

# Statistical Analysis of “Hidden Gem” Score for Movies on Netflix

Zhizhi Wei, Samantha Zhang, Cindy He, Yutian Gu

22/12/2021

```
library(tidyverse)
library(here)
FlixGem <- read_csv(here("/Users/zhizhiwei/Downloads/Final_Project_FlixGem.csv"))
```

```
class(FlixGem)
```

```
## [1] "spec_tbl_df" "tbl_df"      "tbl"        "data.frame"
```

```
# subsetted and cleaned data:
```

```
FlixGem_cleaned <- FlixGem %>% filter(`Series or Movie` == "Movie") %>% drop_na()
FlixGem_cleaned
```

```
## # A tibble: 2,118 x 29
```

```
##   Title      Genre Tags Languages `Series or Movie` `Hidden Gem Score`
##   <chr>      <chr> <chr> <chr>      <chr>              <dbl>
## 1 Joker      Crim~ Dark~ English Movie              3.5
## 2 I          Acti~ Dram~ English,~ Movie              2.8
## 3 Harrys Daughters Adve~ Dram~ English Movie              4.4
## 4 The Closet Come~ Kore~ French Movie              3.8
## 5 Ordinary People Drama Kore~ English Movie              4.2
## 6 Stand by Me Adve~ Kore~ English Movie              4.1
## 7 Wonderstruck Adve~ Chil~ English,~ Movie              3.6
## 8 The Girl on the T~ Crim~ Boll~ English,~ Movie              2.6
## 9 Red        Acti~ Dram~ English,~ Movie              3.4
## 10 Burden     Drama Movi~ English Movie              7.8
```

```
## # i 2,108 more rows
```

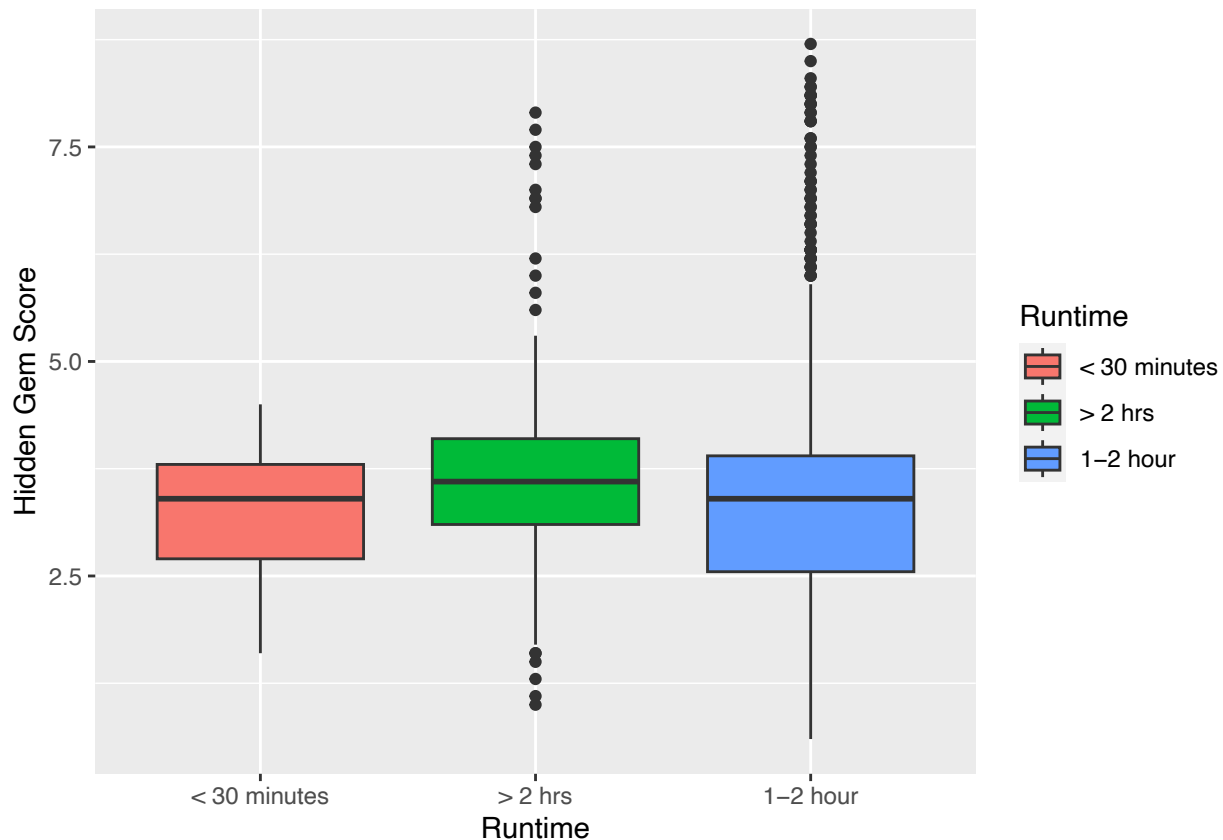
```
## # i 23 more variables: `Country Availability` <chr>, Runtime <chr>,
## #   Director <chr>, Writer <chr>, Actors <chr>, `View Rating` <chr>,
## #   `IMDb Score` <dbl>, `Rotten Tomatoes Score` <dbl>,
## #   `Metacritic Score` <dbl>, `Awards Received` <dbl>,
## #   `Awards Nominated For` <dbl>, Boxoffice <dbl>, `Release Date` <dtm>,
## #   `Netflix Release Date` <dtm>, `Production House` <chr>, ...
```

## Task 1

(a)

Hidden Gem Score vs. Runtime:

```
ggplot(FlixGem_cleaned, aes(x = Runtime, y = `Hidden Gem Score`, group = Runtime,
  fill = Runtime)) + geom_boxplot() + labs(x = "Runtime", y = "Hidden Gem Score")
```



We can see from the boxplot that the Hidden Gem Score mean of movies of different Runtime are very close, but runtime of >2 hours tend to have a higher score of 75% quartile, while runtime of less than 30 minutes tend to have a lower one. Also the outliers of 1-2 hours movies tend to lie in the range of higher scores. Thus we can conclude that Hidden Gem Scores associates with Runtime.

## Hidden Gem Score vs. languages:

### Step1: Re-coding 'Languages'

```
FlixGem_cleaned %>% pull(Languages) %>% unique(.) %>% sort(.)
```

```
FlixGem_cleaned %>% group_by(Languages) %>% summarize(count=n()) %>% arrange(desc(count))
```

```
FlixGem_lang_main <- FlixGem_cleaned %>%
  mutate(Languages_main = word(FlixGem_cleaned$Languages, 1)) %>%
  mutate(Languages_main_alt = recode(Languages_main, `English`,` = "English", `French`,` = "French",
    `Spanish`,` = "Spanish", `Korean`,` = "Korean", `Mandarin`,` = "Mandarin", `Italian`,` = "Italian",
    `Cantonese`,` = "Cantonese", `German`,` = "German", `Japanese`,` = "Japanese",
    `Hindi`,` = "Hindi", `Danish`,` = "Danish", `Norwegian`,` = "Norwegian", `Swedish`,` = "Swedish",
```

```

`Czech`,` = "Czech", `Polish`,` = "Polish", `Russian`,` = "Russian", `Zulu`,` = "Zulu",
`Arabic`,` = "Arabic", `Chinese`,` = "Mandarin", `Dari`,` = "Dari", `Filipino`,` = "Filipino",
`Greek`,` = "Greek", `Hungarian`,` = "Hungarian", `Indonesian`,` = "Indonesian",
`Neapolitan`,` = "Neapolitan", `None`,` = "None", `Persian`,` = "Persian", `Portuguese`,` = "Portuguese",
`Romany`,` = "Romany", `Tamil`,` = "Tamil", `Thai`,` = "Thai", `Tswana`,` = "Tswana",
`Yiddish`,` = "Yiddish"))
FlixGem_lang_main %>%
  group_by(Languages_main_alt) %>%
  summarize(count = n()) %>%
  arrange(desc(count))

```

```

## # A tibble: 36 x 2
##   Languages_main_alt count
##   <chr>             <int>
## 1 English           1891
## 2 French             34
## 3 Spanish            29
## 4 Japanese           27
## 5 Mandarin           20
## 6 Korean             17
## 7 German             14
## 8 Italian            14
## 9 Cantonese           9
## 10 Hindi              7
## # i 26 more rows

```

```

# Filter out Languages with counts > 5
FlixGem_lang_main_filter <- FlixGem_lang_main %>%
  filter(Languages_main_alt %in% c("English", "French", "Spanish", "Japanese",
    "Mandarin", "Korean", "German", "Italian", "Cantonese", "Hindi", "Swedish",
    "Danish", "Norwegian", "Portuguese"))
FlixGem_lang_main_filter %>%
  group_by(Languages_main_alt) %>%
  summarize(count = n()) %>%
  arrange(desc(count))

```

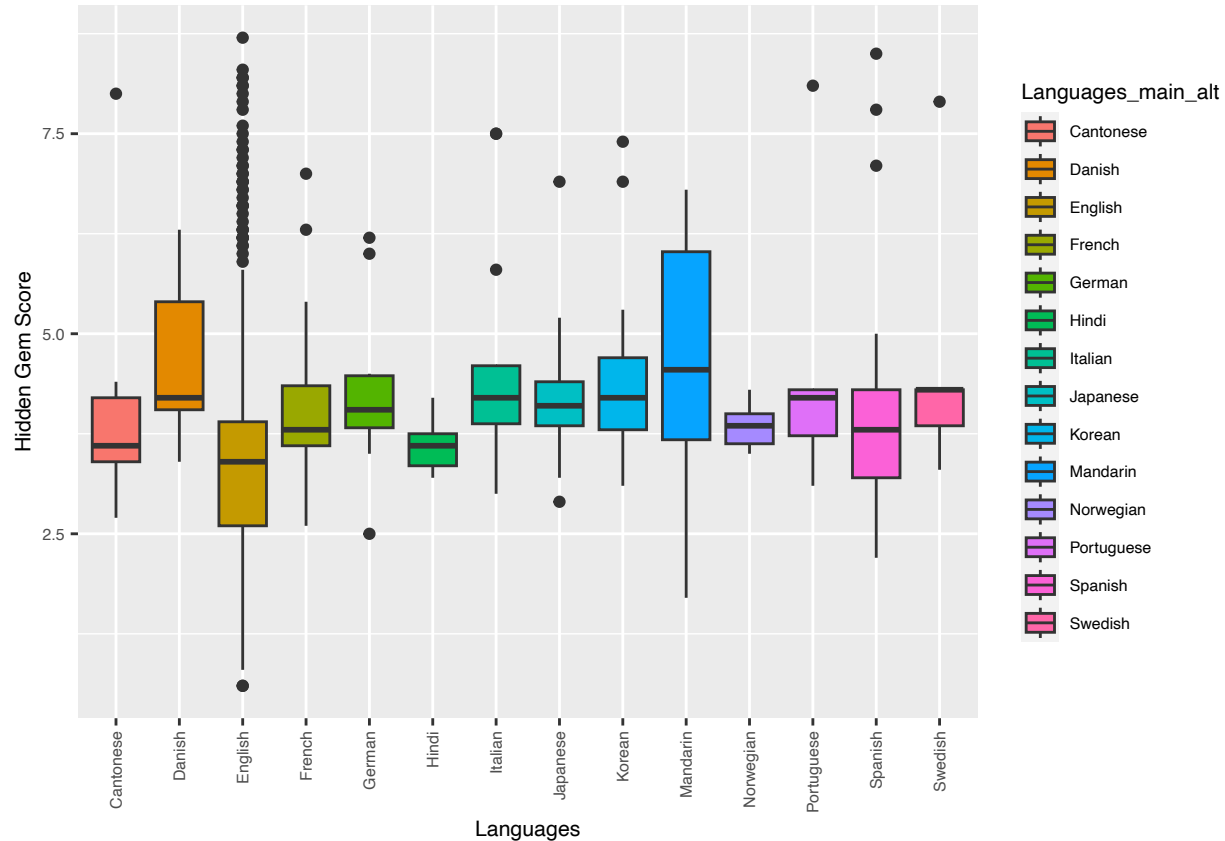
```

## # A tibble: 14 x 2
##   Languages_main_alt count
##   <chr>             <int>
## 1 English           1891
## 2 French             34
## 3 Spanish            29
## 4 Japanese           27
## 5 Mandarin           20
## 6 Korean             17
## 7 German             14
## 8 Italian            14
## 9 Cantonese           9
## 10 Hindi              7
## 11 Swedish            7
## 12 Danish             6
## 13 Norwegian          6
## 14 Portuguese         6

```

## Step2: Plotting ‘Hidden Gem Score vs. languages’

```
ggplot(FlixGem_lang_main_filter, aes(x = Languages_main_alt, y = `Hidden Gem Score`,
  group = Languages_main_alt, fill = Languages_main_alt)) + geom_boxplot() + labs(x = "Languages",
  y = "Hidden Gem Score") + theme(legend.key.size = unit(0.5, "cm"), axis.text.x = element_text(angle
  vjust = 0.5, hjust = 1), text = element_text(size = 8))
```

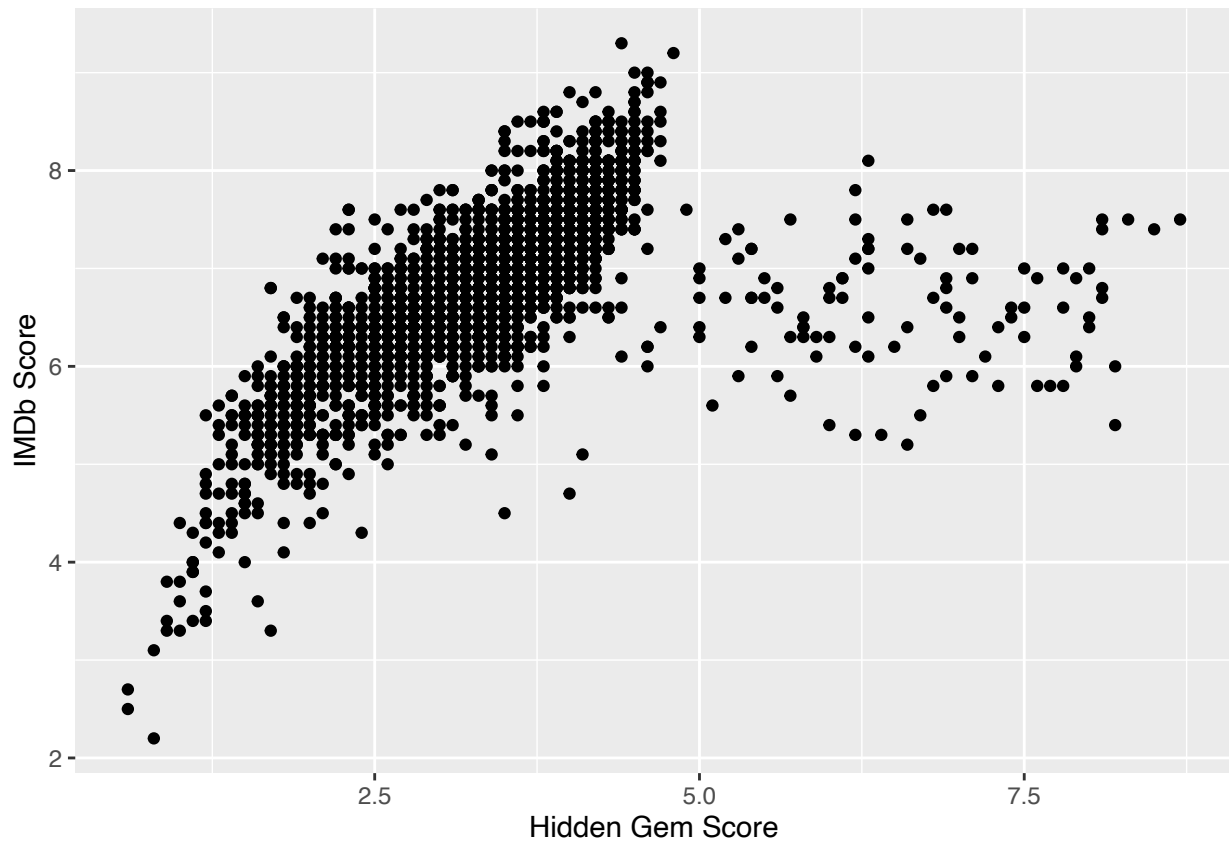


Moreover, the Hidden Gem Score seems not to have association with the language of movies. We can see from the boxplots that the mean, 25th quartile and 75th quartile range of movies of each languages are very similar, except for English and Mandarin. But these two languages' score varies widely because they have much more content and market compared to other languages. So the nature of these outliers is due to those movies' quality, not the factor of language at surface. Movies of large numbers all have different quality themselves, while audience who understand this kind of language in different market have various comments depending on cultural backgrounds etc. Thus, we can conclude that the Hidden Gem Score does not have association with the language.

(b)

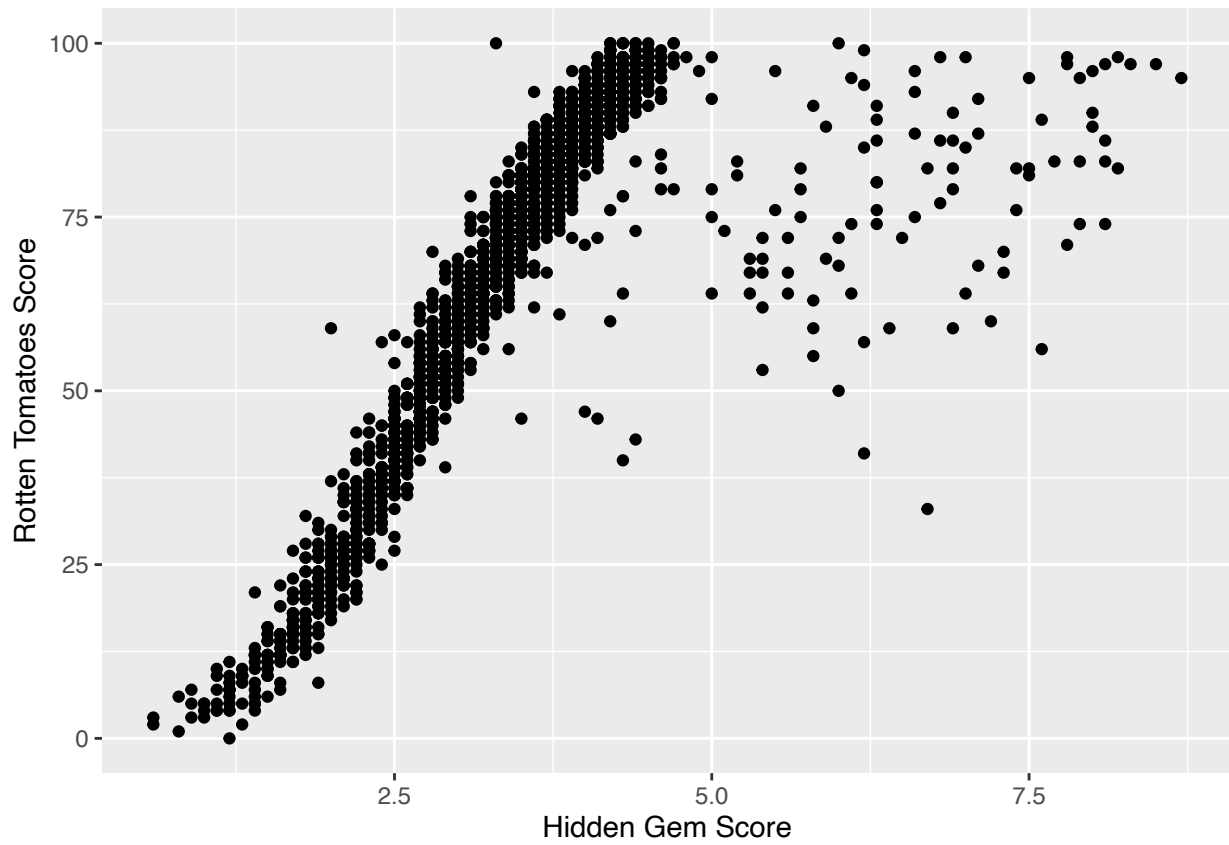
## Hidden Gem Scores vs. IMDb

```
ggplot(FlixGem_cleaned, aes(x = `Hidden Gem Score`, y = `IMDb Score`)) + geom_point() +
  labs(x = "Hidden Gem Score", y = "IMDb Score")
```



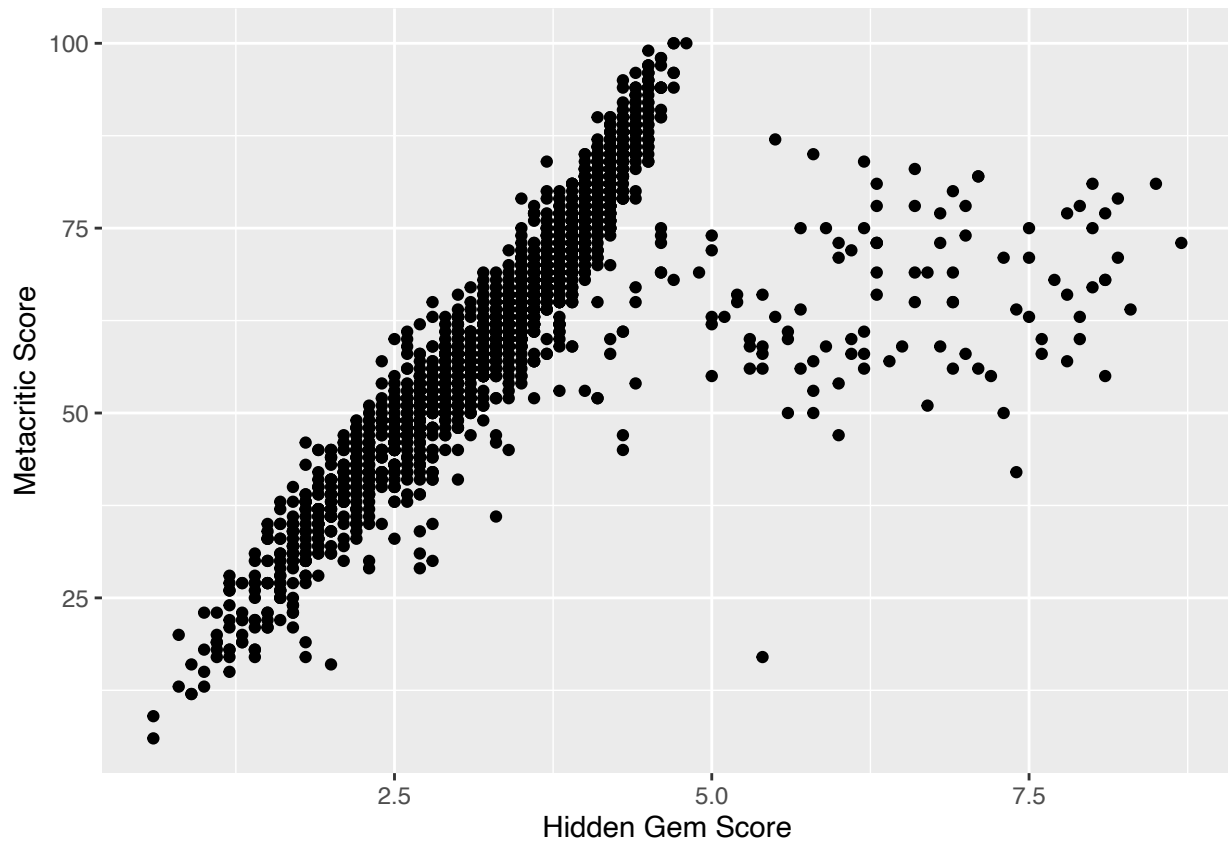
Hidden Gem Scores vs. Rotten Tomatoes Score

```
ggplot(FlixGem_cleaned, aes(x = `Hidden Gem Score`, y = `Rotten Tomatoes Score`)) +  
  geom_point() + labs(x = "Hidden Gem Score", y = "Rotten Tomatoes Score")
```



Hidden Gem Scores vs. Metacritic Score

```
ggplot(FlixGem_cleaned, aes(x = `Hidden Gem Score`, y = `Metacritic Score`)) + geom_point() +  
  labs(x = "Hidden Gem Score", y = "Metacritic Score")
```



The three reviews strongly correlate to the Hidden Gem Score from the three plots. We can see that the dots all form a linear relationship between each review and the Hidden Gem Score, indicating that as the Hidden Gem Score increases, the review increases along with it. By nature, they all reflect the general quality of a movie and they make relatively fair comments. The quality of a movie is consistent.

(c)

```
library(lubridate)
```

```
Release_Date_year <- rep(0, length(FlixGem_cleaned$`Release Date`))
```

```
for (i in seq_along(FlixGem_cleaned$`Release Date`)) {
  Release_Date_year[i] = year(as.POSIXlt(FlixGem_cleaned$`Release Date`[i], format = "%Y-%m-%d"))
}
```

```
FlixGem_cleaned <- FlixGem_cleaned %>%
  mutate(Release_Date_year = Release_Date_year)
```

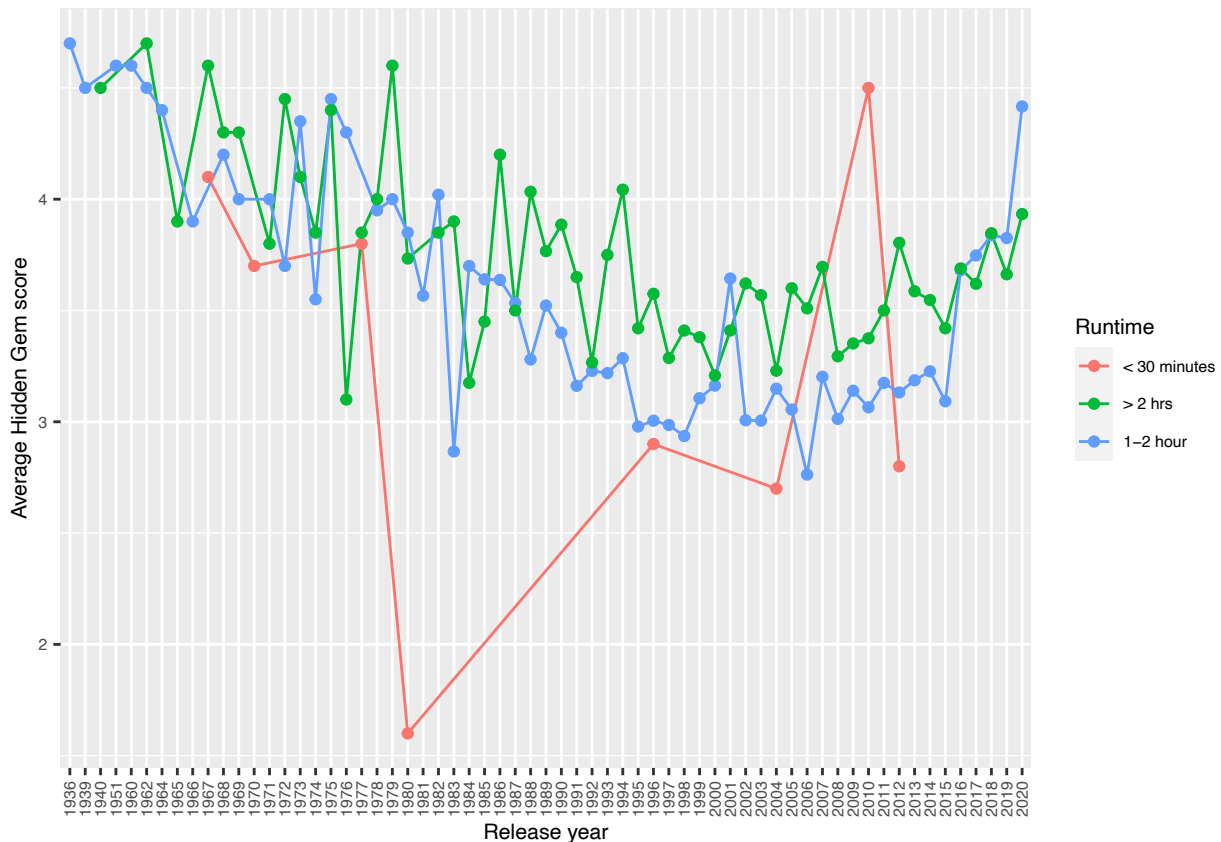
```
Avg_HG_by_year_runtime <- FlixGem_cleaned %>%
  group_by(Runtime, Release_Date_year) %>%
  summarise(Avg_HG_score = mean(`Hidden Gem Score`))
```

```
## 'summarise()' has grouped output by 'Runtime'. You can override using the
## '.groups' argument.
```

```
Avg_HG_by_year_runtime
```

```
## # A tibble: 121 x 3
## # Groups:   Runtime [3]
##   Runtime Release_Date_year Avg_HG_score
##   <chr>         <dbl>         <dbl>
## 1 1-2 hour      1936           4.7
## 2 1-2 hour      1939           4.5
## 3 1-2 hour      1951           4.6
## 4 1-2 hour      1960           4.6
## 5 1-2 hour      1962           4.5
## 6 1-2 hour      1964           4.4
## 7 1-2 hour      1966           3.9
## 8 1-2 hour      1968           4.2
## 9 1-2 hour      1969           4
## 10 1-2 hour     1971           4
## # i 111 more rows
```

```
ggplot(Avg_HG_by_year_runtime, aes(x = factor(Release_Date_year), y = Avg_HG_score,
  group = factor(Runtime), col = factor(Runtime))) + geom_line() + geom_point() +
  labs(x = "Release year", y = "Average Hidden Gem score", col = "Runtime") + theme(legend.key.size =
    "cm"), axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1), text = element_text(size = 8)
```



From the plots we can see that the average Hidden Gem Score of movies of 1-2 hours and >2 hours keep changing on a smaller range, while that of movies of <30 minutes changes dramatically especially during 1977-1997 (although it might be resulted from the lack of data). Movie >2 hours's average Score generally



remain as a leading one in a long term. Since 2015, people seem to prefer longer movies. Longer movies' Score are increasing and 1-2 hours movies' Score increase faster than that of >2 hours movies. Thus it seems like people are more accepting to longer movies.

## Task 2:

```
tinytex::install_tinytex()
library(rsample)
library(tidyverse)
```

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

```
## v ggplot2 3.3.5    v purrr  0.3.4
## v tibble  3.1.4    v dplyr  1.0.7
## v tidyr   1.1.4    v stringr 1.4.0
## v readr   2.0.2    v forcats 0.5.1
```

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

```
library(rpart)
library(rpart.plot)
library(Metrics)
```

```
FlixGem<-read.csv("/Users/samanthazhang/Final_Project_FlixGem.csv",header=TRUE)
```

```
FlixGem_cleaned <- FlixGem[!(FlixGem$Series.or.Movie=="Series"),]
FlixGem_cleaned <- FlixGem_cleaned %>% na.omit(FlixGem_cleaned)
```

```
set.seed(123)
FlixGem_Movie_Selected <- FlixGem_cleaned %>% select(Hidden.Gem.Score, Languages, Runtime, IMDb.Score, Rot)
flix_split <- initial_split(FlixGem_Movie_Selected, prop = .8)
flix_train <- training(flix_split)
flix_test <- testing(flix_split)
```

I divided the data set into 80% vs 20% for training data and testing data respectively.

```
Gem_Score_Tree_1 <-rpart(
  formula = Hidden.Gem.Score ~.,
  data = FlixGem_Movie_Selected,
  method = "anova",
)

Gem_Score_Tree_1
```

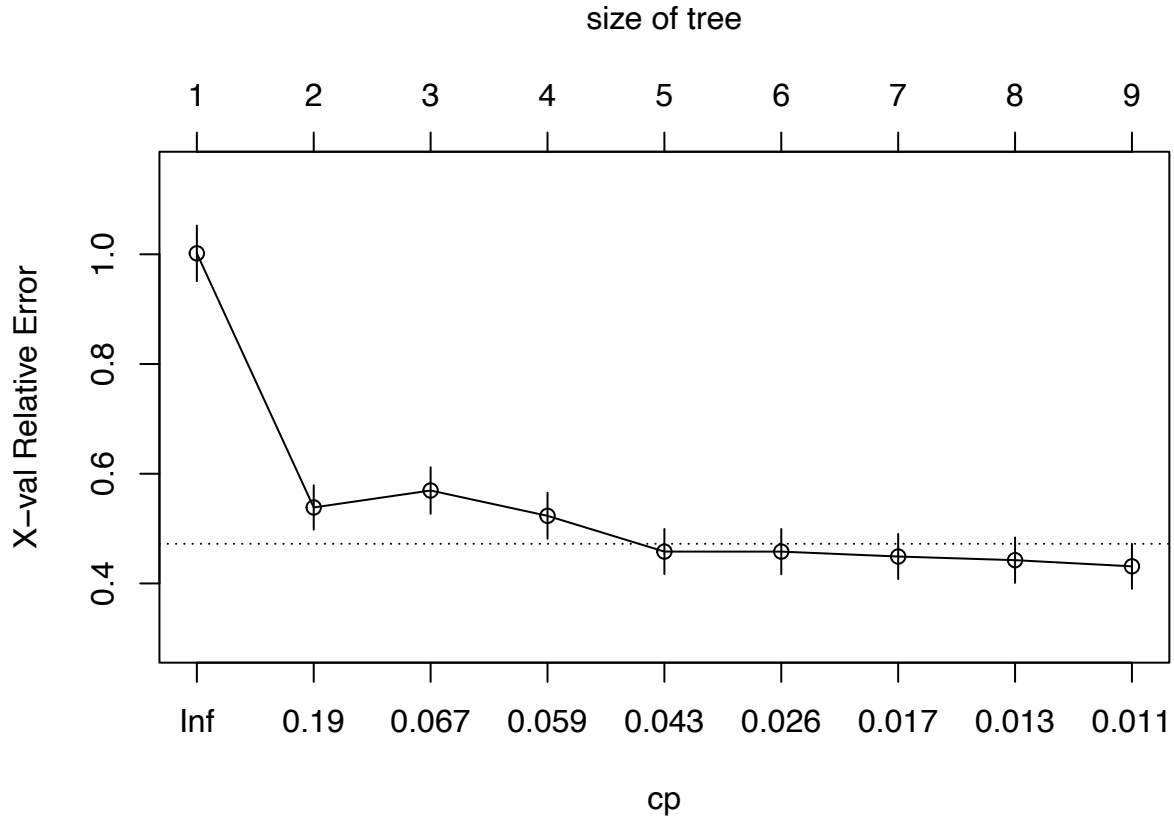
```
## n= 2118
##
## node), split, n, deviance, yval
##      * denotes terminal node
##
## 1) root 2118 2548.053000 3.396412
##    2) Rotten.Tomatoes.Score< 58.5 716 314.626400 2.347765
##      4) Rotten.Tomatoes.Score< 32.5 299 40.345620 1.809030 *
##      5) Rotten.Tomatoes.Score>=32.5 417 125.276500 2.734053
##        10) Languages=Cantonese, Mandarin,English,English, Aboriginal, French, Portuguese,English, Afr
##        11) Languages=English, Catalan, Spanish,English, Hindi, Urdu,German,Italian, French, English,K
##    3) Rotten.Tomatoes.Score>=58.5 1402 1043.968000 3.931954
##      6) Languages=American Sign Language, English,Arabic,Cantonese,Cantonese, English,Cantonese, Eng
##      12) Rotten.Tomatoes.Score< 85.5 800 304.839700 3.569375
##        24) Rotten.Tomatoes.Score< 71.5 292 113.280000 3.300000 *
##        25) Rotten.Tomatoes.Score>=71.5 508 158.192200 3.724213 *
##      13) Rotten.Tomatoes.Score>=85.5 525 238.012400 4.256190 *
##      7) Languages=Arabic, English,Cantonese, English, Hokkien, Mandarin,Czech, English, German, Russ
##        14) Languages=Czech, German, English, Slovak,Danish,Danish, English,Dutch,English, Arabic, Tur
##        28) IMDb.Score>=7.05 39 20.055900 4.543590 *
##        29) IMDb.Score< 7.05 26 43.766540 6.011538 *
##      15) Languages=Arabic, English,Cantonese, English, Hokkien, Mandarin,Czech, English, German, Ru
```

Look for the 4th branch in Gem\_Score\_Tree\_1, you can see that 299 observations and the SSE in this region is 40.345620. Their average Hidden Gem Scores are 1.809030. The largest reduction in SSE initially is Rotten Tomatoes Score with scores under 32.5.

```
rpart.plot(Gem_Score_Tree_1)
```

```
## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```





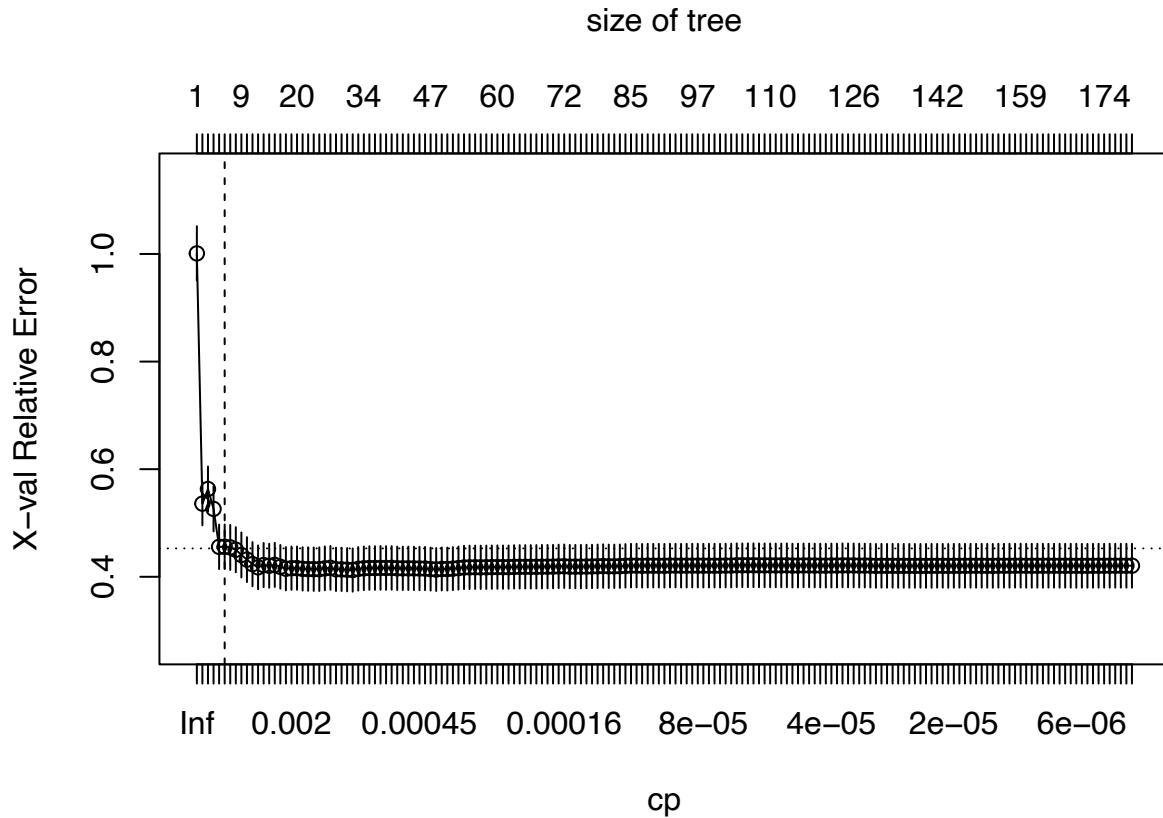
```
Gem_Score_Tree_1$cptable
```

```
##          CP nsplit rel error   xerror   xstd
## 1 0.46681054      0 1.0000000 1.0017931 0.05067076
## 2 0.07745208      1 0.5331895 0.5385332 0.04042646
## 3 0.05868203      2 0.4557374 0.5694724 0.04236183
## 4 0.05847774      3 0.3970554 0.5232722 0.04192998
## 5 0.03219573      4 0.3385776 0.4580775 0.04120398
## 6 0.02092520      5 0.3063819 0.4580370 0.04125149
## 7 0.01319283      6 0.2854567 0.4490885 0.04117956
## 8 0.01309530      7 0.2722639 0.4424352 0.04136893
## 9 0.01000000      8 0.2591686 0.4311886 0.04110606
```

After 6 terminal nodes in the plot 1 of cross-validation error results, there are diminishing returns in error reduction as the tree grows deeper. We also achieve minimal expected error by looking at the cross-validation error of 0.4311886.

```
Gem_Score_Tree_2 <-rpart(
  formula = Hidden.Gem.Score ~.,
  data = FlixGem_Movie_Selected,
  method = "anova",
  control = list(cp=0,xval = 10)
)
```

```
plotcp(Gem_Score_Tree_2)
abline(v=6, lty = "dashed")
```



After 6 terminal nodes in the plot 2 with no penalty results in a fully grown tree, we also seeing diminishing returns in error reduction as the tree grows.

```
pred <- predict(Gem_Score_Tree_1, testdata = flix_test)
actual <- flix_test$Hidden.Gem.Score
rmse(actual, pred)
```

```
## Warning in actual - predicted: longer object length is not a multiple of shorter
## object length
```

```
## [1] 1.462105
```

Rotten Tomatoes Score is the most important feature for predicting the Hidden Gem Score. There has the largest reduction in SSE with Rotten Tomatoes Score scores under 32.5. They have almost half the average Hidden Gem scores. We can see that after 6 terminal nodes both in the plot 1 and the plot 2 of cross-validation error results, there are diminishing returns in error reduction. We achieve minimal expected error by performing this prediction. On average, our predicted HiddenGem Scores is about 1.462105 off from the actual Hidden Gem Scores. Our prediction works well.

```
library(tidyverse)
library(here)
FlixGem <- read_csv(here("/Users/zhizhiwei/Downloads/Final_Project_FlixGem.csv"))
```

### Task 3:

Clean data again (only drop NAs for directors, Hidden gem score and title):

```
FlixGem_cleaned_task3 <- FlixGem %>%
  filter(`Series or Movie` == "Movie") %>%
  drop_na(Director, `Hidden Gem Score`, Title)
```

Select Directors, Movie title and HG-score out:

```
FlixGem_cleaned_HG <- FlixGem_cleaned_task3 %>%
  group_by(Director, Title) %>%
  select(Director, Title, `Hidden Gem Score`) %>%
  arrange(Director)
```

Split a list for each director:

```
FlixGem_cleaned_HG_split <- with(FlixGem_cleaned_HG, split(FlixGem_cleaned_HG, Director))
```

```
length(FlixGem_cleaned_HG_split)
```

```
## [1] 4097
```

Function that takes in a director's list and return his/her HG-H index:

```
calculate_HG_H_for_sapply <- function(each_director_list){
  H <- 0
  for(i in seq_along(each_director_list$`Hidden Gem Score`)){
    if(each_director_list$`Hidden Gem Score`[i]>=i){
      H<-H+1
    }
  }
  return(H)
}
```

Apply the function to the splitted list and returns the vector containing HG-H scores for all directors:

```
HG_H_index_vec <-sapply(FlixGem_cleaned_HG_split, calculate_HG_H_for_sapply)
```

Convert the vector to a tibble:

```
all_HG_H_index <-as.tibble(HG_H_index_vec)
```

```
## Warning: 'as.tibble()' was deprecated in tibble 2.0.0.
## i Please use 'as_tibble()' instead.
## i The signature and semantics have changed, see '?as_tibble'.
## This warning is displayed once every 8 hours.
## Call 'lifecycle::last_lifecycle_warnings()' to see where this warning was
## generated.
```

Combine the Director names and their HG-H scores into one tibble:

```
Director_name_tibble <- FlixGem_cleaned_HG[,1] %>% unique(.)
combined_name_H_index <- bind_cols(Director_name_tibble, all_HG_H_index)
```

Find the top 10 directors according to their HG-H index:

```
top_10 <- combined_name_H_index %>% arrange(desc(value)) %>% head(10)
colnames(top_10) <- c("Director", "HG-H-index")
top_10
```

```
## # A tibble: 10 x 2
## # Groups:   Director [10]
##   Director                                'HG-H-index'
##   <chr>                                <dbl>
## 1 Chan-wook Park                        8
## 2 Rebecca Johnson                      8
## 3 Sara Dosa, Barbara Kopple            8
## 4 Uli Edel                            8
## 5 Darren Lynn Bousman                  7
## 6 Jayaprakash Radhakrishnan            7
## 7 Je-gu Yun                            7
## 8 Jirí Havelka                         7
## 9 Jon Garaño, Aitor Arregi, Jose Mari Goenaga 7
## 10 Lars Kraume                         7
```