Finalproject

Ravi G and Rohit K

3/8/2022

Background:

Every year, the National Basketball Association (NBA) ends the season with a series of championship games between the two best teams in the league. The series is a best-of-7, where the first team to win 4 games is the winner of the series. The championship is a very important accolade, both teams and players are compared by the number of championships they have won. In these comparisons, statistics like field goal percentage, offensive rebounds, steals, and blocks can be used to determine a team's performance and be an indicator of how much better one team is than another.

Statistics in the NBA are highly analyzed, very often before a championship game to try to predict the outcome of the game. Our group wants to create a model that will be able to predict whether or not a team wins the NBA championship given the team's statistics during the NBA finals game. We want to find weights for each statistic that can show us how important each statistic is to predicting the overall winner, which helps our group conduct a greater analysis of how different focuses and strategies can affect the outcome of NBA games. Our group believes that certain statistics such as field goal percentage, turnovers, and offensive rebounds will be prevalent in winning NBA teams.

Data:

The dataset with which we want to create our model is the NBA Finals Team Stats dataset on Kaggle uploaded by Dave Rosenman. The dataset contains final data from 1980 to 2018, and is divided into two tables. The first table contains the data of each winning team and the second contains the losing team. Each observation includes data points like field goals made, field goals attempted, three point shots made, free throws made, total rebounds, assists, steals, turnovers, blocks, and many other statistics that will be covered in the data summary. The data takes averages from each game in the series, and its an average of the performance of the team in this category across all the games played in the series.

The NBA Finals Team Stats Dataset has been analyzed and used to create models by several Kaggle Users. One project to note is a report written by Ziyu Liu (insert citation here) called "Three pointers win championships", in which the author creates a model to see if the number of three point shots made by a team can predict whether or not the team wins the championship. In the study, the model achieves an accuracy of 59%. This tells us that, while three point shots are important, more statistics are required to be able to create a more accurate model.

In order to create the dataset we are using in this study, we started with two separate datasets, one for all of the series winners (NBA Champions) and one of the runner-ups. We created a new column `win` with a 1 if the team won the series and a 0 if the team lost. This will be our predictor variable for the model. Next, we combined the two datsets and randomized the order of the entries. Our goal is to first analyze each of the variables to determine which will be the most useful in creating our model, then going through several iterations of models before choosing the most accurate one.

These are the libraries that will be used to create this model:

```
library(tidyverse)
library(broom)
library(pROC)
library(plotROC)
```

Exploratory Data Analysis

We started by creating the dataset we wish to use for this study (using the process mentioned in the Data section).

```
champs_data <- read_csv("data/champs_series_averages.csv")
```

```
## New names:
## * `` -> ...1
```

```
## Rows: 38 Columns: 22
```

```
## -- Column specification ----------------------------------------------------
## Delimiter: ","
## chr  (2): Status, Team
## dbl (20): ...1, Year, PTS, FG, FGA, FGP, TP, TPA, TPP, FT, FTA, FTP, ORB, DR...
```

```
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
runnerups_data <- read_csv("data/runner_ups_series_averages.csv")
```

```
## New names:
## * `` -> ...1
```

```
## Rows: 38 Columns: 22
```

```
## -- Column specification ----------------------------------------------------
## Delimiter: ","
## chr  (2): Status, Team
## dbl (20): ...1, Year, PTS, FG, FGA, FGP, TP, TPA, TPP, FT, FTA, FTP, ORB, DR...
```

```
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
champs_data <- champs_data %>%
  mutate(win = "1")
runnerups_data <- runnerups_data %>%
  mutate(win = "0")
```

```
all_data <- rbind(champs_data, runnerups_data)
```

Next, we eliminated some variables that we did not wish to explore or view the effect they would have on the model. This includes statistics like `FTA` (Free Throw Attempts), `TPA` (Three Point Attempts), `BLK` (Blocks). Some of these statistics describe the attempts to make a point, however the statistics describing how many points were made in that fashion would be a much more accurate tool in the model. Others simply do not happen often enough to quantifiable change the course of a championship series.

```
useful_data = subset(all_data, select = -c(...1,Year, Status, Team, FT, FTA, FTP, TRB) )
```

Now, we can start our EDA. First, we check the plots of each variable for both the losers and the winners to make sure each distribution is roughly normal.

```
ggplot(data = useful_data %>% filter(win == 1), aes(x = PTS)) +   geom_histogram(binwidth = 4) +
  labs(x = "Points",
       y = "Count",
       title = "Point Distribution")
```
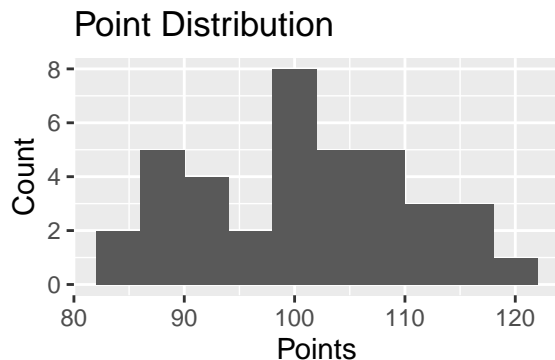


```
ggplot(data = useful_data %>% filter(win == 0), aes(x = PTS)) +   geom_histogram(binwidth = 4) +
  labs(x = "Points",
       y = "Count",
       title = "Point Distribution")
```
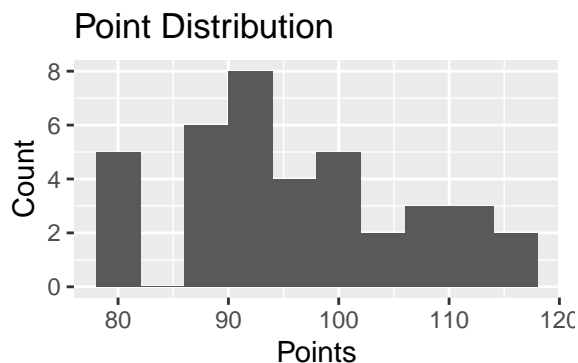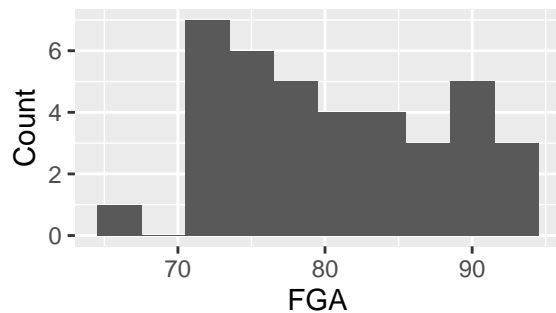


```
ggplot(data = useful_data %>% filter(win == 1), aes(x = FG)) +   geom_histogram(binwidth = 2) +
  labs(x = "FG",
       y = "Count",
       title = "FG Made Distribution")
```

FG Made Distribution
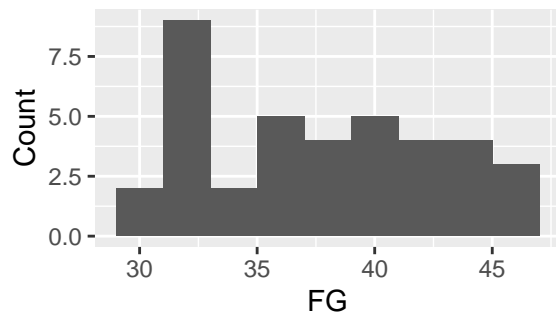

```
ggplot(data = useful_data %>% filter(win == 0), aes(x = FG)) +   geom_histogram(binwidth = 2) +
  labs(x = "FG",
       y = "Count",
       title = "FG Made Distribution")
```
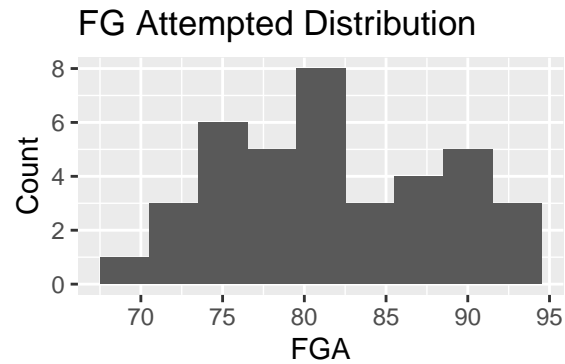
FG Made Distribution
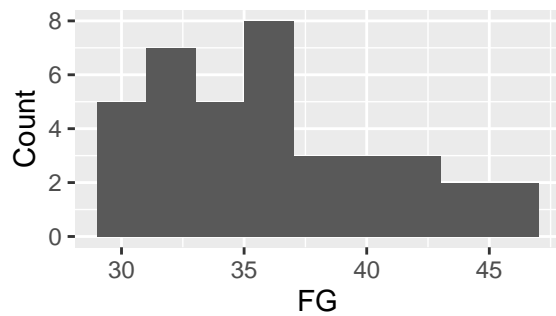

```
ggplot(data = useful_data %>% filter(win == 1), aes(x = FGA)) +   geom_histogram(binwidth = 3) +
  labs(x = "FGA",
       y = "Count",
       title = "FG Attempted Distribution")
```
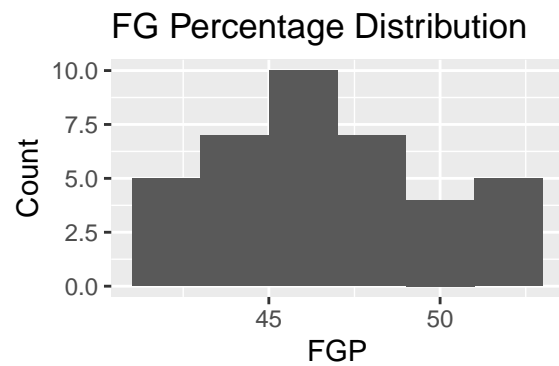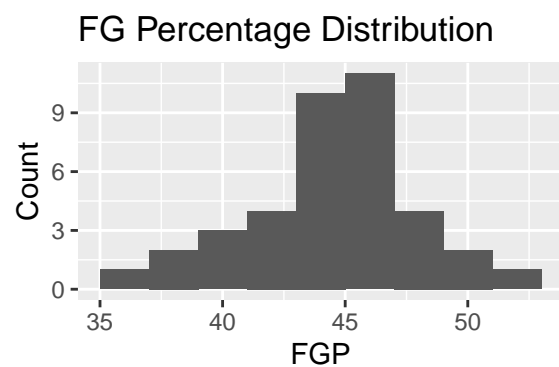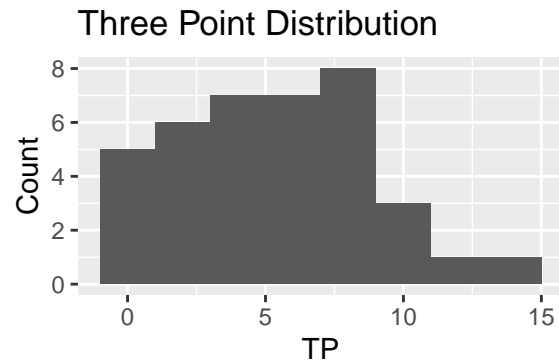
FG Attempted Distribution


```
ggplot(data = useful_data %>% filter(win == 0), aes(x = FGA)) +   geom_histogram(binwidth = 3) +
  labs(x = "FGA",
       y = "Count",
       title = "FG Attempted Distribution")
```

FG Attempted Distribution


```
ggplot(data = useful_data %>% filter(win == 1), aes(x = FGP)) +   geom_histogram(binwidth = 2) +
  labs(x = "FGP",
       y = "Count",
       title = "FG Percentage Distribution")
```
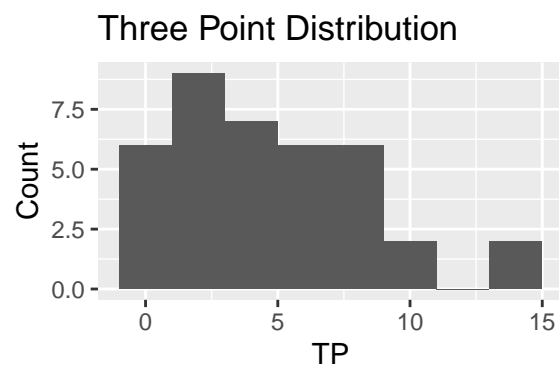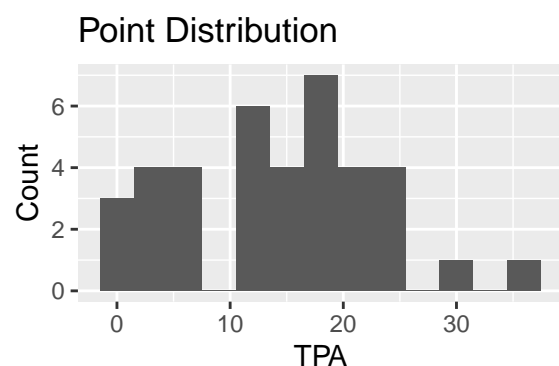
FG Percentage Distribution


```
ggplot(data = useful_data %>% filter(win == 0), aes(x = FGP)) +   geom_histogram(binwidth = 2) +
  labs(x = "FGP",
       y = "Count",
       title = "FG Percentage Distribution")
```

FG Percentage Distribution


```
ggplot(data = useful_data %>% filter(win == 1), aes(x = TP)) +   geom_histogram(binwidth = 2) +
  labs(x = "TP",
       y = "Count",
       title = "Three Point Distribution")
```

Three Point Distribution


```
ggplot(data = useful_data %>% filter(win == 0), aes(x = TP)) +   geom_histogram(binwidth = 2) +
  labs(x = "TP",
       y = "Count",
       title = "Three Point Distribution")
```

Three Point Distribution
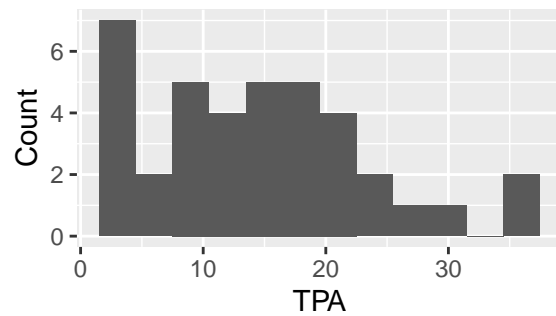


```
ggplot(data = useful_data %>% filter(win == 1), aes(x = TPA)) +   geom_histogram(binwidth = 3) +
  labs(x = "TPA",
       y = "Count",
       title = "Point Distribution")
```
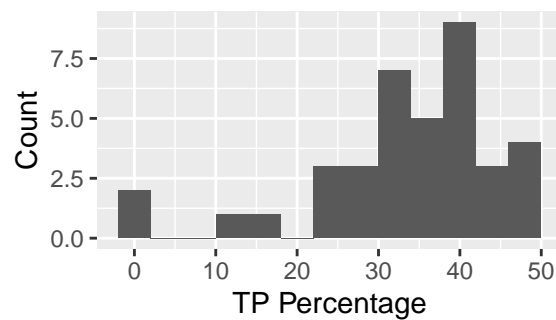
Point Distribution


```
ggplot(data = useful_data %>% filter(win == 0), aes(x = TPA)) +   geom_histogram(binwidth = 3) +
  labs(x = "TPA",
       y = "Count",
       title = "TP Attempted Distribution")
```

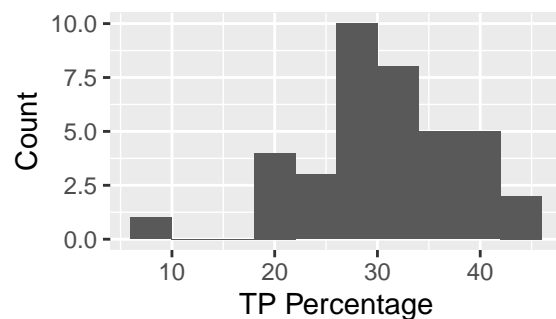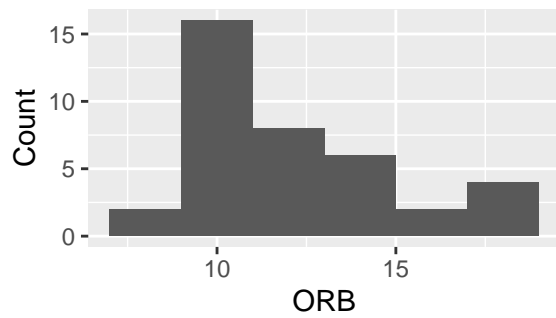## TP Attempted Distribution


```
ggplot(data = useful_data %>% filter(win == 1), aes(x = TPP)) +   geom_histogram(binwidth = 4) +
  labs(x = "TP Percentage",
       y = "Count",
       title = "TP Percentage Distribution")
```

TP Percentage Distribution


```
ggplot(data = useful_data %>% filter(win == 0), aes(x = TPP)) +   geom_histogram(binwidth = 4) +
  labs(x = "TP Percentage",
       y = "Count",
       title = "TP Percentage Distribution")
```

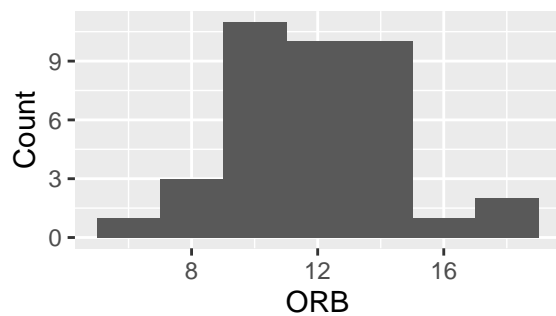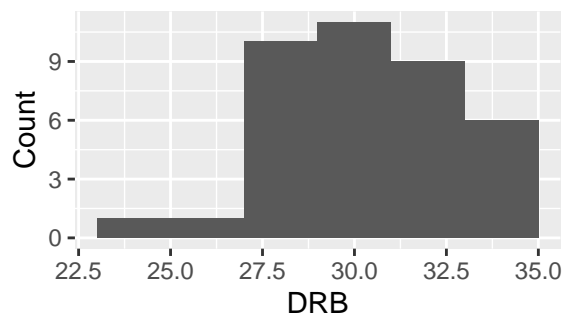## TP Percentage Distribution


```
ggplot(data = useful_data %>% filter(win == 1), aes(x = ORB)) +   geom_histogram(binwidth = 2) +
  labs(x = "ORB",
       y = "Count",
       title = "Offensive Rebounds Distribution")
```
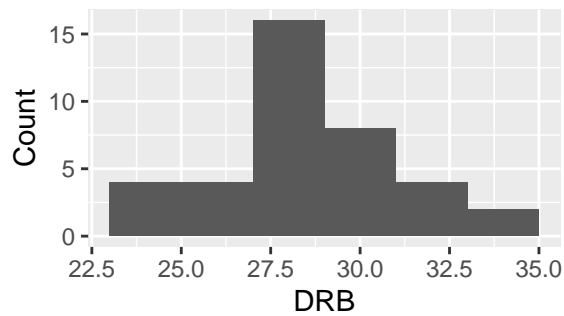
Offensive Rebounds Distribution


```
ggplot(data = useful_data %>% filter(win == 0), aes(x = ORB)) +   geom_histogram(binwidth = 2) +
  labs(x = "ORB",
       y = "Count",
       title = "Offensive Rebounds Distribution")
```
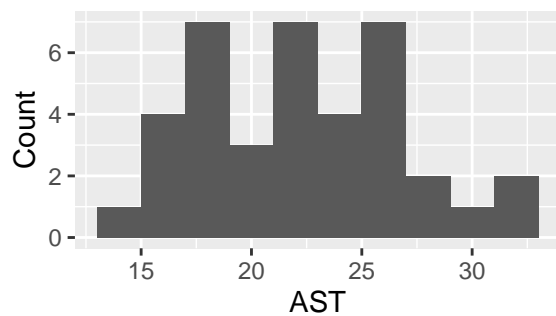
Offensive Rebounds Distribution


```
ggplot(data = useful_data %>% filter(win == 1), aes(x = DRB)) +   geom_histogram(binwidth = 2) +
  labs(x = "DRB",
       y = "Count",
       title = "Defensive Rebound Distribution")
```

Defensive Rebound Distribution


```
ggplot(data = useful_data %>% filter(win == 0), aes(x = DRB)) +   geom_histogram(binwidth = 2) +
  labs(x = "DRB",
       y = "Count",
       title = "Defensive Rebound Distribution")
```

Defensive Rebound Distribution
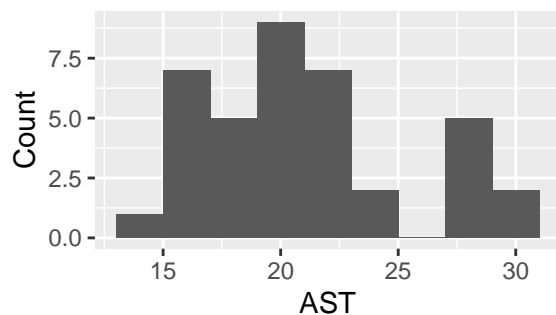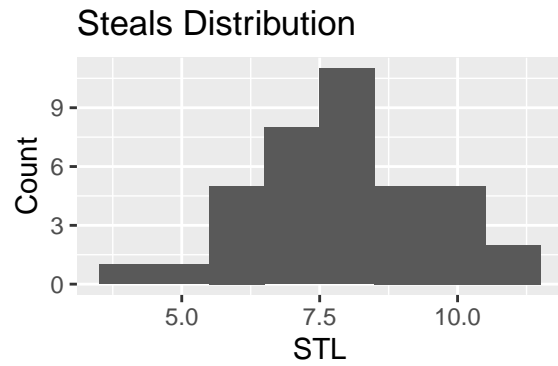


```
ggplot(data = useful_data %>% filter(win == 1), aes(x = AST)) +   geom_histogram(binwidth = 2) +
  labs(x = "AST",
       y = "Count",
       title = "Assists Distribution")
```

Assists Distribution


```
ggplot(data = useful_data %>% filter(win == 0), aes(x = AST)) +   geom_histogram(binwidth = 2) +
  labs(x = "AST",
       y = "Count",
       title = "Assists Distribution")
```
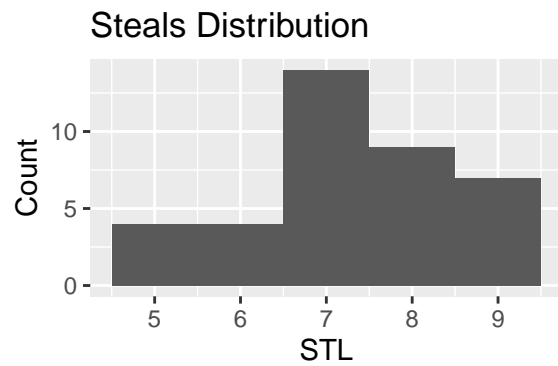
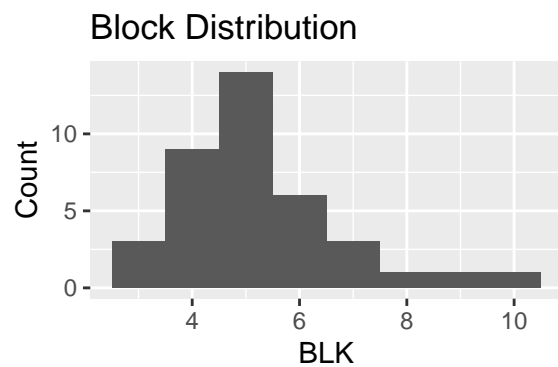## Assists Distribution


```
ggplot(data = useful_data %>% filter(win == 1), aes(x = STL)) +   geom_histogram(binwidth = 1) +
  labs(x = "STL",
       y = "Count",
       title = "Steals Distribution")
```
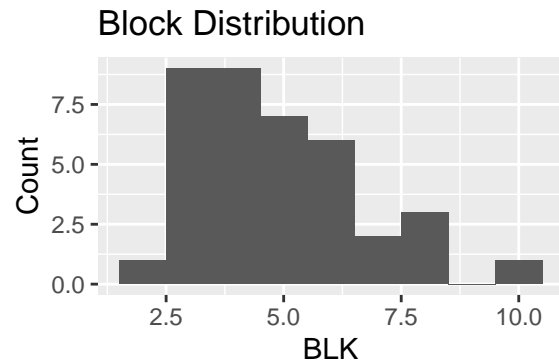
Steals Distribution


```
ggplot(data = useful_data %>% filter(win == 0), aes(x = STL)) +   geom_histogram(binwidth = 1) +
  labs(x = "STL",
       y = "Count",
       title = "Steals Distribution")
```
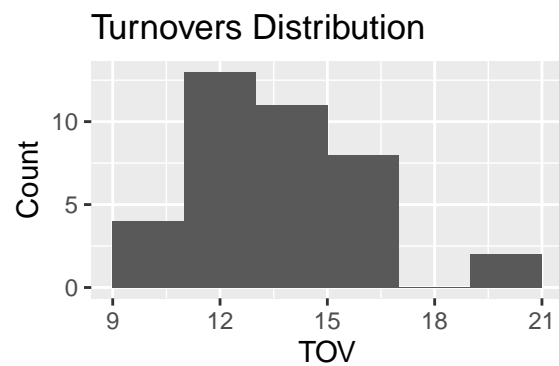
Steals Distribution


```
ggplot(data = useful_data %>% filter(win == 1), aes(x = BLK)) +   geom_histogram(binwidth = 1) +
  labs(x = "BLK",
       y = "Count",
       title = "Block Distribution")
```
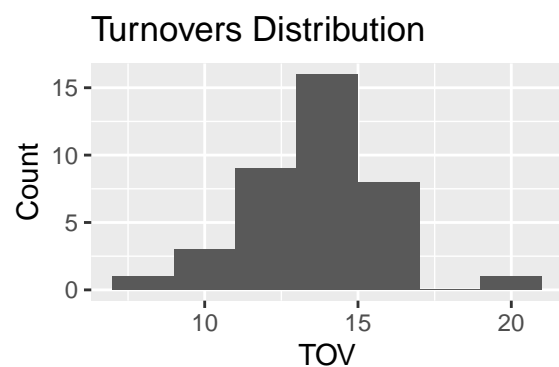
Block Distribution


```
ggplot(data = useful_data %>% filter(win == 0), aes(x = BLK)) +   geom_histogram(binwidth = 1) +
  labs(x = "BLK",
       y = "Count",
       title = "Block Distribution")
```
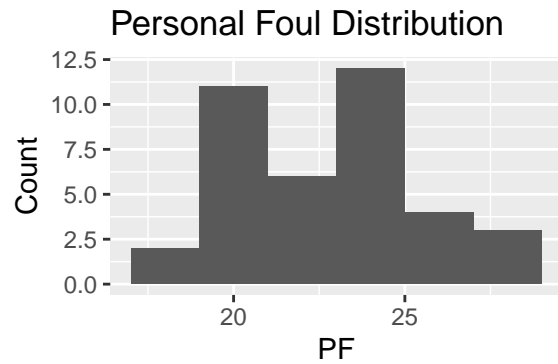
Block Distribution


```r
ggplot(data = useful_data %>% filter(win == 1), aes(x = TOV)) +   geom_histogram(binwidth = 2) +
  labs(x = "TOV",
       y = "Count",
       title = "Turnovers Distribution")
```

Turnovers Distribution


```r
ggplot(data = useful_data %>% filter(win == 0), aes(x = TOV)) +   geom_histogram(binwidth = 2) +
  labs(x = "TOV",
       y = "Count",
       title = "Turnovers Distribution")
```

Turnovers Distribution


```r
ggplot(data = useful_data %>% filter(win == 1), aes(x = PF)) +   geom_histogram(binwidth = 2) +
  labs(x = "PF",
       y = "Count",
       title = "Personal Foul Distribution")
```

Personal Foul Distribution


```
ggplot(data = useful_data %>% filter(win == 0), aes(x = PF)) +   geom_histogram(binwidth = 2) +
  labs(x = "PF",
       y = "Count",
       title = "Personal Foul Distribution")
```

Personal Foul Distribution
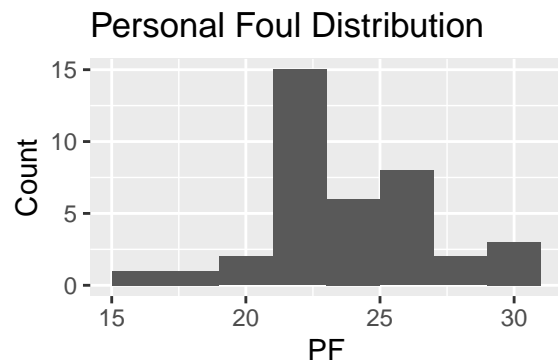
Model Refinement

Now that we have checked that the distributions of the variables are somewhat normal, we can create a model and check the residuals. Our model will be a binomial model (only options are 0 ad 1). The first step will be to plot the residuals of each variable in the model to check the linearity assumption. Then, we will plot the Cook's distance and remove any high-leverage points. Finally, the VIF will be checked and any variables with a high VIF will be removed from the model.

```
useful_data$win <- as.factor(useful_data$win)

model <- glm(win ~ PTS + FG + FGA + FGP + TP + TPA + TPP + ORB + DRB + AST + STL + BLK + TOV + PF, usefu

summary(model)
```

```
##
## Call:
## glm(formula = win ~ PTS + FG + FGA + FGP + TP + TPA + TPP + ORB +
##     DRB + AST + STL + BLK + TOV + PF, family = binomial, data = useful_data)
##
## Deviance Residuals:
##     Min       1Q    Median       3Q      Max
```

```
## -2.09816  -0.55087  -0.01365   0.65435   1.95116
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -11.37225   91.05303  -0.125  0.90061
## PTS           0.05827    0.11315   0.515  0.60658
## FG            0.36495    2.31146   0.158  0.87454
## FGA          -0.63662    1.05228  -0.605  0.54519
## FGP           0.33684    1.96123   0.172  0.86363
## TP            0.14958    0.73911   0.202  0.83962
## TPA           0.01954    0.27121   0.072  0.94257
## TPP           0.04157    0.08372   0.496  0.61956
## ORB           0.98424    0.30193   3.260  0.00111 **
## DRB           0.46885    0.19354   2.422  0.01542 *
## AST          -0.02027    0.14192  -0.143  0.88642
## STL           0.71154    0.29829   2.385  0.01706 *
## BLK           0.01150    0.33301   0.035  0.97246
## TOV          -0.25649    0.19228  -1.334  0.18223
## PF           -0.05103    0.15307  -0.333  0.73887
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 105.358  on 75  degrees of freedom
## Residual deviance:  58.324  on 61  degrees of freedom
## AIC: 88.324
##
## Number of Fisher Scoring iterations: 6
```

```
model_data <- augment(model, useful_data)
head(model_data)
```

```
## # A tibble: 6 x 21
##     PTS    FG   FGA   FGP    TP   TPA   TPP   ORB   DRB   AST   STL   BLK   TOV
##   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 110.   45    92   48.9 0     0.667  0     17.2  34.2  26.7  9.17  6.17  20
## 2  96.5  40.2  85.3 47.1 0.5   2.83  17.6  16.7  30.7  22.8  6.67  5.33  16.8
## 3 112.   45.5  91.8 49.5 0.167 1.5   11.1  18.3  29    31.3 10.7   7     19.3
## 4 110.   43    88.5 48.6 0     0.75   0     18    30    26   11     8     17
## 5 116    42    92.9 45.2 1.29  3.29  39.1  17.4  30    24.1  9.86  4.57  15.9
## 6 116.   46.3  90.5 51.2 1.33  3.83  34.8  11    31.7  32    9.17  4     12.5
## # ... with 8 more variables: PF <dbl>, win <fct>, .fitted <dbl>, .resid <dbl>,
## #   .std.resid <dbl>, .hat <dbl>, .sigma <dbl>, .cooksd <dbl>
```
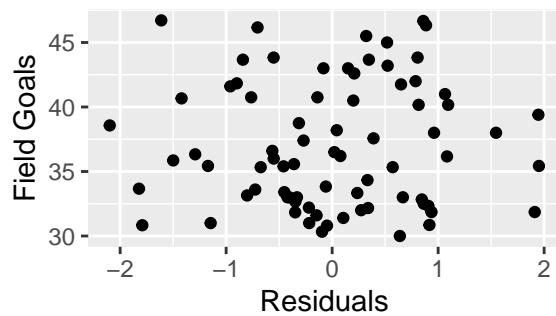
```
ggplot(data = model_data, aes(x=.resid, y=PTS)) + geom_point() +
  labs(x="Residuals",
       y="Points",
       title="Residual Plot of Points")
```

Residual Plot of Points
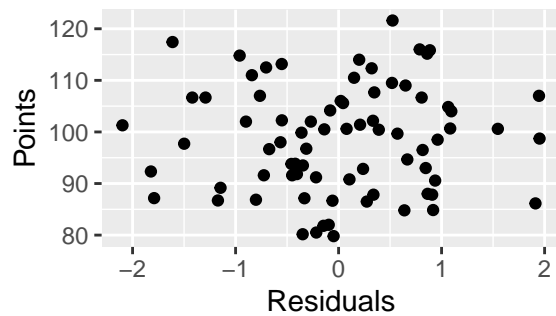

```
ggplot(data = model_data, aes(x=.resid, y=FG)) + geom_point() +
  labs(x="Residuals",
       y="Field Goals",
       title="Residual Plot of Field Goals")
```
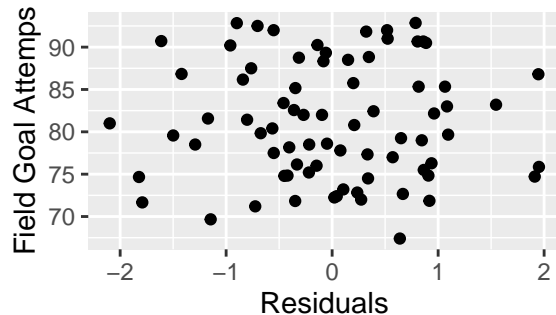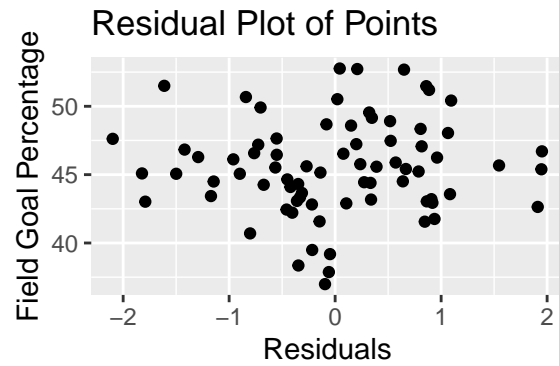
Residual Plot of Field Goals


```
ggplot(data = model_data, aes(x=.resid, y=FGA)) + geom_point() +
  labs(x="Residuals",
       y="Field Goal Attemps",
       title="Residual Plot of Field Goal Attempts")
```
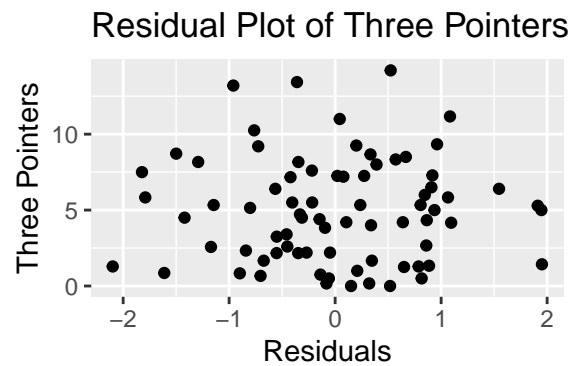
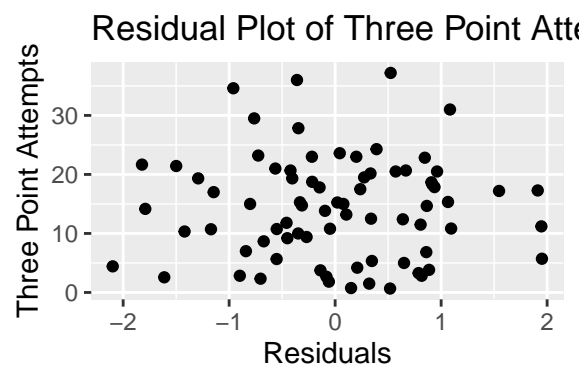## Residual Plot of Field Goal Atteı


```
ggplot(data = model_data, aes(x=.resid, y=FGP)) + geom_point() +
  labs(x="Residuals",
       y="Field Goal Percentage",
       title="Residual Plot of Points")
```

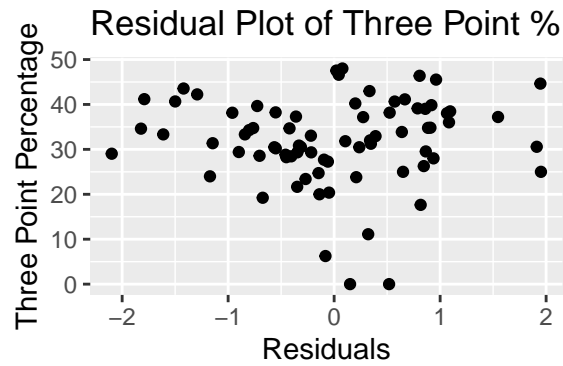## Residual Plot of Points


```
ggplot(data = model_data, aes(x=.resid, y=TP)) + geom_point() +
  labs(x="Residuals",
       y="Three Pointers",
       title="Residual Plot of Three Pointers")
```

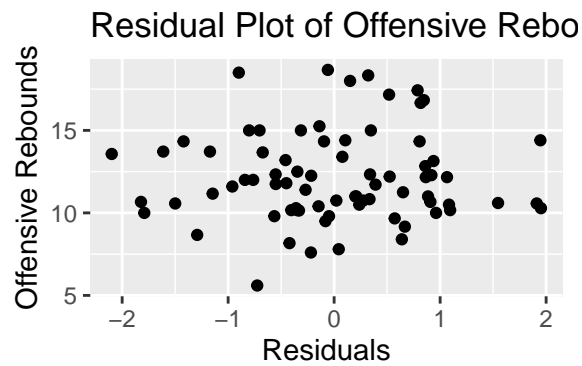## Residual Plot of Three Pointers


```
ggplot(data = model_data, aes(x=.resid, y=TPA)) + geom_point() +
  labs(x="Residuals",
       y="Three Point Attempts",
       title="Residual Plot of Three Point Attemps")
```

Residual Plot of Three Point Att(
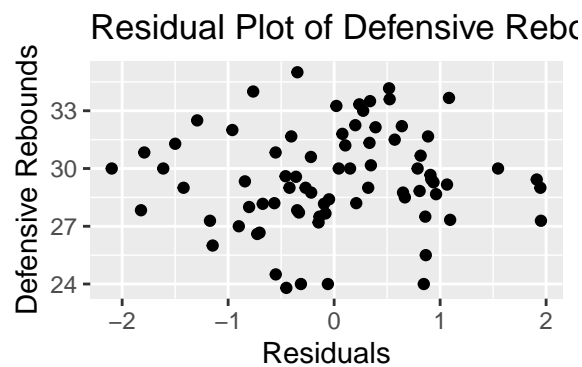


```
ggplot(data = model_data, aes(x=.resid, y=TPP)) + geom_point() +
  labs(x="Residuals",
       y="Three Point Percentage",
       title="Residual Plot of Three Point %")
```

Residual Plot of Three Point %


```r
ggplot(data = model_data, aes(x=.resid, y=ORB)) + geom_point() +
  labs(x="Residuals",
       y="Offensive Rebounds",
       title="Residual Plot of Offensive Rebounds")
```
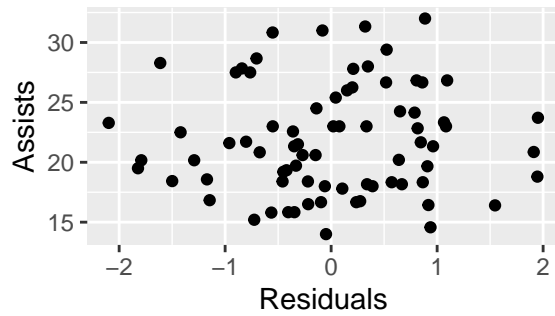
Residual Plot of Offensive Rebo


```r
ggplot(data = model_data, aes(x=.resid, y=DRB)) + geom_point() +
  labs(x="Residuals",
       y="Defensive Rebounds",
       title="Residual Plot of Defensive Rebounds")
```
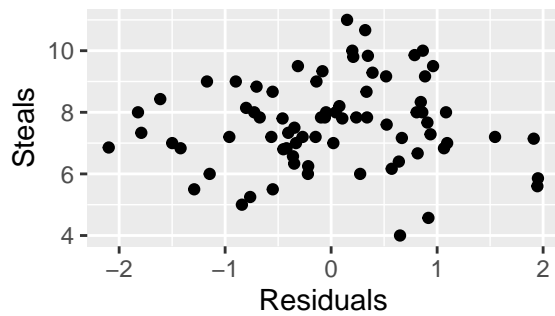
Residual Plot of Defensive Rebo


```r
ggplot(data = model_data, aes(x=.resid, y=AST)) + geom_point() +
  labs(x="Residuals",
       y="Assists",
       title="Residual Plot of Assists")
```
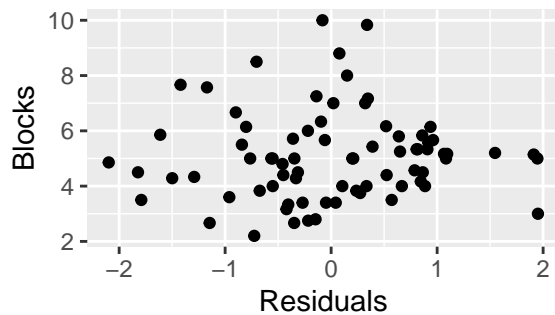
Residual Plot of Assists

```
ggplot(data = model_data, aes(x=.resid, y=STL)) + geom_point() +
  labs(x="Residuals",
       y="Steals",
       title="Residual Plot of Steals")
```
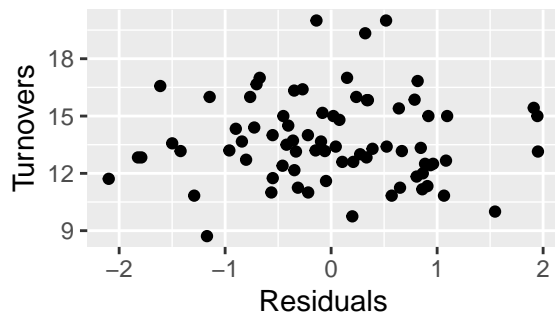
Residual Plot of Steals

```
ggplot(data = model_data, aes(x=.resid, y=BLK)) + geom_point() +
  labs(x="Residuals",
       y="Blocks",
       title="Residual Plot of Blocks")
```

Residual Plot of Blocks
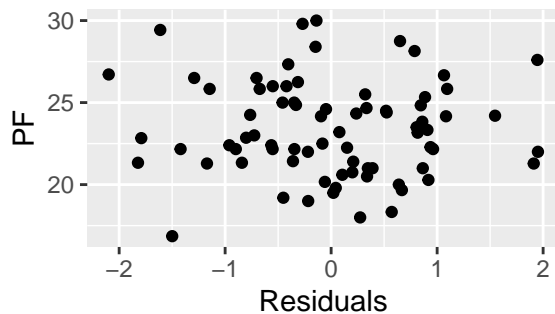
```
ggplot(data = model_data, aes(x=.resid, y=TOV)) + geom_point() +
  labs(x="Residuals",
       y="Turnovers",
       title="Residual Plot of Turnovers")
```

Residual Plot of Turnovers


```
ggplot(data = model_data, aes(x=.resid, y=PF)) + geom_point() +
  labs(x="Residuals",
       y="PF",
       title="Residual Plot of PF")
```

Residual Plot of PF

The residual plots for each variable appear to be random and evenly dispersed, which means that the linearity assumption is satisfied. Before we can test the accuracy of the model, we must also explore how these observations affect the model, and how the variables used in the model affect each other. First, we can plot the leverage (.cooksd) of each observation to see if there are any high-leverage data points.

```
ggplot(data = model_data, aes(x=seq.int(nrow(model_data)), y=.cooksd)) + geom_point() +
  labs(x="Observation",
       y="Cook's Distance",
       title="Cook's Distance of each Observation") +
  geom_hline(yintercept=0.125)
```

Cook's Distance of each Obser

18
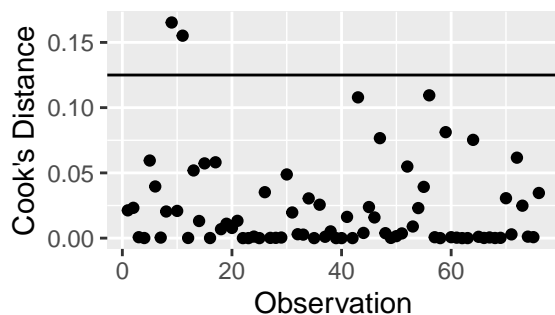
It is obvious that there are two high-leverage data points. If we use a threshold of 0.125, we can eliminate these two high-leverage points to make the model better at prediction.

```
filter_data <- filter(model_data, .cooksd < 0.125)
```

The next step is to check how the variables interact with each other.