Homework 1 (Group 5)

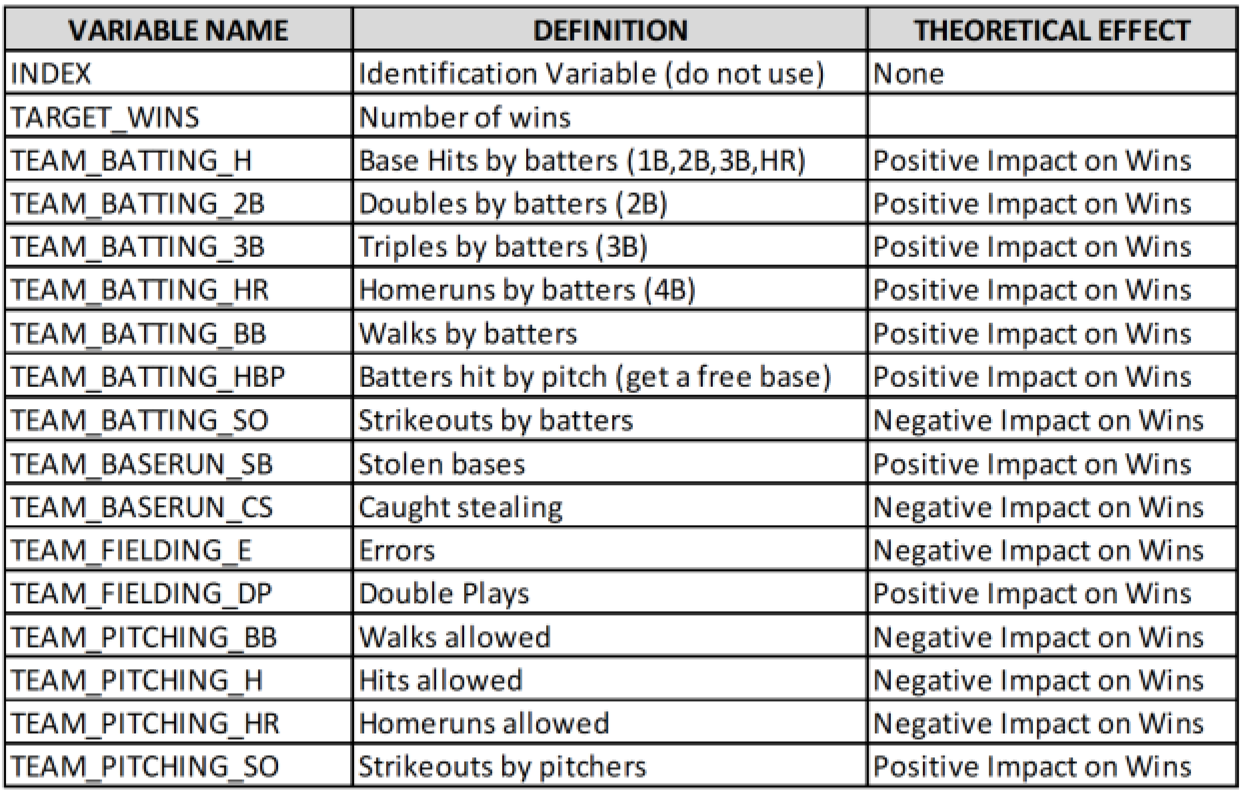
Moneyball Linear Regression

Maria A Ginorio

2/20/2022

Contents

Dataset



VARIABLE NAME	DEFINITION	THEORETICAL EFFECT
INDEX	Identification Variable (do not use)	None
TARGET_WINS	Number of wins	
TEAM_BATTING_H	Base Hits by batters (1B,2B,3B,HR)	Positive Impact on Wins
TEAM_BATTING_2B	Doubles by batters (2B)	Positive Impact on Wins
TEAM_BATTING_3B	Triples by batters (3B)	Positive Impact on Wins
TEAM_BATTING_HR	Homeruns by batters (4B)	Positive Impact on Wins
TEAM_BATTING_BB	Walks by batters	Positive Impact on Wins
TEAM_BATTING_HBP	Batters hit by pitch (get a free base)	Positive Impact on Wins
TEAM_BATTING_SO	Strikeouts by batters	Negative Impact on Wins
TEAM_BASERUN_SB	Stolen bases	Positive Impact on Wins
TEAM_BASERUN_CS	Caught stealing	Negative Impact on Wins
TEAM_FIELDING_E	Errors	Negative Impact on Wins
TEAM_FIELDING_DP	Double Plays	Positive Impact on Wins
TEAM_PITCHING_BB	Walks allowed	Negative Impact on Wins
TEAM_PITCHING_H	Hits allowed	Negative Impact on Wins
TEAM_PITCHING_HR	Homeruns allowed	Negative Impact on Wins
TEAM_PITCHING_SO	Strikeouts by pitchers	Positive Impact on Wins

Overview

In many sports scoring more points than you give up is the recipe for success. That is certainly the recipe for baseball where a winning team will typically have greater runs scored over a season than runs that they allow. We will attempt to predict the number of wins on the team.
-
We will first explore the data looking for issues or challenges (i.e. missing data, outliers, possible coding errors, multicollinearlity, etc). Once we have a handle on the data, we will apply any necessary cleaning steps. Once we have a reasonable dataset to work with, we will build and evaluate three different linear models that predict seasonal wins.
-
Our dataset includes both training data and evaluation data - we will train using the main training data, then evaluate models based on how well they perform against the holdout evaluation data. Finally we will select a final model that offers the best compromise between accuracy and simplicity.
-

---

1. Data Exploration

In this report, we will explore, analyze and model a data set containing approximately 2200 records. We will describe the size and the variables in the moneyball training data set.

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

Objective

- Understand the variables provided
- Build a multiple linear regression model on the training data
- predict the number of wins for the team.

Lets first look at the raw data values by using the skim package

```
skim(ball)
```

Table 2: Data summary

Name	ball
Number of rows	2276
Number of columns	16
Column type frequency:	
numeric	16
Group variables	None

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
TARGET_WINS	0	1.00	80.79	15.75	0	71.0	82.0	92.00	146	
TEAM_BATTING_H	0	1.00	1469.27	144.59	891	1383.0	1454.0	1537.25	2554	
TEAM_BATTING_2B	0	1.00	241.25	46.80	69	208.0	238.0	273.00	458	
TEAM_BATTING_3B	0	1.00	55.25	27.94	0	34.0	47.0	72.00	223	
TEAM_BATTING_BB	0	1.00	501.56	122.67	0	451.0	512.0	580.00	878	
TEAM_BATTING_HR	0	1.00	99.61	60.55	0	42.0	102.0	147.00	264	
TEAM_BATTING_SO	102	0.96	735.61	248.53	0	548.0	750.0	930.00	1399	
TEAM_BASERUN_SB	131	0.94	124.76	87.79	0	66.0	101.0	156.00	697	
TEAM_BASERUN_CS	772	0.66	52.80	22.96	0	38.0	49.0	62.00	201	
TEAM_BATTING_HBP	2085	0.08	59.36	12.97	29	50.5	58.0	67.00	95	
TEAM_PITCHING_H	0	1.00	1779.21	1406.84	1137	1419.0	1518.0	1682.50	30132	
TEAM_PITCHING_HR	0	1.00	105.70	61.30	0	50.0	107.0	150.00	343	
TEAM_PITCHING_BB	0	1.00	553.01	166.36	0	476.0	536.5	611.00	3645	
TEAM_PITCHING_SO	102	0.96	817.73	553.09	0	615.0	813.5	968.00	19278	
TEAM_FIELDING_E	0	1.00	246.48	227.77	65	127.0	159.0	249.25	1898	
TEAM_FIELDING_DP	286	0.87	146.39	26.23	52	131.0	149.0	164.00	228	

Observe that we have several variables with missing values. `TEAM_BATTING_SO, BASERUN_SB, BASERUN_CS, PITCHING_SO, FIELDING_DP, BATTING HBP.`

Distributions

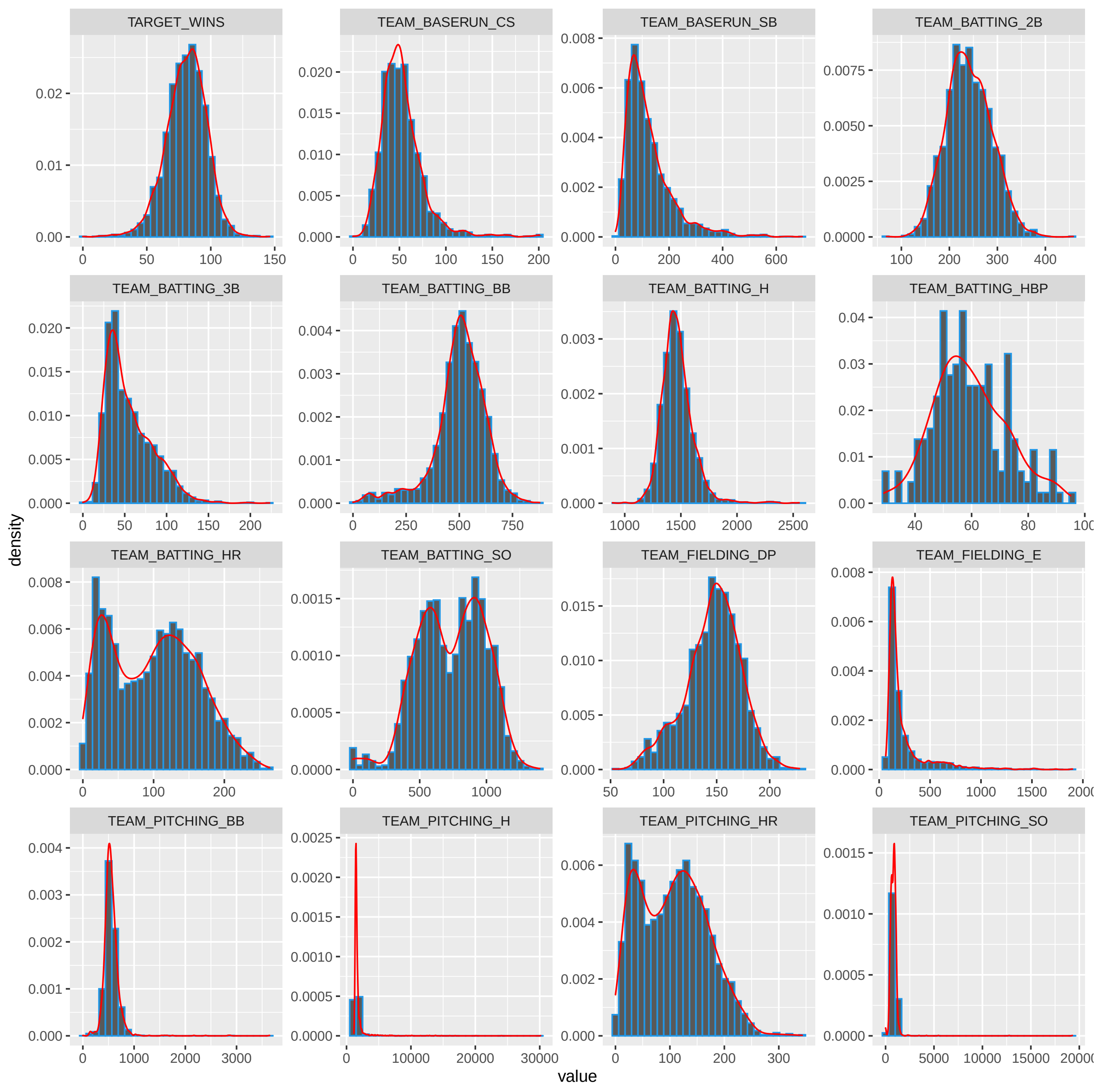
The distribution of our variables refers to how the different values are spread-out across observations. This distributions gives us a sense of what the most common values are for a given variable and how much this values vary.

The distribution of our variables can also alert us of of unusual patterns, in this case we have observed the prevalence of kurtosis for certain variables like: (right skewed) BASERUN_SB, BASERUN_CS, PITCHING_H, PITCHING_BB and PITCHING_SO.

After creating independent histograms for each variable we have found 2 variables that appear to be bi-modal. We notice that the graphs of this variables have two distinct humps or peaks with a valley separating them. We could attribute this observations to possibly different groups or the existence of NA's in the data.

Variable	Type
TEAM_BATTING_SO	BIMODAL
TEAM_PITCHING_HR	BIMODAL (no NA's)

We have no data linking rows with specific years. We might attempt to locate additional data sets that link performance to year and leverage change point detection to see if bimodal relationships are linked with all teams at specific points in time. If change points are present, that suggests something changed affecting all teams (e.g. new rules that improved or lowered the bimodal feature). We would then create a new categorical feature indicating which class the row data came from, before or after the change point.


TARGET_WINS
TEAM_BASERUN_CS
TEAM_BASERUN_SB
TEAM_BATTING_2B
TEAM_BATTING_3B
TEAM_BATTING_BB
TEAM_BATTING_H
TEAM_BATTING_HBP
TEAM_BATTING_HR
TEAM_BATTING_SO
TEAM_FIELDING_DP
TEAM_FIELDING_E
TEAM_PITCHING_BB
TEAM_PITCHING_H
TEAM_PITCHING_HR
TEAM_PITCHING_SO
density
value

Outliers

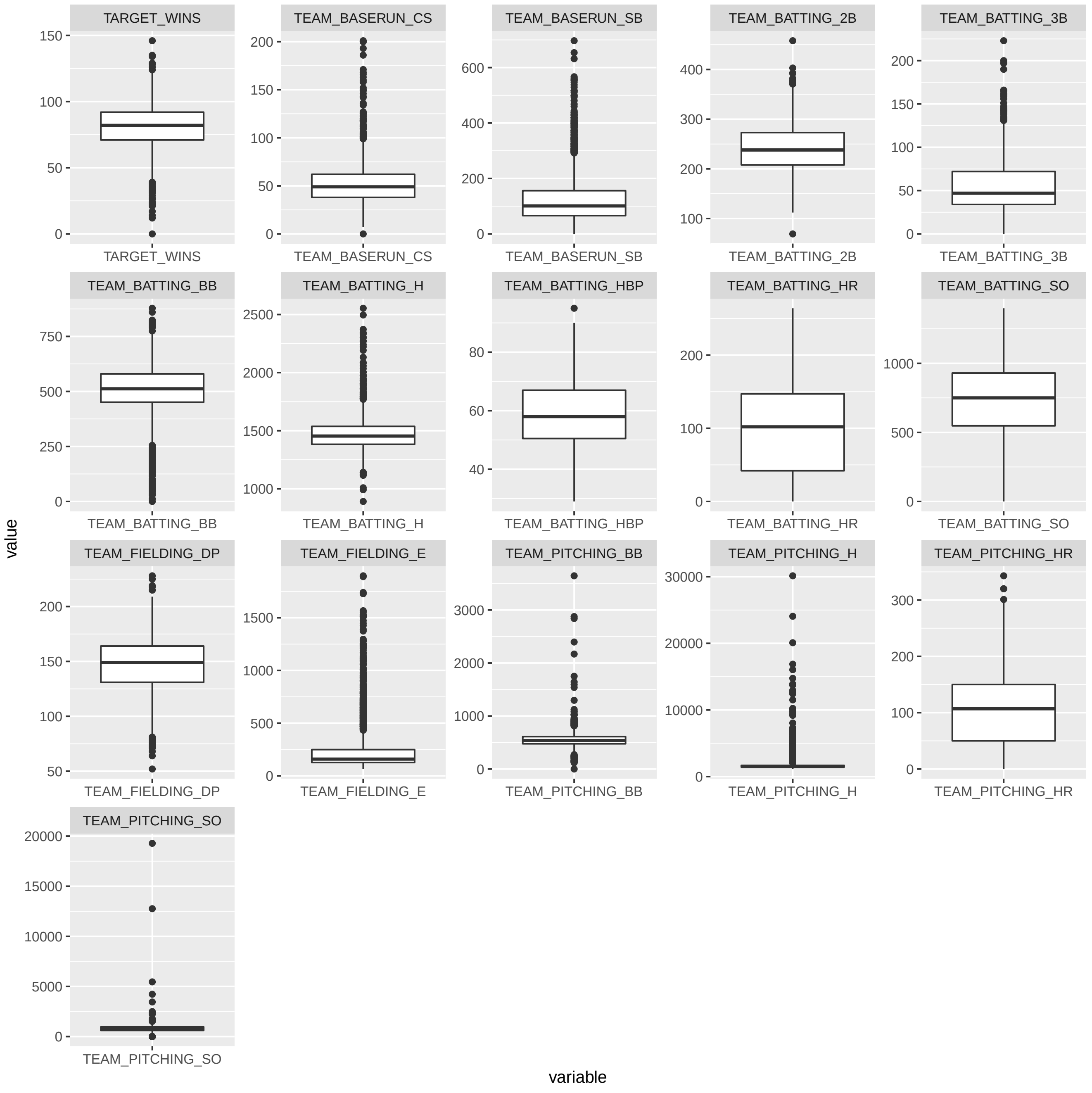
In addition to histogram graph of our variable we thought it was pertinent to take a look at our variables using a boxplot. It will help us quickly visualize the distribution of the values in the dataset and see where the five number summary values are located.

In addition, we will be able to create a clear picture of the median values and the spreads across all the distributions. One of the most important observation we will obtain from this graph however, is outlier detection.

Before we start fitting our data to the model we will have to determine the validity of the data-points corresponding to bad leverage points (outliers). We can observe that variables like TEAM_BATTING_3B, TEAM_PITCHING_BB, TEAM_PITCHING_H, TEAM_PITCHING_SO have outliers. Are these data points unusual or different in some way from the rest of the data? We will have to consider removing this and refit the data if we consider they could be affecting our results.

We shall follow the common practice of labeling points as outliers in small- to moderate-size data sets if the standardized residual for the point falls outside the interval from –2 to 2.

In summary, an outlier is a point whose standardized residual falls outside the interval from –2 to 2. Recall that a bad leverage point is a leverage point which is also an outlier. Thus, a bad leverage point is a leverage point whose standardized residual falls outside the interval from –2 to 2. On the other hand, a good leverage point is a leverage point whose standardized residual falls inside the interval from –2 to 2 .


TARGET_WINS
TEAM_BASERUN_CS
TEAM_BASERUN_SB
TEAM_BATTING_2B
TEAM_BATTING_3B
TEAM_BATTING_BB
TEAM_BATTING_H
TEAM_BATTING_HBP
TEAM_BATTING_HR
TEAM_BATTING_SO
TEAM_FIELDING_DP
TEAM_FIELDING_E
TEAM_PITCHING_BB
TEAM_PITCHING_H
TEAM_PITCHING_HR
TEAM_PITCHING_SO
value
variable

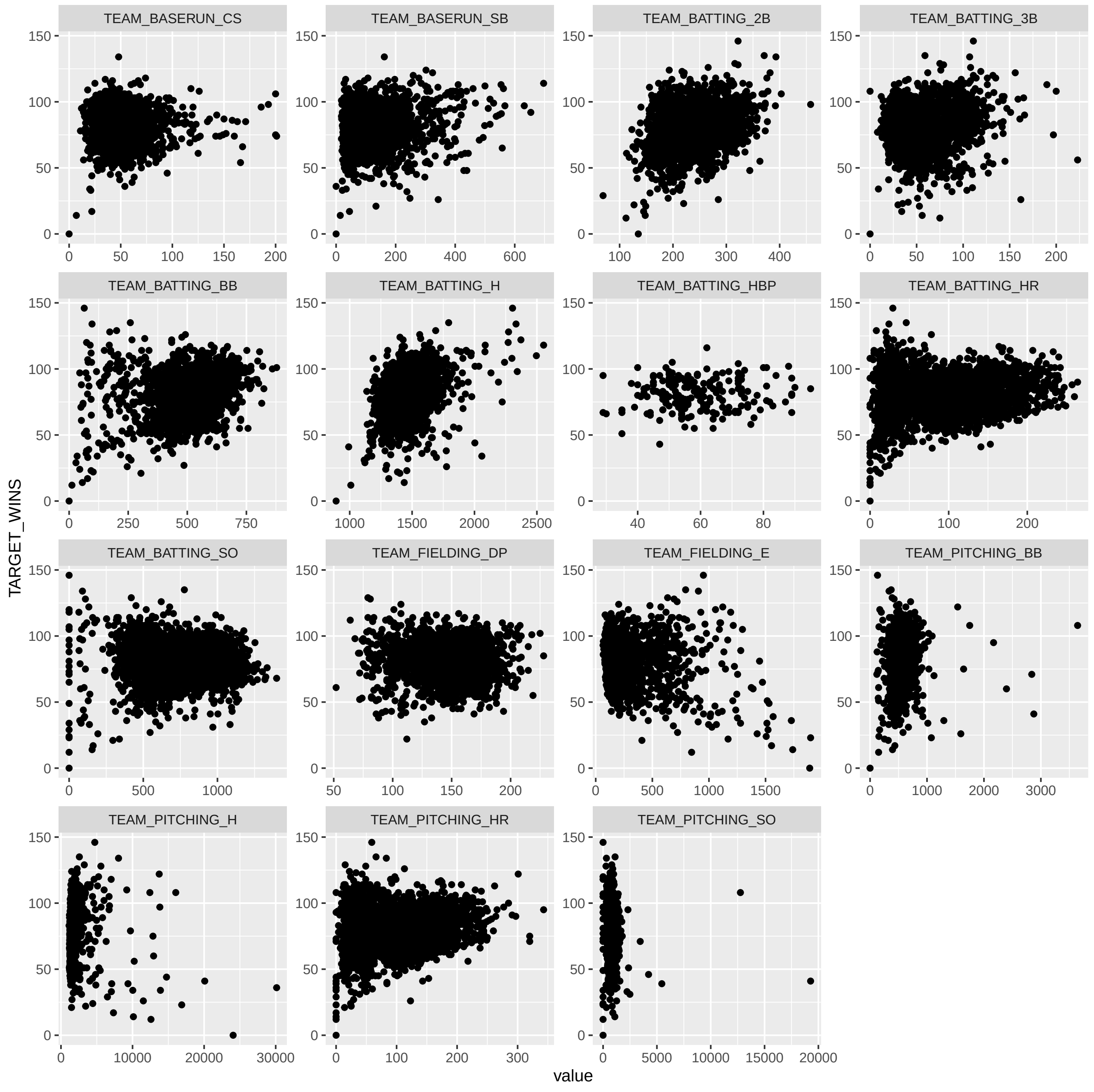
Relationships

We want use scatter plots in each variable versus the target variable, TARGET_WINS to get an idea of the relationship between them. Consider we have not dealt with missing data yet.

The plots indicate interesting relationship between the variable TARGET_WINS, however mostly they indicate issues with our data.

Most of the predictors variables are skewed and not normally distributed, in addition we have several variables with significant portions of the data missing or NA.

The pitching variables include many 0's for instance there are multiple teams with 0 strikeouts by their pitchers over the season which is very unlikely. On the other side of the spectrum for the pitching variables, we find the PITCHING_H with 20k strikeouts. That would be an average of 160 strikeouts per game which is impossible.

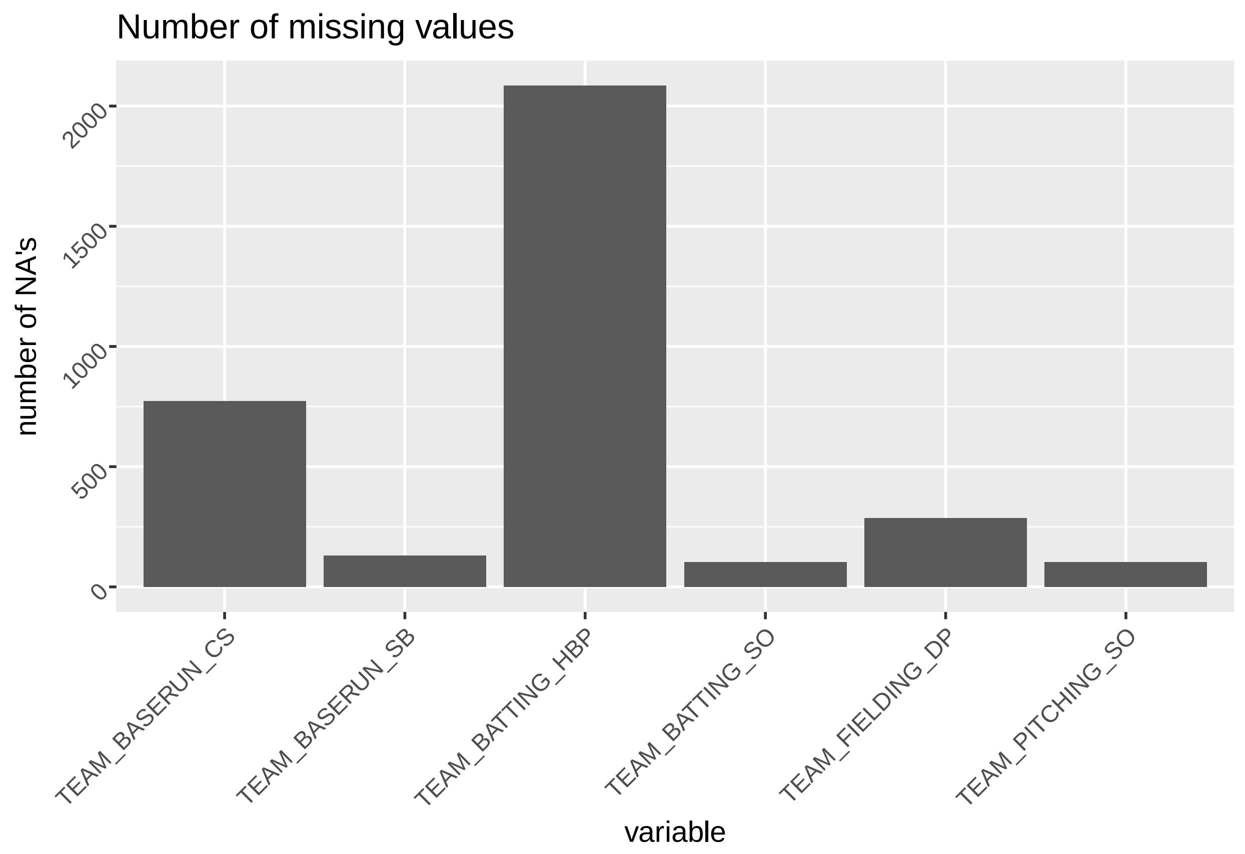


Missing Data

When we initially displayed the data summary we were able to observe the evidence of missing data. Now we take a closer look and we can see that there are several variables with high percentage of missing values.

Notice that TEAM_BATTING_HBP has 91.6% of its data missing, making this variable unusable for analysis. We will proceed and eliminate this variable. The next high percentage of missing data is in TEAM_BASERUN_CS(34%), according to baseball history, stolen bases were not tracked officially until 1887. We could assume that some of the missing data is accounted from 1871-1886.

For the rest of the variables with low percentages of missing values we will proceed to impute the missing values by replacing the missing value of a predictor with the average value of the predictor.



```
knitr::kable(ball_na)
```

variable	total	isna	num.isna	pct
TARGET_WINS	2276	FALSE	2276	100.000000
TEAM_BASERUN_CS	2276	FALSE	1504	66.080844
TEAM_BASERUN_CS	2276	TRUE	772	33.919156
TEAM_BASERUN_SB	2276	FALSE	2145	94.244288
TEAM_BASERUN_SB	2276	TRUE	131	5.755712
TEAM_BATTING_2B	2276	FALSE	2276	100.000000
TEAM_BATTING_3B	2276	FALSE	2276	100.000000
TEAM_BATTING_BB	2276	FALSE	2276	100.000000
TEAM_BATTING_H	2276	FALSE	2276	100.000000
TEAM_BATTING_HBP	2276	FALSE	191	8.391916
TEAM_BATTING_HBP	2276	TRUE	2085	91.608084
TEAM_BATTING_HR	2276	FALSE	2276	100.000000

variable	total	isna	num.isna	pct
TEAM_BATTING_SO	2276	FALSE	2174	95.518453
TEAM_BATTING_SO	2276	TRUE	102	4.481547
TEAM_FIELDING_DP	2276	FALSE	1990	87.434095
TEAM_FIELDING_DP	2276	TRUE	286	12.565905
TEAM_FIELDING_E	2276	FALSE	2276	100.000000
TEAM_PITCHING_BB	2276	FALSE	2276	100.000000
TEAM_PITCHING_H	2276	FALSE	2276	100.000000
TEAM_PITCHING_HR	2276	FALSE	2276	100.000000
TEAM_PITCHING_SO	2276	FALSE	2174	95.518453
TEAM_PITCHING_SO	2276	TRUE	102	4.481547

2. Data Preparation

Data Preparation	Action Taken
Rename Variables	Drop the word TEAM for readability
Remove Variables	Remove BATTING_HBP since it was missing 90% of its data
Missing Data	Data Imputation - median values
Data Normalization	Standardization (Z-score normalization)

Mean Imputation

One of the simplest imputation methods is the mean imputation. It boils down to simply replacing the missing values for each variable with the mean of its observed values. Mean imputation can work well for time-series data that randomly fluctuate around some long-term average, such as stock price changes. However, some practitioners treat mean imputation as a default go-to method also for cross-sectional data, forgetting that it is often a very poor choice. Mean imputation has two major drawbacks: it destroys the relations between the variables and provides no variance in the imputed data.

A final remark: instead of the mean, one might impute with a median or a mode. Median imputation could be a better choice when there are outliers in the data. In such cases, the mean can be driven arbitrarily large or small with even a single outlier, while the median will stay closer to most of the data.

```
#Create Binary indicators for whether each value was originally missing
ball_imp <- ball_clean %>%
  mutate(BASERUN_CS_imp = ifelse(is.na(BASERUN_CS), T, F)) %>%
  mutate(BASERUN_SB_imp = ifelse(is.na(BASERUN_SB), T, F)) %>%
  mutate(BATTING_SO_imp = ifelse(is.na(BATTING_SO), T, F)) %>%
  mutate(FIELDING_DP_imp = ifelse(is.na(FIELDING_DP), T, F)) %>%
  mutate(PITCHING_SO_imp = ifelse(is.na(PITCHING_SO), T, F))
```

```
# Replace mising values in all 5 variables with their respective median
ball_imp <- ball_imp %>%
  mutate(BASERUN_CS = ifelse(is.na(BASERUN_CS),
                             median(BASERUN_CS, na.rm = T),
                             BASERUN_CS)) %>%
  mutate(BASERUN_SB = ifelse(is.na(BASERUN_SB),
                             median(BASERUN_SB, na.rm = T),
                             BASERUN_SB)) %>%
  mutate(BATTING_SO = ifelse(is.na(BATTING_SO),
                             median(BATTING_SO, na.rm = T),
                             BATTING_SO)) %>%
  mutate(FIELDING_DP = ifelse(is.na(FIELDING_DP),
                              median(FIELDING_DP, na.rm = T),
                              FIELDING_DP)) %>%
  mutate(PITCHING_SO = ifelse(is.na(PITCHING_SO),
                              median(PITCHING_SO, na.rm = T),
                              PITCHING_SO))
```

Median Imputed data

```
## # A tibble: 6 x 10
##   BASERUN_SB BASERUN_SB_imp BASERUN_CS BASERUN_CS_imp BATTING_SO BATTING_SO_imp
##        <dbl> <lgl>               <dbl> <lgl>               <dbl> <lgl>
## 1        101 TRUE                   49 TRUE                  842 FALSE
## 2         37 FALSE                  28 FALSE                1075 FALSE
## 3         46 FALSE                      27 FALSE                     917 FALSE
## 4         43 FALSE                      30 FALSE                     922 FALSE
## 5         49 FALSE                      39 FALSE                     920 FALSE
## 6        107 FALSE                      59 FALSE                     973 FALSE
## # ... with 4 more variables: FIELDING_DP <dbl>, FIELDING_DP_imp <lgl>,
## #   PITCHING_SO <dbl>, PITCHING_SO_imp <lgl>
```

Data Normalization

$x' = x - mu/tetha$

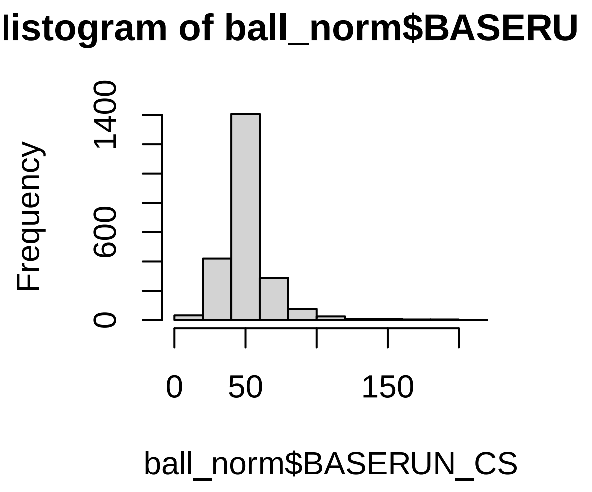
Min-max scaling is a very common way to normalize the data. It scales every feature value between its minimum and maximum. In the scaled version, the minimum value is mapped to zero, the maximum is mapped to one and the rest of the values lie in between.

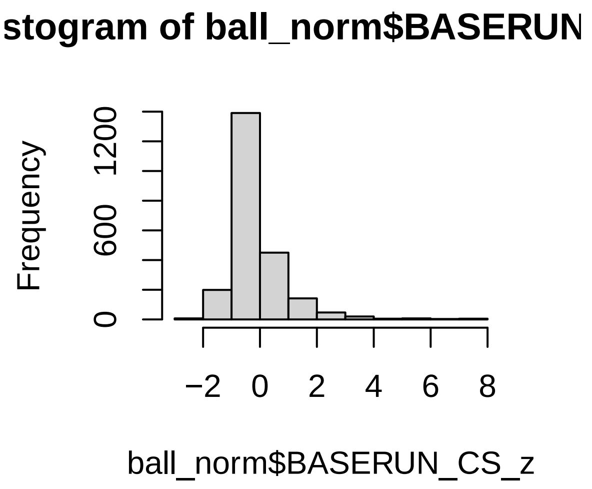
Another popular method is called standardization or Z-score normalization. It represents a numerical value as units of a standard deviation from the feature mean. As a result, values below the mean will be mapped to negative units and values above the mean will be mapped to positive units.

To summarize, min-max normalization ensures that all features will share the exact same scale but does not cope well with outliers. Z-score normalization, on the other hand, is more robust to outliers but produces normalized values in different scales.

Just to provide a brief example of the logic to be applied in all the variables that are not normal we will use the imputed variables (no NA's) to normalize them by using the z_score approach. As we can see in the example below once we use the imputed variable to create a normalized one the histogram showing its distribution has been normalized.

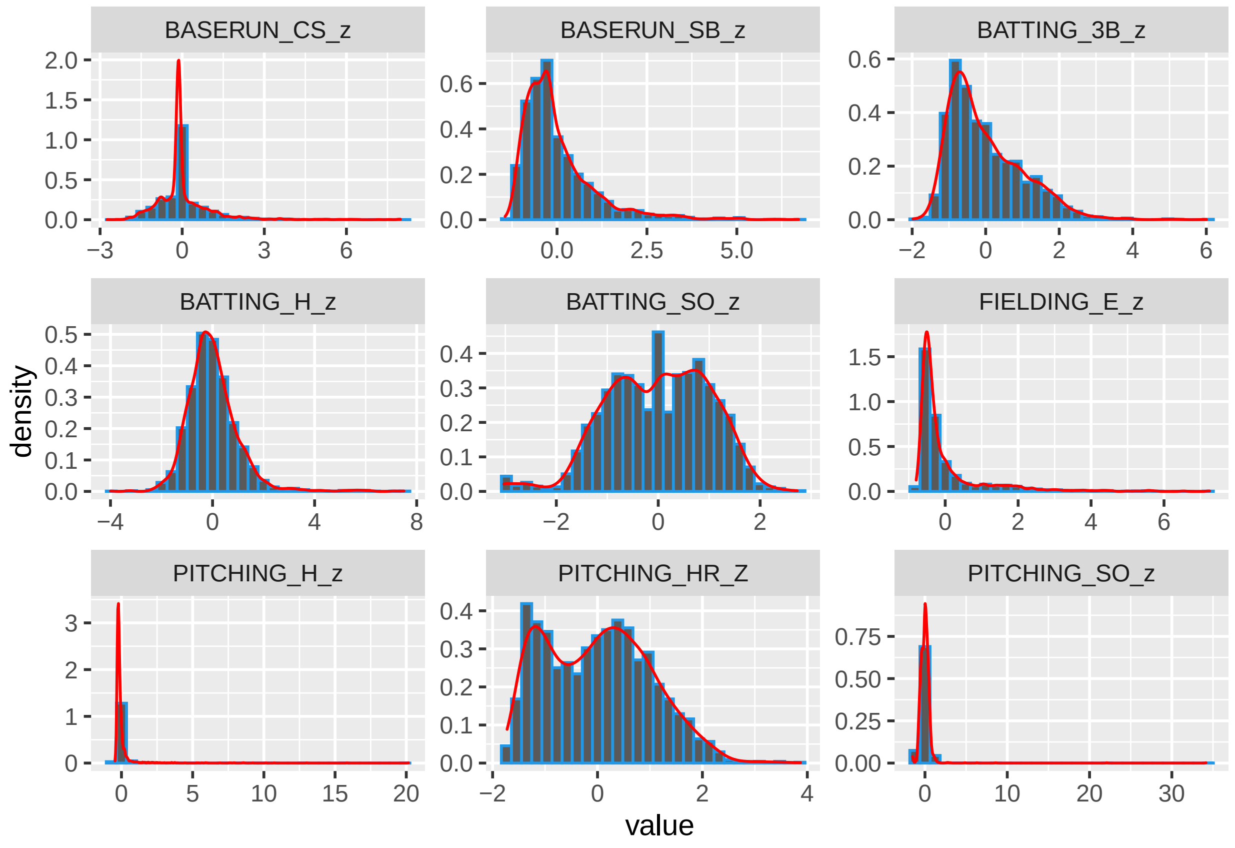
```
ball_norm <- ball_imp %>%
  mutate(BASERUN_CS_z = (
    BASERUN_CS - mean(BASERUN_CS))/sd(BASERUN_CS))
```





Normalized Distributions

```
ggplot(ball_norm_z, aes(value)) +
  geom_histogram(aes(x=value, y = ..density..),
                 colour = 4, bins = 30) +
  geom_density(aes(x=value), color = "red") +
  facet_wrap(~variable, scales = "free")
```



Clean Data

```
# Build clean dataframe with all transformations
df_clean <- ball %>%
  select(-TEAM_BATTING_HBP) %>%
  rename_with(~ sub("TEAM_", "", .x), starts_with("TEAM")) %>%
  mutate(BASERUN_CS = ifelse(is.na(BASERUN_CS),
                             median(BASERUN_CS, na.rm = T),
                             BASERUN_CS)) %>%
  mutate(BASERUN_SB = ifelse(is.na(BASERUN_SB),
                             median(BASERUN_SB, na.rm = T),
                             BASERUN_SB)) %>%
  mutate(BATTING_SO = ifelse(is.na(BATTING_SO),
                             median(BATTING_SO, na.rm = T),
                             BATTING_SO)) %>%
  mutate(FIELDING_DP = ifelse(is.na(FIELDING_DP),
                              median(FIELDING_DP, na.rm = T),
                              FIELDING_DP)) %>%
  mutate(PITCHING_SO = ifelse(is.na(PITCHING_SO),
                              median(PITCHING_SO, na.rm = T),
                              PITCHING_SO)) %>%
  mutate(BASERUN_CS = (
    BASERUN_CS - mean(BASERUN_CS))/sd(BASERUN_CS)) %>%
  mutate(BASERUN_SB = (
    BASERUN_SB - mean(BASERUN_SB)) / sd(BASERUN_SB)) %>%
  mutate(BATTING_3B = (
    BATTING_3B - mean(BATTING_3B)) / sd(BATTING_3B)) %>%
  mutate(BATTING_H = (
    BATTING_H - mean(BATTING_H)) / sd(BATTING_H)) %>%
  mutate(BATTING_SO = (
    BATTING_SO - mean(BATTING_SO)) / sd(BATTING_SO)) %>%
  mutate(FIELDING_E = (
    FIELDING_E - mean(FIELDING_E)) / sd(FIELDING_E)) %>%
  mutate(PITCHING_BB = (
    PITCHING_BB - mean(PITCHING_BB)) / sd(PITCHING_BB)) %>%
  mutate(PITCHING_H = (
    PITCHING_H - mean(PITCHING_H)) / sd(PITCHING_H)) %>%
  mutate(PITCHING_HR = (
    PITCHING_HR - mean(PITCHING_HR)) / sd(PITCHING_HR)) %>%
  mutate(PITCHING_SO = (
    PITCHING_SO - mean(PITCHING_SO)) / sd(PITCHING_SO))
```

```
skim(df_clean)
```

Table 7: Data summary

Name	df_clean
Number of rows	2276
Number of columns	15
Column type frequency:	
numeric	15

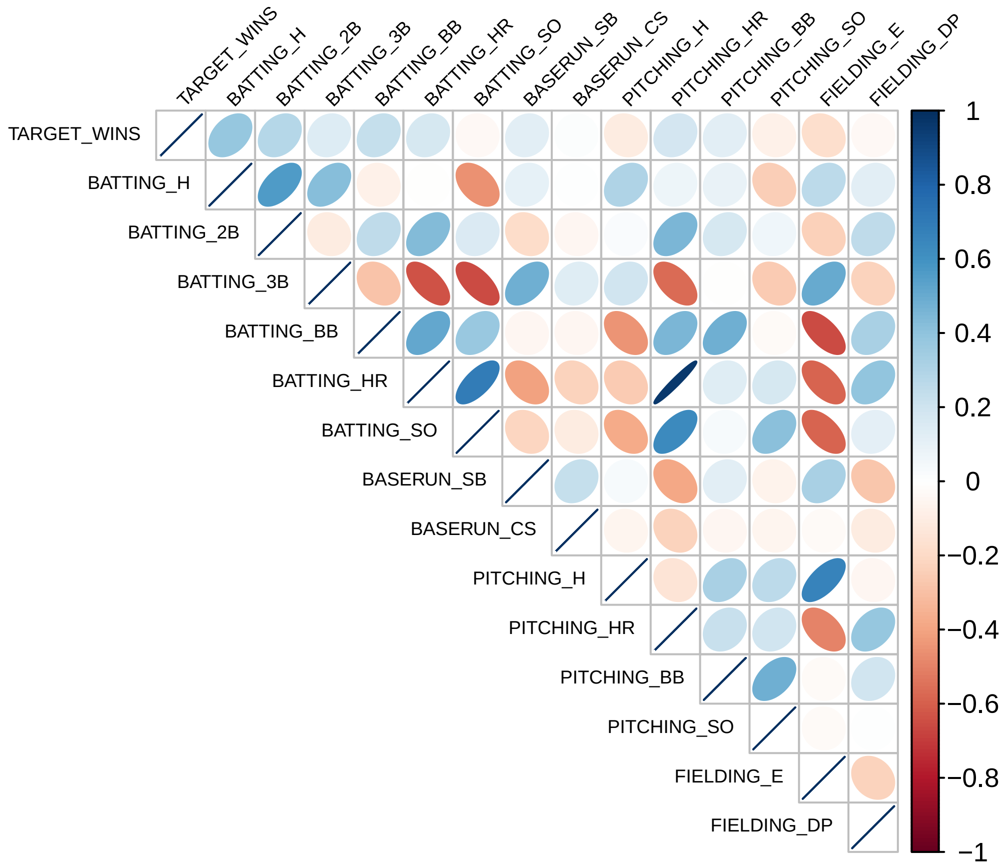
Group variables	None

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
TARGET_WINS	0	1	80.79	15.75	0.00	71.00	82.00	92.00	146.00	
BATTING_H	0	1	0.00	1.00	-4.00	-0.60	-0.11	0.47	7.50	
BATTING_2B	0	1	241.25	46.80	69.00	208.00	238.00	273.00	458.00	
BATTING_3B	0	1	0.00	1.00	-1.98	-0.76	-0.30	0.60	6.00	
BATTING_BB	0	1	501.56	122.67	0.00	451.00	512.00	580.00	878.00	
BATTING_HR	0	1	99.61	60.55	0.00	42.00	102.00	147.00	264.00	
BATTING_SO	0	1	0.00	1.00	-3.03	-0.74	0.06	0.78	2.73	
BASERUN_SB	0	1	0.00	1.00	-1.44	-0.66	-0.26	0.32	6.72	
BASERUN_CS	0	1	0.00	1.00	-2.75	-0.40	-0.13	0.15	7.97	
PITCHING_H	0	1	0.00	1.00	-0.46	-0.26	-0.19	-0.07	20.15	
PITCHING_HR	0	1	0.00	1.00	-1.72	-0.91	0.02	0.72	3.87	
PITCHING_BB	0	1	0.00	1.00	-3.32	-0.46	-0.10	0.35	18.59	
PITCHING_SO	0	1	0.00	1.00	-1.51	-0.35	-0.01	0.26	34.15	
FIELDING_E	0	1	0.00	1.00	-0.80	-0.52	-0.38	0.01	7.25	
FIELDING_DP	0	1	146.72	24.54	52.00	134.00	149.00	161.25	228.00	

Multicolinearity

One problem that can occur with multi-variable regression is correlation between variables, or multicolinearity. A quick check is to run co linearity test.



We can see that some variables are highly correlated with one another, such as PITCHING_BB and BATTING_BB. When we start considering features for our models, we'll need to account for the correlations between features and avoid including pairs with strong correlations.

Many features are also inherently associated, for example, as batter strike outs increase, we would expect a decrease in hit metrics. As Base Hits increase, we would expect to see increases in the 2B, 3B and 4B columns, etc.

Correlation

In order to determine the best predictor for our model we need to detect which are the predictor variables with low correlation value. We use the corrr package to determine all the variables with values <0.10. This will allow us to only manipulate the variables that have significance to our model.

```
clean_cor %>%
  focus(TARGET_WINS) %>%
  fashion()
```

```
##           term TARGET_WINS
## 1    BATTING_H         .39
## 2   BATTING_2B         .29
## 3   BATTING_3B         .14
## 4   BATTING_BB         .23
## 5   BATTING_HR         .18
## 6   BATTING_SO        -.03
## 7   BASERUN_SB         .12
## 8   BASERUN_CS         .02
## 9   PITCHING_H        -.11
## 10 PITCHING_HR         .19
## 11 PITCHING_BB         .12
## 12 PITCHING_SO        -.08
## 13  FIELDING_E        -.18
## 14 FIELDING_DP        -.03
```

Resolution.

We have determined that there are 6 predictor variables that we could eliminate due to their low correlated value.

These variables are:

Excluded Variables	Correlation
BATTING_SO	-0.0306
BASERUN_CS	0.0160
PITCHING_H	-0.110
PITCHING_SO	-0.0758
FIELDING_E	-0.176
FIELDING_DP	-0.0301

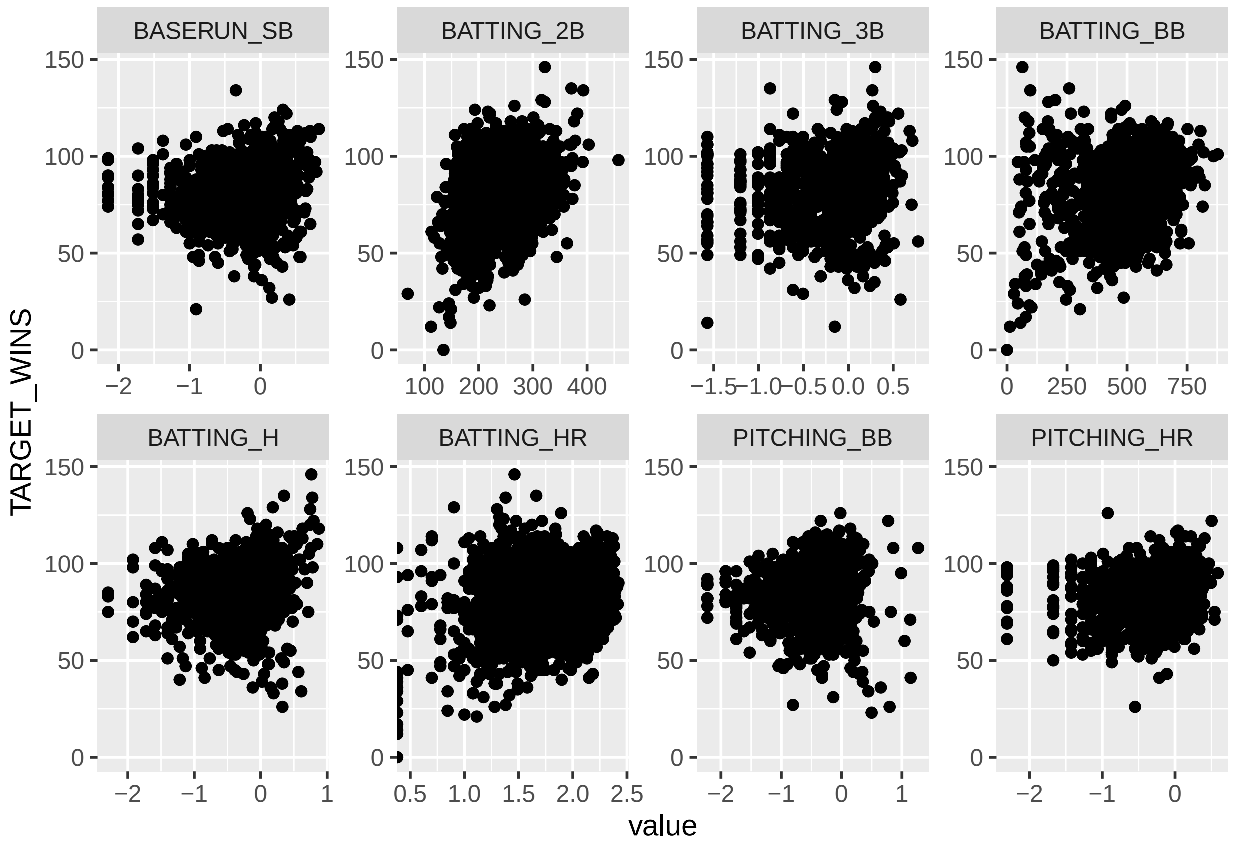
Log10 Transform

Now that we have determined Multicolinearity and Correlation we were able to determine which variables we should choose for our model.

we could look at our data composition and relationship with TARGET_WINS.

Let's transform the values in our data frame by adjusting log10 on right skwed variables; this is due to all predictor variables having the same units and are basically counts.

```
## Warning in mask$eval_all_mutate(quo): NaNs produced

## Warning in mask$eval_all_mutate(quo): NaNs produced

## Warning in mask$eval_all_mutate(quo): NaNs produced

## Warning in mask$eval_all_mutate(quo): NaNs produced

## Warning in mask$eval_all_mutate(quo): NaNs produced

## Warning: Removed 6479 rows containing missing values (geom_point).
```



from the graphs above it seems that we still have to adjust some predictor variables because they have excessive outliers. Once we create our model utilizing the final clean data we will utilize the box cox method on our linear model to adjust.

3. Building Models

Using the training data set, build at least three different multiple linear regression models, using different variables (or the same variables with different transformations). Since we have not yet covered automated variable selection methods, you should select the variables manually (unless you previously learned Forward or Stepwise selection, etc.). Since you manually selected a variable for inclusion into the model or exclusion into the model, indicate why this was done. Discuss the coefficients in the models, do they make sense? For example, if a team hits a lot of Home Runs, it would be reasonably expected that such a team would win more games. However, if the coefficient is negative (suggesting that the team would lose more games), then thatneeds to be discussed. Are you keeping the model even though it is counter intuitive? Why? The boss needs to know.

Model # 1

For the first approach we will utilize dataset provided ball_train with the established relevant variables. This model utilizes raw data , with no transformations, no normalization, no dealing with missing data (imputation).

We use this model for comparison purposes as its results will serve as comparison with the refined dataset we have created.

- The initial regression model is

$$T_{wins} = 1.84 + 0.035 BAT_H - 0.01 BAT_{2B} + 0.06 BAT_{3B} - 0.003 BAT_{HR} + 0.05 BAT_{BB} + 0.031 BRUN_{SB} + 0.06 PIT_{HR} - 0.02 PIT_{BB}$$

- The variable BATTING_H has the largest effect on Target wins since its regression coefficient is the largest

```
##
## Call:
## lm(formula = TARGET_WINS ~ TEAM_BATTING_H + TEAM_BATTING_2B +
##     TEAM_BATTING_3B + TEAM_BATTING_HR + TEAM_BATTING_BB + TEAM_BASERUN_SB +
##     TEAM_PITCHING_HR + TEAM_PITCHING_BB, data = ball_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -46.857  -8.545   0.595   8.587  43.901
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)       1.841165   3.658130   0.503 0.614800
## TEAM_BATTING_H    0.035619   0.003315  10.744  < 2e-16 ***
## TEAM_BATTING_2B  -0.018700   0.009115  -2.052 0.040328 *
## TEAM_BATTING_3B   0.063493   0.017860   3.555 0.000386 ***
## TEAM_BATTING_HR  -0.003001   0.029390  -0.102 0.918670
## TEAM_BATTING_BB   0.056370   0.004791  11.765  < 2e-16 ***
## TEAM_BASERUN_SB   0.031290   0.003833   8.163 5.54e-16 ***
## TEAM_PITCHING_HR  0.060429   0.027161   2.225 0.026199 *
## TEAM_PITCHING_BB -0.020742   0.003601  -5.761 9.59e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12.72 on 2136 degrees of freedom
##   (131 observations deleted due to missingness)
## Multiple R-squared:  0.2611, Adjusted R-squared:  0.2583
## F-statistic: 94.33 on 8 and 2136 DF,  p-value: < 2.2e-16
```

```
confint(model_1)
```

```
##                         2.5 %        97.5 %
## (Intercept)      -5.332702721  9.0150333065
## TEAM_BATTING_H    0.029117774  0.0421207543
## TEAM_BATTING_2B  -0.036574263 -0.0008252804
## TEAM_BATTING_3B   0.028468513  0.0985173217
## TEAM_BATTING_HR  -0.060638161  0.0546353979
## TEAM_BATTING_BB   0.046973754  0.0657659288
## TEAM_BASERUN_SB   0.023772390  0.0388071037
## TEAM_PITCHING_HR  0.007163132  0.1136939650
## TEAM_PITCHING_BB -0.027803499 -0.0136810906
```

- Coefficients: We are able to see from this model that there are 7 predictors that are statistically significant in the calculation prediction of the target variable.
- The estimates `(Estimate)` column gives you the change in y (TARGET_WINS) caused by each element of the regression, in this case BATTING_2B, BATTING_HR, and PTICHING_BB hold negative values.

Model #2

For Model 2 we will use our final clean data set with all the transformations performed. Only this time we will use R sample to create a sample of our data.

- The initial regression model is

$$T_{wins} = 60.10 + 5.53 BAT_H - 0.002 BAT_{2B} + 2.07 BAT_{3B} + 0.07 BAT_{HR} + 0.02 BAT_{BB} + 1.94 BRUN_{SB} - 1.17 PIT_{HR} - 0.81 PIT_{BB}$$

```
library(rsample)
```

```
## Warning: package 'rsample' was built under R version 4.1.2
```

```
# Splitting the dataset into train/testing set with 80/20 split
set.seed(3)
df_split <- initial_split(df_clean, prop = 0.8)
df_train <- training(df_split)
df_test <- testing(df_split)
```

```
##
## Call:
## lm(formula = TARGET_WINS ~ BATTING_H + BATTING_2B + BATTING_3B +
##     BATTING_HR + BATTING_BB + BASERUN_SB + PITCHING_HR + PITCHING_BB,
##     data = df_clean)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -60.851  -8.718   0.386   8.991  69.732
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 60.101992   2.838558  21.173  < 2e-16 ***
## BATTING_H    5.535890   0.449099  12.327  < 2e-16 ***
## BATTING_2B  -0.002639   0.009018  -0.293  0.76985
## BATTING_3B   2.074543   0.477024   4.349 1.43e-05 ***
## BATTING_HR   0.073836   0.026774   2.758  0.00587 **
## BATTING_BB   0.027854   0.003574   7.793 9.90e-15 ***
## BASERUN_SB   1.942575   0.342631   5.670 1.61e-08 ***
## PITCHING_HR -1.176937   1.479359  -0.796  0.42636
## PITCHING_BB -0.813008   0.397563  -2.045  0.04097 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.68 on 2267 degrees of freedom
## Multiple R-squared:  0.2483, Adjusted R-squared:  0.2456
## F-statistic:  93.6 on 8 and 2267 DF,  p-value: < 2.2e-16
```

```
confint(model_2)
```

```
##                   2.5 %      97.5 %
## (Intercept) 54.53554821 65.66843673
## BATTING_H    4.65520156  6.41657880
## BATTING_2B  -0.02032395  0.01504630
## BATTING_3B   1.13909339  3.00999303
## BATTING_HR   0.02133268  0.12633981
## BATTING_BB   0.02084486  0.03486345
## BASERUN_SB   1.27067131  2.61447770
## PITCHING_HR -4.07797636  1.72410192
## PITCHING_BB -1.59263250 -0.03338260
```

- Coefficients: We are able to see from this model that there are 7 predictors that are statistically significant in the calculation prediction of the target variable.
- The estimates `(Estimate)` column gives you the change in y (TARGET_WINS) caused by each element of the regression, in this case BATTING_2B, BATTING_HR, and PTICHING_BB hold negative values not that different from our previous model with no data transformations.
- Different from expected Doubles by batters has a negative impact towards winning in other words the BATTING_2B in the full model reduces the percentage of target wins by 0.002% and the same goes for PITCHING_HR (pitching home runs) by 1.17% and PITCHING_BB (walks allowed) by .0.81%

Model #3

For Model 3 we will use our final clean data set with all the transformations performed. Only this time we will use our final_clean dataset that includes log10 for the variables that were right skewed.

```
##
## Call:
## lm(formula = TARGET_WINS ~ BATTING_H + BATTING_2B + BATTING_3B +
##     BATTING_HR + BATTING_BB + BASERUN_SB + PITCHING_HR + PITCHING_BB,
##     data = final_clean)
##
## Residuals:
##       69      429      431      719     1083     1190     1340     1603
##   3.9709   3.2464  -8.4188   5.7788   5.2595 -10.5435  -6.4080  -0.2611
##     1673     1810     1828     2110     2219
##   3.9563   6.7213  -0.6495   6.2891  -8.9414
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -335.4103   133.3860  -2.515   0.0657 .
## BATTING_H     24.3560    16.3067   1.494   0.2096
## BATTING_2B    -0.2215     0.1379  -1.606   0.1835
## BATTING_3B   -11.6907    19.6385  -0.595   0.5837
## BATTING_HR   316.4236   110.1936   2.872   0.0454 *
## BATTING_BB    -0.2790     0.1418  -1.967   0.1205
## BASERUN_SB   -17.2085    11.7630  -1.463   0.2173
## PITCHING_HR  -31.1070    32.8016  -0.948   0.3967
## PITCHING_BB   97.9372    39.4702   2.481   0.0681 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11.08 on 4 degrees of freedom
##   (2263 observations deleted due to missingness)
## Multiple R-squared:  0.9264, Adjusted R-squared:  0.7793
## F-statistic: 6.295 on 8 and 4 DF,  p-value: 0.0466
```

```
confint(model_3)
```

```
##                     2.5 %      97.5 %
## (Intercept) -705.7492485  34.9287270
## BATTING_H    -20.9188000  69.6307116
## BATTING_2B    -0.6042706   0.1613214
## BATTING_3B   -66.2160232  42.8345681
## BATTING_HR    10.4770490 622.3701965
## BATTING_BB    -0.6726546   0.1147226
## BASERUN_SB   -49.8678248  15.4507642
## PITCHING_HR -122.1789530  59.9649682
## PITCHING_BB  -11.6497362 207.5240769
```

- Coefficients: We are able to see from this model that there are 3 predictors that are statistically significant in the calculation prediction of the target variable.

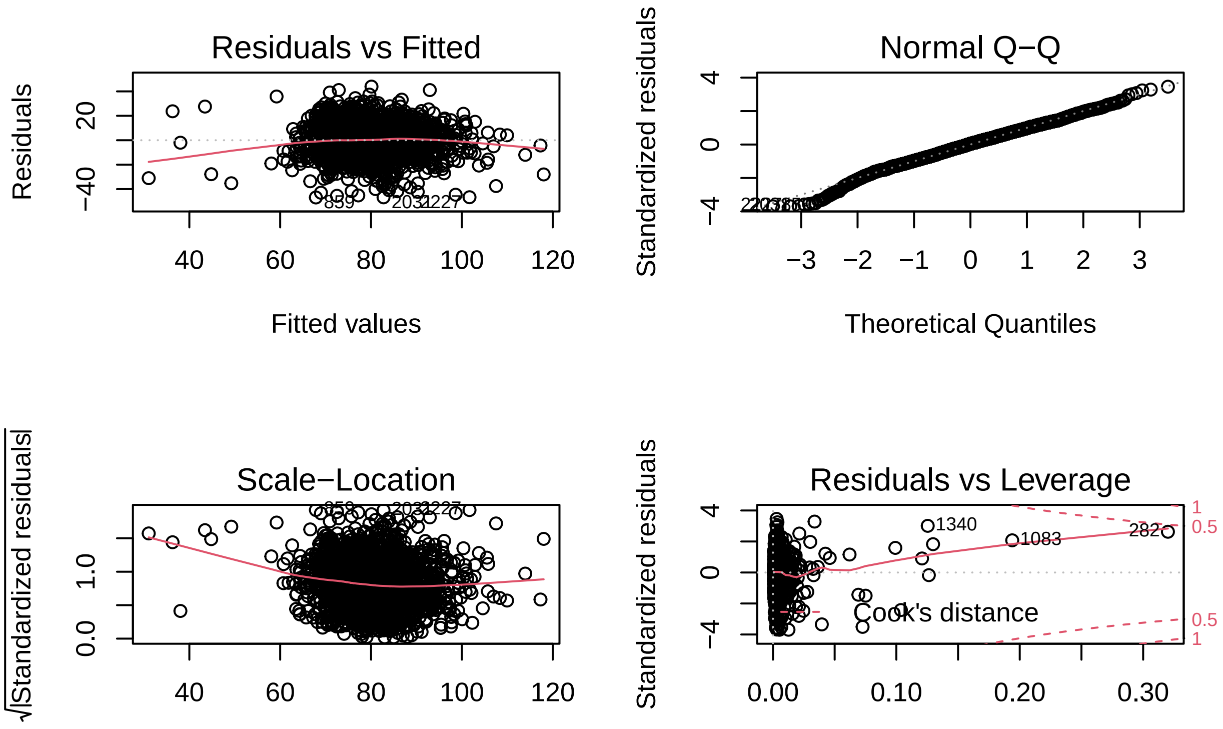
- The estimates (`Estimate`) column gives you the change in y (TARGET_WINS) caused by each element of the regression, in this case our slope intercept has turned negative and we can see more negative values.

- Negative Intercept - This means that the expected value on your dependent variable will be less than 0 when all independent/predictor variables are set to 0.
- If the **beta coefficient** is **negative**, the **interpretation** is that for every 1-unit increase in the predictor variable, the outcome variable will decrease by the **beta coefficient value**.

4. Select Models

Model_1 Testing

We should ensure that the model fitted meets this last assumption, the same variance, to maintain the model. The error term needs to be the same across all values of the independent variables.



Here we can see that the residial errors are not constant across groups and they are not distributed evenly.

Look at the `Normal Q-Q` (top right) which under homoscedasticity it should show points along the line.

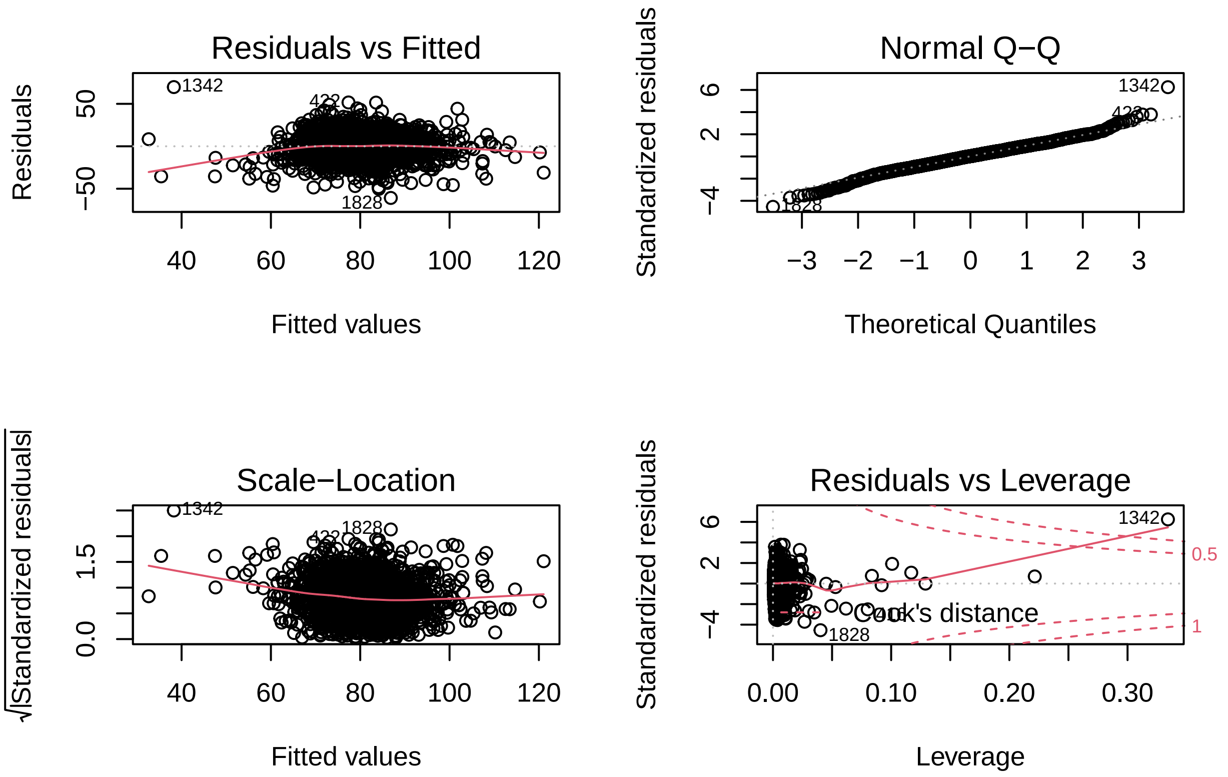
Even though it seems that the Normal Q-Q plot follows the data, we noticed the tails on both ends, this caused due to outliers most likely.

- The strength of the fit of a linear model is commonly evaluated using R^2
- It tells us what percentage of the variability in the response variable is explained by the model. The remainder of the variability is unexplained.
- R^2 also called coefficient determination. Roughly 26% of the variability in target wins can be explained by the predictors.

```
## # A tibble: 1 x 12
##   r.squared adj.r.squared sigma statistic   p.value    df logLik    AIC    BIC
##       <dbl>         <dbl> <dbl>     <dbl>     <dbl> <dbl>  <dbl>  <dbl>  <dbl>
## 1     0.261         0.258  12.7      94.3 1.71e-134     8 -8494. 17007. 17064.
## # ... with 3 more variables: deviance <dbl>, df.residual <int>, nobs <int>
```

Roughly 26% of the variability in wins of baseball games can be explained by the predictors in this model.

Model_2 Testing



```
## # A tibble: 1 x 12
##   r.squared adj.r.squared sigma statistic   p.value    df logLik    AIC    BIC
##       <dbl>         <dbl> <dbl>     <dbl>     <dbl> <dbl>  <dbl>  <dbl>  <dbl>
## 1     0.248         0.246  13.7      93.6 1.19e-134     8 -9179. 18378. 18435.
## # ... with 3 more variables: deviance <dbl>, df.residual <int>, nobs <int>
```

Here we can see that the residial errors are not constant across groups and they are not distributed evenly.

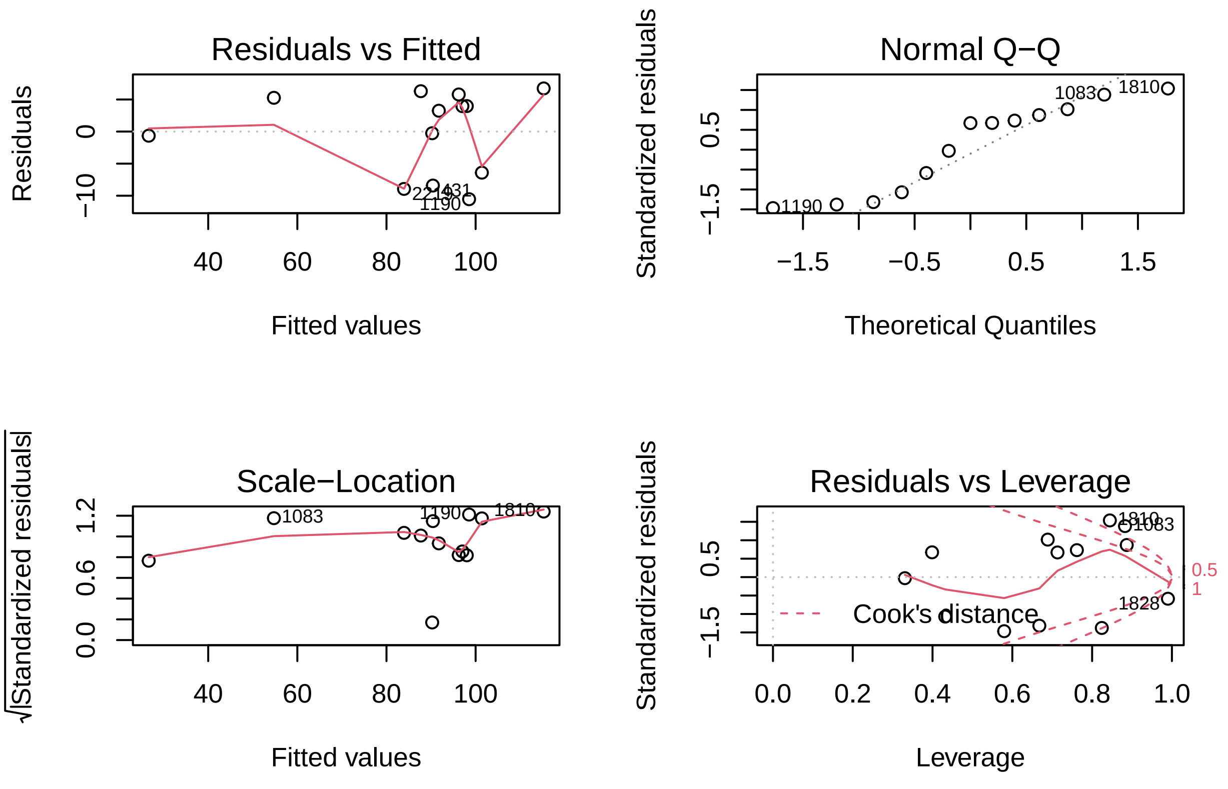
Look at the `Normal Q-Q` (top right) which under homoscedasticity it should show points along the line.

Even though it seems that the Normal Q-Q plot follows the data, we noticed the tails on both ends, this caused due to outliers most likely.

- The strength of the fit of a linear model is commonly evaluated using R^2
- It tells us what percentage of the variability in the response variable is explained by the model. The remainder of the variability is unexplained.
- R^2 also called coefficient determination. Roughly 24.8% of the variability in target wins can be explained by the predictors which is less than our first model that used the data with no transformations

Model_3 Testing

```
## Warning in sqrt(crit * p * (1 - hh)/hh): NaNs produced

## Warning in sqrt(crit * p * (1 - hh)/hh): NaNs produced
```



```
## # A tibble: 1 x 12
##   r.squared adj.r.squared sigma statistic p.value    df logLik   AIC   BIC
##       <dbl>         <dbl> <dbl>     <dbl>   <dbl> <dbl>  <dbl> <dbl> <dbl>
## 1     0.926         0.779  11.1      6.30  0.0466     8  -42.1  104.  110.
## # ... with 3 more variables: deviance <dbl>, df.residual <int>, nobs <int>
```

Here we can see that the residual errors are not constant across groups and they are not distributed evenly.

Look at the `Normal Q-Q` (top right) which under homoscedasticity it should show points along the line, it shows an improvement from the other 2 previous models. However it does not shoe the points in the along the line.

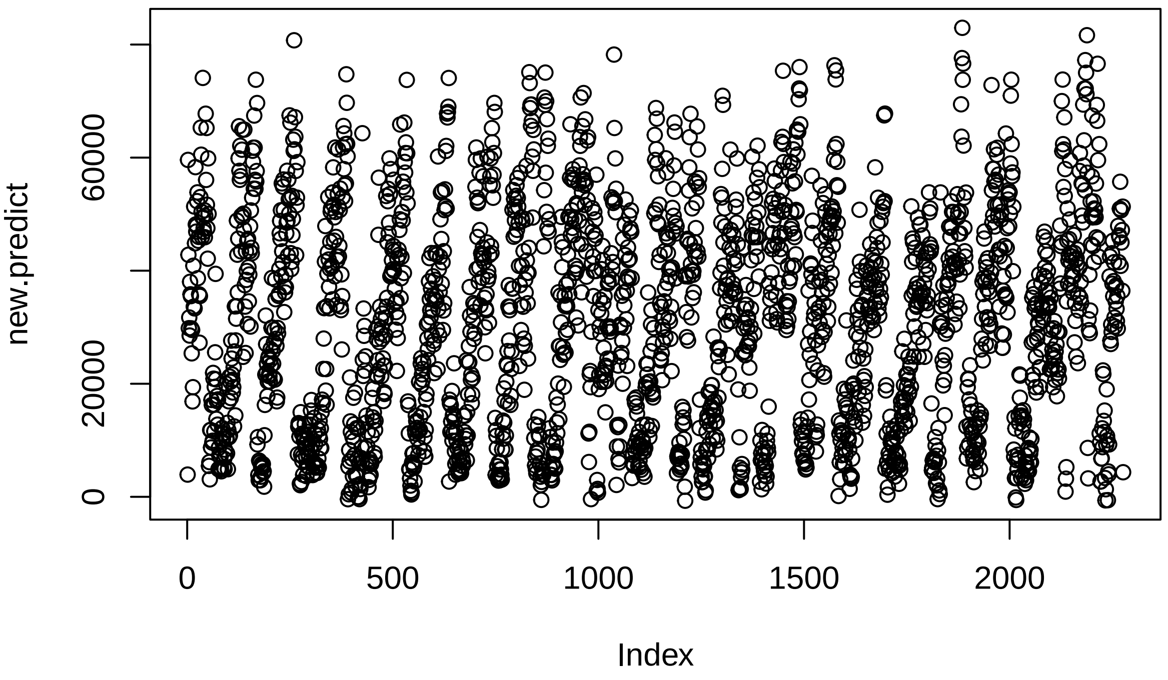
Even though it seems that the Normal Q-Q plot follows the data, we noticed the tails on both ends, this caused due to outliers most likely. We can see clear evidence of outlier 1810 and 1085

- The strength of the fit of a linear model is commonly evaluated using R^2
- It tells us what percentage of the variability in the response variable is explained by the model. The remainder of the variability is unexplained.
- R^2 also called coefficient determination. Roughly 92% of the variability in target wins can be explained by the predictors.

Final Selection

Based on these analyses, Model 3 performed marginally better than Model 1 and Model 2 , it could be selected as the linear model of choice when looking at adjusted R2. However, there is greater statistical significance under the third model relative to the others and uses less unnecessary variables to compute our prediction without sacrificing much in terms of adjusted R^2 value. Because of these factors, as well as its better adjustment for multicollinearity and log10 of skewed variables (we noticeably saw less sign-flipping in coefficients), model 3 would be the model of choice.

```
plot(new.predict)
```



References

Apendix

```
head(new.predict)
```

```
##         1         2         3         4         5         6
##  3933.649 59591.634 42816.790 29824.841 31718.912 28529.182
```