Final Project: Crime and Communities

Kaicheng Luo
University of California, Berkeley
Berkeley, CA, 94720
kevinlkc@berkeley.edu

Priscilla Hu
University of California, Berkeley
Berkeley, CA, 94720
priscilla_hu@berkeley.edu

December 14, 2019

1. Introduction

The crime and communities dataset contains crime data from communities in the United States. 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.
The dataset contains 125 columns total; p = 124 predictive and 1 target (ViolentCrimesPerPop). There are n = 1994 observations. These can be arranged into an n $\times$ p = 1994 $\times$ 127 feature matrix X, and an n $\times$ 1 = 1994 $\times$ 1 response vector y (containing the observations of ViolentCrimesPerPop).
In the first section, we'll conduct data exploration. NA values are removed, explored, and identified. We'll also offer some visualization about the summary statistics of the data. Due to the large scale of the data, we will investigate whether it is possible that the data can be represented in lower dimensions. Those will serve as a fundamental basis on which the numerical prediction proceeds.

2. Data Exploration

Data type

Among 124 of the predictive features, 123 of them are numerical. The only nominal variable is "LemasGangUnitDeploy", which denotes the gang unit deployed. (0 means NO, 10 means YES, 5 means Part Time). For regression tasks, it shall be treated as a categorical variable.

Missing Values

The NA values comes from the limitation that the LEMAS survey was of the police departments with at least 100 officers, plus a random sample of smaller departments. Many communities are missing LEMAS data. There are 23 features coming from the LEMAS survey, each of which contains 1675 identical NA observations. There're 3 heuristic ways to deal with missing values. We can either drop the observations with missing values, or drop the features, or implement different kinds of interpolation techniques. For our purpose, we omitted all the features containing missing values. This is because the total observations that contains missing values is too large in scale (1675/1945), and our final goal is to do numerical prediction, not causal inference. The more we predict (in some sense we intrapolate) the intermediaries, the less stability we obtain in our final prediction.

Normalization

The original dataset is not normalized. The necessity of conducting normalization depends on the model that we're going to utilize. For example, there're no clear reason why normalization shall be helpful in linear regression. Yet in PCA, normalization offers us the handy benefit of balancing the weights of the features. Without standardizing our data, the projection vector will very likely be influenced by the feature with the largest nominal scale, which leads the largest nominal variation. For our task, we normalize our data matrix in Principal Component Regression, Partial Least Square Regression, but keep the original data in the Ordinary Least Square Regression, Ridge Regression and Lasso Regression.

Summary Statistics

Table 1: Summary Statistic of the First 40 Features

Statistic	N	Mean	St. Dev.	Min	Pctl(25)	Pctl(75)	Max
population	1,994	52,251.430	202,147.500	10,005	14,359.2	43,153.8	7,322,564
householdsize	1,994	2.707	0.343	1.600	2.490	2.850	5.280
racepctblack	1,994	9.510	14.102	0.000	0.940	11.965	96.670
racePctWhite	1,994	83.489	16.394	2.680	75.882	95.987	99.630
racePctAsian	1,994	2.751	4.648	0.030	0.612	2.738	57.460
racePctHisp	1,994	8.482	15.209	0.120	0.920	8.610	95.290
agePct12t21	1,994	14.431	4.479	4.580	12.230	15.388	54.400
agePct12t29	1,994	27.617	6.148	9.380	24.380	29.180	70.510
agePct16t24	1,994	13.985	5.897	4.640	11.340	14.357	63.620
agePct65up	1,994	12.005	4.804	1.660	8.922	14.548	52.770
numbUrban	1,994	46,671.760	203,150.800	0	0	41,931.5	7,322,564
pctUrban	1,994	69.622	44.479	0	0	100	100
medIncome	1,994	33,699.330	13,391.740	11,576	23,597	41,214.8	123,625
pctWWage	1,994	78.080	7.844	31.680	73.225	83.705	96.620
pctWFarmSelf	1,994	0.893	0.701	0.000	0.470	1.110	6.530
pctWInvInc	1,994	43.365	12.753	7.910	34.190	52.068	89.040
pctWSocSec	1,994	26.660	8.239	4.810	20.982	31.840	76.390
pctWPubAsst	1,994	6.806	4.490	0.500	3.362	9.150	26.920
pctWRetire	1,994	16.064	4.560	3.460	12.990	18.777	45.510
medFamInc	1,994	39,552.840	14,202.650	13,785	29,307.2	46,682.8	131,315
perCapInc	1,994	15,521.720	6,227.115	5,237	11,548.2	17,774.5	63,302
whitePerCap	1,994	16,535.020	6,311.111	5,472	12,596	18,609.8	68,850
blackPerCap	1,994	11,472.240	9,225.460	0	6,705.5	14,463.8	212,120
indianPerCap	1,994	12,257.250	15,359.540	0	6,336	14,690	480,000
AsianPerCap	1,994	14,284.490	9,770.964	0	8,440.8	17,346	106,165
HispPerCap	1,994	10,989.440	5,818.878	0	7,252.8	13,360	54,648
NumUnderPov	1,994	7,398.380	37,930.920	78	936.2	5,097.5	1,384,994
PctPopUnderPov	1,994	11.796	8.510	0.640	4.692	17.078	48.820
PctLess9thGrade	1,994	9.444	6.844	0.200	4.770	12.245	49.890
PctNotHSGrad	1,994	22.701	11.062	2.090	14.195	29.665	73.660
PctBSorMore	1,994	22.992	12.514	1.630	14.090	28.935	73.630
PctUnemployed	1,994	6.024	2.725	1.320	4.090	7.430	23.830
PctEmploy	1,994	61.777	8.108	24.820	56.353	67.505	84.670
PctEmplManu	1,994	17.789	8.109	2.050	11.940	22.755	50.030
PctEmplProfServ	1,994	24.575	6.654	8.690	20.110	27.628	62.670
PctOccupManu	1,994	13.747	6.410	1.370	9.072	17.465	44.270
PctOccupMgmtProf	1,994	28.254	9.262	6.480	21.920	32.892	64.970
MalePctDivorce	1,994	9.180	2.790	2.130	7.162	11.110	19.090
MalePctNevMarr	1,994	30.668	8.097	12.060	25.410	33.470	76.320
FemalePctDiv	1,994	12.401	3.254	3.350	9.940	14.800	23.460

Multicoliearity

The rank of the matrix X is 98 yet the dimension is 1994×100. Namely two of the columns are linearly dependent on the other dimensions. The feature matrix can only span a linear subspace of 98 dimensions, so exact multicoliearity exists in our data. Exactly 2 NA values will be returned in OLS regression.
Another thing for us to be concerned with is that, as is shown in the correlation graph below, there're also a lot of variables will strong correlation that imply that inexact colinearity exists. However, that's not so awful as we're not touching upon the inference side of the analysis.

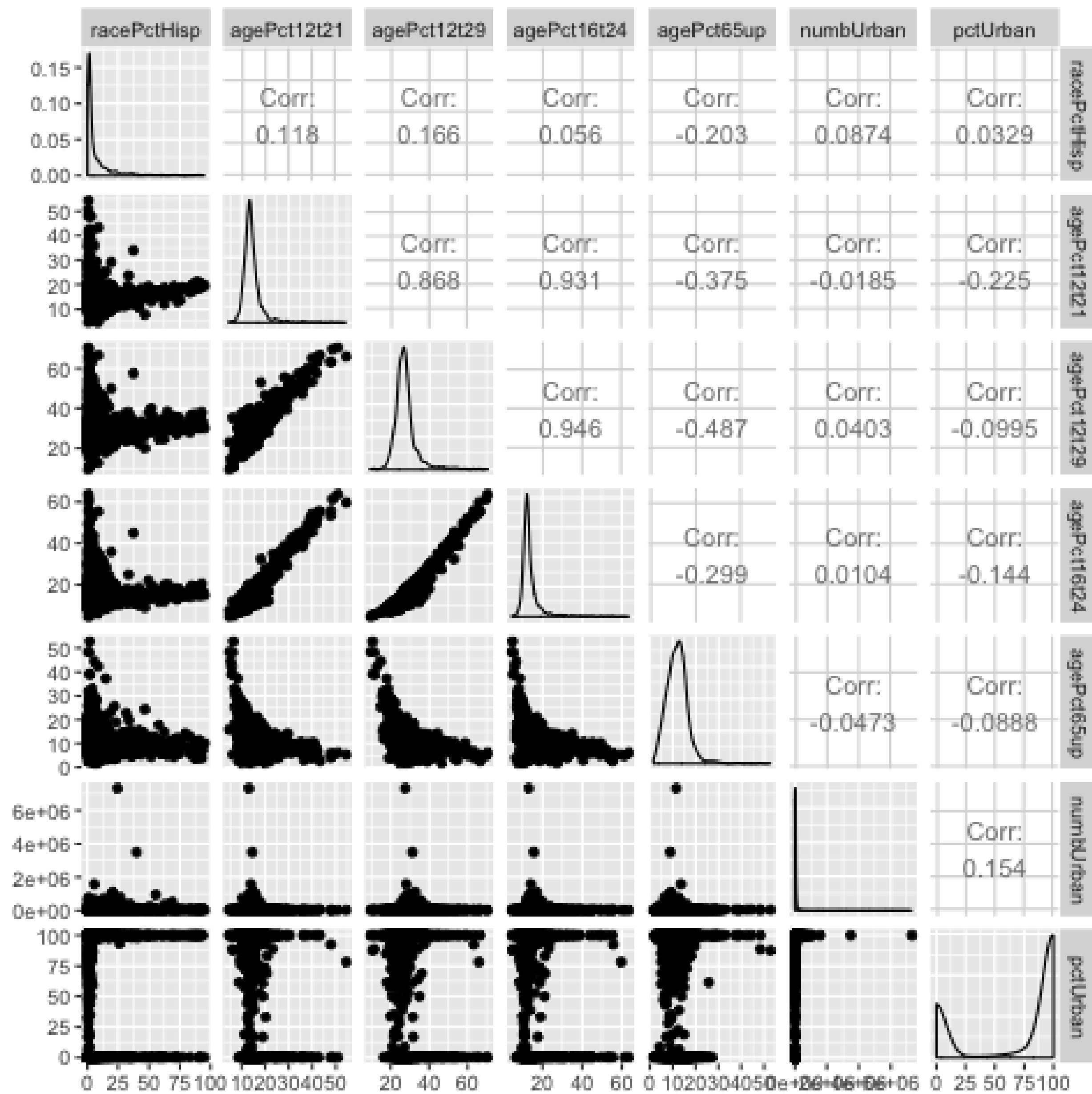


Figure 1: Multicoliearity effects

Heteroskedasticity

According to Gauss-Markov Theorem, the fundamental assumptions for ordinary least square regression involves the homeskedasticity among features. However, the real-life data does not actually fit in the settings. As is illustrated by the plot below, the residuals of a simple linear regression is generally not independent of household sizes – one of the typical features of communities. This suggests that we might improve our prediction accuracy by weighting our coefficients by their variation.

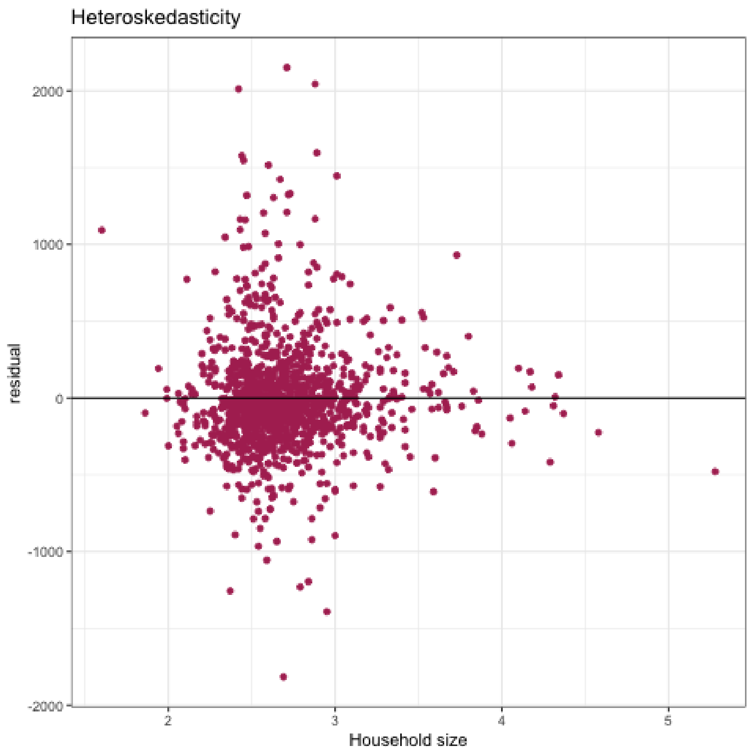


Figure 2: Heteroskedasticity effects

3. Prediction

```
#split data into training set and test set
train<-1:1200
val<-1201:1600
test<-1601:1994
```

3.1 Baseline: OLS Regression

$$\hat{\beta} = \arg\min_{\beta}(Y - X\beta)^2 \tag{1}$$

```
#OLS
lm.fit=lm(y~X,subset=train)
coef=lm.fit$coefficients
coef[is.na(coef)]=0
#XOwnOccQrange's and XRentQrange's coefs are NA.  Multicollinearity as shown in the last section
```

3.2 Principal Component Regression

$$\hat{\beta} = \arg\min_{\beta}(Y - XV\beta)^2 \tag{2}$$

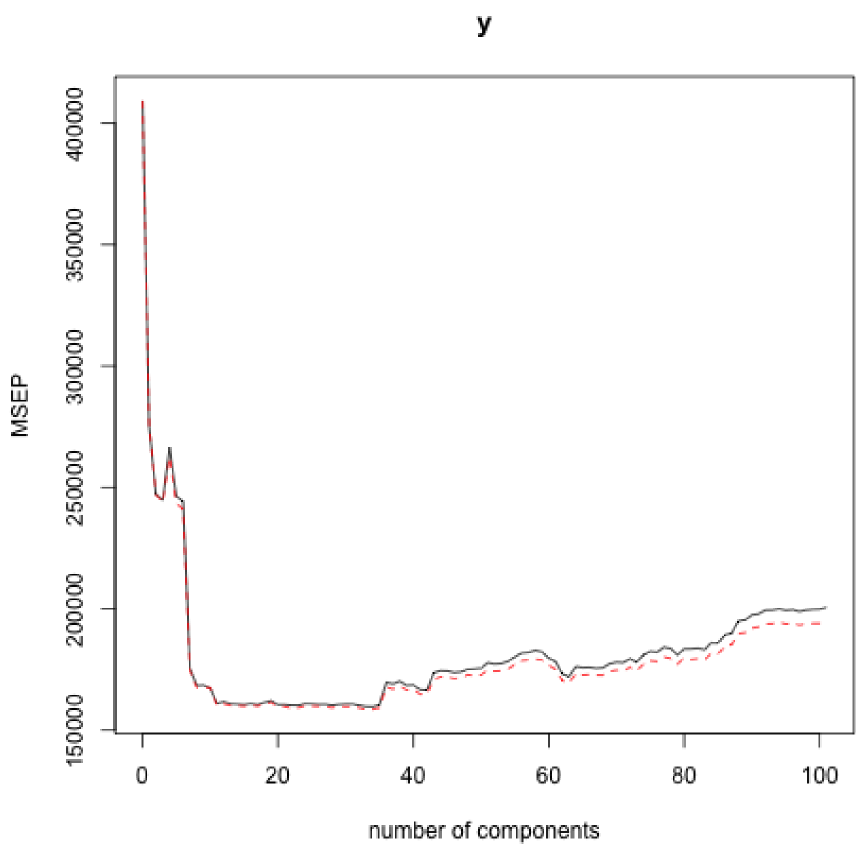
where

$$V_1 = \arg\max_{u}(u^T X^T X u) \quad s.t. \, u^T u = 1 \tag{3}$$

```
#PCR
library(pls)
```

```
pcr.fit=pcr(y~X,scale=TRUE,validation="CV",subset=train)
summary(pcr.fit)
```

```r
validationplot(pcr.fit,val.type="MSEP")
```



The best model, chosen by cross-validation, is the one with 22 principal components retained in the model specification.

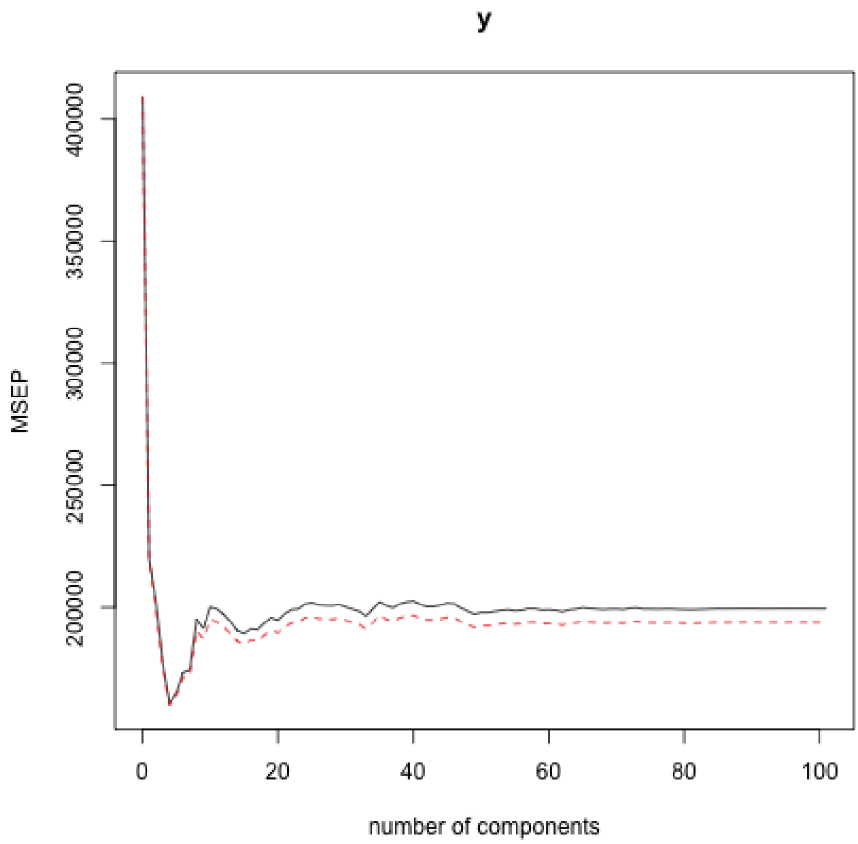
```r
pcr.fit=pcr(y~X,scale=TRUE,ncomp=22,subset=train)
summary(pcr.fit)
```

```
## Data:    X dimension: 1200 101
##  Y dimension: 1200 1
## Fit method: svdpc
## Number of components considered: 22
## TRAINING: \% variance explained
##    1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X    24.28    40.52    49.61    57.58    64.00    68.26    71.82    74.71
## y    32.87    40.89    44.25    46.53    48.85    49.29    60.04    61.54
##    9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
## X    76.86     78.46     80.02     81.47     82.88     83.98     84.96
## y    61.58     61.71     62.74     62.91     63.15     63.16     63.28
##    16 comps  17 comps  18 comps  19 comps  20 comps  21 comps  22 comps
## X     85.88     86.79     87.59     88.37     89.11     89.80     90.40
## y     63.28     63.28     63.41     63.41     63.73     63.76     64.02
```

3.3 Partial Least Square Regression

```r
#PLSR
pls.fit=plsr(y~X,subset=train,scale=TRUE,validation="CV")
summary(pls.fit)
```

```r
validationplot(pls.fit,val.type="MSEP")
```



The best model, chosen by cross-validation, is the one with 22 principal components retained in the model specification.

```r
pls.fit=plsr(y~X,subset=train,scale=TRUE,ncomp=4)
summary(pls.fit)
```

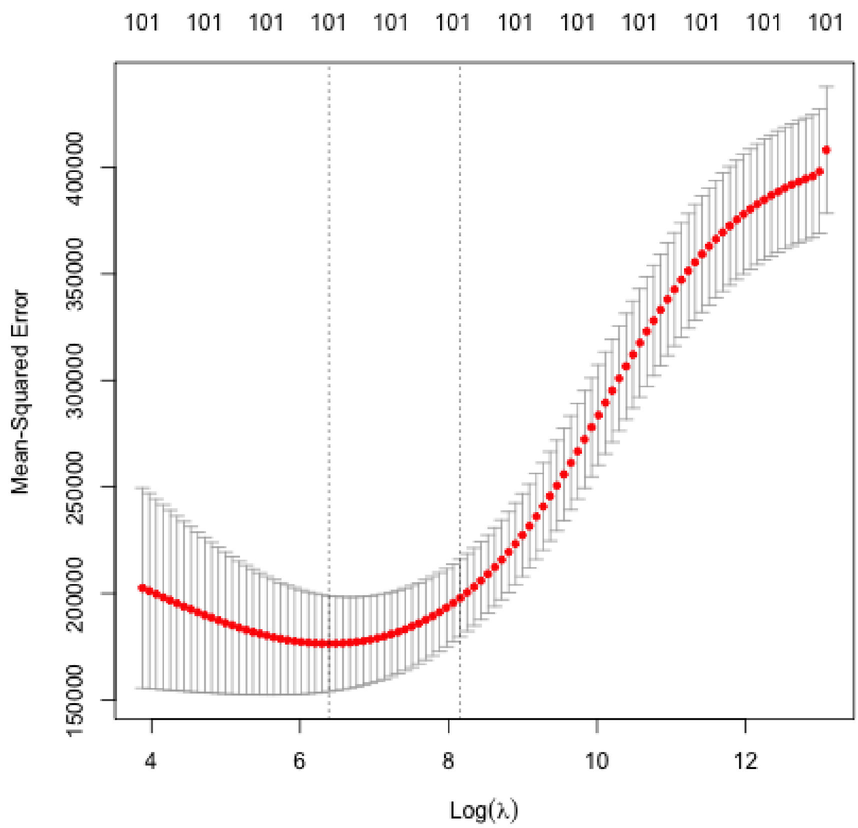
```
## Data:    X dimension: 1200 101
##  Y dimension: 1200 1
## Fit method: kernelpls
## Number of components considered: 4
## TRAINING: \% variance explained
##    1 comps  2 comps  3 comps  4 comps
## X    22.99    34.49    45.13    52.24
## y    48.79    58.80    62.68    64.03
```

3.4 Ridge Regression

```r
#RR
library(glmnet)
```

```r
grid=10^seq(10,-2,length=100)
ridge.mod=glmnet(X[train,],y[train],alpha=0,lambda=grid,thresh=1e-12)
```

```r
#use cross validation to determine the lambda for RR
set.seed(1)
cv.out=cv.glmnet(X[train,],y[train],alpha=0)
plot(cv.out)
```



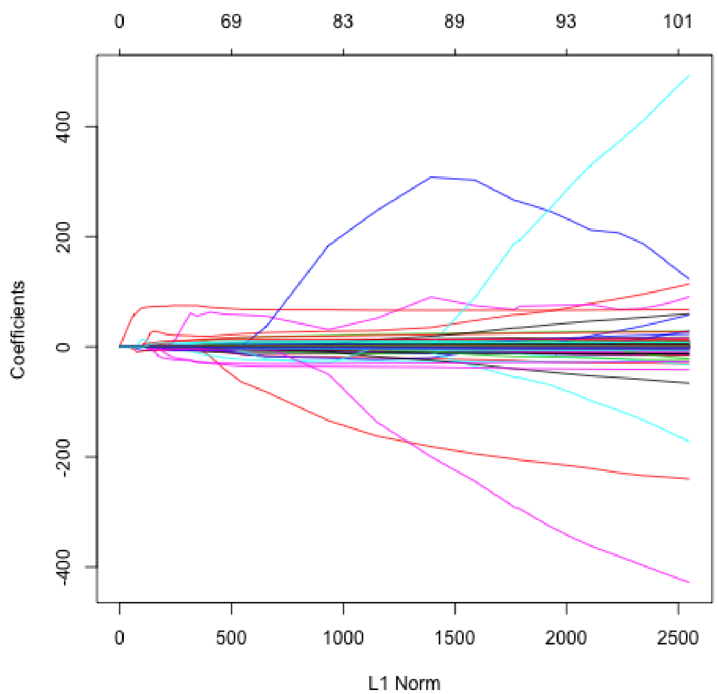

```r
bestlamRR=cv.out$lambda.min
bestlamRR
```

```
## [1] 310.7593
```

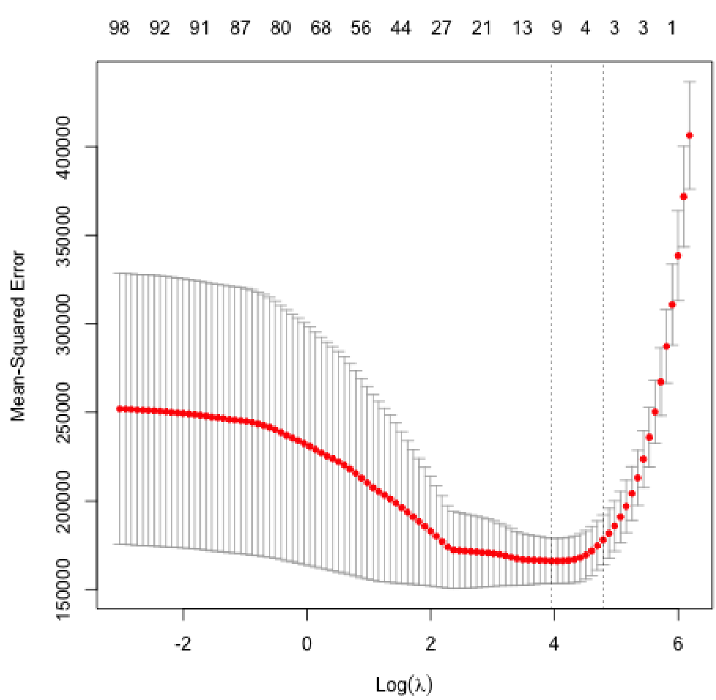
```r
#the coefs of RR
out=glmnet(X[train,],y[train],alpha=0)
predict(out,type="coefficients",s=bestlamRR)[1:102,]
```

3.5 Lasso Regression

```r
#Lasso
lasso.mod=glmnet(X[train,],y[train],alpha=1,lambda=grid)
plot(lasso.mod)
```




```
set.seed(1)
cv.out=cv.glmnet(X[train,],y[train],alpha=1)
plot(cv.out)
```



```
bestlamLso=cv.out$lambda.min
```

```
bestlamLso
```

```
## [1] 15.46656
```

```
out=glmnet(X,y,alpha=1,lambda=grid)
lasso.coef=predict(out,type="coefficients",s=bestlamLso)[1:102,]
lasso.coef
```

```
lasso.coef[lasso.coef!=0]
```

```
##           (Intercept)          racepctblack          racePctWhite
##          1.985369e+03          2.254120e+00         -4.826963e+00
##           agePct12t29              pctUrban            pctWInvInc
##         -2.190052e+00          6.772087e-01         -1.734825e-01
##           AsianPerCap           PctEmplManu        MalePctDivorce
##          2.424487e-04         -1.043838e+00          2.515118e+01
##           PctKids2Par            PctWorkMom   PctKidsBornNeverMar
##         -1.042999e+01         -3.357879e+00          5.320122e+01
##      PctPersDenseHous            HousVacant          PctHousOccup
##          7.102420e+00          5.702318e-03         -4.226322e+00
##      PctVacantBoarded            RentQrange      MedRentPctHousInc
##          7.602265e+00          1.293026e-01          4.908682e-01
## MedOwnCostPctIncNoMtg        PctForeignBorn   LemasPctOfficDrugUn
##         -9.654972e+00          4.285888e-01          7.448798e+00
```

4 Prediction Results, Model Selection and Final Model

```
MSE=rep(0,5)
names(MSE)=c("OLS","PCR","PLSR","Ridge","Lasso")

MSE[1]=mean((y[val]-cbind(1,X[val,])%*%(coef))^2)

pcr.pred=predict(pcr.fit,X[val,],ncomp=20)
MSE[2]=mean((pcr.pred-y[val])^2)

pls.pred=predict(pls.fit,X[val,],ncomp=4)
MSE[3]=mean((pls.pred-y[val])^2)

ridge.pred=predict(ridge.mod,s=bestlamRR,newx=X[val,])
MSE[4]=mean((ridge.pred-y[val])^2)

lasso.pred=predict(lasso.mod,s=bestlamLso,newx=X[val,])
MSE[5]=mean((lasso.pred-y[val])^2)

MSE
```

Table 2: Final Accuracy

	OLS	PCR	PLSR	Ridge	Lasso
MSE	134577.6	122570.0	120356.2	118460.3	121873.2.

```
MSE[which.min(MSE)]
```

```
##    Ridge
## 118460.3
```

%

Final Model: Ridge

```
ridge.mod=glmnet(X[-test,],y[-test],alpha=0,lambda=bestlamRR)

out=glmnet(X[-test,],y[-test],alpha=0)
predict(out,type="coefficients",s=bestlamRR)[1:102,]
```

```
ridge.pred=predict(ridge.mod,s=bestlamRR,newx=X[test,])
MSE_ridge=mean((ridge.pred-y[test])^2)
MSE_ridge
```

```
## [1] 131547
```

References

[1] Hlavac, Marek (2018). stargazer: Well-Formatted Regression and Summary Statistics Tables. R package version 5.2.2. https://CRAN.R-project.org/package=stargazer