

Homework #1 Assignment Requirements

Warner Alexis

2024-09-17

Introduction

In this assignment, I will be exploring, analyzing and modeling 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 ...
## $ 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

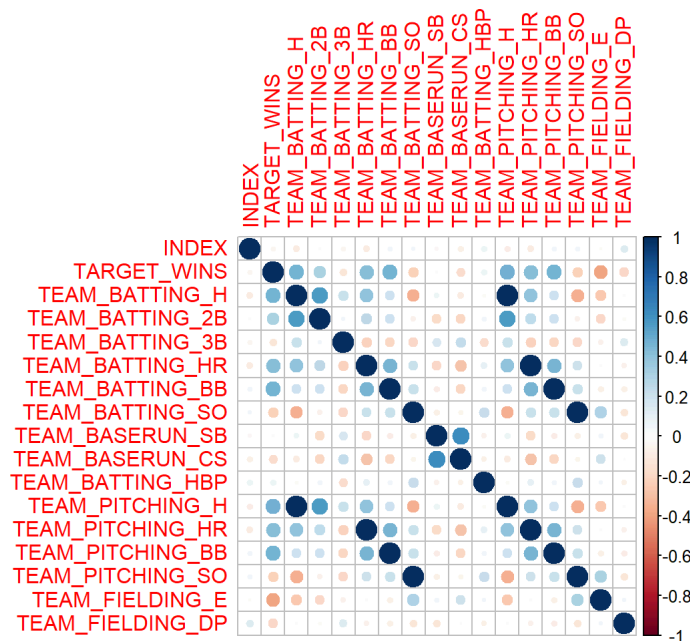
```
str(money_ball_train)
```

```
## '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 ...
## $ 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 centerly.

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

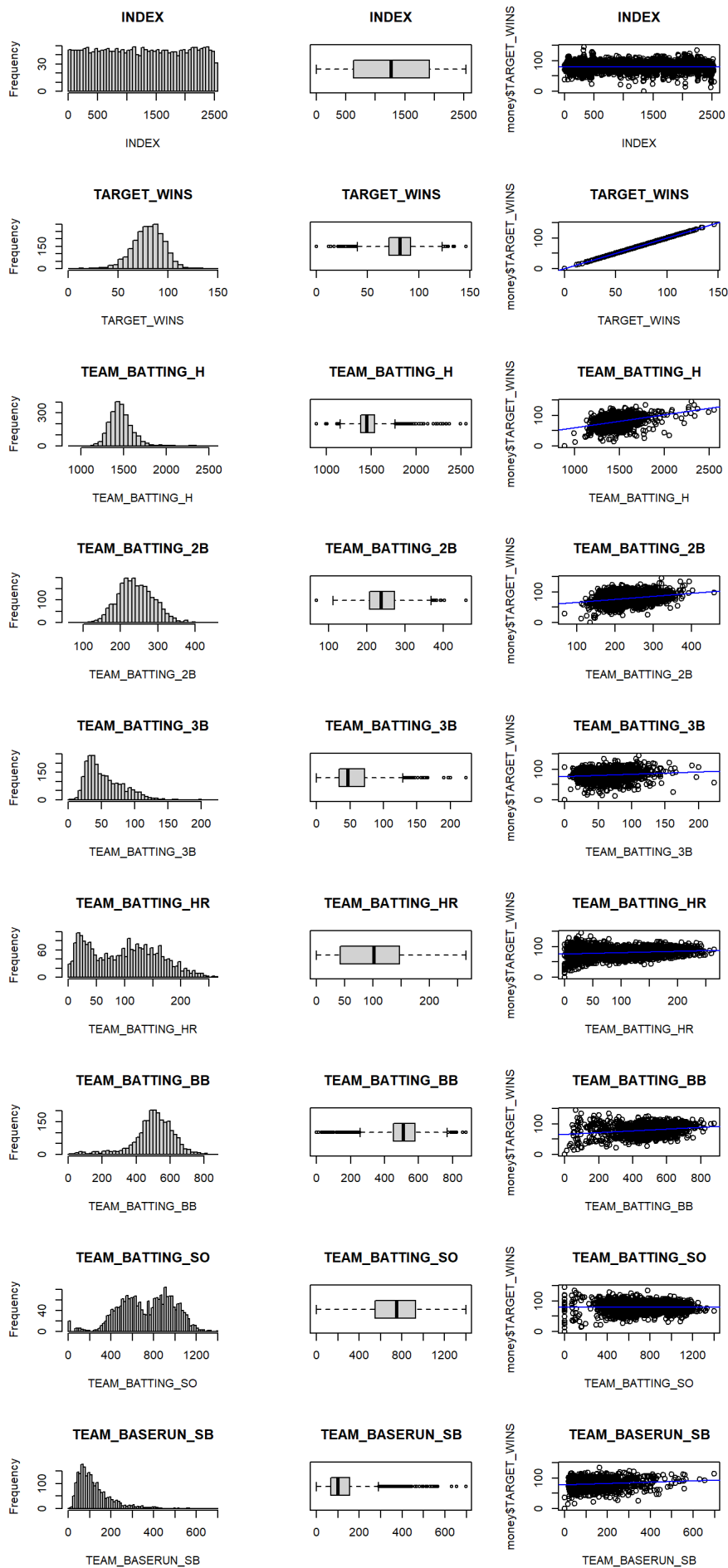


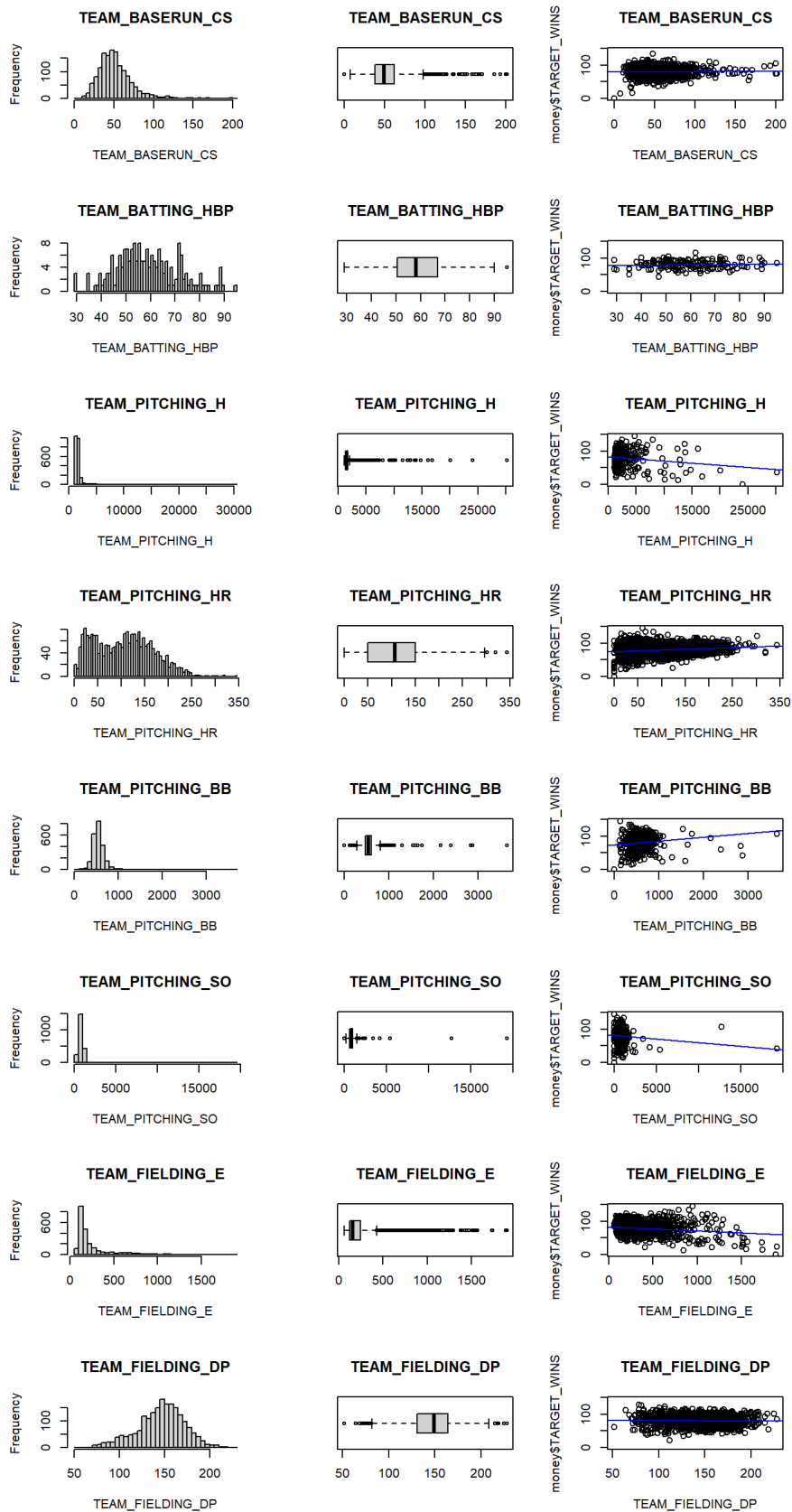
```
money <- money_ball_train
# Create missing data flags

# 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")
}
```





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")

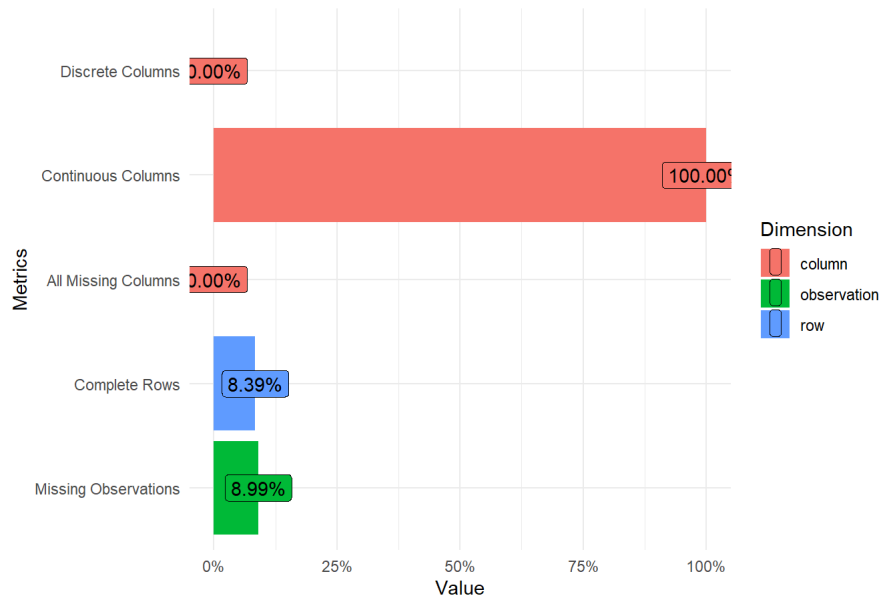
# View the results
print(correlation_results)
```

```
##           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.

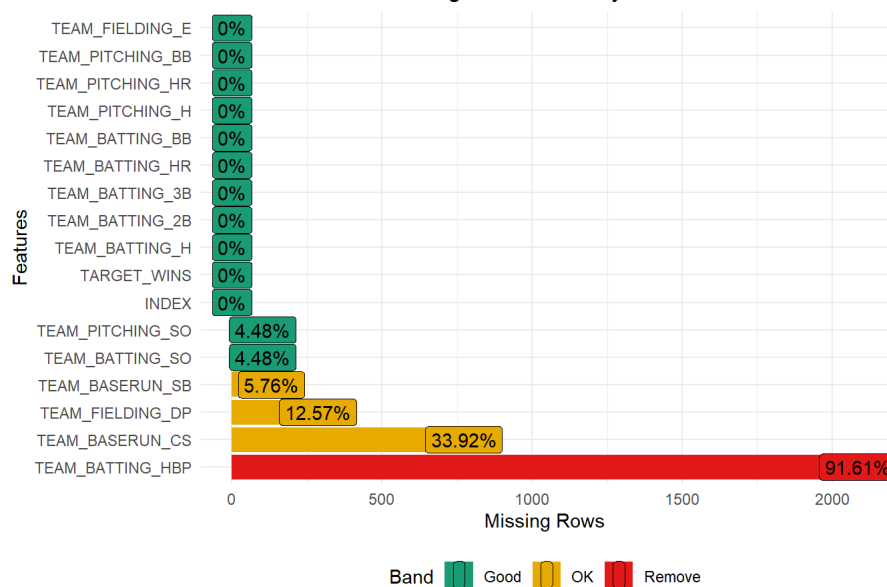
```
#
plot_intro(money_ball_train, title = 'Missing Information on Meny Ball Dataset',
           ggtheme = theme_minimal())
```

Missing Information on Meny Ball Dataset



```
# Plot missing volume in Column
plot_missing(money_ball_train,title = 'Information about Missing Value in money ball dataset',ggtheme = theme_minimal())
```

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?

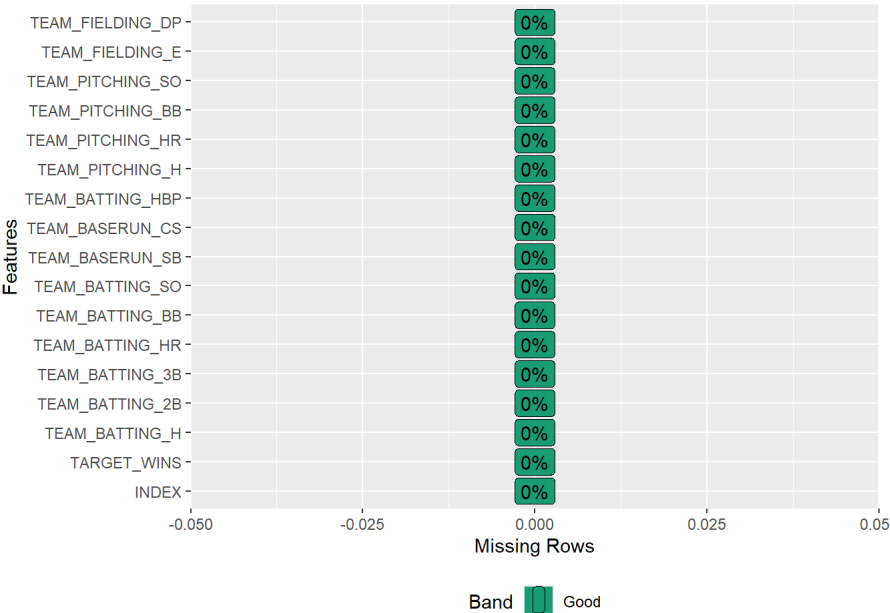
Preserves Relationships: Predictive imputation uses the relationships between variables to estimate missing values, which can lead to more accurate and reliable data sets.

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.

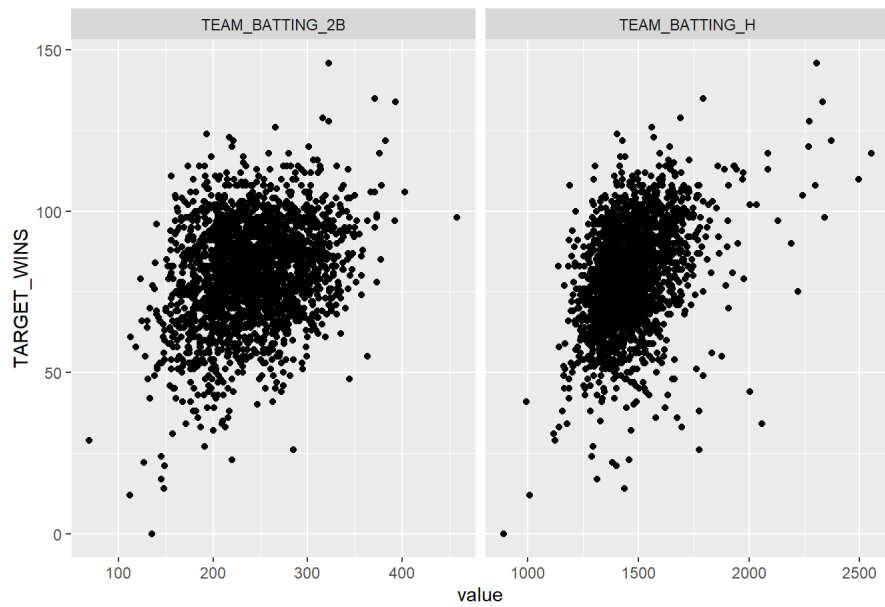
```
# re-assign moneyball
money <- money_ball_train
# create data set with with predictive imputable
# Compute multiple imputation
data1 <- mice(money, method = 'pmm', m=5)
```



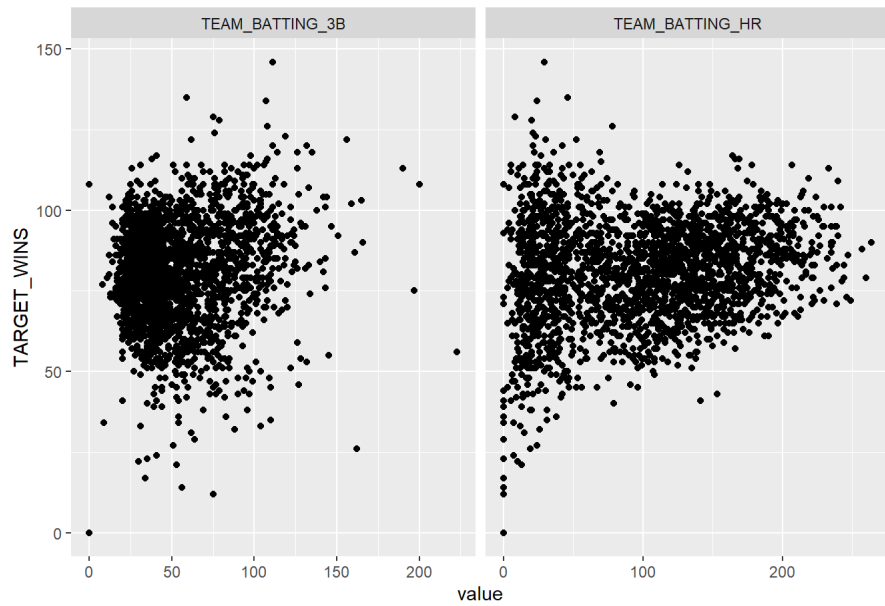
```
money_ball_eval <- complete(money_ball_eval)
# View missing value distribution
plot missing(completed data)
```



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

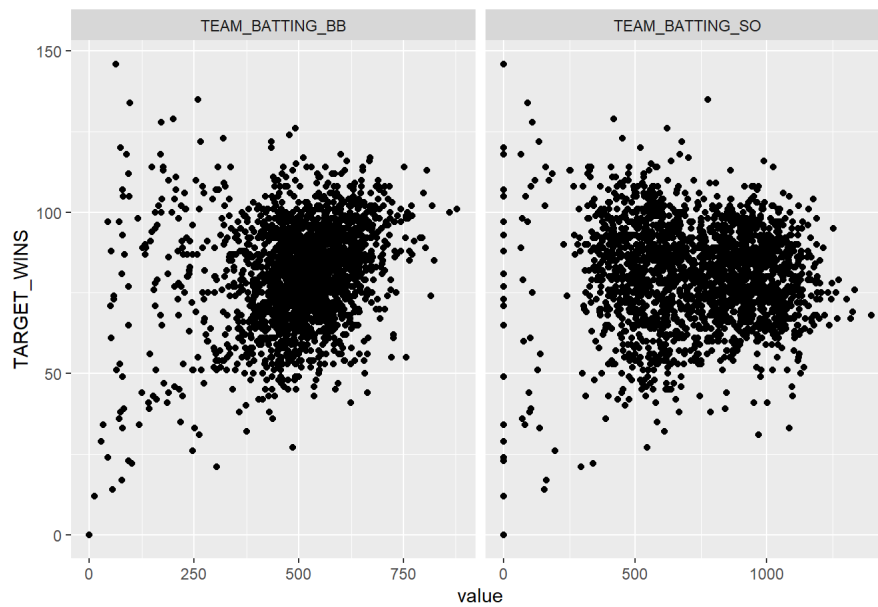


Page 1



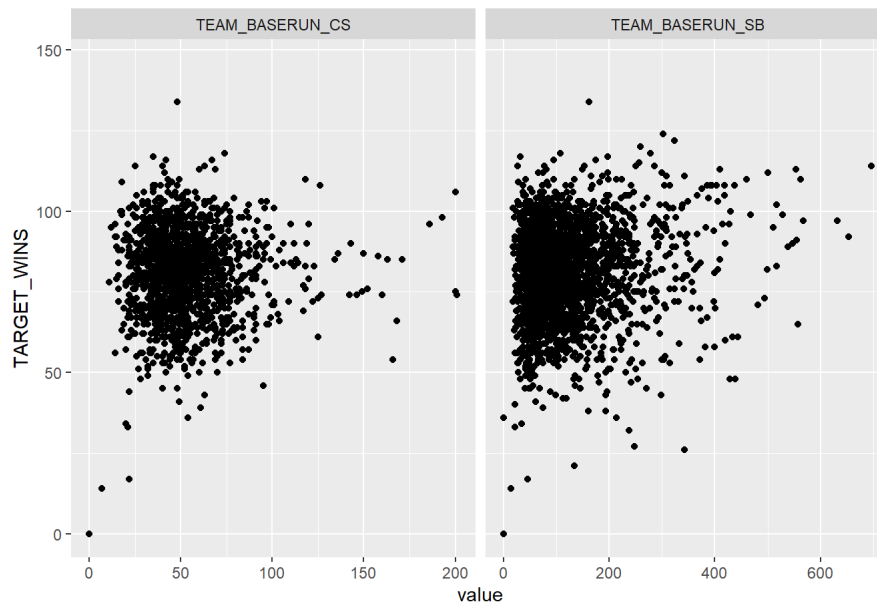
Page 2

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

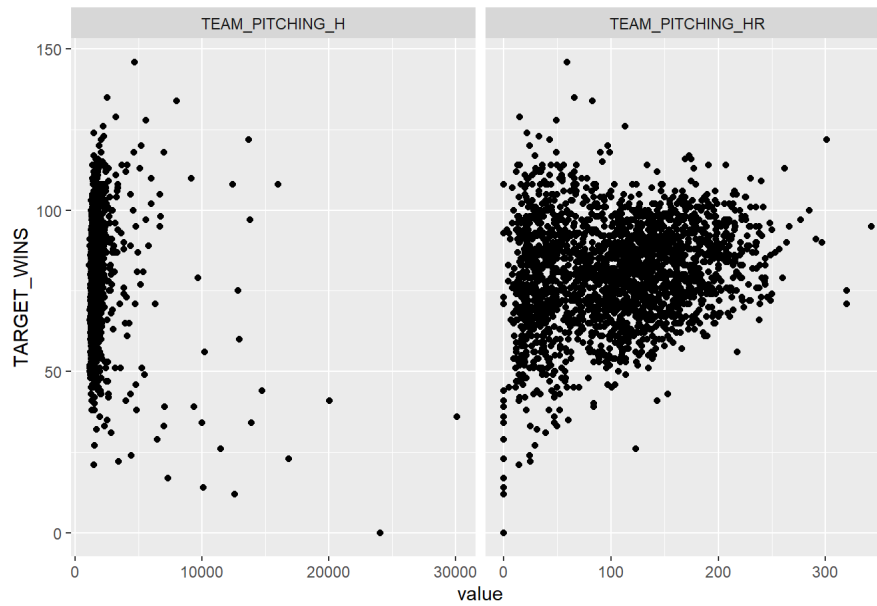


Page 3

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

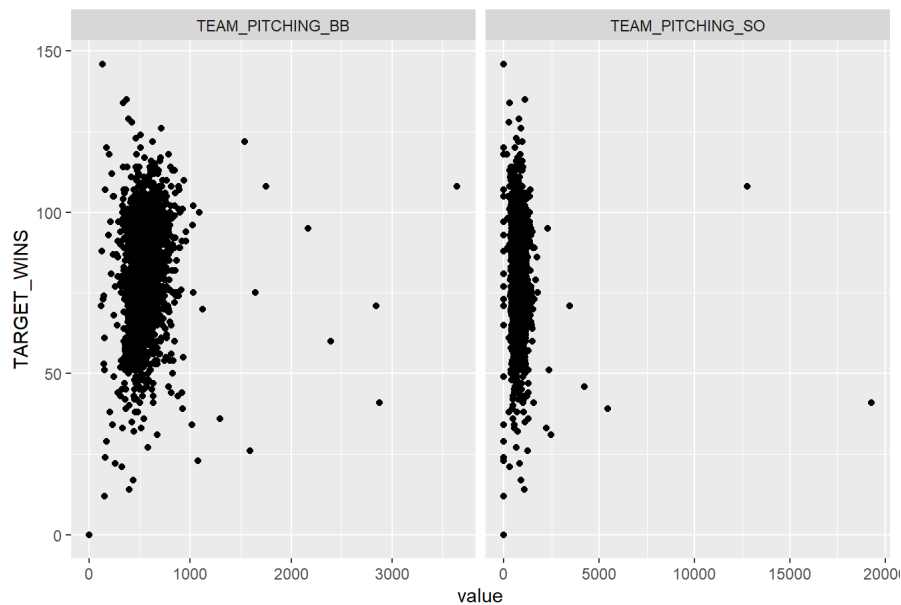


Page 4



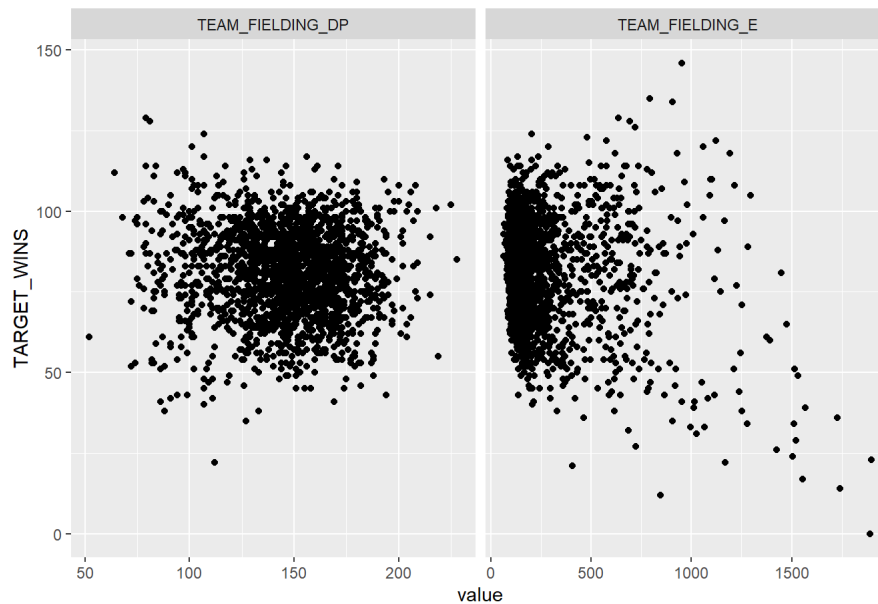
Page 5

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



Page 6

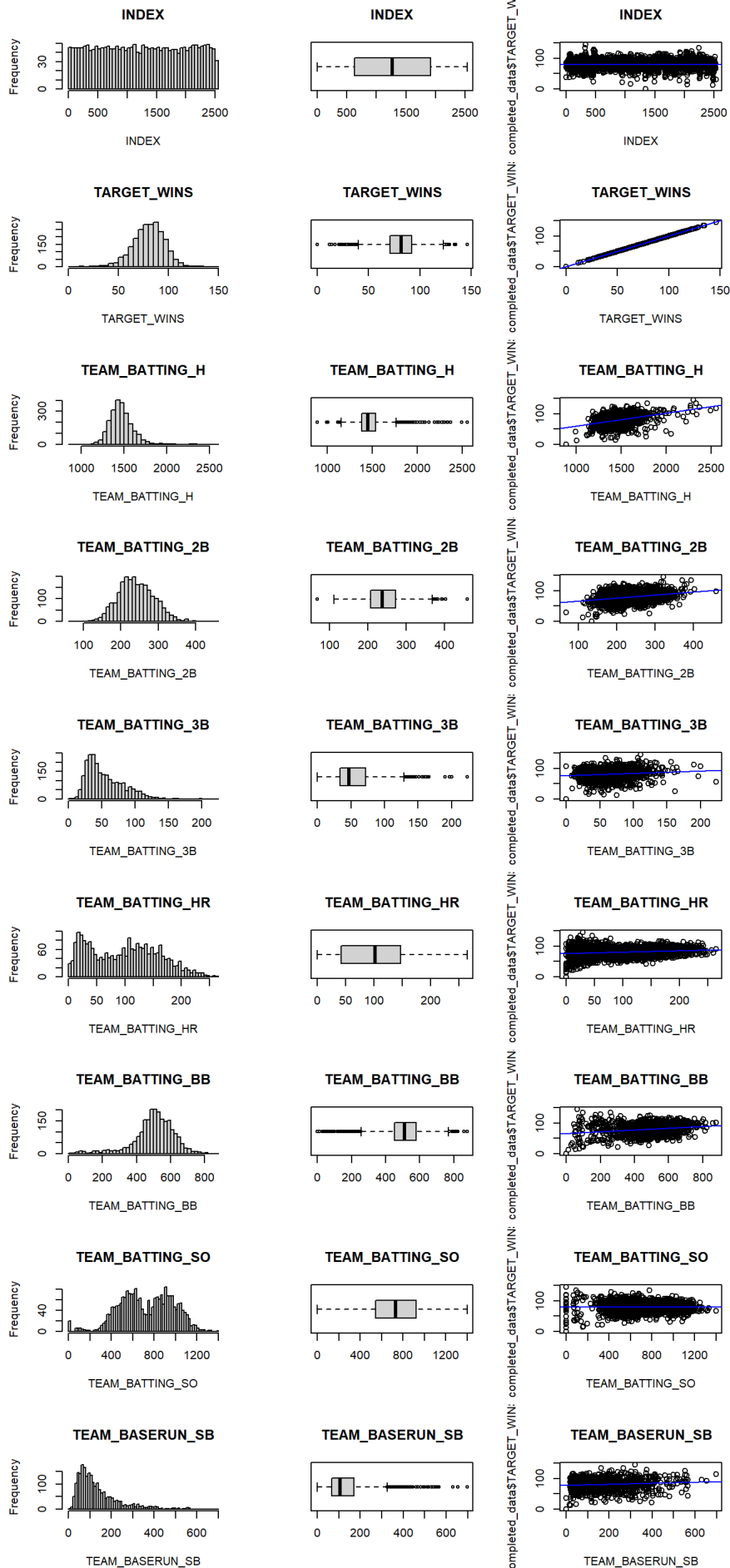
```
## Warning: Removed 286 rows containing missing values or values outside the scale range
## (`geom_point()`).
```

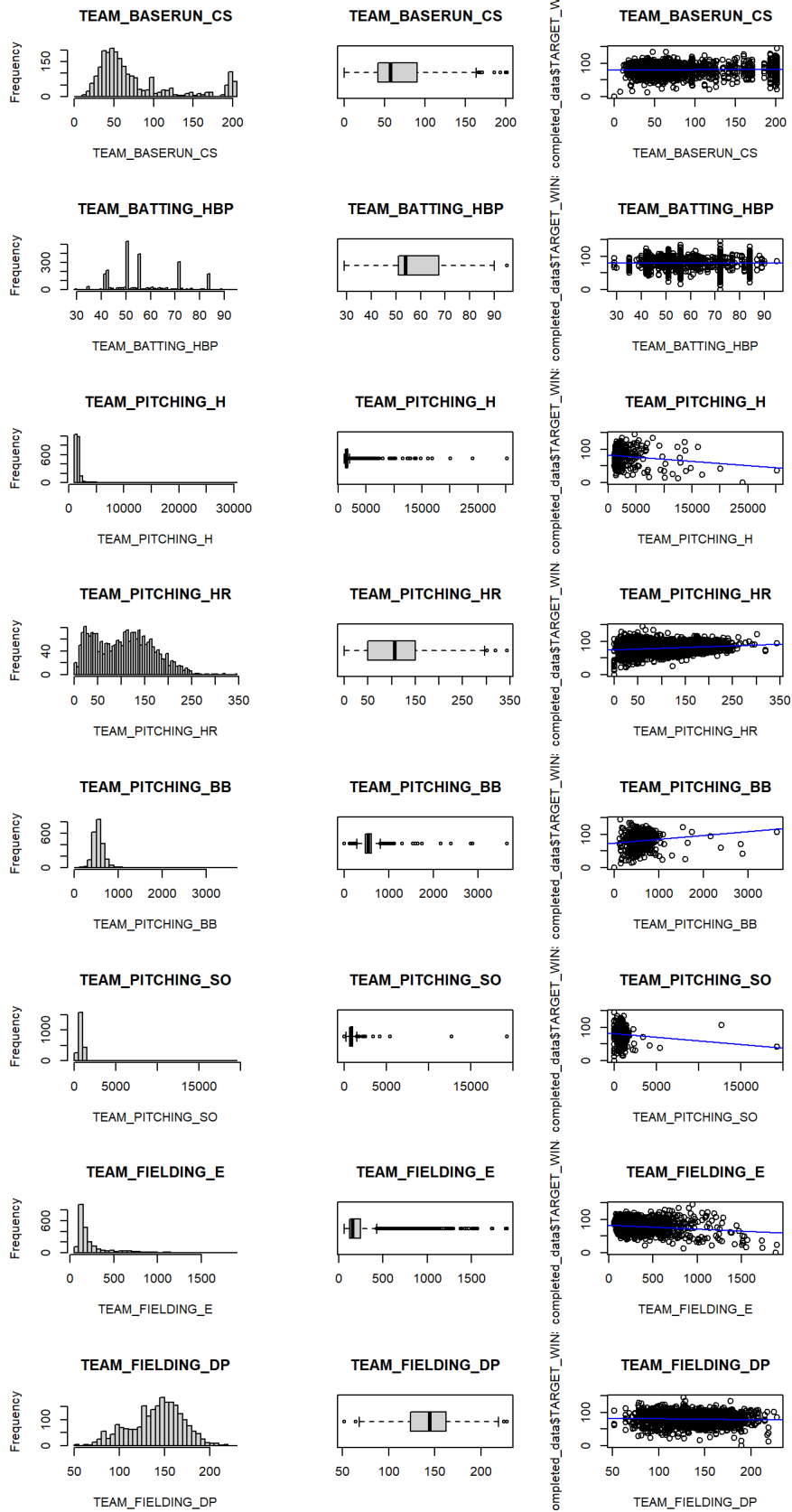


Page 7

```
# 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 $Q1 - 1.5 * IQR$ 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.

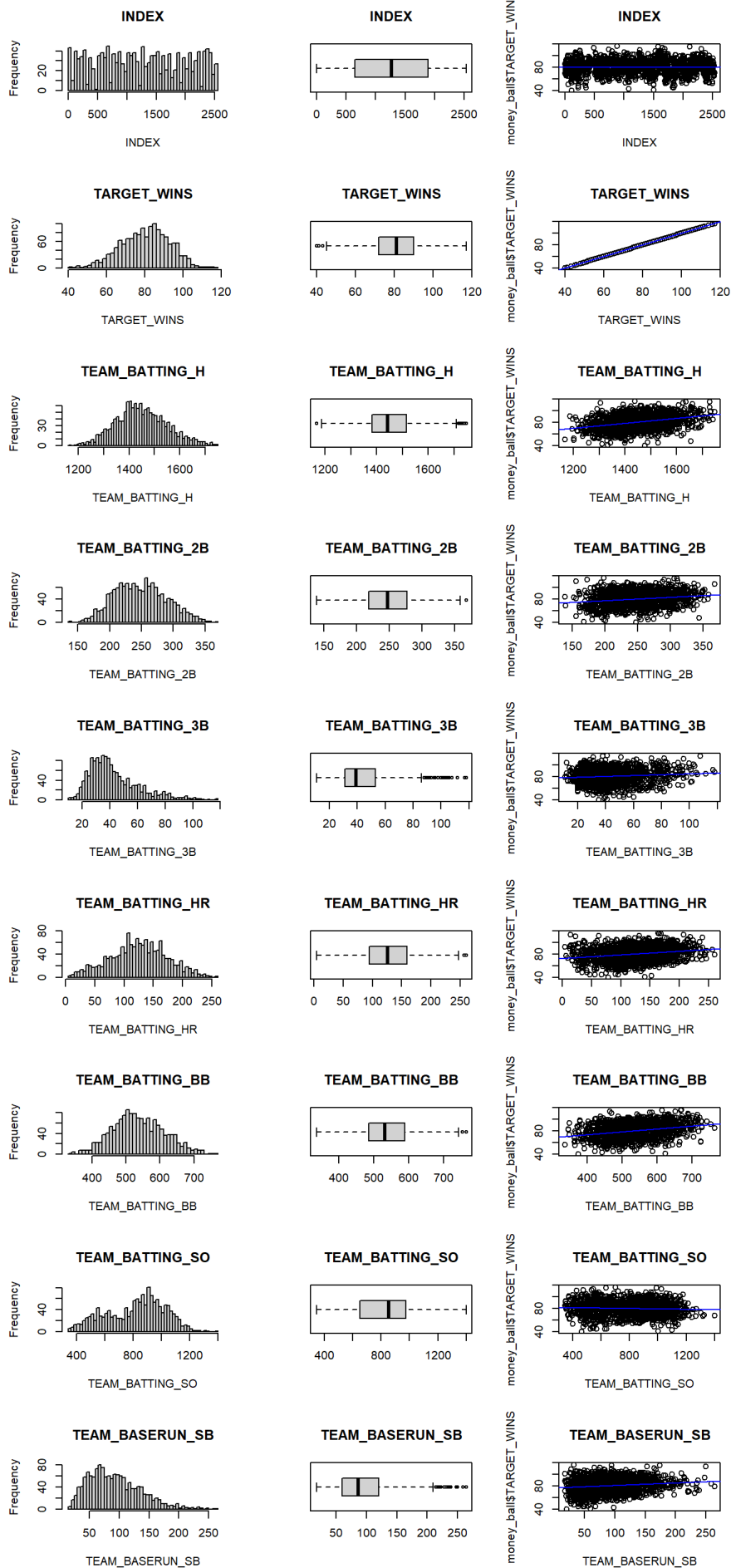
```
remove_outliers_df <- function(df) {
  # Loop through each numeric column in the data frame
  for (col in names(df)) {
    if (is.numeric(df[[col]])) {
      # Calculate Q1, Q3, and IQR for the column
      Q1 <- quantile(df[[col]], 0.25, na.rm = TRUE)
      Q3 <- quantile(df[[col]], 0.75, na.rm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

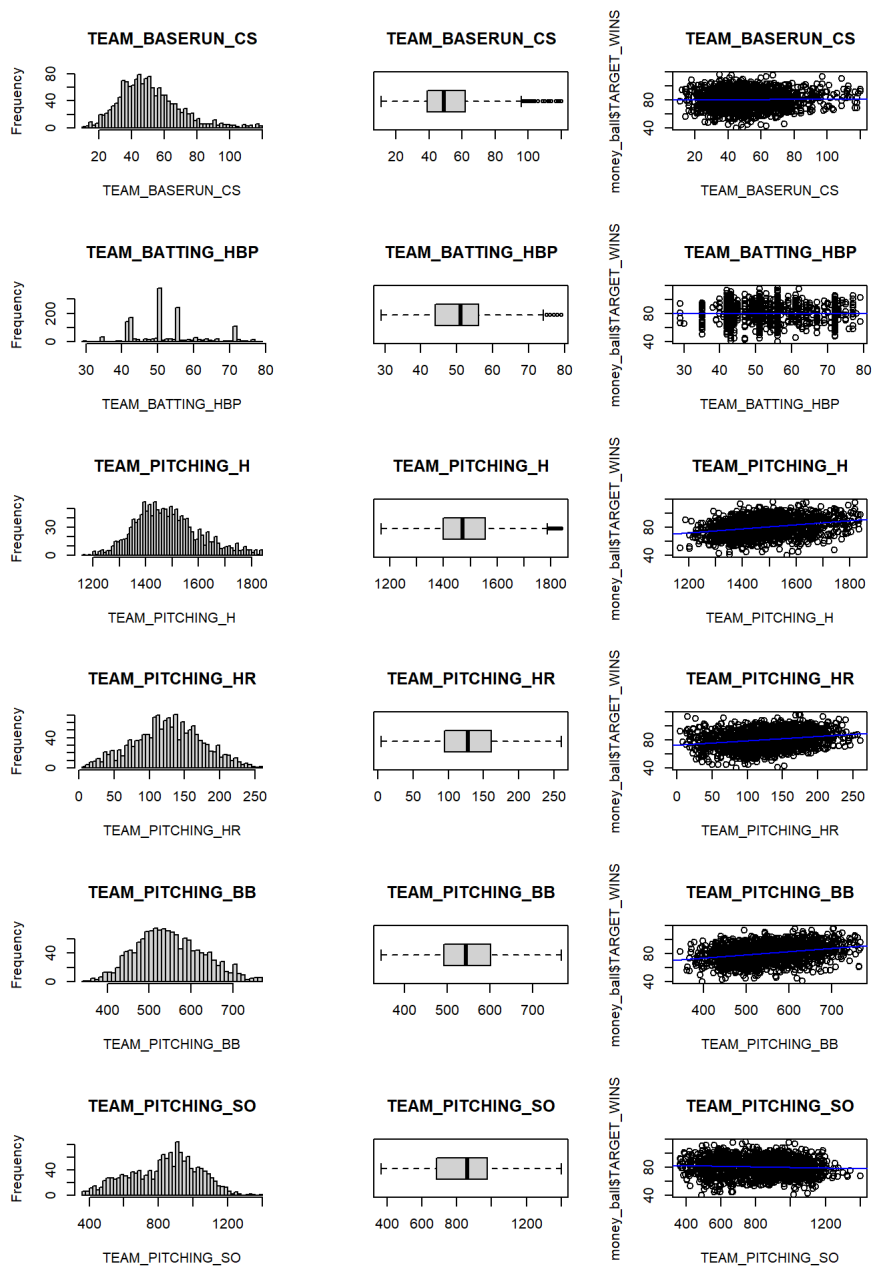
      # Set lower and upper bounds
      lower_bound <- Q1 - 1.5 * IQR_value
      upper_bound <- Q3 + 1.5 * IQR_value

      # 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)

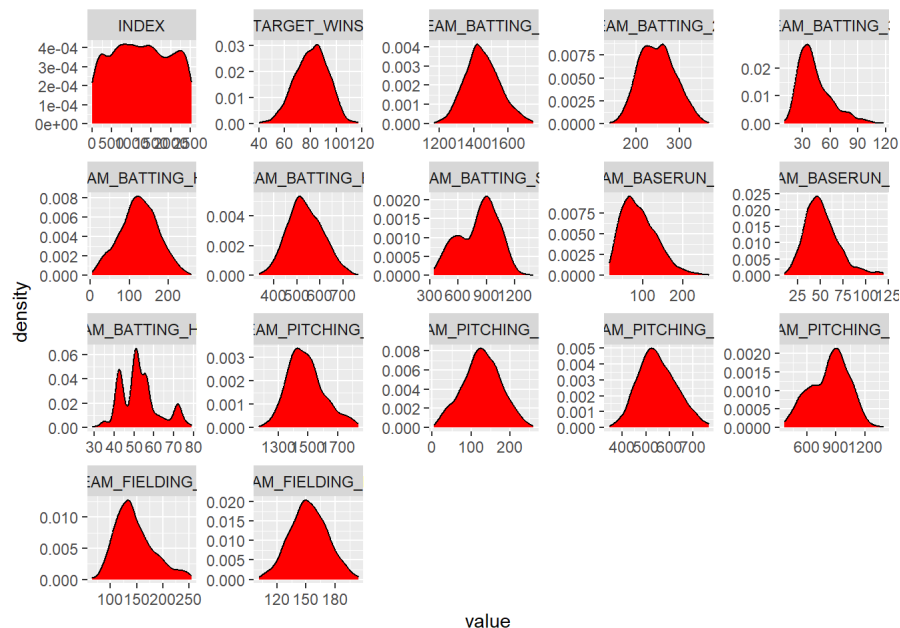
  plot(money_ball[,i], money_ball$TARGET_WINS, main = names(money_ball[i]), xlab=names(money_ball[i]))
  abline(lm(money_ball$TARGET_WINS ~ money_ball[,i], data = money_ball), col = "blue")
}
```





```
# 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	0.008422631	0.7478709021
## cor1	TEAM_BATTING_H	0.350606673	0.0000000000
## cor2	TEAM_BATTING_2B	0.191822367	0.0000000000
## cor3	TEAM_BATTING_3B	0.111664390	0.0000191181
## cor4	TEAM_BATTING_HR	0.236697472	0.0000000000
## cor5	TEAM_BATTING_BB	0.299035092	0.0000000000
## cor6	TEAM_BATTING_SO	-0.055416159	0.0342992936
## cor7	TEAM_BASERUN_SB	0.156024960	0.0000000021
## cor8	TEAM_BASERUN_CS	0.034044870	0.1937143475
## cor9	TEAM_BATTING_HBP	0.010491051	0.6888672519
## cor10	TEAM_PITCHING_H	0.284528562	0.0000000000
## cor11	TEAM_PITCHING_HR	0.240569326	0.0000000000
## cor12	TEAM_PITCHING_BB	0.283175704	0.0000000000
## cor13	TEAM_PITCHING_SO	-0.060685106	0.0204422392
## cor14	TEAM_FIELDING_E	-0.206165061	0.0000000000
## cor15	TEAM_FIELDING_DP	-0.084323398	0.0012647021

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
## -29.7254  -6.7995   0.8316   7.9832  27.1711
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -14.348909   8.138684  -1.763   0.0787 .
## TEAM_BATTING_H    0.071071   0.006925  10.262 <2e-16 ***
## TEAM_BATTING_2B  -0.032190   0.017874  -1.801   0.0725 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.89 on 379 degrees of freedom
## (1077 observations deleted due to missingness)
## Multiple R-squared:  0.2666, Adjusted R-squared:  0.2627
## F-statistic: 68.89 on 2 and 379 DF,  p-value: < 2.2e-16
```

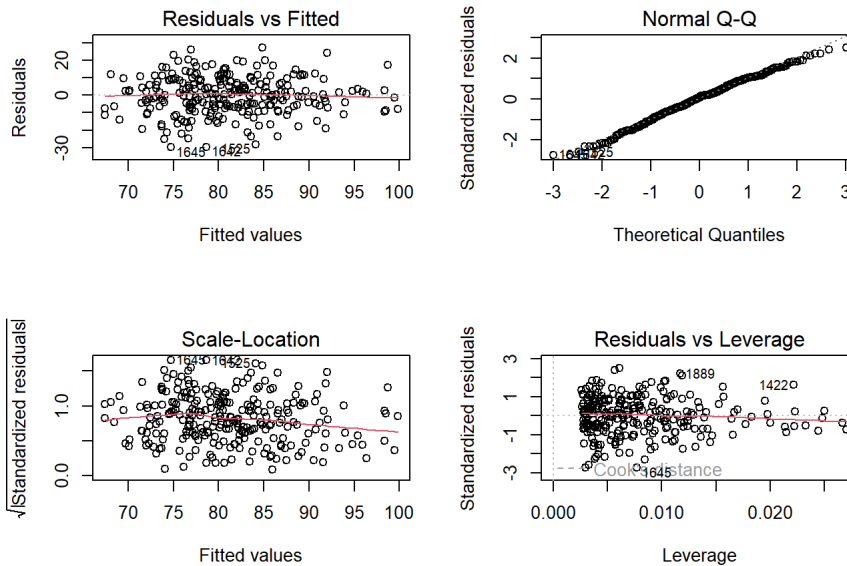
```
# Model 1 evaluation
mse <- mean(model$residuals^2)
r_squared <- summary(model)$r.squared
f_stat <- summary(model)$fstatistic[1]
print(paste("MSE:", mse, "R-squared:", r_squared, "F-statistic:", f_stat))
```

```
## [1] "MSE: 117.749712981721 R-squared: 0.266614543985386 F-statistic: 68.8907254307806"
```

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

```
##      1      2      3      4      5      6
## 66.10329 67.56775 78.90397 85.08222 81.81371 79.75645
```

```
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
## -39.985  -7.713   0.464   7.847  36.254
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  80.8163    0.2893  279.393 < 2e-16 ***
## PC1           0.2117    0.1298   1.631  0.10308
## PC2           2.7194    0.1653  16.447 < 2e-16 ***
## PC3           0.6993    0.2126   3.289  0.00103 **
## PC4          -2.3335    0.2232 -10.454 < 2e-16 ***
## PC5          -0.4105    0.2922  -1.405  0.16030
## PC6          -1.0037    0.3047  -3.294  0.00101 **
## PC7           0.6186    0.3207   1.929  0.05394 .
## PC8          -2.7034    0.4049  -6.677 3.47e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11.05 on 1450 degrees of freedom
## Multiple R-squared:  0.2386, Adjusted R-squared:  0.2344
## F-statistic: 56.8 on 8 and 1450 DF, p-value: < 2.2e-16
```

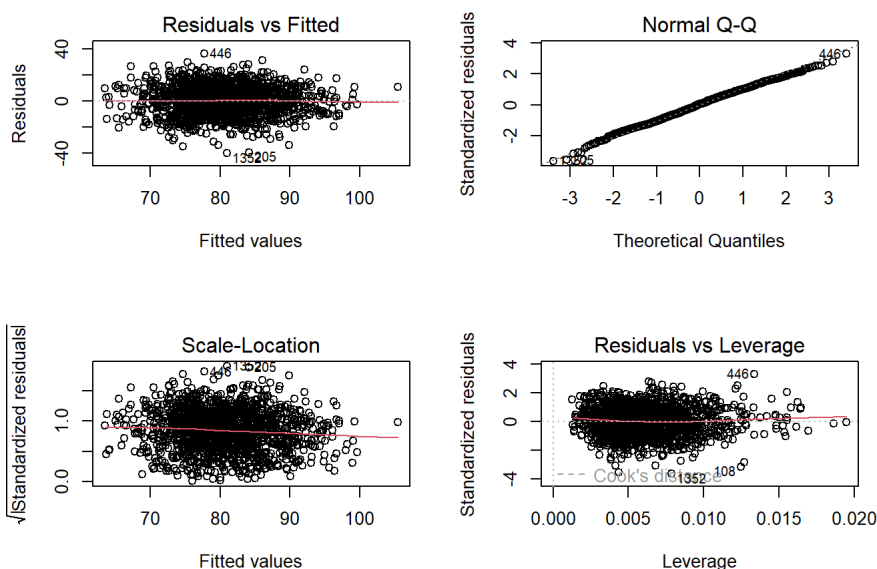
```
mse1 <- mean(model1$residuals^2)
r_squared1 <- summary(model1)$r.squared
f_stat1 <- summary(model1)$fstatistic[1]
print(paste("MSE:", mse1, "R-squared:", r_squared1, "F-statistic:", f_stat1))
```

```
## [1] "MSE: 121.320828721043 R-squared: 0.238603118226332 F-statistic: 56.7993069235844"
```

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

```
##           PC1      PC2      PC3      PC4      PC5      PC6      PC7
## [1,] 465.9440 1322.827 565.4319 -462.6089 27.695628 -2.0105539 -66.799173
## [2,] 357.2001 1400.106 455.2850 -484.2694 40.777104 -0.6514118 -60.430962
## [3,] 191.4738 1578.468 525.4015 -469.1025 16.372364 0.6083925 4.561732
## [4,] 276.2762 1746.431 738.1472 -514.8258 7.271371 17.1923851 50.298446
## [5,] -776.5872 2410.551 1247.2053 -195.2900 -93.431432 -70.8506852 238.264993
## [6,] -631.7089 2043.731 941.4270 -386.8419 -30.675285 -55.7001848 120.519500
##           PC8      PC9      PC10      PC11      PC12      PC13
## [1,] -89.53094 -961.2265 -103.57059 -426.5717 -1451.700 66.96620
## [2,] -109.81589 -819.1534 -94.00623 -423.5102 -1382.211 70.91550
## [3,] -146.01940 -711.2351 -69.02262 -473.1290 -1432.742 72.25565
## [4,] -146.37085 -770.3900 -72.46692 -529.0067 -1496.699 52.81823
## [5,] -557.14326 -856.0674 83.89420 -1028.9597 -2466.804 -1484.24030
## [6,] -418.43962 -646.4252 133.46033 -865.6571 -1827.902 -817.21623
##           PC14      PC15      PC16
## [1,] -78.10942 10.53647 -64.90391
## [2,] -73.98762 13.45572 -60.93782
## [3,] -76.31292 22.25110 -65.18016
## [4,] -85.04284 26.03535 -73.21157
## [5,] -264.78897 773.09017 -394.55557
## [6,] -188.96063 372.11987 -215.08136
```

```
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
##
## 1459 samples
## 16 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 1167, 1167, 1167, 1168, 1167
## Resampling results:
##
## RMSE      Rsquared   MAE
## 9.988737  0.3745199  8.025361
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

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



```
##      intercept      RMSE  Rsquared      MAE      RMSESD  RsquaredSD      MAESD
## 1          TRUE 9.988737 0.3745199 8.025361 0.1646179 0.02405222 0.1387634
```

```
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: 0.389740100146236 F-statistic: 57.5579790415471"
```

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

```
##           1           2           3           4           5           6
## 61.083590 67.678107 72.850636 81.580769 -7.278945  5.905271
```

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
```

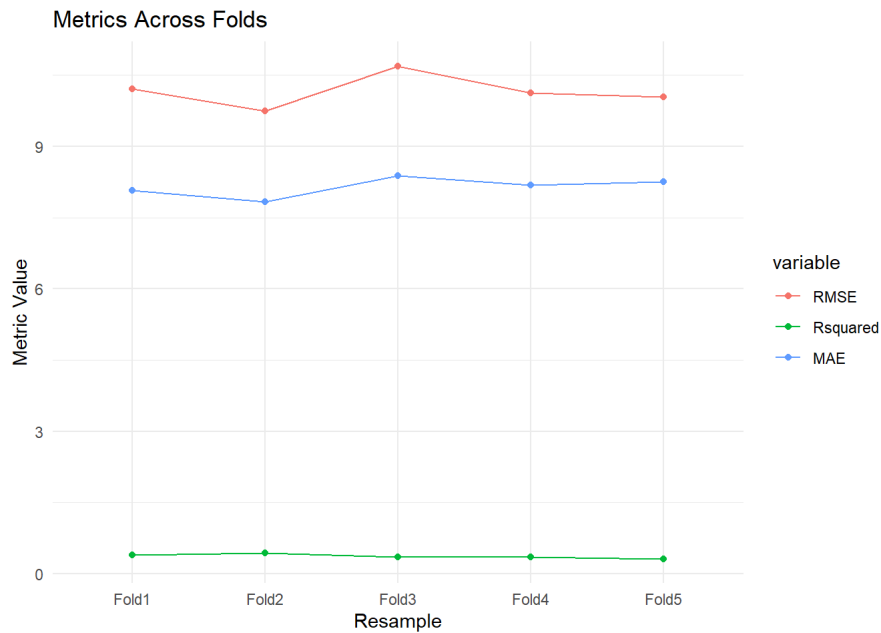
```
##      RMSE  Rsquared      MAE Resample
## 1  9.836874 0.3888343 7.990879   Fold1
## 2 10.168360 0.3802585 8.215199   Fold2
## 3 10.150719 0.3937132 8.119047   Fold3
## 4  9.960314 0.3332440 7.899124   Fold4
## 5  9.827419 0.3765494 7.902557   Fold5
```

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

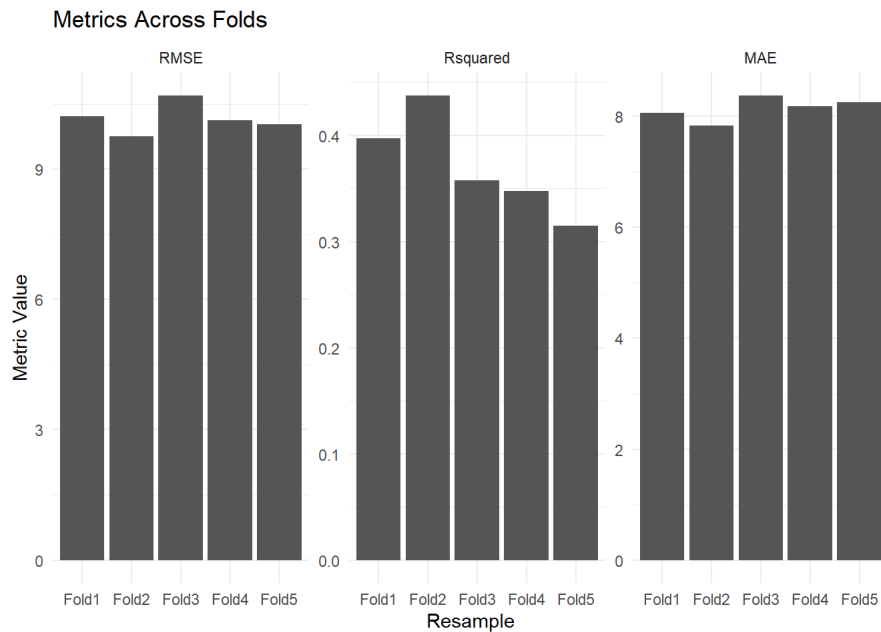
# Sample data
data <- data.frame(
  RMSE = c(10.211374, 9.748915, 10.693919, 10.120101, 10.037255),
  Rsquared = c(0.3975036, 0.4376530, 0.3577975, 0.3476137, 0.3151282),
  MAE = c(8.068065, 7.832611, 8.377911, 8.179323, 8.261450),
  Resample = c('Fold1', 'Fold2', 'Fold3', 'Fold4', 'Fold5')
)

# 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.