Blog 2: dropping predictors in Multiple Linear Regression

Michael O'Donnell

September 27, 2020

In Blog One I explored the metrics of Simple Linear Regression. Now, in this Blog I will dive into Multiple Linear Regression and determine how to drop predictor variables to achieve a higher .

To look at a linear model in R, let's use help

```r
help(lm)
```

```
## starting httpd help server ... done
```

Now, start by loading a dataset This dataset contains regular season data for all NBA teams from 2014-2018

```r
nbaData <- read.csv("data/nba_data.csv")
colnames(nbaData)[1] <- "Team"

head(nbaData, 3)
```

```
##             Team Season SeasonType Win Loss MatchCount WinPercentage
## 1  Atlanta Hawks   2018        REG  28   53         81     0.3456790
## 2 Boston Celtics   2018        REG  49   33         82     0.5975610
## 3  Brooklyn Nets   2018        REG  42   40         82     0.5121951
##      Pts OppPts   Pace OffEff DefEff EFgPercentage OppEFgPercentage
## 1 112.93 119.21 103.46 108.34 114.73         0.521            0.541
## 2 112.39 107.95  98.97 112.98 108.22         0.534            0.514
## 3 112.24 112.32 100.30 110.23 110.23         0.520            0.512
##   TsPercentage OppTsPercentage RebRate EffPts OppEffPts FastBreakPts
## 1        0.555           0.580   50.07 125.25    138.43        15.26
## 2        0.567           0.550   49.25 132.42    119.59        16.24
## 3        0.556           0.548   50.18 122.98    127.00        11.62
##   OppFBPts PointsInPaint OppPointsInPaint PointsOffTO OppPointsOffTO
## 1    16.51         51.19            49.36       21.14          16.88
## 2    13.17         44.78            45.93       14.82          18.12
## 3    11.83         48.76            51.20       17.35          15.38
##   SecondChancePTS OppSecondChancePTS PersonalFoulsPG OppPersonalFoulsPG
## 1           14.11              14.51          23.519             22.124
## 2           12.48              13.52          21.500             22.037
## 3           13.82              14.40          20.354             19.537
##   ShootingFoulsPG ShootingFoulsDrawnPG LessThnEightFeedUsage
## 1          14.889               12.642                 43.55
## 2          12.268               13.415                 43.45
## 3          12.134               10.549                 36.19
##   EightToSixteenFeedUsage SixteenToTwentyFourFeetUsage
## 1                   11.46                         4.80
## 2                   11.46                         4.89
## 3                   14.82                        10.90
```

```
##   TwentyFourPlusFeetUsage AvgShotDistance OppAvgShotDistance
## 1                   39.91           13.06              13.34
## 2                   39.96           13.18              12.89
## 3                   38.00           14.00              13.49
##   AvgMadeShotDistance OppMadeAvgShotDis
## 1               10.34             10.75
## 2               10.70             10.45
## 3               11.64             10.85
```

For this analysis, we model the relationship between Points Variables and WinPercentage Y: WinPercentage X1: Pts X2: FastBreakPts X3: PointsInPaint X4: PointsOffTO X5: SecondChancePts X6: ShootingFoulsDrawnPG

Build simple linear regression model (first variable in Y (response))

```
model1 <- lm(WinPercentage ~ Pts + FastBreakPts + PointsInPaint +
              PointsOffTO + SecondChancePTS + ShootingFoulsDrawnPG, nbaData)
model1
```

```
##
## Call:
## lm(formula = WinPercentage ~ Pts + FastBreakPts + PointsInPaint +
##     PointsOffTO + SecondChancePTS + ShootingFoulsDrawnPG, data = nbaData)
##
## Coefficients:
##          (Intercept)                   Pts          FastBreakPts
##           -0.7708069             0.0180015             0.0044135
##        PointsInPaint           PointsOffTO       SecondChancePTS
##           -0.0081145            -0.0144084            -0.0003683
## ShootingFoulsDrawnPG
##           -0.0080361
```

Now, lets see a summary of the model

```
summary(model1)
```

```
##
## Call:
## lm(formula = WinPercentage ~ Pts + FastBreakPts + PointsInPaint +
##     PointsOffTO + SecondChancePTS + ShootingFoulsDrawnPG, data = nbaData)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.28786 -0.09118  0.01309  0.08489  0.30448
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)          -0.7708069  0.2223130  -3.467 0.000695 ***
## Pts                   0.0180015  0.0024506   7.346 1.43e-11 ***
## FastBreakPts          0.0044135  0.0039489   1.118 0.265594
## PointsInPaint        -0.0081145  0.0033431  -2.427 0.016457 *
## PointsOffTO          -0.0144084  0.0040862  -3.526 0.000567 ***
## SecondChancePTS      -0.0003683  0.0088940  -0.041 0.967023
```

```
## ShootingFoulsDrawnPG -0.0080361  0.0104800  -0.767 0.444463
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1251 on 143 degrees of freedom
## Multiple R-squared:  0.3454, Adjusted R-squared:  0.318
## F-statistic: 12.58 on 6 and 143 DF,  p-value: 2.321e-11
```

From the summary above, we can see that only 3 of the 6 predictor variables are less than our significance value of 0.05 (shown with *'s next to predictors) Also, the Adjusted R-squared of 0.318 is lower than we want.

So, to determine how attain the best predictors for the model, we use the step function:

```r
step(model1, test = "F")
```

```
## Start:  AIC=-616.8
## WinPercentage ~ Pts + FastBreakPts + PointsInPaint + PointsOffTO +
##     SecondChancePTS + ShootingFoulsDrawnPG
##
##                        Df Sum of Sq    RSS     AIC F value    Pr(>F)
## - SecondChancePTS       1   0.00003 2.2374 -618.80  0.0017 0.9670226
## - ShootingFoulsDrawnPG  1   0.00920 2.2466 -618.19  0.5880 0.4444626
## - FastBreakPts          1   0.01954 2.2569 -617.50  1.2491 0.2655936
## <none>                              2.2374 -616.80
## - PointsInPaint         1   0.09218 2.3295 -612.75  5.8916 0.0164566 *
## - PointsOffTO           1   0.19453 2.4319 -606.30 12.4335 0.0005671 ***
## - Pts                   1   0.84427 3.0816 -570.78 53.9618  1.43e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Step:  AIC=-618.8
## WinPercentage ~ Pts + FastBreakPts + PointsInPaint + PointsOffTO +
##     ShootingFoulsDrawnPG
##
##                        Df Sum of Sq    RSS     AIC F value    Pr(>F)
## - ShootingFoulsDrawnPG  1   0.01067 2.2480 -620.09  0.6864 0.4087547
## - FastBreakPts          1   0.01956 2.2569 -619.49  1.2589 0.2637349
## <none>                              2.2374 -618.80
## - PointsInPaint         1   0.09853 2.3359 -614.34  6.3413 0.0128894 *
## - PointsOffTO           1   0.19513 2.4325 -608.26 12.5590 0.0005321 ***
## - Pts                   1   0.86496 3.1023 -571.77 55.6696 7.441e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Step:  AIC=-620.09
## WinPercentage ~ Pts + FastBreakPts + PointsInPaint + PointsOffTO
##
##                 Df Sum of Sq    RSS     AIC F value    Pr(>F)
## - FastBreakPts   1   0.01663 2.2647 -620.98  1.0725 0.3021039
## <none>                       2.2480 -620.09
## - PointsInPaint  1   0.11025 2.3583 -614.91  7.1109 0.0085322 **
## - PointsOffTO    1   0.19223 2.4403 -609.78 12.3990 0.0005746 ***
## - Pts            1   0.85695 3.1050 -573.64 55.2737 8.396e-12 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Step:  AIC=-620.98
## WinPercentage ~ Pts + PointsInPaint + PointsOffTO
##
##                 Df Sum of Sq    RSS     AIC F value    Pr(>F)
## <none>                       2.2647 -620.98
## - PointsInPaint  1   0.09615 2.3608 -616.74  6.1985 0.0139072 *
## - PointsOffTO    1   0.18987 2.4545 -610.90 12.2408 0.0006201 ***
## - Pts            1   0.97269 3.2374 -569.38 62.7082 5.558e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Call:
## lm(formula = WinPercentage ~ Pts + PointsInPaint + PointsOffTO,
##     data = nbaData)
##
## Coefficients:
##   (Intercept)            Pts  PointsInPaint    PointsOffTO
##     -0.862968       0.018281      -0.007731      -0.014209
```

From the step function, we can see the best AIC score is achieved with only 3 variables. Thus, lets create a second model with only the suggested variables:

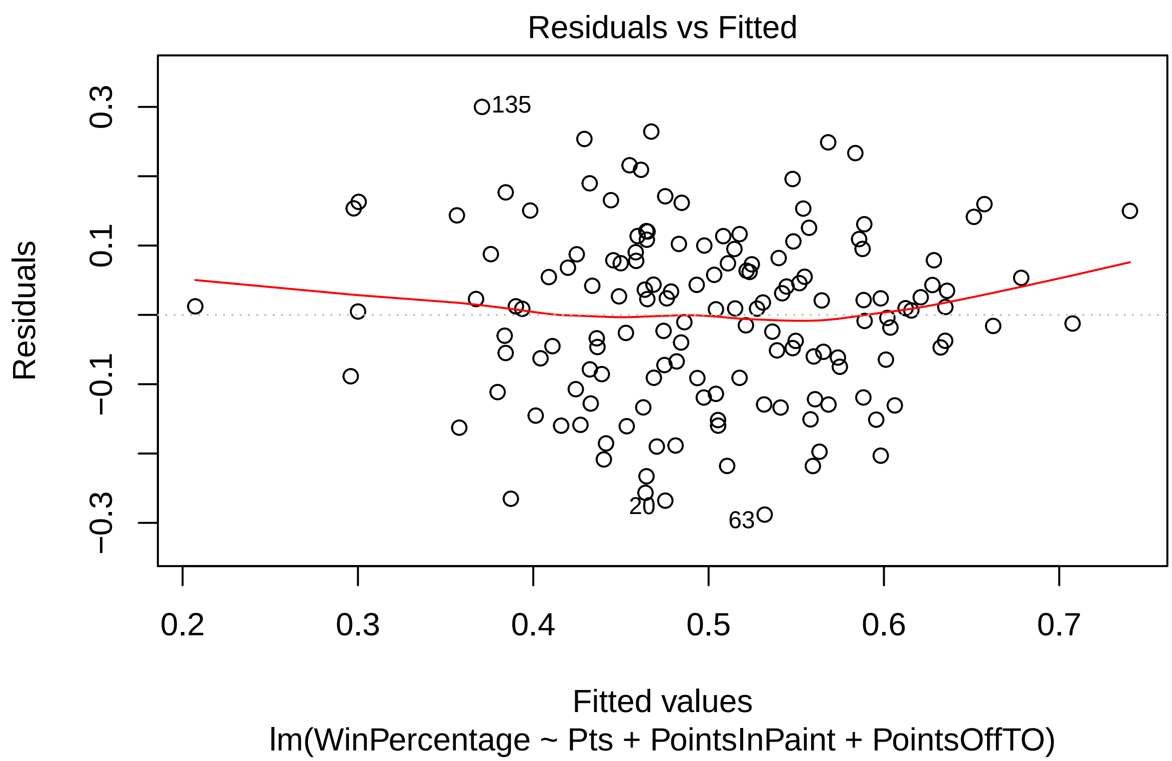
```r
model2 <- lm( WinPercentage ~ Pts + PointsInPaint + PointsOffTO,
    data = nbaData)

summary(model2)
```

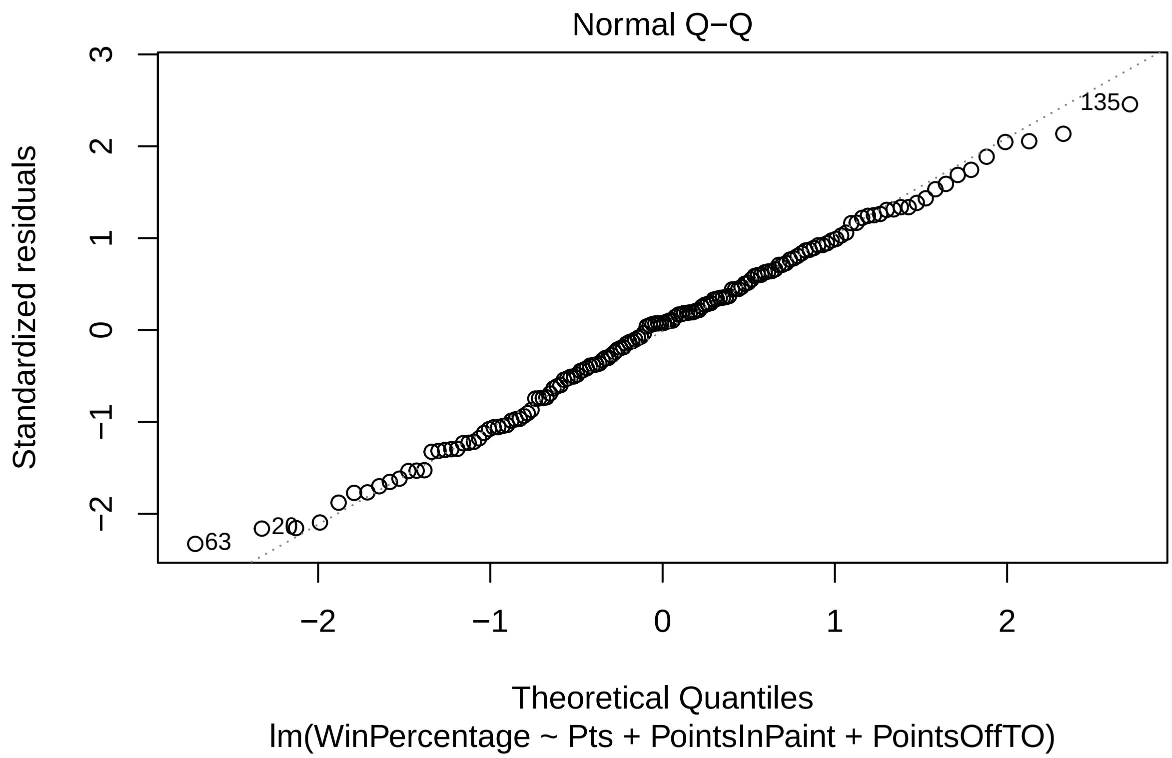
```
##
## Call:
## lm(formula = WinPercentage ~ Pts + PointsInPaint + PointsOffTO,
##     data = nbaData)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.288081 -0.087782  0.009431  0.086038  0.299977
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -0.862968   0.197082  -4.379 2.26e-05 ***
## Pts            0.018281   0.002309   7.919 5.56e-13 ***
## PointsInPaint -0.007731   0.003105  -2.490  0.01391 *
## PointsOffTO   -0.014209   0.004061  -3.499  0.00062 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1245 on 146 degrees of freedom
## Multiple R-squared:  0.3374, Adjusted R-squared:  0.3238
## F-statistic: 24.79 on 3 and 146 DF,  p-value: 5.075e-13
```

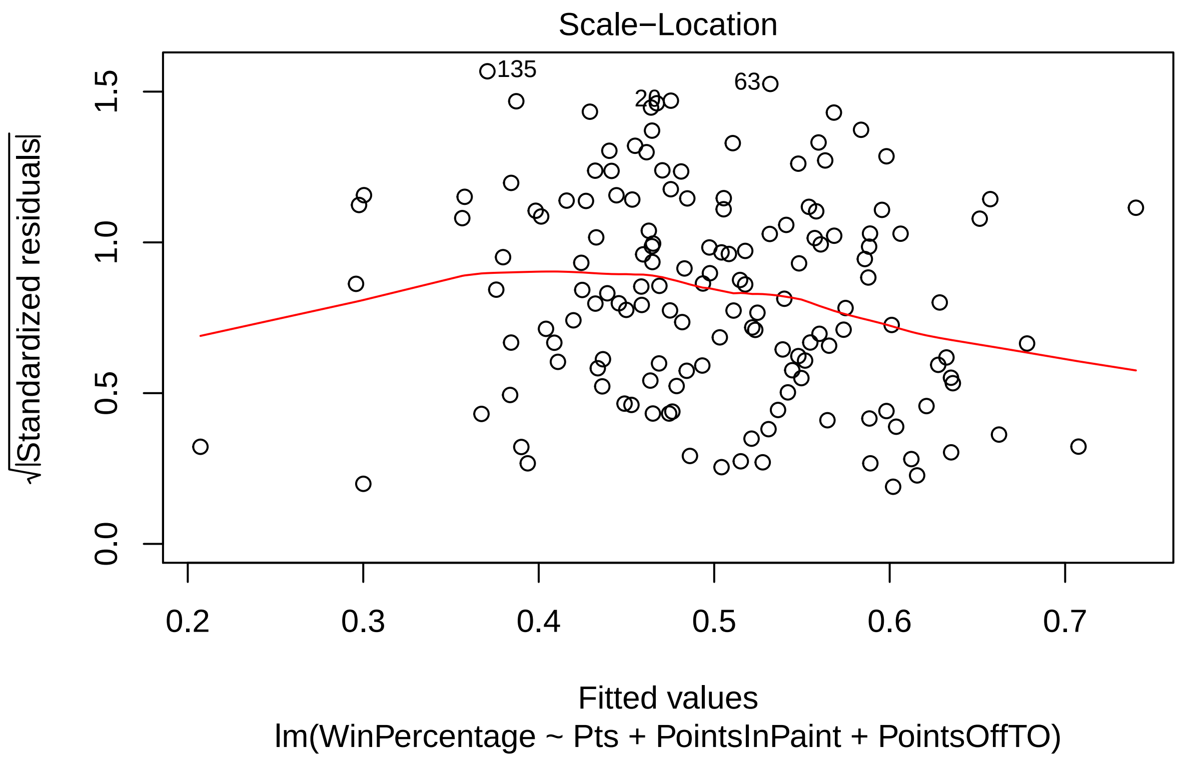
Now, the F-statistic is higher and the adjusted r-squared is a little higher. The p-value is below the significance value, but the model still doesn't look great.

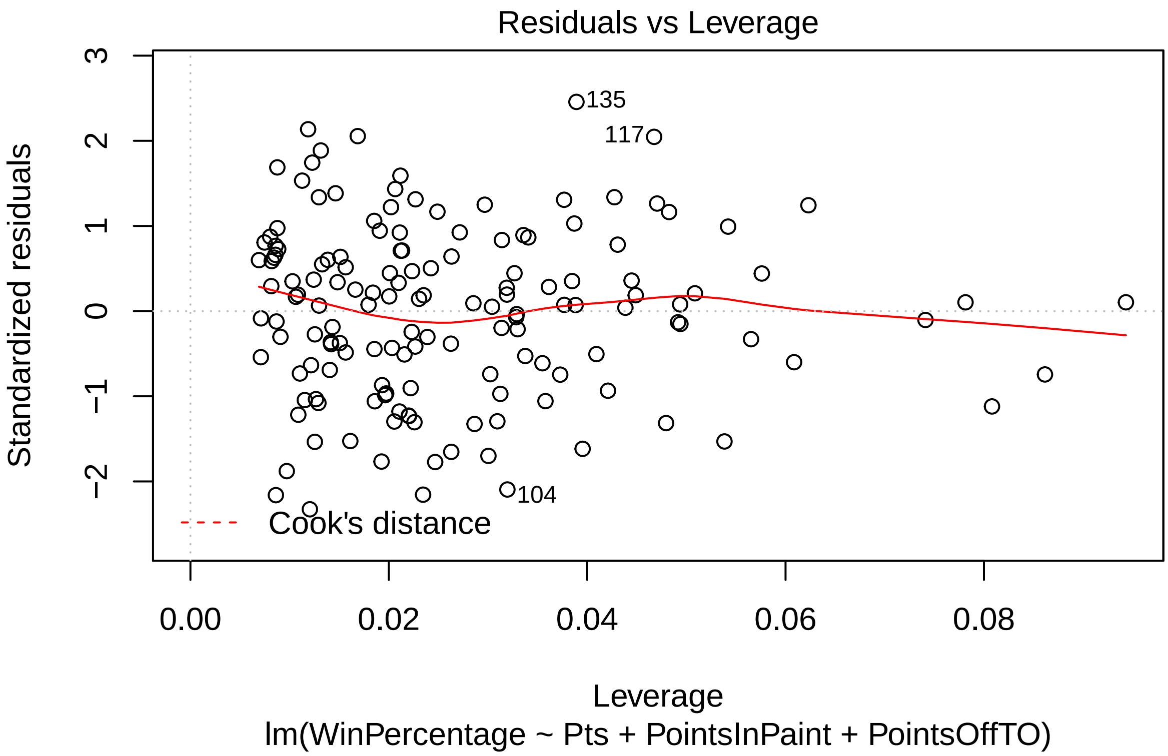
To diagnose some of the problems of the model, we can use diagnotic plots:

```r
plot(model2)
```




Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(WinPercentage ~ Pts + PointsInPaint + PointsOffTO)
135
63
20
−2
−1
0
1
2
3


Scale−Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
135
29
63
0.2
0.3
0.4
0.5
0.6
0.7
Fitted values
lm(WinPercentage ~ Pts + PointsInPaint + PointsOffTO)



In future blog posts, we will dig into the diagnostic plots. For now, we know how to drop predictors from a model.