Best Model For Predicting Home Run Hitters

Renee Yaldoo

12/9/2017

Introduction

Baseball, a very popular played and intricate sport in the United States. While others love to play baseball out in the park, many watch the Major League Baseball (MLB) games on homescreen televisions. People have always been fascinated when MLB players score Home-Runs (HR). My proposal question of the research project is: What is the best statistical model to determine Home-Run predictors a player will hit in a season? We will take a look at the following: R-Squared, R-Squared Adjusted, AIC, BIC, Press, Cp, and cross-validation (CV) to answer this question.

Hypothesis: I believe RBI's (Runs Batted In) most accurately predicts the number of homeruns a player will hit in a season.

Six independent parameters have been chosen for statistical analysis.

Dependent variable - Predicting HR (Home-Runs)

Independent variable(s) - AVG (Batting Average), H (Hit), IBB (Intentional Walk), RBI (Runs Batted In), TB (Total Bases), P/PA (Pitches per Plate Apperance).

Data - The data I have chosen is ESPN's MLB Player Batting Stats - 2017. I have only used all the ranked players who are top 144 in the list I was given.

Source(s): http://www.espn.com/mlb/stats/batting

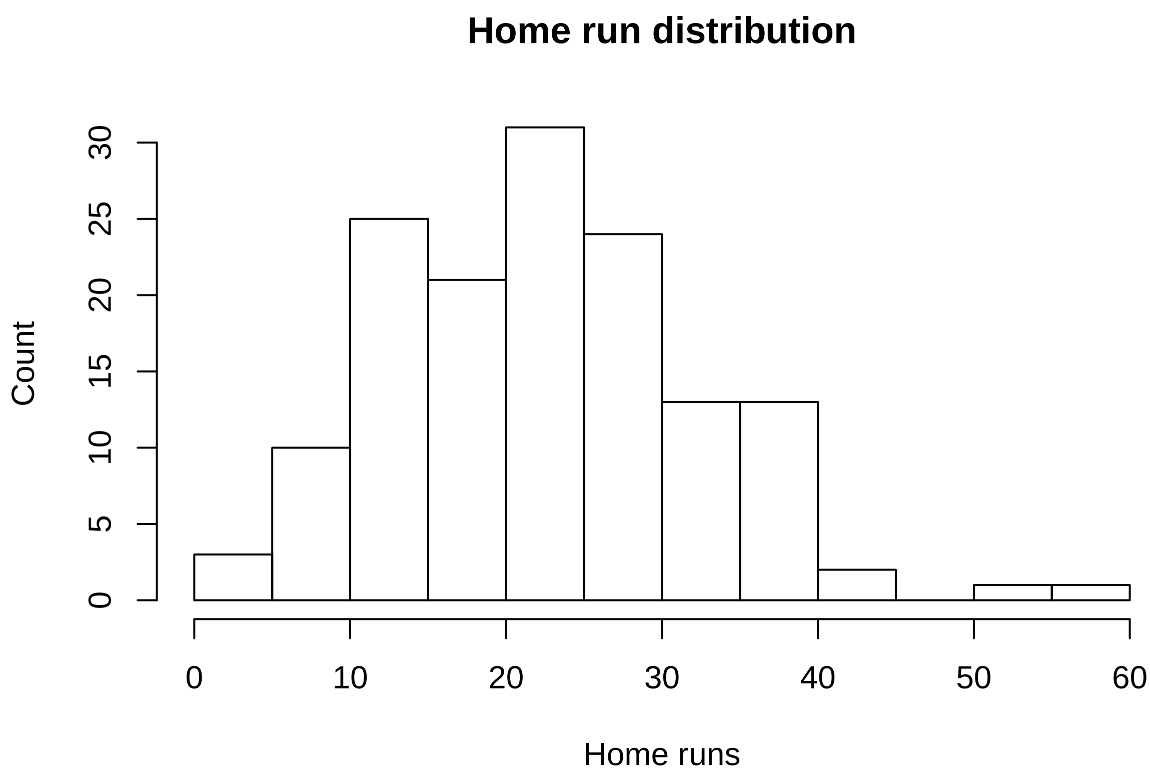
http://www.espn.com/mlb/stats/batting/_/type/sabermetric

Pairwise correlation

```
# MAT 5030 Final Project Code
#setwd("Desktop")
data = read.csv("BaseballData2017.csv")
data2 = data

data$Rank = NULL
data$PLAYER = NULL
data$TEAM = NULL

colnames(data) = c("AVG", "H", "HR", "RBI", "P.PA", "TB",
                   "IBB", "Adj.TB")
```

Histogram of HR

```
hist(data$HR, xlab = "Home runs",
     ylab = "Count",
     main = "Home run distribution")
```



Linear models for individual variables against HR

```r
m1 = lm(HR ~ AVG, data = data) #fit regression line for AVG
summary(m1) #produces summary
```

```
##
## Call:
## lm(formula = HR ~ AVG, data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -21.018  -7.925   0.098   7.103  36.082
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   21.880      8.010   2.732   0.0071 **
## AVG            3.694     29.389   0.126   0.9001
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.861 on 142 degrees of freedom
## Multiple R-squared:  0.0001113,  Adjusted R-squared:  -0.00693
## F-statistic: 0.0158 on 1 and 142 DF,  p-value: 0.9001
```

```r
m2 = lm(HR ~ H, data = data) #fit regression line for H
summary(m2) #produces summary
```

```
##
## Call:
```

```
## lm(formula = HR ~ H, data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -22.957  -7.485   0.196   7.194  35.294
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 17.33910    5.18699   3.343  0.00106 **
## H            0.03790    0.03502   1.082  0.28102
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.821 on 142 degrees of freedom
## Multiple R-squared:  0.00818,    Adjusted R-squared:  0.001195
## F-statistic: 1.171 on 1 and 142 DF,  p-value: 0.281
```

```r
m3 = lm(HR ~ RBI, data = data) #fit regression line for RBI
summary(m3) #produces summary
```

```
##
## Call:
## lm(formula = HR ~ RBI, data = data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -14.7925  -4.4347  -0.4238   4.1994  16.6918
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -6.00663    2.00626  -2.994  0.00325 **
## RBI          0.37894    0.02544  14.895  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.16 on 142 degrees of freedom
## Multiple R-squared:  0.6097, Adjusted R-squared:  0.607
## F-statistic: 221.9 on 1 and 142 DF,  p-value: < 2.2e-16
```

```r
m4 = lm(HR ~ P.PA, data = data) #fit regression line for P.PA
summary(m4) #produces summary
```

```
##
## Call:
## lm(formula = HR ~ P.PA, data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -22.116  -6.540  -0.571   5.822  35.307
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -28.555     13.688  -2.086 0.038760 *
## P.PA          13.227      3.514   3.764 0.000244 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.403 on 142 degrees of freedom
## Multiple R-squared:  0.09072,	Adjusted R-squared:  0.08431
## F-statistic: 14.17 on 1 and 142 DF,  p-value: 0.0002441
```

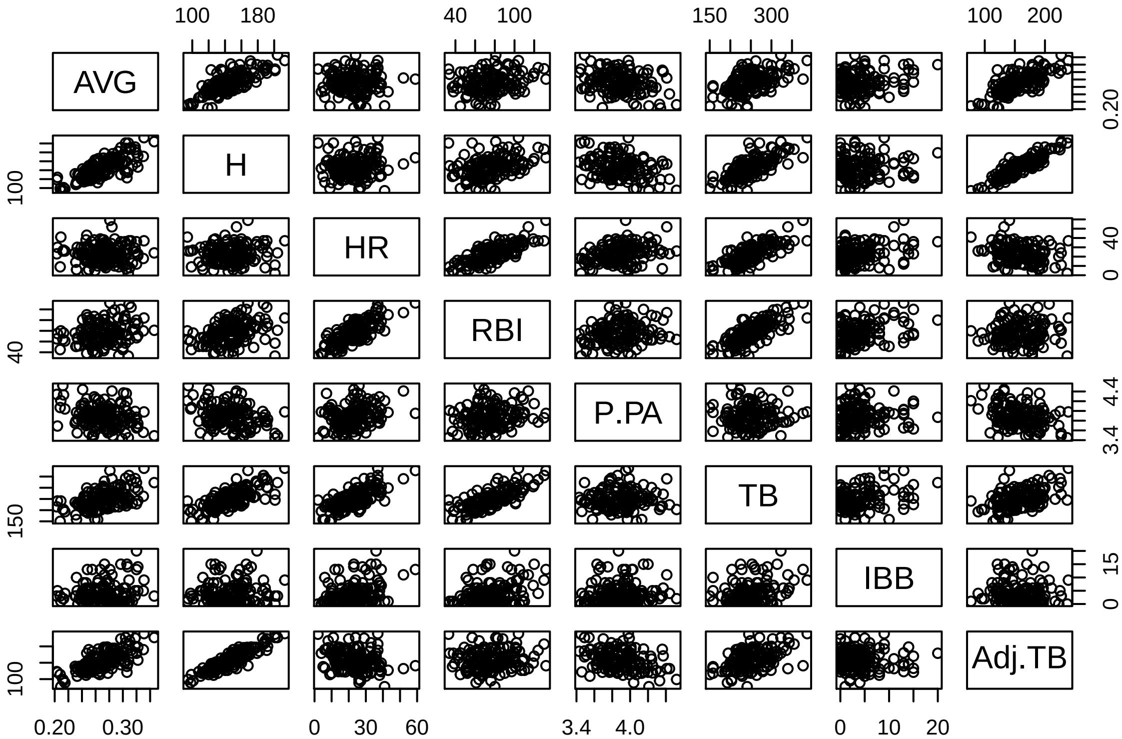
```
m5 = lm(HR ~ TB, data = data) #fit regression line for TB
summary(m5) #produces summary
```

```
##
## Call:
## lm(formula = HR ~ TB, data = data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -20.1520  -4.9321  -0.0885   4.2424  19.5050
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -18.08674    3.32714  -5.436 2.31e-07 ***
## TB            0.16424    0.01314  12.496  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.805 on 142 degrees of freedom
## Multiple R-squared:  0.5237, Adjusted R-squared:  0.5204
## F-statistic: 156.2 on 1 and 142 DF,  p-value: < 2.2e-16
```

```
m6 = lm(HR ~ IBB, data = data) #fit regression line for IBB
summary(m6) #produces summary
```

```
##
## Call:
## lm(formula = HR ~ IBB, data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -21.778  -6.383   0.069   5.704  28.742
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  19.5127     1.1087  17.599  < 2e-16 ***
## IBB           0.8265     0.1947   4.245 3.93e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.29 on 142 degrees of freedom
## Multiple R-squared:  0.1126, Adjusted R-squared:  0.1064
## F-statistic: 18.02 on 1 and 142 DF,  p-value: 3.93e-05
```

```
# Look at alll pairwise scaterplots
pairs(data)
```




```r
# Look at all pairwise correlations
knitr::kable(cor(data))
```

	AVG	H	HR	RBI	P.PA	TB	IBB	Adj.TB
AVG	1.0000000	0.7681342	0.0105479	0.2203242	-0.2485454	0.5235592	0.1719020	0.7196242
H	0.7681342	1.0000000	0.0904407	0.3632243	-0.2925054	0.7327068	0.0569008	0.9109087
HR	0.0105479	0.0904407	1.0000000	0.7808612	0.3011936	0.7237021	0.3355640	-0.2577191
RBI	0.2203242	0.3632243	0.7808612	1.0000000	0.1995418	0.7676313	0.3557898	0.0823039
P.PA	-0.2485454	-0.2925054	0.3011936	0.1995418	1.0000000	0.0283458	0.0880025	-0.3431205
TB	0.5235592	0.7327068	0.7237021	0.7676313	0.0283458	1.0000000	0.2715820	0.4802885
IBB	0.1719020	0.0569008	0.3355640	0.3557898	0.0880025	0.2715820	1.0000000	-0.0462516
Adj.TB	0.7196242	0.9109087	-0.2577191	0.0823039	-0.3431205	0.4802885	-0.0462516	1.0000000

```r
# First Multiple Linear Regression:
fit1 = lm(HR ~ AVG + H + RBI + P.PA + TB + IBB, data = data)
fit1
```

```
##
## Call:
## lm(formula = HR ~ AVG + H + RBI + P.PA + TB + IBB, data = data)
##
## Coefficients:
## (Intercept)          AVG            H          RBI         P.PA
##     5.62289     -0.29958     -0.38544      0.05663     -1.22481
##          TB          IBB
##     0.29758     -0.01755
```

If we really think about the first regression, Total Bases (TB) is heavily correlated with Home-Runs (HR). When doing multiple regression there can be issues with collinearity. So we are going to subtracting TB because the number of bases gained by a batter through his hits is very correlated with the homeruns. When a player scores a homerun, they have 4 bases which is the total amount of bases a player can get in a Home-Run. So the formula I have used to take out the TB is the following: Adj.TB = TB - HR * 4 So instead of using TB, I have used Adjusted TB (Adj.TB).

```r
# Put in all that new code
fit = lm(HR ~ . - TB, data = data)
```

Model selection part I: Stepwise, R-Square, Adjusted R-Square, AIC, BIC, PRESS, and Cp

1.1 Forward Selection

```r
null_fit = lm(HR ~ 1, data = data)
null_fit
```

```
##
## Call:
## lm(formula = HR ~ 1, data = data)
##
## Coefficients:
## (Intercept)
##       22.88
```

```r
step(null_fit, data = data, scope = list(lower = null_fit, upper = fit),
     direction = "forward")
```

```
## Start:  AIC=659.11
## HR ~ 1
##
##          Df Sum of Sq   RSS    AIC
## + RBI     1    8420.0  5389 525.61
## + IBB     1    1554.9 12254 643.91
## + P.PA    1    1252.7 12556 647.42
## + Adj.TB  1     917.2 12892 651.21
## <none>                 13809 659.11
## + H       1     113.0 13696 659.93
## + AVG     1       1.5 13808 661.09
##
## Step:  AIC=525.61
## HR ~ RBI
##
##          Df Sum of Sq    RSS    AIC
## + Adj.TB  1   1441.42 3947.6 482.79
## + H       1    593.70 4795.3 510.80
## + AVG     1    378.52 5010.5 517.13
## + P.PA    1    303.96 5085.1 519.25
## <none>                5389.0 525.61
## + IBB     1     52.71 5336.3 526.20
##
## Step:  AIC=482.79
## HR ~ RBI + Adj.TB
##
##        Df Sum of Sq    RSS    AIC
```

```
## + H       1   1362.25 2585.4 423.84
## + AVG     1    131.58 3816.0 479.91
## <none>                3947.6 482.79
## + IBB     1     17.60 3930.0 484.15
## + P.PA    1     13.82 3933.8 484.29
##
## Step:  AIC=423.84
## HR ~ RBI + Adj.TB + H
##
##        Df Sum of Sq    RSS    AIC
## + P.PA  1   104.257 2481.1 419.92
## <none>              2585.4 423.84
## + IBB   1    24.146 2561.2 424.49
## + AVG   1     0.656 2584.7 425.81
##
## Step:  AIC=419.92
## HR ~ RBI + Adj.TB + H + P.PA
##
##        Df Sum of Sq    RSS    AIC
## <none>              2481.1 419.92
## + IBB   1   25.7594 2455.3 420.41
## + AVG   1    0.5575 2480.6 421.89

##
## Call:
## lm(formula = HR ~ RBI + Adj.TB + H + P.PA, data = data)
##
## Coefficients:
## (Intercept)          RBI       Adj.TB            H         P.PA
##    -14.9581       0.2235      -0.4072       0.4689       4.2516
```

As we can see from the 1.1 Forward Selection, The best AIC that is given is: Step: AIC = 419.92 lm(formula = HR ~ RBI + Adj.TB + H + P.PA, data = data) Taking out AVG and IBB.

1.2 Backward Elimination

```
step(fit, data = data, direction = "backward")
```

```
## Start:  AIC=422.15
## HR ~ (AVG + H + RBI + P.PA + TB + IBB + Adj.TB) - TB
##
##          Df Sum of Sq    RSS    AIC
## - AVG     1      4.44 2455.4 420.41
## - IBB     1     29.64 2480.6 421.89
## <none>                2450.9 422.15
## - P.PA    1    105.77 2556.7 426.24
## - RBI     1   1121.11 3572.0 474.39
## - H       1   1342.97 3793.9 483.07
## - Adj.TB  1   2236.16 4687.1 513.52
##
## Step:  AIC=420.41
## HR ~ H + RBI + P.PA + IBB + Adj.TB
##
##          Df Sum of Sq    RSS    AIC
```

```
## - IBB      1      25.76 2481.1 419.92
## <none>                 2455.4 420.41
## - P.PA     1     105.87 2561.2 424.49
## - RBI      1    1159.42 3614.8 474.11
## - H        1    1460.46 3915.8 485.63
## - Adj.TB   1    2240.49 4695.8 511.79
##
## Step:  AIC=419.92
## HR ~ H + RBI + P.PA + Adj.TB
##
##          Df Sum of Sq    RSS    AIC
## <none>                 2481.1 419.92
## - P.PA     1     104.26 2585.4 423.84
## - RBI      1    1354.98 3836.1 480.66
## - H        1    1452.69 3933.8 484.29
## - Adj.TB   1    2243.78 4724.9 510.67
```

```
##
## Call:
## lm(formula = HR ~ H + RBI + P.PA + Adj.TB, data = data)
##
## Coefficients:
## (Intercept)            H          RBI         P.PA       Adj.TB
##    -14.9581       0.4689       0.2235       4.2516      -0.4072
```

As we can see from the 1.2 Backward Elimination, The best AIC that is given is: Step: AIC = 419.92 lm(formula = HR ~ H + RBI + P.PA + Adj.TB, data = data) Taking out AVG and IBB.

1.3 Forward Stepwise Regression

```
step(null_fit, data = data, scope = list(lower = null_fit, upper = fit),
     direction = "both")
```

```
## Start:  AIC=659.11
## HR ~ 1
##
##          Df Sum of Sq   RSS    AIC
## + RBI      1    8420.0  5389 525.61
## + IBB      1    1554.9 12254 643.91
## + P.PA     1    1252.7 12556 647.42
## + Adj.TB   1     917.2 12892 651.21
## <none>                 13809 659.11
## + H        1     113.0 13696 659.93
## + AVG      1       1.5 13808 661.09
##
## Step:  AIC=525.61
## HR ~ RBI
##
##          Df Sum of Sq    RSS    AIC
## + Adj.TB   1    1441.4  3947.6 482.79
## + H        1     593.7  4795.3 510.80
## + AVG      1     378.5  5010.5 517.13
## + P.PA     1     304.0  5085.1 519.25
## <none>                  5389.0 525.61
```

```
## + IBB     1      52.7  5336.3 526.20
## - RBI     1    8420.0 13809.0 659.11
##
## Step:  AIC=482.79
## HR ~ RBI + Adj.TB
##
##          Df Sum of Sq     RSS    AIC
## + H       1    1362.2  2585.4 423.84
## + AVG     1     131.6  3816.0 479.91
## <none>                 3947.6 482.79
## + IBB     1      17.6  3930.0 484.15
## + P.PA    1      13.8  3933.8 484.29
## - Adj.TB  1    1441.4  5389.0 525.61
## - RBI     1    8944.2 12891.8 651.21
##
## Step:  AIC=423.84
## HR ~ RBI + Adj.TB + H
##
##          Df Sum of Sq    RSS    AIC
## + P.PA    1    104.26 2481.1 419.92
## <none>                2585.4 423.84
## + IBB     1     24.15 2561.2 424.49
## + AVG     1      0.66 2584.7 425.81
## - H       1   1362.25 3947.6 482.79
## - RBI     1   1728.45 4313.8 495.57
## - Adj.TB  1   2209.97 4795.3 510.80
##
## Step:  AIC=419.92
## HR ~ RBI + Adj.TB + H + P.PA
##
##          Df Sum of Sq    RSS    AIC
## <none>                2481.1 419.92
## + IBB     1     25.76 2455.4 420.41
## + AVG     1      0.56 2480.6 421.89
## - P.PA    1    104.26 2585.4 423.84
## - RBI     1   1354.98 3836.1 480.66
## - H       1   1452.69 3933.8 484.29
## - Adj.TB  1   2243.78 4724.9 510.67
```

```
##
## Call:
## lm(formula = HR ~ RBI + Adj.TB + H + P.PA, data = data)
##
## Coefficients:
## (Intercept)          RBI       Adj.TB            H         P.PA
##    -14.9581       0.2235      -0.4072       0.4689       4.2516
```

As we can see from the 1.3 Forward Stepwise Regression, The best AIC that is given is: Step: AIC = 419.92 lm(formula = HR ~ RBI + Adj.TB + H + P.PA, data = data) Taking out AVG and IBB.

1.4 Backwards Stepwise Regression

```
step(fit, data = data, direction = "both")
```

```
## Start:  AIC=422.15
## HR ~ (AVG + H + RBI + P.PA + TB + IBB + Adj.TB) - TB
##
##          Df Sum of Sq    RSS    AIC
## - AVG     1      4.44 2455.4 420.41
## - IBB     1     29.64 2480.6 421.89
## <none>                2450.9 422.15
## - P.PA    1    105.77 2556.7 426.24
## - RBI     1   1121.11 3572.0 474.39
## - H       1   1342.97 3793.9 483.07
## - Adj.TB  1   2236.16 4687.1 513.52
##
## Step:  AIC=420.41
## HR ~ H + RBI + P.PA + IBB + Adj.TB
##
##          Df Sum of Sq    RSS    AIC
## - IBB     1     25.76 2481.1 419.92
## <none>                2455.4 420.41
## + AVG     1      4.44 2450.9 422.15
## - P.PA    1    105.87 2561.2 424.49
## - RBI     1   1159.42 3614.8 474.11
## - H       1   1460.46 3915.8 485.63
## - Adj.TB  1   2240.49 4695.8 511.79
##
## Step:  AIC=419.92
## HR ~ H + RBI + P.PA + Adj.TB
##
##          Df Sum of Sq    RSS    AIC
## <none>                2481.1 419.92
## + IBB     1     25.76 2455.4 420.41
## + AVG     1      0.56 2480.6 421.89
## - P.PA    1    104.26 2585.4 423.84
## - RBI     1   1354.98 3836.1 480.66
## - H       1   1452.69 3933.8 484.29
## - Adj.TB  1   2243.78 4724.9 510.67

##
## Call:
## lm(formula = HR ~ H + RBI + P.PA + Adj.TB, data = data)
##
## Coefficients:
## (Intercept)            H          RBI         P.PA       Adj.TB
##    -14.9581       0.4689       0.2235       4.2516      -0.4072
```

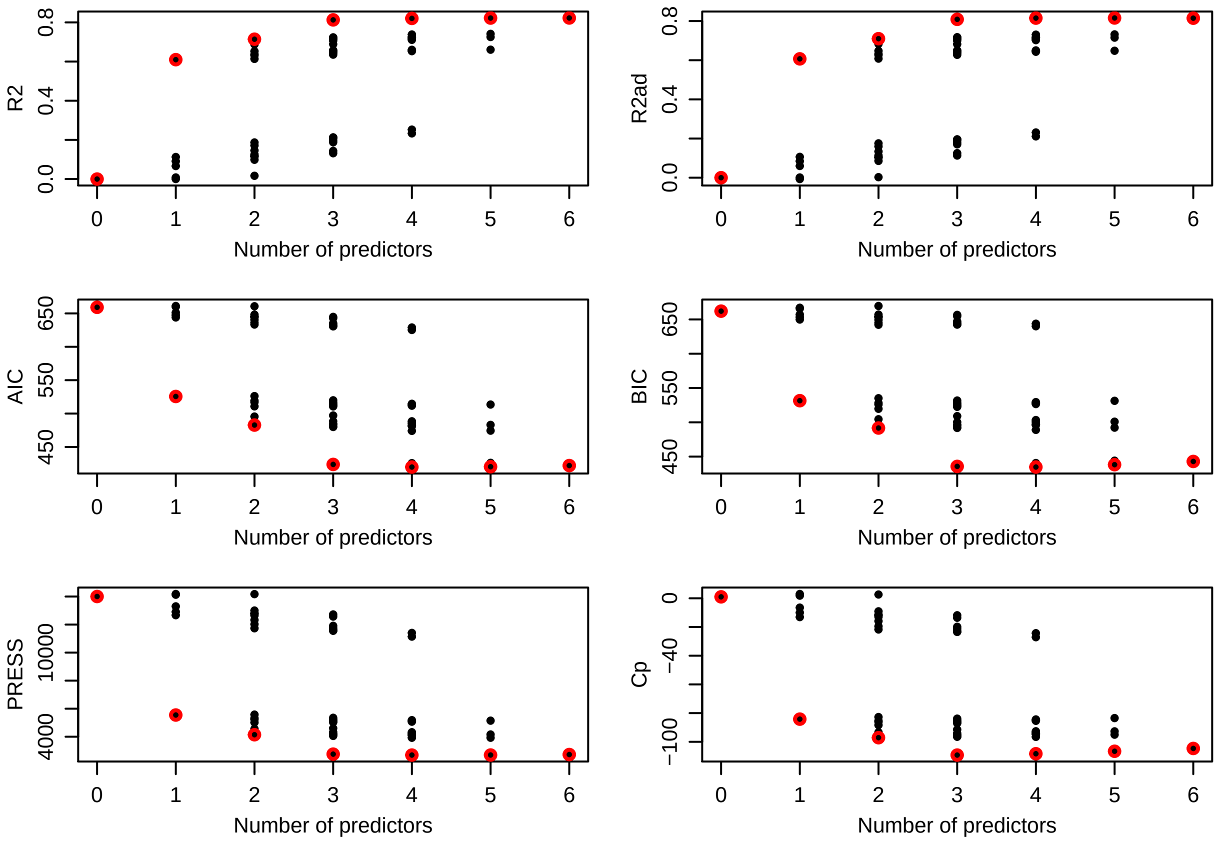
As we can see from the 1.4 Backwards Stepwise Regression, The best AIC that is given is: Step: AIC = 419.92 lm(formula = HR ~ H + RBI + P.PA + Adj.TB, data = data) Taking out AVG and IBB once more.

2 All Possible Regressions

```
# ============================== object: a fitted model
# (lm object) MSEfull: MSE for the full model (needed for Cp only)
CalcCrit = function(object, MSEfull){
  sumObj = summary(object)
  R2 = sumObj$r.squared
```

```r
  R2ad = sumObj$adj.r.squared
  SSE = tail(anova(object)$"Sum Sq", 1)
  n = length(object$fitted.values)
  p = object$rank
  AIC = n * log(SSE) - n * log(n) + 2 * p
  BIC = n * log(SSE) - n * log(n) + log(n) * p
  pr = resid(object)/(1 - lm.influence(object)$hat)
  PRESS = sum(pr^2)
  Cp = SSE/MSEfull - (n - 2 * p)
  return(c(R2, R2ad, AIC, BIC, PRESS, Cp))
}

x_all = c("AVG", "H", "RBI", "P.PA", "IBB", "Adj.TB")
p = length(x_all)

# All combinations of the terms
all_model = expand.grid(data.frame(rbind(rep(FALSE, p), rep(TRUE,p))))
all_model = all_model[-1, ]

MSEfull = (sigma(null_fit))^2
critResults = CalcCrit(null_fit, MSEfull)

# Calculate the criteria for all combinations of models
for (i in 1:nrow(all_model)){
  # Using the paste() function we can determine the formula to
  # use for each combination
  MyForm = formula(paste("HR ~ ", paste(x_all[all_model[i,] == T],
                                        collapse = "+")))
  Fit = lm(MyForm, data = data)
  critResults = rbind(critResults, CalcCrit(Fit, MSEfull))
  if ((i %% 200) == 0){
    # Write on screen every 200th iteration
    print(paste(i, "models done out of", nrow(Comb)))
  }
}

colnames(critResults) = CritNames = c("R2", "R2ad", "AIC", "BIC",
                                      "PRESS", "Cp")
combResults = cbind(c(0, apply(all_model, 1, sum)), rbind(F, all_model),
                    critResults)
names(combResults)[1] = "p-1"


# ------------------------------- Plot criteria for all submodels
par(mfrow = c(3, 2), mar = c(3.5, 3.5, 1, 1), mgp = c(2, 0.8, 0))
for (m in 1:length(CritNames)){
  plot(combResults[, 1], combResults[, CritNames[m]], pch = 20,
       xlab = "Number of predictors", ylab = CritNames[m])
  if (m <= 2){
    # Plot red line R2 and R2adj
    points(0:ncol(all_model), sapply(split(combResults[, CritNames[m]],
                                           combResults[, "p-1"]), max),
           # type = "l",
```

```
            col = "red", lwd = 2)
  } else{
    # Rest of the criteria want minimum
    points(0:ncol(all_model), sapply(split(combResults[, CritNames[m]],
                                           combResults[, "p-1"]), min),
            # type = "1",
            col = "red", lwd = 2)
  }
}
```



Model Selection Results

```
#_________________________________ The five best models in terms of R2
Nbest = 5
Cind = match("R2", names(combResults))
BestR2 = combResults[order(combResults[, "R2"],
                           decreasing = T), ][1:Nbest,c(1:(p + 1), Cind)]
knitr::kable(BestR2)
```

	p-1	X1	X2	X3	X4	X5	X6	R2
64	6	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	0.8225131
63	5	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	0.8221917
48	5	TRUE	TRUE	TRUE	TRUE	FALSE	TRUE	0.8203667
47	4	FALSE	TRUE	TRUE	TRUE	FALSE	TRUE	0.8203263

	p-1	X1	X2	X3	X4	X5	X6	R2
56	5	TRUE	TRUE	TRUE	FALSE	TRUE	TRUE	0.8148534

```
#__________________________________ The five best models in terms of R2ad
Nbest = 5
Cind = match("R2ad", names(combResults))
BestR2ad = combResults[order(combResults[, "R2ad"],
                             decreasing = T), ][1:Nbest,
                                                c(1:(p + 1), Cind)]
knitr::kable(BestR2ad)
```

	p-1	X1	X2	X3	X4	X5	X6	R2ad
63	5	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	0.8157494
47	4	FALSE	TRUE	TRUE	TRUE	FALSE	TRUE	0.8151559
64	6	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	0.8147399
48	5	TRUE	TRUE	TRUE	TRUE	FALSE	TRUE	0.8138582
55	4	FALSE	TRUE	TRUE	FALSE	TRUE	TRUE	0.8091876

```
#__________________________________ The five best models in terms of AIC
Nbest = 5
Cind = match("AIC", names(combResults))
BestAIC = combResults[order(combResults[, "AIC"]), ][1:Nbest,
                                                     c(1:(p + 1), Cind)]
knitr::kable(BestAIC)
```

	p-1	X1	X2	X3	X4	X5	X6	AIC
47	4	FALSE	TRUE	TRUE	TRUE	FALSE	TRUE	419.9175
63	5	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	420.4146
48	5	TRUE	TRUE	TRUE	TRUE	FALSE	TRUE	421.8851
64	6	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	422.1541
39	3	FALSE	TRUE	TRUE	FALSE	FALSE	TRUE	423.8447

```
#-------------------------------- The five best models in terms of BIC
Nbest = 5
Cind = match("BIC", names(combResults))
BestBIC = combResults[order(combResults[, "BIC"]), ][1:Nbest,
                                                     c(1:(p + 1), Cind)]
knitr::kable(BestBIC)
```

	p-1	X1	X2	X3	X4	X5	X6	BIC
47	4	FALSE	TRUE	TRUE	TRUE	FALSE	TRUE	434.7665
39	3	FALSE	TRUE	TRUE	FALSE	FALSE	TRUE	435.7239
63	5	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	438.2335
55	4	FALSE	TRUE	TRUE	FALSE	TRUE	TRUE	439.3426
48	5	TRUE	TRUE	TRUE	TRUE	FALSE	TRUE	439.7040

```
#________________________________ The five best models in terms of PRESS
Nbest = 5
```

```r
Cind = match("PRESS", names(combResults))
BestPRESS = combResults[order(combResults[, "PRESS"]), ][1:Nbest,
                                                         c(1:(p + 1), Cind)]
knitr::kable(BestPRESS)
```

	p-1	X1	X2	X3	X4	X5	X6	PRESS
47	4	FALSE	TRUE	TRUE	TRUE	FALSE	TRUE	2691.085
63	5	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	2695.059
64	6	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	2728.405
48	5	TRUE	TRUE	TRUE	TRUE	FALSE	TRUE	2731.124
39	3	FALSE	TRUE	TRUE	FALSE	FALSE	TRUE	2757.406

```r
#________________________________ The five best models in terms of Cp
Nbest = 5
Cind = match("Cp", names(combResults))
BestCp = combResults[order(combResults[, "Cp"]), ][1:Nbest,
                                                   c(1:(p + 1), Cind)]

knitr::kable(BestCp)
```

	p-1	X1	X2	X3	X4	X5	X6	Cp
39	3	FALSE	TRUE	TRUE	FALSE	FALSE	TRUE	-109.2270
47	4	FALSE	TRUE	TRUE	TRUE	FALSE	TRUE	-108.3067
55	4	FALSE	TRUE	TRUE	FALSE	TRUE	TRUE	-107.4771
40	4	TRUE	TRUE	TRUE	FALSE	FALSE	TRUE	-107.2338
63	5	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	-106.5734

R-Square

As we can see, the more independent parameters we have for R-Squared, the higher our R-Squared value will be. The highest R-Squared chosen is: R2 = 0.8225131 lm(formula = HR ~ AVG + H + RBI + P.PA + IBB + Adj.TB, data = data)

R-Square Adjusted

For R-Squared Adjusted, this value will not necessarily increase as additional terms are introduced into the model. We want a model with the maximum Adjusted R-Square. The highest chosen R-Squared Adjusted is: R2ad = 0.8157494 lm(formula = HR ~ H + RBI + P.PA + IBB + Adj.TB, data = data), where X1 = AVG is FALSE.

AIC

As we can see for AIC, we want the model that gives us the lowest AIC. The model that is chosen for the lowest AIC is: AIC = 419.9175 lm(formula = HR ~ H + RBI + P.PA + Adj.TB, data = data), where X1 = AVG, X5 = IBB and both are FALSE.

BIC

As we can see for BIC, we want the model that gives us the lowest BIC. The model that is chosen for the lowest BIC is: BIC = 434.7665 lm(formula = HR ~ H + RBI + P.PA + Adj.TB, data = data), where X1 = AVG, X5 = IBB and both are FALSE.

PRESS

As we can see for PRESS, we want the model that gives us the lowest PRESS. The model that is chosen for the lowest PRESS is: PRESS = 2691.085 lm(formula = HR ~ H + RBI + P.PA + Adj.TB, data = data), where X1 = AVG, X5 = IBB and both are FALSE.

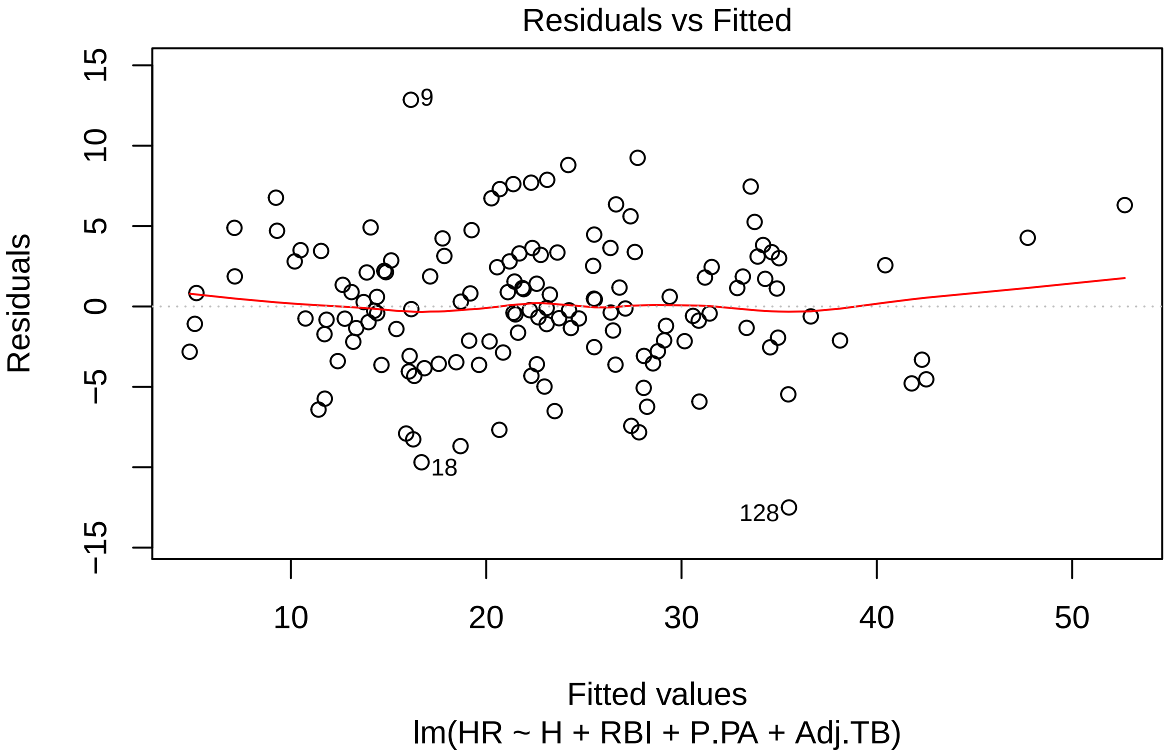
Cp

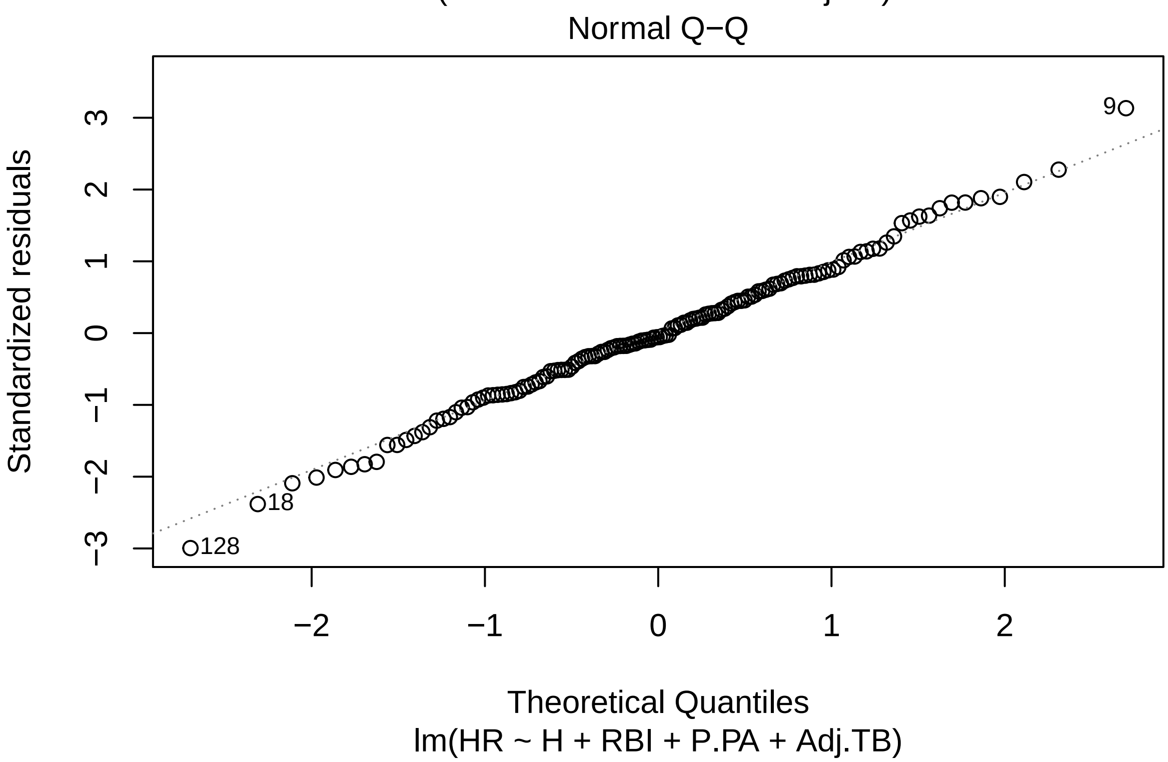
As we can see for Cp, we want the model that gives us the lowest Cp. The model that is chosen for the lowest Cp is: Cp = -109.2270 lm(formula = HR ~ H + RBI + Adj.TB, data = data), where X1 = AVG, X4 = P.PA, and X5 = IBB resulting these three paramaters to be FALSE. Notice how we have a negative Cp value. We must beware of negative values of Cp. This could have been resulted because the MSE for the full model overestimates the true (standard deviation)^2.

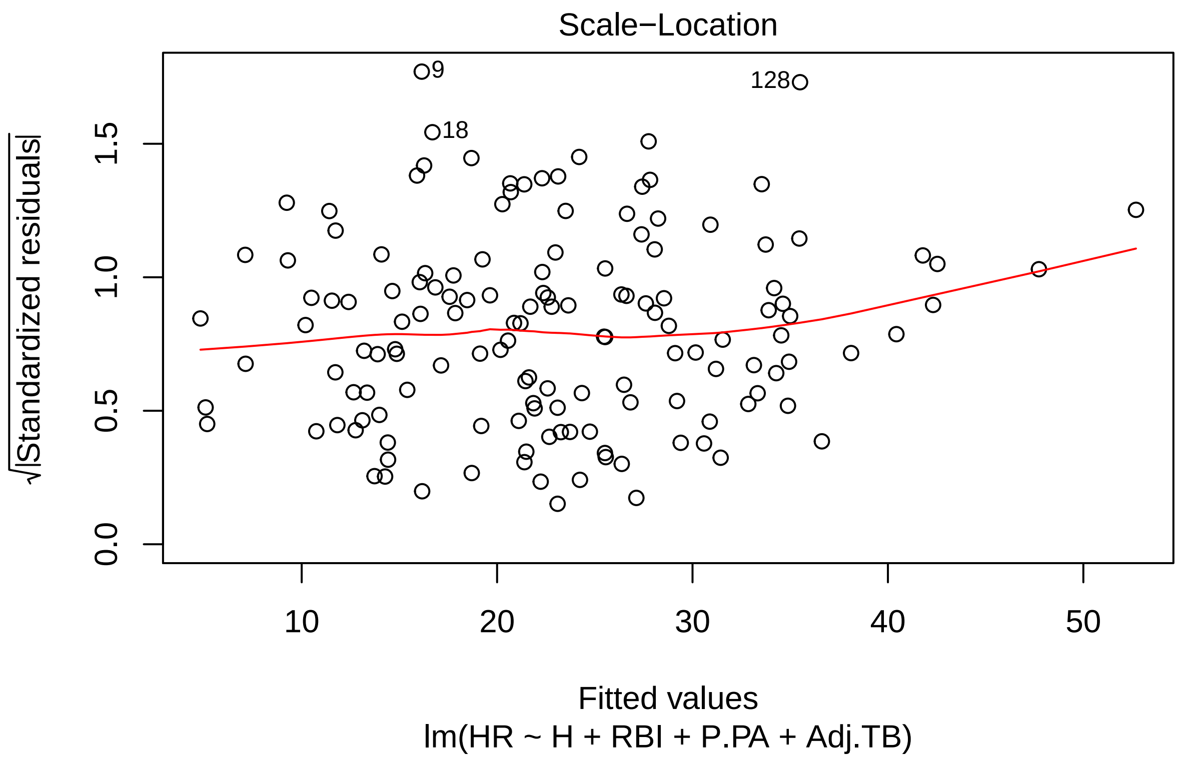
Model Diagnostics

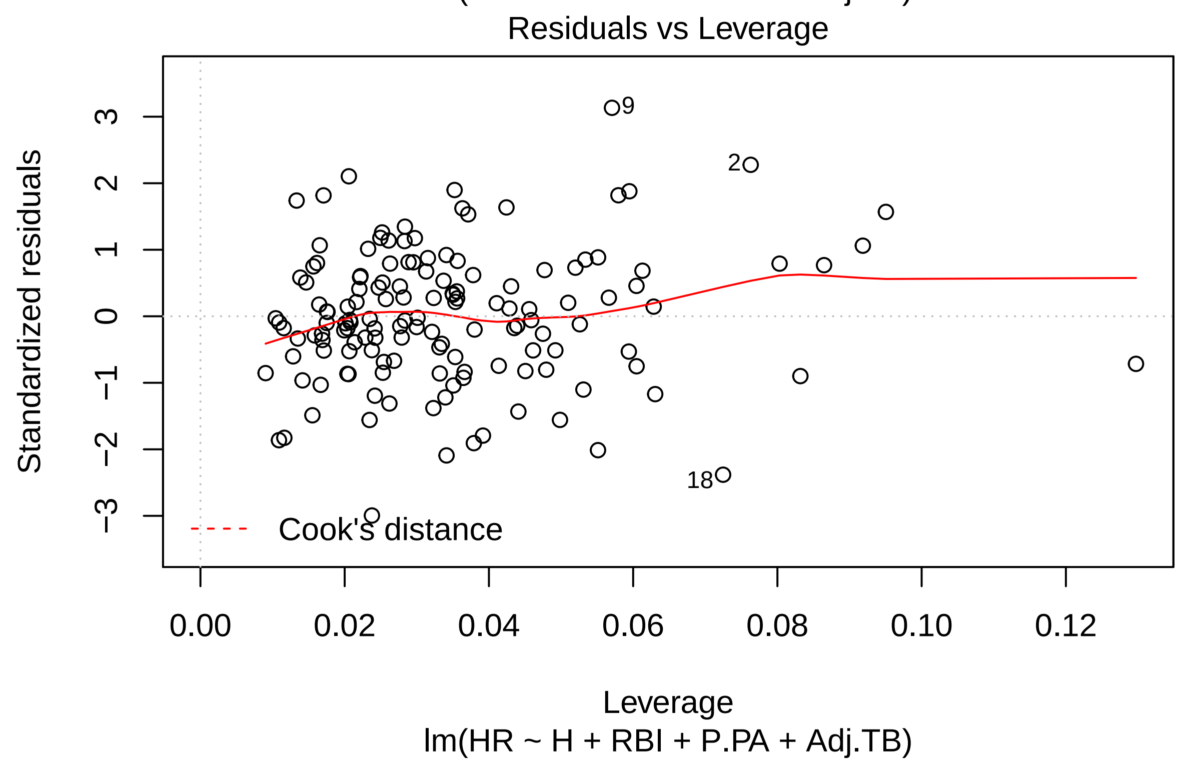
1. Normality plot
2. Residuals vs. Fitted
3. Residuals vs. Leverage

```r
# Simplest diagnostic plot
# This is the best_model chosen from AIC, BIC, and PRESS
best_model = lm(HR ~ H + RBI + P.PA + Adj.TB, data = data)
plot(best_model)
```


Residuals vs Fitted
Residuals
Fitted values
lm(HR ~ H + RBI + P.PA + Adj.TB)
9
18
128


Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(HR ~ H + RBI + P.PA + Adj.TB)
9
18
128





1. For the normality plot most points along the quantile-quantile line meaning that the distribution of residuals is approximately normal. From the normality plot, we can see that 18, and 128 are closse to -3 standard deviation away from the mean. And 9 is near 3 standard deviations from the mean.

2. For the residuals vs. fitted plot there were three points identified as having large residual values, which were 9, 18, and 128. Along the fitted values there appears to be constant variance, which matches our model assumption. It also looks like the relationship between our predictors and the response is linear.

3. In the residuals vs. leverage there are a few points with high leverage but they don't conincide with the points with high residuals, so those points shouldn't have too big an effect on the model fit. All points were within the Cook's Distance of 0.5 so there are no points that are overly influential.

```
knitr::kable(data2[c(9, 18, 128), ])
```

	Rank	PLAYER	TEAM	AVG	H	HR	RBI	P.PA	TB	IBB	Adjusted.TB
9	9	Jose Ramirez	CLE	0.318	186	29	83	3.99	341	5	225
18	18	Joe Mauer	MIN	0.305	160	7	71	4.36	219	3	191
128	128	Albert Pujols	LAA	0.241	143	23	101	3.91	229	5	137

9 : Expected to be around 14 (from the x-axis of Residuals vs. Fitted) Over expectation.

18 : Expected to be around 17 (from the x-axis of Residuals vs. Fitted) Under expectation.

128: Expected to be around 36 (from the x-axis of Residuals vs. Fitted) Under expectation.

Why did only 18 show up in the high-leverage points? His total number of home runs is lower on the scale of total home runs for all players, so the point has more of an effect on the model fit.

```
best_model.hat <- hatvalues(best_model)

## This heuristic value to identify of possible leverage  hatvalue > #2*(k+1)/n
# This idx_hat is the index of points
idx_hat <- which(best_model.hat > (2*(4+1)/nrow(data)))
idx_hat
```

```
##   2  13  14  18  20  46  50 140
##   2  13  14  18  20  46  50 140
```

```
knitr::kable(data2[idx_hat, ])
```

	Rank	PLAYER	TEAM	AVG	H	HR	RBI	P.PA	TB	IBB	Adjusted.TB
2	2	Charlie Blackmon	COL	0.331	213	37	104	3.98	387	9	239
13	13	Nolan Arenado	COL	0.309	187	37	130	3.86	355	9	207
14	14	Dee Gordon	MIA	0.308	201	2	33	3.45	245	0	237
18	18	Joe Mauer	MIN	0.305	160	7	71	4.36	219	3	191
20	20	Ender Inciarte	ATL	0.304	201	11	57	3.54	271	3	227
46	46	Aaron Judge	NYY	0.284	154	52	114	4.41	340	11	132
50	50	Giancarlo Stanton	MIA	0.281	168	59	132	3.95	377	13	141
140	140	Curtis Granderson	LAD/NYM	0.212	95	26	64	4.52	203	2	99

Model selection part II: Cross validation

Model 1 : everything except for TB

fit = lm(HR ~ . - TB, data = data)

Model 2 : chosen using stepwise variable selection

best_model = lm(formula = HR ~ H + RBI + P.PA + Adj.TB, data = data)

The model selected by stepwise variable selection dropped the two variables Intentional Walk (IBB) and Batting Average (AVG).

Now we will take a look at cross validation and see which model performs best using cross validation. We used the bootstrap version of cross-validation using 100 iterations on training sets with 75% of the data and predictions made on test sets with the remaining 25%.

Note to self : There is also a version called 5-fold cross validation and that's not what we used. That version splits the data into 5 test sets and you take the data not in that set to train the model on that remaining data.

Cross Validation

```
err1 <- double(10)
# The best_model is model2
err2 <- double(10)
err3 <- double(10)
err4 <- double(10)

# Set a random
set.seed(1)
runif(10)
```

```
##  [1] 0.26550866 0.37212390 0.57285336 0.90820779 0.20168193 0.89838968
##  [7] 0.94467527 0.66079779 0.62911404 0.06178627
```

```
set.seed(1)
for(k in 1:100){
  # Select 75% of the data to train on
  # idx are the index values of the training set
  idx <- sample(nrow(data), round(nrow(data)*.75))
  # Subseting the row indices for training data and test
  train <- data[idx,  ]
  test  <- data[-idx,  ]

  #Fit models on training data
  model1 <- lm(formula = HR ~ H + RBI + Adj.TB, data = train)
  # This is the best model from above
  model2 <- lm(formula = HR ~ H + RBI + P.PA + Adj.TB, data = train)
  model3 <- lm(formula = HR ~ H + RBI + P.PA + IBB + Adj.TB, data = train)
  model4 <- lm(formula = HR ~ H + RBI + P.PA + IBB + Adj.TB + AVG, data = train)

  # Predict the home-run values for the test data
  pred1 <- predict(model1, newdata = test)
  pred2 <- predict(model2, newdata = test)
  pred3 <- predict(model3, newdata = test)
  pred4 <- predict(model4, newdata = test)

  # (1/n)*sum((y_i - hat-y_i)^2)
  # Compute MSE
  # MSPR
  err1[k] <- mean((pred1 - test$HR)^2)
  err2[k] <- mean((pred2 - test$HR)^2)
  err3[k] <- mean((pred3 - test$HR)^2)
  err4[k] <- mean((pred4 - test$HR)^2)
}
```

```r
mean(err1)
```

```
## [1] 19.45637
```

```r
mean(err2)
```

```
## [1] 19.07233
```

```r
mean(err3)
```

```
## [1] 19.21392
```

```r
mean(err4)
```

```
## [1] 19.51176
```

This is the best model from above:

```r
model2 <- lm(formula = HR ~ H + RBI + P.PA + Adj.TB, data = train)
summary(model2)
```

```
##
## Call:
## lm(formula = HR ~ H + RBI + P.PA + Adj.TB, data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -12.5606  -2.4535  -0.3158   2.6465  11.8899
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -17.36262    8.69802  -1.996   0.0486 *
## H             0.49228    0.06067   8.113 1.10e-12 ***
## RBI           0.21080    0.03044   6.925 3.85e-10 ***
## P.PA          4.49306    2.03020   2.213   0.0291 *
## Adj.TB       -0.41118    0.04193  -9.807  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.259 on 103 degrees of freedom
## Multiple R-squared:  0.8202, Adjusted R-squared:  0.8133
## F-statistic: 117.5 on 4 and 103 DF,  p-value: < 2.2e-16
```

The model chosen as the best model using stepwise variable selection, AIC, BIC, and PRESS also performed best using cross validation.

Do I keep this? (The P.PA variable had the highest p-value in our previous "best" model. The model where we removed P.PA from the predictors had a higher MSRP than, for example, the model which included IBB.)

```r
data_scaled <- lapply(data, scale)
scaled_model <- lm(formula = HR ~ H + RBI + P.PA + Adj.TB, data = data_scaled)
summary(scaled_model)
```

```
##
## Call:
## lm(formula = HR ~ H + RBI + P.PA + Adj.TB, data = data_scaled)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
```

```
## -1.27222 -0.26483 -0.02389  0.28519  1.30819
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  7.934e-16  3.583e-02   0.000    1.000
## H            1.119e+00  1.240e-01   9.021 1.35e-15 ***
## RBI          4.606e-01  5.287e-02   8.713 7.90e-15 ***
## P.PA         9.681e-02  4.006e-02   2.417    0.017 *
## Adj.TB      -1.282e+00  1.143e-01 -11.212  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4299 on 139 degrees of freedom
## Multiple R-squared:  0.8203, Adjusted R-squared:  0.8152
## F-statistic: 158.7 on 4 and 139 DF,  p-value: < 2.2e-16
```

```r
# rounding the number to make easier to read and we are standardizing the model
round(coef(summary(scaled_model)),2)
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)     0.00       0.04    0.00     1.00
## H               1.12       0.12    9.02     0.00
## RBI             0.46       0.05    8.71     0.00
## P.PA            0.10       0.04    2.42     0.02
## Adj.TB         -1.28       0.11  -11.21     0.00
```