Effect of Fifa Player Ratings on Real Life Wages

Alan Qin

04/15/2020

Contents

Project Introduction

My project is about players in the game FIFA 20 and how wages are affected by rating. This is interesting to me because I want to see how the best soccer players' wages are affected by their FIFA rating. The source of the data is from Kaggle.com, specifically (https://www.kaggle.com/stefanoleone992/fifa-20-complete-player-dataset). The Y (outcome) in this data are the wages of the players while the X's are the ratings of the player. There are well over 50 X's in this dataset but I have chosen the 17 most important X's such as overall, potential, age, and others. In this project, I am using different techniques to find out if professional wages are affected by FIFA rating.

These variables are

- **wage_eur**: log(wage) in EUR of the player
- **value_eur**: value in EUR of the player
- **release_clause_eur**: release clause in euros
- **overall**: overall attribute of the player, based on all of the player stats
- **potential**: potential attribute of the player, based on age and other factors
- **age**: age of the player
- **pace**: player pace rating, based on sprint speed and acceleration
- **shooting**: player shooting rating, based on finishing, positioning, shot power, long shots, volley, and penalties
- **passing**: player passing rating, based on vision, crossing, fk. accuracy, short and long passing, and curve
- **dribbling**: player dribbling rating, based on agility, balance, reactions, ball control, dribbling, composure
- **defending**: player defending rating, based on interceptions, heading accuracy, def. awareness, tackling
- **physic**: player physical rating, based on jumping, stamina, strength, and agression
- **height_cm**: height of a player in centimeters
- **weight_kg**: weight of a player in kilograms
- **international_reputation**: international repulation attribute
- **weak_foot**: weak foot rating from 1-5 stars
- **skill_moves**: skill move rating from 1-5 stars

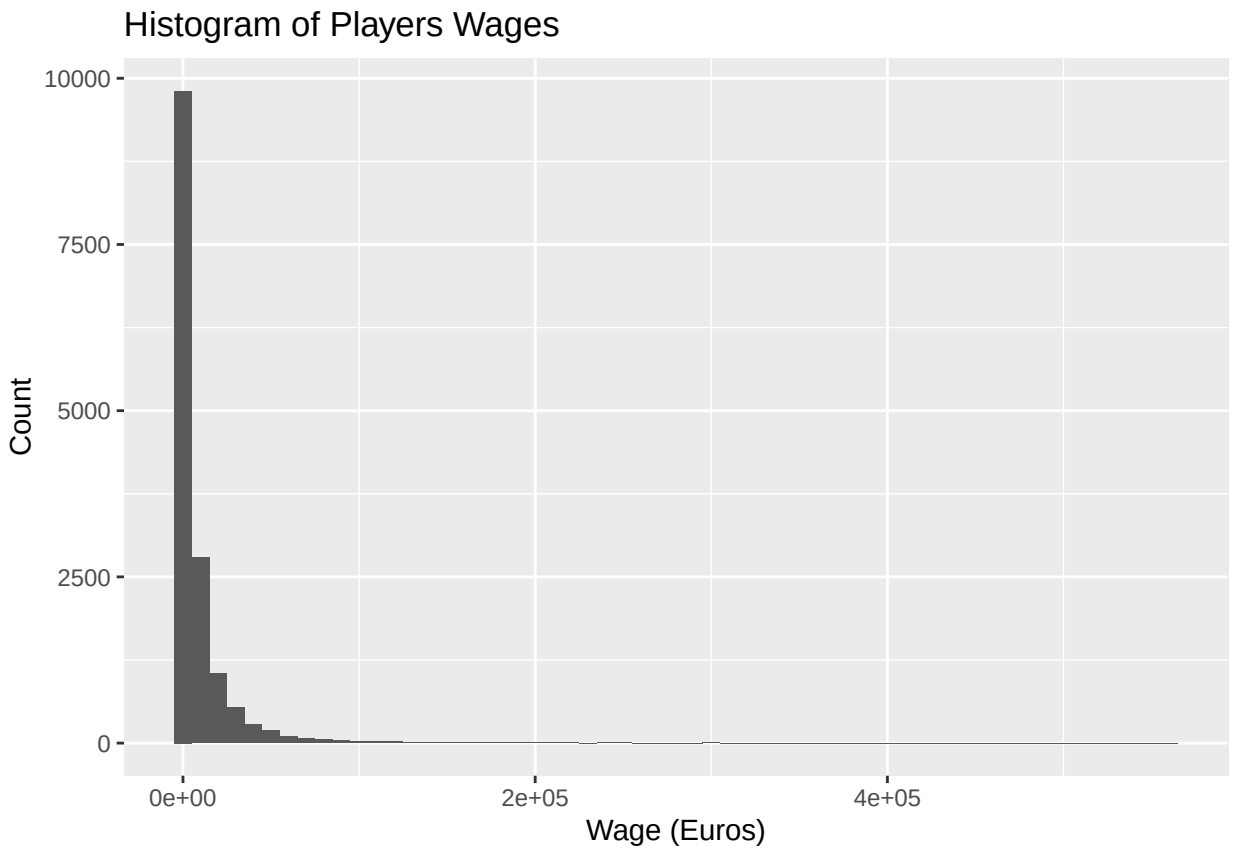
Fixing Data

```
newData = as_tibble(data)
data1 = newData %>%
  dplyr::select(wage_eur, age, height_cm, weight_kg, overall,
         potential, value_eur, international_reputation,
         weak_foot, skill_moves, release_clause_eur, pace,
         shooting, passing, dribbling, defending, physic)
dataOmit = na.omit(data1)
fifa_data = dataOmit
```

Report 1

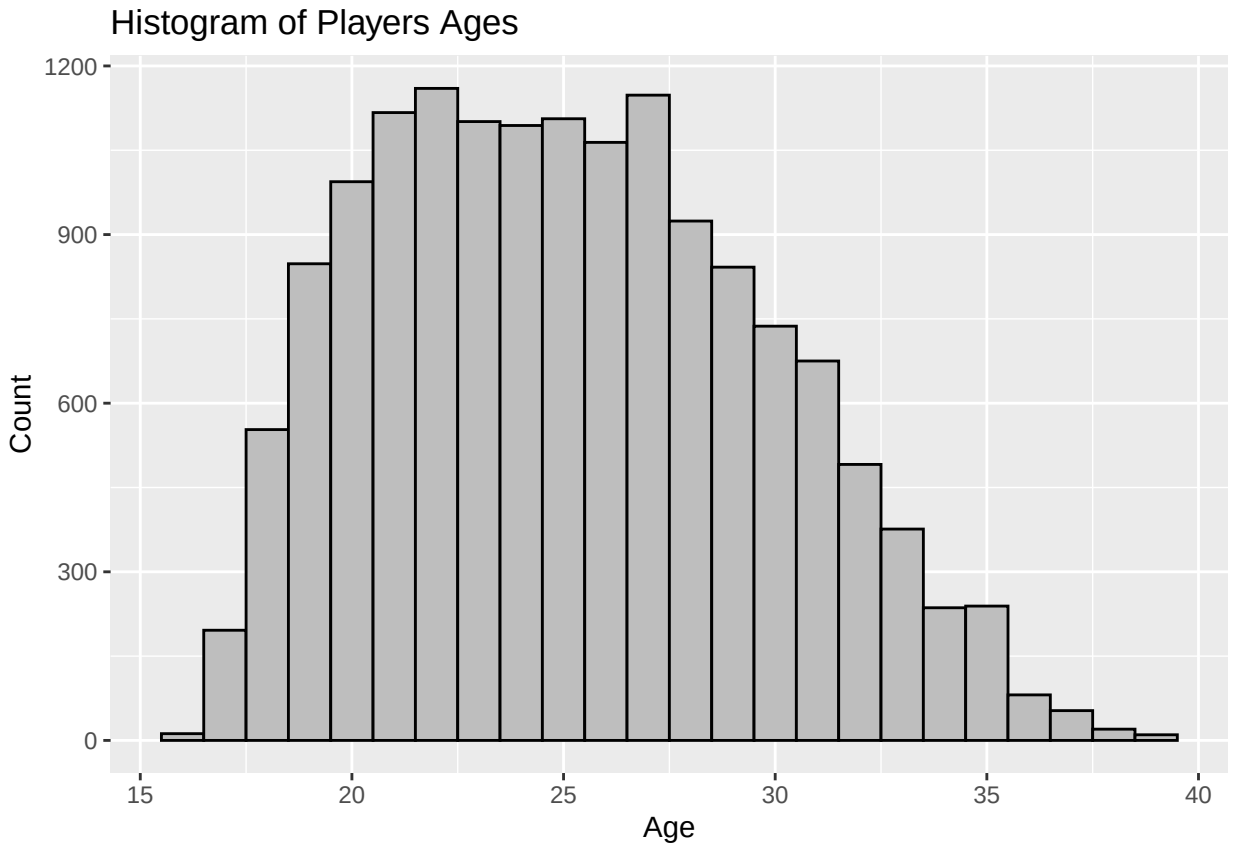
Histogram for my X's and Y's

```
ggplot(data = fifa_data, aes(x = wage_eur)) +
  geom_histogram(bins = 1000, binwidth = 10000) +
  ggtitle('Histogram of Players Wages') +
  labs(x = 'Wage (Euros)', y = "Count")
```



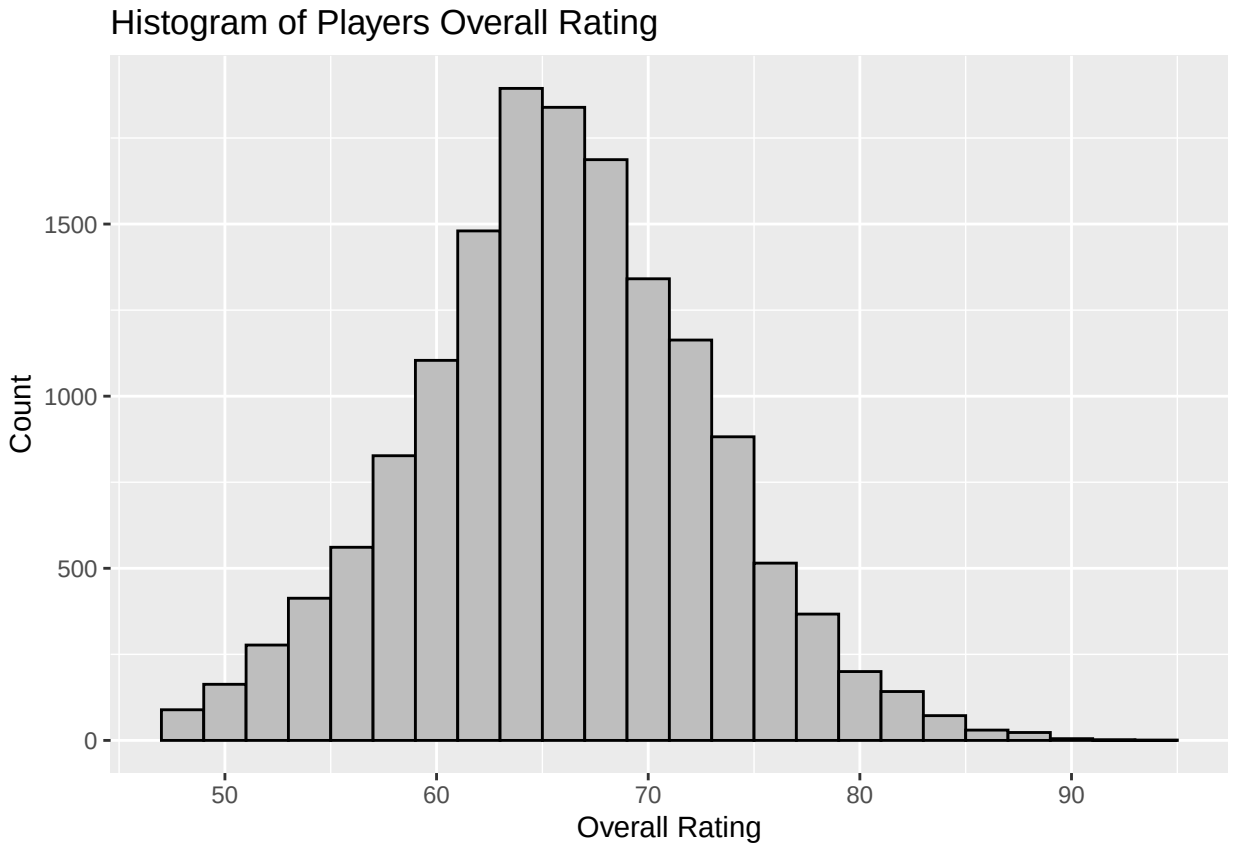
This histogram shows the wages of all the players in fifa. As you can see, there are around 4000 players with wages of around 0. As we go up in wage, there are less and less players with one player (Lionel Messi) making around log(13.4) which is about 500,000 euros a week.

```r
ggplot(data = fifa_data, aes(x = age)) +
  geom_histogram(binwidth = 1, color = 'black', fill = 'gray') +
  ggtitle('Histogram of Players Ages') +
  labs(x = 'Age', y = "Count")
```



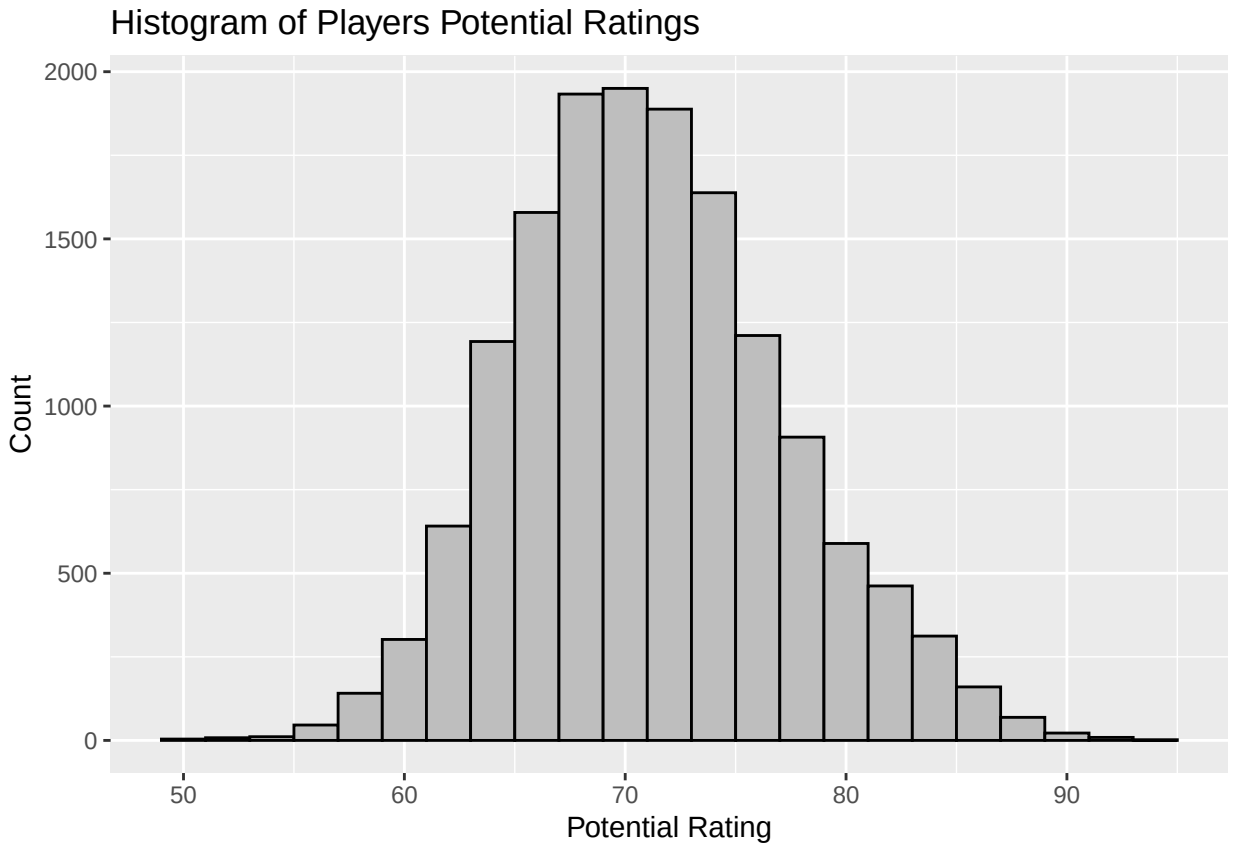
As we can see from this histogram, there are many players within the age range of 20-29 while there are few players younger than 18 and older than 35. There are the most players within the age range of 20-29 because that is their physical peak.

```r
ggplot(data = fifa_data, aes(x = overall)) +
  geom_histogram(binwidth = 2, color = 'black', fill = 'gray') +
  ggtitle('Histogram of Players Overall Rating') +
  labs(x = 'Overall Rating', y = "Count")
```



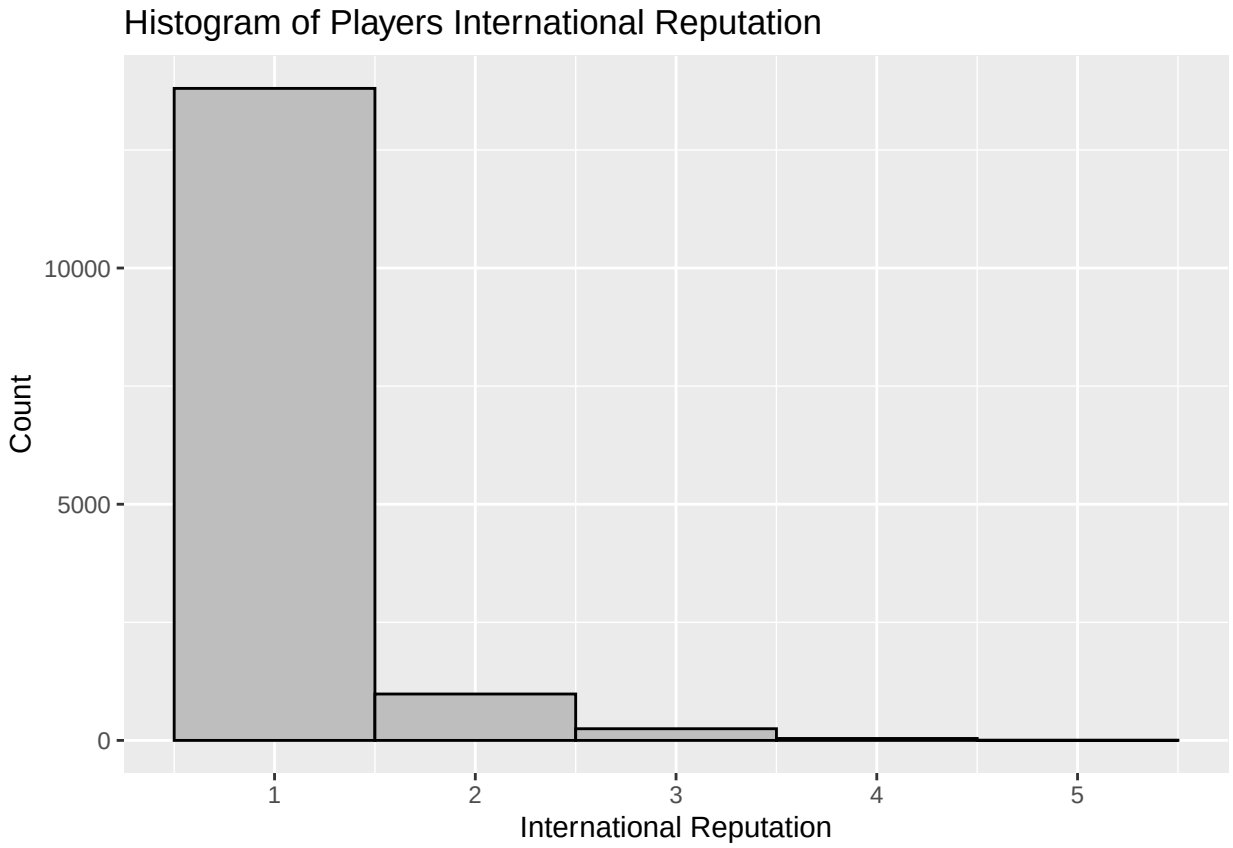
As we can see from this histogram, the mode of the overall rating of a player in FIFA 20 is around 65 points. The histogram also looks like it is normally distributed.

```r
ggplot(data = fifa_data, aes(x = potential)) +
  geom_histogram(binwidth = 2, color = 'black', fill = 'gray') +
  ggtitle('Histogram of Players Potential Ratings') +
  labs(x = 'Potential Rating', y = "Count")
```



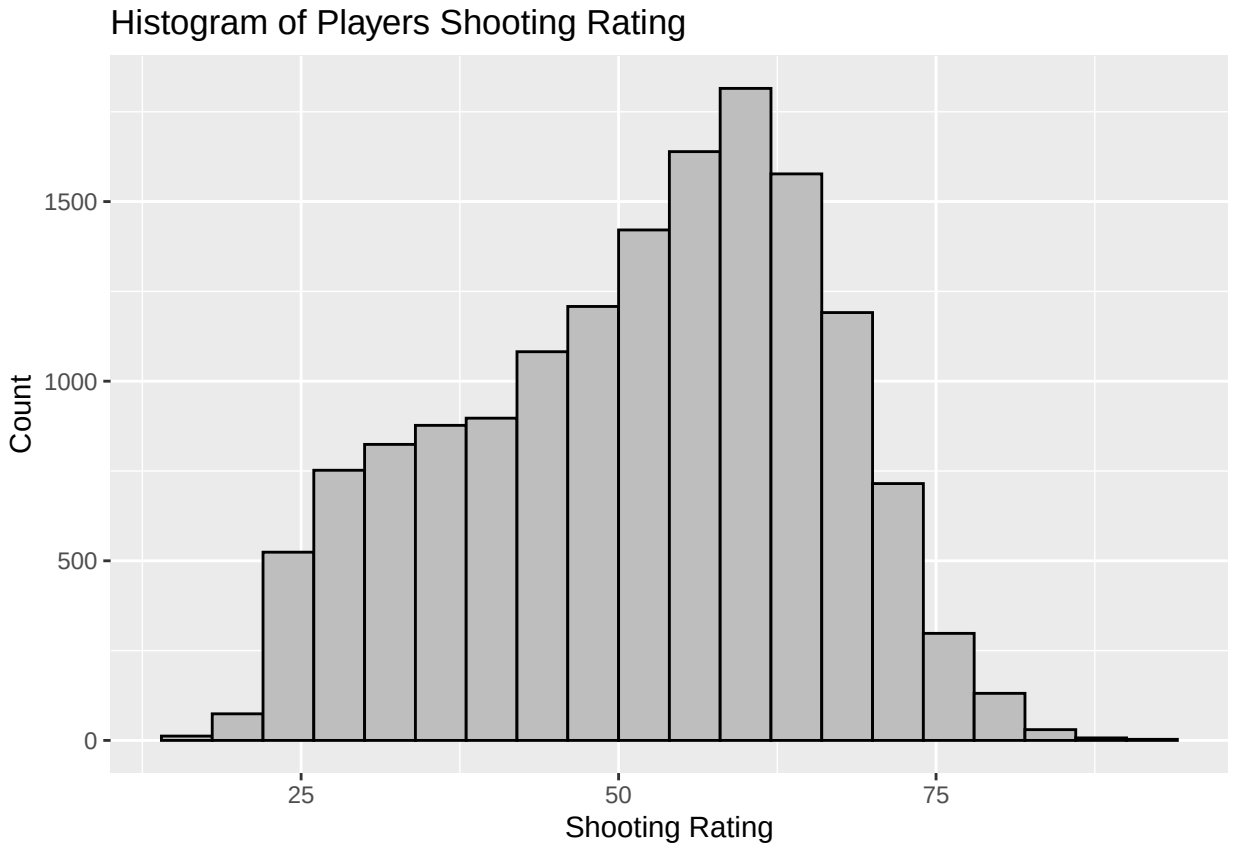
Simliarly to the Overall Rating Histogram, the potential ratings of players are also normally distributed. However, the average potential rating of FIFA players increased to around 70 points instead of 65.

```r
ggplot(data = fifa_data, aes(x = international_reputation)) +
  geom_histogram(binwidth = 1, color = 'black', fill = 'gray') +
  ggtitle('Histogram of Players International Reputation') +
  labs(x = 'International Reputation', y = "Count")
```



This histogram shows that there are many players with an international reputation of 1 with less than 2000 players with a rating above. I think this is the case becuase international repuation is based off of how well the player plays for their national team. Since there are only so many spots on a national team it makes sense that the vast majority of players have an international reputation of 1.

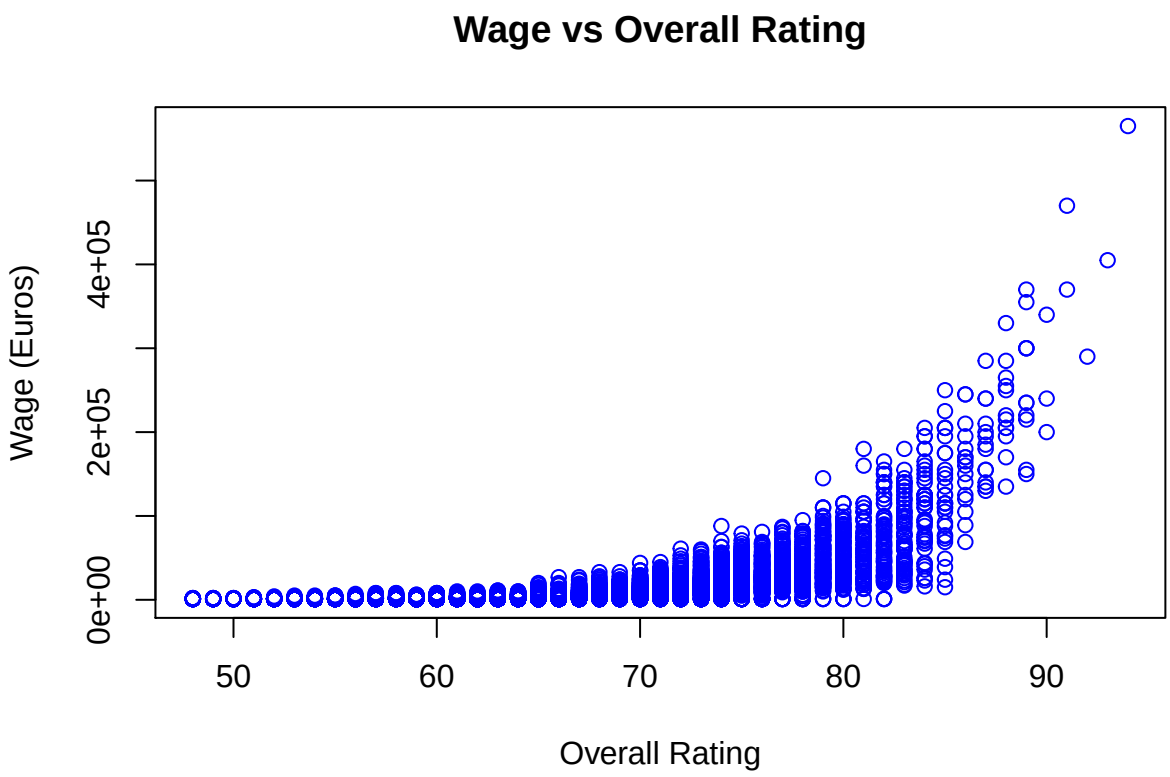
```r
ggplot(data = fifa_data, aes(x = shooting)) +
  geom_histogram(binwidth = 4, color = 'black', fill = 'gray') +
  ggtitle('Histogram of Players Shooting Rating') +
  labs(x = 'Shooting Rating', y = "Count")
```



As we can see from this histogram of players' shooting rating, the shooting ratings look negatively skewed as it appears there are more players with higher shooting rating than not. The mode of the histogram is around 60-65 shooting rating.

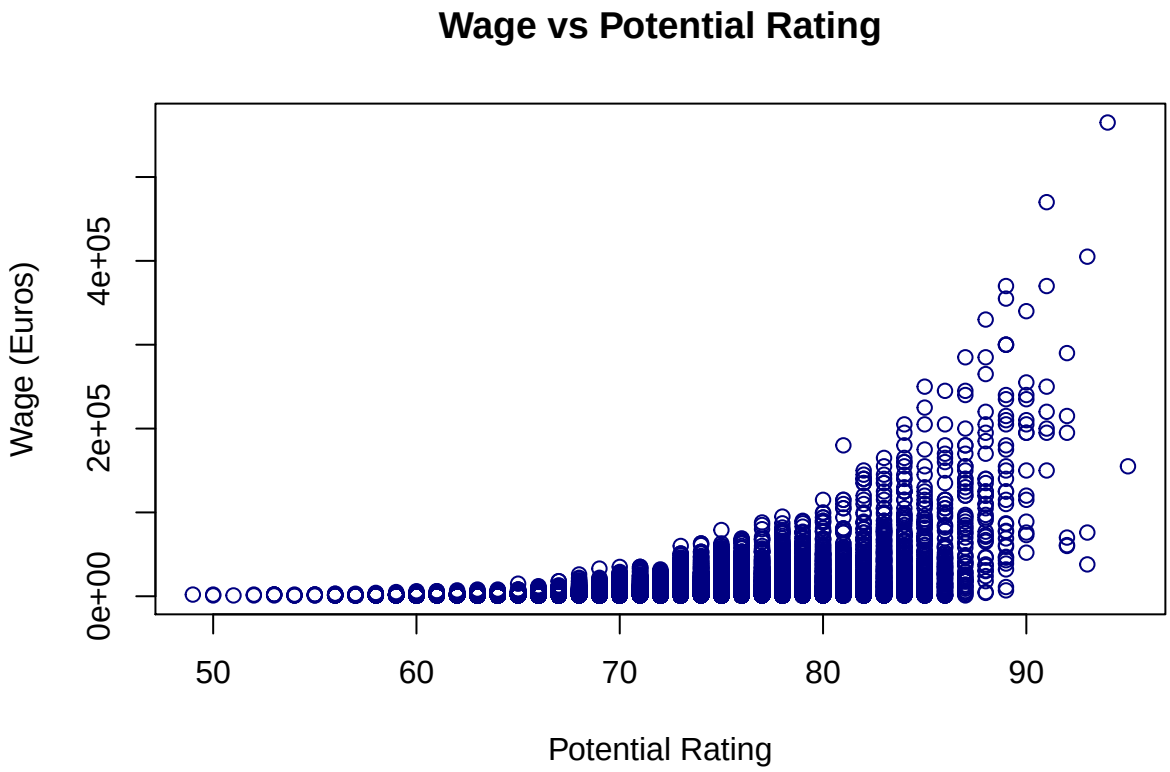
Plots of X's vs Y's

```
plot(x = fifa_data$overall,
     y = fifa_data$wage_eur,
     ylab = "Wage (Euros)",
     xlab = "Overall Rating",
     main = 'Wage vs Overall Rating',
     col = 'blue')
```



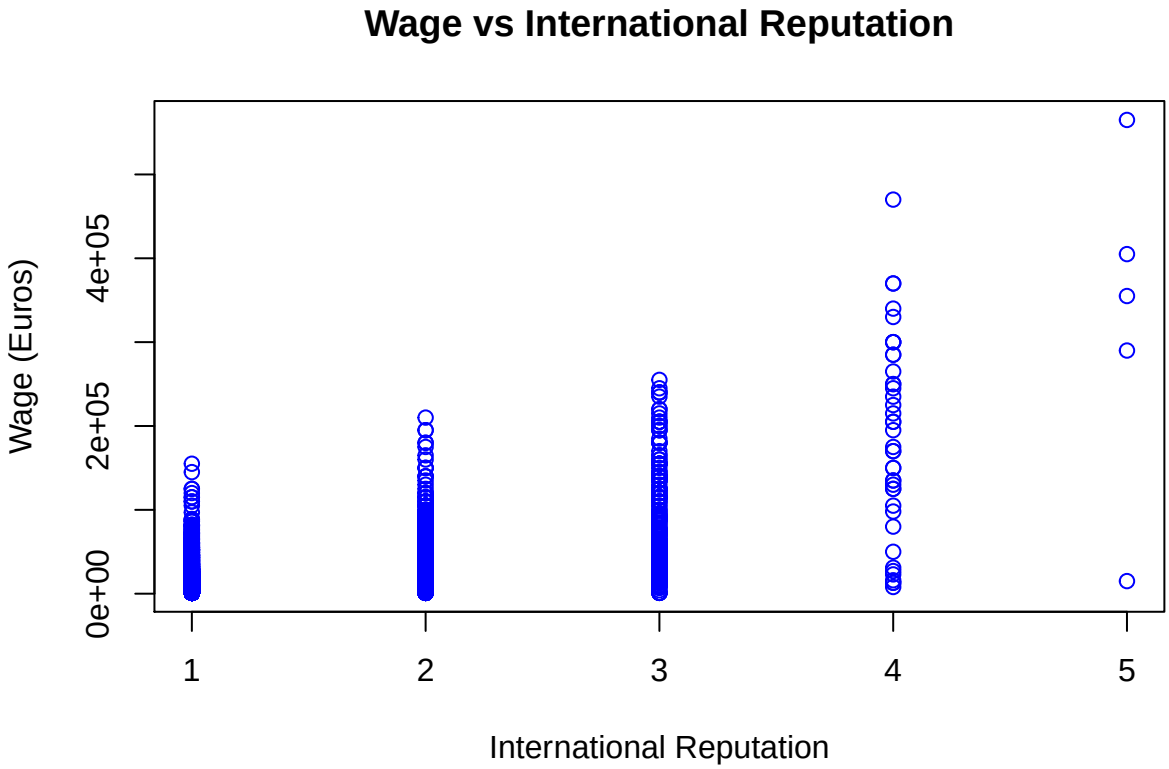
In this plot, you can see that overall rating has a clear correlation with wage. As overall rating goes up, wage also goes up. This plot also looks like an exponential function.

```r
plot(x = fifa_data$potential,
     y = fifa_data$wage_eur,
     ylab = "Wage (Euros)",
     xlab = "Potential Rating",
     main = 'Wage vs Potential Rating',
     col = 'navy')
```



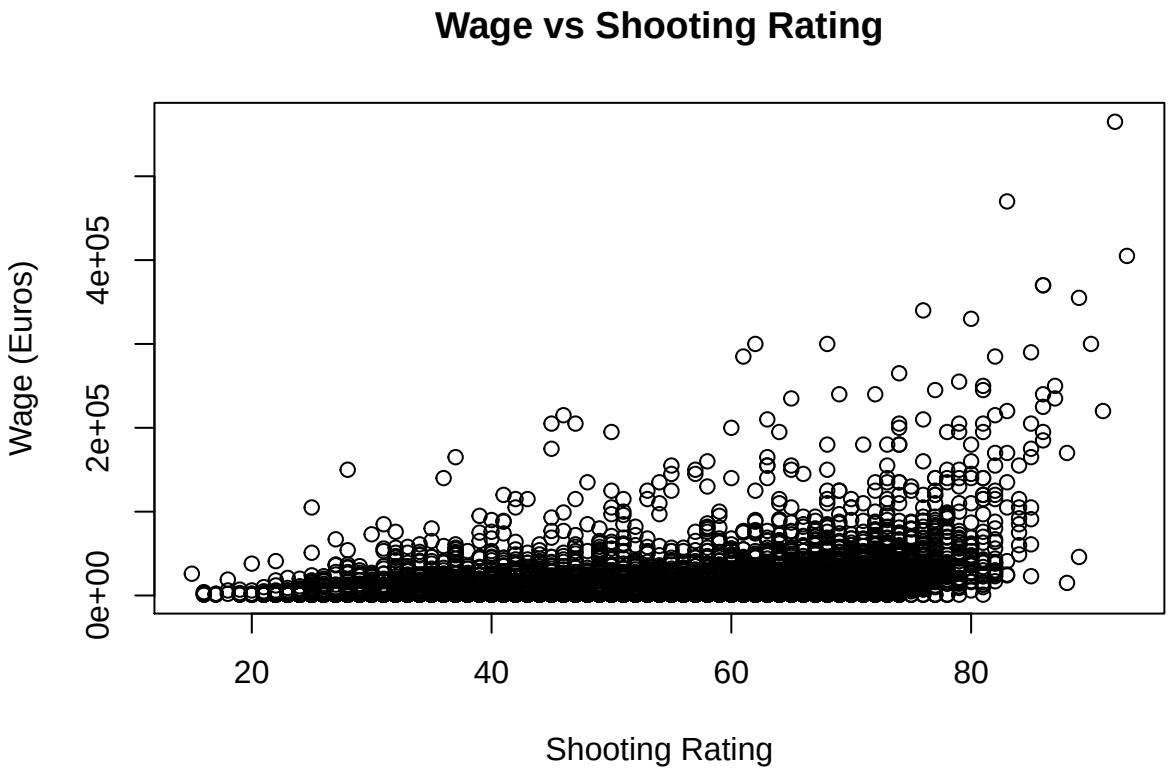
In this plot, you can also see that it is very similar to the results of the overall rating. This is because I think that overall rating is almost the same as potential rating on the players with the highest wages.

```r
plot(x = fifa_data$international_reputation,
     y = fifa_data$wage_eur,ylab = "Wage (Euros)",
     xlab = "International Reputation",
     main = 'Wage vs International Reputation',
     col = 'blue')
```



Again, this box plot shows that there are many players with an international reputation of 3 or less but, they do not make that much money. Most of the players with an international repuation of 4 or higher earn a much larger salary than those with a rating of 3 or less.

```r
plot(x = fifa_data$shooting, y = fifa_data$wage_eur,
     ylab = "Wage (Euros)",
     xlab = "Shooting Rating",
     main = 'Wage vs Shooting Rating')
```



In this plot, shooting rating does not tell us much about the wage of the players unless their shooting stat is very high compared to other players. As we can see, the players with 90 shooting or above has high salaries compared to 85 and below. I think this is because attackers in soccer are paid more than defenders.

```r
plot(x = fifa_data$age,
     y = fifa_data$wage_eur,
     ylab = "Wage (Euros)",
     xlab = "Age",
     main = 'Wage vs Age')
```

As we can see from this plot, most of the fifa players are between 23 and 33 years old. This is also where the players get paid the most because of their combination of physical form and experience.These are peak years for athletes and they have the wages to back that up.

Report 2

Finding the Best Regression

Fixing Release Clause Data

Since there are some release clauses that are NA, I set them to 0 to make sure that the dimensions of my regressions and residuals are the same.

Regression for different X's on wage

Model #1

```
fit_release = lm(data = fifa_data, wage_eur ~ release_clause_eur)
summary(fit_release)$adj.r.squared
```

```
## [1] 0.7286253
```

The variable I chose were players' release clause in Euros. This variable would make sense because the release clause should be in a similar range to wages.

Model #2

```
fit_value = lm(data = fifa_data, wage_eur ~ value_eur)
summary(fit_value)$adj.r.squared
```

```
## [1] 0.7341992
```

The variable I chose were players' value in Euros. This variable would make sense because player value should be in a similar range to wages.

Model #3

```
fit_pass = lm(data = fifa_data, wage_eur ~ passing)
summary(fit_pass)$adj.r.squared
```

```
## [1] 0.1667702
```

The variable I chose were the players' passing rating. This variable would make sense because as a player gets better at passing the more a player should get paid.

Model #4

```
fit_drib = lm(data = fifa_data, wage_eur ~ dribbling)
summary(fit_drib)$adj.r.squared
```

```
## [1] 0.1411672
```

The variable I chose were the players' dribbling rating. This variable would make sense because as a player gets better at dribbling the more a player should get paid.

Model #5

```r
fit_overall = lm(data = fifa_data, wage_eur ~ overall)
summary(fit_overall)$adj.r.squared
```

```
## [1] 0.340778
```

The variable I chose were the players' overall rating. This relationahip would make sense because as a player gets better as a player the more they should get paid.

Model #6

```r
fit_potential = lm(data = fifa_data, wage_eur ~ potential)
summary(fit_potential)$adj.r.squared
```

```
## [1] 0.2358599
```

The variable I chose was the potential rating of the players. The relationship between wage and potential rating makes sense because soccer clubs pay for player's potential.

Model #7

```r
fit_best = lm(wage_eur ~ release_clause_eur +
                value_eur + overall +
                potential, data = fifa_data)
summary(fit_best)$adj.r.squared
```

```
## [1] 0.7382487
```

The variables for this regression were release clause in euros, value in euros, overall rating, and potential rating. The relationship between these variables make sense because all these variables should affect the wage of a player.

Finding the best fit

```
summary(fit_best)
```

```
##
## Call:
## lm(formula = wage_eur ~ release_clause_eur + value_eur + overall +
##     potential, data = fifa_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -159723   -2177    -470    1561  242063
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)         2.836e+03  1.485e+03   1.910   0.0561 .
## release_clause_eur  5.542e-04  7.527e-05   7.363 1.89e-13 ***
## value_eur           2.216e-03  1.460e-04  15.173  < 2e-16 ***
## overall             2.518e+02  1.977e+01  12.738  < 2e-16 ***
## potential          -2.520e+02  2.112e+01 -11.928  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11450 on 15072 degrees of freedom
## Multiple R-squared:  0.7383, Adjusted R-squared:  0.7382
## F-statistic: 1.063e+04 on 4 and 15072 DF,  p-value: < 2.2e-16
```

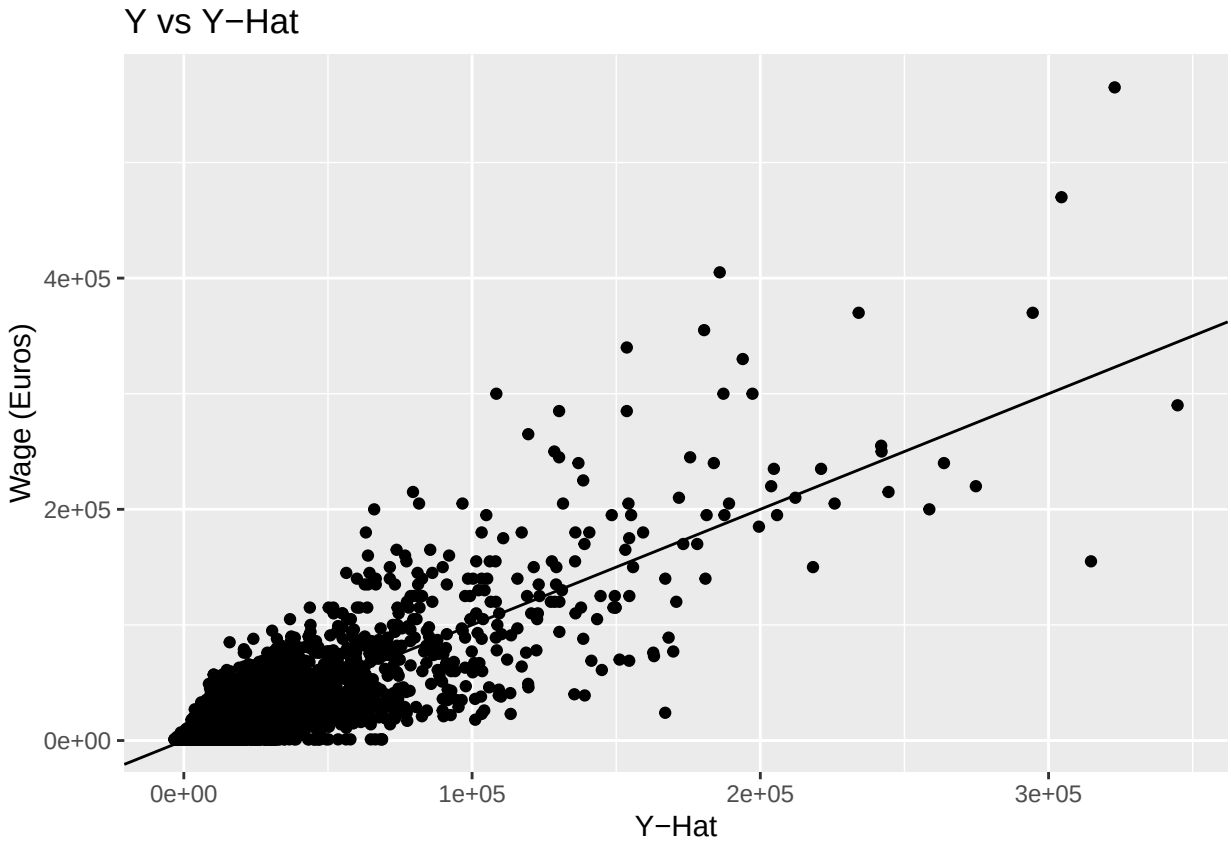
From my testing, I believe that fit_best is the best regression. The regression includes the X variables release clause (euros), value (euros), overall rating, and potential rating. The Adjusted R-Squared value is 0.7391 which is relatively high. In statistics, the Adjusted R-Squared value explains the percentage of the variation around the mean. In other words, the regression can explain 73.91 % of the variation around the mean. The p-value for all the X variables are significant. This means that the variable's p-value is less than .05. As we can see from the summary, the p-values of the variables are far less than .05 resulting in significant variables.

The mean response for release clause was 5.542e-04, mean response for value was 2.216e-03, the mean response for overall rating was 2.518e+02, and the mean response for potential rating was -2.520e+02. This means that holding all other variables constant, an increase of one Euro in release cause will cause wage to increase by .0005542 Euros. An increase of the value of a player by one Euro will cause wage to increase by .002216 Euros. An increase in overall rating will cause an increase in wage by 251.8 Euros. However, an increase in Potential Rating causes the weekly wage to decrease 252 Euros. This makes sense as the better the player, the better the wages. The same thing goes with release clause and value of a player. The more valuable the player, the more the player will get paid. The one weird thing I did notice in the regression was the mean response for potential rating because it was negative. One would think that an increase in potential rating would increase wages but that is simply not the case. One explanation could be that since overall rating and potential rating are very simliar they would cancel each other out in our regression line.

Our F statistic is 1.063e+04 on 4 and 15072 Degrees of Freedom with a p-value of 2.2e-16. This means that our statistics are significant and that our Adjusted R-Squared value is also significant.

Y vs Yhat

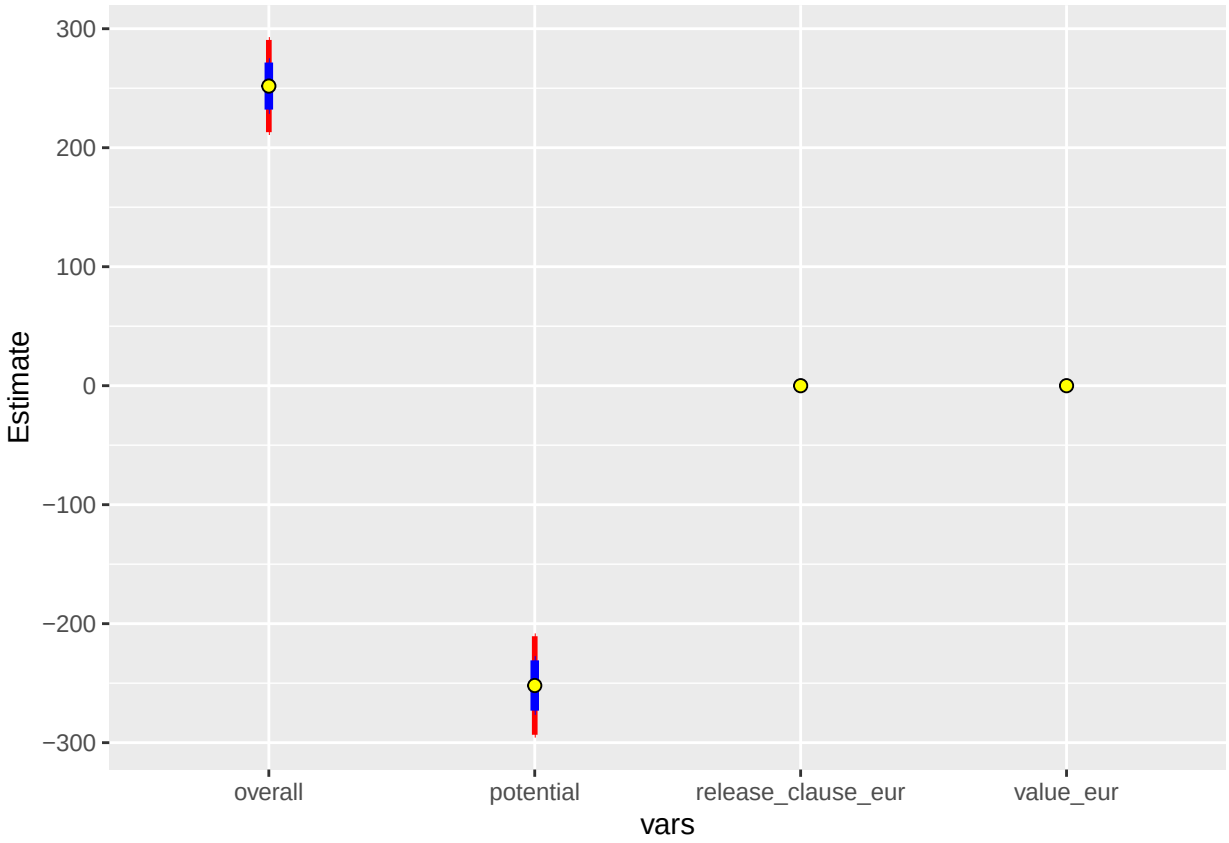
```
pred = fit_best$fitted.values
ggplot() + geom_point(aes(x = pred, y = fifa_data$wage_eur)) +
  geom_abline() + ggtitle('Y vs Y-Hat') +
  labs(x = 'Y-Hat', y = 'Wage (Euros)')
```



When plotting Y-hat onto Y, the desired result is a one-to-one relationship. As we can see from the plot, Y is almost a one to one function of the prediction but, the variance of the residuals increases for high values. So we have heteroscedasticity in the wage dimnesion.

Estimated Coefficients and Standard Error

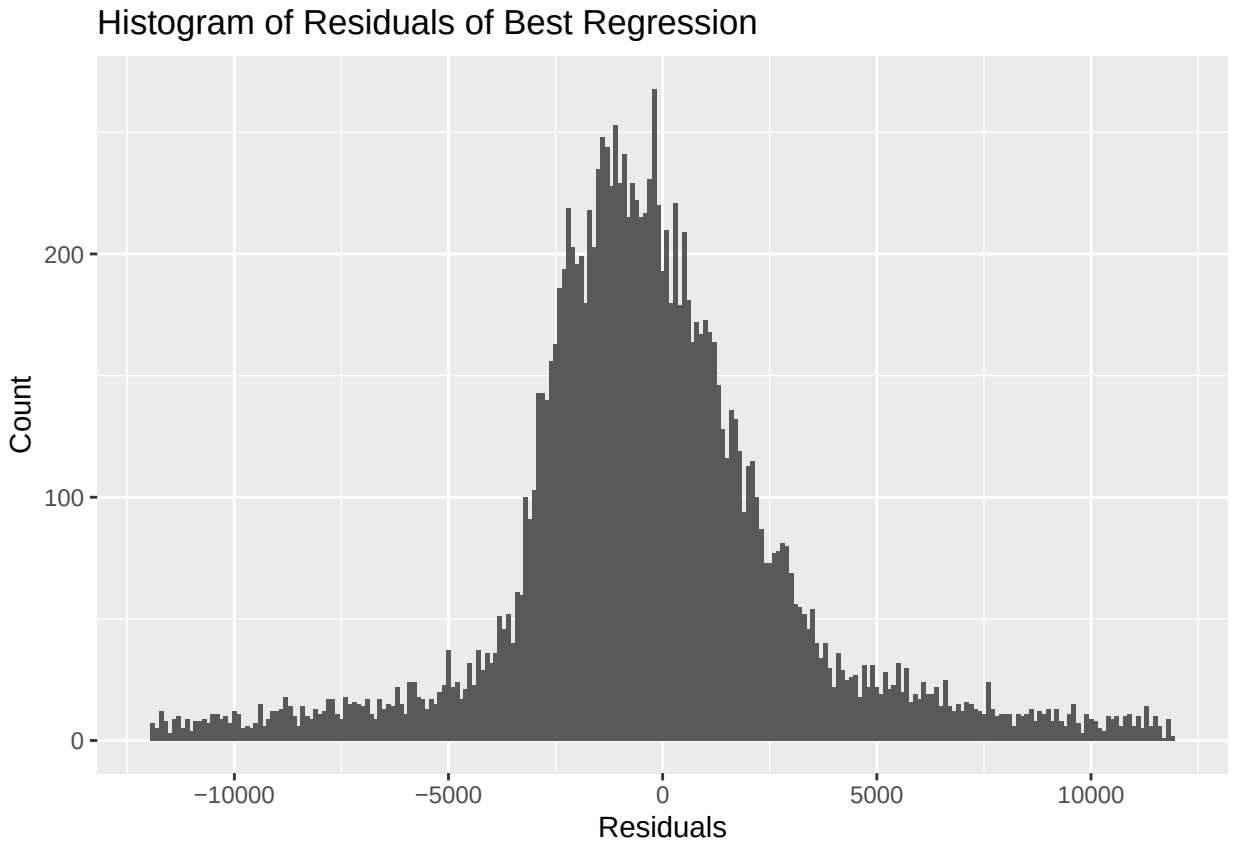
```
fitBest = summary(fit_best)
coefs = as.data.frame(fitBest$coefficients[-1, 1:2])
names(coefs)[2] = "se"
coefs$vars = rownames(coefs)
ggplot(coefs, aes(vars, Estimate)) +
  geom_errorbar(aes(ymin=Estimate - 1.96*se, ymax=Estimate + 1.96*se), lwd=1,
                colour="red", width=0) +
  geom_errorbar(aes(ymin=Estimate - se, ymax=Estimate + se),
                lwd=1.5, colour="blue", width=0) +
  geom_point(size=2, pch=21, fill="yellow")
```



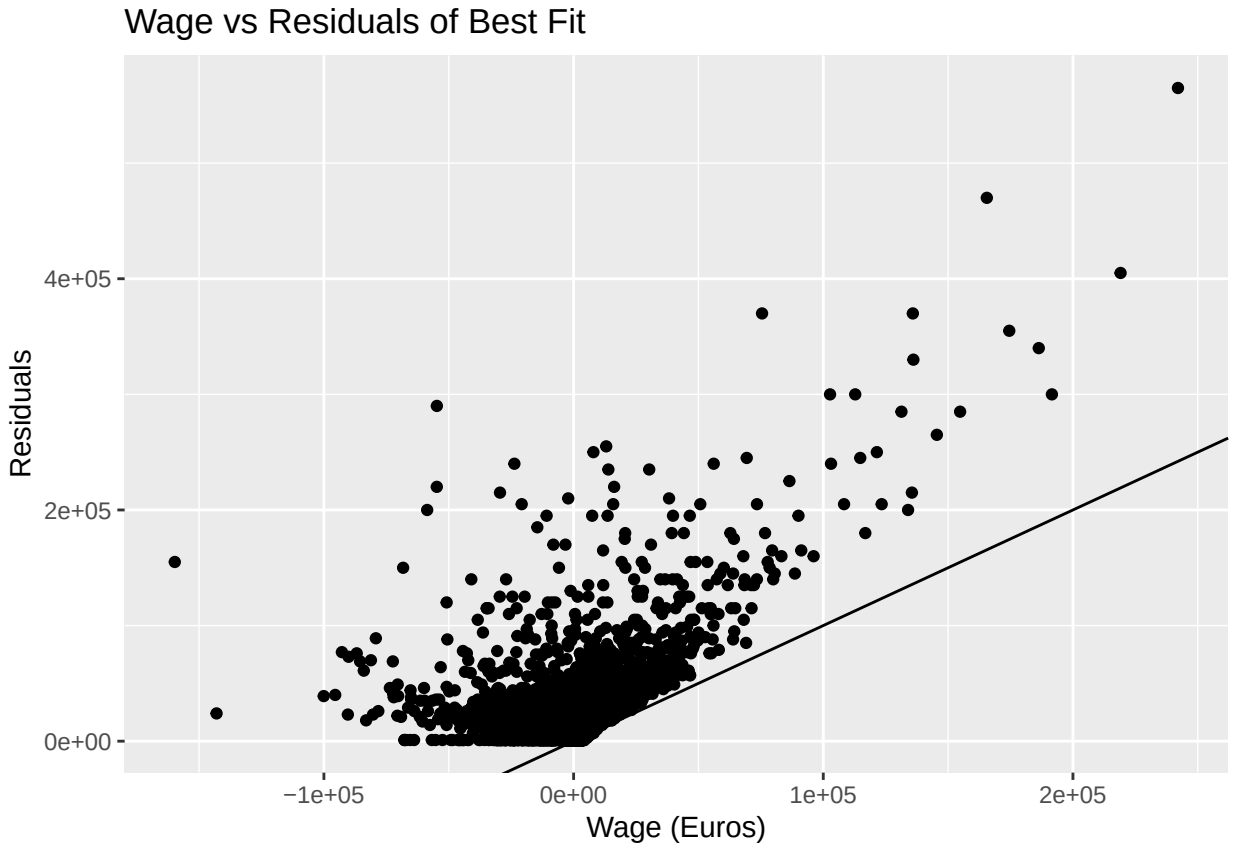
From the standard error plot, we can see the 95% confidence interval of the 4 different X variables in the FIFA 20 dataset. Clearly, overall and potential rating's confidence interval do not include 0. This results in those variables being significant. On the other hand, release clause and value include 0 in their confidence interval, resulting in those variables being insignificant. This is different than fit_best in which I found that all 4 X variables are significant.

Histograms of the Residuals

```
ggplot(data = fifa_data, aes(x = residuals(fit_best))) +
  geom_histogram(binwidth = 100) +
  xlim(-12000,12000) +
  ggtitle('Histogram of Residuals of Best Regression') +
  labs(x = 'Residuals', y = 'Count')
```




```
ggplot() + geom_point(aes(y = fifa_data$wage_eur, x = residuals(fit_best))) +
  ggtitle('Wage vs Residuals of Best Fit') +
  labs(x = 'Wage (Euros)', y = 'Residuals') + geom_abline()
```



The histogram of the residuals are normally distributed which is the desired result. For the plot of Wage vs Residuals, the desired results are a homosketastic plot and to see if the residuals are dependent on wage. As we can see from the plot, the plot is not homosketastic but not super heterosketastic either. However, since the plot is heterosketastic, the regression is not a very good indicator of wage.

Report 3

Question 1: Creating a train and test subset

```
set.seed(1)
train = fifa_data %>% sample_frac(.5)
test = fifa_data %>% setdiff(train)
x_train = model.matrix(wage_eur~., train)[,-1]
x_test = model.matrix(wage_eur~., test)[,-1]
y_train = train$wage_eur
y_test = test$wage_eur
```

Question 2: Creating a Ridge and Lasso Regression

a. Tune the model and find the best model

####Ridge Regression

```
set.seed(1)
x = model.matrix(wage_eur~., dataOmit)[,-1] # Predictors
y = dataOmit$wage_eur # Y-value
grid = 10^seq(10,-2,length = 100)
ridgeMod = glmnet(x, y, alpha = 0, lambda = grid)
cv.out = cv.glmnet(x_train, y_train, alpha = 0)
bestlam = cv.out$lambda.min
bestlam
```

```
## [1] 1838.587
```

The flexibly (lambda) of the ridge model is 1906.656. #### Lasso Regression

```
set.seed(1)
lassoMod = glmnet(x,y,alpha = 1, lambda = grid)
cv.outL = cv.glmnet(x_train, y_train, alpha = 1)
bestlamL = cv.outL$lambda.min
bestlamL
```

```
## [1] 57.46901
```

The flexibly (lambda) of the lasso model is 57.46901

b. Choose the Lambda within 1 SE of the minimum Lambda

####Ridge Regression

```
set.seed(1)
cv.out$lambda.1se
```

```
## [1] 8146.084
```

```
cv.out
```

```
##
## Call:  cv.glmnet(x = x_train, y = y_train, alpha = 0)
##
## Measure: Mean-Squared Error
##
##     Lambda   Measure      SE Nonzero
## min   1839 122273158 6915334      16
## 1se   8146 128884651 9007718      16
```

The range for 1 standard error for the ridge model is 1906.656 - 8146. The lambda I have chosen is 2980 because it is within 1SE of the minimum lambda

Lasso Regression

```
set.seed(1)
cv.outL$lambda.1se
```

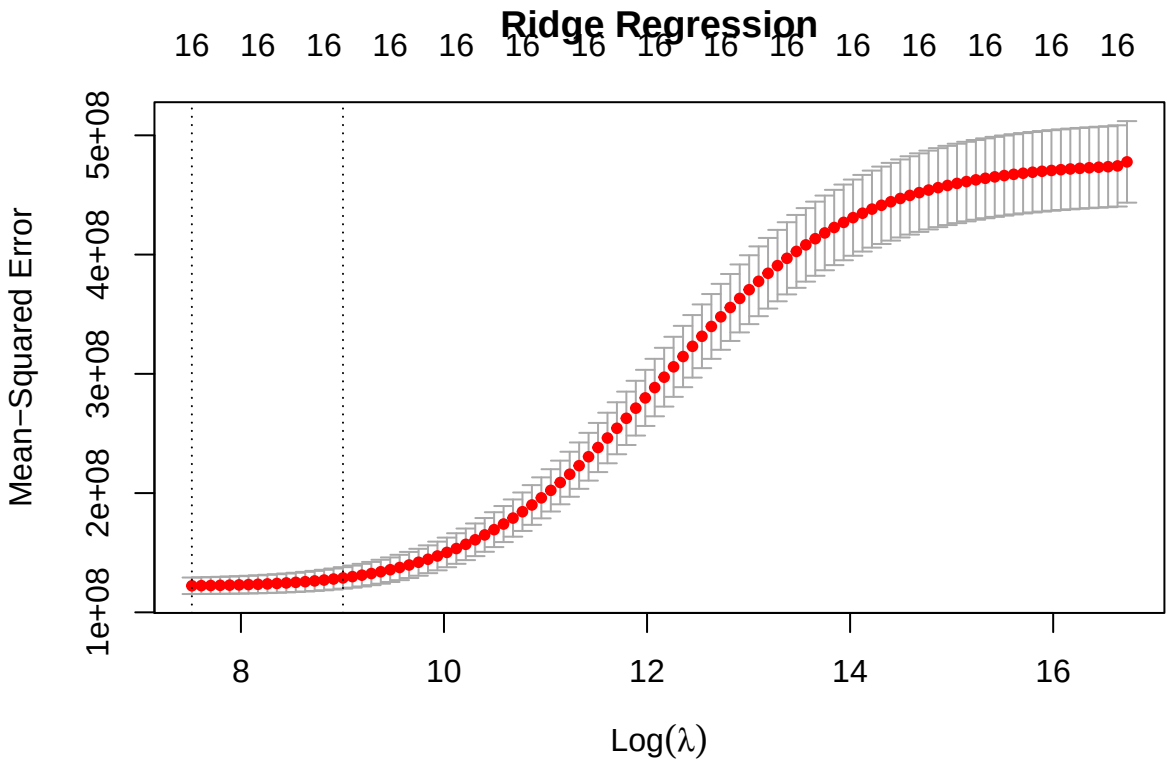
```
## [1] 1971.457
```

The range for 1 standard error for the lasso model is 19.51522 - 2243. The lambda I have chosen is 20 because it is within 1SE of the minimum lambda

c. CV and chosen lambda in a graph

Ridge Regression

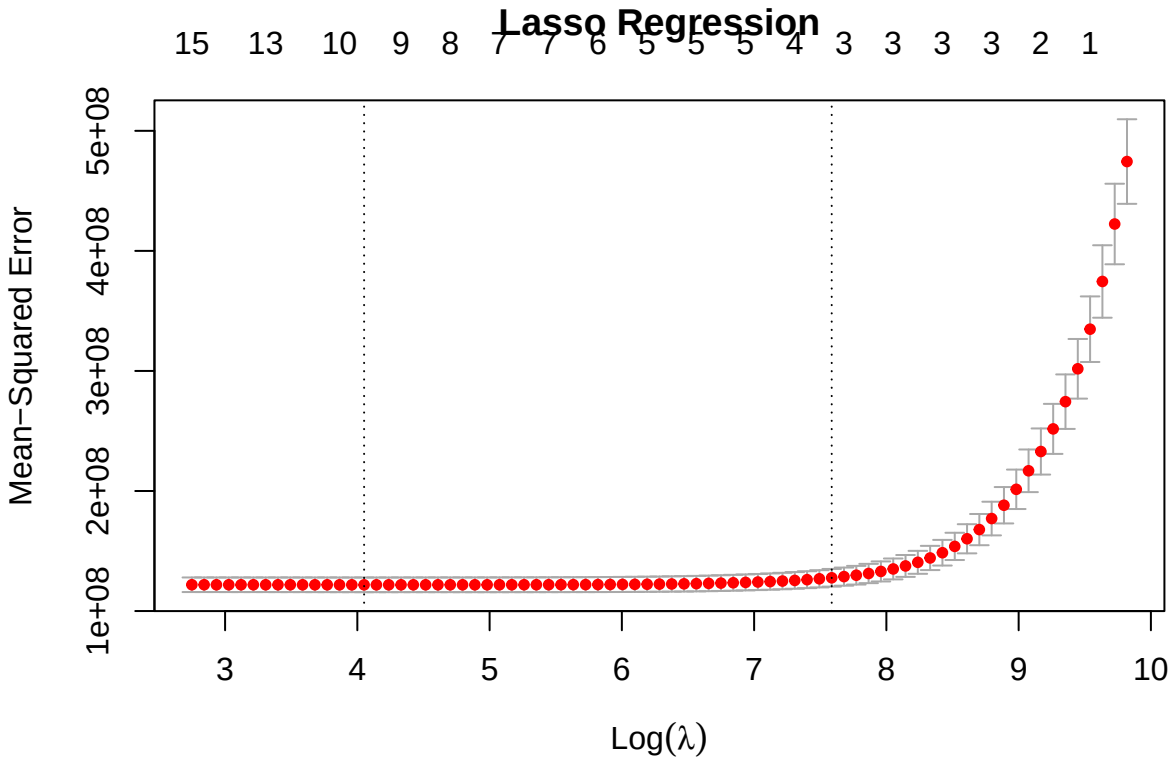
```
plot(cv.out)
title('Ridge Regression')
```



As we can see from the graph of the cross-validation error, the chosen lambda is indeed within 1 standard error of the minimum lambda. The value 2980 is where log(8) is.

Lasso Regression

```
plot(cv.outL, main = 'Lasso Regression')
```

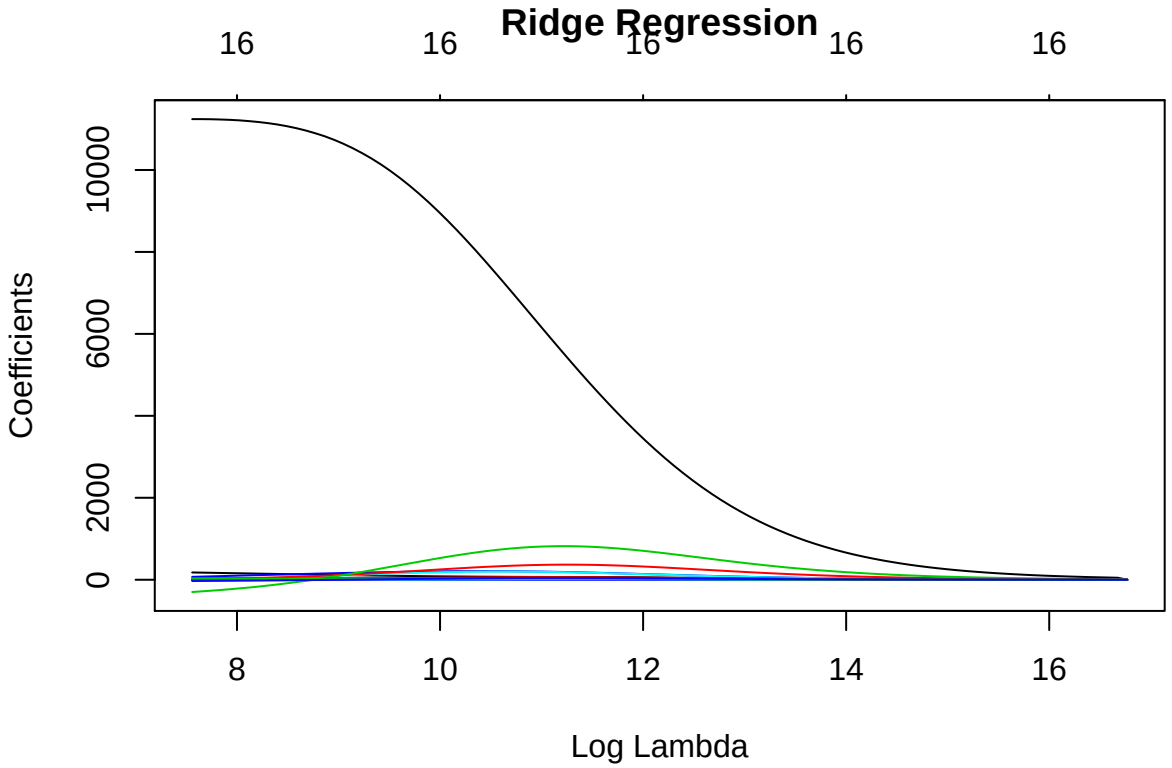


As we can see from the graph of the cross-validation error, the chosen lambda is indeed within 1 standard error of the minimum lambda. The value 20 is where log(3) is.

d. How coefficients change with lambda

####Ridge Regression

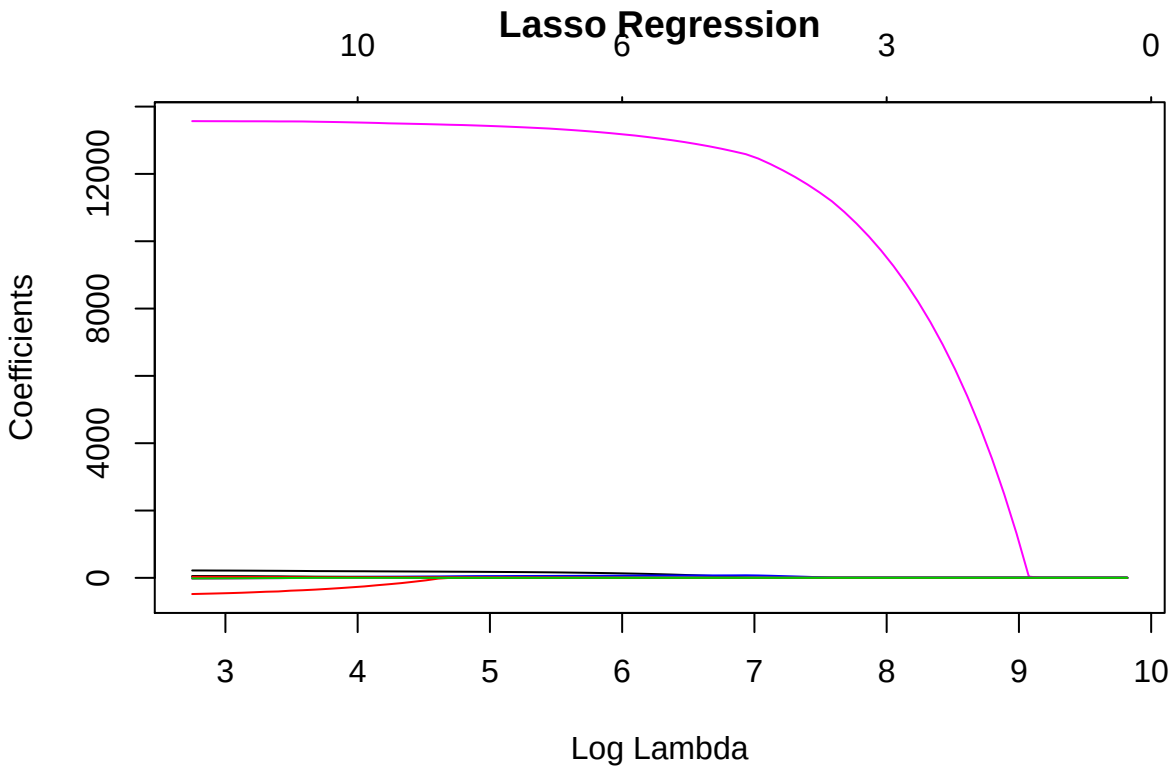
```
out = glmnet(x, y, alpha = 0)
plot(out, xvar = 'lambda', main = 'Ridge Regression')
```



This is graph shows us how as lambda changes, the coefficents also change. This shows the ridge regression shrinking all the coeffients towards zero. This graph shows that after around log(14) lags that all coefficients reach zero.

Lasso Regression

```
lasso_mod = glmnet(x_train, y_train, alpha = 1)
plot(lasso_mod, xvar = 'lambda', main = 'Lasso Regression')
```



Lasso regressions perform variable selection. What this means is that there will be coefficients that will be exactly zero from the start instead of shrinking the varibles like the Ridge regression. As we can see from the graph, we can see that there are coefficients that start from zero, just like it is supposed to.

e. Chosen coefficents correspond with lambda

```
ridge_coef = predict(ridgeMod, type = 'coefficients', s = bestlam)[1:17,]
ridge_coef
```

```
##              (Intercept)                      age               height_cm
##            -2.551984e+04             1.792653e+02             2.368289e+01
##                weight_kg                  overall                potential
##             3.186092e+01             6.728110e+01            -5.554609e+00
##                value_eur international_reputation                weak_foot
##             1.305041e-03             1.123612e+04             1.100030e+01
##              skill_moves       release_clause_eur                     pace
##            -3.119947e+02             7.077200e-04             7.627821e+00
##                 shooting                  passing                dribbling
##            -2.097801e+00            -8.753475e+00             4.003310e+01
##                defending                   physic
##             2.614232e+01            -3.092287e+01
```

Since the ridge regression does not perform variable selection, all the variables are still in the model. This means that the original 17 predictors are still our main predictors in our model.

```
lasso_coef = predict(lassoMod, type = 'coefficients', s = bestlamL)[1:17,]
lasso_coef[lasso_coef != 0]
```

```
##              (Intercept)                      age                weight_kg
##            -1.423630e+04             2.102385e+02             1.375051e+01
##                potential                value_eur international_reputation
##            -2.546854e+01             1.094732e-03             1.117322e+04
##              skill_moves       release_clause_eur                dribbling
##            -2.345110e+02             9.101879e-04             6.907231e+00
##                defending                   physic
##             1.532857e+01            -8.650796e+00
```

On the other hand, lasso regression does perform variable selection however, in this case, ridge regression only removed one variable. That variable was passing, leaving the rest of the variables as the main predictors of our model.

f. MSE of both models

```
set.seed(1)
ridgePred = predict(ridgeMod, s = 2980, newx = x_test)
mean((ridgePred - y_test)^2)
```

```
## [1] 119926096
```

```
set.seed(1)
lassoPred = predict(lassoMod, s = 20, newx = x_test)
mean((lassoPred - y_test)^2)
```

```
## [1] 116569880
```

The MSE for the ridge model is 119,926,096 and the MSE for the lasso model is 116,569,880. We can see that the lasso model is a better fit for our data set because of the lower MSE.

g. Comparing with OLS

```
library(dvmisc)
get_mse(fit_best)
```

```
## [1] 131028294
```

The MSE of the OLS model is 131,028,294, which is higher than both the ridge and lasso models. This means that the ridge and lasso models are better indicators for estimating wage of professional soccer players.

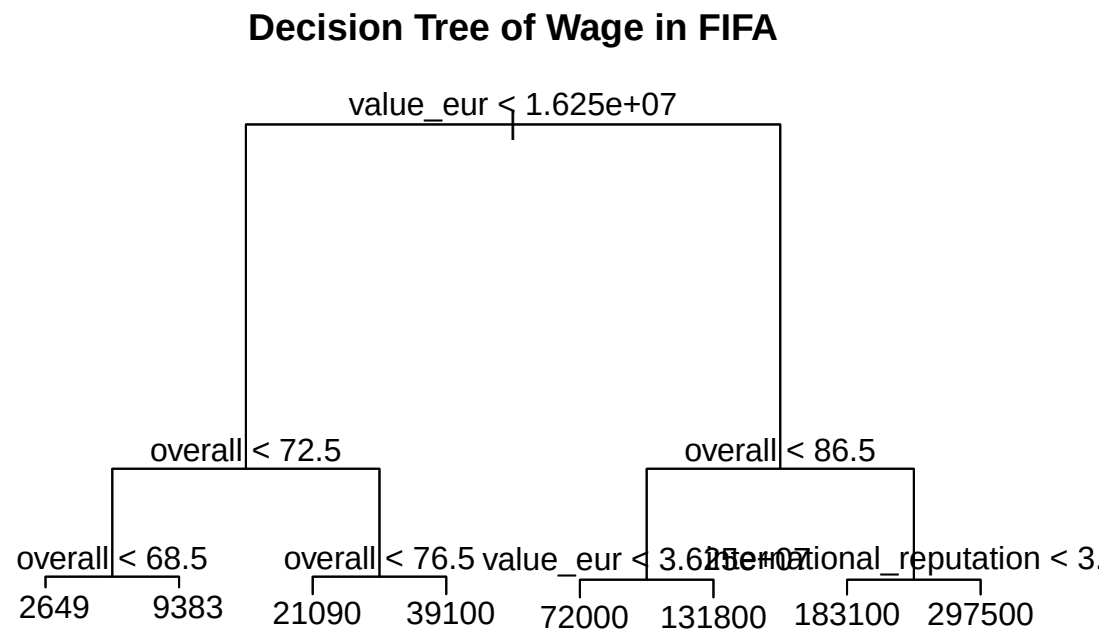
Question 3: Regression Tree

a. Fit and plot a big tree

```
set.seed(1)
fifa_train = fifa_data %>% sample_frac(.7)
fifa_test = fifa_data %>% setdiff(fifa_train)
tree_fifa = tree(wage_eur~., fifa_train)
summary(tree_fifa)
```

```
##
## Regression tree:
## tree(formula = wage_eur ~ ., data = fifa_train)
## Variables actually used in tree construction:
## [1] "value_eur"                "overall"
## [3] "international_reputation"
## Number of terminal nodes:  8
## Residual mean deviance:  107600000 = 1.135e+12 / 10550
## Distribution of residuals:
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -167500.0   -1649.0    -649.3       0.0    1351.0  172500.0
```

```
plot(tree_fifa) # The tree
text(tree_fifa, pretty = 2) # The tree
title('Decision Tree of Wage in FIFA')
```



b. Check the Error rate or MSE on the test subset

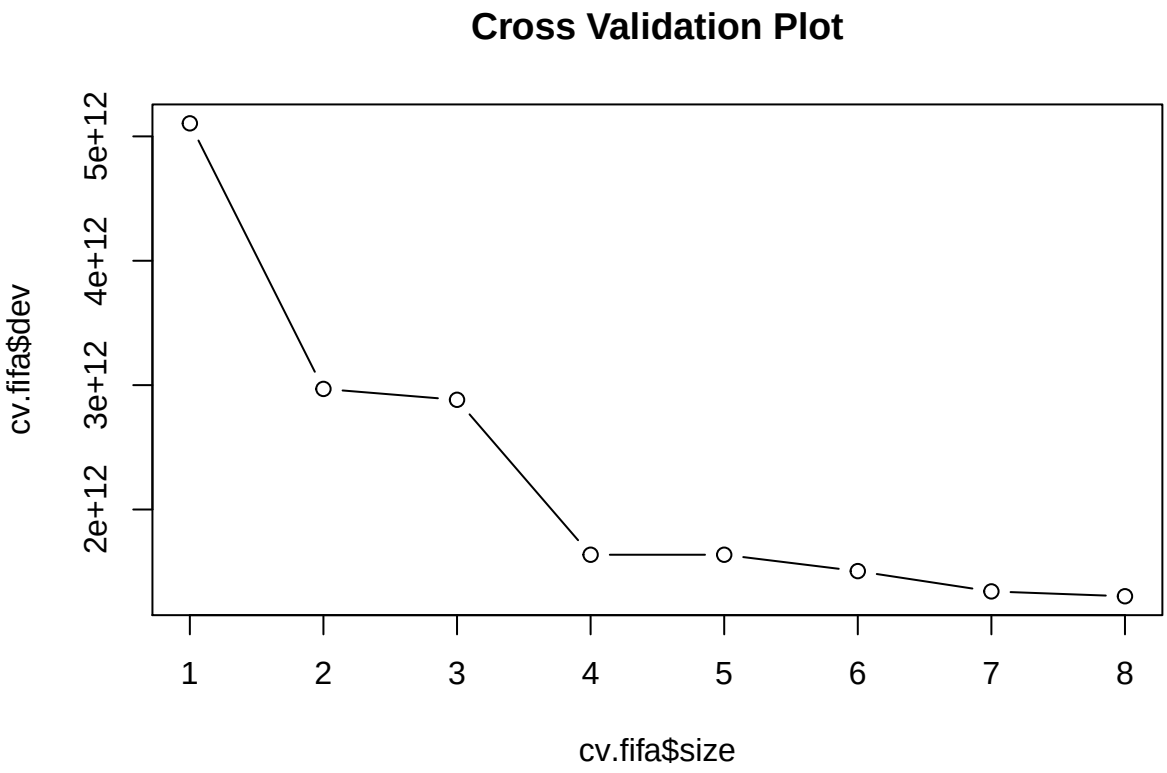
```
single_tree_estimate_tree = predict(tree_fifa, newdata = fifa_test)
mean((single_tree_estimate_tree - fifa_test$wage_eur)^2)
```

```
## [1] 142610192
```

The MSE of the tree above is 142,610,192. This may seem like a huge number but it is reasonable considering that the wage of a player can range from 0 - around 500,000.

c. Use cross validation to prune tree

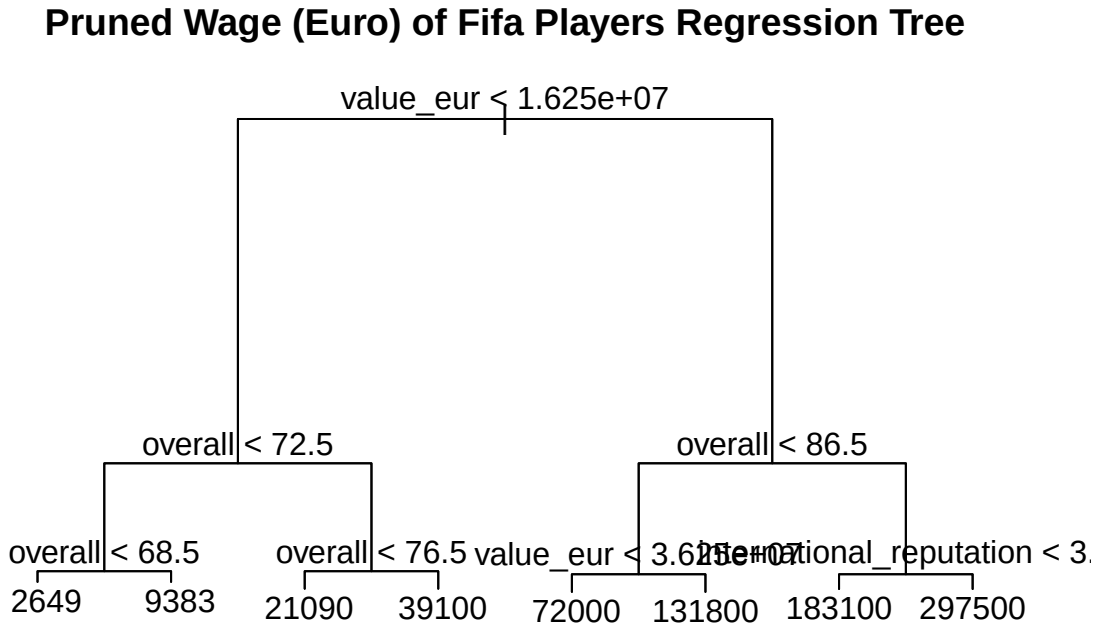
```
cv.fifa = cv.tree(tree_fifa)
plot(cv.fifa$size, cv.fifa$dev, type = 'b', main = 'Cross Validation Plot')
```



As you can see from the plot above, the 8 node tree is selected by cross-validation. This is the same as the unpruned tree. In this case, the tree above is already the best tree possible.

d. Plot the pruned tree

```
prune_fifa = prune.tree(tree_fifa, best = 8)
plot(prune_fifa)
text(prune_fifa, pretty = 0)
title('Pruned Wage (Euro) of Fifa Players Regression Tree')
```



e. Interpretion of Tree

As we can see from the tree summary, the variables that affect the tree are value (Euros), overall rating, and international reputation. The variable value_eur is the measure of the value of the rating by the team. Overall rating is the overall attribute of the player, based on all of the player stats. International reputation is an attribute that is based off international competition performance and is rated from 1-5. The benefit of this tree is that it is very simple and easy to follow. For example, I will estimate David Silva's wage from his given information. His overall rating is 88, his value in euros is 36000000, and his international reputation is 4. Since his value is above 1.625e+07, we traverse to the right side. Silva's overall rating is above 86.5 as well so we would again traverse to the right side of that node. Silva's international rating is 4 which means we traverse to the right of the international rating node, resulting in 297500 for his wage estimation. Silva's actual wage is 265,000 Euros. So, from this tree, it is fairly accurate.

f. Comparing to OLS, lasso, and ridge models

```
single_tree_estimate_prune = predict(prune_fifa, newdata = fifa_test)
ggplot() +
  geom_point(aes(x = fifa_test$wage_eur, y = single_tree_estimate_prune), color = 'blue') +
  geom_abline(color = 'red') +
  labs(x = 'test subset wage', y = 'predicted wage') +
  ggtitle('Predicted vs Test Subset Wage')
```


```
mean((single_tree_estimate_prune - fifa_test$wage_eur)^2)
```

```
## [1] 142610192
```

```
library(dvmisc)
get_mse(fit_best)
```

```
## [1] 131028294
```

The MSE for the best linear model I have chose was 131,028,294 which is less than the regression tree. This means that the wages from the data set follow a more linear model rather than a regression tree.

Report 4

Question 1: Creating a Train and Test Subset

```
set.seed(1)
bag_train = fifa_data %>% sample_frac(.5)
bag_test = fifa_data %>% setdiff(train)
```

I chose a training and test set from the fifa dataset. I will use the same training and test data with bagging and boosting to make sure comparing the models will be as fair as possible.

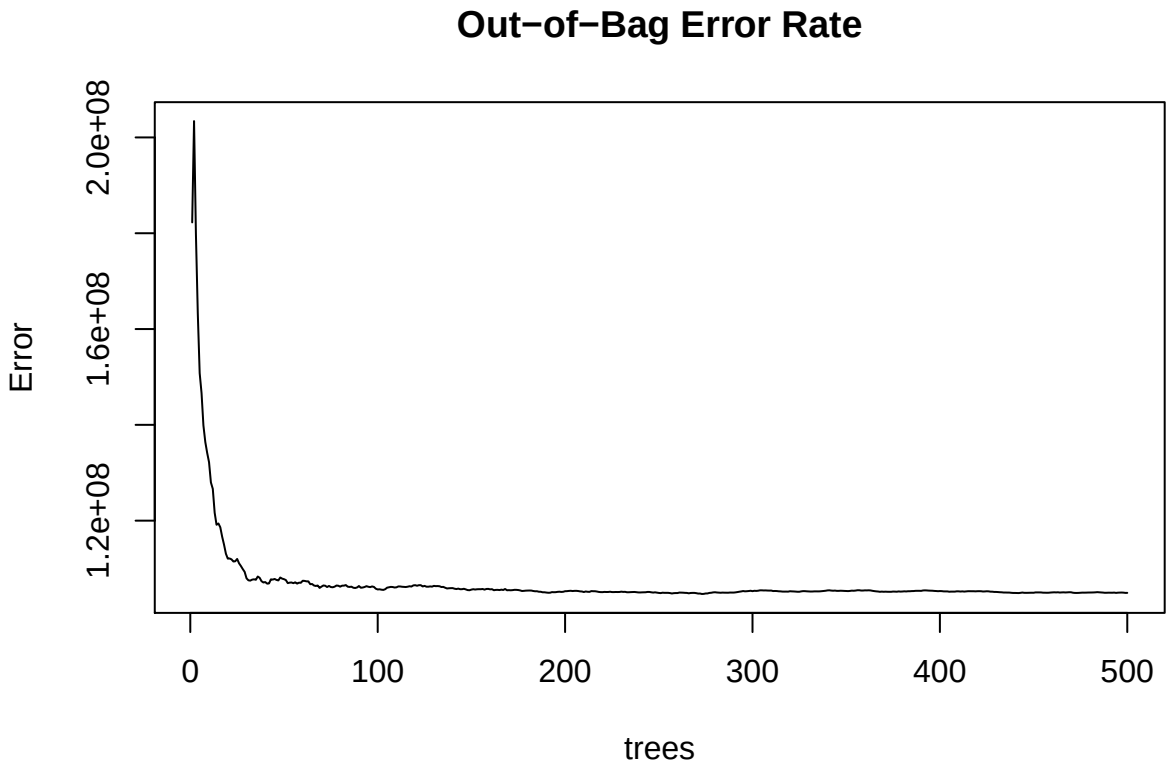
Question 2: Fit a Bagging Regression

a. Out-of-Bag Error Rate Plot

```
set.seed(1)
bag.fifa = randomForest(wage_eur ~., data = bag_train,
                        mtry = ncol(bag_train) - 1,
                        importance = TRUE)
bag.fifa
```

```
##
## Call:
##  randomForest(formula = wage_eur ~ ., data = bag_train, mtry = ncol(bag_train) -      1, importance =
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 16
##
##           Mean of squared residuals: 104928625
##                     % Var explained: 78.04
```

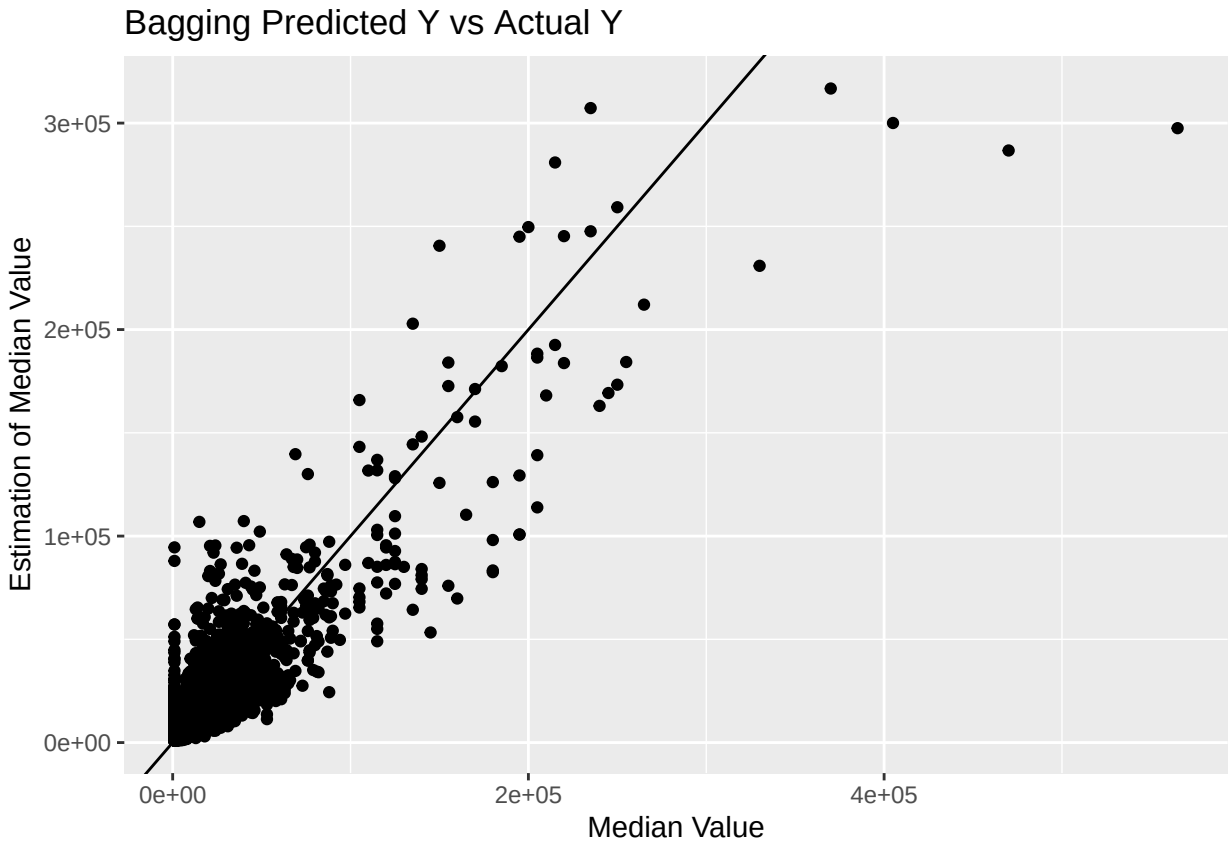
```
plot(bag.fifa, main = 'Out-of-Bag Error Rate')
```



Since Bagging is simply just Random Forest with m equalling the number of parameters, we can just use the randomForest() function. From the plot, as the number of trees increases, the error of the model decreases until a certain asymptote. This asymptote is around 1.1e+08.

b. Predicted Y vs Actual Y Plot

```
yhat.bag = predict(bag.fifa, newdata = bag_test)
ggplot() + geom_point(aes(x = bag_test$wage_eur, y = yhat.bag))+
  geom_abline() +
  labs(x = "Median Value", y = "Estimation of Median Value") +
  ggtitle('Bagging Predicted Y vs Actual Y')
```



As we can see from the results of our prediction vs actual value plot, the plot follows a general slope of 1. This is good because we want our predicted and test results to have a one-to-one relationship. This plot shows only a few outliers in our predictions and is generally a good model for most of the values.

c. MSE of Bagging

```
set.seed(1)
bag_mse = mean((yhat.bag - bag_test$wage_eur)^2)
bag_mse
```

```
## [1] 107274225
```

The MSE from the test set associated with the bagged regression tree is 107,274,225. This MSE is much about 20,000,000 smaller than our result in the ridge and lasso regression model. Comparing to the OLS and regression tree model, the MSE of bagging, the difference is even more substanial. From these results, we can conclude that the bagging regression model is significantly better than all models covered before.

d. Importance Matrix

```
importance = importance(bag.fifa)
importance
```

```
##                            %IncMSE IncNodePurity
## age                      15.831421  4.750416e+10
## height_cm                 2.539036  5.020712e+10
## weight_kg                 8.669967  6.031909e+10
## overall                  36.874785  1.704460e+12
## potential                11.426659  2.855883e+10
## value_eur                21.392925  9.368232e+11
## international_reputation 12.437580  1.172317e+11
## weak_foot                 1.780859  1.958086e+10
## skill_moves               1.037069  7.488531e+09
## release_clause_eur       21.691519  2.792071e+11
## pace                      2.140731  6.137943e+10
## shooting                  9.008388  5.639764e+10
## passing                  -1.010040  5.175290e+10
## dribbling                15.418233  4.085574e+10
## defending                 7.882075  4.504406e+10
## physic                    5.883801  4.876079e+10
```

This importance matrix is based off of two measures. The first is %IncMSE which is based on the **mean decrease of accuracy in predictions** on the out-of-bag samples when a given variable is excluded from the model. As we can see inour importance matrix, overall rating, release clause, and value are the tree most important variables in model. The second is a measure of the **total increase in node impurity** that results from splits over that variable. Thus, a higher IncNodePurity is more important. As you can see, the importance of the variables are generally the same with the most important variables being overall rating, value, and release clause.

Question 3: Random Forest

```
set.seed(1)
rf.fifa = randomForest(wage_eur ~., data = bag_train,
                       mtry = 6,
                       importance = TRUE,
                       do.trace = 100)
```

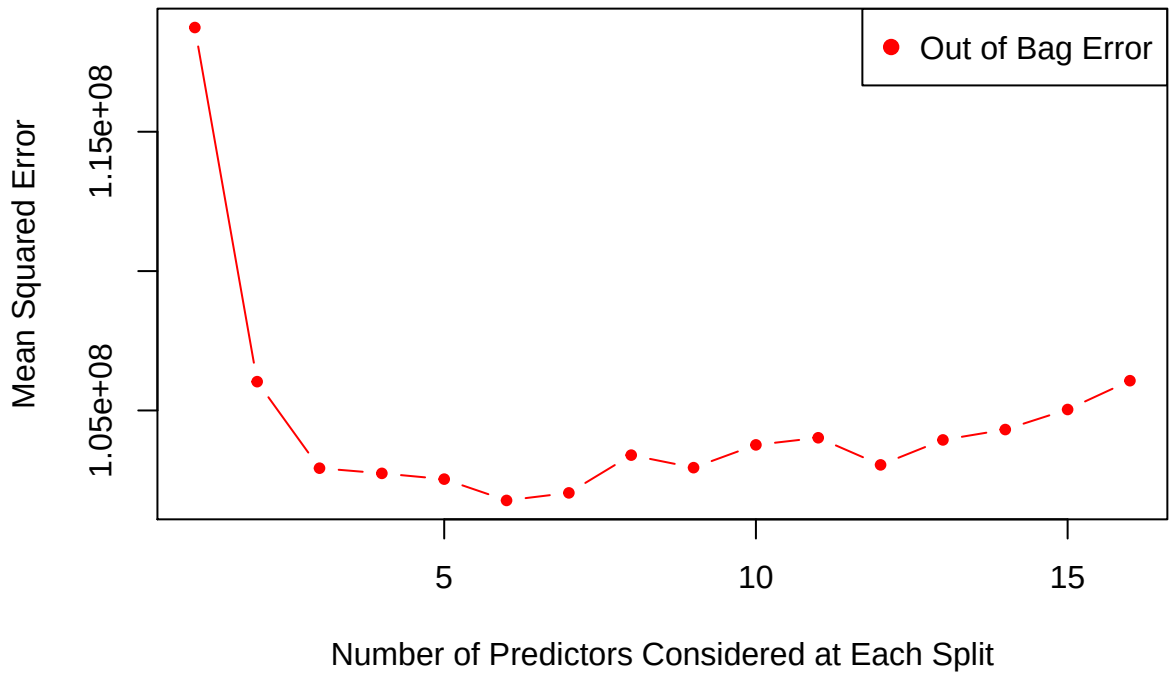
```
##      |      Out-of-bag   |
## Tree |      MSE  %Var(y) |
##  100 | 1.049e+08    21.96 |
##  200 | 1.041e+08    21.78 |
##  300 | 1.023e+08    21.42 |
##  400 | 1.022e+08    21.38 |
##  500 | 1.023e+08    21.40 |
```

```
yhat.rf = predict(rf.fifa, newdata = bag_test)
```

a, b. Out of Bag Error

```
oob.err = double(16)
test.err = double(16)
for(mtry in 1:16) {
  rf = randomForest(wage_eur ~., data = bag_train, mtry = mtry, ntree = 400)
  oob.err[mtry] = rf$mse[400]

  pred = predict(rf, bag_test)
  test.err[mtry] = with(bag_test, mean((wage_eur - pred)^2))
}
matplot(1:mtry ,
        cbind(oob.err),
        pch=20 ,
        col=c("red"),
        type="b",
        ylab="Mean Squared Error",
        xlab="Number of Predictors Considered at Each Split")
legend("topright",
       legend=c("Out of Bag Error"),
       pch=19, col=c("red"))
```

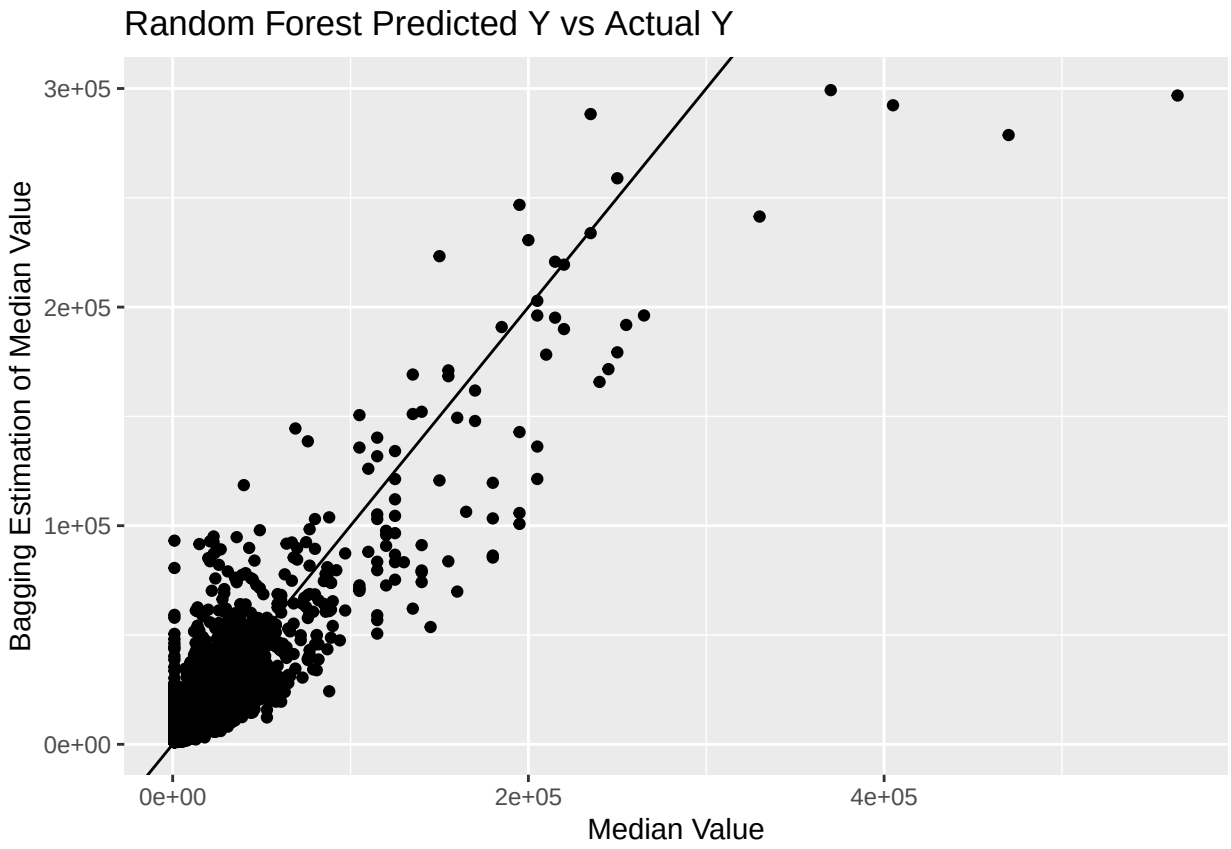


From the OOB error, we can see that the ideal number of parameters to use is 6.

c. Plot Predicted Y vs Actual Y

```
yhat.rf = predict(rf.fifa, newdata = bag_test)

ggplot() +
  geom_point(aes(x = bag_test$wage_eur, y = yhat.rf)) +
  geom_abline() +
  labs(x = 'Median Value', y = 'Bagging Estimation of Median Value') +
  ggtitle('Random Forest Predicted Y vs Actual Y')
```



As we can see from the results of our prediction vs actual value plot, the plot follows a general slope of 1. You may notice that our plot is very similar to the bagging but, this is due to the fact that Random Forest is actually a very similar process to Bagging, resulting in a similar prediction set.

d. MSE

```
mse.rf = mean((yhat.rf - bag_test$wage_eur)^2)
mse.rf
```

```
## [1] 103884216
```

The MSE from the test set associated with the Random Forest regression tree is 103,884,216. Comparing this value to the previous Bagging regression tree MSE (107,274,225), the MSE is even smaller than that which makes the Random Forest regression tree the best model so far.

e. Importance Matrix

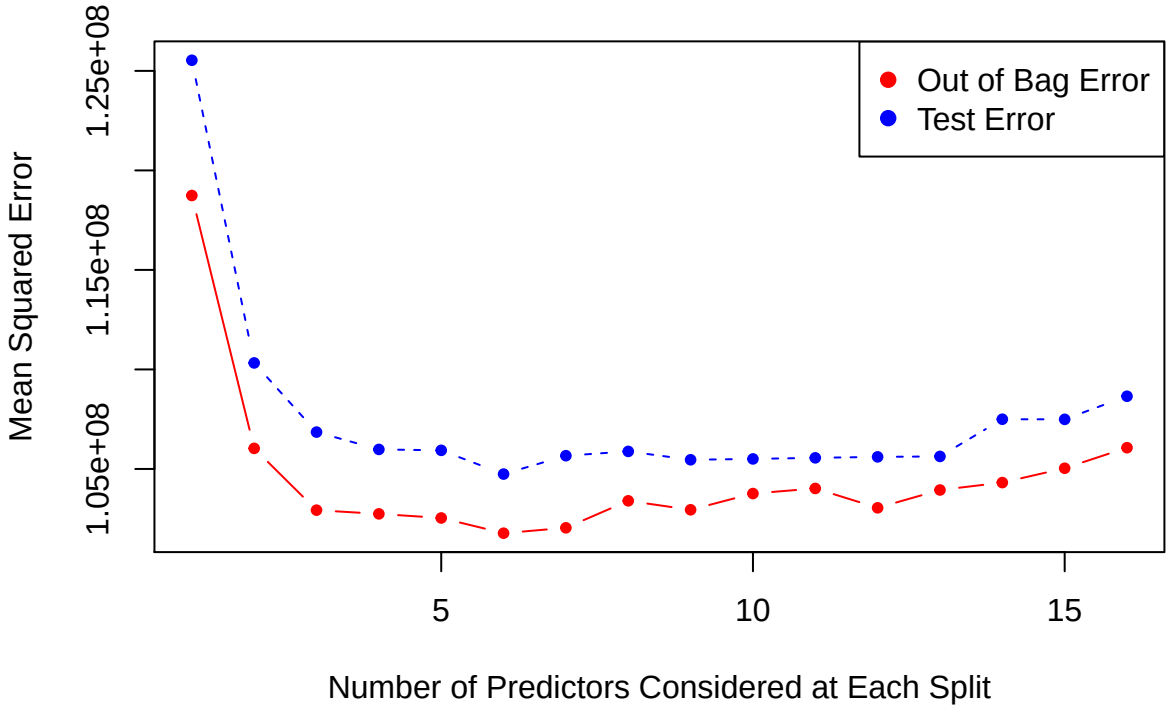
```
importance(rf.fifa)
```

```
##                            %IncMSE IncNodePurity
## age                      13.894729  7.601008e+10
## height_cm                 5.060446  4.984944e+10
## weight_kg                 6.956009  5.783345e+10
## overall                  23.638846  1.002250e+12
## potential                11.592124  9.549013e+10
## value_eur                21.137759  8.307189e+11
## international_reputation 15.572659  2.902827e+11
## weak_foot                 4.752956  1.951143e+10
## skill_moves               3.840237  1.005520e+10
## release_clause_eur       19.090319  6.745111e+11
## pace                      2.657693  5.547740e+10
## shooting                  9.002419  6.767865e+10
## passing                   3.425335  1.021379e+11
## dribbling                14.947394  7.488366e+10
## defending                 9.145918  7.319538e+10
## physic                    5.441585  4.892612e+10
```

From this importance matrix, we can see that the importance of overall rating, value, and release clause are even more important than before. This is because of the increase in %IncMSE as well as IncNodePurity. The higher the %IncMSE and IncNodePurity means the more important that variable is.

f. Test and Out-of-Bag Error

```
matplot(1:mtry ,
        cbind(oob.err, test.err),
        pch=20 ,
        col=c("red", "blue"),
        type="b",
        ylab="Mean Squared Error",
        xlab="Number of Predictors Considered at Each Split")
legend("topright",
       legend=c("Out of Bag Error", "Test Error"),
       pch=19, col=c("red", "Blue"))
```

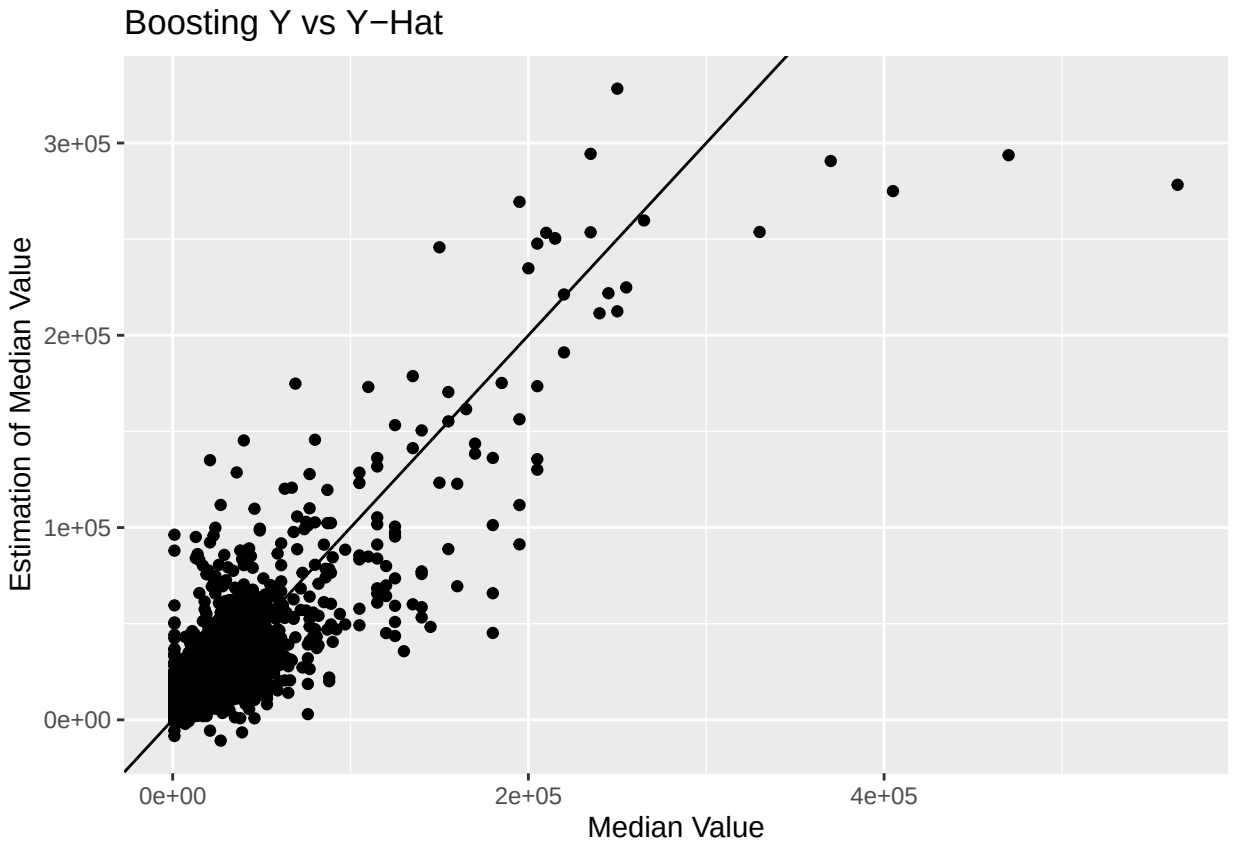


From this plot, we can see that both the Out-of-Bag Error follows a similar patter to the test error which is what we want to see.

Question 4: Fit a Boosting Regression

a. Plot Predicted Y vs Actual Y

```
set.seed(1)
boost.fifa = gbm(wage_eur~.,
                 data = bag_train,
                 distribution = 'gaussian',
                 n.trees = 5000,
                 interaction.depth = 4)
yhat.boost = predict(boost.fifa, bag_test, n.trees = 5000)
ggplot() +
  geom_point(aes(x = bag_test$wage_eur, y = yhat.boost)) +
  geom_abline() +
  labs(x = "Median Value", y = "Estimation of Median Value") +
  ggtitle('Boosting Y vs Y-Hat')
```



Once again, the prediction of Y's vs Y-Hat's are very similar to previous prediction plots with similar outliers. But, where the actual differences come are how accurate the vast majority of the points are with the Boosting model not being as accurate as Bagging or Random Forest.

b. Compare test MSE to other models

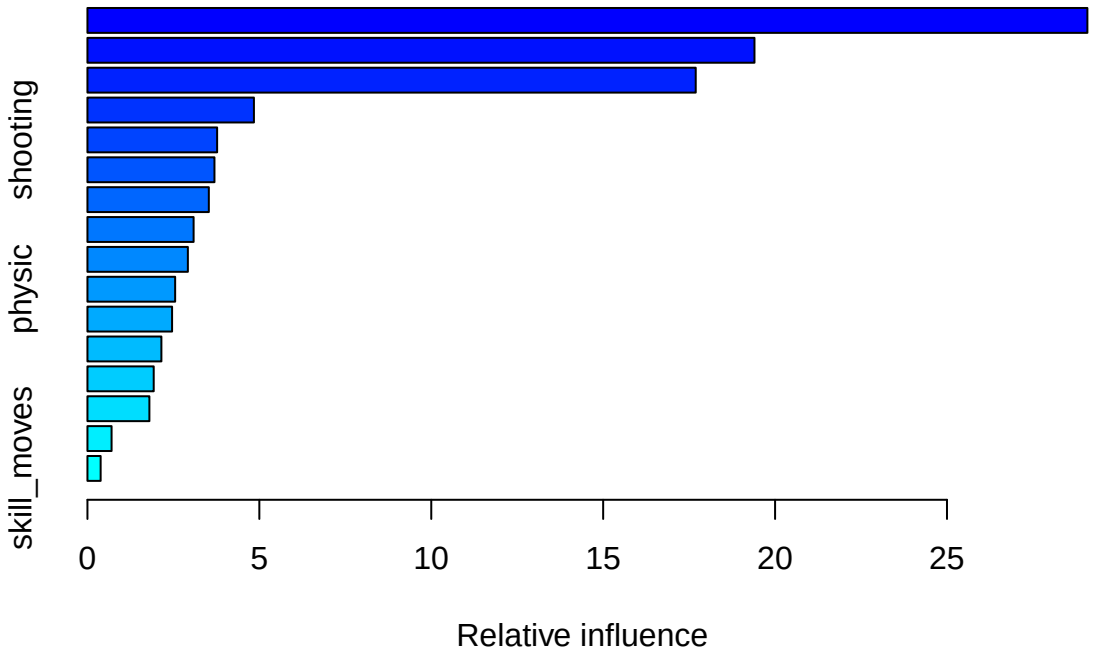
```
mse.boost = mean((yhat.boost-bag_test$wage_eur)^2)
mse.boost
```

```
## [1] 127282596
```

The MSE from the test set associated with the Random Forest regression tree is 127,282,596. Comparing this value to the previous Random Forest regression tree MSE (103,884,216), the MSE is about 25% higher which places the accuracy of this model to around the same as the Ridge or Lasso model.

c. Boosting Importance Matrix

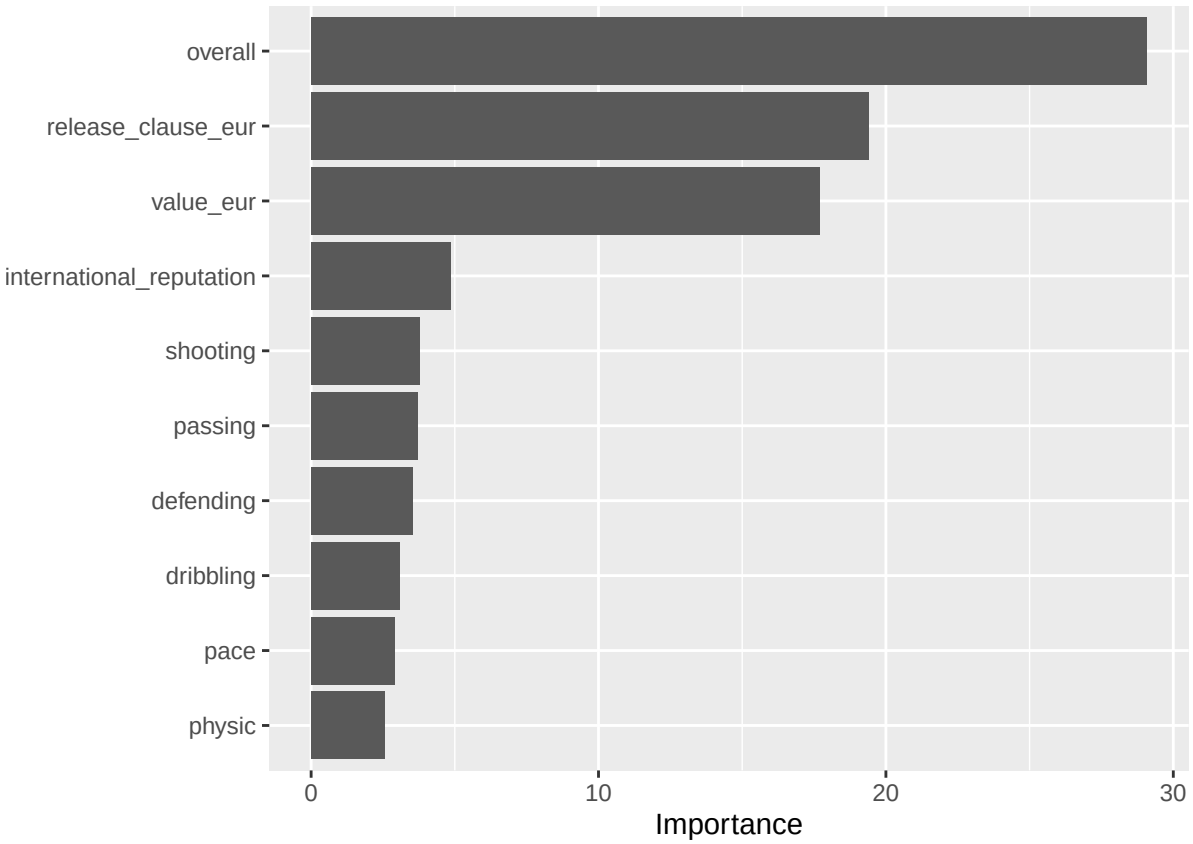
```
summary(boost.fifa)
```




```
##                                                 var    rel.inf
## overall                                     overall 29.0832646
## release_clause_eur               release_clause_eur 19.4002726
## value_eur                                 value_eur 17.6917898
## international_reputation international_reputation  4.8422910
## shooting                                   shooting  3.7717226
## passing                                     passing  3.6944958
## defending                                 defending  3.5302394
## dribbling                                 dribbling  3.0873987
## pace                                           pace  2.9206463
## physic                                       physic  2.5509682
## weight_kg                                 weight_kg  2.4614502
## potential                                 potential  2.1508553
```

```
## age                                    age  1.9273563
## height_cm                          height_cm  1.8015598
## weak_foot                          weak_foot  0.7019662
## skill_moves                      skill_moves  0.3837232
```

```
vip::vip(boost.fifa)
```



Compared to the previous Importance Matrix, there are no values along the variables but just relative importance. But similarly to the previous Importance Matrices, the top three variables are overall rating, release clause, and value.

Question 5: Fit a XGBoost Regression

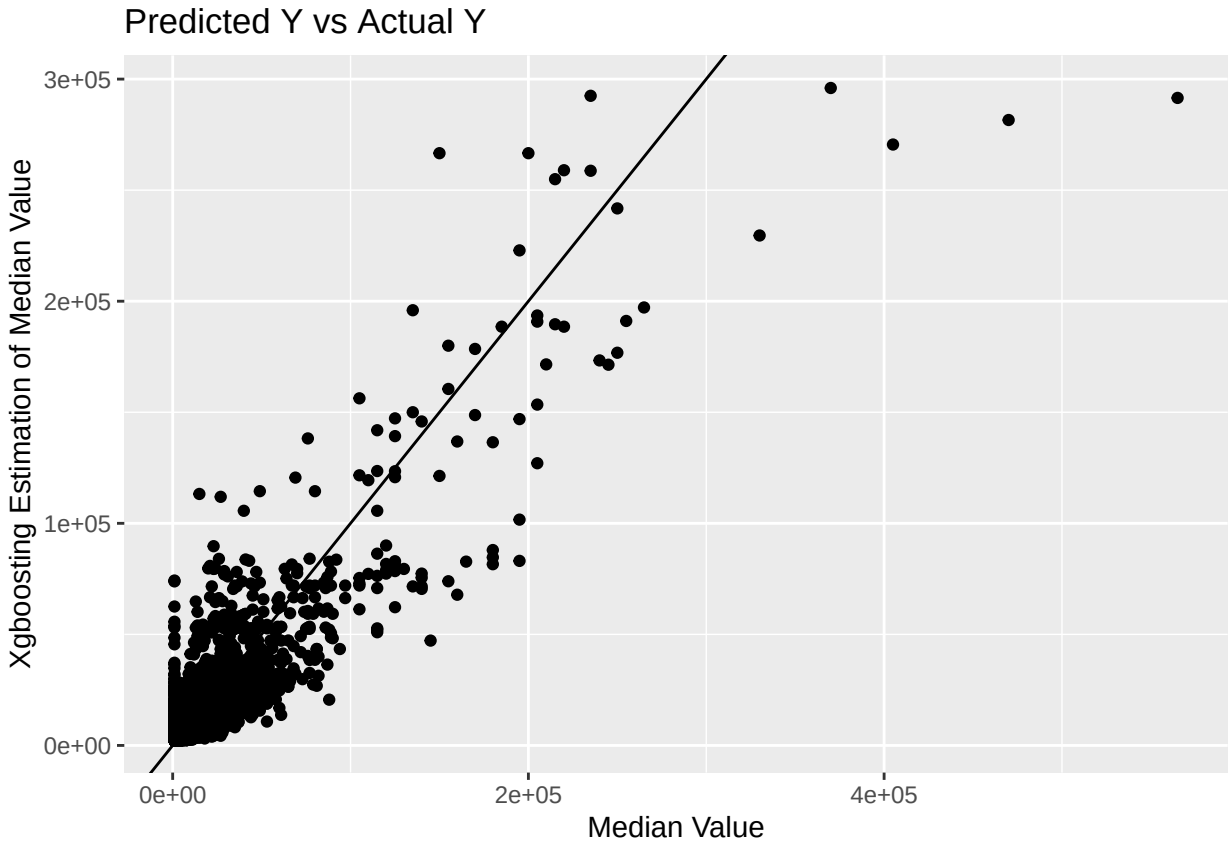
a. Plot Predicted Y vs Actual Y

```
# Putting data in matrix form
```

```
set.seed(1)
fifa.y.train = as.matrix(bag_train[,'wage_eur'])
fifa.x.train = as.matrix(bag_train[!names(bag_train) %in% c('wage_eur')])
dtrain = xgb.DMatrix(data = fifa.x.train, label = fifa.y.train)
fifa.x.test = as.matrix(bag_test[!names(bag_train) %in% c('wage_eur')])
fifa.xgb = xgboost(data = dtrain, max_depth = 2, eta = .1, nrounds = 40,
                   lambda = 0,
                   print_every_n = 10,
                   objective = 'reg:linear')
```

```
## [1]  train-rmse:22311.746094
## [11] train-rmse:12668.156250
## [21] train-rmse:10114.684570
## [31] train-rmse:9453.249023
## [40] train-rmse:9184.758789
```

```
yhat.xgb = predict(fifa.xgb, fifa.x.test)
ggplot() +
  geom_point(aes(x = bag_test$wage_eur, y = yhat.xgb)) +
  geom_abline()+
  labs(x = 'Median Value', y = 'Xgboosting Estimation of Median Value') +
  ggtitle('Predicted Y vs Actual Y')
```



As you can see from this plot, the predictions are also generally very similar to the previous three. However we can see the differences by looking at the MSE's of the regression model.

b. XGboosting MSE

```
set.seed(1)
xgb.mse = mean((yhat.xgb - bag_test$wage_eur)^2)
xgb.mse
```

```
## [1] 110946744
```

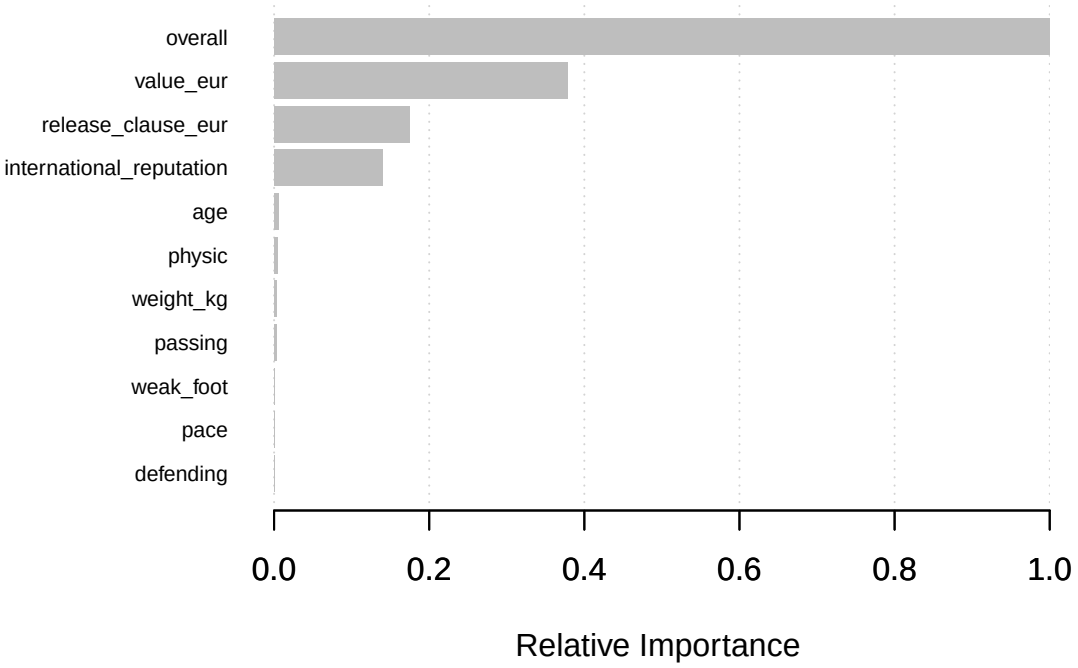
Looking at the MSE of the XGboosting model of 110,946,744, the MSE is lower than OLS and Ridge/Lasso but larger than Bagging/Random Forest. The XGboosting model is also better than the regular boosting model.

c. XGBoosting Importance Matrix

```
importance_xgb = xgb.importance(colnames(fifa.x.train), model = fifa.xgb)
importance_xgb
```

```
##                      Feature         Gain        Cover   Frequency
##  1:                  overall 0.5836994342 2.982096e-01 0.294117647
##  2:                value_eur 0.2211898064 1.499074e-01 0.100840336
##  3:       release_clause_eur 0.1017124957 2.975247e-01 0.268907563
##  4: international_reputation 0.0813669268 1.901867e-01 0.184873950
##  5:                      age 0.0035520415 3.233622e-04 0.033613445
##  6:                   physic 0.0024395346 4.676314e-04 0.025210084
##  7:                weight_kg 0.0021533642 1.099431e-03 0.025210084
##  8:                  passing 0.0019740493 6.198770e-02 0.042016807
##  9:                weak_foot 0.0006993563 1.094457e-04 0.008403361
## 10:                     pace 0.0006172675 1.691433e-04 0.008403361
## 11:                defending 0.0005957235 1.492441e-05 0.008403361
```

```
xgb.plot.importance(importance_xgb, rel_to_first = TRUE, xlab = "Relative Importance")
```



Looking at this importance matrix, we can see that this is much different than the other models we have been using. There are basically only four variables that are important, these being, overall rating, value, release clause, and international reputation.

Conclusion

From the Histograms and Plots of wage against the different X variables I have chosen, we can see that there is definitely a correlation between wage and the X's I have chosen. For example in Wage vs Overall, Shooting, and Potential Rating, there is an exponential correlation between the X's and wage. From the regression analysis, I found that the best regression for finding wage included release clause, value, overall rating, and potential rating. From this regression I found that all those variables were significant resulting in a good model. However, the OLS model had flaws. The biggest flaw was the fact that there was heterosketaticity. Heterosketaticity is not acceptable for a linear model because that means that there is no constant variance and one of the assumptions for OLS is that there is constant variance. The regression tree model was not great either because of the MSE of the model. The MSE of the regression tree model was higher than both the OLS and ridge/lasso models. The best model I found for estimating wages of professional soccer players was the Random Forest Regression model.

The Random Forest Regression model is very similar to the Bagging model with a slight difference in the number of parameters but the Random Forest has the lowest MSE of all the models I've tested. The lower the MSE, the lower the error which means that the Random Forest Regression model is the best model I have tested so far. Bagging and the XGboosting model came in a close second and third place.