Homework 1

Sam Kuhn

11/20/22

Table of contents

Homework 1: Baseball Analysis **3**

Data Exploration: . . . 3

Load data . . . 3

Check for missing values . . . 4

Summary statistics . . . 6

Summary plots . . . 7

Correlation Plot . . . 10

Data Preparation . . . 12

Log transformation . . . 12

Build Models . . . 13

Model 1 . . . 13

Model 2 . . . 15

Model 3 . . . 18

Select Models . . . 20

Model 1 Diagnostics: . . . 20

Model 2 Diagnostics: . . . 20

Model 3 Diagnostics: . . . 20

Evaluation . . . 21

Appendix: R code . . . 23

Summary statistics . . . 23

Plots . . . 23

Transformations 30
Models 30

Homework 1: Baseball Analysis

In this homework assignment, you will explore, analyze and model a data set containing approximately 2200 records. 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.

Your objective is to build a multiple linear regression model on the training data to predict the number of wins for the team. You can only use the variables given to you (or variables that you derive from the variables provided). Below is a short description of the variables of interest in the data set

Data Exploration:

Load data

```
# library(tidyverse)
# library(here)
# library(tidymodels)
# library(corrplot)
# library(MASS)
# library(gt)

# Install pacman package and load libraries
# install.packages("pacman")
pacman::p_load(tidyverse, here, tidymodels, corrplot, MASS, gt, stargazer)

# Makes sure dplyr::filter and dplyr::select will be used
conflicted::conflict_prefer("select", "dplyr")
conflicted::conflict_prefer("filter", "dplyr")

# Load training set from data folder and clean variable names
training_set <- readr::read_csv(
  here::here("data", "moneyball-training-data (1).csv")
) |>
  janitor::clean_names()
```

Check for missing values

To check for NA values, we are going to take the sum of every value matching `NA` across the entire data-frame and print the results. Then, replace all the `NA` values with the median value of the corresponding variable. The variables with the most `NA` observations are: `team_batting_hbp`, `team_baserun_cs`, and `team_fielding_dp`.

```
# Sum NAs across columns
training_set |>
  summarise(across(everything(), ~ sum(is.na(.)))) |>
  glimpse()
```

```
Rows: 1
Columns: 17
$ index            <int> 0
$ target_wins      <int> 0
$ team_batting_h   <int> 0
$ team_batting_2b  <int> 0
$ team_batting_3b  <int> 0
$ team_batting_hr  <int> 0
$ team_batting_bb  <int> 0
$ team_batting_so  <int> 102
$ team_baserun_sb  <int> 131
$ team_baserun_cs  <int> 772
$ team_batting_hbp <int> 2085
$ team_pitching_h  <int> 0
$ team_pitching_hr <int> 0
$ team_pitching_bb <int> 0
$ team_pitching_so <int> 102
$ team_fielding_e  <int> 0
$ team_fielding_dp <int> 286
```

```r
# Replace missing values (NAs) with median values
training_set <- training_set |>
  mutate(across(everything(), ~ tidyr::replace_na(., median(., na.rm = TRUE)))) |>
  glimpse()
```

Just as a check, we will print out the data frame again to ensure no `NA` values remain.

```r
# Verify results
# Sum NAs across columns
training_set |>
  summarise(across(everything(), ~ sum(is.na(.)))) |>
  glimpse()
```

```
Rows: 1
Columns: 17
$ index            <int> 0
$ target_wins      <int> 0
$ team_batting_h   <int> 0
$ team_batting_2b  <int> 0
$ team_batting_3b  <int> 0
$ team_batting_hr  <int> 0
$ team_batting_bb  <int> 0
$ team_batting_so  <int> 0
$ team_baserun_sb  <int> 0
$ team_baserun_cs  <int> 0
$ team_batting_hbp <int> 0
$ team_pitching_h  <int> 0
$ team_pitching_hr <int> 0
$ team_pitching_bb <int> 0
$ team_pitching_so <int> 0
$ team_fielding_e  <int> 0
$ team_fielding_dp <int> 0
```

Summary statistics

Now that we do not have any missing values, we can perform some summary statistics to get a better sense of the data. Some key variables are interest are the regressand `target_wins`, where we can see the median value is slightly higher than the mean, suggesting that there is a possible left-tail distribution. Other key variables include: `team_batting_h` which can help predict total runs, and `team_batting_hr` which are homeruns.

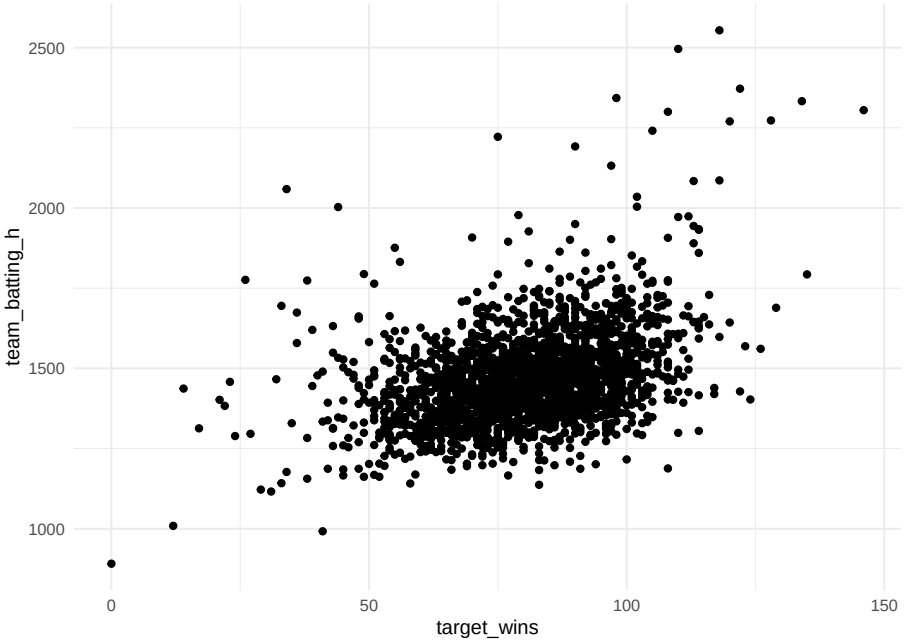
Table 1: Summary Statistics

Variable	Median	Mean	Standard Deviation
target_wins:	82	80.79	15.75215
team_batting_h	1454	1469.27	144.5912
team_batting_2b	238	241.2469	46.80141
team_batting_3b	47	55.25	27.93856
team_batting_hr	102	99.61204	60.54687
team_batting_bb	512	501.5589	122.6709
team_batting_so	750	736.2504	242.9094
team_batting_sb	101	123.3941	85.40565
team_baserun_cs	49	51.51362	18.74587
team_batting_hbp	58	58.1138	3.766219
team_pitching_h	1518	1779.21	1406.843
team_pitching_hr	107	105.6986	61.29875
team_pitching_bb	536.5	553.0079	166.3574
team_pitching_so	813.5	817.5409	540.5447
team_pitching_e	159	246.4807	227.771
team_fielding_dp	149	146.7162	24.53781

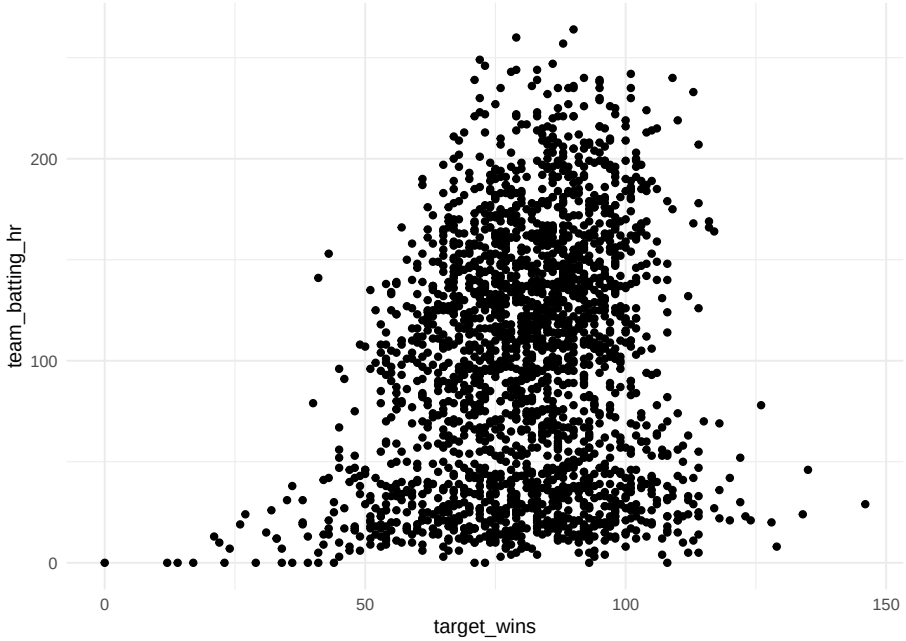
Summary plots

Let's look at some plots to visually inspect the data:

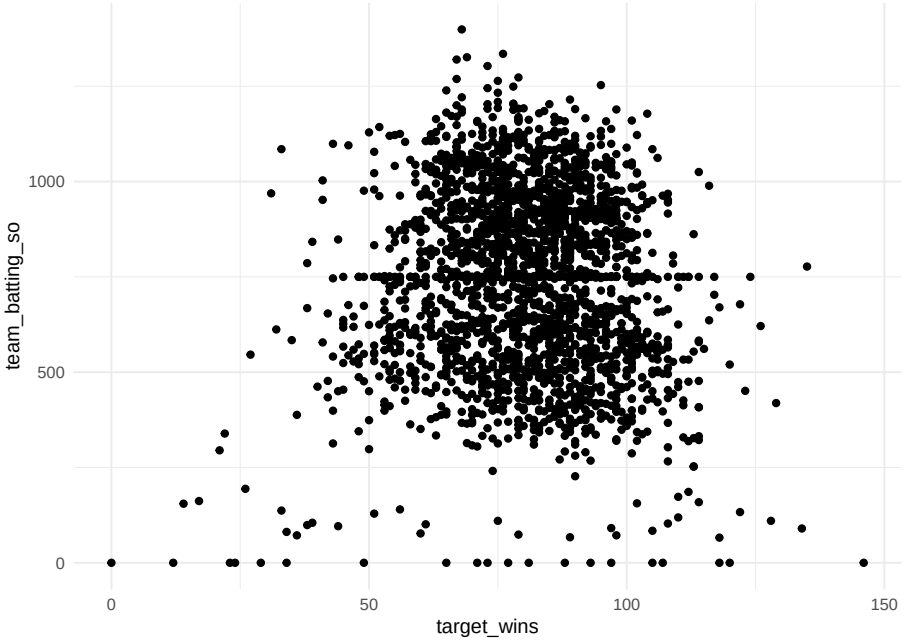
This first plot is ***base hits by batters vs. number of wins***. We can see that most of the observations are centered around 1500 hits, and ~80 wins, with a positive linear relationship.



Let's look at the relationship between ***home runs and wins*** as well. From this plot, it almost has a normal distribution, where the mean is centered around 80 wins, and with a slightly longer left-tail.

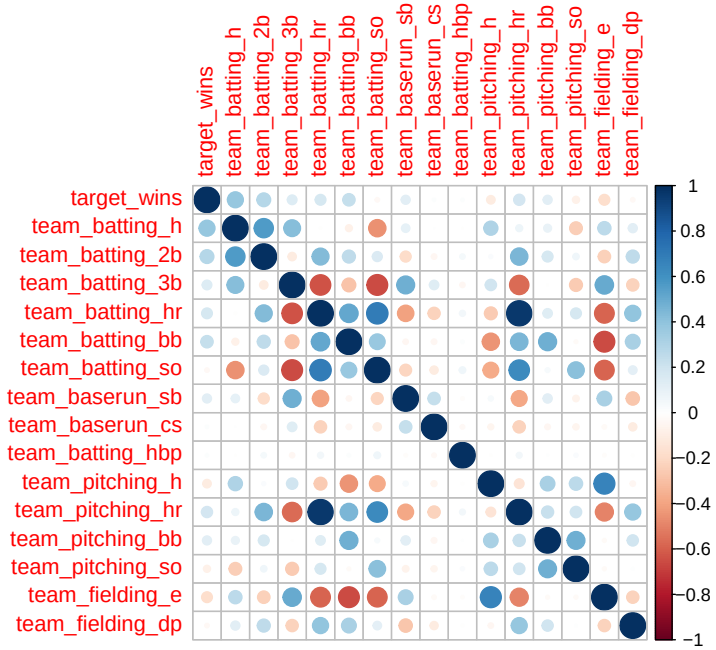


Lastly, let's look at a variable that has a negative impact on wins - ***strikeouts by batters***. This plot looks fairly similar to the plot above, however its shows a pattern that teams win more than 100 games generally don't give up more than 1,000 strikeouts a season.


team_batting_so
1000
500
0
0
50
100
150
target_wins

Correlation Plot

Now that we have a good idea about the distribution of our key variables, let's check the statistical correlation between all the variables and `target_wins`, to understand how each variable is impact it. From the table, the variable with the most positive impact is `team_batting_h`, while the most negative is `team_pitching_h`.



Correlation between variables and Target Wins

Pearson correlation

Target Wins	Variable	Correlation
1	team_batting_h	0.38876752
1	team_batting_2b	0.28910365
1	team_batting_bb	0.23255986
1	team_pitching_hr	0.18901373
1	team_batting_hr	0.17615320
1	team_batting_3b	0.14260841
1	team_pitching_bb	0.12417454
1	team_baserun_sb	0.12361087
1	team_batting_hbp	0.01651641
1	team_baserun_cs	0.01595982
1	team_fielding_dp	-0.03008630
1	team_batting_so	-0.03058135

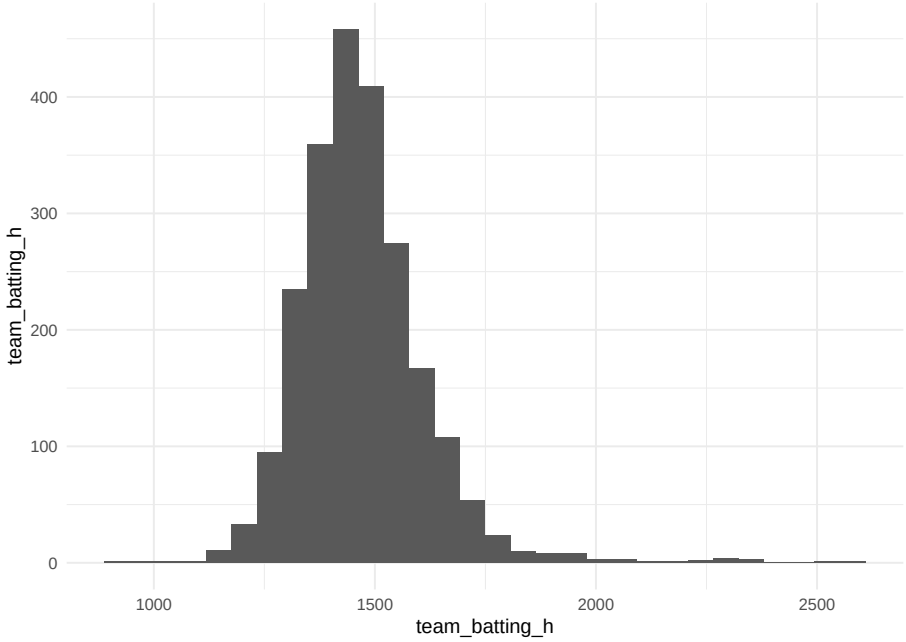
1	team_pitching_so	-0.07579967
1	team_pitching_h	-0.10993705
1	team_fielding_e	-0.17648476

Data Preparation

Since we've imputed missing values with median, let's perform a log transformation on variables with a non-normal distribution.

Log transformation

Let's check some histogram plots of the variables, then perform a log transformation to reduce skew. I'll show the first histogram, then the remaining in the appendix.



Build Models

Model 1

For our first model, let's use all the variables that have a positive correlation with `target_wins`. Our first model specification will be as follows:

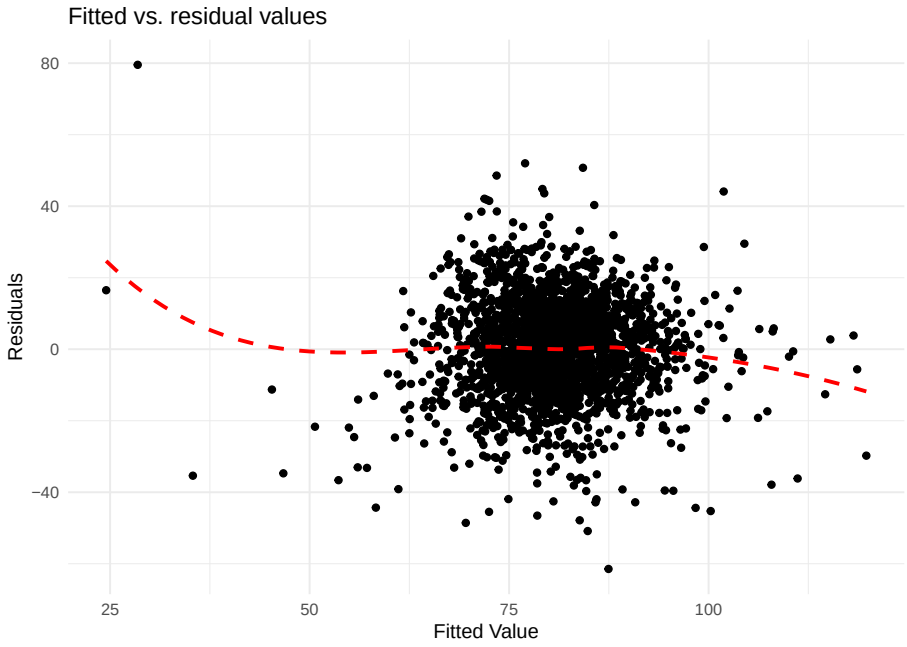
$wins = \beta_0 + \beta_1 BaseHits + \beta_2 Doubles + \beta_3 Walks + \beta_4 Homeruns + \beta_5 Triples + \beta_6 WalksAllowed + \beta_7 StolenBases + \beta_8 PitchesHit + \beta_9 CaughtStealing + \epsilon$

Based on the model specification, we would expect all the point estimates to be positive, since they have a positive correlation. From the results, `team_batting_2b` and `team_pitching_bb` both have negative point estimates. For `team_batting_2b`, the p-value is not significant, so less worry there. However, `team_pitching_bb` has a highly significant p-value, and a negative point estimate. We would expect that a team that allows more walks would perform worse, so perhaps the pearson correlation isn't an accurate statistic to use.

```
Call:
lm(formula = target_wins ~ team_batting_h + team_batting_2b +
    team_batting_bb + team_pitching_hr + team_batting_3b + team_pitching_bb +
    team_baserun_sb + team_batting_hbp + team_baserun_cs, data = training_set)

Residuals:
    Min      1Q  Median      3Q     Max
-61.458  -8.786   0.407   8.950  79.542

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)      -0.6339700  5.7257451  -0.111  0.91185
team_batting_h    0.0373831  0.0030905  12.096  < 2e-16 ***
team_batting_2b  -0.0003255  0.0089854  -0.036  0.97111
team_batting_bb   0.0328532  0.0030566  10.748  < 2e-16 ***
team_pitching_hr  0.0457178  0.0071783   6.369 2.30e-10 ***
team_batting_3b   0.0622612  0.0164639   3.782  0.00016 ***
team_pitching_bb -0.0082051  0.0020412  -4.020 6.02e-05 ***
team_baserun_sb   0.0217026  0.0040816   5.317 1.16e-07 ***
team_batting_hbp  0.0472568  0.0764613   0.618  0.53660
team_baserun_cs   0.0182602  0.0160753   1.136  0.25611
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 13.7 on 2266 degrees of freedom
Multiple R-squared:  0.2463,    Adjusted R-squared:  0.2433
F-statistic: 82.29 on 9 and 2266 DF,  p-value: < 2.2e-16
```

```
[1] "The mean squared error is: 186.92940081673"
```

Let's check the residuals vs. fitted plot.



Model 2

For the next model, let's use the Akaike information criterion algorithm to predict the quality of each model, then choose the model with the lowest AIC value.

The first results is the full model (using all predictors).

```
Call:
lm(formula = target_wins ~ ., data = training_set)

Residuals:
    Min      1Q  Median      3Q     Max
-49.821  -8.616   0.068   8.289  59.070

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)      21.3843494  6.7992701   3.145  0.00168 **
index            -0.0004530  0.0003766  -1.203  0.22918
team_batting_h    0.0488087  0.0036959  13.206  < 2e-16 ***
team_batting_2b  -0.0210728  0.0091813  -2.295  0.02181 *
team_batting_3b   0.0656929  0.0168328   3.903 9.79e-05 ***
team_batting_hr   0.0531978  0.0275007   1.934  0.05319 .
team_batting_bb   0.0102316  0.0058407   1.752  0.07995 .
team_batting_so  -0.0083756  0.0025502  -3.284  0.00104 **
team_baserun_sb   0.0257931  0.0043664   5.907 4.01e-09 ***
team_baserun_cs  -0.0108216  0.0157870  -0.685  0.49312
team_batting_hbp  0.0487185  0.0730953   0.667  0.50516
team_pitching_h  -0.0008239  0.0003678  -2.240  0.02518 *
team_pitching_hr  0.0129919  0.0243930   0.533  0.59436
team_pitching_bb  0.0006724  0.0041580   0.162  0.87154
team_pitching_so  0.0028321  0.0009221   3.071  0.00216 **
team_fielding_e  -0.0196745  0.0024632  -7.987 2.18e-15 ***
team_fielding_dp -0.1209399  0.0129572  -9.334  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 13.07 on 2259 degrees of freedom
Multiple R-squared:  0.3159,    Adjusted R-squared:  0.3111
F-statistic: 65.21 on 16 and 2259 DF,  p-value: < 2.2e-16

[1] "The mean squared error is: 169.659714506108"
```

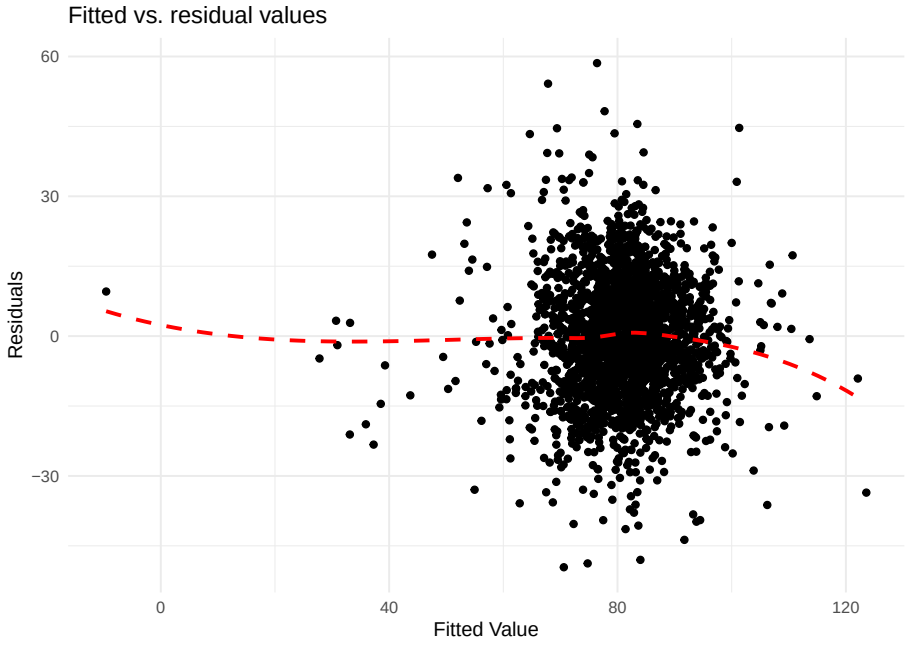
The second result is the model chose by `stepAIC` function from the `MASS` package. From the results below, the following predictors were

excluded: `team_baserun_cs`, `team_batting_hbp`, `team_pitching_bb`, and `team_pitching_hr`. Analyzing the coefficients in the table below, `team_batting_2b` again has a negative coefficient, as well as: `team_fielding_dp`.

For the full model, `team_batting_2b` does have a significant p-value up to the 2% threshold, however the point estimate is just slightly negative. The more interesting regressor in this case is `team_fielding_dp`, with a negative point estimate that is highly statistically significant.

In the stepwise-AIC model, we observe the same pattern, where `team_batting_2b` and `team_fielding_dp` have negative point estimates and are statistically significant.

```
Call:
lm(formula = target_wins ~ team_batting_h + team_batting_2b +
    team_batting_3b + team_batting_hr + team_batting_bb + team_batting_so +
    team_baserun_sb + team_pitching_h + team_pitching_so + team_fielding_e +
    team_fielding_dp, data = training_set)

Residuals:
    Min      1Q  Median      3Q     Max
-49.598  -8.593   0.085   8.445  58.582

Coefficients:
                   Estimate Std. Error t value Pr(>|t|)
(Intercept)      22.3440443  5.2338369   4.269 2.04e-05 ***
team_batting_h    0.0490922  0.0036699  13.377  < 2e-16 ***
team_batting_2b  -0.0213744  0.0091626  -2.333 0.019746 *
team_batting_3b   0.0665763  0.0166230   4.005 6.40e-05 ***
team_batting_hr   0.0674046  0.0096315   6.998 3.40e-12 ***
team_batting_bb   0.0115464  0.0033748   3.421 0.000634 ***
team_batting_so  -0.0085211  0.0024529  -3.474 0.000523 ***
team_baserun_sb   0.0249207  0.0042092   5.920 3.70e-09 ***
team_pitching_h  -0.0007770  0.0003209  -2.421 0.015552 *
team_pitching_so  0.0029662  0.0006719   4.415 1.06e-05 ***
team_fielding_e  -0.0190100  0.0023919  -7.948 2.97e-15 ***
team_fielding_dp -0.1217894  0.0129296  -9.419  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 13.07 on 2264 degrees of freedom
Multiple R-squared:  0.3151,    Adjusted R-squared:  0.3117
F-statistic: 94.68 on 11 and 2264 DF,  p-value: < 2.2e-16

[1] "The mean squared error is: 169.876854024882"
```


Fitted vs. residual values
60
30
0
−30
Residuals
0
40
80
120
Fitted Value

Model 3

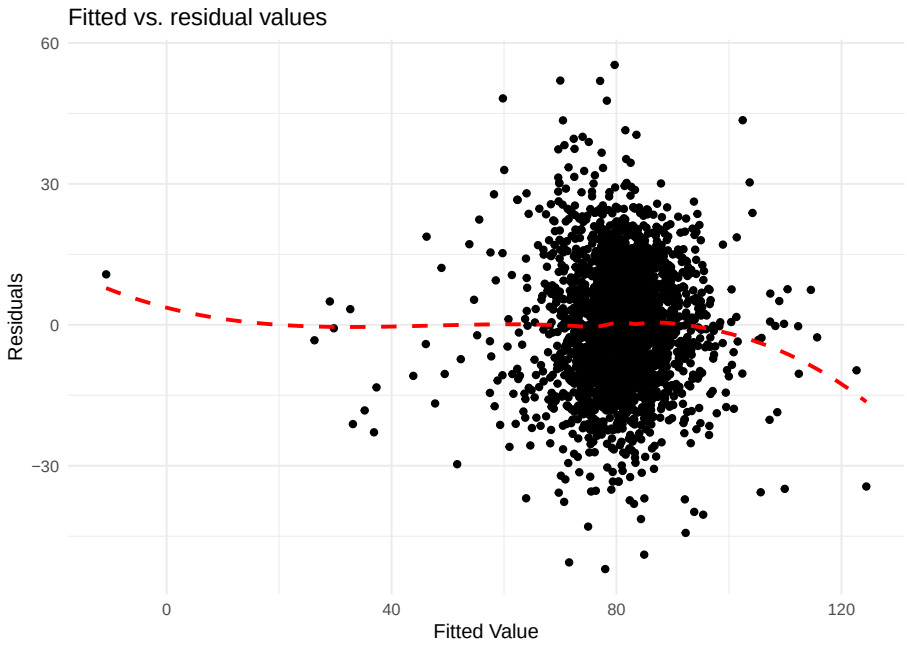
For the last model, let's take an augmented version of model 1, where we include relevant predictors with a negative correlation with the regressand to better fit the model.

$wins = \beta_0 + \beta_1 BaseHits + \beta_2 Doubles + \beta_3 Walks + \beta_4 Homeruns + \beta_5 Triples + \beta_6 WalksAllowed + \beta_7 StolenBases + \beta_8 PitchesHit + \beta_9 CaughtStealing + \beta_{10} HitsAllowed + \beta_{11} Errors + \beta_{12} Strikeouts + \epsilon$

```
Call:
lm(formula = target_wins ~ team_batting_h + team_batting_2b +
    team_batting_bb + team_pitching_hr + team_batting_3b + team_pitching_bb +
    team_baserun_sb + team_batting_hbp + team_pitching_so + team_fielding_e +
    team_pitching_h + team_baserun_cs, data = training_set)

Residuals:
    Min      1Q  Median      3Q     Max
-51.994  -9.001   0.006   8.642  55.326

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)     -3.986e-01  5.994e+00  -0.066  0.94699
team_batting_h   5.142e-02  3.422e-03  15.027  < 2e-16 ***
team_batting_2b -2.616e-02  9.179e-03  -2.850  0.00442 **
team_batting_bb  6.369e-03  5.201e-03   1.225  0.22087
team_pitching_hr 3.158e-02  7.140e-03   4.423 1.02e-05 ***
team_batting_3b  7.180e-02  1.650e-02   4.351 1.42e-05 ***
team_pitching_bb -1.278e-05  3.505e-03  -0.004  0.99709
team_baserun_sb  2.850e-02  4.161e-03   6.850 9.45e-12 ***
team_batting_hbp 5.793e-02  7.443e-02   0.778  0.43651
team_pitching_so 2.109e-03  7.889e-04   2.673  0.00757 **
team_fielding_e -2.064e-02  2.491e-03  -8.287  < 2e-16 ***
team_pitching_h -7.055e-04  3.719e-04  -1.897  0.05798 .
team_baserun_cs -1.560e-02  1.603e-02  -0.973  0.33046
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 13.34 on 2263 degrees of freedom
Multiple R-squared:  0.2871,    Adjusted R-squared:  0.2833
F-statistic: 75.94 on 12 and 2263 DF,  p-value: < 2.2e-16

[1] "The mean squared error is: 176.818083457501"
```

Let's check the residuals vs. fitted plot:

```
`geom_smooth()` using formula = 'y ~ x'
```



Select Models

From the models in the `Build Models` section, to choose the final model we need to think about multicollinearity issues & normality of the residuals. Each section below will have a short-discussion about the performance and inference issues of the three models. The four diagnostics plots can be found in the relevant appendix section.

Model 1 Diagnostics:

Model 1's biggest problem is the strong positive correlation the predictors share. We would expect that as the number of base hits by batters increases, so would doubles, triples and home-runs. The adjusted R^2 for this model is .2433. The residuals vs. fitted chart is showing a slight pattern towards the low end of target wins. Model 1 had the highest mean squared error at 186.9294.

Model 2 Diagnostics:

Model 2's performance looks the most promising based on the adjusted R^2 of .3111 and the diagnostic plots. The residual vs. fitted chart does show a slight pattern towards the high end of target wins, however it is not nearly as pronounced as model 1. The normal QQ plot has a more linear relationship than 1 as well. Model 2 had the lowest mean squared error at 169.8769.

Model 3 Diagnostics:

I would rank model 3's performance as better than model 1, however slightly worse than model 2. The fitted vs. residual plot shows a stronger downward trend on the higher end of target wins compared to 2, and a lower adjusted R^2 at .2833. The normal QQ plot is also slightly less linear than compared to model 2. Model 3's mean squared error was in the middle, at 176.8181.

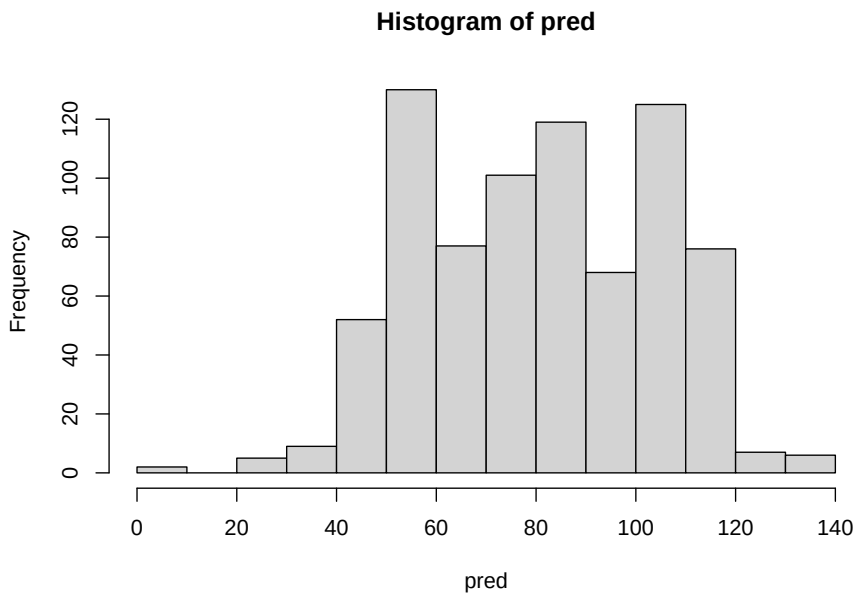
Evaluation

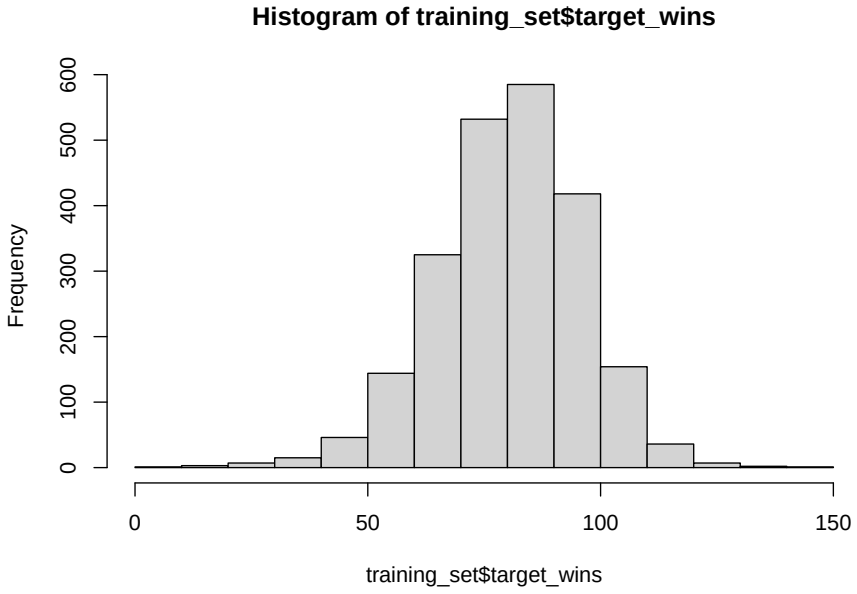
Based on the following metrics: Adjusted R^2, mean squared error and the diagnostic plots, model 2 is the clear favorite. Now we will evaluate its performance against the evaluation data set. We will perform the same data imputation method as the training data-set for accuracy. Overall, model 2 was chose because:

1. It had the lowest mean squared error at 169.8769
2. It had the highest adjusted R^2 at .3111
3. The most linear normal QQ plot and the best fitted. vs residuals plot in terms of non-randomness.
4. It had the highest F-statistic at 94.68 and the most significant amount of variables based on the p-value.

From the histogram plots comparing the target wins of the training set and the predicted values, we can see that the model is over-estimating the amount of wins in the 40-60 range, as well as the 100-120 range. However, still fairly accurate.

Compare the histogram of the training set's target wins and predicted target wins.






```
[1] "The mean squared error of the predicted values is: 763.096578243557"
```

Appendix: R code

Summary statistics

```
## Median
training_set |>
  dplyr::select(-index) |>
  summarise(across(everything(), ~ median(.))) |>
  glimpse()

## Mean
training_set |>
  dplyr::select(-index) |>
  summarise(across(everything(), ~ mean(.))) |>
  glimpse()

## Standard Deviation
training_set |>
  dplyr::select(-index) |>
  summarise(across(everything(), ~ sd(.))) |>
  glimpse()
```

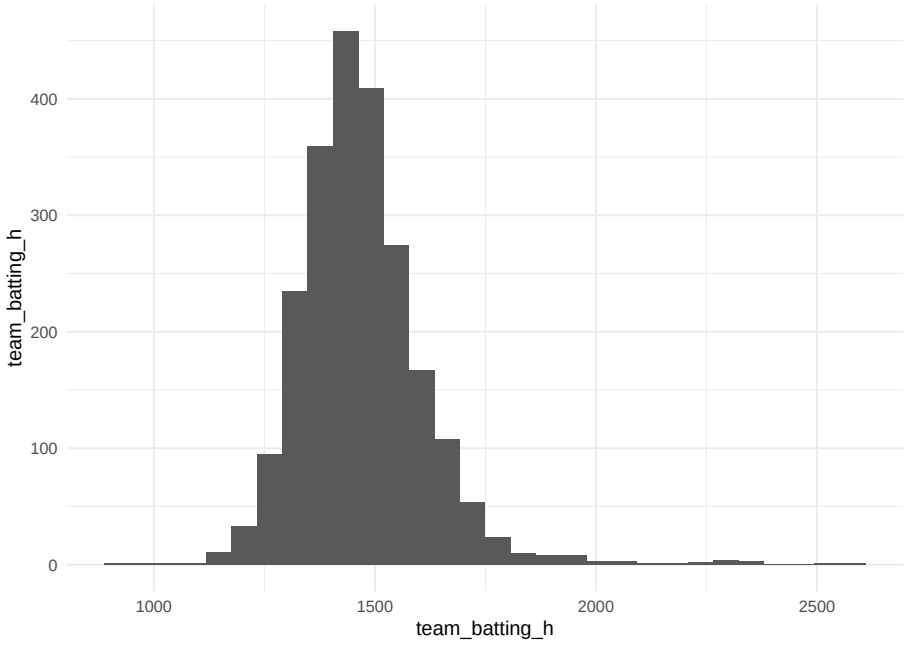
Plots

```
# Create named character vector of variables
vars <- training_set |>
  # select(-index) |>
  names() |>
  set_names()

# Use map function to create a sequence of plots
scatter_plots <- map(vars, ~ ggplot(data = training_set) +
  geom_point(aes(x = target_wins, y = .data[[.x]])) +
  theme_minimal() +
  labs(y = .x))

hist_plots <- map(vars, ~ ggplot(data = training_set) +
  geom_histogram(aes(x = .data[[.x]])) +
  theme_minimal() +
  labs(y = .x))


# Correlation matrix
```

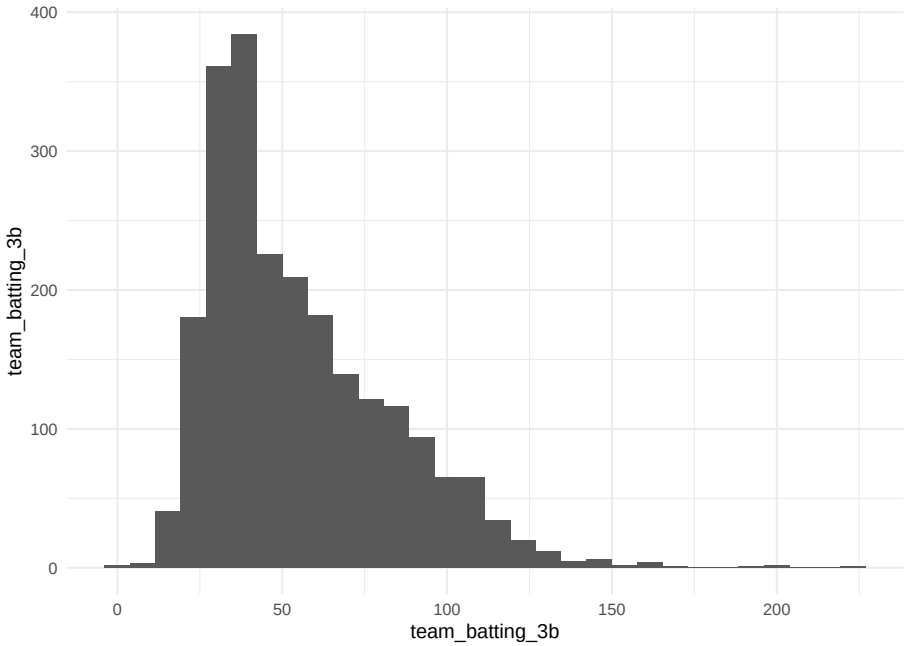
```r
cor_matrix <- training_set |>
  dplyr::select(-index) |>
  cor() |>
  as.matrix()

corrplot(cor_matrix)
```

```r
# Get correlation values as a table, sorted highest to lowest
purrr::map_df(vars, ~ cor(training_set$target_wins, training_set[[.x]])) |>
  pivot_longer(cols = !c("target_wins"), names_to = "correlation") |>
  arrange(desc(value)) |>
  gt::gt()
```

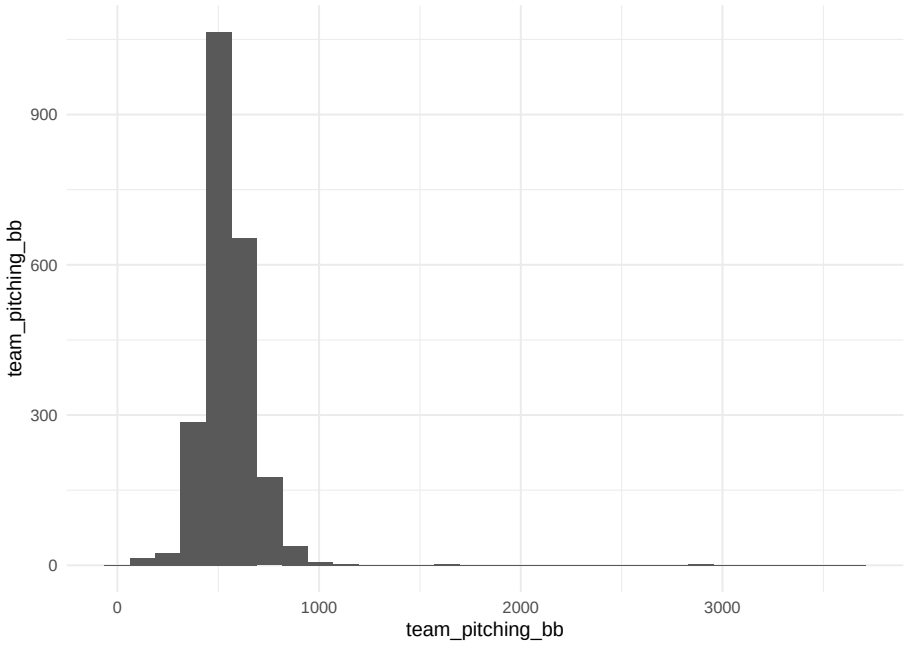
```r
hist_plots$team_batting_h
```



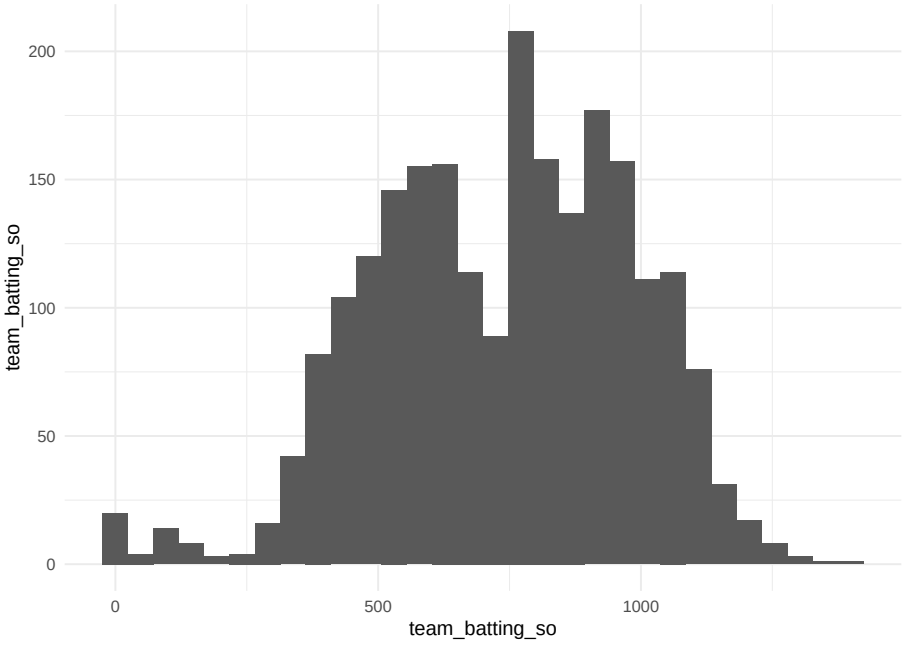

```r
hist_plots$team_batting_3b
```



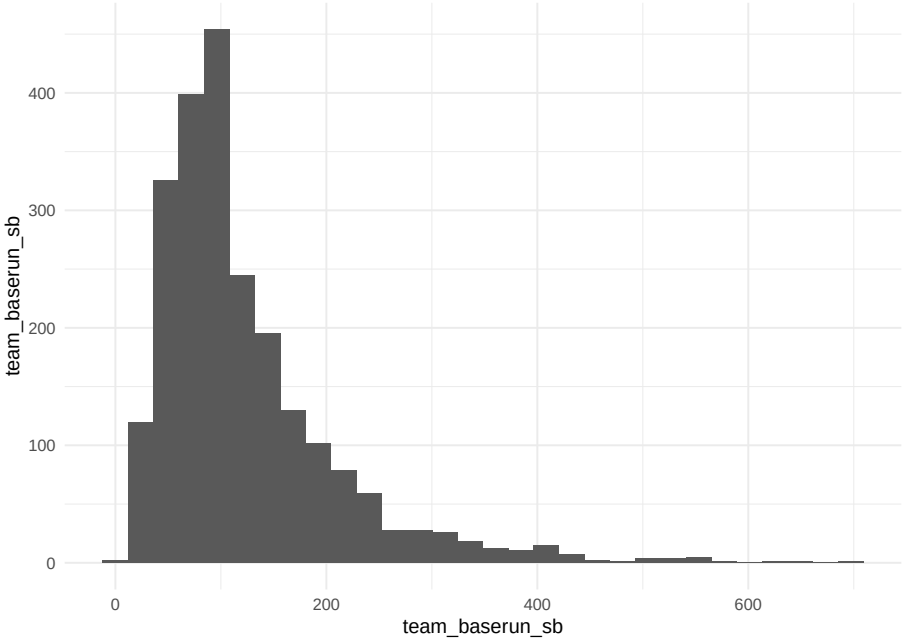

```
hist_plots$team_pitching_bb
```



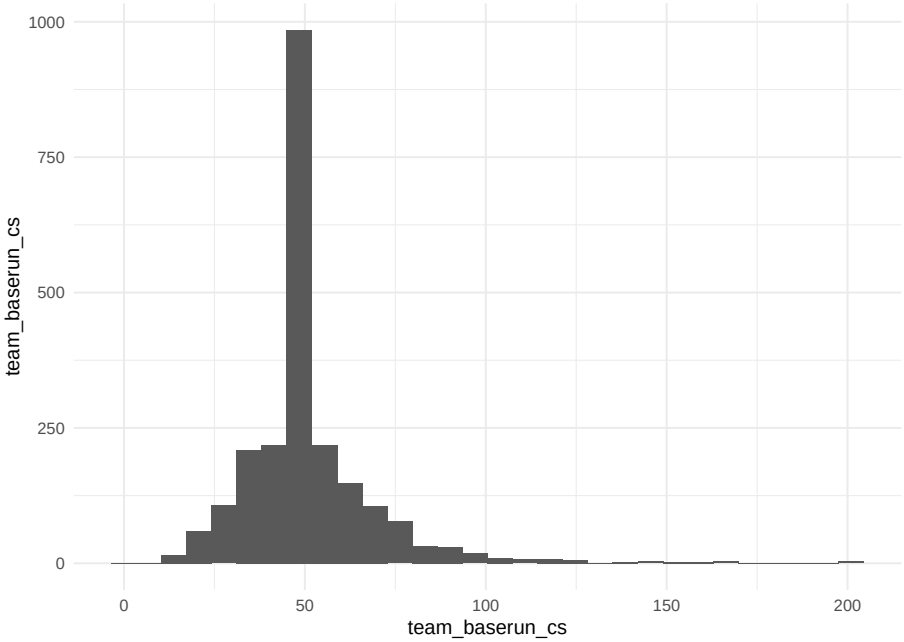
```
hist_plots$team_batting_so
```



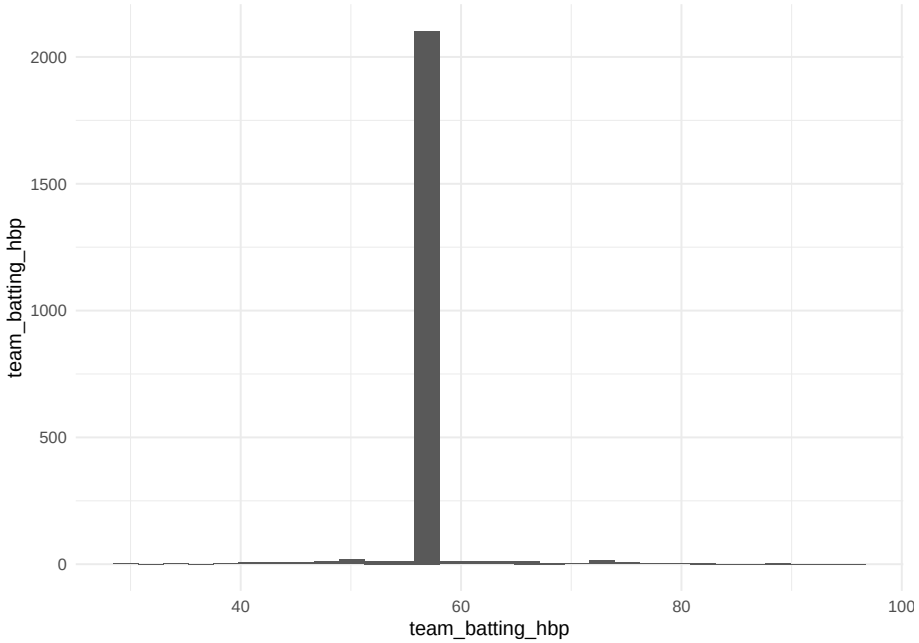

```
hist_plots$team_baserun_sb
```



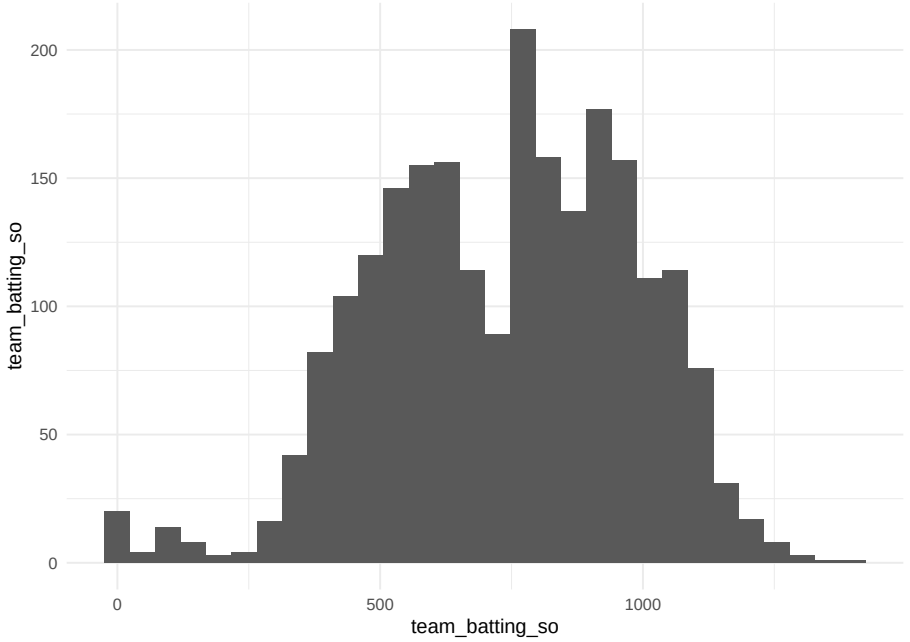

```
hist_plots$team_baserun_cs
```



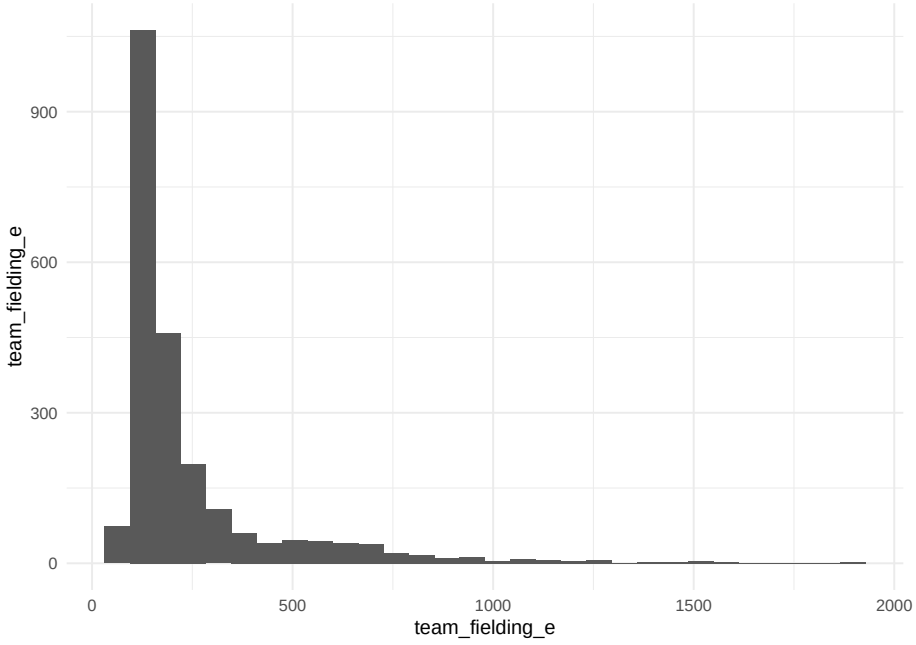
```
hist_plots$team_batting_hbp
```




```
hist_plots$team_batting_so
```




```
hist_plots$team_fielding_e
```



Transformations

```
# Log transformation
training_set |>
  mutate(across(
    .cols = c(
      "team_batting_h", "team_batting_3b", "team_pitching_bb",
      "team_batting_so", "team_baserun_sb", "team_baserun_cs",
      "team_batting_hbp", "team_batting_so", "team_fielding_e"
    ),
    .fns = log
  ))
```

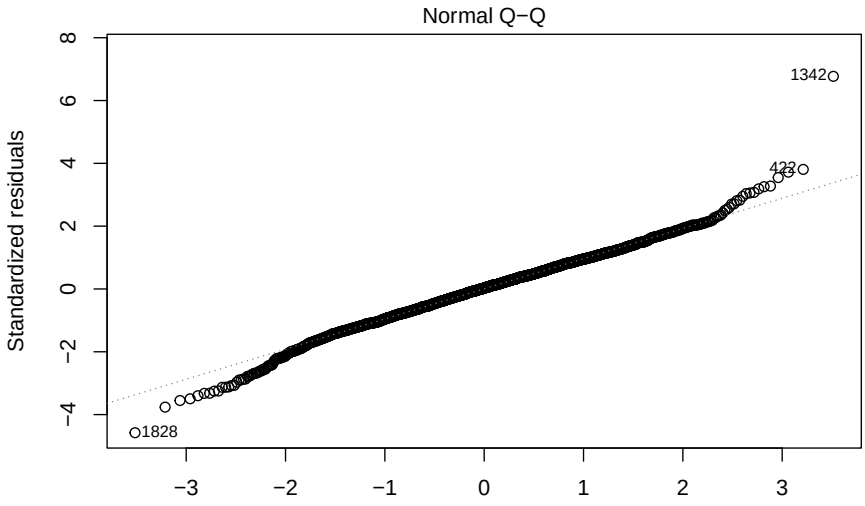
Models

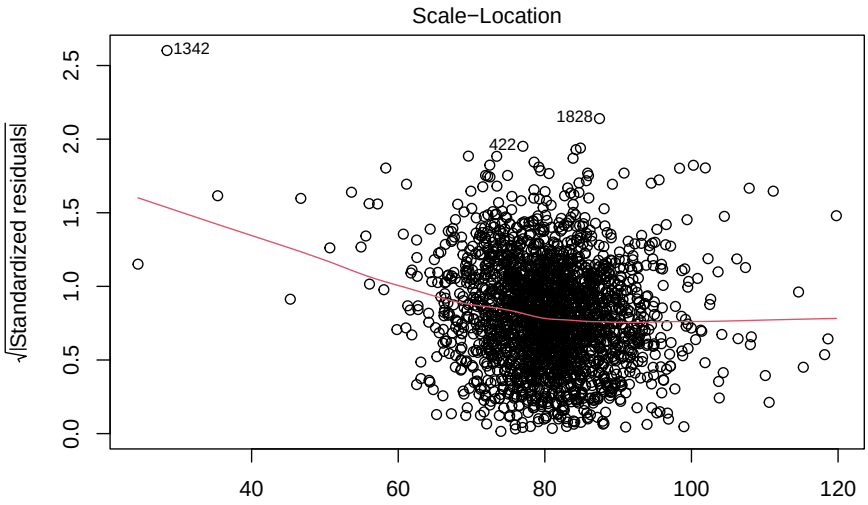
Model 1

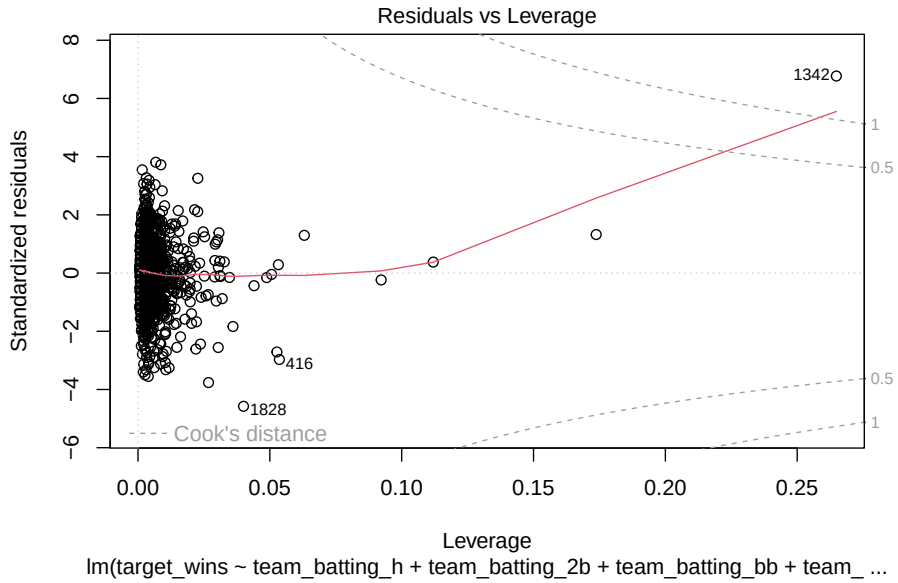
```
## First model: Only use variables with positive correlation coefficient
lm_fit_1 <- lm(target_wins ~ team_batting_h +
  team_batting_2b + team_batting_bb +
  team_pitching_hr + team_batting_3b +
  team_pitching_bb + team_baserun_sb +
  team_batting_hbp + team_baserun_cs, data = training_set)
tidy(lm_fit_1)

lm_df <- broom::augment(lm_fit_1)
lm_df |>
  ggplot(aes(x = .fitted, y = .resid)) +
  geom_point() +
  geom_smooth(method = "loess", se = FALSE, linetype = "dashed", color = "red") +
  labs(
    title = "Fitted vs. residual values"
  ) +
  xlab("Fitted Value") +
  ylab("Residuals") +
  theme_minimal()
```

```
plot(lm_fit_1)
```


Residuals vs Fitted
1342
422
1828
Residuals
50
0
−50
40
60
80
100
120
Fitted values
lm(target_wins ~ team_batting_h + team_batting_2b + team_batting_bb + team_ ...


Normal Q−Q
1342
423
1828
Standardized residuals
8
6
4
2
0
−2
−4
−3
−2
−1
0
1
2
3
Theoretical Quantiles
lm(target_wins ~ team_batting_h + team_batting_2b + team_batting_bb + team_ ...

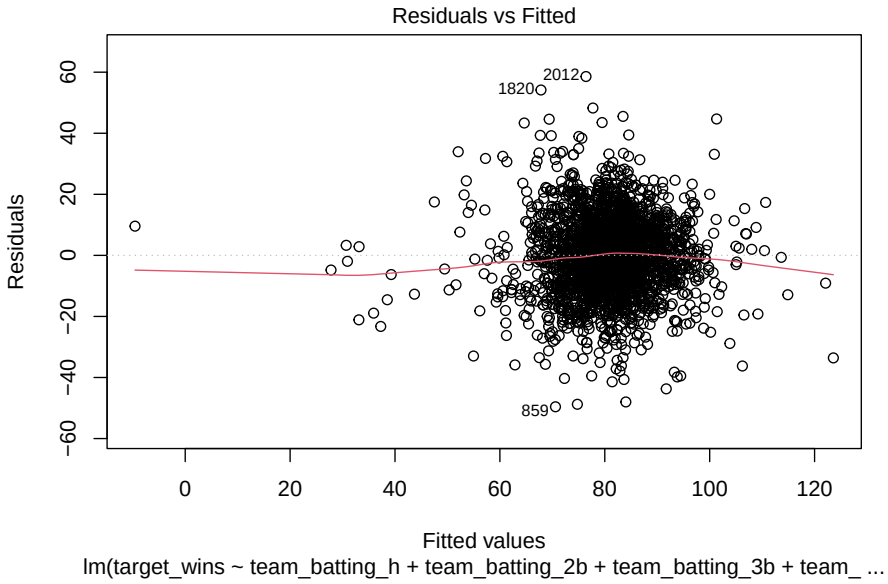

Scale−Location
√|Standardized residuals|
Fitted values
lm(target_wins ~ team_batting_h + team_batting_2b + team_batting_bb + team_ ...
1342
1828
422

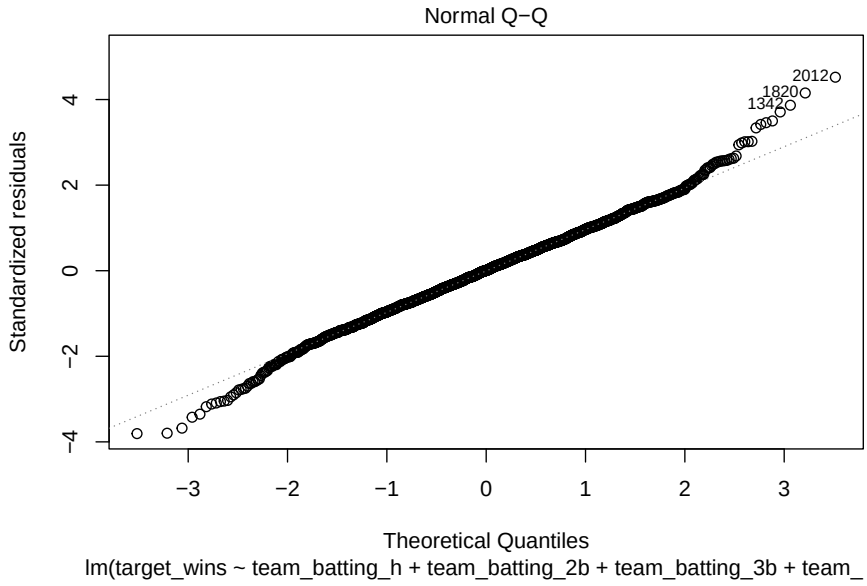

Residuals vs Leverage
Standardized residuals
Leverage
lm(target_wins ~ team_batting_h + team_batting_2b + team_batting_bb + team_ ...
Cook's distance
1342
416
1828

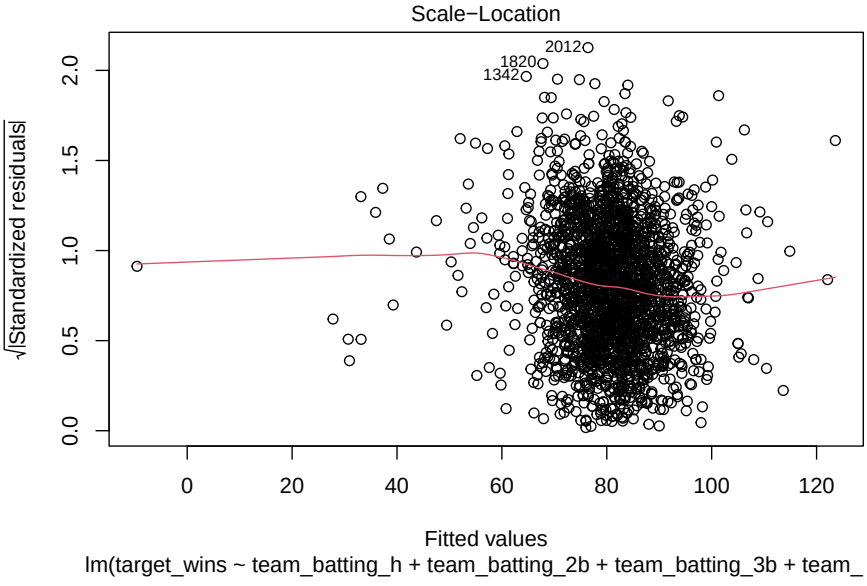
Model 2

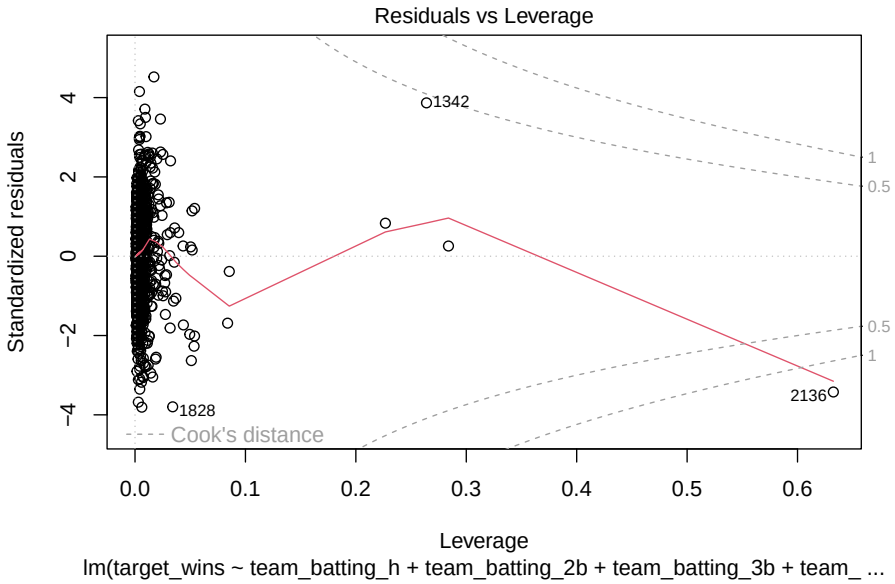
```r
lm_fit_2 <- lm(target_wins ~ ., data = training_set)
tidy(lm_fit_2)

lm_fit_2_stepwise <- MASS::stepAIC(lm_fit_2, direction = "both", trace = FALSE)
tidy(lm_fit_2_stepwise)
lm_df_2 <- broom::augment(lm_fit_2_stepwise)
lm_df_2 |>
  ggplot(aes(x = .fitted, y = .resid)) +
  geom_point() +
  geom_smooth(method = "loess", se = FALSE, linetype = "dashed", color = "red") +
  labs(
    title = "Fitted vs. residual values"
  ) +
  xlab("Fitted Value") +
  ylab("Residuals") +
  theme_minimal()
```

```r
plot(lm_fit_2_stepwise)
```




Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(target_wins ~ team_batting_h + team_batting_2b + team_batting_3b + team_ ...
2012
1820
1342


Scale−Location
√|Standardized residuals|
Fitted values
lm(target_wins ~ team_batting_h + team_batting_2b + team_batting_3b + team_ ...
2012
1820
1342

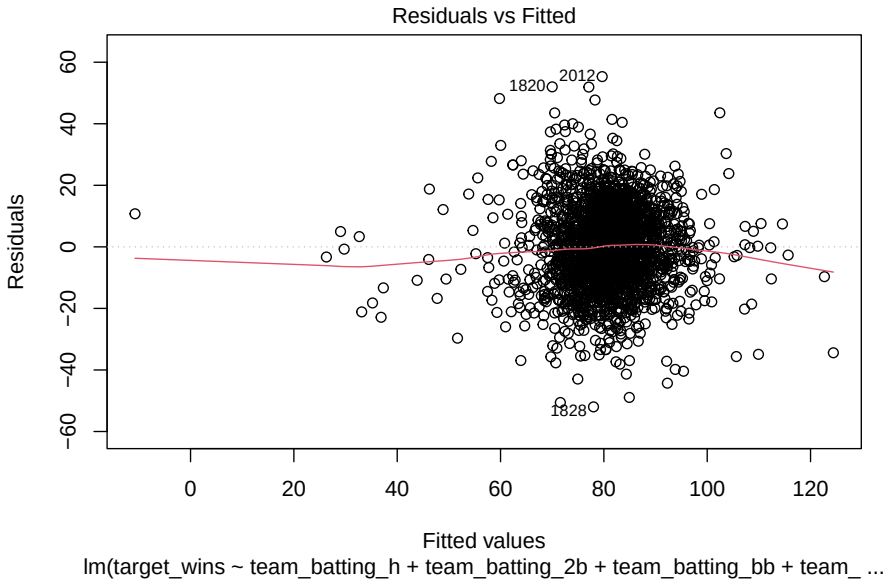


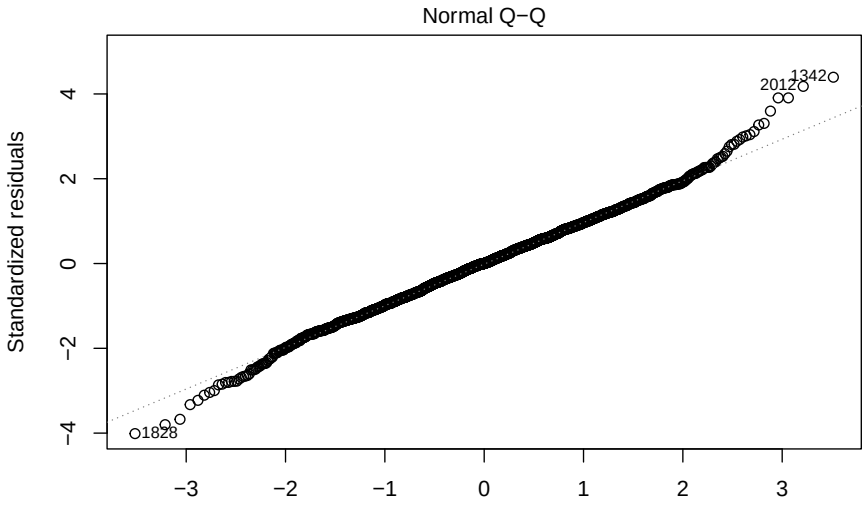
Model 3

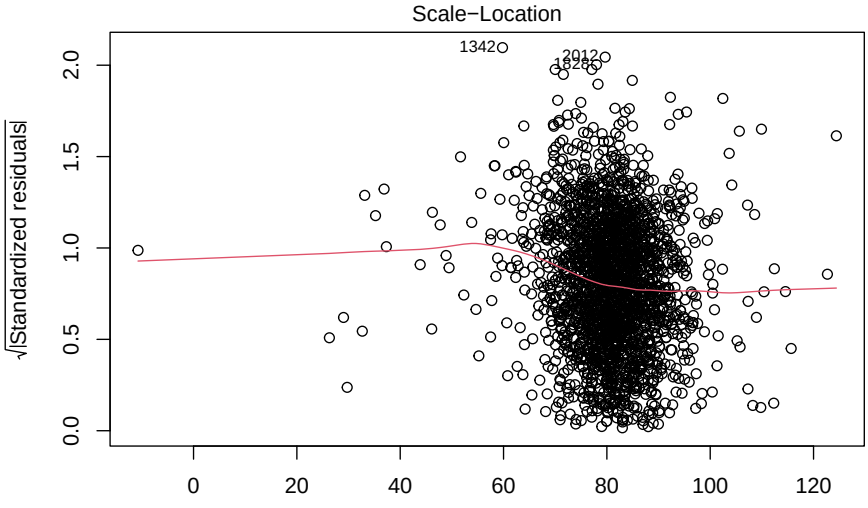
```
lm_fit_3 <- lm(
  target_wins ~ team_batting_h +
    team_batting_2b +
    team_batting_bb +
    team_pitching_hr +
    team_batting_3b +
    team_pitching_bb +
    team_baserun_sb +
    team_batting_hbp +
    team_pitching_so +
    team_fielding_e +
    team_pitching_h +
    team_baserun_cs,
  data = training_set
)

lm_df_3 <- broom::augment(lm_fit_3)
lm_df_3 |>
  ggplot(aes(x = .fitted, y = .resid)) +
  geom_point() +
```

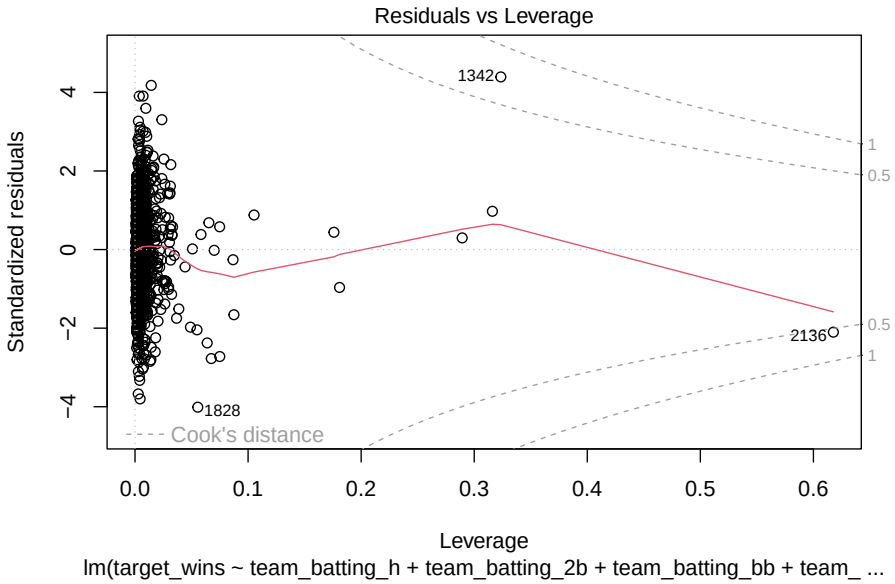
```
  geom_smooth(method = "loess", se = FALSE, linetype = "dashed", color = "red") +
  labs(
    title = "Fitted vs. residual values"
  ) +
  xlab("Fitted Value") +
  ylab("Residuals") +
  theme_minimal()
```

```
plot(lm_fit_3)
```




Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(target_wins ~ team_batting_h + team_batting_2b + team_batting_bb + team_ ...
1342
2012
1828


Scale−Location
√|Standardized residuals|
Fitted values
lm(target_wins ~ team_batting_h + team_batting_2b + team_batting_bb + team_ ...
1342
2012
1828



Evaluation

```
eval_set <- readr::read_csv(
  here::here("data", "moneyball-evaluation-data (1).csv")
) |>
  janitor::clean_names() |>
  glimpse()

eval_set <- eval_set |>
  mutate(across(everything(), ~ tidyr::replace_na(., median(., na.rm = TRUE)))) |>
  glimpse()
```

```
pred <- predict(lm_fit_2_stepwise, eval_set, interval = "prediction")
summary(pred)

pred_2 <- predict(lm_fit_2_stepwise, eval_set, interval = "prediction", level = .99)
summary(pred_2)
hist(pred_2)
```

```
pred_conf <- predict(lm_fit_2_stepwise, eval_set, interval = "confidence", se.fit = TRUE)
summary(pred_conf)
```

```r
training_fit <- training_set |>
  slice_sample(n = 259)

# Mean squared error
print(paste0(
  "The mean squared error of the predicted values is: ",
  mean((training_fit$target_wins -
    predict(lm_fit_2_stepwise, eval_set, interval = "prediction"))^2)
))
```