)
set.seed(12345)
womanWorkPred=makePredLogReg(woman, postMode, postCov, nDraws)
hist(womanWorkPred, breaks=100, main="Histogram of the predicted probabilities")
plot(density(womanWorkPred), main="Density of the predicted probabilities",
     xlab="Probability", ylab="Density")

## c) Now, consider 10 women which all have the same features as the woman in 2(b).
## Rewrite your function and plot the predictive distribution for the number of women,
## out of these 10, that are working. [Hint: Which distribution can be described as a
## sum of Bernoulli random variables?]

makePredLogRegMultiple = function(data, mean, sigma, nDraws, n) {
  multiplePred=c()
  for (i in 1:nDraws) {
    betaDraw = makePredLogReg(data, mean, sigma, 1)
    multiplePred=c(multiplePred, rbinom(1, n, betaDraw))
  }
  hist(multiplePred, breaks=100, main=paste("Distribution for prediction made on", n, "women"),
       xlab="No. of women")
}

makePredLogRegMultiple(woman, postMode, postCov, 10000, 10)
```