Bayesian Learning Lab 2

Nazli Bilgic (nazbi056) & Simge Cinar (simci637)

2024-04-27

Question 1: Linear and polynomial regression

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 defined by

$$\text{time} = \frac{\text{the number of days since the beginning of the year}}{365}$$

A Bayesian analysis of the following quadratic regression model is to be performed:

$$temp = \beta_0 + \beta_1 \cdot time + \beta_2 \cdot time^2 + \epsilon, \epsilon \overset{iid}{\sim} N(0, \sigma^2)$$

```
df <- read_xlsx("Linkoping2022.xlsx")
df$time <- 1:nrow(df) / 365
```

```
n <- nrow(df)
x1 <- rep(1, times = n)
x2 <- df$time
x3 <- (df$time)^2

X <- cbind(x1, x2, x3)
y <- matrix(df$temp, ncol = 1)
```

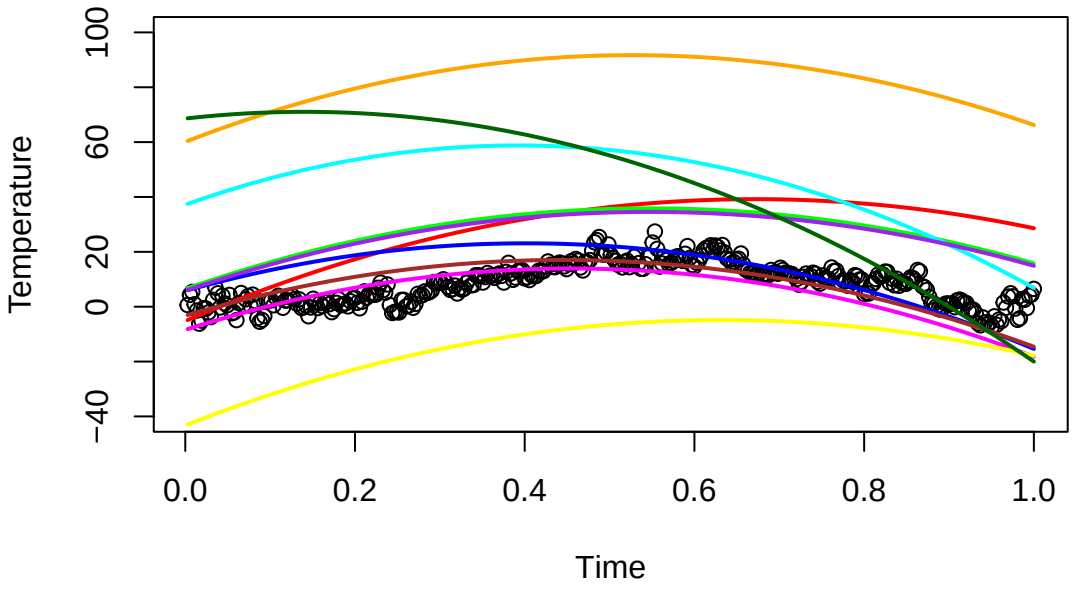
Part a)

Question: Use conjugate prior for the linear regression model. The prior hyperparameters μ_0, Ω_0, ν_0 and σ_0^2 shall be set to sensible values. Start with $\mu_0 = (0, 100, -100)^T, \Omega_0 = 0.01 \cdot I_3, \nu_0 = 1$ and $\sigma_0^2 = 1$. 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. Does 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: R package mvtnorm can be used and your $Inv - \chi^2$ simulator of random draws from Lab 1.]

Answer:

```
set.seed(123)
# Given prior hyperparameters

# mean and standard deviation for guassian distribution
mu0 <- matrix(c(0,100,-100), ncol = 1)
omega0 <- 0.01 * diag(3)
```

```r
# number of chi-squared degrees of freedom and the scaling parameter for inverse chi-square
nu0 <- 1
sigma0_sqr <- 1

# Getting joint prior for beta and sigma^2
nDraws <- 10
sigma_new <- matrix(ncol=1, nrow=nDraws)
beta_new <- matrix(ncol=3, nrow=nDraws)

for (i in 1:nDraws){
  sigma_new[i,] <- (nu0*sigma0_sqr)/ rchisq(1, nu0)
  beta_new[i,] <- rmvnorm(1, mu0, sigma_new[i,]*solve(omega0))
}

y_prior <- matrix(nrow = n, ncol = nDraws)
for (l in 1:nDraws){
  err <- rnorm(1, mean = 0, sd = 1)
  y_prior[,l] <- X[,1] * beta_new[l,1] + X[,2] * beta_new[l,2] + X[,3] * beta_new[l,3] + err
}

plot(df$time, df$temp, ylim = c(-40, 100), ylab = 'Temperature', xlab = 'Time')
colors <- c("red", "blue", "green", "cyan", "magenta", "yellow", "orange", "purple",
            "brown", "darkgreen")
for (i in 1:nDraws){
  lines(df$time, y_prior[,i], col = colors[i], type = 'l', lwd = 2)
}
```




```r
cat("Minimum variance is", min(sigma_new), "\n",
    "Maximum variance is", max(sigma_new), "\n")
```

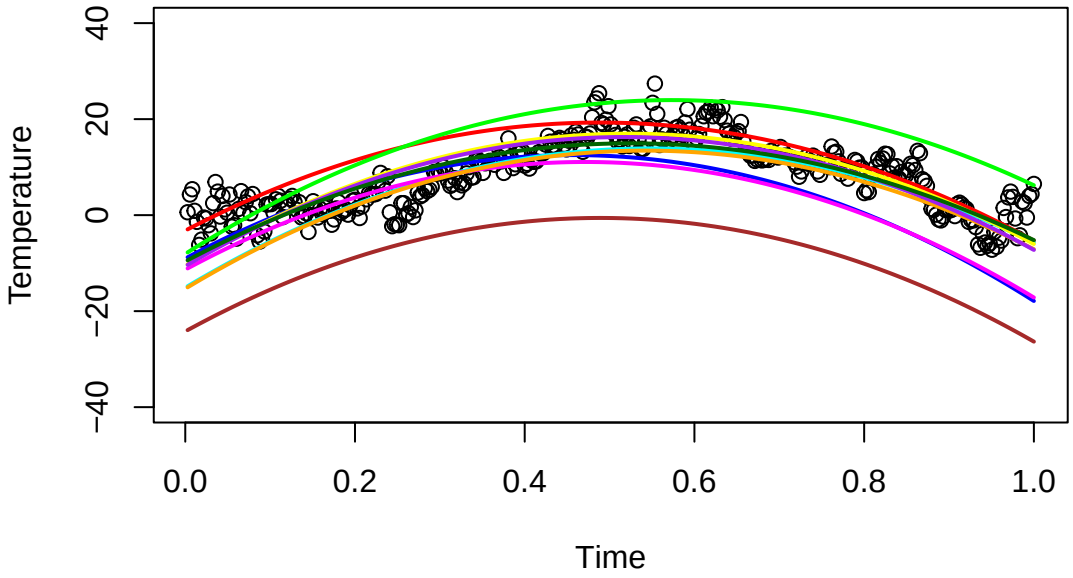
```
## Minimum variance is 0.5375011
##  Maximum variance is 33.71154
```

We fitted 10 different regression curves with the given prior hyperparameters. We can say that prior hyperparameters are not reasonable by looking at the graph above hence we changed to values as $\mu_0 =$

$(0, 100, -100)^T, \Omega_0 = 0.01 \cdot I_3, \nu_0 = 100, \sigma_0^2 = 0.2$. By changing ν_0 and σ_0^2 values, we decreased the variance in the normal distribution for β. The minimum and maximum σ can be seen below and there's so much difference between minimum and maximum value. Moreover changing the first value of μ_0 from 0 to 10 changed the y-intercept of the curve.

The fitted curves below agrees with our prior beliefs about the regression curve and curves are stable.

```
set.seed(123)
# Selected prior hyperparameters

# mean and standard deviation for guassian distribution
mu0 <- matrix(c(-10,100,-100), ncol = 1)
omega0 <- 0.01 * diag(3)

# number of chi-squared degrees of freedom and the scaling parameter for inverse chi-square
nu0 <- 100
sigma0_sqr <- 0.2

# Getting joint prior for beta and sigma^2
nDraws <- 10
sigma_new <- matrix(ncol=1, nrow=nDraws)
beta_new <- matrix(ncol=3, nrow=nDraws)

for (i in 1:nDraws){
  sigma_new[i,] <- (nu0*sigma0_sqr)/ rchisq(1, nu0)
  beta_new[i,] <- rmvnorm(1, mu0, sigma_new[i,]*solve(omega0))
}

y_prior <- matrix(nrow = n, ncol = nDraws)
for (l in 1:nDraws){
  err <- rnorm(1, mean = 0, sd = 1)
  y_prior[,l] <- X[,1] * beta_new[l,1] + X[,2] * beta_new[l,2] + X[,3] * beta_new[l,3] + err
}

plot(df$time, df$temp, ylim = c(-40, 40), ylab = 'Temperature', xlab = 'Time')
colors <- c("red", "blue", "green", "cyan", "magenta", "yellow", "orange", "purple",
            "brown", "darkgreen")
for (i in 1:nDraws){
  lines(df$time, y_prior[,i], col = colors[i], type = 'l', lwd = 2)
}
```




```
cat("Minimum variance is", min(sigma_new), "\n",
    "Maximum variance is", max(sigma_new), "\n")
```

```
## Minimum variance is 0.1928734
##  Maximum variance is 0.2422167
```

Part b)

Question:

Write a function that simulate draws from the joint posterior distribution of β_0, β_1, β_2 and σ^2.

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 $f(time) = E[temp|time] = \beta_0 + \beta_1 \cdot time + \beta_2 \cdot time^2$, 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 percentiles of f(time) is computed for every value of time. Does the posterior probability intervals contain most of the data points? Should they?

Answer: Joint prior β and σ^2 is as follows:

$$\beta|\sigma^2 \sim N(\mu_0, \sigma^2\Omega_0^{-1})$$
$$\sigma^2 \sim Inv - \chi^2(\nu_0, \sigma_0^2)$$

Then the posterior is as follows:

$$\beta|\sigma^2 \sim N(\mu_n, \sigma^2\Omega_n^{-1})$$
$$\sigma^2 \sim Inv - \chi^2(\nu_n, \sigma_n^2)$$

where

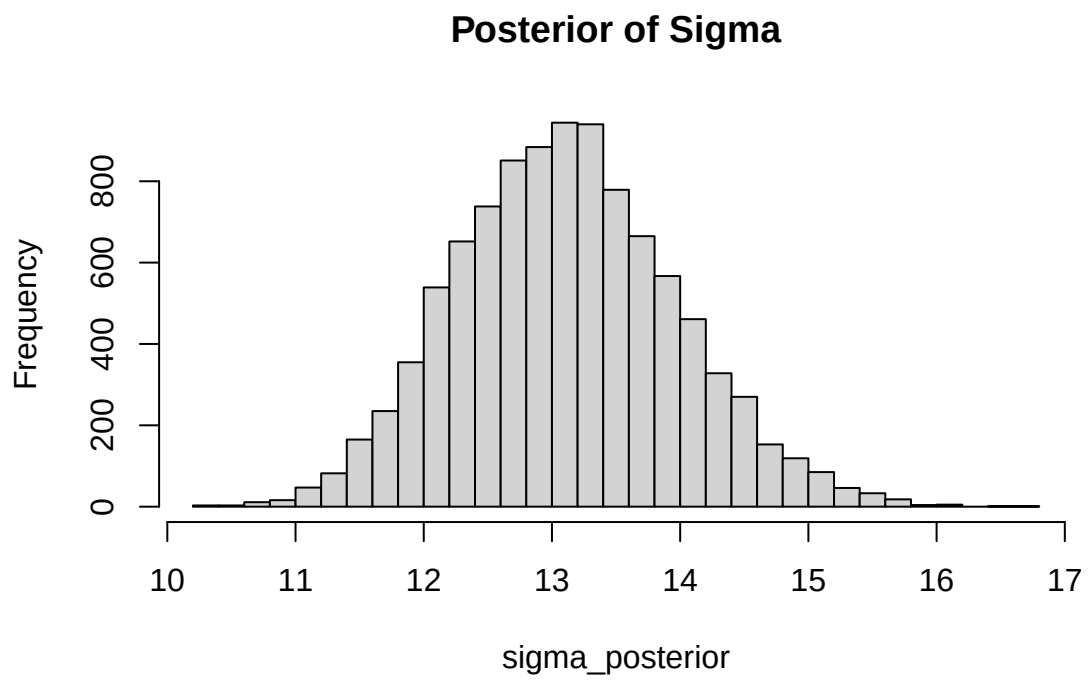
$$\hat{\beta} = (X'X)^{-1}X'y$$
$$\mu_n = (X'X + \Omega_0)^{-1}(X'X\hat{\beta} + \Omega_0\mu_0)$$
$$\Omega_n = X'X + \Omega_0$$
$$\nu_n = \nu_0 + n$$
$$\nu_n\sigma_n^2 = \nu_0\sigma_0^2 + (y'y + \mu_0'\Omega_0\mu_0 - \mu_n'\Omega_n\mu_n)$$

```r
# Posterior hyperparameters
beta_hat <- solve(t(X) %*% X) %*% t(X) %*% y

mu_n <- solve(t(X) %*% X + omega0) %*% ((t(X) %*% X) %*% beta_hat) +
  (omega0 %*% mu0)
omega_n <- t(X) %*% X + omega0
nu_n <- nu0 + n
sigma_n_sqr <- (nu0 %*% sigma0_sqr + (t(y) %*% y) +
                  (t(mu0) %*% omega0 %*% mu0) - (t(mu_n) %*% omega_n %*% mu_n)) / nu_n
```

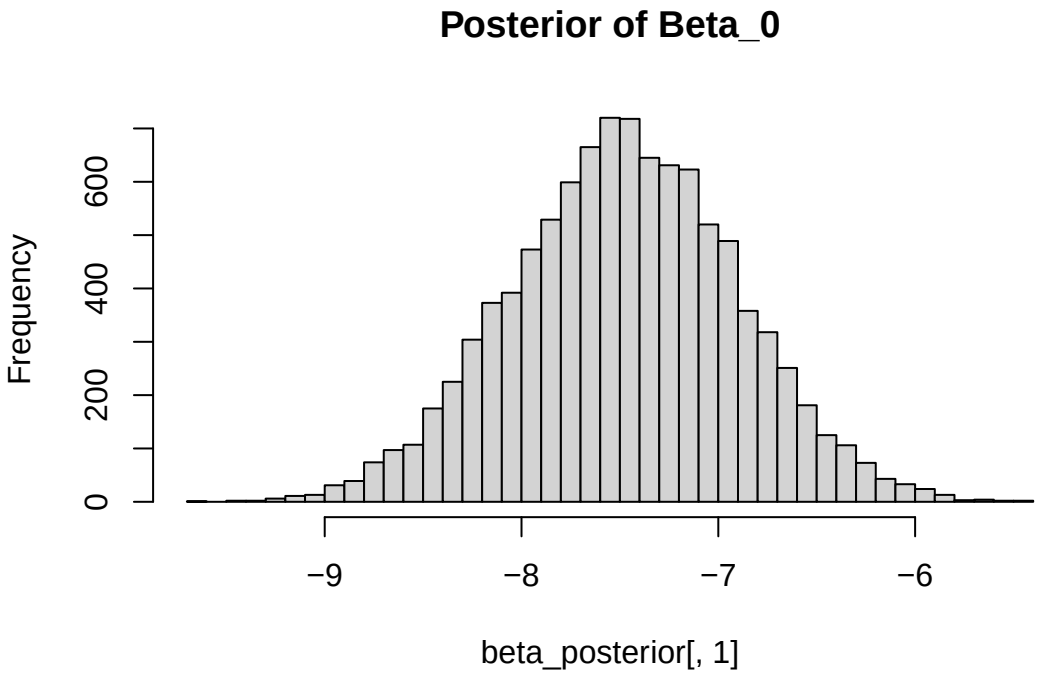
```r
get_posterior_parameters <- function(nDraws){
  sigma_beta_posterior <- matrix(ncol=4, nrow=nDraws)

  for (i in 1:nDraws){
    sigma_beta_posterior[i,1] <- (nu_n*sigma_n_sqr)/ rchisq(1, nu_n)
    sigma_beta_posterior[i,2:4] <- rmvnorm(1, mu_n, sigma_beta_posterior[i,1]*solve(omega_n))
  }
  return(sigma_beta_posterior)
}

posterior <- get_posterior_parameters(nDraws = 10000)
sigma_posterior <- posterior[,1]
beta_posterior <- posterior[,2:4]
```

i) Histogram for each marginal posterior of the parameters can be seen below

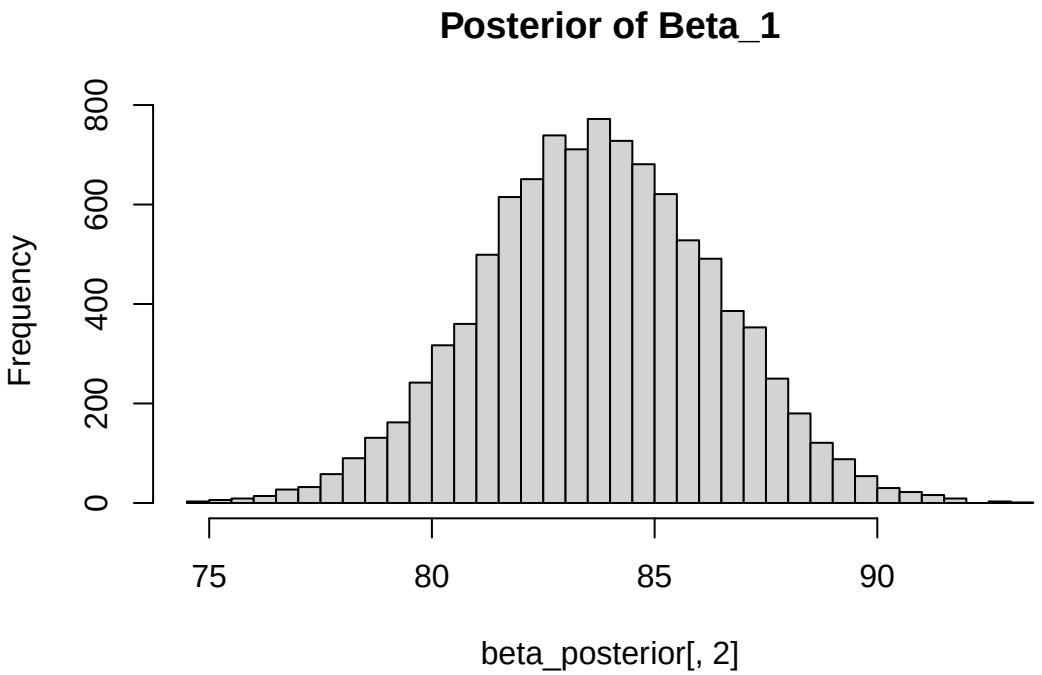
```r
hist(sigma_posterior, breaks = 30, main = "Posterior of Sigma")
```



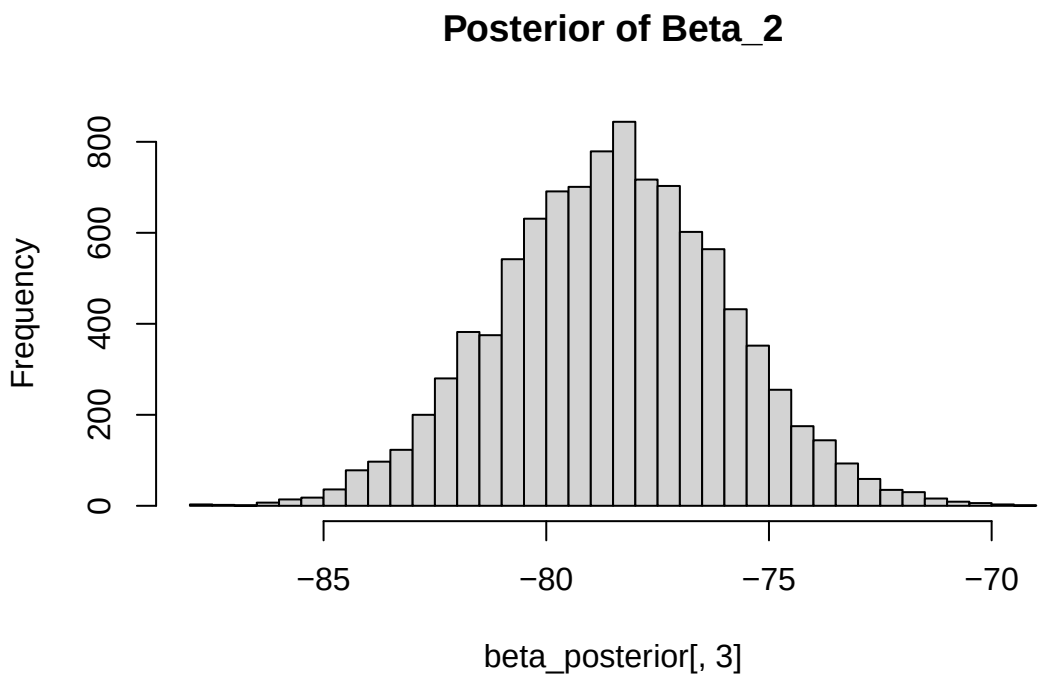

```r
hist(beta_posterior[,1], breaks = 30, main = "Posterior of Beta_0")
```




```
hist(beta_posterior[,2], breaks = 30, main = "Posterior of Beta_1")
```



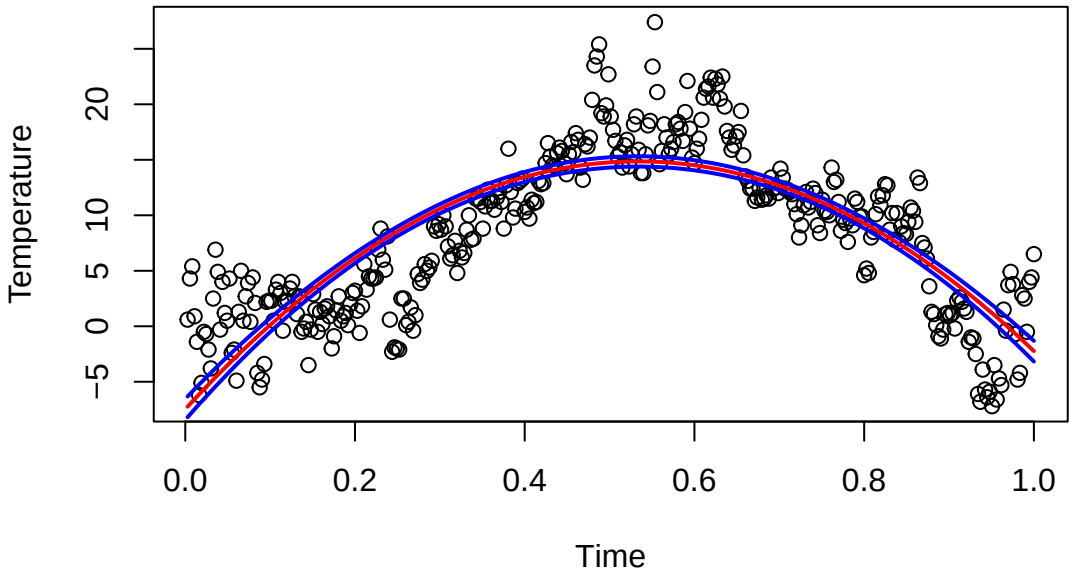

```
hist(beta_posterior[,3], breaks = 30, main = "Posterior of Beta_2")
```



ii) 90% equal tail probability can be seen below. Yes, the posterior probability intervals contain most of the data points but with more complex model by adding higher order terms, we can capture more of the data points.

```
y_posterior <- beta_posterior %*% t(X) # getting predictions
y_posterior_median <- apply(y_posterior, 2, median) # median of the predictions

# calculating 90% equal tail probability
posterior_5th_percentile <- apply(y_posterior, 2, quantile, probs = 0.05)
posterior_95th_percentile <- apply(y_posterior, 2, quantile, probs = 0.95)
```

```
plot(df$time, df$temp, ylab = 'Temperature', xlab = 'Time')
lines(df$time, y_posterior_median, col = 'red', lwd = 2)
lines(df$time, posterior_5th_percentile, col = 'blue', lwd = 2)
lines(df$time, posterior_95th_percentile, col = 'blue', lwd = 2)
```



Part c)

Question: 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 $\tilde{x}$. Use the simulated draws in (b) to simulate from the posterior distribution of $\tilde{x}$. [Hint: the regression curve is a quadratic polynomial. Given each posterior draw of β_0, β_1 and β_2, you can find a simple formula for $\tilde{x}$.]

Answer:

The function can be seen below. Since it is quadratic, a parabola, we can take first derivative and set it to 0. That point will give us the maximum temperature

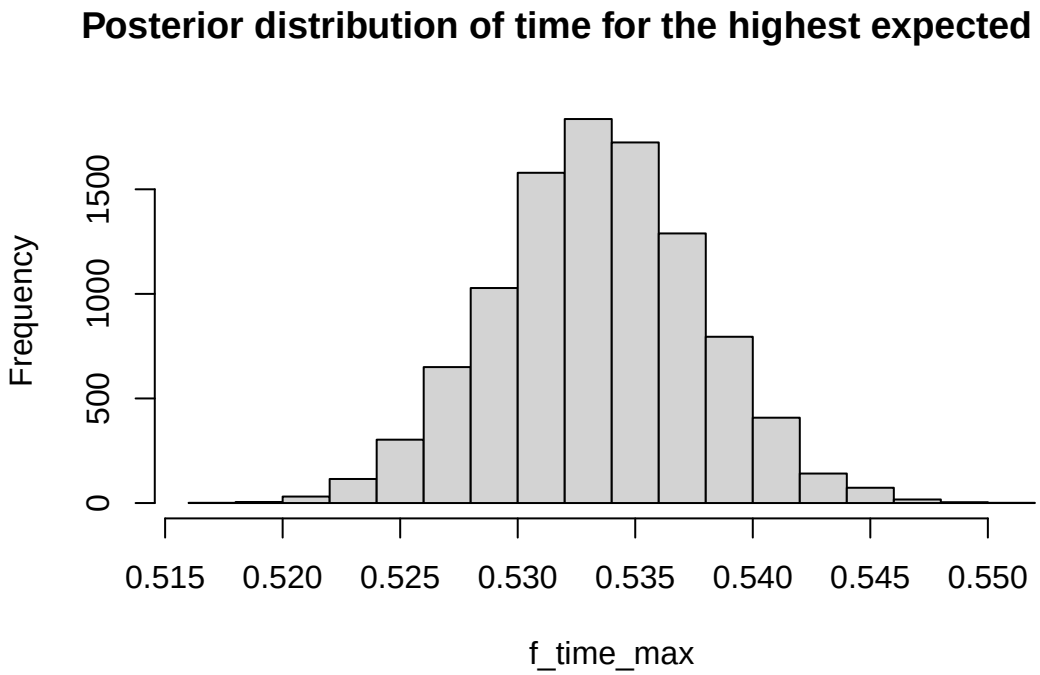
$$f(time) = \beta_0 + \beta_1 \cdot time + \beta_2 \cdot time^2$$

$$f'(time) = \beta_1 + 2\beta_2 \cdot time = 0$$

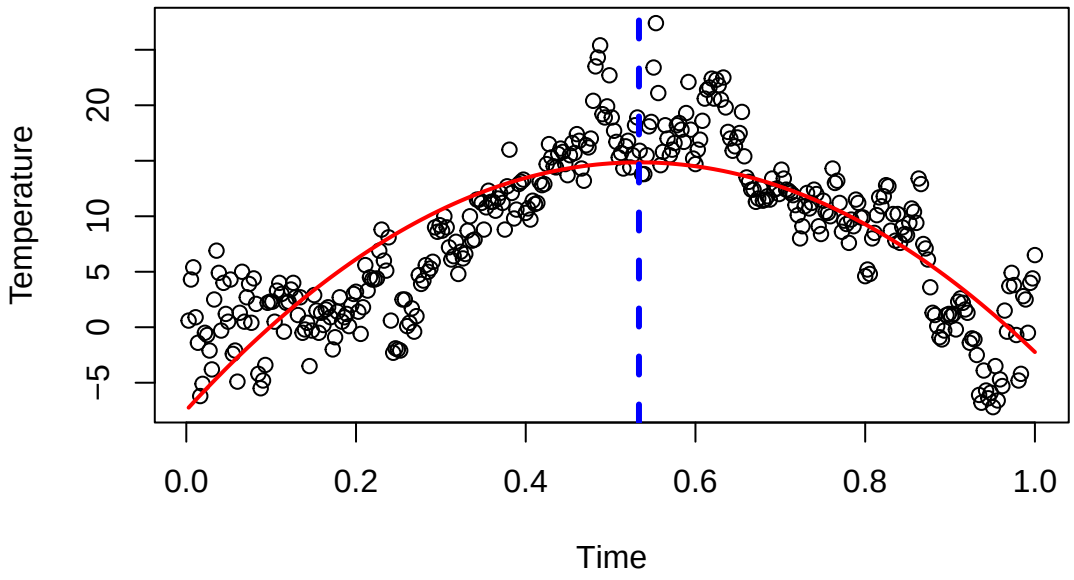
$$time_{maxTemp} = \tilde{x} = \frac{-\beta_1}{2\beta_2}$$

Posterior distribution of $\tilde{x}$ is as follows

```
# First derivative is 0 since it is a parabola
f_time_max <- -beta_posterior[,2] / (2*beta_posterior[,3])
hist(f_time_max, main = "Posterior distribution of time for the highest expected temp.")
```




```
plot(df$time, df$temp, ylab = 'Temperature', xlab = 'Time')
lines(df$time, y_posterior_median, col = 'red', lwd = 2)
abline(v = median(f_time_max), lwd = 3, lty = 2, col = 'blue')
```



Part d)

Question: 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 overfitting 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 μ_0 and Ω_0 in a suitable way.]

Answer: Too many variables leads to overfitting hence we will use regularization prior for higher order terms. Larger λ value gives smoother fit

$$\beta_j|\sigma^2 \sim N(0, \frac{\sigma^2}{\lambda})$$

We choose our priors as $\mu_0 = (-10, 100, -100, 0, 0, 0, 0, 0, 0, 0, 0)$ and $\Omega_0 = (0.01, 0.01, 0.01, 100, 100, 100, 100, 100, 100, 100, 100)$

Question 2: Posterior approximation for classification with logistic regression

The dataset WomenAtWork.dat contains n = 132 observations on the following eight variables related to women:

Variable	Data type	Meaning	Role
Work	Binary	Whether or not the woman works	Response y
Constant	1	Constant to intercept	Feature
HusbandInc	Numeric	Husband's income	Feature
EducYears	Counts	Years of education	Feature
ExpYears	Counts	Years of experience	Feature
Age	Counts	Age	Feature
NSmallChild	Counts	Number of child <= 6 years in household	Feature
NBigChild	Counts	Number of child >= 6 years in household	Feature

```
df2 <- read.table("WomenAtWork.dat", header = TRUE)
```

Part a)

Question:

Consider the logistic regression model:

$$Pr(y = 1|x, \beta) = \frac{exp(x^T \beta)}{1 + exp(x^T \beta)}$$

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

$$\beta|y, x \sim N(\tilde{\beta}, J_y^{-1}(\tilde{\beta})),$$

where $\tilde{\beta}$ is the posterior mode and $J(\tilde{\beta}) = -\frac{\partial^2 lnp(\beta|y)}{\partial\beta\partial\beta^T}|_{\beta=\tilde{\beta}}$ is the negative of the observed Hessian evaluated at the posterior mode. Note that $\frac{\partial^2 lnp(\beta|y)}{\partial\beta\partial\beta^T}$ is a 7×7 matrix with second derivatives on the diagonal and cross-derivatives $\frac{\partial^2 lnp(\beta|y)}{\partial\beta_i\partial\beta_j}$ on the off-diagonal. You can compute this derivative by hand, but we will let the computer do it numerically for you. Calculate both $\tilde{\beta}$ and $J(\tilde{\beta})$ 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 $\beta \sim N(0, \tau^2 I)$, where$\tau = 2$.

Present the numerical values of $\tilde{\beta}$ and $J_y{-}1(\tilde{\beta})$ for the WomenAtWork data. Compute an approximate 95% equal tail posterior probability interval for the regression coeffcient 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).]

Answer:

```
Covs <- c(2:8) # Select which covariates/features to include
X <- as.matrix(df2[,Covs])
Xnames <- colnames(X)
y <- as.matrix(df2[,1])

nObs <- dim(df2)[1]
nPar <- dim(df2)[2] -1 # subtract y
```

```
# Setting up the prior
tau <- 2
mu <- as.matrix(rep(0, nPar)) # Prior mean vector
Sigma <- (tau^2)*diag(nPar) # Prior covariance matrix
```

```
# Now we will use optim. Inputs are;
# 1) log p(theta|y) function
# 2) initial values for betas

# input 1)
LogPostLogistic <- function(betas, y, X, mu, Sigma){
  linPred <- X%*%betas
  logLik <- sum( linPred*y - log(1 + exp(linPred)) )
  logPrior <- dmvnorm(betas, mu, Sigma, log=TRUE) # densities are given as log, calculates density
  return(logLik + logPrior)
}
```

```r
# input 2)
initVal <- matrix(0, nPar, 1)

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

```r
# Printing the results to the screen
posterior_mode <- t(OptimRes$par)
colnames(posterior_mode) <- Xnames
approxPostStd <- sqrt(diag(solve(-OptimRes$hessian))) # Computing approximate standard deviations.
names(approxPostStd) <- Xnames # Naming the coefficient by covariates

print('The posterior mode is:')
```

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

```r
print(posterior_mode)
```

```
##          Constant  HusbandInc EducYears  ExpYears         Age NSmallChild
## [1,] -0.04036943 -0.03730689 0.1786895 0.1207364 -0.04618995   -1.472489
##        NBigChild
## [1,] -0.02014458
```

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

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

```r
print(approxPostStd)
```

```
##    Constant  HusbandInc   EducYears    ExpYears         Age NSmallChild
##  1.38198486  0.02198474  0.08920960  0.03335982  0.02747315  0.47746764
##   NBigChild
##  0.16401959
```

```r
lower_bound <- OptimRes$par[6] - 1.96*approxPostStd[6]
upper_bound <- OptimRes$par[6] + 1.96*approxPostStd[6]
cat("95% credible interval is [", lower_bound, ",", upper_bound, "]")
```

```
## 95% credible interval is [ -2.408326 , -0.5366527 ]
```

```r
# testing approximation
glmModel <- glm(Work ~ 0 + ., data = df2, family = binomial)
glmModel$coefficients
```

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

The approximate posterior mode for NSmallChild is -1.47248930 and the coefficient from the glmModel for that variable is -1.56485140. These values are close to each other.

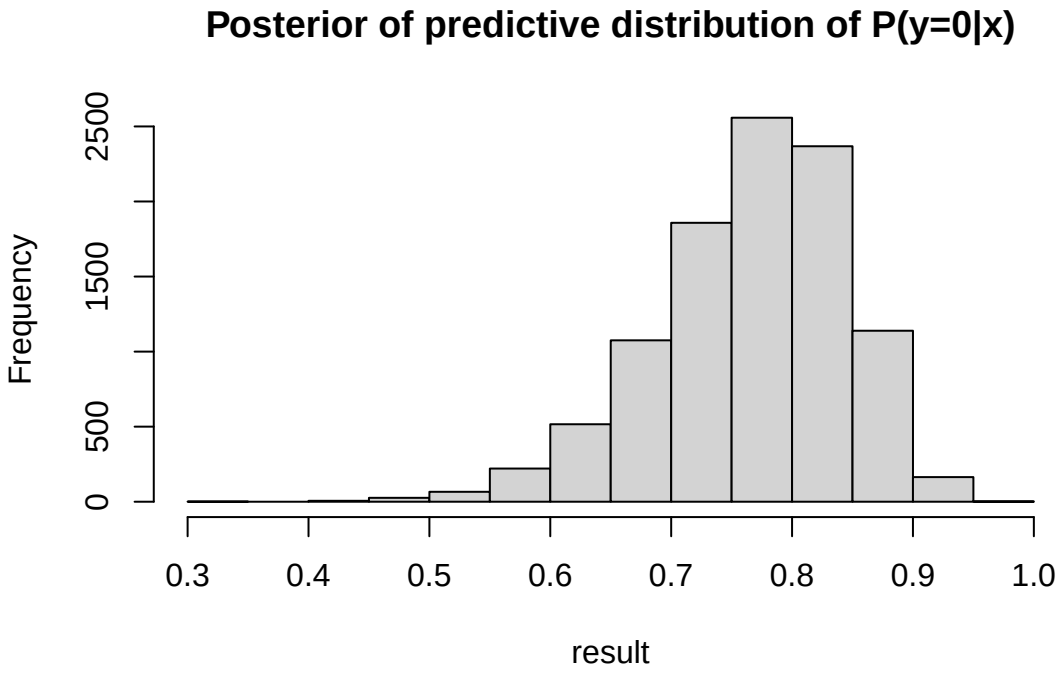
Part b)

Question: 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.]

Answer:

```
mu_posterior <- OptimRes$par
Sigma2_posterior <- solve(-OptimRes$hessian)

input <- c(1, 18, 11, 7, 40, 1, 1)
X_input <- as.matrix(input, ncol = 1)

nDraws <- 10000

generated_beta <- rmvnorm(nDraws, mean = mu_posterior, sigma = Sigma2_posterior)
linPred <- generated_beta %*% X_input

sigmoid_fnc <- function(linPred){
  return(1/(1+exp(linPred)))
}

result <- apply(linPred, 1, sigmoid_fnc)
hist(result, main = "Posterior of predictive distribution of P(y=0|x)")
```



Part c)

Question: 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.]

Answer:

```
nDraws <- 10000
pred_draw <- rep(nDraws, 0)
```

```r
for (i in 1:nDraws){
  # generate posterior draw
  generated_beta <- rmvnorm(1, mean = mu_posterior, sigma = Sigma2_posterior)
  linPred <- generated_beta %*% X_input
  prob_success <- 1/(1+exp(linPred))

  # generate predictive draw
  pred_draw[i] <- rbinom(n = 1, size = 13, prob = prob_success)
}

hist(pred_draw, main = "Posterior predictive dist. for # of women not working")
```