SurvMeth 640 Assignment 2-Revised

Cheng, Chia Wen

2023-03-13

Setup

```
library(glmnet)
library(caret)
library(repr)
library(plyr)
library(readr)
library(dplyr)
library(ggplot2)
```

Data

For this exercise we use the Communities and Crime data from the UCI ML repository, which includes information about communities in the US. "The data combines socio-economic data from the 1990 US Census, law enforcement data from the 1990 US LEMAS survey, and crime data from the 1995 FBI UCR"

Source: https://archive.ics.uci.edu/ml/datasets/Communities+and+Crime

First, some data prep.

```
crime <- read.csv("communities.data", header = FALSE, na.strings = "?")
varnames <- read.delim("communities.txt", header = FALSE)
```

Clean name vector and use as variable names.

```
varnames <- as.character(varnames$V1)
varnames <- gsub("@attribute ", "", varnames)
varnames <- gsub(" numeric", "", varnames)
varnames <- gsub(" string", "", varnames)
names(crime) <- varnames
```

To make things easier, drop columns with missing values.

```
crime <- crime[, colSums(is.na(crime)) == 0]
```

Check whats left.

```
str(crime)
```

```
## 'data.frame':    1994 obs. of  103 variables:
##  $ state                 : int  8 53 24 34 42 6 44 6 21 29 ...
##  $ communityname         : chr  "Lakewoodcity" "Tukwilacity" "Aberdeentown" "Willingborotownship" ...
##  $ fold                  : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ population            : num  0.19 0 0 0.04 0.01 0.02 0.01 0.01 0.03 0.01 ...
##  $ householdsize         : num  0.33 0.16 0.42 0.77 0.55 0.28 0.39 0.74 0.34 0.4 ...
##  $ racepctblack          : num  0.02 0.12 0.49 1 0.02 0.06 0 0.03 0.2 0.06 ...
##  $ racePctWhite          : num  0.9 0.74 0.56 0.08 0.95 0.54 0.98 0.46 0.84 0.87 ...
##  $ racePctAsian          : num  0.12 0.45 0.17 0.12 0.09 1 0.06 0.2 0.02 0.3 ...
##  $ racePctHisp           : num  0.17 0.07 0.04 0.1 0.05 0.25 0.02 1 0 0.03 ...
##  $ agePct12t21           : num  0.34 0.26 0.39 0.51 0.38 0.31 0.3 0.52 0.38 0.9 ...
##  $ agePct12t29           : num  0.47 0.59 0.47 0.5 0.38 0.48 0.37 0.55 0.45 0.82 ...
##  $ agePct16t24           : num  0.29 0.35 0.28 0.34 0.23 0.27 0.23 0.36 0.28 0.8 ...
##  $ agePct65up            : num  0.32 0.27 0.32 0.21 0.36 0.37 0.6 0.35 0.48 0.39 ...
##  $ numbUrban             : num  0.2 0.02 0 0.06 0.02 0.04 0.02 0 0.04 0.02 ...
##  $ pctUrban              : num  1 1 0 1 0.9 1 0.81 0 1 1 ...
##  $ medIncome             : num  0.37 0.31 0.3 0.58 0.5 0.52 0.42 0.16 0.17 0.54 ...
##  $ pctWWage              : num  0.72 0.72 0.58 0.89 0.72 0.68 0.5 0.44 0.47 0.59 ...
##  $ pctWFarmSelf          : num  0.34 0.11 0.19 0.21 0.16 0.2 0.23 1 0.36 0.22 ...
##  $ pctWInvInc            : num  0.6 0.45 0.39 0.43 0.68 0.61 0.68 0.23 0.34 0.86 ...
##  $ pctWSocSec            : num  0.29 0.25 0.38 0.36 0.44 0.28 0.61 0.53 0.55 0.42 ...
##  $ pctWPubAsst           : num  0.15 0.29 0.4 0.2 0.11 0.15 0.21 0.97 0.48 0.02 ...
##  $ pctWRetire            : num  0.43 0.39 0.84 0.82 0.71 0.25 0.54 0.41 0.43 0.31 ...
##  $ medFamInc             : num  0.39 0.29 0.28 0.51 0.46 0.62 0.43 0.15 0.21 0.85 ...
##  $ perCapInc             : num  0.4 0.37 0.27 0.36 0.43 0.72 0.47 0.1 0.23 0.89 ...
##  $ whitePerCap           : num  0.39 0.38 0.29 0.4 0.41 0.76 0.44 0.12 0.23 0.94 ...
##  $ blackPerCap           : num  0.32 0.33 0.27 0.39 0.28 0.77 0.4 0.08 0.19 0.11 ...
##  $ indianPerCap          : num  0.27 0.16 0.07 0.16 0 0.28 0.24 0.17 0.1 0.09 ...
##  $ AsianPerCap           : num  0.27 0.3 0.29 0.25 0.74 0.52 0.86 0.27 0.26 0.33 ...
##  $ HispPerCap            : num  0.41 0.35 0.39 0.44 0.48 0.6 0.36 0.21 0.22 0.8 ...
##  $ NumUnderPov           : num  0.08 0.01 0.01 0.01 0 0.01 0.01 0.03 0.04 0 ...
##  $ PctPopUnderPov        : num  0.19 0.24 0.27 0.1 0.06 0.12 0.11 0.64 0.45 0.11 ...
##  $ PctLess9thGrade       : num  0.1 0.14 0.27 0.09 0.25 0.13 0.29 0.96 0.52 0.04 ...
##  $ PctNotHSGrad          : num  0.18 0.24 0.43 0.25 0.3 0.12 0.41 0.82 0.59 0.03 ...
##  $ PctBSorMore           : num  0.48 0.3 0.19 0.31 0.33 0.8 0.36 0.12 0.17 1 ...
##  $ PctUnemployed         : num  0.27 0.27 0.36 0.33 0.12 0.1 0.28 1 0.55 0.11 ...
##  $ PctEmploy             : num  0.68 0.73 0.58 0.71 0.65 0.65 0.54 0.26 0.43 0.44 ...
##  $ PctEmplManu           : num  0.23 0.57 0.32 0.36 0.67 0.19 0.44 0.43 0.59 0.2 ...
##  $ PctEmplProfServ       : num  0.41 0.15 0.29 0.45 0.38 0.77 0.53 0.34 0.36 1 ...
##  $ PctOccupManu          : num  0.25 0.42 0.49 0.37 0.42 0.06 0.33 0.71 0.64 0.02 ...
##  $ PctOccupMgmtProf      : num  0.52 0.36 0.32 0.39 0.46 0.91 0.49 0.18 0.29 0.96 ...
##  $ MalePctDivorce        : num  0.68 1 0.63 0.34 0.22 0.49 0.25 0.38 0.62 0.3 ...
##  $ MalePctNevMarr        : num  0.4 0.63 0.41 0.45 0.27 0.57 0.34 0.47 0.26 0.85 ...
##  $ FemalePctDiv          : num  0.75 0.91 0.71 0.49 0.2 0.61 0.28 0.59 0.66 0.39 ...
##  $ TotalPctDiv           : num  0.75 1 0.7 0.44 0.21 0.58 0.28 0.52 0.67 0.36 ...
##  $ PersPerFam            : num  0.35 0.29 0.45 0.75 0.51 0.44 0.42 0.78 0.37 0.31 ...
##  $ PctFam2Par            : num  0.55 0.43 0.42 0.65 0.91 0.62 0.77 0.45 0.51 0.65 ...
##  $ PctKids2Par           : num  0.59 0.47 0.44 0.54 0.91 0.69 0.81 0.43 0.55 0.73 ...
##  $ PctYoungKids2Par      : num  0.61 0.6 0.43 0.83 0.89 0.87 0.79 0.34 0.58 0.78 ...
##  $ PctTeen2Par           : num  0.56 0.39 0.43 0.65 0.85 0.53 0.74 0.34 0.47 0.67 ...
##  $ PctWorkMomYoungKids   : num  0.74 0.46 0.71 0.85 0.4 0.3 0.57 0.29 0.65 0.72 ...
```

```
##  $ PctWorkMom           : num  0.76 0.53 0.67 0.86 0.6 0.43 0.62 0.27 0.64 0.71 ...
##  $ NumIlleg             : num  0.04 0 0.01 0.03 0 0 0 0.02 0.02 0 ...
##  $ PctIlleg             : num  0.14 0.24 0.46 0.33 0.06 0.11 0.13 0.5 0.29 0.07 ...
##  $ NumImmig             : num  0.03 0.01 0 0.02 0 0.04 0.01 0.02 0 0.01 ...
##  $ PctImmigRecent       : num  0.24 0.52 0.07 0.11 0.03 0.3 0 0.5 0.12 0.41 ...
##  $ PctImmigRec5         : num  0.27 0.62 0.06 0.2 0.07 0.35 0.02 0.59 0.09 0.44 ...
##  $ PctImmigRec8         : num  0.37 0.64 0.15 0.3 0.2 0.43 0.02 0.65 0.07 0.52 ...
##  $ PctImmigRec10        : num  0.39 0.63 0.19 0.31 0.27 0.47 0.1 0.59 0.13 0.48 ...
##  $ PctRecentImmig       : num  0.07 0.25 0.02 0.05 0.01 0.5 0 0.69 0 0.22 ...
##  $ PctRecImmig5         : num  0.07 0.27 0.02 0.08 0.02 0.5 0.01 0.72 0 0.21 ...
##  $ PctRecImmig8         : num  0.08 0.25 0.04 0.11 0.04 0.56 0.01 0.71 0 0.22 ...
##  $ PctRecImmig10        : num  0.08 0.23 0.05 0.11 0.05 0.57 0.03 0.6 0 0.19 ...
##  $ PctSpeakEnglOnly     : num  0.89 0.84 0.88 0.81 0.88 0.45 0.73 0.12 0.99 0.85 ...
##  $ PctNotSpeakEnglWell  : num  0.06 0.1 0.04 0.08 0.05 0.28 0.05 0.93 0.01 0.03 ...
##  $ PctLargHouseFam      : num  0.14 0.16 0.2 0.56 0.16 0.25 0.12 0.74 0.12 0.09 ...
##  $ PctLargHouseOccup    : num  0.13 0.1 0.2 0.62 0.19 0.19 0.13 0.75 0.12 0.06 ...
##  $ PersPerOccupHous     : num  0.33 0.17 0.46 0.85 0.59 0.29 0.42 0.8 0.35 0.15 ...
##  $ PersPerOwnOccHous    : num  0.39 0.29 0.52 0.77 0.6 0.53 0.54 0.68 0.38 0.34 ...
##  $ PersPerRentOccHous   : num  0.28 0.17 0.43 1 0.37 0.18 0.24 0.92 0.33 0.05 ...
##  $ PctPersOwnOccup      : num  0.55 0.26 0.42 0.94 0.89 0.39 0.65 0.39 0.5 0.48 ...
##  $ PctPersDenseHous     : num  0.09 0.2 0.15 0.12 0.02 0.26 0.03 0.89 0.1 0.03 ...
##  $ PctHousLess3BR       : num  0.51 0.82 0.51 0.01 0.19 0.73 0.46 0.66 0.64 0.58 ...
##  $ MedNumBR             : num  0.5 0 0.5 0.5 0.5 0 0.5 0 0 0 ...
##  $ HousVacant           : num  0.21 0.02 0.01 0.01 0.01 0.02 0.01 0.01 0.04 0.02 ...
##  $ PctHousOccup         : num  0.71 0.79 0.86 0.97 0.89 0.84 0.89 0.91 0.72 0.72 ...
##  $ PctHousOwnOcc        : num  0.52 0.24 0.41 0.96 0.87 0.3 0.57 0.46 0.49 0.38 ...
##  $ PctVacantBoarded     : num  0.05 0.02 0.29 0.6 0.04 0.16 0.09 0.22 0.05 0.07 ...
##  $ PctVacMore6Mos       : num  0.26 0.25 0.3 0.47 0.55 0.28 0.49 0.37 0.49 0.47 ...
##  $ MedYrHousBuilt       : num  0.65 0.65 0.52 0.52 0.73 0.25 0.38 0.6 0.5 0.04 ...
##  $ PctHousNoPhone       : num  0.14 0.16 0.47 0.11 0.05 0.02 0.05 0.28 0.57 0.01 ...
##  $ PctWOFullPlumb       : num  0.06 0 0.45 0.11 0.14 0.05 0.05 0.23 0.22 0 ...
##  $ OwnOccLowQuart       : num  0.22 0.21 0.18 0.24 0.31 0.94 0.37 0.15 0.07 0.63 ...
##  $ OwnOccMedVal         : num  0.19 0.2 0.17 0.21 0.31 1 0.38 0.13 0.07 0.71 ...
##  $ OwnOccHiQuart        : num  0.18 0.21 0.16 0.19 0.3 1 0.39 0.13 0.08 0.79 ...
##  $ RentLowQ             : num  0.36 0.42 0.27 0.75 0.4 0.67 0.26 0.21 0.14 0.44 ...
##  $ RentMedian           : num  0.35 0.38 0.29 0.7 0.36 0.63 0.35 0.24 0.17 0.42 ...
##  $ RentHighQ            : num  0.38 0.4 0.27 0.77 0.38 0.68 0.42 0.25 0.16 0.47 ...
##  $ MedRent              : num  0.34 0.37 0.31 0.89 0.38 0.62 0.35 0.24 0.15 0.41 ...
##  $ MedRentPctHousInc    : num  0.38 0.29 0.48 0.63 0.22 0.47 0.46 0.64 0.38 0.23 ...
##  $ MedOwnCostPctInc     : num  0.46 0.32 0.39 0.51 0.51 0.59 0.44 0.59 0.13 0.27 ...
##  $ MedOwnCostPctIncNoMtg: num  0.25 0.18 0.28 0.47 0.21 0.11 0.31 0.28 0.36 0.28 ...
##  $ NumInShelters        : num  0.04 0 0 0 0 0 0 0 0.01 0 ...
##  $ NumStreet            : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ PctForeignBorn       : num  0.12 0.21 0.14 0.19 0.11 0.7 0.15 0.59 0.01 0.22 ...
##  $ PctBornSameState     : num  0.42 0.5 0.49 0.3 0.72 0.42 0.81 0.58 0.78 0.42 ...
##  $ PctSameHouse85       : num  0.5 0.34 0.54 0.73 0.64 0.49 0.77 0.52 0.48 0.34 ...
##  $ PctSameCity85        : num  0.51 0.6 0.67 0.64 0.61 0.73 0.91 0.79 0.79 0.23 ...
##  $ PctSameState85       : num  0.64 0.52 0.56 0.65 0.53 0.64 0.84 0.78 0.75 0.09 ...
##  $ LandArea             : num  0.12 0.02 0.01 0.02 0.04 0.01 0.05 0.01 0.04 0 ...
##   [list output truncated]
```

Train and test set

Next, we want to split the data into a training (75%) and a test (25%) part. This can be done by random sampling with `sample`. Note that there is a `fold` variable in the data set, but here we want to follow our own train/test procedure.

```
set.seed(3940)

train <- sample(1:nrow(crime), 0.75*nrow(crime))
crime_train <- crime[train,]
crime_test <- crime[-train,]
```

Now, prepare the training data for running regularized regression models via `glmnet`. Our prediction outcome is `ViolentCrimesPerPop`. As X, use all variables except `state`, `communityname`, and `fold`.

```
# Regularized Regression
X <- model.matrix(ViolentCrimesPerPop ~ . - state - communityname - fold, data = crime_train)[, -1]

y <- crime_train$ViolentCrimesPerPop
```

Check whether X looks ok.

```
dim(X) # 1495; 99
```

```
## [1] 1495   99
```

Lasso

Estimate a sequence of Lasso models using `glmnet`. You can stick with the defaults for choosing a range of lambdas.

```
mod_lasso <- glmnet(X, y, alpha = 1)
```

Here we want to display lambda and the coefficients of the first Lasso model.

```
mod_lasso$lambda[1]
```

```
## [1] 0.173677
```

```
mod_lasso$beta[,1]
```

```
##            population        householdsize          racepctblack
##                     0                    0                     0
##          racePctWhite          racePctAsian          racePctHisp
##                     0                    0                     0
##           agePct12t21          agePct12t29          agePct16t24
##                     0                    0                     0
##           agePct65up            numbUrban              pctUrban
##                     0                    0                     0
##            medIncome             pctWWage         pctWFarmSelf
```

```
##                     0                     0                     0
##            pctWInvInc            pctWSocSec           pctWPubAsst
##                     0                     0                     0
##            pctWRetire             medFamInc            perCapInc
##                     0                     0                     0
##           whitePerCap           blackPerCap           indianPerCap
##                     0                     0                     0
##           AsianPerCap           HispPerCap            NumUnderPov
##                     0                     0                     0
##        PctPopUnderPov       PctLess9thGrade          PctNotHSGrad
##                     0                     0                     0
##           PctBSorMore         PctUnemployed             PctEmploy
##                     0                     0                     0
##           PctEmplManu       PctEmplProfServ          PctOccupManu
##                     0                     0                     0
##      PctOccupMgmtProf         MalePctDivorce         MalePctNevMarr
##                     0                     0                     0
##          FemalePctDiv           TotalPctDiv             PersPerFam
##                     0                     0                     0
##            PctFam2Par           PctKids2Par       PctYoungKids2Par
##                     0                     0                     0
##           PctTeen2Par   PctWorkMomYoungKids            PctWorkMom
##                     0                     0                     0
##              NumIlleg              PctIlleg               NumImmig
##                     0                     0                     0
##        PctImmigRecent          PctImmigRec5          PctImmigRec8
##                     0                     0                     0
##         PctImmigRec10        PctRecentImmig          PctRecImmig5
##                     0                     0                     0
##          PctRecImmig8         PctRecImmig10       PctSpeakEnglOnly
##                     0                     0                     0
##   PctNotSpeakEnglWell       PctLargHouseFam      PctLargHouseOccup
##                     0                     0                     0
##      PersPerOccupHous      PersPerOwnOccHous     PersPerRentOccHous
##                     0                     0                     0
##       PctPersOwnOccup       PctPersDenseHous        PctHousLess3BR
##                     0                     0                     0
##             MedNumBR             HousVacant           PctHousOccup
##                     0                     0                     0
##         PctHousOwnOcc       PctVacantBoarded         PctVacMore6Mos
##                     0                     0                     0
##        MedYrHousBuilt         PctHousNoPhone        PctWOFullPlumb
##                     0                     0                     0
##        OwnOccLowQuart          OwnOccMedVal         OwnOccHiQuart
##                     0                     0                     0
##              RentLowQ             RentMedian              RentHighQ
##                     0                     0                     0
##               MedRent      MedRentPctHousInc      MedOwnCostPctInc
##                     0                     0                     0
## MedOwnCostPctIncNoMtg         NumInShelters             NumStreet
##                     0                     0                     0
##        PctForeignBorn       PctBornSameState         PctSameHouse85
##                     0                     0                     0
##         PctSameCity85        PctSameState85              LandArea
```

```
##                    0                    0                    0
##              PopDens        PctUsePubTrans   LemasPctOfficDrugUn
##                    0                    0                    0
```

Same for the last Lasso model.

```
mod_lasso$lambda[ncol(mod_lasso$beta)]
```

```
## [1] 7.694969e-05
```

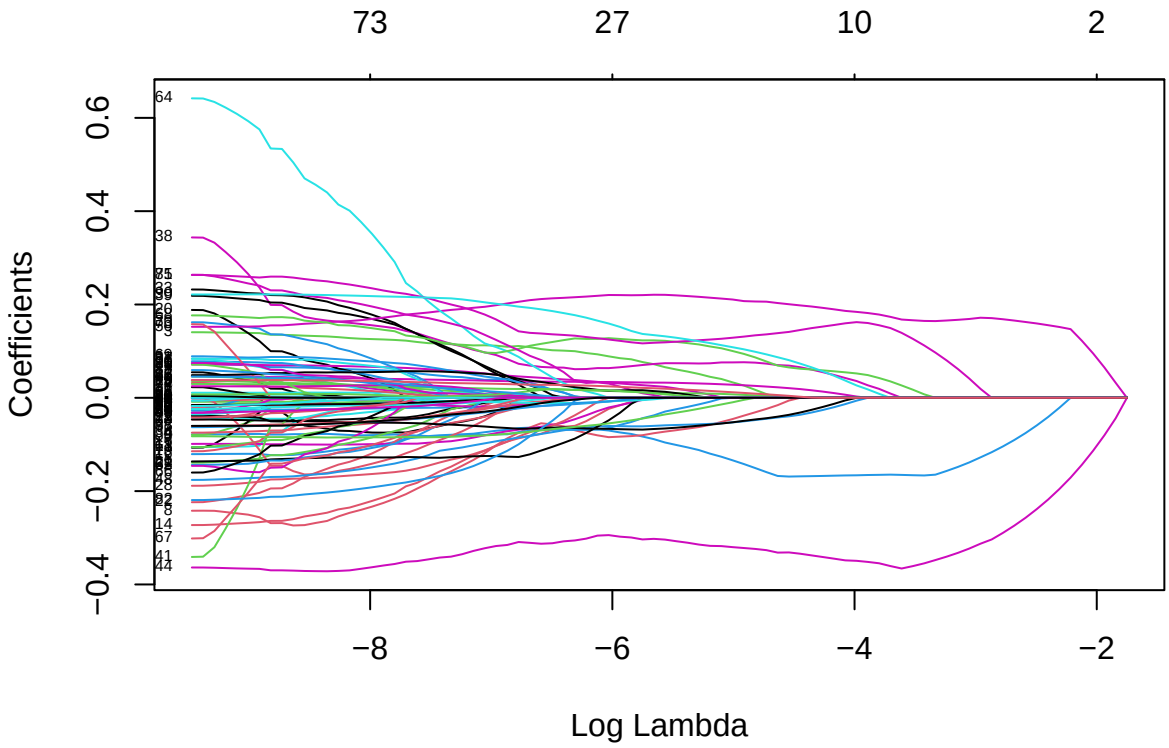
```
mod_lasso$beta[,ncol(mod_lasso$beta)]
```

```
##            population         householdsize          racepctblack
##          0.0000000000         -0.0467119436          0.1405420386
##          racePctWhite          racePctAsian          racePctHisp
##         -0.0784110189          0.0033650951          0.0377839699
##           agePct12t21           agePct12t29           agePct16t24
##          0.0547096664         -0.2420116460         -0.0794665694
##           agePct65up             numbUrban              pctUrban
##          0.0734592372         -0.1057359055          0.0462646090
##            medIncome              pctWWage          pctWFarmSelf
##         -0.1068622443         -0.2727411674          0.0361572030
##            pctWInvInc             pctWSocSec           pctWPubAsst
##         -0.1206884786          0.0000000000         -0.0457339641
##            pctWRetire              medFamInc             perCapInc
##         -0.0987426653          0.1883069483          0.0000000000
##           whitePerCap           blackPerCap          indianPerCap
##         -0.2237584263         -0.0085220184         -0.0269119338
##           AsianPerCap             HispPerCap           NumUnderPov
##          0.0109334338          0.0715714495         -0.0387939791
##        PctPopUnderPov       PctLess9thGrade          PctNotHSGrad
##         -0.1885192255         -0.1390510982          0.1619277742
##           PctBSorMore         PctUnemployed             PctEmploy
##          0.0841713604          0.0000000000          0.2321456493
##           PctEmplManu       PctEmplProfServ          PctOccupManu
##         -0.0747049577         -0.0158531222          0.0783355017
##      PctOccupMgmtProf         MalePctDivorce        MalePctNevMarr
##          0.0835359850          0.3437189719          0.2186017045
##           FemalePctDiv            TotalPctDiv              PersPerFam
##          0.0000000000         -0.3410677185         -0.0232508743
##             PctFam2Par            PctKids2Par      PctYoungKids2Par
##         -0.0304917818         -0.3636826150          0.0011372497
##            PctTeen2Par    PctWorkMomYoungKids            PctWorkMom
##         -0.0056646002          0.0456618960         -0.1759819996
##                NumIlleg              PctIlleg               NumImmig
##         -0.0354110249          0.1521929130         -0.1364410471
##         PctImmigRecent          PctImmigRec5          PctImmigRec8
##          0.0318542348         -0.0156197365         -0.0011773428
##         PctImmigRec10        PctRecentImmig          PctRecImmig5
##          0.0004034034         -0.0317792184          0.0000000000
##          PctRecImmig8         PctRecImmig10      PctSpeakEnglOnly
##          0.0228000793          0.0000000000         -0.0194631398
##    PctNotSpeakEnglWell       PctLargHouseFam     PctLargHouseOccup
```

```
##         -0.1057176919         -0.1438440982          0.0000000000
##      PersPerOccupHous     PersPerOwnOccHous    PersPerRentOccHous
##          0.6418167155         -0.1458678685         -0.1602083407
##       PctPersOwnOccup      PctPersDenseHous        PctHousLess3BR
##         -0.3014884104          0.1766640078          0.0889800426
##              MedNumBR            HousVacant           PctHousOccup
##          0.0021261371          0.2634342568         -0.0462776596
##         PctHousOwnOcc      PctVacantBoarded        PctVacMore6Mos
##          0.1578368435          0.0296965895         -0.0629292133
##        MedYrHousBuilt        PctHousNoPhone        PctWOFullPlumb
##         -0.0240986976          0.0248413661         -0.0153998620
##        OwnOccLowQuart          OwnOccMedVal         OwnOccHiQuart
##         -0.1145319372          0.0000000000          0.0710877960
##              RentLowQ            RentMedian             RentHighQ
##         -0.2191923978          0.0000000000         -0.0063219422
##               MedRent     MedRentPctHousInc      MedOwnCostPctInc
##          0.2636057732          0.0485788475         -0.0604851199
## MedOwnCostPctIncNoMtg         NumInShelters             NumStreet
##         -0.0824573364          0.0594862269          0.2214652393
##        PctForeignBorn      PctBornSameState        PctSameHouse85
##          0.0747635213          0.0037892268         -0.0423618597
##         PctSameCity85        PctSameState85              LandArea
##          0.0072765065          0.0455436207         -0.0176181733
##               PopDens        PctUsePubTrans   LemasPctOfficDrugUn
##         -0.0321320775         -0.0601713493          0.0376963355
```

Now, plot the coefficient paths.

```
plot(mod_lasso, label=T, xvar = "lambda")
```

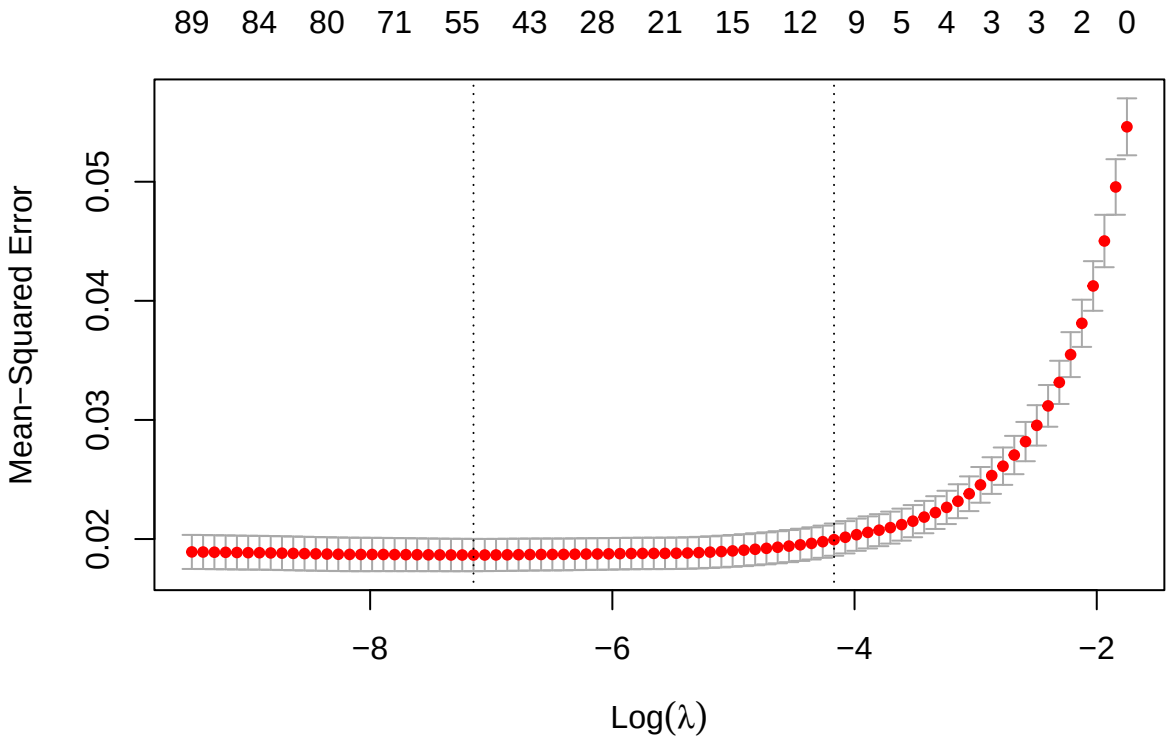


Next, we need to decide which Lasso model to pick for prediction. Use Cross-Validation for this purpose.

```
mod_cv <- cv.glmnet(X, y, alpha = 1)
```

And plot the Cross-validation results.

```
plot(mod_cv)
```



In your own words, briefly describe the CV plot. (1) What is plotted here, (2) what can you infer about the relation between the number of variables and prediction accuracy?

We are looking for the amount of penalty, λ, by cross-validation. The CV plot is resulted from (repeated) K-fold cross-validation. It plots the cross-validation curve and the upper and lower standard deviation curves as a function of the λ values used.

In this CV plot, the numbers across the top of it decrease as λ increases. This indicates the numbers of non-zero coefficients at each λ. The y-axis is mean-square error, which tells us how much error happens across all the tested values. The lower it is, the better the predictive accuracy of our model. The verticle dotted lines are the values of λ with minimum MSE, and 1-standard-error away from the minimum λ, which uses less coefficients and is not too far away from the best predictive model. We would like an optimal λ that gives us the least error in prediction.

From this cross-validation plot, we see that a full model with all features may not lead to the best model in terms of prediction performance. And thus the selection will be based on further examination.

end

Now, store the lambda value of the model with the smallest CV error as `bestlam1`.

```
coef(mod_cv, s = "lambda.min")
```

```
## 100 x 1 sparse Matrix of class "dgCMatrix"
##                                    s1
## (Intercept)            0.5761237089
## population            -0.0463935587
## householdsize          .
## racepctblack           0.1122987422
## racePctWhite          -0.0769206090
## racePctAsian           .
## racePctHisp            .
## agePct12t21            .
## agePct12t29           -0.1331217266
## agePct16t24            .
## agePct65up             0.0415925212
## numbUrban              .
## pctUrban               0.0374199639
## medIncome              .
## pctWWage              -0.1234524257
## pctWFarmSelf           0.0108349501
## pctWInvInc            -0.0798595942
## pctWSocSec             .
## pctWPubAsst           -0.0001217331
## pctWRetire            -0.0851883413
## medFamInc              .
## perCapInc              .
## whitePerCap           -0.0246831075
## blackPerCap            .
## indianPerCap          -0.0205006621
## AsianPerCap            0.0090574630
## HispPerCap             0.0544635459
## NumUnderPov           -0.0154561265
## PctPopUnderPov        -0.1205616310
## PctLess9thGrade       -0.0137631006
## PctNotHSGrad           .
## PctBSorMore            .
## PctUnemployed         -0.0188906868
## PctEmploy              0.0793996251
## PctEmplManu           -0.0279877994
## PctEmplProfServ        .
## PctOccupManu           0.0120857630
## PctOccupMgmtProf       .
## MalePctDivorce         0.1028742830
## MalePctNevMarr         0.0793974927
## FemalePctDiv          -0.0600825339
## TotalPctDiv            .
## PersPerFam             .
## PctFam2Par             .
## PctKids2Par           -0.3281005407
## PctYoungKids2Par       .
## PctTeen2Par            .
## PctWorkMomYoungKids    .
## PctWorkMom            -0.1018552897
```

```
## NumIlleg               .
## PctIlleg               0.1929311510
## NumImmig              -0.1254243135
## PctImmigRecent         0.0021537391
## PctImmigRec5           .
## PctImmigRec8           .
## PctImmigRec10          .
## PctRecentImmig         .
## PctRecImmig5           .
## PctRecImmig8           .
## PctRecImmig10          .
## PctSpeakEnglOnly       .
## PctNotSpeakEnglWell    .
## PctLargHouseFam       -0.0191730165
## PctLargHouseOccup      .
## PersPerOccupHous       0.1327899132
## PersPerOwnOccHous      .
## PersPerRentOccHous     .
## PctPersOwnOccup       -0.0452059194
## PctPersDenseHous       0.1046996609
## PctHousLess3BR         0.0496753648
## MedNumBR              -0.0084225528
## HousVacant             0.1870282212
## PctHousOccup          -0.0601281059
## PctHousOwnOcc          .
## PctVacantBoarded       0.0206533004
## PctVacMore6Mos        -0.0336155725
## MedYrHousBuilt        -0.0100651303
## PctHousNoPhone         0.0185578354
## PctWOFullPlumb        -0.0050402768
## OwnOccLowQuart         .
## OwnOccMedVal           .
## OwnOccHiQuart          .
## RentLowQ              -0.1461080051
## RentMedian             .
## RentHighQ              .
## MedRent                0.1454834738
## MedRentPctHousInc      0.0527676913
## MedOwnCostPctInc      -0.0252975151
## MedOwnCostPctIncNoMtg -0.0814868980
## NumInShelters          .
## NumStreet              0.2079504206
## PctForeignBorn         0.0072449166
## PctBornSameState       .
## PctSameHouse85         .
## PctSameCity85          0.0052332875
## PctSameState85         0.0158747779
## LandArea               .
## PopDens               -0.0071951375
## PctUsePubTrans        -0.0318339905
## LemasPctOfficDrugUn    0.0308895305
```

```
bestlam1 <- mod_cv$lambda.min
bestlam1
```

```
## [1] 0.000787604
```

Create `bestlam2` as the lambda according to the 1-standard error rule.

```
coef(mod_cv, s = "lambda.1se")
```

```
## 100 x 1 sparse Matrix of class "dgCMatrix"
##                                s1
## (Intercept)           0.49770740
## population            .
## householdsize         .
## racepctblack          .
## racePctWhite         -0.16688548
## racePctAsian          .
## racePctHisp           .
## agePct12t21           .
## agePct12t29           .
## agePct16t24           .
## agePct65up            .
## numbUrban             .
## pctUrban              0.01010712
## medIncome             .
## pctWWage              .
## pctWFarmSelf          .
## pctWInvInc            .
## pctWSocSec            .
## pctWPubAsst           .
## pctWRetire            .
## medFamInc             .
## perCapInc             .
## whitePerCap           .
## blackPerCap           .
## indianPerCap          .
## AsianPerCap           .
## HispPerCap            .
## NumUnderPov           .
## PctPopUnderPov        .
## PctLess9thGrade       .
## PctNotHSGrad          .
## PctBSorMore           .
## PctUnemployed         .
## PctEmploy             .
## PctEmplManu           .
## PctEmplProfServ       .
## PctOccupManu          .
## PctOccupMgmtProf      .
## MalePctDivorce        0.04890553
## MalePctNevMarr        .
## FemalePctDiv          .
## TotalPctDiv           .
## PersPerFam            .
## PctFam2Par            .
## PctKids2Par          -0.34473856
```

```
## PctYoungKids2Par       .
## PctTeen2Par            .
## PctWorkMomYoungKids    .
## PctWorkMom            -0.02009211
## NumIlleg               .
## PctIlleg               0.18965431
## NumImmig               .
## PctImmigRecent         .
## PctImmigRec5           .
## PctImmigRec8           .
## PctImmigRec10          .
## PctRecentImmig         .
## PctRecImmig5           .
## PctRecImmig8           .
## PctRecImmig10          .
## PctSpeakEnglOnly       .
## PctNotSpeakEnglWell    .
## PctLargHouseFam        .
## PctLargHouseOccup      .
## PersPerOccupHous       .
## PersPerOwnOccHous      .
## PersPerRentOccHous     .
## PctPersOwnOccup        .
## PctPersDenseHous       0.05683368
## PctHousLess3BR         .
## MedNumBR               .
## HousVacant             0.15601070
## PctHousOccup          -0.01293629
## PctHousOwnOcc          .
## PctVacantBoarded       .
## PctVacMore6Mos         .
## MedYrHousBuilt         .
## PctHousNoPhone         .
## PctWOFullPlumb         .
## OwnOccLowQuart         .
## OwnOccMedVal           .
## OwnOccHiQuart          .
## RentLowQ               .
## RentMedian             .
## RentHighQ              .
## MedRent                .
## MedRentPctHousInc      .
## MedOwnCostPctInc       .
## MedOwnCostPctIncNoMtg  .
## NumInShelters          .
## NumStreet              0.04731944
## PctForeignBorn         .
## PctBornSameState       .
## PctSameHouse85         .
## PctSameCity85          .
## PctSameState85         .
## LandArea               .
## PopDens                .
## PctUsePubTrans         .
```

```
## LemasPctOfficDrugUn    .
```

```
bestlam2 <- mod_cv$lambda.1se
bestlam2
```

```
## [1] 0.01546099
```

Prediction in test set

Finally, we investigate the performance of our models in the test set. For this task, construct a X matrix from the test set.

```
Xt <- model.matrix(ViolentCrimesPerPop ~ . - state - communityname - fold, data = crime_test)[,-1]
```

Use the `predict` function to generate predicted values for both models (i.e., both lambdas stored earlier).

```
p_lasso_min <- predict(mod_lasso, s = bestlam1, newx = Xt)
p_lasso_1se <- predict(mod_lasso, s = bestlam2, newx = Xt)
```

Compute the test MSE of our models.

```
MSE_lasso_min <- mean((p_lasso_min - crime_test$ViolentCrimesPerPop)^2)
MSE_lasso_min ## 0.01856262
```

```
## [1] 0.01856262
```

```
MSE_lasso_1se <- mean((p_lasso_1se - crime_test$ViolentCrimesPerPop)^2)
MSE_lasso_1se ## 0.01970093
```

```
## [1] 0.01970093
```

In addition, use another performance metric and compute the corresponding values for both models.

```
# Compute R^2 from true and predicted values
eval_results <- function(true, predicted, df) {
  SSE <- sum((predicted - true)^2)
  SST <- sum((true - mean(true))^2)
  R_squared <- 1 - SSE / SST
  RMSE = sqrt(SSE/nrow(df))


# Model performance metrics
data.frame(
  RMSE = RMSE,
  Rsquared = R_squared
)

}

# predictions_1 <- predict(mod_lasso, s = bestlam1, newx = Xt)
```

```
# length(crime_test)
# eval_results(y, predictions_1, crime_test) ## RMSE: 0.2260244 ; R-squared: 0.6890346
# predictions_2 <- predict(mod_lasso, s = bestlam2, newx = Xt)
# eval_results(y, predictions_2, crime_test) ## RMSE: 0.2421906; R-squared: 0.6429609
r2_1 = cor(p_lasso_min, crime_test$ViolentCrimesPerPop)^2 # 0.648
r2_2 = cor(p_lasso_1se, crime_test$ViolentCrimesPerPop)^2 # 0.627

# coefs <- as.matrix(mod_lasso$beta)
# par(mfrow=c(3,5), mar=c(1,1,1,1))
# for (i in 1:nrow(coefs))
#   plot(coefs[i,], type = "l")
```

Which model is better? Does it depend on the performance measure that is used?

According to the first prediction performance validating by computing MSE, the second model using "lambda.1se" is better with a slightly lower MSE of 0.08268388. The second prediction performance metric composed of RMSE and R-squared, contrarily, shows a better performance of the first model with "lambda.min" comparing to the second model using "lambda.1se," as a smaller RMSE and a higher R-squared are shown. Based on the performance metric I choose and the assigned one, the results depend on the performance measure that is used.

end

Revised-the first model is better with a greater R-squared and a smaller MSE