H5-ISYE6501x-Summer2018-OA

OA

6/16/2018

Objectives

The first area of focus of Week5's homework is to test the course material for week 5. Primarily, variable selection methodologies (like stepwise, lasso, elasticnet), critical thinking around design of experiments such as A/B testing as applicable in our lives so as to internalize the concept, reducing massive amount of permutations to run through A/B testing by using factorial design methods and again, finding real life examples that fit various mathemetical distributions discussed like binomial, poisson, geometric etc.

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.

- The overall approach we will take in implmenting (one variant of) stepwise regression is:

1. to start with no factors, and each step we'll be removing or adding a factor..
2. initially, we run the regression model by adding one variable.
3. We see each added factors efficacy by looking at its p-value
   - if all of them are effective, we keep it and add another (loop to the top)
   - if any of them is ineffective (p>0.15) we prune it.
4. we loop through all the factors until we drain them.

- First we download and load the data

```
dataFile <- "uscrime.txt"
if (!file.exists(dataFile)) {
  #crimeDataURL <- paste0(c("http://www.statsci.org/data/general/uscrime.txt"))
  crimeDataURL <- paste0(c("https://prod-edxapp.edx-cdn.org/assets/courseware/v1/17b85cea5d0e613bf08025

  download.file(crimeDataURL, dataFile) }

crimeDataTable <- read.table(dataFile, header = TRUE )
```

we will use forward step-wise regression:

note, we are using the step() function, which chooses a model based on the lowest AIC score..

```
stepWiseModel <- lm(Crime~1, data = crimeDataTable)
step(stepWiseModel, scope = formula(lm(Crime~., data = crimeDataTable)), direction = "forward")
```

```
## Start:  AIC=561.02
## Crime ~ 1
##
##           Df Sum of Sq     RSS    AIC
## + Po1      1   3253302 3627626 532.94
## + Po2      1   3058626 3822302 535.39
## + Wealth   1   1340152 5540775 552.84
## + Prob     1   1257075 5623853 553.54
```

```
## + Pop     1    783660 6097267 557.34
## + Ed      1    717146 6163781 557.85
## + M.F     1    314867 6566061 560.82
## <none>                6880928 561.02
## + LF      1    245446 6635482 561.32
## + Ineq    1    220530 6660397 561.49
## + U2      1    216354 6664573 561.52
## + Time    1    154545 6726383 561.96
## + So      1     56527 6824400 562.64
## + M       1     55084 6825844 562.65
## + U1      1     17533 6863395 562.90
## + NW      1      7312 6873615 562.97
##
## Step:  AIC=532.94
## Crime ~ Po1
##
##          Df Sum of Sq     RSS    AIC
## + Ineq    1    739819 2887807 524.22
## + M       1    616741 3010885 526.18
## + M.F     1    250522 3377104 531.57
## + NW      1    232434 3395192 531.82
## + So      1    219098 3408528 532.01
## + Wealth  1    180872 3446754 532.53
## <none>                3627626 532.94
## + Po2     1    146167 3481459 533.00
## + Prob    1     92278 3535348 533.72
## + LF      1     77479 3550147 533.92
## + Time    1     43185 3584441 534.37
## + U2      1     17848 3609778 534.70
## + Pop     1      5666 3621959 534.86
## + U1      1      2878 3624748 534.90
## + Ed      1       767 3626859 534.93
##
## Step:  AIC=524.22
## Crime ~ Po1 + Ineq
##
##          Df Sum of Sq     RSS    AIC
## + Ed      1    587050 2300757 515.53
## + M.F     1    454545 2433262 518.17
## + Prob    1    280690 2607117 521.41
## + LF      1    260571 2627236 521.77
## + Wealth  1    213937 2673871 522.60
## + M       1    181236 2706571 523.17
## + Pop     1    130377 2757430 524.04
## <none>                2887807 524.22
## + NW      1     36439 2851369 525.62
## + So      1     33738 2854069 525.66
## + Po2     1     30673 2857134 525.71
## + U1      1      2309 2885498 526.18
## + Time    1       497 2887310 526.21
## + U2      1       253 2887554 526.21
##
## Step:  AIC=515.53
## Crime ~ Po1 + Ineq + Ed
```

```
##
##          Df Sum of Sq     RSS    AIC
## + M       1    239405 2061353 512.37
## + Prob    1    234981 2065776 512.47
## + M.F     1    117026 2183731 515.08
## <none>                2300757 515.53
## + Wealth  1     79540 2221218 515.88
## + U2      1     62112 2238646 516.25
## + Time    1     61770 2238987 516.26
## + Po2     1     42584 2258174 516.66
## + Pop     1     39319 2261438 516.72
## + U1      1      7365 2293392 517.38
## + LF      1      7254 2293503 517.39
## + NW      1      4210 2296547 517.45
## + So      1      4135 2296622 517.45
##
## Step:  AIC=512.37
## Crime ~ Po1 + Ineq + Ed + M
##
##          Df Sum of Sq     RSS    AIC
## + Prob    1    258063 1803290 508.08
## + U2      1    200988 1860365 509.55
## + Wealth  1    163378 1897975 510.49
## <none>                2061353 512.37
## + M.F     1     74398 1986955 512.64
## + U1      1     50835 2010518 513.20
## + Po2     1     45392 2015961 513.32
## + Time    1     42746 2018607 513.39
## + NW      1     16488 2044865 513.99
## + Pop     1      8101 2053251 514.19
## + So      1      3189 2058164 514.30
## + LF      1      2988 2058365 514.30
##
## Step:  AIC=508.08
## Crime ~ Po1 + Ineq + Ed + M + Prob
##
##          Df Sum of Sq     RSS    AIC
## + U2      1    192233 1611057 504.79
## + Wealth  1     86490 1716801 507.77
## + M.F     1     84509 1718781 507.83
## <none>                1803290 508.08
## + U1      1     52313 1750977 508.70
## + Pop     1     47719 1755571 508.82
## + Po2     1     37967 1765323 509.08
## + So      1     21971 1781320 509.51
## + Time    1     10194 1793096 509.82
## + LF      1       990 1802301 510.06
## + NW      1       797 1802493 510.06
##
## Step:  AIC=504.79
## Crime ~ Po1 + Ineq + Ed + M + Prob + U2
##
##          Df Sum of Sq     RSS    AIC
## <none>                1611057 504.79
```

```
## + Wealth  1     59910 1551147 505.00
## + U1      1     54830 1556227 505.16
## + Pop     1     51320 1559737 505.26
## + M.F     1     30945 1580112 505.87
## + Po2     1     25017 1586040 506.05
## + So      1     17958 1593098 506.26
## + LF      1     13179 1597878 506.40
## + Time    1      7159 1603898 506.58
## + NW      1       359 1610698 506.78
```

```
##
## Call:
## lm.default(formula = Crime ~ Po1 + Ineq + Ed + M + Prob + U2,
##     data = crimeDataTable)
##
## Coefficients:
## (Intercept)          Po1         Ineq           Ed            M
##    -5040.50       115.02        67.65       196.47       105.02
##        Prob           U2
##    -3801.84        89.37
```

From the output above, we can see that the forward step-wise regression model has found us 6 optimum parameters with an AIC score of 504.79.

Just for archival purposes, we also will do a forward & backward step-wise regression here to see if the results are better:

```
swModelForwardBackward <- lm(Crime~1 , data = crimeDataTable)
step(swModelForwardBackward, scope = list(lower = formula(lm(Crime~1, data=crimeDataTable)),
                                          upper = formula(lm(Crime~., data=crimeDataTable))),
                                          direction = "both")
```

```
## Start:  AIC=561.02
## Crime ~ 1
##
##          Df Sum of Sq     RSS    AIC
## + Po1     1   3253302 3627626 532.94
## + Po2     1   3058626 3822302 535.39
## + Wealth  1   1340152 5540775 552.84
## + Prob    1   1257075 5623853 553.54
## + Pop     1    783660 6097267 557.34
## + Ed      1    717146 6163781 557.85
## + M.F     1    314867 6566061 560.82
## <none>                6880928 561.02
## + LF      1    245446 6635482 561.32
## + Ineq    1    220530 6660397 561.49
## + U2      1    216354 6664573 561.52
## + Time    1    154545 6726383 561.96
## + So      1     56527 6824400 562.64
## + M       1     55084 6825844 562.65
## + U1      1     17533 6863395 562.90
## + NW      1      7312 6873615 562.97
##
## Step:  AIC=532.94
## Crime ~ Po1
##
```

```
##          Df Sum of Sq     RSS    AIC
## + Ineq    1    739819 2887807 524.22
## + M       1    616741 3010885 526.18
## + M.F     1    250522 3377104 531.57
## + NW      1    232434 3395192 531.82
## + So      1    219098 3408528 532.01
## + Wealth  1    180872 3446754 532.53
## <none>                3627626 532.94
## + Po2     1    146167 3481459 533.00
## + Prob    1     92278 3535348 533.72
## + LF      1     77479 3550147 533.92
## + Time    1     43185 3584441 534.37
## + U2      1     17848 3609778 534.70
## + Pop     1      5666 3621959 534.86
## + U1      1      2878 3624748 534.90
## + Ed      1       767 3626859 534.93
## - Po1     1   3253302 6880928 561.02
##
## Step:  AIC=524.22
## Crime ~ Po1 + Ineq
##
##          Df Sum of Sq     RSS    AIC
## + Ed      1    587050 2300757 515.53
## + M.F     1    454545 2433262 518.17
## + Prob    1    280690 2607117 521.41
## + LF      1    260571 2627236 521.77
## + Wealth  1    213937 2673871 522.60
## + M       1    181236 2706571 523.17
## + Pop     1    130377 2757430 524.04
## <none>                2887807 524.22
## + NW      1     36439 2851369 525.62
## + So      1     33738 2854069 525.66
## + Po2     1     30673 2857134 525.71
## + U1      1      2309 2885498 526.18
## + Time    1       497 2887310 526.21
## + U2      1       253 2887554 526.21
## - Ineq    1    739819 3627626 532.94
## - Po1     1   3772590 6660397 561.49
##
## Step:  AIC=515.53
## Crime ~ Po1 + Ineq + Ed
##
##          Df Sum of Sq     RSS    AIC
## + M       1    239405 2061353 512.37
## + Prob    1    234981 2065776 512.47
## + M.F     1    117026 2183731 515.08
## <none>                2300757 515.53
## + Wealth  1     79540 2221218 515.88
## + U2      1     62112 2238646 516.25
## + Time    1     61770 2238987 516.26
## + Po2     1     42584 2258174 516.66
## + Pop     1     39319 2261438 516.72
## + U1      1      7365 2293392 517.38
## + LF      1      7254 2293503 517.39
```

```
## + NW      1      4210 2296547 517.45
## + So      1      4135 2296622 517.45
## - Ed      1    587050 2887807 524.22
## - Ineq    1   1326101 3626859 534.93
## - Po1     1   3782666 6083423 559.23
##
## Step:  AIC=512.37
## Crime ~ Po1 + Ineq + Ed + M
##
##          Df Sum of Sq     RSS    AIC
## + Prob    1    258063 1803290 508.08
## + U2      1    200988 1860365 509.55
## + Wealth  1    163378 1897975 510.49
## <none>                2061353 512.37
## + M.F     1     74398 1986955 512.64
## + U1      1     50835 2010518 513.20
## + Po2     1     45392 2015961 513.32
## + Time    1     42746 2018607 513.39
## + NW      1     16488 2044865 513.99
## + Pop     1      8101 2053251 514.19
## + So      1      3189 2058164 514.30
## + LF      1      2988 2058365 514.30
## - M       1    239405 2300757 515.53
## - Ed      1    645219 2706571 523.17
## - Ineq    1    864671 2926024 526.83
## - Po1     1   4000849 6062202 561.07
##
## Step:  AIC=508.08
## Crime ~ Po1 + Ineq + Ed + M + Prob
##
##          Df Sum of Sq     RSS    AIC
## + U2      1    192233 1611057 504.79
## + Wealth  1     86490 1716801 507.77
## + M.F     1     84509 1718781 507.83
## <none>                1803290 508.08
## + U1      1     52313 1750977 508.70
## + Pop     1     47719 1755571 508.82
## + Po2     1     37967 1765323 509.08
## + So      1     21971 1781320 509.51
## + Time    1     10194 1793096 509.82
## + LF      1       990 1802301 510.06
## + NW      1       797 1802493 510.06
## - Prob    1    258063 2061353 512.37
## - M       1    262486 2065776 512.47
## - Ed      1    598315 2401605 519.55
## - Ineq    1    968199 2771489 526.28
## - Po1     1   3268577 5071868 554.69
##
## Step:  AIC=504.79
## Crime ~ Po1 + Ineq + Ed + M + Prob + U2
##
##          Df Sum of Sq     RSS    AIC
## <none>                1611057 504.79
## + Wealth  1     59910 1551147 505.00
```

```
## + U1      1     54830 1556227 505.16
## + Pop     1     51320 1559737 505.26
## + M.F     1     30945 1580112 505.87
## + Po2     1     25017 1586040 506.05
## + So      1     17958 1593098 506.26
## + LF      1     13179 1597878 506.40
## + Time    1      7159 1603898 506.58
## + NW      1       359 1610698 506.78
## - U2      1    192233 1803290 508.08
## - Prob    1    249308 1860365 509.55
## - M       1    400611 2011667 513.22
## - Ed      1    776207 2387264 521.27
## - Ineq    1    949221 2560278 524.56
## - Po1     1   2817067 4428124 550.31
```

```
##
## Call:
## lm.default(formula = Crime ~ Po1 + Ineq + Ed + M + Prob + U2,
##     data = crimeDataTable)
##
## Coefficients:
## (Intercept)          Po1         Ineq           Ed            M
##    -5040.50       115.02        67.65       196.47       105.02
##        Prob           U2
##    -3801.84        89.37
```

As we can see, both forward and fw+bck regressions us the same exact result, so let's create a model using these vars:

```
swModel2 <- lm (formula = Crime ~ Po1 + Ineq + Ed + M + Prob + U2, data = crimeDataTable)
summary(swModel2)
```

```
##
## Call:
## lm.default(formula = Crime ~ Po1 + Ineq + Ed + M + Prob + U2,
##     data = crimeDataTable)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -470.68  -78.41  -19.68  133.12  556.23
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -5040.50     899.84  -5.602 1.72e-06 ***
## Po1           115.02      13.75   8.363 2.56e-10 ***
## Ineq           67.65      13.94   4.855 1.88e-05 ***
## Ed            196.47      44.75   4.390 8.07e-05 ***
## M             105.02      33.30   3.154  0.00305 **
## Prob        -3801.84    1528.10  -2.488  0.01711 *
## U2             89.37      40.91   2.185  0.03483 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 200.7 on 40 degrees of freedom
## Multiple R-squared:  0.7659, Adjusted R-squared:  0.7307
## F-statistic: 21.81 on 6 and 40 DF,  p-value: 3.418e-11
```

Sweet adjusted R-squared of 73% and all the variables have really low p-values, so they are useful covariants. We have a winner!

Lasso

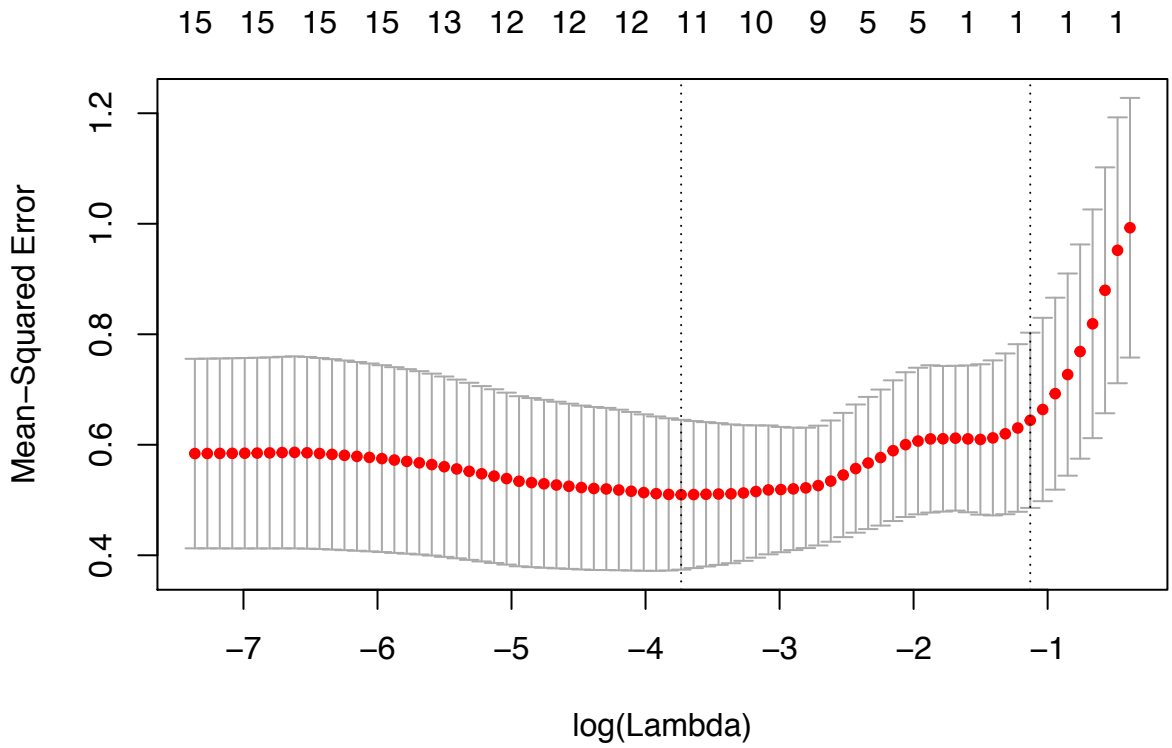
prior to doing Lasso, we need to scale the data first...

we actually standardize instead of scaling as the former does not alter the T and p values the way I understand this is that when you scale the data, you have to scale each variable between its extremem possible bounded values (a and b), vs in the case of standardization you apply the standard approach of (x-u)/sd to all values, so you are just normalizing them...

```
scaledData <- scale(crimeDataTable)
```

- now we run the cv.glmnet() function with alpha (or as in the notes, lambda =1, which makes this model lasso
- by the way, anything 0<x<1 will make this same model elasticnet!

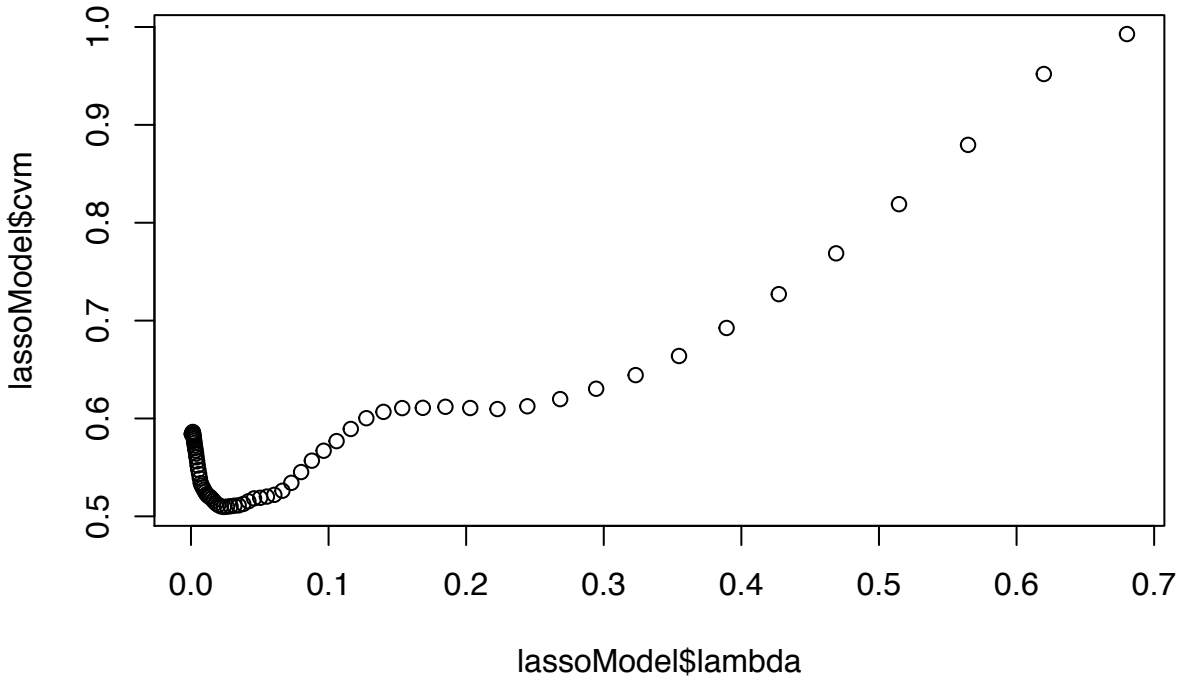
```
lassoModel <- cv.glmnet(x = as.matrix(scaledData[,-16]),
                        y = as.matrix(scaledData[,16]),
                        alpha = 1,
                        nfolds = 5,
                        type.measure = "mse",
                        family = "gaussian")


plot(lassoModel)#
```




```
#cbind(lassoModel$lambda, lassoModel$cvm)
```

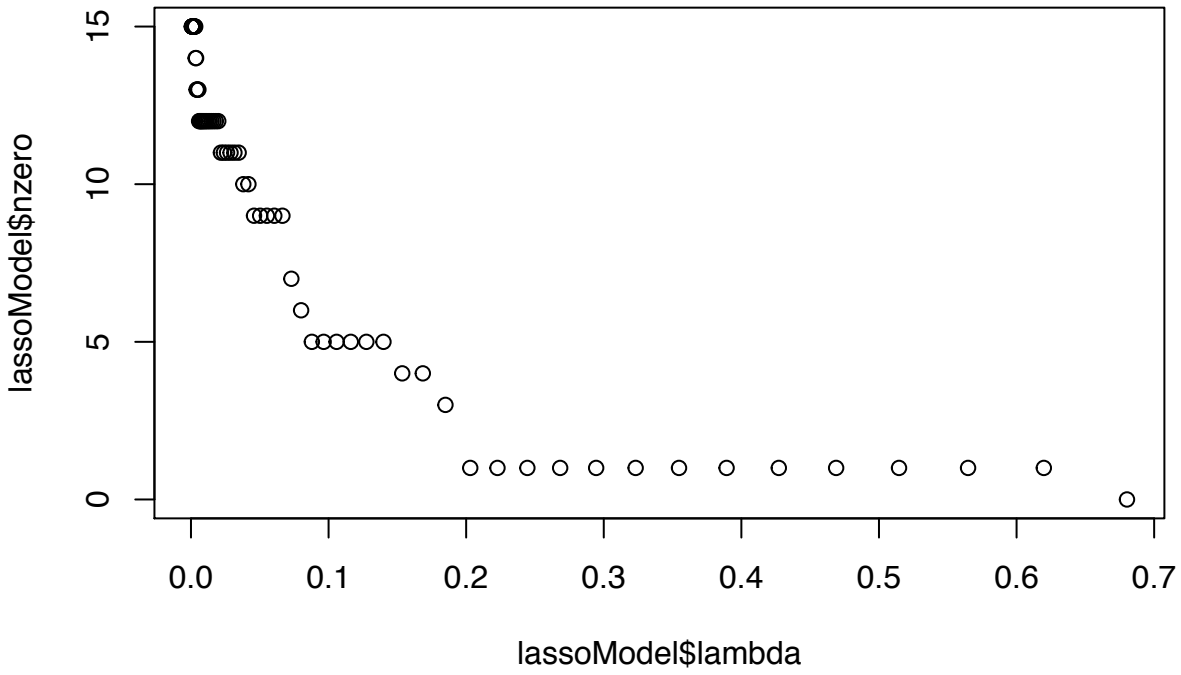
- so for each of the lambda values, the cv.glmnet() generates co-efficients, and if you plot it against the mse (mean squared error) you can find the most optimum lambda value.
- note: these numbers are way smaller than what the TA showed, since i scaled the data and he didnt

```
plot(lassoModel$lambda, lassoModel$cvm)
```



- another plot we can show is how many co-efficients it choses for each lambda value. th

```
plot(lassoModel$lambda, lassoModel$nzero)
```



- we can see that for the higher lambda, it picks a couple of co-efficients only, which is why it has a higher MSE
- the better MSE models, we have more coefficients
- let's find out the co-efficients:

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

```
## 16 x 1 sparse Matrix of class "dgCMatrix"
##                              1
## (Intercept) -3.074901e-16
## M            2.309848e-01
## So           5.507658e-02
## Ed           3.569551e-01
## Po1          7.892436e-01
## Po2          .
## LF           .
## M.F          1.425856e-01
## Pop          .
## NW           1.598125e-02
## U1          -9.390151e-02
## U2           1.858803e-01
## Wealth       1.242575e-02
## Ineq         4.961557e-01
## Prob        -2.160509e-01
## Time         .
```

so let's use these suggested coefficients (that is the ones that are not blank!) into a lm and assses quality:

```
lassoModelWithTightenedCoefficients <- lm (formula =  Crime ~ M + So + Ed + Po1 + LF + M.F + NW + U1 + U

summary(lassoModelWithTightenedCoefficients)
```

```
##
## Call:
## lm.default(formula = Crime ~ M + So + Ed + Po1 + LF + M.F + NW +
##     U1 + U2 + Ineq + Prob, data = crimeDataTable)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -443.2 -101.4    4.1  120.5  486.2
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -6434.101   1253.263  -5.134 1.07e-05 ***
## M              84.825     39.221   2.163  0.03747 *
## So             36.573    139.615   0.262  0.79489
## Ed            186.954     58.101   3.218  0.00278 **
## Po1            99.463     18.338   5.424 4.44e-06 ***
## LF           -264.646   1339.041  -0.198  0.84447
## M.F            25.438     17.353   1.466  0.15159
## NW              1.265      5.783   0.219  0.82814
## U1          -6050.130   3977.786  -1.521  0.13725
## U2            179.349     78.140   2.295  0.02783 *
## Ineq           58.402     16.962   3.443  0.00151 **
## Prob        -4222.327   1740.886  -2.425  0.02059 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 202.9 on 35 degrees of freedom
## Multiple R-squared:  0.7906, Adjusted R-squared:  0.7248
```

```
## F-statistic: 12.01 on 11 and 35 DF,  p-value: 6.965e-09
```

as we can see, Lf, NW have pretty lousy p-values.

However I did notice that lasso's coefficients for these p-values were atleast 1 order of magnitude smaller than all others...

So 5.938496e-02 LF 5.762247e-03 NW 9.368004e-03

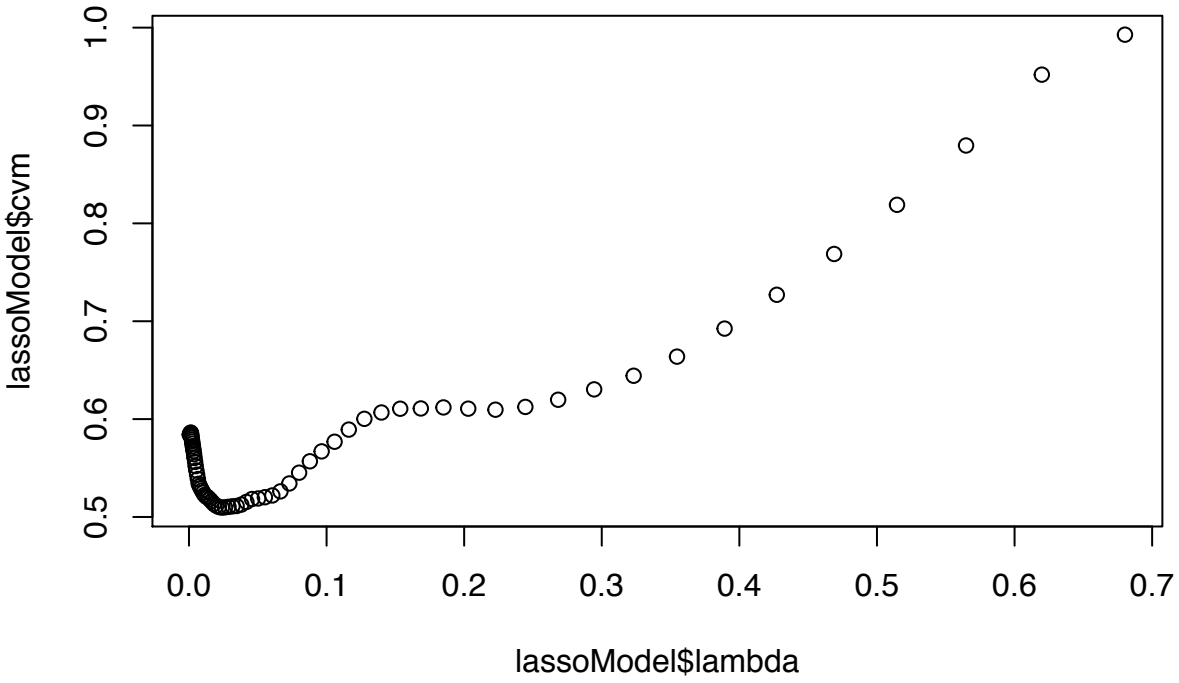
so my final LASSO model would be to to drop these bad variables:

```
lassoModelWithTightenedCoefficientsFINAL <- lm (formula =  Crime ~ M + Ed + Po1 +  M.F + U1 + U2 + Ineq

summary(lassoModelWithTightenedCoefficientsFINAL)
```

```
##
## Call:
## lm.default(formula = Crime ~ M + Ed + Po1 + M.F + U1 + U2 + Ineq +
##     Prob, data = crimeDataTable)
##
## 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) -6426.10    1194.61  -5.379 4.04e-06 ***
## M              93.32      33.50   2.786  0.00828 **
## Ed            180.12      52.75   3.414  0.00153 **
## Po1           102.65      15.52   6.613 8.26e-08 ***
## M.F            22.34      13.60   1.642  0.10874
## U1          -6086.63    3339.27  -1.823  0.07622 .
## U2            187.35      72.48   2.585  0.01371 *
## Ineq           61.33      13.96   4.394 8.63e-05 ***
## Prob        -3796.03    1490.65  -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
```

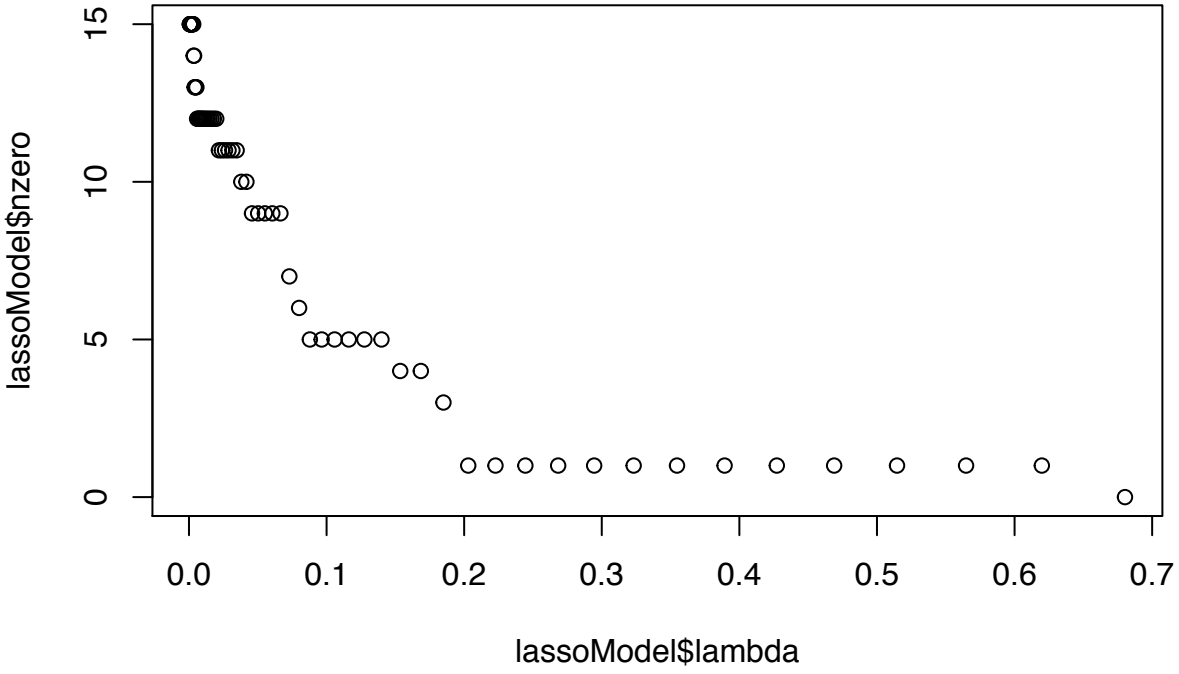
- output is MUCH better. all vars have low p-vales and the adj R-sq is 74% pretty healthy too.
- so for each of the lambda values, the cv.glmnet() generates co-efficients, and if you plot it against the mse (mean squared error) you can find the most optimum lambda value.
- note: these numbers are many fold way smaller than what the TA showed, since i scaled the data and he didnt

```
plot(lassoModel$lambda, lassoModel$cvm)
```



- another plot we can show is how many co-efficients it choses for each lambda value. th

```
plot(lassoModel$lambda, lassoModel$nzero)
```



- we can see that for the higher lambda, it picks a couple of co-efficients only, which is why it has a higher MSE
- the better MSE models, we have more coefficients
- let's find out the co-efficients:

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

```
## 16 x 1 sparse Matrix of class "dgCMatrix"
```

```
##                          1
## (Intercept) -3.074901e-16
## M            2.309848e-01
## So           5.507658e-02
## Ed           3.569551e-01
## Po1          7.892436e-01
## Po2          .
## LF           .
## M.F          1.425856e-01
## Pop          .
## NW           1.598125e-02
## U1          -9.390151e-02
## U2           1.858803e-01
## Wealth       1.242575e-02
## Ineq         4.961557e-01
## Prob        -2.160509e-01
## Time         .
```

so let's use these suggested coefficients (that is the ones that are not blank!) into a lm and assses quality:

```
lassoModelWithTightenedCoefficients <- lm (formula =  Crime ~ M + So + Ed + Po1 + LF + M.F + NW + U1 + U

summary(lassoModelWithTightenedCoefficients)
```

```
##
## Call:
## lm.default(formula = Crime ~ M + So + Ed + Po1 + LF + M.F + NW +
##     U1 + U2 + Ineq + Prob, data = crimeDataTable)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -443.2 -101.4    4.1  120.5  486.2
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -6434.101   1253.263  -5.134 1.07e-05 ***
## M              84.825     39.221   2.163  0.03747 *
## So             36.573    139.615   0.262  0.79489
## Ed            186.954     58.101   3.218  0.00278 **
## Po1            99.463     18.338   5.424 4.44e-06 ***
## LF           -264.646   1339.041  -0.198  0.84447
## M.F            25.438     17.353   1.466  0.15159
## NW              1.265      5.783   0.219  0.82814
## U1          -6050.130   3977.786  -1.521  0.13725
## U2            179.349     78.140   2.295  0.02783 *
## Ineq           58.402     16.962   3.443  0.00151 **
## Prob        -4222.327   1740.886  -2.425  0.02059 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 202.9 on 35 degrees of freedom
## Multiple R-squared:  0.7906, Adjusted R-squared:  0.7248
## F-statistic: 12.01 on 11 and 35 DF,  p-value: 6.965e-09
```

as we can see, Lf, NW have pretty lousy p-values.

However I did notice that lasso's coefficients for these p-values were atleast 1 order of magnitude smaller than all others...

So 5.938496e-02 LF 5.762247e-03 NW 9.368004e-03

so my final LASSO model would be to to drop these bad variables:

```
lassoModelWithTightenedCoefficientsFINAL <- lm (formula =  Crime ~ M + Ed + Po1 +  M.F + U1 + U2 + Ineq

summary(lassoModelWithTightenedCoefficientsFINAL)
```

```
##
## Call:
## lm.default(formula = Crime ~ M + Ed + Po1 + M.F + U1 + U2 + Ineq +
##     Prob, data = crimeDataTable)
##
## 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) -6426.10    1194.61  -5.379 4.04e-06 ***
## M              93.32      33.50   2.786  0.00828 **
## Ed            180.12      52.75   3.414  0.00153 **
## Po1           102.65      15.52   6.613 8.26e-08 ***
## M.F            22.34      13.60   1.642  0.10874
## U1          -6086.63    3339.27  -1.823  0.07622 .
## U2            187.35      72.48   2.585  0.01371 *
## Ineq           61.33      13.96   4.394 8.63e-05 ***
## Prob        -3796.03    1490.65  -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
```

- output is MUCH better. all vars have low p-vales and the adj R-sq is 74% pretty healthy too.

ElasticNET

- we will now attempt to loop over alpha values to find the minimum lambda

```
set.seed(1)
alphaBase <- 0.1 ;
alphaWhichGivesMinimumLambdas <- c()
for (i in 1:9) {

elasticModel <- cv.glmnet(x = as.matrix(scaledData[,-16]),
                          y = as.matrix(scaledData[,16]),
                          alpha = i*alphaBase,
                          nfolds = 5,
                          type.measure = "mse",
                          family = "gaussian")
```

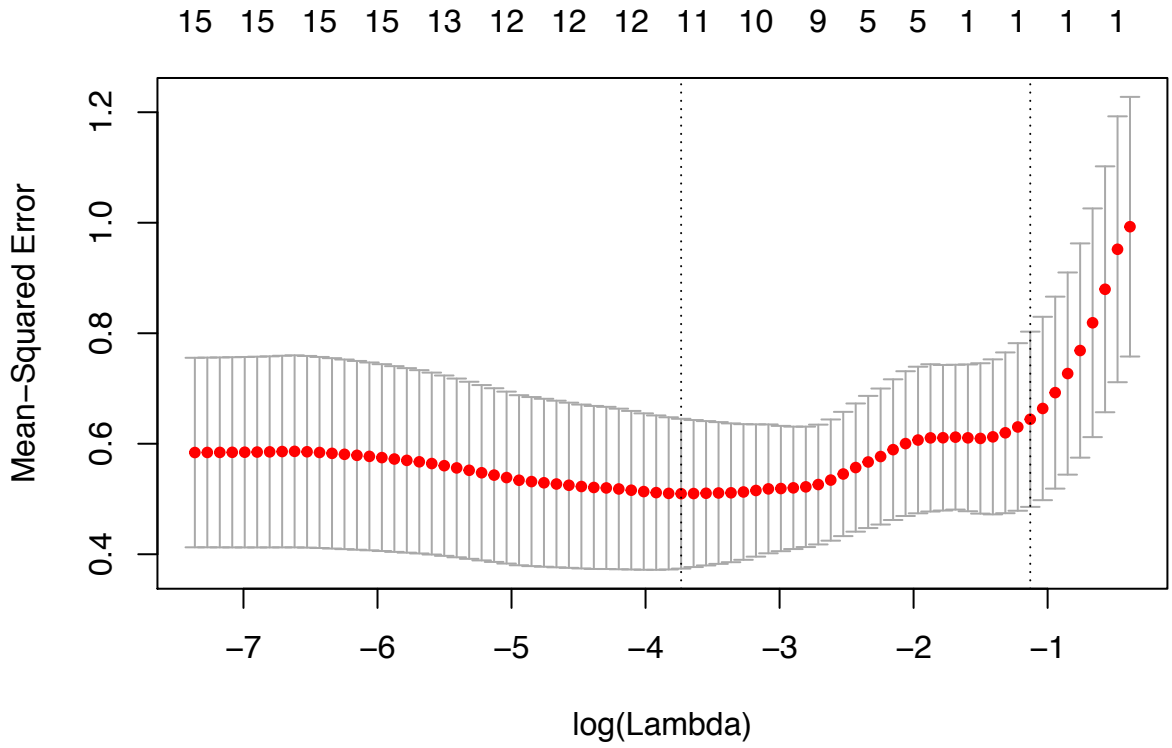
```
alphaWhichGivesMinimumLambdas[i] <- elasticModel$lambda.min

}
print("the alpha which gives the minimum lambda is :")
```

```
## [1] "the alpha which gives the minimum lambda is :"
```

```
which.min(alphaWhichGivesMinimumLambdas)*alphaBase
```

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

```
plot(lassoModel)#
```




```
#cbind(lassoModel$lambda, lassoModel$cvm)
```

using seed as 1, the alpha which gets us the minimal mean squared error out of all models is 0.6 and we can confirm this by doing log(0.6) = -0.22 which is about the lowest part of the curve above as well.

so , now equipped with our alpha , we just run the elasticNet one more time to craft the model and evaluate the coefficients it recommends:

```
elasticModelWithBestAlpha <- cv.glmnet(x = as.matrix(scaledData[,-16]),
                        y = as.matrix(scaledData[,16]),
                        alpha = which.min(alphaWhichGivesMinimumLambdas)*alphaBase,
                        nfolds = 5,
                        type.measure = "mse",
                        family = "gaussian")

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

```
## 16 x 1 sparse Matrix of class "dgCMatrix"
##                          1
## (Intercept) -3.758700e-16
```

```
## M            1.410817e-01
## So           2.959117e-02
## Ed           1.106166e-01
## Po1          5.294965e-01
## Po2          1.981124e-01
## LF           5.680589e-03
## M.F          1.414120e-01
## Pop          .
## NW           3.497910e-02
## U1           .
## U2           3.217698e-02
## Wealth       .
## Ineq         2.731812e-01
## Prob        -1.758349e-01
## Time         .
```

it has printed a ton of variables. using what we learnt from inference in lasso, we will drop the really low vars here, namely So, Po2, Pop, NW, Wealth - so we're left with:

```
elasticModelWithTightenedCoefficientsFINAL <- lm (formula =  Crime ~ M + Ed + Po1 +  M.F + U1 + U2 + In

summary(elasticModelWithTightenedCoefficientsFINAL)
```

```
##
## Call:
## lm.default(formula = Crime ~ M + Ed + Po1 + M.F + U1 + U2 + Ineq +
##     Prob, data = crimeDataTable)
##
## 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) -6426.10    1194.61  -5.379 4.04e-06 ***
## M              93.32      33.50   2.786  0.00828 **
## Ed            180.12      52.75   3.414  0.00153 **
## Po1           102.65      15.52   6.613 8.26e-08 ***
## M.F            22.34      13.60   1.642  0.10874
## U1          -6086.63    3339.27  -1.823  0.07622 .
## U2            187.35      72.48   2.585  0.01371 *
## Ineq           61.33      13.96   4.394 8.63e-05 ***
## Prob        -3796.03    1490.65  -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
```

** Incidently this is the same exact variables that lasso also recommended! shows that both models are functioning and fairly accurate.

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.

At work, I'm responsible for our continuous delivery platform where all our software is developed. thousands of developers heavily use the central git tools (we employ bitbucket) for their code repository and day to day work, and we get hundreds of code commits daily.

The interface is also used for reviewing and approving code. However, I've noticed that the code quality is very variable. We have code quality tools like sonarqube linked from the bitbucket portal but i'm not sure people use that link to review their code. I suspect it may be a matter of the link not being prominent.

So I'd like to set up a design of experiment, where for some userIDs I would change the front end to show the link as more prominent with red background than others, and see if we get better clickthrough. This type of A/B testing would give us a better idea of clickthrough success.

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.

This is a simple question. We just need to run FrF2 with the number of runs which is 16 (houses) and 10 features (large yard etc)

We pass these two parameters to the FrF2 function and get back a matrix of 10*16, where the columns are the value TRUE/FALSE for the feature and each row just identifies the 16 runs (houses)

```
set.seed(1)
#numberOfRuns <- 50*16 #each buyer sees 16 houses so total number of runs is 50x16
numberOfRuns <- 16
numberOfFactors <- 10

frf2Output <- FrF2(numberOfRuns, numberOfFactors)

head(frf2Output)
```

```
##    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
```

To take an example of House # 5. It does NOT have feature A,C, D etc, while it does have feature B, F etc (+1) Each factor should have 8 factors in them. This was determined after randomly adding up each positive (+1) in each row..e.g. `sum(frf2Output$D == 1)`

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 In

binomial distribution, we consider probability distributions for which there are just two possible outcomes with fixed probabilities summing to one.

The chance that a republican vs a democrat congressman is selected for each area.

b. Geometric

The geometric distribution represents the number of failures before you get a success in a series of Bernoulli trials.

If you're a couple and trying to get pregnant, then the number of times you try and fail until you finally conceive could be considered as a geometric distribution.

c. Poisson

The Poisson distribution can be used to calculate the probabilities of various numbers of "successes" based on the mean number of successes. In order to apply the Poisson distribution, the various events must be independent.

The number of incoming shipments of mail coming to the post office every day is an independent variable and could be a good example here.

d. Exponential

Questions concerning the time we need to wait before a given event occurs are related to the exponential distribution. If this waiting time is unknown, it is often appropriate to think of it as a random variable having an exponential distribution.

An example could be : how much time will elapse before an earthquake occurs in a given island

e. Weibull - we will now attempt to loop over alpha values.. alpha <- 0.1 ;

for i in 1:10 {

}

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.

At work, I'm responsible for our continuous delivery platform where all our software is developed. thousands of developers heavily use the central git tools (we employ bitbucket) for their code repository and day to day work, and we get hundreds of code commits daily.

The interface is also used for reviewing and approving code. However, I've noticed that the code quality is very variable. We have code quality tools like sonarqube linked from the bitbucket portal but i'm not sure people use that link to review their code. I suspect it may be a matter of the link not being prominent.

So I'd like to set up a design of experiment, where for some userIDs I would change the front end to show the link as more prominent with red background than others, and see if we get better clickthrough. This type of A/B testing would give us a better idea of clickthrough success.

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.

This is a simple question. We just need to run FrF2 with the number of runs which is 16 (houses) and 10 features (large yard etc)

We pass these two parameters to the FrF2 function and get back a matrix of 10*16, where the columns are the value TRUE/FALSE for the feature and each row just identifies the 16 runs (houses)

```
set.seed(1)
#numberOfRuns <- 50*16 #each buyer sees 16 houses so total number of runs is 50x16
numberOfRuns <- 16
numberOfFactors <- 10

frf2Output <- FrF2(numberOfRuns, numberOfFactors)

head(frf2Output)
```

```
##    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
```

To take an example of House # 5. It does NOT have feature A,C, D etc, while it does have feature B, F etc (+1) Each factor should have 8 factors in them. This was determined after randomly adding up each positive (+1) in each row..e.g. `sum(frf2Output$D == 1)`

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 In binomial distribution, we consider probability distributions for which there are just two possible outcomes with fixed probabilities summing to one.

The chance that a republican vs a democrat congressman is selected for each area.

b. Geometric

The geometric distribution represents the number of failures before you get a success in a series of Bernoulli trials.

If you're a couple and trying to get pregnant, then the number of times you try and fail until you finally conceive could be considered as a geometric distribution.

c. Poisson

The Poisson distribution can be used to calculate the probabilities of various numbers of "successes" based on the mean number of successes. In order to apply the Poisson distribution, the various events must be independent.

The number of incoming shipments of mail coming to the post office every day is an independent variable and could be a good example here.

d. Exponential

Questions concerning the time we need to wait before a given event occurs are related to the exponential distribution. If this waiting time is unknown, it is often appropriate to think of it as a random variable having an exponential distribution.

An example could be : how much time will elapse before an earthquake occurs in a given island

e. Weibull

The Weibull distribution is a continuous probability distribution From Google: Today, it's commonly used to assess product reliability, analyze life data and model failure times

With this in mind, we can state that Weibull can be used to estimate the life of a perishable product in a grocery store, to see how much time should it be kept out before it needs to be thrown away