STA360 Homework 8 (Ken Ye)

```r
library(latex2exp)
library(ggplot2)
library(MASS)
library(ggrepel)
library(mvtnorm)
set.seed(0)
```

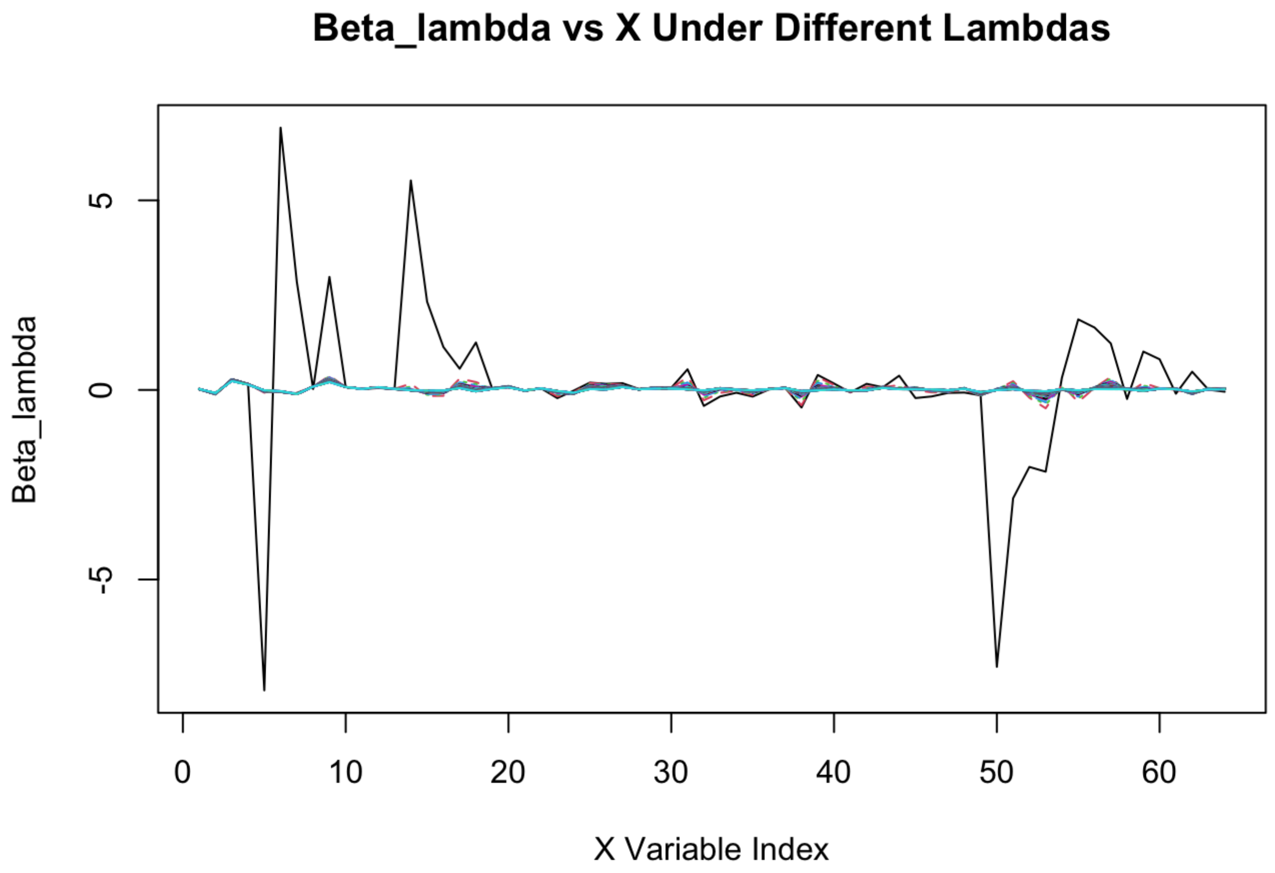
Question 3

```r
# load diabetes data
yX <- dget(url("https://www2.stat.duke.edu/~pdh10/FCBS/Inline/yX.diabetes.train"))
y <- yX[,1]
X <- yX[,-1]
```

Part a

```r
# compute beta_lambda hat for each lambda in {0, ..., 100}
lambdas <- seq(0, 100, by = 1)
beta_lambdas <- matrix(0, nrow = 64, ncol = 101)
for (lam in lambdas){
  beta_lambda <- solve(t(X) %*% X + lam * diag(rep(1, 64))) %*% t(X) %*% y
  beta_lambdas[,lam+1] <- beta_lambda
}
```

```r
# plot with matplot
matplot(beta_lambdas,
        type = 'l',
        main = 'Beta_lambda vs X Under Different Lambdas',
        ylab = 'Beta_lambda',
        xlab = 'X Variable Index')
```



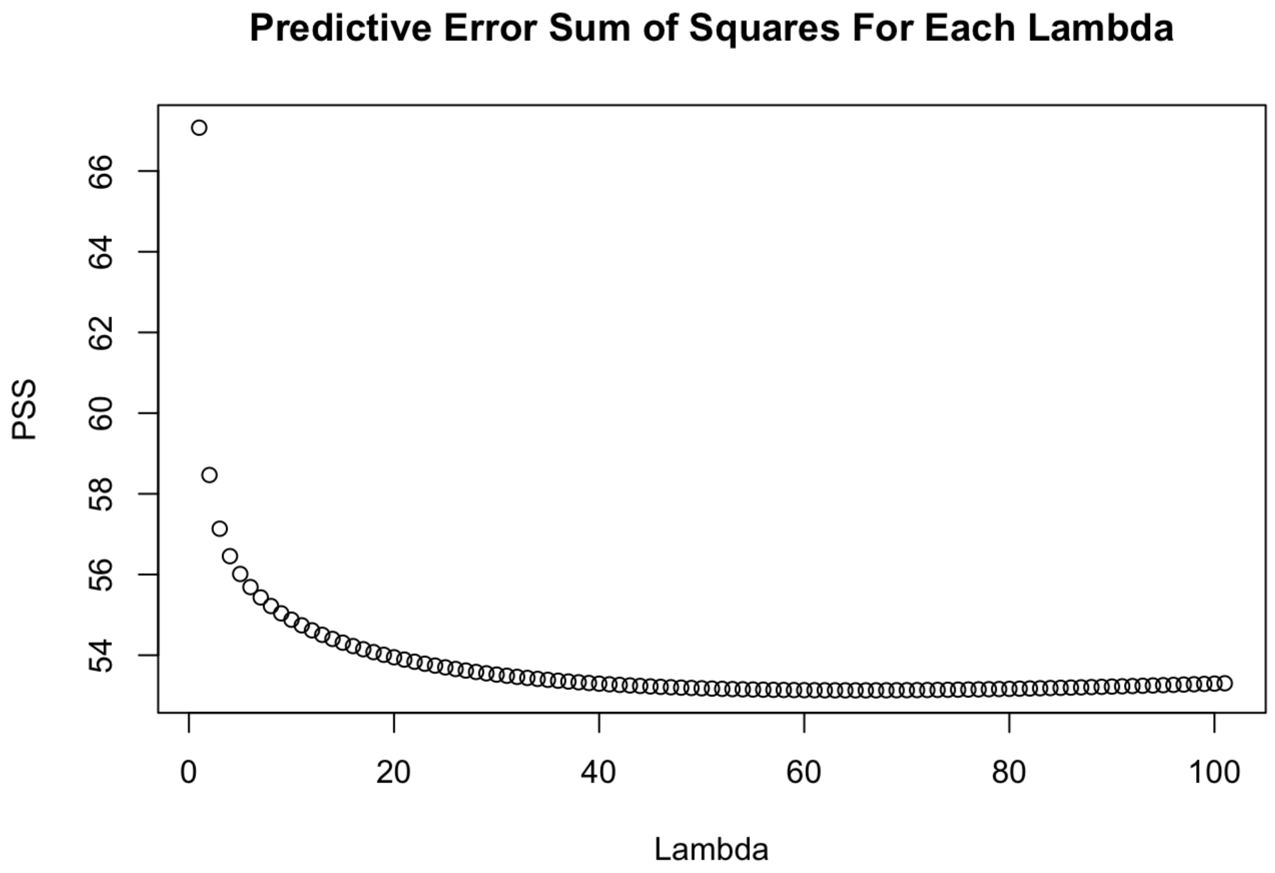
There are 101 lines representing 101 beta_lambda vectors (each 64 by 1) and their respective value for each X variable. It's hard to discern a single beta_lambda vector in the graph because there are so many of them, but this graph shows the beta_lambda estimate for each lambda in {0, 1, … , 99, 100}, and for each beta_lambda, how its beta_lambda_i varies for each X_i, for i from 1 to 64 (there are 64 x variables in total).

Part b

```
# load diabetes data, test set
yX.test <- dget(url("https://www2.stat.duke.edu/~pdh10/FCBS/Inline/yX.diabetes.test"))
y.test <- yX.test[,1]
X.test <- yX.test[,-1]
```

```
# calculate predictive error sum of squares
PSS <- rep(0,101)
for (i in 1:101){
  PSS[i] <- sum((y.test - (X.test %*% beta_lambdas[,i]))^2)
}
```

```
# plot
plot(PSS,
     main = 'Predictive Error Sum of Squares For Each Lambda',
     ylab = 'PSS',
     xlab = 'Lambda')
```



```
# OLS estimate predictive error sum of squares
PSS[1]
```

```
## [1] 67.07489
```

We know that beta_OLS = beta_lambda when lambda = 0. The unbiased OLS estimate for prediction has a predictive error sum of squares of 67.07, which is the highest among all beta estimates (which can be told from the graph). Beta_OLS doesn’t perform well.

Part c

```
# identify the value of lambda that has the best predictive performance
index <- which.min(PSS)
# this is the index, the value of lambda =  index - 1 since lambda starts from 0
index
```

```
## [1] 65
```

The value of lambda that has the best predictive performance is lambda = 64.

```
# find x-variables that have the largest effects
beta_lambda.best <- array(beta_lambdas[,65])
names(beta_lambda.best) <- colnames(X.test)
sort(beta_lambda.best, decreasing = TRUE)
```

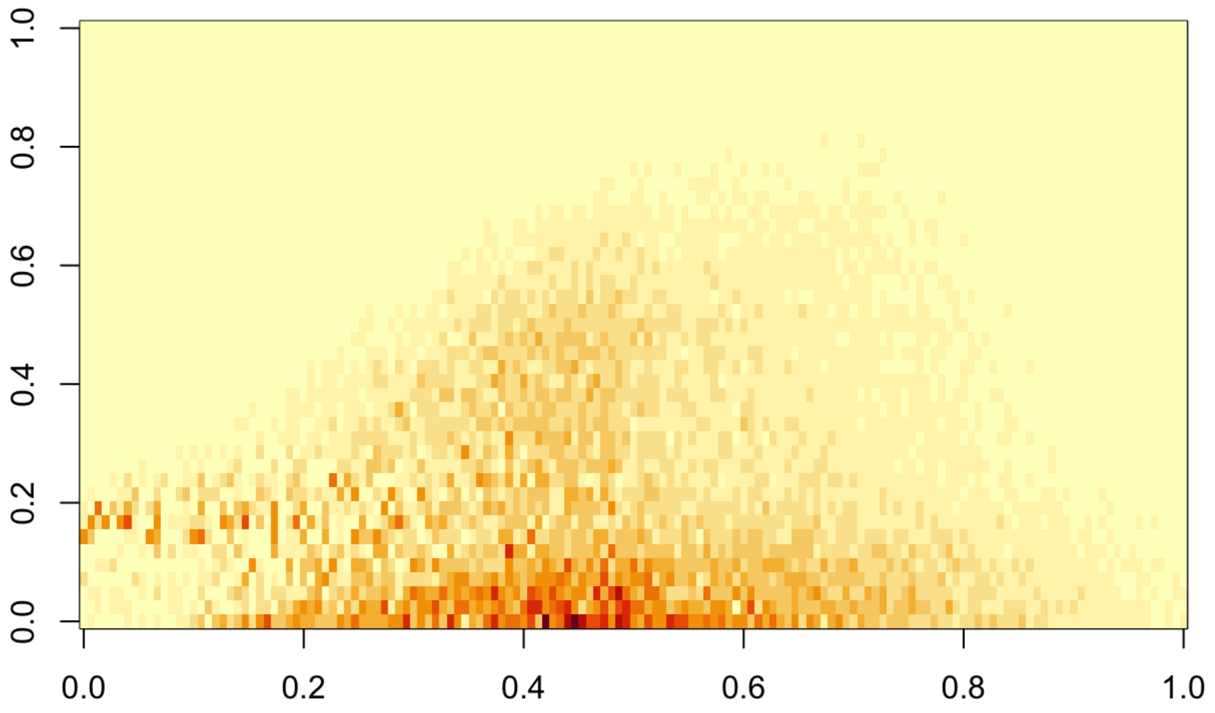
```
##          bmi          ltg          map      age:ltg          tch          glu 
##  0.252909731  0.228397916  0.147254764  0.087941087  0.087934685  0.077338474 
##      age:sex        bmi^2        tch^2      bmi:glu      bmi:map      sex:hdl 
##  0.076324787  0.071152071  0.058732597  0.058276976  0.049170486  0.043500542 
##      sex:bmi      sex:map      age:map      ldl:ltg      map:ltg      hdl:ltg 
##  0.043490113  0.041124402  0.040800031  0.038342222  0.035184329  0.034370233 
##      age:hdl        glu^2        age^2      ldl:tch        map^2       tc:hdl 
##  0.033915390  0.031990446  0.028071754  0.026620010  0.024993749  0.024568090 
##      map:ldl      sex:glu      ldl:glu          age       map:tc      hdl:glu 
##  0.023869197  0.022266772  0.022153006  0.021963242  0.021280595  0.019970286 
##      bmi:hdl       sex:tc      ltg:glu      age:tch       tc:glu      age:glu 
##  0.018778877  0.018338931  0.017996024  0.017610083  0.017397253  0.016284681 
##      tch:glu      map:hdl      sex:tch       tc:ldl      bmi:ldl         tc^2 
##  0.014990058  0.009804745  0.009214922 -0.001376226 -0.003008950 -0.007634208 
##      map:tch      bmi:ltg      bmi:tch      sex:ltg      age:bmi      ldl:hdl 
## -0.011186191 -0.011266451 -0.014018574 -0.014901732 -0.015406394 -0.016465109 
##           tc      sex:ldl        hdl^2        ldl^2      hdl:tch       tc:tch 
## -0.017435685 -0.017546233 -0.018174450 -0.024323567 -0.025176356 -0.025552851 
##       age:tc        ltg^2       bmi:tc          ldl       tc:ltg      tch:ltg 
## -0.031782045 -0.034849965 -0.035749440 -0.040005158 -0.047686902 -0.052317906 
##      map:glu      age:ldl          sex          hdl 
## -0.073321473 -0.080450910 -0.091657677 -0.109873439
```

The x-variables that have the largest effects (top 5) are bmi, ltg, map, age:ltg, and tch. The rest are printed above in decreasing effect order.

Question 4

```
# load water data
yX <- readRDS("yXSS.rds")
y <- yX[,1]
X <- yX[,-1]
```

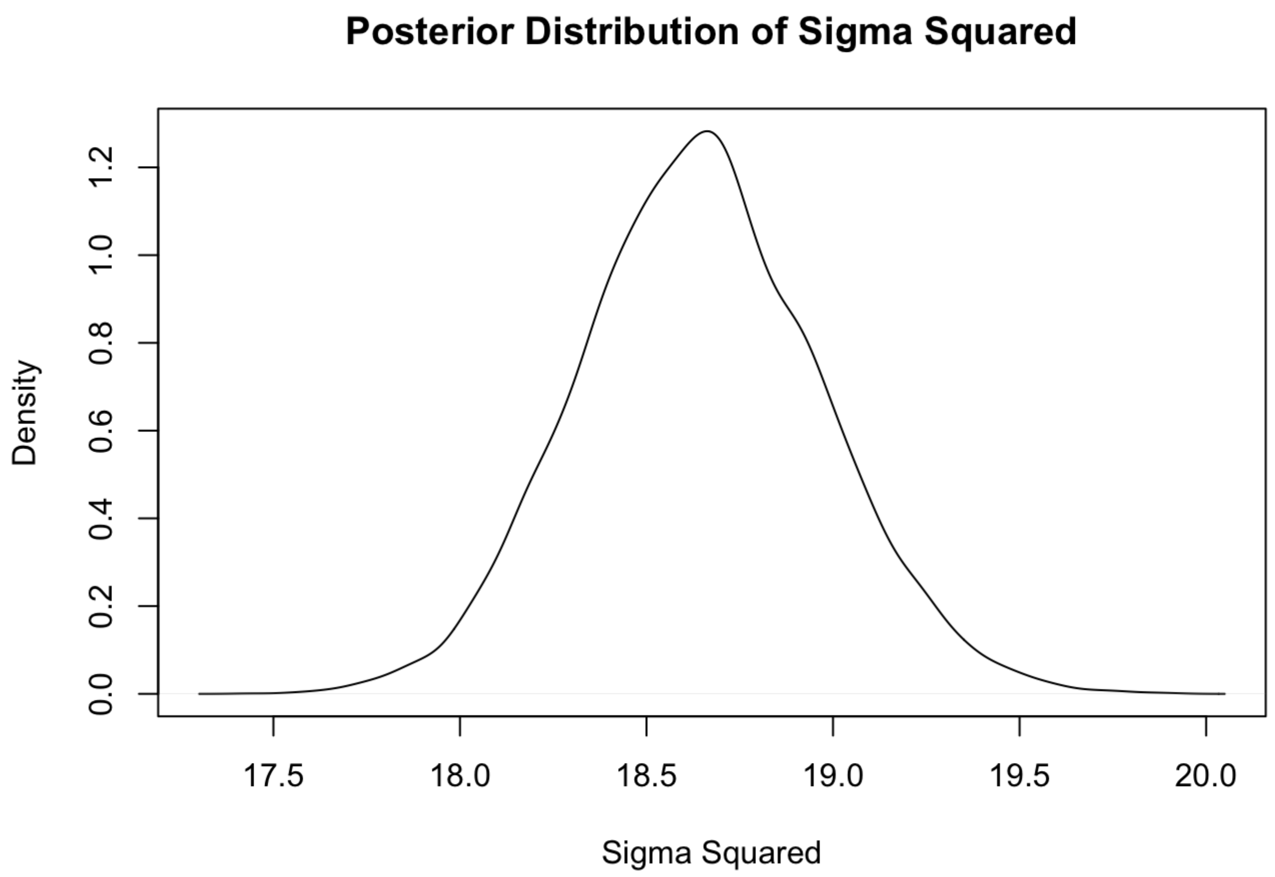
```
# view image
y <- yX[,1]
image(matrix(y,151,43))
```



Part a

```
# obtain posterior distribution of beta and sigma2 given y with MCMC

n <- dim(X)[1]
p <- dim(X)[2]

# priors
nu.0 <- 1
beta.0 <- rep(1/9, p)
sigma.0 <- matrix(0, p, p)
diag(sigma.0) <- 1 # sigma.0 = I_9
sigma2.0 <- diag(p)
sample.size <- 10000
BETA <- matrix(nrow = sample.size, ncol = p) # posterior storage
SIGMA2 <- rep(NA, sample.size) # posterior storage

# common quantities
X <- as.matrix(X)
y <- as.matrix(y)
isigma.0 <- solve(sigma.0)
XtX <- t(X) %*% X
Xty <- t(X) %*% y

# start values
sigma2 <- var(residuals(lm(y ~ 0 + X)))

# Gibbs sampling
for (s in 1 : sample.size) {
  # update beta
  beta.V <- solve(isigma.0 + XtX / sigma2)
  beta.E <- beta.V %*% (isigma.0 %*% beta.0 + Xty / sigma2)
  beta <- mvrnorm(1, beta.E, beta.V)

  # update sigma2
  nu.n <- nu.0 + n
  ss.n <- nu.0 * sigma2.0 + sum((y - X %*% beta)^2)
  sigma2 <- 1/rgamma(1, nu.n / 2, ss.n / 2)

  # store sample
  BETA[s,] <- beta
  SIGMA2[s] <- sigma2
}
```

```
# posterior distribution of sigma2
plot(density(SIGMA2),
     main = 'Posterior Distribution of Sigma Squared',
     xlab = 'Sigma Squared')
```



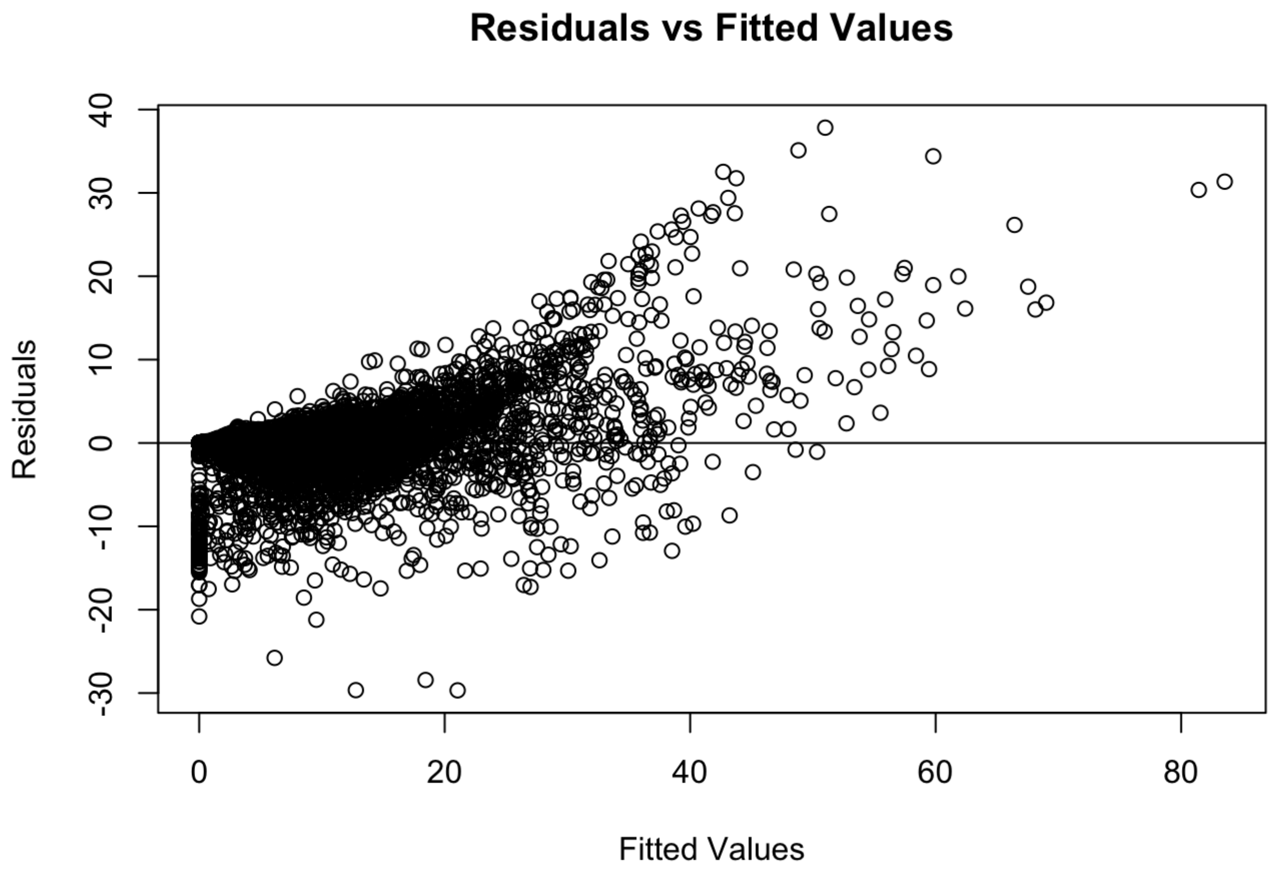
```
# 95% confidence intervals for each element of beta
CI <- NULL
for (i in 1: dim(BETA)[2]){
  print(paste('The 95% CI for Beta', i, "is: "))
  print(quantile(BETA[,i], c(0.0025, 0.975)))
  CI <- c(CI, quantile(BETA[,i], c(0.0025, 0.975)))
}
```

```
## [1] "The 95% CI for Beta 1 is: "
##     0.25%     97.5%
## 0.3794342 0.6672999
## [1] "The 95% CI for Beta 2 is: "
##       0.25%       97.5%
## -0.07597283  0.02150432
## [1] "The 95% CI for Beta 3 is: "
##       0.25%       97.5%
## -0.02184565  0.01643633
## [1] "The 95% CI for Beta 4 is: "
##       0.25%       97.5%
## -0.01622588  0.09902869
## [1] "The 95% CI for Beta 5 is: "
##         0.25%         97.5%
## -0.0315009542  0.0009156621
## [1] "The 95% CI for Beta 6 is: "
##     0.25%     97.5%
## 0.1707828 0.3449928
## [1] "The 95% CI for Beta 7 is: "
##      0.25%      97.5%
## 0.09072786 0.20750827
## [1] "The 95% CI for Beta 8 is: "
##       0.25%       97.5%
## 0.002263687 0.006809795
## [1] "The 95% CI for Beta 9 is: "
##       0.25%       97.5%
## -0.30283697  0.06189633
```

Effluent, soil, street (weak, very small coefficient), and swine are the main sources of the water sample as their 95% CI do not contain 0.

Part b

```r
# check constant variance condition

# obtain residual
beta.mean <- apply(BETA, 2, mean)
y.pred <- X %*% beta.mean
residual <- y - y.pred

# residual plot
plot(y, residual,
    main = 'Residuals vs Fitted Values',
    ylab = 'Residuals',
    xlab = 'Fitted Values')
abline(0,0)
```

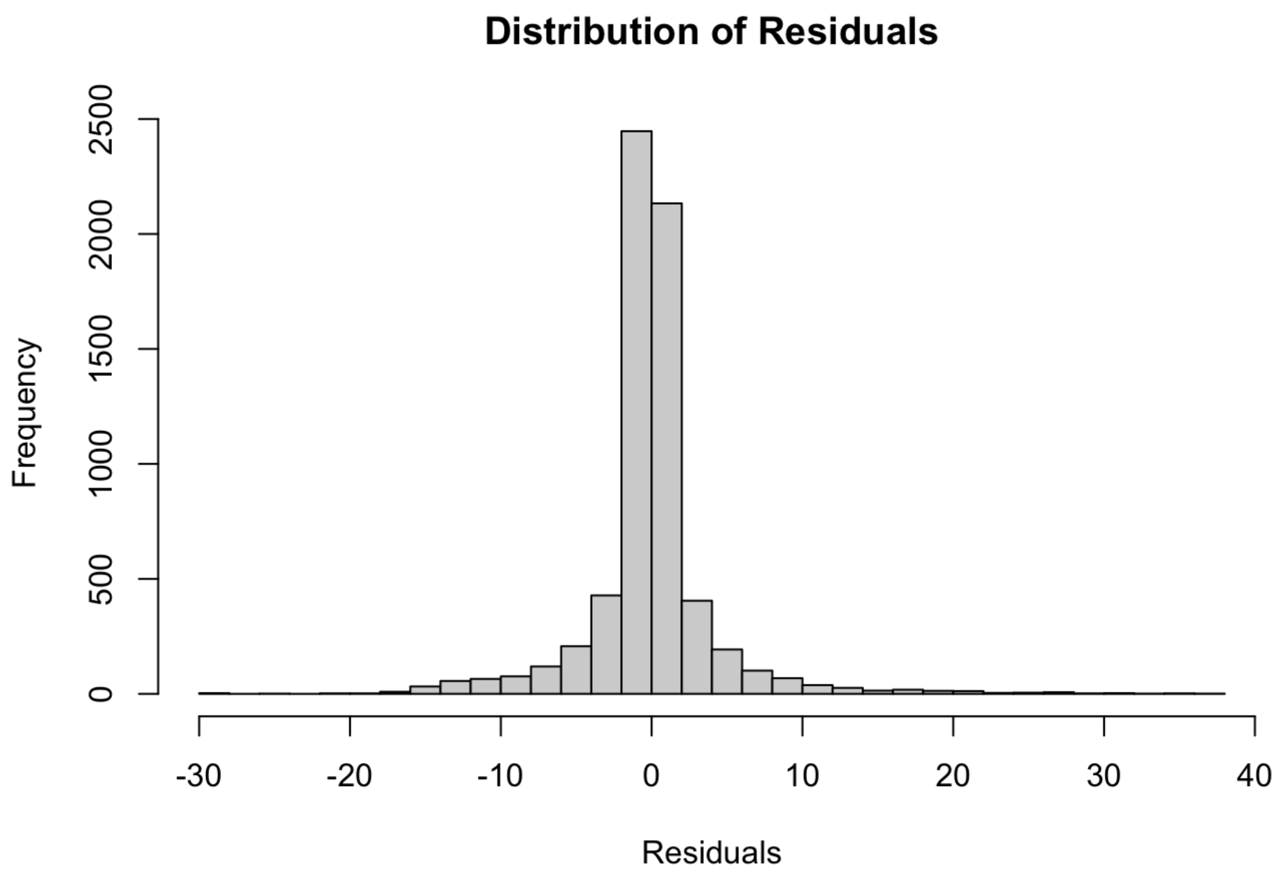


The constant variance condition seems to be violated as the residuals are not randomly spread across all fitted values of y. In fact, as the fitted value increases, the residual increases.

Check Independence Condition

The independence condition seems to be satisfied since y is the vectorization of a spectroscopy image of a water sample taken from the Neuse River in North Carolina. It is reasonable to assume the entries of the vectors are uncorrelated.

```
# check normal condition
# distribution of y
hist(residual,
     main = 'Distribution of Residuals',
     xlab = 'Residuals',
     breaks = 30)
```



The distribution of the residuals approximately normal, and the sample size is larger than 30, thus the normal condition is satisfied.

Part c

Since it doesn't make sense for the coefficients of beta to be negative, a modification to the prior distribution for beta could be picking a distribution with a positive support, such as beta or gamma.

Suppose we use beta distribution as the prior distribution for beta. We first need to derive the posterior distribution of beta. For the Gibbs sampler, with starting values for sigma2 and beta, in each iteration we update and get a new beta from the posterior calculation formula, and get a new sigma2 (just as we did in part a). In the end, we would get a storage vector BETA of all positive values, since beta distribution lies in the first quadrant.