harvardx-fantasy-football-predictions

Executive Summary

In this project, we are attempting to determine how to predict which players will perform best in a fantasy football season.

For those who don't know, fantasy football is an online game where a person selects a collection of NFL football players to form a team. Then that person competes against other people who have their own teams. A team's score is the sum of the scores of the collective players, where each player's score is based on their "real-life" performance in their game that week. The objective is to score more points than the other person's team for a given week.

The dataset comes from the 2018 dataset from a respected source: Pro Football Reference. This has basic statistics about football players from the 2018 season.

The goal of this analysis is to minimize RMSE (a measure of prediction error) in our fantasy point estimates. Thus, we will be making rating predictions based on the other data, and RMSE will measure how far off our predictions are. The better our predictions, the lower the RMSE.

Beyond this goal, there are several other goals to put learnings into practice. This report applies concepts such as web scraping, data exploration, visualization, model fitting, and model validation, to name a few.

The dataset itself can be found here:
https://www.pro-football-reference.com/years/2018/fantasy.htm (https://www.pro-football-reference.com/years/2018/fantasy.htm)

Analysis

The first thing we need to do is to import the data. Below we scrape data from Pro Football Reference's 2018 fantasy football website.

```
#set url variable
pro_fball_ref_2018 <- 'https://www.pro-football-reference.com/years/2018/fantasy.htm'

#use getURL to obtain the html content from the webpage
html <- getURL(pro_fball_ref_2018)

#read the html table in the content
pro_fball_ref_ffb_2018 <- readHTMLTable(html, header = TRUE, as.data.frame = TRUE, stringsAsFactor
s = FALSE)


#below is another potential approach, but not one that I used as the above worked well. I'll leave
it here for reference.
#http://bradleyboehmke.github.io/2015/12/scraping-html-tables.html
#html2 <- read_html('https://www.pro-football-reference.com/years/2018/fantasy.htm')
#pro_fball_ref_ffb_2018 <- html2 %>%
#  html_nodes("table") %>%
#  html_table(header = FALSE, trim = TRUE)
```

Now that we've imported the data, we'll clean up the table to get it ready for further processing.

```
#store the table as a dataframe (it currently exists as a list)
pro_fball_ref_ffb_2018_df <- ldply(pro_fball_ref_ffb_2018, data.frame)

#the header is repeated several times throughout the table. We'll remove the rows that just displa
y the header to clean up the data by first identifying the "bad" rows.
remove_row_ind <- which((with(pro_fball_ref_ffb_2018_df, FantPos == "FantPos" & Age == "Age")))

#and now remove all the bad rows from our data frame
pro_fball_ref_ffb_2018_df <- pro_fball_ref_ffb_2018_df[-remove_row_ind, ]

#much of the data exists as characters. convert the data types to appropriate values (numeric).
pro_fball_ref_ffb_2018_df <- pro_fball_ref_ffb_2018_df %>%
  mutate(Age = as.numeric(Age),
         G = as.numeric(G),
         GS = as.numeric(GS),
         Pass_Cmp = as.numeric(Cmp),
         Pass_Att = as.numeric(Att),
         Pass_Yds = as.numeric(Yds),
         Pass_TD = as.numeric(TD),
         Pass_Int = as.numeric(Int),
         Rush_Att = as.numeric(Att.1),
         Rush_Yds = as.numeric(Yds.1),
         Rush_Yds_per_Att = as.numeric(Y.A),
         Rush_TD = as.numeric(TD.1),
         Rec_Tgt = as.numeric(Tgt),
         Rec_Receptions = as.numeric(Rec),
         Rec_Yds = as.numeric(Yds.2),
         Rec_Yds_per_Rec = as.numeric(Y.R),
         Rec_TD = as.numeric(TD.2),
         Fmb = as.numeric(Fmb),
         Fmb_Loss = as.numeric(FL),
         Total_TD = as.numeric(TD.3),
         Two_Point_Conv = as.numeric(X2PM),
         Two_Point_Pass = as.numeric(X2PP),
         Fant_Pts = as.numeric(FantPt),
         PPR_Pts = as.numeric(PPR),
         DraftKing_Pts = as.numeric(DKPt),
         FanDuel_Pts = as.numeric(FDPt),
         Value_Over_Baseline = as.numeric(VBD),
         Rank_Pos = as.numeric(PosRank),
         Rank_Ovrl = as.numeric(OvRank)
         )

#remove the old character columns by identifying the old  columns
drop <- c("Cmp", "Att", "Yds", "TD", "Int", "Att.1", "Yds.1", "Y.A", "TD.1", "Tgt", "Rec", "Yds.2"
, "Y.R", "TD.2", "Fmb", "FL", "TD.3", "X2PM", "X2PP", "FantPt", "PPR", "DKPt", "FDPt", "VBD", "Pos
Rank", "OvRank")

#and remove these old columns
pro_fball_ref_ffb_2018_df <- pro_fball_ref_ffb_2018_df[, !names(pro_fball_ref_ffb_2018_df) %in% dr
op]
```

Outliers have the potential to distort and skew estimates to accommodate the outlier. We'll look for outliers and determine how to address them.

```
#use grubbs.test to statistically test for outliers. The overall goal is to predict fantasy point
s, so we'll use that variables when checking for outliers.


#check for outlier. Start with the high-end (opposite = FALSE)
grubbs.test(pro_fball_ref_ffb_2018_df$Fant_Pts,        #the vector for which we want to test if th
ere are outliers
            type = 10,             #test for one outlier
            opposite = FALSE,      #test for largest difference from mean
            two.sided = FALSE)      #the outlier can be on either end (high or low) but not two-si
ded
```

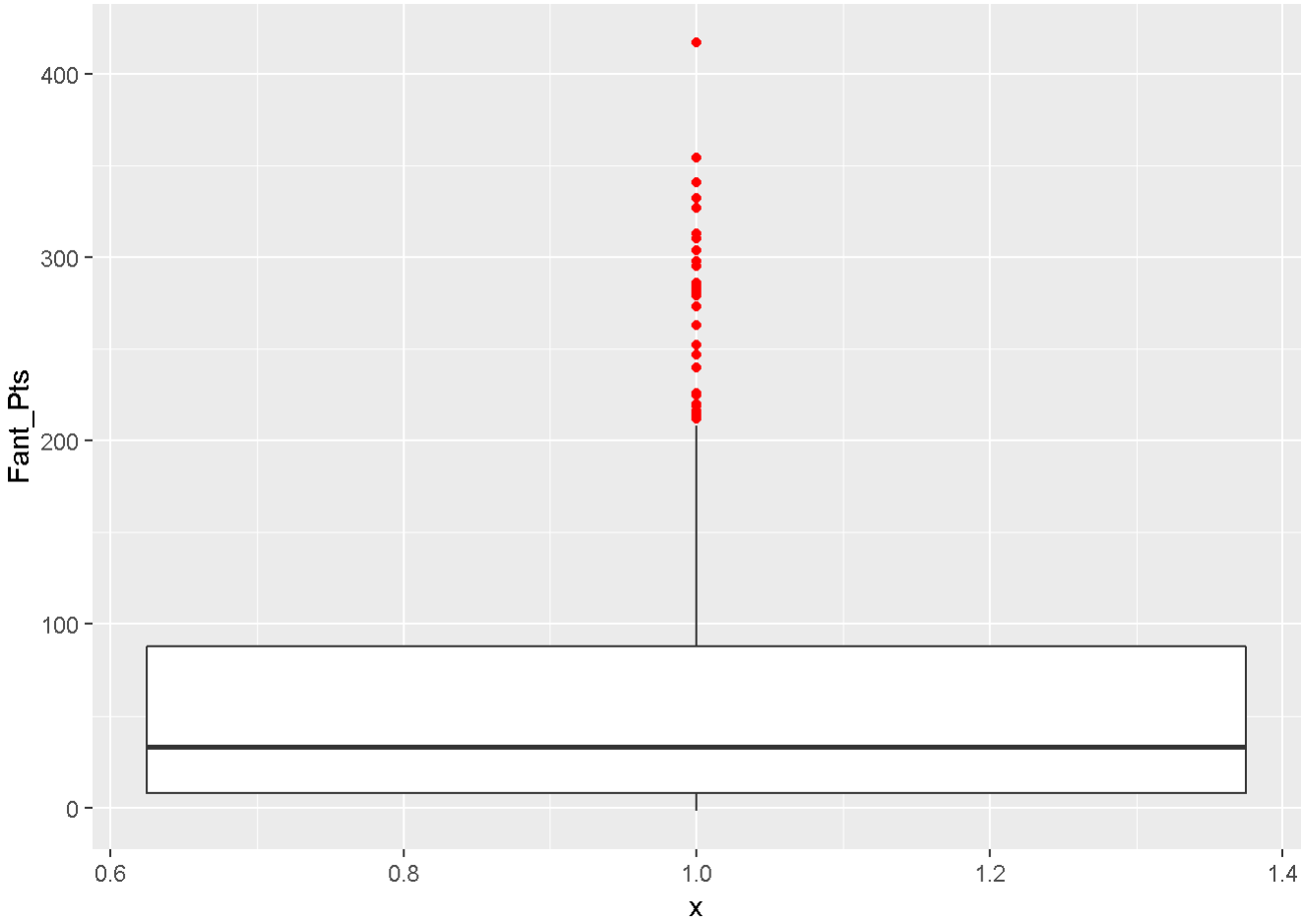
```
##
##  Grubbs test for one outlier
##
## data:  pro_fball_ref_ffb_2018_df$Fant_Pts
## G = 4.79800, U = 0.95776, p-value = 0.0003418
## alternative hypothesis: highest value 417 is an outlier
```

```
#This test suggests that 417 is an outlier. The p-value is below .01, so the results are statistic
ally significant.


#check for one outlier at a time. Now test the low-end (opposite = TRUE)
grubbs.test(pro_fball_ref_ffb_2018_df$Fant_Pts,        #the vector for which we want to test if th
ere are outliers
            type = 10,             #test for one outlier
            opposite = TRUE,      #test for largest difference from mean
            two.sided = FALSE)      #the outlier can be on either end (high or low) but not two-si
ded
```

```
##
##  Grubbs test for one outlier
##
## data:  pro_fball_ref_ffb_2018_df$Fant_Pts
## G = 0.86936, U = 0.99861, p-value = 1
## alternative hypothesis: lowest value -2 is an outlier
```

```
#This test suggests that -2 is not an outlier. The p-value is 1, so the results are not statistica
lly significant.

#look at the results visually - we see there are many values counted as outliers, but we'll focus
 on the most extreme value as if we can explain this outlier, the others should also be explainabl
e
pro_fball_ref_ffb_2018_df %>%
  ggplot(aes(x = 1, y = Fant_Pts)) +
  geom_boxplot(outlier.color = 'red')
```

```
## Warning: Removed 75 rows containing non-finite values (stat_boxplot).
```




```
#we can look at the entry associated with the 417 Fantasy Points ranking, which is coming through
 as an outlier. However, this datapoint makes sense and is for a player who performed extremely we
ll in 2018 (Patrick Mahomes). So we will not remove the outliers for this dataset as they represen
t true values.
pro_fball_ref_ffb_2018_df[which(pro_fball_ref_ffb_2018_df$Fant_Pts == 417), 1:5]
```

```
##       .id Rk            Player  Tm FantPos
## 5 fantasy  5 Patrick Mahomes*+ KAN      QB
```

Now that we've decided to keep the outliers, we'll summarize the dataset.

```
#summarize the dataset
summary(pro_fball_ref_ffb_2018_df)
```

```
##      .id                 Rk               Player
##  Length:622         Length:622         Length:622
##  Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character
##
##
##
##
##       Tm              FantPos               Age              G
##  Length:622         Length:622         Min.   :21.00   Min.   : 0.00
##  Class :character   Class :character   1st Qu.:24.00   1st Qu.: 6.00
##  Mode  :character   Mode  :character   Median :25.00   Median :12.00
##                                        Mean   :26.06   Mean   :10.55
##                                        3rd Qu.:28.00   3rd Qu.:16.00
##                                        Max.   :41.00   Max.   :16.00
##
##        GS            Pass_Cmp         Pass_Att         Pass_Yds
##  Min.   : 0.000   Min.   :  0.00   Min.   :  0.00   Min.   :   0.0
##  1st Qu.: 0.000   1st Qu.:  0.00   1st Qu.:  0.00   1st Qu.:   0.0
##  Median : 3.000   Median :  0.00   Median :  0.00   Median :   0.0
##  Mean   : 4.942   Mean   : 18.41   Mean   : 28.39   Mean   : 209.2
##  3rd Qu.: 9.000   3rd Qu.:  0.00   3rd Qu.:  0.00   3rd Qu.:   0.0
##  Max.   :16.000   Max.   :452.00   Max.   :675.00   Max.   :5129.0
##
##     Pass_TD          Pass_Int          Rush_Att         Rush_Yds
##  Min.   : 0.000   Min.   : 0.0000   Min.   :  0.00   Min.   : -16.00
##  1st Qu.: 0.000   1st Qu.: 0.0000   1st Qu.:  0.00   1st Qu.:   0.00
##  Median : 0.000   Median : 0.0000   Median :  1.00   Median :   0.00
##  Mean   : 1.357   Mean   : 0.6736   Mean   : 21.31   Mean   :  93.98
##  3rd Qu.: 0.000   3rd Qu.: 0.0000   3rd Qu.: 10.00   3rd Qu.:  44.00
##  Max.   :50.000   Max.   :16.0000   Max.   :304.00   Max.   :1434.00
##
##  Rush_Yds_per_Att     Rush_TD           Rec_Tgt       Rec_Receptions
##  Min.   :-11.000   Min.   : 0.0000   Min.   :  0.00   Min.   :  0.00
##  1st Qu.:  2.465   1st Qu.: 0.0000   1st Qu.:  1.00   1st Qu.:  1.00
##  Median :  4.130   Median : 0.0000   Median : 11.00   Median :  8.00
##  Mean   :  4.148   Mean   : 0.7042   Mean   : 27.57   Mean   : 18.41
##  3rd Qu.:  5.450   3rd Qu.: 0.0000   3rd Qu.: 41.75   3rd Qu.: 27.00
##  Max.   : 21.000   Max.   :17.0000   Max.   :170.00   Max.   :125.00
##  NA's   :295
##     Rec_Yds       Rec_Yds_per_Rec       Rec_TD          Fmb_Loss
##  Min.   : -11.0   Min.   :-11.000   Min.   : 0.000   Min.   :0.0000
##  1st Qu.:   4.0   1st Qu.:  7.817   1st Qu.: 0.000   1st Qu.:0.0000
##  Median :  77.0   Median : 10.045   Median : 0.000   Median :0.0000
##  Mean   : 209.3   Mean   : 10.521   Mean   : 1.355   Mean   :0.4228
##  3rd Qu.: 286.8   3rd Qu.: 12.880   3rd Qu.: 2.000   3rd Qu.:1.0000
##  Max.   :1677.0   Max.   : 48.000   Max.   :15.000   Max.   :7.0000
##                   NA's   :138
##     Total_TD     Two_Point_Conv  Two_Point_Pass    Fant_Pts
##  Min.   : 0.00   Min.   :1.000   Min.   :1.00   Min.   : -2.00
##  1st Qu.: 0.00   1st Qu.:1.000   1st Qu.:1.00   1st Qu.:  8.00
##  Median : 1.00   Median :1.000   Median :1.00   Median : 33.00
##  Mean   : 2.08   Mean   :1.182   Mean   :1.84   Mean   : 62.27
##  3rd Qu.: 3.00   3rd Qu.:1.000   3rd Qu.:2.00   3rd Qu.: 88.00
```

```
##  3rd Qu.: 9.00   3rd Qu.:1.000   3rd Qu.:2.00   3rd Qu.: 38.00
##  Max.   :21.00   Max.   :3.000   Max.   :5.00   Max.   :417.00
##                  NA's   :567     NA's   :597    NA's   :75
##     PPR_Pts       DraftKing_Pts     FanDuel_Pts     Value_Over_Baseline
##  Min.   : -2.20   Min.   : -1.20   Min.   : -2.00   Min.   :  1.00
##  1st Qu.: 11.80   1st Qu.: 12.10   1st Qu.:  9.90   1st Qu.: 20.75
##  Median : 48.70   Median : 53.00   Median : 41.50   Median : 46.50
##  Mean   : 81.69   Mean   : 85.38   Mean   : 72.19   Mean   : 52.43
##  3rd Qu.:127.20   3rd Qu.:133.60   3rd Qu.:106.80   3rd Qu.: 75.00
##  Max.   :417.10   Max.   :437.10   Max.   :429.10   Max.   :178.00
##  NA's   :65       NA's   :65       NA's   :65       NA's   :550
##     Rank_Pos        Rank_Ovrl
##  Min.   :  1.00   Min.   : 1.00
##  1st Qu.: 39.00   1st Qu.:20.25
##  Median : 79.00   Median :39.50
##  Mean   : 90.35   Mean   :39.50
##  3rd Qu.:131.00   3rd Qu.:58.75
##  Max.   :246.00   Max.   :78.00
##                   NA's   :544
```

```
#this gives the view of the first few rows of the highest-scoring players.
pro_fball_ref_ffb_2018_df %>% arrange(desc(Fant_Pts)) %>% head()
```

```
##       .id Rk             Player  Tm FantPos Age  G GS Pass_Cmp Pass_Att
## 1 fantasy  5  Patrick Mahomes*+ KAN      QB  23 16 16      383      580
## 2 fantasy 21          Matt Ryan ATL      QB  33 16 16      422      608
## 3 fantasy 27 Ben Roethlisberger PIT      QB  36 16 16      452      675
## 4 fantasy 31    Deshaun Watson* HOU      QB  23 16 16      345      505
## 5 fantasy 38       Andrew Luck* IND      QB  29 16 16      430      639
## 6 fantasy  1      Todd Gurley*+ LAR      RB  24 14 14        0        0
##   Pass_Yds Pass_TD Pass_Int Rush_Att Rush_Yds Rush_Yds_per_Att Rush_TD
## 1     5097      50       12       60      272             4.53       2
## 2     4924      35        7       33      125             3.79       3
## 3     5129      34       16       31       98             3.16       3
## 4     4165      26        9       99      551             5.57       5
## 5     4593      39       15       46      148             3.22       0
## 6        0       0        0      256     1251             4.89      17
##   Rec_Tgt Rec_Receptions Rec_Yds Rec_Yds_per_Rec Rec_TD Fmb_Loss Total_TD
## 1       0              0       0              NA      0        2        2
## 2       1              1       5            5.00      1        5        4
## 3       1              1      -1           -1.00      0        2        3
## 4       0              0       0              NA      0        3        5
## 5       2              1       4            4.00      0        1        0
## 6      81             59     580            9.83      4        1       21
##   Two_Point_Conv Two_Point_Pass Fant_Pts PPR_Pts DraftKing_Pts FanDuel_Pts
## 1              1             NA      417   417.1         437.1       429.1
## 2             NA              2      354   355.0         373.0       361.5
## 3             NA              4      341   341.9         362.9       357.4
## 4             NA             NA      332   331.7         349.7       340.7
## 5             NA              2      327   327.9         349.9       342.4
## 6              3             NA      313   372.1         379.1       342.6
##   Value_Over_Baseline Rank_Pos Rank_Ovrl
## 1                 134        1         5
## 2                  71        2        21
## 3                  58        3        27
## 4                  49        4        31
## 5                  44        5        38
## 6                 178        1         1
```

From the summary, we saw missing values for several variables. In most cases, NA's are expected as part of the nature of the game. For example, we don't expect quarterbacks to have receiving yards, so some blank reception variables are reasonable.

```
#After looking at the types of positions having these blank values, it makes sense overall.

# - Rush_Yds_per_Att - highest counts for QB and WR, as expected
pro_fball_ref_ffb_2018_df[is.na(pro_fball_ref_ffb_2018_df$Rush_Yds_per_Att),] %>% group_by(FantPo
s) %>% tally()
```

```
## # A tibble: 5 x 2
##   FantPos     n
##   <chr>   <int>
## 1 ""         67
## 2 QB          4
## 3 RB          7
## 4 TE        107
## 5 WR        110
```

```
# - Rec_Yds_per_Rec - highest counts for QB and RB, as expected
pro_fball_ref_ffb_2018_df[is.na(pro_fball_ref_ffb_2018_df$Rec_Yds_per_Rec),] %>% group_by(FantPos)
%>% tally()
```

```
## # A tibble: 4 x 2
##   FantPos     n
##   <chr>   <int>
## 1 ""         67
## 2 QB         57
## 3 RB         13
## 4 WR          1
```

```
# - Two_Point_Conv - a mix of positions, as expected
pro_fball_ref_ffb_2018_df[is.na(pro_fball_ref_ffb_2018_df$Two_Point_Conv),] %>% group_by(FantPos)
 %>% tally()
```

```
## # A tibble: 5 x 2
##   FantPos     n
##   <chr>   <int>
## 1 ""         67
## 2 QB         69
## 3 RB        139
## 4 TE        109
## 5 WR        183
```

```
# - Two_Point_Pass - no N/A's
pro_fball_ref_ffb_2018_df[is.na(pro_fball_ref_ffb_2018_df$Rush_Yds_per_Pass),] %>% group_by(FantPo
s) %>% tally()
```

```
## # A tibble: 0 x 2
## # ... with 2 variables: FantPos <chr>, n <int>
```

```
# - Value_Over_Baseline
# - Rank_Ovrl


#However, NA's can cause issues when analyzing data. As such, we'll replace these NA's with 0's.
#Blank values in these columns suggest the player did not accumulate stats in that area, thus the
 amount is 0. This is backed up by the original source dataset, where blanks and 0's both exist. F
or the most part, the blank values are usually for calculated fields that either calculate to 0 or
are divided by 0, whereas 0's are for recorded data.

# - Rush_Yds_per_Att
pro_fball_ref_ffb_2018_df$Rush_Yds_per_Att[is.na(pro_fball_ref_ffb_2018_df$Rush_Yds_per_Att)] <- 0
# - Rec_Yds_per_Rec
pro_fball_ref_ffb_2018_df$Rec_Yds_per_Rec[is.na(pro_fball_ref_ffb_2018_df$Rec_Yds_per_Rec)] <- 0
# - Two_Point_Conv
pro_fball_ref_ffb_2018_df$Two_Point_Conv[is.na(pro_fball_ref_ffb_2018_df$Two_Point_Conv)] <- 0
# - Two_Point_Pass
pro_fball_ref_ffb_2018_df$Two_Point_Pass[is.na(pro_fball_ref_ffb_2018_df$Two_Point_Pass)] <- 0


#We'll also look at where the fantasy points value is NA since that's what we're predicting, and i
f we should be concerned about a lack of value there. For clean output, we'll only include the fir
st 9 columns
pro_fball_ref_ffb_2018_df[is.na(pro_fball_ref_ffb_2018_df$Fant_Pts), 1:9]
```

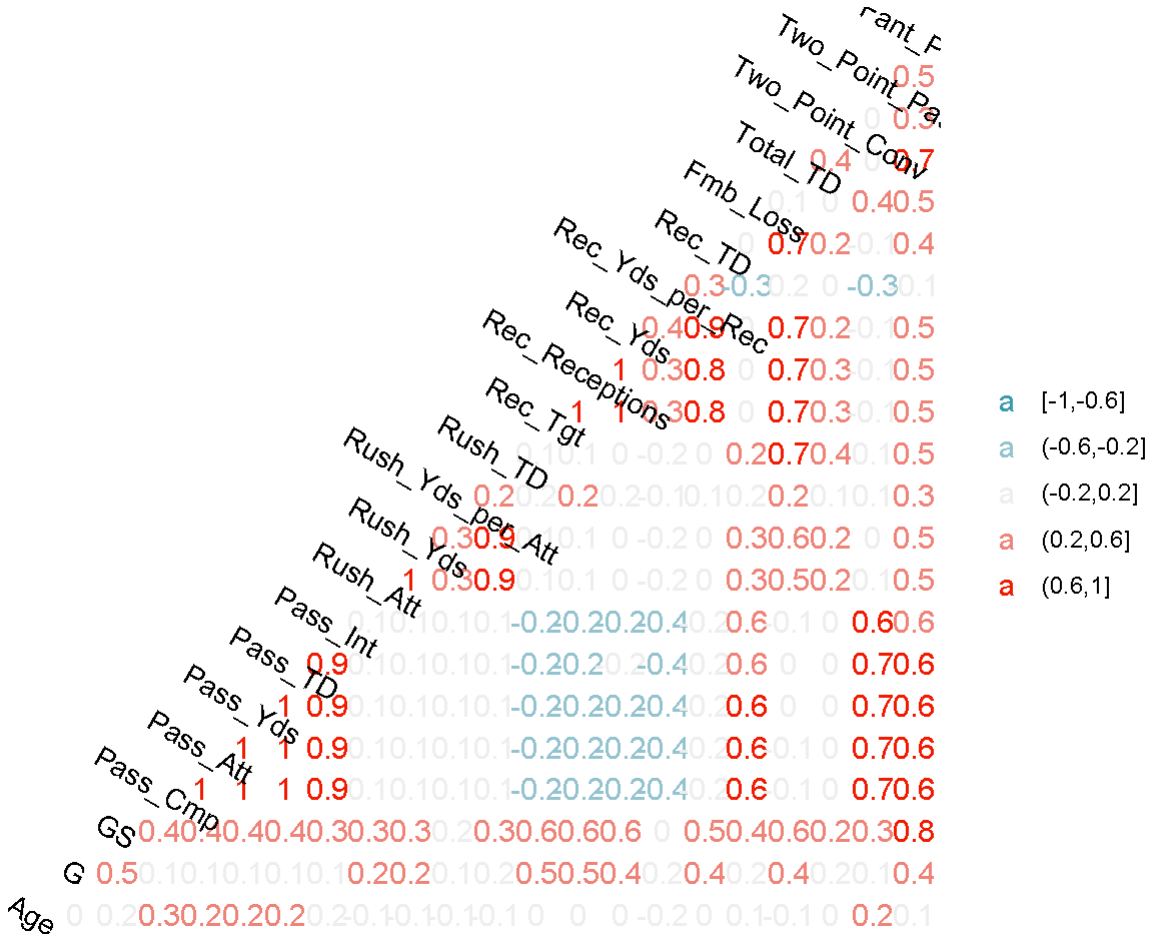
```
##         .id  Rk              Player  Tm FantPos Age  G GS Pass_Cmp
## 79  fantasy  77         Shane Smith NYG          25  2  0        0
## 555 fantasy 538     Saeed Blacknall OAK          22  1  0        0
## 556 fantasy 539        Daniel Brown CHI          26 14  0        0
## 558 fantasy 540       Deante Burton 2TM          24  3  0        0
## 559 fantasy 541        Jehu Chesson WAS      WR  25 12  0        0
## 560 fantasy 542       Lavon Coleman GNB          24  1  0        0
## 561 fantasy 543         Stacy Coley 2TM          24  3  0        0
## 562 fantasy 544       Pharoh Cooper 2TM          23  7  0        0
## 563 fantasy 545   Jerome Cunningham DET          27  3  0        0
## 564 fantasy 546     Darrell Daniels 2TM          24 11  3        0
## 565 fantasy 547        Geremy Davis LAC          26 14  0        0
## 566 fantasy 548        Trevor Davis GNB          25  2  0        0
## 567 fantasy 549       Dalyn Dawkins TEN          24  2  0        0
## 568 fantasy 550       Matthew Dayes SFO          24  7  0        0
## 569 fantasy 551   Garrett Dickerson NYG          23  4  0        0
## 570 fantasy 552      Quinton Dunbar WAS          26  7  6        0
## 571 fantasy 553       Steven Dunbar SFO          23  1  0        0
## 572 fantasy 554       Malachi Dupre ARI          23  1  0        0
## 573 fantasy 555        Trey Edmunds PIT          24  4  0        0
## 574 fantasy 556          Alex Ellis KAN          25  2  0        0
## 575 fantasy 557   Donnie Ernsberger TAM          22  2  0        0
## 576 fantasy 558         Isaiah Ford MIA          22  1  0        0
## 577 fantasy 559    Daurice Fountain IND          23  1  0        0
## 578 fantasy 560        Mose Frazier CAR          25  1  0        0
## 579 fantasy 561        Rico Gafford OAK          22  1  0        0
## 580 fantasy 562        C.J. Goodwin DAL          28  3  0        0
## 581 fantasy 563      Janarion Grant BAL          25  2  0        0
## 582 fantasy 564     Garrett Griffin NOR          24  0  0        0
## 583 fantasy 565        Clark Harris CIN          34 16  0        0
## 584 fantasy 566 Temarrick Hemingway DEN          25  5  0        0
## 585 fantasy 567 De'Angelo Henderson NYJ      RB  26  3  0        0
## 586 fantasy 568   Quadree Henderson NYG          22  5  0        0
## 587 fantasy 569        Hunter Henry LAC          24  0  0        0
## 589 fantasy 570 Darrius Heyward-Bey PIT      WR  31 14  2        0
## 590 fantasy 571         Gabe Holmes ARI          27  8  4        0
## 591 fantasy 572       Johnny Holton OAK          27  1  0        0
## 592 fantasy 573        Buddy Howell HOU          22  4  0        0
## 593 fantasy 574      Gregory Howell HOU          22 15  0        0
## 594 fantasy 575           Chad Kelly DEN      QB  24  1  0        0
## 595 fantasy 576          Ben Koyack JAX          25  7  2        0
## 596 fantasy 577          Chris Lacy DET          22  1  0        0
## 597 fantasy 578        Lance Lenoir DAL          23  7  0        0
## 598 fantasy 579        Tony Lippett NYG          26  3  0        0
## 599 fantasy 580        Sean Mannion LAR      QB  26  3  0        2
## 600 fantasy 581     Bradley Marquez DET          26  1  0        0
## 601 fantasy 582     Freddie Martino TAM          27  4  0        0
## 602 fantasy 583       A.J. McCarron OAK      QB  28  2  0        1
## 603 fantasy 584       Max McCaffrey SFO          24  1  0        0
## 604 fantasy 585       Tanner McEvoy MIA          25  2  0        0
## 605 fantasy 586      Jaydon Mickens JAX          24  6  0        0
## 606 fantasy 587     Steven Mitchell HOU          24  1  0        0
## 607 fantasy 588     Dare Ogunbowale TAM          24  2  0        0
```

```
## 608 fantasy 589      Senorise Perry MIA          27 16  0        0
## 609 fantasy 590        Thomas Rawls CIN          25  1  0        0
## 610 fantasy 591     Keenan Reynolds SEA          25  2  0        0
## 611 fantasy 592         Cooper Rush DAL          25  1  0        0
## 612 fantasy 593      Alonzo Russell NYG          26  1  0        0
## 613 fantasy 594        Mark Sanchez WAS      QB  32  2  1       19
## 614 fantasy 595         Matt Schaub ATL      QB  37  3  0        5
## 615 fantasy 596       Mason Schreck CIN          25  6  0        0
## 616 fantasy 597        Boston Scott PHI          23  2  0        0
## 617 fantasy 598       Da'Mari Scott BUF          23  3  0        0
## 618 fantasy 599            Cam Sims WAS          22  1  0        0
## 620 fantasy 600      Matthew Slater NWE          33 16  0        0
## 621 fantasy 601         Nate Solder NYG          30 16 16        0
## 622 fantasy 602        Brandon Tate NOR          31  1  0        0
## 623 fantasy 603         Mike Thomas LAR          24  1  0        0
## 624 fantasy 604        Danny Vitale GNB      RB  25  5  0        0
## 625 fantasy 605       Clive Walford NYJ          27  1  1        0
## 626 fantasy 606        Mike Wallace PHI          32  2  2        0
## 627 fantasy 607      Brandon Weeden HOU      QB  35  1  0        0
## 628 fantasy 608      Markus Wheaton PHI          27  1  0        0
## 629 fantasy 609           Cole Wick SFO          25  5  0        0
## 630 fantasy 610   Jonathan Williams NOR      RB  24  3  0        0
## 631 fantasy 611         Deon Yelder KAN          23  3  0        0
```

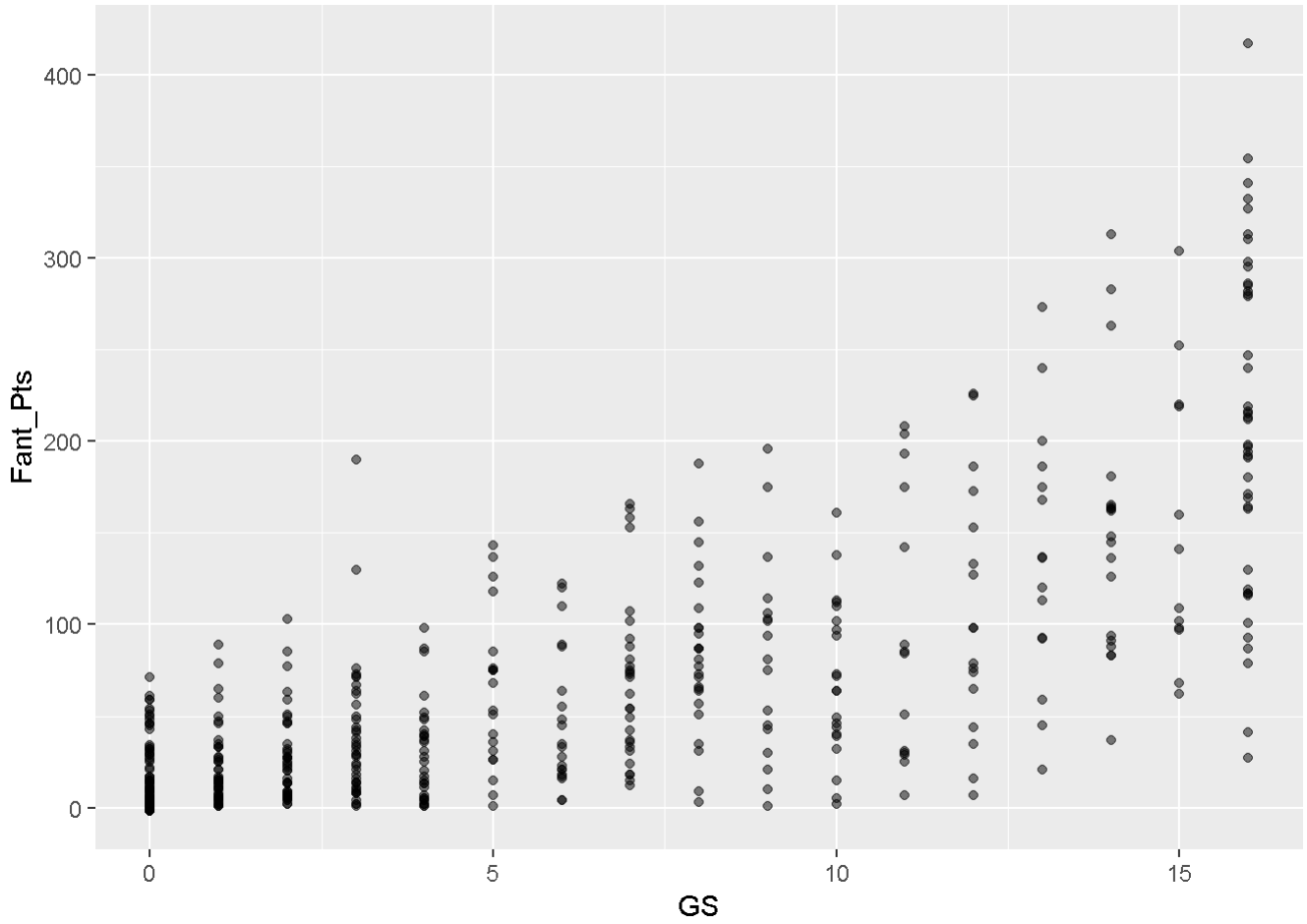
```
#looking through, the values where the fantasy points are NA look correct, as these are backup pla
yers who did not accumulate statistics during the season. So these can be removed. Since we need t
he fantasy points, we'll remove these null values from our dataframe.
pro_fball_ref_ffb_2018_df <- pro_fball_ref_ffb_2018_df[complete.cases(pro_fball_ref_ffb_2018_df[ ,
"Fant_Pts"]), ]
```

We'll also explore different correlations between different variables.

```
#create a matrix with just numeric values, and ignore other dependent variables that are not fanta
sy points as that's not what we're predicting
pro_fball_ref_ffb_2018_matrix <- as.matrix(pro_fball_ref_ffb_2018_df %>% select('Age', 'G', 'GS',
'Pass_Cmp', 'Pass_Att', 'Pass_Yds', 'Pass_TD', 'Pass_Int', 'Rush_Att', 'Rush_Yds', 'Rush_Yds_per_A
tt', 'Rush_TD', 'Rec_Tgt', 'Rec_Receptions', 'Rec_Yds', 'Rec_Yds_per_Rec', 'Rec_TD', 'Fmb_Loss',
'Total_TD', 'Two_Point_Conv', 'Two_Point_Pass', 'Fant_Pts' ))

#create a correlation matrix based on the above matrix
ggcorr(pro_fball_ref_ffb_2018_matrix, nbreaks = 5, geom = 'text', label_alpha = TRUE, angle = -30)
```




```
#note that while rushing and receiving yards per attempt and age seem to have little correlation w
ith Fant_Pts, all other variables have some correlation. This is not surprising, as fantasy points
are a direct function of some of these values (TD's, yards).

#while not that interesting, the largest correlation appears to be Games Started (GS) and Fantasy
 Points (Fant_Pts). We'll plot this association just to get an idea of how one of the strongest co
rrelations looks like. Note that GS has values between 0 and 16, so there's a limited number of va
lues GS can take,
pro_fball_ref_ffb_2018_df %>%
  ggplot(aes(GS, Fant_Pts)) +
  geom_point(alpha = .5)
```




```
#There are some seemingly surprising results, such as the high positive correlation between negati
ve plays (fumbles, interceptions), but this makes sense because the best players play the most, so
have accumulate these negative values as a nature of playing so much. We'll use linear regression
 to get a better understanding of what's needed soon.
```

The first model we'd like to build is a linear regression model to see the effects of the variables on overall points scored. Before doing that, we'll split the data into training and test sets. The training set will be used to build the model. The test set will be used to measure the model's quality.

```
#split the data into training and test set.
set.seed(1)
test_index1 <- createDataPartition(y = pro_fball_ref_ffb_2018_df$Fant_Pts, times = 1, p = 0.1, lis
t = FALSE)
train_pro_ffball <- pro_fball_ref_ffb_2018_df[-test_index1,]
test_pro_ffball <- pro_fball_ref_ffb_2018_df[test_index1,]
```

```
#linear regression on training data
Lin_Reg_2018 <- lm (Fant_Pts ~ Age + G + GS + Pass_Cmp + Pass_Att + Pass_Yds + Pass_TD + Pass_Int
 + Rush_Att + Rush_Yds + Rush_Yds_per_Att + Rush_TD + Rec_Tgt + Rec_Receptions + Rec_Yds + Rec_Yds
_per_Rec + Rec_TD + Fmb_Loss + Total_TD + Two_Point_Conv + Two_Point_Pass,
                    data = train_pro_ffball)

#we can check the results
summary(Lin_Reg_2018)
```

```
##
## Call:
## lm(formula = Fant_Pts ~ Age + G + GS + Pass_Cmp + Pass_Att +
##     Pass_Yds + Pass_TD + Pass_Int + Rush_Att + Rush_Yds + Rush_Yds_per_Att +
##     Rush_TD + Rec_Tgt + Rec_Receptions + Rec_Yds + Rec_Yds_per_Rec +
##     Rec_TD + Fmb_Loss + Total_TD + Two_Point_Conv + Two_Point_Pass,
##     data = train_pro_ffball)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.7411 -0.2514  0.0151  0.2543  1.8532
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)      -0.2587131  0.2213234  -1.169  0.24302
## Age               0.0121313  0.0077378   1.568  0.11760
## G                -0.0004418  0.0063746  -0.069  0.94477
## GS               -0.0030045  0.0081582  -0.368  0.71283
## Pass_Cmp          0.0193951  0.0064001   3.030  0.00258 **
## Pass_Att         -0.0019216  0.0046255  -0.415  0.67801
## Pass_Yds          0.0385106  0.0004983  77.278  < 2e-16 ***
## Pass_TD           4.0098485  0.0213788 187.562  < 2e-16 ***
## Pass_Int         -1.9995526  0.0268797 -74.389  < 2e-16 ***
## Rush_Att         -0.0039790  0.0026542  -1.499  0.13452
## Rush_Yds          0.1009500  0.0006166 163.720  < 2e-16 ***
## Rush_Yds_per_Att  0.0196920  0.0076389   2.578  0.01025 *
## Rush_TD           0.9914753  0.1326353   7.475 3.81e-13 ***
## Rec_Tgt          -0.0016715  0.0044132  -0.379  0.70504
## Rec_Receptions    0.0112490  0.0060196   1.869  0.06228 .
## Rec_Yds           0.0989205  0.0003798 260.437  < 2e-16 ***
## Rec_Yds_per_Rec  -0.0011712  0.0049233  -0.238  0.81207
## Rec_TD            1.0543792  0.1319033   7.994 1.03e-14 ***
## Fmb_Loss         -2.0733593  0.0341267 -60.755  < 2e-16 ***
## Total_TD          4.9910259  0.1297385  38.470  < 2e-16 ***
## Two_Point_Conv    2.0266193  0.0716110  28.300  < 2e-16 ***
## Two_Point_Pass    1.8931460  0.0751301  25.198  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5102 on 469 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:      1
## F-statistic: 4.855e+05 on 21 and 469 DF,  p-value: < 2.2e-16
```

From this, we see certain variables are more significant than others. We'll start off with a higher threshold, removing variables with a p-value greater than 0.15. And we'll re-run the model accordingly.

```
#linear regression on training data.
Lin_Reg_2018_2 <- lm (Fant_Pts ~ Age + Pass_Cmp + Pass_Yds + Pass_TD + Pass_Int + Rush_Att + Rush_
Yds + Rush_Yds_per_Att + Rush_TD + Rec_Receptions + Rec_Yds + Rec_TD + Fmb_Loss + Total_TD + Two_P
oint_Conv + Two_Point_Pass,
                    data = train_pro_ffball)

#we can check the results
summary(Lin_Reg_2018_2)
```

```
##
## Call:
## lm(formula = Fant_Pts ~ Age + Pass_Cmp + Pass_Yds + Pass_TD +
##     Pass_Int + Rush_Att + Rush_Yds + Rush_Yds_per_Att + Rush_TD +
##     Rec_Receptions + Rec_Yds + Rec_TD + Fmb_Loss + Total_TD +
##     Two_Point_Conv + Two_Point_Pass, data = train_pro_ffball)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.7192 -0.2490  0.0227  0.2527  1.8430
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)      -0.2789672  0.1981719  -1.408  0.15987
## Age               0.0120088  0.0074969   1.602  0.10986
## Pass_Cmp          0.0173476  0.0042326   4.099 4.89e-05 ***
## Pass_Yds          0.0384108  0.0004415  87.008  < 2e-16 ***
## Pass_TD           4.0141621  0.0187462 214.132  < 2e-16 ***
## Pass_Int         -2.0052881  0.0218533 -91.761  < 2e-16 ***
## Rush_Att         -0.0038338  0.0025979  -1.476  0.14067
## Rush_Yds          0.1008964  0.0005978 168.777  < 2e-16 ***
## Rush_Yds_per_Att  0.0200593  0.0074832   2.681  0.00761 **
## Rush_TD           0.9888417  0.1309047   7.554 2.19e-13 ***
## Rec_Receptions    0.0097841  0.0036623   2.672  0.00781 **
## Rec_Yds           0.0988014  0.0003222 306.658  < 2e-16 ***
## Rec_TD            1.0509628  0.1302003   8.072 5.75e-15 ***
## Fmb_Loss         -2.0716349  0.0338065 -61.279  < 2e-16 ***
## Total_TD          4.9941297  0.1279622  39.028  < 2e-16 ***
## Two_Point_Conv    2.0230160  0.0710269  28.482  < 2e-16 ***
## Two_Point_Pass    1.9009154  0.0726325  26.172  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5079 on 474 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:      1
## F-statistic: 6.431e+05 on 16 and 474 DF,  p-value: < 2.2e-16
```

From this, we see certain variables are more significant than others. We'll have a stricter threshold, removing variables with a p-value greater than 0.05. And we'll re-run the model accordingly.

```
#linear regression on training data.
Lin_Reg_2018_3 <- lm (Fant_Pts ~ Pass_Cmp + Pass_Yds + Pass_TD + Pass_Int + Rush_Yds + Rush_Yds_pe
r_Att + Rush_TD + Rec_Receptions + Rec_Yds + Rec_TD + Fmb_Loss + Total_TD + Two_Point_Conv + Two_P
oint_Pass,
                    data = train_pro_ffball)

#we can check the results
summary(Lin_Reg_2018_3)
```

```
##
## Call:
## lm(formula = Fant_Pts ~ Pass_Cmp + Pass_Yds + Pass_TD + Pass_Int +
##     Rush_Yds + Rush_Yds_per_Att + Rush_TD + Rec_Receptions +
##     Rec_Yds + Rec_TD + Fmb_Loss + Total_TD + Two_Point_Conv +
##     Two_Point_Pass, data = train_pro_ffball)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.7400 -0.2516  0.0195  0.2601  1.8874

##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)       0.0259527  0.0343896   0.755  0.45082
## Pass_Cmp          0.0177747  0.0042018   4.230 2.80e-05 ***
## Pass_Yds          0.0384166  0.0004411  87.099  < 2e-16 ***
## Pass_TD           4.0108997  0.0187177 214.284  < 2e-16 ***
## Pass_Int         -2.0083204  0.0215777 -93.074  < 2e-16 ***
## Rush_Yds          0.1000802  0.0002279 439.075  < 2e-16 ***
## Rush_Yds_per_Att  0.0219925  0.0073387   2.997  0.00287 **
## Rush_TD           0.9792426  0.1311020   7.469 3.88e-13 ***
## Rec_Receptions    0.0096200  0.0036510   2.635  0.00869 **
## Rec_Yds           0.0988187  0.0003220 306.921  < 2e-16 ***
## Rec_TD            1.0468200  0.1304421   8.025 7.98e-15 ***
## Fmb_Loss         -2.0830391  0.0334097 -62.348  < 2e-16 ***
## Total_TD          4.9994909  0.1282124  38.994  < 2e-16 ***
## Two_Point_Conv    2.0286189  0.0711348  28.518  < 2e-16 ***
## Two_Point_Pass    1.9104127  0.0725400  26.336  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.509 on 476 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:      1
## F-statistic: 7.316e+05 on 14 and 476 DF,  p-value: < 2.2e-16
```

```
tidy(Lin_Reg_2018_3, conf.int = TRUE)
```

```
## # A tibble: 15 x 7
##    term           estimate std.error statistic   p.value conf.low conf.high
##    <chr>             <dbl>     <dbl>     <dbl>     <dbl>    <dbl>     <dbl>
##  1 (Intercept)     0.0260   0.0344       0.755 4.51e-  1 -0.0416     0.0935
##  2 Pass_Cmp        0.0178   0.00420      4.23  2.80e-  5  0.00952    0.0260
##  3 Pass_Yds        0.0384   0.000441    87.1   1.29e-294  0.0375     0.0393
##  4 Pass_TD         4.01     0.0187     214.    0.         3.97       4.05
##  5 Pass_Int       -2.01     0.0216     -93.1   1.42e-307 -2.05      -1.97
##  6 Rush_Yds        0.100    0.000228   439.    0.         0.0996     0.101
##  7 Rush_Yds_per_~  0.0220   0.00734      3.00  2.87e-  3  0.00757    0.0364
##  8 Rush_TD         0.979    0.131        7.47  3.88e- 13  0.722      1.24
##  9 Rec_Receptions  0.00962  0.00365      2.63  8.69e-  3  0.00245    0.0168
## 10 Rec_Yds         0.0988   0.000322   307.    0.         0.0982     0.0995
## 11 Rec_TD          1.05     0.130        8.03  7.98e- 15  0.791      1.30
## 12 Fmb_Loss       -2.08     0.0334     -62.3   3.82e-231 -2.15      -2.02
## 13 Total_TD        5.00     0.128       39.0   2.67e-150  4.75       5.25
## 14 Two_Point_Conv  2.03     0.0711      28.5   4.69e-105  1.89       2.17
## 15 Two_Point_Pass  1.91     0.0725      26.3   5.68e- 95  1.77       2.05
```

This model has significant coefficients, and we'll use this model on our test set.

```
#RMSE can be used as a measure of prediction quality

#predict the test set data based on the model
y_hat_lin_reg <- predict(Lin_Reg_2018_3, test_pro_ffball)

#calculate RMSE as the difference between our predictions and the actual testset values
RMSE_lin_reg <- sqrt(mean((y_hat_lin_reg - test_pro_ffball$Fant_Pts)^2))
```

The resulting RMSE of this linear regression model is: 0.3658. This suggests our predicted points are very similar to the true points value.

To do some further analysis, we'd like to scale the data so that the effect of certain variables do not overwhelm values on other variables. For example, passing yards can be on the order of 10^4, while passing touchdowns would be on the order of 10^1, so we don't want a 1 unit increase in passing yards to be treated the same as a 1 unit increase in passing touchdowns.

```
#we're going to be selecting certain variables and analyzing the effects of those variables on pro
jected points. We have to scale the data to prevent outsized effects of certain variables. This ne
eds to be done on the continuous (i.e. not categorical) variables
scaled_ffball_2018 <- pro_fball_ref_ffb_2018_df %>% mutate_each_(funs(scale(.) %>% as.vector),
                        vars = c('Age', 'G', 'GS', 'Pass_Cmp', 'Pass_Att', 'Pass_Yds', 'Pass_TD',
'Pass_Int', 'Rush_Att', 'Rush_Yds', 'Rush_Yds_per_Att', 'Rush_TD', 'Rec_Tgt', 'Rec_Receptions', 'R
ec_Yds', 'Rec_Yds_per_Rec', 'Rec_TD', 'Fmb_Loss', 'Total_TD', 'Two_Point_Conv', 'Two_Point_Pass',
'Fant_Pts', 'PPR_Pts', 'DraftKing_Pts', 'FanDuel_Pts', 'Value_Over_Baseline', 'Rank_Pos', 'Rank_Ov
rl' ))
```

```
## Warning: mutate_each() is deprecated
## Please use mutate_if(), mutate_at(), or mutate_all() instead:
##
##   - To map `funs` over all variables, use mutate_all()
##   - To map `funs` over a selection of variables, use mutate_at()
## This warning is displayed once per session.
```

```
## Warning: funs() is soft deprecated as of dplyr 0.8.0
## please use list() instead
##
## # Before:
## funs(name = f(.)
##
## # After:
## list(name = ~f(.))
## This warning is displayed once per session.
```

```
head(scaled_ffball_2018)
```

```
##       .id Rk              Player  Tm FantPos        Age         G       GS
## 1 fantasy  1       Todd Gurley*+ LAR      RB -0.6312202 0.5458080 1.572650
## 2 fantasy  2     Saquon Barkley* NYG      RB -1.5173042 0.9689930 1.944968
## 3 fantasy  3 Christian McCaffrey CAR      RB -1.2219429 0.9689930 1.944968
## 4 fantasy  4       Alvin Kamara* NOR      RB -0.9265815 0.7574005 1.386491
## 5 fantasy  5   Patrick Mahomes*+ KAN      QB -0.9265815 0.9689930 1.944968
## 6 fantasy  6       Tyreek Hill*+ KAN      WR -0.6312202 0.9689930 1.944968
##     Pass_Cmp   Pass_Att   Pass_Yds     Pass_TD   Pass_Int    Rush_Att
## 1 -0.2728685 -0.2771236 -0.2715465 -0.25666624 -0.2831228  4.44510410
## 2 -0.2728685 -0.2771236 -0.2715465 -0.25666624 -0.2831228  4.54098460
## 3 -0.2598054 -0.2685151 -0.2143905 -0.09031975 -0.2831228  3.73558840
## 4 -0.2728685 -0.2771236 -0.2715465 -0.25666624 -0.2831228  3.25618591
## 5  4.7303024  4.7157910  5.5549384  8.06065794  4.1842276  0.68658851
## 6 -0.2728685 -0.2771236 -0.2715465 -0.25666624 -0.2831228 -0.04210328
##    Rush_Yds Rush_Yds_per_Att    Rush_TD    Rec_Tgt Rec_Receptions
## 1 4.8377721        0.6898852 7.80288051  1.3486926      1.5259664
## 2 5.0745577        0.7241630 4.91279431  2.4344351      2.8085227
## 3 4.1908399        0.7241630 2.98607018  2.5158658      3.4498008
## 4 3.2817522        0.5927648 6.35783741  2.0001381      2.4077238
## 5 0.6982517        0.5870518 0.57766501 -0.8499359     -0.8387466
## 6 0.1866256        1.2526125 0.09598397  2.8687321      2.6482031
##      Rec_Yds Rec_Yds_per_Rec     Rec_TD   Fmb_Loss   Total_TD
## 1  1.0979476      0.09099185  1.0500697  0.5285764  5.9447181
## 2  1.5505231     -0.22950425  1.0500697 -0.4859194  4.0306040
## 3  2.0191474     -0.19930043  1.9041784  0.5285764  3.3925660
## 4  1.5120061     -0.09023108  1.0500697 -0.4859194  4.9876610
## 5 -0.7637105     -1.55847235 -0.6581477  1.5430722 -0.1166431
## 6  3.9835177      1.29411070  4.4665045 -0.4859194  3.7115850
##   Two_Point_Conv Two_Point_Pass Fant_Pts  PPR_Pts DraftKing_Pts
## 1      7.4676492     -0.1839348 3.391284 3.263035      3.190608
## 2      2.2838876     -0.1839348 3.147818 3.417755      3.329295
## 3     -0.3079931     -0.1839348 2.931404 3.414367      3.336939
## 4      7.4676492     -0.1839348 2.850249 3.060882      2.984216
## 5      2.2838876     -0.1839348 4.797974 3.771239      3.823979
## 6     -0.3079931     -0.1839348 2.498576 2.832754      2.763628
##   FanDuel_Pts Value_Over_Baseline  Rank_Pos Rank_Ovrl
## 1    3.283396            2.982451 -1.436123 -1.697426
## 2    3.255333            2.531174 -1.417766 -1.652772
## 3    3.154062            2.151152 -1.399408 -1.608119
## 4    2.930778            2.032395 -1.381051 -1.563465
## 5    4.338809            1.937389 -1.436123 -1.518811
## 6    2.647707            1.866135 -1.436123 -1.474157
```

Now that we've scaled the data, we can use LASSO regression to select certain values based on a budget on the sum of the coefficients. LASSO is a global optimization variable selection method, that keeps the most important coefficients and drops the less important coefficients to 0 (thus removing the variable from the model). There is a more generalized version of LASSO (Elastic Net), which we'll explore here too.

```
#prepare data to be used in glmnet, by creating a predictors matrix holding the numeric variables
 and a response matrix for the fantasy poitns
ffball_predictors <- as.matrix(scaled_ffball_2018[, 6:26])
ffball_response_fantpts <- as.matrix(scaled_ffball_2018[, 27]) %>% `colnames<-`('Fant_Pts')


set.seed(1)

#we can use elastic net to tune alpha. The closer alpha is to 1, the more it behaves like lasso re
gression, which tends to be better for picking variables. The closer alpha is to 1, the more it be
haves like ridge regression, which tends to be better for minimizing prediction error. We'll use R
^2 as the measuure of quality for each iteration.

######################

#run for alpha = 0
elastic_net_glm_0 <- cv.glmnet(x = ffball_predictors, y = ffball_response_fantpts, family = "gauss
ian", nfolds = 10, alpha = 0)
small_lambda_index_0 <- which(elastic_net_glm_0$lambda == elastic_net_glm_0$lambda.min)
small_lambda_betas_0 <- elastic_net_glm_0$glmnet.fit$beta[, small_lambda_index_0]


#calculate the r-squared
r2 <- elastic_net_glm_0$glmnet.fit$dev.ratio[which(elastic_net_glm_0$glmnet.fit$lambda == elastic_
net_glm_0$lambda.min)]

#add results to a dataframe
elastic_net_results <- tibble(Alpha = '0', Rsquared = r2)


######################

#run for alpha .25
elastic_net_glm_0.25 <- cv.glmnet(x = ffball_predictors, y = ffball_response_fantpts, family = "ga
ussian", nfolds = 10, alpha = 0.25)
small_lambda_index_0.25 <- which(elastic_net_glm_0.25$lambda == elastic_net_glm_0.25$lambda.min)
small_lambda_betas_0.25 <- elastic_net_glm_0.25$glmnet.fit$beta[, small_lambda_index_0.25]

#calculate the r-squared
r2_0.25 <- elastic_net_glm_0.25$glmnet.fit$dev.ratio[which(elastic_net_glm_0.25$glmnet.fit$lambda
 == elastic_net_glm_0.25$lambda.min)]

#create an entry in the R^2 table for this method
elastic_net_results <- bind_rows(elastic_net_results,
                          tibble(Alpha = '0.25', Rsquared = r2_0.25))

elastic_net_results %>% knitr::kable()
```

Alpha	Rsquared
0	0.9939809
0.25	0.9989241

```
######################

#run for alpha .5
elastic_net_glm_0.5 <- cv.glmnet(x = ffball_predictors, y = ffball_response_fantpts, family = "gau
ssian", nfolds = 10, alpha = 0.5)
small_lambda_index_0.5 <- which(elastic_net_glm_0.5$lambda == elastic_net_glm_0.5$lambda.min)
small_lambda_betas_0.5 <- elastic_net_glm_0.5$glmnet.fit$beta[, small_lambda_index_0.5]


#calculate the r-squared
r2_0.5 <- elastic_net_glm_0.5$glmnet.fit$dev.ratio[which(elastic_net_glm_0.5$glmnet.fit$lambda ==
 elastic_net_glm_0.5$lambda.min)]


#add results to table
elastic_net_results <- bind_rows(elastic_net_results,
                          tibble(Alpha = '0.5', Rsquared = r2_0.5))

elastic_net_results %>% knitr::kable()
```

Alpha	Rsquared
0	0.9939809
0.25	0.9989241
0.5	0.9989257

```
######################

#run for alpha .75
elastic_net_glm_0.75 <- cv.glmnet(x = ffball_predictors, y = ffball_response_fantpts, family = "ga
ussian", nfolds = 10, alpha = 0.75)
small_lambda_index_0.75 <- which(elastic_net_glm_0.75$lambda == elastic_net_glm_0.75$lambda.min)
small_lambda_betas_0.75 <- elastic_net_glm_0.75$glmnet.fit$beta[, small_lambda_index_0.75]


#calculate the r-squared
r2_0.75 <- elastic_net_glm_0.75$glmnet.fit$dev.ratio[which(elastic_net_glm_0.75$glmnet.fit$lambda
 == elastic_net_glm_0.75$lambda.min)]


#add results to table
elastic_net_results <- bind_rows(elastic_net_results,
                          tibble(Alpha = '0.75', Rsquared = r2_0.75))

elastic_net_results %>% knitr::kable()
```

Alpha	Rsquared
0	0.9939809

Alpha	Rsquared
0.25	0.9989241
0.5	0.9989257
0.75	0.9988479

```
####################

#run for alpha 1
elastic_net_glm_1 <- cv.glmnet(x = ffball_predictors, y = ffball_response_fantpts, family = "gauss
ian", nfolds = 10, alpha = 1)
small_lambda_index_1 <- which(elastic_net_glm_1$lambda == elastic_net_glm_1$lambda.min)
small_lambda_betas_1 <- elastic_net_glm_1$glmnet.fit$beta[, small_lambda_index_1]


#calculate the r-squared
r2_1 <- elastic_net_glm_1$glmnet.fit$dev.ratio[which(elastic_net_glm_1$glmnet.fit$lambda == elasti
c_net_glm_1$lambda.min)]


#add results to table
elastic_net_results <- bind_rows(elastic_net_results,
                         tibble(Alpha = '1', Rsquared = r2_1))

elastic_net_results %>% knitr::kable()
```

Alpha	Rsquared
0	0.9939809
0.25	0.9989241
0.5	0.9989257
0.75	0.9988479
1	0.9987575

```
##############
#All the R-squareds are close to each other. But from this, we see the best R-squared is when alph
a = 0.5. We'll output this model's coefficients:
small_lambda_betas_0.5
```

```
##              Age                G               GS         Pass_Cmp
##     0.000000e+00     0.000000e+00     9.679100e-03     1.051799e-01
##         Pass_Att         Pass_Yds          Pass_TD         Pass_Int
##     2.083075e-03     2.513130e-01     3.641936e-01    -1.198885e-02
##         Rush_Att         Rush_Yds Rush_Yds_per_Att          Rush_TD
##     2.649128e-02     2.708252e-01     5.989227e-05     1.052280e-03
##          Rec_Tgt   Rec_Receptions          Rec_Yds  Rec_Yds_per_Rec
##     2.179613e-03     3.550388e-02     3.639022e-01     2.491381e-04
##           Rec_TD         Fmb_Loss         Total_TD   Two_Point_Conv
##     0.000000e+00    -9.184916e-03     2.664913e-01     9.596486e-03
##   Two_Point_Pass
##     1.151000e-02
```

From this, we see the best R^2 value is when alpha = 0.5. This is a global variable selection approach which selects which variables are most useful, subject to a constraint of how much can be allocated to each variable. Values with 0 can definitely be removed from the model. And values with values very close to 0 are practically not significant for scaled data, so can also be ignored. We'll make |0.1| the threshold. That leaves these variables:
- Pass_Cmp
- Pass_Yds
- Pass_TD
- Rush_Yds
- Rec_Yds
- Total_TD

Now we'll also run linear regression with above variables to determine effect of each in that model.

```
#split the data into training and test set. Training set used to build the model and test set is t
o assess accuracy.

#Now we will use our dataset to split into training data (90%) and (initial) validation data (1
0%)...using the scaled data
set.seed(1)
test_index_scaled <- createDataPartition(y = scaled_ffball_2018$Fant_Pts, times = 1, p = 0.1, list
= FALSE)
train_pro_ffball_scaled <- scaled_ffball_2018[-test_index_scaled,]
test_pro_ffball_scaled <- scaled_ffball_2018[test_index_scaled,]
```

```
#predict using linear regression, calculate RMSE
fit_2018_scaled <- lm(Fant_Pts ~ Pass_Cmp + Pass_Yds + Pass_TD + Rush_Yds + Rec_Yds + Total_TD, da
ta = train_pro_ffball_scaled)

summary(fit_2018_scaled)
```

```
##
## Call:
## lm(formula = Fant_Pts ~ Pass_Cmp + Pass_Yds + Pass_TD + Rush_Yds +
##     Rec_Yds + Total_TD, data = train_pro_ffball_scaled)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.24214 -0.00421  0.00478  0.01054  0.37377
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.482e-05  1.729e-03  -0.009    0.993
## Pass_Cmp    -2.420e-02  2.390e-02  -1.013    0.312
## Pass_Yds     3.791e-01  2.836e-02  13.369   <2e-16 ***
## Pass_TD      3.695e-01  8.140e-03  45.390   <2e-16 ***
## Rush_Yds     3.103e-01  2.563e-03 121.073   <2e-16 ***
## Rec_Yds      4.156e-01  2.826e-03 147.055   <2e-16 ***
## Total_TD     2.634e-01  3.482e-03  75.635   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.03829 on 484 degrees of freedom
## Multiple R-squared:  0.9985, Adjusted R-squared:  0.9985
## F-statistic: 5.512e+04 on 6 and 484 DF,  p-value: < 2.2e-16
```

```
#use the fitted model to predict the test data and calculate the RMSE.
y_hat_scaled <- predict(fit_2018_scaled, test_pro_ffball_scaled)
RMSE_lin_reg_lasso_scaled <- sqrt(mean((y_hat_scaled - test_pro_ffball_scaled$Fant_Pts)^2))


#now use the unscaled data to give us a linear model
fit_2018 <- lm(Fant_Pts ~ Pass_Cmp + Pass_Yds + Pass_TD + Rush_Yds + Rec_Yds + Total_TD, data = tr
ain_pro_ffball)
summary(fit_2018)
```

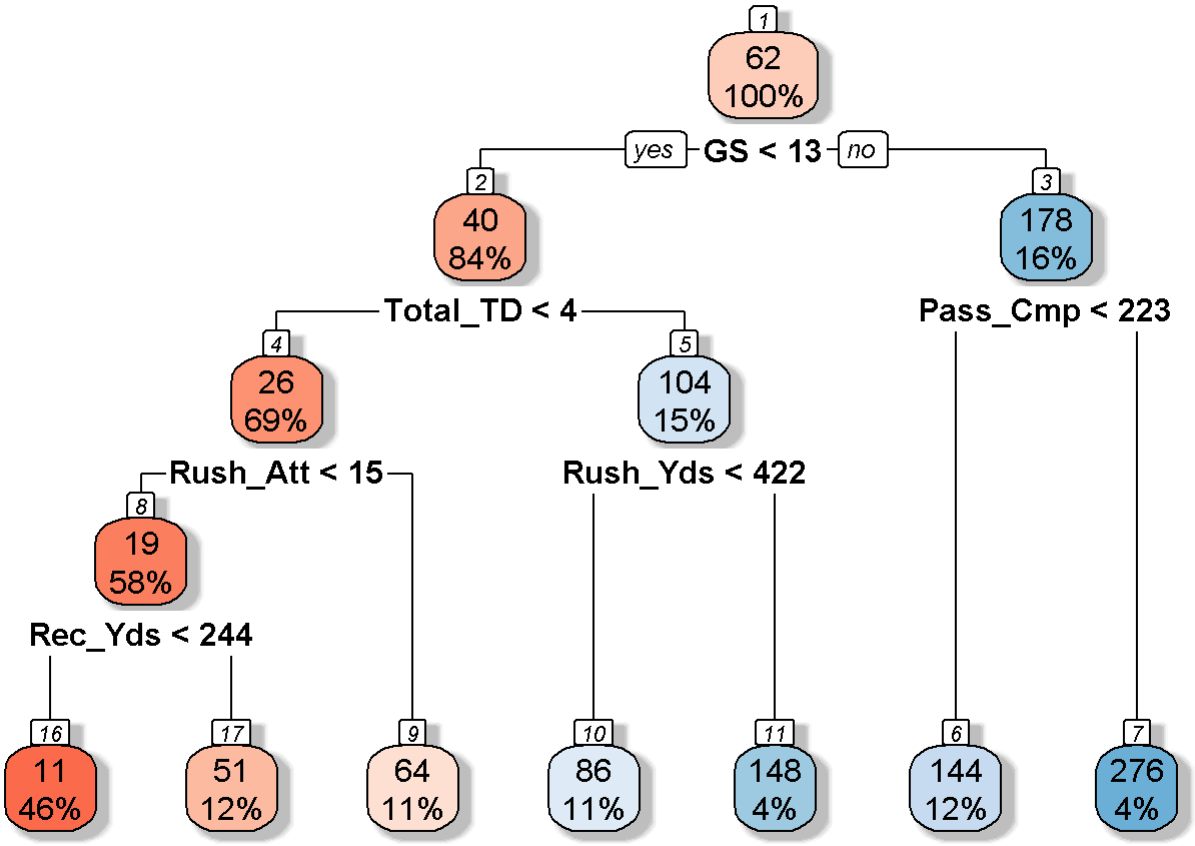
```
##
## Call:
## lm(formula = Fant_Pts ~ Pass_Cmp + Pass_Yds + Pass_TD + Rush_Yds +
##     Rec_Yds + Total_TD, data = train_pro_ffball)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -17.9020  -0.3115   0.3537   0.7793  27.6337
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.3899815  0.1784851  -2.185   0.0294 *
## Pass_Cmp    -0.0233736  0.0230790  -1.013   0.3117
## Pass_Yds     0.0320427  0.0023969  13.369   <2e-16 ***
## Pass_TD      4.5438759  0.1001079  45.390   <2e-16 ***
## Rush_Yds     0.0970166  0.0008013 121.073   <2e-16 ***
## Rec_Yds      0.0986262  0.0006707 147.055   <2e-16 ***
## Total_TD     6.2117159  0.0821274  75.635   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.831 on 484 degrees of freedom
## Multiple R-squared:  0.9985, Adjusted R-squared:  0.9985
## F-statistic: 5.512e+04 on 6 and 484 DF,  p-value: < 2.2e-16
```

```
#use the fitted model to predict the test data and calculate the RMSE.
y_hat <- predict(fit_2018, test_pro_ffball)
RMSE_lin_reg_lasso <- sqrt(mean((y_hat - test_pro_ffball$Fant_Pts)^2))

#and we see the RMSE is 2.557708 on the unscaled data, suggesting it's predicted results  are off
 by 2.5 points.
```

From this, we see that Passing Yards and Touchdowns, Rushing Yards, Receiving Yards, and Total Touchdowns have the largest influence on overall fantasy points. This model's RMSE is 2.56 on the unscaled data, meaning it's predicting values with a small amount of error.

Regression tree - for continuous data:
- Use decision tree on fantasy football data to plot a decision tree, which continues to split until we get to k partitions.It stops at k because that's where the RSS is only marginally increasing, according to the complexity parameter.
- A minimum number of observations to be partitioned is part of the minsplit argument in rpart (defaulted to 20).
- A minimum number of observations in each partition is part of the minbucket argument in rpart. If a bucket is smaller than this, it won't create a new partition (defaulted to round(minsplit/3)).

```
#fit the tree using rpart...after several iterations, minsplit = 60 was used
fit_tree <- rpart(Fant_Pts ~ Age + G + GS + Pass_Cmp + Pass_Att + Pass_Yds + Pass_TD + Pass_Int +
 Rush_Att + Rush_Yds + Rush_Yds_per_Att + Rush_TD + Rec_Tgt + Rec_Receptions + Rec_Yds + Rec_Yds_p
er_Rec + Rec_TD + Fmb_Loss + Total_TD + Two_Point_Conv + Two_Point_Pass,
                  data = train_pro_ffball,
                  minsplit = 60)

#visualize the  tree
rpart.plot(fit_tree, type = 2, extra = "auto", box.palette="RdBu", shadow.col="gray", nn=TRUE)
```




```
#output the complexity parameter info
printcp(fit_tree)
```

```
##
## Regression tree:
## rpart(formula = Fant_Pts ~ Age + G + GS + Pass_Cmp + Pass_Att +
##     Pass_Yds + Pass_TD + Pass_Int + Rush_Att + Rush_Yds + Rush_Yds_per_Att +
##     Rush_TD + Rec_Tgt + Rec_Receptions + Rec_Yds + Rec_Yds_per_Rec +
##     Rec_TD + Fmb_Loss + Total_TD + Two_Point_Conv + Two_Point_Pass,
##     data = train_pro_ffball, minsplit = 60)
##
## Variables actually used in tree construction:
## [1] GS       Pass_Cmp Rec_Yds  Rush_Att Rush_Yds Total_TD
##
## Root node error: 2654286/491 = 5405.9
##
## n= 491
##
##         CP nsplit rel error  xerror     xstd
## 1 0.469708      0   1.00000 1.00523 0.102063
## 2 0.139402      1   0.53029 0.57875 0.052750
## 3 0.097781      2   0.39089 0.41855 0.039435
## 4 0.035304      3   0.29311 0.33458 0.035990
## 5 0.028609      4   0.25780 0.32108 0.034071
## 6 0.022109      5   0.22920 0.30470 0.034830
## 7 0.010000      6   0.20709 0.28358 0.034393
```

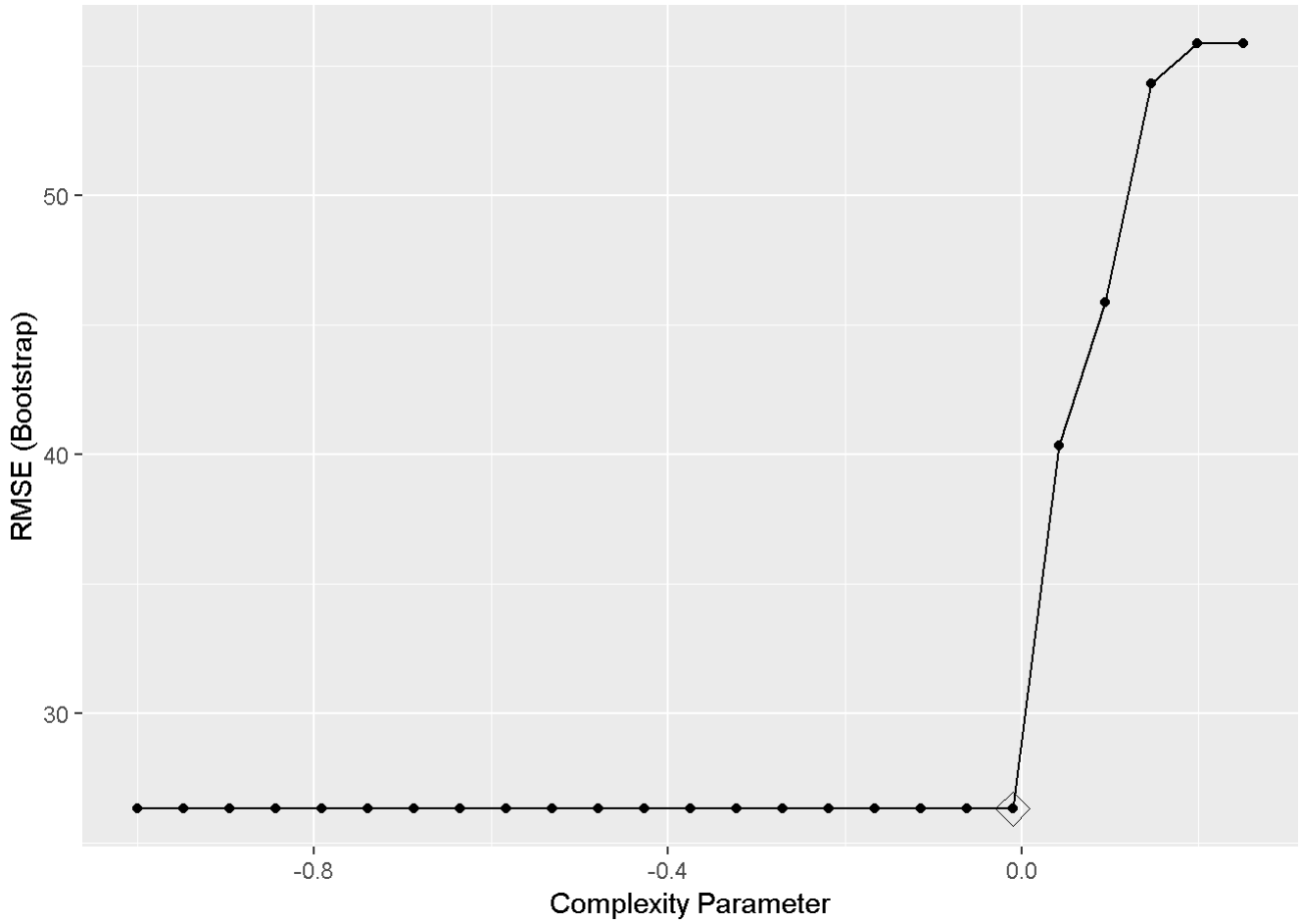
The complexity parameter is the amount by which splitting that node improved the relative error. So in this example, splitting the original root node dropped the relative error from 1.0 to 0.53029, so the CP of the root node is 0.469708. The smaller the cp, the bigger the tree.

We'll use this information as a starting point to choose an optimal model. We want to choose the optimal cp.

```
#How to choose optimal CP? Train several values of cp, see which one minimizes the error. Note tha
t the tuning parameter for train's rpart method is the Complexity Parameter, as per http://topepo.
github.io/caret/available-models.html
fit_rpart <- train(Fant_Pts ~ Age + G + GS + Pass_Cmp + Pass_Att + Pass_Yds + Pass_TD + Pass_Int +
Rush_Att + Rush_Yds + Rush_Yds_per_Att + Rush_TD + Rec_Tgt + Rec_Receptions + Rec_Yds + Rec_Yds_pe
r_Rec + Rec_TD + Fmb_Loss + Total_TD + Two_Point_Conv + Two_Point_Pass,
                     method = 'rpart',
                     tuneGrid = data.frame(cp = seq(-1, .25, len = 25)),
                     data = train_pro_ffball)


#see the best value of CP
fit_rpart$bestTune
```

```
##             cp
## 20 -0.01041667
```

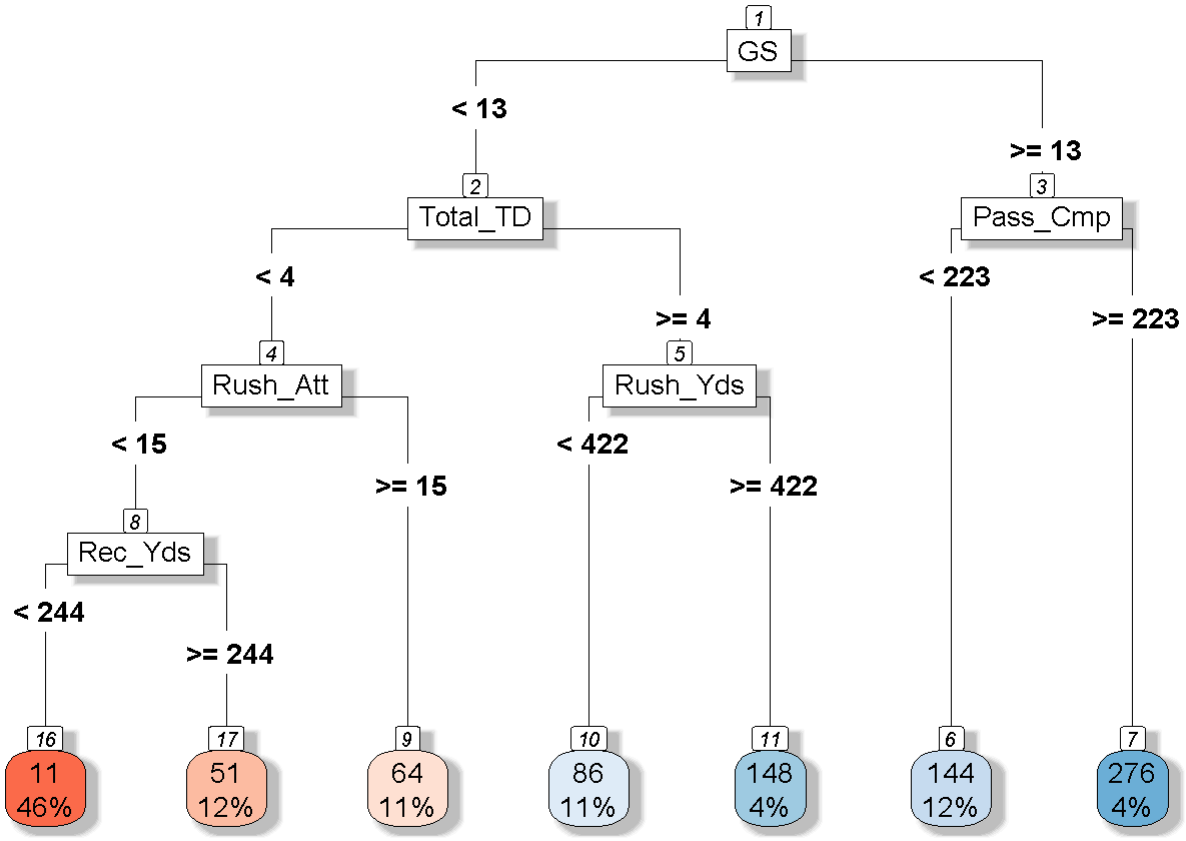
```
#plot the complexity parameter vs the RMSE. This aligns with the above results about the optimal c
omplexity parameter that minimizes the RMSE
ggplot(fit_rpart, highlight = TRUE)
```




```
#find the smallest RMSE
RMSE_reg_tree_min <- fit_rpart$results$RMSE[which.min(fit_rpart$results$RMSE)]
```

The cp that minimizes the RMSE is -0.0104. Use that cp when pruning the tree.

```
#prune fit
pruned_fit <- prune.rpart(fit_tree, cp = -0.0104)

#visualize the pruned fit
rpart.plot(pruned_fit, type = 5, extra = "auto", box.palette="RdBu", shadow.col="gray", nn=TRUE)
```




```
#calculate RMSE of pruned fit
y_hat_pruned <- predict(pruned_fit, test_pro_ffball)
RMSE_reg_tree_pruned <- sqrt(mean(y_hat_pruned - test_pro_ffball$Fant_Pts)^2)
```

The above pruned tree gives a view of how different variables impact the prediction of fantasy points. From this, we see GS, Total_TD, Rush_Att, Rec_Yds, Rush_Yds, and Pass_Cmp influence the prediction of fantasy points, with different thresholds leading to different decisions.

From above, the root node was Games Started. This makes sense that it's a key value, because the more games started, the more points a player can accumulate. We want to do recaculate based on Fantasy Points per game. But we also don't want to overreact to players who have short-term success in a few games in the season. So we want to look at players who start at least half the games and play at least 3/4th the games.

```
#recreate the dataset with a new column displaying fantasy points per game, removing entries with
 0 games started so the calculation can be done and only including players who played at least 12
 games, to avoid players who had only a couple quality games in a small sample size
train_pro_ffball_ppg <- train_pro_ffball %>%
  filter(GS >= 8 & G >= 12) %>%
  mutate(Fant_PPG = Fant_Pts/G)

test_pro_ffball_ppg <- test_pro_ffball %>%
  filter(GS >= 8 & G >= 12) %>%
  mutate(Fant_PPG = Fant_Pts/G)


summary(train_pro_ffball_ppg)
```

```
##      .id                 Rk               Player
##  Length:145         Length:145         Length:145
##  Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character
##
##
##
##
##       Tm              FantPos               Age              G
##  Length:145         Length:145         Min.   :21.00   Min.   :12.00
##  Class :character   Class :character   1st Qu.:24.00   1st Qu.:14.00
##  Mode  :character   Mode  :character   Median :26.00   Median :16.00
##                                        Mean   :26.57   Mean   :14.99
##                                        3rd Qu.:29.00   3rd Qu.:16.00
##                                        Max.   :39.00   Max.   :16.00
##
##        GS            Pass_Cmp         Pass_Att         Pass_Yds
##  Min.   : 8.00   Min.   :  0.00   Min.   :  0.00   Min.   :   0.0
##  1st Qu.:10.00   1st Qu.:  0.00   1st Qu.:  0.00   1st Qu.:   0.0
##  Median :13.00   Median :  0.00   Median :  0.00   Median :   0.0
##  Mean   :12.61   Mean   : 53.21   Mean   : 81.08   Mean   : 606.5
##  3rd Qu.:16.00   3rd Qu.:  0.00   3rd Qu.:  1.00   3rd Qu.:   0.0
##  Max.   :16.00   Max.   :452.00   Max.   :675.00   Max.   :5129.0
##
##     Pass_TD          Pass_Int         Rush_Att         Rush_Yds
##  Min.   : 0.000   Min.   : 0.000   Min.   :  0.00   Min.   :  -8.0
##  1st Qu.: 0.000   1st Qu.: 0.000   1st Qu.:  0.00   1st Qu.:   0.0
##  Median : 0.000   Median : 0.000   Median :  2.00   Median :  11.0
##  Mean   : 4.062   Mean   : 1.738   Mean   : 43.72   Mean   : 201.4
##  3rd Qu.: 0.000   3rd Qu.: 0.000   3rd Qu.: 43.00   3rd Qu.: 151.0
##  Max.   :50.000   Max.   :16.000   Max.   :304.00   Max.   :1434.0
##
##  Rush_Yds_per_Att    Rush_TD          Rec_Tgt      Rec_Receptions
##  Min.   :-7.000   Min.   : 0.000   Min.   :  0.0   Min.   :  0.00
##  1st Qu.: 0.000   1st Qu.: 0.000   1st Qu.: 18.0   1st Qu.: 13.00
##  Median : 2.800   Median : 0.000   Median : 64.0   Median : 37.00
##  Mean   : 3.109   Mean   : 1.614   Mean   : 60.9   Mean   : 40.32
##  3rd Qu.: 5.310   3rd Qu.: 2.000   3rd Qu.: 95.0   3rd Qu.: 64.00
##  Max.   :14.500   Max.   :17.000   Max.   :170.0   Max.   :125.00
##
##     Rec_Yds       Rec_Yds_per_Rec       Rec_TD          Fmb_Loss
##  Min.   : -11.0   Min.   :-11.000   Min.   : 0.000   Min.   :0.0000
##  1st Qu.:  99.0   1st Qu.:  7.250   1st Qu.: 0.000   1st Qu.:0.0000
##  Median : 402.0   Median : 10.500   Median : 2.000   Median :0.0000
##  Mean   : 482.5   Mean   :  9.595   Mean   : 2.979   Mean   :0.9172
##  3rd Qu.: 736.0   3rd Qu.: 12.920   3rd Qu.: 5.000   3rd Qu.:1.0000
##  Max.   :1677.0   Max.   : 24.000   Max.   :13.000   Max.   :7.0000
##
##     Total_TD      Two_Point_Conv   Two_Point_Pass      Fant_Pts
##  Min.   : 0.000   Min.   :0.0000   Min.   :0.0000   Min.   :  1.0
##  1st Qu.: 2.000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.: 68.0
##  Median : 4.000   Median :0.0000   Median :0.0000   Median :113.0
##  Mean   : 4.621   Mean   :0.2276   Mean   :0.2552   Mean   :132.2
##  3rd Qu.: 6.000   3rd Qu.:0.0000   3rd Qu.:0.0000   3rd Qu.:186.0
```

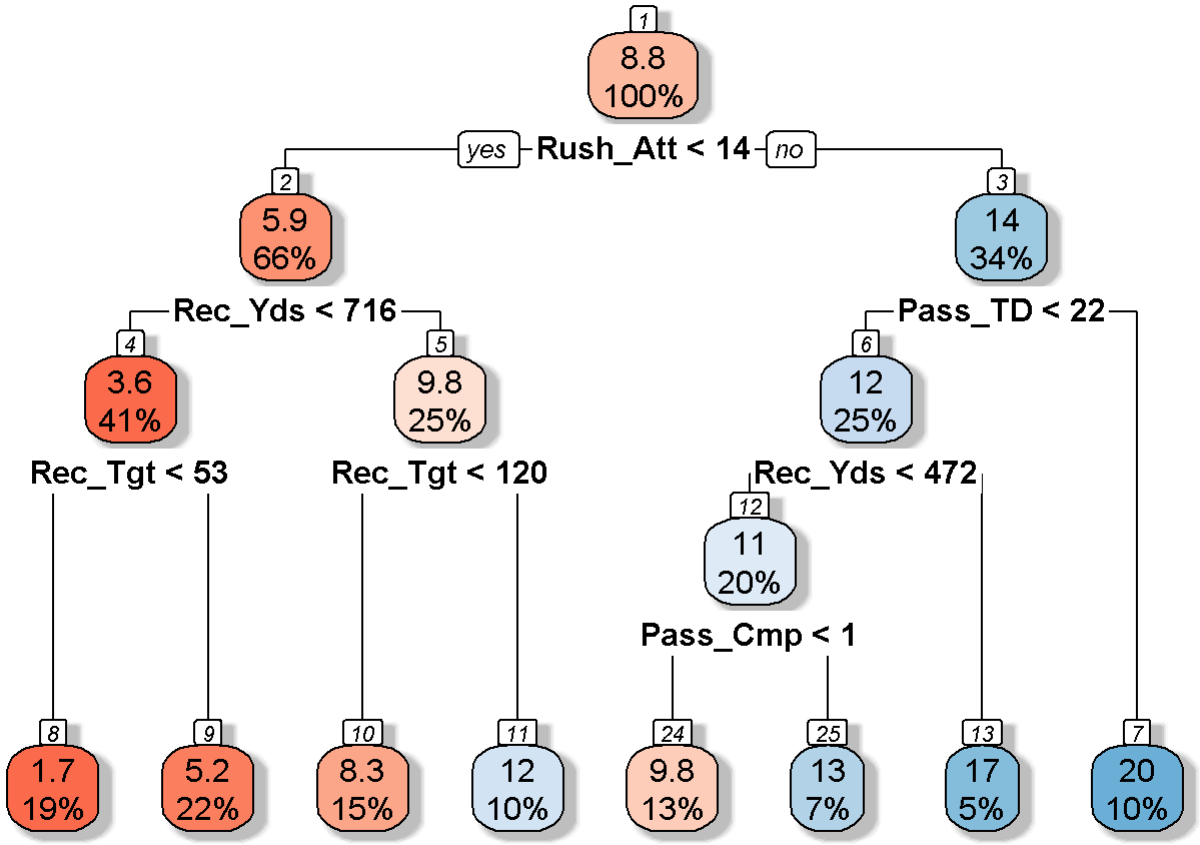
```
##  3rd Qu.: 0.000   3rd Qu.:0.0000   3rd Qu.:0.0000   3rd Qu.:180.0
##  Max.   :21.000   Max.   :3.0000   Max.   :5.0000   Max.   :417.0
##
##     PPR_Pts      DraftKing_Pts    FanDuel_Pts    Value_Over_Baseline
##  Min.   :  3.4   Min.   :  3.4   Min.   :  2.6   Min.   :  1.00
##  1st Qu.:104.9   1st Qu.:108.6   1st Qu.: 90.6   1st Qu.: 24.00
##  Median :165.2   Median :168.4   Median :137.2   Median : 47.00
##  Mean   :172.5   Mean   :178.9   Mean   :154.2   Mean   : 54.86
##  3rd Qu.:240.0   3rd Qu.:246.2   3rd Qu.:220.1   3rd Qu.: 78.00
##  Max.   :417.1   Max.   :437.1   Max.   :429.1   Max.   :178.00
##                                                  NA's   :88
##     Rank_Pos        Rank_Ovrl        Fant_PPG
##  Min.   :  1.00   Min.   : 1.00   Min.   : 0.08333
##  1st Qu.: 12.00   1st Qu.:19.00   1st Qu.: 4.53333
##  Median : 25.00   Median :37.00   Median : 7.43750
##  Mean   : 34.03   Mean   :37.84   Mean   : 8.79708
##  3rd Qu.: 49.00   3rd Qu.:57.00   3rd Qu.:12.37500
##  Max.   :145.00   Max.   :78.00   Max.   :26.06250
##                   NA's   :84
```

```
summary(test_pro_ffball_ppg)
```

```
##      .id                 Rk               Player
##  Length:14          Length:14          Length:14
##  Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character
##
##
##
##
##       Tm              FantPos               Age              G
##  Length:14          Length:14          Min.   :23.00   Min.   :12.00
##  Class :character   Class :character   1st Qu.:24.25   1st Qu.:15.25
##  Mode  :character   Mode  :character   Median :26.50   Median :16.00
##                                        Mean   :28.00   Mean   :15.29
##                                        3rd Qu.:30.00   3rd Qu.:16.00
##                                        Max.   :41.00   Max.   :16.00
##
##        GS            Pass_Cmp         Pass_Att         Pass_Yds
##  Min.   :10.00   Min.   :  0.00   Min.   :  0.00   Min.   :   0.00
##  1st Qu.:12.25   1st Qu.:  0.00   1st Qu.:  0.00   1st Qu.:   0.00
##  Median :13.50   Median :  0.00   Median :  0.00   Median :   0.00
##  Mean   :13.79   Mean   : 51.57   Mean   : 77.14   Mean   : 613.43
##  3rd Qu.:16.00   3rd Qu.:  0.75   3rd Qu.:  1.75   3rd Qu.:   3.75
##  Max.   :16.00   Max.   :375.00   Max.   :570.00   Max.   :4355.00
##
##     Pass_TD      Pass_Int         Rush_Att         Rush_Yds
##  Min.   : 0   Min.   : 0.000   Min.   :  0.00   Min.   :  0.0
##  1st Qu.: 0   1st Qu.: 0.000   1st Qu.:  1.00   1st Qu.:  0.5
##  Median : 0   Median : 0.000   Median :  4.50   Median : 32.0
##  Mean   : 4   Mean   : 1.429   Mean   : 53.14   Mean   :241.1
##  3rd Qu.: 0   3rd Qu.: 0.000   3rd Qu.: 80.00   3rd Qu.:428.2
##  Max.   :29   Max.   :11.000   Max.   :234.00   Max.   :885.0
##
##  Rush_Yds_per_Att    Rush_TD          Rec_Tgt       Rec_Receptions
##  Min.   : 0.000   Min.   : 0.000   Min.   :  0.00   Min.   :  0.00
##  1st Qu.: 0.380   1st Qu.: 0.000   1st Qu.: 33.50   1st Qu.: 21.50
##  Median : 4.135   Median : 0.500   Median : 70.00   Median : 45.50
##  Mean   : 4.967   Mean   : 2.714   Mean   : 69.36   Mean   : 44.71
##  3rd Qu.: 5.442   3rd Qu.: 4.250   3rd Qu.: 92.75   3rd Qu.: 63.25
##  Max.   :20.000   Max.   :14.000   Max.   :168.00   Max.   :104.00
##
##     Rec_Yds       Rec_Yds_per_Rec      Rec_TD          Fmb_Loss
##  Min.   :   0.0   Min.   : 0.000   Min.   : 0.000   Min.   :0.0000
##  1st Qu.: 202.8   1st Qu.: 9.012   1st Qu.: 2.000   1st Qu.:0.0000
##  Median : 529.5   Median :10.770   Median : 4.000   Median :0.5000
##  Mean   : 514.4   Mean   :10.067   Mean   : 3.786   Mean   :0.7857
##  3rd Qu.: 757.8   3rd Qu.:12.125   3rd Qu.: 4.000   3rd Qu.:1.0000
##  Max.   :1297.0   Max.   :18.880   Max.   :15.000   Max.   :3.0000
##
##     Total_TD    Two_Point_Conv Two_Point_Pass       Fant_Pts
##  Min.   : 2.0   Min.   :0.00   Min.   :0.00000   Min.   : 29.00
##  1st Qu.: 2.5   1st Qu.:0.00   1st Qu.:0.00000   1st Qu.: 80.25
##  Median : 5.0   Median :0.00   Median :0.00000   Median :123.00
##  Mean   : 6.5   Mean   :0.50   Mean   :0.07143   Mean   :151.79
##  3rd Qu.: 6.0   3rd Qu.:0.75   3rd Qu.:0.00000   3rd Qu.:224.50
```

```
##  3rd Qu.: 0.0   3rd Qu.:0.75   3rd Qu.:0.00000   3rd Qu.:224.50
##  Max.   :18.0   Max.   :3.00   Max.   :1.00000   Max.   :332.00
##
##     PPR_Pts      DraftKing_Pts    FanDuel_Pts    Value_Over_Baseline
##  Min.   : 43.5   Min.   : 46.5   Min.   : 36.0   Min.   : 10.00
##  1st Qu.:134.7   1st Qu.:138.0   1st Qu.:107.3   1st Qu.: 27.25
##  Median :166.8   Median :170.3   Median :144.8   Median : 69.50
##  Mean   :196.5   Mean   :202.4   Mean   :175.6   Mean   : 68.33
##  3rd Qu.:279.9   3rd Qu.:293.4   3rd Qu.:266.4   3rd Qu.: 99.75
##  Max.   :354.2   Max.   :360.2   Max.   :340.7   Max.   :138.00
##                                                  NA's   :8
##     Rank_Pos       Rank_Ovrl        Fant_PPG
##  Min.   : 2.00   Min.   : 4.00   Min.   : 1.812
##  1st Qu.: 6.00   1st Qu.:11.00   1st Qu.: 5.016
##  Median :18.00   Median :31.00   Median : 8.312
##  Mean   :27.29   Mean   :35.43   Mean   :10.133
##  3rd Qu.:32.25   3rd Qu.:58.00   3rd Qu.:16.792
##  Max.   :96.00   Max.   :75.00   Max.   :20.750
##                  NA's   :7
```

```
#fit the tree using rpart...after several iterations, minsplit = 20 was used
fit_tree_ppg <- rpart(Fant_PPG ~ Age + G + GS + Pass_Cmp + Pass_Att + Pass_Yds + Pass_TD + Pass_In
t + Rush_Att + Rush_Yds + Rush_Yds_per_Att + Rush_TD + Rec_Tgt + Rec_Receptions + Rec_Yds + Rec_Yd
s_per_Rec + Rec_TD + Fmb_Loss + Total_TD + Two_Point_Conv + Two_Point_Pass,
                  data = train_pro_ffball_ppg,
                  minsplit = 20)

#visualize the  tree
rpart.plot(fit_tree_ppg, type = 2, extra = "auto", box.palette="RdBu", shadow.col="gray", nn=TRUE)
```




```
#output the complexity parameter info
printcp(fit_tree_ppg)
```

```
##
## Regression tree:
## rpart(formula = Fant_PPG ~ Age + G + GS + Pass_Cmp + Pass_Att +
##     Pass_Yds + Pass_TD + Pass_Int + Rush_Att + Rush_Yds + Rush_Yds_per_Att +
##     Rush_TD + Rec_Tgt + Rec_Receptions + Rec_Yds + Rec_Yds_per_Rec +
##     Rec_TD + Fmb_Loss + Total_TD + Two_Point_Conv + Two_Point_Pass,
##     data = train_pro_ffball_ppg, minsplit = 20)
##
## Variables actually used in tree construction:
## [1] Pass_Cmp Pass_TD  Rec_Tgt  Rec_Yds  Rush_Att
##
## Root node error: 4785.3/145 = 33.002
##
## n= 145
##
##         CP nsplit rel error  xerror     xstd
## 1 0.475264      0   1.00000 1.01610 0.110552
## 2 0.179326      1   0.52474 0.61635 0.071685
## 3 0.124959      2   0.34541 0.41036 0.059532
## 4 0.043679      3   0.22045 0.28085 0.044477
## 5 0.036479      4   0.17677 0.28078 0.043804
## 6 0.023949      5   0.14029 0.24067 0.048483
## 7 0.015146      6   0.11634 0.23351 0.048678
## 8 0.010000      7   0.10120 0.22091 0.047919
```

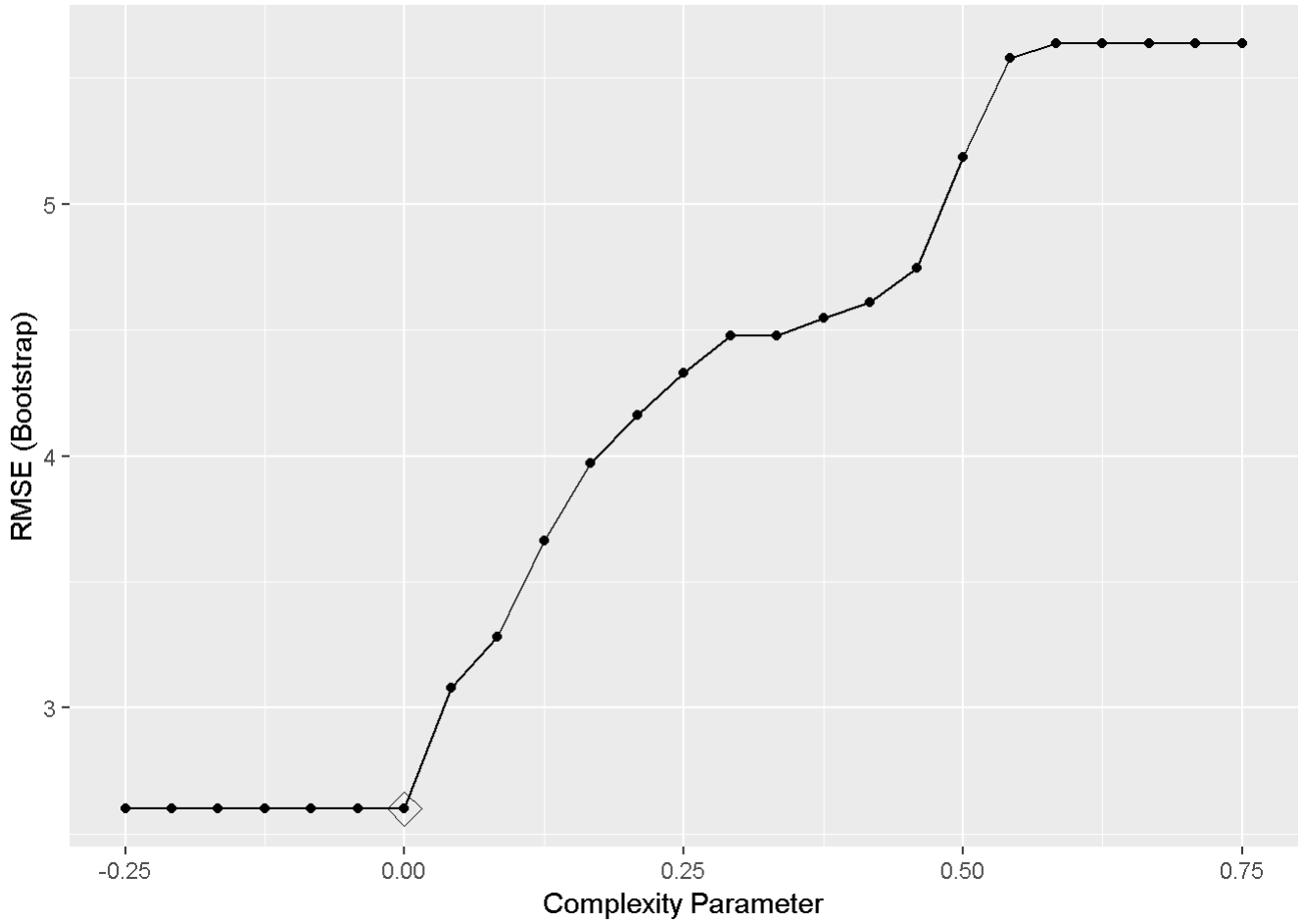
```
#How to choose optimal CP? Train several values of cp, see which one minimizes the error. Note tha
t the tuning parameter for train's rpart method is the Complexity Parameter, as per http://topepo.
github.io/caret/available-models.html
fit_rpart_ppgs <- train(Fant_PPG ~ Age + G + GS + Pass_Cmp + Pass_Att + Pass_Yds + Pass_TD + Pass_
Int + Rush_Att + Rush_Yds + Rush_Yds_per_Att + Rush_TD + Rec_Tgt + Rec_Receptions + Rec_Yds + Rec_
Yds_per_Rec + Rec_TD + Fmb_Loss + Total_TD + Two_Point_Conv + Two_Point_Pass,
                     method = 'rpart',
                     tuneGrid = data.frame(cp = seq(-.25, .75, len = 25)),
                     data = train_pro_ffball_ppg)
```

```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info =
## trainInfo, : There were missing values in resampled performance measures.
```

```
#see the best value of CP
fit_rpart_ppgs$bestTune
```

```
##   cp
## 7  0
```

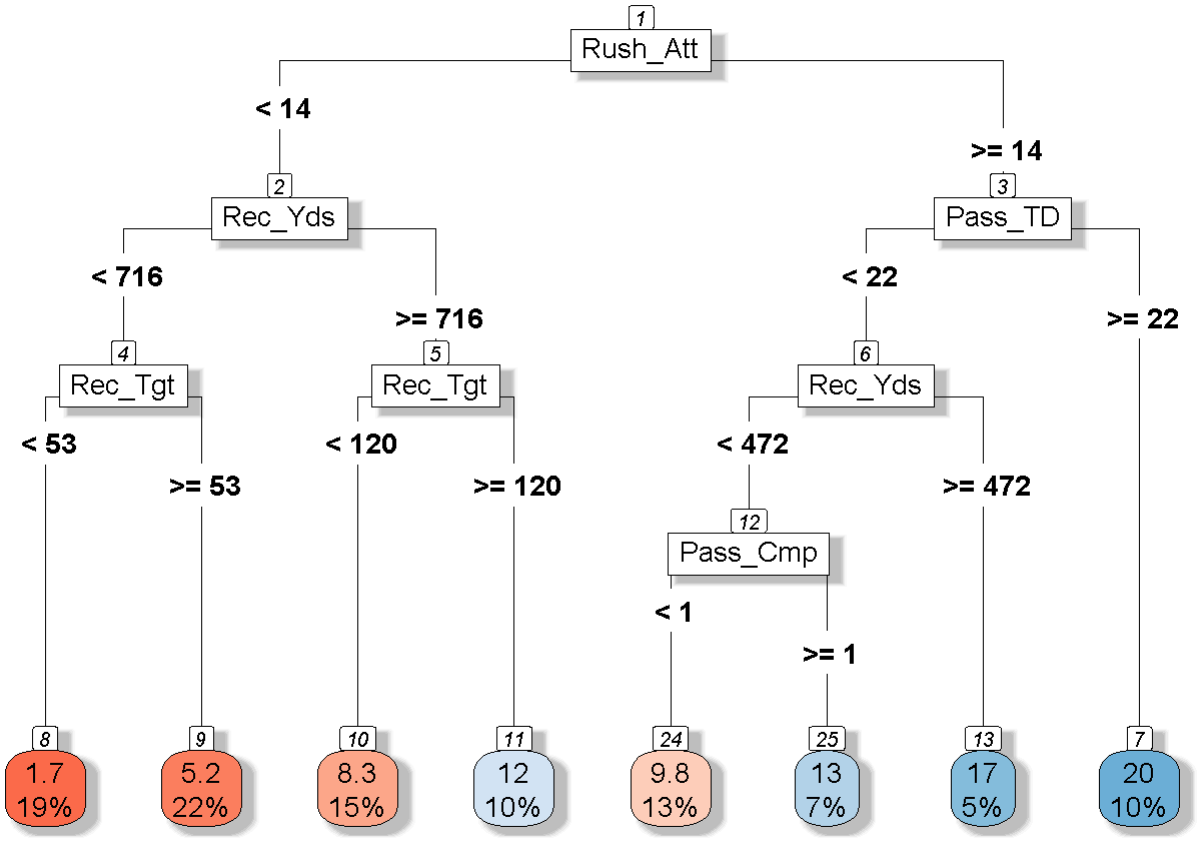
```
#plot the complexity parameter vs the RMSE. This aligns with the above results about the optimal c
omplexity parameter that minimizes the RMSE
ggplot(fit_rpart_ppgs, highlight = TRUE)
```




```
#find the smallest RMSE
RMSE_reg_tree_ppgs_min <- fit_rpart_ppgs$results$RMSE[which.min(fit_rpart_ppgs$results$RMSE)]
```

The cp that minimizes the RMSE is 0. Use that cp when pruning the tree.

```
#prune fit
pruned_fit_ppg <- prune.rpart(fit_tree_ppg, cp = 0)

#visualize the pruned fit
rpart.plot(pruned_fit_ppg, type = 5, extra = "auto", box.palette="RdBu", shadow.col="gray", nn=TRU
E)
```




```
#calculate RMSE of pruned fit
y_hat_pruned_ppg <- predict(pruned_fit_ppg, test_pro_ffball_ppg)
RMSE_reg_tree_pruned_ppg <- sqrt(mean(y_hat_pruned_ppg - test_pro_ffball_ppg$Fant_Pts)^2)
```

After ony including players who started more than half the games (8 or more) and played in at least 3/4 of the games (12 or more), we see the tree shifts its important variables to Rushing Attempts, Receiving Yards, Receiving Targets, Pass TD, and Pass Completions as key varaibles.

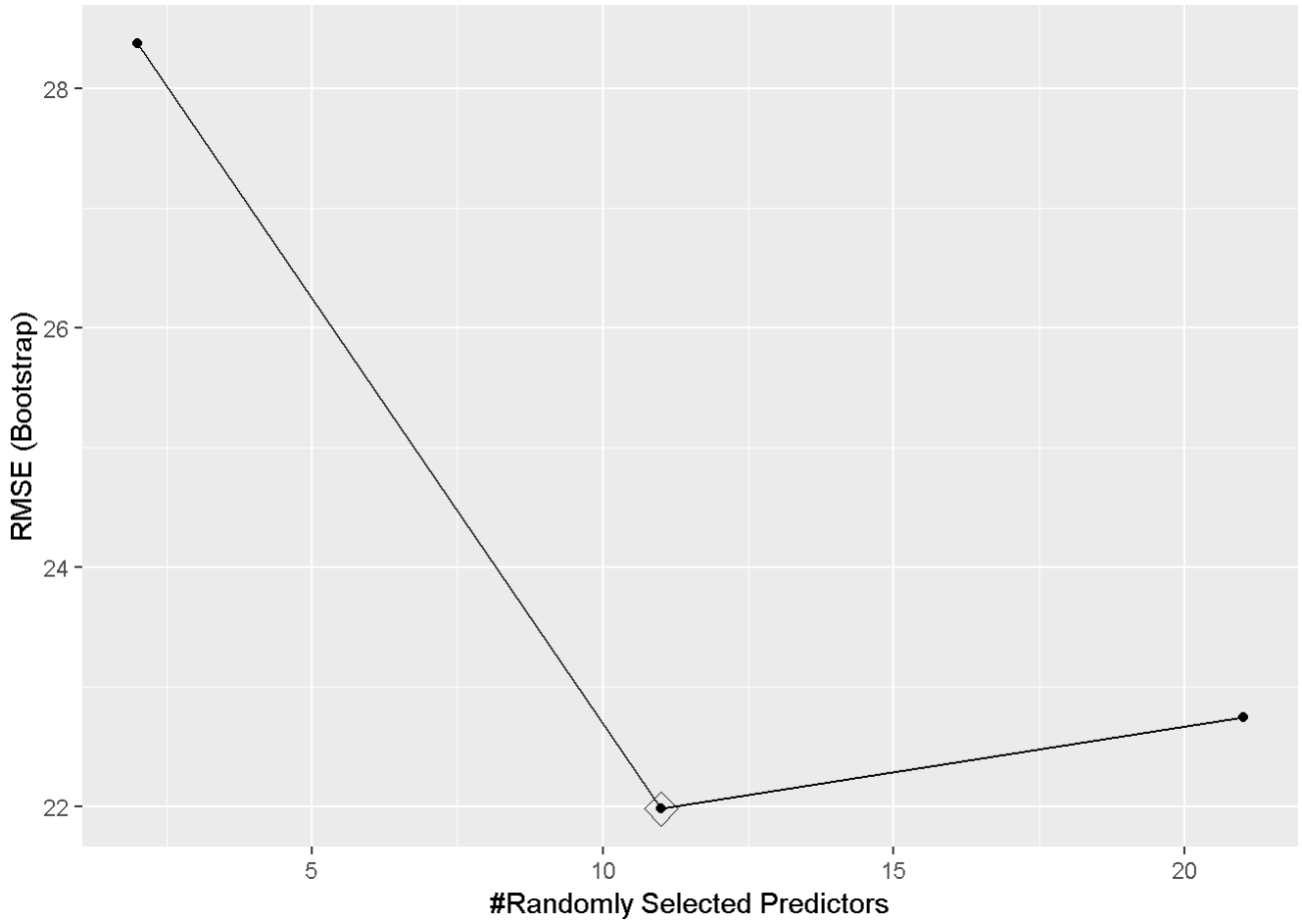
It's also a balanced tree that has a reasonable percentage of the overall dataset in each leaf. Likewise, the predicted amount in each leaf is reasonable. This alone doesn't guarantee a better model, but are good qualities of a tree-based model.

The combination of many trees results in the random forest algorithm. We'll try random forest next to predict fantasy points based on other variables.

```
#set the new grid with tuning parameters minNode and predFixed
grid_rf <- expand.grid(predFixed = 1, minNode = seq(25, 100, 25))

#random forest training...random forest is able to take many features and each run only uses a sub
set of the factors. Then the output of the model uses only select features.
rf_fit <- train(Fant_Pts ~ Age + G + GS + Pass_Cmp + Pass_Att + Pass_Yds + Pass_TD + Pass_Int + Ru
sh_Att + Rush_Yds + Rush_Yds_per_Att + Rush_TD + Rec_Tgt + Rec_Receptions + Rec_Yds + Rec_Yds_per_
Rec + Rec_TD + Fmb_Loss + Total_TD + Two_Point_Conv + Two_Point_Pass,
                method = 'Rborist',
                tune_grid = grid_rf,
                data = train_pro_ffball)

#we can check the results of the random forest...
rf_fit$results
```

```
##   predFixed minNode     RMSE  Rsquared       MAE   RMSESD RsquaredSD
## 1         2       3 28.37671 0.9273646 13.442320 4.040698 0.01636257
## 2        11       3 21.97991 0.9446709  9.811386 3.638742 0.01118988
## 3        21       3 22.74324 0.9411935 10.346960 4.104278 0.01144572
##      MAESD
## 1 2.015436
## 2 1.545768
## 3 1.651295
```

```
#plot the output
ggplot(rf_fit, highlight = TRUE)
```




```
#fit the smallest RMSE
RMSE_rf_min <- rf_fit$results$RMSE[which.min(rf_fit$results$RMSE)]
```

Results

As discussed above, we use RMSE to determine the model error

```
#evaluate the RMSE of different models. The one with the lowest RMSE will be used.
RMSEs <- tibble(Model = c("Linear Regression - p-value variable selection", "Linear Regression - l
asso variable selection", "Regression Tree", "Pruned Regression Tree", "Pruned Points Per Game Reg
ression Tree", "Random Forest"), RMSE = c(RMSE_lin_reg, RMSE_lin_reg_lasso, RMSE_reg_tree_min, RMS
E_reg_tree_pruned, RMSE_reg_tree_pruned_ppg, RMSE_rf_min))

#output the results
RMSEs
```

```
## # A tibble: 6 x 2
##   Model                                          RMSE
##   <chr>                                         <dbl>
## 1 Linear Regression - p-value variable selection   0.366
## 2 Linear Regression - lasso variable selection     2.56
## 3 Regression Tree                                 26.3
## 4 Pruned Regression Tree                           0.252
## 5 Pruned Points Per Game Regression Tree         142.
## 6 Random Forest                                   22.0
```

From this, we see that the Pruned Regression Tree and the original Linear Regression model using p-values to select variables have the best RMSEs. Both have an RMSE below 1, suggesting these models offer strong predictions of Fantasy Points. We should also consider the Pruned Points per Game Regression Tree, because this is predicting data on a different amount (points per game rather than total points), so is a separate scale, but appears to be a quality model based on the tree output.

Conclusion

Ultimately, this analysis has shown that several variables, especially Touchdown-related and Yard-related variables are strong predictors of Fantasy Points scored. Therefore, it's best to target players who are expected to score many touchdowns and accumulate many yards to have better overall scores.

However, the interest of this is limited, as there are direct relationships between some of these variables and fantasy points. In other words, it's known ahead of time that more touchdowns and more yards lead to more points.

I see this effort as a starting point. Further topics that can be explored include: deriving advanced statistics that tell of underlying qualities of successful players, optimizing a lineup subject to positional constraints, desiging separate models for different positions, further explore a player's points per game totals instead of total points for players, analyzing publicly available player projections to analyze quality of each source's projections, and determining an optimal draft strategy, among others.

However, this project was a learning experience and gave me an opportunity to practice several applications that have made me more confident in my skills. I look forward to continue growing and continue practicing.