ISYE 6501 HW Wk5 Solution

```
# Clear environment
rm(list = ls())

# Set seed
set.seed(19)

# library
# Cross Validation
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
## Registered S3 methods overwritten by 'ggplot2':
##   method         from
##   [.quosures     rlang
##   c.quosures     rlang
##   print.quosures rlang
```

```
library(MASS)
library(glmnet)
```

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

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

```
## Loaded glmnet 2.0-18
```

Question 11.1

Using the crime data set uscrime.txt from Questions 8.2, 9.1, and 10.1, build a regression model using: 1. Stepwise regression 2. Lasso 3. Elastic net

For Parts 2 and 3, remember to scale the data first – otherwise, the regression coefficients will be on different scales and the constraint won't have the desired effect. For Parts 2 and 3, use the glmnet function in R.**

```
# First to read the uscrime data to R
df_crime <- read.table("uscrime.txt", stringsAsFactors = FALSE, header = TRUE)

# Since we've use this data set for several weeks so I just pull the columns summary via str()
str(df_crime)
```

```
## 'data.frame':    47 obs. of  16 variables:
##  $ M     : num  15.1 14.3 14.2 13.6 14.1 12.1 12.7 13.1 15.7 14 ...
##  $ So    : int  1 0 1 0 0 0 1 1 1 0 ...
##  $ Ed    : num  9.1 11.3 8.9 12.1 12.1 11 11.1 10.9 9 11.8 ...
##  $ Po1   : num  5.8 10.3 4.5 14.9 10.9 11.8 8.2 11.5 6.5 7.1 ...
```

```
##  $ Po2   : num  5.6 9.5 4.4 14.1 10.1 11.5 7.9 10.9 6.2 6.8 ...
##  $ LF    : num  0.51 0.583 0.533 0.577 0.591 0.547 0.519 0.542 0.553 0.632 ...
##  $ M.F   : num  95 101.2 96.9 99.4 98.5 ...
##  $ Pop   : int  33 13 18 157 18 25 4 50 39 7 ...
##  $ NW    : num  30.1 10.2 21.9 8 3 4.4 13.9 17.9 28.6 1.5 ...
##  $ U1    : num  0.108 0.096 0.094 0.102 0.091 0.084 0.097 0.079 0.081 0.1 ...
##  $ U2    : num  4.1 3.6 3.3 3.9 2 2.9 3.8 3.5 2.8 2.4 ...
##  $ Wealth: int  3940 5570 3180 6730 5780 6890 6200 4720 4210 5260 ...
##  $ Ineq  : num  26.1 19.4 25 16.7 17.4 12.6 16.8 20.6 23.9 17.4 ...
##  $ Prob  : num  0.0846 0.0296 0.0834 0.0158 0.0414 ...
##  $ Time  : num  26.2 25.3 24.3 29.9 21.3 ...
##  $ Crime : int  791 1635 578 1969 1234 682 963 1555 856 705 ...
```

Build Scaled raw dataframe for modeling

```
# As requested, need to scale the data first before buidling the model
# Though it is not requested for Stepwise, but as suggested in the office hour, it would be better to s

# Crime as reponse does not need to scale
# factor 'So' as factor 0, 1 does not need to scale
df_to_scale <- df_crime[,c(1,3,4,5,6,7,8,9,10,11,12,13,14,15)]

# Need to convert to matrix first to use scale, then convert back to DataFrame
df_scale <- scale(as.matrix(df_to_scale))
df_scale <- as.data.frame(df_scale)

# add back response and column 'So'
df_crime_s <- cbind.data.frame(df_crime['So'],df_scale,df_crime['Crime'])
str(df_crime_s)
```

```
## 'data.frame':    47 obs. of  16 variables:
##  $ So    : int  1 0 1 0 0 0 1 1 1 0 ...
##  $ M     : num  0.989 0.352 0.273 -0.205 0.193 ...
##  $ Ed    : num  -1.309 0.658 -1.487 1.373 1.373 ...
##  $ Po1   : num  -0.909 0.606 -1.346 2.154 0.808 ...
##  $ Po2   : num  -0.867 0.528 -1.296 2.173 0.743 ...
##  $ LF    : num  -1.267 0.54 -0.698 0.391 0.738 ...
##  $ M.F   : num  -1.1206 0.9834 -0.4758 0.3726 0.0671 ...
##  $ Pop   : num  -0.095 -0.62 -0.489 3.162 -0.489 ...
##  $ NW    : num  1.94374 0.00848 1.1463 -0.20546 -0.69171 ...
##  $ U1    : num  0.6951 0.0295 -0.0814 0.3623 -0.2478 ...
##  $ U2    : num  0.831 0.239 -0.116 0.595 -1.655 ...
##  $ Wealth: num  -1.362 0.328 -2.149 1.53 0.545 ...
##  $ Ineq  : num  1.679 0 1.404 -0.677 -0.501 ...
##  $ Prob  : num  1.65 -0.769 1.597 -1.376 -0.25 ...
##  $ Time  : num  -0.056 -0.183 -0.324 0.466 -0.748 ...
##  $ Crime : int  791 1635 578 1969 1234 682 963 1555 856 705 ...
```

Stepwise

```
# Perform Setpwise Regression
model_step <- lm(Crime~., data=df_crime_s)
```

```r
step(model_step,
     scope = list(lower=formula(lm(Crime~1, data=df_crime_s)),
                  upper=formula(lm(Crime~., data=df_crime_s))),
     direction="both")
```

```
## Start:  AIC=514.65
## Crime ~ So + M + Ed + Po1 + Po2 + LF + M.F + Pop + NW + U1 +
##     U2 + Wealth + Ineq + Prob + Time
##
##          Df Sum of Sq     RSS    AIC
## - So      1        29 1354974 512.65
## - LF      1      8917 1363862 512.96
## - Time    1     10304 1365250 513.00
## - Pop     1     14122 1369068 513.14
## - NW      1     18395 1373341 513.28
## - M.F     1     31967 1386913 513.74
## - Wealth  1     37613 1392558 513.94
## - Po2     1     37919 1392865 513.95
## <none>                1354946 514.65
## - U1      1     83722 1438668 515.47
## - Po1     1    144306 1499252 517.41
## - U2      1    181536 1536482 518.56
## - M       1    193770 1548716 518.93
## - Prob    1    199538 1554484 519.11
## - Ed      1    402117 1757063 524.86
## - Ineq    1    423031 1777977 525.42
##
## Step:  AIC=512.65
## Crime ~ M + Ed + Po1 + Po2 + LF + M.F + Pop + NW + U1 + U2 +
##     Wealth + Ineq + Prob + Time
##
##          Df Sum of Sq     RSS    AIC
## - Time    1     10341 1365315 511.01
## - LF      1     10878 1365852 511.03
## - Pop     1     14127 1369101 511.14
## - NW      1     21626 1376600 511.39
## - M.F     1     32449 1387423 511.76
## - Po2     1     37954 1392929 511.95
## - Wealth  1     39223 1394197 511.99
## <none>                1354974 512.65
## - U1      1     96420 1451395 513.88
## + So      1        29 1354946 514.65
## - Po1     1    144302 1499277 515.41
## - U2      1    189859 1544834 516.81
## - M       1    195084 1550059 516.97
## - Prob    1    204463 1559437 517.26
## - Ed      1    403140 1758114 522.89
## - Ineq    1    488834 1843808 525.13
##
## Step:  AIC=511.01
## Crime ~ M + Ed + Po1 + Po2 + LF + M.F + Pop + NW + U1 + U2 +
##     Wealth + Ineq + Prob
##
```

```
##          Df Sum of Sq     RSS    AIC
## - LF      1     10533 1375848 509.37
## - NW      1     15482 1380797 509.54
## - Pop     1     21846 1387161 509.75
## - Po2     1     28932 1394247 509.99
## - Wealth  1     36070 1401385 510.23
## - M.F     1     41784 1407099 510.42
## <none>                1365315 511.01
## - U1      1     91420 1456735 512.05
## + Time    1     10341 1354974 512.65
## + So      1        65 1365250 513.00
## - Po1     1    134137 1499452 513.41
## - U2      1    184143 1549458 514.95
## - M       1    186110 1551425 515.01
## - Prob    1    237493 1602808 516.54
## - Ed      1    409448 1774763 521.33
## - Ineq    1    502909 1868224 523.75
##
## Step:  AIC=509.37
## Crime ~ M + Ed + Po1 + Po2 + M.F + Pop + NW + U1 + U2 + Wealth +
##     Ineq + Prob
##
##          Df Sum of Sq     RSS    AIC
## - NW      1     11675 1387523 507.77
## - Po2     1     21418 1397266 508.09
## - Pop     1     27803 1403651 508.31
## - M.F     1     31252 1407100 508.42
## - Wealth  1     35035 1410883 508.55
## <none>                1375848 509.37
## - U1      1     80954 1456802 510.06
## + LF      1     10533 1365315 511.01
## + Time    1      9996 1365852 511.03
## + So      1      3046 1372802 511.26
## - Po1     1    123896 1499744 511.42
## - U2      1    190746 1566594 513.47
## - M       1    217716 1593564 514.27
## - Prob    1    226971 1602819 514.54
## - Ed      1    413254 1789103 519.71
## - Ineq    1    500944 1876792 521.96
##
## Step:  AIC=507.77
## Crime ~ M + Ed + Po1 + Po2 + M.F + Pop + U1 + U2 + Wealth + Ineq +
##     Prob
##
##          Df Sum of Sq     RSS    AIC
## - Po2     1     16706 1404229 506.33
## - Pop     1     25793 1413315 506.63
## - M.F     1     26785 1414308 506.66
## - Wealth  1     31551 1419073 506.82
## <none>                1387523 507.77
## - U1      1     83881 1471404 508.52
## + NW      1     11675 1375848 509.37
## + So      1      7207 1380316 509.52
## + LF      1      6726 1380797 509.54
```

```
## + Time    1      4534 1382989 509.61
## - Po1     1    118348 1505871 509.61
## - U2      1    201453 1588976 512.14
## - Prob    1    216760 1604282 512.59
## - M       1    309214 1696737 515.22
## - Ed      1    402754 1790276 517.74
## - Ineq    1    589736 1977259 522.41
##
## Step:  AIC=506.33
## Crime ~ M + Ed + Po1 + M.F + Pop + U1 + U2 + Wealth + Ineq +
##     Prob
##
##          Df Sum of Sq     RSS    AIC
## - Pop     1     22345 1426575 505.07
## - Wealth  1     32142 1436371 505.39
## - M.F     1     36808 1441037 505.54
## <none>                1404229 506.33
## - U1      1     86373 1490602 507.13
## + Po2     1     16706 1387523 507.77
## + NW      1      6963 1397266 508.09
## + So      1      3807 1400422 508.20
## + LF      1      1986 1402243 508.26
## + Time    1       575 1403654 508.31
## - U2      1    205814 1610043 510.76
## - Prob    1    218607 1622836 511.13
## - M       1    307001 1711230 513.62
## - Ed      1    389502 1793731 515.83
## - Ineq    1    608627 2012856 521.25
## - Po1     1   1050202 2454432 530.57
##
## Step:  AIC=505.07
## Crime ~ M + Ed + Po1 + M.F + U1 + U2 + Wealth + Ineq + Prob
##
##          Df Sum of Sq     RSS    AIC
## - Wealth  1     26493 1453068 503.93
## <none>                1426575 505.07
## - M.F     1     84491 1511065 505.77
## - U1      1     99463 1526037 506.24
## + Pop     1     22345 1404229 506.33
## + Po2     1     13259 1413315 506.63
## + NW      1      5927 1420648 506.87
## + So      1      5724 1420851 506.88
## + LF      1      5176 1421398 506.90
## + Time    1      3913 1422661 506.94
## - Prob    1    198571 1625145 509.20
## - U2      1    208880 1635455 509.49
## - M       1    320926 1747501 512.61
## - Ed      1    386773 1813348 514.35
## - Ineq    1    594779 2021354 519.45
## - Po1     1   1127277 2553852 530.44
##
## Step:  AIC=503.93
## Crime ~ M + Ed + Po1 + M.F + U1 + U2 + Ineq + Prob
##
```

```
##          Df Sum of Sq     RSS    AIC
## <none>                1453068 503.93
## + Wealth  1     26493 1426575 505.07
## - M.F     1    103159 1556227 505.16
## + Pop     1     16697 1436371 505.39
## + Po2     1     14148 1438919 505.47
## + So      1      9329 1443739 505.63
## + LF      1      4374 1448694 505.79
## + NW      1      3799 1449269 505.81
## + Time    1      2293 1450775 505.86
## - U1      1    127044 1580112 505.87
## - Prob    1    247978 1701046 509.34
## - U2      1    255443 1708511 509.55
## - M       1    296790 1749858 510.67
## - Ed      1    445788 1898855 514.51
## - Ineq    1    738244 2191312 521.24
## - Po1     1   1672038 3125105 537.93
```

```
##
## Call:
## lm(formula = Crime ~ M + Ed + Po1 + M.F + U1 + U2 + Ineq + Prob,
##     data = df_crime_s)
##
## Coefficients:
## (Intercept)            M           Ed          Po1          M.F
##      905.09       117.28       201.50       305.07        65.83
##          U1           U2         Ineq         Prob
##     -109.73       158.22       244.70       -86.31
```

Model Validation

Use libarary 'Caret' to perform K-fold cross-validation Need to train the stepwise regression model in each training set

```
# create 10 fold cross-validation
train_control <- trainControl(method='cv', number=10)

# method "lmStepAIC" is the stepwise regression method in library 'Caret'
# use trace = 0 to jump directly to the stepwise result
step.model <- train(Crime~., data=df_crime_s,method="lmStepAIC",
                    scope = list(lower=Crime~1,upper=Crime~.),
                    direction='both',
                    trControl = train_control,
                    trace=0)
summary(step.model)
```

```
##
## Call:
## lm(formula = .outcome ~ M + Ed + Po1 + M.F + U1 + U2 + Ineq +
##     Prob, data = dat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
```

```
## -444.70 -111.07    3.03  122.15  483.30
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   905.09      28.52  31.731  < 2e-16 ***
## M             117.28      42.10   2.786  0.00828 **
## Ed            201.50      59.02   3.414  0.00153 **
## Po1           305.07      46.14   6.613 8.26e-08 ***
## M.F            65.83      40.08   1.642  0.10874
## U1           -109.73      60.20  -1.823  0.07622 .
## U2            158.22      61.22   2.585  0.01371 *
## Ineq          244.70      55.69   4.394 8.63e-05 ***
## Prob          -86.31      33.89  -2.547  0.01505 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 195.5 on 38 degrees of freedom
## Multiple R-squared:  0.7888, Adjusted R-squared:  0.7444
## F-statistic: 17.74 on 8 and 38 DF,  p-value: 1.159e-10
```

From the result, we can see it has the same result as performing the stepwise regression on the entire data set. So the stepwise regression model output is the local best model we have From the model, we can see the adjusted R-squared is 0.744, all p value is small

```
step.model
```

```
## Linear Regression with Stepwise Selection
##
## 47 samples
## 15 predictors
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 43, 43, 42, 42, 42, 44, ...
## Resampling results:
##
##   RMSE      Rsquared   MAE
##   236.2107  0.5816262  201.1269
```

Above as the cross validation result. The Rsquared is still 0.6. Looks ok.

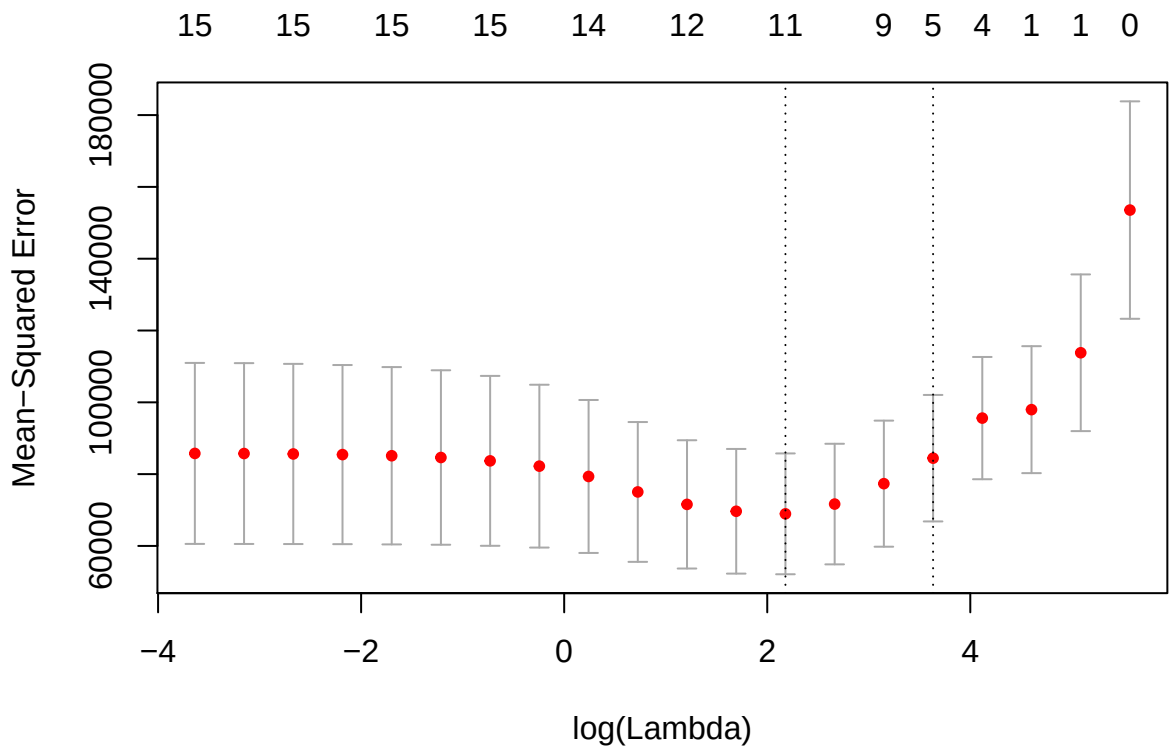
LASSO

```
# use glmnet, 10 fold, with alpha set as 1 for LASSO, lambda start as 20
model_lasso <- cv.glmnet(x=as.matrix(df_crime_s[,-16]),
                         y=as.matrix(df_crime_s[,16]),
                         alpha=1, #LASSO
                         nfolds=10,
                         nlambda=20,
                         type.measure='mse',
                         family='gaussian')
model_lasso
```

```
## $lambda
##  [1] 263.09539664 162.02682877  99.78393228  61.45175597  37.84495384
##  [6]  23.30674704  14.35341842   8.83952702   5.44380688   3.35255872
## [11]   2.06466728   1.27152165   0.78306433   0.48224876   0.29699204
## [16]   0.18290201   0.11263988   0.06936907   0.04272082   0.02630954
##
## $cvm
##  [1] 153547.24 113805.53  97959.72  95606.34  84442.91  77347.36  71663.23
##  [8]  68941.30  69654.13  71572.33  75038.67  79348.73  82223.02  83697.24
## [15]  84629.27  85123.31  85443.72  85622.32  85737.11  85770.16
##
## $cvsd
##  [1] 30278.77 21821.07 17678.71 17026.86 17613.66 17557.58 16790.34
##  [8] 16824.24 17357.85 17863.40 19467.43 21303.44 22675.78 23657.20
## [15] 24295.16 24695.00 24941.21 25089.13 25182.29 25205.55
##
## $cvup
##  [1] 183826.01 135626.60 115638.43 112633.20 102056.57  94904.94  88453.57
##  [8]  85765.54  87011.98  89435.73  94506.10 100652.17 104898.80 107354.45
## [15] 108924.43 109818.30 110384.92 110711.45 110919.41 110975.71
##
## $cvlo
##  [1] 123268.48  91984.47  80281.00  78579.48  66829.25  59789.78  54872.90
##  [8]  52117.06  52296.28  53708.92  55571.24  58045.28  59547.23  60040.04
## [15]  60334.12  60428.31  60502.51  60533.19  60554.82  60564.61
##
## $nzero
##  s0  s1  s2  s3  s4  s5  s6  s7  s8  s9 s10 s11 s12 s13 s14 s15 s16 s17
##   0   1   1   4   5   9  10  11  12  12  13  14  15  15  15  15  14  15
## s18 s19
##  15  15
##
## $name
##                 mse
## "Mean-Squared Error"
##
## $glmnet.fit
##
## Call:  glmnet(x = as.matrix(df_crime_s[, -16]), y = as.matrix(df_crime_s[,      16]), alpha = 1, nla
##
##       Df   %Dev    Lambda
##  [1,]  0 0.0000 263.10000
##  [2,]  1 0.2935 162.00000
##  [3,]  1 0.4048  99.78000
##  [4,]  4 0.4772  61.45000
##  [5,]  5 0.6197  37.84000
##  [6,]  9 0.6990  23.31000
##  [7,] 10 0.7467  14.35000
##  [8,] 11 0.7743   8.84000
##  [9,] 12 0.7882   5.44400
## [10,] 12 0.7937   3.35300
## [11,] 13 0.7959   2.06500
## [12,] 14 0.7968   1.27200
## [13,] 15 0.8003   0.78310
```

```
## [14,] 15 0.8019   0.48220
## [15,] 15 0.8026   0.29700
## [16,] 15 0.8029   0.18290
## [17,] 14 0.8030   0.11260
## [18,] 15 0.8030   0.06937
## [19,] 15 0.8031   0.04272
## [20,] 15 0.8031   0.02631
##
## $lambda.min
## [1] 8.839527
##
## $lambda.1se
## [1] 37.84495
##
## attr(,"class")
## [1] "cv.glmnet"
```

```
# plot model with lambda
plot(model_lasso)
```



Below to show the lambda.min, use lambda.min as the best lambda to use for the model (min of cvm)

```
# build table show the change of model quality based on lambda
cbind(model_lasso$lambda, model_lasso$cvm, model_lasso$nzero)
```

```
##             [,1]      [,2] [,3]
```

```
## s0  263.09539664 153547.24    0
## s1  162.02682877 113805.53    1
## s2   99.78393228  97959.72    1
## s3   61.45175597  95606.34    4
## s4   37.84495384  84442.91    5
## s5   23.30674704  77347.36    9
## s6   14.35341842  71663.23   10
## s7    8.83952702  68941.30   11
## s8    5.44380688  69654.13   12
## s9    3.35255872  71572.33   12
## s10   2.06466728  75038.67   13
## s11   1.27152165  79348.73   14
## s12   0.78306433  82223.02   15
## s13   0.48224876  83697.24   15
## s14   0.29699204  84629.27   15
## s15   0.18290201  85123.31   15
## s16   0.11263988  85443.72   14
## s17   0.06936907  85622.32   15
## s18   0.04272082  85737.11   15
## s19   0.02630954  85770.16   15
```

```
model_lasso$lambda.min
```

```
## [1] 8.839527
```

At lambda min, the model is using 10 factors.

```
coef(model_lasso, s=model_lasso$lambda.min)
```

```
## 16 x 1 sparse Matrix of class "dgCMatrix"
##                       1
## (Intercept) 889.854605
## So           44.739597
## M            90.279192
## Ed          140.289856
## Po1         304.140909
## Po2           .
## LF            .
## M.F          55.640579
## Pop           .
## NW            6.487469
## U1          -38.645259
## U2           74.618077
## Wealth        7.441720
## Ineq        194.791647
## Prob        -83.865228
## Time          .
```

```
lambda_lasso <- model_lasso$lambda.min
# Calculate SSR and R Square
model_lasso_cv <- glmnet(x=as.matrix(df_crime_s[,-16]),
                         y=as.matrix(df_crime_s[,16]),
```

```
                        alpha=1, #LASSO
                        lambda = lambda_lasso,
                        family='gaussian')
X <- as.matrix(df_crime_s[,-16])
y <- as.matrix(df_crime_s[,16])
y_hat <- predict(model_lasso_cv,X)

# Calcualte SSR
ssr_lasso <- t(y - y_hat) %*% (y - y_hat)
ssr_lasso
```

```
##         s0
## s0 1551876
```

```
# Calcuate R Square
rsq_lasso <- cor(y, y_hat)^2
rsq_lasso
```

```
##             s0
## [1,] 0.7801183
```

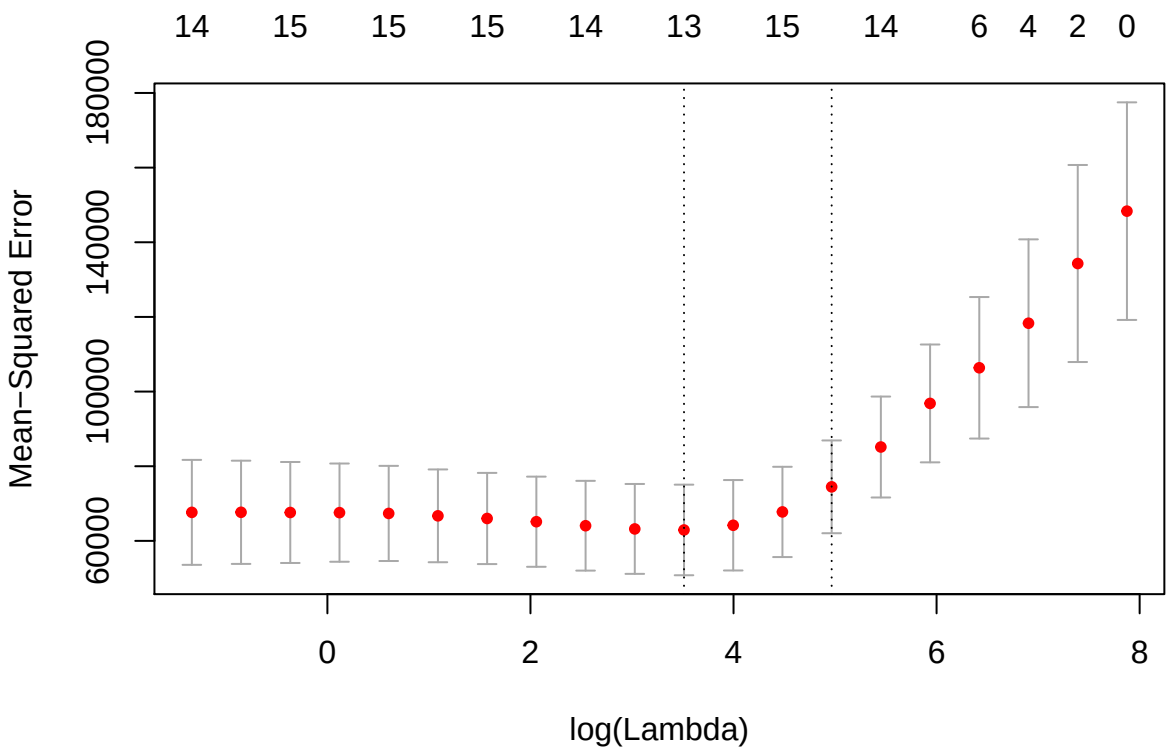
Not as good as in the stepwise R square.

Elastic Net

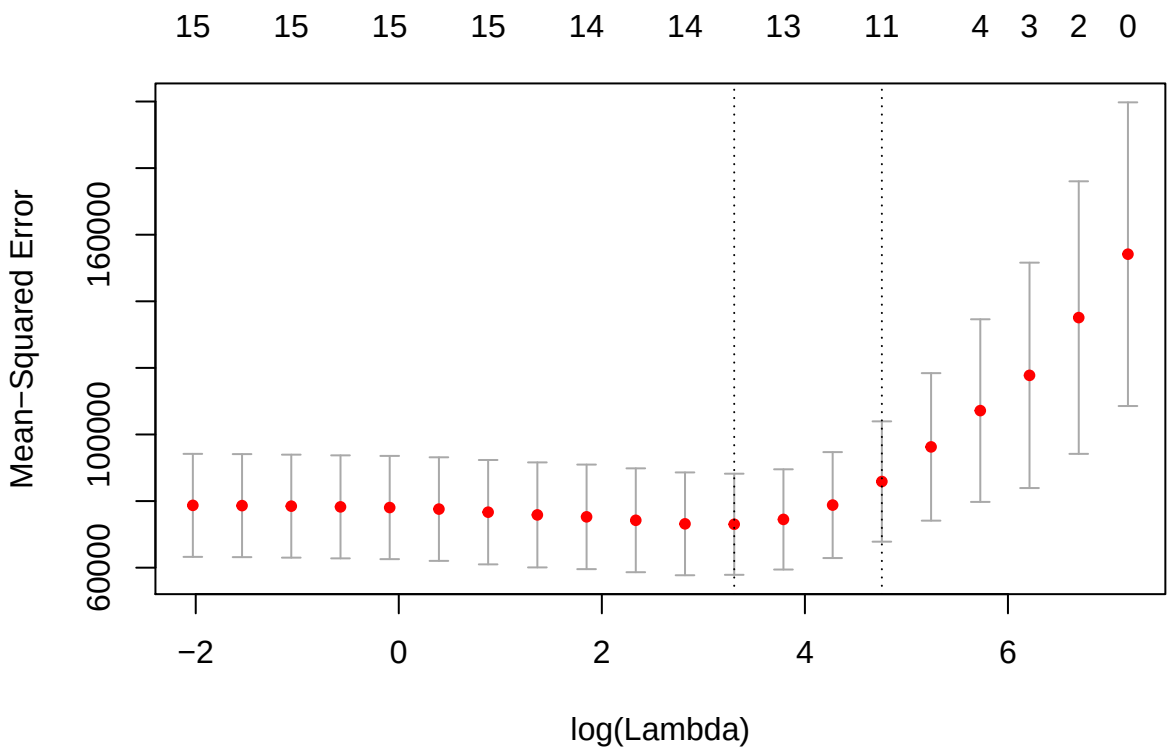
```
# use glmnet for Elastic Net
# build loop to try alpha from 0.1 to 1
list_rsq <- c()

for (i in seq(0.1, 1, 0.1)){
  model_elastic <- cv.glmnet(x=as.matrix(df_crime_s[,-16]),
                        y=as.matrix(df_crime_s[,16]),
                        alpha=i,
                        nfolds=10,
                        nlambda=20,
                        type.measure='mse',
                        family='gaussian')
  lambda_elastic = model_elastic$lambda.min
  y_hat <- predict(model_elastic,X)
  rsq_elastic <- cor(y, y_hat)^2
  print(i)
  plot(model_elastic)
  print(model_elastic$lambda.min)
  list_rsq <- cbind(list_rsq, rsq_elastic)
}
```

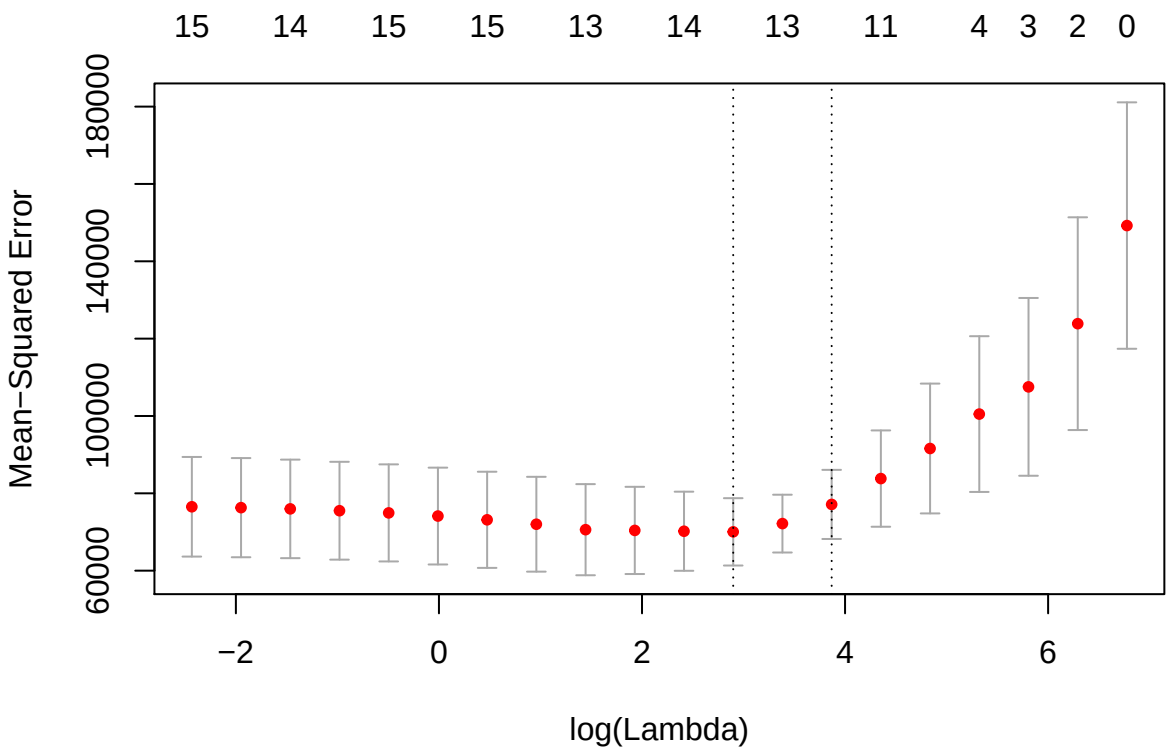
```
## [1] 0.1
```



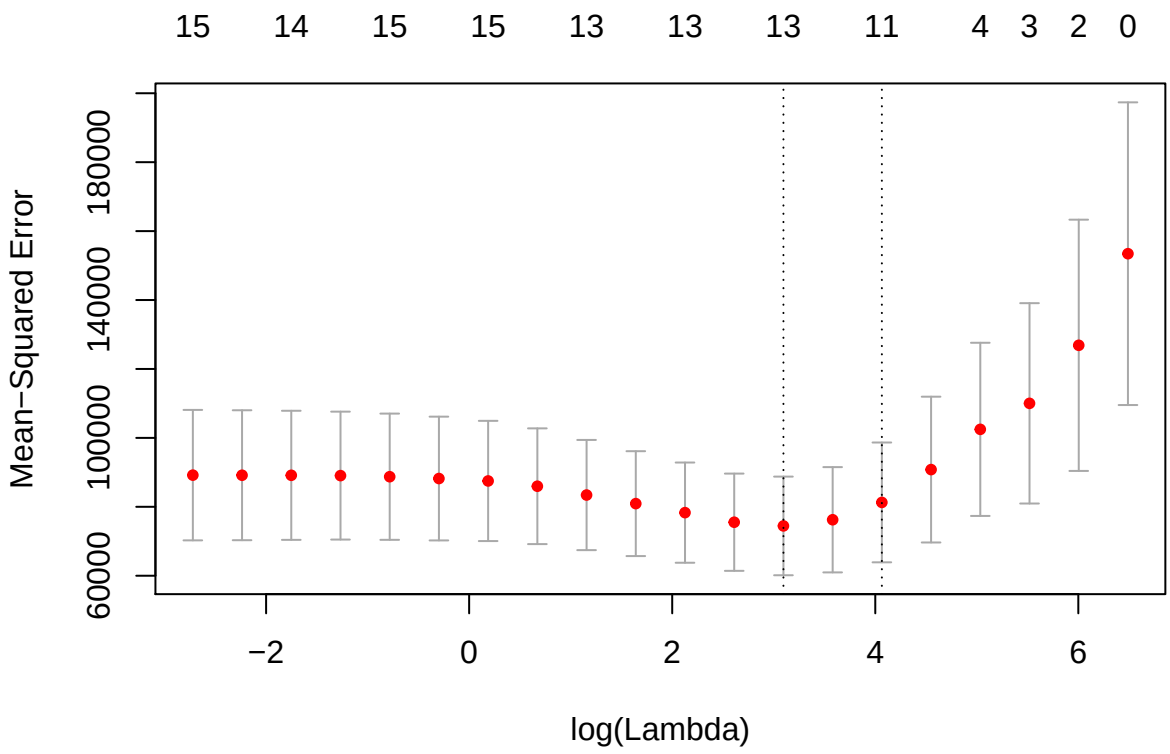

```
## [1] 33.52559
## [1] 0.2
```



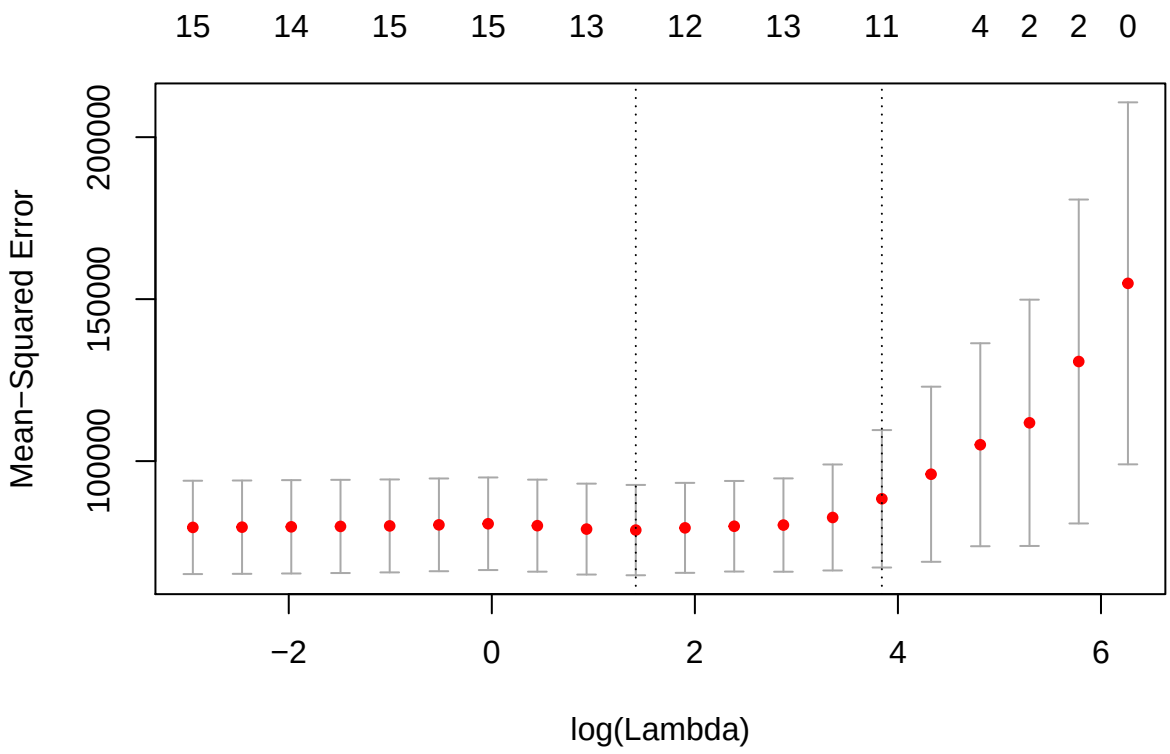

```
## [1] 27.21903
## [1] 0.3
```



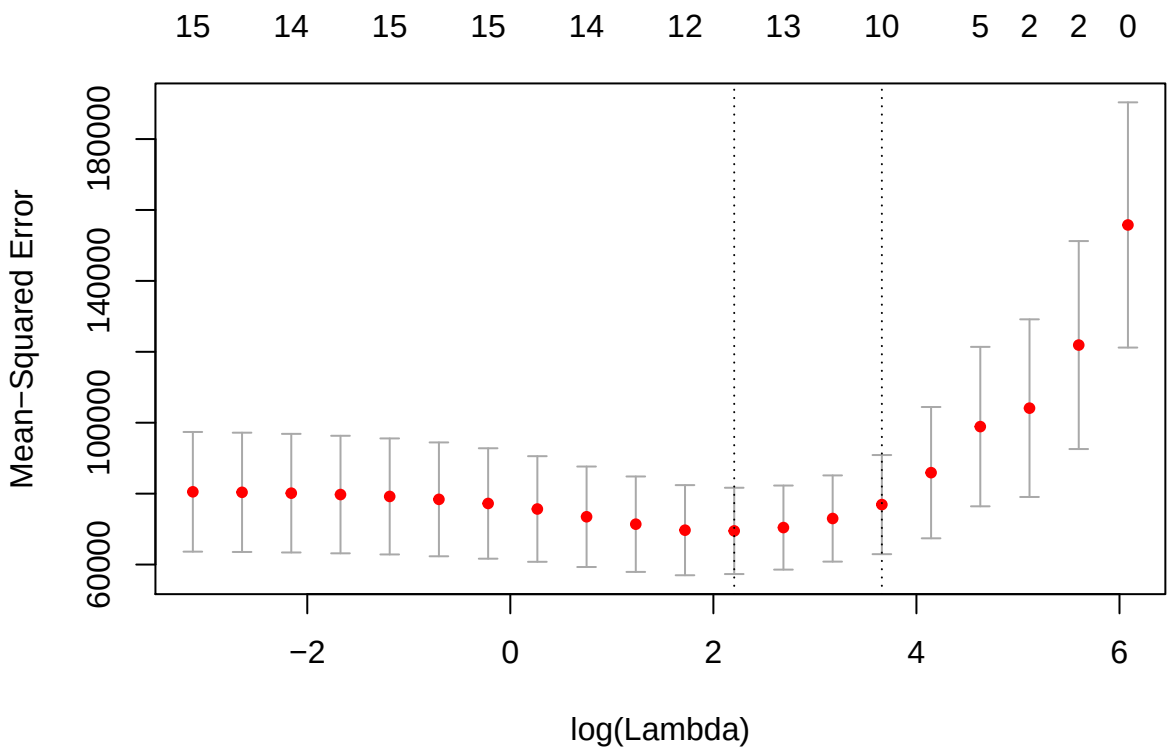

```
## [1] 18.14602
## [1] 0.4
```



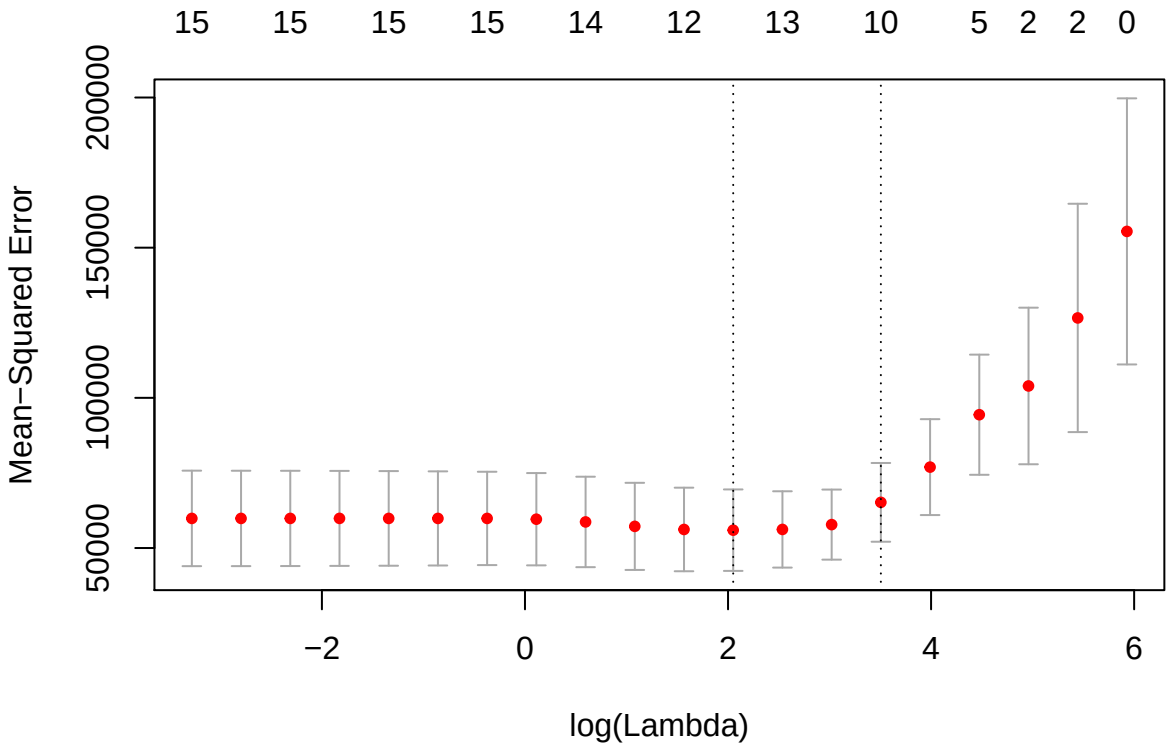

```
## [1] 22.09882
## [1] 0.5
```



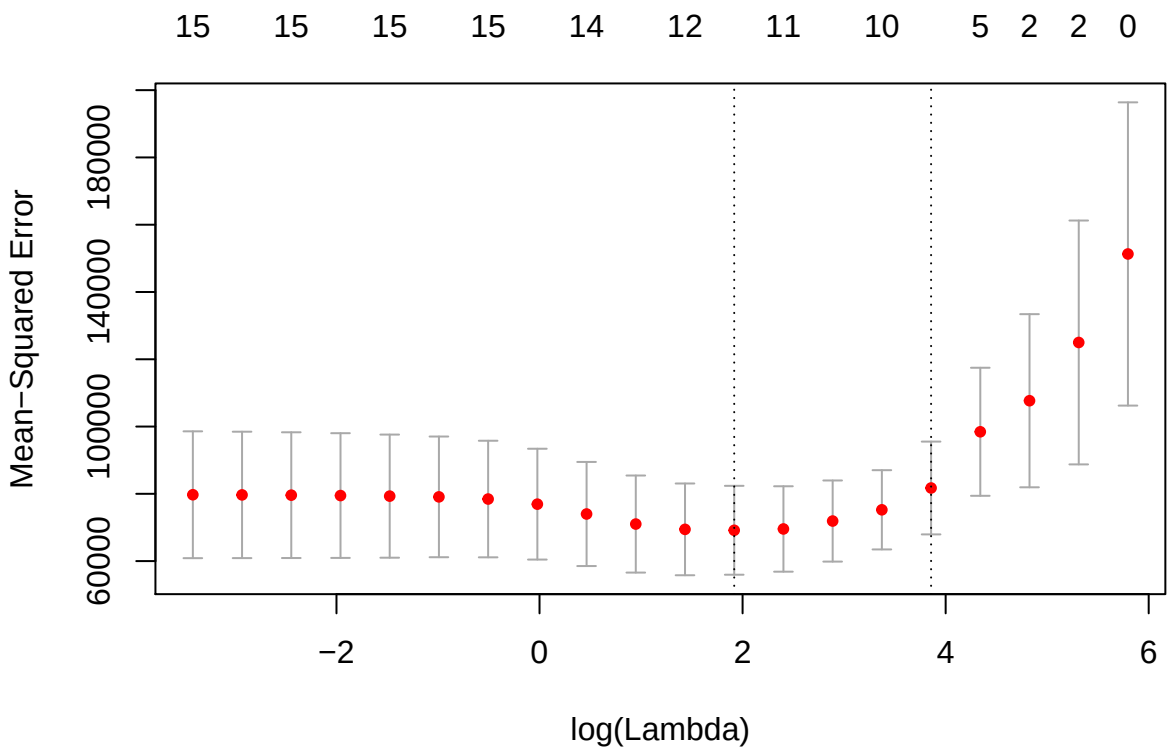

```
## [1] 4.129335
## [1] 0.6
```



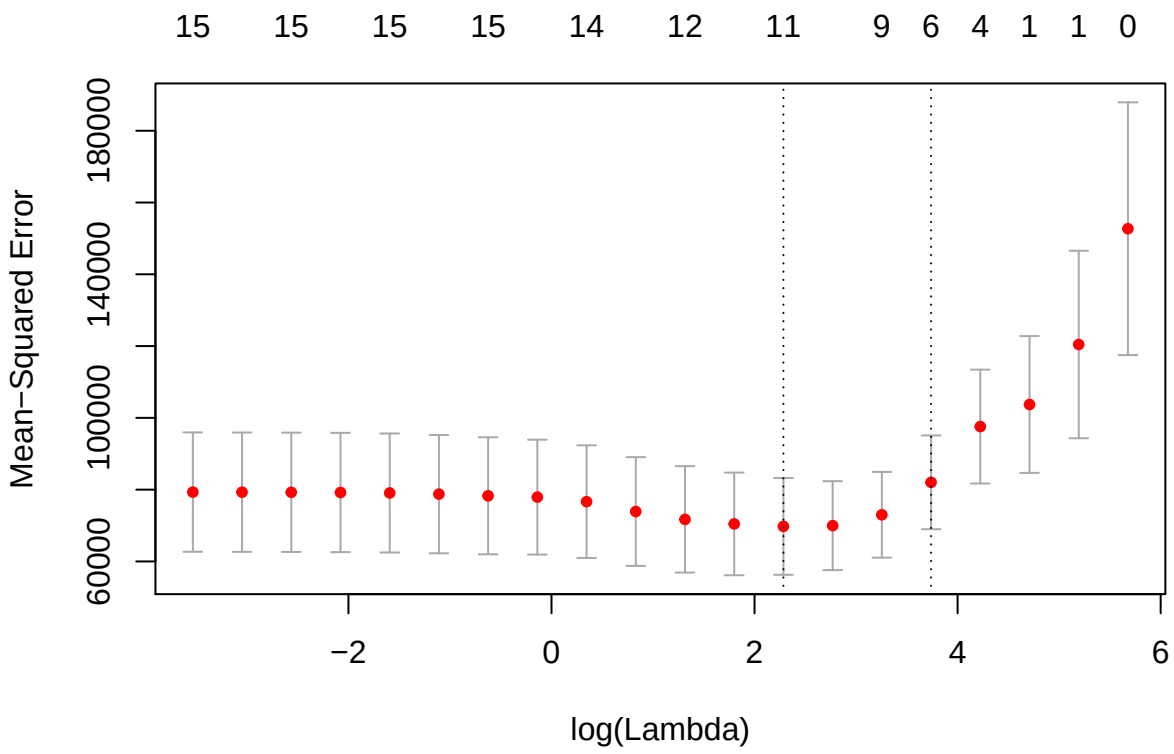

```
## [1] 9.073011
## [1] 0.7
```



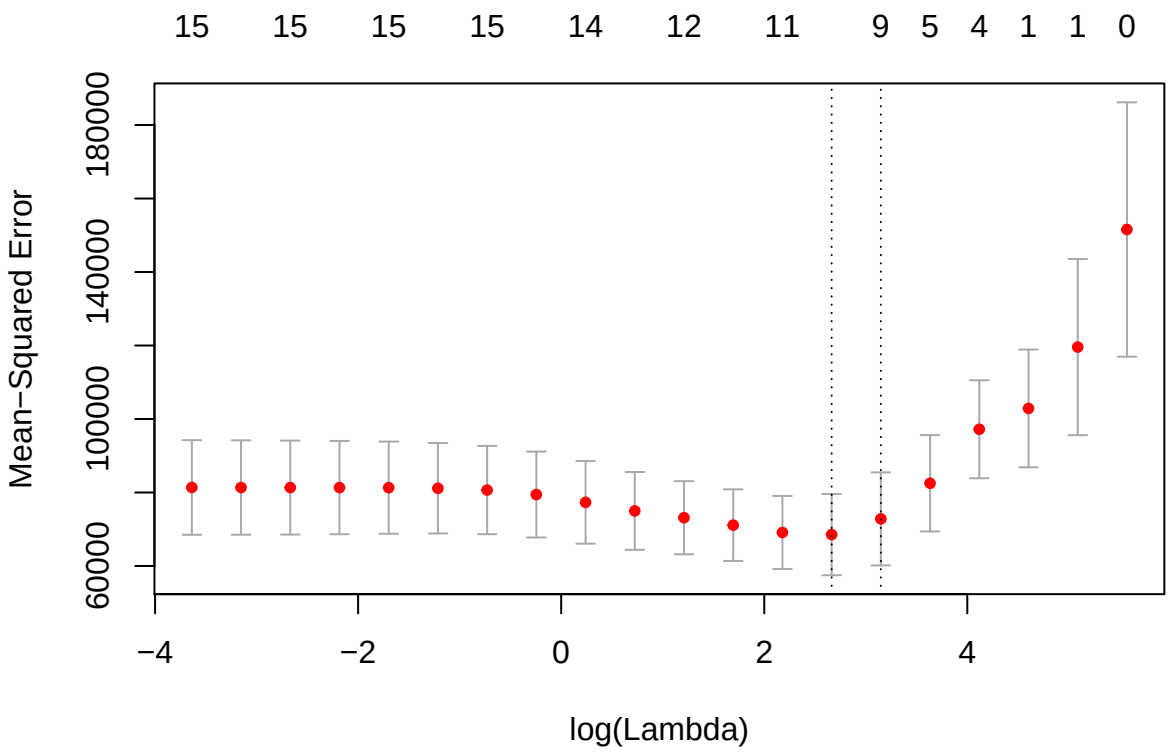

```
## [1] 7.776867
## [1] 0.8
```




```
## [1] 6.804759
## [1] 0.9
```


15 15 15 15 14 12 11 9 6 4 1 1 0
Mean−Squared Error
60000
100000
140000
180000
−2
0
2
4
6
log(Lambda)

```
## [1] 9.821697
## [1] 1
```




```
## [1] 14.35342
```

```
list_rsq
```

```
##              1         1         1         1         1         1         1
## [1,] 0.6937902 0.6714468 0.7341968 0.6956099 0.7019801 0.7068959 0.7117134
##              1         1         1
## [1,] 0.6469549 0.6510223 0.7203543
```

From the list of R square, the best R square is 0.76, where alpha = 0.3

```
# Build the model with alpha=0.3. Then find the best lambda
model_elastic_best <- cv.glmnet(x=as.matrix(df_crime_s[,-16]),
                          y=as.matrix(df_crime_s[,16]),
                          alpha=0.3,
                          nfolds=10,
                          nlambda=20,
                          type.measure='mse',
                          family='gaussian')
coef(model_elastic_best, s=model_elastic_best$lambda.min)
```

```
## 16 x 1 sparse Matrix of class "dgCMatrix"
##                       1
## (Intercept) 884.63078
```

```
## So           60.08460
## M            87.74941
## Ed          128.12725
## Po1         197.01553
## Po2          83.94984
## LF            8.48205
## M.F          63.46701
## Pop           .
## NW           19.04544
## U1          -48.82776
## U2           85.61816
## Wealth       26.67129
## Ineq        178.85583
## Prob        -87.27783
## Time          .
```

From above, the Elastic Net model will use all 15 factors, with R square as 0.76, which is not really good because Elastic Net turns out to be the most complicated model using the most factors.

Question 12.1

Describe a situation or problem from your job, everyday life, current events, etc., for which a design of experiments approach would be appropriate.

Answer:

I am working at the cruise line company. The target marketing team usually has lots of different onboard or shoreside entertainment items to recommend to the guests. They need to do lots of DOE, A/B tests to determine the selection of recommendations is the most profitable. They need to make the list short and simple and fit for different target groups otherwise people usually will not go through the full list of the options.

Question 12.2

To determine the value of 10 different yes/no features to the market value of a house (large yard, solar roof, etc.), a real estate agent plans to survey 50 potential buyers, showing a fictitious house with different combinations of features. To reduce the survey size, the agent wants to show just 16 fictitious houses. Use R's FrF2 function (in the FrF2 package) to find a fractional factorial design for this experiment: what set of features should each of the 16 fictitious houses have? Note: the output of FrF2 is "1" (include) or "-1" (don't include) for each feature.

```
# Use library FrF2 as requested.
library(FrF2)
```

```
## Loading required package: DoE.base
```

```
## Loading required package: grid
```

```
## Loading required package: conf.design
```

```
## Registered S3 method overwritten by 'partitions':
##   method            from
##   print.equivalence lava

## Registered S3 method overwritten by 'DoE.base':
##   method           from
##   factorize.factor conf.design

##
## Attaching package: 'DoE.base'

## The following objects are masked from 'package:stats':
##
##     aov, lm

## The following object is masked from 'package:graphics':
##
##     plot.design

## The following object is masked from 'package:base':
##
##     lengths
```

```
# From the question, 16 houses, 10 features, and 50 buyers
model<-FrF2(nruns = 16,nfactors = 10)
desnum(model)
```

```
##     A  B  C  D  E  F  G  H  J  K
## 1  -1 -1  1  1  1 -1 -1 -1 -1  1
## 2  -1  1 -1  1 -1  1 -1 -1 -1  1
## 3   1  1 -1 -1  1 -1 -1 -1  1  1
## 4   1 -1  1 -1 -1  1 -1 -1  1  1
## 5   1 -1 -1  1 -1 -1  1  1  1  1
## 6  -1 -1 -1  1  1  1  1 -1  1 -1
## 7  -1 -1 -1 -1  1  1  1  1 -1  1
## 8  -1  1  1  1 -1 -1  1 -1  1 -1
## 9   1 -1 -1 -1 -1 -1  1 -1 -1 -1
## 10  1 -1  1  1 -1  1 -1  1 -1 -1
## 11 -1  1  1 -1 -1 -1  1  1 -1  1
## 12 -1  1 -1 -1 -1  1 -1  1  1 -1
## 13  1  1 -1  1  1 -1 -1  1 -1 -1
## 14  1  1  1  1  1  1  1  1  1  1
## 15 -1 -1  1 -1  1 -1 -1  1  1 -1
## 16  1  1  1 -1  1  1  1 -1 -1 -1
```

From above output matrix, for 16 different houses from 1 to 16, for the 10 features from A-K, use 1 (include) and -1 (not include) as listed to do the DOE.

Question 13.1

For each of the following distributions, give an example of data that you would expect to follow this distribution (besides the examples already discussed in class). a. Binomial b. Geometric c. Poisson d. Exponential e. Weibull

Answer:

1. Binomial Assuming the chance if a flight is going to be arrived ontime/delayed at the airport is fixed. The distribution of the onetime/delay of all the flights over certain period at the airport
2. Geometric For the flight ontime/delayed arrival, how many flights are needed before there is a delayed arrival.
3. Poisson At the airport, the flight arrivals are independent and identically distributed. The distribution of the flight arrival is Poisson.
4. Exponential The inter-arrival time of the flight arrival at the airport
5. Weibull The time between each delayed flights at the airport.