Final Project: Data Science Culmination Project

Jordon Zeigler

Project Proposal: Week of November 15-18th

Project Due: Our Final Exam Time.

Getting started

Here are the steps for getting started:

- Start with the assignment link that creates a repo on GitHub with starter documents. I have sent this to you through email.
- Clone this repo in RStudio
- Make any changes needed as outlined by the tasks you need to complete for the assignment
- Periodically commit changes (the more often the better, for example, once per each new task)
  - Remember, git will yell at you when you try to commit before running the following lines in the terminal
    - `git config --global user.name "Your Name Here"`
    - `git config --global user.email "Your Email Here"`
- Push all your changes back to your GitHub repo

and voila, you're done! Once you push your changes back you do not need to do anything else to "submit" your work. And you can of course push multiple times throughout the assignment. At the time of the deadline I will take whatever is in your repo and consider it your final submission, and grade the state of your work at that time (which means even if you made mistakes before then, you wouldn't be penalized for them as long as the final state of your work is correct).

Assignment Description

This is it! This is the culmination of all your work in both of the data science courses! The parameters of this project are very general because I want to give you the chance to explore your interests and be creative. Generally, your project will go through the entire data science process. The project will involve a formal written document as well as a presentation. The written document will be worth 2/3rd's of the final grade and the presentation will be worth the other 1/3rd.

Here are the sections on how the project will be evaluated (A rubric will be released later with more specific parameters).

- Questions and Goals: The questions you wish to answer and the goals of the project.
- Data Acquisition: The project describes how the data was obtained.
- Data Preprocessing: Throughout the project, the proper preprocessing techniques (variable transformations, reshaping data, etc.) are utilized
- Exploratory Data Analysis: Proper exploratory plots and summarizes are utilized to describe the data and showcase certain interesting aspects of the data that you will explore later in the project.
- Modeling and Analysis: This is a large portion of the project! You work toward answer the questions/goals you stated at the beginning. Your project needs to include at least 3 modeling techniques we discussed in class. You will fit, tune, and compare the methods. You will discuss the results and why they make sense in context. This section can include a wide range of modeling techniques. Your proposal should be focused on describing what you want to do in this section.

- Data Product: You present your data, models, and conclusions in a professional manner. This could include an interactive data product.

Place Work Below!!

```
library("tidyverse");theme_set(theme_bw())
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.1.0     v forcats 0.5.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library("tidymodels")
```

```
## Registered S3 method overwritten by 'tune':
##   method                   from
##   required_pkgs.model_spec parsnip
```

```
## -- Attaching packages --------------------------------------- tidymodels 0.1.4 --
```

```
## v broom        0.7.10     v rsample      0.1.1
## v dials        0.0.10     v tune         0.1.6
## v infer        1.0.0      v workflows    0.2.4
## v modeldata    0.1.1      v workflowsets 0.1.0
## v parsnip      0.1.7      v yardstick    0.0.8
## v recipes      0.1.17
```

```
## -- Conflicts ------------------------------------------ tidymodels_conflicts() --
## x scales::discard() masks purrr::discard()
## x dplyr::filter()   masks stats::filter()
## x recipes::fixed()  masks stringr::fixed()
## x dplyr::lag()      masks stats::lag()
## x yardstick::spec() masks readr::spec()
## x recipes::step()   masks stats::step()
## * Use tidymodels_prefer() to resolve common conflicts.
```

```
library("janitor")
```

```
##
## Attaching package: 'janitor'
```

```
## The following objects are masked from 'package:stats':
##
##     chisq.test, fisher.test
```

```
library("knitr")
library("caret")
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following objects are masked from 'package:yardstick':
```

```
##
##     precision, recall, sensitivity, specificity

## The following object is masked from 'package:purrr':
##
##     lift
```

```
library("leaps")
library("olsrr")
```

```
##
## Attaching package: 'olsrr'

## The following object is masked from 'package:datasets':
##
##     rivers
```

```
library("glmnet")
```

```
## Loading required package: Matrix

##
## Attaching package: 'Matrix'

## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack

## Loaded glmnet 4.1-3
```

```
library("Metrics")
```

```
##
## Attaching package: 'Metrics'

## The following objects are masked from 'package:caret':
##
##     precision, recall

## The following objects are masked from 'package:yardstick':
##
##     accuracy, mae, mape, mase, precision, recall, rmse, smape
```

```
library("tree")
```

```
## Registered S3 method overwritten by 'tree':
##   method     from
##   print.tree cli
```

dataset acquired from kaggle and is focused on showcasing all of the board games featured on the board game geeks website

```
Board_Games <- read_csv("bgg_dataset_complete.csv") %>% clean_names()
```

```
## Warning: One or more parsing issues, see `problems()` for details

## Rows: 20343 Columns: 14

## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (6): ID, Name, Max Players, Play Time, Mechanics, Domains
## dbl (8): Year Published, Min Players, Min Age, Users Rated, Rating Average, ...
```

```
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
Board_Games$play_time <- as.numeric(Board_Games$play_time)
```

```
## Warning: NAs introduced by coercion
```

```
Board_Games$max_players <- as.numeric(Board_Games$max_players)
```

```
## Warning: NAs introduced by coercion
```

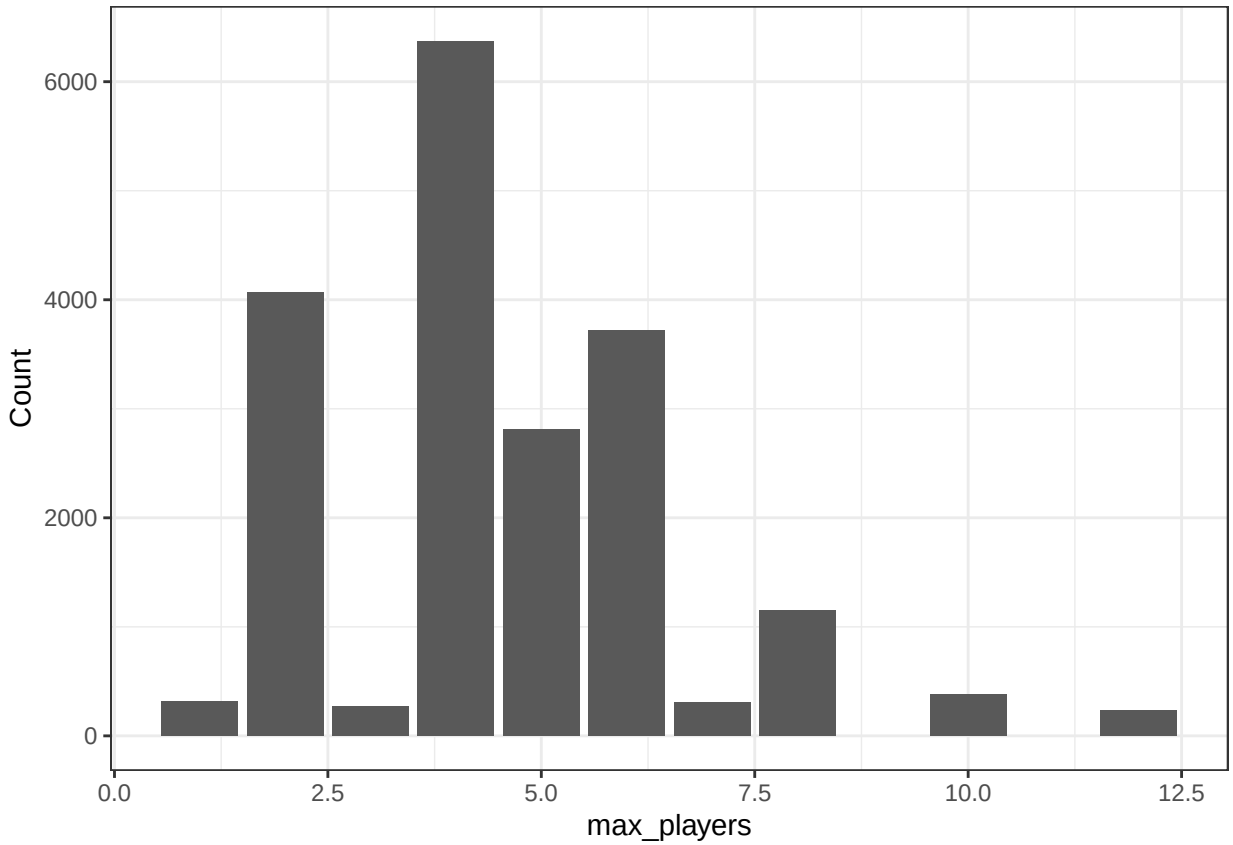
the main purpose of this data analysis is to see which variables are the most relevant in determining the popularity of a board game, represented by the BGG rank, which is an overall ranking for each board game featured on the BoardGameGeek website, the main variables that will be looked at are, Min players, Max players, Year Published, Play Time, Min Age, Rating average, Users rated and Complexity Average

```
summary(Board_Games)
```

```
##       id                name            year_published   min_players
##  Length:20343       Length:20343       Min.   :-3500   Min.   :  0.000
##  Class :character   Class :character   1st Qu.: 2001   1st Qu.:  2.000
##  Mode  :character   Mode  :character   Median : 2011   Median :  2.000
##                                        Mean   : 1984   Mean   :  2.136
##                                        3rd Qu.: 2016   3rd Qu.:  2.000
##                                        Max.   : 2022   Max.   :435.000
##                                        NA's   :8
##   max_players        play_time           min_age        users_rated
##  Min.   :  0.000   Min.   :    0.00   Min.   : 0.000   Min.   :    30.0
##  1st Qu.:  4.000   1st Qu.:   30.00   1st Qu.: 8.000   1st Qu.:    55.0
##  Median :  4.000   Median :   45.00   Median :10.000   Median :   120.0
##  Mean   :  5.679   Mean   :   91.29   Mean   : 9.605   Mean   :   841.5
##  3rd Qu.:  6.000   3rd Qu.:   90.00   3rd Qu.:12.000   3rd Qu.:   385.0
##  Max.   :999.000   Max.   :60000.00   Max.   :58.000   Max.   :102214.0
##  NA's   :12        NA's   :12         NA's   :12       NA's   :13
##  rating_average         bgg_rank     complexity_average  owned_users
##  Min.   :    1.050   Min.   :    1   Min.   :  0.000    Min.   :     0
##  1st Qu.:    5.820   1st Qu.: 5084   1st Qu.:  1.330    1st Qu.:   146
##  Median :    6.430   Median :10168   Median :  1.970    Median :   309
##  Mean   :    7.003   Mean   :10170   Mean   :  2.001    Mean   :  1409
##  3rd Qu.:    7.030   3rd Qu.:15257   3rd Qu.:  2.545    3rd Qu.:   865
##  Max.   :12197.000   Max.   :20344   Max.   :210.000    Max.   :155312
##  NA's   :12          NA's   :13      NA's   :12         NA's   :36
##   mechanics           domains
##  Length:20343       Length:20343
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
##
```

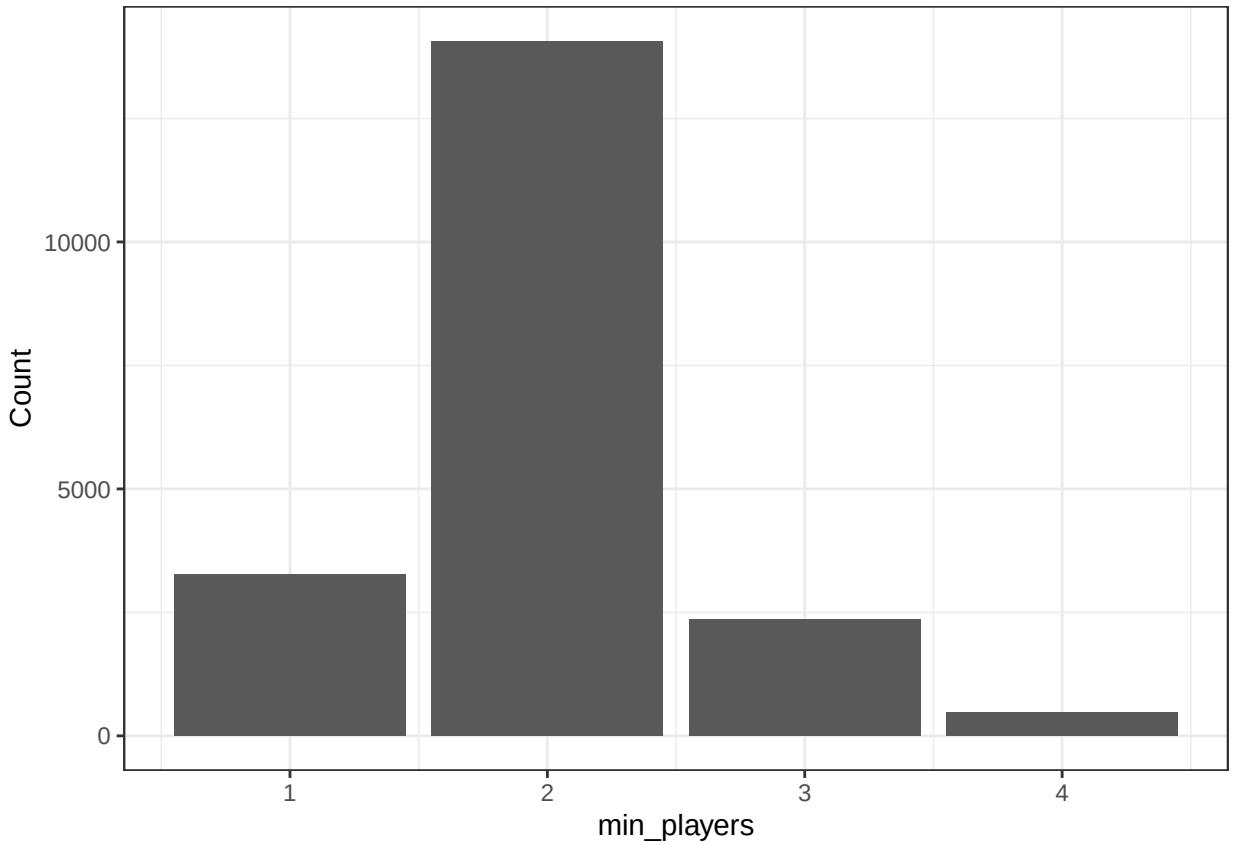
```
# produces a histogram indicating how often board games are produced with high numbers of players that
Max_Player_plot <- Board_Games %>% group_by(max_players) %>%  summarize(Count = n()) %>% arrange(desc(C

ggplot(Max_Player_plot, aes(x = max_players, y = Count), ylim = c(0, 2000)) +
    geom_col()
```



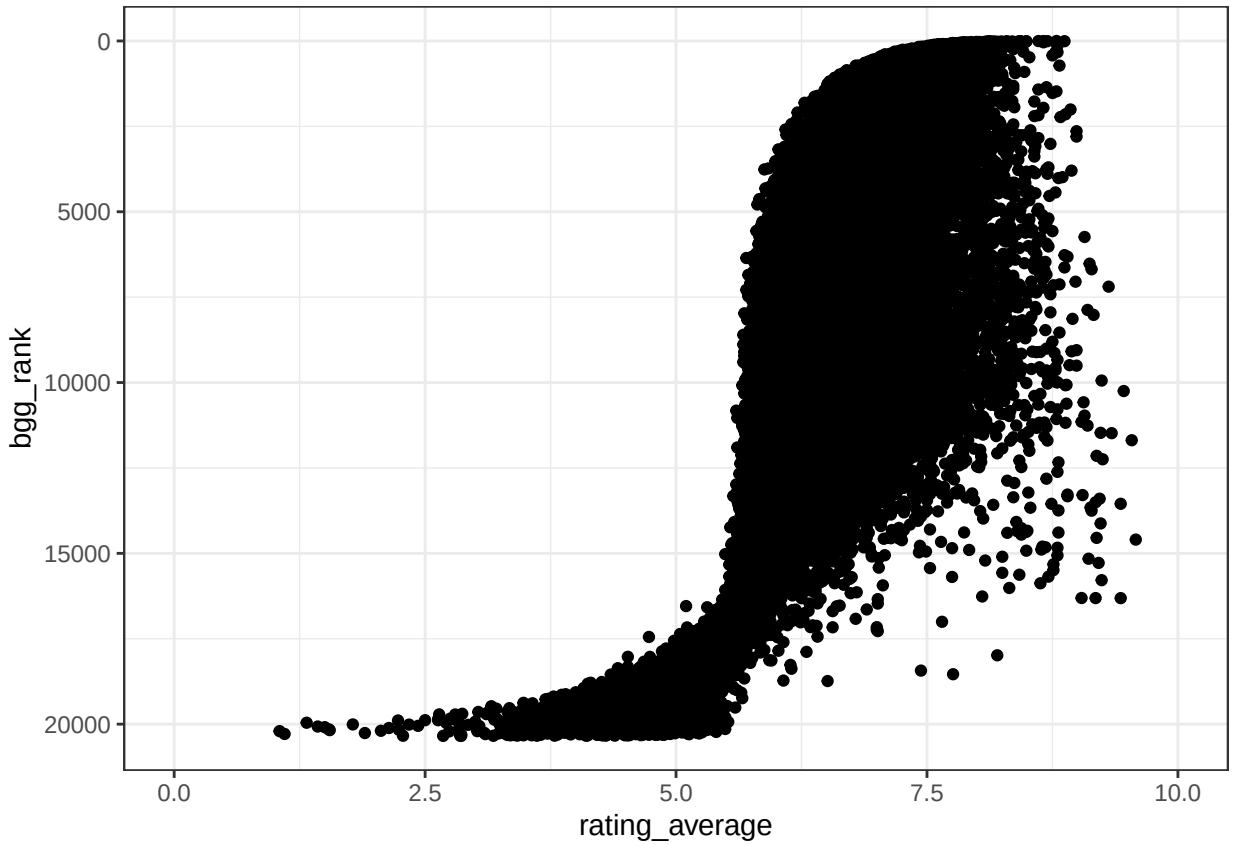

```
# will produce a histogram involving the minimum number of players that a board game can support
Min_Player_plot <- Board_Games %>% group_by(min_players) %>% summarize(Count = n()) %>% arrange(desc(Co

ggplot(Min_Player_plot, aes(x = min_players, y = Count), ylim = c(0, 2000)) + geom_col()
```



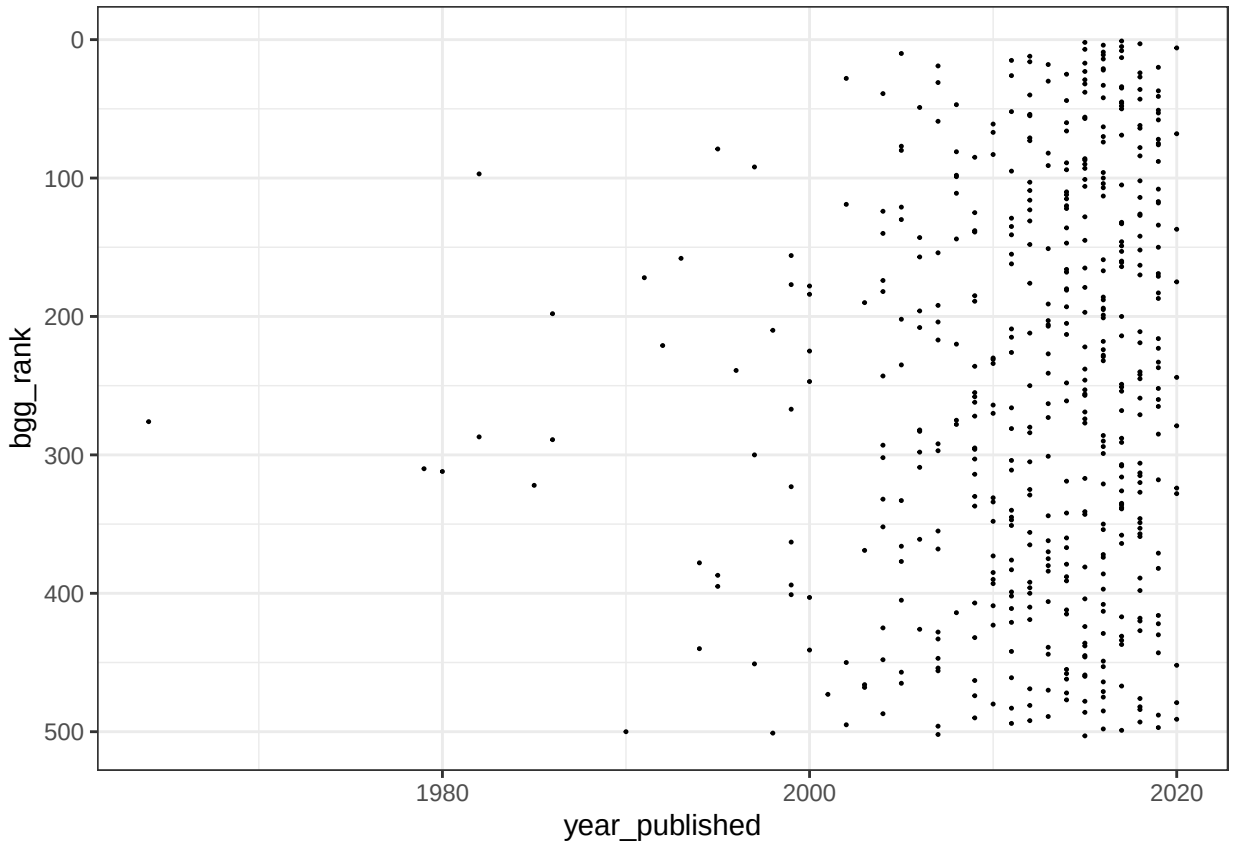

```
# will produce a point graph involving the correlation between the rating average and the BGG rank, whi
ggplot(Board_Games, aes(x = rating_average, y = bgg_rank)) +
    geom_point() + xlim(0,10) + scale_y_reverse()
```

```
## Warning: Removed 13 rows containing missing values (geom_point).
```



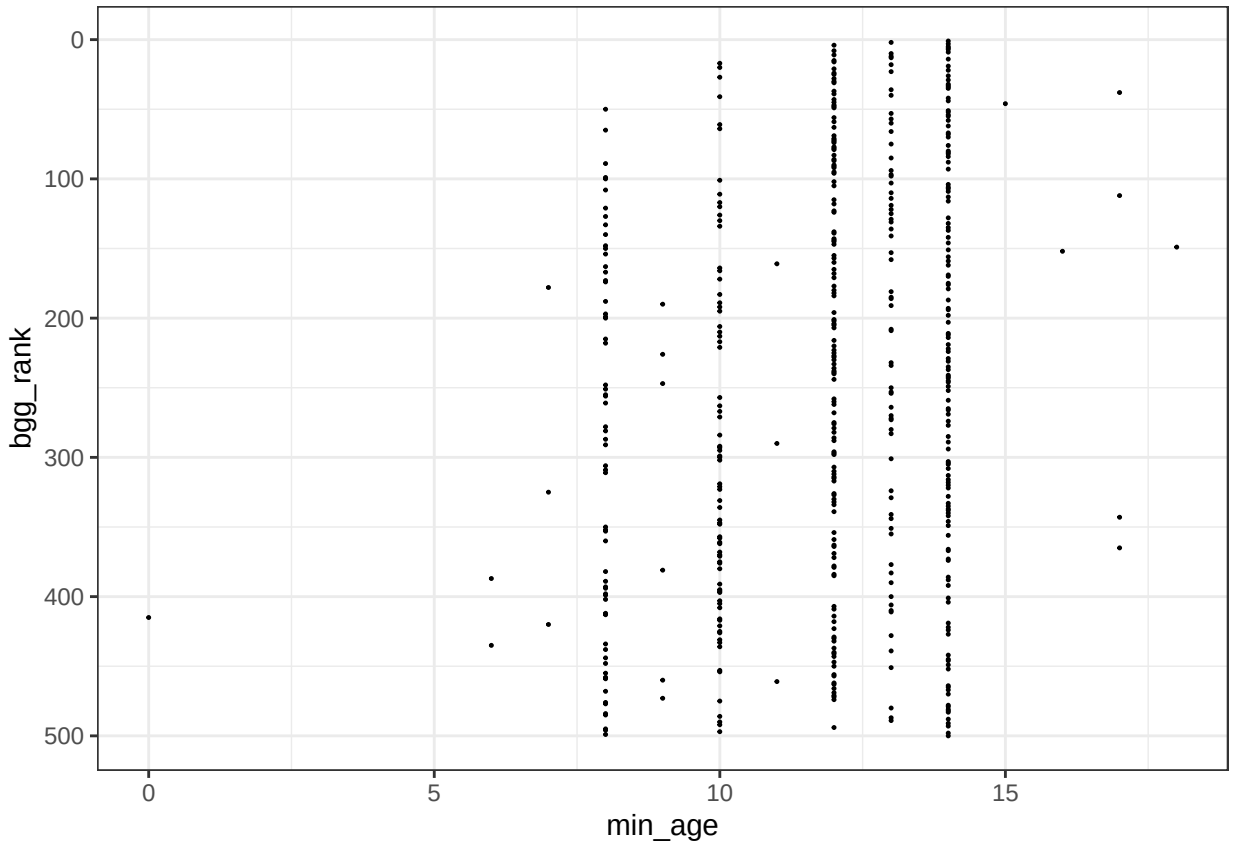

```
# will produce a point graph involving the correlation between the year a board game is published and t
Year_var_graph <- Board_Games %>% filter(year_published > 1950) %>% head(500)

ggplot(Year_var_graph, aes(x = year_published, y = bgg_rank)) +
    geom_point(size = 0.2) + scale_y_reverse()
```



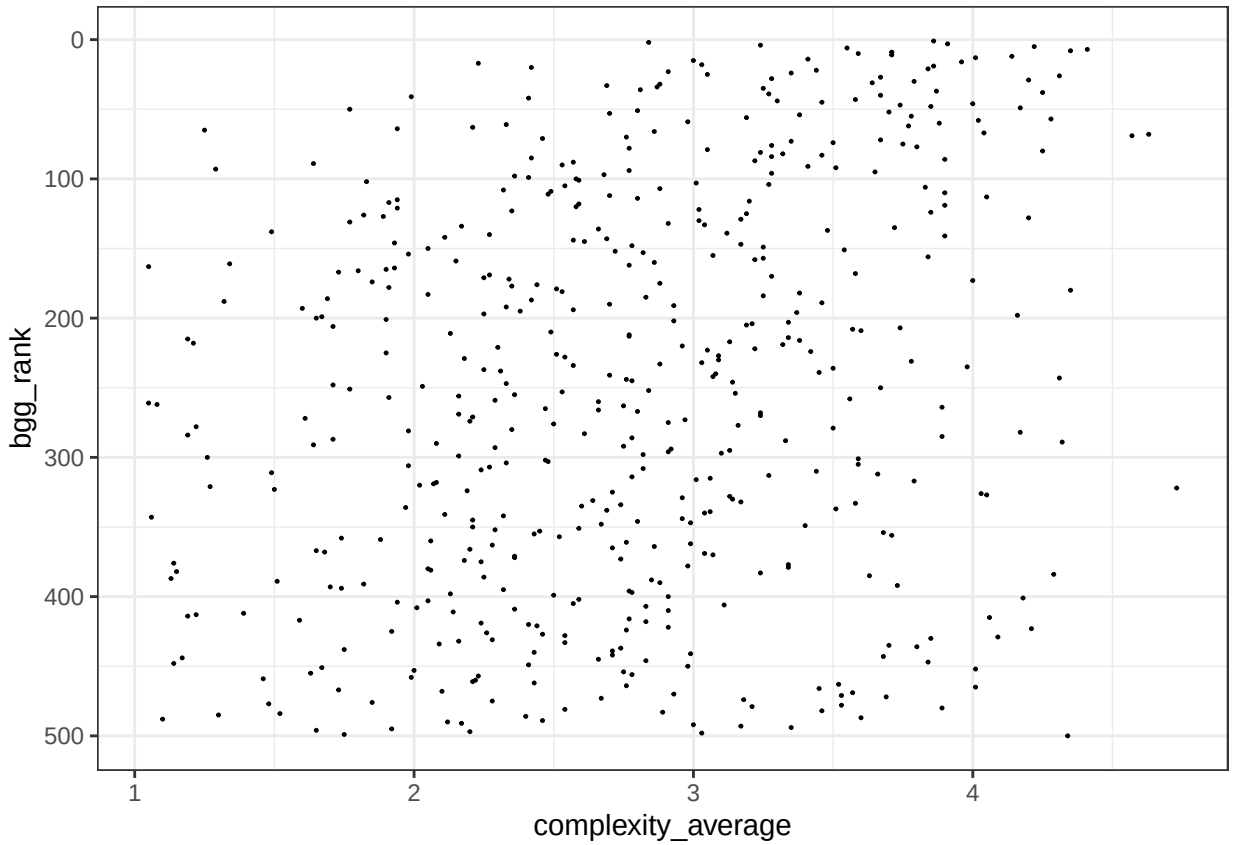

```
# will produce a point plot involving the correlation between Min age and BGG Rank
Min_Age_var_graph <- Board_Games %>% head(500)

ggplot(Min_Age_var_graph, aes(x = min_age, y = bgg_rank)) +
    geom_point(size = 0.2) + scale_y_reverse()
```



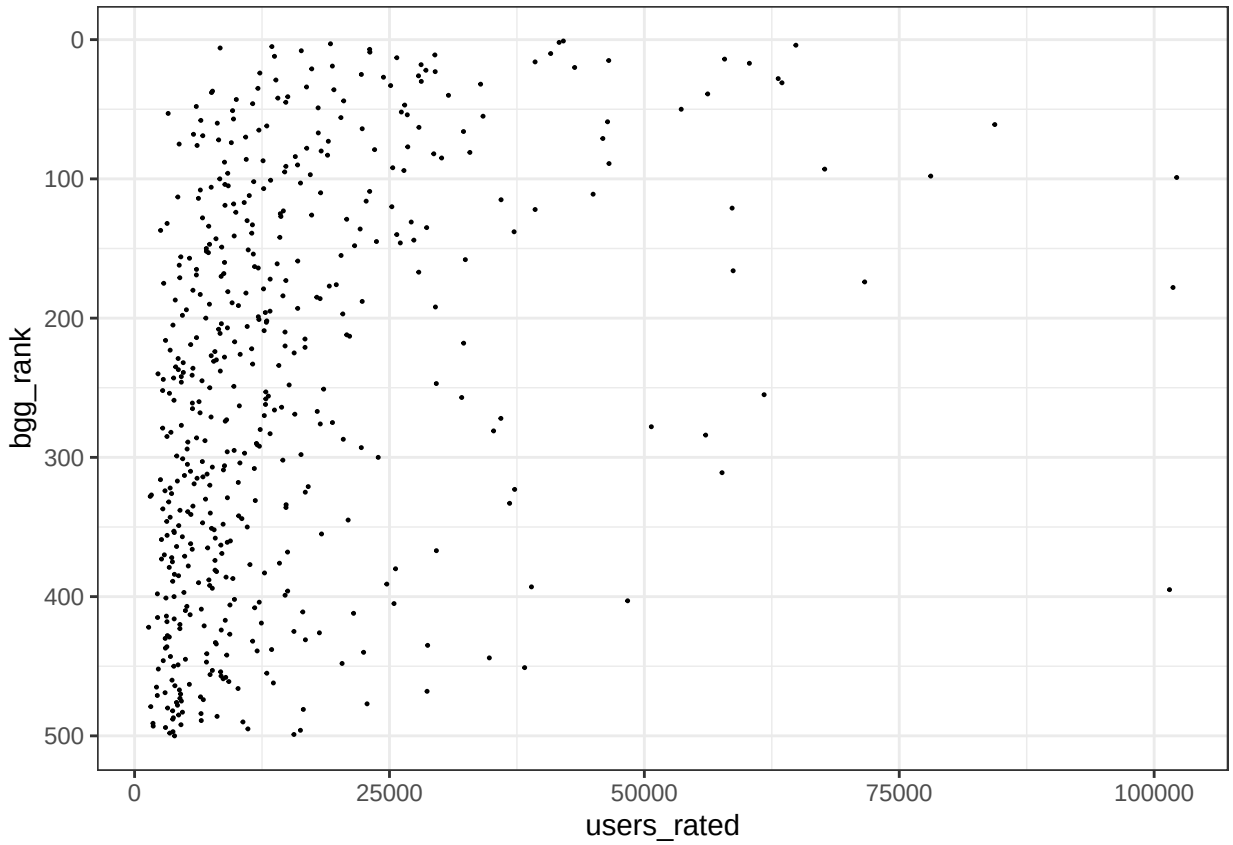

```
# will produce a point plot involving the correlation between the complexity average and the BGG Rank
Complexity_var_graph <- Board_Games %>% head(500)

ggplot(Complexity_var_graph, aes(x = complexity_average, y = bgg_rank)) +geom_point(size = 0.2) + scale_
```



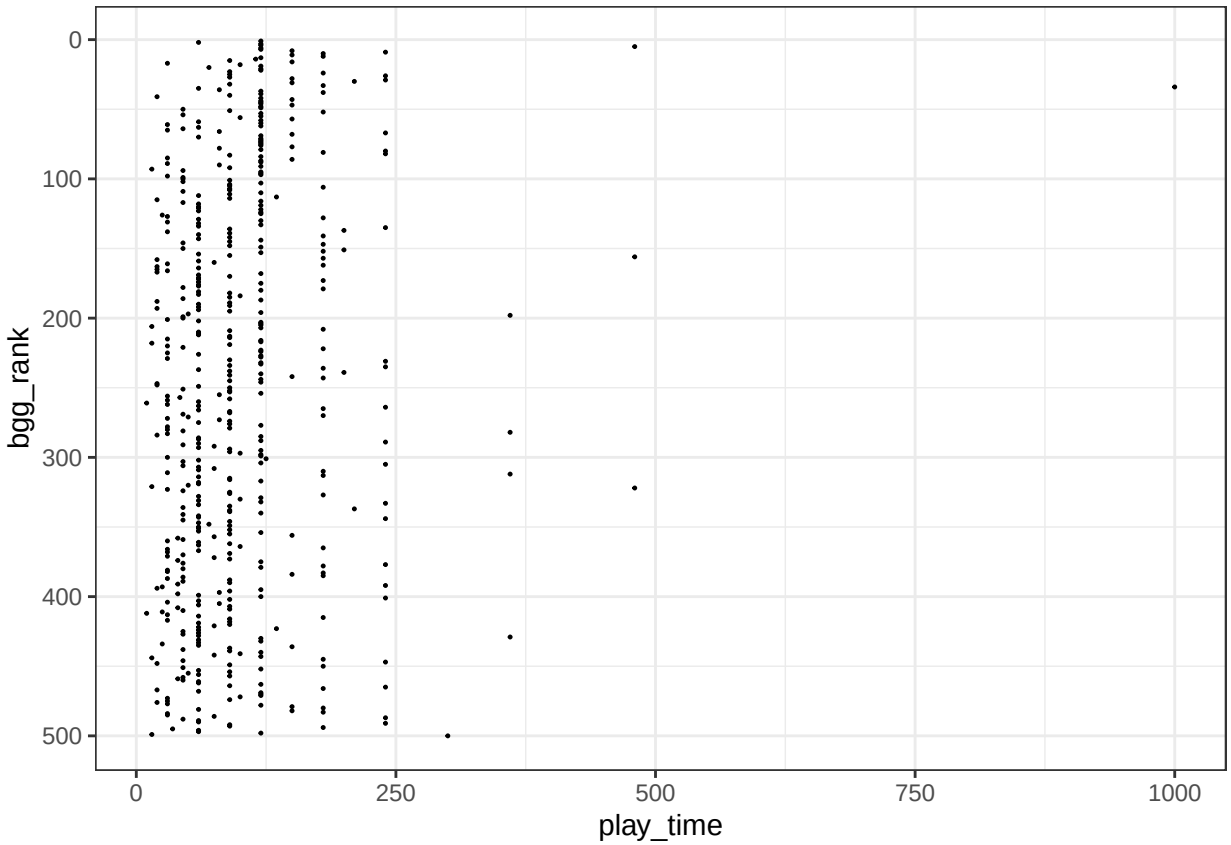

```
# point plot involving the correlation between the users rated variable and BGG rank variable
Users_Rated_var_graph <- Board_Games %>% head(500)

ggplot(Users_Rated_var_graph, aes(x = users_rated, y = bgg_rank)) +
    geom_point(size = 0.2) + scale_y_reverse()
```




```
# point plot involving the correlation between the play time variable and BGG rank variable
play_time_var_graph <- Board_Games %>% head(500)

ggplot(play_time_var_graph, aes(x = play_time, y = bgg_rank)) +
    geom_point(size = 0.2) + scale_y_reverse()
```



The EDA has shown that most board games featured on board game geeks tend to have a max player count of about 4 or so and very large amount of them have a minimum player count of 2. The EDA shows that a majority of board games that have a high bgg rank tend to have a rating greater than 6.0. The analysis has shown that a majority of high ranking board games tend to appear after 2000. The EDA has shown that a majority of high ranking board games tend to have a minimum age rating between 8 and 15. The EDA didn't seem to indicate any obvious trends for the complexity average or users rated variable in relation to the bgg rank variable. Finally the EDA has shown that the majority of high ranking board games tend to have an average play time lower or equal to 120 minutes or two hours.

predictor variables: Year Published, Min Players, Max Players, Min Players, Play time, min age, users rated, rating average, Complexity average, response variable: BGG rank Linear Regression Model

```
split_data <- split(Board_Games, sample(1:nrow(Board_Games) > round(nrow(Board_Games) * .1)))
Training_Board_Games <- split_data$`TRUE`
Test_Board_Games <- split_data$`FALSE`
```

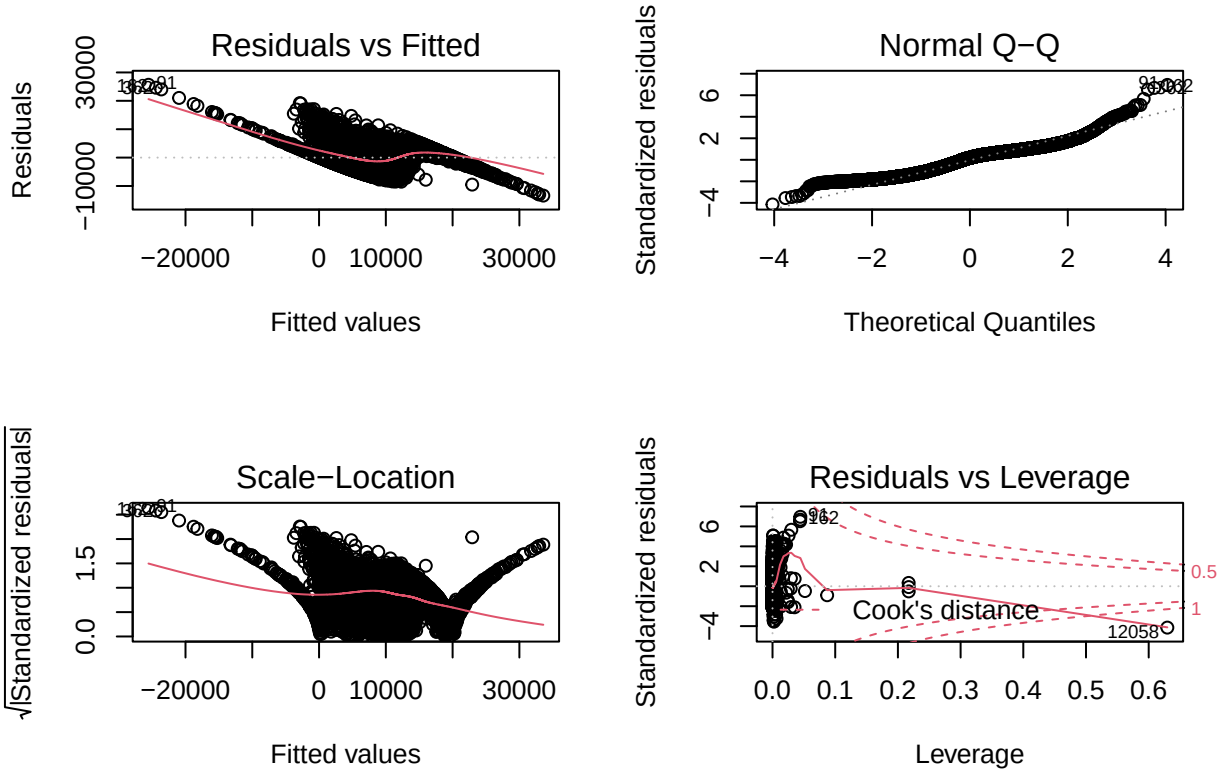
```
BGG_predictor_model = lm(bgg_rank~year_published + min_players + max_players + play_time + min_age + use

summary(BGG_predictor_model)
```

```
##
## Call:
## lm(formula = bgg_rank ~ year_published + min_players + max_players +
##     play_time + min_age + users_rated + rating_average + complexity_average,
##     data = Training_Board_Games)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -13377.0  -3051.8    721.4   2715.0  25643.6
```

```
##
## Coefficients:
##                       Estimate Std. Error  t value Pr(>|t|)
## (Intercept)          4.074e+04  3.306e+02  123.213  < 2e-16 ***
## year_published      -5.113e-01  1.292e-01   -3.957 7.62e-05 ***
## min_players         -1.355e+02  4.159e+01   -3.258  0.00112 **
## max_players         -1.731e+00  1.770e+00   -0.978  0.32788
## play_time            2.151e-01  5.020e-02    4.285 1.84e-05 ***
## min_age             -9.384e+01  8.068e+00  -11.631  < 2e-16 ***
## users_rated         -3.013e-01  7.873e-03  -38.273  < 2e-16 ***
## rating_average      -4.343e+03  3.463e+01 -125.415  < 2e-16 ***
## complexity_average -1.587e+02  3.938e+01   -4.029 5.63e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3768 on 18288 degrees of freedom
##   (12 observations deleted due to missingness)
## Multiple R-squared:  0.5895, Adjusted R-squared:  0.5893
## F-statistic:  3282 on 8 and 18288 DF,  p-value: < 2.2e-16
```

```
par(mfrow =c(2,2))
plot(BGG_predictor_model)
```



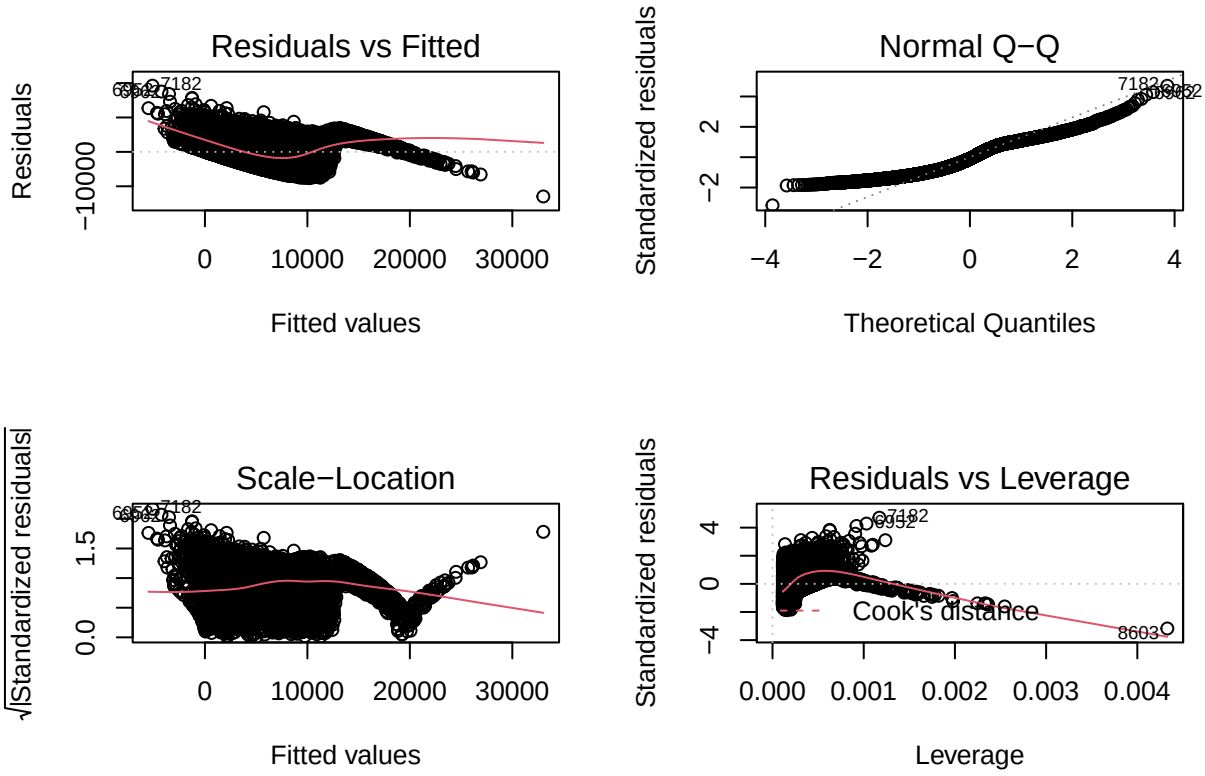

```
best_predictor_linear <- ols_step_best_subset(BGG_predictor_model)
best_predictor_linear
```

```
##                                             Best Subsets Regression
## ---------------------------------------------------------------------------------------------------
## Model Index    Predictors
## ---------------------------------------------------------------------------------------------------
```

```
##      1         rating_average
##      2         users_rated rating_average
##      3         min_age users_rated rating_average
##      4         year_published min_age users_rated rating_average
##      5         year_published play_time min_age users_rated rating_average
##      6         year_published play_time min_age users_rated rating_average complexity_average
##      7         year_published min_players play_time min_age users_rated rating_average complexity_ave
##      8         year_published min_players max_players play_time min_age users_rated rating_average co
## ----------------------------------------------------------------------------------------------------
##
##                                                                  Subsets Regression Summary
## ----------------------------------------------------------------------------------------------------
##                        Adj.        Pred
## Model    R-Square    R-Square    R-Square      C(p)           AIC           SBIC            SBC
## ----------------------------------------------------------------------------------------------------
##   1        0.5502      0.5502      0.5501    1744.0769    354921.2067    302996.2076    354944.6502
##   2        0.5842      0.5842      0.5835     230.4451    353484.0535    301559.3438    353515.3115
##   3        0.5883      0.5882      0.5876      51.4594    353306.3817    301381.7261    353345.4542
##   4        0.5886      0.5885      0.5879      38.3196    353293.2715    301368.6194    353340.1585
##   5        0.5889      0.5888      0.5875      27.3534    353282.3208    301357.6740    353337.0222
##   6        0.5892      0.5891      0.5874      17.0115    353271.9851    301347.3461    353334.5010
##   7        0.5894      0.5893      0.5877       7.9573    353262.9294    301338.2997    353333.2598
##   8        0.5895      0.5893      0.5877       9.0000    353263.9716    301339.3438    353342.1165
## ----------------------------------------------------------------------------------------------------
## AIC: Akaike Information Criteria
##  SBIC: Sawa's Bayesian Information Criteria
##  SBC: Schwarz Bayesian Criteria
##  MSEP: Estimated error of prediction, assuming multivariate normality
##  FPE: Final Prediction Error
##  HSP: Hocking's Sp
##  APC: Amemiya Prediction Criteria
```

#based on the results of the ols subset the most prominent predictor variable for BGG Rank is rating average, the following models will be set up either using all predictor variables or just the rating average as the predictor

```
final_BGG_predictor_model_lin <- lm(bgg_rank ~ rating_average, data = na.omit(Training_Board_Games))
par(mfrow =c(2,2))
plot(final_BGG_predictor_model_lin)
```




```
Test_Board_Games <- na.omit(Test_Board_Games)

Metrics::rmse(predict(BGG_predictor_model, Test_Board_Games), Test_Board_Games$bgg_rank)
```

```
## [1] 4186.165
```

```
Metrics::rmse(predict(final_BGG_predictor_model_lin, Test_Board_Games), Test_Board_Games$bgg_rank)
```

```
## [1] 4214.896
```

based on the results the ols indicates that the most important variable for estimating ranking is the rating average and while the rmse score for the linear model that uses all of the predictor variables versus just the rating average is slightly lower, indicating that it is more accurate, it is not a significant difference between the two indicating that the rating average variable is very substantial in estimating the ranking

Penalized Regression Model

```
Training_Board_Games <- na.omit(Training_Board_Games)
Board_Game_recipe <- recipe(bgg_rank ~ year_published + min_players + max_players + play_time + min_age
Board_Game_recipe <- Board_Game_recipe %>% step_center(all_numeric_predictors()) %>% step_scale(all_num

folds <- vfold_cv(Training_Board_Games, v = 10)

Board_Game_regression_model <- linear_reg(penalty = tune(), mixture = 1) %>% set_engine("glmnet")

Board_Game_workflow <- workflow() %>% add_recipe(Board_Game_recipe) %>% add_model(Board_Game_regression_


tuning_grid <- grid_regular(penalty(), levels = 50)


tuning_grid <- tune_grid(Board_Game_workflow, resamples = folds, grid = tuning_grid)
```

```
tuning_grid %>% collect_metrics() %>% filter(.metric == "rmse") %>% arrange(mean)
```

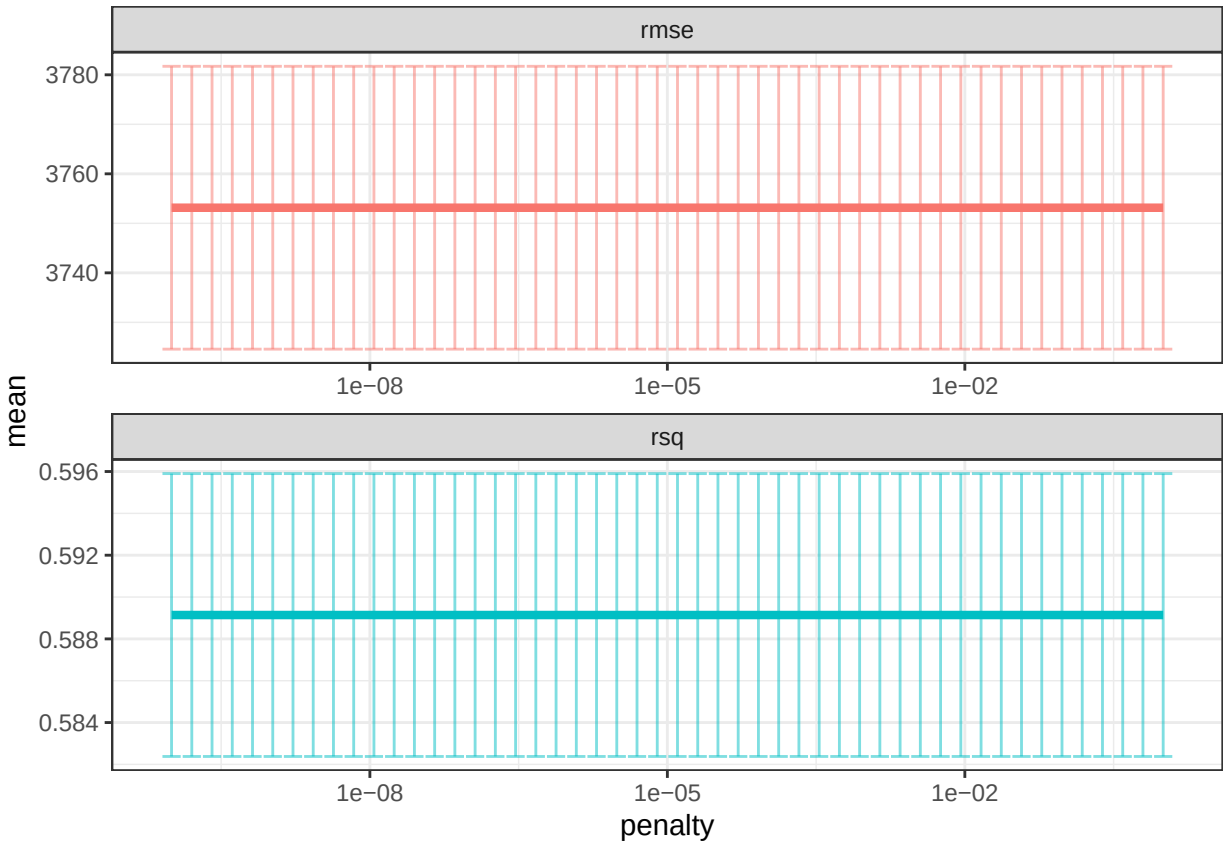
```
## # A tibble: 50 x 7
##     penalty .metric .estimator  mean     n std_err .config
##       <dbl> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
##  1 1   e-10 rmse    standard   3753.    10    28.6 Preprocessor1_Model01
##  2 1.60e-10 rmse    standard   3753.    10    28.6 Preprocessor1_Model02
##  3 2.56e-10 rmse    standard   3753.    10    28.6 Preprocessor1_Model03
##  4 4.09e-10 rmse    standard   3753.    10    28.6 Preprocessor1_Model04
##  5 6.55e-10 rmse    standard   3753.    10    28.6 Preprocessor1_Model05
##  6 1.05e- 9 rmse    standard   3753.    10    28.6 Preprocessor1_Model06
##  7 1.68e- 9 rmse    standard   3753.    10    28.6 Preprocessor1_Model07
##  8 2.68e- 9 rmse    standard   3753.    10    28.6 Preprocessor1_Model08
##  9 4.29e- 9 rmse    standard   3753.    10    28.6 Preprocessor1_Model09
## 10 6.87e- 9 rmse    standard   3753.    10    28.6 Preprocessor1_Model10
## # ... with 40 more rows
```

```
Board_Game_regression_model2 <- linear_reg(penalty = 0.0000000001, mixture = 1) %>% set_engine("glmnet")

Board_Game_workflow2 <- workflow() %>% add_recipe(Board_Game_recipe) %>% add_model(Board_Game_regressio

fit_Board_Game_workflow <- fit(Board_Game_workflow2, Training_Board_Games)

tuning_grid %>%
    collect_metrics() %>%
    ggplot(aes(penalty, mean, color = .metric)) +
    geom_errorbar(aes(
        ymin = mean - std_err,
        ymax = mean + std_err
    ),
    alpha = 0.5
    ) +
    geom_line(size = 1.5) +
    facet_wrap(~.metric, scales = "free", nrow = 2) +
    scale_x_log10() +
    theme(legend.position = "none")
```




```
penalized_predictions <- predict(fit_Board_Game_workflow, Test_Board_Games)
penalized_predictions <- na.omit(penalized_predictions)

Metrics::rmse(Test_Board_Games$bgg_rank, penalized_predictions$.pred)
```

```
## [1] 3794.355
```

due to errors with trying to tune the alternate penalized model that only used the rating average as the predictor variable, however the penalized regression model that uses all of the predictor variables is shown to be more accurate than the linear model based on the rmse score

decision_tree_model

```
Board_Games_refined <- na.omit(Training_Board_Games)

bgg_rank_tree <- tree(bgg_rank ~ year_published + min_players + max_players + play_time + min_age + use

bgg_rank_tree_rating <- tree(bgg_rank ~ rating_average, data = Board_Games_refined)

Metrics::rmse(predict(bgg_rank_tree, Test_Board_Games), Test_Board_Games$bgg_rank)
```

```
## [1] 1641.346
```

```
Metrics::rmse(predict(bgg_rank_tree_rating, Test_Board_Games), Test_Board_Games$bgg_rank)
```

```
## [1] 3722.533
```

the results from the decision tree show that the accuracy of the decision tree that uses only the rating average as a predictor variable is much less accurate then the decision tree that uses all of the predictor variables, however it is close in accuracy to the penalized regression model that uses all of the predictor variables

Conclusion: Among the three different models built during this project the decision tree would be considered the most accurate as it's rmse score was shown to be much lower than the standard linear and penalized regression models although it does come with the caveat that the decision tree model is noted as having less predictive power than other models and is very susceptible to outliers. The penalized model would be considered the second most accurate model among the three shown here.

in terms of which variable has the most influence, the rating average variable has been shown to be the most influential based on the ols best subset result and the various models formed using only rating average having comparable accuracy to the other models formed using all of the predictor variables.