NFL: Predicting 'Wins'

```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
```

```r
#x <- getURL("https://raw.githubusercontent.com/WeiKangda/Data-Challenge-Football/main/FootballDatasets,
#NFL2000Full <- read.csv(text = x)
#head(NFL2000Full)
nfl_all_data <- read.csv("nfl_all_data.csv")
head(nfl_all_data)
```

```
##   Year          TeamName Wins ScoreOff FirstDown RushAttOff RushYdsOff
## 1 2000 Arizona Cardinals    9      178       253        342       1284
## 2 2000   Atlanta Falcons    7      238       256        350       1214
## 3 2000  Baltimore Ravens   12      355       319        619       2480
## 4 2000     Buffalo Bills    8      288       309        476       1921
## 5 2000 Carolina Panthers    7      272       304        363       1186
## 6 2000     Chicago Bears    9      216       238        416       1736
##   PassAttOff PassCompOff PassYdsOff PassIntOff FumblesOff SackYdsOff PenYdsOff
## 1        554         316       3478         24         20        239       756
## 2        515         285       3166         20         14        386       720
## 3        553         309       3539         20          8        349       905
## 4        546         312       3936         10         12        359       913
## 5        566         340       3850         19         16        382       683
## 6        542         304       3005         16         13        206       696
##   PuntAvgOff ScoreDef FirstDownDef RushAttDef RushYdsDef PassAttDef PassCompDef
## 1        710      443          344        580       2609        458         295
## 2        654      413          308        453       1983        515         306
## 3        741      181          260        430       1162        650         357
## 4        610      350          252        444       1559        480         283
## 5        607      310          304        425       1949        552         352
## 6        593      355          297        469       1828        530         332
##   PassYdsDef PassIntDef FumblesDef SackYdsDef PenYdsDef
## 1       3263         10         10        126        32
## 2       3766         15         10        142        14
## 3       3735         29         27        245        39
## 4       3175         16         13        308        27
## 5       3938         17         21        231        38
## 6       3635         11          9        231         0
```

```r
#x <- getURL("https://raw.githubusercontent.com/WeiKangda/Data-Challenge-Football/main/FootballDatasets,
#CFB2003Full <- read.csv(text = x)
#CFB2003Full <- subset(CFB2003Full, ScoreOff != 0)
#head(CFB2003Full)
```

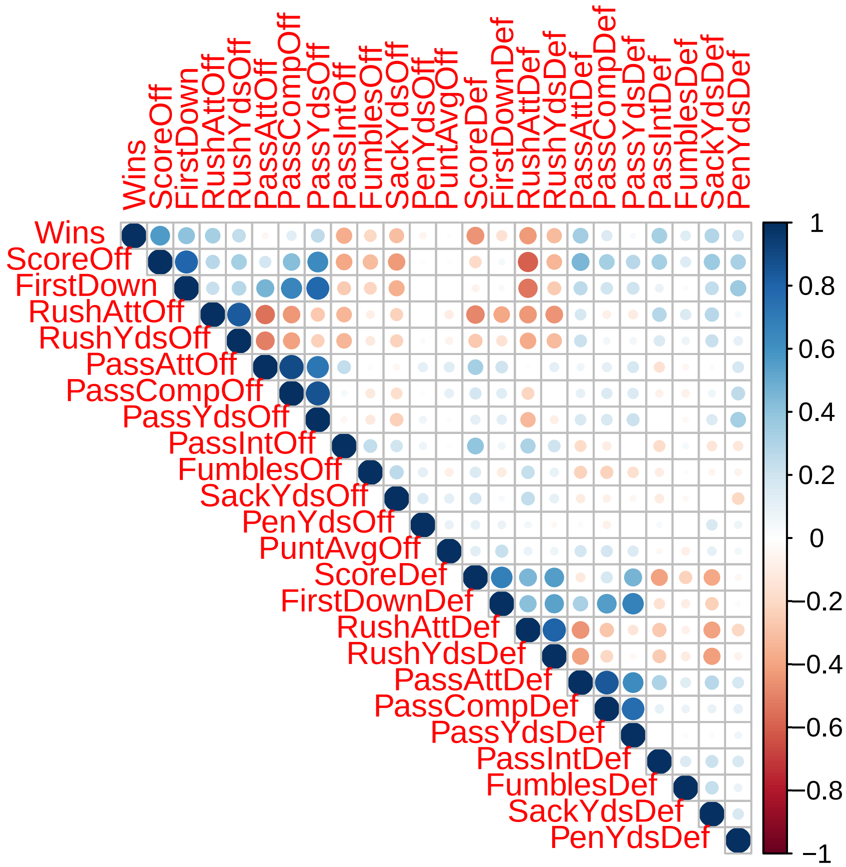
I) Predicting wins of an NFL team

1. Correlation between predictors and Full Model with all predictors

```r
#nfl_tr = subset(nfl_all_data, Year != 2013)[3:26]
#nfl_te = subset(nfl_all_data, Year == 2013)[3:26]

nfl_data = nfl_all_data[3:26]

Fullnfl = lm(Wins~., nfl_data)
summary(Fullnfl)
```

```
##
## Call:
## lm(formula = Wins ~ ., data = nfl_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.3121 -1.1823 -0.0084  1.0205  5.2621
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   1.069e+00  3.013e+00   0.355 0.722925
## ScoreOff      1.396e-02  3.175e-03   4.398 1.38e-05 ***
## FirstDown     7.533e-03  8.183e-03   0.921 0.357808
## RushAttOff    6.002e-03  4.040e-03   1.486 0.138114
## RushYdsOff   -1.440e-03  5.703e-04  -2.526 0.011916 *
## PassAttOff   -3.551e-03  4.727e-03  -0.751 0.452951
## PassCompOff   6.511e-05  6.586e-03   0.010 0.992117
## PassYdsOff   -1.269e-04  4.311e-04  -0.294 0.768607
## PassIntOff   -2.120e-02  2.236e-02  -0.948 0.343629
## FumblesOff    2.516e-02  2.556e-02   0.984 0.325435
## SackYdsOff   -2.083e-03  1.388e-03  -1.501 0.134188
## PenYdsOff     3.721e-05  6.486e-04   0.057 0.954279
## PuntAvgOff    1.917e-03  1.972e-03   0.972 0.331569
## ScoreDef     -1.295e-02  3.320e-03  -3.899 0.000112 ***
## FirstDownDef -3.233e-03  9.150e-03  -0.353 0.724040
## RushAttDef    1.037e-03  4.679e-03   0.222 0.824800
## RushYdsDef    9.407e-04  6.139e-04   1.532 0.126188
## PassAttDef    1.072e-02  4.977e-03   2.154 0.031798 *
## PassCompDef  -6.487e-03  6.467e-03  -1.003 0.316378
## PassYdsDef    4.217e-04  4.557e-04   0.925 0.355337
## PassIntDef    2.571e-03  2.069e-02   0.124 0.901172
## FumblesDef   -1.597e-02  2.617e-02  -0.610 0.541953
## SackYdsDef   -1.464e-04  1.798e-03  -0.081 0.935136
## PenYdsDef    -2.977e-03  8.905e-03  -0.334 0.738301
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.718 on 422 degrees of freedom
## Multiple R-squared:  0.4684, Adjusted R-squared:  0.4394
## F-statistic: 16.17 on 23 and 422 DF,  p-value: < 2.2e-16
```

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




```
vif(Fullnfl)
```

```
##     ScoreOff    FirstDown   RushAttOff   RushYdsOff   PassAttOff  PassCompOff
##     7.079632    14.051817     7.226508     5.867960    11.872236    14.434912
##   PassYdsOff   PassIntOff   FumblesOff   SackYdsOff    PenYdsOff  PuntAvgOff
##     9.824538     1.714364     1.391552     1.529516     1.423398     1.221156
##     ScoreDef FirstDownDef   RushAttDef   RushYdsDef   PassAttDef  PassCompDef
##     6.041174    10.594835     7.062549     5.375413     8.293251     7.750347
##   PassYdsDef   PassIntDef   FumblesDef   SackYdsDef    PenYdsDef
##     4.786737     1.649307     1.410333     1.656283     1.334503
```

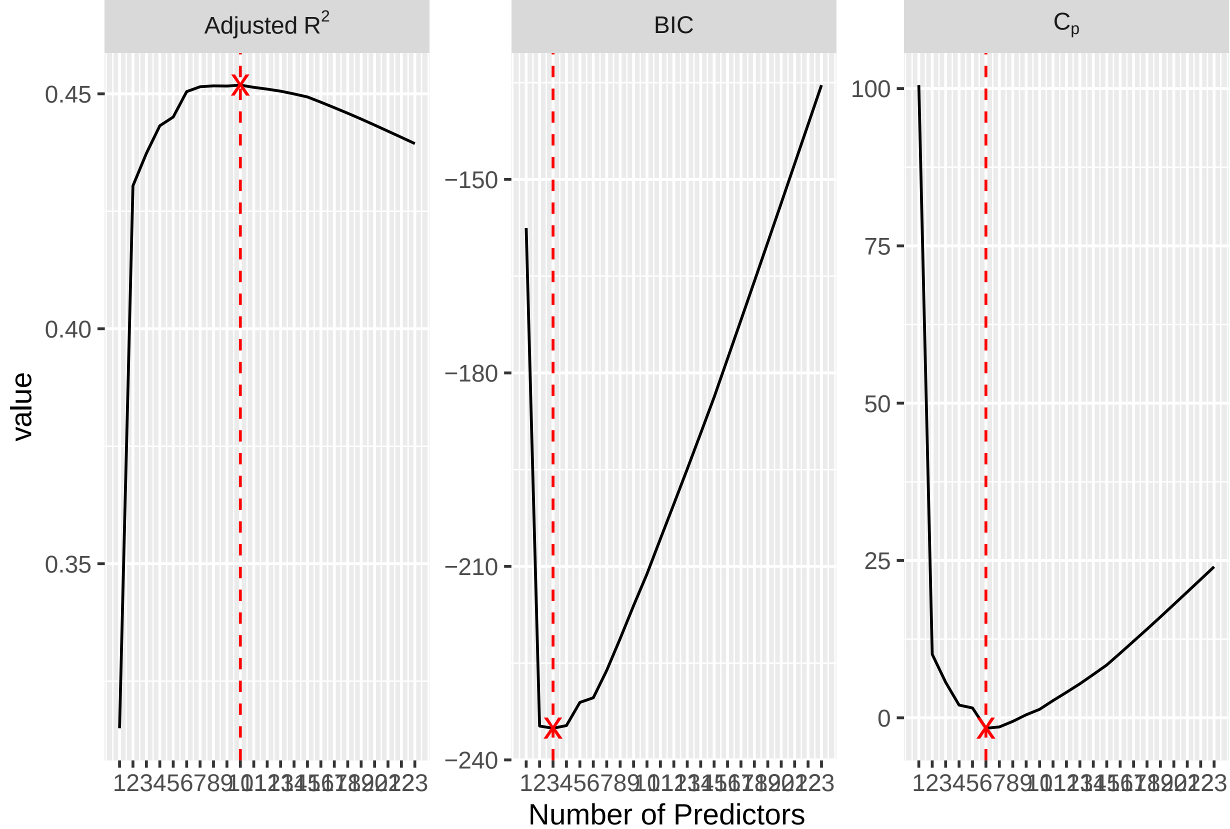
```
#cfb_tr = subset(cfb_all_data, Year != 2013)[3:19]
#cfb_te = subset(cfb_all_data, Year == 2013)[3:19]

#Fullcfb = lm(Wins~., cfb_tr)
#summary(Fullcfb)

#corr = cor(cfb_tr, use="pairwise.complete.obs")
#corrplot(corr, type="upper")

#vif(Fullcfb)
```

We plot the correlation graph of our 23 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.

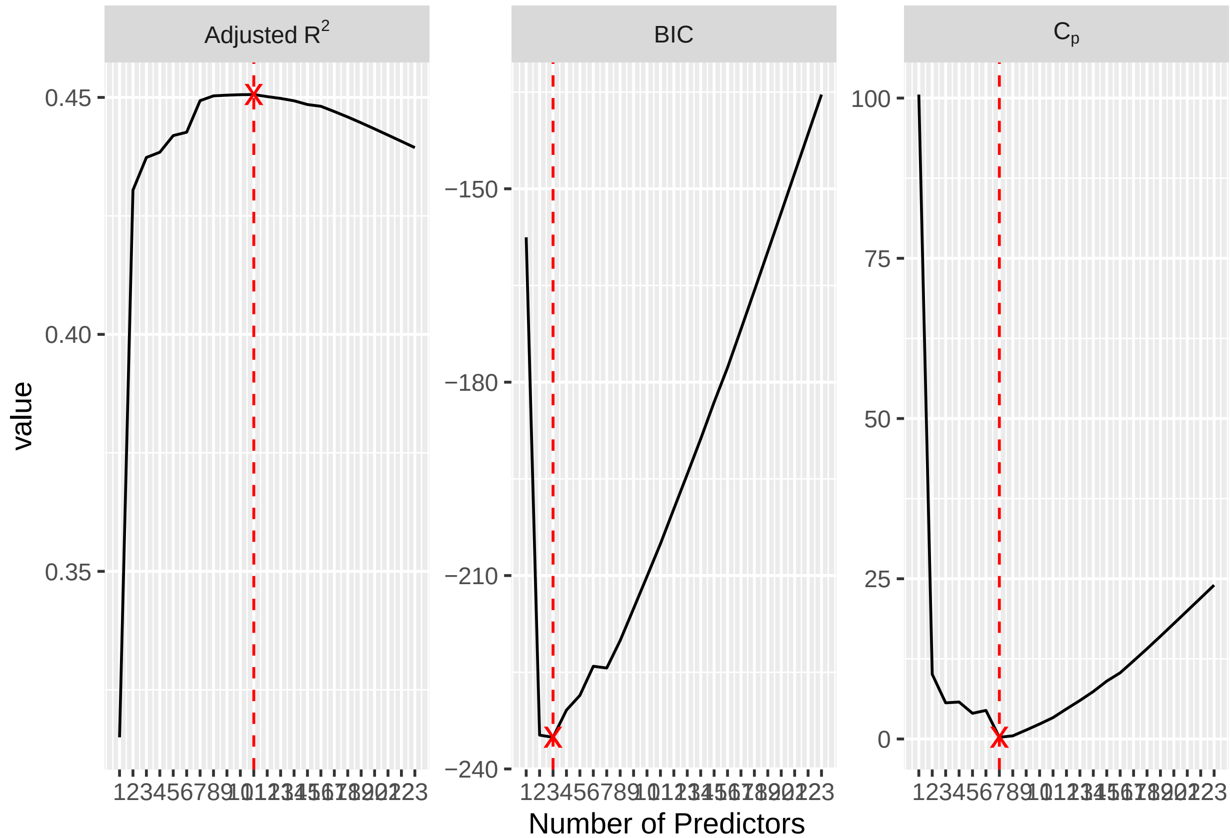
```
# create datasets for plots
### criteria labels (in plotmath, see expression syntax in ?plotmath)
degree <- 23
model.regsubsets <- summary(regsubsets(Wins ~ ., data = nfl_data, nvmax = degree))

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.plotmat
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 10 predictors.
According to BIC, our optimal model would have 3 predictors.
According to Cp, our optimal model would have 6 predictors.
Since these subset selection process according to these three criteria do not quite match with each other, we decide to seek for better conclusion considering forward stepwise selection and also using backwards stepwise selection.

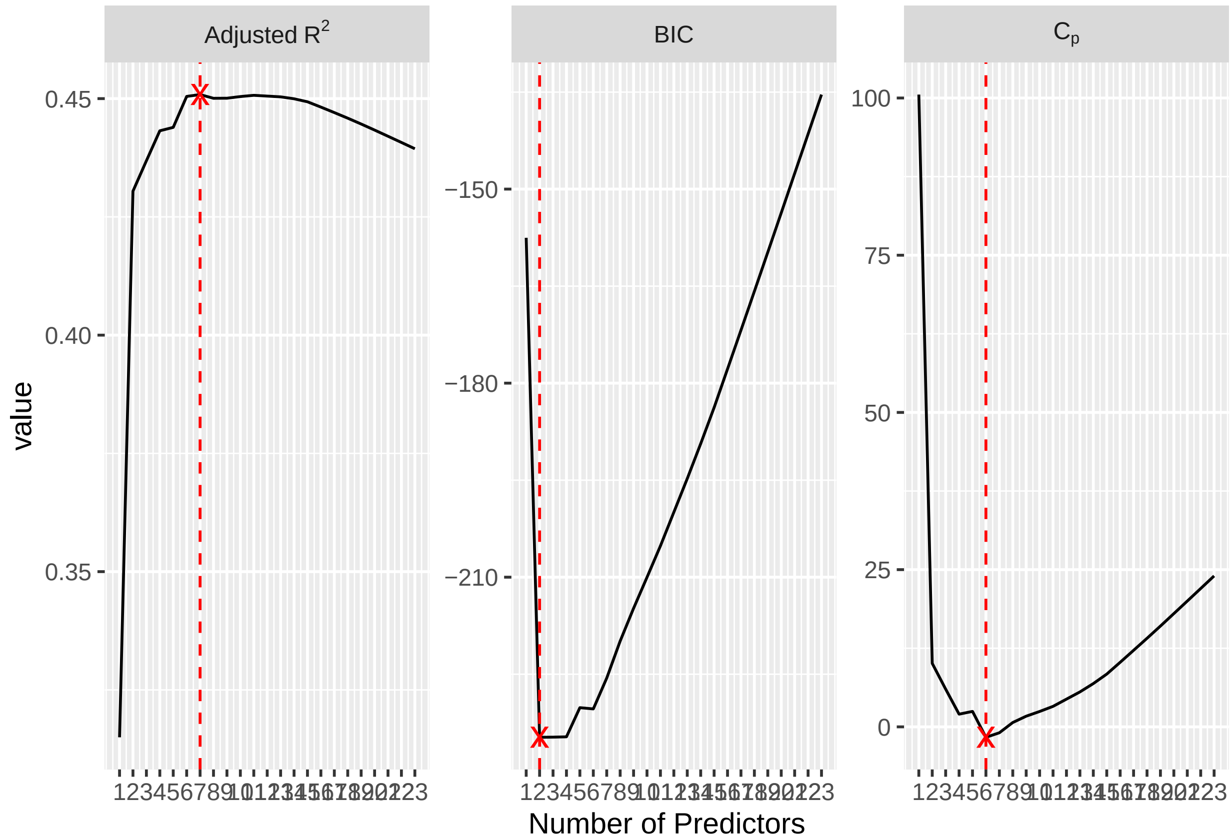
2b. Stepwise selection i) Forward Stepwise Selection Method

```
model.regsubsets <- regsubsets(Wins ~ ., data = nfl_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.plotmat
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 10 predictors.
According to BIC, our optimal model would have 3 predictors.
According to Cp, our optimal model would have 7 predictors.
The results concluded from forward stepwise method are similar to the results from best subset method.
b) Backward Stepwise Selection Method

```
model.regsubsets <- regsubsets(Wins ~ ., data = nfl_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 7 predictors.
According to BIC, our optimal model would have 2 predictors.
According to Cp, our optimal model would have 6 predictors.

COnsidering the results from all three subset selection methods, we would like to use Mallow's Cp value as the most important criterion for the following reasons:
1) The optimal number of predictors considering Cp stays quite stable (6, 7, 6) 2) The value of other criteria when the number of predictors is around 6-7 does not relatively vary significantly from their critical points, whereas Mallow's Cp varies much away from the critical point near at 6 or 7.
Hence, we come to a conclusion that we will include 6 predictors for predicting the response "Wins" for an NFL team based on its season statistics from the NFL datasets.

3. Fitting our Model and Testing its Performance

a) Model with 7 variables obtained from Best Subset Selection

```
all = regsubsets(Wins~., nfl_data, nbest = 2, nvmax = 7)
ShowSubsets(all)
```

```
##          ScoreOff FirstDown RushAttOff RushYdsOff PassAttOff PassCompOff
## 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 )        *                   *          *
##          PassYdsOff PassIntOff FumblesOff SackYdsOff PenYdsOff PuntAvgOff
## 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 )                    *
##          ScoreDef FirstDownDef RushAttDef RushYdsDef PassAttDef PassCompDef
## 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 )        *                                     *          *
##          PassYdsDef PassIntDef FumblesDef SackYdsDef PenYdsDef   Rsq adjRsq
## 1  ( 1 )                                                       31.65  31.50
## 1  ( 2 )                                                       19.97  19.79
## 2  ( 1 )                                                       43.30  43.04
## 2  ( 2 )                                                       35.07  34.78
## 3  ( 1 )                                                       44.11  43.73
## 3  ( 2 )                                                       44.07  43.69
## 4  ( 1 )                                                       44.82  44.32
## 4  ( 2 )                                                       44.45  43.95
## 5  ( 1 )                                                       45.13  44.51
## 5  ( 2 )                                                       45.02  44.39
## 6  ( 1 )                                                       45.79  45.05
## 6  ( 2 )          *                                            45.38  44.63
## 7  ( 1 )                                *                      46.01  45.15
## 7  ( 2 )                                                       45.96  45.10
```

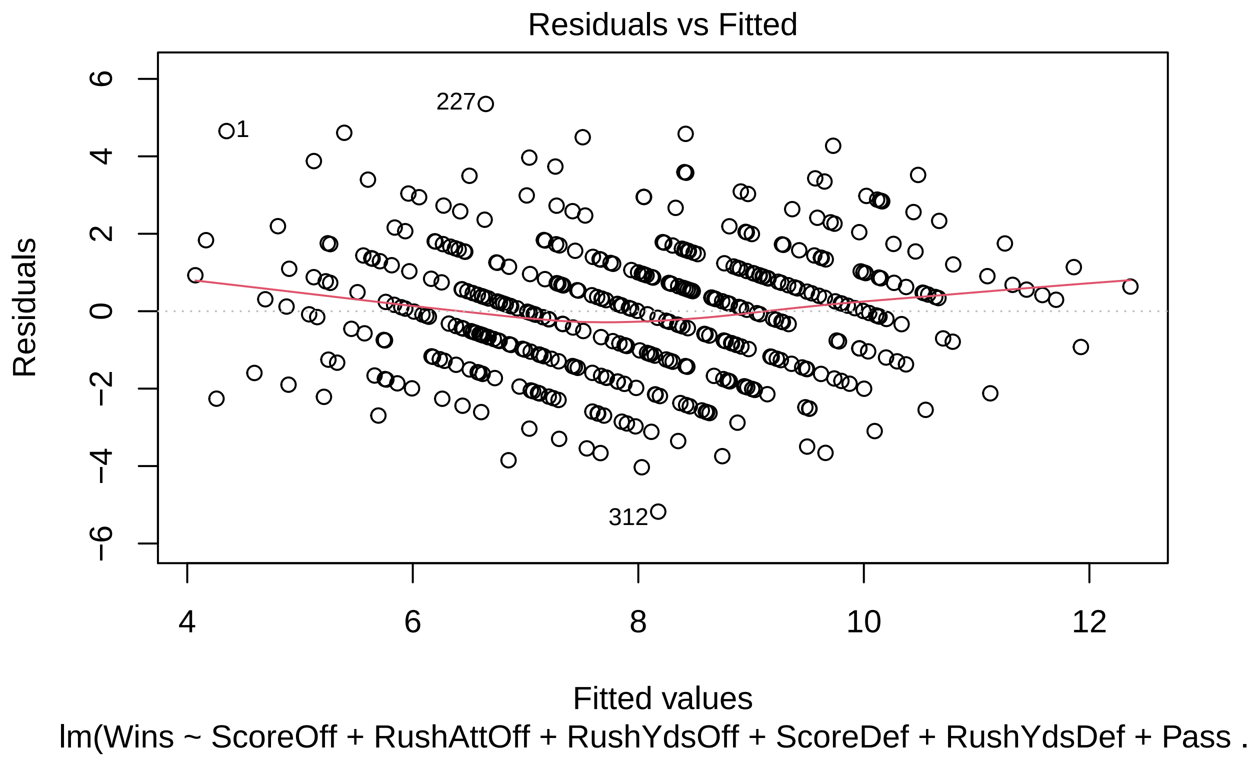
```
##              Cp
## 1  ( 1 ) 100.55
## 1  ( 2 ) 193.27
## 2  ( 1 )  10.09
## 2  ( 2 )  75.39
## 3  ( 1 )   5.64
## 3  ( 2 )   6.00
## 4  ( 1 )   2.02
## 4  ( 2 )   4.95
## 5  ( 1 )   1.56
## 5  ( 2 )   2.45
## 6  ( 1 )  -1.66
## 6  ( 2 )   1.60
## 7  ( 1 )  -1.45
## 7  ( 2 )  -1.06
```

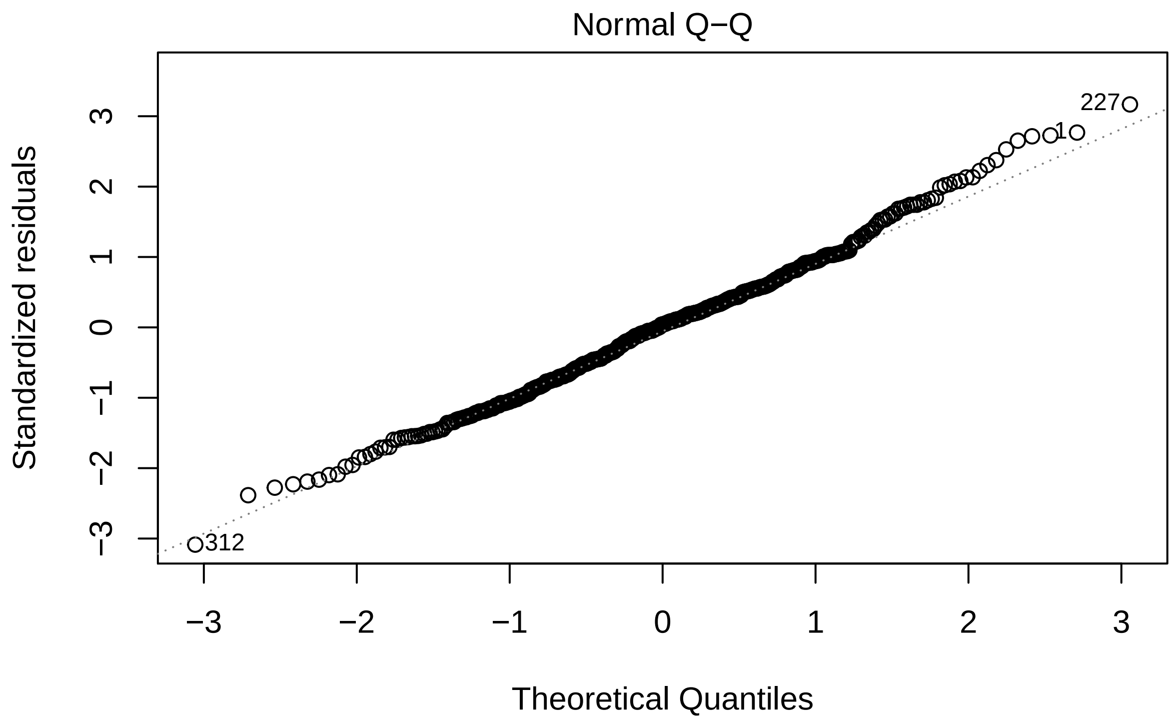
The best subset method yields a result of 7 best predictors: ScoreOff, RushAttOff, RushYdsOff, ScoreDef, RushYdsDef, PassAttDef, FumblesDef.

```r
nfl_mod1 = lm(Wins~ScoreOff + RushAttOff + RushYdsOff + ScoreDef + RushYdsDef + PassAttDef + FumblesDef
summary(nfl_mod1)
```

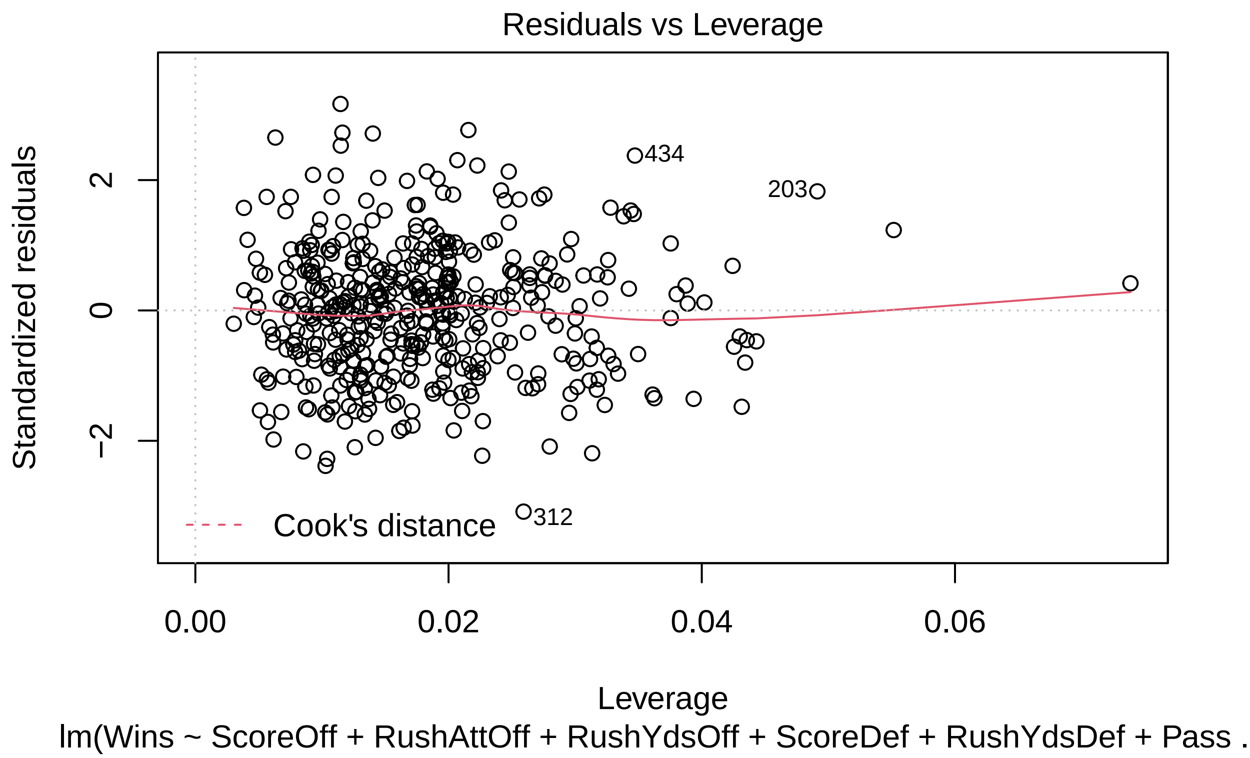
```
##
## Call:
## lm(formula = Wins ~ ScoreOff + RushAttOff + RushYdsOff + ScoreDef +
##     RushYdsDef + PassAttDef + FumblesDef, data = nfl_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.1766 -1.1823  0.0701  0.9977  5.3522
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.2484138  1.8230815   0.136 0.891678
## ScoreOff     0.0160735  0.0013999  11.482  < 2e-16 ***
## RushAttOff   0.0093808  0.0032289   2.905 0.003855 **
## RushYdsOff  -0.0012169  0.0004656  -2.614 0.009264 **
## ScoreDef    -0.0138864  0.0018090  -7.676 1.08e-13 ***
## RushYdsDef   0.0010504  0.0003572   2.941 0.003445 **
## PassAttDef   0.0072325  0.0020595   3.512 0.000491 ***
## FumblesDef  -0.0308390  0.0227994  -1.353 0.176874
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.699 on 438 degrees of freedom
## Multiple R-squared:  0.4601, Adjusted R-squared:  0.4515
## F-statistic: 53.33 on 7 and 438 DF,  p-value: < 2.2e-16
```

```r
plot(nfl_mod1, c(1,2,5))
```


Residuals vs Fitted
Residuals
Fitted values
lm(Wins ~ ScoreOff + RushAttOff + RushYdsOff + ScoreDef + RushYdsDef + Pass .
227
1
312
6
4
2
0
−2
−4
−6
4
6
8
10
12



lm(Wins ~ ScoreOff + RushAttOff + RushYdsOff + ScoreDef + RushYdsDef + Pass .



The predicted value of Wins = 0.24841 + 0.01607*ScoreOff + 0.009381*RushAttOff -0.0012169*RushYdsOff - 0.0138864*ScoreDef + 0.0010504*RushYdsDef + 0.0072325*PassAttDef - 0.0308390*FumblesDef.

```
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(nfl_data, - Wins) %>% as.matrix
y <- nfl_data$Wins
# full model
```

```r
which1 <- rep(TRUE, ncol(X))
seed <- 424
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
## 2.774465 3.148486
```

Using seed 424, we used cross validation to test the performance of our model with 7 variables obtained with best subset method. The Average Mean Squared Prediction Error (Avg.MSPE) is 3.148486.
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
```

```r
seed <- 424
predictors <- select(nfl_data, - Wins) %>% as.matrix()
response <- nfl_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 0.08683439 3.054585
```

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

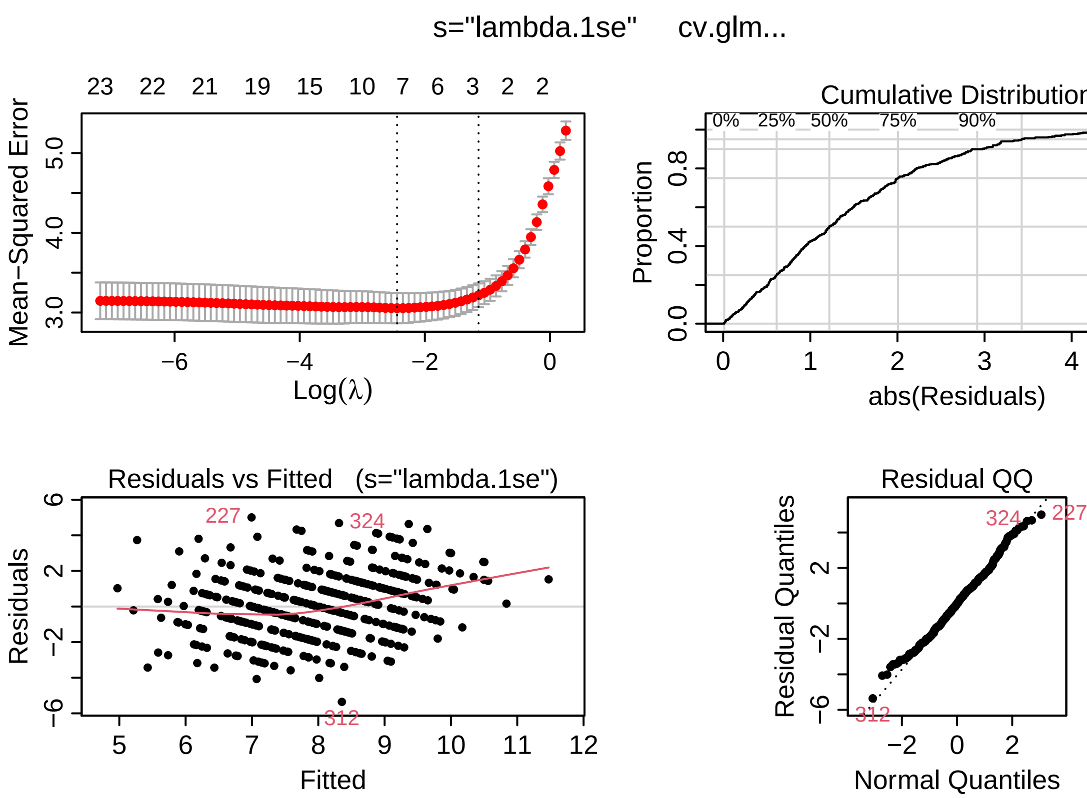
```
##  [1]  5.8390266653  0.0134088250  0.0000000000  0.0002563563  0.0000000000
##  [6]  0.0000000000  0.0000000000  0.0000000000 -0.0177875389  0.0000000000
## [11] -0.0009563607  0.0000000000  0.0000000000 -0.0110562491  0.0000000000
## [16]  0.0000000000  0.0000000000  0.0035465599  0.0000000000  0.0000000000
## [21]  0.0085213837  0.0000000000  0.0000000000  0.0000000000
```

The LASSO Regression yields a optimal lambda of 0.08683439 and gives the following model with also 7 variables: Wins = 5.8390266653 + 0.0134088250*ScoreOff + 0.0002564*RushAttOff - 0.0177875389*PassIntOff - 0.0009563607*SackYdsOff - 0.0110562491*ScoreDef + 0.0035465599*PassAttDef + 0.0085213837*PassInt-Def. This model has an Avg.MSPE of 3.054585, which is "better" than the previous model.

```r
library(glmnet)
library(plotmo) # for plotres
```

```
## Loading required package: Formula

## Loading required package: plotrix

## Loading required package: TeachingDemos
```

```r
plotres(model.ridge_CV)
```



As we observe in the LASSO model summary above, we find notice that although there's some curvature in

the Residuals vs Fitted plot, the overall linearity, zero-mean property, constant variance and normality of residuals are quite satisfied.
Therefore, our final model for predicting the "Wins" of an NFL team is Wins = 5.8390266653 + 0.0134088250*ScoreOff + 0.0002564*RushAttOff - 0.0177875389*PassIntOff - 0.0009563607*SackYdsOff - 0.0110562491*ScoreDef + 0.0035465599*PassAttDef + 0.0085213837*PassIntDef.