IAM Homework 5

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.
Notes on R:
• For the elastic net model, what we called λ in the videos, `glmnet` calls "alpha"; you can get a range of results by varying alpha from 1 (lasso) to 0 (ridge regression) [and, of course, other values of alpha in between].
• In a function call like `glmnet(x,y,family="mgaussian",alpha=1)` the predictors `x` need to be in R's matrix format, rather than data frame format. You can convert a data frame to a matrix using `as.matrix` – for example, `x <- as.matrix(data[,1:n-1])`
- **Rather than specifying a value of T, `glmnet` returns models for a variety of values of T.**

Part 1 – Stepwise

Step 1
Read the crime data and scale the data

Input
```
#Read the crime data and scale
uscrime <- read.table("uscrime.txt", stringsAsFactors = FALSE, header = TRUE)
head(uscrime)
uscrime2 <- as.data.frame(scale(uscrime))
head(uscrime2)
```

Step 2
Next I run a combination of forwards and backwards stepwise regression to minimize the AIC value of the model by reducing the number of coefficients. Originally, I ran forward and backward regression separately but after learning how to run them simultaneously I decided that was all I needed to show. The minimum AIC I received from running backward stepwise regression and running the simultaneous forward and backward stepwise regressions were the same.

Input
```
#Both forward and backwards stepwise regression
model_1 <- lm(Crime~., data = uscrime2)
step(model_1, scope = list(lower = formula(lm(Crime~1, data=uscrime2)),
                           upper = formula(lm(Crime~., data=uscrime2))),
                           direction = "both")
```

Output

```
Step:  AIC=-56.1
Crime ~ M + Ed + Po1 + M.F + U1 + U2 + Ineq + Prob

         Df Sum of Sq     RSS     AIC
<none>                 9.7140 -56.099
+ Wealth  1    0.1771  9.5369 -54.964
- M.F     1    0.6896 10.4036 -54.876
+ Pop     1    0.1116  9.6024 -54.643
+ Po2     1    0.0946  9.6194 -54.559
+ So      1    0.0624  9.6516 -54.402
+ LF      1    0.0292  9.6847 -54.241
+ NW      1    0.0254  9.6886 -54.222
+ Time    1    0.0153  9.6986 -54.174
- U1      1    0.8493 10.5633 -54.160
- Prob    1    1.6578 11.3717 -50.694
- U2      1    1.7077 11.4216 -50.488
- M       1    1.9841 11.6981 -49.364
- Ed      1    2.9802 12.6941 -45.523
- Ineq    1    4.9353 14.6492 -38.791
- Po1     1   11.1778 20.8918 -22.107

Call:
lm.default(formula = Crime ~ M + Ed + Po1 + M.F + U1 + U2 + Ineq +
    Prob, data = uscrime2)

Coefficients:
(Intercept)            M           Ed          Po1          M.F           U1
U2
 -3.591e-16    3.032e-01    5.210e-01    7.888e-01    1.702e-01   -2.837e-01
4.091e-01
       Ineq         Prob
  6.327e-01   -2.232e-01
```

Step 3

I now have reduced the amount of factors to 8. I will create the regression model with the 8 factors and review the model. The R squared adjusted value of this new model is 0.744 (74% of the variance explained).

Input

```
#Review the new model with certain factors removed
model_2 <- lm(formula = Crime~ M + Ed + Po1 + M.F + U1 + U2 + Ineq + Prob, data = uscrime2)
summary(model_2)
```

Output

```
Call:
lm.default(formula = Crime ~ M + Ed + Po1 + M.F + U1 + U2 + Ineq +
    Prob, data = uscrime2)
```

```
Residuals:
     Min       1Q   Median       3Q      Max
-1.14981 -0.28718  0.00784  0.31581  1.24960

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -3.591e-16  7.375e-02   0.000  1.00000
M            3.032e-01  1.088e-01   2.786  0.00828 **
Ed           5.210e-01  1.526e-01   3.414  0.00153 **
Po1          7.888e-01  1.193e-01   6.613 8.26e-08 ***
M.F          1.702e-01  1.036e-01   1.642  0.10874
U1          -2.837e-01  1.557e-01  -1.823  0.07622 .
U2           4.091e-01  1.583e-01   2.585  0.01371 *
Ineq         6.327e-01  1.440e-01   4.394 8.63e-05 ***
Prob        -2.232e-01  8.763e-02  -2.547  0.01505 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.5056 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
```

Part 2 – LASSO

Step 1

I first call the glmnet pack and set the seed for reproducible results.

Input

```
#Call package and set seed for reproducible results
library(glmnet)
set.seed(42)
```

Step 2

I am now ready to perform LASSO by using the cv.glmnet function. I first must convert the data into a matrix so that it is compatible with the function. I am choosing an alpha value of 1 and an nfolds value of 5 (cross validation across 5 folds). Then I review the results of the model and plot the values. Lambda min corresponds to the lowest cross validated mean squared error. In the first plot you can see the optimum values of log(lambda) before the mean squared error starts increasing. You can see this again in the second plot I created where the optimum value of lambda (or the LASSO threshold) looks to be in be the same as the lambda min at ~0.0103.

Input

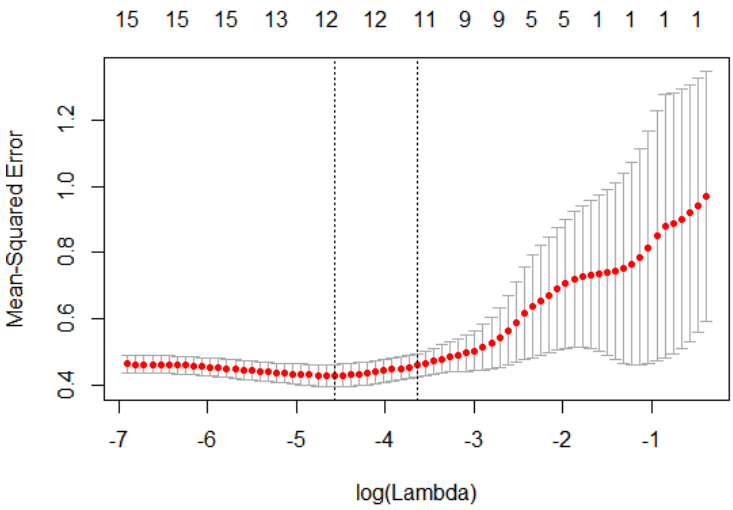
```
#Run LASSO function by first converting data into a matrix. 5 Folds for cross validation. MSE automates the choice of the lambda value or "T"
model_lasso <-cv.glmnet(x=as.matrix(uscrime2[,-16]),
                        y=as.matrix(uscrime2[,16]),
                        alpha=1,
```

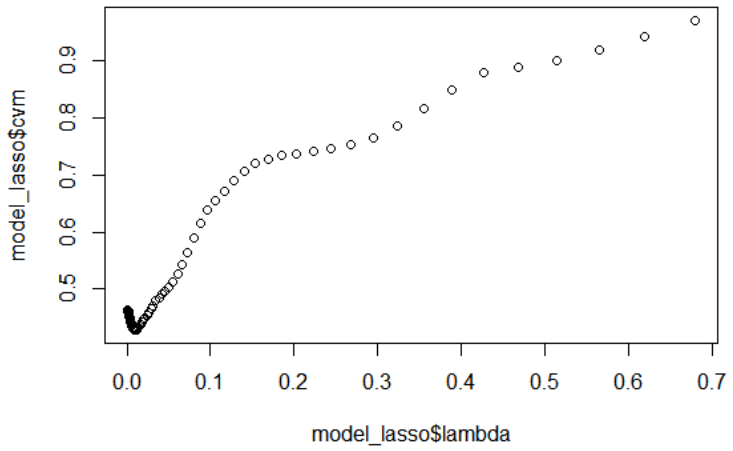
```
            nfolds=5,
            type.measure = "mse",
            family = "gaussian")
model_lasso
plot(model_lasso)
```

Output

```
Call:  glmnet(x = as.matrix(uscrime2[, -16]), y = as.matrix(uscrime2[,      1
6]), alpha = 1, family = "gaussian")

      Df    %Dev    Lambda
 [1,]  0 0.00000 0.6803000
 [2,]  1 0.08027 0.6198000
 [3,]  1 0.14690 0.5648000
 [4,]  1 0.20220 0.5146000
 [5,]  1 0.24820 0.4689000
 [6,]  1 0.28630 0.4272000
 [7,]  1 0.31800 0.3893000
 [8,]  1 0.34430 0.3547000
 [9,]  1 0.36610 0.3232000
[10,]  1 0.38420 0.2945000
[11,]  1 0.39920 0.2683000
[12,]  1 0.41170 0.2445000
[13,]  1 0.42210 0.2228000
[14,]  1 0.43070 0.2030000
[15,]  3 0.44240 0.1849000
[16,]  4 0.45870 0.1685000
[17,]  4 0.48700 0.1535000
[18,]  5 0.52490 0.1399000
[19,]  5 0.55650 0.1275000
[20,]  5 0.58260 0.1161000
[21,]  5 0.60430 0.1058000
[22,]  5 0.62240 0.0964200
[23,]  5 0.63730 0.0878600
[24,]  6 0.64980 0.0800500
[25,]  7 0.66700 0.0729400
[26,]  9 0.68240 0.0664600
[27,]  9 0.69830 0.0605600
[28,]  9 0.71130 0.0551800
[29,]  9 0.72220 0.0502800
[30,]  9 0.73120 0.0458100
[31,] 10 0.73870 0.0417400
[32,] 10 0.74510 0.0380300
[33,] 11 0.75150 0.0346500
[34,] 11 0.75830 0.0315700
[35,] 11 0.76370 0.0287700
[36,] 11 0.76830 0.0262100
[37,] 11 0.77260 0.0238800
[38,] 11 0.77620 0.0217600
[39,] 12 0.77930 0.0198300
[40,] 12 0.78240 0.0180700
[41,] 12 0.78490 0.0164600
[42,] 12 0.78690 0.0150000
[43,] 12 0.78870 0.0136700
[44,] 12 0.79010 0.0124500
```

```
[45,] 12 0.79130 0.0113500
[46,] 12 0.79230 0.0103400
[47,] 12 0.79310 0.0094210
[48,] 12 0.79380 0.0085840
[49,] 12 0.79440 0.0078210
[50,] 12 0.79480 0.0071260
[51,] 12 0.79520 0.0064930
[52,] 12 0.79550 0.0059160
[53,] 13 0.79580 0.0053910
[54,] 13 0.79610 0.0049120
[55,] 13 0.79630 0.0044760
[56,] 13 0.79650 0.0040780
[57,] 14 0.79660 0.0037160
[58,] 14 0.79670 0.0033860
[59,] 15 0.79690 0.0030850
[60,] 15 0.79790 0.0028110
[61,] 15 0.79870 0.0025610
[62,] 15 0.79940 0.0023340
[63,] 15 0.80000 0.0021260
[64,] 15 0.80050 0.0019370
[65,] 15 0.80090 0.0017650
[66,] 15 0.80130 0.0016080
[67,] 15 0.80160 0.0014660
[68,] 15 0.80180 0.0013350
[69,] 15 0.80200 0.0012170
[70,] 15 0.80220 0.0011090
[71,] 15 0.80230 0.0010100
[72,] 15 0.80240 0.0009204
[73,] 15 0.80250 0.0008386
[74,] 15 0.80260 0.0007641
[75,] 15 0.80270 0.0006963
[76,] 15 0.80270 0.0006344
[77,] 15 0.80280 0.0005780
[78,] 15 0.80280 0.0005267
[79,] 15 0.80290 0.0004799
[80,] 15 0.80290 0.0004373
[81,] 14 0.80290 0.0003984
[82,] 14 0.80290 0.0003630
[83,] 14 0.80300 0.0003308
[84,] 14 0.80300 0.0003014
[85,] 14 0.80300 0.0002746
[86,] 14 0.80300 0.0002502
[87,] 14 0.80300 0.0002280
[88,] 15 0.80300 0.0002077

$lambda.min
[1] 0.0103392

$lambda.1se
[1] 0.02621364

attr(,"class")
[1] "cv.glmnet"
```





Step 3

In this step I will determine which factors are within the threshold for consideration. I will only include these selected factors in the model.

Input

```
#Determine which coefficients meet the threshold
coef(model_lasso, s = model_lasso$lambda.min)
coef(model_lasso)
```

Output

```
(Intercept) -3.046715e-16
M            2.247676e-01
So           5.738635e-02
```

```
Ed            3.407168e-01
Po1           7.955957e-01
Po2           .
LF            3.021196e-04
M.F           1.397577e-01
Pop           .
NW            1.340837e-02
U1           -7.733692e-02
U2            1.665824e-01
Wealth        .
Ineq          4.787490e-01
Prob         -2.148360e-01
Time          .
```

Step 4

Now that I have determined the factors I will use and their coefficients, I create a new model that outputs an adjusted R-Squared value of 0.7248. I could potentially take this analysis a step further by reviewing each factors p value and removing them if they are not statistically significant. I tried removing So, LF, M.F, NW, U1, and U2 – this gave me an adjusted R^2 of 0.7307 which is not much of an improvement. I decided not to show this and stick with my original model.

Input

```
#Review the new model with certain factors removed
model_3 <- lm(formula = Crime~ M + So + Ed + Po1 + LF + M.F + NW + U1 + U2 + Ineq + Prob, data = uscrime2)
summary(model_3)
```

Output

```
Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -3.602e-16  7.652e-02   0.000  1.00000
M            2.756e-01  1.274e-01   2.163  0.03747 *
So           4.529e-02  1.729e-01   0.262  0.79489
Ed           5.408e-01  1.681e-01   3.218  0.00278 **
Po1          7.643e-01  1.409e-01   5.424 4.44e-06 ***
LF          -2.765e-02  1.399e-01  -0.198  0.84447
M.F          1.938e-01  1.322e-01   1.466  0.15159
NW           3.363e-02  1.537e-01   0.219  0.82814
U1          -2.820e-01  1.854e-01  -1.521  0.13725
U2           3.916e-01  1.706e-01   2.295  0.02783 *
Ineq         6.024e-01  1.750e-01   3.443  0.00151 **
Prob        -2.482e-01  1.023e-01  -2.425  0.02059 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.5246 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
```

Part 3 – Elastic Net

Step 1

To perform Elastic Net I will use the cv.glmnet function and cycle through 10 alpha values between 0 and 1 to achieve an optimized result.

Input

```
for (i in 1:10){
model.elasticnet <- cv.glmnet(x = as.matrix(uscrime2[,-16]),
                      y = as.matrix(uscrime2[,16]),
                      alpha = i/10,
                      nfolds = 5,
                      type.measure = "mse",
                      family = "gaussian")}
model.elasticnet
```

Output

```
Call:  glmnet(x = as.matrix(uscrime2[, -16]), y = as.matrix(uscrime2[,      1
6]), alpha = i/10, family = "gaussian")

      Df    %Dev    Lambda
 [1,]  0 0.00000 0.6803000
 [2,]  1 0.08027 0.6198000
 [3,]  1 0.14690 0.5648000
 [4,]  1 0.20220 0.5146000
 [5,]  1 0.24820 0.4689000
 [6,]  1 0.28630 0.4272000
 [7,]  1 0.31800 0.3893000
 [8,]  1 0.34430 0.3547000
 [9,]  1 0.36610 0.3232000
[10,]  1 0.38420 0.2945000
[11,]  1 0.39920 0.2683000
[12,]  1 0.41170 0.2445000
[13,]  1 0.42210 0.2228000
[14,]  1 0.43070 0.2030000
[15,]  3 0.44240 0.1849000
[16,]  4 0.45870 0.1685000
[17,]  4 0.48700 0.1535000
[18,]  5 0.52490 0.1399000
[19,]  5 0.55650 0.1275000
[20,]  5 0.58260 0.1161000
[21,]  5 0.60430 0.1058000
[22,]  5 0.62240 0.0964200
[23,]  5 0.63730 0.0878600
[24,]  6 0.64980 0.0800500
[25,]  7 0.66700 0.0729400
[26,]  9 0.68240 0.0664600
[27,]  9 0.69830 0.0605600
[28,]  9 0.71130 0.0551800
[29,]  9 0.72220 0.0502800
```

```
[30,]  9 0.73120 0.0458100
[31,] 10 0.73870 0.0417400
[32,] 10 0.74510 0.0380300
[33,] 11 0.75150 0.0346500
[34,] 11 0.75830 0.0315700
[35,] 11 0.76370 0.0287700
[36,] 11 0.76830 0.0262100
[37,] 11 0.77260 0.0238800
[38,] 11 0.77620 0.0217600
[39,] 12 0.77930 0.0198300
[40,] 12 0.78240 0.0180700
[41,] 12 0.78490 0.0164600
[42,] 12 0.78690 0.0150000
[43,] 12 0.78870 0.0136700
[44,] 12 0.79010 0.0124500
[45,] 12 0.79130 0.0113500
[46,] 12 0.79230 0.0103400
[47,] 12 0.79310 0.0094210
[48,] 12 0.79380 0.0085840
[49,] 12 0.79440 0.0078210
[50,] 12 0.79480 0.0071260
[51,] 12 0.79520 0.0064930
[52,] 12 0.79550 0.0059160
[53,] 13 0.79580 0.0053910
[54,] 13 0.79610 0.0049120
[55,] 13 0.79630 0.0044760
[56,] 13 0.79650 0.0040780
[57,] 14 0.79660 0.0037160
[58,] 14 0.79670 0.0033860
[59,] 15 0.79690 0.0030850
[60,] 15 0.79790 0.0028110
[61,] 15 0.79870 0.0025610
[62,] 15 0.79940 0.0023340
[63,] 15 0.80000 0.0021260
[64,] 15 0.80050 0.0019370
[65,] 15 0.80090 0.0017650
[66,] 15 0.80130 0.0016080
[67,] 15 0.80160 0.0014660
[68,] 15 0.80180 0.0013350
[69,] 15 0.80200 0.0012170
[70,] 15 0.80220 0.0011090
[71,] 15 0.80230 0.0010100
[72,] 15 0.80240 0.0009204
[73,] 15 0.80250 0.0008386
[74,] 15 0.80260 0.0007641
[75,] 15 0.80270 0.0006963
[76,] 15 0.80270 0.0006344
[77,] 15 0.80280 0.0005780
[78,] 15 0.80280 0.0005267
[79,] 15 0.80290 0.0004799
[80,] 15 0.80290 0.0004373
[81,] 14 0.80290 0.0003984
[82,] 14 0.80290 0.0003630
[83,] 14 0.80300 0.0003308
[84,] 14 0.80300 0.0003014
[85,] 14 0.80300 0.0002746
[86,] 14 0.80300 0.0002502
```

```
[87,] 14 0.80300 0.0002280
[88,] 15 0.80300 0.0002077

$lambda.min
[1] 0.04173952

$lambda.1se
[1] 0.08005266

attr(,"class")
[1] "cv.glmnet"
```

Step 2

Similar to how we did in the LASSO approach we determine the coefficients that meet the designated threshold output by the glmnet function.

Input

```
#Determine which coefficients meet the threshold
coef(model.elasticnet, s = model.elasticnet$lambda.min)
coef(model.elasticnet)
```

Output

```
> coef(model.elasticnet)
16 x 1 sparse Matrix of class "dgCMatrix"
                            1
(Intercept) -3.694731e-16
M            1.157874e-01
So           .
Ed           1.884049e-04
Po1          7.311501e-01
Po2          .
LF           .
M.F          1.403030e-01
Pop          .
NW           .
U1           .
U2           .
Wealth       .
Ineq         2.130764e-01
Prob        -1.342423e-01
Time         .
```

Step 4

Lastly, we use the factors we are left with to create our final model. The R^2 adjusted value of the model we created using the Elastic Net approach is 0.7127

Input

```
#Review the new model with certain factors removed
```

```
model_4 <- lm(formula = Crime~ M + Ed + Po1 + M.F + Ineq + Prob, data = uscrime2)
summary(model_4)
```

Output

```
Call:
lm.default(formula = Crime ~ M + Ed + Po1 + M.F + Ineq + Prob,
    data = uscrime2)

Residuals:
     Min       1Q   Median       3Q      Max
-1.25316 -0.27561 -0.00631  0.35134  1.30748

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -3.016e-16  7.818e-02   0.000   1.0000
M            2.372e-01  1.059e-01   2.240   0.0307 *
Ed           3.620e-01  1.436e-01   2.521   0.0158 *
Po1          9.418e-01  1.071e-01   8.797 6.80e-11 ***
M.F          1.311e-01  9.349e-02   1.402   0.1685
Ineq         6.712e-01  1.504e-01   4.464 6.41e-05 ***
Prob        -2.319e-01  9.283e-02  -2.498   0.0167 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.536 on 40 degrees of freedom
Multiple R-squared:  0.7502,   Adjusted R-squared:  0.7127
F-statistic: 20.02 on 6 and 40 DF,  p-value: 1.2e-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.

My father works in the HVAC industry in South Florida and one example of where DOE could be effective is in his internet marketing approach. We could create two different advertisement designs then run an A/B testing approach to select the best option. Alternatively we could use utilize a factorial design method where we create several advertisement designs and then run an analysis to understand which factors are the most attractive to prospective customers. In each case it would be beneficial to use a “multi-armed bandit” approach to make sure we are finding the balance between gathering information and exploiting our customer base to potentially ineffective advertising.

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.

Step 1

Call the FrF2 library and set the seed for reproducible results.

Input

```
#Call package and set seed for reproducible results

library(FrF2)

set.seed(42
```

Step 2

Run the FrF2 function for 16 houses and include 10 features. As you can see in the table there are 16 rows to represent each house and 10 columns to represent each factor (A-K). The factors for each house that should be used are signified by the number 1. If a factor should be tossed then it will be marked -1.

For example house 1 should use features B, C, D, G, and J (all marked 1).

Input

```
#Run FrF2 function

FrF2(nruns = 16, nfactors = 10)
```

Output

```
    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
class=design, type= FrF2
```

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

A. Binomial – A Binomial distribution has only two outcomes. We could use a binomial distribution to show the probability that an employee will either pass or fail a company's internal training program
B. Geometric – A Beometric distribution tells you the number of attempts necessary to get a response. We could use a geometric distribution to show the probability curve for how many times a person must crank a lawn mower until he or she is able to start it up
C. Poisson – A Poisson distribution models the probability that a certain number of events will occur during a specific time period given the mean. We could use a Poisson distribution to show the probability that x customers will go through the Chick Fil A drive thru during a certain time interval (given the average time interval in between customers)
D. Exponential – An Exponential distribution tells us the probability that a certain interval of time will pass in between events. We could flip our example from Part C and use an exponential model to tell us the probability of witnessing a certain time interval in between each customer. The difference is that with the Poisson distribution we are analyzing each customer arrival and with the exponential distribution we are analyzing the inter-arrival time
E. Weibull – A Weibull distribution models the time between failures or how long it takes something to fail. This can be applied to a continuously running air conditioner. The Weibull distribution will give us the probability that the AC will fail after X number of hours. Assuming there are no defects we should set K to be greater than 1 because we are assuming the failure rate increases with time.