NFL Data Analysis: Taking a look at what affects Expected Points Added (EPA) on pass plays

Lawrence Ho, Paul Jung

2024-12-06

EPA is a statistic that was created to evaluate a team or player’s performance relative to expectations.

Expected points are based off the notion that not all yards gained in football are of equal value. A gain of 5 yards has many implications depending on the scenario/circumstance regarding the gain and Expected Points are meant to quantify these values.

The dataset we used for the bulk of our analysis was “plays”

Plays has 16124 Observations and 49 features. Within the data, there are columns for what seems like solely identification purposes (ex: gameId, playId, playDescription, etc.), some basic football statistics/categorizations (yardsGained, offensiveFormation, passResult, passLength, etc.), as well as some advanced statistics/categorizations (expectedPointsAdded, timeToThrow, teamwinProbabilities, unblockedPressure, pff_manZone, etc.)

For our project, we used EPA as the outcome “grade” of a play and we wanted to focus on team controlled predictors before and during the play.

This being said, we avoided features that are “indepdnedent” of game decisions made by the teams such as yardLine and gameClock.

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.3.2
```

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.3.2
```

```
## Warning: package 'tidyr' was built under R version 4.3.2
```

```
## Warning: package 'readr' was built under R version 4.3.2
## Warning: package 'purrr' was built under R version 4.3.2
```

```
## Warning: package 'dplyr' was built under R version 4.3.2
```

```
## Warning: package 'stringr' was built under R version 4.3.2
```

```
## Warning: package 'forcats' was built under R version 4.3.2
```

```
## Warning: package 'lubridate' was built under R version 4.3.2
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✔ dplyr     1.1.3     ✔ readr     2.1.4
## ✔ forcats   1.0.0     ✔ stringr   1.5.0
## ✔ lubridate 1.9.3     ✔ tibble    3.2.1
## ✔ purrr     1.0.2     ✔ tidyr     1.3.0
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becom
e errors
```

```
#install.packages("skimr")
library(skimr)
```

```
## Warning: package 'skimr' was built under R version 4.3.3
```

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 4.3.2
```

```
## randomForest 4.7-1.1
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
##
## The following object is masked from 'package:dplyr':
##
##     combine
##
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.3.2
```

```
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(class)
set.seed(123)

##glimpse(plays)
setwd("~/APPSTAT")
plays<- read.csv("plays.csv")
```

Exploratory analysis

We are trying to find any clear and obvious relationships between features and Expected Points Added (EPA) on plays.

To get an idea of the types of strategies teams are employing, we showed a distribution of intended pass plays vs run plays. We then filtered out the run data to create a passing dataset and began our exploratory analysis. Throughout the exploratory analysis of some features we picked to predict EPA, we noticed there were very few features that were "obviously" correlated to EPA. This being said, we created a new statistic called passDifficulty where we combined all of the features we believed had an effect on the pass into one standardized "score". We determined the weights of certain categories through prior assumptions as well as some exploratory analysis.

Assumption Weights: the dropbackType feature weights were assigned by prior assumptions that making an accurate throw while running is more difficult than making an accurate throw while in a more "stable" position like in a traditional dropback. We are also assuming naturally that the longer the passLength is, the more difficult it is to throw a accurate "successful" ball. For pass location, we assume that throws outside the numbers will be more difficult as the ball travels in the air longer (more chances for the defense to interrupt the offense). For play action, we assume that it's easier to throw when there's a designed "fake run play" as opposed to without because it forces a slight hesitation in the defense reacting to the pass.

Exploratory weights: For judging which coverage made throwing a +EPA ball more difficult, we looked at the average EPA of each type of coverage (lower averages->more difficult to find success against ___ coverage). The weights of pff_manZone accurately reflect our findings.For judging the weights on an unblockedPressure, we created boxplots that showed that there tends to be more time to throw if there is no unblockedPressure. We can also use prior knowledge, with unblocked pressure, the longer the quarterback holds the ball, the more difficult the throw will be (as there's a free rusher running straight to the quarterback), hence a greater weight being applied to normalizedTimeToThrow.

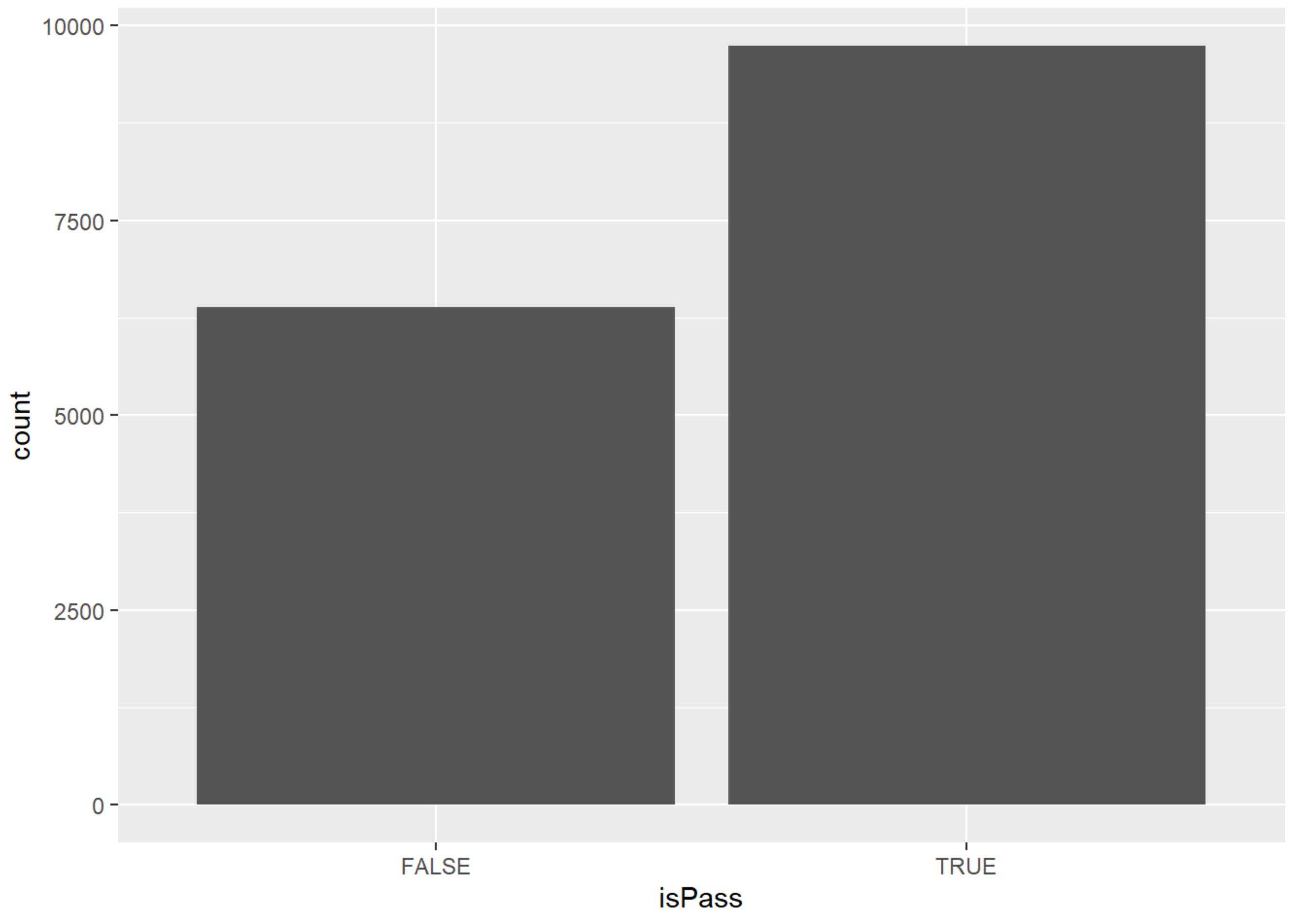
The computation of passDifficulty was:

Pressure Adjustment (0.6 or 0.05) × Norm Time to Throw+Defensive Coverage (0.3(Man), 0.35(Zone), .6(Other))+Pass Length (0.6 × Norm Pass Length)+Dropback Type Weight+Pass Location Weight+Play Action Adjustment (0.4 or 0.6). We then normalized this statistic with (x-mean(passDifficulty))/sd(passDifficulty). We ended up also running a simple regression between normPassDifficulty and EPA to see whether there is a more obvious relationship but we ended up observing yet another seemingly no obvious correlation between the two variables.

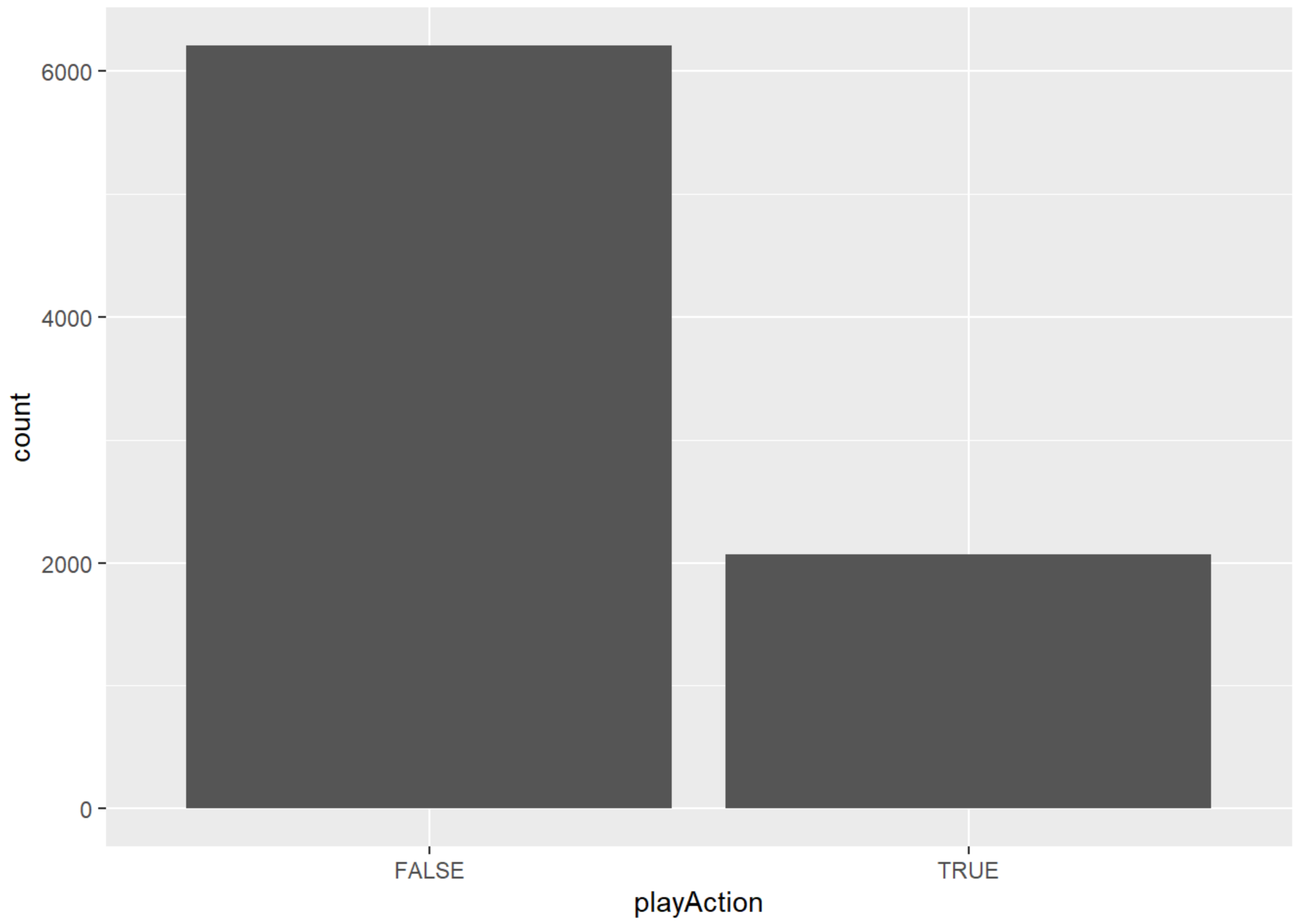
Imputation of missing values

During the calculation of the mean and standard deviation of passDifficulty, columns with NA values of passDifficulty were ignored and replaced with the average pass difficulty. Missing receiverAlignments, offensiveFormations, and quarters were imputed with the median of their respective columns as well.

```
## More passing than running in the NFL

ggplot(data = plays, mapping = aes(x = isDropback, ))+geom_bar()+labs(x="isPass")
```

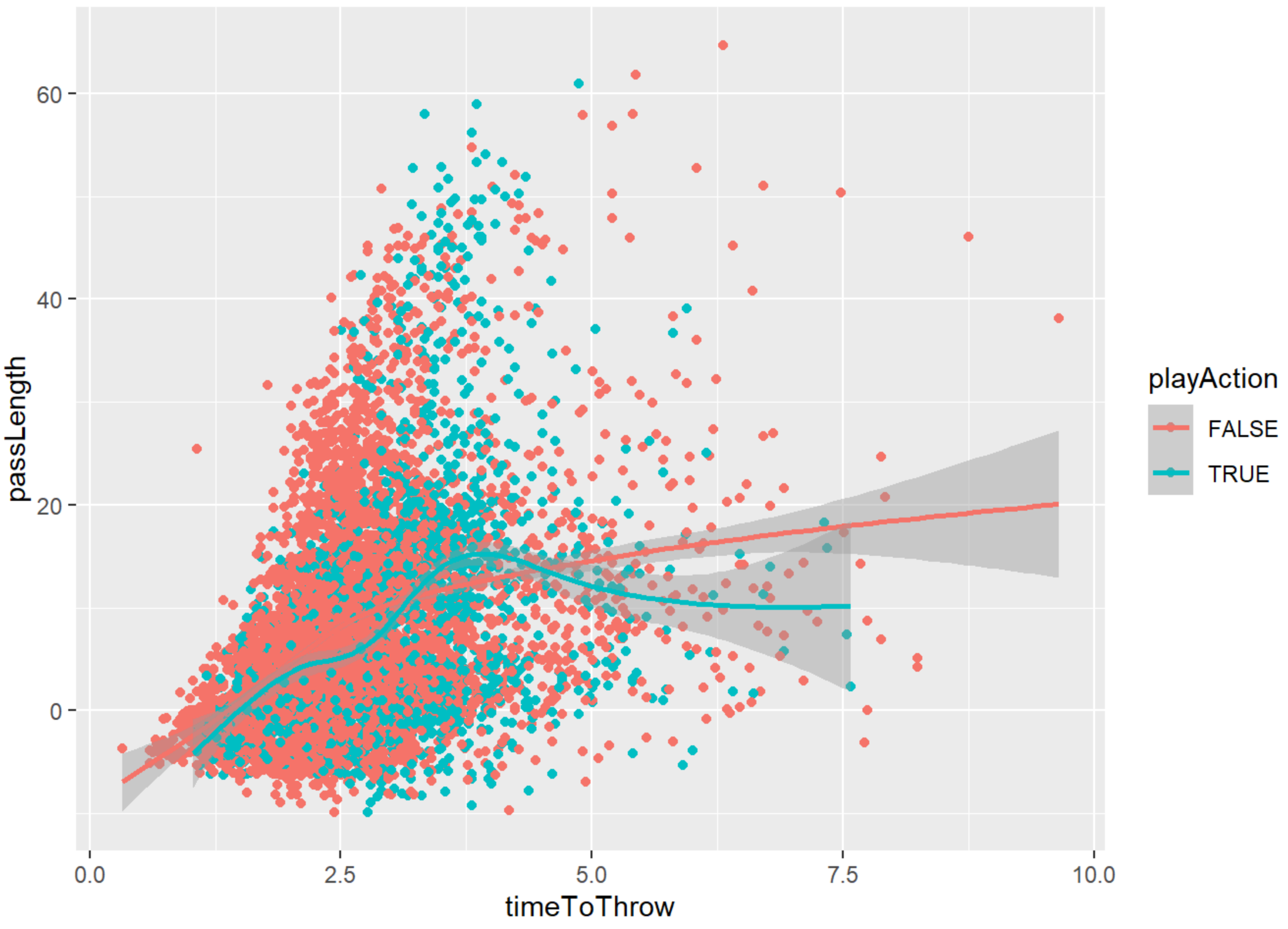


```
## Decided to work with pass data
plays <- plays %>% select(-penaltyYards)
#glimpse(plays)
pass_data <- plays %>% filter(isDropback == TRUE) %>% select(-pff_runConceptPrimary)%>%
  select(-pff_runConceptSecondary)%>%select(-pff_runPassOption)%>%select(-rushLocationType)%>%se
lect(-qbSpike)%>%
  select(-qbKneel)%>%select(-qbSneak)%>%mutate(timeToSack=replace_na(0))

pass_data<- drop_na(pass_data)

## Teams are employing play action within their
ggplot(data = pass_data, mapping = aes(x = playAction, ))+geom_bar()
```



```
## No obvious correlation between time to throw and yards gained
ggplot(data = pass_data, mapping = aes(x = timeToThrow, y=passLength, colour = playAction))+geom
_jitter()+geom_smooth()
```

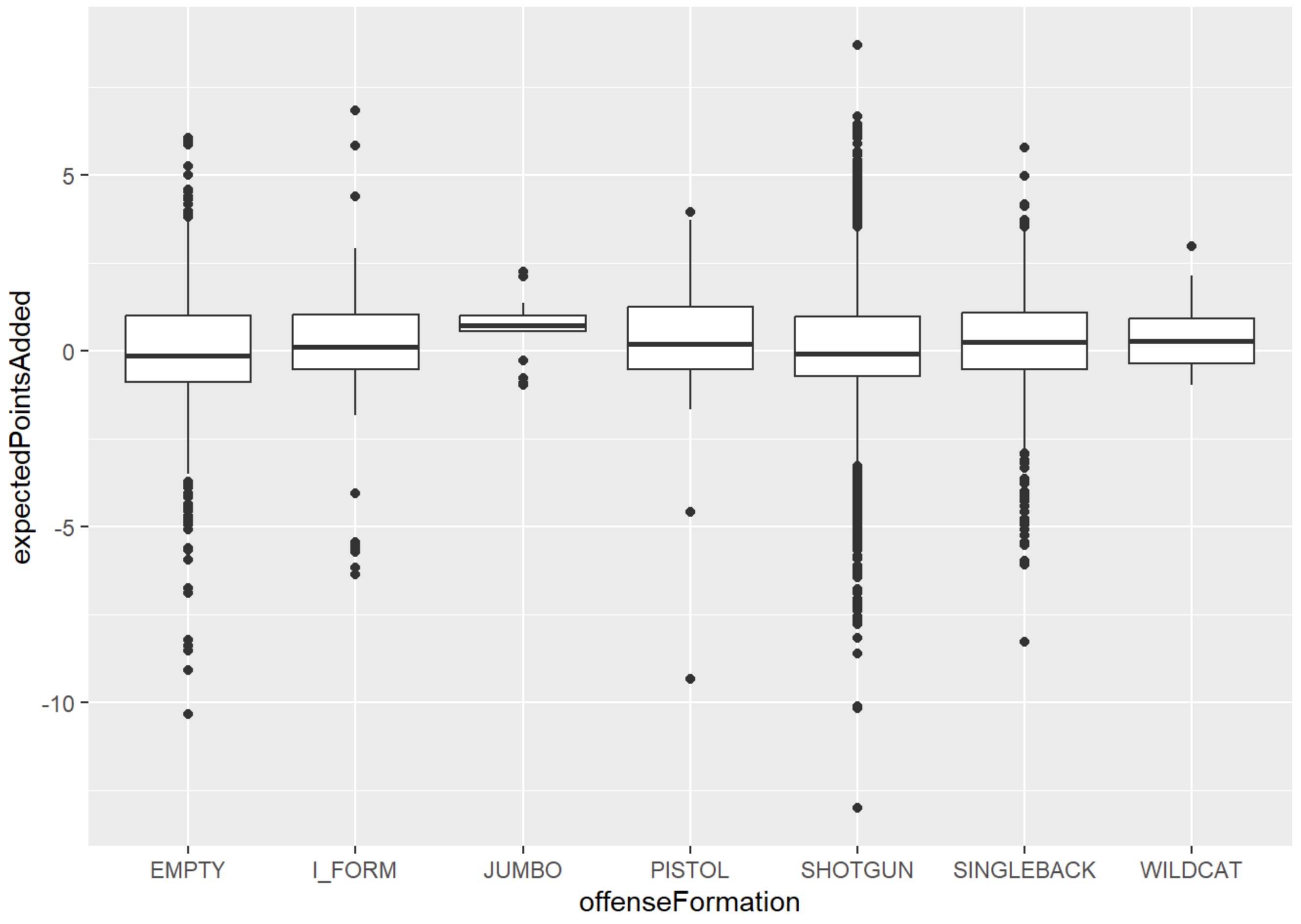
```
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```



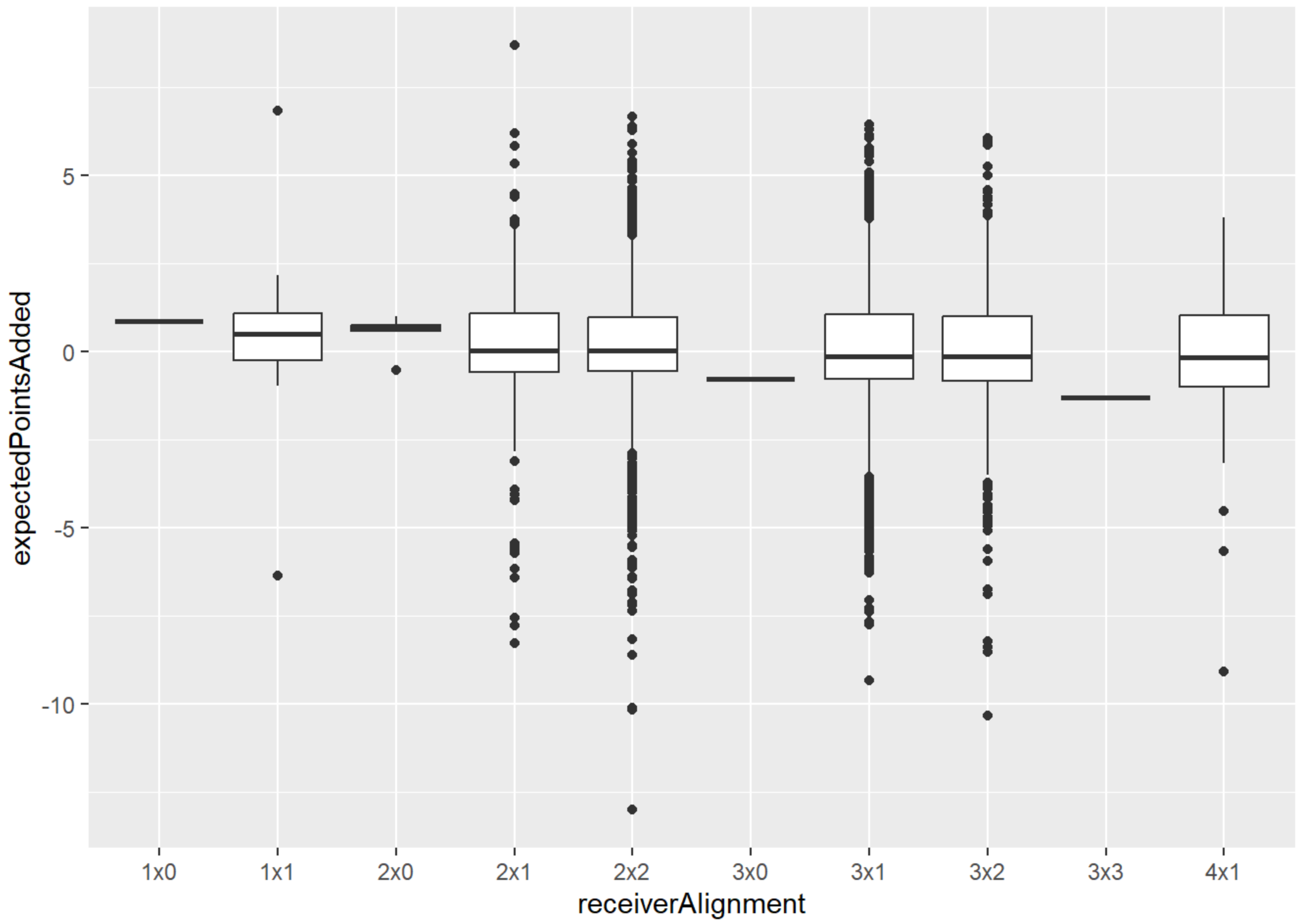
```
cor(plays$timeToThrow, plays$passLength)
```

```
## [1] NA
```

```
ggplot(data = pass_data, mapping = aes(x = offenseFormation, y= expectedPointsAdded))+geom_boxpl
ot()
```



```
ggplot(data = pass_data, mapping = aes(x = receiverAlignment, y= expectedPointsAdded))+geom_boxp
lot()
```



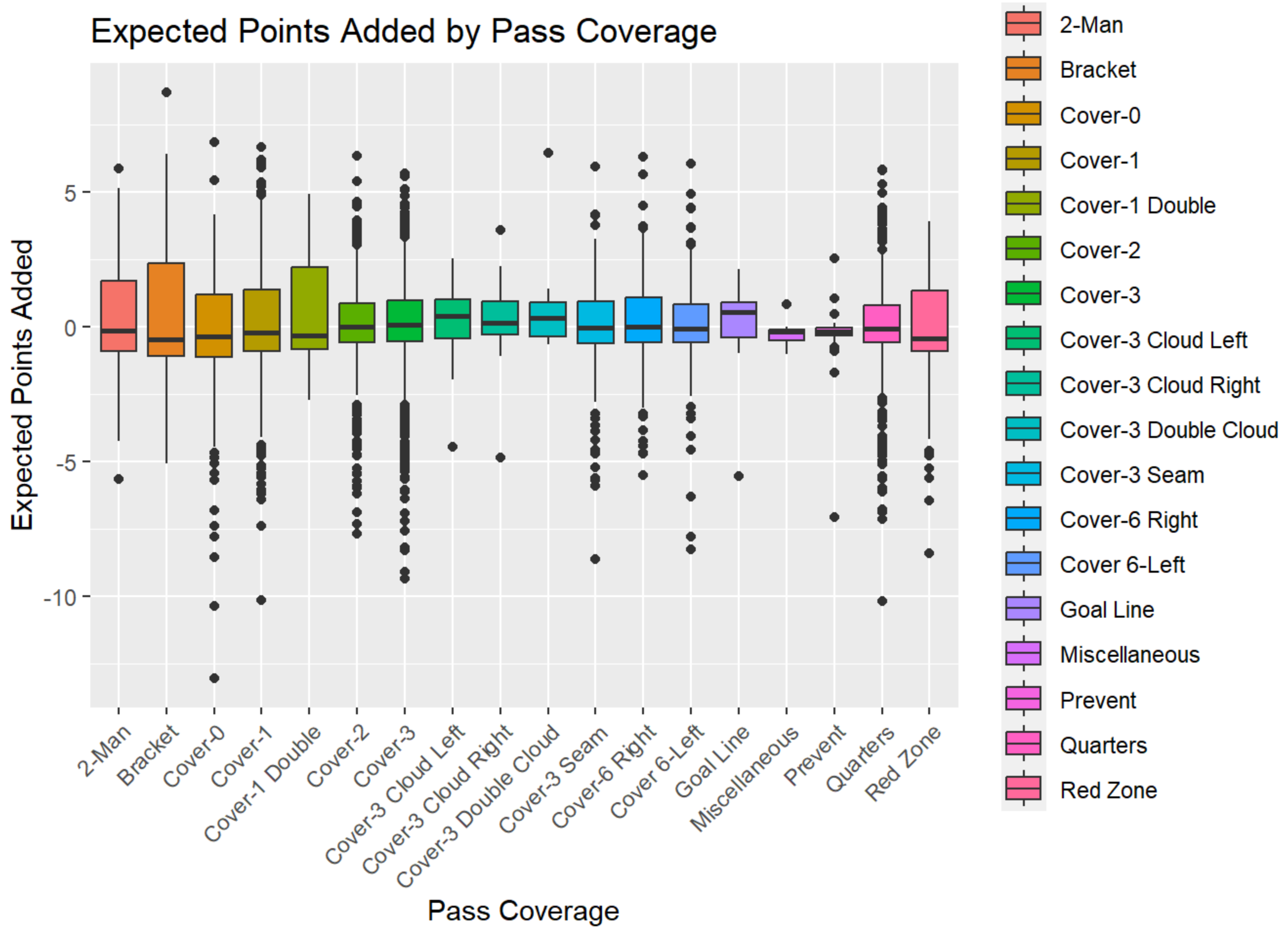
```
exploratory1<- lm(formula = expectedPointsAdded~offenseFormation, data = pass_data )
summary(exploratory1)
```

```
##
## Call:
## lm(formula = expectedPointsAdded ~ offenseFormation, data = pass_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -13.1405  -0.8077  -0.1807   0.8782   8.5821
##
## Coefficients:
##                             Estimate Std. Error t value Pr(>|t|)
## (Intercept)                  0.02694    0.04710   0.572  0.56733
## offenseFormationI_FORM       0.16393    0.11118   1.474  0.14039
## offenseFormationJUMBO        0.61287    0.37451   1.636  0.10179
## offenseFormationPISTOL       0.33307    0.12568   2.650  0.00806 **
## offenseFormationSHOTGUN      0.08999    0.05159   1.744  0.08116 .
## offenseFormationSINGLEBACK  0.21605    0.06712   3.219  0.00129 **
## offenseFormationWILDCAT      0.44343    0.45747   0.969  0.33243
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.576 on 8264 degrees of freedom
## Multiple R-squared:  0.002108,   Adjusted R-squared:  0.001384
## F-statistic:  2.91 on 6 and 8264 DF,  p-value: 0.007775
```

```
exploratory2<- lm(formula = expectedPointsAdded~receiverAlignment, data = pass_data )
summary(exploratory2)
```

```
##
## Call:
## lm(formula = expectedPointsAdded ~ receiverAlignment, data = pass_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -13.1995  -0.7953  -0.1810   0.8806   8.5045
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)            0.8742     1.5769   0.554    0.579
## receiverAlignment1x1  -0.3832     1.6058  -0.239    0.811
## receiverAlignment2x0  -0.3447     1.7033  -0.202    0.840
## receiverAlignment2x1  -0.6798     1.5784  -0.431    0.667
## receiverAlignment2x2  -0.6983     1.5771  -0.443    0.658
## receiverAlignment3x0  -1.6506     2.2301  -0.740    0.459
## receiverAlignment3x1  -0.7674     1.5772  -0.487    0.627
## receiverAlignment3x2  -0.8332     1.5777  -0.528    0.597
## receiverAlignment3x3  -2.1673     2.2301  -0.972    0.331
## receiverAlignment4x1  -1.0340     1.5849  -0.652    0.514
##
## Residual standard error: 1.577 on 8261 degrees of freedom
## Multiple R-squared:  0.001717,   Adjusted R-squared:  0.0006299
## F-statistic: 1.579 on 9 and 8261 DF,  p-value: 0.1152
```

```
ggplot(data = pass_data, aes(x = pff_passCoverage, y = expectedPointsAdded, fill = pff_passCover
age)) +
  geom_boxplot() +
  labs(
    title = "Expected Points Added by Pass Coverage",
    x = "Pass Coverage",
    y = "Expected Points Added",
    fill = "Pass Coverage"
  ) +
  theme(
    legend.position = "right",
    axis.text.x = element_text(angle = 45, hjust = 1)
  )
```



```
exploratory3<- lm(formula = expectedPointsAdded~pff_passCoverage, data = pass_data )
summary(exploratory3)
```

```
##
## Call:
## lm(formula = expectedPointsAdded ~ pff_passCoverage, data = pass_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -12.8276  -0.7891  -0.1696   0.8846   8.2223
##
## Coefficients:
##                                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)                           0.190021   0.130836   1.452   0.1464
## pff_passCoverageBracket               0.286682   0.237733   1.206   0.2279
## pff_passCoverageCover-0              -0.385999   0.158672  -2.433   0.0150 *
## pff_passCoverageCover-1              -0.006341   0.136378  -0.046   0.9629
## pff_passCoverageCover-1 Double        0.403587   0.273572   1.475   0.1402
## pff_passCoverageCover-2              -0.086415   0.138773  -0.623   0.5335
## pff_passCoverageCover-3              -0.009230   0.135137  -0.068   0.9455
## pff_passCoverageCover-3 Cloud Left   -0.124011   0.427309  -0.290   0.7717
## pff_passCoverageCover-3 Cloud Right   0.047526   0.360474   0.132   0.8951
## pff_passCoverageCover-3 Double Cloud  0.689255   0.541211   1.274   0.2029
## pff_passCoverageCover-3 Seam         -0.072169   0.150947  -0.478   0.6326
## pff_passCoverageCover-6 Right         0.019737   0.152669   0.129   0.8971
## pff_passCoverageCover 6-Left         -0.040832   0.153674  -0.266   0.7905
## pff_passCoverageGoal Line            -0.017541   0.360474  -0.049   0.9612
## pff_passCoverageMiscellaneous        -0.447825   0.515102  -0.869   0.3847
## pff_passCoveragePrevent              -0.527427   0.303865  -1.736   0.0826 .
## pff_passCoverageQuarters             -0.152961   0.139229  -1.099   0.2720
## pff_passCoverageRed Zone             -0.193539   0.161290  -1.200   0.2302
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.575 on 8253 degrees of freedom
## Multiple R-squared:  0.0045, Adjusted R-squared:  0.00245
## F-statistic: 2.195 on 17 and 8253 DF,  p-value: 0.003102
```

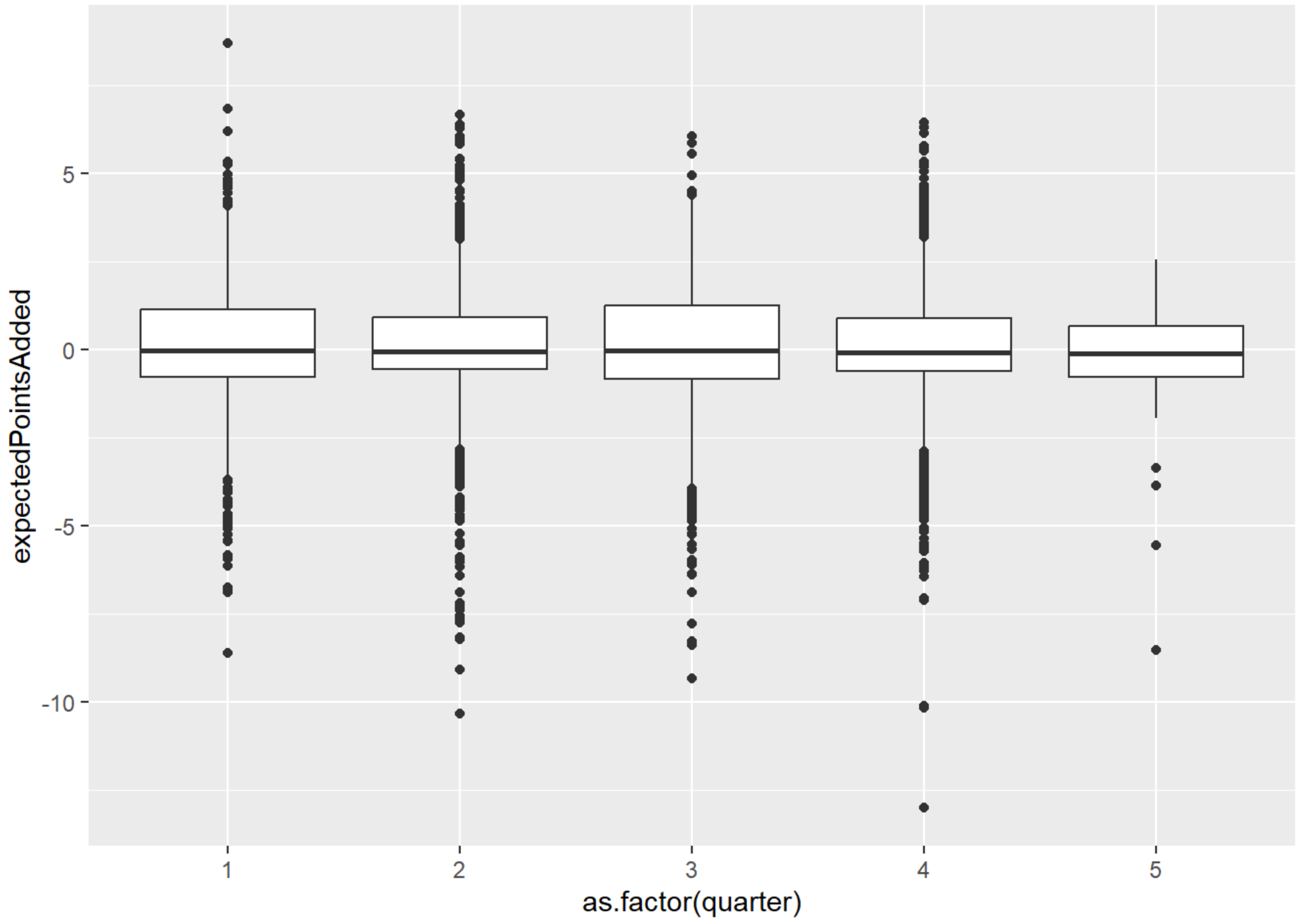
```
exploratory4<- lm(formula = expectedPointsAdded~pff_manZone, data = pass_data)
summary(exploratory4)
```

```
##
## Call:
## lm(formula = expectedPointsAdded ~ pff_manZone, data = pass_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -13.1620  -0.7853  -0.1865   0.8834   8.6520
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)       0.138382   0.033848   4.088 4.39e-05 ***
## pff_manZoneOther -0.091372   0.085205  -1.072    0.284
## pff_manZoneZone  -0.004873   0.039785  -0.122    0.903
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.577 on 8268 degrees of freedom
## Multiple R-squared:  0.0001469,  Adjusted R-squared:  -9.493e-05
## F-statistic: 0.6075 on 2 and 8268 DF,  p-value: 0.5447
```

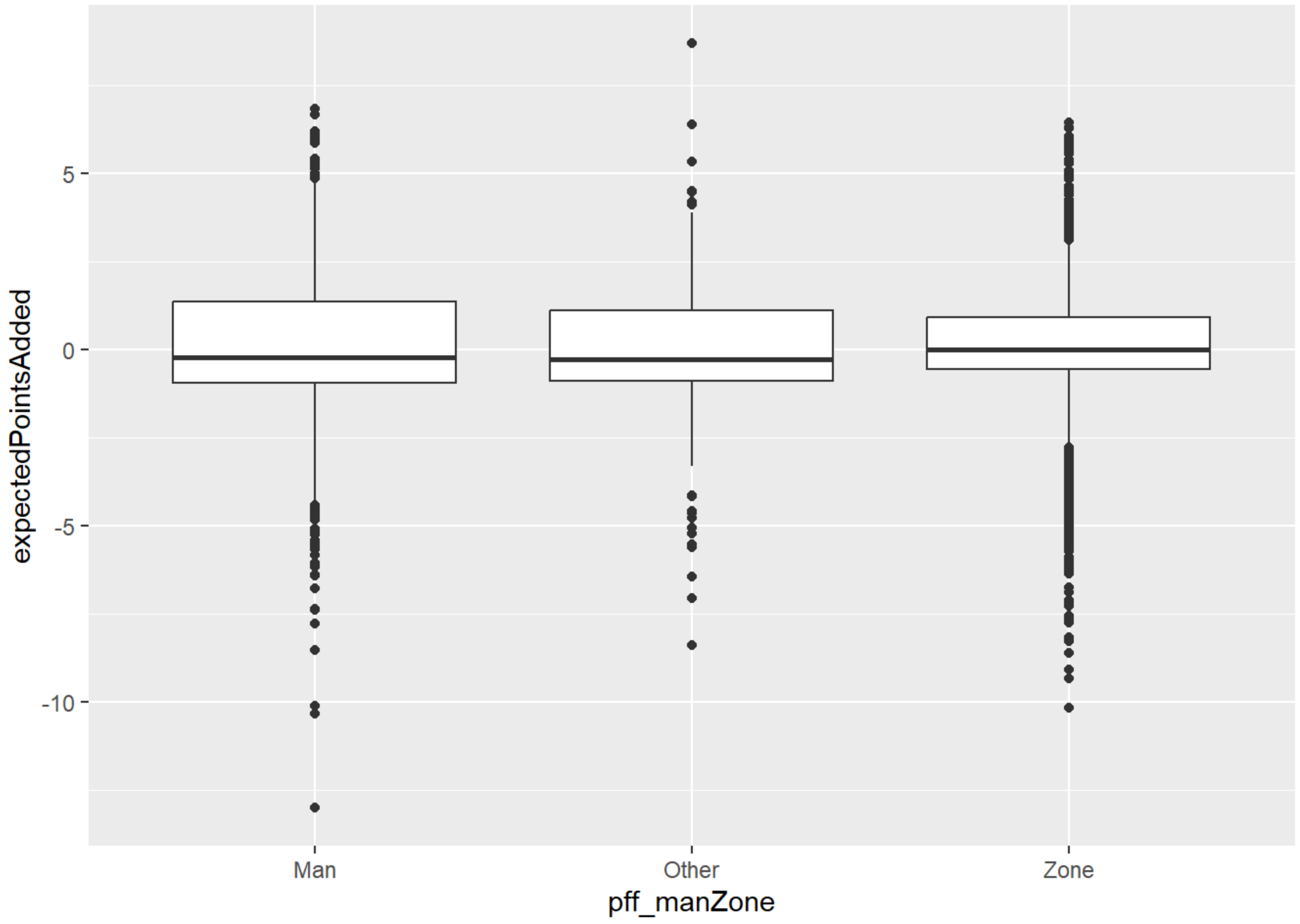
```
cor(x = pass_data$expectedPointsAdded, y = pass_data$quarter)
```

```
## [1] -0.02401855
```

```
ggplot(data = pass_data, mapping = aes(x=as.factor(quarter), y=expectedPointsAdded))+geom_boxplo
t()
```



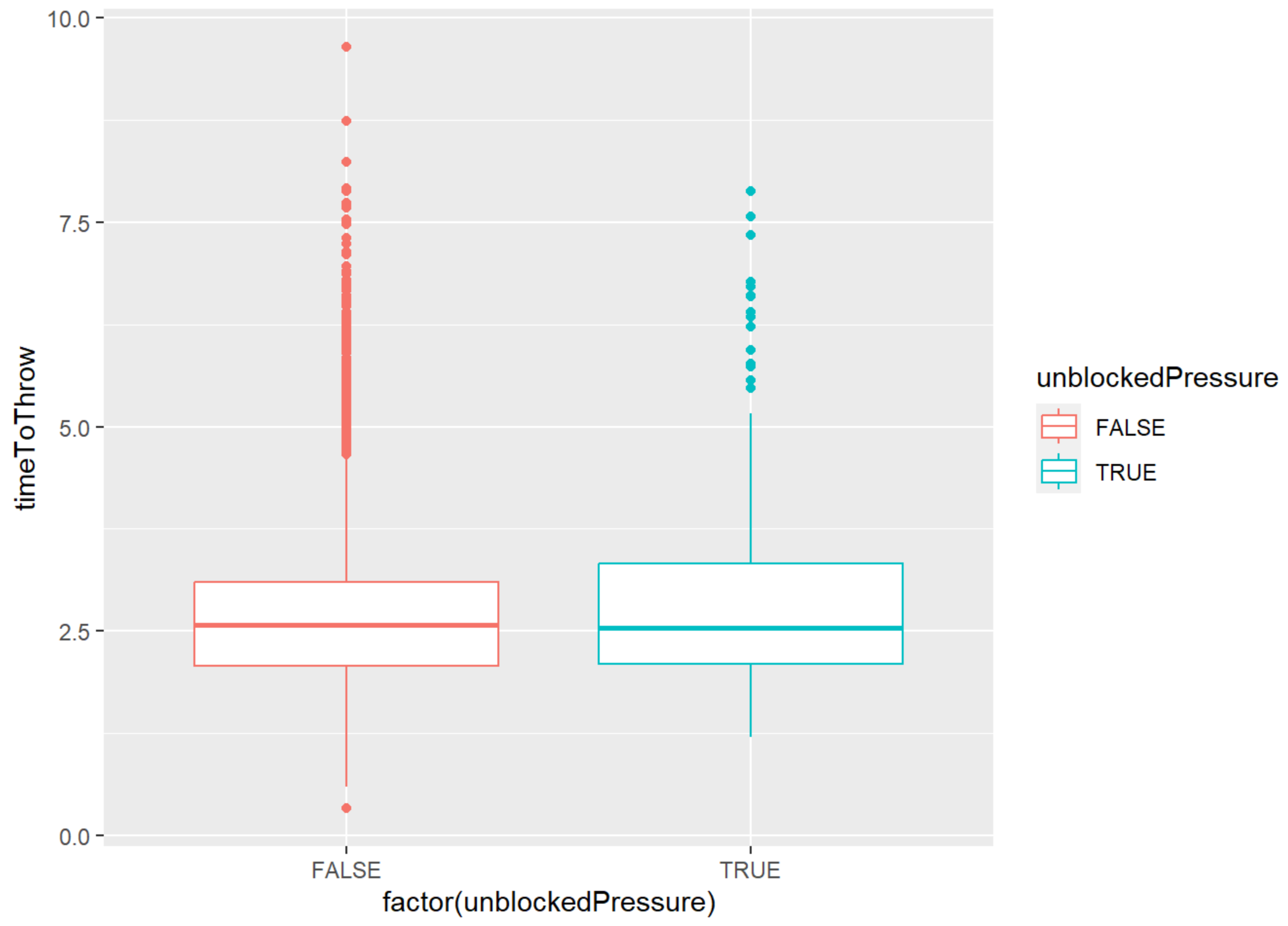
```
## Exploring type of coverage's effect on relative success of a play (EPA)
ggplot(data = pass_data, aes(x = pff_manZone, y = expectedPointsAdded,)) +
  geom_boxplot()
```



```
epa_by_cov <- pass_data %>%
  group_by(pff_manZone) %>%
  summarise(
    mean_epa = mean(expectedPointsAdded, na.rm = TRUE),
  )
epa_by_cov %>%drop_na()%>%
  arrange(mean_epa)
```

```
## # A tibble: 3 × 2
##   pff_manZone mean_epa
##   <chr>          <dbl>
## 1 Other         0.0470
## 2 Zone          0.134
## 3 Man           0.138
```

```
## Explore pressure time to throw

ggplot(data = pass_data, mapping = aes(x = factor(unblockedPressure), y = timeToThrow, colour =
unblockedPressure)) +
  geom_boxplot()
```



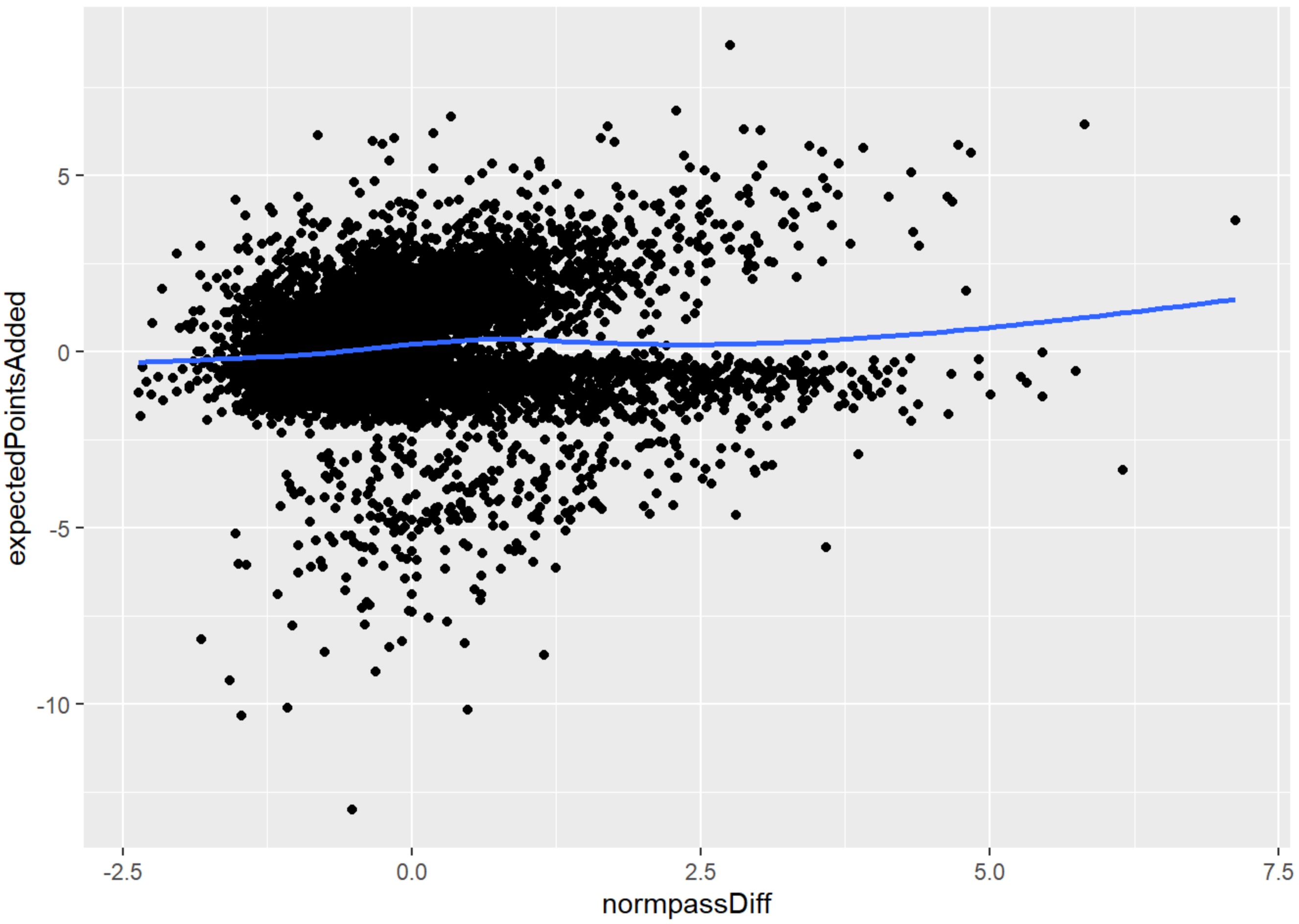
```
timexpressure<- lm(data = pass_data, formula = timeToThrow~unblockedPressure, )
summary(timexpressure)
```

```
##
## Call:
## lm(formula = timeToThrow ~ unblockedPressure, data = pass_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.3476 -0.6076 -0.1256  0.4264  6.9664
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)            2.67656    0.01057 253.326  < 2e-16 ***
## unblockedPressureTRUE  0.18051    0.04560   3.958 7.61e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9347 on 8269 degrees of freedom
## Multiple R-squared:  0.001891,	Adjusted R-squared:  0.001771
## F-statistic: 15.67 on 1 and 8269 DF,  p-value: 7.606e-05
## Engineering the statistic passData

pass_data2<- pass_data%>%
  mutate(
    normPassLength= ((passLength-mean(passLength))/sd(passLength)),
    normtimeToThrow = ((timeToThrow-mean(timeToThrow)/sd(timeToThrow))),
    normtimeInBox= ((timeInTackleBox-mean(timeInTackleBox))/sd(timeInTackleBox)),
    manZone= (ifelse(pff_manZone=="Man", 0.3, ifelse(pff_manZone=="Other", 0.6, ifelse(pff_manZo
ne == "NA", NA, 0.35)))),
    normdropbackDistance = ((dropbackDistance-mean(dropbackDistance)/sd(dropbackDistance)))
  )
#colSums(is.na(pass_data2))
pass_data2<- pass_data2%>% mutate(
  passDifficulty =
      ifelse(unblockedPressure==TRUE, .6*normtimeToThrow, .05*normtimeToThrow)+
      ifelse(manZone==1, .6, .4)+
    0.6*normPassLength+
    ifelse(dropbackType=="TRADITIONAL", .2,
           ifelse(dropbackType=="SCRAMBLE", .5, ifelse(dropbackType=="DESIGNED_ROLLOUT_RIGHT", .
3,
           ifelse("DESIGNED_ROLLOUT_LEFT", .35,
           ifelse(dropbackType=="SCRAMBLE_ROLLOUT_RIGHT", .5, ifelse(dropbackType=="SCRAMBLE_ROL
LOUT_LEFT", .55, 0))))))+
    ifelse(passLocationType=="INSIDE_BOX", .5, ifelse(passLocationType=="OUTSIDE_LEFT", .7, ifel
se(passLocationType== "OUTSIDE_RIGHT", .7, 0)))+
    ifelse(playAction=="FALSE", .6, ifelse(playAction=="TRUE", .4, 0))



  )

#glimpse(pass_data2)
#colSums(is.na(pass_data2))
pass_data2<- pass_data2%>%mutate(
  normpassDiff=(passDifficulty-mean(passDifficulty,  na.rm = TRUE))/sd(passDifficulty,  na.rm =
TRUE)
)
#print(table(pass_data2$quarter))

## Impute NA
pass_data2 <- pass_data2 %>%
  mutate(
    offenseFormation = ifelse(
      is.na(offenseFormation),
      as.character(names(which.max(table(offenseFormation, useNA = "SHOTGUN")))),
      offenseFormation
    ),
    receiverAlignment = ifelse(
      is.na(receiverAlignment),
      as.character(names(which.max(table(receiverAlignment, useNA = "2x2")))),
      receiverAlignment
    ),
    quarter = ifelse(
      is.na(quarter),
      median(quarter, na.rm = TRUE),
      quarter
    ),
    passDifficulty = ifelse(
      is.na(passDifficulty),
      mean(passDifficulty, na.rm = TRUE),
      passDifficulty
    )
  )
#colSums(is.na(pass_data2))
pass_data2<- pass_data2%>%mutate(
  normpassDiff=(passDifficulty-mean(passDifficulty,  na.rm = TRUE))/sd(passDifficulty,  na.rm =
TRUE)
)

#colSums(is.na(pass_data2))

## Analyze whether there is a direct relationship
explore6<-lm(formula = expectedPointsAdded~normpassDiff, data = pass_data2,)
summary(explore6)
```

```
##
## Call:
## lm(formula = expectedPointsAdded ~ normpassDiff, data = pass_data2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -13.0885  -0.8213  -0.1246   0.9015   8.2145
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.13053    0.01729   7.550 4.80e-14 ***
## normpassDiff  0.12839    0.01729   7.426 1.23e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.572 on 8269 degrees of freedom
## Multiple R-squared:  0.006625,   Adjusted R-squared:  0.006504
## F-statistic: 55.14 on 1 and 8269 DF,  p-value: 1.232e-13
```

```
ggplot(data = pass_data2, mapping = aes(x=normpassDiff, y=expectedPointsAdded))+geom_jitter()+ge
om_smooth(se=FALSE)
```

```
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```



RandomForest 1- Predicting numerical EPA using each individual predictor.

The variable importance plot shows that the most influential feature is the “passDifficulty” statistic. A lot of features like unblockedPressure, passLocationType, dropbackType, offenseFormation, and manZone seem very insignificant towards the predictive power of the model. The residuals are not really normally distributed either as shown in the residual plot. The overall Root Mean Squared Error is around 0.968.

```
predictors <- pass_data[, c("passLength", "passResult", "passLocationType", "dropbackType", "dro
pbackDistance", "timeToThrow", "timeInTackleBox", "offenseFormation", "receiverAlignment", "quar
ter", "unblockedPressure", "pff_passCoverage", "pff_manZone")]  # Select predictor columns
response <- pass_data$expectedPointsAdded  # Define the response variable
#rf_model <- randomForest(x = predictors, y =plays$response, )

#ggplot(data = plays, mapping = aes(x = yardsGained, y = expectedPointsAdded, colour = offenseFo
rmation))+geom_point()+geom_smooth(se=FALSE)

splitIndex <- createDataPartition(pass_data$expectedPointsAdded, p = 0.8, list = FALSE)
train_data <- pass_data[splitIndex, c("passLength", "passResult", "passLocationType", "dropbackT
ype",
                                      "dropbackDistance", "timeToThrow", "timeInTackleBox", "off
enseFormation",
                                      "receiverAlignment", "quarter", "unblockedPressure", "pff_
passCoverage", "pff_manZone")]  # Select predictors for training

train_out <- pass_data$expectedPointsAdded[splitIndex]

#glimpse(train_data)
#glimpse(train_out)
#glimpse(train_data)
test_data <- pass_data[-splitIndex, c("passLength", "passResult", "passLocationType", "dropbackT
ype",
                                      "dropbackDistance", "timeToThrow", "timeInTackleBox", "off
enseFormation",
                                      "receiverAlignment", "quarter", "unblockedPressure", "pff_
passCoverage", "pff_manZone")]

test_out <- pass_data$expectedPointsAdded[-splitIndex]


dim(train_data)
```

```
## [1] 6619   13
```

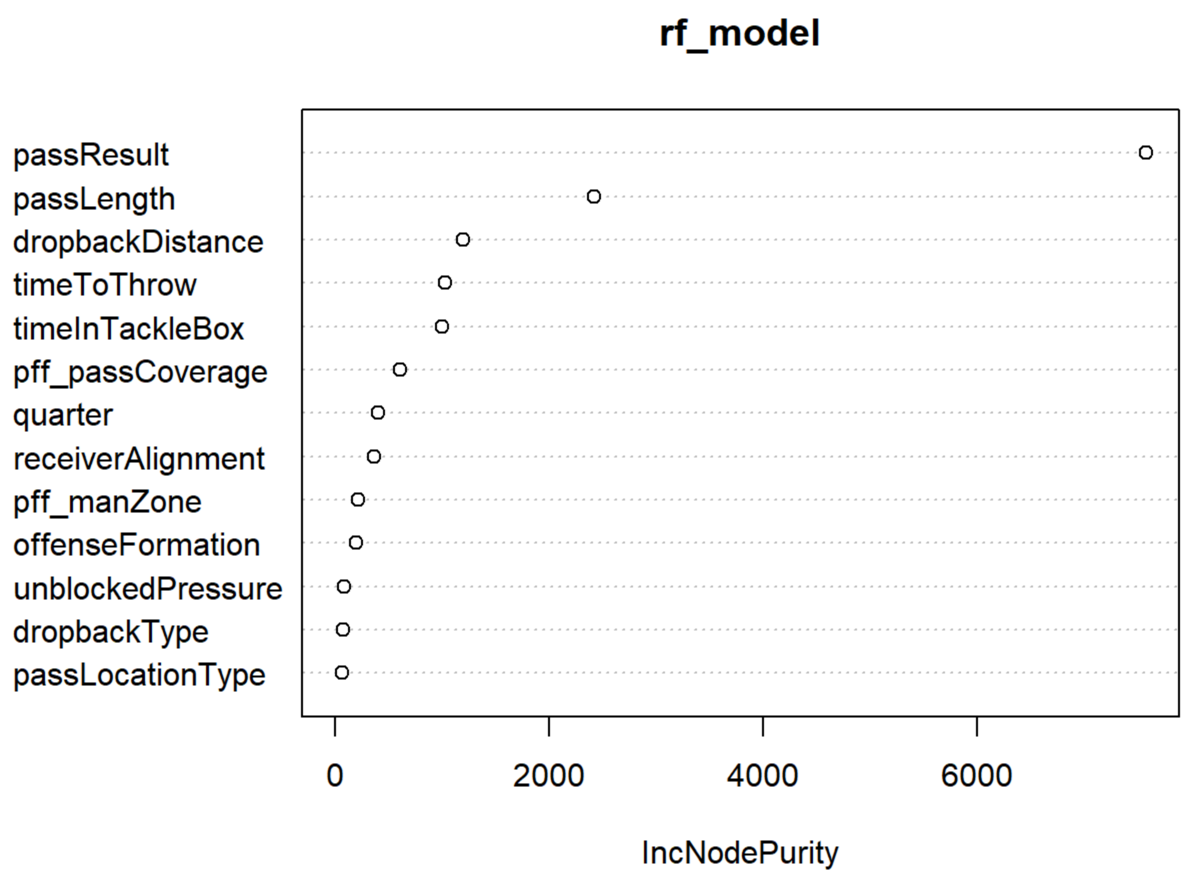
```
dim(test_data)
```

```
## [1] 1652   13
```

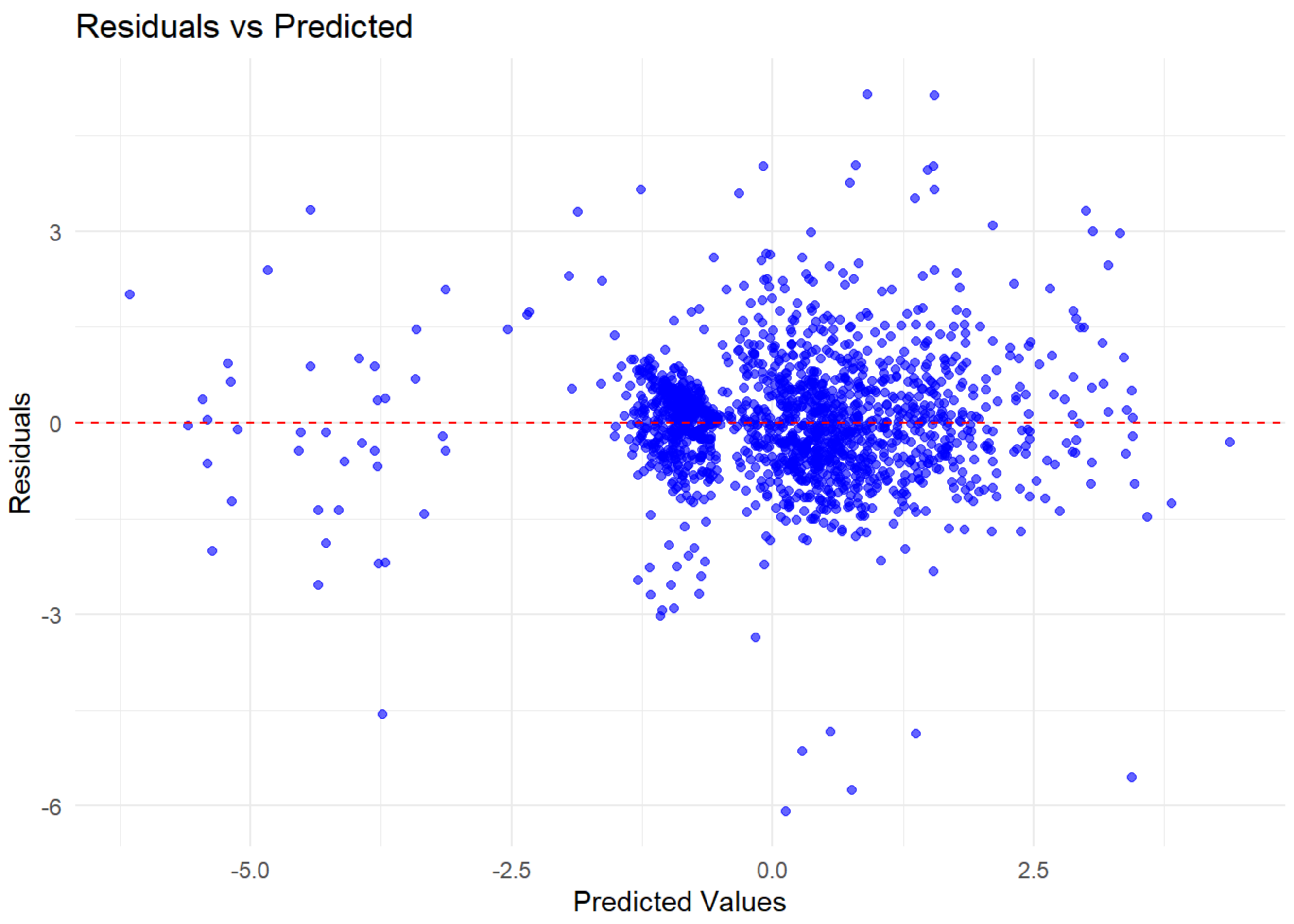
```
rf_model <- randomForest(x = train_data, y =train_out, proximity = TRUE)
Y_pred = predict(rf_model, test_data, type="response")
 rf_err = sqrt( mean( (Y_pred - test_out)^2 ))
print(rf_err)
```

```
## [1] 0.9678091
```

```
varImpPlot(rf_model)
```



```
residuals_data <- data.frame(
  Predicted = Y_pred,
  Residuals = test_out - Y_pred
)

# Plot using ggplot
ggplot(residuals_data, aes(x = Predicted, y = Residuals)) +
  geom_point(color = "blue", alpha = 0.6) +
  geom_hline(yintercept = 0, color = "red", linetype = "dashed") +
  labs(
    title = "Residuals vs Predicted",
    x = "Predicted Values",
    y = "Residuals"
  ) +
  theme_minimal()
```



RandomForest 2- Predicting numerical EPA using the normPassDifficulty statistic.

We randomly select the training and test data with a 80/20 split and run another randomForest algorithm. The variable importance plot shows that the most influential feature is the "passDifficulty" statistic. The residuals are not really normally distributed with a overall Root Mean Squared Error of around 1.55. The error is heavily right skewed as shown in a the histogram of the Root Mean Squared Error.

```
splitIndex <- createDataPartition(pass_data2$expectedPointsAdded, p = 0.8, list = FALSE)

train_data2 <- pass_data2[splitIndex, c("offenseFormation", "receiverAlignment", "quarter", "nor
mpassDiff")]
test_data2 <- pass_data2[-splitIndex, c("offenseFormation", "receiverAlignment", "quarter", "nor
mpassDiff")]
train_out2 <- pass_data2$expectedPointsAdded[splitIndex]
test_out2 <- pass_data2$expectedPointsAdded[-splitIndex]

# Verify dimensions of training and testing datasets
cat("\nTrain set dimensions (predictors only):\n")
```

```
##
## Train set dimensions (predictors only):
```

```
print(dim(train_data))
```

```
## [1] 6619   13
```

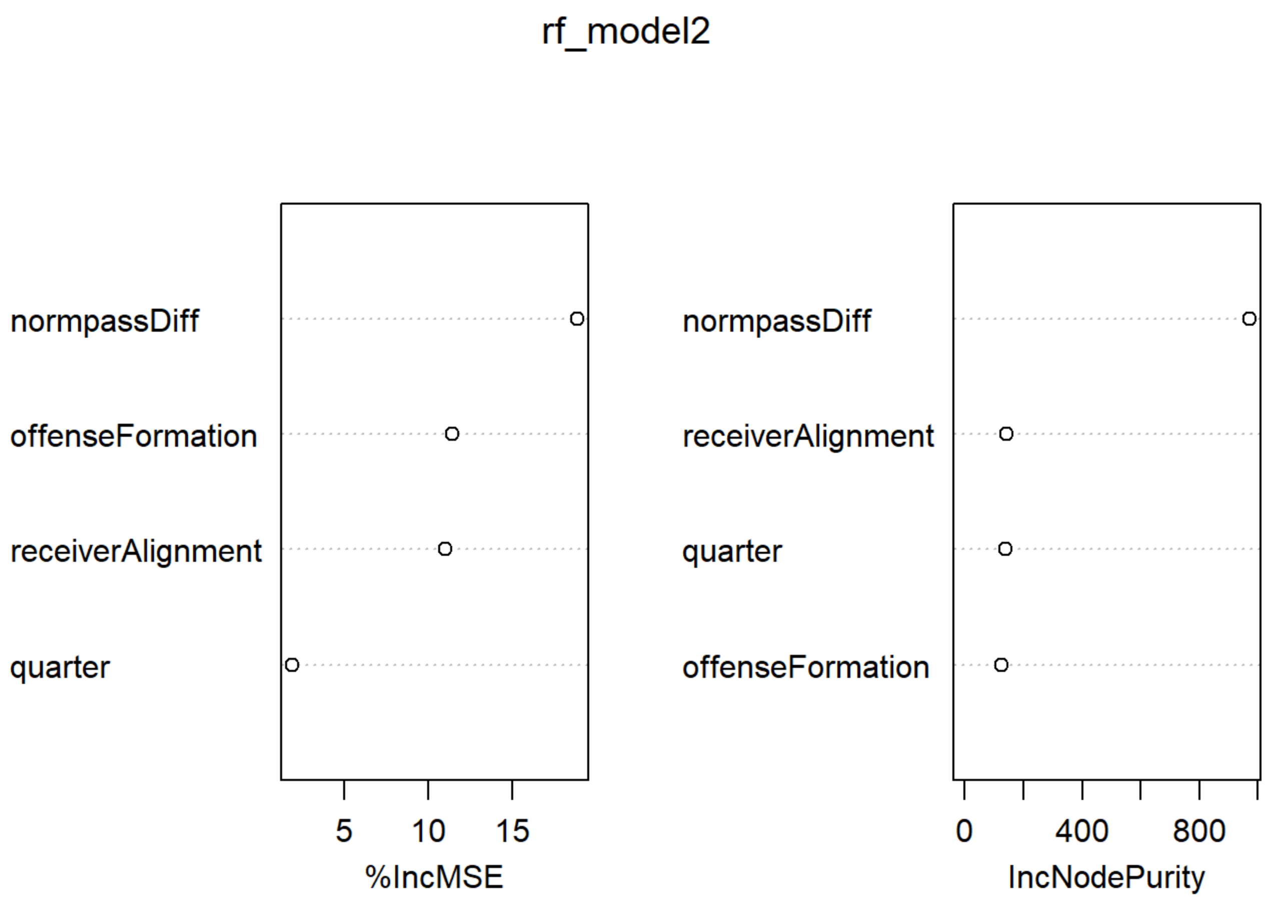
```
cat("\nTest set dimensions (predictors only):\n")
```

```
##
## Test set dimensions (predictors only):
```

```
print(dim(test_data))
```

```
## [1] 1652   13
```

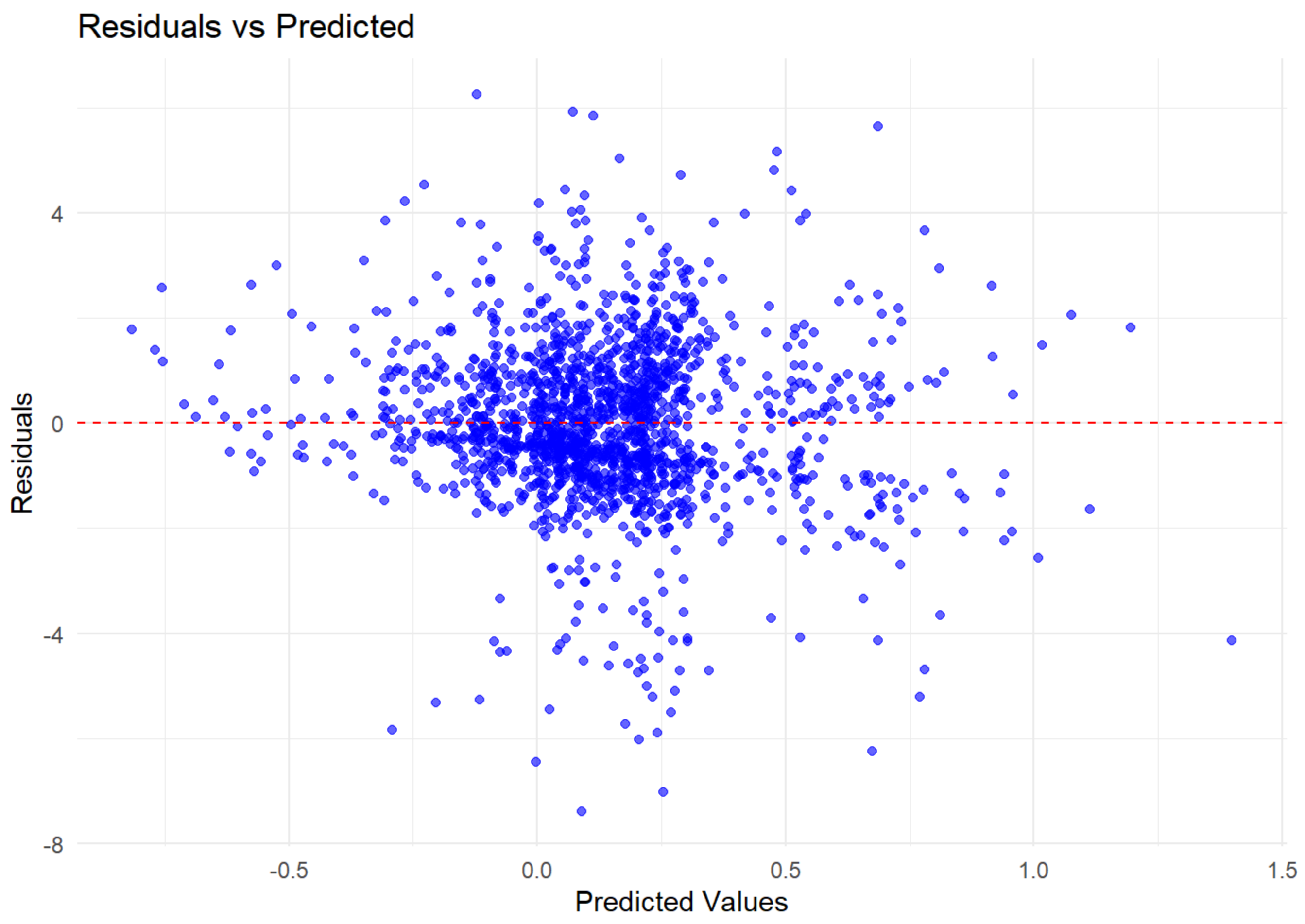
```
# Build the random forest model

rf_model2 <- randomForest(
  x = train_data2,
  y = train_out2,
  proximity = TRUE,
  importance = TRUE
)
varImpPlot(rf_model2)
```



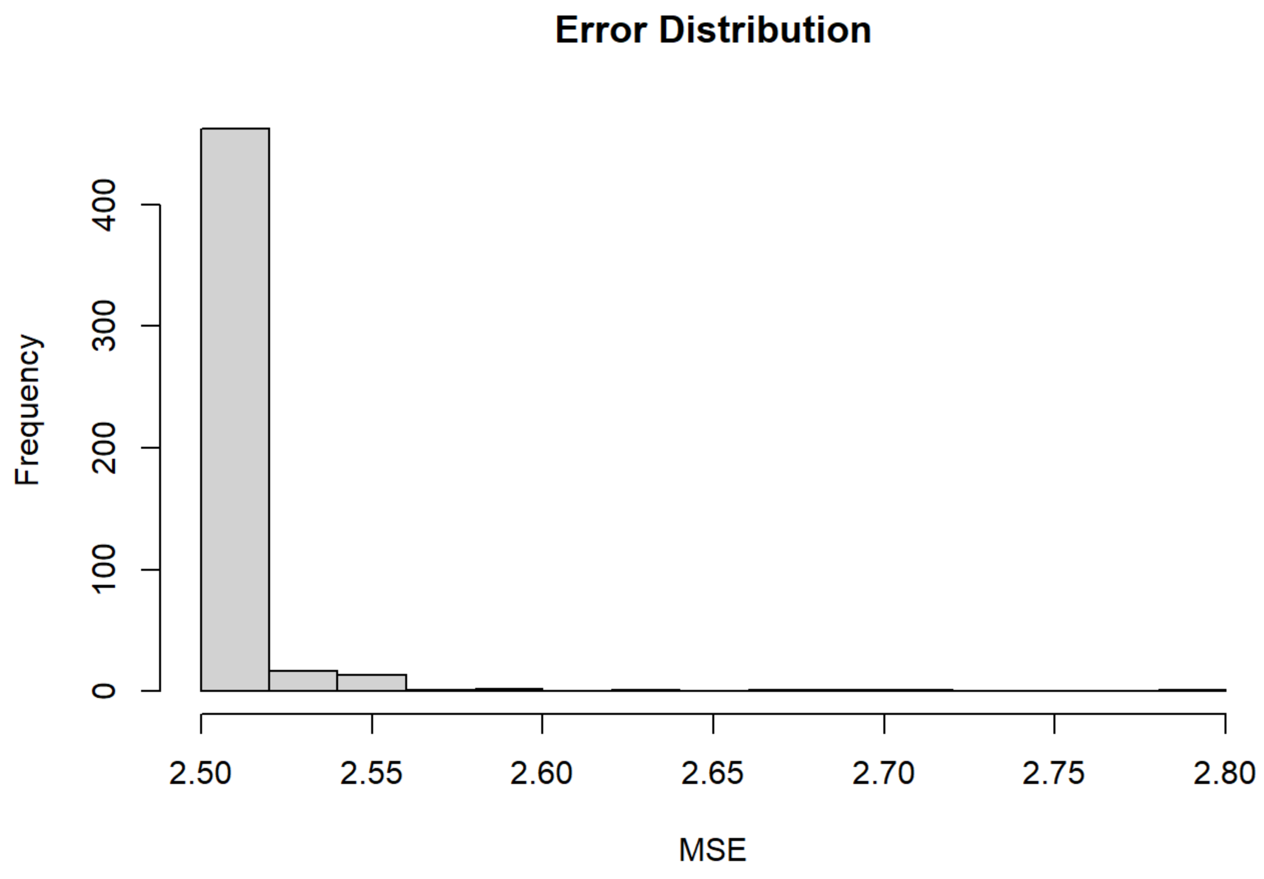
```
Y_pred2 <- predict(rf_model2, newdata = test_data2)
rf_err2 <- sqrt(mean((Y_pred2 - test_out2)^2))
print(rf_err2)
```

```
## [1] 1.549537
```

```
 #colSums(is.na(pass_data2))
residuals_data2 <- data.frame(
  Predicted = Y_pred2,
  Residuals = test_out2 - Y_pred2
)

# Plot using ggplot
ggplot(residuals_data2, aes(x = Predicted, y = Residuals)) +
  geom_point(color = "blue", alpha = 0.6) +
  geom_hline(yintercept = 0, color = "red", linetype = "dashed") +
  labs(
    title = "Residuals vs Predicted",
    x = "Predicted Values",
    y = "Residuals"
  ) +
  theme_minimal()
```



```
## MSE distribution
hist(rf_model2$mse, main = "Error Distribution", xlab = "MSE")
```



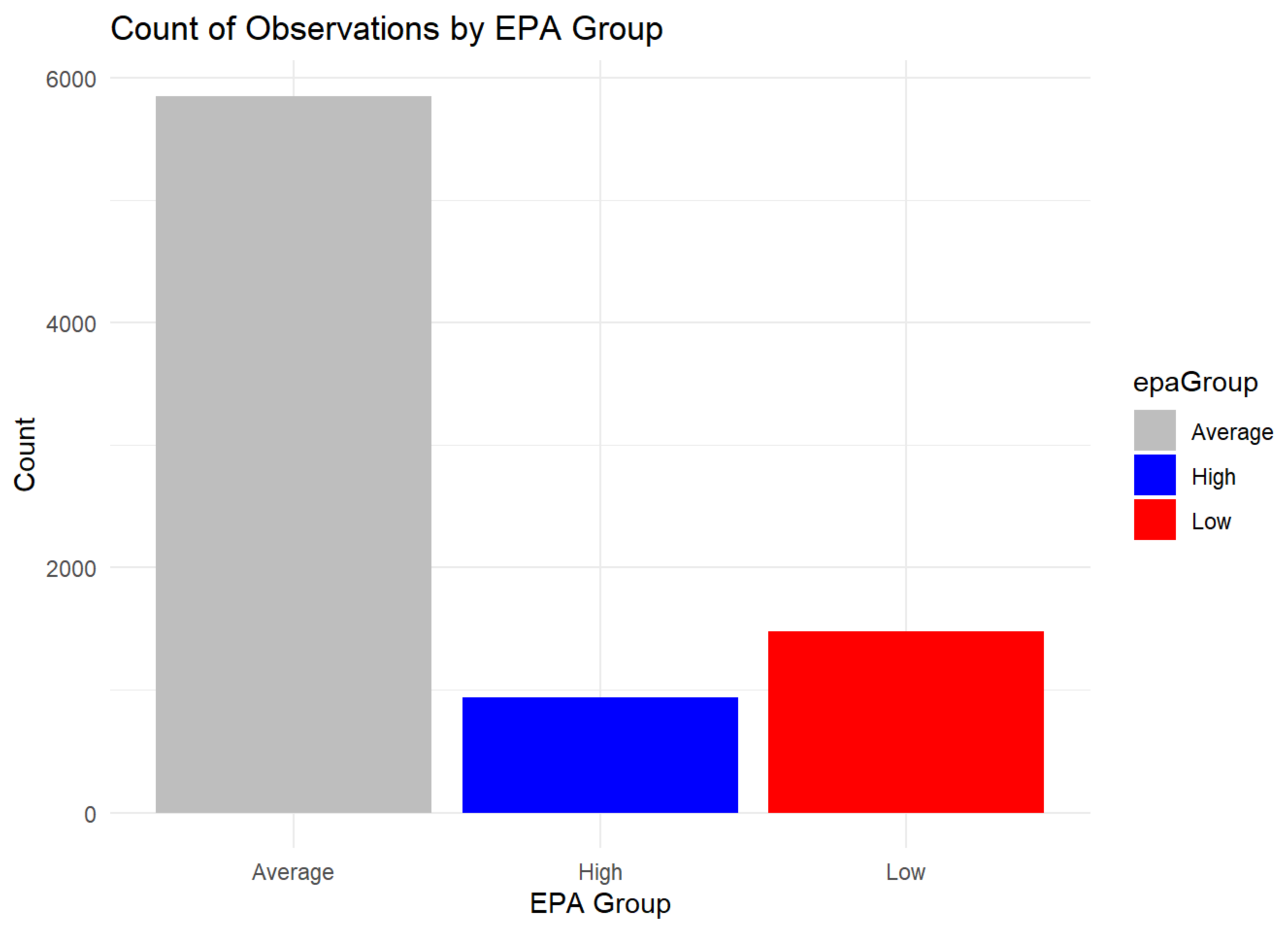
RandomForest 3- Predicting categorizations of EPA using the same features.

I found the Q1 and Q3 of EPA in the filtered pass data and assigned anything under Q1 being a "Low" EPA and anything above is Q3. These categories are distributed as shown in the bargraph. We randomly select the training and test data with a 80/20 split and run another randomForest algorithm. The model has a accuracy of 0.7036. The variable importance plot shows that the most influential feature is the "passDifficulty" statistic. The model performs very well at identifying "Average" plays but struggles significantly with distinguishing "High" and "Low" plays. There is high sensitivity for "Average", but low sensitivity for "Low" and "High" categorized plays.

```
## EPA categories

pass_data2%>% summarise(
  mean = mean(expectedPointsAdded, na.rm = TRUE),
  median = median(expectedPointsAdded, na.rm = TRUE),
  sd = sd(expectedPointsAdded, na.rm = TRUE),
  min = min(expectedPointsAdded, na.rm = TRUE),
  max = max(expectedPointsAdded, na.rm = TRUE),
  q1 = quantile(expectedPointsAdded, 0.25, na.rm = TRUE),
  q3 = quantile(expectedPointsAdded, 0.75, na.rm = TRUE)
)
```

```
##         mean      median       sd      min      max         q1       q3
## 1 0.1305321 -0.05681065 1.577408 -13.0236 8.698986 -0.6560297 1.016368
```

```
pass_data3<- pass_data2%>%mutate(
  epaGroup= ifelse(expectedPointsAdded>=1.911744, "High", ifelse(expectedPointsAdded<=-0.920628,
"Low", "Average"))
)

ggplot(pass_data3, aes(x = epaGroup, fill = epaGroup)) +
  geom_bar() +
  scale_fill_manual(values = c("High" = "blue", "Low" = "red", "Average" = "gray")) +
  labs(
    title = "Count of Observations by EPA Group",
    x = "EPA Group",
    y = "Count"
  ) +
  theme_minimal()
```



```
splitIndex <- createDataPartition(pass_data3$epaGroup, p = 0.8, list = FALSE)

train_data3 <- pass_data3[splitIndex, c("offenseFormation", "receiverAlignment", "quarter", "nor
mpassDiff")]
test_data3 <- pass_data3[-splitIndex, c("offenseFormation", "receiverAlignment", "quarter", "nor
mpassDiff")]
train_out3 <- pass_data3$epaGroup[splitIndex]
test_out3 <- pass_data3$epaGroup[-splitIndex]
train_out3 <- as.factor(train_out3)
test_out3 <- as.factor(test_out3)

# Verify dimensions of training and testing datasets
cat("\nTrain set dimensions (predictors only):\n")
```

```
##
## Train set dimensions (predictors only):
```

```
print(dim(train_data3))
```

```
## [1] 6618    4
```

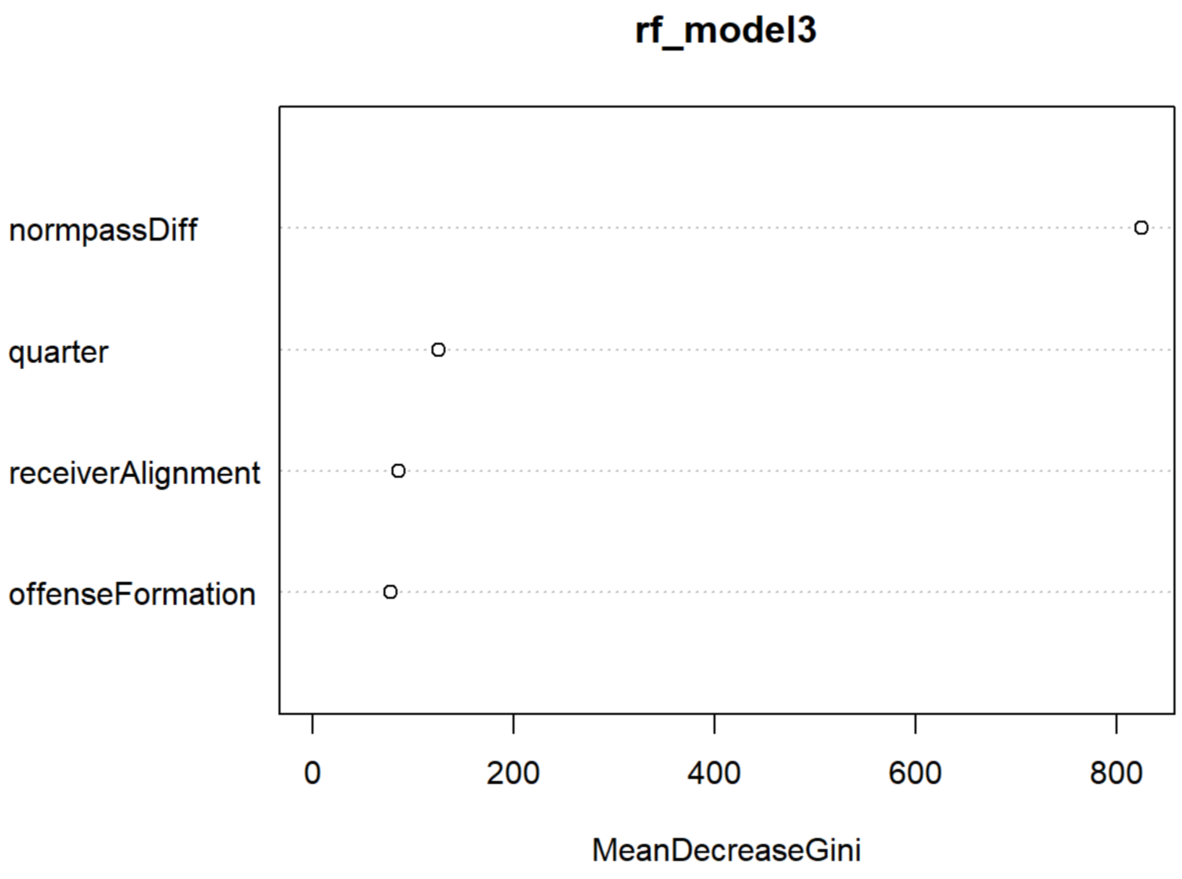
```
cat("\nTest set dimensions (predictors only):\n")
```

```
##
## Test set dimensions (predictors only):
```

```
print(dim(test_data3))
```

```
## [1] 1653    4
```

```
# Build the random forest model

rf_model3 <- randomForest(x = train_data3, y = train_out3)
Y_pred3<- predict(rf_model3, test_data3)
varImpPlot(rf_model3)
```



```
confusion_matrix <- table(Predicted = Y_pred3, Actual = test_out3)
confusionMatrix(confusion_matrix)
```

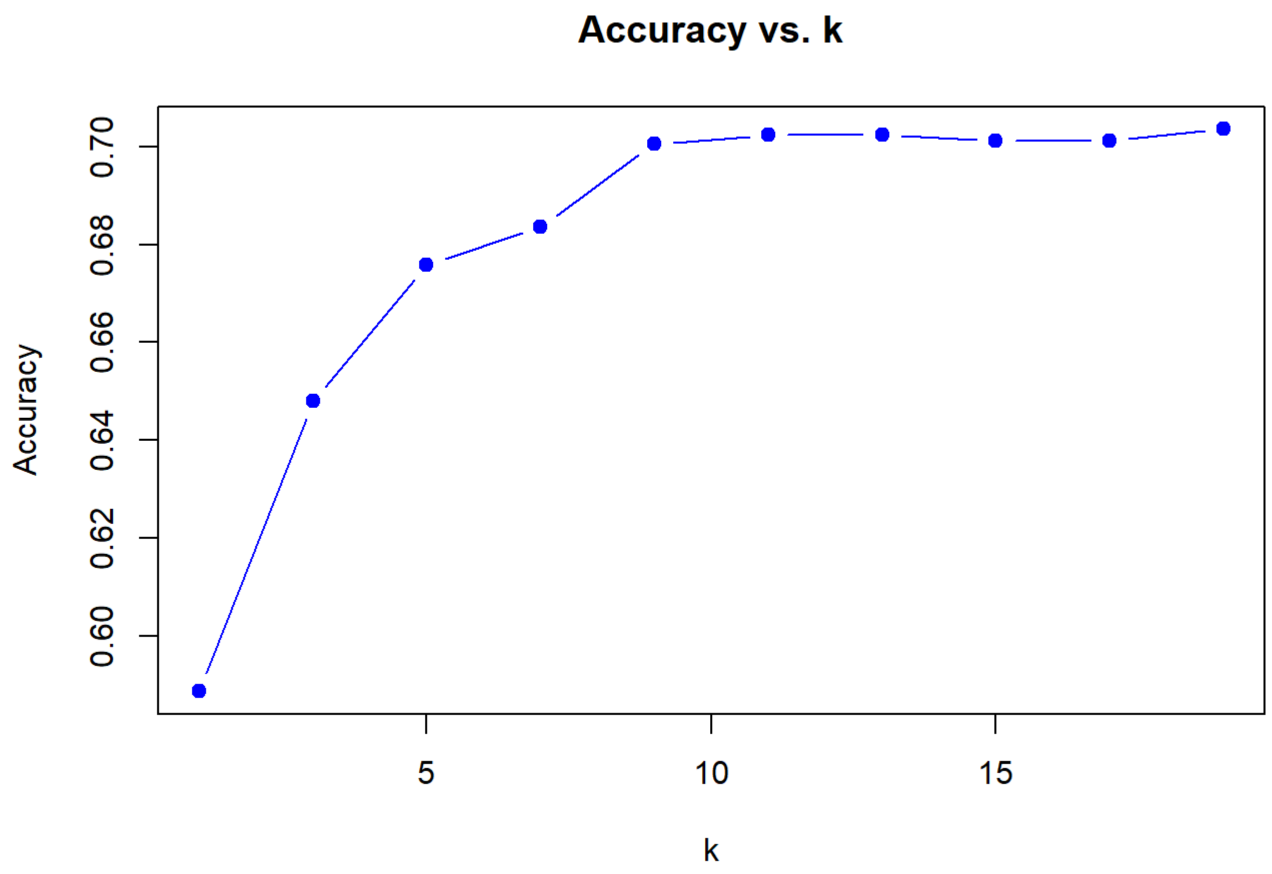
```
## Confusion Matrix and Statistics
##
##          Actual
## Predicted Average High  Low
##   Average    1139  172  274
##   High         17   10    7
##   Low          14    6   14
##
## Overall Statistics
##
##                Accuracy : 0.7036
##                  95% CI : (0.6809, 0.7255)
##     No Information Rate : 0.7078
##     P-Value [Acc > NIR] : 0.6586
##
##                   Kappa : 0.0599
##
##  Mcnemar's Test P-Value : <2e-16
##
## Statistics by Class:
##
##                      Class: Average Class: High Class: Low
## Sensitivity                  0.9735     0.05319   0.047458
## Specificity                  0.0766     0.98362   0.985272
## Pos Pred Value               0.7186     0.29412   0.411765
## Neg Pred Value               0.5441     0.89006   0.826436
## Prevalence                   0.7078     0.11373   0.178463
## Detection Rate               0.6891     0.00605   0.008469
## Detection Prevalence         0.9589     0.02057   0.020569
## Balanced Accuracy            0.5251     0.51840   0.516365
```

KNN Algorithm

KNN model was used to predict EPA categories using the same categorical variables as random forest: offenseFormation, receiverAlignment, and quarter. All predictor variables were scaled to standardize their range. This step ensures that no variable dominates due to its scale. This is important to KNN as it is distance based and is sensitive to differing magnitudes. The epaGroup, response variable, was converted into numeric values to be compatible with the KNN algorithm. Odd k values from a range of 1-20 were tested to determine optimal k. Accuracy for each k was calculated on the validation set, and the results were plotted to identify when the accuracy of k falls off.

The optimal K was 19 and this was found by maximizing accuracy. The optimized accuracy we found in the confusion matrix was 0.7011 but the Baseline accuracy is 0.7024 C1- Average, C2- High, C3-Low. The model performs well in identifying Class 1 instances, as seen in its high sensitivity. Class 2 and Class 3 have lower sensitivity, indicating that the model struggles to correctly classify these observations. The overall accuracy is close to the baseline No Information Rate, suggesting limited improvement in performance.

```
#library(caret)


# Ensure all predictors are numeric and scale them
knn_train_data <- train_data3 %>%
  mutate(
    offenseFormation = as.numeric(factor(offenseFormation)),
    receiverAlignment = as.numeric(factor(receiverAlignment)),
    quarter = as.numeric(quarter)
  ) %>%
  scale() %>%
  as.data.frame()

knn_test_data <- test_data3 %>%
  mutate(
    offenseFormation = as.numeric(factor(offenseFormation)),
    receiverAlignment = as.numeric(factor(receiverAlignment)),
    quarter = as.numeric(quarter)
  ) %>%
  scale() %>%
  as.data.frame()

# Convert output variable to numeric factors for KNN
train_out_knn <- as.numeric(factor(train_out3)) # Convert to numeric
test_out_knn <- as.numeric(factor(test_out3))   # Convert to numeric

# Find the optimal value of k using cross-validation
k_values <- seq(1, 20, by = 2) # Test odd k values
accuracy <- sapply(k_values, function(k) {
  pred <- knn(train = knn_train_data, test = knn_test_data, cl = train_out_knn, k = k)
  mean(pred == test_out_knn) # Calculate accuracy
})

# Plot accuracy vs. k
plot(
  k_values, accuracy, type = "b", col = "blue", pch = 19,
  xlab = "k", ylab = "Accuracy", main = "Accuracy vs. k"
)
```



```
# Select the optimal k
optimal_k <- k_values[which.max(accuracy)]
cat("Optimal k:", optimal_k, "\n")
```

```
## Optimal k: 19
```

```
# Train the final KNN model with optimal k
knn_predictions <- knn(train = knn_train_data, test = knn_test_data, cl = train_out_knn, k = opt
imal_k)

# Evaluate the KNN model
confusion_matrix <- table(Predicted = knn_predictions, Actual = test_out_knn)
confusionMatrix(confusion_matrix)
```

```
## Confusion Matrix and Statistics
##
##          Actual
## Predicted    1    2    3
##         1 1131  150  255
##         2   17   12   22
##         3   22   26   18
##
## Overall Statistics
##
##                Accuracy : 0.7024
##                  95% CI : (0.6797, 0.7243)
##     No Information Rate : 0.7078
##     P-Value [Acc > NIR] : 0.6973
##
##                   Kappa : 0.1026
##
##  Mcnemar's Test P-Value : <2e-16
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3
## Sensitivity            0.9667  0.06383  0.06102
## Specificity            0.1615  0.97338  0.96465
## Pos Pred Value         0.7363  0.23529  0.27273
## Neg Pred Value         0.6667  0.89014  0.82546
## Prevalence             0.7078  0.11373  0.17846
## Detection Rate         0.6842  0.00726  0.01089
## Detection Prevalence   0.9292  0.03085  0.03993
## Balanced Accuracy      0.5641  0.51860  0.51284
```

Findings and Future Directions:

When merging multiple aspects of the passing, the normpassDiff was the most critical predictor for EPA. Quarter had less influence but still provided context. Predicting the numerical values with the individual features also produces a similar error as the other randomForest models and KNN model.

The randomForest3 model had a very similar performance to the KNN model, both hover around 70% accuracy. It is clear that the “Average” EPA categorization dominated in the dataset, as both model’s confusion matrix showed similar specificity values.

For future directions, perhaps choosing more representative cutoffs for EPA groupings could lead to more robust predictions as well as incoorperating more speciic data such as individual player’s statistics in the corresponding player_plays.csv dataset. With more variables, perhaps you could find stronger correlations within the data to make better predictions.