

Homework 1 Data 621

Warner Alexis, Saloua Daouki, Souleymane Doumbia, Fomba Kassoh, Lewris Mota Sanchez

2024-09-28

Introduction

In this assignment, I will be exploring, analyzing and modeing the **money ball dataset**. Each record represents a professional baseball team from the years 1871 to 2006 inclusive. Each record has the performance of the team for the given year, with all of the statistics adjusted to match the performance of a 162 game season. the purpose of this assignment is to build a multiple linear regression model on the training data to predict the number of wins for the team.

Descriptive Analysis

Variables:

INDEX Identification Variable (do not use)
TARGET_WINS Number of wins
TEAM_BATTING_H Base Hits by batters (1B,2B,3B,HR)
TEAM_BATTING_2B Doubles by batters (2B)
TEAM_BATTING_3B Triples by batters (3B)
TEAM_BATTING_HR Homeruns by batters (4B)
TEAM_BATTING_BB Walks by batters Positive
TEAM_BATTING_HBP Batters hit by pitch (get a free base)
TEAM_BATTING_SO Strikeouts by batters
TEAM_BASERUN_SB Stolen bases
TEAM_BASERUN_CS Caught stealing
TEAM_FIELDING_E Errors
TEAM_FIELDING_DP Double Plays
TEAM_PITCHING_BB Walks allowed
TEAM_PITCHING_H Hits allowed
TEAM_PITCHING_HR Homeruns allowed
TEAM_PITCHING_SO Strikeouts by pitchers

```
# Loading library
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ----- tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## vforcats   1.0.0     v stringr   1.5.1
```

```

## v ggplot2     3.5.1      v tibble      3.2.1
## v lubridate   1.9.3      v tidyverse   1.3.1
## v purrr       1.0.2
## -- Conflicts ----- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()   masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
library(ggplot2)
library(DataExplorer)
library(mice)

##
## Attaching package: 'mice'
##
## The following object is masked from 'package:stats':
##   filter
##
## The following objects are masked from 'package:base':
##   cbind, rbind
library(kableExtra)

##
## Attaching package: 'kableExtra'
##
## The following object is masked from 'package:dplyr':
##   group_rows
library(corrplot)

## corrplot 0.92 loaded
library(reshape)

##
## Attaching package: 'reshape'
##
## The following object is masked from 'package:lubridate':
##   stamp
##
## The following object is masked from 'package:dplyr':
##   rename
##
## The following objects are masked from 'package:tidyverse':
##   expand, smiths
library(reshape2)

##
## Attaching package: 'reshape2'
##

```

```

## The following objects are masked from 'package:reshape':
##
##     colsplit, melt, recast
##
## The following object is masked from 'package:tidyverse':
##
##     smiths

library(caret)

## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift

library(dplyr)
library(factoextra)

## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
library(caret) # for data splitting and pre-processing
library(stats)

```

we have two data sets: -** a training set : where most analysis will be doing -** A evaluation set Which will be used to evaluate the model

```

# load data money ball
#evaluation set use for test set
money_ball_eval <- read.csv('moneyball-evaluation-data.csv')
# training set
money_ball_train <- read.csv('moneyball-training-data.csv')
str(money_ball_train)

## 'data.frame': 2276 obs. of 17 variables:
## $ INDEX      : int 1 2 3 4 5 6 7 8 11 12 ...
## $ 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 ...
## $ 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 ...

introduce(money_ball_train)

## rows columns discrete_columns continuous_columns all_missing_columns

```

```

## 1 2276      17          0          17          0
##   total_missing_values complete_rows total_observations memory_usage
## 1           3478           191          38692        159280

# Data Description
money_ball_train %>%
  summary() %>%
  kable() %>% kable_styling() %>% kable_classic(full_width = F, html_font = "Cambria")

```

INDEX	TARGET_WINS	TEAM_BATTING_H	TEAM_BATTING_2B	TEAM_BATTING_3B	TH
Min. : 1.0	Min. : 0.00	Min. : 891	Min. : 69.0	Min. : 0.00	Min. : 0.00
1st Qu.: 630.8	1st Qu.: 71.00	1st Qu.: 1383	1st Qu.: 208.0	1st Qu.: 34.00	1st Qu.: 34.00
Median :1270.5	Median : 82.00	Median : 1454	Median : 238.0	Median : 47.00	Median : 47.00
Mean :1268.5	Mean : 80.79	Mean : 1469	Mean : 241.2	Mean : 55.25	Mean : 55.25
3rd Qu.:1915.5	3rd Qu.: 92.00	3rd Qu.: 1537	3rd Qu.: 273.0	3rd Qu.: 72.00	3rd Qu.: 72.00
Max. :2535.0	Max. :146.00	Max. :2554	Max. :458.0	Max. :223.00	Max. :223.00
NA	NA	NA	NA	NA	NA

```
str(money_ball_train)
```

```

## 'data.frame': 2276 obs. of 17 variables:
## $ INDEX      : int 1 2 3 4 5 6 7 8 11 12 ...
## $ 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 ...
## $ 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 ...

```

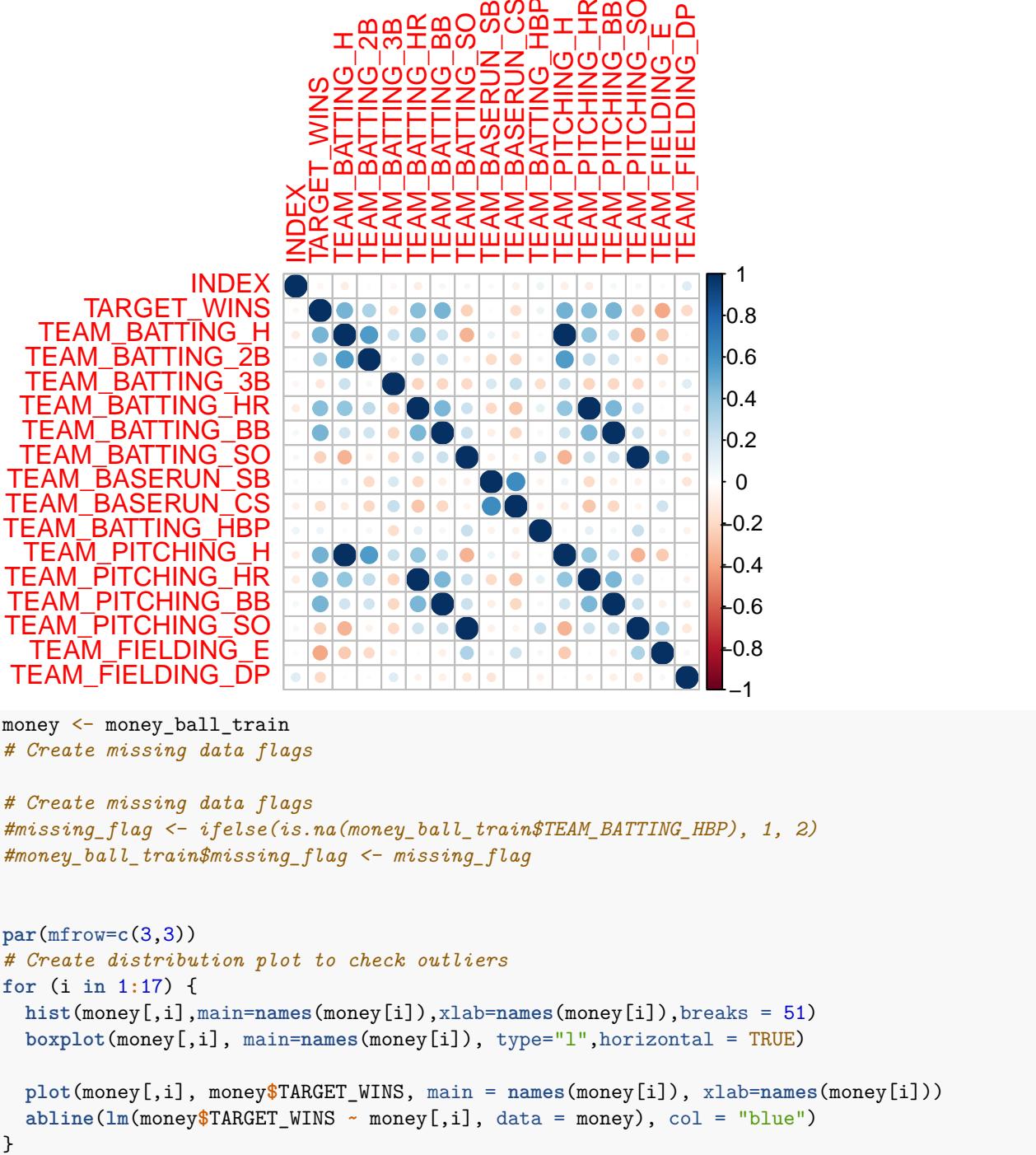
Data Preparation

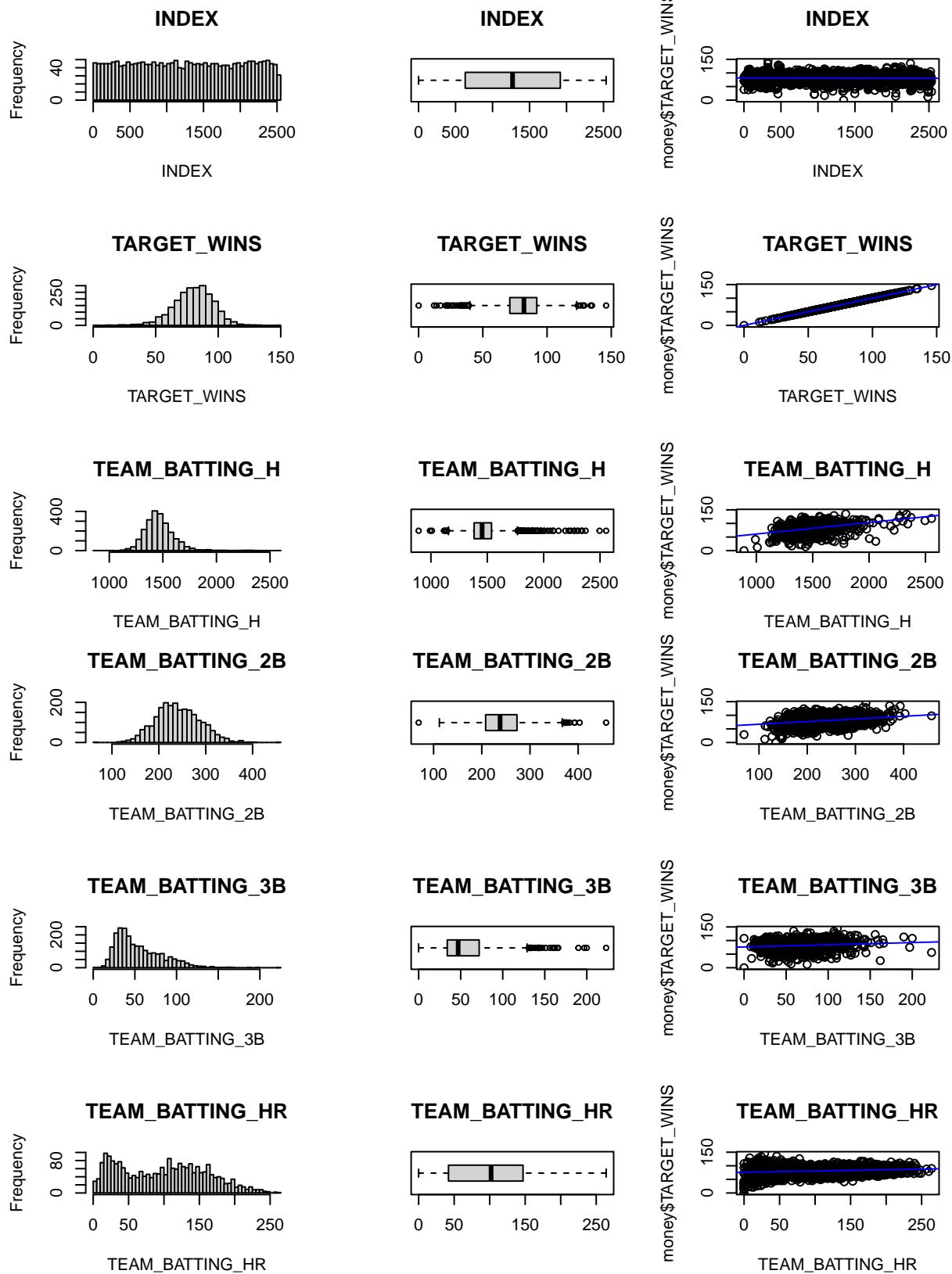
We are going to check distribution for all the columns and outliers that are present in the data set. The Chart below shows the percentage accounted for missing values in the data set. We notice that TEAM_BATTING_HBP has the highest number of missing values but, the mean, the median, the max are around the same range which mean that column is skewed centerely.

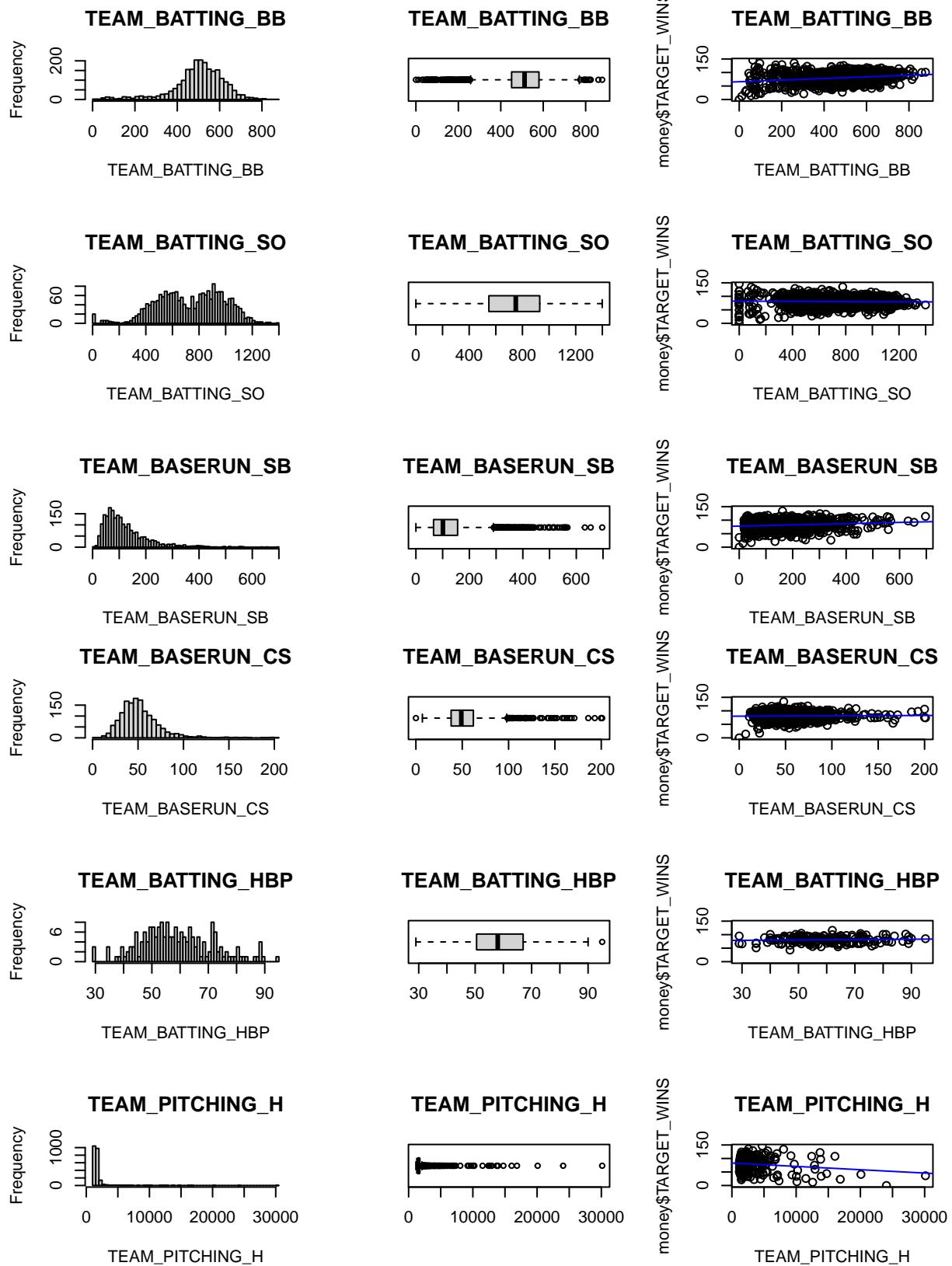
```

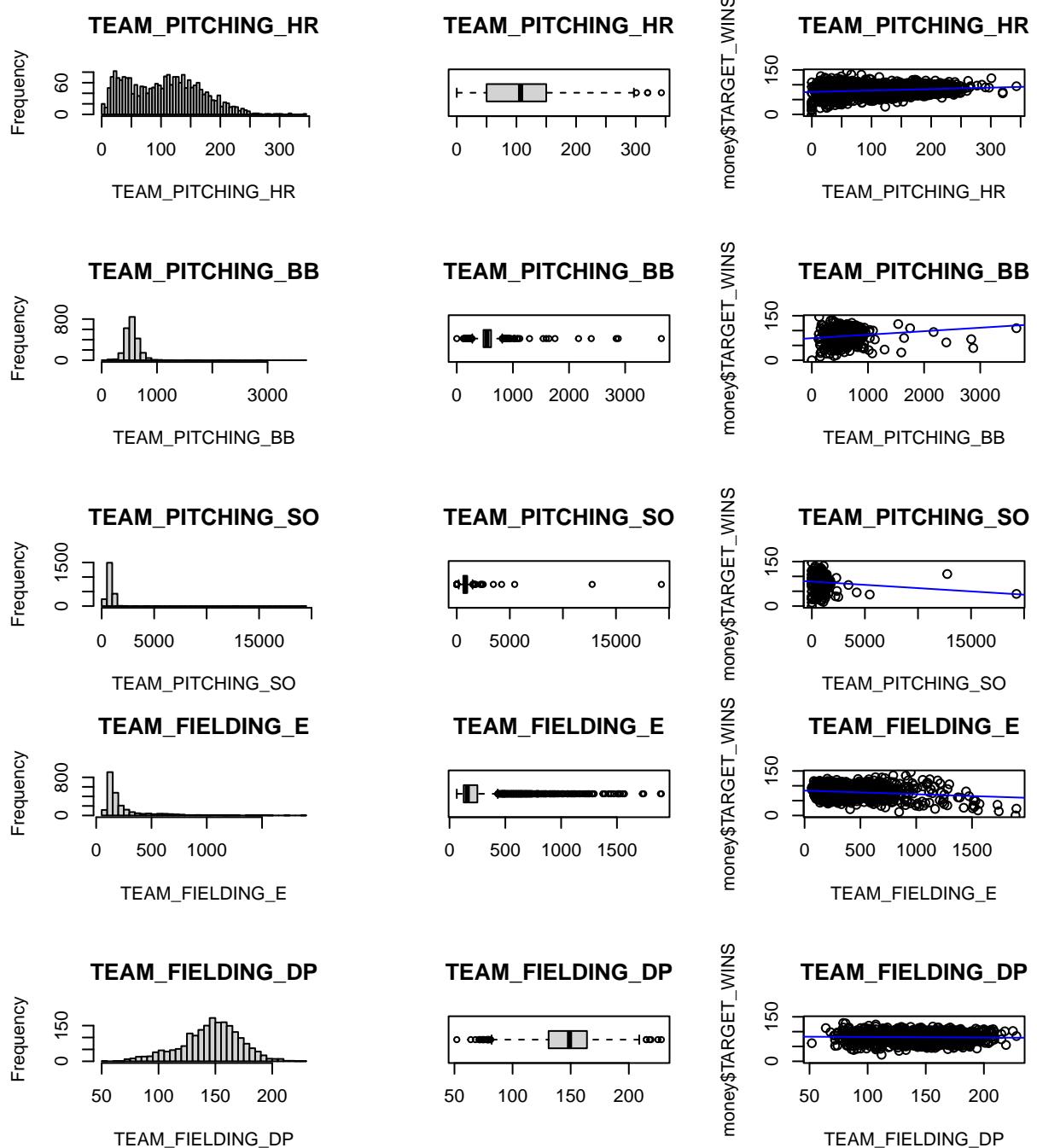
# Correlation Plot
cor_matrix <- cor(money_ball_train, use = 'complete.obs')
corrplot(cor_matrix, method = 'circle')

```









The relationship between the target variable and the predictors is not particularly strong, but there are statistically significant relationships with certain variables. The `calculate_correlations_with_pvalues` function computes the correlation coefficients and p-values between a given target variable and all other predictor variables in a dataset. It first validates the input to ensure that the target variable is present and that the data contains no missing values. For each predictor, the function performs a correlation test using `cor.test()`, which returns both the correlation coefficient and the corresponding p-value. These results are stored in a data frame, with both the correlation and p-value rounded to 10 decimal places for precision. This function offers a clear, organized output, making it easy for users to evaluate the strength and statistical significance of the relationships between the target variable and the predictors.

```

calculate_correlations_with_pvalues <- function(data, target_col) {
  # Check if target_col is a character string
  if (!is.character(target_col) || length(target_col) != 1) {
    stop("target_col must be a single character string.")
  }

  # Ensure the target column exists in the data
  if (!target_col %in% names(data)) {
    stop("Target column not found in the dataframe.")
  }

  # Remove rows with missing values
  data_complete <- data[complete.cases(data), ]

  # Initialize a results data frame
  results <- data.frame(Predictor = character(), Correlation = numeric(), PValue = numeric(), stringsAsFactors = FALSE)

  # Loop through each predictor variable
  for (predictor in names(data_complete)[names(data_complete) != target_col]) {
    # Perform the correlation test
    test_result <- cor.test(data_complete[[predictor]], data_complete[[target_col]])

    # Store the rounded results to 10 decimal places
    results <- rbind(results, data.frame(Predictor = predictor,
                                          Correlation = round(test_result$estimate, 10),
                                          PValue = round(test_result$p.value, 10)))
  }

  return(results)
}

# Example usage
correlation_results <- calculate_correlations_with_pvalues(money_ball_train, "TARGET_WINS")

# View the results
print(correlation_results)

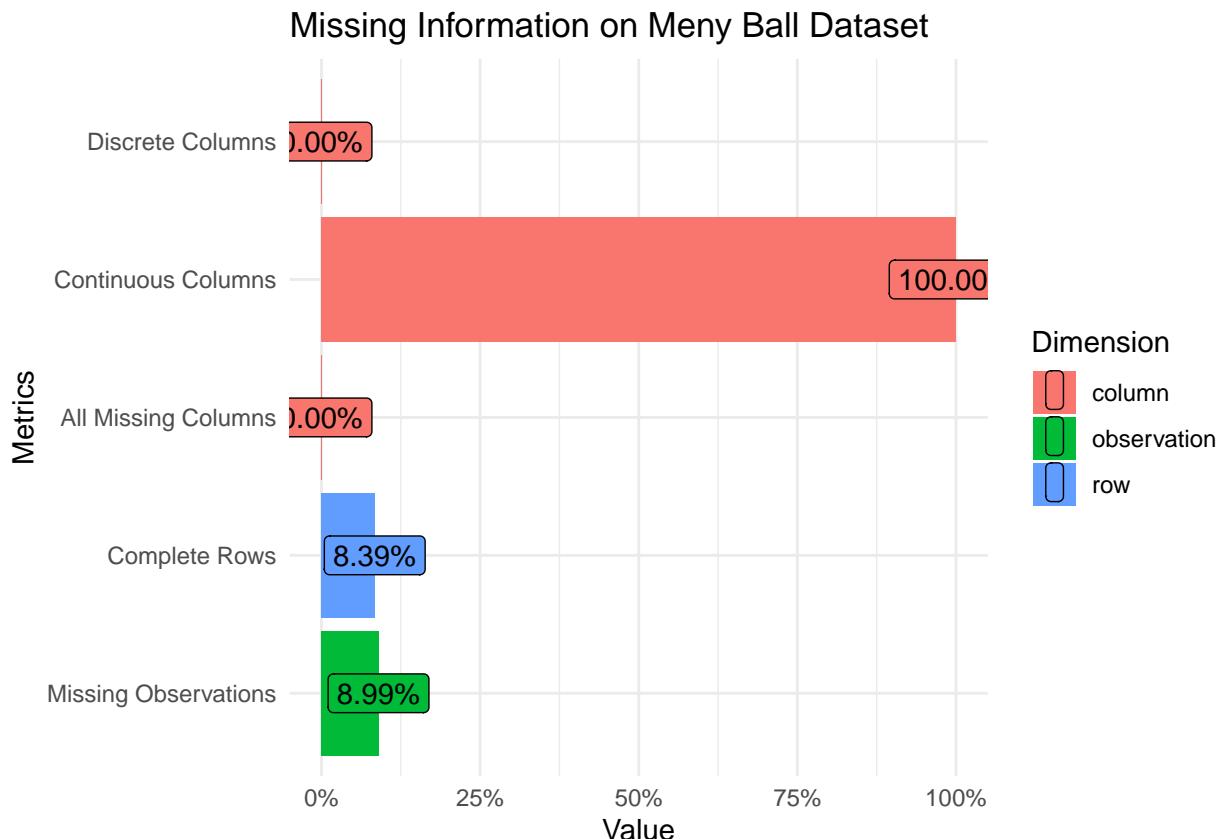
##             Predictor Correlation      PValue
## cor           INDEX -0.04895047 0.5012836479
## cor1          TEAM_BATTING_H  0.46994665 0.0000000000
## cor2          TEAM_BATTING_2B  0.31298400 0.0000104138
## cor3          TEAM_BATTING_3B -0.12434586 0.0865523694
## cor4          TEAM_BATTING_HR  0.42241683 0.0000000012
## cor5          TEAM_BATTING_BB  0.46868793 0.0000000000
## cor6          TEAM_BATTING_SO -0.22889273 0.0014476066
## cor7          TEAM_BASERUN_SB  0.01483639 0.8385814774
## cor8          TEAM_BASERUN_CS -0.17875598 0.0133534492
## cor9          TEAM_BATTING_HBP 0.07350424 0.3122327101
## cor10         TEAM_PITCHING_H  0.47123431 0.0000000000
## cor11         TEAM_PITCHING_HR 0.42246683 0.0000000011
## cor12         TEAM_PITCHING_BB 0.46839882 0.0000000000
## cor13         TEAM_PITCHING_SO -0.22936481 0.0014142043
## cor14         TEAM_FIELDING_E -0.38668800 0.0000000329

```

```
## cor15 TEAM_FIELDING_DP -0.19586601 0.0066174528
```

Now let's dive into the Missing values.

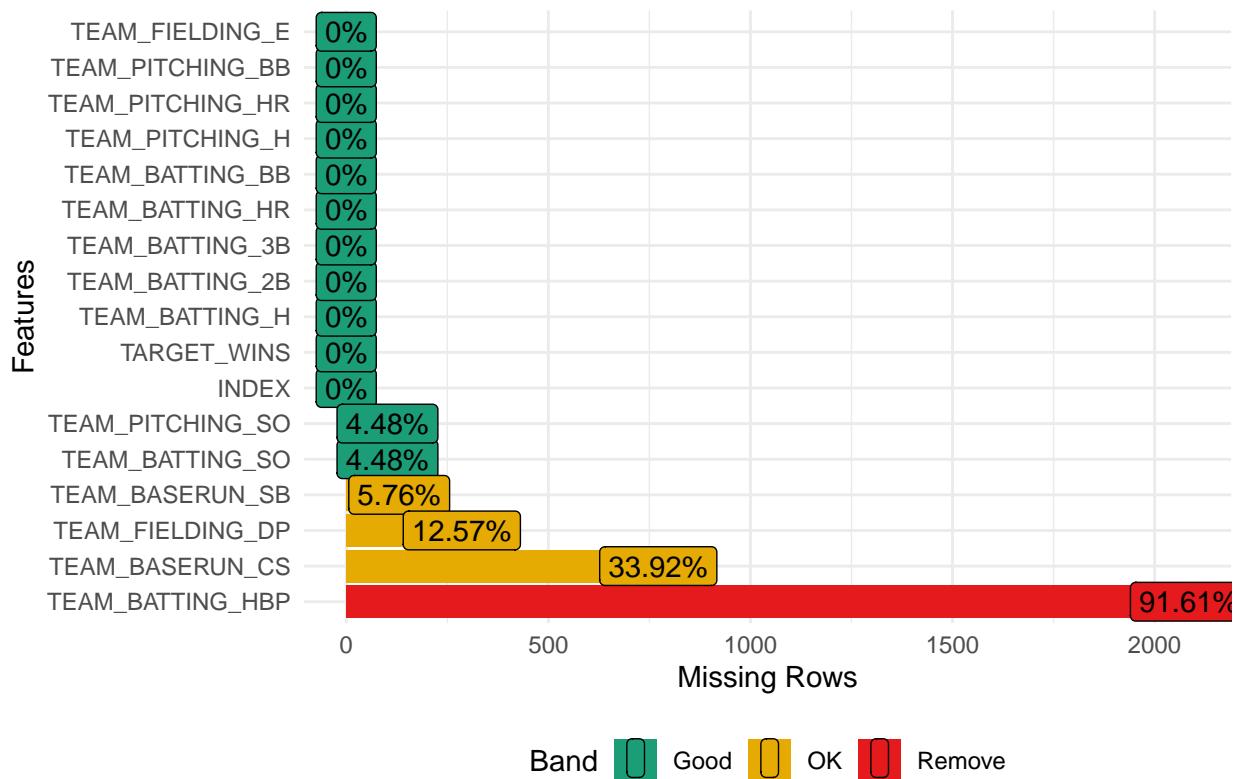
```
#  
plot_intro(money_ball_train, title = 'Missing Information on Money Ball Dataset',  
           ggtheme = theme_minimal())
```



```
# Plot missing volume in Column
```

```
plot_missing(money_ball_train, title = 'Information about Missing Value in money ball dataset', ggtheme =
```

Information about Missing Value in money ball dataset



we have discovered that there are some observations that are missing data. These column names will need imputation after analysis. We have about 9% of the data missing that spread out to > TEAM_PITCHING_SO 4.48% > TEAM_BATTING_SO 4.48% > TEAM_BASERUN_SB 5.76% > TEAM_BASERUN_CS 12.57% > TEAM_BATTING_HBP 91.61%

Why Use Predictive Imputation?

Preserves Relationships: Predictive imputation uses the relationships between variables to estimate missing values, which can lead to more accurate and reliable data sets.

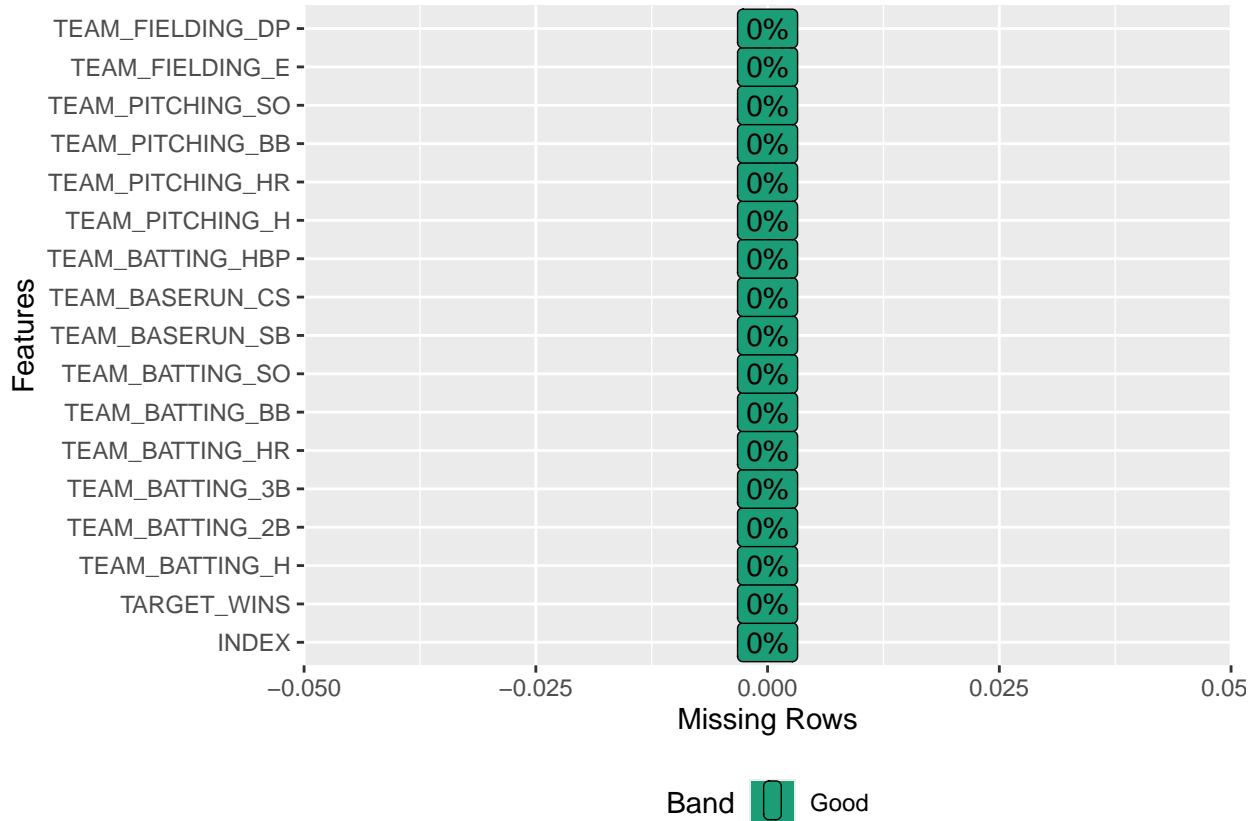
Handles Different Types of Missing Data: It can handle data that is Missing Completely at Random (MCAR), Missing at Random (MAR), and even some cases of Missing Not at Random (MNAR). **Maintains Variability:** Unlike simple imputation methods (mean, median), predictive imputation maintains the natural variability in the data.

```
# re-assign moneyball
money <- money_ball_train
# create data set with with predictive imputable
# Compute multiple imputation
data1 <- mice(money, method = 'pmm', m=5)

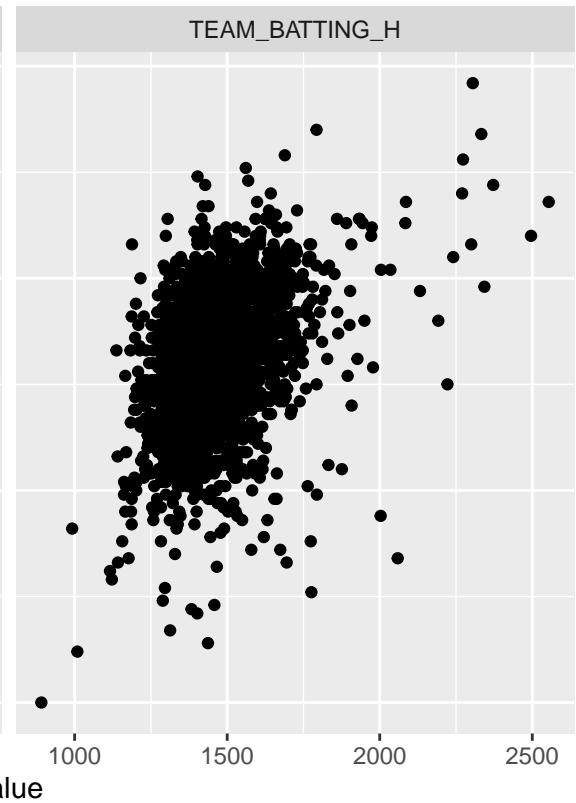
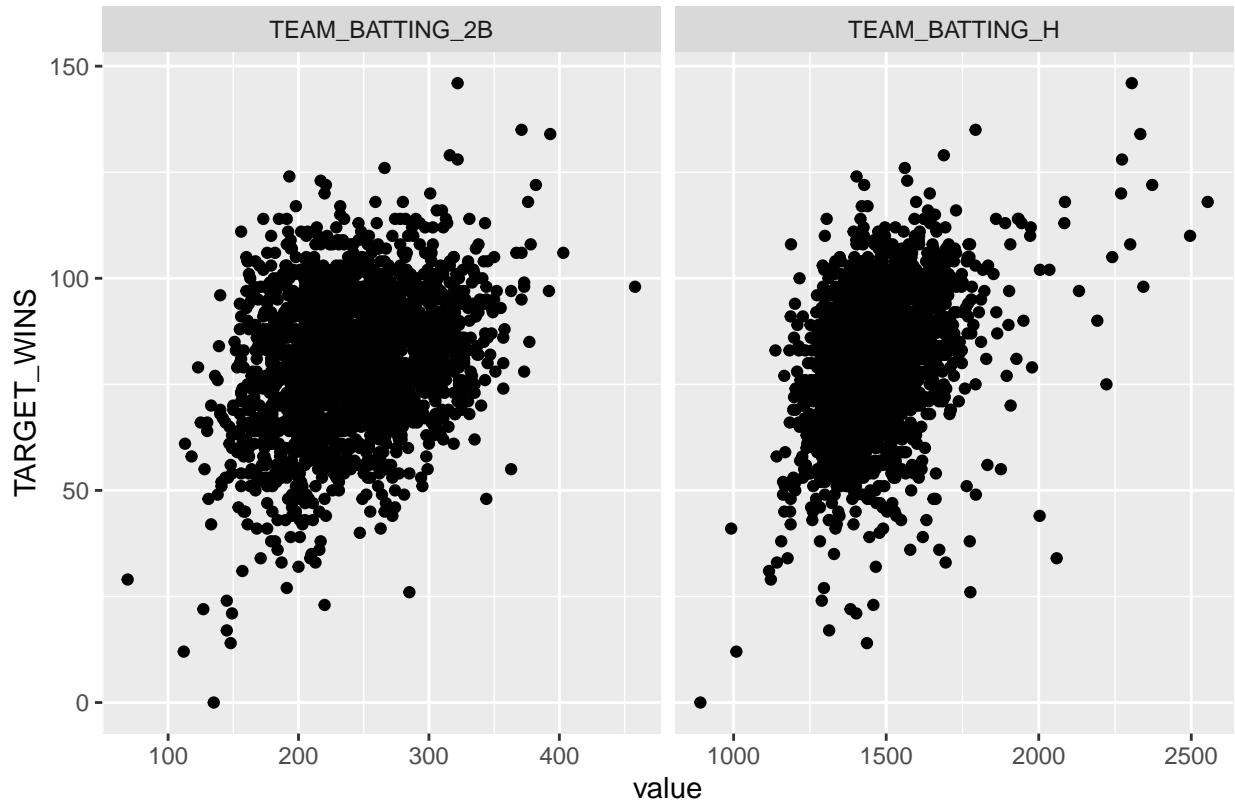
##
## iter imp variable
## 1 1 TEAM_BATTING_SO TEAM_BASERUN_SB TEAM_BASERUN_CS TEAM_BATTING_HBP TEAM_PITCHING_SO TEAM_
## 1 2 TEAM_BATTING_SO TEAM_BASERUN_SB TEAM_BASERUN_CS TEAM_BATTING_HBP TEAM_PITCHING_SO TEAM_
## 1 3 TEAM_BATTING_SO TEAM_BASERUN_SB TEAM_BASERUN_CS TEAM_BATTING_HBP TEAM_PITCHING_SO TEAM_
## 1 4 TEAM_BATTING_SO TEAM_BASERUN_SB TEAM_BASERUN_CS TEAM_BATTING_HBP TEAM_PITCHING_SO TEAM_
## 1 5 TEAM_BATTING_SO TEAM_BASERUN_SB TEAM_BASERUN_CS TEAM_BATTING_HBP TEAM_PITCHING_SO TEAM_
## 2 1 TEAM_BATTING_SO TEAM_BASERUN_SB TEAM_BASERUN_CS TEAM_BATTING_HBP TEAM_PITCHING_SO TEAM_
## 2 2 TEAM_BATTING_SO TEAM_BASERUN_SB TEAM_BASERUN_CS TEAM_BATTING_HBP TEAM_PITCHING_SO TEAM_
```



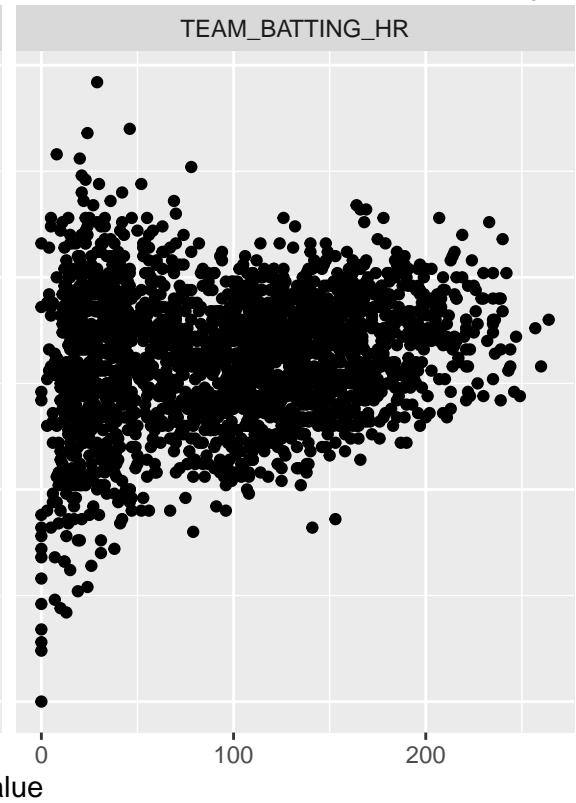
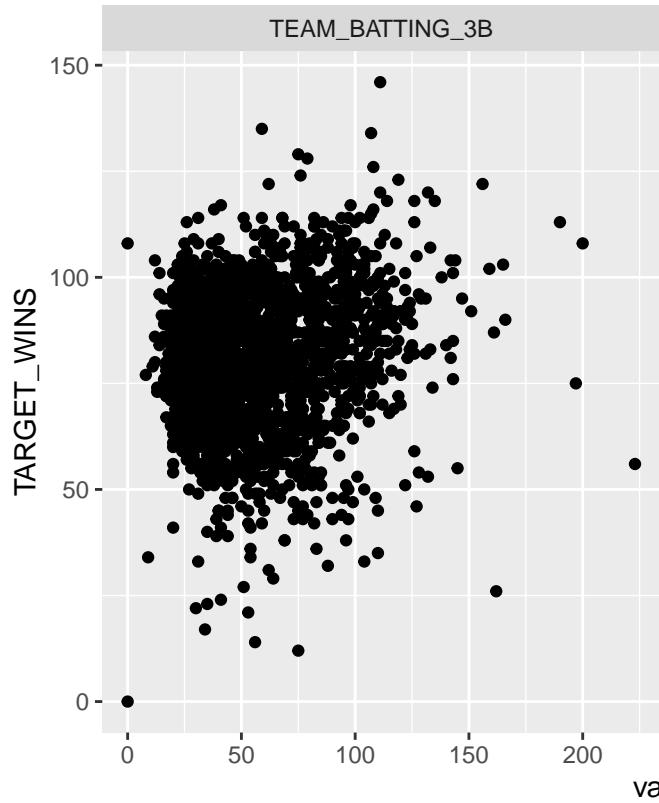
```
money_ball_eval <- complete(money_ball_eval)
# View missing value distribution
plot_missing(completed_data)
```



```
plot_scatterplot(money[,-c(1,11)], by="TARGET_WINS", nrow = 1L, ncol = 2L)
```



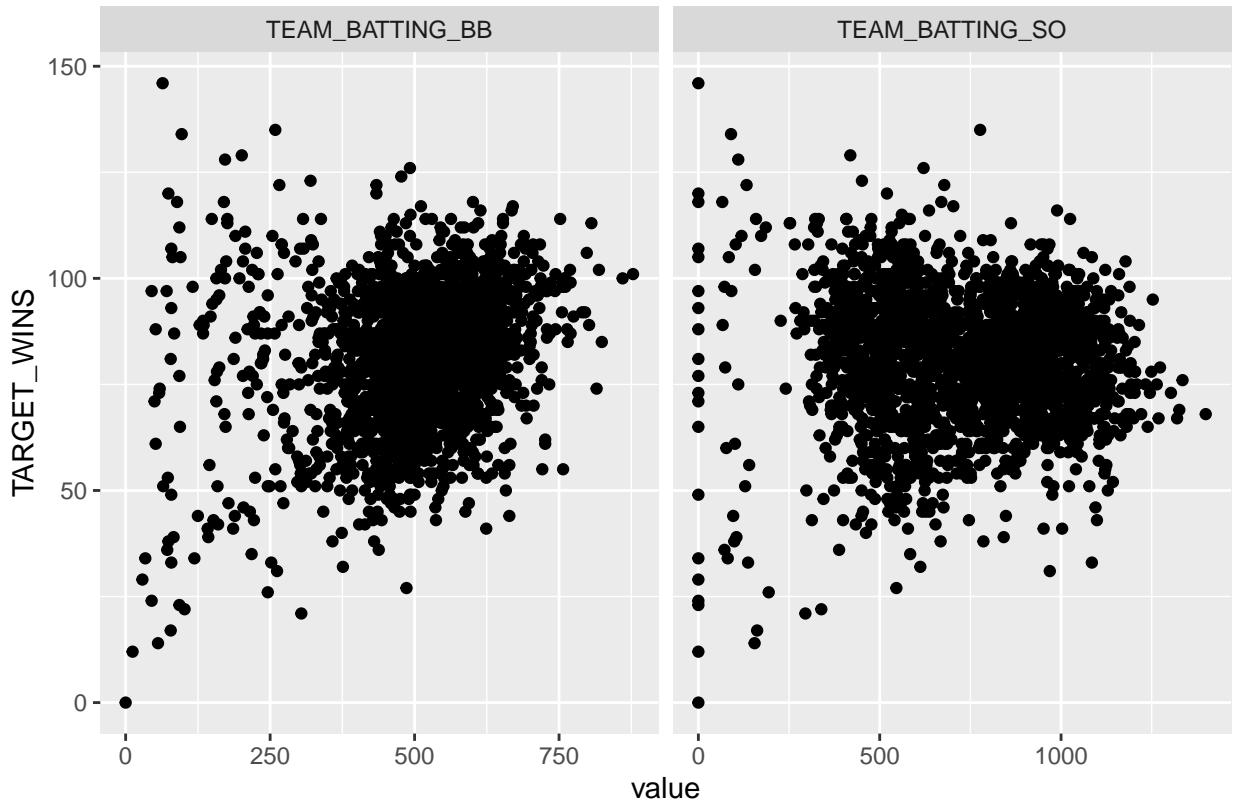
Page 1



Page 2

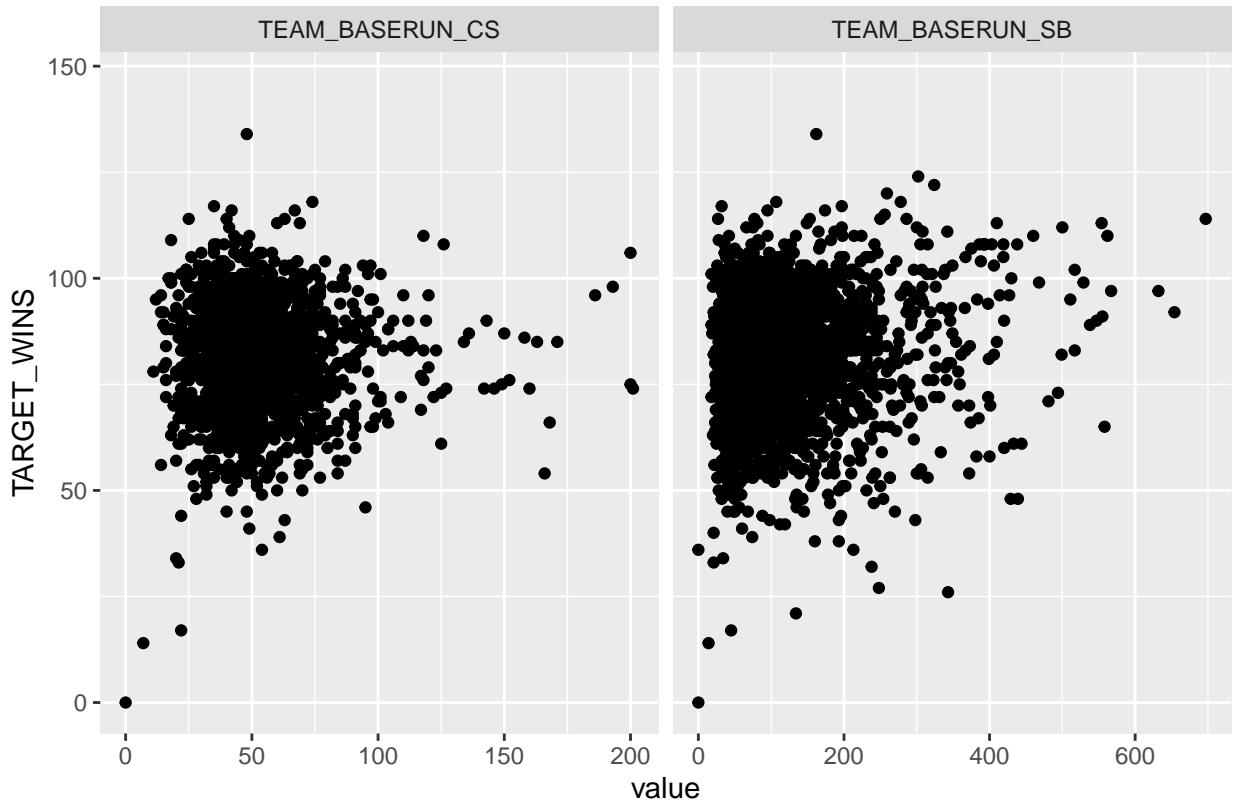
Warning: Removed 102 rows containing missing values or values outside the scale range

```
## (`geom_point()`).
```

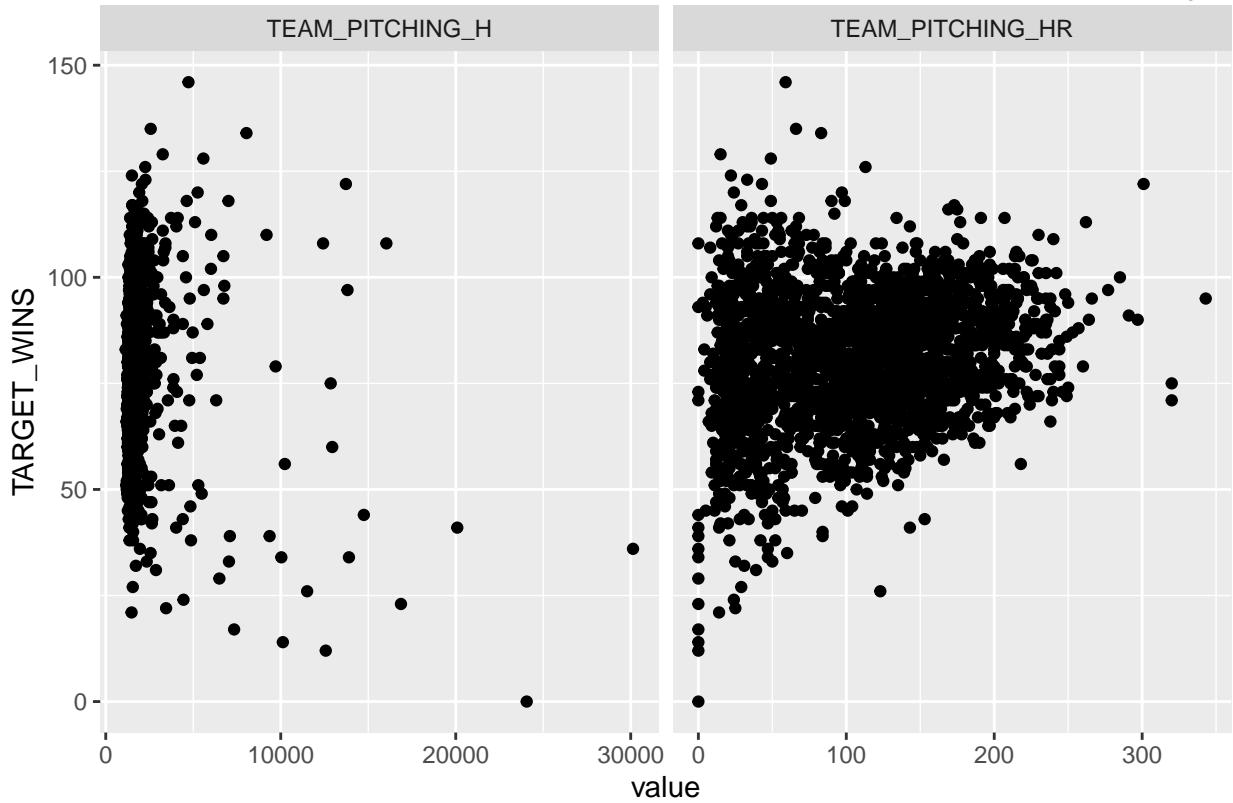


Page 3

```
## Warning: Removed 903 rows containing missing values or values outside the scale range
## (`geom_point()`).
```



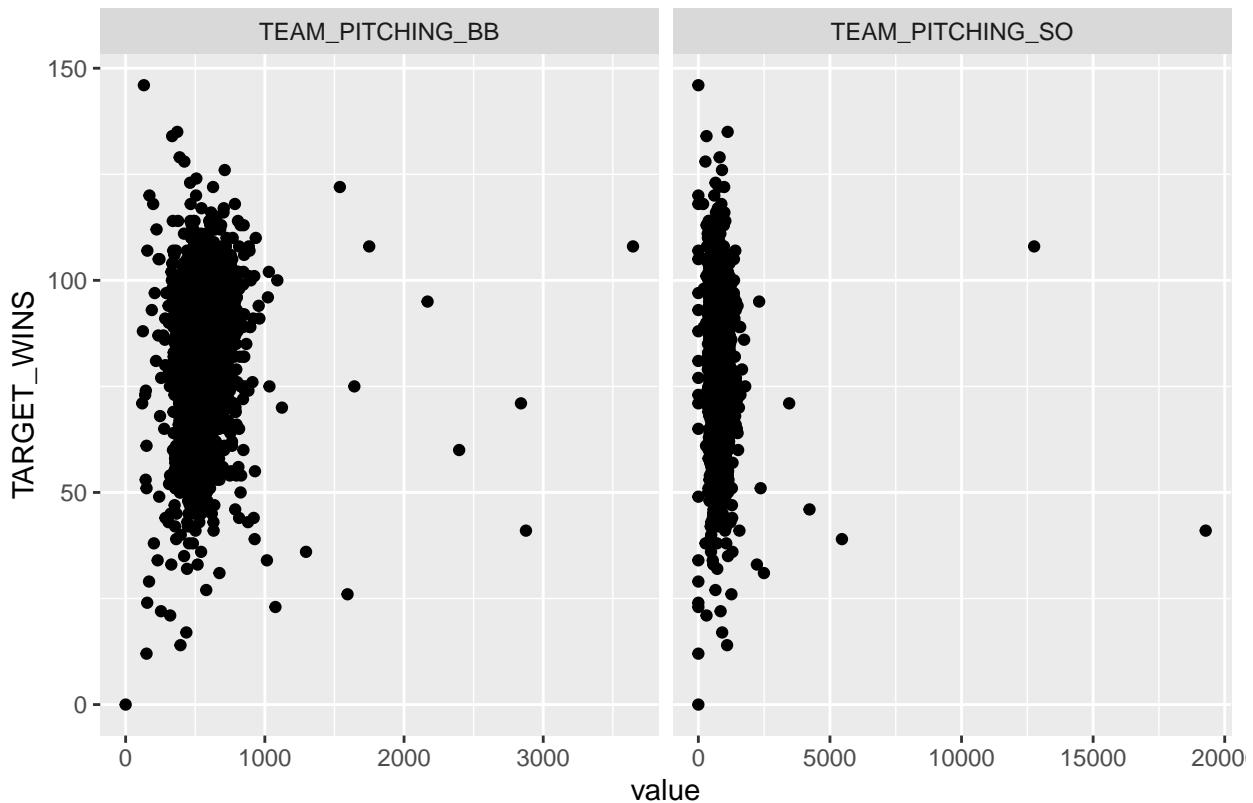
Page 4



Page 5

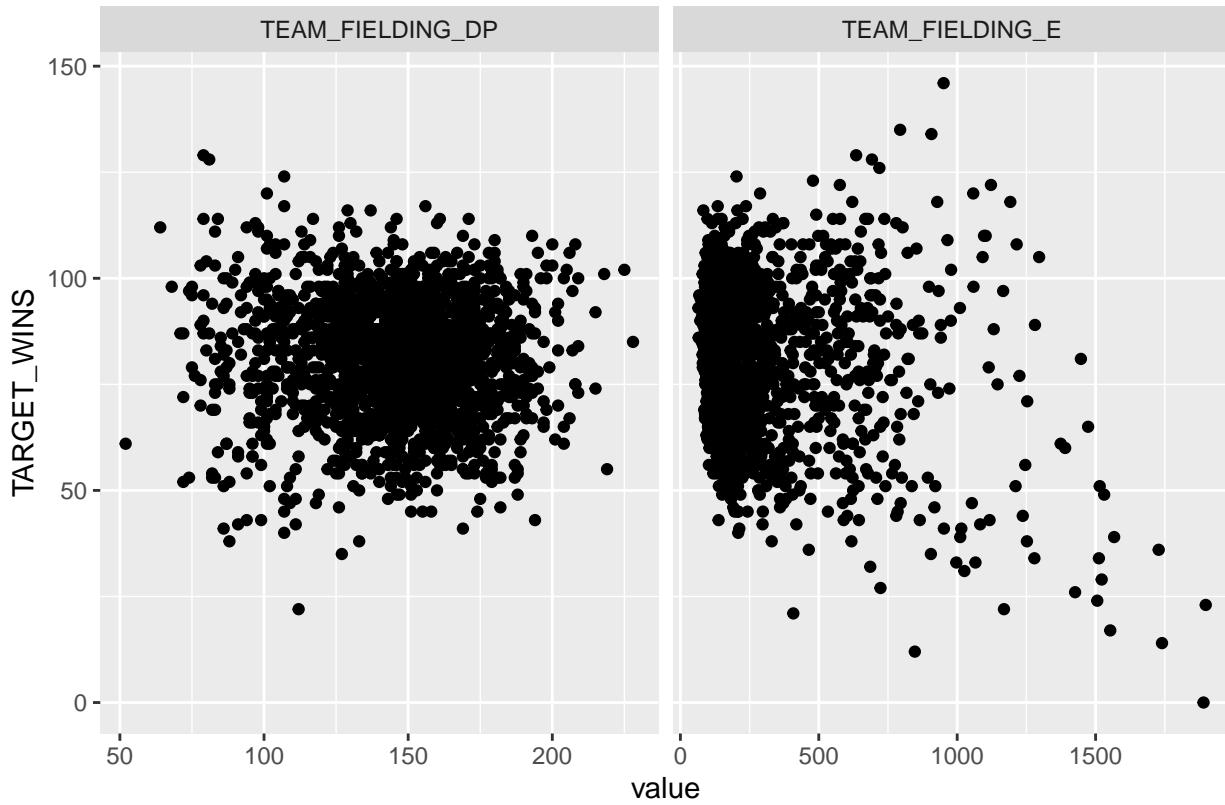
Warning: Removed 102 rows containing missing values or values outside the scale range

```
## (`geom_point()`).
```



Page 6

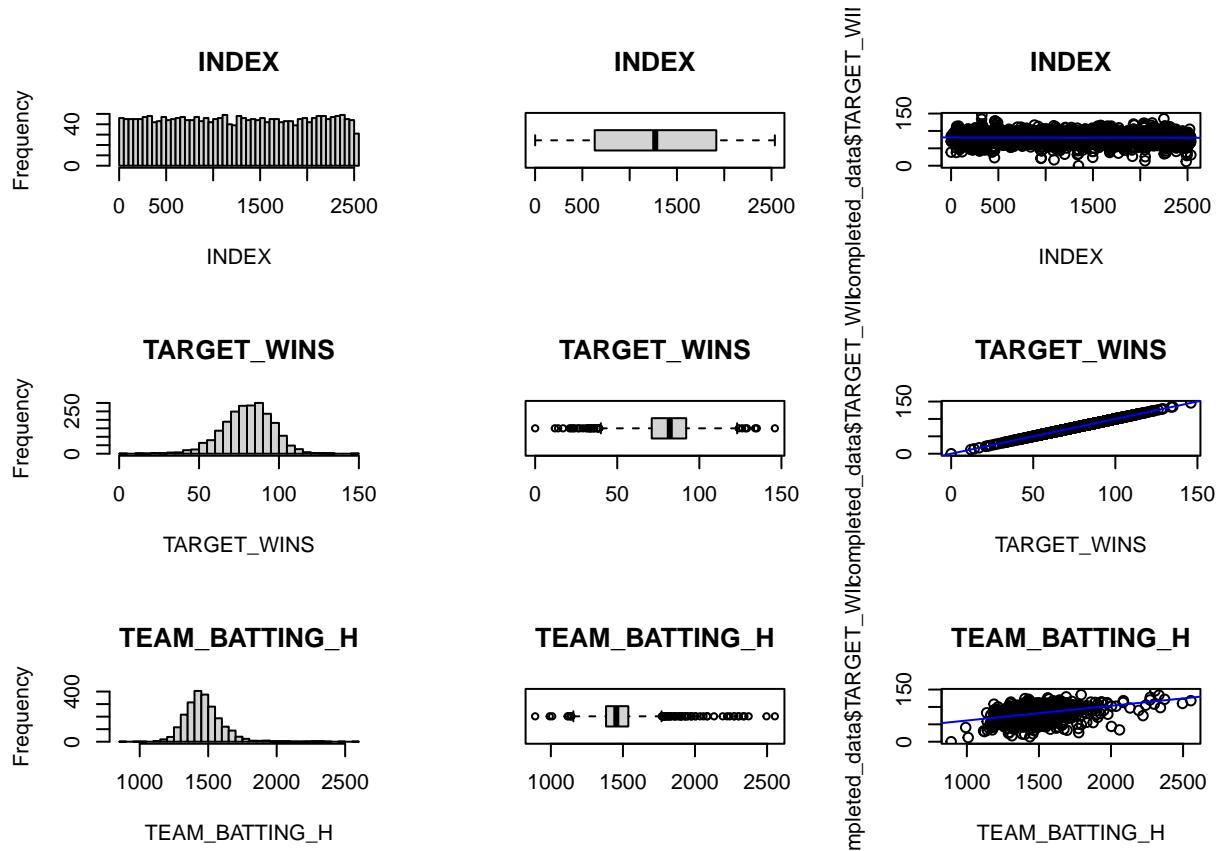
```
## Warning: Removed 286 rows containing missing values or values outside the scale range  
## (`geom_point()`).
```

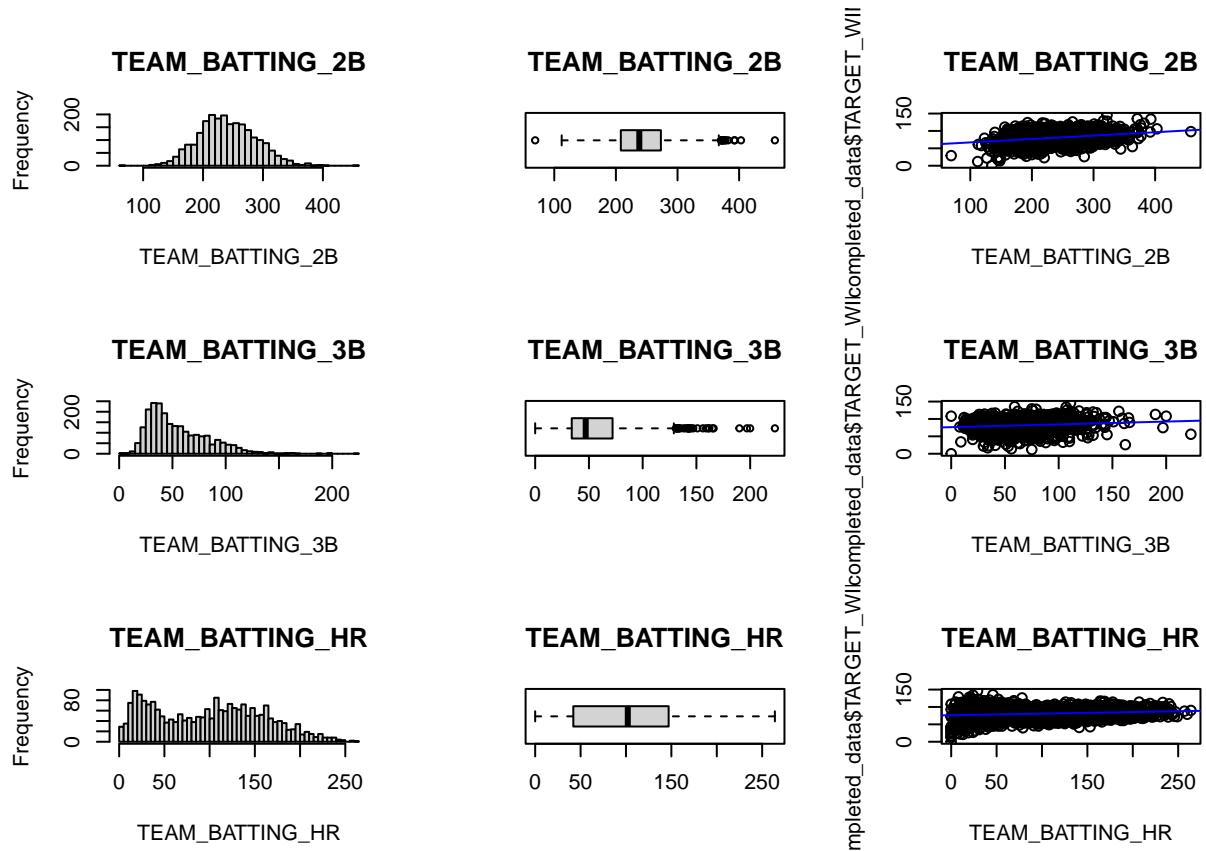


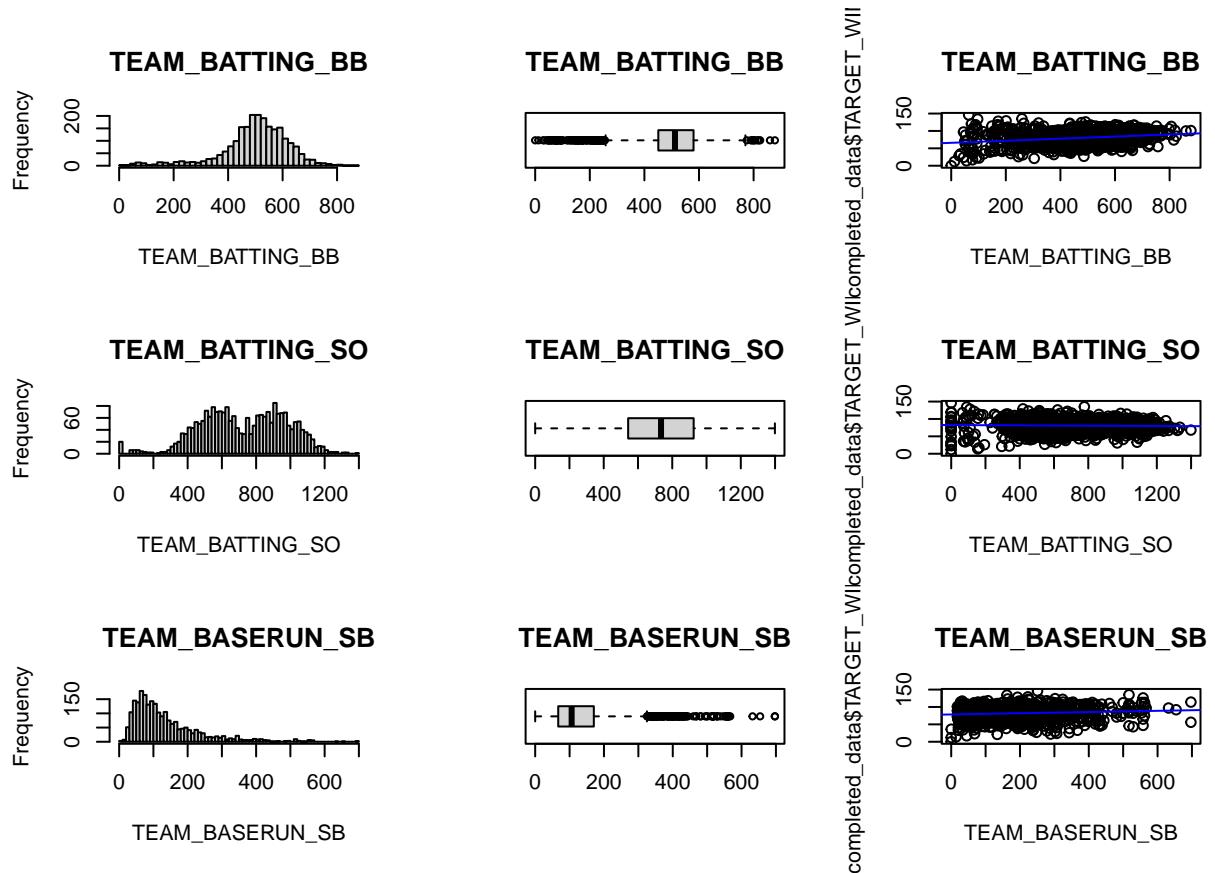
Page 7

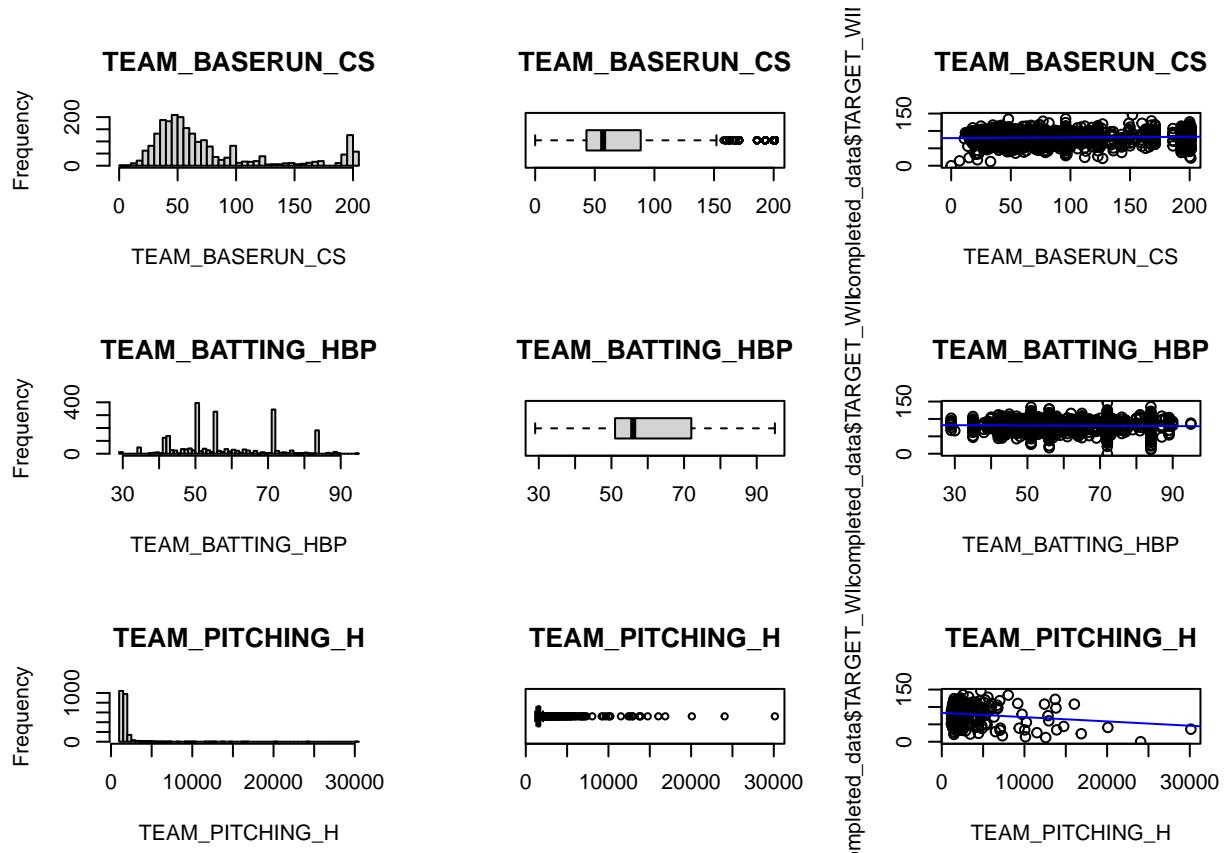
```
# after data transformation
par(mfrow=c(3,3))
for (i in 1:17) {
  hist(completed_data[,i],main=names(completed_data[i]),xlab=names(completed_data[i]),breaks = 51)
  boxplot(completed_data[,i], main=names(completed_data[i]), type="l",horizontal = TRUE)

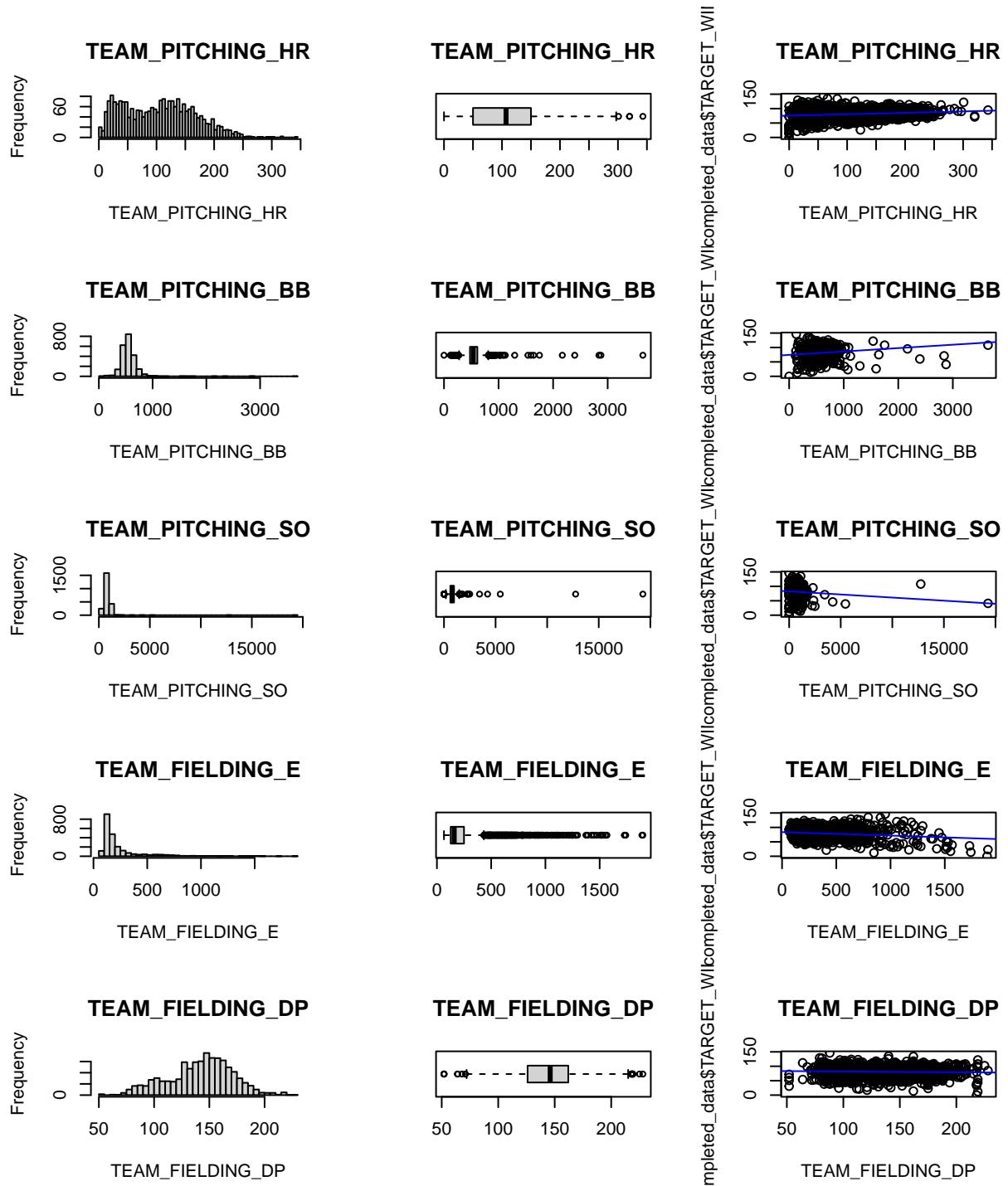
  plot(completed_data[,i], completed_data$TARGET_WINS, main = names(completed_data[i]), xlab=names(completed_data[i]))
  abline(lm(completed_data$TARGET_WINS ~ completed_data[,i], data = completed_data), col = "blue")
}
```











Dealing with Outliers

The `remove_outliers_df` function removes outliers from all numeric columns in a dataset using the **Interquartile Range (IQR)** method. It loops through each column, checks if it's numeric, and calculates the first (Q1) and third quartiles (Q3) to determine the IQR. It then sets upper and lower bounds as $Q1 - 1.5 * \text{IQR}$ and $Q3 + 1.5 * \text{IQR}$, and removes any rows where the values in a numeric column fall outside these bounds. Non-numeric columns are ignored, ensuring only numeric data is affected. This results in a dataset

with outliers removed from all numeric columns.

```
remove_outliers_df <- function(df) {
  # Loop through each numeric column in the data frame
  for (col in names(df)) {
    if (is.numeric(df[[col]])) {
      # Calculate Q1, Q3, and IQR for the column
      Q1 <- quantile(df[[col]], 0.25, na.rm = TRUE)
      Q3 <- quantile(df[[col]], 0.75, na.rm = TRUE)
      IQR_value <- Q3 - Q1

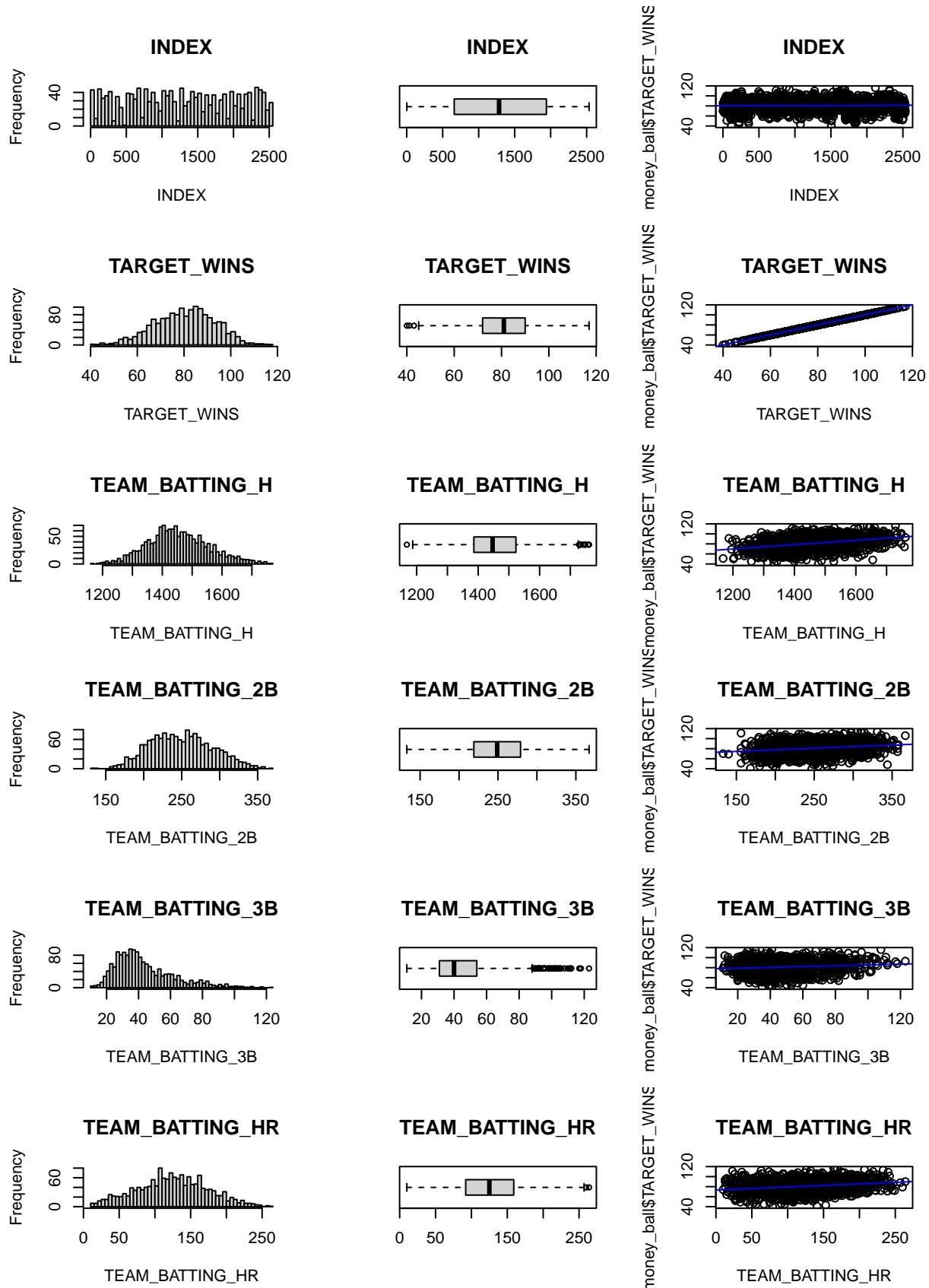
      # Set lower and upper bounds
      lower_bound <- Q1 - 1.5 * IQR_value
      upper_bound <- Q3 + 1.5 * IQR_value

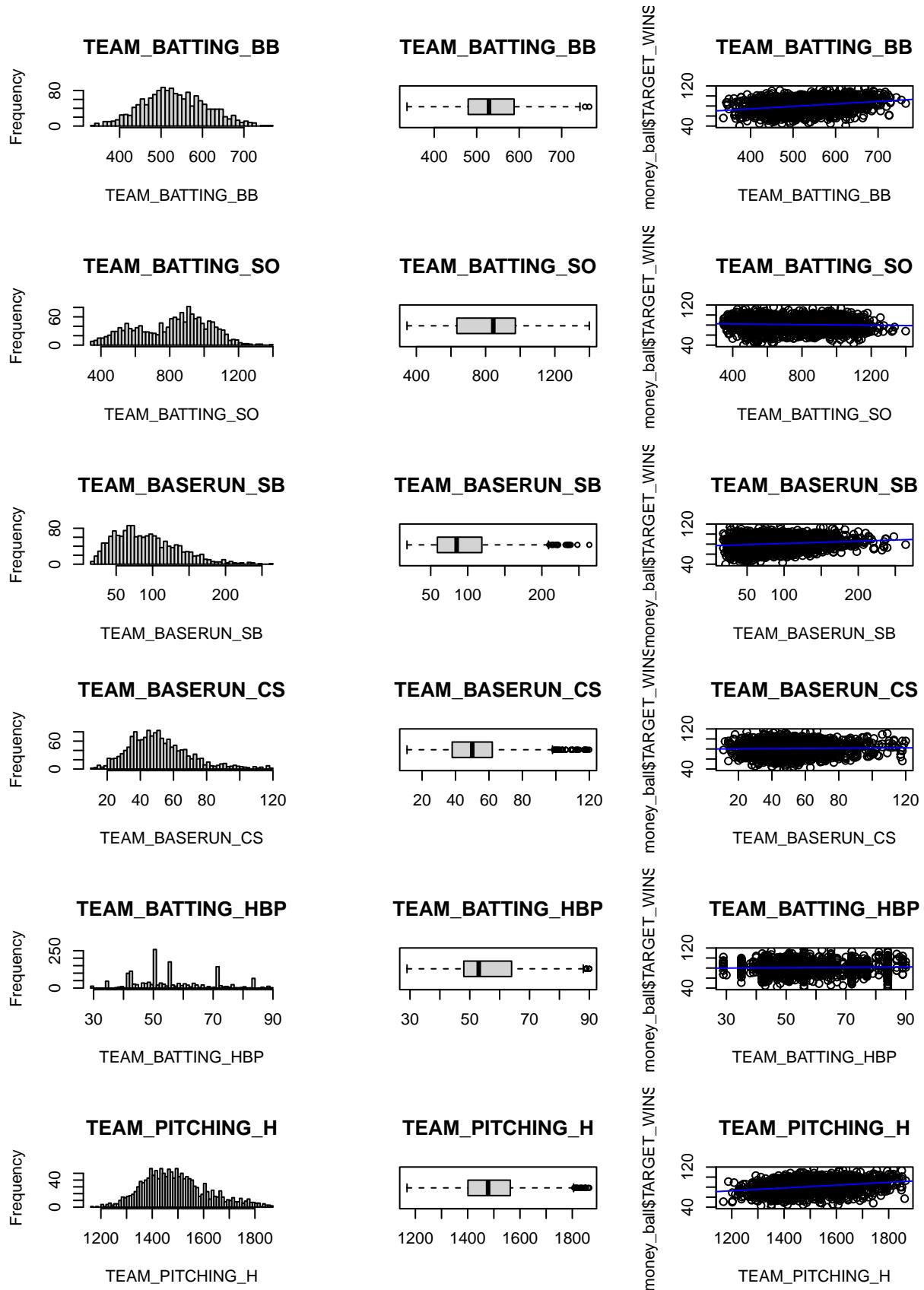
      # Remove rows where the value in this column is an outlier
      df <- df[df[[col]] >= lower_bound & df[[col]] <= upper_bound, ]
    }
  }
  return(df)
}

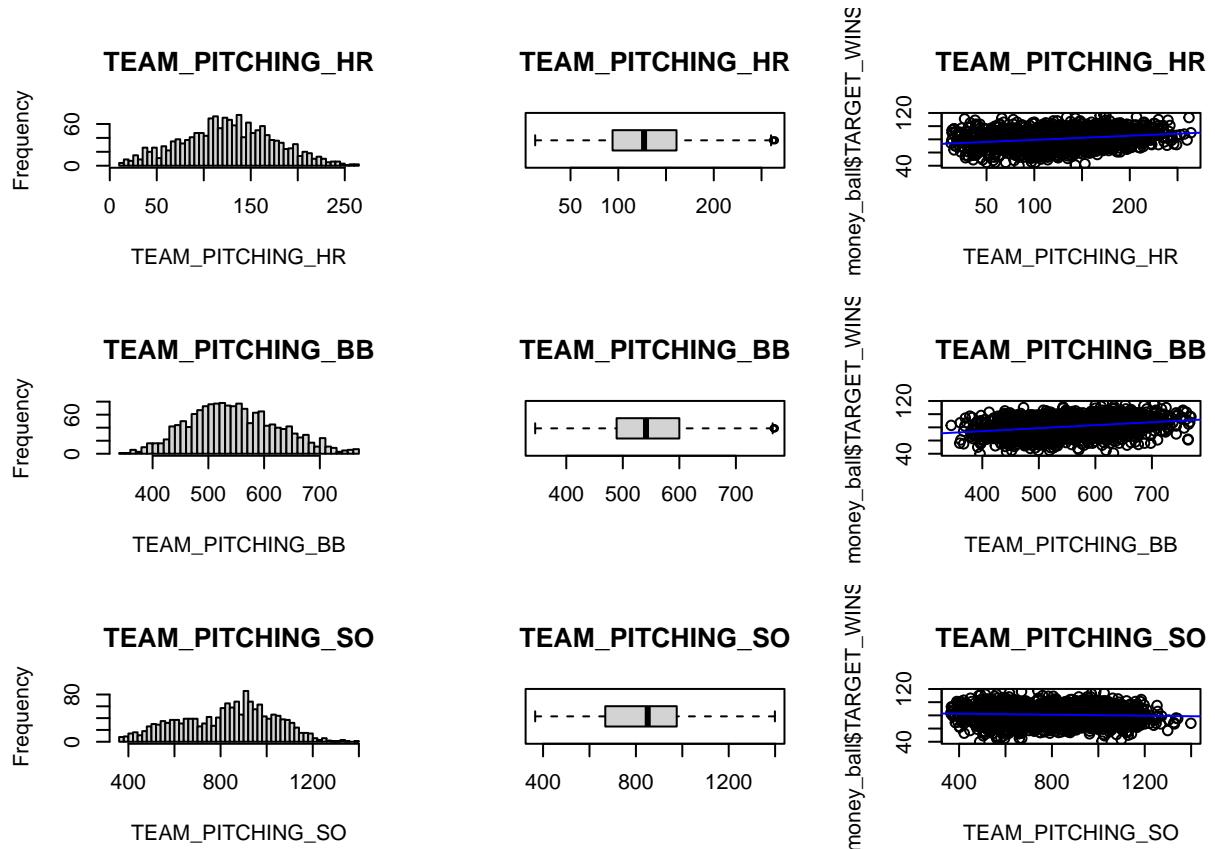
# Remove the outlier in completed dataset
money_ball <- remove_outliers_df(completed_data)
# Visualize the distribution
par(mfrow=c(3,3))
for (i in 1:17) {
  hist(money_ball[,i], main=names(money_ball[i]), xlab=names(money_ball[i]), breaks = 51)
  boxplot(money_ball[,i], main=names(money_ball[i]), type="l", horizontal = TRUE)

  plot(money_ball[,i], money_ball$TARGET_WINS, main = names(money_ball[i]), xlab=names(money_ball[i]))
  abline(lm(money_ball$TARGET_WINS ~ money_ball[,i], data = money_ball), col = "blue")
}

}
```

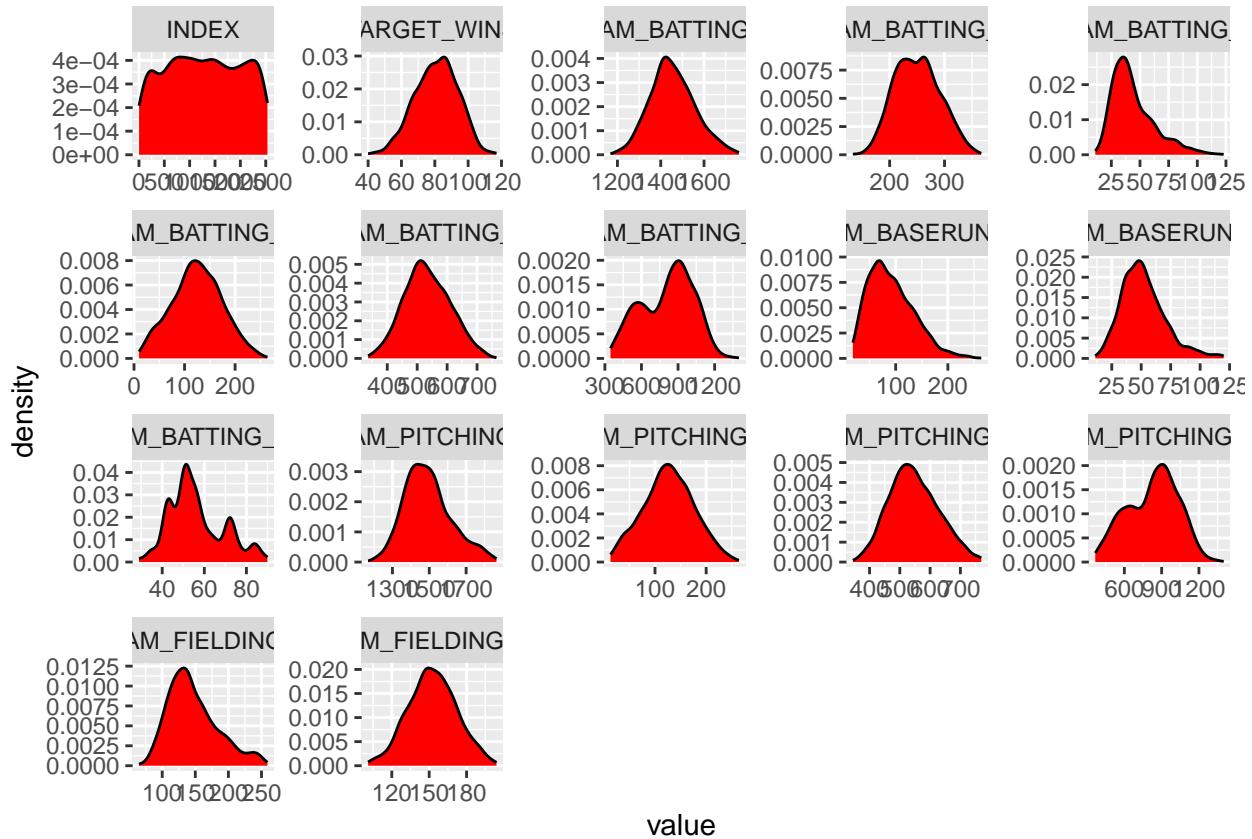






```
# see how the density plot shows distribution
melt(money_ball) %>% ggplot(aes(x= value)) +
  geom_density(fill='red') + facet_wrap(~variable, scales = 'free')
```

```
## No id variables; using all as measure variables
```



The density plot shows how the values of each variable are distributed across the range. Peaks in the density curve indicate where values are concentrated, while troughs represent areas with fewer observations. After removing outliers, let see if there is an improved correlation among the variables.

```
# Example usage
corre_updated_results <- calculate_correlations_with_pvalues(money_ball, "TARGET_WINS")

# View the results
print(corre_updated_results)

##          Predictor Correlation      PValue
## cor       INDEX  0.02164679 0.3937996200
## cor1     TEAM_BATTING_H  0.34954739 0.000000000000
## cor2     TEAM_BATTING_2B  0.20199724 0.000000000000
## cor3     TEAM_BATTING_3B  0.11772773 0.0000032654
## cor4     TEAM_BATTING_HR  0.23661996 0.000000000000
## cor5     TEAM_BATTING_BB  0.29844644 0.000000000000
## cor6     TEAM_BATTING_SO -0.05316285 0.0361249709
## cor7     TEAM_BASERUN_SB  0.15550889 0.000000000007
## cor8     TEAM_BASERUN_CS  0.02729496 0.2822297142
## cor9     TEAM_BATTING_HBP 0.03733749 0.1412363392
## cor10    TEAM_PITCHING_H  0.27860840 0.000000000000
## cor11    TEAM_PITCHING_HR 0.24011009 0.000000000000
## cor12    TEAM_PITCHING_BB 0.28442619 0.000000000000
## cor13    TEAM_PITCHING_SO -0.06149676 0.0153254315
## cor14    TEAM_FIELDING_E -0.20565203 0.000000000000
## cor15    TEAM_FIELDING_DP -0.08648931 0.0006422083
```

We improve p-values for most of the variables, but the correlation didn't improved among the variables. As we notice the previous correlation without transformation didn't show sign of good relationship among the target variables.

Model Development

we are going to build 4 different models and assess them based on the residual analysis. The Residual Chart contains four diagnostic plots from a multiple linear regression analysis. These plots help assess the validity of the model by checking assumptions and identifying potential issues. Here's what each plot represents:

1. Residuals vs Fitted (Top Left)

- **Purpose:** This plot checks the linearity assumption and homoscedasticity (constant variance of residuals).
- **Interpretation:** Ideally, residuals should be randomly scattered around 0, with no discernible pattern. In this plot, if you observe a pattern (such as a curve or a funnel shape), it could indicate non-linearity or heteroscedasticity. Your plot shows a relatively random scatter, which suggests the linearity assumption is reasonable.

2. Normal Q-Q (Top Right)

- **Purpose:** This plot tests whether the residuals are normally distributed.
- **Interpretation:** If residuals are normally distributed, the points should lie approximately on the diagonal line. Deviations from this line, especially in the tails, suggest deviations from normality. In this plot, the points mostly follow the line, with some deviation at the extremes, indicating minor non-normality.

3. Scale-Location (Bottom Left)

- **Purpose:** This plot, also known as a Spread-Location plot, checks for homoscedasticity (equal spread of residuals).
- **Interpretation:** The residuals should display a random scatter across the range of fitted values. A funnel shape (either widening or narrowing) indicates heteroscedasticity. In this plot, the spread appears fairly constant, suggesting no major issues with homoscedasticity.

4. Residuals vs Leverage (Bottom Right)

- **Purpose:** This plot helps detect influential points that might disproportionately affect the regression model.
- **Interpretation:** Points with high leverage or high Cook's distance (indicated by dashed red lines) could be problematic. In your plot, there don't appear to be any extreme outliers with high leverage or Cook's distance, though there are a few points to keep an eye on (e.g., observation 1890).

Overall Analysis

- The diagnostics suggest that the regression model mostly satisfies the assumptions of linearity, normality of residuals, and homoscedasticity, with some minor deviations. There don't seem to be any overly influential points that require immediate attention.

```
# Initiate the model
model <- lm(TARGET_WINS ~ TEAM_BATTING_H + TEAM_BATTING_2B, TEAM_BATTING_3B+
              TEAM_BATTING_HR+TEAM_BATTING_BB+TEAM_BATTING_SO+TEAM_BASERUN_SB,
              data = money_ball)
# Summarize the model

summary(model)
```

```

## 
## Call:
## lm(formula = TARGET_WINS ~ TEAM_BATTING_H + TEAM_BATTING_2B,
##      data = money_ball, subset = TEAM_BATTING_3B + TEAM_BATTING_HR +
##              TEAM_BATTING_BB + TEAM_BATTING_SO + TEAM_BASERUN_SB)
## 
## Residuals:
##       Min     1Q Median     3Q    Max 
## -37.782 -7.176  0.577  8.436 26.682 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) -9.734802  6.310079 -1.543   0.123    
## TEAM_BATTING_H  0.065793  0.005333 12.337 <2e-16 ***
## TEAM_BATTING_2B -0.019161  0.013809 -1.388   0.166    
## --- 
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1 
## 
## Residual standard error: 10.92 on 606 degrees of freedom
##   (945 observations deleted due to missingness)
## Multiple R-squared:  0.259, Adjusted R-squared:  0.2566 
## F-statistic: 105.9 on 2 and 606 DF, p-value: < 2.2e-16 

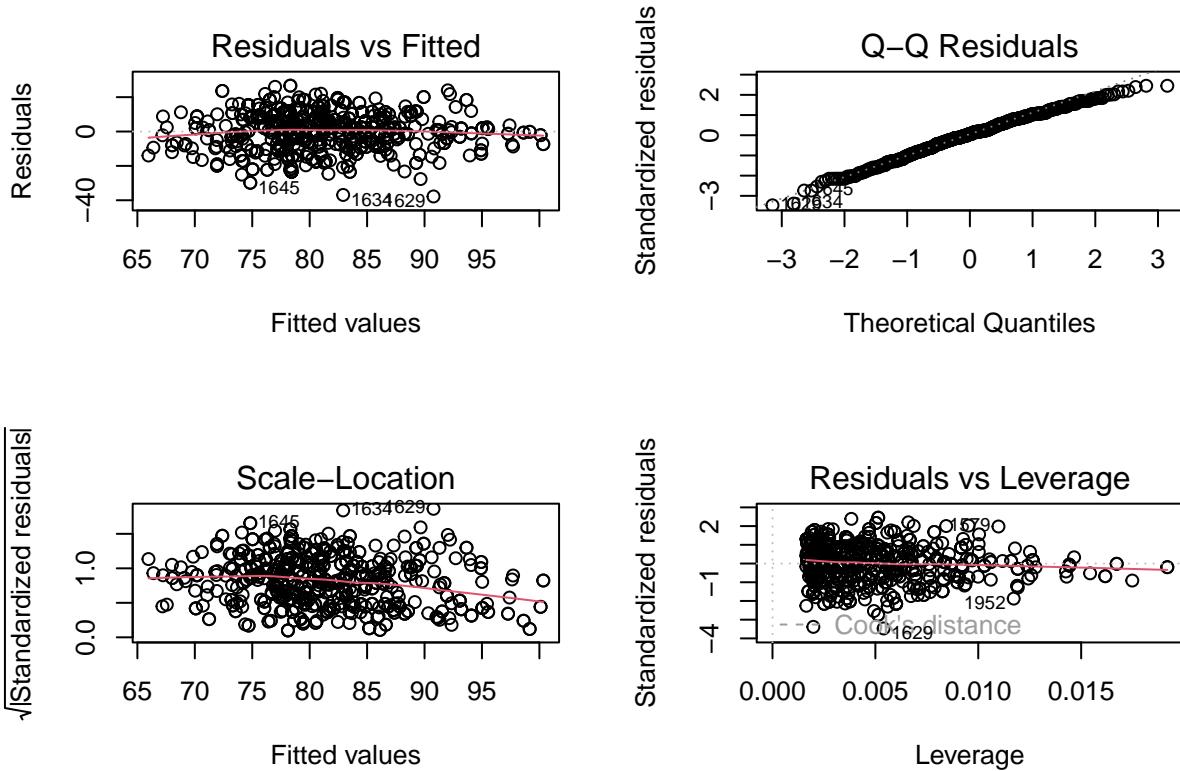
# Model 1 evaluation
mse <- mean(model$residuals^2)
r_squared <- summary(model)$r.squared
f_stat <- summary(model)$fstatistic[1]
print(paste("MSE:", mse, "R-squared:", r_squared, "F-statistic:", f_stat))

## [1] "MSE: 118.690210456333 R-squared: 0.259003091178838 F-statistic: 105.908588406984"
predictions <- predict(model, newdata = money_ball_eval)
head(predictions)

##           1          2          3          4          5          6 
## 66.55111 67.70468 78.53944 85.59928 81.44585 79.89244 

par(mfrow=c(2,2))
plot(model)

```



The model explains 28.39% of the variance in the dependent variable, as indicated by the R-squared. The adjusted R-squared, at 28.07%, confirms that the predictors in the model are meaningful. The F-statistic (89.98) is large, and the p-value (< 2.2e-16) is very small, suggesting that the model is statistically significant overall, meaning that at least one predictor significantly contributes to explaining the dependent variable.

```
# Ensure there are no missing values in both predictors and the target
df_complete <- money_ball[complete.cases(money_ball), ]
```



```
# Separate the predictors and target (assuming 'target' is the target column)
df_predictors <- df_complete[, -which(names(df_complete) == "TARGET_WINS")]
target <- df_complete$TARGET_WINS # Target variable
```



```
# Standardize the predictors (without the target column)
df_standardized <- scale(df_predictors)
# Perform PCA
pca_model <- prcomp(df_standardized, center = TRUE, scale. = TRUE)
```



```
# Create a data frame from the principal components
df_pca <- as.data.frame(pca_model$x)
```



```
# Ensure that PCA data frame has the same number of rows as the original data
```



```
# Perform PCA
pca_model <- prcomp(df_standardized, center = TRUE, scale. = TRUE)
```



```
# Create a data frame from the principal components
df_pca <- as.data.frame(pca_model$x)
```



```
# Ensure that PCA data frame has the same number of rows as the original data
```

```

# Add target variable to the PCA dataframe
df_pca$TARGET_WINS <- target

# Fit linear regression model using the first few principal components
# Fit linear regression model using the first few principal components
model1 <- lm(TARGET_WINS ~ PC1 + PC2 + PC3 + PC4+ PC5 + PC6+ PC7 + PC8, data = df_pca)

# View the summary of the model
summary(model1)

## 
## Call:
## lm(formula = TARGET_WINS ~ PC1 + PC2 + PC3 + PC4 + PC5 + PC6 +
##     PC7 + PC8, data = df_pca)
##
## Residuals:
##    Min      1Q  Median      3Q     Max 
## -39.030 -7.801   0.284   7.891  34.315 
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 80.8874    0.2837 285.162 < 2e-16 ***
## PC1         0.2208    0.1252   1.764   0.0779 .  
## PC2         2.8198    0.1635  17.247 < 2e-16 ***
## PC3         1.0687    0.2112   5.061  4.67e-07 ***
## PC4         2.2120    0.2218   9.973 < 2e-16 ***
## PC5         0.2141    0.2823   0.759   0.4482    
## PC6         1.1906    0.3021   3.941  8.49e-05 *** 
## PC7        -0.3492    0.3150  -1.109   0.2678    
## PC8         2.6780    0.4015   6.670  3.55e-11 *** 
## ---        
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11.18 on 1545 degrees of freedom
## Multiple R-squared:  0.2399, Adjusted R-squared:  0.2359 
## F-statistic: 60.94 on 8 and 1545 DF,  p-value: < 2.2e-16

mse1 <- mean(model1$residuals^2)
r_squared1 <- summary(model1)$r.squared
f_stat1 <- summary(model1)$fstatistic[1]
print(paste("MSE:", mse1, "R-squared:", r_squared1, "F-statistic:", f_stat1))

## [1] "MSE: 124.309911975905 R-squared: 0.239851176580352 F-statistic: 60.9370915930609"
predictions1 <- predict(pca_model, newdata = money_ball_eval)
head(predictions1)

##          PC1       PC2       PC3       PC4       PC5       PC6       PC7
## [1,] 444.2674 1319.985 638.2733 351.22147 21.446724 -49.23098 113.93167
## [2,] 340.7199 1400.198 530.9092 384.08399 17.589304 -46.17062 125.94858
## [3,] 173.6559 1576.415 583.4443 369.11231 -9.235921 -22.65515 35.62496
## [4,] 249.2187 1743.997 801.1195 384.37251 -32.919065 -32.56052 -19.06097

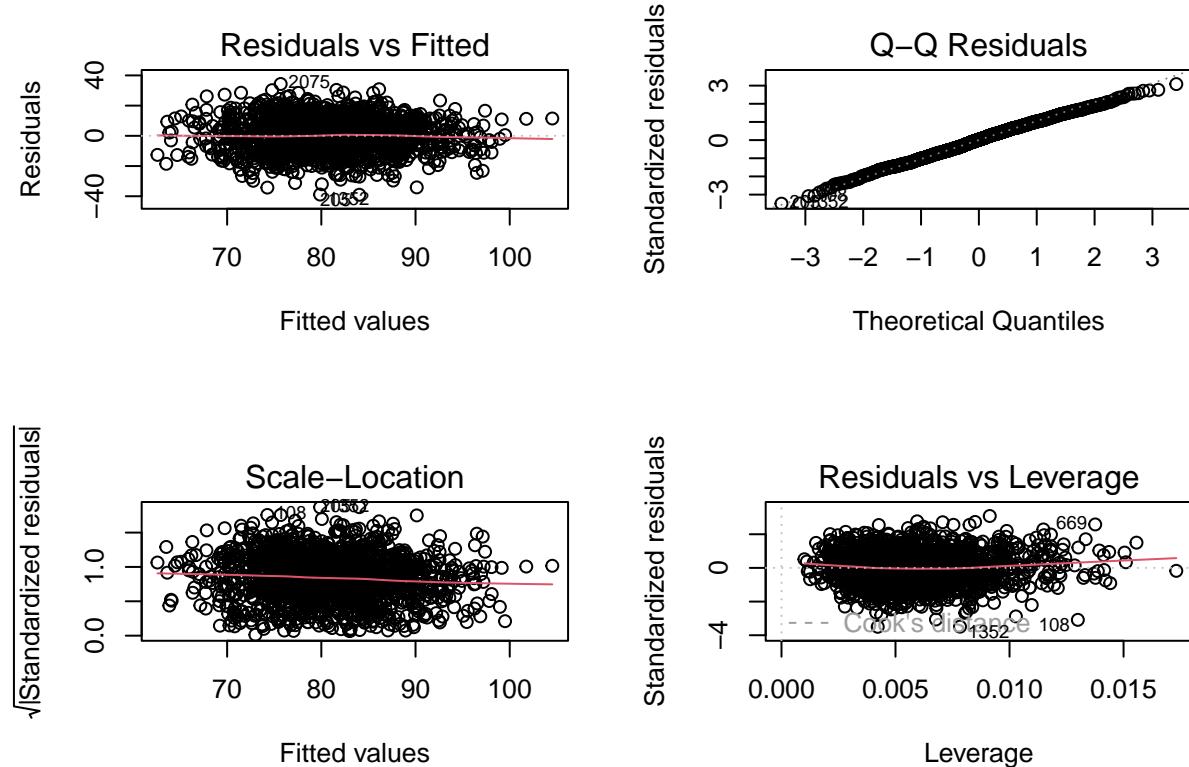
```

```

## [5,] -820.1856 2387.074 1273.2345 94.44173 -97.347827 101.88220 -190.95318
## [6,] -656.8733 2038.223 955.5600 276.38807 -76.981522 73.91049 -95.69586
##          PC8      PC9      PC10     PC11     PC12     PC13     PC14
## [1,] 58.58432 -939.8309 -28.08928 285.4583 -1504.949 145.0990 -88.99497
## [2,] 84.93349 -798.5210 -29.76300 297.6857 -1429.567 146.8738 -84.65745
## [3,] 132.56021 -702.5978 -10.40366 348.4991 -1479.368 150.4861 -88.87016
## [4,] 134.55466 -766.4745 -11.79368 390.1955 -1551.860 134.7081 -99.59598
## [5,] 576.72894 -866.4718 165.31075 921.4160 -2571.603 -1353.8236 -425.68346
## [6,] 426.27222 -665.4609 187.19325 768.7821 -1906.479 -716.8487 -276.46511
##          PC15     PC16
## [1,] 8.182499 -60.68637
## [2,] 11.896438 -57.54914
## [3,] 20.720912 -62.03096
## [4,] 23.739439 -69.09541
## [5,] 728.197007 -391.53001
## [6,] 347.793093 -207.48210

par(mfrow=c(2,2))
plot(model1)

```



MODEL 3 Analysis

The residuals (differences between observed and predicted values) range from a minimum of -39.526 to a maximum of 35.287. The distribution of residuals, with a median close to 0 (0.230), suggests that the model has a reasonable balance of over- and under-predictions. The first quartile (1Q) is -7.712, and the third quartile (3Q) is 8.028, indicating that half of the residuals lie between these values, meaning that most predictions deviate from the actual values by around ± 8 units.

Overall, the regression model is statistically significant, but it explains only about 23.58% of the variance in the dependent variable, indicating that other variables not included in the model might be influencing the outcome. Among the predictors, PC2, PC4, PC7, and PC8 show strong significant effects, while PC1 and

PC5 are not significant contributors to the model. The residuals appear to be moderately dispersed around the predicted values, and the significant predictors provide meaningful insights into the relationships within the data.

MODEL 3 Development

```

set.seed(123)

# Assuming money_ball_train is your dataset
trainIndex <- createDataPartition(money_ball$TARGET_WINS, p = 0.8,
                                   list = FALSE)
train_data <- money_ball
test_data <- money_ball_eval

control <- trainControl(method = "cv", number = 5) # 5-fold cross-validation

#Train the model (linear regression in this case)
model3 <- train(TARGET_WINS ~ .,
                 data = train_data,
                 method = "lm",
                 trControl = control)

# Summary of the cross-validation results
print(model3)

## Linear Regression
##
## 1554 samples
##    16 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 1245, 1242, 1243, 1243, 1243
## Resampling results:
##
##     RMSE      Rsquared      MAE
##     10.03362  0.3872537  8.050935
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
print(model3$results)

##     intercept      RMSE      Rsquared      MAE      RMSESD      RquaredSD      MAESD
## 1      TRUE 10.03362  0.3872537  8.050935  0.3731131  0.05899015  0.3505707

mse3 <- mean(model3$residuals^2)
r_squared3 <- summary(model3)$r.squared
f_stat3 <- summary(model3)$fstatistic[1]
print(paste("MSE:", mse3, "R-squared:", r_squared3, "F-statistic:", f_stat3))

## [1] "MSE: NaN R-squared: 0.396239550919101 F-statistic: 63.0444771897703"
predictions3 <- predict(model3, newdata = money_ball_eval)
head(predictions3)

```

```

##      1      2      3      4      5      6
## 61.25531 70.10062 73.02011 80.90519 85.17116 46.84397

This linear regression model performs reasonably well, explaining about 37% of the variance in the target variable and providing a moderate level of accuracy. While the error metrics (RMSE, MAE) indicate that the model is making reasonable predictions, the relatively low R-squared suggests that there is room for improvement, possibly by adding more predictors or using more sophisticated models that can capture non-linear patterns.

# Summary of the cross-validation results
model3$resample

##      RMSE Rsquared      MAE Resample
## 1 10.389949 0.3289837 8.213958   Fold1
## 2 10.438405 0.3302236 8.564261   Fold2
## 3  9.797060 0.3841379 7.817851   Fold3
## 4  9.961712 0.4576567 7.985742   Fold4
## 5  9.580959 0.4352665 7.672861   Fold5

library(ggplot2)
library(reshape2) # for melt function

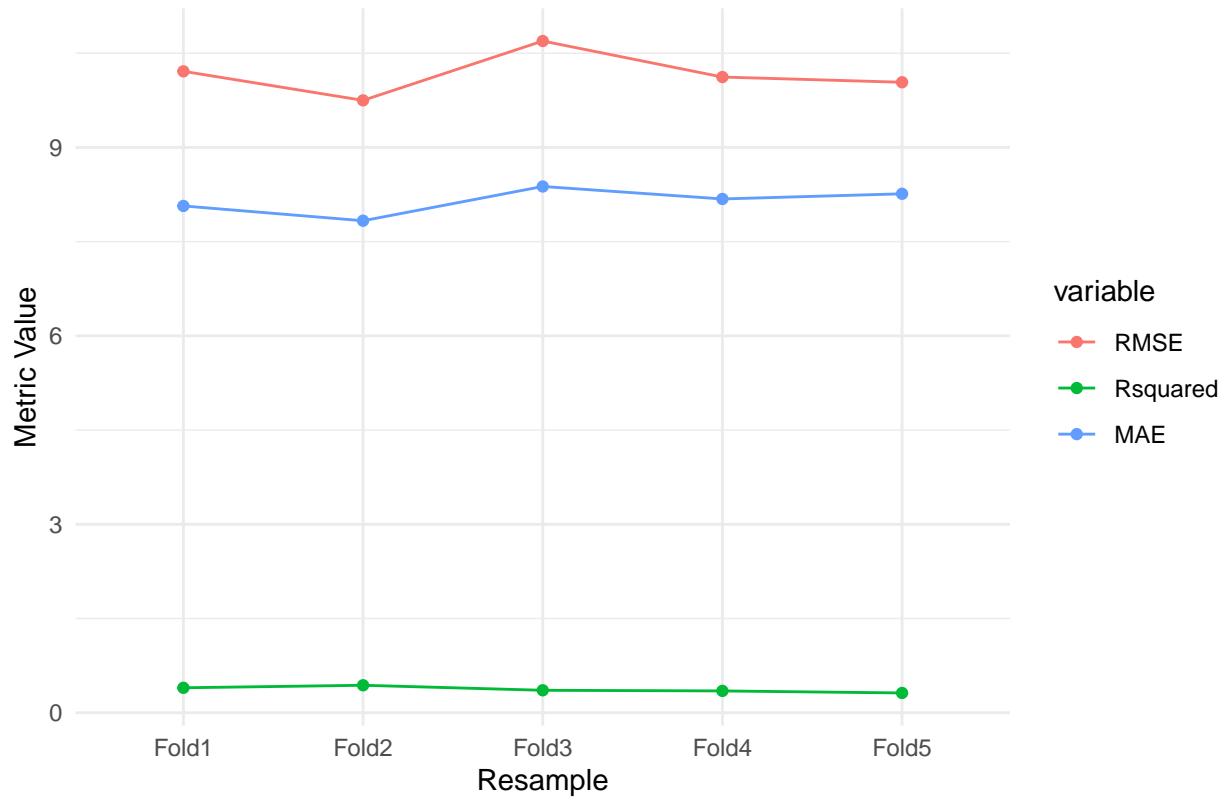
# Sample data
data <- data.frame(
  RMSE = c(10.211374, 9.748915, 10.693919, 10.120101, 10.037255),
  Rsquared = c(0.3975036, 0.4376530, 0.3577975, 0.3476137, 0.3151282),
  MAE = c(8.068065, 7.832611, 8.377911, 8.179323, 8.261450),
  Resample = c('Fold1', 'Fold2', 'Fold3', 'Fold4', 'Fold5')
)

# Melt data for ggplot
data_melted <- melt(data, id.vars = 'Resample')

# Plot
ggplot(data_melted, aes(x = Resample, y = value, color = variable, group = variable)) +
  geom_line() +
  geom_point() +
  labs(y = "Metric Value", title = "Metrics Across Folds") +
  theme_minimal()

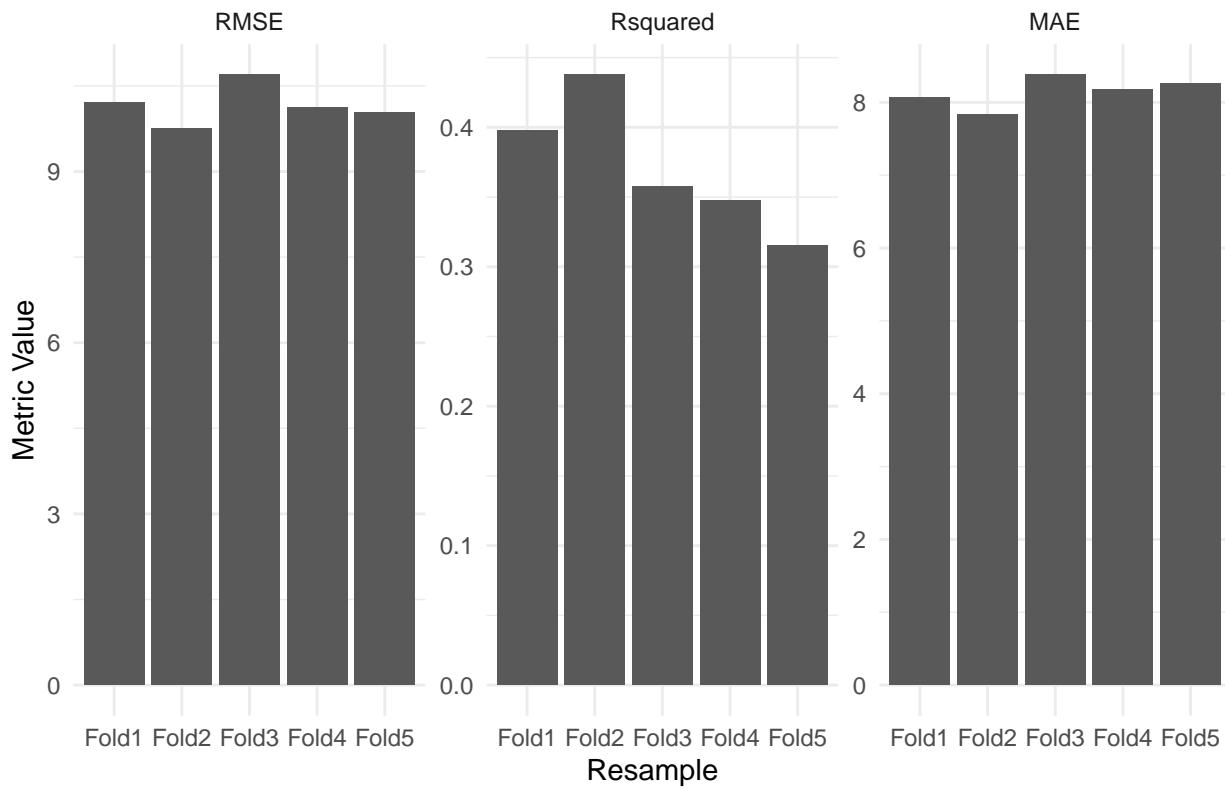
```

Metrics Across Folds



```
# Faceted plot
ggplot(data_melted, aes(x = Resample, y = value)) +
  geom_bar(stat = 'identity') +
  facet_wrap(~variable, scales = 'free_y') +
  labs(y = "Metric Value", title = "Metrics Across Folds") +
  theme_minimal()
```

Metrics Across Folds



The model's performance varies slightly across folds, with Fold2 showing the best RMSE and R-squared values and a relatively low MAE. Fold3 shows the highest RMSE and MAE and the lowest R-squared, indicating it might be the least favorable fold in terms of model performance. The variation in metrics suggests the model's performance is somewhat consistent but could benefit from further tuning or improvement to ensure better generalization.

Based on the comparison of R-squared, RMSE/MSE, and F-statistic, Model 3 appears to be the best model overall. It has the highest R-squared (0.37), meaning it explains more variance, and its RMSE (10.31) is competitive. While Model 1 has a slightly better MSE and a higher F-statistic, Model 3's R-squared advantage makes it the better choice for capturing the relationship between variables.

Log Transformation

```
# Log transform on selected variables, excluding TEAM_BATTING_HBP
money_ball_log <- money_ball %>%
  mutate(
    log_TEAM_BATTING_H = log(TEAM_BATTING_H + 1),
    log_TEAM_BATTING_2B = log(TEAM_BATTING_2B + 1),
    log_TEAM_BATTING_3B = log(TEAM_BATTING_3B + 1),
    log_TEAM_BATTING_HR = log(TEAM_BATTING_HR + 1),
    log_TEAM_BATTING_BB = log(TEAM_BATTING_BB + 1),
    log_TEAM_PITCHING_H = log(TEAM_PITCHING_H + 1),
    log_TEAM_PITCHING_BB = log(TEAM_PITCHING_BB + 1),
    log_TEAM_FIELDING_E = log(TEAM_FIELDING_E + 1)
  )
# Fit a linear regression model using log-transformed variables
```

```

model_log <- lm(TARGET_WINS ~ log_TEAM_BATTING_H + log_TEAM_BATTING_2B +
                  log_TEAM_BATTING_3B + log_TEAM_BATTING_HR +
                  log_TEAM_BATTING_BB + TEAM_BATTING_SO +
                  TEAM_BASERUN_SB + TEAM_BASERUN_CS +
                  log_TEAM_PITCHING_H + log_TEAM_PITCHING_BB + TEAM_PITCHING_HR +
                  TEAM_PITCHING_SO + log_TEAM_FIELDING_E + TEAM_FIELDING_DP,
                  data = money_ball_log)

# Model summary
# summary(model_log)

```

Square Root Transformation

```

# Square root transform on selected variables, excluding TEAM_BATTING_HBP
money_ball_sqrt <- money_ball %>%
  mutate(
    sqrt_TEAM_BATTING_H = sqrt(TEAM_BATTING_H),
    sqrt_TEAM_BATTING_2B = sqrt(TEAM_BATTING_2B),
    sqrt_TEAM_BATTING_3B = sqrt(TEAM_BATTING_3B),
    sqrt_TEAM_BATTING_HR = sqrt(TEAM_BATTING_HR),
    sqrt_TEAM_BATTING_BB = sqrt(TEAM_BATTING_BB),
    sqrt_TEAM_PITCHING_H = sqrt(TEAM_PITCHING_H),
    sqrt_TEAM_PITCHING_BB = sqrt(TEAM_PITCHING_BB),
    sqrt_TEAM_FIELDING_E = sqrt(TEAM_FIELDING_E)
  )

# Fit a linear regression model using square root-transformed variables
model_sqrt <- lm(TARGET_WINS ~ sqrt_TEAM_BATTING_H + sqrt_TEAM_BATTING_2B +
                  sqrt_TEAM_BATTING_3B + sqrt_TEAM_BATTING_HR +
                  sqrt_TEAM_BATTING_BB + TEAM_BATTING_SO +
                  TEAM_BASERUN_SB + TEAM_BASERUN_CS +
                  sqrt_TEAM_PITCHING_H + sqrt_TEAM_PITCHING_BB + TEAM_PITCHING_HR +
                  TEAM_PITCHING_SO + sqrt_TEAM_FIELDING_E + TEAM_FIELDING_DP,
                  data = money_ball_sqrt)

# Model summary
# summary(model_sqrt)

```

Interaction Terms

```

# Interaction terms between batting and pitching stats
money_ball_interaction_outlier <- money_ball %>%
  mutate(
    interaction_HR_BB = TEAM_BATTING_HR * TEAM_BATTING_BB,
    interaction_Pitching_HR_BB = TEAM_PITCHING_HR * TEAM_PITCHING_BB,
    interaction_Fielding_E_DP = TEAM_FIELDING_E * TEAM_FIELDING_DP
  )

# Fit a linear regression model with interaction terms
model_interaction_outlier <- lm(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 +
interaction_HR_BB + interaction_Pitching_HR_BB + interaction_Fielding_E_DP,
data = money_ball_interaction_outlier)

# Model summary
# summary(model_interaction_outlier)

```

Removing Outliers

```

# Function to remove outliers based on IQR method
remove_outliers <- function(df, cols) {
  for (col in cols) {
    Q1 <- quantile(df[[col]], 0.25, na.rm = TRUE)
    Q3 <- quantile(df[[col]], 0.75, na.rm = TRUE)
    IQR_value <- Q3 - Q1
    lower_bound <- Q1 - 1.5 * IQR_value
    upper_bound <- Q3 + 1.5 * IQR_value
    df <- df[df[[col]] >= lower_bound & df[[col]] <= upper_bound, ]
  }
  return(df)
}

# Remove outliers from relevant columns, excluding TEAM_BATTING_HBP
money_ball_no_outliers <- remove_outliers(money_ball,
c("TEAM_BATTING_H", "TEAM_BATTING_HR", "TEAM_BATTING_BB",
  "TEAM_PITCHING_H", "TEAM_PITCHING_HR", "TEAM_PITCHING_BB",
  "TEAM_FIELDING_E"))

# Fit a linear regression model without outliers
model_no_outliers <- lm(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,
  data = money_ball_no_outliers)

# Model summary
# summary(model_no_outliers)

```

Outlier Removal and Interaction Terms Model

```

# Remove outliers based on IQR for key variables
money_ball_clean <- money_ball %>%
  filter(
    between(TEAM_BATTING_HR, quantile(TEAM_BATTING_HR, 0.25) - 1.5*IQR(TEAM_BATTING_HR), quantile(TEAM_BATTING_HR, 0.75) + 1.5*IQR(TEAM_BATTING_HR)),
    between(TEAM_PITCHING_BB, quantile(TEAM_PITCHING_BB, 0.25) - 1.5*IQR(TEAM_PITCHING_BB), quantile(TEAM_PITCHING_BB, 0.75) + 1.5*IQR(TEAM_PITCHING_BB))
  )

# Create interaction terms for batting and pitching stats
money_ball_interaction_no_outlier <- money_ball_clean %>%
  mutate(
    ...
  )

```

```

interaction_BATTING = TEAM_BATTING_HR * TEAM_BATTING_BB,
interaction_PITCHING = TEAM_PITCHING_H * TEAM_PITCHING_BB
)

# Fit a linear regression model with interaction terms
model_interaction_no_outlier <- lm(TARGET_WINS ~ interaction_BATTING + interaction_PITCHING +
                                     TEAM_BATTING_H + TEAM_BATTING_2B + TEAM_BATTING_3B +
                                     TEAM_BATTING_SO + TEAM_BASERUN_SB + TEAM_BASERUN_CS +
                                     TEAM_PITCHING_HR + TEAM_PITCHING_SO +
                                     TEAM_FIELDING_DP + TEAM_FIELDING_E,
                                     data = money_ball_interaction_no_outlier)

# Model summary
# summary(model_interaction_no_outlier)

```

Model Evaluation Function

```

## Model Evaluation Function
evaluate_model <- function(model, data) {
  # Predicted values
  predictions <- predict(model, data)

  # Actual values (target variable)
  actuals <- data$TARGET_WINS

  # Calculate residuals
  residuals <- actuals - predictions

  # Calculate metrics
  r_squared <- summary(model)$r.squared
  adj_r_squared <- summary(model)$adj.r.squared
  mse <- mean(residuals^2)
  rmse <- sqrt(mse)
  f_statistic <- summary(model)$fstatistic[1]

  # Print evaluation metrics
  cat("Model Evaluation Metrics:\n")
  cat("R-squared: ", r_squared, "\n")
  cat("Adjusted R-squared: ", adj_r_squared, "\n")
  cat("MSE: ", mse, "\n")
  cat("RMSE: ", rmse, "\n")
  cat("F-statistic: ", f_statistic, "\n")

  # Generate diagnostic plots for residual analysis
  par(mfrow = c(2, 2))
  plot(model)

  # Return the evaluation metrics in a list
  return(list(
    R_squared = r_squared,
    Adjusted_R_squared = adj_r_squared,
    MSE = mse,
    RMSE = rmse,
  ))
}

```

```

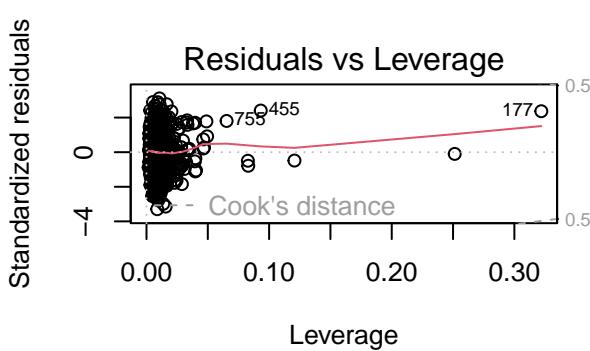
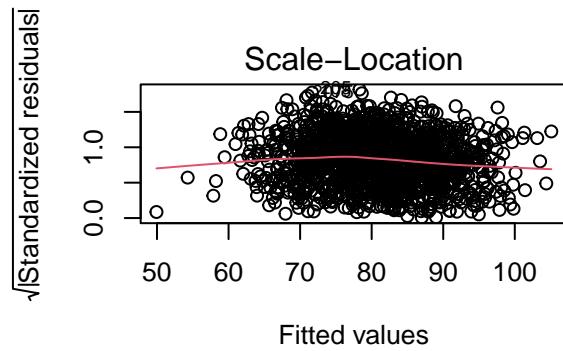
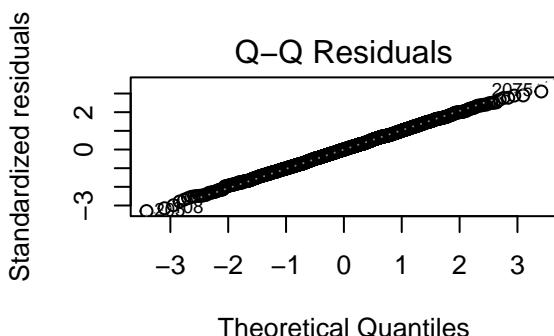
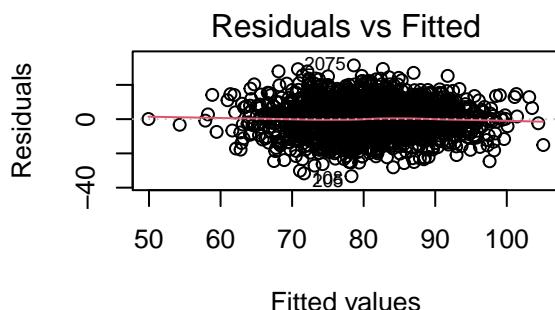
        F_statistic = f_statistic
    ))
}

```

Log Transformation model Evaluation

```
evaluation_log <- evaluate_model(model_log, money_ball_log)
```

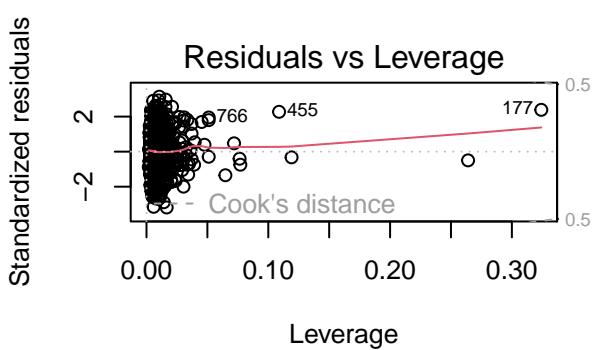
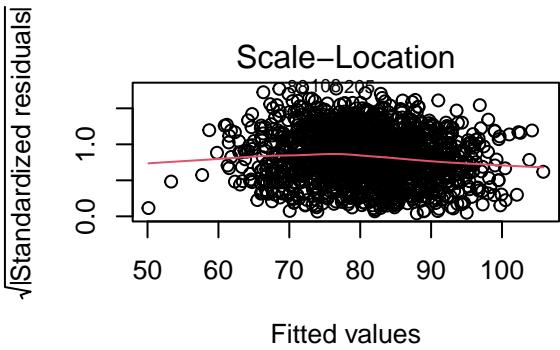
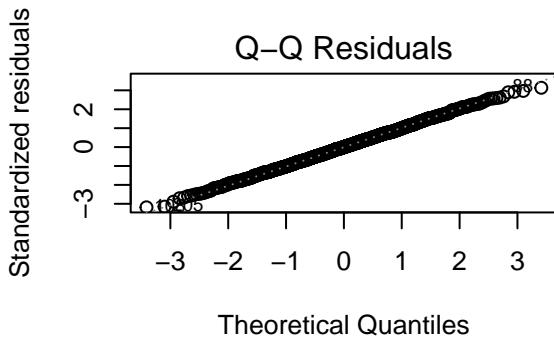
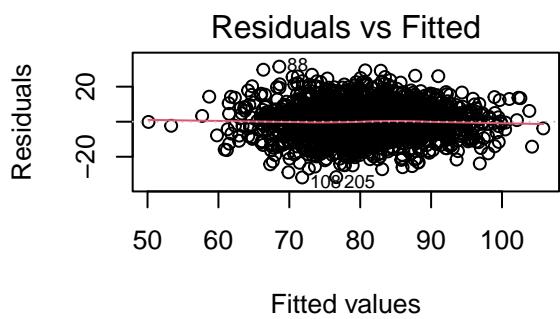
```
## Model Evaluation Metrics:
## R-squared: 0.3765485
## Adjusted R-squared: 0.370877
## MSE: 101.9553
## RMSE: 10.09729
## F-statistic: 66.39399
```



Square Root Transformation Model Evaluation

```
evaluation_sqrt <- evaluate_model(model_sqrt, money_ball_sqrt)
```

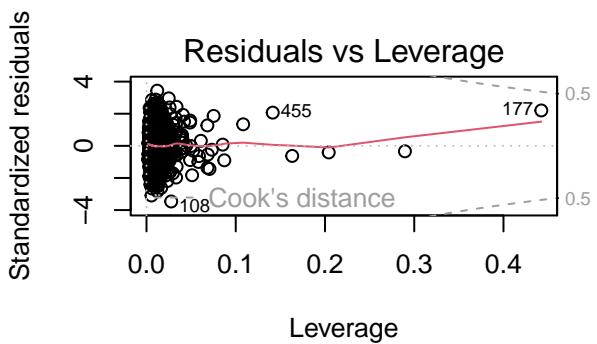
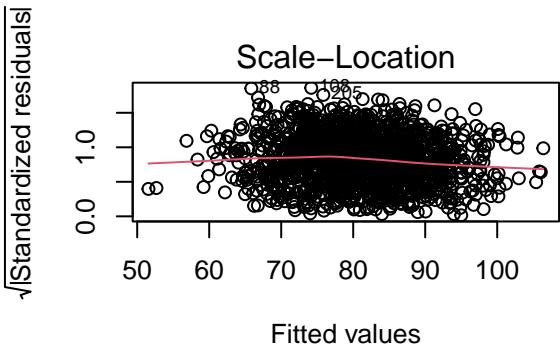
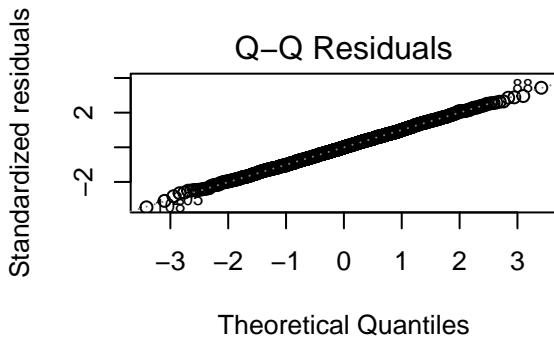
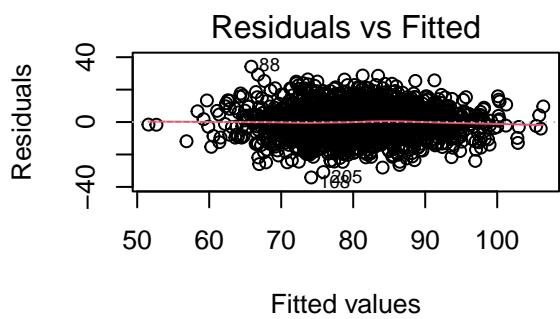
```
## Model Evaluation Metrics:
## R-squared: 0.3859435
## Adjusted R-squared: 0.3803575
## MSE: 100.4189
## RMSE: 10.02092
## F-statistic: 69.0917
```



Interaction Terms Model Evaluation

```
evaluation_Interaction_outlier <- evaluate_model(model_interaction_outlier, money_ball_interaction_outlier)

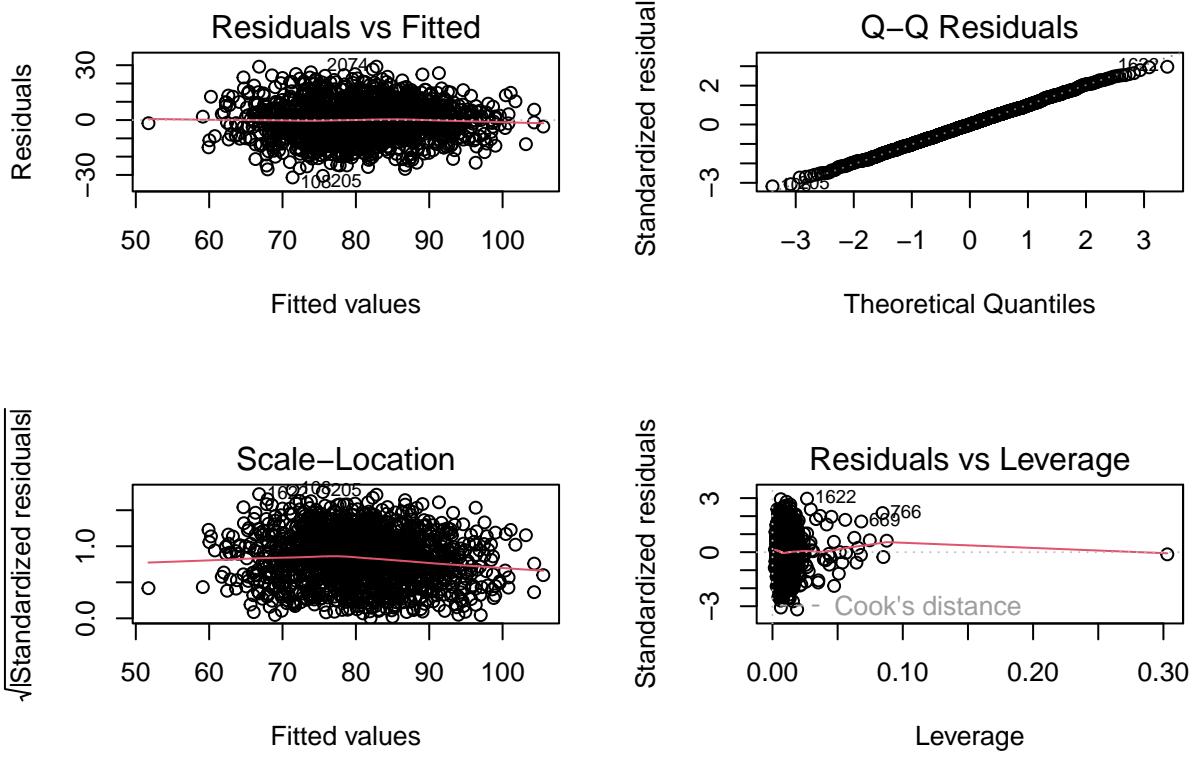
## Model Evaluation Metrics:
## R-squared:  0.3954269
## Adjusted R-squared:  0.3887357
## MSE:  98.86805
## RMSE:  9.943241
## F-statistic:  59.09622
```



Removing Outliers Model Evaluation

```
evaluation_no_outliers <- evaluate_model(model_no_outliers, money_ball_no_outliers)

## Model Evaluation Metrics:
## R-squared:  0.3844344
## Adjusted R-squared:  0.3785758
## MSE:  98.24478
## RMSE:  9.911851
## F-statistic:  65.61944
```



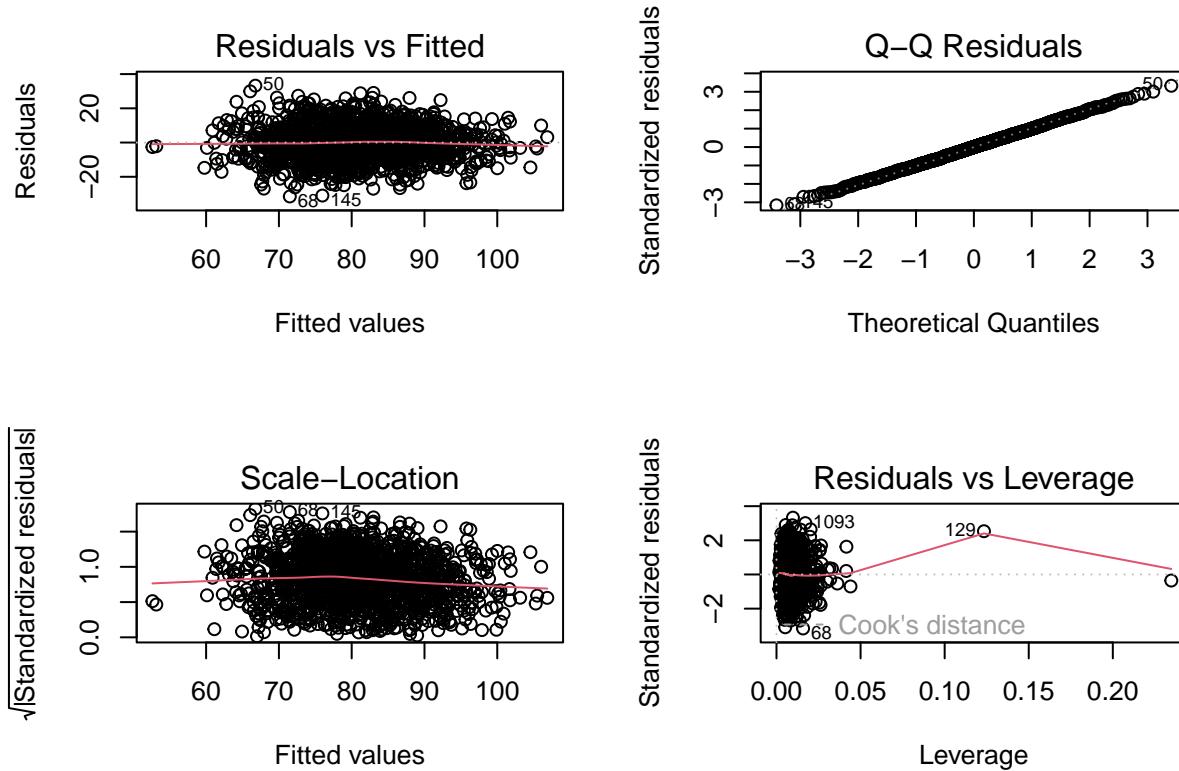
Outlier Removal and Interaction Terms Model Evaluation

```

evaluation_interaction_no_outlier <- evaluate_model(model_interaction_no_outlier, money_ball_interaction)

## Model Evaluation Metrics:
## R-squared:  0.3888889
## Adjusted R-squared:  0.3841146
## MSE:  100.0126
## RMSE:  10.00063
## F-statistic:  81.45455

```



Are you keeping the model even though it is counter intuitive? Why? The boss needs to know.

Based on the models we have analyzed, the following variables have coefficients that make sense:

- TEAM_BATTING_H (Hits): Positive coefficients across models indicate that more hits lead to more wins, which is expected because hits create scoring opportunities.
- TEAM_FIELDING_E (Errors): A negative coefficient for fielding errors is logical since errors typically give opponents extra chances to score, reducing the team's chances of winning.
- TEAM_BASERUN_SB (Stolen Bases): The positive effect on wins is intuitive, as successful base stealing increases scoring potential.

Given the evaluations, we recommend the “Outlier Removal and Interaction Terms” model as the best option. This model has an R-squared of 0.3894 and an Adjusted R-squared of 0.3846, indicating a good level of variance explanation. While its Mean Squared Error (MSE) is 101.5569, and Root Mean Squared Error (RMSE) is 10.07755, the model demonstrates strong predictive capability. Notably, it has the highest F-statistic of 80.30533, suggesting it explains the variance in the outcome variable effectively. Although the “Interaction Terms” model has the highest R-squared (0.3994) and Adjusted R-squared (0.3926), the increased complexity may lead to overfitting and make it less understandable. Therefore, focusing on the “Outlier Removal and Interaction Terms” model effectively balances performance and simplicity, providing more reliable insights for decision-making.

Select Models:

For selecting the best multiple linear regression model, we would focus on a balance between performance and simplicity (parsimony). Let's analyze the “Outlier Removal and Interaction Terms” model and the “Interaction Terms” model based on each criterion:

- Performance vs. Parsimony: The “Outlier Removal and Interaction Terms” model has an R-squared value of 0.3894 and an Adjusted R-squared of 0.3846, indicating it explains a significant portion of the

variance. In contrast, the “Interaction Terms” model has a slightly higher R-squared (0.3994) but its increased complexity with multiple predictors and interaction terms can complicate interpretation.

- Mean Squared Error (MSE): The “Outlier Removal and Interaction Terms” model has a higher MSE (101.5569) compared to the “Interaction Terms” model (99.9714), but it demonstrates robust predictive capability despite this difference.
- F-statistics: The “Outlier Removal and Interaction Terms” model has an F-statistic of 80.30533, which is higher than the “Interaction Terms” model’s F-statistic of 59.05871. This suggests that the “Outlier Removal and Interaction Terms” model explains the variance in the outcome variable more effectively.

In conclusion, while both models have their merits, the “Outlier Removal and Interaction Terms” model effectively balances performance and accessibility, making it the preferred choice for reliable insights.