# HW 1

Teddy Kelly

## Table of contents

**Motivation** 3

**Setup** 3

**1 Data Exploration** 5
- 1.1 Summary Statistics . . . 5
- 1.2 Correlation Table . . . 10

**2 Data Preparation** 12
- 2.1 Visualizing Missing Values . . . 12
- 2.2 Imputing Missing Values . . . 13
- 2.3 Impute Values of 0 . . . 14
- 2.4 Addressing Outlier Values . . . 16
- 2.5 Variable Transformations . . . 17
- 2.6 Updated Summary Statistics . . . 18

**3 Build Models** 19
- 3.1 Model Construction . . . 19
  - Model 1 . . . 19
  - Model 2 . . . 21
  - Model 3 . . . 22
  - Other Models . . . 22
- 3.3 Model Interpretations . . . 22

**4 Select Models** 27
- 4.1 Model Diagnostics . . . 27
  - Variance Inflation Factor . . . 27
  - Mean Squared Error, R-Squared, and F-Statistic . . . 28
- 4.2 Cleaning Evaluation Data . . . 30
- 4.3 Predictions . . . 35

**Conclusion** **40**

## Motivation

Major League Baseball, or the MLB as it is widely known as, is comprised of 30 professional baseball teams. Every year, these teams compete against one another over an 162 game regular season and an extensive post-season for the honor of being crowned as World Series champions. And an honor it is as only one team every year can achieve this glory, however, the ultimate glory is the substantial amount of data that each team gains access to from the entire season. That's data on every single team from every one of their 162 games, totaling to precisely 2,430 MLB games each year. And that's just for one season. Professional baseball has been going on since 1871 and it has continued through the MLB for the last 150 or so. With access to that much data, current teams can investigate the statistics of teams who had great success and teams who did poorly to learn what factors most significantly contribute to winning games.

This is precisely the goal of this report: to develop a model that can accurately predict MLB wins based on the statistical categories provided to us. We are given a training data set of about 2,276 baseball teams from between the years 1871-2006 with data on their team's batting, pitching, and fielding numbers with their corresponding number of wins for that season. Using this data, I have developed three distinct linear regression models that each uniquely quantify the significance of different statistical features in determining wins. I will discuss the rationale behind each model and I will select the best model, based on various diagnostic tests, to predict win totals for each team in the evaluation data set.

The regression model will not only allow us to predict wins, but it will also allow us to determine which statistical categories are most significantly related to winning. This information is extremely useful for general managers who are seeking to understand what statistical areas they should focus on improving to give themselves the best chance to win baseball games.

## Setup

In this document, we will build a multiple linear regression to predict the number of wins for a baseball team using the given training data set.

Before we do so, we must clear our environment and extra things that might get in the way before importing the data sets.

```
# Clearing the environment
rm(list=ls())

# Clearing unused memory
gc()

# Clear the console
```

```r
cat("\014")

# Clearing all of the pots
while (!is.null(dev.list())) {
  dev.off()
}
```

Now, that we have cleared all of the unnecessary items in our environment, we will load all of the necessary packages for this assignment.

```r
# Defining the packages I will use
packages <- c("tidyverse",
              "stargazer",
              "knitr",
              "kableExtra",
              "visdat",
              "psych",
              "ggplot2",
              "car",
              "ggfortify")
# Load all of the packages
for (package in packages) {
  library(package, character.only = TRUE)
}
remove(packages)
```

```r
# Loading the data sets
train_data <- read.csv(
  "/Users/teddykelly/Downloads/moneyball-training-data-1-1.csv")

eval_data  <- read.csv(
  "/Users/teddykelly/Downloads/moneyball-evaluation-data-1-1.csv")
```

# 1 Data Exploration

The Money ball training data set contains 2,276 observations with each observation representing a professional baseball team's regular season from 1871-2006. For each team, there are 16 variables of interest that contain data on total wins, as well as several batting, base running, fielding, and pitching statistics. Below is an overview of the variables included in the data set. Our goal is to determine which batting, base running, fielding, and pitching statistics are strongly associated with the total number of wins these teams achieved. We will then use those results to predict the number of wins for the teams in the evaluation data set. Below is a table with all of the relevant variables in the Moneyball data set.

Table 1: Variable Names and Meaning

| **Variable Names** | **meaning** |
|---|---|
| wins | Total Wins |
| bat_hits | Total hits for |
| pitch_hits | Total hits allowed |
| doubles | Total doubles |
| triples | Total triples |
| bat_hr | Total home runs hit |
| pitch_hr | Total home runs allowed |
| bat_walks | Walks for |
| pitch_walks | Walks allowed |
| cs | Number of times caught stealing |
| hbp | Number of times hit by pitch |
| bat_so | Total strike outs pitched |
| pitch_so | Total strike outs swing |
| sb | Stolen bases |
| errors | Errors |
| dp | Double plays |

## 1.1 Summary Statistics

Below are the summary statistics of each of the variables, most notably including information on the mean, median, standard deviation, and the standard error. Note that I renamed the variables to clearly indicate what each variable means. I also removed the index column from the data set because it contains no valuable information that needs to be summarized.

```r
# Use pipes and kable to create a table of summary statistics
desc_df <- as.data.frame(describe(train_data))
desc_df |> dplyr::select(n, mean, median, sd, min, max) |>
  kable(digits = 2, caption = "Initital Summary Statistics")
```

Table 2: Initital Summary Statistics

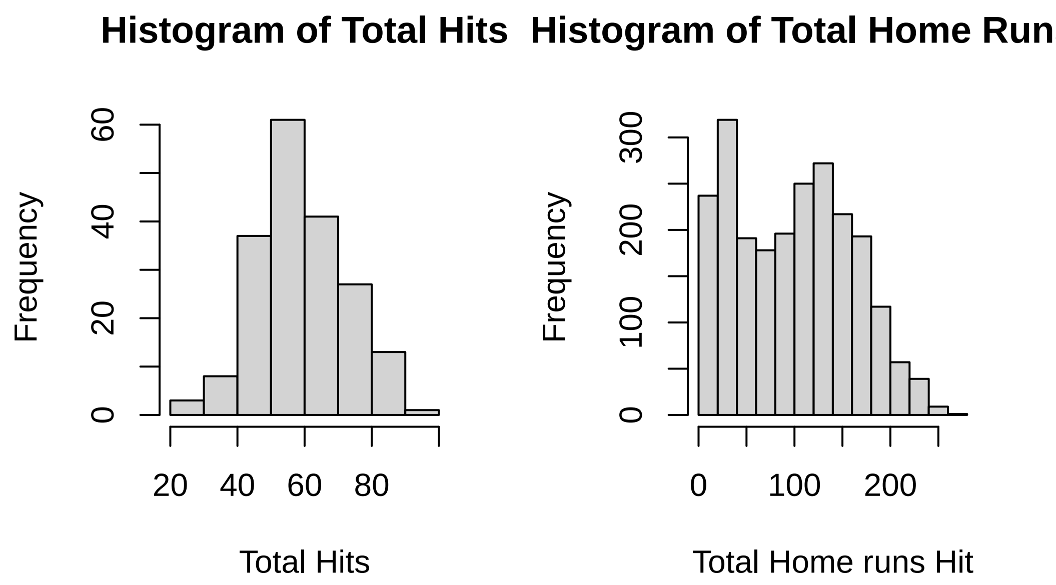
| | n | mean | median | sd | min | max |
|---|---|---|---|---|---|---|
| wins | 2276 | 80.79 | 82.0 | 15.75 | 0 | 146 |
| bat_hits | 2276 | 1469.27 | 1454.0 | 144.59 | 891 | 2554 |
| doubles | 2276 | 241.25 | 238.0 | 46.80 | 69 | 458 |
| triples | 2276 | 55.25 | 47.0 | 27.94 | 0 | 223 |
| bat_hr | 2276 | 99.61 | 102.0 | 60.55 | 0 | 264 |
| bat_walks | 2276 | 501.56 | 512.0 | 122.67 | 0 | 878 |
| bat_so | 2174 | 735.61 | 750.0 | 248.53 | 0 | 1399 |
| sb | 2145 | 124.76 | 101.0 | 87.79 | 0 | 697 |
| cs | 1504 | 52.80 | 49.0 | 22.96 | 0 | 201 |
| hbp | 191 | 59.36 | 58.0 | 12.97 | 29 | 95 |
| pitch_hits | 2276 | 1779.21 | 1518.0 | 1406.84 | 1137 | 30132 |
| pitch_hr | 2276 | 105.70 | 107.0 | 61.30 | 0 | 343 |
| pitch_walks | 2276 | 553.01 | 536.5 | 166.36 | 0 | 3645 |
| pitch_so | 2174 | 817.73 | 813.5 | 553.09 | 0 | 19278 |
| errors | 2276 | 246.48 | 159.0 | 227.77 | 65 | 1898 |
| dp | 1990 | 146.39 | 149.0 | 26.23 | 52 | 228 |

**Initial Observations:**

- Looking at the mean and median of several of the keys variables, we can see that our dependent variable `wins` is slightly skewed to the left since median > mean, while one of the main independent variables `bat_hits` his slightly skewed to the right as mean > median.
- `bat_hr` is another independent variable that I will be interested in studying its effect on wins, and it appears to be left skewed since median > mean.
- Below is a side by side comparison of the histograms representing the distribution the key independent variables `bat_hits` and `bat_hr`.

```r
# Side by side of histogram for total hits and wins
par(mfrow = c(1,2))
hist(train_data$h,
     xlab = "Total Hits",
```

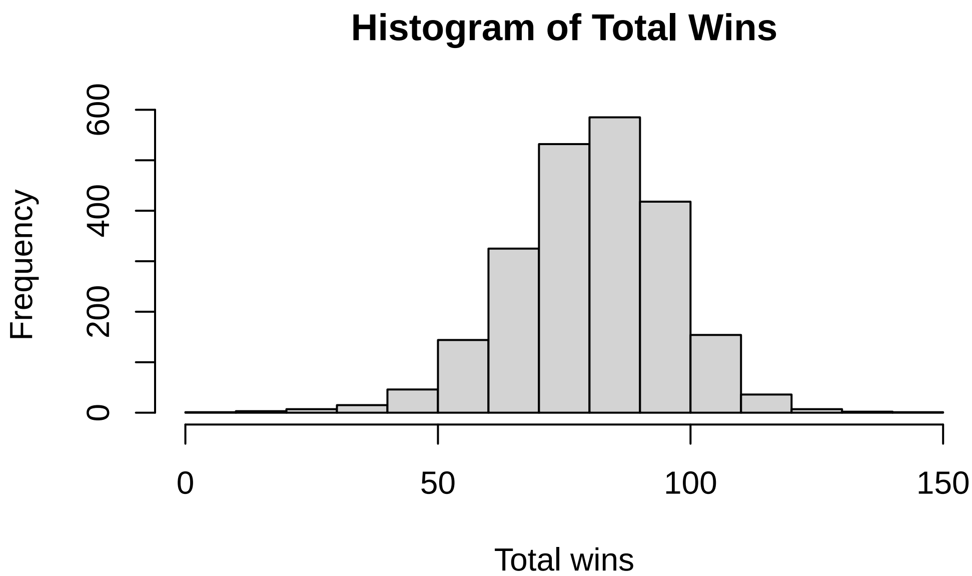
```
     main = "Histogram of Total Hits")
hist(train_data$bat_hr,
     xlab = "Total Home runs Hit",
     main = "Histogram of Total Home Runs Hit")
```




There do not appear to be any major issues with the `bat_hits` histogram other than some slight skewness. However, the histogram for `bat_hr` is very strange an appears almost bi-modal. This could be a result of the accidental values of zero that we can see in the summary statistics, so I will have to clean the data to get a more accurate depiction of home runs.

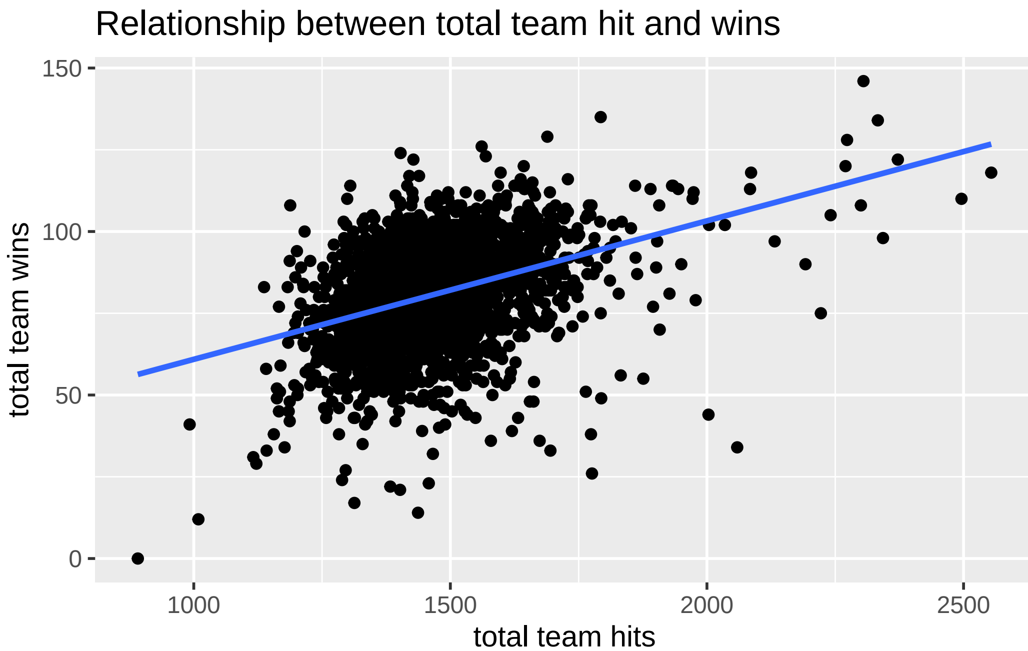
Below is a histogram of the dependent variable `wins` which confirms that it is slightly left skewed.

```
# Histogram of wins
hist(train_data$wins,
     xlab = "Total wins",
     main = "Histogram of Total Wins")
```




We can also get a glimpse of the the initial relationship between total team hits and total wins before cleaning the data. Below is a scatter plot with total hits on the x-axis and wins on the y-axis. I also included the regression line to visualize the strength on their relationship. I also calculated the correlation coefficient value of about 0.389 which indicates a positive correlation between the amount of hits a team recorded and their corresponding number of wins.

```
# Using ggplot to create a graph with regression line to show relationship of wins and hits
ggplot(data    = train_data,
       mapping = aes(x = bat_hits, y = wins)) +
  geom_point()+
  geom_smooth(method = "lm", formula = y ~ x, se = F)+
  labs(x     = "total team hits",
       y     = "total team wins",
       title = "Relationship between total team hit and wins")
```




As we can see in the graph above, and indicated by the blue regression line, total team hits and total wins appear to be positively associated. This is just a first glance at the relationship between these variables, but it gives us an understanding of what to expect after we clean the data and run our regressions.

Note that I did not include a graph showing the relationship between home runs and wins because of the unusual nature of the distribution of `bat_hr`.

**Other Important observations about the Summary Statistics**

- **Missing observations (n column)**
  - There are missing observations for both total strikeouts for and against, total times caught stealing, total stolen bases, total times hit by pitches, and total double plays.
  - In the Data Preparation section, we will visualize these missing observations and delete any variables that have a substantial number of missing values. For variables with a limited number of missing entries, we will impute those `NA` values with the median values of the corresponding variable.
- **Min and Max Columns**
  - 10 of the variables have a minimum value of zero. This is data for entire seasons, so it is highly improbable that a team had zero of any of these statistics for a whole season. It is likely that there are missing values for those variables, but instead of having `NA`, they were replaced with zeros.

- The maximum number of hits allowed and strike outs for are 30,132 and 19,278 respectively. These are values are extremely high given their mean and median values. It's likely that these values are typos by whoever entered in the data.
- We also address these problems in the Data Preparation section.

## 1.2 Correlation Table

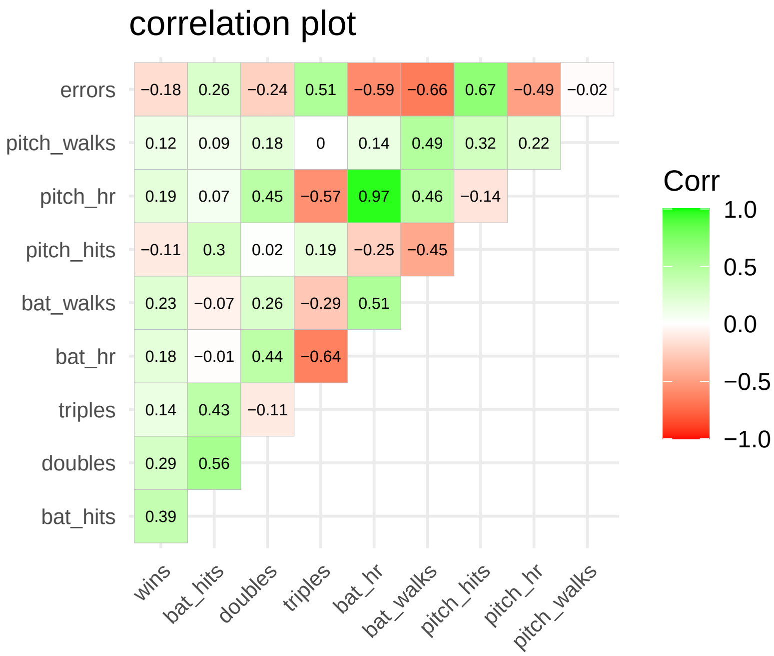
Let's know get an idea of the correlation between some of the key independent variables I will focus on and the dependent variables `TARGET_WINS` . Below is the correlation representing the strength of correlation between the observed variables.

```
library(ggcorrplot)

# Create the correlation table with specified variables
corr_tab <- cor(x = train_data[, c(1:16)])

# Use ggcorplot to create the upper correlation plot
corr_plot <- ggcorrplot(corr     = corr_tab,
                        type     = "upper",
                        method   = "square",
                        title    = "correlation plot",
                        colors   = c("red", "white", "green"),
                        lab      = T,
                        lab_size = 2,
                        insig    = "pch",
                        tl.cex   = 8,
                        digits = 2

                        )
corr_plot
```




**Correlation Table Analysis**

From the table above, we can see that only 10 of the 16 variables are included in the plot because the remaining 6 variables have missing observations. Thus, we do not have a complete depiction of the correlation between the variables and must clean the data to get an accurate illustration.

Nevertheless, we can get an initial idea of what variables appear to be correlated with wins. The green tiles represent positive correlations, the red represent negative correlations, white means no correlation, and the shade indicates the strength of the relationship.

There is a positive correlation with all of the team batting metrics and team wins which makes sense, and the strong relationship being with hits. Also, both total hits allowed and fielding errors are negatively correlated with total wins which makes sense. However, it is surprising to see that total walks allowed and home runs allowed are both positively correlated with wins. This could perhaps be a result of the uncleanliness. In the next section, we address these problems of missing and inaccurate observations in our data set by using deletion and imputation techniques.

## 2 Data Preparation

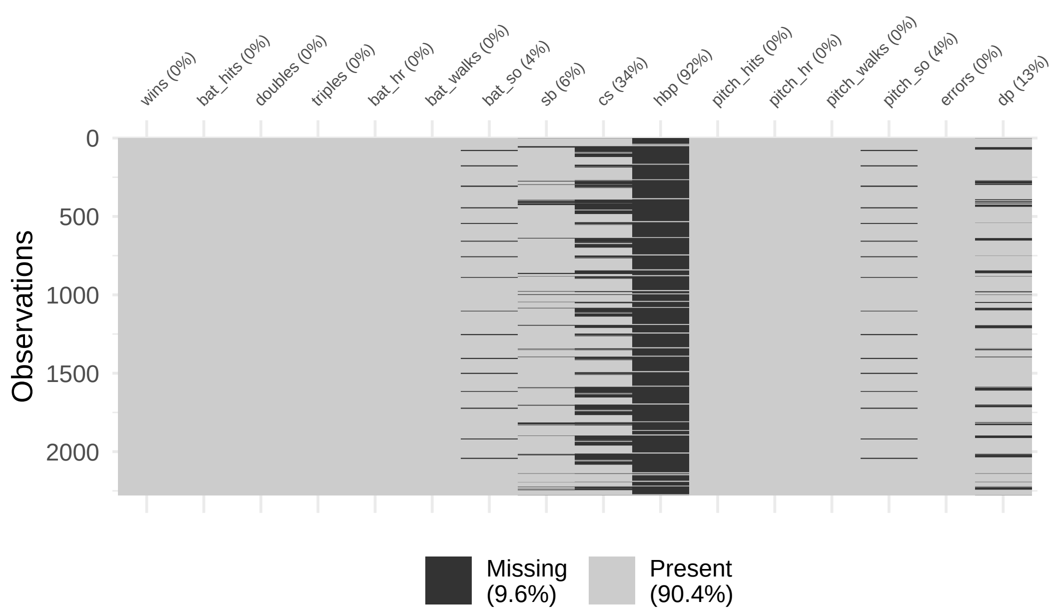
In the previous section, we identified several problems in the training data set including: missing observations, observations with 0 that should have `NA`, and inaccurately high observations. The goal of this section is to clean and prepare the data so that we can perform regressions to accurately determine the variables most associated with winning in professional baseball.

### 2.1 Visualizing Missing Values

Let's first visualize the missing observations in the data using a graph.

We can see that most of the missing observations belong to the `hbp` and `cs` variables. It also appears that their may be a pattern in when observations are missing. For example, it looks like when there is missing data for batting strike outs, there is also missing data for pitching strike outs for the same observations. There is a similar pattern for caught stealing and double plays. We can also visualize the missing observations with the percentage if values that are missing:

```
# Visually display missing values and adjust font size to fit on page
vis_miss(train_data) +
  theme(axis.text.x = element_text(size = 6))
```

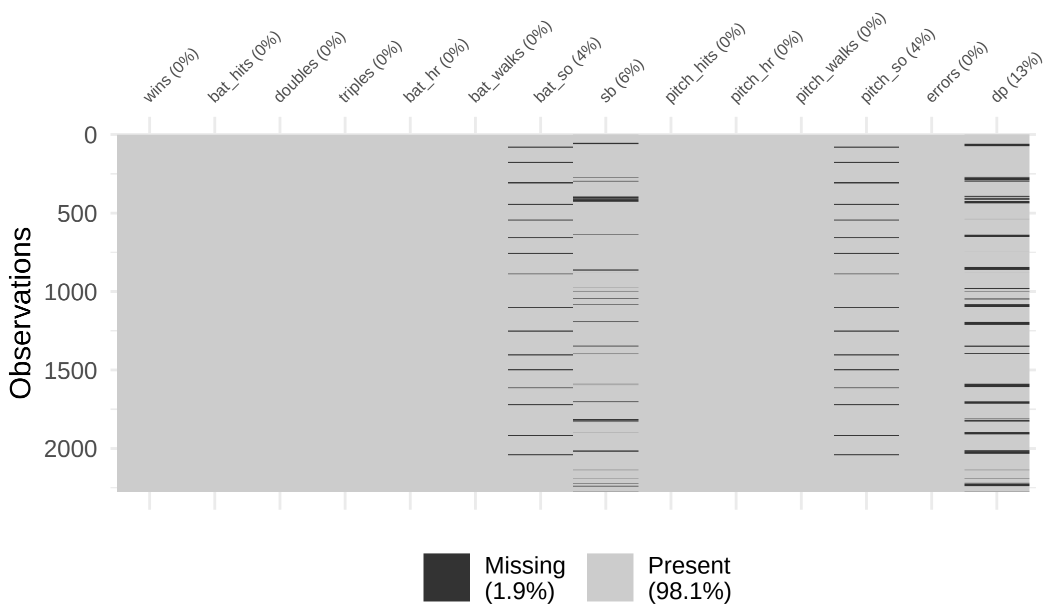


We can see that over 30% of `cs` and over 90% of `hbp` column entries are missing. These are substantial amounts of missing data that are making data analysis tricky. To solve this problem, I deleted these variables from the data set.

```
# Use pipes and select command in dplyr to delete hbp and caught stealing variables
train_data <-
train_data |> dplyr::select(-c(hbp, cs))
```

We can know see that only 1.8% of the observations are now missing compared to 9% from before.

```
# Visualize updated missing variables with small font size
vis_miss(train_data) +
  theme(axis.text.x = element_text(size = 6))
```




## 2.2 Imputing Missing Values

The next goal is to impute the remaining missing values in our data. The missing values in our data appears to be missing at random, such that the observations that have missing information appears to be random, but features that have missing information can be explained by other variables. As stated previously, there is missing data for strike outs for and strike outs against for the same observations and likewise for the fielding/base running variables like caught stealing and double plays. For this instance of missing values, the best strategies for imputing the missing data are using the K nearest neighbors imputation or MissForest methods. However, we have not learned those strategies for this class yet, and so I will instead impute the missing values with the median values of the corresponding variables. Note that this method is not the best solution, however, imputing these missing values with the median is a better alternative to using the mean because the median is more resistant to skewness and outliers.

**Median Imputation**

```
# Imputing team batting strikeout missing values with median
train_data$bat_so[is.na(train_data$bat_so)] <-
median(train_data$bat_so, na.rm = T)

# Pitching strike outs
train_data$pitch_so[is.na(train_data$pitch_so)] <-
median(train_data$pitch_so, na.rm = T)

# Stolen bases
train_data$sb[is.na(train_data$sb)] <-
median(train_data$sb, na.rm = T)

# double plays
train_data$dp[is.na(train_data$dp)] <-
median(train_data$dp, na.rm = T)
```

## 2.3 Impute Values of 0

Now I will address the values that have zero. It's highly improbable that some teams either won zero games, hit no triples, no home runs, had no walks, etc for an entire 162 game season. Therefore, we must conclude that these were meant to be missing values, but instead had zeros put in place of NA values. Looking at the summary statistics generated in section 1, we can see which variables have zero entries. Using the `which()` command, we can locate the index of where the zero entries occur in the data set to determine if any observations had multiple variables with zero entries..

```
# Identify which observations have values of zero
which(train_data$wins == 0)
which(train_data$triples == 0)
which(train_data$bat_hr== 0)
which(train_data$bat_walks== 0)
which(train_data$bat_so == 0)
```

```
which(train_data$sb == 0)
which(train_data$pitch_hr == 0)
which(train_data$pitch_walks == 0)
which(train_data$pitch_so == 0)
```

I have hidden the output because it is not very clean, but from using the command, I noticed that observations 415, 861, 1211, 1342, 2233, and 2239 have several values of zero, and therefore I have decided delete these observations from the training data set. My reasoning for deleting them is that they have too many missing values that could negatively effect our data analysis and also these are only 6 rows that will be deleted which will not be too drastic.

```
# Deleting observations with multiple zero values
train_data <- train_data[-c(415, 861, 1211, 1342, 2233, 2239), ]
```

Now, that I have deleted those problematic observations, I will now impute the values of zero with the median of their corresponding column as I did previously for the NA values. I have first converted the zero entries to NA values so that those entries can then be imputed with the accurate median values.

```
# Transforming the zero values to NA values first adn then imputing values with corresponding

#Batting home runs
train_data$bat_hr[train_data$bat_hr== 0] <- NA
train_data$bat_hr[is.na(train_data$bat_hr)] <-
  median(train_data$bat_hr, na.rm = T)

# Batting strikouts
train_data$bat_so[train_data$bat_so == 0] <- NA
train_data$bat_so[is.na(train_data$bat_so)] <-
  median(train_data$bat_so, na.rm = T)

#Stolen Bases
train_data$sb[train_data$sb == 0] <- NA
train_data$sb[is.na(train_data$sb)] <-
  median(train_data$sb, na.rm = T)

# Pitching home runs
train_data$pitch_hr[train_data$pitch_hr == 0] <- NA
train_data$pitch_hr[is.na(train_data$pitch_hr)] <-
  median(train_data$pitch_hr, na.rm = T)
```

```r
# Pitching Strike outs
train_data$pitch_so[train_data$pitch_so == 0] <- NA
train_data$pitch_so[is.na(train_data$pitch_so)] <-
  median(train_data$pitch_so, na.rm = T)
```

## 2.4 Addressing Outlier Values

Along with the NA and accidental values of zero, there are possible outliers in the data set that we must address. I have installed the `rstatix` package to use the `identify_outliers` command which returns what observations have outliers for the specified variable. As I noted in the summary statistics analysis, there are some extreme and impossible values for the `pitch_hits`, `so`, and `pitch_walks` variables that were likely mistake entries. I have imputed these values with the median for the corresponding variable.

After careful consideration for the other reasonable outliers, I have decided not to remove or impute any reasonable outliers because the interpretation of these outliers is very important. Our goal is to understand what variables are associated with winning in professional baseball, and the data on the very best and the very worst of teams is crucial to fully understand of that. General managers who are in the process of shaping their team will want to know what statistical areas the teams with the most wins focused on.

**Imputing extreme values**

```r
# hits against impossible values
train_data$pitch_hits[train_data$pitch_hits>= 2500] <- NA
train_data$pitch_hits[is.na(train_data$pitch_hits)] <-
  median(train_data$pitch_hits, na.rm=T)

# Walks against impossible values
train_data$pitch_walks[train_data$pitch_walks > 835] <- NA
train_data$pitch_walks[is.na(train_data$pitch_walks)] <-
  median(train_data$pitch_walks, na.rm=T)

#Strike outs for impossible values
train_data$pitch_so[train_data$pitch_so > 1700] <- NA
train_data$pitch_so[is.na(train_data$pitch_so)] <-
  median(train_data$pitch_so, na.rm=T)
# Errors for impossible values
train_data$errors[train_data$errors > 210] <- NA
train_data$errors[is.na(train_data$errors)] <- 210
```

## 2.5 Variable Transformations

The next step before performing regression analysis on the data is transforming any variables that violate any assumption of linear regression and creating brand new variables.

**New variables:**

- **on__base__for**: This variable adds together the total hits and walks for each team. This variable is useful because it indicates the level of success that teams have in getting on base by combining all of the important hitting statistics like singles, doubles, triples, home runs, and walks.
- **on__base__against**: Combines total hits against and walks against using the same rationale as mentioned above. This is a potential indicator of team wins because teams who are able to prevent the opponent from getting on base are more likely to have success.
- **extra__base__hits**: This variable combines total doubles, triples, and home runs hit. This will be useful in measuring the level of success that teams have in generating extra base hits. There could be multicollinearity because doubles, triples, and home runs are used to create the on base variable
- **log variables**: I created log variables for `on_base_for`, `on_base_against`, and `extra_base_hits` to try to combat some of the skewness associated with the variables. However, these variables may lead to the coefficient estimates to be much more difficult to interpret.

```
# Using pipes to and mutate command to create new variables
train_data <-
  train_data |> mutate(on_base_for = bat_hits + bat_walks,
                       extra_base_hits = doubles + triples + bat_hr,
                       on_base_against = pitch_hits + pitch_walks,
                       log_wins = log(wins),
                       log_on_base_for = log(on_base_for),
                       log_on_base_against = log(on_base_against))
```

Now that I have cleaned up all of the missing and accidental values and inserted the new variables into the training data set, we can now look at the updated summary statistics.

## 2.6 Updated Summary Statistics

Here are the updated summary statistics below after removing observations and imputing observations with the mean.

```
# Using pipes, dplyr, kable to create updated summary statistics
new_desc <- as.data.frame(describe(train_data))
new_desc |> dplyr::select(mean, median, sd, min, max) |>
  kable(digits = 2, caption = "Updated Summary Statistics") |>
  footnote("n = 2270 for all variables")
```

Table 3: Updated Summary Statistics

| | mean | median | sd | min | max |
|---|---|---|---|---|---|
| wins | 80.90 | 82.00 | 15.52 | 14.00 | 146.00 |
| bat_hits | 1469.91 | 1454.00 | 143.57 | 992.00 | 2554.00 |
| doubles | 241.41 | 238.00 | 46.54 | 113.00 | 458.00 |
| triples | 55.31 | 47.00 | 27.92 | 8.00 | 223.00 |
| bat_hr | 100.28 | 103.00 | 60.08 | 3.00 | 264.00 |
| bat_walks | 502.68 | 512.50 | 120.78 | 34.00 | 878.00 |
| bat_so | 742.74 | 750.00 | 233.07 | 66.00 | 1399.00 |
| sb | 123.54 | 101.00 | 85.43 | 14.00 | 697.00 |
| pitch_hits | 1560.76 | 1507.00 | 218.66 | 1137.00 | 2498.00 |
| pitch_hr | 106.41 | 108.00 | 60.77 | 3.00 | 343.00 |
| pitch_walks | 540.04 | 534.00 | 105.14 | 119.00 | 834.00 |
| pitch_so | 803.08 | 813.50 | 226.39 | 181.00 | 1659.00 |
| errors | 162.27 | 158.00 | 40.85 | 65.00 | 210.00 |
| dp | 146.71 | 149.00 | 24.57 | 52.00 | 228.00 |
| on_base_for | 1972.60 | 1973.00 | 179.01 | 1134.00 | 2820.00 |
| extra_base_hits | 397.00 | 396.00 | 80.69 | 162.00 | 649.00 |
| on_base_against | 2100.80 | 2048.00 | 261.86 | 1534.00 | 3304.00 |
| log_wins | 4.37 | 4.41 | 0.21 | 2.64 | 4.98 |
| log_on_base_for | 7.58 | 7.59 | 0.09 | 7.03 | 7.94 |
| log_on_base_against | 7.64 | 7.62 | 0.12 | 7.34 | 8.10 |

*Note:*

n = 2270 for all variables

There are no more missing observations, no more extreme accidental values, and we can see the new variables added to the data set. Now, it is time to build linear regression models to best predict which statistics contributes the most to winning professional baseball games.

# 3 Build Models

In this section, I built 3 multivariate linear regression models that can be used to determine the statistical factors that are significantly associated with winning in professional baseball. I will then choose one of the models that I believe to be the best and use that model to make predictions on the number of wins for teams in the evaluation data set.

## 3.1 Model Construction

### Model 1

For the first model, I predict number of wins as a function of the number of times that teams get on base, the number of times teams allow their opponents to get on base, how many extra base hits teams earn, how many stolen bases teams get, and how many errors they concede.

$$wage_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \beta_3 x_{3i} + \beta_4 x_{4i} + \epsilon_i$$

**Predictor Variables**

- $x_{1i}$: **on_base_for**
  - Total number of hits and walks that teams earn
  - This variable encompasses all positive hitting statistics such as singles, doubles, triples, home runs, and walks.
  - Combining all of these statistics allows for a simpler and cleaner model
  - In theory, getting on base increases a team's chances of scoring, resulting in a higher chance of winning games.
- $x_{2i}$: **on_base_against**
  - Total hits and walks that teams give up
  - Exactly the same as the variable above, except it measures the on base metrics for the opponents fo the observed team
  - In theory, preventing the opposing teams from getting on base leads limits the other team's chances of scoring, resulting in a higher chance of winning games.
- $x_{3i}$: **extra_base_hits**
  - Total number of doubles, triples, and home runs hit by a team

  - This variable is largely included in `on_base_for` and thus my lead to multicollinearity
  - I have still included extra base hits as a predictor of wins because it's a necessary statistic to analyze a team's ability to score and drive in runs.
  - In theory, teams who can hit more doubles, triples, and home runs as opposed to only singles and getting walks have a much better chance of scoring and winning as a result

- $x_{4i}$: **sb**
  - Total number of stolen bases
  - If teams are good at stealing bases, then this means that they do not necessarily have to earn a hit or a walk to advance a current base runner.
  - In theory, this gives teams a huge advantage of putting themselves in scoring position, making them more likely to score runs, and have a better chance to win. This makes stolen bases a possible predictor of wins.

- $x_{5i}$: **errors**
  - Total number of errors
  - Teams who frequently concede errors allow the other team to reach base without earning a hit or a walk
  - This is devastating for teams since it lets their opponents move into better scoring positions and possibly give up runs that should have been prevented
  - This is a potential valuable indicator of team success

I have run the regression below and stored the results in `reg1`. I will display the output with the other regression models at the end of this section to compared the results.

```
# First linear regression model using the lm command
reg1 <- lm(data = train_data,
           formula = wins ~ on_base_for + on_base_against + extra_base_hits +
             sb + errors)
```

#### Model 2

The second model is the same as the previous model but with the addition of two more interesting variables. Model 2 includes predicting wins as a function of total strike outs thrown and total batting strike outs, as well all of the variables included in Model 1.

$$wage_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \beta_3 x_{3i} + \beta_4 x_{4i} + \beta_5 x_{5i} + \beta_6 x_{6i} + \epsilon_i$$

**New Predictor Variables**

- $x_{5i}$: **bat_so**
  - Total number of strike outs from the batter of observed teams
  - When teams strike out less often, it means they make more contact with the baseball, increasing their chances of reaching base from either a hit or an error
  - Adding strike outs to the model can further explain the success of baseball teams
- $x_{6i}$: **pitch_so**
  - Total number of strike outs thrown by observed teams
  - Similar to the variable above, when teams strike out a batter out, this means that the fielders don't have to play defense to achieve an out.
  - Striking out batters often means the opponents make contact less often, reducing the chances of getting hit or reaching base from an error.

Notes:

There does appear to be multicollinearity between **bat_so** and **pitch_so** when I run the variance inflation factor in section 4. This is likely because there were missing values for both of these variables for many of the same observations. I imputed these observations with the medians of their respective columns, meaning that those imputed entries are not independent of one another.

Regression 2 is run below. The results are displayed in Table 4.

```
# Second linear regression model using lm command
reg2 <- lm(data    = train_data,
           formula = wins ~ on_base_for + on_base_against + extra_base_hits +
             sb + errors + bat_so + pitch_so)
```

### Model 3

The third model is generally the same as the previous model, except I have removed the variable **pitch_so** to address the multicollinearity of Model 2. This is not ideal because assessing how well teams strike out their opponents is likely important for predicting wins. However, because of the high dependence of pitching strike outs and batting strike outs as noted previously, I want to test a model that only includes one of those variables. I decided to keep **bat_so** because in theory, limiting strike outs on the offensive end is more important than striking batters out. Teams can still get an out without striking the opponent out. However, teams have no way of reaching base if they strike out, assuming the catcher does not make an error. The new estimating equation is below removing the term $\beta_6 x_{6i}$.

$$wage_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \beta_3 x_{3i} + \beta_4 x_{4i} + \beta_5 x_{5i} + \epsilon_i$$

The regression for model 3 is below. The results are in Table 4 in the next section.

```
# Third linear regression model using lm command
reg3 <- lm(data = train_data,
           formula = wins ~ on_base_for + on_base_against +
             sb + errors + extra_base_hits + bat_so)
```

### Other Models

These are some other models that I played around with. I experimented with creating regression models that included log terms are seen below. I decided not to include these in my final regression table because the coefficients are more difficult to interpret in the context of baseball compared to the level-level models I proposed above. Also, when I performed variable transformation and combined some of the variables into broader categories like **on_base_for** and **extra_base_hits**, I largely solved the problem of skewness that may have altered those regression results.

This model predicts predicts `log_wins` based on `log_on_base_for` and `log_on_base_against`.

## 3.3 Model Interpretations

Below is the regression table which displays the regression results from the three models I developed to explain wins.

```r
# Using stargazer to create regression Table
# Attempted to use tiny font size to fit table onto page
table <- stargazer(reg1, reg2, reg3,
                   type = "text",
                   title = "Updated Regression", font.size = "tiny")
```

```
Updated Regression
==========================================================================================
                                         Dependent variable:
                    ----------------------------------------------------------------------
                                                 wins
                              (1)                     (2)                        (3)
------------------------------------------------------------------------------------------
on_base_for                0.036***                0.034***                   0.032***
                           (0.003)                 (0.003)                    (0.003)

on_base_against            -0.003**               -0.005***                  -0.004***
                           (0.001)                 (0.002)                    (0.001)

extra_base_hits             0.011*                 0.018***                   0.020***
                           (0.006)                 (0.006)                    (0.006)

sb                         0.030***                0.033***                   0.033***
                           (0.004)                 (0.004)                    (0.004)

errors                    -0.033***               -0.065***                  -0.065***
                           (0.009)                 (0.012)                    (0.012)

bat_so                                            -0.012***                  -0.009***
                                                   (0.003)                    (0.002)

pitch_so                                            0.005
                                                   (0.003)

Constant                  13.027***               29.620***                  31.603***
                           (3.597)                 (5.767)                    (5.635)

------------------------------------------------------------------------------------------
Observations                2,270                   2,270                      2,270
R2                          0.226                   0.233                      0.232
Adjusted R2                 0.224                   0.231                      0.230
```

```
Residual Std. Error    13.666 (df = 2264)       13.609 (df = 2262)      13.614 (df = 2263)
F Statistic         132.350*** (df = 5; 2264) 98.297*** (df = 7; 2262) 114.172*** (df = 6; 22
==========================================================================================
Note:                                                                *p<0.1; **p<0.05; ***p<0
```

**Coefficient Interpretations**

The coefficient estimates for each of the models are presented above. In section 4, I discuss my reasoning for selecting model 1 as the best regression for predicting wins, so for simplicity, I will only interpret the coefficients for model 1. Note that the coefficients for each variable across the models are largely similar, so the coefficient interpretations do not change very much from model to model.

**Coefficient Sign**

- **Positive effect on wins:**
  - `on_base_for`, `extra_base_hits`, and `sb` all have positive coefficients which is no surprise.
  - This makes sense because getting on base, hitting extra base hits, and stealing bases are all beneficial plays for a team that increases their chances of scoring and winning games.
  - Note that `pitch_so` also has a positive coefficient in regression 2 which makes sense because striking out batters increases the chances for teams to win.
- **Negative effect on wins:**
  - `on_base_against` and `errors` all have negative coefficients which is again no surprise.
  - This makes sense because making fielding errors, striking out, and allowing the opponent to get on base are all negative plays that reduce the chances of winning games.
  - Note that `bat_so` also has a negative coefficient in regression 2 which makes sense because striking out will negatively impact a team and reduce their chance of winning.

**Magnitude**

Are the coefficient estimates large enough for teams to care about them? It is difficult to tell based on how the coefficients appear in the table above. For example, looking at the `on_base_for` coefficient, its interpretation is that holding all other variables constant, one additional time reaching base will result in an increase of 0.032 wins on average. This does not make much sense in context and does not give much insight on how improving on base frequency will actually help teams win. However, all of the data collected is on teams for a whole 162 game season. Therefore, instead of trying the analyze the effect of an increase of each variable by one unit for the entire season, we can look at the effect of increasing each variable by one unit per game. This means that we can multiply the coefficient estimates by 162 to analyze the effect of increasing the respective statistics by one unit per game. This will give us a much clearer interpretation of the coefficients and allow us to determine the effect that these variables have on winning.

- `on_base_for`: $0.036 * 162 = 5.832$.
  - For every one additional player that reaches on base per game, the number of wins will increase by just just under 6 games for a season on average, ceteris paribus.
- `on_base_against`: $-0.003 * 162 = -0.486$.
  - For every additional opponent who reaches on base, the number of wins will decrease by 0.486 games, ceteris paribus.
  - This is not significant at all, and therefore teams will likely not prioritize this statistic.
- `extra_base_hits`:$0.011 * 162 = 1.782$.
  - For every additional extra base hit per game, the expected number of wins for a team in a season will increase by just under 2 games, holding all else constant.
- `sb`: $0.03 * 162 = 4.86$.
  - For every additional stolen base per game, wins will increase by just under 5 wins per season, ceteris paribus.
- `errors`: $-0.033 * 162 = -5.346$.
  - For every additional error conceded per game, wins will decrease by over 5 games per season, holding all else constant.
- `bat_so`: $-0.012 * 162 = -1.944$.
  - For every additional strikeout per game from batters, wins will decrease on average by about 1.94 games per season, holding all else constant.
- `pitch_so`(2nd model): $0.005 * 162 = 0.81$.
  - For every additional strikeout per game thrown by pitchers, wins will increase by only about 0.81 games per season, holding all else constant.
- `constant`:

– If all other variables are zero, then on average, a team will only win about 13 games in a season.

**Statistical significance**

After determining the magnitude of each coefficient in the context of baseball success, we must now determine how confident we are in these findings using statistical significance.

**0.1 significance level**:

- `extra_base_hits` is statistically significant at the $\alpha = 0.1$ suggesting that hitting extra base hits is fairly statistically significant in predicting wins.

**0.05 significance level**:

- `on_base_against` is statistically significant at the $\alpha = 0.05$ level meaning that limiting opponents from reaching base is more statistically significantly associated with wins than hitting extra base hits.
- Note this only means that the coefficient estimate for `on_base_against` is less likely due to random chance and that we are more confident in that estimate than for `extra_base_hits`. It does not mean that `on_base_against` has a greater effect on wins. That is determined by the magnitude of the coefficient estimate

**0.01 significance level:**

- `on_base_for`
- `sb`
- `errors`
- `constant`
- The coefficient estimates for these variables are all statistically significant at the $\alpha = 0.01$ level meaning that we are most certain that these estimates are not the result of random chance.

Now that I have interpreted the regression results, I will now evaluate which of the three models is the best for predicting wins in professional baseball.

# 4 Select Models

## 4.1 Model Diagnostics

The main model diagnostics I used to evaluate the regression models are the variance inflation factor (VIF), the mean squared error, R-squared, and the F-statistic.

### Variance Inflation Factor

I have calculated the variance inflation factor for each model below and I have displayed the results side by side in table for comparison.

```
# Calculating the VIF for regression 1
vif1 <- vif(reg1)

#Model 2
vif2 <- vif(reg2)

#Model 3
vif3 <- vif(reg3)


# Create table comparing them
# I incorporated ChatGPT to create this table for me
vif_table <- data.frame(
  Model1 = round(vif1[match(unique(c(names(vif1), names(vif2), names(vif3))),
                            names(vif1))], digits = 2),
  Model2 = round(vif2[match(unique(c(names(vif1), names(vif2), names(vif3))),
                            names(vif2))], digits = 2),
  Model3 = round(vif3[match(unique(c(names(vif1), names(vif2), names(vif3))),
                            names(vif3))], digits = 2
))
```

```r
# Used Kable to create the VIF table
vif_table |> kable(caption = "Comparison of the Variance Inflation Factor")
```

Table 4: Comparison of the Variance Inflation Factor

| | Model1 | Model2 | Model3 |
|---|---|---|---|
| on_base_for | 2.50 | 3.69 | 2.99 |
| on_base_against | 1.77 | 2.21 | 1.79 |
| extra_base_hits | 2.75 | 3.15 | 3.06 |
| sb | 1.30 | 1.36 | 1.36 |
| errors | 1.79 | 2.89 | 2.89 |
| bat_so | NA | 6.42 | 2.71 |
| pitch_so | NA | 5.12 | NA |

**VIF Analysis**

- The variables for both model 1 and model 3 have low VIF values, indicating that there are no problems with multicollinearity negatively effecting the model.
- VIF is slightly higher for `on_base_for` and `extra_base_hits` because extra base hits are a way to get on base, however, the vast majority of players who reach base do so by way of singles or walks which are not extra base hits. This suggests why the multicollinearity between `on_base_for` and `extra_base_hits` is low and we can keep them both in the model.
- The VIF values for `bat_so` and `pitch_so` are 6.42 and 5.12 respectively which is very high, suggesting high multicolinearity between the two variables. In theory, this does not make sense because strike outs pitched and batting strike outs should not be correlated in reality. However, these variables had many missing values for the same observations, meaning that these median imputed entries are not independent of one another.
- Therefore, the multicollinearity present in model 2 suggests that it's coefficient estimates are unreliable, and thus, either `bat_so` or `pitch_so` should be removed for a more reliable model.
- Specifically comparing models 1 and 3, the VIF values are lower for all of the variables in Model 1, suggesting that there is less multicollinearity in Model 1 than Model 3.

### Mean Squared Error, R-Squared, and F-Statistic

Below is a side by side comparison of the mean squared error, r-squared, and f-statistic values for models 1, 2, and 3 in Table 5. I discuss the results below the table.

```r
# Mean Squared Error Table
mse_table <- data.frame(
  model1 = mean(reg1$residuals**2),
  model2 = mean(reg2$residuals**2),
  model3 = mean(reg3$residuals**2)
)

# F-statistic Table
fstat <- data.frame(
  model1 = (summary(reg1))$fstatistic[1],
  model2 = (summary(reg2))$fstatistic[1],
  model3 = (summary(reg3))$fstatistic[1]
)

# r-squared Table
r_sq <- data.frame(
  model1 = (summary(reg1))$r.squared,
  model2 = (summary(reg2))$r.squared,
  model3 = (summary(reg3))$r.squared
)

# using row bind to combine all of the tables into one big table
diagnostics <- rbind(mse_table, fstat, r_sq)
rownames(diagnostics) <- c("MSE", "f-stat", "r-squared")

diagnostics |> kable(digits = 4, caption = "Diagnostic Comparison")
```

Table 5: Diagnostic Comparison

| | model1 | model2 | model3 |
|---|---|---|---|
| MSE | 186.2579 | 184.5586 | 184.7687 |
| f-stat | 132.3503 | 98.2975 | 114.1717 |
| r-squared | 0.2262 | 0.2332 | 0.2324 |

**MSE**

- The mean squared error measures how precise our point estimates are by calculating the average of the squared residuals. We can see that the MSE for all three of the models are very similar.

- Model 2 does have the lowest MSE but this is most likely because it has the most number of variables out of the three regressions. And it is true that the MSE will always decrease when more variables are added.
- Given that Model 1 has the least amount of predictors, the MSE of Model 1 being so close to the other models is telling of the strength of this regression model.

**F-Statistic**

- The f-statistic measures the statistical significance of the model as a whole. Large values for the F-statistic indicate a better model and that there is a relationship between the independent variables and the dependent variable.
- Model 1 clearly has the highest f-statistic with a value of 132.35 and this value is statistically significant at a confidence level of $\alpha = 0.01$.
- Therefore, Model 1 has the most significant relationship between its predictor variables and wins.

**R-Squared**

- The coefficient of determination measures the percentage of the variation in the dependent variable that can be explained by the independent variables.
- Model 2 has the highest $R^2$ value and the highest adjusted $R^2$ value, although all of the R-squared values are very close together.
- For all other models, about 22-23% of the variation in wins can be explained by the independent variables.

After running our diagnostic tests on the different models, I have decided to use **Model 1** to predict wins for the evaluation data set because it has the lowest VIF values for its coefficients, suggesting no multicollinearity, it has the highest f-statistic value, and its R-squared and mean squared error values are very similar to the other two models despite having fewer variables to predict wins.

## 4.2 Cleaning Evaluation Data

I will now use Model 1 to make predictions on the evaluation data set. I have first displayed the summary statistics of the evaluation data set to determine if there are any missing values that need to be addressed before making predictions.

```r
# Create summary statistics for evaluation data using kable
eval_desc <- as.data.frame(describe(eval_data))
eval_desc |> dplyr::select(n, mean, median, min, max) |>
  kable(digits = 2, caption = "Evaluation Data Summary Statistics")
```

Table 6: Evaluation Data Summary Statistics

| | n | mean | median | min | max |
|---|---|---|---|---|---|
| INDEX | 259 | 1263.93 | 1249.0 | 9 | 2525 |
| TEAM_BATTING_H | 259 | 1469.39 | 1455.0 | 819 | 2170 |
| TEAM_BATTING_2B | 259 | 241.32 | 239.0 | 44 | 376 |
| TEAM_BATTING_3B | 259 | 55.91 | 52.0 | 14 | 155 |
| TEAM_BATTING_HR | 259 | 95.63 | 101.0 | 0 | 242 |
| TEAM_BATTING_BB | 259 | 498.96 | 509.0 | 15 | 792 |
| TEAM_BATTING_SO | 241 | 709.34 | 686.0 | 0 | 1268 |
| TEAM_BASERUN_SB | 246 | 123.70 | 92.0 | 0 | 580 |
| TEAM_BASERUN_CS | 172 | 52.32 | 49.5 | 0 | 154 |
| TEAM_BATTING_HBP | 19 | 62.37 | 62.0 | 42 | 96 |
| TEAM_PITCHING_H | 259 | 1813.46 | 1515.0 | 1155 | 22768 |
| TEAM_PITCHING_HR | 259 | 102.15 | 104.0 | 0 | 336 |
| TEAM_PITCHING_BB | 259 | 552.42 | 526.0 | 136 | 2008 |
| TEAM_PITCHING_SO | 241 | 799.67 | 745.0 | 0 | 9963 |
| TEAM_FIELDING_E | 259 | 249.75 | 163.0 | 73 | 1568 |
| TEAM_FIELDING_DP | 228 | 146.06 | 148.0 | 69 | 204 |

The evaluation data set appears to have the same problems as the training data with missing values, values of zero, and impossibly large maximum values for certain variables. Also, the evaluation data does not contain the variable transformations that I created for the training data. Therefore, I have added these new variables and I will need to impute the missing and outlier values to provide accurate predictions with Model 1.

```
# Rename the evaluation data set variables using transmute command
eval_data <- eval_data |> transmute(
                       bat_hits = TEAM_BATTING_H,
                       doubles = TEAM_BATTING_2B,
                       triples = TEAM_BATTING_3B,
                       bat_hr= TEAM_BATTING_HR,
                       bat_walks= TEAM_BATTING_BB,
                       bat_so = TEAM_BATTING_SO,
                       sb = TEAM_BASERUN_SB,
                       pitch_hits = TEAM_PITCHING_H,
                       pitch_hr = TEAM_PITCHING_HR,
                       pitch_walks = TEAM_PITCHING_BB,
                       pitch_so = TEAM_PITCHING_SO,
                       errors = TEAM_FIELDING_E,
                       dp = TEAM_FIELDING_DP)
```

### Imputing Missing Values

Below, I have imputed the missing values with the corresponding median values for batting strike outs, stolen bases, pitching strike outs, and double plays.

```
# Imputing missing values with median values

# Batting Strike outs
eval_data$bat_so[is.na(eval_data$bat_so)] <-
  median(eval_data$bat_so, na.rm = TRUE)

# Stolen bases
eval_data$sb[is.na(eval_data$sb)] <-
  median(eval_data$sb, na.rm = TRUE)

# Pitching Strike Outs
eval_data$pitch_so[is.na(eval_data$pitch_so)] <-
  median(eval_data$pitch_so, na.rm = TRUE)

#Double Plays
eval_data$dp[is.na(eval_data$dp)] <-
  median(eval_data$dp, na.rm = TRUE)
```

### Imputing Impossibly high values

Using the `which` command, I have located the entries that have opponent hits at over 10,000 hits. This is an impossible number of hits in a season as this would break down to about 60 hits per game which is beyond impossible. It is likely that they are typos where an extra digit was added on to the total number of hits.

```
# identify which observations have pitch hits above 1000
which(eval_data$pitch_hits > 10000)
```

```
[1]  92 153 185
```

Observations 92, 135, and 185 have impossible values, so I have manually changed these entries by removing one of the extra digits for each observation. There are still some very high entries for `pitch_hits` that were causing causing the model to predict some of the teams to have negative wins, so I had to impute these values as well. However, it is less clear as to whether an extra digit was unintentionally added or if the data is just wrong, so I imputed them with the median values.

```
# Manually imputing values by deleting extra digit
eval_data[92, 8]  <- 1814
eval_data[153, 8] <- 2276
eval_data[185, 8] <- 1935

# Imputing remaining high values with median
eval_data$pitch_hits[eval_data$pitch_hits > 3000] <- NA

eval_data$pitch_hits[is.na(eval_data$pitch_hits)] <-
  median(eval_data$pitch_hits, na.rm = TRUE)

# imputing high pitching strike out values
eval_data$pitch_so[eval_data$pitch_so > 5000] <- NA

eval_data$pitch_so[is.na(eval_data$pitch_so)] <-
  median(eval_data$pitch_so, na.rm = TRUE)

# Impute high errors
eval_data$errors[eval_data$errors > 700] <- NA

eval_data$errors[is.na(eval_data$errors)] <-
  median(eval_data$errors, na.rm = TRUE)
```

### Imputing zero values

There are also some entries that have values of zero which is likely a mistake given the length of the 162 game season. Using the same strategy as for the training data, I have first converted the zero values to NA values so they do not effect the true median that we will impute the values with.

```
# Convert zero values to NAs so the true median value is not disrupted and then impute values

# Batting home runs
eval_data$bat_hr[eval_data$bat_hr== 0] <- NA
eval_data$bat_hr[is.na(eval_data$bat_hr)] <-
  median(eval_data$bat_hr, na.rm = T)

# Batting strike outs
eval_data$bat_so[eval_data$bat_so == 0] <- NA
eval_data$bat_so[is.na(eval_data$bat_so)] <-
  median(eval_data$bat_so, na.rm = T)
```

```r
# Stolen Bases
eval_data$sb[eval_data$sb == 0] <- NA
eval_data$sb[is.na(eval_data$sb)] <-
  median(eval_data$sb, na.rm = T)

# Pitching home runs
eval_data$pitch_hr[eval_data$pitch_hr == 0] <- NA
eval_data$pitch_hr[is.na(eval_data$pitch_hr)] <-
  median(eval_data$pitch_hr, na.rm = T)

# Pitching strike outs
eval_data$pitch_so[eval_data$pitch_so == 0] <- NA
eval_data$pitch_so[is.na(eval_data$pitch_so)] <-
  median(eval_data$pitch_so, na.rm = T)
```

**Adding Variable Transformations to Evaluation Data**

I have added the variable transformation that I created for the training data so that I can accurately predict wins for the evaluation data.

```r
# Adding the new variables to evaluation data
eval_data <-
  eval_data |> mutate(on_base_for = bat_hits + bat_walks,
                      on_base_against = pitch_hits + pitch_walks,
                      extra_base_hits = doubles + triples + bat_hr,
                      log_on_base_for = log(on_base_for),
                      log_on_base_against = log(on_base_against))
```

Below are the updated summary statistics of the evaluation data set to verify that we can move on to win prediction with Model 1.

```r
# Create updated summary statistics table for evaluation data
eval_desc1 <- as.data.frame(describe(eval_data))
eval_desc1 |> dplyr::select(n, mean, median, min, max) |>
  kable(digits = 2, caption = "Evaluation Data Summary Statistics")
```

Table 7: Evaluation Data Summary Statistics

| | n | mean | median | min | max |
|---|---|---|---|---|---|
| bat_hits | 259 | 1469.39 | 1455.00 | 819.00 | 2170.00 |
| doubles | 259 | 241.32 | 239.00 | 44.00 | 376.00 |
| triples | 259 | 55.91 | 52.00 | 14.00 | 155.00 |
| bat_hr | 259 | 96.02 | 101.00 | 3.00 | 242.00 |
| bat_walks | 259 | 498.96 | 509.00 | 15.00 | 792.00 |
| bat_so | 259 | 713.01 | 686.00 | 44.00 | 1268.00 |
| sb | 259 | 122.47 | 92.00 | 14.00 | 580.00 |
| pitch_hits | 259 | 1581.61 | 1506.00 | 1155.00 | 2985.00 |
| pitch_hr | 259 | 102.55 | 104.00 | 7.00 | 336.00 |
| pitch_walks | 259 | 552.42 | 526.00 | 136.00 | 2008.00 |
| pitch_so | 259 | 766.03 | 745.00 | 315.00 | 1462.00 |
| errors | 259 | 206.19 | 157.00 | 73.00 | 680.00 |
| dp | 259 | 146.29 | 148.00 | 69.00 | 204.00 |
| on_base_for | 259 | 1968.35 | 1974.00 | 1017.00 | 2631.00 |
| on_base_against | 259 | 2134.03 | 2040.00 | 1513.00 | 3822.00 |
| extra_base_hits | 259 | 393.25 | 397.00 | 158.00 | 615.00 |
| log_on_base_for | 259 | 7.58 | 7.59 | 6.92 | 7.88 |
| log_on_base_against | 259 | 7.65 | 7.62 | 7.32 | 8.25 |

### 4.3 Predictions

I created predictions for the number of wins that the teams in the evaluation data set will achieve using Model 1. I appended those predictions to the evaluation data frame to be studied.

```r
# Creating win predictions using the predict command
predictions_new <- predict(reg1, newdata = eval_data)
eval_data$predicted_wins <- predictions_new
eval_data$predicted_wins
```

```
 [1] 68.44858 70.86620 76.28023 86.18913 48.71391 49.67596 76.84185
 [8] 67.04995 70.01268 72.99709 75.55111 79.36088 78.48955 79.42434
```

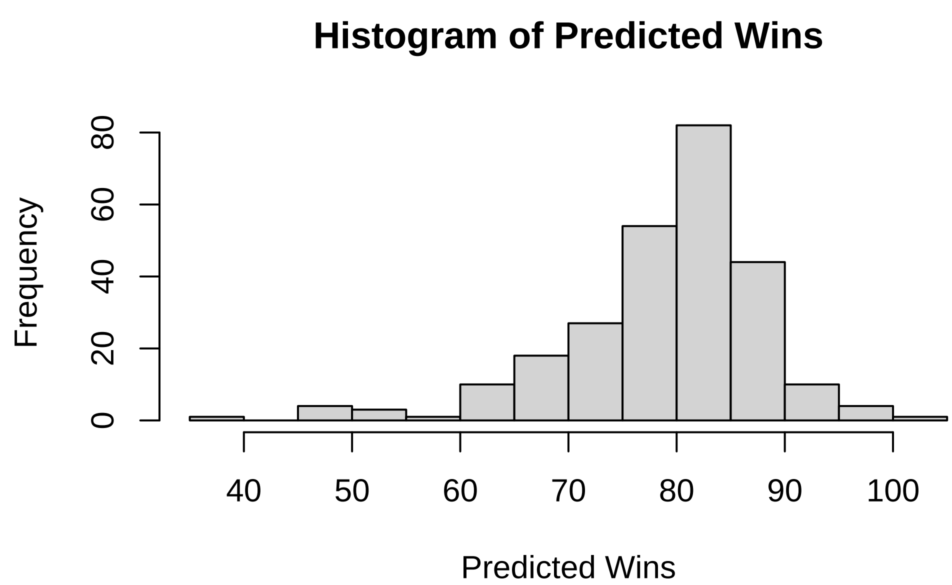
```
 [15]  77.98938  76.15447  73.74204  83.59242  64.46963  87.79008  85.35058
 [22]  87.78382  82.56362  75.57138  81.98631  86.00844  64.82041  72.43420
 [29]  85.38044  75.90192  91.41433  86.28599  88.66368  89.77564  80.75341
 [36]  82.68664  77.74139  86.01802  82.39607  84.58361  85.03395  90.04533
 [43]  49.14721  95.34422  83.94001  81.95956  83.99296  70.02842  68.09838
 [50]  74.37294  76.34220  85.44410  77.40460  72.14353  76.95268  76.68644
 [57]  82.21356  69.95448  60.92552  73.15071  82.61632  84.98647  86.14861
 [64]  83.96903  81.97764  88.49919  66.87641  72.38283  75.13249  83.71359
 [71]  82.30711  76.97192  81.58027  83.09785  79.61689  82.22277  84.90223
 [78]  80.88788  67.76446  72.56449  86.67001  85.92265  92.91538  81.91825
 [85]  84.33053  79.41190  78.93281  82.48623  87.61531  91.73379  72.90010
 [92]  93.55207  66.95351  73.33242  77.54978  76.09857  86.72923  93.58797
 [99]  88.33977  89.42693  81.86865  74.72887  83.29095  81.33160  81.65028
[106]  51.76117  61.09926  82.29371  86.20302  53.35327  84.59999  85.00807
[113]  89.92315  86.17262  79.52205  82.95187  89.07765  82.55291  77.61276
[120]  66.43697  83.00982  64.34027  63.10762  57.94980  68.59598  78.92103
[127]  83.38960  70.87478  85.49659  88.93451  86.33454  83.52139  77.39877
[134]  82.43462  84.80723  65.78720  77.39564  76.89828  88.10504  82.98151
[141]  60.88042  65.82034  91.25847  77.82540  78.50357  75.88748  79.88541
[148]  84.45780  83.97797  81.25254  81.75672  83.77949  61.34837  69.58854
[155]  79.19231  73.81640  87.26141  50.44283  76.50865  67.96662  96.12804
[162] 102.03801  89.05567  99.77485  93.97685  88.23385  83.65307  81.20355
[169]  74.69544  83.22284  83.64634  84.56285  80.30065  89.13523  81.00421
[176]  79.72976  82.88812  75.46708  76.48390  82.80566  86.23379  85.32656
[183]  86.45421  85.20446  93.24810  80.43478  82.15885  66.32141  49.13690
[190]  98.97778  64.61167  74.58130  70.44422  76.43107  79.48727  69.85401
[197]  74.87837  83.14884  79.86146  84.52111  76.31573  80.39099  76.48657
[204]  83.44436  81.00211  82.66689  82.97650  80.67625  74.76216  66.20106
[211]  89.79230  80.08467  79.22245  70.26878  72.43386  83.32191  80.66016
[218]  88.44112  77.23507  80.09399  80.51049  77.15020  85.82960  78.13151
[225]  91.31578  76.97158  79.88932  84.51137  84.27103  74.30445  74.84900
[232]  87.08449  82.97680  83.55494  79.42385  75.21259  82.62066  77.15428
[239]  77.39560  68.37345  84.00720  86.71250  84.31966  83.70761  69.44656
[246]  85.56771  77.65688  82.70483  73.85225  85.37579  83.32367  64.69485
[253]  84.29173  38.86954  71.35733  81.47277  80.51806  84.17765  78.28957
```

**Prediction Analysis**:

- All of the win predictions appear to be very normal with no indication of any insanely high or low win predictions.
- The lowest win total predicted is about 39 wins and the highest predicted number of wins is about 102 which are very reasonably win total values.

- Also, it is worth noting that I tested the predictions using models 2 and 3 and I got back some very low win totals which were very close to zero. This indicates the strength of Model 1 in predicting wins.
- Below is a histogram of the predicted wins and it is slightly left-skewed. There are a few outliers with lower win totals which is likely a result of the large amount of high error and hits allowed values that some teams had that may have still been below the imputation threshold I used.

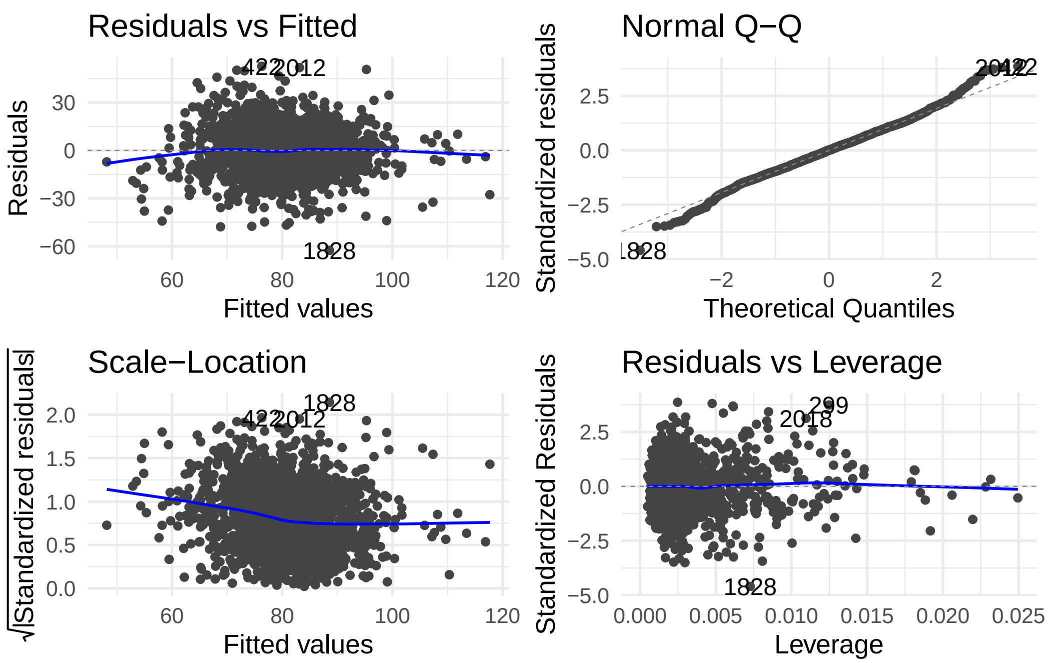
```
hist(predictions_new,
     main = "Histogram of Predicted Wins",
     xlab = "Predicted Wins")
```




**Does Model 1 obey the Gauss-Markov Assumptions?**

To determine whether Model 1 generally follows the Gauss-Markov Assumptions, I have included its four residual plots below. I used the package `ggforify` with the `autoplot` command to display the four plots all together more clearly than I could do in base R.

```
#Create residual plots using autoplot command from ggfortify library
autoplot(reg1, label.size = 3, size = 1) +
  theme_minimal(base_size = 10)
```




**Residual vs Fitted Plot**

- The residuals on this plot appear to be relatively randomly scattered around the blue regression line with no systematic curvature.
- There is also no indication of heteroscedasticity as the variance of the residuals appears to be fairly constant across all of the fitted values.
- The blue regression line is relatively linear around zero which is a good sign that this regression model follows the assumption of linearity.

**Normal Q-Q Plot**

- The residuals are relatively normally distributed around the ideal normal distribution represented by the dotted line.
- The tails do shift away from the normal distribution line slightly as a result of some likely skewness and outlier values from the data, however it is not drastic.
- I played around with trying to fix this skewness through log transformations, however this did not solve the problem and only complicated the coefficient interpretations.
- We can conclude that the normality assumption of the residuals is not heavily violated.

**Scale-Location Plot**

- The main focus of this plot is to determine if there is constant variance among the residuals.
- There does not appear any points where the variance of the residuals is greater than at other points.
- However, the residuals appear to be slightly decreasing as the fitted values increase, suggesting the possibility of heteroscedasticity.
- Therefore, the homoscedasticity assumption is likely violated, however as I noted previously, trying to apply a log or square root transformation will not solve this problem and perhaps make things worse.

**Residuals vs Leverage**

- There are a couple of points that are highlighted by the plot, such as observations 299, 1,828, and 2,018.
- However, these do not appear to have any large effect on the blue regression line as it is still linear and tracing the value of zero.

## Conclusion

Using a professional baseball training data set on many key baseball statistics, I created a 3 multivariate linear regression models with the goal of predicting win totals for professional baseball teams. I performed various diagnostics tests, such as VIF, MSE, R-squared, and f-statistic, to evaluated each models prediction performance and overall srength in the relationship between the predictors and the dependent variable wins. Ultimately, I selected **Model 1** to be the strongest model, using it to predict the win totals for professional baseball teams in the evaluation data set.

Below, I have listed the independent variables in Model 1 and I have identified which factors are the strongest predictors of wins and which are the most statistically significant:

- `on_base_for`
  - The **strongest** positive predictor of wins. For every additional time a team reaches base per game, the expected number of wins for that team will increase by just under 6 wins per season.
  - Statistically significant under $\alpha = 0.01$.
- `on_base_against`
  - A relatively weak negative predictor of wins with only on average about a half a game lost for every additional opponent to reach base per game.
  - Statistically significant under $\alpha = 0.05$.
- `extra_base_hits`
  - A moderate positive predictor of wins. Teams who record an additional extra base hit per game will win only about 2 more games on average.
  - Statistically significant under $\alpha = 0.1$.
- `sb`
  - An **unexpectedly strong** positive predictor of wins. For every additional stolen base teams record per game, their win total will increase by just under 5 wins per season.
  - Statistically significant under $\alpha = 0.01$.
- `errors`
  - The **strongest negative** predictor of wins. For every additional error a team makes, their expected win total will decrease by just over 5 games per season.

  - Statistically significant under $\alpha = 0.01$.

These regression results from Model 1 suggest that teams should focus on building rosters that excel at **getting on base**, **stealing extra bases** when they reach base, and **minimizing fielding errors** on defense. Also, all three of these indicators have the highest statistical significance, suggesting that these coefficient values are not random and are associated with winning more than any of the other variables.

I used Model 1 to make predictions in the evaluation data set and received back very promising predictions results with no team being predicted to have absurdly high or low win totals. Furthermore, the regression plots indicate no serious violations of the Gauss-Markov Assumptions for ordinary least squares regression.

I believe that Model 1 can continue to be utilized to make predictions on win totals for professional baseball teams, and I hope that these regression results provide clear insight

The regression results from Model 1 provide clear insight on which baseball statistics are most associated with winning professional baseball games, and I believe we can continue to utilize Model 1 on more data to make predictions on professional baseball win totals.