# Chess Project

Taylor Anderson

8/8/2019

## Part 1

Chess

• Introduction: This is the data set I am most interested in. My goal with this data set is to be able to predict the winner of a chess game based on the color they are playing and the rating of each player. This way, someone could see the probability of them winning the game before it even starts. There is also data showing all moves in the game that I would like to use somehow but I am not sure how yet.

• Questions:

o Does the length of the game favor either player? o How much does difference in rating impact the probability to win? o Does the opening that is used impact win rates? o How much of an advantage does one color have over the other? o Are their longer games and more draws at higher or lower ratings?

• Explain: I will start by using logistic regression to make my prediction of who will win the game. The rating of each player will likely be the biggest indicator of who will win the game so I will start there and add other variables to see which are significant and which are not. I would also like to try some machine learning because it is likely that the relationships may not be linear. This is something that I don't have much experience with so I will have to do quite a bit of work on this portion over the next couple of weeks.

• Data:

o https://www.kaggle.com/datasnaek/chess o This dataset was collected in 2017. o The games were the 20,000 most recent games off of lichess.com o The data is clean but may need some work. For example, there are too many time increments to be useful so I may split them into the common categories of bullet, blitz, rapid, and classic.

• Libraries: ggplot2, plotly, broom, DataScienceR

• Types of Plots: Scatter, correlation, diagnostic plots, bar plots, heat map

## Part 2

### a.

*Data importing and cleaning steps are explained in the text and in the DataCamp exercises.(Tell me why you are doing the data cleaning activities that you perform). Follow a logical process.*

First, I checked the data to see if there were any N/A values that I needed to worry about. Luckily, the data did not contain any.

```
apply(games, 2, function(x) any (is.na(x)))
```

```
##             id          rated     created_at   last_move_at          turns
##          FALSE          FALSE          FALSE          FALSE          FALSE
## victory_status         winner increment_code       white_id   white_rating
##          FALSE          FALSE          FALSE          FALSE          FALSE
##       black_id   black_rating          moves    opening_eco   opening_name
##          FALSE          FALSE          FALSE          FALSE          FALSE
##    opening_ply
##          FALSE
```

Next, I looked through the columns to make sure the variables made sense.

- id
  - While there are 20,058 games in this data set there are 19,113 unique ids. This is a little concerning because there is likely duplicated data in this dataset. I removed these duplicates with the dplyr package.

```
library(dplyr)

games <- distinct(games)
```

- rated
  - The "rated" column showed that there were 4 factors when really it should only be a boolean. I found that the column had TRUE, FALSE, True, and False. So I created a new column called NewRated that converted them all to TRUE and FALSE.

```
games$NewRated <- ifelse(games$rated==FALSE|games$rated=="False", FALSE, TRUE
)
```

### b.

*With a clean dataset, show what the final data set looks like. However, do not print off a data frame with 200+ rows; show me the data in the most condensed form possible.*

```
str(games)
```

```
## 'data.frame':    19629 obs. of  17 variables:
##  $ id            : Factor w/ 19113 levels "005lW0Xz","009mKOEz",..: 15433
10012 10832 9930 3074 11034 13555 14110 5539 3341 ...
##  $ rated         : Factor w/ 4 levels "False","FALSE",..: 2 4 4 4 4 2 4 2
```

```
4 4 ...
##  $ created_at    : num  1.5e+12 1.5e+12 1.5e+12 1.5e+12 1.5e+12 ...
##  $ last_move_at  : num  1.5e+12 1.5e+12 1.5e+12 1.5e+12 1.5e+12 ...
##  $ turns         : int  13 16 61 61 95 5 33 9 66 119 ...
##  $ victory_status: Factor w/ 4 levels "draw","mate",..: 3 4 2 2 2 1 4 4 4
2 ...
##  $ winner        : Factor w/ 3 levels "black","draw",..: 3 1 3 3 3 2 3 1 1
3 ...
##  $ increment_code: Factor w/ 400 levels "0+12","0+13",..: 111 299 299 188
248 20 20 115 103 20 ...
##  $ white_id      : Factor w/ 9438 levels "--jim--","-l-_jedi_knight_-l-",.
.: 1288 17 3824 2018 6058 8569 1468 2023 2526 2023 ...
##  $ white_rating  : int  1500 1322 1496 1439 1523 1250 1520 1413 1439 1381
...
##  $ black_id      : Factor w/ 9331 levels "-0olo0-","-l-_jedi_knight_-l-",.
.: 12 7749 12 135 135 3066 2041 7867 2041 5691 ...
##  $ black_rating  : int  1191 1261 1500 1454 1469 1002 1423 2108 1392 1209
...
##  $ moves         : Factor w/ 18920 levels "a3 b6 d4 c5 Nf3 Bb7 e3 cxd4 exd
4 d6 c4 d5 Nc3 dxc4 Bxc4 Bxf3 Qxf3 Nh6 Bxh6 Nd7",..: 1823 4372 10772 3199 122
01 8481 2951 17373 10374 9595 ...
##  $ opening_eco   : Factor w/ 365 levels "A00","A01","A02",..: 249 72 172 2
42 193 97 240 72 202 73 ...
##  $ opening_name  : Factor w/ 1477 levels "Alekhine Defense",..: 1388 752 6
51 993 784 1293 92 755 504 1142 ...
##  $ opening_ply   : int  5 4 3 3 5 4 10 5 6 4 ...
##  $ NewRated      : logi  FALSE TRUE TRUE TRUE TRUE FALSE ...
```

### c.

*What do you not know how to do right now that you need to learn to import and cleanup your dataset?*

There are a few variables that I am not sure how to use very well right now. For example; moves, opening_eco, opening_name, and opening_ply. I may need to do additional work to make these variables useful to me.

### d.

*Discuss how you plan to uncover new information in the data that is not self-evident.*

My first plan is to try to predict wins and losses based on the ratings of the two players. Then to improve that prediction I will test how much some other variables play into this. The variables that come to mind are the color they are playing and the increment. I would also like to see if there are any changes to the style of game depending on rating. Are there certain openings that are popular at different ratings? Does the average number of turns change? Are there more draws in higher or lower rated games? etc.

## e.

*What are different ways you could look at this data to answer the questions you want to answer?*

I think that logistic and linear regression are going to be the main two ways to answer the questions that I have. There may be some k-nearest applications that I could use as well but I haven't thought that through quite as much yet.

## f.

*Do you plan to slice and dice the data in different ways, create new variables, or join separate data frames to create new summary information? Explain.*

I had some concern about games that are rated (meaning that they impact your rating) vs. games that are not rated. I was worried that people may not play their best in unrated games or they may try to play in a way that doesn't reflect their ability (blind-folded, trick games, etc.) I am going to analize this and see if there is a major difference. I am hoping I can keep them in because unrated games are a large part of the data set.

I have already made a new variable in the data cleaning process called NewRated but that wasn't for analytical reasons. One new variable I am sure that I will make is to break up the increments. Increments are in the format X+Y where X is the amount of minutes in the game and Y is the number of seconds you get after each move. For example, a common increment is 15+10 where each player starts with 15 minutes and gets 10 seconds added to the timer after each move. There are 400 different increments in this data set which is too many to work with. I plan on breaking these into 3 to 5 groups ranging from short games to long games. I may do a similar thing with openings. There are 1,477 different openings in the data set but I may break them into groups for the first 1 or two moves so it is more manageable.

## g.

*How could you summarize your data to answer key questions?*

Like I have mentioned in other question, by using plots, linear regression, and possibly knn models I will be able to see what kinds of variables will impact the results of a game of chess. After finding which variables have the largest impact on the game, I will create training and testing data sets to see if I can accuratly predict the outcome of games which would answer my question.

## h.

*What types of plots and tables will help you to illustrate the findings to your questions?*

I will use histograms to summarize some of the different variables in the data set. Scatter plots will be used to show correlation along with linear models. I have seen a few clever uses of heat maps for chess data that I would like to try as well.

### i.

*Do you plan on incorporating any machine learning techniques to answer your research questions? Explain.*

I will be creating a knn model to see how it compares to a linear model. I am not sure how well it will work with the data set so I am excited to learn more about the methods advantages and limitations.

## Part 3

```
games$WhiteWins <- ifelse(games$winner=="white", TRUE, FALSE)
games$RatingDifference <- games$white_rating - games$black_rating

# pulls out the number before the plus sign in increment_code which represent
s minutes
games$Minutes <- as.integer(sub('\\+.*', '', games$increment_code))

games$Length <- factor(ifelse(games$Minutes < 3, "Bullet",
                      ifelse(games$Minutes >= 3 & games$Minutes < 10, "Blitz"
,
                       ifelse(games$Minutes>=10 & games$Minutes < 60, "Rapid"
, "Classic"))))

games_no1500 <- games[!(games$white_rating == 1500 | games$black_rating == 15
00),]
games_nodraw <- games_no1500[!(games_no1500$winner == "draw"),]
```

While working with the data set some more a decided to make a few other variables:

- WhiteWins: This was to create a boolean to determine if white won or not. This will be used later for the logistic regression.
- RatingDifference: This variable shows the rating difference between the two players.
- Minutes: This is the number of minutes each game is limited to for each player after removing increments.
- Length: This puts each game into the four standard chess game length categories.

I also created two subsets of the main dataset:

- games_no1500: I will discribe why in the next section with histograms but this has removed all games where either player was rated 1,500.
- games_nodraw: This is a subset of games from games_no1500 where there are no draws for a more clear binary dataset.

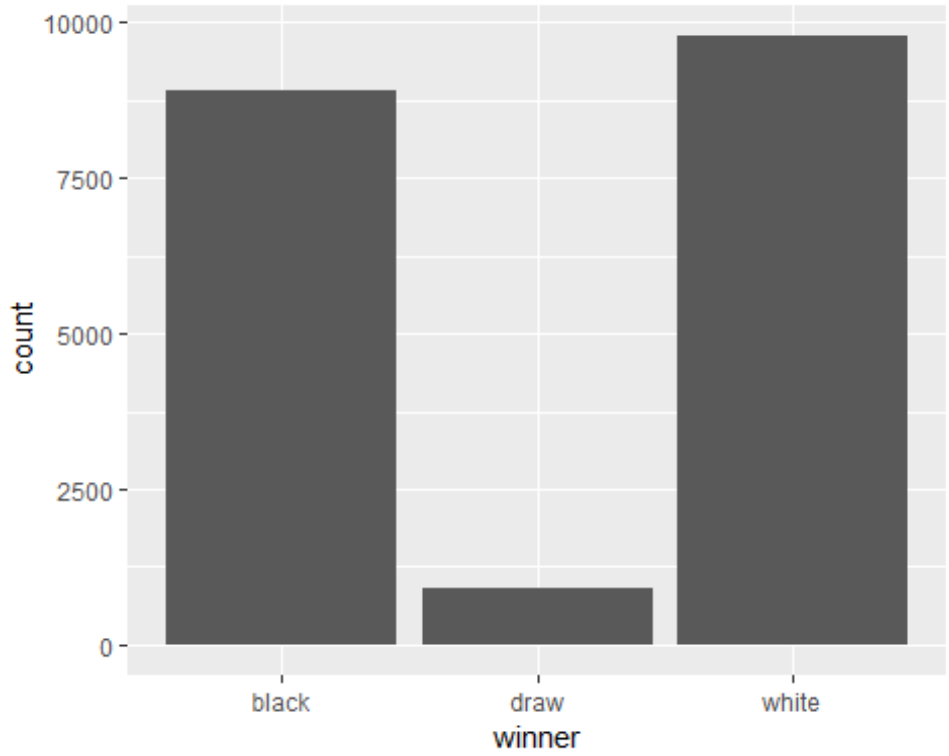
```
library(ggplot2)

## Registered S3 methods overwritten by 'ggplot2':
##   method         from
##   [.quosures     rlang
```

```
##   c.quosures     rlang
##   print.quosures rlang
```

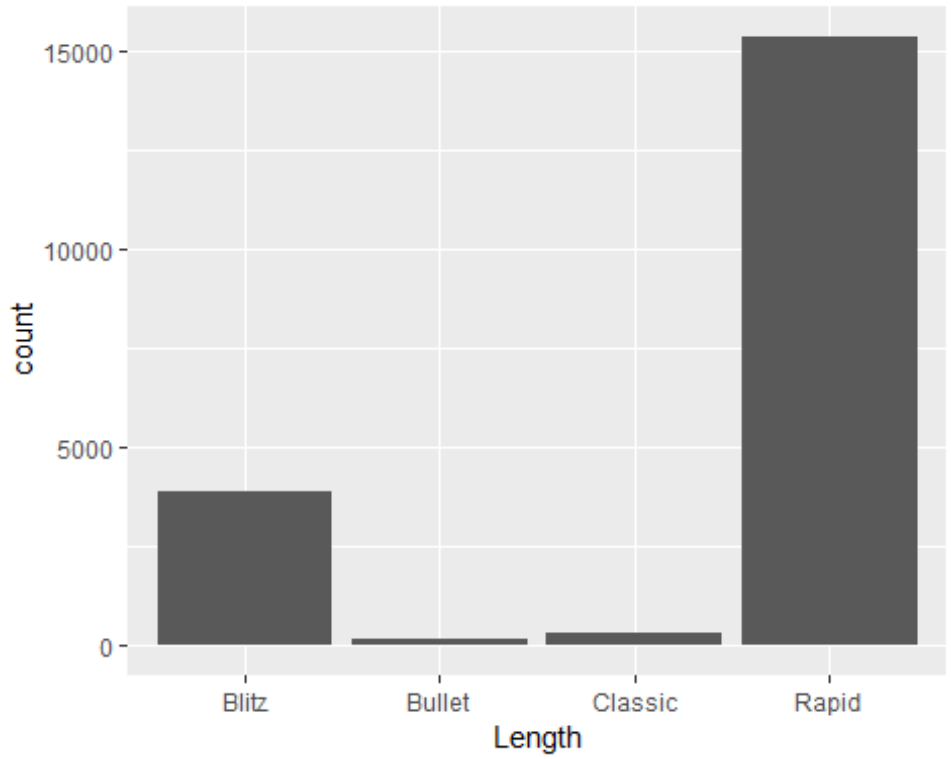
```
ggplot(games, aes(x = winner)) + geom_histogram(stat="count")
```

```
## Warning: Ignoring unknown parameters: binwidth, bins, pad
```

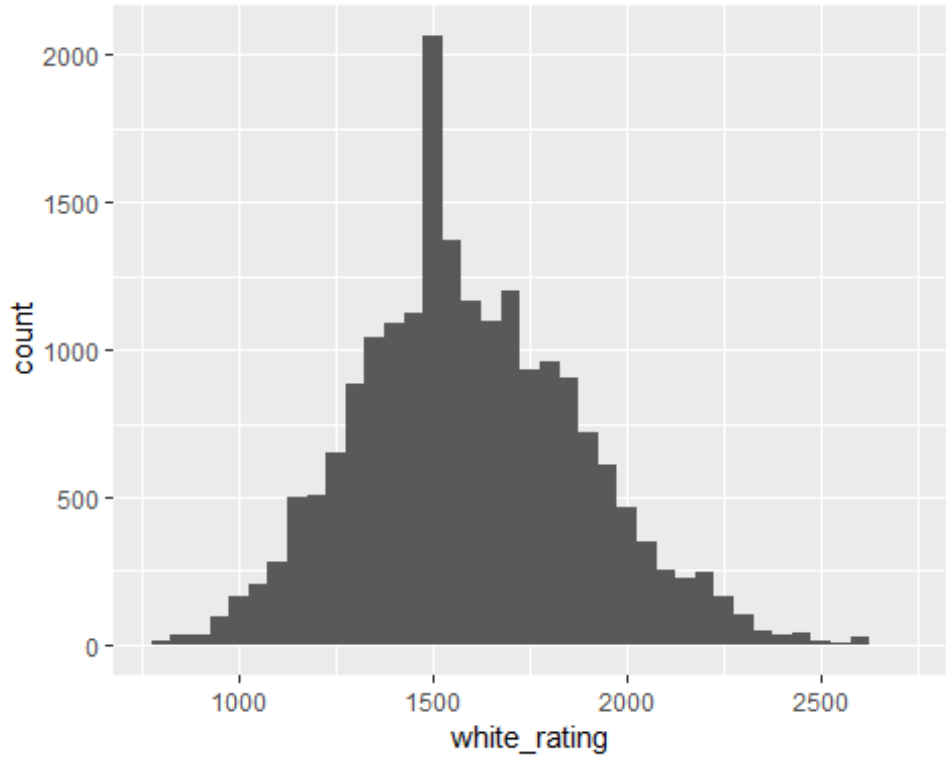



```
ggplot(games, aes(x = Length)) + geom_histogram(stat="count")
```

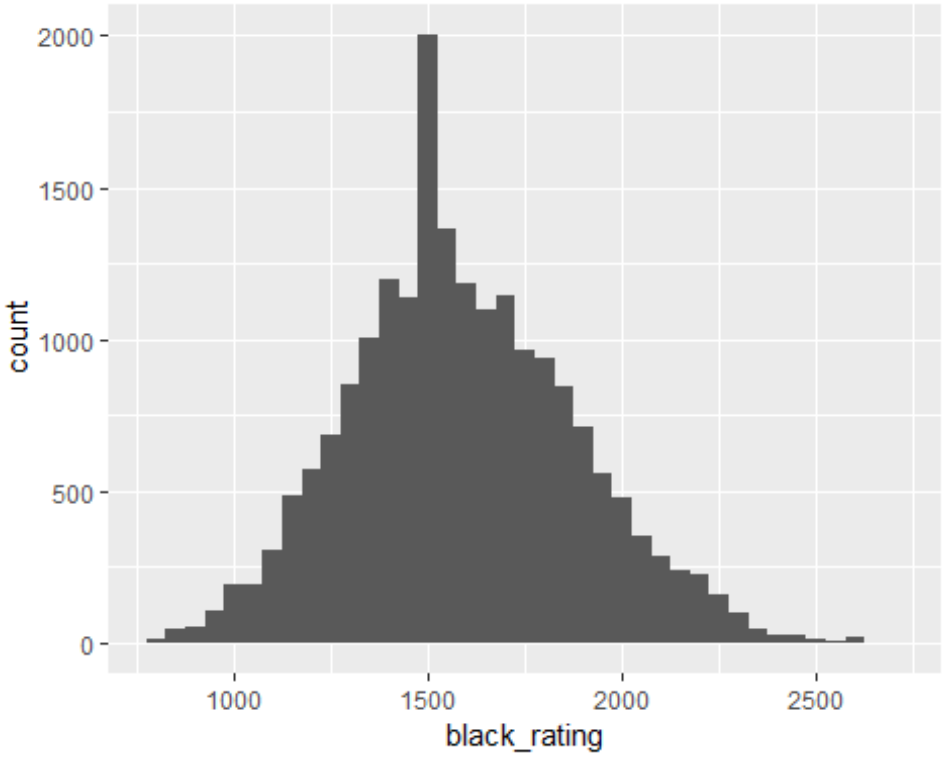
```
## Warning: Ignoring unknown parameters: binwidth, bins, pad
```




```
ggplot(games, aes(x = white_rating)) + geom_histogram(binwidth = 50)
```



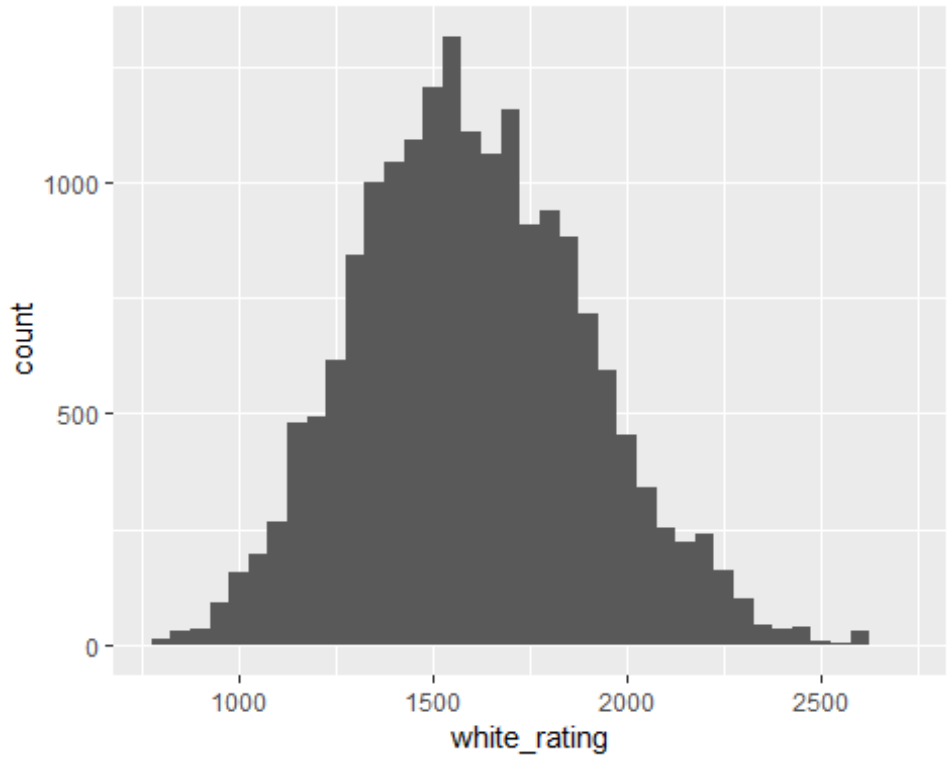

```
ggplot(games, aes(x = black_rating)) + geom_histogram(binwidth = 50)
```



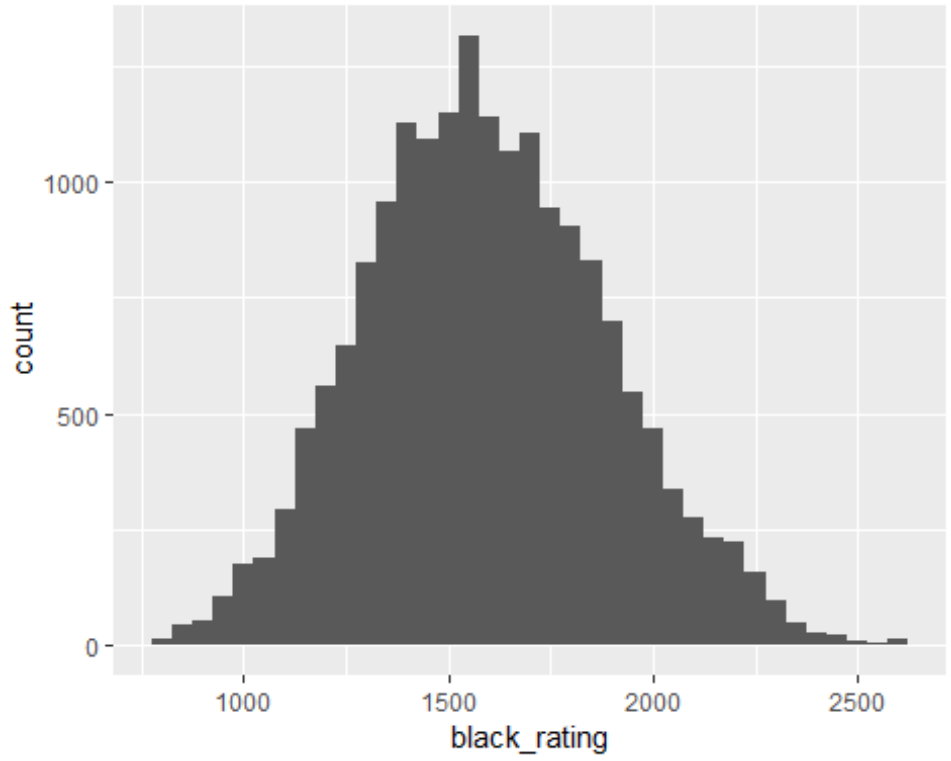

The winner distribution is as I would have expected. Draws are not very common and white has an advantage over black. The length distribution is also as I expected for the most part. Rapid is the most common and also has the most lengths that fit into it.

We can see that the distribution of ratings in the data is realitively normal but with a slight right skew and an unusual peak at 1,500 rating. I did some research on this and found that when you start a lichess account you are put at 1,500 rating because they don't know your rating. This was a big concern to me because if we have this many games of people with 1,500 rating, they are really people that we have no idea what their rating is. Below I look at the distribution when we use the data where I have removed all games with a 1,500 player.

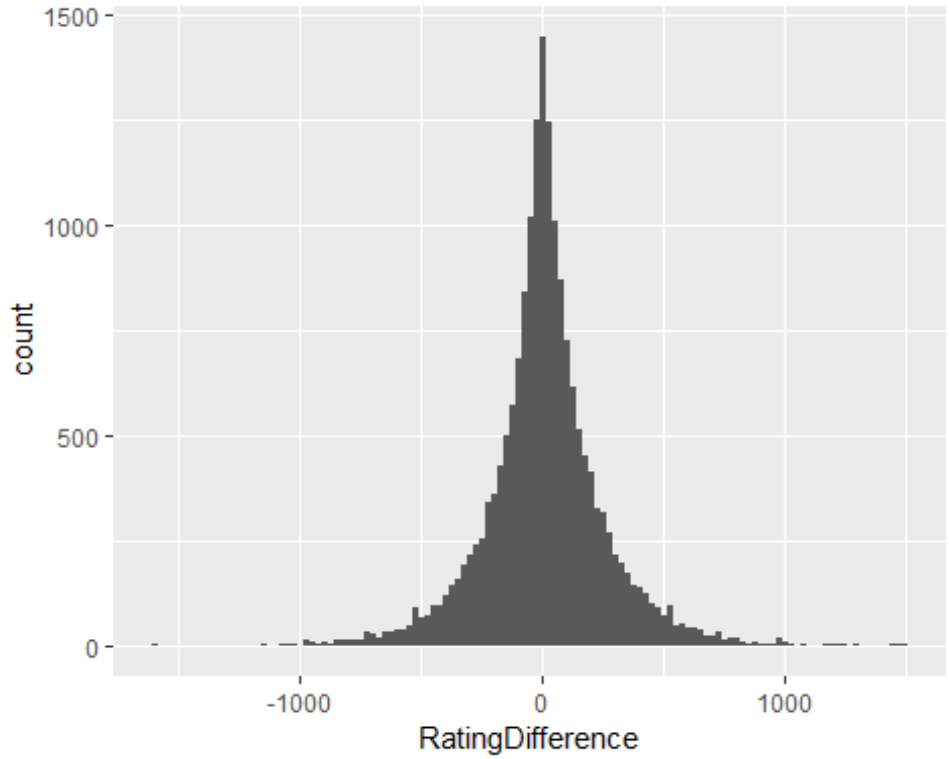
```
ggplot(games_no1500, aes(x = white_rating)) + geom_histogram(binwidth = 50)
```




```
ggplot(games_no1500, aes(x = black_rating)) + geom_histogram(binwidth = 50)
```



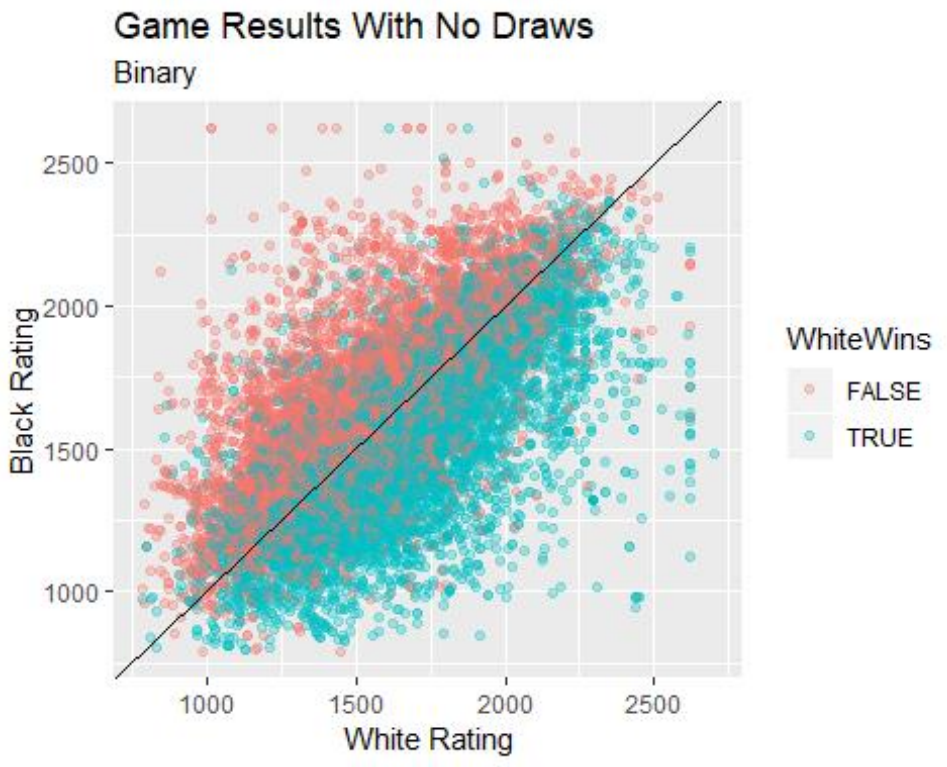
```
ggplot(games_no1500, aes(x = RatingDifference)) + geom_histogram(binwidth = 25)
```




Here we can see that without the people at 1,500 the distribution looks much more like what we would expect to see. Because we can't know the number of games anyone has played there is always a risk that some that is really 1,000 started at 1,500 and played one game that brought them down to 1,450 or something. That means there is likely some bias towards that 1,500 rating still but we got quite a bit of it out.

The second problem I have come across is the last histogram which shows the rating difference between the two players. If games were randomly assigned on lichess we would expect to see a more normal distribution. Lichess does not randomly assign games though, as we can see, they try match players against a similarly rated player. This will make predicting a winner much more difficult.

```
ggplot(games_nodraw, aes(x = white_rating, y = black_rating, col = WhiteWins)) + geom_point(alpha = 0.3) +
  labs( title = "Game Results With No Draws",subtitle = "Binary", x= "White Rating", y = "Black Rating") +
  geom_abline(intercept = 0, slope = 1)
```




This plot shows us White Rating vs. Black Rating and who wins the game with draws removed. I have also plotted a line where White Rating = Black Rating. Player rating is clearly an important factor in determining who wins the game. So much so, that using this line alone would be a relatively good way to predict who wins each game. This plot also shows that lichess.com tends to match players against similar ratings.

```
games.glm <- glm(WhiteWins ~ white_rating + black_rating + turns, data = game
s_no1500, family = binomial)

summary(games.glm)
```

```
##
## Call:
## glm(formula = WhiteWins ~ white_rating + black_rating + turns,
##     family = binomial, data = games_no1500)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -2.775  -1.076  -0.222   1.086   2.893
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)   4.083e-01  9.786e-02   4.172 3.02e-05 ***
## white_rating  3.882e-03  9.324e-05  41.635  < 2e-16 ***
## black_rating -3.983e-03  9.365e-05 -42.532  < 2e-16 ***
## turns        -4.572e-03  4.868e-04  -9.392  < 2e-16 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 25169  on 18155  degrees of freedom
## Residual deviance: 22283  on 18152  degrees of freedom
## AIC: 22291
##
## Number of Fisher Scoring iterations: 4
```

```
modelChi <- games.glm$null.deviance - games.glm$deviance

chidf <- games.glm$df.null - games.glm$df.residual

chisq.prob <- 1 - pchisq(modelChi, chidf)
chisq.prob
```

```
## [1] 0
```

```
R2.hl <- modelChi/games.glm$null.deviance
R2.hl
```

```
## [1] 0.1146861
```

I worked through several different logistic models but this one (at first) was the best. From the summary, we can see that white_rating, black_rating, and turns are all significant. Surprisingly, with all of the work done to break down the lengths of games, they had no significant impact in the outcome of a game. I also calculated chisq.prob which tells us the p-value is near 0. and Hosmer and Lemeshow's measure which is 0.115.

```
games2.glm <- glm(WhiteWins ~ RatingDifference, data = games_no1500, family =
binomial)

summary(games2.glm)
```

```
##
## Call:
## glm(formula = WhiteWins ~ RatingDifference, family = binomial,
##     data = games_no1500)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.8092  -1.0865  -0.2224   1.0941   2.8848
##
## Coefficients:
##                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)      -3.669e-02  1.601e-02  -2.292   0.0219 *
## RatingDifference  3.932e-03  8.813e-05  44.612   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 25169  on 18155  degrees of freedom
## Residual deviance: 22382  on 18154  degrees of freedom
## AIC: 22386
##
## Number of Fisher Scoring iterations: 4
```

```
modelChi <- games2.glm$null.deviance - games2.glm$deviance

chidf <- games2.glm$df.null - games2.glm$df.residual

chisq.prob <- 1 - pchisq(modelChi, chidf)
chisq.prob
```

```
## [1] 0
```

```
R2.hl <- modelChi/games2.glm$null.deviance
R2.hl
```

```
## [1] 0.1107213
```

I wasn't very satisfied with the first model so I decided to try a simple second one. This model uses an interaction instead by only looking at the difference between the two ratings. The AIC and Hosmer and Lemeshow's measure of this model is very similar to the last one so it may be a better option due to its simplicity.

```
library(caTools)
```

```
## Warning: package 'caTools' was built under R version 3.6.1
```

```
library(class)
```

```
## Warning: package 'class' was built under R version 3.6.1
```

```
# create data frame with ratings and WhiteWins
games_knn = games_nodraw[c("white_rating", "black_rating", "WhiteWins")]
# split into train and test
split <- sample.split(games_knn$WhiteWins, SplitRatio = 0.8)
training <- subset(games_knn, split == "TRUE")
testing <- subset(games_knn, split == "FALSE")

games.predictions <- knn(training, testing, training$WhiteWins, k = sqrt(nrow
(training)))

confmatrix <- table(Actual_Value=testing$WhiteWins, Predicted_Value = games.p
redictions)
confmatrix
```

```
##             Predicted_Value
## Actual_Value FALSE TRUE
```

```
##        FALSE   976  679
##        TRUE    602 1206
```

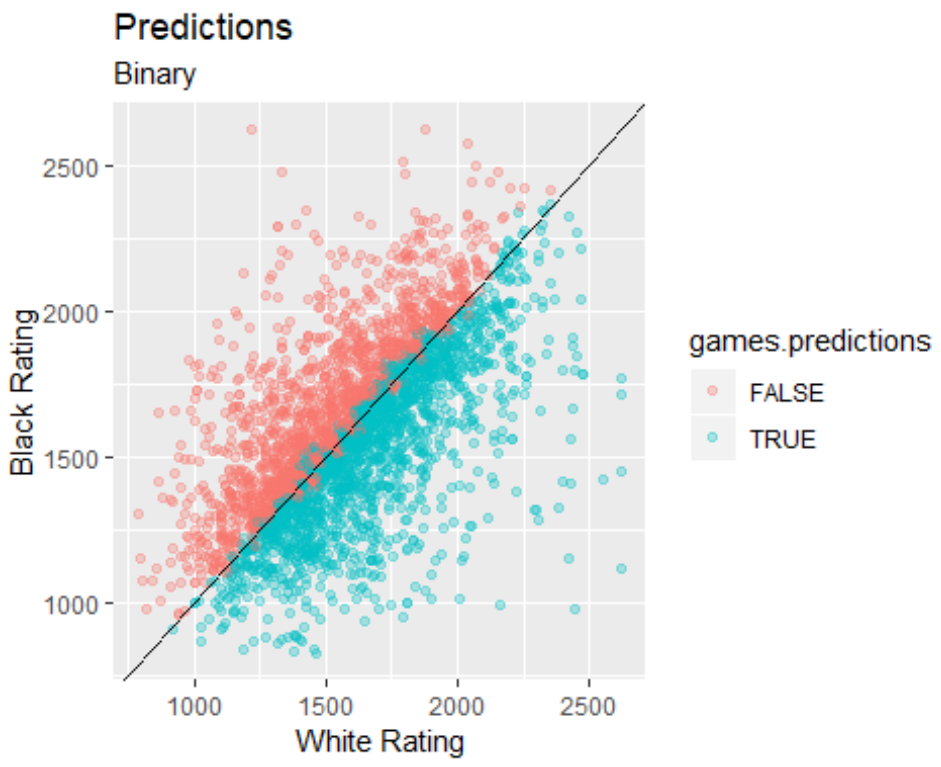
```
(confmatrix [[1,1]] + confmatrix[[2,2]]) / sum(confmatrix)
```

```
## [1] 0.6300895
```

This is my attempt at a knn model to predict who wins each game. For this version I used the dataset with draws removed. I created a data frame with ratings from both players and who won. Then I split them into test and train subsets. The confusion matrix shows that the model is only around 65% accurate and seems to favor white winning in its predictions. At first I was disappointed by this result but thinking again about how lichess works, many of these games will have players very close to eachother in rating. If two players are similar rating it would be best to assume white will win as they have the advantage in chess.

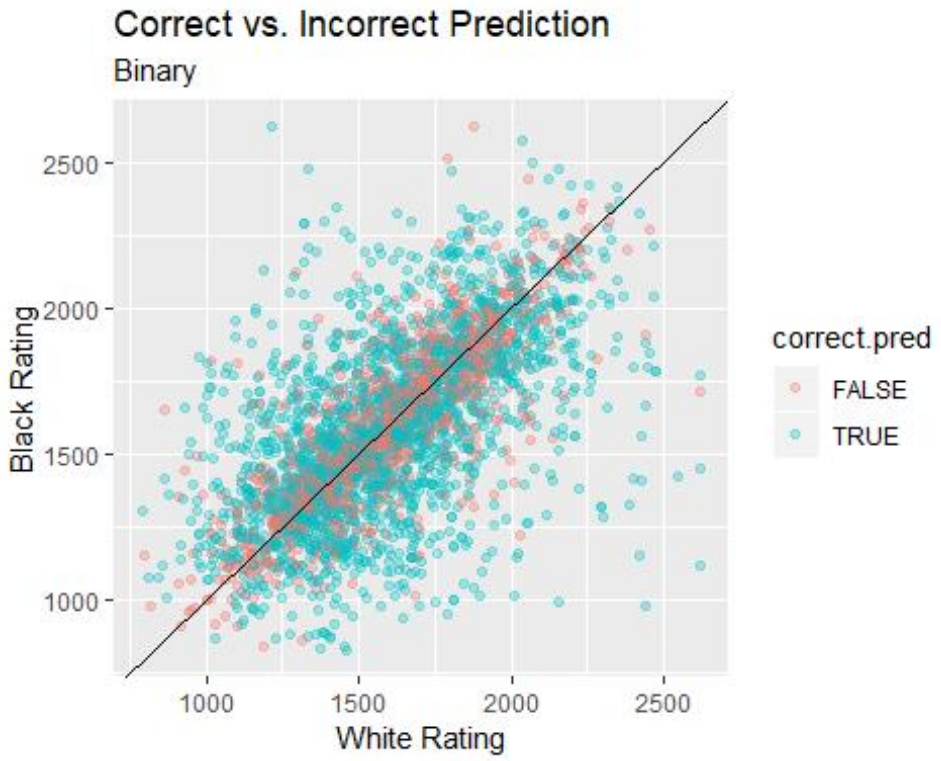
```
comp <- testing
comp$prediction <- data.frame(games.predictions)
comp$correct.pred <- ifelse(comp$prediction == comp$WhiteWins, TRUE, FALSE)

ggplot(comp, aes(x = white_rating, y = black_rating, col = games.predictions)
) + geom_point(alpha = 0.3) +
  labs( title = "Predictions", subtitle = "Binary", x= "White Rating", y = "B
lack Rating") +
  geom_abline(intercept = 0, slope = 1)
```



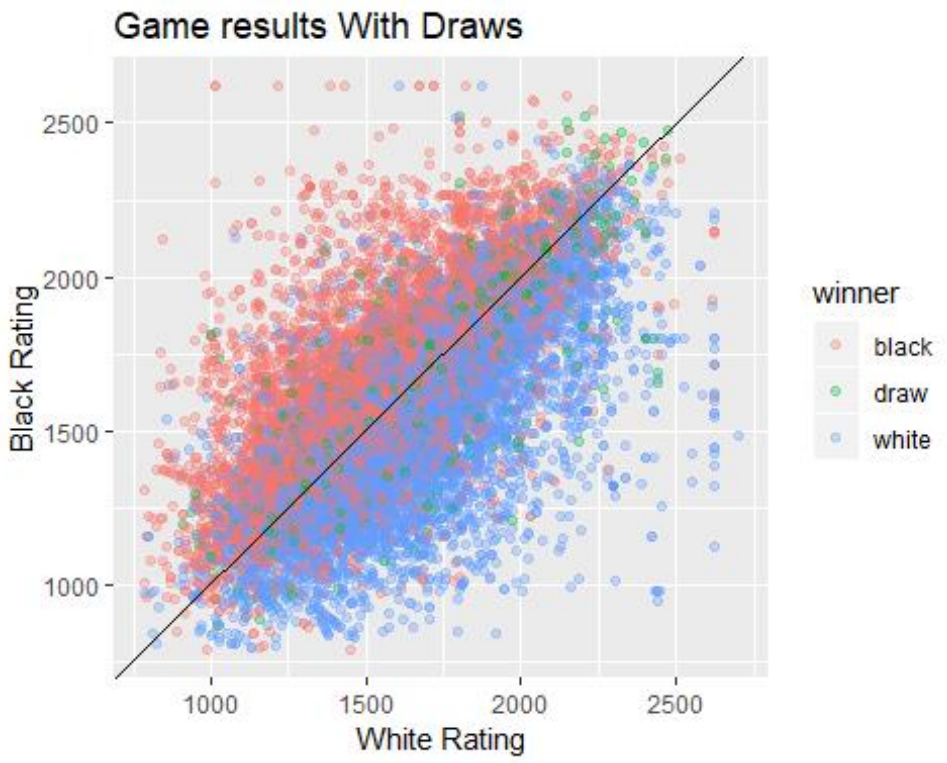

```
ggplot(comp, aes(x = white_rating, y = black_rating, col = correct.pred)) + g
eom_point(alpha = 0.3) +
```

```
  labs( title = "Correct vs. Incorrect Prediction", subtitle = "Binary", x= "
White Rating", y = "Black Rating") +
  geom_abline(intercept = 0, slope = 1)
```




These two plots may be my two favorite in my analysis. The first plot titled "Predictions" shows if the model predicted white to win or not. We can see that the white_rating = black_rating line almost exactly splits the models predictions with a slight favor to white when the ratings are almost exactly the same. The second plot titled "Correct vs. Incorrect Predictions" shows where the model was right and where it was wrong. This makes it clear to see that the further from the white_rating = black_rating line we get, the more accurate the model. Near the line though, the model is unsure of who will win the game.

```
ggplot(games_no1500, aes(x = white_rating, y = black_rating, col = winner)) +
geom_point(alpha = 0.3) +
  labs( title = "Game results With Draws", x= "White Rating", y = "Black Rati
ng", legend = "Winner") +
  scale_fill_discrete(name = "Winner") +
  geom_abline(intercept = 0, slope = 1)
```




I did also run a model with draws back in the data set. You will see that it does not work very well.

```
# Create a data frame that includes white wins, black wins, and draws.
games_tri_knn = games_no1500[c("white_rating", "black_rating", "winner")]
# set white, black, and draw, to 0,1, and 2 respectively
games_tri_knn$winner = ifelse(games_tri_knn$winner == "white", 0, ifelse(game
s_tri_knn$winner == "black", 1, 2))

split_tri <- sample.split(games_tri_knn$winner, SplitRatio = 0.8)
training_tri <- subset(games_tri_knn, split == "TRUE")
testing_tri <- subset(games_tri_knn, split == "FALSE")

games.tri.knn <- knn(training_tri, testing_tri, training_tri$winner, k = sqrt
(nrow(training_tri)))

confmatrix <- table(Actual_Value=testing_tri$winner, Predicted_Value = games.
tri.knn)
confmatrix

##              Predicted_Value
## Actual_Value    0    1    2
##            0 1219  554    0
##            1  658 1006    0
##            2   99   90    0

(confmatrix [[1,1]] + confmatrix[[2,2]]) / sum(confmatrix)
```

```
## [1] 0.6136238
```

This example confirms that there are too few draws in the data set to use knn to predict them. Every draw in the plot is surrounded by many more wins and loses so the draws could never outnumber their neighbors.

In summary, these models did not work as well as I had hoped but I guess that is how data science goes sometimes. This model fails most often near the white_rating = black_rating line and because lichess tries to start games where rating is similar the success rate is likely lower than it really is. If the games were randomly assigned and there were more games far from the line then it would appear to be more accurate. All this model really confirms is the rating is incredibly important in chess and if two people are a similar rating then pick white to win. I was hoping to be able to use moves in this analysis but it turned out that my R knowledge was too limited for me to make much use of those variables. I hope that in the future I can come back and take another look at this data and try to use some more of it.