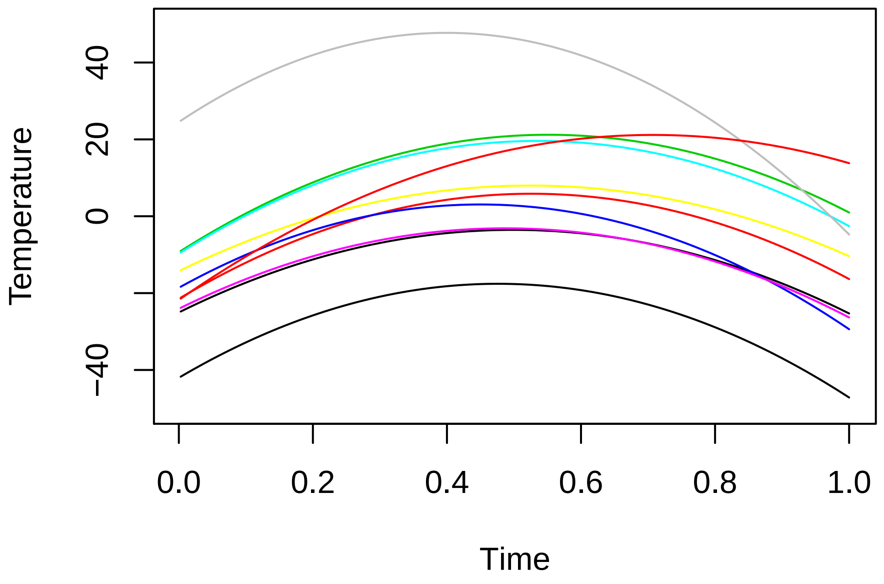
732A91 Bayesian Learning- Computer Lab 2

Namita Sharma, Aman Kumar Nayak

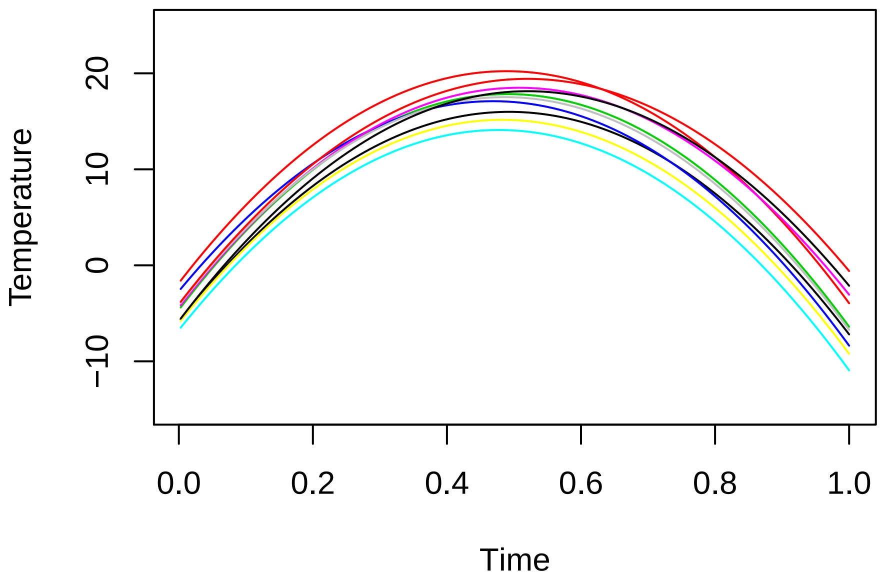
4/27/2020

1. Linear and polynomial regression

(a) Determining the prior distribution of the model parameters

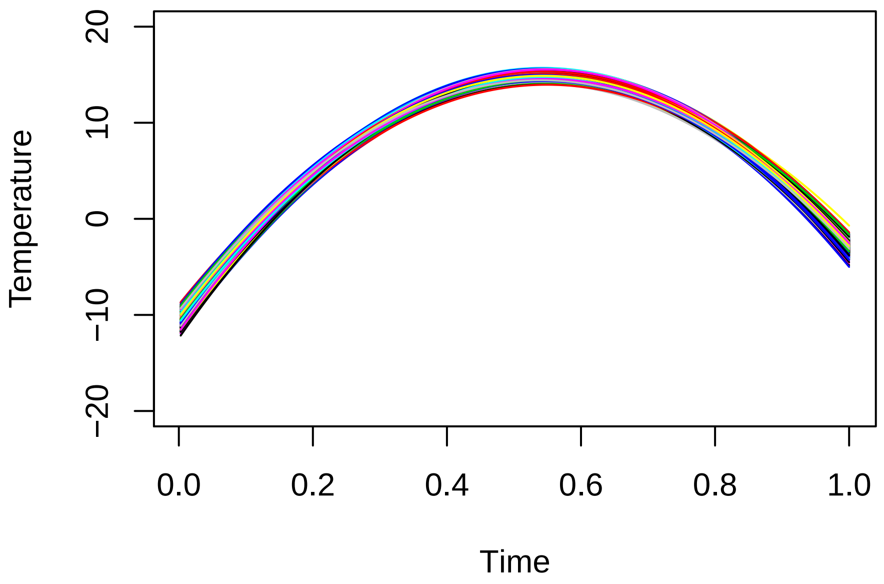


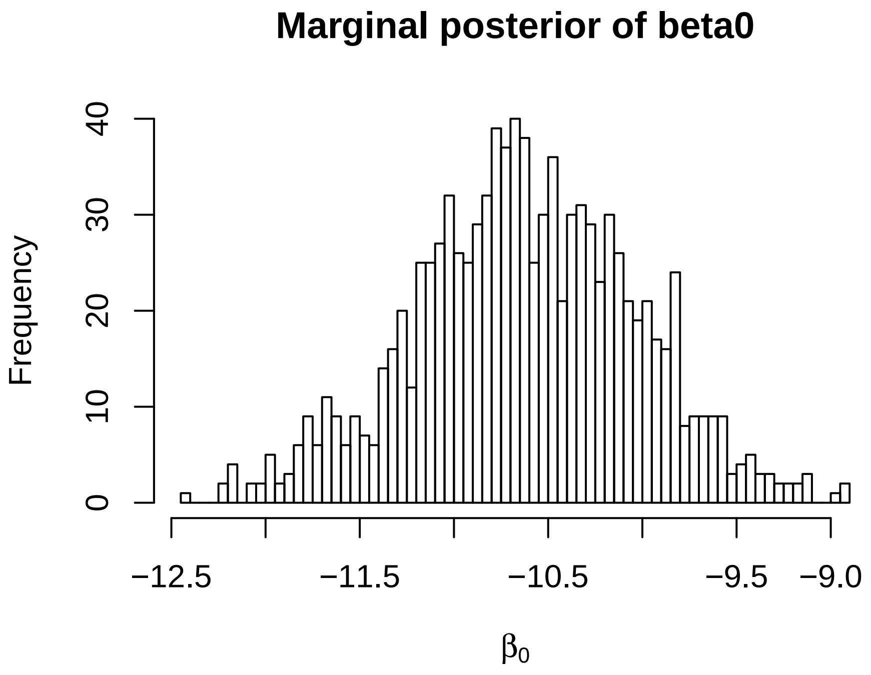
We can see that the predicted temperatures have high variance and do not align with our beliefs about the temperatures in linkoping. Based on the observed temperatures in the past in Linkoping, the extreme high and low temperatures during winters is approx. 10 and -10 respectively. Similarly, during summers they are about 25 and 10.

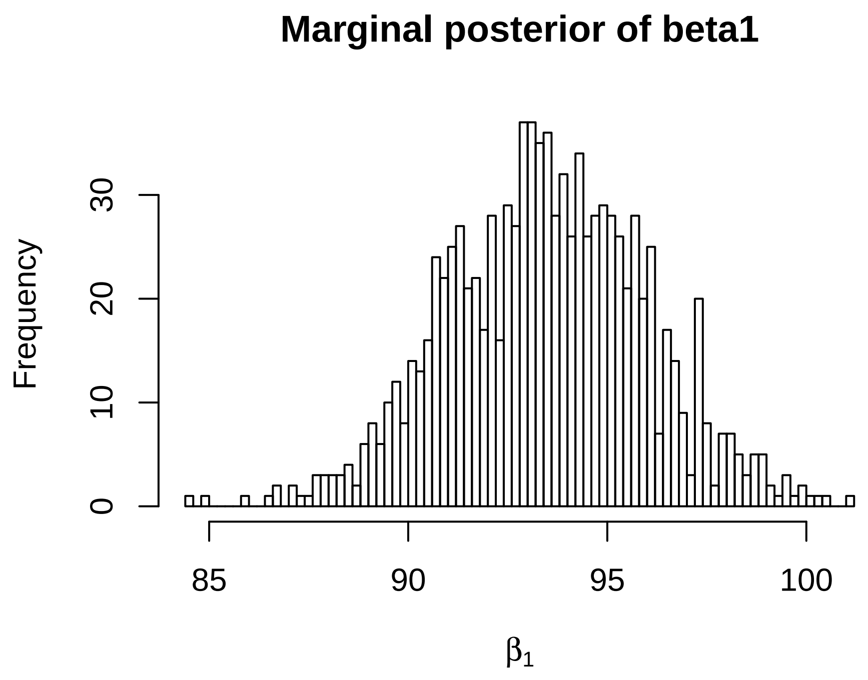


Our prior beliefs about the regression curves is based on the observed temperatures in Linkoping in the past. From the plot of the prior regression curves, we can see that the temperatures are in the range [10, 20] during the middle of the year and in the range [-10, 10] during the start and end of the year, which quite agrees with our prior beliefs about the temperatures in Linkoping.

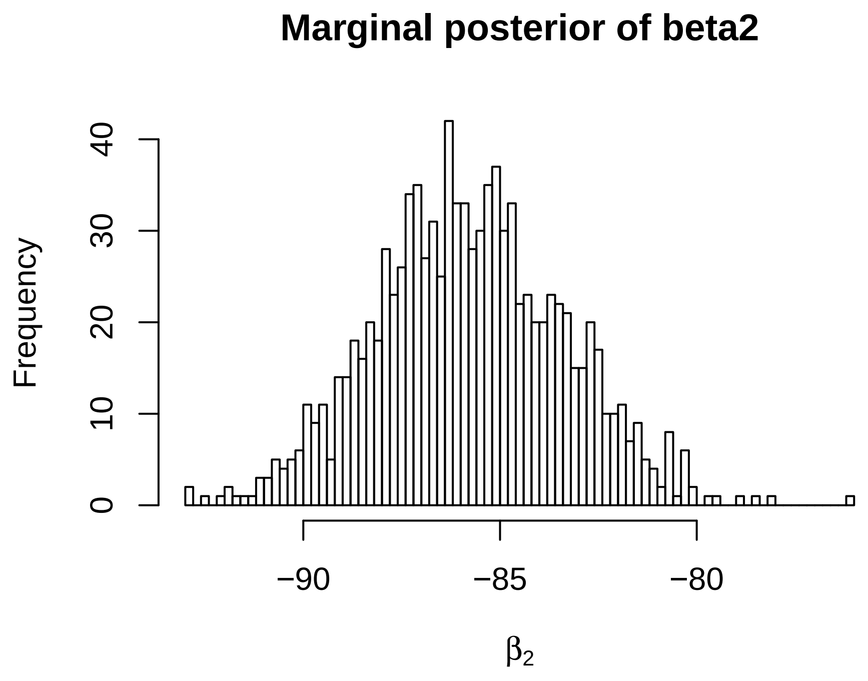
(b) Simulate from the joint posterior distribution of β_0, β_1, β_2 and σ^2



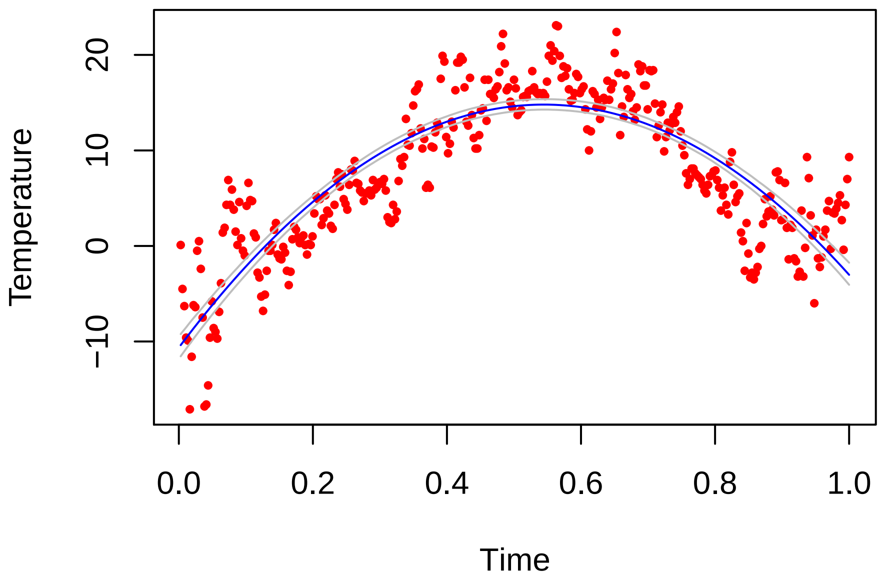




Marginal posterior of beta1
Frequency
0
10
20
30
85
90
95
100
β_1

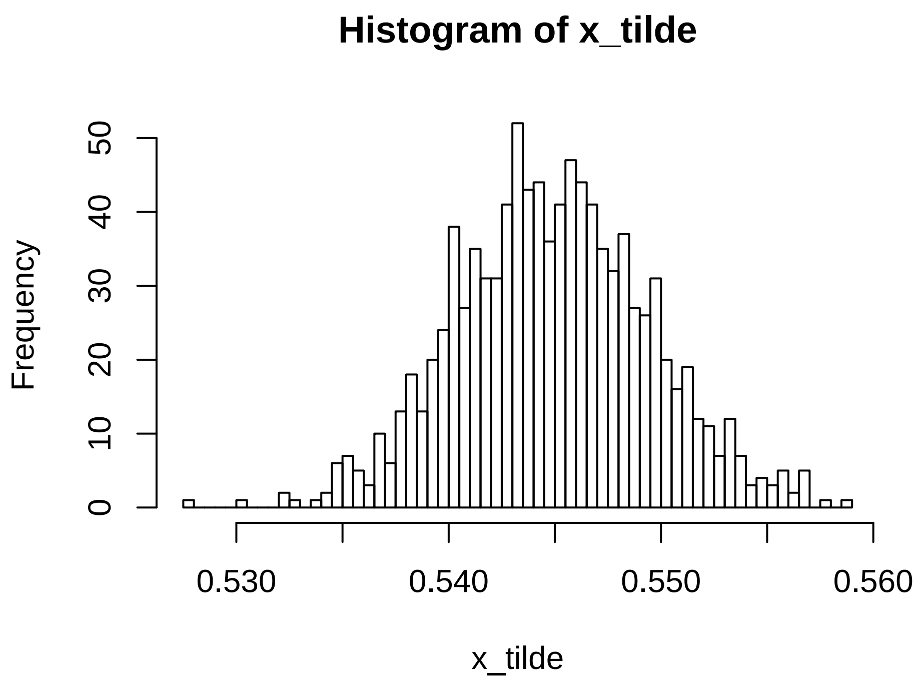


Marginal posterior of beta2
Frequency
0
10
20
30
40
−90
−85
−80
β_2



The 95% equal tail posterior interval curves do not contain most of the data points. Credible intervals capture our current uncertainty in the location of the parameter values and hence the smaller it is, the better our regression model is. We can see that the lower 2.5% and upper 97.5% posterior credible interval curves follow closely around the posterior median of the regression function.

(c) Posterior distribution of highest expected temperature $\tilde{x}$



We can see that the highest expected temperature follows a fairly normal distribution around the mean time=0.545 with a very small standard deviation as the entire range of the distribution of $\tilde{x}$ is [0.53, 0.56]. It can be said that the highest temperature is noted always in the middle of the year when the time is between 0.53 and 0.56.

(d) Estimate a polynomial model of order 7 without overfitting

To prevent over-fitting, we can regularize the prior of beta. $\beta_i|\sigma^2 \sim N(0, \sigma^2/\lambda)$ where $\Omega_0 = \lambda I$ Larger the lambda, smoother is the fit i.e more is the shrinkage of beta coefficients towards zero. As $\lambda - > \infty, \tilde{\beta} - > 0$ By setting mean to zero and lambda to a high value, we can regularize the beta prior to shrink most of the β_i to zero and keep some β_i large. For instance, $\mu = 0$ and $\lambda = 100$ is a suitable prior to estimate a polynomial model of order 7.

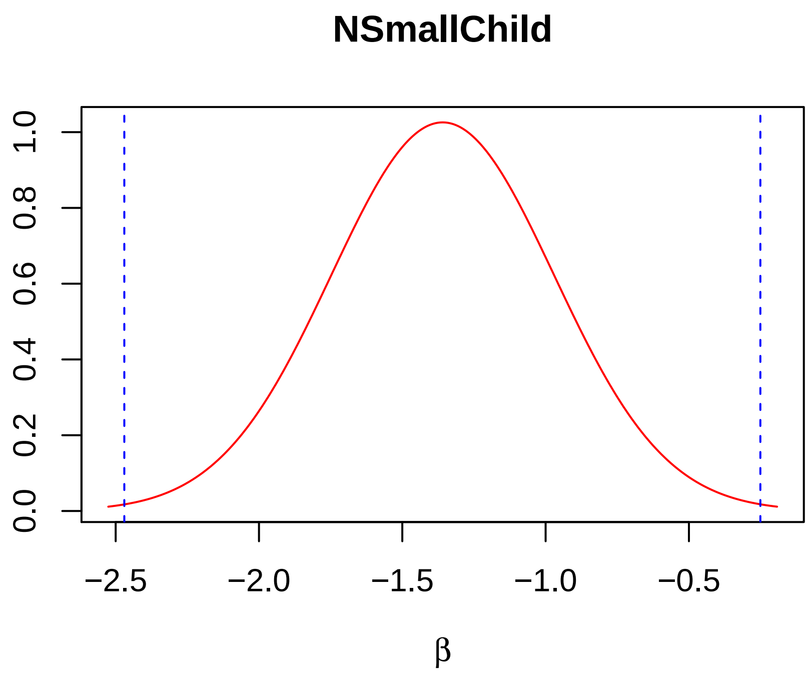
2. Posterior approximation for classification with logistic regression

(a) Logistic Regression

```
##
## Call:  glm(formula = Work ~ 0 + ., family = binomial, data = WomenWork)
##
## Coefficients:
##    Constant   HusbandInc    EducYears     ExpYears    ExpYears2          Age
##     0.64430     -0.01977      0.17988      0.16751     -0.14436     -0.08234
## NSmallChild    NBigChild
##    -1.36250     -0.02543
##
## Degrees of Freedom: 200 Total (i.e. Null);  192 Residual
## Null Deviance:       277.3
## Residual Deviance: 222.7     AIC: 238.7
```

(b) Approximating posterior distribution of the 8-dimensional parameter $\vec{\beta}$

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

## [1]  0.62672884 -0.01979113  0.18021897  0.16756670 -0.14459669 -0.08206561
## [7] -1.35913317 -0.02468351

## [1] "The posterior variance-covariance matrix is:"

##              [,1]          [,2]          [,3]          [,4]          [,5]
## [1,]  2.266022568  3.338861e-03 -6.545121e-02 -1.179140e-02  0.0457807243
## [2,]  0.003338861  2.528045e-04 -5.610225e-04 -3.125413e-05  0.0001414915
## [3,] -0.065451206 -5.610225e-04  6.218199e-03 -3.558209e-04  0.0018962893
## [4,] -0.011791404 -3.125413e-05 -3.558209e-04  4.351716e-03 -0.0142490853
## [5,]  0.045780724  1.414915e-04  1.896289e-03 -1.424909e-02  0.0555786706
## [6,] -0.030293450 -3.588562e-05 -3.240448e-06 -1.340888e-04 -0.0003299398
## [7,] -0.188748354  5.066847e-04 -6.134564e-03 -1.468951e-03  0.0032082535
## [8,] -0.098023929 -1.444223e-04  1.752732e-03  5.437105e-04  0.0005120144
##               [,6]          [,7]          [,8]
## [1,] -3.029345e-02 -0.1887483542 -0.0980239285
## [2,] -3.588562e-05  0.0005066847 -0.0001444223
## [3,] -3.240448e-06 -0.0061345645  0.0017527317
## [4,] -1.340888e-04 -0.0014689508  0.0005437105
## [5,] -3.299398e-04  0.0032082535  0.0005120144
## [6,]  7.184611e-04  0.0051841611  0.0010952903
## [7,]  5.184161e-03  0.1512621814  0.0067688739
## [8,]  1.095290e-03  0.0067688739  0.0199722657

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

## [1] 1.50533138 0.01589983 0.07885556 0.06596754 0.23575129 0.02680412 0.38892439
## [8] 0.14132327

## [1] "95% credible interval for NSmallChild variable:"

## [1] -2.4698452 -0.2507571
```

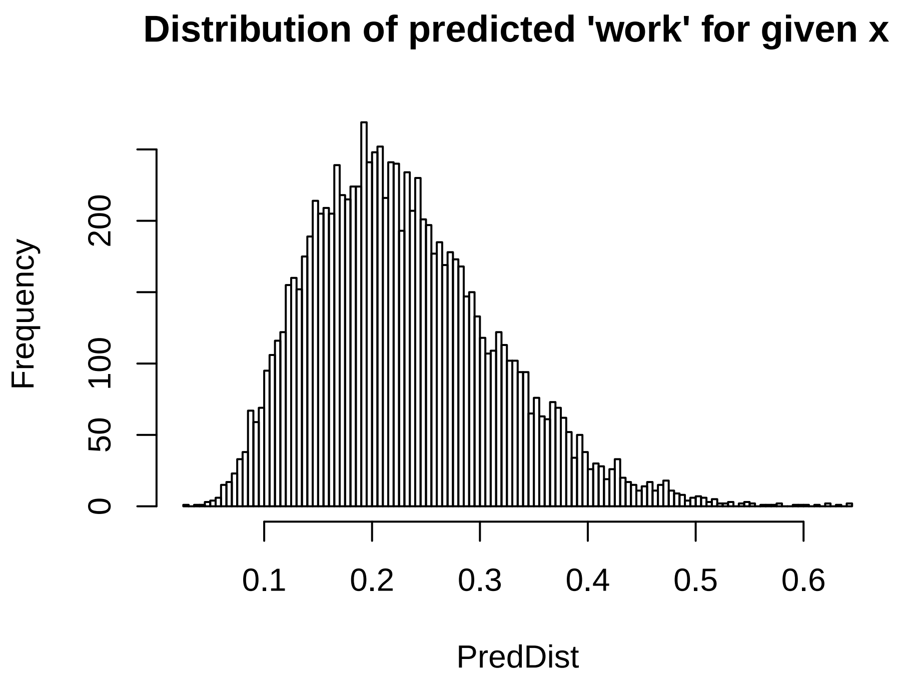


From the distribution of marginal beta (for variable NSmallChild), we can see that the uncertainity in the coefficient values is given in the range -2.5 and -0.5. Since zero is not included in this interval, it can be said that the coefficient is significant as it impacts the response negatively. In other words, having more number of small children makes it less likely that the woman will work.

(c) Simulate the predictive ditribution of response variable in logistic regression

```
## [1] "Predictor x:"
```

```
##    Constant  HusbandInc   EducYears    ExpYears   ExpYears2         Age 
##           1          10           8          10           1          40 
## NSmallChild   NBigChild 
##           1           1
```



From the distribution of the predicted response work for the given predictor values x, we can see that the probability that this woman works is quite low. As the distribution of the response variable is right skewed with a posterior mode probability of 0.2 and all the high probability density regions have probability less than 0.5, there is a very low chance that the woman works, when we use a threshold value of 0.5.

Appendix

```
########################################################################
# 1. Linear and polynomial regression
########################################################################
library("mvtnorm")

# a) Determining the prior distribution of the model parameters
tempLinkoping <- read.delim(
  file="C:/Users/namit/Downloads/Bayesian Learning/R files/Lab2/TempLinkoping.txt",
  header=TRUE)
n <- nrow(tempLinkoping)

#lm(formula=temp~time+I(time^2), data=tempLinkoping)

# Simulate from Inv-chisq
simulate_inv_chisq <- function(nDraws, df, s_sq) {
  X      <- rchisq(nDraws, df)
  ichisq <- df*s_sq / X

  return(ichisq)
}
```

```r
# Compute the regression curve for given beta parameter values
regression_curve <- function(beta_vec) {
  reg_curve <- beta_vec[1] + beta_vec[2] * tempLinkoping$time + beta_vec[3] * tempLinkoping$time^2

  return(reg_curve)
}

# Setting up the Prior
mu0    <- c(-10, 100, -100)          # Mean of the beta values (changing b0 shifts the curve up or down
omega0 <- diag(0.01, nrow=3, ncol=3) # Smoothness of the fit (tighter/looser band of curves)
nu0    <- 4                          # Confidence about the guess of variance of beta
sigsq0 <- 1                          # Best guess for the variance of beta/temp

# Simulate sigsq from Inv-chisq(nu0, sigsq0)
sigsq <- simulate_inv_chisq(nDraws=10, df=nu0, s_sq=sigsq0)

# Simulate beta from its joint conjugate prior i.e Normal(mu0, sigsq*inv_omega0)
inv_omega0 <- solve(omega0)
for (i in 1:length(sigsq)) {
  beta_vec  <- rmvnorm(1, mean=mu0, sigma=sigsq[i]*inv_omega0)
  reg_curve <- regression_curve(beta_vec)

  # Plot the regression curve computed for each beta drawn from the prior
  if(i==1)
    plot(tempLinkoping$time, reg_curve, type="l", col=i, ylim=c(-50,50),
         ylab="Temperature", xlab="Time")
  else
    points(tempLinkoping$time, reg_curve, type="l", col=i)
}

# Changing the hyperparameters in the Prior
mu0    <- c(-5, 90, -90)             # Mean of the beta values (changing b0 shifts the curve up or down
omega0 <- diag(0.5, nrow=3, ncol=3)  # Smoothness of the fit (tighter/looser band of curves)
nu0    <- 4                          # Confidence about the guess of variance of beta
sigsq0 <- 1                          # Best guess for the variance of beta/temp

# Simulate sigsq from Inv-chisq(nu0, sigsq0)
sigsq <- simulate_inv_chisq(nDraws=10, df=nu0, s_sq=sigsq0)

# Simulate beta from its joint conjugate prior i.e Normal(mu0, sigsq*inv_omega0)
inv_omega0 <- solve(omega0)
for (i in 1:length(sigsq)) {
  beta_vec  <- rmvnorm(1, mean=mu0, sigma=sigsq[i]*inv_omega0)
  reg_curve <- regression_curve(beta_vec)

  # Plot the regression curve computed for each beta drawn from the prior
  if(i==1)
    plot(tempLinkoping$time, reg_curve, type="l", col=i, ylim=c(-15,25),
         ylab="Temperature", xlab="Time")
  else
    points(tempLinkoping$time, reg_curve, type="l", col=i)
}
```

```r
# b) Simulate from the joint posterior distribution of beta0, beta1, beta2 and sigsq
X   <- cbind(1, poly(tempLinkoping$time, degree=2, raw=TRUE))
y   <- as.matrix(tempLinkoping$temp)
mu0 <- as.matrix(mu0)
XX  <- t(X) %*% X
yy  <- t(y) %*% y

beta_hat  <- solve(XX) %*% t(X) %*% y
mu_n      <- solve(XX+omega0) %*% ((XX %*% beta_hat) + (omega0 %*% mu0))
omega_n   <- XX + omega0
nu_n      <- nu0 + n
nusigsq_n <- nu0*sigsq0 + (yy+(t(mu0) %*% omega0 %*% mu0)-(t(mu_n) %*% omega_n %*% mu_n))
sigsq_n   <- nusigsq_n /nu_n

# Simulate sigsq from Inv-chisq(nu_n, sigsq_n)
sigsq <- simulate_inv_chisq(nDraws=1000, df=nu_n, s_sq=c(sigsq_n))

# Simulate beta from its joint posterior i.e Normal(mu_n, sigsq*inv_omega_n)
inv_omega_n <- solve(omega_n)
beta_vec    <- matrix(0, nrow=length(sigsq), ncol=3) # k=3
reg_curve   <- matrix(0, nrow=length(sigsq), ncol=n)
x_tilde1    <- numeric(length(sigsq))

for (i in 1:length(sigsq)) {
  beta_vec[i, ]  <- rmvnorm(1, mean=mu_n, sigma=sigsq[i]*inv_omega_n)
  reg_curve[i, ] <- regression_curve(beta_vec[i, ])
  x_tilde1[i]    <- tempLinkoping$time[which.max(reg_curve)]

  # Plot the regression curve computed for each beta drawn from the prior
  if(i==1)
    plot(tempLinkoping$time, reg_curve[i, ], type="l", col=i, ylim=c(-20,20),
         ylab="Temperature", xlab="Time")
  else
    points(tempLinkoping$time, reg_curve[i, ], type="l", col=i)
}

# Marginal posterior of beta0, beta1 and beta2
hist(beta_vec[, 1], breaks=100, main="Marginal posterior of beta0", xlab=expression(beta[0]))
hist(beta_vec[, 2], breaks=100, main="Marginal posterior of beta1", xlab=expression(beta[1]))
hist(beta_vec[, 3], breaks=100, main="Marginal posterior of beta2", xlab=expression(beta[2]))

# Posterior median of regression function
beta_median <- apply(beta_vec, 2, median)
reg_median  <- beta_median[1] + beta_median[2] * tempLinkoping$time + beta_median[3] * tempLinkoping$ti

# 95% equal-tail posterior probability interval
credInterval <- function(credible=0.95, data_vec) {
  eq_tail  <- (1-credible)/2                                             # tail region
  tail     <- eq_tail*length(data_vec)                                   # tail % of the posterior samples
  CredI    <- data_vec[order(data_vec)][tail:(length(data_vec)-tail)] # credible interval
  CredI_L  <- CredI[1]                                                   # lower limit
  CredI_U  <- CredI[length(CredI)]                                       # upper limit
  return(c(CredI_L, CredI_U))
}

CredI <- matrix(0, nrow=ncol(reg_curve), ncol=2) # ncol=Upper and Lower limits of the Credible Interval
for (i in 1:ncol(reg_curve)) {
  temp       <- reg_curve[, i]
  CredI[i, ] <- credInterval(credible=0.95, data_vec=temp)
}

# Scatterplot of temperature data
plot(tempLinkoping$time, tempLinkoping$temp, ylab="Temperature", xlab="Time",
     col="red", pch=16, cex=0.6)
points(tempLinkoping$time, reg_median, type="l", col="blue")
points(tempLinkoping$time, CredI[, 1], type="l", col="gray")
points(tempLinkoping$time, CredI[, 2], type="l", col="gray")

# c) Posterior distribution of highest expected temperature
x_tilde <- -beta_vec[, 2]/ (2*beta_vec[, 3])
hist(x_tilde, breaks=100)


##########################################################################
# 2. Posterior approximation for classification with logistic regression
##########################################################################

# a) Logistic Regression
WomenWork <- read.table(
  file="C:/Users/namit/Downloads/Bayesian Learning/R files/Lab2/WomenWork.dat",
  header=TRUE)

glmModel <- glm(Work ~ 0 + ., data=WomenWork, family=binomial)
print(glmModel)

# b) Approximating posterior distribution of the 8-dimensional parameter vector beta

# Function that returns log posterior of beta
logPost <- function(beta_vec, X, y, mu, sigma) {
  ncovariates <- length(beta_vec)
  linPred     <- X %*% beta_vec

  # Log likelihood of data y
  logLik      <- sum(linPred*y - log(1 + exp(linPred)))

  # Log likelihood of prior of beta ~ N(0, Tau*I)
  logPrior    <- dmvnorm(beta_vec, mu, sigma, log=TRUE)

  # Log posterior = Log likelihood + Log prior
  return(logLik + logPrior)
}

# Predictors and response variables
X <- as.matrix(WomenWork[, -1])
y <- WomenWork[, 1]
```

```r
# Covariates
ncovariates <- ncol(X)
covNames    <- names(WomenWork)[-1]

# Set up prior parameters
mu    <- rep(0, times=ncovariates)
tau   <- rep(10, times=ncovariates)
sigma <- diag(tau^2)

# Find the optimum beta that maximizes the log posterior of beta
beta_init  <- as.vector(rep(0, ncovariates))
optim_beta <- optim(beta_init, logPost, gr=NULL, X, y, mu, sigma, method=c("BFGS"),
                    control=list(fnscale=-1), hessian=TRUE)

postMode   <- optim_beta$par              # Posterior mode=Optimum beta that maximizes the log posterior
postCov    <- -solve(optim_beta$hessian) # Posterior covariance matrix is -inv(Hessian)
PostStd    <- sqrt(diag(postCov))         # Computing approximate standard deviations.

print('The posterior mode is:')
print(postMode)
print('The posterior variance-covariance matrix is:')
print(postCov)
print('The approximate posterior standard deviation is:')
print(PostStd)

# Marginal posterior of beta for the covariate NSmallChild - 99.5% of the data lies
# within 3 SD of the mean. Hence we sample marginal posterior as follows
NSmallChild_grid <- seq(postMode[7] - 3*PostStd[7], postMode[7] + 3*PostStd[7], length=1000)

# 95% credible interval for NSmallChild
CredI <- credInterval(credible=0.95, data=NSmallChild_grid)
print('95% credible interval for NSmallChild variable:')
print(CredI)

# Plot marginal posterior of NSmallChild with 95% equal tail interval
plot(NSmallChild_grid, dnorm(x=NSmallChild_grid, mean=postMode[7], sd=PostStd[7]),
     type="l", col="red", main=covNames[7], ylab='', xlab=expression(beta))
abline(v=CredI, col="blue", lty="dashed")

# c) Simulate the predictive ditribution of response variable in logistic regression

# Simulate from approximate posterior of beta ~ N(beta_mode, J_inv(beta_mode))
SimulateBetaPost <- function(nDraws, mu, sigma) {
  betaPost <- rmvnorm(n=nDraws, mean=mu, sigma=sigma)
  return(betaPost)
}

# Logistic Regression
logisticReg <- function(beta_vec, x) {
  linPred <- x %*% as.matrix(beta_vec)
  logReg  <- exp(linPred) / (1+exp(linPred))

  return(logReg)
```

```r
}

# Simulate the distribution of predicted response in logistic regression
SimulatePredLogReg <- function(x, postMode, postCov) {
  BetaPost <- SimulateBetaPost(nDraws=10000, mu=postMode, sigma=postCov)
  WorkPred <- apply(BetaPost, 1, logisticReg, x)

  return(WorkPred)
}

# Predictor variable
x        <- c(1, 10, 8, 10, 1, 40, 1, 1)
names(x) <- covNames
print('Predictor x:')
print(x)

# Distribution of predicted response in Logistic regression
PredDist <- SimulatePredLogReg(x=x, postMode, postCov)

# Plot the predicted response
hist(PredDist, breaks=100, main="Distribution of predicted 'work' for given x")
```