Homework 3 - Stat 495

Justin Papagelis

Due Friday, Sept. 30th by midnight

Practicing Academic Integrity

If you worked with others or used resources outside of provided course material (anything besides our textbook(s), course materials in Moodle/Git repo, R help menu) to complete this assignment, please acknowledge them below using a bulleted list.

I acknowledge the following individuals with whom I worked on this assignment:

Name(s) and corresponding problem(s)

-

I used the following sources to help complete this assignment:

Source(s) and corresponding problem(s)

- https://glmnet.stanford.edu/articles/glmnet.html, 2. (g)

PROBLEMS TO TURN IN: Add(itional) 1-2

Add 1 - Applications to College

Adapted from ISLR

```
data(College)
```

This data set contains information from 1995 on US Colleges from US News and World Report (see help file for details). Our goal is to predict the number of applications received (Apps) using the other variables as potential predictors.

part a: Split the data into training and test data sets. (2/3 - 1/3 split is fine.)

SOLUTION:

```
set.seed(1)
n <- nrow(College)
train_index <- sample(1:n, 2/3 * n)
test_index <- setdiff(1:n, train_index)

train <- College[train_index, ]
test <- College[test_index, ]

xColl <- model.matrix(Apps ~ ., College)[, -1]
yColl <- College$Apps

yColl.test <- yColl[test_index]
#n
```

part b: Fit a "kitchen sink" linear regression model on the training set. Report the test error obtained, and comment on any issues with the model (such as conditions, etc.).

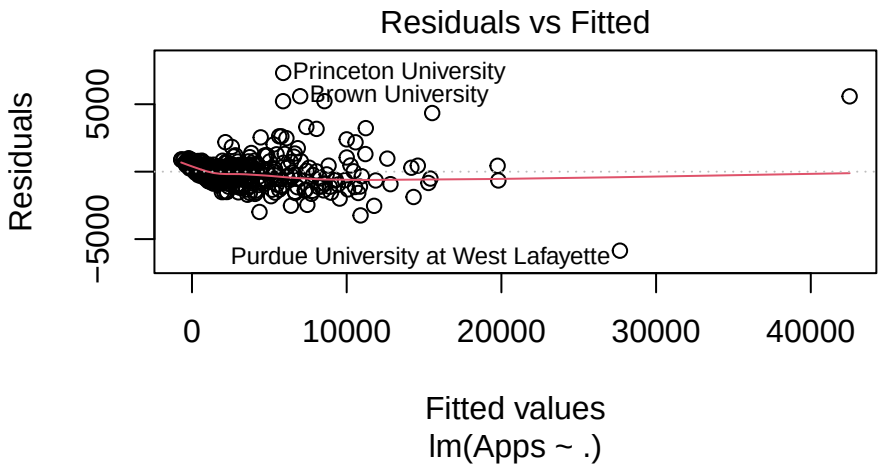
SOLUTION:

The test error that is obtained from this model is 1218487. From the residuals vs fitted plot, there appears to be some issues with linearity because the residuals are clumped to the left of the graph and fan out towards the right. There also appears to be some issues with the Normality condition because the Q-Q Plot deviates from a linear relationship.

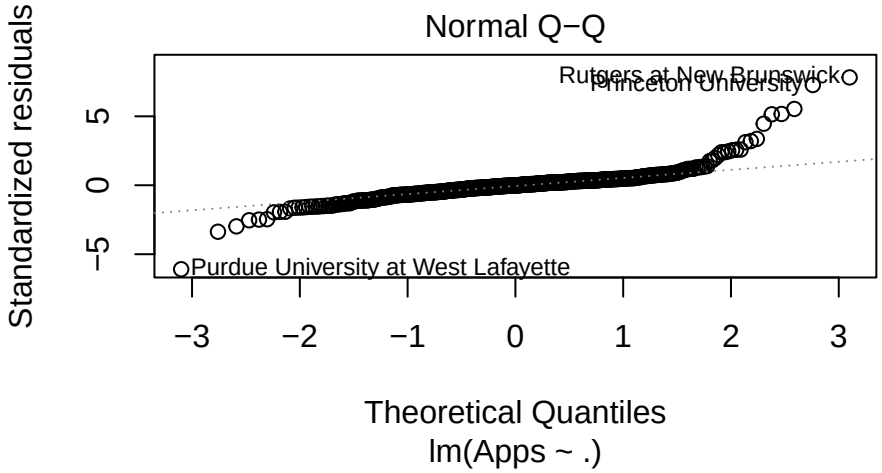
```
mod1 <- lm(Apps ~ ., data = train)
summary(mod1)
```

```
##
## Call:
## lm(formula = Apps ~ ., data = train)
##
## Residuals:
##    Min     1Q Median     3Q    Max
##  -5859   -454     -3    347   7322
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -681.3618   523.9253   -1.30   0.1940
## PrivateYes  -471.6327   168.9366   -2.79   0.0054 **
## Accept         1.7160     0.0485   35.36  < 2e-16 ***
## Enroll        -0.9966     0.2537   -3.93  9.8e-05 ***
## Top10perc     56.1571     6.6556    8.44  3.5e-16 ***
## Top25perc    -16.8725     5.2383   -3.22   0.0014 **
## F.Undergrad    0.0239     0.0450    0.53   0.5952
## P.Undergrad    0.0713     0.0370    1.92   0.0549 .
## Outstate      -0.0910     0.0223   -4.08  5.3e-05 ***
## Room.Board     0.1801     0.0584    3.08   0.0022 **
## Books          0.3828     0.3270    1.17   0.2424
## Personal      -0.0139     0.0751   -0.19   0.8529
## PhD          -12.8984     5.7056   -2.26   0.0242 *
## Terminal       1.6509     6.2262    0.27   0.7910
## S.F.Ratio     21.3476    16.5323    1.29   0.1972
## perc.alumni    2.2706     4.9858    0.46   0.6490
## Expend         0.0597     0.0137    4.35  1.7e-05 ***
## Grad.Rate      7.1511     3.6085    1.98   0.0481 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1030 on 500 degrees of freedom
## Multiple R-squared:  0.933,  Adjusted R-squared:  0.931
## F-statistic:  409 on 17 and 500 DF,  p-value: <2e-16
```

```
plot(mod1, 1)
```




```
plot(mod1, 2)
```




```
full_pred <- predict.lm(mod1, test)

mean((full_pred - yColl.test)^2)
```

```
## [1] 1218487
```

part c: Fit a linear regression model using an automated variable selection method of your choice (from Stat 230) on the training set. Report the test error obtained, and comment on any issues with the model (such as conditions, etc.).

SOLUTION: The test error that is obtained from this model is 1200584. There appears to be similar issues with this model as the previous one with Linearity and Normality.

```
intercept_only <- lm(Apps ~ 1, data = train)

all <- lm(Apps ~ ., data = train)

both <- MASS::stepAIC(intercept_only, direction = 'both', scope = list(upper = all, lower = ~1), trace =

both$anova
```

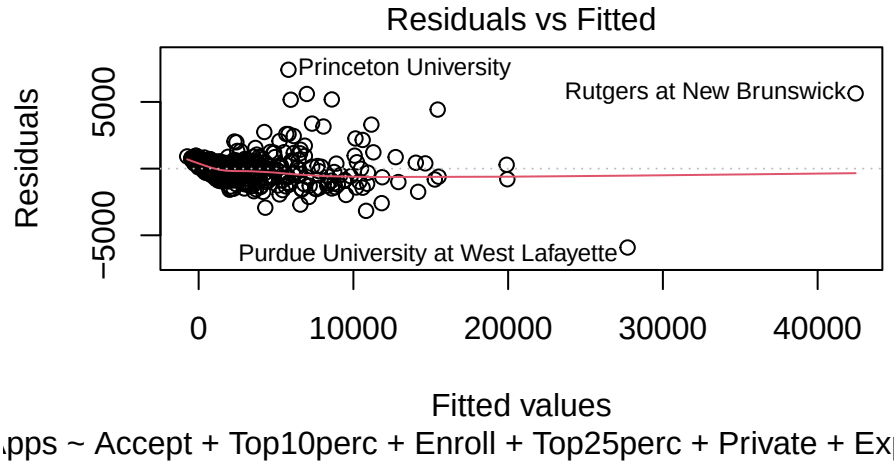
```
## Stepwise Model Path
## Analysis of Deviance Table
##
## Initial Model:
## Apps ~ 1
##
## Final Model:
## Apps ~ Accept + Top10perc + Enroll + Top25perc + Private + Expend +
##     Outstate + Room.Board + PhD + Grad.Rate + P.Undergrad
##
##
##              Step Df   Deviance Resid. Df Resid. Dev     AIC
## 1                                     517 7930734542 8571.81
## 2       + Accept  1 7078198526       516  852536017 7418.52
## 3    + Top10perc  1  202742351       515  649793666 7279.85
## 4       + Enroll  1   20841078       514  628952587 7264.97
## 5    + Top25perc  1   18417667       513  610534921 7251.57
## 6      + Private  1   20752310       512  589782611 7235.66
## 7       + Expend  1    9624694       511  580157917 7229.14
## 8     + Outstate  1   15618283       510  564539634 7217.00
## 9   + Room.Board  1   12758027       509  551781606 7207.16
## 10         + PhD  1    7998026       508  543783580 7201.60
## 11   + Grad.Rate  1    3299705       507  540483875 7200.44
## 12 + P.Undergrad  1    4801023       506  535682853 7197.82
```

```
#both$coefficients

summary(both)
```

```
##
## Call:
## lm(formula = Apps ~ Accept + Top10perc + Enroll + Top25perc +
##     Private + Expend + Outstate + Room.Board + PhD + Grad.Rate +
##     P.Undergrad, data = train)
##
## Residuals:
##    Min     1Q Median     3Q    Max
##  -5916   -458     -6    326   7414
```

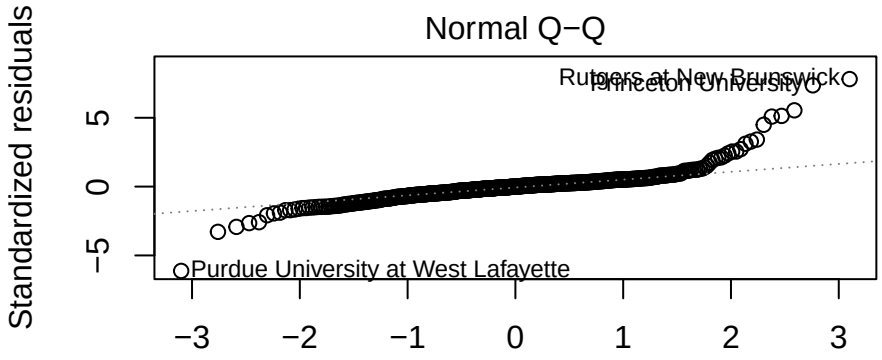
```
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -45.5073   321.1055   -0.14   0.8874
## Accept         1.7169     0.0480   35.80  < 2e-16 ***
## Top10perc     56.1460     6.5443    8.58  < 2e-16 ***
## Enroll        -0.8798     0.1471   -5.98  4.2e-09 ***
## Top25perc    -16.4799     5.1211   -3.22   0.0014 **
## PrivateYes  -527.6617   160.2414   -3.29   0.0011 **
## Expend         0.0537     0.0126    4.25  2.5e-05 ***
## Outstate      -0.0927     0.0211   -4.39  1.4e-05 ***
## Room.Board     0.1846     0.0570    3.24   0.0013 **
## PhD          -11.6606     3.9397   -2.96   0.0032 **
## Grad.Rate      7.3744     3.4378    2.15   0.0324 *
## P.Undergrad    0.0760     0.0357    2.13   0.0337 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1030 on 506 degrees of freedom
## Multiple R-squared:  0.932,  Adjusted R-squared:  0.931
## F-statistic:  635 on 11 and 506 DF,  p-value: <2e-16
```

```
plot(both, 1)
```



ıpps ~ Accept + Top10perc + Enroll + Top25perc + Private + Expend

```
plot(both, 2)
```



pps ~ Accept + Top10perc + Enroll + Top25perc + Private + Expend

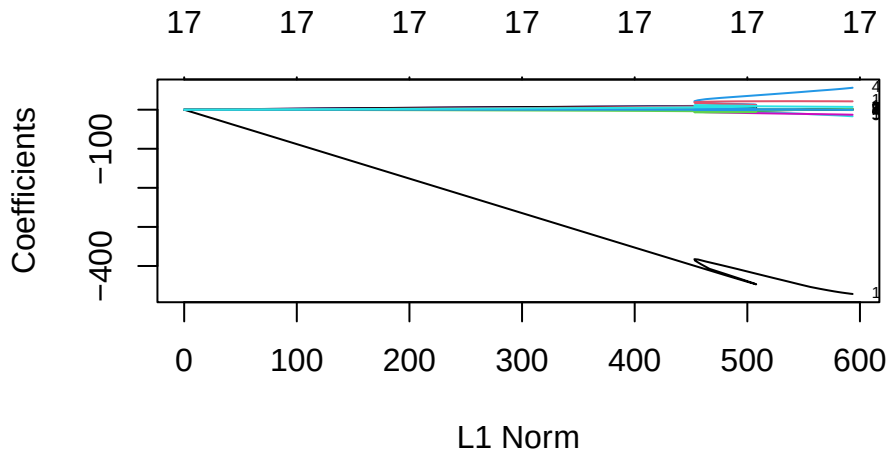
```
selection_pred <- predict.lm(both, test)
mean((selection_pred - yColl.test)^2)
```

```
## [1] 1200584
```

part d: Fit a ridge regression model on the training set, with lambda chosen by cross-validation. Report the lambda chosen and the test error obtained.

SOLUTION:

```
xColl <- model.matrix(Apps ~ ., College)[, -1]
yColl <- College$Apps

x_train <- model.matrix(Apps ~ ., train)[, -1]
x_test <- model.matrix(Apps ~ ., test)[, -1]


y_train <- train %>%
  select(Apps) %>%
  unlist() %>%
  as.numeric()

y_test <- test %>%
  select(Apps) %>%
  unlist() %>%
  as.numeric()

grid <- 10^seq(10, -2, length = 100)

ridge_mod <- glmnet(x_train, y_train, alpha = 0, lambda = grid)

plot(ridge_mod, label = TRUE)
```

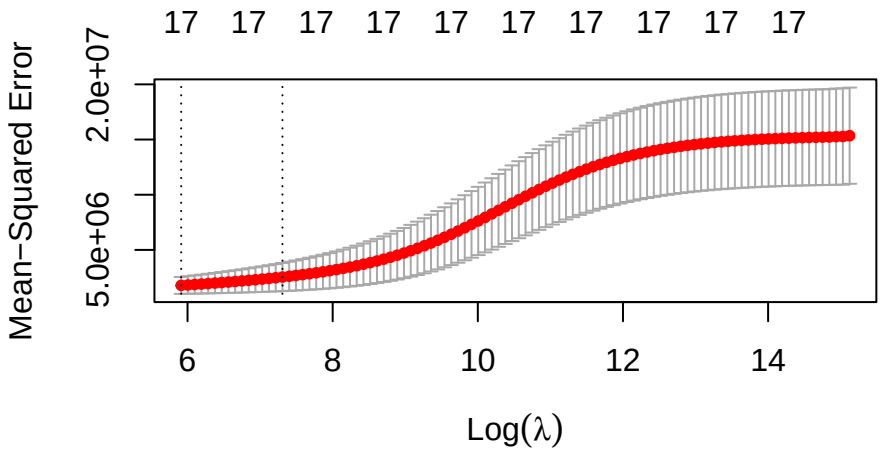



```
#ridge_pred <- predict(ridge_mod, s = 4, newx = x_test)
#mean((ridge_pred - y_test)^2)

set.seed(1)

cv.out <- cv.glmnet(x_train, y_train, alpha = 0)
```

```
bestlam <- cv.out$lambda.min
bestlam
```

```
## [1] 369.655
```

```
plot(cv.out)
```




```
predict(cv.out, type = "coefficients", s = bestlam)[1:18,]
```

```
##  (Intercept)   PrivateYes       Accept       Enroll    Top10perc    Top25perc
## -1.80296e+03 -3.90059e+02  1.10919e+00  4.12673e-01  2.79652e+01 -1.25214e-01
##  F.Undergrad  P.Undergrad     Outstate   Room.Board        Books     Personal
##  5.54690e-02  3.21280e-02 -3.53207e-02  2.33974e-01  4.80206e-01 -3.92415e-02
##          PhD     Terminal    S.F.Ratio  perc.alumni       Expend    Grad.Rate
## -6.65143e+00 -3.15874e+00  2.06381e+01 -4.72147e+00  6.44701e-02  9.77869e+00
```

```
ridge_pred <- predict(ridge_mod, s = bestlam, newx = x_test)
mean((ridge_pred - y_test)^2)
```

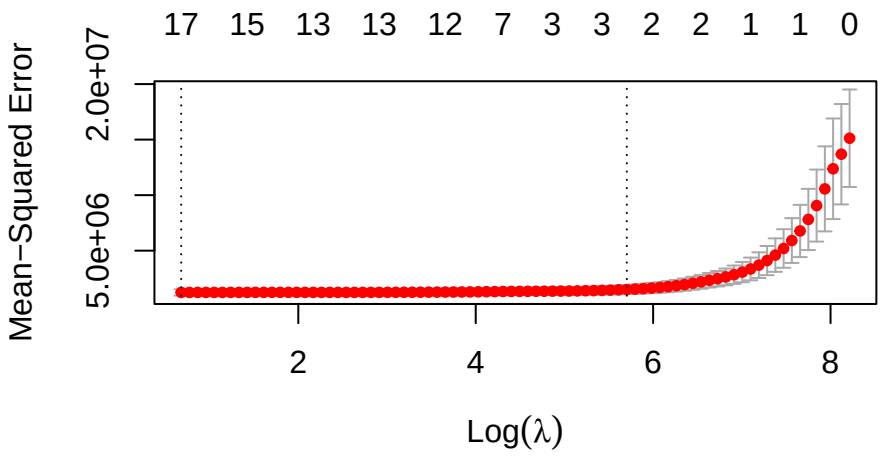
```
## [1] 1111300
```

The lambda that was chosen by cross-validation is 369.655 and the test error obtained was 1111300.

part e: Fit a lasso model on the training set, with lambda chosen by cross-validation. Report the lambda chosen and the test error obtained.

SOLUTION: The lambda that was chosen was 1.97274 and the test error obtained was 1210193.

```
set.seed(1)
cvlassoCol.mod <- cv.glmnet(xColl[train_index,], yColl[train_index], alpha = 1)

plot(cvlassoCol.mod)
```




```
bestlamColl <- cvlassoCol.mod$lambda.min
bestlamColl
```

```
## [1] 1.97274
```

```
predict(cvlassoCol.mod, type = "coefficients", s = bestlamColl)[1:18,]
```

```
##  (Intercept)   PrivateYes       Accept       Enroll    Top10perc    Top25perc
## -6.82190e+02 -4.71309e+02  1.70045e+00 -8.64632e-01  5.44989e+01 -1.56287e+01
##  F.Undergrad  P.Undergrad     Outstate   Room.Board        Books     Personal
##  6.04164e-03  7.02688e-02 -8.80059e-02  1.78550e-01  3.71807e-01 -8.59078e-03
##          PhD     Terminal    S.F.Ratio  perc.alumni       Expend    Grad.Rate
## -1.18837e+01  6.78491e-01  2.03724e+01  1.57963e+00  5.92435e-02  6.90918e+00
```

```
lassoCol.pred <- predict(cvlassoCol.mod, s = bestlamColl, newx = xColl[test_index,])
mean((lassoCol.pred - yColl.test)^2)
```

```
## [1] 1210193
```

part f: Comment on the results obtained across the four models. Address the following questions as part of your response. Do the test errors differ much? Do the coefficients differ greatly? In particular, if any variables were left out of the model in (c) or (e), is there any insight that they might have been removed based on the models in (b) or (d)? Which final model would you select here? Why?

SOLUTION:

The MSE for the "kitchen sink" model is 1,218,487. The MSE for the model that was obtained using Stepwise Regression has an MSE of 1,200,584 which is a slight decrease from the kitchen sink model. The MSE for the ridge model is 1,111,300 which is a greater improvement on the Stepwise Regression model. Finally, the MSE for the lasso model is 1,210,193 which is worse than the ridge model. Relatively, the test MSEs were similar except for the ridge model which was lower than the rest. It appears the the coefficients for the linear regression models are similar, but the coefficients for the ridge and lasso models are different.

The variables that were left out of the model in (c) were `PrivateYes, F.Undergrad, Outstate, Books, Personal, Terminal, S.F.Ratio, perc.alumni` which are many of the predictors that are not significant in the kitchen sink model. In the ridge regression model, none of the coefficients are made too small, which might be why none of the predictors are removed in the lasso model. We would select the ridge regression model in this case because since it had the smallest MSE for the test set, the model had the best predictions.

Add 2 - Soil predictions

The data comes from a Kaggle competition: https://www.kaggle.com/c/afsis-soil-properties. The original data set contained 3600 variables, 3599 possible predictors (really, 3578 and some other variables) and a response, Sand. The 3599 predictors were reduced to 106 (methods to be taught later this semester) that can "best" distinguish between the two levels of Depth (another variable in the data set). The resulting data set of 107 variables (106 predictors and the response, Sand) was saved in the data set "newsoil". The row numbers should be removed as demonstrated below.

```
newsoil <- read.csv("https://awagaman.people.amherst.edu/stat495/newsoil.csv",
                    header = T)
newsoil <- select(newsoil, -X)
#names(newsoil)
```

Our focus is on predicting the response variable Sand, using the selected variables from previous work.

part a: Split the data set into a training and test set (75/25), with a seed of your choice. (Hint: see lab for code.) You may also wish to create appropriate x and y matrices for future function inputs at the same time.

SOLUTION:

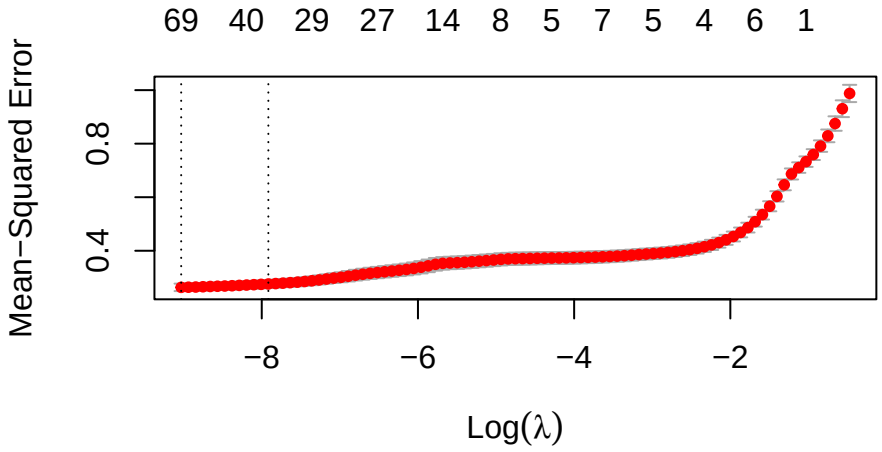
```
set.seed(1)
n <- nrow(newsoil)
train_index_soil <- sample(1:n, 3/4 * n)
test_index_soil <- setdiff(1:n, train_index_soil)

train_soil <- newsoil[train_index_soil, ]
test_soil <- newsoil[test_index_soil, ]


xSoil <- model.matrix(Sand ~ ., newsoil)[, -1]
ySoil <- newsoil$Sand

ySoil.test <- ySoil[test_index_soil]
```

part b: Fit lasso models to predict Sand using all the possible predictors. Choose two lasso models - one that has a "best" lambda determined in some appropriate way, and another model with a different non-zero lambda of your choice. How many slope coefficients are set to 0 in each of your chosen lasso models?

SOLUTION: There are 37 coefficients set to 0 in the model with the "best" lambda of 0.000119548 which was found using cross-validation. We made another lasso model with a lambda of 0.001 which 79 of the coefficients were set to 0.

FINSIH ANSWER

```
set.seed(1)
cvlassoSoil.mod <- cv.glmnet(xSoil[train_index_soil,], ySoil[train_index_soil], alpha = 1)

plot(cvlassoSoil.mod)
```




```
bestlamSoil <- cvlassoSoil.mod$lambda.min
bestlamSoil
```

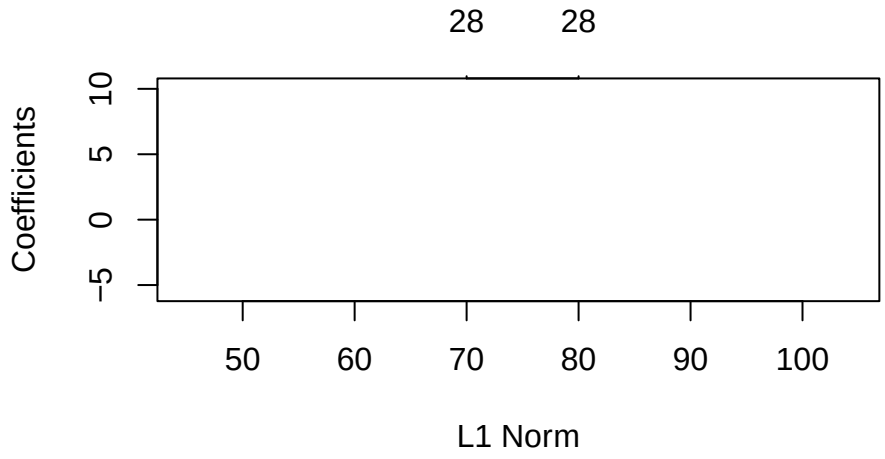
```
## [1] 0.000119548
```

```
predict(cvlassoSoil.mod, type = "coefficients", s = bestlamSoil)[1:107,]
```

```
##  (Intercept)     m3748.98     m3747.05     m3745.13      m3743.2     m3741.27
##  7.74035e-02  5.46344e-02  1.02598e+00  5.92556e-03  0.00000e+00 -4.20297e-01
##     m3739.34     m3737.41     m3735.48     m3733.55     m3731.63      m3729.7
## -1.36869e+00 -4.64005e+00  0.00000e+00  9.05499e-01  3.95229e-04  1.26416e-02
##     m3727.77     m3725.84     m3723.91     m3721.98     m3720.05     m3718.13
##  4.85134e-01  1.88340e+00  6.57217e-01  2.46403e-01  6.86651e-01  0.00000e+00
##      m3716.2     m3714.27     m3712.34     m3710.41     m3708.48     m3706.56
##  0.00000e+00 -6.88539e-01 -1.92684e+01  8.15698e-05  2.28430e+01  0.00000e+00
##     m3704.63      m3702.7     m3700.77     m3698.84     m3696.91     m3694.98
##  0.00000e+00  0.00000e+00 -9.87983e-05 -1.06648e+00 -1.02649e+01  0.00000e+00
```

```
##     m3693.06     m3691.13      m3689.2     m3687.27     m3685.34     m3683.41
##  9.48413e-02  5.36794e+00  0.00000e+00 -2.68817e-02  0.00000e+00  2.89592e+00
##     m3681.48     m3679.56     m3677.63      m3675.7     m3673.77     m3671.84
##  1.04217e+00  1.10494e-04  3.20862e-03  1.83739e-01  0.00000e+00 -7.42151e-05
##     m3669.91     m3667.99     m3666.06     m3664.13      m3662.2     m3660.27
## -7.12848e-01  0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00 -7.60659e+00
##     m3658.34     m3656.41     m3654.49     m3652.56     m3650.63      m3648.7
## -6.82074e+00 -1.84489e+00  0.00000e+00  0.00000e+00  1.55660e-01  1.77833e+01
##     m3646.77     m3644.84     m3642.92     m3640.99     m3639.06     m3637.13
##  3.21739e+00  0.00000e+00 -3.43734e-05 -1.70311e-01 -1.30465e+00 -2.42532e+00
##      m3635.2     m3633.27     m3631.34     m3629.42     m3627.49     m3625.56
## -8.42401e+00 -3.07095e-01  0.00000e+00  1.50817e+00  5.29293e+00  0.00000e+00
##     m3623.63      m3621.7     m3619.77     m3617.84     m3615.92     m3613.99
## -3.54644e+00 -4.70823e+00  7.61513e-05  1.04512e+01  4.96269e-02  0.00000e+00
##     m3612.06     m3610.13      m3608.2     m3606.27     m3604.35     m3602.42
## -5.30606e+00 -1.08471e+01 -4.89376e-01 -8.19368e-01  0.00000e+00  0.00000e+00
##     m3600.49     m1839.78     m1837.85     m1835.92     m1833.99     m1832.06
##  1.25613e+01  1.36879e+01  0.00000e+00  0.00000e+00 -8.80296e-01 -4.52430e+00
##     m1830.14     m1828.21     m1826.28     m1824.35     m1822.42     m1820.49
## -7.05790e+00 -6.86806e-01  0.00000e+00  0.00000e+00  0.00000e+00  4.12647e+00
##      m1577.5     m1575.58     m1573.65     m1571.72     m1569.79     m1567.86
## -3.47647e+00 -8.28673e-02  0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00
##     m1565.93        m1564     m1562.08     m1560.15     m1558.22     m1556.29
## -1.29997e-01 -5.22259e-05 -9.74659e-01  0.00000e+00  0.00000e+00 -1.25097e-02
##     m1554.36     m1552.43      m1550.5     m1548.58     m1546.65
## -9.09412e-03  0.00000e+00  0.00000e+00  0.00000e+00  3.38105e+00
```

```
set.seed(1)
lassoSoil.mod <- glmnet(xSoil[train_index_soil,], ySoil[train_index_soil], alpha = 1, lambda = 0.001)

plot(lassoSoil.mod)
```




```
predict(lassoSoil.mod, type = "coefficients", s = 0.001)[1:107,]
```

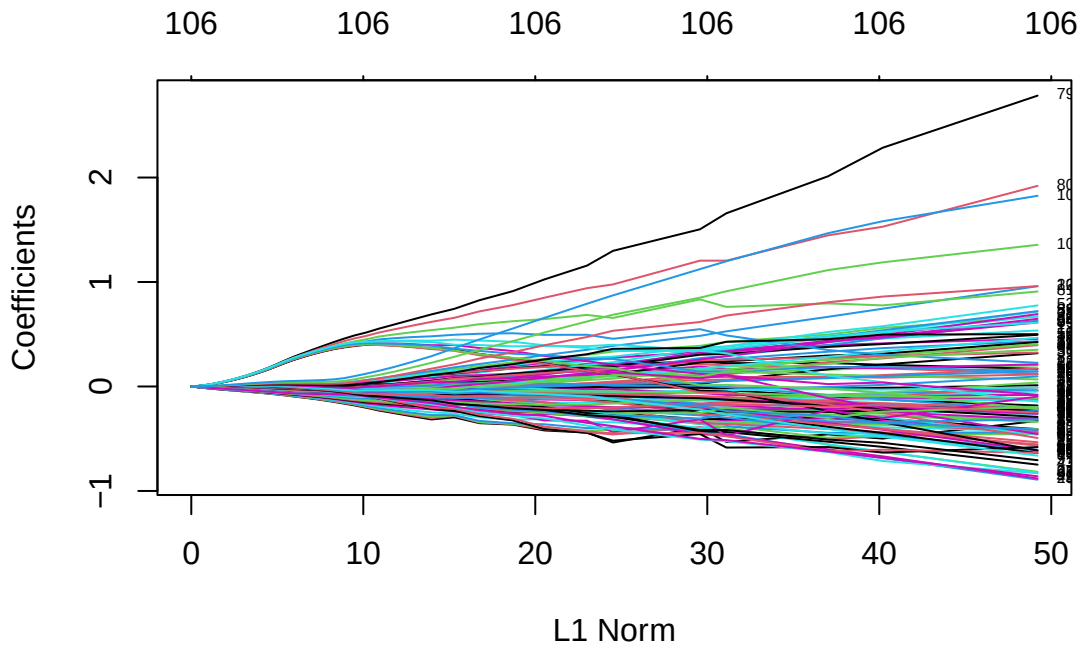
```
##  (Intercept)     m3748.98     m3747.05     m3745.13      m3743.2     m3741.27
```

```
## -0.152642984  0.000000000  0.000000000  0.000000000  0.000000000  0.000000000
##     m3739.34     m3737.41     m3735.48     m3733.55     m3731.63      m3729.7
## -0.728043473 -2.208964584  0.000000000  0.000000000  0.000000000  0.000000000
##     m3727.77     m3725.84     m3723.91     m3721.98     m3720.05     m3718.13
##  0.000000000  0.000000000  0.000000000  0.000000000  0.000000000  0.000000000
##      m3716.2     m3714.27     m3712.34     m3710.41     m3708.48     m3706.56
##  0.000000000 -4.635562911 -2.294841331  0.000000000 10.165344279  0.000000000
##     m3704.63      m3702.7     m3700.77     m3698.84     m3696.91     m3694.98
##  0.000000000  0.000000000  0.000000000 -2.175997586 -4.142339310  0.000000000
##     m3693.06     m3691.13      m3689.2     m3687.27     m3685.34     m3683.41
##  0.000000000  0.000000000  0.000000000  0.000000000  2.802541411  4.133741497
##     m3681.48     m3679.56     m3677.63      m3675.7     m3673.77     m3671.84
##  0.938334704  0.000000000  0.000000000  0.000000000  0.000000000 -2.527718405
##     m3669.91     m3667.99     m3666.06     m3664.13      m3662.2     m3660.27
## -4.517293620  0.000000000  0.000000000  0.000000000  0.000000000 -0.002203729
##     m3658.34     m3656.41     m3654.49     m3652.56     m3650.63      m3648.7
## -0.000397358  0.000000000  0.000000000  0.000000000  0.000000000  7.374188600
##     m3646.77     m3644.84     m3642.92     m3640.99     m3639.06     m3637.13
##  0.000000000  0.000000000  0.000000000  0.000000000  0.000000000 -0.925435089
##      m3635.2     m3633.27     m3631.34     m3629.42     m3627.49     m3625.56
## -5.600490200 -1.976689442  0.000000000  0.000000000  0.000000000  0.000000000
##     m3623.63      m3621.7     m3619.77     m3617.84     m3615.92     m3613.99
##  0.000000000  0.000000000  0.139694246  0.392355554  0.000000000  0.000000000
##     m3612.06     m3610.13      m3608.2     m3606.27     m3604.35     m3602.42
##  0.000000000  0.000000000  0.000000000  0.000000000  0.000000000  0.000000000
##     m3600.49     m1839.78     m1837.85     m1835.92     m1833.99     m1832.06
##  1.980648573  6.402594978  0.000000000  0.000000000  0.000000000  0.000000000
##     m1830.14     m1828.21     m1826.28     m1824.35     m1822.42     m1820.49
## -1.706407558 -0.041474145  0.000000000  0.000000000  0.000000000  0.000000000
##      m1577.5     m1575.58     m1573.65     m1571.72     m1569.79     m1567.86
##  0.000000000 -2.143993184  0.000000000  0.000000000  0.000000000  0.000000000
##     m1565.93         m1564     m1562.08     m1560.15     m1558.22     m1556.29
## -1.629542693 -0.104438711  0.000000000  0.000000000  0.000000000  0.000000000
##     m1554.36     m1552.43      m1550.5     m1548.58     m1546.65
##  0.000000000  0.000000000  0.000000000  0.000000000  2.933747179
```

part c: Fit a ridge model to predict Sand using all the possible predictors. How many slope coefficients are set to 0 in your ridge model?

SOLUTION: None of the slope coefficients are set to 0 in the ridge model.

```
x_train_soil <- model.matrix(Sand ~ ., train_soil)[, -1]
x_test_soil <- model.matrix(Sand ~ ., test_soil)[, -1]


y_train_soil <- train_soil %>%
  select(Sand) %>%
  unlist() %>%
  as.numeric()

y_test_soil <- test_soil %>%
  select(Sand) %>%
  unlist() %>%
  as.numeric()

grid <- 10^seq(10, -2, length = 100)

ridge_mod_soil <- glmnet(x_train_soil, y_train_soil, alpha = 0, lambda = grid)

plot(ridge_mod_soil, label = TRUE)
```




```
set.seed(1)

cv.out <- cv.glmnet(x_train_soil, y_train_soil, alpha = 0)
```

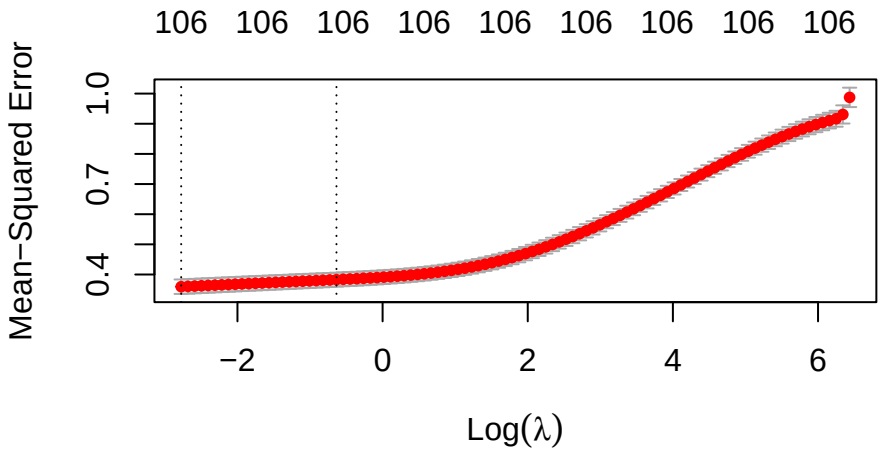
```r
bestlam <- cv.out$lambda.min
bestlam
```

```
## [1] 0.0623326
```

```r
predict(cv.out, type = "coefficients", s = bestlam)[1:107,]
```

```
##  (Intercept)     m3748.98     m3747.05     m3745.13      m3743.2     m3741.27
## -0.891433529 -0.474465263 -0.436031850 -0.401578986 -0.373287728 -0.323335982
##     m3739.34     m3737.41     m3735.48     m3733.55     m3731.63      m3729.7
## -0.271291728 -0.239760445 -0.242481331 -0.236371878 -0.186944114 -0.136776434
##     m3727.77     m3725.84     m3723.91     m3721.98     m3720.05     m3718.13
## -0.106188835 -0.058230213  0.009469775  0.089582916  0.170269521  0.224856061
##      m3716.2     m3714.27     m3712.34     m3710.41     m3708.48     m3706.56
##  0.237579571  0.198913640  0.190172218  0.275407607  0.308404897  0.233272604
##     m3704.63      m3702.7     m3700.77     m3698.84     m3696.91     m3694.98
##  0.119322457  0.004248959 -0.110438852 -0.203410097 -0.242648790 -0.204369937
##     m3693.06     m3691.13      m3689.2     m3687.27     m3685.34     m3683.41
## -0.110919574 -0.019383382  0.011109022  0.034053185  0.090126567  0.120272485
##     m3681.48     m3679.56     m3677.63      m3675.7     m3673.77     m3671.84
##  0.115167644  0.072582325  0.004222299 -0.086802130 -0.172331044 -0.219963127
##     m3669.91     m3667.99     m3666.06     m3664.13      m3662.2     m3660.27
## -0.233233693 -0.203248387 -0.150742283 -0.090940929 -0.048180512 -0.025892342
##     m3658.34     m3656.41     m3654.49     m3652.56     m3650.63      m3648.7
## -0.013127900 -0.000438825  0.027446078  0.079304662  0.150361254  0.166734487
##     m3646.77     m3644.84     m3642.92     m3640.99     m3639.06     m3637.13
##  0.149157633  0.144403522  0.122568141  0.074706405  0.013266350 -0.039074915
##      m3635.2     m3633.27     m3631.34     m3629.42     m3627.49     m3625.56
## -0.080000096 -0.101278626 -0.107515995 -0.094076404 -0.079683645 -0.115780234
##     m3623.63      m3621.7     m3619.77     m3617.84     m3615.92     m3613.99
## -0.182524374 -0.226136302 -0.207849683 -0.170784287 -0.098092407 -0.064698527
##     m3612.06     m3610.13      m3608.2     m3606.27     m3604.35     m3602.42
## -0.081567181 -0.074117980 -0.059533347 -0.043276520  0.004339871  0.058698368
##     m3600.49     m1839.78     m1837.85     m1835.92     m1833.99     m1832.06
##  0.086654311  1.079641494  0.858314010  0.625231351  0.469275979  0.374570386
##     m1830.14     m1828.21     m1826.28     m1824.35     m1822.42     m1820.49
##  0.275819224  0.206452197  0.209876960  0.247350354  0.308232112  0.404689282
##      m1577.5     m1575.58     m1573.65     m1571.72     m1569.79     m1567.86
## -0.384329028 -0.445058884 -0.394996786 -0.249233271 -0.243501400 -0.323789494
##     m1565.93        m1564     m1562.08     m1560.15     m1558.22     m1556.29
## -0.294531382 -0.226748487 -0.091881496  0.069243773 -0.073294052 -0.067463979
##     m1554.36     m1552.43      m1550.5     m1548.58     m1546.65
##  0.119391685  0.287998466  0.420103293  0.535331225  0.669175325
```

```r
plot(cv.out)
```


106 106 106 106 106 106 106 106 106
Mean−Squared Error
0.4
0.7
1.0
−2
0
2
4
6
Log(λ)

part d: Compute test MSEs for both of your lasso models and your ridge model.

SOLUTION:

The test MSE for the Cross-Validated Lasso Model was 0.262755. The test MSE for the Lasso Model with a lambda of 0.001 is 0.30326. The test MSE for the ridge model is 0.349251.

```
# test MSE for CV Lasso
cvlassoSoil.pred <- predict(cvlassoSoil.mod, s = bestlamSoil, newx = xSoil[test_index_soil,])
mean((cvlassoSoil.pred - ySoil.test)^2)
```

```
## [1] 0.262755
```

```
# test MSE for Lasso
lassoSoil.pred <- predict(lassoSoil.mod, s = 0.001, newx = xSoil[test_index_soil,])
mean((lassoSoil.pred - ySoil.test)^2)
```

```
## [1] 0.30326
```

```
# test MSE for ridge model
ridge_pred_soil <- predict(ridge_mod_soil, s = bestlam, newx = xSoil[test_index_soil,])
mean((ridge_pred_soil - ySoil.test)^2)
```

```
## [1] 0.349251
```

part e: Write a few sentences to address the following questions. Does the test MSE from the model with the "best" lambda suggest it is in fact a better predictive model than your other lasso model? Is ridge better than the lasso models? Which final model would you choose here from these three models? Why?

SOLUTION: The test MSE for the model with the "best" lambda appears to be a better predictive model than the other lasso model because the test MSE for the cross-validated lambda model is 0.262755 whereas the test MSE for the other lasso model is 0.30326. It also appears that the lasso models are better predictive models than the ridge model because the test MSE of the ridge model is 0.349251 which is greater than that of both lasso models. The final model that we would choose is the cross-validated lasso model with a lambda of 0.000119548 because it did the best predictions.

part f: What is the default setting for the normalize option in lars and the standardize option in glmnet? Why is this setting important to the model fit?

SOLUTION: The default setting for the normalize option in lars and the standardize option in glmnet is TRUE. This standardizes the variables so that they are on the same scale. This is important because id we did not standardize, then the variables that are on larger scales would cause their coefficient estimates to have a much larger impact on the model and potentially over influence the model.

part g: Explain what option you would change in order to fit an elastic net penalty (not OLS or ridge) using glmnet.

SOLUTION: In order to fit an elastic net penalty, we would have to adjust α. Since lasso regression has an $\alpha = 1$ and ridge regression has $\alpha = 0$, an $0 < \alpha < 1$ would cause a different behavior than both in selecting and removing predictors.