project part 1

Nicholas Mikhail and Pete Boyle

2024-02-09

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

Source of Data and description of all relevant variables:

For this project, we used three different Csv. 1. the first csv is called "Seasons_Stats" and it has over 50 columns of statistics from the last 50 NBA seasons ending in 2017. We definitely are not using all of these columns stats, as the main ones we are focusing on are PER, and a created PPG column which I made by dividing total points by games played, both of which were available in the csv 2. The second csv is called "nba_2016_17_salary" which had the salary for every player in the 2016 season 3. The third csv is called "NBA_season1718_salary" and has the salary for each player in the 2017

We then narrowed down the data to just be the years 2016 and 2017 as those were the years we had data for the players salary as well, and then we merged the salary datasets into the "Seasons_stats" so we could have all the necessary data in one csv file. The relevant variables are, PPG (points scored per game), PER (player efficiency rating), SALARY (annual salary for each player) The observational unit is Player, or basically every nba player's name.

```
Seasons_Stats <- read.csv("~/statistical linear modeling /project/Seasons_Stats.csv")
nba_2016_17_salary <- read.csv("~/statistical linear modeling /project/nba_2016-17_salary.csv")
NBA_season1718_salary <- read.csv("~/statistical linear modeling /project/NBA_season1718_salary.csv")

# Remove the columns 'blank2', 'blanl', and 'DRB%' from the data set
Seasons_Stats <- Seasons_Stats[, !names(Seasons_Stats) %in% c("blank2", "blanl", "DRB%", "TS%", "TRB%")]

# Filter the dataset for years after 2015
Seasons_stats_2016_2017 <- subset(Seasons_Stats, Year > 2015)

# Add a new column 'PPG' by dividing 'PTS' by 'G'
Seasons_stats_2016_2017$PPG <- with(Seasons_stats_2016_2017, PTS / G)


# Rename "NAME" column to "Player" in nba_2016_17_salary
colnames(nba_2016_17_salary)[colnames(nba_2016_17_salary) == "NAME"] <- "Player"

# Merge 'Seasons_stats_after_2010' with the 'SALARY' column from 'nba_2016_17_salary'
merged_data <- merge(Seasons_stats_2016_2017, nba_2016_17_salary[, c("Player", "SALARY")], by = "Player
colnames(NBA_season1718_salary)[colnames(NBA_season1718_salary) == "season17_18"] <- "SALARY"

# Merge with the 2017 salary data, adding suffixes to distinguish between the different salary columns
merged_data_full <- merge(merged_data, NBA_season1718_salary[, c("Player", "SALARY")], by = "Player", a

# Update the SALARY for 2017 entries
merged_data_full$SALARY <- ifelse(merged_data_full$Year == 2017 & !is.na(merged_data_full$SALARY.2017),

# Remove the now unnecessary 'SALARY.2017' column
merged_data_full$SALARY.2017 <- NULL

# Rename merged_data_full
stats_2016_17_with_salary <- merged_data_full

# divide salary by 1,000,000 so it works for graphs
stats_2016_17_with_salary$SALARY <- stats_2016_17_with_salary$SALARY / 1e6
```

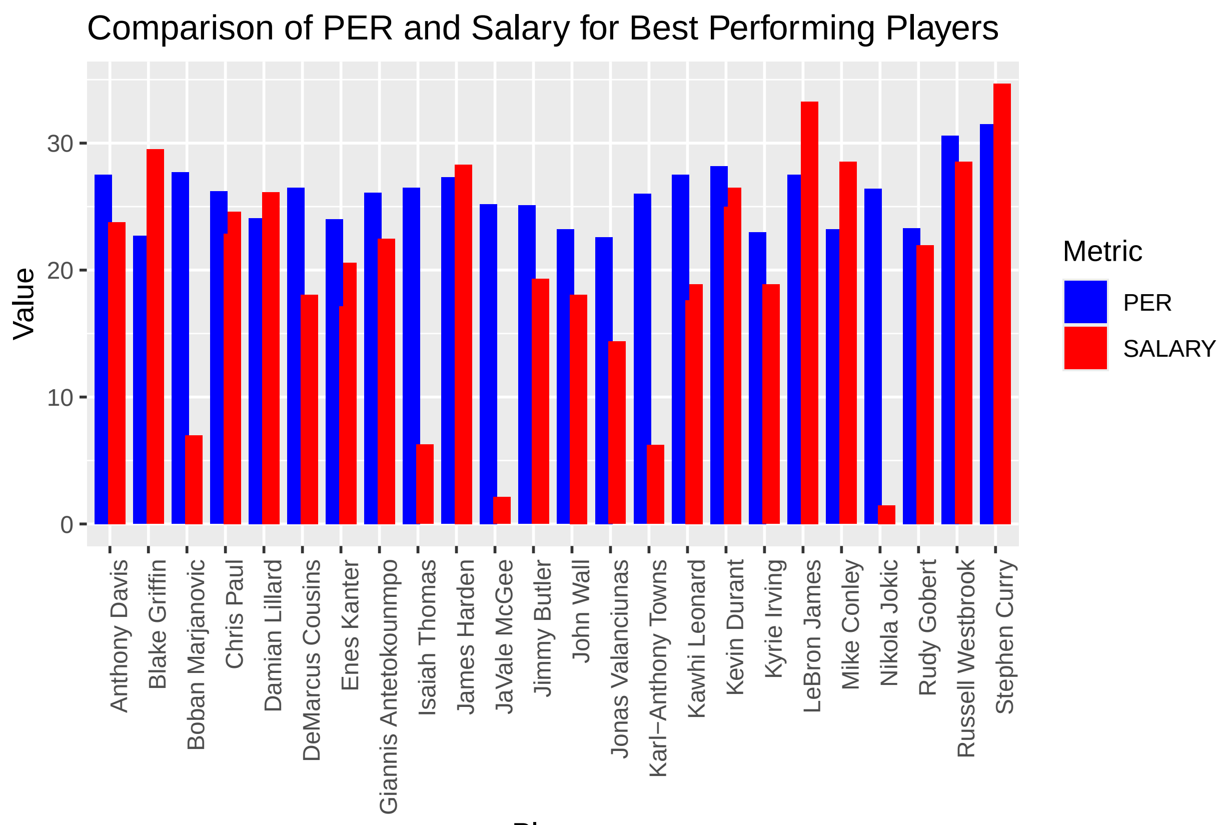
For now we are primarily looking at Salary and PER and PPG, but going forward, Here are some defintions of variables we might look at investigating:

```
variable_table <- data.frame(
  Variable = c("Salary", "MP", "PER", "3par", "OWS", "DWS", "WS/48", "BPM", "VORP", "EFG%", "PPG"),
  Description = c("How much a player is paid for that season", "Minutes played that season", "Player ef
)

print(variable_table)
```

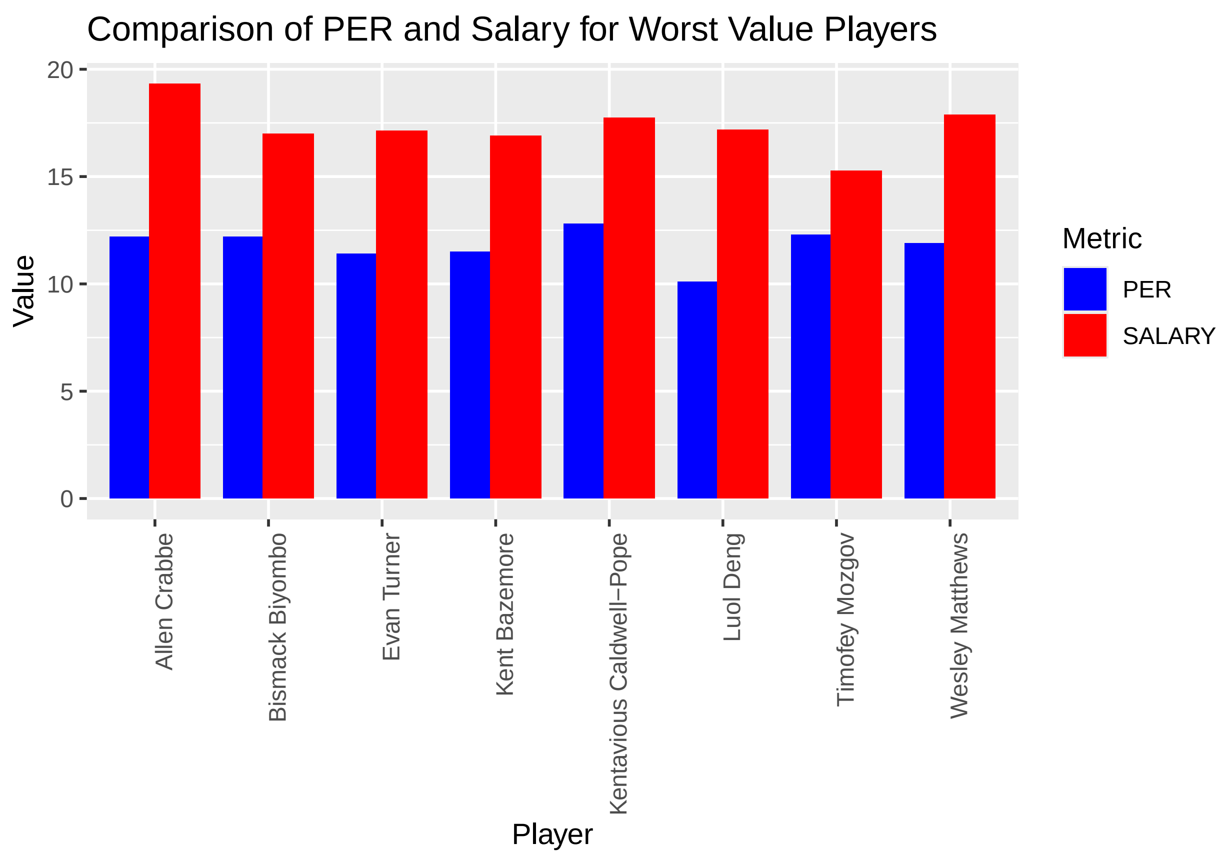
```
##    Variable
## 1    Salary
## 2        MP
## 3       PER
## 4      3par
## 5       OWS
## 6       DWS
## 7     WS/48
## 8       BPM
## 9      VORP
## 10     EFG%
## 11      PPG
##
## 1
## 2
## 3
## 4
## 5
## 6
## 7
## 8  Box plus-minus, which uses a player's box score information, position, and the team's overall per
## 9
## 10
## 11
```

Here are some graphs:

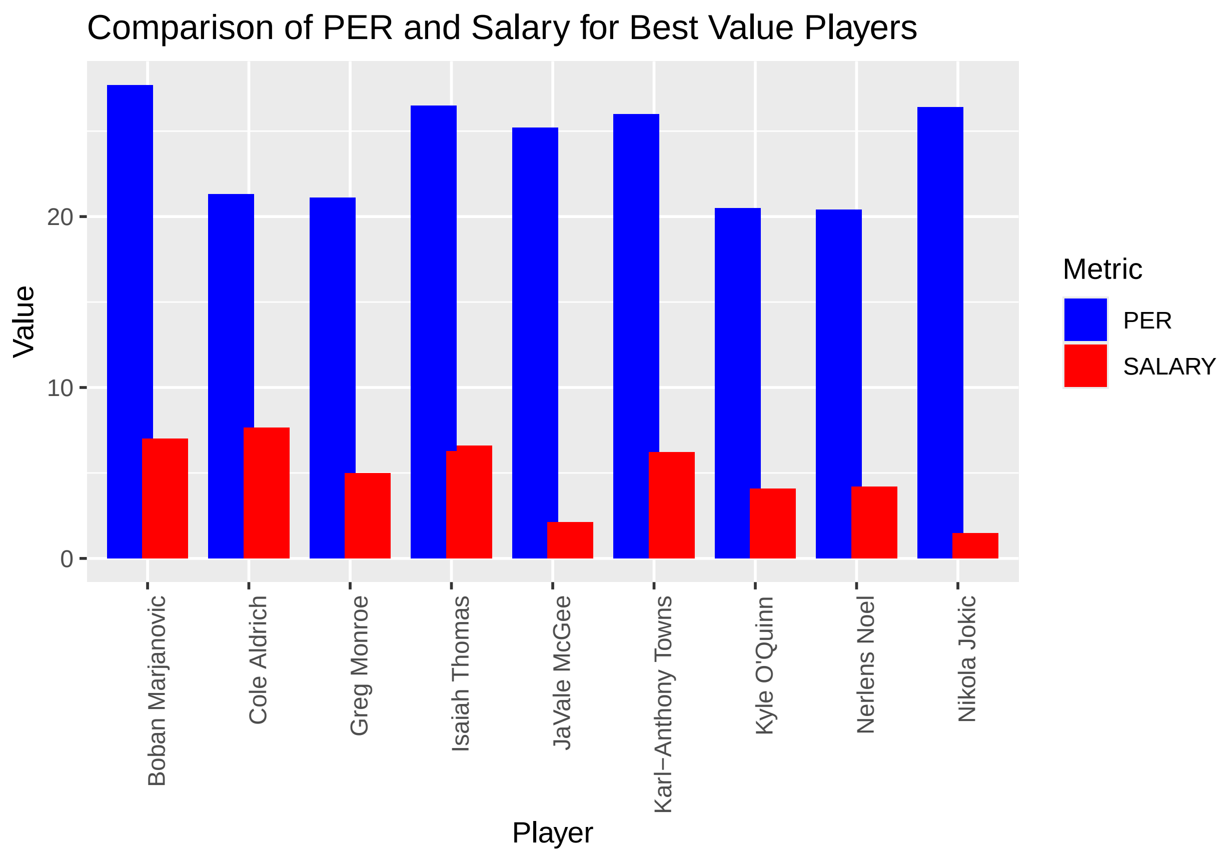
```r
#getting players that played at least 41 games (half the season)
half_season <- subset(stats_2016_17_with_salary, G > 40)
#getting players that played at least 41 games, and have the highest PER
best_per <- subset(half_season, PER > 22.5)
#getting players that played at least 41 games, and also have the worse value PER compared to salary
worse_per <- subset(half_season, PER < 13)
worse_per <- subset(worse_per, SALARY > 15 )
# getting players that played at least 41 games and have the best value PER compared to salary
best_value <- subset(half_season, PER > 20)
best_value <- subset(best_value, SALARY < 10)


# Reshaping the data to a long format
best_per_long <- pivot_longer(best_per,
                              cols = c("PER", "SALARY"),
                              names_to = "Metric",
                              values_to = "Value")

# Plotting
ggplot(best_per_long, aes(x = Player, y = Value, fill = Metric)) +
  geom_bar(stat = "identity", position = position_dodge(width = 0.7)) +
  scale_fill_manual(values = c("PER" = "blue", "SALARY" = "red")) +
  labs(x = "Player", y = "Value", fill = "Metric") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  ggtitle("Comparison of PER and Salary for Best Performing Players")
```



```r
# Reshaping the data to a long format
worse_per_long <- pivot_longer(worse_per,
                               cols = c("PER", "SALARY"),
                               names_to = "Metric",
                               values_to = "Value")
# Plotting
ggplot(worse_per_long, aes(x = Player, y = Value, fill = Metric)) +
  geom_bar(stat = "identity", position = position_dodge(width = 0.7)) +
  scale_fill_manual(values = c("PER" = "blue", "SALARY" = "red")) +
  labs(x = "Player", y = "Value", fill = "Metric") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  ggtitle("Comparison of PER and Salary for Worst Value Players")
```

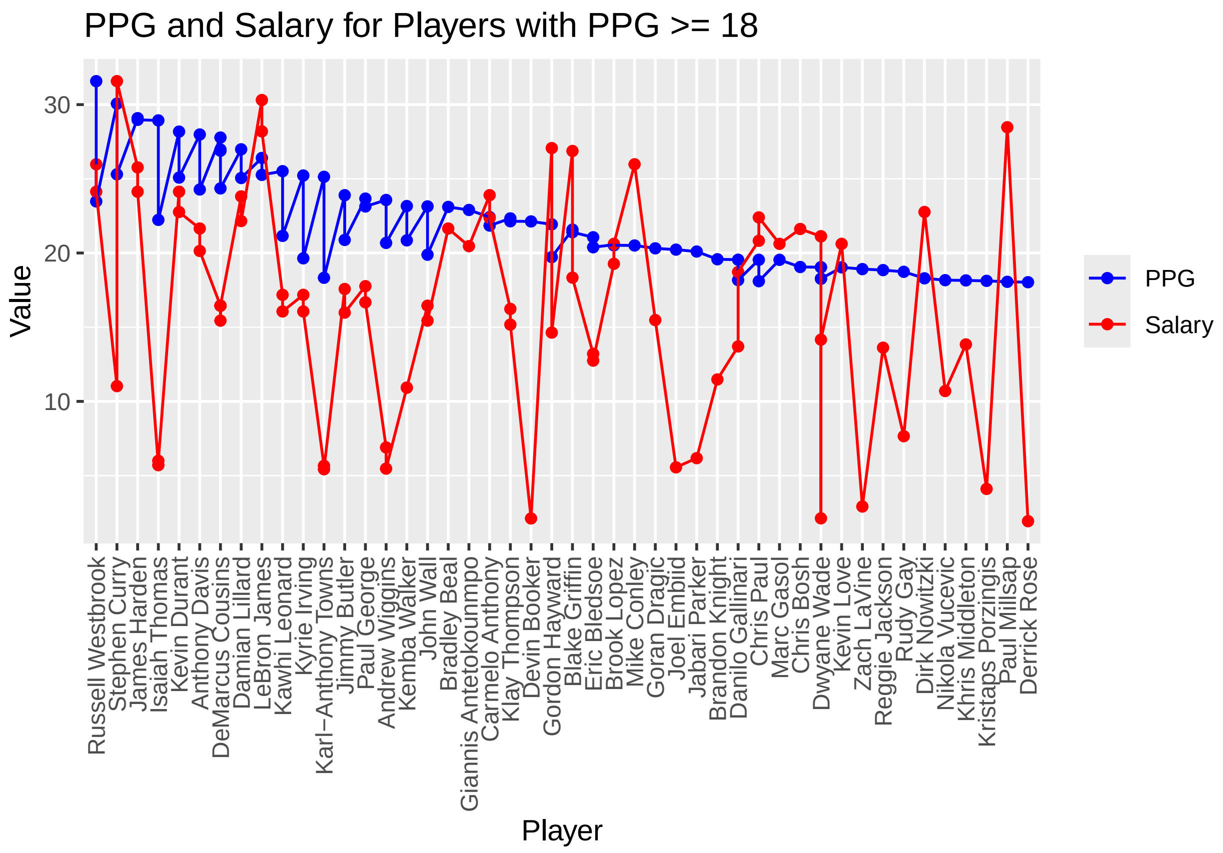


```r
# Reshaping the data to a long format
best_value_long <- pivot_longer(best_value,
                               cols = c("PER", "SALARY"),
                               names_to = "Metric",
                               values_to = "Value")

# Plotting
ggplot(best_value_long, aes(x = Player, y = Value, fill = Metric)) +
  geom_bar(stat = "identity", position = position_dodge(width = 0.7)) +
  scale_fill_manual(values = c("PER" = "blue", "SALARY" = "red")) +
  labs(x = "Player", y = "Value", fill = "Metric") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  ggtitle("Comparison of PER and Salary for Best Value Players")
```



These three graphs start to look at which players are providing the best and worst value. The first graph just shows the salaries for the players with the best PER over 2016 and 2017 seasons, which can also usually suggest the best overall performing players. The next two graphs provide better insight on the best and worst value contracts. One shows the lowest PER for players still getting paid at least 15 million annually, while the other shows the highest PER for players getting paid under 10 million annually.

```
# Filter to include only players with PPG >= 20.0
high_scorers <- subset(stats_2016_17_with_salary, PPG >= 18.0)

# Order players by PPG
high_scorers <- high_scorers[order(-high_scorers$PPG),]
# Ensure Player is treated as a factor for plotting
high_scorers$Player <- factor(high_scorers$Player, levels = unique(high_scorers$Player))
# Normalize SALARY to the same scale as PPG for visualization purposes
max_ppg <- max(high_scorers$PPG, na.rm = TRUE)
max_salary <- max(high_scorers$SALARY, na.rm = TRUE)
high_scorers$SALARY_normalized <- (high_scorers$SALARY / max_salary) * max_ppg

# Plot PPG (blue) and normalized Salary (red)
ggplot(high_scorers, aes(x = Player)) +
  geom_point(aes(y = PPG, color = "PPG")) +
  geom_point(aes(y = SALARY_normalized, color = "Salary")) +
  geom_line(aes(y = PPG, group = 1, color = "PPG")) +
  geom_line(aes(y = SALARY_normalized, group = 1, color = "Salary")) +
  scale_color_manual(values = c("PPG" = "blue", "Salary" = "red")) +
  labs(x = "Player", y = "Value", title = "PPG and Salary for Players with PPG >= 18") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1)) +
  theme(legend.title = element_blank())  # Remove legend title
```



This Graph shows points per game and salary, portraying the players that averaged the highest points scored a game and what their salaries were for those years. The players that have 2 values for PPG or Salary means in both 2016 and 2017 they averaged over 18 points per game, and the players with only one point means that they only averaged over 18 points per game one of the two seasons, and the respective salary that year.

One interesting thing we discovered already that we might look further into is the amount of value playeres there are when it comes to PPG. Perhaps going forward we will look to the other stats to see if/how we can predict a player might produce great value for their salary amount. For this round we just looked at the disparity between best PER players and worst, going forward we will likely look to take a random sample of players from the season when we do any sort of regression analysis. Getting a random sample should be doable given the large amount of data points we have.

PART 2 Intro For the second part of our project, we looked to expand upon finding linear relationships in the NBA salary dataset we explored in Part 1 of the project. In our initial study, we looked into the relationship between what a player was paid vs. their actual performance on the court. We identified statistics such as points per game (PPG) and player efficiency rating (PER) to have potentially interesting relationships with salary. In this second report on our project, we continued to look at these variables, as well as a few more, and further expanded our study looking into the value of players (how well they play vs. how much they're paid). We continued to utilize the NBA salary/stats data that we used before, but manipulated the data a bit (more on that ahead). The data we use is sourced from Kaggle, where a user scraped the components from basketballreference.com. Once again, we are using a specific collection of players as a sample to try to make claims about the greater population (all NBA players). Before we get into variables and our model, it is key to mention that our objective changed a bit. Our direction remained with value, but shifted a bit to a more specific problem, which is using statistics to try to predict what a given player will get paid in their next contract in free agency. Free agency in the NBA is a period over the summer where players whose contracts expired can either be re-signed by the team they just played for, or can go into the open market and have teams bid for them with certain contracts. Since our data spanned both the 2016 and 2017 seasons, we thought it would be interesting to look at the prospective free agents in 2016, take their stats, and predict what kind of salary they would be paid that summer. We could then compare it to what they actually ended up making. This brought our sample size down to around 61 players, which is notably small, but still gave us some interesting findings.

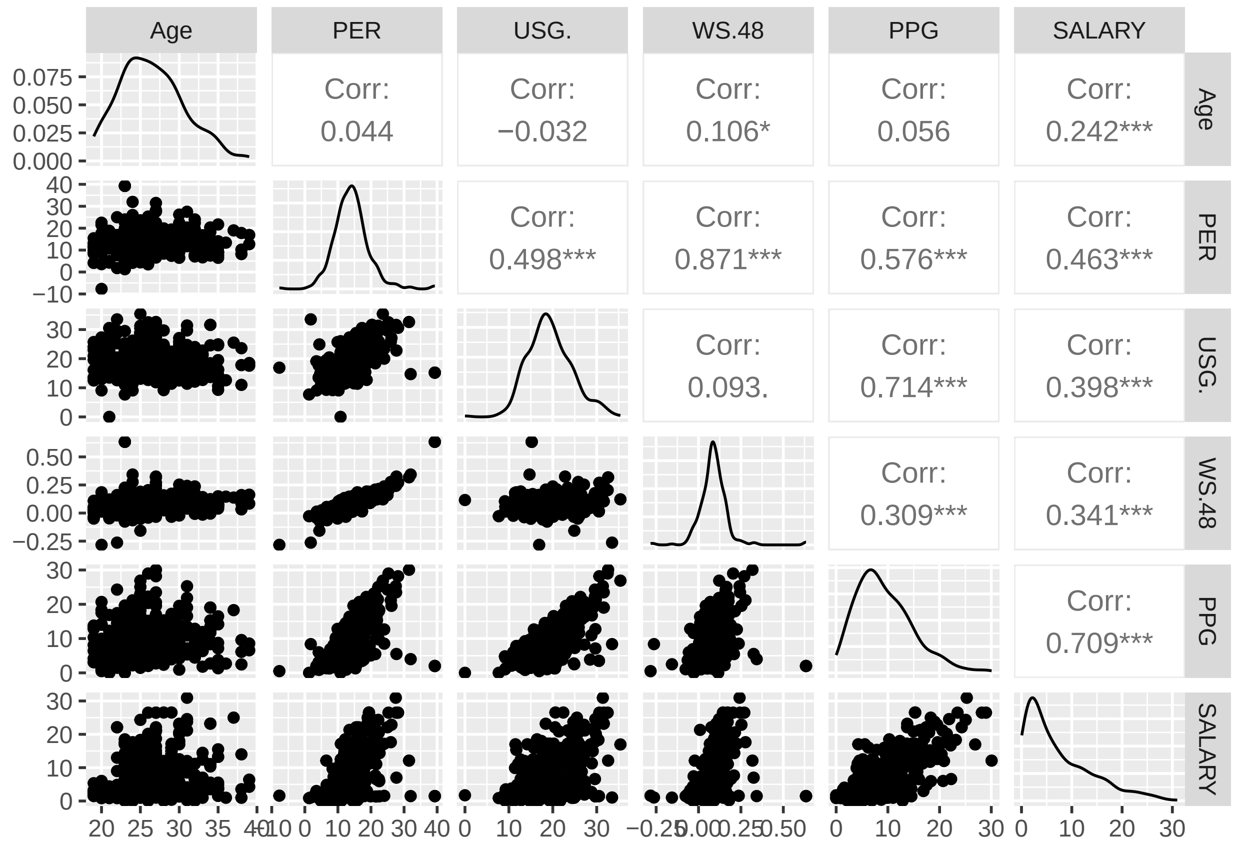
Exploring variables To identify which variables we should include in our model, we did a series of tests. First, we examined a table using the ggpairs() function in R:

```
require(GGally)
```

```
## Loading required package: GGally
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
ggpairs(stats_2016_salary, columns = c(5, 10, 21, 25, 48, 49))
```



Salary being our response variable, we identified what we thought could be some possible explanatory variables. We can see that PPG had the best correlation to salary, but the others were still worth looking at. Using this, along with our understanding of basketball, we collected a group of variables to test model combinations with: Age, PER, defensive win shares (DWS), box plus-minus (BPM), FG%, expected 3 point percentage (X3P), usage rate (USG), assist rate (AST), total rebound rate (TRB), and PPG.

```
## Loading required package: airports

## Loading required package: cherryblossom

## Loading required package: usdata

##
## Attaching package: 'openintro'

## The following object is masked from 'package:GGally':
##
##     tips

## Loading required package: broom
```

```
max_var <- 10  # Assuming 'SALARY' is one of the columns, and you want to include all other variables

col_best_lm <- regsubsets(SALARY ~ ., data = filtered2_stats2016,
                          nvmax = max_var)
col_best_lm %>%
  tidy()
```

```
## # A tibble: 10 x 15
##    `(Intercept)` Age   PER   DWS   BPM   FG.   X3P.  USG.  AST.  TRB.  PPG
##    <lgl>         <lgl> <lgl> <lgl> <lgl> <lgl> <lgl> <lgl> <lgl> <lgl> <lgl>
##  1 TRUE          FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE TRUE
##  2 TRUE          TRUE  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE TRUE
##  3 TRUE          TRUE  FALSE TRUE  FALSE FALSE FALSE FALSE FALSE FALSE TRUE
##  4 TRUE          TRUE  TRUE  TRUE  FALSE FALSE FALSE FALSE FALSE FALSE TRUE
##  5 TRUE          TRUE  TRUE  TRUE  FALSE FALSE FALSE FALSE FALSE TRUE  TRUE
##  6 TRUE          TRUE  TRUE  TRUE  FALSE FALSE FALSE TRUE  FALSE TRUE  TRUE
##  7 TRUE          TRUE  TRUE  TRUE  FALSE TRUE  FALSE TRUE  FALSE TRUE  TRUE
##  8 TRUE          TRUE  TRUE  TRUE  TRUE  TRUE  FALSE TRUE  FALSE TRUE  TRUE
##  9 TRUE          TRUE  TRUE  TRUE  TRUE  TRUE  FALSE TRUE  TRUE  TRUE  TRUE
## 10 TRUE          TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## # i 4 more variables: r.squared <dbl>, adj.r.squared <dbl>, BIC <dbl>,
## #   mallows_cp <dbl>
```

```
col_best_lm %>%
  coef(5)  # pull out the best 5 variable model
```

```
## (Intercept)         Age         PER         DWS        TRB.         PPG
## -13.8063494   0.4671929  -0.4320545   0.8067600   0.2477308   1.0180794
```

```r
col_best_summary <- summary(col_best_lm)

# You can then access the R-squared, adjusted R-squared, BIC, and Mallows Cp like this:
rsq <- col_best_summary$rsq
adjr2 <- col_best_summary$adjr2
bic <- col_best_summary$bic
cp <- col_best_summary$cp

# To view them, you can put them in a data frame
results <- data.frame(
  R.Squared = rsq,
  Adjusted.R.Squared = adjr2,
  BIC = bic,
  Mallows.Cp = cp
)

# Print the results
print(results)
```

```
##    R.Squared Adjusted.R.Squared       BIC Mallows.Cp
## 1  0.4989213          0.4899735 -31.95666  12.607481
## 2  0.5925216          0.5777042 -39.88918   2.165363
## 3  0.6103488          0.5887015 -38.42342   1.795628
## 4  0.6217776          0.5932325 -36.08962   2.276418
## 5  0.6358640          0.6008509 -34.23056   2.403942
## 6  0.6392992          0.5968638 -30.71987   3.947309
## 7  0.6450235          0.5953268 -27.58727   5.186389
## 8  0.6458560          0.5880365 -23.66300   7.075727
## 9  0.6462585          0.5799319 -19.66852   9.022223
## 10 0.6464256          0.5711971 -15.63549  11.000000
```

```r
regfit.full <- regsubsets(SALARY ~ ., data = filtered2_stats2016,
                           nvmax=NCOL(filtered2_stats2016)-1)

# Extract the summary of the regsubsets object
reg.summary <- summary(regfit.full)
reg.summary
```

```
## Subset selection object
## Call: regsubsets.formula(SALARY ~ ., data = filtered2_stats2016, nvmax = NCOL(filtered2_stats2016) -
##     1)
## 10 Variables  (and intercept)
##      Forced in Forced out
## Age      FALSE      FALSE
## PER      FALSE      FALSE
## DWS      FALSE      FALSE
## BPM      FALSE      FALSE
## FG.      FALSE      FALSE
## X3P.     FALSE      FALSE
## USG.     FALSE      FALSE
## AST.     FALSE      FALSE
## TRB.     FALSE      FALSE
## PPG      FALSE      FALSE
## 1 subsets of each size up to 10
## Selection Algorithm: exhaustive
##           Age PER DWS BPM FG. X3P. USG. AST. TRB. PPG
## 1  ( 1 )  " " " " " " " " " " " "  " "  " "  " "  "*"
## 2  ( 1 )  "*" " " " " " " " " " "  " "  " "  " "  "*"
## 3  ( 1 )  "*" " " "*" " " " " " "  " "  " "  " "  "*"
## 4  ( 1 )  "*" "*" "*" " " " " " "  " "  " "  " "  "*"
## 5  ( 1 )  "*" "*" "*" " " " " " "  " "  " "  "*"  "*"
## 6  ( 1 )  "*" "*" "*" " " " " " "  "*"  " "  "*"  "*"
## 7  ( 1 )  "*" "*" "*" " " "*" " "  "*"  " "  "*"  "*"
## 8  ( 1 )  "*" "*" "*" "*" "*" " "  "*"  " "  "*"  "*"
## 9  ( 1 )  "*" "*" "*" "*" "*" " "  "*"  "*"  "*"  "*"
## 10  ( 1 ) "*" "*" "*" "*" "*" "*"  "*"  "*"  "*"  "*"
```

```r
# Coefficient for best model according to Cp
reg.summary <- summary(regfit.full)
best.cp.model <- which.min(reg.summary$cp)
reg.summary$cp[best.cp.model]
```

```
## [1] 1.795628
```

```r
# Coefficient for best model according to Adjusted R squared
reg.summary <- summary(regfit.full)
best.adjr2.model <- which.max(reg.summary$adjr2)
reg.summary$adjr2[best.adjr2.model]
```

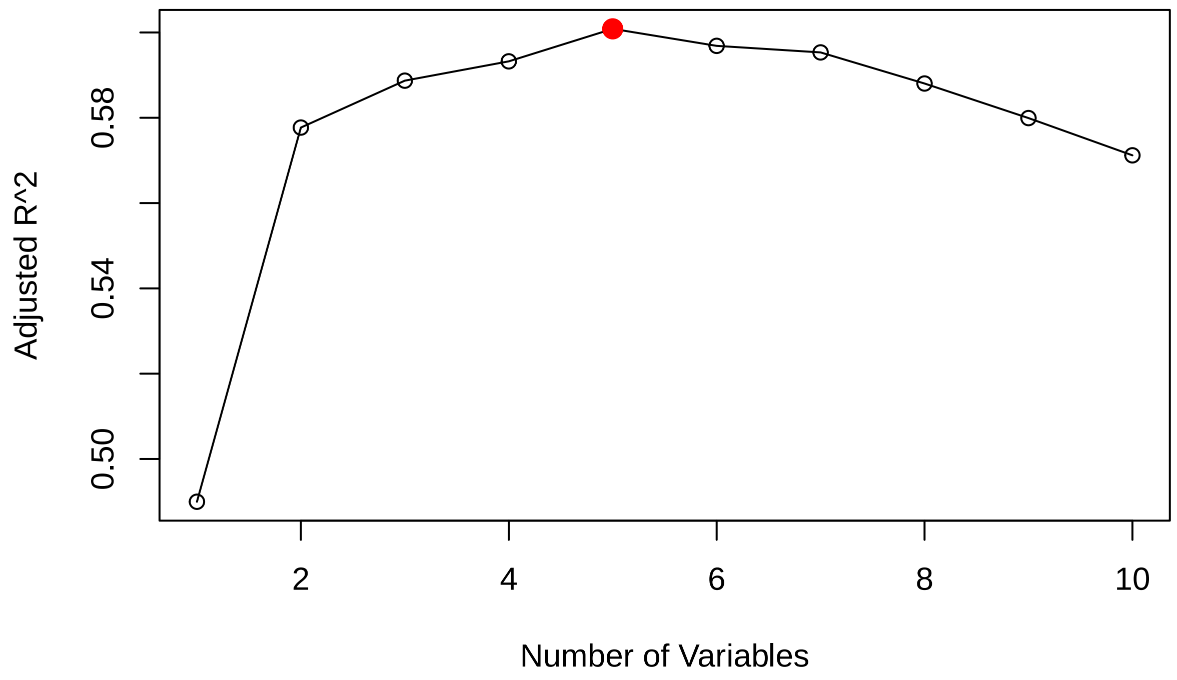
```
## [1] 0.6008509
```

```r
# Coefficients for best model according to BIC
reg.summary <- summary(regfit.full)
best.bic.model <- which.min(reg.summary$bic)
reg.summary$bic[best.bic.model]
```

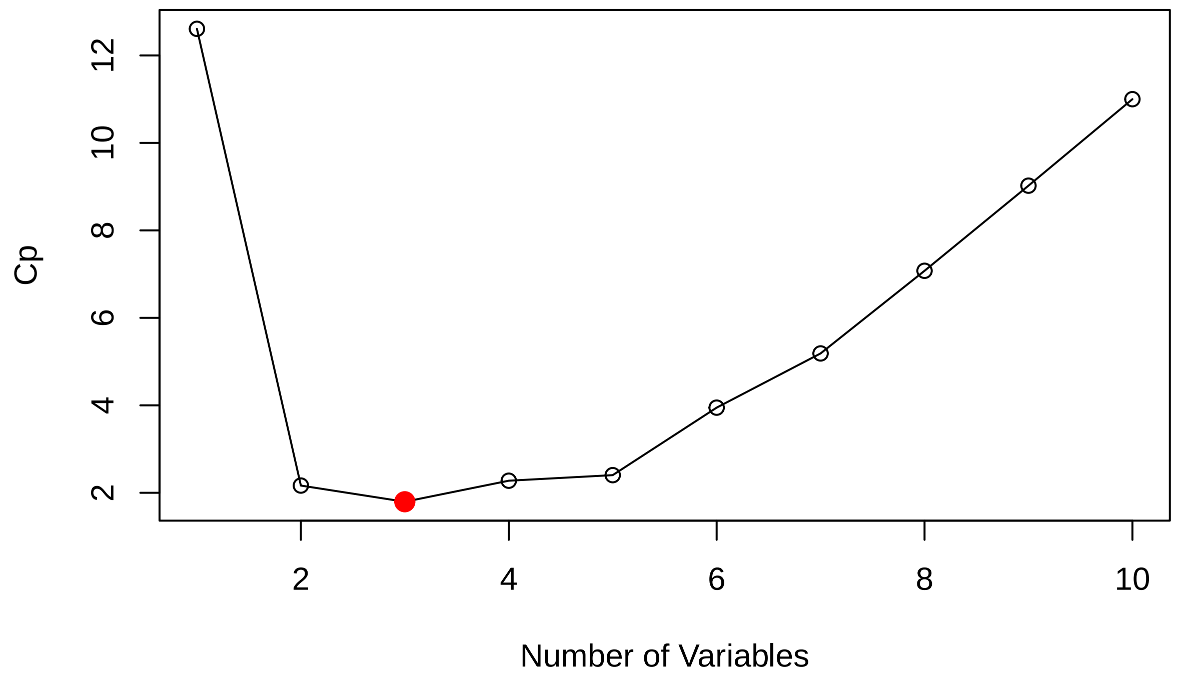
```
## [1] -39.88918
```

Selecting a Model Using the regsubsets() function in R, we were able to try a best subset selection approach to model selection. For each amount of variables, the output informed us which explanatory variables to select along with that model's adjusted R2 , BIC, and Cp values. From our studies, we know to look for models with higher adjusted R2 and a lower BIC and Cp values. Those three respective models were: R2 model: B0 +B1Age+B2PER +B3DWS +B4PPG + B55TR BIC model: B0 +B1Age + B2PPG Cp model: B0 + B1Age + B2DWS + B3PPG Identifying these became easy using a graph like such:

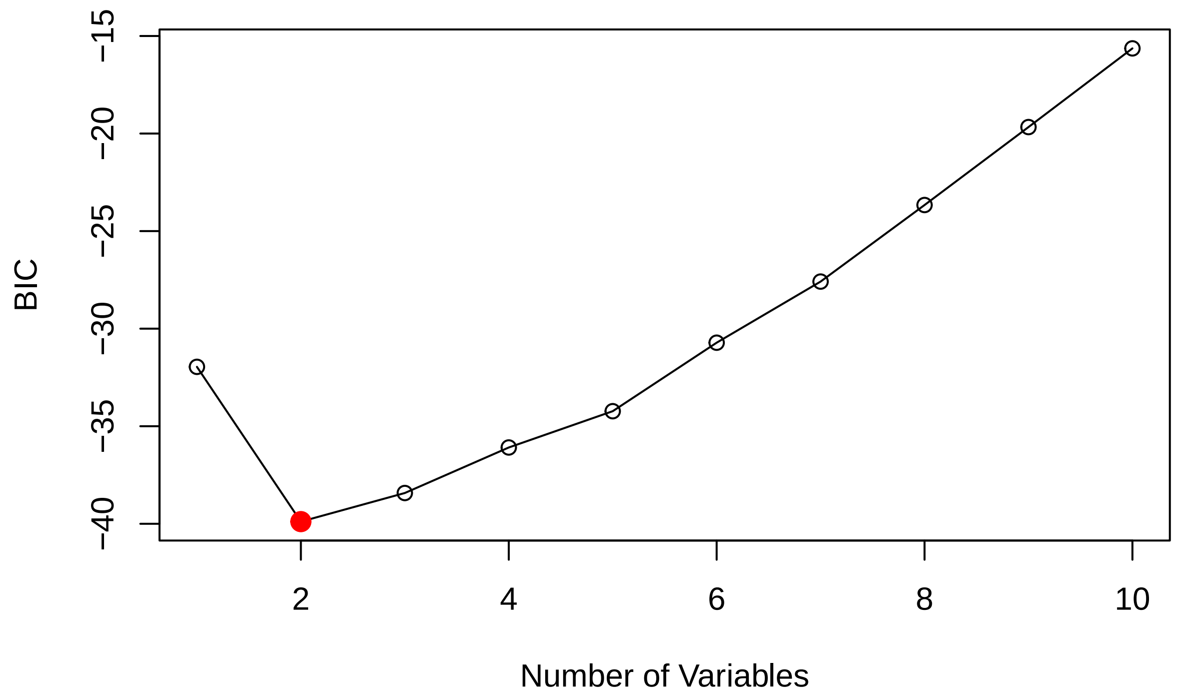
```r
plot(reg.summary$adjr2, xlab="Number of Variables", ylab="Adjusted R^2", type="o")
points(which.max(reg.summary$adjr2), reg.summary$adjr2[which.max(reg.summary$adjr2)], col="red", cex=2,
```



```
plot(reg.summary$cp, xlab="Number of Variables", ylab="Cp", type="o")
points(which.min(reg.summary$cp), reg.summary$cp[which.min(reg.summary$cp)], col="red", cex=2, pch=20)
```



```
plot(reg.summary$bic, xlab="Number of Variables", ylab="BIC", type="o")
points(which.min(reg.summary$bic), reg.summary$bic[which.min(reg.summary$bic)], col="red", cex=2, pch=2
```



```r
# Fit the models
model_adjr2 <- lm(SALARY ~ Age + PER + DWS + PPG + TRB., data = filtered2_stats2016)

model_cp <- lm(SALARY ~ Age + DWS + PPG, data = filtered2_stats2016)

model_bic <-  lm(SALARY ~ Age + PPG, data = filtered2_stats2016)

# Summarize the models
summary(model_adjr2)
```

```
##
## Call:
## lm(formula = SALARY ~ Age + PER + DWS + PPG + TRB., data = filtered2_stats2016)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.9595 -2.9056 -0.1898  1.9509  9.8242
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -11.5346     4.0298  -2.862  0.00594 **
## Age           0.3801     0.1265   3.005  0.00400 **
## PER          -0.2956     0.2226  -1.328  0.18973
## DWS           0.5812     0.5218   1.114  0.27024
## PPG           0.8868     0.1746   5.080 4.65e-06 ***
## TRB.          0.2775     0.1727   1.606  0.11394
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.05 on 55 degrees of freedom
## Multiple R-squared:  0.6031, Adjusted R-squared:  0.5671
## F-statistic: 16.72 on 5 and 55 DF,  p-value: 5.157e-10
```

```
summary(model_cp)
```

```
##
## Call:
## lm(formula = SALARY ~ Age + DWS + PPG, data = filtered2_stats2016)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -10.2429  -2.8660   0.1734   2.0438  10.2517
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -11.2271     3.7428  -3.000   0.0040 **
## Age           0.3832     0.1273   3.009   0.0039 **
## DWS           0.6737     0.5142   1.310   0.1955
## PPG           0.6810     0.1025   6.642 1.26e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.077 on 57 degrees of freedom
## Multiple R-squared:  0.5831, Adjusted R-squared:  0.5612
## F-statistic: 26.58 on 3 and 57 DF,  p-value: 6.976e-11
```

```
summary(model_bic)
```

```
##
## Call:
## lm(formula = SALARY ~ Age + PPG, data = filtered2_stats2016)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.0748  -2.5274   0.0402   2.1977   9.5679
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -11.23029    3.76582  -2.982  0.00418 **
## Age           0.39985    0.12748   3.136  0.00269 **
## PPG           0.74492    0.09076   8.208 2.75e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.103 on 58 degrees of freedom
## Multiple R-squared:  0.5706, Adjusted R-squared:  0.5558
## F-statistic: 38.53 on 2 and 58 DF,  p-value: 2.257e-11
```

```
stats::step(lm(SALARY ~ 1, data = ommited_stats2016),
SALARY ~ Age +PER+ DWS+ BPM+ FG. +X3P.+ USG.+ AST.+ TRB.+ PPG,
direction = "forward", test = "F")
```

```
## Start:  AIC=213.51
## SALARY ~ 1
##
##         Df Sum of Sq    RSS    AIC F value    Pr(>F)
## + PPG    1   1109.72 1114.5 175.43 55.7589 5.848e-10 ***
## + USG.   1    654.25 1570.0 195.31 23.3363 1.095e-05 ***
## + PER    1    628.07 1596.2 196.27 22.0350 1.771e-05 ***
## + BPM    1    490.81 1733.4 201.05 15.8560 0.0001991 ***
## + DWS    1    436.25 1788.0 202.85 13.6633 0.0004990 ***
## + AST.   1    182.36 2041.9 210.55  5.0013 0.0293296 *
## + Age    1    130.15 2094.1 212.01  3.4805 0.0673391 .
## + X3P.   1     75.66 2148.6 213.50  1.9720 0.1657607
## <none>               2224.2 213.51
## + FG.    1     16.60 2207.6 215.08  0.4212 0.5189893
## + TRB.   1     10.50 2213.8 215.24  0.2655 0.6083651
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Step:  AIC=175.43
## SALARY ~ PPG
##
##         Df Sum of Sq     RSS    AIC F value    Pr(>F)
## + Age    1   208.190  906.33 165.44 12.6338 0.0007871 ***
## + DWS    1    52.121 1062.40 174.66  2.6983 0.1061637
## <none>               1114.52 175.43
## + TRB.   1    18.139 1096.38 176.48  0.9099 0.3443071
## + USG.   1     9.871 1104.65 176.92  0.4915 0.4862163
## + X3P.   1     8.131 1106.39 177.01  0.4042 0.5275725
## + AST.   1     6.510 1108.01 177.09  0.3231 0.5720473
## + PER    1     4.492 1110.03 177.20  0.2226 0.6389390
## + BPM    1     3.468 1111.06 177.25  0.1717 0.6802477
## + FG.    1     0.008 1114.51 177.43  0.0004 0.9841147
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Step:  AIC=165.44
## SALARY ~ PPG + Age
##
##         Df Sum of Sq    RSS    AIC F value Pr(>F)
## + DWS    1    39.652 866.68 164.84  2.4706 0.1218
## <none>               906.33 165.44
## + USG.   1    15.572 890.76 166.44  0.9440 0.3356
## + PER    1    13.966 892.37 166.54  0.8452 0.3620
## + TRB.   1     8.783 897.55 166.88  0.5284 0.4704
## + AST.   1     6.559 899.77 167.02  0.3936 0.5330
## + FG.    1     2.084 904.25 167.31  0.1244 0.7257
## + BPM    1     1.225 905.11 167.36  0.0731 0.7879
## + X3P.   1     0.819 905.51 167.39  0.0488 0.8259
##
```

```
## Step:  AIC=164.85
## SALARY ~ PPG + Age + DWS
##
##        Df Sum of Sq    RSS    AIC F value Pr(>F)
## <none>              866.68 164.84
## + PER   1   25.4205 841.26 165.12  1.6015 0.2112
## + BPM   1   20.6245 846.06 165.45  1.2920 0.2608
## + AST.  1    5.5231 861.16 166.47  0.3399 0.5623
## + TRB.  1    2.1891 864.49 166.70  0.1342 0.7156
## + USG.  1    2.1107 864.57 166.70  0.1294 0.7205
## + X3P.  1    1.0223 865.66 166.78  0.0626 0.8034
## + FG.   1    0.3482 866.33 166.82  0.0213 0.8845

##
## Call:
## lm(formula = SALARY ~ PPG + Age + DWS, data = ommited_stats2016)
##
## Coefficients:
## (Intercept)          PPG          Age          DWS
##    -14.2732       0.7301       0.4547       0.7993
```

```
stats::step(lm(SALARY ~ Age +PER+ DWS+ BPM+ FG. +X3P.+ USG.+ AST.+ TRB.+ PPG, data=ommited_stats2016),
SALARY ~ Age +PER+ DWS+ BPM+ FG. +X3P.+ USG.+ AST.+ TRB.+ PPG,
direction = "backward", test = "F")
```

```
## Start:  AIC=173.21
## SALARY ~ Age + PER + DWS + BPM + FG. + X3P. + USG. + AST. + TRB. +
##     PPG
##
##        Df Sum of Sq     RSS    AIC F value    Pr(>F)
## - X3P.  1     0.372  786.81 171.24  0.0222 0.8821326
## - AST.  1     1.073  787.51 171.29  0.0641 0.8011952
## - BPM   1     1.222  787.66 171.30  0.0730 0.7881943
## - USG.  1    13.265  799.70 172.18  0.7928 0.3777948
## - FG.   1    14.624  801.06 172.28  0.8740 0.3546342
## - DWS   1    22.483  808.92 172.84  1.3437 0.2522430
## <none>               786.44 173.21
## - TRB.  1    27.767  814.20 173.22  1.6595 0.2039835
## - PER   1    47.848  834.29 174.64  2.8596 0.0974534 .
## - Age   1   210.078  996.51 184.94 12.5549 0.0009053 ***
## - PPG   1   279.699 1066.14 188.86 16.7157 0.0001685 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Step:  AIC=171.24
## SALARY ~ Age + PER + DWS + BPM + FG. + USG. + AST. + TRB. + PPG
##
##        Df Sum of Sq     RSS    AIC F value    Pr(>F)
## - AST.  1     0.895  787.70 169.30  0.0546 0.8162107
## - BPM   1     1.244  788.05 169.33  0.0759 0.7841363
## - USG.  1    13.332  800.14 170.21  0.8133 0.3716350
## - FG.   1    14.532  801.34 170.30  0.8865 0.3511358
## - DWS   1    24.197  811.01 170.99  1.4761 0.2303213
## <none>               786.81 171.24
## - TRB.  1    29.972  816.78 171.41  1.8285 0.1826444
## - PER   1    47.511  834.32 172.64  2.8984 0.0951347 .
## - Age   1   215.063 1001.87 183.25 13.1201 0.0007031 ***
## - PPG   1   279.641 1066.45 186.88 17.0598 0.0001440 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Step:  AIC=169.3
## SALARY ~ Age + PER + DWS + BPM + FG. + USG. + TRB. + PPG
##
##        Df Sum of Sq     RSS    AIC F value    Pr(>F)
## - BPM   1     1.852  789.56 167.44  0.1152 0.7357670
## - FG.   1    14.386  802.09 168.35  0.8949 0.3487880
## - USG.  1    15.509  803.21 168.43  0.9647 0.3308207
## - DWS   1    23.660  811.36 169.02  1.4718 0.2308811
## <none>               787.70 169.30
## - TRB.  1    29.832  817.54 169.46  1.8557 0.1793462
## - PER   1    46.835  834.54 170.65  2.9134 0.0941765 .
## - Age   1   214.612 1002.32 181.28 13.3502 0.0006287 ***
## - PPG   1   286.793 1074.50 185.31 17.8403 0.0001040 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Step:  AIC=167.44
## SALARY ~ Age + PER + DWS + FG. + USG. + TRB. + PPG
##
##        Df Sum of Sq     RSS    AIC F value    Pr(>F)
## - FG.   1     12.73  802.29 166.37  0.8063 0.3735224
## - USG.  1     16.16  805.71 166.62  1.0233 0.3166057
## <none>               789.56 167.44
## - TRB.  1     28.60  818.15 167.50  1.8109 0.1844681
## - DWS   1     46.22  835.77 168.74  2.9268 0.0933196 .
## - PER   1     68.88  858.44 170.29  4.3621 0.0418602 *
## - Age   1    212.91 1002.46 179.29 13.4828 0.0005857 ***
## - PPG   1    324.29 1113.85 185.40 20.5364 3.662e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Step:  AIC=166.37
## SALARY ~ Age + PER + DWS + USG. + TRB. + PPG
##
##        Df Sum of Sq     RSS    AIC F value    Pr(>F)
## - USG.  1     7.641  809.93 164.92  0.4857 0.4890139
## <none>               802.29 166.37
## - TRB.  1    36.663  838.95 166.96  2.3306 0.1330288
## - DWS   1    44.635  846.92 167.51  2.8374 0.0982069 .
## - PER   1    59.143  861.43 168.49  3.7596 0.0580463 .
## - Age   1   200.176 1002.46 177.29 12.7248 0.0007956 ***
## - PPG   1   312.066 1114.35 183.42 19.8375 4.621e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Step:  AIC=164.92
## SALARY ~ Age + PER + DWS + TRB. + PPG
##
##        Df Sum of Sq     RSS    AIC F value    Pr(>F)
## <none>               809.93 164.92
## - TRB.  1     31.33  841.26 165.12  2.0116 0.1620674
## - DWS   1     37.89  847.82 165.57  2.4328 0.1248890
## - PER   1     54.56  864.49 166.70  3.5031 0.0668833 .
## - Age   1    204.47 1014.40 175.97 13.1274 0.0006612 ***
## - PPG   1    483.17 1293.10 190.05 31.0212 9.124e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Call:
## lm(formula = SALARY ~ Age + PER + DWS + TRB. + PPG, data = ommited_stats2016)
##
## Coefficients:
## (Intercept)          Age          PER          DWS         TRB.          PPG
##    -13.8063       0.4672      -0.4321       0.8068       0.2477       1.0181
```

Not only did we look at best subset selection, but we also ran code in R for forward and backward selection with F-tests.

```
model_adjr2 %>%
augment(newdata = filtered2_stats2016) %>% mutate(resid=.fitted-SALARY)%>% summarize(SSEtest=sum(resid^
```

```
## # A tibble: 1 x 1
##   SSEtest
##     <dbl>
## 1    902.
```

```
predictions <- predict(model_adjr2, newdata = filtered2_stats2016)  # get model predictions
actuals <- filtered_stats2017$SALARY
mse <- mean((predictions - actuals)^2)
print(mse)
```

```
## [1] 49.34215
```

```
model_cp %>%
augment(newdata = filtered2_stats2016) %>% mutate(resid=.fitted-SALARY)%>% summarize(SSEtest=sum(resid^
```

```
## # A tibble: 1 x 1
##   SSEtest
##     <dbl>
## 1    948.
```

```
predictions2 <- predict(model_cp, newdata = filtered2_stats2016)  # get model predictions
actuals2 <- filtered_stats2017$SALARY
mse2 <- mean((predictions2 - actuals2)^2)
print(mse2)
```

```
## [1] 47.74868
```

```
model_bic %>%
augment(newdata = filtered2_stats2016) %>% mutate(resid=.fitted-SALARY)%>% summarize(SSEtest=sum(resid^
```

```
## # A tibble: 1 x 1
##   SSEtest
##     <dbl>
## 1    976.
```

```
predictions3 <- predict(model_bic, newdata = filtered2_stats2016)  # get model predictions
actuals3 <- filtered_stats2017$SALARY
mse3 <- mean((predictions3 - actuals3)^2)
print(mse3)
```

```
## [1] 51.47601
```

```
forward_model <- lm(SALARY ~ Age +PER+ DWS+ BPM+ USG.+ AST.+ PPG, data = filtered2_stats2016)
forward_model %>%
augment(newdata = filtered2_stats2016) %>% mutate(resid=.fitted-SALARY)%>% summarize(SSEtest=sum(resid^
```

```
## # A tibble: 1 x 1
##   SSEtest
##     <dbl>
## 1    920.
```

```
predictions4 <- predict(forward_model, newdata = filtered2_stats2016)  # get model predictions
actuals4 <- filtered_stats2017$SALARY
mse4 <- mean((predictions4 - actuals4)^2)
print(mse4)
```

```
## [1] 49.50457
```

```
backward_model <- lm(SALARY ~ Age + PPG + DWS + PER, data = filtered2_stats2016)
backward_model %>%
augment(newdata = filtered2_stats2016) %>% mutate(resid=.fitted-SALARY)%>% summarize(SSEtest=sum(resid^
```

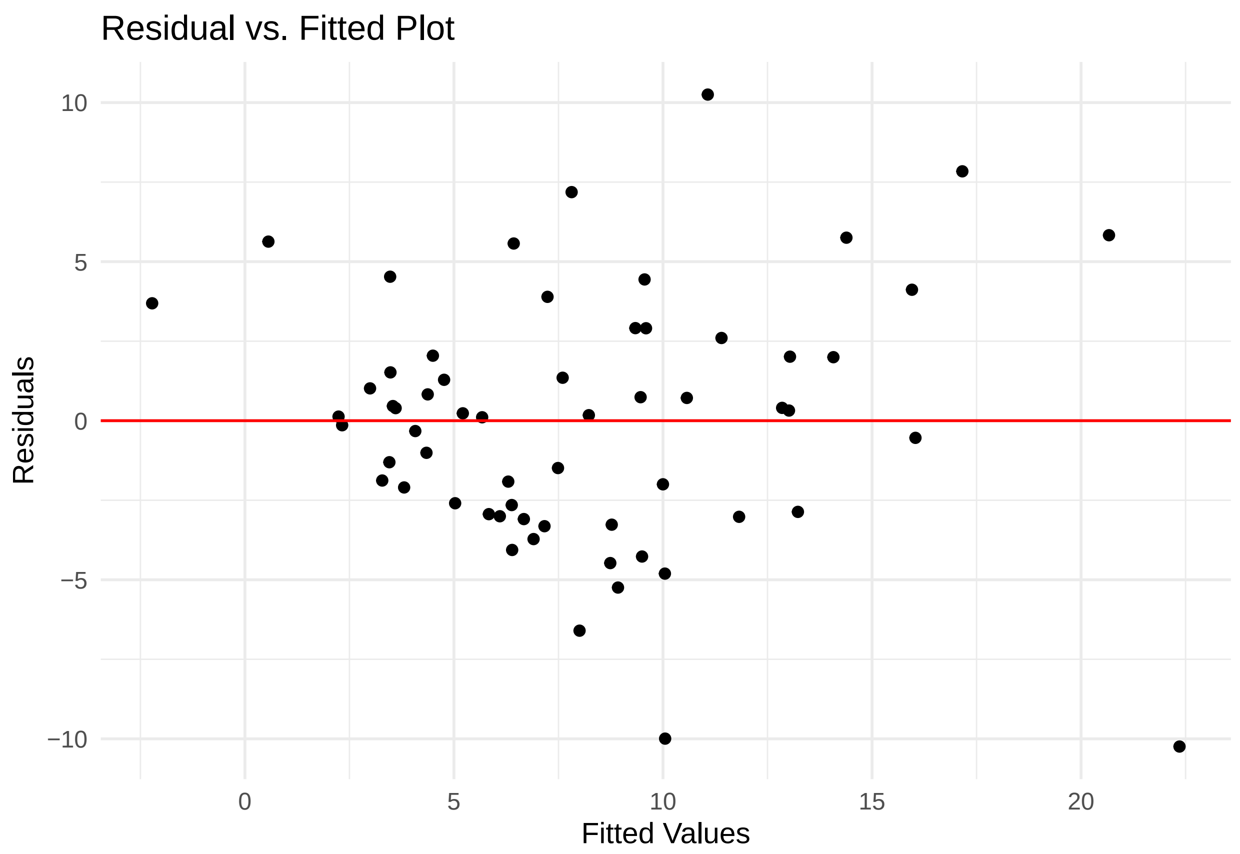
```
## # A tibble: 1 x 1
##   SSEtest
##     <dbl>
## 1    945.
```

```
predictions5 <- predict(backward_model, newdata = filtered2_stats2016)  # get model predictions
actuals5 <- filtered_stats2017$SALARY
mse5 <- mean((predictions5 - actuals5)^2)
print(mse5)
```

```
## [1] 47.78799
```

Above is a snapshot of the summary from our forward stepwise selection, where we see our largest jump in AIC value. We also see many more variables become significant in this step. A similar trend was seen in selecting the backward model. Those two looked like such: Forward Model: B0 +B1Age + B2PER + B3DWS + B4PPG + B5AST + B6BPM + B7USG Backward Model: B0 + B1Age + B2DWS + B3PPG +B4PER With five available models to pick from, all for separate reasons, we decided to look at the mean squared error of each of the models. We calculated this by looking at what each of the models projected as a salary for each of the players and compared that to the 2017 data which contained what they were actually paid. The 2017 data is acting as our test data in this instance. Taking the mean of the squared errors gave us an idea of how off each of the models were in their predictions, with a lower MSE being better. Two of the models tied for the lowest MSE at around 47 ;), one being the Cp model and one being the backward selection model. The only difference between the two models was that the backward model contained PER and the Cp did not. Taking all of the numbers into account, along with the situation we had with the data, we decided to go with the simpler model, the Cp model. It encapsulates age, PPG, and DWS.

```r
residual_data <- data.frame(
  Fitted = fitted(model_cp),
  Residuals = residuals(model_cp)
)

ggplot(residual_data, aes(x = Fitted, y = Residuals)) +
  geom_point() +
  geom_hline(yintercept = 0, color = "red") +
  labs(x = "Fitted Values", y = "Residuals", title = "Residual vs. Fitted Plot") +
  theme_minimal()
```

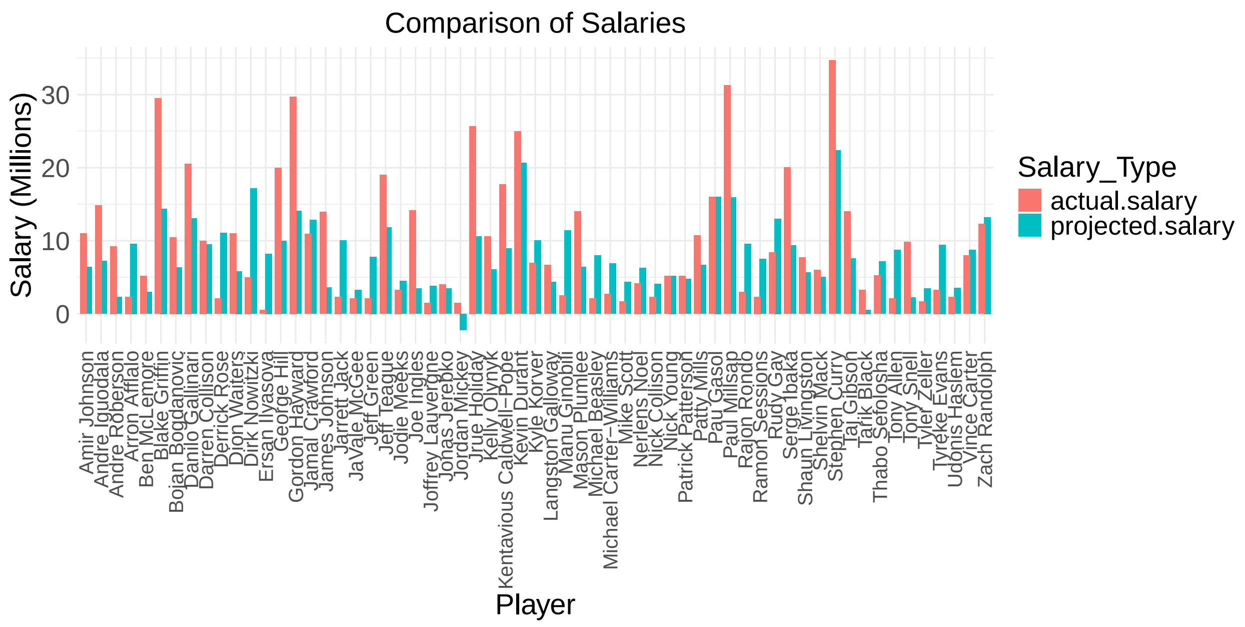


Residuals For the Cp model, above is what our residual plot looked like. Given our smaller sample size, it is promising to see a balanced distribution about the zero line (which is what the expected residual would be). There are a few points such as the one in the bottom right corner that could be troubling; three of our points have a residual greater than |10|.

Coefficients The B coefficients in our model are as follows: B0: -11.2271, B1: 0.3832, B2: 0.6737, B3: 0.6810. Three of our four coefficients are significant. We can see that PPG has *** significance, while B0 and Age have ** significance. DWS has a significance of 0.19, so not very significant, but DWS is a bit of a flawed stat. However, we think it is important to at least try to cover defense in our model, so we kept it in.

To see how well we predicted free agent's salaries, we plotted our expected salary values vs. what they actually got.

```r
library(tidyr)
# Assuming combined_data is your dataframe
combined_data_long <- pivot_longer(combined_data,
                                   cols = c("projected.salary", "actual.salary"),
                                   names_to = "Salary_Type",
                                   values_to = "Salary")




library(ggplot2)

ggplot(combined_data_long, aes(x = Player, y = Salary, fill = Salary_Type)) +
    geom_bar(stat = "identity", position = position_dodge(width = 0.7)) +
    labs(x = "Player", y = "Salary (Millions)", title = "Comparison of Salaries") +
    theme_minimal() +
    theme(
        axis.text.x = element_text(angle = 90, hjust = 1, vjust = 0.5, size = 14), # X-axis text size
        axis.text.y = element_text(size = 18), # Y-axis text size
        axis.title.x = element_text(size = 20), # X-axis title size
        axis.title.y = element_text(size = 20), # Y-axis title size
        plot.title = element_text(size = 20, hjust = 0.5), # Plot title size
        legend.title = element_text(size = 20), # Legend title size
        legend.text = element_text(size = 18) # Legend text size
    )
```



This graph shows our general trend of underestimating player salaries. The cases where we overestimate, such as Dirk Nowitski, Michael Beasley, and Rajon Rondo, all are old players. While our model takes into account age, the reality is that once players become extremely old basketball-wise (into their late 30s), they go on what are called "veteran-minimum contracts". These contracts are low-paying. This does not apply to every old player, but quite a few. There are a few reasons our model might be underestimating player salaries. The year before 2016, the salary cap jumped from $70 million to $94 million, which was and still is the largest salary cap jump in NBA history. When this cap jump occurred, players salaries did as well, and the players we are looking at in our sample were all on salaries that were established before this new massive cap. Our model does not take this cap jump into account, thus we are underestimating most players.

In terms of the actual components of our model, we find that it makes a good amount of sense looking at the terms we ended up selecting. PPG is the most important stat given that points are what win games, DWS brings in the defensive aspect of basketball, which is half of the game, and age demonstrates a players ability to stay in the league. Oftentimes a player who has made it to their late 20s in the league is a solid-good player and thus will be making more money. Obviously there are many more components that go into what a player will be paid; we a stat sheet won't show how good of a teammate a player is, it won't show a team's salary cap situation, it won't show a player's situation off the court. Teams take into account so much more than just stats when offering a player a contract, but we think it is still interesting to see how close we can get in predicting salaries. Going forward in our project, we will look to fully employ a testing/training data setup, and we will look to incorporate some new strategies to examine the data. We might do some sort of ridge regression or we might do something having to do with time series: the possibilities are endless. We will use the 2016 data that we just used to predict their salaries as our training data, and will then test our model on 2017 players and try to project their potential stats.

```r
library(car)
```

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

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

```
## The following object is masked from 'package:openintro':
##
##     densityPlot
```

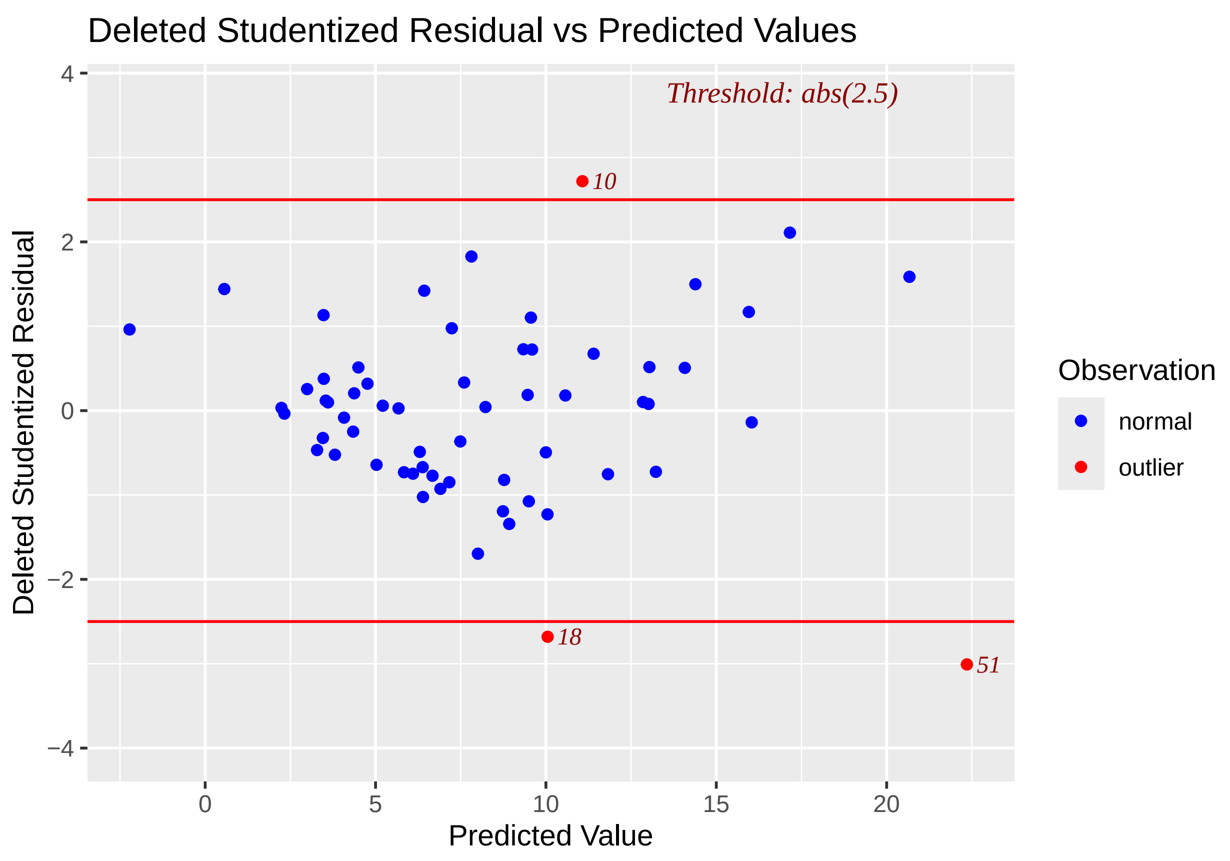
```
## The following object is masked from 'package:dplyr':
##
##     recode
```

```r
library(olsrr)
```

```
##
## Attaching package: 'olsrr'
```

```
## The following object is masked from 'package:datasets':
##
##     rivers
```

```r
ols_plot_resid_stud_fit(model_cp, threshold = 2.5, print_plot = TRUE)
```



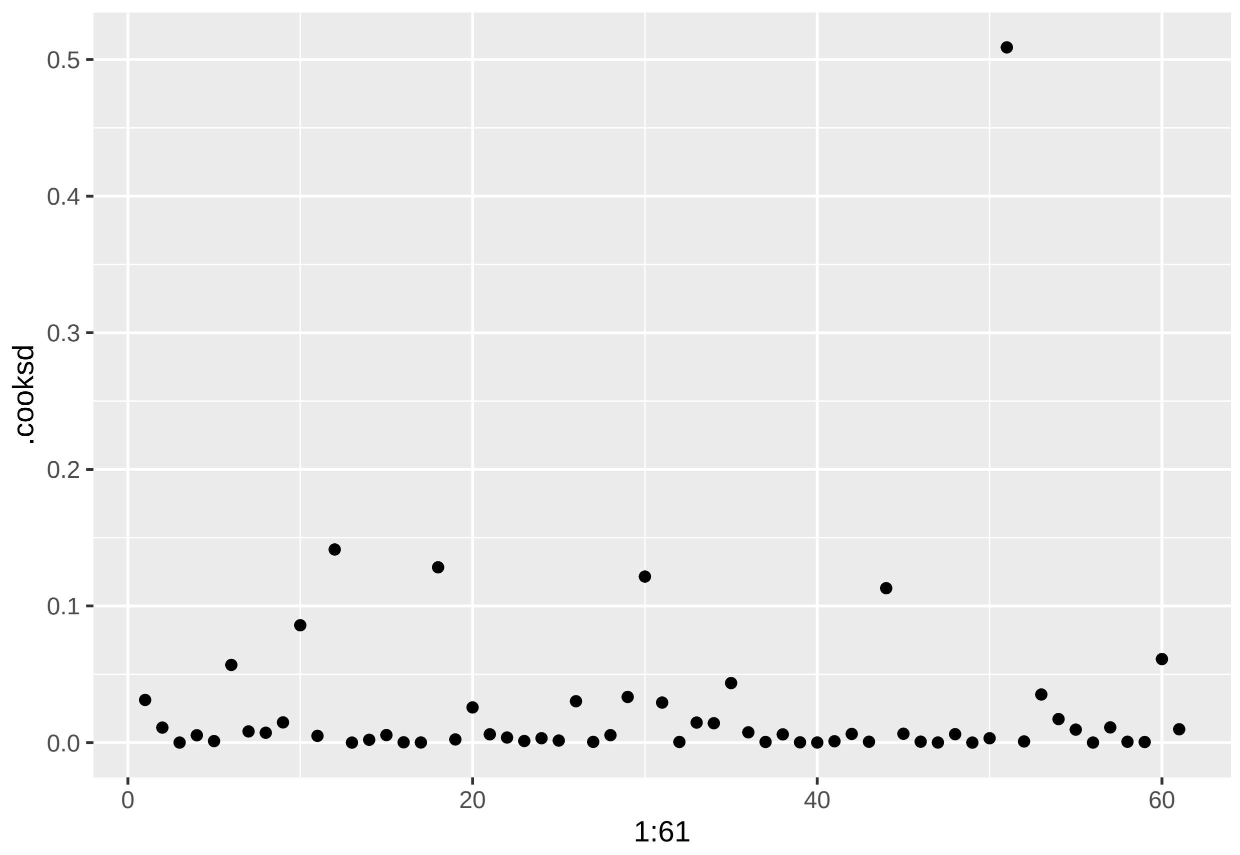
```
outlierTest(model_cp, cutoff = 0.01)
```

```
## No Studentized residuals with Bonferroni p < 0.01
## Largest |rstudent|:
##     rstudent unadjusted p-value Bonferroni p
## 51 -3.008363          0.0039305      0.23976
```

```
col_augment <- model_cp %>% augment() %>%
  mutate(Player = row.names(combined_data))
```

```
combined_data$dffits <- dffits(model_cp)
combined_data$dffits <- dfbetas(model_cp)
combined_data$cooksd <- cooks.distance(model_cp)
col_augment$cooksd <- cooks.distance(model_cp)
```

```
ggplot(data = col_augment) +
  geom_point(aes(x = 1:61, y = .cooksd))
```



```
position_codes <- setNames(c(1, 2, 3, 4, 5), c("PG", "SG", "SF", "PF", "C"))
stats_2016_salary$Pos <- position_codes[stats_2016_salary$Pos]

stats_2016_salary <- subset(stats_2016_salary, select = -Tm)
```

```
player_names_2016statssalary <- stats_2016_salary$Player  # Store player names
stats_2016_salary <- subset(stats_2016_salary, select = -Player)
  # Remove names from the main dataset

# Later, you can reference 'player_names' to relate findings back to individuals
stats_2016_salary <- na.omit(stats_2016_salary)
```

```
library(rsample)
split_stats_2016_salary <- initial_split(stats_2016_salary, prop = 8.5/10)
train_stats_2016_salary <- training(split_stats_2016_salary)
test_stats_2016_salary <- testing(split_stats_2016_salary)
```

RIDGE REGRESSION

```
library(glmnet)
```

```
## Loading required package: Matrix

##
## Attaching package: 'Matrix'
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
```

```
## Loaded glmnet 4.1-8
```

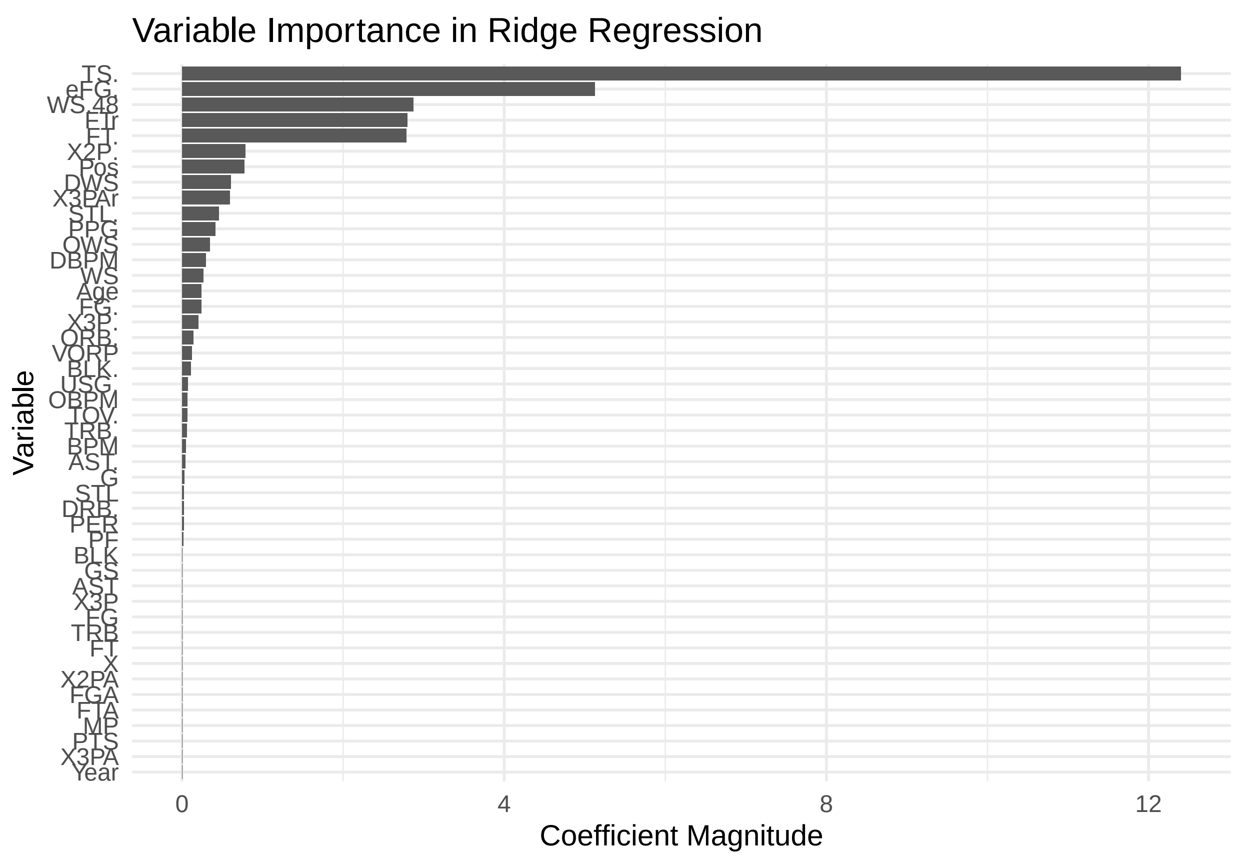
```
train_x_ridge <- as.matrix(train_stats_2016_salary [, -which(names(train_stats_2016_salary) == "SALARY")
train_y_ridge <- train_stats_2016_salary$SALARY


test_x <- model.matrix(~ . - SALARY, data = test_stats_2016_salary)[, -1]
test_y <- test_stats_2016_salary$SALARY  # This assumes SALARY is the target variable in your test data
```

```
set.seed(123)  # for reproducibility
ridge_model <- glmnet(train_x_ridge, train_y_ridge, alpha = 0)
cv_ridge <- cv.glmnet(train_x_ridge, train_y_ridge, alpha = 0, nfolds = 10)
best_lambda <- cv_ridge$lambda.min  # lambda that gives minimum mean cross-validated error

best_lambda
```

```
## [1] 0.4785973
```

```
coefficients <- coef(ridge_model, s = best_lambda)  # Extract coefficients at the selected lambda
# Convert to a regular numeric vector, removing the intercept if not needed
coef_vector <- as.numeric(coefficients[-1])
names(coef_vector) <- colnames(train_x_ridge)  # Assuming 'x' is your model matrix from training
important_vars_ridge <- sort(abs(coef_vector), decreasing = TRUE)




ridge_predictions <- predict(ridge_model, newx = test_x, s = best_lambda)
```

```
library(ggplot2)
coef_data <- data.frame(Variable = names(important_vars_ridge), Coef = important_vars_ridge)

ggplot(coef_data, aes(x = reorder(Variable, Coef), y = Coef)) +
    geom_bar(stat = "identity") +
    theme_minimal() +
    labs(title = "Variable Importance in Ridge Regression",
         x = "Variable",
         y = "Coefficient Magnitude") +
    coord_flip()  # Flipping coordinates for easier reading of variable names
```



```r
# Assuming you have 'predictions' from your ridge model and 'actuals' from your test data
ridge_predictions <- predict(ridge_model, newx = test_x, s = best_lambda)
actuals <- test_y  # Assuming test_y contains the actual salary values

# Calculate RMSE
rmse <- sqrt(mean((ridge_predictions - actuals)^2))

# Calculate MAE
mae <- mean(abs(ridge_predictions - actuals))

# Calculate R-squared
rss <- sum((ridge_predictions - actuals)^2)
tss <- sum((ridge_predictions - mean(actuals))^2)
r_squared <- 1 - (rss/tss)

# Print the metrics
print(paste("RMSE:", rmse))
```

```
## [1] "RMSE: 3.77120531034147"
```

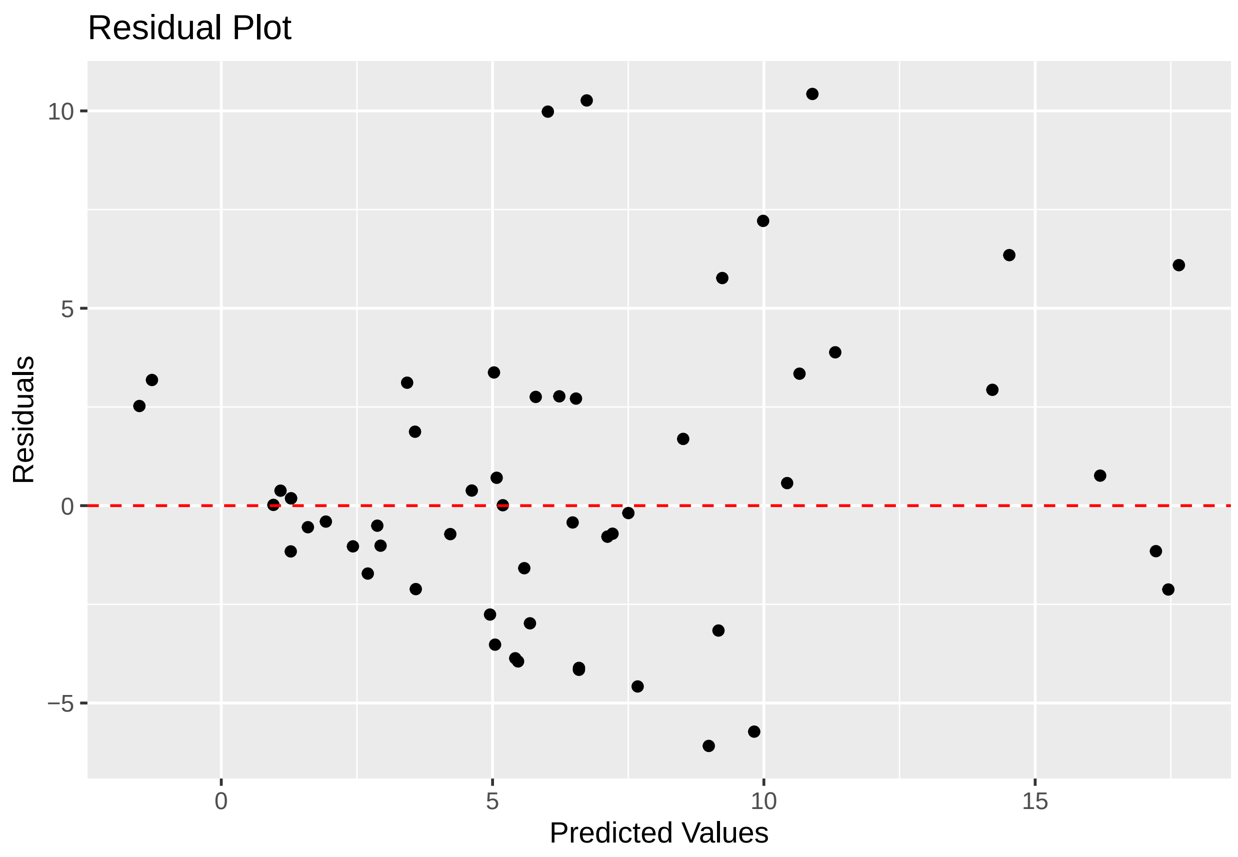
```r
print(paste("MAE:", mae))
```

```
## [1] "MAE: 2.77355579600853"
```

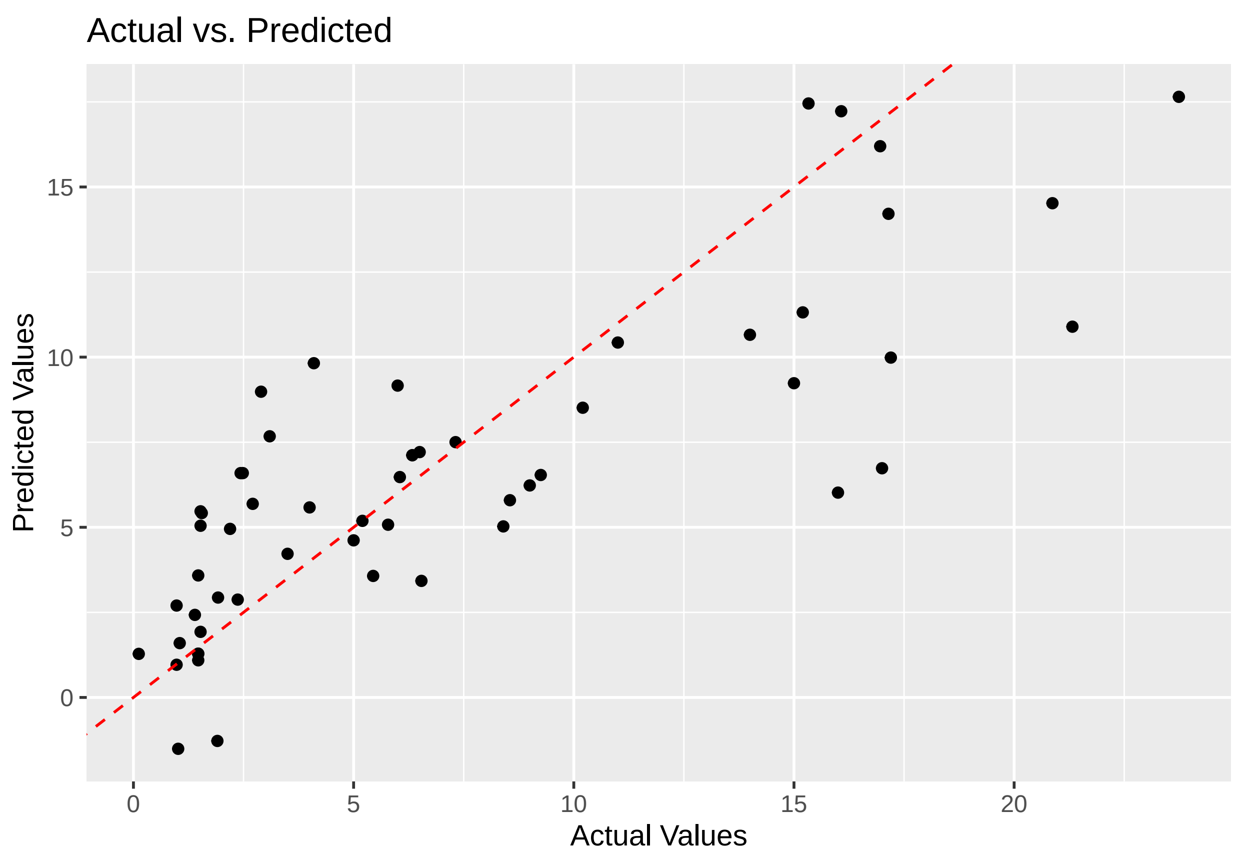
```
print(paste("R-squared:", r_squared))
```

```
## [1] "R-squared: 0.301262546618808"
```

```
library(ggplot2)

# Residual plot
residuals <- actuals - ridge_predictions
ggplot(data = NULL, aes(x = ridge_predictions, y = residuals)) +
    geom_point() +
    geom_hline(yintercept = 0, linetype = "dashed", color = "red") +
    labs(title = "Residual Plot", x = "Predicted Values", y = "Residuals")
```

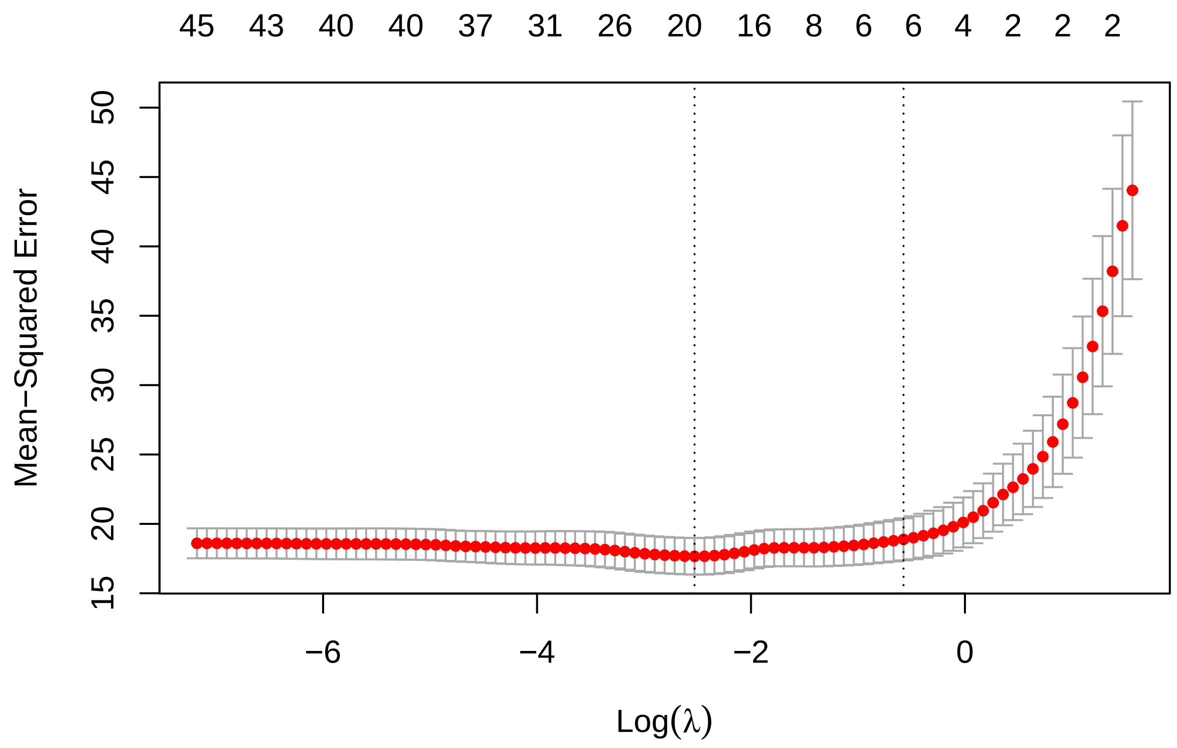


```
# Actual vs Predicted plot
ggplot(data = NULL, aes(x = actuals, y = ridge_predictions)) +
    geom_point() +
    geom_abline(intercept = 0, slope = 1, linetype = "dashed", color = "red") +
    labs(title = "Actual vs. Predicted", x = "Actual Values", y = "Predicted Values")
```



LASSO

```
set.seed(123)  # Set seed for reproducibility
cv_lasso <- cv.glmnet(train_x_ridge, train_y_ridge, alpha = 1, type.measure = "mse", nfolds = 10)

best_lambda_lasso <- cv_lasso$lambda.min  # Lambda with minimum MSE
lambda_1se_lasso <- cv_lasso$lambda.1se  # More conservative choice


plot(cv_lasso)
```



```
final_lasso_model <- glmnet(train_x_ridge, train_y_ridge, alpha = 1, lambda = best_lambda_lasso)

lasso_coefficients <- coef(final_lasso_model, s = best_lambda_lasso)
print(lasso_coefficients)
```

```
## 47 x 1 sparse Matrix of class "dgCMatrix"
##                        s1
## (Intercept) 28.3618008491
## X           -0.0012116015
## Year         .
## Pos          0.6323138853
## Age          0.2525413604
## G            .
## GS           .
## MP           0.0001389752
## PER          .
## TS.         -6.4201714816
## X3PAr        .
## FTr          0.5801571978
## ORB.        -0.1097506883
## DRB.         .
## TRB.         .
## AST.         .
## STL.         .
## BLK.         .
## TOV.        -0.0015333247
## USG.        -0.1449099395
## OWS          .
## DWS          0.4819545496
## WS           0.5215444280
## WS.48        .
## OBPM         .
## DBPM         0.0770859005
## BPM          .
## VORP         .
## FG           .
## FGA          .
## FG.          .
## X3P         -0.0010778292
## X3PA         .
## X3P.         .
## X2PA         0.0015685029
## X2P.         .
## eFG.         .
## FT           .
## FTA          .
## FT.         -2.9526532416
## TRB          .
## AST          0.0059340364
## STL         -0.0232490803
## BLK          .
## PF          -0.0110279845
## PTS          .
## PPG          0.6896400792
```

```r
predictions_lasso <- predict(final_lasso_model, newx = test_x, s = best_lambda_lasso)
```

```r
rmse <- sqrt(mean((predictions_lasso - actuals)^2))

# Calculate MAE
mae <- mean(abs(predictions_lasso - actuals))

# Calculate R-squared
ss_res <- sum((actuals - predictions_lasso)^2)  # Sum of squares of residuals
ss_tot <- sum((actuals - mean(actuals))^2)  # Total sum of squares
r_squared <- 1 - ss_res / ss_tot

# Print the metrics
print(paste("RMSE:", rmse))
```

```
## [1] "RMSE: 3.73030091395677"
```

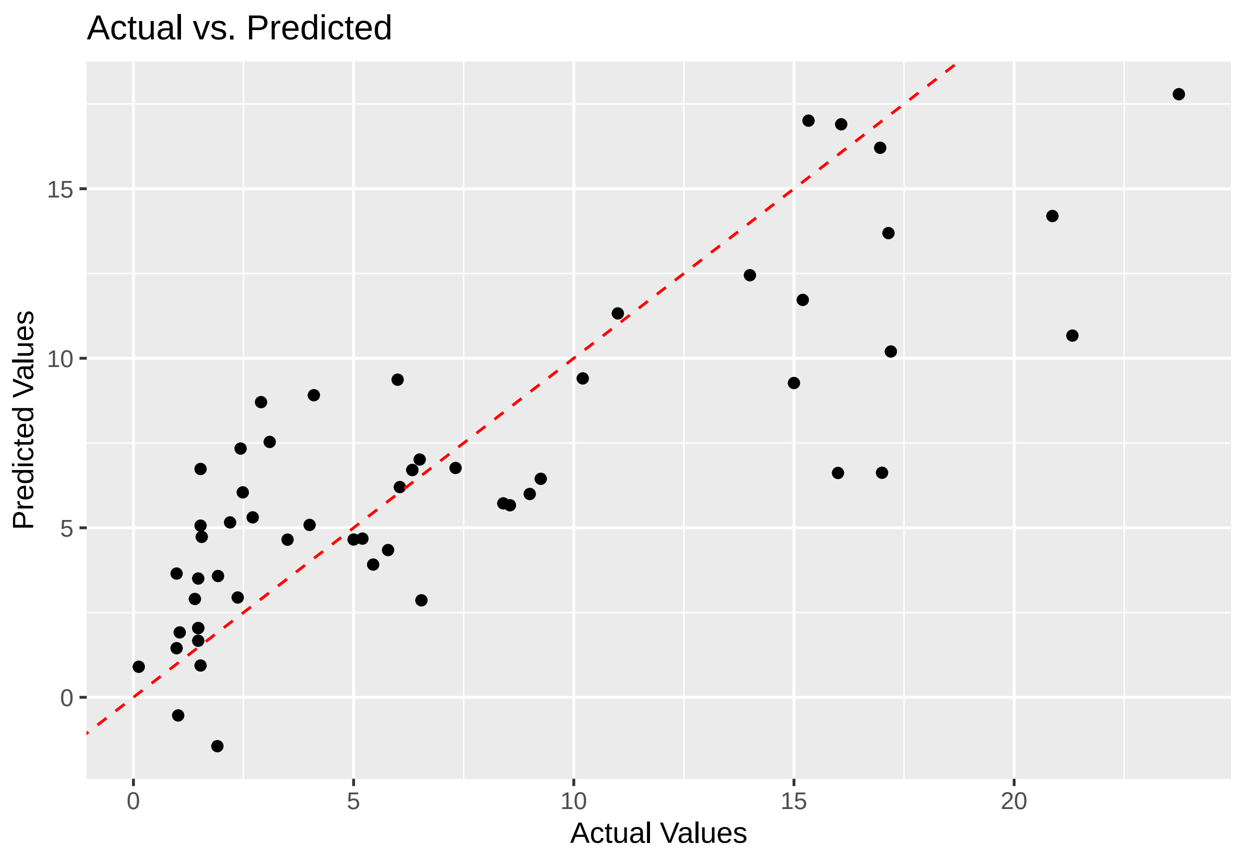
```r
print(paste("MAE:", mae))
```

```
## [1] "MAE: 2.74048682593163"
```

```
print(paste("R-squared:", r_squared))
```

```
## [1] "R-squared: 0.650868998856367"
```

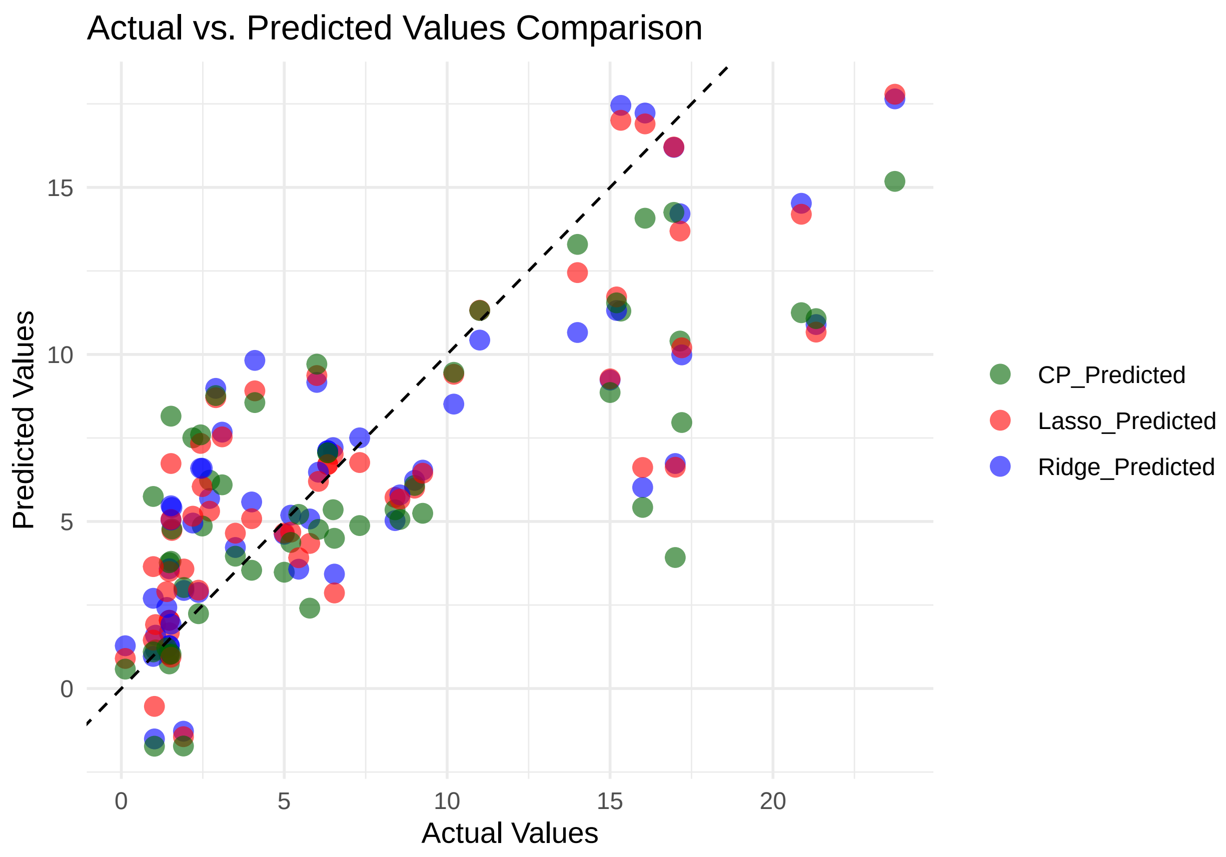
```
ggplot(data = NULL, aes(x = actuals, y = predictions_lasso)) +
    geom_point() +
    geom_abline(intercept = 0, slope = 1, linetype = "dashed", color = "red") +
    labs(title = "Actual vs. Predicted", x = "Actual Values", y = "Predicted Values")
```



```
#testing cp_model on training data
test_x_df <- as.data.frame(test_x)
cp_predictions <- predict(model_cp, newdata = test_x_df)
```

```
library(tidyr)



# Ensure 'cp_predictions' contains the predictions from the 'model_cp'
# and that 'ridge_predictions' and 'lasso_predictions' are already defined as previously
data_to_plot <- data.frame(
    Actual = test_y,
    Ridge_Predicted = as.numeric(ridge_predictions),
    Lasso_Predicted = as.numeric(predictions_lasso),
    CP_Predicted = as.numeric(cp_predictions)  # Ensure conversion to numeric if not already
```

```
)


# Reshape the data
data_long <- pivot_longer(
    data_to_plot,
    cols = c("Ridge_Predicted", "Lasso_Predicted", "CP_Predicted"),
    names_to = "Model",
    values_to = "Predicted"
)


# Load the ggplot2 package if not already loaded
library(ggplot2)

# Create the scatter plot
ggplot(data_long, aes(x = Actual, y = Predicted, color = Model)) +
    geom_point(alpha = 0.6, size = 3) +  # Increase the size parameter as needed
    geom_abline(intercept = 0, slope = 1, linetype = "dashed", color = "black") +
    labs(title = "Actual vs. Predicted Values Comparison",
         x = "Actual Values",
         y = "Predicted Values") +
    scale_color_manual(values = c("Ridge_Predicted" = "blue", "Lasso_Predicted" = "red", "CP_Predicted"
    theme_minimal() +
    theme(legend.title = element_blank())
```



SMOOTHING

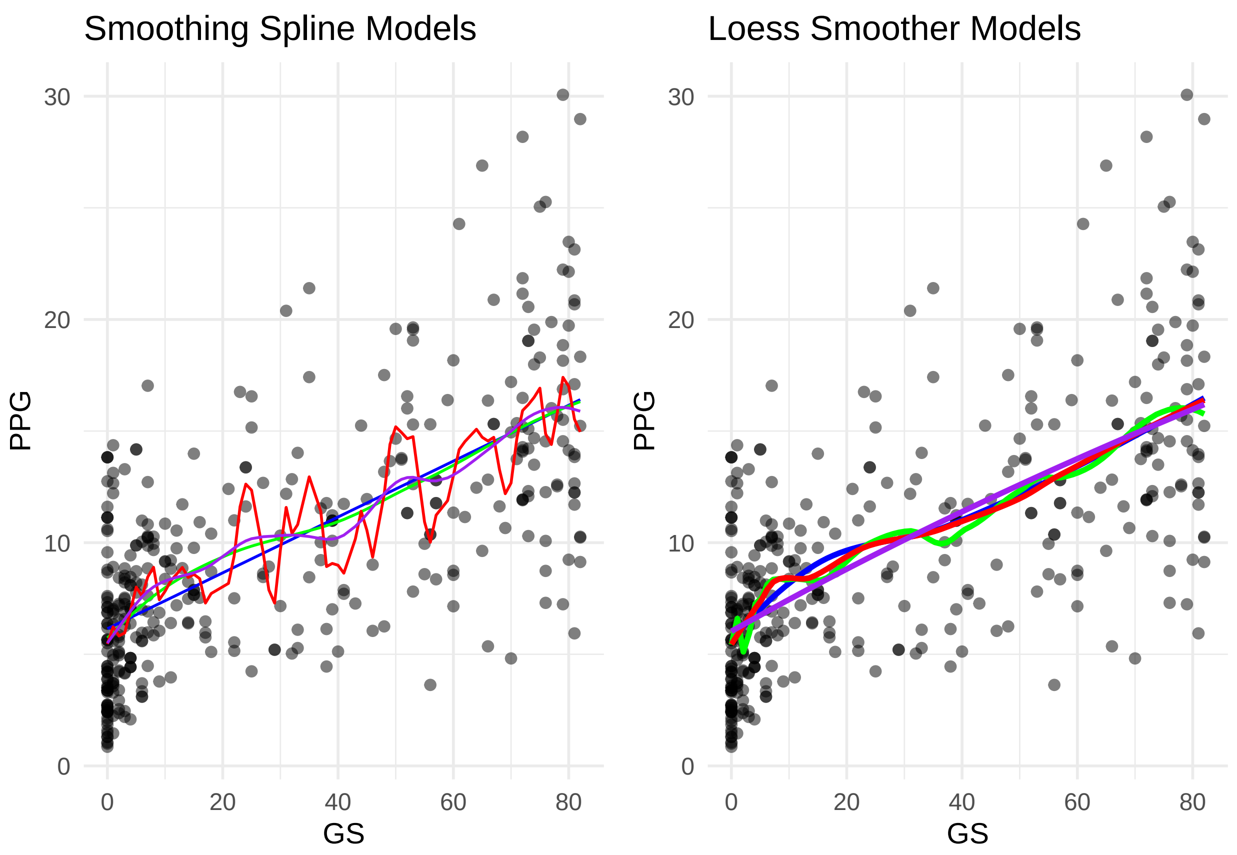
```r
# Fitting smoothing spline models with varying degrees of freedom
spline_model1 <- smooth.spline(stats_2016_salary$GS, stats_2016_salary$PPG, df = 2)
spline_model2 <- smooth.spline(stats_2016_salary$GS, stats_2016_salary$PPG, df = 5)
spline_model3 <- smooth.spline(stats_2016_salary$GS, stats_2016_salary$PPG, spar = 0.25)
spline_model4 <- smooth.spline(stats_2016_salary$GS, stats_2016_salary$PPG, spar = 0.65)



# Assuming 'spline_model1', 'spline_model2', etc., are your smoothing spline objects
data_spline1 <- data.frame(GS = spline_model1$x, PPG = predict(spline_model1)$y)
data_spline2 <- data.frame(GS = spline_model2$x, PPG = predict(spline_model2)$y)
data_spline3 <- data.frame(GS = spline_model3$x, PPG = predict(spline_model3)$y)
data_spline4 <- data.frame(GS = spline_model4$x, PPG = predict(spline_model4)$y)
```

```r
library(ggplot2)
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'

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

```r
# Basic plot setup
base_plot <- ggplot(stats_2016_salary, aes(x = GS, y = PPG)) +
  geom_point(alpha = 0.5) +
  theme_minimal()

# Add smoothing spline predictions
plot_spline <- base_plot +
  geom_line(data = data_spline1, aes(x = GS, y = PPG), col = "blue") +
  geom_line(data = data_spline2, aes(x = GS, y = PPG), col = "green") +
  geom_line(data = data_spline3, aes(x = GS, y = PPG), col = "red") +
  geom_line(data = data_spline4, aes(x = GS, y = PPG), col = "purple") +
  labs(title = "Smoothing Spline Models")


# Add loess predictions
plot_loess <- base_plot +
  geom_smooth(method = "loess", span = 0.7, col = "blue", se = FALSE) +
  geom_smooth(method = "loess", span = 0.25, col = "green", se = FALSE) +
  geom_smooth(method = "loess", span = 0.5, col = "red", se = FALSE) +
  geom_smooth(method = "loess", span = 5.0, col = "purple", se = FALSE) +
  labs(title = "Loess Smoother Models")

# Display plots side by side
grid.arrange(plot_spline, plot_loess, ncol = 2)
```

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

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,
## : pseudoinverse used at -0.41

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,
## : neighborhood radius 2.41

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,
## : reciprocal condition number 9.1679e-16

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,
## : There are other near singularities as well. 4

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



would probably choose the green or purple lines from the spline models (df = 5, or spar =0.65) as they both seem to capture the data but also dont overfit and is not as volatile as some of the other ones. From Loess smoother models I would probably select the model with span = 0.25 as that doesnt overfit while the otehr three either overfit or underfit and that one has the best variance bias tradeoff

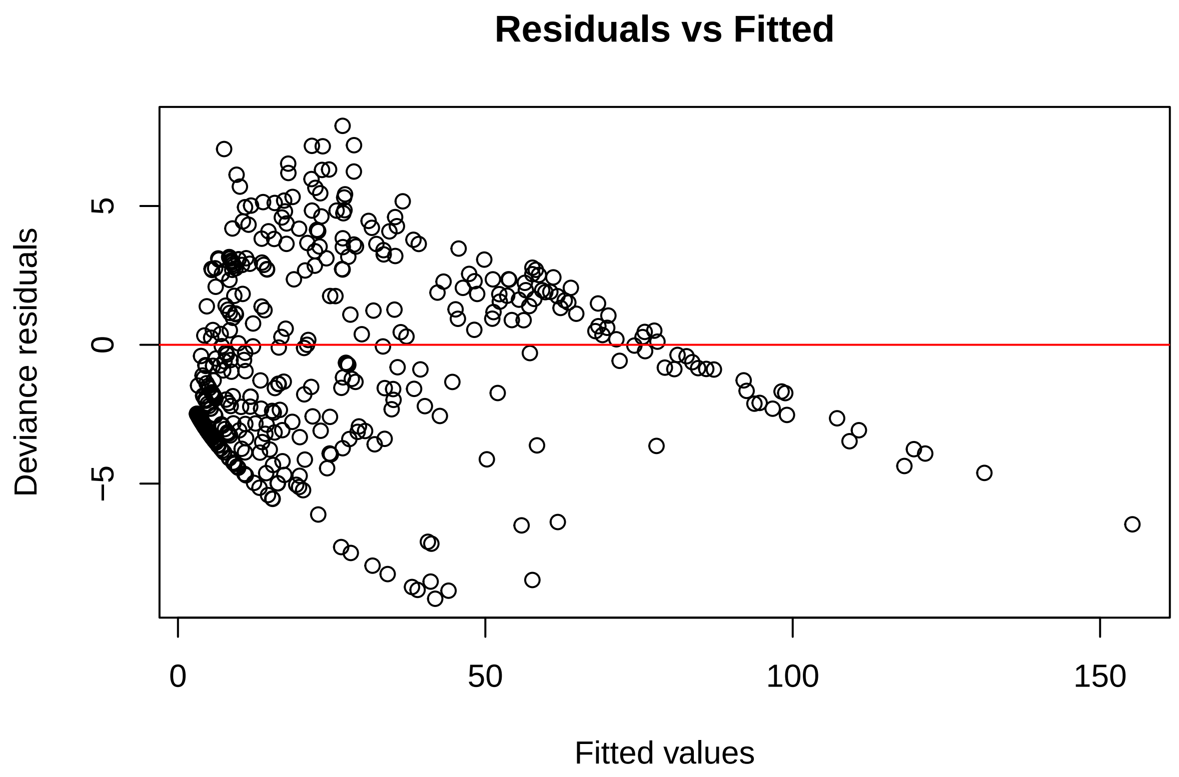
SOMETHING NEW

```r
stats_2017_salary$scored_20_plus <- as.integer(stats_2017_salary$PPG >= 20)
stats_2017_salary$Pos <- position_codes[stats_2017_salary$Pos]

# Aggregate data to get the count of games scoring 20+ points by player per season
stats_2017_salary_aggregated <- aggregate(GS ~ Player + SALARY + Pos + MP + PPG + Age + AST. + TRB. + DW
```

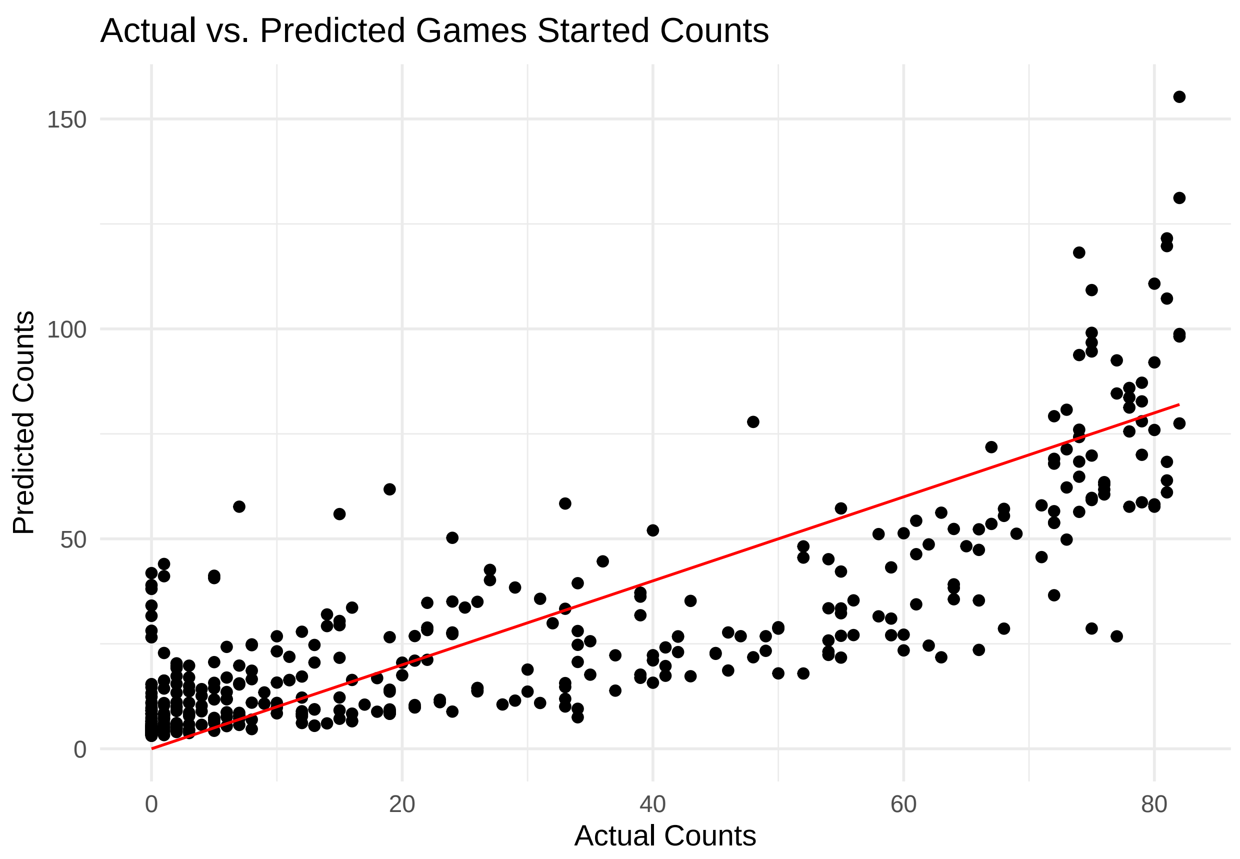
```r
library(glm2)

poisson_model <- glm(GS ~ SALARY + Pos + MP + PPG + Age + AST. + TRB. + DWS, family = poisson(), data =

# Summary of the model to see the effects
summary(poisson_model)
```

```
##
## Call:
## glm(formula = GS ~ SALARY + Pos + MP + PPG + Age + AST. + TRB. +
##     DWS, family = poisson(), data = stats_2017_salary_aggregated)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -9.148  -3.076  -1.410   1.921   7.887
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  1.0236212  0.0848371  12.066  < 2e-16 ***
## SALARY       0.0103585  0.0016131   6.421 1.35e-10 ***
## Pos         -0.0075962  0.0136088  -0.558   0.5767
## MP           0.0011776  0.0000235  50.113  < 2e-16 ***
## PPG          0.0023774  0.0023934   0.993   0.3206
## Age          0.0048223  0.0025483   1.892   0.0584 .
## AST.        -0.0057254  0.0013765  -4.160 3.19e-05 ***
## TRB.         0.0256908  0.0036823   6.977 3.02e-12 ***
## DWS         -0.0559863  0.0134476  -4.163 3.14e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 14868.1  on 447  degrees of freedom
## Residual deviance:  5048.3  on 439  degrees of freedom
## AIC: 6686.5
##
## Number of Fisher Scoring iterations: 5
```

```r
plot(residuals(poisson_model, type = "deviance") ~ fitted(poisson_model),
     xlab = "Fitted values",
     ylab = "Deviance residuals",
     main = "Residuals vs Fitted")
abline(h = 0, col = "red")
```



```
library(ggplot2)
actual_vs_predicted <- data.frame(Actual = stats_2017_salary_aggregated$GS,
                                  Predicted = predict(poisson_model, type = "response"))

ggplot(actual_vs_predicted, aes(x = Actual, y = Predicted)) +
    geom_point() +
    geom_line(aes(x = Actual, y = Actual), color = "red") +  # Adds a y=x reference line
    labs(x = "Actual Counts", y = "Predicted Counts", title = "Actual vs. Predicted Games Started Count
    theme_minimal()
```



because the 0 outcomes were higher then poisson predicted

```
library(pscl)
```

```
## Classes and Methods for R originally developed in the
## Political Science Computational Laboratory
## Department of Political Science
## Stanford University (2002-2015),
## by and under the direction of Simon Jackman.
## hurdle and zeroinfl functions by Achim Zeileis.
```

```
zip_model <- zeroinfl(GS ~ SALARY + Pos + MP + PPG + Age + AST. + TRB. + DWS, data = stats_2017_salary_a
summary(zip_model)
```

```
## Warning in sqrt(diag(object$vcov)): NaNs produced
```

```
##
## Call:
## zeroinfl(formula = GS ~ SALARY + Pos + MP + PPG + Age + AST. + TRB. +
##     DWS, data = stats_2017_salary_aggregated, dist = "poisson")
##
## Pearson residuals:
##     Min      1Q  Median      3Q     Max
## -6.0548 -1.1462 -0.4649  1.3572  6.6701
##
```

```
## Count model coefficients (poisson with log link):
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  1.732e+00  7.722e-02  22.434  < 2e-16 ***
## SALARY       1.149e-02  1.624e-03   7.075 1.50e-12 ***
## Pos         -4.973e-02  1.413e-02  -3.518 0.000434 ***
## MP           9.299e-04        NaN     NaN      NaN
## PPG          3.810e-03  1.802e-03   2.114 0.034524 *
## Age         -3.661e-05  2.532e-03  -0.014 0.988464
## AST.        -6.686e-03  1.368e-03  -4.887 1.02e-06 ***
## TRB.         3.045e-02  3.633e-03   8.381  < 2e-16 ***
## DWS         -3.222e-02  1.028e-02  -3.135 0.001716 **
##
## Zero-inflation model coefficients (binomial with logit link):
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  1.9658523  1.0512803   1.870  0.06149 .
## SALARY       0.0285968  0.0429615   0.666  0.50564
## Pos         -0.4981170  0.1822935  -2.733  0.00629 **
## MP          -0.0031756  0.0006874  -4.620 3.84e-06 ***
## PPG         -0.1119258  0.0583668  -1.918  0.05516 .
## Age          0.0207657  0.0353898   0.587  0.55736
## AST.        -0.0128372  0.0231412  -0.555  0.57908
## TRB.         0.0476049  0.0468665   1.016  0.30975
## DWS          0.7048629  0.5788396   1.218  0.22333
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Number of iterations in BFGS optimization: 22
## Log-likelihood: -2557 on 18 Df
```