Final Project

Names

- Vishnu Yarlagadda (9078073161)
- Nikolas Brendel (9080659643)
- Andrew Olson (9079183050)
- Jake Goecke (9079018827)

Abstract

The data we have selected for our group project is player statistics from the National Basketball League(NBA). This data includes everything from game statistics like points per game, all the way to categorical variables, like team or position. For our model we chose mostly game statistics, with the notable exception of age. With this data we hoped to discover if we could use the player statistics and salaries from one season to predict salaries for another season. In order to do this we used multiple regression on statistics from the 2019-2020 season, and then predicted on the 2020-2021 season. Despite initially finding a very poor R-squared value, we pressed on and corrected for heteroscedasticity. When then using a development set and a test set, we showed our model to be about 95% accurate in predicting player salaries.

Introduction

The NBA is one of the most popular sports leagues in the world. Not just the US, but many other countries follow the NBA closely. This attention leads to a lot of income for the NBA. A good portion of that income goes into paying the players that are a part of the NBA and its teams.

This goal of this project is to attempt to predict NBA player's salary based on their individual statistics. We will be using data from the 2019-2020 season to attempt to accurately predict the salaries from the 2020-2021 season.

This research is important because if it succeeds, it may be a tool that helps teams determine how much to pay certain players in the future. That could enable teams to put together the best possible team and get the most out of their limited money each season.

Data Resources

All of the data in our datasets originate from the NBA organization. The NBA is a professional basketball league in the USA that features the highest level of basketball in the world. The stats that are tracked are all of the numbers the players put up within the multiple regular season games. Then the salaries that are tracked are what players are being payed at the start of each season.

Link: https://www.basketball-reference.com/leagues/NBA_2021_per_game.html

This dataset is from basketball reference, which is a website that holds stats for a lot of different categories about the NBA. This particular one has the per game statistics of the NBA players from the 2020-2021 season. We used basketball reference because it allows us to get the information in usable form.

Link: https://www.basketball-reference.com/leagues/NBA_2020_per_game.html

This dataset is also from basketball reference, but instead of the 2020-2021 season it contains the per game stats from players in the 2019-2020 season.

Link: https://hoopshype.com/salaries/players/2020-2021/

This dataset contains all of the salaries of players from the 2020-2021 season. This dataset comes from the website hoopshype which just gives us this information from the NBA is a nice usuable table format.

Link: https://www.kaggle.com/junfenglim/nba-player-salaries-201920

This dataset contains all of the salaries of players from the 2019-220 season. This dataset comes from kaggle which just gives us a usuable format to use the salary data from the NBA.

Variables

Variables we used:

- Player - The player's name that they used when they played for the given year of the dataset
- salary - The amount in USD that the player will be paid for the season
- Age - The player's age in years at the start of the season
- G - The number of games played in during the season. This includes any game where the player was on the court for any amount of time
- GS - The number of games the player started in. This means that they were playing right when the game started.
- BLK - The average blocks per game. Takes the total amount of blocks divided by games played
- TOV - The average turnovers per game. Takes the total amount of assists divided by games played
- PF - The average of fouls committed per game. Takes the total amount of fouls the player committed divided by games played
- PTS - The average total points per game. Takes the total amount of points the player scored divided by games played

Variables in our dataset that we didnt use:

- team - The team that the player played for
- position - The position that the player plays
- season - The year that the data was recorded
- ï..Rk - The rank of the player
- Pos - same as position
- MP - Minutes played per game
- FG - Field goals made per game
- FGA - Field goals attempted per game
- FG. - Field goal percentage for the season
- X3p - 3 pointers made per game
- X3PA - 3 pointers attempted per game
- X3P. - 3 point make percentage for the season
- X2P - Amount of two pointers made per game
- X2PA - Amount of two pointers attempted
- X2P. - Two point make percentage
- eFG. - Field goal effiecency percentage
- FT - Free throws attempted per game
- FTA - Free throws attempted per game
- FT. - Free throw percentage
- ORB - Offensive rebounds per game
- DRB - Defensive rebounds per game
- TRB - Total rebounds per game
- AST - Assists per game
- STL - Steals per game

Data Calculations

The following block reads in the data sets containing player statistics and salaries for the 2019-2020 and 2020-2021 seasons. It also changes a few variable names to ensure they match for the join.

```
player_stats_2019_20 <- read.csv("2019-20_player_stats.csv")
player_salaries_2019_20 <- read.csv("2019-20_player_salaries.csv")
colnames(player_salaries_2019_20)[3] <- "Player"

player_stats_2020_21 <- read.csv("2020-21_player_stats.csv")
player_salaries_2020_21 <- read.csv("2020-21_player_salaries.csv")
colnames(player_salaries_2020_21)[3] <- "salary"
```

The following code joins the player statistics and player salary data sets into one data frame by different years. It also removes the cases where players were traded in the middle of the season to prevent any duplicate salary information from being included. Then it shows the first few rows of the data.

```
merged_2019_20 <- merge(player_salaries_2019_20, player_stats_2019_20, by = 'Player')
merged_2019_20 <- merged_2019_20 %>%
  distinct(Player, .keep_all = TRUE)
head(merged_2019_20)
```

```
##              Player                  team   salary position    season ï..Rk Pos
## 1      Aaron Gordon         Orlando Magic 19863636       PF 2019-2020   179  PF
## 2     Aaron Holiday        Indiana Pacers  2329200       PG 2019-2020   223  PG
## 3       Abdel Nader Oklahoma City Thunder  1618520       SF 2019-2020   362  SF
## 4 Admiral Schofield    Washington Wizards  1000000       SG 2019-2020   441  PF
## 5   Al-Farouq Aminu         Orlando Magic  9258000       PF 2019-2020     9  PF
## 6        Al Horford    Philadelphia 76ers 28000000       PF 2019-2020   229   C
##   Age  Tm  G GS   MP  FG  FGA   FG. X3P X3PA  X3P. X2P X2PA  X2P.  eFG.  FT FTA
## 1  24 ORL 62 62 32.5 5.4 12.4 0.437 1.2  3.8 0.308 4.2  8.5 0.494 0.484 2.4 3.6
## 2  23 IND 66 33 24.5 3.5  8.5 0.414 1.3  3.3 0.394 2.2  5.2 0.427 0.491 1.1 1.3
## 3  26 OKC 55  6 15.8 2.2  4.8 0.468 0.9  2.3 0.375 1.4  2.5 0.556 0.559 0.9 1.2
## 4  22 WAS 33  2 11.2 1.1  2.8 0.380 0.6  1.8 0.311 0.5  0.9 0.516 0.484 0.3 0.5
## 5  29 ORL 18  2 21.1 1.4  4.8 0.291 0.5  2.0 0.250 0.9  2.8 0.320 0.343 1.1 1.6
## 6  33 PHI 67 61 30.2 4.8 10.6 0.450 1.5  4.2 0.350 3.3  6.4 0.516 0.520 0.9 1.2
##     FT. ORB DRB TRB AST STL BLK TOV  PF  PTS
## 1 0.674 1.7 5.9 7.7 3.7 0.8 0.6 1.6 2.0 14.4
## 2 0.851 0.3 2.0 2.4 3.4 0.8 0.2 1.3 1.8  9.5
## 3 0.773 0.3 1.6 1.8 0.7 0.4 0.4 0.8 1.4  6.3
## 4 0.667 0.2 1.2 1.4 0.5 0.2 0.1 0.2 1.5  3.0
## 5 0.655 1.3 3.5 4.8 1.2 1.0 0.4 0.9 1.5  4.3
## 6 0.763 1.5 5.3 6.8 4.0 0.8 0.9 1.2 2.1 11.9
```

```
merged_2020_21 <- merge(player_salaries_2020_21, player_stats_2020_21, by = 'Player')
merged_2020_21 <- merged_2020_21 %>%
  distinct(Player, .keep_all = TRUE)
head(merged_2020_21)
```

```
##            Player ï..Column1   salary ï..Rk Pos Age  Tm  G GS   MP  FG  FGA
## 1    Aaron Gordon          60 18136364   182  PF  25 ORL 25 25 29.4 5.1 11.7
## 2   Aaron Holiday         319  2345640   224  PG  24 IND 66  8 17.8 2.6  6.6
## 3   Aaron Nesmith         269  3458400   364  SF  21 BOS 46  1 14.5 1.7  3.9
## 4     Abdel Nader         379  1752950   362  SF  27 PHO 24  0 14.8 2.4  4.8
## 5     Adam Mokoka         511   449115   349  SG  22 CHI 14  0  4.0 0.5  1.4
## 6 Al-Farouq Aminu         126  9720900    10  PF  30 TOT 23 14 18.9 1.7  4.3
##     FG. X3P X3PA  X3P. X2P X2PA  X2P.  eFG.  FT FTA   FT. ORB DRB TRB AST STL
```

```
## 1 0.437 1.7  4.5 0.375 3.4  7.2 0.475 0.509 2.6 4.2 0.629 1.6 5.1 6.6 4.2 0.6
## 2 0.390 1.0  2.8 0.368 1.6  3.8 0.406 0.467 1.0 1.3 0.819 0.2 1.1 1.3 1.9 0.7
## 3 0.438 0.9  2.3 0.370 0.8  1.5 0.543 0.551 0.5 0.6 0.786 0.6 2.2 2.8 0.5 0.3
## 4 0.491 0.8  1.8 0.419 1.6  3.0 0.534 0.569 1.2 1.5 0.757 0.3 2.3 2.6 0.8 0.4
## 5 0.368 0.1  0.7 0.100 0.4  0.6 0.667 0.395 0.0 0.1 0.000 0.1 0.3 0.4 0.4 0.1
## 6 0.384 0.3  1.6 0.216 1.3  2.7 0.484 0.424 0.8 1.0 0.818 1.0 3.8 4.8 1.3 0.8
##   BLK TOV  PF  PTS
## 1 0.8 2.7 2.0 14.6
## 2 0.2 1.0 1.4  7.2
## 3 0.2 0.5 1.9  4.7
## 4 0.4 0.8 1.4  6.7
## 5 0.1 0.4 0.4  1.1
## 6 0.4 1.2 1.3  4.4
```

The following code divides our tables into 3 different sets. We will train our multiple-regression model on the 2019-2020 season. The train data frame will additionally be divided into a dev data frame to compare with our test data frame to make sure our multiple-regression model remains consistent over different data frames. We also end up using the 2020-2021 season as our test data frame.

```
data_split <- sort(sample(nrow(merged_2019_20), nrow(merged_2019_20)*0.8))
train <- merged_2019_20[data_split,]
dev <- merged_2019_20[-data_split,]
test <- merged_2020_21
```

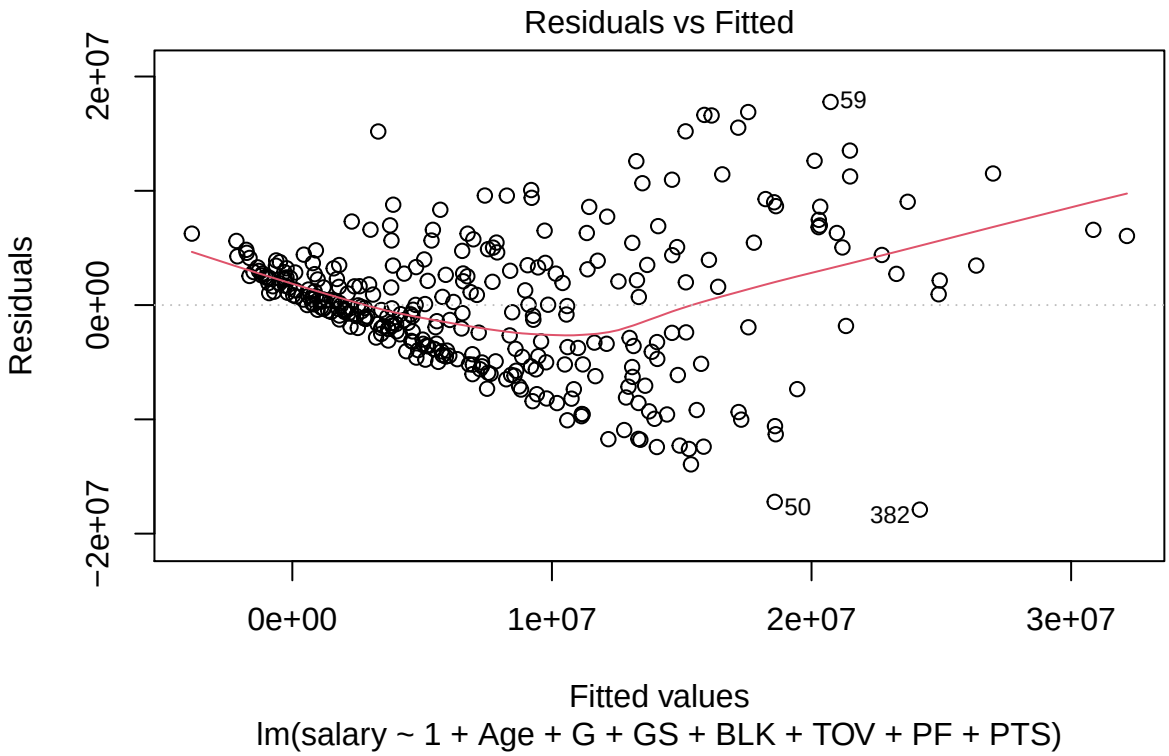
Here we create our multiple-regression model.

```
model <- lm(salary ~ 1 + Age + G + GS + BLK + TOV + PF + PTS, train)
```

Based on the variables provided on the players, without using their names, we are able to achieve an R-squared value of **0.5622**. The most important variables are "Age", "PTS", and "G". This makes sense as younger players lack experience that would make them more valuable to the team and older players are more likely to have a harder time keeping up with the speed of the game. As for points, it is pretty self explanatory as players who score more points are more valuable to their team. Additionally, games played is important because a team does not want to pay a player that doesn't play.

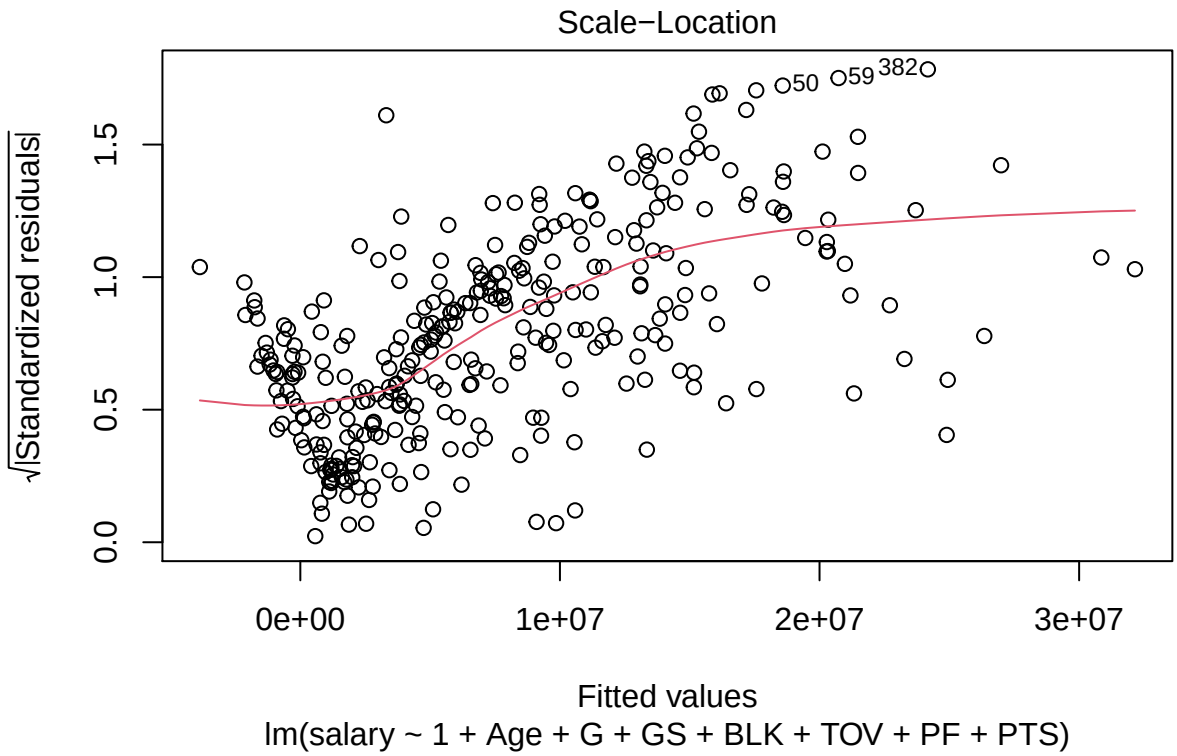
As the primary goal of this model is to predict the salary variable, we can use the R-squared statistic to analyze how good the model. It is clear that a R-squared of **0.5622** is not very good. In addition, if we look at the model diagnostics, it becomes pretty clear that our current model is not very good. The first one we will look at is the Residual vs Fitted plot.

```
plot(model, which = 1)
```



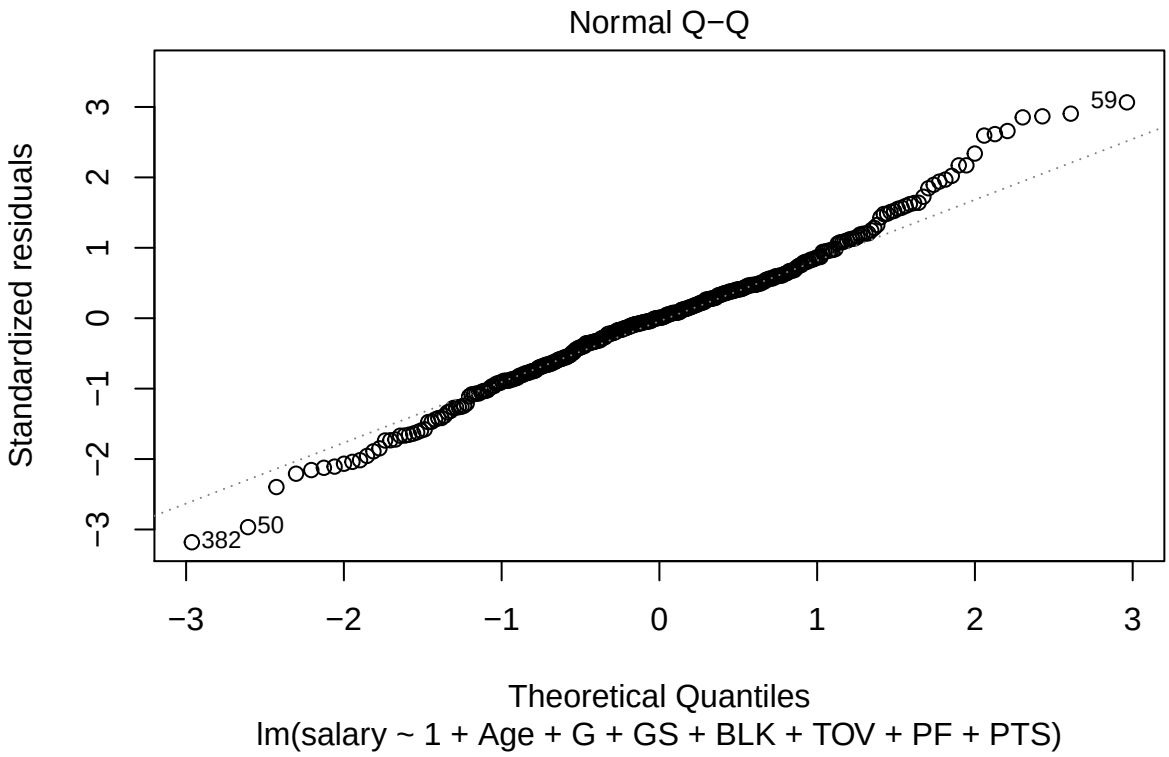
You can see that there is a "funneling" effect on the graph. The distribution of the residuals is concentrated around 0 for small fitted values, but they get more and more spread out as the fitted values increase. This phenomena is called increasing variance. And, the Scale-Location graph also confirms this below.

```
plot(model, which = 3)
```



There is a clear upward trend in the red line which reaffirms our previous statement that the model has non-constant variance. One more thing that we should look at is the Normal QQ plot which lets us assess whether the residuals are normally distributed.

```
plot(model, which = 2)
```



We observe heavier tails in this graph which tells us that the model has larger values that we would expect under the standard modeling assumptions. But, it looks like normality assumption is still valid.

It is now clear that our model is not very good, so we have to resort to other methods. Our original plan was to use ridge regression to achieve a better model but an assumption of ridge regression is constant variance so we cannot use it. Lasso regression also has the same assumptions so we cannot use that one either. We had to resort to some online research and we found a method to achieve a linear model with non-constant variances. We found out that the technical term for non-constant variance is heteroskedasticity. We found a good resource to solve this problem and I have provided the link below: http://bkenkel.com/psci8357/notes/05-ncv.html

The resource tells us that in order to solve heteroskedasticity of an unknown form we can use something called White's heteroskedasticity-consistent estimator which is what we did below

```
library(car)
```

```
## Warning: package 'car' was built under R version 4.0.5
```

```
## Loading required package: carData
```

```
##
## Attaching package: 'car'
```

```
## The following object is masked from 'package:dplyr':
##
##     recode
```

```
## The following object is masked from 'package:purrr':
##
##     some
```

```
library(lmtest)
```

```
## Warning: package 'lmtest' was built under R version 4.0.5
```

```
## Loading required package: zoo
```

```
## Warning: package 'zoo' was built under R version 4.0.5
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
vcv0 <- hccm(model, type = "hc0")

library("broom")
```

```
## Warning: package 'broom' was built under R version 4.0.5
```

```
modelv2 <- coeftest(model, vcov = vcv0)
modelv2
```

```
##
## t test of coefficients:
##
##              Estimate Std. Error t value  Pr(>|t|)
## (Intercept) -15487362    2381079 -6.5043 2.991e-10 ***
## Age            695654      94785  7.3393 1.775e-12 ***
## G              -61540      16442 -3.7428 0.0002155 ***
## GS              82273      22148  3.7147 0.0002399 ***
## BLK           3179208     984302  3.2299 0.0013665 **
## TOV           2294958    1010500  2.2711 0.0238021 *
## PF           -1717375     635618 -2.7019 0.0072609 **
## PTS            520017     127158  4.0895 5.470e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

This model know has accounted for the heteroskedasticity and it shows very little difference between our initial model. Therefore, we can now perform the predictions on the initial model.

Now we will predict on our dev data frame and test data frame. Once the predictions are made, we will determine the accuracy of the model by seeing how many of the actual values fall within the 95% prediction interval.

```
dev_predictions <- predict(model, newdata = dev, interval = "prediction")
dev_predictions <- as.data.frame(dev_predictions)
mean(dev_predictions$lwr <= dev$salary & dev$salary <= dev_predictions$upr)
```

```
## [1] 0.9277108
```

```
test_predictions <- predict(model, newdata = test, interval = "prediction")
test_predictions <- as.data.frame(test_predictions)
mean(test_predictions$lwr <= test$salary & test$salary <= test_predictions$upr)
```

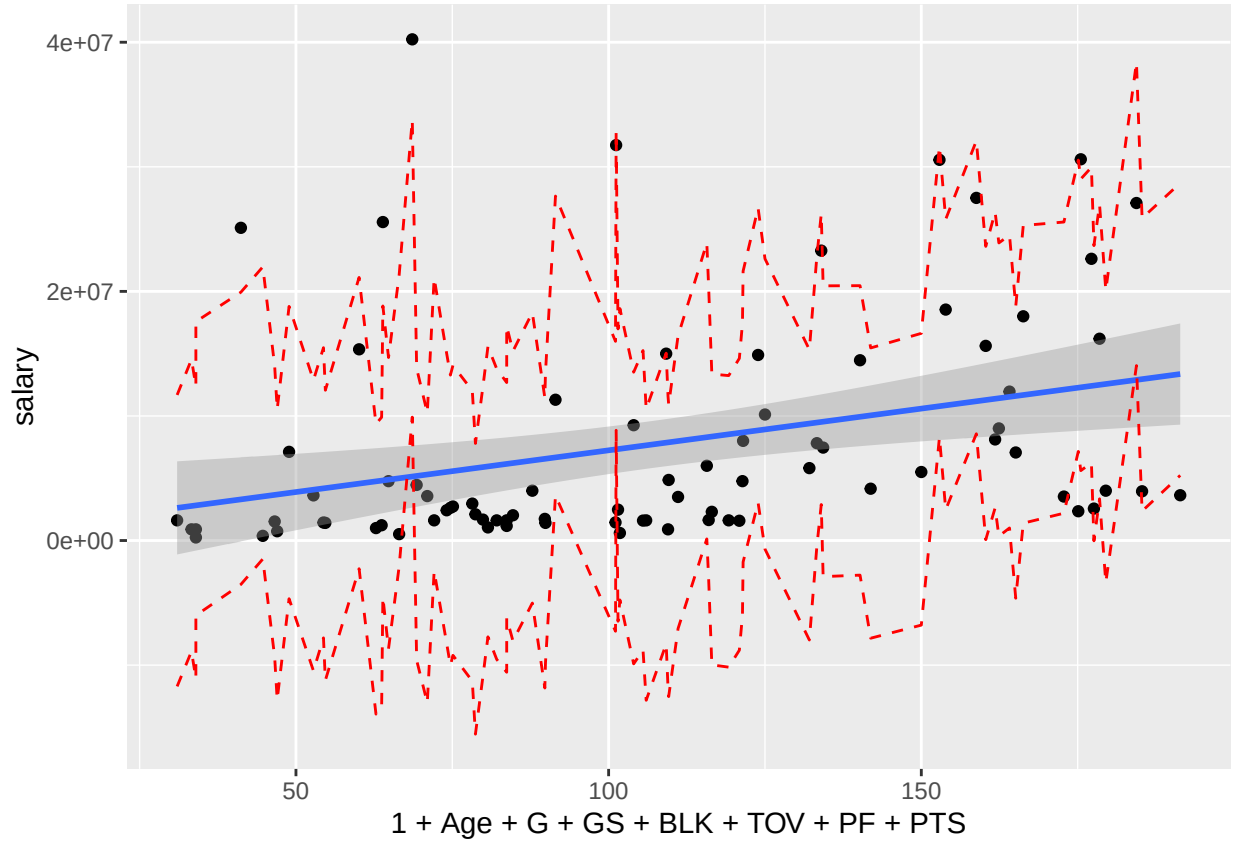
```
## [1] 0.9410526
```

Based on our model and our predictions, we can accurately predict at a rate of approximately 0.94. Compare this to our dev rate of approximately 0.93 and these value match well.

The following code will graph our predictions, actual values, linear model, and the 95% prediction intervals for both our dev data and our test data.

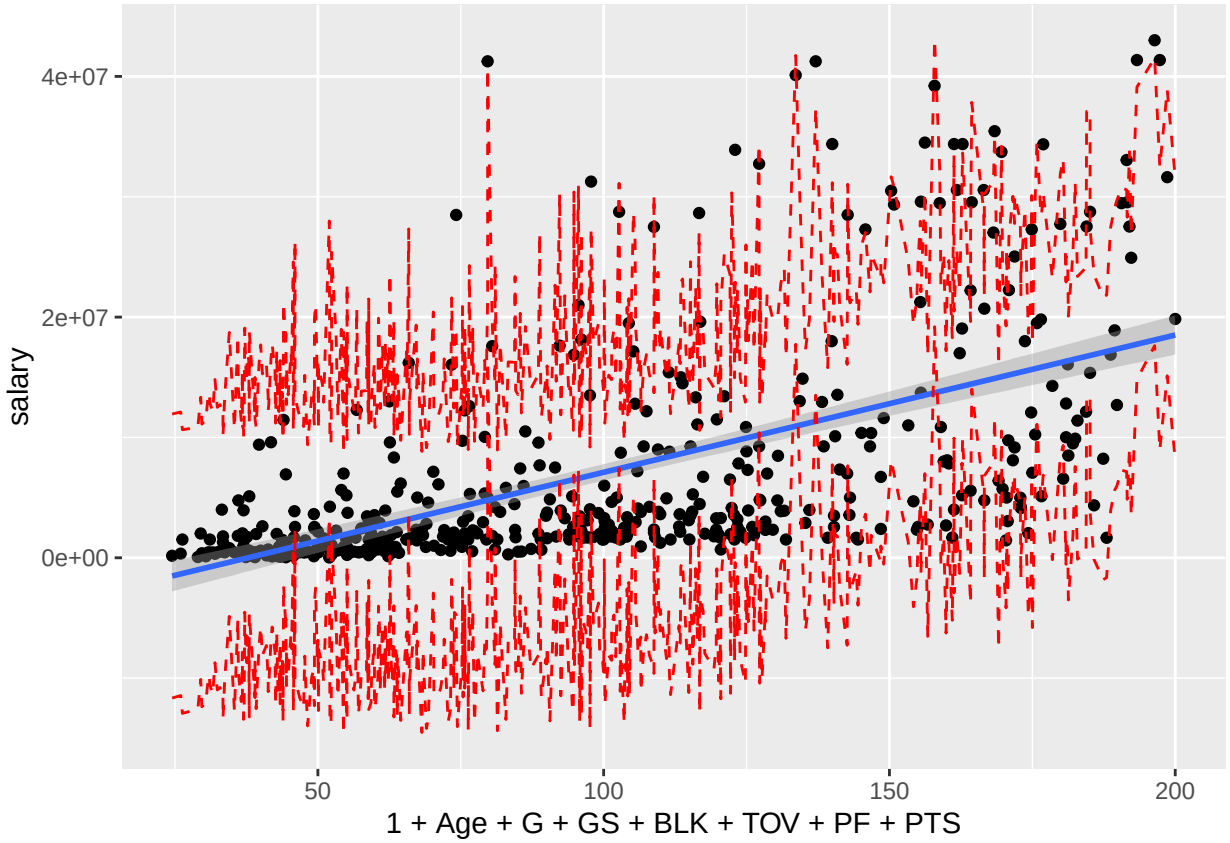
```
library("ggplot2")
devbind <- cbind(dev, dev_predictions)
devplot <- ggplot(devbind, aes(1 + Age + G + GS + BLK + TOV + PF + PTS, salary)) +
  geom_point() +
  stat_smooth(method = lm)
devplot + geom_line(aes(y = lwr), color = "red", linetype = "dashed") +
    geom_line(aes(y = upr), color = "red", linetype = "dashed")
```

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




```
testbind <- cbind(test, test_predictions)
testplot <- ggplot(testbind, aes(1 + Age + G + GS + BLK + TOV + PF + PTS, salary)) +
  geom_point() +
  stat_smooth(method = lm)
testplot + geom_line(aes(y = lwr), color = "red", linetype = "dashed") +
    geom_line(aes(y = upr), color = "red", linetype = "dashed")
```

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



Conclusion

To conclude, we were able to fit a model that predicted player salaries for the 2020-2021 season with about 95% accuracy within the prediction intervals. However, this is likely due to the proximity of the train and test sets, and NBA salaries have been rising for many years now. This increase is due to the increasing popularity of the NBA and since the salary cap is determined by the previous years income for the NBA, the salary cap has been increasing at a large rate for many years. Our model would most likely improve if given information on the salary cap in the NBA the year a player signed their current contract. This would provide more insight as exactly how NBA teams are determining the value of their players compared to the new salary cap. Unfortunately, our model does not test on any other seasons, and would likely perform much worse on a season from even five years ago. With that being said, something we could research in the future is attempting to fit the model across multiple years, and using year as a predictor. Our model was slightly successful in predicting salaries inside prediction intervals but would not be seen as successful in predicting values exactly.