Lab2

Andreas C Charitos-andch552 Jiawei Wu-jiawu449

22 April 2019

Exercise 1-Linear and polynomial regression

Question 1.a

The response variable is temp and the covariate is :

$$time = \frac{(the\ number\ of\ days\ since\ beginning\ of\ year)}{366}$$

The task is to perform a Bayesian analysis of a quadratic regression

$$temp = \beta_0 + \beta_1 * time + \beta_2 * time^2 + \epsilon,\ \epsilon \sim N(0, \sigma^2)$$

Use the conjugate prior for the linear regression model. Your task is to set the prior hyperparameters μ_0,Ω_0,v_0 and σ_0^2 to sensible values. Start with $\mu_0 = (--10, 100, -100)^T$,$\Omega_0 = 0,01 * I_3$, $v_0 = 4$ and $\sigma_o^2 = 1$.Check if this prior agrees with your prior opinions by simulating draws from the joint prior of all parameters and forevery draw compute the regression curve. This gives a collection of regressioncurves,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 do agree with your prior beliefs about the regression curve.

Joint prior for β ,σ^2

The joint prior is given by the formulas above

$$\beta/\sigma^2 \sim N(\mu_0, \sigma^2\Omega_0^-1)$$
$$\sigma^2 \sim Inv - \chi^2(v_0, \sigma_0^2)$$

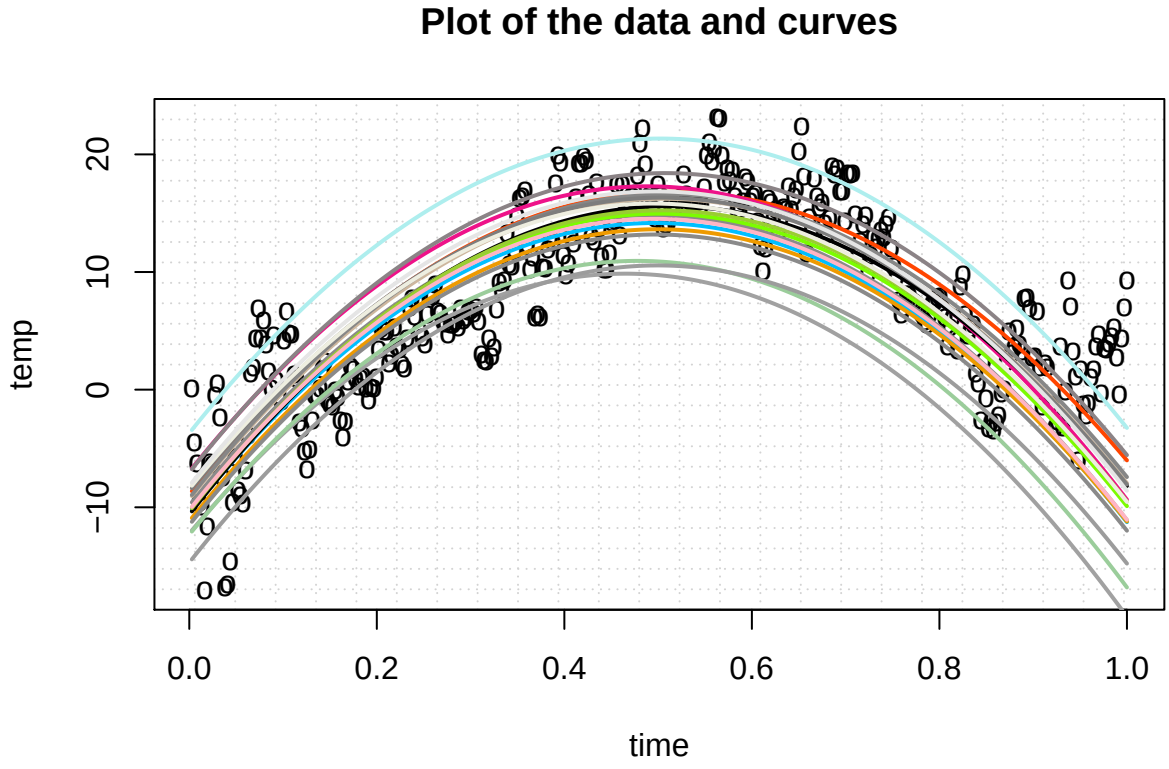
```
# import dataset
templink<-read.delim("TempLinkoping.txt")
# import library
library(mvtnorm)
```

```
## Warning: package 'mvtnorm' was built under R version 3.5.2
```

```
# setting the priors mu,sigma,omega,v
prior_m0=c(-10,100,-100)
prior_v=4
prior_sigma_sq0=1
prior_omega=0.5*diag(3)
# inverse chi-sqare function
NormalNonInfoPrior<-function(v0,sigma_sq0){
  # returns one sample from Inv-chi square for given (df,sigma)
  PostDraws<-(v0*sigma_sq0)/rchisq(1,v0)
  return(PostDraws)
}
```

```r
# simulation function for hte conjugate normal prior
conjugate_prior_sim<-function(N,v0,sigma_sq0,mu0,omega0){
  # given number of sample,v0,sigma0,mu0,omega0 returns sample
  # for prior predictions and prior betas
  temp_pred<-matrix(0,N,dim(templink)[1]) # matrix to store the preds
  betas<-matrix(0,N,length(mu0)) # matrix to store the betas
  for(i in 1:N){
    sigmaSq_prior<-NormalNonInfoPrior(v0,sigma_sq0) # take sigma from Inv-chi square
    betas[i,]<-rmvnorm(1,mean=mu0,sigma=sigmaSq_prior*solve(omega0)) # draw sample for betas
    temp_pred[i,]<-betas[i,1]+betas[i,2]*templink[,1]+
      betas[i,3]*templink[,1]^2+rnorm(1,0,sigmaSq_prior)
  }
  return(list("temp_pred"=temp_pred,"beta_preds"=betas))
}
```

```r
# sample of betas prior
prior_sample<-conjugate_prior_sim(20,prior_v,prior_sigma_sq0,prior_m0,prior_omega)
# plot the respose and covariate plus curves from fitted regression from beta prior sample
plot(templink[,1],templink[,2],panel.first=grid(25,25),ylab = "temp",xlab="time",
     main="Plot of the data and curves",pch="o")
for (j in 1:dim(prior_sample$temp_pred)[1]) lines(templink[,1], prior_sample$temp_pred[j,],
                                                  col=sample(colors(), 1),lwd=2)
```



Question 1.b

Write a program that simulates from the joint posterior distribution of $\beta_0, \beta_1, \beta_2, \sigma^2$,Plot the marginal posteriors for 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 $f(time) = \beta_0 + \beta_1 * time + \beta_2 * time^2$, computed for every value of time. Also overlay curvesfor 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?

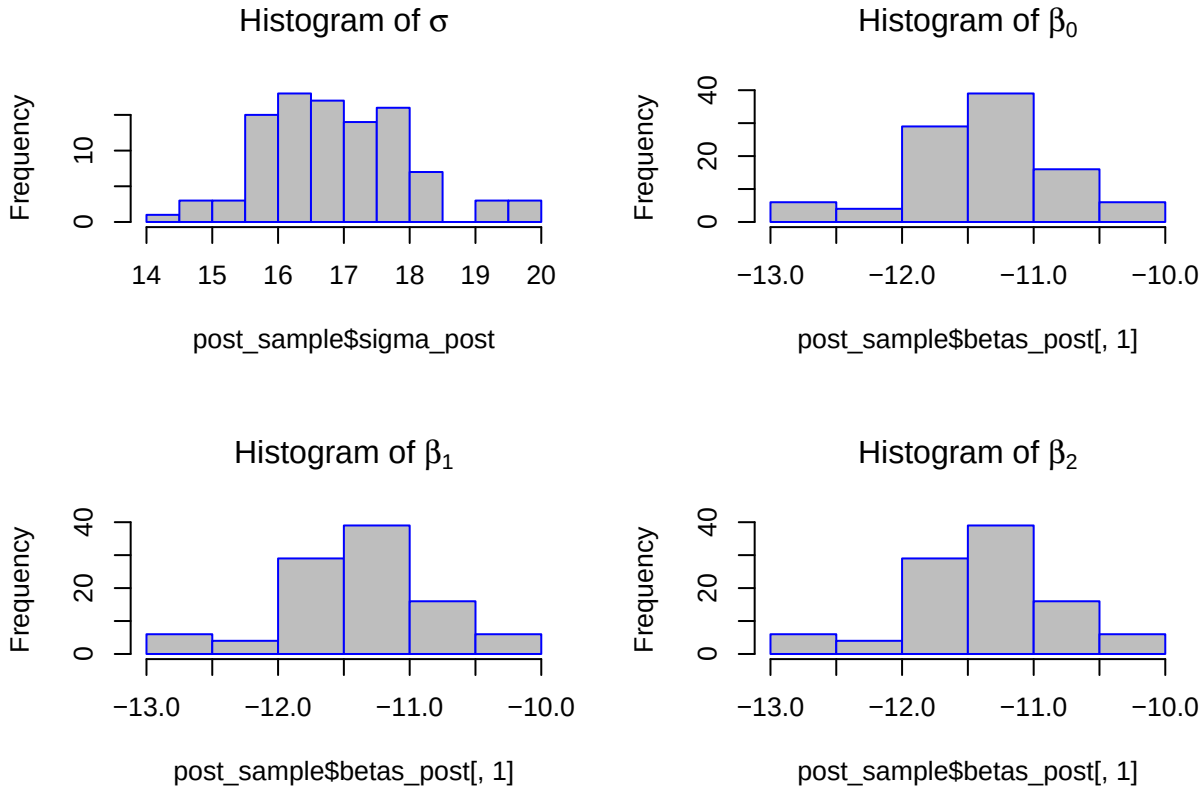
The posterior prior is given by

$$\beta/\sigma^2, y \sim N(\mu_n, \sigma^2\Omega_n^-1)$$
$$\sigma^2/y \sim Inv - \chi^2(v_0, \sigma_n^2)$$
$$\mu_n = (X'X + \Omega_0^-1)(X'X\hat{\beta} + \Omega_0\mu_0)$$
$$\Omega_n = X'X + \Omega_0$$
$$v_n = v_0 + n$$
$$v_n\sigma_n^2 = v_0\sigma_0^2 + (y'y + \mu_0'\Omega_0\mu_0 - \mu_n'\Omega_n\mu_n)$$

```
posterior_sim<-function(nsample,X,y,v0,mu0,sigma_sq0,omega0){
  # function that given nsample,X,y,v0,mu0,sigma_sq0,omega0
  # returns list with sigma posteriors and betas
  betas_post=matrix(0,nsample,length(mu0)) # initiate betas matrix
  sigma_post=rep(0,nsample) # initiate sigmas matrix
  beta_hat<-solve(t(X)%*%X)%*%t(X)%*%y # we need this to calculations
  for (i in 1:nsample){
    # calculate mu_n
    mu_n<-solve( t(X)%*%X+omega0 )%*%(t(X)%*%X%*%beta_hat+omega0%*%mu0 )
    # calculate omega_n
    omega_n<-t(X)%*%X+omega0
    # calculate v_n
    v_n<-v0+dim(templink)[1]
    # calculate sigma_n
    sigma_n<-( v0*sigma_sq0+(t(y)%*%y+t(mu0)%*%omega0%*%mu0-t(mu_n)%*%omega_n%*%mu_n) )/v_n
    # posterior sigma from Inv-cho square
    sim_sigma<-NormalNonInfoPrior(v_n,sigma_n)
    # store the posterior sigma
    sigma_post[i]<-sim_sigma
    # posterior beta
    sim_betas<-rmvnorm(1,mean=mu_n,sigma=sim_sigma[1]*solve(omega_n))
    # store posterior beta
    betas_post[i,]<-sim_betas
  }

  return(list("sigma_post"=sigma_post,"betas_post"=betas_post))

}
```

```
# the matrix X has 3 columns (1,time,time^2)
Xt<-cbind(1,templink[,1],templink[,1]^2)
# the response
yt=templink[,2]
# take a sample from posterior function
post_sample<-posterior_sim(100,Xt,yt,prior_v,prior_m0,prior_sigma_sq0,prior_omega)
```

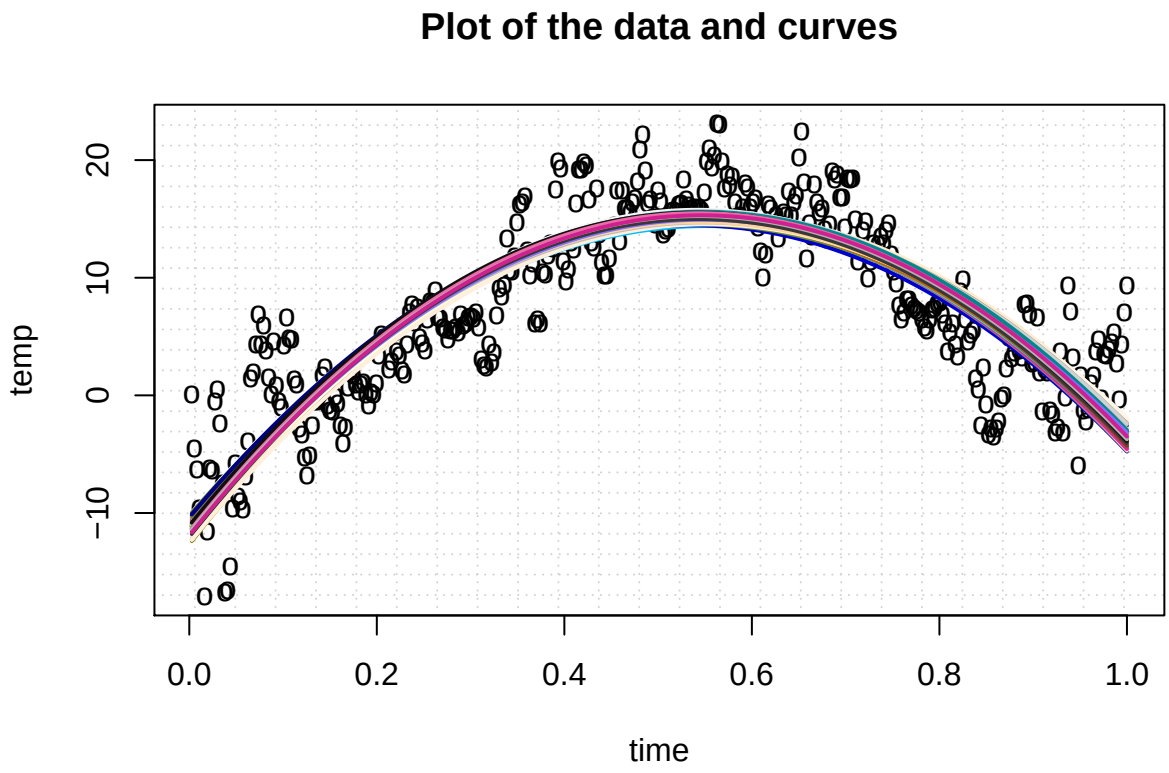
```r
# take only the betas from the list
betas_pr<-post_sample$betas_post

par(mfrow=c(2,2))
hist(post_sample$sigma_post,main=expression(paste("Histogram of ",sigma)),
     col="gray",border="blue")
hist(post_sample$betas_post[,1],main=expression(paste("Histogram of ",beta[0] )),
     col="gray",border = "blue")
hist(post_sample$betas_post[,1],main=expression(paste("Histogram of ",beta[1] )),
     col="gray",border = "blue")
hist(post_sample$betas_post[,1],main=expression(paste("Histogram of ",beta[2] )),
     col="gray",border = "blue")
```



Scaterplot with overlay curves from posterior regresion function

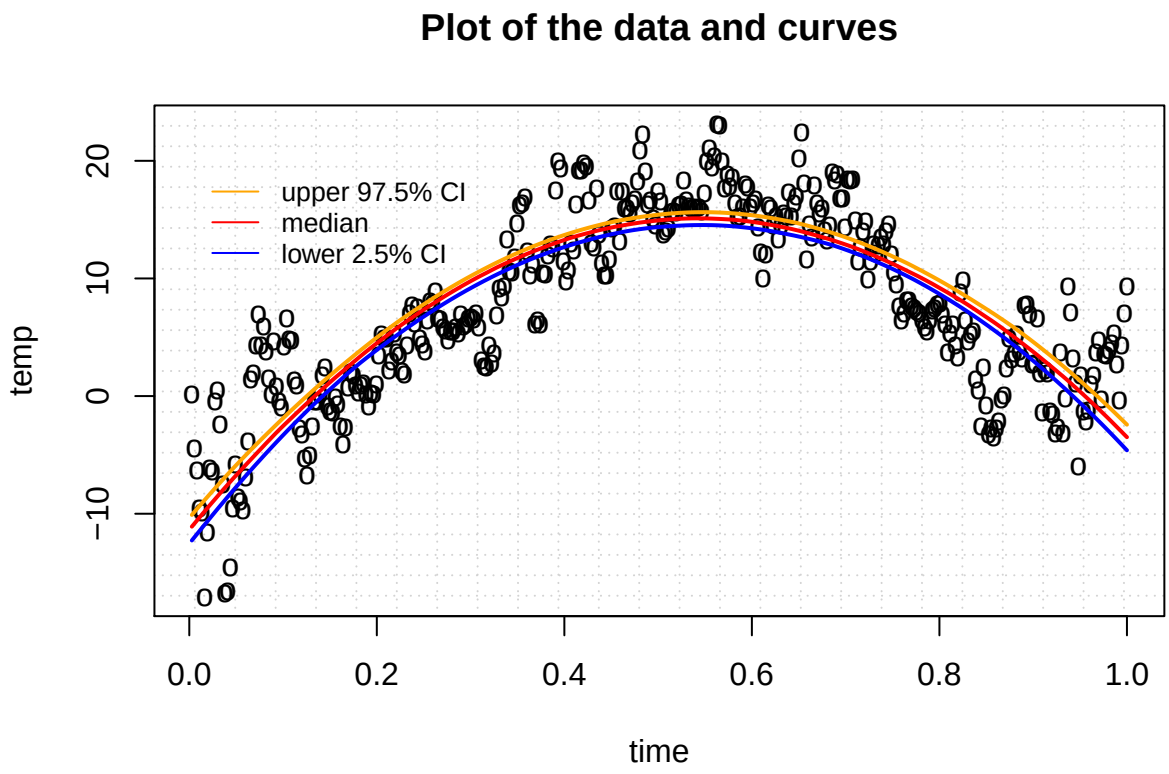
```r
# matrix to store the result for every regression line
# rows is the nsample and columns are the number of observations
Dt<-matrix(0,dim(betas_pr)[1],length(templink[,1]) )
# we apply the regression function f(time) for every row of betas
for (i in 1:dim(betas_pr)[1]){

  Dt[i,]<-betas_pr[i,1]+betas_pr[i,2]*templink[,1]+betas_pr[i,3]*(templink[,1]^2)

}
```

```r
plot(templink[,1],templink[,2],panel.first=grid(25,25),ylab = "temp",xlab="time",
     main="Plot of the data and curves",pch="o")
for (j in 1:dim(Dt)[1]) lines(templink[,1], Dt[j,],col=sample(colors(), 1),lwd=2)
```



Scatterplot of data with posterior credible interval and median

```r
# calculate CI for every point of time
posterior_CI<-apply(Dt,2,function(y) quantile(y, probs = c(0.025, 0.975)))
posterior_median<-apply(Dt,2,function(x) median(x))

plot(templink[,1],templink[,2],panel.first=grid(25,25),ylab = "temp",xlab="time",
     main="Plot of the data and curves",pch="o")
lines(templink[,1],posterior_CI[1,],col="blue",lwd=2)
lines(templink[,1],posterior_CI[2,],col="orange",lwd=2)
lines(templink[,1],posterior_median,col="red",lwd=2)
legend(0,20, legend=c("upper 97.5% CI","median","lower 2.5% CI"),
       col=c("orange", "red","blue"), lty=1, cex=0.8,
       box.lty=0)
```



Question 1.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 $\tilde{x}$. Use the simulations in b) to simulate from the posterior distribution of $\tilde{x}$.

```
#calcutate derivative of time function and set to 0
timehat=betas_pr[,2]/(-2*betas_pr[,3])
max(timehat)
```

```
## [1] 0.5553199
```

Question 1.d

Exercise 2-Posterior approximation for classification with logistic regression

Question 2.a

Consider the logistic regression

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

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). Fit the logistic regression using maximum likelihood estimation by the command: glmModel <- glm(Work ~ 0 + ., data = WomenWork, family = binomial).

```
df<-read.table("WomenWork.dat",header=T)
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loading required package: foreach
```

```
## Loaded glmnet 2.0-16
```

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

```
library(knitr)
out_func<-function(){

  foo1 <- function () {
     mytable<-data.frame(glmModel$coefficients)
     kable(mytable)
    }
  cat("The models coefficients are : \n")
  print(foo1())
  cat("\n")
  cat("================================================\n")
  cat("\n")
  cat("The models missclassification error is : ",
      mean(ifelse(glmModel$fitted.values>0.5,1,0)==df$Work))
  cat("\n")
  cat("\n")
  cat("================##  Model Summary  ##================")
  cat("\n")
  summary(glmModel)
}
out_func()
```

The models coefficients are :

	glmModel.coefficients
Constant	0.6443036
HusbandInc	-0.0197746
EducYears	0.1798806
ExpYears	0.1675127
ExpYears2	-0.1443595
Age	-0.0823403
NSmallChild	-1.3625024
NBigChild	-0.0254299

==========

The models missclassification error is : 0.71

==========## Model Summary ##==========

Call: glm(formula = Work ~ 0 + ., family = binomial, data = df)

Deviance Residuals: Min 1Q Median 3Q Max
-2.1662 -0.9299 0.4391 0.9494 2.0582

Coefficients: Estimate Std. Error z value Pr(>|z|)

Constant 0.64430 1.52307 0.423 0.672274
HusbandInc -0.01977 0.01590 -1.243 0.213752
EducYears 0.17988 0.07914 2.273 0.023024 *
ExpYears 0.16751 0.06600 2.538 0.011144 *
ExpYears2 -0.14436 0.23585 -0.612 0.540489
Age -0.08234 0.02699 -3.050 0.002285 ** NSmallChild -1.36250 0.38996 -3.494 0.000476 *** NBigChild -0.02543 0.14172 -0.179 0.857592
— Signif. codes: 0 ‘’ ***0.001*** ’’ *0.01* ’’ 0.05 ‘.’ 0.1 ‘’ 1

(Dispersion parameter for binomial family taken to be 1)

`Null deviance: 277.26  on 200  degrees of freedom`

Residual deviance: 222.73 on 192 degrees of freedom AIC: 238.73

Number of Fisher Scoring iterations: 4

Question 2.b

```
# b
chooseCov <- c(2:9)
tau <- 10; # Prior scaling factor such that Prior Covariance = (tau^2)*I

# Data from the read.table function is a data frame. Let's convert y and X to vector and matrix.
y <- as.vector(df[,which(names(df)=="Work")]);
X <- as.matrix(df[,-which(names(df)=="Work")]);
covNames <- names(df)[-1];
#X <- X[,chooseCov]; # Here we pick out the chosen covariates.
nPara <- dim(X)[2];

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

# Defining the functions that returns the log posterior (Logistic and Probit models).
# Note that the first input argument of

# this function must be the one that we optimize on, i.e. the regression coefficients.

LogPostLogistic <- function(betaVect,y,X,mu,Sigma){

  nPara <- length(betaVect);
  linPred <- X%*%betaVect;

  # evaluating the log-likelihood
  logLik <- sum( linPred*y -log(1 + exp(linPred)));
  if (abs(logLik) == Inf) logLik = -20000; # Likelihood is not finite,
                                           # stear the optimizer away from here!

  # evaluating the prior
  logPrior <- dmvnorm(betaVect, matrix(0,nPara,1), Sigma, log=TRUE);

  # add the log prior and log-likelihood together to get log posterior
  return(logLik + logPrior)
}
```

```r
# Different starting values. Ideally, any random starting value gives you
# the same optimum (i.e. optimum is unique)
initVal <- as.vector(rep(0,dim(X)[2]));
# Or a random starting vector: as.vector(rnorm(dim(X)[2]))
# Set as OLS estimate: as.vector(solve(crossprod(X,X))%*%t(X)%*%y); # Initial values by OLS

logPost = LogPostLogistic;

OptimResults<-optim(initVal,logPost,gr=NULL,y,X,mu,Sigma,method=c("BFGS"),
                    control=list(fnscale=-1),hessian=TRUE)

# Printing the results to the screen
postMode <- OptimResults$par
postCov <- -solve(OptimResults$hessian) # Posterior covariance matrix is -inv(Hessian)
names(postMode) <- covNames # Naming the coefficient by covariates
approxPostStd <- sqrt(diag(postCov)) # Computing approximate standard deviations.
names(approxPostStd) <- covNames # Naming the coefficient by covariates
print('The posterior mode is:')
```

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

```r
print(postMode)
```

```
##    Constant  HusbandInc   EducYears    ExpYears   ExpYears2         Age
##  0.62672884 -0.01979113  0.18021897  0.16756670 -0.14459669 -0.08206561
## NSmallChild   NBigChild
## -1.35913317 -0.02468351
```

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

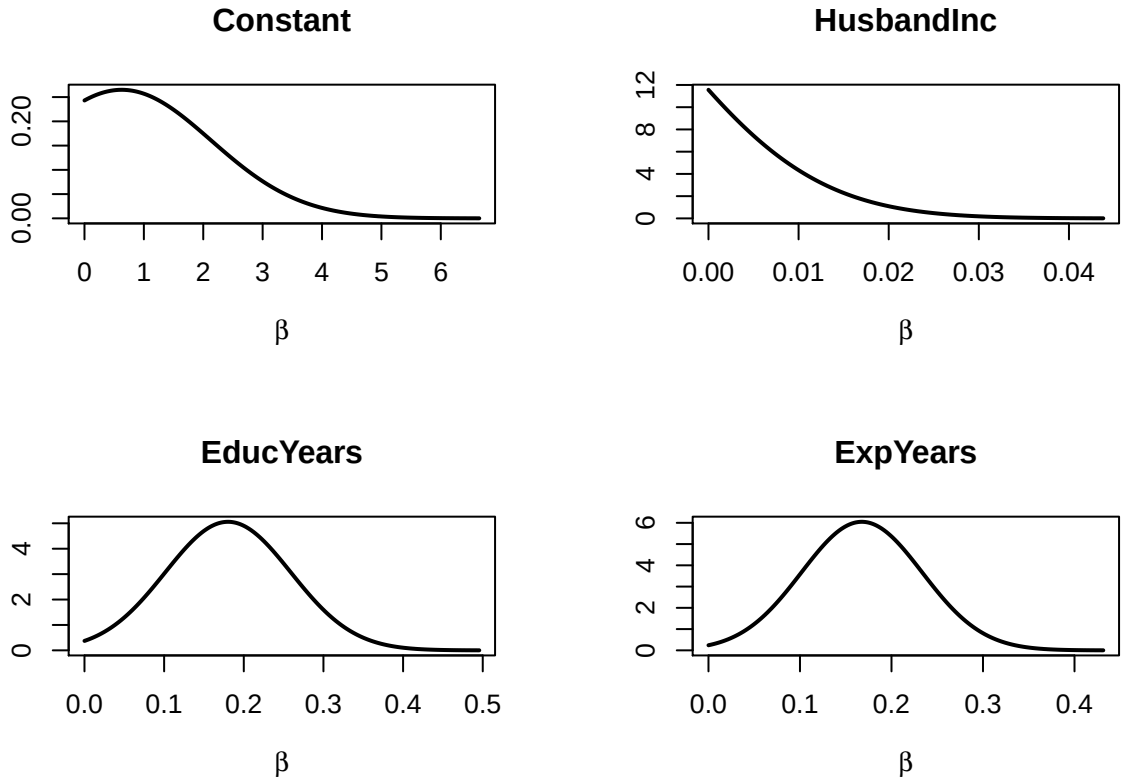
```
## [1] "The approximate posterior standard deviation is:"
```

```r
print(approxPostStd)
```

```
##    Constant  HusbandInc   EducYears    ExpYears   ExpYears2         Age
##  1.50533138  0.01589983  0.07885556  0.06596754  0.23575129  0.02680412
## NSmallChild   NBigChild
##  0.38892439  0.14132327
```

Plot some of marginal posteriors

```r
# Plotting some of the marginal posteriors
par(mfrow = c(2,2))
for (k in 1:4){
  betaGrid <- seq(0, postMode[k] + 4*approxPostStd[k], length = 1000)
  plot(betaGrid, dnorm(x = betaGrid, mean = postMode[k], sd = approxPostStd[k]),
       type = "l", lwd = 2, main = names(postMode)[k],
       ylab = '', xlab = expression(beta))
}
```




```
credInter=quantile(df$NSmallChild, c(0.025, 0.975))
cat("The credible interval for NSmallChild is :\n")
```

```
## The credible interval for NSmallChild is :
```

```
credInter
```

```
##  2.5% 97.5%
##     0     2
```

Question 2.c

Write a function that simulates from the predictive distribution of the response variable in a logistic regression. Use your normal approximation from 2(b). 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.

```
pred_dist_logreg<-function(nSample,mode,Cov,query){
  # given number of sample and posterior beta ,Hessian and query
  # returns the predictions using the logistic regression
  sample_postMode<-rmvnorm(nSample,mean=mode,sigma=Cov)

  y_preds<-query%*%t(sample_postMode)


  y_logpreds<-ifelse(exp(y_preds)/(1+exp(y_preds))>runif(nSample,0,1),1,0)

  return(y_logpreds)
```

```
}
```

```
# the query we wish to predict distribution
query<-c(1,10,8,10,1,40,1,1)
# simulate sample of log predictions
pr<-pred_dist_logreg(10000,postMode,postCov,query)
# plot the density and histogram of the distribution of the query
par(mfrow=c(1,2))
plot(density(pr),main="Density of the predictive \ndistirbution for the query",
     panel.first=grid(25,25),col="mediumvioletred",lwd=2)
polygon(density(pr), density = 18, angle = 45,col="lightslateblue")

hist(pr,main="Histogram of the predictive \ndistirbution for the query",
     panel.first=grid(25,25),border="dodgerblue",col="gray",lwd=2)
```