Bayesian Learning - Lab2

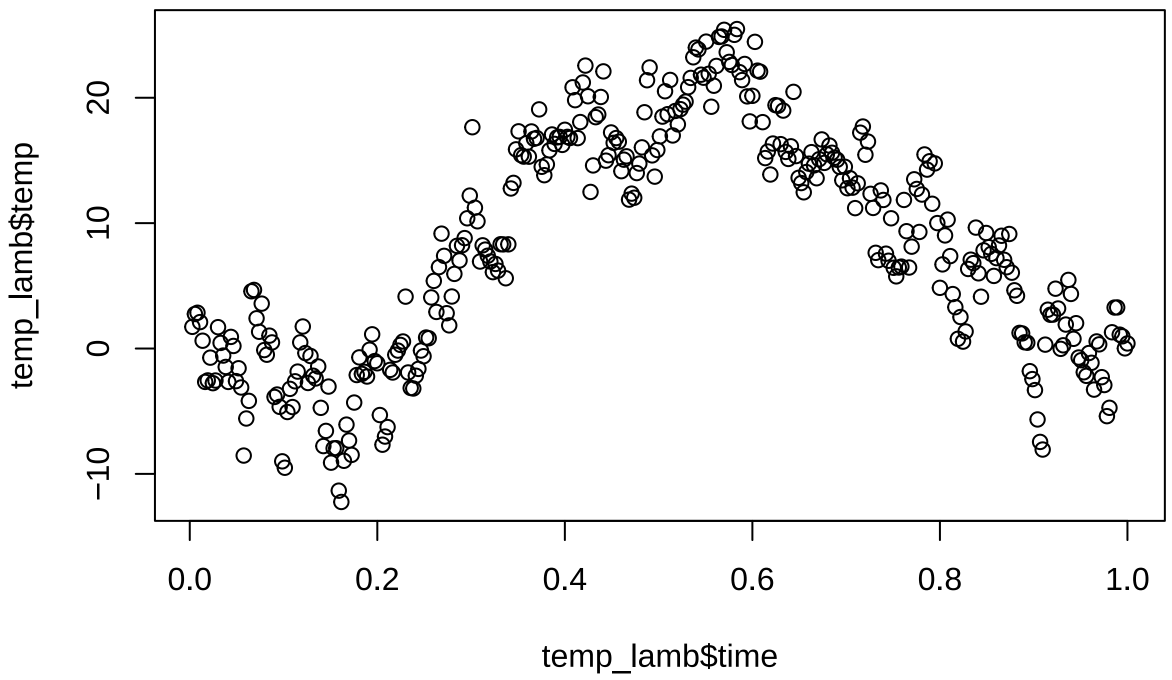
Jaskirat S Marar & Dinuke Jayaweera

4/14/2022

Question1: Polynomial Regression

Part A

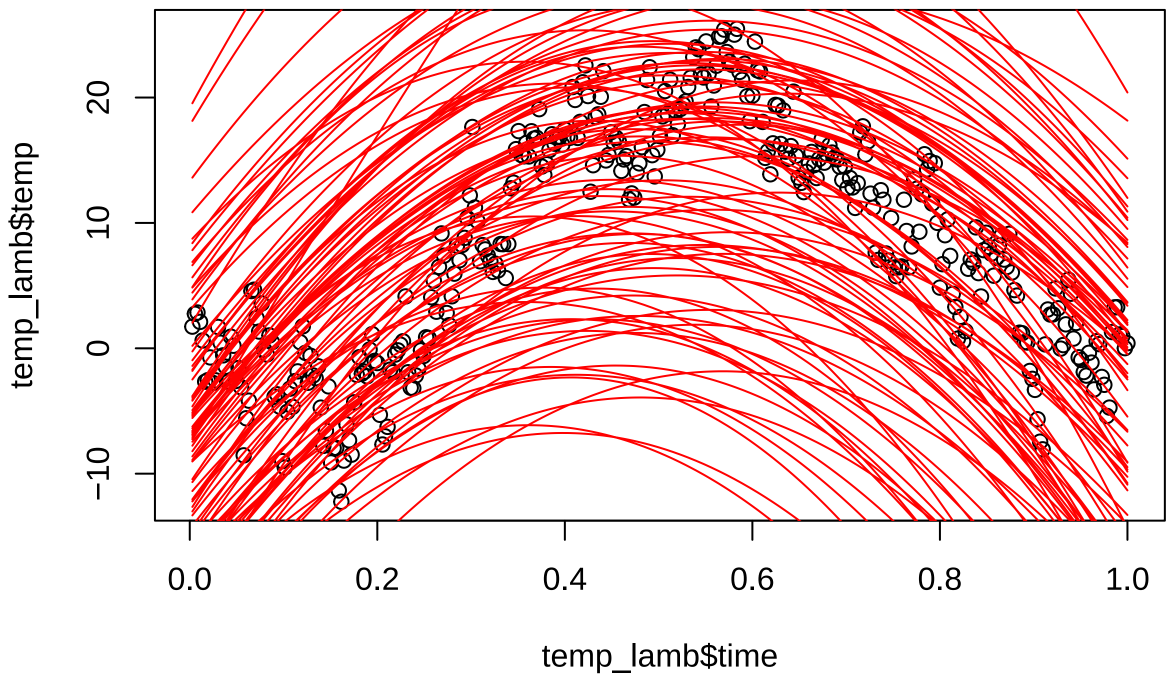
```r
# Question 1

# PART A

#read the temperature data
temp_lamb <- read.table("TempLambohov.txt", header = TRUE)
plot(temp_lamb$time, temp_lamb$temp)
```



```r
#set the starting parameters
```

```r
mu_0 <- c(-10,100,-100)
omega_0 <- 0.02 * diag(3)
nu_0 <- 3
sigma2_0 <- 2

Y <- temp_lamb$temp
X <- cbind(1, temp_lamb$time, temp_lamb$time^2)

#prior draw

prior_fn <- function(covariates = X,
                     mu = mu_0,
                     omega = omega_0,
                     sigma2 = sigma2_0,
                     nu = nu_0) {

  #Inverse chi sq draw for variance
  prior_sigma2 <- rchisq(1, df = nu) #draw from chi-sq with nu df
  prior_sigma2 <- nu * sigma2/prior_sigma2 #compute for inv-chi-sq draws

  #prior_sigma2 <- rinvchisq(n = 1, df = nu, scale = sigma2)

  prior_beta <-rmvnorm(1, mean = mu, sigma = prior_sigma2 * solve(omega))

  prior_final <- covariates %*% t(prior_beta)

  return(prior_final)

}
```

```r
plot(temp_lamb$time, temp_lamb$temp)
for (i in 1:100) {
  lines(x = temp_lamb$time, y = prior_fn(), col = 'red')
}
```

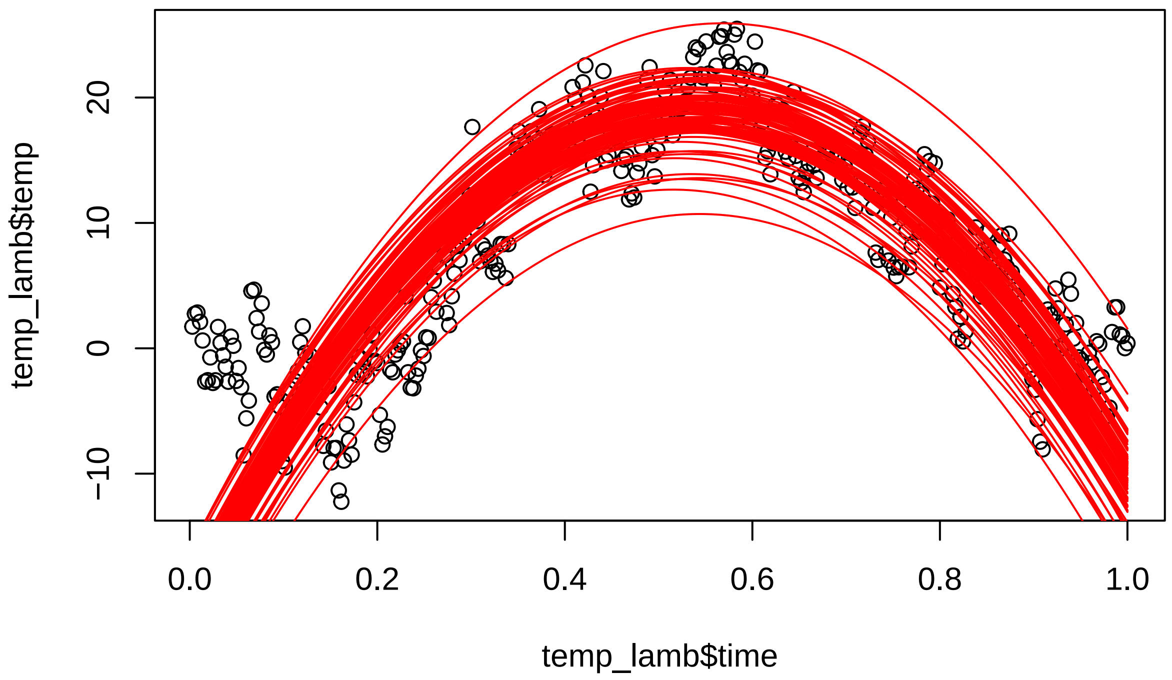


Initial values of the hyperparameters actually did not result in a set of regression curves that match the data distribution. So we played around quite a bit with the hyperparameters and observed the following about each hyper parameter:

1. Ω : the initial value was pretty small and resulted in a set of curves that whose parabola did not align at all. With higher values we started getting more close fitting parabolas, but with too high a value, the resulting parabolas were overlapping too tightly, so we settled on a value of 0.07.
2. ν : A smaller starting value was resulting in a higher variation in the width of the parabolas, so we settled on a slightly higher value of 5.
3. σ^2 : The initial assumption resulted in highly varied parabolic curves, so we cut down the value of the hyper parameter to 0.15
4. μ : This was more tricky to work out. The 3 values served as positional parameters for the curve and after a bit of playing around with the values, we settled on (-20, 145, -135) as the final setting.

With these settings, our regression curves looked like this:

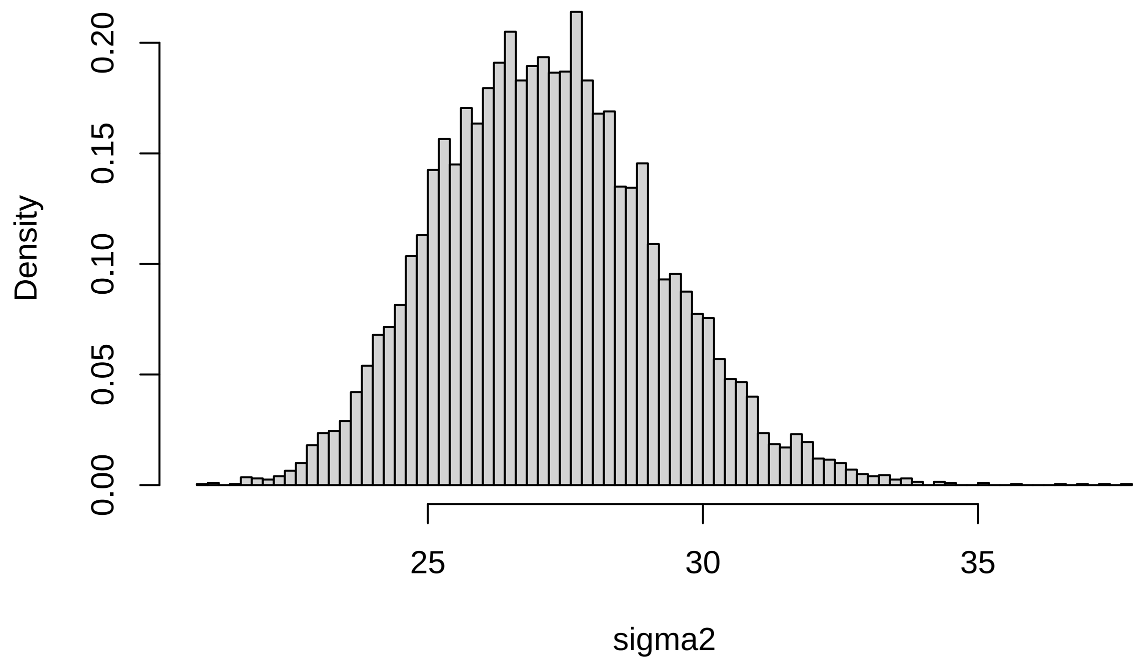
```r
plot(temp_lamb$time, temp_lamb$temp)
for (i in 1:100) {
  lines(x = temp_lamb$time,
        y = prior_fn(mu= c(-20,145,-135),
                     omega = 0.07 * diag(3),
                     nu = 5,
                     sigma2 = 0.15),
        col = 'red')
}
```



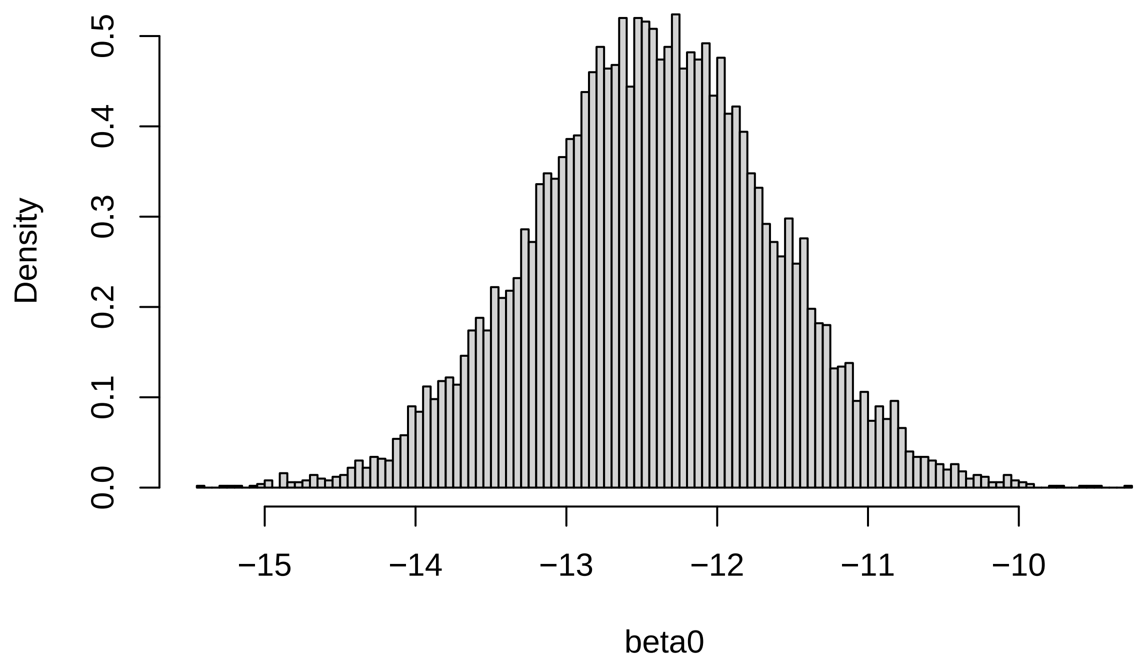
```
#setting new initial params

mu_set= c(-20,145,-135)
omega_set = 0.07 * diag(3)
nu_set = 5
sigma2_set = 0.15
```

Part B.i)

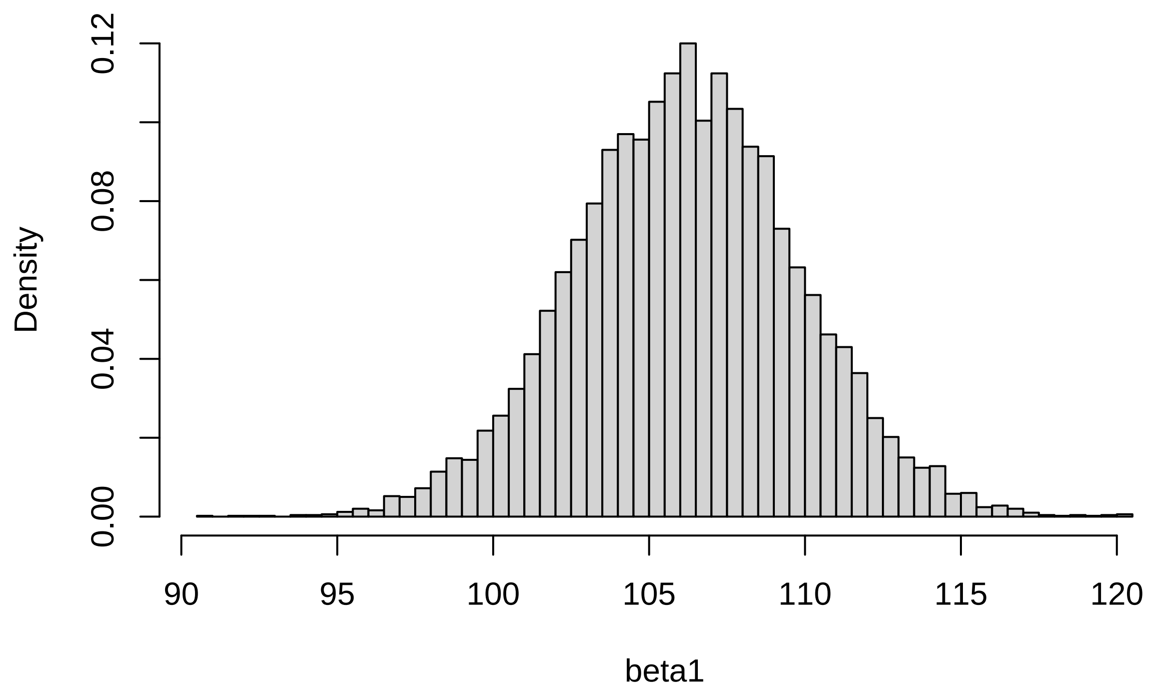
```
#PART B.1.

#Joint Posterior Distribution

joint_posterior <- function(covariates = X,
                            responses = Y,
                            omega = omega_set,
                            mu = mu_set,
                            nu = nu_set,
                            sigma2 = sigma2_set) {

  #Setup new params for Posterior

  beta_hat <- solve(t(covariates) %*% covariates) %*% t(covariates) %*% responses
  mu_n <- (solve(t(covariates) %*% covariates + omega)) %*% (t(covariates) %*% covariates %*% beta_hat +
  omega_n <- t(covariates) %*% covariates + omega
  nu_n <- nu + length(responses)
  sigma2_n <- 1/nu_n * (nu * sigma2 + t(responses) %*% responses + t(mu) %*% omega %*% mu - t(mu_n) %*%


  #marginal posteriors of betas and variance

  #sigma_posterior <- rinvchisq(n = 10000, df = nu_n, scale = as.numeric(sigma2_n))
  sigma_posterior <- rchisq(10000, df = nu_n) #draw from chi-sq with nu_n df
  sigma_posterior <- nu_n * as.numeric(sigma2_n)/sigma_posterior #compute for inv-chi-sq draws

  beta_posterior <- rmvnorm(n = 10000, mean = mu_n, sigma = as.numeric(sigma2_n) * solve(omega_n))

  return(cbind.data.frame(sigma_posterior, beta_posterior))

}

posterior <- joint_posterior()

hist(posterior[,1], breaks = 100, probability = TRUE, xlab = "sigma2", main = NULL)
```



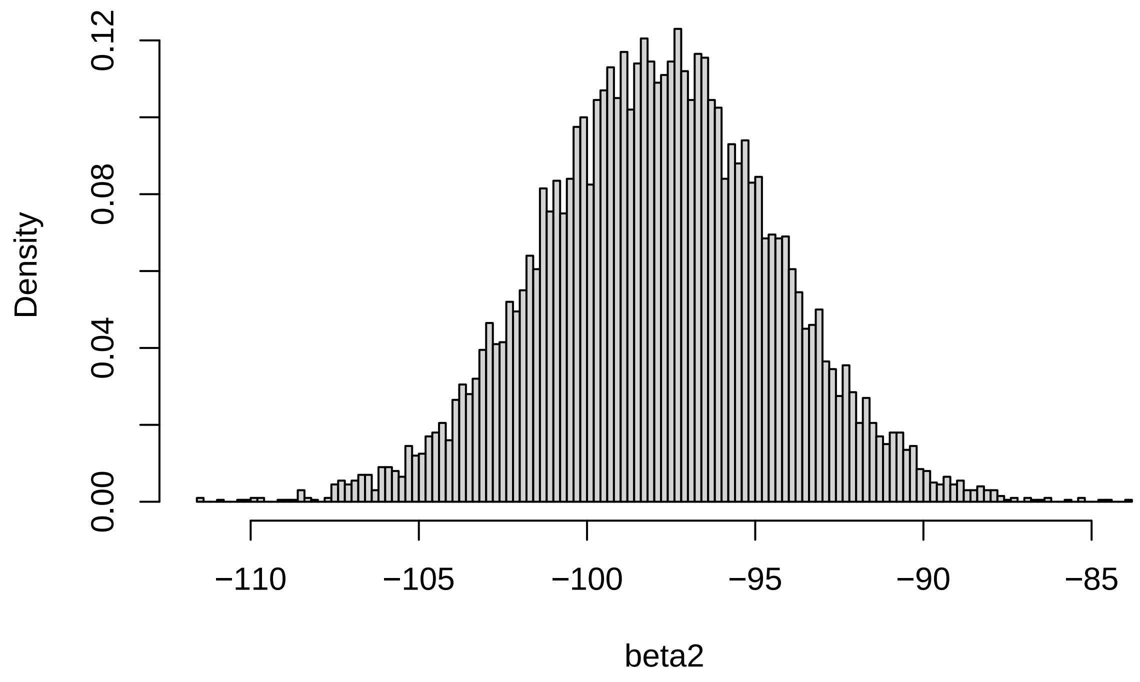
```
hist(posterior[,2], breaks = 100, probability = TRUE, xlab = "beta0", main = NULL)
```



```r
hist(posterior[,3], breaks = 100, probability = TRUE, xlab = "beta1", main = NULL)
```

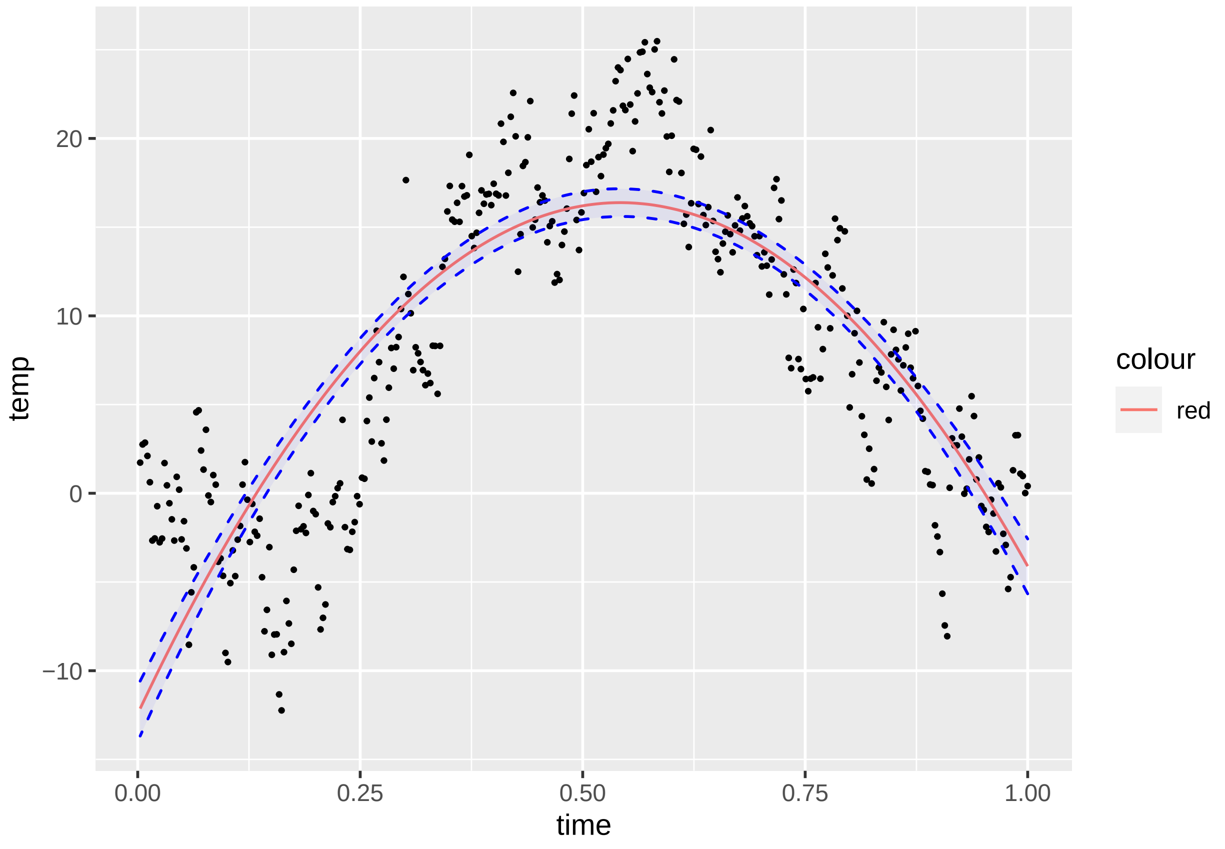


```r
hist(posterior[,4], breaks = 100, probability = TRUE, xlab = "beta2", main = NULL)
```



Part B.ii)

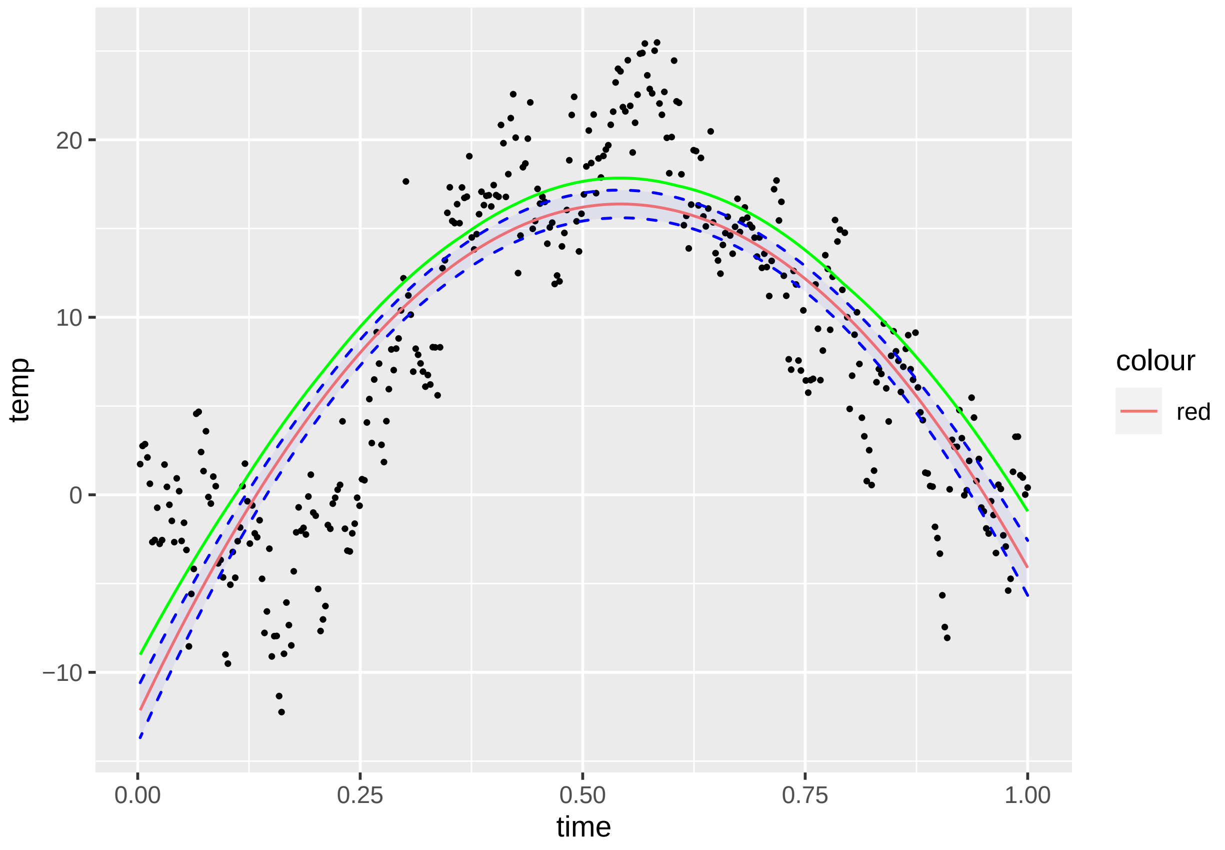
```r
#PART B.2.

beta_post <- posterior[,-1]

pred_all <- as.matrix(beta_post) %*% t(X)

pred_median <- apply(pred_all, 2, FUN = median)

temp_lamb_pred <- temp_lamb
temp_lamb_pred$pred_median <- pred_median

pred_pi <- data.frame(lower = double(), upper = double())

for (i in 1:length(pred_median)) {
    pred_pi[i,] <- quantile(pred_all[,i], c(0.025,0.975))
}

temp_lamb_pred <- cbind.data.frame(temp_lamb_pred, pred_pi)

gp <- ggplot(data = temp_lamb_pred, aes(x = time)) +
  geom_point(aes(y = temp), size = 0.5) +
  geom_line(aes(y = pred_median, color = "red")) +
  geom_ribbon(aes(ymin = lower, ymax = upper),
              alpha = 0.05,
              color = "blue",
              fill = "blue",
              linetype = "dashed")

gp
```



From plotting the PI we can see that most of the points are not lying inside the PI. The explanation for this is that this PI corresponds to the variability of the median of the predictions i.e. the median of all the regression lines. But the true prediction also requires that we add an error term to the regression equation. Since in this model we haven't added any error term anywhere till now, we have also not accounted for the variability of the error term. Hence, our prediction bands don't cover most of the data points.

Part C

```
#PART C

max_preds <- apply(pred_all, 2, max)

temp_lamb_pred$max_pred <- max_preds

ggplot(data = temp_lamb_pred, aes(x = time)) +
  geom_point(aes(y = temp), size = 0.5) +
  geom_line(aes(y = pred_median, color = "red")) +
  geom_ribbon(aes(ymin = lower, ymax = upper),
              alpha = 0.05,
              color = "blue",
              fill = "blue",
              linetype = "dashed") +
  geom_line(aes(y = max_pred), color = "green")
```



Part D

To estimate a higher order polynomial regression, we can use a normal shrinkage prior that help us deal with the problem of multiple knots (as in this case) leading to overfitting. The general form of the normal prior is:

$$\beta_j|\sigma^2 \sim N(\mu_0, \frac{\sigma^2}{\lambda \cdot I_8})$$

For μ_0 we suggest that we can use suitable magnitudes for the lower order β_j but for higher order β_j we can use a mean value close to 0 or even equal to 0.

For the variance, the shrinkage factor would be $\Omega_0 = \lambda \cdot I_8$. We can recycle values from our earlier experiments for the lower order β_j and for higher order β_j we can choose a smaller variance i.e. close to 0 around the already negligible mean values by choosing very high values for Ω_0 as it will go into the denominator.

This way we can regularize the higher order β_j

Question 2: Logisitic Regression

PART A

```r
# Question 2

rm(list = ls())

# PART A

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

#setting up data & parameters for prior

#data
X <- as.matrix(women_work[,2:8])
Y <- as.matrix(women_work[,1], NCOL = 1)

#prior
nfeatures <- dim(X)[2]
tau <- 5
mu <- as.matrix(rep(0,nfeatures)) # Prior mean vector
sigma <- (tau^2)*diag(nfeatures) # Prior covariance matrix
```

Since our model is of the logistic form:

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

We will have to derive the function for the posterior as follows:

Likelihood:

$$P(Y|X, \beta) = \prod_{i=1}^{n} \frac{[exp(x_i^T\beta)]^{y_i}}{1 + exp(x_i^T\beta)}$$

taking logs and simplifying:

$$log(P(Y|X, \beta)) = \sum_{i=1}^{n} \left[y_i \cdot (x_i^T \cdot \beta) - log(1 + exp(x_i^T\beta)) \right]$$

Prior:

$$\beta \sim N(0,\ \tau^2 I_{nfeatures})$$

Hence prior will be drawn from the multivariate normal distribution with the specified parameters

The final posterior function will written as a sum of the log liklihood and multivariate normal prior

```r
#loglikelihood function

logPosterior <- function(betas, Y, X, mu, sigma){
  LL <- sum((X %*% betas) * Y - log(1 + exp(X %*% betas)))
  LPrior <- dmvnorm(betas, mu, sigma, log=TRUE)
  return(LL + LPrior)
}
```

We'll now utilize the posterior function to compute the posterior mode $(\tilde{\beta})$ and Hessian $(-J_y(\tilde{\beta}))$

```r
#initialize beta for optim
beta_init <- matrix(0, nfeatures, ncol = 1)

#maxinmize logPosterior
OptimRes <- optim(beta_init,
                  logPosterior,
                  gr=NULL,
                  Y,X,mu,sigma,
                  method=c("BFGS"),
                  control=list(fnscale=-1),
                  hessian=TRUE)

#posterior mode
beta_mode <- OptimRes$par
names(beta_mode) <- colnames(X)

#Hessian is -J(beta)
beta_jacobian <- -OptimRes$hessian

#diagonal of hessian is the second derivates corresponding to the covariance
beta_inverse_jacobian <- solve(beta_jacobian)
colnames(beta_inverse_jacobian) <- colnames(X)

cat("\n", "The posterior mode is: ", "\n")
```

```
##
##  The posterior mode is:
```

```r
print(beta_mode)
```

```
##             [,1]
## [1,]  0.99295290
## [2,] -0.03435024
## [3,]  0.17941763
## [4,]  0.12306284
## [5,] -0.07279229
## [6,] -1.62277354
## [7,] -0.08388832
## attr(,"names")
## [1] "Constant"    "HusbandInc"  "EducYears"   "ExpYears"    "Age"
## [6] "NSmallChild" "NBigChild"
```

```r
cat("\n", "The Inverse Jacobian is: ", "\n")
```

```
##
##  The Inverse Jacobian is:
```

```r
print(beta_inverse_jacobian)
```

```
##          Constant    HusbandInc     EducYears      ExpYears           Age
## [1,]  2.529263305  3.937069e-03 -7.750733e-02  8.892590e-04 -3.410000e-02
## [2,]  0.003937069  3.914736e-04 -8.068840e-04 -1.050793e-06 -5.158944e-05
## [3,] -0.077507334 -8.068840e-04  7.341507e-03  6.096778e-05  4.959525e-05
## [4,]  0.000889259 -1.050793e-06  6.096778e-05  9.005838e-04 -2.492037e-04
## [5,] -0.034099996 -5.158944e-05  4.959525e-05 -2.492037e-04  8.071705e-04
## [6,] -0.228103267  1.149967e-03 -8.024499e-03 -9.930033e-04  6.095688e-03
## [7,] -0.113700817 -6.395369e-05  1.925629e-03  6.121508e-04  1.278748e-03
##        NSmallChild     NBigChild
## [1,] -0.2281032670 -1.137008e-01
## [2,]  0.0011499674 -6.395369e-05
## [3,] -0.0080244993  1.925629e-03
## [4,] -0.0009930033  6.121508e-04
## [5,]  0.0060956879  1.278748e-03
## [6,]  0.1918381358  9.420943e-03
## [7,]  0.0094209427  2.221631e-02
```

To compute the 95% ETI for posterior probability of the beta corresponding to variable NSmallChild, we will use the above computed parameters and draw a large sample from the normal posterior. We know from the slides that the approximate normal posterior in large samples is:

$$\theta|y \; approx. \sim N[\tilde{\theta}, J_y^{-1}(\tilde{\theta})]$$

```
#draw large sample for beta corresponding to NSmallChild i.e. beta6
beta_sim <- rmvnorm(n = 1000, mean = beta_mode, sigma = solve(beta_jacobian))

#isolate draws for beta6
beta6_sim <- beta_sim[,6]

#95% PI for beta6
beta6_pi <- quantile(beta6_sim, prob = c(0.025,0.975))
names(beta6_pi) <- c("Lower", "Upper")

cat("\n", "The PI for Beta_nSmallChild:", "\n")
```

```
##
##  The PI for Beta_nSmallChild:
```

```
print(beta6_pi)
```

```
##      Lower      Upper
## -2.5197956 -0.7099312
```

Since, the PI doesn't contain 0, we can say that the null hypothesis that $\beta_6 = 0$ is rejected and that this feature is significant. We have already seen that this feature has the largest magnitude for its beta coefficient with a negative sign. Hence, number of small children has a negative impact on working women.

We can compare this with the results of the glm model:

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

summary(glmModel)
```

```
##
## Call:
## glm(formula = Work ~ 0 + ., family = binomial, data = women_work)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2858  -0.9512   0.4042   0.9588   2.0847
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## Constant       1.12243    1.68159   0.667 0.504466
## HusbandInc  -0.03425    0.01981  -1.729 0.083730 .
## EducYears    0.17651    0.08724   2.023 0.043054 *
## ExpYears     0.12317    0.03003   4.102  4.1e-05 ***
## Age         -0.07475    0.02944  -2.539 0.011110 *
## NSmallChild -1.64598    0.44450  -3.703 0.000213 ***
## NBigChild   -0.08973    0.15132  -0.593 0.553189
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 232.9  on 168  degrees of freedom
## Residual deviance: 185.9  on 161  degrees of freedom
## AIC: 199.9
##
## Number of Fisher Scoring iterations: 4
```

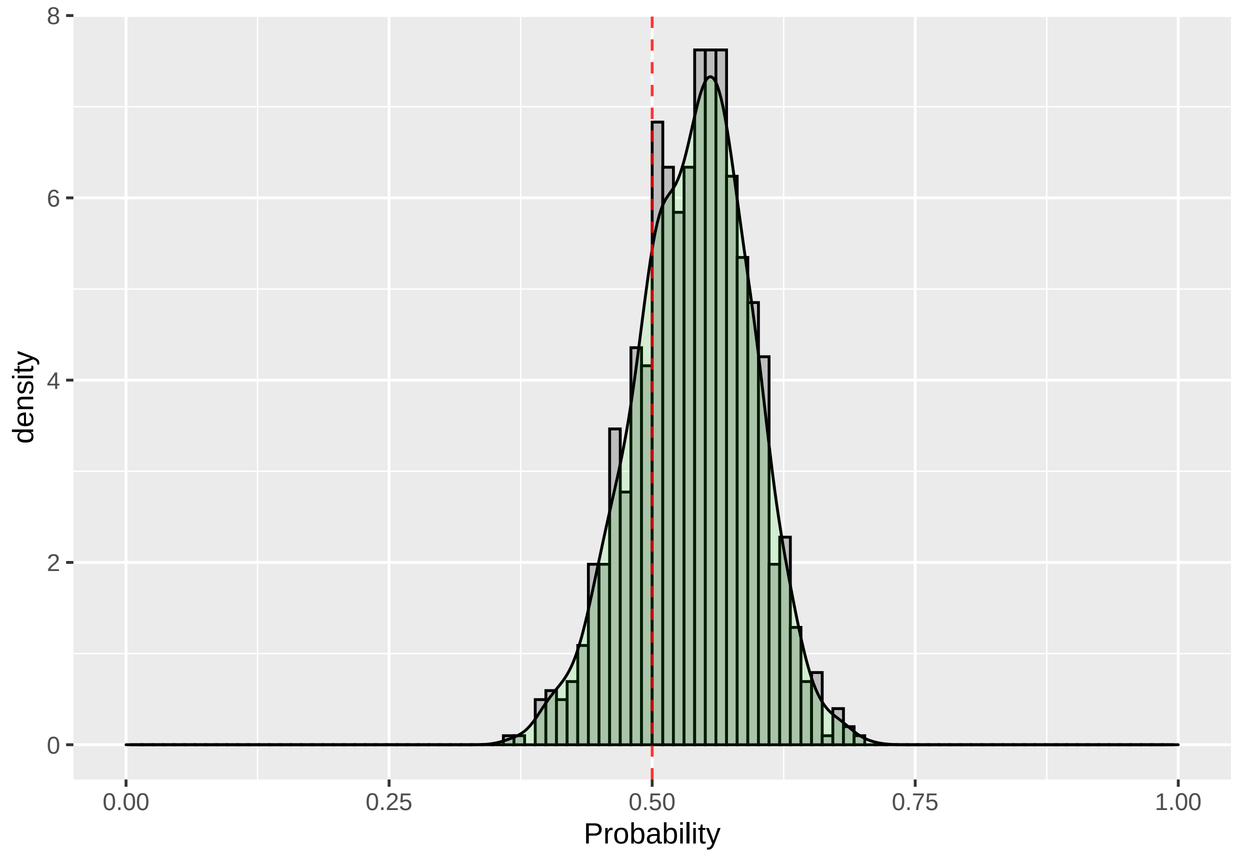
The glm model also agrees with our findings that the this is an important feature to the model.

PART B

We'll be reusing the simulated posterior samples of β_j with the given sample features to find a prediction sample for all sample values of betas. We use the sigmoid function to convert all the predictions into probabilities and plot the histogram to see the distribution of the prediction probability. We will use the decision boundary at 0.5 to model $Pr(y = 1|x)$

```r
#Part B

#Create matrix of sample features
X_data <- as.matrix(c(1, 20, 12, 8, 43, 0, 2))

#reuse the betas simulated earlier
#use sigmoid function to model probabilities of predictions

post_pred <- function(X, beta) {

  pred <- beta %*% X #linear predictions

  pred_logit <- exp(pred)/(1 + exp(pred)) #sigmoid function to model probabilities

  return(data.frame(Probability = pred_logit))
}

#function call on sample data
post_pred_sim <- post_pred(X_data, beta_sim)

#plotting the posterior prediction density & histogram
ggplot(data = post_pred_sim, aes(x = Probability)) + xlim(c(0,1)) +
  geom_histogram(aes(y = ..density..),
                 bins = 100,
                 color = "black",
                 fill = "grey") +
  geom_density(alpha = 0.1, fill = "green") +
  geom_vline(xintercept = 0.5, alpha = 0.8, linetype = "dashed", color = "red")
```

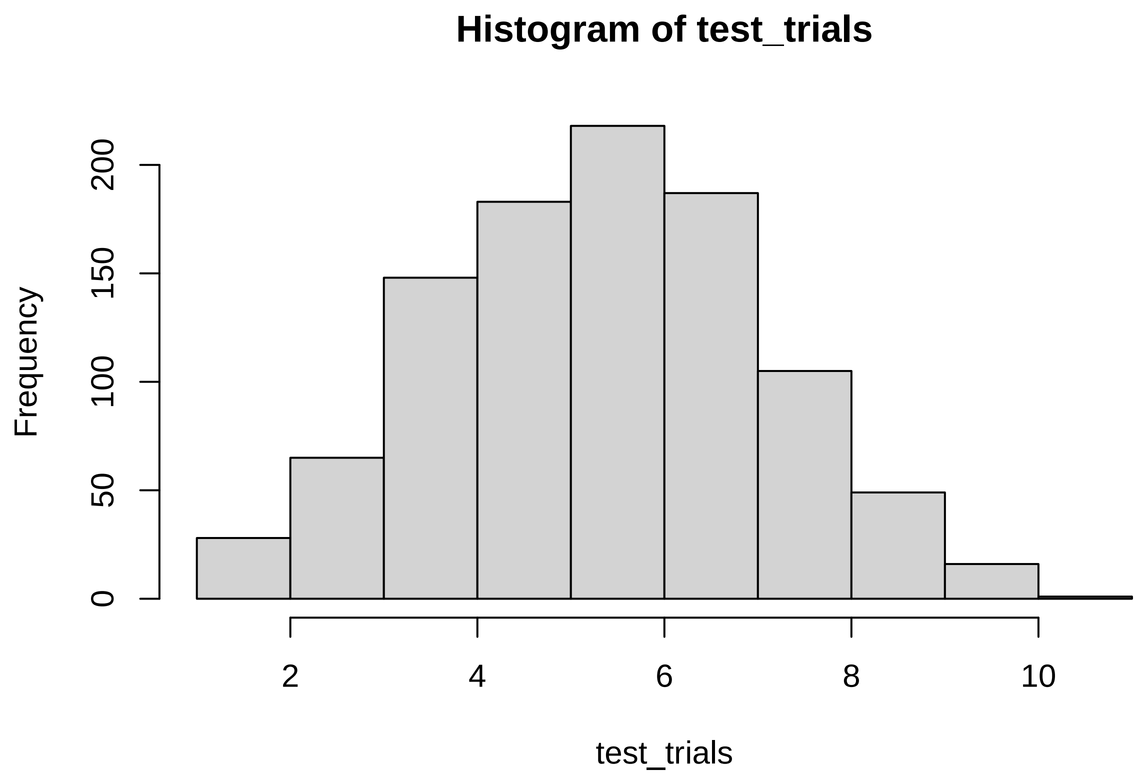
```
## Warning: Removed 2 rows containing missing values (geom_bar).
```



From the density plot we can see that the probability of the woman with the given life conditions, has a higher probability to work rather than not work if we choose our decision boundary to be 0.5

PART C

```
#PART C

post_pred_binom <- function(X, beta) {

  pred <- beta %*% X_data #linear predictions

  pred_logit <- 1/(1 + exp(-pred)) #sigmoid function to model probabilities

  trials <- c()

  for (i in 1:length(pred_logit)) {
    trials[i] <- rbinom(n = 1, size = 11, prob = pred_logit[i])
  }

  return(trials)
}

test_trials <- post_pred_binom(X_data, beta_sim)
hist(test_trials)
```



So, based on the histogram we can say that based on the posterior prediction probabilities 4 to 6 women out of 11 are likely to be working with the given features.

Appendix - CODE

```
knitr::opts_chunk$set(echo = TRUE, cache = TRUE)
library(mvtnorm)
library(geoR)
library(ggplot2)
library(caret)
# Question 1

# PART A

#read the temperature data
temp_lamb <- read.table("TempLambohov.txt", header = TRUE)
plot(temp_lamb$time, temp_lamb$temp)

#set the starting parameters

mu_0 <- c(-10,100,-100)
omega_0 <- 0.02 * diag(3)
nu_0 <- 3
sigma2_0 <- 2

Y <- temp_lamb$temp
X <- cbind(1, temp_lamb$time, temp_lamb$time^2)

#prior draw

prior_fn <- function(covariates = X,
                     mu = mu_0,
                     omega = omega_0,
                     sigma2 = sigma2_0,
                     nu = nu_0) {

  #Inverse chi sq draw for variance
  prior_sigma2 <- rchisq(1, df = nu) #draw from chi-sq with nu df
  prior_sigma2 <- nu * sigma2/prior_sigma2 #compute for inv-chi-sq draws

  #prior_sigma2 <- rinvchisq(n = 1, df = nu, scale = sigma2)

  prior_beta <-rmvnorm(1, mean = mu, sigma = prior_sigma2 * solve(omega))

  prior_final <- covariates %*% t(prior_beta)

  return(prior_final)

}



plot(temp_lamb$time, temp_lamb$temp)
for (i in 1:100) {
  lines(x = temp_lamb$time, y = prior_fn(), col = 'red')
}
```

```
plot(temp_lamb$time, temp_lamb$temp)
for (i in 1:100) {
  lines(x = temp_lamb$time,
        y = prior_fn(mu= c(-20,145,-135),
                     omega = 0.07 * diag(3),
                     nu = 5,
                     sigma2 = 0.15),
        col = 'red')
}

#setting new initial params

mu_set= c(-20,145,-135)
omega_set = 0.07 * diag(3)
nu_set = 5
sigma2_set = 0.15

#PART B.1.

#Joint Posterior Distribution

joint_posterior <- function(covariates = X,
                            responses = Y,
                            omega = omega_set,
                            mu = mu_set,
                            nu = nu_set,
                            sigma2 = sigma2_set) {

  #Setup new params for Posterior

  beta_hat <- solve(t(covariates) %*% covariates) %*% t(covariates) %*% responses
  mu_n <- (solve(t(covariates) %*% covariates + omega)) %*% (t(covariates) %*% covariates %*% beta_hat
  omega_n <- t(covariates) %*% covariates + omega
  nu_n <- nu + length(responses)
  sigma2_n <- 1/nu_n * (nu * sigma2 + t(responses) %*% responses + t(mu) %*% omega %*% mu - t(mu_n) %*%


  #marginal posteriors of betas and variance

  #sigma_posterior <- rinvchisq(n = 10000, df = nu_n, scale = as.numeric(sigma2_n))
  sigma_posterior <- rchisq(10000, df = nu_n) #draw from chi-sq with nu_n df
  sigma_posterior <- nu_n * as.numeric(sigma2_n)/sigma_posterior #compute for inv-chi-sq draws

  beta_posterior <- rmvnorm(n = 10000, mean = mu_n, sigma = as.numeric(sigma2_n) * solve(omega_n))

  return(cbind.data.frame(sigma_posterior, beta_posterior))

}

posterior <- joint_posterior()

hist(posterior[,1], breaks = 100, probability = TRUE, xlab = "sigma2", main = NULL)
```

```r
hist(posterior[,2], breaks = 100, probability = TRUE, xlab = "beta0", main = NULL)
hist(posterior[,3], breaks = 100, probability = TRUE, xlab = "beta1", main = NULL)
hist(posterior[,4], breaks = 100, probability = TRUE, xlab = "beta2", main = NULL)
#PART B.2.

beta_post <- posterior[,-1]

pred_all <- as.matrix(beta_post) %*% t(X)

pred_median <- apply(pred_all, 2, FUN = median)

temp_lamb_pred <- temp_lamb
temp_lamb_pred$pred_median <- pred_median

pred_pi <- data.frame(lower = double(), upper = double())

for (i in 1:length(pred_median)) {
    pred_pi[i,] <- quantile(pred_all[,i], c(0.025,0.975))
}

temp_lamb_pred <- cbind.data.frame(temp_lamb_pred, pred_pi)

gp <- ggplot(data = temp_lamb_pred, aes(x = time)) +
  geom_point(aes(y = temp), size = 0.5) +
  geom_line(aes(y = pred_median, color = "red")) +
  geom_ribbon(aes(ymin = lower, ymax = upper),
              alpha = 0.05,
              color = "blue",
              fill = "blue",
              linetype = "dashed")

gp
#PART C

max_preds <- apply(pred_all, 2, max)

temp_lamb_pred$max_pred <- max_preds

ggplot(data = temp_lamb_pred, aes(x = time)) +
  geom_point(aes(y = temp), size = 0.5) +
  geom_line(aes(y = pred_median, color = "red")) +
  geom_ribbon(aes(ymin = lower, ymax = upper),
              alpha = 0.05,
              color = "blue",
              fill = "blue",
              linetype = "dashed") +
  geom_line(aes(y = max_pred), color = "green")


# Question 2

rm(list = ls())
```

```r
# PART A

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

#setting up data & parameters for prior

#data
X <- as.matrix(women_work[,2:8])
Y <- as.matrix(women_work[,1], NCOL = 1)

#prior
nfeatures <- dim(X)[2]
tau <- 5
mu <- as.matrix(rep(0,nfeatures)) # Prior mean vector
sigma <- (tau^2)*diag(nfeatures) # Prior covariance matrix

#loglikelihood function

logPosterior <- function(betas, Y, X, mu, sigma){
  LL <- sum((X %*% betas) * Y - log(1 + exp(X %*% betas)))
  LPrior <- dmvnorm(betas, mu, sigma, log=TRUE)
  return(LL + LPrior)
}


#initialize beta for optim
beta_init <- matrix(0, nfeatures, ncol = 1)

#maxinmize logPosterior
OptimRes <- optim(beta_init,
                  logPosterior,
                  gr=NULL,
                  Y,X,mu,sigma,
                  method=c("BFGS"),
                  control=list(fnscale=-1),
                  hessian=TRUE)

#posterior mode
beta_mode <- OptimRes$par
names(beta_mode) <- colnames(X)

#Hessian is -J(beta)
beta_jacobian <- -OptimRes$hessian

#diagonal of hessian is the second derivates corresponding to the covariance
beta_inverse_jacobian <- solve(beta_jacobian)
colnames(beta_inverse_jacobian) <- colnames(X)

cat("\n", "The posterior mode is: ", "\n")
print(beta_mode)

cat("\n", "The Inverse Jacobian is: ", "\n")
print(beta_inverse_jacobian)
```

```r
#draw large sample for beta corresponding to NSmallChild i.e. beta6
beta_sim <- rmvnorm(n = 1000, mean = beta_mode, sigma = solve(beta_jacobian))

#isolate draws for beta6
beta6_sim <- beta_sim[,6]

#95% PI for beta6
beta6_pi <- quantile(beta6_sim, prob = c(0.025,0.975))
names(beta6_pi) <- c("Lower", "Upper")

cat("\n", "The PI for Beta_nSmallChild:", "\n")
print(beta6_pi)
glmModel <- glm(Work ~ 0 + ., data = women_work, family = binomial)

summary(glmModel)
#Part B

#Create matrix of sample features
X_data <- as.matrix(c(1, 20, 12, 8, 43, 0, 2))

#reuse the betas simulated earlier
#use sigmoid function to model probabilities of predictions

post_pred <- function(X, beta) {

  pred <- beta %*% X #linear predictions

  pred_logit <- exp(pred)/(1 + exp(pred)) #sigmoid function to model probabilities

  return(data.frame(Probability = pred_logit))
}

#function call on sample data
post_pred_sim <- post_pred(X_data, beta_sim)

#plotting the posterior prediction density & histogram
ggplot(data = post_pred_sim, aes(x = Probability)) + xlim(c(0,1)) +
  geom_histogram(aes(y = ..density..),
                 bins = 100,
                 color = "black",
                 fill = "grey") +
  geom_density(alpha = 0.1, fill = "green") +
  geom_vline(xintercept = 0.5, alpha = 0.8, linetype = "dashed", color = "red")


#PART C

post_pred_binom <- function(X, beta) {

  pred <- beta %*% X_data #linear predictions

  pred_logit <- 1/(1 + exp(-pred)) #sigmoid function to model probabilities
  trials <- c()

  for (i in 1:length(pred_logit)) {
    trials[i] <- rbinom(n = 1, size = 11, prob = pred_logit[i])
  }

  return(trials)
}

test_trials <- post_pred_binom(X_data, beta_sim)
hist(test_trials)
```