Lab_06R

36-290 – Statistical Research Methodology

Week 6 Thursday – Fall 2021

Preliminaries

Goal

The goal of this lab is to code and interpret ridge regression and lasso analyses using the first dataset that you looked at during Tuesday's lab.

One thing to keep in mind is that lasso and ridge regression can be applied in a logistic regression context! We don't do this here, and it is important to note, I didn't test logistic ridge regression and lasso for speed. But it is a possibility for those of you with, e.g., categorial response data in your semester project. Just see the documentation for `glmnet()` *and note that while the default family is* `gaussian` *, you can specify* `binomial` *.*

Data

```
rm(list = ls())
```

We'll begin by importing the first dataset from Tuesday:

```
file.path = "https://raw.githubusercontent.com/pefreeman/36-290/master/EXAMPLE_DATASETS/PHOTO_MORPH/photo_morph.Rda
ta"
load(url(file.path))
rm(file.path)
```

See Tuesday's lab and the README file on GitHub for a description of these data.

Note that $n \gg p$ here, so at the end of the lab we will repeat the analyses after selecting 100 rows randomly from the dataset. When/if you apply lasso regression and/or the lasso to your semester-project dataset, don't do this! Always use all your data. Here, we'll cut data just to build some intuition about what happens when the actual sample size is small.

Questions

To answer the questions below, it will help you to refer to Sections 6.2 and 6.6 of ISLR; it might also help you to refer to your previous lab work (and, as always, to Google).

Question 1

Split your data into a training set of size 2,000 and a test set of size 1,419. (Remember: these sets are *disjoint*!) Call your training set variables `pred.train` and `resp.train` , and your test set variables `pred.test` and `resp.test` . (Remember: set the seed!)

```
set.seed(100)

p=nrow(predictors)

s = sample(p,2000)
pred.train = predictors[s,]
resp.train = response[s]
pred.test = predictors[-s,]
resp.test = response[-s]

df_train = cbind(pred.train, resp.train)
df_test = cbind(pred.test, resp.test)
```

Question 2

First, install the `glmnet` package if you need to. Then use `model.matrix()` as shown on page 251 of ISLR to transform the `pred.train` and `pred.test` data frames to matrices. (Read the explanation around the `model.matrix()` code block to understand why. tl;dr – the coders of `glmnet` did not follow typical `R` conventions.) Run ridge regression on the training data, i.e., run the `glmnet()` function with argument `alpha` = 0. Assume the default range for `lambda` . (Note that `glmnet()` will standardize the data for you by default…you don't have to do it separately.) Show the dimensionality of the output `coef` matrix. How many λ values are used by default? (The values are stored in the `lambda` variable within the model-fit output variable.)

```
y = response

library(glmnet)
```

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

```
## Loaded glmnet 4.1-2
```

```
pred.train = model.matrix(resp.train~., df_train)[,-1]
pred.test = model.matrix(resp.test~., df_test)[,-1]


ridge_mod = glmnet(pred.train, resp.train, alpha=0)
dim(coef(ridge_mod))
```

```
## [1]  17 100
```

```
ridge_pred = predict(ridge_mod, s = 4, newx = pred.test)
mean((ridge_pred - resp.test)^2)
```

```
## [1] 0.5577555
```

```
out = glmnet(predictors, response, alpha=0)
out
```

```
##
## Call:  glmnet(x = predictors, y = response, alpha = 0)
##
##     Df  %Dev Lambda
## 1   16  0.00 602.20
## 2   16  0.46 548.70
## 3   16  0.51 500.00
## 4   16  0.56 455.50
## 5   16  0.61 415.10
## 6   16  0.67 378.20
## 7   16  0.74 344.60
## 8   16  0.81 314.00
## 9   16  0.88 286.10
## 10  16  0.97 260.70
## 11  16  1.06 237.50
## 12  16  1.16 216.40
## 13  16  1.27 197.20
## 14  16  1.39 179.70
## 15  16  1.52 163.70
## 16  16  1.67 149.20
## 17  16  1.82 135.90
## 18  16  1.99 123.80
## 19  16  2.18 112.80
## 20  16  2.38 102.80
## 21  16  2.60  93.68
## 22  16  2.84  85.36
## 23  16  3.10  77.78
## 24  16  3.39  70.87
## 25  16  3.69  64.57
## 26  16  4.02  58.84
## 27  16  4.38  53.61
## 28  16  4.77  48.85
## 29  16  5.19  44.51
## 30  16  5.63  40.55
## 31  16  6.12  36.95
## 32  16  6.64  33.67
## 33  16  7.19  30.68
## 34  16  7.79  27.95
## 35  16  8.42  25.47
## 36  16  9.09  23.21
## 37  16  9.81  21.14
## 38  16 10.57  19.27
## 39  16 11.37  17.55
## 40  16 12.22  16.00
## 41  16 13.10  14.57
## 42  16 14.03  13.28
## 43  16 15.01  12.10
```

```
## 44  16 16.02  11.02
## 45  16 17.06  10.05
## 46  16 18.15   9.15
## 47  16 19.26   8.34
## 48  16 20.41   7.60
## 49  16 21.58   6.92
## 50  16 22.77   6.31
## 51  16 23.98   5.75
## 52  16 25.21   5.24
## 53  16 26.45   4.77
## 54  16 27.69   4.35
## 55  16 28.93   3.96
## 56  16 30.17   3.61
## 57  16 31.41   3.29
## 58  16 32.63   3.00
## 59  16 33.85   2.73
## 60  16 35.05   2.49
## 61  16 36.23   2.27
## 62  16 37.39   2.07
## 63  16 38.53   1.88
## 64  16 39.64   1.72
## 65  16 40.74   1.56
## 66  16 41.80   1.42
## 67  16 42.85   1.30
## 68  16 43.86   1.18
## 69  16 44.85   1.08
## 70  16 45.80   0.98
## 71  16 46.73   0.89
## 72  16 47.63   0.81
## 73  16 48.51   0.74
## 74  16 49.35   0.68
## 75  16 50.16   0.62
## 76  16 50.94   0.56
## 77  16 51.69   0.51
## 78  16 52.41   0.47
## 79  16 53.09   0.42
## 80  16 53.75   0.39
## 81  16 54.37   0.35
## 82  16 54.96   0.32
## 83  16 55.52   0.29
## 84  16 56.04   0.27
## 85  16 56.54   0.24
## 86  16 57.00   0.22
## 87  16 57.44   0.20
## 88  16 57.84   0.18
## 89  16 58.22   0.17
## 90  16 58.57   0.15
## 91  16 58.89   0.14
## 92  16 59.19   0.13
## 93  16 59.46   0.12
## 94  16 59.71   0.11
## 95  16 59.94   0.10
## 96  16 60.14   0.09
## 97  16 60.33   0.08
## 98  16 60.50   0.07
## 99  16 60.66   0.07
## 100 16 60.80   0.06
```

```
100 values?
```

Question 3

Display the model coefficients for the largest and smallest values of λ. What differences do you see?

```
out$lambda[1]
```

```
## [1] 602.2078
```

```
coef(out)[,100]
```

```
## (Intercept)       mag.i      col.Vi      col.iJ      col.JH         V.G
##  1.70133808  0.38649318 -0.08423764 -0.61224765  0.40411456 -0.04020086
##       V.M20         V.C      V.size         J.G       J.M20         J.C
##  0.04305132  0.04100069 -0.10137122  0.61923729  0.11898447  0.08999207
##      J.size         H.G       H.M20         H.C      H.size
## -0.15496097  0.22583213  0.27061878  0.09481334  0.43213923
```

```
out$lambda[100]
```

```
## [1] 0.06022078
```

```
coef(out)[,1]
```

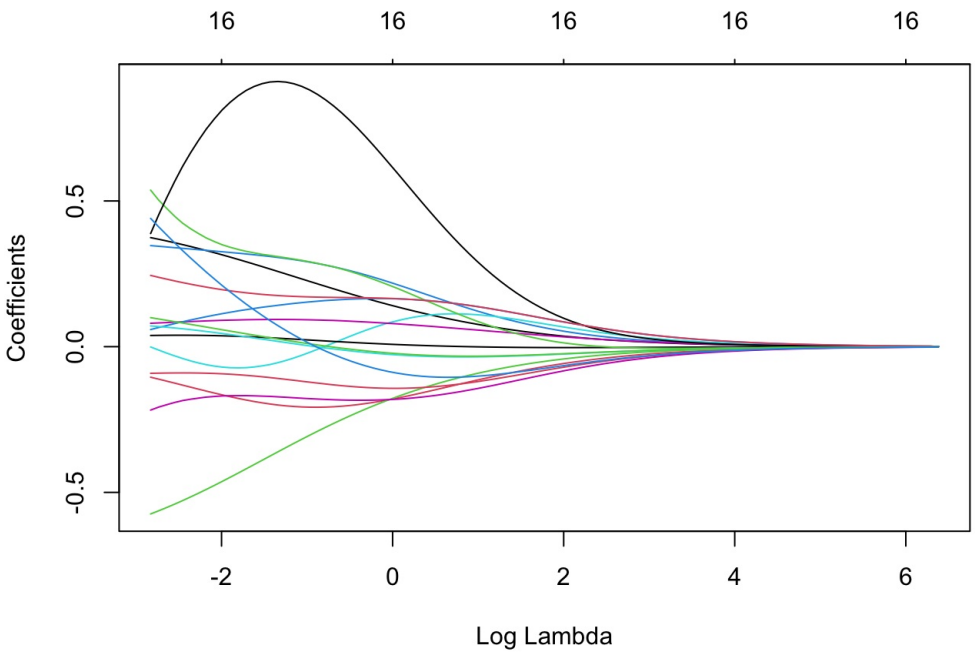
```
##   (Intercept)         mag.i        col.Vi        col.iJ        col.JH
##  1.303179e+00  4.085844e-37 -7.010053e-37 -5.071740e-37  5.354758e-37
##           V.G         V.M20           V.C        V.size           J.G
##  5.786861e-37  4.981674e-37 -1.070046e-37 -8.798106e-37 -6.814075e-37
##         J.M20           J.C        J.size           H.G         H.M20
##  1.255664e-36 -4.234901e-37 -1.039236e-36 -1.297201e-37  1.277798e-36
##           H.C        H.size
## -4.319659e-37 -8.666286e-37
```

```
Compared to the largest value of lambda, the smallest value of lambda has coefficient values that are larger.
```

Question 4

Run `plot()` using the output from your ridge regression fit. Use the argument `xvar="lambda"` , which gives you the most intuitive output. Explain concisely what the plot is showing.

```
plot(ridge_mod, xvar="lambda")
```

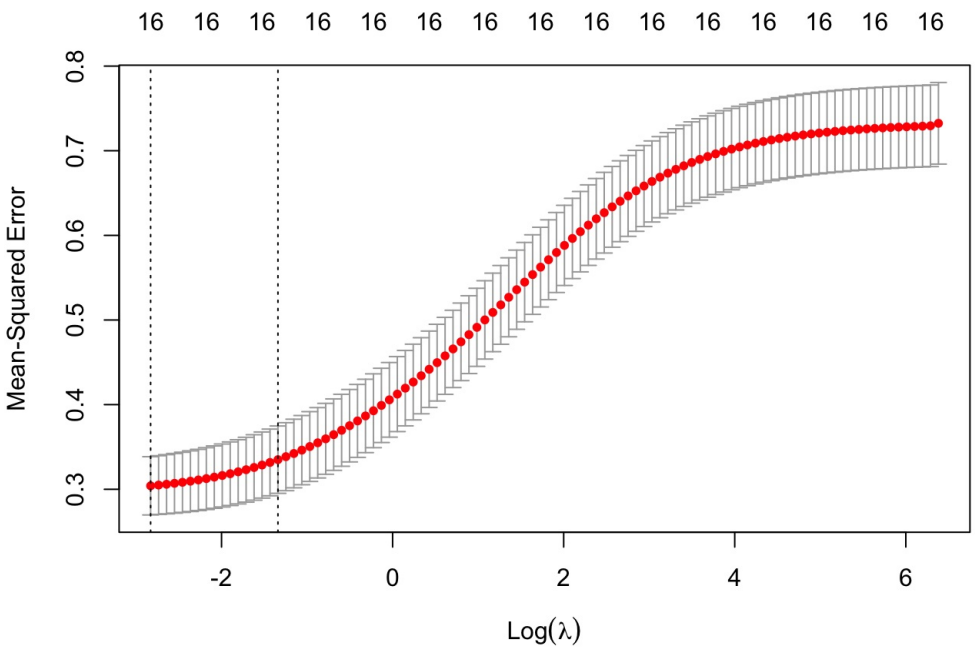



```
The plot is showing the log lambda values and the associated coefficient values. It shows the range of values of th
e coefficients, including if it is zero, associated with each log lambda.
```

Question 5

Follow the code on page 254 of ISLR and use cross-validation to select the best value of λ, then use the value of λ to compute the test-set MSE. Display the test-set MSE value; below, we'll see if we get a lower value using lasso. (Include the plot of the cross-validation MSE versus λ.) Is there any evidence that shrinking the coefficients is helpful? (To help answer this question, you could rerun the prediction and test-set MSE steps using `lm()` .)

```
set.seed(100)

cv.out = cv.glmnet(pred.train,resp.train, alpha=0)
plot(cv.out)
```




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

```
## [1] 0.05907809
```

```
ridge.pred=predict(ridge_mod, s=bestlam, newx=pred.test)

mean((ridge.pred - resp.test)^2)
```

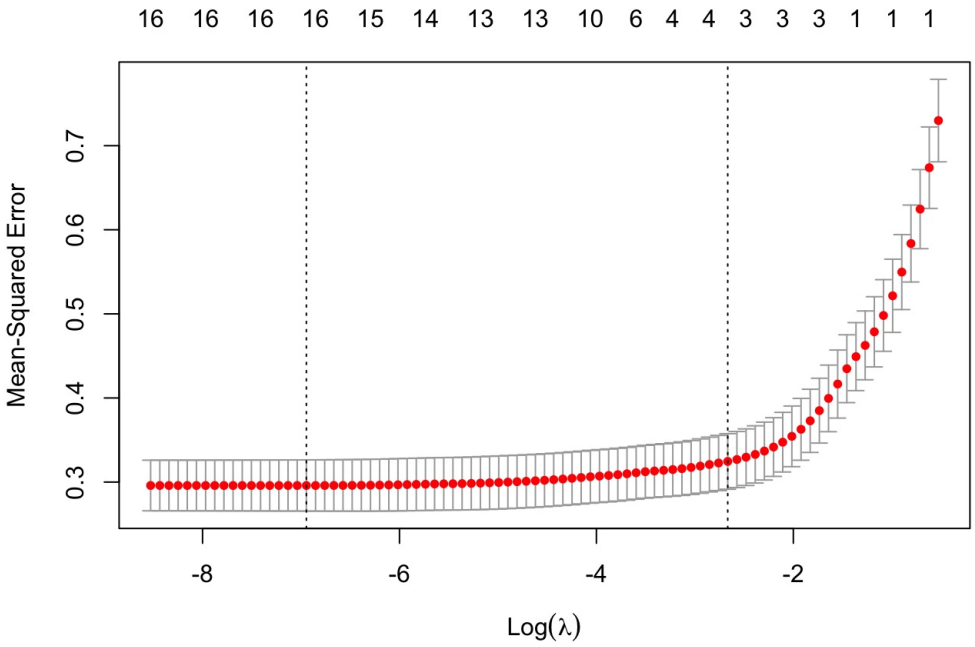
```
## [1] 0.2999011
```

```
MSE: 0.2998286

yes there is evidence that shrinking coefficients is useful because there is a point where the MSE is reduced with
a certain number of unshrinked coefficients
```

Question 6

Repeat the fitting done in Q2, Q4, and Q5 with the lasso (`glmnet` with `alpha` set to 1). Add to this a computation of `lasso.coef` like what is done on page 255 of ISLR, so that you can see which coefficients are non-zero. Set the same random number seed at you set in Q5 prior to performing cross-validation, so that the same data are placed into the same folds. Do you see any difference here compared to the ridge regression fit?

```
library(glmnet)



pred.train = model.matrix(resp.train~., df_train)[,-1]
pred.test = model.matrix(resp.test~., df_test)[,-1]

set.seed(100)
cv.out2= cv.glmnet(pred.train,resp.train, alpha=1)
plot(cv.out2)
```



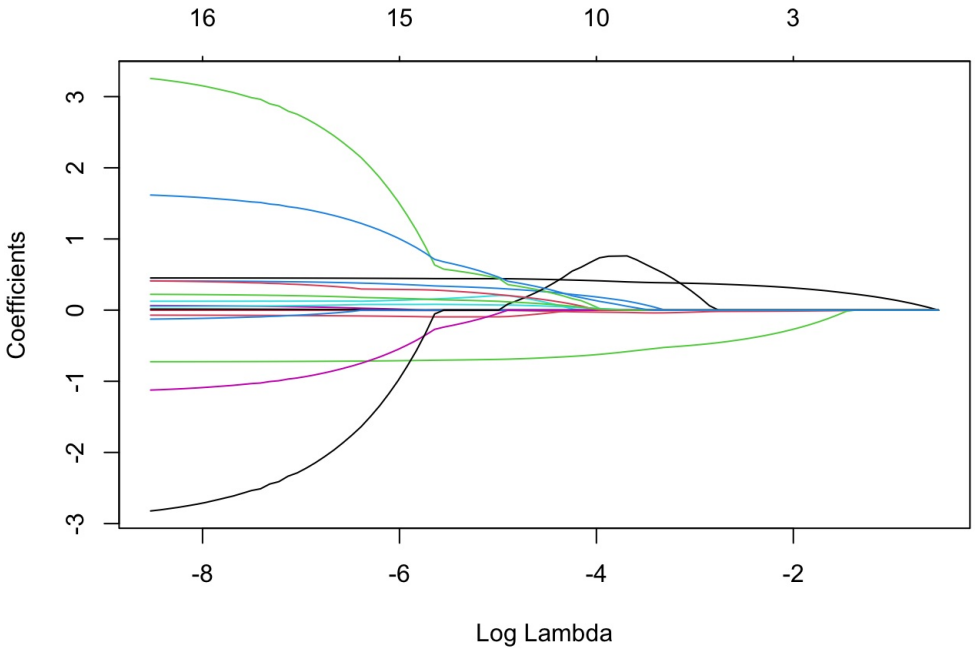

```
bestlam2 = cv.out2$lambda.min
bestlam2
```

```
## [1] 0.0009628257
```

```
lasso_mod = glmnet(pred.train, resp.train, alpha=1)

dim(coef(lasso_mod))
```

```
## [1] 17 87
```

```
plot(lasso_mod, xvar="lambda")
```




```
lasso_pred = predict(lasso_mod, s = bestlam2, newx = pred.test)
mean((lasso_pred - resp.test)^2)
```

```
## [1] 0.2851116
```

```
ridge.pred=predict(ridge_mod, s=bestlam, newx=pred.test)

mean((ridge.pred - resp.test)^2)
```

```
## [1] 0.2999011
```

```
out2=glmnet(predictors, response, alpha=1)
lasso.coef2 = predict(out2, type="coefficients", s=bestlam2)
lasso.coef2
```

```
## 17 x 1 sparse Matrix of class "dgCMatrix"
##                      s1
## (Intercept)  1.45873896
## mag.i        0.46530341
## col.Vi       0.01831607
## col.iJ      -0.75817637
## col.JH       0.45125318
## V.G          0.15285695
## V.M20        0.01849860
## V.C          0.02116820
## V.size      -0.10587529
## J.G          2.62991664
## J.M20        .
## J.C          0.08901357
## J.size      -0.79730900
## H.G         -2.34170845
## H.M20        0.35867354
## H.C          0.18814612
## H.size       1.36406643
```

```
Yes I do see a difference, the graph of the lambda and MSE is differently shaped and the number of coefficients tha
t are zero are different
```

Now, let's select a subset of the data randomly. Uncomment this block before running it.

```
set.seed(101)
nrow(predictors)
```

```
## [1] 3419
```

```
s = sample(nrow(predictors),2000)
pred.train = predictors[s,]
resp.train = response[s]
pred.test = predictors[-s,]
resp.test = response[-s]



set.seed(101)
s.train = sample(nrow(pred.train),35)
s.test  = sample(nrow(pred.test),15)
pred.train.small = pred.train[s.train,]
resp.train.small = resp.train[s.train]
pred.test.small = pred.test[s.test,]
resp.test.small = resp.test[s.test]

x.train.small = model.matrix(resp.train.small~.,pred.train.small)[,-1]
y.train.small = resp.train.small
x.test.small  = model.matrix(resp.test.small~.,pred.test.small)[,-1]
y.test.small  = resp.test.small
```

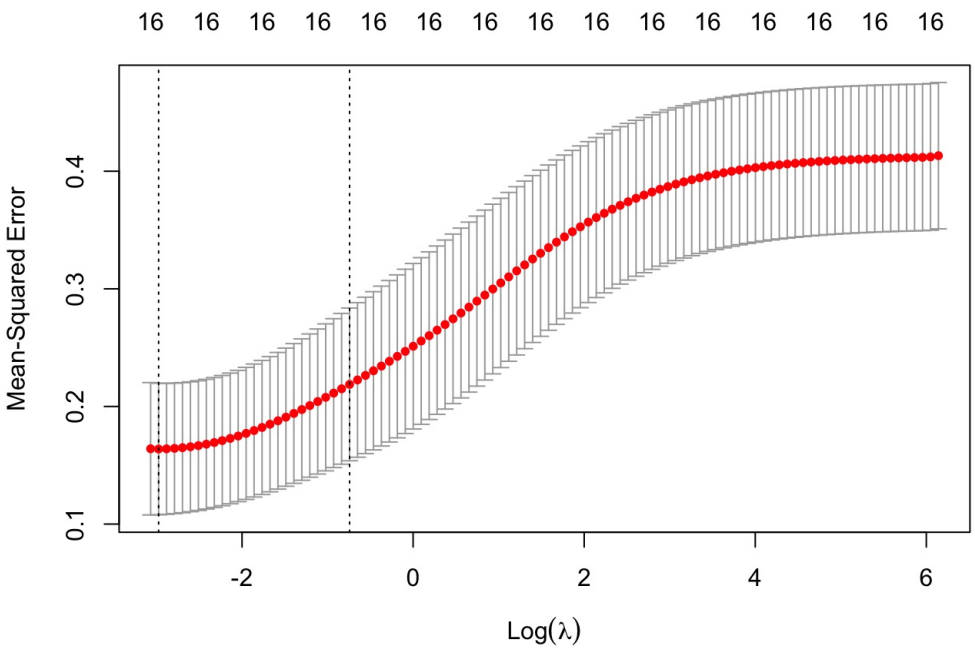
Question 7

Repeat the ridge regression analysis from above using the small datasets. (Use your code from Q6, with `alpha` set to 0.) Your last step should be to compute an MSE.

```
set.seed(101)
nrow(predictors)
```

```
## [1] 3419
```

```
s = sample(nrow(predictors),2000)
pred.train = predictors[s,]
resp.train = response[s]
pred.test = predictors[-s,]
resp.test = response[-s]


set.seed(101)
s.train = sample(nrow(pred.train),35)
s.test  = sample(nrow(pred.test),15)
pred.train.small = pred.train[s.train,]
resp.train.small = resp.train[s.train]
pred.test.small = pred.test[s.test,]
resp.test.small = resp.test[s.test]



x.train.small = model.matrix(resp.train.small~.,pred.train.small)[,-1]
y.train.small = resp.train.small
x.test.small  = model.matrix(resp.test.small~.,pred.test.small)[,-1]
y.test.small  = resp.test.small

out3 = glmnet(x.train.small, y.train.small, alpha=0)


set.seed(100)
cv.out3= cv.glmnet(x.train.small,y.train.small, alpha=0)
plot(cv.out3)
```




```
bestlam3 = cv.out3$lambda.min
bestlam3
```

```
## [1] 0.05102077
```

```
coef(cv.out3)
```

```
## 17 x 1 sparse Matrix of class "dgCMatrix"
##                      s1
## (Intercept)  2.55871471
## mag.i        0.10578433
## col.Vi      -0.37896242
## col.iJ       0.15998059
## col.JH       0.18923881
## V.G         -0.08306386
## V.M20        0.10027477
## V.C          0.03599318
## V.size      -0.09213944
## J.G         -0.68426339
## J.M20        0.07687850
## J.C          0.04032311
## J.size      -0.30642202
## H.G         -0.54551033
## H.M20        0.10767320
## H.C          0.03934846
## H.size      -0.21901367
```

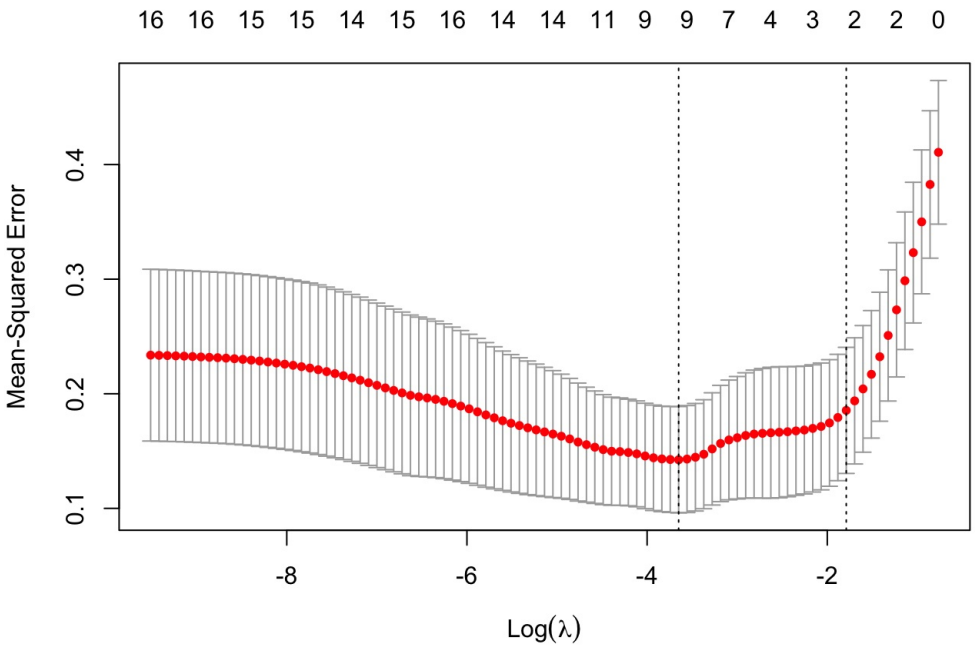
```
ridges_pred = predict(out3, s = bestlam3, newx = x.test.small)
ridge_mse= mean((ridges_pred - y.test.small)^2)

ridge_mse
```

```
## [1] 0.138608
```

Question 8

Repeat the lasso analysis from above using the small datasets. (Use your code from Q6, with `alpha` set to 1.) Do you observe any qualitative difference in the result in the small data limit? Are those coefficients that are shrunk to zero brightness coefficients, or morphological ones? (Or a mix?) Is the result surprising? And which gives the smaller test-set MSE: ridge regression or lasso? Run a full linear regression on these data, and compute the linear regression test-set MSE. How much does the MSE improve when we use lasso and ridge regression as opposed to just straight-up linear regression?

```
out4 = glmnet(x.train.small, y.train.small, alpha=1)

set.seed(100)
cv.out4= cv.glmnet(x.train.small,y.train.small, alpha=1)
plot(cv.out4)
```




```
bestlam4 = cv.out4$lambda.min
bestlam4
```

```
## [1] 0.02599069
```

```
coef(cv.out4)
```

```
## 17 x 1 sparse Matrix of class "dgCMatrix"
##                     s1
## (Intercept)  1.5510951
## mag.i        0.1604592
## col.Vi      -0.2412879
## col.iJ       .
## col.JH       .
## V.G          .
## V.M20        .
## V.C          .
## V.size       .
## J.G          .
## J.M20        .
## J.C          .
## J.size       .
## H.G          .
## H.M20        .
## H.C          .
## H.size       .
```

```
las_pred = predict(out4, s = bestlam4, newx = x.test.small)
las_pred_mse= mean((las_pred - y.test.small)^2)


names(predictors)
```

```
##  [1] "mag.i"  "col.Vi" "col.iJ" "col.JH" "V.G"    "V.M20"  "V.C"    "V.size"
##  [9] "J.G"    "J.M20"  "J.C"    "J.size" "H.G"    "H.M20"  "H.C"    "H.size"
```

```
lm.mod=lm(resp.train~.,data=df_train)
lm.mod
```

```
##
## Call:
## lm(formula = resp.train ~ ., data = df_train)
##
## Coefficients:
## (Intercept)        mag.i       col.Vi       col.iJ       col.JH          V.G
##    1.403110     0.452703    -0.002304    -0.726179     0.417476     0.120532
##       V.M20          V.C       V.size          J.G        J.M20          J.C
##    0.069417     0.014847    -0.067729     3.512991    -0.145402     0.044207
##      J.size          H.G        H.M20          H.C       H.size
##   -1.210477    -3.088045     0.427324     0.229849     1.706867
```

```
pred.lm.mod= predict.lm(lm.mod, df_test)
lm_pred_mse= mean((df_test$response - pred.lm.mod)^2)

ridge_mse
```

```
## [1] 0.138608
```

```
las_pred_mse
```

```
## [1] 0.09273938
```

```
lm_pred_mse
```

```
## [1] NaN
```

```
The MSE improves significantly when we use either lasso or ridge regression over linear regression. Out of the two
lasso and ridge regression, lasso has lower MSE associated with it.

The coefficients that are shrunk to zero are brightness coefficients

There are slight qualitative differencesin the small data limit.
```