Homework 1

Sam Kuhn

11/20/22

Table of contents

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)

#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`.

```r
#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.

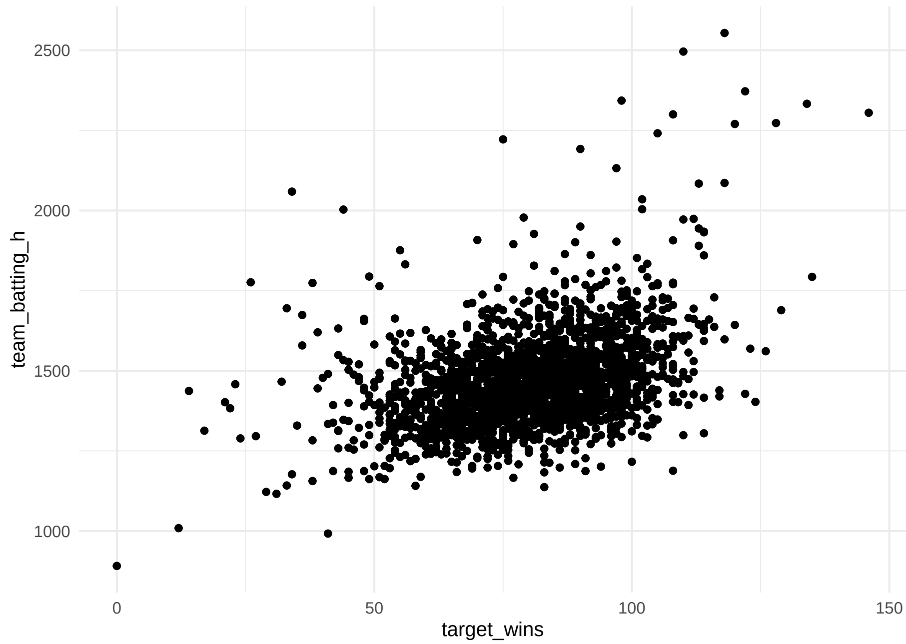
Table 1: Summary Statistics

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

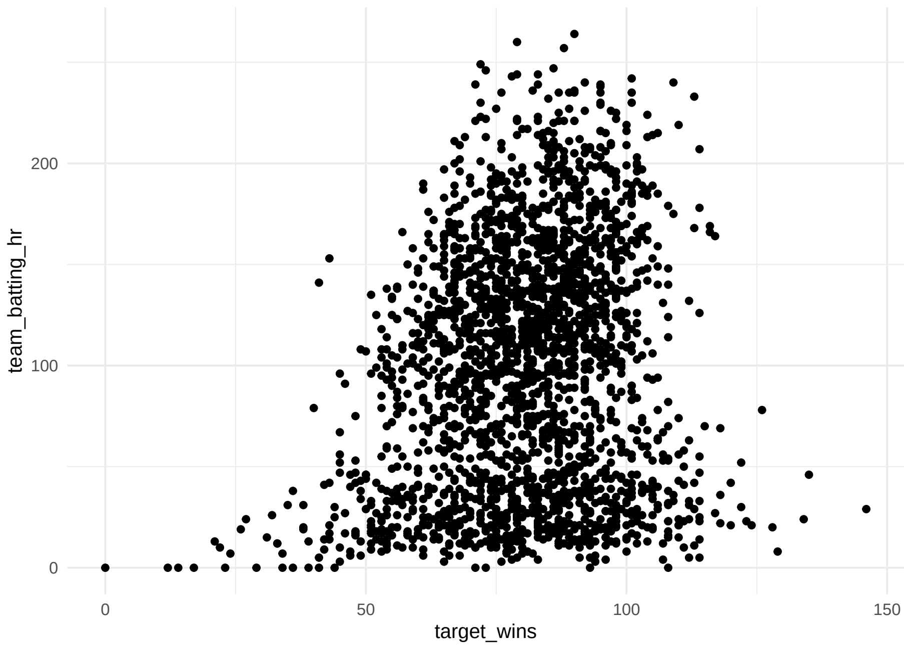
Summary plots

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

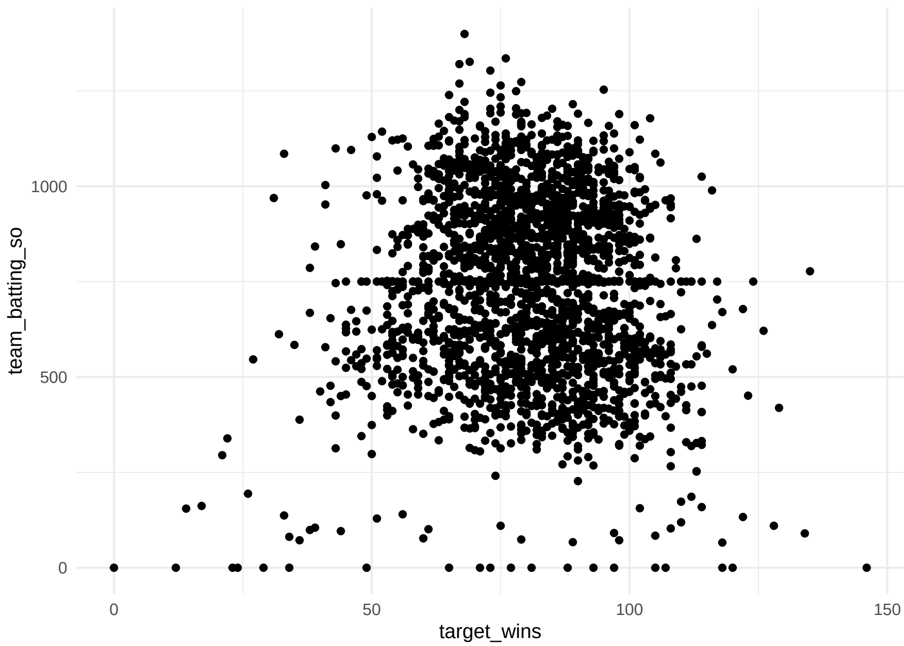
This first plot is ***base hits by batters vs. number of wins***. We can see that most of the observations are centered around 1500 hits, and ~80 wins, with a positive linear relationship.



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.

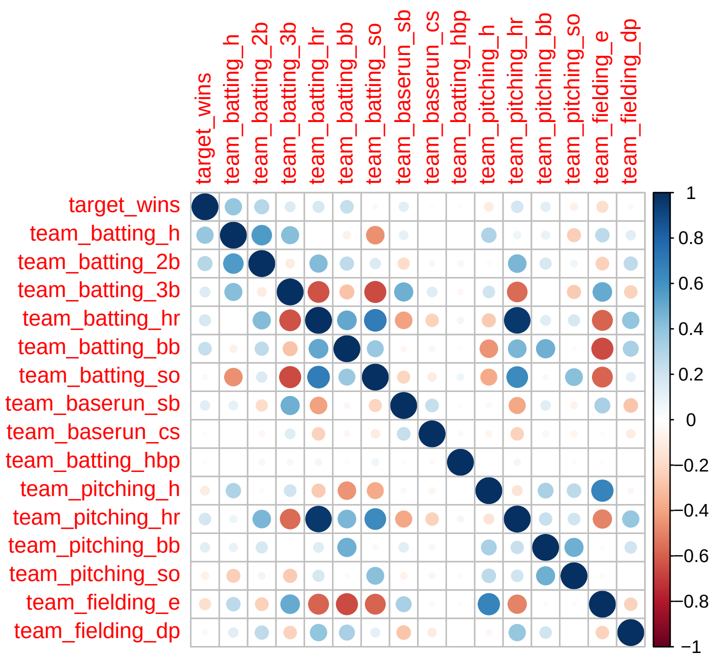


Lastly, let's look at a variable that has a negative impact on wins - ***strikeouts by batters***. This plot looks fairly similar to the plot above, however its shows a pattern that teams win more than 100 games generally don't give up more than 1,000 strikeouts a season.


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

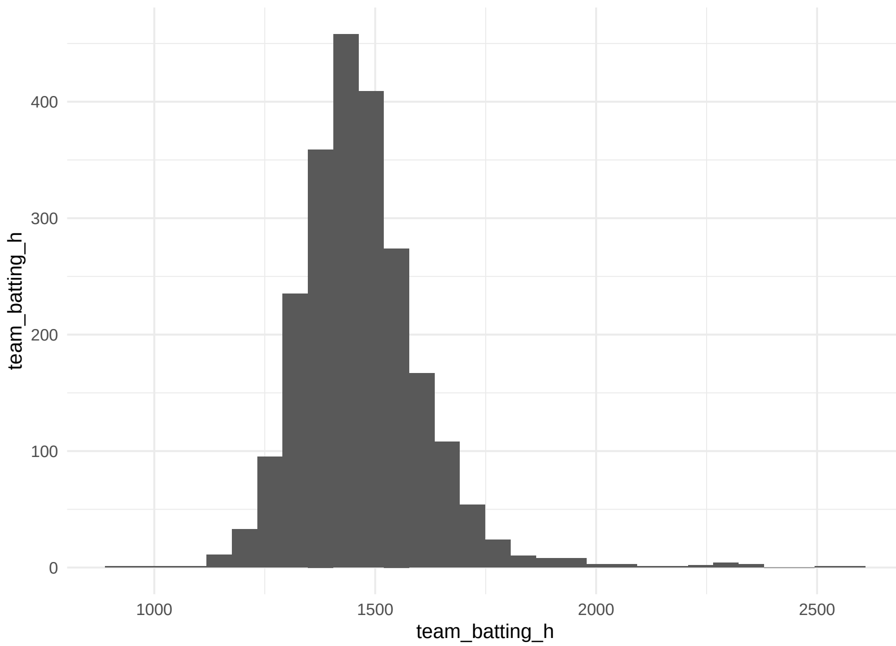
1	team_pitching_so	-0.07579967
1	team_pitching_h	-0.10993705
1	team_fielding_e	-0.17648476

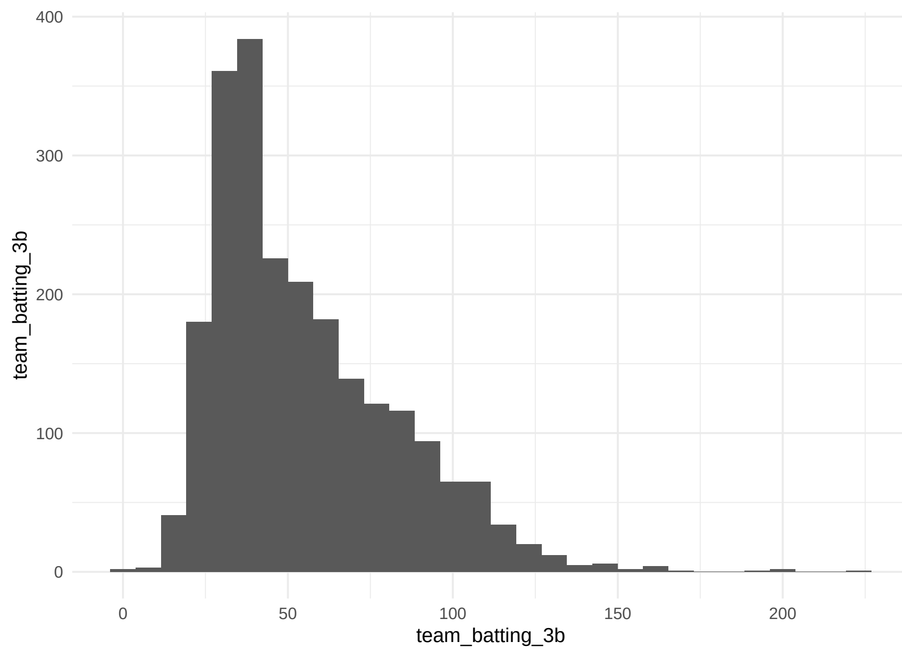
Data Preparation

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

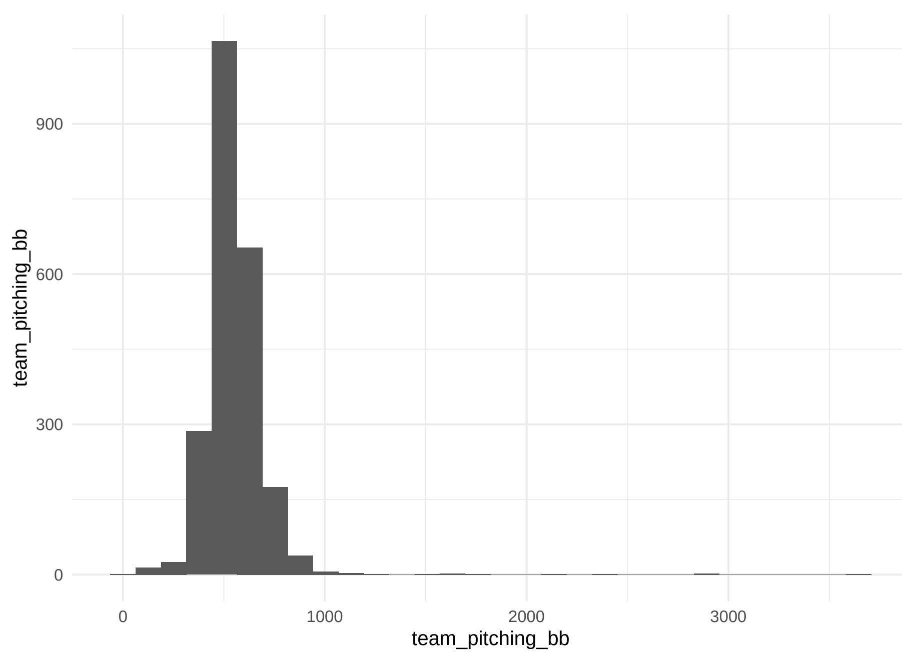
Log transformation

Let's check some histogram plots of the variables, then perform a log transformation to reduce skew.

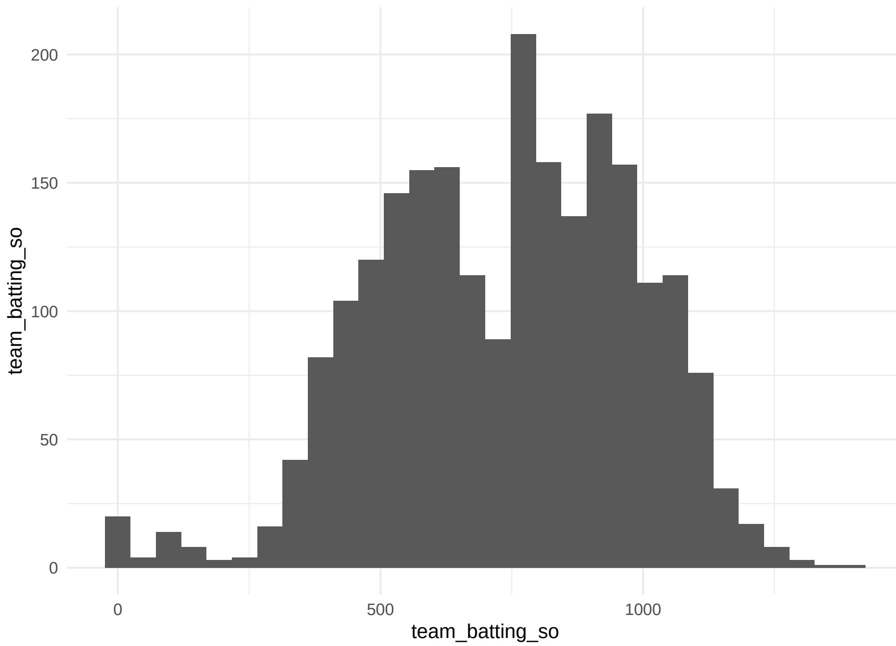


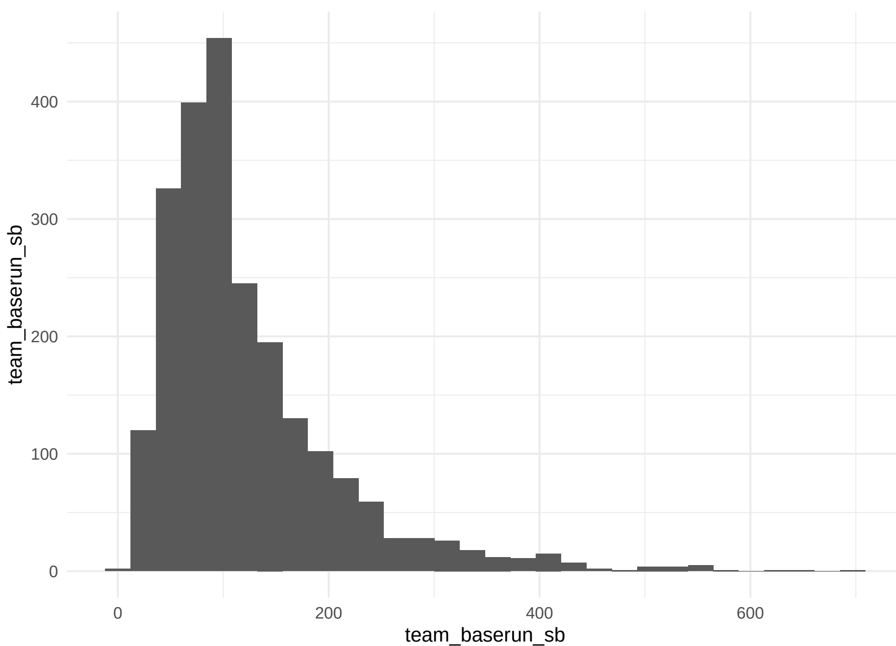


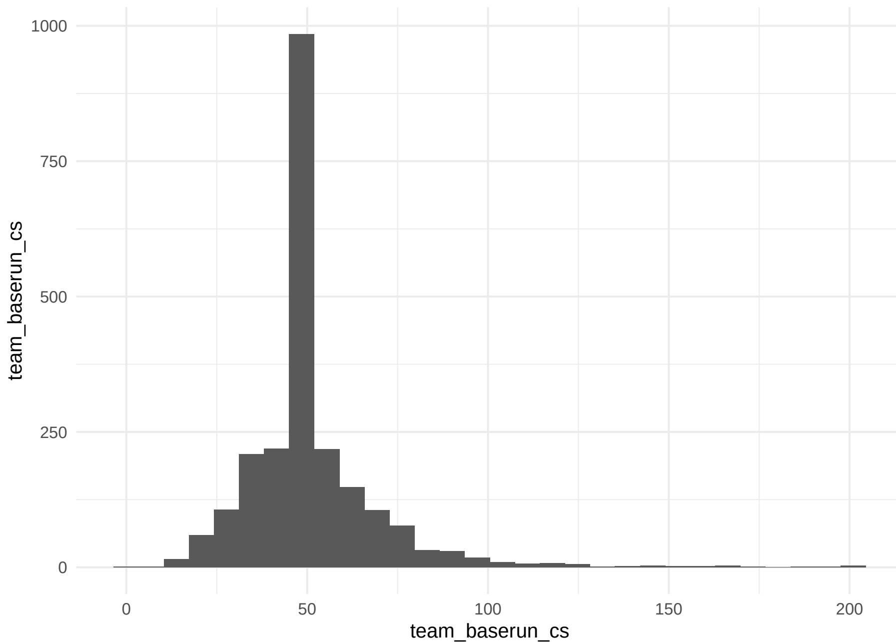
400
300
200
100
0
team_batting_3b
0
50
100
150
200
team_batting_3b

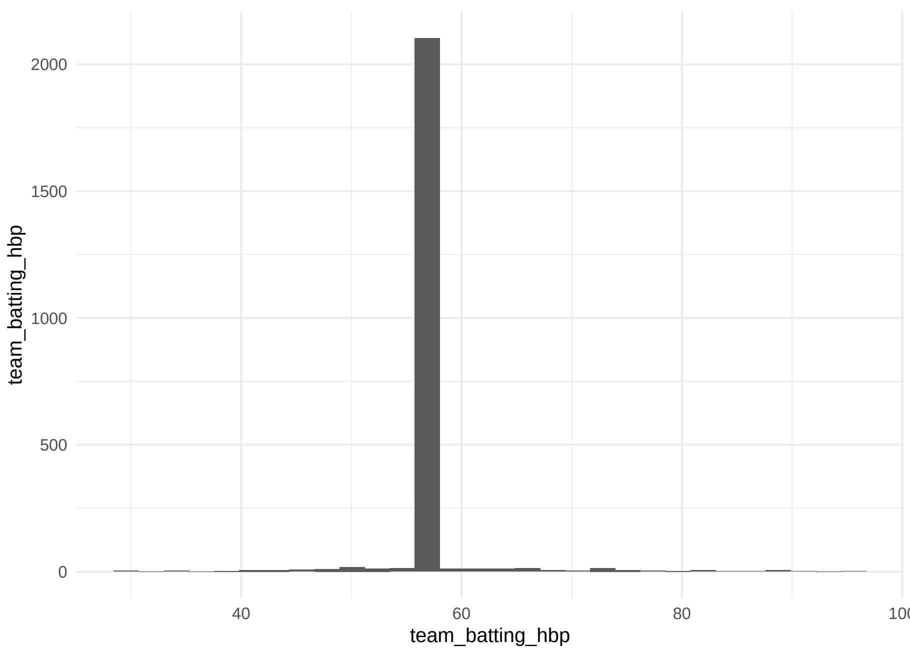


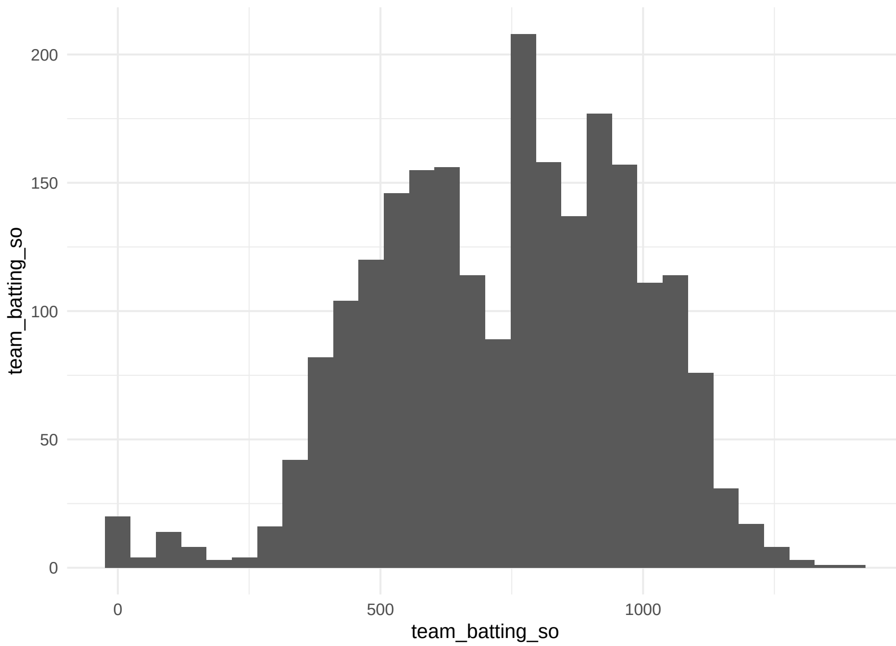
900
600
300
0
team_pitching_bb
0
1000
2000
3000
team_pitching_bb

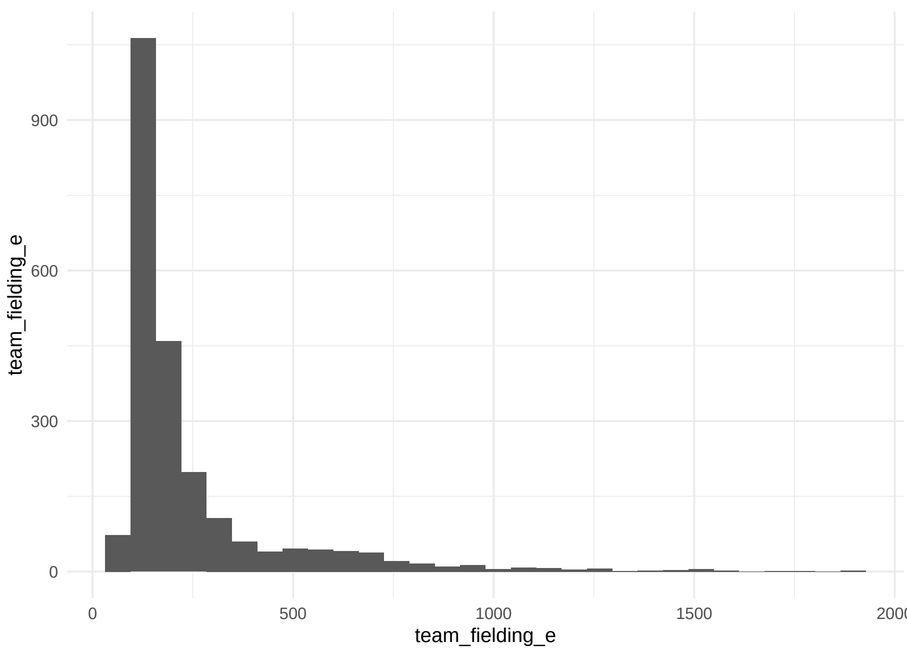

team_batting_so
team_batting_so
0
50
100
150
200
0
500
1000


team_baserun_sb
team_baserun_sb
0
100
200
300
400
0
200
400
600


1000
750
500
250
0
team_baserun_cs
0
50
100
150
200
team_baserun_cs


2000
1500
1000
500
0
team_batting_hbp
40
60
80
100
team_batting_hbp


team_batting_so
200
150
100
50
0
0
500
1000
team_batting_so


team_fielding_e
900
600
300
0
0
500
1000
1500
2000
team_fielding_e

Build Models

Model 1

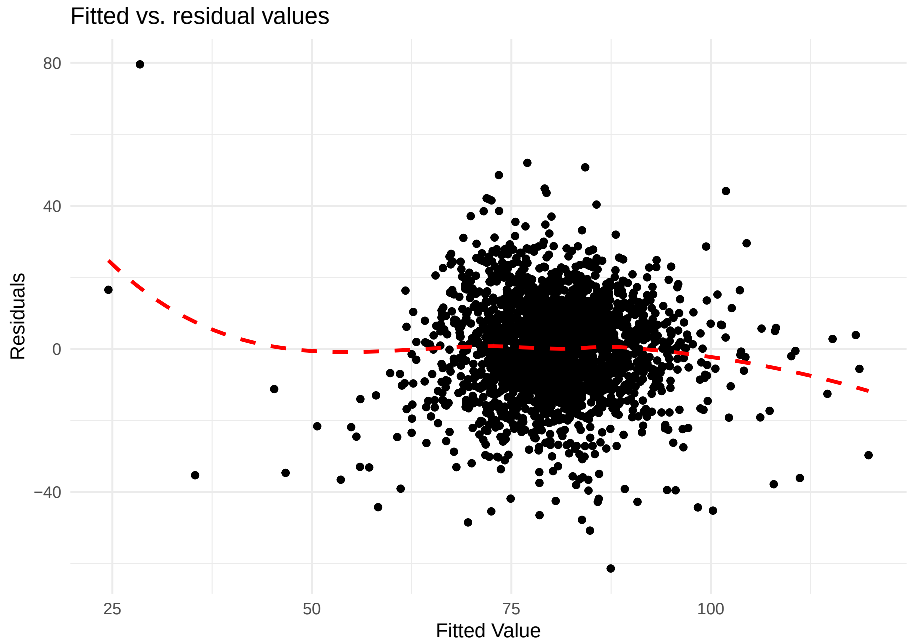
For our first model, let's use all the variables that have a positive correlation with `target_wins`. Our first model specification will be as follows:

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

```
# A tibble: 10 x 5
   term             estimate std.error statistic  p.value
   <chr>               <dbl>     <dbl>     <dbl>    <dbl>
 1 (Intercept)      -0.634      5.73      -0.111  9.12e- 1
 2 team_batting_h    0.0374     0.00309   12.1    1.10e-32
 3 team_batting_2b  -0.000325   0.00899   -0.0362 9.71e- 1
 4 team_batting_bb   0.0329     0.00306   10.7    2.57e-26
 5 team_pitching_hr  0.0457     0.00718    6.37   2.30e-10
 6 team_batting_3b   0.0623     0.0165     3.78   1.60e- 4
 7 team_pitching_bb -0.00821    0.00204   -4.02   6.02e- 5
 8 team_baserun_sb   0.0217     0.00408    5.32   1.16e- 7
 9 team_batting_hbp  0.0473     0.0765     0.618  5.37e- 1
10 team_baserun_cs   0.0183     0.0161     1.14   2.56e- 1
```

Let's check the residuals vs. fitted plot.

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



Select Models

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)
)
```

```r
#Correlation matrix
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()
```

Transformations

```r
# 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
  ))
```

```
# A tibble: 2,276 x 17
   index targe~1 team_~2 team_~3 team_~4 team_~5 team_~6 team_~7 team_~8 team_~9
   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
 1     1      39    7.28     194    3.66      13     143    6.74    4.62    3.89
 2     2      70    7.20     219    3.09     190     685    6.98    3.61    3.33
 3     3      86    7.23     232    3.56     137     602    6.82    3.83    3.30
 4     4      70    7.23     209    3.64      96     451    6.83    3.76    3.40
 5     5      82    7.17     186    3.30     102     472    6.82    3.89    3.66
 6     6      75    7.15     200    3.58      92     443    6.88    4.67    4.08
 7     7      80    7.13     179    3.99     122     525    6.97    4.38    3.99
 8     8      85    7.15     171    3.61     115     456    6.93    3.69    3.58
 9    11      86    7.24     197    3.69     114     447    6.83    4.23    3.30
10    12      76    7.15     213    2.89      96     441    6.72    4.28    3.53
# ... with 2,266 more rows, 7 more variables: team_batting_hbp <dbl>,
#   team_pitching_h <dbl>, team_pitching_hr <dbl>, team_pitching_bb <dbl>,
#   team_pitching_so <dbl>, team_fielding_e <dbl>, team_fielding_dp <dbl>, and
#   abbreviated variable names 1: target_wins, 2: team_batting_h,
```

```
#   3: team_batting_2b, 4: team_batting_3b, 5: team_batting_hr,
#   6: team_batting_bb, 7: team_batting_so, 8: team_baserun_sb,
#   9: team_baserun_cs
```

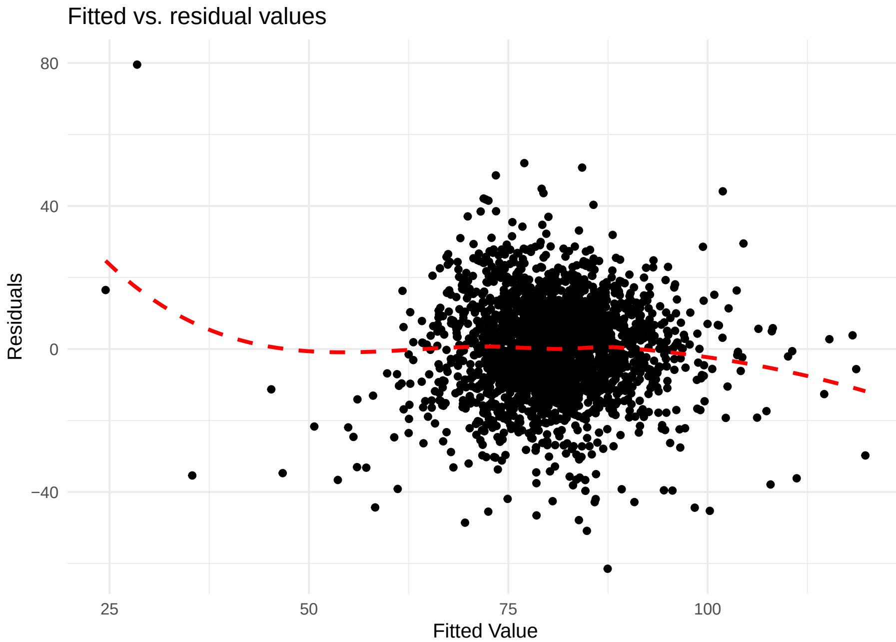
Models

```r
##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_pi
                 team_pitching_bb + team_baserun_sb + team_batting_hbp + team_baserun_cs,
tidy(lm_fit_1)
```

```
# A tibble: 10 x 5
   term              estimate std.error statistic  p.value
   <chr>                <dbl>     <dbl>     <dbl>    <dbl>
 1 (Intercept)      -0.634      5.73      -0.111  9.12e- 1
 2 team_batting_h    0.0374     0.00309   12.1    1.10e-32
 3 team_batting_2b  -0.000325   0.00899   -0.0362 9.71e- 1
 4 team_batting_bb   0.0329     0.00306   10.7    2.57e-26
 5 team_pitching_hr  0.0457     0.00718    6.37   2.30e-10
 6 team_batting_3b   0.0623     0.0165     3.78   1.60e- 4
 7 team_pitching_bb -0.00821    0.00204   -4.02   6.02e- 5
 8 team_baserun_sb   0.0217     0.00408    5.32   1.16e- 7
 9 team_batting_hbp  0.0473     0.0765     0.618  5.37e- 1
10 team_baserun_cs   0.0183     0.0161     1.14   2.56e- 1
```

Let’s check the residuals vs. fitted plot.

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


Fitted vs. residual values
80
40
0
−40
Residuals
25
50
75
100
Fitted Value