Homework #1 Assignment Requirements

Warner Alexis

2024-09-17

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 ──
## ✓ dplyr     1.1.4     ✓ readr     2.1.5
## ✓ forcats   1.0.0     ✓ stringr   1.5.1
## ✓ ggplot2   3.5.1     ✓ tibble    3.2.1
## ✓ lubridate 1.9.3     ✓ tidyr     1.3.1
## ✓ purrr     1.0.2
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ 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 0.94 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:tidyr':
##
##     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:tidyr':
##
##     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 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	TEAM_BATTING_HR	TEAM_BATTING_BB	TEAM_BATTING_SO
Min. : 1.0	Min. : 0.00	Min. : 891	Min. : 69.0	Min. : 0.00	Min. : 0.00	Min. : 0.0	Min. : 0.0
1st Qu.: 630.8	1st Qu.: 71.00	1st Qu.:1383	1st Qu.:208.0	1st Qu.: 34.00	1st Qu.: 42.00	1st Qu.:451.0	1st Qu.: 548.0
Median :1270.5	Median : 82.00	Median :1454	Median :238.0	Median : 47.00	Median :102.00	Median :512.0	Median : 750.0
Mean :1268.5	Mean : 80.79	Mean :1469	Mean :241.2	Mean : 55.25	Mean : 99.61	Mean :501.6	Mean : 735.6
3rd Qu.:1915.5	3rd Qu.: 92.00	3rd Qu.:1537	3rd Qu.:273.0	3rd Qu.: 72.00	3rd Qu.:147.00	3rd Qu.:580.0	3rd Qu.: 930.0
Max. :2535.0	Max. :146.00	Max. :2554	Max. :458.0	Max. :223.00	Max. :264.00	Max. :878.0	Max. :1399.0
NA	NA	NA	NA	NA	NA	NA	NA's :102

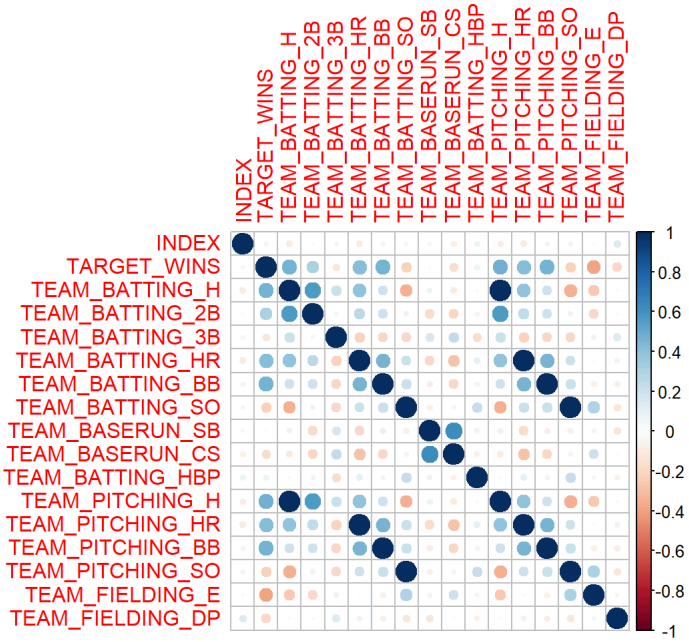
```
str(money_ball_train)
```

```
## '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 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 ...
```

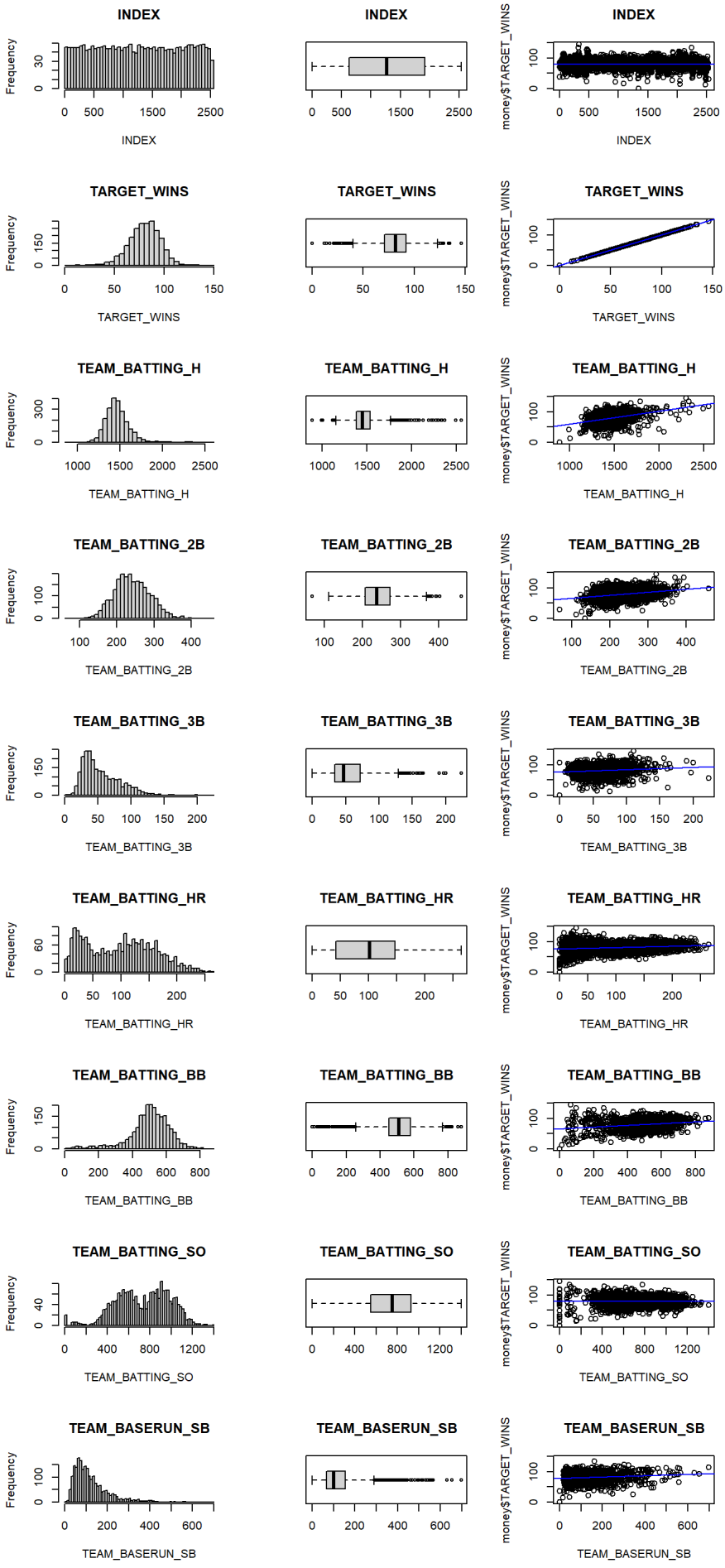
Data Preparation

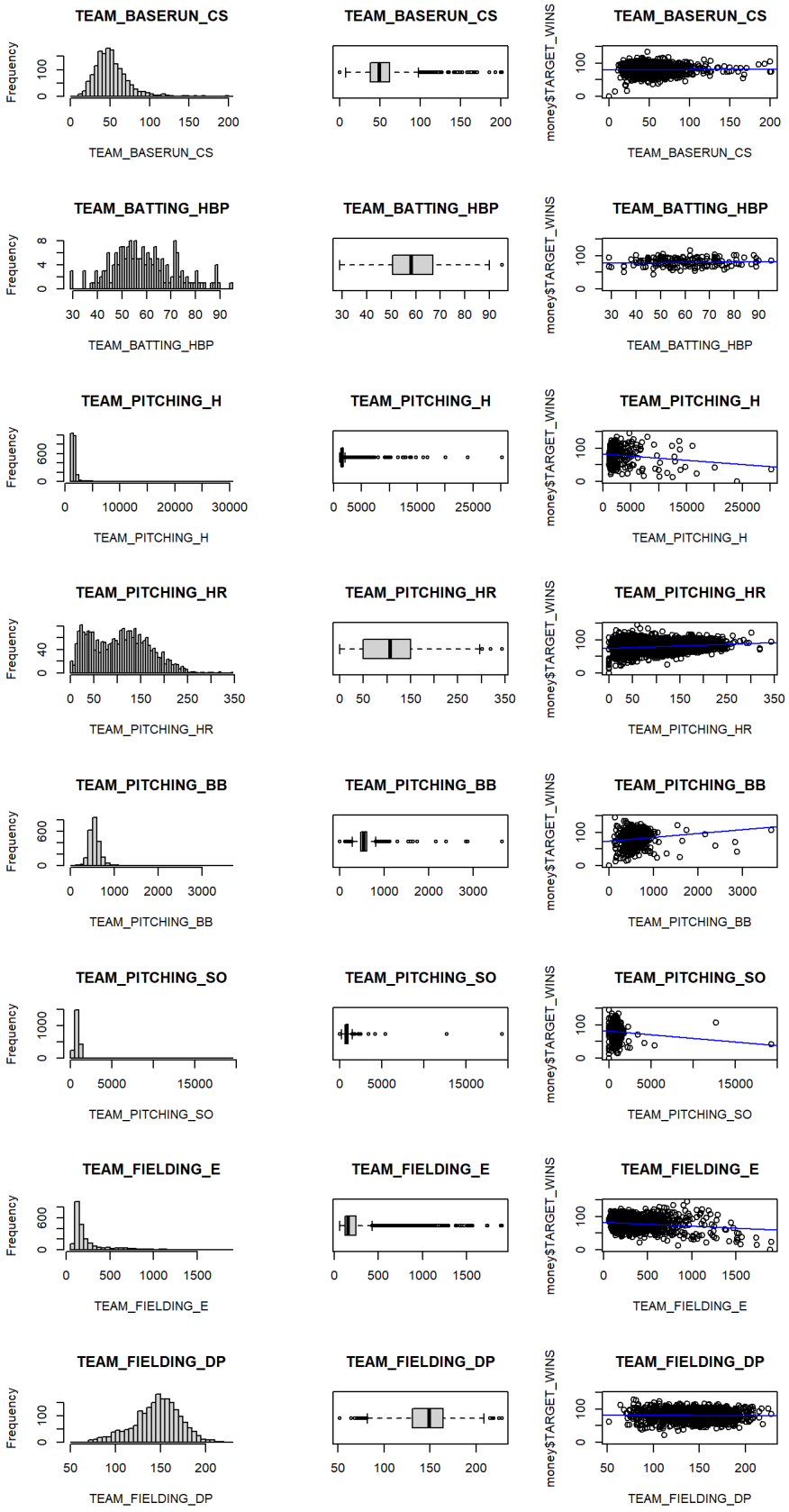
We are going to check distribution for all the columns and outliers that are present in the data set. The Chart below shows the percentage accounted for missing values in the data set. We notice that TEAM_BATTING_HBP has the highest number of missing values but, the mean, the median, the max are around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

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




```
money <- money_ball_train
# Create missing data flags

# Create missing data flags
#missing_flag <- ifelse(is.na(money_ball_train$TEAM_BATTING_HBP), 1, 2)
#money_ball_train$missing_flag <- missing_flag


par(mfrow=c(3,3))
# Create distribution plot to check outliers
for (i in 1:17) {
  hist(money[,i],main=names(money[i]),xlab=names(money[i]),breaks = 51)
  boxplot(money[,i], main=names(money[i]), type="l",horizontal = TRUE)

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


INDEX
INDEX
INDEX
Frequency
money$TARGET_WINS
INDEX
TARGET_WINS
TARGET_WINS
TARGET_WINS
TARGET_WINS
TEAM_BATTING_H
TEAM_BATTING_H
TEAM_BATTING_H
TEAM_BATTING_H
TEAM_BATTING_2B
TEAM_BATTING_2B
TEAM_BATTING_2B
TEAM_BATTING_2B
TEAM_BATTING_3B
TEAM_BATTING_3B
TEAM_BATTING_3B
TEAM_BATTING_3B
TEAM_BATTING_HR
TEAM_BATTING_HR
TEAM_BATTING_HR
TEAM_BATTING_HR
TEAM_BATTING_BB
TEAM_BATTING_BB
TEAM_BATTING_BB
TEAM_BATTING_BB
TEAM_BATTING_SO
TEAM_BATTING_SO
TEAM_BATTING_SO
TEAM_BATTING_SO
TEAM_BASERUN_SB
TEAM_BASERUN_SB
TEAM_BASERUN_SB
TEAM_BASERUN_SB


TEAM_BASERUN_CS
Frequency
TEAM_BASERUN_CS
money$TARGET_WINS
TEAM_BATTING_HBP
TEAM_PITCHING_H
TEAM_PITCHING_HR
TEAM_PITCHING_BB
TEAM_PITCHING_SO
TEAM_FIELDING_E
TEAM_FIELDING_DP

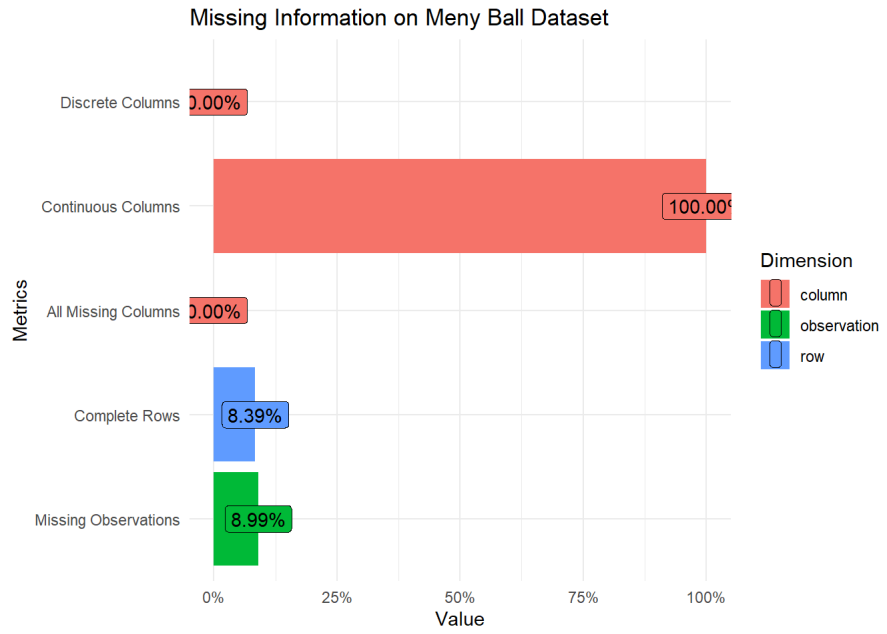
The relationship between the target variable and the predictors is not particularly strong, but there are statistically significant relationships with certain variables. The calculate_correlations_with_pvalues function computes the correlation coefficients and p-values between a given target variable 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)
```

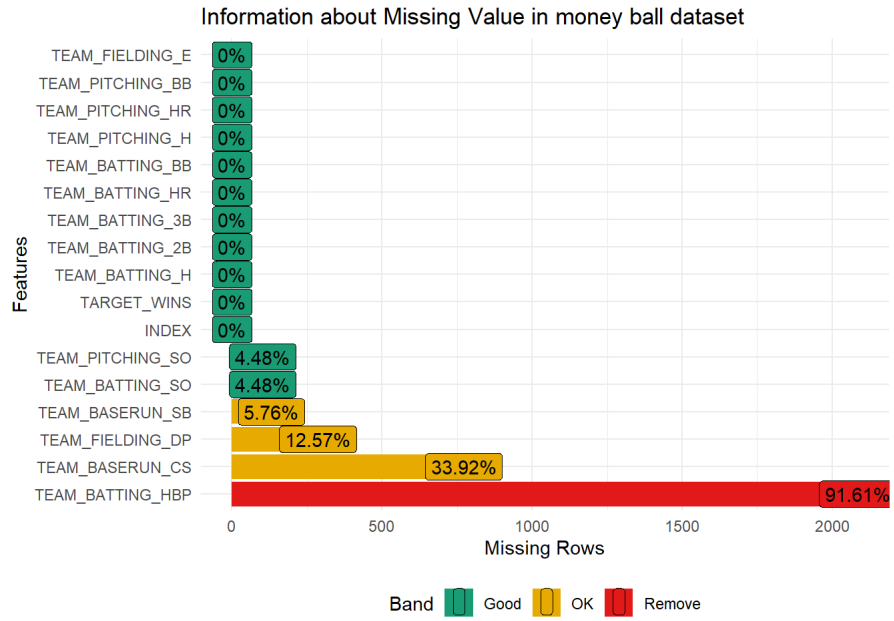
```
##              Predictor Correlation       PValue
## cor              INDEX -0.04895047 0.5012836479
## cor1    TEAM_BATTING_H  0.46994665 0.0000000000
## cor2   TEAM_BATTING_2B  0.31298400 0.0000104138
## cor3   TEAM_BATTING_3B -0.12434586 0.0865523694
## cor4   TEAM_BATTING_HR  0.42241683 0.0000000012
## 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 dive into the Missing values.

```
#
plot_intro(money_ball_train, title = 'Missing Information on Meny 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 = theme_minimal())
```



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.

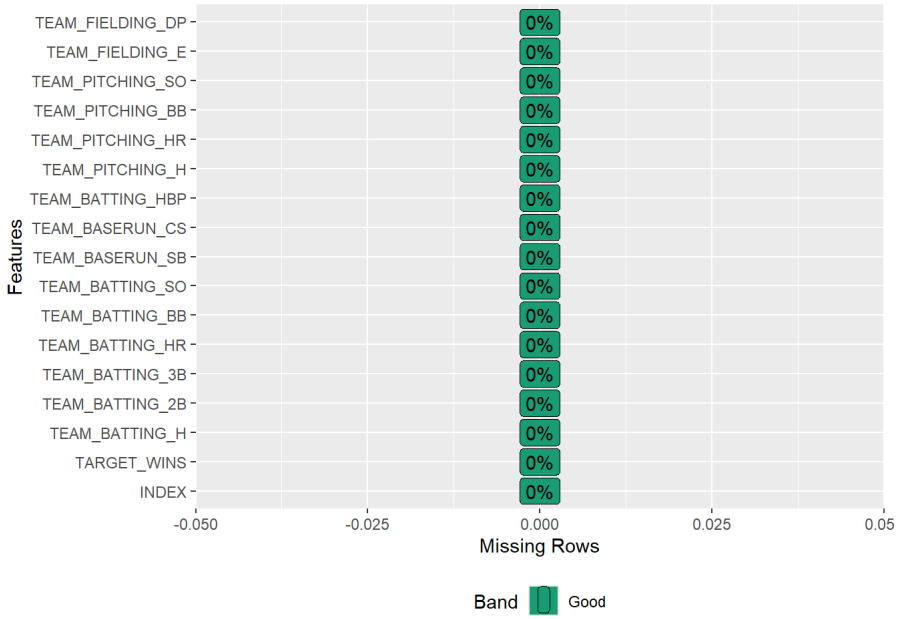
Handles Different Types of Missing Data: It can handle data that is Missing Completely at Random (MCAR), Missing at Random (MAR), and even some cases of Missing Not at Random (MNAR). **Maintains Variability**: Unlike simple imputation methods (mean, median), predictive imputation maintains the natural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

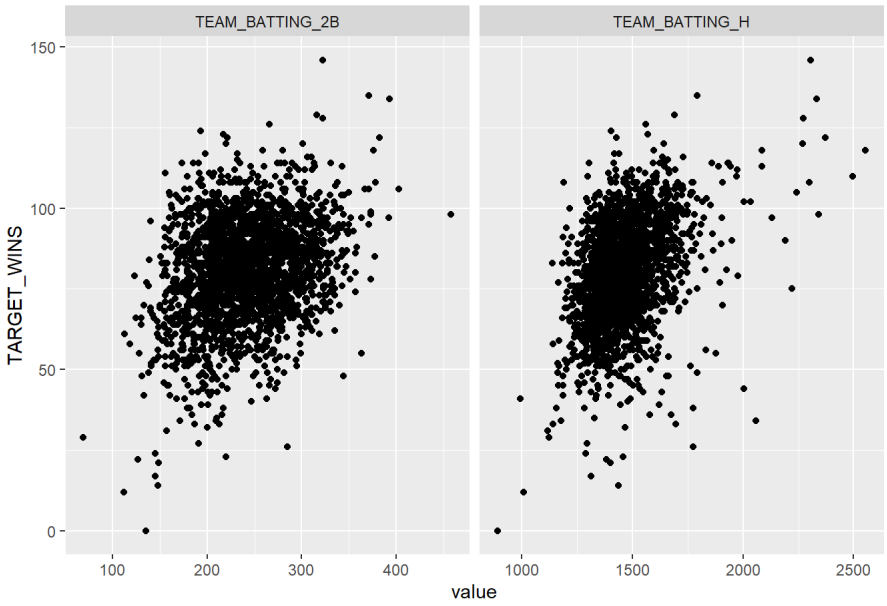
```
##
##  iter imp variable
##    1   1  TEAM_BATTING_SO  TEAM_BASERUN_SB  TEAM_BASERUN_CS  TEAM_BATTING_HBP  TEAM_PITCHING_SO  TEAM_FIELDING_DP
##    1   2  TEAM_BATTING_SO  TEAM_BASERUN_SB  TEAM_BASERUN_CS  TEAM_BATTING_HBP  TEAM_PITCHING_SO  TEAM_FIELDING_DP
##    1   3  TEAM_BATTING_SO  TEAM_BASERUN_SB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

```
## Warning: Number of logged events: 25
```

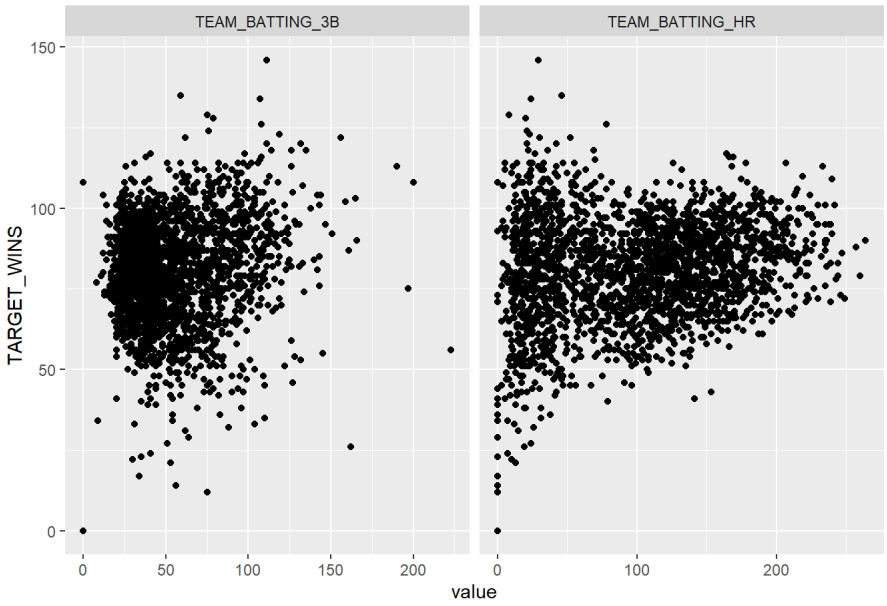
```
completed_data <- complete(data1)
# View missing value distribution
plot_missing(completed_data)
```




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

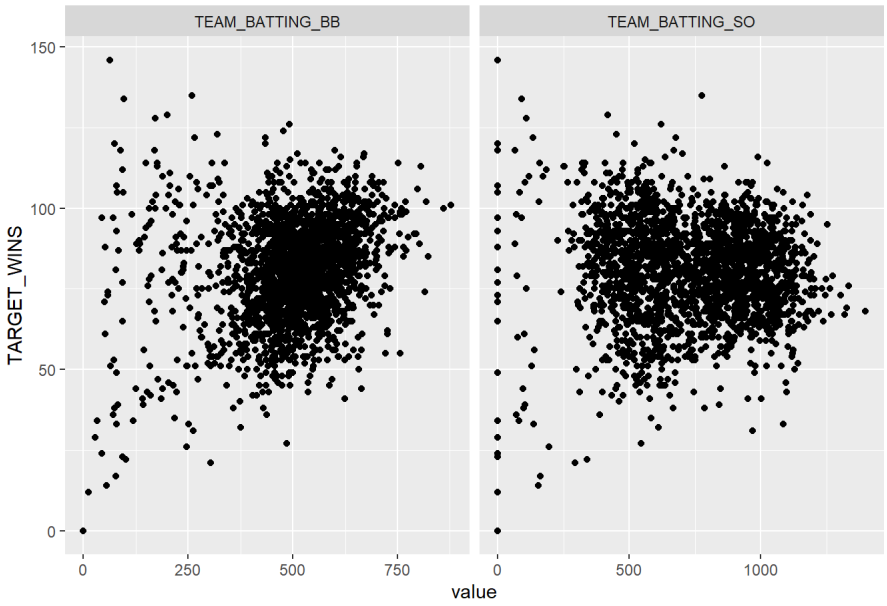


Page 1

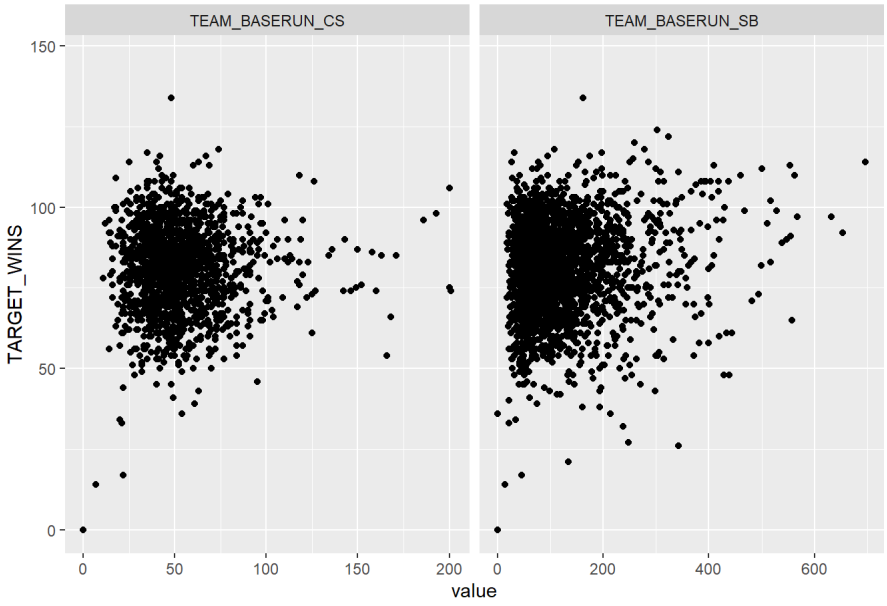


Page 2

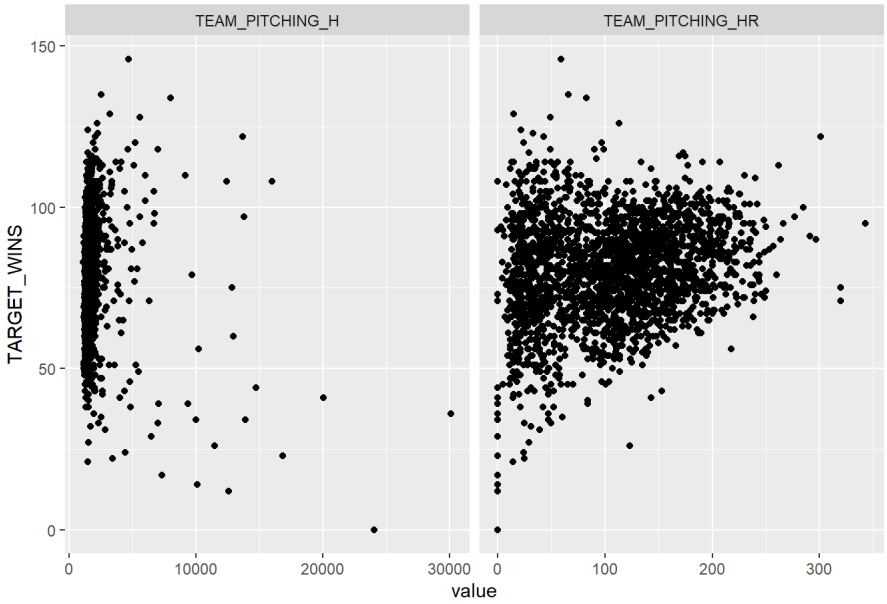
```
## Warning: Removed 102 rows containing missing values or values outside the scale range
## (`geom_point()`).
```




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

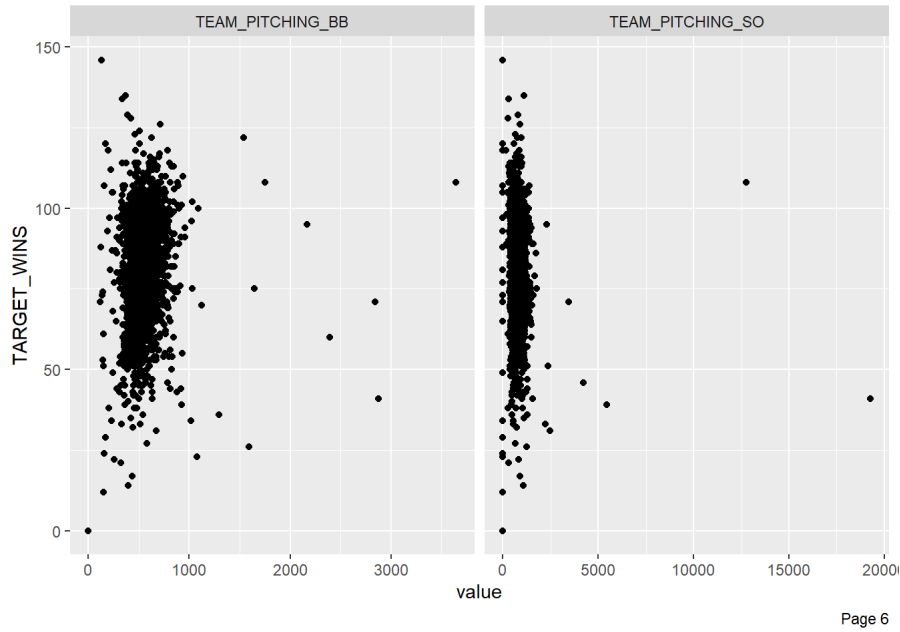


Page 4



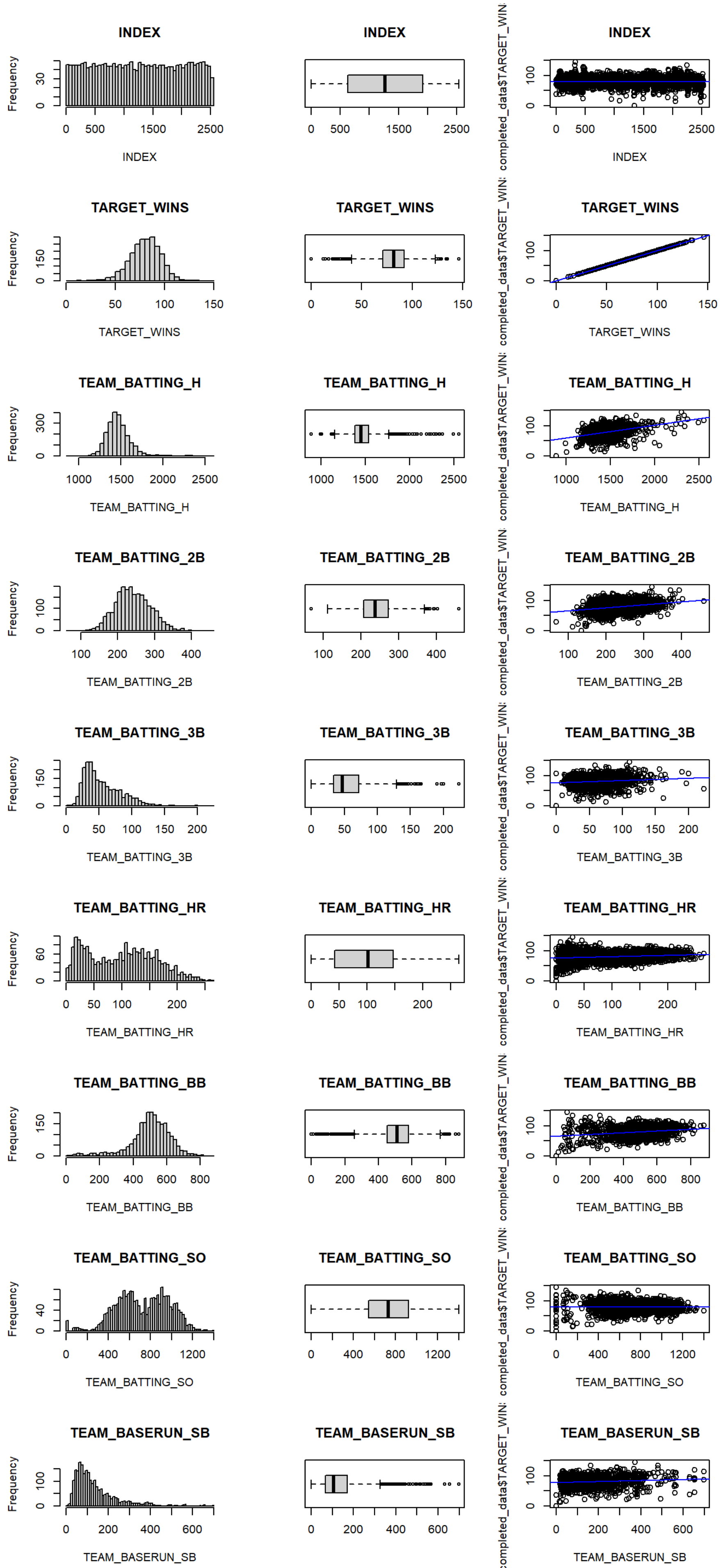
Page 5

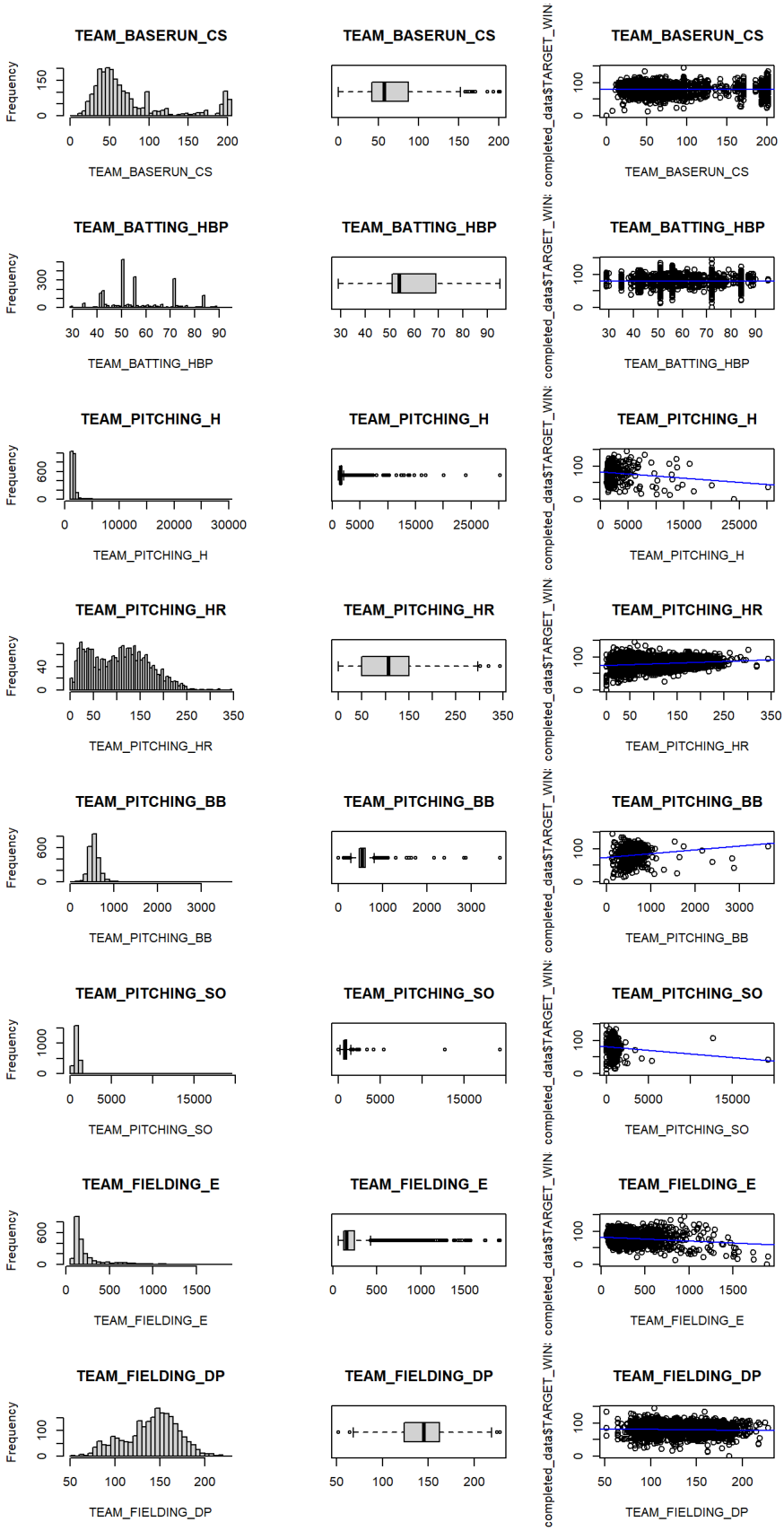
```
## Warning: Removed 102 rows containing missing values or values outside the scale range
## (`geom_point()`).
```




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


```
# 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")
}
```

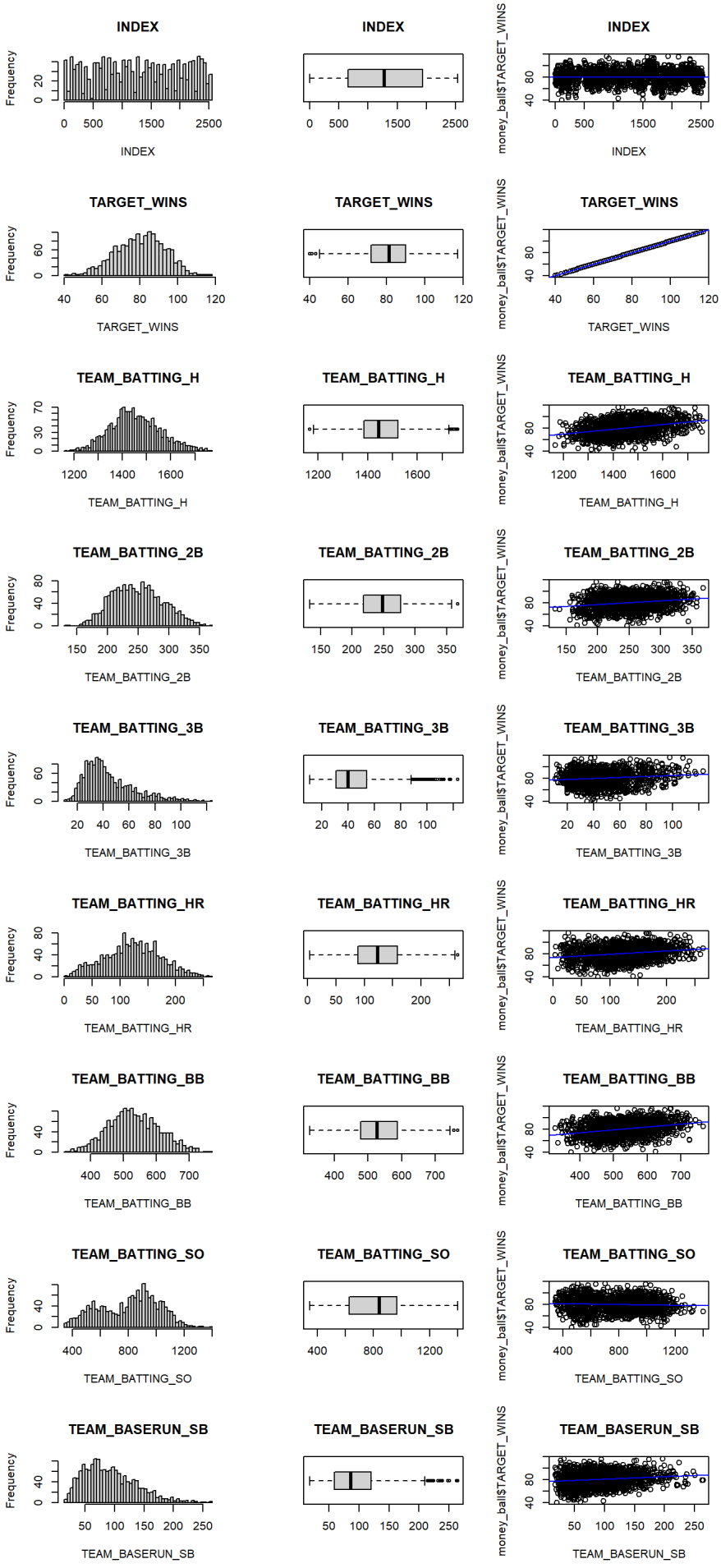

INDEX
TARGET_WINS
TEAM_BATTING_H
TEAM_BATTING_2B
TEAM_BATTING_3B
TEAM_BATTING_HR
TEAM_BATTING_BB
TEAM_BATTING_SO
TEAM_BASERUN_SB
Frequency
completed_data$TARGET_WIN: completed_data$TARGET_WIN: completed_data$TARGET_WIN:

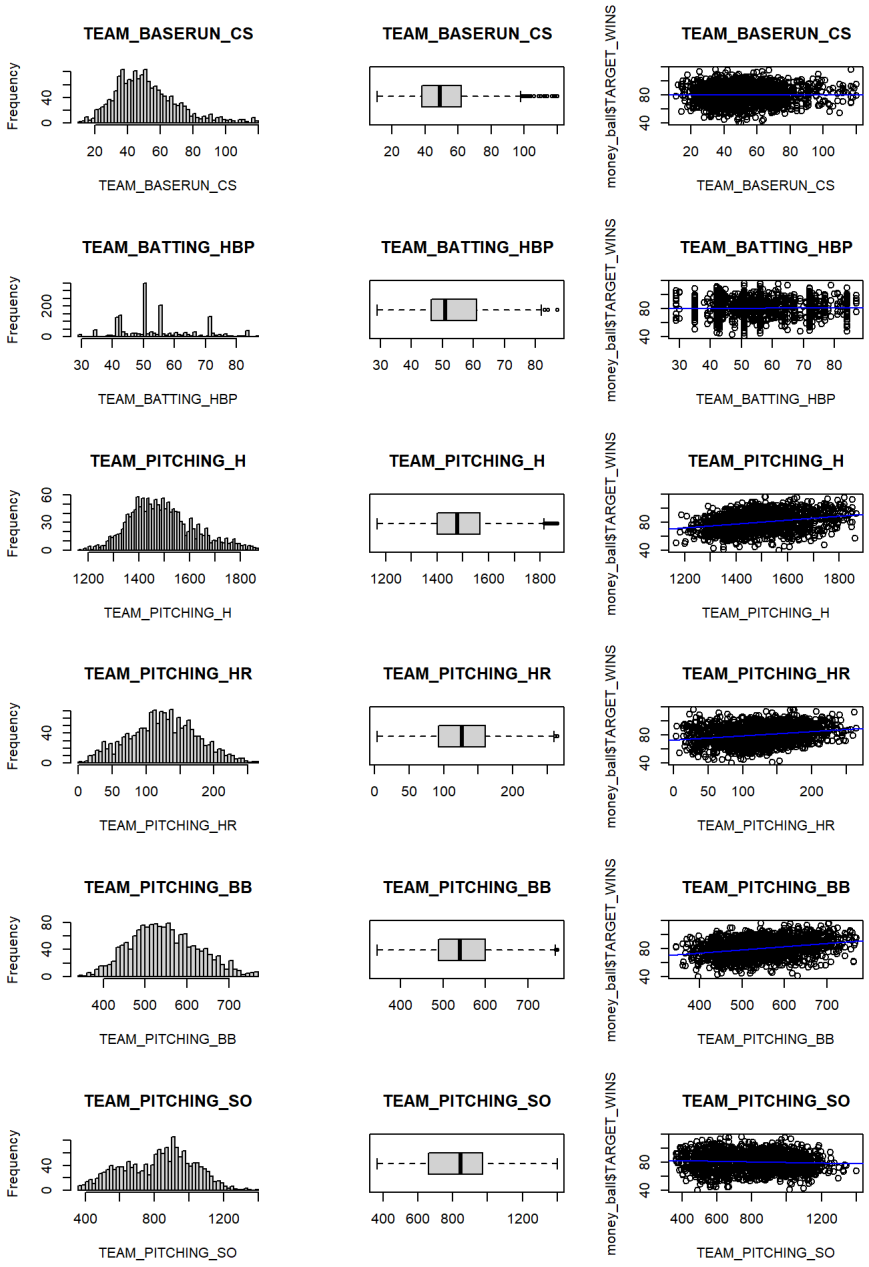

TEAM_BASERUN_CS
TEAM_BASERUN_CS
TEAM_BASERUN_CS
Frequency
TEAM_BASERUN_CS
TEAM_BATTING_HBP
TEAM_BATTING_HBP
TEAM_BATTING_HBP
Frequency
TEAM_BATTING_HBP
TEAM_PITCHING_H
TEAM_PITCHING_H
TEAM_PITCHING_H
Frequency
TEAM_PITCHING_H
TEAM_PITCHING_HR
TEAM_PITCHING_HR
TEAM_PITCHING_HR
Frequency
TEAM_PITCHING_HR
TEAM_PITCHING_BB
TEAM_PITCHING_BB
TEAM_PITCHING_BB
Frequency
TEAM_PITCHING_BB
TEAM_PITCHING_SO
TEAM_PITCHING_SO
TEAM_PITCHING_SO
Frequency
TEAM_PITCHING_SO
TEAM_FIELDING_E
TEAM_FIELDING_E
TEAM_FIELDING_E
Frequency
TEAM_FIELDING_E
TEAM_FIELDING_DP
TEAM_FIELDING_DP
TEAM_FIELDING_DP
Frequency
TEAM_FIELDING_DP
completed_data$TARGET_WIN: completed_data$TARGET_WIN: completed_data$TARGET_WIN: completed_data$TARGET_WIN: completed_data$TARGET_WIN: completed_data$TARGET_WIN

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 * IQR` and `Q3 + 1.5 * 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.

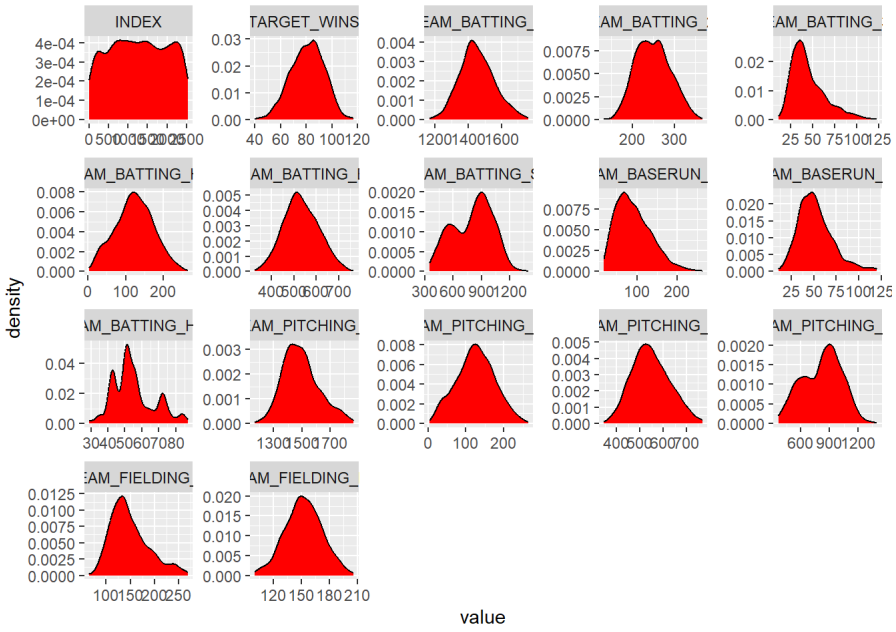
```
remove_outliers_df <- function(df) {
  # Loop through each numeric column in the data frame
  for (col in names(df)) {
    if (is.numeric(df[[col]])) {
      # Calculate Q1, Q3, and IQR for the column
      Q1 <- quantile(df[[col]], 0.25, na.rm = TRUE)
      Q3 <- quantile(df[[col]], 0.75, na.rm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

}
```


INDEX
INDEX
INDEX
Frequency
money_ball$TARGET_WINS
INDEX
TARGET_WINS
TARGET_WINS
TARGET_WINS
TARGET_WINS
TEAM_BATTING_H
TEAM_BATTING_H
TEAM_BATTING_H
TEAM_BATTING_H
TEAM_BATTING_2B
TEAM_BATTING_2B
TEAM_BATTING_2B
TEAM_BATTING_2B
TEAM_BATTING_3B
TEAM_BATTING_3B
TEAM_BATTING_3B
TEAM_BATTING_3B
TEAM_BATTING_HR
TEAM_BATTING_HR
TEAM_BATTING_HR
TEAM_BATTING_HR
TEAM_BATTING_BB
TEAM_BATTING_BB
TEAM_BATTING_BB
TEAM_BATTING_BB
TEAM_BATTING_SO
TEAM_BATTING_SO
TEAM_BATTING_SO
TEAM_BATTING_SO
TEAM_BASERUN_SB
TEAM_BASERUN_SB
TEAM_BASERUN_SB
TEAM_BASERUN_SB




```
# 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  0.02506545 0.3218626747
## cor1    TEAM_BATTING_H  0.34428440 0.0000000000
## cor2   TEAM_BATTING_2B  0.20336876 0.0000000000
## cor3   TEAM_BATTING_3B  0.11990206 0.0000019826
## cor4   TEAM_BATTING_HR  0.22954667 0.0000000000
## cor5   TEAM_BATTING_BB  0.29855326 0.0000000000
## cor6   TEAM_BATTING_SO -0.05614231 0.0264010760
## cor7   TEAM_BASERUN_SB  0.15853922 0.0000000003
## cor8   TEAM_BASERUN_CS  0.01168791 0.6441723107
## cor9  TEAM_BATTING_HBP  0.03184740 0.2081042079
## cor10  TEAM_PITCHING_H  0.27825105 0.0000000000
## cor11 TEAM_PITCHING_HR  0.23324536 0.0000000000
## cor12 TEAM_PITCHING_BB  0.28794611 0.0000000000
## cor13 TEAM_PITCHING_SO -0.06166403 0.0147269140
## cor14  TEAM_FIELDING_E -0.19302535 0.0000000000
## cor15 TEAM_FIELDING_DP -0.08236666 0.0011128370
```

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
## -37.251  -7.089   0.050   7.596  37.828
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)     -7.949679   6.085601  -1.306  0.19191
## TEAM_BATTING_H   0.068153   0.005098  13.368  < 2e-16 ***
## TEAM_BATTING_2B -0.041621   0.013449  -3.095  0.00205 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11.29 on 647 degrees of freedom
##   (914 observations deleted due to missingness)
## Multiple R-squared:  0.2494, Adjusted R-squared:  0.2471
## F-statistic: 107.5 on 2 and 647 DF,  p-value: < 2.2e-16
```

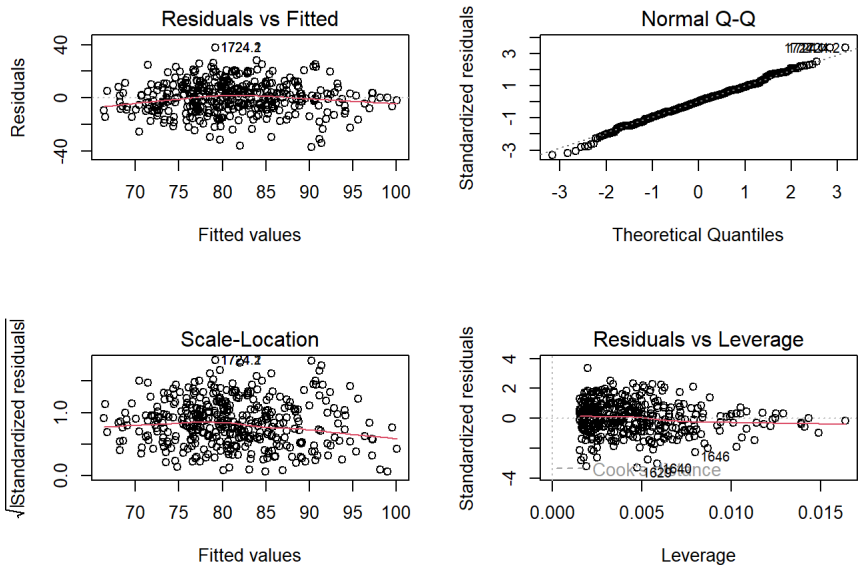
```
# Model 1 evaluation
mse <- mean(model$residuals^2)
r_squared <- summary(model)$r.squared
f_stat <- summary(model)$fstatistic[1]
print(paste("MSE:", mse, "R-squared:", r_squared, "F-statistic:", f_stat))
```

```
## [1] "MSE: 126.838066656534 R-squared: 0.249428438697806 F-statistic: 107.504872391845"
```

```
predictions <- predict(model, newdata = money_ball_eval)
head(predictions)
```

```
##        1        2        3        4        5        6
## 67.37181 68.98044 79.50720 84.07703 82.08244 79.75481
```

```
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 dataframe 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 dataframe 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
## -39.397  -7.821   0.370   7.979  35.242
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  80.8549     0.2839 284.826  < 2e-16 ***
## PC1           0.2566     0.1252   2.050  0.04055 *
## PC2           2.7532     0.1629  16.900  < 2e-16 ***
## PC3           0.9652     0.2068   4.668 3.30e-06 ***
## PC4           2.3629     0.2237  10.564  < 2e-16 ***
## PC5           0.1175     0.2890   0.407  0.68434
## PC6           0.9438     0.2973   3.175  0.00153 **
## PC7          -0.9658     0.3225  -2.995  0.00279 **
## PC8           3.0046     0.4094   7.339 3.46e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11.23 on 1555 degrees of freedom
## Multiple R-squared:  0.2419, Adjusted R-squared:  0.238
## F-statistic: 62.03 on 8 and 1555 DF,  p-value: < 2.2e-16
```

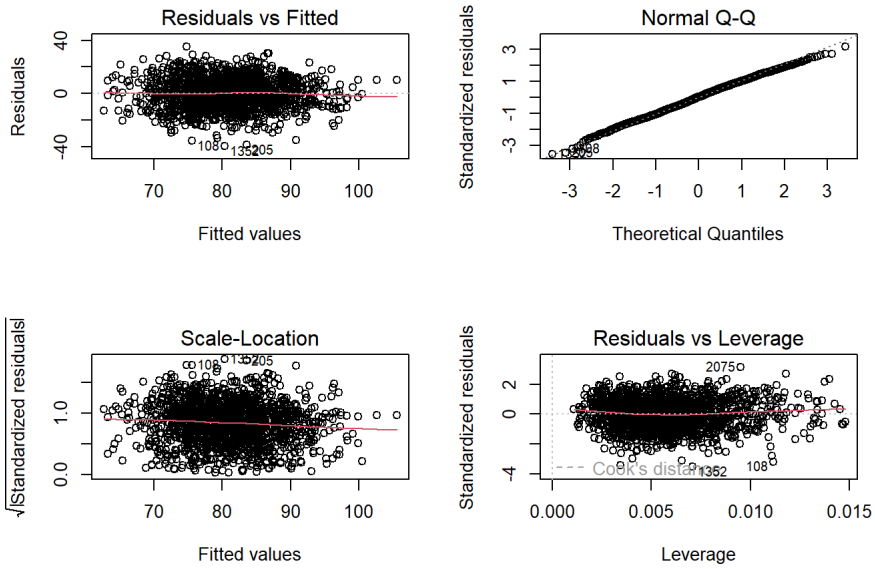
```
mse1 <- mean(model1$residuals^2)
r_squared1 <- summary(model1)$r.squared
f_stat1 <- summary(model1)$fstatistic[1]
print(paste("MSE:", mse1, "R-squared:", r_squared1, "F-statistic:", f_stat1))
```

```
## [1] "MSE: 125.309693892351 R-squared: 0.241933992628414 F-statistic: 62.0340700675911"
```

```
predictions1 <- predict(pca_model, newdata = money_ball_eval)
head(predictions1)
```

```
##           PC1      PC2      PC3      PC4      PC5       PC6       PC7      PC8
## [1,]       NA       NA       NA       NA       NA        NA        NA       NA
## [2,]       NA       NA       NA       NA       NA        NA        NA       NA
## [3,]       NA       NA       NA       NA       NA        NA        NA       NA
## [4,] 310.9073 1702.555 798.4776 469.1161 69.14733 -106.4234 -92.52125 170.7631
## [5,]       NA       NA       NA       NA       NA        NA        NA       NA
## [6,]       NA       NA       NA       NA       NA        NA        NA       NA
##            PC9      PC10     PC11      PC12      PC13      PC14     PC15
## [1,]        NA        NA       NA        NA        NA        NA       NA
## [2,]        NA        NA       NA        NA        NA        NA       NA
## [3,]        NA        NA       NA        NA        NA        NA       NA
## [4,] -754.3944 -34.41136 397.8718 -1554.106 -169.3464 -109.2177 25.09542
## [5,]        NA        NA       NA        NA        NA        NA       NA
## [6,]        NA        NA       NA        NA        NA        NA       NA
##          PC16
## [1,]       NA
## [2,]       NA
## [3,]       NA
## [4,] 58.55733
## [5,]       NA
## [6,]       NA
```

```
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[trainIndex, ]
test_data <- money_ball[-trainIndex, ]


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
##
## 1252 samples
##   16 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 1002, 1001, 1002, 1001, 1002
## Resampling results:
##
##   RMSE      Rsquared   MAE
##   10.31872  0.3729891  8.25913
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```
print(model3$results)
```

```
##   intercept     RMSE  Rsquared     MAE    RMSESD RsquaredSD     MAESD
## 1      TRUE 10.31872 0.3729891 8.25913 0.4692538 0.02691546 0.3648173
```

```
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: 0.388081650887269 F-statistic: 48.9526951942122"
```

```
predictions3 <- predict(model3, newdata = money_ball_eval)
head(predictions3)
```

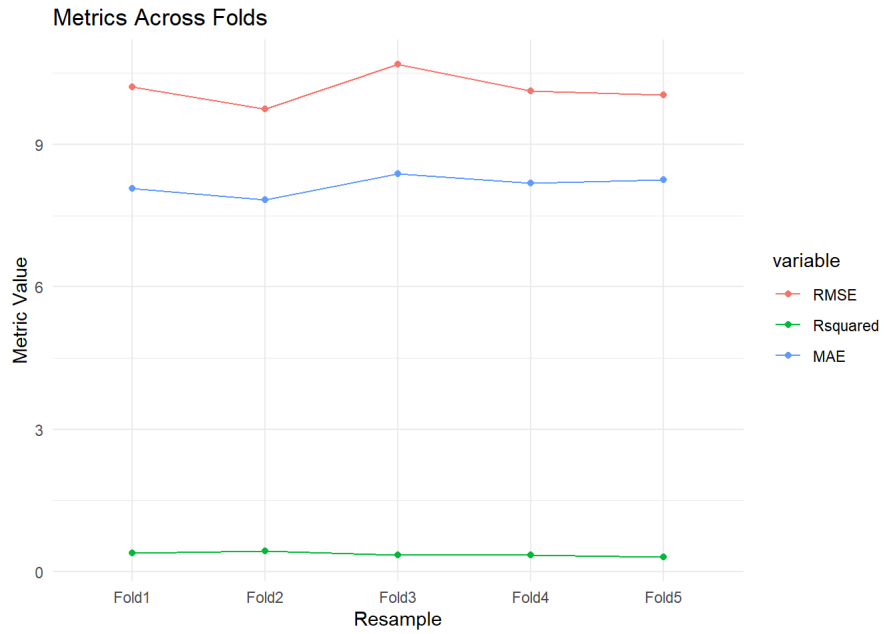
```
##        4       18       25       26       64       65
## 83.28540 76.95452 78.79971 85.67588 83.79540 83.97394
```

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.

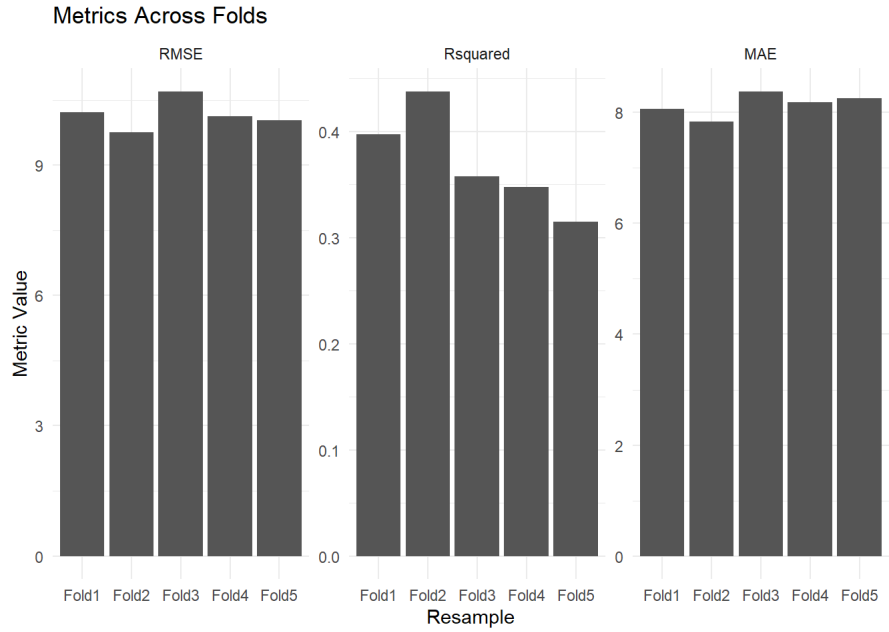
```
# Summary of the cross-validation results
model3$resample
```

```
##        RMSE  Rsquared      MAE Resample
## 1 10.397129 0.3917399 8.575679    Fold1
## 2  9.543752 0.4071564 7.650937    Fold2
## 3 10.580207 0.3531944 8.226493    Fold3
## 4 10.771898 0.3712323 8.486923    Fold4
## 5 10.300611 0.3416227 8.355617    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()
```




```
# 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()
```



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.