Homework 3 - Stat 495

Example Solution

Due Monday, Oct. 2 by midnight

PROBLEMS TO TURN IN: Additional 1-3

Additional 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(495)

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, ]

# For ridge and LASSO, you need a different structure, so
# I went ahead and constructed those here as well

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

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

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

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:

We fit a kitchen sink model and check diagnostics, as well as obtain test error using the code below.

```
ksmodel <- lm(Apps ~ . , data = train)
msummary(ksmodel)
```

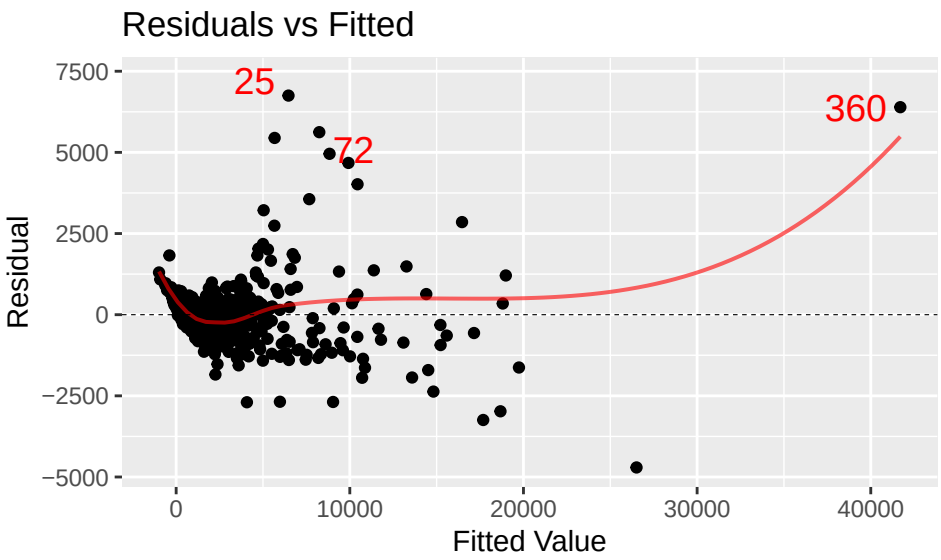
```
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.00e+03   4.97e+02   -2.01    0.045 *
## PrivateYes  -4.12e+02   1.68e+02   -2.45    0.015 *
## Accept       1.68e+00   4.54e-02   36.97  < 2e-16 ***
## Enroll      -1.46e+00   2.26e-01   -6.44  2.8e-10 ***
## Top10perc    2.85e+01   7.09e+00    4.03  6.6e-05 ***
## Top25perc   -6.10e+00   5.54e+00   -1.10    0.271
## F.Undergrad  1.09e-01   3.88e-02    2.82    0.005 **
## P.Undergrad  6.83e-02   3.90e-02    1.75    0.081 .
## Outstate    -1.47e-01   2.37e-02   -6.21  1.1e-09 ***
## Room.Board   8.71e-02   6.04e-02    1.44    0.150
## Books        7.06e-03   2.69e-01    0.03    0.979
## Personal     7.61e-02   7.47e-02    1.02    0.309
## PhD         -1.20e+01   5.52e+00   -2.18    0.030 *
## Terminal    -4.92e+00   6.07e+00   -0.81    0.418
## S.F.Ratio    3.61e+01   1.70e+01    2.12    0.034 *
## perc.alumni  4.01e+00   4.76e+00    0.84    0.400
## Expend       1.93e-01   1.98e-02    9.71  < 2e-16 ***
## Grad.Rate    1.56e+01   3.68e+00    4.22  2.9e-05 ***
##
## Residual standard error: 1020 on 500 degrees of freedom
## Multiple R-squared:  0.94,   Adjusted R-squared:  0.938
## F-statistic:  460 on 17 and 500 DF,  p-value: <2e-16
```

```
kspred <- predict(ksmodel, newdata = test)
mean((kspred - test$Apps)^2)
```

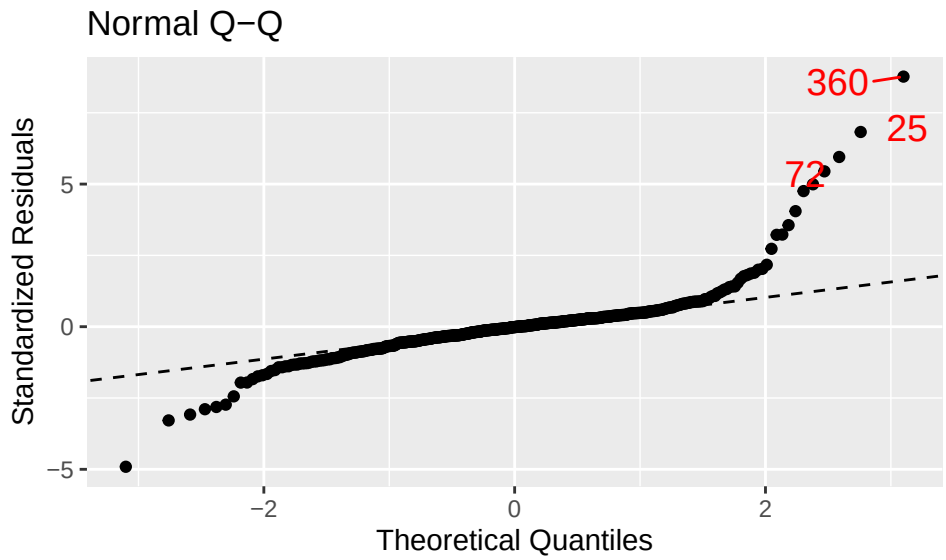
```
## [1] 1634314
```

```
mplot(ksmodel, which = 1)
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



```
mplot(ksmodel, which = 2)
```



For Apps, we find a significant model with an R^2 of about 94 percent on the training data. The MSE from the test data set is 1634314. However, we see some issues with conditions. The residual versus fitted plot shows some increasing variability, outliers, and a potential boundary issue. The QQplot of residuals confirms the presence of some outliers, though the overall pattern does not indicate serious non-normality, except for issues in the upper tail.

Note that based on your seed, you'll see some variability in the test MSE. Your values should be comparable for the rest of your models (i.e., appropriate to compare), but may vary considerably from classmates.

> 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:

Stepwise regression is a popular choice, so that is presented here. Other alternatives include best subsets, forward selection, or backward elimination. Those can all be called with regsubsets from leaps. Note that the MASS library called for stepAIC can interfere with select, so I called it directly from the library.

```
appStep <- MASS::stepAIC(ksmodel, trace = FALSE, direction = "both")
appStep$anova
```

```
## Stepwise Model Path
## Analysis of Deviance Table
##
## Initial Model:
## Apps ~ Private + Accept + Enroll + Top10perc + Top25perc + F.Undergrad +
##     P.Undergrad + Outstate + Room.Board + Books + Personal +
##     PhD + Terminal + S.F.Ratio + perc.alumni + Expend + Grad.Rate
##
## Final Model:
## Apps ~ Private + Accept + Enroll + Top10perc + F.Undergrad +
##     P.Undergrad + Outstate + PhD + S.F.Ratio + Expend + Grad.Rate
##
##
##            Step Df    Deviance Resid. Df Resid. Dev     AIC
```

```
## 1                                    500  516778439 7191.21
## 2       - Books  1     713.638       501  516779152 7189.21
## 3    - Terminal  1  687757.066       502  517466909 7187.90
## 4 - perc.alumni  1  596238.203       503  518063148 7186.50
## 5    - Personal  1  816826.545       504  518879974 7185.31
## 6  - Room.Board  1 1552990.197       505  520432964 7184.86
## 7   - Top25perc  1 1542897.180       506  521975861 7184.40
```

```
appSteplm <- lm(Apps ~ Private + Accept + Enroll + Top10perc + Top25perc + F.Undergrad +
    P.Undergrad + Outstate + Room.Board + PhD + S.F.Ratio + perc.alumni +
    Expend + Grad.Rate, data = train)

msummary(appSteplm)
```

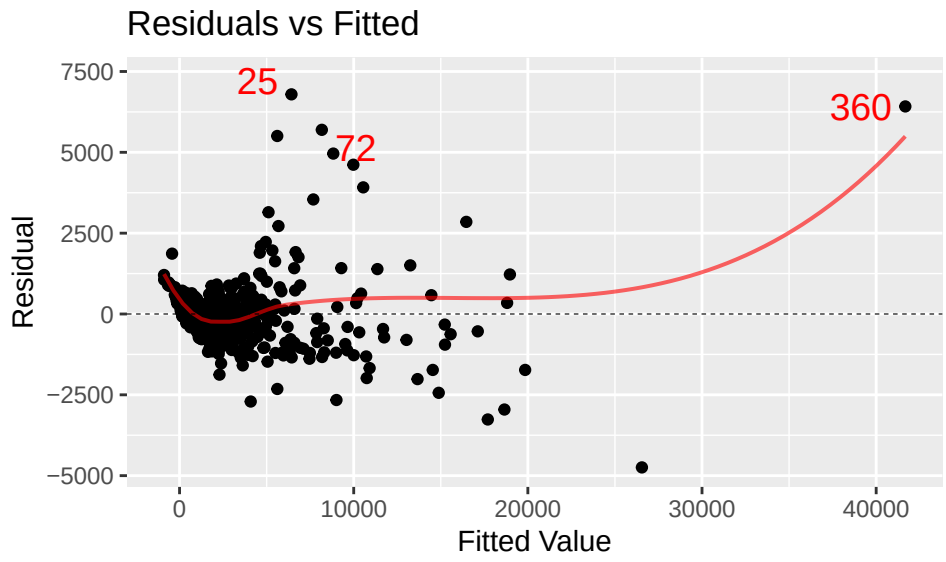
```
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -963.5115   445.8397   -2.16   0.0312 *
## PrivateYes  -381.2466   165.4994   -2.30   0.0217 *
## Accept         1.6790     0.0453   37.06  < 2e-16 ***
## Enroll        -1.4589     0.2260   -6.45  2.6e-10 ***
## Top10perc     29.6426     6.9858    4.24  2.6e-05 ***
## Top25perc     -6.8867     5.4510   -1.26   0.2070
## F.Undergrad    0.1111     0.0386    2.88   0.0042 **
## P.Undergrad    0.0741     0.0386    1.92   0.0556 .
## Outstate      -0.1517     0.0234   -6.49  2.1e-10 ***
## Room.Board     0.0767     0.0590    1.30   0.1941
## PhD          -15.1720     3.7438   -4.05  5.9e-05 ***
## S.F.Ratio     35.6600    16.9440    2.10   0.0358 *
## perc.alumni    2.7609     4.6552    0.59   0.5534
## Expend         0.1928     0.0197    9.77  < 2e-16 ***
## Grad.Rate     15.6920     3.6681    4.28  2.3e-05 ***
##
## Residual standard error: 1020 on 503 degrees of freedom
## Multiple R-squared:  0.94,   Adjusted R-squared:  0.938
## F-statistic:  560 on 14 and 503 DF,  p-value: <2e-16
```

```
pred <- predict(appSteplm, newdata = test)
mean((pred - test$Apps)^2)
```

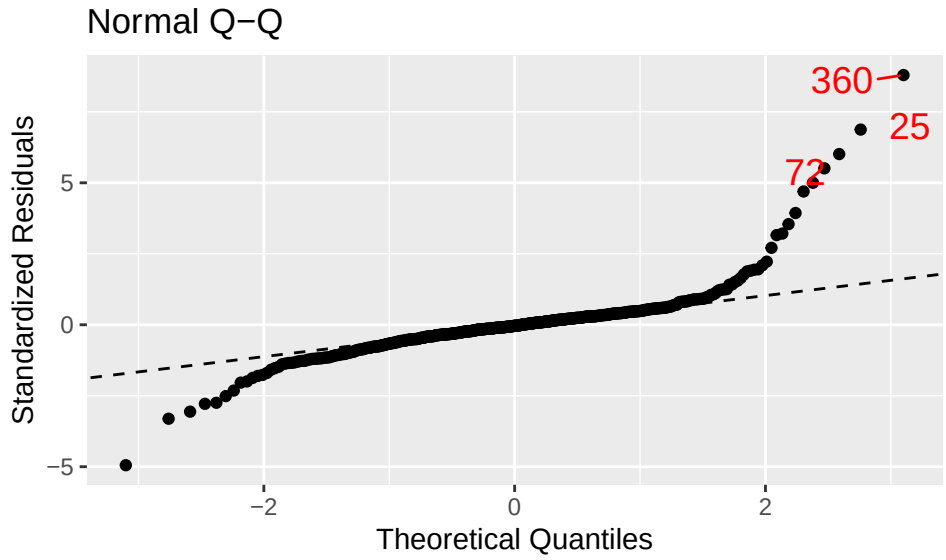
```
## [1] 1629782
```

```
mplot(appSteplm, which = 1)
```

```
## `geom_smooth()` using formula = 'y ~ x'
```




```
mplot(appSteplm, which = 2)
```



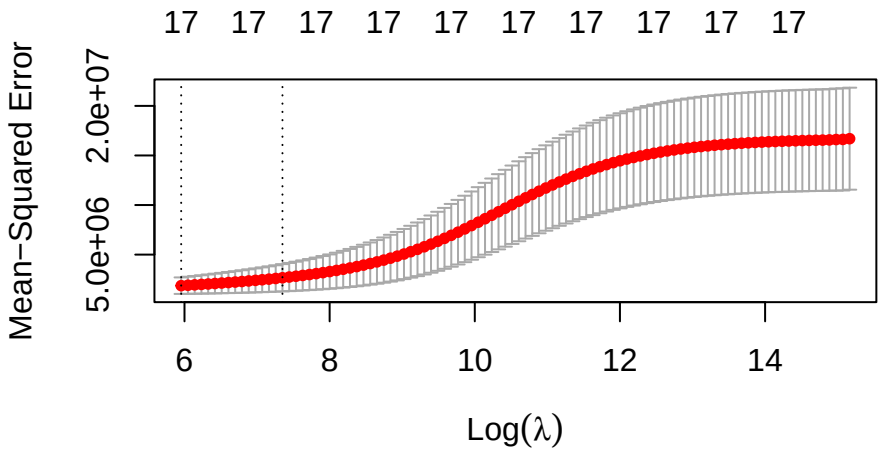
Our stepwise model removed a few predictors from the full model. The final R^2 is about 94 percent. A few predictors are still not significant, but the overall model seems to be doing a decent job. The test MSE is 1629782, slightly better than the kitchen sink model. The plots to assess conditions still show some extreme outliers (in both tails), with some concern for non-normality of the error terms.

> 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:

Now that we have our data structured, we can run cv.glmnet to identify the appropriate lambda for ridge regression via cross-validation.

```
set.seed(495)
cv.out.ridge <- cv.glmnet(x_train, y_train, alpha = 0) # Fit ridge regression model on training data
plot(cv.out.ridge)
```




```
bestlamridge <- cv.out.ridge$lambda.min  # Select lambda that minimizes training MSE
bestlamridge
```

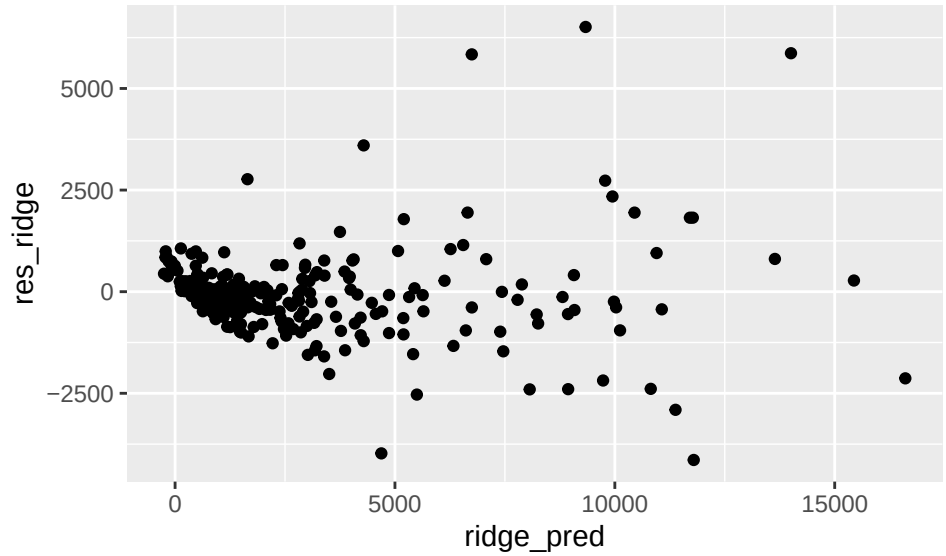
```
## [1] 385.428
```

With the appropriate lambda now found (385.428), we can obtain the test MSE, and check conditions (though this is not required).

```
# note you could just use the cv.out.ridge object instead of refitting this
ridge_mod <- glmnet(x_train, y_train, alpha = 0, thresh = 1e-12)
ridge_pred <- predict(ridge_mod, s = bestlamridge, newx = x_test) # Use best lambda to predict test dat
mean((ridge_pred - y_test)^2) # Calculate test MSE
```

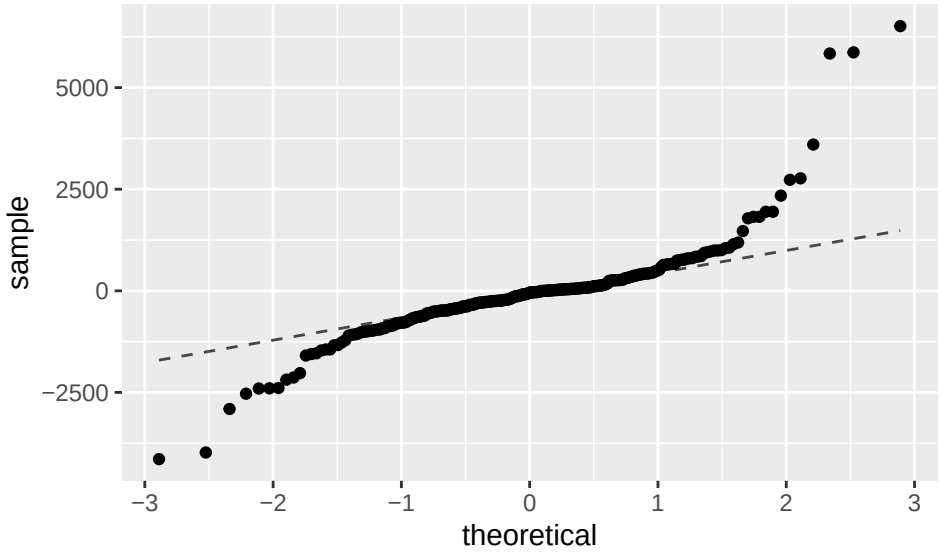
```
## [1] 1238407
```

```
res_ridge <- y_test - ridge_pred
gf_point(res_ridge ~ ridge_pred)
```



```
gf_qq(~ res_ridge) %>% gf_qqline()
```

```
## Warning: The following aesthetics were dropped during statistical transformation: sample
## i This can happen when ggplot fails to infer the correct grouping structure in
##   the data.
## i Did you forget to specify a `group` aesthetic or to convert a numerical
##   variable into a factor?
```



The test MSE for the ridge on Apps is 1238407. The residual plot shows a similar issue to those seen before.

Finally, we pull out the coefficients to examine.

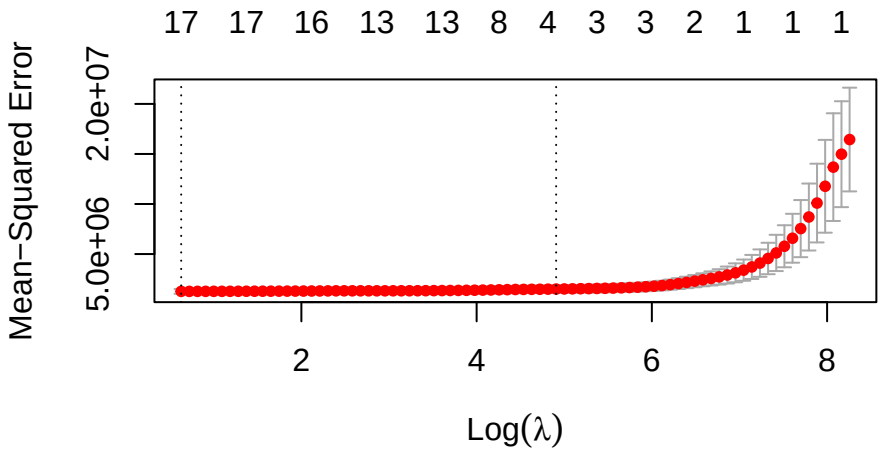
```
round(predict(ridge_mod, type = "coefficients", s = bestlamridge)[1:18,], 2)
```

```
##  (Intercept)   PrivateYes       Accept       Enroll    Top10perc    Top25perc
##     -1772.17      -542.96         1.05         0.32        15.65         3.84
##  F.Undergrad P.Undergrad    Outstate  Room.Board       Books     Personal
##         0.07         0.04        -0.05         0.18         0.08         0.05
##          PhD     Terminal   S.F.Ratio perc.alumni       Expend   Grad.Rate
##        -6.79        -7.01        29.44        -4.75         0.14        14.29
```

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:

```
set.seed(495)
cv.out.lasso <- cv.glmnet(x_train, y_train, alpha = 1) # Fit lasso regression model on training data
plot(cv.out.lasso)
```




```
bestlamlasso <- cv.out.lasso$lambda.min  # Select lambda that minimizes training MSE
bestlamlasso
```

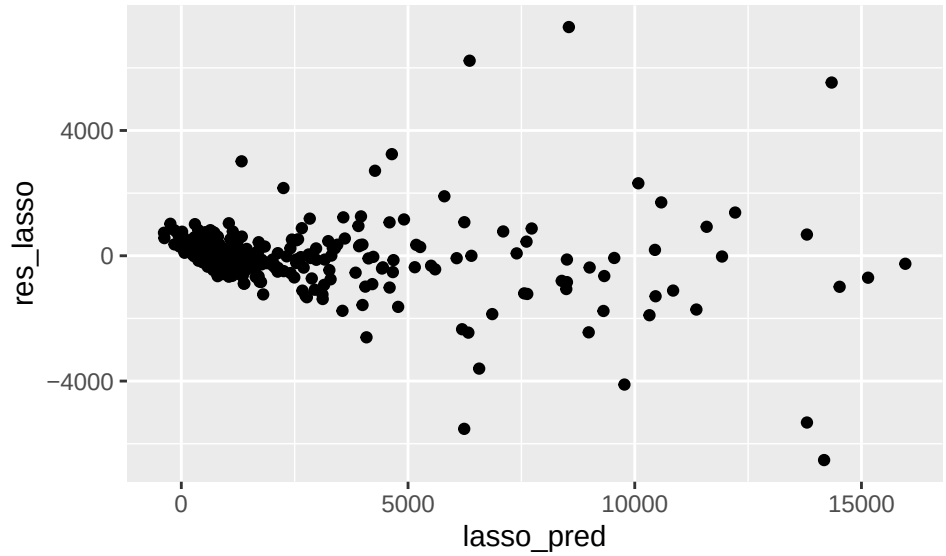
```
## [1] 1.87418
```

With the appropriate lambda now found (1.87418), we can obtain the test MSE, and check conditions (though this is not required).

```
# again, you could just use the cv.out.lasso object instead of refitting
lasso_mod <- glmnet(x_train, y_train, alpha = 1, thresh = 1e-12)
lasso_pred <- predict(lasso_mod, s = bestlamlasso, newx = x_test) # Use best lambda to predict test dat
mean((lasso_pred - y_test)^2) # Calculate test MSE
```

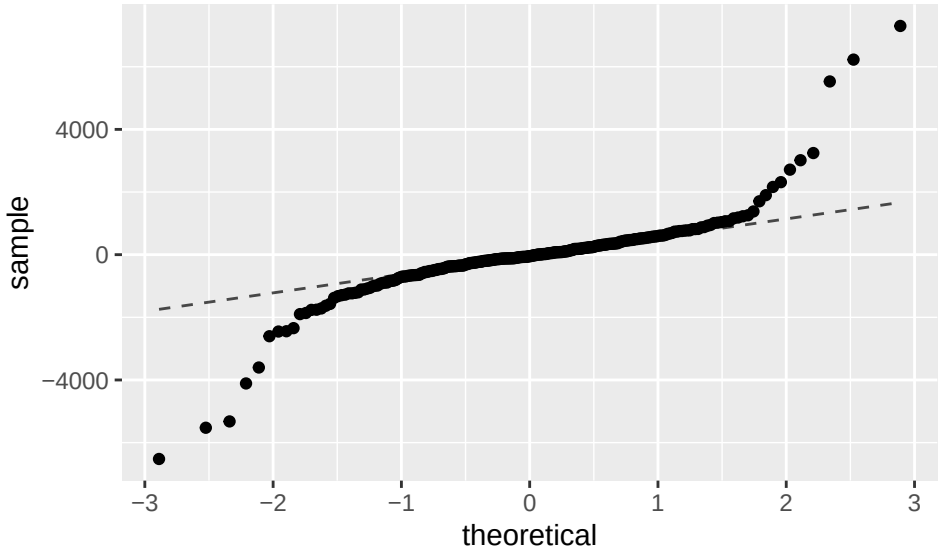
```
## [1] 1606762
```

```
res_lasso <- y_test - lasso_pred
gf_point(res_lasso ~ lasso_pred)
```



```
gf_qq(~ res_lasso) %>% gf_qqline()
```

```
## Warning: The following aesthetics were dropped during statistical transformation: sample
## i This can happen when ggplot fails to infer the correct grouping structure in
##   the data.
## i Did you forget to specify a `group` aesthetic or to convert a numerical
##   variable into a factor?
```



The test MSE for the lasso model on Apps is 1606762. The residual plot shows a similar issue to those seen before.

Finally, we pull out the coefficients to examine.

```
round(predict(lasso_mod, type = "coefficients", s = bestlamlasso)[1:18,], 2)
```

```
##  (Intercept)   PrivateYes       Accept       Enroll    Top10perc    Top25perc
##     -1018.40      -407.39         1.67        -1.35        27.01        -4.75
## F.Undergrad P.Undergrad    Outstate  Room.Board       Books     Personal
##         0.09         0.07        -0.14         0.08         0.00         0.07
##         PhD     Terminal   S.F.Ratio perc.alumni      Expend   Grad.Rate
##      -11.67        -4.90        35.24         3.35         0.19        15.06
```

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:

Multiple questions are asked here. We try to take their respective answers and mold it into a coherent response.

Four models were fit - kitchen sink OLS, stepwise regression (will vary), ridge, and lasso. Test MSEs and coefficients were computed for all models.

The OLS regression has a test MSE of 1634314, while the stepwise regression had a test MSE of 1629782. When we turn to the shrinkage methods, ridge had a test MSE of 1238407, while lasso had a test MSE of 1606762. In terms of MSE, the ridge is clearly the superior method. Lasso and stepwise perform better than

OLS (though all three test MSEs are somewhat similar), but not nearly as well as ridge (test MSE clearly lower). It looks like just employing shrinkage here is better than including selection.

To consider the impact of shrinkage, we compare the coefficients. Since there aren't 100s of predictors, we re-print their coefficients here for easier comparison.

```
ksmodel$coefficients
```

```
##  (Intercept)   PrivateYes       Accept       Enroll    Top10perc    Top25perc
## -1.00189e+03 -4.11869e+02  1.67979e+00 -1.45720e+00  2.85372e+01 -6.09969e+00
##  F.Undergrad  P.Undergrad     Outstate   Room.Board        Books     Personal
##  1.09225e-01  6.82576e-02 -1.47194e-01  8.70886e-02  7.06201e-03  7.61093e-02
##          PhD     Terminal    S.F.Ratio  perc.alumni       Expend    Grad.Rate
## -1.20147e+01 -4.92490e+00  3.60502e+01  4.01236e+00  1.92523e-01  1.55509e+01
```

```
appSteplm$coefficients
```

```
##  (Intercept)   PrivateYes       Accept       Enroll    Top10perc    Top25perc
## -963.5114611 -381.2465998    1.6789545   -1.4589143   29.6426035   -6.8867306
##  F.Undergrad  P.Undergrad     Outstate   Room.Board          PhD    S.F.Ratio
##    0.1110962    0.0740656   -0.1517421    0.0767308  -15.1719803   35.6599888
##  perc.alumni       Expend    Grad.Rate
##    2.7608522    0.1927984   15.6920253
```

```
round(predict(ridge_mod, type = "coefficients", s = bestlamridge)[1:18,], 2)
```

```
## (Intercept)  PrivateYes      Accept      Enroll   Top10perc   Top25perc
##    -1772.17     -542.96        1.05        0.32       15.65        3.84
## F.Undergrad P.Undergrad    Outstate  Room.Board       Books    Personal
##        0.07        0.04       -0.05        0.18        0.08        0.05
##         PhD    Terminal   S.F.Ratio perc.alumni      Expend   Grad.Rate
##       -6.79       -7.01       29.44       -4.75        0.14       14.29
```

```
round(predict(lasso_mod, type = "coefficients", s = bestlamlasso)[1:18,], 2)
```

```
## (Intercept)  PrivateYes      Accept      Enroll   Top10perc   Top25perc
##    -1018.40     -407.39        1.67       -1.35       27.01       -4.75
## F.Undergrad P.Undergrad    Outstate  Room.Board       Books    Personal
##        0.09        0.07       -0.14        0.08        0.00        0.07
##         PhD    Terminal   S.F.Ratio perc.alumni      Expend   Grad.Rate
##      -11.67       -4.90       35.24        3.35        0.19       15.06
```

The stepwise model removed 3 variables overall. Lasso looks to have removed just Books. The coefficients between OLS and stepwise are very similar, even with the three variables removed. Many of the coefficients remain the similar even in the lasso and ridge. The biggest notes I see to make are that some coefficients in the ridge solution have different signs than in the other three solutions. For example, "Enroll" is + in ridge, but negative in the other three. Similar behavior occurs with Top25perc and perc.alumni. This could be related to the large difference in intercept observed (ridge starts about -700 below the rest).

To think about which variables were removed in selection, the ridge coefficients don't help too much, since they are on the original scale. We can think about the kitchen sink OLS model however.

```
msummary(ksmodel)
```

```
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.00e+03   4.97e+02   -2.01    0.045 *
## PrivateYes  -4.12e+02   1.68e+02   -2.45    0.015 *
## Accept       1.68e+00   4.54e-02   36.97  < 2e-16 ***
## Enroll      -1.46e+00   2.26e-01   -6.44  2.8e-10 ***
```

```
## Top10perc    2.85e+01   7.09e+00    4.03  6.6e-05 ***
## Top25perc   -6.10e+00   5.54e+00   -1.10    0.271
## F.Undergrad  1.09e-01   3.88e-02    2.82    0.005 **
## P.Undergrad  6.83e-02   3.90e-02    1.75    0.081 .
## Outstate    -1.47e-01   2.37e-02   -6.21  1.1e-09 ***
## Room.Board   8.71e-02   6.04e-02    1.44    0.150
## Books        7.06e-03   2.69e-01    0.03    0.979
## Personal     7.61e-02   7.47e-02    1.02    0.309
## PhD         -1.20e+01   5.52e+00   -2.18    0.030 *
## Terminal    -4.92e+00   6.07e+00   -0.81    0.418
## S.F.Ratio    3.61e+01   1.70e+01    2.12    0.034 *
## perc.alumni  4.01e+00   4.76e+00    0.84    0.400
## Expend       1.93e-01   1.98e-02    9.71  < 2e-16 ***
## Grad.Rate    1.56e+01   3.68e+00    4.22  2.9e-05 ***
##
## Residual standard error: 1020 on 500 degrees of freedom
## Multiple R-squared:  0.94,   Adjusted R-squared:  0.938
## F-statistic:  460 on 17 and 500 DF,  p-value: <2e-16
```

Books has an extremely large p-value in the kitchen sink model, so it's no surprise it was removed with variable selection. Other terms with high p-values could be removed as well, though we must remember that these values change as the model changes. Personal and Terminal were also removed in stepwise and their p-values are above 0.3, but perc.alumni was kept and it's p-value in (b) was 0.4. So, we see some logical connections to what variables were removed, but we can't predict it 100% based on these p-values.

Additional 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 variable, 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)
```

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. You may also wish to create appropriate x and y matrices for future function inputs at the same time.

SOLUTION:

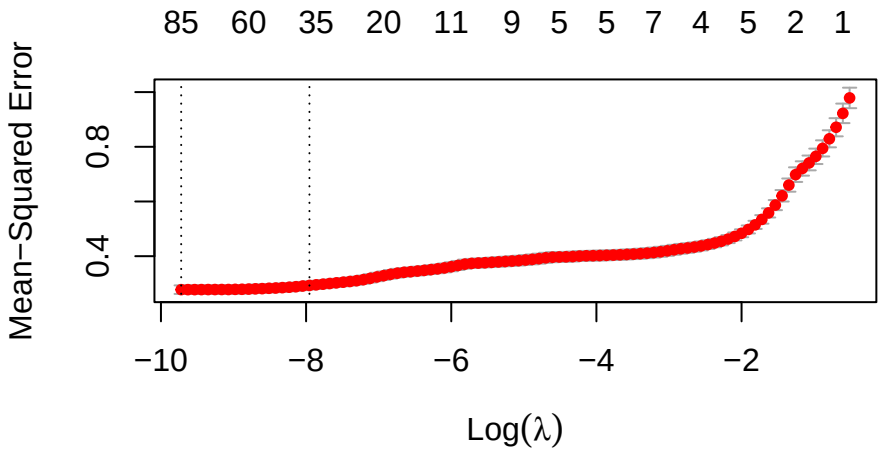
```
set.seed(495)

n <- nrow(newsoil)

# uses same names as above be careful that you don't get confused about which object is which
# better suggestion is to add a 2 or Sand or something here to names to show for second prob
train_index <- sample(1:n, (0.75) * n)
test_index <- setdiff(1:n, train_index)

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

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

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

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

> 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:

Lasso model 1 - Lambda chosen through CV

```
set.seed(495)
cv.out.lasso2 <- cv.glmnet(x_train, y_train, alpha = 1) # Fit lasso regression model on training data
plot(cv.out.lasso2)
```




```
bestlamlasso2 <- cv.out.lasso2$lambda.min  # Select lambda that minimizes training MSE
bestlamlasso2
```

```
## [1] 5.99889e-05
```

The best lambda chosen from CV here is 5.99889 times 10^-5. It is very small.

To get the coefficients and see how many are set to 0, we do some computations after running the model and saving the coefficients.

```
# could use cv.out.lasso2 object instead of refitting
lasso_mod <- glmnet(x_train, y_train, alpha = 1, thresh = 1e-12)
```

```
## Warning: from glmnet C++ code (error code -62); Convergence for 62th lambda
## value not reached after maxit=100000 iterations; solutions for larger lambdas
## returned
```

```
mycoefs <- as.numeric(predict(lasso_mod, type = "coefficients", s = bestlamlasso2))
length(mycoefs) - length(which(abs(mycoefs)>0))
```

```
## [1] 95
```

We can see that 95 of the coefficients were set to 0. Note, that setting this threshold is making the solution different from the default (the default threshold is 1e-7). A solution with that threshold results in many fewer coefficients being set to 0.

Lasso model 2 - Lambda chosen by me

The previous lasso model had an error that it didn't converge for lambdas that were too big along it's grid, so I'm going to try a lambda of 1 to see what happens. It turned out that set all predictors to 0, so I tried 0.5, and only had 1 predictor with a non-zero coefficient, so I went down one more time and picked a final chosen lambda of 0.25. In this final model, 102 of the coefficients were set to 0. Note that none of the previous code needs to be re-run - we just ask for the coefficients at our different lambda value.

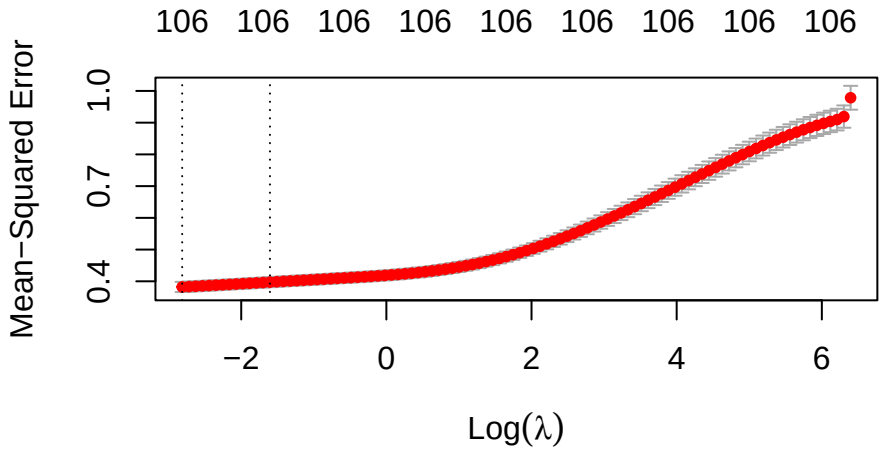
```
mycoefs2 <- as.numeric(predict(lasso_mod, type = "coefficients", s = 0.25))
length(mycoefs2) - length(which(abs(mycoefs2)>0))
```

```
## [1] 102
```

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:

```
set.seed(495)
cv.out.ridge2 <- cv.glmnet(x_train, y_train, alpha = 0) # Fit ridge regression model on training data
plot(cv.out.ridge2)
```




```
bestlamridge2 <- cv.out.ridge2$lambda.min  # Select lambda that minimizes training MSE
bestlamridge2
```

```
## [1] 0.0599889
```

The best lambda chosen from CV here is 0.0599889. It is small, but not as small as the previous one chosen for lasso.

To get the coefficients and see how many are set to 0, we do some computations after running the model and saving the coefficients. Note that because this is ridge, it would be expected that none are set exactly to 0.

```
ridge_mod <- glmnet(x_train, y_train, alpha = 0, thresh = 1e-12)
mycoefsr <- as.numeric(predict(ridge_mod, type = "coefficients", s = bestlamridge2))
length(mycoefsr) - length(which(abs(mycoefsr)>0))
```

```
## [1] 0
```

Indeed, we learn that none of the coefficients were set to 0 in the ridge model.

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

SOLUTION:

Lasso lambda from CV:

```
lasso_pred <- predict(lasso_mod, s = bestlamlasso2, newx = x_test) # Use best lambda to predict test da
mean((lasso_pred - y_test)^2) # Calculate test MSE
```

```
## [1] 0.264417
```

Lasso lambda my choice:

```
lasso_pred2 <- predict(lasso_mod, s = 0.25, newx = x_test) # Use lambda = 0.25 to predict test data
mean((lasso_pred2 - y_test)^2) # Calculate test MSE
```

```
## [1] 0.553946
```

Ridge lambda from CV:

```
ridge_pred <- predict(ridge_mod, s = bestlamridge2, newx = x_test) # Use best lambda to predict test da
mean((ridge_pred - y_test)^2) # Calculate test MSE
```

```
## [1] 0.279
```

Just for curiosity - get OLS MSE:

```
ksmodel <- lm(Sand ~ . , data = train)
ks_pred <- predict(ksmodel, newdata = test)
mean((ks_pred - test$Sand)^2) # Calculate test MSE
```

```
## [1] 0.193121
```

Let's check how much shrinkage we are getting (also, just out of curiosity):

```
# check sums and squares due to different constraints
sum(abs(mycoefs)) # sum of abs lasso coefs from best lambda
```

```
## [1] 29.3674
```

```
sum(abs(mycoefs2)) # sum of abs lasso coefs with lambda = 0.25
```

```
## [1] 3.02988
```

```
sum(abs(mycoefsr)) # sum of abs ridge coefs with best lambda
```

```
## [1] 22.3522
```

```
sum(abs(ksmodel$coefficients)) # sum of abs OLS coefs
```

```
## [1] 135901
```

```
sum(mycoefs^2) #same order just sum of squares
```

```
## [1] 91.9251
```

```
sum(mycoefs2^2)
```

```
## [1] 2.71345
```

```
sum(mycoefsr^2)
```

```
## [1] 7.74469
```

```
sum(ksmodel$coefficients^2)
```

```
## [1] 329399704
```

Clearly, the lasso and ridge are doing a lot of shrinkage. The model with my chosen lambda of 0.25 for lasso is doing the most shrinkage, followed by ridge and then the lasso both with their cross-validated lambdas. It's a lot of shrinkage based on the OLS coefficients.

Note that because coefficients are returned by glmnet on the original scale - these are okay to compare, even if the standardized betas were being included in the optimization - we can get a sense of shrinkage from this.

> 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:

My curiosity above showed me the OLS model actually has the lowest MSE, but it has no coefficients set to 0. With this many predictors, we want some selection here. So, I'll limit my comparison to the 3 models requested.

The lasso with the best lambda (chosen via CV) has the lowest MSE of these 3 models, but the ridge (with lambda chosen similarly) is not far behind (0.26 and 0.28). The lasso with lambda I chose performs more poorly, with a test MSE of 0.55. So, yes, the "best" lambda lasso model is better than the one with the lambda I chose. However, ridge is not better than the best lasso model. From just these three models, I would select the lasso model with lambda chosen by CV. It has the lowest test MSE and 95 predictors have coefficients set to 0, so it is doing considerable variable selection.

> 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 settings for both of these are TRUE. This is so that variable scaling doesn't affect the model solution. Because coefficients are being driven towards 0, if one variable has a large coefficient due to scaling, it could be kept the "largest" while the others are shrunk, and that's not what we want in our solution. Standardizing puts the variables on equal footing.

Coefficients are still returned on the original scale!

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

SOLUTION:

The alpha argument in glmnet is what governs which penalty is being fit. So, changing the alpha would let you fit the various elastic net penalties. alpha = 0 is ridge and alpha = 1 is lasso and anything in between is an elastic net.

Additional 3

After reading through your portfolio reflections, I decided to try to adapt some class activities and assignments to better align with your goals. This is a little bit challenging because they are quite varied. However, the *College* data set that we used for Additional 1 does permit a host of different analyses, so we're going to try using it to help with this.

For this problem, your assignment is to tackle some aspect of your goals for class using the *College* data set. Include your work here, in the outline below. I'll give you feedback and work to correct any statistical issues, and note that while assessing this, I'm not looking for any one specific thing from any of you. I'm including some examples of what you might do below, based on some of the goals I read.

Examples, if you said ...

- I want to demonstrate my understanding of method X. Can you apply method X to this data set? (You may have to do some work like create a variable for example to run an ANOVA; take one of the quantitative variables and cut it into 4 groups.) Try it. Write a summary of your findings.
- I want to work on my EDA. We know the (overall) goal here is to predict Apps. What EDA would you do (we skipped it above!)? Describe it, do some of it, etc.
- I want to practice commenting my code and making visuals extremely clear. Pick a simple analysis (predict Apps with like 2-3 other predictors), make your code awesome and visuals really good.
- I want to practice writing more precisely / shorter paragraphs for my results. Pick a simple analysis (predict Apps with like 2-3 other predictors), and write your results the way you typically would. Then, leave that there, and try a revision, working to improve the writing (this way, you see the original and the revision).

Other guidance:

- Spend at most 2 hours on this question. This is designed to let you practice something you wanted to work on, not take up all your time.
- If none of your goals seem to align with this, you can pick one based on the examples I listed above, or something similar that you want to work on.
- Use your best judgement for any models you need to fit here. If you just need a model to practice writing, it is okay if that's not the overall best model you might pick. On the other hand, if you want to practice model fitting, you should be spending time discussing that and showing your work for it. We will practice the entire data analysis process in future work.

part a: What aspect of your goals for class are you going to tackle for this assignment? How?

SOLUTION:

Answers will vary.

part b: Include your work here!

SOLUTION:

Answers will vary. Work should make sense based on what is stated in part a.