Predicting NBA Championship Winners Using Data from Past Championship Series

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)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.1.1     v forcats 0.5.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(broom)
library(pROC)
```

```
## Type ’citation("pROC")’ for a citation.
```

```
##
## Attaching package: ’pROC’
```

```
## The following objects are masked from ’package:stats’:
##
##     cov, smooth, var
```

```
library(plotROC)
```

```
##
## Attaching package: ’plotROC’
```

```
## The following object is masked from ’package:pROC’:
##
##     ggroc
```

```
library(rms)
```

```
## Loading required package: Hmisc
```

```
## Loading required package: lattice
```

```
## Loading required package: survival
```

```
## Loading required package: Formula
```

```
##
## Attaching package: ’Hmisc’

## The following objects are masked from ’package:dplyr’:
##
##     src, summarize

## The following objects are masked from ’package:base’:
##
##     format.pval, units

## Loading required package: SparseM

##
## Attaching package: ’SparseM’

## The following object is masked from ’package:base’:
##
##     backsolve
```

```
library(caret)
```

```
##
## Attaching package: ’caret’

## The following object is masked from ’package:survival’:
##
##     cluster

## The following object is masked from ’package:purrr’:
##
##     lift
```

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.

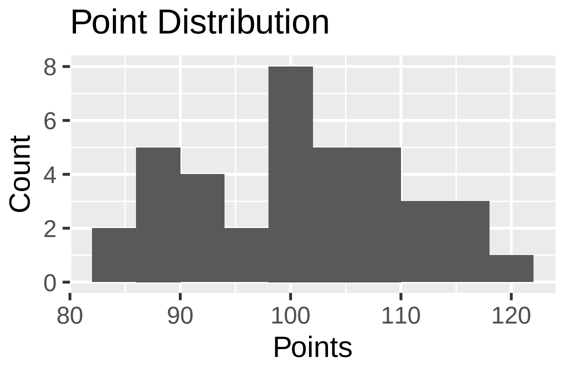
```
useful_data = subset(all_data, select = -c(...1,Year, Status, Team, FT, FTA, FTP, TRB) )
summary(useful_data$FGA)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   67.40   75.42   80.60   81.18   87.00   92.86
```

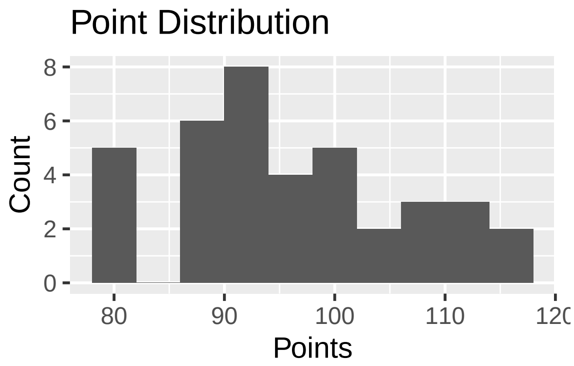
Now, we can start our EDA. First, we check the plots of each variable for both the losers and the winners to make sure each distribution is roughly normal.

Distribution of Points:

```
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")
```



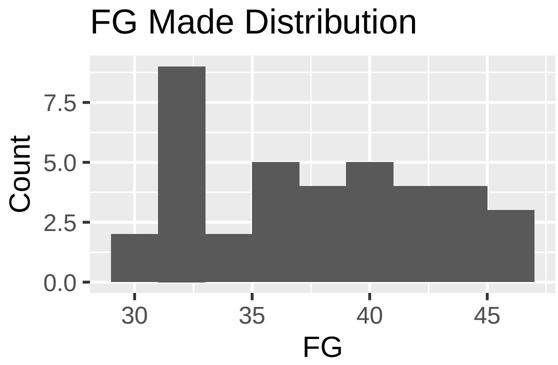
Summary of Points:

```
summary(useful_data$PTS)
```

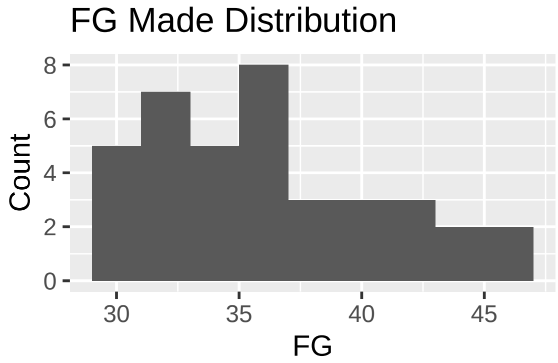
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   79.80   90.74   99.19   98.56  106.67  121.60
```

Distribution of Field Goals Made:

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



Summary of Field Goals:

```
summary(useful_data$FG)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   30.00   32.96   36.08   37.02   40.81   46.71
```

The distributions and summaries of the other variables can be seen in the appendix. All of the distributions appear to be somewhat normal with no outliers, We have a large sample size, so we can continue to make our model and check the residuals.

Creating the Model

Model Refinement

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.

```r
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)
```

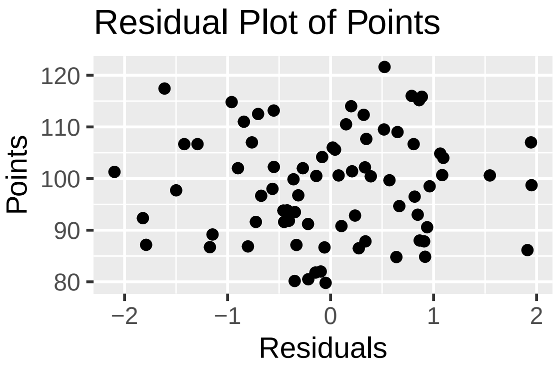
```
##
## Call:
## glm(formula = win ~ PTS + FG + FGA + FGP + TP + TPA + TPP + ORB +
##     DRB + AST + STL + BLK + TOV + PF, family = binomial, data = useful_data)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.09816  -0.55087  -0.01365   0.65435   1.95116
##
## 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
```

```r
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>
```

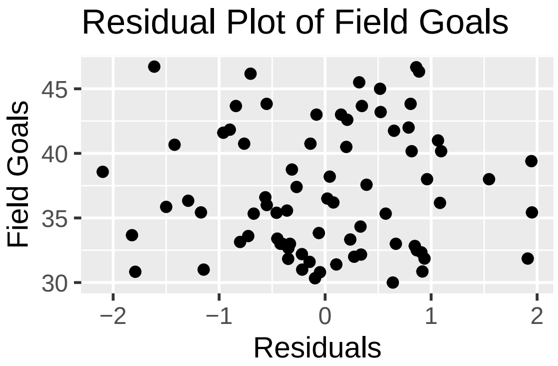
Residual Plot of Points:

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



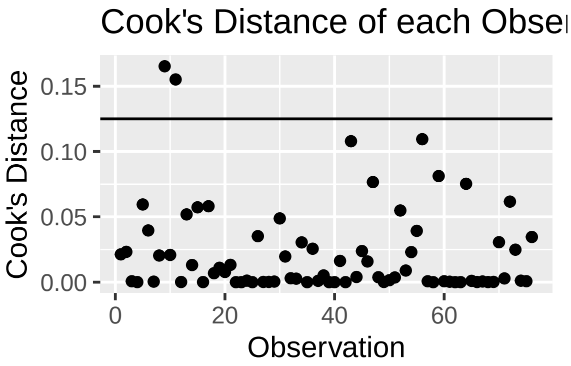
Residual Plot of Field Goals:

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



The residual plots for each variable appear to be random and evenly dispersed (the rest can be seen in the appendix), 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)
```



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. After this, the model must be trained on the newly-filtered data.

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

filter_data <- select(filter_data, 1:15)

filter_model <- glm(win ~ PTS + FG + FGA + FGP + TP +
                      TPA + TPP + ORB + DRB + AST + STL + BLK + TOV + PF, filter_data, family=binomial)

summary(filter_model)
```

```
##
## Call:
## glm(formula = win ~ PTS + FG + FGA + FGP + TP + TPA + TPP + ORB +
##     DRB + AST + STL + BLK + TOV + PF, family = binomial, data = filter_data)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.12838  -0.32438  -0.00962   0.66696   2.05077
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -13.34302   98.44346  -0.136  0.89218
## PTS          -0.03651    0.13731  -0.266  0.79033
## FG            1.03938    2.46225   0.422  0.67293
## FGA          -1.02089    1.12581  -0.907  0.36451
## FGP           0.30264    2.11119   0.143  0.88601
## TP           -0.32034    0.89001  -0.360  0.71890
## TPA           0.28922    0.33879   0.854  0.39328
## TPP           0.09593    0.10564   0.908  0.36384
## ORB           1.30905    0.41405   3.162  0.00157 **
## DRB           0.72101    0.28588   2.522  0.01166 *
## AST          -0.05620    0.18053  -0.311  0.75559
## STL           1.20375    0.42457   2.835  0.00458 **
## BLK           0.16852    0.38747   0.435  0.66362
## TOV          -0.29451    0.21214  -1.388  0.16506
## PF           -0.01274    0.18549  -0.069  0.94523
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 102.532  on 73  degrees of freedom
## Residual deviance:  47.107  on 59  degrees of freedom
## AIC: 77.107
##
## Number of Fisher Scoring iterations: 7
```

```
filter_data <- augment(filter_model, filter_data)
```

We must first re_train the model on the filter data The next step is to check how the variables interact with each other. To measure this, we want to calculate the Variable Inflation Factor, or VIF.

```
vif(filter_model)
```

```
##         PTS          FG         FGA         FGP          TP         TPA
##   14.554416 1116.928102  468.327785  307.483725   68.519839   65.480931
##         TPP         ORB         DRB         AST         STL         BLK
##    5.335238    8.903163    3.340391    4.573557    2.795145    1.839274
##         TOV          PF
##    1.638405    2.045241
```

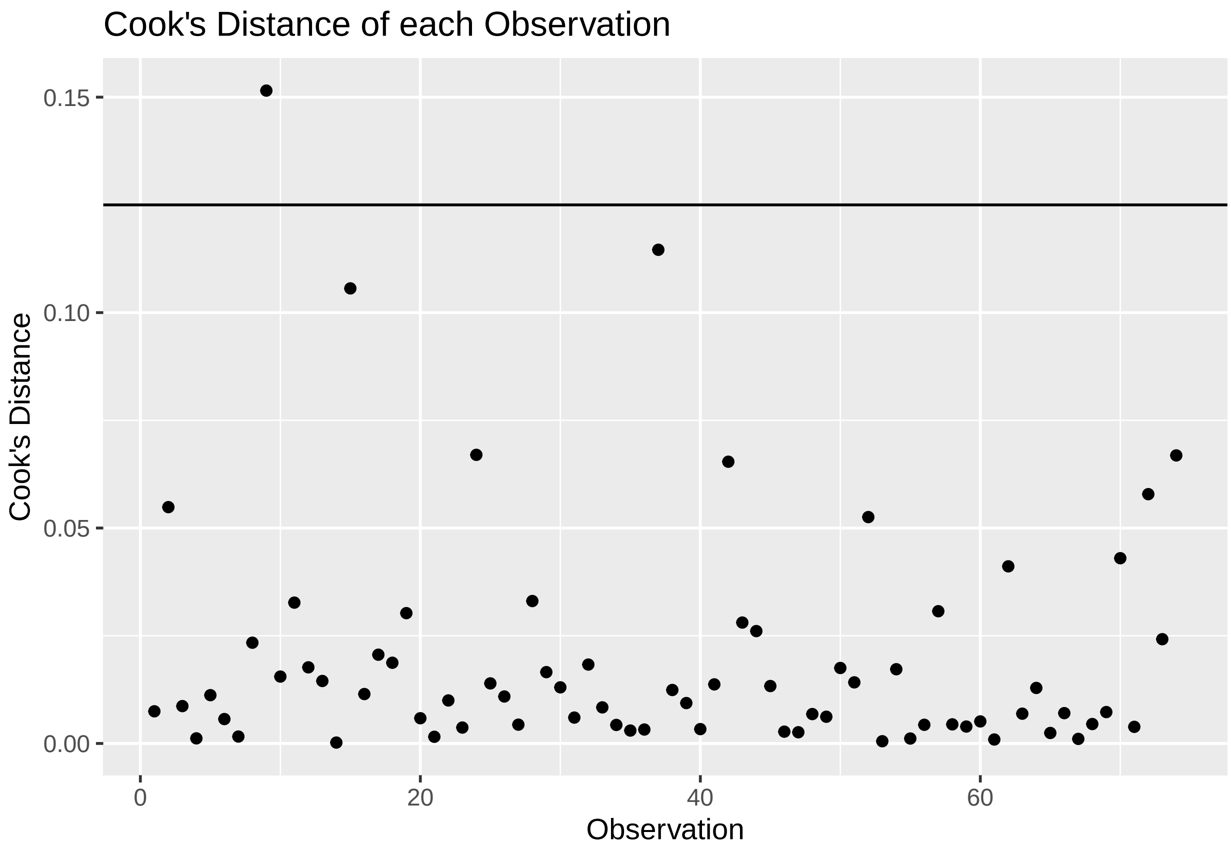
Some of the variables have a very high VIF, so we can only include certain variables to keep the score lower. To see which variables are better included and not included, we must create multiple models and assess which one has the best accuracy. We can leave all of the variables who's VIF is lower than 10, but the others must be removed for higher accuracy. We will create three models, one that will include only the Field Goals made and the Three Point shots made; one that will only include the Field Goal percentage and Three Point percentage; and finally an attempts model that will include the number of Field Goal attempts and Three Point attempts.

```
points_model <- glm(win ~ PTS + FG + TP + ORB + DRB + AST + STL + BLK + TOV + PF,
                    data = filter_data, na.action = na.omit, family = binomial)

percentage_model <- glm(win ~ PTS + FGP + TPP + ORB + DRB + AST + STL + BLK + TOV + PF,
                    data = filter_data, na.action = na.omit, family = binomial)

attempts_model <- glm(win ~ PTS + FGA + TPA + ORB + DRB + AST + STL + BLK + TOV + PF,
                    data = filter_data, na.action = na.omit, family = binomial)
```

Since we created three new models, we need to repeat the same steps we did before with our previous model. We need to check for high leverage points and the Variable Inflation Factor.

Points Model:

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



```
points_data <- filter(points_data, .cooksd < 0.125)

points_data <- select(points_data, 1:15)

points_model <- glm(win ~ PTS + FG + TP + ORB + DRB + AST + STL + BLK + TOV + PF, points_data, family=bi

summary(points_model)
```

```
##
## Call:
## glm(formula = win ~ PTS + FG + TP + ORB + DRB + AST + STL + BLK +
##     TOV + PF, family = binomial, data = points_data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0296  -0.8185  -0.2856   0.8062   1.9676
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -9.367459   5.581552  -1.678   0.0933 .
## PTS          0.028799   0.099409   0.290   0.7720
## FG          -0.122202   0.227124  -0.538   0.5905
## TP           0.006431   0.135718   0.047   0.9622
## ORB          0.066055   0.134003   0.493   0.6221
## DRB          0.328022   0.143179   2.291   0.0220 *
## AST          0.094175   0.128648   0.732   0.4641
## STL          0.644231   0.272177   2.367   0.0179 *
## BLK          0.011796   0.223029   0.053   0.9578
## TOV         -0.112954   0.168900  -0.669   0.5036
## PF          -0.215486   0.123094  -1.751   0.0800 .
## ---
## Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 95.640  on 68  degrees of freedom
## Residual deviance: 72.872  on 58  degrees of freedom
## AIC: 94.872
##
## Number of Fisher Scoring iterations: 4
```

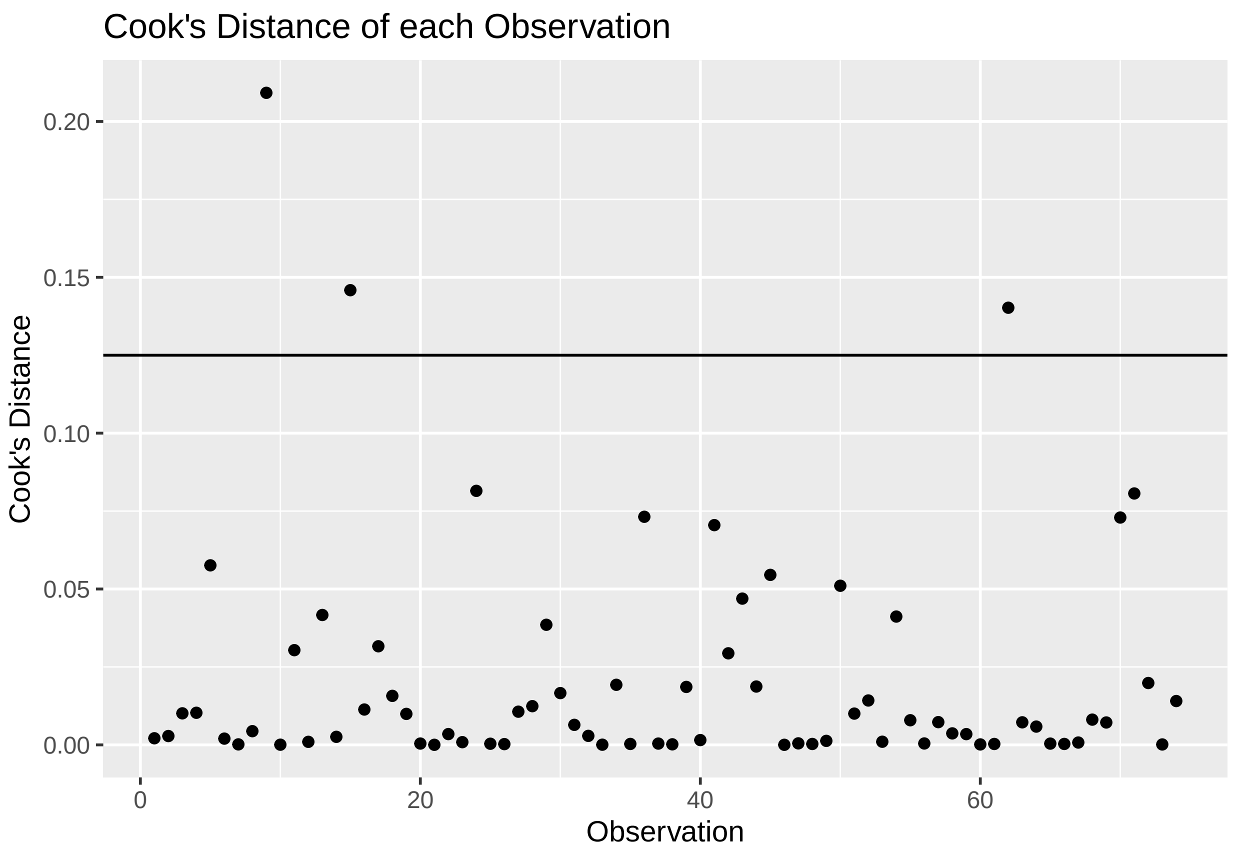
```
points_data <- augment(points_model, points_data)
```

```
vif(points_model)
```

```
##       PTS        FG        TP       ORB       DRB       AST       STL       BLK
## 11.390655 14.142110  2.622132  1.623297  1.544988  4.126773  1.391867  1.460206
##       TOV        PF
##  1.485508  1.210932
```

Percentage Model:

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



```
percentage_data <- filter(percentage_data, .cooksd < 0.125)

percentage_data <- select(percentage_data, 1:15)

percentage_model <- glm(win ~ PTS + FGP + TPP + ORB + DRB + AST + STL + BLK + TOV + PF, percentage_data

summary(percentage_model)
```

```
##
## Call:
## glm(formula = win ~ PTS + FGP + TPP + ORB + DRB + AST + STL +
##     BLK + TOV + PF, family = binomial, data = percentage_data)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.81294  -0.59238  -0.06549   0.53815   2.73812
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -47.929386  14.743572  -3.251 0.001151 **
## PTS          -0.265332   0.095819  -2.769 0.005621 **
## FGP           0.926183   0.276969   3.344 0.000826 ***
## TPP           0.106139   0.065706   1.615 0.106235
## ORB           0.479028   0.236989   2.021 0.043248 *
## DRB           0.716774   0.226651   3.162 0.001564 **
## AST          -0.088665   0.142965  -0.620 0.535134
## STL           1.042354   0.363801   2.865 0.004168 **
## BLK          -0.006623   0.336867  -0.020 0.984314
## TOV          -0.199169   0.229544  -0.868 0.385572
## PF           -0.070473   0.155795  -0.452 0.651018
## ---
## Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 95.640  on 68  degrees of freedom
## Residual deviance: 52.363  on 58  degrees of freedom
## AIC: 74.363
##
## Number of Fisher Scoring iterations: 6
```

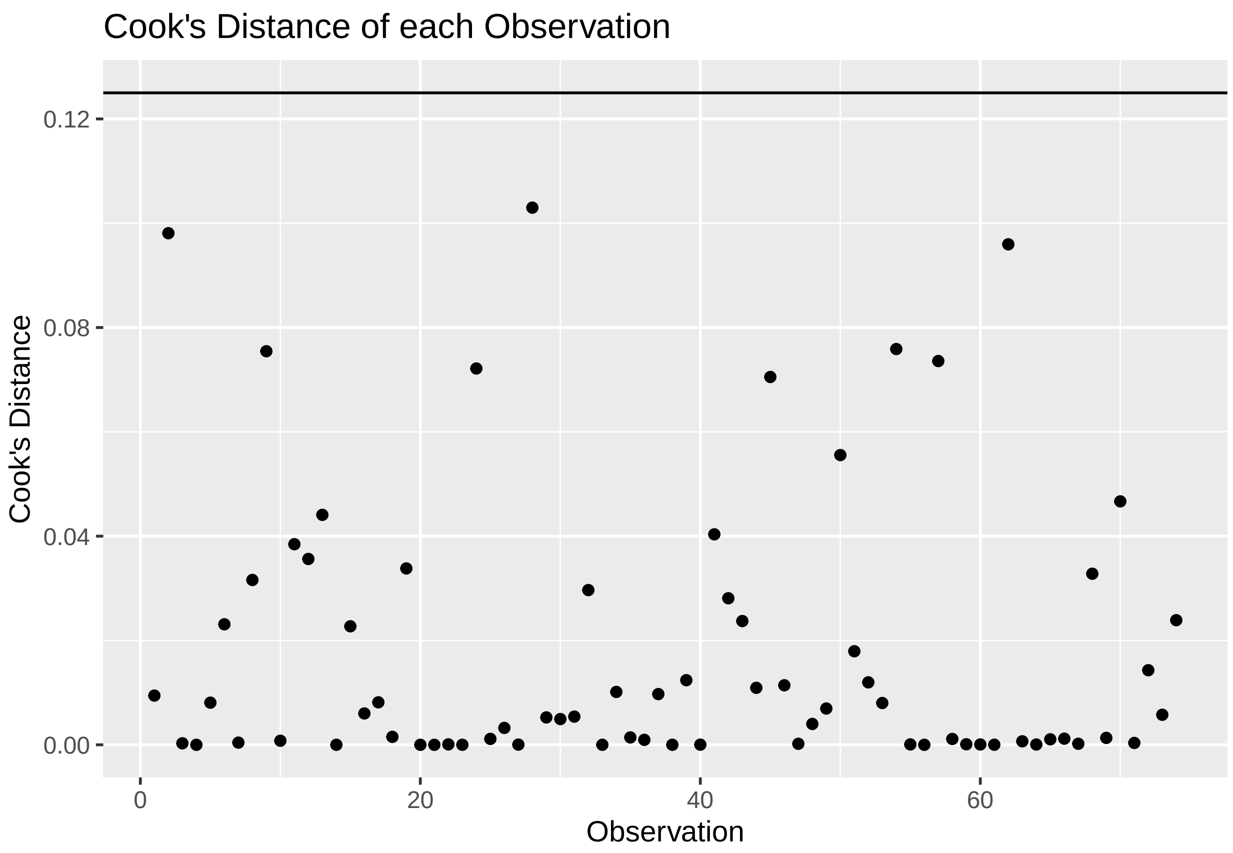
```
percentage_data <- augment(percentage_model, percentage_data)
```

```
vif(percentage_model)
```

```
##      PTS      FGP      TPP      ORB      DRB      AST      STL      BLK
## 7.778355 6.031811 2.582414 3.365817 2.432547 3.760035 1.839702 1.860066
##      TOV       PF
## 1.974326 1.262596
```

Attempts Model:

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



```
attempts_data <- filter(attempts_data, .cooksd < 0.125)

attempts_data <- select(attempts_data, 1:15)

attempts_model <- glm(win ~ PTS + FGA + TPA + ORB + DRB + AST + STL + BLK + TOV + PF, attempts_data, fa

summary(attempts_model)
```

```
##
## Call:
## glm(formula = win ~ PTS + FGA + TPA + ORB + DRB + AST + STL +
##     BLK + TOV + PF, family = binomial, data = attempts_data)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.07293  -0.50037  -0.08972   0.44787   1.87676
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  2.529472   7.986361   0.317 0.751453
## PTS          0.205520   0.093663   2.194 0.028217 *
## FGA         -0.653575   0.180862  -3.614 0.000302 ***
## TPA          0.008557   0.060514   0.141 0.887545
## ORB          0.736764   0.252719   2.915 0.003553 **
## DRB          0.533894   0.193973   2.752 0.005916 **
## AST          0.214568   0.164647   1.303 0.192505
## STL          1.051963   0.374068   2.812 0.004920 **
## BLK          0.362170   0.348814   1.038 0.299136
## TOV         -0.430452   0.228282  -1.886 0.059347 .
## PF          -0.125846   0.160135  -0.786 0.431940
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 95.640  on 68  degrees of freedom
## Residual deviance: 48.045  on 58  degrees of freedom
## AIC: 70.045
##
## Number of Fisher Scoring iterations: 6
```

```
attempts_data <- augment(attempts_model, attempts_data)
```

```
vif(attempts_model)
```

```
##       PTS       FGA       TPA       ORB       DRB       AST       STL       BLK
##  5.569006 10.609734  2.042433  3.035805  1.780642  4.297328  1.661419  1.710561
##       TOV        PF
##  1.550367  1.185355
```

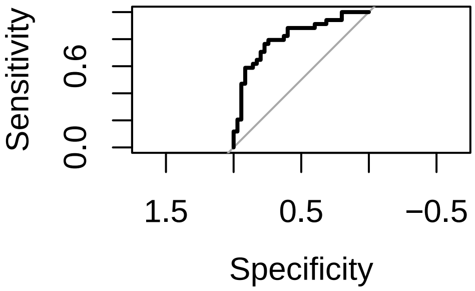
To evaluate the accuracy of each model, we will train each model to our dataset to make predictions and measure the accuracy of these predictions. To do this, for each model we will plot the ROC curve, find the ideal threshold of each model, create predictions, then measure the accuracy.

Points Model:

```
points_roc <- roc(points_data, win, .fitted, plot=TRUE)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```



```
threshold <- coords(points_roc, "best", ret = "threshold")
print(threshold)
```

```
##    threshold
## 1 -0.2346563
```

The ideal threshold for the points model has been shown as `0.2346563`. Using this threshold, we can create a confusion matrix and draw conclusions about the accuracy of the model.

```
points_data <- mutate(points_data, pred = ifelse(.fitted > 0.2346563, 1, 0))

points_data$pred <- as.factor(points_data$pred)

confusionMatrix(points_data$pred, points_data$win)
```

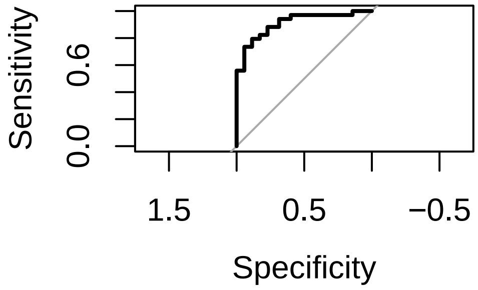
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 28 12
##          1  7 22
##
##                Accuracy : 0.7246
##                  95% CI : (0.6038, 0.8254)
##     No Information Rate : 0.5072
##     P-Value [Acc > NIR] : 0.0001931
##
##                   Kappa : 0.448
##
##  Mcnemar’s Test P-Value : 0.3587954
##
##             Sensitivity : 0.8000
##             Specificity : 0.6471
##          Pos Pred Value : 0.7000
##          Neg Pred Value : 0.7586
##              Prevalence : 0.5072
##          Detection Rate : 0.4058
##    Detection Prevalence : 0.5797
##       Balanced Accuracy : 0.7235
##
##        ’Positive’ Class : 0
##
```

Percentage Model:

```
percentage_roc <- roc(percentage_data, win, .fitted, plot=TRUE)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```



```
threshold <- coords(percentage_roc, "best", ret = "threshold")
print(threshold)
```

```
##   threshold
## 1 0.4107141
```

The ideal threshold for the percentage model has been shown as `0.4107141`. Using this threshold, we can create a confusion matrix and draw conclusions about the accuracy of the model.

```
percentage_data <- mutate(percentage_data, pred = ifelse(.fitted > 0.4107141, 1, 0))

percentage_data$pred <- as.factor(percentage_data$pred)

confusionMatrix(percentage_data$pred, percentage_data$win)
```

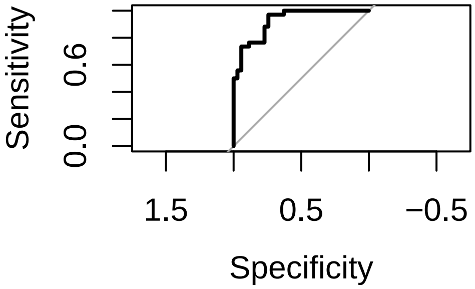
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 31  7
##          1  4 27
##
##                Accuracy : 0.8406
##                  95% CI : (0.7326, 0.9176)
##     No Information Rate : 0.5072
##     P-Value [Acc > NIR] : 7.46e-09
##
##                   Kappa : 0.6807
##
##  Mcnemar's Test P-Value : 0.5465
##
##             Sensitivity : 0.8857
##             Specificity : 0.7941
##          Pos Pred Value : 0.8158
##          Neg Pred Value : 0.8710
##              Prevalence : 0.5072
##          Detection Rate : 0.4493
##    Detection Prevalence : 0.5507
##       Balanced Accuracy : 0.8399
```

```
##
##        'Positive' Class : 0
##
```

Attempts Model:

```
attempts_roc <- roc(attempts_data, win, .fitted, plot=TRUE)
```

```
## Setting levels: control = 0, case = 1

## Setting direction: controls < cases
```



```
threshold <- coords(attempts_roc, "best", ret = "threshold")
print(threshold)
```

```
##   threshold
## 1 -1.059852
```

The ideal threshold for the percentage model has been shown as `-1.059852`. Using this threshold, we can create a confusion matrix and draw conclusions about the accuracy of the model.

```
attempts_data <- mutate(attempts_data, pred = ifelse(.fitted > -1.059852, 1, 0))

attempts_data$pred <- as.factor(attempts_data$pred)

confusionMatrix(attempts_data$pred, attempts_data$win)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 26  1
##          1  9 33
##
##                Accuracy : 0.8551
##                  95% CI : (0.7496, 0.9283)
##     No Information Rate : 0.5072
##     P-Value [Acc > NIR] : 1.398e-09
```

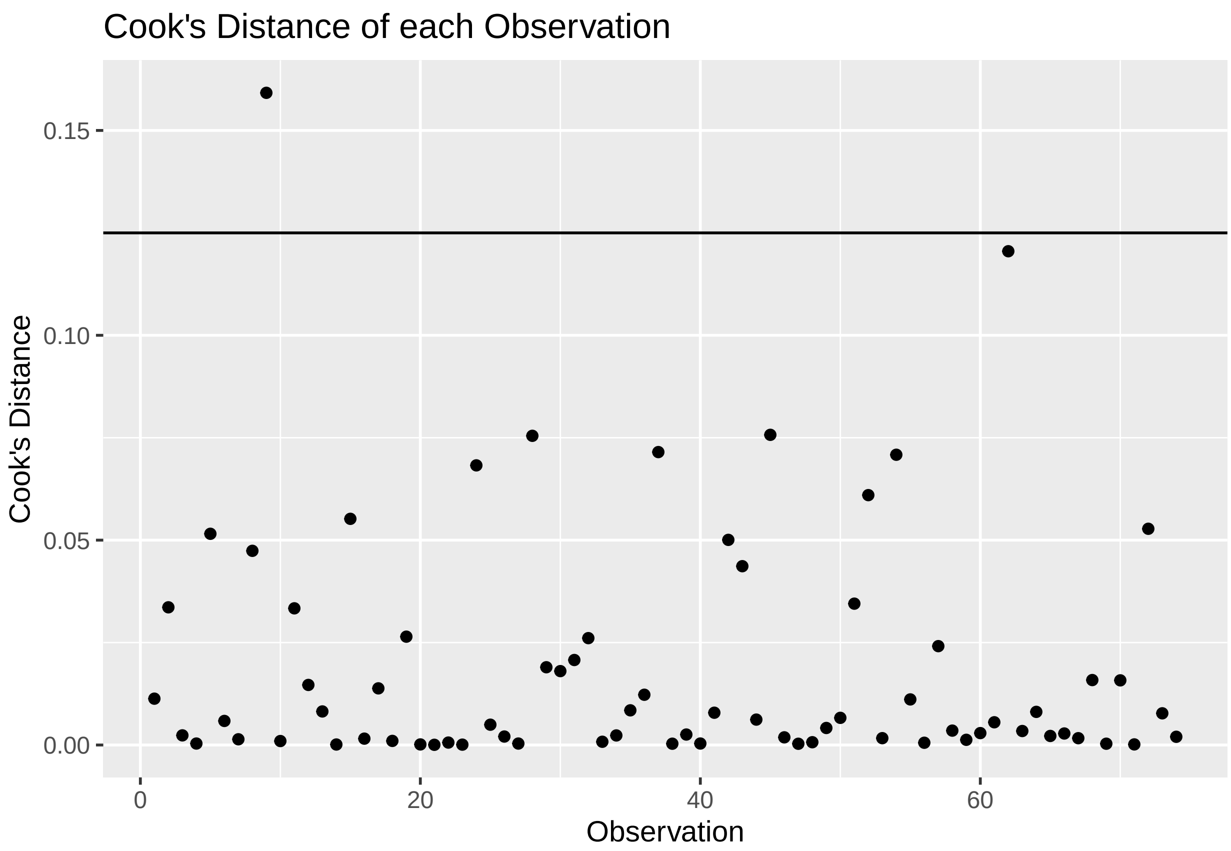
```
##
##                   Kappa : 0.7111
##
##  Mcnemar's Test P-Value : 0.02686
##
##             Sensitivity : 0.7429
##             Specificity : 0.9706
##          Pos Pred Value : 0.9630
##          Neg Pred Value : 0.7857
##              Prevalence : 0.5072
##          Detection Rate : 0.3768
##    Detection Prevalence : 0.3913
##       Balanced Accuracy : 0.8567
##
##        'Positive' Class : 0
##
```

The accuracy of the points model was found to be 72.46%. Using this same method for the other models, we found that the accuracy of the percentage model was 84.06% and the accuracy of the attempts model was 85.51%. In this case, solely using the attempts of field goals and three pointers made the model more accurate. We will use the attempts model as our current model for the next step of tests. Since the VIF was close to 10 for all the models with points, we are going to try a model without PTS. We will use the attempts model since it produced the highest accuracy from our previous models

No points Model:

```
nopoints_model <- glm(win ~ FGA + TPA + ORB + DRB + AST + STL + BLK + TOV + PF,
                    data = filter_data, na.action = na.omit, family = binomial)
```

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



```
nopoints_data <- filter(nopoints_data, .cooksd < 0.125)

nopoints_data <- select(nopoints_data, 1:15)

nopoints_model <- glm(win ~ FGA + TPA + ORB + DRB + AST + STL + BLK + TOV + PF, nopoints_data, family=bi

summary(nopoints_model)
```

```
##
## Call:
## glm(formula = win ~ FGA + TPA + ORB + DRB + AST + STL + BLK +
##     TOV + PF, family = binomial, data = nopoints_data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0174  -0.6216  -0.1373   0.5350   2.0782
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  5.01376    7.26215   0.690  0.48995
## FGA         -0.45048    0.13697  -3.289  0.00101 **
## TPA          0.01265    0.05208   0.243  0.80804
## ORB          0.55935    0.21621   2.587  0.00968 **
## DRB          0.59392    0.19011   3.124  0.00178 **
## AST          0.39118    0.14241   2.747  0.00602 **
## STL          0.86801    0.32187   2.697  0.00700 **
## BLK          0.23799    0.30374   0.784  0.43332
## TOV         -0.50357    0.23004  -2.189  0.02859 *
## PF          -0.09926    0.14458  -0.687  0.49239
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 95.640  on 68  degrees of freedom
## Residual deviance: 54.294  on 59  degrees of freedom
## AIC: 74.294
##
## Number of Fisher Scoring iterations: 6
```

```
nopoints_data <- augment(nopoints_model, nopoints_data)
```

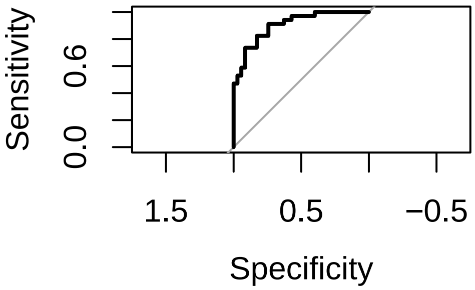
```
vif(nopoints_model)
```

```
##      FGA      TPA      ORB      DRB      AST      STL      BLK      TOV
## 6.867910 1.792545 2.751608 1.936756 3.918850 1.513993 1.663684 1.973618
##       PF
## 1.165934
```

```
nopoints_roc <- roc(nopoints_data, win, .fitted, plot=TRUE)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```



```
threshold <- coords(nopoints_roc, "best", ret = "threshold")
print(threshold)
```

```
##    threshold
## 1 -0.9611865
```

```
nopoints_data <- mutate(nopoints_data, pred = ifelse(.fitted > -0.3855574, 1, 0))

nopoints_data$pred <- as.factor(nopoints_data$pred)

confusionMatrix(nopoints_data$pred, nopoints_data$win)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 26  6
##          1  9 28
##
##                Accuracy : 0.7826
##                  95% CI : (0.6669, 0.8729)
##     No Information Rate : 0.5072
##     P-Value [Acc > NIR] : 2.306e-06
##
##                   Kappa : 0.5657
##
##  Mcnemar's Test P-Value : 0.6056
##
##             Sensitivity : 0.7429
##             Specificity : 0.8235
##          Pos Pred Value : 0.8125
##          Neg Pred Value : 0.7568
##              Prevalence : 0.5072
##          Detection Rate : 0.3768
##    Detection Prevalence : 0.4638
##       Balanced Accuracy : 0.7832
##
##        'Positive' Class : 0
##
```

The accuracy of this model is 87.84%. This model provided the best accuracy and passed all the assumptions. We decided that this model will be our final model and we can make conclusions based on this model.

```
final_model <- attempts_model
```

Conclusion

```
summary(final_model)
```

```
##
## Call:
## glm(formula = win ~ PTS + FGA + TPA + ORB + DRB + AST + STL +
##     BLK + TOV + PF, family = binomial, data = attempts_data)
##
## Deviance Residuals:
```

```
##      Min        1Q    Median        3Q       Max
## -2.07293  -0.50037  -0.08972   0.44787   1.87676
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  2.529472   7.986361   0.317 0.751453
## PTS          0.205520   0.093663   2.194 0.028217 *
## FGA         -0.653575   0.180862  -3.614 0.000302 ***
## TPA          0.008557   0.060514   0.141 0.887545
## ORB          0.736764   0.252719   2.915 0.003553 **
## DRB          0.533894   0.193973   2.752 0.005916 **
## AST          0.214568   0.164647   1.303 0.192505
## STL          1.051963   0.374068   2.812 0.004920 **
## BLK          0.362170   0.348814   1.038 0.299136
## TOV         -0.430452   0.228282  -1.886 0.059347 .
## PF          -0.125846   0.160135  -0.786 0.431940
## ---
## Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 95.640  on 68  degrees of freedom
## Residual deviance: 48.045  on 58  degrees of freedom
## AIC: 70.045
##
## Number of Fisher Scoring iterations: 6
```

By using this model, we can predict whether or not an NBA team won a finals series given their statistics from the series with an accuracy of 87.84%, making the model useful (better than guessing win or lose for each year).

Observations

We found it interesting that the attempts of field goals and three pointers provided the most accurate model for predicting who won the series. It intuitively makes sense that the field goals scored matter more than attempts because basketball games are decided by the score, not the attempts; however, attempts might signal how ineffective a team's offense really can be.

It is worth noting that, while the coefficients for most variables make sense (ie. turnovers having a negative coefficient and points having a positive coefficient), others have a surprising effect on the model.

For example, the coefficients for field goals attempted (`FGA`) is negative, implying that more field goal attempts indicate a lesser likelihood that a team won the game. This can be attributed to the fact that basketball is about making baskets rather than taking shots. If you aren't making these shots you are taking, you most likely won't win the game.

Some variables with high positive coefficients include: Points (`ORB`), Offensive Rebounds (`ORB`), Defensive Rebounds (`DRB`), and Steals (`STL`). These all seem to be good at indicating whether or not teams won the series, which is fascinating because they cover different aspects of the game. A team cannot solely rely on shooting, defense, or size (in the case of offensive and defensive rebounds) to win a championship; they must have all aspects of the game.

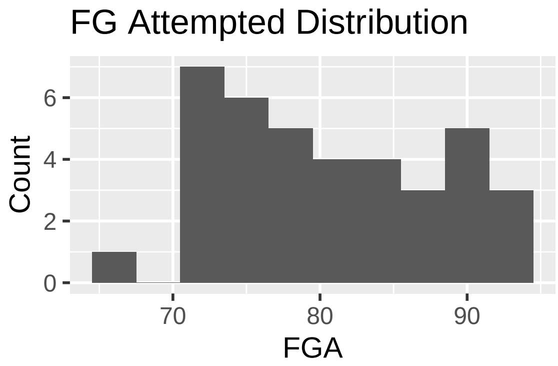
The two variables with low positive coefficients include: Field Goals Attempted (`FGA`) and Turnovers (`TOV`). These also seem good in indicating whether or not teams won the series. Having low turnovers is good as it will give you more opportunities to shoot the ball and prevent free baskets. The other variables don't have that big of an impact on the final result, but that doesn't mean they don't matter at all.

Appendix

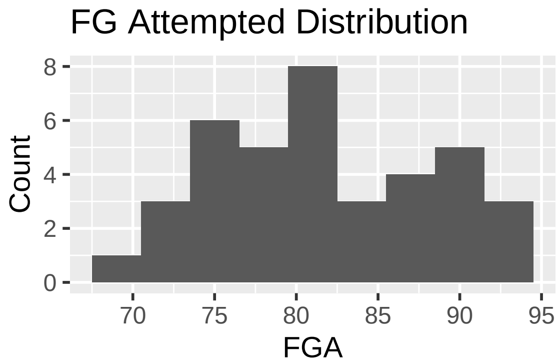
Distributions of variables for Winners and Losers

Distribution of Field Goals Attemped:

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



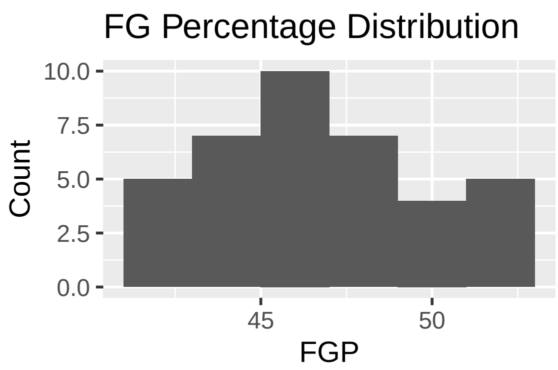
Summary of Field Goal Attempts:

```
summary(useful_data$FGA)
```

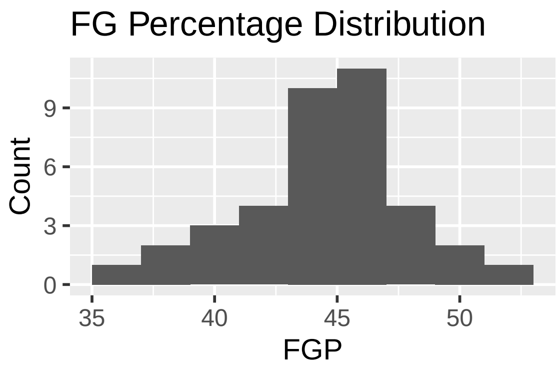
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   67.40   75.42   80.60   81.18   87.00   92.86
```

Distribution of Field Goal Percentage:

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



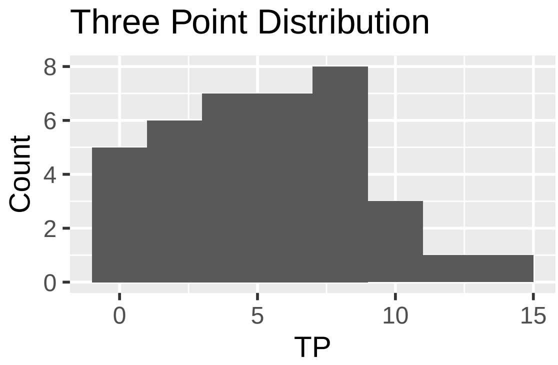
Summary of Field Goal Percentage:

```
summary(useful_data$FGP)
```

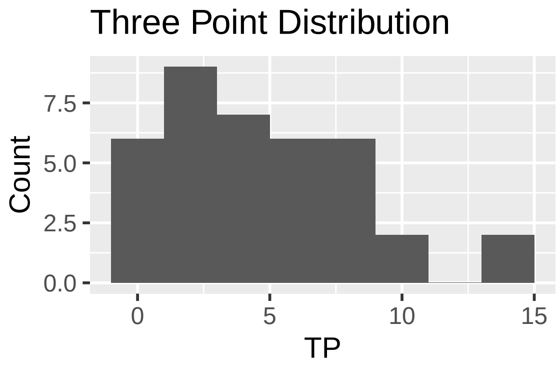
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   36.99   43.20   45.40   45.52   47.51   52.76
```

Distribution of Three Point Distribution:

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



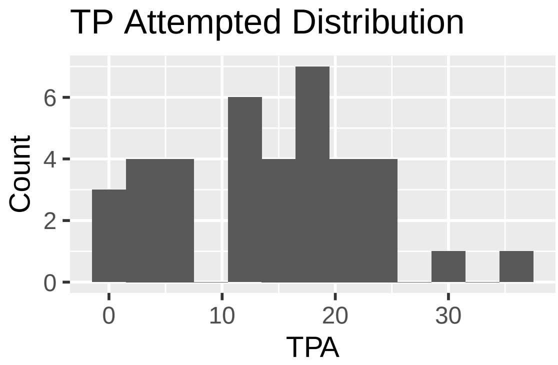
Summary of Three Point:

```
summary(useful_data$TP)
```

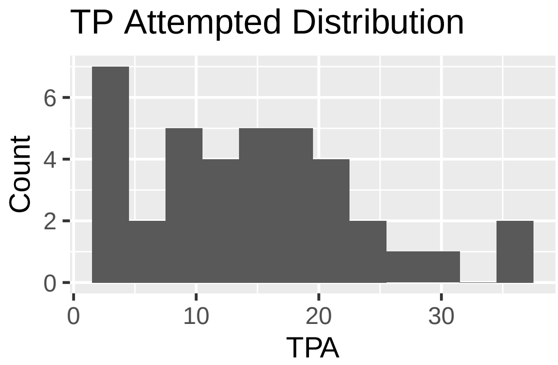
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   0.000   2.167   4.857   4.987   7.259  14.200
```

Distribution of Three Point Attempted:

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



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



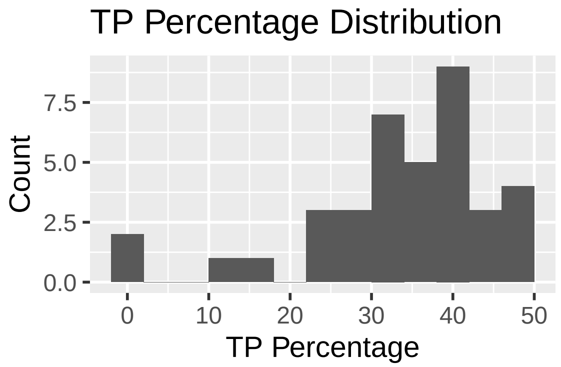
Summary of Three Point Attempts:

```
summary(useful_data$TPA)
```

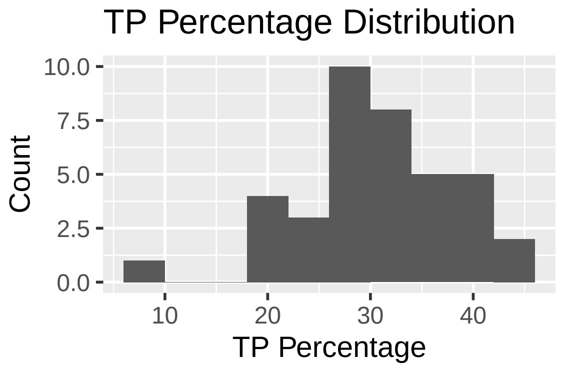
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##  0.6667  6.9583 14.7083 14.3523 20.2500 37.2000
```

Distribution of Three Point Percentage:

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



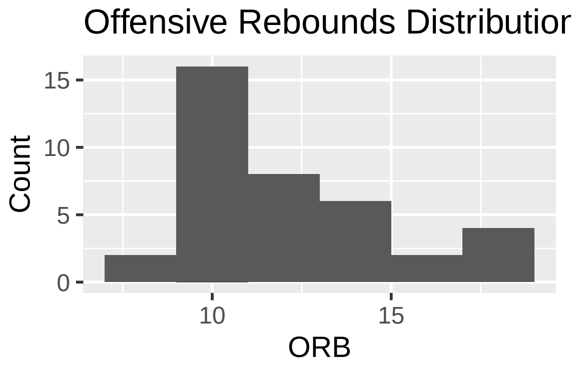
Summary of Three Point Percentage:

```
summary(useful_data$TPP)
```

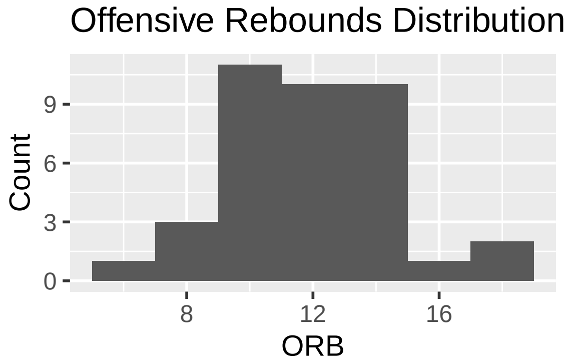
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00   28.20   32.47   31.89   38.29   48.00
```

Distribution of Offensive Rebounds:

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



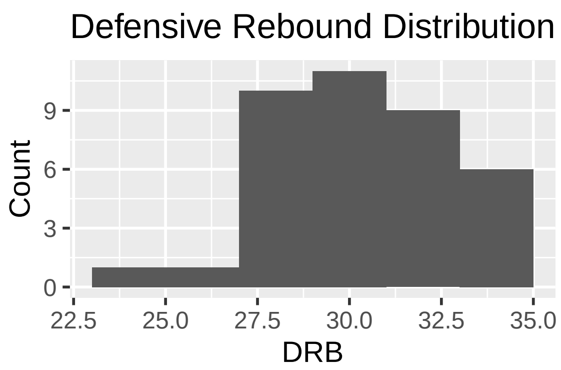
Summary of Offensive Rebounds:

```
summary(useful_data$ORB)
```

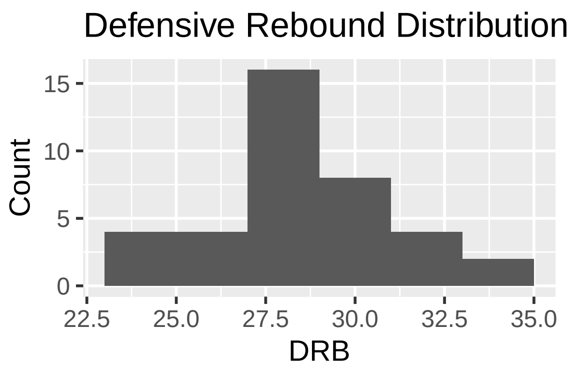
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    5.60   10.37   11.73   12.16   13.71   18.67
```

Distribution of Defensive Rebounds:

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



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



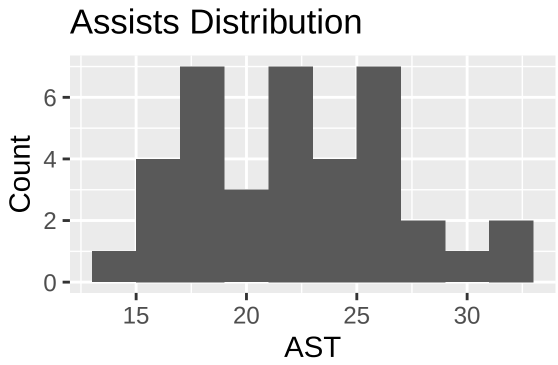
Summary of Defensive Rebounds:

```r
summary(useful_data$DRB)
```

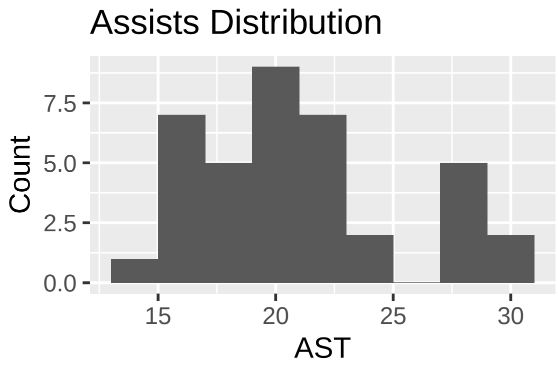
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   23.80   27.83   29.31   29.45   31.30   35.00
```

Distribution of Assists:

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



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



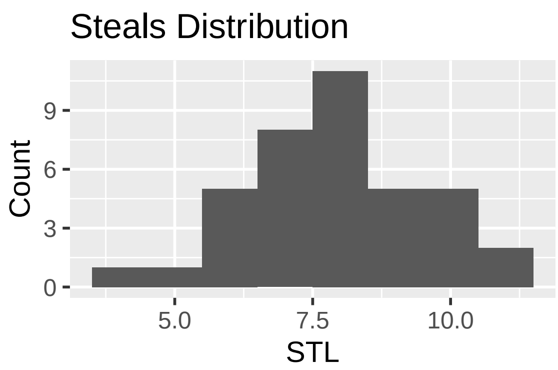
Summary of Assists:

```
summary(useful_data$AST)
```

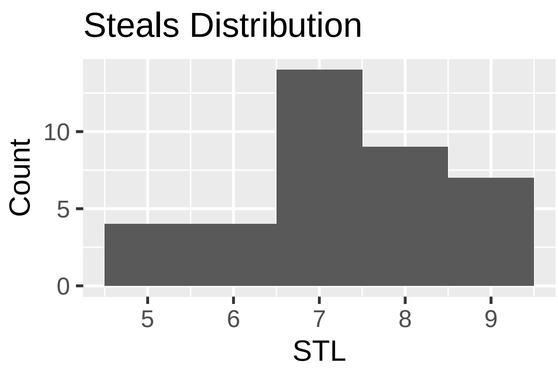
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   14.00   18.33   21.33   21.81   24.73   32.00
```

Distribution of Steals:

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



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



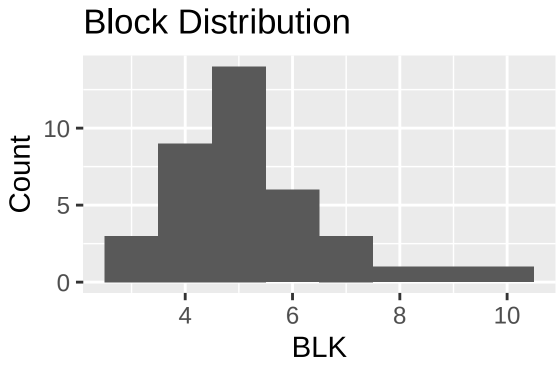
Summary of Steals:

```
summary(useful_data$STL)
```

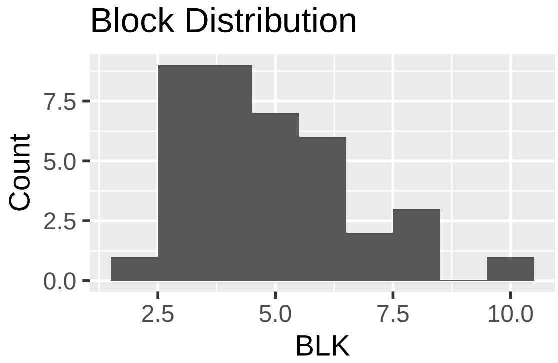
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   4.000   6.833   7.633   7.618   8.488  11.000
```

Distribution of Blocks:

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



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



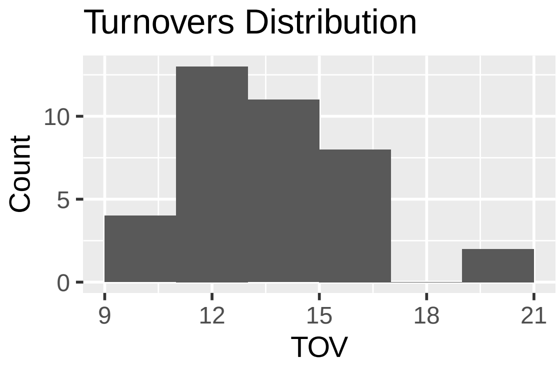
Summary of Blocks:

```
summary(useful_data$BLK)
```

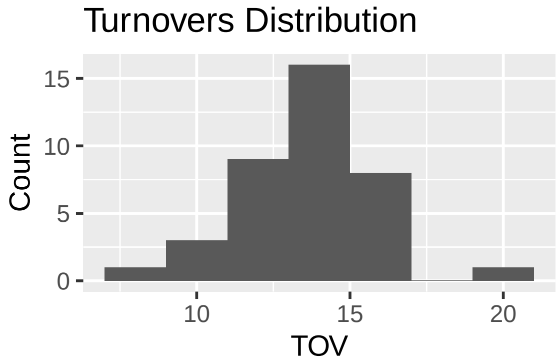
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   2.200   4.000   5.000   5.100   5.808  10.000
```

Distribution of Turnovers:

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



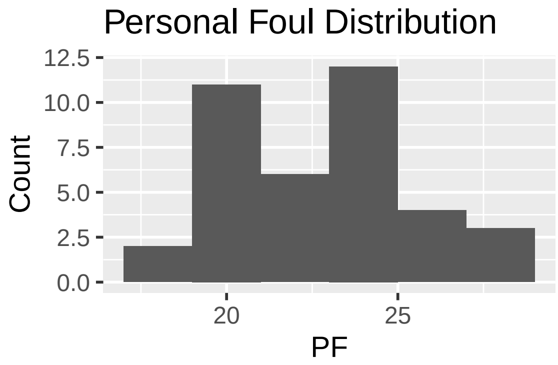
Summary of Turnovers:

```r
summary(useful_data$TOV)
```

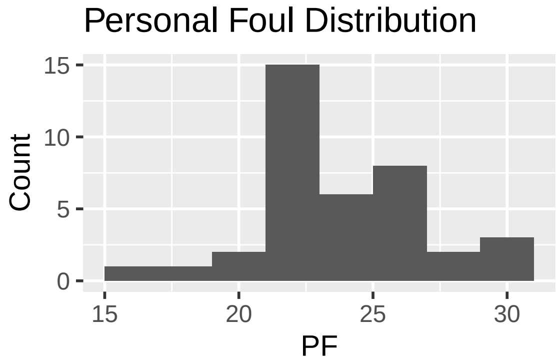
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   8.714  12.421  13.310  13.728  15.042  20.000
```

Distribution of Personal Fouls:

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



Summary of Personal Fouls:

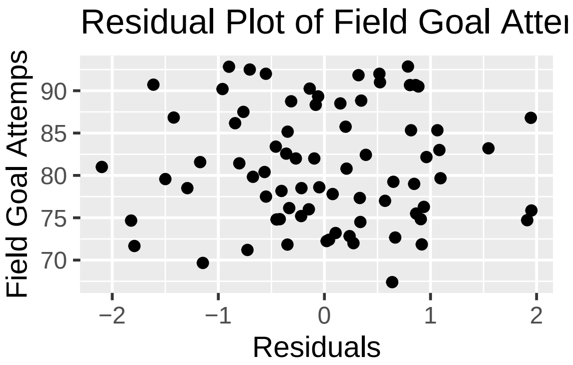
```
summary(useful_data$PF)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   16.86   21.32   23.08   23.39   25.38   30.00
```

Residuals of Each Variable

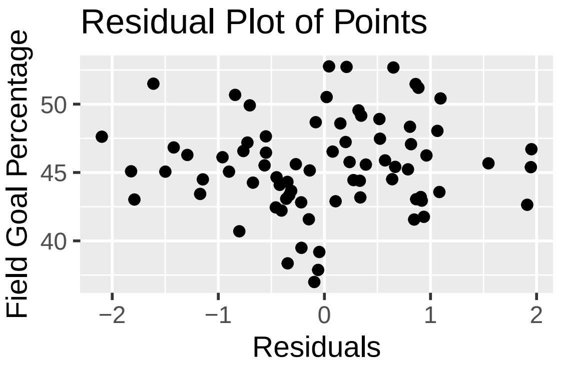
Residual Plot of Field Goal Attempts:

```
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")
```



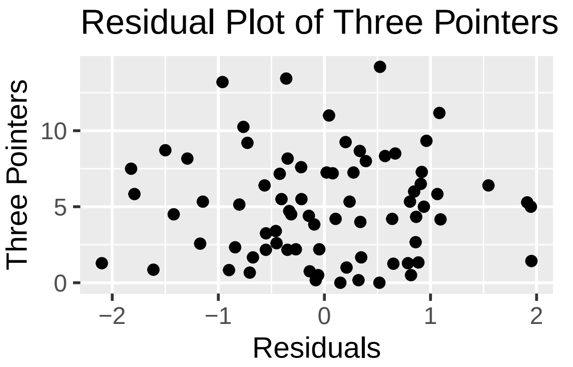
Residual Plot of Points:

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



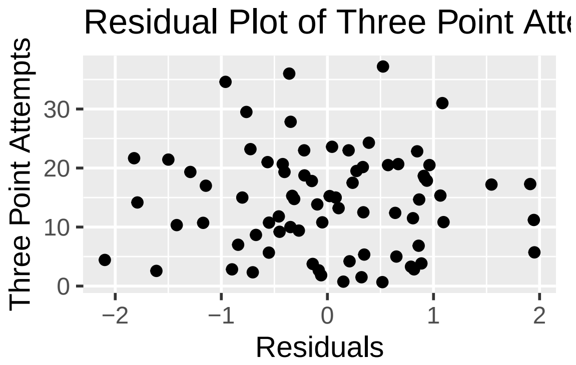
Residual Plot of Three Pointers:

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



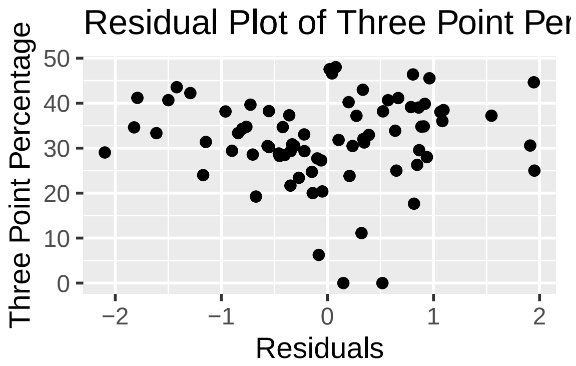
Residual Plot of Three Point Attempts:

```
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")
```



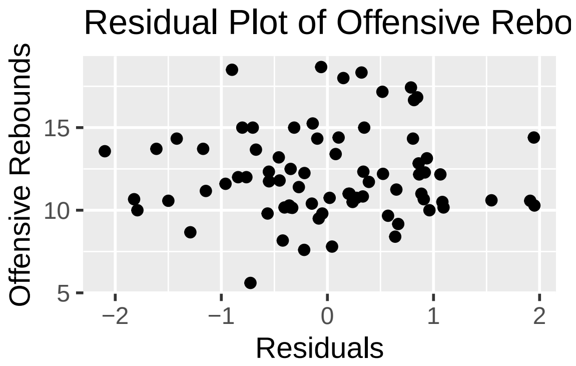
Residual Plot of Three Point Percentage:

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



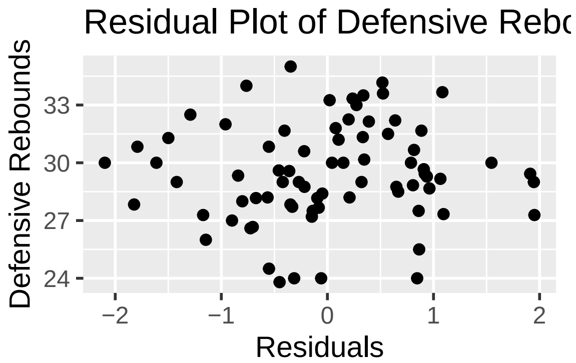
Residual Plot of Offensive Rebounds:

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



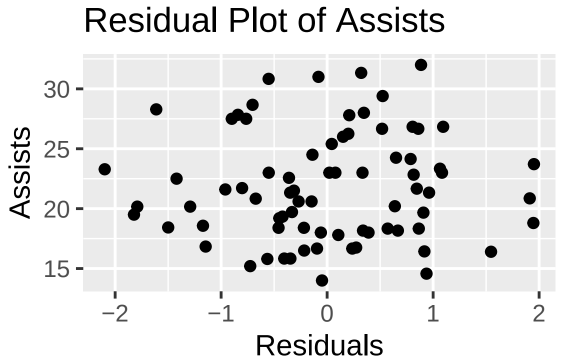
Residual Plot of Defensive Rebounds:

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



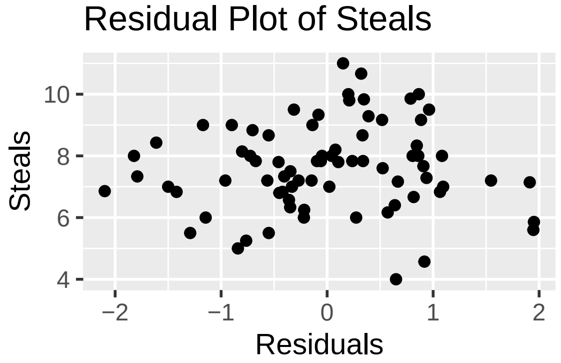
Residual Plot of Assists:

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



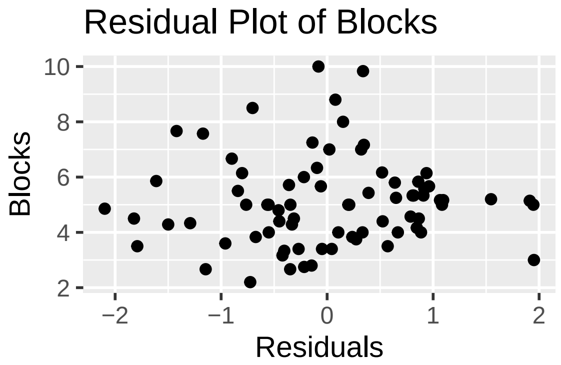
Residual Plot of Steals:

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



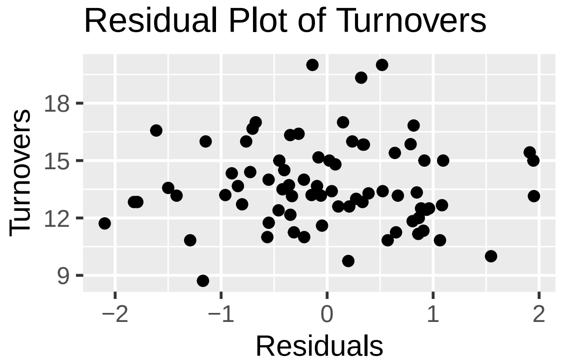
Residual Plot of Blocks:

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



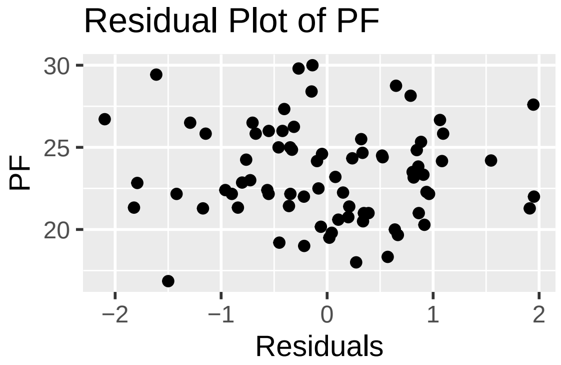
Residual Plot of Turnovers:

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



Residual Plot of PF:

```r
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
PF
Residuals