Large Data, Sparsity and LASSO

Objectives

- Linear model with least squared estimates can be easily produced and interpreted. It often works well for the purpose of prediction.
- However, when there are many predictors, it is hard to find a set of "important predictors".
- In addition, when the number of predictors p is larger than the number of observations n, we cannot estate all the coefficients.
- In this lecture we introduce LASSO (Least Absolute Shrinkage and Selection Operator) to produce a sparse model. One may view LASSO as a model selection scheme.
- `K-Fold` Cross Validation Errors will be introduced and used.

Objectives

0. Suggested reading

   - Section 5.1.1 to 5.1.4 (easy reading to have an idea about k-fold cross validation)
   - Data set: `CrimeData_FL.csv`,`CrimeData_clean.csv` and `CrimeData.csv`

1. Case Study: How to reduce Crime Rates in the United States?

   - Large data: High dimensional data
   - US Heatmap

2. LASSO (Least Absolute Shrinkage and Selection Operator)
3. K-Fold Cross Validation (Appendix V)

   - Introduce cross validation
   - Estimate prediction errors

Objectives

4. Package `elasticnet`
   - `glmnet()`
   - `cv.glmnet()`
5. Final Model
   - Backward Selection `lm`
6. Heatmap
   - `usmap`
   - `plotly`
7. Summary

Case study: violent crime rates

- Case: Violent crimes and other type of crimes can raise broad concerns among residents in a city or a community. It has great impacts on our daily life. It will be helpful for both residents and policy makers to identify factors that are related to crime rates.
- Goal:
  - identify factors that are highly related to crimerates
  - provide possible actions to policy makers to reduce crime rates
  - provide suggestions for people on where to live safely
- Data:
  - Crimes and other information about the population, the police enforcement for almost all the states
  - 147 variables
    - ★ 18 are various crimes in 1995 for communities in US
    - ★ socio-economic information
    - ★ law enforcementdata from 1990
  - response variable: violentcrimes.perpop: violent crimes per 100K people in 1995

Goal and data

Goal of the study:

- Find important factors related to `violentcrimes.perpop` in Florida.
- Build the best possible linear prediction equation to predict `violentcrimes.perpop`.

Crime Data and EDA

For the sake of coherence, we move the detailed EDA into Appendix. We highlight the data and the nature of the study here.

- `violentcrimes.perpop` is the response variable
- We clearly should not use other crime rates as predictors
- Due to too many missing values, we exclude information for police divisions
- We finally have `n`=90 communities with `p`=97 variables.

Data

Data sets available to us:

- `CrimeData_FL.csv`: a subset of `CrimeData_clean.csv` only for Florida
- `CrimeData.csv`: original data (2215 by 147)
- `CrimeData_clean.csv`: eliminated some variables and no missing values (1994 by 99)

Read `CrimeData_FL.csv`

Notice that there are only 90 observations or `communities` but 97 predictors.

```
data.fl <- read.csv("data/CrimeData_FL.csv", header=T, na.string=c("", "?"))
dim(data.fl)
```

```
## [1] 90 98
```

Data

```
names(data.fl) # sum(is.na(data.fl)); summary(data.fl)
```

```
##  [1] "population"                "household.size"
##  [3] "race.pctblack"             "race.pctwhite"
##  [5] "race.pctasian"             "race.pcthisp"
##  [7] "age.pct12to21"             "age.pct12to29"
##  [9] "age.pct16to24"             "age.pct65up"
## [11] "pct.urban"                 "med.income"
## [13] "pct.wage.inc"              "pct.farmself.inc"
## [15] "pct.inv.inc"               "pct.socsec.inc"
## [17] "pct.pubasst.inc"           "pct.retire"
## [19] "med.family.inc"            "percap.inc"
## [21] "white.percap"              "black.percap"
## [23] "indian.percap"             "asian.percap"
## [25] "hisp.percap"               "pct.pop.underpov"
## [27] "pct.less9thgrade"          "pct.not.hsgrad"
## [29] "pct.bs.ormore"             "pct.unemployed"
## [31] "pct.employed"              "pct.employed.manuf"
## [33] "pct.employed.profserv"     "pct.occup.manuf"
## [35] "pct.occup.mgmtprof"        "male.pct.divorce"
## [37] "male.pct.nvrmarried"       "female.pct.divorce"
## [39] "total.pct.divorce"         "ave.people.per.fam"
## [41] "pct.fam2parents"           "pct.kids2parents"
## [43] "pct.youngkids2parents"     "pct.teens2parents"
## [45] "pct.workmom.youngkids"     "pct.workmom"
## [47] "num.kids.nvrmarried"       "pct.kids.nvrmarried"
## [49] "num.immig"                 "pct.immig.recent"
## [51] "pct.immig.recent5"         "pct.immig.recent8"
## [53] "pct.immig.recent10"        "pct.pop.immig"
## [55] "pct.pop.immig5"            "pct.pop.immig8"
## [57] "pct.pop.immig10"           "pct.english.only"
## [59] "pct.no.english.well"       "pct.fam.hh.large"
## [61] "pct.occup.hh.large"        "ave.people.per.hh"
```

Heatmap

- A heatmap is useful when the data has geographical information.
- In this section, we create a heat map to display summary statistics at the state level.
- Let us take a look at a few statistics: the mean of income: med.income. We will display some statistics by states in a heat map.

Heatmap: first 'summarise()'

We first extract the mean of med.income, mean crime rate by state among other statistics. n=number of the obs'n in each state.

Remark: Regarding mean crime rate by state, merely taking mean in the following chunk is not correct due to different county level population sizes. Can you create **proper average crime rate per 100k** for each state first.

```
crime.data <- read.csv("data/CrimeData.csv", header=T, na.string=c("", "?"))
data.s <- crime.data %>%
  group_by(state) %>%
  summarise(
    mean.income=mean(med.income),
    income.min=min(med.income),
    income.max=max(med.income),
    crime.rate=mean(violentcrimes.perpop, na.rm=TRUE), #ignore the missing values
    n=n())
```

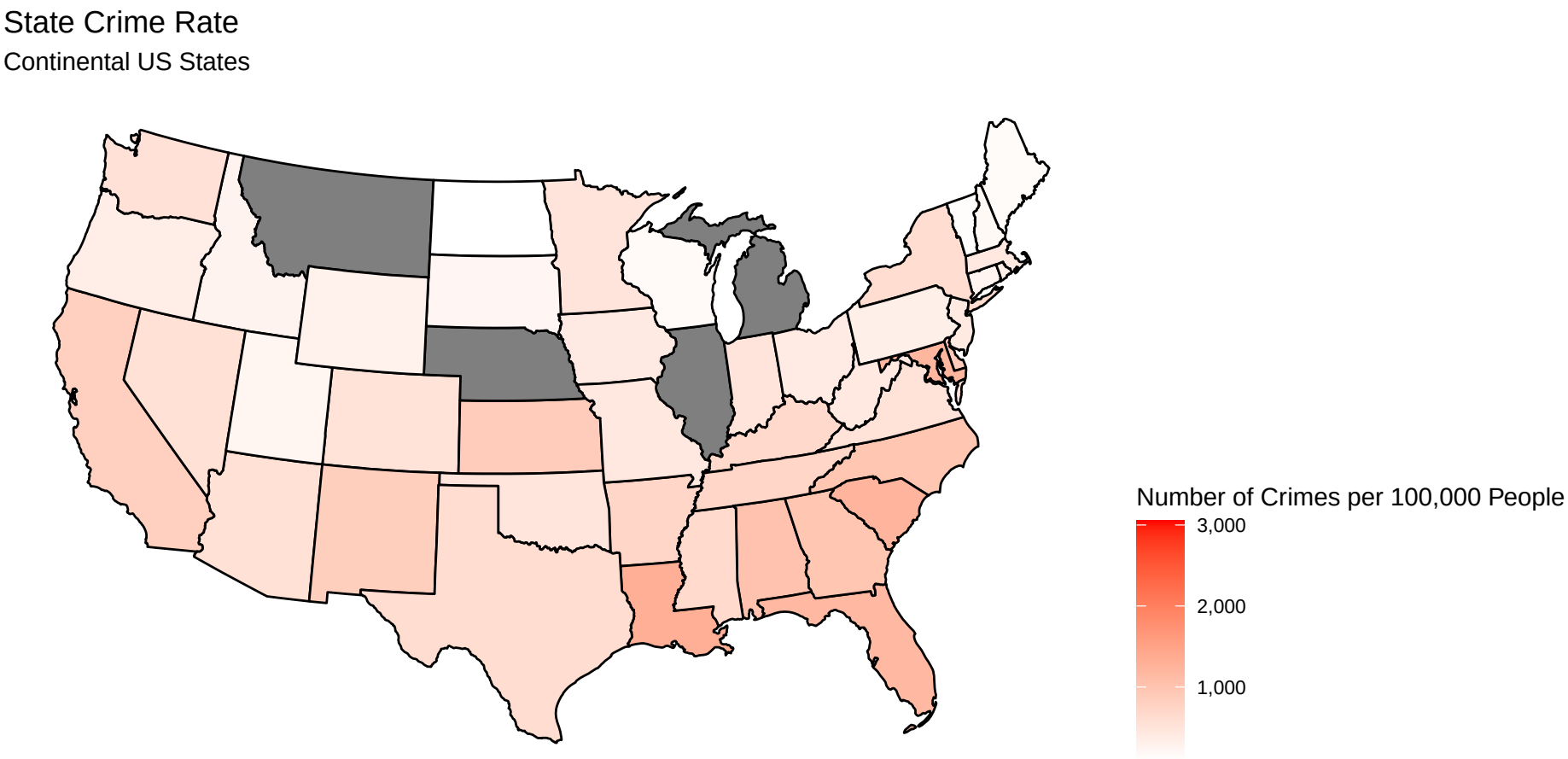
Heatmap: then ‘usmap::plot_usmap()‘

Use `usmap` library:

```
library(usmap)
#names(data.s)
#levels(data.s$state)

stat.crime.plot <- plot_usmap(regions = "state",
                              #regions = "counties", for county level summary
    data = data.s,
    values = "crime.rate", exclude = c("Hawaii", "Alaska"), color = "black") +
    scale_fill_gradient(
      low = "white", high = "red",
      name = "Number of Crimes per 100,000 People",
      label = scales::comma) +
    labs(title = "State Crime Rate", subtitle = "Continental US States") +
    theme(legend.position = "right")
```

Heatmap: ‘plot_usmap()‘

```
stat.crime.plot
```



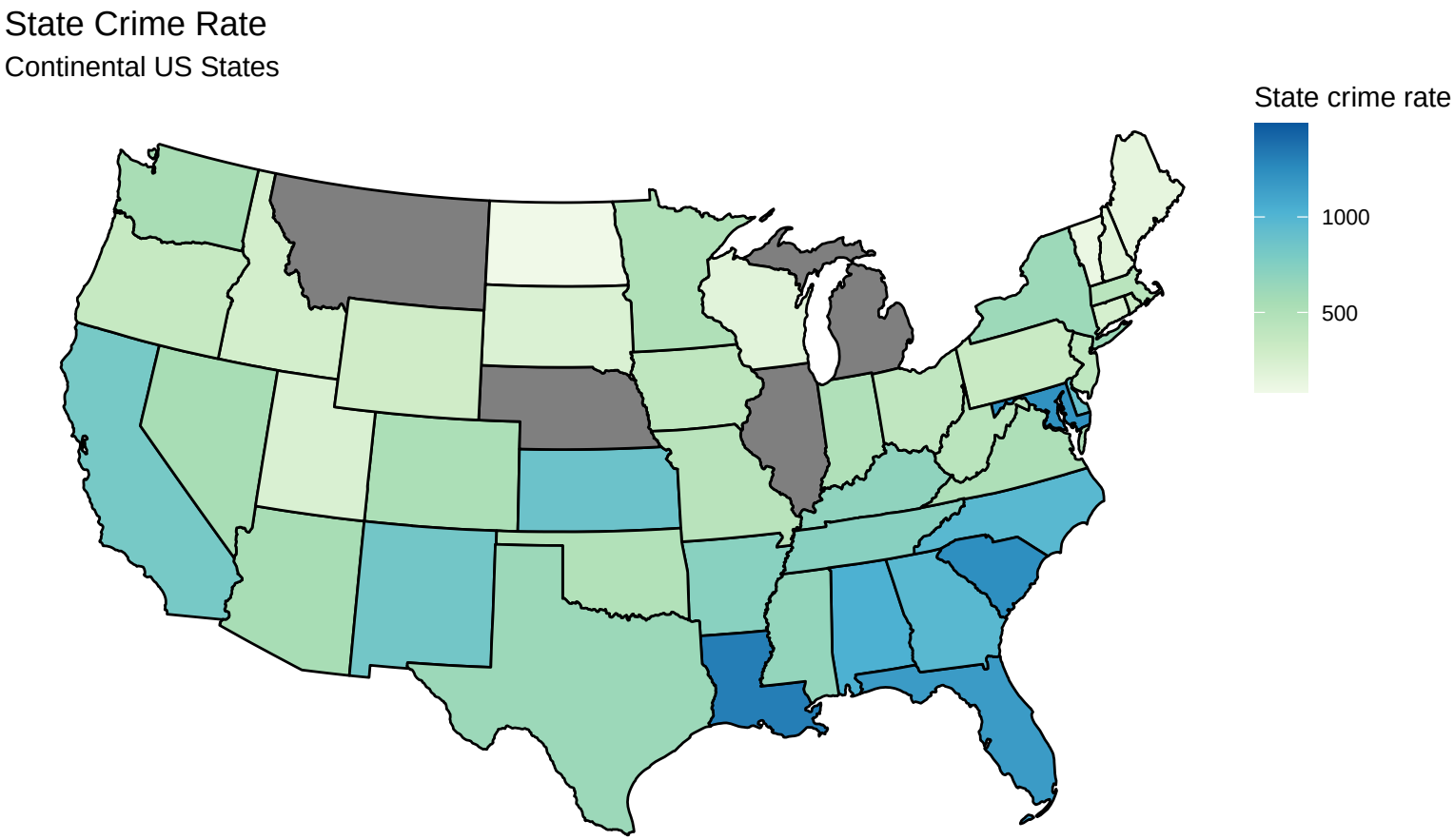
Heatmap: ‘plot_usmap()‘

Notice better corlor variation

```
# set color range limits
max_crime_col <- quantile(data.s$crime.rate, .98, na.rm = T)
min_crime_col <- quantile(data.s$crime.rate, 0, na.rm = T)

stat.crime.plot <- plot_usmap(regions = "state",
    data = data.s,
    values = "crime.rate", exclude = c("Hawaii", "Alaska"), color = "black") +
    scale_fill_distiller(palette = "GnBu",
                         direction = 1,
                         # limits sets range of color
                         name = "State crime rate",
                         limits = c(min_crime_col, max_crime_col)) +
    labs(title = "State Crime Rate", subtitle = "Continental US States") +
    theme(legend.position = c(1,.6))
```

Heatmap

```
stat.crime.plot
```




```
# ggplotly(stat.crime.plot +
#            theme(legend.position = "none"))
```

Heatmap

Remarks:

- Color control: `scale_fill_distiller()` controls the range of the coloring. It is important that we pay attention to the contrast of coloring. In our case, maximum `crime rate` dominates the rest of numbers. Without controlling the color range, the heat map will be more or less the same color.
- `ggplotly`: Applying `ggplotly` in the hope that we can hover on the state to show the statistics.

Heatmap

Above is the result of our new map showing the Mean Income. This gives a quick view of the income disparities by state.

- The two states with no values are marked as grey, without a state name.
- As expected the east/west have higher incomes.
- While this gives us a good visualization about income by state, does this agree with reality? Do you trust this EDA (Nothing is wrong with the map. Rather you may look into the sample used here.)
- Very cool plots. You can make heat maps with other summary statistics.

Linear model with all variables

We will use linear model to identify important factors related to `violentcrimes.perpop`. For a quick result let us run a linear model with all the variables.

```
fit.fl.lm <- lm(violentcrimes.perpop~., data.fl) # dump everything in the model
summary(fit.fl.lm)  #Anova(fit.fl.lm)
```

```
##
## Call:
## lm(formula = violentcrimes.perpop ~ ., data = data.fl)
##
## Residuals:
## ALL 90 residuals are 0: no residual degrees of freedom!
##
## Coefficients: (8 not defined because of singularities)
##                                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)                     3.86e+05         NA      NA       NA
## population                     -5.22e-03         NA      NA       NA
## household.size                 -2.46e+03         NA      NA       NA
## race.pctblack                   5.16e+01         NA      NA       NA
## race.pctwhite                   1.97e+02         NA      NA       NA
## race.pctasian                  -6.64e+02         NA      NA       NA
## race.pcthisp                   -1.40e+02         NA      NA       NA
## age.pct12to21                   2.22e+02         NA      NA       NA
## age.pct12to29                   5.23e+02         NA      NA       NA
## age.pct16to24                  -3.67e+02         NA      NA       NA
## age.pct65up                    -8.24e+01         NA      NA       NA
## pct.urban                       2.32e+00         NA      NA       NA
## med.income                     -6.44e-02         NA      NA       NA
## pct.wage.inc                    3.82e+02         NA      NA       NA
## pct.farmself.inc               -4.47e+02         NA      NA       NA
```

Linear model with all variables: problems

We immediately notice

- **We can not estimate all the coefficients**.
- **We can not make inferences for any coefficients where the estimates exist**.

Linear model with all variables: why and solution

- The reason that the least squared solutions does not exist for all the coefficients is that the number of unknown parameters is larger than the sample size.
- Even we have lots of communities in the data, we expect that many coefficients are not significant. So it will be hard to identify a set of important factors related to the violent crime rates.
- One way out is to impose sparsity. Suppose there are only a small collection of factors affecting the response, we can then apply restrictions on the coefficients to look for a constraint least squared solution via regularization.

Linear model with all variables

Consider linear multiple regression models with p predictors

$$Y = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}$$

The OLS may

- Overfit the data
- When p is large there are no unique solution for OLS
- A smaller model is preferable for interpretation

One way to avoid the above problems is to add constraints on the coefficients.

LASSO regularization

LASSO (Least Absolute Shrinkage and Selection Operator) produces a restricted least squared solution by adding the following constraints over unknown parameters.

$$\min_{\substack{\beta_0,\beta_1,\beta_2,\ldots,\beta_p \\ |\beta_1|+|\beta_2|+\cdots+|\beta_p|\leq t}} \left\{ \frac{1}{2n} \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_{i1} - \beta_2 x_{i2} - \cdots - \beta_p x_{ip})^2 \right\}$$

Remark on LASSO solutions:

- $|\beta_1| + |\beta_2| + \cdots + |\beta_p|$ is called L-1 penalty
- t is called the tuning parameter which control the sparsity
- The solution $\hat{\beta}_i^t$ depends on t
- $\hat{\beta}_i^t$ are OLS when $t \to \infty$ and
- $\hat{\beta}_i^t = 0$ for some i when t is small

Which tuning parameter t to use?

L1 Penalty

The above restricted least squared problem motivates the sparse solution to choose a small set of variables. It is not easy to find the solution. Equivalently we can solve the following problem:

$$\min_{\beta_0,\,\beta_1,\,\beta_2,\ldots,\beta_p} \left\{ \frac{1}{2n} \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_{i1} - \beta_2 x_{i2} - \cdots - \beta_p x_{ip})^2 + \lambda(|\beta_1| + |\beta_2| + \cdots + |\beta_p|) \right\}$$

Remark on LASSO:

- The two minimization problems are equivalent.
- λ is called the tuning parameter
- The solution $\hat{\beta}_i^{\lambda}$ depends on λ
- $\hat{\beta}_i^{\lambda}$ are OLS when $\lambda = 0$ and
- $\hat{\beta}_i^{\lambda} = 0$ when $\lambda \to \infty$

K-fold cross validation

Given a λ we will have a set of solutions for all the $\beta_i's$. We then have a prediction equation. If we have another testing data, we can then estimate the prediction error. The simplest way to estimate prediction errors with a training set is called K-Fold Cross Validation.

K-fold cross validation

To compute K-fold cross validation error, we use the following algorithm.

1. We split our data into K sections, called folds. So if our training set has 1000 observations, we can set $K = 10$ & randomly split our data set into 10 different folds each containing 100 observations.
2. We then train a LASSO model on all the folds except 1, i.e. we train the model on 9 out of the 10 folds.
3. Next, we produce fitted values for all points in the fold that was **NOT** used to train the model on, and then we store the MSE on that fold.
4. We repeat this procedure for each fold. So each fold is left out of the training set once and used to test the model on. This means that we will have K MSE values, one for each fold.
5. We then average these MSE values, and this gives us an estimate of the testing error.

‘glmnet()‘ and ‘cv.glmnet()‘

glmnet

glmnet is a well-developed package to produce LASSO estimates for each λ value together with K-fold prediction errors. It has two main functions, glmnet() and cv.glmnet(). Using the glmnet we will go through in details the LASSO solution to identify important factors related to ViolentCrime in Florida

'glmnet()': Data Preparation

Prepare the input X matrix and the response Y. `glmnet()` requires inputting the design matrix $X = (x_1, ...x_p)$ and the response variable Y.

```
library(glmnet)
Y <- data.fl[, 98] # extract Y
X.fl <- model.matrix(violentcrimes.perpop~., data=data.fl)[, -1]
# get X variables as a matrix. it will also code the categorical
# variables correctly!. The first col of model.matrix is vector 1
# dim(X.fl)
```

'glmnet()': Data Preparation

```
colnames(X.fl)   # X.fl[1:2, 1:5]
```

```
##  [1] "population"                "household.size"
##  [3] "race.pctblack"             "race.pctwhite"
##  [5] "race.pctasian"             "race.pcthisp"
##  [7] "age.pct12to21"             "age.pct12to29"
##  [9] "age.pct16to24"             "age.pct65up"
## [11] "pct.urban"                 "med.income"
## [13] "pct.wage.inc"              "pct.farmself.inc"
## [15] "pct.inv.inc"               "pct.socsec.inc"
## [17] "pct.pubasst.inc"           "pct.retire"
## [19] "med.family.inc"            "percap.inc"
## [21] "white.percap"              "black.percap"
## [23] "indian.percap"             "asian.percap"
## [25] "hisp.percap"               "pct.pop.underpov"
## [27] "pct.less9thgrade"          "pct.not.hsgrad"
## [29] "pct.bs.ormore"             "pct.unemployed"
## [31] "pct.employed"              "pct.employed.manuf"
## [33] "pct.employed.profserv"     "pct.occup.manuf"
## [35] "pct.occup.mgmtprof"        "male.pct.divorce"
## [37] "male.pct.nvrmarried"       "female.pct.divorce"
## [39] "total.pct.divorce"         "ave.people.per.fam"
## [41] "pct.fam2parents"           "pct.kids2parents"
## [43] "pct.youngkids2parents"     "pct.teens2parents"
## [45] "pct.workmom.youngkids"     "pct.workmom"
## [47] "num.kids.nvrmarried"       "pct.kids.nvrmarried"
## [49] "num.immig"                 "pct.immig.recent"
## [51] "pct.immig.recent5"         "pct.immig.recent8"
## [53] "pct.immig.recent10"        "pct.pop.immig"
## [55] "pct.pop.immig5"            "pct.pop.immig8"
## [57] "pct.pop.immig10"           "pct.english.only"
## [59] "pct.no.english.well"       "pct.fam.hh.large"
## [61] "pct.occup.hh.large"        "ave.people.per.hh"
```

‘glmnet()‘ with fixed λ

We first run `glmnet()` with $\lambda = 100$. From the output, we can see that the features selected by LASSO. Read and run the following code line by line to understand the output. We provide detailed comments inside the following R-chunk to get familiar with function `glmnet`

```
fit.fl.lambda <- glmnet(X.fl, Y, alpha=1, lambda = 100) # alpha =1 corresponding to LASSO solutions
names(fit.fl.lambda) # to see the possible output
```

```
##  [1] "a0"        "beta"      "df"        "dim"       "lambda"
##  [6] "dev.ratio" "nulldev"   "npasses"   "jerr"      "offset"
## [11] "call"      "nobs"
```

```
fit.fl.lambda$lambda # lambda used
```

```
## [1] 100
```

'glmnet()': coefficients

```
tmp_coeffs <- fit.fl.lambda$beta
# The coefficients are functions of lambda. It only outputs the coefs of features.
# Notice many of the coef's are 0
fit.fl.lambda$df    # number of non-zero coeff's
```

```
## [1] 9
```

```
fit.fl.lambda$a0    # est. of beta_0
```

```
##   s0
## 3197
```

'glmnet()': coefficients

```
# An intuitive way to extract LASSO solutions
coef(fit.fl.lambda) # beta hat of predictors in the original scales, same as fit.fl.lambda$beta
```

```
## 98 x 1 sparse Matrix of class "dgCMatrix"
##                                       s0
## (Intercept)                     3196.7501
## population                         .
## household.size                     .
## race.pctblack                      .
## race.pctwhite                      .
## race.pctasian                      .
## race.pcthisp                       .
## age.pct12to21                      .
## age.pct12to29                      .
## age.pct16to24                      .
## age.pct65up                        .
## pct.urban                          .
## med.income                         .
## pct.wage.inc                       .
## pct.farmself.inc                   .
## pct.inv.inc                        .
## pct.socsec.inc                     .
## pct.pubasst.inc                    .
## pct.retire                         .
## med.family.inc                     .
## percap.inc                         .
## white.percap                       .
## black.percap                       .
## indian.percap                      .
## asian.percap                       .
## hisp.percap                        .
## pct.pop.underpov                 25.5200
## pct.less9thgrade                   .
```

'glmnet()': Non-zero coefficients

```
# more advanced way to extract non-zero output using sparse matrix `dgCMatrix`
tmp_coeffs <- coef(fit.fl.lambda)  # output the LASSO estimates
tmp_coeffs.min <- tmp_coeffs[which(tmp_coeffs!=0),]   # get the non=zero coefficients
tmp_coeffs.min # the set of predictors chosen
```

```
##           (Intercept)      pct.pop.underpov      male.pct.divorce
##             3196.7501               25.5200               28.2519
##      pct.kids2parents pct.youngkids2parents   num.kids.nvrmarried
##               -5.5490               -2.5848                0.0137
##   pct.kids.nvrmarried   pct.people.dense.hh    med.yr.house.built
##               83.2388               13.0108               -1.2945
##     pct.house.nophone
##                1.1880
```

```
rownames(as.matrix(tmp_coeffs.min)) # shows only names, not estimates
```

```
##  [1] "(Intercept)"           "pct.pop.underpov"
##  [3] "male.pct.divorce"      "pct.kids2parents"
##  [5] "pct.youngkids2parents" "num.kids.nvrmarried"
##  [7] "pct.kids.nvrmarried"   "pct.people.dense.hh"
##  [9] "med.yr.house.built"    "pct.house.nophone"
```

‘glmnet()‘ with fixed λ

When $\lambda = 100$, we return 9 variables with non-zero coefficient.

- What is the testing error or `K-fold` cross validation error for the linear model using the above variables selected?
- How to find the `lambda` with smallest cross validation error or how to control the sparsity among the models with similar cross validation errors?

'glmnet()' with a set of λ's

LASSO path: estimators for a set of λ's

Now glmnet will output results for 100 different λ values suggested by the algorithm. The output will consist of LASSO fits, one for each λ.

```
fit.fl.lambda <- glmnet(X.fl, Y, alpha=1)
str(fit.fl.lambda)
```

```
## List of 12
##  $ a0       : Named num [1:100] 1159 1119 1081 1044 1007 ...
##   ..- attr(*, "names")= chr [1:100] "s0" "s1" "s2" "s3" ...
##  $ beta     :Formal class 'dgCMatrix' [package "Matrix"] with 6 slots
##   .. ..@ i       : int [1:1494] 47 47 47 25 47 25 47 25 47 25 ...
##   .. ..@ p       : int [1:101] 0 0 1 2 3 5 7 9 11 13 ...
##   .. ..@ Dim     : int [1:2] 97 100
##   .. ..@ Dimnames:List of 2
##   .. .. ..$ : chr [1:97] "population" "household.size" "race.pctblack" "race.pctwhite" ...
##   .. .. ..$ : chr [1:100] "s0" "s1" "s2" "s3" ...
##   .. ..@ x       : num [1:1494] 10.143 19.825 29.067 0.438 37.089 ...
##   .. ..@ factors : list()
##  $ df       : int [1:100] 0 1 1 1 2 2 2 2 2 2 ...
##  $ dim      : int [1:2] 97 100
##  $ lambda   : num [1:100] 711 679 648 618 590 ...
##  $ dev.ratio: num [1:100] 0 0.0664 0.1269 0.1821 0.2332 ...
##  $ nulldev  : num 60854001
##  $ npasses  : int 1785
##  $ jerr     : int 0
##  $ offset   : logi FALSE
##  $ call     : language glmnet(x = X.fl, y = Y, alpha = 1)
##  $ nobs     : int 90
##  - attr(*, "class")= chr [1:2] "elnet" "glmnet"
```

‘cv.glmnet()‘: choose λ via CV

`cv.glmnet()`: Cross Validation to select a λ

To compare the Cross validation errors for the set of `lambda`, we use `cv.glmnet()`. For each λ, it will report the K fold CV errors together with some summary statistics among the K errors:

Cross-validation over λ gives us a set of errors:

- `cvm`: mse among the K cv errors
- `cvsd`: estimate of standard error of `cvm`,
- `cvlo`: `cvm - cvsd`
- `cvup`: `cvm + cvsd`

‘cv.glmnet()‘

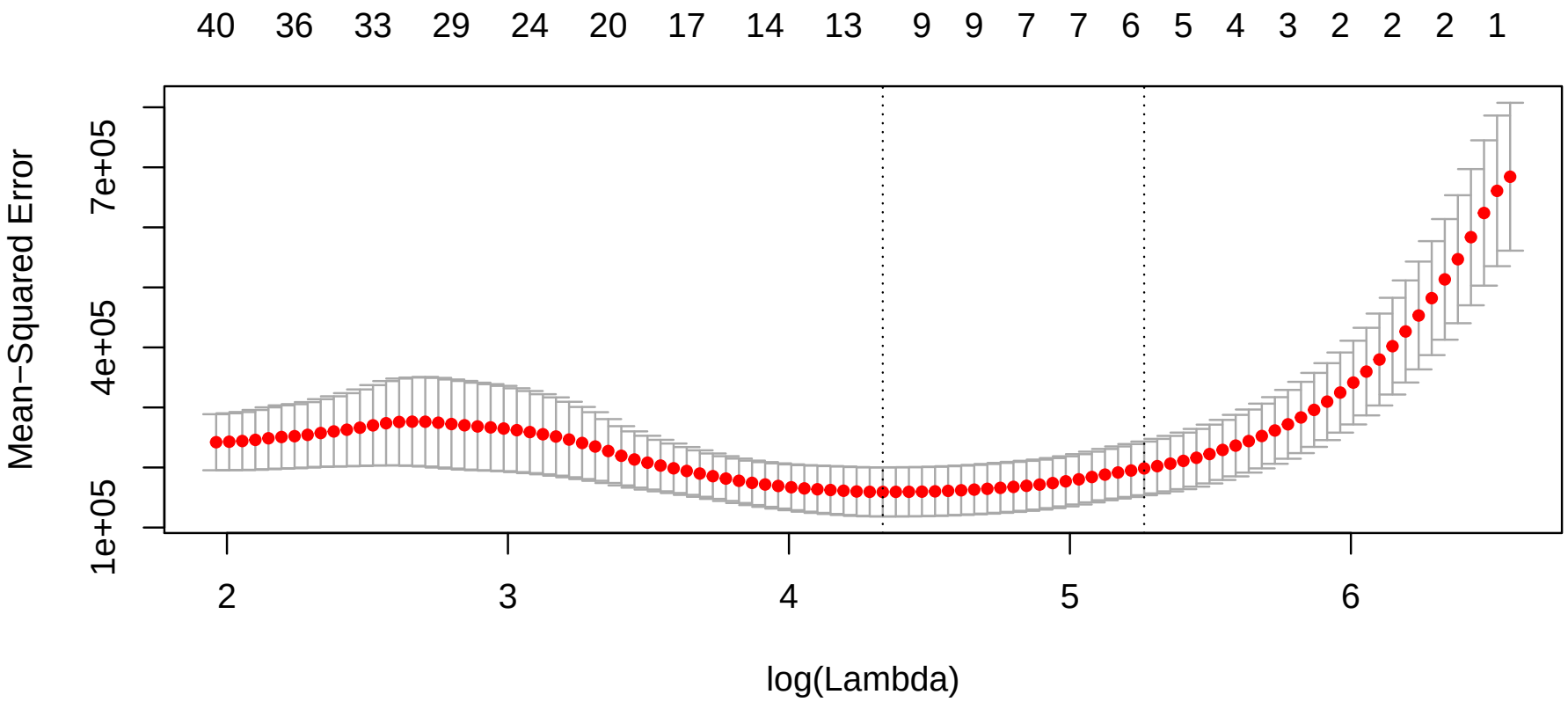
Read through the following R-chunk to get familiar with `cv.glmnet()`.

```
Y <- data.fl[, 98] # extract Y
X.fl <- model.matrix(violentcrimes.perpop~., data=data.fl)[, -1] # get design matrix without the first col of 1
set.seed(10)  # to control the randomness in K folds
fit.fl.cv <- cv.glmnet(X.fl, Y, alpha=1, nfolds=10)

# names(fit.fl.cv); summary(fit.fl.cv)
# fit.fl.cv$cvm               # the mean cv error for each lambda
# fit.fl.cv$lambda.min # lambda.min returns the min point amoth all the cvm.
# fit.fl.cv$lambda.1se
# fit.fl.cv$nzero     # string of number of non-zero coeff's returned for each lambda
```

'cv.glmnet()'

We can now use the default plot to explore possible values of lambda:

```
plot(fit.fl.cv)    # fit.fl.cv$lambda.min; fit.fl.cv$lambda.1s
```



The default graph combines all the plots we just looked at in detail.

‘cv.glmnet()‘

Plot Description:

- Top margin: number of nonzero β's for each λ
- Red points: cvm, the mean cross validation error for each λ
- Vertical bar around each λ: cvm (red point) +/- 1 cvsd
- First vertical line: lambda.min, λ which gives the smallest cvm
- Second vertical line: lambda.1se, the largest λ whose cvm is within the cvsd bar for the lambda.min value.

‘cv.glmnet()‘

Important Remarks:

- Cross validation errors vary a lot.
- The above plot changes as we run the function again with different K-folds

Which model to use?

Because of the large variability in CV errors, minimizing the `cvm` will not guarantee the model the smallest prediction error.

- One may choose any lambda between lambda.min and lambda.1se
- One may also choose lambda controlled by `nzero`, the number of non-zero elements

'cv.glmnet()': lambda.min

Output variables for the λ chosen

Once a specific λ is chosen, we will output the corresponding predictors.

1. Output β's from lambda.min, as an example

```
coef.min <- coef(fit.fl.cv, s="lambda.min")  #s=c("lambda.1se","lambda.min") or lambda value
coef.min <- coef.min[which(coef.min !=0),]   # get the non=zero coefficients
coef.min  # the set of predictors chosen
```

```
##           (Intercept)         indian.percap      pct.pop.underpov
##              7.51e+03              6.08e-04              2.32e+01
##      male.pct.divorce      pct.kids2parents pct.youngkids2parents
##              2.85e+01             -5.15e+00             -3.20e+00
##   num.kids.nvrmarried   pct.kids.nvrmarried   pct.people.dense.hh
##              1.73e-02              8.48e+01              1.33e+01
##    med.yr.house.built     pct.house.nophone    pct.house.no.plumb
##             -3.48e+00              6.22e+00              6.83e+00
```

```
rownames(as.matrix(coef.min)) # shows only names, not estimates
```

```
##  [1] "(Intercept)"           "indian.percap"
##  [3] "pct.pop.underpov"      "male.pct.divorce"
##  [5] "pct.kids2parents"      "pct.youngkids2parents"
##  [7] "num.kids.nvrmarried"   "pct.kids.nvrmarried"
##  [9] "pct.people.dense.hh"   "med.yr.house.built"
## [11] "pct.house.nophone"     "pct.house.no.plumb"
```

'cv.glmnet()': lambda.1se

2. Output β's from lambda.1se (this way you are using smaller set of variables.)

```
coef.1se <- coef(fit.fl.cv, s="lambda.1se")
coef.1se <- coef.1se[which(coef.1se !=0),]
coef.1se
```

```
##          (Intercept)    pct.pop.underpov    male.pct.divorce
##            550.74319            25.38939            15.69568
##    pct.kids2parents num.kids.nvrmarried pct.kids.nvrmarried
##             -4.26350             0.00117            89.17276
## pct.people.dense.hh
##              6.03319
```

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

```
## [1] "(Intercept)"         "pct.pop.underpov"    "male.pct.divorce"
## [4] "pct.kids2parents"    "num.kids.nvrmarried" "pct.kids.nvrmarried"
## [7] "pct.people.dense.hh"
```

'cv.glmnet()': number of non-zero

3. Choose the number of non-zero coefficients

Suppose you want to use λ such that it will return 9 n-zero predictors.

```
coef.nzero <- coef(fit.fl.cv, nzero = 9)  #fit.fl.cv$nzero
coef.nzero <- coef.nzero[which(coef.nzero !=0), ]
rownames(as.matrix(coef.nzero)) #notice nzero may not take any integer.
```

'cv.glmnet()': 's' value

4. We may specify the s value ourselves. We may want to use a number between lambda.min and lambda.1se, say we take $s = e^{4.6}$.

```
coef.s <- coef(fit.fl.cv, s=exp(4.6))
#coef.s <- coef(fit.fl.cv, s=fit.fl.cv$lambda[3])
coef.s <- coef.s[which(coef.s !=0),]
coef.s
```

```
##           (Intercept)      pct.pop.underpov      male.pct.divorce
##             3257.6097               25.4120               28.2125
##      pct.kids2parents pct.youngkids2parents   num.kids.nvrmarried
##               -5.8210               -2.4171                0.0137
##   pct.kids.nvrmarried   pct.people.dense.hh    med.yr.house.built
##               83.1364               13.1355               -1.3225
##     pct.house.nophone
##                1.2367
```

```
var.4.6 <- rownames(as.matrix(coef.s))
var.4.6
```

```
##  [1] "(Intercept)"           "pct.pop.underpov"
##  [3] "male.pct.divorce"      "pct.kids2parents"
##  [5] "pct.youngkids2parents" "num.kids.nvrmarried"
##  [7] "pct.kids.nvrmarried"   "pct.people.dense.hh"
##  [9] "med.yr.house.built"    "pct.house.nophone"
```

5 Final model

- LASSO produces a sparse solution. But it does not provide inference for variables selected by itself. + In our Crime Data study we combine LASSO results and feed the variables to linear model to have a final model. This is called `Relaxed LASSO`. Let us recap the LASSO process, continuing to build a final model.

LASSO equation

First get the LASSO output. As an example we decide to use the `lambda.min`. We will have the following variables chosen together with the LASSO equation

```
#Step 1: Prepare design matrix
Y <- data.fl[, 98] # extract Y
X.fl <- model.matrix(violentcrimes.perpop~., data=data.fl)[, -1] # take the first column's of 1 out
#Step 2: Find x's output from LASSO with min cross-validation error
set.seed(10)  # to control the ramdomness in K folds
fit.fl.cv <- cv.glmnet(X.fl, Y, alpha=1, nfolds=10, intercept = T)
coef.min <- coef(fit.fl.cv, s="lambda.min")  #s=c("lambda.1se","lambda.min") or lambda value
coef.min <- coef.min[which(coef.min !=0),]   # get the non=zero coefficients
var.min <- rownames(as.matrix(coef.min))[-1] # output the names  dim(as.matrix(coef.min))
```

LASSO equation

Alternatively, we can get the LASSO non-zero variables through sparse matrix:

```
# Another faster way to get the LASSO output
coef.min <- coef(fit.fl.cv, s="lambda.min") #s=c("lambda.1se","lambda.min") or lambda value
var.min <- coef.min@Dimnames[[1]][coef.min@i + 1][-1] # we listed those variables with non-zero estimates. [-1]
```

Force in

- Sometimes we may want to lock some variables of interests in the linear model.
- One way to achieve this goal is to not impose any penalty on the corresponding coefficients. We can force in these variables using the `penalty.factor` argument in `cv.glmnet`. `penalty.factor` takes a vector of length as the number of variables in the model matrix.
- For example, if we think population has to be included in the final model, then set the corresponding element of `penalty.factor` as 0 while keeping others as 1.
- Here population is the 1st variable, so we feed a vector `c(0, rep(1, ncol(X.fl)-1))` to `penalty.factor`.

Force in

```
fit.fl.force.cv <- cv.glmnet(X.fl, Y, alpha=1, nfolds=10, intercept = T,
                        penalty.factor = c(0, rep(1, ncol(X.fl)-1)))
coef.force.min <- coef(fit.fl.force.cv, s="lambda.min")
var.force.min <- coef.force.min@Dimnames[[1]][coef.force.min@i + 1][-1]
var.force.min
```

```
##  [1] "population"            "race.pcthisp"
##  [3] "pct.retire"            "indian.percap"
##  [5] "asian.percap"          "pct.pop.underpov"
##  [7] "male.pct.divorce"      "pct.kids2parents"
##  [9] "pct.youngkids2parents" "pct.workmom"
## [11] "pct.kids.nvrmarried"   "med.yr.house.built"
## [13] "pct.house.nophone"     "pct.house.no.plumb"
```

Remark if we want to force in a categorical variable, we need to first look for their corresponding indices in the model matrix, then set the corresponding element in `penalty.factor` as 0.

5.2 Relaxed LASSO

To make inferences using the LASSO chosen variables, we will run the linear model analyses. Assuming all the linear model assumptions are being true.

Relaxed LASSO

```
data.fl.sub <-  data.fl[,c("violentcrimes.perpop", var.min)] # get a subset with response and LASSO output
#names(data.fl.sub)
fit.min.lm <- lm(violentcrimes.perpop~., data=data.fl.sub)  # debiased or relaxed LASSO
summary(fit.min.lm)
```

```
##
## Call:
## lm(formula = violentcrimes.perpop ~ ., data = data.fl.sub)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -882.3 -163.1   -6.9  163.6  978.5
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)             2.03e+04   1.26e+04    1.62   0.1101
## indian.percap           5.63e-03   2.18e-03    2.58   0.0116 *
## pct.pop.underpov        1.78e+01   1.47e+01    1.21   0.2293
## male.pct.divorce        3.18e+01   2.01e+01    1.58   0.1183
## pct.kids2parents        7.94e-01   9.99e+00    0.08   0.9368
## pct.youngkids2parents -7.12e+00   9.16e+00   -0.78   0.4394
## num.kids.nvrmarried     2.92e-02   9.47e-03    3.09   0.0028 **
## pct.kids.nvrmarried     9.49e+01   2.88e+01    3.30   0.0015 **
## pct.people.dense.hh     1.01e+01   1.53e+01    0.66   0.5122
## med.yr.house.built     -1.01e+01   6.35e+00   -1.59   0.1152
## pct.house.nophone       2.28e+01   1.84e+01    1.24   0.2185
## pct.house.no.plumb      7.26e+01   1.13e+02    0.64   0.5222
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 324 on 78 degrees of freedom
## Multiple R-squared:  0.865,  Adjusted R-squared:  0.846
## F-statistic: 45.5 on 11 and 78 DF,  p-value: <2e-16
```

Relaxed LASSO

An alternative way to do the final lm fit. We first prepare a formula for lm function:

```
coef.min <- coef(fit.fl.cv, s="lambda.min")  #s=c("lambda.1se","lambda.min") or lambda value
coef.min <- coef.min[which(coef.min !=0),]   # get the non=zero coefficients
var.min <- rownames(as.matrix(coef.min))[-1] # output the names without intercept
lm.input <- as.formula(paste("violentcrimes.perpop", "~", paste(var.min, collapse = "+")))
# prepare for lm fomulae
lm.input
```

```
## violentcrimes.perpop ~ indian.percap + pct.pop.underpov + male.pct.divorce +
##     pct.kids2parents + pct.youngkids2parents + num.kids.nvrmarried +
##     pct.kids.nvrmarried + pct.people.dense.hh + med.yr.house.built +
##     pct.house.nophone + pct.house.no.plumb
```

Relaxed LASSO

We then fit the linear model with LASSO output variables.

```
fit.min.lm <- lm(lm.input, data=data.fl)  # debiased or relaxed LASSO
summary(fit.min.lm)
```

```
##
## Call:
## lm(formula = lm.input, data = data.fl)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -882.3 -163.1   -6.9  163.6  978.5
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)             2.03e+04   1.26e+04    1.62   0.1101
## indian.percap           5.63e-03   2.18e-03    2.58   0.0116 *
## pct.pop.underpov        1.78e+01   1.47e+01    1.21   0.2293
## male.pct.divorce        3.18e+01   2.01e+01    1.58   0.1183
## pct.kids2parents        7.94e-01   9.99e+00    0.08   0.9368
## pct.youngkids2parents -7.12e+00   9.16e+00   -0.78   0.4394
## num.kids.nvrmarried     2.92e-02   9.47e-03    3.09   0.0028 **
## pct.kids.nvrmarried     9.49e+01   2.88e+01    3.30   0.0015 **
## pct.people.dense.hh     1.01e+01   1.53e+01    0.66   0.5122
## med.yr.house.built     -1.01e+01   6.35e+00   -1.59   0.1152
## pct.house.nophone       2.28e+01   1.84e+01    1.24   0.2185
## pct.house.no.plumb      7.26e+01   1.13e+02    0.64   0.5222
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 324 on 78 degrees of freedom
## Multiple R-squared:  0.865,  Adjusted R-squared:  0.846
```

Relaxed LASSO

Remark: Not all the predictors in the above lm() are significant at .05 level. We will go one more step further to eliminate some insignificant predictors. We could refine the above model to eliminate some variables which are not significant using backward selection as well.

5.3 Fine-tuning

We noticed that some variables are not significant (p-value > 0.05) in our current model. One may use backward selection to fine tune the final model. Let us take the predictors from the above LASSO output and use backward selection to lower the dimension. We fit the current model first. Kick out the variable with highest p-value or smallest t-value and repeat until reaching the model size determined.

Fine-tuning

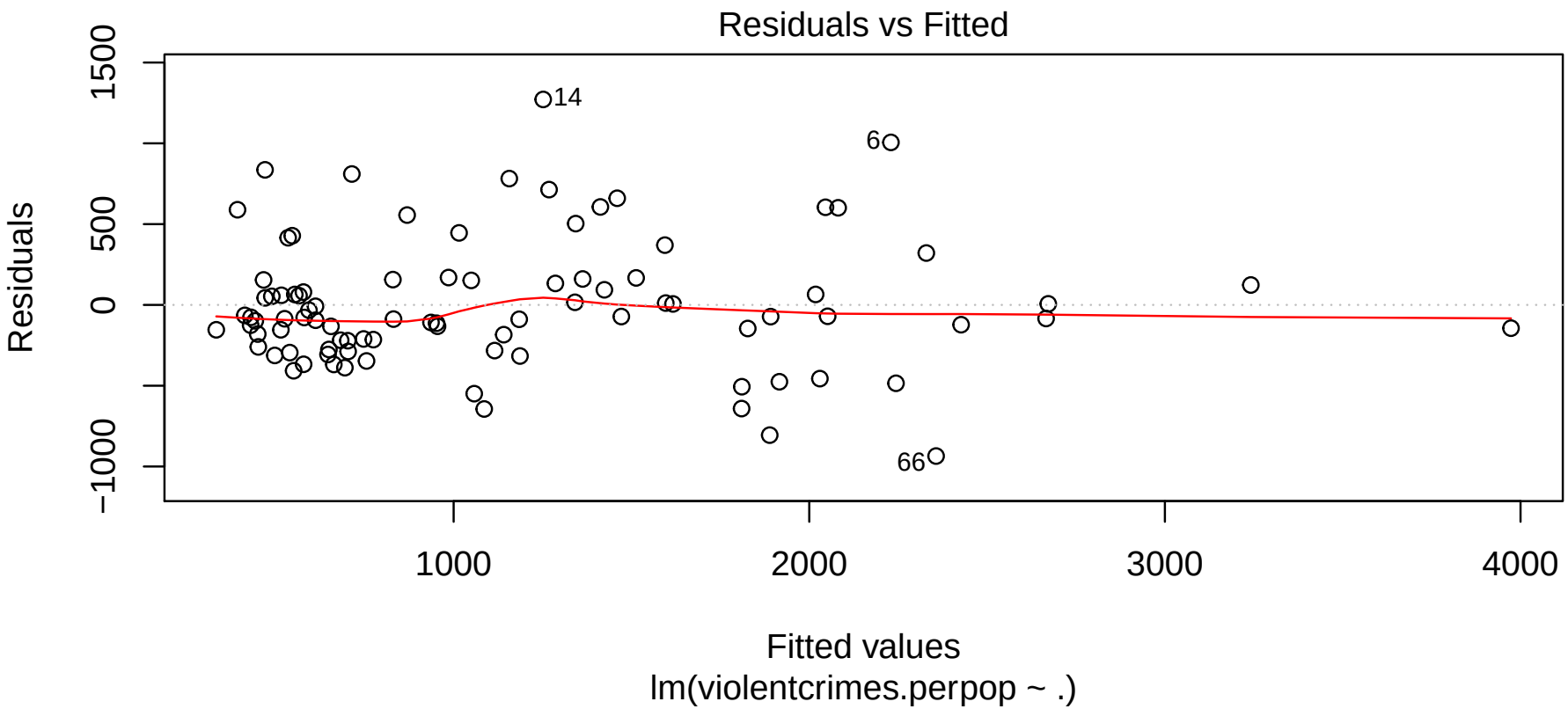
```
data.fl.sub <-  data.fl[,c("violentcrimes.perpop", var.min)] # get a subset with response and LASSO output
var.back <- data.frame(summary(fit.min.lm)$coef[summary(fit.min.lm)$coef[,4] <= .05, 4])
var.back <- rownames(var.back)[-1]
data.fl.sub <- data.fl[,c("violentcrimes.perpop",var.back)]

fit.final <- lm(violentcrimes.perpop ~., data = data.fl.sub)
summary(fit.final)
```

```
##
## Call:
## lm(formula = violentcrimes.perpop ~ ., data = data.fl.sub)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -935.2 -220.7  -77.3  153.9 1271.6
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)         303.9997    68.0993    4.46  2.4e-05 ***
## num.kids.nvrmarried   0.0307     0.0108    2.85   0.0055 **
## pct.kids.nvrmarried 201.4338    15.3311   13.14  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 402 on 87 degrees of freedom
## Multiple R-squared:  0.769,  Adjusted R-squared:  0.764
## F-statistic:  145 on 2 and 87 DF,  p-value: <2e-16
```

Model Diagnosis

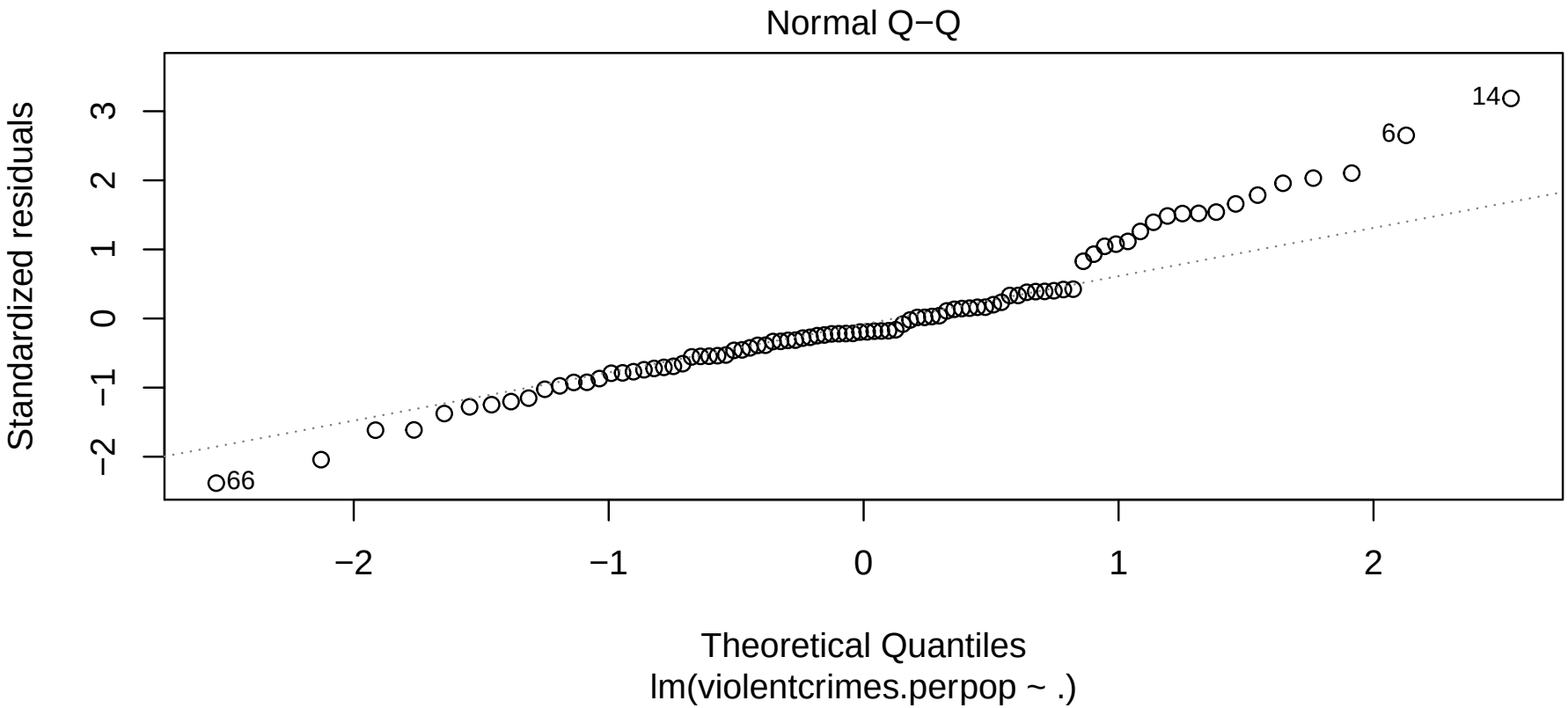
We may take the above model as our final model. Let's do a quick model diagnosis

```
plot(fit.final, 1)
```



Model Diagnosis

```
plot(fit.final, 2)
```



Though not perfect, we will say the linear model assumptions are reasonable.

5.5 Summary

To summarize our findings:

The crime rates increases when the following features increases:

- indian.percap
- pct.pop.underpov
- male.pct.divorce
- num.kids.nvrmarried
- pct.kids.nvrmarried

To control the crime rates, we have to make long term efforts to reduce populations under poverty, to educate people by promoting family values. . .

LASSO prediction

- Although we use relaxed LASSO for inference and prediction, we can still use the LASSO estimator for prediction using the `predict()` function.
- Note that for the `predict()` function of `glmnet` or `cv.glmnet` object, the new `x` has to be a matrix.
- Similarly, the `s` argument is to select the model with specific penalty parameter.

Case Findings

Out of a large number of factors, we see that

- Family structure is very important
- As expected income is another important variable

LASSO Summary

Given a set of variables and responsex, we can use LASSO to choose a set of variables.

- We solve a penalized least squared solutions
- Among many possible sparse models, we use K-fold Cross Validation Errors to choose a candidate model
- Package `glmnet` and `cv.glmnet` give us the LASSO solutions
- The selected variables can be fitted again using multiple regression method to get inference

Pipeline Functions

```
X.fl <- model.matrix(violentcrimes.perpop~., data=data.fl)[, -1] # take the first column's of 1 out
#Step 2: Find x's output from LASSO with min cross-validation error
set.seed(10)  # to control the ramdomness in K folds
fit.fl.cv <- cv.glmnet(X.fl, Y, alpha=1, nfolds=10, intercept = T)
coef.min <- coef(fit.fl.cv, s="lambda.min")  #s=c("lambda.1se","lambda.min") or lambda value
coef.min <- coef.min[which(coef.min !=0),]   # get the non=zero coefficients
var.min <- rownames(as.matrix(coef.min))[-1] # output the names  dim(as.matrix(coef.min))
```

Heatmap using 'ggplot()'

ggplot way:

An original method creating a heat map. Here we show the mean income. First create a new data frame with mean income and corresponding state name

```
income <- data.s[, c("state", "mean.income")]
```

We now need to use standard state names instead of abbreviations, so we have to change the names in the raw data. For example: PA –> Pennsylvania, CA –> California

```
income$region <- tolower(state.name[match(income$state, state.abb)])
```

Next, add the center coordinate for each state. `state.center` contains the coordinate corresponding to `state.abb` in order.

```
income$center_lat  <- state.center$x[match(income$state, state.abb)]
income$center_long <- state.center$y[match(income$state, state.abb)]
```

Next, load US map info - for each state it includes a vector of coordinates describing the shape of the state.

```
states <- map_data("state")
```

Heatmap using ‘ggplot()‘

Notice the column “order” describes the order these points should be linked. Therefore this column should always be ordered properly

Combine the US map data with the income data

```
map <- merge(states, income, sort=FALSE, by="region", all.x=TRUE)
```

Re-establish the point order - very important, try without it!

```
map <- map[order(map$order),]
```

Heatmap using ‘ggplot()‘

Now, plot it using ggplot function. The first two lines create the map with color filled in

- geom_path - create boundary lines
- geom_text - overlay text at the provided coordinates
- scale_fill_continuous - used to tweak the heatmap color

Heatmap using 'ggplot()'

```
ggplot(map, aes(x=long, y=lat, group=group))+
  geom_polygon(aes(fill=mean.income))+
  geom_path()+
  geom_label(data=income,
             aes(x=center_lat, y=center_long, group=NA, label=state),
             size=3, label.size = 0) +
  scale_fill_distiller(palette = "YlGnBu", direction = 1)
```

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