Tidy Tuesday Board Games

M. Zhang

2/21/2022

Introduction

We will be working with two datasets, `ratings` and `details`, sourced from BoardGameGeek via the Tidy Tuesday. This analysis will not be styled as a "final report", but rather as a quick walk through of some data analysis and wrangling that took place while exploring the data.

The MSDSO Discord group for the University of Texas Masters in Data Science Online program will be doing weekly explorations of TidyTuesday as an exercise for improving their data science skill sets in a collaborative environment.

Data Details

This dataset is pulled from the Tidy Tuesday Repository:

Thomas Mock (2022). Tidy Tuesday: A weekly data project aimed at the R ecosystem. https://github.com /rfordatascience/tidytuesday (https://github.com/rfordatascience/tidytuesday).

The original data is from Kaggle (https://www.kaggle.com/jvanelteren/boardgamegeek-reviews/version /3?select=2022-01-08.csv) via Board Game Geek (https://boardgamegeek.com/). The two data sets are joinable in the `id` column, and contain board game rating information, and board game details information.

Data Loading

```
ratings <- readr::read_csv('https://raw.githubusercontent.com/rfordatascience/tidytuesday/master/data/2022/2022-01-25/ratings.csv')
details <- readr::read_csv('https://raw.githubusercontent.com/rfordatascience/tidytuesday/master/data/2022/2022-01-25/details.csv')
```

Data Dictionary

The following tables contain information in regards to the columns available.

ratings.csv

variable	class	description
num	double	Game number
id	double	Game ID
name	character	Game name
year	double	Game year
rank	double	Game rank
average	double	Average rating
bayes_average	double	Bayes average rating
users_rated	double	Users rated
url	character	Game url
thumbnail	character	Game thumbnail

details.csv

variable	class	description
num	double	Game number
id	double	Game ID
primary	character	Primary name
description	character	Description of game
yearpublished	double	Year published
minplayers	double	Min n of players
maxplayers	double	Max n of players
playingtime	double	Playing time in minutes
minplaytime	double	Min play time
maxplaytime	double	Max plat tome
minage	double	minimum age

variable	class	description
boardgamecategory	character	Category
boardgamemechanic	character	Mechanic
boardgamefamily	character	Board game family
boardgameexpansion	character	Expansion
boardgameimplementation	character	Implementation
boardgamedesigner	character	Designer
boardgameartist	character	Artist
boardgamepublisher	character	Publisher
owned	double	Num owned
trading	double	Num trading
wanting	double	Num wanting
wishing	double	Num wishing

Data Exploration and wrangling

Based on the data dictionary and information provided on the dataset, this appears to be a good opportunity to practice using `join` functions. Additionally, we will explore which columns are good candidates for exploring summary data using the `group_by` function. In the end, we will attempt to put together a few meaningful graphics on the data summarizing user ratings and game rankings.

Examining the raw data

First, we will use the `names` function to look at the information to quickly confirm the descriptions provided by the data dictionary.

```
ratings
```

```
# A tibble: 21,831 x 10
     num     id name          year  rank average bayes_average users_rated url
   <dbl>  <dbl> <chr>        <dbl> <dbl>   <dbl>         <dbl>       <dbl> <chr>
 1   105  30549 Pandemic      2008   106    7.59          7.49      108975 /boa~
 2   189    822 Carcassonne   2000   190    7.42          7.31      108738 /boa~
 3   428     13 Catan         1995   429    7.14          6.97      108024 /boa~
 4    72  68448 7 Wonders     2010    73    7.74          7.63       89982 /boa~
 5   103  36218 Dominion      2008   104    7.61          7.50       81561 /boa~
 6   191   9209 Ticket to R~  2004   192    7.41          7.30       76171 /boa~
 7   100 178900 Codenames     2015   101    7.6           7.51       74419 /boa~
 8     3 167791 Terraformin~  2016     4    8.42          8.27       74216 /boa~
 9    15 173346 7 Wonders D~  2015    16    8.11          7.98       69472 /boa~
10    35  31260 Agricola      2007    36    7.93          7.81       66093 /boa~
# ... with 21,821 more rows, and 1 more variable: thumbnail <chr>
```

```
names(ratings)
```

```
 [1] "num"           "id"            "name"          "year"
 [5] "rank"          "average"       "bayes_average" "users_rated"
 [9] "url"           "thumbnail"
```

```
ratings_count <- ratings %>% summarize(count = n())
```

```
details
```

```
# A tibble: 21,631 x 23
     num     id primary          description yearpublished minplayers maxplayers
   <dbl>  <dbl> <chr>            <chr>                <dbl>      <dbl>      <dbl>
 1     0  30549 Pandemic         In Pandemi~           2008          2          4
 2     1    822 Carcassonne      Carcassonn~           2000          2          5
 3     2     13 Catan            In CATAN (~           1995          3          4
 4     3  68448 7 Wonders        You are th~           2010          2          7
 5     4  36218 Dominion         &amp;quot;You ~       2008          2          4
 6     5   9209 Ticket to Ride   With elega~           2004          2          5
 7     6 178900 Codenames        Codenames ~           2015          2          8
 8     7 167791 Terraforming Ma~ In the 240~           2016          1          5
 9     8 173346 7 Wonders Duel   In many wa~           2015          2          2
10     9  31260 Agricola         Descriptio~           2007          1          5
# ... with 21,621 more rows, and 16 more variables: playingtime <dbl>,
#   minplaytime <dbl>, maxplaytime <dbl>, minage <dbl>,
#   boardgamecategory <chr>, boardgamemechanic <chr>, boardgamefamily <chr>,
#   boardgameexpansion <chr>, boardgameimplementation <chr>,
#   boardgamedesigner <chr>, boardgameartist <chr>, boardgamepublisher <chr>,
#   owned <dbl>, trading <dbl>, wanting <dbl>, wishing <dbl>
```

```
names(details)
```

```
 [1] "num"                   "id"
 [3] "primary"               "description"
 [5] "yearpublished"         "minplayers"
 [7] "maxplayers"            "playingtime"
 [9] "minplaytime"           "maxplaytime"
[11] "minage"                "boardgamecategory"
[13] "boardgamemechanic"     "boardgamefamily"
[15] "boardgameexpansion"    "boardgameimplementation"
[17] "boardgamedesigner"     "boardgameartist"
[19] "boardgamepublisher"    "owned"
[21] "trading"               "wanting"
[23] "wishing"
```

```
details_count <- details %>% summarize(count = n())
```

- Ratings table total entries: 21831
- Details table total entries: 21631

Joining two tables

Based off the first 6 entries shown in each data table, it does appear that the `id` column should allow for linkage of the two tables. There are less entries in the details dataset though, so we will left join on the `details` dataset and drop non-matching entries.

```
board_games <- ratings %>% left_join(details, by = "id")
board_games
```

```
# A tibble: 21,831 x 32
   num.x     id name          year  rank average bayes_average users_rated url
   <dbl>  <dbl> <chr>        <dbl> <dbl>   <dbl>         <dbl>       <dbl> <chr>
 1   105  30549 Pandemic      2008   106    7.59          7.49      108975 /boa~
 2   189    822 Carcassonne   2000   190    7.42          7.31      108738 /boa~
 3   428     13 Catan         1995   429    7.14          6.97      108024 /boa~
 4    72  68448 7 Wonders     2010    73    7.74          7.63       89982 /boa~
 5   103  36218 Dominion      2008   104    7.61          7.50       81561 /boa~
 6   191   9209 Ticket to R~  2004   192    7.41          7.30       76171 /boa~
 7   100 178900 Codenames     2015   101    7.6           7.51       74419 /boa~
 8     3 167791 Terraformin~  2016     4    8.42          8.27       74216 /boa~
 9    15 173346 7 Wonders D~  2015    16    8.11          7.98       69472 /boa~
10    35  31260 Agricola      2007    36    7.93          7.81       66093 /boa~
# ... with 21,821 more rows, and 23 more variables: thumbnail <chr>,
#   num.y <dbl>, primary <chr>, description <chr>, yearpublished <dbl>,
#   minplayers <dbl>, maxplayers <dbl>, playingtime <dbl>, minplaytime <dbl>,
#   maxplaytime <dbl>, minage <dbl>, boardgamecategory <chr>,
#   boardgamemechanic <chr>, boardgamefamily <chr>, boardgameexpansion <chr>,
#   boardgameimplementation <chr>, boardgamedesigner <chr>,
#   boardgameartist <chr>, boardgamepublisher <chr>, owned <dbl>, ...
```

We can see based off the total number of entries in the new table `board_games` that we were successful in our `left_join`

Checking for unique entries

Next, we will see if each game (`primary`) has a unique entry in the data set.

```
board_games %>%
  group_by(primary) %>%
  summarize(
    n = n()) %>%
  arrange(desc(n))
```

```
# A tibble: 21,237 x 2
   primary                      n
   <chr>                    <int>
 1 <NA>                       200
 2 Robin Hood                   6
 3 Chaos                        4
 4 Cosmic Encounter             4
 5 Gangster                     4
 6 Gettysburg                   4
 7 Saga                         4
 8 Warhammer 40,000: Kill Team  4
 9 Airlines                     3
10 Ali Baba                     3
# ... with 21,227 more rows
```

It can be seen that there are multiple entries per game. Based off the data set information, it may be that the same game with different publication dates maintain separate entries. We will modify the original summary to check:

```
board_games %>%
  group_by(primary, yearpublished) %>%
  summarize(
    n = n(), .groups = 'drop') %>%
  arrange(desc(n))
```

```
# A tibble: 21,627 x 3
   primary                                                  yearpublished     n
   <chr>                                                            <dbl> <int>
 1  <NA>                                                             NA   200
 2 "Cahoots"                                                        2018     2
 3 "Chaos"                                                          2010     2
 4 "Columbus"                                                       1991     2
 5 "DIG"                                                            2017     2
 6 "Loch Ness"                                                      2010     2
 7 "'65: Squad-Level Combat in the Jungles of Vietnam"              2016     1
 8 "'CA' Tactical Naval Warfare in the Pacific, 1941-45"            1973     1
 9 "'Wacht am Rhein': The Battle of the Bulge, 16 Dec 44 - ~        1977     1
10 "\"La Garde recule!\""                                           2011     1
# ... with 21,617 more rows
```

We still see some duplicate entries so we will pull up a specific game to examine the data:

```
board_games %>%
  select(primary, boardgamepublisher, average, description) %>%
  filter(primary == "Cahoots")
```

```
# A tibble: 2 x 4
  primary boardgamepublisher                                 average description
  <chr>   <chr>                                                <dbl> <chr>
1 Cahoots ['Gamewright', 'Oya', 'Brain Picnic', 'Lifestyle ~    7.18 Cooperatio~
2 Cahoots ['Mayday Games']                                     6.47 Welcome to~
```

It appears that the games have separate publishers and therefore have separate entries. As a crude solution for the purposes of this analysis, will move forward since the number of entries in which multiples appear are small.

Grouping candidates

For summarizing and visualizing the data, a few potential grouping candidates based off the `details` dataset are `primary`, `boardgamecategory`, `boardgamemechanic`, and `boardgamepublisher`. We will use `sapply` and `n_discinct`.

```
board_games %>%
  select(primary, boardgamecategory, boardgamemechanic, boardgamepublisher) %>%
  sapply(n_distinct)
```

```
          primary  boardgamecategory  boardgamemechanic boardgamepublisher
            21237               6731               8292              11266
```

Unfortunately, we can see from the above that the number of unique entries for each row is much higher than anticipated. To understand why, we will look at the actual data contained in these columns.

```
board_games %>%
  select(
    primary,
    boardgamecategory,
    boardgamemechanic,
    boardgamepublisher) %>%
  head
```

```
# A tibble: 6 x 4
  primary        boardgamecategory             boardgamemechan~ boardgamepublis~
  <chr>          <chr>                         <chr>            <chr>
1 Pandemic       ['Medical']                   ['Action Points~ ['Z-Man Games',~
2 Carcassonne    ['City Building', 'Medieval'~ ['Area Majority~ ['Hans im Glück~
3 Catan          ['Economic', 'Negotiation']   ['Dice Rolling'~ ['KOSMOS', '999~
4 7 Wonders      ['Ancient', 'Card Game', 'Ci~ ['Drafting', 'H~ ['Repos Product~
5 Dominion       ['Card Game', 'Medieval']     ['Deck, Bag, an~ ['Rio Grande Ga~
6 Ticket to Ride ['Trains']                    ['Card Drafting~ ['Days of Wonde~
```

It appears that the `boardgame` columns all actually contain more than single value entries, leading to large number of permutations of values. These columns will not be suitable for quick data analysis. Thus, we will examine some other parameters. (This will be examined at the end **Extra Work** section where we tokenize some of these columns).

```
board_games %>%
  select(playingtime, minplayers, maxplayers, minage) %>%
  sapply(n_distinct)
```

```
playingtime  minplayers  maxplayers      minage
        120          12          53          22
```

```
board_games %>%
  select(
    playingtime,
    minplayers,
    maxplayers,
    minage) %>%
  head
```

```
# A tibble: 6 x 4
  playingtime minplayers maxplayers minage
        <dbl>      <dbl>      <dbl>  <dbl>
1          45          2          4      8
2          45          2          5      7
3         120          3          4     10
4          30          2          7     10
5          30          2          4     13
6          60          2          5      8
```

```
board_games %>%
  select(
    playingtime,
    minplayers,
    maxplayers,
    minage) %>%
  summary()
```

```
  playingtime         minplayers       maxplayers         minage
 Min.   :    0.00   Min.   : 0.000   Min.   :  0.00   Min.   : 0.000
 1st Qu.:   25.00   1st Qu.: 2.000   1st Qu.:  4.00   1st Qu.: 8.000
 Median :   45.00   Median : 2.000   Median :  4.00   Median :10.000
 Mean   :   90.51   Mean   : 2.007   Mean   :  5.71   Mean   : 9.611
 3rd Qu.:   90.00   3rd Qu.: 2.000   3rd Qu.:  6.00   3rd Qu.:12.000
 Max.   :60000.00   Max.   :10.000   Max.   :999.00   Max.   :25.000
 NA's   :200        NA's   :200      NA's   :200      NA's   :200
```

These are all numerical values with reasonable distributions (though there appears to be some outlier values which may be removed later).

Data cleaning for NA values

```
board_games %>%
  summarize(across(everything(), ~ sum(is.na(.)))) %>%
  tidyr::pivot_longer(everything()) %>%
  arrange(desc(value)) %>%
  deframe()
```

```
boardgameimplementation      boardgameexpansion         boardgameartist
                   16969                   16325                    6107
        boardgamefamily       boardgamemechanic       boardgamedesigner
                    3961                    1790                     796
      boardgamecategory             description      boardgamepublisher
                     483                     201                     201
                 num.y                 primary          yearpublished
                     200                     200                     200
             minplayers              maxplayers            playingtime
                     200                     200                     200
            minplaytime             maxplaytime                  minage
                     200                     200                     200
                  owned                 trading                 wanting
                     200                     200                     200
                wishing               thumbnail                   num.x
                     200                       6                       0
                     id                    name                    year
                      0                       0                       0
                   rank                 average           bayes_average
                      0                       0                       0
            users_rated                     url
                      0                       0
```

NA counting methodology was taken from stackexchange (https://stackoverflow.com/questions/63198917/using-tidyverse-pipeline-to-count-nas-and-reorder)

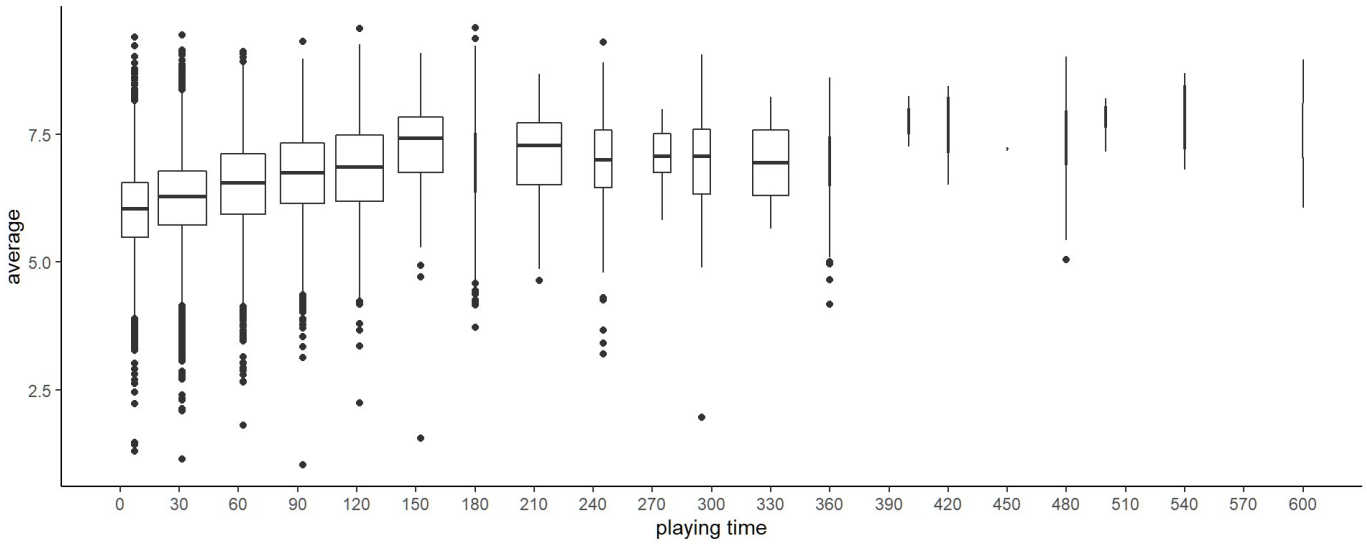
We see there are a number of NA values, but that `playingtime`, `rank`, and `average` are all populated. Thus, we will now move onto data visualizations.

Data Visualizations

```
board_games %>%
  select(
    average,
    rank,
    playingtime,
    minplayers,
    maxplayers,
    minage) %>%
  ggplot() +
  aes(x = playingtime, y = average, group = cut_width(playingtime, 30)) +
  geom_boxplot() +
  scale_x_continuous(
    name = "playing time",
    breaks = seq(0, 600, 30),
    labels = seq(0, 600, 30),
    limits = c(0, 600)) +
  theme_classic()
```

```
Warning: Removed 392 rows containing missing values (stat_boxplot).
```

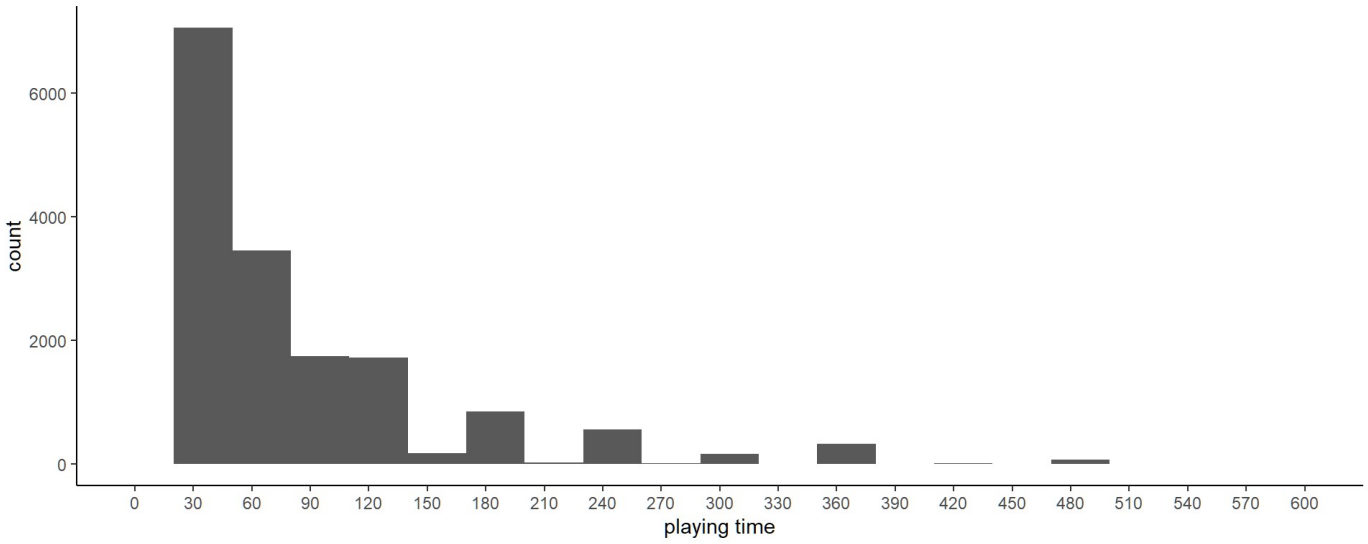
```
Warning: Removed 1 rows containing missing values (geom_segment).
```




```
board_games %>%
  select(
    average,
    rank,
    playingtime,
    minplayers,
    maxplayers,
    minage) %>%
  ggplot() +
  aes(playingtime) +
  geom_histogram(
    binwidth = 30,
    center = 5) +
  scale_x_continuous(
    name = "playing time",
    breaks = seq(0, 600, 30),
    labels = seq(0, 600, 30),
    limits = c(0, 600)) +
  theme_classic()
```

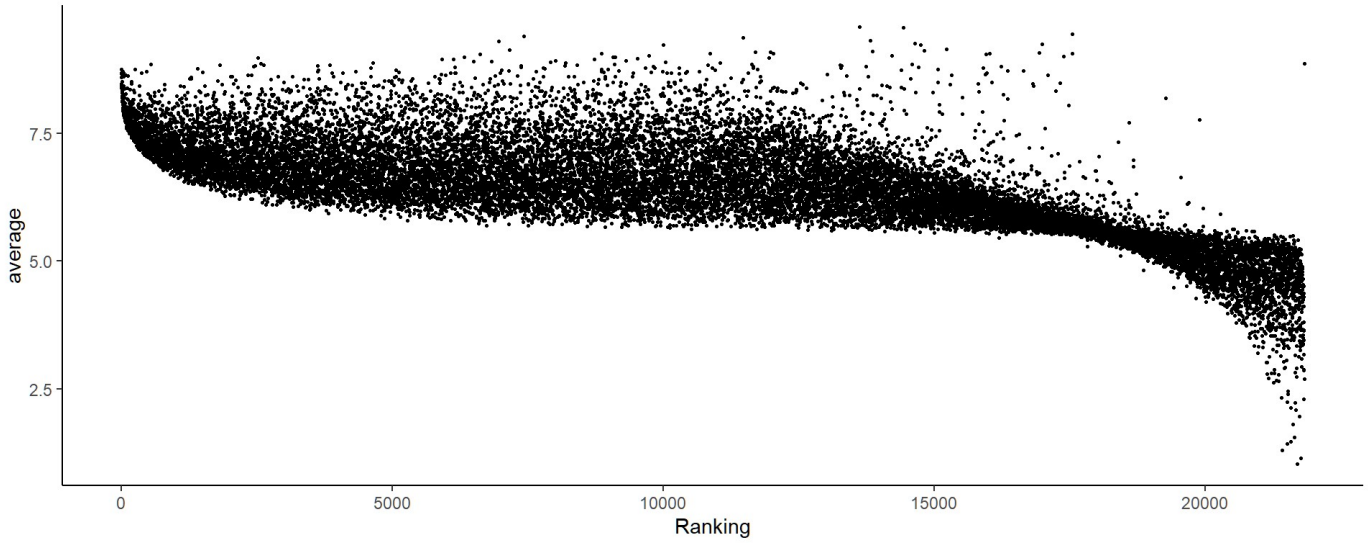
```
Warning: Removed 392 rows containing non-finite values (stat_bin).
```

```
Warning: Removed 2 rows containing missing values (geom_bar).
```



The range of scores appears to diminish as the playing time for the game increases. It can also be seen that the number of games which extend to longer playtimes tapers off very quickly. There were also more rows of data being dropped than expected - this is a point for future analysis if this dataset is revisited.

```
board_games %>%
  select(
    average,
    rank,
    playingtime,
    minplayers,
    maxplayers,
    minage) %>%
  ggplot() +
  aes(x = rank, y = average) +
  geom_point(size = 0.5) +
  scale_x_continuous(
    name = "Ranking") +
  theme_classic()
```



Ranking and score have relationship where the second order rate changes signs at larger numerical values for ranking.

Conclusion

This ended up being a shorter initial exploration of the board games dataset. One finding of note was that there was a lot of game information which still needs to be parsed out of the `boardgame` columns which actually contain delimited lists of information which could allow for more interesting analysis and charts in the future.

Thanks for reading and joining me on my journey to improve my skills in data analysis.

EXTRA - Predict Rating using ML

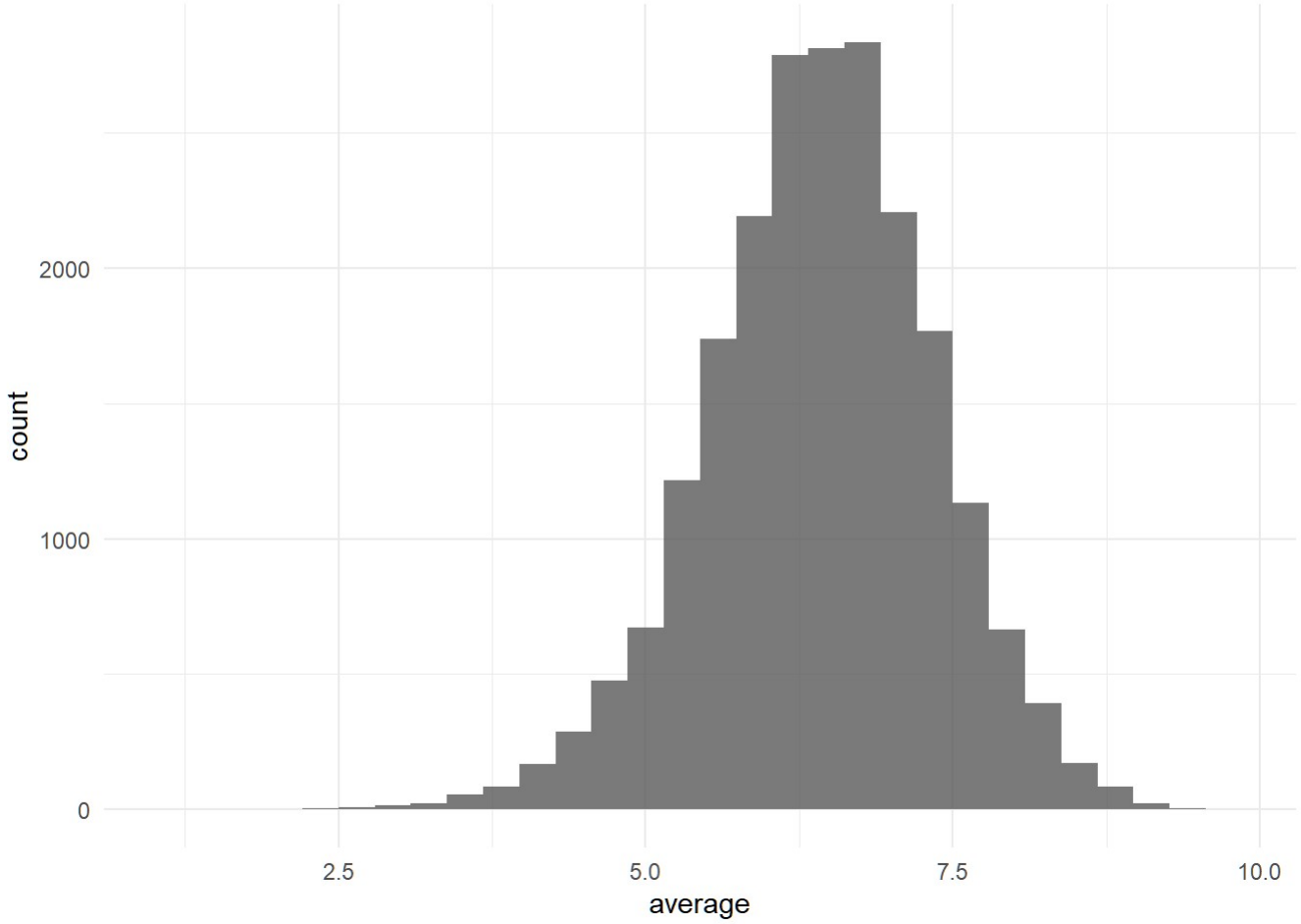
This additional analysis was done at a later date (3/10/2022) following the tutorial done by Julia Silge (https://youtu.be/HTJ0nt3codo) on this same data set

Additional Exploration

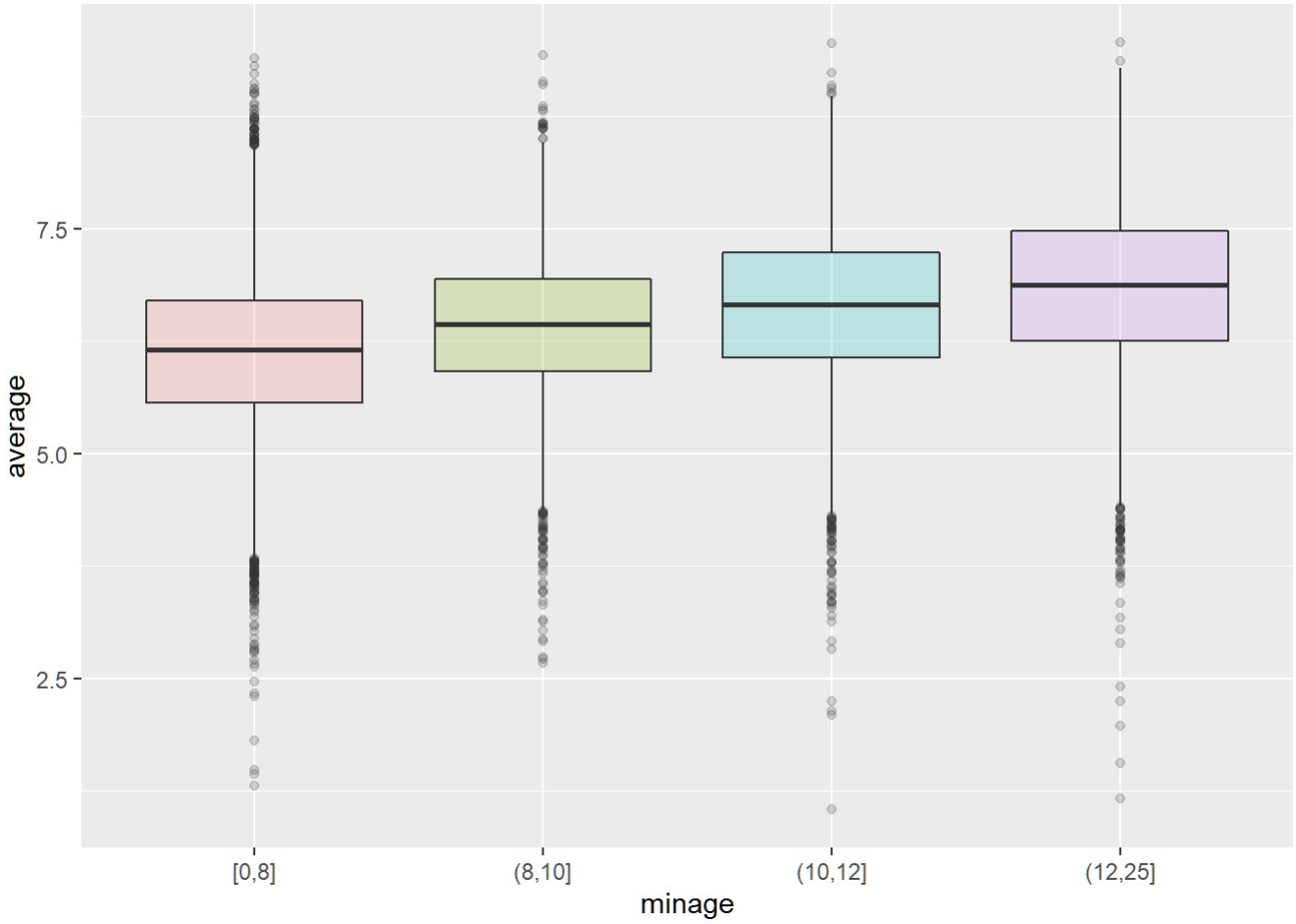
Some additional data exploration ideas for viewing information in data set.

```
ggplot(board_games, aes(average)) +
  geom_histogram(alpha = 0.8) + theme_minimal()
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



```
board_games %>%
  filter(!is.na(minage)) %>%
  mutate(minage = cut_number(minage, 4)) %>%
  ggplot(aes(minage, average, fill = minage)) +
  geom_boxplot(alpha = 0.2, show.legend = FALSE)
```



Tune an XGBoost Model

We will be using an XGBoost model to predict average rating using boardgamecategory, and min|max columns.

Training split will be done using vfold average strata method.

```
library(tidymodels)
```

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

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

```
v broom        0.7.12     v rsample      0.1.1
v dials        0.1.0      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      0.2.0      v yardstick    0.0.9
v recipes      0.2.0
```

```
-- 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()
x tune::tune()      masks parsnip::tune()
* Learn how to get started at https://www.tidymodels.org/start/
```

```
set.seed(123)
game_split <-
  board_games %>%
  select(name, average, matches("min|max"), boardgamecategory) %>%
  na.omit() %>%
  initial_split(strata = average)

game_train <- training(game_split)
game_test <- testing(game_split)


set.seed(234)
game_folds <- vfold_cv(game_train, strata = average)
game_folds
```

```
# A tibble: 10 x 2
   splits               id
   <list>               <chr>
 1 <split [14407/1602]> Fold01
 2 <split [14407/1602]> Fold02
 3 <split [14407/1602]> Fold03
 4 <split [14408/1601]> Fold04
 5 <split [14408/1601]> Fold05
 6 <split [14408/1601]> Fold06
 7 <split [14408/1601]> Fold07
 8 <split [14408/1601]> Fold08
 9 <split [14410/1599]> Fold09
10 <split [14410/1599]> Fold10
```

Since the `boardgamecategory` is in string format, it needs to be tokenized for feature engineering.

```
library(textrecipes)

# create a customized tokenizer

split_category <- function(x) {
  x %>%
    str_split(", ") %>%
    map(str_remove_all, "[:punct:]") %>%
    map(str_to_lower) %>%
    map(str_squish) %>%
    map(str_replace_all, " ", "_")
}

# Feature engineering

game_rec <-
  recipe(average ~., data = game_train) %>%
  update_role(name, new_role = "id") %>%
  step_tokenize(boardgamecategory, custom_token = split_category) %>%
  step_tokenfilter(boardgamecategory, max_tokens = 30) %>%
  step_tf(boardgamecategory)

# Not required for model training - just for pre-examination
game_prep <- prep(game_rec)
bake(game_prep, new_data = NULL) %>% str()
```

```
tibble [16,009 x 37] (S3: tbl_df/tbl/data.frame)
 $ name                                    : Factor w/ 15781 levels "'65: Squad-Lev
el Combat in the Jungles of Vietnam",..: 10875 8605 14660 876 15746 6837 13331 1508 3
161 9951 ...
 $ minplayers                              : num [1:16009] 2 2 2 4 2 1 2 2 4 2 ...
 $ maxplayers                              : num [1:16009] 6 8 10 10 6 8 6 2 16 6
...
 $ minplaytime                             : num [1:16009] 120 60 30 30 60 20 60 30
60 45 ...
 $ maxplaytime                             : num [1:16009] 120 180 30 30 90 20 60 3
0 60 45 ...
 $ minage                                  : num [1:16009] 10 8 6 12 15 6 8 8 13 8
...
 $ average                                 : num [1:16009] 5.59 4.37 5.41 5.79 5.8
5.62 4.31 4.66 5.68 5.14 ...
 $ tf_boardgamecategory_abstract_strategy  : num [1:16009] 0 0 0 0 0 0 0 0 0 0 ...
 $ tf_boardgamecategory_action_dexterity   : num [1:16009] 0 0 0 0 0 1 0 0 0 0 ...
 $ tf_boardgamecategory_adventure          : num [1:16009] 0 0 0 0 0 0 0 0 0 0 ...
 $ tf_boardgamecategory_ancient            : num [1:16009] 0 0 0 0 0 0 0 0 0 0 ...
 $ tf_boardgamecategory_animals            : num [1:16009] 0 0 0 0 0 0 0 0 0 0 ...
 $ tf_boardgamecategory_bluffing           : num [1:16009] 0 0 0 0 0 0 0 0 0 0 ...
 $ tf_boardgamecategory_card_game          : num [1:16009] 0 0 1 1 0 0 0 0 0 1 ...
 $ tf_boardgamecategory_childrens_game     : num [1:16009] 0 0 0 0 0 0 1 1 0 0 ...
 $ tf_boardgamecategory_deduction          : num [1:16009] 0 0 0 0 0 0 0 1 0 0 ...
 $ tf_boardgamecategory_dice               : num [1:16009] 0 0 0 0 0 0 0 0 0 0 ...
 $ tf_boardgamecategory_economic           : num [1:16009] 0 1 0 0 0 0 1 0 0 0 ...
 $ tf_boardgamecategory_exploration        : num [1:16009] 0 0 0 0 1 0 0 0 0 0 ...
 $ tf_boardgamecategory_fantasy            : num [1:16009] 0 0 0 0 0 0 0 0 0 0 ...
 $ tf_boardgamecategory_fighting           : num [1:16009] 0 0 0 0 1 0 0 0 0 0 ...
 $ tf_boardgamecategory_horror             : num [1:16009] 0 0 0 0 1 0 0 0 0 0 ...
 $ tf_boardgamecategory_humor              : num [1:16009] 0 0 0 1 0 0 0 0 0 0 ...
 $ tf_boardgamecategory_medieval           : num [1:16009] 0 0 0 0 0 0 0 0 0 0 ...
 $ tf_boardgamecategory_miniatures         : num [1:16009] 0 0 0 0 1 0 0 0 0 0 ...
 $ tf_boardgamecategory_movies_tv_radio_theme: num [1:16009] 0 0 1 0 1 0 0 0 0 0 ...
 $ tf_boardgamecategory_nautical           : num [1:16009] 0 0 0 0 0 0 0 1 0 0 ...
 $ tf_boardgamecategory_negotiation        : num [1:16009] 0 1 0 0 0 0 0 0 0 0 ...
 $ tf_boardgamecategory_party_game         : num [1:16009] 0 0 0 1 0 1 0 0 1 0 ...
 $ tf_boardgamecategory_print_play         : num [1:16009] 0 0 0 0 0 0 0 0 0 0 ...
 $ tf_boardgamecategory_puzzle             : num [1:16009] 0 0 0 0 0 0 0 0 1 0 ...
 $ tf_boardgamecategory_racing             : num [1:16009] 0 0 0 0 0 0 0 0 0 0 ...
 $ tf_boardgamecategory_realtime           : num [1:16009] 0 0 0 0 0 0 0 0 0 0 ...
 $ tf_boardgamecategory_science_fiction    : num [1:16009] 0 0 0 0 0 0 0 0 0 0 ...
 $ tf_boardgamecategory_trivia             : num [1:16009] 0 0 0 0 0 0 0 0 1 0 ...
 $ tf_boardgamecategory_wargame            : num [1:16009] 1 0 0 0 0 0 0 1 0 0 ...
 $ tf_boardgamecategory_world_war_ii       : num [1:16009] 0 0 0 0 0 0 0 0 0 0 ...
```

We will now create a model specification for XGBoost.

```
# Create a model specification

xgb_spec <-
  boost_tree(
    trees = tune(),
    mtry = tune(),
    min_n = tune(),
    learn_rate = 0.01
  ) %>%
  set_engine("xgboost") %>%
  set_mode("regression")


xgb_wf <- workflow(game_rec, xgb_spec)
xgb_wf
```

```
== Workflow ====================================================================
Preprocessor: Recipe
Model: boost_tree()

-- Preprocessor ----------------------------------------------------------------
3 Recipe Steps

* step_tokenize()
* step_tokenfilter()
* step_tf()

-- Model -----------------------------------------------------------------------
Boosted Tree Model Specification (regression)

Main Arguments:
  mtry = tune()
  trees = tune()
  min_n = tune()
  learn_rate = 0.01

Computational engine: xgboost
```

For the unspecified parameters, we will utilize the library finetune and a 20-race grid to find the best fit model.

```
library(doParallel)
```

```
Loading required package: foreach
```

```

Attaching package: 'foreach'
```

```
The following objects are masked from 'package:purrr':

    accumulate, when
```

```
Loading required package: iterators
```

```
Loading required package: parallel
```

```
registerDoParallel(cores=2)
getDoParWorkers()
```

```
[1] 2
```

```
registerDoSEQ()
getDoParWorkers()
```

```
[1] 1
```

```
set.seed(234)
library(finetune)
```

```
Registered S3 method overwritten by 'finetune':
  method            from
  obj_sum.tune_race tune
```

Note: I was unable to get doParallel to work with XGBoost on Windows but leaving code block for future reference.

```
xgb_game_rs <-
  tune_race_anova(
    xgb_wf,
    game_folds,
    grid = 20,
    control = control_race(verbose_elim = TRUE, pkgs = c("stringr"))
  )
```

```
i Creating pre-processing data to finalize unknown parameter: mtry
```

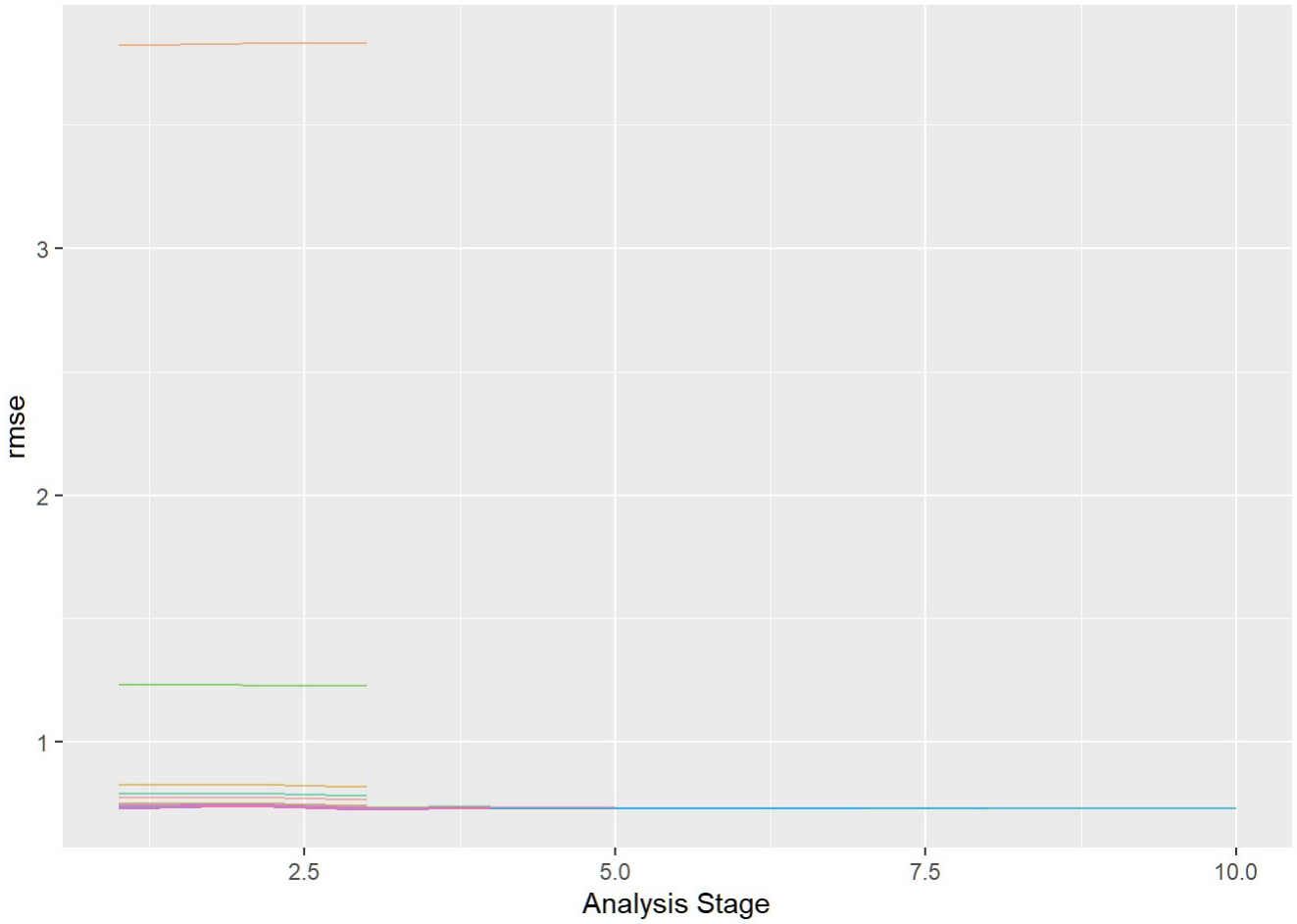
```
i Racing will minimize the rmse metric.
i Resamples are analyzed in a random order.
i Fold10:  7 eliminated; 13 candidates remain.
i Fold06:  8 eliminated;  5 candidates remain.
i Fold08:  1 eliminated;  4 candidates remain.
i Fold01:  0 eliminated;  4 candidates remain.
i Fold04:  1 eliminated;  3 candidates remain.
i Fold02:  1 eliminated;  2 candidates remain.
i Fold09:  0 eliminated;  2 candidates remain.
```

```
xgb_game_rs
```

```
# A tibble: 10 x 5
   splits               id     .order .metrics          .notes
   <list>               <chr>   <int> <list>            <list>
 1 <split [14407/1602]> Fold03      1 <tibble [40 x 7]> <tibble [0 x 1]>
 2 <split [14408/1601]> Fold05      2 <tibble [40 x 7]> <tibble [0 x 1]>
 3 <split [14410/1599]> Fold10      3 <tibble [40 x 7]> <tibble [0 x 1]>
 4 <split [14408/1601]> Fold06      4 <tibble [26 x 7]> <tibble [0 x 1]>
 5 <split [14408/1601]> Fold08      5 <tibble [10 x 7]> <tibble [0 x 1]>
 6 <split [14407/1602]> Fold01      6 <tibble [8 x 7]>  <tibble [0 x 1]>
 7 <split [14408/1601]> Fold04      7 <tibble [8 x 7]>  <tibble [0 x 1]>
 8 <split [14407/1602]> Fold02      8 <tibble [6 x 7]>  <tibble [0 x 1]>
 9 <split [14410/1599]> Fold09      9 <tibble [4 x 7]>  <tibble [0 x 1]>
10 <split [14408/1601]> Fold07     10 <tibble [4 x 7]>  <tibble [0 x 1]>
```

Evaluate Model

```
plot_race(xgb_game_rs)
```




```
show_best(xgb_game_rs)
```

```
Warning: No value of `metric` was given; metric 'rmse' will be used.
```

```
# A tibble: 2 x 9
   mtry trees min_n .metric .estimator  mean     n std_err .config
  <int> <int> <int> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
1    14  1709    17 rmse    standard   0.735    10 0.00549 Preprocessor1_Model08
2    25  1941    22 rmse    standard   0.735    10 0.00531 Preprocessor1_Model14
```

```
xgb_last <-
  xgb_wf %>%
  finalize_workflow(select_best(xgb_game_rs, "rmse")) %>%
  last_fit(game_split)

xgb_last
```

```
# A tibble: 1 x 6
  splits               id              .metrics .notes   .predictions .workflow
  <list>               <chr>           <list>   <list>   <list>       <list>
1 <split [16009/5339]> train/test spl~ <tibble> <tibble> <tibble>     <workflow>
```

```
xgb_last %>% collect_metrics()
```

```
# A tibble: 2 x 4
  .metric .estimator .estimate .config
  <chr>   <chr>          <dbl> <chr>
1 rmse    standard       0.744 Preprocessor1_Model1
2 rsq     standard       0.362 Preprocessor1_Model1
```

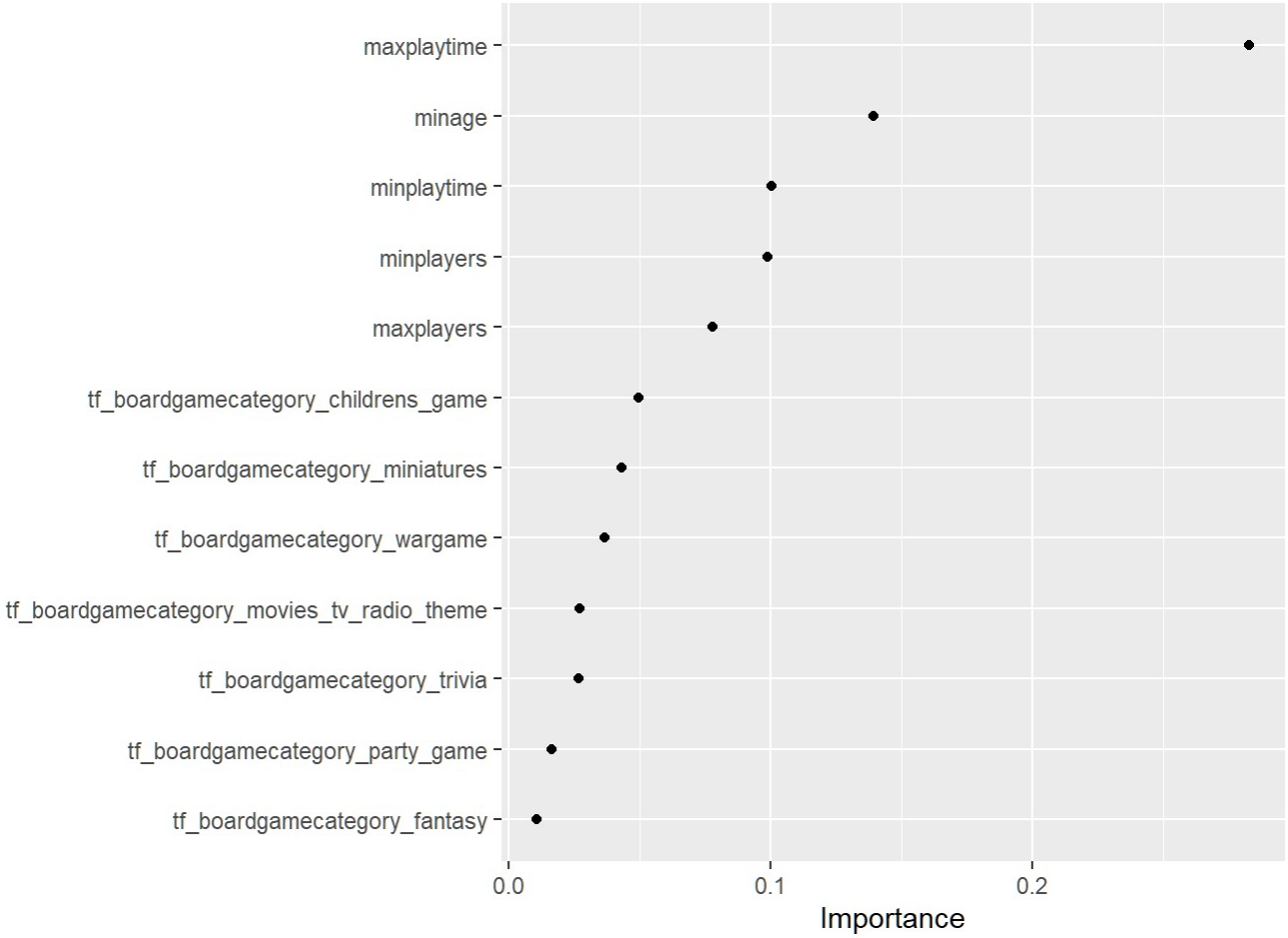
Examine model variable importance by looking at only model parameters:

```
library(vip)
```

```
Attaching package: 'vip'
```

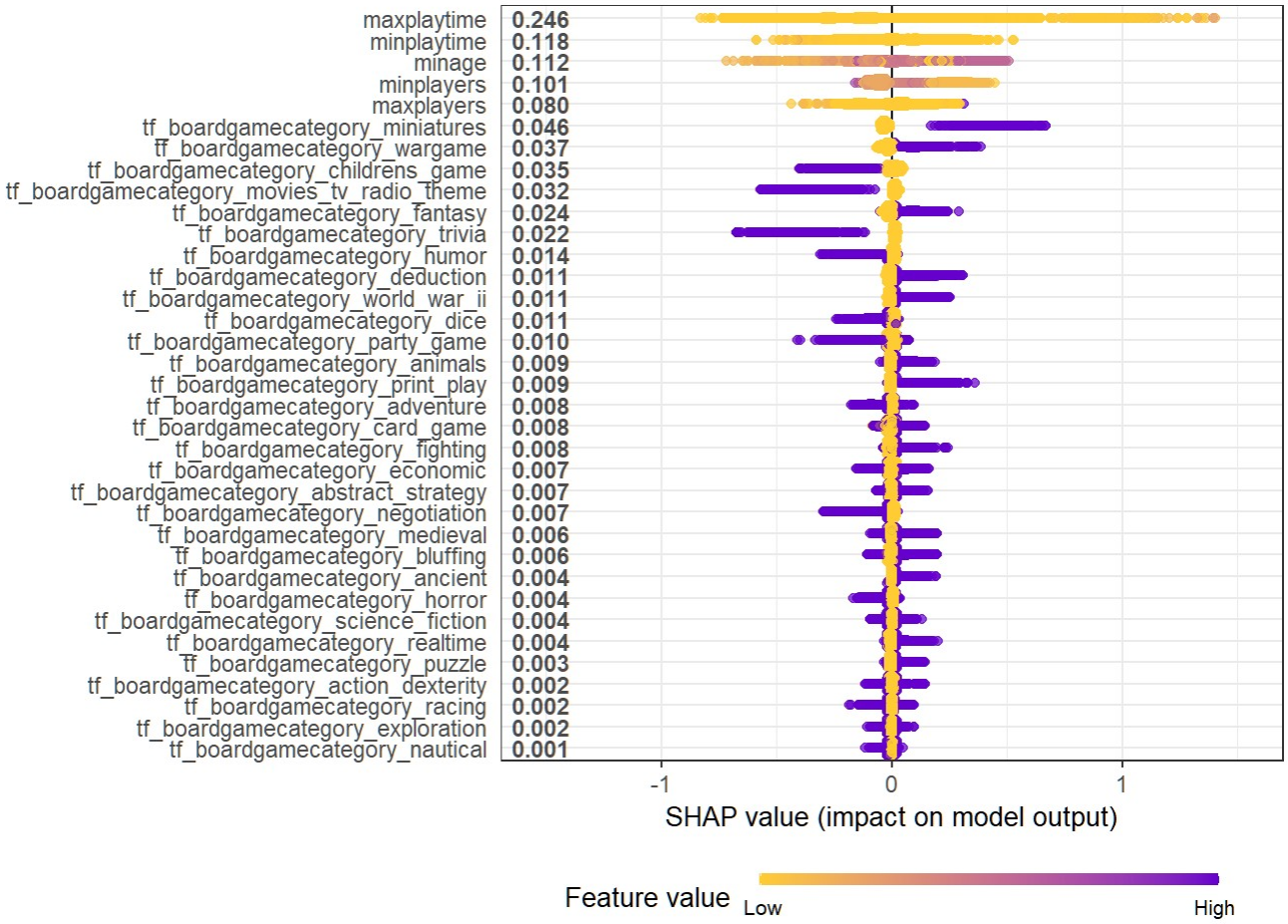
```
The following object is masked from 'package:utils':

    vi
```

```
xgb_fit <- extract_fit_parsnip(xgb_last)

vip(xgb_fit, geom = "point", num_features = 12)
```

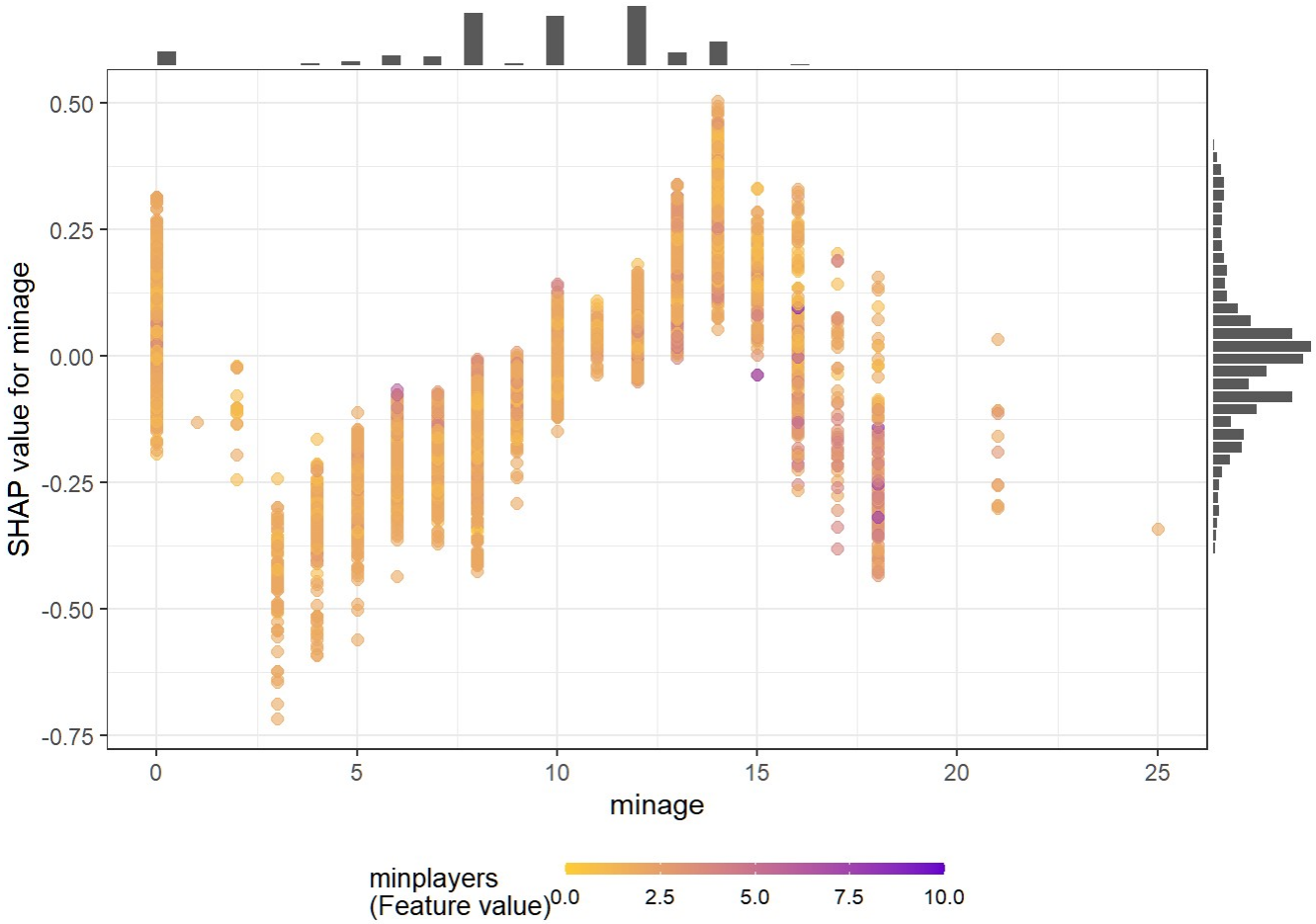


Lastly, another library allows for quick analysis of the SHAP dependence:

```
library(SHAPforxgboost)

game_shap <-
  shap.prep(
  xgb_model = extract_fit_engine(xgb_fit),
  X_train = bake(game_prep,
                 has_role("predictor"),
                 new_data = NULL,
                 composition = "matrix")
  )

shap.plot.summary(game_shap)
```




```
shap_plot <- shap.plot.dependence(
  game_shap,
  x = "minage",
  color_feature = "minplayers",
  size0 = 2,
  smooth = FALSE, add_hist = TRUE
)

shap_plot
```


0.50
0.25
0.00
-0.25
-0.50
-0.75
SHAP value for minage
0
5
10
15
20
25
minage
minplayers
(Feature value)
0.0
2.5
5.0
7.5
10.0