Homework 5: Bayesian Regression and Model Slection

Alexander Grunenwald

2024-11-13

```
source("../../demos/demoInclass/regression_gprior.R")
source("../../demos/demoInclass/backselect.R")
library(mvtnorm)
library(ggplot2)
```

Crime data Set: Fitting and compare OLS to Bayes Regression

```
crime.data <- read.table("crime.txt", header = TRUE)
yf <- crime.data[, 1]; Xf <- as.matrix(crime.data[,-1])
```

Question 1.1

Fitting a OLS baseline model.

```
crime.regression <- lm(yf~Xf - 1)
coef(crime.regression)
```

```
##           XfM          XfSo          XfEd         XfPo1         XfPo2
##  0.2865177028 -0.0001179958  0.5445161778  1.4716146465 -0.7817757455
##          XfLF         XfM.F         XfPop          XfNW          XfU1
## -0.0659672893  0.1313002714 -0.0702910179  0.1090590127 -0.2705407273
##          XfU2         XfGDP        XfIneq        XfProb        XfTime
##  0.3687335028  0.2380580097  0.7262918200 -0.2852262729 -0.0615771841
```

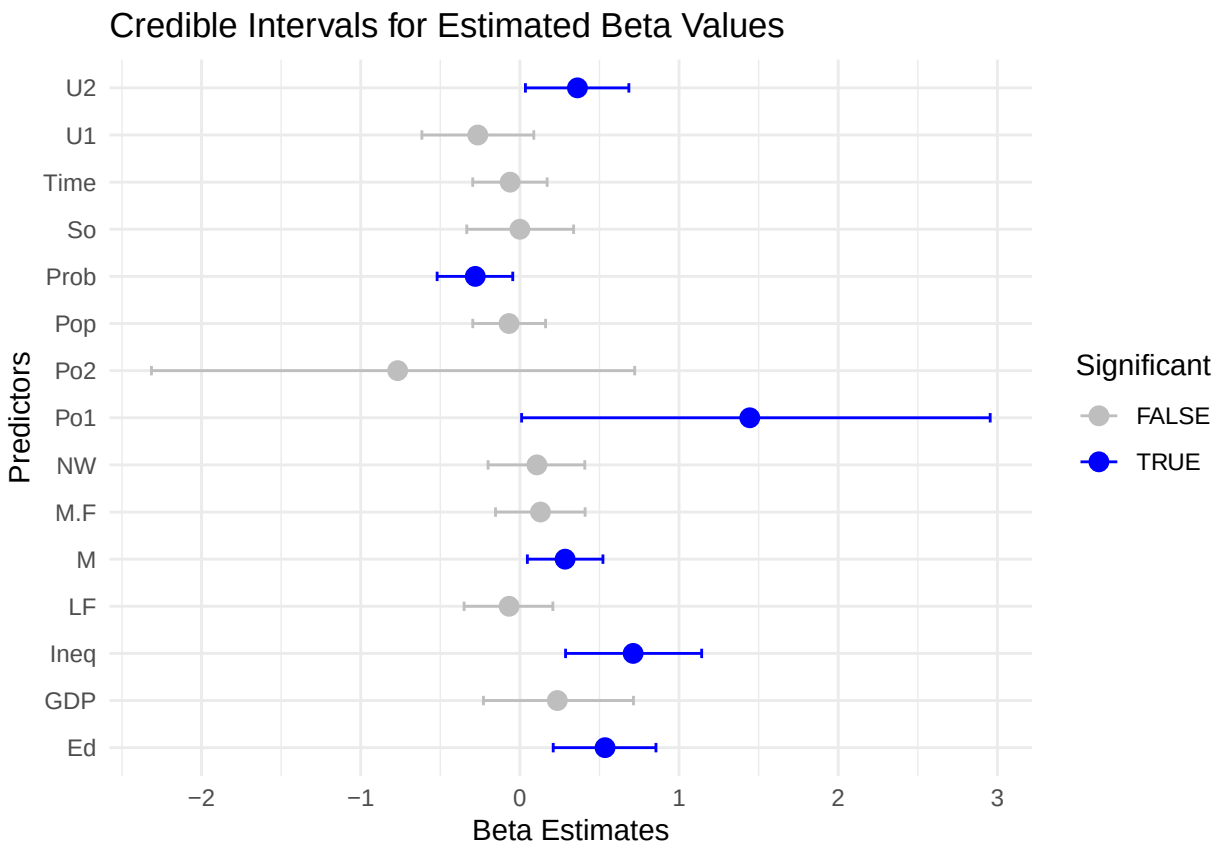
Fitting Bayesian g-prior linear regression model with parameters g=n=47 and a=b=1 with 5000 samples.

```
# g = n=47,a=1,b=1,
g <- 47; n <- nrow(crime.data); a <- 1; b <- 1; S <- 5000
beta_sam <- matrix(0, nrow = S, ncol = 15)

## sample sigma^2
a_tilde <- a + n / 2
b_tilde <- b + sum(yf ^ 2) / 2 - g / (2 * (g + 1)) *
  t(yf) %*% Xf %*% solve(t(Xf) %*% Xf) %*% t(Xf) %*% yf
sigma2_sam <- 1 / rgamma(S, a_tilde, b_tilde)

## sample beta
Sig <- g / (g + 1) * solve(t(Xf) %*% Xf)
beta_n <- Sig %*% t(Xf) %*% yf
for(i in 1 : S){
  beta_sam[i, ] <- rmvnorm(1, beta_n, Sig * sigma2_sam[i])
}
```

Evaluation of the credible Intervals and determination of significant predictors.

Significant predictors according the plot below are: - U2 - Prob - M - Ineq - Ed

```
lower.bound <- apply(beta_sam, 2, quantile, prob = 0.025)
upper.bound <- apply(beta_sam, 2, quantile, prob = 0.975)
beta.cred.interval <- data.frame(lowerBound = lower.bound, upperBound = upper.bound, row.names = col

beta.cred.interval$Predictor <- rownames(beta.cred.interval)
beta.cred.interval$Significant <- with(beta.cred.interval, lowerBound > 0 | upperBound < 0)

ggplot(beta.cred.interval, aes(x = Predictor, y = meanBeta, color = Significant)) +
  geom_point(size = 3) +
  geom_errorbar(aes(ymin = lowerBound, ymax = upperBound), width = 0.2) +
  labs(title = "Credible Intervals for Estimated Beta Values",
       x = "Predictors", y = "Beta Estimates") +
  scale_color_manual(values = c("TRUE" = "blue", "FALSE" = "gray")) +
  theme_minimal() +
  coord_flip()
```

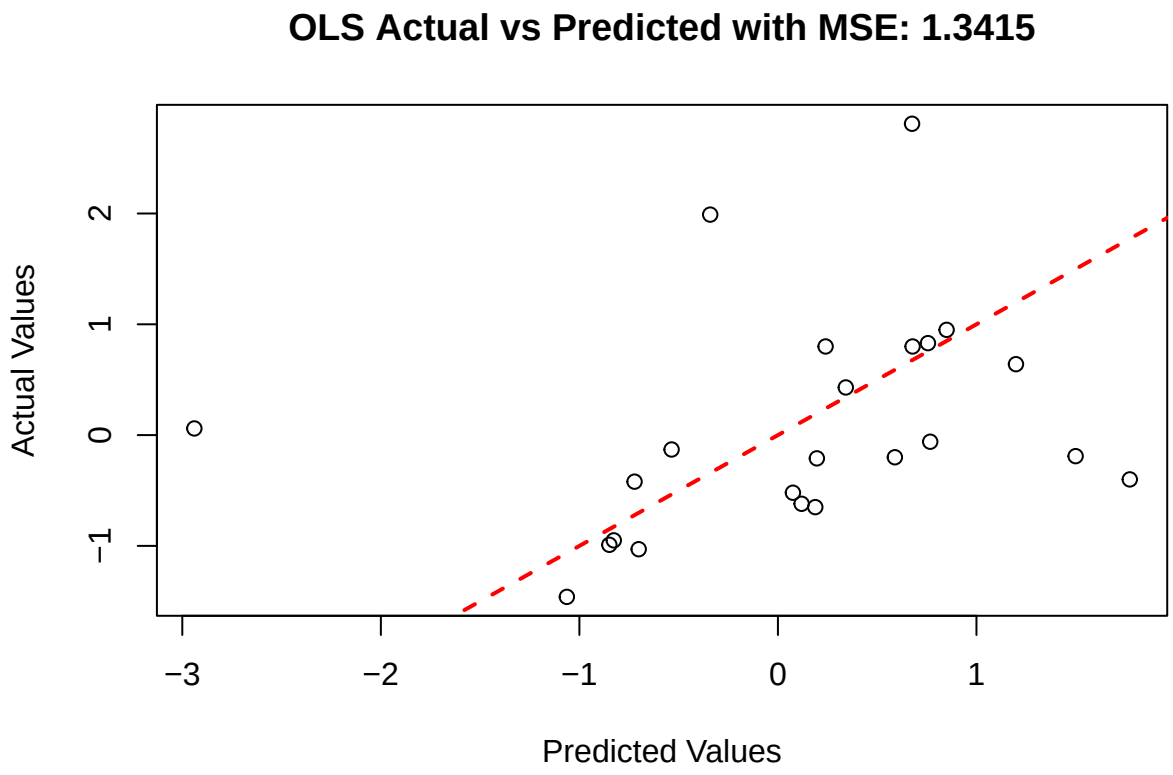


Question 1.2.1

```
set.seed(42)
train.ratio <- 0.5
train.indices <- sample(1:n, size = round(train.ratio * n))
Xf.train <- Xf[train.indices,]; Xf.test <- Xf[-train.indices,]
yf.train <- yf[train.indices ]; yf.test <- yf[-train.indices ]
```

```
betas <- solve(t(Xf.train) %*% Xf.train) %*% t(Xf.train) %*% yf.train
ols.predicts <- Xf.test %*% betas
```

```r
mse <- mean((yf.test - ols.predicts)^2)

plot(ols.predicts, yf.test,
     main = paste("OLS Actual vs Predicted with MSE:", round(mse, 4)),
     xlab = "Predicted Values", ylab = "Actual Values")
abline(0, 1, col = "red", lwd = 2, lty = 2)
```

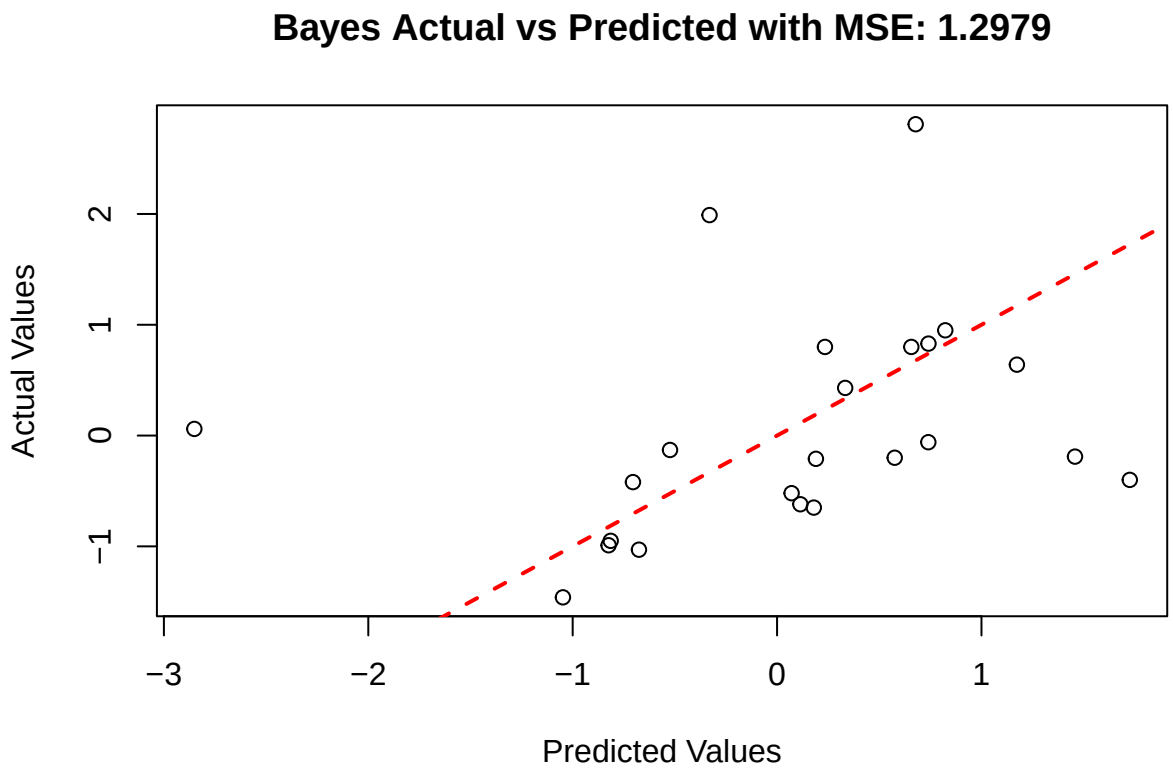


Question 1.2.2

```r
# g = n=47,a=1,b=1,
g <- 47; n.train <- nrow(Xf.train); a <- 1; b <- 1; S <- 5000
beta_sam <- matrix(0, nrow = S, ncol = 15)

## sample sigma^2
a_tilde <- a + n.train / 2
b_tilde <- b + sum(yf.train ^ 2) / 2 - g / (2 * (g + 1)) *
  t(yf.train) %*% Xf.train %*% solve(t(Xf.train) %*% Xf.train) %*% t(Xf.train) %*% yf.train
sigma2_sam <- 1 / rgamma(S, a_tilde, b_tilde)

## sample beta
Sig <- g / (g + 1) * solve(t(Xf.train) %*% Xf.train)
beta_n <- Sig %*% t(Xf.train) %*% yf.train
for(i in 1 : S){
  beta_sam[i, ] <- rmvnorm(1, beta_n, Sig * sigma2_sam[i])
}
```

```r
lower.bound <- apply(beta_sam, 2, quantile, prob = 0.025)
upper.bound <- apply(beta_sam, 2, quantile, prob = 0.975)
beta.cred.interval <- data.frame(lowerBound = lower.bound, upperBound = upper.bound, row.names = col
```

```r
beta.cred.interval$Predictor <- rownames(beta.cred.interval)
beta.cred.interval$Significant <- with(beta.cred.interval, lowerBound > 0 | upperBound < 0)

bayes.predict <- Xf.test %*% beta.cred.interval$meanBeta

mse <- mean((yf.test - bayes.predict)^2)

plot(bayes.predict, yf.test,
     main = paste("Bayes Actual vs Predicted with MSE:", round(mse, 4)),
     xlab = "Predicted Values", ylab = "Actual Values")
abline(0, 1, col = "red", lwd = 2, lty = 2)
```



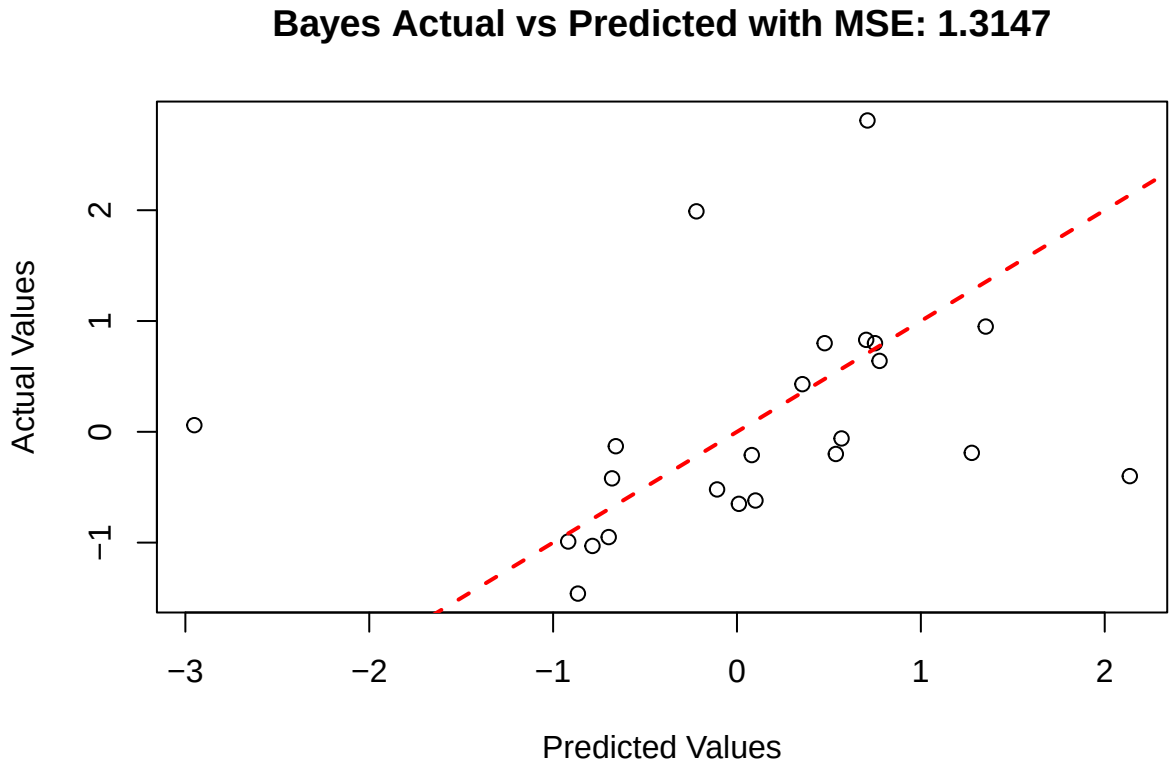
When comparing the two metrics, the bayes model outperforms the OLS model by about 0.04 MSE.

Question 1.2.3

```r
S <- 10000
Z <- matrix(NA, S, dim(Xf)[2]) ## store the MCMC samples

## starting value
z <- rep(1, dim(Xf)[2])

## the initial log of p(y|X,z); the prior parameter values that we
## specified are the default values in this function
lpy.c <- lpy.X(yf.train, Xf.train[, z == 1, drop = FALSE])

for(s in 1 : S){

  ## update each coordinate of z
  for(j in sample(1 : dim(Xf)[2])){
```

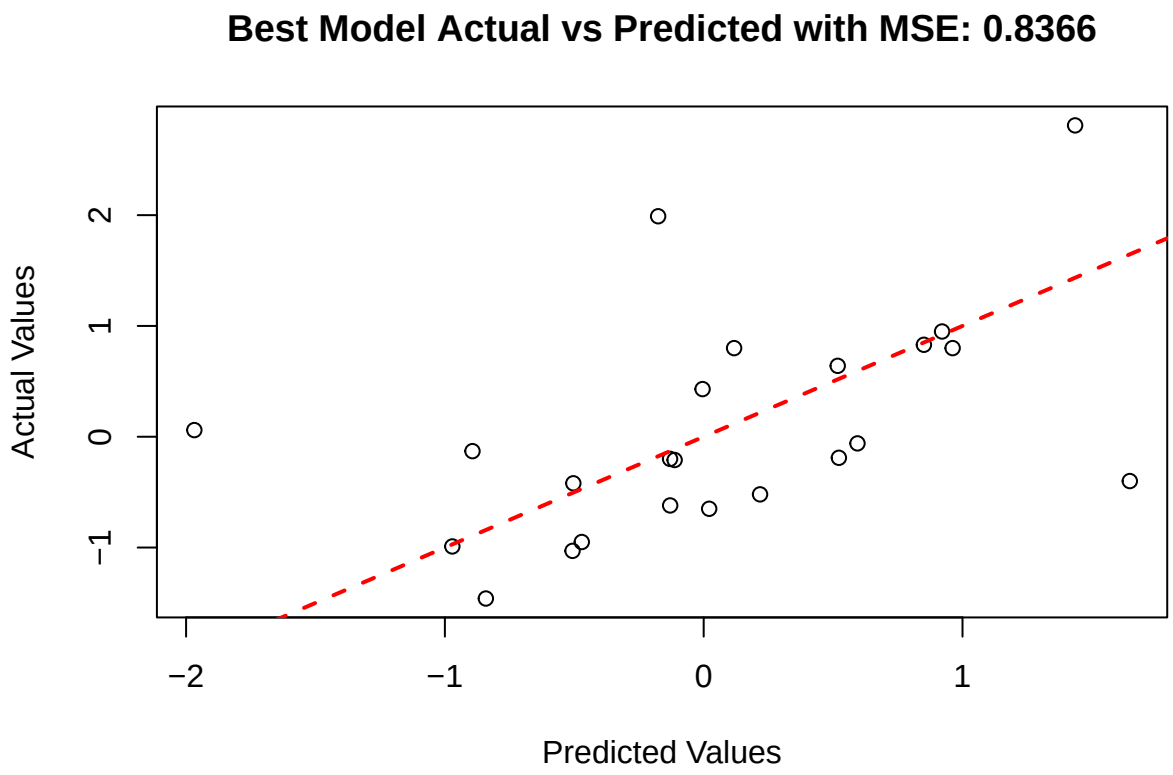
```
    zp <- z; zp[j] <- 1 - zp[j];
    lpy.p <- lpy.X(yf.train, Xf.train[, zp == 1, drop = FALSE])
    r <- (lpy.p - lpy.c) * (-1) ^ (zp[j] == 0)
    z[j] <- rbinom(1, 1, 1 / (1 + exp(-r))) ## sampling from bernoulli
    if(z[j] == zp[j]) {lpy.c <- lpy.p}
  }

  Z[s, ] <- z
}
```

```
inclusion_probs <- colMeans(Z)
best_model <- ifelse(inclusion_probs > 0.5, 1, 0)

Xf_best.train <- Xf.train[, best_model == 1, drop = FALSE]

# g = n=47,a=1,b=1,
g <- 47; n.train <- nrow(Xf_best.train); a <- 1; b <- 1; S <- 5000
beta_sam <- matrix(0, nrow = S, ncol = dim(Xf_best.train)[2])

## sample sigma^2
a_tilde <- a + n.train / 2
b_tilde <- b + sum(yf.train ^ 2) / 2 - g / (2 * (g + 1)) *
  t(yf.train) %*% Xf_best.train %*% solve(t(Xf_best.train) %*% Xf_best.train) %*% t(Xf_best.train) %
sigma2_sam <- 1 / rgamma(S, a_tilde, b_tilde)

## sample beta
Sig <- g / (g + 1) * solve(t(Xf_best.train) %*% Xf_best.train)
beta_n <- Sig %*% t(Xf_best.train) %*% yf.train
for(i in 1 : S){
  beta_sam[i, ] <- rmvnorm(1, beta_n, Sig * sigma2_sam[i])
}
# Make predictions on the test set
best_model_betas <- apply(beta_sam, 2, mean)
Xf_test_best <- Xf.test[, best_model == 1, drop = FALSE]
best_model_predicts <- Xf_test_best %*% best_model_betas

mse <- mean((yf.test - best_model_predicts)^2)

plot(best_model_predicts, yf.test,
     main = paste("Bayes Actual vs Predicted with MSE:", round(mse, 4)),
     xlab = "Predicted Values", ylab = "Actual Values")
abline(0, 1, col = "red", lwd = 2, lty = 2)
```




```r
S <- 10000
BETA <- Z <- matrix(NA, S, dim(Xf.train)[2]) ## store the MCMC samples
z <- rep(1, dim(Xf.train)[2])
lpy.c <- lpy.X(yf.train, Xf.train[, z == 1, drop = FALSE])

for(s in 1 : S){
  for(j in sample(1 : dim(Xf.train)[2])){
    zp <- z; zp[j] <- 1 - zp[j];
    lpy.p <- lpy.X(yf.train, Xf.train[, zp == 1, drop = FALSE])
    r <- (lpy.p - lpy.c) * (-1) ^ (zp[j] == 0)
    z[j] <- rbinom(1, 1, 1 / (1 + exp(-r)))
    if(z[j] == zp[j]) {lpy.c <- lpy.p}
  }
  beta <- z

  ## the function lm.gprior generates samples for beta and sigma^2
  if(sum(z) > 0) {beta[z == 1] <- lm.gprior(yf.train, Xf.train[, z == 1, drop = FALSE], S = 1)$beta}

  Z[s, ] <- z
  BETA[s, ] <- beta
}

inclusion_probs <- colMeans(Z)
best_model <- ifelse(inclusion_probs > 0.5, 1, 0)

best_BETA <- BETA[, best_model == 1, drop = FALSE]

beta_avg <- apply(best_BETA, 2, mean)

Xf_test_best <- Xf.test[, best_model == 1, drop = FALSE]
model_avg_predicts <- Xf_test_best %*% beta_avg

mse_avg <- mean((yf.test - model_avg_predicts)^2)
```

```r
plot(model_avg_predicts, yf.test,
     main = paste("Best Model Actual vs Predicted with MSE:", round(mse_avg, 4)),
     xlab = "Predicted Values", ylab = "Actual Values")
abline(0, 1, col = "red", lwd = 2, lty = 2)
```



Diabetes Data Set: Fitting and compare Bayes Regression model Selection

Qustion 2.1

```r
diabetes.data <- read.table("azdiabetes.txt", header = TRUE)
yf <- diabetes.data[, 2]
Xf <- cbind(rep(1, length(yf)), as.matrix(diabetes.data[,-c(2, 8)]))

# Fitting ols model
diab.regression <- lm(yf~Xf)
coef(diab.regression)
```

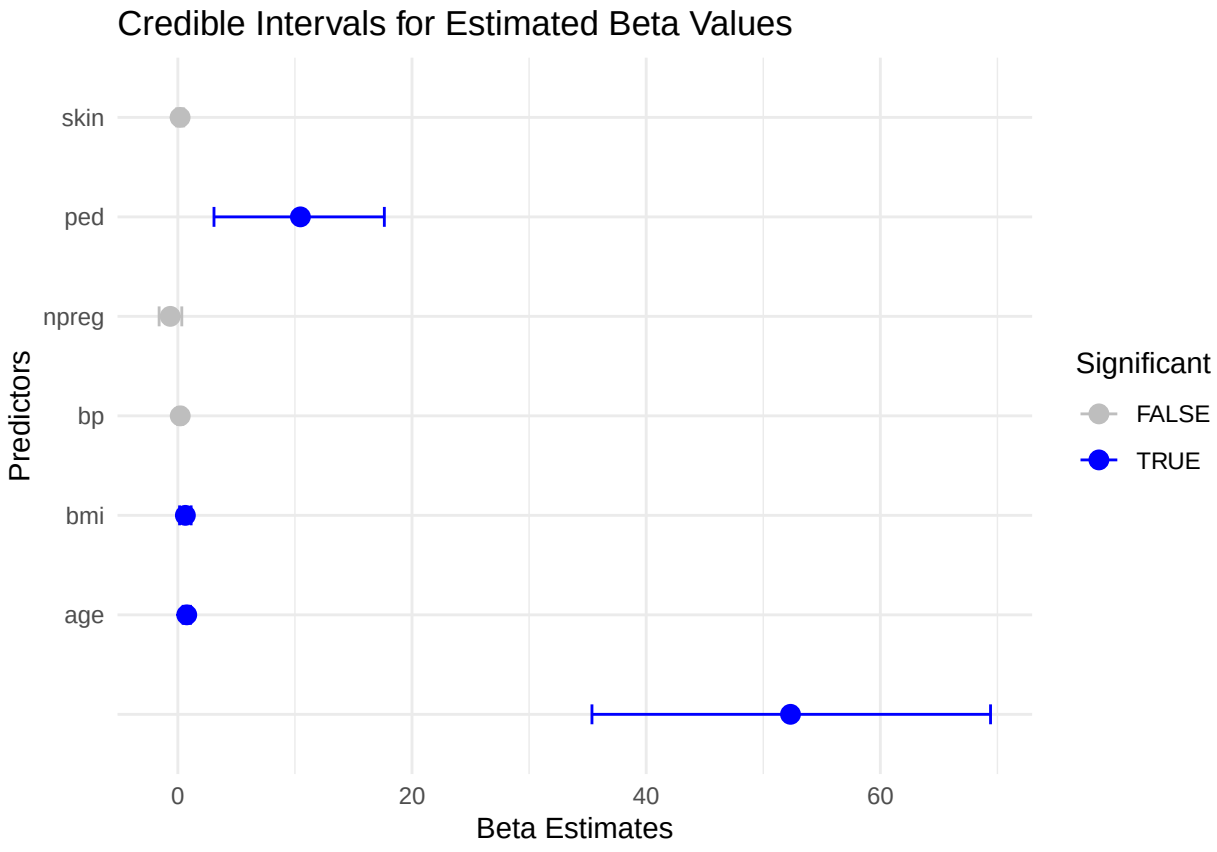
```
## (Intercept)          Xf     Xfnpreg        Xfbp      Xfskin       Xfbmi
##  52.3052289          NA  -0.6571310   0.2052805   0.1925988   0.6443543
##       Xfped       Xfage
##  10.5484016   0.7666783
```

```r
#  g=n=532,a=1,b=1,
g <- n <- nrow(Xf); a <- b <- 1; S <- 5000
beta_sam <- matrix(0, nrow = S, ncol = dim(Xf)[2])

## sample sigma^2
a_tilde <- a + n / 2
b_tilde <- b + sum(yf ^ 2) / 2 - g / (2 * (g + 1)) *
```

```
  t(yf) %*% Xf %*% solve(t(Xf) %*% Xf) %*% t(Xf) %*% yf
sigma2_sam <- 1 / rgamma(S, a_tilde, b_tilde)

## sample beta
Sig <- g / (g + 1) * solve(t(Xf) %*% Xf)
beta_n <- Sig %*% t(Xf) %*% yf
for(i in 1 : S){
  beta_sam[i, ] <- rmvnorm(1, beta_n, Sig * sigma2_sam[i])
}
```

The variables that seem strongly predictive of Glucose level are ped, bmi, and age

```
lower.bound <- apply(beta_sam, 2, quantile, prob = 0.025)
upper.bound <- apply(beta_sam, 2, quantile, prob = 0.975)
beta.cred.interval <- data.frame(lowerBound = lower.bound, upperBound = upper.bound, row.names = col

beta.cred.interval$Predictor <- rownames(beta.cred.interval)
beta.cred.interval$Significant <- with(beta.cred.interval, lowerBound > 0 | upperBound < 0)

ggplot(beta.cred.interval, aes(x = Predictor, y = meanBeta, color = Significant)) +
  geom_point(size = 3) +
  geom_errorbar(aes(ymin = lowerBound, ymax = upperBound), width = 0.2) +
  labs(title = "Credible Intervals for Estimated Beta Values",
       x = "Predictors", y = "Beta Estimates") +
  scale_color_manual(values = c("TRUE" = "blue", "FALSE" = "gray")) +
  theme_minimal() +
  coord_flip()
```



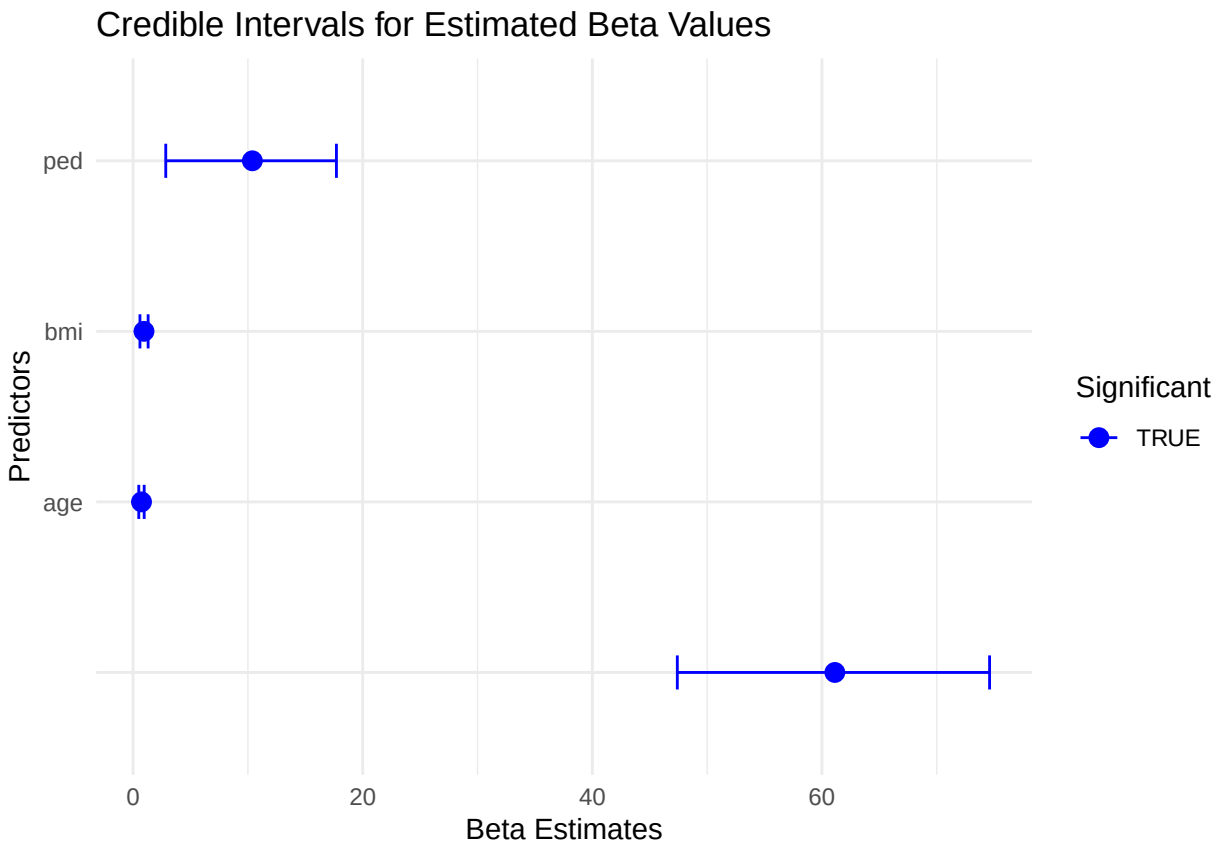
Question 2.2

```
S <- 10000
Z <- matrix(NA, S, dim(Xf)[2]) ## store the MCMC samples

## starting value
z <- rep(1, dim(Xf)[2])

## the initial log of p(y|X,z); the prior parameter values that we
## specified are the default values in this function
lpy.c <- lpy.X(yf, Xf[, z == 1, drop = FALSE], nu0 = 2, s20 = 1)

for(s in 1 : S){

  ## update each coordinate of z
  for(j in sample(1 : dim(Xf)[2])){

    zp <- z; zp[j] <- 1 - zp[j];
    lpy.p <- lpy.X(yf, Xf[, zp == 1, drop = FALSE], nu0 = 2, s20 = 1)
    r <- (lpy.p - lpy.c) * (-1) ^ (zp[j] == 0)
    z[j] <- rbinom(1, 1, 1 / (1 + exp(-r))) ## sampling from bernoulli
    if(z[j] == zp[j]) {lpy.c <- lpy.p}
  }

  Z[s, ] <- z
}
```

```
inclusion_probs <- colMeans(Z)
best_model <- ifelse(inclusion_probs > 0.5, 1, 0)

Xf_best <- Xf[, best_model == 1, drop = FALSE]

# g = n=47,a=1,b=1,
g <- n <- nrow(Xf_best); a <- b <- 1; S <- 5000
beta_sam <- matrix(0, nrow = S, ncol = dim(Xf_best)[2])

## sample sigma^2
a_tilde <- a + n / 2
b_tilde <- b + sum(yf ^ 2) / 2 - g / (2 * (g + 1)) *
  t(yf) %*% Xf_best %*% solve(t(Xf_best) %*% Xf_best) %*% t(Xf_best) %*% yf
sigma2_sam <- 1 / rgamma(S, a_tilde, b_tilde)

## sample beta
Sig <- g / (g + 1) * solve(t(Xf_best) %*% Xf_best)
beta_n <- Sig %*% t(Xf_best) %*% yf
for(i in 1 : S){
  beta_sam[i, ] <- rmvnorm(1, beta_n, Sig * sigma2_sam[i])
}
```

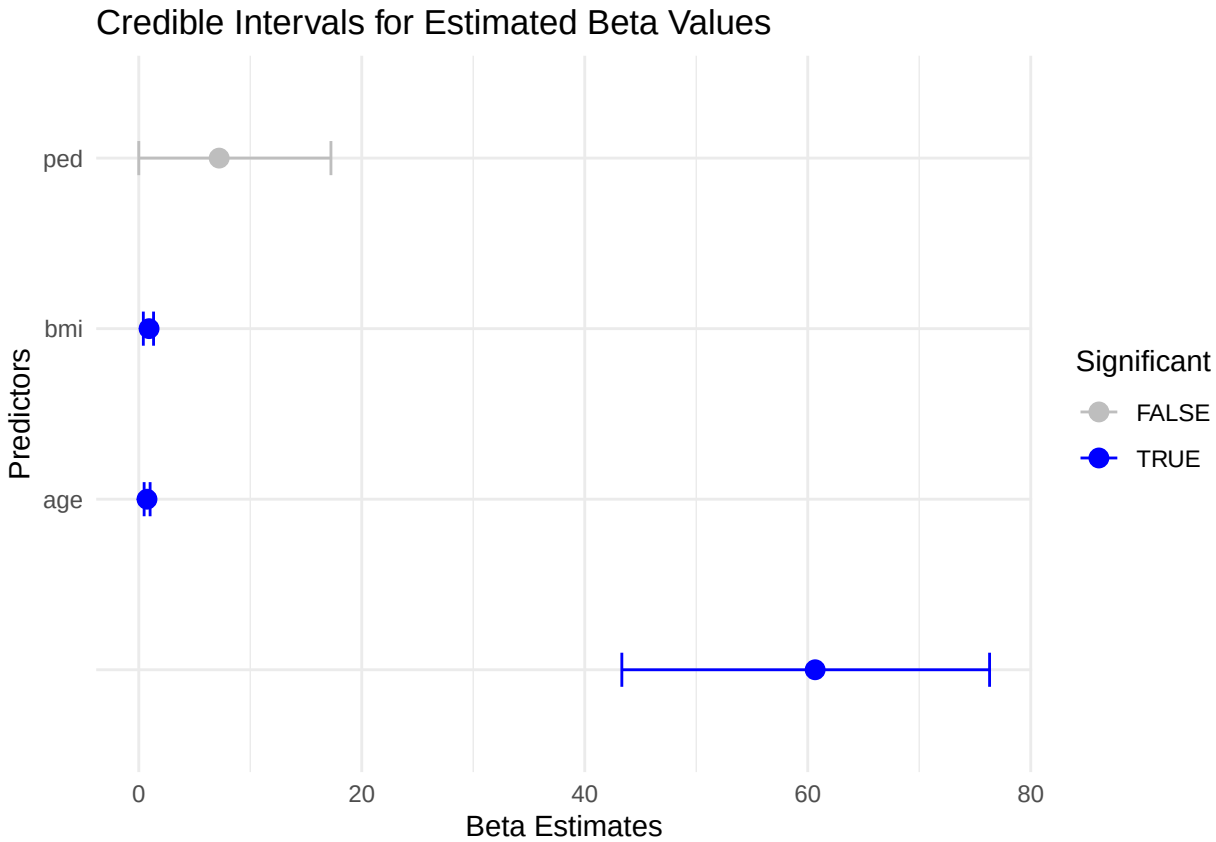
```
lower.bound <- apply(beta_sam, 2, quantile, prob = 0.025)
upper.bound <- apply(beta_sam, 2, quantile, prob = 0.975)
beta.cred.interval <- data.frame(lowerBound = lower.bound, upperBound = upper.bound, row.names = col

beta.cred.interval$Predictor <- rownames(beta.cred.interval)
beta.cred.interval$Significant <- with(beta.cred.interval, lowerBound > 0 | upperBound < 0)

ggplot(beta.cred.interval, aes(x = Predictor, y = meanBeta, color = Significant)) +
```

```
  geom_point(size = 3) +
  geom_errorbar(aes(ymin = lowerBound, ymax = upperBound), width = 0.2) +
  labs(title = "Credible Intervals for Estimated Beta Values",
       x = "Predictors", y = "Beta Estimates") +
  scale_color_manual(values = c("TRUE" = "blue", "FALSE" = "gray")) +
  theme_minimal() +
  coord_flip()
```




```
S <- 10000
BETA <- Z <- matrix(NA, S, dim(Xf)[2]) ## store the MCMC samples
z <- rep(1, dim(Xf)[2])
lpy.c <- lpy.X(yf, Xf[, z == 1, drop = FALSE], nu0 = 2, s20 = 1)

for(s in 1 : S){
  for(j in sample(1 : dim(Xf)[2])){
    zp <- z; zp[j] <- 1 - zp[j];
    lpy.p <- lpy.X(yf, Xf[, zp == 1, drop = FALSE])
    r <- (lpy.p - lpy.c) * (-1) ^ (zp[j] == 0)
    z[j] <- rbinom(1, 1, 1 / (1 + exp(-r)))
    if(z[j] == zp[j]) {lpy.c <- lpy.p}
  }
  beta <- z

  ## the function lm.gprior generates samples for beta and sigma^2
  if(sum(z) > 0) {beta[z == 1] <- lm.gprior(yf, Xf[, z == 1, drop = FALSE], S = 1, nu0 = 2, s20 = 1)

  Z[s, ] <- z
  BETA[s, ] <- beta
}
```

```r
inclusion_probs <- colMeans(Z)
best_model <- ifelse(inclusion_probs > 0.5, 1, 0)

best_BETA <- BETA[, best_model == 1, drop = FALSE]
Xf_best <- Xf[, best_model == 1, drop = FALSE]



lower.bound <- apply(best_BETA, 2, quantile, prob = 0.025)
upper.bound <- apply(best_BETA, 2, quantile, prob = 0.975)
beta.cred.interval <- data.frame(lowerBound = lower.bound, upperBound = upper.bound, row.names = col

beta.cred.interval$Predictor <- rownames(beta.cred.interval)
beta.cred.interval$Significant <- with(beta.cred.interval, lowerBound > 0 | upperBound < 0)

ggplot(beta.cred.interval, aes(x = Predictor, y = meanBeta, color = Significant)) +
  geom_point(size = 3) +
  geom_errorbar(aes(ymin = lowerBound, ymax = upperBound), width = 0.2) +
  labs(title = "Credible Intervals for Estimated Beta Values",
       x = "Predictors", y = "Beta Estimates") +
  scale_color_manual(values = c("TRUE" = "blue", "FALSE" = "gray")) +
  theme_minimal() +
  coord_flip()
```



When comparing the answer from 2.a) to 2.b), the pedestrian variable is no longer significant in the model.