DATA_621_HW1

Chi Pong, Euclid Zhang, Jie Zou, Joseph Connolly, LeTicia Cancel

1/30/2022

```
train_raw_df <- read.csv("https://raw.githubusercontent.com/ezaccountz/DATA_621/main/HW1/moneyball-trai
test_raw_df <- read.csv("https://raw.githubusercontent.com/ezaccountz/DATA_621/main/HW1/moneyball-evalu

train_raw_df$INDEX <- NULL
test_raw_df$INDEX <- NULL
```

DATA EXPLORATION

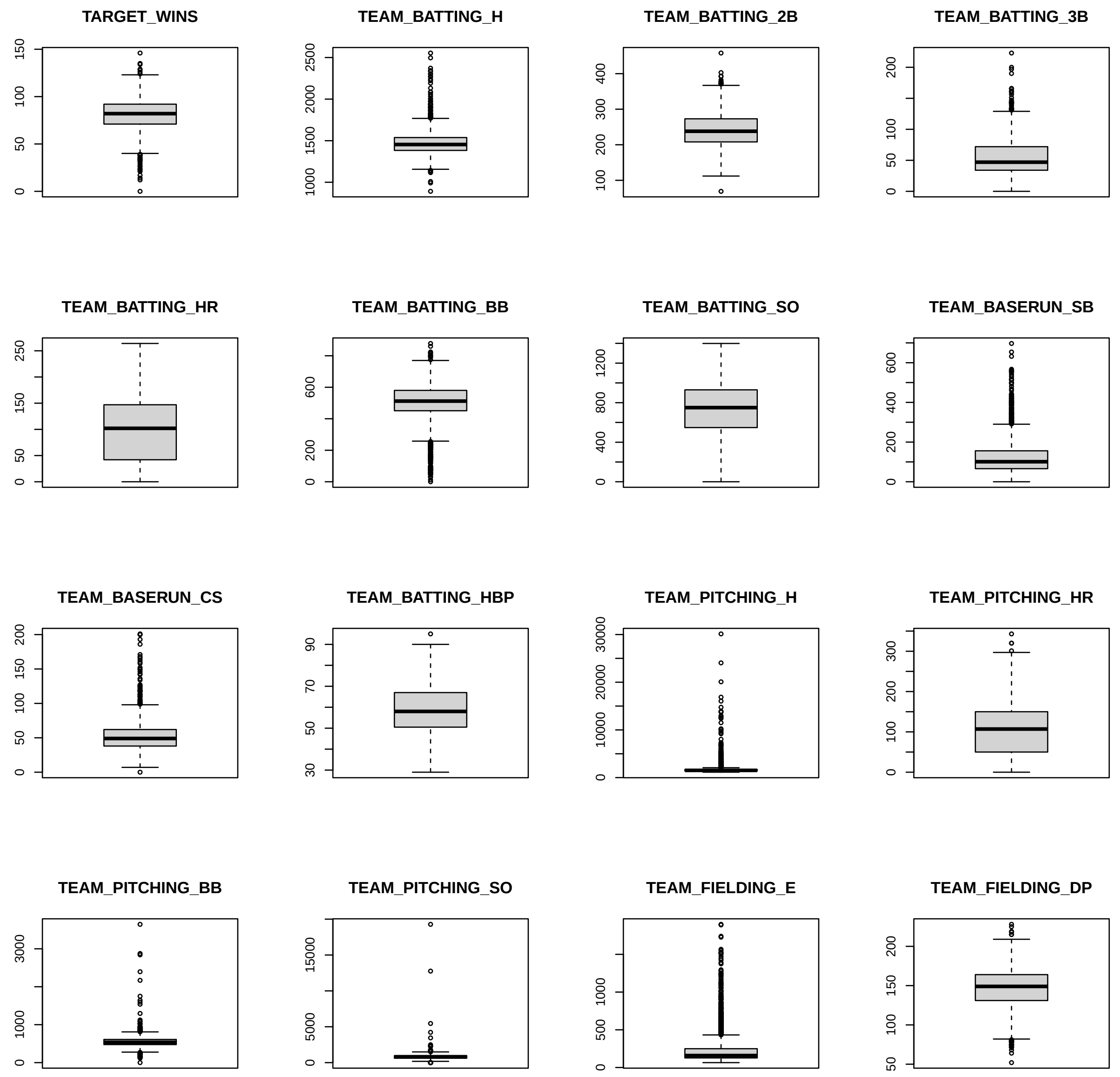
```
summary(train_raw_df)
```

```
##   TARGET_WINS     TEAM_BATTING_H TEAM_BATTING_2B TEAM_BATTING_3B
##  Min.   :  0.00   Min.   : 891   Min.   : 69.0   Min.   :  0.00
##  1st Qu.: 71.00   1st Qu.:1383   1st Qu.:208.0   1st Qu.: 34.00
##  Median : 82.00   Median :1454   Median :238.0   Median : 47.00
##  Mean   : 80.79   Mean   :1469   Mean   :241.2   Mean   : 55.25
##  3rd Qu.: 92.00   3rd Qu.:1537   3rd Qu.:273.0   3rd Qu.: 72.00
##  Max.   :146.00   Max.   :2554   Max.   :458.0   Max.   :223.00
##
##  TEAM_BATTING_HR  TEAM_BATTING_BB TEAM_BATTING_SO  TEAM_BASERUN_SB
##  Min.   :  0.00   Min.   :  0.0   Min.   :   0.0   Min.   :  0.0
##  1st Qu.: 42.00   1st Qu.:451.0   1st Qu.: 548.0   1st Qu.: 66.0
##  Median :102.00   Median :512.0   Median : 750.0   Median :101.0
##  Mean   : 99.61   Mean   :501.6   Mean   : 735.6   Mean   :124.8
##  3rd Qu.:147.00   3rd Qu.:580.0   3rd Qu.: 930.0   3rd Qu.:156.0
##  Max.   :264.00   Max.   :878.0   Max.   :1399.0   Max.   :697.0
##                                   NA's   :102      NA's   :131
##  TEAM_BASERUN_CS TEAM_BATTING_HBP TEAM_PITCHING_H TEAM_PITCHING_HR
##  Min.   :  0.0   Min.   :29.00    Min.   : 1137   Min.   :  0.0
##  1st Qu.: 38.0   1st Qu.:50.50    1st Qu.: 1419   1st Qu.: 50.0
##  Median : 49.0   Median :58.00    Median : 1518   Median :107.0
##  Mean   : 52.8   Mean   :59.36    Mean   : 1779   Mean   :105.7
##  3rd Qu.: 62.0   3rd Qu.:67.00    3rd Qu.: 1682   3rd Qu.:150.0
##  Max.   :201.0   Max.   :95.00    Max.   :30132   Max.   :343.0
##  NA's   :772     NA's   :2085
##  TEAM_PITCHING_BB TEAM_PITCHING_SO  TEAM_FIELDING_E  TEAM_FIELDING_DP
##  Min.   :   0.0   Min.   :    0.0   Min.   :  65.0   Min.   : 52.0
##  1st Qu.: 476.0   1st Qu.:  615.0   1st Qu.: 127.0   1st Qu.:131.0
##  Median : 536.5   Median :  813.5   Median : 159.0   Median :149.0
##  Mean   : 553.0   Mean   :  817.7   Mean   : 246.5   Mean   :146.4
##  3rd Qu.: 611.0   3rd Qu.:  968.0   3rd Qu.: 249.2   3rd Qu.:164.0
##  Max.   :3645.0   Max.   :19278.0   Max.   :1898.0   Max.   :228.0
##                   NA's   :102                        NA's   :286
```

Outliers

From the summaries The maximum values for TEAM_PITCHING_H, TEAM_PITCHING_BB, TEAM_PITCHING_SO and TEAM_FIELDING_E seem abnormally large. There may be outliers in the columns. We can confirm this finding by checking the distributions of the values:

```r
par(mfrow=c(4,4))
for(i in c(1:16)) {
  boxplot(train_raw_df[,i],main=colnames(train_raw_df)[i])
}
```



From the boxplots, there are indeed values in TEAM_PITCHING_H, TEAM_PITCHING_BB, TEAM_PITCHING_SO and TEAM_FIELDING_E that are extremly off from the majority of the data. We would handle these outliers later with the other problems.

Missing values

From the summaries, we see that are are missing values for TEAM_BATTING_SO, TEAM_BASERUN_SB, TEAM_BASERUN_CS, EAM_BATTING_HBP, TEAM_PITCHING_SO, and TEAM_FIELDING_DP. Now we check the portion of missing data in each field:

```r
sapply(train_raw_df,function(x)sum(is.na(x)))/nrow(train_raw_df)
```

```
##      TARGET_WINS   TEAM_BATTING_H  TEAM_BATTING_2B  TEAM_BATTING_3B
##       0.00000000       0.00000000       0.00000000       0.00000000
##   TEAM_BATTING_HR  TEAM_BATTING_BB  TEAM_BATTING_SO  TEAM_BASERUN_SB
##       0.00000000       0.00000000       0.04481547       0.05755712
##   TEAM_BASERUN_CS TEAM_BATTING_HBP  TEAM_PITCHING_H TEAM_PITCHING_HR
##       0.33919156       0.91608084       0.00000000       0.00000000
## TEAM_PITCHING_BB TEAM_PITCHING_SO  TEAM_FIELDING_E TEAM_FIELDING_DP
##       0.00000000       0.04481547       0.00000000       0.12565905
```

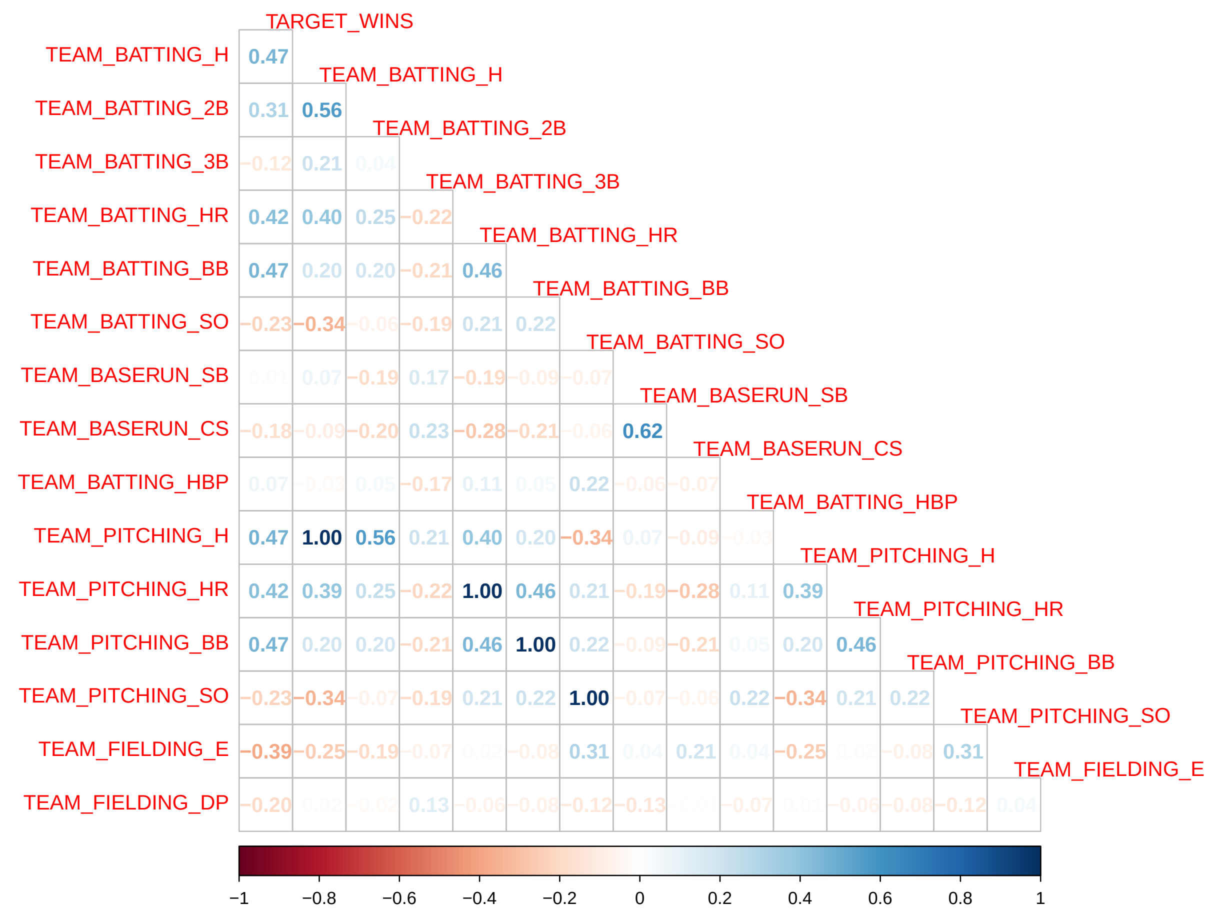
91.6% of the data in TEAM_BATTING_HBP are missing. Since the minimum of TEAM_BATTING_HBP is 29, it is not plausible that the missing values are all 0. We will drop this field from our analysis as there are too many missing values and there is no good way of imputing the values.

For TEAM_BATTING_SO, TEAM_BASERUN_SB, TEAM_BASERUN_CS, TEAM_PITCHING_SO, and TEAM_FIELDING_DP, we will do imputation and may be handled with other problems.

Correlations

Now let's look at the correlations between the variables

```
corrplot(cor(train_raw_df, use = "na.or.complete"), method = 'number',
         type = 'lower', diag = FALSE, tl.srt = 0.1)
```



The following variables are nearly perfectly correlated

- TEAM_BATTING_H and TEAM_PITCHING_H
- TEAM_BATTING_HR and TEAM_PITCHING_HR
- TEAM_BATTING_BB and TEAM_PITCHING_BB
- TEAM_BATTING_SO and TEAM_PITCHING_SO

We take a more careful look at the correlation between the TARGET_WINS and other variables, and compare it with the theoretical effects

```r
corr_table <- data.frame(correlation_with_TARGET_WINS =
                             round(cor(train_raw_df, use = "na.or.complete"),4)[-1,"TARGET_WINS"])
corr_table$Theoretical_Effect <- c("Positive","Positive","Positive","Positive",
                                    "Positive","Negative","Positive","Negative",
                                    "Positive","Negative","Negative","Negative",
                                    "Positive","Negative","Positive")
corr_table
```

```
##                  correlation_with_TARGET_WINS Theoretical_Effect
## TEAM_BATTING_H                         0.4699           Positive
## TEAM_BATTING_2B                        0.3130           Positive
## TEAM_BATTING_3B                       -0.1243           Positive
## TEAM_BATTING_HR                        0.4224           Positive
## TEAM_BATTING_BB                        0.4687           Positive
## TEAM_BATTING_SO                       -0.2289           Negative
## TEAM_BASERUN_SB                        0.0148           Positive
## TEAM_BASERUN_CS                       -0.1788           Negative
## TEAM_BATTING_HBP                       0.0735           Positive
## TEAM_PITCHING_H                        0.4712           Negative
## TEAM_PITCHING_HR                       0.4225           Negative
## TEAM_PITCHING_BB                       0.4684           Negative
## TEAM_PITCHING_SO                      -0.2294           Positive
## TEAM_FIELDING_E                       -0.3867           Negative
## TEAM_FIELDING_DP                      -0.1959           Positive
```

**The following variables do not have correlation matching with the theoretical effect:

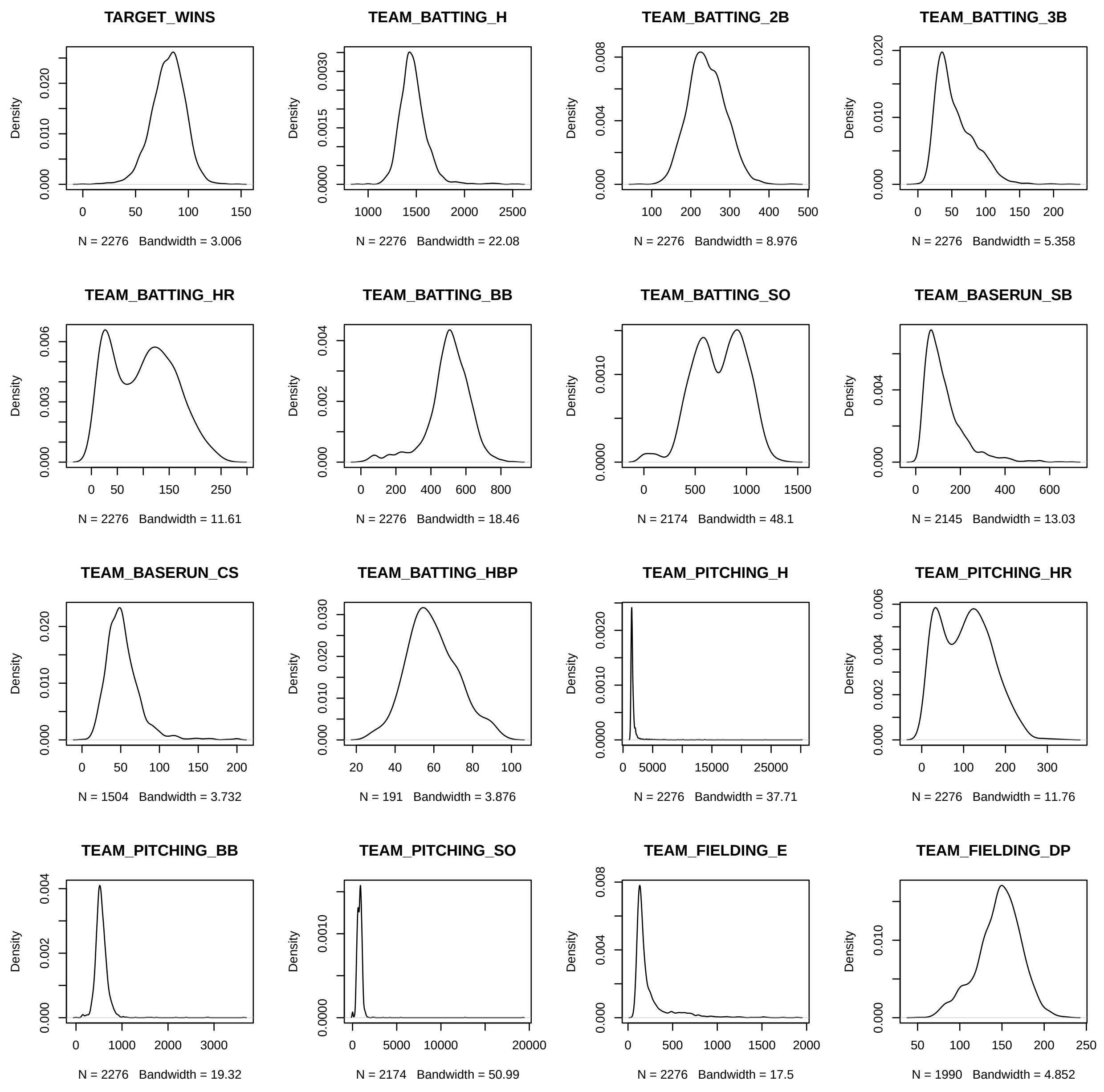
- TEAM_BATTING_3B
- TEAM_PITCHING_H
- TEAM_PITCHING_HR
- TEAM_PITCHING_BB
- TEAM_PITCHING_SO
- TEAM_FIELDING_DP

For TEAM_BATTING_3B and TEAM_FIELDING_DP, we would need to perform deeper analysis on this finding.
For TEAM_PITCHING_H, TEAM_PITCHING_HR, TEAM_PITCHING_BB, TEAM_PITCHING_SO, we may consider dropping the variables since they are amlost perfectly correlated with one other variable. Also, TEAM_PITCHING_H, TEAM_PITCHING_BB and TEAM_PITCHING_SO have outlier problem as found above.

Normalities

We check the normalities of the variables to determine if transformation is needed.

```
par(mfrow=c(4,4))
for(i in c(1:16)) {
  plot(density(train_raw_df[,i],na.rm=TRUE),main=colnames(train_raw_df)[i])
}
```



TEAM_PITCHING_H, TEAM_PITCHING_BB, TEAM_PITCHING_SO and TEAM_FIELDING_E are largely right skewed because of the outliers. We will check the distributions later again when the outliers are handled.
TEAM_BATTING_HR, TEAM_BATTING_SO, TEAM_PITCHING_HR are bimodal that may need to be transformed.
There are some variables such as TEAM_BASERUN_SB that are slightly right skewed. We may keep them as it for easier interpretation of the result.

DATA PREPARATION

Dropping variables

TEAM_BATTING_HBP is dropped for the following reason(s)

- 91.6% of the data are missing

TEAM_PITCHING_HR is dropped for the following reason(s)

- nearly perfectly correlated with one other variable
- do not have correlation matching with the theoretical effect

TEAM_PITCHING_H,TEAM_PITCHING_BB,TEAM_PITCHING_SO are dropped for the following reason(s)

- nearly perfectly correlated with one other variable
- do not have correlation matching with the theoretical effect
- large outliers

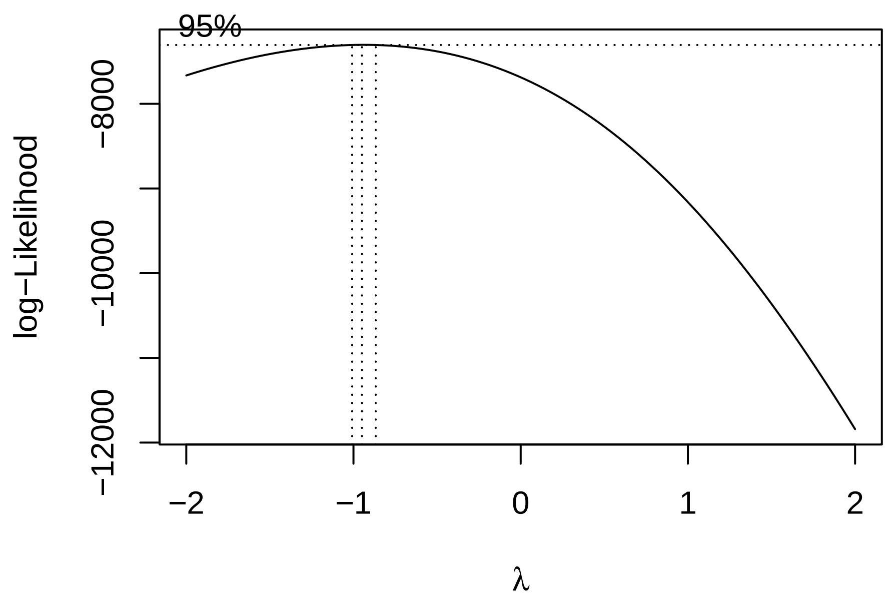
```
train_prepared_df <- train_raw_df
train_prepared_df$TEAM_BATTING_HBP <- NULL
train_prepared_df$TEAM_PITCHING_HR <- NULL
train_prepared_df$TEAM_PITCHING_H <- NULL
train_prepared_df$TEAM_PITCHING_BB <- NULL
train_prepared_df$TEAM_PITCHING_SO <- NULL
```

Transforming variables

Since **TEAM_FIELDING_E** is extremely right skewed, we will transform the variable using Box-Cox transformation

The optimal lamba from the following result plot is near -1, so we will transform the variable using the power of -1

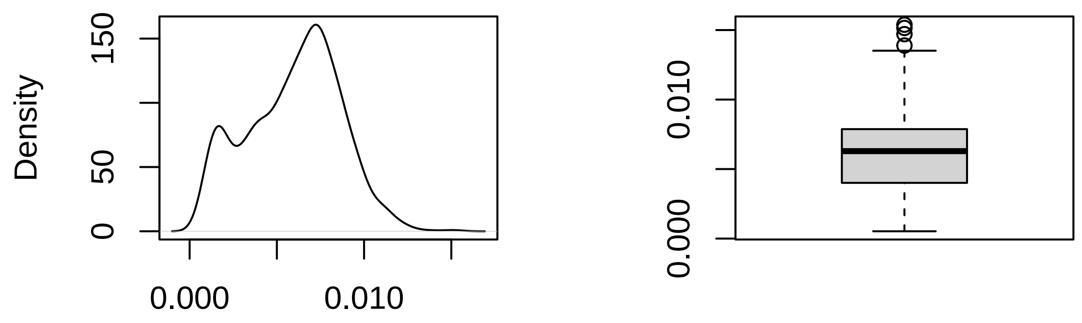
```
boxcox(lm(train_prepared_df$TEAM_FIELDING_E ~ 1))
```



```
train_prepared_df$TEAM_FIELDING_E_Transformed <- train_prepared_df$TEAM_FIELDING_E^(-1)
train_prepared_df$TEAM_FIELDING_E <- NULL
```

The following density plot and box plot show that the distribution is closer to normal and there are no extreme ourliers.

```
par(mfrow=c(1,2))
plot(density(train_prepared_df$TEAM_FIELDING_E_Transformed),main="",xlab="")
boxplot(train_prepared_df$TEAM_FIELDING_E_Transformed,main="")
```

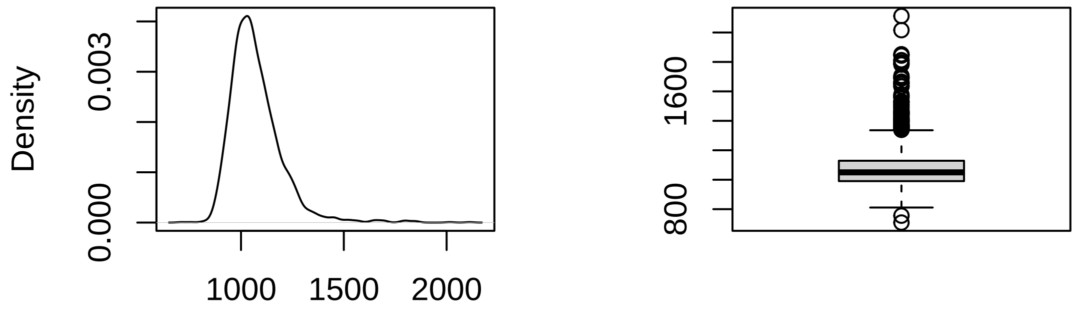


Since **TEAM_BATTING_H** also counts the number of **TEAM_BATTING_2B**, **TEAM_BATTING_3B** and **TEAM_BATTING_3B**.
Instead of using **TEAM_BATTING_H**, we will create a new variable **TEAM_BATTING_1B** by subtracting **TEAM_BATTING_2B**, **TEAM_BATTING_3B** and **TEAM_BATTING_3B** from **TEAM_BATTING_H**

```
train_prepared_df$TEAM_BATTING_1B <- train_prepared_df$TEAM_BATTING_H -
                                    train_prepared_df$TEAM_BATTING_2B -
                                    train_prepared_df$TEAM_BATTING_3B -
                                    train_prepared_df$TEAM_BATTING_HR
train_prepared_df$TEAM_BATTING_H <- NULL
```

The distribution of the new variable is slightly right skewed. We will keep it as it for now unless transformation is necessary when developing a model.

```
par(mfrow=c(1,2))
plot(density(train_prepared_df$TEAM_BATTING_1B),main="",xlab="")
boxplot(train_prepared_df$TEAM_BATTING_1B,main="")
```



Imputation for Missing Values

The only problem now is the missing values in **TEAM_BATTING_SO**, **TEAM_BASERUN_SB**, **TEAM_BASERUN_CS** and **TEAM_FIELDING_DP**

We will impute the missing values by using linear regression models. We will not go deep into evaluating these models in this project. The purpose here is to impute missing values have better explained variance than simply using the means or medians.

In the imputation models, **TARGET_WINS** is not included as an independent variable since the values are not provided in the test data set. We will need to use the same models to impute missing values in the test data set.

We will perform separate imputations in the following order, based on the number of missing values: 1. **TEAM_BATTING_SO** 2. **TEAM_BASERUN_SB** 3. **TEAM_FIELDING_DP** 4. **TEAM_BASERUN_CS**

Variables that are already imputed are included in the models after and un-imputed variables are not included. We will keep the trained models and use them to impute missing values in the test data set.

Before imputations:

```
missing_df <- data.frame(train_prepared_df$TEAM_BATTING_SO,
                         train_prepared_df$TEAM_BASERUN_SB,
                         train_prepared_df$TEAM_BASERUN_CS,
                         train_prepared_df$TEAM_FIELDING_DP)

(colMeans(is.na(missing_df)))*100
```

```
##  train_prepared_df.TEAM_BATTING_SO  train_prepared_df.TEAM_BASERUN_SB
##                           4.481547                           5.755712
##  train_prepared_df.TEAM_BASERUN_CS train_prepared_df.TEAM_FIELDING_DP
##                          33.919156                          12.565905
```

```
lm_team_bat_so <- lm(TEAM_BATTING_SO ~ . - TEAM_BASERUN_SB - TEAM_BASERUN_CS -
                  TEAM_FIELDING_DP, data = train_prepared_df[,2:ncol(train_prepared_df)])

train_prepared_df[is.na(train_prepared_df$TEAM_BATTING_SO),]$TEAM_BATTING_SO <-
      predict(lm_team_bat_so,train_prepared_df[is.na(train_prepared_df$TEAM_BATTING_SO),])

lm_team_bas_sb <- lm(TEAM_BASERUN_SB ~ . - TEAM_BASERUN_CS - TEAM_FIELDING_DP,
                     data = train_prepared_df[,2:ncol(train_prepared_df)])
train_prepared_df[is.na(train_prepared_df$TEAM_BASERUN_SB),]$TEAM_BASERUN_SB <-
      predict(lm_team_bas_sb,train_prepared_df[is.na(train_prepared_df$TEAM_BASERUN_SB),])

#convert negative values of imputed TEAM_BASERUN_SB to 0
train_prepared_df[train_prepared_df$TEAM_BASERUN_SB<0,]$TEAM_BASERUN_SB <- 0

lm_team_fld_dp <- lm(TEAM_FIELDING_DP ~ . - TEAM_BASERUN_CS, data =
                       train_prepared_df[,2:ncol(train_prepared_df)])
train_prepared_df[is.na(train_prepared_df$TEAM_FIELDING_DP),]$TEAM_FIELDING_DP <-
      predict(lm_team_fld_dp,train_prepared_df[is.na(train_prepared_df$TEAM_FIELDING_DP),])

lm_team_bas_cs <- lm(TEAM_BASERUN_CS ~ ., data = train_prepared_df[,2:ncol(train_prepared_df)])
train_prepared_df[is.na(train_prepared_df$TEAM_BASERUN_CS),]$TEAM_BASERUN_CS <-
      predict(lm_team_bas_cs,train_prepared_df[is.na(train_prepared_df$TEAM_BASERUN_CS),])
```

After imputations:

```
missing_df <- data.frame(train_prepared_df$TEAM_BATTING_SO,
                         train_prepared_df$TEAM_BASERUN_SB,
                         train_prepared_df$TEAM_BASERUN_CS,
                         train_prepared_df$TEAM_FIELDING_DP)

(colMeans(is.na(missing_df)))*100
```

```
##  train_prepared_df.TEAM_BATTING_SO  train_prepared_df.TEAM_BASERUN_SB
##                                  0                                  0
##  train_prepared_df.TEAM_BASERUN_CS train_prepared_df.TEAM_FIELDING_DP
##                                  0                                  0
```

As you can all see, the 4 variables, *TEAM_BATTING_SO*, *TEAM_BASERUN_SB*, *TEAM_BASERUN_CS*, *TEAM_FIELDING_DP*, all are having 0 missing values. Meaning they are no longer presented with missing values.

The R-squared values of the models are:

```
print(paste0("TEAM_BATTING_SO - R-squared:",toString(round(summary(lm_team_bat_so)$r.squared,4))))
```

```
## [1] "TEAM_BATTING_SO - R-squared:0.7045"
```

```
print(paste0("TEAM_BASERUN_SB - R-squared:",toString(round(summary(lm_team_bas_sb)$r.squared,4))))
```

```
## [1] "TEAM_BASERUN_SB - R-squared:0.1571"
```

```
print(paste0("TEAM_BASERUN_CS - R-squared:",toString(round(summary(lm_team_bas_cs)$r.squared,4))))
```

```
## [1] "TEAM_BASERUN_CS - R-squared:0.599"
```

```
print(paste0("TEAM_FIELDING_DP - R-squared:",toString(round(summary(lm_team_fld_dp)$r.squared,4))))
```

```
## [1] "TEAM_FIELDING_DP - R-squared:0.1227"
```

The numbers in the summary of our prepared data set all look plausible, we are ready for our model development

```
summary(train_prepared_df)
```

```
##   TARGET_WINS     TEAM_BATTING_2B TEAM_BATTING_3B  TEAM_BATTING_HR
##  Min.   :  0.00   Min.   : 69.0   Min.   :  0.00   Min.   :  0.00
##  1st Qu.: 71.00   1st Qu.:208.0   1st Qu.: 34.00   1st Qu.: 42.00
##  Median : 82.00   Median :238.0   Median : 47.00   Median :102.00
##  Mean   : 80.79   Mean   :241.2   Mean   : 55.25   Mean   : 99.61
##  3rd Qu.: 92.00   3rd Qu.:273.0   3rd Qu.: 72.00   3rd Qu.:147.00
##  Max.   :146.00   Max.   :458.0   Max.   :223.00   Max.   :264.00
##  TEAM_BATTING_BB TEAM_BATTING_SO  TEAM_BASERUN_SB TEAM_BASERUN_CS
##  Min.   :  0.0   Min.   :   0.0   Min.   :  0.0   Min.   :  0.00
##  1st Qu.:451.0   1st Qu.: 542.0   1st Qu.: 67.0   1st Qu.: 43.00
##  Median :512.0   Median : 732.0   Median :101.0   Median : 57.00
##  Mean   :501.6   Mean   : 727.6   Mean   :123.4   Mean   : 69.27
##  3rd Qu.:580.0   3rd Qu.: 925.0   3rd Qu.:152.0   3rd Qu.: 86.89
##  Max.   :878.0   Max.   :1399.0   Max.   :697.0   Max.   :272.21
##  TEAM_FIELDING_DP TEAM_FIELDING_E_Transformed TEAM_BATTING_1B
##  Min.   : 52.0    Min.   :0.0005269           Min.   : 709.0
##  1st Qu.:130.0    1st Qu.:0.0040120           1st Qu.: 990.8
##  Median :146.0    Median :0.0062893           Median :1050.0
##  Mean   :144.9    Mean   :0.0059708           Mean   :1073.2
##  3rd Qu.:162.0    3rd Qu.:0.0078740           3rd Qu.:1129.0
##  Max.   :228.0    Max.   :0.0153846           Max.   :2112.0
```

BUILD MODELS

1. Full model:

TEAM_BATTING_1B, TEAM_BATTING_2B, TEAM_BATTING_3B, TEAM_BATTING_HR, TEAM_BATTING_BB, TEAM_BATTING_SO, TEAM_BASERUN_SB, TEAM_BASERUN_CS, TEAM_FIELDING_DP, TEAM_FIELDING_E_Transformed

```
lm_win_full <- lm(TARGET_WINS ~ .,data = train_prepared_df)
```

```
summary(lm_win_full)
```

```
##
## Call:
## lm(formula = TARGET_WINS ~ ., data = train_prepared_df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -68.884  -8.302   0.085   8.208  66.698
##
## Coefficients:
##                                Estimate Std. Error t value Pr(>|t|)
## (Intercept)                   1.222e+01  5.369e+00   2.277 0.022887 *
## TEAM_BATTING_2B               2.067e-02  7.396e-03   2.795 0.005240 **
## TEAM_BATTING_3B               1.437e-01  1.713e-02   8.389  < 2e-16 ***
## TEAM_BATTING_HR               8.623e-02  9.275e-03   9.297  < 2e-16 ***
## TEAM_BATTING_BB               2.181e-02  2.942e-03   7.414 1.72e-13 ***
## TEAM_BATTING_SO              -9.109e-03  2.383e-03  -3.823 0.000136 ***
## TEAM_BASERUN_SB               1.534e-02  8.865e-03   1.730 0.083685 .
## TEAM_BASERUN_CS               1.215e-02  2.325e-02   0.522 0.601455
## TEAM_FIELDING_DP             -1.368e-01  1.369e-02  -9.993  < 2e-16 ***
## TEAM_FIELDING_E_Transformed   2.373e+03  2.113e+02  11.232  < 2e-16 ***
## TEAM_BATTING_1B               4.254e-02  3.716e-03  11.450  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.1 on 2265 degrees of freedom
## Multiple R-squared:  0.3109, Adjusted R-squared:  0.3079
## F-statistic: 102.2 on 10 and 2265 DF,  p-value: < 2.2e-16
```

In the full model, The sign of **TEAM_BASERUN_CS** and **TEAM_FIELDING_DP** do not match with the theoretical effects.
By looking at the below confidence intervals with 5% significance level, The positive sign for **TEAM_BASERUN_CS** may just happen by chance. The high p-value of **TEAM_BASERUN_CS** also indicates that the variable is not significant here. With the fact that more than 30% of the values of **TEAM_BASERUN_CS** are imputed. We suggest to drop this variable from our model.
For **TEAM_FIELDING_DP**, confidence interval is below 0 and the correlation between **TARGET_WINS** and **TEAM_FIELDING_DP** is -0.2. In the correlation matrix, it doesn't show any strong correlation with any other variables. One explanation is that double play happens when there is already a runner on a base. So a higher **TEAM_FIELDING_DP** means the team let the an opponent player stay on a base more frequently. A good team will get their opponents out before double play can happen. We may need to observe the behaviors of the baseball players directly to find out the true reason

of the negative correlation. But, in this analysis, we will compare the performance of the models with or without **TEAM_FIELDING_DP** using the test data set to determine whether we should keep this variable in our model.

```
confint(lm_win_full, level = 0.95)
```

```
##                                      2.5 %        97.5 %
## (Intercept)                   1.695863e+00  2.275411e+01
## TEAM_BATTING_2B               6.165689e-03  3.517413e-02
## TEAM_BATTING_3B               1.101057e-01  1.772853e-01
## TEAM_BATTING_HR               6.804495e-02  1.044223e-01
## TEAM_BATTING_BB               1.604367e-02  2.758226e-02
## TEAM_BATTING_SO              -1.378154e-02 -4.435984e-03
## TEAM_BASERUN_SB              -2.043957e-03  3.272634e-02
## TEAM_BASERUN_CS              -3.345244e-02  5.774641e-02
## TEAM_FIELDING_DP             -1.636101e-01 -1.099324e-01
## TEAM_FIELDING_E_Transformed  1.958557e+03  2.787093e+03
## TEAM_BATTING_1B               3.525880e-02  4.983134e-02
```

2. Adjusted model 1:

exclude **TEAM_BASERUN_CS**

```
lm_win_eff_adj <- lm(TARGET_WINS ~ .-TEAM_BASERUN_CS,data = train_prepared_df)
```

```
summary(lm_win_eff_adj)
```

```
##
## Call:
## lm(formula = TARGET_WINS ~ . - TEAM_BASERUN_CS, data = train_prepared_df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -68.884  -8.330   0.106   8.252  66.698
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                  1.315e+01  5.068e+00   2.595 0.009519 **
## TEAM_BATTING_2B              2.029e-02  7.360e-03   2.757 0.005876 **
## TEAM_BATTING_3B              1.463e-01  1.637e-02   8.938  < 2e-16 ***
## TEAM_BATTING_HR              8.526e-02  9.083e-03   9.386  < 2e-16 ***
## TEAM_BATTING_BB              2.184e-02  2.941e-03   7.428 1.55e-13 ***
## TEAM_BATTING_SO             -9.211e-03  2.374e-03  -3.879 0.000108 ***
## TEAM_BASERUN_SB              1.937e-02  4.353e-03   4.451 8.97e-06 ***
## TEAM_FIELDING_DP            -1.373e-01  1.365e-02 -10.056  < 2e-16 ***
## TEAM_FIELDING_E_Transformed  2.357e+03  2.091e+02  11.274  < 2e-16 ***
## TEAM_BATTING_1B              4.225e-02  3.672e-03  11.505  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.1 on 2266 degrees of freedom
## Multiple R-squared:  0.3109, Adjusted R-squared:  0.3081
## F-statistic: 113.6 on 9 and 2266 DF,  p-value: < 2.2e-16
```

The significance of **TEAM_BASERUN_SB** increases and the Adjusted R-squared increases. Dropping **TEAM_BASERUN_CS** gives us a better result. We will also verify this in our test data set.

3. Adjusted model 2:

exclude **TEAM_BASERUN_CS** and **TEAM_FIELDING_DP**

```
lm_win_eff_adj2 <- lm(TARGET_WINS ~ .-TEAM_BASERUN_CS-TEAM_FIELDING_DP,data = train_prepared_df)
```

```
summary(lm_win_eff_adj2)
```

```
##
## Call:
## lm(formula = TARGET_WINS ~ . - TEAM_BASERUN_CS - TEAM_FIELDING_DP,
##     data = train_prepared_df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -70.701  -8.440   0.110   8.463  64.470
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                 1.731e+00  5.047e+00   0.343   0.7316
## TEAM_BATTING_2B             1.915e-02  7.520e-03   2.546   0.0110 *
## TEAM_BATTING_3B             1.527e-01  1.672e-02   9.136  < 2e-16 ***
## TEAM_BATTING_HR             7.170e-02  9.179e-03   7.811 8.59e-15 ***
## TEAM_BATTING_BB             1.560e-02  2.937e-03   5.310 1.20e-07 ***
## TEAM_BATTING_SO            -7.351e-03  2.419e-03  -3.039   0.0024 **
## TEAM_BASERUN_SB             3.324e-02  4.219e-03   7.880 5.06e-15 ***
## TEAM_FIELDING_E_Transformed  2.160e+03  2.127e+02  10.152  < 2e-16 ***
## TEAM_BATTING_1B             3.671e-02  3.710e-03   9.893  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.39 on 2267 degrees of freedom
## Multiple R-squared:  0.2801, Adjusted R-squared:  0.2776
## F-statistic: 110.3 on 8 and 2267 DF,  p-value: < 2.2e-16
```

The significance of **TEAM_BATTING_2B** and **TEAM_BATTING_SO** decreases and the Adjusted R-squared also decreases. It seems that it is better to keep **TEAM_FIELDING_DP** in our model.

4. Adjusted model 3:

exclude all variables with imputed values (**TEAM_BATTING_SO**, **TEAM_BASERUN_SB**, **TEAM_BASERUN_CS** and **TEAM_FIELDING_DP**)

```
lm_win_exc_mis <- lm(TARGET_WINS ~ .-TEAM_BATTING_SO-TEAM_BASERUN_SB-
                       TEAM_BASERUN_CS-TEAM_FIELDING_DP,data = train_prepared_df)
```

```
summary(lm_win_exc_mis)
```

```
##
## Call:
## lm(formula = TARGET_WINS ~ . - TEAM_BATTING_SO - TEAM_BASERUN_SB -
##     TEAM_BASERUN_CS - TEAM_FIELDING_DP, data = train_prepared_df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -66.320  -8.342   0.075   8.584  64.966
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                -6.779e+00  3.601e+00  -1.882   0.0599 .
## TEAM_BATTING_2B             1.457e-02  7.590e-03   1.919   0.0551 .
## TEAM_BATTING_3B             1.923e-01  1.608e-02  11.960  < 2e-16 ***
## TEAM_BATTING_HR             5.544e-02  8.527e-03   6.502 9.72e-11 ***
## TEAM_BATTING_BB             2.237e-02  2.831e-03   7.901 4.27e-15 ***
## TEAM_FIELDING_E_Transformed  1.816e+03  2.082e+02   8.722  < 2e-16 ***
## TEAM_BATTING_1B             4.272e-02  3.085e-03  13.848  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.56 on 2269 degrees of freedom
## Multiple R-squared:  0.2604, Adjusted R-squared:  0.2584
## F-statistic: 133.1 on 6 and 2269 DF,  p-value: < 2.2e-16
```

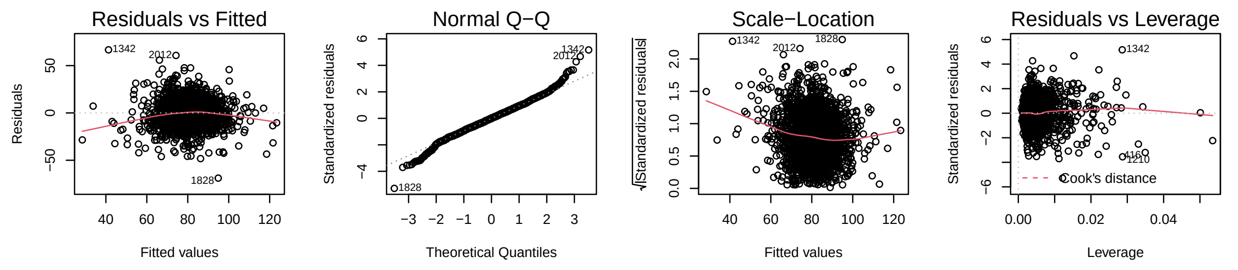
As expected, including less variables produce a model with lower R-squared value. The Adjusted R-squared is also lower in this case. However, this model is much simpler and all the variables have no missing values. We will compare the performance of this model to the other ones we have built using the test data set.

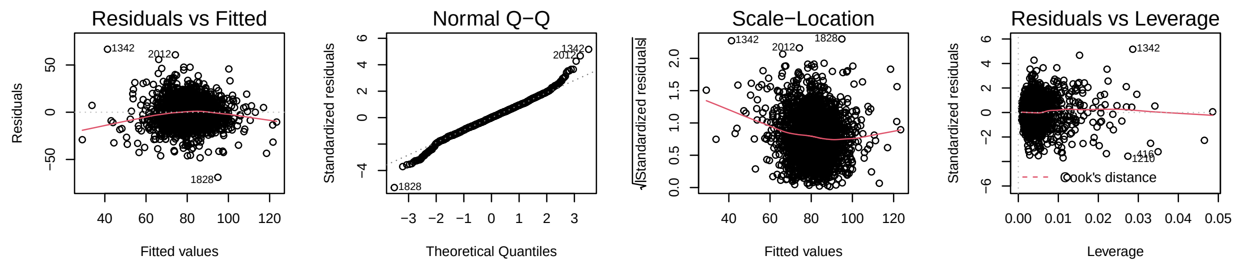
SELECT MODELS

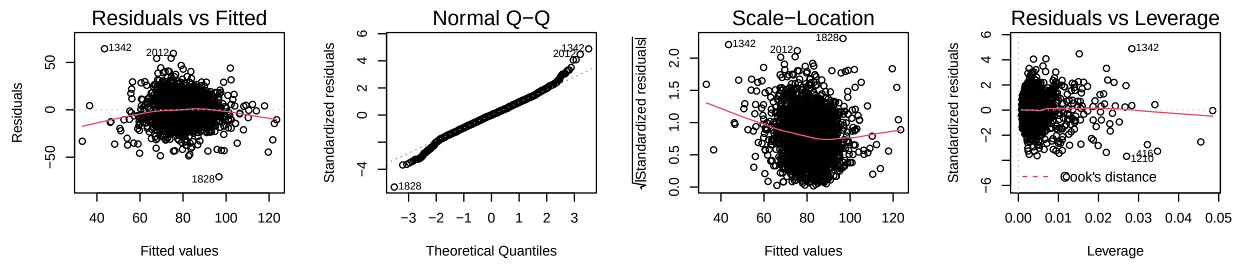
Inference Plots

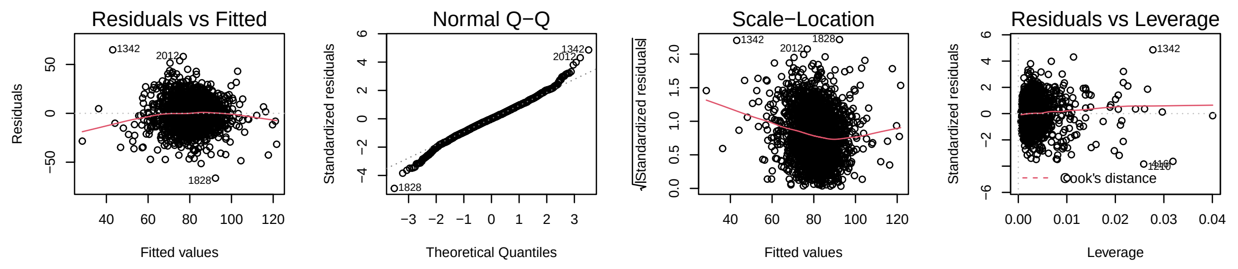
```
par(mfrow=c(4,4))

plot(lm_win_full)
plot(lm_win_eff_adj)
plot(lm_win_eff_adj2)
plot(lm_win_exc_mis)
```









The plots for all four models all look the same.

Constant Variance

The residual plots and standardized residual plots show that the residuals are independent and approximately constant with mean 0 within the cloud of data. Basically, we see in all 4 residuals vs fitted plots (Column 1) that there is approximately symmetrical variation around 0, i.e. homoscedasticity. Nor did we detect any non-linear patterns.

The Q-Q plots show that the residuals are approximately normal except the two tails but the problem is mild.
The Residuals vs Leverage plots show no point outside of the Cook's distance. There is no strong influence

point.
We conclude that all four models follow the assumptions of OLS regression and so they are valid.

F-Statistic

The F-statistic is used to measure the significance of one or more variables if they are added to a base model.
The base model is our model with **TEAM_BATTING_SO**, **TEAM_BASERUN_SB**, **TEAM_BASERUN_CS** and **TEAM_FIELDING_DP** excluded
We compare it with the model with **TEAM_BATTING_SO** and **TEAM_BASERUN_SB** added

```
anova(lm_win_exc_mis, lm_win_eff_adj2)
```

```
## Analysis of Variance Table
##
## Model 1: TARGET_WINS ~ (TEAM_BATTING_2B + TEAM_BATTING_3B + TEAM_BATTING_HR +
##     TEAM_BATTING_BB + TEAM_BATTING_SO + TEAM_BASERUN_SB + TEAM_BASERUN_CS +
##     TEAM_FIELDING_DP + TEAM_FIELDING_E_Transformed + TEAM_BATTING_1B) -
##     TEAM_BATTING_SO - TEAM_BASERUN_SB - TEAM_BASERUN_CS - TEAM_FIELDING_DP
## Model 2: TARGET_WINS ~ (TEAM_BATTING_2B + TEAM_BATTING_3B + TEAM_BATTING_HR +
##     TEAM_BATTING_BB + TEAM_BATTING_SO + TEAM_BASERUN_SB + TEAM_BASERUN_CS +
##     TEAM_FIELDING_DP + TEAM_FIELDING_E_Transformed + TEAM_BATTING_1B) -
##     TEAM_BASERUN_CS - TEAM_FIELDING_DP
##   Res.Df    RSS Df Sum of Sq      F    Pr(>F)
## 1   2269 417514
## 2   2267 406375  2     11138 31.068 4.885e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The F-value with degree of freedom 2 is 31.068, indicating that adding **TEAM_BATTING_SO** and **TEAM_BASERUN_SB** does make an improvement to the model.

Next use our improved model as our new base model check if adding **TEAM_FIELDING_DP** will make another improvement.

```
anova(lm_win_eff_adj2, lm_win_eff_adj)
```

```
## Analysis of Variance Table
##
## Model 1: TARGET_WINS ~ (TEAM_BATTING_2B + TEAM_BATTING_3B + TEAM_BATTING_HR +
##     TEAM_BATTING_BB + TEAM_BATTING_SO + TEAM_BASERUN_SB + TEAM_BASERUN_CS +
##     TEAM_FIELDING_DP + TEAM_FIELDING_E_Transformed + TEAM_BATTING_1B) -
##     TEAM_BASERUN_CS - TEAM_FIELDING_DP
## Model 2: TARGET_WINS ~ (TEAM_BATTING_2B + TEAM_BATTING_3B + TEAM_BATTING_HR +
##     TEAM_BATTING_BB + TEAM_BATTING_SO + TEAM_BASERUN_SB + TEAM_BASERUN_CS +
##     TEAM_FIELDING_DP + TEAM_FIELDING_E_Transformed + TEAM_BATTING_1B) -
##     TEAM_BASERUN_CS
##   Res.Df    RSS Df Sum of Sq      F    Pr(>F)
## 1   2267 406375
## 2   2266 389014  1     17362 101.13 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The result shows that the effect of **TEAM_FIELDING_DP** is significant

Last, let's check if adding **TEAM_BASERUN_CS** will help

```
anova(lm_win_eff_adj, lm_win_full)
```

```
## Analysis of Variance Table
##
## Model 1: TARGET_WINS ~ (TEAM_BATTING_2B + TEAM_BATTING_3B + TEAM_BATTING_HR +
##     TEAM_BATTING_BB + TEAM_BATTING_SO + TEAM_BASERUN_SB + TEAM_BASERUN_CS +
##     TEAM_FIELDING_DP + TEAM_FIELDING_E_Transformed + TEAM_BATTING_1B) -
##     TEAM_BASERUN_CS
## Model 2: TARGET_WINS ~ TEAM_BATTING_2B + TEAM_BATTING_3B + TEAM_BATTING_HR +
##     TEAM_BATTING_BB + TEAM_BATTING_SO + TEAM_BASERUN_SB + TEAM_BASERUN_CS +
##     TEAM_FIELDING_DP + TEAM_FIELDING_E_Transformed + TEAM_BATTING_1B
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1   2266 389014
## 2   2265 388967  1    46.862 0.2729 0.6015
```

The variable **TEAM_BASERUN_CS** insignificant as we have checked before using the p-value of the coefficient.

From the F-Statistics, the optimal model is the model with the following predictors:

- TEAM_BATTING_2B
- TEAM_BATTING_3B
- TEAM_BATTING_HR
- TEAM_BATTING_BB
- TEAM_BATTING_SO
- TEAM_BASERUN_SB
- TEAM_FIELDING_DP
- TEAM_FIELDING_E_Transformed
- TEAM_BATTING_1B

Adjusted R-squared and RMSD (Root Mean Square Deviation)

The R-squared or Adjusted R-squared is used to measure how well a model fit in the train data. Since our models have different number of predictor variables, it is better to compare the Adjusted R-squared of the 4 models

The RMSD (Root Mean Square Deviation) is a measurement for the difference between a model's predicted values and the actual values.

The Adjusted R-squared and RMSD for all 4 models are as following:

```
data.frame(
    model = c("Full model","Exclude TEAM_BASERUN_CS",
              "EXclude TEAM_BASERUN_CS and TEAM_FIELDING_DP",
              "Exclude variables with missing values"),
    Adjusted_R_Squared = c(summary(lm_win_full)$adj.r.squared,
                           summary(lm_win_eff_adj)$adj.r.squared,
                           summary(lm_win_eff_adj2)$adj.r.squared,
                           summary(lm_win_exc_mis)$adj.r.squared),
    Root_Mean_Square_Deviation = c(sqrt(mean(lm_win_full$residuals^2)),
                                   sqrt(mean(lm_win_eff_adj$residuals^2)),
                                   sqrt(mean(lm_win_eff_adj2$residuals^2)),
                                   sqrt(mean(lm_win_exc_mis$residuals^2)))
    )
```

```
##                                           model Adjusted_R_Squared
## 1                                    Full model          0.3079071
## 2                        Exclude TEAM_BASERUN_CS          0.3081292
## 3 EXclude TEAM_BASERUN_CS and TEAM_FIELDING_DP          0.2775695
## 4        Exclude variables with missing values          0.2584226
##   Root_Mean_Square_Deviation
## 1                   13.07284
## 2                   13.07363
## 3                   13.36219
## 4                   13.54407
```

The second model has the highest Adjusted_R_Squared value and an RMSD slightly higher than the smallest RMSD produced by the full model. We conclude that the optimal model is the same as we found from the F-Statistic tests.

Test data set prediction

Now, let's check if the models are producing plausible prediction values using the test data set.

let's look at the summary of our test data set:

```
summary(test_raw_df)
```

```
##  TEAM_BATTING_H TEAM_BATTING_2B TEAM_BATTING_3B  TEAM_BATTING_HR
##  Min.   : 819   Min.   : 44.0   Min.   : 14.00   Min.   :  0.00
##  1st Qu.:1387   1st Qu.:210.0   1st Qu.: 35.00   1st Qu.: 44.50
##  Median :1455   Median :239.0   Median : 52.00   Median :101.00
##  Mean   :1469   Mean   :241.3   Mean   : 55.91   Mean   : 95.63
##  3rd Qu.:1548   3rd Qu.:278.5   3rd Qu.: 72.00   3rd Qu.:135.50
##  Max.   :2170   Max.   :376.0   Max.   :155.00   Max.   :242.00
##
##  TEAM_BATTING_BB TEAM_BATTING_SO  TEAM_BASERUN_SB TEAM_BASERUN_CS
##  Min.   : 15.0   Min.   :   0.0   Min.   :  0.0   Min.   :  0.00
##  1st Qu.:436.5   1st Qu.: 545.0   1st Qu.: 59.0   1st Qu.: 38.00
##  Median :509.0   Median : 686.0   Median : 92.0   Median : 49.50
##  Mean   :499.0   Mean   : 709.3   Mean   :123.7   Mean   : 52.32
##  3rd Qu.:565.5   3rd Qu.: 912.0   3rd Qu.:151.8   3rd Qu.: 63.00
##  Max.   :792.0   Max.   :1268.0   Max.   :580.0   Max.   :154.00
##                  NA's   :18       NA's   :13      NA's   :87
##  TEAM_BATTING_HBP TEAM_PITCHING_H TEAM_PITCHING_HR TEAM_PITCHING_BB
##  Min.   :42.00    Min.   : 1155   Min.   :  0.0    Min.   : 136.0
##  1st Qu.:53.50    1st Qu.: 1426   1st Qu.: 52.0    1st Qu.: 471.0
##  Median :62.00    Median : 1515   Median :104.0    Median : 526.0
##  Mean   :62.37    Mean   : 1813   Mean   :102.1    Mean   : 552.4
##  3rd Qu.:67.50    3rd Qu.: 1681   3rd Qu.:142.5    3rd Qu.: 606.5
##  Max.   :96.00    Max.   :22768   Max.   :336.0    Max.   :2008.0
##  NA's   :240
##  TEAM_PITCHING_SO TEAM_FIELDING_E  TEAM_FIELDING_DP
##  Min.   :   0.0   Min.   :  73.0   Min.   : 69.0  
##  1st Qu.: 613.0   1st Qu.: 131.0   1st Qu.:131.0  
##  Median : 745.0   Median : 163.0   Median :148.0  
##  Mean   : 799.7   Mean   : 249.7   Mean   :146.1  
##  3rd Qu.: 938.0   3rd Qu.: 252.0   3rd Qu.:164.0  
##  Max.   :9963.0   Max.   :1568.0   Max.   :204.0  
##  NA's   :18                        NA's   :31
```

First, let's transform and impute the variables in the test data set.

```
test_prepared_df <- test_raw_df
test_prepared_df$TEAM_FIELDING_E_Transformed <- test_prepared_df$TEAM_FIELDING_E^(-1)

test_prepared_df$TEAM_BATTING_1B <- test_prepared_df$TEAM_BATTING_H -
                                    test_prepared_df$TEAM_BATTING_2B -
                                    test_prepared_df$TEAM_BATTING_3B -
                                    test_prepared_df$TEAM_BATTING_HR
test_prepared_df$TEAM_BATTING_H <- NULL

test_prepared_df$TEAM_BATTING_HBP <- NULL
test_prepared_df$TEAM_PITCHING_HR <- NULL
test_prepared_df$TEAM_PITCHING_H <- NULL
test_prepared_df$TEAM_PITCHING_BB <- NULL
test_prepared_df$TEAM_PITCHING_SO <- NULL
```

```
test_prepared_df[is.na(test_prepared_df$TEAM_BATTING_SO),]$TEAM_BATTING_SO <-
    predict(lm_team_bat_so,test_prepared_df[is.na(test_prepared_df$TEAM_BATTING_SO),])

test_prepared_df[is.na(test_prepared_df$TEAM_BASERUN_SB),]$TEAM_BASERUN_SB <-
    predict(lm_team_bas_sb,test_prepared_df[is.na(test_prepared_df$TEAM_BASERUN_SB),])

#convert negative values of imputed TEAM_BASERUN_SB to 0
test_prepared_df[test_prepared_df$TEAM_BASERUN_SB<0,]$TEAM_BASERUN_SB <- 0

test_prepared_df[is.na(test_prepared_df$TEAM_FIELDING_DP),]$TEAM_FIELDING_DP <-
    predict(lm_team_fld_dp,test_prepared_df[is.na(test_prepared_df$TEAM_FIELDING_DP),])

test_prepared_df[is.na(test_prepared_df$TEAM_BASERUN_CS),]$TEAM_BASERUN_CS <-
    predict(lm_team_bas_cs,test_prepared_df[is.na(test_prepared_df$TEAM_BASERUN_CS),])
```

The following is the summary of the prepared test data

```
summary(test_prepared_df)
```

```
##  TEAM_BATTING_2B TEAM_BATTING_3B  TEAM_BATTING_HR  TEAM_BATTING_BB
##  Min.   : 44.0   Min.   : 14.00   Min.   :  0.00   Min.   : 15.0  
##  1st Qu.:210.0   1st Qu.: 35.00   1st Qu.: 44.50   1st Qu.:436.5  
##  Median :239.0   Median : 52.00   Median :101.00   Median :509.0  
##  Mean   :241.3   Mean   : 55.91   Mean   : 95.63   Mean   :499.0  
##  3rd Qu.:278.5   3rd Qu.: 72.00   3rd Qu.:135.50   3rd Qu.:565.5  
##  Max.   :376.0   Max.   :155.00   Max.   :242.00   Max.   :792.0
##  TEAM_BATTING_SO  TEAM_BASERUN_SB TEAM_BASERUN_CS  TEAM_FIELDING_E
##  Min.   :   0.0   Min.   :  0.0   Min.   :  0.00   Min.   :  73.0
##  1st Qu.: 534.9   1st Qu.: 59.0   1st Qu.: 43.00   1st Qu.: 131.0
##  Median : 677.0   Median : 92.0   Median : 57.00   Median : 163.0
##  Mean   : 699.7   Mean   :121.8   Mean   : 69.77   Mean   : 249.7
##  3rd Qu.: 904.5   3rd Qu.:149.0   3rd Qu.: 88.12   3rd Qu.: 252.0
##  Max.   :1268.0   Max.   :580.0   Max.   :240.08   Max.   :1568.0
##  TEAM_FIELDING_DP TEAM_FIELDING_E_Transformed TEAM_BATTING_1B
##  Min.   : 69.0    Min.   :0.0006378           Min.   : 657.0
##  1st Qu.:131.0    1st Qu.:0.0039685           1st Qu.: 990.5
##  Median :146.0    Median :0.0061350           Median :1059.0
##  Mean   :144.7    Mean   :0.0057873           Mean   :1076.5
##  3rd Qu.:162.0    3rd Qu.:0.0076336           3rd Qu.:1134.0
##  Max.   :204.0    Max.   :0.0136986           Max.   :1846.0
```

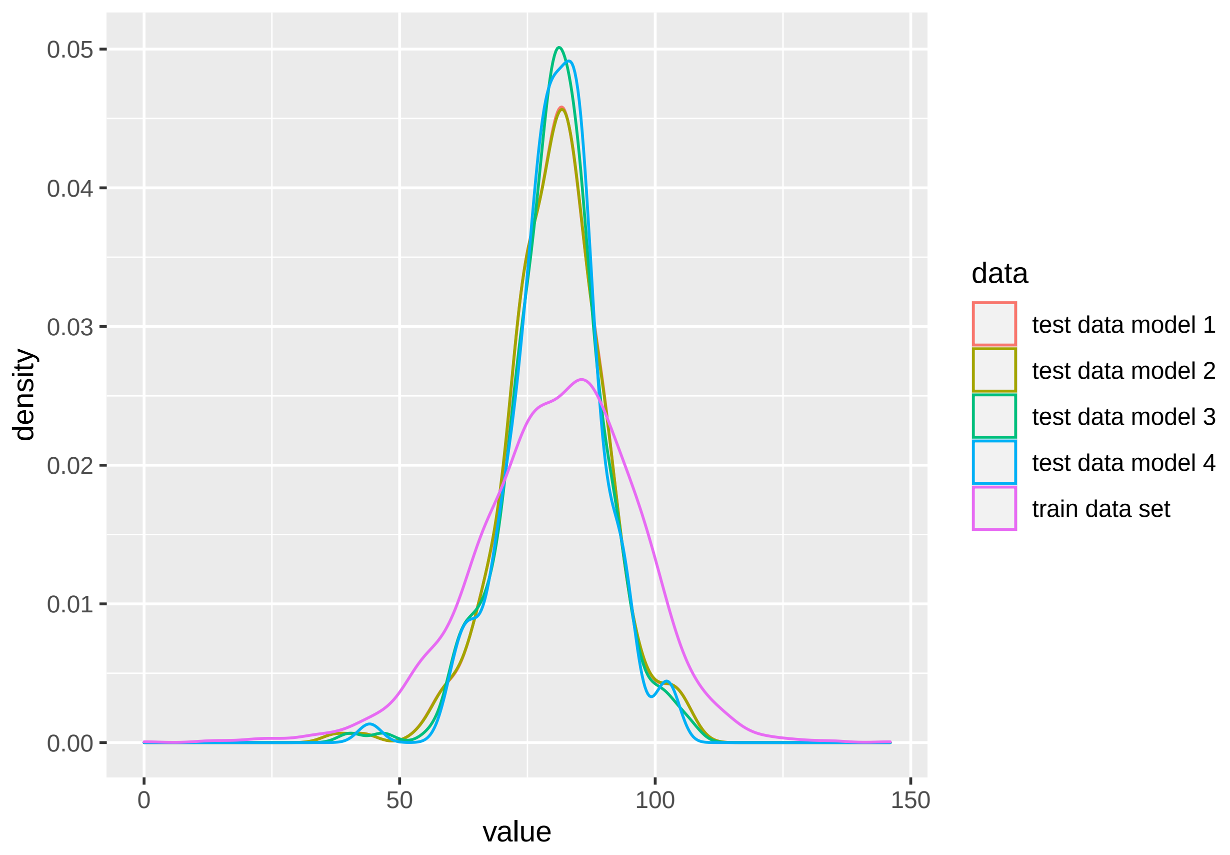
The follow plot shows the distribution of the actual Target_Win value from the train data set and the distributions of the predicted values from the 4 models

```
m1_predict <- predict(lm_win_full, test_prepared_df)
m2_predict <- predict(lm_win_eff_adj, test_prepared_df)
m3_predict <- predict(lm_win_eff_adj2, test_prepared_df)
m4_predict <- predict(lm_win_exc_mis, test_prepared_df)
```

```
dist_df <- data.frame(rbind(
      cbind(train_prepared_df$TARGET_WINS,"train data set"),
      cbind(m1_predict,"test data model 1"),
      cbind(m2_predict,"test data model 2"),
      cbind(m3_predict,"test data model 3"),
      cbind(m4_predict,"test data model 4")
      ),stringsAsFactors=FALSE)
colnames(dist_df) <- c("value","data")
dist_df$value <- as.numeric(dist_df$value)
```

```
library(ggplot2)

ggplot(dist_df, aes(x=value, color=data)) +
  geom_density()
```



The distribution of the predicted values for all 4 models are similar, with the same mean and variance. The means are also close to the mean of the Target_Win from our train data set. The variances are different because the test data has smaller sample size. All conclude that all 4 models produce plausible prediction values.

Based on above findings, the optimal model is

```
summary(lm_win_eff_adj)
```

```
##
## Call:
## lm(formula = TARGET_WINS ~ . - TEAM_BASERUN_CS, data = train_prepared_df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -68.884  -8.330   0.106   8.252  66.698
##
## Coefficients:
##                                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)                    1.315e+01  5.068e+00   2.595 0.009519 **
## TEAM_BATTING_2B                2.029e-02  7.360e-03   2.757 0.005876 **
## TEAM_BATTING_3B                1.463e-01  1.637e-02   8.938  < 2e-16 ***
## TEAM_BATTING_HR                8.526e-02  9.083e-03   9.386  < 2e-16 ***
## TEAM_BATTING_BB                2.184e-02  2.941e-03   7.428 1.55e-13 ***
## TEAM_BATTING_SO               -9.211e-03  2.374e-03  -3.879 0.000108 ***
## TEAM_BASERUN_SB                1.937e-02  4.353e-03   4.451 8.97e-06 ***
## TEAM_FIELDING_DP            -1.373e-01  1.365e-02 -10.056  < 2e-16 ***
## TEAM_FIELDING_E_Transformed  2.357e+03  2.091e+02  11.274  < 2e-16 ***
## TEAM_BATTING_1B              4.225e-02  3.672e-03  11.505  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.1 on 2266 degrees of freedom
## Multiple R-squared:  0.3109, Adjusted R-squared:  0.3081
## F-statistic: 113.6 on 9 and 2266 DF,  p-value: < 2.2e-16
```