Homework 3

John Brandt

2/21/2018

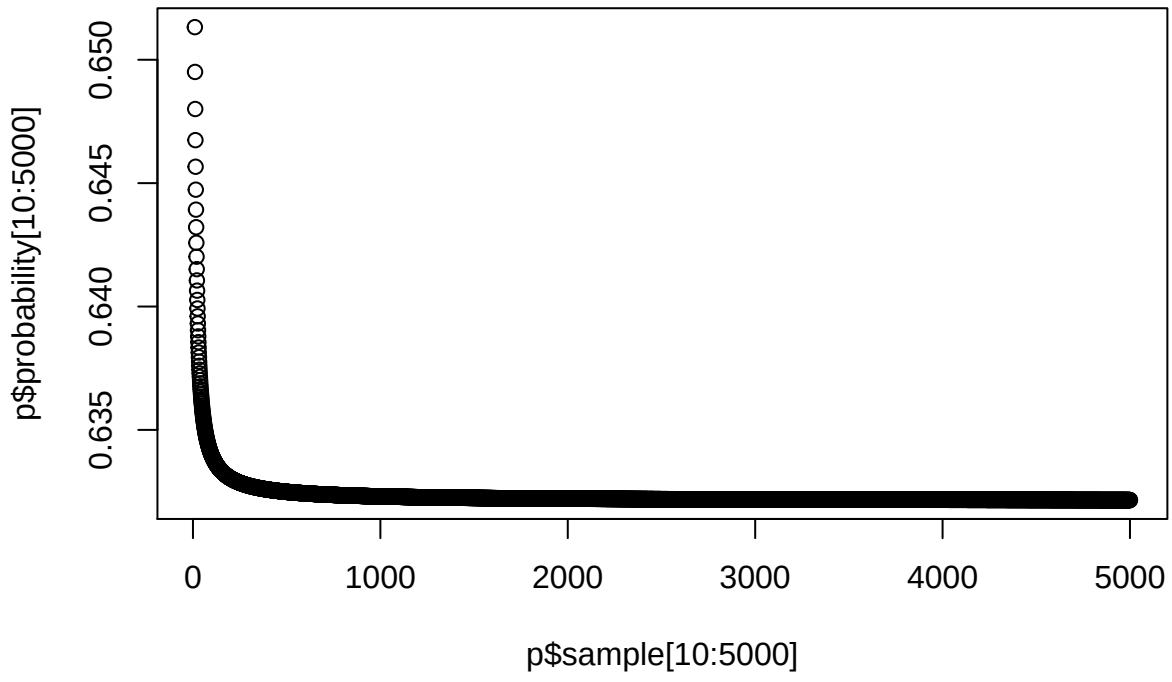
Question 1

a) The probability that observation i gets included one or more times in a bootstrap sample, in terms of n is $P = 1 - \left(1 - (\frac{1}{n})\right)^n$

The probability of a draw containing some sample with replacement is simply $\frac{1}{n}$, and so the probability that it is not chosen is $1 - \frac{1}{n}$. For n number of draws, the probability that a sample is not drawn in any of them is simply the probability that it isn't drawn in one sample raised to the power of n. The probability that the sample is drawn at least once during the bootstrap is then 1 - the probability that it is not.

b) The probability that i is included one or more times in an n sample botostrap converges on e as n increases from zero.

```
sample <- seq(10,5000, by = 1)

p <- list(rep(NA, 5000))
p <- as.data.frame(p)
colnames(p) <- c("probability")
p$sample <- seq(1,5000, by =1)

for (n in sample) {
  p[n,1] <- 1-((1-(1/n))^n)
}

plot(p$sample[10:5000], p$probability[10:5000])
```



Question 2 - Ridge Regression

Let $Y \in \mathbb{R}^n, X \in \mathbb{R}^{n \times p}$, and $\beta \in \mathbb{R}^p$. Consider the ridge regression cost function:

$$\begin{aligned} f(\beta) &= \|Y - X\beta\|_2^2 + \lambda\|\beta\|_2^2 \\ &= \sum_{i=1}^{n}(Y_i - X_{i1}\beta_1 - ... - X_{ip}\beta_p)^2 + \lambda\sum_{j=1}^{p}\beta_j^2, \end{aligned}$$

where λ is a fixed number. What value of β in $\mathbb{R}^p$ minimizes $f(\beta)$?

$$\begin{aligned} f(\beta) &= \|Y - X\beta\|_2^2 + \lambda\|\beta\|_2^2 \\ &= \sum_{i=1}^{n}(Y_i - X_{i1}\beta_1 - ... - X_{ip}\beta_p)^2 + \lambda\sum_{j=1}^{p}\beta_j^2 \\ &= (Y - X\beta)^T(Y - X\beta) + \lambda\beta^T\beta \\ &= Y^TY - \beta^TX^TY - Y^TX\beta + \beta^TX^TX\beta + \lambda\beta^T\beta \\ &= Y^TY - \beta^TX^TY - \beta^TX^TY + \beta^TX^TX\beta + \beta^T\lambda I\beta \\ &= Y^TY - 2\beta^TX^TY + \beta^T(X^TX + \lambda I)\beta \end{aligned}$$

In a partial derivative with respect to β, Y^TY is just a constant and will drop out. The $\frac{\delta}{\delta\beta}$ of $-2\beta^TX^TY$ is simply $-2X^TY$. Using the matrix differentiation rules $\frac{\delta X^TAx}{\delta X} = X^T(A + A^T)$ and $\frac{\delta(X^TA)}{\delta X} = \frac{\delta(A^TX)}{\delta X} = A^T$ and the chain rule:

$$\begin{aligned} \frac{\delta RSS}{\delta\beta} &= -2X^TY + 2(X^TX + \lambda I)\beta \\ 2X^TY &= 2(X^TX + \lambda I)\beta \\ X^TY &= \beta(X^TX + \lambda I) \\ \beta &= X^TY(X^TX + \lambda I)^{-1} \end{aligned}$$

Question 3 - Lasso Regression

The quantities below, (a) through (e), change in the below specified manner as s increases from 0.

(a) training RSS of $\hat{\beta}^{(s)}$ - as s increases, the training RSS of $\hat{\beta}^{(s)}$ will initially decrease to a minimum and then exponentially increase and level off. This is because, initially, the variance decreases faster than the bias increases. At some point, the likelihood that the constrained space does not include the true MLE of β causes the bias to increase at a much faster rate than the variance decreases, so the RSS starts to increase.

(b) test RSS of $\hat{\beta}^{(s)}$ as s increases, the test RSS of $\hat{\beta}^{(s)}$ will initially decrease to a minimum and then exponentially increase and level off.

(c) variance of $\hat{\beta}^{(s)}$ - as s increases, the variance of $\hat{\beta}^{(s)}$ decreases because increasing s reduces the space where $\hat{\beta}$ can reside.

(d) squared bias of $\hat{\beta}^{(s)}$ as s increases, the squared bias of $\hat{\beta}^{(s)}$ increases. This is because, as s becomes larger and β has reduced space to exist, we increase the chance that the true MLE of β does not reside in this space.

(e) irreducible error of $\hat{\beta}^{(s)}$ as s increases, the the irreducible error of $\hat{\beta}^{(s)}$ does not change.

Question 4 - Applied Question

a) Data import and cleaning

```
data <- read.csv("communities.csv")
```

Outliers were removed by using the `summary` function. I remove variables that have an excessive (>20%) NA values, using the `colSums` function. 24 of the 128 variables were missing between 60 and 80% of values. `communityname` is removed because having a 1900-level factor is not feasible.

I made QQ plots of the remaining 102 variables to look for outlying datapoints and normality.

```
# summary(data)

names(data[, colSums(is.na(data)) > 0.5])
```

```
 [1] "county"               "community"            "OtherPerCap"
 [4] "LemasSwornFT"         "LemasSwFTPerPop"      "LemasSwFTFieldOps"
 [7] "LemasSwFTFieldPerPop" "LemasTotalReq"        "LemasTotReqPerPop"
[10] "PolicReqPerOffic"     "PolicPerPop"          "RacialMatchCommPol"
[13] "PctPolicWhite"        "PctPolicBlack"        "PctPolicHisp"
[16] "PctPolicAsian"        "PctPolicMinor"        "OfficAssgnDrugUnits"
[19] "NumKindsDrugsSeiz"    "PolicAveOTWorked"     "PolicCars"
[22] "PolicOperBudg"        "LemasPctPolicOnPatr"  "LemasGangUnitDeploy"
[25] "PolicBudgPerPop"
```

```
data <- data[, colSums(is.na(data)) < 0.5]
data <- data[, -c(2)]
data$state <- as.factor(data$state)
```

```
# The below code is not run but demonstrates how
# transformation and factorization was tested for
op <- par(mfrow=c(4,4))
for (i in c(1:(1+15))) {
  qqnorm(data[,i])
}
par(op)
```

Based upon QQ plots of all 102 variables, the following columns were log + 1 transformed.

```
names(data[,c(51,52,53,58,63,70,73,91,92,98,101)])
```

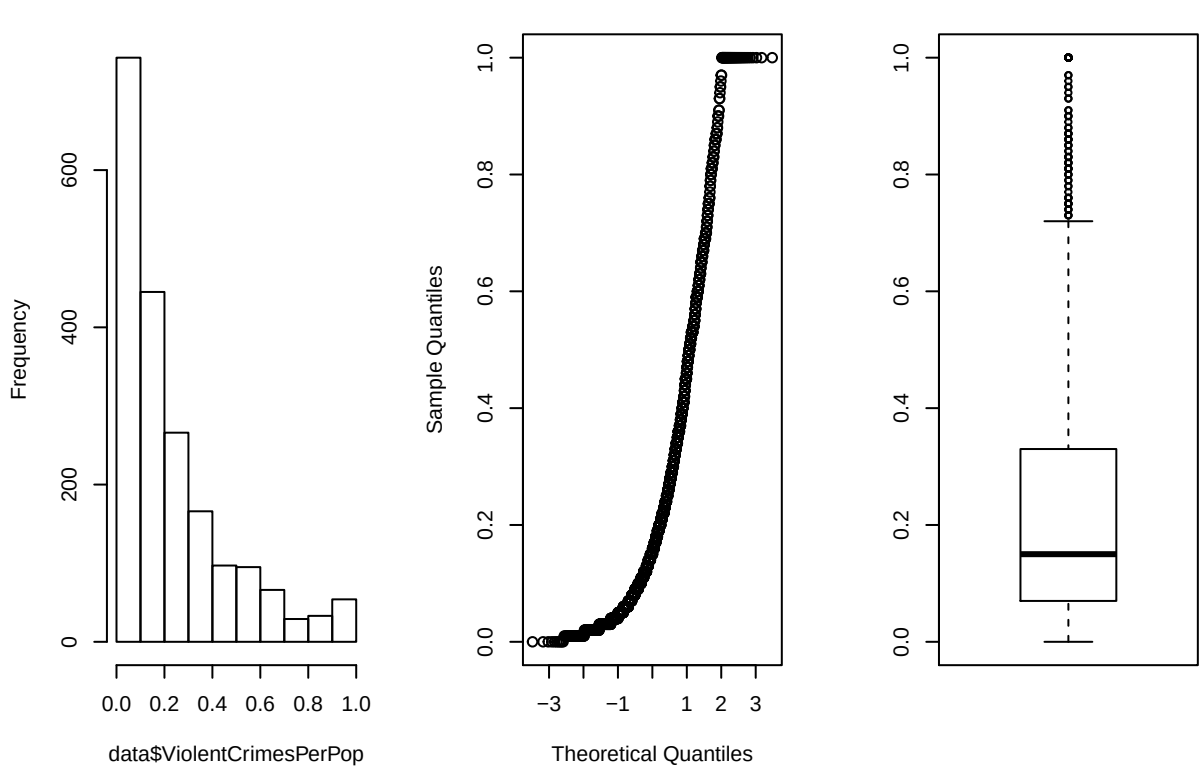
```
 [1] "NumIlleg"            "PctIlleg"            "NumImmig"
 [4] "PctRecentImmig"      "PctNotSpeakEnglWell" "PctPersDenseHous"
 [7] "HousVacant"          "NumInShelters"       "NumStreet"
[10] "LandArea"            "LemasPctOfficDrugUn"
```

```
for (i in c(51,52,53,58,63,70,73,91,92,98,101)) {
  data[,i] <- log(data[,i] + 1)
}
```

The response variable, ViolentCrimesPerPop, is also log transformed based upon seperate histograms and quantile plots.

```
op <- par(mfrow=c(1,3))

hist(data$ViolentCrimesPerPop)
```

```
qqnorm(data$ViolentCrimesPerPop)
boxplot(data$ViolentCrimesPerPop)
```




```
par(op)
```

```
data$ViolentCrimesPerPop <- log(data$ViolentCrimesPerPop + 1)
```

MedNumBR is converted to a factor because it only has 3 unique values.

```
unique(data$MedNumBR)
```

```
[1] 0.5 0.0 1.0
```

```
data$MedNumBR <- factor(data$MedNumBR)
```

Next, I remove highly colinear variables (r $>$ 0.9), in each case maintaining the correlate with the higher correlation with the response variable.

```
# create dataframe with correlation matrix identifying correlations >0.8
correlation <- round(cor(data[,-c(1,2,72)]),1)
correlation <- abs(correlation)
correlation[correlation < 0.8] <- 0
correlation[correlation == 1.0] <- 0
correlation <- as.data.frame(correlation)

# Subset the dataframe to only include columns and rows with at least one high correlation
i <- (colSums(correlation, na.rm=T) != 0)
corr <- correlation[,i]
i <- (rowSums(correlation, na.rm=T) != 0)
corr <- corr[i,]
```

```
# Immediately remove variables with highest correlations
# PctForeignBorn, RentLowQ, RentMedian,RentHighQ, OwnOccHiQuart, OwnOccLowQuart,
# OwnOccMedVal, PctPopUnderPov, PersPerFam, medFamIncome
# Highly correlated variables

# correlations between correlated variables and response concluded:
# agePct16t24, agePct12t21 -> remove agePct12t21

# agePct65up, pctWSocSec -> remove agePct65up

# racePctHisp, PctSpeakEnglOnly -> remove pct speak english only

# householdsize, PersPerOccupHous -> remove householdsize

# population, HousVacant -> remove pctHousVacant - also covered by PctHousOccup

#medIncome, rentHighQ, rentMedian, medIncome, medRent, perCapInc, medIncome
# remove rentMedian, perCapInc, rentHighQ, medRent

#agePct16t24, agePt12t29 -> remove agePct16t24

#TotalPctDiv, FemalePctDiv, MalePctDiv -> remove Female, Male PctDiv

#PctLess9thGrade, PctNotHSGrad -> remove PctLess9thGrade

# PctFam2Par, PctKids2Par, PctYoungKids2Par, PctTeen2Par
# remove PctKids2Par, PctYoungKids2Par, PctTeen2Par
```

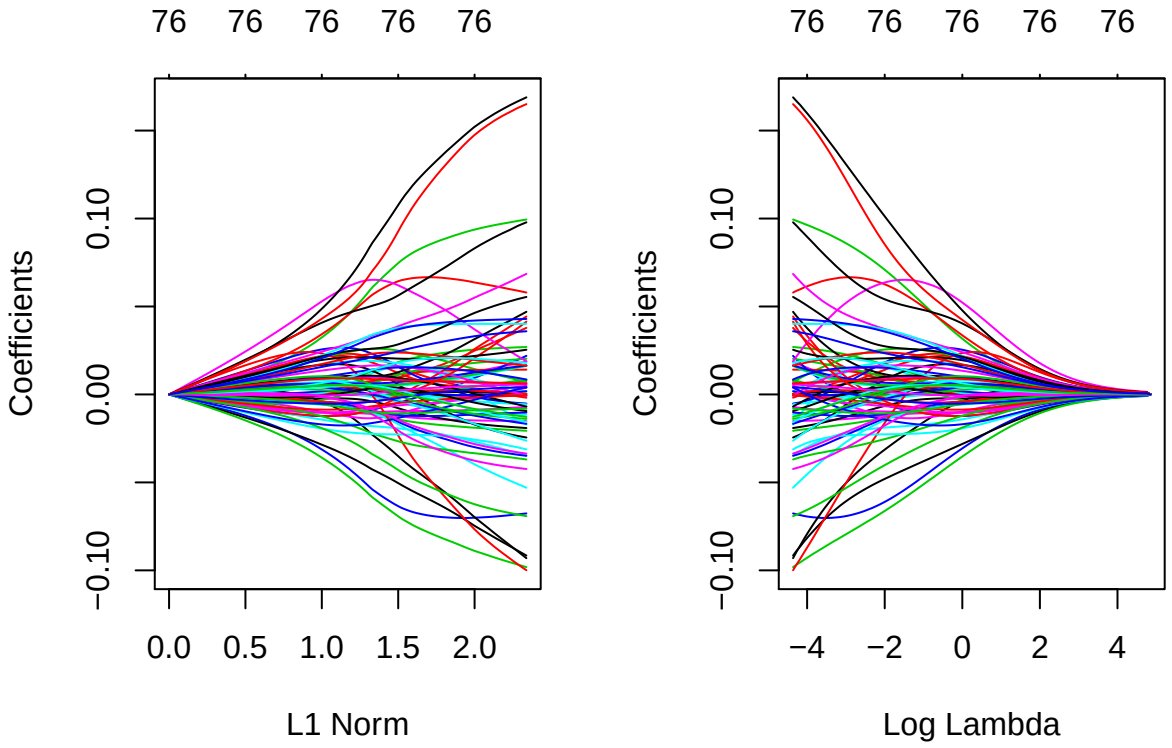
```
data <- data[,-c(9, 12, 62, 4, 73, 85, 23, 86, 11, 87, 42, 40, 31, 46,
                 47, 48, 93, 84, 86, 83, 81, 82, 30, 44, 22)]
```

b) Lasso and Ridge Regression

The lasso coefficient/lambda plots differ from the ridge coefficient/lambda plots in that the ridge plots are much smoother and simply converge towards zero in a cone shape, while the lasso plots are much more variable and converge towards zero in a much more dramatic and almost spontaneous manner.

Ridge Regression

```
ridge_model <- glmnet(as.matrix(data[,-c(2,78)]),data[,78],alpha=0, standardize=TRUE)
par(mfrow=c(1,2))
plot(ridge_model, xvar = "norm")
plot(ridge_model, xvar = "lambda")
```

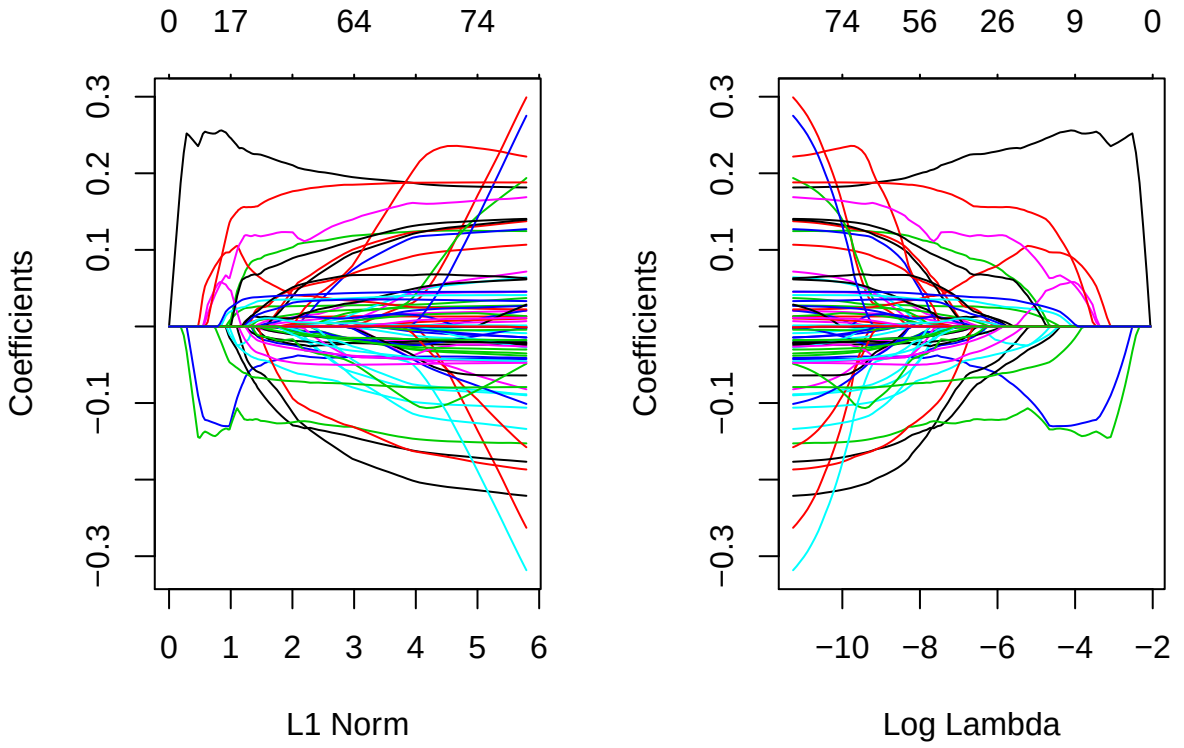



```
range(ridge_model$lambda)
```

```
[1]   0.01274856 127.48562580
```

Lasso regression

```
lasso_model <- glmnet(as.matrix(data[,-c(2,78)]),data[,78],alpha=1, standardize=TRUE)
par(mfrow=c(1,2))
plot(lasso_model, xvar = "norm")
plot(lasso_model, xvar = "lambda")
```




```
range(lasso_model$lambda)
```

```
[1] 1.274856e-05 1.274856e-01
```

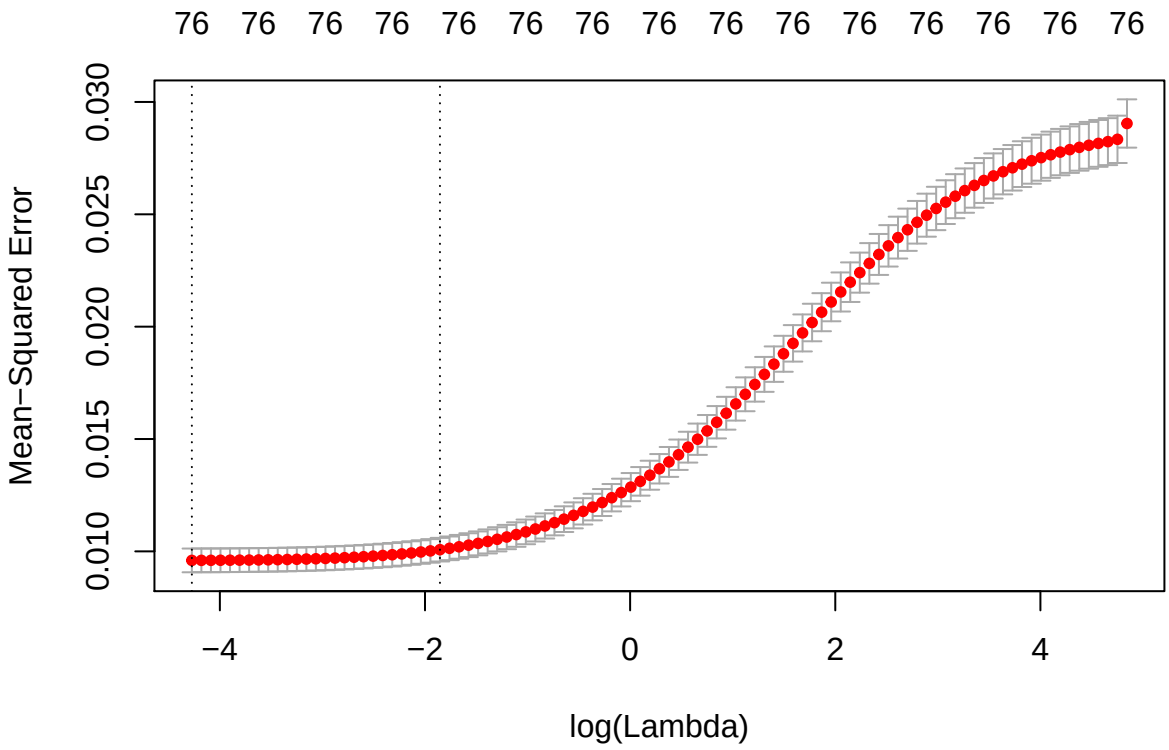
c) & d) Cross validation and lambda selection

```
# create a 90% train set 10% test set
s <- sample(1:nrow(data), nrow(data)/10, replace=FALSE)
test <- data[s,]
train <- data[-s,]
```

Train - ridge

cv.glmnet

```
set.seed(365)
ridge_train <- cv.glmnet(data.matrix(train[,-c(2,78)]), train[,78], alpha=0,
                         standardize=TRUE, foldid=train$fold)
plot(ridge_train)
```



Lambda selection

```
ridge_lmin <- ridge_train$lambda.min
ridge_lmin
```

```
[1] 0.0139399
```

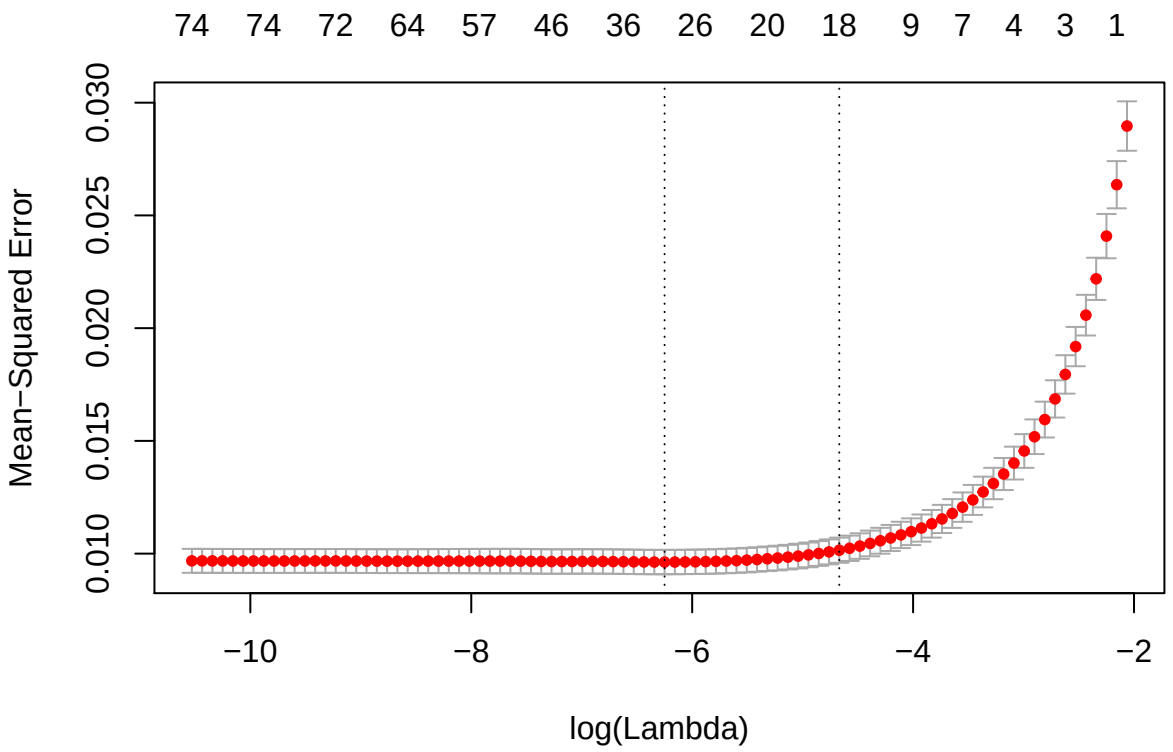
```
ridge_lse <- ridge_train$lambda.1se
ridge_lse
```

```
[1] 0.1565903
```

Train - lasso

cv.glmnet

```
lasso_train <- cv.glmnet(data.matrix(train[,-c(2,78)]), train[,78], alpha=1,
                         standardize=TRUE, foldid=train$fold)
plot(lasso_train)
```



Lambda selection

```
lasso_lmin <- lasso_train$lambda.min
lasso_lmin
```

```
[1] 0.001930518
```

```
lasso_lse <- lasso_train$lambda.1se
lasso_lse
```

```
[1] 0.00938734
```

Predictions

Create an empty dataframe to store MSE for the 4 possible results

```
mse_results <- as.data.frame(matrix(ncol=2, nrow=4))
colnames(mse_results) <- c("name", "mse")
mse_results$name <- c("ridge_lmin", "ridge_lse", "lasso_lmin", "lasso_lse")
```

Ridge

Minimum MSE lambda

```
pred_ridge_lmin <- predict(ridge_train, s=ridge_lmin, newx=data.matrix(test[,-c(2,78)]))
mse_results[1,2] <- round(mean((pred_ridge_lmin-test[,78])^2),5)
```

1 Standard Error lambda

```
pred_ridge_lse <- predict(ridge_train, s=ridge_lse, newx=data.matrix(test[,-c(2,78)]))
mse_results[2,2] <- round(mean((pred_ridge_lse-test[78])^2),4)
```

Lasso

Minimum MSE lambda

```
pred_lasso_lmin <- predict(lasso_train, s=lasso_lmin, newx=data.matrix(test[,-c(2,78)]))
mse_results[3,2] <- round(mean((pred_lasso_lmin-test[,78])^2),4)
```

1 Standard Error lambda

```
pred_lasso_lse <- predict(lasso_train, s=lasso_lse, newx=data.matrix(test[,-c(2,78)]))
mse_results[4,2] <- lasso_mse_se <- round(mean((pred_lasso_lse-test[,78])^2),4)
```

Results

```
print(mse_results)
```

```
        name     mse
1 ridge_lmin 0.00717
2  ridge_lse 0.00810
3 lasso_lmin 0.00720
4  lasso_lse 0.00760
```

e) Stepwise regression

For some reason, if I do not remove `state` from `full_model`, any models incorporating `state` fail the predict function, noting that that there is a factor level in the test set not apparent in the train set. Extracting test and train levels for the `state` variable and running the following code did not show any differences in levels.

```
testlevels <- levels(test$state)
trainlevels <- levels(train$state)

testlevels[!(testlevels %in% trainlevels)]
```

```
character(0)
```

```
full_model <- lm(ViolentCrimesPerPop ~ . - ViolentCrimesPerPop
                 - fold - state, data = train)
empty_model <- lm(ViolentCrimesPerPop ~ 1, data = train)
```

Forward

AIC criterion

```
forward_aic <- step(object = empty_model, scope=list(upper=full_model),
                    direction = "forward", trace = FALSE)
```

```
forward_aic$call
```

```
lm(formula = ViolentCrimesPerPop ~ PctIlleg + TotalPctDiv + racePctWhite +
    NumStreet + PctHousOccup + PctWorkMom + pctUrban + pctWWage +
    pctWInvInc + MedOwnCostPctIncNoMtg + PctEmploy + MedRentPctHousInc +
    PctSameCity85 + NumInShelters + racepctblack + PctPersDenseHous +
    PctFam2Par + medIncome + LemasPctOfficDrugUn + PctLargHouseFam +
    NumImmig + LandArea + indianPerCap + PctVacMore6Mos + PctVacantBoarded +
```

```
    pctWPubAsst + agePct12t29 + MalePctNevMarr + PctWOFullPlumb +
    MedOwnCostPctInc + whitePerCap + pctWRetire + HispPerCap,
    data = train)
```

BIC criterion

```
forward_bic <- step(object = empty_model, scope=list(upper=full_model),
                    direction = "forward", trace = FALSE, k = log(nrow(train)))
```

```
forward_bic$call
```

```
lm(formula = ViolentCrimesPerPop ~ PctIlleg + TotalPctDiv + racePctWhite +
    NumStreet + PctHousOccup + PctWorkMom + pctUrban + pctWWage +
    pctWInvInc + MedOwnCostPctIncNoMtg + PctEmploy, data = train)
```

Backwards

AIC criterion

```
backwards_aic <- step(object = full_model, direction = "backward", trace = FALSE)
```

```
backwards_aic$call
```

```
lm(formula = ViolentCrimesPerPop ~ population + racepctblack +
    racePctHisp + agePct12t29 + numbUrban + pctUrban + medIncome +
    pctWWage + pctWInvInc + pctWSocSec + pctWRetire + whitePerCap +
    indianPerCap + PctEmploy + PctEmplManu + PctOccupManu + PctOccupMgmtProf +
    MalePctNevMarr + PctFam2Par + PctWorkMom + PctIlleg + NumImmig +
    PctImmigRecent + PctImmigRec8 + PctRecImmig5 + PctRecImmig8 +
    PctNotSpeakEnglWell + PctLargHouseFam + PersPerOccupHous +
    PersPerRentOccHous + PctPersOwnOccup + PctPersDenseHous +
    PctHousLess3BR + PctHousOccup + PctHousOwnOcc + PctVacantBoarded +
    PctVacMore6Mos + PctWOFullPlumb + MedRentPctHousInc + MedOwnCostPctInc +
    MedOwnCostPctIncNoMtg + NumInShelters + NumStreet + LandArea +
    PctUsePubTrans + LemasPctOfficDrugUn, data = train)
```

BIC criterion

```
backwards_bic <- step(object = full_model, direction = "backward",
                      trace = FALSE, k = log(nrow(train)))
```

```
backwards_bic$call
```

```
lm(formula = ViolentCrimesPerPop ~ population + racepctblack +
    agePct12t29 + pctUrban + pctWInvInc + PctEmploy + MalePctNevMarr +
    PctFam2Par + PctWorkMom + PctIlleg + PersPerRentOccHous +
    PctPersOwnOccup + PctPersDenseHous + PctHousOccup + PctHousOwnOcc +
    MedOwnCostPctIncNoMtg + NumStreet, data = train)
```

Bidirectional

AIC criterion

```
both_aic <- step(object = empty_model, scope=list(upper=full_model),
                 direction = "both", trace = FALSE)
```

```
both_aic$call
```

```
lm(formula = ViolentCrimesPerPop ~ PctIlleg + NumStreet + PctHousOccup +
    PctWorkMom + pctUrban + pctWWage + pctWInvInc + MedOwnCostPctIncNoMtg +
    PctEmploy + MedRentPctHousInc + NumInShelters + racepctblack +
    PctPersDenseHous + PctFam2Par + medIncome + LemasPctOfficDrugUn +
    PctLargHouseFam + NumImmig + LandArea + indianPerCap + PctWOFullPlumb +
    agePct12t29 + MalePctNevMarr + MedOwnCostPctInc + PctVacMore6Mos +
    PctVacantBoarded, data = train)
```

BIC criterion

```
both_bic <- step(object = empty_model, scope=list(upper=full_model),
                 direction = "both", trace = FALSE, k = log(nrow(train)))
```

```
both_bic$call
```

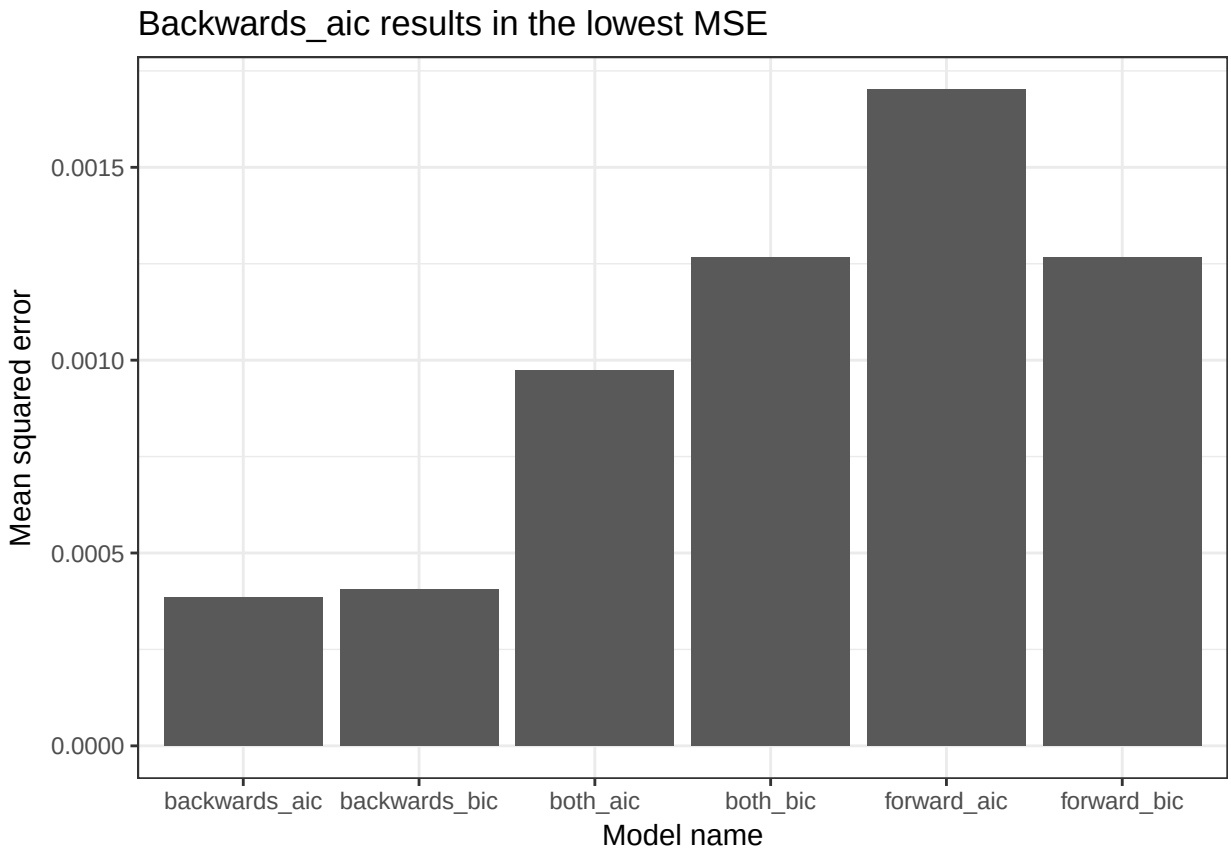
```
lm(formula = ViolentCrimesPerPop ~ PctIlleg + TotalPctDiv + racePctWhite +
    NumStreet + PctHousOccup + PctWorkMom + pctUrban + pctWWage +
    pctWInvInc + MedOwnCostPctIncNoMtg + PctEmploy, data = train)
```

Stepwise regression performance

```
# create an empty dataframe to store MSE
step_mse <- as.data.frame(matrix(nrow=6, ncol=2))
colnames(step_mse) <- c("name", "mse")

# store names of models into dataframe
models_list <- c("forward_aic", "forward_bic",
                 "backwards_aic", "backwards_bic",
                 "both_aic", "both_bic")
step_mse$name <- models_list

# create function that extracts and calculates MSE
mse_extract <- function(m) {
  pred <- (predict(m, newdata=test[,-c(2,78)]))
  return((sum(pred - test$ViolentCrimesPerPop)^2)/length(pred))
}

# Run function, storing result in appropriate row of dataframe
step_mse[1,2] <- mse_extract(forward_aic)
step_mse[2,2] <- mse_extract(forward_bic)
step_mse[3,2] <- mse_extract(backwards_aic)
step_mse[4,2] <- mse_extract(backwards_bic)
```

```r
step_mse[5,2] <- mse_extract(both_aic)
step_mse[6,2] <- mse_extract(both_bic)
```

```r
# Create plot showing mse for each stepwise model
mse_plot <- ggplot(data=step_mse, aes(x=name, y=mse))+
  geom_bar(stat="identity")+
  theme_bw()+
  xlab("Model name")+
  ylab("Mean squared error")+
  ggtitle("Backwards_aic results in the lowest MSE")

print(mse_plot)
```



f) Lasso/ridge/stepwise comparison

Non-zero lasso coefficients

The below code is not run to save space, but the 1 standard error lambda results in the following model coefficients:

`state`, `racePctWhite`, `racePctblack` `PctWWage` `pctUrban`, `pctWInvInc`, `TotalPctDiv`, `PctFam2Par`, `PctWorkMom`, `NumIlleg`, `PctIlleg`, `PctPersDenseHous`, `PctHousOccup`, `PctVacantBoarded`, `NumStreet`, `LemasPctOfficDrugUn`

```r
predict(lasso_train, s=lasso_lse, type="coefficients")
```

The minimum MSE lambda results in a model with 30 predictors.

```r
predict(lasso_train, s=lasso_lmin, type="coefficients")
```

```r
# create function that extracts the coefficients of an input model
# and saves the coefficients + the model name into a 2 column dataframe
get_predictors <- function(model, modelname) {
  dataframe <- as.data.frame(model$coefficients)
  dataframe <- rownames_to_column(dataframe, var="rowname")
  dataframe$name <- modelname
  return(dataframe[,-2])
}

# create seperate dataframes with names and coefficients for each
# of the six stepwise models
f_aic_pred <- get_predictors(forward_aic, "forward_aic")
f_bic_pred <- get_predictors(forward_bic, "forward_bic")
ba_aic_pred <- get_predictors(backwards_aic, "backwards_aic")
ba_bic_pred <- get_predictors(backwards_bic, "backwards_bic")
bo_aic_pred <- get_predictors(both_aic, "both_aic")
bo_bic_pred <- get_predictors(both_aic, "both_bic")

# bind all the dataframes just created into one
stepwise_coefficients <- rbind(f_aic_pred, f_bic_pred,
                               ba_aic_pred, ba_bic_pred,
                               bo_aic_pred, bo_bic_pred)

# create dataframe containing coefficient along with the
# number of models (out of 6) that it appeared in
stepwise_counts <- stepwise_coefficients %>%
  group_by(rowname) %>%
  tally() %>%
  arrange(desc(n))
```

Predictors in lasso and stepwise models

The following predictors showed up as coefficients in at least 5 of the 6 stepwise models.

Of these, `racepctblack`, `pcturban`, `pctwinvinc`, `PctFam2Par`, `PctWorkMom`, `PctIlleg`, `PctHousOccup`, `NumSt`, and `LemasPctOfficDrugUn` also appeared in the 1SE lasso model.

```r
stepwise_counts$rowname[stepwise_counts$n >= 5]
```

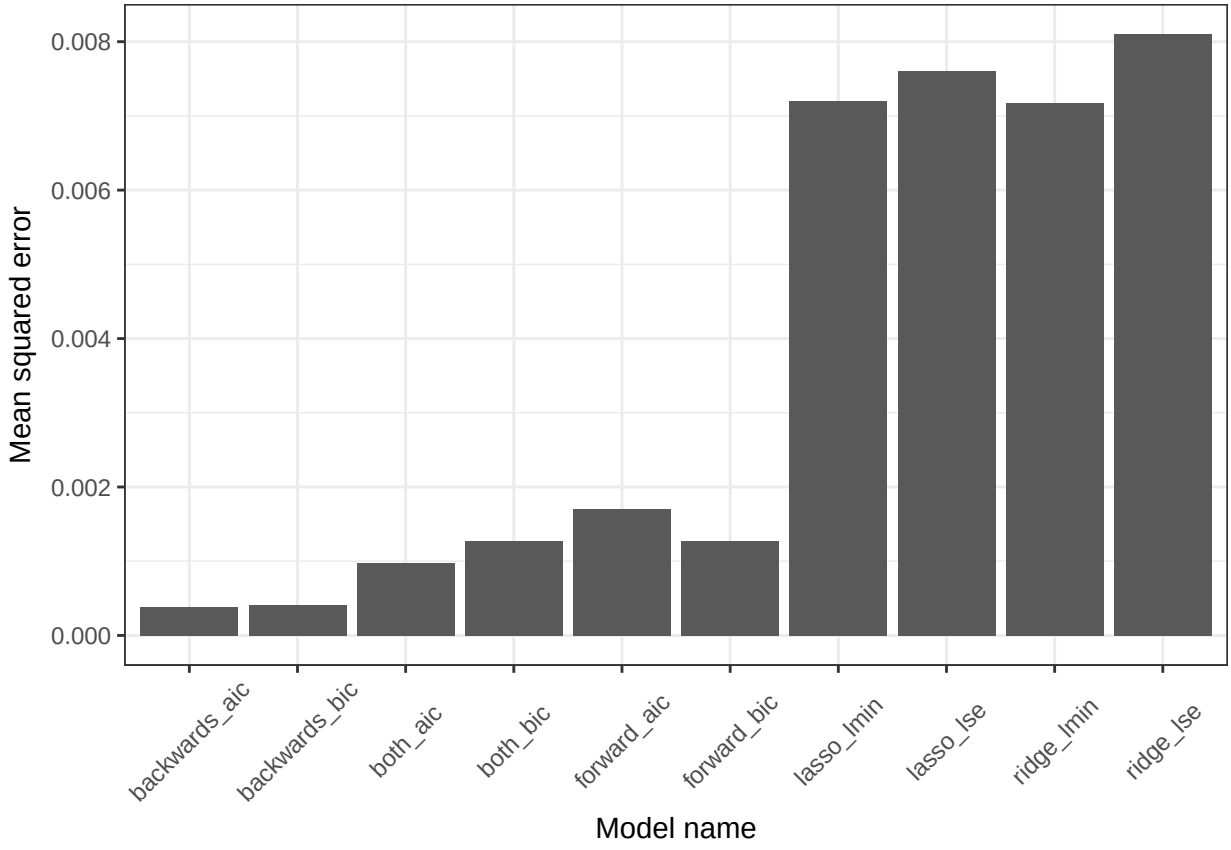
```
 [1] "(Intercept)"           "MedOwnCostPctIncNoMtg"
 [3] "NumStreet"             "PctEmploy"
 [5] "PctHousOccup"          "PctIlleg"
 [7] "pctUrban"              "pctWInvInc"
 [9] "PctWorkMom"            "agePct12t29"
[11] "MalePctNevMarr"        "PctFam2Par"
[13] "PctPersDenseHous"      "pctWWage"
[15] "racepctblack"
```

Comparing the performance of all lasso, ridge, and stepwise models

Lasso, ridge MSE

```
total_mse <- rbind(step_mse, mse_results)
```

```
all_mse_plot <- ggplot(data=total_mse, aes(x=name, y=mse))+
  geom_bar(stat="identity")+
  theme_bw()+
  xlab("Model name")+
  ylab("Mean squared error")+
  theme(axis.text.x = element_text(angle=45, vjust=0.5))

print(all_mse_plot)
```



G) bootstrapping

backwards_aic was my best stepwise model selected in (e)

```
set.seed(365)
N <- 10000
rsquared <- rep(NA, N)
for (i in 1:N) {
  s <- sample(1:nrow(data), nrow(data), replace = TRUE)
  bootsample <- data[s,]
  mod <- lm(formula = ViolentCrimesPerPop ~ population + racepctblack +
    racePctHisp + agePct12t29 + numbUrban + pctUrban + medIncome +
    pctWInvInc + pctWSocSec + pctWRetire + whitePerCap + indianPerCap +
    HispPerCap + PctEmploy + PctEmplManu + PctOccupManu + PctOccupMgmtProf +
```

```
    MalePctNevMarr + PctFam2Par + PctWorkMom + PctIlleg + NumImmig +
    PctRecImmig5 + PctRecImmig8 + PctNotSpeakEnglWell + PctLargHouseFam +
    PersPerOccupHous + PersPerRentOccHous + PctPersOwnOccup +
    PctPersDenseHous + PctHousOccup + PctHousOwnOcc + PctVacantBoarded +
    PctVacMore6Mos + MedRentPctHousInc + MedOwnCostPctIncNoMtg +
    NumInShelters + NumStreet + PctUsePubTrans + LemasPctOfficDrugUn,
    data = bootsample)
  rsquared[i] <- summary(mod)$r.squared
}
```

Confidence interval

99% confidence interval

```
round(quantile(rsquared, c(0.005, 0.995)),3)
```

```
 0.5% 99.5%
0.667 0.735
```