Board Game Analysis

Team Maximus

Chris Dinkins, Jackson Hanchek, Harrison Peloquin, Hunter Sawyer

2022-11-27

Library Installations

```
library("pander")
library("ggplot2")
library("DT")
library("corrplot")
library("datarium")
bgg_df <- read.csv("./bgg_dataset.csv", sep = ";")
```

Explanation of Data Source

```
sub_df <- bgg_df[1:15,]
pander(sub_df)
```

Table 1: Table continues below

ID	Name	Year.Published	Min.Players
174430	Gloomhaven	2017	1
161936	Pandemic Legacy: Season 1	2015	2
224517	Brass: Birmingham	2018	2
167791	Terraforming Mars	2016	1
233078	Twilight Imperium: Fourth Edition	2017	3
291457	Gloomhaven: Jaws of the Lion	2020	1
182028	Through the Ages: A New Story of Civilization	2015	2
220308	Gaia Project	2017	1
187645	Star Wars: Rebellion	2016	2
12333	Twilight Struggle	2005	2
193738	Great Western Trail	2016	2
115746	War of the Ring: Second Edition	2012	2
162886	Spirit Island	2017	1
169786	Scythe	2016	1
84876	The Castles of Burgundy	2011	2

Table 2: Table continues below

Max.Players	Play.Time	Min.Age	Users.Rated	Rating.Average	BGG.Rank
4	120	14	42055	8,79	1
4	60	13	41643	8,61	2
4	120	14	19217	8,66	3
5	120	12	64864	8,43	4
6	480	14	13468	8,70	5
4	120	14	8392	8,87	6
4	120	14	23061	8,43	7
4	150	12	16352	8,49	8
4	240	14	23081	8,42	9
2	180	13	40814	8,29	10
4	150	12	29457	8,30	11
4	180	13	13725	8,49	12
4	120	13	25722	8,35	13
5	115	14	57871	8,24	14
4	90	12	46507	8,13	15

Table 3: Table continues below

Complexity.Average	Owned.Users	Mechanics
3,86	68323	Action Queue, Action Retrieval, Campaign / Battle Card Driven, Card Play Conflict Resolution, Communication Limits, Cooperative Game, Deck Construction, Deck Bag and Pool Building, Grid Movement, Hand Management, Hexagon Grid, Legacy Game, Modular Board, Once-Per-Game Abilities, Scenario / Mission / Campaign Game, Simultaneous Action Selection, Solo / Solitaire Game, Storytelling, Variable Player Powers
2,84	65294	Action Points, Cooperative Game, Hand Management, Legacy Game, Point to Point Movement, Set Collection, Trading, Variable Player Powers
3,91	28785	Hand Management, Income, Loans, Market, Network and Route Building, Score-and-Reset Game, Tech Trees / Tech Tracks, Turn Order: Stat-Based, Variable Set-up
3,24	87099	Card Drafting, Drafting, End Game Bonuses, Hand Management, Hexagon Grid, Income, Set Collection, Solo / Solitaire Game, Take That, Tile Placement, Turn Order: Progressive, Variable Player Powers
4,22	16831	Action Drafting, Area Majority / Influence, Area-Impulse, Dice Rolling, Follow, Grid Movement, Hexagon Grid, Modular Board, Trading, Variable Phase Order, Variable Player Powers, Voting

Complexity.Average	Owned.Users	Mechanics
3,55	21609	Action Queue, Campaign / Battle Card Driven, Communication Limits, Cooperative Game, Critical Hits and Failures, Deck Construction, Grid Movement, Hand Management, Hexagon Grid, Legacy Game, Line of Sight, Once-Per-Game Abilities, Scenario / Mission / Campaign Game, Simultaneous Action Selection, Solo / Solitaire Game, Variable Player Powers
4,41	26985	Action Points, Auction/Bidding, Auction: Dutch, Card Drafting, Events, Income, Take That
4,35	20312	End Game Bonuses, Hexagon Grid, Income, Modular Board, Network and Route Building, Solo / Solitaire Game, Tech Trees / Tech Tracks, Turn Order: Pass Order, Variable Player Powers, Variable Set-up, Victory Points as a Resource
3,71	34849	Area Majority / Influence, Area Movement, Area-Impulse, Delayed Purchase, Dice Rolling, Hand Management, Team-Based Game, Variable Player Powers
3,59	56219	Action/Event, Advantage Token, Area Majority / Influence, Campaign / Battle Card Driven, Dice Rolling, Hand Management, Simulation, Simultaneous Action Selection, Sudden Death Ending, Tug of War
3,71	35804	Deck Bag and Pool Building, Hand Management, Ownership, Point to Point Movement, Rondel, Set Collection, Track Movement, Variable Set-up
4,14	22281	Area Majority / Influence, Area Movement, Campaign / Battle Card Driven, Card Play Conflict Resolution, Deck Bag and Pool Building, Dice Rolling, Hand Management, Hidden Movement, Movement Points, Simulation, Team-Based Game
4,01	38254	Action Retrieval, Area Majority / Influence, Campaign / Battle Card Driven, Cooperative Game, Events, Hand Management, Modular Board, Set Collection, Simultaneous Action Selection, Solo / Solitaire Game, Variable Player Powers
3,41	75640	Area Majority / Influence, Card Play Conflict Resolution, Force Commitment, Grid Movement, Hexagon Grid, King of the Hill, Movement Points, Moving Multiple Units, Narrative Choice / Paragraph, Solo / Solitaire Game, Tech Trees / Tech Tracks, Variable Player Powers

Complexity.Average	Owned.Users	Mechanics
3,00	63058	Dice Rolling, Grid Coverage, Hexagon Grid, Set Collection, Tile Placement, Turn Order: Stat-Based, Worker Placement with Dice Workers

Domains
Strategy Games, Thematic Games
Strategy Games, Thematic Games
Strategy Games
Strategy Games
Strategy Games, Thematic Games
Strategy Games, Thematic Games
Strategy Games
Strategy Games
Thematic Games
Strategy Games, Wargames
Strategy Games
Thematic Games, Wargames
Strategy Games
Strategy Games
Strategy Games

https://www.kaggle.com/datasets/andrewmvd/board-games?resource=download

Our dataset is maintained on kaggle.com, a platform for data science related tools and discussion. Our data set, titled "Board Games", was uploaded to the site by a senior data scientist from the Hospital Israelita Albert Einstein, the highest prestige hospital in Latin America. This data set, however, does not involve medical data as it was a leisure project. The data set, which contains over twenty thousand different board games, was taken from the website BoardGameGeek in February of 2021. BoardGameGeek is currently the largest board game ranking platform, and it contains many different stats about over one hundred thousand board games. The twenty thousand board games in the data set are a subset of the larger hundred thousand. They are ranked in popularity, which requires user accounts to vote on them. Alongside popularity and name, the data includes publishing year, minimum and maximum player count, play length, suggested age, and rating from a scale of one through ten.

Initial Observations

When our group first attempted to handle our dataset, we realized the number of rows was so large that our computers had difficulty processing it. In fact, the dataset's size was gargantuan enough to give us issues when trying to create tables using pander. We have decided the best course of action to solve this predicament was to take a smaller portion of the data and use that to represent the dataset. By choosing the first 1,000 entries, we eliminate the bias that would come from handpicking certain data. We used this smaller data set for most testing purposes, but we did end up using the full data set. When we used the full data set we found a number of missing values in the number of owned users. We removed the contaminated observations.

After observing our data closely, we have noticed that there are a few cases of missing data. One such case can be found in the 'BoardGamesGeek ID', where board games can be found that do not have any IDs entered at all. We have decided that these null values do not justify throwing the entire rows away as the

'BoardGamesGeek ID' does not directly affect any way that we are analyzing or wanting to use this data. Instead, we have decided to leave these values empty.

One of the most prominent predicaments we have come across in the early stages of observing our data is how to handle noisy data. The first instances we noticed were in the 'Min Players' and the 'Max Players' columns. Here, we saw board games hold the value of 0, which seems meaningless and does not make much sense at first.However, considering the context of what the minimum players and the maximum players means, we theorize that a 0 value means the board game did not specify a minimum or a maximum number of players. With that said, we have decided to simply remove these rows from our data sets. This accounted for a very small amount of the data and would not massively effect analysis.

We have also seen that the 'Year Published' column has negative values in it, which is completely illogical given the context. At first, we considered that the negative signs were a mistake, and we could take the absolute value. However, values reach -3500 so an absolute value would result in a year that has not yet occurred at the time of observing the data. Perhaps negative values represent years from BC as opposed to AD. This could be the case, but it is difficult to imagine humans in 1300 BC passing time by playing a friendly match of Tic-Tac-Toe, so it would probably be in our best interest to consult a domain expert before making assumptions.

We also noticed that the average rating and average complexity were in decimal format with commas instead of decimal points. This is common practice in Brazil, where this data was collected. We changed the commas to decimals to make performing calculations easier.

The last thing we noticed was that there were some missing/negative values for the number of people who own some of the games. These also accounted for a tiny portion of the data so we simply omit the relevant observations.

```
#useful categorizations of column names

variables <- colnames(bgg_df)
numeric_variables <- variables[c(3,4,5,6,7,8,9,10,11,12)]
continues_variables <- numeric_variables[c(8,10)]
numeric_variables2 <- variables[c(3,4,5,6,7,8,10,12)]


#print(colnames(bgg_df))
#print(numeric_variables)




#create both data sets with noisy data removed
#removed_zero_playerCount_df is no min or max players of 0
removed_zero_playerCount_df <- subset(bgg_df,(bgg_df$Min.Players != 0 & bgg_df$Max.Players != 0))

#removed_zero_years_df is df with no published year of 0
removed_zero_years_df <- subset(bgg_df,(bgg_df$Year.Published > 0))

removed_both_df <- subset(bgg_df,(bgg_df$Min.Players != 0 & bgg_df$Max.Players != 0 &
                                  bgg_df$Year.Published > 0 & bgg_df$Owned.Users > 0))
above_2000_df <- na.omit(
subset(bgg_df,(bgg_df$Min.Players != 0 & bgg_df$Max.Players != 0 & bgg_df$Year.Published > 1990)))
```

Variable explanations

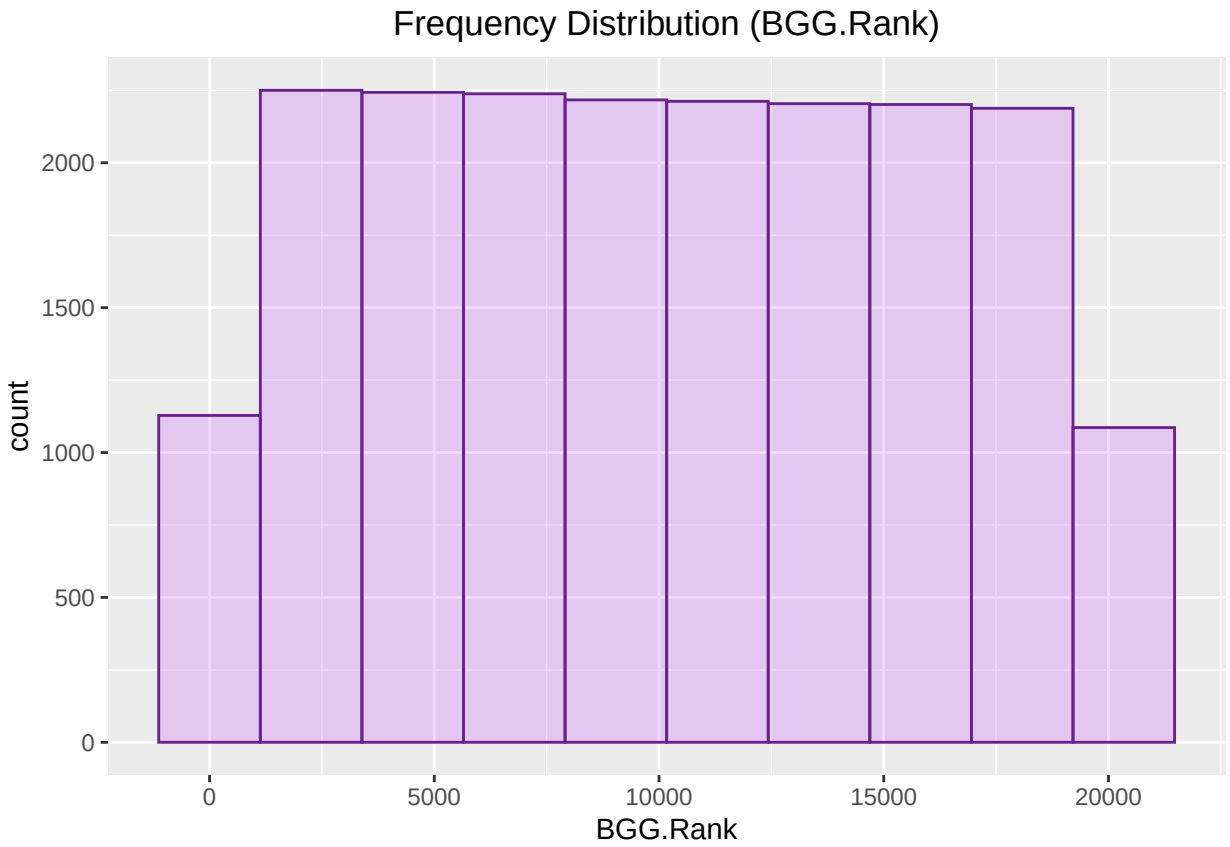
- ID represents the unique ID each board game is given in the BoardGamesGeek database. This is an integer.
- Name is the name of each board game in the database. This is a string.
- Year Published represents the (approximated in some cases) publishing year of the board game. Note that this is a signed integer, as dates * older than 1 AD are displayed as negatives.
- Min Players is the publisher suggested minimum number of players. This is an integer. Note that some board games display a suggested * minimum players of zero.
- Max Players is the publisher suggested maximum number of players. This is also an integer.
- Play Time is the publisher suggested average play time. This is an integer and is in minutes.
- Min Age is the publisher suggested minimum age of play. This is an integer.
- Users Rated is the number of users of BoardGameGeek who have given a rating to the board game. This is an integer.
- Rating Average is the average of all of the individual user ratings of the board game on the site. This is a float with two decimal places.
  - Note the commas instead of periods to denote decimal points, which is the common standard in Brazil.
- BGG Rank is the ranking of the board game on BoardGameGeek's "Best of All Time" list. This rank is assumed to be correlated with Rating Average and Users Rated. This is an integer. Note that each entry is unique, as there can only be one "best game of all time" or "second best game of all time", etc.

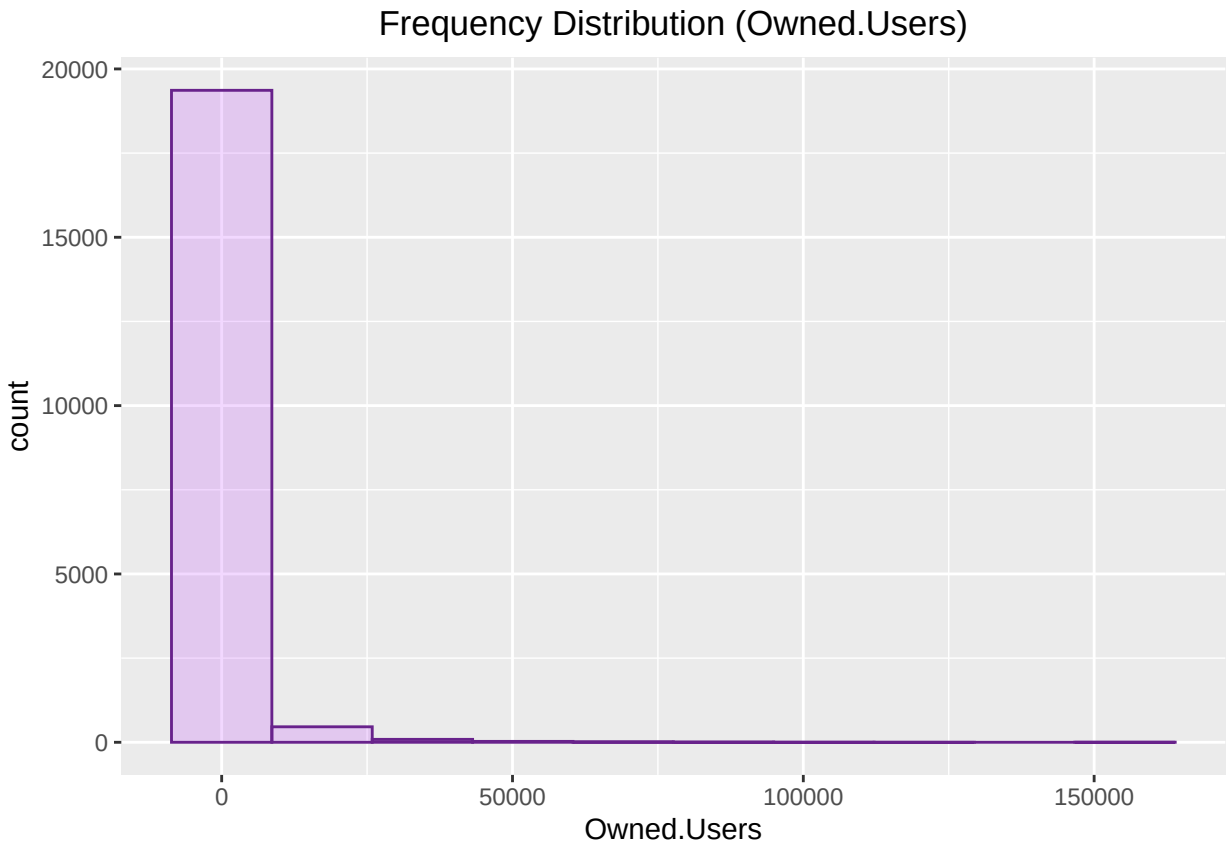
Histograms

The two histograms below are based upon the BGG rank and the User count respectively.
The BGG rank histogram is a flat distribution,which is what we would expect because of the fact that BGG ranks are unique to each game and assigned sequentially starting at 1.The histogram for the number of owned users looks like a Pareto distribution. It has the majority of games having player counter below 50,000 and the number of games with a higher player count decreasing as you get higher play counts.

```
# histogram creation

for(variable in continues_variables) {

  print(
    ggplot(removed_both_df, aes_string(x=variable))
    + geom_histogram(
      colour="darkorchid4", fill="darkorchid1", position="identity", bins=10, alpha=0.2
    )
    + ggtitle(paste("Frequency Distribution (", variable, ")", sep=""))
    + theme(plot.title=element_text(hjust = 0.5))
  )
}
```

```
## Warning: `aes_string()` was deprecated in ggplot2 3.0.0.
## i Please use tidy evaluation ideoms with `aes()`
```


Frequency Distribution (BGG.Rank)
count
2000
1500
1000
500
0
0
5000
10000
15000
20000
BGG.Rank




```
removed_both_df$Complexity.Average <- as.numeric(gsub(",",".",removed_both_df$Complexity.Average))
removed_both_df$Rating.Average <- as.numeric(gsub(",",".",removed_both_df$Rating.Average))


above_2000_df$Complexity.Average <- as.numeric(gsub(",",".",above_2000_df$Complexity.Average))
above_2000_df$Rating.Average <- as.numeric(gsub(",",".",above_2000_df$Rating.Average))

display_df_removed_outliers <- subset(bgg_df,(bgg_df$Min.Players != 0 &
          bgg_df$Max.Players != 0 &bgg_df$Max.Players < 20 ))
```

Correlation Analysis

We created a correlation chart for all of the numeric variables. We have also observed a few correlations that we found interesting upon our initial analysis. For example, we have found a very high correlation between the 'Rating Average' and the 'BGG Rank' columns. If we view the board games that are ranked near the top of 'BGG Rank', We can see the associated 'Rating Average' values are typically among the highest.

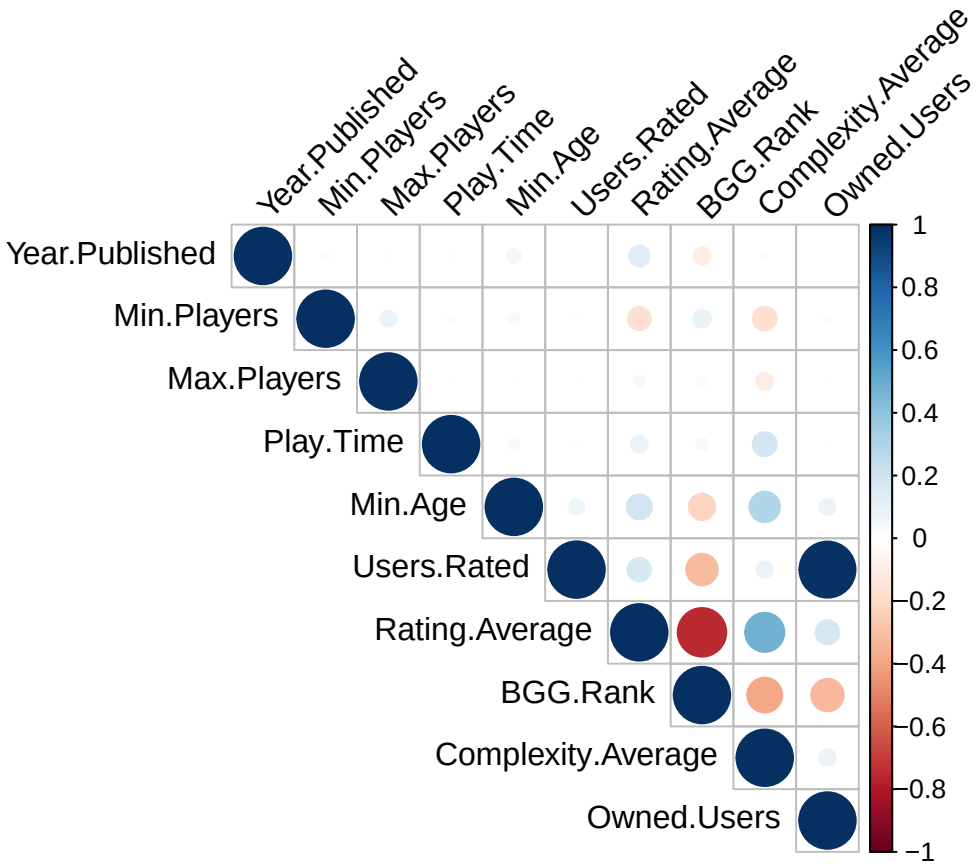
We also can see that there is a nearly 1 positive correlation for the number of users that own a game, and the number of users that rated a game. The reason for this is that the data set contains only ranked games. This is likely because of the likelihood that every person who rates a game on BGG, also checks on the website that they own the game.

We also notice that there is a very positive relation between the play time and the complexity. It seems likely that this is a causal relationship where the more complex a game is, the longer the play time becomes.

```
print(numeric_variables)
```

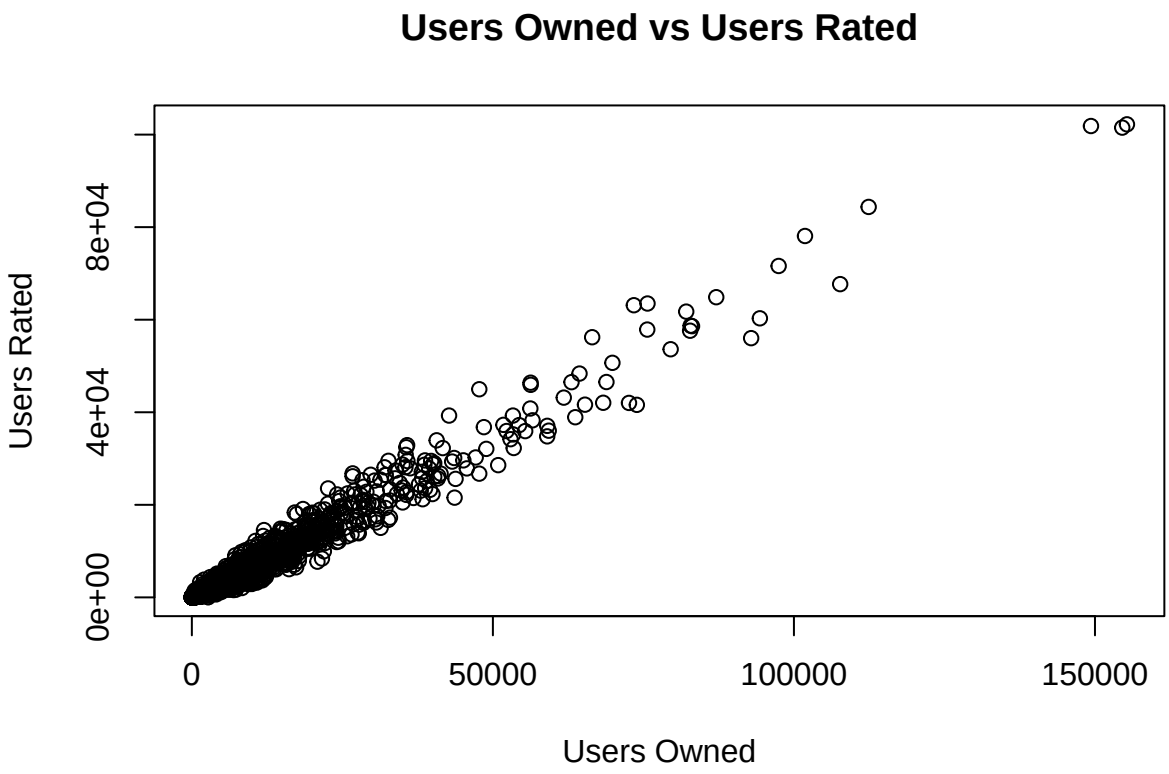
```
##  [1] "Year.Published"     "Min.Players"        "Max.Players"
##  [4] "Play.Time"          "Min.Age"            "Users.Rated"
##  [7] "Rating.Average"     "BGG.Rank"           "Complexity.Average"
## [10] "Owned.Users"
```

```
correlations <- cor(removed_both_df[numeric_variables[]])
corrplot(
  correlations,
  type="upper",
  tl.col="black",
  tl.srt=45
)
```

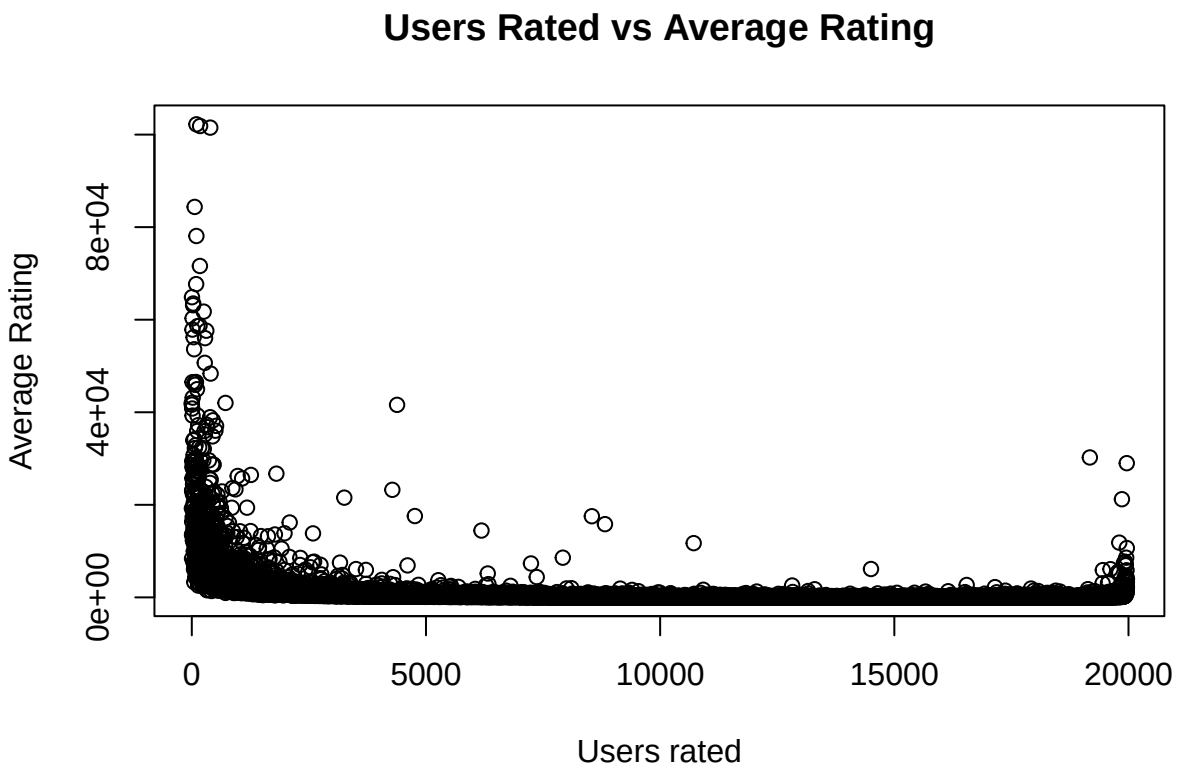


Scatter Plots

```
plot(bgg_df$Owned.Users,
  bgg_df$Users.Rated, type = "p",  xlim = NULL, ylim = NULL,log = "",
  main = "Users Owned vs Users Rated", sub = NULL, xlab = "Users Owned", ylab = "Users Rated",
ann = par("ann"), axes = TRUE)
```



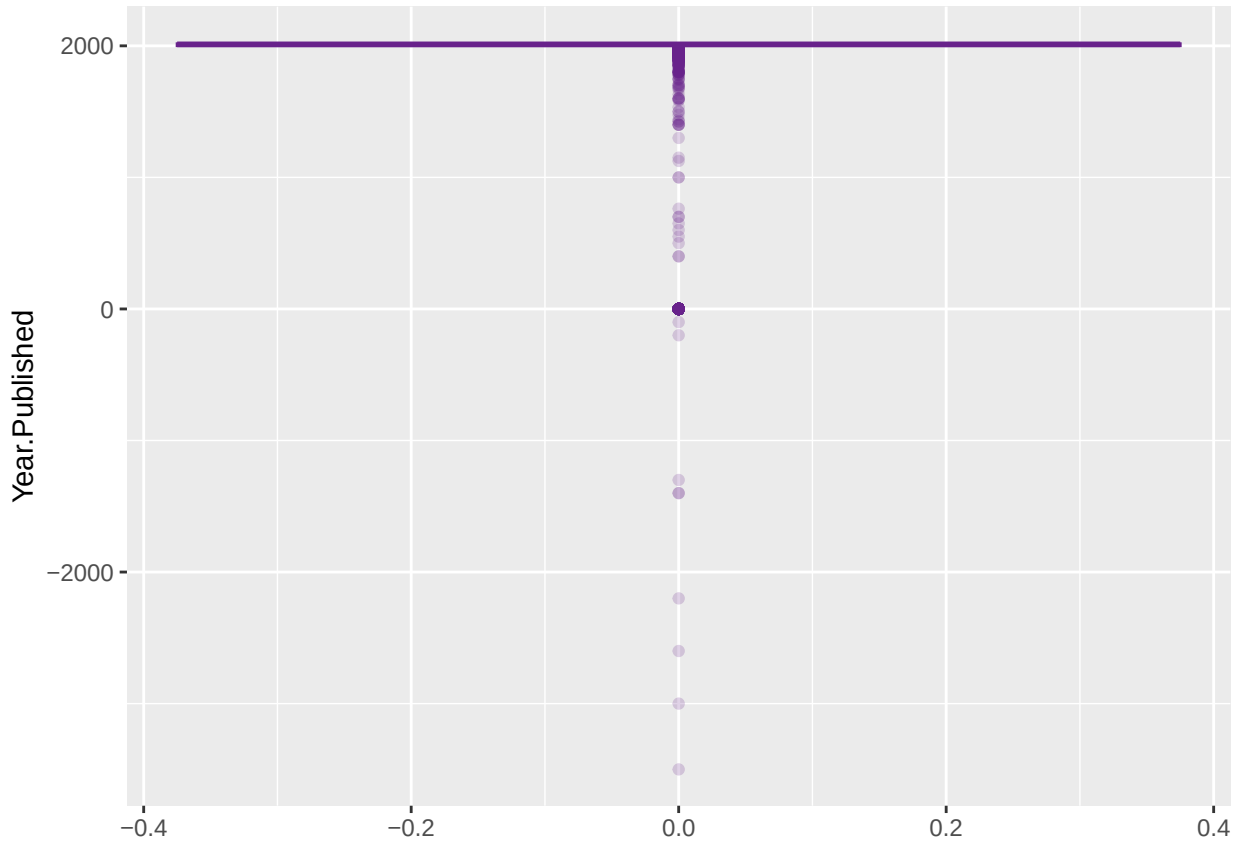

```
plot(removed_both_df$Users.Rated,
     removed_both_df$Rated.Average, type = "p",  xlim = NULL, ylim = NULL,
log = "", main = "Users Rated vs Average Rating", sub = NULL, xlab = "Users rated",
ylab = "Average Rating",ann = par("ann"), axes = TRUE)
```



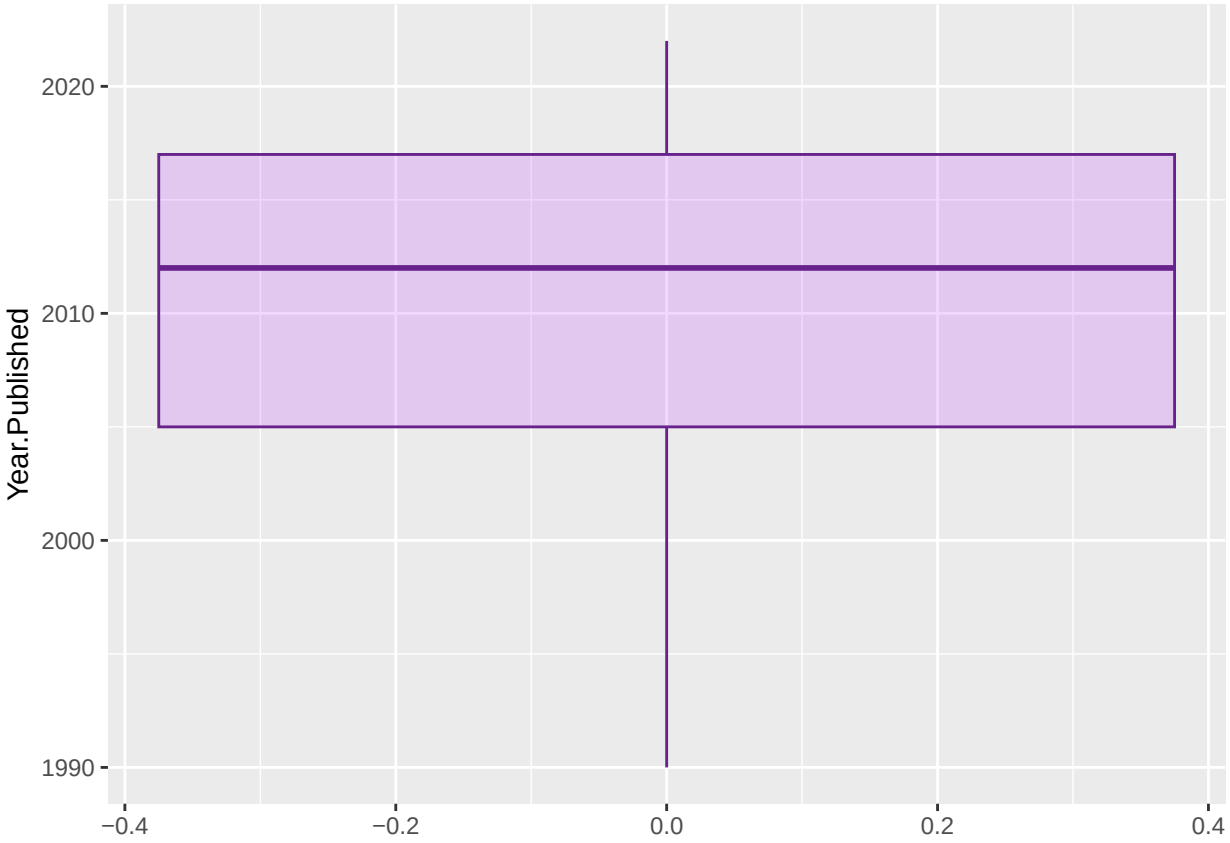
Box Plots

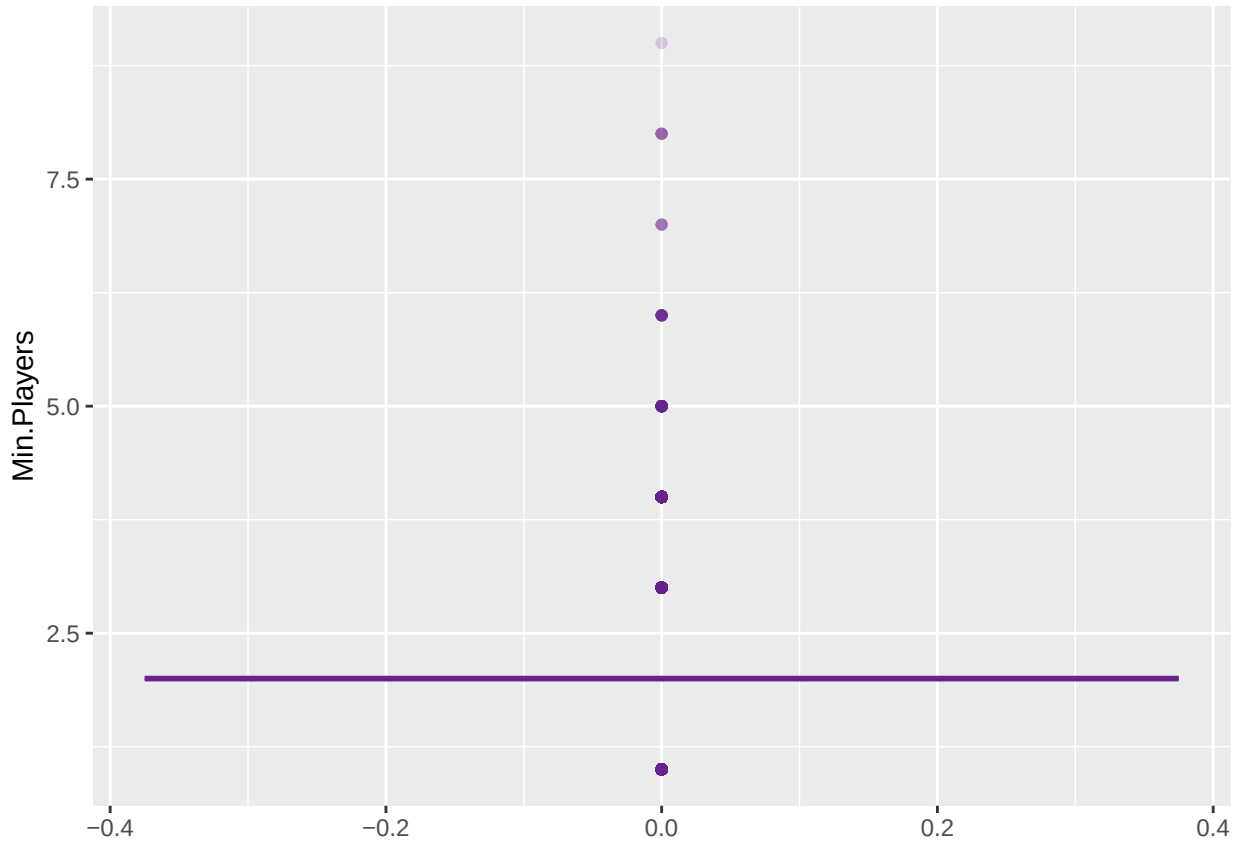
Note : We made two graphs for the year the games were published in. One with all the data included, and one with all games published before 1990 excluded. We did this so it is easier to see the distribution of publishing dates for the more modern games, but you can still see that there are some outliers way back in history.

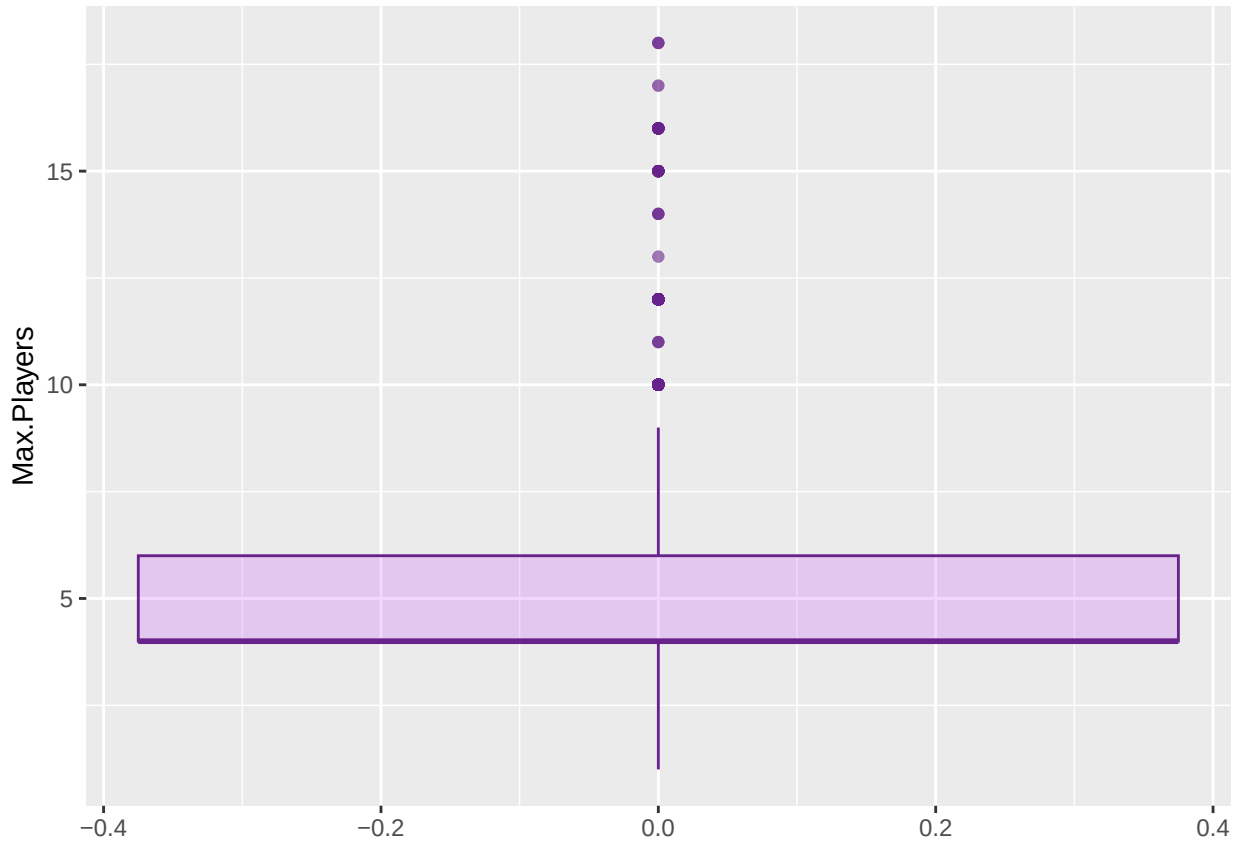
```
print(
    ggplot(display_df_removed_outliers, aes_string(y="Year.Published"))
    + geom_boxplot(colour="darkorchid4", fill="darkorchid1", alpha=0.2)
  )
```

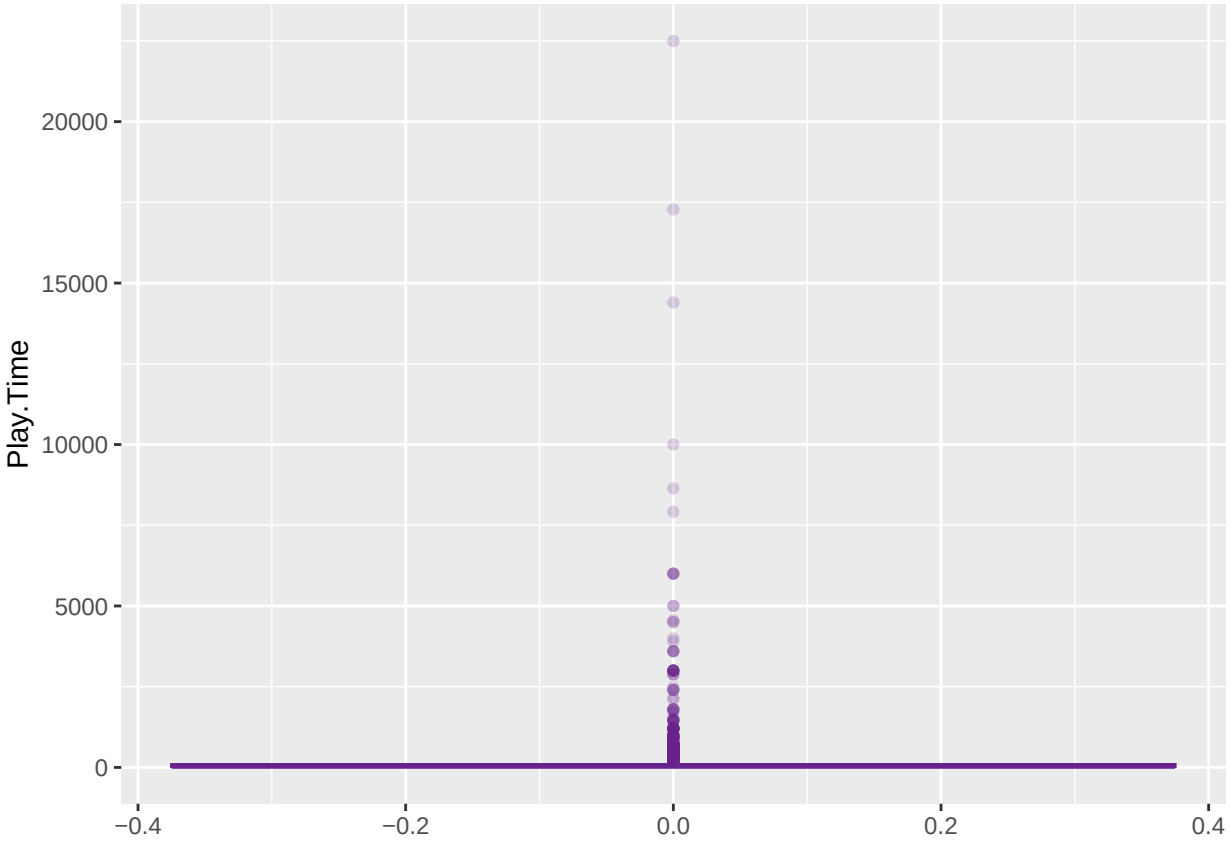


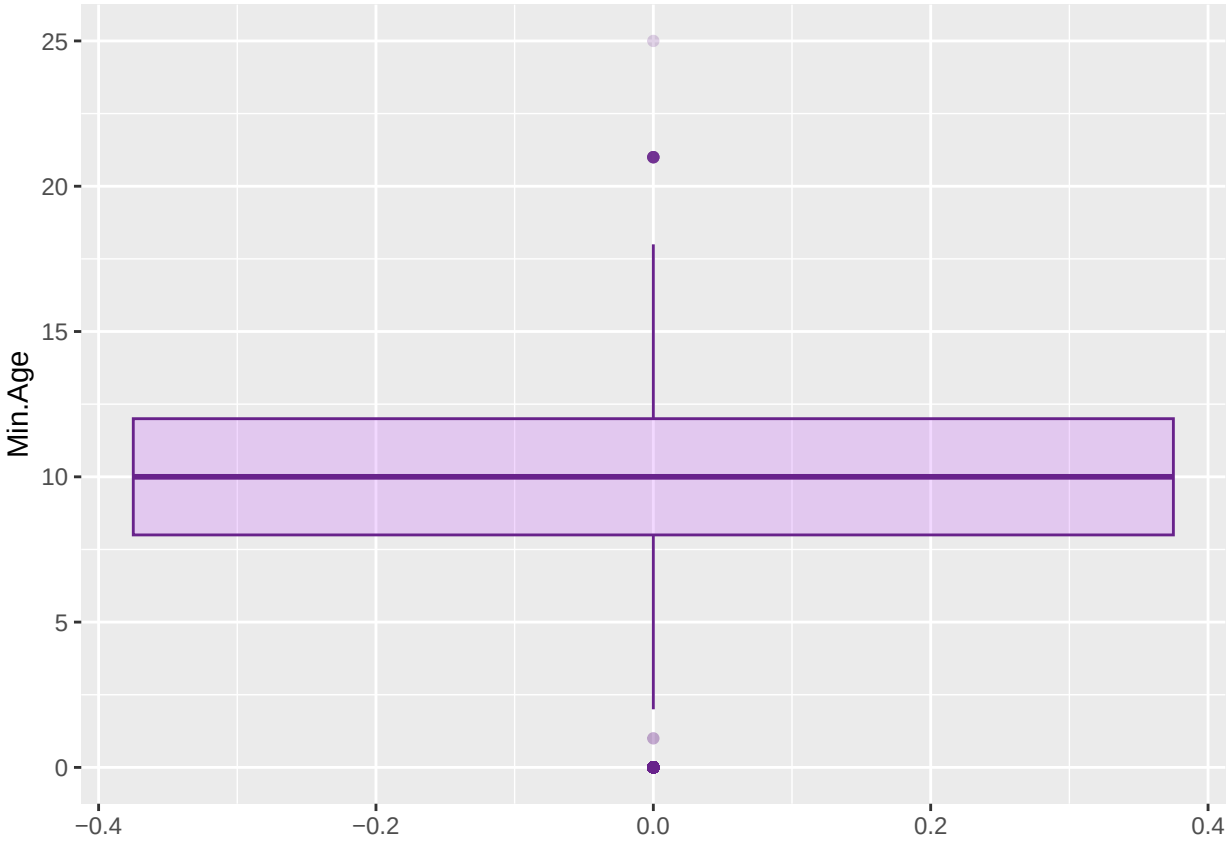

```
display_df_removed_outliers <-
subset(display_df_removed_outliers,(display_df_removed_outliers$Year.Published >= 1990))

for(var in numeric_variables2){
  group <- "color"
  print(
    ggplot(display_df_removed_outliers, aes_string(y=var))
    + geom_boxplot(colour="darkorchid4", fill="darkorchid1", alpha=0.2)
  )
}
```

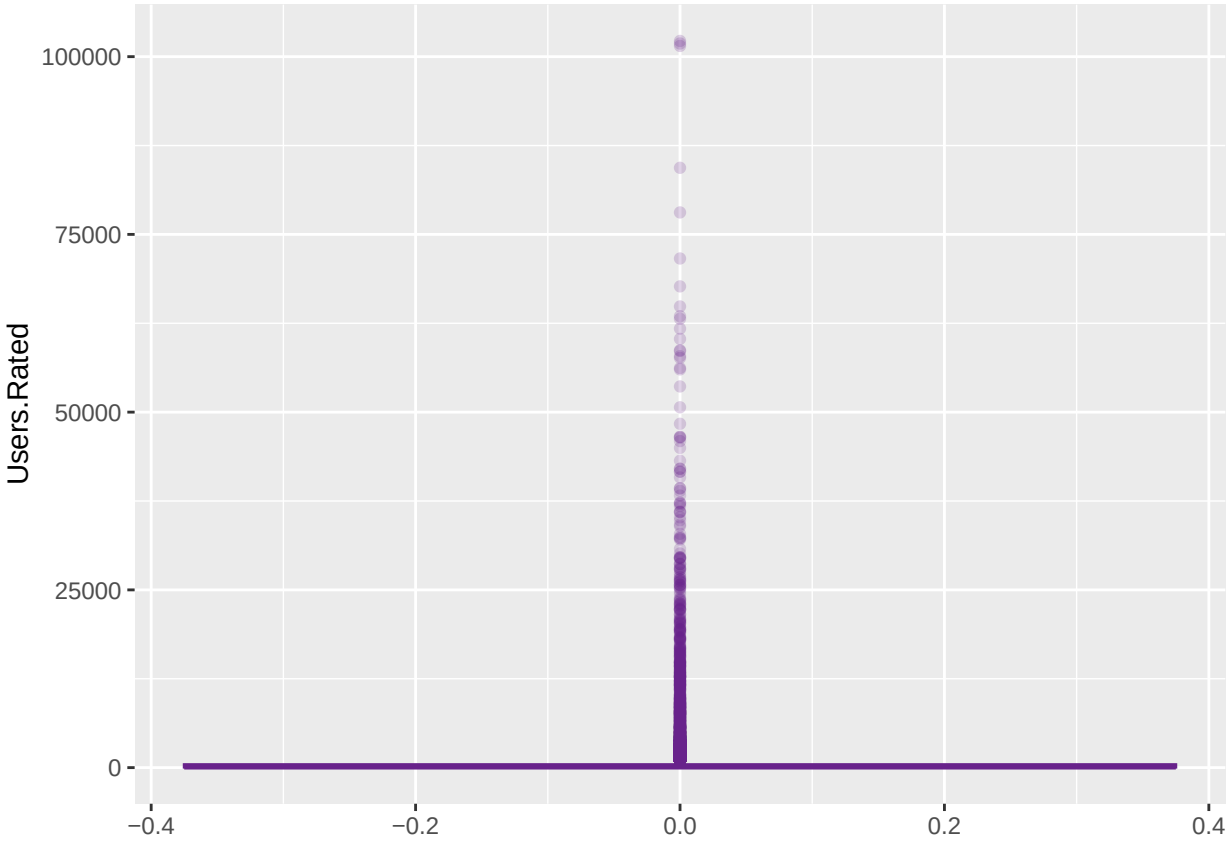

Year.Published
2020
2010
2000
1990
−0.4
−0.2
0.0
0.2
0.4

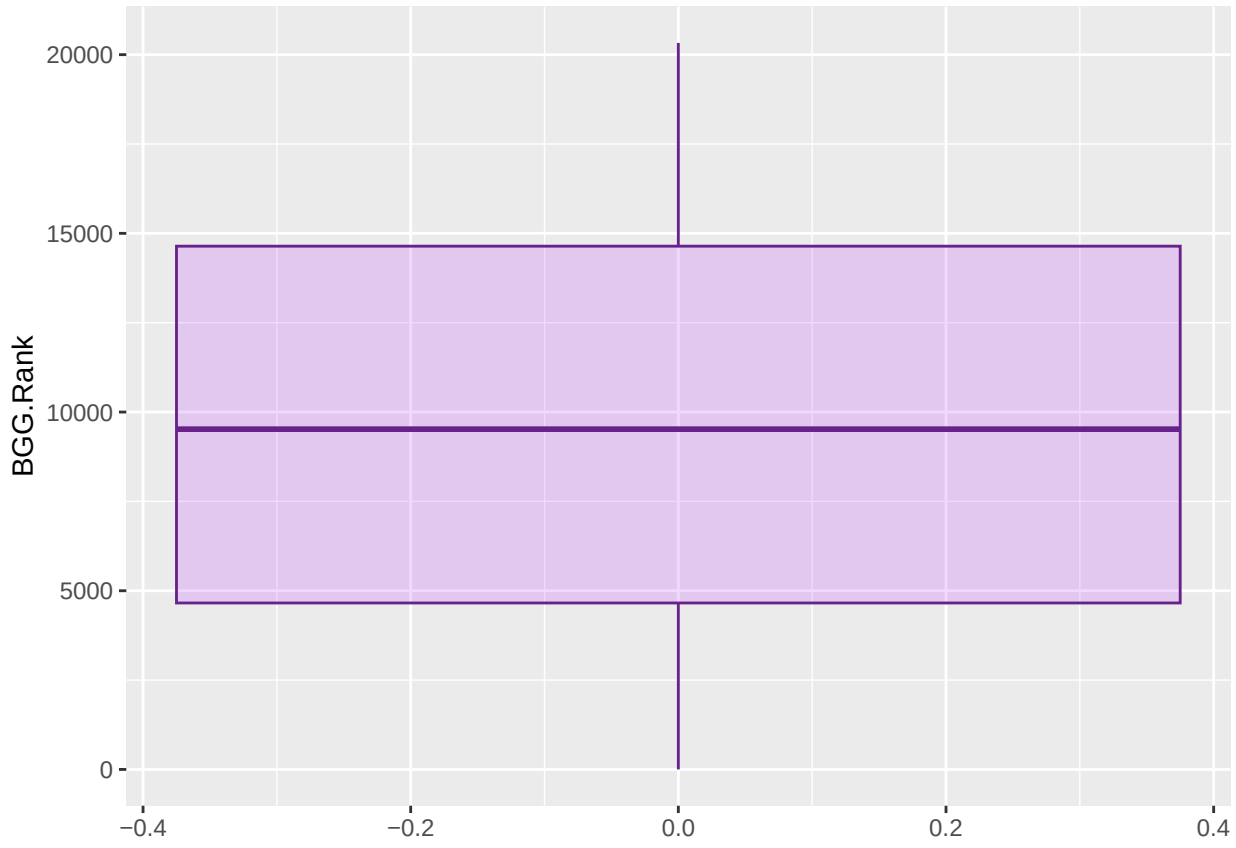

Min.Players
7.5
5.0
2.5
−0.4
−0.2
0.0
0.2
0.4

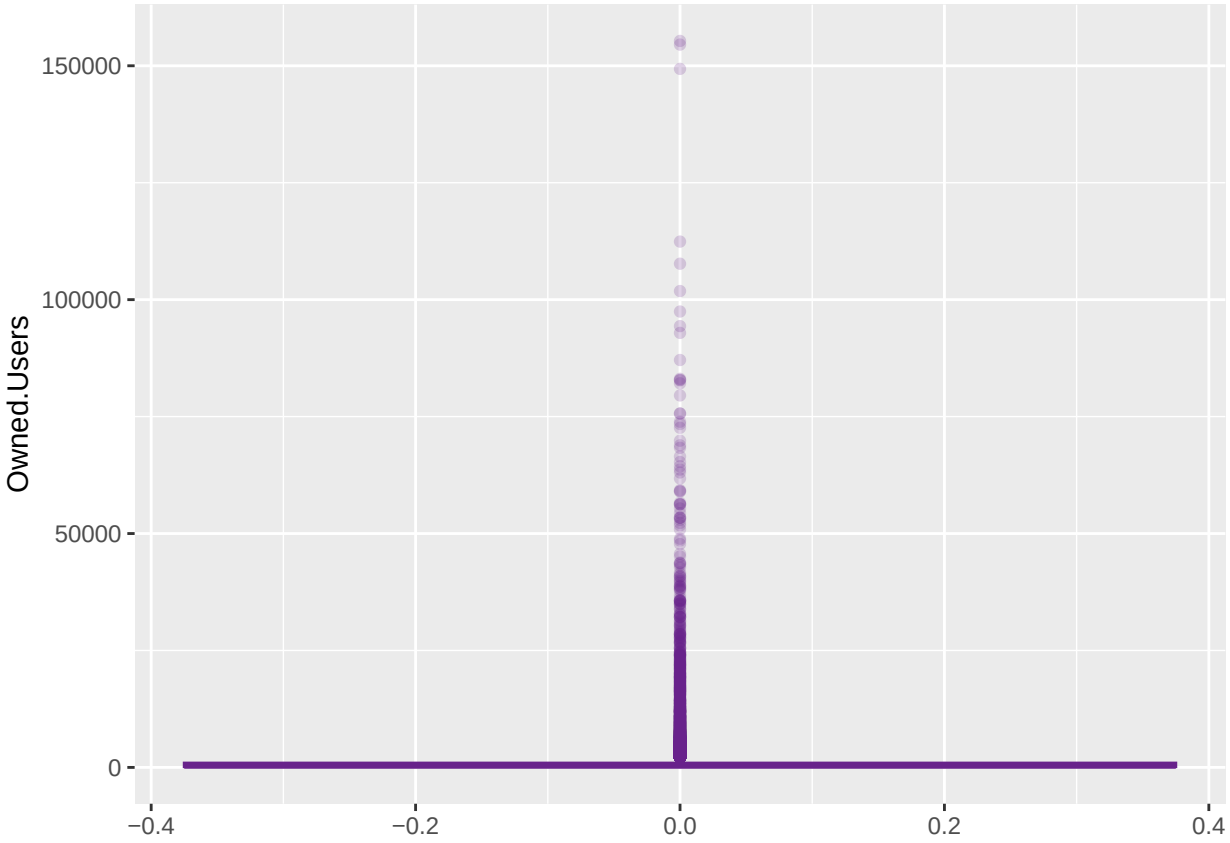

Max.Players
15
10
5
−0.4
−0.2
0.0
0.2
0.4


Play.Time
20000
15000
10000
5000
0
−0.4
−0.2
0.0
0.2
0.4


Min.Age
25
20
15
10
5
0
−0.4
−0.2
0.0
0.2
0.4


Users.Rated
100000
75000
50000
25000
0
−0.4
−0.2
0.0
0.2
0.4


BGG.Rank
20000
15000
10000
5000
0
−0.4
−0.2
0.0
0.2
0.4



Project Goals

When our group first observed the data, we began to wonder what traits make games good. Are more complex games more desirable, or is simplicity key? Are newer games preferred, or are older games better? Through analysis of this data, we aim to develop an understanding of which traits are desired in a board game.

Our group intends to create two regression models based on the attributes of our data. One model will predict the quality of the game in terms of average user rating, and the other will predict the popularity its popularity in terms of users owned. The coefficients assigned in the regression model will tell us the significance of each attribute in the overall opinion of the game. The use of two regression models generates two separate views. It tells us what traits result in games which are good, and what traits result in games which are popular.

Regressions

```
print(colnames(removed_zero_years_df))
```

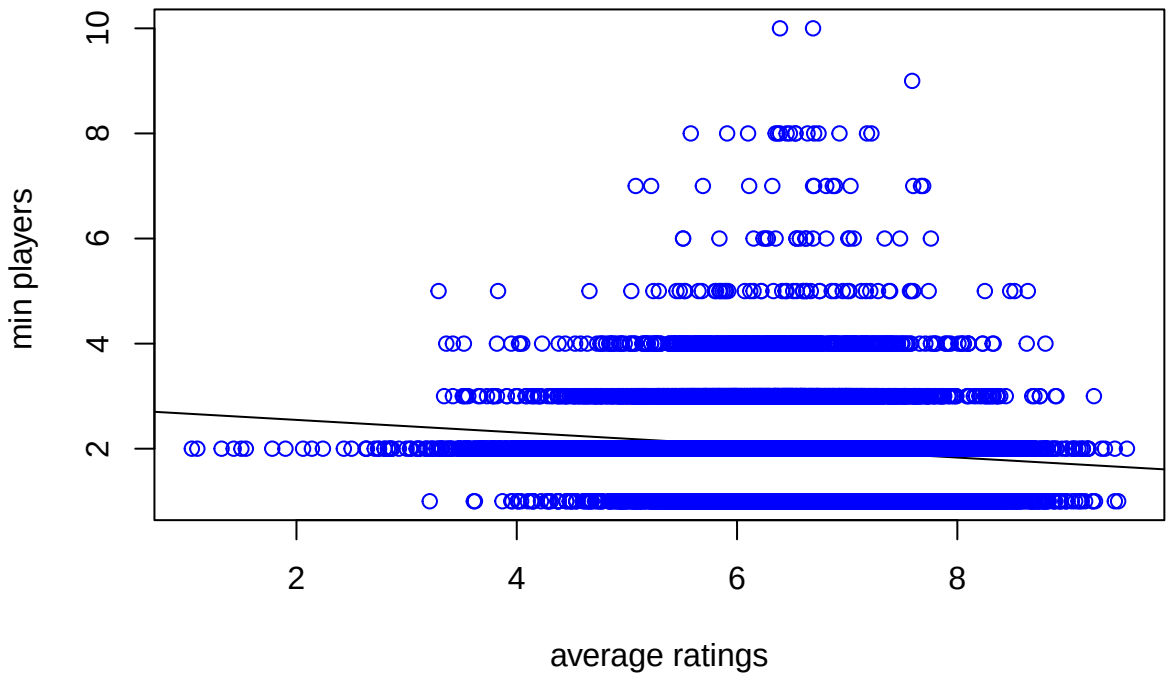
```
##  [1] "ID"                "Name"              "Year.Published"
##  [4] "Min.Players"       "Max.Players"       "Play.Time"
##  [7] "Min.Age"           "Users.Rated"       "Rating.Average"
## [10] "BGG.Rank"          "Complexity.Average" "Owned.Users"
```

```
## [13] "Mechanics"          "Domains"
```

```
x <- as.numeric(unlist( removed_both_df[c(4)]))
y <- as.numeric(unlist( removed_both_df[c(9)]))

regression <- lm(y~x)

print(summary(regression))
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.3638 -0.5638  0.0362  0.6077  3.1262
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  6.856820   0.020280   338.1   <2e-16 ***
## x           -0.221515   0.009508   -23.3   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.915 on 19965 degrees of freedom
## Multiple R-squared:  0.02646,    Adjusted R-squared:  0.02642
## F-statistic: 542.7 on 1 and 19965 DF,  p-value: < 2.2e-16
```

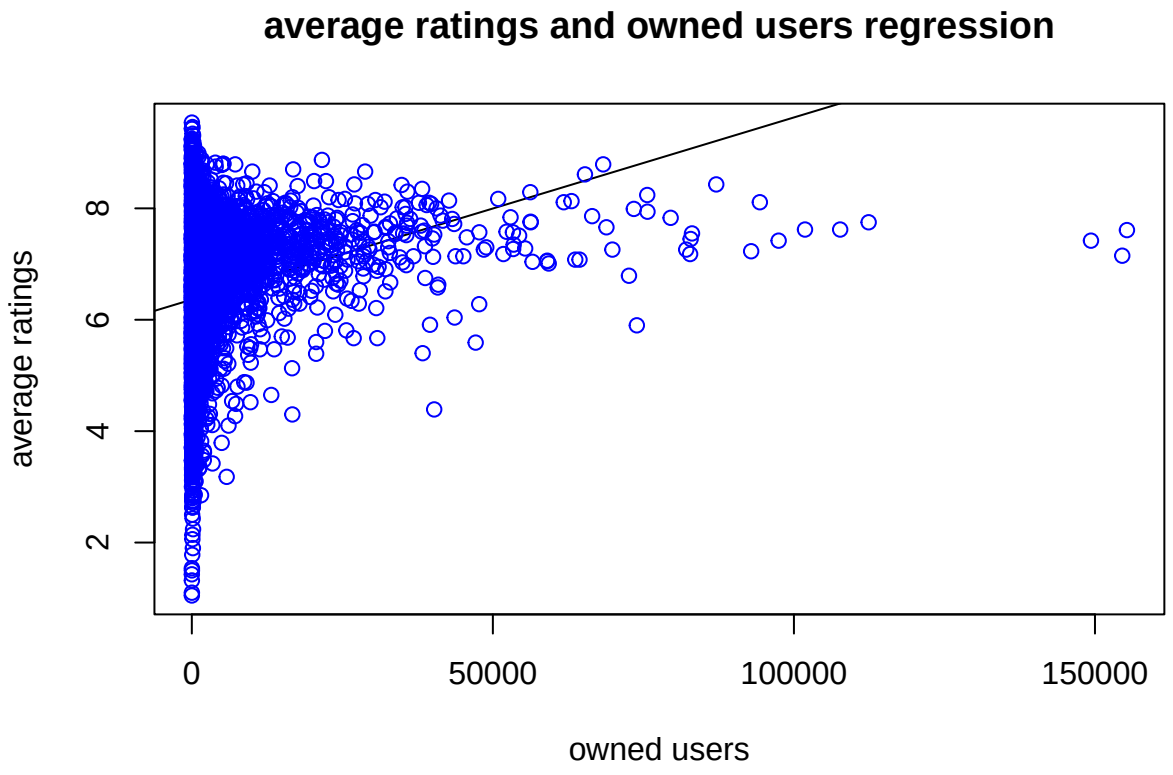
```
plot(y,x,col = "blue",main = "minimum players and average rating regression",
abline(lm(x~y)),xlab = "average ratings",ylab = "min players ")
```




```
x <- as.numeric(unlist( removed_both_df[c(9)]))
y <- as.numeric(unlist( removed_both_df[c(12)]))

regression <- lm(y~x)

print(summary(regression))
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##    Min     1Q Median     3Q    Max
##  -4485  -1410   -739      7 152707
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -4858.35     246.94  -19.67   <2e-16 ***
## x             980.68      38.13   25.72   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4997 on 19965 degrees of freedom
## Multiple R-squared:  0.03206,    Adjusted R-squared:  0.03202
## F-statistic: 661.4 on 1 and 19965 DF,  p-value: < 2.2e-16
```

```r
plot(y,x,col = "blue",main = "average ratings and owned users regression",
abline(lm(x~y)),xlab = "owned users",ylab = "average ratings")
```

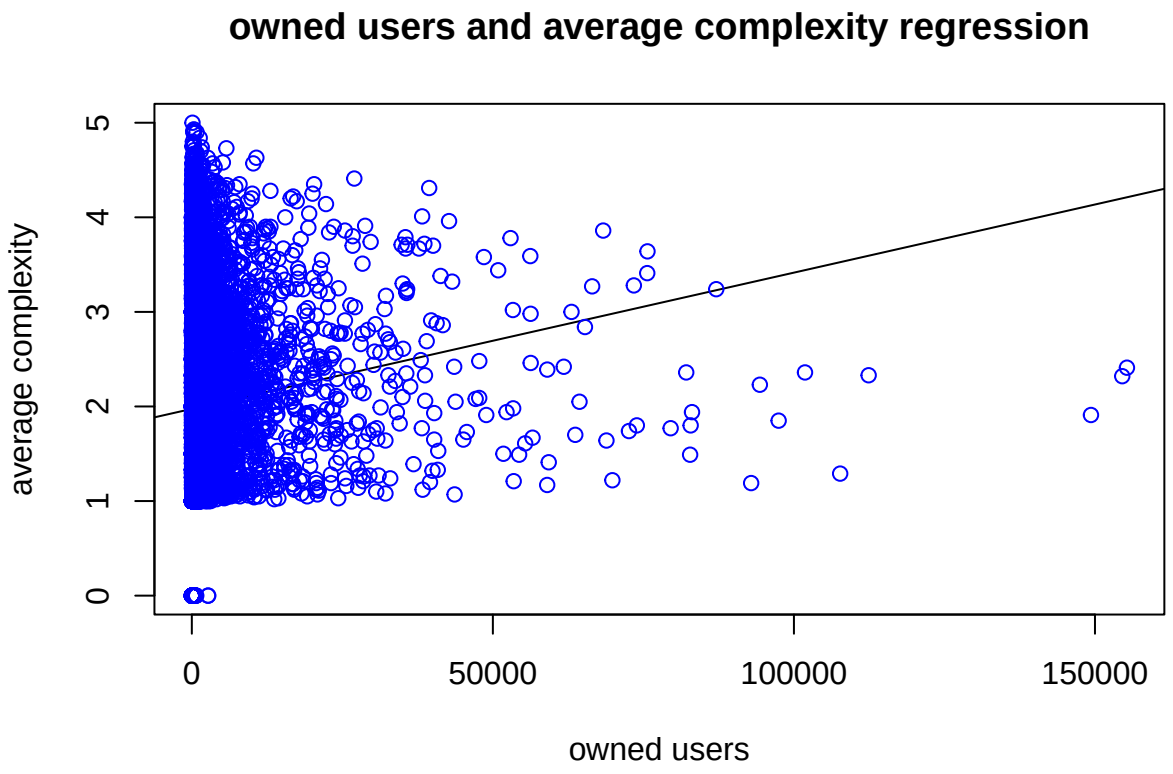



```r
x <- as.numeric(unlist( removed_both_df[c(11)]))
y <- as.numeric(unlist( removed_both_df[c(12)]))

regression <- lm(y~x)

print(summary(regression))
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##    Min     1Q Median     3Q    Max
##  -2874  -1290   -872   -392 153670
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   394.78      91.58   4.311 1.64e-05 ***
## x             517.37      42.25  12.245  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 5060 on 19965 degrees of freedom
## Multiple R-squared:  0.007454,   Adjusted R-squared:  0.007404
## F-statistic: 149.9 on 1 and 19965 DF,  p-value: < 2.2e-16
```

```
plot(y,x,col = "blue",main = "owned users and average complexity regression",
abline(lm(x~y)),xlab = "owned users",ylab = "average complexity")
```

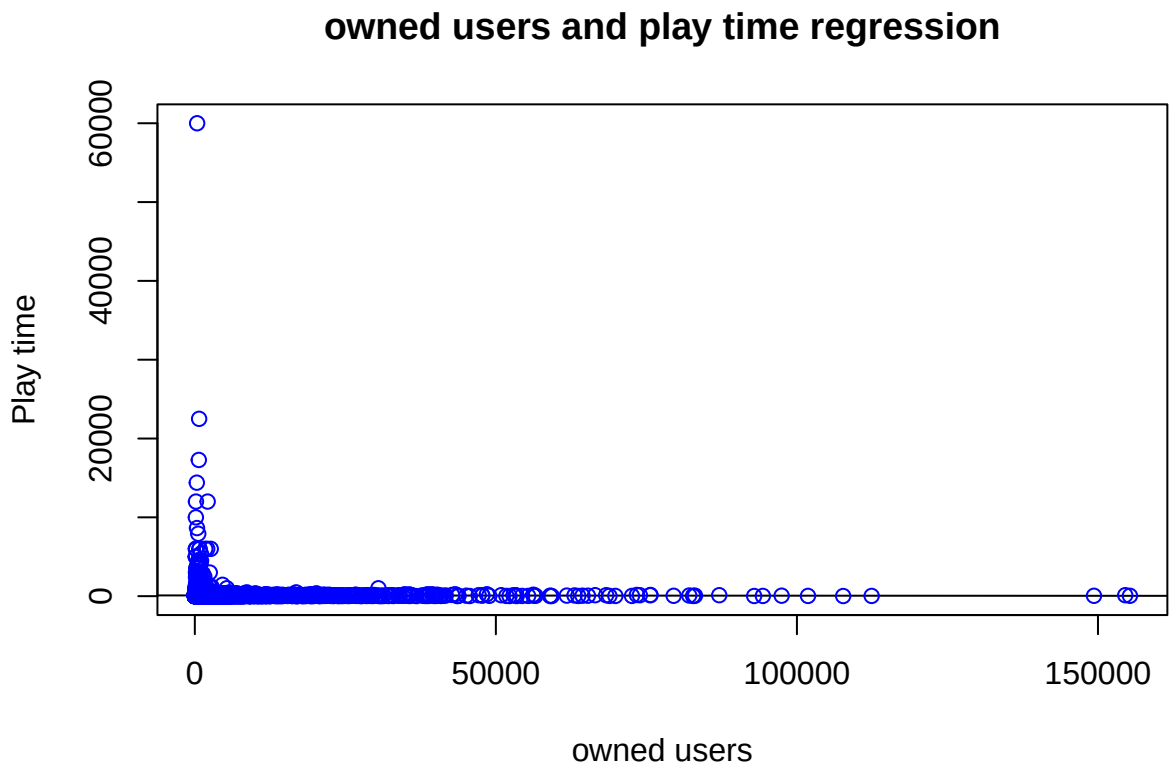



```
x <- as.numeric(unlist( removed_both_df[c(6)]))
y <- as.numeric(unlist( removed_both_df[c(12)]))

regression <- lm(y~x)

print(summary(regression))
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##    Min     1Q Median     3Q    Max
##  -1427  -1280  -1111   -549 153884
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1429.54768   36.44100  39.229   <2e-16 ***
```

```
## x             -0.02861    0.06536  -0.438    0.662
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5079 on 19965 degrees of freedom
## Multiple R-squared:  9.6e-06,    Adjusted R-squared:  -4.049e-05
## F-statistic: 0.1917 on 1 and 19965 DF,  p-value: 0.6615
```

```
plot(y,x,col = "blue",main = "owned users and play time regression",
abline(lm(x~y)),xlab = "owned users",ylab = "Play time")
```

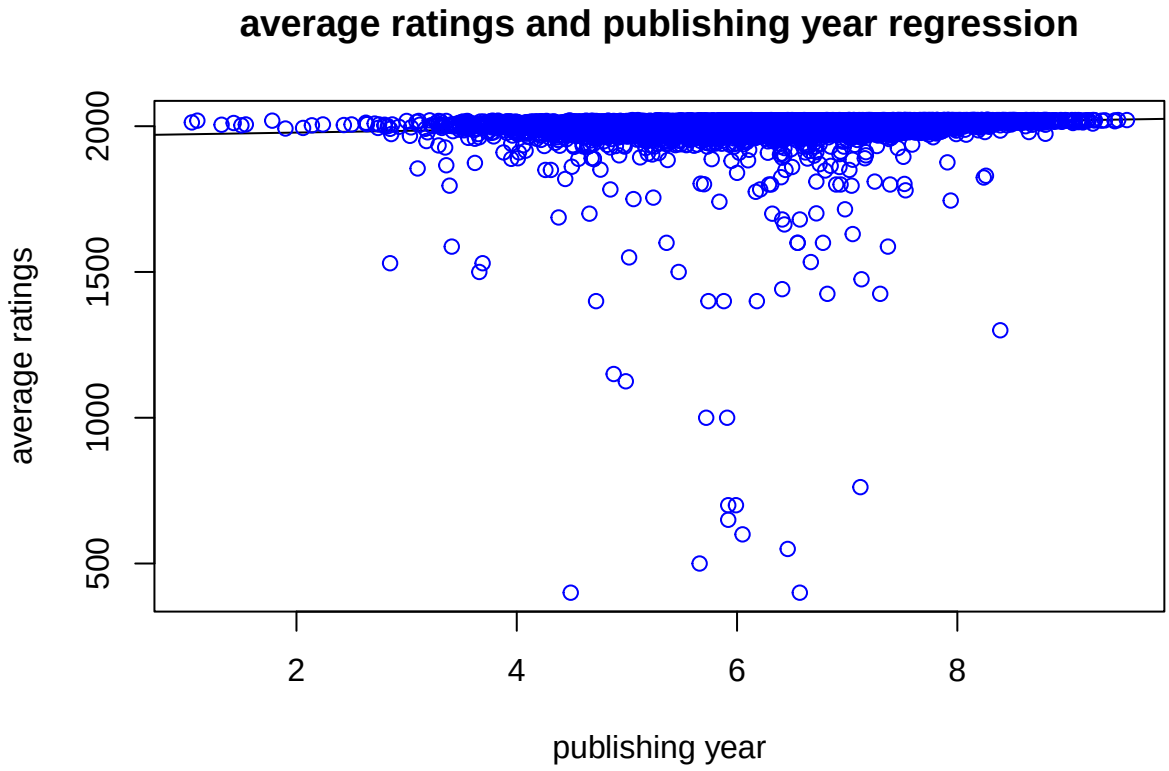



```
x <- as.numeric(unlist( removed_both_df[c(3)]))
y <- as.numeric(unlist( removed_both_df[c(9)]))

regression <- lm(y~x)

print(summary(regression))
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.3851 -0.5769  0.0249  0.6088  5.0204
```

```
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.3380578  0.3168831   1.067    0.286
## x           0.0030288  0.0001581  19.163   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.919 on 19965 degrees of freedom
## Multiple R-squared:  0.01806,    Adjusted R-squared:  0.01801
## F-statistic: 367.2 on 1 and 19965 DF,  p-value: < 2.2e-16
```

```
plot(y,x,col = "blue",main = "average ratings and publishing year regression",
abline(lm(x~y)),xlab = "publishing year",ylab = "average ratings")
```

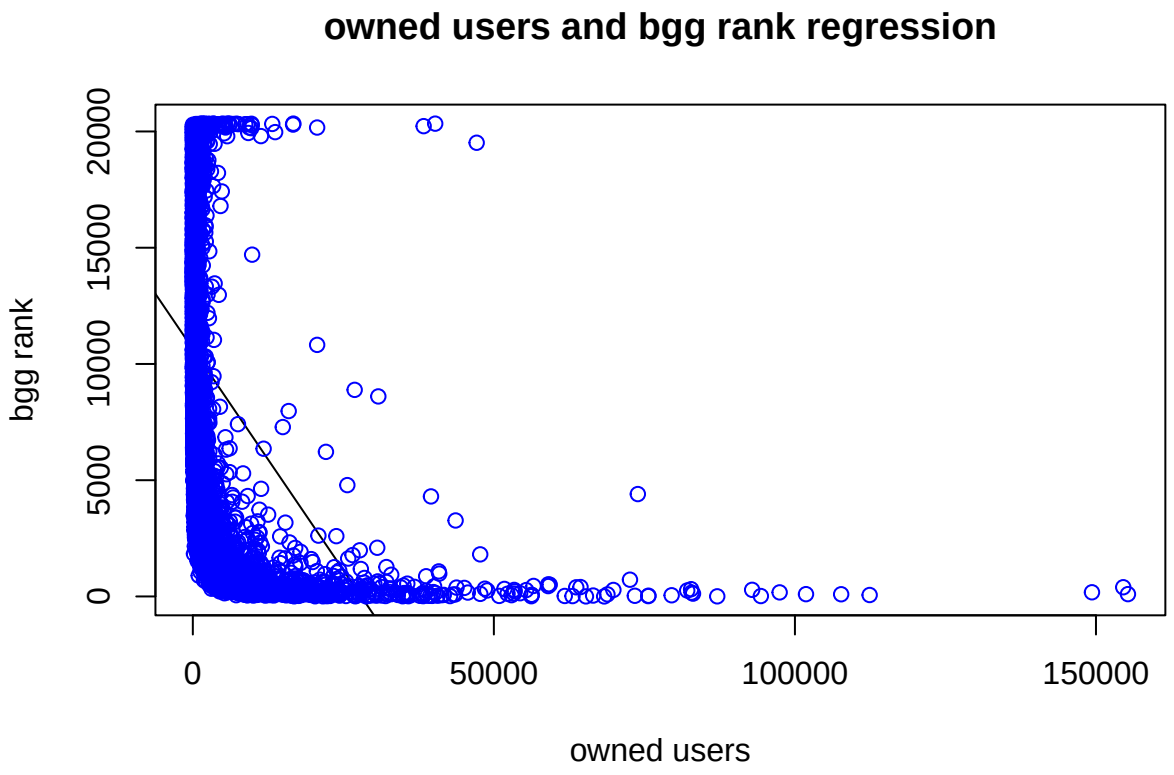



```
x <- as.numeric(unlist( removed_both_df[c(10)]))
y <- as.numeric(unlist( removed_both_df[c(12)]))

regression <- lm(y~x)

print(summary(regression))
```

```
##
## Call:
## lm(formula = y ~ x)
```

```
##
## Residuals:
##    Min     1Q Median     3Q    Max
##  -3596  -1677   -690    555 151033
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.308e+03  6.756e+01   63.75   <2e-16 ***
## x           -2.850e-01  5.781e-03  -49.31   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4795 on 19965 degrees of freedom
## Multiple R-squared:  0.1085, Adjusted R-squared:  0.1085
## F-statistic:  2431 on 1 and 19965 DF,  p-value: < 2.2e-16
```

```
plot(y,x,col = "blue",main = "owned users and bgg rank regression",
abline(lm(x~y)),xlab = "owned users",ylab = "bgg rank")
```




```
dv_df <- read.csv("./dataset_for_viewing.csv", sep = ",")
dv_df_removed <- subset(dv_df,(dv_df$Min.Players != 0 & dv_df$Max.Players != 0 &
                                 dv_df$Year.Published > 0 & dv_df$Owned.Users > 0))
dv_df_all <- dv_df_removed[c(3,4,5,6,7,8,9,10,11,12)]
dv_df_numeric <- dv_df_removed[c(3,4,5,6,7,9,11,12)]

cor(dv_df_all, y = dv_df_all, method="pearson")
```

```
##                     Year.Published  Min.Players   Max.Players    Play.Time
## Year.Published        1.000000000 -0.019941851  0.0027281252 -0.007364623
## Min.Players          -0.019941851  1.000000000  0.0816658970  0.018412980
## Max.Players           0.002728125  0.081665897  1.0000000000 -0.007594862
## Play.Time            -0.007364623  0.018412980 -0.0075948617  1.000000000
## Min.Age               0.058878796  0.037779189  0.0070763684  0.039454911
## Users.Rated          -0.003127421 -0.009943433 -0.0014018981 -0.004818999
## Rating.Average        0.134388597 -0.162680603 -0.0416335492  0.089449034
## BGG.Rank             -0.094189090  0.098055790  0.0262532043 -0.044890240
## Complexity.Average   -0.013302679 -0.176956684 -0.0910101490  0.180891440
## Owned.Users           0.001219603 -0.019645460 -0.0004440037 -0.003098339
##                         Min.Age  Users.Rated Rating.Average    BGG.Rank
## Year.Published      0.058878796 -0.003127421     0.13438860 -0.09418909
## Min.Players         0.037779189 -0.009943433    -0.16268060  0.09805579
## Max.Players         0.007076368 -0.001401898    -0.04163355  0.02625320
## Play.Time           0.039454911 -0.004818999     0.08944903 -0.04489024
## Min.Age             1.000000000  0.071363567     0.19597631 -0.21582838
## Users.Rated         0.071363567  1.000000000     0.17188488 -0.31086333
## Rating.Average      0.195976306  0.171884883     1.00000000 -0.74024336
## BGG.Rank           -0.215828377 -0.310863335    -0.74024336  1.00000000
## Complexity.Average  0.291825090  0.084983703     0.47994306 -0.38029424
## Owned.Users         0.080107178  0.986037771     0.17906641 -0.32946809
##                     Complexity.Average   Owned.Users
## Year.Published             -0.01330268  0.0012196032
## Min.Players                -0.17695668 -0.0196454600
## Max.Players                -0.09101015 -0.0004440037
## Play.Time                   0.18089144 -0.0030983393
## Min.Age                     0.29182509  0.0801071783
## Users.Rated                 0.08498370  0.9860377705
## Rating.Average              0.47994306  0.1790664129
## BGG.Rank                   -0.38029424 -0.3294680857
## Complexity.Average          1.00000000  0.0863343718
## Owned.Users                 0.08633437  1.0000000000
```

```
head(dv_df_numeric)
```

```
##   Year.Published Min.Players Max.Players Play.Time Min.Age Rating.Average
## 1           2017           1           4       120      14           8.79
## 2           2015           2           4        60      13           8.61
## 3           2018           2           4       120      14           8.66
## 4           2016           1           5       120      12           8.43
## 5           2017           3           6       480      14           8.70
## 6           2020           1           4       120      14           8.87
##   Complexity.Average Owned.Users
## 1               3.86       68323
## 2               2.84       65294
## 3               3.91       28785
## 4               3.24       87099
## 5               4.22       16831
## 6               3.55       21609
```

```r
cor(dv_df_numeric, y = dv_df_numeric, method="pearson")
```

```
##                     Year.Published Min.Players   Max.Players    Play.Time
## Year.Published        1.000000000 -0.01994185  0.0027281252 -0.007364623
## Min.Players          -0.019941851  1.00000000  0.0816658970  0.018412980
## Max.Players           0.002728125  0.08166590  1.0000000000 -0.007594862
## Play.Time            -0.007364623  0.01841298 -0.0075948617  1.000000000
## Min.Age               0.058878796  0.03777919  0.0070763684  0.039454911
## Rating.Average        0.134388597 -0.16268060 -0.0416335492  0.089449034
## Complexity.Average   -0.013302679 -0.17695668 -0.0910101490  0.180891440
## Owned.Users           0.001219603 -0.01964546 -0.0004440037 -0.003098339
##                         Min.Age Rating.Average Complexity.Average   Owned.Users
## Year.Published     0.058878796     0.13438860        -0.01330268  0.0012196032
## Min.Players        0.037779189    -0.16268060        -0.17695668 -0.0196454600
## Max.Players        0.007076368    -0.04163355        -0.09101015 -0.0004440037
## Play.Time          0.039454911     0.08944903         0.18089144 -0.0030983393
## Min.Age            1.000000000     0.19597631         0.29182509  0.0801071783
## Rating.Average     0.195976306     1.00000000         0.47994306  0.1790664129
## Complexity.Average 0.291825090     0.47994306         1.00000000  0.0863343718
## Owned.Users        0.080107178     0.17906641         0.08633437  1.0000000000
```

```r
Rating_Model <- lm(Rating.Average ~ Year.Published + Min.Players + Max.Players + Play.Time + Min.Age + C
summary(Rating_Model)
```

```
##
## Call:
## lm(formula = Rating.Average ~ Year.Published + Min.Players +
##     Max.Players + Play.Time + Min.Age + Complexity.Average, data = dv_df_numeric)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.3433 -0.4638  0.0333  0.5055  4.8183
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)        -6.001e-01  2.771e-01  -2.165   0.0304 *
## Year.Published      3.047e-03  1.377e-04  22.135   <2e-16 ***
## Min.Players        -1.137e-01  8.499e-03 -13.375   <2e-16 ***
## Max.Players         3.186e-04  3.710e-04   0.859   0.3906
## Play.Time           1.441e-05  1.046e-05   1.377   0.1685
## Min.Age             1.544e-02  1.653e-03   9.341   <2e-16 ***
## Complexity.Average  4.907e-01  7.261e-03  67.571   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7983 on 19960 degrees of freedom
## Multiple R-squared:  0.2593, Adjusted R-squared:  0.259
## F-statistic:  1164 on 6 and 19960 DF,  p-value: < 2.2e-16
```

```r
Pop_Model <- lm(Owned.Users ~ Year.Published + Min.Players + Max.Players + Play.Time + Min.Age + Comple
summary(Pop_Model)
```

```
##
## Call:
## lm(formula = Owned.Users ~ Year.Published + Min.Players + Max.Players +
##     Play.Time + Min.Age + Complexity.Average, data = dv_df_numeric)
##
## Residuals:
##    Min     1Q Median     3Q    Max
##  -2950  -1330   -848   -337 153845
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)         337.84047 1753.51516   0.193   0.8472
## Year.Published       -0.21585    0.87109  -0.248   0.8043
## Min.Players         -71.76022   53.77939  -1.334   0.1821
## Max.Players           2.06089    2.34784   0.878   0.3801
## Play.Time            -0.16694    0.06619  -2.522   0.0117 *
## Min.Age              85.48714   10.45975   8.173 3.19e-16 ***
## Complexity.Average  424.05403   45.94698   9.229  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5051 on 19960 degrees of freedom
## Multiple R-squared:  0.0112, Adjusted R-squared:  0.01091
## F-statistic: 37.69 on 6 and 19960 DF,  p-value: < 2.2e-16
```

Citations

Dilini Samarasinghe, July 5, 2021, "BoardGameGeek Dataset on Board Games", IEEE Dataport, doi: https://dx.doi.org/10.21227/9g61-bs59.