# Task_1

Ny Dang, Yves-langston Mays, Uyen Vi Phan, Connie Yang

## Intro

Importing the data set from github repo and loading libraries

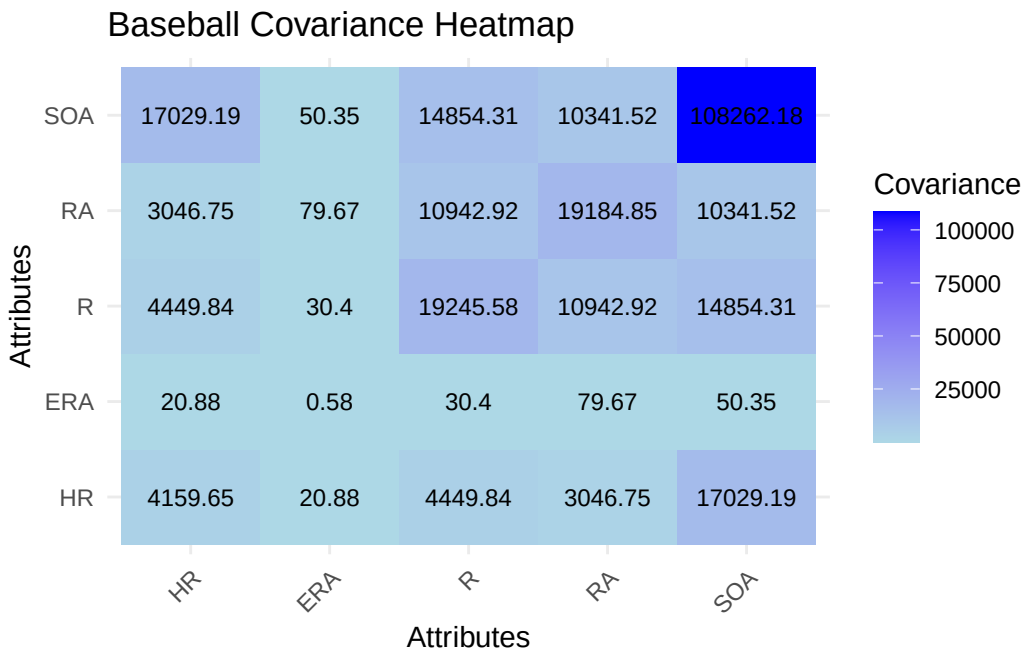
```
library(readr)
library(ggplot2)
library(reshape2)
library(corrplot)
library(dplyr)
library(tidyr)
library(ranger)
library(rpart)
library(rpart.plot)

url = "https://raw.githubusercontent.com/RaunakDune/3337_Fall2024_EDA/bf7b9ca17ed5e08ff25677

data = read.csv(url)
head(data, 3)
attach(data)
```

## 1. Covariance Matrices

Covariance matrices for HR, ERA, R, RA, SOA

```
selected_data = data[, c("HR", "ERA","R", "RA", "SOA")]
covariance_matrix = cov(selected_data)

cov_melted = melt(covariance_matrix)

ggplot(cov_melted, aes(Var1, Var2, fill = value)) +
```

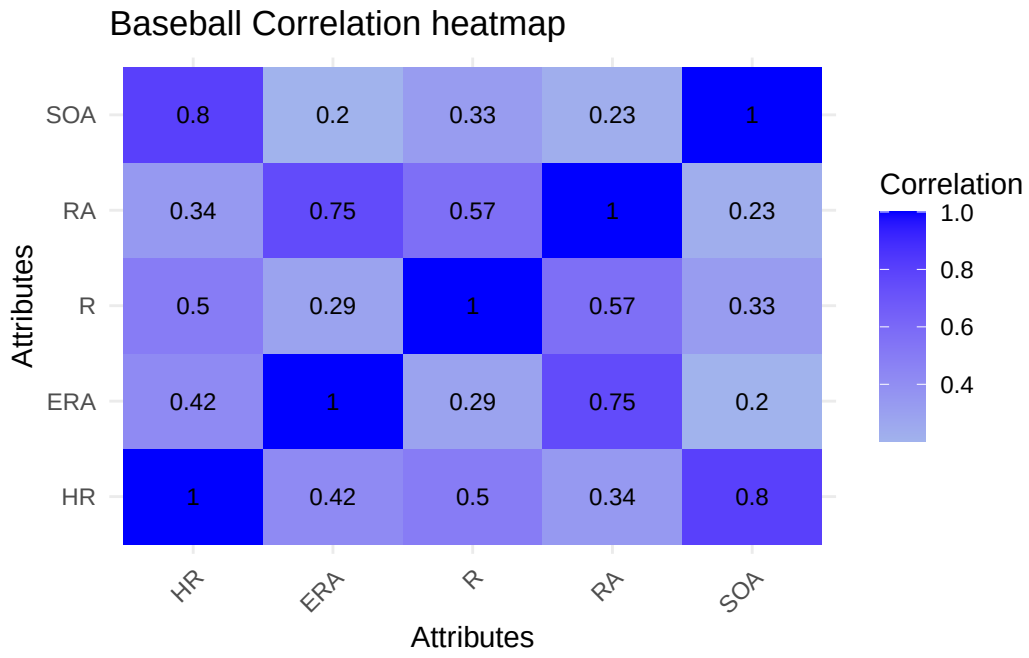
```
  geom_tile() +
  scale_fill_gradient2(low = "white", mid = "lightblue", high = "blue",
                       midpoint = 0, limit = c(min(cov_melted$value), max(cov_melted$value)))
  theme_minimal() +
  labs(title = "Baseball Covariance Heatmap", x = "Attributes", y = "Attributes", fill = "Cov
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  geom_text(aes(label = round(value, 2)), color = "black", size = 3)
```




## 1. Correlation Matrices

```
correlation_matrix = cor(selected_data)

cor_melted = melt(correlation_matrix)

ggplot(cor_melted, aes(Var1, Var2, fill = value)) +
  geom_tile() +
  scale_fill_gradient2(low = "white", mid = "lightblue", high = "blue",
                       limit = c(min(cor_melted$value), max(cor_melted$value))) +
  theme_minimal() +
  labs(title = "Baseball Correlation heatmap", x = "Attributes", y = "Attributes", fill = "Co
```

```
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  geom_text(aes(label = round(value, 2)), color = "black", size = 3)
```




## 1. Interpretation of Correlation and Covariance Matrices

SOA has a correlation of .80 with HR, suggesting that teams that strike out a lot of batters, also hit a lot of home runs. This could be because the strike outs are due to good pitchers, the same pitchers that batters will be practicing with, iron sharpening iron.

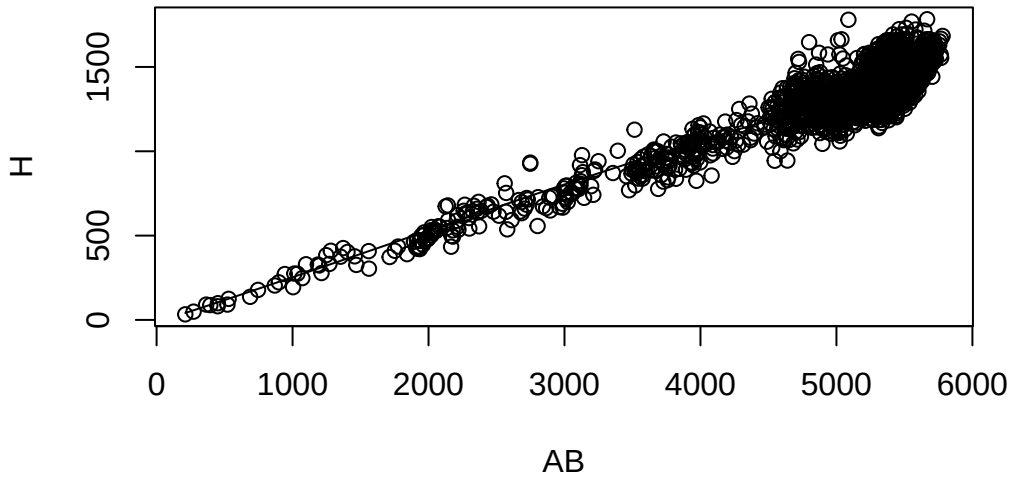
The next highest correlation is 0.75 between ERA and RA. This is expected since ERA is a measure of a pitcher's effectiveness. If the pitcher is less effective at striking batters out, meaning the pitcher has a higher ERA, then there will be more runs scored, and a higher RA.

Everything else is roughly 0.50 or less, and can be explained by domain knowledge.

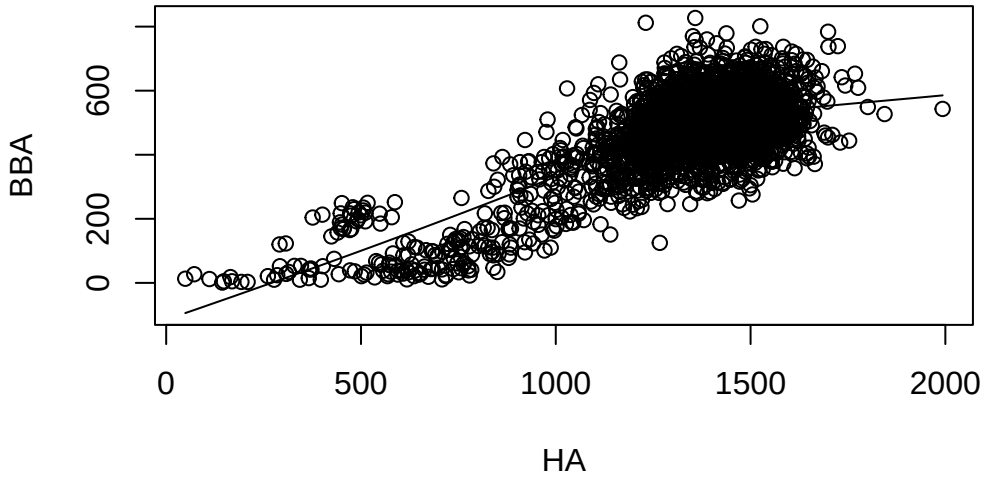
## 2. Scatter Plot for AB/H and HA/BBA

AB = At bats, H = hits by batters, HA = hits allows, BBA = walks allowed

```
scatter.smooth(AB, H)
```




```
scatter.smooth(HA, BBA)
```



At bats is a measure of how many times a team has had a batter at the plate. Hits tells us how many times a batter got a hit AND got to first base. If a batter hit the ball, and it was foul, it is not considered a hit. Only hits that result in placement on base count.

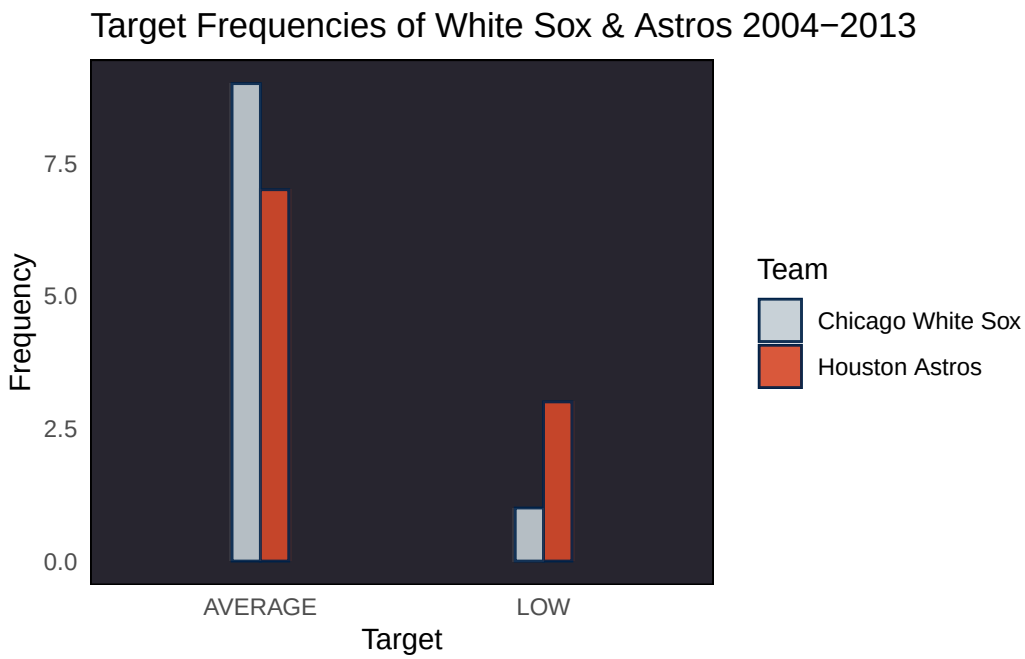
Based on our AB/H plot, there is a strong positive correlation between the two, which is to be expected. There seems to be more variability in clusters around 2000, 4000, and 4700+ at bats.

Walks allowed are recorded when a pitcher either hits a batter with the ball, or throws 4 balls outside of the strike zone without the batter swinging at them. Hits allowed is the offensive component of Hits. There seems to be a lot of variation between these two variables, though there seems to be a clear positive correlation with a large concentration at roughly between 1500 to 1750 hits allowed.

## 3. Two Teams + Astros Histograms

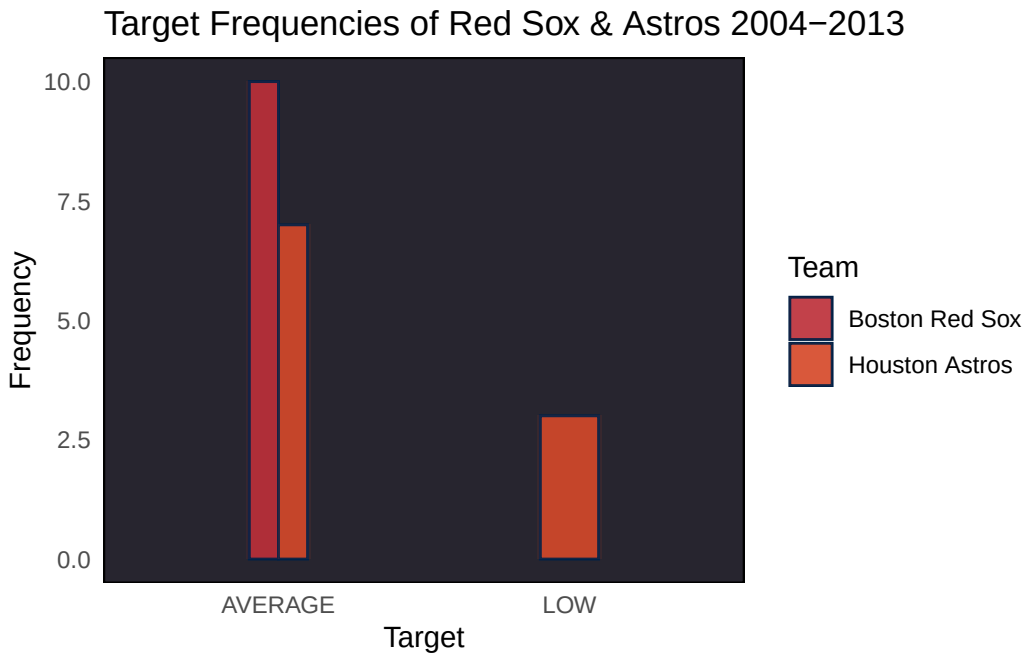
I will use the White Sox, Red Sox, and Astros. The histograms will be for the years 2004-2013 and 2014-2023. I will measure the High, Low, and Average.

```
team_hist_data_2013_Boston = data %>%
  filter(yearID <= 2013 & yearID >= 2004 & (name == "Boston Red Sox" | name == "Houston Astro
  select( name, TARGET)


team_hist_data_2013_Chicago = data %>%
  filter(yearID <= 2013 & yearID >= 2004 & (name == "Chicago White Sox" | name == "Houston As
  select(name, TARGET)


ggplot(team_hist_data_2013_Chicago, aes(x = TARGET, fill = name)) +
  geom_bar(position = "dodge", color = "#002147", width = 0.2, alpha = 0.9) +
  scale_fill_manual(values = c("Chicago White Sox" = "#C4CED4", "Houston Astros" = "#D6492A"
  labs(x = "Target", y = "Frequency", title = "Target Frequencies of White Sox & Astros 2004-
  labs(fill = "Team") +
  theme_minimal() +
    theme(
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    panel.background = element_rect(fill = "#27252F")
    ) +
    geom_bar(aes(x = TARGET, fill = name), position = "dodge", width = 0.22, alpha = 0.1)
```




The White Sox had more average targets than the Astros and fewer low targets as well.

```
ggplot(team_hist_data_2013_Boston, aes(x = TARGET, fill = name)) +
  geom_bar(position = "dodge", color = "#002147", width = 0.2, alpha = 0.9) +
  scale_fill_manual(values = c("Boston Red Sox" = "#BD3039", "Houston Astros" = "#D6492A")) +
  labs(x = "Target", y = "Frequency", title = "Target Frequencies of Red Sox & Astros 2004-20
  labs(fill = "Team") +
  theme_minimal() +
    theme(
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    panel.background = element_rect(fill = "#27252F")
    ) +
    geom_bar(aes(x = TARGET, fill = name), position = "dodge", width = 0.22, alpha = 0.1)
```




The Red Sox had no low season targets within this 10 year period, and had higher average targets while the Astros had mixed seasons.

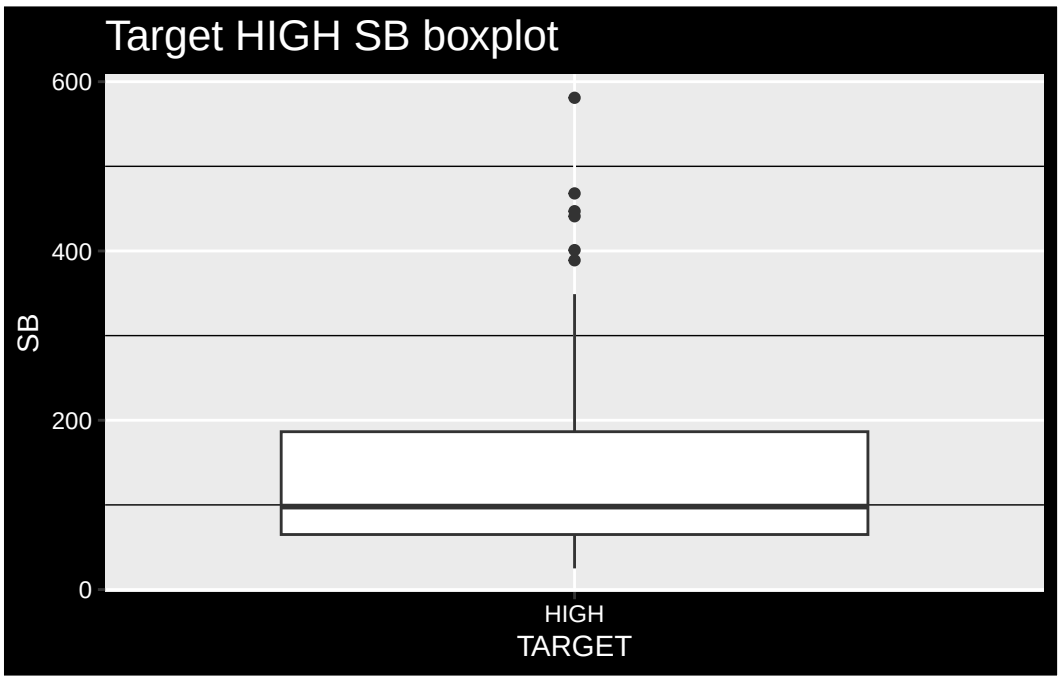
## 4. Box plots for BB and SB

I will create box plots for BB and SB, each with TARGET ave, low, hi, and then each with the whole data set.

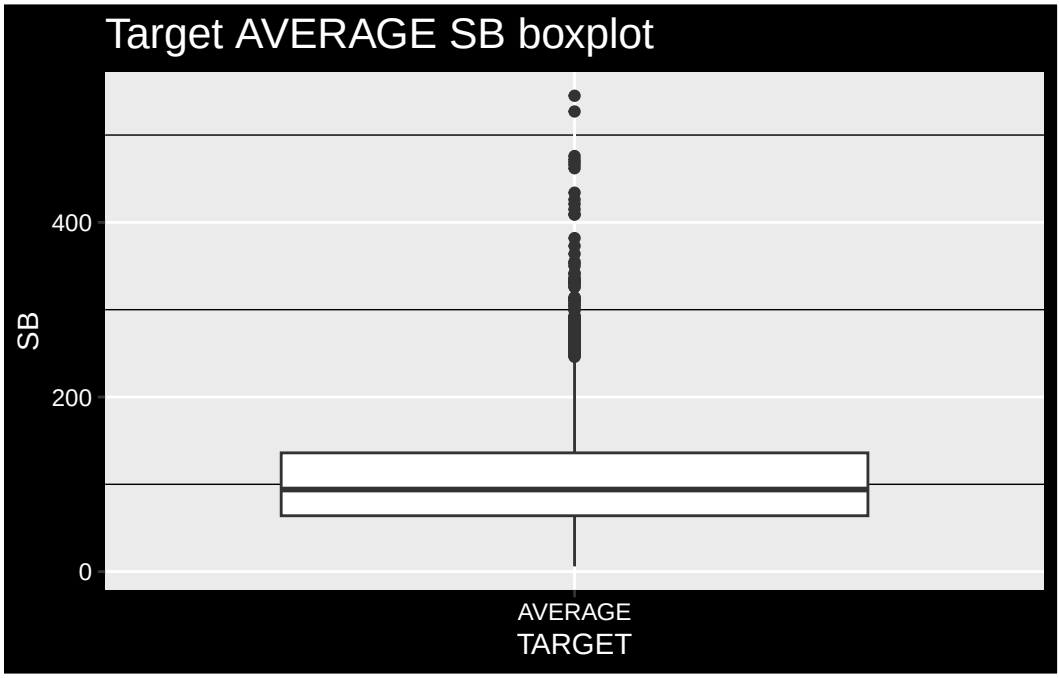
BB is walks by batters, while SB is stolen bases.

```
box_sb_hi = data %>%
  filter(TARGET == "HIGH") %>%
  select(TARGET, SB)

box_sb_ave = data %>%
  filter(TARGET == "AVERAGE") %>%
  select(TARGET, SB)

box_sb_low = data %>%
  filter(TARGET == "LOW") %>%
  select(TARGET, SB)
```

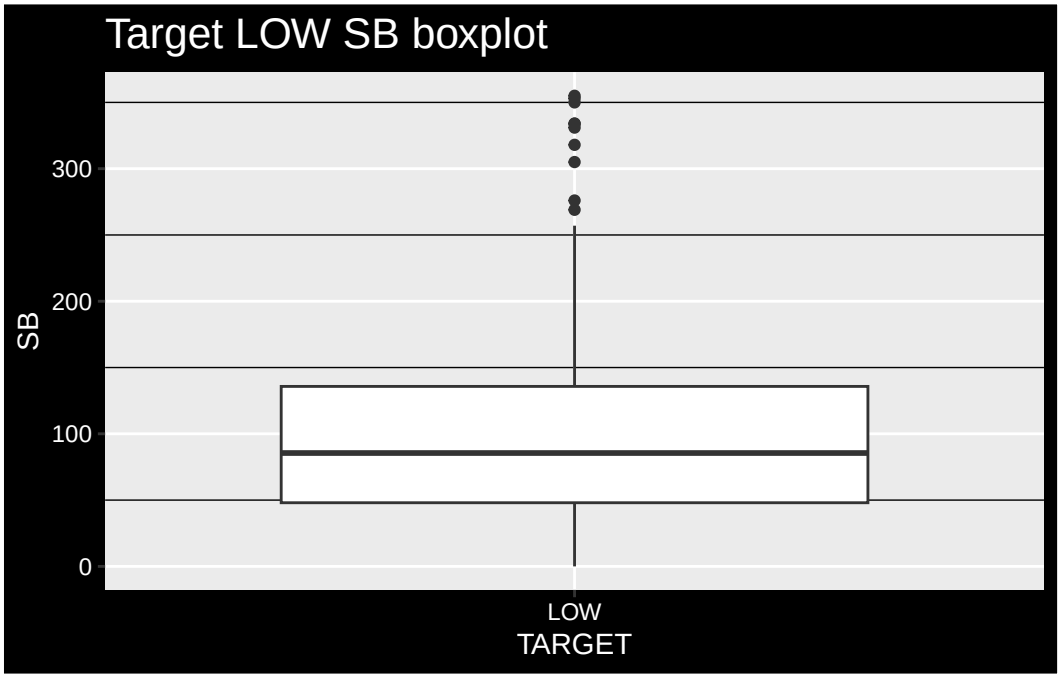
```r
box_bb_hi = data %>%
  filter(TARGET == "HIGH") %>%
  select(TARGET, BB)

box_bb_ave = data %>%
  filter(TARGET == "AVERAGE") %>%
  select(TARGET, BB)

box_bb_low = data %>%
  filter(TARGET == "LOW") %>%
  select(TARGET, BB)


box_bb_all = data %>%
  select(TARGET, BB) %>%
  arrange(TARGET)

box_sb_all = data %>%
  select(TARGET, SB) %>%
  arrange(TARGET)


ggplot(box_sb_hi, aes(x = TARGET, y = SB)) +
  geom_boxplot() + labs(title ="Target HIGH SB boxplot") +
  theme(
    panel.grid.minor = element_line(color = "black"),
    plot.background = element_rect(fill = "black"),
    axis.text = element_text(color = "white"),
    axis.title = element_text(color = "white"),
    plot.title = element_text(color = "white", size = 16),
    legend.text = element_text(color = "white")
  )
```



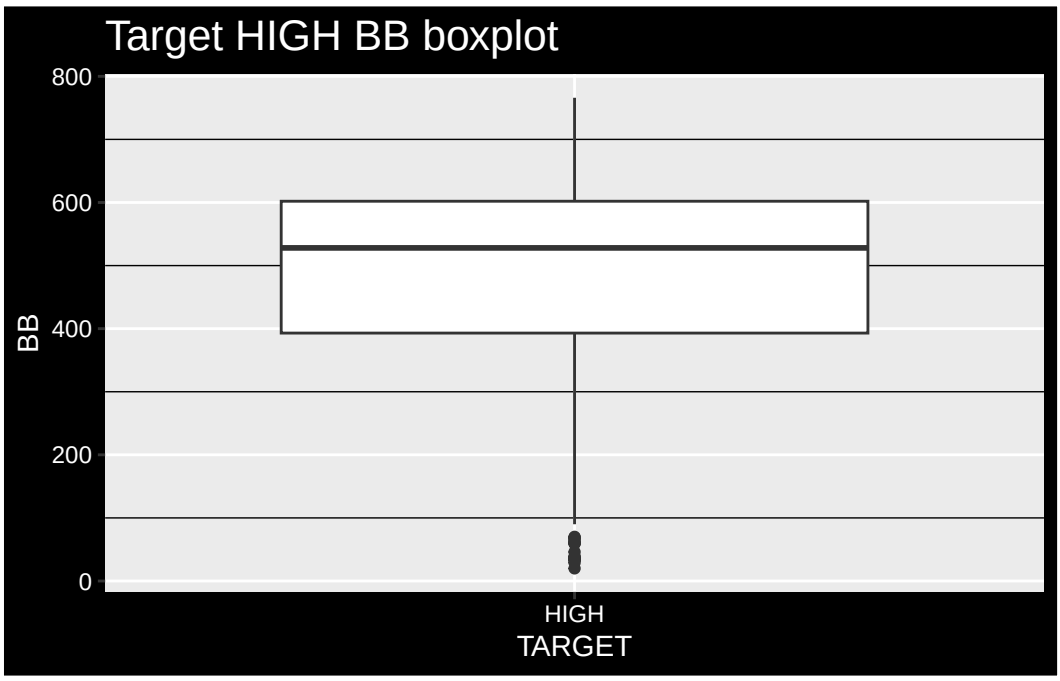

```
ggplot(box_sb_ave, aes(x = TARGET, y = SB)) +
  geom_boxplot() +
  labs(title ="Target AVERAGE SB boxplot") +
  theme(
    panel.grid.minor = element_line(color = "black"),
    plot.background = element_rect(fill = "black"),
    axis.text = element_text(color = "white"),
    axis.title = element_text(color = "white"),
    plot.title = element_text(color = "white", size = 16),
    legend.text = element_text(color = "white")
  )
```



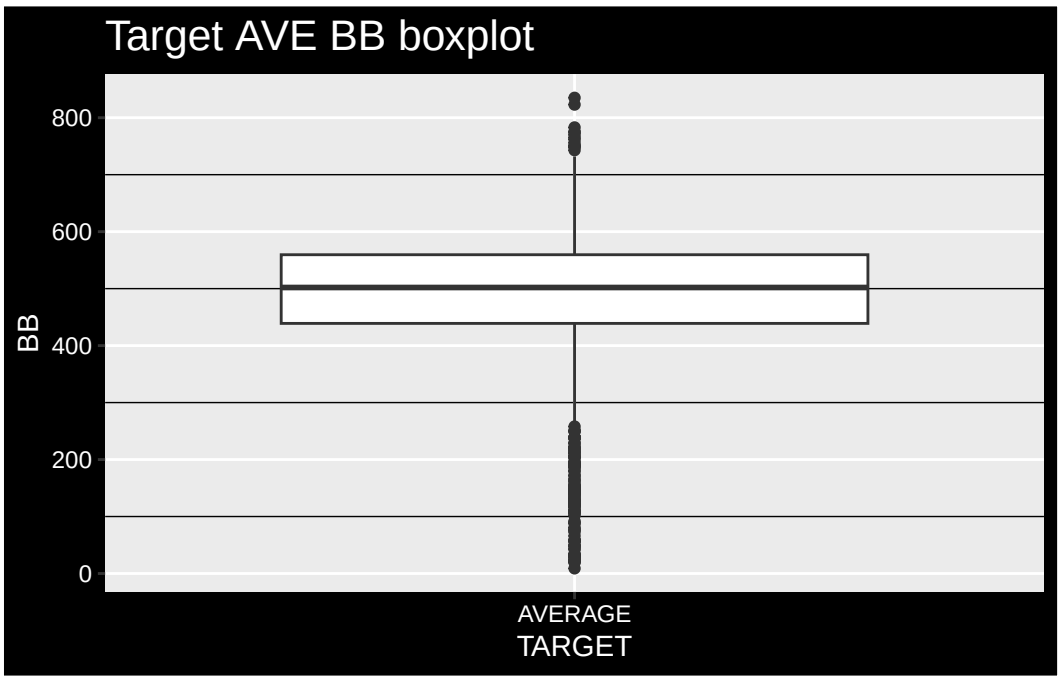

```
ggplot(box_sb_low, aes(x = TARGET, y = SB)) +
  geom_boxplot() +
  labs(title ="Target LOW SB boxplot") +
  theme(
    panel.grid.minor = element_line(color = "black"),
    plot.background = element_rect(fill = "black"),
    axis.text = element_text(color = "white"),
    axis.title = element_text(color = "white"),
    plot.title = element_text(color = "white", size = 16),
    legend.text = element_text(color = "white")
  )
```



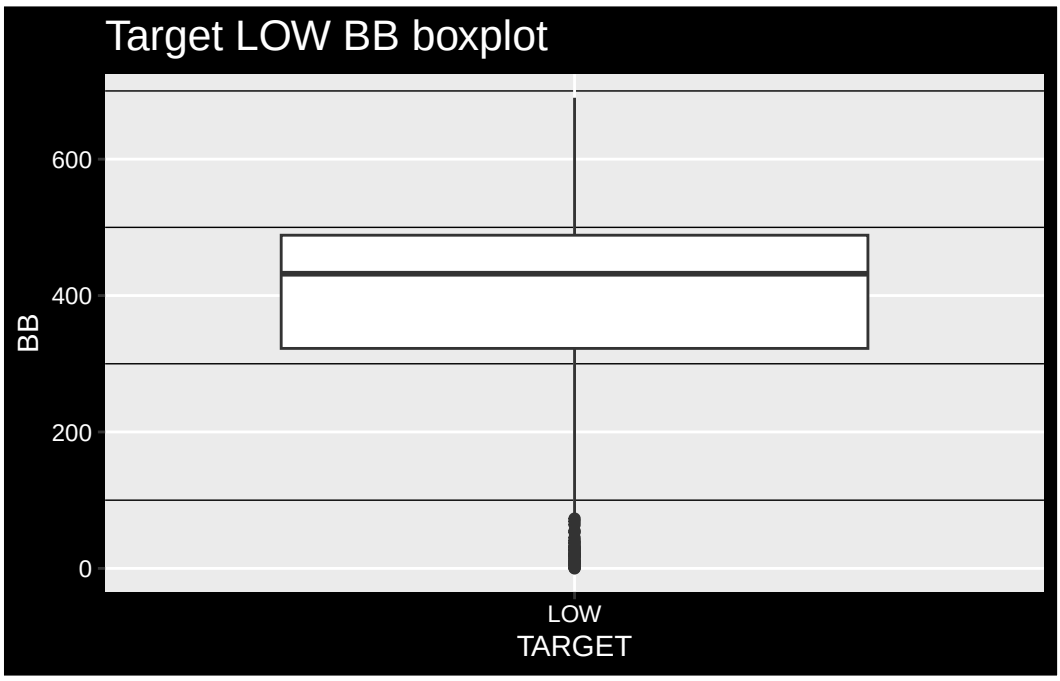

```
ggplot(box_bb_hi, aes(x = TARGET, y = BB)) +
  geom_boxplot() + labs(title ="Target HIGH BB boxplot") +
  theme(
    panel.grid.minor = element_line(color = "black"),
    plot.background = element_rect(fill = "black"),
    axis.text = element_text(color = "white"),
    axis.title = element_text(color = "white"),
    plot.title = element_text(color = "white", size = 16),
    legend.text = element_text(color = "white")
  )
```



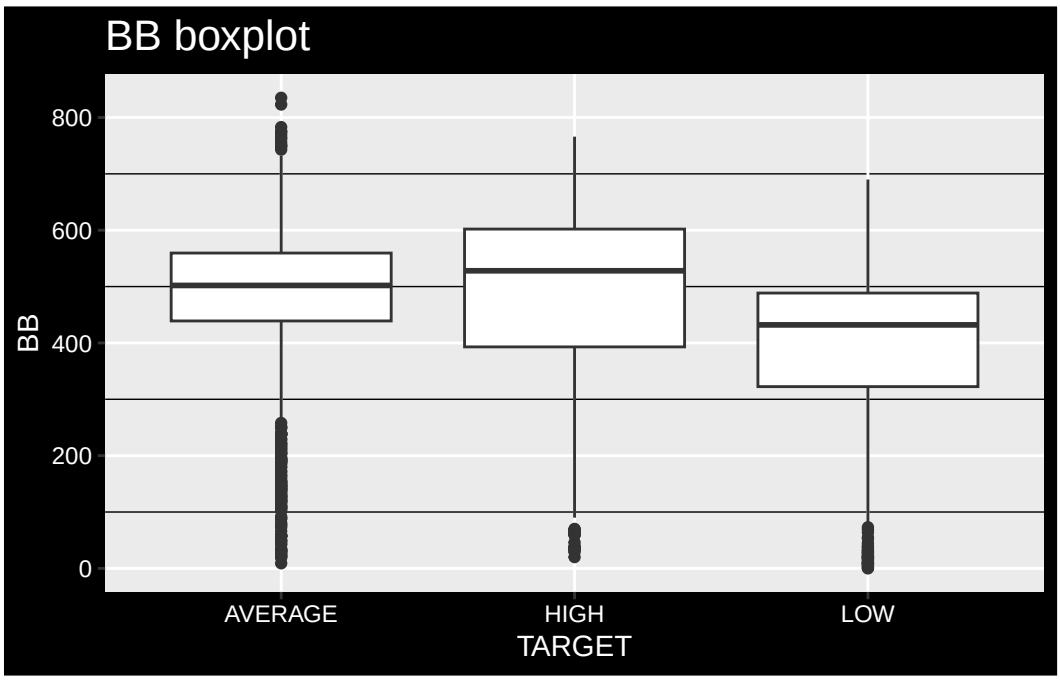

```
ggplot(box_bb_ave, aes(x = TARGET, y = BB)) +
  geom_boxplot() + labs(title ="Target AVE BB boxplot") +
  theme(
    panel.grid.minor = element_line(color = "black"),
    plot.background = element_rect(fill = "black"),
    axis.text = element_text(color = "white"),
    axis.title = element_text(color = "white"),
    plot.title = element_text(color = "white", size = 16),
    legend.text = element_text(color = "white")
  )
```



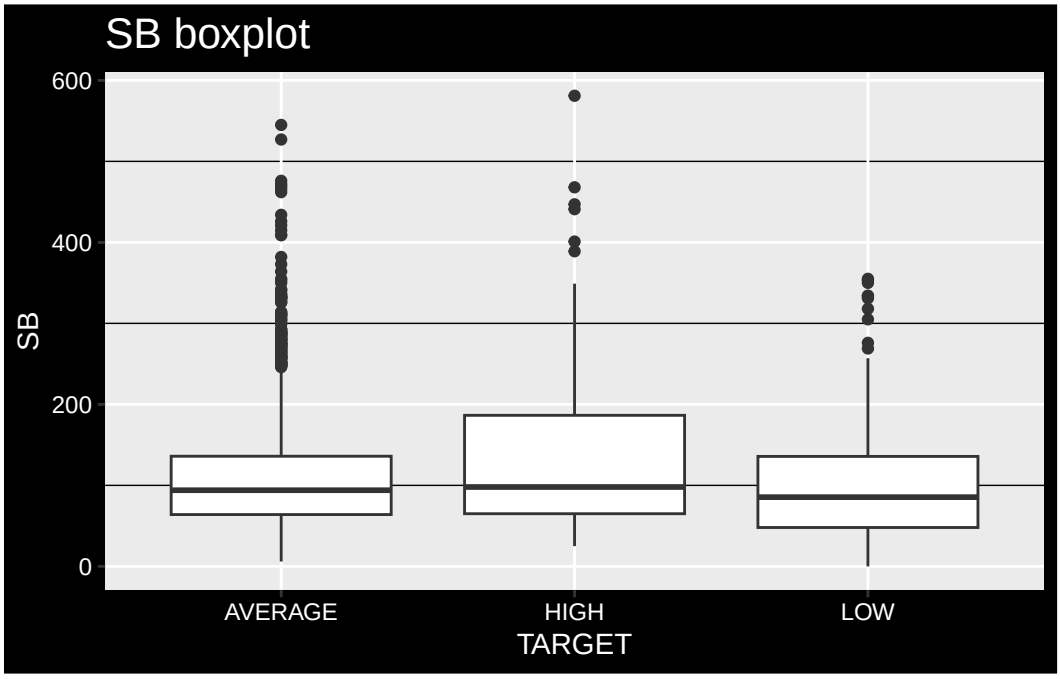

```
ggplot(box_bb_low, aes(x = TARGET, y = BB)) +
  geom_boxplot() + labs(title ="Target LOW BB boxplot") +
  theme(
    panel.grid.minor = element_line(color = "black"),
    plot.background = element_rect(fill = "black"),
    axis.text = element_text(color = "white"),
    axis.title = element_text(color = "white"),
    plot.title = element_text(color = "white", size = 16),
    legend.text = element_text(color = "white")
  )
```




```
ggplot(box_bb_all, aes(x = TARGET, y = BB)) +
  geom_boxplot() + labs(title ="BB boxplot") +
  theme(
    panel.grid.minor = element_line(color = "black"),
    plot.background = element_rect(fill = "black"),
    axis.text = element_text(color = "white"),
    axis.title = element_text(color = "white"),
    plot.title = element_text(color = "white", size = 16),
    legend.text = element_text(color = "white")
  )
```




```
ggplot(box_sb_all, aes(x = TARGET, y = SB)) +
  geom_boxplot() + labs(title ="SB boxplot") +
  theme(
    panel.grid.minor = element_line(color = "black"),
    plot.background = element_rect(fill = "black"),
    axis.text = element_text(color = "white"),
    axis.title = element_text(color = "white"),
    plot.title = element_text(color = "white", size = 16),
    legend.text = element_text(color = "white")
  )
```




For seasons with the HIGH target, stolen bases seems to reach its median at roughly 100 stolen bases, which is slightly above the AVERAGE target stolen bases median. The average target seasons seem to have a much tighter distribution and far more outliers than the others.

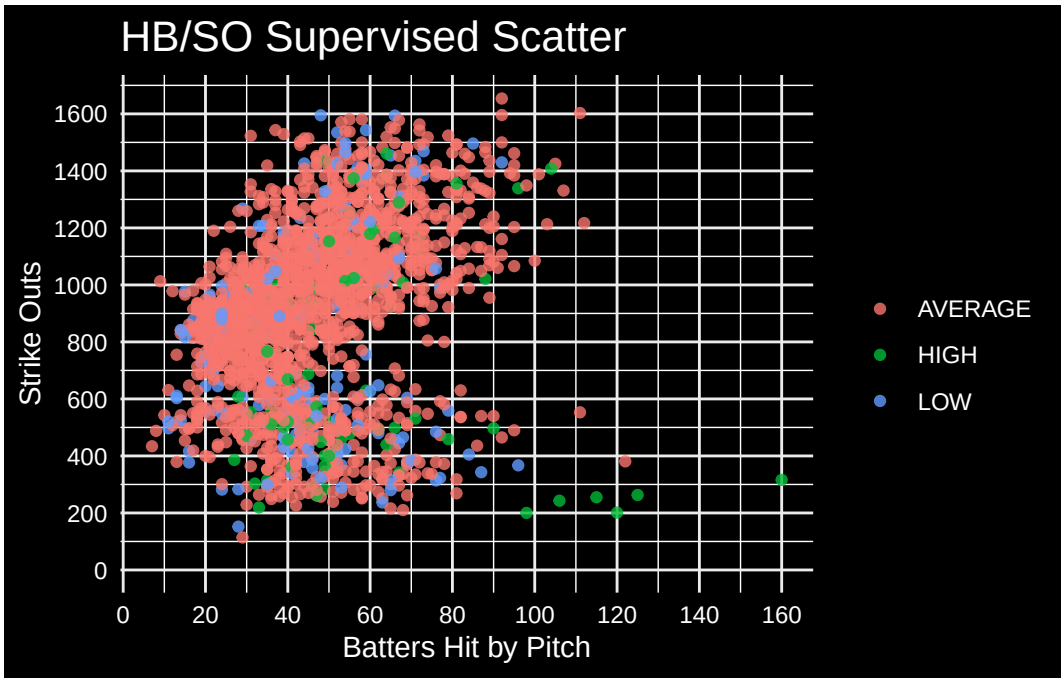
The average target seasons seemed to have a very tight distribution with a median barely above 500. The low target seasons had a median of 425 and the high 525. The latter two also had much wider distributions. There are very few outliers for the high target group.

On average, BB seems to have a fairly wide distribution with the fewest amount of outliers, all being below the lower quartile by a significant difference. The Average seems to vary widely, with lots of outliers.

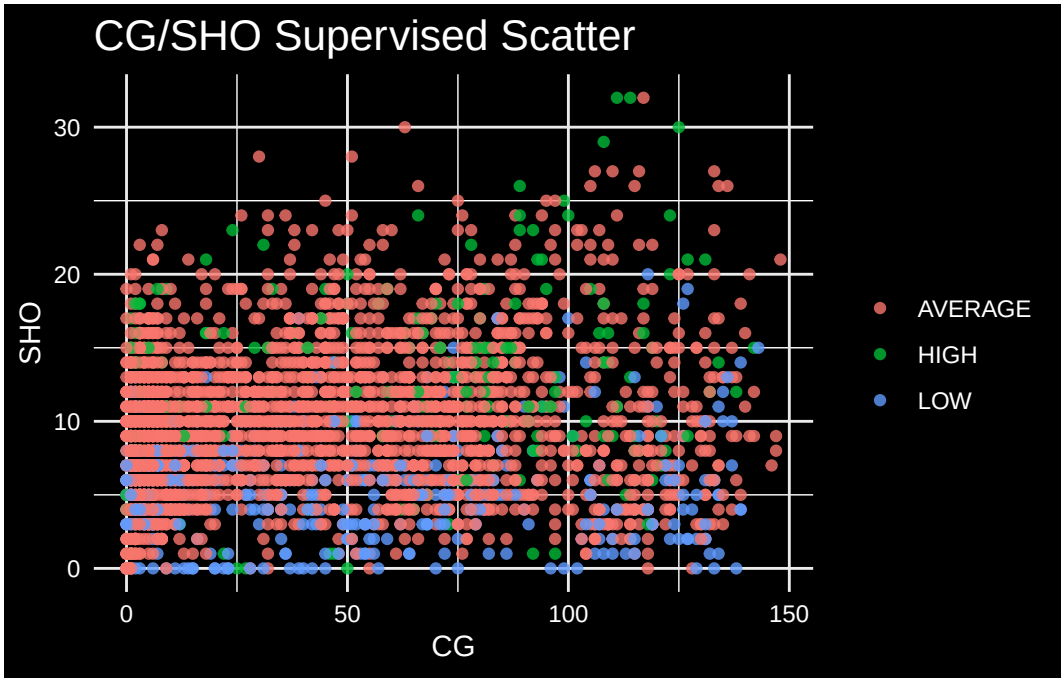
## 5. Supervised Scatterplot for HB/SO, CG/SHO, IPOuts/DP

```
supervised_data = data %>%
  select(HBP, SO, CG, SHO, IPouts, DP, TARGET)

ggplot(supervised_data, aes(x = HBP, y = SO, color = TARGET)) +
  geom_point(alpha = 0.8) +
  labs(title = "HB/SO Supervised Scatter", y = "Strike Outs", x = "Batters Hit by Pitch") +
  theme_minimal() +
    theme(
```

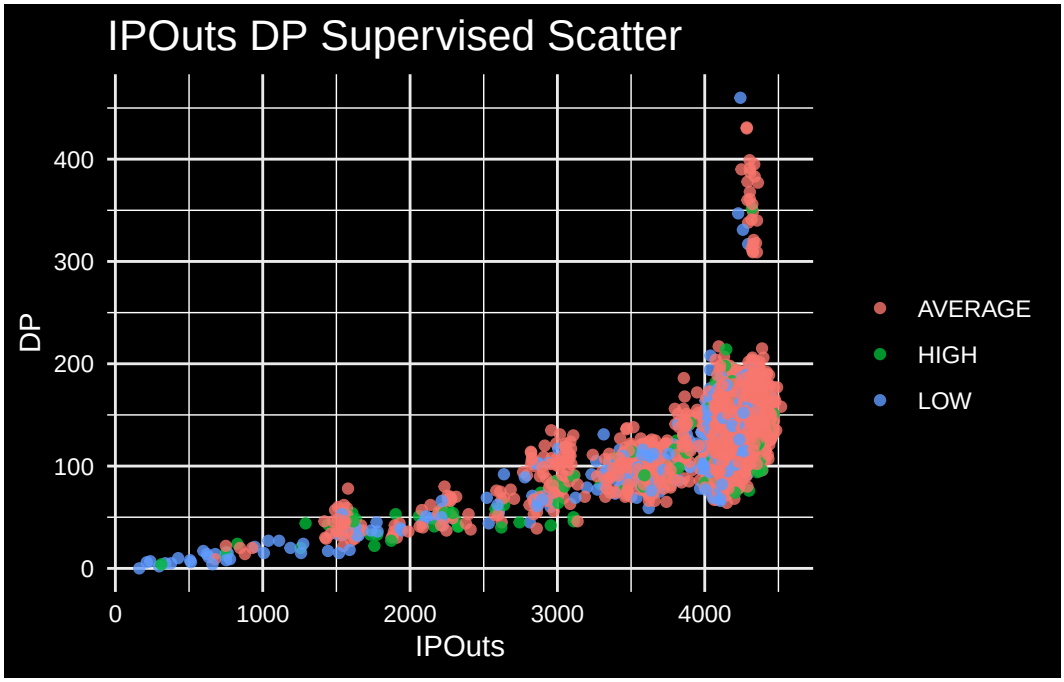
```
    panel.grid.minor = element_line(color = "white"),
    plot.background = element_rect(fill = "black"),
    axis.text = element_text(color = "white"),
    axis.title = element_text(color = "white"),
    plot.title = element_text(color = "white", size = 16),
    legend.text = element_text(color = "white")
  ) +
  scale_x_continuous(breaks = seq(0, 200, 20)) +
  scale_y_continuous(breaks = seq(0, 2000, 200))
```




```
ggplot(supervised_data, aes(x = CG, y = SHO, color = TARGET)) +
  geom_point(alpha = 0.8) +
  labs(title = "CG/SHO Supervised Scatter", y = "SHO", x = "CG") +
  theme_minimal() +
  theme(
    panel.grid.minor = element_line(color = "white"),
    plot.background = element_rect(fill = "black"),
    axis.text = element_text(color = "white"),
    axis.title = element_text(color = "white"),
    plot.title = element_text(color = "white", size = 16),
    legend.text = element_text(color = "white")
  )
```




```
ggplot(supervised_data, aes(x = IPouts, y = DP, color = TARGET)) +
  geom_point(alpha = 0.8) +
  labs(title = "IPOuts DP Supervised Scatter", y = "DP", x = "IPOuts") +
  theme_minimal() +
  theme(
    panel.grid.minor = element_line(color = "white"),
    plot.background = element_rect(fill = "black"),
    axis.text = element_text(color = "white"),
    axis.title = element_text(color = "white"),
    plot.title = element_text(color = "white", size = 16),
    legend.text = element_text(color = "white")
  )
```



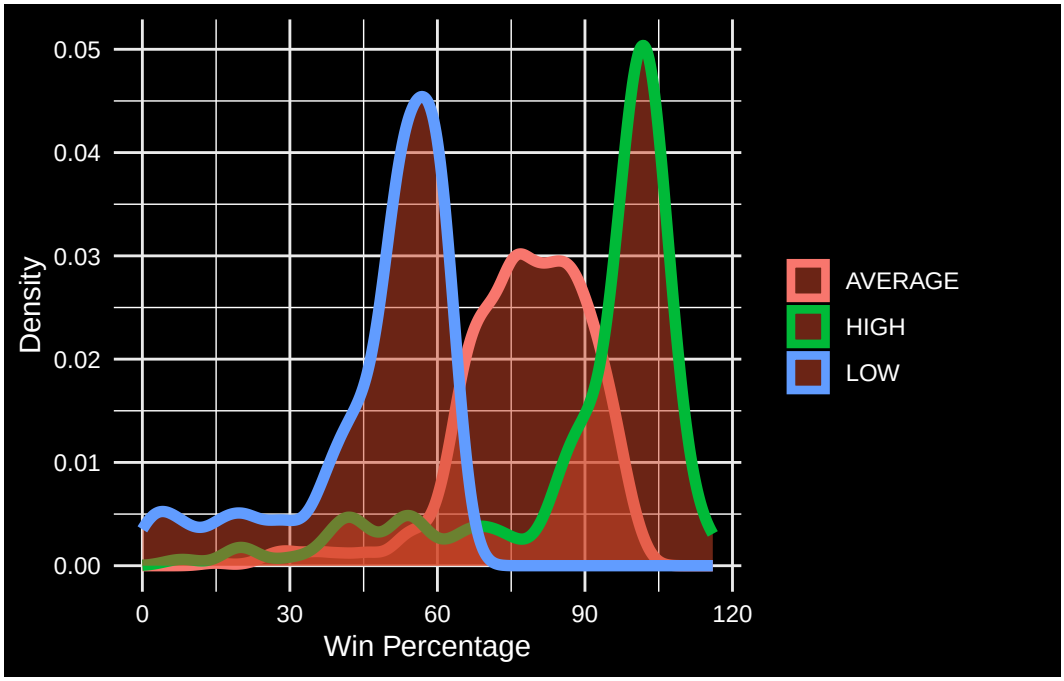

Predicting target, based on the above scatterplots, will be very difficult. Each target is fairly well distributed through the x and y axes. The HB/SO plot seems to follow a curved C shaped pattern, though it might not indicate any relationship. The CG/SHO plot has its points evenly distributed from 0-75 CG and 0-15 SHO, the points begin to become farther apart. It may be slightly easier to predict using the IPOuts and DP plot, with the exception of values above 4200 where the values become more unpredictable.

## 6. Density Plots W v E

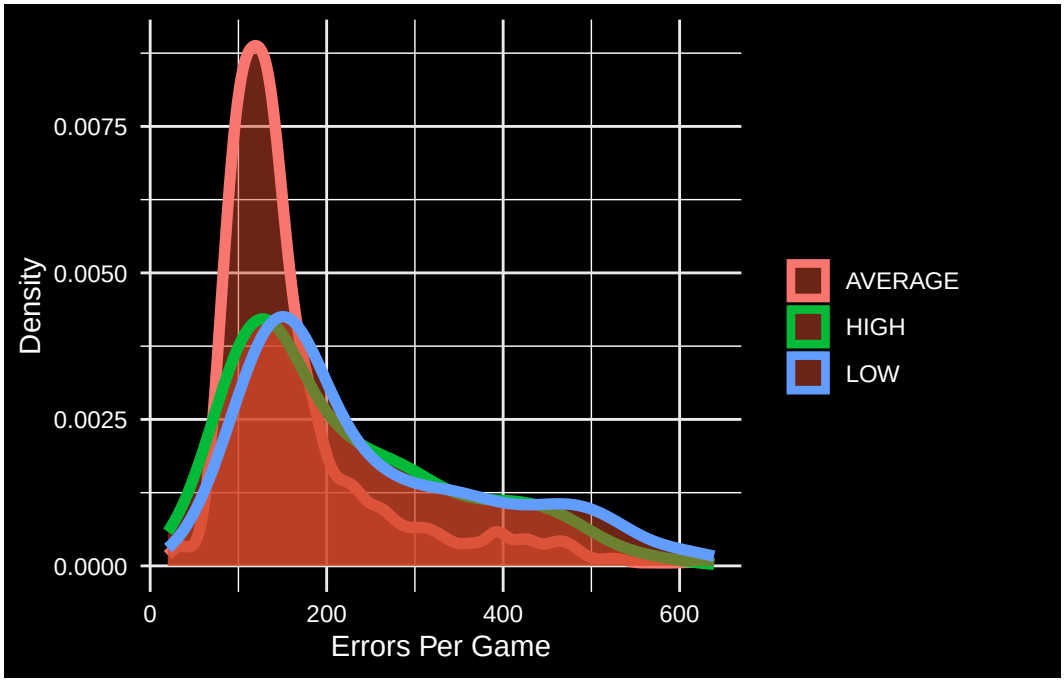
We will view density plots against Win Percentage W, and Errors per game E

```
density_data = data %>%
  select(W, E, TARGET)

ggplot(density_data, aes(x = W, color = as.factor(TARGET))) +
  geom_density(fill = "#D6492A", alpha = 0.5, size = 2) +
  theme_minimal() +
  labs(x = "Win Percentage", y = "Density") +
  theme(
    panel.grid.minor = element_line(color = "white"),
    plot.background = element_rect(fill = "black"),
    axis.text = element_text(color = "white"),
```

```
    axis.title = element_text(color = "white"),
    plot.title = element_text(color = "white", size = 16),
    legend.text = element_text(color = "white")
  )
```




```
ggplot(density_data, aes(x = E, color = as.factor(TARGET))) +
  geom_density(fill = "#D6492A", alpha = 0.5, size = 2) +
  theme_minimal() +
  labs(x = "Errors Per Game", y = "Density") +
  theme(
    panel.grid.minor = element_line(color = "white"),
    plot.background = element_rect(fill = "black"),
    axis.text = element_text(color = "white"),
    axis.title = element_text(color = "white"),
    plot.title = element_text(color = "white", size = 16),
    legend.text = element_text(color = "white")
  )
```




There seems to be a very small difference in density for E when comparing high and low Target's. Maybe because of more conservative play? If a variable can be engineered that represents the aggressiveness of a team, it can be used to validate that hypothesis. The win percentage density is as expected, a bimodal density plot where the average is in the middle of both peaks.

## 7. World Series Wins Table

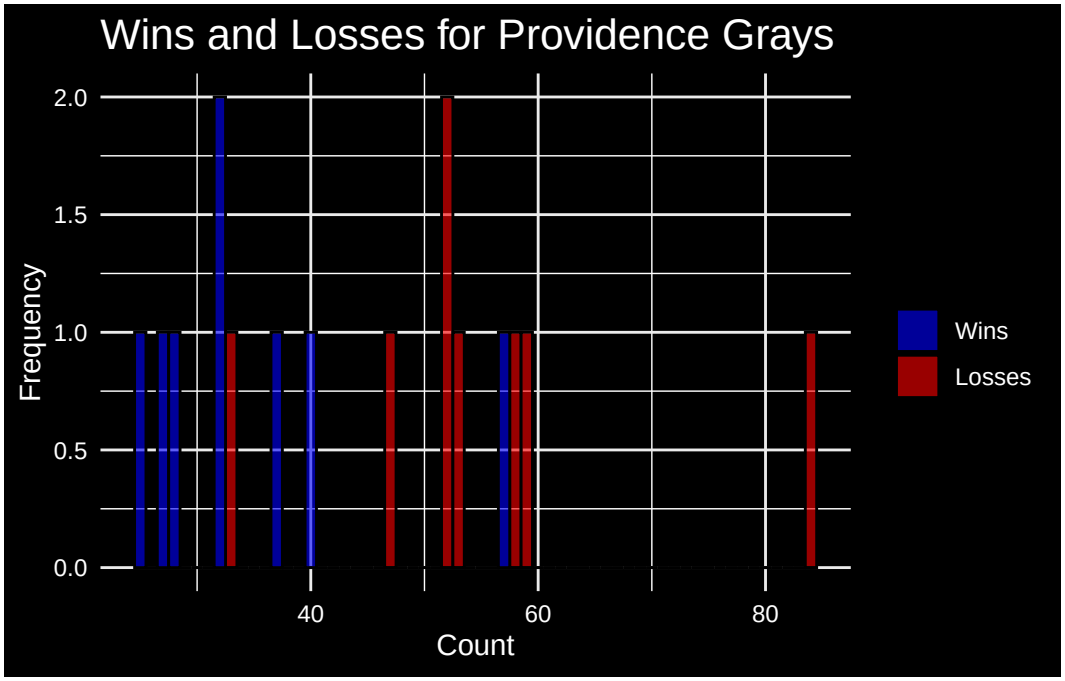
```
table(name[WSWin=="Y"], TARGET[WSWin=="Y"])
```

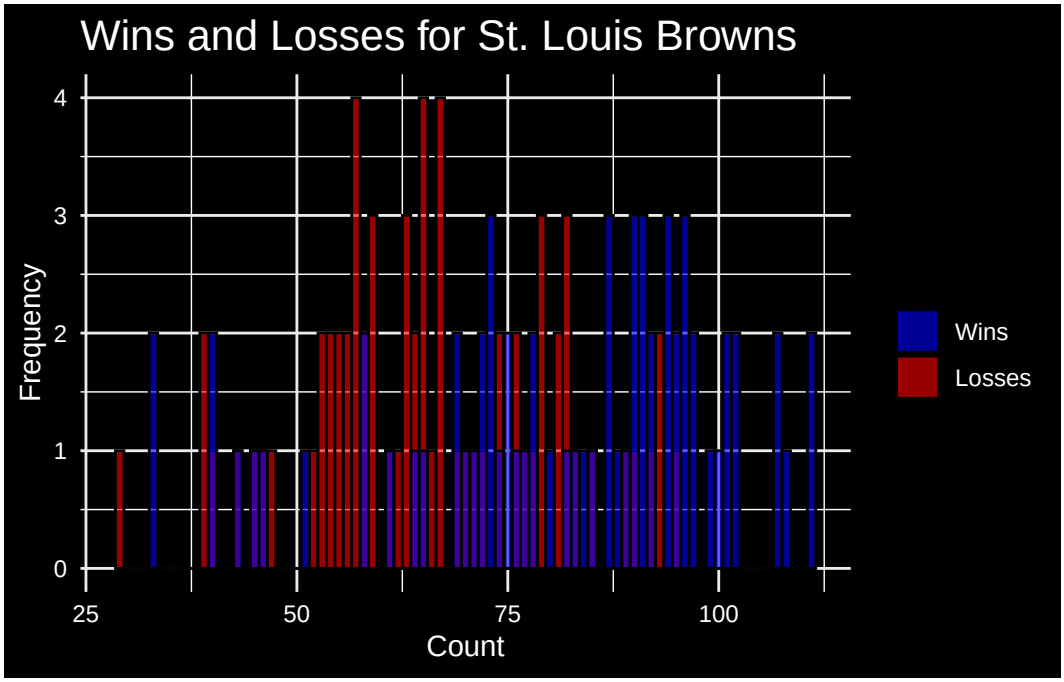
```
                        AVERAGE HIGH
  Anaheim Angels              1    0
  Arizona Diamondbacks        1    0
  Atlanta Braves              1    1
  Baltimore Orioles           2    1
  Boston Americans            0    1
  Boston Braves               1    0
  Boston Red Sox              5    3
  Brooklyn Dodgers            0    1
```

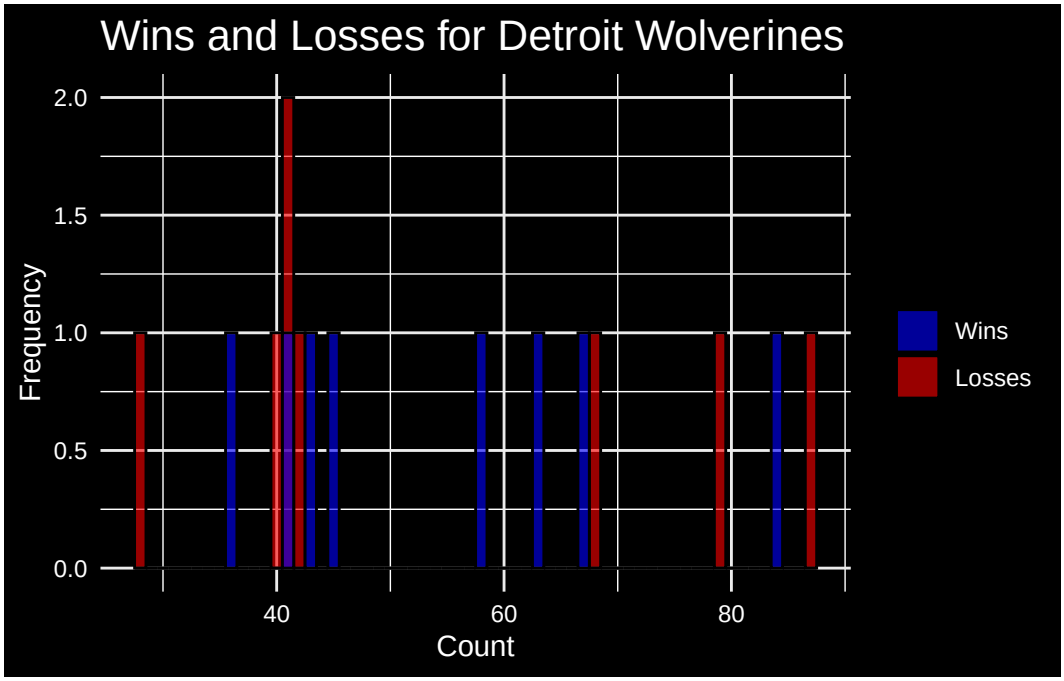
```
  Chicago Cubs                0    3
  Chicago White Sox           2    1
  Cincinnati Reds             1    4
  Cleveland Indians           1    1
  Detroit Tigers              2    2
  Detroit Wolverines          1    0
  Florida Marlins             2    0
  Houston Astros              1    1
  Kansas City Royals          2    0
  Los Angeles Dodgers         5    1
  Milwaukee Braves            1    0
  Minnesota Twins             2    0
  New York Giants             4    3
  New York Mets               1    1
  New York Yankees            8   19
  Oakland Athletics           4    0
  Philadelphia Athletics      0    5
  Philadelphia Phillies       2    0
  Pittsburgh Pirates          4    1
  Providence Grays            0    1
  San Francisco Giants        3    0
  St. Louis Browns            0    1
  St. Louis Cardinals         6    5
  Texas Rangers               1    0
  Toronto Blue Jays           2    0
  Washington Nationals        1    0
  Washington Senators         1    0
```

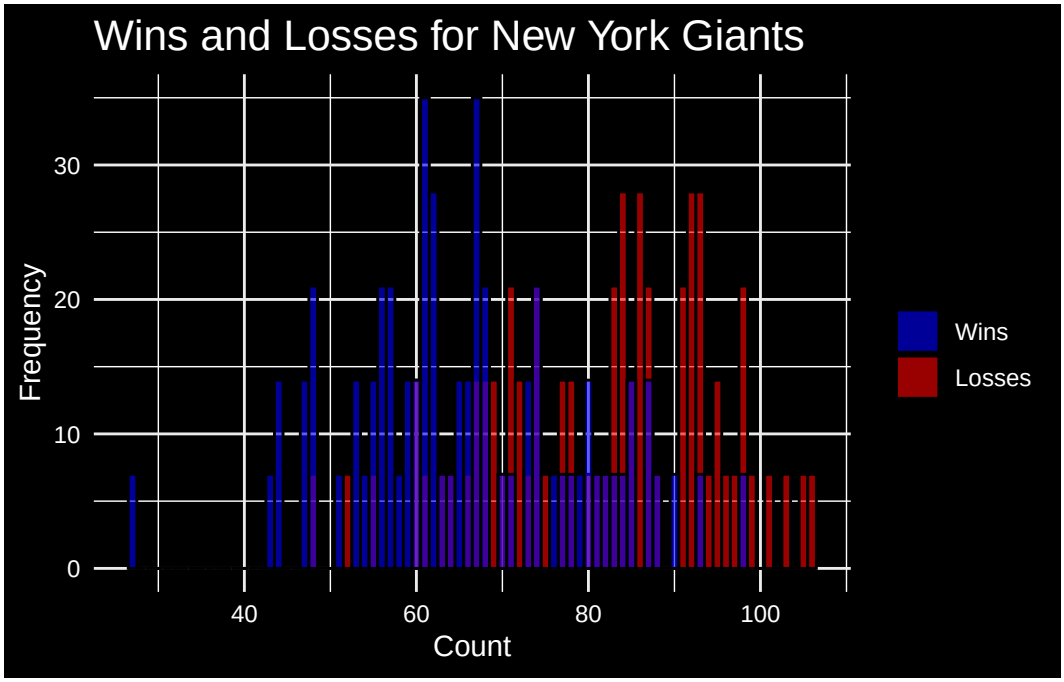
```r
world_series <- data %>%
  filter(WSWin == "Y") %>%
  select(name, WSWin) %>%
  mutate(
    LOW = sum(TARGET == "LOW"),
    AVERAGE = sum(TARGET == "AVERAGE"),
    HIGH = sum(TARGET == "HIGH")
  )


world_series = world_series %>%
  left_join(data %>% select(name, W, L), by = "name")
```

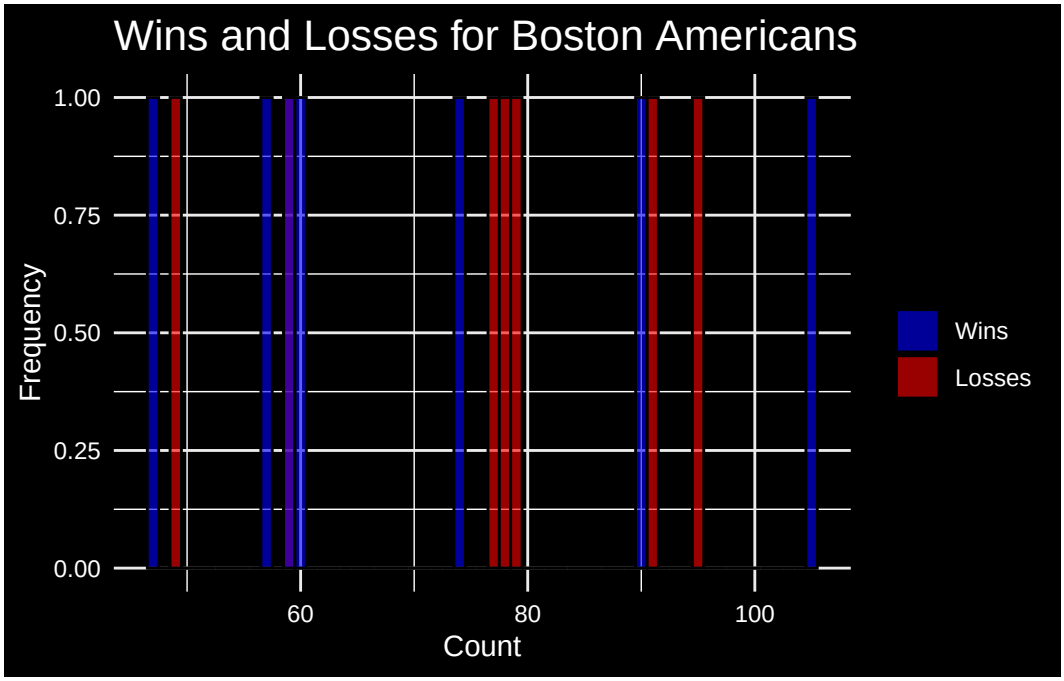
```r
teams = unique(world_series$name)

for (team in teams) {
  team_data = world_series %>% filter(name == team)

  looped_plots = ggplot(team_data) +
    geom_histogram(aes(x = W, fill = "Wins"), binwidth = 1, alpha = 0.6, color = "black") +
    geom_histogram(aes(x = L, fill = "Losses"), binwidth = 1, alpha = 0.6, color = "black") +
    scale_fill_manual(values = c("blue", "red"), name = "Type", labels = c("Wins", "Losses")) +
    labs(x = "Count", y = "Frequency", title = paste("Wins and Losses for", team)) +
    theme_minimal() +
    theme(
    panel.grid.minor = element_line(color = "white"),
    plot.background = element_rect(fill = "black"),
    axis.text = element_text(color = "white"),
    axis.title = element_text(color = "white"),
    plot.title = element_text(color = "white", size = 16),
    legend.text = element_text(color = "white")
  )

  print(looped_plots)
}
```

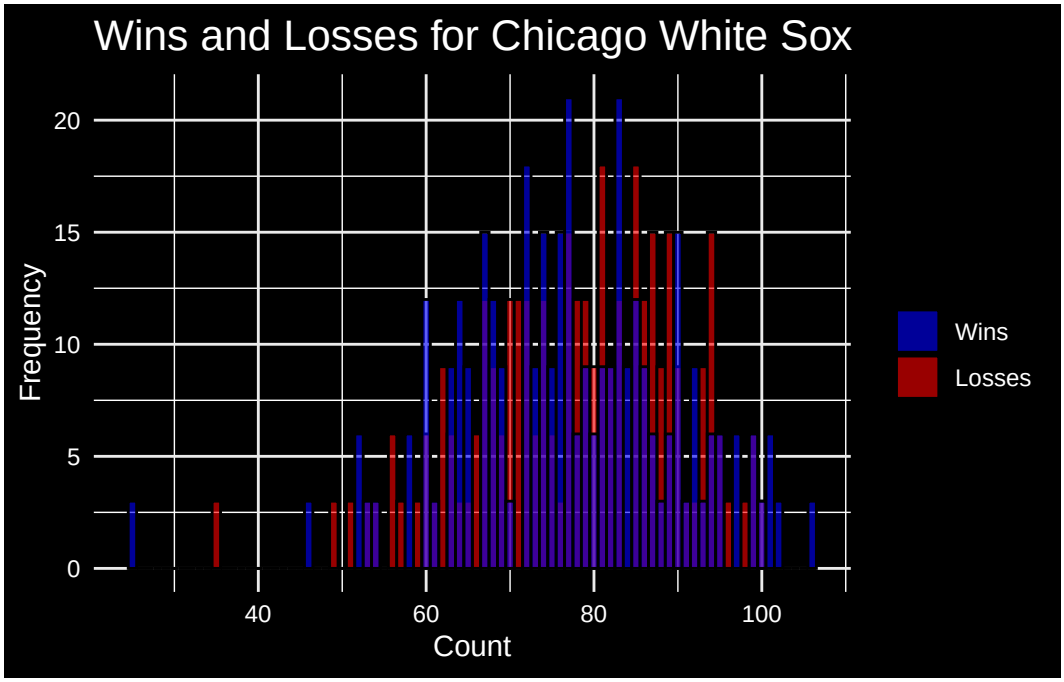


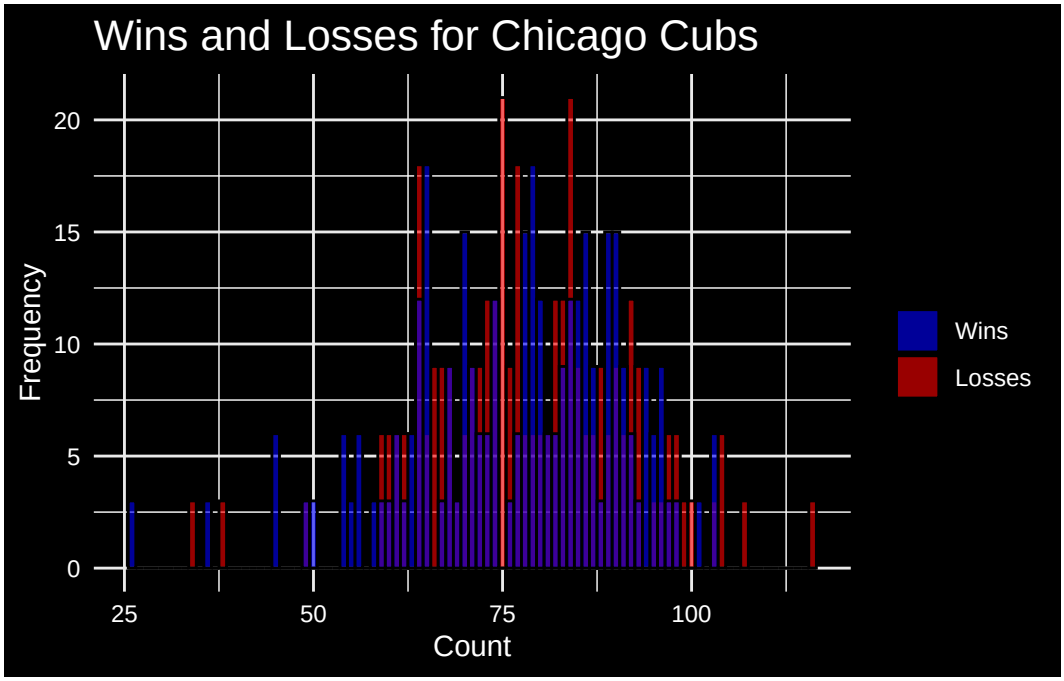

Wins and Losses for St. Louis Browns
Frequency
Count
0
1
2
3
4
25
50
75
100
Wins
Losses

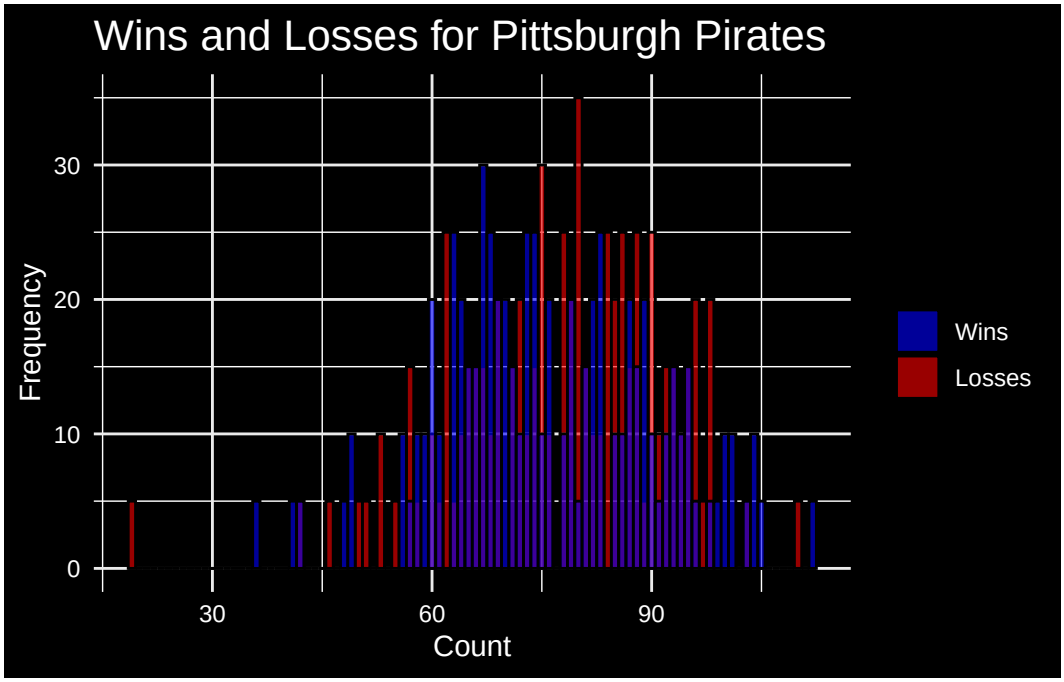

Wins and Losses for Detroit Wolverines
Frequency
Count
0.0
0.5
1.0
1.5
2.0
40
60
80
Wins
Losses

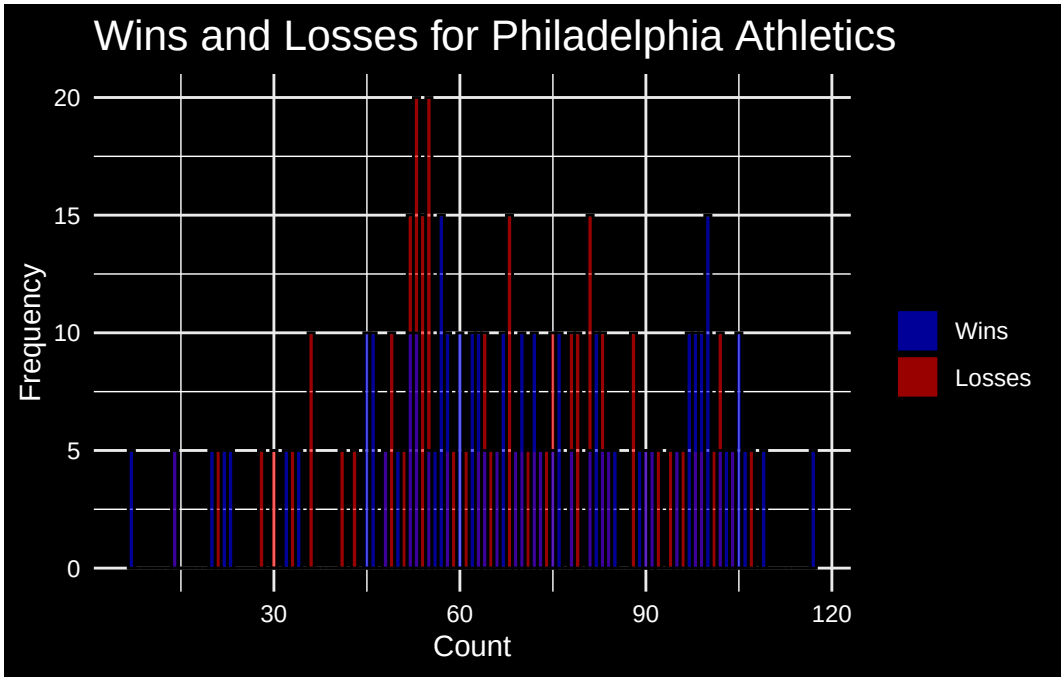

Wins and Losses for New York Giants
Frequency
30
20
10
0
40
60
80
100
Count
Wins
Losses

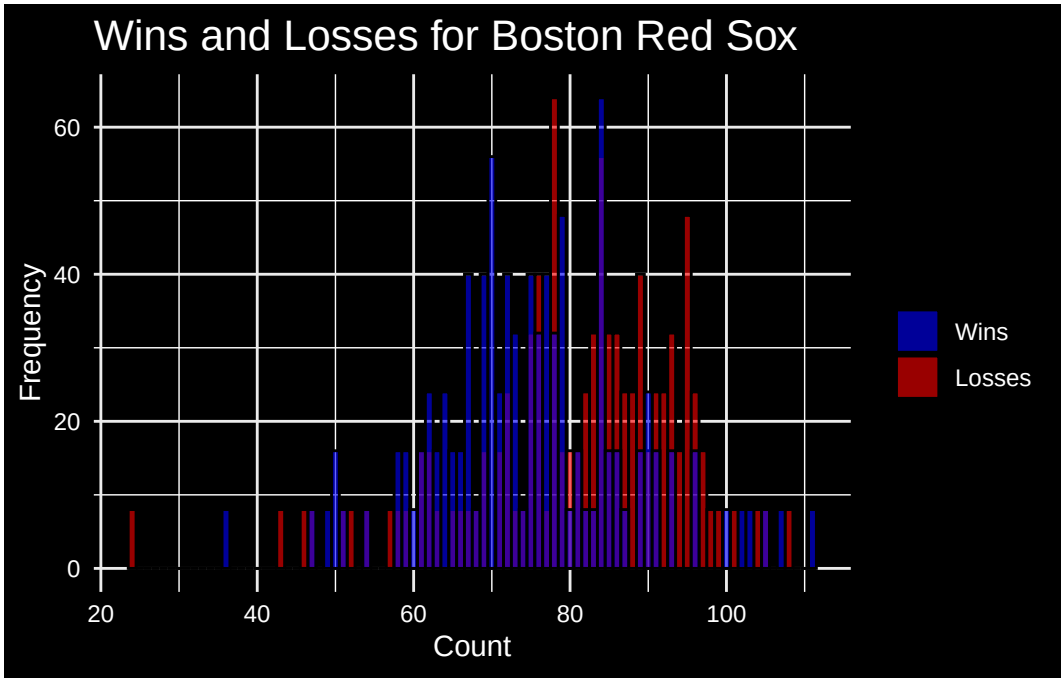

Wins and Losses for Boston Americans
Frequency
1.00
0.75
0.50
0.25
0.00
60
80
100
Count
Wins
Losses

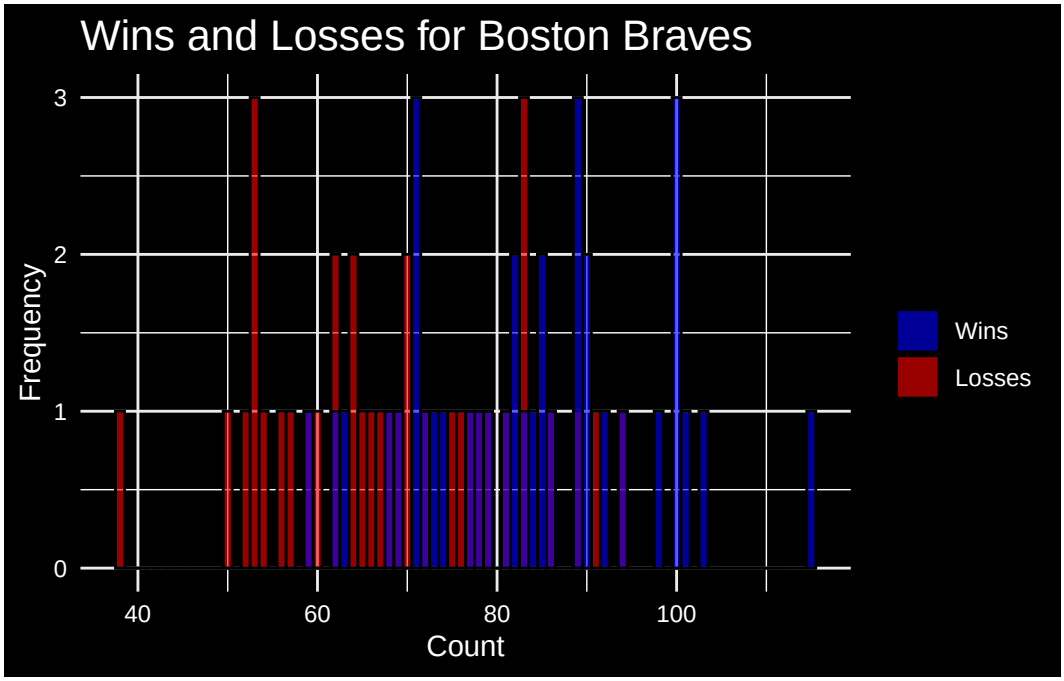

Wins and Losses for Chicago White Sox
Frequency
20
15
10
5
0
40
60
80
100
Count
Wins
Losses

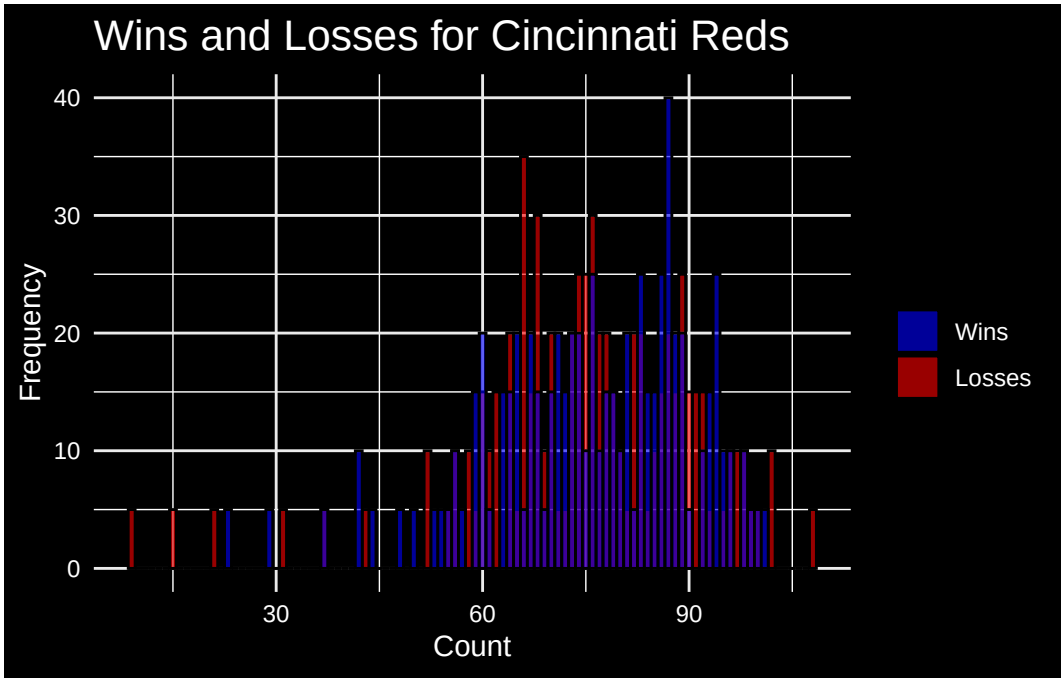

Wins and Losses for Chicago Cubs
Frequency
20
15
10
5
0
25
50
75
100
Count
Wins
Losses

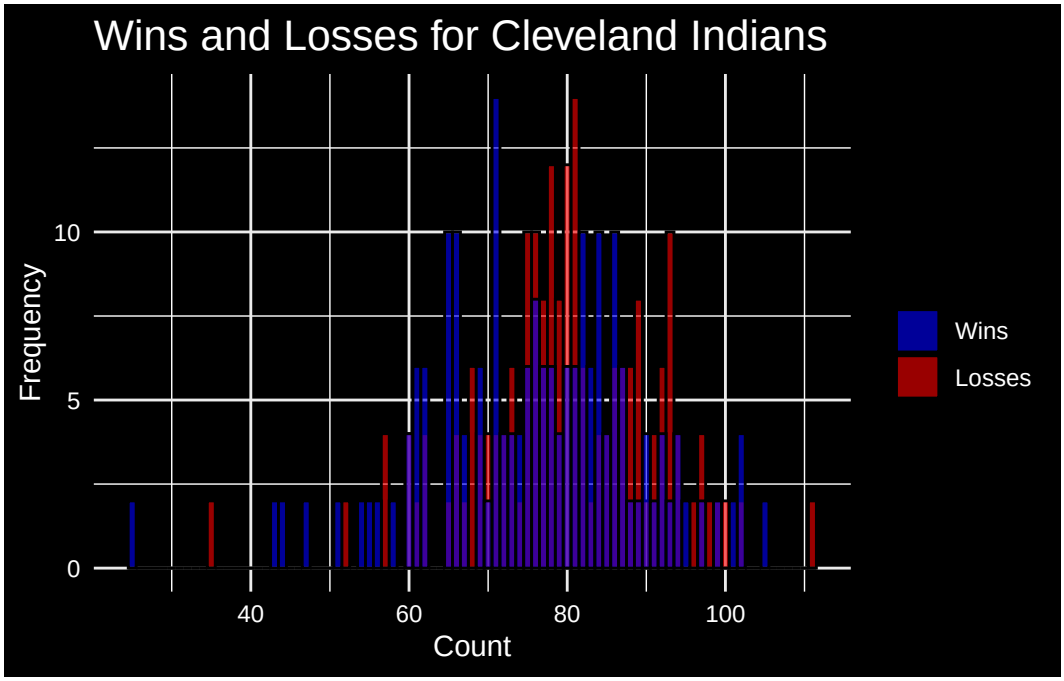

Wins and Losses for Pittsburgh Pirates
Frequency
0
10
20
30
30
60
90
Count
Wins
Losses

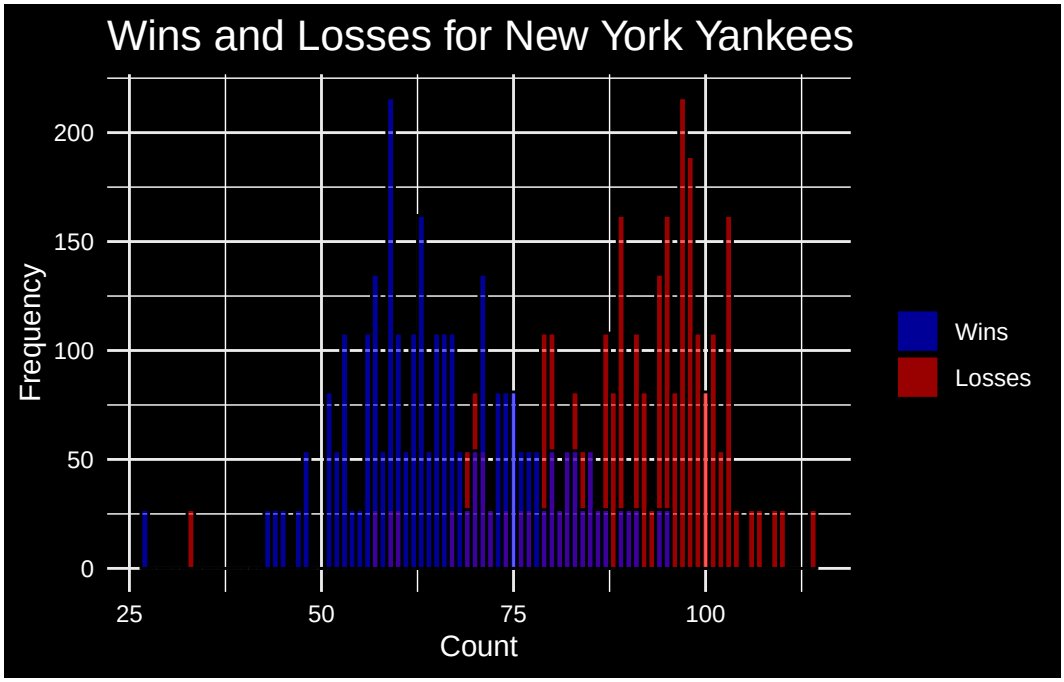

Wins and Losses for Philadelphia Athletics
Frequency
0
5
10
15
20
30
60
90
120
Count
Wins
Losses

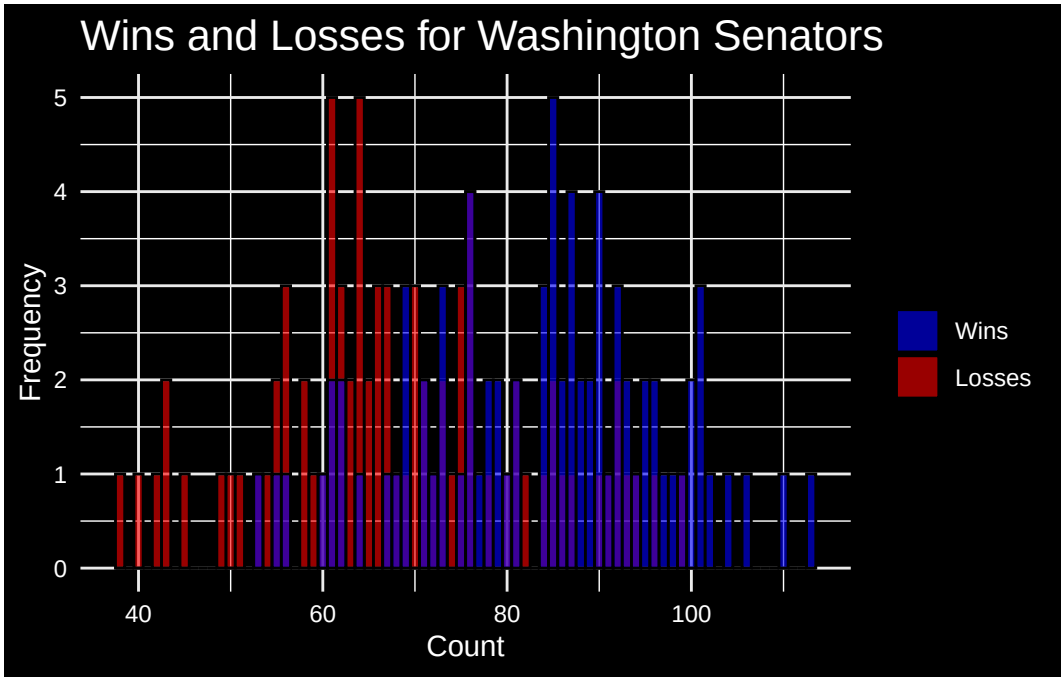

Wins and Losses for Boston Red Sox
Frequency
0
20
40
60
20
40
60
80
100
Count
Wins
Losses

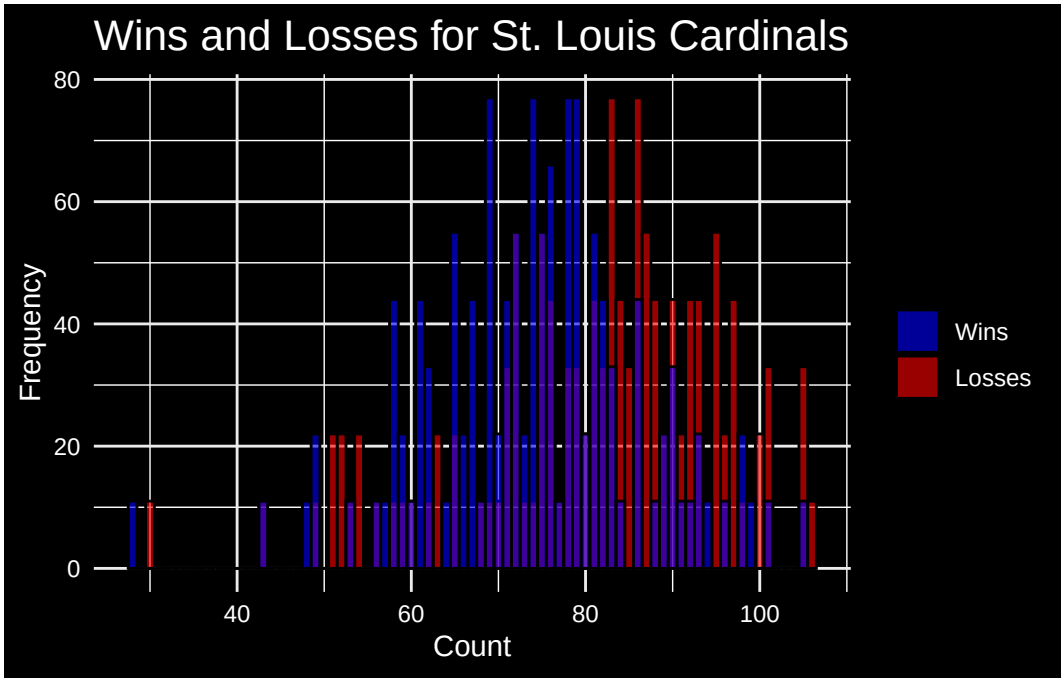

Wins and Losses for Boston Braves
Frequency
0
1
2
3
40
60
80
100
Count
Wins
Losses

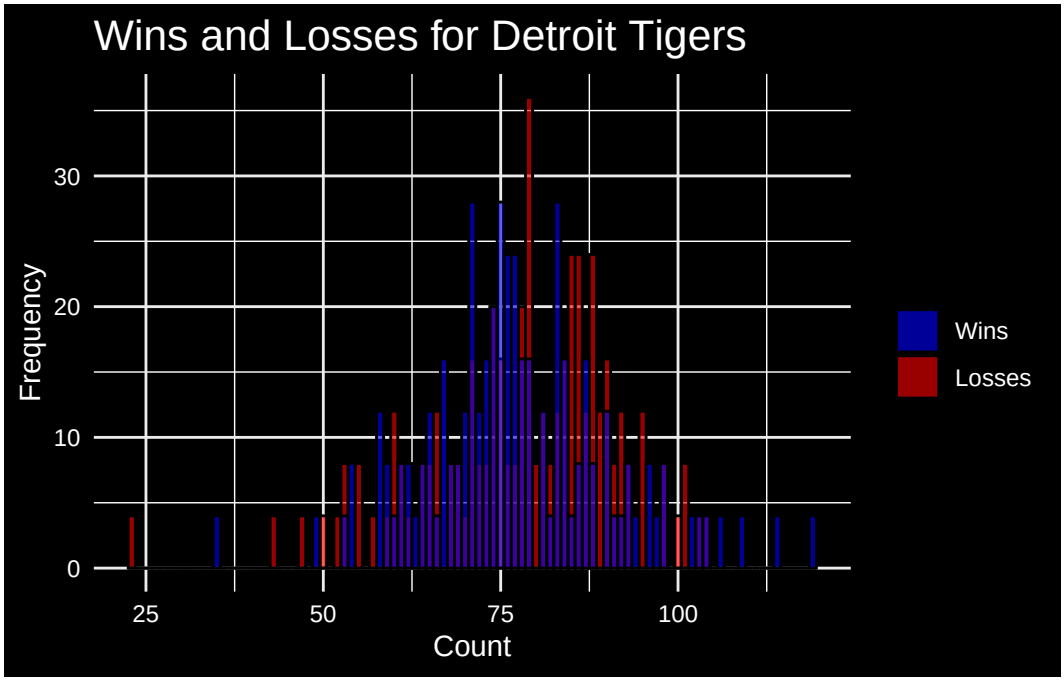

Wins and Losses for Cincinnati Reds
Frequency
40
30
20
10
0
30
60
90
Count
Wins
Losses

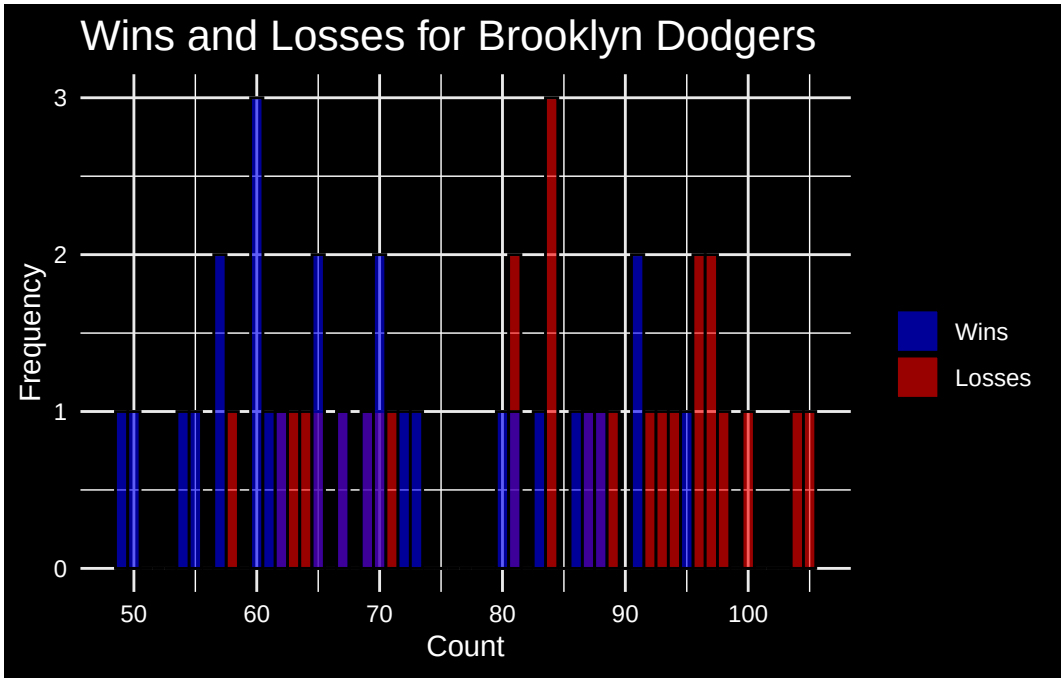

Wins and Losses for Cleveland Indians
Frequency
10
5
0
40
60
80
100
Count
Wins
Losses

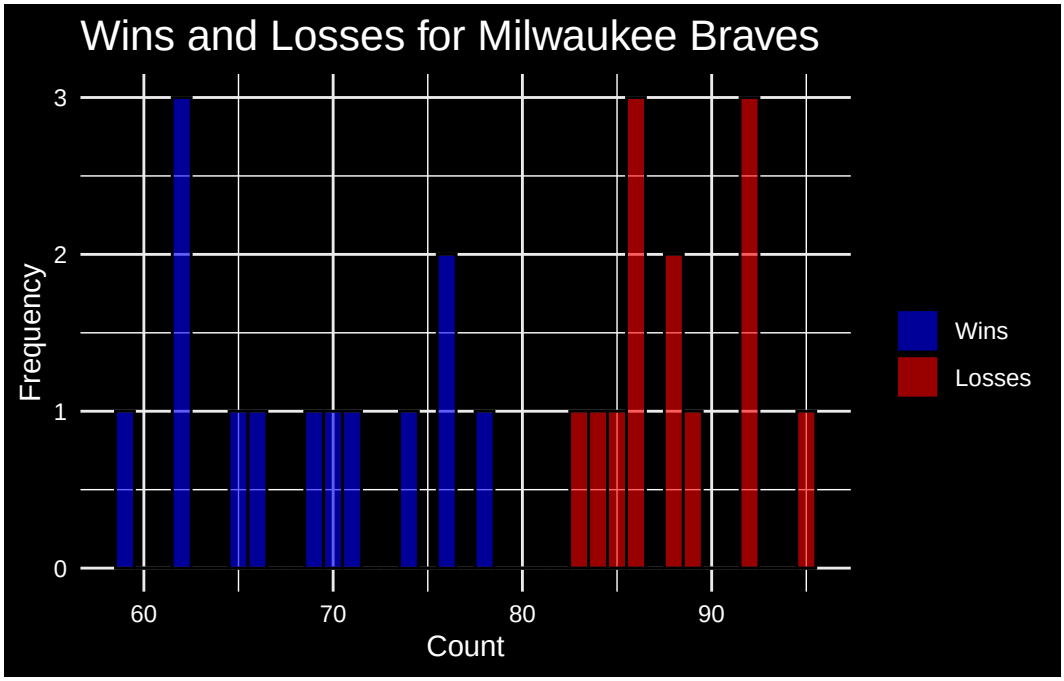

Wins and Losses for New York Yankees
Frequency
200
150
100
50
0
25
50
75
100
Count
Wins
Losses

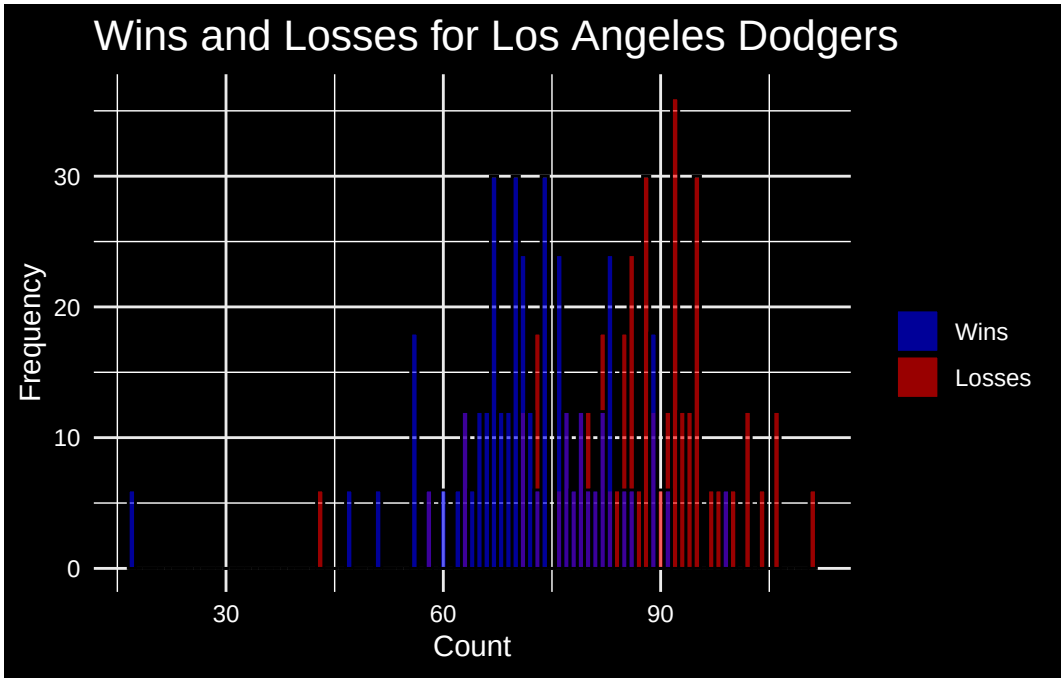

Wins and Losses for Washington Senators
Frequency
5
4
3
2
1
0
40
60
80
100
Count
Wins
Losses

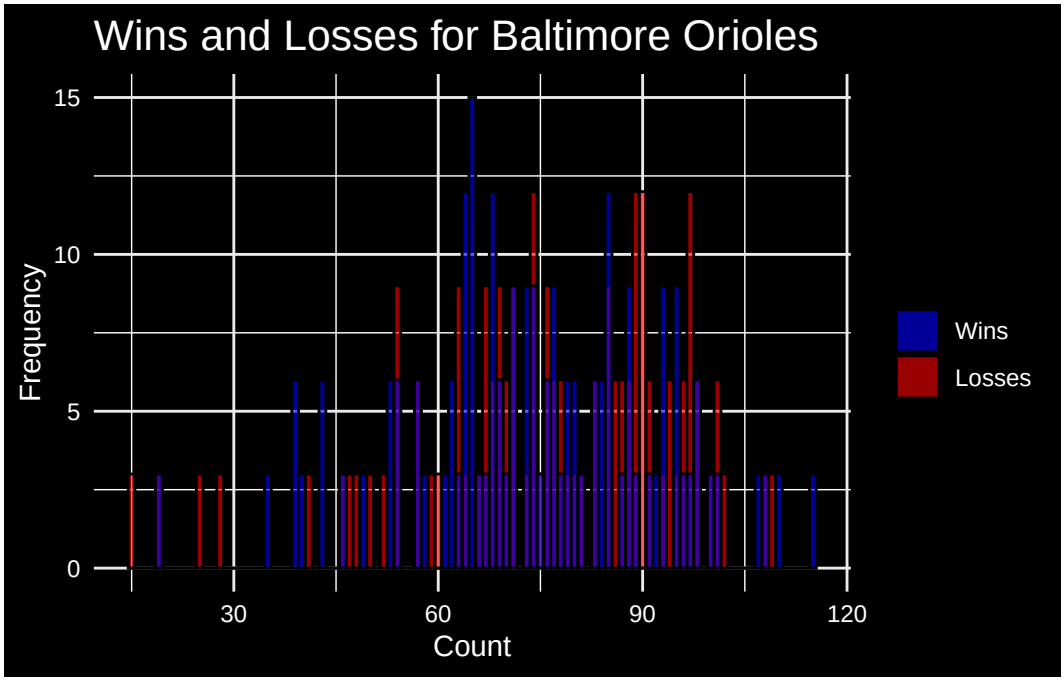

Wins and Losses for St. Louis Cardinals
Frequency
80
60
40
20
0
40
60
80
100
Count
Wins
Losses

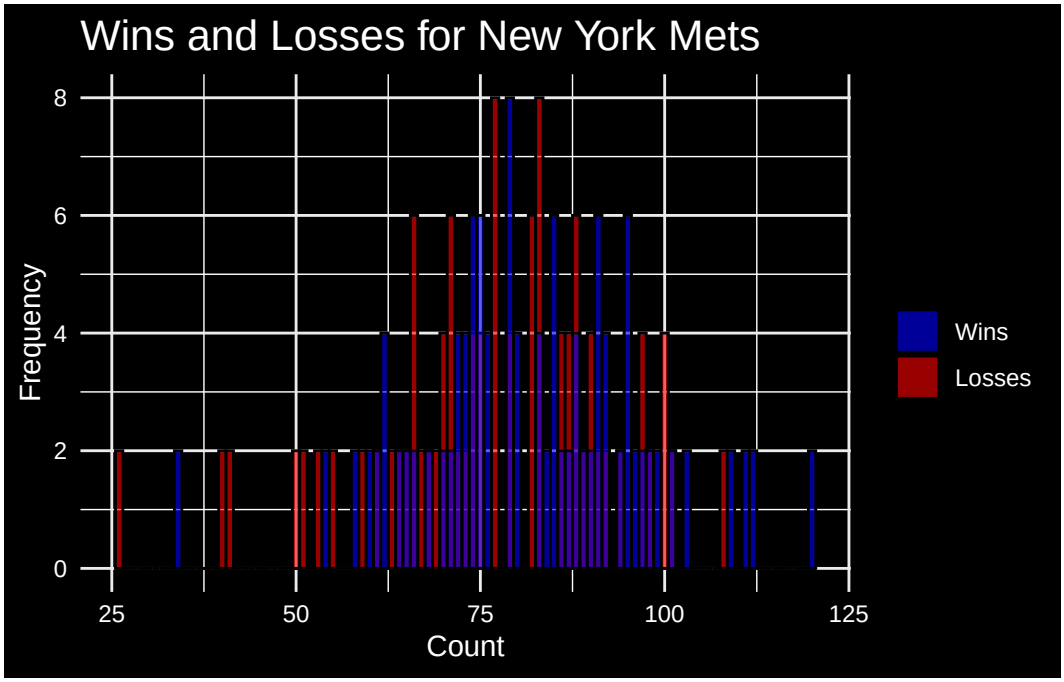

Wins and Losses for Detroit Tigers
Frequency
30
20
10
0
25
50
75
100
Count
Wins
Losses

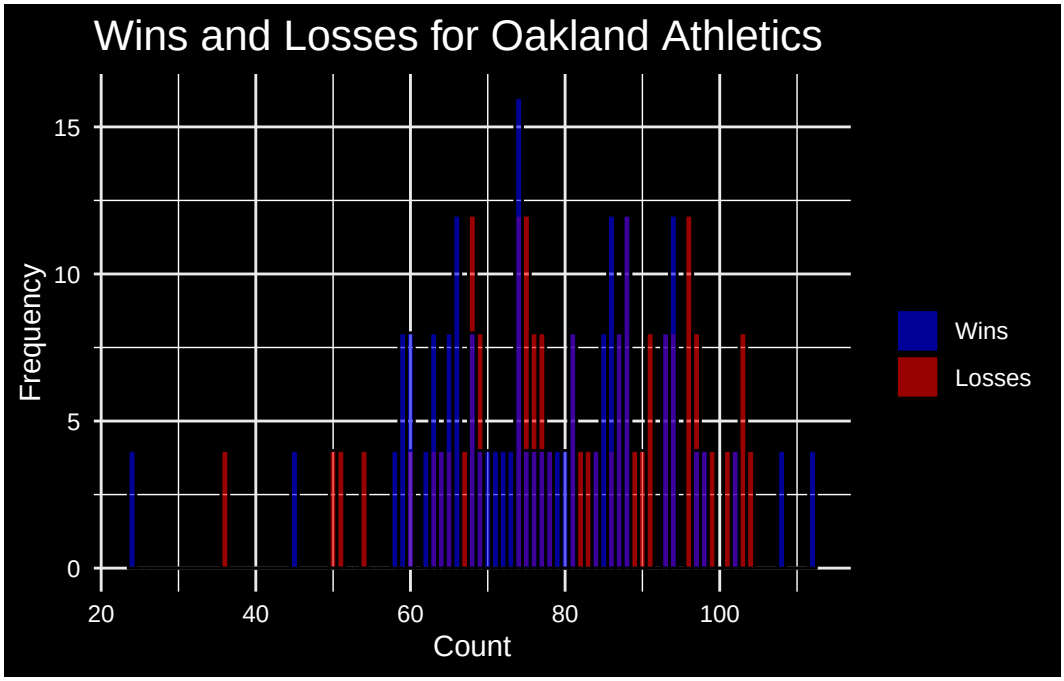

Wins and Losses for Brooklyn Dodgers
Frequency
0
1
2
3
50
60
70
80
90
100
Count
Wins
Losses

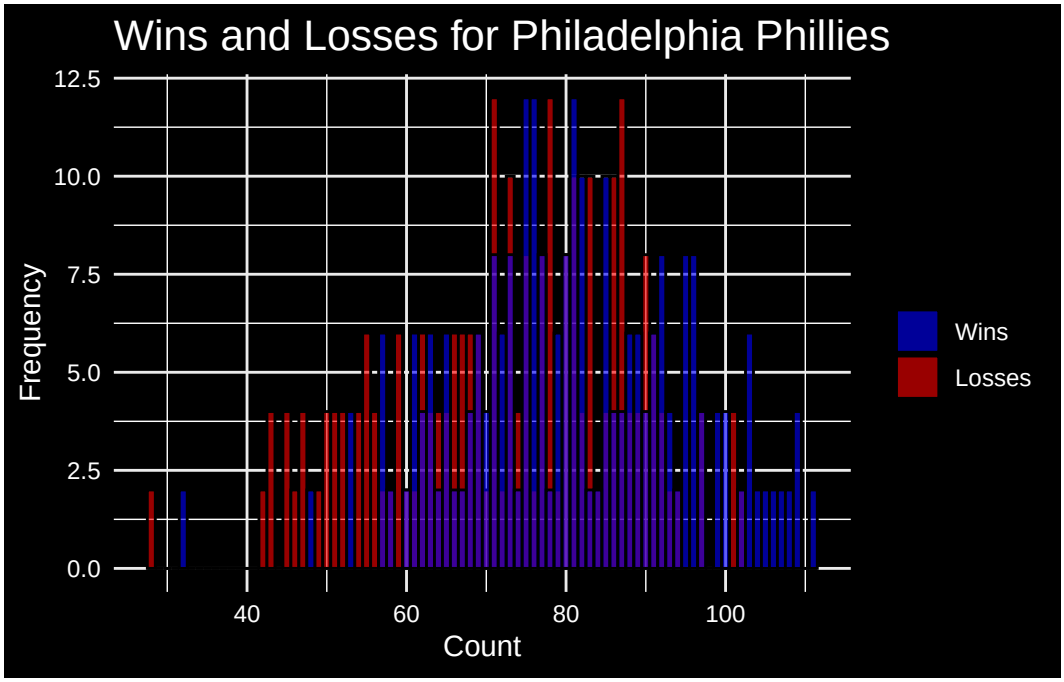

Wins and Losses for Milwaukee Braves
Frequency
0
1
2
3
60
70
80
90
Count
Wins
Losses

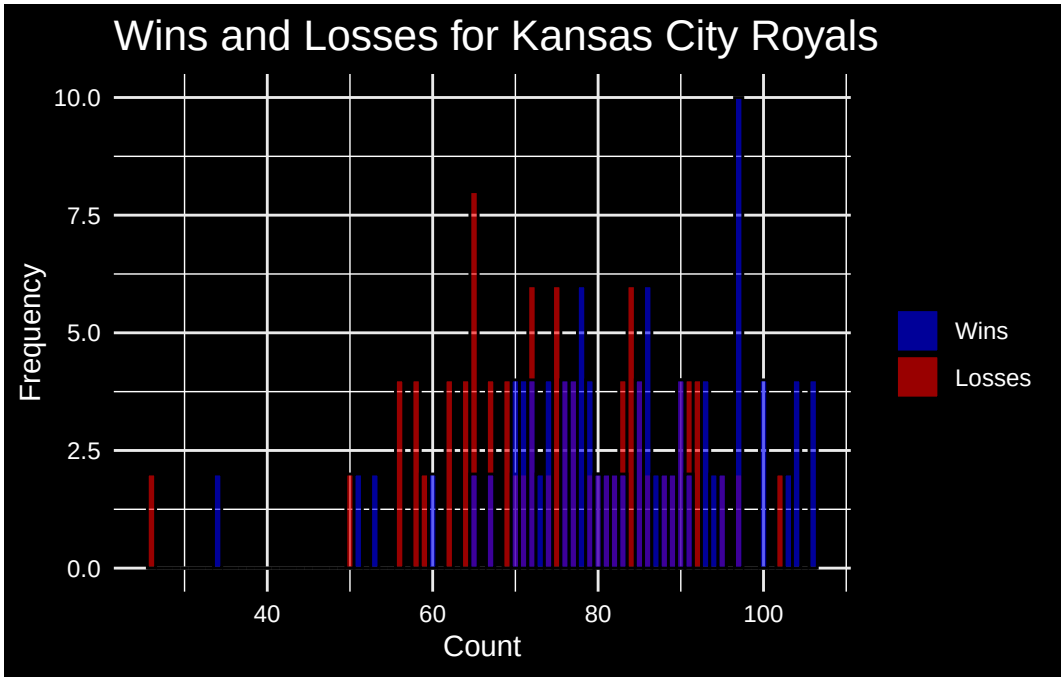

Wins and Losses for Los Angeles Dodgers
Frequency
0
10
20
30
30
60
90
Count
Wins
Losses

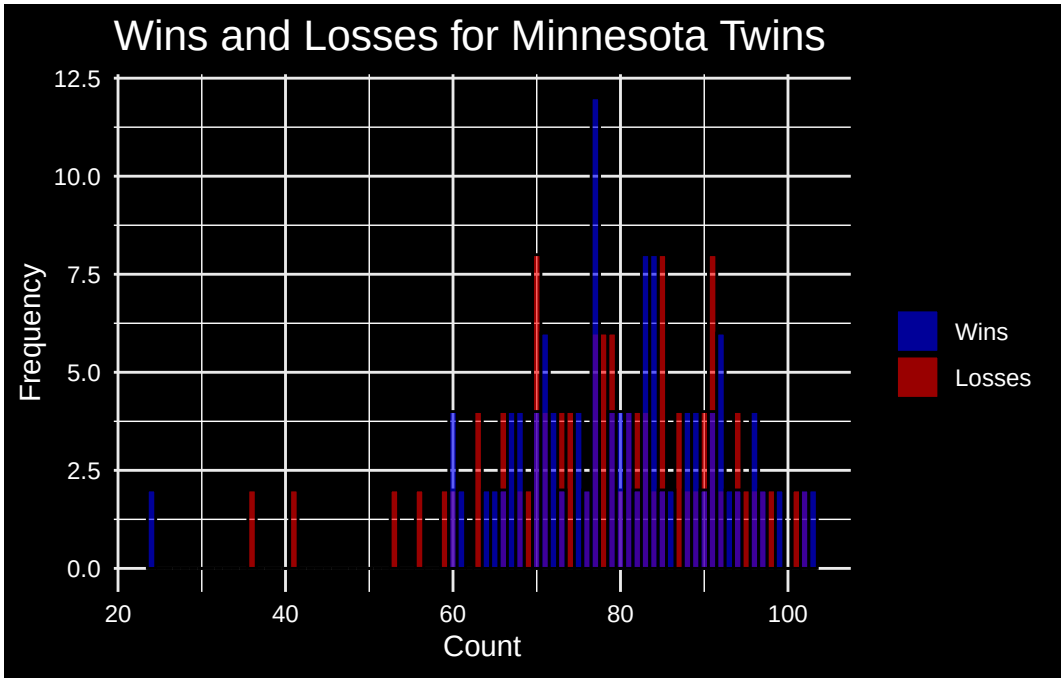

Wins and Losses for Baltimore Orioles
Frequency
0
5
10
15
30
60
90
120
Count
Wins
Losses

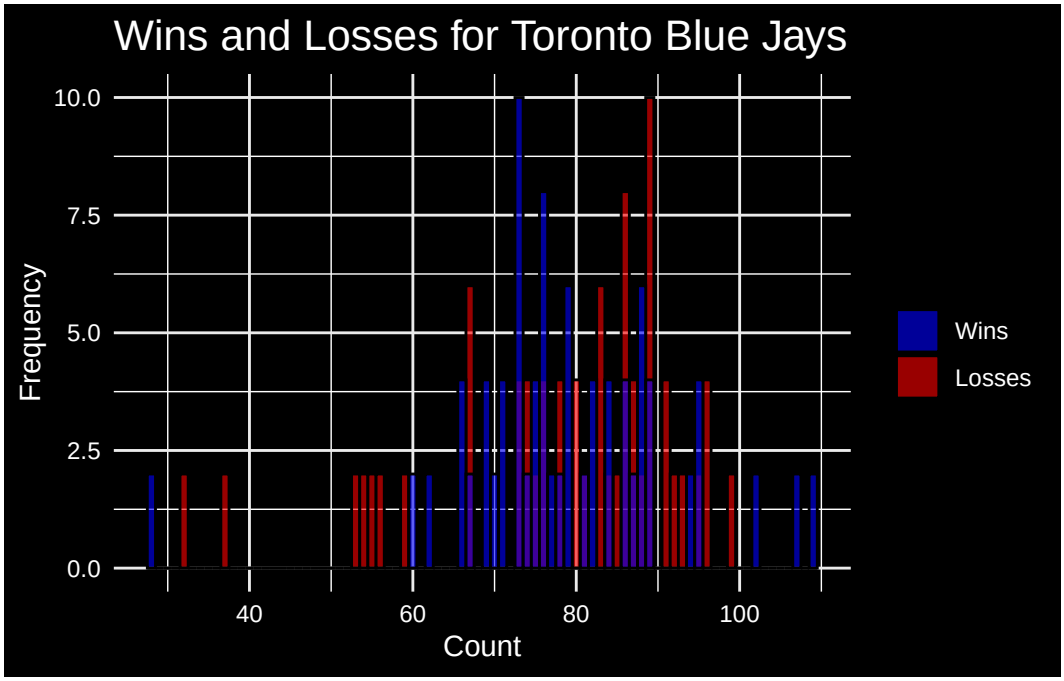

Wins and Losses for New York Mets
Frequency
Count
Wins
Losses
0
2
4
6
8
25
50
75
100
125

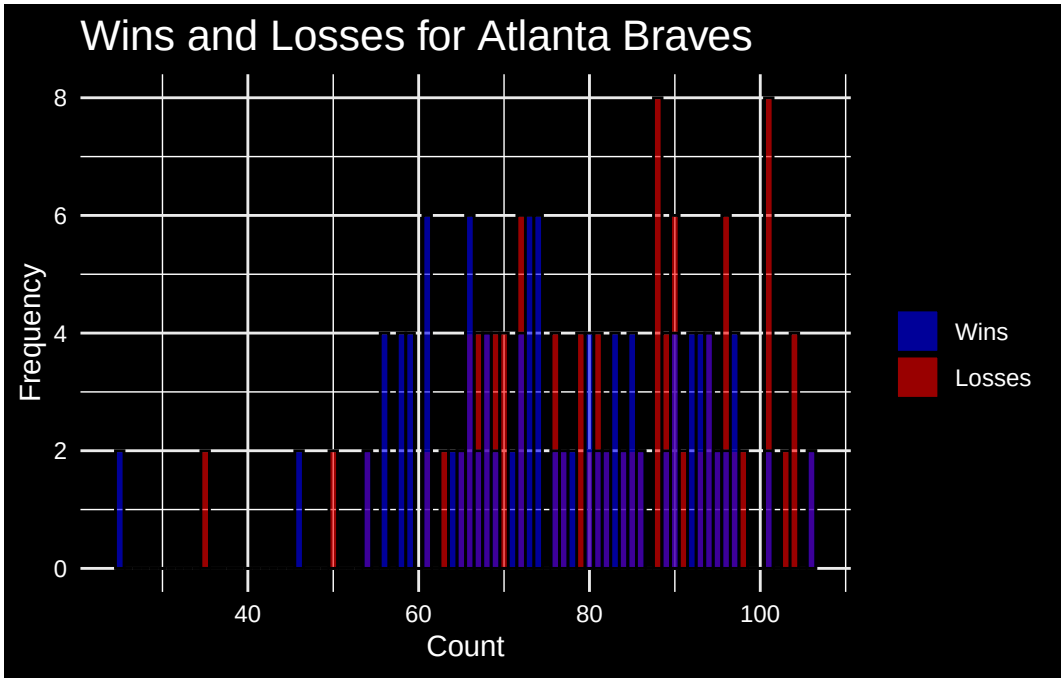

Wins and Losses for Oakland Athletics
Frequency
Count
Wins
Losses
0
5
10
15
20
40
60
80
100

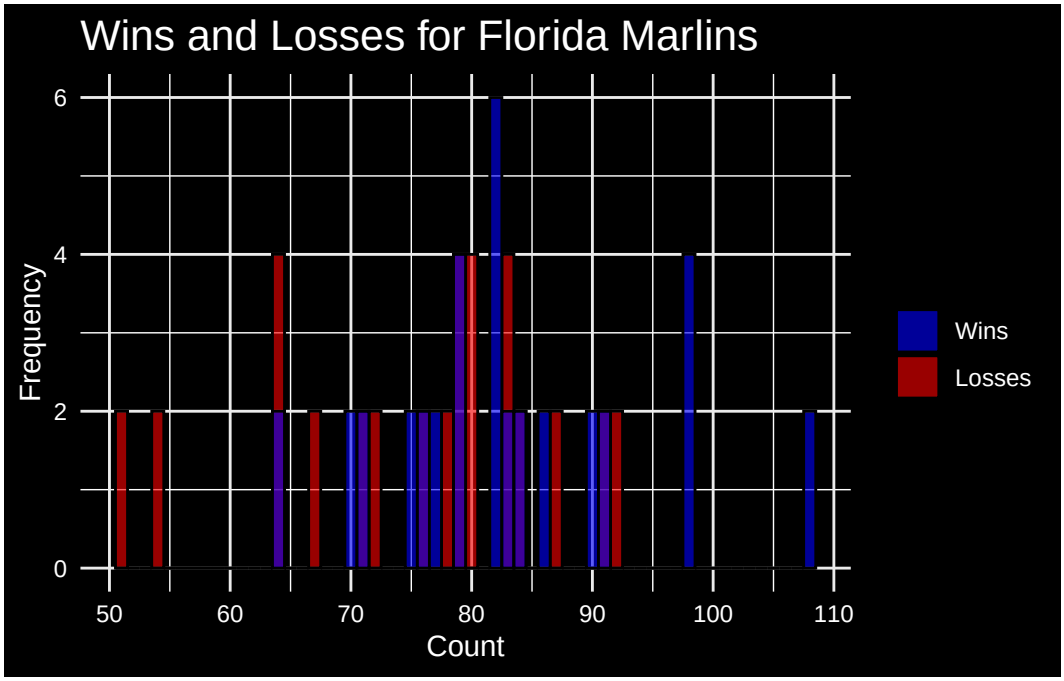

Wins and Losses for Philadelphia Phillies
12.5
10.0
7.5
5.0
2.5
0.0
Frequency
40
60
80
100
Count
Wins
Losses

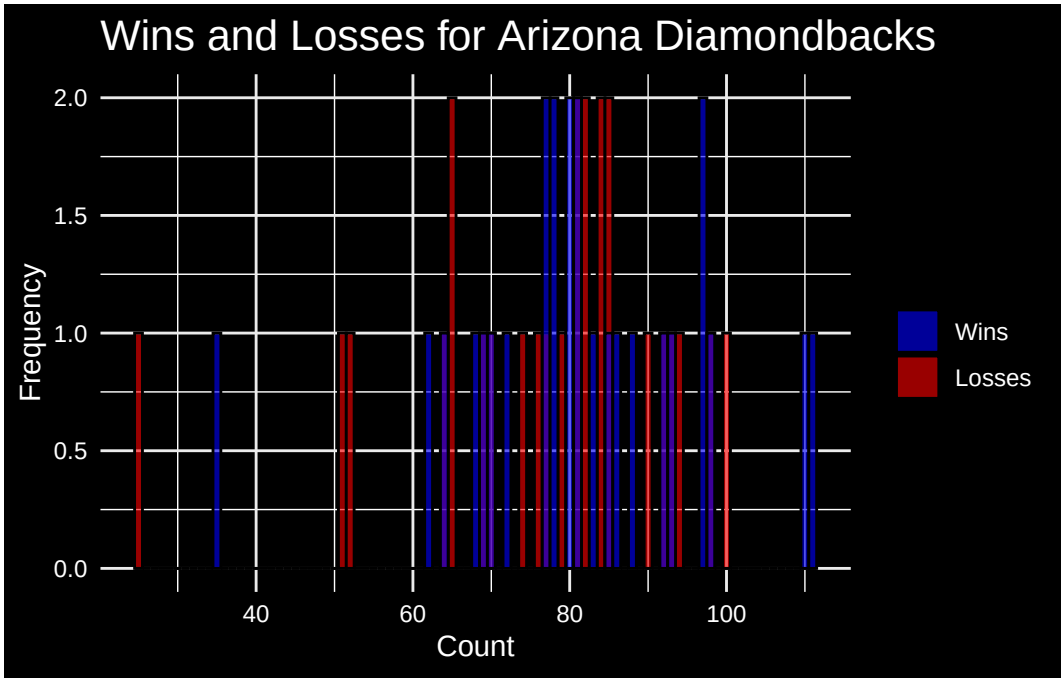

Wins and Losses for Kansas City Royals
10.0
7.5
5.0
2.5
0.0
Frequency
40
60
80
100
Count
Wins
Losses

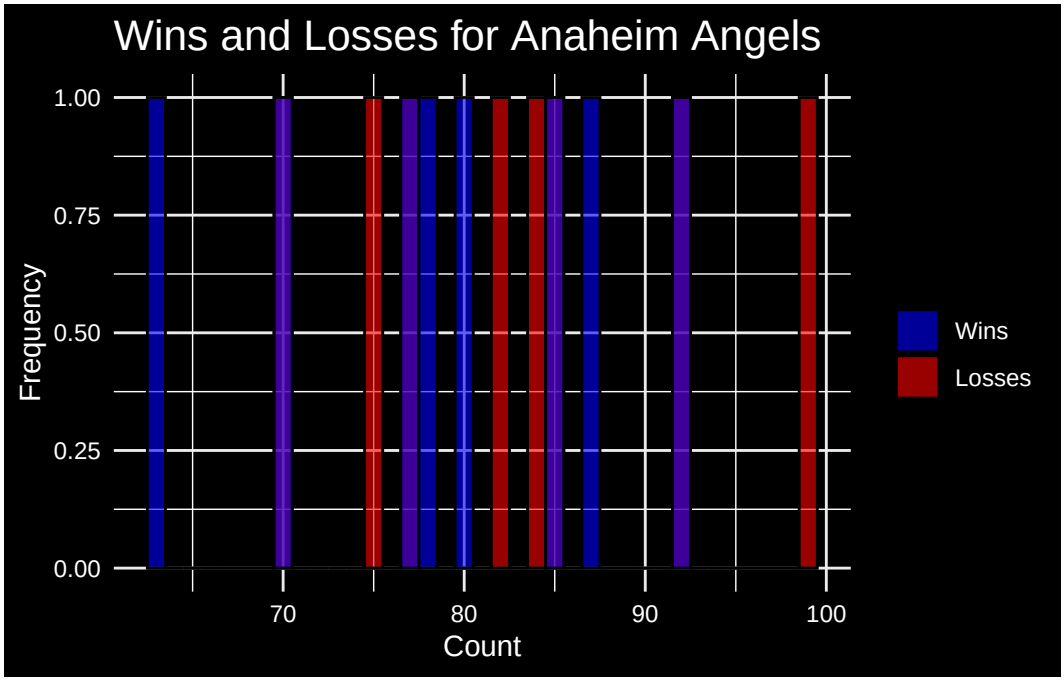

Wins and Losses for Minnesota Twins
12.5
10.0
7.5
5.0
2.5
0.0
Frequency
20
40
60
80
100
Count
Wins
Losses

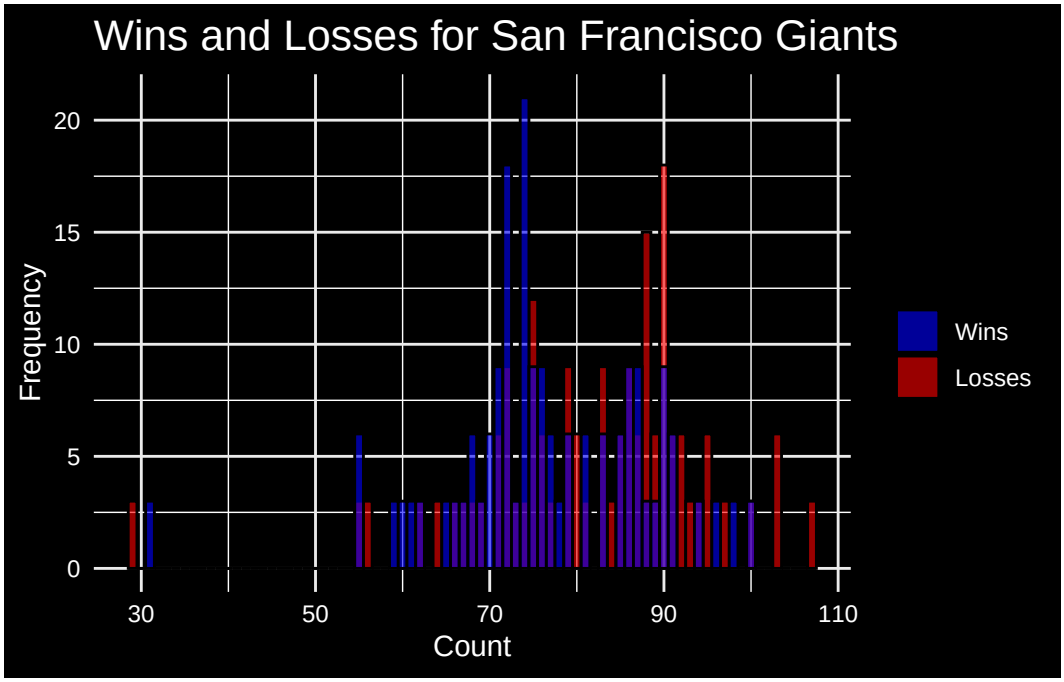

Wins and Losses for Toronto Blue Jays
10.0
7.5
5.0
2.5
0.0
Frequency
40
60
80
100
Count
Wins
Losses

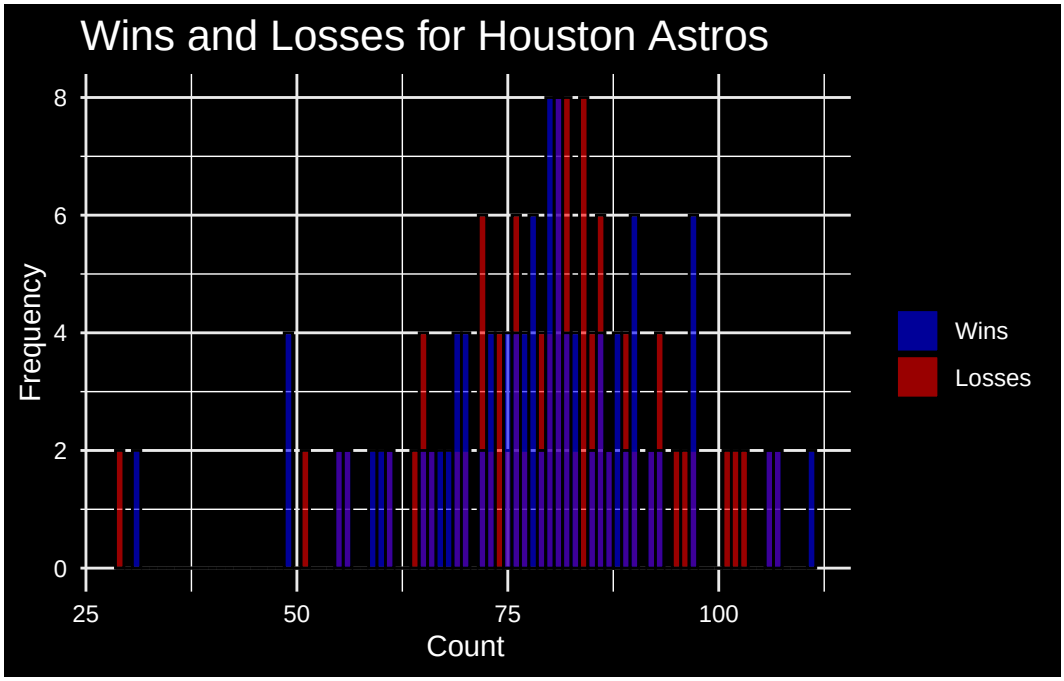

Wins and Losses for Atlanta Braves
Frequency
Count
Wins
Losses
0
2
4
6
8
40
60
80
100

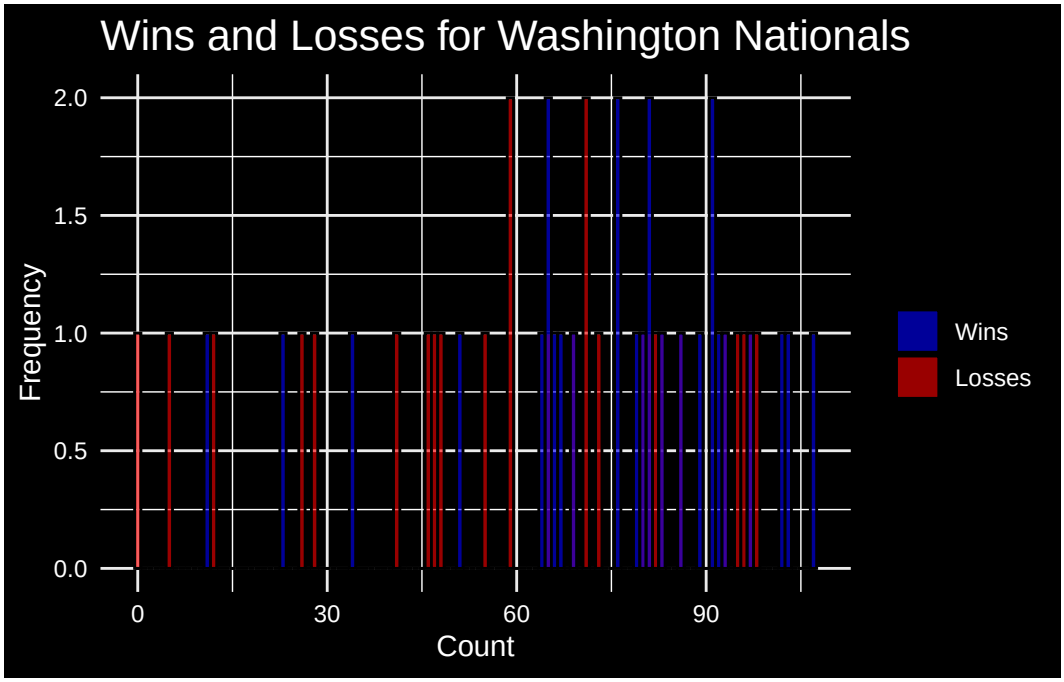

Wins and Losses for Florida Marlins
Frequency
Count
Wins
Losses
0
2
4
6
50
60
70
80
90
100
110

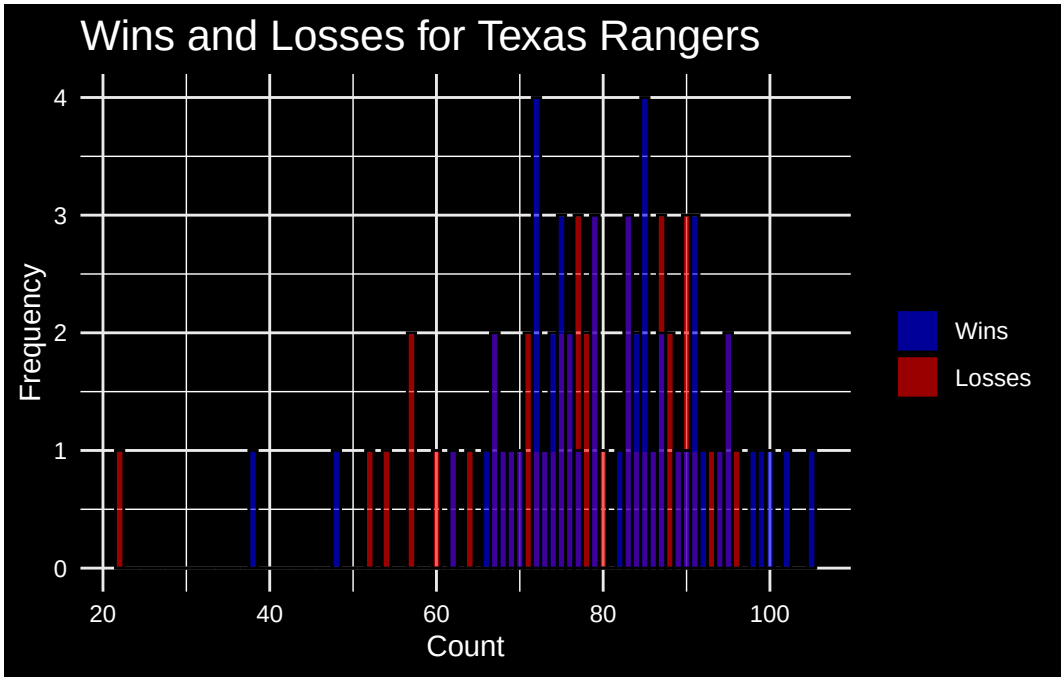

Wins and Losses for Arizona Diamondbacks
Frequency
2.0
1.5
1.0
0.5
0.0
40
60
80
100
Count
Wins
Losses


Wins and Losses for Anaheim Angels
Frequency
1.00
0.75
0.50
0.25
0.00
70
80
90
100
Count
Wins
Losses


Wins and Losses for San Francisco Giants
Frequency
20
15
10
5
0
30
50
70
90
110
Count
Wins
Losses


Wins and Losses for Houston Astros
Frequency
8
6
4
2
0
25
50
75
100
Count
Wins
Losses




The table shows no lows which makes sense because the teams must be able to make higher winning percentages in order to win the world series. Most of these winning percentages are average though, and the frequency of these Highs and Average stay under 4 for the most part. The histograms are highly variable but many follow a clear trend of a peak or peak(s) for

both wins and losses. And wins and losses tend to follow the same pattern per team. If wins for a team is peak shaped the losses are too. If wins follow no pattern, the losses follow no pattern.

## 8. Transformed Data

```
transformed_data = data %>%
  mutate(
    z_H = (H - mean(H, na.rm = TRUE)) / sd(H, na.rm = TRUE),
    z_SO = (SO - mean(SO, na.rm = TRUE)) / sd(SO, na.rm = TRUE),
    z_SOA = (SOA - mean(SOA, na.rm = TRUE)) / sd(SOA, na.rm = TRUE),
    z_SHO = (SHO - mean(SHO, na.rm = TRUE)) / sd(SHO, na.rm = TRUE),
    z_FP = (FP - mean(FP, na.rm = TRUE)) / sd(FP, na.rm = TRUE)
  )

lm_model = lm(WP ~ z_H + z_SO + z_SOA + z_SHO + z_FP, data = transformed_data)

model_summary = summary(lm_model)

summary_table = data.frame(c(model_summary$r.squared, model_summary$coefficients))

model_summary
```

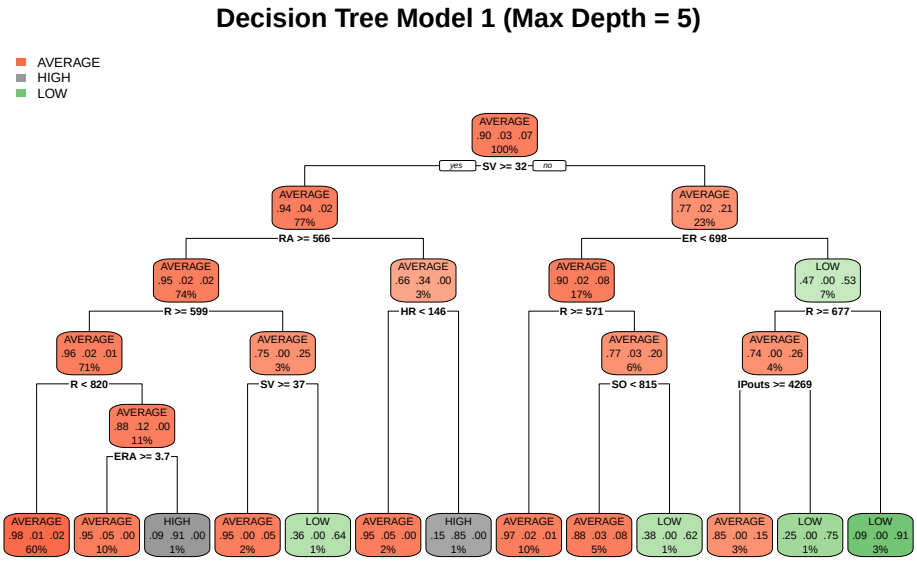
```
Call:
lm(formula = WP ~ z_H + z_SO + z_SOA + z_SHO + z_FP, data = transformed_data)

Residuals:
     Min       1Q   Median       3Q      Max
-0.34912 -0.05030 -0.00125  0.04825  0.39146

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.495559   0.001419 349.136   <2e-16 ***
z_H          0.019678   0.001943  10.128   <2e-16 ***
z_SO        -0.076725   0.004212 -18.215   <2e-16 ***
z_SOA        0.073244   0.004197  17.453   <2e-16 ***
z_SHO        0.030149   0.001545  19.509   <2e-16 ***
z_FP        -0.001228   0.002412  -0.509    0.611
---
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.07812 on 3023 degrees of freedom
  (16 observations deleted due to missingness)
Multiple R-squared:  0.3061,    Adjusted R-squared:  0.3049
F-statistic: 266.7 on 5 and 3023 DF,  p-value: < 2.2e-16
```

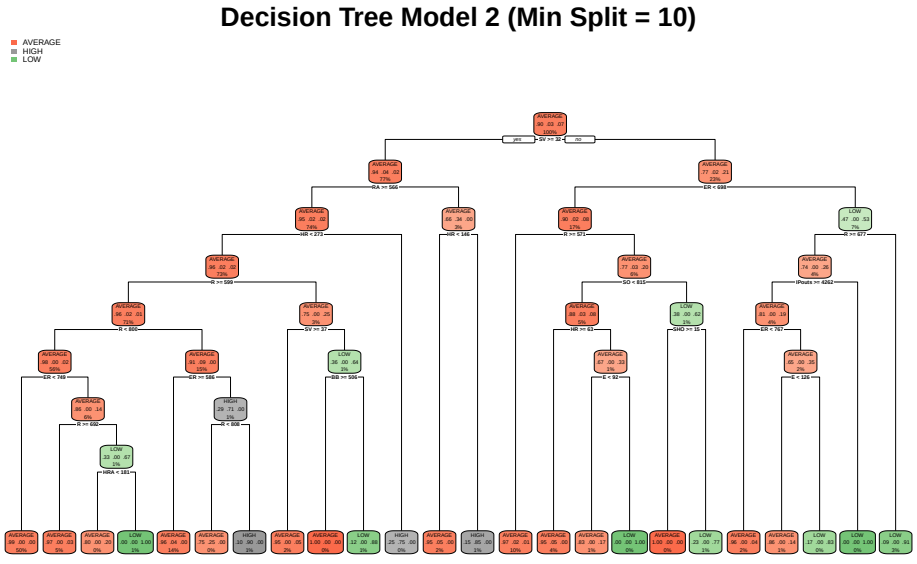
z_FP and z_SO are negatively correlated with the win percentage attribute, while the others are positively correlated. z_FP is not significant in the model. And the model only accounts for 30.6% of the variation.

## 9. Decision Trees

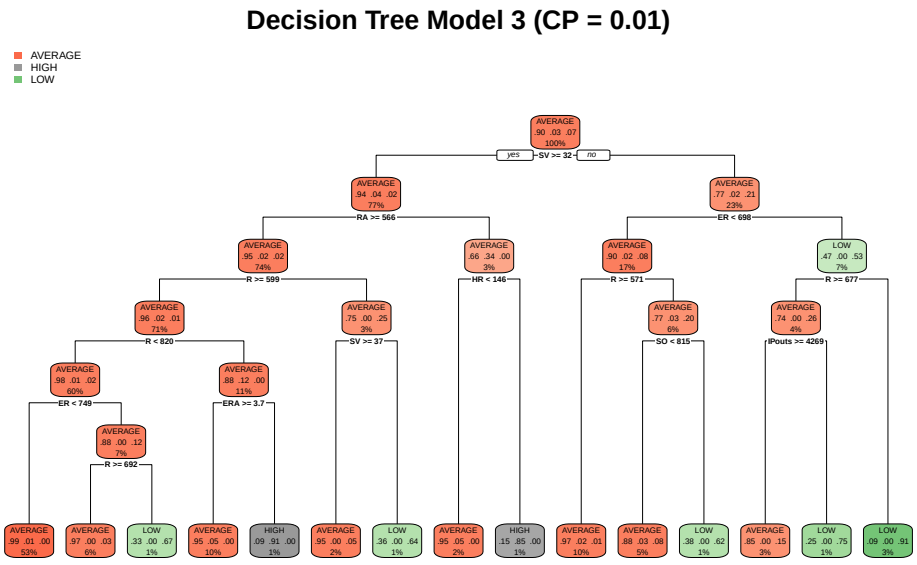
```r
subset_data = data[, 9:ncol(data)]
subset_data = na.omit(subset_data)
subset_data$TARGET = as.factor(subset_data$TARGET)

set.seed(123)

train_index = sample(1:nrow(subset_data), 0.8 * nrow(subset_data))
train_data = subset_data[train_index, ]
test_data = subset_data[-train_index, ]

tree_model1 = rpart(TARGET ~ ., data = train_data, control = rpart.control(maxdepth = 5))
tree_model2 = rpart(TARGET ~ ., data = train_data, control = rpart.control(minsplit = 10))
tree_model3 = rpart(TARGET ~ ., data = train_data, control = rpart.control(cp = 0.01))

rpart.plot(tree_model1, main = "Decision Tree Model 1 (Max Depth = 5)")
```




```
rpart.plot(tree_model2, main = "Decision Tree Model 2 (Min Split = 10)")
```



```
rpart.plot(tree_model3, main = "Decision Tree Model 3 (CP = 0.01)")
```




```
calculate_accuracy = function(model, data) {
  predictions = predict(model, data, type = "class")
  accuracy = sum(predictions == data$TARGET) / nrow(data)
  return(accuracy)
}

train_acc_model1 = calculate_accuracy(tree_model1, train_data)
test_acc_model1 = calculate_accuracy(tree_model1, test_data)

train_acc_model2 = calculate_accuracy(tree_model2, train_data)
test_acc_model2 = calculate_accuracy(tree_model2, test_data)

train_acc_model3 = calculate_accuracy(tree_model3, train_data)
test_acc_model3 = calculate_accuracy(tree_model3, test_data)

cat("Model 1 Training Accuracy: ", train_acc_model1, "\n")
```

```
Model 1 Training Accuracy:  0.9501247
```

```r
cat("Model 1 Testing Accuracy: ", test_acc_model1, "\n")
```

```
Model 1 Testing Accuracy:  0.9202658
```

```r
cat("Model 2 Training Accuracy: ", train_acc_model2, "\n")
```

```
Model 2 Training Accuracy:  0.967581
```

```r
cat("Model 2 Testing Accuracy: ", test_acc_model2, "\n")
```

```
Model 2 Testing Accuracy:  0.9335548
```

```r
cat("Model 3 Training Accuracy: ", train_acc_model3, "\n")
```

```
Model 3 Training Accuracy:  0.9534497
```

```r
cat("Model 3 Testing Accuracy: ", test_acc_model3, "\n")
```

```
Model 3 Testing Accuracy:  0.923588
```

By capping the depth for the first model, we ensure that the tree remains relatively simple and doesn't overfit the training data by learning highly specific, potentially noisy patterns. In essence, it prevents the tree from becoming overly complex while still being able to capture significant patterns in the data.

The second model was designed to prevent the tree from splitting too often by enforcing a minimum node size of 10 (i.e., a split will only happen if at least 10 observations are present in a node). By doing this, we reduce the likelihood of small, insignificant splits that might overfit to minor fluctuations in the data.

The third model was designed so that a smaller CP (like 0.01) allows the model to prune less impactful splits, reducing the number of nodes in the tree. This approach helps ensure that only the most meaningful splits remain.

The three decision trees reflect how different hyperparameters influence the model's structure and performance in classifying the target variable ('average,' 'high,' 'low'). The results show that **Model 2** has the highest training and testing accuracy, suggesting it captures more patterns in the data.

## Conclusion

The analysis of attributes 7-30 for predicting a successful team, based on both Rank and Target, highlights key findings. Attributes like Winning Percentage (WP), Runs scored (R), and Earned run average (ERA) emerged as strong predictors of success. Teams with higher Winning Percentages (WP) generally ranked higher and fell into the "HIGH" category for Target, indicating strong overall performance.

Other offensive metrics, such as Hits (H), Home Runs (HR), and Doubles (2B), also contributed to team success by boosting scoring opportunities. On the defensive side, Earned runs allowed (ER), Shutouts (SHO), and Strikeouts by pitchers (SOA) were important in limiting opponents' scoring chances, directly impacting a team's Rank and Target outcomes.

The findings suggest that a combination of both offensive and defensive strategies is essential for predicting a good team. Specifically, teams that excel in scoring while minimizing opponent runs perform better. This dataset reveals that a balanced focus on both aspects is key to predicting team success. By analyzing these attributes, one can learn the importance of comprehensive team dynamics rather than just individual performance, helping refine strategies for evaluating team success.