Assignment1_Data621

Mubashira Qari, Marco Castro, Puja Roy, Zach Rose, Erick Hadi

2025-02-09

```
##
## 1 function (expr, envir = parent.frame(), enclos = if (is.list(envir) ||
## 2     is.pairlist(envir)) parent.frame() else baseenv())
## 3 .Internal(eval(expr, envir, enclos))
```

```
##
## 1 function (x, ...)
## 2 {
## 3     UseMethod("train")
```

Section 1: Data Exploration

In this section, we perform exploratory analysis to understand our dataset and identify potential issues such as missing values and outliers

Shape of Data

Our evaluation dataframe consists of 16 variables (excluding an INDEX field) all consisting of integer values. A cursory look demonstrates the presence of null values in some of our fields. Note that the TARGET_WINS field — the number of games a team won in a given baseball season — wil be used as the dependent variable in our analysis.

```
str(train_df)
```

```
## 'data.frame':    2276 obs. of  16 variables:
##  $ TARGET_WINS     : int  39 70 86 70 82 75 80 85 86 76 ...
##  $ TEAM_BATTING_H  : int  1445 1339 1377 1387 1297 1279 1244 1273 1391 1271 ...
##  $ TEAM_BATTING_2B : int  194 219 232 209 186 200 179 171 197 213 ...
##  $ TEAM_BATTING_3B : int  39 22 35 38 27 36 54 37 40 18 ...
##  $ TEAM_BATTING_HR : int  13 190 137 96 102 92 122 115 114 96 ...
##  $ TEAM_BATTING_BB : int  143 685 602 451 472 443 525 456 447 441 ...
##  $ TEAM_BATTING_SO : int  842 1075 917 922 920 973 1062 1027 922 827 ...
##  $ TEAM_BASERUN_SB : int  NA 37 46 43 49 107 80 40 69 72 ...
##  $ TEAM_BASERUN_CS : int  NA 28 27 30 39 59 54 36 27 34 ...
##  $ TEAM_BATTING_HBP: int  NA NA NA NA NA NA NA NA NA NA ...
##  $ TEAM_PITCHING_H : int  9364 1347 1377 1396 1297 1279 1244 1281 1391 1271 ...
##  $ TEAM_PITCHING_HR: int  84 191 137 97 102 92 122 116 114 96 ...
##  $ TEAM_PITCHING_BB: int  927 689 602 454 472 443 525 459 447 441 ...
##  $ TEAM_PITCHING_SO: int  5456 1082 917 928 920 973 1062 1033 922 827 ...
##  $ TEAM_FIELDING_E : int  1011 193 175 164 138 123 136 112 127 131 ...
##  $ TEAM_FIELDING_DP: int  NA 155 153 156 168 149 186 136 169 159 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

Analysis of the training Dataset

Before fitting a multiple linear regression (MLR) model, we analyze the dataset for potential issues such as missing values, extreme outliers, multicollinearity, and variable distributions.

The summary statistics below give us the mean, median, interquartile range, standard deviation and number of missing variables for the charts.

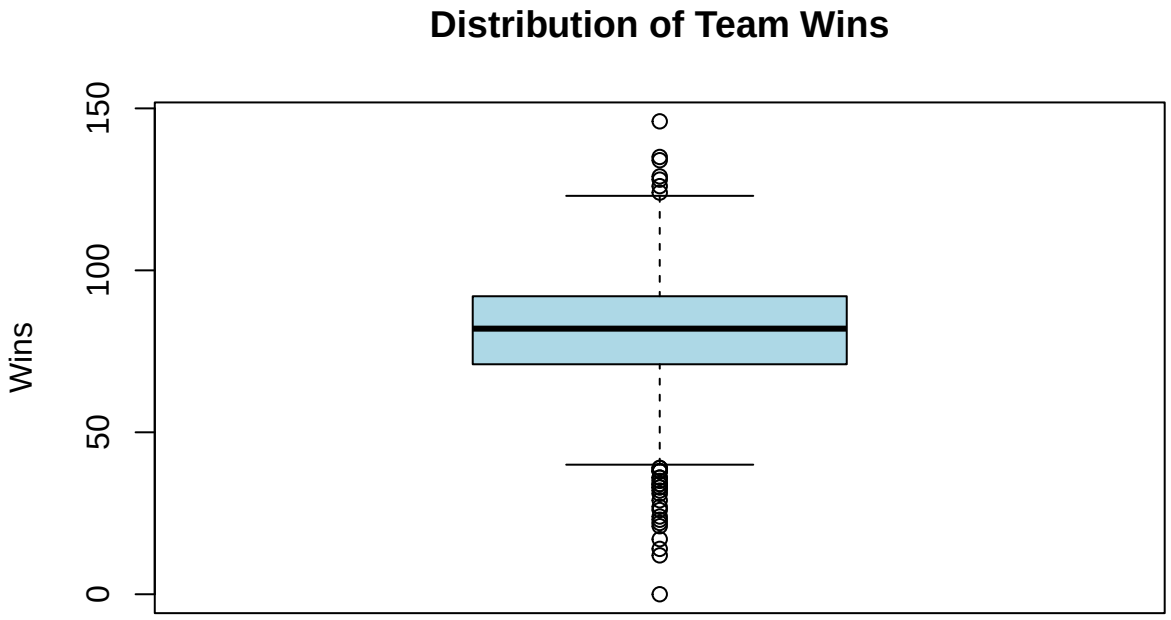
```
# skim(train_df) # won't knit into pdf
summary(train_df)
```

```
##   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
```

Here's what we can infer from the summary statistics:

1. TARGET_WINS The min value of 0 and max of 146 suggest some potential outliers or erroneous data points, since most teams win between 50-110 games in a season. Below we examine a Box Plot of TARGET_WINS.

```
boxplot(train_df$TARGET_WINS, main="Distribution of Team Wins", ylab="Wins", col="lightblue")
```



The Box Plot for Outlier Detection & Distribution Analysis shows an Interquartile Range - IQR with a range roughly from 70 to 92 wins.

The Box Plot suggests that there are several small circles (outliers) below the lower whisker. Additionally, there are some outliers above the upper whisker, but visually fewer than the low-end.

Since The TARGET_WINS column has missing values, we should remove those rows since we can't predict missing outcomes. Similarly, if it has a zero (0) value, we may also want to drop this row, as it is highly suspicious that a team would have 0 wins in 162 games.

```
train_df <- train_df %>%
  filter(!is.na(TARGET_WINS) & TARGET_WINS >0)
```

2. Missing Values Some variables have a significant number of missing values. In particular:

- TEAM_BATTING_HBP (2085 missing values) <- very unreliable
- TEAM_BASERUN_CS (772 missing values) <- potentially unreliable
- TEAM_BATTING_SO (102 missing values)
- TEAM_BASERUN_SB (131 missing values)
- TEAM_PITCHING_SO (102 missing values)
- TEAM_FIELDING_DP (286 missing values)

Additionally, four variables have values of zero (0) reported that appear suspicious. In particular: - TEAM_BATTING_SO & TEAM_PITCHING_SO have the same rows entered as zero suggesting that data may not have been available for these entries. - TEAM_BATTING_HR & TEAM_PITCHING_HR have the same rows entered as zero suggesting that data may not have been available for these entries.

Actionable Steps: - Removing TEAM_BATTING_HBP since most of it's values are missing. - Impute missing values (later on).

```
train_df <- train_df[, !names(train_df) %in% "TEAM_BATTING_HBP"]
```

Faceted Scatter Plot with Linear Regression Lines:

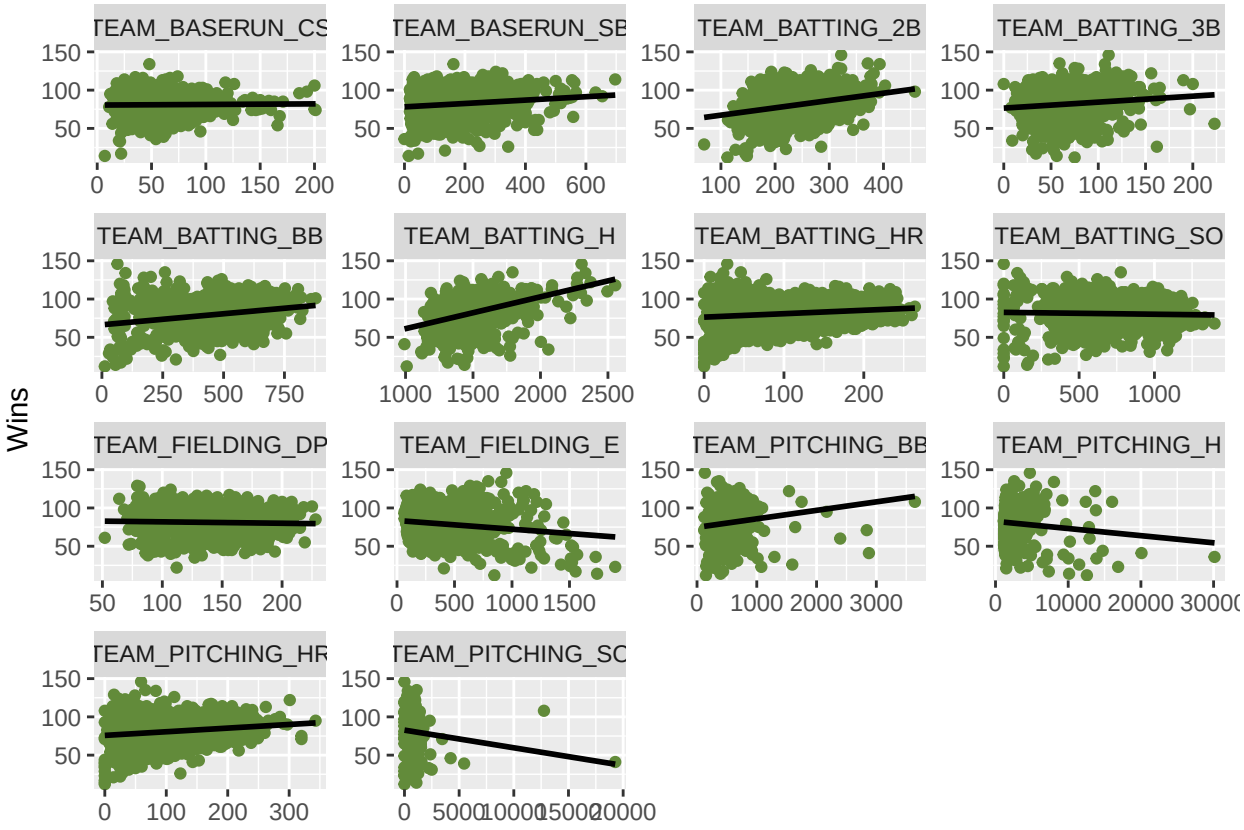
These scatter plots give us a sense of the relationship between the each variable and TARGET_WINS. Data points are plotted where the x-axis represents the predictor variable, and the y-axis represents the number of wins. A black trend line is fitted using linear regression to show the general direction and strength of the relationship between each variable and TARGET_WINS.

```
train_df %>%
  gather(variable, value, -TARGET_WINS) %>%
  ggplot(., aes(value, TARGET_WINS)) +
  geom_point(fill = "#628B3A", color="#628B3A")  +
  geom_smooth(method = "lm", se = FALSE, color = "black") +
  facet_wrap(~variable, scales ="free", ncol = 4) +
  labs(x = element_blank(), y = "Wins")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

```
## Warning: Removed 1392 rows containing non-finite outside the scale range
## (`stat_smooth()`).
```

```
## Warning: Removed 1392 rows containing missing values or values outside the scale range
## (`geom_point()`).
```



The slope of the regression line in each facet is used to determine the strength of relationship between the independent variable represent on the x-axis vs the dependent variable y (TARGET_WINS). The steeper the slope, the stronger the relationship is between the two variables. The direction of the slope tells whether the relationship is positive or negative: - if the line is sloped to the right, it is a positive relationship meaning we can expect an increase in y as x increases - if the line is sloped to the left, it is a negative relationship meaning we can expect an decrease in y as x increases - if the trend line is flat, there is likely no meaningful relationship between that variable and TARGET_WINS.

If the points are closely clustered around the line, it suggests a stronger linear relationship. If the points are widely scattered, the variable may not strongly predict TARGET_WINS.

Positive Predictors of Wins: TEAM_BATTING_2B, TEAM_BATTING_BB, TEAM_BATTING_H, TEAM_PITCHING_BB (unexpected).

Negative Predictors of Wins: TEAM_FIELDING_E, TEAM_PITCHING_H, TEAM_PITCHING_SO.

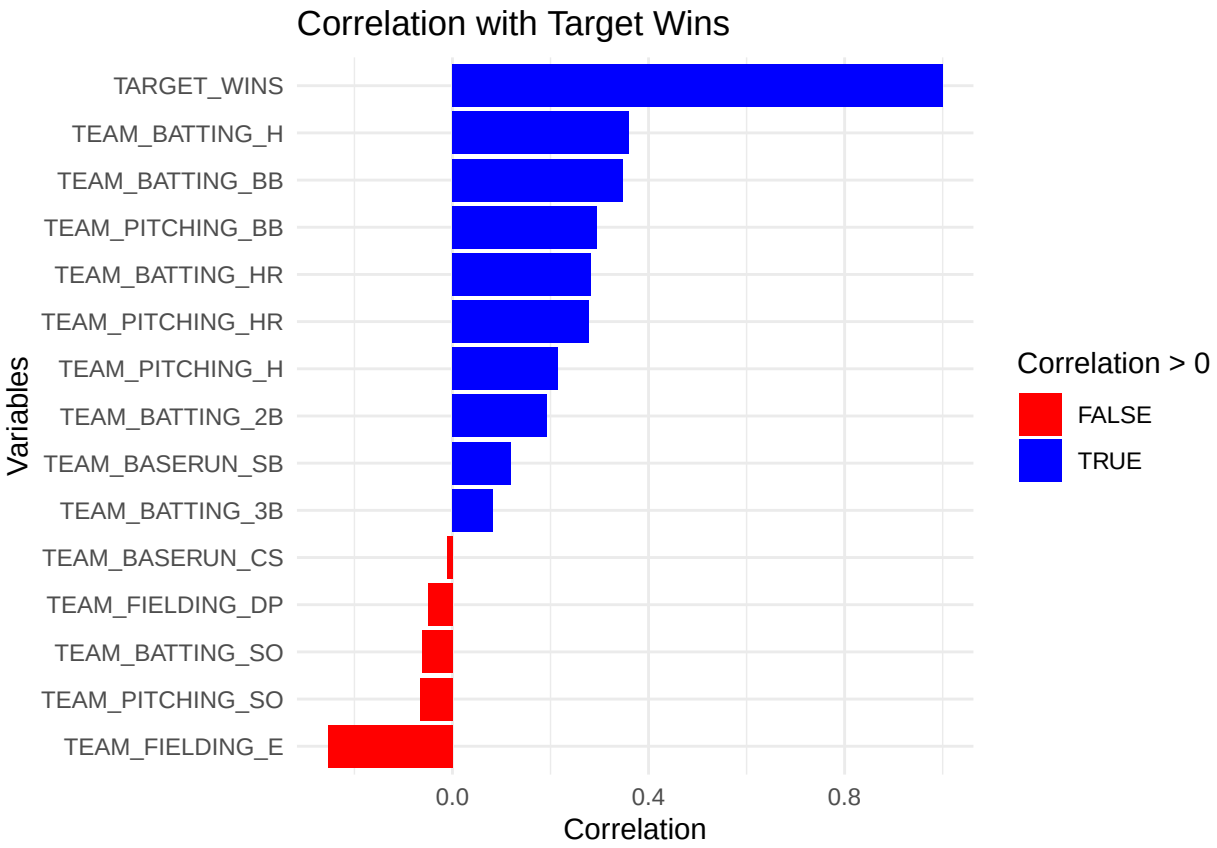
Weak or No Influence: TEAM_BATTING_3B, TEAM_BATTING_SO, TEAM_FIELDING_DP, TEAM_PITCHING_HR, , TEAM_BATTING_HBP (Removed).

Correlations

In this section, we examine two different types of correlations: - correlation between dependent and independent variables - correlation between all variables

Correlation Between Dependent and Independent variables The correlation between the dependent variable TARGET_WINS and each of the independent variables. In this context, correlation pertains to the strength (0:1) and direction (+/-) of the relationship between the dependent and independent variable. A higher strength is indicated by a number closer to 1 and describes a greater change on the dependent variable by the independent variable. The direction describes whether the change is increasing (positive) or decreasing (negative). The chart below suggest that there is a relatively weak relationship between the TARGET_WINS and TEAM_BASERUN_SB and should be considered for omission from our models.

```
correlation_with_target <- cor(train_df, use = "complete.obs")["TARGET_WINS", ] %>%
  sort(decreasing = TRUE)  # Sort from highest to lowest correlation

correlation_data <- data.frame(Variable = names(correlation_with_target), Correlation = correlation_wit

ggplot(correlation_data, aes(x = reorder(Variable, Correlation), y = Correlation, fill = Correlation > 
  geom_bar(stat = "identity") +
  coord_flip() +  # Flip for better readability
  labs(title = "Correlation with Target Wins", x = "Variables", y = "Correlation") +
  scale_fill_manual(values = c("red", "blue")) +
  theme_minimal()
```



Correlation Between All Variables Testing the correlation between all variables shows how the change in one variable affects the change in another. In this case, a high correlation value between two independent variables, regardless of direction, suggests multicollinearity between the variables meaning that they are not independent from one another and thus breaks our assumption of independence.

```
cor_matrix <- cor(train_df, use = "complete.obs")
cor_df <- as.data.frame(as.table(cor_matrix))
cor_df <- cor_df[cor_df$Var1 != cor_df$Var2, ]
cor_df <- cor_df[order(cor_df$Freq, decreasing = TRUE), ]
```

The correlation matrix shows that the following variables are very highly correlated with one another:

Top 3 Positive Correlation

```
head(cor_df, 3)
```

```
##                  Var1             Var2      Freq
## 71   TEAM_PITCHING_HR  TEAM_BATTING_HR 0.9716522
## 155   TEAM_BATTING_HR TEAM_PITCHING_HR 0.9716522
## 103 TEAM_PITCHING_SO  TEAM_BATTING_SO 0.9322116
```

Top 3 Negative Correlation

```
head(cor_df[order(cor_df$Freq), ], 3)
```

```
##                Var1            Var2       Freq
## 52   TEAM_BATTING_SO TEAM_BATTING_3B -0.6903472
## 94   TEAM_BATTING_3B TEAM_BATTING_SO -0.6903472
## 58 TEAM_PITCHING_SO TEAM_BATTING_3B -0.6392928
```

Potential Outliers & Data Issues

We observed the following values that seem suspicious:

- TEAM_PITCHING_H (Max = 30,132) <- Likely an error since typical values range from 1,200 - 1,700.
- TEAM_PITCHING_SO (Max = 19,278) <- Suspiciously high (typical range: 500 - 1,500).
- TEAM_PITCHING_BB (Max = 3,645) <- Very high (typical range: 300 - 700).
- TEAM_FIELDING_E (Max = 1,898) <- Likely an error since the normal range is ~ 70-200.

We will take a look at only potential outliers that have a significant leverage when we start fitting our model.

Section 3: Data Preparation

Before fitting our model, we will consider the following data cleaning steps:

- Consider dropping or imputing variables with too many missing values (e.g., TEAM_BATTING_HBP).
- Remove or Adjust Extreme Outliers

Once cleaned, feature selection and multicollinearity checks will be essential to ensure a robust and interpretable model for predicting team wins.

Identifying Missing Values We previously identified several variables with many missing fields. Our first step was to drop *TEAM_BATTING_HBP* as most of its values were missing. In this next section, we will address the following missing values

```
missing_values <- train_df %>%
  summarise(across(everything(), ~ sum(is.na(.)))) %>%
  pivot_longer(cols = everything(), names_to = "Variable", values_to = "Missing_Count") %>%
  filter(Missing_Count > 0) %>%
  arrange(desc(Missing_Count))

print(missing_values)
```

```
## # A tibble: 5 x 2
##   Variable         Missing_Count
##   <chr>                    <int>
## 1 TEAM_BASERUN_CS            772
## 2 TEAM_FIELDING_DP           285
## 3 TEAM_BASERUN_SB            131
## 4 TEAM_BATTING_SO            102
## 5 TEAM_PITCHING_SO           102
```

The variables *TEAM_BATTING_SO*, *TEAM_PITCHING_SO*, *TEAM_BATTING_HR*, and *TEAM_PITCHING_HR* included several observations with a value of zero. As the rows were the same for both variables, it appears that these may also be missing observations as the likelihood that a team's batters did not have a single strikeout nor did their pitchers pitch a single strikeout over the course of a 162 game season is highly unlikely. We will therefore treat these as missing observations and impute using the median value which is more robust to outliers then using the mean values. We will also drop the single row where the team did not win a single game, as this is also suspicious.

Handling Missing Values Next, we will address the remaining missing values. We will weight several options:

Removing Missing Values Dropping missing values could result in significantly reducing the sample size and thus the predictive power of your model. It can also introduce bias if the missing data is not missing completely at random (MCAR) or if too many observations are dropped from certain categories. We will therefore explore other methods.

Mean Imputation Mean imputation makes the imputed values less variable and could lead to an underestimation of the variability in your model. It may also over-simplify the observations and create artificial relationships. It can also introduce bias if the missing data is not missing completely at random (MCAR) or if too many observations are dropped from certain categories.

Median Imputation Median imputation has many of the same issues as mean imputation but is more robust to the effects of skewing and outliers. However, it can also introduce bias if the missing data is not missing completely at random (MCAR) or if too many observations are dropped from certain categories.

Regression Imputation Regression Imputation uses predictive models to predict missing values based on our dataframe and is one of the suggested techniques for values Missing at Random (MAR). However, if data is not Missing at Random (MAR), we can inadvertently introduce bias in our data.

Imputing Upon further analysis, we noticed a pattern between 4 parameters with missing values: TEAM_BATTING_SO, TEAM_PITCHING_SO, TEAM_BATTING_HR, and TEAM_PITCHING_HR suggesting that these missing values may be MAR. However, no pattern was evident for the other three parameters with missing values (TEAM_BASERUN_CS, TEAM_FIELDING_DP, TEAM_BASERUN_SB) suggesting that they may be MCAR. As such, we should apply two separate techniques for the missing values suitable for their specific classification. Specifically, we will apply:

- Median imputation to TEAM_BASERUN_CS, TEAM_FIELDING_DP, TEAM_BASERUN_SB as it is a more robust form of imputation for MCAR values than mean imputation which is more affected by outliers
- MICE imputation to TEAM_BATTING_SO, TEAM_PITCHING_SO, TEAM_BATTING_HR, and TEAM_PITCHING_HR as it as more robust method for MAR values.

```
library(miscTools)

train_df_median <- train_df
median_val <- colMedians(train_df_median, na.rm = TRUE)

# Impute using medians
for(i in c('TEAM_BASERUN_CS',   'TEAM_FIELDING_DP', 'TEAM_BASERUN_SB'))
    train_df_median[,i][is.na(train_df_median[,i])] <- median_val[i]
```

```
# Convert dubious stats to NAs for pitching
# and drop unnecessary columns
train_df_median <- train_df_median |>
  mutate(
    TEAM_BATTING_SO = if_else(TEAM_BATTING_SO > 0, TEAM_BATTING_SO, NA_integer_),
    TEAM_PITCHING_SO = if_else(TEAM_PITCHING_SO > 0, TEAM_PITCHING_SO, NA_integer_),
    TEAM_BATTING_HR = if_else(TEAM_BATTING_HR > 0, TEAM_BATTING_HR, NA_integer_),
    TEAM_PITCHING_HR = if_else(TEAM_PITCHING_HR > 0, TEAM_PITCHING_HR, NA_integer_)
  )
```

```
regimp <- lm(TEAM_BATTING_SO ~ . - TEAM_PITCHING_SO - TEAM_BATTING_HR - TEAM_PITCHING_HR, data = train_

# Predict missing values
train_df_median$TEAM_BATTING_SO[is.na(train_df_median$TEAM_BATTING_SO)] <- predict(regimp, newdata = tra

regimp <- lm(TEAM_PITCHING_SO ~ . - TEAM_BATTING_SO - TEAM_BATTING_HR - TEAM_PITCHING_HR, data = train_

# Predict missing values
train_df_median$TEAM_PITCHING_SO[is.na(train_df_median$TEAM_PITCHING_SO)] <- predict(regimp, newdata =
```

```
regimp <- lm(TEAM_BATTING_HR ~ . - TEAM_PITCHING_SO - TEAM_BATTING_SO - TEAM_PITCHING_HR, data = train_

# Predict missing values
train_df_median$TEAM_BATTING_HR[is.na(train_df_median$TEAM_BATTING_HR)] <- predict(regimp, newdata = tra

regimp <- lm(TEAM_PITCHING_HR ~ . - TEAM_PITCHING_SO - TEAM_BATTING_SO - TEAM_BATTING_HR, data = train_

# Predict missing values
train_df_median$TEAM_PITCHING_HR[is.na(train_df_median$TEAM_PITCHING_HR)] <- predict(regimp, newdata =

train_df_clean <- train_df_median |>
  filter(!TEAM_BATTING_HR < 0)

print(colSums(is.na(train_df_clean)))
```

```
##      TARGET_WINS   TEAM_BATTING_H  TEAM_BATTING_2B  TEAM_BATTING_3B
##                0                0                0                0
##  TEAM_BATTING_HR  TEAM_BATTING_BB  TEAM_BATTING_SO  TEAM_BASERUN_SB
##                0                0                0                0
##  TEAM_BASERUN_CS  TEAM_PITCHING_H TEAM_PITCHING_HR TEAM_PITCHING_BB
##                0                0                0                0
## TEAM_PITCHING_SO  TEAM_FIELDING_E TEAM_FIELDING_DP
##                0                0                0
```

Splitting the training dataset

We split the original training dataset into a training and testing dataset in order to test the strength of our model without testing the model on data that the model has already seen. The training dataset will contain 75% randomly selected observations from our original dataset of 2276 observations, while the test dataset will hold 25%. Some potential drawbacks of this technique are that we will make our sample size smaller, which may negatively affect our model if our sample size is small. As our split training set will still contain 1706 observations, it should still provide an adequate sample size. We also explore Cross-Validation techniques later on in our analysis.

```
smp_size <- floor(0.75 * nrow(train_df_clean))
 nrow(train_df_clean)
```

```
## [1] 2272
```

```
## set the seed to make your partition reproducible
set.seed(123)
train_ind <- sample(seq_len(nrow(train_df_clean)), size = smp_size)

stp75_train_df <- train_df_clean[train_ind, ]
stp25_test_df <- train_df_clean[-train_ind, ]
```

Handling Outliers

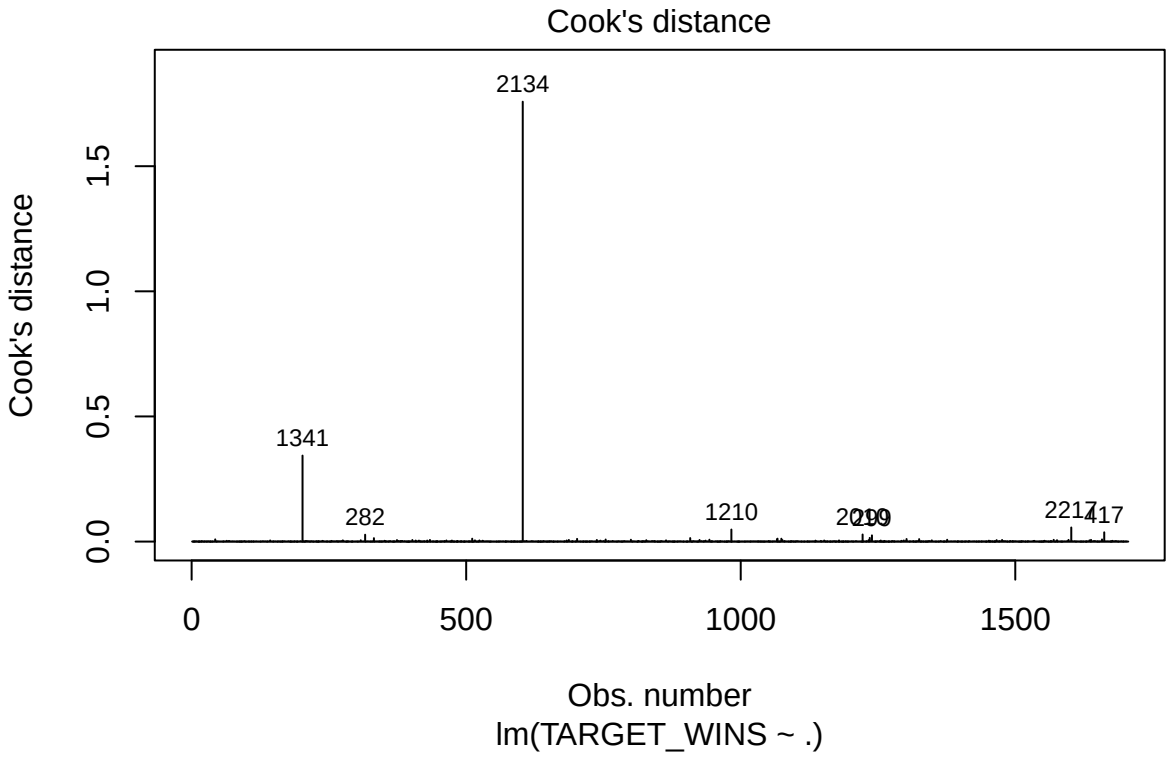
The Residuals vs. Fitted and QQ Plots show a fairly linear pattern, while Scale-Location plot suggest Homoscedasticity. However, the Residuals vs Leverage plot reveals the presence of some outliers.

```
stp_model_full <- lm(TARGET_WINS ~ ., data = stp75_train_df)
summary(stp_model_full)
```

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

```
## lm(formula = TARGET_WINS ~ ., data = stp75_train_df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -49.368  -8.679   0.066   8.688  56.218
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      20.4600045  6.3967791   3.198 0.001407 **
## TEAM_BATTING_H    0.0500658  0.0046097  10.861  < 2e-16 ***
## TEAM_BATTING_2B  -0.0155618  0.0107659  -1.445 0.148512
## TEAM_BATTING_3B   0.0684310  0.0199331   3.433 0.000611 ***
## TEAM_BATTING_HR   0.0088616  0.0380506   0.233 0.815876
## TEAM_BATTING_BB   0.0071417  0.0066875   1.068 0.285710
## TEAM_BATTING_SO  -0.0050954  0.0029167  -1.747 0.080820 .
## TEAM_BASERUN_SB   0.0272244  0.0049691   5.479 4.93e-08 ***
## TEAM_BASERUN_CS  -0.0167830  0.0178406  -0.941 0.346984
## TEAM_PITCHING_H  -0.0013517  0.0005253  -2.573 0.010163 *
## TEAM_PITCHING_HR  0.0481588  0.0343963   1.400 0.161663
## TEAM_PITCHING_BB  0.0015495  0.0047460   0.326 0.744102
## TEAM_PITCHING_SO  0.0017487  0.0009339   1.872 0.061321 .
## TEAM_FIELDING_E  -0.0187658  0.0028495  -6.586 6.03e-11 ***
## TEAM_FIELDING_DP -0.1158577  0.0149777  -7.735 1.76e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.16 on 1689 degrees of freedom
## Multiple R-squared:  0.3059, Adjusted R-squared:  0.3001
## F-statistic: 53.17 on 14 and 1689 DF,  p-value: < 2.2e-16
```

```
# check for outliers using cooks-distance plot
plot(stp_model_full, which = 4,  id.n = 8)
```




```
# get points of influence
influence <- influence.measures(stp_model_full)
influential_points <- influence$infmat
cooks_d <- influence$infmat[, "cook.d"]
max_influence_index <- which.max(cooks_d)
```

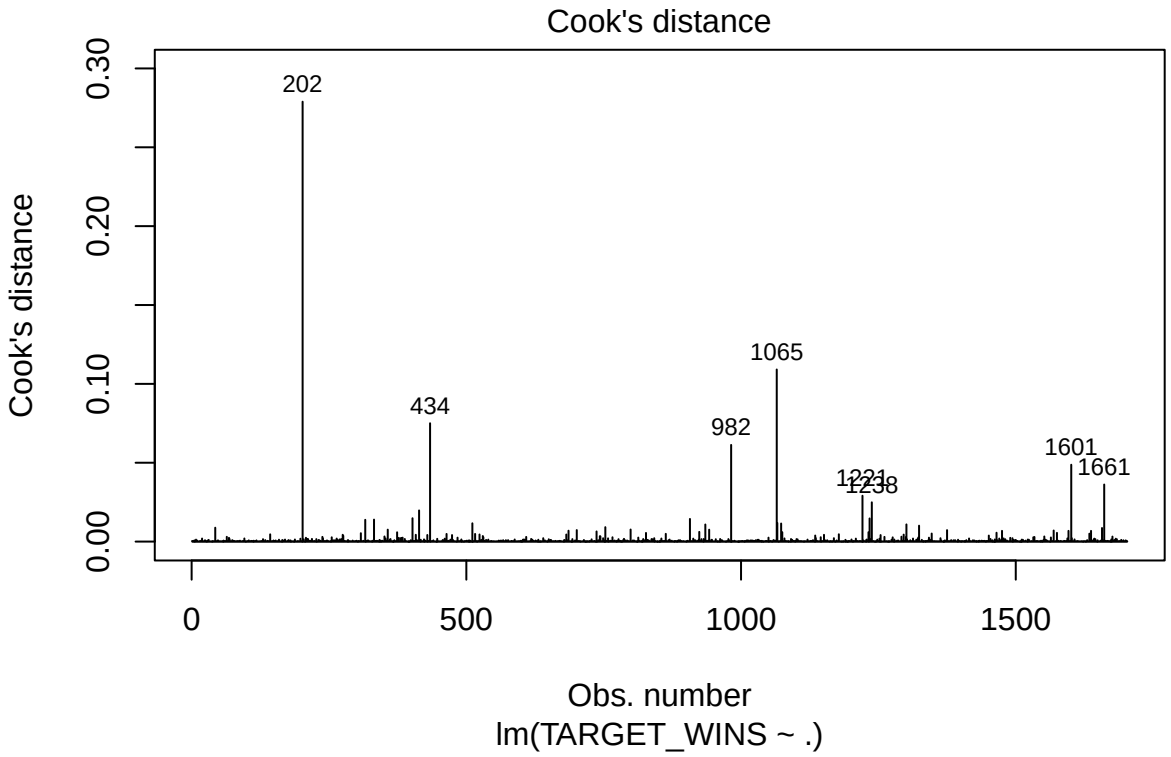
The observation with index 2135 is particularly problematic. A closer examination reveals *TEAM_PITCHING_SO* is almost 75% higher as the next highest value (19278 vs 12758). *TEAM_PITCHING_H* is also unusually high for this year.

```
influential_data_point <- stp75_train_df[max_influence_index, ]
print(influential_data_point)
```

```
##      TARGET_WINS TEAM_BATTING_H TEAM_BATTING_2B TEAM_BATTING_3B TEAM_BATTING_HR
## 2134          41            992             263              20        99.83087
##      TEAM_BATTING_BB TEAM_BATTING_SO TEAM_BASERUN_SB TEAM_BASERUN_CS
## 2134             142             952             101              49
##      TEAM_PITCHING_H TEAM_PITCHING_HR TEAM_PITCHING_BB TEAM_PITCHING_SO
## 2134           20088         407.0946             2876            19278
##      TEAM_FIELDING_E TEAM_FIELDING_DP
## 2134             952              149
```

We will drop this row since it has such a high leverage on the model.

```
# remove outlier
stp75_train_df <- stp75_train_df |>
  filter(TEAM_PITCHING_SO < 15000)

# confirm outliers
stp_model_full <- lm(TARGET_WINS ~ ., data = stp75_train_df)
plot(stp_model_full, which = 4,  id.n = 8)
```

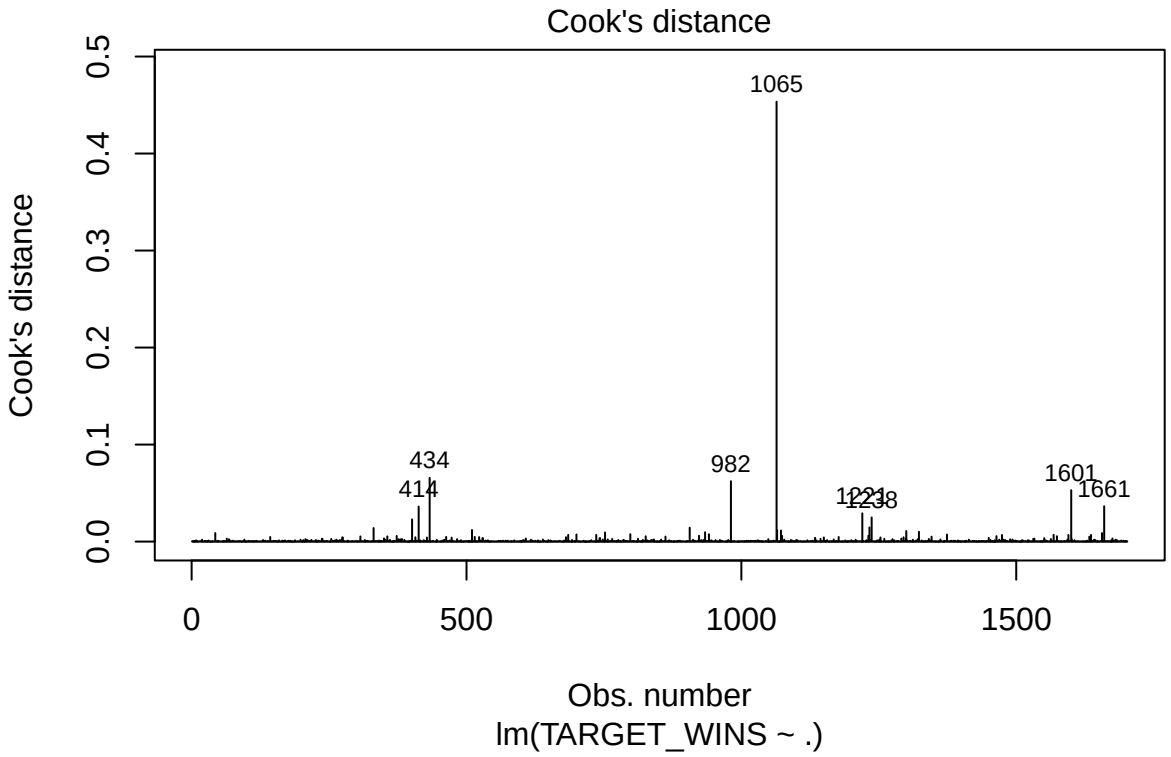


Charting the plot shows another point (202) with high influence. This record has a TEAM_PITCHING_SO that is more than twice the next closest value.

```
# get points of influence
influence <- influence.measures(stp_model_full)
influential_points <- influence$infmat
cooks_d <- influence$infmat[, "cook.d"]
max_influence_index <- which.max(cooks_d)
influential_data_point <- stp75_train_df[max_influence_index, ]
print(influential_data_point)
```

```
##     TARGET_WINS TEAM_BATTING_H TEAM_BATTING_2B TEAM_BATTING_3B TEAM_BATTING_HR
## 202         108           1188             338               0        159.8919
##     TEAM_BATTING_BB TEAM_BATTING_SO TEAM_BASERUN_SB TEAM_BASERUN_CS
## 202             270             945             101              49
##     TEAM_PITCHING_H TEAM_PITCHING_HR TEAM_PITCHING_BB TEAM_PITCHING_SO
## 202           16038         490.5011             3645            12758
##     TEAM_FIELDING_E TEAM_FIELDING_DP
## 202             716              149
```

```
# remove outlier at row 202
stp75_train_df <- stp75_train_df[-c(202), ]

# confirm outliers
stp_model_full <- lm(TARGET_WINS ~ ., data = stp75_train_df)

plot(stp_model_full, which = 4,  id.n = 8)
```



As the Cooks Distance (Di) value is less than 0.5 for all of the remaining outliers appear, they are not significantly influential and can be left in our dataset.

Section 3: Building Models

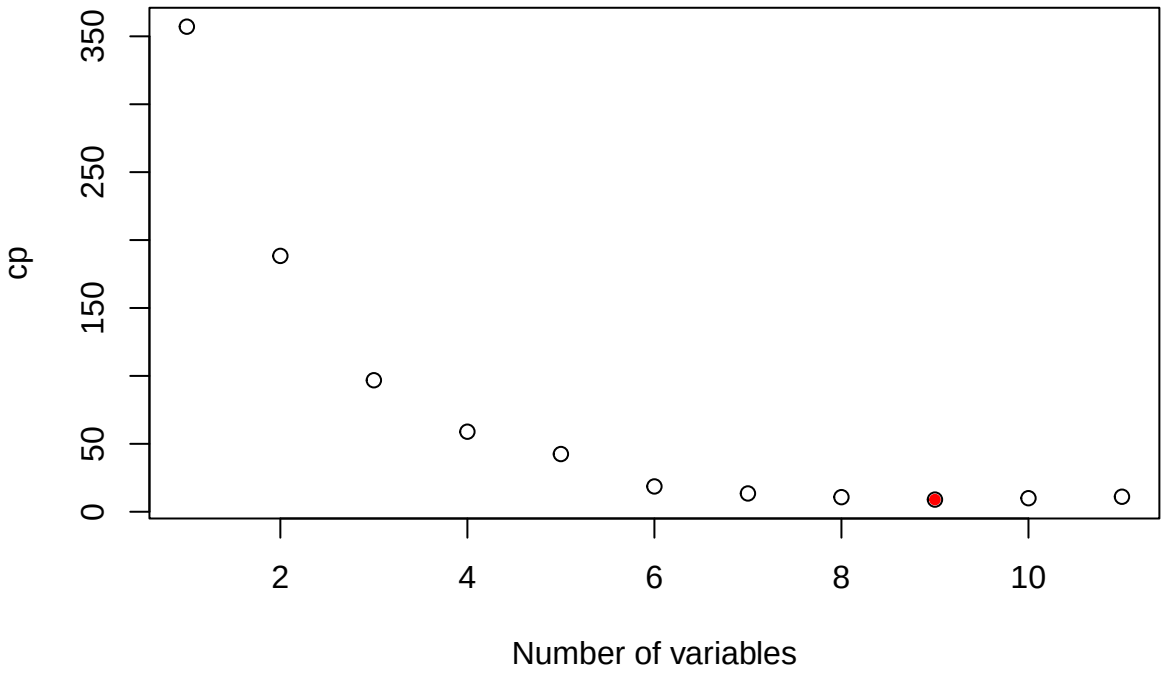
In this section we will build several models which we will compare and evaluate before selecting a final model.

Feature Selection Analysis

In this section, we use Best Subset Selection and Cross-Validation techniques to get a estimate of the optimal number of predictors in our model.

Best Subset Selection Best Subset Selection uses Mallows' Cp to calculate the optimal number of predictors in our model.

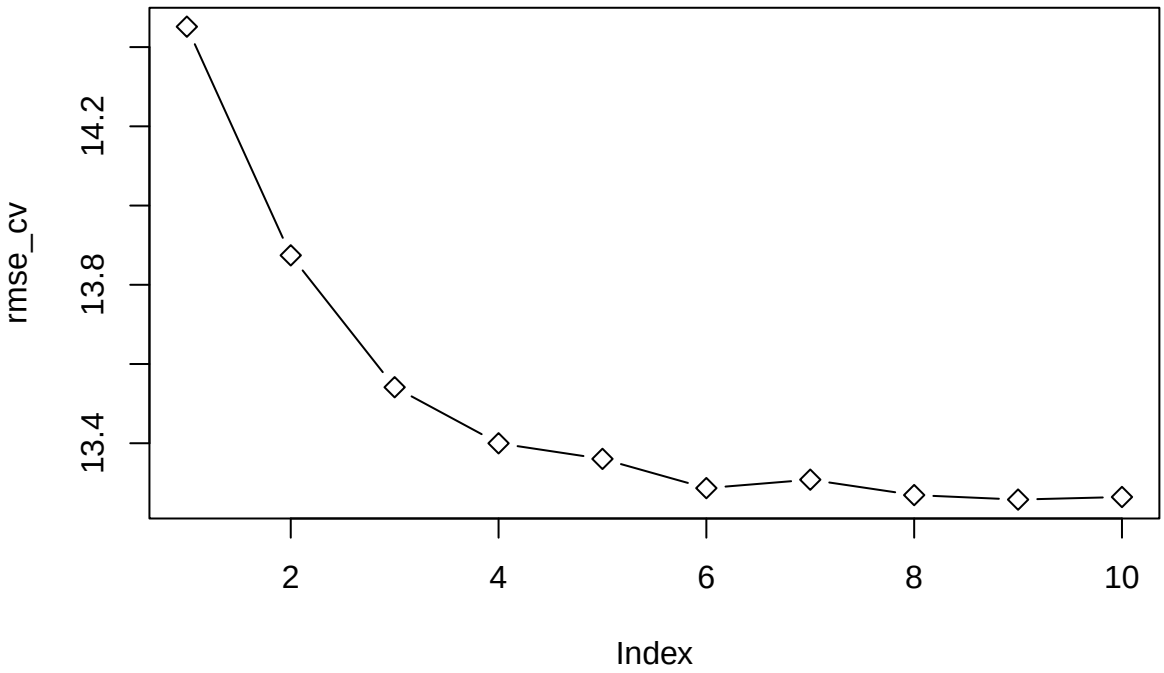
```
regfit_full = regsubsets(TARGET_WINS ~ ., data = stp75_train_df, nvmax = 11)
regfit_summary = summary(regfit_full)
plot(regfit_summary$cp, xlab="Number of variables", ylab="cp")
points(which.min(regfit_summary$cp),regfit_summary$cp[which.min(regfit_summary$cp)], pch=20,col="red")
```



Based on the *Best Subset Selection* method, we estimate that our model should have 9 observations.

Cross Validation As an alternative to *Best Subset Selection*, we used the *Cross Validation* method to estimate the optimal number of predictors in our model. Cross Validation divides our training dataset into k - 1 number of "folds", then tests the data on the kth "fold". For our test, we used five folds (k = 5).

```
set.seed(11)
folds=sample(rep(1:5,length=nrow(stp75_train_df)))

cv_errors = matrix(NA,5,10)
for(k in 1:5) {
  best_fit = regsubsets(TARGET_WINS ~ ., data=stp75_train_df[folds!=k,], nvmax=10, method="forward")
  for(i in 1:10) {
    # Extract the selected coefficients for the i-th model
    selected_coefs = coef(best_fit, id = i)

    # Predict manually by calculating the linear combination of the features
    # First, subset the data for the k-th fold
    test_data = stp75_train_df[folds == k, ]

    # Only include the predictors that were selected
    predictors = names(selected_coefs)[-1]  # Exclude the intercept term

    # Calculate the predictions (including the intercept)
    pred = as.matrix(test_data[, predictors]) %*% selected_coefs[predictors] + selected_coefs[1]

    cv_errors[k,i]=mean((stp75_train_df$TARGET_WINS[folds==k] - pred)^2)
  }
}

rmse_cv = sqrt(apply(cv_errors,2,mean))
plot(rmse_cv, pch=5, type="b")
```

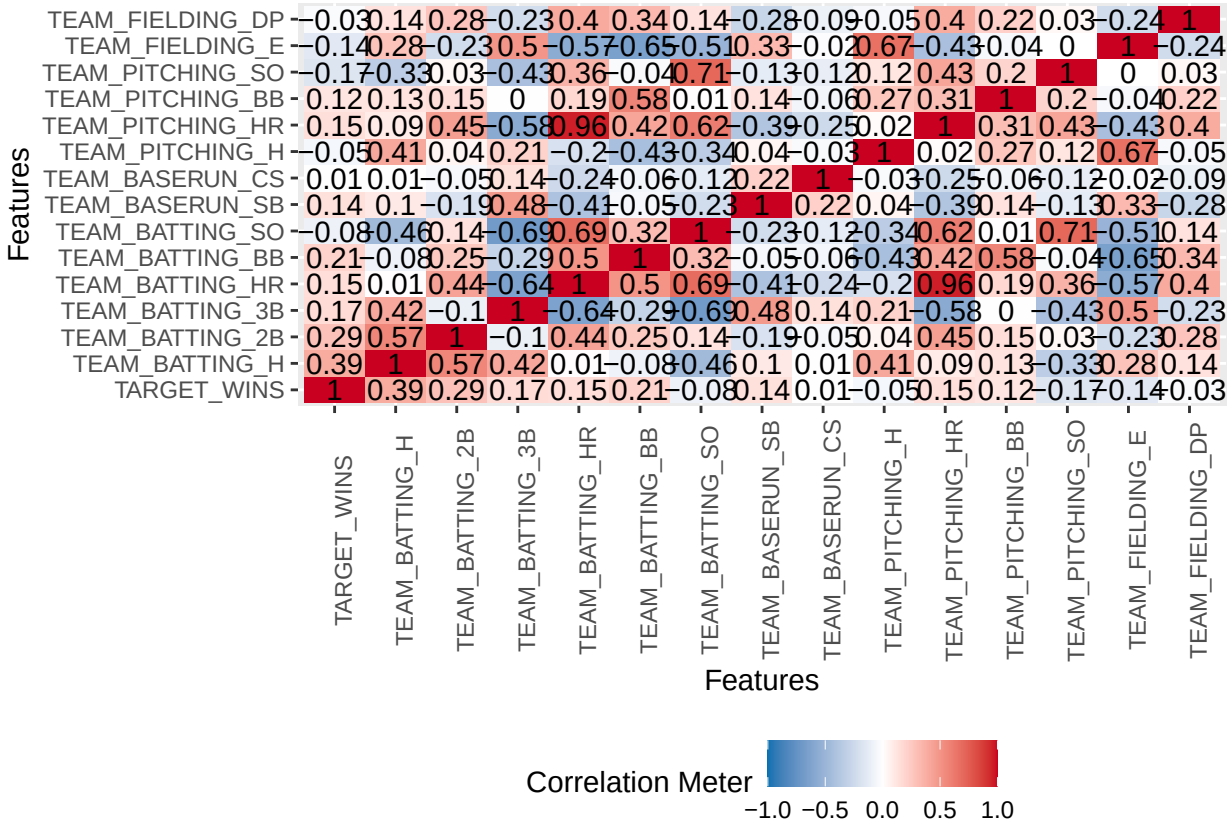


Based on the *Cross Validation* method, we estimate that our model should have 9 observations.

Correlation with Clean Dataset We created another correlation heatmap with our clean training dataset. This time we see strong evidence of multicollinearity between: - TEAM_BATTING_HR and TEAM_PITCHING_HR (0.97) - TEAM_BATTING_HR and TEAM_BATTING_3B (-0.64) - TEAM_BATTING_HR and TEAM_BATTING_SO (0.69) - TEAM_BATTING_3B and TEAM_BATTING_SO (-0.69) - TEAM_PITCHING_H and TEAM_FIELDING_E (0.67) - TEAM_BATTING_BB and TEAM_FIELDING_E (-0.65)

We may need to consider omitting one of the predictors from each pair in our models to ensure that we are not introducing multicollinearity in our model and making our model unnecessarily complex, particulary for the two homerun fields.

```
plot_correlation(stp75_train_df, type = "all")
```



Variable Selection Using Backward Selection

To better understand our variables and their relationships, we first created a model using backward selection to arrive at the smallest number of variables with statistical significance. In this case, we begin by creating a model containing all predictors, then gradually remove the variable with the highest P-value until only the variables with statistical significance remain.

```
summary(stp_model_full)
```

```
##
## Call:
## lm(formula = TARGET_WINS ~ ., data = stp75_train_df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -49.372  -8.651   0.143   8.685  56.455
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)      19.6615875  6.3951726   3.074  0.00214 **
## TEAM_BATTING_H    0.0500740  0.0045988  10.889  < 2e-16 ***
## TEAM_BATTING_2B  -0.0170239  0.0107470  -1.584  0.11337
## TEAM_BATTING_3B   0.0714545  0.0199017   3.590  0.00034 ***
## TEAM_BATTING_HR   0.0118670  0.0383169   0.310  0.75682
## TEAM_BATTING_BB   0.0143694  0.0071848   2.000  0.04566 *
## TEAM_BATTING_SO  -0.0069273  0.0040999  -1.690  0.09128 .
## TEAM_BASERUN_SB   0.0281308  0.0050343   5.588 2.68e-08 ***
## TEAM_BASERUN_CS  -0.0178992  0.0178007  -1.006  0.31479
```

```
## TEAM_PITCHING_H  -0.0010225  0.0005381  -1.900  0.05757 .
## TEAM_PITCHING_HR  0.0448272  0.0345405   1.298  0.19453
## TEAM_PITCHING_BB -0.0046446  0.0052701  -0.881  0.37828
## TEAM_PITCHING_SO  0.0034838  0.0025256   1.379  0.16796
## TEAM_FIELDING_E  -0.0189357  0.0030243  -6.261 4.84e-10 ***
## TEAM_FIELDING_DP -0.1147824  0.0149445  -7.681 2.67e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.13 on 1687 degrees of freedom
## Multiple R-squared:  0.3065, Adjusted R-squared:  0.3008
## F-statistic: 53.27 on 14 and 1687 DF,  p-value: < 2.2e-16
```

```
AIC(stp_model_full)
```

```
## [1] 13611.46
```

TEAM_BATTING_HR has the highest p-value. We removed TEAM_BATTING_HR from our predictors and updated the model.

```
back_select_model <- update(stp_model_full, . ~ . - TEAM_BATTING_HR)
summary(back_select_model)
```

```
##
## Call:
## lm(formula = TARGET_WINS ~ TEAM_BATTING_H + TEAM_BATTING_2B +
##     TEAM_BATTING_3B + TEAM_BATTING_BB + TEAM_BATTING_SO + TEAM_BASERUN_SB +
##     TEAM_BASERUN_CS + TEAM_PITCHING_H + TEAM_PITCHING_HR + TEAM_PITCHING_BB +
##     TEAM_PITCHING_SO + TEAM_FIELDING_E + TEAM_FIELDING_DP, data = stp75_train_df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -49.398  -8.650   0.120   8.703  56.267
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)      19.3930534  6.3344248   3.062 0.002237 **
## TEAM_BATTING_H    0.0503535  0.0045082  11.169  < 2e-16 ***
## TEAM_BATTING_2B  -0.0172263  0.0107242  -1.606 0.108396
## TEAM_BATTING_3B   0.0702542  0.0195154   3.600 0.000327 ***
## TEAM_BATTING_BB   0.0151153  0.0067673   2.234 0.025641 *
## TEAM_BATTING_SO  -0.0066056  0.0039651  -1.666 0.095908 .
## TEAM_BASERUN_SB   0.0280062  0.0050169   5.582 2.76e-08 ***
## TEAM_BASERUN_CS  -0.0184305  0.0177131  -1.040 0.298257
## TEAM_PITCHING_H  -0.0010874  0.0004954  -2.195 0.028309 *
## TEAM_PITCHING_HR  0.0550220  0.0104606   5.260 1.63e-07 ***
## TEAM_PITCHING_BB -0.0052399  0.0049057  -1.068 0.285619
## TEAM_PITCHING_SO  0.0033743  0.0025001   1.350 0.177303
## TEAM_FIELDING_E  -0.0188561  0.0030126  -6.259 4.90e-10 ***
## TEAM_FIELDING_DP -0.1146672  0.0149358  -7.677 2.73e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.12 on 1688 degrees of freedom
## Multiple R-squared:  0.3065, Adjusted R-squared:  0.3012
## F-statistic: 57.39 on 13 and 1688 DF,  p-value: < 2.2e-16
```

```
AIC(back_select_model)
```

```
## [1] 13609.56
```

TEAM_PITCHING_SO has the highest p-value. We removed TEAM_PITCHING_SO from our predictors and update the model.

```
back_select_model <- update(back_select_model, . ~ . - TEAM_PITCHING_SO)
```

TEAM_BATTING_SO has the highest p-value. We removed TEAM_BATTING_SO from our predictors and update the model.

```
back_select_model <- update(back_select_model, . ~ . - TEAM_BATTING_SO)
```

TEAM_BASERUN_CS has the highest p-value. We removed TEAM_BASERUN_CS from our predictors and update the model.

```
back_select_model <- update(back_select_model, . ~ . - TEAM_BASERUN_CS)
```

TEAM_PITCHING_BB has the highest p-value. We removed TEAM_PITCHING_BB from our predictors and update the model.

```
back_select_model <- update(back_select_model, . ~ . - TEAM_PITCHING_BB)
summary(back_select_model)
```

```
##
## Call:
## lm(formula = TARGET_WINS ~ TEAM_BATTING_H + TEAM_BATTING_2B +
##     TEAM_BATTING_3B + TEAM_BATTING_BB + TEAM_BASERUN_SB + TEAM_PITCHING_H +
##     TEAM_PITCHING_HR + TEAM_FIELDING_E + TEAM_FIELDING_DP, data = stp75_train_df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -48.704  -8.583   0.042   8.784  55.405
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)      13.9929607  4.1169190   3.399 0.000692 ***
## TEAM_BATTING_H    0.0523599  0.0037458  13.978  < 2e-16 ***
## TEAM_BATTING_2B  -0.0198370  0.0101932  -1.946 0.051808 .
## TEAM_BATTING_3B   0.0752418  0.0186636   4.031 5.79e-05 ***
## TEAM_BATTING_BB   0.0101791  0.0038645   2.634 0.008516 **
## TEAM_BASERUN_SB   0.0243069  0.0045846   5.302 1.30e-07 ***
## TEAM_PITCHING_H  -0.0012278  0.0003957  -3.103 0.001948 **
## TEAM_PITCHING_HR  0.0501199  0.0081982   6.114 1.21e-09 ***
## TEAM_FIELDING_E  -0.0171801  0.0027913  -6.155 9.36e-10 ***
## TEAM_FIELDING_DP -0.1138395  0.0148655  -7.658 3.16e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.12 on 1692 degrees of freedom
## Multiple R-squared:  0.3049, Adjusted R-squared:  0.3012
## F-statistic: 82.47 on 9 and 1692 DF,  p-value: < 2.2e-16
```

```
AIC(back_select_model)
```

```
## [1] 13605.45
```

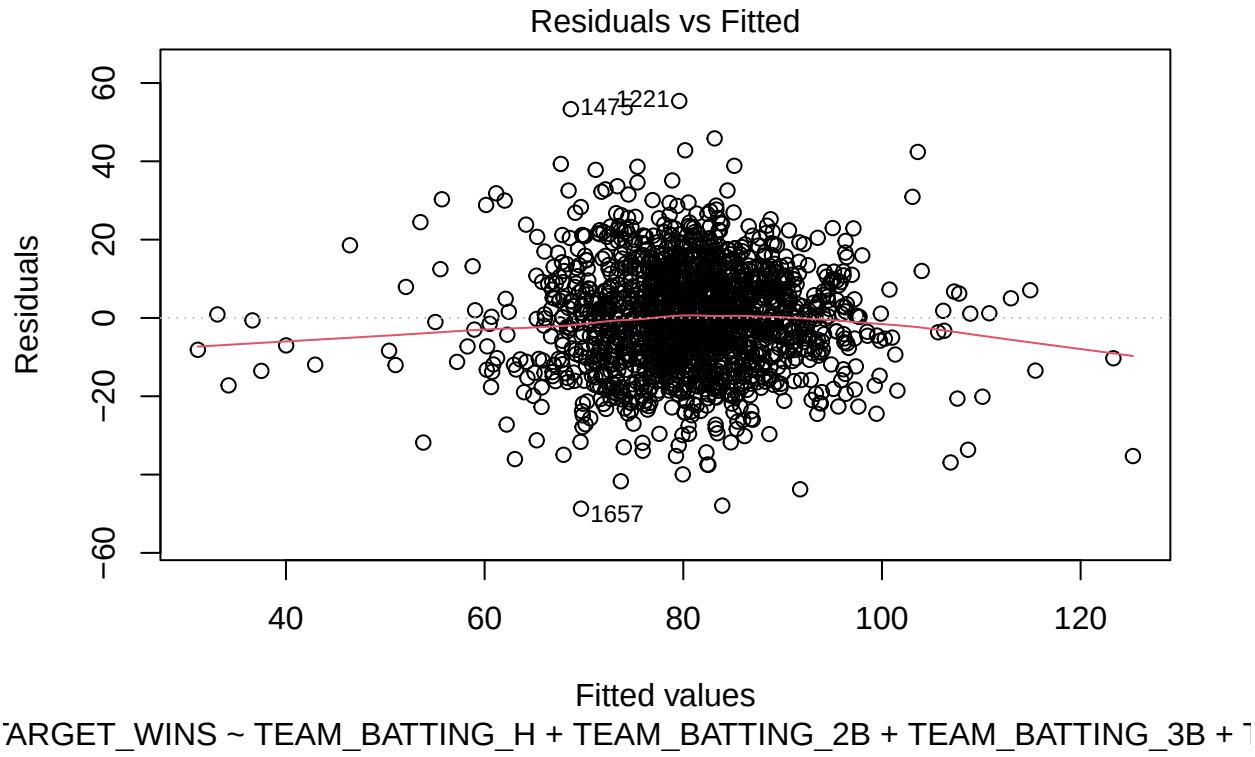
Backward selection using o-values arrives at a model with six variables with high statistical significance ((p-value < 0.001) and three with moderate (p-value $\sim= 0.1$) statistical significance. Arriving at 9 predictors is consistent with our Best Subset Selection test. A VIF test shows that there are no evidence of strong multicollinearity.

```
vif(back_select_model)
```

```
##   TEAM_BATTING_H  TEAM_BATTING_2B  TEAM_BATTING_3B  TEAM_BATTING_BB
##         2.897373         2.289513         2.613693         2.171553
##  TEAM_BASERUN_SB TEAM_PITCHING_H TEAM_PITCHING_HR  TEAM_FIELDING_E
##         1.551374         2.514131         2.541082         3.790194
## TEAM_FIELDING_DP
##         1.329128
```

Linearity

```
plot(back_select_model, which=1)
```

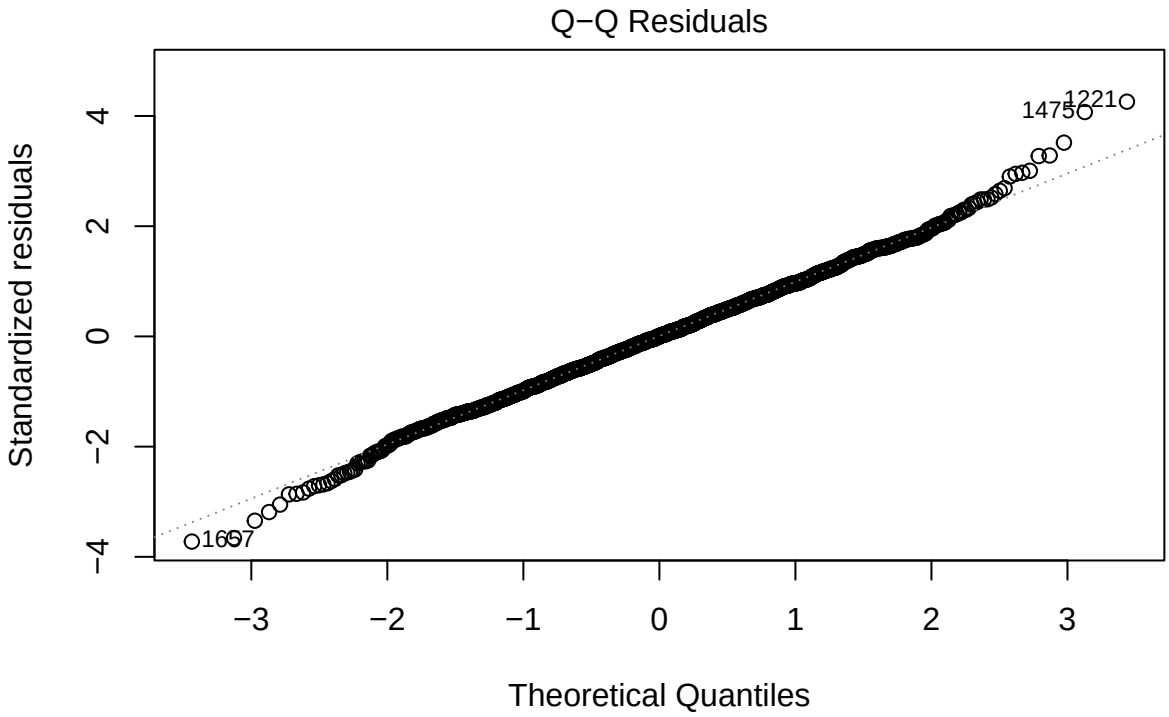


TARGET_WINS ~ TEAM_BATTING_H + TEAM_BATTING_2B + TEAM_BATTING_3B + T

Our diagnostic plots show a fairly linear model.

Normality

```
plot(back_select_model, which=2)
```



TARGET_WINS ~ TEAM_BATTING_H + TEAM_BATTING_2B + TEAM_BATTING_3B + T

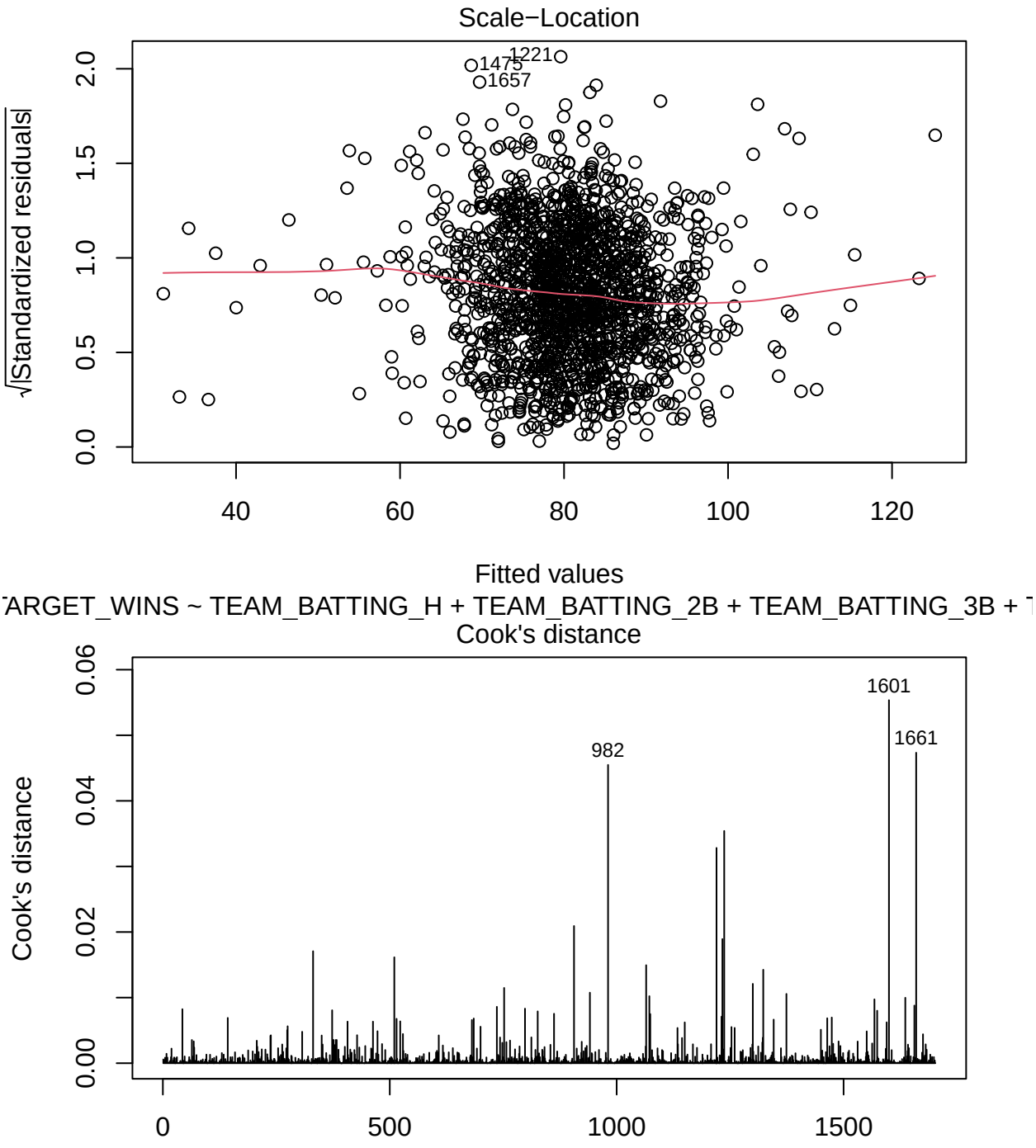
```
shapiro.test(residuals(back_select_model))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(back_select_model)
## W = 0.99724, p-value = 0.004263
```

```
# Shapiro-Wilk normality test: look for high p-value
```

Our QQ plot suggests normality thought there is obvious skewing on the tails, particularly on the right. This is confirmed by a Shapiro Wilk's Test statistic of 0.9967 which suggesting normality.

Heteroscedasticity:



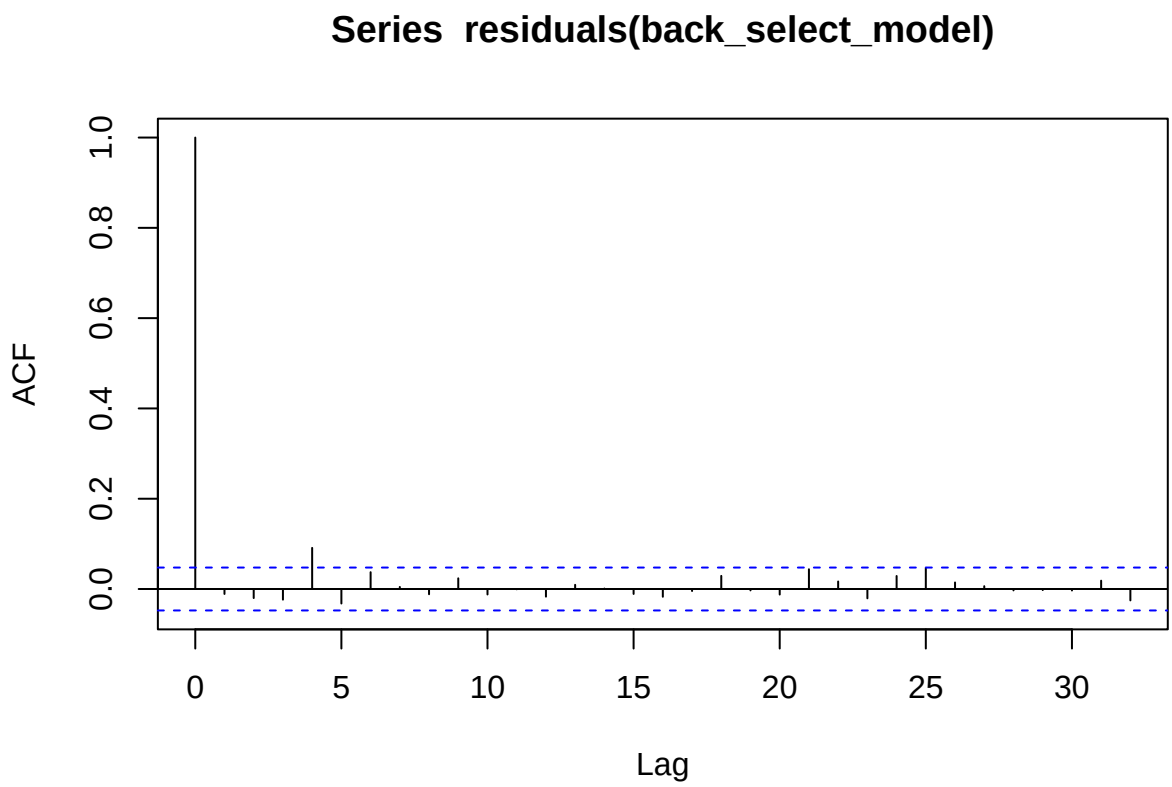

```
##
##  studentized Breusch-Pagan test
##
## data:  back_select_model
## BP = 177.7, df = 9, p-value < 2.2e-16
```

Our Scale-Location plot shows that points appear somewhat evenly distributed above and below the trend

line. While there is no obvious fan/wedge pattern, there is clustering in the center suggesting underfitting, high leverage outliers or that additional transformation may be needed. As our Cook's Distance plot has no values with a greater than 1, we can rule our the effects of high leverage points.

The Breusch-Page test statistic BP (238.69) and the small p-value (2.2e-16) suggest evidence of heteroscedasticity. Though the values are different that we may need to transform the data to meet the Assumption of Homoscedasticity.

```
acf(residuals(back_select_model))
```



Independence:

```
durbinWatsonTest(back_select_model)
```

```
##  lag Autocorrelation D-W Statistic p-value
##    1     -0.01100517       2.02107   0.656
##  Alternative hypothesis: rho != 0
```

```
# Durbin Watson should be close to 2
```

Our Autocorrelation Function shows that there are lags above the blue dashed line, suggesting no autocorrelation. This is confirmed through a Durbin-Watson test statistic value of 2.06 and an autocorrelation value of -0.0342. Furthermore, as our p-value (0.222) is greater than 0.05, we do not have enough evidence to reject the null hypothesis that there is no autocorrelation. In other words, the test results suggest that our model's residuals are independent and therefore do not violate the Independence Assumption.

Hand-Selected Models

The following models were created by hand-selecting predictors based on their perceived importance to a team's performance.

Model H1: Base Baseball Stats Model This model includes fundamental offensive, defensive, and pitching stats that logically contribute to wins and includes the following variables for the respective reasons

- TEAM_BATTING_H (Hits): More hits increase the chances of scoring.
- TEAM_BATTING_HR (Home Runs): Home runs are a major contributor to runs.
- TEAM_PITCHING_SO (Strikeouts): More strikeouts reduce opponent scoring.
- TEAM_FIELDING_E (Errors): More errors lead to more opponent runs (negative predictor).

Why we excluded some variables? - TEAM_BASERUN_SB (Stolen Bases): Limited impact on overall wins. - TEAM_PITCHING_BB (Walks Allowed): May not be as predictive when combined with strikeouts.

```
base_model <- lm(TARGET_WINS ~ TEAM_BATTING_H + TEAM_BATTING_HR + TEAM_PITCHING_SO + TEAM_FIELDING_E, da

# View model summary
summary(base_model)
```

Model Summary Statistics

```
##
## Call:
## lm(formula = TARGET_WINS ~ TEAM_BATTING_H + TEAM_BATTING_HR +
##     TEAM_PITCHING_SO + TEAM_FIELDING_E, data = stp75_train_df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -53.589  -9.354  -0.044   9.474  49.980
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)      11.8558072  4.3576985   2.721  0.00658 **
## TEAM_BATTING_H    0.0505808  0.0028672  17.641  < 2e-16 ***
## TEAM_BATTING_HR   0.0004848  0.0083025   0.058  0.95344
## TEAM_PITCHING_SO -0.0009588  0.0014849  -0.646  0.51856
## TEAM_FIELDING_E  -0.0193389  0.0021956  -8.808  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.83 on 1697 degrees of freedom
## Multiple R-squared:  0.2257, Adjusted R-squared:  0.2239
## F-statistic: 123.7 on 4 and 1697 DF,  p-value: < 2.2e-16
```

Interpreting the Coefficients of the Regression Model

1. Intercept (15.15) When all predictor variables (TEAM_BATTING_H, TEAM_BATTING_HR, TEAM_PITCHING_SO, TEAM_FIELDING_E) are zero, a team is expected to have 15.15 wins.
2. TEAM_BATTING_H (Hits) For every additional hit, the team is expected to win 0.048 more games. More hits lead to more wins, which is expected in baseball.
3. TEAM_BATTING_HR (Home Runs) More home runs slightly increases wins, but the effect is very small. We should consider removing this variable as the t-value and p-value suggest that it is not statistically significant.
4. TEAM_PITCHING_SO (Strikeouts by Pitchers) More strikeouts slightly decreases wins, but the effect is very small and not statistically significant. We should consider adding walks (TEAM_PITCHING_BB) to capture pitching effectiveness better.

5. TEAM_FIELDING_E For every additional error, a team is expected to lose 0.02 games. More errors directly hurt a team's chances of winning, which makes sense in baseball.

The F-statistic of 127.1 ($p < 2.2e\text{-}16$) means our model is highly statistically significant. This suggests that at least one of our variables—such as home runs, hits, strikeouts, or errors—has a real impact on predicting wins.

However, we still need to check which specific variables are the most meaningful (p-values of individual coefficients) and whether we can improve the model further.

Diagnosing our Model Checking for Multicollinearity (VIF Test)

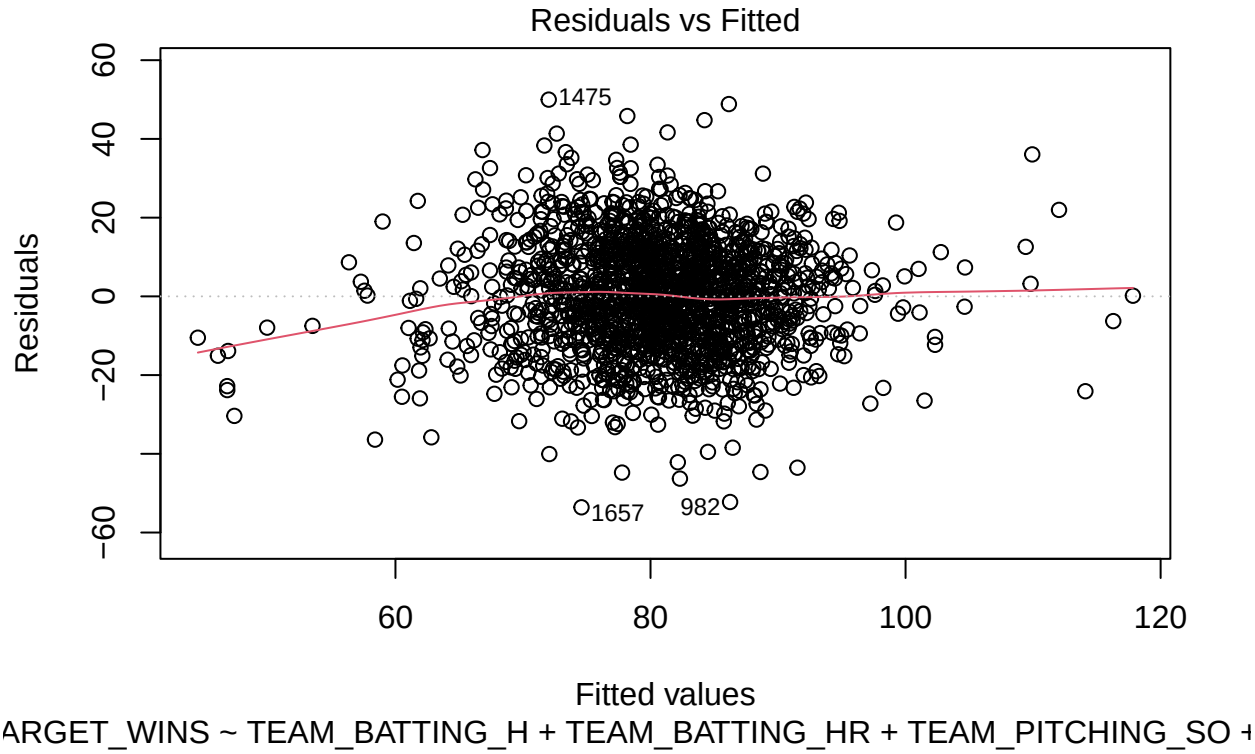
Multicollinearity occurs when predictor variables are highly correlated, leading to unstable coefficients and inflated standard errors. Using the Variance Inflation Factor (VIF) test we see no evidence of strong multicollinearity in the model.

```
vif(base_model)
```

```
##   TEAM_BATTING_H  TEAM_BATTING_HR TEAM_PITCHING_SO  TEAM_FIELDING_E
##         1.528358         2.264255         1.663513         2.111267
```

Linearity

```
plot(base_model, which=1)
```

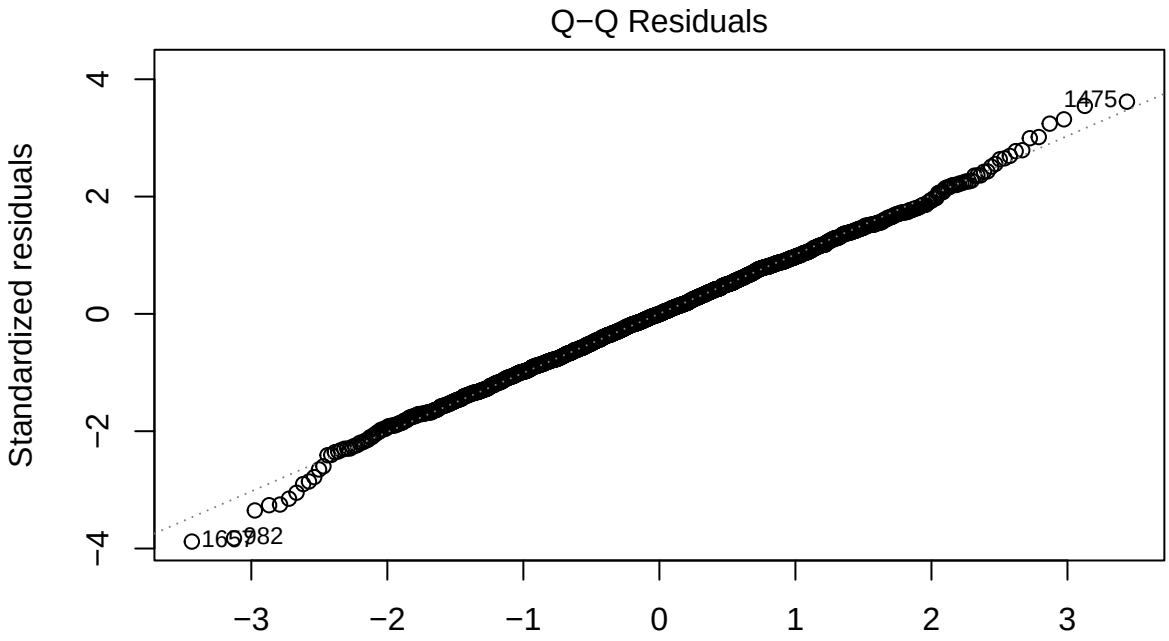


ARGET_WINS ~ TEAM_BATTING_H + TEAM_BATTING_HR + TEAM_PITCHING_SO +

Our diagnostic plots show a fairly linear model.

Normality Check:

```
plot(base_model, which=2)
```



ARGET_WINS ~ TEAM_BATTING_H + TEAM_BATTING_HR + TEAM_PITCHING_SO +

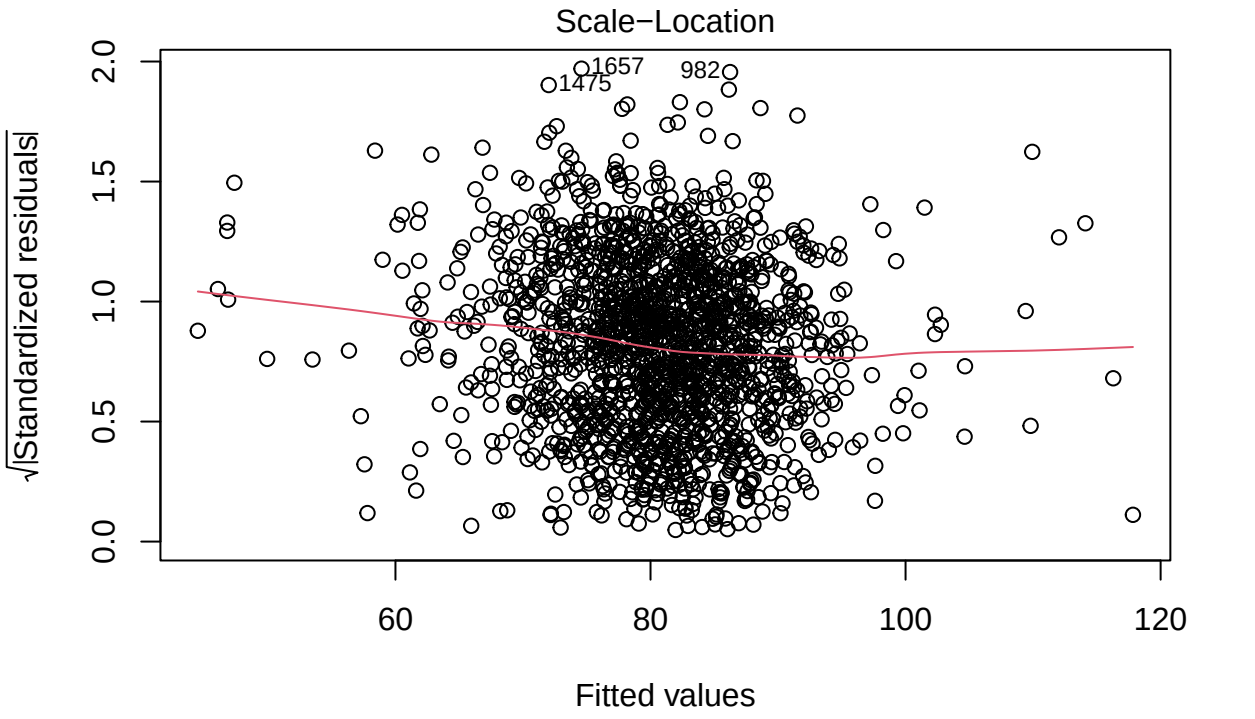
```
shapiro.test(residuals(base_model))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(base_model)
## W = 0.99826, p-value = 0.07177
```

```
# Shapiro-Wilk normality test: look for high p-value
```

Our QQ plot suggests normality thought there is obvious skewing on the tails, particularly on the right. This is confirmed by A Shapiro Wilk's Test statistic of W = 0.9980 suggesting normality.

Heterocedasticity:



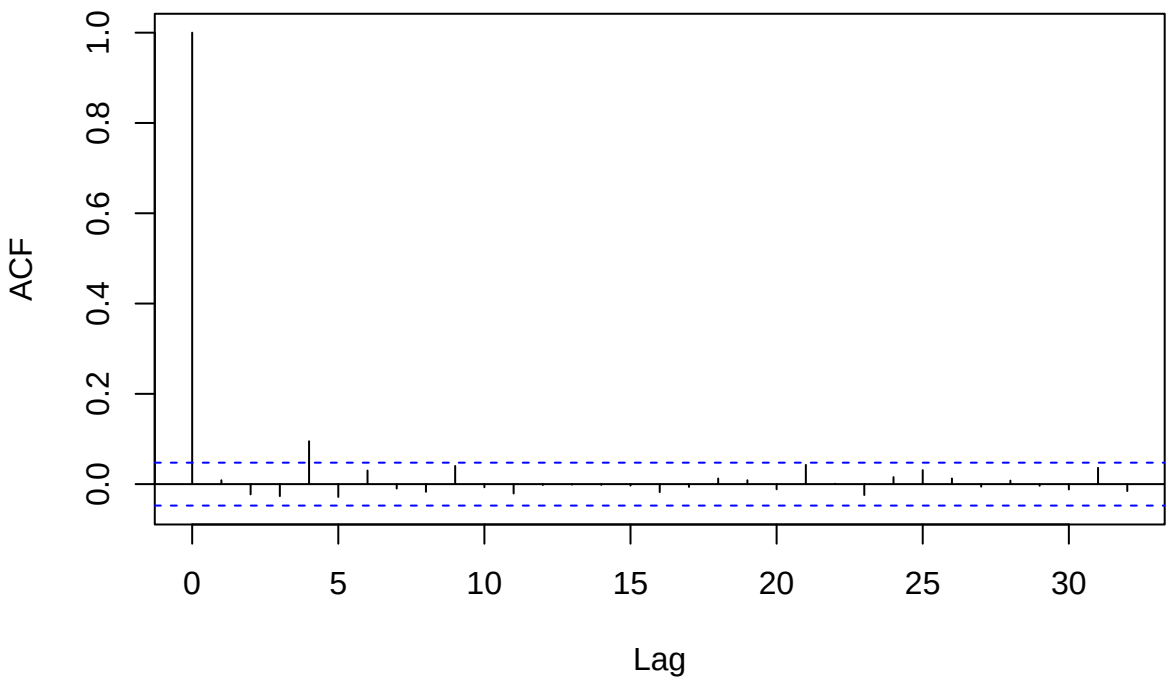
ARGET_WINS ~ TEAM_BATTING_H + TEAM_BATTING_HR + TEAM_PITCHING_SO +

```
##
##  studentized Breusch-Pagan test
##
## data:  base_model
## BP = 159.2, df = 4, p-value < 2.2e-16
```

Test Statistic (BP = 188.29) suggests evidence of heteroscedasticity. Additionally, the p-value is extremely small (much less than 0.05), suggesting higher evidence of heteroscedasticity and that we may need to transform the data to meet the Assumption of Homoscedasticity.

Independence:

```
acf(residuals(base_model))
```




```
durbinWatsonTest(base_model)
```

```
##  lag Autocorrelation D-W Statistic p-value
##    1      0.00860228      1.982265   0.722
##  Alternative hypothesis: rho != 0
```

```
# Durbin Watson should be close to 2
```

Our Autocorrelation Function shows that there are lags above the blue dashed line, suggesting no autocorrelation. This is confirmed through a Durbin-Watson which suggest that our model's residuals are independent and therefore do not violate the Independence Assumption.

Model H2: Test Interaction Term (TEAM_BATTING_HR * TEAM_BATTING_BB) If a team hits more home runs and draws more walks, they likely to score more runs. We test if walks amplify the impact of home runs on wins.

```
interaction_model <- lm(TARGET_WINS ~ TEAM_BATTING_H + TEAM_BATTING_HR + TEAM_BATTING_BB +
                          TEAM_BATTING_HR:TEAM_BATTING_BB +
                          TEAM_PITCHING_SO + TEAM_FIELDING_E,
                        data = stp75_train_df)

summary(interaction_model)
```

```
##
## Call:
## lm(formula = TARGET_WINS ~ TEAM_BATTING_H + TEAM_BATTING_HR +
##     TEAM_BATTING_BB + TEAM_BATTING_HR:TEAM_BATTING_BB + TEAM_PITCHING_SO +
##     TEAM_FIELDING_E, data = stp75_train_df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
```

```
## -55.305  -8.895   0.054   9.465  51.049
##
## Coefficients:
##                                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)                   19.4170412  5.3678460   3.617 0.000306 ***
## TEAM_BATTING_H                 0.0501524  0.0028237  17.761  < 2e-16 ***
## TEAM_BATTING_HR               -0.1929137  0.0310142  -6.220 6.24e-10 ***
## TEAM_BATTING_BB               -0.0110335  0.0056680  -1.947 0.051745 .
## TEAM_PITCHING_SO              -0.0007137  0.0014771  -0.483 0.629024
## TEAM_FIELDING_E               -0.0212658  0.0025984  -8.184 5.33e-16 ***
## TEAM_BATTING_HR:TEAM_BATTING_BB  0.0003377  0.0000544   6.207 6.76e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.62 on 1695 degrees of freedom
## Multiple R-squared:  0.2504, Adjusted R-squared:  0.2478
## F-statistic: 94.38 on 6 and 1695 DF,  p-value: < 2.2e-16
```

This model includes an interaction term (TEAM_BATTING_HR:TEAM_BATTING_BB) to see if walks (BB) affect the impact of home runs (HR) on wins.

Like model H1, this model is statistically significant. Our Adjusted R^2 is higher than that of model H1 suggesting that the additional predictors has added value to the model.

More Home Runs (HR) Alone → Fewer Wins (Unexpected): The negative coefficient (-0.1650) on TEAM_BATTING_HR suggests that hitting more home runs alone does not necessarily lead to more wins.

Walks (BB) Alone Have a Weak Impact on Wins: The coefficient for TEAM_BATTING_BB is negative (-0.0074) and not statistically significant ($p = 0.1387$). This means that walks alone do not have a strong impact on wins.

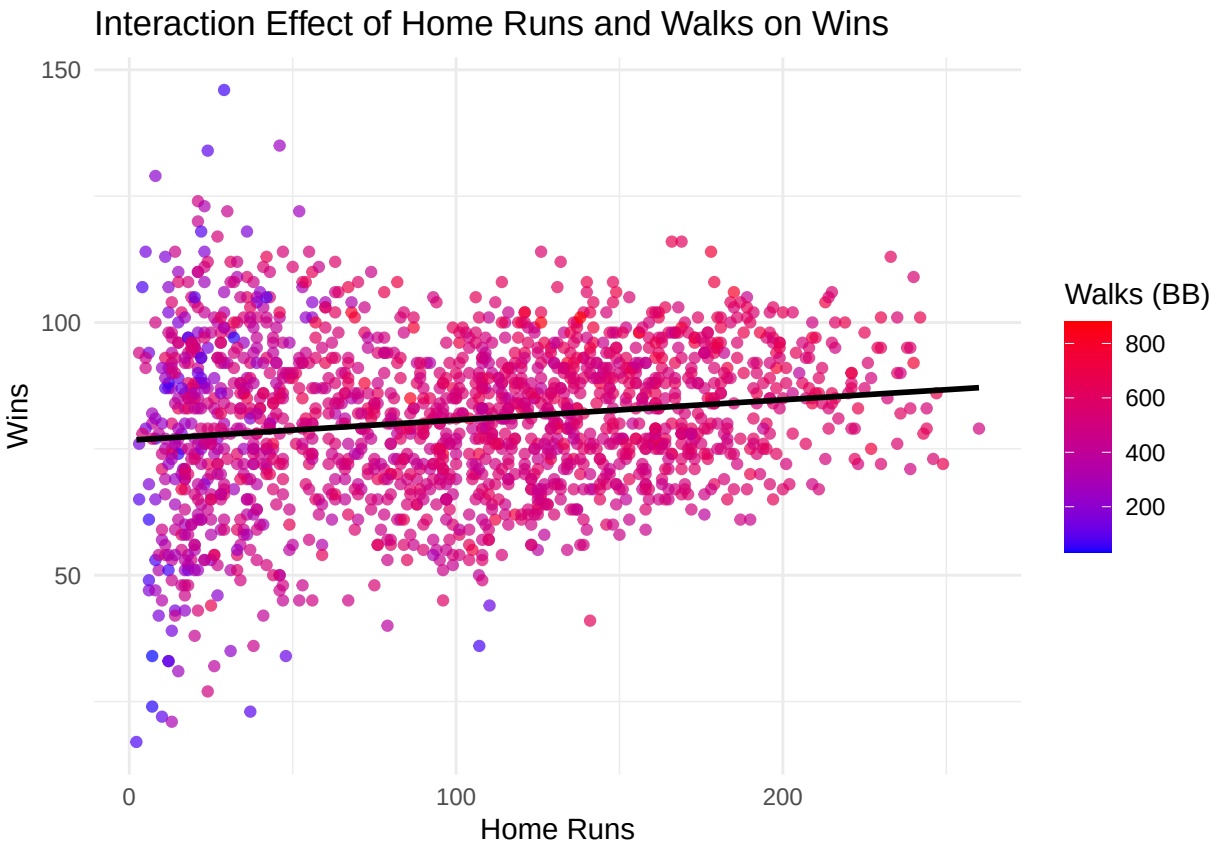
The Interaction Term (TEAM_BATTING_HR * TEAM_BATTING_BB) is Highly Significant ($p = 3.39e\text{-}10$) Positive Coefficient (+0.000301)

Teams that hit home runs AND get on base with walks tend to win more games. This confirms that home runs are more valuable when combined with walks.

Visualizing the Interaction Effect: We want to see how home runs (TEAM_BATTING_HR) and walks (TEAM_BATTING_BB) impact wins (TARGET_WINS) together.

```
library(ggplot2)

ggplot(stp75_train_df, aes(x = TEAM_BATTING_HR, y = TARGET_WINS, color = TEAM_BATTING_BB)) +
  geom_point(alpha = 0.7) +
  geom_smooth(method = "lm", se = FALSE, color = "black") +
  scale_color_gradient(low = "blue", high = "red") +
  labs(title = "Interaction Effect of Home Runs and Walks on Wins",
       x = "Home Runs",
       y = "Wins",
       color = "Walks (BB)") +
  theme_minimal()
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



Red (high walks) teams should have higher wins for the same HRs. Blue (low walks) teams may not benefit as much from HRs. The trendline is steeper for teams with more walks, confirming that walks amplify HR impact.

Model H3 Adding TEAM_BATTING_2B (Doubles) to the Model: Doubles (2B) are a strong indicator of offensive power and often correlate with scoring more runs. If a team doesn't hit home runs, but hits many doubles, it can still score efficiently.

```
improved_model <- lm(TARGET_WINS ~ TEAM_BATTING_H + TEAM_BATTING_HR + TEAM_BATTING_BB + TEAM_BATTING_2B
                     data = stp75_train_df)

summary(improved_model)
```

```
##
## Call:
## lm(formula = TARGET_WINS ~ TEAM_BATTING_H + TEAM_BATTING_HR +
##     TEAM_BATTING_BB + TEAM_BATTING_2B + TEAM_BATTING_HR:TEAM_BATTING_BB +
##     TEAM_PITCHING_SO + TEAM_FIELDING_E, data = stp75_train_df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -56.407  -8.812   0.088   9.089  53.901
##
## Coefficients:
##                                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)                    1.465e+01  5.582e+00   2.625  0.00874 **
## TEAM_BATTING_H                 5.811e-02  3.856e-03  15.071  < 2e-16 ***
## TEAM_BATTING_HR               -1.883e-01  3.098e-02  -6.080 1.48e-09 ***
```

```
## TEAM_BATTING_BB                -1.067e-02  5.656e-03  -1.886  0.05941 .
## TEAM_BATTING_2B                -3.263e-02  1.079e-02  -3.024  0.00253 **
## TEAM_PITCHING_SO                5.050e-04  1.528e-03   0.331  0.74100
## TEAM_FIELDING_E                -2.366e-02  2.710e-03  -8.729  < 2e-16 ***
## TEAM_BATTING_HR:TEAM_BATTING_BB  3.359e-04  5.427e-05   6.190 7.55e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.58 on 1694 degrees of freedom
## Multiple R-squared:  0.2544, Adjusted R-squared:  0.2514
## F-statistic: 82.59 on 7 and 1694 DF,  p-value: < 2.2e-16
```

Key Findings Adding TEAM_BATTING_2B slightly improves model performance

R^2 increased from 27.2% → 27.9% (small improvement). Residual Standard Error decreased from 13.31 → 13.25 (better fit). Unexpected negative coefficient for doubles (-0.0429, p = 3.09e-06)

Suggests that more doubles lead to fewer wins, which is counterintuitive. Possible reasons: Multicollinearity with TEAM_BATTING_H (hits). Bad teams might hit many doubles but still lose. Interaction term (TEAM_BATTING_HR * TEAM_BATTING_BB) remains strong and positive

Diagnosing our Model:

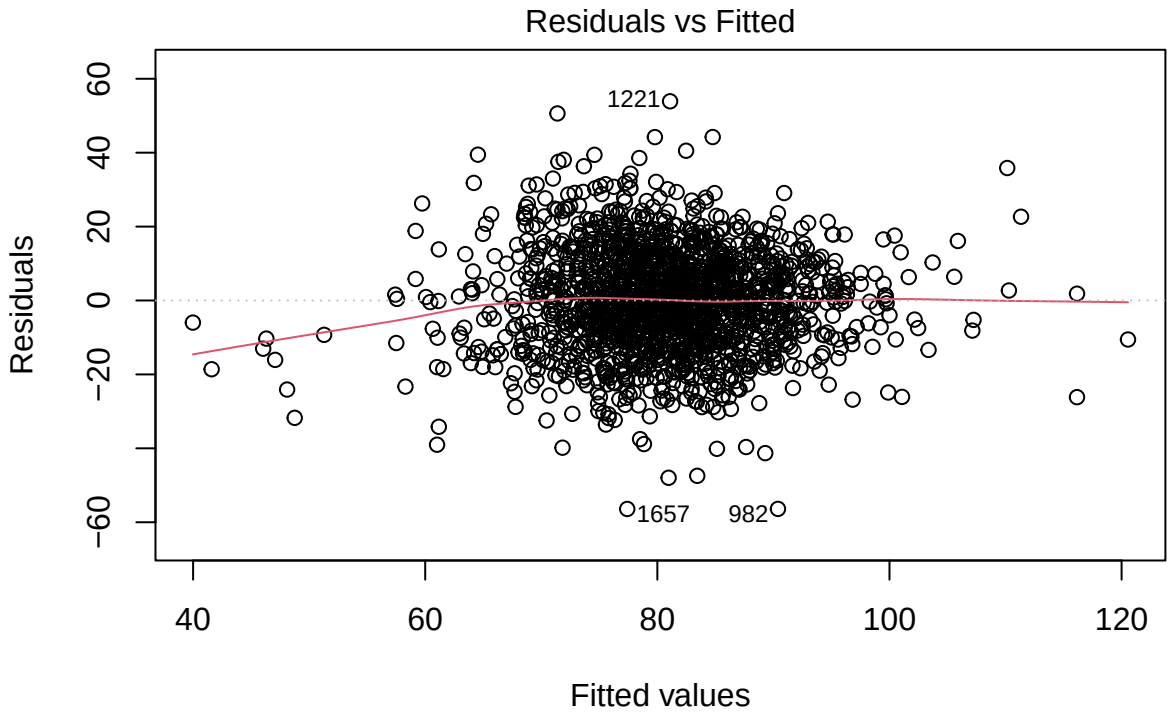
Multicolinearity: Our VIF test shows no strong evidence of multicollienarity between our predictors when adjusting for our interactions.

```
# Check Variance Inflation Factor (VIF)
vif(improved_model, type='predictor')
```

```
## GVIFs computed for predictors
```

```
##                      GVIF Df GVIF^(1/(2*Df))  Interacts With
## TEAM_BATTING_H   2.865889  1        1.692894             --
## TEAM_BATTING_HR  4.026183  3        1.261292 TEAM_BATTING_BB
## TEAM_BATTING_BB  4.026183  3        1.261292 TEAM_BATTING_HR
## TEAM_BATTING_2B  2.395133  1        1.547622             --
## TEAM_PITCHING_SO 1.825484  1        1.351105             --
## TEAM_FIELDING_E  3.335994  1        1.826470             --
##                                                                                  Other Predictors
## TEAM_BATTING_H   TEAM_BATTING_HR, TEAM_BATTING_BB, TEAM_BATTING_2B, TEAM_PITCHING_SO, TEAM_FIELDING_E
## TEAM_BATTING_HR                   TEAM_BATTING_H, TEAM_BATTING_2B, TEAM_PITCHING_SO, TEAM_FIELDING_E
## TEAM_BATTING_BB                   TEAM_BATTING_H, TEAM_BATTING_2B, TEAM_PITCHING_SO, TEAM_FIELDING_E
## TEAM_BATTING_2B   TEAM_BATTING_H, TEAM_BATTING_HR, TEAM_BATTING_BB, TEAM_PITCHING_SO, TEAM_FIELDING_E
## TEAM_PITCHING_SO   TEAM_BATTING_H, TEAM_BATTING_HR, TEAM_BATTING_BB, TEAM_BATTING_2B, TEAM_FIELDING_E
## TEAM_FIELDING_E   TEAM_BATTING_H, TEAM_BATTING_HR, TEAM_BATTING_BB, TEAM_BATTING_2B, TEAM_PITCHING_SO
```

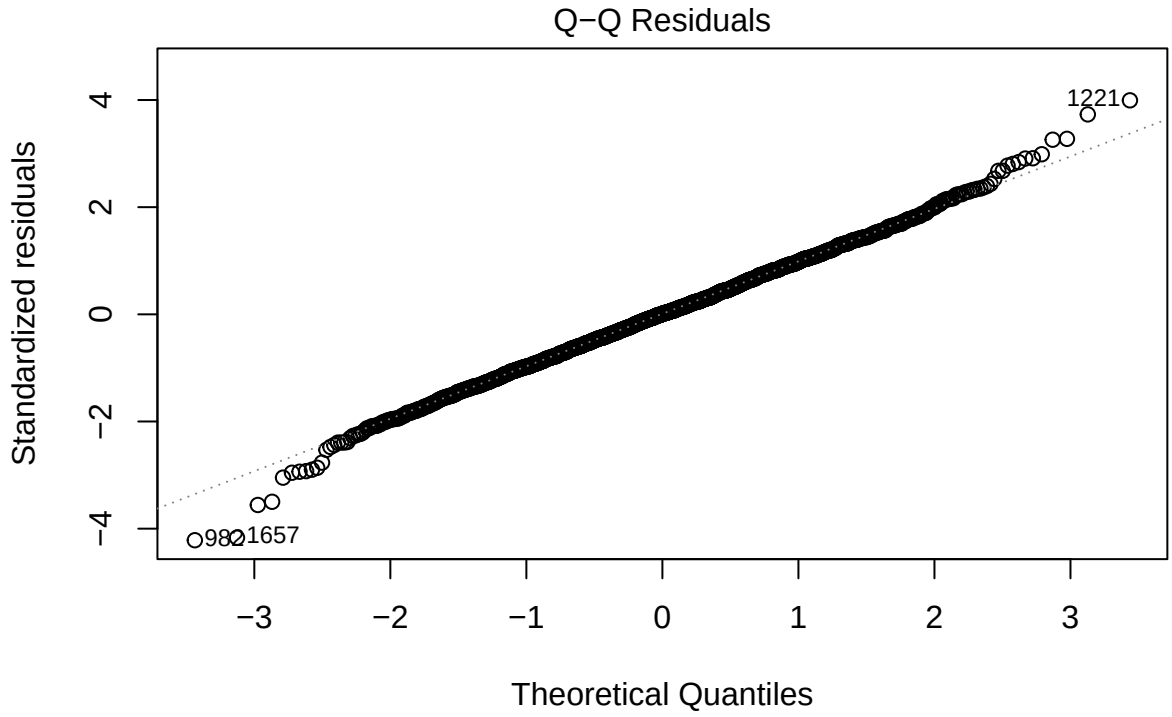
```
plot(improved_model, which=1)
```



Linearity ARGET_WINS ~ TEAM_BATTING_H + TEAM_BATTING_HR + TEAM_BATTING_BB +

Our diagnostic plots show a fairly linear model. ##### Normality Check:

```
plot(improved_model, which=2)
```



ARGET_WINS ~ TEAM_BATTING_H + TEAM_BATTING_HR + TEAM_BATTING_BB +

```
shapiro.test(residuals(improved_model))
```

```
##
```

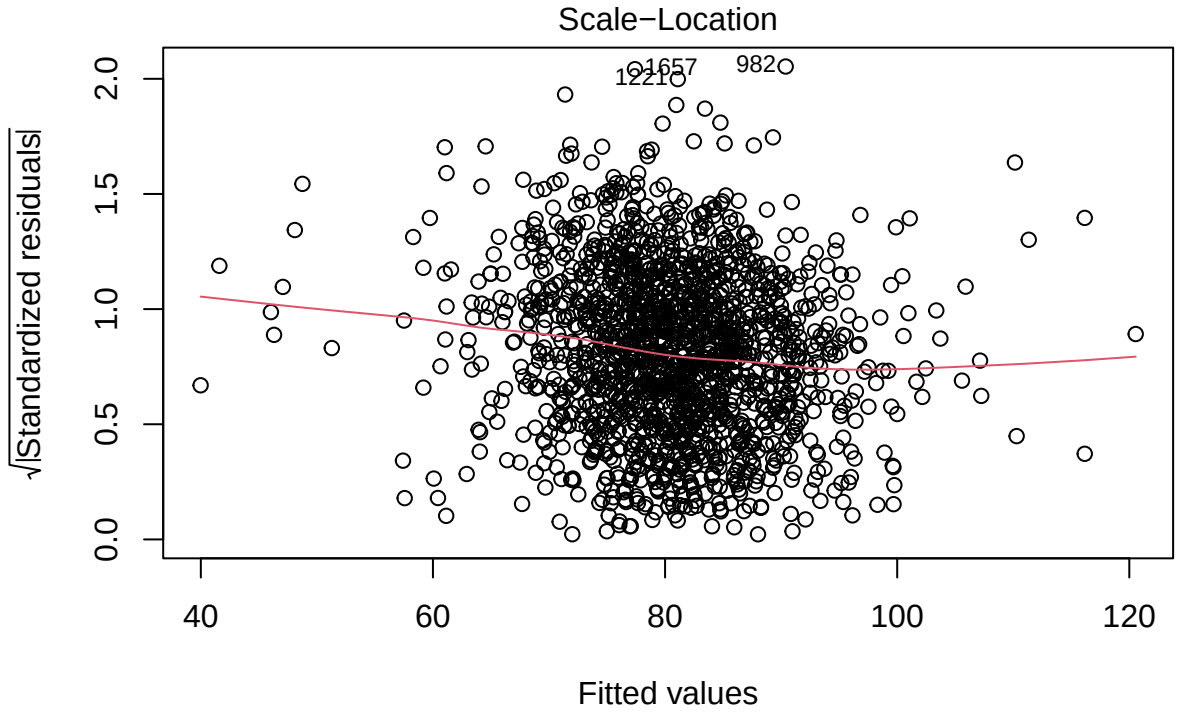
```
##  Shapiro-Wilk normality test
##
## data:  residuals(improved_model)
## W = 0.99726, p-value = 0.00452
```

```
# Shapiro-Wilk normality test: look for high p-value
```

Our QQ plot suggests normality thought there is obvious skewing on the tails, particularly on the right.

A Shapiro Wilk's Test statistic had a value of W = 0.9971 suggesting normality.

However, the p-value (0.0033) is less than < 0.05 suggesting that residuals do not follow a normal distribution.

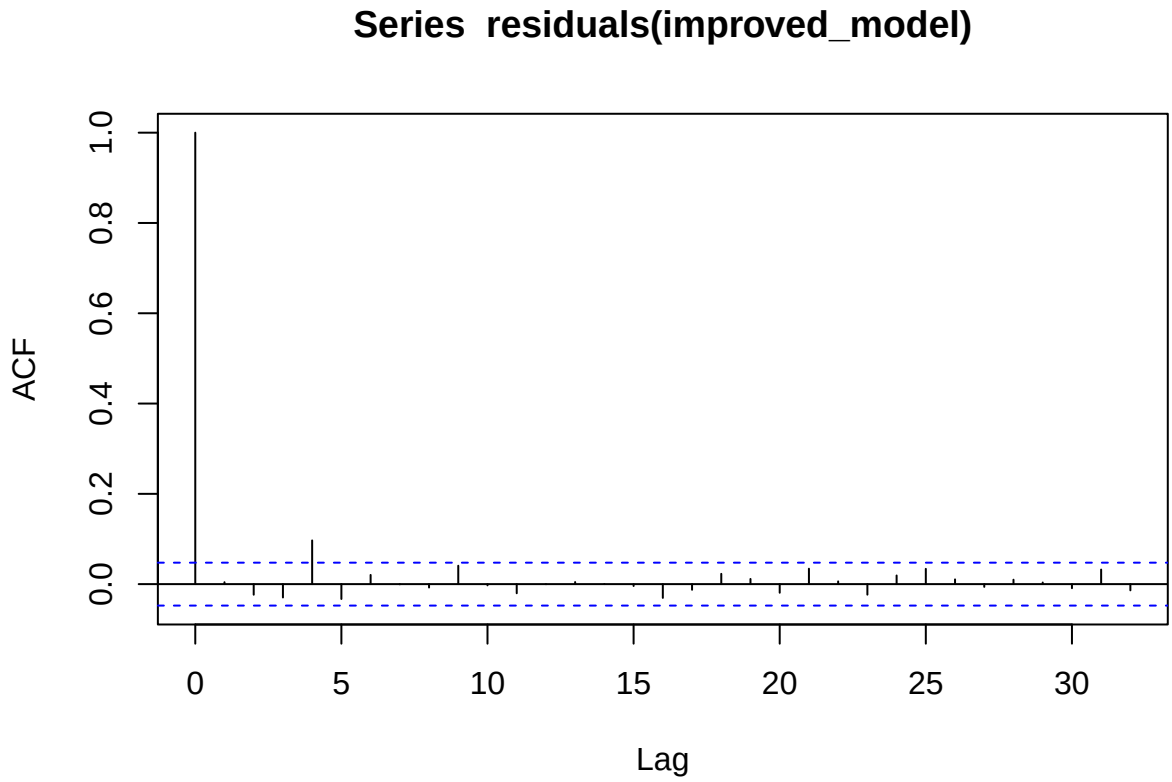


Heterocedasticity:

```
##
##  studentized Breusch-Pagan test
##
## data:  improved_model
## BP = 173.8, df = 7, p-value < 2.2e-16
```

Test Statistic (BP = 200.46) suggest higher evidence of heteroscedasticity. Additionally, the p-value is extremely small (much less than 0.05), suggesting higher evidence of heteroscedasticity and that we may need to transform the data to meet the Assumption of Homoscedasticity.

```
acf(residuals(improved_model))
```



Independence:

```
durbinWatsonTest(improved_model)
```

```
##  lag Autocorrelation D-W Statistic p-value
##    1     0.004261393       1.99092   0.892
##  Alternative hypothesis: rho != 0
```

```
# Durbin Watson should be close to 2
```

Our Autocorrelation Function shows that there are lags above the blue dashed line, suggesting no autocorrelation. This is confirmed through a Durbin-Watson test statistic value of 2.03 and an autocorrelation value of -0.0017 which suggest that our model's residuals are independent and do not violate the Independence Assumption.

TEAM_BATTING_H, TEAM_BATTING_2B, TEAM_PITCHING_SO, TEAM_FIELDING_E are in the range of (1-5) - No significant multicollinearity (good)

TEAM_BATTING_HR, TEAM_BATTING_HR:TEAM_BATTING_BB are in the range of (> 10) shows Severe multicollinearity (highly problematic).

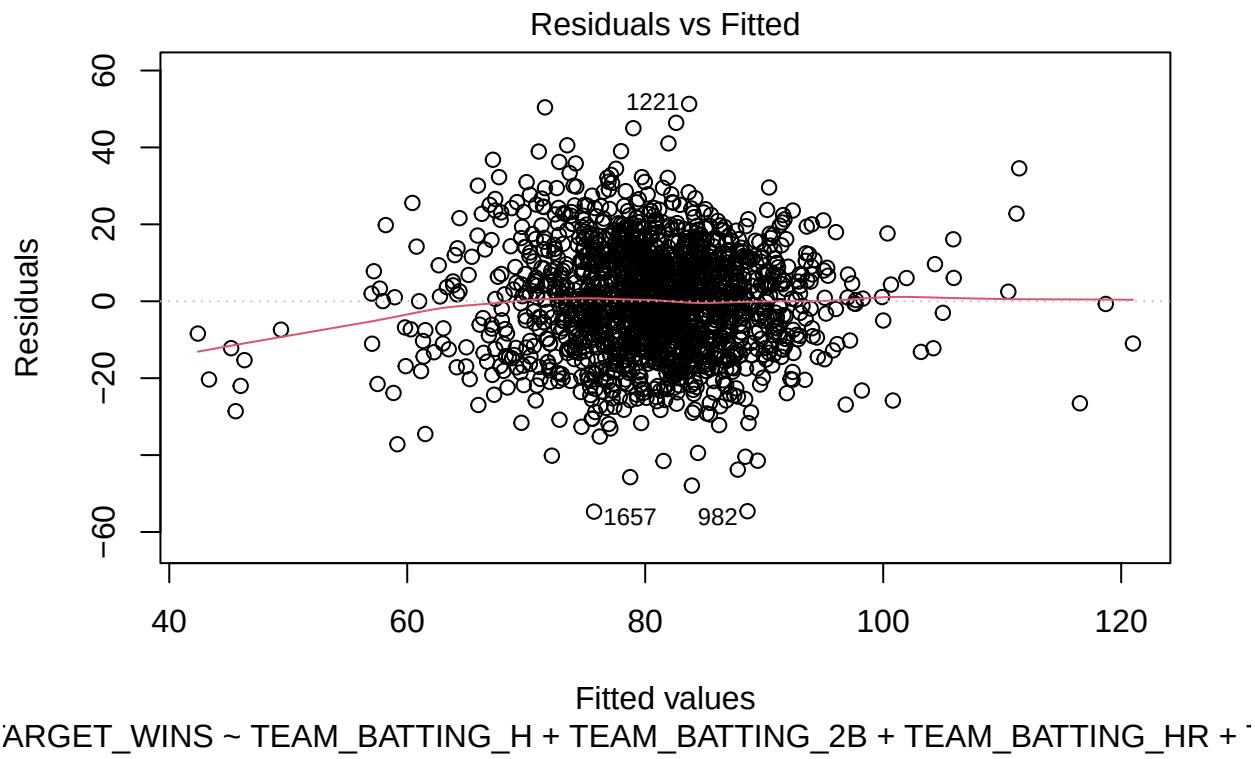
Model H4: High-Impact Features Model (Based on Correlation & VIF) We select variables based on correlation with TARGET_WINS and ensure they are not highly correlated with each other (VIF < 5).

```
library(car)

# Manually selected high-impact variables
high_impact_model <- lm(TARGET_WINS ~ TEAM_BATTING_H + TEAM_BATTING_2B + TEAM_BATTING_HR +
                        TEAM_PITCHING_HR + TEAM_PITCHING_SO + TEAM_FIELDING_E, data = stp75_train_df)

# View model summary
summary(high_impact_model)
```

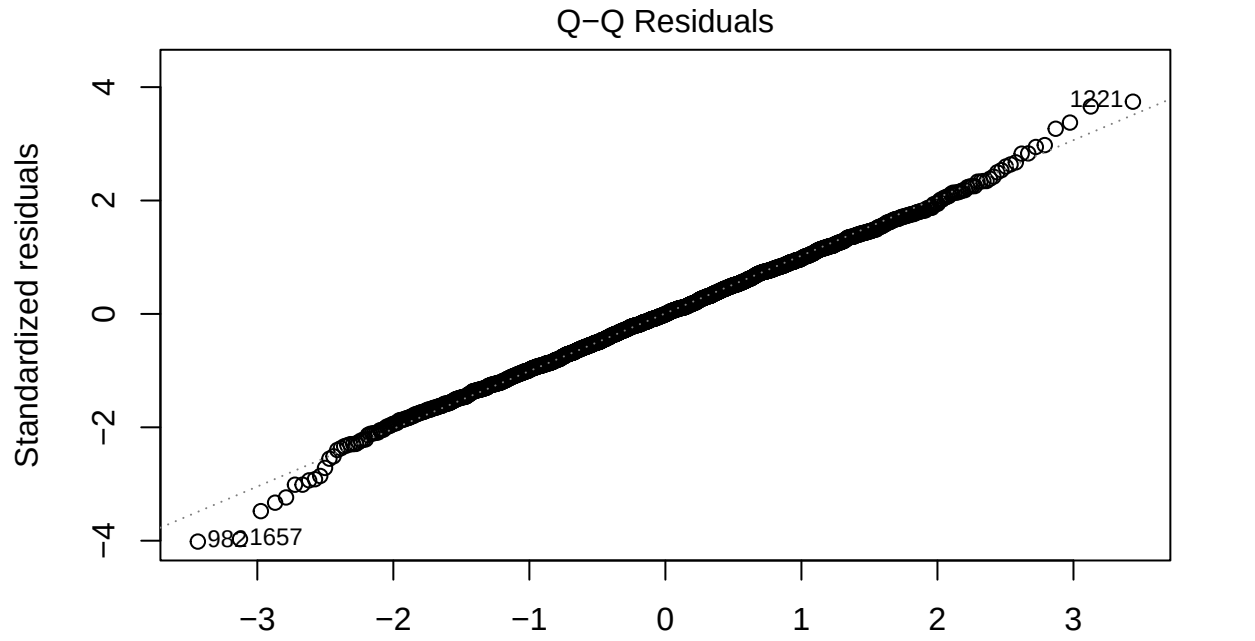
```
##
```

```
## Call:
## lm(formula = TARGET_WINS ~ TEAM_BATTING_H + TEAM_BATTING_2B +
##     TEAM_BATTING_HR + TEAM_PITCHING_HR + TEAM_PITCHING_SO + TEAM_FIELDING_E,
##     data = stp75_train_df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -54.696  -9.327  -0.052   9.563  51.309
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)       5.7981897  4.9547266   1.170  0.24207
## TEAM_BATTING_H    0.0593675  0.0040529  14.648  < 2e-16 ***
## TEAM_BATTING_2B  -0.0329084  0.0109767  -2.998  0.00276 **
## TEAM_BATTING_HR   0.0245345  0.0255845   0.959  0.33772
## TEAM_PITCHING_HR -0.0202476  0.0240754  -0.841  0.40046
## TEAM_PITCHING_SO  0.0006704  0.0016199   0.414  0.67905
## TEAM_FIELDING_E  -0.0212629  0.0024343  -8.735  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.8 on 1695 degrees of freedom
## Multiple R-squared:   0.23,  Adjusted R-squared:  0.2272
## F-statistic: 84.37 on 6 and 1695 DF,  p-value: < 2.2e-16
```

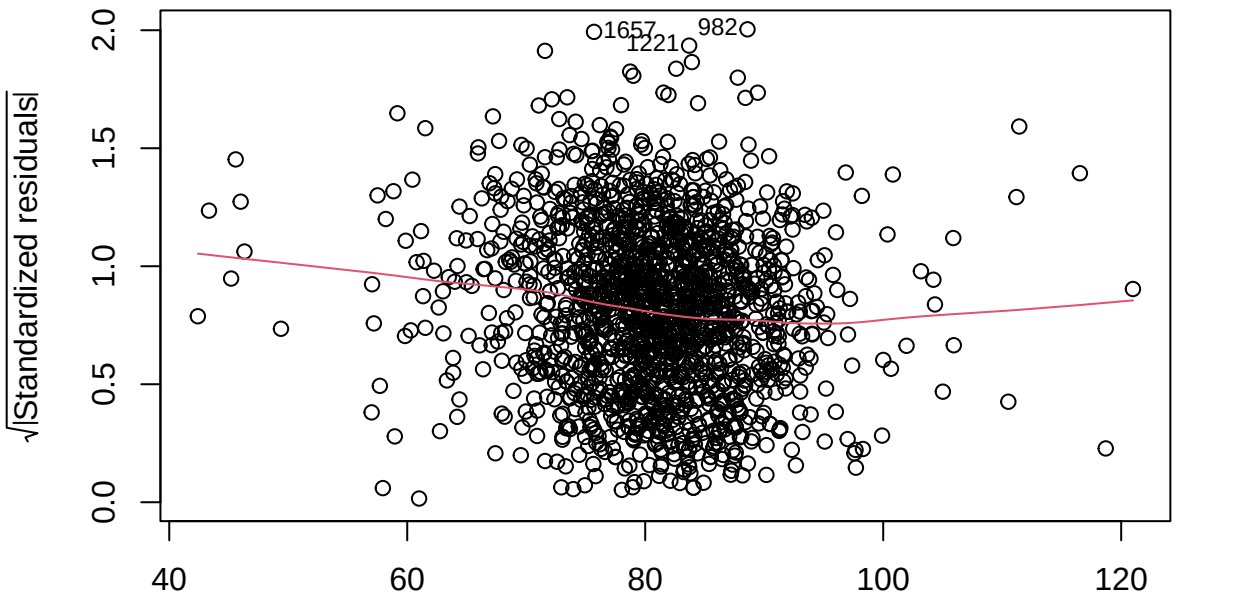
```
plot(high_impact_model)
```



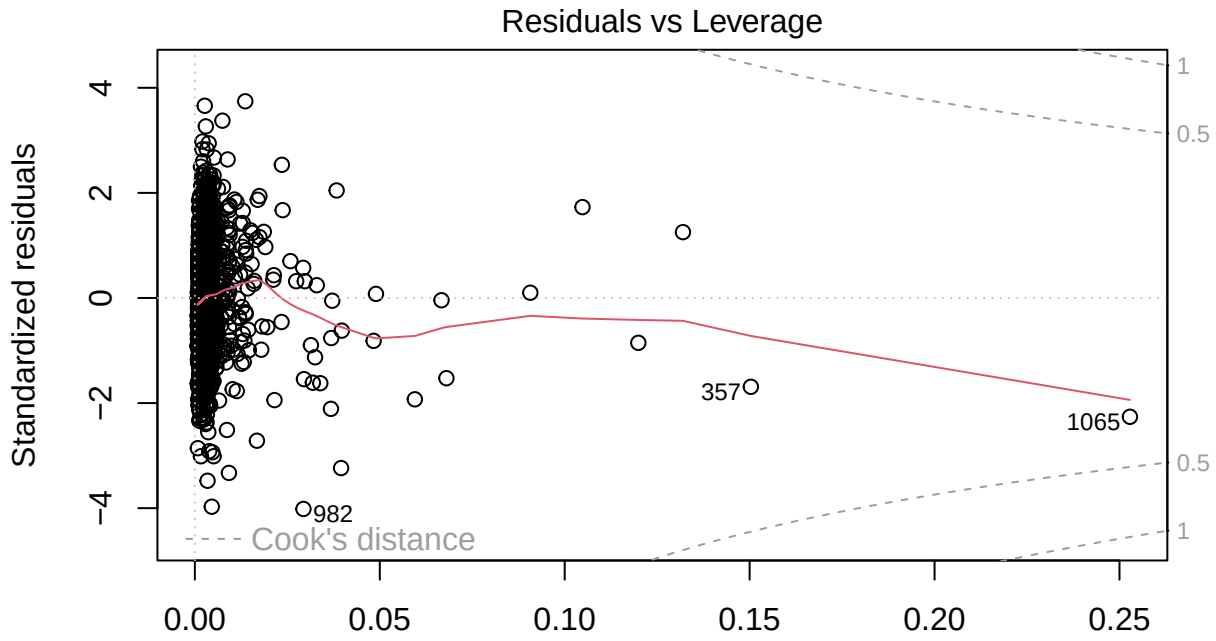
ARGET_WINS ~ TEAM_BATTING_H + TEAM_BATTING_2B + TEAM_BATTING_HR +



ARGET_WINS ~ TEAM_BATTING_H + TEAM_BATTING_2B + TEAM_BATTING_HR +



ARGET_WINS ~ TEAM_BATTING_H + TEAM_BATTING_2B + TEAM_BATTING_HR +



ARGET_WINS ~ TEAM_BATTING_H + TEAM_BATTING_2B + TEAM_BATTING_HR + T

```
# Check for multicollinearity
vif(high_impact_model)
```

```
##   TEAM_BATTING_H  TEAM_BATTING_2B  TEAM_BATTING_HR TEAM_PITCHING_HR
##         3.067139         2.400852        21.595294        19.816415
## TEAM_PITCHING_SO  TEAM_FIELDING_E
##         1.988444         2.606747
```

Model H5: Basic Multiple Linear Regression with Key Predictors Selected based on key offensive and defensive metrics impacting wins

```
# Selected based on key offensive and defensive metrics impacting wins
target_vars1 <- c("TEAM_BATTING_H", "TEAM_BATTING_HR", "TEAM_BATTING_BB", "TEAM_BASERUN_SB", "TEAM_FIELI
pj_model1 <- lm(TARGET_WINS ~ ., data = stp75_train_df[, c("TARGET_WINS", target_vars1)])
summary(pj_model1)
```

```
##
## Call:
## lm(formula = TARGET_WINS ~ ., data = stp75_train_df[, c("TARGET_WINS",
##     target_vars1)])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -51.189  -9.157  -0.198   9.026  52.306
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)      4.164407   3.753335   1.110   0.2674
## TEAM_BATTING_H   0.049443   0.002427  20.375   <2e-16 ***
## TEAM_BATTING_HR  0.014907   0.007280   2.048   0.0407 *
## TEAM_BATTING_BB  0.004943   0.003836   1.289   0.1977
```

```
## TEAM_BASERUN_SB  0.039060   0.004406   8.865   <2e-16 ***
## TEAM_FIELDING_E -0.020160   0.002292  -8.794   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.46 on 1696 degrees of freedom
## Multiple R-squared:  0.2673, Adjusted R-squared:  0.2652
## F-statistic: 123.8 on 5 and 1696 DF,  p-value: < 2.2e-16
```

Model H6: Adding Interaction Terms Exploring interaction between hits and home runs as a factor in wins/

```
# Exploring interaction between hits and home runs as a factor in wins
target_vars2 <- c("TEAM_BATTING_H", "TEAM_BATTING_HR", "TEAM_BATTING_BB", "TEAM_BASERUN_SB", "TEAM_FIELI
pj_model2 <- lm(TARGET_WINS ~ TEAM_BATTING_H * TEAM_BATTING_HR + TEAM_BATTING_BB + TEAM_BASERUN_SB + TE
summary(pj_model2)
```

```
##
## Call:
## lm(formula = TARGET_WINS ~ TEAM_BATTING_H * TEAM_BATTING_HR +
##     TEAM_BATTING_BB + TEAM_BASERUN_SB + TEAM_FIELDING_E, data = stp75_train_df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -50.740  -9.092  -0.133   8.976  52.556
##
## Coefficients:
##                                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)                    -4.207e+00  5.772e+00  -0.729   0.4662
## TEAM_BATTING_H                  5.492e-02  3.755e-03  14.623   <2e-16 ***
## TEAM_BATTING_HR                 1.361e-01  6.393e-02   2.129   0.0334 *
## TEAM_BATTING_BB                 5.442e-03  3.842e-03   1.416   0.1568
## TEAM_BASERUN_SB                 3.968e-02  4.415e-03   8.988   <2e-16 ***
## TEAM_FIELDING_E                -2.031e-02  2.292e-03  -8.862   <2e-16 ***
## TEAM_BATTING_H:TEAM_BATTING_HR -8.218e-05  4.307e-05  -1.908   0.0565 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.45 on 1695 degrees of freedom
## Multiple R-squared:  0.2689, Adjusted R-squared:  0.2663
## F-statistic: 103.9 on 6 and 1695 DF,  p-value: < 2.2e-16
```

Model H7: Incorporating Pitching and Defensive Metrics Introducing pitching metrics to assess their impact on win prediction

```
# Introducing pitching metrics to assess their impact on win prediction
target_vars4 <- c("TEAM_BATTING_H", "TEAM_BATTING_HR", "TEAM_PITCHING_SO", "TEAM_PITCHING_BB", "TEAM_FI
pj_model4 <- lm(TARGET_WINS ~ ., data = stp75_train_df[, c("TARGET_WINS", target_vars4)])
summary(pj_model4)
```

```
##
## Call:
## lm(formula = TARGET_WINS ~ ., data = stp75_train_df[, c("TARGET_WINS",
##     target_vars4)])
##
```

```
## Residuals:
##     Min      1Q  Median      3Q     Max
## -52.836  -9.316  -0.051   9.409  50.424
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)      11.4450377  4.3542451   2.628  0.00865 **
## TEAM_BATTING_H    0.0490599  0.0029278  16.757  < 2e-16 ***
## TEAM_BATTING_HR   0.0001236  0.0082912   0.015  0.98811
## TEAM_PITCHING_SO -0.0017865  0.0015197  -1.176  0.23993
## TEAM_PITCHING_BB  0.0059032  0.0023796   2.481  0.01321 *
## TEAM_FIELDING_E  -0.0189410  0.0021981  -8.617  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.81 on 1696 degrees of freedom
## Multiple R-squared:  0.2285, Adjusted R-squared:  0.2262
## F-statistic: 100.5 on 5 and 1696 DF,  p-value: < 2.2e-16
```

Model H8: Including Caught Stealing and Doubles Analyzing the impact of baserunning decisions on team wins

```
# Analyzing the impact of baserunning decisions on team wins 5
target_vars5 <- c("TEAM_BATTING_H", "TEAM_BATTING_HR", "TEAM_BATTING_2B", "TEAM_BASERUN_CS", "TEAM_FIELI
pj_model5 <- lm(TARGET_WINS ~ ., data = stp75_train_df[, c("TARGET_WINS", target_vars5)])
summary(pj_model5)
```

```
##
## Call:
## lm(formula = TARGET_WINS ~ ., data = stp75_train_df[, c("TARGET_WINS",
##     target_vars5)])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -54.826  -9.301  -0.022   9.588  51.730
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      7.909423   3.702699   2.136  0.03281 *
## TEAM_BATTING_H   0.058189   0.003314  17.560  < 2e-16 ***
## TEAM_BATTING_HR  0.004533   0.007644   0.593  0.55328
## TEAM_BATTING_2B -0.031924   0.010577  -3.018  0.00258 **
## TEAM_BASERUN_CS -0.002427   0.018223  -0.133  0.89408
## TEAM_FIELDING_E -0.021685   0.002070 -10.475  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.8 on 1696 degrees of freedom
## Multiple R-squared:  0.2296, Adjusted R-squared:  0.2274
## F-statistic: 101.1 on 5 and 1696 DF,  p-value: < 2.2e-16
```

Model H9: Weighted Batting and Pitching Metrics Creating composite metric for batting and pitching efficiency

```r
# Creating composite metric for batting and pitching efficiency
pj_model6 <- lm(TARGET_WINS ~ (TEAM_BATTING_H + TEAM_BATTING_BB) / TEAM_BATTING_SO + (TEAM_PITCHING_SO
summary(pj_model6)
```

```
##
## Call:
## lm(formula = TARGET_WINS ~ (TEAM_BATTING_H + TEAM_BATTING_BB)/TEAM_BATTING_SO +
##     (TEAM_PITCHING_SO - TEAM_PITCHING_BB), data = stp75_train_df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -60.050  -9.033   0.334   9.451  48.060
##
## Coefficients:
##                                              Estimate Std. Error t value
## (Intercept)                                 7.943e+00  4.813e+00   1.650
## TEAM_BATTING_H                              4.291e-02  2.497e-03  17.183
## TEAM_BATTING_BB                             1.768e-02  4.967e-03   3.561
## TEAM_PITCHING_SO                           -5.282e-03  1.697e-03  -3.113
## TEAM_BATTING_H:TEAM_BATTING_BB:TEAM_BATTING_SO  9.344e-09  2.943e-09   3.175
##                                            Pr(>|t|)
## (Intercept)                                 0.09907 .
## TEAM_BATTING_H                              < 2e-16 ***
## TEAM_BATTING_BB                             0.00038 ***
## TEAM_PITCHING_SO                            0.00188 **
## TEAM_BATTING_H:TEAM_BATTING_BB:TEAM_BATTING_SO  0.00152 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.9 on 1697 degrees of freedom
## Multiple R-squared:  0.2175, Adjusted R-squared:  0.2157
## F-statistic:   118 on 4 and 1697 DF,  p-value: < 2.2e-16
```

Models with Applied Transformations

So far, our diagnostic plots show that our model is somewhat linear, independent, heteroschedastic, but our residuals vs leverage indicates the presence of outliers. We will attempt to use transformations to improve our model.

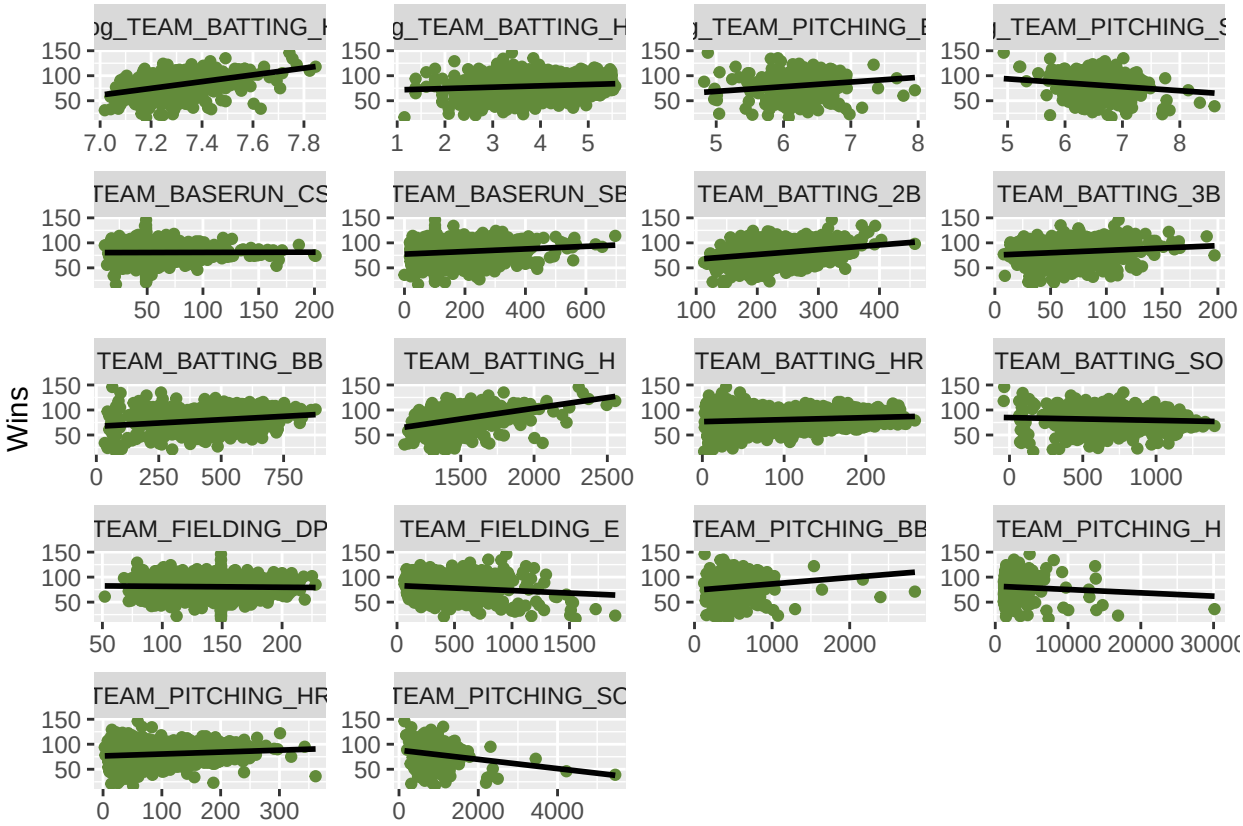
Log transformation Log Transformations reduce the influence of outliers by producing stable regression coefficients with reduced variability and can improves the model fit for non-linear relationships. They can address right-skewed distributions (e.g., extreme HR and SO values) and help meet the linear regression assumption of normality.

Model T1: Log-Transformed Model (Handling Skewness & Outliers) Some baseball statistics (e.g., Home Runs, Strikeouts) have skewed distributions. We apply log transformation to stabilize variance.

```r
# Apply log transformation to selected variables
train_df_log <- stp75_train_df %>%
  mutate(
    log_TEAM_BATTING_H = log1p(TEAM_BATTING_H),
    log_TEAM_BATTING_HR = log1p(TEAM_BATTING_HR),
    log_TEAM_PITCHING_SO = log1p(TEAM_PITCHING_SO),
    log_TEAM_PITCHING_BB = log(TEAM_PITCHING_BB),
```

```
  )

test_df_log <- stp25_test_df %>%
  mutate(
    log_TEAM_BATTING_H = log1p(TEAM_BATTING_H),
    log_TEAM_BATTING_HR = log1p(TEAM_BATTING_HR),
    log_TEAM_PITCHING_SO = log1p(TEAM_PITCHING_SO),
    log_TEAM_PITCHING_BB = log1p(TEAM_PITCHING_BB),
  )

train_df_log %>%
  gather(variable, value, -TARGET_WINS) %>%
  ggplot(., aes(value, TARGET_WINS)) +
  geom_point(fill = "#628B3A", color="#628B3A")  +
  geom_smooth(method = "lm", se = FALSE, color = "black") +
  facet_wrap(~variable, scales ="free", ncol = 4) +
  labs(x = element_blank(), y = "Wins")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



Our scatterplots show some improvements.

```
# Fit model with transformed variables
log_model1 <- lm(TARGET_WINS ~ log_TEAM_BATTING_H + log_TEAM_BATTING_HR + log_TEAM_PITCHING_SO + TEAM_F

# View model summary
summary(log_model1)
```

```
##
## Call:
## lm(formula = TARGET_WINS ~ log_TEAM_BATTING_H + log_TEAM_BATTING_HR +
##     log_TEAM_PITCHING_SO + TEAM_FIELDING_E, data = train_df_log)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -52.985  -9.340   0.243   9.474  48.553
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)          -5.681e+02  3.915e+01 -14.509  < 2e-16 ***
## log_TEAM_BATTING_H    8.819e+01  4.779e+00  18.452  < 2e-16 ***
## log_TEAM_BATTING_HR  -2.398e+00  6.554e-01  -3.658 0.000262 ***
## log_TEAM_PITCHING_SO  3.371e+00  1.389e+00   2.427 0.015333 *
## TEAM_FIELDING_E      -2.457e-02  2.319e-03 -10.593  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.8 on 1697 degrees of freedom
## Multiple R-squared:  0.2297, Adjusted R-squared:  0.2279
## F-statistic: 126.5 on 4 and 1697 DF,  p-value: < 2.2e-16
```

Model Diagnostics Multicollinearity

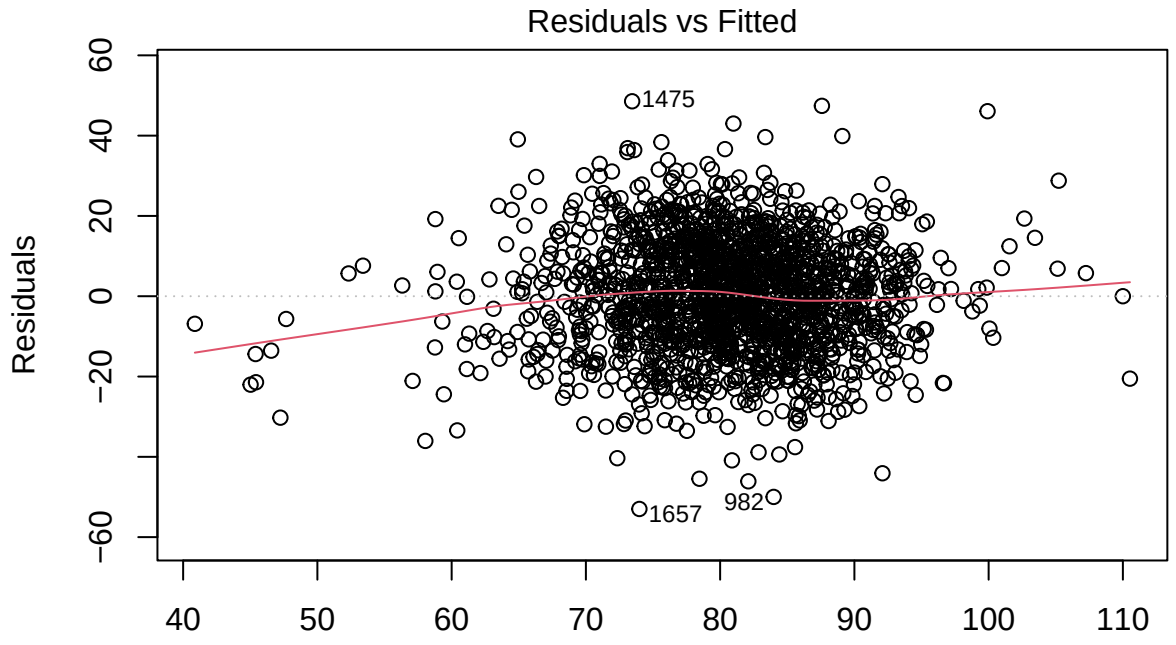
Our VIF Test

```
vif(log_model1)
```

```
##   log_TEAM_BATTING_H  log_TEAM_BATTING_HR log_TEAM_PITCHING_SO
##             1.779623             2.716588             1.810904
##      TEAM_FIELDING_E
##             2.368167
```

Linearity

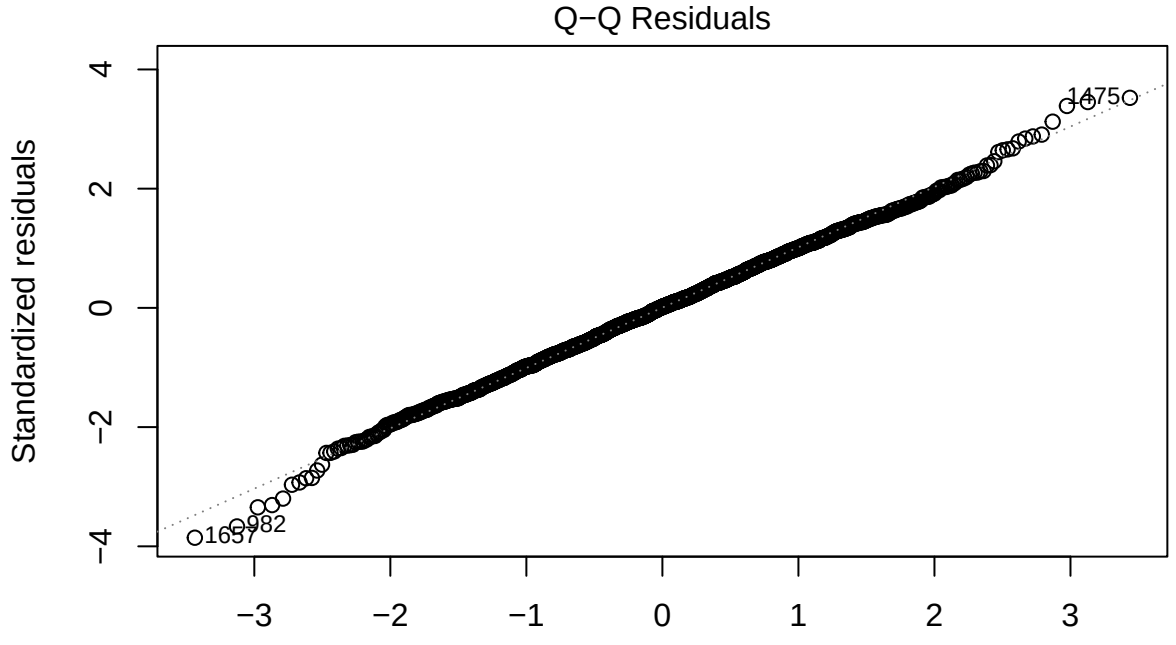
```
plot(log_model1, which=1)
```



Our diagnostic plots show a fairly linear model.

Normality

```
plot(log_model1, which=2)
```



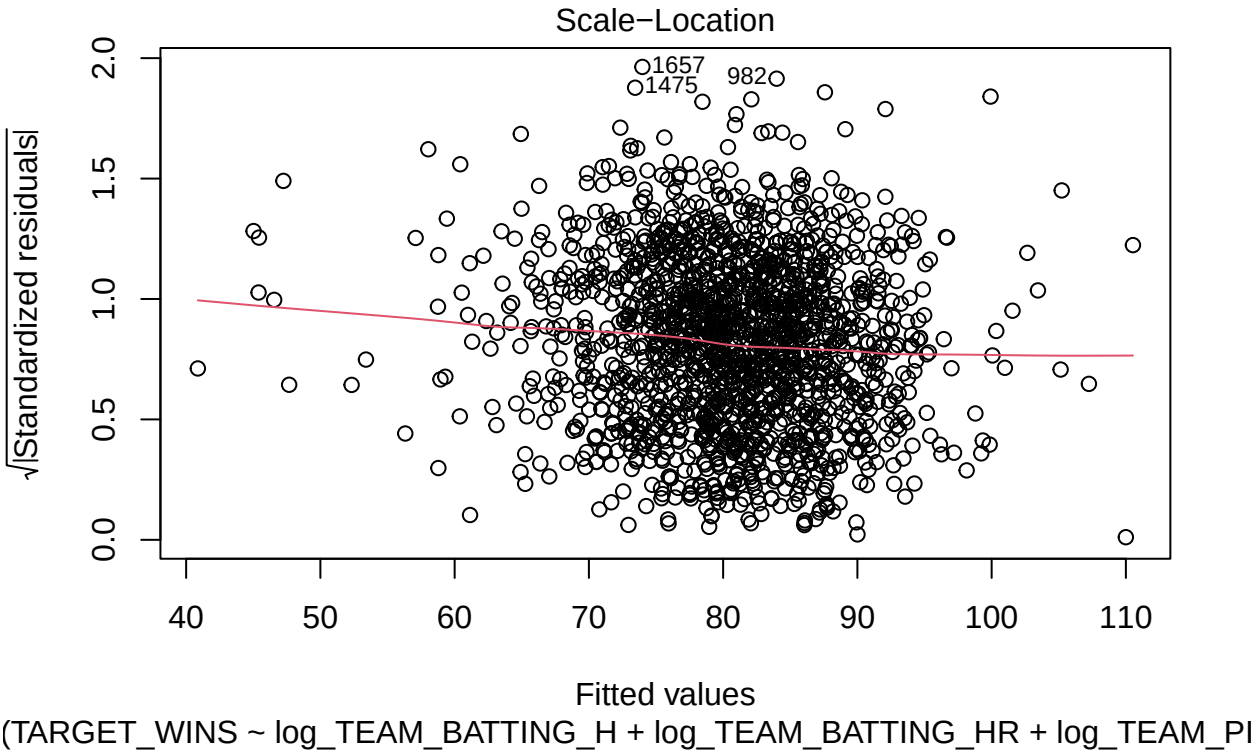
```
shapiro.test(residuals(log_model1))
```

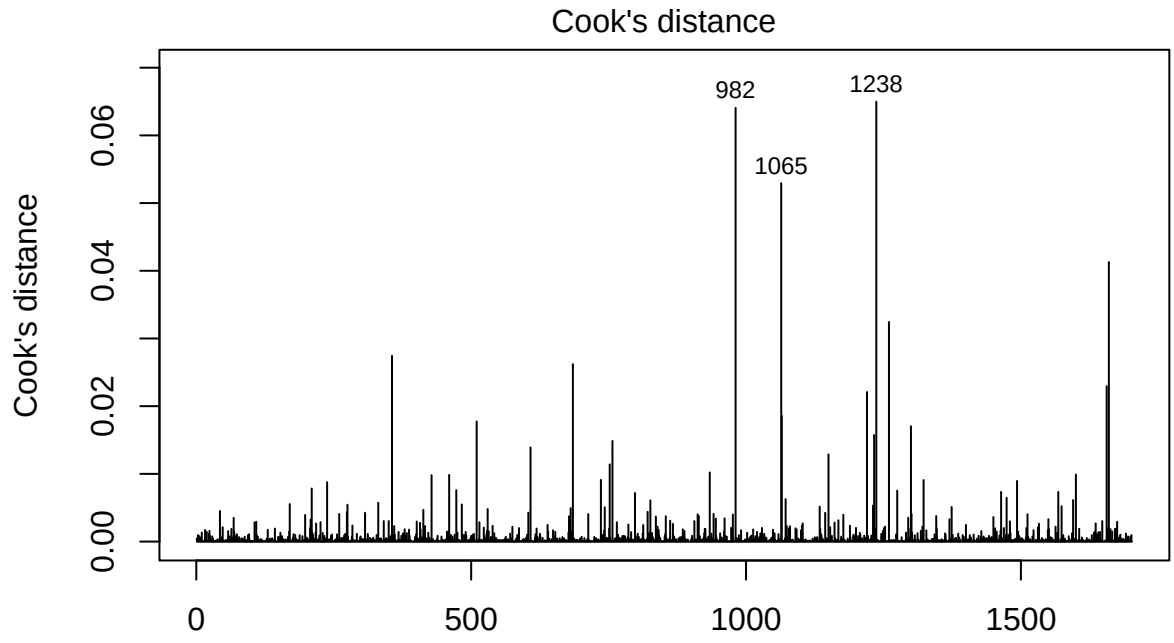
```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(log_model1)
## W = 0.99862, p-value = 0.1927
```

```
# Shapiro-Wilk normality test: look for high p-value
```

Our QQ plot suggests normality thought there is obvious skewing on the tails, particularly on the right. A Shapiro Wilk's Test statistic restulted in a value of 0.9986 suggesting normality.

Heteroscedasticity:





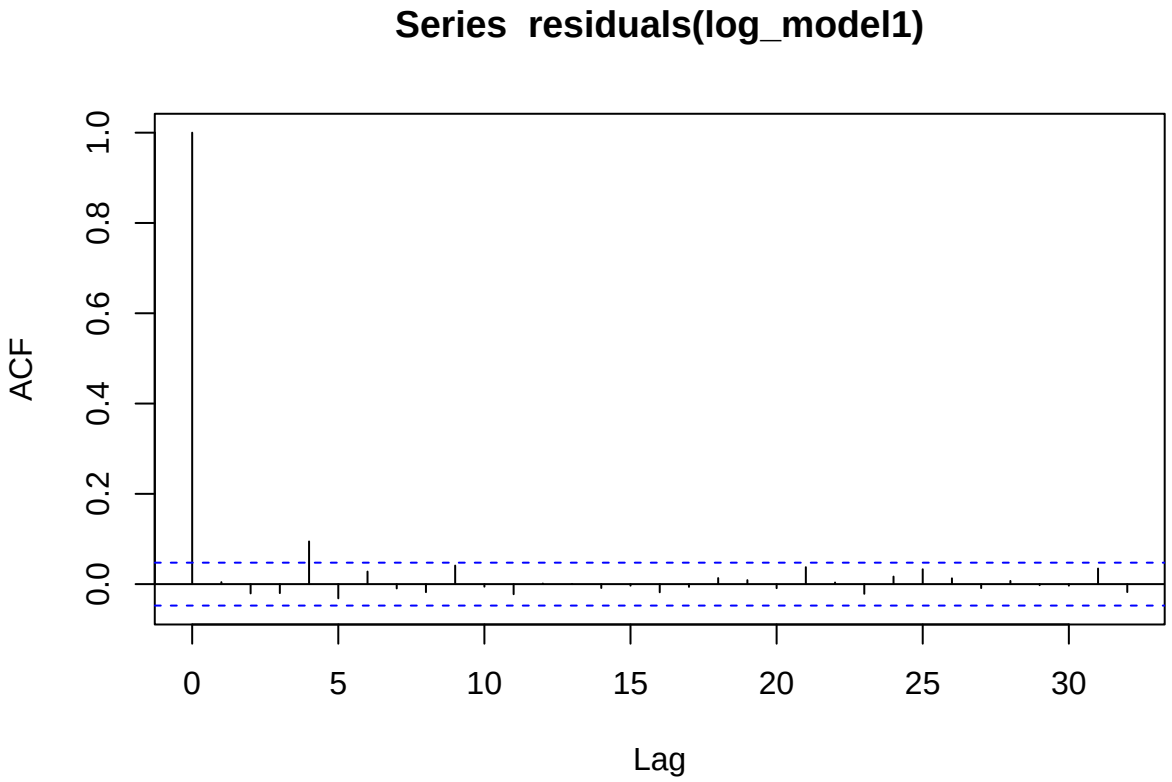

```
##
##  studentized Breusch-Pagan test
##
## data:  log_model1
## BP = 152.45, df = 4, p-value < 2.2e-16
```

Our Scale-Location plot shows no major improvement than our previous models. While the points appear somewhat evenly distributed above and below the trend line and there is no obvious fan/wedge pattern, there is clustering in the center suggesting underfitting, high leverage outliers or that additional transformation may be needed. As our Cook's Distance plot has no values with a greater than 1, we can rule our the effects of high leverage points.

The Breusch-Page test statistic BP (152.45) and the small p-value (2.2e-16) suggest evidence of heteroscedasticity. Though the values are different that we may need to transform the data to meet the Assumption of Homoscedasticity.

Independence:

```
acf(residuals(log_model1))
```




```
durbinWatsonTest(log_model1)
```

```
##  lag Autocorrelation D-W Statistic p-value
##    1     0.004317842      1.990818   0.812
##  Alternative hypothesis: rho != 0
```

```
# Durbin Watson should be close to 2
```

Our Autocorrelation Function and Durbin-Watson test suggest that our model's residuals are independent and therefore do not violate the Independence Assumption.

Model T2: Applying Log Transformation to Key Predictors Below is a variation of the above with some minor adjustments that exclude log_PITCHING_SO and include TEAM_BASERUN_SB.

```
# Transformation applied to account for skewed distributions in key predictors

target_vars3 <- c("log_TEAM_BATTING_H", "log_TEAM_BATTING_HR", "TEAM_BATTING_BB", "TEAM_BASERUN_SB", "TE
log_model2 <- lm(TARGET_WINS ~ ., data = train_df_log[, c("TARGET_WINS", target_vars3)])
summary(log_model2)
```

```
##
## Call:
## lm(formula = TARGET_WINS ~ ., data = train_df_log[, c("TARGET_WINS",
##     target_vars3)])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -52.560  -8.851  -0.049   8.855  52.489
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
```

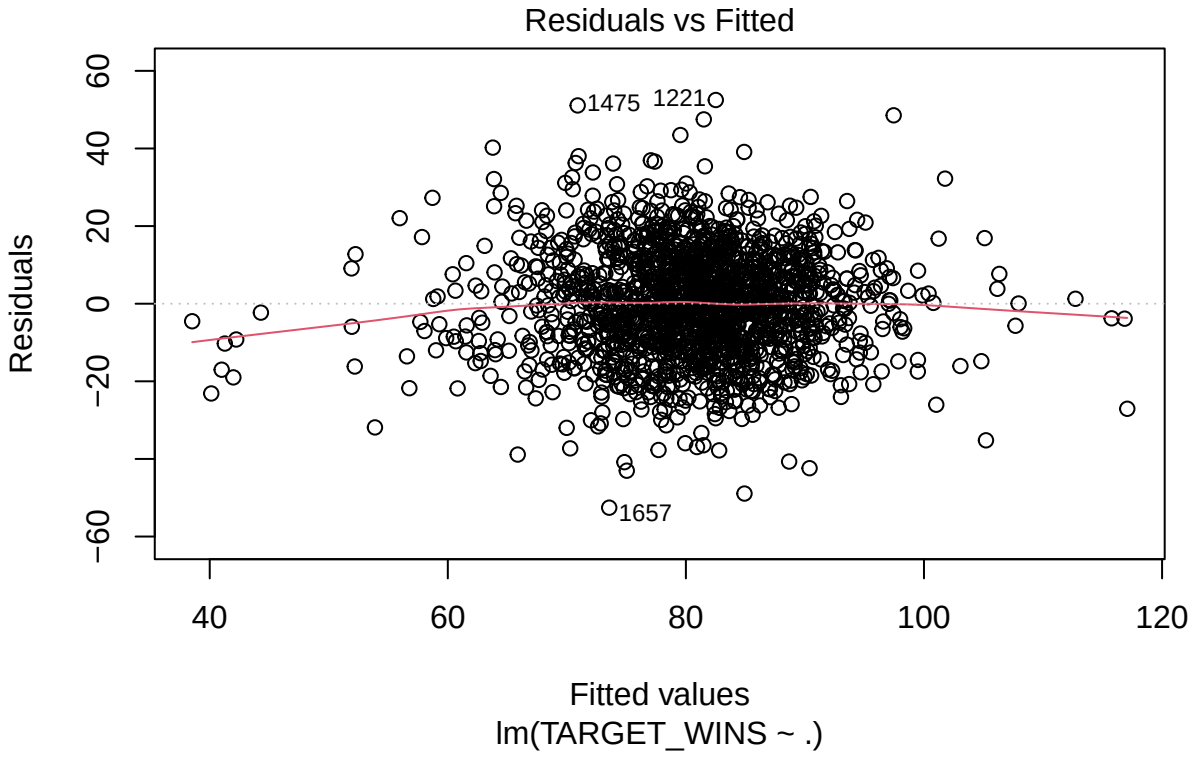
```
## (Intercept)         -4.867e+02  2.681e+01 -18.153  < 2e-16 ***
## log_TEAM_BATTING_H   7.771e+01  3.794e+00  20.480  < 2e-16 ***
## log_TEAM_BATTING_HR -4.234e-01  5.819e-01  -0.728   0.4670
## TEAM_BATTING_BB      7.382e-03  3.887e-03   1.899   0.0577 .
## TEAM_BASERUN_SB      3.488e-02  4.419e-03   7.893 5.23e-15 ***
## TEAM_FIELDING_E     -2.113e-02  2.372e-03  -8.909  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.49 on 1696 degrees of freedom
## Multiple R-squared:  0.2636, Adjusted R-squared:  0.2614
## F-statistic: 121.4 on 5 and 1696 DF,  p-value: < 2.2e-16
```

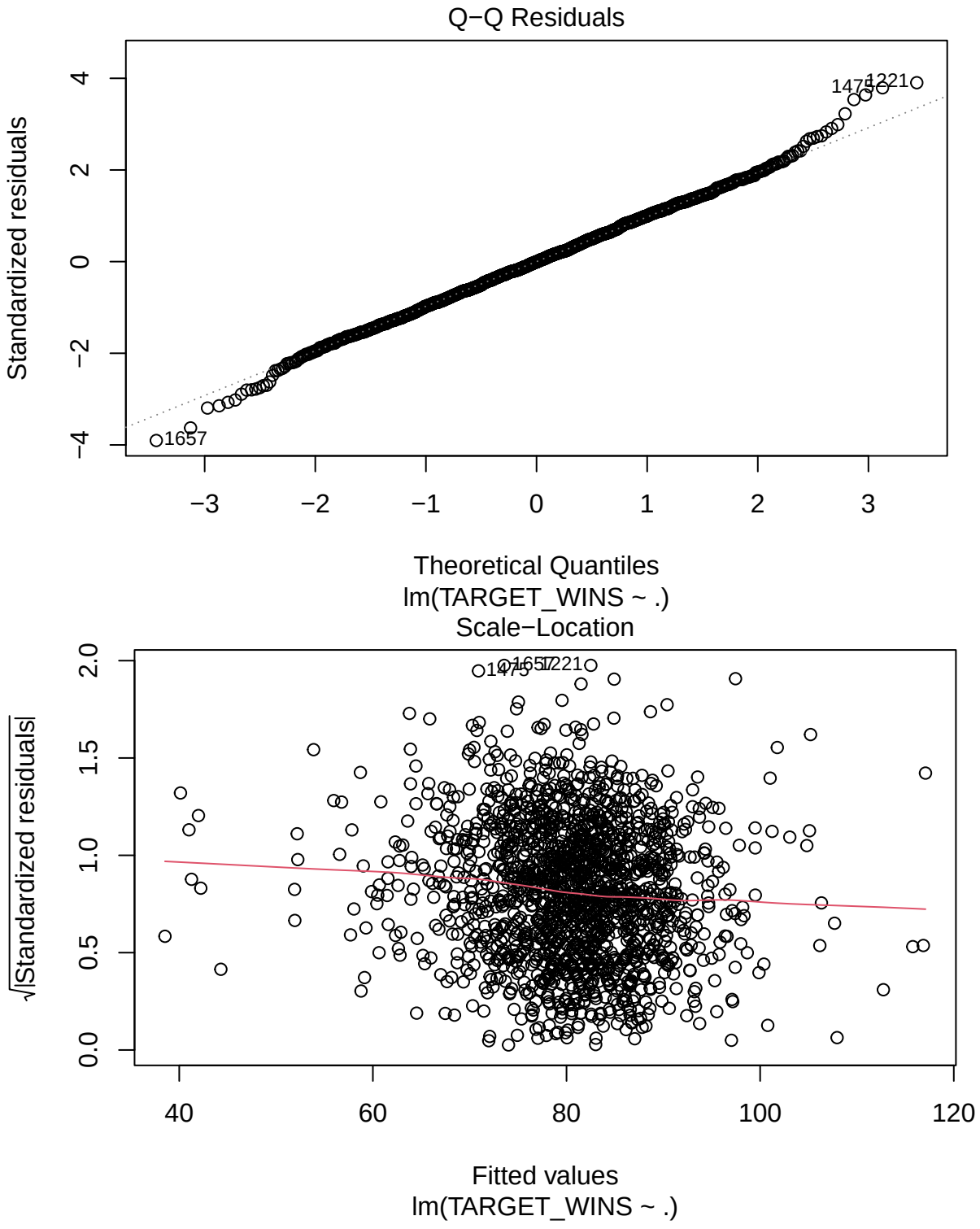
Model Diagnostics Our VIF Test and diagnostic plots show similar results as the diagnostics for Model T1.

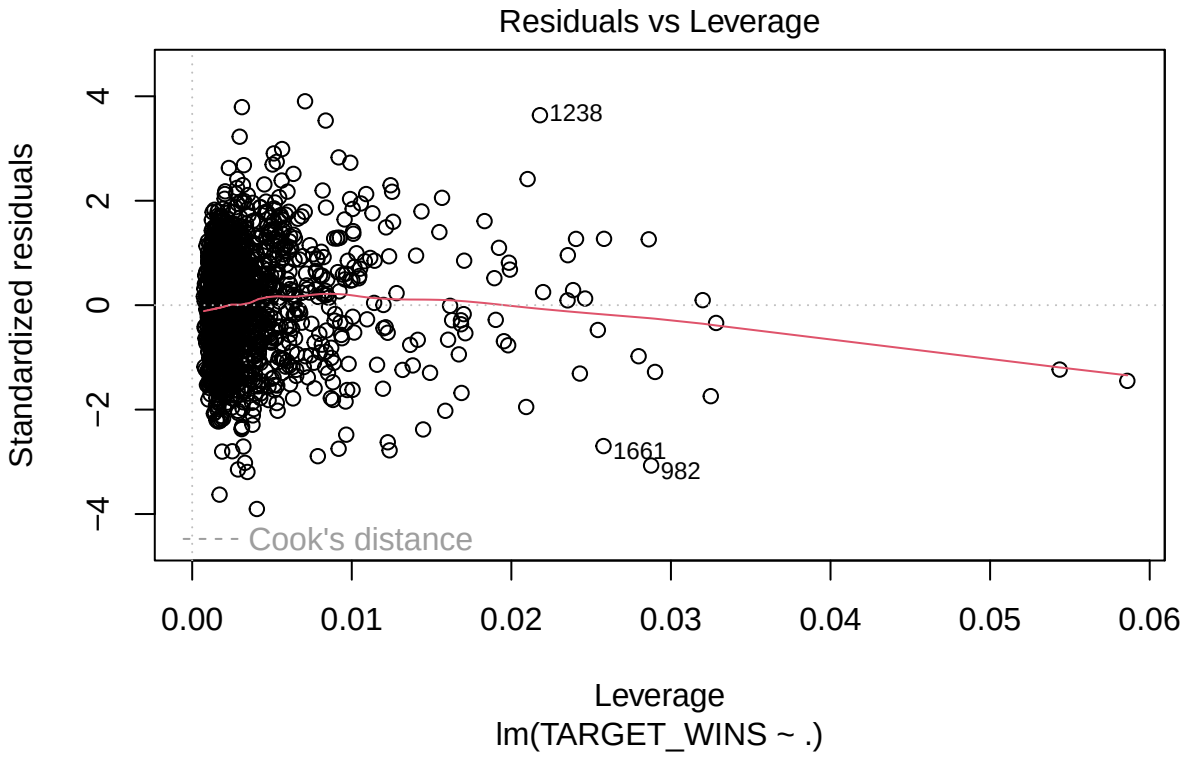
```
vif(log_model2)
```

```
##  log_TEAM_BATTING_H log_TEAM_BATTING_HR     TEAM_BATTING_BB     TEAM_BASERUN_SB
##            1.172671           2.238759            2.078787            1.363852
##     TEAM_FIELDING_E
##            2.589296
```

```
plot(log_model2)
```




Q−Q Residuals
Standardized residuals
Theoretical Quantiles
lm(TARGET_WINS ~ .)
Scale−Location
√|Standardized residuals|
Fitted values
lm(TARGET_WINS ~ .)



Our diagnostic plots show a fairly linear model.

```
stp_logy_model <- lm(log(TARGET_WINS + 1) ~., data = stp75_train_df)

# Backward step-wise regression
stpb_logy_model <- step(stp_logy_model, direction = "backward")
```

Model T3: Log Transformation on the Dependent Variable (Y)

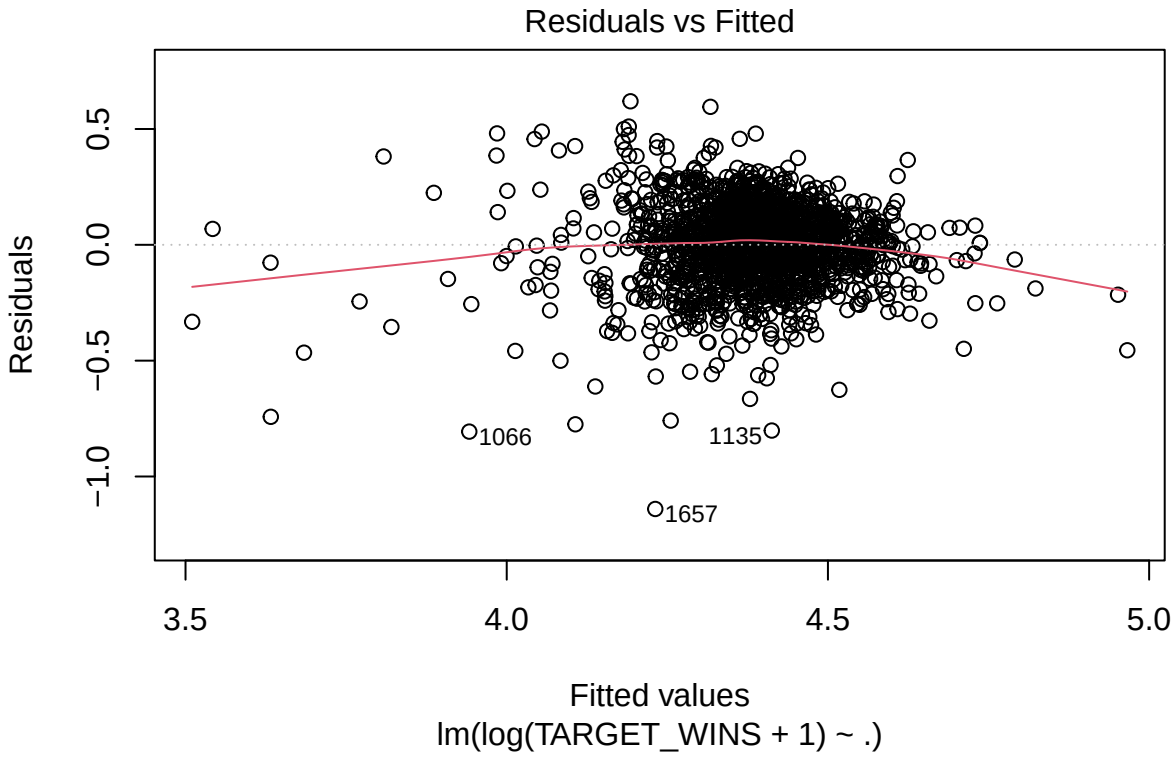
```
## Start:  AIC=-5905.5
## log(TARGET_WINS + 1) ~ TEAM_BATTING_H + TEAM_BATTING_2B + TEAM_BATTING_3B +
##     TEAM_BATTING_HR + TEAM_BATTING_BB + TEAM_BATTING_SO + TEAM_BASERUN_SB +
##     TEAM_BASERUN_CS + TEAM_PITCHING_H + TEAM_PITCHING_HR + TEAM_PITCHING_BB +
##     TEAM_PITCHING_SO + TEAM_FIELDING_E + TEAM_FIELDING_DP
##
##                     Df Sum of Sq    RSS     AIC
## - TEAM_BATTING_HR    1    0.0017 52.051 -5907.4
## - TEAM_PITCHING_BB   1    0.0034 52.053 -5907.4
## - TEAM_PITCHING_SO   1    0.0039 52.053 -5907.4
## - TEAM_BATTING_SO    1    0.0077 52.057 -5907.3
## - TEAM_BASERUN_CS    1    0.0112 52.060 -5907.1
## - TEAM_BATTING_BB    1    0.0165 52.066 -5907.0
## - TEAM_PITCHING_HR   1    0.0468 52.096 -5906.0
## <none>                          52.049 -5905.5
## - TEAM_BATTING_2B    1    0.0885 52.138 -5904.6
## - TEAM_PITCHING_H    1    0.2636 52.313 -5898.9
## - TEAM_BATTING_3B    1    0.4776 52.527 -5892.0
## - TEAM_BASERUN_SB    1    0.9296 52.979 -5877.4
## - TEAM_FIELDING_DP   1    1.7237 53.773 -5852.0
## - TEAM_FIELDING_E    1    2.1095 54.159 -5839.9
```

```
## - TEAM_BATTING_H    1    3.8849 55.934 -5785.0
##
## Step:  AIC=-5907.45
## log(TARGET_WINS + 1) ~ TEAM_BATTING_H + TEAM_BATTING_2B + TEAM_BATTING_3B +
##     TEAM_BATTING_BB + TEAM_BATTING_SO + TEAM_BASERUN_SB + TEAM_BASERUN_CS +
##     TEAM_PITCHING_H + TEAM_PITCHING_HR + TEAM_PITCHING_BB + TEAM_PITCHING_SO +
##     TEAM_FIELDING_E + TEAM_FIELDING_DP
##
##                    Df Sum of Sq    RSS     AIC
## - TEAM_PITCHING_BB  1    0.0021 52.053 -5909.4
## - TEAM_PITCHING_SO  1    0.0033 52.054 -5909.3
## - TEAM_BATTING_SO   1    0.0064 52.057 -5909.2
## - TEAM_BASERUN_CS   1    0.0122 52.063 -5909.0
## - TEAM_BATTING_BB   1    0.0228 52.074 -5908.7
## <none>                          52.051 -5907.4
## - TEAM_BATTING_2B   1    0.0903 52.141 -5906.5
## - TEAM_PITCHING_H   1    0.3306 52.382 -5898.7
## - TEAM_BATTING_3B   1    0.4849 52.536 -5893.7
## - TEAM_PITCHING_HR  1    0.7126 52.764 -5886.3
## - TEAM_BASERUN_SB   1    0.9292 52.980 -5879.3
## - TEAM_FIELDING_DP  1    1.7221 53.773 -5854.0
## - TEAM_FIELDING_E   1    2.1146 54.166 -5841.7
## - TEAM_BATTING_H    1    4.0738 56.125 -5781.2
##
## Step:  AIC=-5909.38
## log(TARGET_WINS + 1) ~ TEAM_BATTING_H + TEAM_BATTING_2B + TEAM_BATTING_3B +
##     TEAM_BATTING_BB + TEAM_BATTING_SO + TEAM_BASERUN_SB + TEAM_BASERUN_CS +
##     TEAM_PITCHING_H + TEAM_PITCHING_HR + TEAM_PITCHING_SO + TEAM_FIELDING_E +
##     TEAM_FIELDING_DP
##
##                    Df Sum of Sq    RSS     AIC
## - TEAM_PITCHING_SO  1    0.0104 52.064 -5911.0
## - TEAM_BASERUN_CS   1    0.0125 52.066 -5911.0
## - TEAM_BATTING_SO   1    0.0137 52.067 -5910.9
## <none>                          52.053 -5909.4
## - TEAM_BATTING_2B   1    0.0901 52.143 -5908.4
## - TEAM_BATTING_BB   1    0.1039 52.157 -5908.0
## - TEAM_PITCHING_H   1    0.4160 52.469 -5897.8
## - TEAM_BATTING_3B   1    0.4965 52.550 -5895.2
## - TEAM_PITCHING_HR  1    0.7393 52.792 -5887.4
## - TEAM_BASERUN_SB   1    1.0027 53.056 -5878.9
## - TEAM_FIELDING_DP  1    1.7211 53.774 -5856.0
## - TEAM_FIELDING_E   1    2.1317 54.185 -5843.1
## - TEAM_BATTING_H    1    4.1341 56.187 -5781.3
##
## Step:  AIC=-5911.04
## log(TARGET_WINS + 1) ~ TEAM_BATTING_H + TEAM_BATTING_2B + TEAM_BATTING_3B +
##     TEAM_BATTING_BB + TEAM_BATTING_SO + TEAM_BASERUN_SB + TEAM_BASERUN_CS +
##     TEAM_PITCHING_H + TEAM_PITCHING_HR + TEAM_FIELDING_E + TEAM_FIELDING_DP
##
##                    Df Sum of Sq    RSS     AIC
## - TEAM_BATTING_SO   1    0.0045 52.068 -5912.9
## - TEAM_BASERUN_CS   1    0.0117 52.075 -5912.7
## <none>                          52.064 -5911.0
```

```
## - TEAM_BATTING_2B   1    0.0850 52.149 -5910.3
## - TEAM_BATTING_BB   1    0.1013 52.165 -5909.7
## - TEAM_PITCHING_H   1    0.4113 52.475 -5899.6
## - TEAM_BATTING_3B   1    0.5197 52.583 -5896.1
## - TEAM_PITCHING_HR  1    0.7525 52.816 -5888.6
## - TEAM_BASERUN_SB   1    0.9956 53.059 -5880.8
## - TEAM_FIELDING_DP  1    1.7162 53.780 -5857.8
## - TEAM_FIELDING_E   1    2.2464 54.310 -5841.1
## - TEAM_BATTING_H    1    4.2185 56.282 -5780.4
##
## Step:  AIC=-5912.89
## log(TARGET_WINS + 1) ~ TEAM_BATTING_H + TEAM_BATTING_2B + TEAM_BATTING_3B +
##     TEAM_BATTING_BB + TEAM_BASERUN_SB + TEAM_BASERUN_CS + TEAM_PITCHING_H +
##     TEAM_PITCHING_HR + TEAM_FIELDING_E + TEAM_FIELDING_DP
##
##                     Df Sum of Sq    RSS     AIC
## - TEAM_BASERUN_CS   1    0.0119 52.080 -5914.5
## <none>                          52.068 -5912.9
## - TEAM_BATTING_2B   1    0.1060 52.174 -5911.4
## - TEAM_BATTING_BB   1    0.1130 52.181 -5911.2
## - TEAM_PITCHING_H   1    0.4112 52.479 -5901.5
## - TEAM_BATTING_3B   1    0.5809 52.649 -5896.0
## - TEAM_PITCHING_HR  1    1.0350 53.103 -5881.4
## - TEAM_BASERUN_SB   1    1.0365 53.104 -5881.3
## - TEAM_FIELDING_DP  1    1.7139 53.782 -5859.8
## - TEAM_FIELDING_E   1    2.2461 54.314 -5843.0
## - TEAM_BATTING_H    1    5.9379 58.006 -5731.1
##
## Step:  AIC=-5914.5
## log(TARGET_WINS + 1) ~ TEAM_BATTING_H + TEAM_BATTING_2B + TEAM_BATTING_3B +
##     TEAM_BATTING_BB + TEAM_BASERUN_SB + TEAM_PITCHING_H + TEAM_PITCHING_HR +
##     TEAM_FIELDING_E + TEAM_FIELDING_DP
##
##                     Df Sum of Sq    RSS     AIC
## <none>                          52.080 -5914.5
## - TEAM_BATTING_2B   1    0.1092 52.189 -5912.9
## - TEAM_BATTING_BB   1    0.1217 52.202 -5912.5
## - TEAM_PITCHING_H   1    0.4353 52.515 -5902.3
## - TEAM_BATTING_3B   1    0.5855 52.665 -5897.5
## - TEAM_BASERUN_SB   1    1.0369 53.117 -5882.9
## - TEAM_PITCHING_HR  1    1.1443 53.224 -5879.5
## - TEAM_FIELDING_DP  1    1.7267 53.807 -5861.0
## - TEAM_FIELDING_E   1    2.2913 54.371 -5843.2
## - TEAM_BATTING_H    1    5.9315 58.011 -5732.9
```

Diagnosing our Model Linearity

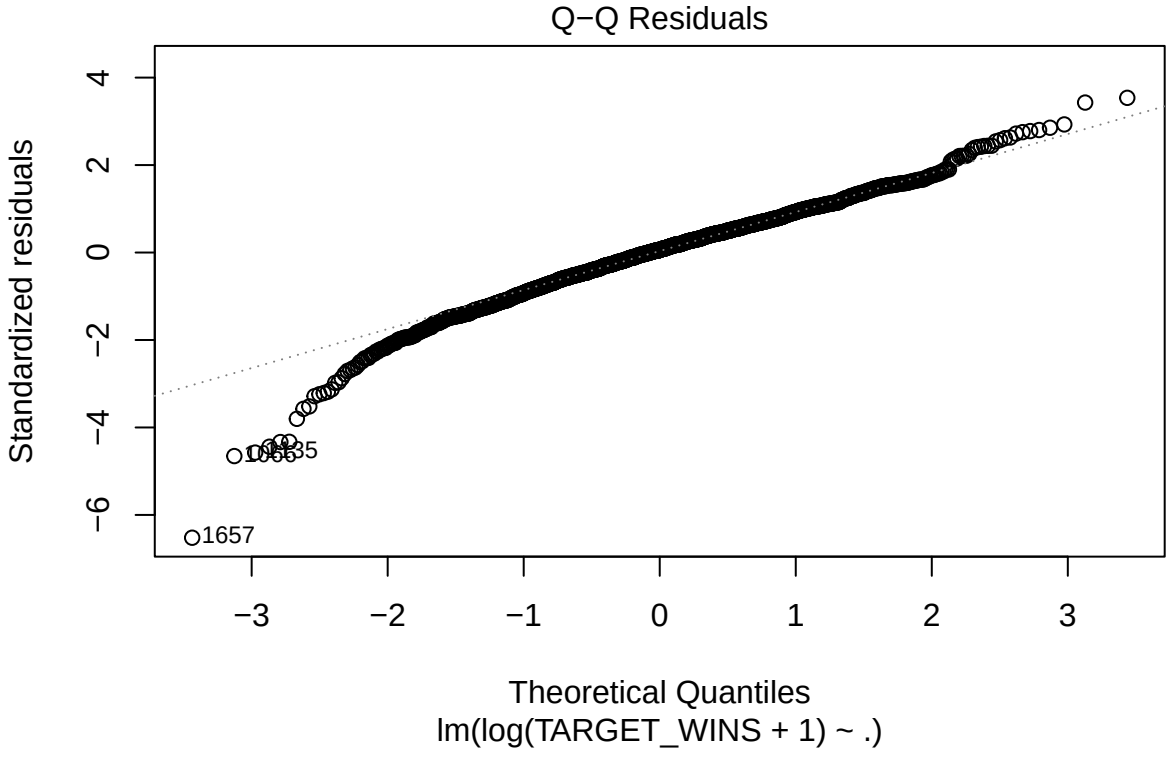
```
plot(stp_logy_model, which=1)
```



Our Residuals vs. Fitted plot suggests a somewhat linear model but there is visible bowing in the trend line.

Normality

```
plot(stp_logy_model, which = 2)
```



```
shapiro.test(residuals(stp_logy_model))
```

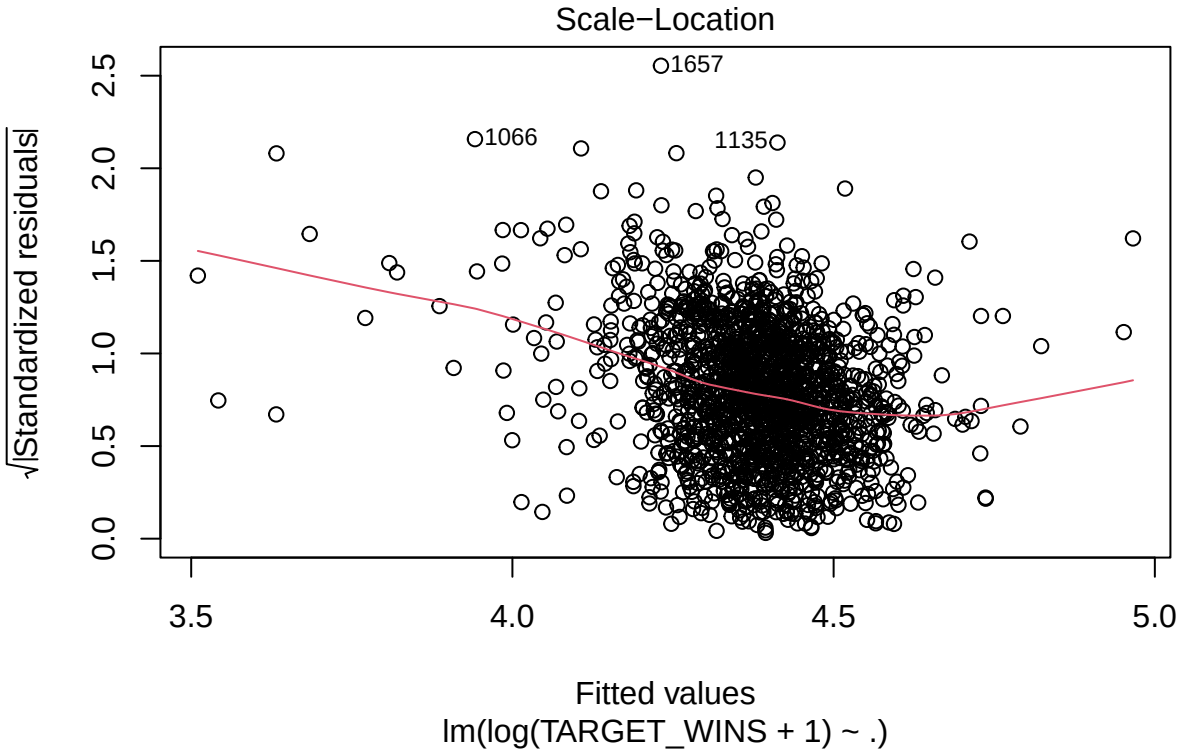
```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(stp_logy_model)
## W = 0.9762, p-value = 3.367e-16
```

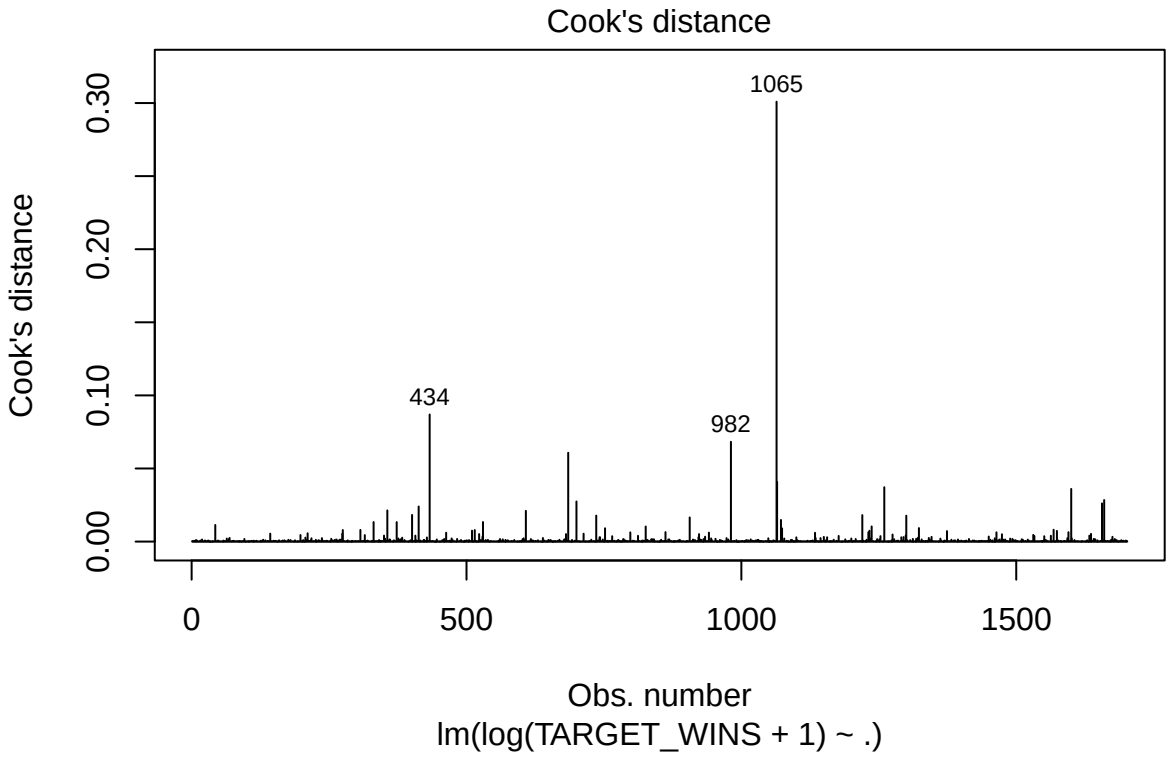
```
# Shapiro-Wilk normality test: look for high p-value
```

Our QQ plot suggests normality thought there is obvious skewing on the tails, particularly on the left.

A Shapiro Wilk's Test statistic had a value of 0.980, also suggesting normality. However, since the p-value (1.416e-14) is less than < 0.05 we may still be violating our normality assumption.

Heteroscedasticity





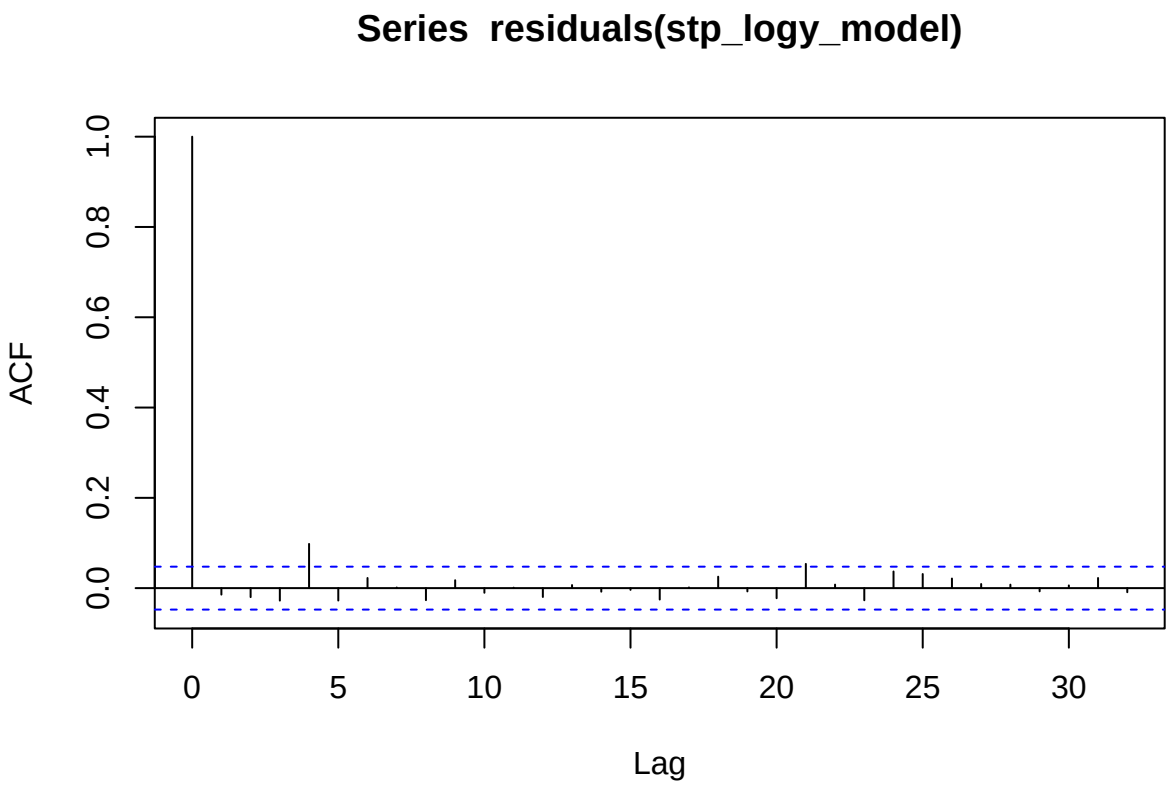

```
##
##  studentized Breusch-Pagan test
##
## data:  stp_logy_model
## BP = 270.46, df = 14, p-value < 2.2e-16
```

Our Scale-Location plot shows clear bowing in the trend line, suggesting that the variance of the residuals is not constant and therefore heteroschedastic. While there is no obvious fan/wedge pattern, there is clustering in the center suggesting underfitting, high leverage outliers or that additional transformation may be needed. As our Cook's Distance plot has no values with a greater than 1, we can rule our the effects of high leverage points. However, it should be pointed the leverage points appear to be more influential than in our backwards selected model or our Box-Cox transformed model.

The Breusch-Page test statistic BP (383.07) indicates a substantial relationship between the residual variance and the predictors and the small p-value (2.2e-16) suggest evidence of heteroscedasticity. Though the values are different that we may need to transform the data to meet the Assumption of Homoscedasticity.

Independence

```
acf(residuals(stp_logy_model))
```




```
durbinWatsonTest(stp_logy_model)
```

```
##  lag Autocorrelation D-W Statistic p-value
##    1     -0.01440497      2.028171   0.582
##  Alternative hypothesis: rho != 0
```

```
# Durbin Watson should be close to 2
```

Our Autocorrelation Function shows that there are lags above the blue dashed line, suggesting no autocorrelation. This is confirmed through a Durbin-Watson test statistic value of 2.00 and an autocorrelation value of -0.0041. Furthermore, as our p-value (0.96) is greater than 0.05, we do not have enough evidence to reject the null hypothesis that there is no autocorrelation. In other words, the test results suggest that our model's residuals are independent and therefore do not violate the Independence Assumption.

Conclusion Applying a log transformation the dependent variable in our model appears to inferior to the backward selected model and the Box-Cox transformed model. We should therefore avoid using this model.

Model T4: Log Transformation on Independent Variables (X) Some of our predictors (TEAM_PITCHING_SO, TEAM_PITCHING_BB, & TEAM_PITCHING_H) showed the presence of outliers. We will transform these variables with a log transformation.

Backwards Selection We will use backward step-wise selection to build our model.

```
stp_logx_model_full <- lm(TARGET_WINS ~ . - TEAM_BATTING_H - TEAM_BATTING_HR - TEAM_PITCHING_SO - TEAM_

# Backward step-wise regression
stp_logx_model <- step(stp_logx_model_full, direction = "backward")
```

```
## Start:  AIC=8743.2
## TARGET_WINS ~ (TEAM_BATTING_H + TEAM_BATTING_2B + TEAM_BATTING_3B +
##     TEAM_BATTING_HR + TEAM_BATTING_BB + TEAM_BATTING_SO + TEAM_BASERUN_SB +
```

```
##     TEAM_BASERUN_CS + TEAM_PITCHING_H + TEAM_PITCHING_HR + TEAM_PITCHING_BB +
##     TEAM_PITCHING_SO + TEAM_FIELDING_E + TEAM_FIELDING_DP + log_TEAM_BATTING_H +
##     log_TEAM_BATTING_HR + log_TEAM_PITCHING_SO + log_TEAM_PITCHING_BB) -
##     TEAM_BATTING_H - TEAM_BATTING_HR - TEAM_PITCHING_SO - TEAM_PITCHING_BB
##
##                         Df Sum of Sq    RSS    AIC
## - TEAM_PITCHING_H        1      26.4 284658 8741.4
## - TEAM_BASERUN_CS        1     170.2 284801 8742.2
## <none>                               284631 8743.2
## - TEAM_BATTING_2B        1     396.4 285028 8743.6
## - log_TEAM_BATTING_HR    1    1218.3 285849 8748.5
## - TEAM_BATTING_3B        1    3580.3 288211 8762.5
## - TEAM_BATTING_SO        1    3726.0 288357 8763.3
## - log_TEAM_PITCHING_SO   1    4182.4 288814 8766.0
## - log_TEAM_PITCHING_BB   1    4656.5 289288 8768.8
## - TEAM_PITCHING_HR       1    4673.5 289305 8768.9
## - TEAM_BATTING_BB        1    5801.7 290433 8775.5
## - TEAM_FIELDING_DP       1    6585.0 291216 8780.1
## - TEAM_BASERUN_SB        1    6984.5 291616 8782.5
## - TEAM_FIELDING_E        1   11106.8 295738 8806.4
## - log_TEAM_BATTING_H     1   21311.0 305942 8864.1
##
## Step:  AIC=8741.36
## TARGET_WINS ~ TEAM_BATTING_2B + TEAM_BATTING_3B + TEAM_BATTING_BB +
##     TEAM_BATTING_SO + TEAM_BASERUN_SB + TEAM_BASERUN_CS + TEAM_PITCHING_HR +
##     TEAM_FIELDING_E + TEAM_FIELDING_DP + log_TEAM_BATTING_H +
##     log_TEAM_BATTING_HR + log_TEAM_PITCHING_SO + log_TEAM_PITCHING_BB
##
##                         Df Sum of Sq    RSS    AIC
## - TEAM_BASERUN_CS        1     186.1 284844 8740.5
## <none>                               284658 8741.4
## - TEAM_BATTING_2B        1     388.8 285046 8741.7
## - log_TEAM_BATTING_HR    1    1196.6 285854 8746.5
## - TEAM_BATTING_SO        1    3761.9 288420 8761.7
## - TEAM_BATTING_3B        1    3893.4 288551 8762.5
## - log_TEAM_PITCHING_SO   1    4354.1 289012 8765.2
## - TEAM_PITCHING_HR       1    4754.0 289412 8767.5
## - TEAM_FIELDING_DP       1    6560.5 291218 8778.1
## - log_TEAM_PITCHING_BB   1    6889.6 291547 8780.1
## - TEAM_BASERUN_SB        1    7323.2 291981 8782.6
## - TEAM_BATTING_BB        1    8126.4 292784 8787.3
## - TEAM_FIELDING_E        1   13833.0 298491 8820.1
## - log_TEAM_BATTING_H     1   21625.3 306283 8864.0
##
## Step:  AIC=8740.47
## TARGET_WINS ~ TEAM_BATTING_2B + TEAM_BATTING_3B + TEAM_BATTING_BB +
##     TEAM_BATTING_SO + TEAM_BASERUN_SB + TEAM_PITCHING_HR + TEAM_FIELDING_E +
##     TEAM_FIELDING_DP + log_TEAM_BATTING_H + log_TEAM_BATTING_HR +
##     log_TEAM_PITCHING_SO + log_TEAM_PITCHING_BB
##
##                         Df Sum of Sq    RSS    AIC
## <none>                               284844 8740.5
## - TEAM_BATTING_2B        1     406.4 285250 8740.9
## - log_TEAM_BATTING_HR    1    1192.6 286036 8745.6
```

```
## - TEAM_BATTING_SO       1    3760.6 288604 8760.8
## - TEAM_BATTING_3B       1    3975.0 288819 8762.1
## - log_TEAM_PITCHING_SO  1    4354.3 289198 8764.3
## - TEAM_PITCHING_HR      1    5012.9 289857 8768.2
## - TEAM_FIELDING_DP      1    6635.4 291479 8777.7
## - log_TEAM_PITCHING_BB  1    7046.7 291890 8780.1
## - TEAM_BASERUN_SB       1    7142.4 291986 8780.6
## - TEAM_BATTING_BB       1    8481.9 293326 8788.4
## - TEAM_FIELDING_E       1   13710.7 298554 8818.5
## - log_TEAM_BATTING_H    1   21510.4 306354 8862.4
```

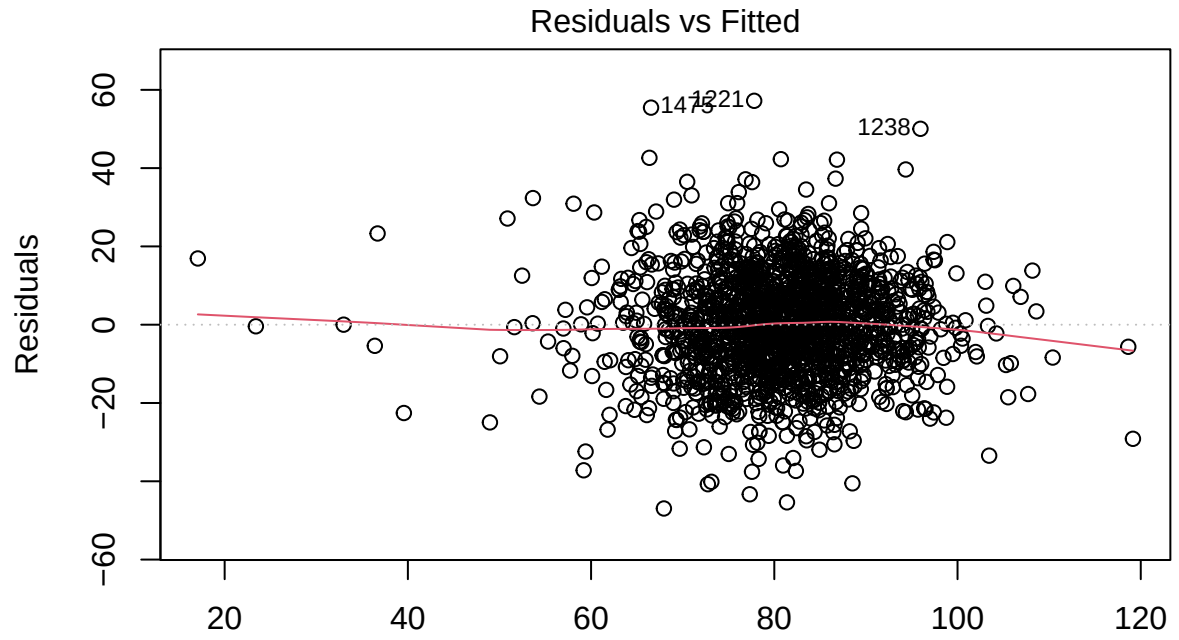
Looking at the summary for this model shows that all predictors have high statistic significance.

```
summary(stp_logx_model)
```

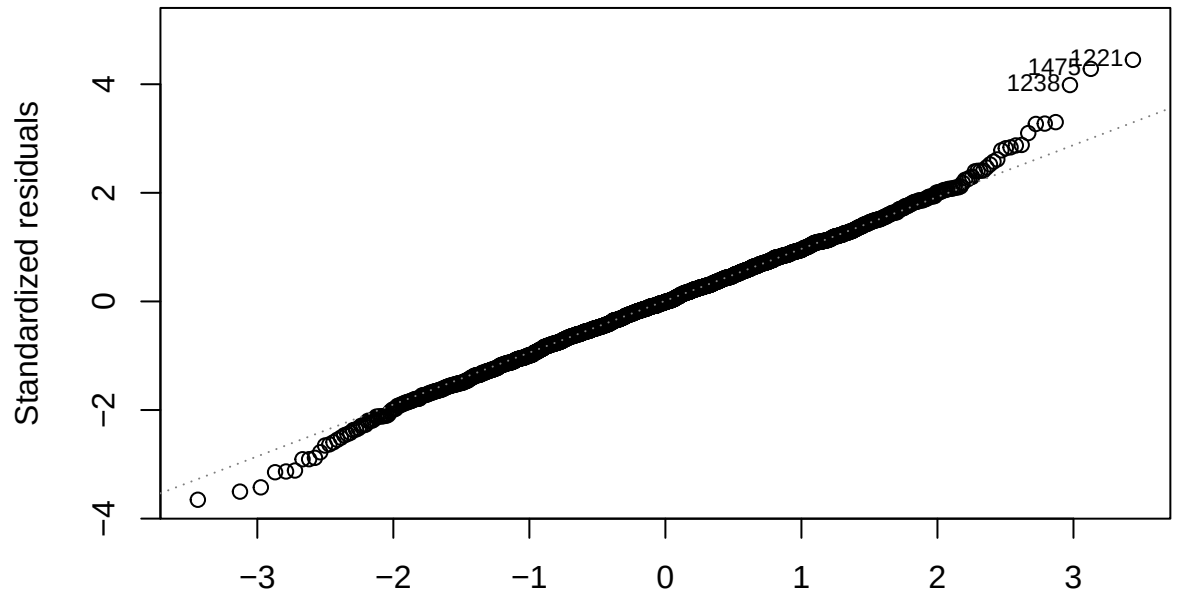
```
##
## Call:
## lm(formula = TARGET_WINS ~ TEAM_BATTING_2B + TEAM_BATTING_3B +
##     TEAM_BATTING_BB + TEAM_BATTING_SO + TEAM_BASERUN_SB + TEAM_PITCHING_HR +
##     TEAM_FIELDING_E + TEAM_FIELDING_DP + log_TEAM_BATTING_H +
##     log_TEAM_BATTING_HR + log_TEAM_PITCHING_SO + log_TEAM_PITCHING_BB,
##     data = train_df_log)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -46.947  -8.180  -0.139   8.503  57.195
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)          -4.665e+02  5.633e+01  -8.281 2.45e-16 ***
## TEAM_BATTING_2B      -1.678e-02  1.081e-02  -1.552  0.12075
## TEAM_BATTING_3B       9.285e-02  1.913e-02   4.855 1.32e-06 ***
## TEAM_BATTING_BB       5.165e-02  7.283e-03   7.092 1.94e-12 ***
## TEAM_BATTING_SO      -2.377e-02  5.033e-03  -4.722 2.53e-06 ***
## TEAM_BASERUN_SB       3.123e-02  4.799e-03   6.508 1.00e-10 ***
## TEAM_PITCHING_HR      8.489e-02  1.557e-02   5.452 5.72e-08 ***
## TEAM_FIELDING_E      -2.582e-02  2.863e-03  -9.017  < 2e-16 ***
## TEAM_FIELDING_DP     -9.686e-02  1.544e-02  -6.273 4.50e-10 ***
## log_TEAM_BATTING_H    7.897e+01  6.992e+00  11.294  < 2e-16 ***
## log_TEAM_BATTING_HR  -3.269e+00  1.229e+00  -2.659  0.00791 **
## log_TEAM_PITCHING_SO  1.634e+01  3.216e+00   5.081 4.17e-07 ***
## log_TEAM_PITCHING_BB -1.984e+01  3.070e+00  -6.464 1.33e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12.99 on 1689 degrees of freedom
## Multiple R-squared:  0.3206, Adjusted R-squared:  0.3158
## F-statistic: 66.43 on 12 and 1689 DF,  p-value: < 2.2e-16
```

The Residuals vs. Fitted and QQ Plots show a fairly linear pattern, while Scale-Location plot suggest Homoscedasticity. The Residuals vs Leverage plot reveals that our outliers have less leverage than in the pre-log transformed model.

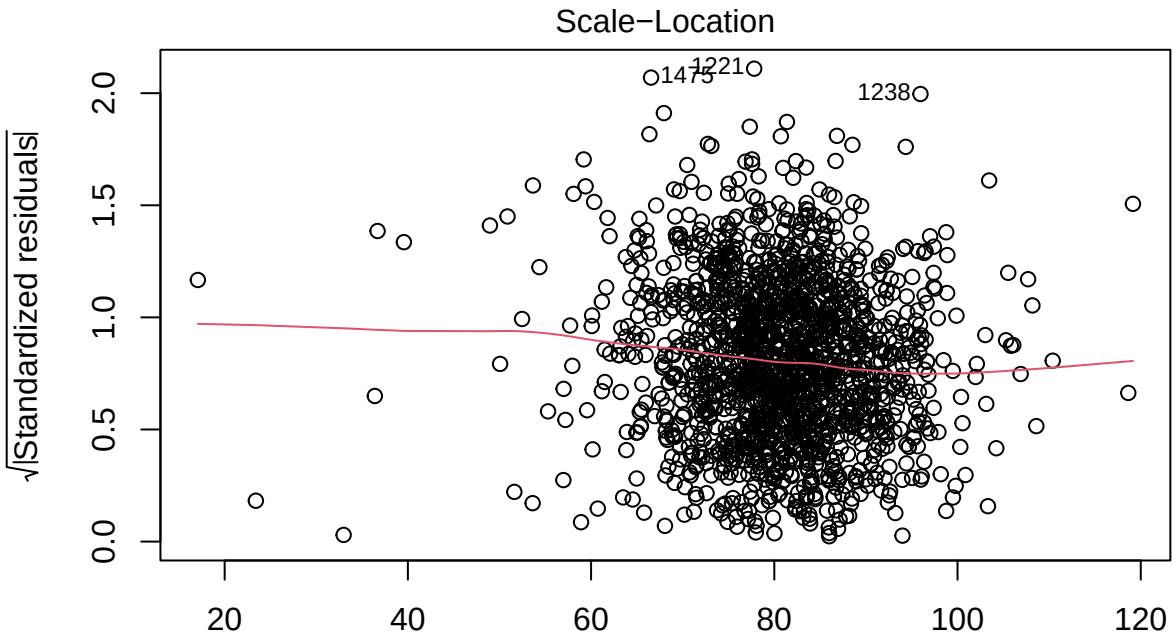
```
plot(stp_logx_model)
```

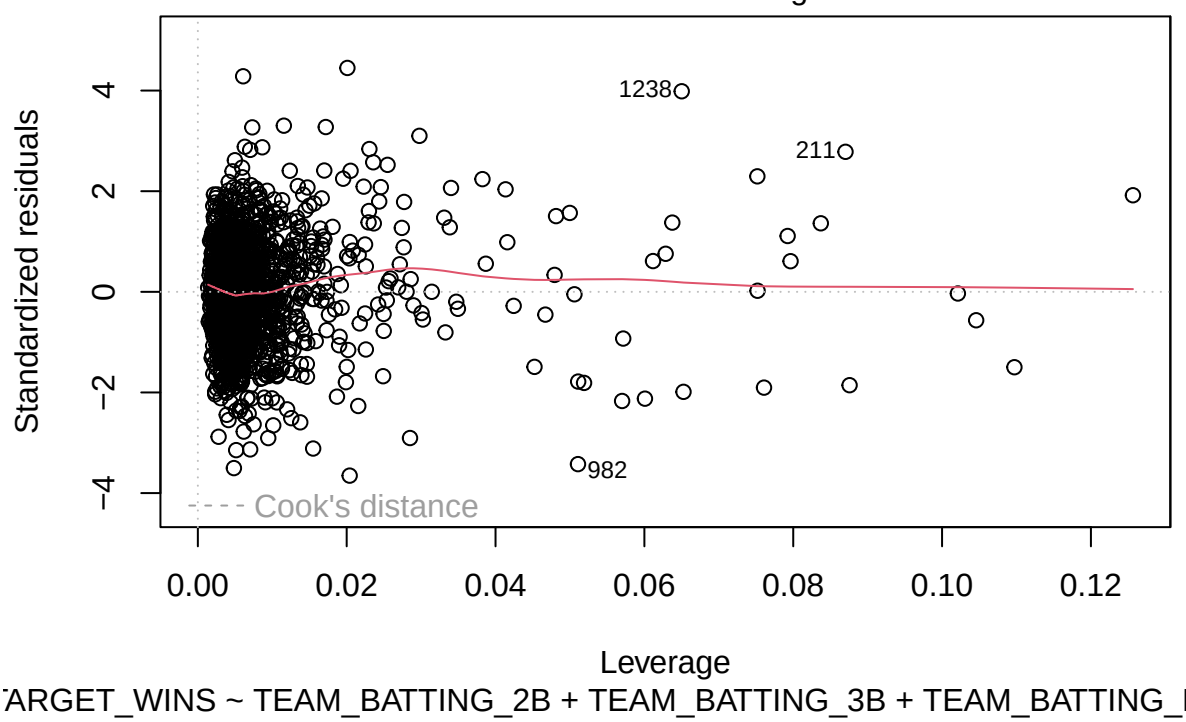


'ARGET_WINS ~ TEAM_BATTING_2B + TEAM_BATTING_3B + TEAM_BATTING_BB +



'ARGET_WINS ~ TEAM_BATTING_2B + TEAM_BATTING_3B + TEAM_BATTING_BB +





Multicolinearity A VIF Test shows several variables that are highly correlated, including TEAM_PITCHING_H, TEAM_BATTING_BB, TEAM_PITCHING_SO, TEAM_PITCHING_BB, TEAM_FIELDING_E & TEAM_BATTING_H. We may choose to leave these out.

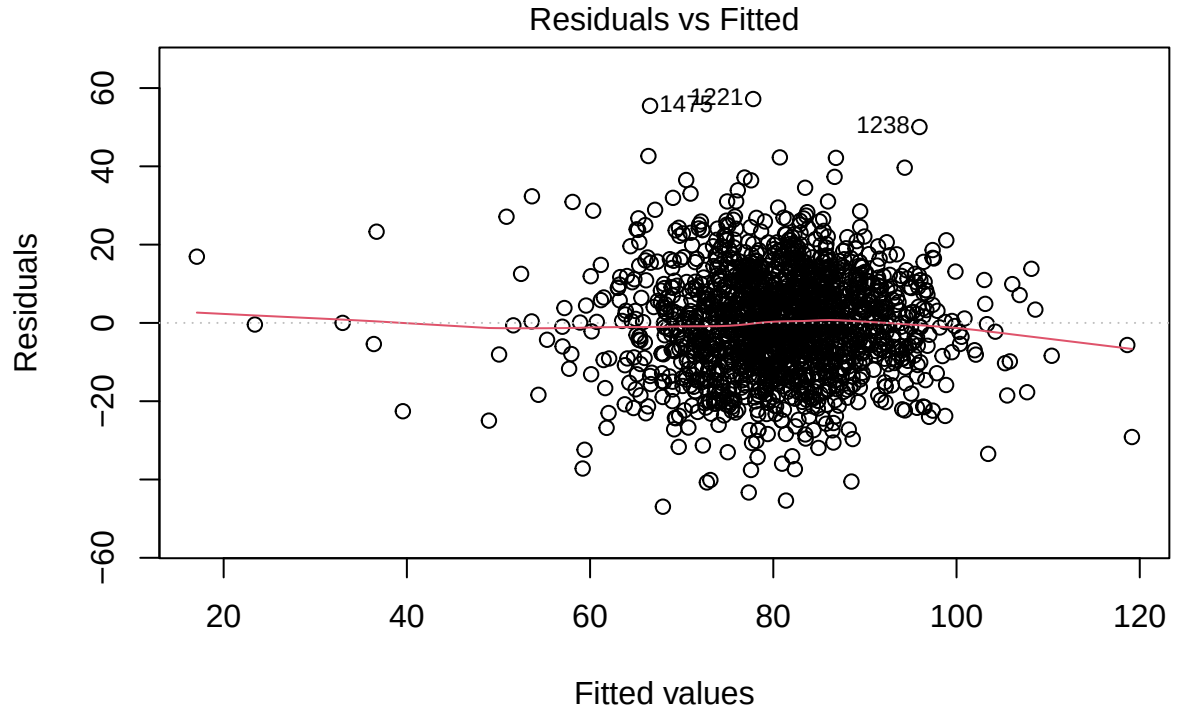
```
vif(stp_logx_model)
```

```
##      TEAM_BATTING_2B      TEAM_BATTING_3B      TEAM_BATTING_BB
```

```
##             2.628155             2.803296             7.877815
##      TEAM_BATTING_SO      TEAM_BASERUN_SB     TEAM_PITCHING_HR
##            14.385392             1.735966             9.361936
##      TEAM_FIELDING_E     TEAM_FIELDING_DP   log_TEAM_BATTING_H
##             4.073234             1.464900             4.298651
##  log_TEAM_BATTING_HR log_TEAM_PITCHING_SO log_TEAM_PITCHING_BB
##            10.786426            10.955741             5.533907
```

Diagnosing our Model Linearity

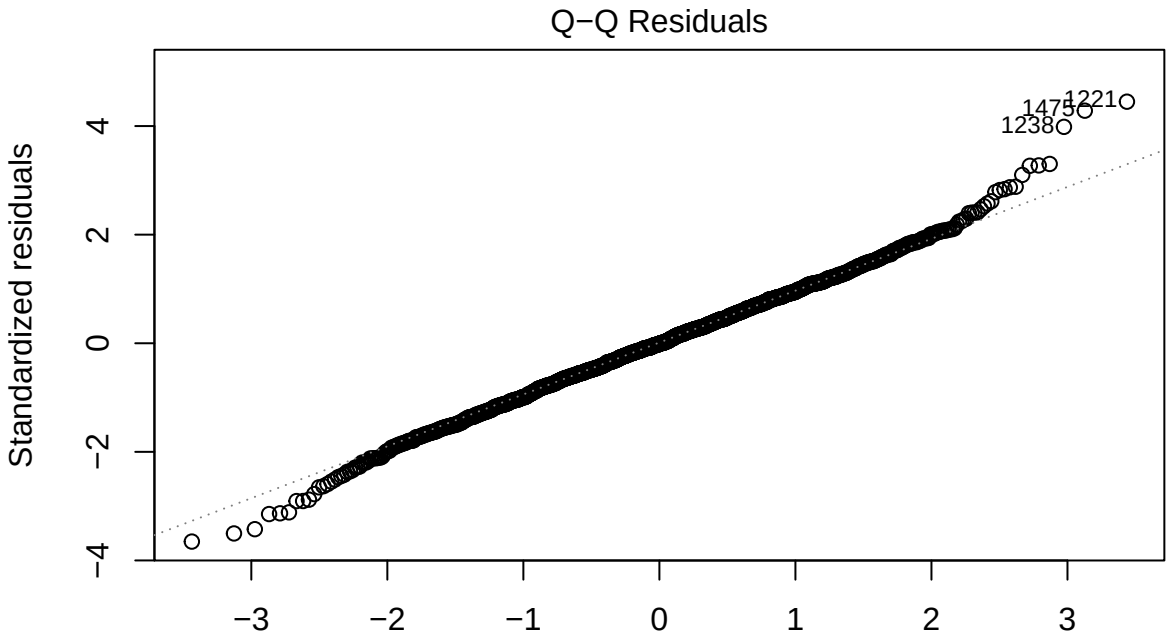
```
plot(stp_logx_model, which=1)
```



Our Residuals vs. Fitted plot suggest a fairly linear model.

Normality

```
plot(stp_logx_model, which = 2)
```



TARGET_WINS ~ TEAM_BATTING_2B + TEAM_BATTING_3B + TEAM_BATTING_BB +

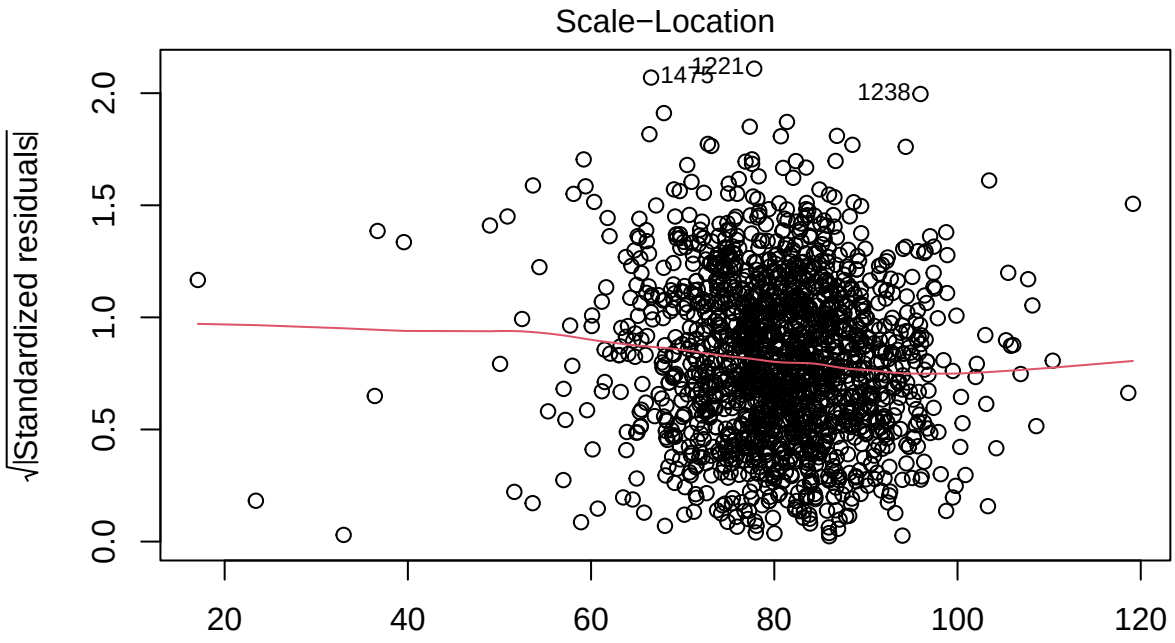
```
# Shapiro-Wilk normality test: look for high p-value
shapiro.test(residuals(stp_logx_model))
```

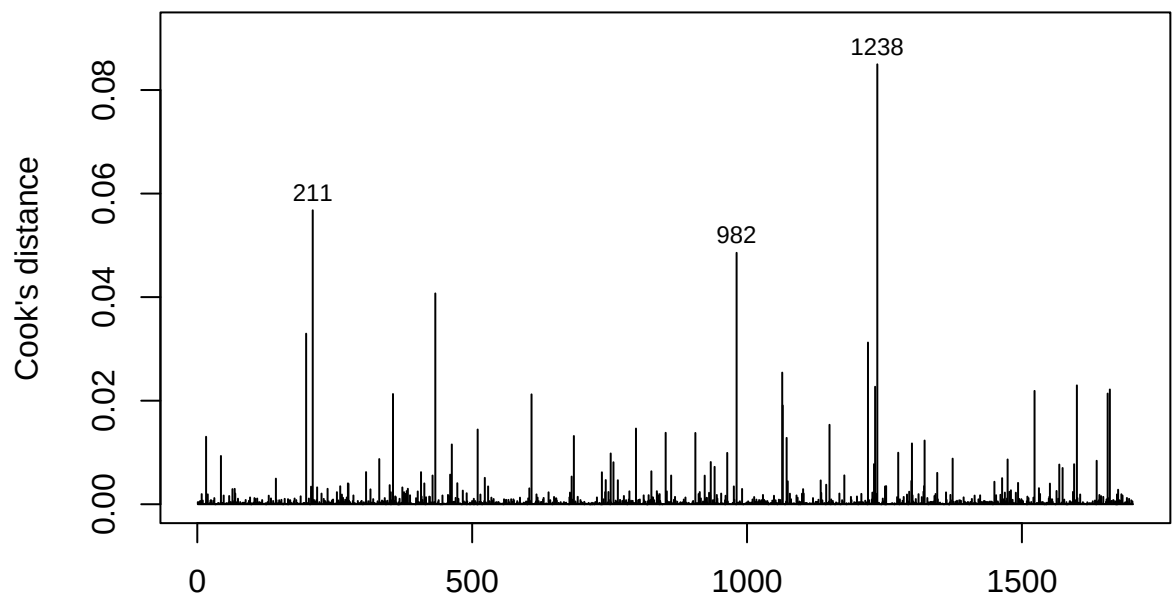
```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(stp_logx_model)
## W = 0.9961, p-value = 0.000232
```

Our QQ plot suggests normality thought there is some skewing on the tails, particularly on the right. The skewing appears to be less pitched than in our backward-selected model or Box-Cox Transformed model.

A Shapiro Wilk's Test statistic had a value of 0.9943, also suggesting normality. However, since the p-value (4.273e-06) is less than < 0.05 we may still be violating our normality assumption.

Heteroscedasticity






```
##
##  studentized Breusch-Pagan test
##
## data:  stp_logx_model
## BP = 233.06, df = 12, p-value < 2.2e-16
```

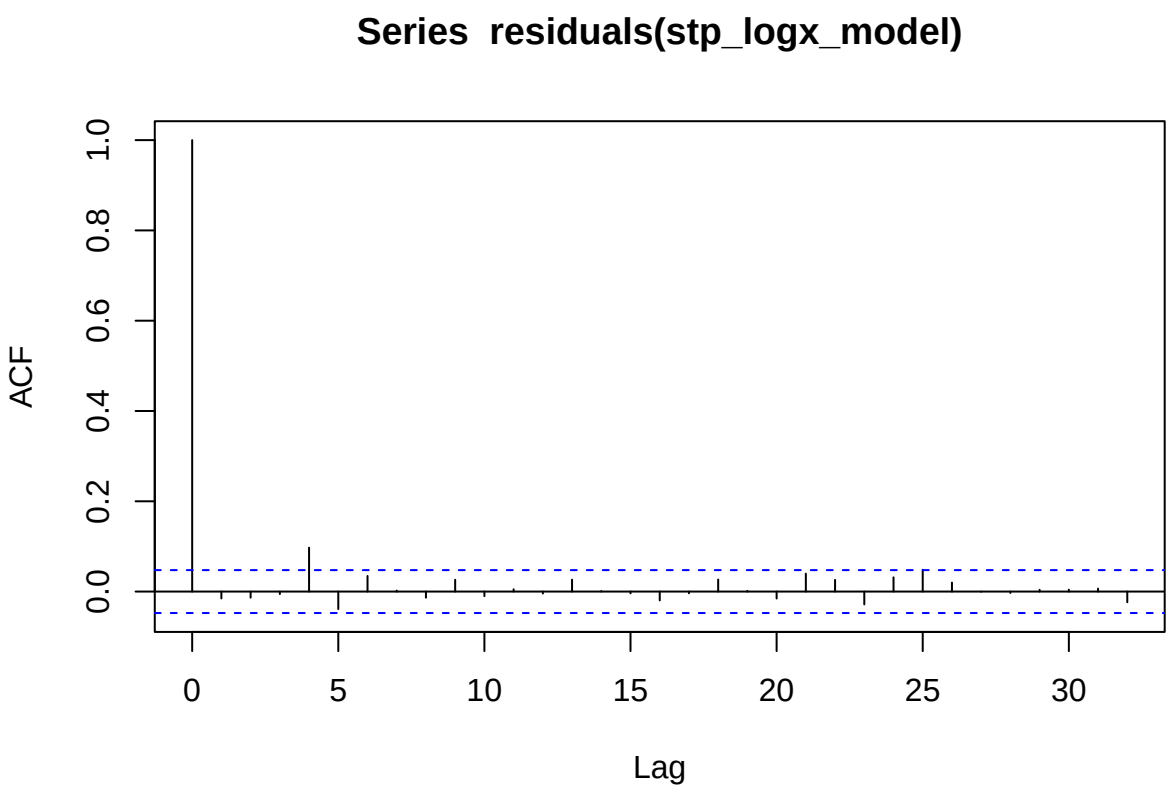
Our Scale-Location plot shows that points appear somewhat evenly distributed above and below the trend

line but there is clear bowing in the trend line, suggesting that the variance of the residuals is not constant and therefore heteroschedastic. While there is no obvious fan/wedge pattern, there is clustering in the center suggesting underfitting, high leverage outliers or that additional transformation may be needed. As our Cook's Distance plot has no values with a greater than 1, we can rule our the effects of high leverage points.

The Breusch-Page test statistic BP (302.21) and the small p-value (2.2e-16) suggest evidence of heteroscedasticity. Though the values are different that we may need to transform the data to meet the Assumption of Homoscedasticity.

Independence

```
acf(residuals(stp_logx_model))
```




```
durbinWatsonTest(stp_logx_model)
```

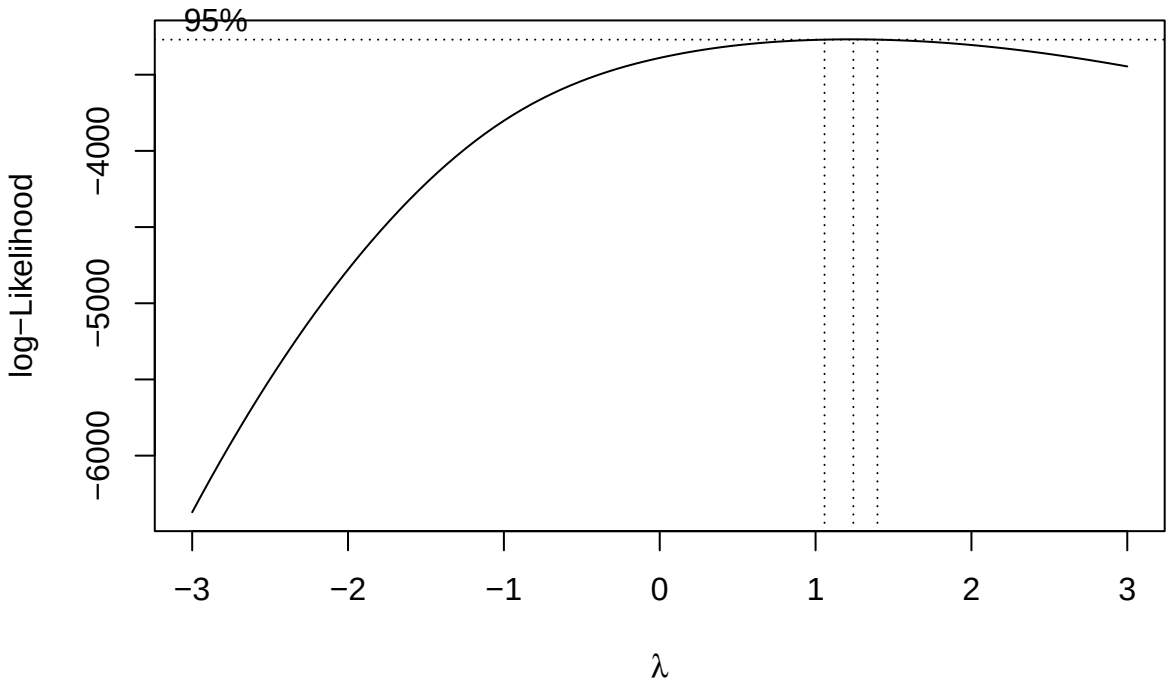
```
##  lag Autocorrelation D-W Statistic p-value
##    1     -0.01529779      2.029836   0.574
##  Alternative hypothesis: rho != 0
```

```
# Durbin Watson should be close to 2
```

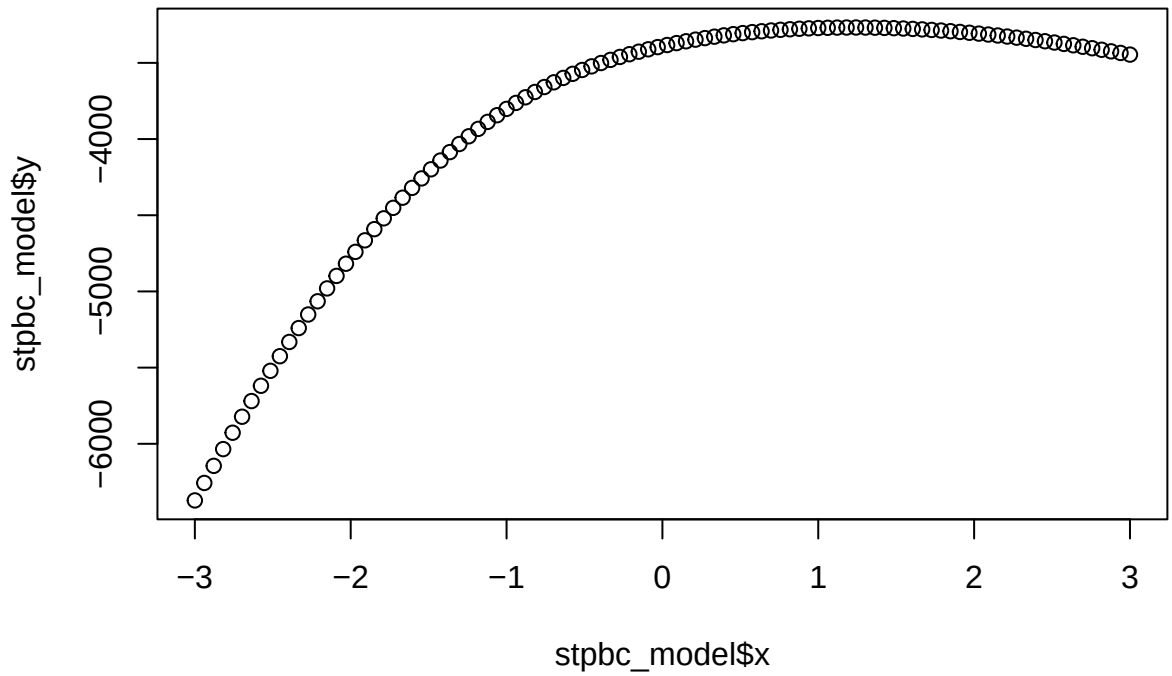
Our Autocorrelation Function shows that there are lags above the blue dashed line, suggesting no autocorrelation. This is confirmed through a Durbin-Watson test statistic value of 2.011 and an autocorrelation value of -0.008. Furthermore, as our p-value (0.822) is greater than 0.05, we do not have enough evidence to reject the null hypothesis that there is no autocorrelation. In other words, the test results suggest that our model's residuals are independent and therefore do not violate the Independence Assumption.

Model T5: Box-Cox Transformation on Dependent Variable In this section, we apply a Box-Cox Transformation to our backward selected mode and data.

```
stpbc_model <- boxcox(back_select_model, lambda = seq(-3,3))
```




```
plot(stpbc_model)
```




```
best_lambda <- stpbc_model$x[which(stpbc_model$y==max(stpbc_model$y))]

stp_model_inv <- lm((TARGET_WINS)^best_lambda ~ TEAM_BATTING_H + TEAM_BATTING_2B + TEAM_BATTING_3B + TE

summary(stp_model_inv)
```

```
##
## Call:
## lm(formula = (TARGET_WINS)^best_lambda ~ TEAM_BATTING_H + TEAM_BATTING_2B +
```

```
##     TEAM_BATTING_3B + TEAM_BATTING_SO + TEAM_BASERUN_SB + TEAM_PITCHING_HR +
##     TEAM_PITCHING_BB + TEAM_PITCHING_SO + TEAM_FIELDING_E + TEAM_FIELDING_DP,
##     data = stp75_train_df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -160.002  -31.694   -1.753   30.262  218.369
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      30.439025  22.090363   1.378    0.168
## TEAM_BATTING_H    0.168744   0.015834  10.657  < 2e-16 ***
## TEAM_BATTING_2B  -0.049023   0.038413  -1.276    0.202
## TEAM_BATTING_3B   0.280560   0.068908   4.072 4.89e-05 ***
## TEAM_BATTING_SO  -0.005103   0.013188  -0.387    0.699
## TEAM_BASERUN_SB   0.107337   0.017380   6.176 8.22e-10 ***
## TEAM_PITCHING_HR  0.182375   0.036190   5.039 5.17e-07 ***
## TEAM_PITCHING_BB  0.003137   0.009882   0.317    0.751
## TEAM_PITCHING_SO -0.002248   0.007738  -0.291    0.771
## TEAM_FIELDING_E  -0.086413   0.008583 -10.068  < 2e-16 ***
## TEAM_FIELDING_DP -0.371205   0.052845  -7.024 3.10e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 47.18 on 1691 degrees of freedom
## Multiple R-squared:  0.2928, Adjusted R-squared:  0.2886
## F-statistic: 70.02 on 10 and 1691 DF,  p-value: < 2.2e-16
```

Looking at the summary of our new Box-Cox transformed model shows three variables are not statistically significant. We should therefore remove them one at a time.

```
stp_model_inv <- update(stp_model_inv, . ~ . - TEAM_BATTING_SO)
summary(stp_model_inv)
```

```
##
## Call:
## lm(formula = (TARGET_WINS)^best_lambda ~ TEAM_BATTING_H + TEAM_BATTING_2B +
##     TEAM_BATTING_3B + TEAM_BASERUN_SB + TEAM_PITCHING_HR + TEAM_PITCHING_BB +
##     TEAM_PITCHING_SO + TEAM_FIELDING_E + TEAM_FIELDING_DP, data = stp75_train_df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -159.912  -31.656   -1.876   30.221  217.584
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      25.274155  17.597392   1.436    0.151
## TEAM_BATTING_H    0.170846   0.014870  11.490  < 2e-16 ***
## TEAM_BATTING_2B  -0.051759   0.037748  -1.371    0.170
## TEAM_BATTING_3B   0.285658   0.067620   4.224 2.52e-05 ***
## TEAM_BASERUN_SB   0.104773   0.016063   6.523 9.10e-11 ***
## TEAM_PITCHING_HR  0.175985   0.032194   5.466 5.28e-08 ***
## TEAM_PITCHING_BB  0.004891   0.008778   0.557    0.577
## TEAM_PITCHING_SO -0.004313   0.005602  -0.770    0.441
## TEAM_FIELDING_E  -0.084844   0.007563 -11.218  < 2e-16 ***
```

```
## TEAM_FIELDING_DP -0.371106   0.052831  -7.024 3.10e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 47.17 on 1692 degrees of freedom
## Multiple R-squared:  0.2928, Adjusted R-squared:  0.289
## F-statistic: 77.82 on 9 and 1692 DF,  p-value: < 2.2e-16
```

```
AIC(back_select_model)
```

```
## [1] 13605.45
```

```
stp_model_inv <- update(stp_model_inv, . ~ . - TEAM_PITCHING_BB)
summary(stp_model_inv)
```

```
##
## Call:
## lm(formula = (TARGET_WINS)^best_lambda ~ TEAM_BATTING_H + TEAM_BATTING_2B +
##     TEAM_BATTING_3B + TEAM_BASERUN_SB + TEAM_PITCHING_HR + TEAM_PITCHING_SO +
##     TEAM_FIELDING_E + TEAM_FIELDING_DP, data = stp75_train_df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -159.945  -31.724   -1.658   30.724  216.857
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      25.545291  17.587082   1.453    0.147
## TEAM_BATTING_H    0.171217   0.014852  11.528  < 2e-16 ***
## TEAM_BATTING_2B  -0.052598   0.037710  -1.395    0.163
## TEAM_BATTING_3B   0.290885   0.066952   4.345 1.48e-05 ***
## TEAM_BASERUN_SB   0.106926   0.015588   6.860 9.65e-12 ***
## TEAM_PITCHING_HR  0.180261   0.031260   5.767 9.60e-09 ***
## TEAM_PITCHING_SO -0.003855   0.005540  -0.696    0.487
## TEAM_FIELDING_E  -0.085052   0.007553 -11.261  < 2e-16 ***
## TEAM_FIELDING_DP -0.365976   0.052012  -7.036 2.85e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 47.16 on 1693 degrees of freedom
## Multiple R-squared:  0.2926, Adjusted R-squared:  0.2893
## F-statistic: 87.54 on 8 and 1693 DF,  p-value: < 2.2e-16
```

```
AIC(back_select_model)
```

```
## [1] 13605.45
```

```
stp_model_inv <- update(stp_model_inv, . ~ . - TEAM_PITCHING_SO)
summary(stp_model_inv)
```

```
##
## Call:
## lm(formula = (TARGET_WINS)^best_lambda ~ TEAM_BATTING_H + TEAM_BATTING_2B +
##     TEAM_BATTING_3B + TEAM_BASERUN_SB + TEAM_PITCHING_HR + TEAM_FIELDING_E +
##     TEAM_FIELDING_DP, data = stp75_train_df)
##
## Residuals:
```

```
##      Min       1Q   Median       3Q      Max
## -159.684  -31.752   -1.647   30.624  215.869
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      17.875917  13.703459   1.304    0.192
## TEAM_BATTING_H    0.176195   0.013013  13.540  < 2e-16 ***
## TEAM_BATTING_2B  -0.059243   0.036475  -1.624    0.105
## TEAM_BATTING_3B   0.295325   0.066638   4.432 9.94e-06 ***
## TEAM_BASERUN_SB   0.105857   0.015510   6.825 1.22e-11 ***
## TEAM_PITCHING_HR  0.169940   0.027513   6.177 8.18e-10 ***
## TEAM_FIELDING_E  -0.087586   0.006616 -13.239  < 2e-16 ***
## TEAM_FIELDING_DP -0.363023   0.051830  -7.004 3.57e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 47.15 on 1694 degrees of freedom
## Multiple R-squared:  0.2924, Adjusted R-squared:  0.2895
## F-statistic:   100 on 7 and 1694 DF,  p-value: < 2.2e-16
```

```
AIC(back_select_model)
```

```
## [1] 13605.45
```

```
stp_model_inv <- update(stp_model_inv, . ~ . - TEAM_BATTING_2B)
summary(stp_model_inv)
```

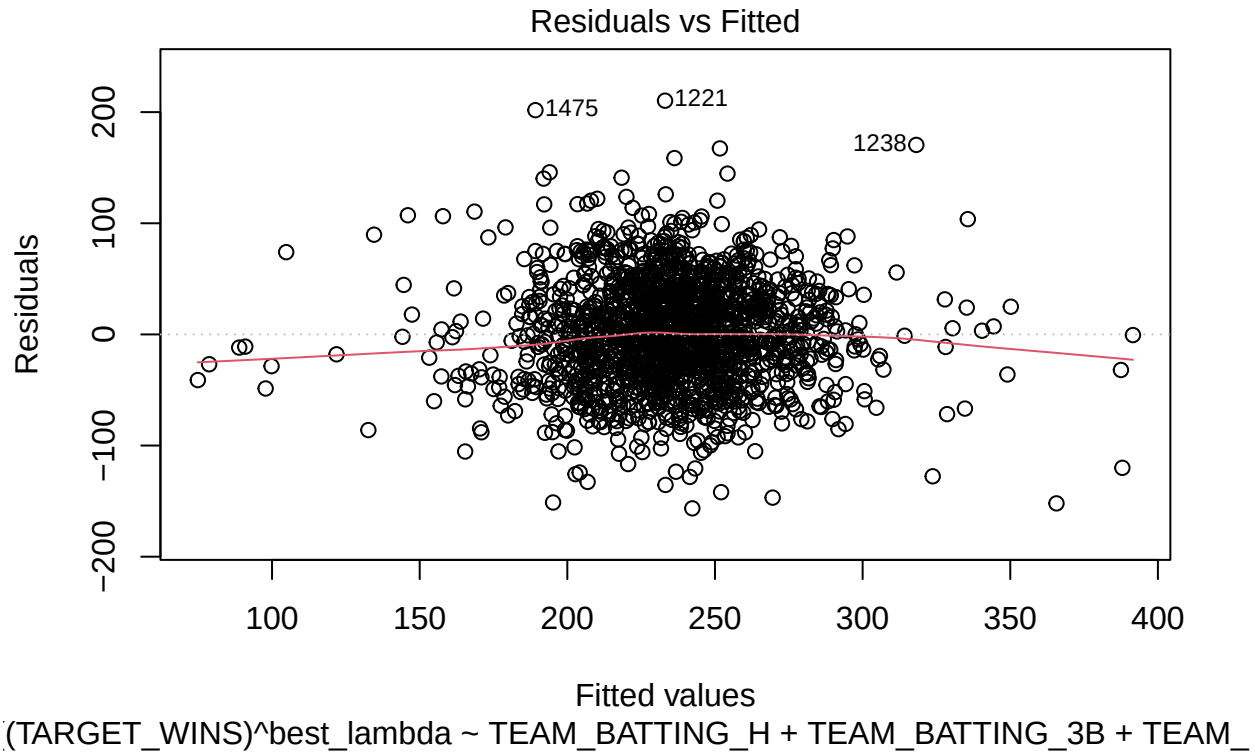
```
##
## Call:
## lm(formula = (TARGET_WINS)^best_lambda ~ TEAM_BATTING_H + TEAM_BATTING_3B +
##     TEAM_BASERUN_SB + TEAM_PITCHING_HR + TEAM_FIELDING_E + TEAM_FIELDING_DP,
##     data = stp75_train_df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -156.515  -32.052   -1.391   30.475  210.291
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      22.357240  13.429300   1.665   0.0961 .
## TEAM_BATTING_H    0.162907   0.010124  16.091  < 2e-16 ***
## TEAM_BATTING_3B   0.308245   0.066193   4.657 3.46e-06 ***
## TEAM_BASERUN_SB   0.107080   0.015499   6.909 6.89e-12 ***
## TEAM_PITCHING_HR  0.161572   0.027039   5.975 2.79e-09 ***
## TEAM_FIELDING_E  -0.084195   0.006281 -13.405  < 2e-16 ***
## TEAM_FIELDING_DP -0.363575   0.051854  -7.011 3.39e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 47.18 on 1695 degrees of freedom
## Multiple R-squared:  0.2913, Adjusted R-squared:  0.2888
## F-statistic: 116.1 on 6 and 1695 DF,  p-value: < 2.2e-16
```

```
AIC(back_select_model)
```

```
## [1] 13605.45
```

Diagnosing our Model Linearity

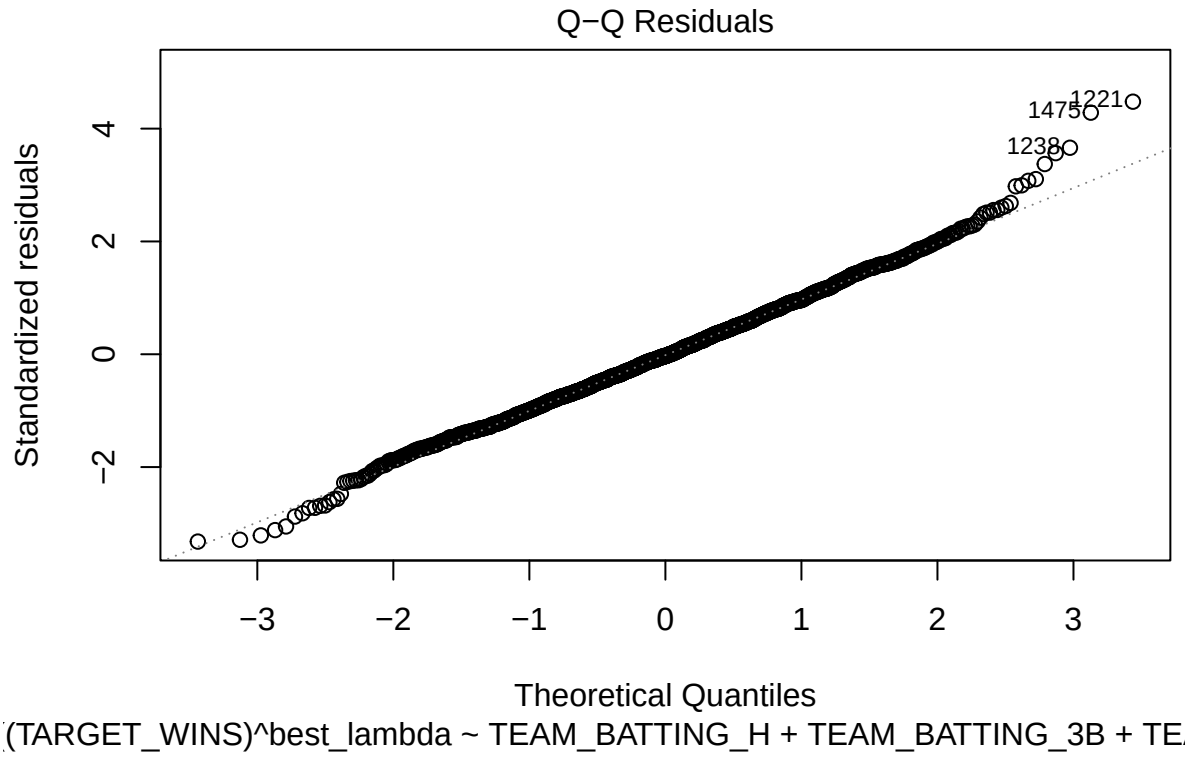
```
plot(stp_model_inv, which=1)
```



Our Residuals vs. Fitted plot suggest a fairly linear model.

Normality

```
plot(stp_model_inv, which = 2)
```




```
shapiro.test(residuals(stp_model_inv))
```

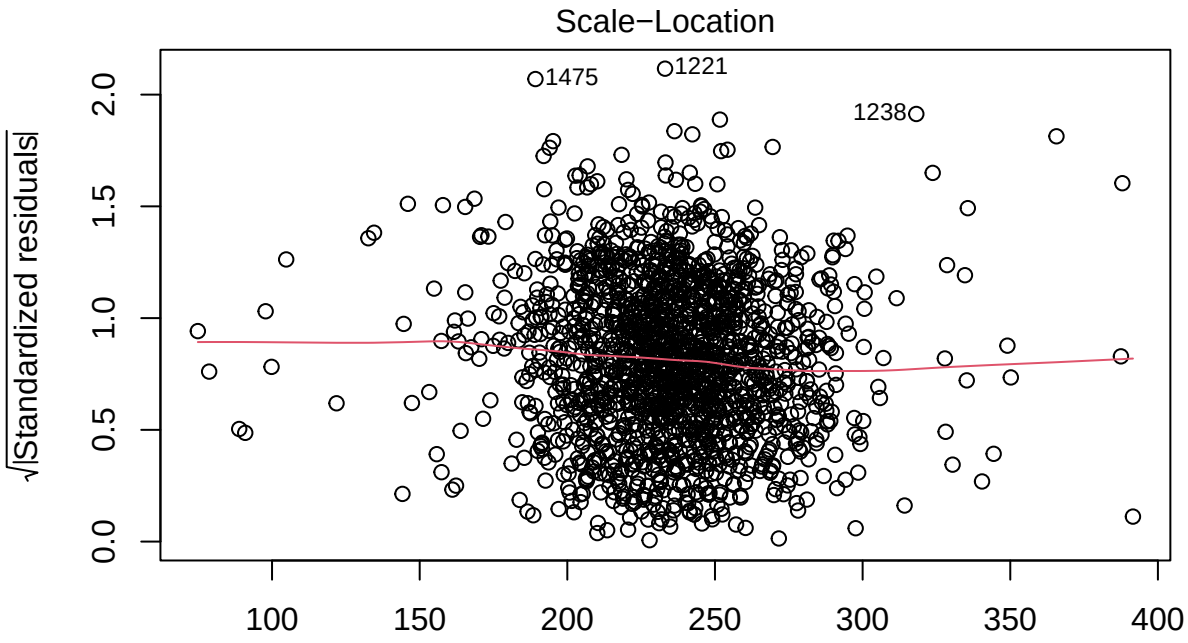
```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(stp_model_inv)
## W = 0.99631, p-value = 0.0003925
```

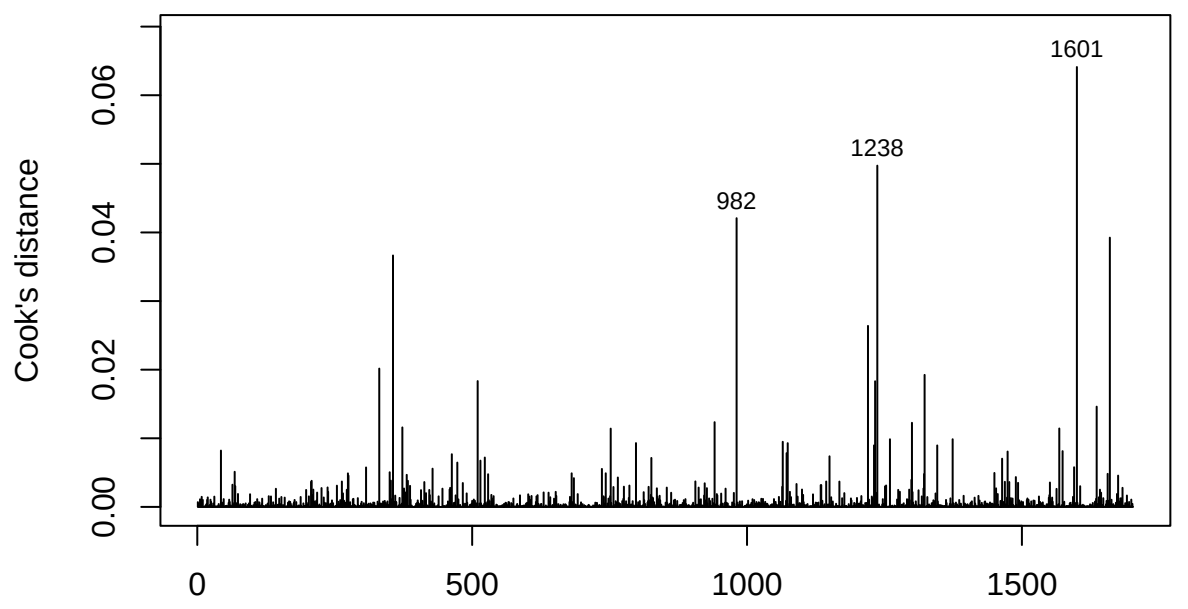
```
# Shapiro-Wilk normality test: look for high p-value
```

Our QQ plot suggests normality thought there is obvious skewing on the tails, particularly on the right.

A Shapiro Wilk's Test statistic had a value of 0.996, also suggesting normality. However, since the p-value (7.132e-05) is less than < 0.05 we may still be violating our normality assumption.

Heteroscedasticity






```
##
##  studentized Breusch-Pagan test
##
## data:  stp_model_inv
## BP = 154.07, df = 6, p-value < 2.2e-16
```

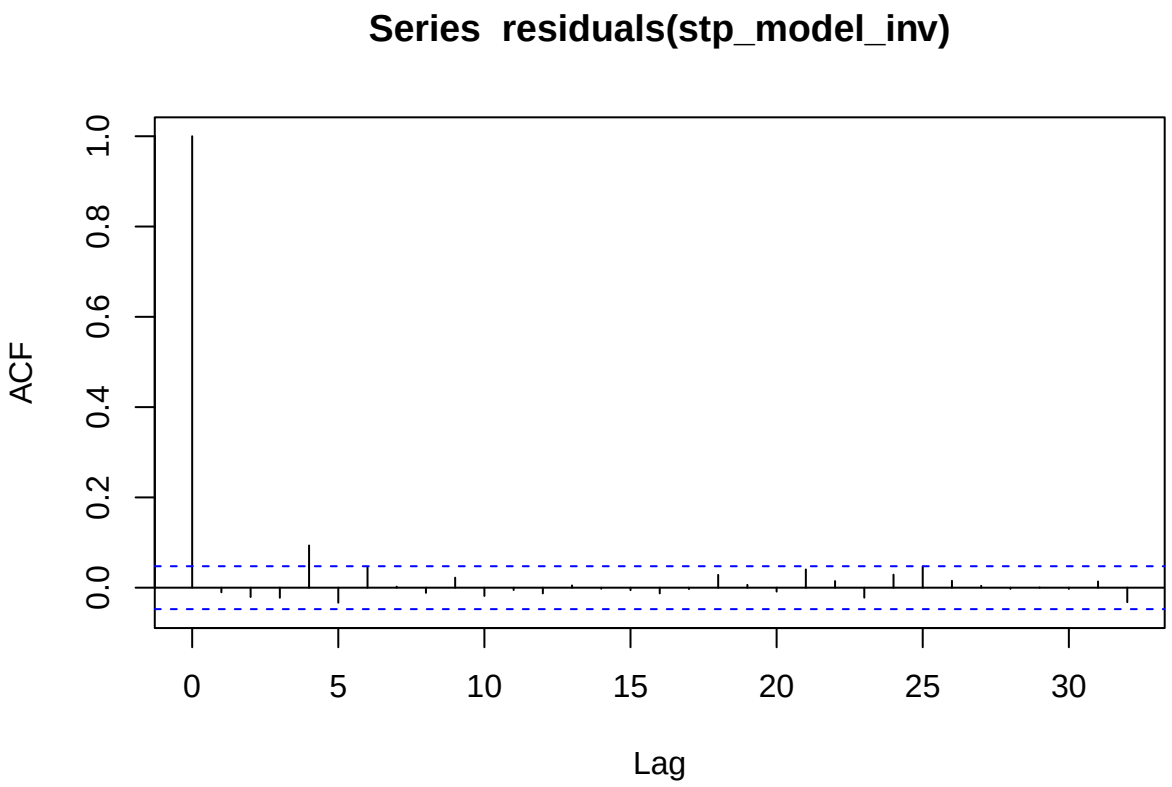
Our Scale-Location plot shows that points appear somewhat evenly distributed above and below the trend

line. While there is no obvious fan/wedge pattern, there is clustering in the center suggesting underfitting, high leverage outliers or that additional transformation may be needed. As our Cook's Distance plot has no values with a greater than 1, we can rule our the effects of high leverage points.

The Breusch-Page test statistic BP (176.76) and the small p-value (2.2e-16) suggest evidence of heteroscedasticity. Though the values are different that we may need to transform the data to meet the Assumption of Homoscedasticity.

Independence

```
acf(residuals(stp_model_inv))
```




```
durbinWatsonTest(stp_model_inv)
```

```
##  lag Autocorrelation D-W Statistic p-value
##    1     -0.01014699      2.019358   0.674
##  Alternative hypothesis: rho != 0
```

```
# Durbin Watson should be close to 2
```

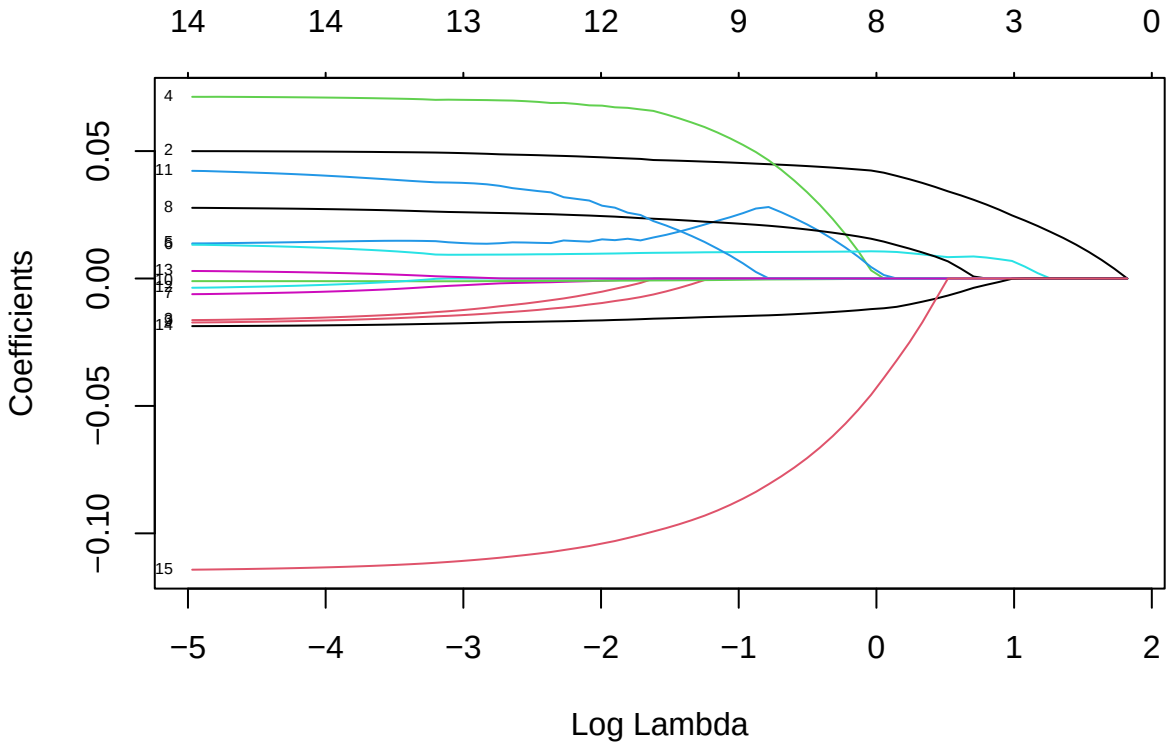
Our Autocorrelation Function shows that there are lags above the blue dashed line, suggesting no autocorrelation. This is confirmed through a Durbin-Watson test statistic value of 2.03 and an autocorrelation value of -0.021. Furthermore, as our p-value (0.416) is greater than 0.05, we do not have enough evidence to reject the null hypothesis that there is no autocorrelation. In other words, the test results suggest that our model's residuals are independent and therefore do not violate the Independence Assumption.

Conclusion Applying a Box-Cox transformation to our model appears does not seem to have improved the results of our backward selected model. This model continues to violate the Homoscedasticity Assumption and has mixed results for our Normality Assumption.

Advanced Techniques

Model A1: Lasso Regression The Least Absolute Shrinkage and Selection Operator (LASSO) selection method uses a shrinkage approach to determining the optimal predictors by attempting to find a balance between simplicity and accuracy.It applies a penalty to the standard linear regression model to encourage the coefficients of features with weak influence to equal zero to prevent overfitting.

```
x = model.matrix(TARGET_WINS ~ ., data = stp75_train_df)
y = stp75_train_df$TARGET_WINS

fit_lasso = glmnet(x, y, alpha=1)
plot(fit_lasso, xvar="lambda", label=TRUE)
```




```
cv_lasso = cv.glmnet(x,y,alpha=1)
#plot(cv.lasso)
coef(cv_lasso)
```

```
## 16 x 1 sparse Matrix of class "dgCMatrix"
##                               s1
## (Intercept)      19.5221297165
## (Intercept)       .
## TEAM_BATTING_H    0.0431544433
## TEAM_BATTING_2B   .
## TEAM_BATTING_3B   0.0173524152
## TEAM_BATTING_HR   0.0121590365
## TEAM_BATTING_BB   0.0105502074
## TEAM_BATTING_SO   .
## TEAM_BASERUN_SB   0.0172781093
## TEAM_BASERUN_CS   .
## TEAM_PITCHING_H  -0.0002011034
## TEAM_PITCHING_HR  .
## TEAM_PITCHING_BB  .
```

```
## TEAM_PITCHING_SO  .
## TEAM_FIELDING_E  -0.0128248793
## TEAM_FIELDING_DP -0.0569377921
```

Performing LASSO on our training subset – 75% of full training set with log transformation applied – gives us 7 predictors: - TEAM_BATTING_H - TEAM_BATTING_3B - TEAM_BATTING_BB - TEAM_BASERUN_SB - TEAM_PITCHING_HR - TEAM_FIELDING_E - TEAM_FIELDING_DP

```
lasso_model <- lm(TARGET_WINS ~ TEAM_BATTING_H + TEAM_BATTING_3B + TEAM_BATTING_BB + TEAM_BASERUN_SB + 

summary(lasso_model)
```

```
##
## Call:
## lm(formula = TARGET_WINS ~ TEAM_BATTING_H + TEAM_BATTING_3B +
##     TEAM_BATTING_BB + TEAM_BASERUN_SB + TEAM_PITCHING_HR + TEAM_FIELDING_E +
##     TEAM_FIELDING_DP, data = stp75_train_df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -47.063  -8.718  -0.031   8.628  56.109
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      17.688724   3.979040   4.445 9.34e-06 ***
## TEAM_BATTING_H    0.045567   0.002825  16.129  < 2e-16 ***
## TEAM_BATTING_3B   0.083852   0.018519   4.528 6.38e-06 ***
## TEAM_BATTING_BB   0.010914   0.003865   2.824   0.0048 **
## TEAM_BASERUN_SB   0.026415   0.004559   5.794 8.20e-09 ***
## TEAM_PITCHING_HR  0.040350   0.007740   5.213 2.09e-07 ***
## TEAM_FIELDING_E  -0.021314   0.002205  -9.666  < 2e-16 ***
## TEAM_FIELDING_DP -0.111953   0.014899  -7.514 9.21e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.16 on 1694 degrees of freedom
## Multiple R-squared:  0.2998, Adjusted R-squared:  0.2969
## F-statistic: 103.6 on 7 and 1694 DF,  p-value: < 2.2e-16
```

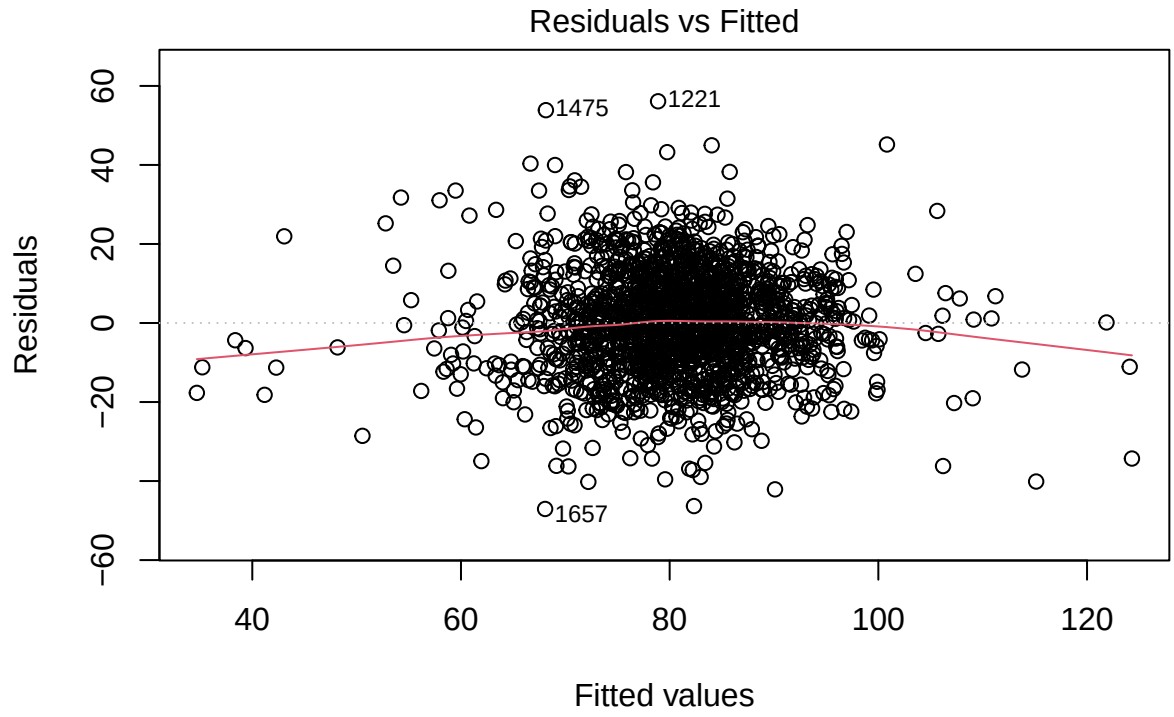
```
vif(lasso_model)
```

```
##   TEAM_BATTING_H  TEAM_BATTING_3B  TEAM_BATTING_BB  TEAM_BASERUN_SB
##         1.637967         2.557648         2.158278         1.524902
## TEAM_PITCHING_HR  TEAM_FIELDING_E TEAM_FIELDING_DP
##         2.251225         2.351028         1.326869
```

Our VIF test shows no strong correlation between predictors.

Testing Our Assumptions Linearity

```
plot(lasso_model, which=1)
```

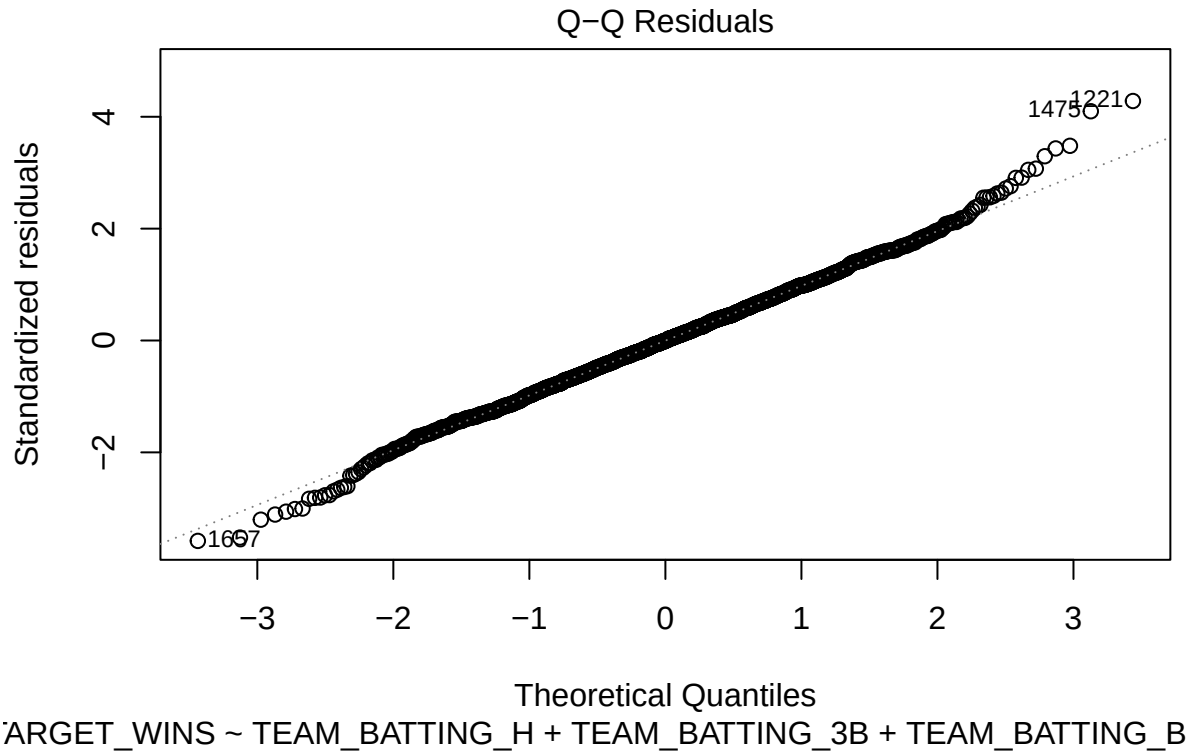


'ARGET_WINS ~ TEAM_BATTING_H + TEAM_BATTING_3B + TEAM_BATTING_BB + T

Our Residuals vs. Fitted plot suggest a fairly linear model.

Normality

```
plot(lasso_model, which = 2)
```



'ARGET_WINS ~ TEAM_BATTING_H + TEAM_BATTING_3B + TEAM_BATTING_BB + T

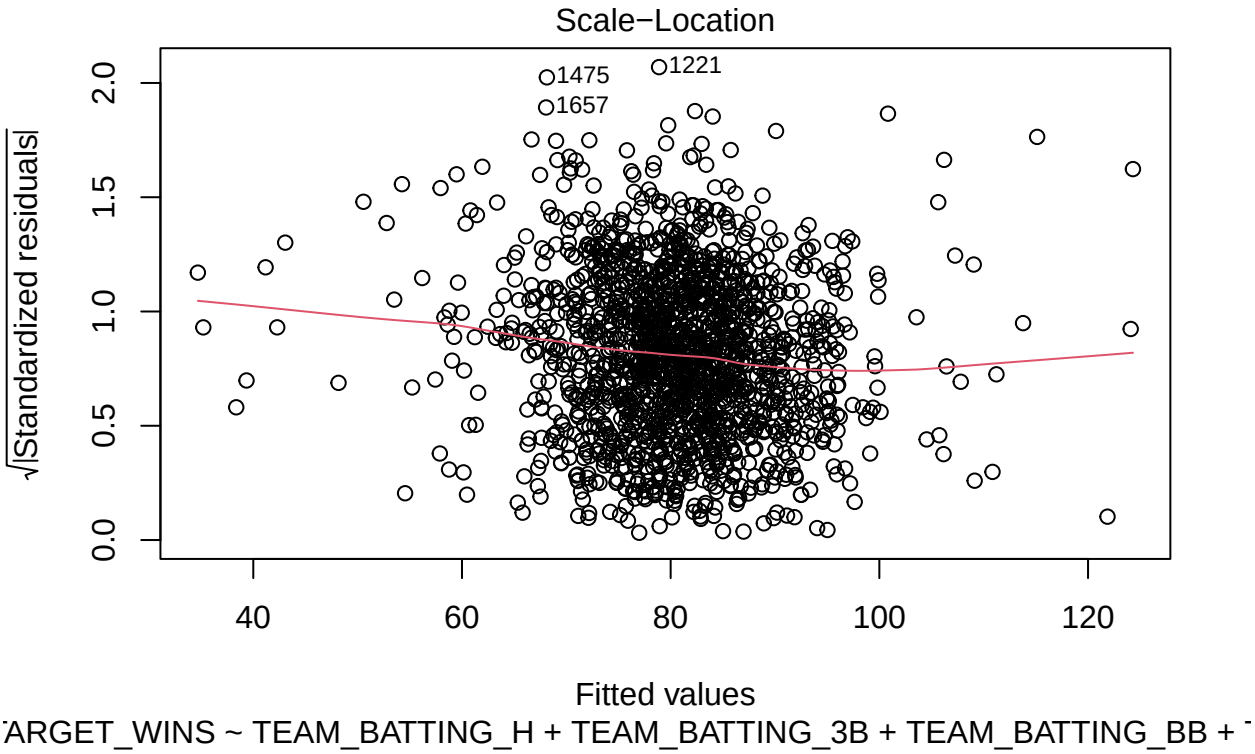
```
# Shapiro-Wilk normality test: look for high p-value
shapiro.test(residuals(lasso_model))
```

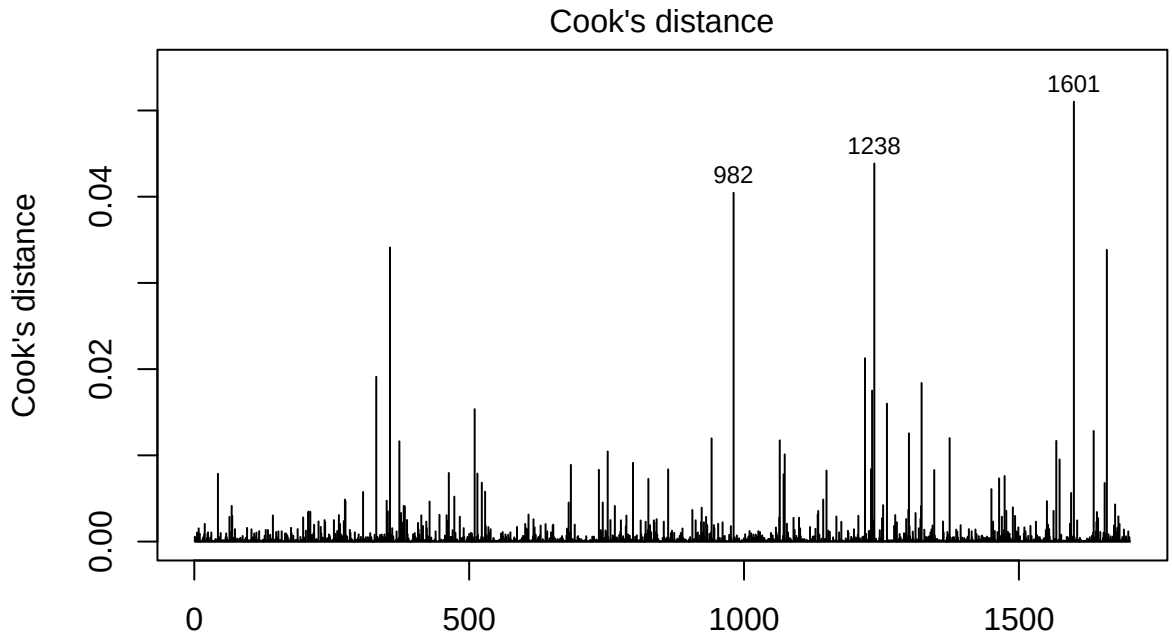
```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(lasso_model)
## W = 0.99689, p-value = 0.001695
```

Our QQ plot suggests normality thought there is some skewing on the tails, particularly on the right. The skewing appears to be less pitched than in our backward-selected model or Box-Cox Transformed model.

A Shapiro Wilk's Test statistic had a value of 0.9964, also suggesting normality. However, since the p-value (0.0005) is less than < 0.05 we may still be violating our normality assumption.

Heteroscedasticity





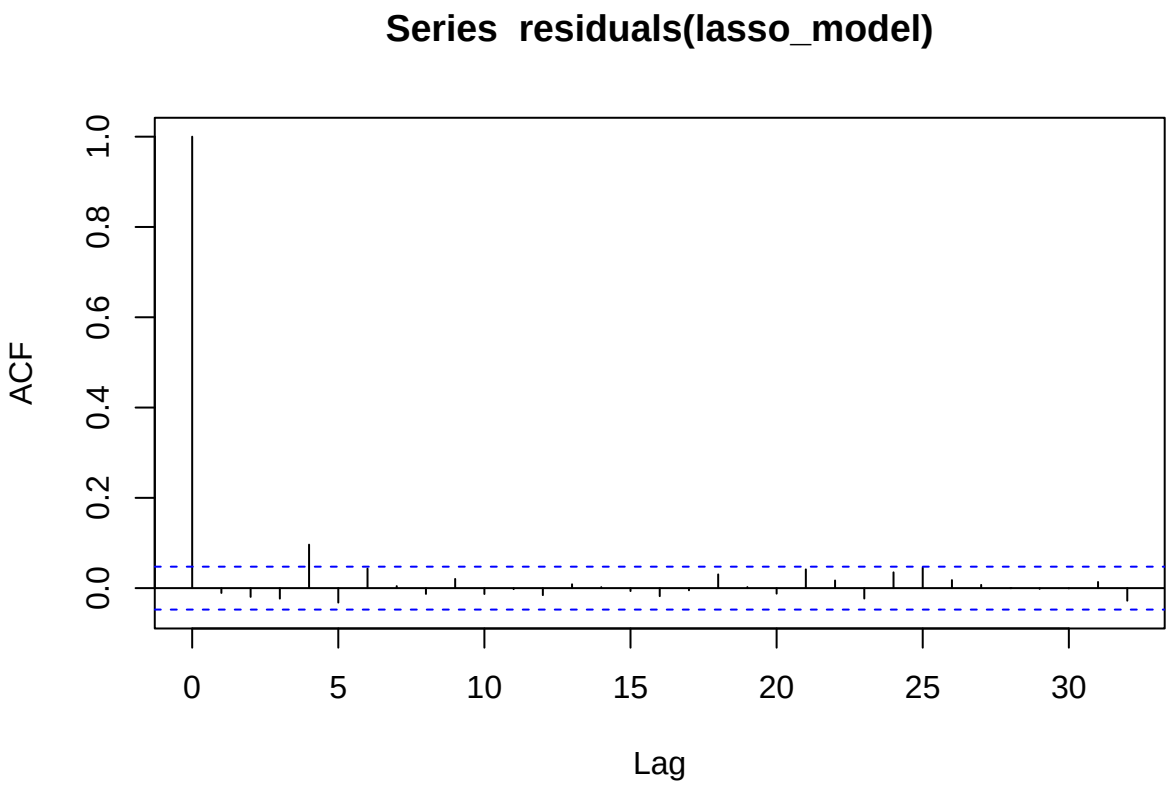

```
##
##  studentized Breusch-Pagan test
##
## data:  lasso_model
## BP = 188.81, df = 7, p-value < 2.2e-16
```

Our Scale-Location plot shows that points appear somewhat evenly distributed above and below the trend line but there is slight bowing in the trend line, suggesting that the variance of the residuals is not constant and therefore heteroschedastic. While there is no obvious fan/wedge pattern, there is clustering in the center suggesting underfitting, high leverage outliers or that additional transformation may be needed. As our Cook's Distance plot has no values with a greater than 1, we can rule our the effects of high leverage points.

The Breusch-Page test statistic BP (209.89) and the small p-value (2.2e-16) suggest evidence of heteroscedasticity. Though the values are different that we may need to transform the data to meet the Assumption of Homoscedasticity.

Independence

```
acf(residuals(lasso_model))
```




```
durbinWatsonTest(lasso_model)
```

```
##  lag Autocorrelation D-W Statistic p-value
##    1     -0.01068798      2.020502   0.646
##  Alternative hypothesis: rho != 0
```

```
# Durbin Watson should be close to 2
```

Our Autocorrelation Function shows that there are lags above the blue dashed line, suggesting no autocorrelation. This is confirmed through a Durbin-Watson test statistic value of 2.03 and an autocorrelation value of -0.0177. Furthermore, as our p-value (0.504) is greater than 0.05, we do not have enough evidence to reject the null hypothesis that there is no autocorrelation. In other words, the test results suggest that our model's residuals are independent and therefore do not violate the Independence Assumption.

Model A2: Elastic Net Regression Model Elastic Net Regression Model is being tested to balance LASSO and Ridge Regression, helpful when predictors are highly correlated. Adding Random Forest Regression as well as a non-parametric alternative, giving us the ability to capture complex nonlinear relationships and interactions between variables that linear models may miss.

```
#Zachs Models (Elastic Net)


x_train <- model.matrix(TARGET_WINS ~ ., data = stp75_train_df)[, -1]
y_train <- stp75_train_df$TARGET_WINS

x_test <- model.matrix(TARGET_WINS ~ ., data = stp25_test_df)[, -1]
y_test <- stp25_test_df$TARGET_WINS

# Define the tuning grid
elastic_net_grid <- expand.grid(alpha = seq(0, 1, by = 0.1), lambda = 10^seq(-3, 3, length = 100))
```

```r
# Perform cross-validation to find the best hyperparameters
cv_model <- train(x = x_train, y = y_train,
                  method = "glmnet",
                  trControl = trainControl(method = "cv", number = 5),
                  tuneGrid = elastic_net_grid)
```

```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo,
## : There were missing values in resampled performance measures.
```

```r
# Best model
best_alpha <- cv_model$bestTune$alpha
best_lambda <- cv_model$bestTune$lambda

# Fit final Elastic Net model
elastic_net_model <- glmnet(x_train, y_train, alpha = best_alpha, lambda = best_lambda)

# Predictions
elastic_net_predictions <- predict(elastic_net_model, newx = x_test, s = best_lambda)

# Model Evaluation
elastic_net_mae <- mae(y_test, elastic_net_predictions)
elastic_net_rmse <- rmse(y_test, elastic_net_predictions)
elastic_net_r2 <- cor(y_test, elastic_net_predictions)^2

cat("Elastic Net Regression Performance:\n")
```

```
## Elastic Net Regression Performance:
```

```r
cat("MAE:", elastic_net_mae, "\n")
```

```
## MAE: 10.00153
```

```r
cat("RMSE:", elastic_net_rmse, "\n")
```

```
## RMSE: 12.86902
```

```r
cat("R²:", elastic_net_r2, "\n")
```

```
## R²: 0.2438929
```

```r
#Zachs Models (Random Forest Regression)
library(randomForest)
```

Model A3: Random Forest Regression Model

```
## randomForest 4.7-1.2
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
## The following object is masked from 'package:dplyr':
##
##     combine
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```r
set.seed(000)
random_forest_model <- randomForest(TARGET_WINS ~ ., data = stp75_train_df, ntree = 500, importance = T

# Predictions
random_forest_predictions <- predict(random_forest_model, newdata = stp25_test_df)

# Model Evaluation
random_forest_mae <- mae(y_test, random_forest_predictions)
random_forest_rmse <- rmse(y_test, random_forest_predictions)
random_forest_r2 <- cor(y_test, random_forest_predictions)^2

cat("\nRandom Forest Regression Performance:\n")
```

```
##
## Random Forest Regression Performance:
```

```r
cat("MAE:", random_forest_mae, "\n")
```

```
## MAE: 8.990392
```

```r
cat("RMSE:", random_forest_rmse, "\n")
```

```
## RMSE: 11.73108
```

```r
cat("R²:", random_forest_r2, "\n")
```

```
## R²: 0.3827465
```

Section 4: Model Selection

Testing the Model In this section, we will use the test dataframe to test the accuracy of our model's predictions

```r
# Apply predictions for each model
# Stats Model predictions
predictedWins = predict(base_model, stp25_test_df)
stp25_test_df$PREDICTED_WINS = predictedWins

# Improved Model predictions
predictedWins = predict(improved_model, stp25_test_df)
stp25_test_df$PREDICTED_WINS_IMP = predictedWins

# High-Impact Model predictions
predictedWins = predict(high_impact_model, stp25_test_df)
stp25_test_df$PREDICTED_WINS_HIM = predictedWins

# PJ1 predictions
predictedWins = predict(pj_model1, stp25_test_df)
stp25_test_df$PREDICTED_WINS_PJ1 = predictedWins

# PJ2 predictions
predictedWins = predict(pj_model2, stp25_test_df)
stp25_test_df$PREDICTED_WINS_PJ2 = predictedWins

# PJ4 predictions
predictedWins = predict(pj_model4, stp25_test_df)
stp25_test_df$PREDICTED_WINS_PJ4 = predictedWins
```

```r
# PJ5 predictions
predictedWins = predict(pj_model5, stp25_test_df)
stp25_test_df$PREDICTED_WINS_PJ5 = predictedWins

# PJ6 predictions
predictedWins = predict(pj_model6, stp25_test_df)
stp25_test_df$PREDICTED_WINS_PJ6 = predictedWins

# Log-Transformed Model predictions
predictedWins = predict(log_model1, test_df_log)
test_df_log$PREDICTED_WINS_LOG1 = predictedWins

# Log-Transformed Model predictions
predictedWins = predict(log_model2, test_df_log)
test_df_log$PREDICTED_WINS_LOG2 = predictedWins

# Log-Transformed Step-wise Model predictions
predictedWins = predict(stp_logx_model, test_df_log)
test_df_log$PREDICTED_WINS_STP = predictedWins

# Lasso Model predictions
predictedWins = predict(lasso_model, stp25_test_df)
test_df_log$PREDICTED_WINS_LASSO = predictedWins


# Convert Elastic Net predictions to numeric vector
predictedWins <- as.numeric(predict(elastic_net_model, newx = x_test, s = best_lambda)[, 1])

# Store the predictions in the dataframe
stp25_test_df$PREDICTED_WINS_ELASTIC <- predictedWins


# Random Forest Model predictions
stp25_test_df$PREDICTED_WINS_RF <- predict(random_forest_model, newdata = stp25_test_df)
```

```r
#df to store results
evaluation_metrics <- data.frame(
  model_name = character(0),
  MAE = numeric(0),
  RMSE = numeric(0),
  RSquared = numeric(0),
  AIC = numeric(0),
  BIC = numeric(0)
)

evalution_plots <- list()
i <- 1

eval_predictions <- function(model, predicted_col, target_col, model_name) {

  # Ensure predict_col and target_col are valid vectors
  if (is.null(predicted_col) || is.null(target_col)) {
    stop("Error: The prediction or target column is NULL.")
```

```
  }

  # Residual plot
  residuals = target_col - predicted_col
  plot_data <- data.frame(Index = 1:length(residuals), Residuals = residuals)

  evalution_plots[[i]] <<- ggplot(plot_data, aes(x = Index, y = Residuals)) +
    geom_point() +
    geom_hline(yintercept = 0, linetype = "dashed") +
    labs(title = paste(model_name, "Residual Plot"), x = "Index", y = "Residuals") +
    theme_minimal()
  i <<- i + 1

  # Compute model metrics
  model_metrics = data.frame(
    model_name = model_name,
    MAE = mae(target_col, predicted_col),
    RMSE = rmse(target_col, predicted_col),
    RSquared = cor(target_col, predicted_col)^2,
    AIC = tryCatch(AIC(model), error = function(e) NA),  # Handle AIC for unsupported models
    BIC = tryCatch(BIC(model), error = function(e) NA)   # Handle BIC for unsupported models
  )

  # Append results to global evaluation metrics
  evaluation_metrics <<- rbind(evaluation_metrics, model_metrics)
}

# call eval_predictions for each model
eval_predictions(base_model, stp25_test_df$PREDICTED_WINS, stp25_test_df$TARGET_WINS, "Stats")
eval_predictions(improved_model, stp25_test_df$PREDICTED_WINS_IMP, stp25_test_df$TARGET_WINS, "Improved
eval_predictions(high_impact_model, stp25_test_df$PREDICTED_WINS_HIM, stp25_test_df$TARGET_WINS, "High-
eval_predictions(pj_model1, stp25_test_df$PREDICTED_WINS_PJ1, stp25_test_df$TARGET_WINS, "PJ1")
eval_predictions(pj_model2, stp25_test_df$PREDICTED_WINS_PJ2, stp25_test_df$TARGET_WINS, "PJ2")
eval_predictions(pj_model4, stp25_test_df$PREDICTED_WINS_PJ4, stp25_test_df$TARGET_WINS, "PJ4")
eval_predictions(pj_model5, stp25_test_df$PREDICTED_WINS_PJ5, stp25_test_df$TARGET_WINS, "PJ5")
eval_predictions(pj_model6, stp25_test_df$PREDICTED_WINS_PJ6, stp25_test_df$TARGET_WINS, "PJ6")
eval_predictions(log_model1, test_df_log$PREDICTED_WINS_LOG1, test_df_log$TARGET_WINS, "Log-Tranformed
eval_predictions(log_model2, test_df_log$PREDICTED_WINS_LOG2, test_df_log$TARGET_WINS, "Log-Tranformed
eval_predictions(stp_logx_model, test_df_log$PREDICTED_WINS_STP, test_df_log$TARGET_WINS, "LT Step-wise
eval_predictions(lasso_model, test_df_log$PREDICTED_WINS_LASSO, test_df_log$TARGET_WINS, "Lasso")
eval_predictions(elastic_net_model, stp25_test_df$PREDICTED_WINS_ELASTIC, stp25_test_df$TARGET_WINS, "E
eval_predictions(random_forest_model, stp25_test_df$PREDICTED_WINS_RF, stp25_test_df$TARGET_WINS, "Rand
```

```
evaluation_metrics <- evaluation_metrics[order(evaluation_metrics$MAE), ]
print(evaluation_metrics)
```

```
##          model_name       MAE     RMSE  RSquared      AIC      BIC
## 14    Random Forest  8.990392 11.73108 0.3827465       NA       NA
## 11     LT Step-wise  9.917226 12.80085 0.2533108 13572.54 13648.69
## 13      Elastic Net 10.001529 12.86902 0.2438929       NA       NA
## 2          Improved 10.028299 12.89528 0.2406957 13720.76 13769.72
## 12            Lasso 10.030810 12.94165 0.2369009 13613.94 13662.90
## 5               PJ2 10.202310 13.20175 0.2055416 13685.45 13728.96
## 4               PJ1 10.228264 13.22372 0.2033742 13687.10 13725.18
```

```
## 8               PJ6 10.273194 13.36420 0.1844932 13797.00 13829.63
## 10 Log-Tranformed 2 10.304164 13.29283 0.1951432 13695.71 13733.79
## 6               PJ4 10.391404 13.36286 0.1852952 13775.04 13813.11
## 3       High-Impact 10.391479 13.29616 0.1927413 13773.74 13817.25
## 7               PJ5 10.405873 13.31412 0.1906206 13772.45 13810.53
## 1             Stats 10.453357 13.39290 0.1816053 13779.20 13811.84
## 9  Log-Tranformed 1 10.520719 13.43778 0.1769022 13770.32 13802.96
```

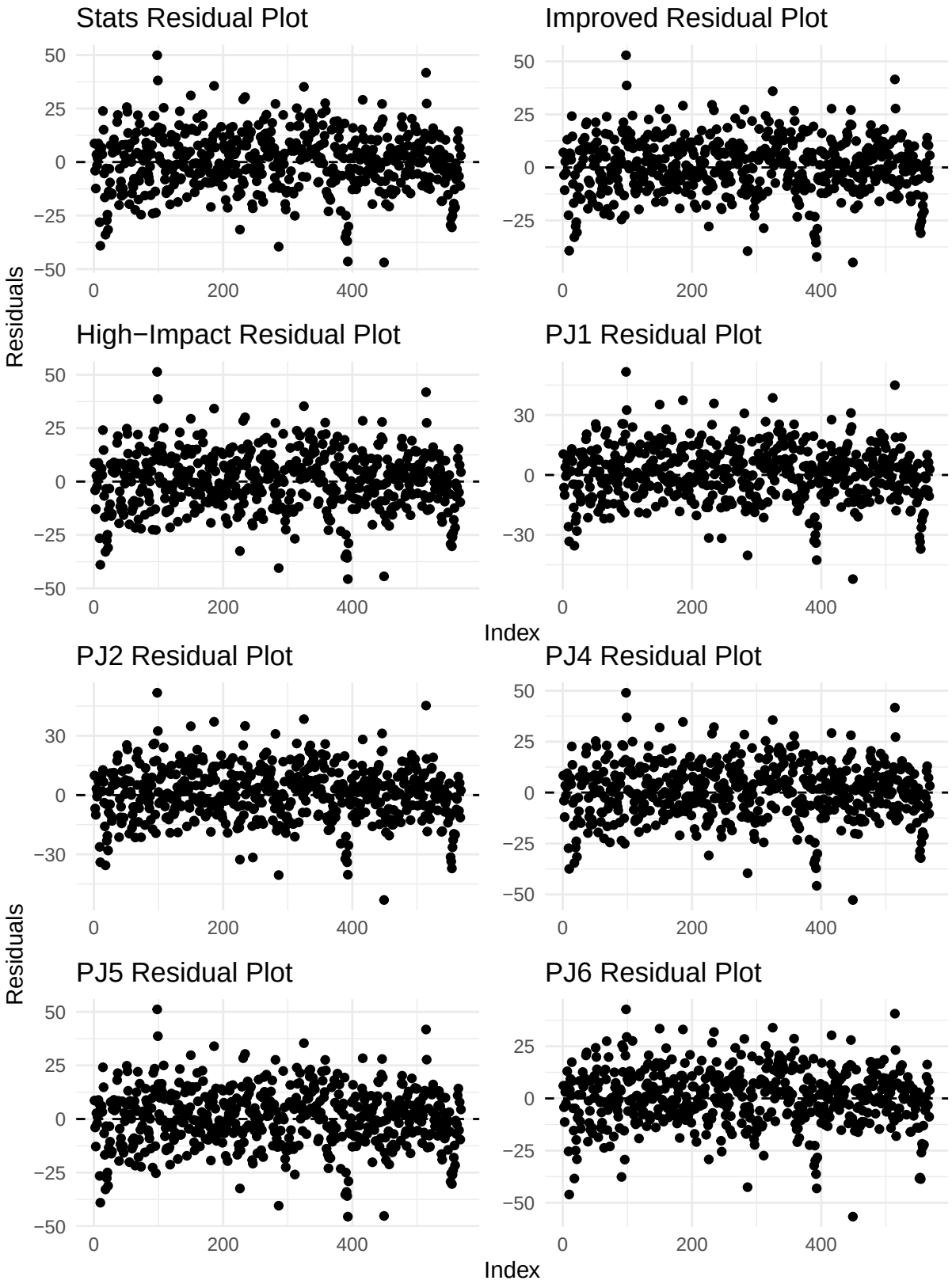
```
library(patchwork)
```

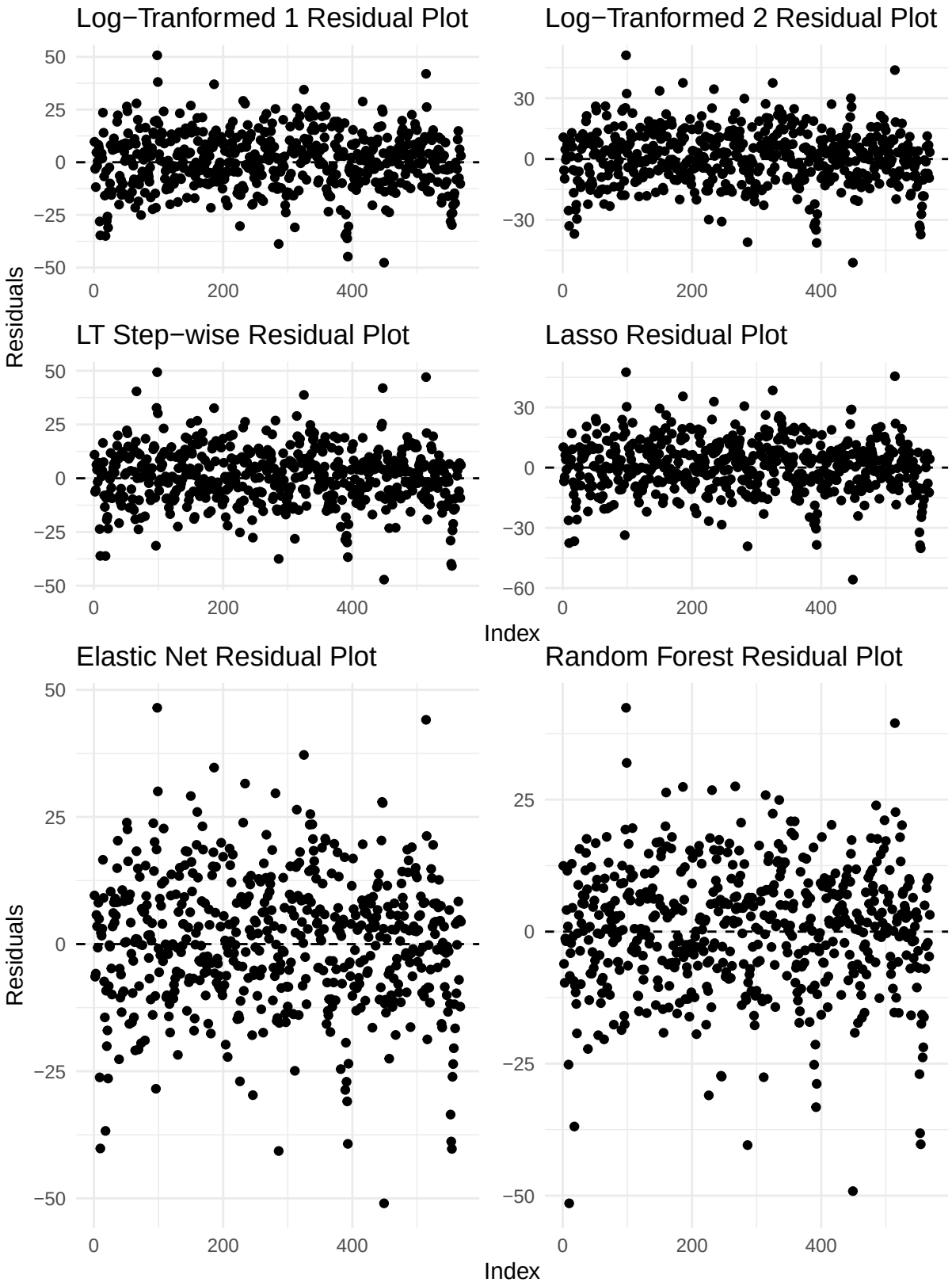
Comparing All Models

```
##
## Attaching package: 'patchwork'
```

```
## The following object is masked from 'package:MASS':
##
##     area
```

```
performance_df <- compare_performance(base_model, high_impact_model, pj_model1, pj_model2, pj_model4, p


# Loop to print the plots in batches of 4
for (i in seq(1, length(evalution_plots), by = 4)) {

  # Create a subset of the plots for the current batch
  plot_batch <- evalution_plots[i:min(i + 4 - 1, length(evalution_plots))]

  # Display the current batch of plots
  print(wrap_plots(plot_batch, ncol = 2, axis_titles='collect'))
}
```


Stats Residual Plot
Improved Residual Plot
High−Impact Residual Plot
PJ1 Residual Plot
PJ2 Residual Plot
PJ4 Residual Plot
PJ5 Residual Plot
PJ6 Residual Plot
Residuals
Index



We selected the Lasso model for its balance of predictive performance and simplicity. It automatically performs feature selection through its regularization term, helping us prevent overfitting while identifying the most relevant features for predicting TARGET_WINS. Although the model's R-squared indicates that it

doesn't capture all of the variation in the target, its moderate MAE and RMSE provide a reliable baseline. Additionally, Lasso's interpretability allows us to better understand the relationship between predictors and the outcome. By choosing Lasso, we opted for a model that offers good performance, simplicity, and interpretability, making it easier to refine or extend in future iterations.

Below we note our lasso perfomance metrics:

```
lasso_metrics <- evaluation_metrics %>%
  filter(model_name == "Lasso") %>%
  distinct(model_name, .keep_all = TRUE)

print(lasso_metrics)
```

```
##   model_name      MAE     RMSE  RSquared      AIC     BIC
## 1      Lasso 10.03081 12.94165 0.2369009 13613.94 13662.9
```

Applying Prediction to Our Evalution

With our models tested and evaluated, we can now apply our model to our final evaluation dataframe.

Preparing the Evalulation Dataset First we apply the same imputations and transformation our test dataframe that we applied to our training dataframe.

```
numeric_cols <- sapply(stp25_test_df, is.numeric)

# Impute missing values for numeric columns only
stp25_test_df[numeric_cols][is.na(stp25_test_df[numeric_cols])] <-
  lapply(stp25_test_df[numeric_cols], function(x) mean(x, na.rm = TRUE))
```

We use the selected model to predict wins on our test dataframe.

```
# Apply the Lasso model to the final evaluation dataframe
predictedWins <- predict(lasso_model, stp25_test_df)
stp25_test_df$PREDICTED_WINS <- predictedWins
```

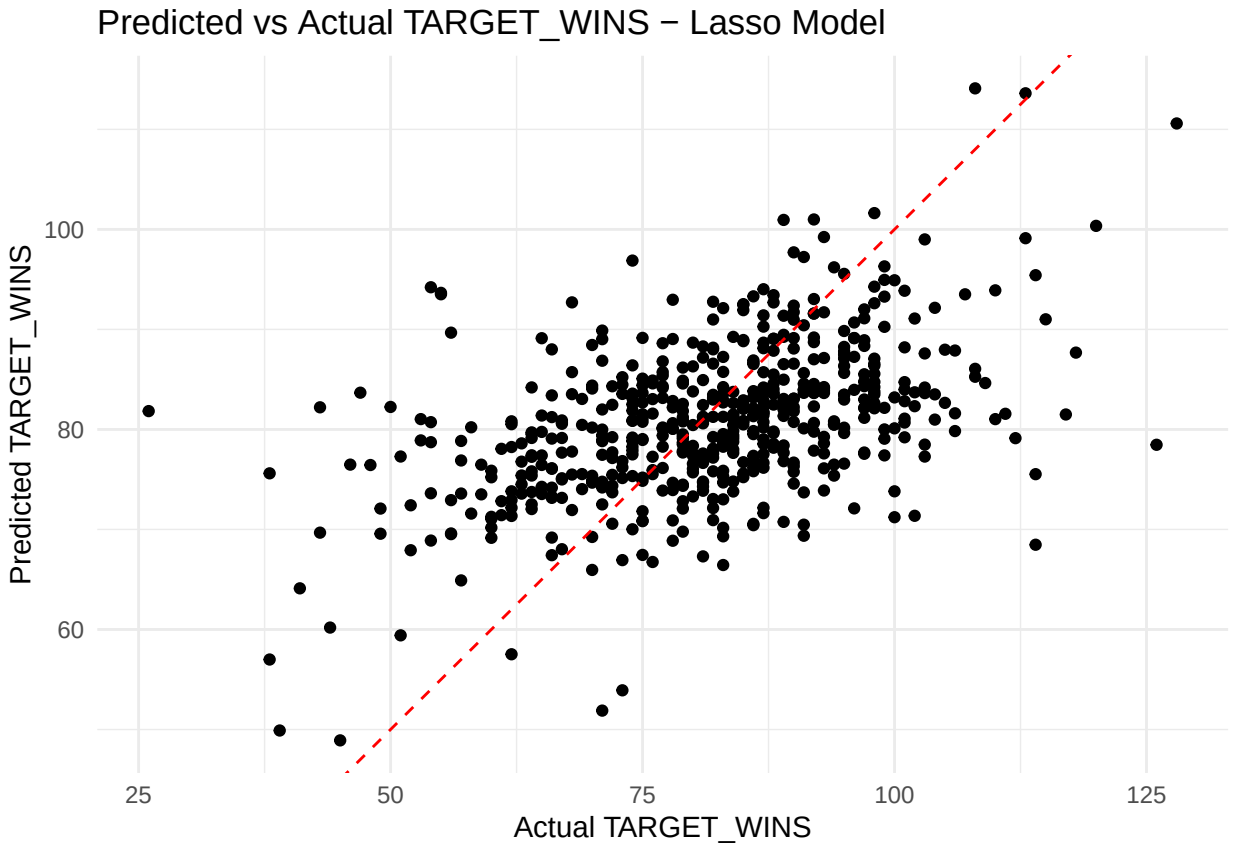
We also back transform our predicted wins for ease of comparison.

```
# Backward transform (inverse Box-Cox) for Box-Cox transformed model
# Assuming 'best_lambda' is the value of lambda used for Box-Cox transformation

# If 'best_lambda' is zero, use exponential transformation
if (best_lambda == 0) {
  predictedWins_inv <- exp(predictedWins)  # Exponential transformation if lambda = 0
} else {
  predictedWins_inv <- ((predictedWins * best_lambda) + 1)^(1 / best_lambda)
}

# Store the back-transformed predictions in the dataframe
stp25_test_df$PREDICTED_WINS_INV <- predictedWins_inv

# Evaluate the back-transformed predictions
eval_predictions(base_model, stp25_test_df$PREDICTED_WINS_INV, stp25_test_df$TARGET_WINS, "Base Model (
```

Below is a plot of our Predicted vs Actual wins.

```
# Predicted vs Actual plot for the Lasso Model
ggplot(stp25_test_df, aes(x = TARGET_WINS, y = PREDICTED_WINS)) +
  geom_point() +
```

```
  geom_abline(slope = 1, intercept = 0, linetype = "dashed", color = "red") +
  labs(title = "Predicted vs Actual TARGET_WINS - Lasso Model",
       x = "Actual TARGET_WINS",
       y = "Predicted TARGET_WINS") +
  theme_minimal()
```



The Lasso model achieved an MAE of 10.0308097 which suggests the model has a low average error in predicting TARGET_WINS. The RMSE value of 12.9416521 supports this conclusion. The R-squared value of 0.2369009 indicates that the Lasso model explains a substantial portion of the variance in the TARGET_WINS.

```
### Interpretation of Predicted TARGET_WINS Values

The plot featured above displays predicted wins versus actual values of target wins using a LASSO model

### Evaluating the Accuracy of Predictions Against Actual TARGET_WINS

#### Interpretation of Model Accuracy Metrics

The LASSO model's performance metrics provide insights into its accuracy. The Mean Absolute Error (MAE)

#### Final Analysis

The Lasso model was selected for its balance between predictive performance and simplicity. Lasso is pa
```

Applying selcted model (Lasso) to the final evaluation data

Preparing final evaluation dataset

``` r
na_per_column <- colSums(is.na(eval_df))
print(na_per_column)
```

```
##   TEAM_BATTING_H  TEAM_BATTING_2B  TEAM_BATTING_3B  TEAM_BATTING_HR
##                0                0                0                0
##  TEAM_BATTING_BB  TEAM_BATTING_SO  TEAM_BASERUN_SB  TEAM_BASERUN_CS
##                0               18               13               87
## TEAM_BATTING_HBP  TEAM_PITCHING_H TEAM_PITCHING_HR TEAM_PITCHING_BB
##              240                0                0                0
## TEAM_PITCHING_SO  TEAM_FIELDING_E TEAM_FIELDING_DP
##               18                0               31
```

```r
eval_df <- eval_df[, !names(eval_df) %in% "TEAM_BATTING_HBP"]
numeric_cols_eval <- sapply(eval_df, is.numeric)

str(eval_df)
```

```
## 'data.frame':    259 obs. of  14 variables:
##  $ TEAM_BATTING_H  : int  1209 1221 1395 1539 1445 1431 1430 1385 1259 1397 ...
##  $ TEAM_BATTING_2B : int  170 151 183 309 203 236 219 158 177 212 ...
##  $ TEAM_BATTING_3B : int  33 29 29 29 68 53 55 42 78 42 ...
##  $ TEAM_BATTING_HR : int  83 88 93 159 5 10 37 33 23 58 ...
##  $ TEAM_BATTING_BB : int  447 516 509 486 95 215 568 356 466 452 ...
##  $ TEAM_BATTING_SO : int  1080 929 816 914 416 377 527 609 689 584 ...
##  $ TEAM_BASERUN_SB : int  62 54 59 148 NA NA 365 185 150 52 ...
##  $ TEAM_BASERUN_CS : int  50 39 47 57 NA NA NA NA NA NA ...
##  $ TEAM_PITCHING_H : int  1209 1221 1395 1539 3902 2793 1544 1626 1342 1489 ...
##  $ TEAM_PITCHING_HR: int  83 88 93 159 14 20 40 39 25 62 ...
##  $ TEAM_PITCHING_BB: int  447 516 509 486 257 420 613 418 497 482 ...
##  $ TEAM_PITCHING_SO: int  1080 929 816 914 1123 736 569 715 734 622 ...
##  $ TEAM_FIELDING_E : int  140 135 156 124 616 572 490 328 226 184 ...
##  $ TEAM_FIELDING_DP: int  156 164 153 154 130 105 NA 104 132 145 ...
```

```r
numeric_cols_eval
```

```
##   TEAM_BATTING_H  TEAM_BATTING_2B  TEAM_BATTING_3B  TEAM_BATTING_HR
##             TRUE             TRUE             TRUE             TRUE
##  TEAM_BATTING_BB  TEAM_BATTING_SO  TEAM_BASERUN_SB  TEAM_BASERUN_CS
##             TRUE             TRUE             TRUE             TRUE
##  TEAM_PITCHING_H TEAM_PITCHING_HR TEAM_PITCHING_BB TEAM_PITCHING_SO
##             TRUE             TRUE             TRUE             TRUE
##  TEAM_FIELDING_E TEAM_FIELDING_DP
##             TRUE             TRUE
```

```r
# Impute missing values for numeric columns only
eval_df[numeric_cols_eval] <-
  lapply(eval_df[numeric_cols_eval], function(x) {
    x[is.na(x)] <- mean(x, na.rm = TRUE)
    return(x)
  })
```

```r
# Ensure any transformations match the ones used during training
eval_df$log_TEAM_BATTING_H <- log(eval_df$TEAM_BATTING_H + 1)
```

```r
# Apply the Lasso model to the final evaluation dataframe
predictedWinsEval <- round(predict(lasso_model, eval_df))
eval_df$PREDICTED_WINS <- predictedWinsEval

glimpse(eval_df)
```

Predicting Wins using Lasso

```
## Rows: 259
## Columns: 16
## $ TEAM_BATTING_H     <dbl> 1209, 1221, 1395, 1539, 1445, 1431, 1430, 1385, 125~
## $ TEAM_BATTING_2B    <dbl> 170, 151, 183, 309, 203, 236, 219, 158, 177, 212, 2~
## $ TEAM_BATTING_3B    <dbl> 33, 29, 29, 29, 68, 53, 55, 42, 78, 42, 40, 55, 57,~
## $ TEAM_BATTING_HR    <dbl> 83, 88, 93, 159, 5, 10, 37, 33, 23, 58, 50, 164, 18~
## $ TEAM_BATTING_BB    <dbl> 447, 516, 509, 486, 95, 215, 568, 356, 466, 452, 49~
## $ TEAM_BATTING_SO    <dbl> 1080.0000, 929.0000, 816.0000, 914.0000, 416.0000, ~
## $ TEAM_BASERUN_SB    <dbl> 62.0000, 54.0000, 59.0000, 148.0000, 123.7033, 123.~
## $ TEAM_BASERUN_CS    <dbl> 50.00000, 39.00000, 47.00000, 57.00000, 52.31977, 5~
## $ TEAM_PITCHING_H    <dbl> 1209, 1221, 1395, 1539, 3902, 2793, 1544, 1626, 134~
## $ TEAM_PITCHING_HR   <dbl> 83, 88, 93, 159, 14, 20, 40, 39, 25, 62, 53, 173, 1~
## $ TEAM_PITCHING_BB   <dbl> 447, 516, 509, 486, 257, 420, 613, 418, 497, 482, 5~
## $ TEAM_PITCHING_SO   <dbl> 1080.000, 929.000, 816.000, 914.000, 1123.000, 736.~
## $ TEAM_FIELDING_E    <dbl> 140, 135, 156, 124, 616, 572, 490, 328, 226, 184, 2~
## $ TEAM_FIELDING_DP   <dbl> 156.000, 164.000, 153.000, 154.000, 130.000, 105.00~
## $ log_TEAM_BATTING_H <dbl> 7.098376, 7.108244, 7.241366, 7.339538, 7.276556, 7~
## $ PREDICTED_WINS     <dbl> 65, 65, 74, 86, 66, 70, 78, 76, 72, 74, 71, 81, 80,~
```

```r
write.csv(eval_df, "predictions.csv", row.names = F)
```