Lab2Report

aswma317, varsi146

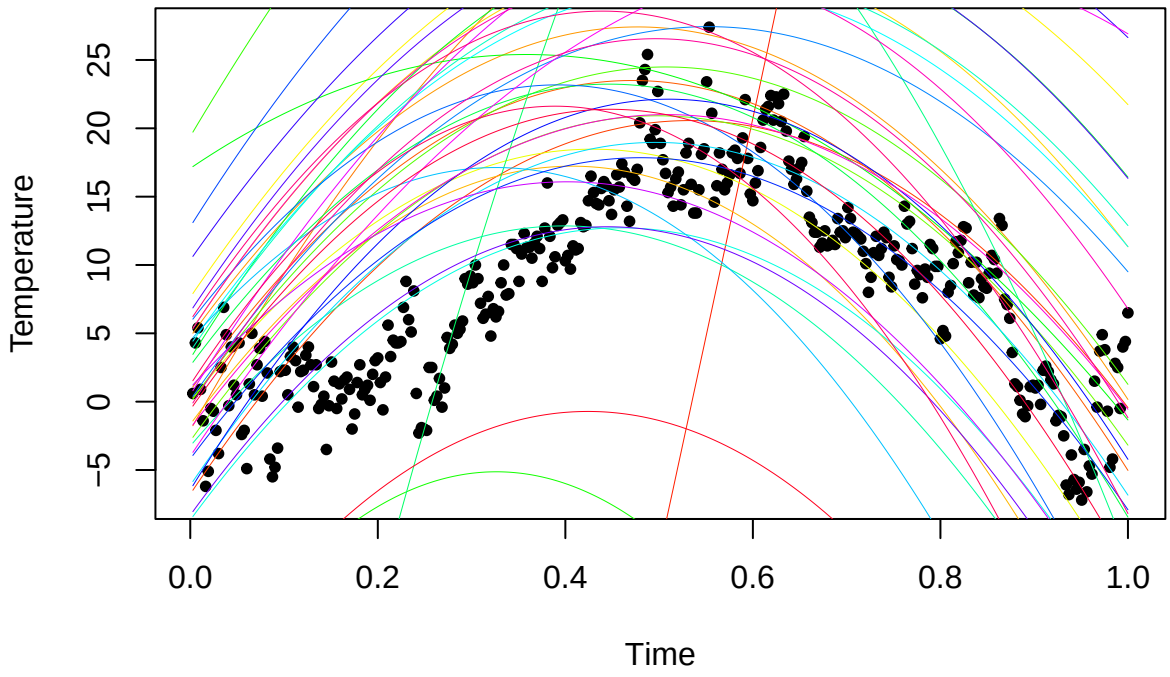
2023-04-21

Question 1

1 a).

```
####Question1: Linear Regression ####
#Given - data
data <- readxl::read_excel("Linkoping2022.xlsx")
n <- nrow(data)
k <- 3 #3covariates
data$time <- as.numeric(format(as.Date(data$datetime), "%j"))/365

#Given - starting values for prior parameters
prior_precision <- 0.01 * diag(3)#omega_Not
prior_df <- 1#new_not
prior_sd0_sq <- 1#sd_not
prior_mean <- c(0, 100, -100)

#Plot the response data for prior betas
plot(data$time, data$temp, pch = 20, xlab = "Time", ylab = "Temperature")
perform_prior_sim <- function(){
  #Part a - Get appropriate prior values that matches with the original regression
  #Step 1: Get beta_0, beta_1, beta_2 by drawing from the posterior of beta(L3, S5)
  #Step 1.a: draws from chi squared distribution
  n_sim <- 50

  plot_df <- as.data.frame(matrix(data = 0,nrow = 0, ncol = 4))
  colnames(plot_df) <- c('prior_b0', 'prior_b1', 'prior_b2', 'y_pred')

  for (i in 1:n_sim) {
    X <- rchisq(1, df = prior_df)
    #Step 1.b: compute prior_sd_sq
    prior_sd_sq <- (prior_df * prior_sd0_sq)/X
    #Step 1.c: plug prior_sd_sq into rmvnorm to get draws for betas
    prior_beta <- mvtnorm::rmvnorm(n = 1, mean = prior_mean,
                                   sigma = prior_sd_sq * solve(prior_precision))
    #print(prior_beta)
    #Step2: Get temp values from prior betas
    temp <- list(prior_beta[1] + prior_beta[2] * data$time + prior_beta[3] * data$time^2)
    plot_df <- rbind(plot_df, data.frame(
      prior_b0 = prior_beta[1],
      prior_b1 = prior_beta[2],
```

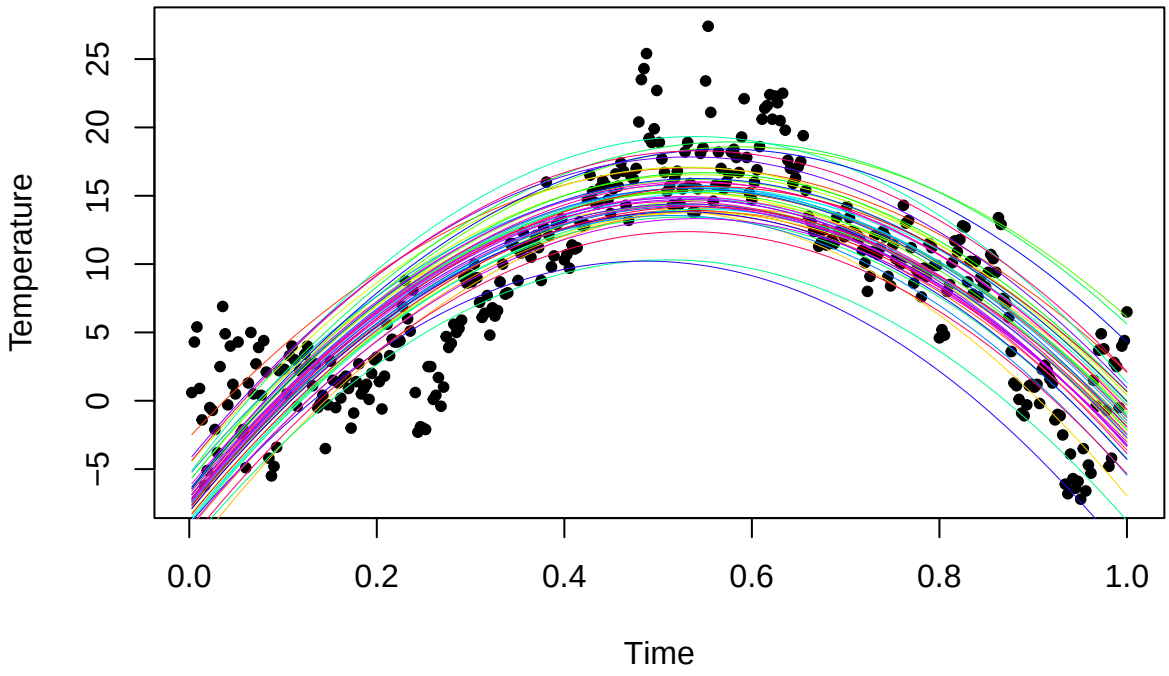
```
      prior_b2 = prior_beta[3],
      y_pred = I(temp)
      )
      )
  }
  return(plot_df)
}
plot_df <- perform_prior_sim()
colors <- rainbow(nrow(plot_df))
for (i in 1:nrow(plot_df)) {
  lines(data$time, plot_df$y_pred[[i]], col = colors[i], lwd = 0.5)
}
```



Using the starting parameters, we get the above graph. The collection of curves doesn't look reasonable. We tweaked the parameters, such that the curves are more aligned to the distribution of the data. We aligned the prior mean to the OLSE of the data, and this changes the positioning of the curves. We increased the precision so that all the curves were concentrated around the prior mean. Changing the prior degrees of the freedom ν_0 and the prior σ_0 changes the shape of the distribution from which σ^2 is drawn from.

```
prior_precision <- 6 * diag(3)#omega_Not
prior_df <- 3#new_not
prior_sd0_sq <- 12#sd_not
#prior_mean <- c(-13, 115, -110)
#Calculate X/covariate matrix
covariate <- as.matrix(cbind(
  rep(1,nrow(data)),#Column 1 with intercept
```

```
  data$time,#time
  data$time^2#time squared
))

#Calculate beta_hat/OLSE
beta_hat <- solve(t(covariate)%*%covariate) %*% (t(covariate)%*%data$temp)
prior_mean <- t(matrix(data = c(-7.532457,
                                83.578936,
                                -78.298468), nrow = 1))



#Plot the response data for prior betas
plot(data$time, data$temp, pch = 20, xlab = "Time", ylab = "Temperature")
plot_df <- perform_prior_sim()
colors <- rainbow(nrow(plot_df))
for (i in 1:nrow(plot_df)) {
  lines(data$time, plot_df$y_pred[[i]], col = colors[i], lwd = 0.5)
}
```



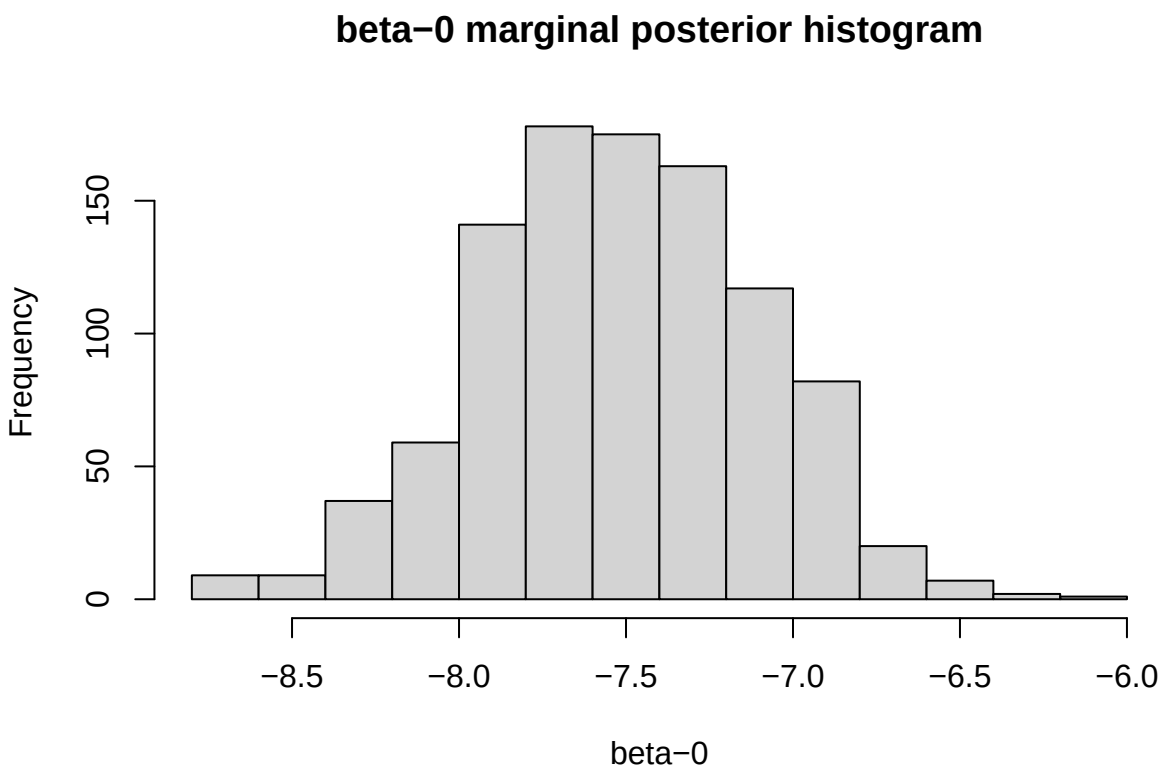
1 b (i).

draw_from_join_post_fn is the function that simulates draws from the joint posterior distribution of β_0, β_1, β_2 and σ^2

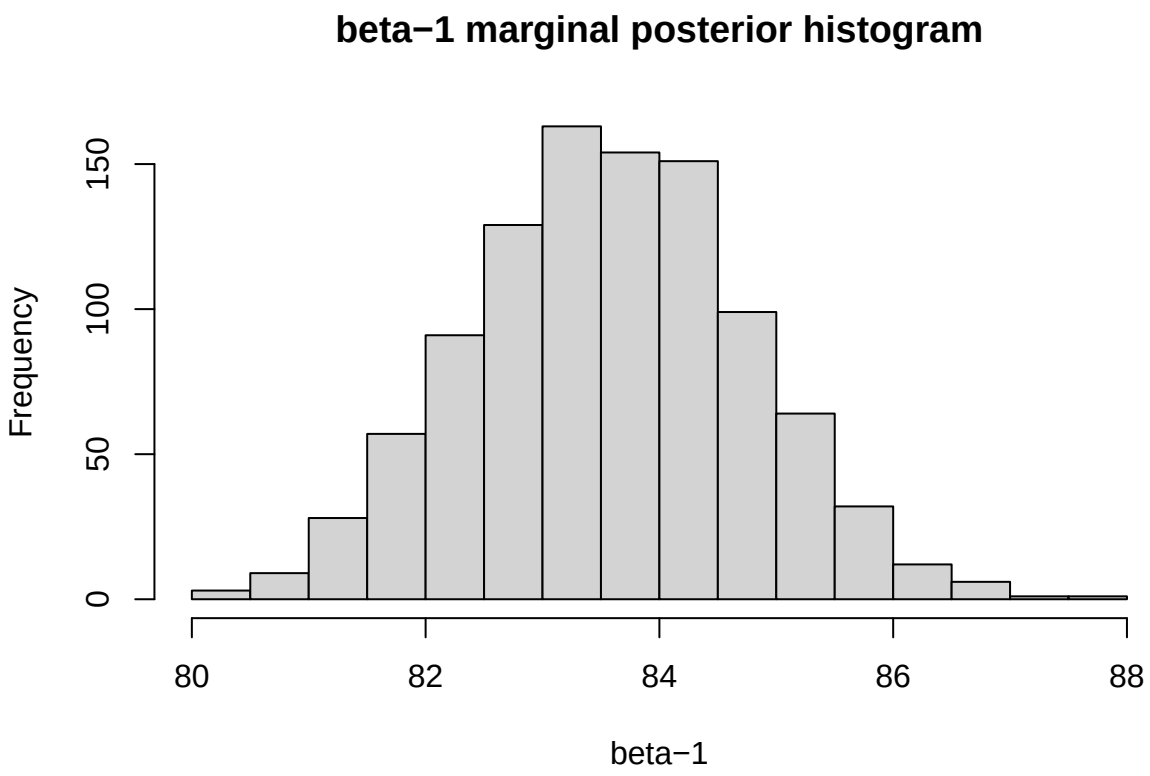
```
#Part b(i) - Function to simulate draws from joint posterior distribution of
#beta0, beta1, beta2, sd_sq & plot the marginal posterior of the parameters

#Task1: Function to simulate draws from joint posterior distribution of
#beta0, beta1, beta2, sd_sq
#Step 1: See L5, S5 to calculate the posterior parameters
#This is Linear Regression with conjugate prior
#Calculate posterior mean, precision, df, si
post_mean <- solve((t(covariate)%*%covariate) + prior_precision) %*%
  ((t(covariate)%*%covariate%*%beta_hat) + prior_precision%*%prior_mean)
post_precision <- (t(covariate) %*% covariate) + prior_precision
post_df <- prior_df+n
post_sdn_sq <- (prior_df * prior_sd0_sq +
                  ((t(data$temp)%*%data$temp) +
                     (t(prior_mean)%*%prior_precision%*%prior_mean)-
                     t(post_mean)%*%post_precision%*%post_mean))/post_df

#The mean and the sd would be similar to the prior and data values
draw_from_join_post_fn <- function(){
  #Step 2: Draw from beta0, beta1, beta2, sd_sq
  #Step 2.a: draws from chi squared distribution
  X <- rchisq(1, df = post_df)
  #Step 2.b: compute post_sd_sq
  #Step 2.b: compute prior_sd_sq
  post_sd_sq <- as.numeric((post_df * post_sdn_sq)/X)
  #Step 2.c: plug post_sd_sq into rmvnorm to get draws for betas
  post_beta <- mvtnorm::rmvnorm(n = 1, mean = post_mean,
                                sigma = (post_sd_sq) * solve(post_precision))
  return(cbind(post_beta, post_sd_sq))
}
joint_post_df <- as.data.frame(matrix(nrow = 0, ncol = 4))
joint_post_df <- rbind(joint_post_df,
                       t(sapply(1:1000, function(x) draw_from_join_post_fn())))
colnames(joint_post_df) <- c('beta_0', 'beta_1', 'beta_2', 'sd')

#Task2: plot the marginal posterior of the parameters - L5, S4 - t-distribution
#This is for non-informative prior. For informative prior,
#use the joint posterior itself.
s_sq <-  (t(data$temp - (covariate %*%beta_hat)) %*%
  (data$temp - (covariate %*%beta_hat)))/(n-k)
s_sq <- as.numeric(s_sq)
```

Following are the histograms of the marginal posterior of the parameters:

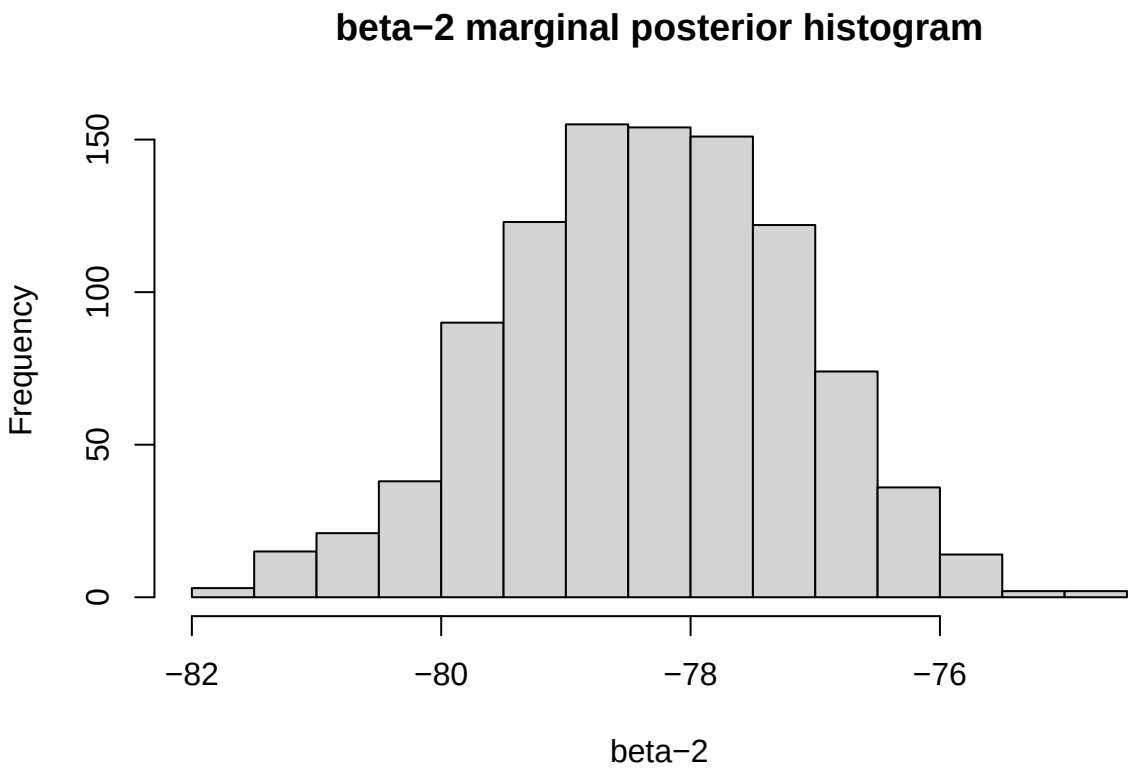
```
#Same as joint posterior distr
hist(joint_post_df$beta_0, xlab = 'beta-0',
     main = 'beta-0 marginal posterior histogram')
```



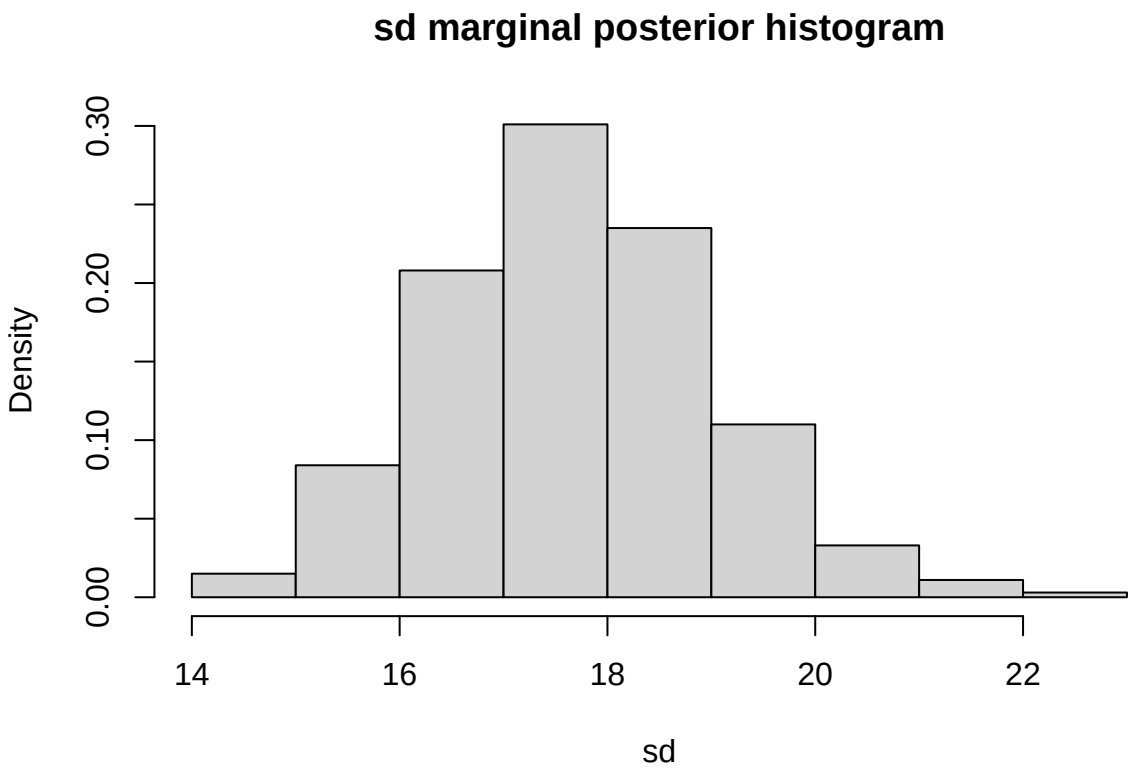

```
hist(joint_post_df$beta_1, xlab = 'beta-1',
     main = 'beta-1 marginal posterior histogram')
```




```
hist(joint_post_df$beta_2, xlab = 'beta-2',
     main = 'beta-2 marginal posterior histogram')
```



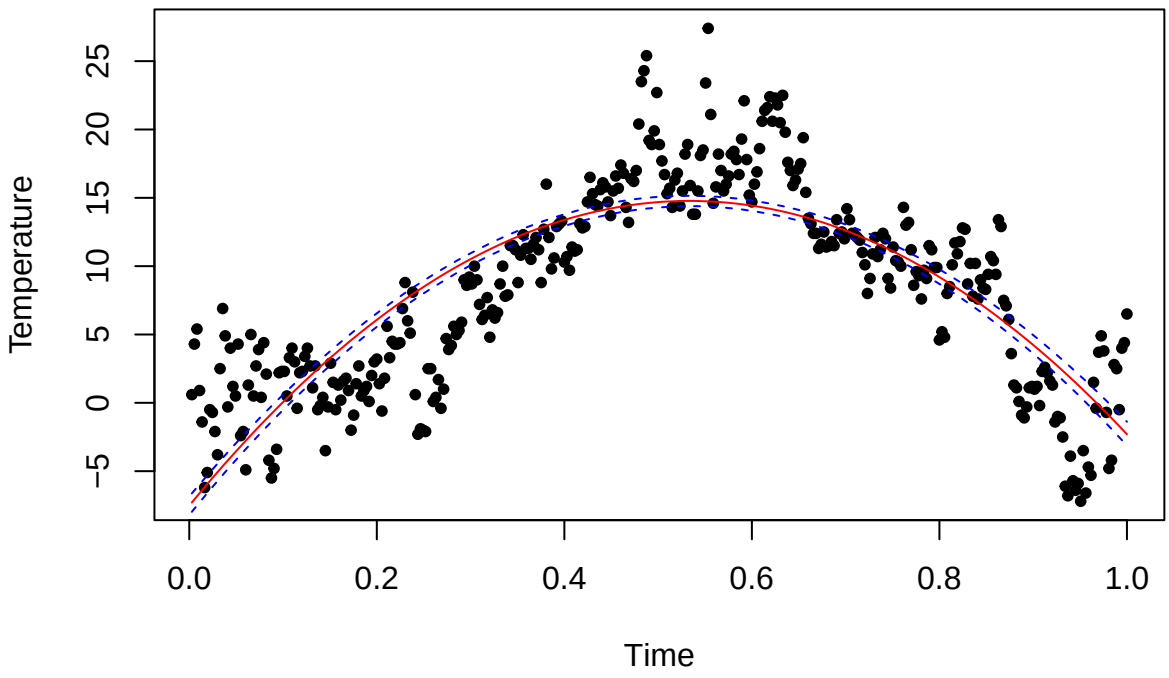

```
hist(as.numeric(joint_post_df$sd), xlab = 'sd',
     main = 'sd marginal posterior histogram', prob = TRUE)
```



1 b (ii).

Following is the scatter plot of the temperature data with an overlay of a curve for the posterior median of the regression function $f(time)$, along with the 90% equi-tailed posterior probability interval.

```
#Part b(ii) - Plot the posterior median of the regression with the 5% and 95%
#posterior percentiles
#Marginal posterior beta %*% covariates gives the posterior response
pos_responses <- as.matrix(joint_post_df[,1:3]) %*% t(covariate)#1column for each day/x
pos_response_median <- apply(pos_responses, 2, median)
plot(data$time, data$temp, pch = 20, xlab = "Time", ylab = "Temperature")
lines(data$time, pos_response_median, col = 'red')#Median line
#Get the 90% equi-tailed posterior probability
pos_responses_ci = apply(pos_responses, 2, quantile, probs = c(0.05, 0.95))
lines(data$time, pos_responses_ci[1,], col = 'blue', lty = 2)#5% CI
lines(data$time, pos_responses_ci[2,], col = 'blue', lty = 2)#95% CI
```



As seen above, the posterior probability intervals do not contain most of the data points as there is a lot of noise in the data and the 90% intervals is not able to capture all these data points. And this is expected that these interval capture all the data points, as there can be noise or outliers in the data.

1 c.

To get the time which gives the highets expected temperature, we can analytically solve this by taking the derivative of the f(temp) w.r.t time, as shown below:

$temp = \beta_0 + \beta_1 \cdot time + \beta_2 \cdot time^2$

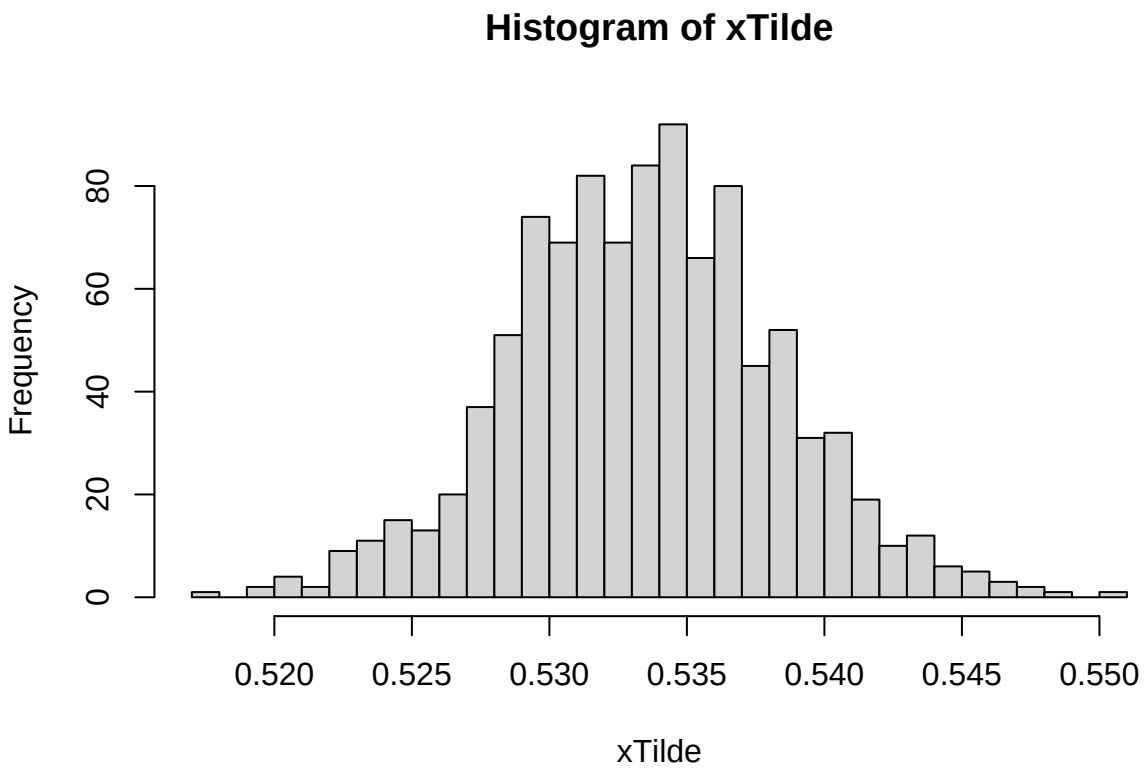
$\frac{\partial temp}{\partial time} = \frac{\partial(\beta_0 + \beta_1 \cdot time + \beta_2 \cdot time^2)}{\partial time} = \beta_1 + 2\beta_2$

Equating this to 0 will gives the time with maximum temp

$time = \frac{-\beta_1}{2 * \beta_2}$

We use the simulated posterior distribution from above to get the time which gives highest temp.

```
#Part c: Find the time(covariate) which gives the maximum temp(response)
calc_x <- function(beta) {
  #Get the formula by taking the derivative of the reg expression w.r.t covariate
  #and equating it to 0 - optimization problem
  return(-beta[2] / (2 * beta[3]))
}
xTilde <- apply(joint_post_df[,1:3], 1, calc_x)
hist(xTilde, breaks = 25)
```



1 d.

Suggest a suitable prior to estimate a polynomial regression of order 10

Answer: The following is a suitable prior:

$$\beta \mid \sigma^2 \overset{iid}{\sim} \mathcal{N}(\mu_0, \frac{\sigma^2}{\lambda})$$

where $\mu_0 = 0$ and $\Omega_0 = \lambda \cdot I$

Question 2

2 a).

```
#Given
data <- read.table("WomenAtWork.dat", header = TRUE)
n <- nrow(data)
X <- as.matrix(data[,2:ncol(data)])
y <- data[,1]
Xnames <- colnames(X)
Npar <- dim(X)[2]
# Setting up the prior
mu <- as.matrix(rep(0,Npar)) # Prior mean vector
```

```r
tau <- 2
Sigma <- (tau^2)*diag(Npar) # Prior covariance matrix

#Part a(i): Draw samples from posterior distribution - L6,S5
#Step1: Calculate the mean, beta_hat using numerical optimization - optim
# Functions that returns the log posterior for the logistic and probit regression.
# First input argument of this function must be the parameters we optimize on,
# i.e. the regression coefficients beta.
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 h
  logPrior <- dmvnorm(betas, mu, Sigma, log=TRUE);

  return(logLik + logPrior)
}
# Select the initial values for beta
initVal <- matrix(0,Npar,1)
# The argument control is a list of options to the optimizer optim,
# where fnscale=-1 means that we minimize
# the negative log posterior. Hence, we maximize the log posterior.
OptimRes <- optim(initVal,
                  LogPostLogistic,
                  gr=NULL,y, X, mu, Sigma,
                  method=c("BFGS"),
                  control=list(fnscale=-1),#Maximize log posterior
                  hessian=TRUE)#Returns the hessian
beta_hat <- OptimRes$par
print('The posterior mode is:')
```

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

```r
print(beta_hat)
```

```
##             [,1]
## [1,] -0.04036943
## [2,] -0.03730689
## [3,]  0.17868950
## [4,]  0.12073637
## [5,] -0.04618995
## [6,] -1.47248930
## [7,] -0.02014458
```

```r
approxPostStd <- sqrt(diag(solve(-OptimRes$hessian))) # Computing approximate standard deviations.
print('The approximate posterior standard deviation is:')
```

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

```r
print(approxPostStd)
```

```
## [1] 1.38198486 0.02198474 0.08920960 0.03335982 0.02747315 0.47746764 0.16401959
```

```
print('Values of inverse of observed information at mode')
```

```
## [1] "Values of inverse of observed information at mode"
```

```
solve(-OptimRes$hessian)
```

```
##              [,1]          [,2]          [,3]          [,4]          [,5]
## [1,]  1.909882159  4.032517e-03 -6.280726e-02  1.041874e-03 -0.0257559994
## [2,]  0.004032517  4.833287e-04 -9.147892e-04 -2.666479e-05 -0.0000642848
## [3,] -0.062807260 -9.147892e-04  7.958354e-03  5.508998e-05 -0.0003181372
## [4,]  0.001041874 -2.666479e-05  5.508998e-05  1.112877e-03 -0.0002845111
## [5,] -0.025755999 -6.428480e-05 -3.181372e-04 -2.845111e-04  0.0007547741
## [6,] -0.137712005  1.585545e-03 -1.438778e-02 -1.336628e-03  0.0055481315
## [7,] -0.088876440  4.986972e-06  1.133513e-04  7.206537e-04  0.0010449347
##              [,6]          [,7]
## [1,] -0.137712005 -8.887644e-02
## [2,]  0.001585545  4.986972e-06
## [3,] -0.014387780  1.133513e-04
## [4,] -0.001336628  7.206537e-04
## [5,]  0.005548132  1.044935e-03
## [6,]  0.227975343  1.122711e-02
## [7,]  0.011227114  2.690243e-02
```

```
post_samples <- mvtnorm::rmvnorm(n = 1000,
                                 mean = beta_hat,
                                 sigma = solve(-OptimRes$hessian)
                                 )
# Part a(ii):Compute an approximate 95% equal tail posterior probability interval
# for the regression coefficient to the variable NSmallChild
post_NSmallChild_CI <- quantile(post_samples[,6], probs = c(0.025, 0.975))
print('Lower bound and upper bound for the 95% equi-tailed posterior prob
      interval for regression coefficient of NSmallChild')
```

```
## [1] "Lower bound and upper bound for the 95% equi-tailed posterior prob \n      interval for regressi
```

```
post_NSmallChild_CI
```

```
##       2.5%      97.5%
## -2.3970550 -0.6078829
```

```
glmModel <- glm(Work ~ 0 + ., data = data, family = binomial)
print('Difference between posterior parameters and parameters calaculated from glm')
```

```
## [1] "Difference between posterior parameters and parameters calaculated from glm"
```

```
abs(glmModel$coefficients - apply(post_samples, 2, mean))
```

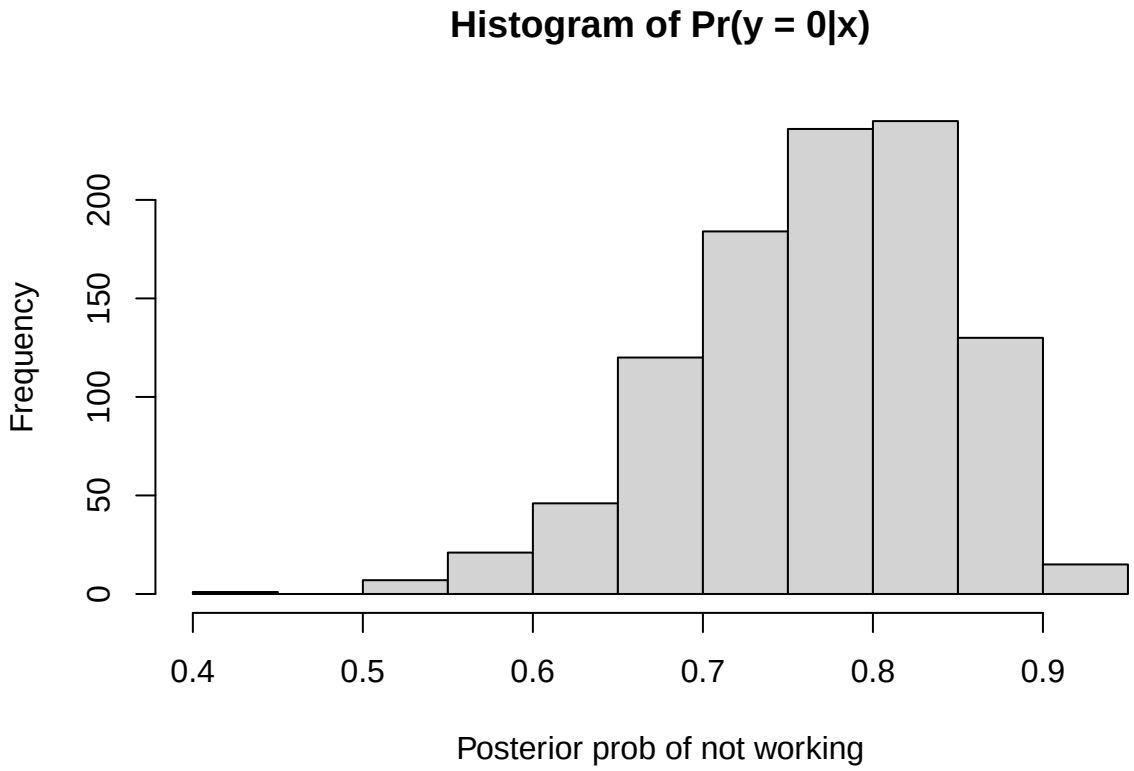
```
##     Constant   HusbandInc    EducYears     ExpYears          Age  NSmallChild
## 0.0483137852 0.0008442890 0.0071968890 0.0002520907 0.0022172052 0.0857464871
##    NBigChild
## 0.0123490848
```

Would you say that this feature is of importance for the probability that a woman works?

Answer: Since the 95% equal tail posterior probability interval for the regression coefficient to the variable NSmallChild is between -2.5 and -.55, and doesn't cover 0. If the coefficient value is 0, then the variable would not have any influence on the response

2 b).

```
# Part b: Write a function that simulate draws from the  posterior predictive
# distribution of Pr(y = 0|x), where the values of x are given
prob_not_working <- function(post_samples){
  new_X <- matrix(c(1, 18, 11, 7, 40, 1, 1), ncol = 1)
  linPred <- post_samples%*%new_X
  post_prob_working <- exp(linPred)/(1+exp(linPred))
  post_prob_not_working <- (1-post_prob_working)
  return(post_prob_not_working)
}
post_prob_not_working <- prob_not_working(post_samples)
hist(post_prob_not_working, main='Histogram of Pr(y = 0|x)',
     xlab = 'Posterior prob of not working')
```



2 c).

```
# Part c: plot the posterior predictive distribution for the number of women,
# out of these 13, that are not working
post_no_not_working_fn <- function(post_samples){
  new_X <- matrix(c(1, 18, 11, 7, 40, 1, 1), ncol = 1)
  linPred <- post_samples%*%new_X
  post_prob_working <- exp(linPred)/(1+exp(linPred))
  post_prob_not_working <- (1-post_prob_working)
  post_no_not_working <- rbinom(n = length(post_prob_not_working),
                                size = 13, prob = post_prob_not_working)
  return(post_no_not_working)
}
post_13_not_working <- post_no_not_working_fn(post_samples)
hist(post_13_not_working, main='Histogram of n women not working out of 13',
     xlab = 'n women not working')
```