ANUSHA MUPPALLA
16286311

Lab Assignment5

1.In this exercise, we will generate simulated data, and will then use this data to perform best subset selection.

(a) (5 points) Use the rnorm() function to generate a predictor X of length n = 100, as well as a noise vector ε of length n = 100.

```
> set.seed(1)
> X <- rnorm(100)
> noise <- rnorm(100)
```

(b) (5 points) Generate a response vector Y of length n = 100 according to the model $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \varepsilon$,

where β_0, β_1, β_2, and β_3 are constants of your choice.

```
> set.seed(1)
> X <- rnorm(100)
> noise <- rnorm(100)
> Y <- 3 + 1*X + 4*X^2 - 1*X^3 + noise
```

(c) (5 points) Use the regsubsets() function to perform best subset selection in order to choose the best model containing the predictors $X, X^2, \ldots, X^{10}$. What is the best model obtained according to C_p, BIC, and adjusted R^2? Show some plots to provide evidence for your answer, and report the coefficients of the best model obtained. Note you will need to use the data.frame() function to create a single data set containing both X and Y

(d) (5 points) Repeat (c), using forward stepwise selection and also using backwards stepwise selection. How does your answer compare to the results in (c)?

```
> require(caret)
>
> model_back <- train(Y ~ poly(X, 10), data = df,
+                     method = 'glmStepAIC', direction = 'backward',
+                     trace = 0,
+                     trControl = trainControl(method = 'none', verboseI
ter = FALSE))
>
> postResample(predict(model_back, df), df$Y)
     RMSE  Rsquared       MAE
0.9314956 0.9569843 0.7488821
> 
```

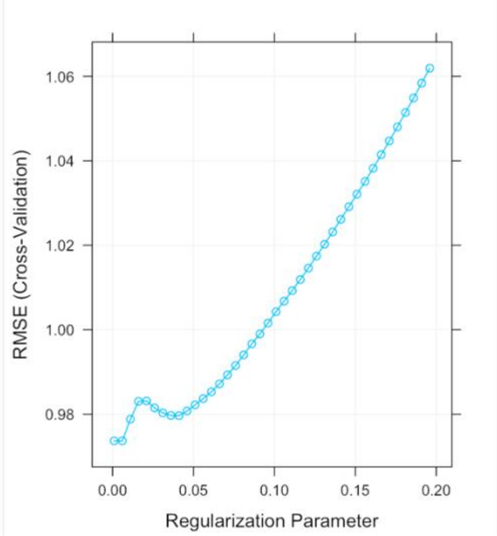
```
> summary(model_back$finalModel)

Call:
NULL

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.8914  -0.5860  -0.1516   0.5892   2.1794

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)      6.10265    0.09557  63.856  < 2e-16 ***
`poly(X, 10)1`  -7.19295    0.95569  -7.526 2.96e-11 ***
`poly(X, 10)2`  40.74405    0.95569  42.633  < 2e-16 ***
`poly(X, 10)3` -14.70908    0.95569 -15.391  < 2e-16 ***
`poly(X, 10)5`   1.48019    0.95569   1.549    0.125
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for gaussian family taken to be 0.9133516)

    Null deviance: 2017.132  on 99  degrees of freedom
Residual deviance:   86.768  on 95  degrees of freedom
AIC: 281.59

Number of Fisher Scoring iterations: 2

>
```

```
> x_poly <- poly(df$X, 10)
>
> colnames(x_poly) <- paste0('poly', 1:10)
> model_forw <- train(y = Y, x = x_poly,
+                     method = 'glmStepAIC', direction = 'forward',
+                     trace = 0,
+                     trControl = trainControl(method = 'none', verboseI
ter = FALSE))
>
> postResample(predict(model_forw, data.frame(x_poly)), df$Y)
     RMSE  Rsquared       MAE
0.9314956 0.9569843 0.7488821
>
```

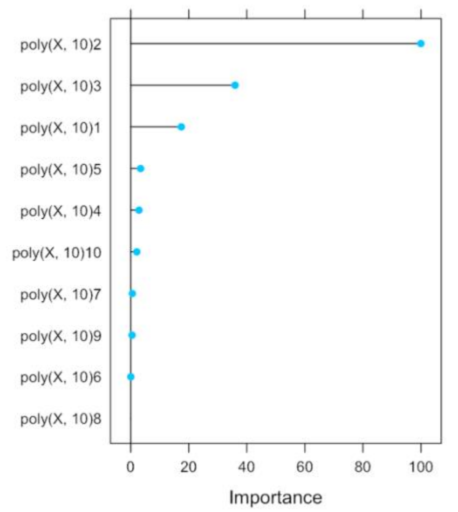
```
> summary(model_forw$finalModel)

Call:
NULL

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.8914  -0.5860  -0.1516   0.5892   2.1794

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)   6.10265    0.09557  63.856  < 2e-16 ***
poly2        40.74405    0.95569  42.633  < 2e-16 ***
poly3       -14.70908    0.95569 -15.391  < 2e-16 ***
poly1        -7.19295    0.95569  -7.526 2.96e-11 ***
poly5         1.48019    0.95569   1.549    0.125
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for gaussian family taken to be 0.9133516)

    Null deviance: 2017.132  on 99  degrees of freedom
Residual deviance:   86.768  on 95  degrees of freedom
AIC: 281.59

Number of Fisher Scoring iterations: 2
```

(e) (5 points) Now fit a lasso model to the simulated data, again using X, X^2, . . . , X^{10} as predictors. Use cross-validation to select the optimal value of λ. Create plots of the cross-validation error as a function of λ. Report the resulting coefficient estimates, and discuss the results obtained.

```
Enter an item from the menu, or 0 to exit
Selection:
Enter an item from the menu, or 0 to exit
Selection:
Enter an item from the menu, or 0 to exit
Selection:
Enter an item from the menu, or 0 to exit
Selection: 1
trying URL 'https://cran.rstudio.com/bin/macosx/el-capitan/contrib/3.6/glmnet_
2.0-18.tgz'
Content type 'application/x-gzip' length 1565374 bytes (1.5 MB)
==================================================
downloaded 1.5 MB


The downloaded binary packages are in
	/var/folders/w5/msff5yg55hvgnrf5gxk_qk680000gn/T//RtmpcWJjAZ/downloaded
_packages
> lasso_model <- train(Y ~ poly(X, 10), data = df,
+                      method = 'glmnet',
+                      trControl = trainControl(method = 'cv', number = 10),
+                      tuneGrid = expand.grid(alpha = 1,
+                                             lambda = seq(0.001, 0.2, by = 0.0
05)))
>
> plot(lasso_model)
>
```



```
Selection:
Enter an item from the menu, or 0 to exit
Selection:
Enter an item from the menu, or 0 to exit
Selection:
Enter an item from the menu, or 0 to exit
Selection: 1
trying URL 'https://cran.rstudio.com/bin/macosx/el-capitan/contrib/3.6/glmnet_
2.0-18.tgz'
Content type 'application/x-gzip' length 1565374 bytes (1.5 MB)
==================================================
downloaded 1.5 MB


The downloaded binary packages are in
	/var/folders/w5/msff5yg55hvgnrf5gxk_qk680000gn/T//RtmpcWJjAZ/downloaded
_packages
> lasso_model <- train(Y ~ poly(X, 10), data = df,
+                      method = 'glmnet',
+                      trControl = trainControl(method = 'cv', number = 10),
+                      tuneGrid = expand.grid(alpha = 1,
+                                             lambda = seq(0.001, 0.2, by = 0.0
05)))
>
> plot(lasso_model)
> plot(varImp(lasso_model))
>
```



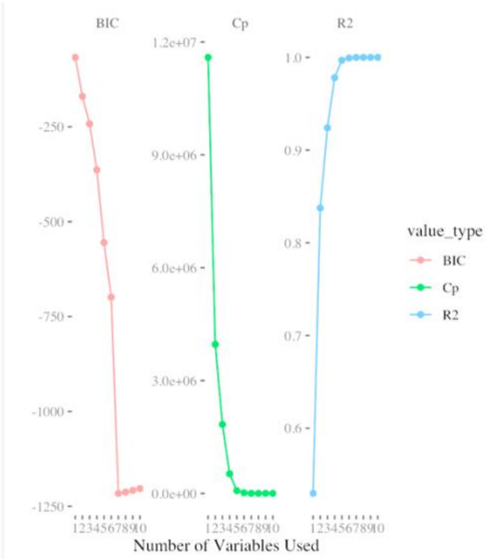

```
> coef(lasso_model$finalModel, lasso_model$bestTune$lambda)
11 x 1 sparse Matrix of class "dgCMatrix"
                        1
(Intercept)    6.10264724
poly(X, 10)1  -7.09661230
poly(X, 10)2  40.64770776
poly(X, 10)3 -14.61274414
poly(X, 10)4   1.16075614
poly(X, 10)5   1.38384950
poly(X, 10)6   0.02266061
poly(X, 10)7  -0.23342691
poly(X, 10)8  -0.01160772
poly(X, 10)9  -0.19950210
poly(X, 10)10 -0.85489064
>
```

```
> postResample(predict(lasso_model, df), df$Y)
     RMSE  Rsquared       MAE
0.9173549 0.9582909 0.7336870
>
```

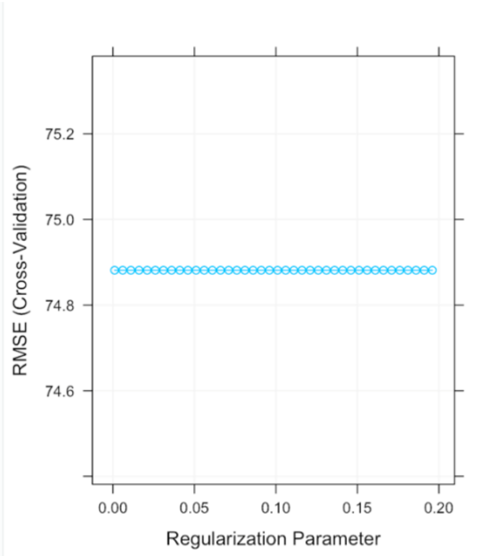
(f) (5 points)Now generate a response vector Y according to the model $Y = \beta_0 + \beta_7 X^7 + \varepsilon$,

and perform best subset selection and the lasso. Discuss the results obtained.

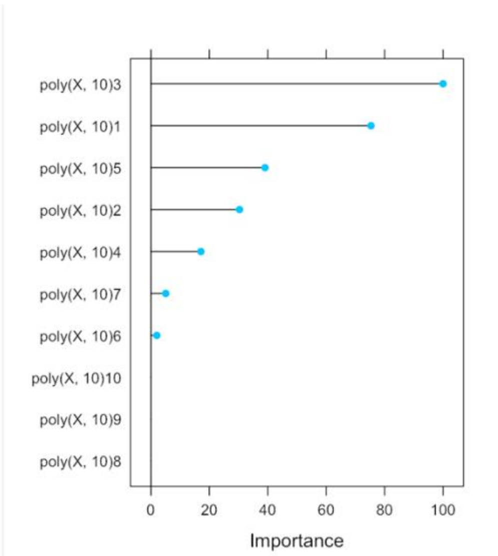
```
poly(X, 10)5    1.38384950
poly(X, 10)6    0.02266061
poly(X, 10)7   -0.23342691
poly(X, 10)8   -0.01160772
poly(X, 10)9   -0.19950210
poly(X, 10)10  -0.85489064
> postResample(predict(lasso_model, df), df$Y)
     RMSE  Rsquared       MAE
0.9173549 0.9582909 0.7336870
> Y_7 <- 3 + 8*X^7 + noise
> df_2 <- data_frame(Y_7 = Y_7, X = df[,-1])
>
> fit <- regsubsets(Y_7 ~ poly(X, 10), data = df_2, nvmax = 10)
>
> fit_summary <- summary(fit)
>
> data_frame(Cp = fit_summary$cp,
+            BIC = fit_summary$bic,
+            R2 = fit_summary$adjr2) %>%
+     mutate(id = row_number()) %>%
+     gather(value_type, value, -id) %>%
+     ggplot(aes(id, value, col = value_type)) +
+     geom_line() + geom_point() + ylab('') + xlab('Number of Variables Used')
 +
+     facet_wrap(~ value_type, scales = 'free') +
+     theme_tufte() + scale_x_continuous(breaks = 1:10)
>
```




```
0.9173549 0.9582909 0.7336870
> Y_7 <- 3 + 8*X^7 + noise
> df_2 <- data_frame(Y_7 = Y_7, X = df[,-1])
>
> fit <- regsubsets(Y_7 ~ poly(X, 10), data = df_2, nvmax = 10)
>
> fit_summary <- summary(fit)
>
> data_frame(Cp = fit_summary$cp,
+            BIC = fit_summary$bic,
+            R2 = fit_summary$adjr2) %>%
+     mutate(id = row_number()) %>%
+     gather(value_type, value, -id) %>%
+     ggplot(aes(id, value, col = value_type)) +
+     geom_line() + geom_point() + ylab('') + xlab('Number of Variables Used')
 +
+     facet_wrap(~ value_type, scales = 'free') +
+     theme_tufte() + scale_x_continuous(breaks = 1:10)
> lasso_y7_model <- train(Y_7 ~ poly(X, 10), data = df_2,
+                         method = 'glmnet',
+                         trControl = trainControl(method = 'cv', number = 5),
+                         tuneGrid = expand.grid(alpha = 1,
+                                                lambda = seq(0.001, 0.2, by =
 0.005)))
>
> plot(lasso_y7_model)
>
```




```
> Y_7 <- 3 + 8*X^7 + noise
> df_2 <- data_frame(Y_7 = Y_7, X = df[,-1])
>
> fit <- regsubsets(Y_7 ~ poly(X, 10), data = df_2, nvmax = 10)
>
> fit_summary <- summary(fit)
>
> data_frame(Cp = fit_summary$cp,
+            BIC = fit_summary$bic,
+            R2 = fit_summary$adjr2) %>%
+     mutate(id = row_number()) %>%
+     gather(value_type, value, -id) %>%
+     ggplot(aes(id, value, col = value_type)) +
+     geom_line() + geom_point() + ylab('') + xlab('Number of Variables Used')
 +
+     facet_wrap(~ value_type, scales = 'free') +
+     theme_tufte() + scale_x_continuous(breaks = 1:10)
> lasso_y7_model <- train(Y_7 ~ poly(X, 10), data = df_2,
+                         method = 'glmnet',
+                         trControl = trainControl(method = 'cv', number = 5),
+                         tuneGrid = expand.grid(alpha = 1,
+                                                lambda = seq(0.001, 0.2, by =
 0.005)))
>
> plot(lasso_y7_model)
> plot(varImp(lasso_y7_model))
>
```



```
> coef(lasso_y7_model$finalModel, lasso_y7_model$bestTune$lambda)
11 x 1 sparse Matrix of class "dgCMatrix"
                       1
(Intercept)     36.72505
poly(X, 10)1  2630.24239
poly(X, 10)2  1059.42846
poly(X, 10)3  3491.14380
poly(X, 10)4   597.14287
poly(X, 10)5  1363.96802
poly(X, 10)6    70.93052
poly(X, 10)7   177.79560
poly(X, 10)8     .
poly(X, 10)9     .
poly(X, 10)10    .
> postResample(predict(lasso_y7_model, df_2), df_2$Y_7)
      RMSE   Rsquared        MAE
14.2854376  0.9996164  4.9531386
> |
```

2.(35 points total) In this exercise, we will predict the number of applications received using the other variables in the College data set.

(a) (5 points) Split the data set into a training set and a test set.

```
> require(ISLR)
> require(caret)
> require(tidyverse)
> data('College')
> set.seed(1)
>
> inTrain <- createDataPartition(College$Apps, p = 0.75, list = FALSE)
>
> training <- College[inTrain,]
> testing <- College[-inTrain,]
>
> preObj <- preProcess(training, method = c('center', 'scale'))
>
> training <- predict(preObj, training)
> testing <- predict(preObj, testing)
>
> y_train <- training$Apps
> y_test <- testing$Apps
>
> one_hot_encoding <- dummyVars(Apps ~ ., data = training)
> x_train <- predict(one_hot_encoding, training)
> x_test <- predict(one_hot_encoding, testing)
```

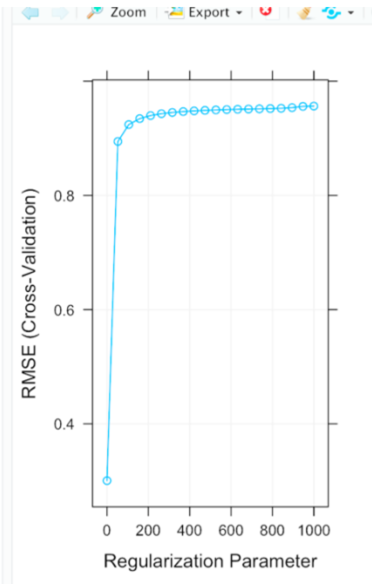
(b) (5 points) Fit a linear model using least squares on the training set, and report the test error obtained.

```
> lin_model <- lm(Apps ~ ., data = training)
>
> pred <- predict(lin_model, testing)
>
> (lin_info <- postResample(pred, testing$Apps))
     RMSE  Rsquared       MAE
0.2799768 0.9201765 0.1568743
```

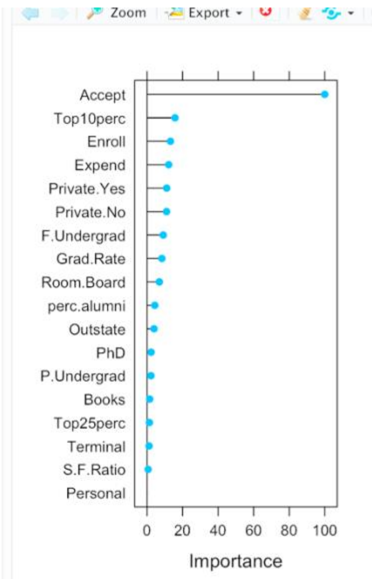
(c) (5 points) Fit a ridge regression model on the training set, with λ chosen by cross-validation. Report the test error obtained.

```
> ridge_fit <- train(x = x_train, y = y_train,
+                    method = 'glmnet',
+                    trControl = trainControl(method = 'cv', number = 10),
+                    tuneGrid = expand.grid(alpha = 0,
+                                           lambda = seq(0, 10e2, length.out =20)))
Warning message:
In nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo,  :
  There were missing values in resampled performance measures.
>
> (ridge_info <- postResample(predict(ridge_fit, x_test), y_test))
     RMSE  Rsquared       MAE
0.2853247 0.9211286 0.1645806
> coef(ridge_fit$finalModel, ridge_fit$bestTune$lambda)
19 x 1 sparse Matrix of class "dgCMatrix"
                       1
(Intercept)  0.034871314
Private.No   0.075423210
Private.Yes -0.076037580
Accept       0.665628733
Enroll       0.090243372
Top10perc    0.107160248
Top25perc    0.011628030
F.Undergrad  0.063308801
P.Undergrad  0.017427317
Outstate    -0.028995432
Room.Board   0.048720533
Books        0.012799145
Personal    -0.002894430
PhD         -0.017989250
```

```
> (ridge_info <- postResample(predict(ridge_fit, x_test), y_test))
     RMSE  Rsquared       MAE 
0.2853247 0.9211286 0.1645806 
> coef(ridge_fit$finalModel, ridge_fit$bestTune$lambda)
19 x 1 sparse Matrix of class "dgCMatrix"
                       1
(Intercept)  0.034871314
Private.No   0.075423210
Private.Yes -0.076037580
Accept       0.665628733
Enroll       0.090243372
Top10perc    0.107160248
Top25perc    0.011628030
F.Undergrad  0.063308801
P.Undergrad  0.017427317
Outstate    -0.028995432
Room.Board   0.048720533
Books        0.012799145
Personal    -0.002894430
PhD         -0.017989250
Terminal    -0.010434665
S.F.Ratio    0.006920126
perc.alumni -0.031683867
Expend       0.083525070
Grad.Rate    0.058131023
> plot(ridge_fit)
> 
> |
```




```
     RMSE  Rsquared       MAE 
0.2853247 0.9211286 0.1645806 
> coef(ridge_fit$finalModel, ridge_fit$bestTune$lambda)
19 x 1 sparse Matrix of class "dgCMatrix"
                       1
(Intercept)  0.034871314
Private.No   0.075423210
Private.Yes -0.076037580
Accept       0.665628733
Enroll       0.090243372
Top10perc    0.107160248
Top25perc    0.011628030
F.Undergrad  0.063308801
P.Undergrad  0.017427317
Outstate    -0.028995432
Room.Board   0.048720533
Books        0.012799145
Personal    -0.002894430
PhD         -0.017989250
Terminal    -0.010434665
S.F.Ratio    0.006920126
perc.alumni -0.031683867
Expend       0.083525070
Grad.Rate    0.058131023
> plot(ridge_fit)
> 
> plot(varImp(ridge_fit))
> 
```

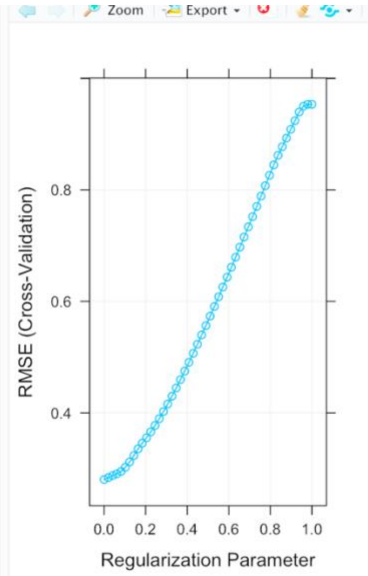


(d) (5 points) Fit a lasso model on the training set, with λ chosen by crossvalidation. Report the test error obtained, along with the number of non-zero coefficient estimates.

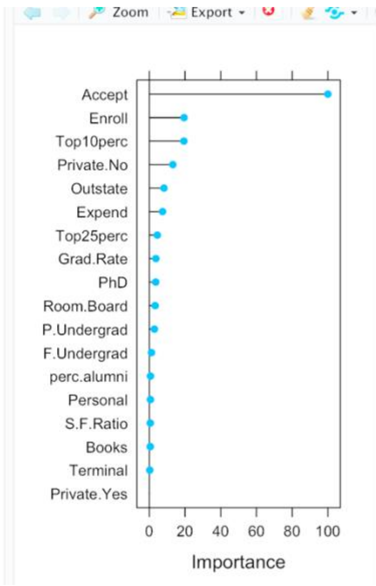
```
>
> plot(varImp(ridge_fit))
> lasso_fit <- train(x = x_train, y = y_train,
+                    method = 'glmnet',
+                    trControl = trainControl(method = 'cv', number = 10),
+                    tuneGrid = expand.grid(alpha = 1,
+                                           lambda = seq(0.0001, 1, length.out = 50)))
Warning message:
In nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo,  :
  There were missing values in resampled performance measures.
>
> (lasso_info <- postResample(predict(lasso_fit, x_test), y_test))
     RMSE  Rsquared       MAE
0.2802352 0.9201823 0.1561301
> coef(lasso_fit$finalModel, lasso_fit$bestTune$lambda)
19 x 1 sparse Matrix of class "dgCMatrix"
                       1
(Intercept) -0.037243095
Private.No   0.137026483
Private.Yes  .
Accept       1.041851224
Enroll      -0.202744295
Top10perc    0.201576111
Top25perc   -0.046294002
F.Undergrad  0.012507811
P.Undergrad  0.029491934
Outstate    -0.085333127
Room.Board   0.033826427
Books        0.005116779
```


```
  There were missing values in resampled performance measures.
>
> (lasso_info <- postResample(predict(lasso_fit, x_test), y_test))
     RMSE  Rsquared       MAE
0.2802352 0.9201823 0.1561301
> coef(lasso_fit$finalModel, lasso_fit$bestTune$lambda)
19 x 1 sparse Matrix of class "dgCMatrix"
                       1
(Intercept) -0.037243095
Private.No   0.137026483
Private.Yes  .
Accept       1.041851224
Enroll      -0.202744295
Top10perc    0.201576111
Top25perc   -0.046294002
F.Undergrad  0.012507811
P.Undergrad  0.029491934
Outstate    -0.085333127
Room.Board   0.033826427
Books        0.005116779
```


```
Personal     0.006295093
PhD         -0.037015361
Terminal    -0.002461461
S.F.Ratio    0.005385825
perc.alumni -0.006575661
Expend       0.077037030
Grad.Rate    0.037985756
> plot(lasso_fit)
>
```



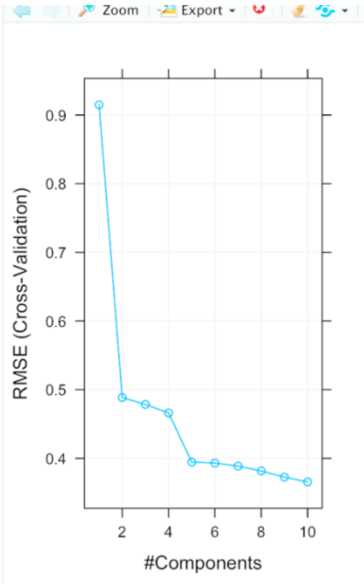
```
> (lasso_info <- postResample(predict(lasso_fit, x_test), y_test))
     RMSE  Rsquared       MAE
0.2802352 0.9201823 0.1561301
> coef(lasso_fit$finalModel, lasso_fit$bestTune$lambda)
19 x 1 sparse Matrix of class "dgCMatrix"
                        1
(Intercept) -0.037243095
Private.No   0.137026483
Private.Yes  .
Accept       1.041851224
Enroll      -0.202744295
Top10perc    0.201576111
Top25perc   -0.046294002
F.Undergrad  0.012507811
P.Undergrad  0.029491934
Outstate    -0.085333127
Room.Board   0.033826427
Books        0.005116779
Personal     0.006295093
PhD         -0.037015361
Terminal    -0.002461461
S.F.Ratio    0.005385825
perc.alumni -0.006575661
Expend       0.077037030
Grad.Rate    0.037985756
> plot(lasso_fit)
> plot(varImp(lasso_fit))
>
```



(e) (5 points) Fit a PCR model on the training set, with M chosen by crossvalidation. Report the test error obtained, along with the value of M selected by cross- validation.

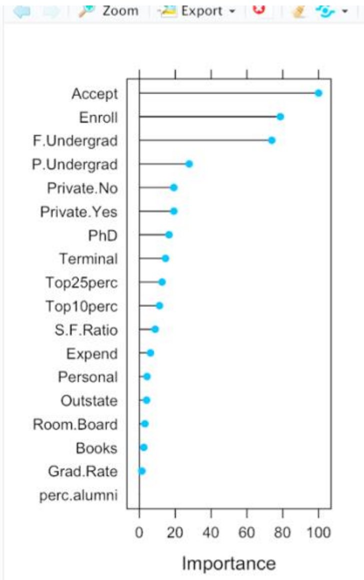
```
> pcr_model <- train(x = x_train, y = y_train,
+                    method = 'pcr',
+                    trControl = trainControl(method = 'cv', number = 10),
+                    tuneGrid = expand.grid(ncomp = 1:10))
> (pcr_info <- postResample(predict(pcr_model, x_test), y_test))
     RMSE  Rsquared       MAE
0.3231292 0.8916531 0.1986075
> coef(pcr_model$finalModel)
, , 10 comps

                 .outcome
Private.No    0.031985972
Private.Yes  -0.031985972
Accept        0.343576750
Enroll        0.305359773
Top10perc     0.042630417
Top25perc     0.027790893
F.Undergrad   0.273818439
P.Undergrad  -0.049487667
Outstate      0.038573119
Room.Board    0.070607615
Books         0.016433593
Personal     -0.023529455
PhD          -0.023992433
Terminal     -0.024182230
S.F.Ratio     0.003741623
perc.alumni  -0.070567887
Expend        0.090126298
Grad.Rate     0.071302714
```

```
> (pcr_info <- postResample(predict(pcr_model, x_test), y_test))
     RMSE  Rsquared       MAE 
0.3231292 0.8916531 0.1986075 
> coef(pcr_model$finalModel)
, , 10 comps

                .outcome
Private.No   0.031985972
Private.Yes -0.031985972
Accept       0.343576750
Enroll       0.305359773
Top10perc    0.042630417
Top25perc    0.027790893
F.Undergrad  0.273818439
P.Undergrad -0.049487667
Outstate     0.038573119
Room.Board   0.070607615
Books        0.016433593
Personal    -0.023529455
PhD         -0.023992433
Terminal    -0.024182230
S.F.Ratio    0.003741623
perc.alumni -0.070567887
Expend       0.090126298
Grad.Rate    0.071302714

> plot(pcr_model)
> 
```




```
     RMSE  Rsquared       MAE 
0.3231292 0.8916531 0.1986075 
> coef(pcr_model$finalModel)
, , 10 comps

                .outcome
Private.No   0.031985972
Private.Yes -0.031985972
Accept       0.343576750
Enroll       0.305359773
Top10perc    0.042630417
Top25perc    0.027790893
F.Undergrad  0.273818439
P.Undergrad -0.049487667
Outstate     0.038573119
Room.Board   0.070607615
Books        0.016433593
Personal    -0.023529455
PhD         -0.023992433
Terminal    -0.024182230
S.F.Ratio    0.003741623
perc.alumni -0.070567887
Expend       0.090126298
Grad.Rate    0.071302714

> plot(pcr_model)
```

```
> plot(varImp(pcr_model))
> 
```

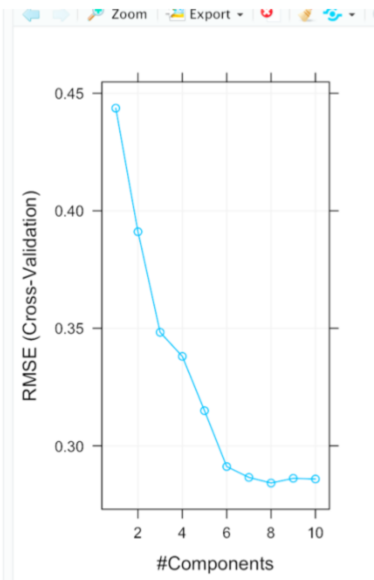


(f) (5 points) Fit a PLS model on the training set, with M chosen by crossvalidation. Report the test error obtained, along with the value of M selected by cross- validation.

```
> pls_model <- train(x = x_train, y = y_train,
+                    method = 'pls',
+                    trControl = trainControl(method = 'cv', number = 10),
+                    tuneGrid = expand.grid(ncomp = 1:10))
> (pls_info <- postResample(predict(pls_model, x_test), y_test))
     RMSE  Rsquared       MAE 
0.2838297 0.9185383 0.1589992 
> coef(pls_model$finalModel)
, , 8 comps

                .outcome
Private.No   0.071464730
Private.Yes -0.071464730
Accept       1.034690648
Enroll      -0.123546500
Top10perc    0.213894280
Top25perc   -0.058237828
F.Undergrad -0.062708027
P.Undergrad  0.032841252
Outstate    -0.091066817
Room.Board   0.028320810
Books        0.009007362
Personal     0.006781888
PhD         -0.038723144
Terminal     0.005282905
S.F.Ratio   -0.004984041
perc.alumni -0.005719117
Expend       0.066720575
Grad.Rate    0.042981237
```


```
> (pls_info <- postResample(predict(pls_model, x_test), y_test))
     RMSE  Rsquared       MAE 
0.2838297 0.9185383 0.1589992 
> coef(pls_model$finalModel)
, , 8 comps

                .outcome
Private.No   0.071464730
Private.Yes -0.071464730
Accept       1.034690648
Enroll      -0.123546500
Top10perc    0.213894280
Top25perc   -0.058237828
F.Undergrad -0.062708027
P.Undergrad  0.032841252
Outstate    -0.091066817
Room.Board   0.028320810
Books        0.009007362
Personal     0.006781888
PhD         -0.038723144
Terminal     0.005282905
S.F.Ratio   -0.004984041
perc.alumni -0.005719117
Expend       0.066720575
Grad.Rate    0.042981237
```

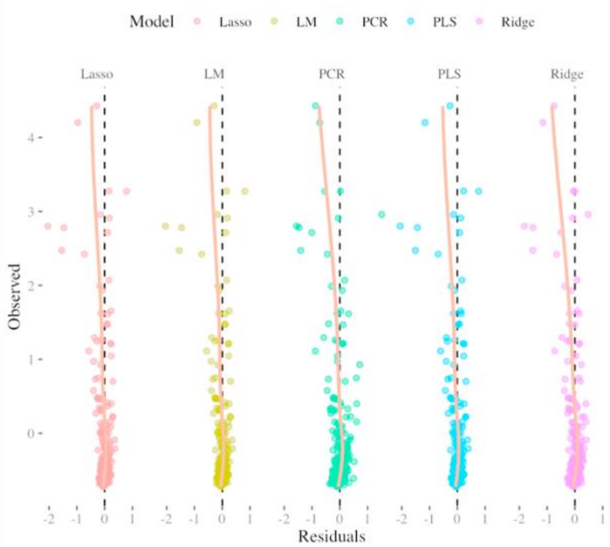

```
> plot(pls_model)
>
```



(g) (5 points) Comment on the results obtained. How accurately can we predict the number of college applications received? Is there much difference among the test errors resulting from these five approaches?

```
> as_data_frame(rbind(lin_info,
+                     ridge_info,
+                     lasso_info,
+                     pcr_info,
+                     pls_info)) %>%
+     mutate(model = c('Linear', 'Ridge', 'Lasso', 'PCR', 'PLS')) %>%
+     select(model, RMSE, Rsquared)
# A tibble: 5 x 3
  model   RMSE Rsquared
  <chr>  <dbl>    <dbl>
1 Linear 0.280    0.920
2 Ridge  0.285    0.921
3 Lasso  0.280    0.920
4 PCR    0.323    0.892
5 PLS    0.284    0.919
Warning message:
`as_data_frame()` is deprecated, use `as_tibble()` (but mind the new semantics).
This warning is displayed once per session.
> testing %>%
+     summarize(sd = sd(Apps))
       sd
1 0.9818241
```

```
+     theme(legend.position = 'top') +
+     coord_flip()
Warning message:
`data_frame()` is deprecated, use `tibble()`.
This warning is displayed once per session.
> require(ggthemes)
>
> residfunc <- function(fit, data) {
+     predict(fit, data) - testing$Apps
+ }
>
> data_frame(Observed = testing$Apps,
+            LM = residfunc(lin_model, testing),
+            Ridge = residfunc(ridge_fit, x_test),
+            Lasso = residfunc(lasso_fit, x_test),
+            PCR = residfunc(pcr_model, x_test),
+            PLS = residfunc(pls_model, x_test)) %>%
+     gather(Model, Residuals, -Observed) %>%
+     ggplot(aes(Observed, Residuals, col = Model)) +
+     geom_hline(yintercept = 0, lty = 2) +
+     geom_point(alpha = 0.6) +
+     geom_smooth(method = 'loess', alpha = 0.01, col = 'lights
almon2') +
+     facet_wrap(~ Model, ncol = 5) +
+     theme_tufte() +
+     theme(legend.position = 'top') +
+     coord_flip()
>
```



3.(35 points total) We have seen that as the number of features used in a model increases, the training error will necessarily decrease, but the test error may not. We will now explore this in a simulated data set.

(a) (5 points) Generate a data set with p = 20 features, n = 1, 000 observations, and an associated quantitative response vector generated according to the model $Y = X\beta + \varepsilon$,
where β has some elements that are exactly equal to zero.

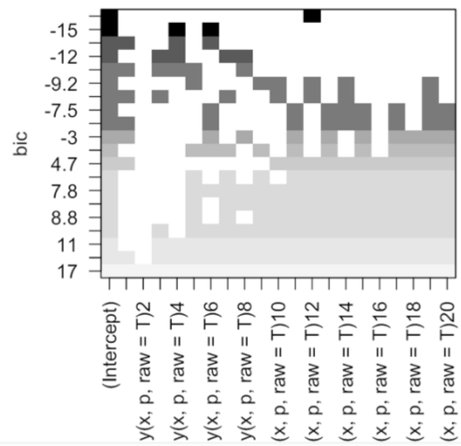
```
Console ~/
>
> X = matrix(rnorm(p * n), ncol = p, nrow = n)
>
>
> beta               = rnorm(p, sd = 10)
> num_rand_zeroes    = sample(0:p/3)
> rand_zeroes        = sample(seq(1, length(beta)), num_rand_zeroes, replac
e = F)
> beta[rand_zeroes] = 0
>
>
> e = rnorm(n)
> Y = as.vector(X * beta + e)
```

(b) (5 points) Split your data set into a training set containing 100 observations and a test set containing 900 observations.

```
> n_training_observations = 100
> train    = sample(1:nrow(X), n_training_observations)
> test     = (-train)
> X.train = X[train]
> Y.train = Y[train]
> X.test  = X[test ]
> Y.test  = Y[test ]
>
>
> df.train = data.frame(y = Y.train, x = X.train)
> df.test  = data.frame(y = Y.test,  x = X.test)
```

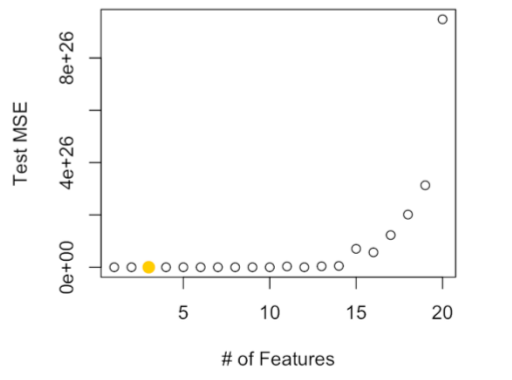
(c) (5 points) Perform best subset selection on the training set, and plot the training set MSE associated with the best model of each size.

```
> beta               = rnorm(p, sd = 10)
> num_rand_zeroes    = sample(0:p/3)
> rand_zeroes        = sample(seq(1, length(beta)), num_rand_zeroes, replac
e = F)
> beta[rand_zeroes] = 0
>
>
> e = rnorm(n)
> Y = as.vector(X * beta + e)
> n_training_observations = 100
> train    = sample(1:nrow(X), n_training_observations)
> test     = (-train)
> X.train = X[train]
> Y.train = Y[train]
> X.test  = X[test ]
> Y.test  = Y[test ]
>
>
> df.train = data.frame(y = Y.train, x = X.train)
> df.test  = data.frame(y = Y.test,  x = X.test)
> library(leaps)
> fit      = regsubsets(y ~ poly(x, p, raw = T), data = df.train, nvmax =
 p)
> plot(fit)
>
```



(d) (5 points) Plot the test set MSE associated with the best model of each size.

```
> library(leaps)
> fit      = regsubsets(y ~ poly(x, p, raw = T), data = df.train, nvmax =
 p)
> plot(fit)
> mse = function(prediction, real) {
+     mean((prediction - real)^2)
+ }
> predict_regsubsets = function(obj, newdata, id) {
+     form   = as.formula(obj$call[[2]]) # Extract formula.
+     matrix = model.matrix(form, newdata)
+     coefic = coef(obj, id = id)
+     xvars  = names(coefic)
+     matrix[, xvars] * coefic
+ }
>
>
> test.mse = sapply(1:p, function(id) {
+     prediction = predict_regsubsets(fit, df.test, id)
+     mse(prediction, Y.test)
+ })
>
>
> plot(seq(1:p), test.mse, xlab = '# of Features', ylab = 'Test MSE')
> points(which.min(test.mse), test.mse[which.min(test.mse)], col = 'orang
e', cex = 2, pch = 20)
> |
```




```
>
> plot(seq(1:p), test.mse, xlab = '# of Features', ylab = 'Test MSE')
> points(which.min(test.mse), test.mse[which.min(test.mse)], col = 'orang
e', cex = 2, pch = 20)
> coef(fit, id = which.min(test.mse))
         (Intercept) poly(x, p, raw = T)1 poly(x, p, raw = T)4 
           0.3229499           -1.2818614            2.7091984 
poly(x, p, raw = T)6 
          -0.8543577 
>
```

(e) (5 points) For which model size does the test set MSE take on its minimum value? Comment on your results. If it takes on its minimum value for a model containing only an intercept or a model containing all of the features, then play around with the way that you are generating the data in (a) until you come up with a scenario in which the test set MSE is minimized for an intermediate model size.

The test MSE is low and approximately constant from 0-11 features. After that, it shoots up. This is expected – since we set beta to 0 for some of the features, it's better to simply throw those out of our model since they don't provide any information.

(f) (5 points) How does the model at which the test set MSE is minimized compare to the true model used to generate the data? Comment on the coefficient values. √Σp ˆr 2

The test MSE is low until we begin to include the features whose beta values are `0`. This makes sense and matches the reality of our model.

(g) (5 points) Create a plot displaying $_{j=1}(\beta j - \beta_j)$ for a range of values of r, where $\hat{\beta}_j^r$ is the jth coefficient estimate for the best model containing r coefficients. Comment on what you observe. How does this compare to the test MSE plot from (d)?

```
> plot(seq(1:p), test.mse, xlab = '# of Features', ylab = 'Test MSE')
> points(which.min(test.mse), test.mse[which.min(test.mse)], col = 'orang
e', cex = 2, pch = 20)
> coef(fit, id = which.min(test.mse))
         (Intercept) poly(x, p, raw = T)1 poly(x, p, raw = T)4
           0.3229499           -1.2818614            2.7091984
poly(x, p, raw = T)6
          -0.8543577
> rsqdiffs = sapply(1:p, function(r) {
+      coefics     = coef(fit, id = r)
+      coef_names = names(coefics)
+      beta.est    = sapply(1:p, function(i) {
+          id = sprintf('Feature #%d', i)
+          if (id %in% coef_names) {
+              return(coefics[id])
+          } else return(0)
+      })
+      return(sqrt( sum((beta - beta.est)^2)) )
+ })
> plot(seq(1:p), rsqdiffs, xlab = '# of Features', ylab = 'Root Squared Di
fference of Betas')
> points(which.min(rsqdiffs), rsqdiffs[which.min(rsqdiffs)], col = 'orang
e', cex = 2, pch = p)
> |
```