Predicting Baseball Wins using Multiple Linear Regression

DATA621 Homework 01

William Outcault, Kevin Potter, Mengqin Cai, Philip Tanofsky, Robert Welk, Zhi Ying Chen

27 September 2020

Assignment

In this homework assignment, you will explore, analyze and model a data set containing approximately 2200 records. Each record represents a professional baseball team from the years 1871 to 2006 inclusive. Each record has the performance of the team for the given year, with all of the statistics adjusted to match the performance of a 162 game season.

We have been given a dataset with 2276 records summarizing a major league baseball team's season. The records span 1871 to 2006 inclusive. All statistics have been adjusted to match the performance of a 162 game season.

Your objective is to build a multiple linear regression model on the training data to predict the number of wins for the team. You can only use the variables given to you (or variables that you derive from the variables provided).

DATA EXPLORATION

Before modeling it's important you understand your data. The following cells use exploratory data methods to understand the distribution of the data. The techniques used will explore differences feature type, summary statistics, number of missing values, and a visual representation of the density of each variable.

```
# Load dataframes without index column
raw_mbed <- read.csv("https://raw.githubusercontent.com/Zchen116/data-621/master/moneyball-evaluation-da
raw_mbtd <- read.csv("https://raw.githubusercontent.com/Zchen116/data-621/master/moneyball-training-data
mbed <- raw_mbed[,-1]
mbtd <- raw_mbtd[,-1]
```

```
# Check variable types
sapply(mbtd, class)
```

```
##      TARGET_WINS   TEAM_BATTING_H  TEAM_BATTING_2B  TEAM_BATTING_3B
##        "integer"        "integer"        "integer"        "integer"
##  TEAM_BATTING_HR  TEAM_BATTING_BB  TEAM_BATTING_SO  TEAM_BASERUN_SB
##        "integer"        "integer"        "integer"        "integer"
##  TEAM_BASERUN_CS TEAM_BATTING_HBP  TEAM_PITCHING_H TEAM_PITCHING_HR
##        "integer"        "integer"        "integer"        "integer"
## TEAM_PITCHING_BB TEAM_PITCHING_SO  TEAM_FIELDING_E TEAM_FIELDING_DP
##        "integer"        "integer"        "integer"        "integer"
```

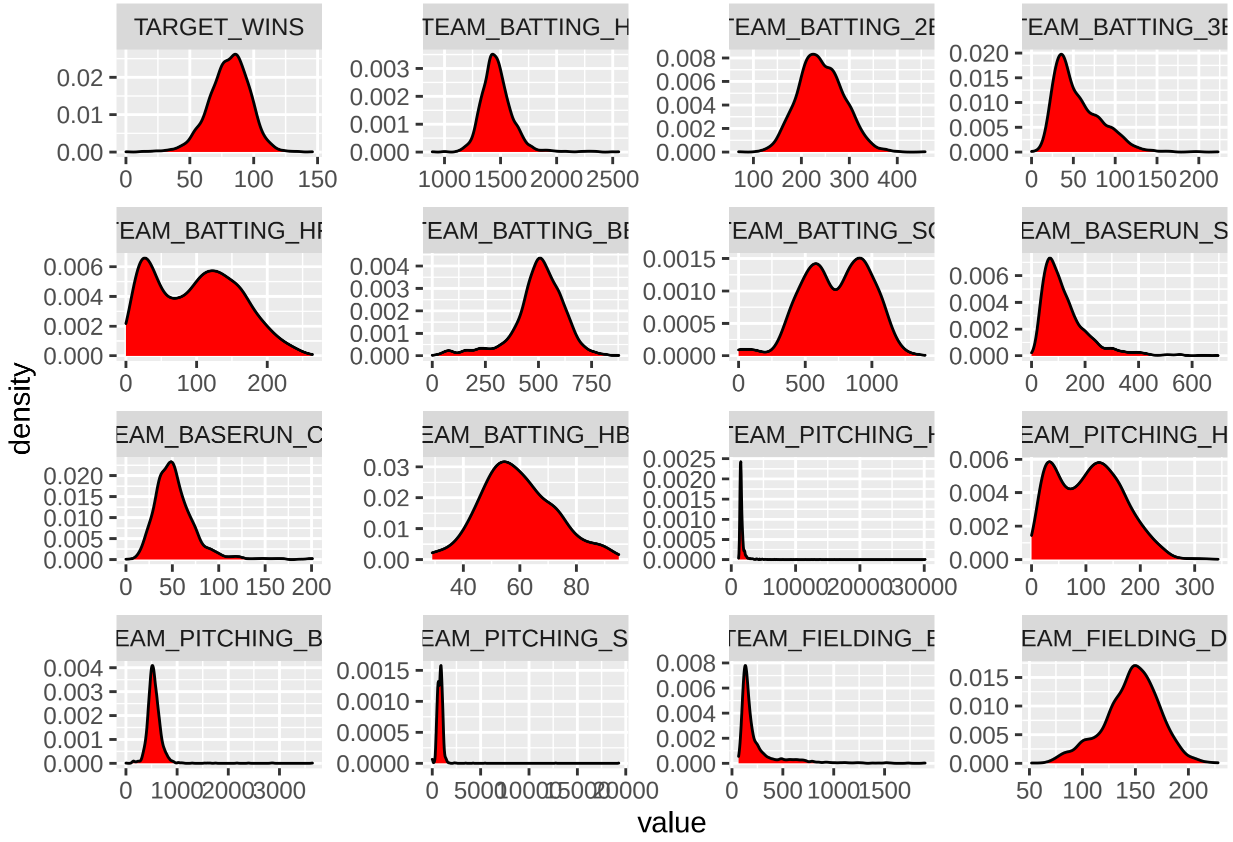
```
# Summarize variables
summary(mbtd)
```

```
##   TARGET_WINS     TEAM_BATTING_H TEAM_BATTING_2B TEAM_BATTING_3B
##  Min.   :  0.00   Min.   : 891   Min.   : 69.0   Min.   :  0.00
##  1st Qu.: 71.00   1st Qu.:1383   1st Qu.:208.0   1st Qu.: 34.00
##  Median : 82.00   Median :1454   Median :238.0   Median : 47.00
##  Mean   : 80.79   Mean   :1469   Mean   :241.2   Mean   : 55.25
##  3rd Qu.: 92.00   3rd Qu.:1537   3rd Qu.:273.0   3rd Qu.: 72.00
##  Max.   :146.00   Max.   :2554   Max.   :458.0   Max.   :223.00
##
##  TEAM_BATTING_HR  TEAM_BATTING_BB TEAM_BATTING_SO  TEAM_BASERUN_SB
##  Min.   :  0.00   Min.   :  0.0   Min.   :   0.0   Min.   :  0.0
##  1st Qu.: 42.00   1st Qu.:451.0   1st Qu.: 548.0   1st Qu.: 66.0
##  Median :102.00   Median :512.0   Median : 750.0   Median :101.0
##  Mean   : 99.61   Mean   :501.6   Mean   : 735.6   Mean   :124.8
##  3rd Qu.:147.00   3rd Qu.:580.0   3rd Qu.: 930.0   3rd Qu.:156.0
##  Max.   :264.00   Max.   :878.0   Max.   :1399.0   Max.   :697.0
##                                   NA's   :102      NA's   :131
##  TEAM_BASERUN_CS TEAM_BATTING_HBP TEAM_PITCHING_H TEAM_PITCHING_HR
##  Min.   :  0.0   Min.   :29.00    Min.   : 1137   Min.   :  0.0
##  1st Qu.: 38.0   1st Qu.:50.50    1st Qu.: 1419   1st Qu.: 50.0
##  Median : 49.0   Median :58.00    Median : 1518   Median :107.0
##  Mean   : 52.8   Mean   :59.36    Mean   : 1779   Mean   :105.7
##  3rd Qu.: 62.0   3rd Qu.:67.00    3rd Qu.: 1682   3rd Qu.:150.0
##  Max.   :201.0   Max.   :95.00    Max.   :30132   Max.   :343.0
##  NA's   :772     NA's   :2085
##  TEAM_PITCHING_BB TEAM_PITCHING_SO  TEAM_FIELDING_E  TEAM_FIELDING_DP
##  Min.   :   0.0   Min.   :     0.0   Min.   :  65.0   Min.   : 52.0
##  1st Qu.: 476.0   1st Qu.:   615.0   1st Qu.: 127.0   1st Qu.:131.0
##  Median : 536.5   Median :   813.5   Median : 159.0   Median :149.0
##  Mean   : 553.0   Mean   :   817.7   Mean   : 246.5   Mean   :146.4
##  3rd Qu.: 611.0   3rd Qu.:   968.0   3rd Qu.: 249.2   3rd Qu.:164.0
##  Max.   :3645.0   Max.   :19278.0   Max.   :1898.0   Max.   :228.0
##                   NA's   :102                         NA's   :286
```

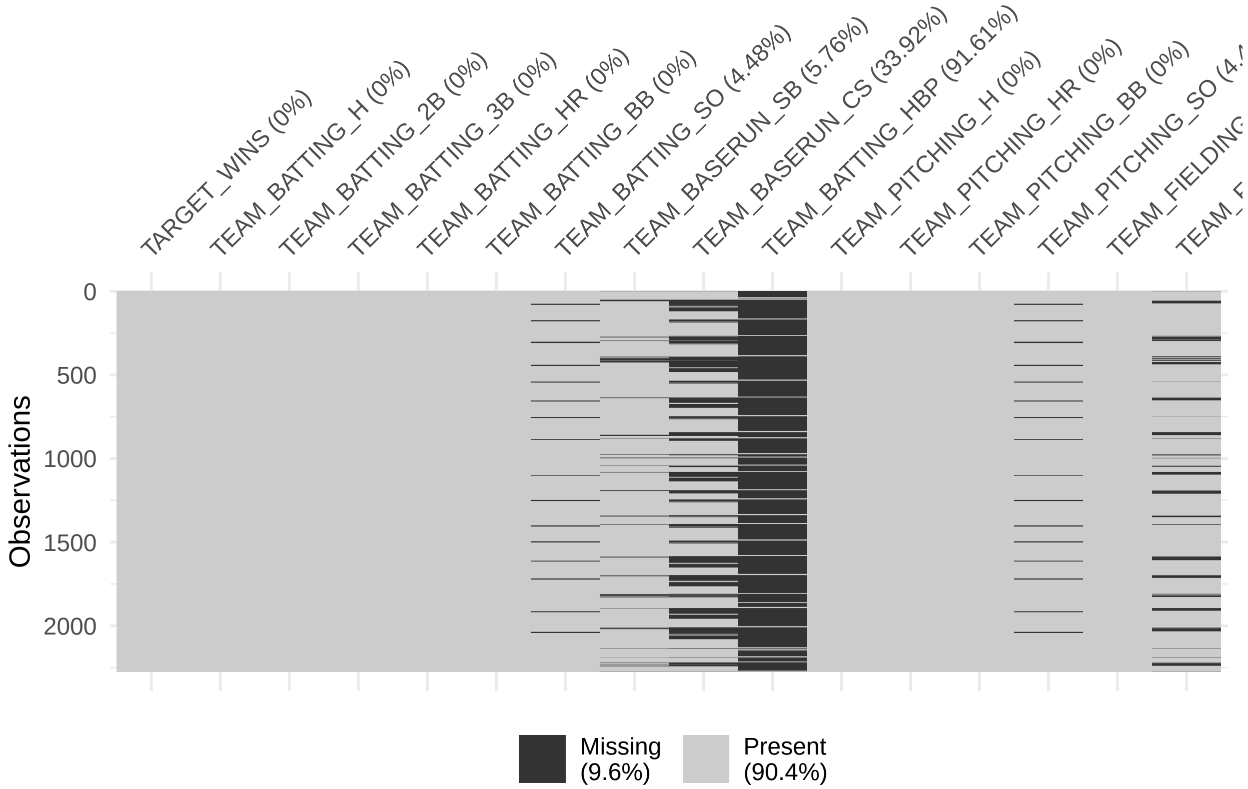
```
# Plot distributions
par(mfrow = c(4, 4))

datasub = melt(mbtd)
```

```
## Using  as id variables
```

```
ggplot(datasub, aes(x= value)) +
    geom_density(fill='red') + facet_wrap(~variable, scales = 'free')
```



```
# Show missing values
vis_miss(mbtd)
```



Let's go piece by piece through the information we uncovered.

Variable Type: We are dealing with only numerical types.
Summary: We have zero values which doesn't seem feasible within the context of the analysis.
Distributions: We have bimodal and skewed distributions.
Missing Values: The `TEAM_BATTING_HBP` and `TEAM_BASERUN_CS` variables are each missing over a third of their values.

Data Preparation

We begin by splitting the data into a training and testing set using a 75/25 split. Next each zero value is set to NA because zero is not exactly feasible in this context therefore we treat the data as missing or an anomaly. Following this, the variables with over 10% missing values were removed and only the complete cases were used for the training set. Lastly we view the distributions to better understand the prepped data.

```
# Split train/test data
set.seed(100)

smp_size <- floor(0.75 * nrow(raw_mbtd))

train_ind <- sample(seq_len(nrow(raw_mbtd)), size = smp_size)

train <- raw_mbtd[train_ind, -1]
test <- raw_mbtd[-train_ind, -1]
```

```
dim(train)
```

```
## [1] 1707   16
```

```
dim(test)
```

```
## [1] 569  16
```

```
# Set 0 equal to NA
train[train == 0] <- NA
test[is.na(test)] <- 0
```

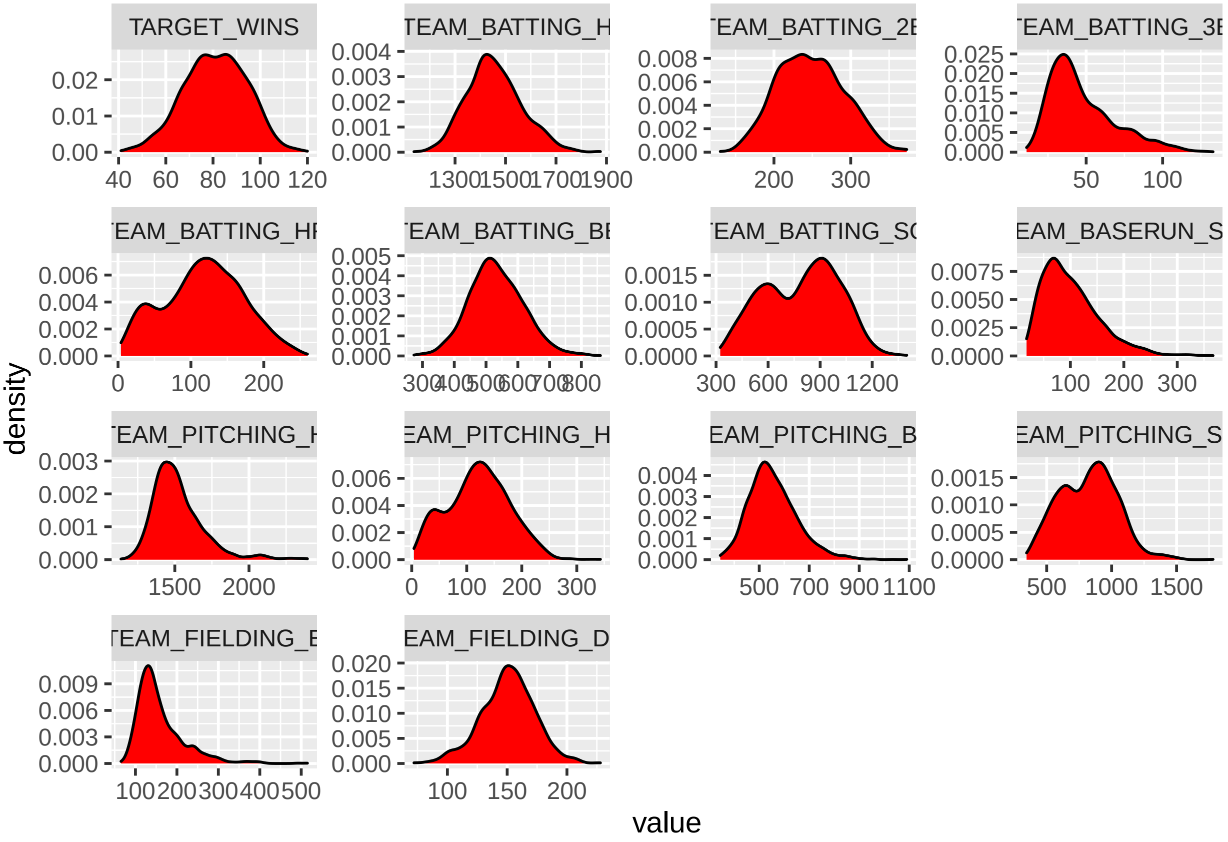
```
# Remove variables with excessive missing values
train <- dplyr::select(train, -TEAM_BATTING_HBP, -TEAM_BASERUN_CS)
test <- dplyr::select(test, -TEAM_BATTING_HBP, -TEAM_BASERUN_CS)
```

```
# Filter complete cases
train <- train[complete.cases(train),]
```

```
# Plot clean distributions
par(mfrow = c(4, 4))

datasub = melt(train)
```

```
## Using  as id variables
```

```
ggplot(datasub, aes(x= value)) +
    geom_density(fill='red') + facet_wrap(~variable, scales = 'free')
```



Our distributions look much more normally distributed.

Build Model

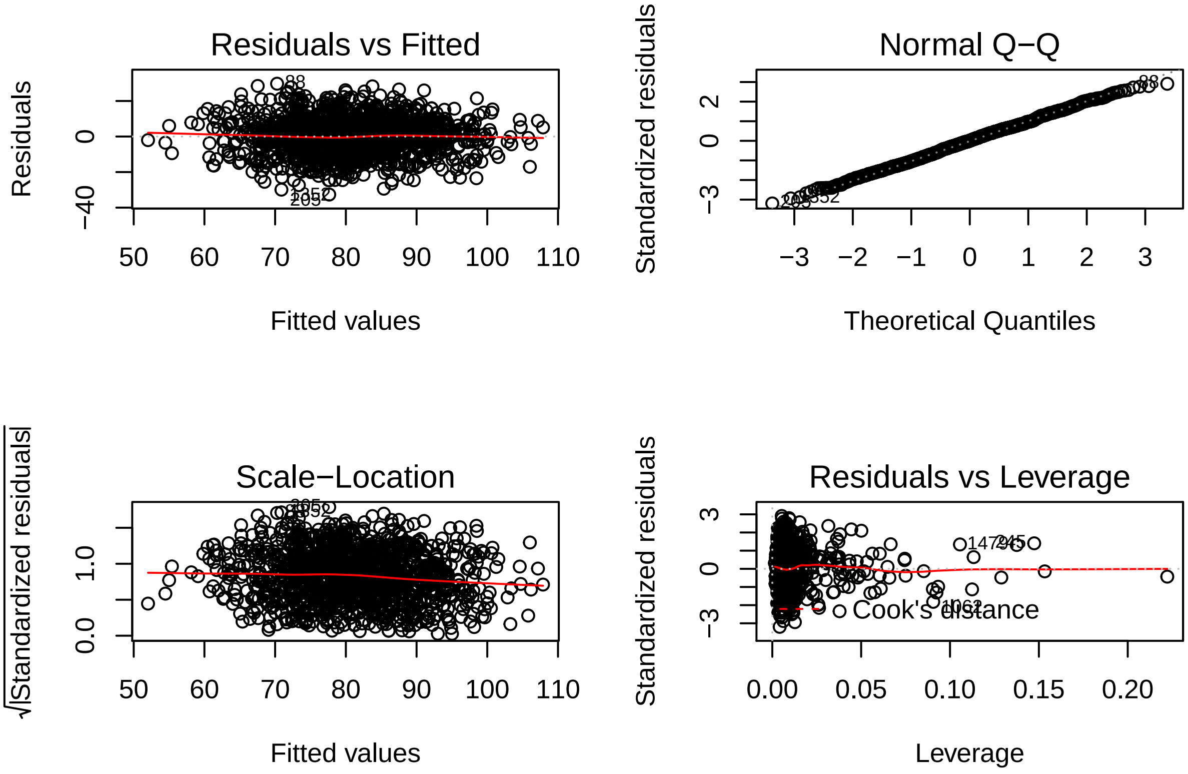
After the data has been cleaned and analyzed we are ready to create a linear regression model to predict team wins. It's important to note we are only using the training data to create our best fit line. We use the testing data set to evaluate the model on unseen data and prevent overfitting. The following cells explore three methods for creating a model.

Raw Model

The raw model is our base for evaluation. The raw model uses the base `lm` package to create the best fit line. We fit the model on the training dataset using all of the features that met the data preparation criteria and evaluate the performance.

```
# Build stepwise model
raw_model <- lm(TARGET_WINS ~ .,
            data = train)
```

```
par(mfrow = c(2, 2))
plot(raw_model)
```



```
summary(raw_model)
```

```
##
## Call:
## lm(formula = TARGET_WINS ~ ., data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -32.649  -7.197   0.041   6.871  29.723
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      53.494993   7.078501   7.557 7.56e-14 ***
## TEAM_BATTING_H   -0.022496   0.018879  -1.192  0.23364
## TEAM_BATTING_2B  -0.062370   0.010397  -5.999 2.55e-09 ***
## TEAM_BATTING_3B   0.181747   0.022614   8.037 2.00e-15 ***
## TEAM_BATTING_HR   0.076105   0.097768   0.778  0.43646
## TEAM_BATTING_BB   0.105832   0.048941   2.162  0.03076 *
## TEAM_BATTING_SO   0.036566   0.027663   1.322  0.18645
## TEAM_BASERUN_SB   0.069201   0.006389  10.832  < 2e-16 ***
## TEAM_PITCHING_H   0.053786   0.017216   3.124  0.00182 **
## TEAM_PITCHING_HR  0.019047   0.093823   0.203  0.83916
## TEAM_PITCHING_BB -0.064902   0.046544  -1.394  0.16342
## TEAM_PITCHING_SO -0.055724   0.026409  -2.110  0.03504 *
## TEAM_FIELDING_E  -0.117770   0.008528 -13.809  < 2e-16 ***
## TEAM_FIELDING_DP -0.117371   0.014551  -8.066 1.59e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.24 on 1347 degrees of freedom
## Multiple R-squared:  0.4087, Adjusted R-squared:  0.403
## F-statistic: 71.61 on 13 and 1347 DF,  p-value: < 2.2e-16
```

```
#Calculate RMSE and R.Squared
predictions <- predict.lm(raw_model, newdata = test[,-1])

rmse <- rmse(test[,1], predictions)

R.sq <- summary(raw_model)$adj.r.squared

raw <- cbind(rmse, R.sq)
raw
```

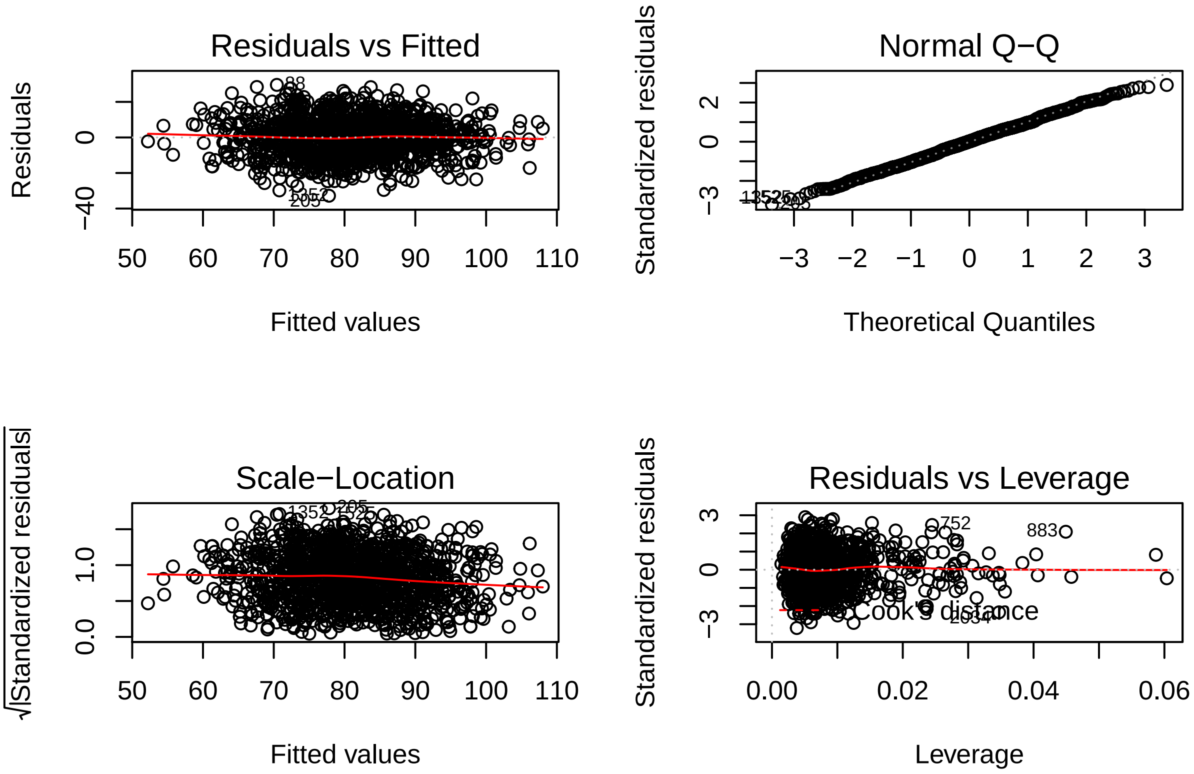
```
##          rmse      R.sq
## [1,] 49.33253 0.4029764
```

Stepwise Model

This model uses the raw model created above with the addition of the `stepAIC` package. `stepAIC` is a common package used to help with feature selection. This version of the model uses this package with no additional constraints to train and evaluate model performance.

```r
stepwise_model <- stepAIC(raw_model, direction = c("both"), trace = FALSE)
```

```r
par(mfrow = c(2, 2))
plot(stepwise_model)
```



```
summary(stepwise_model)
```

```
##
## Call:
## lm(formula = TARGET_WINS ~ TEAM_BATTING_2B + TEAM_BATTING_3B +
##     TEAM_BATTING_HR + TEAM_BATTING_BB + TEAM_BATTING_SO + TEAM_BASERUN_SB +
##     TEAM_PITCHING_H + TEAM_PITCHING_SO + TEAM_FIELDING_E + TEAM_FIELDING_DP,
##     data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -32.801  -7.255   0.179   6.981  29.563
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      52.318016   6.729296   7.775 1.49e-14 ***
## TEAM_BATTING_2B  -0.063941   0.010033  -6.373 2.54e-10 ***
## TEAM_BATTING_3B   0.179940   0.022360   8.047 1.84e-15 ***
## TEAM_BATTING_HR   0.096079   0.010436   9.207  < 2e-16 ***
## TEAM_BATTING_BB   0.037738   0.003640  10.366  < 2e-16 ***
## TEAM_BATTING_SO   0.039743   0.009998   3.975 7.40e-05 ***
## TEAM_BASERUN_SB   0.069608   0.006370  10.927  < 2e-16 ***
## TEAM_PITCHING_H   0.033166   0.004407   7.525 9.56e-14 ***
## TEAM_PITCHING_SO -0.058534   0.008336  -7.022 3.46e-12 ***
## TEAM_FIELDING_E  -0.115813   0.008284 -13.981  < 2e-16 ***
## TEAM_FIELDING_DP -0.118113   0.014494  -8.149 8.27e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.24 on 1350 degrees of freedom
## Multiple R-squared:  0.4078, Adjusted R-squared:  0.4034
## F-statistic: 92.96 on 10 and 1350 DF,  p-value: < 2.2e-16
```

```
#Calculate RMSE and R.Squared
predictions2 <- predict.lm(stepwise_model, newdata = test[,-1])

rmse2 <- rmse(test[,1], predictions2)

R.sq2 <- summary(stepwise_model)$adj.r.squared

stepwise <- cbind(rmse2, R.sq2)

stepwise
```

```
##        rmse2     R.sq2
## [1,] 32.5123 0.4034098
```

Stepwise with Constraints

Here we add constraints to the stepwise model created to tune model performance. Tuning a model is a technique used in data science to direct the model to adjust the feature weights to minimize a specific loss function. In this case we use RMSE (root-mean-square deviation).

```
#Calculate RMSE and R.Squared
predictions2 <- set_constraints(predictions2)

rmse3 <- rmse(test[,1], predictions2)

R.sq3 <- summary(stepwise_model)$adj.r.squared

constrained_stepwise <- cbind(rmse3, R.sq3)

constrained_stepwise
```

```
##        rmse3     R.sq3
## [1,] 17.8522 0.4034098
```

Select Model

Lastly all the models will be compared in order to select the model that will produce the most accurate and precise results. The metrics we will be focused on are RMSE and adjusted R-Squared. The models compared were the original raw model which included all variables. Next was a stepwise model which minimizes AIC in order to determine the variables which are necessary to include. The last model was a stepwise model with constraints implemented on the predictions.

```
kk = rbind(round(raw, 4), round(stepwise, 4), round(constrained_stepwise, 4))
k1 = as.data.frame(kk)
rownames(k1) = c("raw", "stepwise", "constrained_stepwise")
k1 %>%
  kable() %>%
  kable_styling(bootstrap_options = c('striped','bordered'), full_width = FALSE)
```

	rmse	R.sq
raw	49.3325	0.4030
stepwise	32.5123	0.4034
constrained_stepwise	17.8522	0.4034

Our third model produced the best metrics. Including constraints on a stepwise model produced an RMSE and adjusted R-squared of 17.8522 and 0.4034 respectively.

```
preds <- cbind(head(predictions2, 10), head(test[,1], 10))

k2 = as.data.frame(preds)
colnames(k2) = c("Predictions", "Actual")
k2 %>%
  kable() %>%
  kable_styling(bootstrap_options = c('striped','bordered'), full_width = FALSE)
```

	Predictions	Actual
9	73.68378	86
10	65.47806	76
19	74.78397	75
22	78.05182	81
27	78.60546	91
35	88.24212	84
39	81.79130	75
40	87.09483	99
41	86.61000	77
45	99.61626	100

Write Predictions

```
mbed[is.na(mbed)]<-0
final_predictions <- round(predict.lm(stepwise_model, newdata = mbed), 3)
final_predictions <- set_constraints(final_predictions)
final_predictions <- cbind(TARGET_WINS=final_predictions, mbed)
write.csv(final_predictions, "moneyball-prediction-data.csv")
```