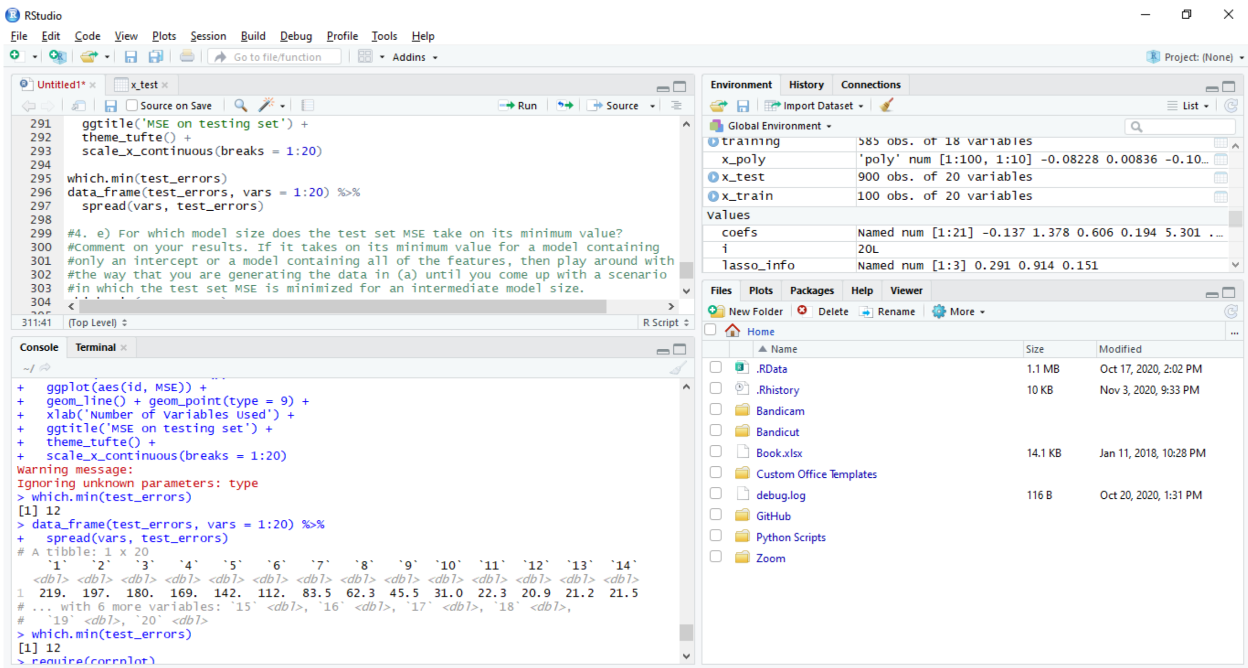
Introduction to Statistical Learning Lab5

Name: Sandeep Reddy Salkuti

Student id: 16296868

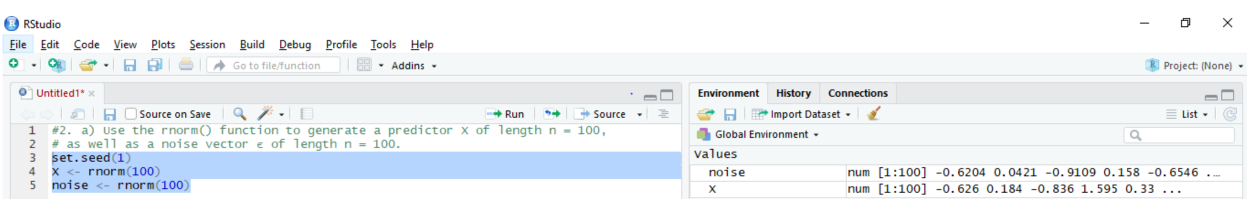
Email: sswf7@umsystem.edu

1. **You may download the R Code for Labs and the Data Sets to use from the textbook website.**



2. (30 points total) 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.



(b) (5 points) Generate a response vector Y of length n = 100 according to the model Y = β0 + β1X + β2X 2 + β3X 3 + ϵ, where β0, β1, β2, and β3 are constants of your choice.

Console | Terminal

```
> #2. b)Generate a response vector Y of length n = 100 according to the model
> #Y = β0 + β1X + β2X2 + β3X3 + ϵ, where β0, β1, β2, and β3 are constants of your choice.
> Y <- 3 + 1*X + 4*X^2 - 1*X^3 + noise
>
```

Environment | History | Connections

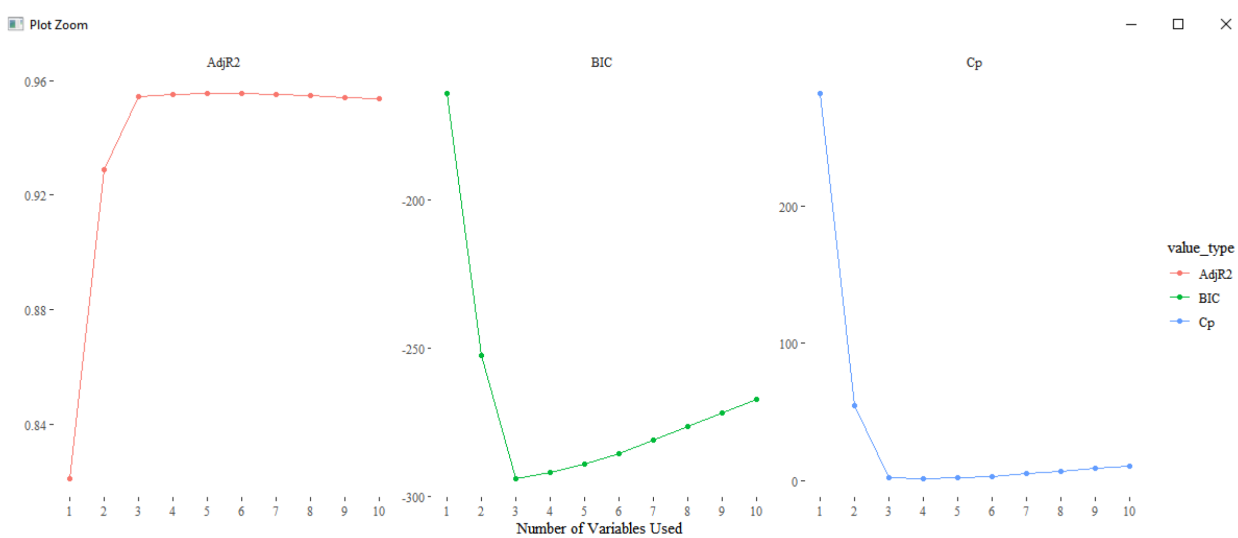
Import Dataset | List

Global Environment

Values	
noise	num [1:100] -0.6204 0.0421 -0.9109 0.158 -0.6546 ...
X	num [1:100] -0.626 0.184 -0.836 1.595 0.33 ...
Y	num [1:100] 3.57 3.35 4.63 10.87 3.07 ...

(c) (5 points) Use the regsubsets() function to perform best subset selection in order to choose the best model containing the predictors X, X2 , . . . , X10. What is the best model obtained according to Cp, BIC, and adjusted R2 ? 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 .

```
Console   Terminal
~/
> require(leaps)
> df <- data.frame(Y, X)
> fit <- regsubsets(Y ~ poly(X, 10), data = df, nvmax = 10)
> fit_summary <- summary(fit)
> require(tidyverse);require(ggplot2);require(ggthemes);
Loading required package: tidyverse
-- Attaching packages --------------------------------------- tidyverse 1.3.0 --
v tibble  3.0.4     v dplyr   1.0.2
v tidyr   1.1.2     v stringr 1.4.0
v readr   1.3.1     v forcats 0.4.0
v purrr   0.3.4
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
Warning messages:
1: package ‘tidyverse’ was built under R version 3.6.3
2: package ‘tibble’ was built under R version 3.6.3
3: package ‘tidyr’ was built under R version 3.6.3
4: package ‘purrr’ was built under R version 3.6.3
5: package ‘dplyr’ was built under R version 3.6.3
Loading required package: ggthemes
Warning message:
package ‘ggthemes’ was built under R version 3.6.3
> data_frame(Cp = fit_summary$cp,
+            BIC = fit_summary$bic,
+            AdjR2 = 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)
Warning message:
`data_frame()` is deprecated as of tibble 1.1.0.
Please use `tibble()` instead.
This warning is displayed once every 8 hours.
Call `lifecycle::last_warnings()` to see where this warning was generated.
>
```



Regsubsets chooses three as the optimal number of parameters, just like we declared the Y variable.

(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)?

Console Terminal

```
The downloaded binary packages are in
        C:\Users\SandeepReddy\AppData\Local\Temp\Rtmpgx3Hdg\downloaded_packages
> require(caret)
Loading required package: caret
Loading required package: lattice

Attaching package: ‘caret’

The following object is masked from ‘package:purrr’:

    lift

Warning messages:
1: package ‘caret’ was built under R version 3.6.3
2: package ‘lattice’ was built under R version 3.6.3
>
> model_back <- train(Y ~ poly(X, 10), data = df,
+                     method = 'glmStepAIC', direction = 'backward',
+                     trace = 0,
+                     trControl = trainControl(method = 'none', verboseIter = FALSE))
>
> postResample(predict(model_back, df), df$Y)
     RMSE  Rsquared       MAE
0.9314956 0.9569843 0.7488821
>
```

Console Terminal

```
> 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

>
```

From above backward stepwise model agrees with the best subsets model.

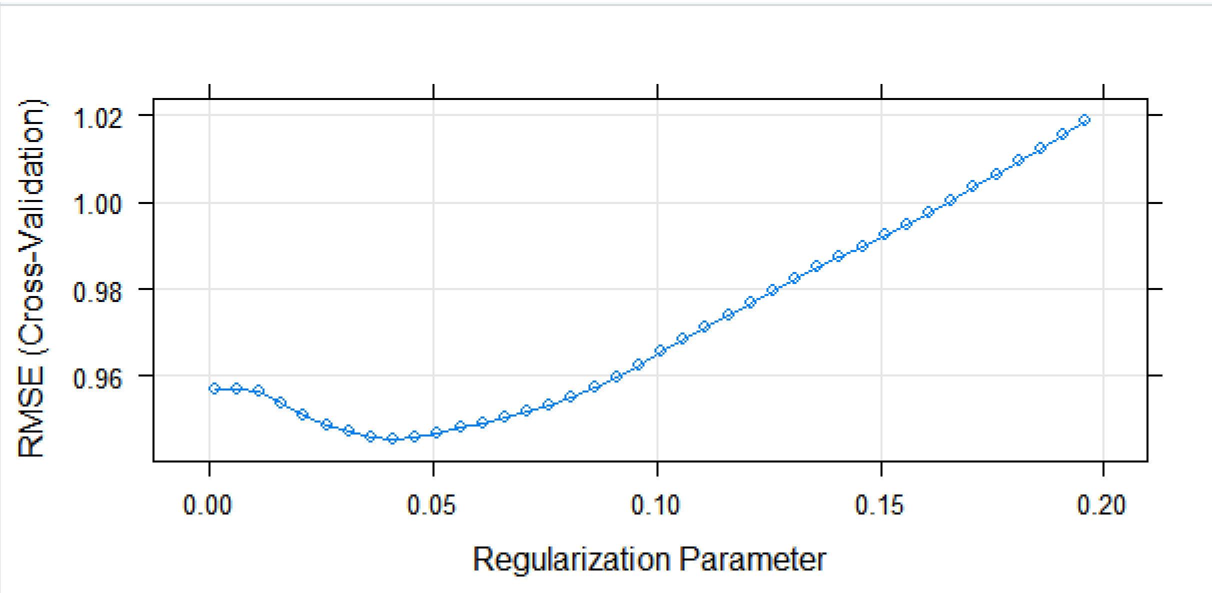
```
> colnames(x_poly) <- paste0('poly', 1:10)
> model_forw <- train(y = Y, x = x_poly,
+                     method = 'glmStepAIC', direction = 'forward',
+                     trace = 0,
+                     trControl = trainControl(method = 'none', verboseIter = FALSE))
>
> postResample(predict(model_forw, 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

>
```

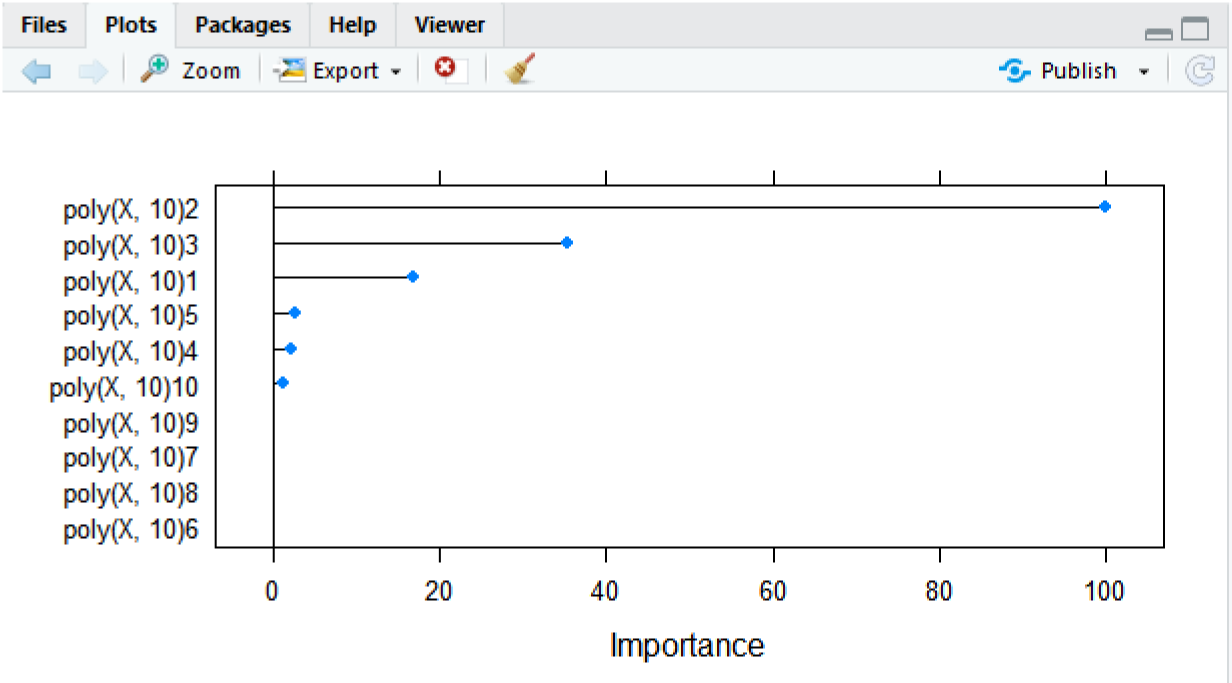
From above it is clear that forward stepwise model also agrees.

(e) (5 points) Now fit a lasso model to the simulated data, again using X, X2 , . . . , X10 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.

```
package ‘shape’ successfully unpacked and MD5 sums checked
package ‘glmnet’ successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\SandeepReddy\AppData\Local\Temp\Rtmpgx3Hdg\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 = 0.005)))
>
> plot(lasso_model)
>
```




```
> plot(varImp(lasso_model))
>
```



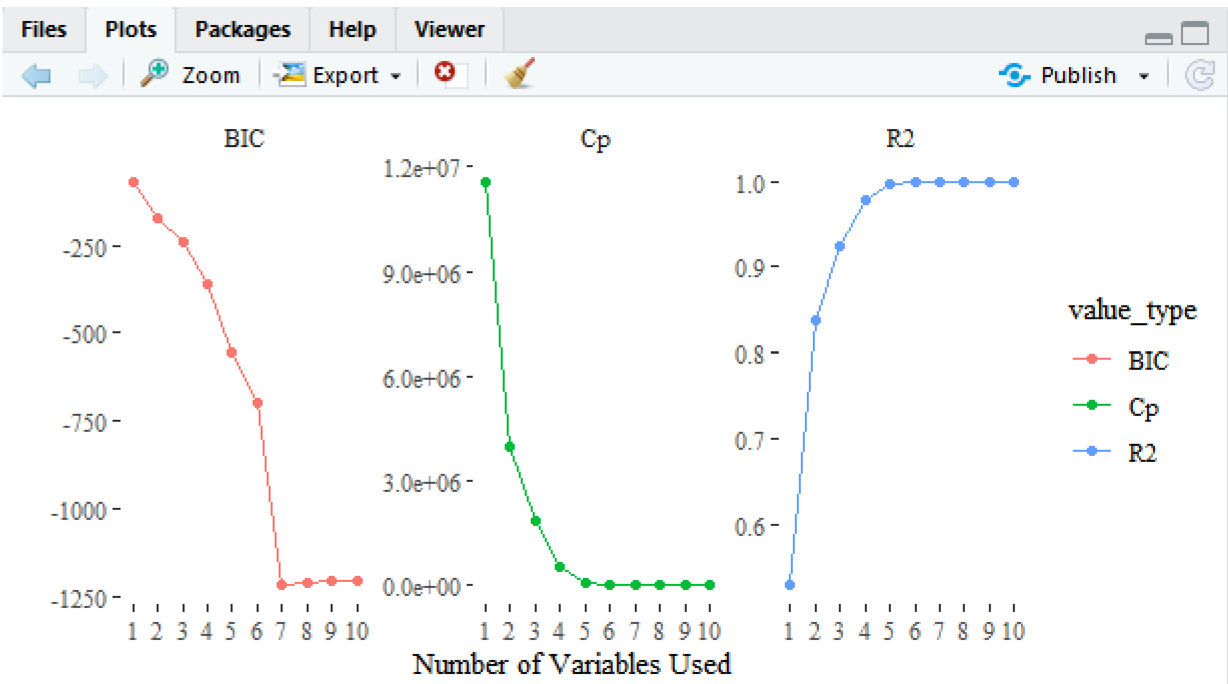

```
> coef(lasso_model$finalModel, lasso_model$bestTune$lambda)
11 x 1 sparse Matrix of class "dgCMatrix"
                        1
(Intercept)     6.1026472
poly(X, 10)1   -6.7829512
poly(X, 10)2   40.3340466
poly(X, 10)3  -14.2990830
poly(X, 10)4    0.8470950
poly(X, 10)5    1.0701884
poly(X, 10)6    .
poly(X, 10)7    .
poly(X, 10)8    .
poly(X, 10)9    .
poly(X, 10)10  -0.5412295
> 
```

```
> postResample(predict(lasso_model, df), df$Y)
     RMSE  Rsquared       MAE 
0.9235360 0.9578972 0.7511252 
> 
```

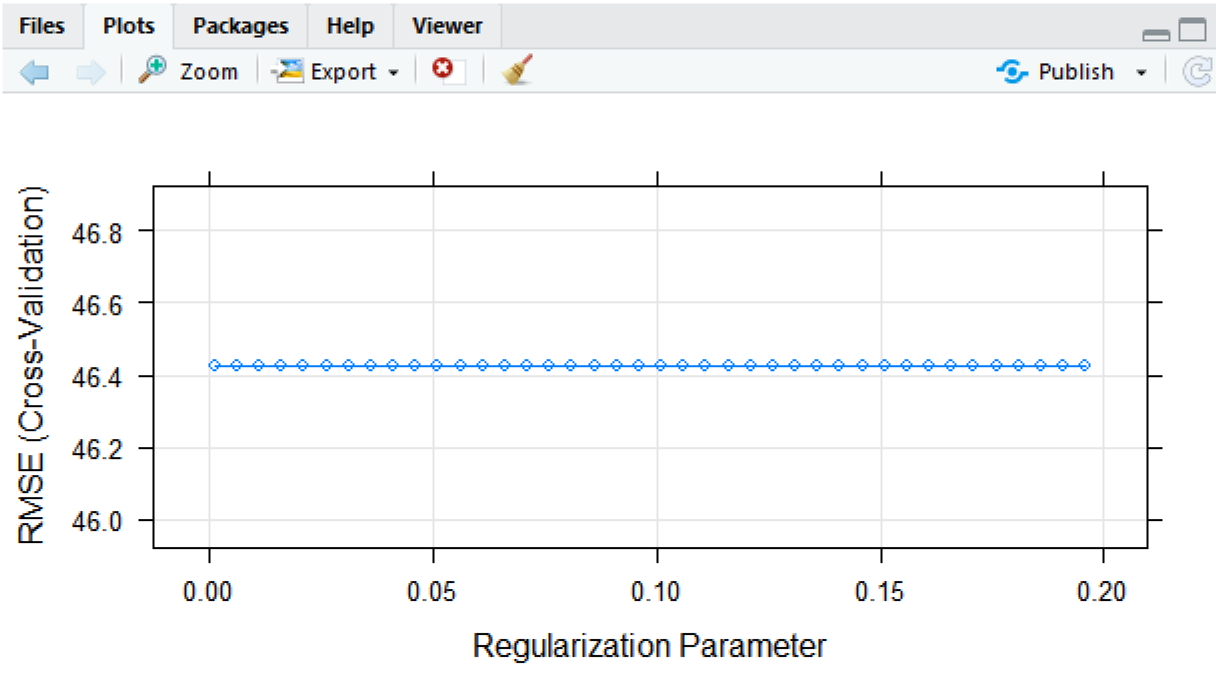
The Lasso model overestimates the number of predictors needed. This might be expected since we used only RSS to select the optimal model but not the Bayesian Inference Criterion and Adjusted R2R2 as regsubsets does or the Aikake Information Criterion as the stepwise selection does.

(f) (5 points)Now generate a response vector Y according to the model $Y = \beta_0 + \beta_7 X^7 + \epsilon$, and perform best subset selection and the lasso. Discuss the results obtained.

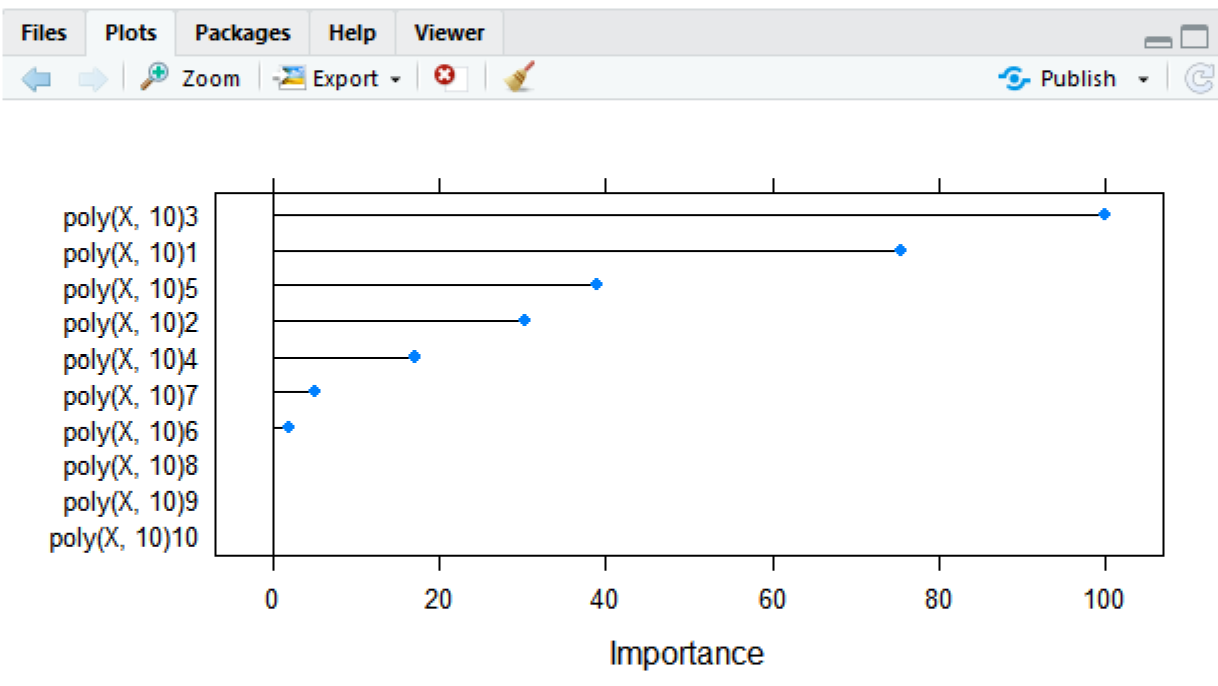
```
> Y_7 <- 3 + 8*X^7 + noise
> df_2 <- data_frame(Y_7 = Y_7, X = df[,-1])
>
> fit <- regsubsets(Y_7 ~ poly(X, 10), data = df_2, nvmax = 10)
>
> fit_summary <- summary(fit)
>
> data_frame(Cp = fit_summary$cp,
+            BIC = fit_summary$bic,
+            R2 = fit_summary$adjr2) %>%
+   mutate(id = 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
>
```

3. (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.

```
Console   Terminal
~/
> require(ISLR)
Loading required package: ISLR
Warning message:
package ‘ISLR’ was built under R version 3.6.3
> 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)
>
```

Environment | History | Connections

Import Dataset | List

Global Environment

Data	
college	777 obs. of 18 variables
df	100 obs. of 2 variables
df_2	100 obs. of 2 variables
fit	List of 28
fit_summary	List of 8
inTrain	int [1:585, 1] 1 2 3 4 5 8 9 11 12 13 ...
lasso_model	Large train (23 elements, 1.5 Mb)
lasso_y7_model	Large train (23 elements, 1.5 Mb)
model_back	List of 23
model_forw	List of 20
one_hot_encoding	List of 9
preObj	List of 21
testing	192 obs. of 18 variables
training	585 obs. of 18 variables
x_poly	'poly' num [1:100, 1:10] -0.08228 0.00836 -0.1056...
x_test	num [1:192, 1:18] 0 0 0 0 0 0 0 0 0 0 ...
x_train	num [1:585, 1:18] 0 0 0 0 0 0 0 0 0 0 ...
Values	
noise	num [1:100] -0.6204 0.0421 -0.9109 0.158 -0.6546 ...
X	num [1:100] -0.626 0.184 -0.836 1.595 0.33 ...
Y	num [1:100] 3.57 3.35 4.63 10.87 3.07 ...
Y_7	num [1:100] 2.077 3.042 -0.187 213.512 2.349 ...
y_test	num [1:192] -0.613 -0.673 -0.614 -0.637 -0.398 ...
y_train	num [1:585] -0.337 -0.202 -0.396 -0.656 -0.714 ...

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

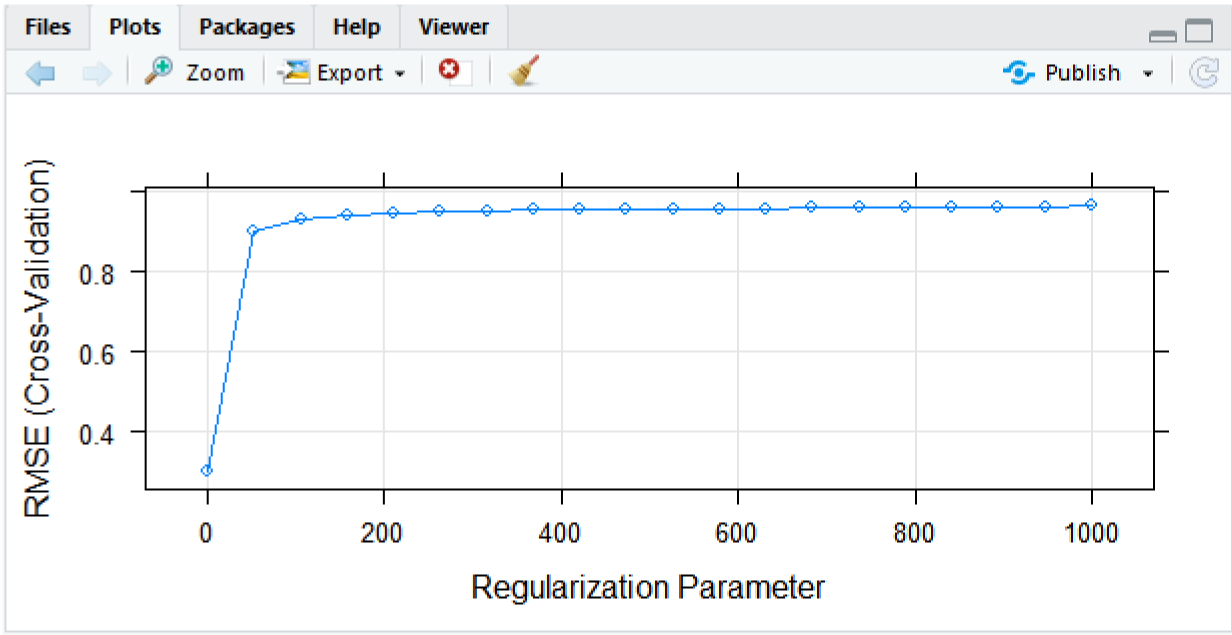
```
> lin_model <- lm(Apps ~ ., data = training)
>
> pred <- predict(lin_model, testing)
>
> (lin_info <- postResample(pred, testing$Apps))
     RMSE  Rsquared       MAE
0.2799768 0.9201765 0.1568743
>
```

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

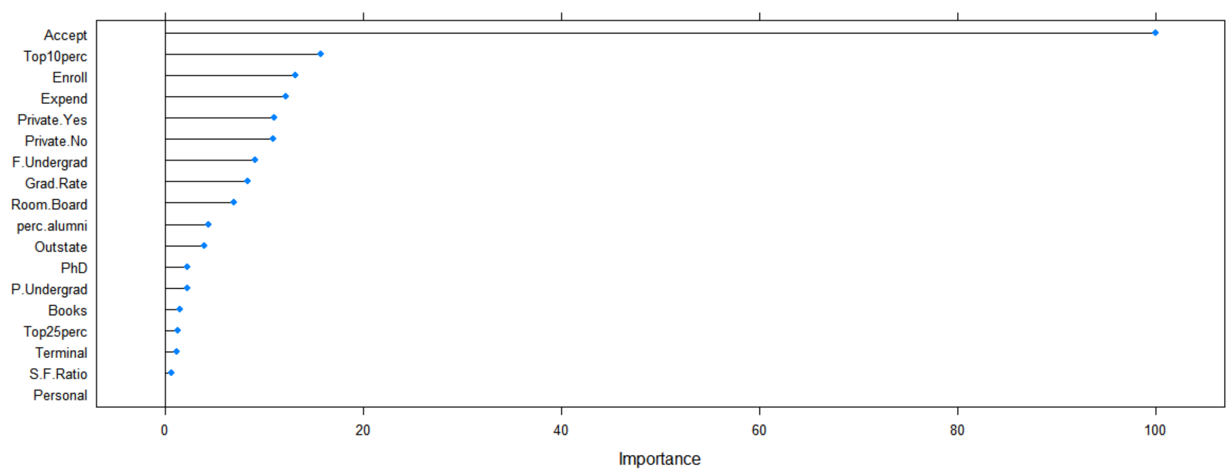
```
> ridge_fit <- train(x = x_train, y = y_train,
+                    method = 'glmnet',
+                    trControl = trainControl(method = 'cv', number = 10),
+                    tuneGrid = expand.grid(alpha = 0,
+                                           lambda = seq(0, 10e2, length.out = 20)))
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
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))
>
```

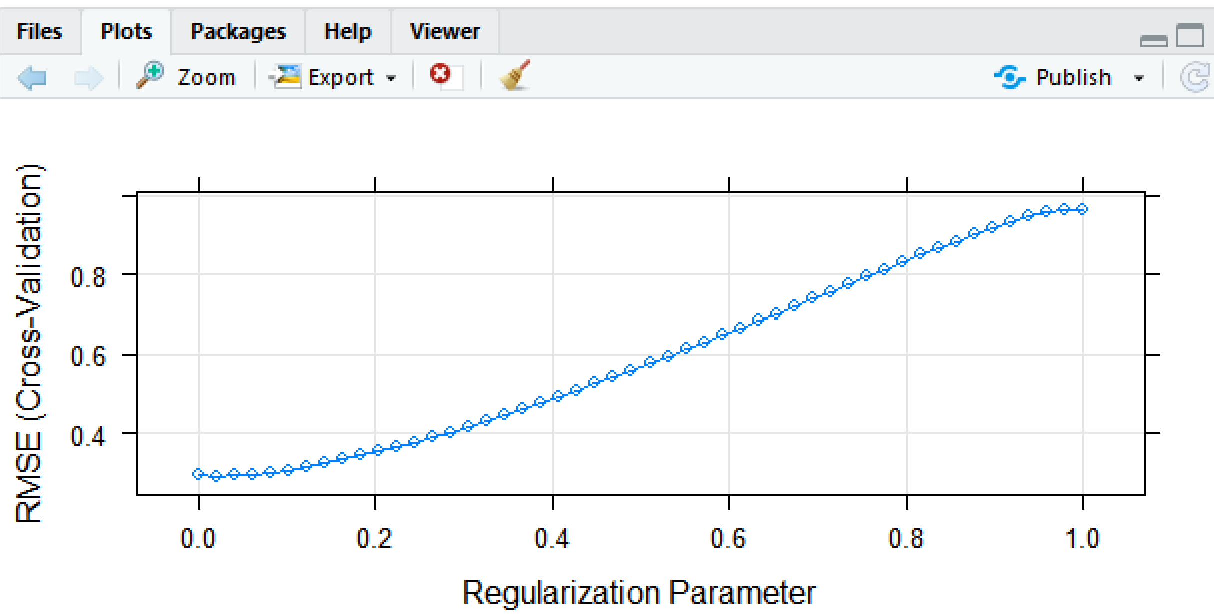


(d) (5 points) Fit a lasso model on the training set, with λ chosen by cross validation. Report the test error obtained, along with the number of non-zero coefficient estimates.

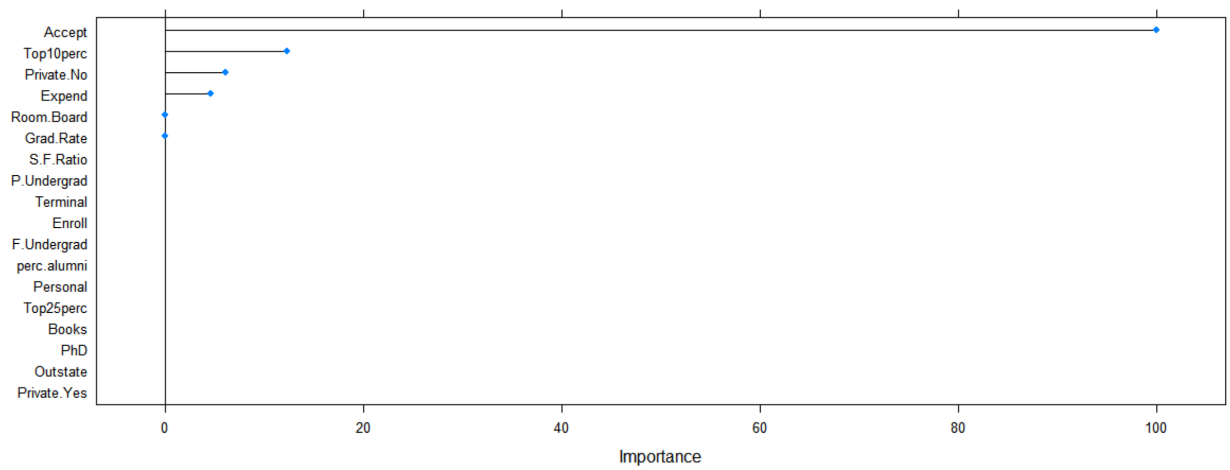
```
> lasso_fit <- train(x = x_train, y = y_train,
+                    method = 'glmnet',
+                    trControl = trainControl(method = 'cv', number = 10),
+                    tuneGrid = expand.grid(alpha = 1,
+                                           lambda = seq(0.0001, 1, length.out = 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 
0.2914812 0.9141364 0.1511801 
>
```

```
> coef(lasso_fit$finalModel, lasso_fit$bestTune$lambda)
19 x 1 sparse Matrix of class "dgCMatrix"
                        1
(Intercept) -1.470609e-02
Private.No   5.410732e-02
Private.Yes  .
Accept       8.883839e-01
Enroll       .
Top10perc    1.092201e-01
Top25perc    .
F.Undergrad  .
P.Undergrad  .
Outstate     .
Room.Board   1.483337e-04
Books        .
Personal     .
PhD          .
Terminal     .
S.F.Ratio    .
perc.alumni  .
Expend       4.067011e-02
Grad.Rate    4.062721e-06
>
```

```
> plot(lasso_fit)
>
```




```
> plot(varImp(lasso_fit))
>
```

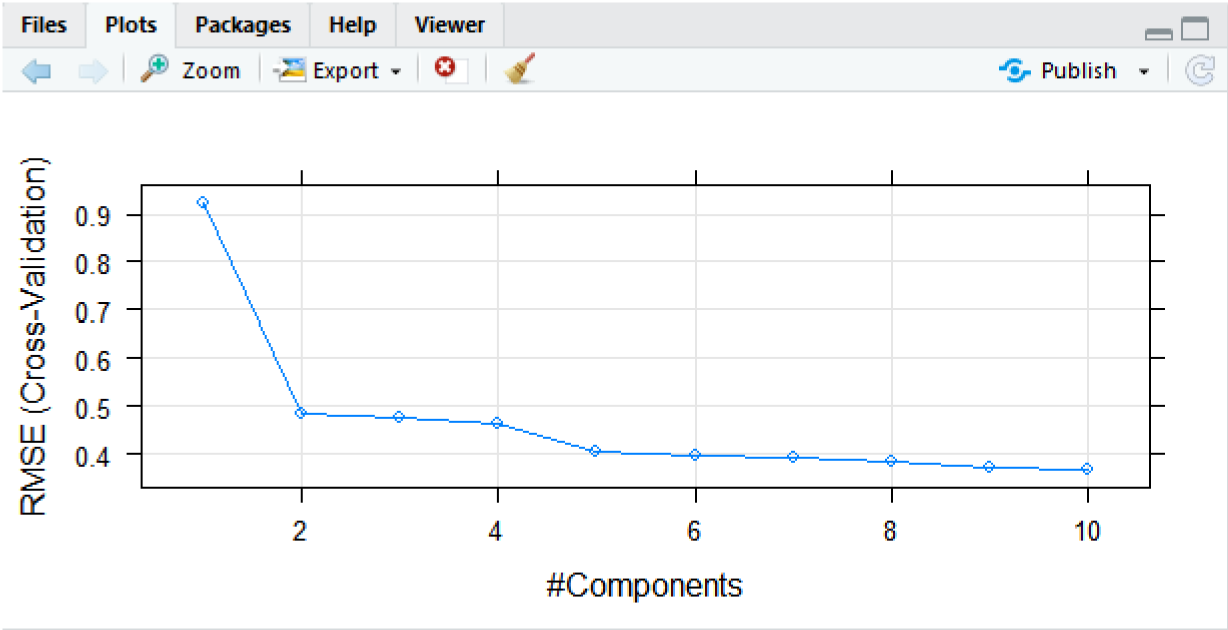


(e) (5 points) Fit a PCR model on the training set, with M chosen by cross validation. Report the test error obtained, along with the value of M selected by cross validation.

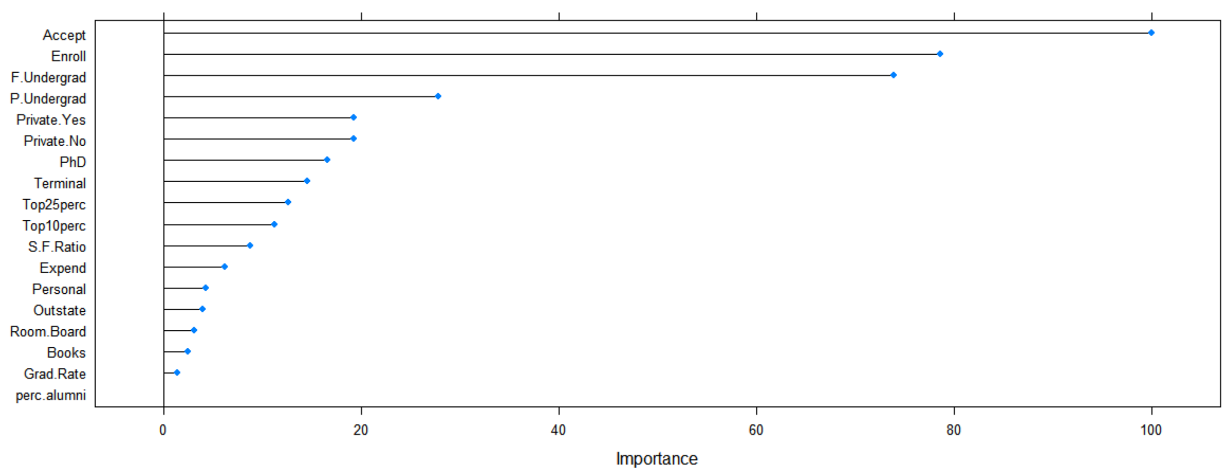
```
> pcr_model <- train(x = x_train, y = y_train,
+                    method = 'pcr',
+                    trControl = trainControl(method = 'cv', number = 10),
+                    tuneGrid = expand.grid(ncomp = 1:10))
> (pcr_info <- postResample(predict(pcr_model, x_test), y_test))
     RMSE  Rsquared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

> 
```

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




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

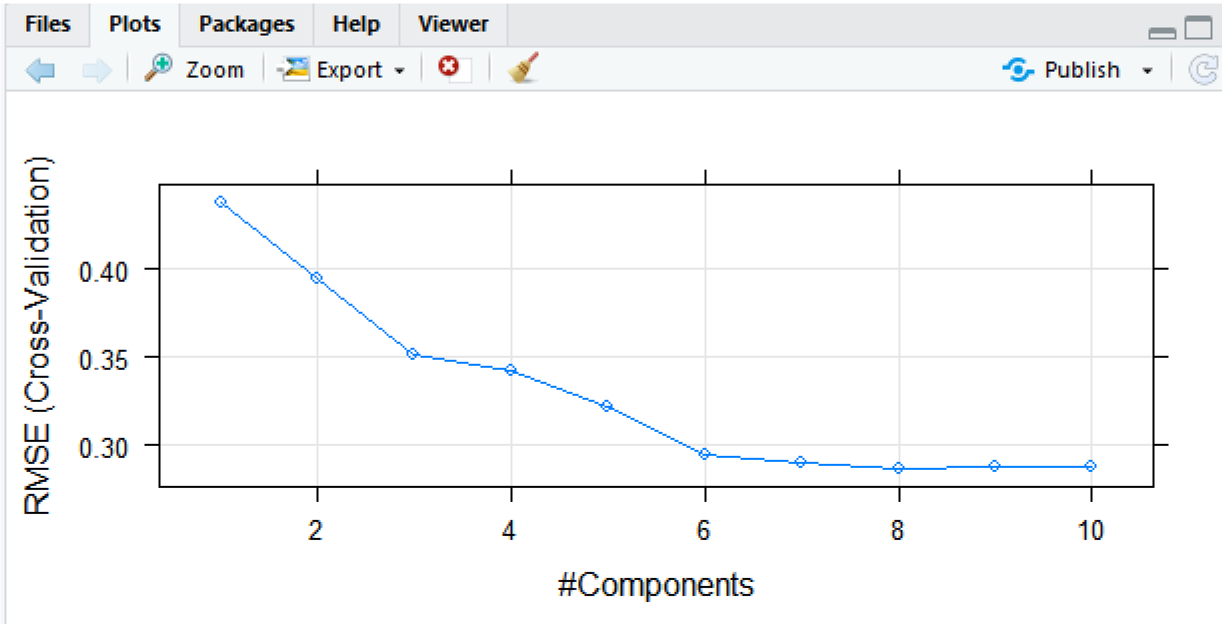


(f) (5 points) Fit a PLS model on the training set, with M chosen by cross validation. Report the test error obtained, along with the value of M selected by cross validation.

```
> 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

> 
```

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



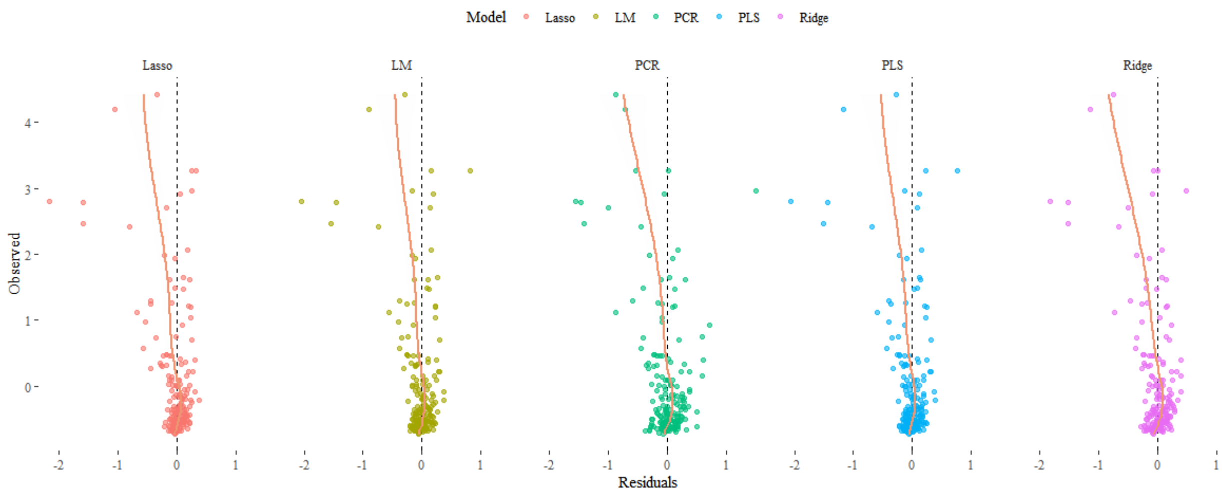
(g) (5 points) Comment on the results obtained. How accurately can we predict the number of college applications received? Is there much difference among the test errors resulting from these five approaches?

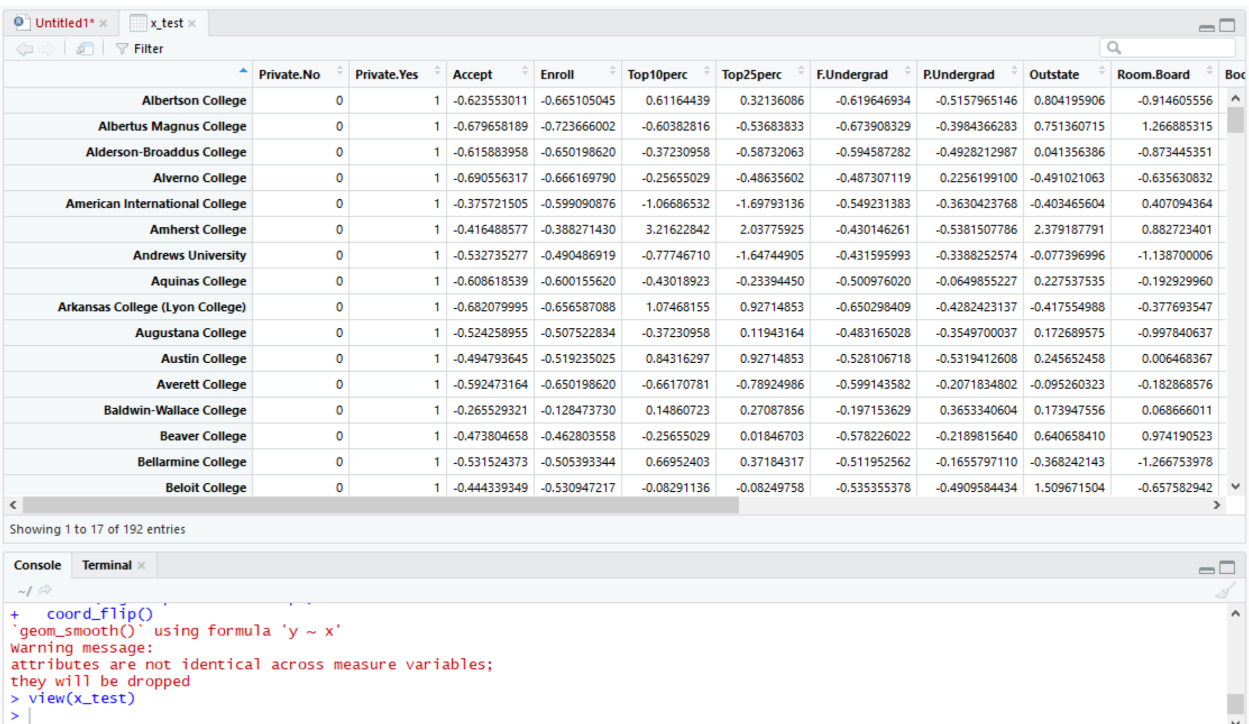
```
> as_data_frame(rbind(lin_info,
+                     ridge_info,
+                     lasso_info,
+                     pcr_info,
+                     pls_info)) %>%
+   mutate(model = c('Linear', 'Ridge', 'Lasso', 'PCR', 'PLS')) %>%
+   select(model, RMSE, Rsquared)
# A tibble: 5 x 3
  model    RMSE Rsquared
  <chr>   <dbl>    <dbl>
1 Linear  0.280    0.920
2 Ridge   0.285    0.921
3 Lasso   0.291    0.914
4 PCR     0.323    0.892
5 PLS     0.284    0.919
Warning message:
`as_data_frame()` is deprecated as of tibble 2.0.0.
Please use `as_tibble()` instead.
The signature and semantics have changed, see `?as_tibble`.
This warning is displayed once every 8 hours.
Call `lifecycle::last_warnings()` to see where this warning was generated.
> 
```

```
> testing %>%
+   summarize(sd = sd(Apps))
         sd
1 0.9818241
> 
```

The models all perform similarly, with the exception of the PCR model. $R2 \geq 94$R2≥94 for them all and $RMSE \leq 20$RMSE≤20. When we compare the RMSE scores with the mean and standard deviation of the response variable we see that the models all have phenomenal accuracy!

```
Console   Terminal ×
~/
> 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 = 'lightsalmon2') +
+   facet_wrap(~ Model, ncol = 5) +
+   theme_tufte() +
+   theme(legend.position = 'top') +
+   coord_flip()
`geom_smooth()` using formula 'y ~ x'
Warning message:
attributes are not identical across measure variables;
they will be dropped
> 
```





Untitled1* | x_test

Filter

	Private.No	Private.Yes	Accept	Enroll	Top10perc	Top25perc	F.Undergrad	P.Undergrad	Outstate	Room.Board
Albertson College	0	1	-0.623553011	-0.665105045	0.61164439	0.32136086	-0.619646934	-0.5157965146	0.804195906	-0.914605556
Albertus Magnus College	0	1	-0.679658189	-0.723666002	-0.60382816	-0.53683833	-0.673908329	-0.3984366283	0.751360715	1.266885315
Alderson-Broaddus College	0	1	-0.615883958	-0.650198620	-0.37230958	-0.58732063	-0.594587282	-0.4928212987	0.041356386	-0.873445351
Alverno College	0	1	-0.690556317	-0.666169790	-0.25655029	-0.48635602	-0.487307119	0.2256199100	-0.491021063	-0.635630832
American International College	0	1	-0.375721505	-0.599090876	-1.06686532	-1.69793136	-0.549231383	-0.3630423768	-0.403465604	0.407094364
Amherst College	0	1	-0.416488577	-0.388271430	3.21622842	2.03775925	-0.430146261	-0.5381507786	2.379187791	0.882723401
Andrews University	0	1	-0.532735277	-0.490486919	-0.77746710	-1.64744905	-0.431595993	-0.3388252574	-0.077396996	-1.138700006
Aquinas College	0	1	-0.608618539	-0.600155620	-0.43018923	-0.23394450	-0.500976020	-0.0649855227	0.227537535	-0.192929960
Arkansas College (Lyon College)	0	1	-0.682079995	-0.656587088	1.07468155	0.92714853	-0.650298409	-0.4282423137	-0.417554988	-0.377693547
Augustana College	0	1	-0.524258955	-0.507522834	-0.37230958	0.11943164	-0.483165028	-0.3549700037	0.172689575	-0.997840637
Austin College	0	1	-0.494793645	-0.519235025	0.84316297	0.92714853	-0.528106718	-0.5319412608	0.245652458	0.006468367
Averett College	0	1	-0.592473164	-0.650198620	-0.66170781	-0.78924986	-0.599143582	-0.2071834802	-0.095260323	-0.182868576
Baldwin-Wallace College	0	1	-0.265529321	-0.128473730	0.14860723	0.27087856	-0.197153629	0.3653340604	0.173947556	0.068666011
Beaver College	0	1	-0.473804658	-0.462803558	-0.25655029	0.01846703	-0.578226022	-0.2189815640	0.640658410	0.974190523
Bellarmine College	0	1	-0.531524373	-0.505393344	0.66952403	0.37184317	-0.511952562	-0.1655797110	-0.368242143	-1.266753978
Beloit College	0	1	-0.444339349	-0.530947217	-0.08291136	-0.08249758	-0.535355378	-0.4909584434	1.509671504	-0.657582942

Showing 1 to 17 of 192 entries

Console | Terminal

```
+   coord_flip()
`geom_smooth()` using formula 'y ~ x'
Warning message:
attributes are not identical across measure variables;
they will be dropped
> View(x_test)
>
```

4. (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 = Xβ + ϵ, where β has some elements that are exactly equal to zero.

```
> require(tidyverse)
> set.seed(1)
> df <- data.frame(replicate(20, rnorm(n = 1000)))
>
> df %>%
+   reduce(function(y, x) y + ifelse(runif(1) < 0.5,
+                                    rnorm(1, mean = 5, sd = 1),
+                                    0)*x + rnorm(1000)) -> df$Y
>
```

Environment | History | Connections

Import Dataset | List

Global Environment

lin_model	List of 13
model_back	List of 23
model_forw	List of 20
one_hot_encoding	List of 9
pcr_model	Large train (20 elements, 844.4 Kb)
pls_model	Large train (20 elements, 1.2 Mb)
preObj	List of 21
ridge_fit	Large train (20 elements, 1.8 Mb)
testing	192 obs. of 18 variables
training	585 obs. of 18 variables
x_poly	'poly' num [1:100, 1:10] -0.08228 0.0083...
x_test	900 obs. of 20 variables
x_train	100 obs. of 20 variables
Values	
lasso_info	Named num [1:3] 0.291 0.914 0.151
lin_info	Named num [1:3] 0.28 0.92 0.157
noise	num [1:100] -0.6204 0.0421 -0.9109 0.158 -...
pcr_info	Named num [1:3] 0.323 0.892 0.199
pls_info	Named num [1:3] 0.284 0.919 0.159
pred	Named num [1:192] -0.582 -0.804 -0.593 -0....
ridge_info	Named num [1:3] 0.285 0.921 0.165
X	num [1:100] -0.626 0.184 -0.836 1.595 0.33...
Y	num [1:100] 3.57 3.35 4.63 10.87 3.07 ...
Y_7	num [1:100] 2.077 3.042 -0.187 213.512 2.3...
y_test	num [1:900] -25.88 16.27 7.68 -15.45 -28.4...
y_train	num [1:100] 8.79 9.07 -28.84 10.56 -6.58 ...
Functions	
residfunc	function (fit, data)

I use the reduce function from the purr package to more easily compute the *y* variable.

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

```
> require(caret)
>
> inTrain <- createDataPartition(df$Y, p = 0.1, list = F)
>
> x_train <- df[inTrain, -21]
> y_train <- df[inTrain, 21]
> x_test <- df[-inTrain, -21]
> y_test <- df[-inTrain, 21]
>
```

Untitled1* x_test

Filter

	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10
1	-0.626453811	1.134965089	-0.88614959	7.391149e-01	-1.134630182	-1.51637331	-0.61882708	-1.325417721	0.263703401	-1.21712008
2	0.183643324	1.111931845	-1.92225490	3.866087e-01	0.764557099	0.62914119	-1.10942196	0.951979720	-0.829451847	-0.94622927
3	-0.835628612	-0.870777634	1.61970074	1.296397e+00	0.570710138	-1.67819404	-2.17033523	0.860004385	-1.461634774	0.09140980
5	0.329507772	0.069395647	-0.05584993	-1.602626e+00	-2.029885469	1.11765454	-0.26039848	-0.350583963	-1.544324288	0.67342236
6	-0.820468384	-1.662648853	0.69641761	9.332510e-01	0.590478654	-1.23773594	0.53443047	-0.130765556	-0.190887125	1.26555336
8	0.738324705	-1.912345796	-1.31028350	-5.650363e-02	1.610341551	0.59779092	1.60837019	-0.493905734	0.547126202	1.41541830
9	0.575781352	-1.246753429	-2.12306606	1.885911e+00	1.840442547	0.29886441	0.55663975	1.113359848	0.755154008	-1.58503378
10	-0.305388387	0.998154445	-0.20807859	1.578383e+00	1.368297910	-0.11013937	0.18562248	1.458962722	-0.419804197	0.24575719
11	1.511781168	-0.540872745	-0.31278658	5.022846e-01	-1.255573135	-0.80767502	-1.03940831	0.634593124	1.810782062	0.45023310
12	0.389843236	-0.216375791	-1.05823571	4.299142e-01	-1.384347088	0.11453917	-0.36338204	1.816680242	-0.110802356	0.94763282
13	-0.621240581	-1.621937293	0.41722360	-1.265646e+00	-0.019579679	-0.17952006	-1.37689058	-0.320281897	0.360255450	-0.25447147
14	-2.214699887	-1.450963965	-0.31545153	2.236232e+00	0.162585655	0.05484490	-0.53547274	1.437864528	-0.106309480	0.08533976
15	1.124930918	0.350909731	0.82554913	3.319684e-01	-0.134708351	1.29913984	0.27483471	-1.901251459	-0.692710708	1.20511062
16	-0.044933609	-0.174546929	1.29127204	-1.392824e-01	-0.084798638	-0.43456121	1.31569555	0.249615402	1.056548366	0.17067269
17	-0.016190263	-0.591428470	-0.62510498	-7.345938e-01	-0.572961857	-0.80892302	-0.17049581	-0.055960843	0.763626971	2.27598615
18	0.943836211	-1.334027261	-0.87514675	-2.777593e+00	1.667606436	-0.52235147	1.44695365	-1.549279524	-2.230514322	1.32860267
19	0.821221195	-1.097298501	0.14335694	-3.228540e-01	-0.576701735	-0.28771574	1.64721351	0.063395598	0.392795259	0.30935080
20	0.593901321	2.036103609	1.42518963	-1.035586e+00	0.301856343	-0.64321325	1.01349143	0.678469773	-0.763776844	-0.64305417
21	0.918977372	-0.326489593	-1.73474943	5.475265e-02	-0.389822328	0.34281872	0.05319124	0.197444290	-0.160173691	-0.73722431

Showing 1 to 20 of 900 entries

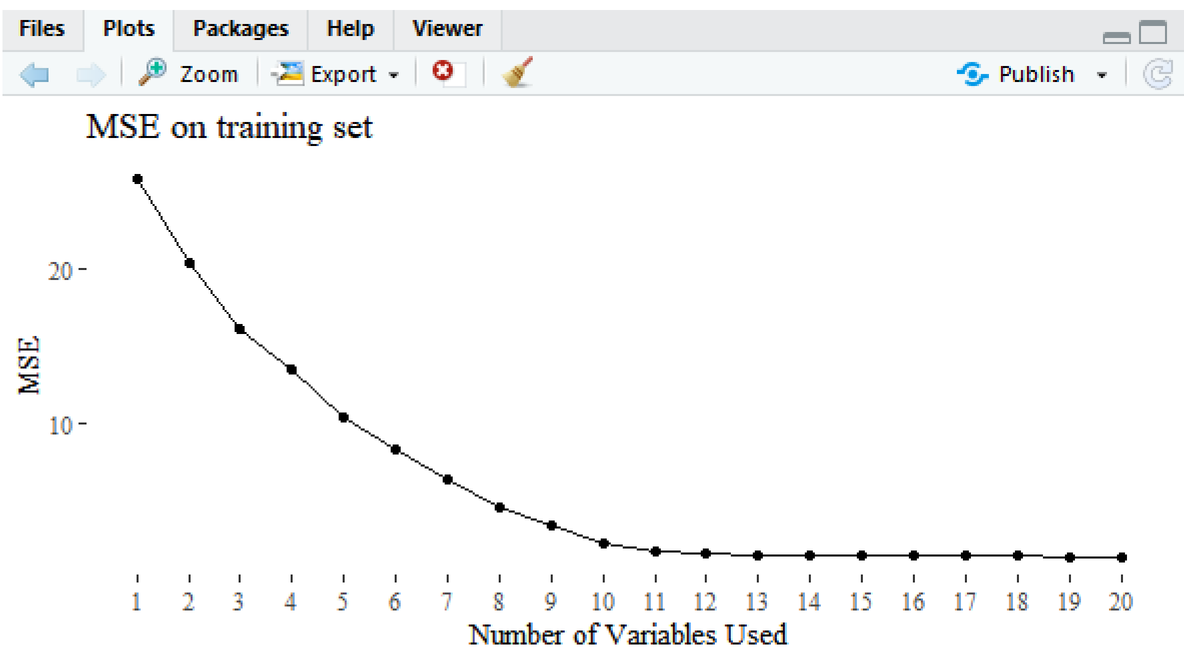
Console Terminal

```
~/
> View(x_test)
>
```

The *createDataPartition* function from the *caret* creates train/test splits. In case of unbalanced classes it also makes sure one gets an even split of the response variable.

(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.

```
Console   Terminal ×
~/
> require(leaps); require(ggplot2); require(dplyr); require(ggthemes)
>
> best_set <- regsubsets(x = x_train, y = y_train, nvmax = 20)
>
> best_set_summary <- summary(best_set)
>
> data_frame(MSE = best_set_summary$rss/900) %>%
+   mutate(id = row_number()) %>%
+   ggplot(aes(id, MSE)) +
+   geom_line() + geom_point(type = 9) +
+   xlab('Number of Variables Used') +
+   ggtitle('MSE on training set') +
+   theme_tufte() +
+   scale_x_continuous(breaks = 1:20)
Warning message:
Ignoring unknown parameters: type
> 
```

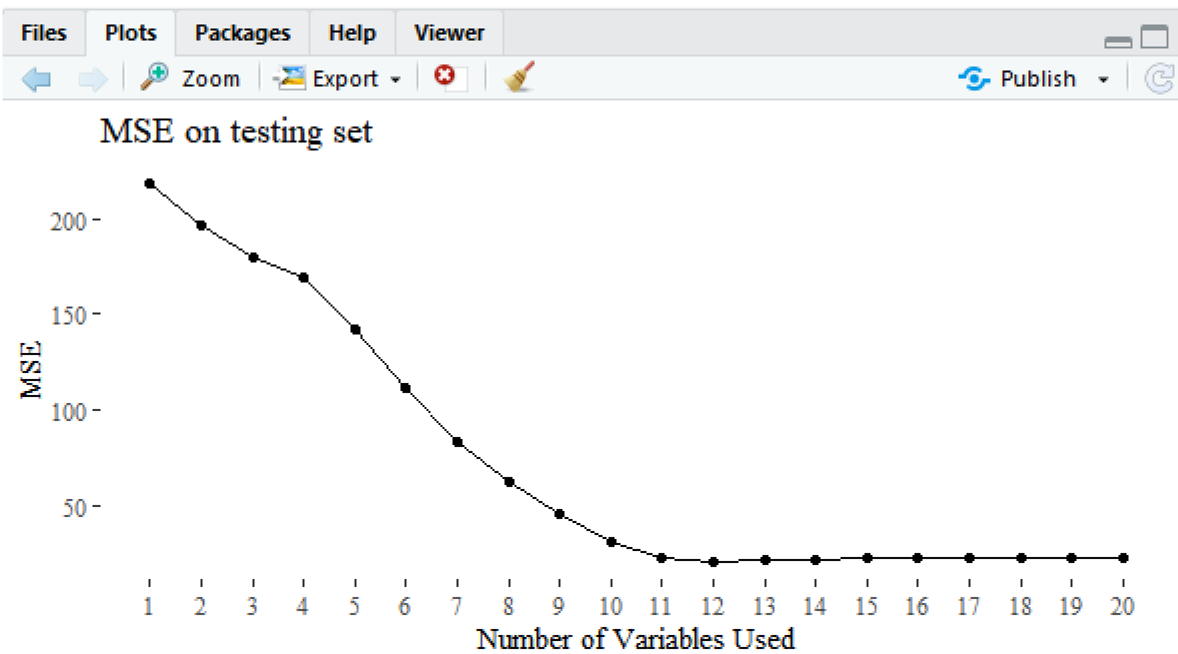



```
> data_frame(train_error = best_set_summary$rss/900, vars = 1:20) %>%
+   spread(vars, train_error)
# A tibble: 1 x 20
    `1`   `2`   `3`   `4`   `5`   `6`   `7`   `8`   `9`  `10`  `11`  `12`  `13`  `14`
  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1  25.9  20.4  16.2  13.5  10.4  8.32  6.46  4.68  3.41  2.31  1.78  1.63  1.56  1.52
# ... with 6 more variables: `15` <dbl>, `16` <dbl>, `17` <dbl>, `18` <dbl>,
#   `19` <dbl>, `20` <dbl>
> 
```

From above ss one would expect, the training MSE reduces with the addition of any new variable. Even after we've reached 10 predictors.

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

```
> test_errors = rep(NA,19)
> test.mat <- model.matrix(Y ~ ., data = df[-inTrain,])
> for (i in 1:20){
+   coefs = coef(best_set, id=i)
+   pred = test.mat[,names(coefs)]%*%coefs
+   test_errors[i] = mean((y_test-pred)^2)
+ }
>
>
> data_frame(MSE = test_errors) %>%
+   mutate(id = row_number()) %>%
+   ggplot(aes(id, MSE)) +
+   geom_line() + geom_point(type = 9) +
+   xlab('Number of Variables Used') +
+   ggtitle('MSE on testing set') +
+   theme_tufte() +
+   scale_x_continuous(breaks = 1:20)
Warning message:
Ignoring unknown parameters: type
>
```




```
> which.min(test_errors)
[1] 12
```

```
> data_frame(test_errors, vars = 1:20) %>%
+   spread(vars, test_errors)
# A tibble: 1 x 20
    `1`   `2`   `3`   `4`   `5`   `6`   `7`   `8`   `9`  `10`  `11`  `12`  `13`  `14`
  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1  219.  197.  180.  169.  142.  112.  83.5  62.3  45.5  31.0  22.3  20.9  21.2  21.5
# ... with 6 more variables: `15` <dbl>, `16` <dbl>, `17` <dbl>, `18` <dbl>,
#   `19` <dbl>, `20` <dbl>
>
```

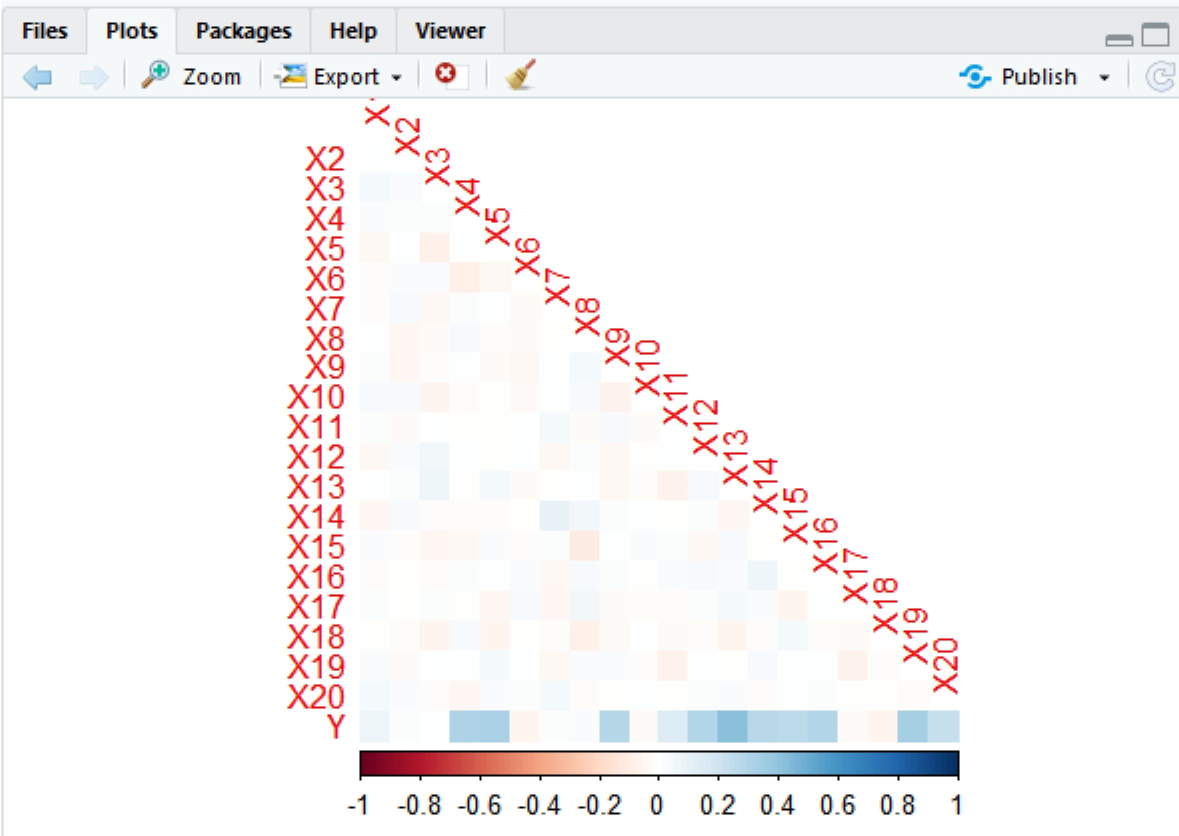
The test error settles at a minimum test error at 11 predictors and then stops decreasing.

(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.

```
> which.min(test_errors)
[1] 12
>
```

The reported MSE minimum on the test set is achieved with 11 coefficients. This makes sense when we look at the corrplot below.

```
> require(corrplot)
Loading required package: corrplot
corrplot 0.84 loaded
Warning message:
package ‘corrplot’ was built under R version 3.6.3
> corrplot(cor(df), method = 'color', type = 'lower',diag = F)
>
```



We can see on the corrplot that there are about 10 variables that correlate well with the response variable.

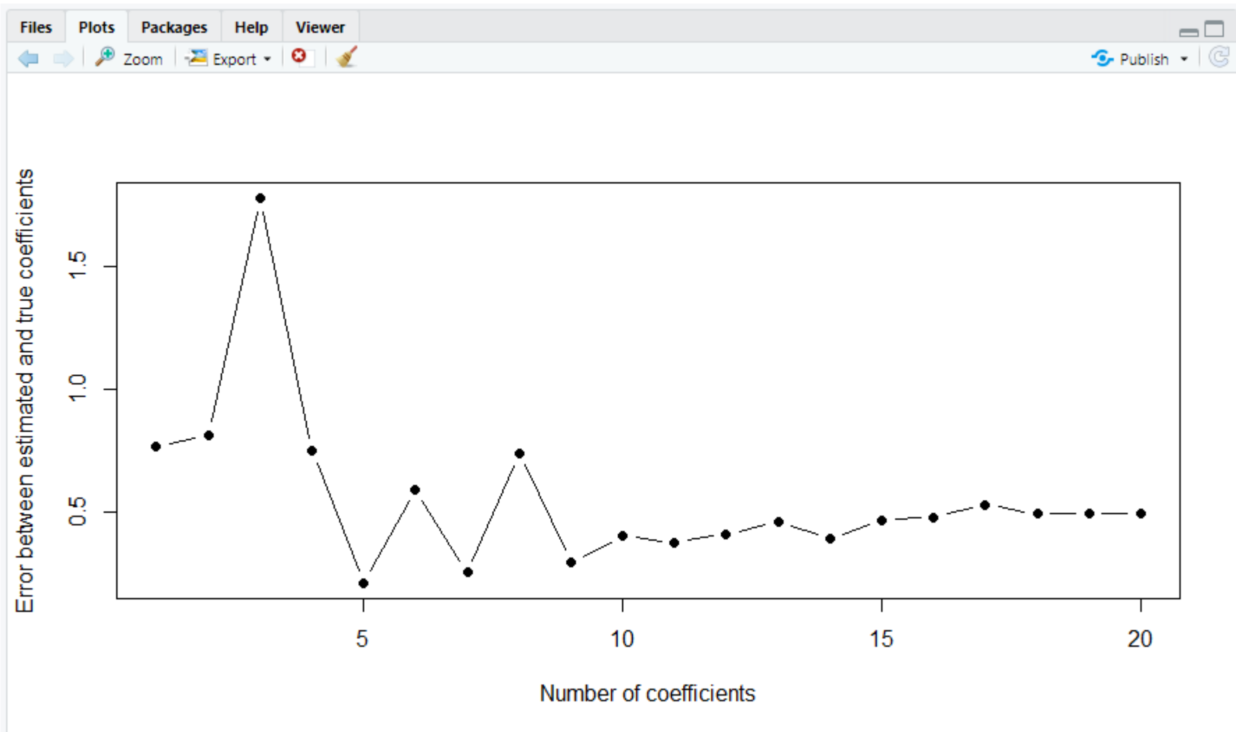
(f) (5 points) How does the model at which the test set MSE is minimized compare to the true model used to generate the data? Comment on the coefficient values.

Thinking back to the calculation of *Y*: If `runif(1) > 0.5` the coefficient would be 0. That means in about 50% of cases the coefficient will be 0. 50% of 20 is 10.

```
> coef(regfit.full, which.min(val.errors))
 (Intercept)          x.2          x.4          x.5          x.6          x.7          x.8         x.11         x.12
-0.003933937  0.359127426  0.202707344  1.036265913 -0.253843053 -1.282753293  0.691581077  0.895769881  0.526887865
        x.13         x.14         x.15         x.16         x.17         x.18         x.20
-0.207638251 -0.507929833 -0.892604795 -0.343062241  0.184479252  1.646950451 -1.060191640
> |
```

(g) (5 points) Create a plot displaying $\sqrt{\sum_{j=1}^{p}(\beta_j - \hat{\beta}_j^r)^2}$ 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)?

```
>
> val.errors <- rep(NA, 20)
> x_cols = colnames(x, do.NULL = FALSE, prefix = "x.")
> for (i in 1:20) {
+     coefi <- coef(regfit.full, id = i)
+     val.errors[i] <- sqrt(sum((b[x_cols %in% names(coefi)] - coefi[names(coefi) %in% x_cols])^2) + sum(b[!(x_cols %in% na
mes(coefi))])^2)
+ }
> plot(val.errors, xlab = "Number of coefficients", ylab = "Error between estimated and true coefficients", pch = 19, type
 = "b")
> |
```



We may see that the model with 3 variables minimizes the error between the estimated and true coefficients. However test error is minimized by the model with 14 variables. So, a better fit of true coefficients doesn't necessarily mean a lower test MSE.