Data Analysis

Ariane Stark

4/28/2021

```
dat <- read_csv("case_and_demographics.csv") %>%
  subset(select = -c(1)) %>%
  subset(select = c(1,3:4,2,5:6,43,56:61 )) %>%
  mutate(Cum_Case_Median = cut_number(cum_case,2, labels= c(0,1))) %>%
  na.omit() #removes Rio Arriba County NM which has NA in the pov. dem.
```

Cross Validation of Best Subset, Forward and Backward Stepwise

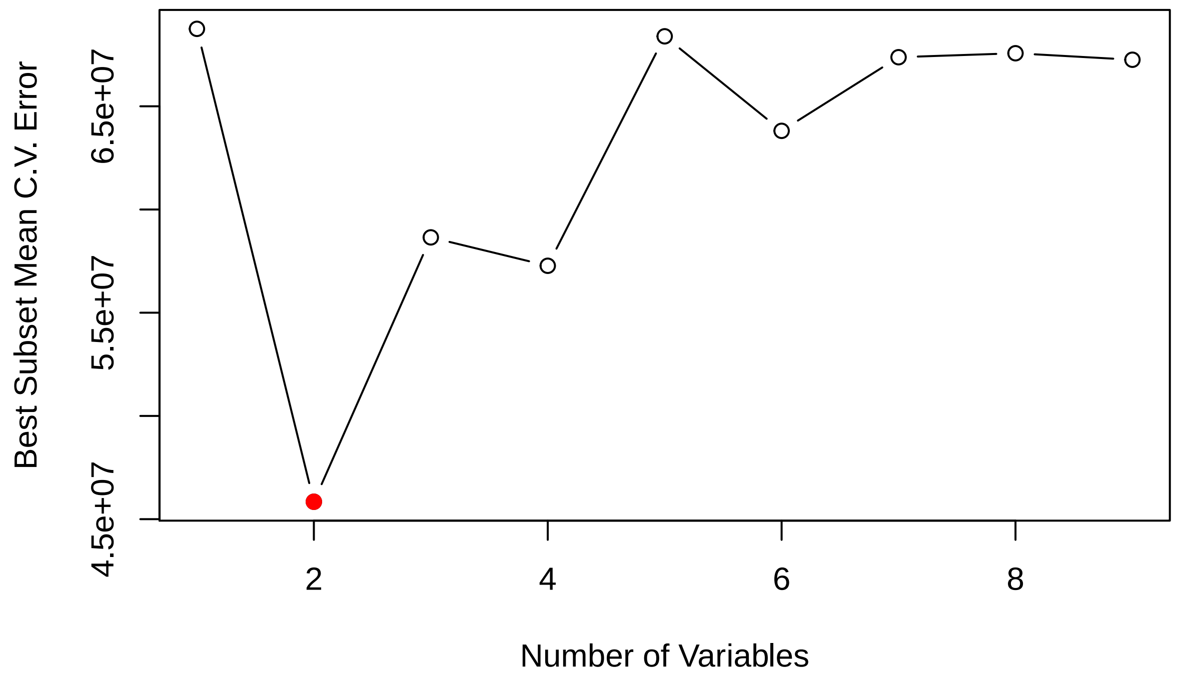
```
predict.regsubsets <- function (object ,newdata ,id ,...){
  form<-as.formula(object$call [[2]])
  mat<-model.matrix(form,newdata)
  coefi<-coef(object ,id=id)
  xvars<-names(coefi)
  mat[,xvars]%*%coefi
}
```

```
set.seed(100)

k<-10

folds<-sample(1:k,nrow(dat[,c(4:13)]),replace=TRUE)
cv.errors.best_subset <- matrix(NA,k,9, dimnames=list(NULL, paste(1:9)))
cv.errors.forward <- matrix(NA,k,9, dimnames=list(NULL, paste(1:9)))
cv.errors.backward <- matrix(NA,k,9, dimnames=list(NULL, paste(1:9)))


for(j in 1:k){
  # best subset select
  best_subset.fit <- regsubsets(cum_case~.,data = dat[folds!=j,c(4:13)],
                               method = "exhaustive", nvmax = 9)
  # forward stepwise select
  forward.fit <- regsubsets(cum_case~.,data = dat[folds!=j,c(4:13)],
                               method = "forward", nvmax = 9)
  # backward stepwise select
  backward.fit <- regsubsets(cum_case~.,data = dat[folds!=j,c(4:13)],
                               method = "backward", nvmax = 9)

  for(i in 1:9){
    # best subset error
    pred.best <-
      predict.regsubsets(best_subset.fit, dat[folds==j,c(4:13)], id=i)
    cv.errors.best_subset[j,i] <-
      mean((dat$cum_case[folds==j]-pred.best)^2)

    # forward stepwise error
    pred.forward <-
      predict.regsubsets(forward.fit, dat[folds==j,c(4:13)], id=i)
    cv.errors.forward[j,i] <-
      mean((dat$cum_case[folds==j]-pred.forward)^2)

    # backward stepwise error
    pred.backward <-
      predict.regsubsets(backward.fit, dat[folds==j,c(4:13)], id=i)
    cv.errors.backward[j,i] <-
      mean((dat$cum_case[folds==j]-pred.backward)^2)


  } }
```

```
#best subset CV model selection
mean.cv.errors.best <- apply(cv.errors.best_subset,2,mean)
mean.cv.errors.best
```

```
##        1        2        3        4        5        6        7        8
## 68750976 45843477 58650271 57273927 68389310 63810273 67377272 67568839
##        9
## 67254330
```

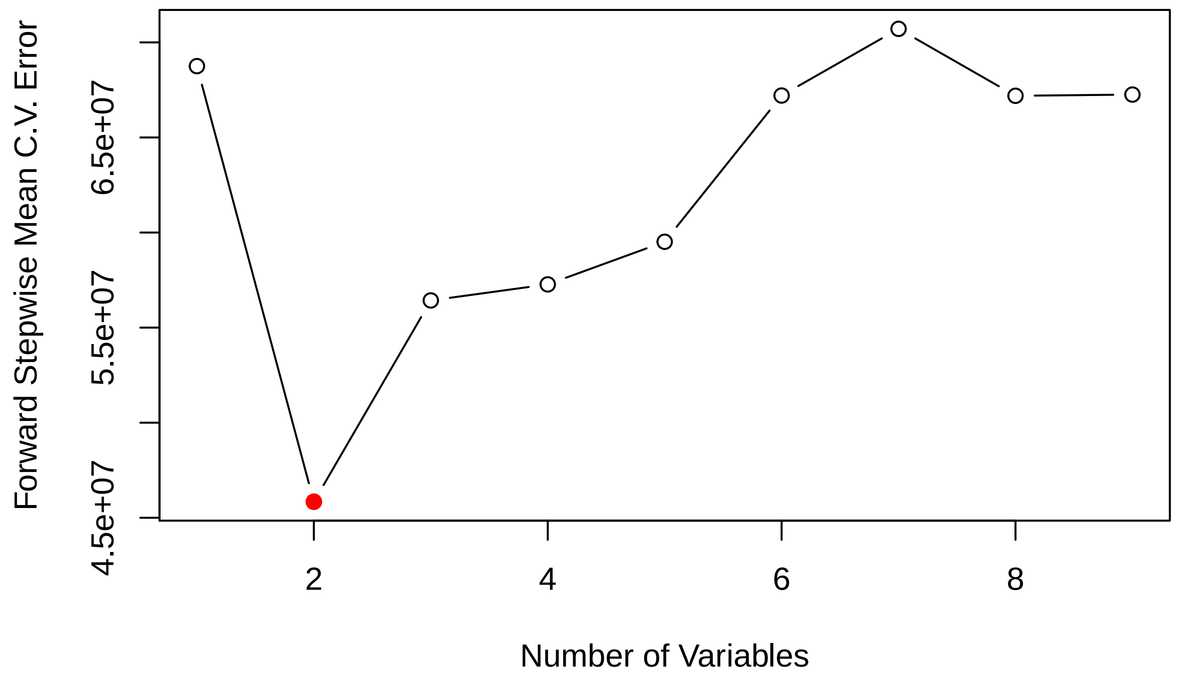
```
plot(mean.cv.errors.best,type="b",
     xlab="Number of Variables",
     ylab="Best Subset Mean C.V. Error")
points(which.min(mean.cv.errors.best),
       mean.cv.errors.best[which.min(mean.cv.errors.best)],
       col="red", cex=1.5,pch=20)
```



```
#forward stepwise CV model selection
mean.cv.errors.forward <- apply(cv.errors.forward,2,mean)
mean.cv.errors.forward
```

```
##        1        2        3        4        5        6        7        8
## 68750976 45843477 56429254 57273927 59513523 67205055 70711271 67190749
##        9
## 67254330
```

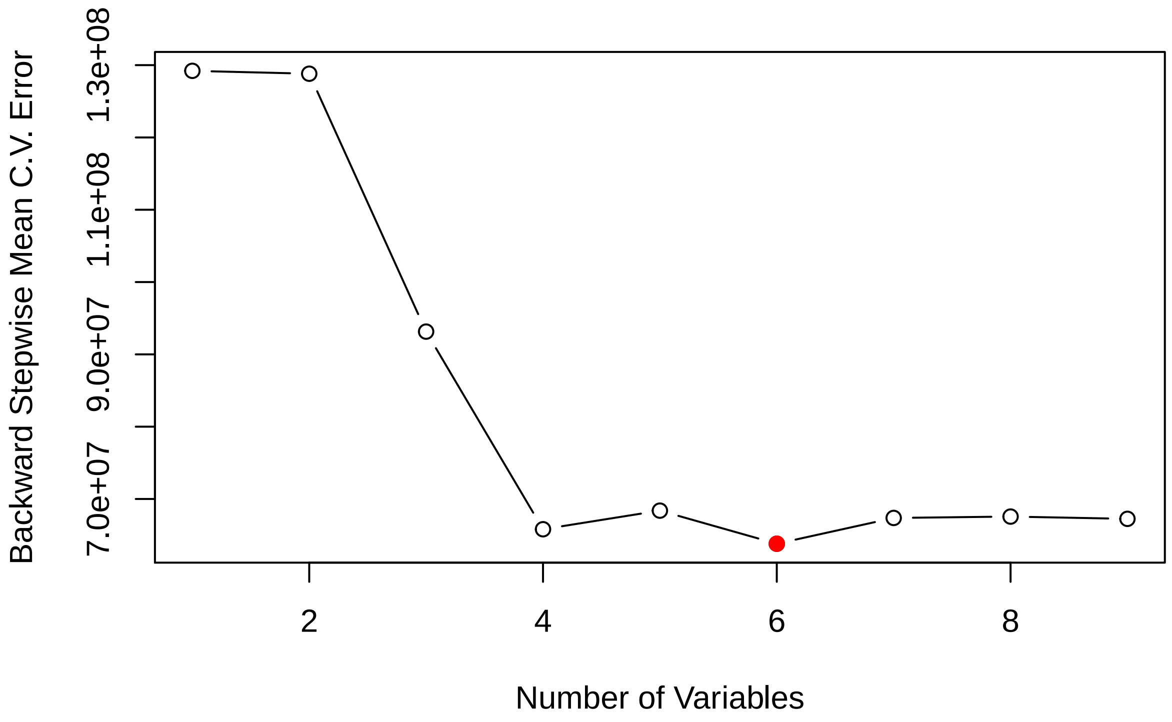
```
plot(mean.cv.errors.forward,type="b",
     xlab="Number of Variables",
     ylab="Forward Stepwise Mean C.V. Error")
points(which.min(mean.cv.errors.forward),
       mean.cv.errors.forward[which.min(mean.cv.errors.forward)],
       col="red", cex=1.5,pch=20)
```



```r
#backward stepwise CV model selection
mean.cv.errors.backward <- apply(cv.errors.backward,2,mean)
mean.cv.errors.backward
```

```
##         1         2         3         4         5         6         7         8
## 129210484 128813623  93146095  65815873  68389310  63810273  67377272  67568839
##         9
##  67254330
```

```r
plot(mean.cv.errors.backward,type="b",
     xlab="Number of Variables",
     ylab="Backward Stepwise Mean C.V. Error")
points(which.min(mean.cv.errors.backward),
       mean.cv.errors.backward[which.min(mean.cv.errors.backward)],
       col="red", cex=1.5,pch=20)
```



Best Subset Overall

```
best_subset.fit <- regsubsets(cum_case~.,data = dat[,c(4:13)],
                              method = "exhaustive", nvmax = 9)
best_subset.summary <- summary(best_subset.fit)
best_subset.summary.frame <- data_frame("Parameters" = seq(1:9),
                                        "R^2"=best_subset.summary$rsq,
                                      "AdjR^2"=best_subset.summary$adjr2,
                                      "CP"=best_subset.summary$cp,
                                      "BIC"=best_subset.summary$bic)
```

```
## Warning: `data_frame()` was deprecated in tibble 1.1.0.
## Please use `tibble()` instead.
```

```
best_subset.summary.frame
```

```
## # A tibble: 9 x 5
##   Parameters `R^2` `AdjR^2`      CP     BIC
##        <int> <dbl>    <dbl>   <dbl>   <dbl>
## 1          1 0.945    0.945 2488.    -9330.
## 2          2 0.964    0.964  546.   -10664.
## 3          3 0.966    0.966  306.   -10870.
## 4          4 0.967    0.967  178.   -10983.
## 5          5 0.968    0.968   89.5  -11063.
## 6          6 0.969    0.969    5.65 -11139.
## 7          7 0.969    0.969    6.14 -11133.
## 8          8 0.969    0.969    8.07 -11125.
## 9          9 0.969    0.969   10.0  -11117.
```

Based on Cross Validation we pick the model with 2 parameters

```
coef(best_subset.fit,2)
```

```
##    (Intercept)      Total_Pop Total_Hispanic
##   315.11931521     0.07300349     0.08304726
```

Forward Stepwise Overall

```
forward.fit <- regsubsets(cum_case~.,data = dat[,c(4:13)],
                              method = "forward", nvmax = 9)
forward.summary <- summary(forward.fit)
forward.summary.frame <- data_frame("Parameters" = seq(1:9),
                                        "R^2"=forward.summary$rsq,
                                      "AdjR^2"=forward.summary$adjr2,
                                      "CP"=forward.summary$cp,
                                      "BIC"=forward.summary$bic)
forward.summary.frame
```

```
## # A tibble: 9 x 5
##   Parameters `R^2` `AdjR^2`      CP     BIC
##        <int> <dbl>    <dbl>   <dbl>   <dbl>
## 1          1 0.945    0.945 2488.    -9330.
## 2          2 0.964    0.964  546.   -10664.
## 3          3 0.966    0.966  306.   -10870.
## 4          4 0.967    0.967  178.   -10983.
## 5          5 0.967    0.967  176.   -10979.
## 6          6 0.969    0.969    5.65 -11139.
## 7          7 0.969    0.969    6.14 -11133.
## 8          8 0.969    0.969    8.07 -11125.
## 9          9 0.969    0.969   10.0  -11117.
```

Based on Cross Validation we pick the model with 2 parameters

```
coef(forward.fit,2)
```

```
##    (Intercept)      Total_Pop Total_Hispanic
##   315.11931521     0.07300349     0.08304726
```

Backward Stepwise Overall

```r
backward.fit <- regsubsets(cum_case~.,data = dat[,c(4:13)],
                              method = "backward", nvmax = 9)
backward.summary <- summary(backward.fit)
backward.summary.frame <- data_frame("Parameters" = seq(1:9),
                                        "R^2"=backward.summary$rsq,
                                      "AdjR^2"=backward.summary$adjr2,
                                      "CP"=backward.summary$cp,
                                      "BIC"=backward.summary$bic)
backward.summary.frame
```

```
## # A tibble: 9 x 5
##   Parameters `R^2` `AdjR^2`      CP     BIC
##        <int> <dbl>    <dbl>   <dbl>   <dbl>
## 1          1 0.903    0.903 6923.    -7478.
## 2          2 0.956    0.956 1375.   -10022.
## 3          3 0.965    0.965  452.   -10739.
## 4          4 0.967    0.967  271.   -10896.
## 5          5 0.968    0.968   89.5  -11063.
## 6          6 0.969    0.969    5.65 -11139.
## 7          7 0.969    0.969    6.14 -11133.
## 8          8 0.969    0.969    8.07 -11125.
## 9          9 0.969    0.969   10.0  -11117.
```

Based on Cross Validation we pick the model with 6 parameters

```r
coef(best_subset.fit,6)
```

```
##                    (Intercept)                      Total_Pop
##                  -338.60412493                    -0.38868473
##                    Total_White                    Total_Black
##                     0.58210948                     0.55858828
##                 Total_Hispanic                    Total_Other
##                     0.05514001                     0.51284419
## Total_Households_Above_Poverty
##                    -0.29137162
```

Ridge and Lasso

```r
set.seed(100)
k <- 10
cv.error.lasso <- rep(NA,k)
cv.error.ridge <- rep(NA,k)
cv.lam.lasso <- rep(NA,k)
cv.lam.ridge <- rep(NA,k)
set.seed(1)
folds<-sample(1:k,nrow(dat[,c(4:13)]),replace=TRUE)
for (i in 1:k){
  train <- dat[folds!=i,c(4:13)] #Set the training set
  test <- dat[folds==i,c(4:13)] #Set testing set
  x.train <- model.matrix(cum_case~., data=train)[,-1]
  y.train <- train$cum_case
  x.test <- model.matrix(cum_case~., data=test)[,-1]
  y.test <- test$cum_case

  #Lasso
  lasso.mod <- glmnet(x.train,y.train,alpha=1)
  cv.out <- cv.glmnet(x.train,y.train,alpha=1)
  bestlam <- cv.out$lambda.min
  cv.lam.lasso[i] <- bestlam
  lasso.pred <- predict(lasso.mod,s=bestlam ,newx=x.test)
  cv.error.lasso[i] <- mean((lasso.pred-y.test)^2)


  #Ridge
  ridge.mod <- glmnet(x.train,y.train,alpha=0)
  cv.out <- cv.glmnet(x.train,y.train,alpha=0)
  bestlam <- cv.out$lambda.min
  cv.lam.ridge[i] <- bestlam
  ridge.pred <- predict(ridge.mod,s=bestlam ,newx=x.test)
  cv.error.ridge[i] <- mean((ridge.pred-y.test)^2)

}
```

```
# Lasso CV Errors and Mean
cv.error.lasso[which.min(cv.error.lasso)]
```

```
## [1] 21711998
```

```
mean(cv.error.lasso)
```

```
## [1] 64144413
```

```
# Ridge CV Errors and Mean
cv.error.ridge[which.min(cv.error.ridge)]
```

```
## [1] 24901393
```

```
mean(cv.error.ridge)
```

```
## [1] 58650401
```

Ridge Regression performs better when the response is a function of many predictors of roughly equal size

Ridge With Best CV Error Lambda

```r
x<-model.matrix(cum_case~., data=dat[,c(4:13)])[,-1]
y<-dat$cum_case

ridge.fit<-glmnet(x,y,alpha=0)
ridge.coef<-predict(ridge.fit,type="coefficients",
                    s=cv.lam.ridge[which.min(cv.error.ridge)])[1:9,]
ridge.coef[ridge.coef!=0]
```

```
##                     (Intercept)                     POP_DENSITY
##                   2659.26735391                     -1.46826014
##                       Total_Pop                      Median_Age
##                      0.01675252                    -60.01296350
##                     Total_White                     Total_Black
##                      0.03389108                      0.04046944
##                  Total_Hispanic                     Total_Other
##                      0.06843143                      0.03043294
## Total_Households_Below_Poverty
##                      0.27874317
```

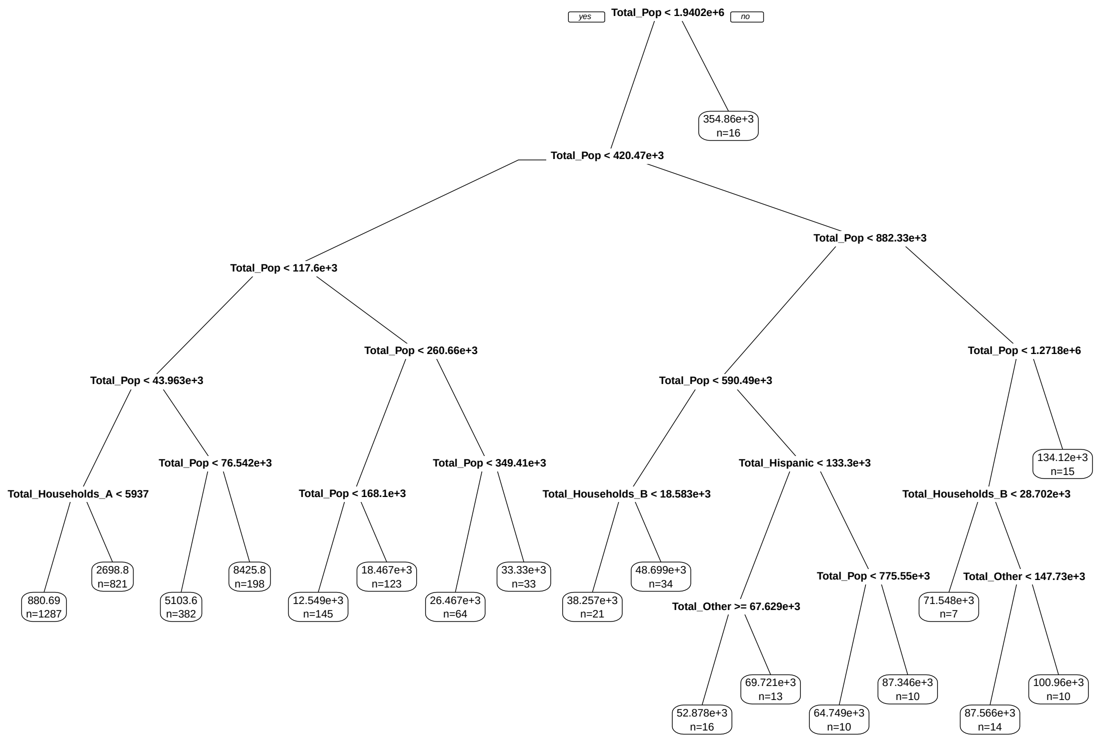
Lasso With Best CV Error Lambda

```r
lasso.fit<-glmnet(x,y,alpha=1)
lasso.coef<-predict(lasso.fit,type="coefficients",
                    s=cv.lam.lasso[which.min(cv.error.lasso)])[1:9,]
lasso.coef[lasso.coef!=0]
```

```
##                     (Intercept)                       Total_Pop
##                    286.95400883                      0.04444873
##                     Total_White                     Total_Black
##                      0.02943025                      0.01685243
##                  Total_Hispanic Total_Households_Below_Poverty
##                      0.08547591                      0.09854572
```

Regression Trees

```r
tree_dat <- dat %>%
  subset(select = c(4:13))

tree <- rpart(cum_case ~.,
              data=tree_dat,control=rpart.control(cp=.0001))

#printcp(tree)

best <- tree$cptable[which.min(tree$cptable[,"xerror"]),"CP"]
pruned_tree <- prune(tree, cp=best)
prp(pruned_tree,
    faclen=0, #use full names for factor labels
    extra=1, #display number of obs. for each terminal node
    roundint=F, #don't round to integers in output
    digits=5) #display 5 decimal places in output
```


yes
Total_Pop < 1.9402e+6
no
354.86e+3 n=16
Total_Pop < 420.47e+3
Total_Pop < 117.6e+3
Total_Pop < 882.33e+3
Total_Pop < 43.963e+3
Total_Pop < 260.66e+3
Total_Pop < 590.49e+3
Total_Pop < 1.2718e+6
Total_Households_A < 5937
Total_Pop < 76.542e+3
Total_Pop < 168.1e+3
Total_Pop < 349.41e+3
Total_Households_B < 18.583e+3
Total_Hispanic < 133.3e+3
134.12e+3 n=15
Total_Households_B < 28.702e+3
880.69 n=1287
2698.8 n=821
5103.6 n=382
8425.8 n=198
12.549e+3 n=145
18.467e+3 n=123
26.467e+3 n=64
33.33e+3 n=33
38.257e+3 n=21
48.699e+3 n=34
Total_Other >= 67.629e+3
Total_Pop < 775.55e+3
71.548e+3 n=7
Total_Other < 147.73e+3
52.878e+3 n=16
69.721e+3 n=13
64.749e+3 n=10
87.346e+3 n=10
87.566e+3 n=14
100.96e+3 n=10