Homework 3

STAT 471

Jacob Kahn
Devesh Dayal
Meghana Jayam

30 October, 2016

Preflight Tasks

We have a lot of existing variables, so we'll clean them out.

```
rm(list=ls())
```

Check our working directory. This changed for various group members, so we each set it locally:

```
# setwd(dir)
# getwd() # check that working directory
```

Problem 1

Part a

Generate a predictor X of length n=100, as well as a noise vector epsilon of length n=100.

```
set.seed(10)
X <- rnorm(100)
epsilon <- rnorm(100)
```

Part b

Generate a response vector Y of length n=100 with B0, B1, B2, B3

```
B0 <- -2
B1 <- 0.1
B2 <- 1
B3 <- 5
Y <- B0+B1*X+B2*X^2+B3*X^3+epsilon
```

Part c

Perform best subset selection (find Cp, BIC, adjr2)

```r
library(leaps) # for regsubsets
data <- data.frame(y=Y, x=X)
regsub <- regsubsets(y~poly(x,degree=10,raw=TRUE), data=data, nvmax=10)
reg.sum <- summary(regsub)
names(reg.sum)
```

```
## [1] "which"  "rsq"    "rss"    "adjr2"  "cp"     "bic"    "outmat" "obj"
```

```r
# find optimal size by getting min Cp, BIC, adjr2
which.min(reg.sum$cp)       # 2
```

```
## [1] 2
```

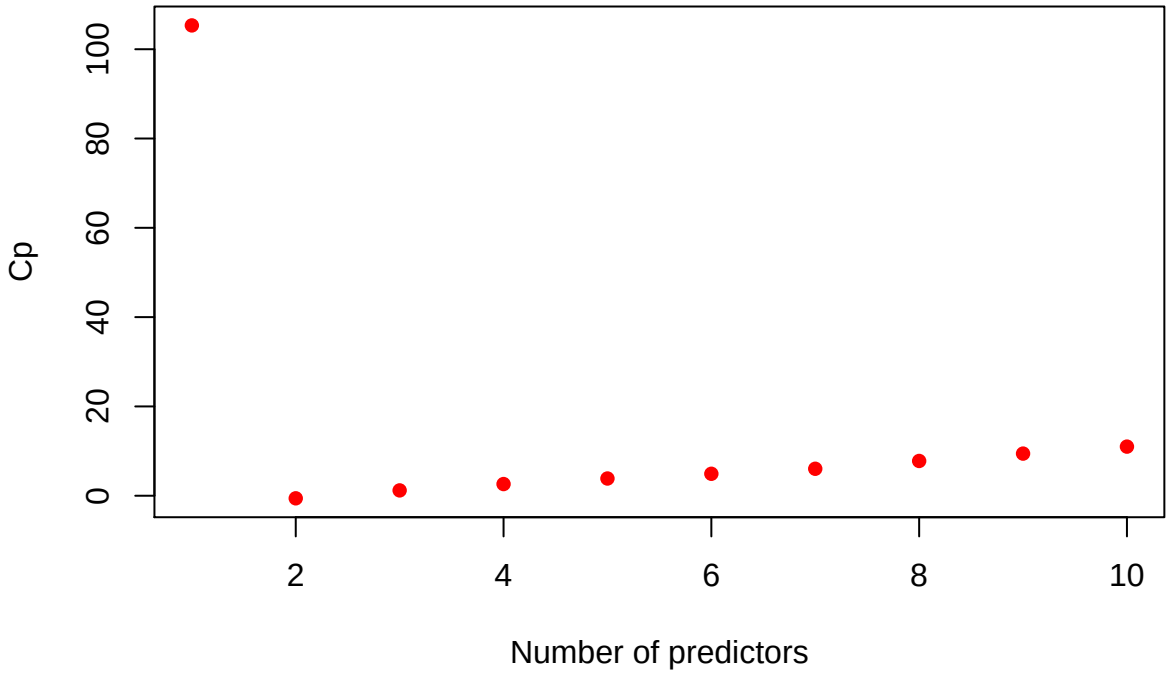
```r
which.min(reg.sum$bic)      # 2
```

```
## [1] 2
```

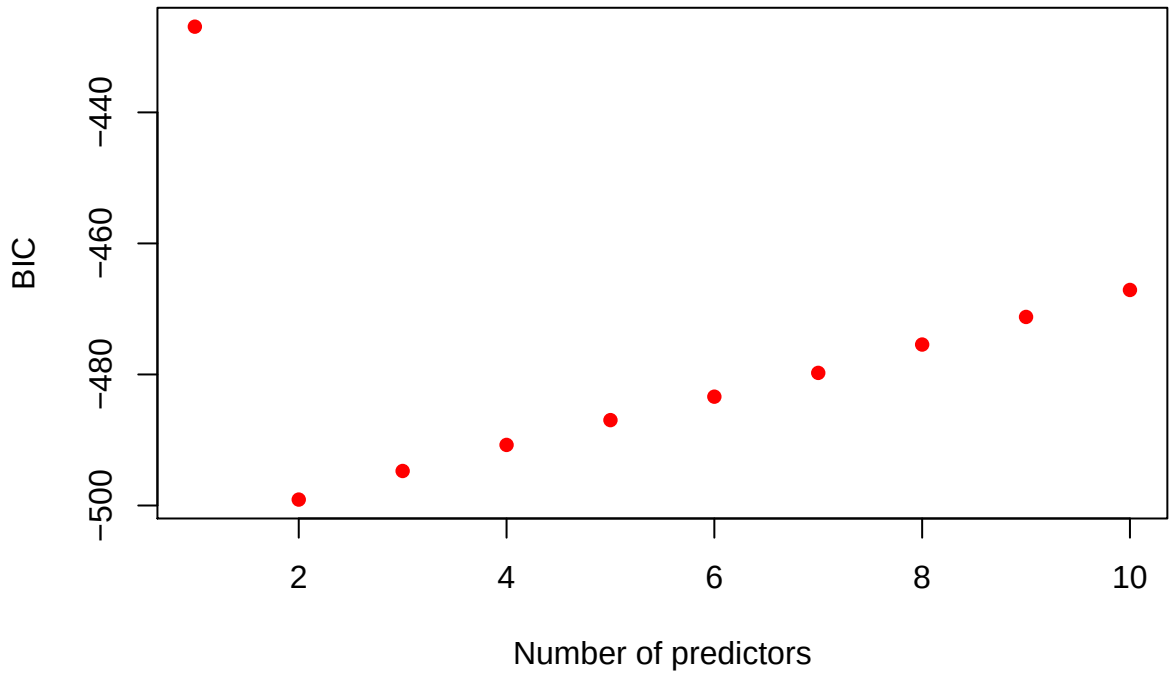
```r
which.max(reg.sum$adjr2)    # 2
```

```
## [1] 2
```

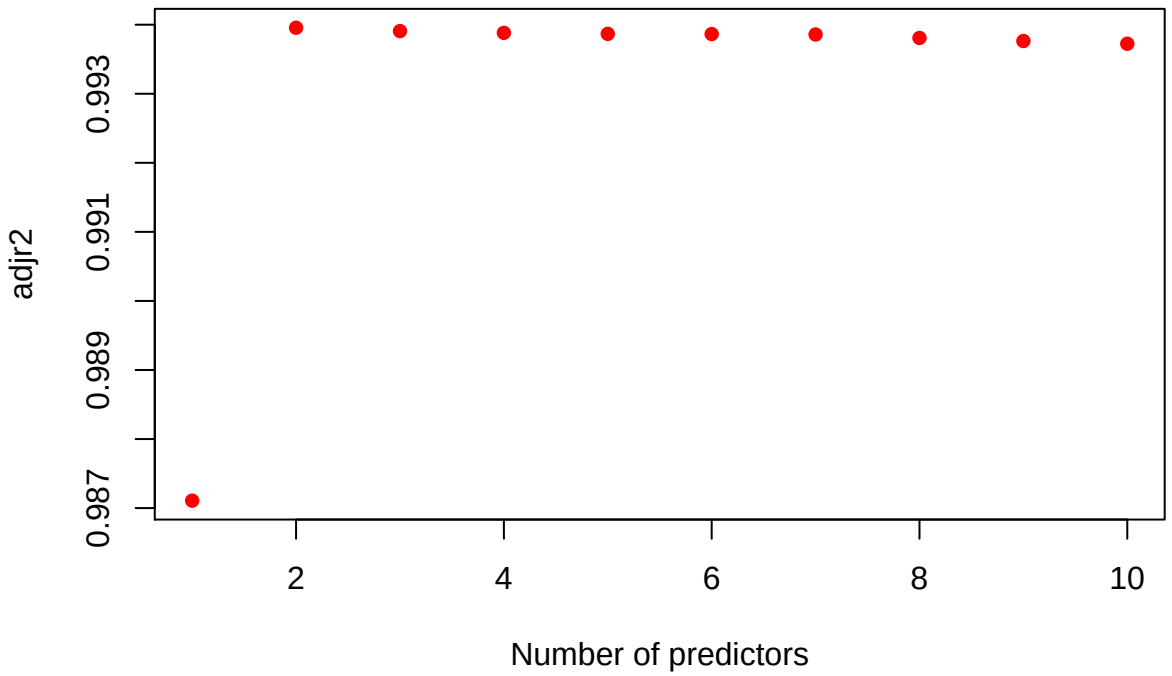
```r
# plot Cp, BIC, adjr2
plot(reg.sum$cp, xlab="Number of predictors", ylab="Cp", col="red", type="p", pch=16) # exhibit 1
```




```r
plot(reg.sum$bic, xlab="Number of predictors", ylab="BIC", col="red", type="p", pch=16) # exhibit 2
```




```
plot(reg.sum$adjr2, xlab="Number of predictors", ylab="adjr2", col="red", type="p", pch=16) # exhibit 3
```



As seen by the plots, we would use a 2-variable model with Cp, a 2-variable model with BIC, and a 2-variable model with Adjusted R^2.

```
coef(regsub, id=2)
```

```
##                       (Intercept) poly(x, degree = 10, raw = TRUE)2
##                        -2.0710776                         0.9642645
## poly(x, degree = 10, raw = TRUE)3
##                         5.0163042
```

The model will use x^2 and x^3.

Part d

Use forward selection and the backward selection to compare esults with part c. First, we will do forward selection:

```
regsub.f <- regsubsets(y~poly(x,degree=10,raw=TRUE), data=data, nvmax=10, method="forward")
reg.sum.f <- summary(regsub.f)

which.min(reg.sum.f$cp)       # 2
```

```
## [1] 2
```

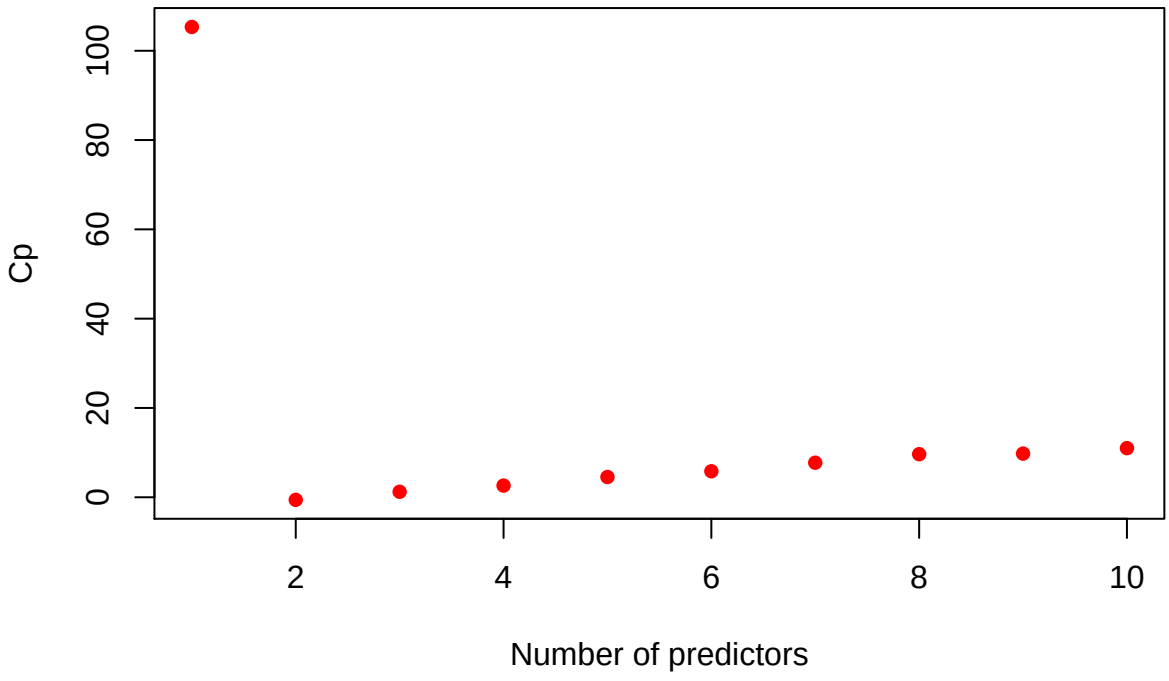
```
which.min(reg.sum.f$bic)      # 2
```

```
## [1] 2
```

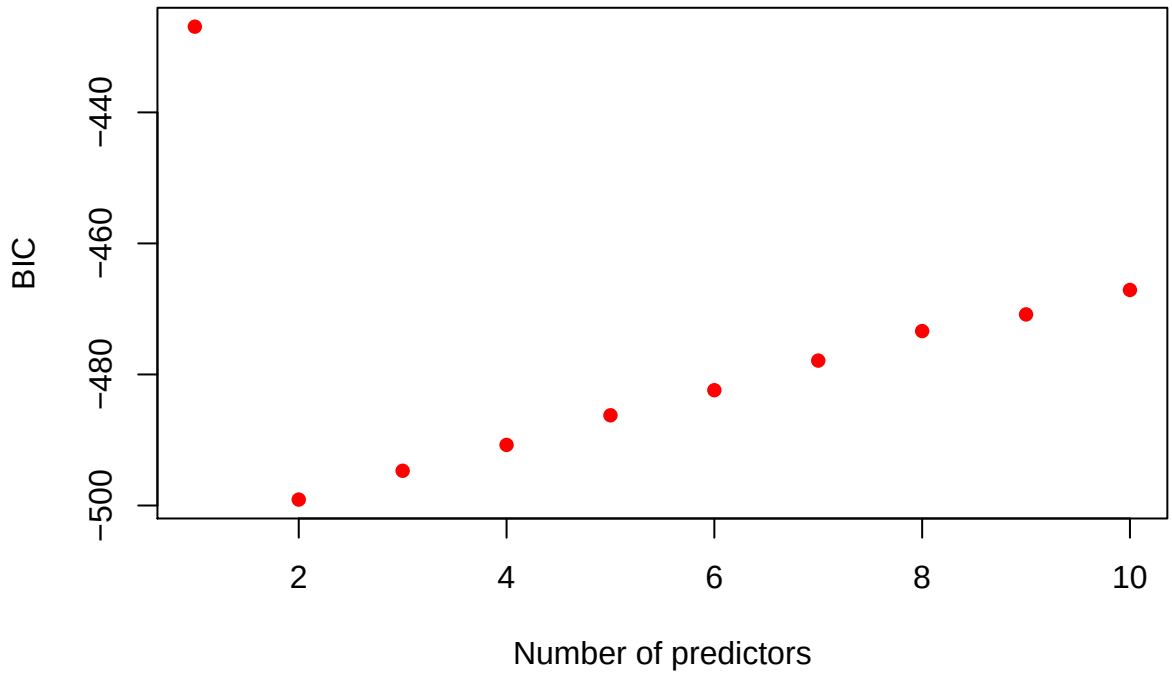
```
which.max(reg.sum.f$adjr2)    # 2
```

```
## [1] 2
```

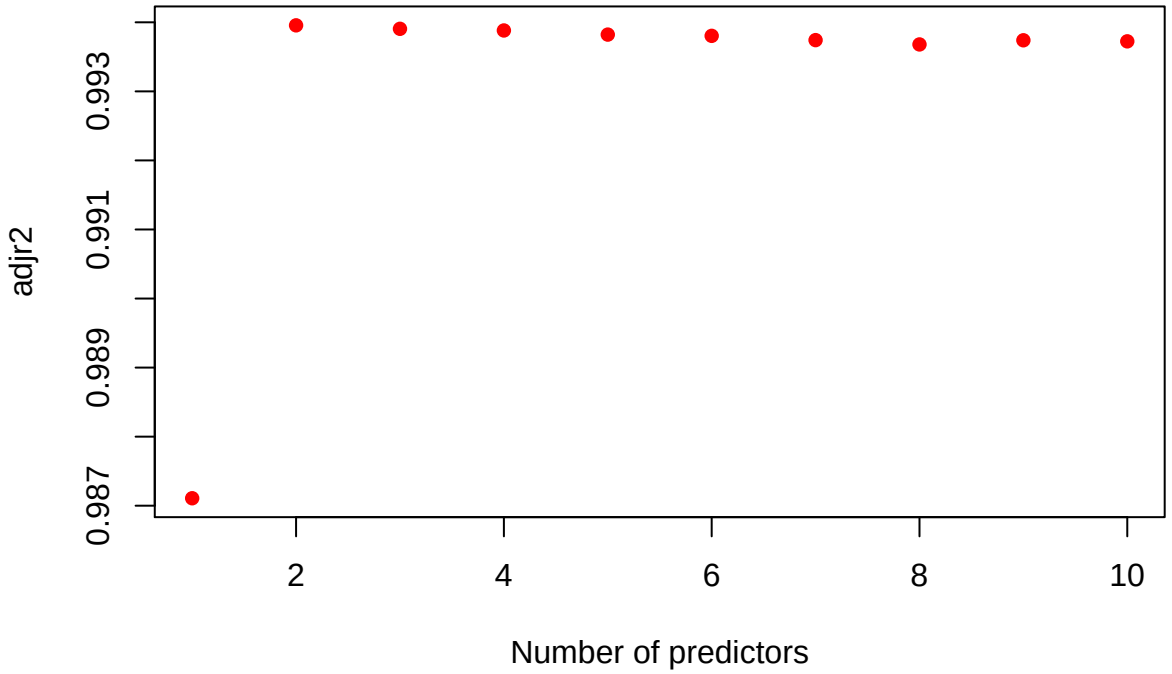
```
plot(reg.sum.f$cp, xlab="Number of predictors", ylab="Cp", col="red", type="p", pch=16) # exhibit 4
```




```
plot(reg.sum.f$bic, xlab="Number of predictors", ylab="BIC", col="red", type="p", pch=16) # exhibit 5
```




```
plot(reg.sum.f$adjr2, xlab="Number of predictors", ylab="adjr2", col="red", type="p", pch=16) # exhibit
```




```
coef(regsub.f, id=2)
```

```
##                       (Intercept) poly(x, degree = 10, raw = TRUE)2
##                        -2.0710776                         0.9642645
## poly(x, degree = 10, raw = TRUE)3
##                         5.0163042
```

For forward selection, we see the same reults as in part c. Now, we will do backward selection:

```
regsub.b <- regsubsets(y~poly(x,degree=10,raw=TRUE), data=data, nvmax=10, method="backward")
reg.sum.b <- summary(regsub.b)

which.min(reg.sum.b$cp)       # 2
```

```
## [1] 2
```

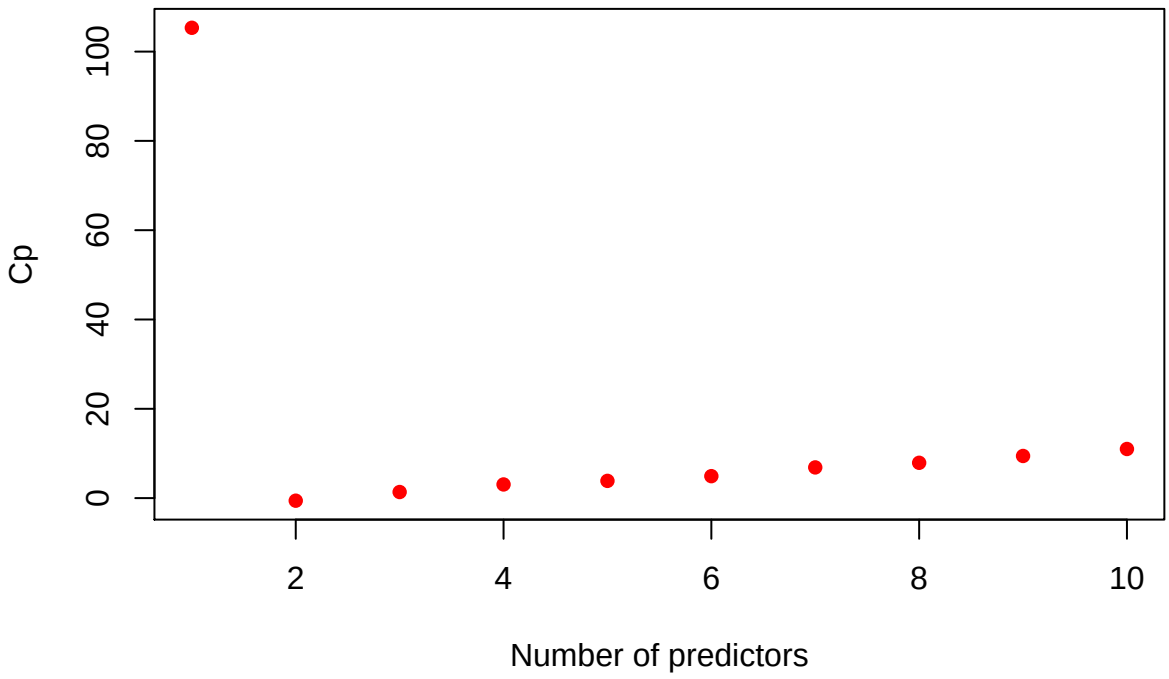
```
which.min(reg.sum.b$bic)      # 2
```

```
## [1] 2
```

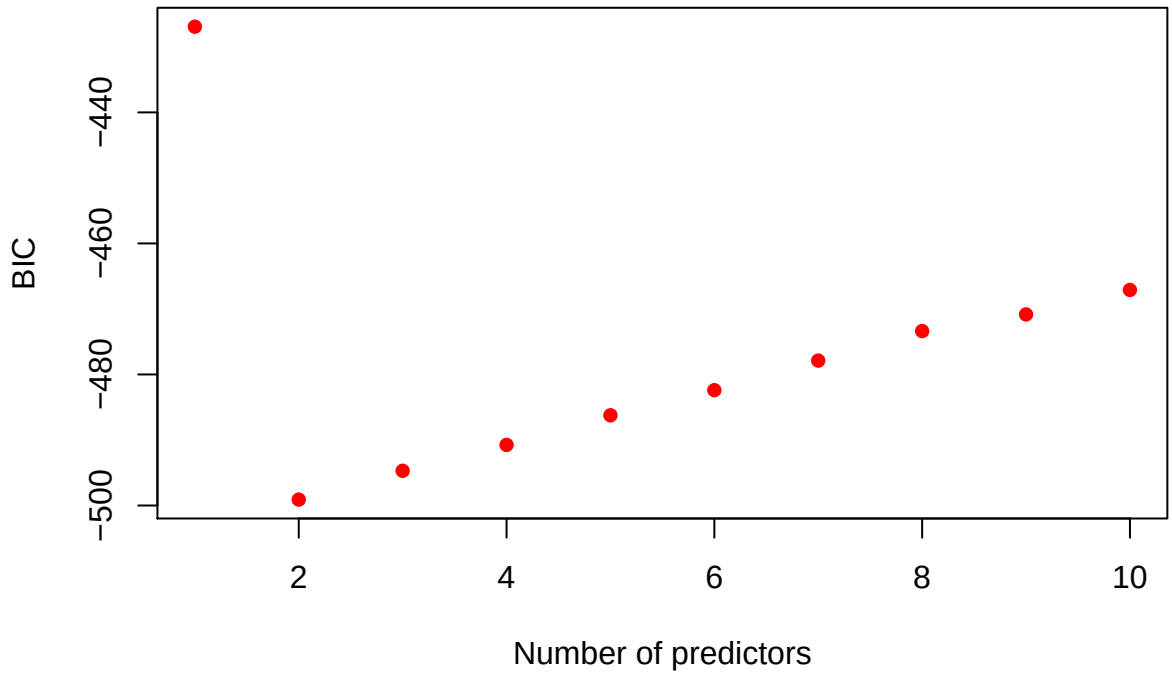
```
which.max(reg.sum.b$adjr2)    # 2
```

```
## [1] 2
```

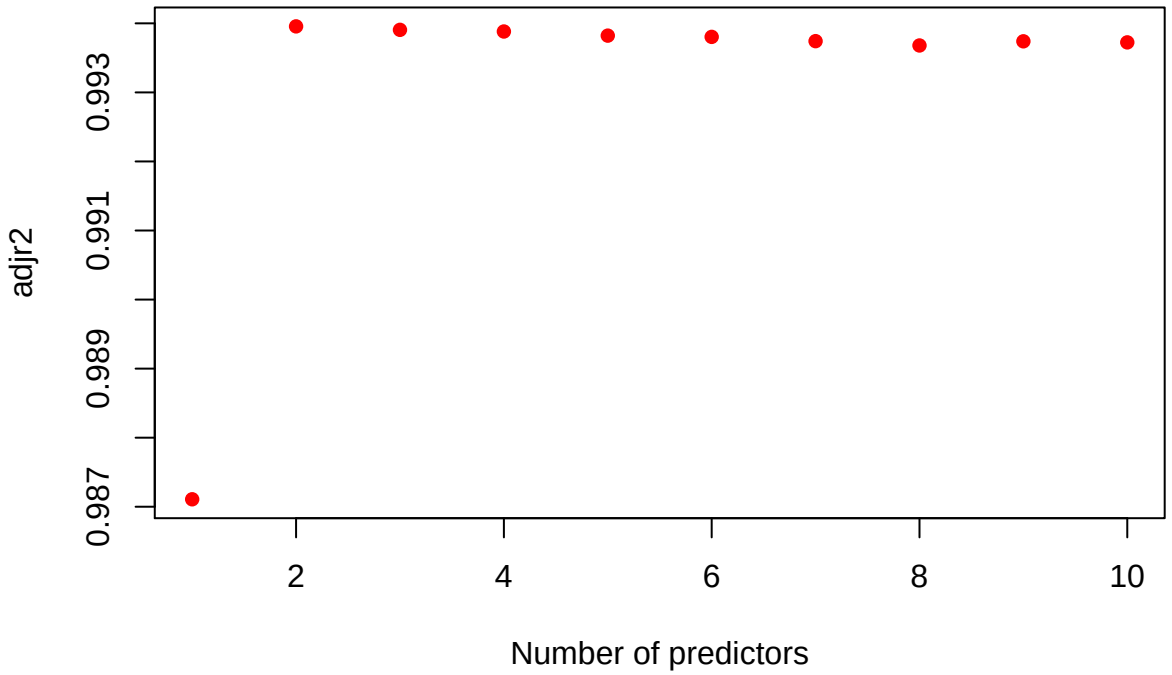
```
plot(reg.sum.b$cp, xlab="Number of predictors", ylab="Cp", col="red", type="p", pch=16) # exhibit 7
```




```
plot(reg.sum.f$bic, xlab="Number of predictors", ylab="BIC", col="red", type="p", pch=16) # exhibit 8
```




```
plot(reg.sum.f$adjr2, xlab="Number of predictors", ylab="adjr2", col="red", type="p", pch=16) # exhibit
```




```
coef(regsub.b, id=2)
```

```
##                       (Intercept) poly(x, degree = 10, raw = TRUE)2
##                        -2.0710776                         0.9642645
## poly(x, degree = 10, raw = TRUE)3
##                         5.0163042
```

Again, we see the same model and values.

Part e

Fit a LASSO model and use cross-validation to select the optimal value for lambda.

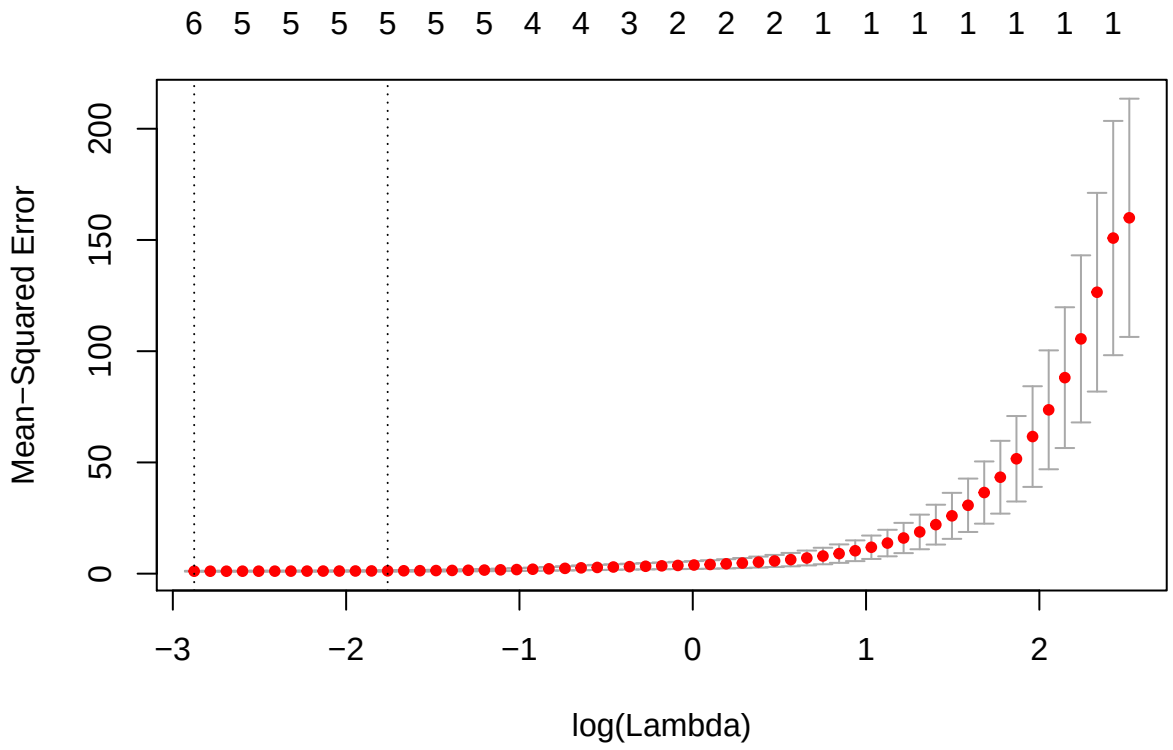
```
library(glmnet) # for LASSO
```

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

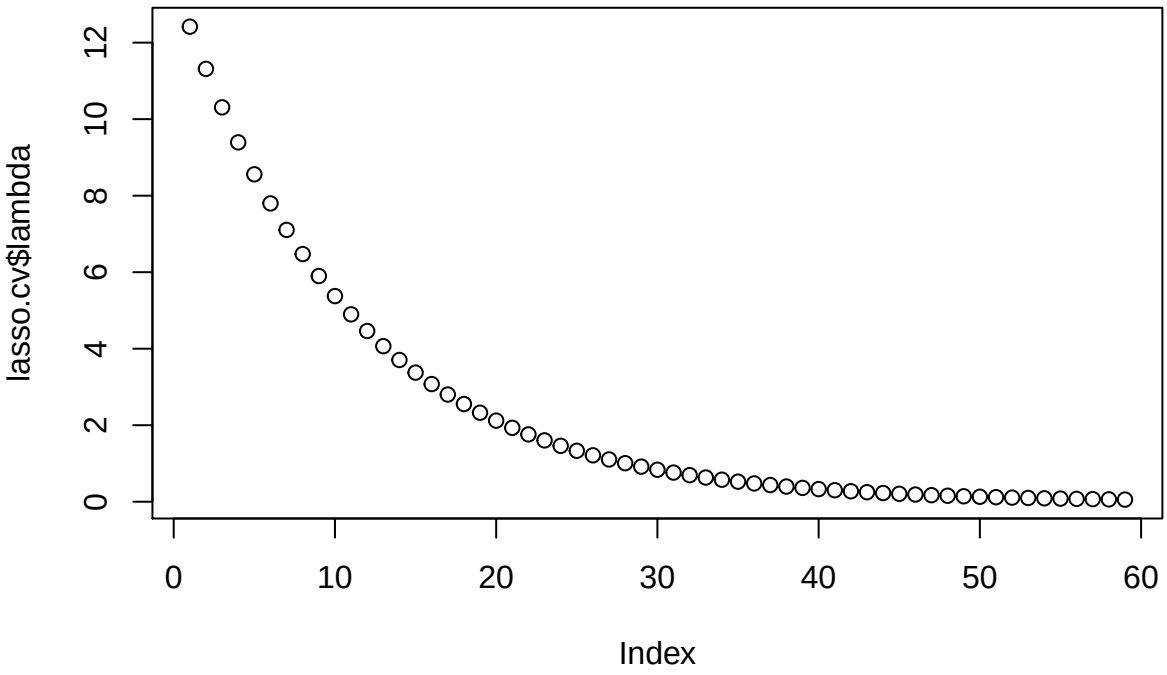
```
## Loading required package: foreach
```

```
## Loaded glmnet 2.0-5
```

```
X.lasso <- model.matrix(y~poly(x,degree=10,raw=TRUE), data=data)[,-1]
Y.lasso <- Y
lasso.cv <- cv.glmnet(X.lasso, Y.lasso, alpha=1, nfolds=10)
plot(lasso.cv) # exhibit 10
```




```
plot(lasso.cv$lambda) # exhibit 11
```




```
lambda.final <- lasso.cv$lambda.min # 0.05631109

beta <- coef(lasso.cv, s=lambda.final)
beta <- as.matrix(beta)
beta
```

```
##                                             1
## (Intercept)                       -1.9625901145
## poly(x, degree = 10, raw = TRUE)1  0.0056439028
## poly(x, degree = 10, raw = TRUE)2  0.7397802716
## poly(x, degree = 10, raw = TRUE)3  4.9391488225
## poly(x, degree = 10, raw = TRUE)4  0.0361210858
## poly(x, degree = 10, raw = TRUE)5  0.0041259611
## poly(x, degree = 10, raw = TRUE)6  0.0000000000
## poly(x, degree = 10, raw = TRUE)7  0.0000000000
## poly(x, degree = 10, raw = TRUE)8  0.0000000000
## poly(x, degree = 10, raw = TRUE)9  0.0003331596
## poly(x, degree = 10, raw = TRUE)10 0.0000000000
```

Using the LASSO method, the model picks x, x^2, x^3, x^4, x^5, x^9. The coefficients fo x^9, x, and x^5 are more negligible than the others.

Part f

```
B7 <- 7
Y.f <- B0 + B7*X^7 + epsilon
data.f <- data.frame(y=Y.f, x=X)
```

Model selection using regsubsets:

```
regsub.partf <- regsubsets(y~poly(x,degree=10,raw=TRUE), data=data.f, nvmax=10)
reg.sum.partf <- summary(regsub.partf)

which.min(reg.sum.partf$cp)       # 1
```

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

```
which.min(reg.sum.partf$bic)      # 1
```

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

```
which.max(reg.sum.partf$adjr2)    # 1
```

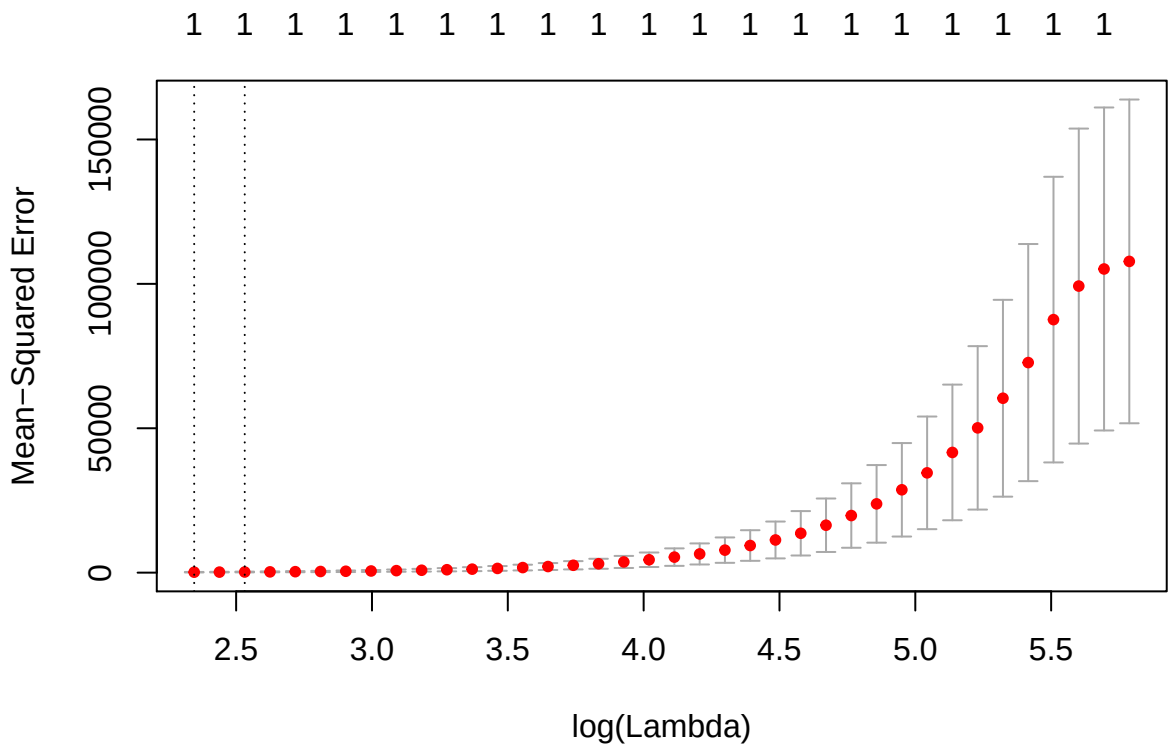
```
## [1] 1
```

```
coef(regsub.partf, id=1)
```

```
##                        (Intercept) poly(x, degree = 10, raw = TRUE)7
##                          -2.095239                          6.999867
```

All three methods chose one-variable models. Now, using LASSO:

```
X.lasso.f <- model.matrix(y~poly(x,degree=10,raw=TRUE), data=data.f)[,-1]
Y.lasso.f <- Y.f
lasso.cv.f <- cv.glmnet(X.lasso.f, Y.lasso.f, alpha=1, nfolds=10)
plot(lasso.cv.f) # exhibit 12
```



```
lambda.final.f <- lasso.cv.f$lambda.min # 10.43878

beta <- coef(lasso.cv.f, s=lambda.final.f)
beta <- as.matrix(beta)
beta
```

```
##                                          1
## (Intercept)                      -2.561234
## poly(x, degree = 10, raw = TRUE)1   0.000000
## poly(x, degree = 10, raw = TRUE)2   0.000000
## poly(x, degree = 10, raw = TRUE)3   0.000000
## poly(x, degree = 10, raw = TRUE)4   0.000000
## poly(x, degree = 10, raw = TRUE)5   0.000000
## poly(x, degree = 10, raw = TRUE)6   0.000000
## poly(x, degree = 10, raw = TRUE)7   6.775923
## poly(x, degree = 10, raw = TRUE)8   0.000000
## poly(x, degree = 10, raw = TRUE)9   0.000000
## poly(x, degree = 10, raw = TRUE)10  0.000000
```

Using LASSO, the model selected is also a one-variable model.

Problem 2

Part 1 - Data Summary & Inspection

```
# Library imports
library(dplyr)
library(ggplot2)
library(mapproj)
library(viridis)
# Import the crime data set
crime.data <- read.csv("CrimeData.csv", header=T, na.string=c("", "?"))
dim(crime.data)
```

```
## [1] 2215  147
```

We use the example code (provided below) from the file "Rcode_CrimeRate_Summary_dplyr_heatmap.r", for preprocessing data and drawing heatmaps.

```
preprocessing <- function (data) {
  # Preprocess the data for plotting the heatmap
  #
  # Args:
  #   data: data with two columns including the abbre of states
  #         along with the corresponding target number
  #
  # Returns:
  #   Add standard state name with state coordination
```

```
  # standard state name to match with mapdata
  data$region <- tolower(state.name[match(data$state,state.abb)])

  # state coordination, i.e. latitutde and longitude
  data$center_lat  <- state.center$x[match(data$state, state.abb)]
  data$center_long <- state.center$y[match(data$state, state.abb)]

  data
}

plot.heatmap <- function(data, mapdata, target) {
  # Plot out the heatmap
  #
  # Args:
  #   data: data of standard state name and coordination with corresponding
  #         target data
  #   mapdata: mapdata following the format of map_data in ggplot
  #   target: name of the target
  #
  # Returns:
  #   The heatmap of the target

  data <- preprocessing(data)

  # merge the data with the map
  map <- merge(mapdata, data, sort=FALSE, by="region", all.x=TRUE)
  map <- map[order(map$order),]

  # calculate the target range
  min <- eval(parse(text=paste("min(data$", target,")")))
  min_digits <- unlist(strsplit(as.character(floor(min)), ""))
  min_range <- as.numeric(min_digits[1]) * 10^(length(min_digits)-1)

  max <- eval(parse(text=paste("max(data$", target, ")")))
  max_range <- round(max, -floor(log10(max)))

  # plot the map
  heapmap <- ggplot(map, aes(x=long, y=lat, group=group))
  heapmap <- heapmap + eval(parse(text=paste("geom_polygon(aes(fill=", target,"))")))
  heapmap <- heapmap + geom_path()
  heapmap <- heapmap + geom_text(data=data, aes(x=center_lat, y=center_long, group=NA, label=state, siz
  legend_name <- target
  heapmap <- heapmap + scale_fill_continuous(limits=c(min_range, max_range), name=target)
  # you can specify the color by the Hex Color Code
  # heapmap <- heapmap + scale_fill_gradient(low="#0099CC", high="#003366")
  heapmap <- heapmap + scale_fill_viridis()
  # heapmap <- heapmap + scale_fill_viridis(option="magma")
  heapmap
}
```

We can now process the input data and plot a heatmap for our two selected variables of interest: mean percentage unemployed and mean percentage of the population under poverty. We create a dataframe to encapsulate these variables from the original data source and use aggregated data to create heatmaps. We also retain in a summarized crime rate field, as calculated in the example code, for later reference.

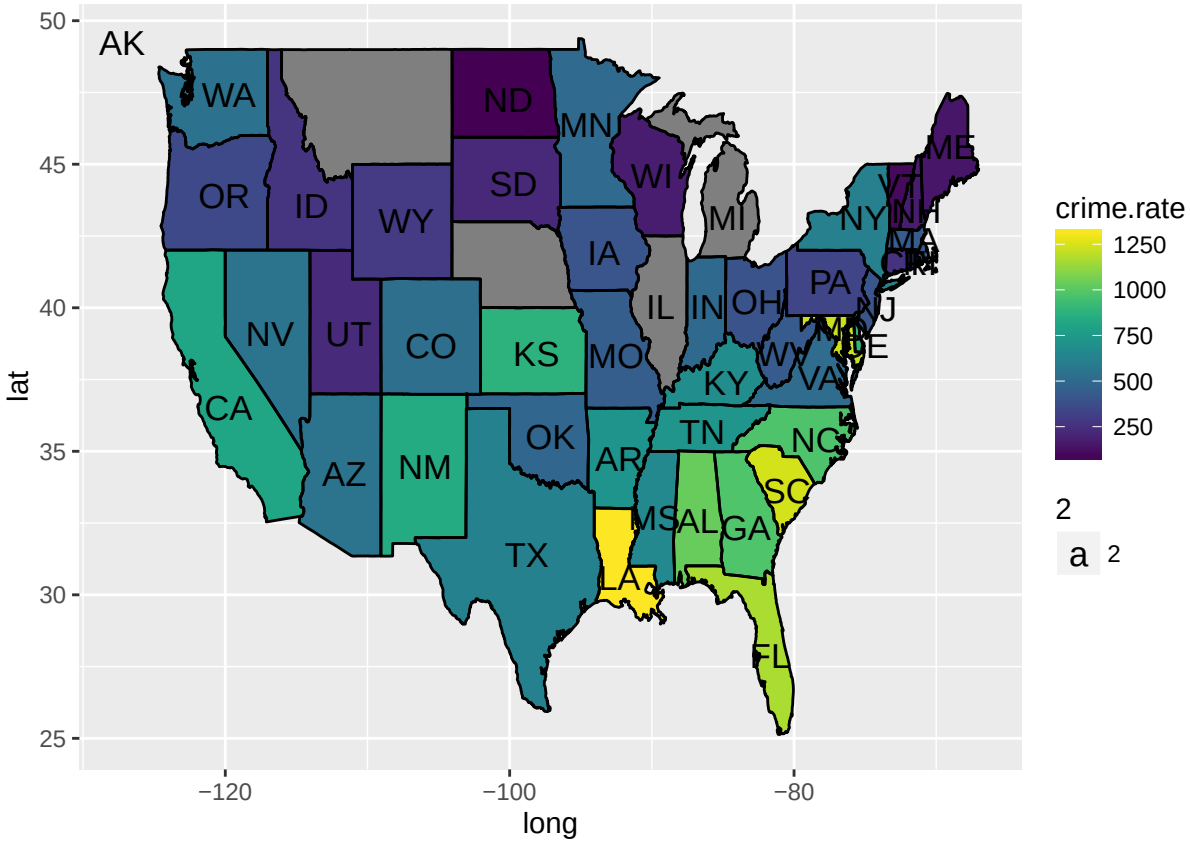
```
data.s <- summarize(group_by(crime.data, state),
            mean.pct.unemployed=mean(pct.unemployed),
            mean.pct.pop.underpov=mean(pct.pop.underpov),
            crime.rate=mean(violentcrimes.perpop, na.rm=TRUE), #ignore the missing values
            n=n())
# mapdata
states <- map_data("state")

# Plot of crime rate
crime.rate <- data.s[, c("state", "crime.rate")]
plot.heatmap(crime.rate, states, "crime.rate")
```

```
## Warning in plot.heatmap(crime.rate, states, "crime.rate"): NAs introduced
## by coercion

## Scale for 'fill' is already present. Adding another scale for 'fill',
## which will replace the existing scale.

## Warning: Removed 1 rows containing missing values (geom_text).
```



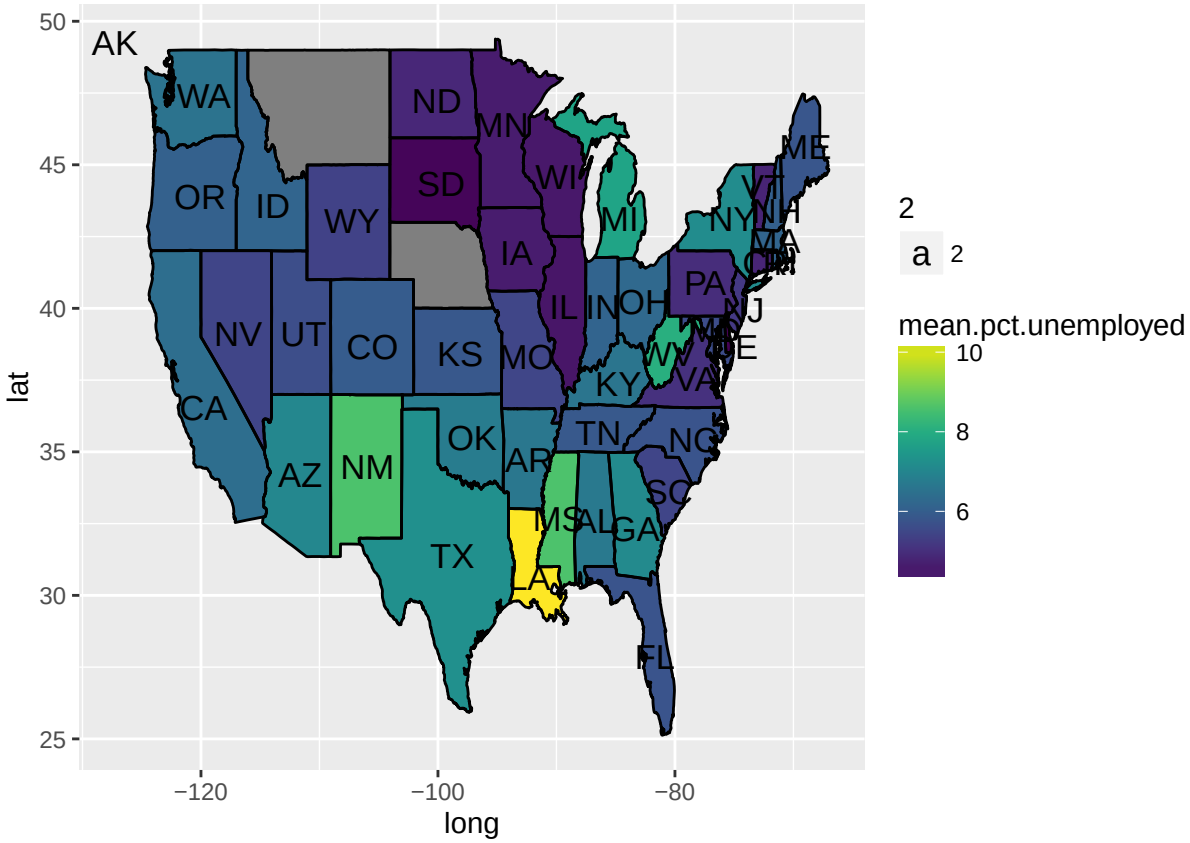
With this information in mind, we can now plot our two variables of interest.

```
# warnings suppressed
# Plotting mean percentage unemployed
pct.unemployed <- data.s[, c("state", "mean.pct.unemployed")]
pct.unemployed
```

```
## # A tibble: 48 × 2
##     state mean.pct.unemployed
##    <fctr>               <dbl>
## 1      AK            6.610000
## 2      AL            6.635581
## 3      AR            6.596000
## 4      AZ            7.012000
## 5      CA            6.350860
## 6      CO            5.896400
## 7      CT            4.729859
## 8      DC            7.030000
## 9      DE            4.070000
## 10     FL            5.713444
## # ... with 38 more rows
```

```
plot.heatmap(pct.unemployed, states, "mean.pct.unemployed")
```

```
## Scale for 'fill' is already present. Adding another scale for 'fill',
## which will replace the existing scale.
```



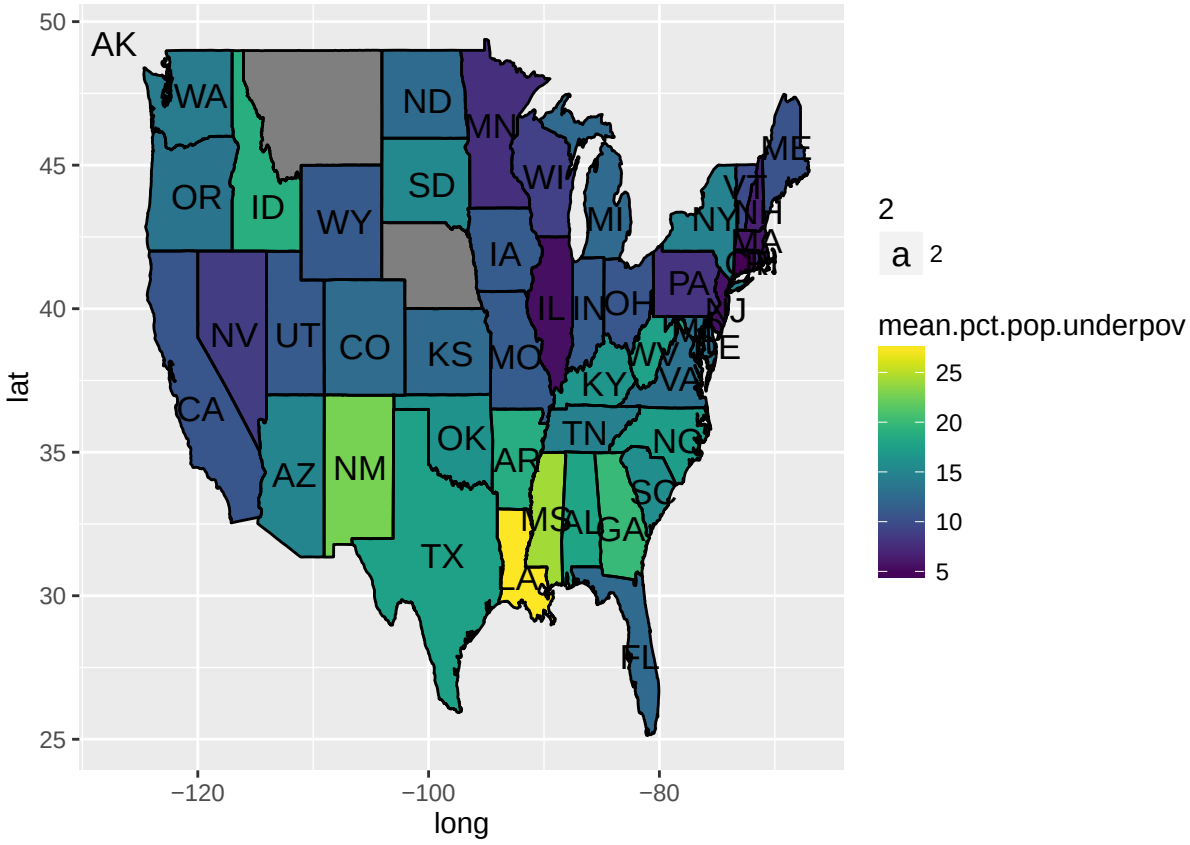

```
# Plotting mean population under poverty
pct.pop.underpov <- data.s[, c("state", "mean.pct.pop.underpov")]
pct.pop.underpov
```

```
## # A tibble: 48 × 2
```

```
##     state mean.pct.pop.underpov
##    <fctr>                 <dbl>
## 1      AK              7.683333
## 2      AL             17.863256
## 3      AR             18.780800
## 4      AZ             15.036500
## 5      CA             10.789140
## 6      CO             12.586800
## 7      CT              4.812394
## 8      DC             16.870000
## 9      DE             12.450000
## 10     FL             12.471444
## # ... with 38 more rows
```

```
plot.heatmap(pct.pop.underpov, states, "mean.pct.pop.underpov")
```

```
## Scale for 'fill' is already present. Adding another scale for 'fill',
## which will replace the existing scale.
```



Observations

Mean Percentage Unemployed: We can see a relatively strong relationship between crime rate and mean percent unemployed based on the data in the heatmap. However, this may not be an accurate isolated predictor of crime rate, given the large variety of factors that can affect it in the real world. There are states

like South Carolina and Minnesota for instance where we can see that the values for percent unemployed and crime rate are not correlated.

Mean Percentage Under Poverty: This heatmap is very strongly correlated to the values displayed for crime rate, as would be expected - areas under the natural rate of poverty in the country are prone to higher crime rates. As with unemployment however, there are exceptions to this matching, like the state of Idaho which has a high percent of population under poverty, but a relatively low crime rate.

Part 2 - Analysis with LASSO and Elasticnet

Before we proceed with the analysis, as done in lecture, we remove redudant variables from the data set (which can be calculated later if needed). We finally combine clean data for only the states of FL and CA.

```
# remove variables about police departments because of large number of missing values
data1 <- crime.data[,c(2,6:103,121,122,123, 130:147)]

# remove redudant variables (only need one of num.X and pct.X for instance)
var_names_out <- c("num.urban","other.percap", "num.underpov",
                   "num.vacant.house","num.murders","num.rapes",
                   "num.robberies", "num.assaults", "num.burglaries",
                   "num.larcenies", "num.autothefts", "num.arsons")

data1 <- data1[!(names(data1) %in% var_names_out)] # take some redundant var's out

# remove variables that can be recalculated when needed
names_other_crimes <- c( "murder.perpop", "rapes.perpop",
                         "robberies.perpop",  "assaults.perpop",
                         "burglaries.perpop", "larcenies.perpop",
                         "autothefts.perpop", "arsons.perpop",
                          "nonviolentcrimes.perpop")
# Take other crimes out
data2 <- data1[!(names(data1) %in% names_other_crimes)] # take other crimes out.


# Lastly, we combine information for the states of FL and CA only
data.fl <- data2[data2$state=="FL",-1] # take state column out
data.ca <- data2[data2$state=="CA",-1]

# Combined data ready!
comb.data <- rbind(data.fl, data.ca)
```

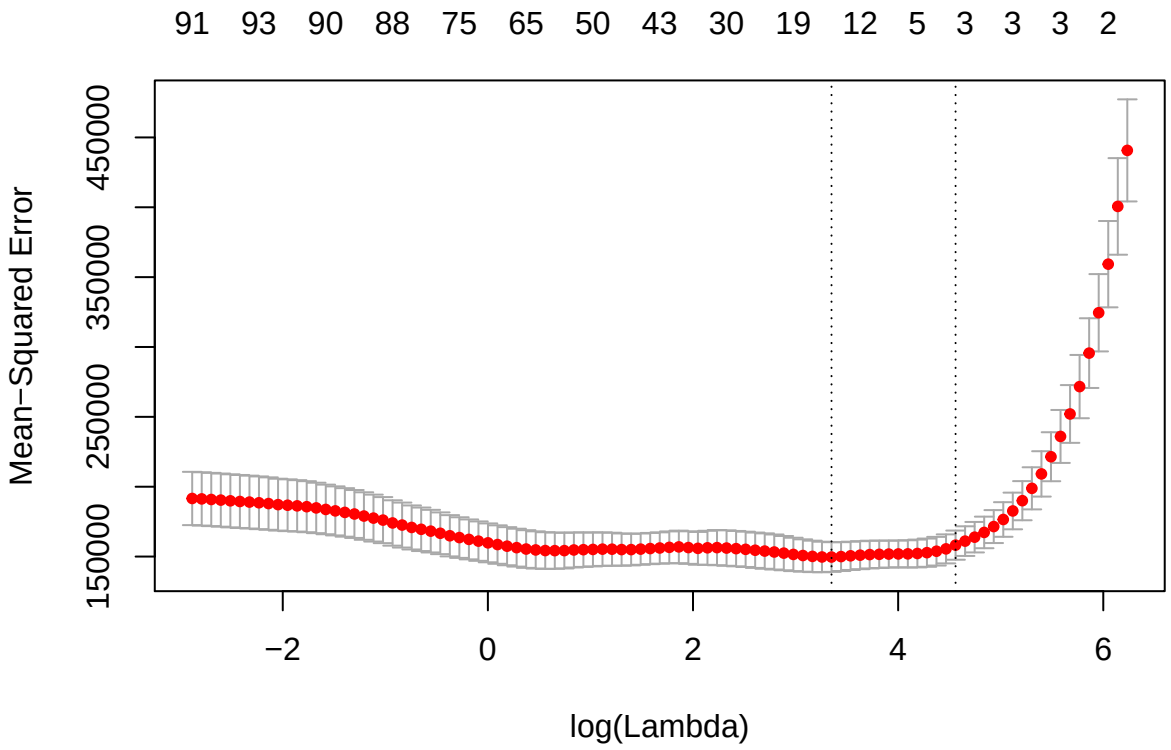
Now we can proceed to generate a LASSO model.

```
comb.data <- comb.data[complete.cases(comb.data),]

# X variables as a categorical matrix
X2 <- model.matrix(violentcrimes.perpop~., comb.data)[,-1]

# response variable
Y2 <- comb.data$violentcrimes.perpop

# alpha set close to 1, with 10 folds of cross-validation
fit2.cv <- cv.glmnet(X2, Y2, alpha=1, nfolds=10)
plot(fit2.cv)
```




```
# extract betas coefficients values for lambda.1se (fewer number of variables in the model for simplici
coef.1se <- coef(fit2.cv, s="lambda.1se")
coef.1se <- coef.1se[which(coef.1se !=0),]
coef.1se
```

```
##          (Intercept)       race.pctblack    pct.kids2parents
##        1843.0067721           8.9724646         -18.9603695
## pct.kids.nvrmarried    pct.house.vacant
##          81.1150441           0.5217146
```

```
rownames(as.matrix(coef.1se))
```

```
## [1] "(Intercept)"         "race.pctblack"       "pct.kids2parents"
## [4] "pct.kids.nvrmarried" "pct.house.vacant"
```

```
# use lambda values from above to
fit2.lasso <- glmnet(X2, Y2, alpha=1, lambda=fit2.cv$lambda.1se)
variables.final <- coef(fit2.lasso)
variables.final <- variables.final[which(variables.final !=0),]
variables.final <- rownames(as.matrix(variables.final))
variables.final
```

```
## [1] "(Intercept)"         "race.pctblack"       "pct.kids2parents"
## [4] "pct.kids.nvrmarried" "pct.house.vacant"
```

```
# join together the extracted betas for the final model
fit_formula = as.formula(paste("violentcrimes.perpop", "~", paste(variables.final[-1], collapse = "+")))
fit.final = lm(fit_formula, data=comb.data)
summary(fit.final)
```

```
##
## Call:
## lm(formula = fit_formula, data = comb.data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1046.86  -224.16   -48.04   157.19  1898.65
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)         1999.426    264.411   7.562 3.29e-13 ***
## race.pctblack         13.112      2.743   4.780 2.56e-06 ***
## pct.kids2parents     -22.686      3.346  -6.779 4.90e-11 ***
## pct.kids.nvrmarried   85.510     12.746   6.709 7.54e-11 ***
## pct.house.vacant      27.754     11.036   2.515   0.0123 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 375.6 on 363 degrees of freedom
## Multiple R-squared:  0.6846, Adjusted R-squared:  0.6811
## F-statistic:   197 on 4 and 363 DF,  p-value: < 2.2e-16
```

The predicted formula generated through the LASSO model (reading from the summary provided above) is violentcrimes.perpop = 2012.95 + 12.96(race.pctblack) - 22.68(pct.kids2parents) + 94.95(pct.kids.nvrmarried).

Cross validating lambdas and alphas

We now loop through 20 different values of alpha and lambdas and compare CVMs, choosing only the final model with the lowest CVM.

```
Xb <- model.matrix(violentcrimes.perpop~., comb.data)[,-1]
Yb <- comb.data[, 98]

# test alpha values
alpha <- seq(0, 1, .05)
results <- data.frame()

for(i in 1:21){
  # 10 folds of CV each iteration
 temp_model<- cv.glmnet(Xb, Yb, alpha = alpha[i], family="gaussian", nfolds = 10, type.measure = "devia
 pos<-which(temp_model$lambda==temp_model$lambda.1se)
 # store alpha, CVM pairs for easy reporting
 results[i,1] <- alpha[i]
 results[i,2] <- temp_model$cvm[pos]
}

pos.cvm<-which.min(results[,2])
cvm_final<-results[pos.cvm,2]
cvm_final
```

```
## [1] 160559.3
```

```
alpha_final<-results[pos.cvm,1]
alpha_final
```

```
## [1] 0.9
```

We can see that the best model results (averaged over a series of iterations) with an alpha value of 0.85 (and corresponding CVM of 158220).

Below are our outputs based on the model we found above by crossvalidating the alphas and lambdas

```
fit_alpha_final <- cv.glmnet(Xb, Yb, alpha = alpha_final, family="gaussian", nfolds = 10, type.measure =
fit_alpha_final$lambda.1se
```

```
## [1] 106.1783
```

```
coef_alpha_final.1se <- coef(fit_alpha_final, s="lambda.1se")
coef_alpha_final.1se <- coef_alpha_final.1se[which(coef_alpha_final.1se !=0),]
coef_alpha_final.1se
```

```
##         (Intercept)       race.pctblack    pct.kids2parents
##         1836.954833            9.103510          -18.831129
## pct.kids.nvrmarried    pct.house.vacant
##           79.447691            1.590453
```

```
rownames(as.matrix(coef_alpha_final.1se))
```

```
## [1] "(Intercept)"         "race.pctblack"       "pct.kids2parents"
## [4] "pct.kids.nvrmarried" "pct.house.vacant"
```

Below we create an OLS model using the variables from the model determined through LASSO.

```
fitQ2.lm<-lm(fit_formula, data=comb.data)
summary(fitQ2.lm)
```

```
##
## Call:
## lm(formula = fit_formula, data = comb.data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1046.86  -224.16   -48.04   157.19  1898.65
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)         1999.426    264.411   7.562 3.29e-13 ***
## race.pctblack         13.112      2.743   4.780 2.56e-06 ***
## pct.kids2parents     -22.686      3.346  -6.779 4.90e-11 ***
## pct.kids.nvrmarried   85.510     12.746   6.709 7.54e-11 ***
## pct.house.vacant      27.754     11.036   2.515   0.0123 *
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 375.6 on 363 degrees of freedom
## Multiple R-squared:  0.6846, Adjusted R-squared:  0.6811
## F-statistic:   197 on 4 and 363 DF,  p-value: < 2.2e-16
```

Finally, we can see that the results in both types of models are significant at the 0.001 level, and that the OLS model is remarkably similar to the crossvalidated model generated with the best fit version of alpha.