Final Project

MATH 208

Seif Abdelkefi, Yesmine Abdelkefi, Alexandra Demos, Alex DePani

Part I: Data wrangling and exploratory data analysis

```
Final_Project_FlixGem = read_csv(file = "Final_Project_FlixGem.csv")
```

```
## Rows: 9425 Columns: 29
```

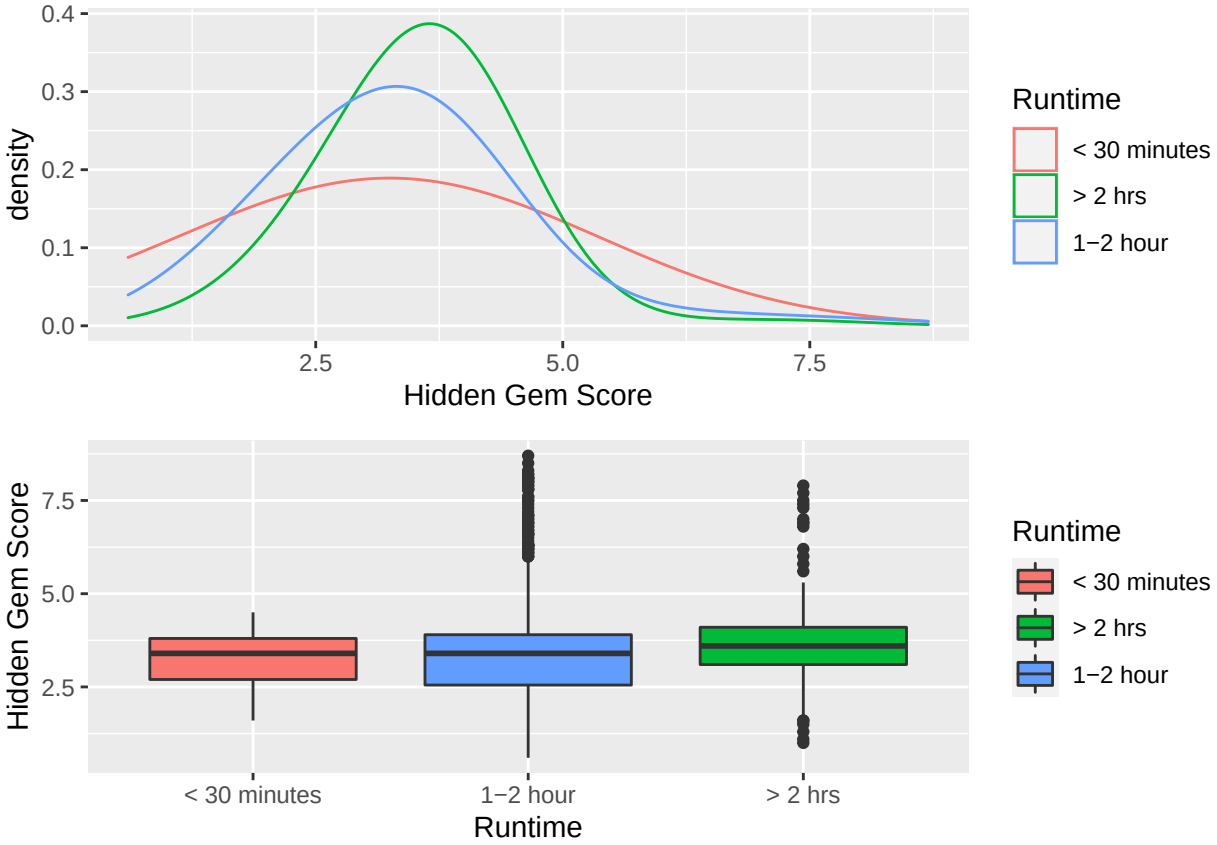
```
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (19): Title, Genre, Tags, Languages, Series or Movie, Country Availabil...
## dbl   (8): Hidden Gem Score, IMDb Score, Rotten Tomatoes Score, Metacritic S...
## dttm  (2): Release Date, Netflix Release Date
```

```
##
## 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.
```

```
Final_Project_FlixGem<-na.omit(Final_Project_FlixGem%>%filter(`Series or
↪  Movie`=="Movie"))

Final_Project_FlixGem_final = Final_Project_FlixGem
```

a) Is there any association between: Hidden gem score and runtime, Hidden gem score and languages?

```
run_box = ggplot(Final_Project_FlixGem,aes(x=reorder(Runtime,`Hidden Gem
↪  Score`),y=`Hidden Gem Score`,fill=Runtime))+geom_boxplot()+xlab("Runtime")

run_dens = ggplot()+
  geom_density(data=Final_Project_FlixGem,aes(x=`Hidden Gem Score`, colour =
  ↪  Runtime),adjust=4)

grid.arrange(run_dens, run_box)
```



Looking at our data, we can claim that the distribution of Hidden Gem Scores does not depend heavily on the Runtime. Longer movies, those that are more than two hours, seem to have on average a higher score than other movies, as well as less deviation from the mean. It would be hard to claim anything about short movies (less than 30 minutes) as we only have a very small sample (9 movies), which might not be representative of how well rated short movies are in general.

Is there an association between: Hidden gem score and language?

We will analyze the Hidden Gem Score distributions of movies available exclusively in the top 3 North American languages (English only, French only and Spanish only) as well as the distribution of Hidden Gem Scores of movies available in only one foreign language (thus movies available in one language that isn't English, French or Spanish). Finally, we will be taking a look at the distribution of movies available in multiple languages.

Does one category seem to fare better than the others?

```
English_or_other<-Final_Project_FlixGem%>%
  filter(grepl('English',Languages))%>%
  filter(!grepl('French|Spanish',Languages))
French_or_other<-Final_Project_FlixGem%>%
  filter(grepl('French',Languages))%>%
  filter(!grepl('English|Spanish',Languages))
Spanish_or_other<-Final_Project_FlixGem %>%
  filter(grepl('Spanish',Languages))%>%filter(!grepl('English|French',Languages))
One_Foreign_Language_Movies<-Final_Project_FlixGem%>%
  filter(!grepl(',',Languages))%>%
  filter(!grepl('English|French|Spanish',Languages))
```

```
Multi_Languages_Movies<-Final_Project_FlixGem %>%
  filter(grepl(",",Languages, fixed = TRUE))


English_or_other$Languages="English and Foreign"
French_or_other$Languages="French and Foreign"
Spanish_or_other$Languages="Spanish and Foreign"
Multi_Languages_Movies$Languages="Multiple Languages"
One_Foreign_Language_Movies$Languages="One Foreign Language"




Final_Project_FlixGem<-rbind(English_or_other, French_or_other, Spanish_or_other
↪  ,One_Foreign_Language_Movies, Multi_Languages_Movies)
```

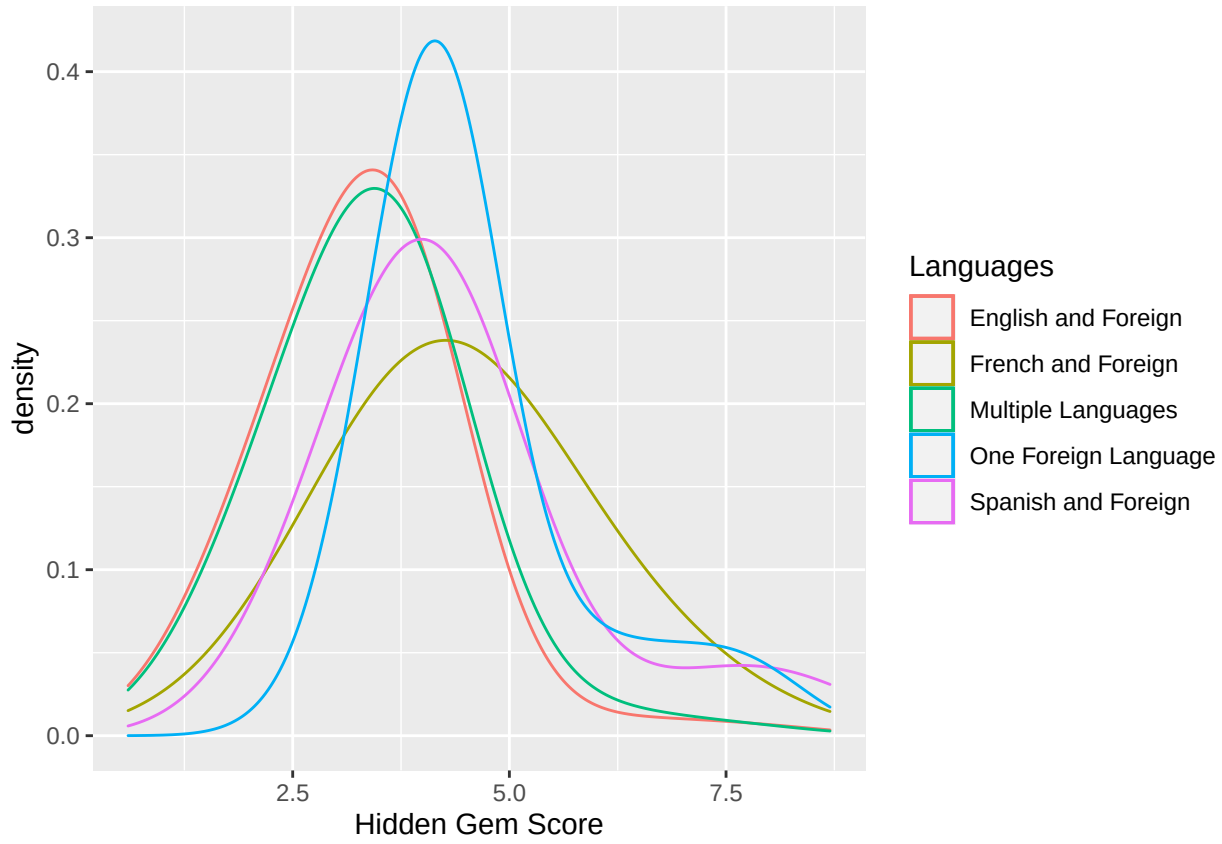
Now, let's get the summary statistic for all these subgroups.

```
Distribution_English_or_other<-English_or_other%>%summarise(mean(`Hidden Gem
↪  Score`),median(`Hidden Gem Score`),sd(`Hidden Gem Score`))
colnames(Distribution_English_or_other)<-c("Mean_score","Median_score","SD_score")

Distribution_French_or_other<-French_or_other%>%summarise(mean(`Hidden Gem
↪  Score`),median(`Hidden Gem Score`),sd(`Hidden Gem Score`))
colnames(Distribution_French_or_other)<-c("Mean_score","Median_score","SD_score")

Distribution_Spanish_or_other<-Spanish_or_other%>%summarise(mean(`Hidden Gem
↪  Score`),median(`Hidden Gem Score`),sd(`Hidden Gem Score`))
colnames(Distribution_Spanish_or_other)<-c("Mean_score","Median_score","SD_score")

Distribution_One_Foreign_Language_Movies<-One_Foreign_Language_Movies%>%summarise(mean(`Hidden
↪  Gem Score`),median(`Hidden Gem Score`),sd(`Hidden Gem Score`))
colnames(Distribution_One_Foreign_Language_Movies)<-c("Mean_score","Median_score","SD_score")

Distribution_Multi_Languages_Movies<-Multi_Languages_Movies%>%summarise(mean(`Hidden Gem
↪  Score`),median(`Hidden Gem Score`),sd(`Hidden Gem Score`))
colnames(Distribution_Multi_Languages_Movies)<-c("Mean_score","Median_score","SD_score")




mergedlang = do.call("rbind", list(Distribution_English_or_other,
                                   Distribution_French_or_other,
                                   Distribution_Spanish_or_other,
                                   Distribution_One_Foreign_Language_Movies,
                                   Distribution_Multi_Languages_Movies
                                   ))
rownames(mergedlang) = c("English and Foreign", "French and Foreign", "Spanish and
↪  Foreign", "One Foreign Language", "Multilanguage")
kable(mergedlang, row.names = TRUE) %>%
kable_classic(full_width = TRUE, html_font = "Cambria")
```

	Mean_score	Median_score	SD_score
English and Foreign	3.311193	3.4	1.059043
French and Foreign	4.464706	4.2	1.058266
Spanish and Foreign	4.388000	4.2	1.432748
One Foreign Language	4.578571	4.2	1.170147
Multilanguage	3.399299	3.4	1.043656

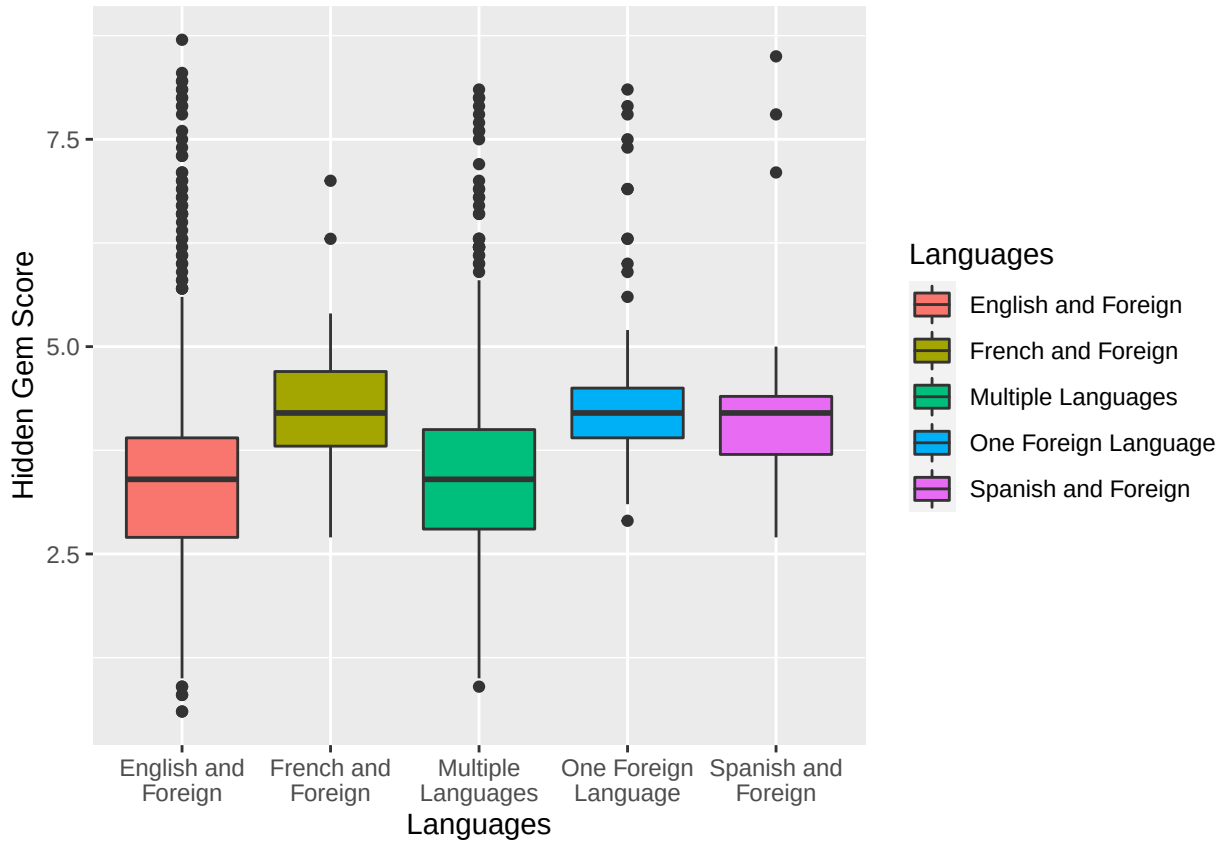
Now, let's look at the plots of these categories:

```
Languages_Plot<-ggplot()+
  geom_density(data=English_or_other,aes(x=`Hidden Gem Score`, colour =
  ↪  Languages),adjust=4)+
  geom_density(data=French_or_other,aes(x=`Hidden Gem Score`,color= Languages),adjust=4)+
  geom_density(data=Multi_Languages_Movies,aes(x=`Hidden Gem
  ↪  Score`,color=Languages),adjust=4)+
  geom_density(data=Spanish_or_other,aes(x=`Hidden Gem Score`,color=
  ↪  Languages),adjust=4)+
  geom_density(data = One_Foreign_Language_Movies, aes(x=`Hidden Gem Score`, color=
  ↪  Languages),adjust=4)+
  theme(legend.position="right")

Languages_Plot
```




```
boxplots = Final_Project_FlixGem %>%
  group_by(Languages) %>%
  ggplot(mapping = aes(y = `Hidden Gem Score`, x = Languages, fill = Languages)) +
  geom_boxplot() +
```

```
  theme(axis.text.x = element_text(angle = 0)) +
  scale_x_discrete(labels = function(x) stringr::str_wrap(x, width = 15))


boxplots
```



- Movies available in English but not French and Spanish, as well as movies available in multiple languages seem to have similar distributions of Hidden Gem Scores. Movies available exclusively in French, Spanish, or other languages excluding English do seem to fare better on average than other English movies as well as movies available in multiple languages, but small samples of these categories bring large variance to their Hidden Gem Scores distributions: we thus can't conclude with a sufficient degree of certainty that French and Spanish movies actually fare better. We do have a large sample of movies available in only one foreign language, so looking at their higher mean Hidden Gem Scores in contrast to English but not French and Spanish movies and movies available in multiple languages, we can conclude with high certainty that movies available in only foreign language actually are getting high mean Hidden Gem Scores than most other movies.
- If we included French and Spanish Movies in the "One Foreign Language" category, thus having all movies available in only one language that isn't English, we could say as a rule of thumb that movies available in English seem to have worse Hidden Gem Scores than other movies.

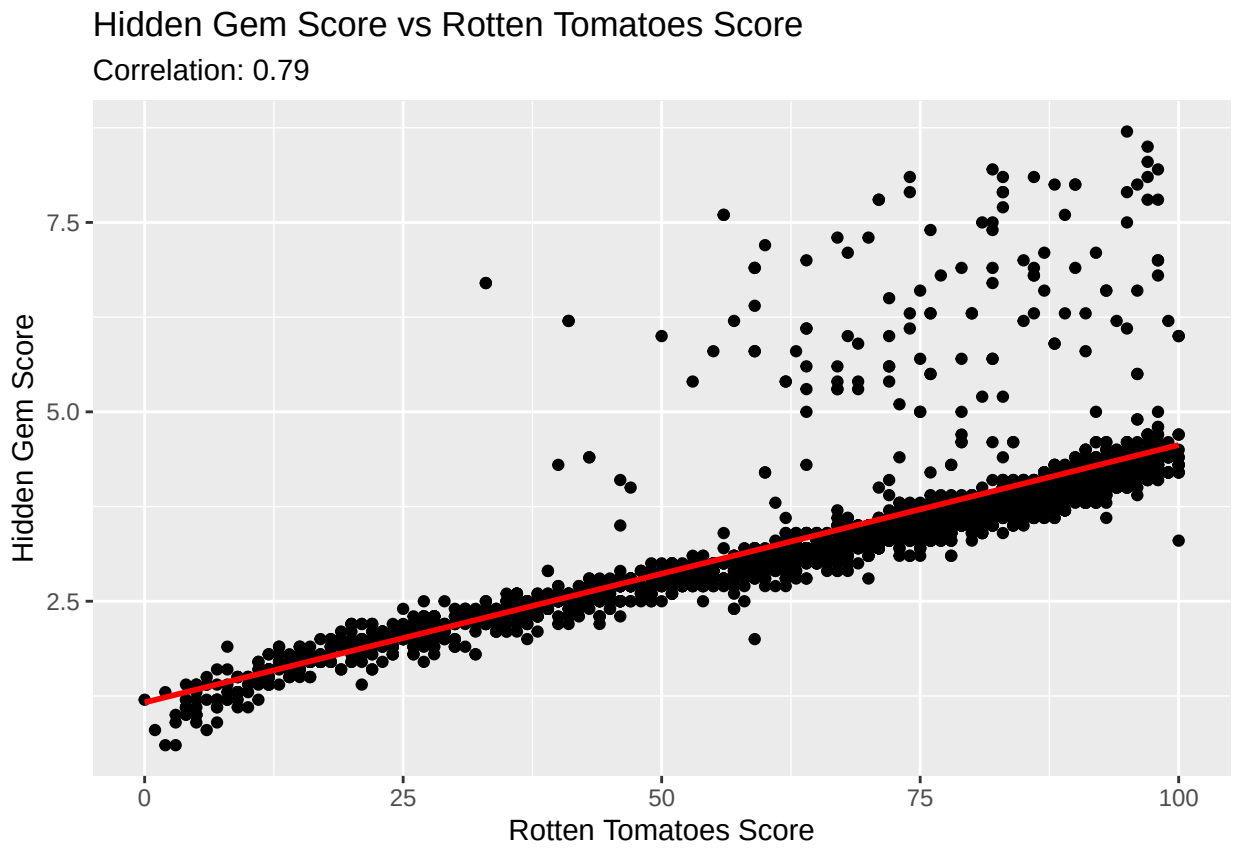
---

b) Associations between hidden gem score and other scores

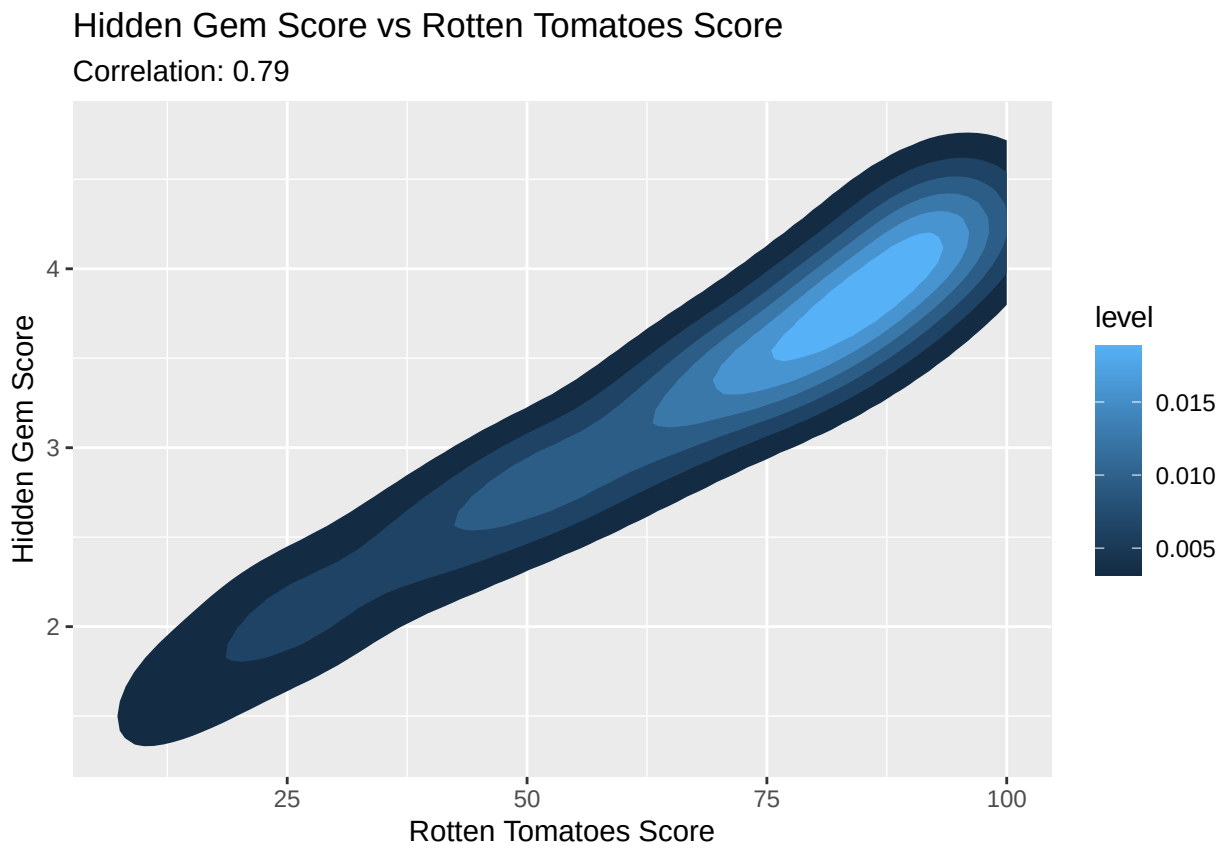
First, let's look at Rotten Tomatoes score and Hidden Gem Score

```
correlation1 = round(with(Final_Project_FlixGem,cor(`Rotten Tomatoes Score`,`Hidden Gem
↪  Score`)), digits = 2)


association_plot1<-ggplot(Final_Project_FlixGem,aes(x=`Rotten Tomatoes Score`,y=`Hidden
↪  Gem Score`))+geom_point()+geom_smooth(method="lm",col="red",se=F)+ggtitle("Hidden Gem
↪  Score vs Rotten Tomatoes Score", subtitle = "Correlation: 0.79")

contour1 = ggplot(Final_Project_FlixGem, aes(x = `Rotten Tomatoes Score`,
                                             y = `Hidden Gem Score`)) +
  stat_density2d(bins = 7,
                 aes(fill = ..level..),
                 geom = "polygon") +
  ggtitle("Hidden Gem Score vs Rotten Tomatoes Score", subtitle = "Correlation: 0.79")

print(association_plot1)
```

```
## `geom_smooth()` using formula 'y ~ x'
```



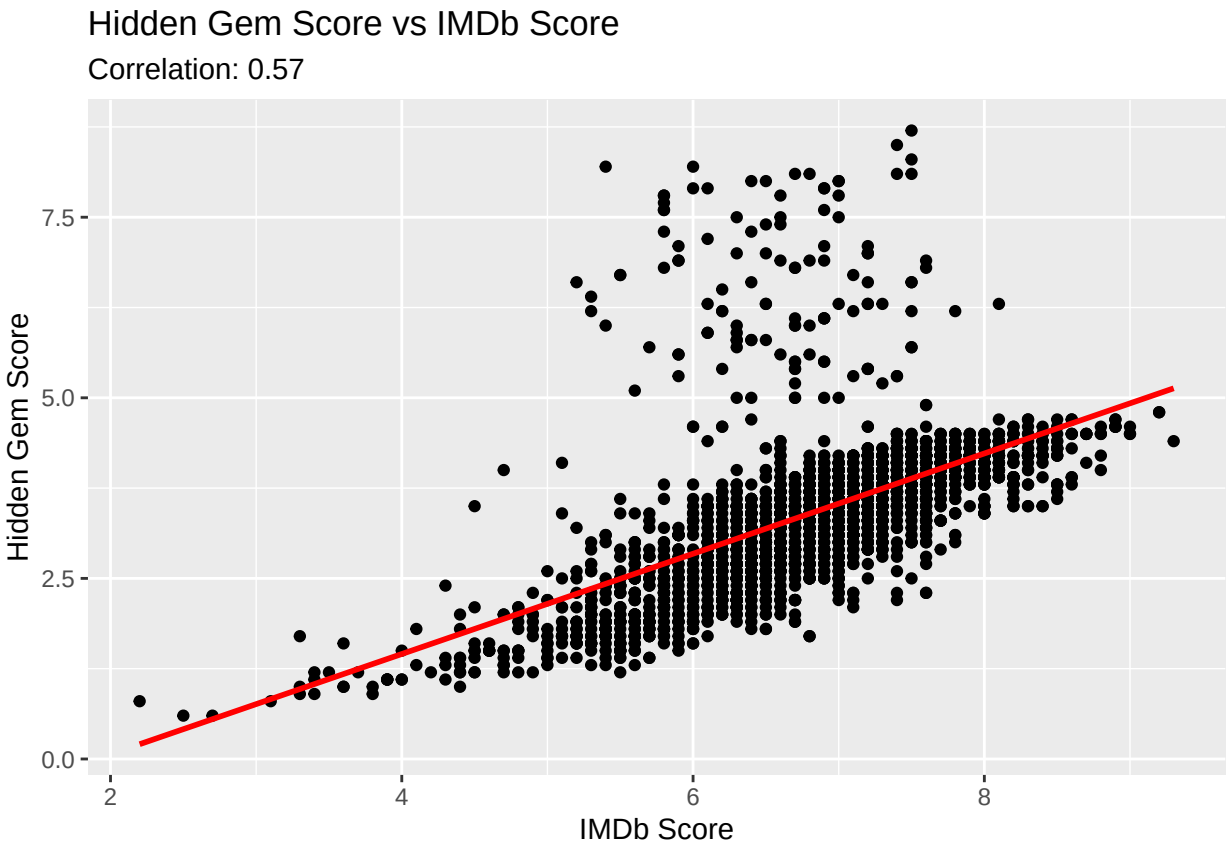

```
print(contour1)
```



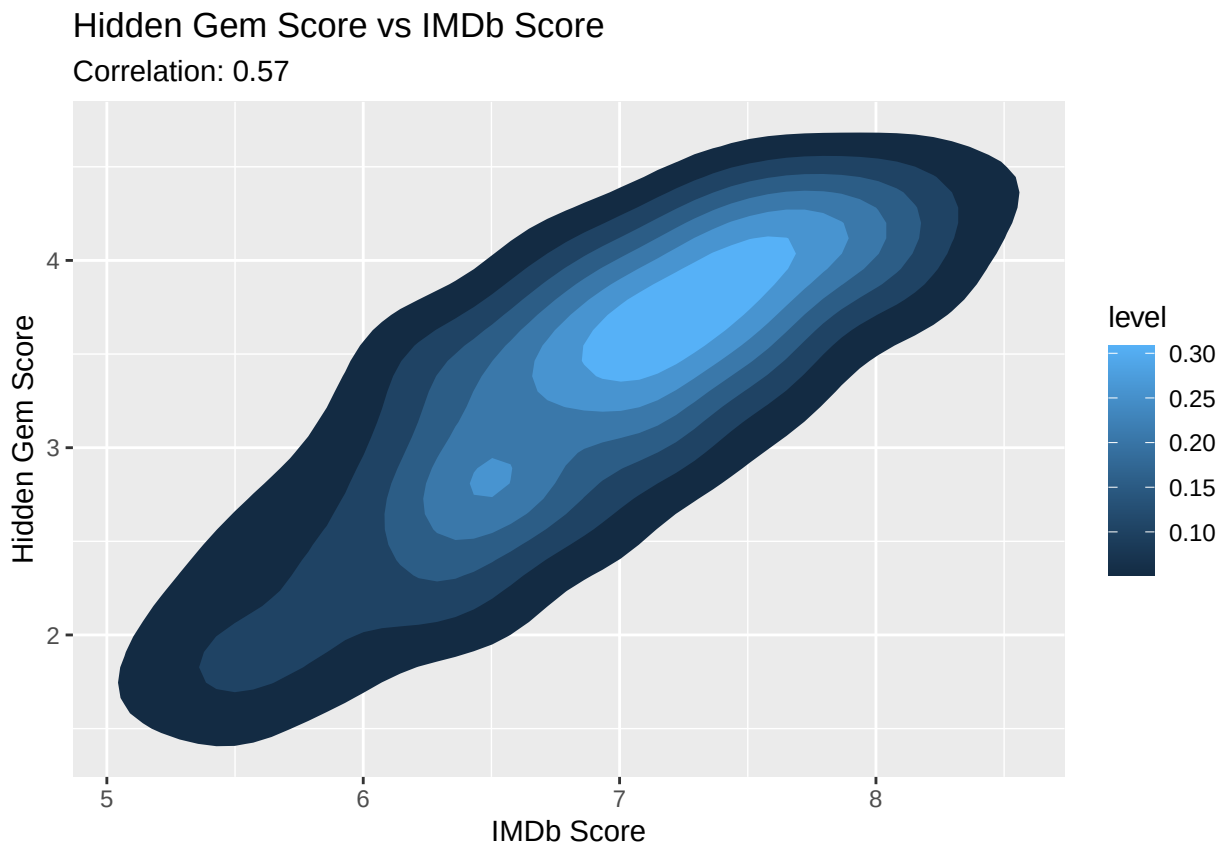
Lets repeat the process for IMDb and Metacritic

```
correlation2 = round(with(Final_Project_FlixGem,cor(`IMDb Score`,`Hidden Gem Score`)),
↪  digits = 2)

association_plot2<-ggplot(Final_Project_FlixGem,aes(x=`IMDb Score`,y=`Hidden Gem
↪  Score`))+geom_point()+geom_smooth(method="lm",col="red",se=F)+ggtitle("Hidden Gem
↪  Score vs IMDb Score", subtitle = "Correlation: 0.57")

contour2 = ggplot(Final_Project_FlixGem, aes(x = `IMDb Score`,
                                             y = `Hidden Gem Score`)) +
  stat_density2d(bins = 7,
                 aes(fill = ..level..),
                 geom = "polygon") +
  ggtitle("Hidden Gem Score vs IMDb Score", subtitle = "Correlation: 0.57")


print(association_plot2)
```

```
## `geom_smooth()` using formula 'y ~ x'
```

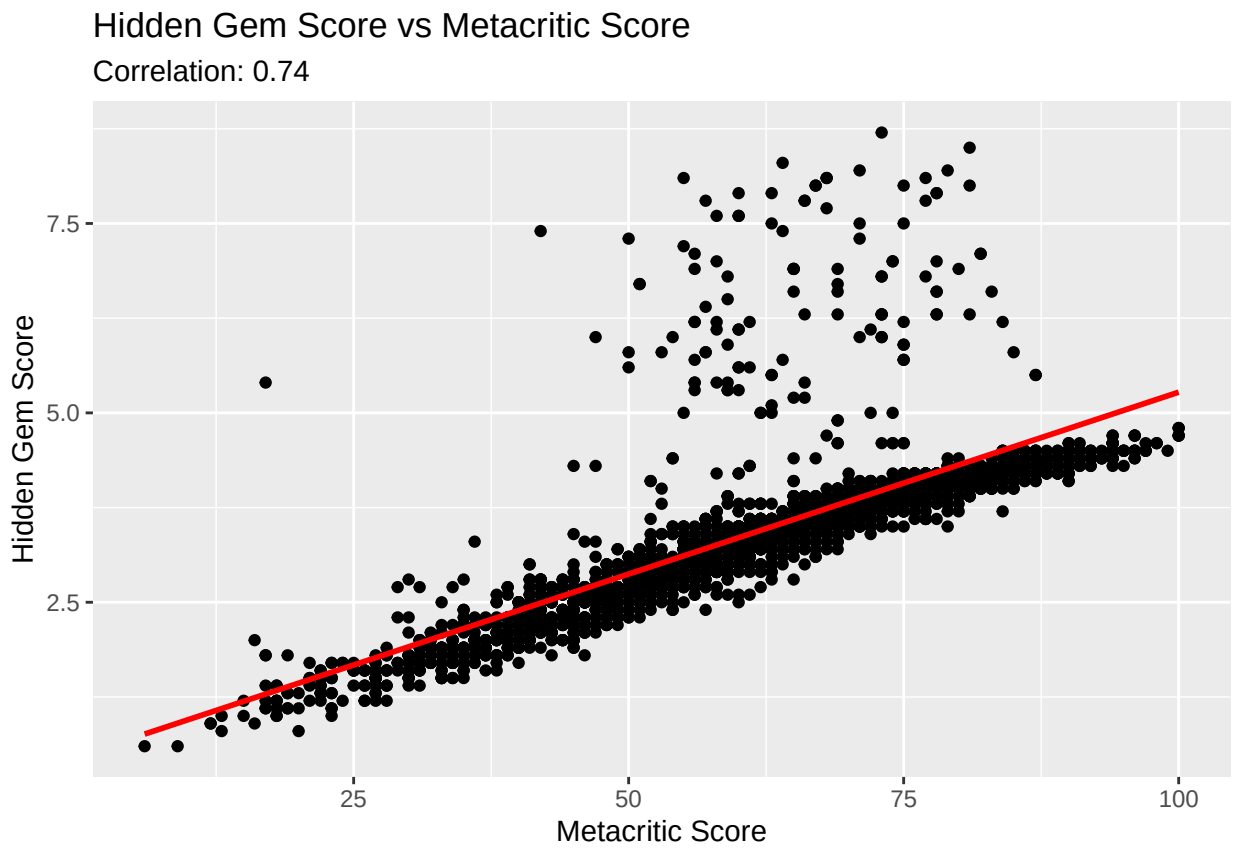



```
print(contour2)
```

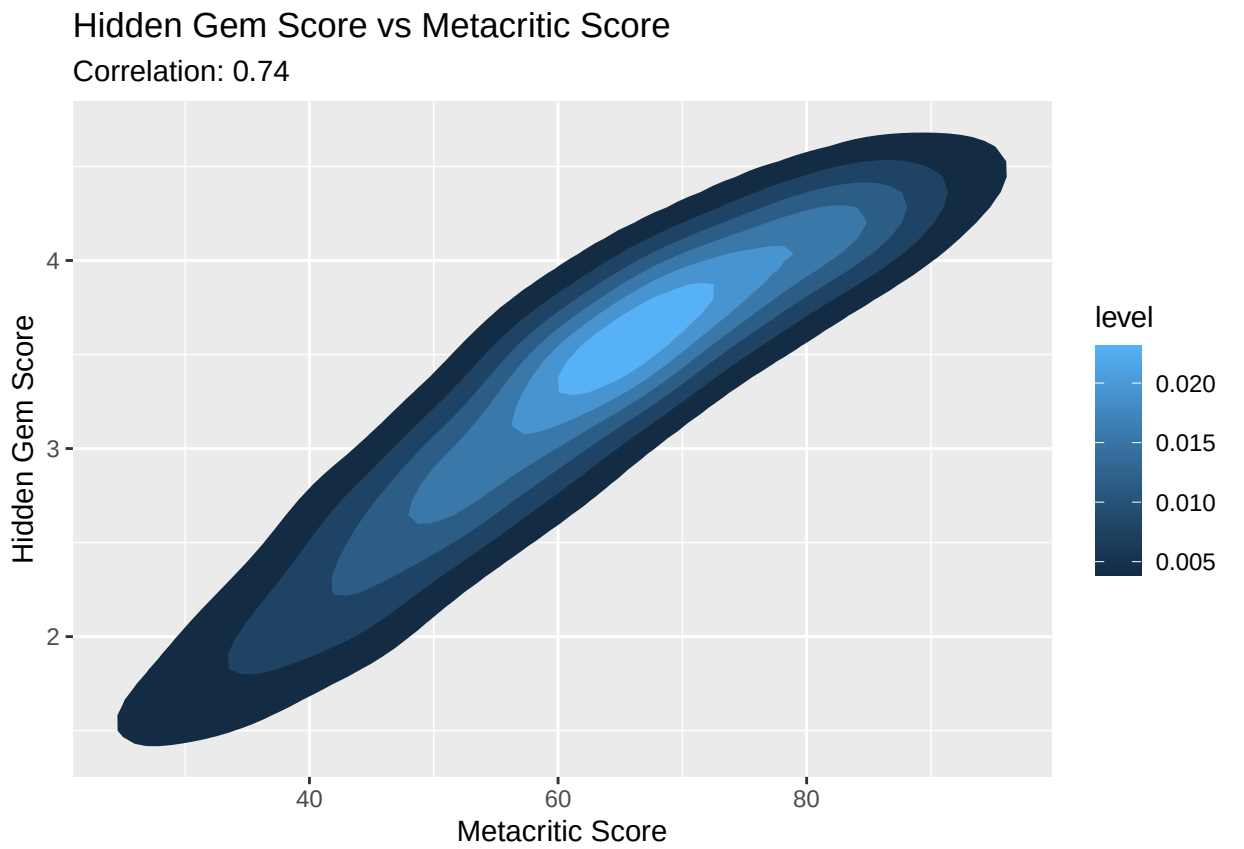



```
correlation3 = round(with(Final_Project_FlixGem,cor(`Metacritic Score`,`Hidden Gem
↪  Score`)), digits = 2)

association_plot3<-ggplot(Final_Project_FlixGem,aes(x=`Metacritic Score`,y=`Hidden Gem
↪  Score`))+geom_point()+geom_smooth(method="lm",col="red",se=F)+ggtitle("Hidden Gem
↪  Score vs Metacritic Score", subtitle = "Correlation: 0.74")

contour3 = ggplot(Final_Project_FlixGem, aes(x = `Metacritic Score`,
                                             y = `Hidden Gem Score`)) +
  stat_density2d(bins = 7,
                 aes(fill = ..level..),
                 geom = "polygon") +
  ggtitle("Hidden Gem Score vs Metacritic Score", subtitle = "Correlation: 0.74")

print(association_plot3)
```

```
## `geom_smooth()` using formula 'y ~ x'
```




```
print(contour3)
```

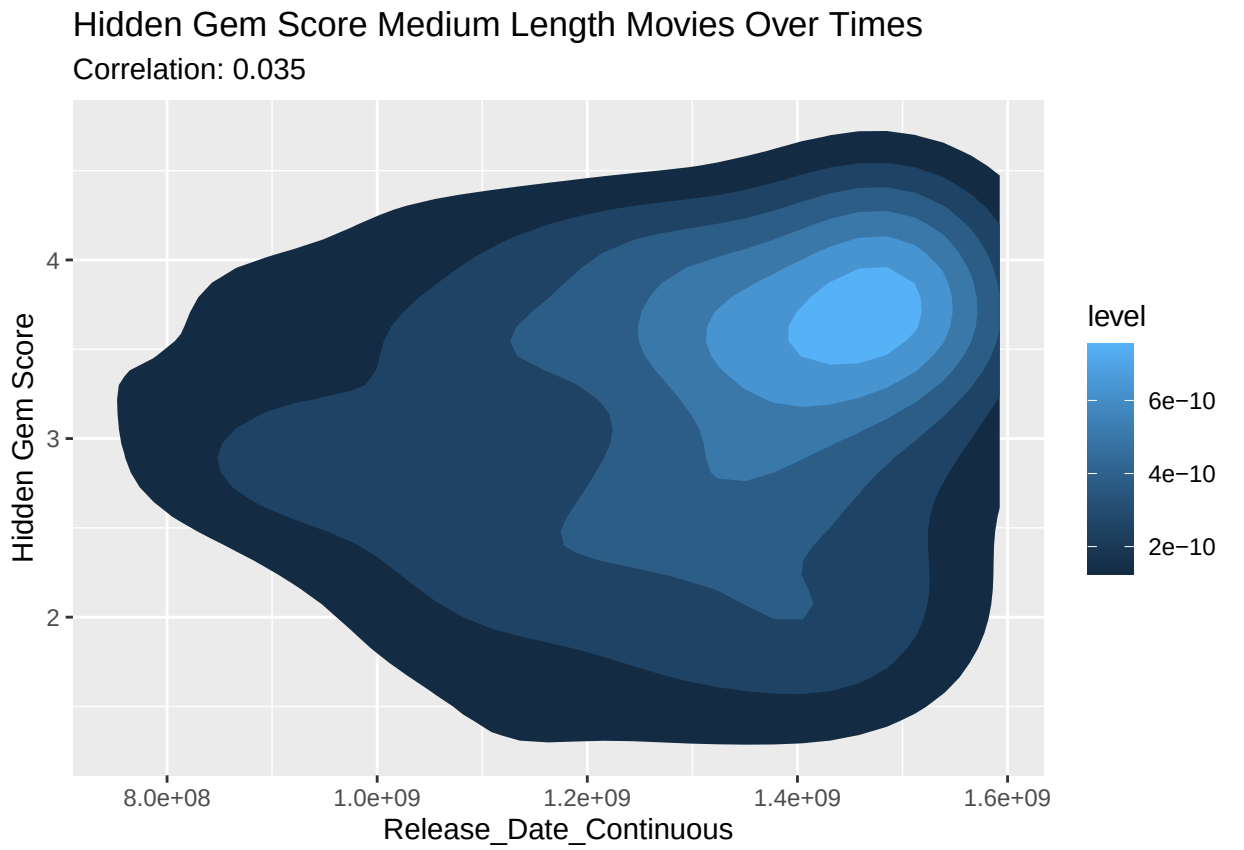


From the six plots above, there seems to be a positive association between Hidden Gem Scores and Rotten Tomatoes scores, Hidden Gem Scores and IMDb scores, as well as Hidden Gem Scores and Metacritic scores. Having an increased score on the three alternate sites clearly, on average, increases the Hidden Gem Score. Looking at the computed correlation between each pair of variables, we can say that IMDb scores have a weaker positive association with Hidden Gem Scores than the two other pairs (correlation coefficient of 0.57 versus correlation coefficients above 0.74).

##c) Do longer movies seem to be prefered to normal movies over time?

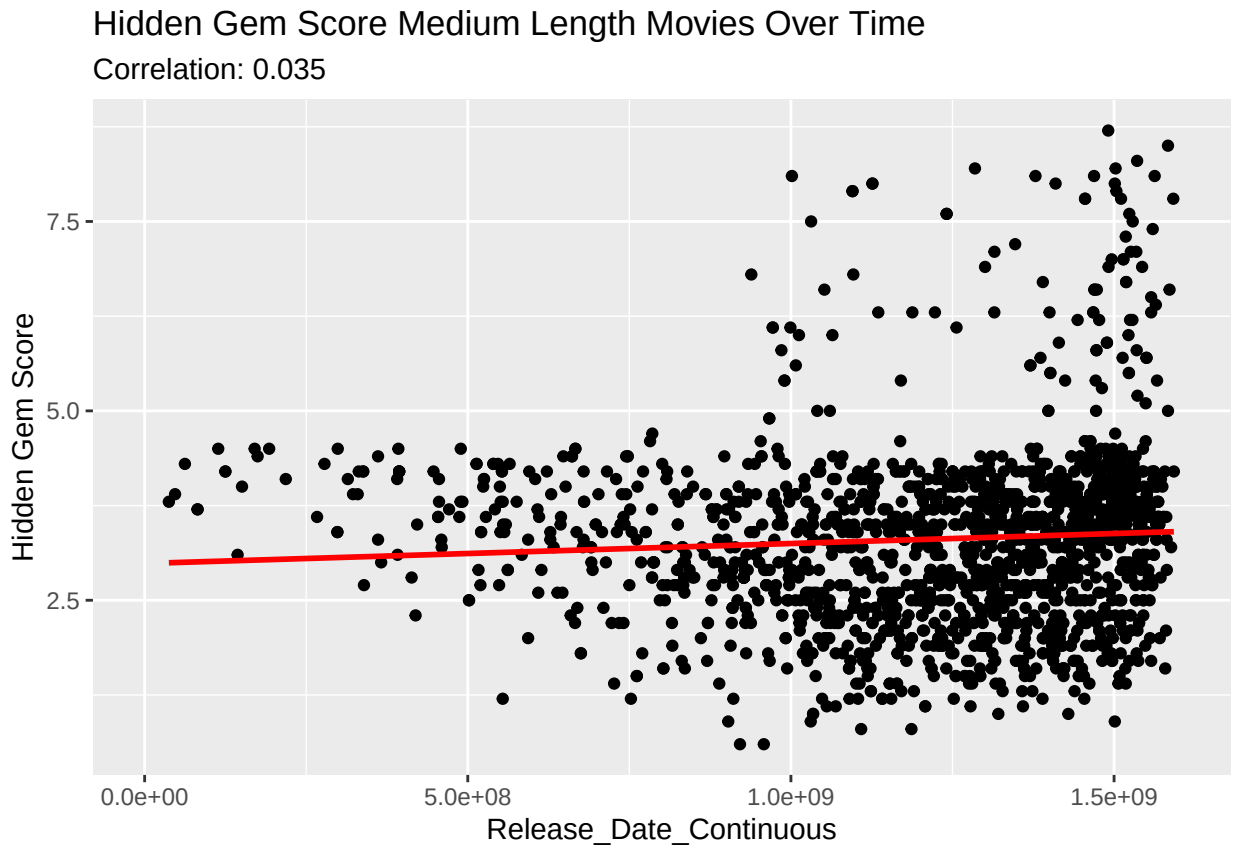
```
##Step 1: turn dates into a continuous variable

Final_Project_FlixGem<-Final_Project_FlixGem%>%mutate(Release_Date_Continuous
↪  =as.numeric(`Release Date`))

##step 2: analyze normal movie preference over time

Final_Project_FlixGem_med_movie<-Final_Project_FlixGem%>%filter(Runtime=="1-2 hour")

Hidden_Gem_Score_Medium_Length_Movies_Over_Time<-ggplot(Final_Project_FlixGem_med_movie,aes(x=Release_Da
↪  Gem Score`))+geom_point()+geom_smooth(method="lm",col="red",se=F)+ggtitle("Hidden Gem
↪  Score Medium Length Movies Over Time", subtitle = "Correlation:
↪  0.035")+xlim(0,1.6*10^9)

contour_dates_med = ggplot(Final_Project_FlixGem_med_movie, aes(x =
↪  Release_Date_Continuous,
                                          y = `Hidden Gem Score`)) +
  stat_density2d(bins = 7,
```

```
                aes(fill = ..level..),
                geom = "polygon") +
  ggtitle("Hidden Gem Score Medium Length Movies Over Times", subtitle = "Correlation:
  ↪  0.035")

print(contour_dates_med)
```




```
print(Hidden_Gem_Score_Medium_Length_Movies_Over_Time)
```

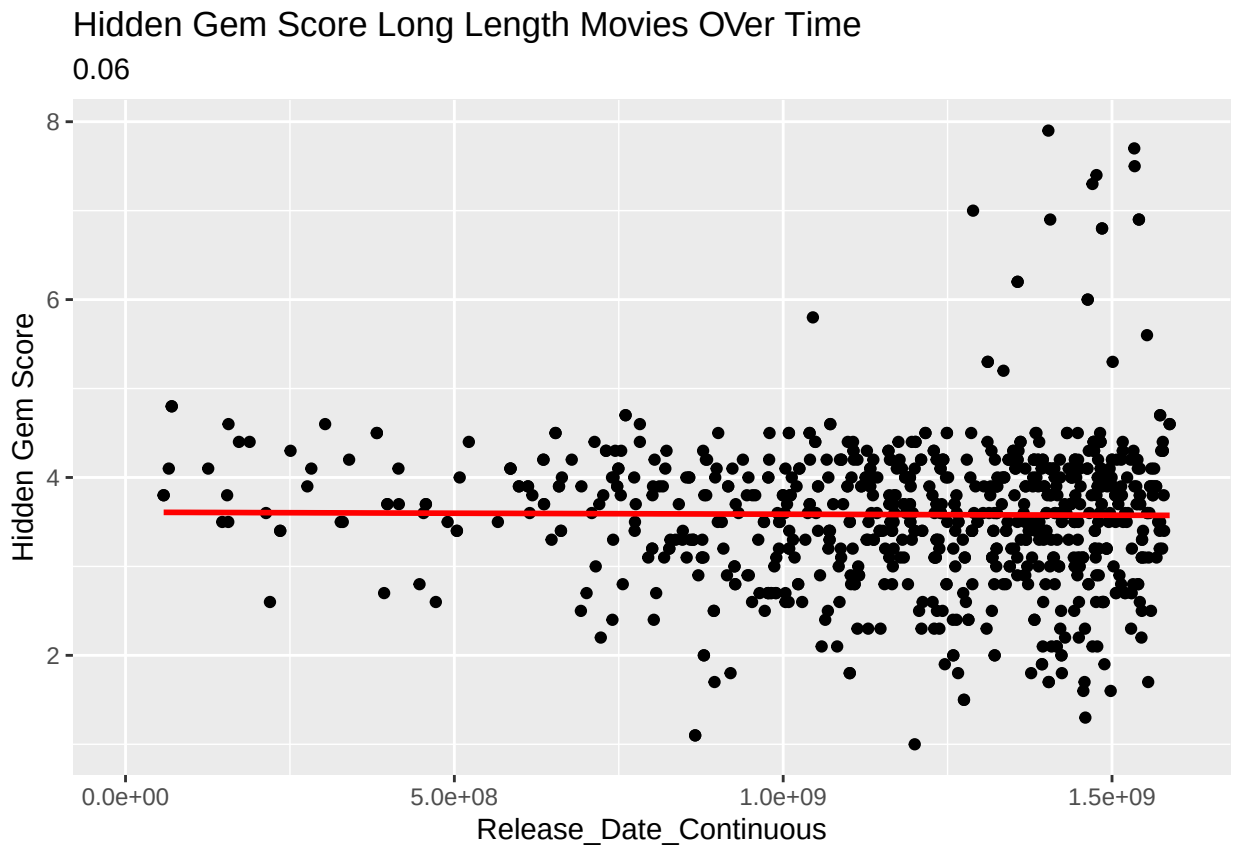
```
## `geom_smooth()` using formula 'y ~ x'
```



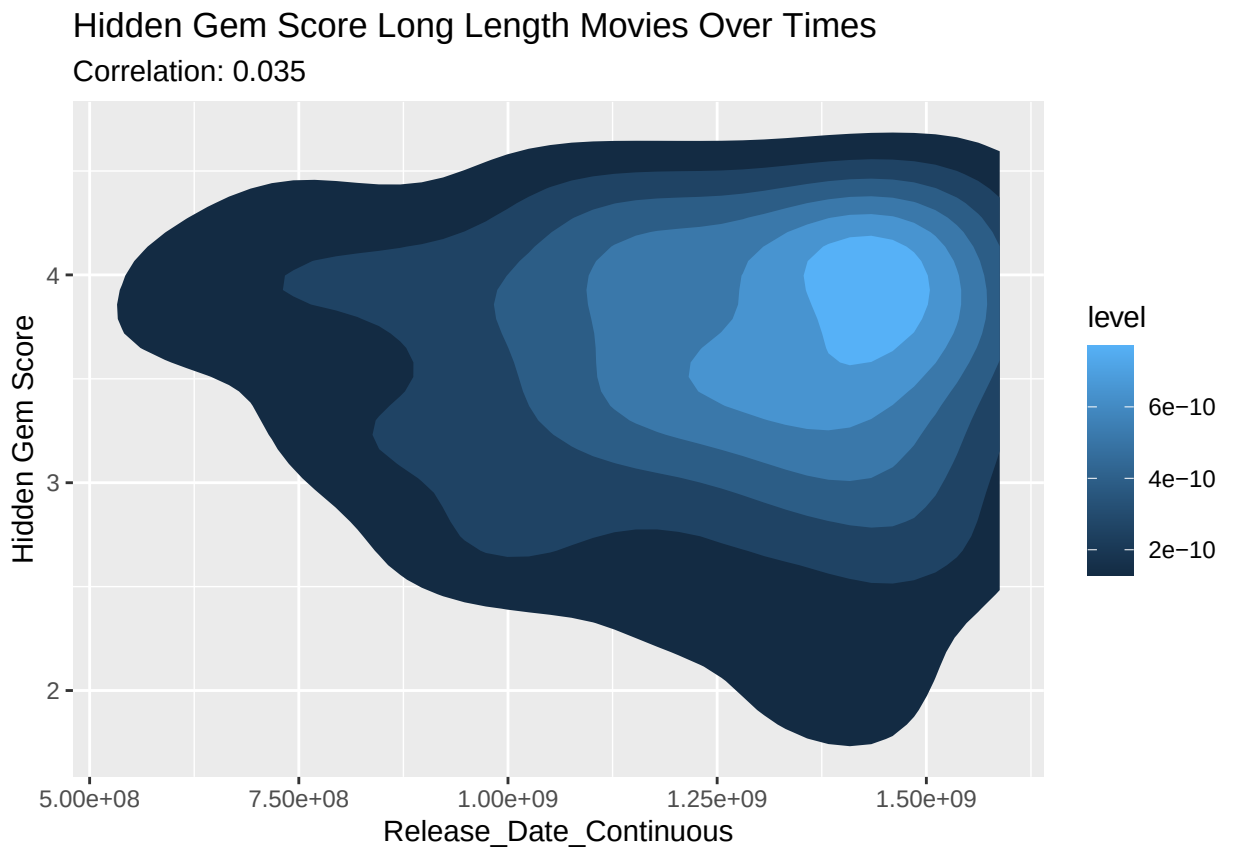

```
# with(Final_Project_FlixGem_med_movie,cor(Release_Date_Continuous,`Hidden Gem Score`))
↪  generates correlation, commented to remove output

##Correlation of 0.03464
```

```
##Step 3: Analyze association for longer movies

Final_Project_FlixGem_long_movie<-Final_Project_FlixGem%>%filter(Runtime=="> 2 hrs")

Hidden_Gem_Score_Long_Length_Movies_Over_Time<-ggplot(Final_Project_FlixGem_long_movie,aes(x=Release_Da
↪  Gem Score`))+
  geom_point()+
  geom_smooth(method="lm",col="red",se=F)+
  ggtitle("Hidden Gem Score Long Length Movies OVer Time", subtitle =
  ↪  "0.06")+xlim(0,1.6*10^9)

contour_dates_long = ggplot(Final_Project_FlixGem_long_movie, aes(x =
↪  Release_Date_Continuous,
                                                  y = `Hidden Gem Score`)) +
  stat_density2d(bins = 7,
                 aes(fill = ..level..),
                 geom = "polygon") +
  ggtitle("Hidden Gem Score Long Length Movies Over Times", subtitle = "Correlation:
  ↪  0.035")

print(Hidden_Gem_Score_Long_Length_Movies_Over_Time)
```

```
## `geom_smooth()` using formula 'y ~ x'
```




```
print(contour_dates_long)
```




```
# with(Final_Project_FlixGem_long_movie,cor(Release_Date_Continuous,`Hidden Gem Score`))

##Correlation of -0.0558
```

In this analysis, we did not include shorter length movies (< 30 minutes) as they are not adequately represented in our sample: trying to draw conclusions from 9 data points will lead us to erroneous conclusions with too high a probability. We will thus stick with the length categories that have several hundred data points.

Looking at the above plots, it seems that there is no trend showing further appreciation of longer movies over time. Looking at the correlation coefficient between time and scores for long length movies (>2 hours), we even see a very weak negative correlation. On the contrary, we have an extremely weak positive correlation (although still above zero) between time and scores for medium length movies. Assuming that people watched movies on Netflix at home in greater proportion over time, it seems impossible to conclude that the comfort of your own home makes you appreciate longer movies more. Their scores, relative to other medium length movies, don't increase over time.

Part II: Factors of the Hidden Gem Score

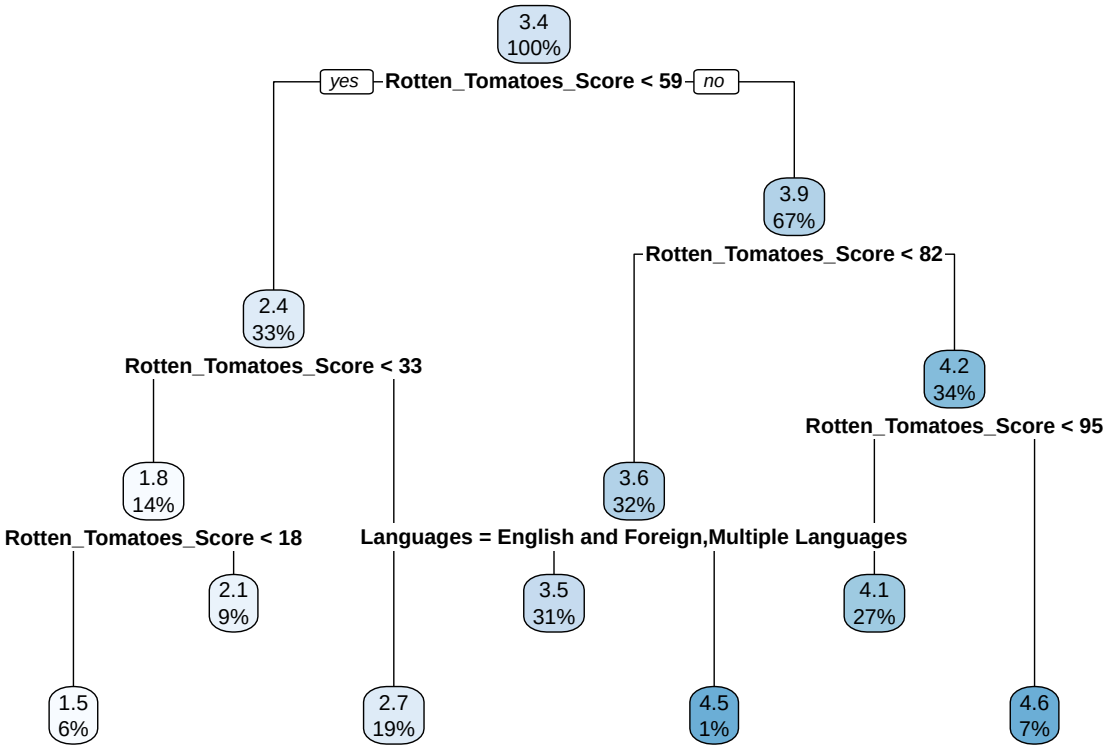
First off, let's select data we want to use:

```
names = colnames(Final_Project_FlixGem)
newnames = sub(" ", "_", x = names)
newnames2 = sub(" ", "_", x = newnames) # not really sure why I needed to do this twice
↪  but anyway
colnames(Final_Project_FlixGem) = newnames2


Final_Project_FlixGem_Tree<-Final_Project_FlixGem%>%
  select(Hidden_Gem_Score,Languages,`Runtime`,IMDb_Score,Rotten_Tomatoes_Score,
  ↪  Metacritic_Score)

Final_Project_FlixGem_Tree
```

```
## # A tibble: 2,602 x 6
##    Hidden_Gem_Score Languages           Runtime  IMDb_Score Rotten_Tomatoes_Sco~
##               <dbl> <chr>               <chr>         <dbl>                <dbl>
##  1              3.5 English and Foreign 1-2 hour        8.4                   68
##  2              2.8 English and Foreign 1-2 hour        6.5                   52
##  3              4.4 English and Foreign 1-2 hour        8.1                   96
##  4              4.2 English and Foreign 1-2 hour        7.7                   89
##  5              4.1 English and Foreign 1-2 hour        8.1                   91
##  6              3.4 English and Foreign > 2 hrs         7                     72
##  7              7.8 English and Foreign 1-2 hour        6.6                   97
##  8              2.8 English and Foreign 1-2 hour        6.5                   52
##  9              2.8 English and Foreign > 2 hrs         6.8                   59
## 10              6.9 English and Foreign > 2 hrs         5.9                   59
## # ... with 2,592 more rows, and 1 more variable: Metacritic_Score <dbl>
```

Now we must split the data into training and testing sets, then create an initial tree. Both this step and the section on tuning were influenced and contain some code from the UC Business Analytics R Programming gudide found here: https://uc-r.github.io/regression_trees [1]

```
set.seed(614)

tree_split <- initial_split(Final_Project_FlixGem_Tree, prop = .7)
tree_train <- training(tree_split)
tree_test  <- testing(tree_split)

m1 = rpart(
  formula = Hidden_Gem_Score ~. ,
  data = tree_train,
  method = "anova",
  minsplit = 15
)

rpart.plot(m1)
```



Thus we have obtained our initial results. We now would like to tune the model so as to determine the maximum numbers of times we want to create a new split in the tree, and the minimum number of observations in our final splitted nodes. We will preform the analysis carried out in the UC Business Analytics R Programming Guide in order to determine the optimal minimum number of data points to split at, as well as the optimal node count[1].

```
set.seed(614)

hypergrid = expand.grid(
  minsplit = seq(5, 20, 1),
  maxdepth = seq(4, 10, 1)
)
nrow(hypergrid)
```

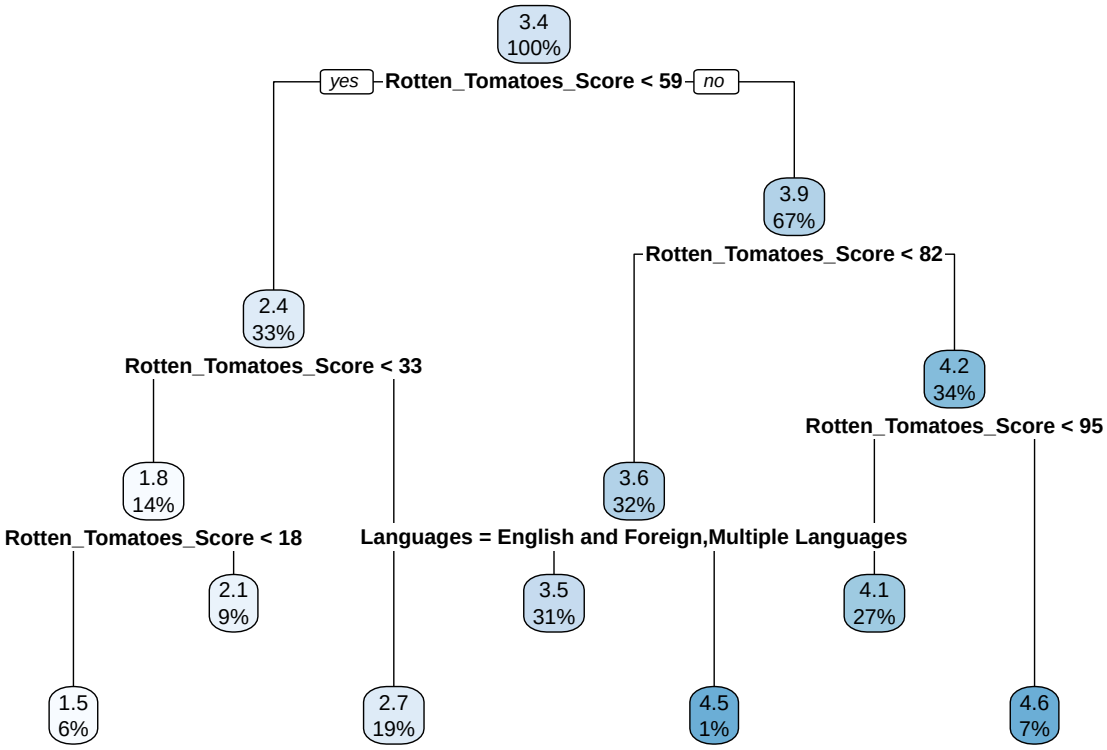
```
## [1] 112
```

```
models = list()

for(i in 1:nrow(hypergrid)) {
  minsplit = hypergrid$minsplit[i]
  maxdepth = hypergrid$maxdepth[i]

  models[[i]] = rpart(
    formula = Hidden_Gem_Score ~ . ,
    data = tree_train,
    method = "anova",
    control = list(minsplit = minsplit, maxdepth = maxdepth)
```

```
  )
}



get_cp <- function(x) {
  min    <- which.min(x$cptable[, "xerror"])
  cp <- x$cptable[min, "CP"]
}

# function to get minimum error
get_min_error <- function(x) {
  min    <- which.min(x$cptable[, "xerror"])
  xerror <- x$cptable[min, "xerror"]
}

hypergrid %>%
  mutate(
    cp    = purrr::map_dbl(models, get_cp),
    error = purrr::map_dbl(models, get_min_error)
    ) %>%
  arrange(error) %>%
  top_n(-5, wt = error)
```

```
##   minsplit maxdepth   cp     error
## 1       10        9 0.01 0.3947232
## 2       20        8 0.01 0.3954217
## 3       16        6 0.01 0.3956723
## 4       12        8 0.01 0.3965789
## 5        5        4 0.01 0.3978486
```

```
m2 = rpart(
  formula = Hidden_Gem_Score ~. ,
  data = tree_train,
  method = "anova",
  control = list(minsplit = 10, maxdepth = 9, cp = 0.01)
)

rpart.plot(m2)
```



After selecting the optimal maxdepth, cp, and minsplit, we can find the Mean Square Error by trying it on our testing data

```
pred = predict(m2, newdata = tree_test)
RMSE(pred = pred, obs = tree_test$Hidden_Gem_Score)
```

```
## [1] 0.7497471
```

Thus we can conclude that the predicted Hidden Gem Score using our model differs, on average, by about 0.75 from the true Hidden Gem Score for a given movie. We can also conclude from our analysis that of the variables we used to construct our tree, the Rotten Tomatoes Score is the most important predictor for Hidden Gem Scores. Whether or not a movie is either multilanguage or has English or other languages is also a significant predictor. Movies obtaining higher scores on Rotten Tomatoes will, on average, obtain higher Hidden Gem Scores. Movies available in English, as well as movies available in multiple languages, will on average obtain weaker Hidden Gem Scores in contrast to movies in these three categories: available in French and foreign languages, Spanish and foreign languages, as well as one foreign language.

Part III: An H-index for directors

The top 10 directors in the dataset according to the Hidden Gem /h-index are listed below.

```
directors_vector = c()
number_of_movies_vector= c()
rows_of_movies_list =list()

for (row in 1:nrow(Final_Project_FlixGem_final)){
  director_name = Final_Project_FlixGem_final[[row, "Director"]]

  if (director_name %in% directors_vector){
    i = which(directors_vector == director_name)
    number_of_movies_vector[i] = number_of_movies_vector[i] + 1

    rows_of_movies_list[i] = list(c(rows_of_movies_list[[i]],row))
  }else{
    directors_vector = c(directors_vector,director_name)
    i = which(directors_vector == director_name)
    number_of_movies_vector[i] = 1

    rows_tracker = c(row)
    rows_of_movies_list[i] = list(rows_tracker)



  }


}

t = tibble(d= directors_vector, m= number_of_movies_vector)


#now the h index
hindex_vector = c()

for (roww in 1:length(directors_vector)){
  movies_rows = rows_of_movies_list[[roww]]
  max_h = t[[roww, "m"]]

  for (h in max_h:0){
    counter = 0
    for ( movie_r in movies_rows){
      if(Final_Project_FlixGem_final[[movie_r, "Hidden Gem Score"]]>=h){
        counter = counter +1
      }
    }
    if (counter >= h){
      hindex_vector[roww] = h
      break
    }
  }
```

```
}

director_hindex_t = tibble(Director = directors_vector, HG_H =hindex_vector) %>%
↪  arrange(desc(HG_H))
director_hindex_t_top10 = head(director_hindex_t, n=10)

kable(director_hindex_t_top10) %>% kable_classic(full_width = F, html_font = "Cambria")
```

Director	HG_H
Steven Spielberg	4
Quentin Tarantino	4
Ang Lee	4
David Fincher	4
Bong Joon Ho	4
Woody Allen	4
David Mackenzie	4
Hayao Miyazaki	4
Peter Jackson	4
Paul Thomas Anderson	4

However, for this dataset, all of the top 10 directors have a Hidden Gem H-index of 4 and actually more than 10 directors may have a Gem H-index of 4. All of those directors should be equal and displayed in the top directors list. Therefore, this table displays the top 10 directors according to their HG-H index, but if the 11th,12th,13th ... i'th director also have the same HG-H as the 10th row director, they will also be displayed

```
i = 10
last_row_index =  director_hindex_t[10,2]
new_row_index = director_hindex_t[11,2]
while (new_row_index == last_row_index){
  i = i+1
  new_row_index = director_hindex_t[i,2]
}

#inside the while loop we increment i by 1 but the last incrementation should not happen
i = i-1

#get the first i directors from director_ranking vector.
#directors from the 11th row to the ith row have the same HG-H index than the 10th row
↪  director.
director_hindex_t_topi = head(director_hindex_t, n=i)


kable(list(director_hindex_t_topi)) %>% kable_classic(html_font = "Cambria")
```

Director	HG_H
Steven Spielberg	4
Quentin Tarantino	4
Ang Lee	4
David Fincher	4
Bong Joon Ho	4
Woody Allen	4
David Mackenzie	4
Hayao Miyazaki	4
Peter Jackson	4
Paul Thomas Anderson	4
Ridley Scott	4
Edgar Wright	4
Christopher Nolan	4
Steven Soderbergh	4
Pedro Almodóvar	4
Martin Scorsese	4
Alfonso Cuarón	4

References

[1] Boehmke, B. (2018) UC Business Analytics R Programming Guide: Regression Trees [Source code]. https://uc-r.github.io/regression_trees