DATA 621 - Homework 1

Deepak Mongia & Soumya Ghosh

February 25, 2020

Libraries

```
library(kableExtra)
library(tidyverse)
library(ggplot2)
library(dplyr)
library(MASS)
library(corrplot)
library(RColorBrewer)
library(GGally)
library(ggResidpanel)
library(psych)
library(mice)
library(reshape2)
library(cowplot)
library(car)
library(caTools)
library(VIM)
library(broom)
```

Introduction

The dataset that we have been given for this assignment is a collection of historical baseball team performances. 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 original Training data set is comprised of 17 elements and 2276 total observations. Out of those 17 elements, INDEX is simply an index value used for sorting while TARGET_WINS represents the response variable we are to use within our regression models. The remaining 15 elements are all potential predictor variables for our linear models. A summary table for the data set is provided below.

Objective

Our objective is to build a multiple linear regression model on the training data to predict the number of wins for the team. We can only use the variables listed above (or variables that can be derived from the variables provided).

Data Load

Loaded Training and Evalutaion data sets into respective data frames.

```
train_df <- read.csv("https://raw.githubusercontent.com/soumya2g/CUNYDataMiningHomeWork/master/HomeWork_

eval_df <- read.csv("https://raw.githubusercontent.com/soumya2g/CUNYDataMiningHomeWork/master/HomeWork_
```

Training Data

Excluded the 'INDEX' attribute from training and evaluation(test) data frames.

```
train_df <- train_df[-c(1)]
```

Sample snapshot of training data frame -

```
head(train_df, 20) %>% kable() %>% kable_styling(bootstrap_options = c("striped", "hover", "condensed",
```

TARGET_WINS	TEAM_BATTING_H	TEAM_BATTING_2B	TEAM_BATTING_3B	TEAM_BATTING_H
39	1445	194	39	
70	1339	219	22	19
86	1377	232	35	13
70	1387	209	38	9
82	1297	186	27	10
75	1279	200	36	9
80	1244	179	54	12
85	1273	171	37	11
86	1391	197	40	11
76	1271	213	18	9
78	1305	179	27	8
68	1372	203	31	9
72	1332	196	41	5
76	1265	210	23	6
74	1380	233	40	13
87	1417	226	28	10
88	1563	242	43	10
66	1460	239	32	10
75	1390	197	24	14
93	1518	268	26	18

```
str(train_df)
```

```
## 'data.frame':    2276 obs. of  16 variables:
##  $ TARGET_WINS     : int  39 70 86 70 82 75 80 85 86 76 ...
##  $ TEAM_BATTING_H  : int  1445 1339 1377 1387 1297 1279 1244 1273 1391 1271 ...
##  $ TEAM_BATTING_2B : int  194 219 232 209 186 200 179 171 197 213 ...
##  $ TEAM_BATTING_3B : int  39 22 35 38 27 36 54 37 40 18 ...
##  $ TEAM_BATTING_HR : int  13 190 137 96 102 92 122 115 114 96 ...
##  $ TEAM_BATTING_BB : int  143 685 602 451 472 443 525 456 447 441 ...
##  $ TEAM_BATTING_SO : int  842 1075 917 922 920 973 1062 1027 922 827 ...
##  $ TEAM_BASERUN_SB : int  NA 37 46 43 49 107 80 40 69 72 ...
##  $ TEAM_BASERUN_CS : int  NA 28 27 30 39 59 54 36 27 34 ...
##  $ TEAM_BATTING_HBP: int  NA NA NA NA NA NA NA NA NA NA ...
##  $ TEAM_PITCHING_H : int  9364 1347 1377 1396 1297 1279 1244 1281 1391 1271 ...
##  $ TEAM_PITCHING_HR: int  84 191 137 97 102 92 122 116 114 96 ...
##  $ TEAM_PITCHING_BB: int  927 689 602 454 472 443 525 459 447 441 ...
```

```
##  $ TEAM_PITCHING_SO: int  5456 1082 917 928 920 973 1062 1033 922 827 ...
##  $ TEAM_FIELDING_E : int  1011 193 175 164 138 123 136 112 127 131 ...
##  $ TEAM_FIELDING_DP: int  NA 155 153 156 168 149 186 136 169 159 ...
```

PART I: Data Exploration

We wanted to start off data exploration process with high level descriptive statistical summary and missing/exception value analysis.

Descriptive Statistical Summary

Basic statistical summary of all features and dependant variable (TARGET_WINS).

```
stat_summary <- function(df){
  df %>%
    summary() %>%
    kable() %>%
    kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive")) %>%  scroll_box
}
stat_summary(train_df)
```

	TARGET_WINS	TEAM_BATTING_H	TEAM_BATTING_2B	TEAM_BATTING_3B	TEAM_BATTING_
	Min. : 0.00	Min. : 891	Min. : 69.0	Min. : 0.00	Min. : 0.00
	1st Qu.: 71.00	1st Qu.:1383	1st Qu.:208.0	1st Qu.: 34.00	1st Qu.: 42.00
	Median : 82.00	Median :1454	Median :238.0	Median : 47.00	Median :102.00
	Mean : 80.79	Mean :1469	Mean :241.2	Mean : 55.25	Mean : 99.61
	3rd Qu.: 92.00	3rd Qu.:1537	3rd Qu.:273.0	3rd Qu.: 72.00	3rd Qu.:147.00
	Max. :146.00	Max. :2554	Max. :458.0	Max. :223.00	Max. :264.00
	NA	NA	NA	NA	NA

We also used describe() function of 'psych' package to summarize additional statistical measurements like Standard Deviation, Skewness, Kurtois, Standard Error etc.

```
stat_desc <- function(df){
df %>%
    describe() %>%
    kable() %>%
    kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive")) %>%  scroll_box
}

stat_desc(train_df)
```

	vars	n	mean	sd	median	trimmed	mad	min	max
TARGET_WINS	1	2276	80.79086	15.75215	82.0	81.31229	14.8260	0	146
TEAM_BATTING_H	2	2276	1469.26977	144.59120	1454.0	1459.04116	114.1602	891	2554
TEAM_BATTING_2B	3	2276	241.24692	46.80141	238.0	240.39627	47.4432	69	458
TEAM_BATTING_3B	4	2276	55.25000	27.93856	47.0	52.17563	23.7216	0	223
TEAM_BATTING_HR	5	2276	99.61204	60.54687	102.0	97.38529	78.5778	0	264
TEAM_BATTING_BB	6	2276	501.55888	122.67086	512.0	512.18331	94.8864	0	878
TEAM_BATTING_SO	7	2174	735.60534	248.52642	750.0	742.31322	284.6592	0	1399
TEAM_BASERUN_SB	8	2145	124.76177	87.79117	101.0	110.81188	60.7866	0	697
TEAM_BASERUN_CS	9	1504	52.80386	22.95634	49.0	50.35963	17.7912	0	201
TEAM_BATTING_HBP	10	191	59.35602	12.96712	58.0	58.86275	11.8608	29	95
TEAM_PITCHING_H	11	2276	1779.21046	1406.84293	1518.0	1555.89517	174.9468	1137	30132
TEAM_PITCHING_HR	12	2276	105.69859	61.29875	107.0	103.15697	74.1300	0	343
TEAM_PITCHING_BB	13	2276	553.00791	166.35736	536.5	542.62459	98.5929	0	3645
TEAM_PITCHING_SO	14	2174	817.73045	553.08503	813.5	796.93391	257.2311	0	19278
TEAM_FIELDING_E	15	2276	246.48067	227.77097	159.0	193.43798	62.2692	65	1898
TEAM_FIELDING_DP	16	1990	146.38794	26.22639	149.0	147.57789	23.7216	52	228

Missing Value Analysis

Below are the features that has NA values. It is clear from the table below, TEAM_BATTING_HBP(>90%) and TEAM_BASERUN_CS (>33%) have significant NA Values -

```
## Counts of missing data per feature
train_na_df <- data.frame(apply(train_df, 2, function(x) length(which(is.na(x)))))
train_na_df1 <- data.frame(apply(train_df, 2,function(x) {sum(is.na(x)) / length(x) * 100}))

train_na_df <- cbind(Feature = rownames(train_na_df), train_na_df, train_na_df1)
colnames(train_na_df) <- c('Feature Name','No. of NA Recocrds','Percentage of NA Records')
rownames(train_na_df) <- NULL


train_na_df%>% filter(`No. of NA Recocrds` != 0) %>% arrange(desc(`No. of NA Recocrds`)) %>% kable() %>%
```

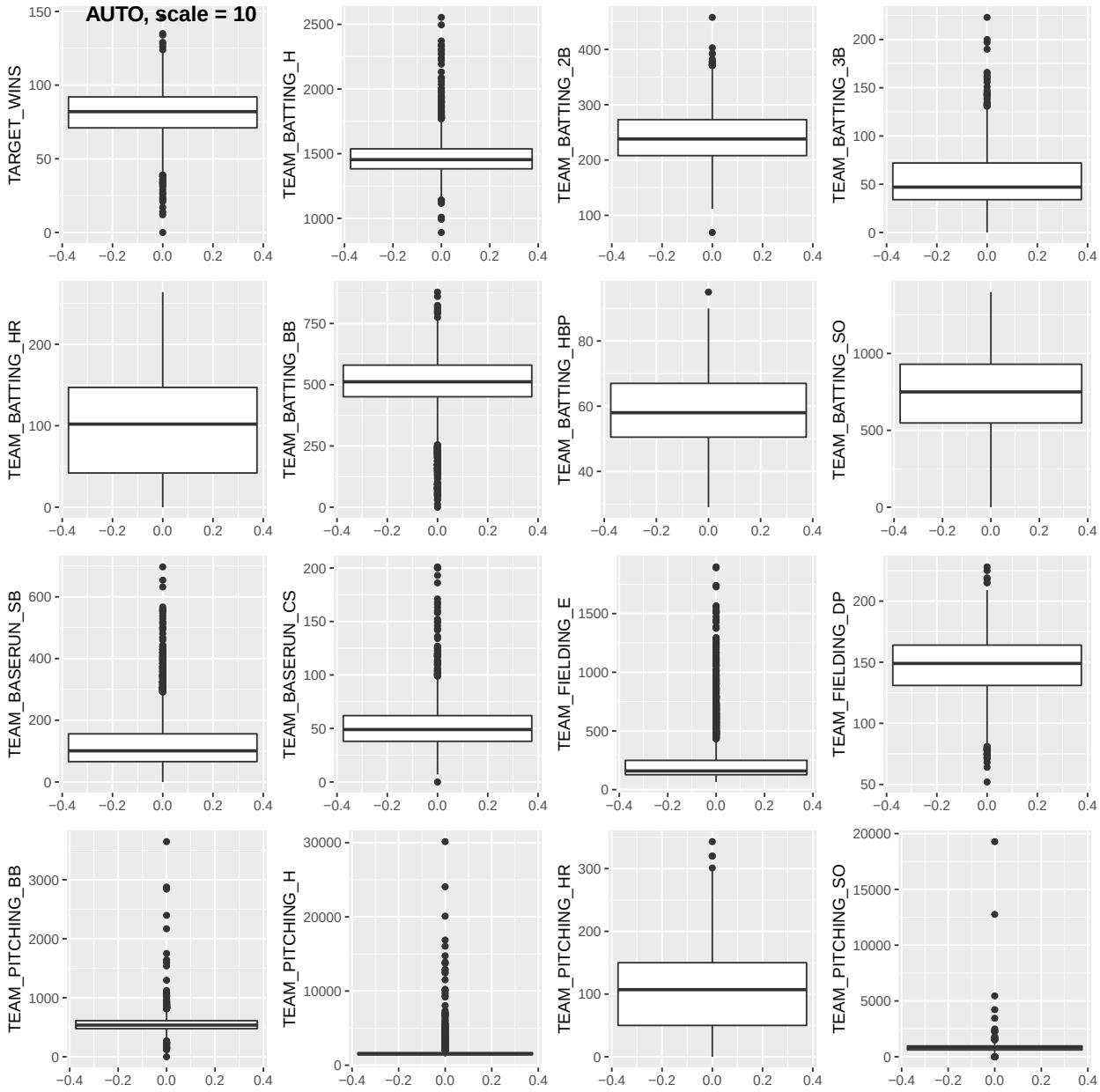
Feature Name	No. of NA Recocrds	Percentage of NA Records
TEAM_BATTING_HBP	2085	91.608084
TEAM_BASERUN_CS	772	33.919156
TEAM_FIELDING_DP	286	12.565905
TEAM_BASERUN_SB	131	5.755712
TEAM_BATTING_SO	102	4.481547
TEAM_PITCHING_SO	102	4.481547

Descriptive Statistical Plots

Box Plots

```
## Box plots:
gb1 <- ggplot(data = train_df, aes(y = TARGET_WINS)) + geom_boxplot()
gb2 <- ggplot(data = train_df, aes(y = TEAM_BATTING_H)) + geom_boxplot()
gb3 <- ggplot(data = train_df, aes(y = TEAM_BATTING_2B)) + geom_boxplot()
gb4 <- ggplot(data = train_df, aes(y = TEAM_BATTING_3B)) + geom_boxplot()
gb5 <- ggplot(data = train_df, aes(y = TEAM_BATTING_HR)) + geom_boxplot()
```

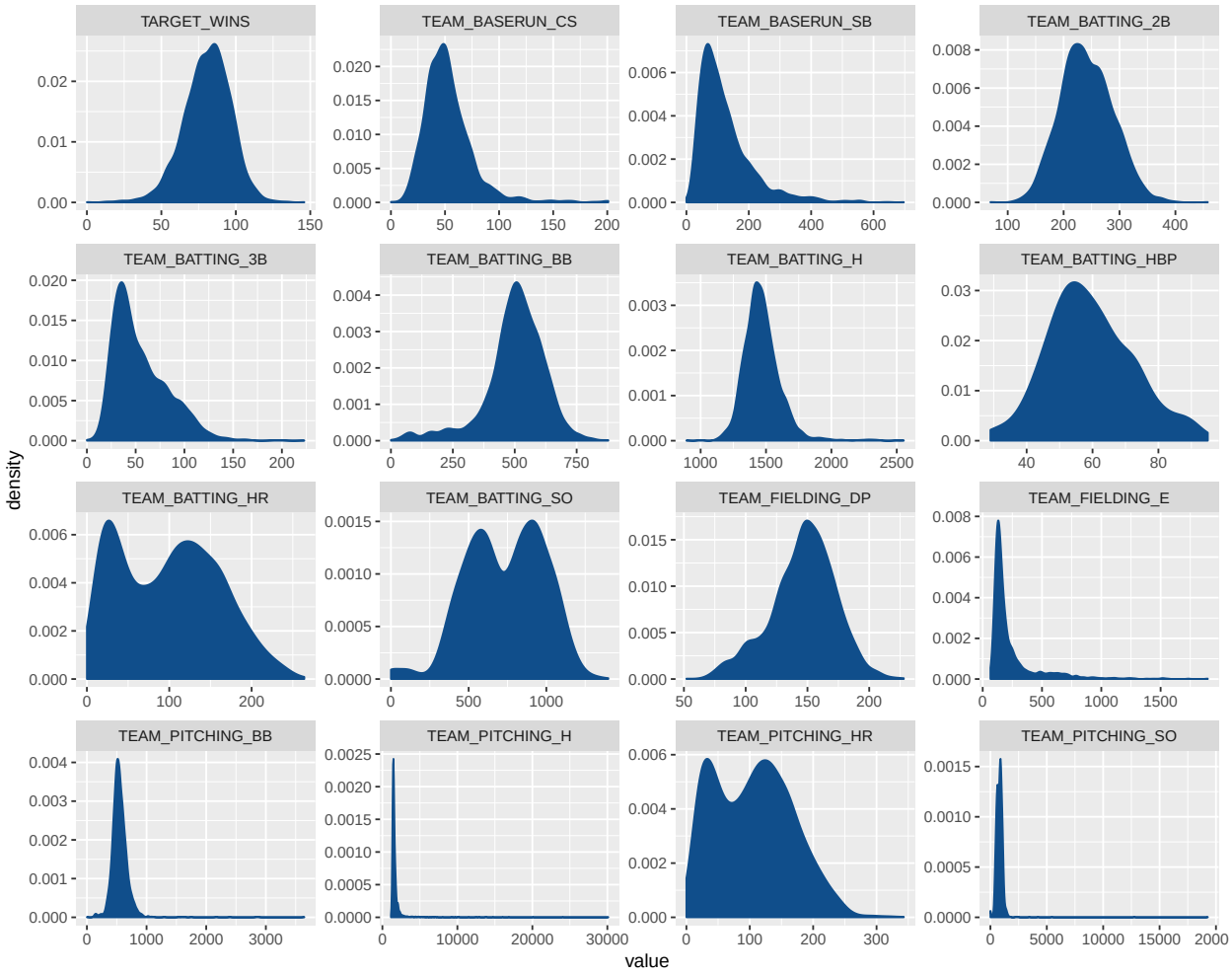
```r
gb6 <- ggplot(data = train_df, aes(y = TEAM_BATTING_BB)) + geom_boxplot()
gb7 <- ggplot(data = train_df, aes(y = TEAM_BATTING_HBP)) + geom_boxplot()
gb8 <- ggplot(data = train_df, aes(y = TEAM_BATTING_SO)) + geom_boxplot()
gb9 <- ggplot(data = train_df, aes(y = TEAM_BASERUN_SB)) + geom_boxplot()
gb10 <- ggplot(data = train_df, aes(y = TEAM_BASERUN_CS)) + geom_boxplot()
gb11 <- ggplot(data = train_df, aes(y = TEAM_FIELDING_E)) + geom_boxplot()
gb12 <- ggplot(data = train_df, aes(y = TEAM_FIELDING_DP)) + geom_boxplot()
gb13 <- ggplot(data = train_df, aes(y = TEAM_PITCHING_BB)) + geom_boxplot()
gb14 <- ggplot(data = train_df, aes(y = TEAM_PITCHING_H)) + geom_boxplot()
gb15 <- ggplot(data = train_df, aes(y = TEAM_PITCHING_HR)) + geom_boxplot()
gb16 <- ggplot(data = train_df, aes(y = TEAM_PITCHING_SO)) + geom_boxplot()

plot_grid(gb1, gb2, gb3, gb4, gb5, gb6, gb7, gb8, gb9, gb10,
          gb11, gb12, gb13, gb14, gb15, gb16, labels = "AUTO, scale = 10")
```



Density Plots

```
train_df %>%
  gather(variable, value, TARGET_WINS:TEAM_FIELDING_DP) %>%
  ggplot(., aes(value)) +
  geom_density(fill = "dodgerblue4", color="dodgerblue4") +
  facet_wrap(~variable, scales ="free", ncol = 4) +
  labs(x = element_blank(), y = element_blank())
```

```
## Warning: Removed 3478 rows containing non-finite values (stat_density).
```



Observations Summary

- The dataset has many outliers with some observations that are more extreme than the 1.5 * IQR of the box plot whiskers. We also see that the variance of some of the explanatory variables greatly exceeds the variance of the response "win" variable.
- Dependant variable **TARGET_WINS** is normally distributed which indicates that data set includes a balanced amount of good, bad and average team performances
- **TEAM_BATTING_HR,TEAM_BATTING_SO** and **TEAM_PITCHING_HR** features are showing bi-modal distribution
- Some of the features like **TEAM_BASERUN_CS, TEAM_BASERUN_SB and TEAM_FIELDING_E** are right skewed
- Observing the results of missing value analysis, we need to remove **TEAM_BATTING_HBP** due to large amount of missing values. We will need to come up with an imputation mechanism for the other features with missing values.

PART II: Data Preparation

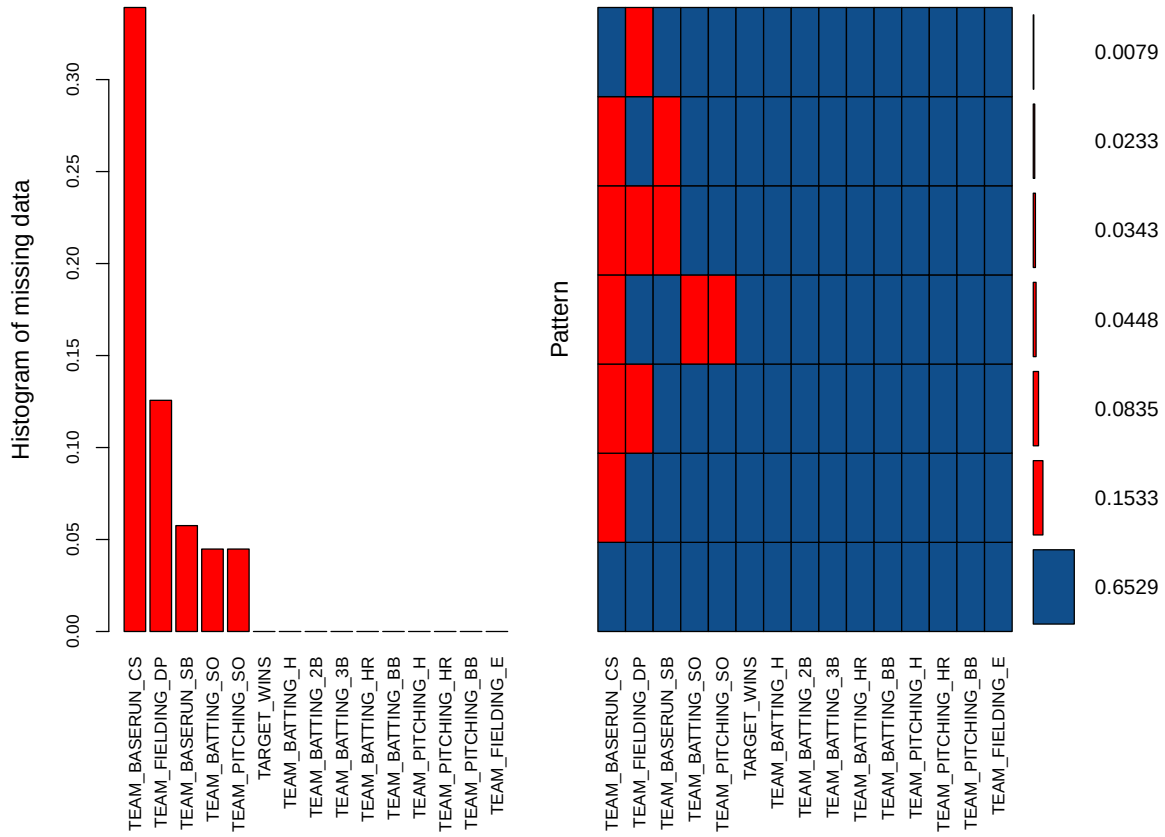
Remove the feature: TEAM_BATTING_HBP

The TEAM_BATTING_HBP variable contains 2085 missing values out of a total of 2277. Since it would be very difficult to accurately impute such a large proportion of any variable's missing values, so we choose to exclude the variable from our further analysis.

```
train_df <- train_df %>% dplyr::select(-c(TEAM_BATTING_HBP))
eval_df <-  eval_df %>% dplyr::select(-c(TEAM_BATTING_HBP))
```

Plotting the missing data pattern after removing the max. missing feature -

```
aggr_plot <- aggr(train_df, col=c('dodgerblue4','red'),
                  numbers=TRUE, sortVars=TRUE,oma = c(10,5,5,3),
                  labels=names(train_df), cex.axis=.8, gap=3, axes = TRUE, Prop = TRUE,
                  ylab=c("Histogram of missing data","Pattern"))
```




```
##
##  Variables sorted by number of missings:
##          Variable      Count
##   TEAM_BASERUN_CS 0.33919156
##  TEAM_FIELDING_DP 0.12565905
```

```
##   TEAM_BASERUN_SB 0.05755712
##   TEAM_BATTING_SO 0.04481547
##  TEAM_PITCHING_SO 0.04481547
##       TARGET_WINS 0.00000000
##    TEAM_BATTING_H 0.00000000
##   TEAM_BATTING_2B 0.00000000
##   TEAM_BATTING_3B 0.00000000
##   TEAM_BATTING_HR 0.00000000
##   TEAM_BATTING_BB 0.00000000
##   TEAM_PITCHING_H 0.00000000
##  TEAM_PITCHING_HR 0.00000000
##  TEAM_PITCHING_BB 0.00000000
##   TEAM_FIELDING_E 0.00000000
```

Imputation

Imputing the missing data for the other 4 features, which we are keeping for our analysis. We have used the 'Predictive Mean Matching'(pmm) method included in MICE package for imputation purposes.

```
impute_data <- function(df){

  df <- mice(data = df, m = 1, method = "pmm", maxit = 5, seed = 500)
  df <- mice::complete(df, 1)
}

train_df <- impute_data(train_df)
```

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

```
eval_df <- impute_data(eval_df)
```

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

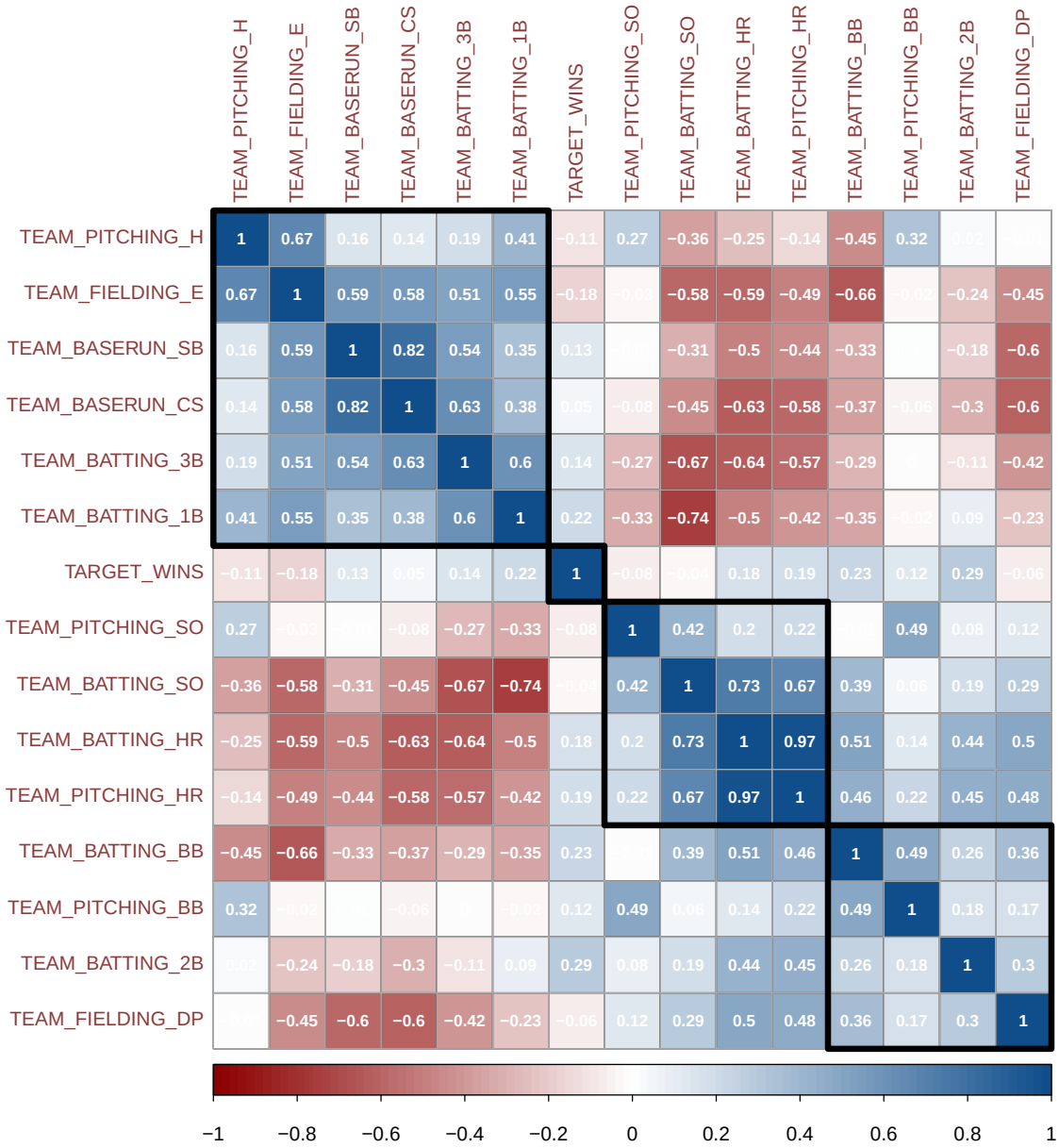
Feature Engineering

We created a new variable called TEAM_BATTING_1B which represents offensive single base hits. (created by subtracting out the TEAM_BATTING doubles, triples and home runs from the TEAM_BATTING_H variable). We believe that separating out singles from the other unique hit values will minimize collinearity. The TEAM_BATTING_H variable is then removed from the data set since it is simply a linear combination of its component variables.

```r
add_batting1b_feature <- function(df){
  df <-df %>%
    mutate(TEAM_BATTING_1B = TEAM_BATTING_H - TEAM_BATTING_2B - TEAM_BATTING_3B - TEAM_BATTING_HR)
}

train_df <- add_batting1b_feature(train_df)
eval_df <- add_batting1b_feature(eval_df)

train_df <- train_df %>% dplyr::select(-c(TEAM_BATTING_H))
eval_df <-  eval_df %>% dplyr::select(-c(TEAM_BATTING_H))
```

Correlation Plot

```r
corrMatrix <- round(cor(train_df),4)

corrMatrix %>% corrplot(., method = "color", outline = T, addgrid.col = "darkgray", order="hclust", addr
```



Based on the Correlation plot above, there is a high degree of collinearity amogst independent variables like **TEAM_BATTING_HR, TEAM_PITCHING_HR,TEAM_BASERUN_SB, TEAM_BASERUN_CS** etc.

Handling multicollinearity

TEAM_BASERUN_CS: This variable is strongly correlated (82%) with the TEAM_BASERUN_SB variable and is the 2nd largest source of NA's in our data set. Hence we decided to exclude the variable from our analysis.

TEAM_PITCHING_HR: This variable is highly (97%) correlated with TEAM_BATTING_HR. The fact that these two variables are nearly perfectly correlated indicates that one of them can legitimately be removed from the data set, and we chose TEAM_PITCHING_HR to be removed from our model for further analysis.

```
train_df <- train_df %>% dplyr::select(-c(TEAM_BASERUN_CS,TEAM_PITCHING_HR))
eval_df <- eval_df %>% dplyr::select(-c(TEAM_BASERUN_CS,TEAM_PITCHING_HR))
```

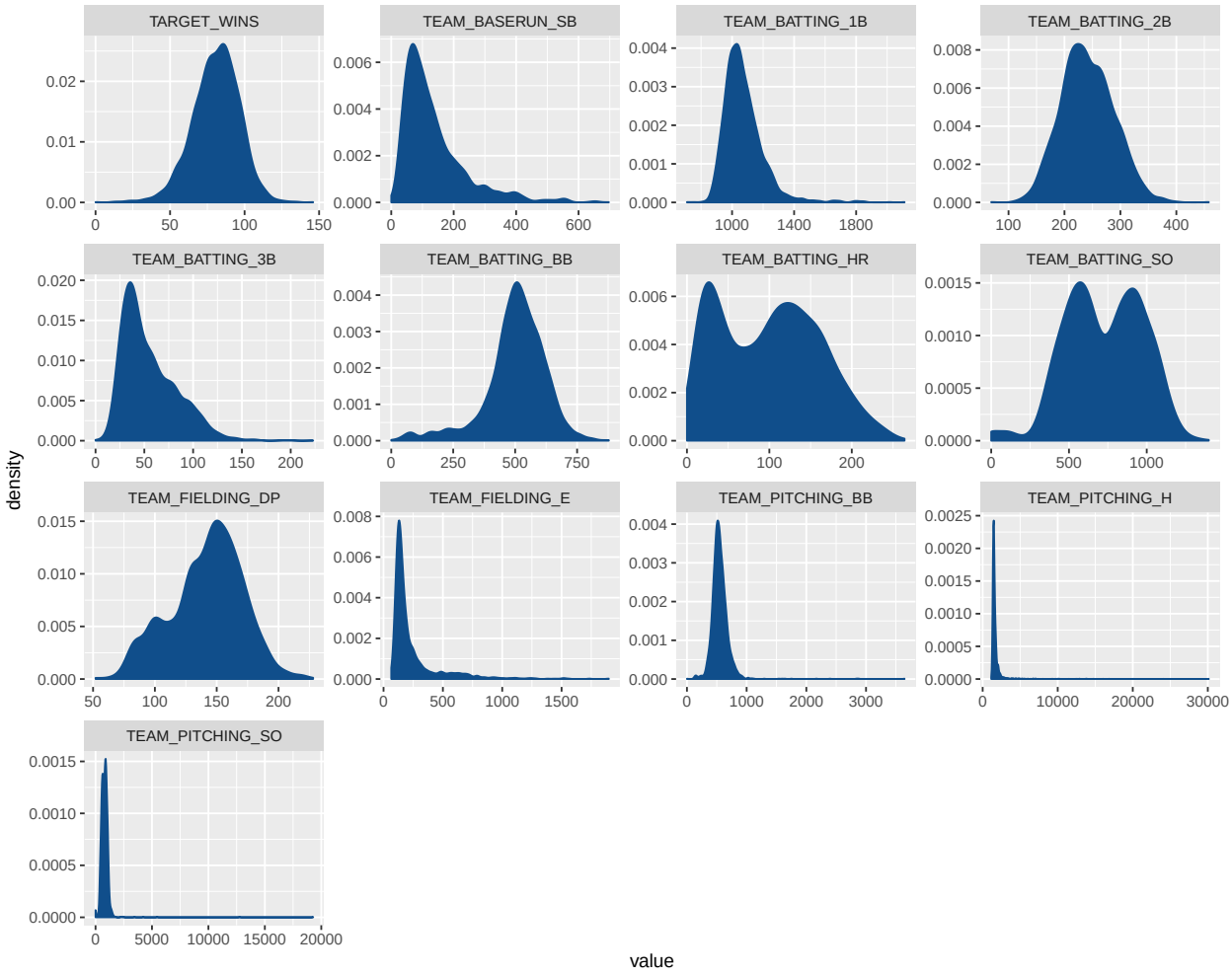
Data Preparation Results

After making feature removal and imputation we computed the statistical summary of the resultant training data and created feature density plots.

```
stat_desc(train_df)
```

	vars	n	mean	sd	median	trimmed	mad	min	max	r
TARGET_WINS	1	2276	80.79086	15.75215	82.0	81.31229	14.8260	0	146	
TEAM_BATTING_2B	2	2276	241.24692	46.80141	238.0	240.39627	47.4432	69	458	
TEAM_BATTING_3B	3	2276	55.25000	27.93856	47.0	52.17563	23.7216	0	223	
TEAM_BATTING_HR	4	2276	99.61204	60.54687	102.0	97.38529	78.5778	0	264	
TEAM_BATTING_BB	5	2276	501.55888	122.67086	512.0	512.18331	94.8864	0	878	
TEAM_BATTING_SO	6	2276	727.16037	247.06360	733.0	731.75906	284.6592	0	1399	
TEAM_BASERUN_SB	7	2276	135.98330	100.99464	106.0	118.92481	68.1996	0	697	
TEAM_PITCHING_H	8	2276	1779.21046	1406.84293	1518.0	1555.89517	174.9468	1137	30132	2
TEAM_PITCHING_BB	9	2276	553.00791	166.35736	536.5	542.62459	98.5929	0	3645	
TEAM_PITCHING_SO	10	2276	808.52021	543.41454	800.5	787.86169	258.7137	0	19278	1
TEAM_FIELDING_E	11	2276	246.48067	227.77097	159.0	193.43798	62.2692	65	1898	
TEAM_FIELDING_DP	12	2276	141.88664	29.21855	146.0	142.88968	28.1694	52	228	
TEAM_BATTING_1B	13	2276	1073.16081	128.92464	1050.0	1059.11196	99.3342	709	2112	

```
train_df %>%
  gather(variable, value, TARGET_WINS:TEAM_BATTING_1B) %>%
  ggplot(., aes(value)) +
  geom_density(fill = "dodgerblue4", color="dodgerblue4") +
  facet_wrap(~variable, scales ="free", ncol = 4) +
  labs(x = element_blank(), y = element_blank())
```



PART III: Building Models

Test train approach

We have divided the traning dataset into training and test sets using a 80/20 split. We will build our models on the training set and evaluate it on the test set.

```
set.seed(123)
split <- sample.split(train_df$TARGET_WINS, SplitRatio = 0.8)
training_set <- subset(train_df, split == TRUE)
test_set <- subset(train_df, split == FALSE)
```

We will build four different models to see which one yields the best performance.

Model 1:

```
g1 <- lm(TARGET_WINS ~ TEAM_BATTING_1B + TEAM_BATTING_2B + TEAM_BATTING_3B +
           TEAM_BATTING_HR + TEAM_BATTING_BB + TEAM_BATTING_SO +
           TEAM_BASERUN_SB +  TEAM_PITCHING_H +
            TEAM_PITCHING_BB + TEAM_PITCHING_SO +
```

```
          TEAM_FIELDING_E + TEAM_FIELDING_DP, data = training_set)

summary(g1)
```

```
##
## Call:
## lm(formula = TARGET_WINS ~ TEAM_BATTING_1B + TEAM_BATTING_2B +
##     TEAM_BATTING_3B + TEAM_BATTING_HR + TEAM_BATTING_BB + TEAM_BATTING_SO +
##     TEAM_BASERUN_SB + TEAM_PITCHING_H + TEAM_PITCHING_BB + TEAM_PITCHING_SO +
##     TEAM_FIELDING_E + TEAM_FIELDING_DP, data = training_set)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -46.143  -8.443   0.218   8.144  50.190
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)      34.5795058  5.4745350   6.316 3.36e-10 ***
## TEAM_BATTING_1B   0.0426304  0.0038183  11.165  < 2e-16 ***
## TEAM_BATTING_2B   0.0264099  0.0077580   3.404 0.000678 ***
## TEAM_BATTING_3B   0.0826533  0.0169375   4.880 1.15e-06 ***
## TEAM_BATTING_HR   0.1319750  0.0093123  14.172  < 2e-16 ***
## TEAM_BATTING_BB   0.0301181  0.0064215   4.690 2.93e-06 ***
## TEAM_BATTING_SO  -0.0222146  0.0036738  -6.047 1.79e-09 ***
## TEAM_BASERUN_SB   0.0569348  0.0046146  12.338  < 2e-16 ***
## TEAM_PITCHING_H   0.0019434  0.0004166   4.665 3.31e-06 ***
## TEAM_PITCHING_BB -0.0173696  0.0047283  -3.674 0.000246 ***
## TEAM_PITCHING_SO  0.0060394  0.0022425   2.693 0.007144 **
## TEAM_FIELDING_E  -0.0427808  0.0030067 -14.228  < 2e-16 ***
## TEAM_FIELDING_DP -0.1309038  0.0139311  -9.397  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12.46 on 1813 degrees of freedom
## Multiple R-squared:  0.3996, Adjusted R-squared:  0.3956
## F-statistic: 100.5 on 12 and 1813 DF,  p-value: < 2.2e-16
```

VIF Results

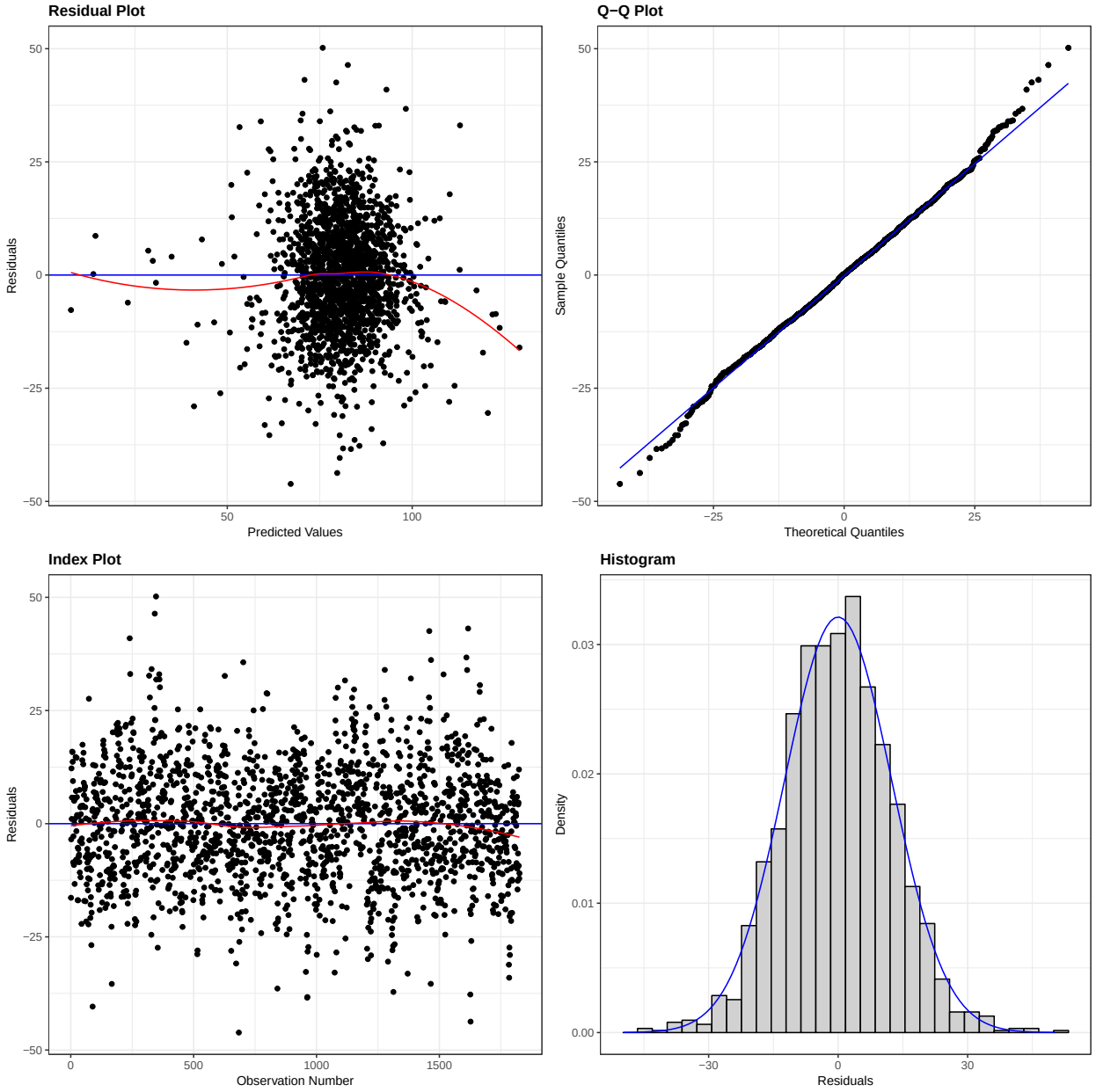
Multicollinearity can assessed by computing a score called the Variance Inflation Factor (or VIF), which measures how much the variance of a regression coefficient is inflated due to multicollinearity in the model. Features with VIF scores higher than 5 or 10 are still showing relatively higher degree of Collinearity. But we decided against removing any more features at this point.

```
vif(g1)
```

```
##  TEAM_BATTING_1B  TEAM_BATTING_2B  TEAM_BATTING_3B  TEAM_BATTING_HR
##         2.917513         1.546951         2.655454         3.793152
##  TEAM_BATTING_BB  TEAM_BATTING_SO  TEAM_BASERUN_SB  TEAM_PITCHING_H
##         7.508851         9.663351         2.643337         3.945886
## TEAM_PITCHING_BB TEAM_PITCHING_SO  TEAM_FIELDING_E TEAM_FIELDING_DP
##         5.230981         5.068137         5.649857         1.928087
```

Model Diagnostic Plots

```
resid_panel(g1, plots='default', smoother = TRUE)
```



RMSE Calculation

```
rmse_calc <- function(actual, predicted) {
  rmse_val <- sqrt(sum((actual - predicted)^2) / length(actual))

  return(rmse_val)
}

### RMSE of first model - training dataset
rmse_calc(training_set$TARGET_WINS, predict(g1, newdata = training_set))
```

```
## [1] 12.41448
```

```
### RMSE of first model - test dataset
rmse_calc(test_set$TARGET_WINS, predict(g1, newdata = test_set))
```

```
## [1] 12.72149
```

Model Summary

We have used the glance() methid from broom package to gather the model summary statistics in one data frame and appended our RMSE calculations so that we can compare these statistics in the model selection section.

```
model_1 <- as.data.frame(glance(g1))
model_1$Train_RMSE <- rmse_calc(training_set$TARGET_WINS, predict(g1, newdata = training_set))
model_1$Test_RMSE <- rmse_calc(test_set$TARGET_WINS, predict(g1, newdata = test_set))
```

Model 2:

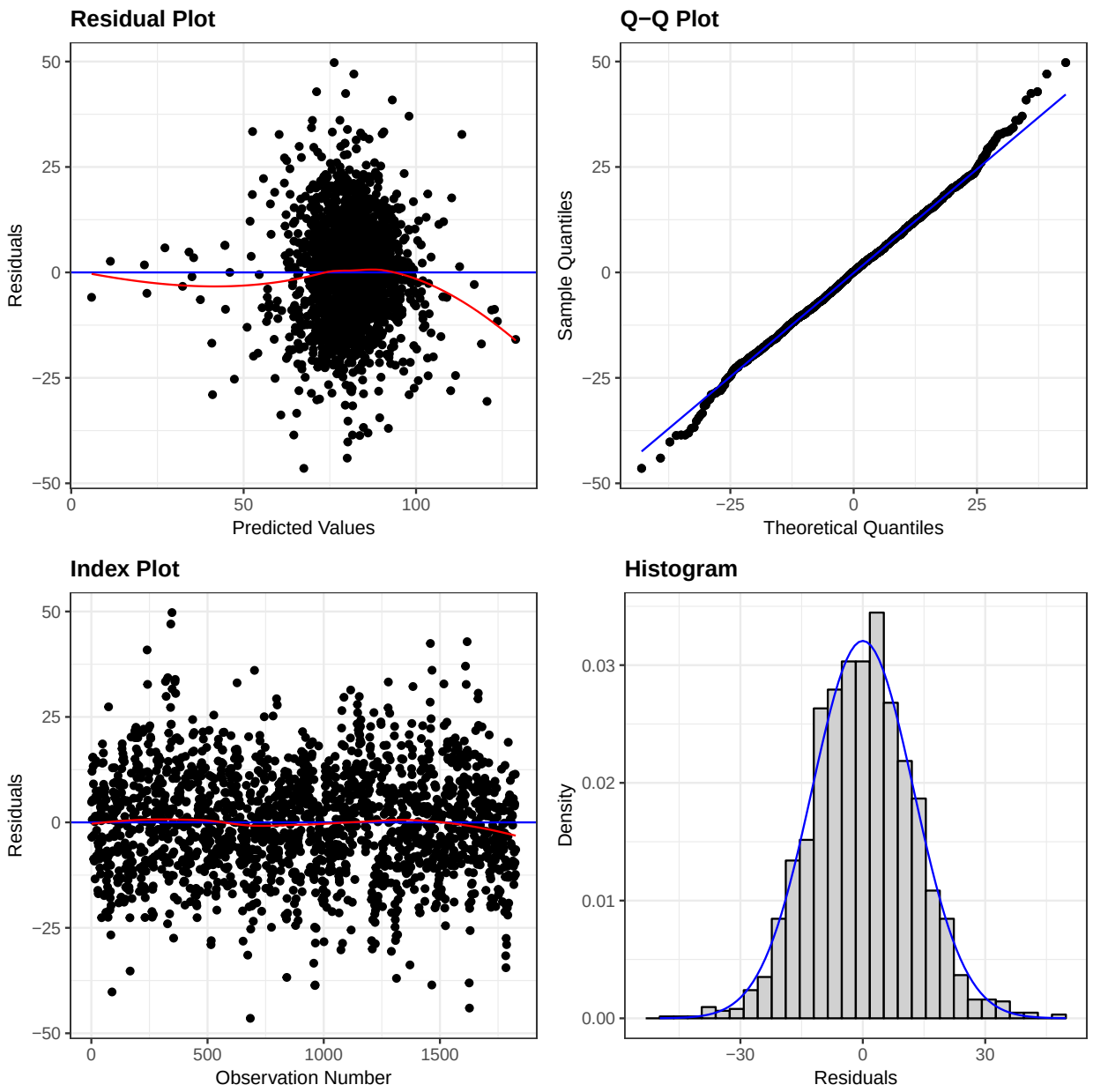
```
# Removing : TEAM_PITCHING_SO - based on p-values

g2 <- lm(TARGET_WINS ~TEAM_BATTING_1B + TEAM_BATTING_2B + TEAM_BATTING_3B +
           TEAM_BATTING_HR + TEAM_BATTING_BB + TEAM_BATTING_SO +
           TEAM_BASERUN_SB + TEAM_PITCHING_H + TEAM_PITCHING_BB +
           TEAM_FIELDING_E + TEAM_FIELDING_DP, data = training_set)


summary(g2)
```

```
##
## Call:
## lm(formula = TARGET_WINS ~ TEAM_BATTING_1B + TEAM_BATTING_2B +
##     TEAM_BATTING_3B + TEAM_BATTING_HR + TEAM_BATTING_BB + TEAM_BATTING_SO +
##     TEAM_BASERUN_SB + TEAM_PITCHING_H + TEAM_PITCHING_BB + TEAM_FIELDING_E +
##     TEAM_FIELDING_DP, data = training_set)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -46.458  -8.386   0.166   8.132  49.739
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      34.529218   5.483930   6.296 3.81e-10 ***
## TEAM_BATTING_1B   0.042182   0.003821  11.039  < 2e-16 ***
## TEAM_BATTING_2B   0.027113   0.007767   3.491 0.000493 ***
## TEAM_BATTING_3B   0.081998   0.016965   4.833 1.46e-06 ***
## TEAM_BATTING_HR   0.128944   0.009260  13.925  < 2e-16 ***
## TEAM_BATTING_BB   0.020983   0.005462   3.842 0.000126 ***
## TEAM_BATTING_SO  -0.014900   0.002478  -6.013 2.20e-09 ***
## TEAM_BASERUN_SB   0.057051   0.004622  12.342  < 2e-16 ***
## TEAM_PITCHING_H   0.001734   0.000410   4.229 2.46e-05 ***
## TEAM_PITCHING_BB -0.009780   0.003803  -2.571 0.010205 *
## TEAM_FIELDING_E  -0.041431   0.002970 -13.951  < 2e-16 ***
## TEAM_FIELDING_DP -0.126497   0.013858  -9.128  < 2e-16 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12.48 on 1814 degrees of freedom
## Multiple R-squared:  0.3972, Adjusted R-squared:  0.3935
## F-statistic: 108.7 on 11 and 1814 DF,  p-value: < 2.2e-16
```

Model Diagnostic Plots

```
resid_panel(g2, plots='default', smoother = TRUE)
```



RMSE Calculation

```
### RMSE of second model - training dataset
rmse_calc(training_set$TARGET_WINS, predict(g2, newdata = training_set))
```

```
## [1] 12.43929
```

```
### RMSE of second model - test dataset
rmse_calc(test_set$TARGET_WINS, predict(g2, newdata = test_set))
```

```
## [1] 12.98812
```

Model Summary

```
model_2 <- as.data.frame(glance(g2))
model_2$Train_RMSE <- rmse_calc(training_set$TARGET_WINS, predict(g2, newdata = training_set))
model_2$Test_RMSE <- rmse_calc(test_set$TARGET_WINS, predict(g2, newdata = test_set))
```

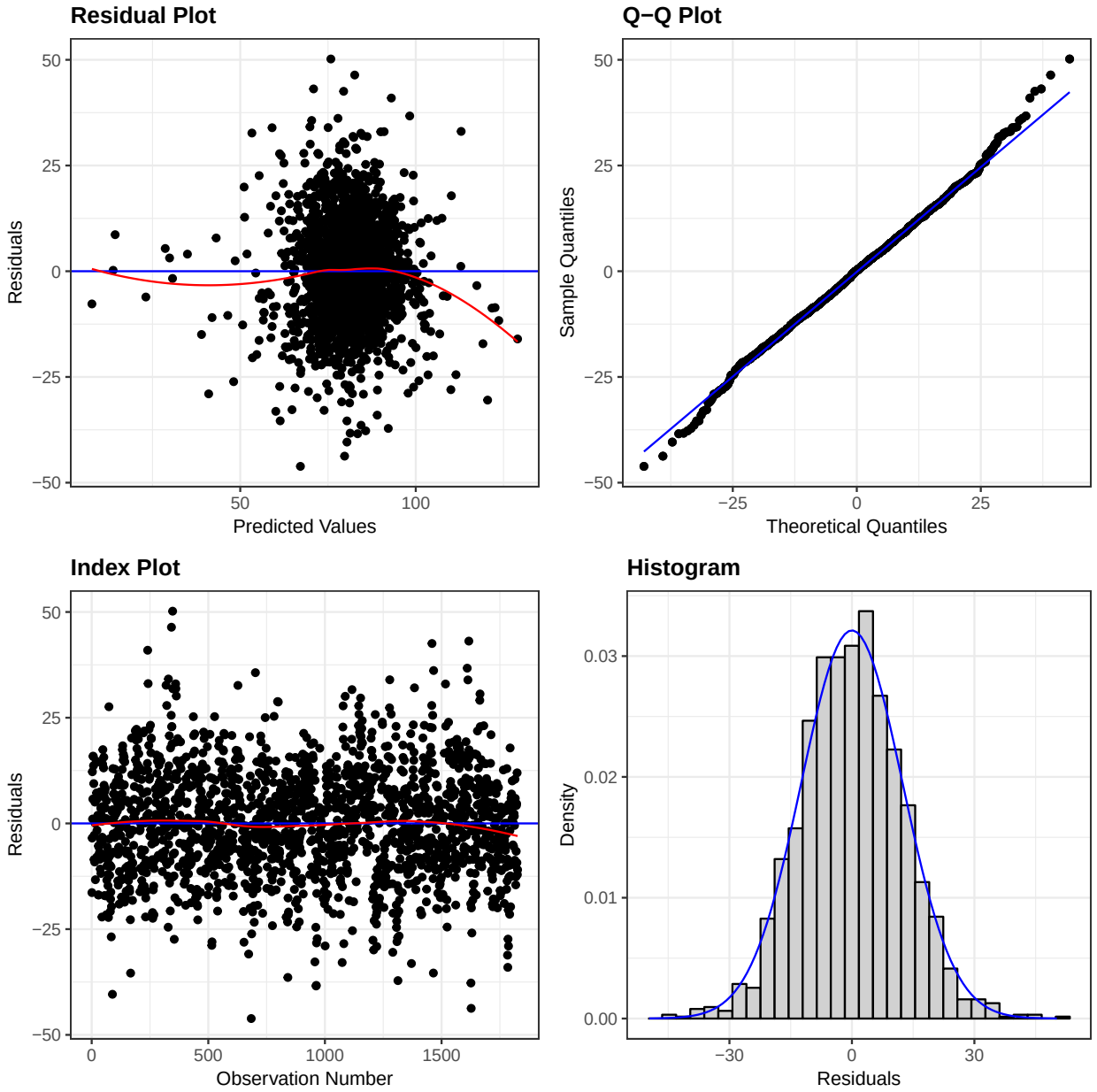
Model 3:

```
# Removing : TEAM_PITCHING_BB - based on p-value
g3 <- lm(TARGET_WINS ~TEAM_BATTING_1B + TEAM_BATTING_2B +
           TEAM_BATTING_HR + TEAM_BATTING_BB + TEAM_BATTING_SO +
           TEAM_BASERUN_SB + TEAM_PITCHING_H +
           TEAM_FIELDING_E + TEAM_FIELDING_DP, data = training_set)


summary(g3)
```

```
##
## Call:
## lm(formula = TARGET_WINS ~ TEAM_BATTING_1B + TEAM_BATTING_2B +
##     TEAM_BATTING_HR + TEAM_BATTING_BB + TEAM_BATTING_SO + TEAM_BASERUN_SB +
##     TEAM_PITCHING_H + TEAM_FIELDING_E + TEAM_FIELDING_DP, data = training_set)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -48.454  -8.516   0.493   8.234  51.173
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)      38.9664493  5.4295694   7.177 1.04e-12 ***
## TEAM_BATTING_1B   0.0444661  0.0038055  11.685  < 2e-16 ***
## TEAM_BATTING_2B   0.0320488  0.0077448   4.138 3.66e-05 ***
## TEAM_BATTING_HR   0.1224962  0.0091307  13.416  < 2e-16 ***
## TEAM_BATTING_BB   0.0113947  0.0033496   3.402 0.000684 ***
## TEAM_BATTING_SO  -0.0185025  0.0023936  -7.730 1.77e-14 ***
## TEAM_BASERUN_SB   0.0609333  0.0044339  13.743  < 2e-16 ***
## TEAM_PITCHING_H   0.0010385  0.0003508   2.961 0.003108 **
## TEAM_FIELDING_E  -0.0410182  0.0029818 -13.756  < 2e-16 ***
## TEAM_FIELDING_DP -0.1282933  0.0139448  -9.200  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12.57 on 1816 degrees of freedom
## Multiple R-squared:  0.3881, Adjusted R-squared:  0.3851
## F-statistic:   128 on 9 and 1816 DF,  p-value: < 2.2e-16
```

Model Diagnostic Plots

```
resid_panel(g1, plots='default', smoother = TRUE)
```



RMSE Calculation

```
### RMSE of third model - training dataset
rmse_calc(training_set$TARGET_WINS, predict(g3, newdata = training_set))
```

```
## [1] 12.53268
```

```
### RMSE of third model - test dataset
rmse_calc(test_set$TARGET_WINS, predict(g3, newdata = test_set))
```

```
## [1] 12.57679
```

Model Summary

```
model_3 <- as.data.frame(glance(g3))
model_3$Train_RMSE <- rmse_calc(training_set$TARGET_WINS, predict(g3, newdata = training_set))
model_3$Test_RMSE <- rmse_calc(test_set$TARGET_WINS, predict(g3, newdata = test_set))
```

Model 4: Stepwise Regression Model

For the fourth model, we have used backward elimination process with all variables plus higher order polynomials for the features. We have used the stepAIC() function from MASS package, which choose the best model by AIC.

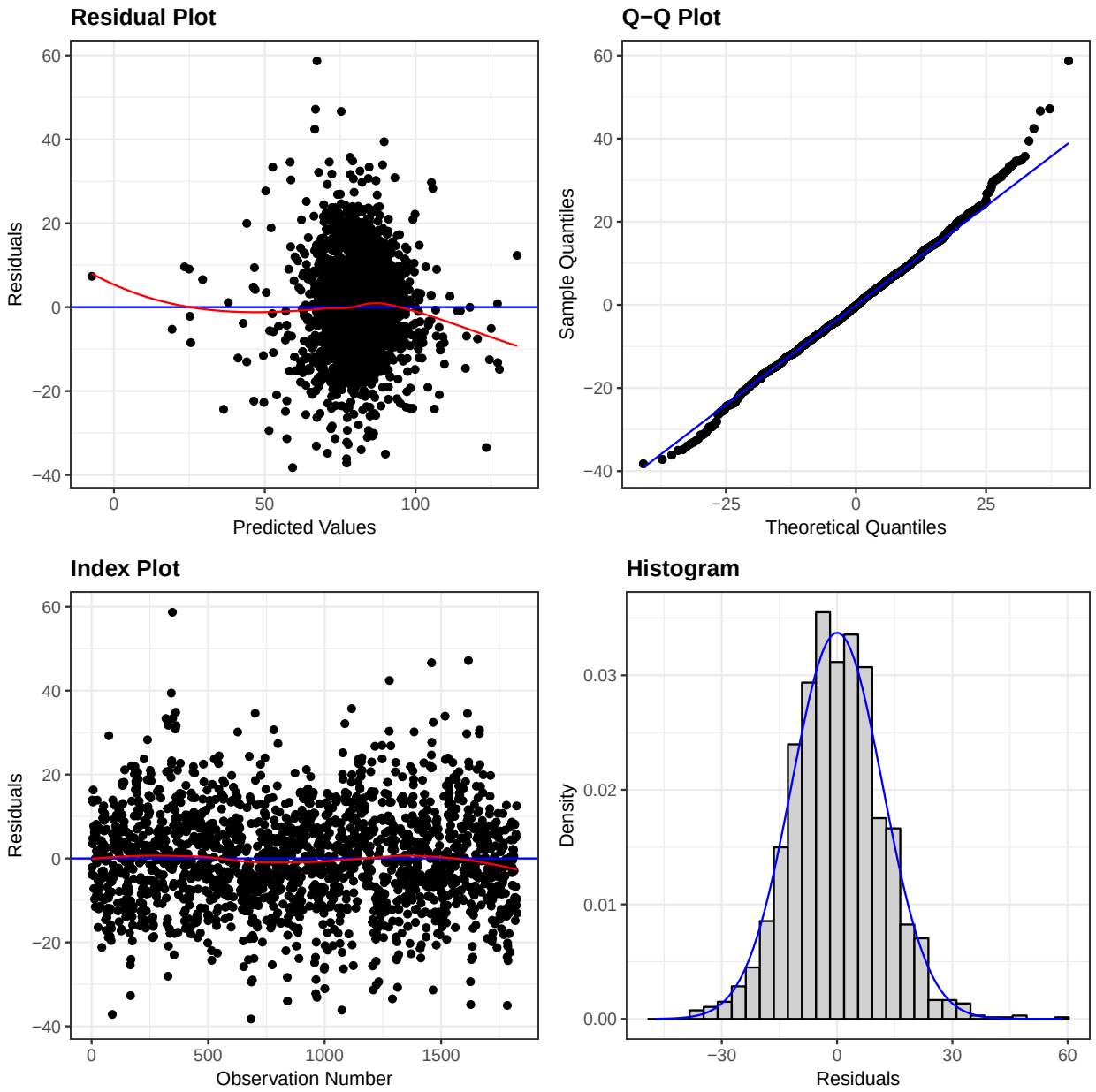
```
full_formula <- "TARGET_WINS ~ TEAM_BATTING_1B + TEAM_BATTING_2B + TEAM_BATTING_3B + TEAM_BATTING_HR + T

full_model <- lm(full_formula, training_set)

step_model <- stepAIC(full_model, direction = "backward",
                      trace = FALSE)

summary(step_model)
```

```
##
## Call:
## lm(formula = TARGET_WINS ~ TEAM_BATTING_1B + TEAM_BATTING_2B +
##     TEAM_BATTING_3B + TEAM_BATTING_HR + TEAM_BATTING_BB + TEAM_BATTING_SO +
##     TEAM_BASERUN_SB + TEAM_PITCHING_BB + TEAM_FIELDING_E + TEAM_FIELDING_DP +
##     I(TEAM_BATTING_1B^2) + I(TEAM_BATTING_3B^2) + I(TEAM_BATTING_BB^2) +
##     I(TEAM_BATTING_SO^2) + I(TEAM_BASERUN_SB^2) + I(TEAM_PITCHING_BB^2) +
##     I(TEAM_FIELDING_E^2) + I(TEAM_FIELDING_DP^2), data = training_set)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -38.245  -7.737  -0.246   7.499  58.689
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)            1.415e+02  1.410e+01  10.036  < 2e-16 ***
## TEAM_BATTING_1B       -5.314e-02  2.010e-02  -2.643 0.008283 **
## TEAM_BATTING_2B        2.902e-02  7.545e-03   3.847 0.000124 ***
## TEAM_BATTING_3B        2.063e-01  4.496e-02   4.589 4.76e-06 ***
## TEAM_BATTING_HR        1.217e-01  9.350e-03  13.016  < 2e-16 ***
## TEAM_BATTING_BB       -1.672e-01  1.745e-02  -9.580  < 2e-16 ***
## TEAM_BATTING_SO        2.900e-02  8.201e-03   3.536 0.000416 ***
## TEAM_BASERUN_SB        7.621e-02  9.608e-03   7.932 3.75e-15 ***
## TEAM_PITCHING_BB       1.618e-02  7.719e-03   2.096 0.036195 *
## TEAM_FIELDING_E       -8.866e-02  6.505e-03 -13.630  < 2e-16 ***
## TEAM_FIELDING_DP      -4.769e-01  8.126e-02  -5.870 5.19e-09 ***
## I(TEAM_BATTING_1B^2)   4.129e-05  8.048e-06   5.130 3.20e-07 ***
## I(TEAM_BATTING_3B^2)  -4.483e-04  2.686e-04  -1.669 0.095266 .
## I(TEAM_BATTING_BB^2)   1.684e-04  1.554e-05  10.840  < 2e-16 ***
## I(TEAM_BATTING_SO^2)  -2.873e-05  5.272e-06  -5.449 5.77e-08 ***
## I(TEAM_BASERUN_SB^2)  -2.452e-05  1.687e-05  -1.454 0.146182
## I(TEAM_PITCHING_BB^2) -7.055e-06  3.142e-06  -2.245 0.024868 *
## I(TEAM_FIELDING_E^2)   2.228e-05  3.945e-06   5.649 1.87e-08 ***
## I(TEAM_FIELDING_DP^2)  1.172e-03  2.793e-04   4.197 2.83e-05 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11.89 on 1807 degrees of freedom
## Multiple R-squared:  0.4552, Adjusted R-squared:  0.4498
## F-statistic:  83.9 on 18 and 1807 DF,  p-value: < 2.2e-16
```

Model Diagnostic Plots

```
resid_panel(step_model, plots='default', smoother = TRUE)
```



RMSE Calculation

```
### RMSE of fourth model - training dataset
rmse_calc(training_set$TARGET_WINS, predict(step_model, newdata = training_set))
```

```
## [1] 11.82498
```

```
### RMSE of fourth model - test dataset
rmse_calc(test_set$TARGET_WINS, predict(step_model, newdata = test_set))
```

```
## [1] 13.20033
```

Model Summary

```
model_4 <- as.data.frame(glance(step_model))
model_4$Train_RMSE <- rmse_calc(training_set$TARGET_WINS, predict(step_model, newdata = training_set))
model_4$Test_RMSE <- rmse_calc(test_set$TARGET_WINS, predict(step_model, newdata = test_set))
```

PART IV: Selecting Models

Compare Key statistics

The table below summarizes the model statistics for all four of our models. The models are listed from left to right in accordance with the order in which they were described in Part III.

```
model_summary <- rbind(model_1,model_2,model_3,model_4)

Model_Name <- c('Model 1(All Features)','Model 2 (Selective Features)','Model 3(Selective Features)','M

model_summary <- t(model_summary)
colnames(model_summary) <- Model_Name

model_summary %>% kable() %>%
    kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive")) %>%  scroll_box
```

	Model 1(All Features)	Model 2 (Selective Features)	Model 3(Selective Features)	Model 4(Stepwise -
r.squared	3.995815e-01	3.971795e-01	3.880940e-01	
adj.r.squared	3.956074e-01	3.935240e-01	3.850615e-01	
sigma	1.245891e+01	1.248036e+01	1.256714e+01	
statistic	1.005467e+02	1.086534e+02	1.279751e+02	
p.value	0.000000e+00	0.000000e+00	0.000000e+00	
df	1.300000e+01	1.200000e+01	1.000000e+01	
logLik	-7.190427e+03	-7.194072e+03	-7.207730e+03	
AIC	1.440885e+04	1.441414e+04	1.443746e+04	
BIC	1.448599e+04	1.448577e+04	1.449807e+04	
deviance	2.814219e+05	2.825477e+05	2.868062e+05	
df.residual	1.813000e+03	1.814000e+03	1.816000e+03	
Train_RMSE	1.241448e+01	1.243929e+01	1.253268e+01	
Test_RMSE	1.272149e+01	1.298812e+01	1.257679e+01	

Model Interpretation

Between the four models, we looked at the relevant diagnostic plots like residual plot, Q-Q plot, Histogram etc. We also revewed the adjusted R square and computed RMSE numbers for all 4 models.

We have decided to go with Model 4 which showed maximum Adj. R squared and minimum RMSE values. Below are the coefficients for the selected best model.

```
model_coefficients <- step_model$coefficients

model_coefficients
```

```
##           (Intercept)       TEAM_BATTING_1B       TEAM_BATTING_2B
##          1.415360e+02         -5.313552e-02          2.902182e-02
##       TEAM_BATTING_3B       TEAM_BATTING_HR       TEAM_BATTING_BB
##          2.063066e-01          1.217027e-01         -1.671953e-01
##       TEAM_BATTING_SO       TEAM_BASERUN_SB      TEAM_PITCHING_BB
##          2.899771e-02          7.620586e-02          1.618197e-02
##       TEAM_FIELDING_E      TEAM_FIELDING_DP  I(TEAM_BATTING_1B^2)
##         -8.865551e-02         -4.769411e-01          4.129007e-05
##  I(TEAM_BATTING_3B^2)  I(TEAM_BATTING_BB^2)  I(TEAM_BATTING_SO^2)
##         -4.483305e-04          1.684286e-04         -2.872571e-05
##  I(TEAM_BASERUN_SB^2) I(TEAM_PITCHING_BB^2)  I(TEAM_FIELDING_E^2)
##         -2.451895e-05         -7.055294e-06          2.228347e-05
## I(TEAM_FIELDING_DP^2)
##          1.172112e-03
```

Model Prediction

```
lm_predicted <- round(predict(step_model, newdata = eval_df),0)

lm_predicted_df <- as.data.frame(cbind(eval_df$INDEX, lm_predicted))

colnames(lm_predicted_df) <- c('INDEX','TARGET_WINS')

lm_predicted_df %>% kable() %>%
    kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive")) %>%  scroll_box
```

INDEX	TARGET_WINS
9	58
10	62
14	70
47	87
60	87
63	73
74	78
83	76
98	71
120	70
123	65
135	82
138	82
140	80
151	85
153	76
171	71
184	76
193	75
213	86
217	82
226	83
230	82
241	68
291	81
294	87
300	35
348	71
350	82
357	70
367	96
368	86
372	89
382	91
388	78
396	85
398	75
403	89
407	83
410	89
412	82
414	95
436	13
440	119
476	96
479	80
481	97
501	79
503	67
506	77
519	74
522	82
550	73
554	24 75
566	72
578	80
596	89

Statistical summary of the Prediction Output

```
result_summary <- round(rbind(as.data.frame(describe(train_df %>% dplyr::select(TARGET_WINS))),
                        as.data.frame(describe(lm_predicted_df %>% dplyr::select(TARGET_WINS)))),0)

dataset_label <- data.frame(c('Training Data','Model Prediction'))

result_summary <- cbind(dataset_label,result_summary)
names(result_summary)[1] <- 'Data Set'
rownames(result_summary) <- NULL

result_summary  %>% kable() %>%
    kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive")) %>%  scroll_box
```

Data Set	vars	n	mean	sd	median	trimmed	mad	min	max	range	skew	kurtosis	se
Training Data	1	2276	81	16	82	81	15	0	146	146	0	1	0
Model Prediction	1	259	80	12	80	80	9	13	122	109	-1	5	1

Conclusion

Based on the above table, mean and median of the training data set and prediction output are pretty close with prediction range 13 and 122. So even if not perfect, but the model output seems to be within acceptable levels of accuracy. But definitely there are further improvements that can be applied on this model. Probably more advanced technique and domain knowledge can be used to better handle outliers present in the training data set to improve the model quality.

Model output

```
# export to .csv for submission
write.csv(lm_predicted_df, file = "C:/CUNY/Semester4(Spring)/DATA 621/Assignments/Homework1/Output/money
```

Our model prediction output can be found in the below GitHub location -

Model Output

References:

Multicollinearity & VIF - Multicollinearity essentials and VIF in R

Broom Package - Model Summary Statistics

MICE Tutorial for Imputation - MICE Package