CFB: Predicting 'Wins'

Smells Like Team Spirit

```r
library(readr)
library(car)
```

```
## Loading required package: carData
```

```r
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```r
library(leaps)
library(Stat2Data)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following object is masked from 'package:car':
##
##     recode

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(tidyr)
library(ggplot2)
library(GGally)
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```r
source("ShowSubsets.R")
source("VIF.R")

library(RCurl)
```

```
##
## Attaching package: 'RCurl'

## The following object is masked from 'package:tidyr':
##
##     complete
```

```
cfb_all_data <- read.csv("cfb_all_data.csv")
cfb_all_data <- subset(cfb_all_data, ScoreOff != 0)
head(cfb_all_data)
```

```
##   Year      TeamName Wins ScoreOff RushAttOff RushYdsOff PassAttOff PassCompOff
## 1 2002         Akron    4      311        489       1890        404         265
## 2 2002    Ball State    7      255        499       1955        348         199
## 3 2002 Bowling Green    9      439        500       2629        398         227
## 4 2002   Connecticut    5      359        450       1639        392         232
## 5 2002  Fresno State    7      332        463       1668        452         250
## 6 2002        Hawaii    9      442        286       1467        679         375
##   PassYdsOff PassIntOff FumblesOff ScoreDef RushAttDef RushYdsDef PassAttDef
## 1       2962         12         14      379        492       2008        340
## 2       2144         16         10      333        440       2035        390
## 3       2758         11          7      297        469       1844        398
## 4       2671         12          8      270        459       1868        322
## 5       3194         14         10      358        537       2104        433
## 6       5043         25         13      353        552       2218        456
##   PassCompDef PassYdsDef PassIntDef FumblesDef
## 1         214       2726          9         14
## 2         246       2834         10         23
## 3         214       2537         12         51
## 4         162       1925         20         14
## 5         245       3214         13         16
## 6         233       2928         18         32
```

I) Predicting wins of an CFB team

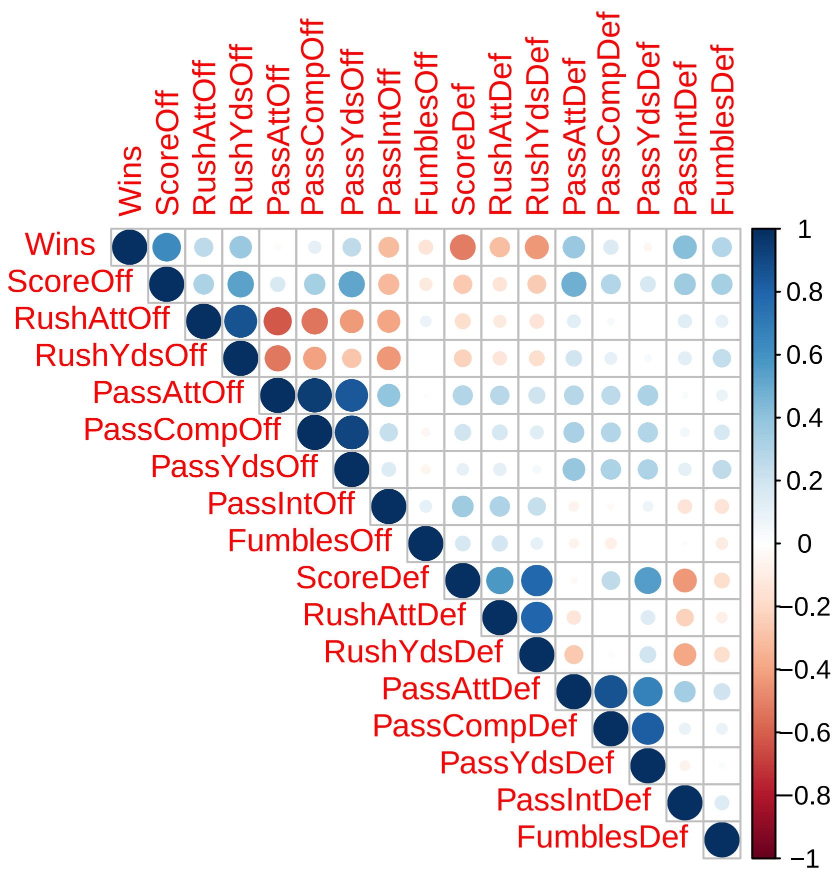
1. Correlation between predictors and Full Model with all predictors

```
cfb_data = cfb_all_data[3:19]

Fullcfb = lm(Wins~., cfb_data)
summary(Fullcfb)
```

```
##
## Call:
## lm(formula = Wins ~ ., data = cfb_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.6409 -1.0409 -0.0422  1.0188  4.9838
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept)  2.520e+00  6.217e-01   4.052 5.34e-05 ***
## ScoreOff     1.093e-02  1.300e-03   8.409  < 2e-16 ***
## RushAttOff   2.300e-03  1.231e-03   1.868 0.061943 .
## RushYdsOff  -2.261e-04  1.872e-04  -1.207 0.227473
## PassAttOff  -5.051e-03  2.146e-03  -2.353 0.018748 *
## PassCompOff  5.888e-03  2.933e-03   2.008 0.044877 *
## PassYdsOff   5.995e-05  2.003e-04   0.299 0.764765
## PassIntOff   3.742e-03  1.215e-02   0.308 0.758097
## FumblesOff  -2.036e-02  1.262e-02  -1.613 0.106862
## ScoreDef    -9.955e-03  1.360e-03  -7.317 4.21e-13 ***
## RushAttDef  -7.028e-04  1.481e-03  -0.475 0.635189
## RushYdsDef   3.212e-04  2.216e-04   1.449 0.147487
## PassAttDef   8.244e-03  2.472e-03   3.335 0.000876 ***
## PassCompDef -2.138e-03  3.337e-03  -0.641 0.521738
## PassYdsDef  -5.902e-05  2.344e-04  -0.252 0.801201
## PassIntDef   3.762e-02  1.210e-02   3.109 0.001917 **
## FumblesDef   6.544e-03  3.033e-03   2.158 0.031115 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.528 on 1423 degrees of freedom
## Multiple R-squared:  0.5606, Adjusted R-squared:  0.5556
## F-statistic: 113.5 on 16 and 1423 DF,  p-value: < 2.2e-16
```

```
corr = cor(cfb_data, use="pairwise.complete.obs")
corrplot(corr, type="upper")
```



```r
vif(Fullcfb)
```

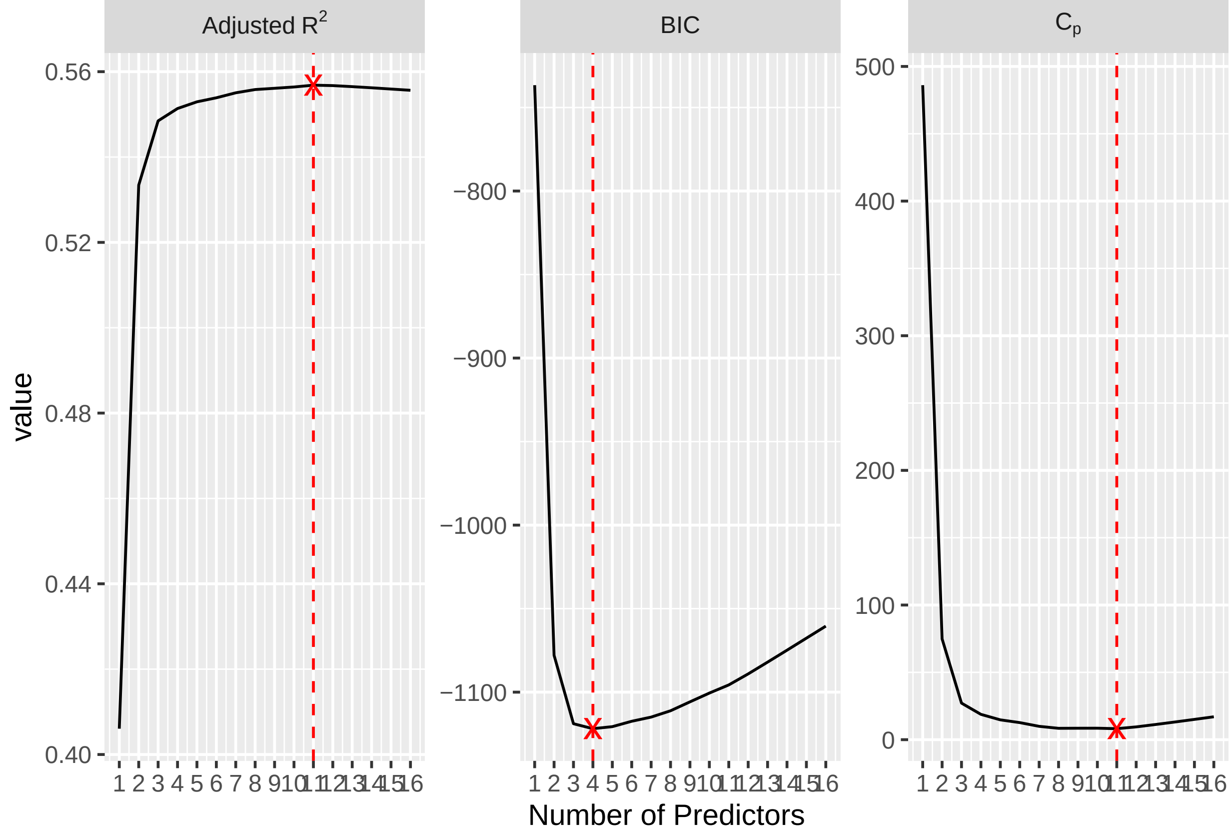
```
##    ScoreOff  RushAttOff  RushYdsOff  PassAttOff PassCompOff  PassYdsOff
##    7.886399    5.613167    8.732710   21.807444   20.636451   13.340274
##  PassIntOff  FumblesOff    ScoreDef  RushAttDef  RushYdsDef  PassAttDef
##    1.760629    1.226930    7.988964    3.606800    7.273741    9.980231
## PassCompDef  PassYdsDef  PassIntDef  FumblesDef
##    8.752594    6.579129    1.658463    1.380966
```

We plot the correlation graph of our 16 potential predictors to check if we need to take out possible highly related predictors. (The bigger the circle implies higher correction between the two predictors that have their row and column intersect at that circle) Most of the predictors look in shape, so we will keep all predictors for further subset selection process.

2. Determining number of predictors to include in our model.

2a. Use the regsubsets function from leaps package to perform best subset selection in order to choose the best model containing our 23 predictors according to Cp, BIC, and adjusted R2.

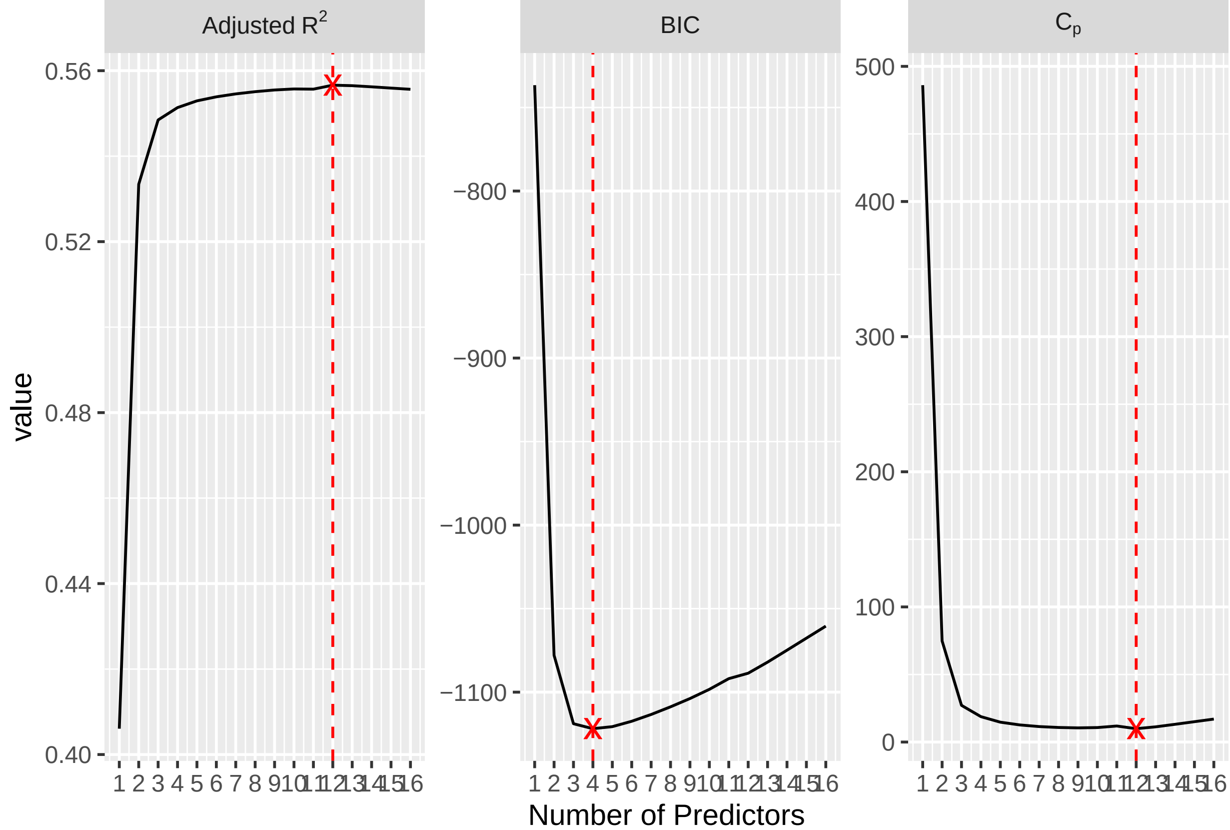
```r
# create datasets for plots
### criteria labels (in plotmath, see expression syntax in ?plotmath)
degree <- 16
model.regsubsets <- summary(regsubsets(Wins ~ ., data = cfb_data, nvmax = degree))

criteria.plotmath <- c(
cp = "C[p]",
bic = "BIC",
adjr2 = "Adjusted~R^2"
)
criteria_names <- c("cp", "bic", "adjr2")
data_plot <- model.regsubsets[criteria_names] %>% data.frame(size = 1:degree) %>%
  gather(cp, bic, adjr2, key = "criteria", value = "value") %>% mutate(criteria_label = criteria.plotma
data_best <- data_plot %>% group_by(criteria) %>% # min of Cp, BIC; max of Adjusted R^2
top_n(1, ifelse(criteria == "adjr2", value, - value))
# generate plots of criteria with respect to the number of predictors
data_plot %>%
ggplot(aes(x = size, y = value)) +
geom_line() +
geom_point(data = data_best, colour = "red", shape = "x", size = 5) +
geom_vline(data = data_best, aes(xintercept = size), colour = "red", linetype = "dashed") +
scale_x_continuous(name = "Number of Predictors", breaks = 1:degree) +
facet_wrap(~ criteria_label, scales = "free_y", labeller = label_parsed)
```



According to Adjusted R^2, our optimal model would have 11 predictors.
According to BIC, our optimal model would have 4 predictors.
According to Mallow's Cp, our optimal model would have 11 predictors.
The optimal number of variables suggested by Adjusted R^2 and Mallow's Cp are both 11. We decide to seek for stronger conclusion considering forward stepwise selection and also using backwards stepwise selection.

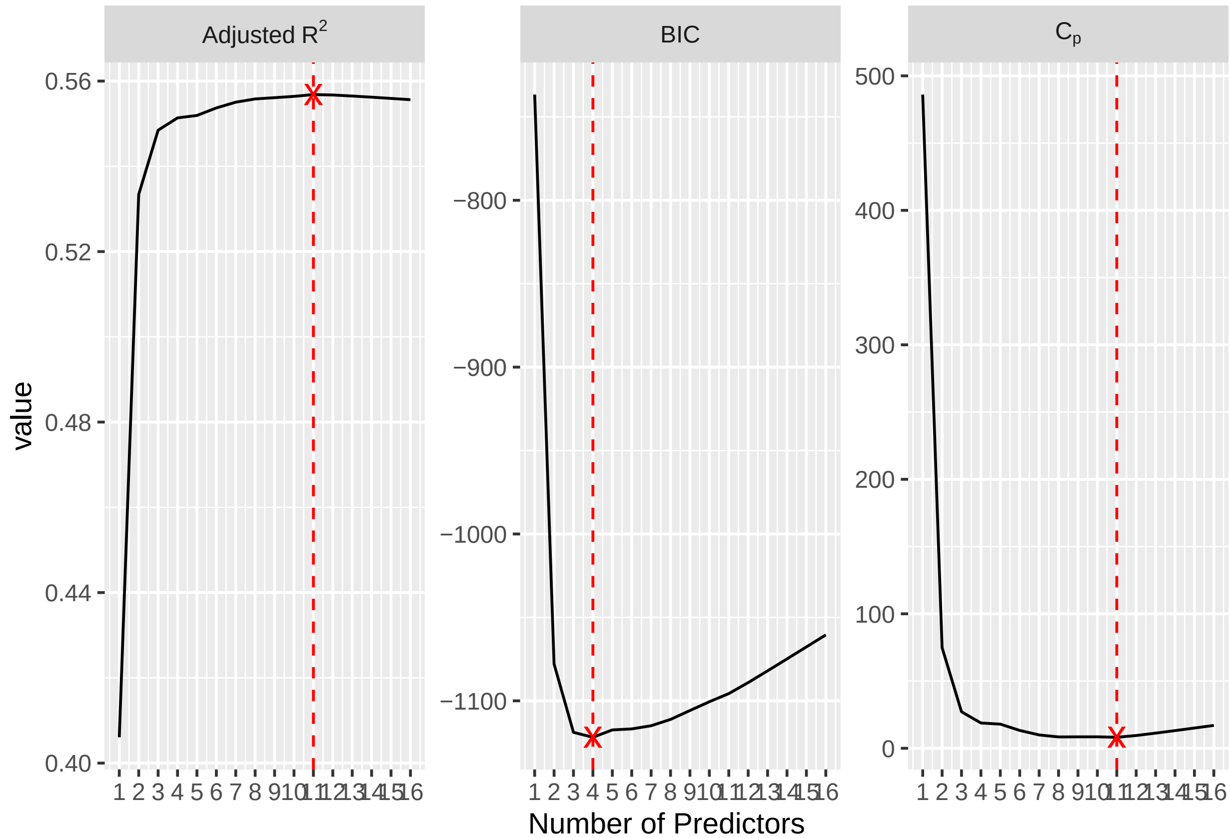
2b. Stepwise selection i) Forward Stepwise Selection Method

```r
model.regsubsets <- regsubsets(Wins ~ ., data = cfb_data,
method = "forward",
nvmax = degree) %>% summary()

data_plot <- model.regsubsets[criteria_names] %>% data.frame(size = 1:degree) %>%
  gather(cp, bic, adjr2, key = "criteria", value = "value") %>% mutate(criteria_label = criteria.plotma
data_best <- data_plot %>% group_by(criteria) %>% # min of Cp, BIC; max of Adjusted R^2
top_n(1, ifelse(criteria == "adjr2", value, - value))
# generate plots of criteria with respect to the number of predictors
data_plot %>%
ggplot(aes(x = size, y = value)) +
geom_line() +
geom_point(data = data_best, colour = "red", shape = "x", size = 5) +
geom_vline(data = data_best, aes(xintercept = size), colour = "red", linetype = "dashed") +
scale_x_continuous(name = "Number of Predictors", breaks = 1:degree) +
facet_wrap(~ criteria_label, scales = "free_y", labeller = label_parsed)
```



According to Adjusted R^2, our optimal model would have 11 predictors.
According to BIC, our optimal model would have 4 predictors.
According to Cp, our optimal model would have 11 predictors.
The results concluded from forward stepwise method are the same to the results from best subset method.
b) Backward Stepwise Selection Method

```
model.regsubsets <- regsubsets(Wins ~ ., data = cfb_data,
method = "backward",
nvmax = degree) %>% summary()

data_plot <- model.regsubsets[criteria_names] %>% data.frame(size = 1:degree) %>%
  gather(cp, bic, adjr2, key = "criteria", value = "value") %>% mutate(criteria_label = criteria.plotma
data_best <- data_plot %>% group_by(criteria) %>% # min of Cp, BIC; max of Adjusted R^2
top_n(1, ifelse(criteria == "adjr2", value, - value))
# generate plots of criteria with respect to the number of predictors
data_plot %>%
ggplot(aes(x = size, y = value)) +
geom_line() +
geom_point(data = data_best, colour = "red", shape = "x", size = 5) +
geom_vline(data = data_best, aes(xintercept = size), colour = "red", linetype = "dashed") +
scale_x_continuous(name = "Number of Predictors", breaks = 1:degree) +
facet_wrap(~ criteria_label, scales = "free_y", labeller = label_parsed)
```



According to Adjusted R^2, our optimal model would have 11 predictors.
According to BIC, our optimal model would have 4 predictors.
According to Cp, our optimal model would have 11 predictors.

COnsidering the results from all three subset selection methods, we would like to use Adjusted R^2 and Mallow's Cp value as the most important criterion since they both suggest that 11 is the optimal number of predictors.
Hence, we come to a conclusion that we will include 11 predictors for predicting the response "Wins" for an College Football team based on its season statistics from the College Football datasets.

3. Fitting our Model and Testing its Performance

a) Model with 11 variables obtained from Best Subset Selection

```
all = regsubsets(Wins~., cfb_data, nbest = 2, nvmax = 11)
ShowSubsets(all)
```

```
##             ScoreOff RushAttOff RushYdsOff PassAttOff PassCompOff PassYdsOff
## 1  ( 1 )        *
## 1  ( 2 )
## 2  ( 1 )        *
## 2  ( 2 )        *
## 3  ( 1 )        *
## 3  ( 2 )        *
## 4  ( 1 )        *
```

```
## 4  ( 2 )         *
## 5  ( 1 )         *
## 5  ( 2 )         *          *
## 6  ( 1 )         *          *
## 6  ( 2 )         *                                *           *
## 7  ( 1 )         *                                *           *
## 7  ( 2 )         *          *
## 8  ( 1 )         *                                *           *
## 8  ( 2 )         *                                *           *
## 9  ( 1 )         *                                *           *
## 9  ( 2 )         *          *                     *           *
## 10  ( 1 )        *          *                     *           *
## 10  ( 2 )        *          *          *          *           *
## 11  ( 1 )        *          *          *          *           *
## 11  ( 2 )        *          *                     *           *
##           PassIntOff FumblesOff ScoreDef RushAttDef RushYdsDef PassAttDef
## 1  ( 1 )
## 1  ( 2 )                               *
## 2  ( 1 )                               *
## 2  ( 2 )                                                     *
## 3  ( 1 )                               *                                *
## 3  ( 2 )                               *
## 4  ( 1 )                               *                                *
## 4  ( 2 )                               *                                *
## 5  ( 1 )                               *                                *
## 5  ( 2 )                               *                                *
## 6  ( 1 )                               *                                *
## 6  ( 2 )                               *                                *
## 7  ( 1 )                               *                                *
## 7  ( 2 )                      *        *                                *
## 8  ( 1 )                               *                     *          *
## 8  ( 2 )                      *        *                                *
## 9  ( 1 )                      *        *                     *          *
## 9  ( 2 )                               *                     *          *
## 10  ( 1 )                     *        *                     *          *
## 10  ( 2 )                              *                     *          *
## 11  ( 1 )                     *        *                     *          *
## 11  ( 2 )                     *        *                     *          *
##           PassCompDef PassYdsDef PassIntDef FumblesDef   Rsq adjRsq     Cp
## 1  ( 1 )                                               40.65  40.61 486.07
## 1  ( 2 )                                               26.23  26.18 953.02
## 2  ( 1 )                                               53.41  53.34  74.82
## 2  ( 2 )                                               48.81  48.74 223.77
## 3  ( 1 )                                               54.94  54.85  27.15
## 3  ( 2 )            *                                  54.46  54.36  42.81
## 4  ( 1 )                                  *            55.26  55.14  18.80
## 4  ( 2 )                                             * 55.12  55.00  23.35
## 5  ( 1 )                                  *          * 55.45  55.29  14.72
## 5  ( 2 )                                  *            55.38  55.23  16.79
## 6  ( 1 )                                  *          * 55.57  55.39  12.67
## 6  ( 2 )                                  *            55.56  55.37  13.27
## 7  ( 1 )                                  *          * 55.72  55.50   9.91
## 7  ( 2 )                                  *          * 55.67  55.46  11.46
## 8  ( 1 )                                  *          * 55.83  55.58   8.44
```

```
## 8  ( 2 )                                *          * 55.79  55.55   9.57
## 9  ( 1 )                                *          * 55.89  55.61   8.51
## 9  ( 2 )                                *          * 55.87  55.59   9.10
## 10  ( 1 )                               *          * 55.95  55.64   8.51
## 10  ( 2 )                               *          * 55.94  55.63   8.71
## 11  ( 1 )                               *          * 56.02  55.68   8.16
## 11  ( 2 )          *                    *          * 55.98  55.65   9.36
```

The best subset method yields a result of 11 best predictors: ScoreOff, RushAttOff, RushYdsOff, PassAttOff, PassCompOff, FumblesOff, ScoreDef, RushYdsDef, PassAttDef, PassIntDef and FumblesDef.

```r
cfb_mod1 = lm(Wins~ScoreOff + RushAttOff + RushYdsOff + PassAttOff + PassCompOff + FumblesOff + ScoreDe
summary(cfb_mod1)
```

```
##
## Call:
## lm(formula = Wins ~ ScoreOff + RushAttOff + RushYdsOff + PassAttOff +
##     PassCompOff + FumblesOff + ScoreDef + RushYdsDef + PassAttDef +
##     PassIntDef + FumblesDef, data = cfb_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.6558 -1.0402 -0.0206  1.0349  4.9855
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.3961435  0.5597496   4.281 1.99e-05 ***
## ScoreOff     0.0112604  0.0009373  12.014  < 2e-16 ***
## RushAttOff   0.0024546  0.0011776   2.084  0.03730 *
## RushYdsOff  -0.0002631  0.0001713  -1.536  0.12484
## PassAttOff  -0.0044841  0.0017803  -2.519  0.01189 *
## PassCompOff  0.0056807  0.0025558   2.223  0.02639 *
## FumblesOff  -0.0190055  0.0118873  -1.599  0.11009
## ScoreDef    -0.0103028  0.0009173 -11.232  < 2e-16 ***
## RushYdsDef   0.0002596  0.0001465   1.772  0.07668 .
## PassAttDef   0.0064052  0.0010710   5.980 2.81e-09 ***
## PassIntDef   0.0384746  0.0117450   3.276  0.00108 **
## FumblesDef   0.0069604  0.0028641   2.430  0.01521 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.526 on 1428 degrees of freedom
## Multiple R-squared:  0.5602, Adjusted R-squared:  0.5568
## F-statistic: 165.4 on 11 and 1428 DF,  p-value: < 2.2e-16
```

The predicted value of Wins = 2.39614 + 0.01126*ScoreOff + 0.002455*RushAttOff - 0.0002631*RushYdsOff - 0.0044841*PassAttOff + 0.005681*PassCompOff - -0.019001*FumblesOff - 0.010303*ScoreDef + 0.0002596*RushYdsDef + 0.0064052*PassAttDef + 0.0384746*PassIntDef + 0.00696*FumblesDef.

```r
kfold.cv.lm <- function(X, y, which.betas = rep(TRUE, ncol(X)), k = 10, seed = 0) {
X <- X[,which.betas]
data <- data.frame(X, y)
n <- nrow(data)
MSEs <- MSPEs <- rep(0, k)
set.seed(seed)
ids_fold <- cut(sample(n), breaks = k, labels = 1:k)
for(fold in 1:k) {
data_in <- subset(data, fold != ids_fold)
data_out <- subset(data, fold == ids_fold)
model <- lm(y ~ ., data = data_in)
data_in$pred <- predict(model)
data_out$pred <- predict(model, newdata = data_out)
MSEs[fold] <- with(data_in, mean((y - pred)^{2}))
MSPEs[fold] <- with(data_out, mean((y - pred)^{2}))
}
return(c(Avg.MSE = mean(MSEs), Avg.MSPE = mean(MSPEs)))
}

X <- select(cfb_data, - Wins) %>% as.matrix
y <- cfb_data$Wins
# full model
which1 <- rep(TRUE, ncol(X))
seed <- 527
kfold1 <- kfold.cv.lm(X, y, which1, 10, seed)
cat(paste0(paste(rep("#", 80), collapse = ""), "\n"))
```

```
## ################################################################################
```

```r
cat("##### Our Model:\n")
```

```
## ##### Our Model:
```

```r
kfold1
```

```
##  Avg.MSE Avg.MSPE
## 2.302953 2.362195
```

Using seed 527, we used cross validation to test the performance of our model with 11 variables obtained with best subset method. The Average Mean Squared Prediction Error (Avg.MSPE) is 2.362195.
b) 5-fold LASSO Regression and Performance
We also performed LASSO Regression on our model because LASSO tends to generate models with high power of prediction despite its lack of simplicity. However, since the dimension of our dataset is not so large, we would sacrifice some simplicity for greater accuracy.

```r
library(glmnet)
```

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

```
##
## Attaching package: 'Matrix'
```

```
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack

## Loaded glmnet 4.0-2
```

```
seed <- 527
predictors <- select(cfb_data, - Wins) %>% as.matrix()
response <- cfb_data$Wins
model.ridge_CV <- cv.glmnet(x = predictors, y = response, nfolds = 10, alpha = 1)
# reporting the lambda with minimal CV-MSE
with(model.ridge_CV, data.frame(lambda = lambda, CV_MSE = cvm)) %>%
top_n(1, - CV_MSE) # minimal CV-MSE
```

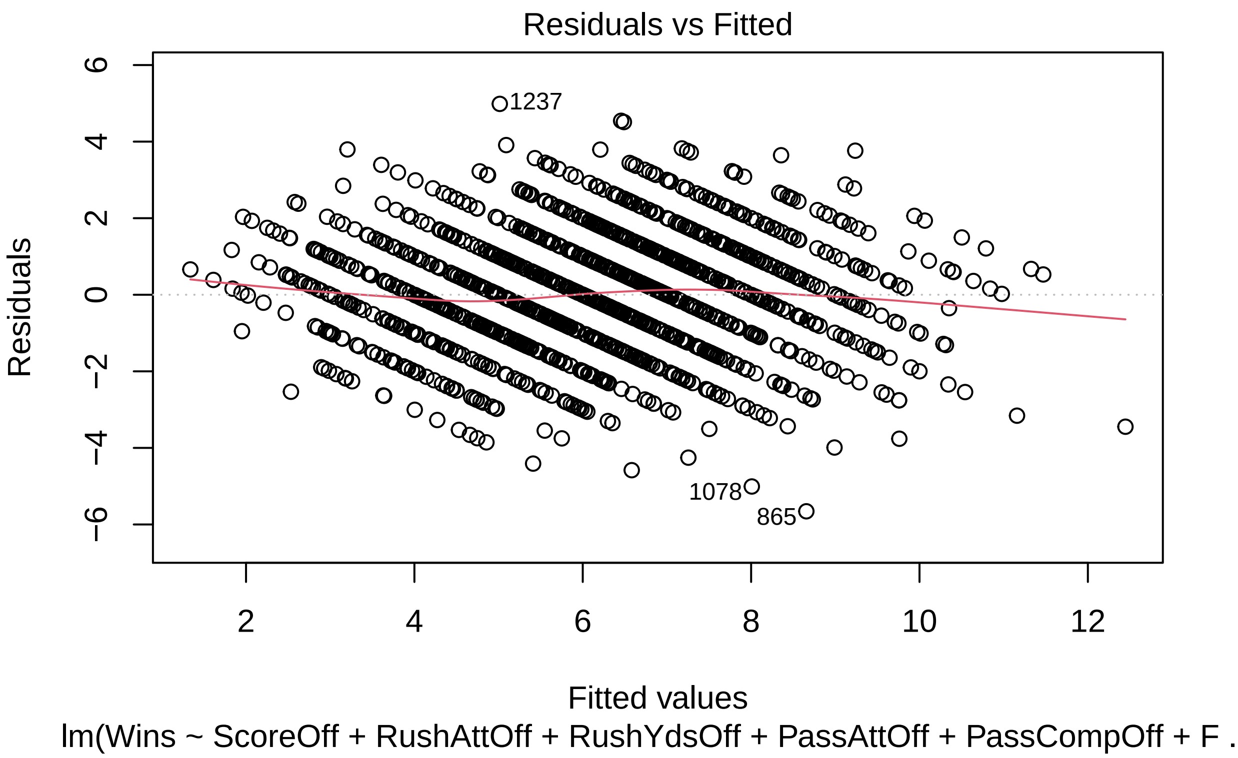
```
##       lambda   CV_MSE
## 1 0.02436196 2.363092
```

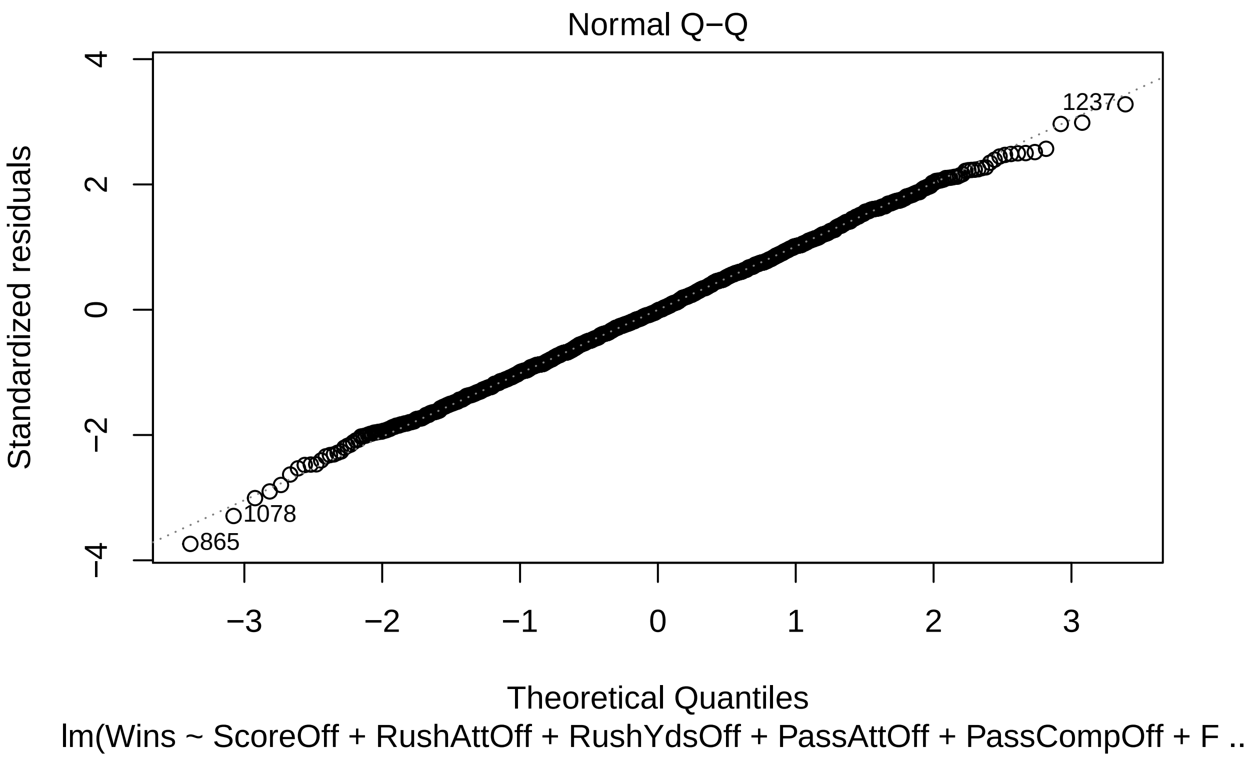
```
betas <- coef(model.ridge_CV, s = "lambda.min") %>% as.numeric()
betas
```

```
##  [1]  2.8495687952  0.0113769631  0.0009087268  0.0000000000  0.0000000000
##  [6]  0.0000000000  0.0000000000 -0.0046248797 -0.0144783827 -0.0092103333
## [11]  0.0000000000  0.0000000000  0.0050355900  0.0000000000  0.0000000000
## [16]  0.0370165902  0.0056100276
```

This model has an Avg.MSPE of 2.363092, which is basically the same as the previous model.

```
plot(cfb_mod1, c(1,2))
```


Residuals vs Fitted
1237
1078
865
Residuals
Fitted values
lm(Wins ~ ScoreOff + RushAttOff + RushYdsOff + PassAttOff + PassCompOff + F ..
6
4
2
0
−2
−4
−6
2
4
6
8
10
12



Since the conditions of linearity seem very well qualified from the summary plot of our model, we will keep the linear model with predictors predicted by best subset selection.
Therefore, our final model for predicting the “Wins” of an CFB team is Wins = 2.39614 + 0.01126*ScoreOff + 0.002455*RushAttOff - 0.0002631*RushYdsOff - 0.0044841*PassAttOff + 0.005681*PassCompOff - -0.019001*FumblesOff - 0.010303*ScoreDef + 0.0002596*RushYdsDef + 0.0064052*PassAttDef + 0.0384746*PassIntDef + 0.00696*FumblesDef.