Project 1: Streaming Services

Shariq Mallick 3/20/2021

The Setup

Using the website Kaggle, multiple datasets pertaining to content found on different streaming services were collected. Two datasets that had the most similar data to one another were chosen in order to have the cleanest joining and most reasonable conclusions to the analysis. One dataset contains details regarding 50 Netflix TV shows, while the other contains details regarding Amazon Prime TV shows. This data exploration will focus on rating, year, and audience age. This data is interesting to me as an exploration of the range of quality on different services could inform one's decision on choosing one service over another.

A limitation of this data is that the Netflix data is specifically the top 50 most popular shows on Netflix while the prime shows are a wider range. The prime TV shows dataset contain many of the most popular shows on Amazon Prime, but it also contains many of the less acclaimed shows as well. It is expected that shows aimed at younger audiences will have lower IMDB scores than those aimed at older audiences, and that shows provided in the Netflix data set will see higher IMDB scores.

```
prime<-read.csv("D:/Downloads/Rstuff/Prime.csv")
netflix<-read.csv("D:/Downloads/Rstuff/Netflix.csv")

# Due to a lack of numeric variables, another dataset which includes an extra numeric
tv_shows<-read.csv("D:/Downloads/Rstuff/tv_shows.csv")

glimpse(netflix)
```

```
## Rows: 50
## Columns: 5
## $ Titles      <chr> "Breaking Bad", "Game of Thrones", "Rick and Morty", "Dark~
## $ Year        <int> 2008, 2011, 2013, 2017, 2016, 2005, 2010, 2019, 1994, 2005~
## $ Rating      <chr> "18+", "18+", "18+", "16+", "16+", "7+", "16+", "18+", "16~
## $ IMDB_Rating <dbl> 9.5, 9.3, 9.2, 8.8, 8.8, 9.2, 9.1, 9.4, 8.9, 8.9, 8.7, 9.3~
## $ Netflix     <int> 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1~
```

```
glimpse(prime)
```

```
## Rows: 404
## Columns: 8
## $ S.no.                   <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14,~
## $ Name.of.the.show        <chr> "Pataal Lok", "Upload", "The Marvelous Mrs. Ma~
## $ Year.of.release         <int> 2020, 2020, 2017, 2019, 2016, 2019, 2018, 2018~
## $ No.of.seasons.available <int> 1, 1, 3, 2, 2, 1, 2, 1, 1, 1, 2, 2, 2, 2, 1, 1~
## $ Language                <chr> "Hindi", "English", "English", "Hindi", "Engli~
## $ Genre                   <chr> "Drama", "Sci-fi comedy", "Drama, Comedy", "Dr~
## $ IMDb.rating             <dbl> 7.5, 8.1, 8.7, 5.3, 8.7, 8.3, 7.5, 8.5, 8.6, 8~
## $ Age.of.viewers          <chr> "18+", "16+", "16+", "18+", "18+", "18+", "16+~
```

These datasets contain some unneeded variables for this exploration. Therefore, the data must be tidied.

#Citations for data: Bansal, Shivam. “Netflix Movies and TV Shows.” Kaggle, 18 Jan. 2021, www.kaggle.com/shivamb/netflix-shows/metadata. Jauhari, Neelima. “Amazon Prime TV Shows.” Kaggle, 13 Oct. 2020, www.kaggle.com/nilimajauhari/amazon-prime-tv-shows. Bhatia, Ruchi. “TV Shows on Netflix, Prime Video, Hulu and Disney+.” Kaggle, 25 May 2020, www.kaggle.com/ruchi798/tv-shows-on-netflix-prime-video-hulu-and-disney.

Tidying Up and Joining

```
# The tidyverse package contains the needed functions to tidy and visualize the data.
library(tidyverse)

# The prime data contains a redundant variable in s.no, and there are several variabl
tdprime<-prime%>%
  select(-S.no.,-Language, -Genre,-No.of.seasons.available)%>%
# The remaining variables need to be renamed to be the same as the netflix data
  rename(Titles="Name.of.the.show", Year="Year.of.release", IMDB_Rating="IMDb.rating"
# Missing values need to be removed as well
  drop_na()

# Creating a new variable that labels these shows as being from Prime video will help
tdprime$Service<-"Prime"
glimpse(tdprime)
```

```
## Rows: 182
## Columns: 5
```

```
## $ Titles      <chr> "Pataal Lok", "Upload", "The Marvelous Mrs. Maisel", "Four~
## $ Year        <int> 2020, 2020, 2017, 2019, 2016, 2019, 2018, 2018, 2019, 2019~
## $ IMDB_Rating <dbl> 7.5, 8.1, 8.7, 5.3, 8.7, 8.3, 7.5, 8.5, 8.6, 8.0, 8.0, 8.7~
## $ Rating      <chr> "18+", "16+", "16+", "18+", "18+", "18+", "16+", "18+", "1~
## $ Service     <chr> "Prime", "Prime", "Prime", "Prime", "Prime", "Prime", "Pri~
```

The netflix dataset is already tidy except for the netflix variable which denotes whether a show is a netflix original. This data is unnecessary, but using pivot_longer this variable can be used to denote netflix as the streaming service

```
tdnetflix<-netflix%>%
  pivot_longer(cols = Netflix, names_to = "Service", values_to = "Netflix")%>%
# Once the new variable was created, the redundant one could be removed
  select(-Netflix)
glimpse(tdnetflix)
```

```
## Rows: 50
## Columns: 5
## $ Titles      <chr> "Breaking Bad", "Game of Thrones", "Rick and Morty", "Dark~
## $ Year        <int> 2008, 2011, 2013, 2017, 2016, 2005, 2010, 2019, 1994, 2005~
## $ Rating      <chr> "18+", "18+", "18+", "16+", "16+", "7+", "16+", "18+", "16~
## $ IMDB_Rating <dbl> 9.5, 9.3, 9.2, 8.8, 8.8, 9.2, 9.1, 9.4, 8.9, 8.9, 8.7, 9.3~
## $ Service     <chr> "Netflix", "Netflix", "Netflix", "Netflix", "Netflix", "Ne~
```

```
# Now that the datasets are more similar, a full join function can be used to merge t
fulldata<-tdnetflix%>%
  full_join(tdprime)
```

```
## Joining, by = c("Titles", "Year", "Rating", "IMDB_Rating", "Service")
```

In order to have another numeric variable to test, another dataset containing information on Rotten Tomatoes scores will be joined as well. Rotten Tomato scores differ from IMDB ratings as RT scores are a measure of how many positive vs negative critical review there are, while IMDB ratings are an average of the scores critics gave out of 10.

```
RT<-tv_shows%>%
# First the variables were renamed to be easier to type and match the previous data
  rename(Titles="Title",RT="Rotten.Tomatoes")%>%
# Only the RT variable is necessary, while the Titles serve as the key for the joinin
  select(Titles,RT)%>%
# Missing values can be removed since they add no useful information to the data
```

```
  drop_na()
# Using a leftjoin ensuress only data that is already relevant is added.
fulldata<- fulldata%>%
  left_join(RT,by="Titles")

glimpse(fulldata)
```

```
## Rows: 233
## Columns: 6
## $ Titles      <chr> "Breaking Bad", "Game of Thrones", "Rick and Morty", "Dark~
## $ Year        <int> 2008, 2011, 2013, 2017, 2016, 2005, 2010, 2019, 1994, 2005~
## $ Rating      <chr> "18+", "18+", "18+", "16+", "16+", "7+", "16+", "18+", "16~
## $ IMDB_Rating <dbl> 9.5, 9.3, 9.2, 8.8, 8.8, 9.2, 9.1, 9.4, 8.9, 8.9, 8.9, 8.7~
## $ Service     <chr> "Netflix", "Netflix", "Netflix", "Netflix", "Netflix", "Ne~
## $ RT          <dbl> 0.96, NA, 0.94, 0.94, 0.93, 1.00, 0.78, NA, NA, 0.81, 0.96~
```

Some cases had to be dropped, specifically cases that did not include IMDB ratings in the prime video dataset as missing values in crucial numeric variables would make it difficult to explore the data. Later on, values that are missing in the RT variable may need to be dropped as well; however, for now it is possible to work around the missing data. There are cases of multiple RT scores per title, and this is due to the presence of multiple seasons.

Data Exploration

```
# Create a list of summary statistics for each numeric variable, starting with RT sco
fulldata%>%
# Select for the relevant variables
  select(-Rating,-Titles)%>%
# The summarize function allows one to list out the different statistics
  summarize(mean_RT=mean(RT, na.rm=T),
            sd_RT=sd(RT, na.rm=T),
            min_RT=min(RT,na.rm = T),
            max_RT=max(RT,na.rm=T),
# The n funciton simply lists the number of observations included in the data
            n(),
# The standard error is calculated by dividing the sd by the sqrt of n
            se_RT=sd(RT,na.rm = T)/sqrt(n()))%>%
# The kbl function allows for the creation of a clean table for the statistics
  kbl(caption = "RT score Summary Stats")%>%
# The bootstrap options create specific effects to aid in aesthetics
  kable_styling(bootstrap_options = c("striped", "hover","bordered"))
```

RT score Summary Stats

mean_RT	sd_RT	min_RT	max_RT	n()	se_RT
0.8195652	0.1777856	0.17	1	233	0.0116471

The mean is relatively high for the RT scores, with a mean over 80%. There is a large amount of variation in this variable though, as seen by the sd of over 17%, which is approximately one quarter of the mean. The mean being so high suggests, based on the sample, RT scores are generally scored highly, with a majority above 50%. This suggests that there is a disparity between reviewers perception of an above average score and Rotten Tomato's. If a reviewer scores above a 5/10, Rotten Tomatoes takes it as a positive score. Positive and negative scores are looked at as a binary operator, then the proportion of positive scores to total number of observations is what outputs the score. However, the sample is made of data coming from more popular shows, which would be more likely to score highly. This could be the source of the seeming disparity in this dataset.

```
# Repeat the above with IMDB ratings
fulldata%>%
  select(-Rating,-Titles)%>%
  summarize(mean_IMDB=mean(IMDB_Rating, na.rm=T),
            sd_IMDB=sd(IMDB_Rating, na.rm=T),
            min_IMDB=min(IMDB_Rating,na.rm = T),
            max_IMDB=max(IMDB_Rating,na.rm=T),
            n=n(),
            se_IMDB=sd(IMDB_Rating,na.rm = T)/sqrt(n()))%>%
  kbl(caption = "IMDB Rating Summary Stats")%>%
  kable_styling(bootstrap_options = c("striped", "hover","bordered"))
```

IMDB Rating Summary Stats

mean_IMDB	sd_IMDB	min_IMDB	max_IMDB	n	se_IMDB
7.654506	1.033528	3.7	9.5	233	0.0677087

The mean is lower than the RT scores, which is to be expected based on the difference in how the scores are calculated. The variation is also lower, so the scores are less likely to reach the extremes in the IMDB ratings. This can be seen by the fact that the max is less than 10, while the RT scores maxed at 100%.

```
# Repeat the above with a mutated variable that depicts the ratio of the other two nu
fulldata%>%
    select(-Rating,-Titles)%>%
# The RT values are multiplied by ten to scale them to the IMDB ratings properly sinc
```

```
  mutate(RTDBratio=RT*10/IMDB_Rating)%>%
  summarize(mean_ratio=mean(RTDBratio,na.rm=T),
            sd_ratio=sd(RTDBratio,na.rm=T),
            min_ratio=min(RTDBratio,na.rm = T),
            max_ratio=max(RTDBratio,na.rm = T),
            n(),
            se_ratio=sd(RTDBratio,na.rm = T)/sqrt(n()))%>%
  drop_na()%>%
  kbl(caption = "RT*10/IMDB ratio Summary Stats")%>%
  kable_styling(bootstrap_options = c("striped", "hover","bordered"))
```

RT*10/IMDB ratio Summary Stats

mean_ratio	sd_ratio	min_ratio	max_ratio	n()	se_ratio
1.001567	0.2013255	0.2575758	1.371429	233	0.0131893

Looking at the ratio between these two variables, it is apparent that the RT score tends to be similar to the IMDB score on average, but the large sd and especially low minimum suggests that RT scores have a much larger range. Specifically, an RT score that varies greatly from its IMDB rating usually is much lower than the IMDB score if it is significantly different.

```
# Repeat the above, but this time group the data by the categorical variables
fulldata %>%
  select(-Rating,-Titles)%>%
# The ratio between the RT score and IMDB rating is an interesting method of measurin
  mutate(RTDBratio=RT*10/IMDB_Rating)%>%
# Grouping by Year and Service allows for conclusions to be made regarding the choice
  group_by(Year,Service) %>%
# Remove missing values to avoid errors when computing the summary stats
  drop_na()%>%
# Creating summary statistics for mean and sd of the numeric variables allows for
  summarize(n=n(),
            mean_RT=mean(RT, na.rm=T),
            sd_RT=sd(RT, na.rm=T),
            min_RT=min(RT,na.rm = T),
            max_RT=max(RT,na.rm=T),
            se_RT=sd(RT,na.rm = T)/sqrt(n()),
            mean_IMDB=mean(IMDB_Rating, na.rm=T),
            sd_IMDB=sd(IMDB_Rating, na.rm=T),
            min_IMDB=min(IMDB_Rating,na.rm = T),
            max_IMDB=max(IMDB_Rating,na.rm=T),
            mean_ratio=mean(RTDBratio, na.rm=T),
            se_IMDB=sd(IMDB_Rating,na.rm = T)/sqrt(n()),
            sd_ratio=sd(RTDBratio,na.rm=T),
            min_ratio=min(RTDBratio,na.rm = T),
```

```
            max_ratio=max(RTDBratio,na.rm = T),
            se_ratio=sd(RTDBratio,na.rm = T)/sqrt(n()))%>%
# Arrange the data by year in order to directly compare the two platforms more easily
  arrange(Year)%>%
  kbl(caption = "TV Summary Stats Grouped by Year and Service")%>%
  kable_styling(bootstrap_options = c("striped", "hover","bordered"))
```

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

Year	Service	n	mean_RT	sd_RT	min_RT	max_RT	se_
1989	Netflix	1	0.8500000	NA	0.85	0.85	
1990	Prime	1	0.8800000	NA	0.88	0.88	
1995	Prime	1	1.0000000	NA	1.00	1.00	
1997	Netflix	1	0.8100000	NA	0.81	0.81	
1999	Netflix	1	0.9200000	NA	0.92	0.92	
2001	Netflix	1	0.9400000	NA	0.94	0.94	
2002	Netflix	2	0.8950000	0.0636396	0.85	0.94	0.0450(
2004	Netflix	2	0.8750000	0.0353553	0.85	0.90	0.0250(
2004	Prime	1	0.4000000	NA	0.40	0.40	
2005	Netflix	5	0.9320000	0.0725948	0.81	1.00	0.0324(
2005	Prime	1	0.8300000	NA	0.83	0.83	
2006	Netflix	1	0.7200000	NA	0.72	0.72	
2006	Prime	2	0.8250000	0.1484924	0.72	0.93	0.1050(
2007	Netflix	1	0.9400000	NA	0.94	0.94	
2007	Prime	2	0.7550000	0.2616295	0.57	0.94	0.1850(
2008	Netflix	1	0.9600000	NA	0.96	0.96	
2009	Netflix	4	0.9175000	0.0623832	0.86	1.00	0.0311(

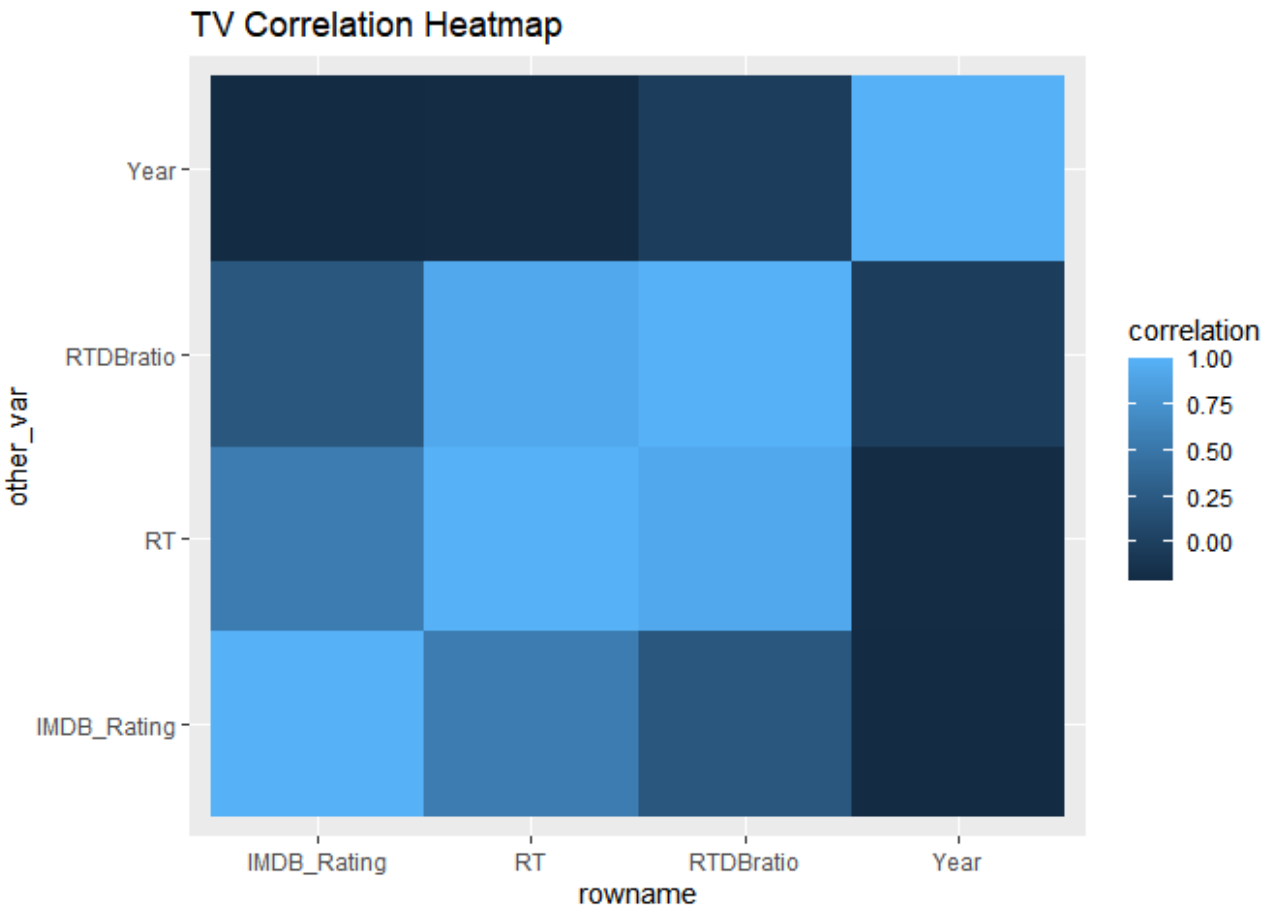
Year	Service	n	mean_RT	sd_RT	min_RT	max_RT	se_
2010	Netflix	2	0.7950000	0.0212132	0.78	0.81	0.0150(
2010	Prime	2	0.8950000	0.0636396	0.85	0.94	0.0450(
2011	Netflix	2	0.8650000	0.0494975	0.83	0.90	0.0350(
2011	Prime	3	0.8866667	0.0321455	0.85	0.91	0.0185
2012	Prime	1	0.7700000	NA	0.77	0.77	
2013	Netflix	6	0.9366667	0.0186190	0.92	0.97	0.0076(
2013	Prime	3	0.7733333	0.2324507	0.51	0.95	0.1342(
2014	Netflix	1	0.9600000	NA	0.96	0.96	
2014	Prime	7	0.7785714	0.2316915	0.27	0.94	0.0875
2015	Netflix	4	0.9300000	0.0336650	0.89	0.97	0.0168
2015	Prime	9	0.7888889	0.2161275	0.30	0.97	0.0720
2016	Netflix	2	0.9000000	0.0424264	0.87	0.93	0.0300(
2016	Prime	13	0.7292308	0.3005678	0.17	1.00	0.0833(
2017	Netflix	4	0.9050000	0.0665833	0.81	0.96	0.0332
2017	Prime	10	0.7860000	0.1635848	0.46	0.96	0.0517
2018	Prime	6	0.7633333	0.2256251	0.50	0.98	0.0921
2019	Netflix	3	0.7900000	0.1311488	0.67	0.93	0.0757
2019	Prime	6	0.7366667	0.1636663	0.57	1.00	0.0668
2020	Prime	3	0.7100000	0.0888819	0.64	0.81	0.0513

There are many missing data points that exist in this table due to these years only containing a single observation. Without a second data point, there is no standard deviation or standard error.

Visualizing Data

A correlation heatmap that shows the similarity between the RT scores and the IMDB ratings would be a suitable way of visualizing the ratio between the two that is displayed by the ratio variables above.

```
fulldata_num<-fulldata%>%
# Mutate to include the ratio from above
  mutate(RTDBratio=RT*10/IMDB_Rating)%>%
# Select for numeric variables
  select_if(is.numeric)
# Create heatmap
cor(fulldata_num,use = "pairwise.complete.obs")%>%
# Save as data frame
  as.data.frame()%>%
# Convert rownames to an explicit variable
  rownames_to_column()%>%
# Pivot so that all the correlations appear in the same column
  pivot_longer(-1, names_to = "other_var",values_to = "correlation")%>%
# Use geom_tile
  ggplot(aes(rowname, other_var, fill=correlation)) + geom_tile() + ggtitle("TV Corre
```



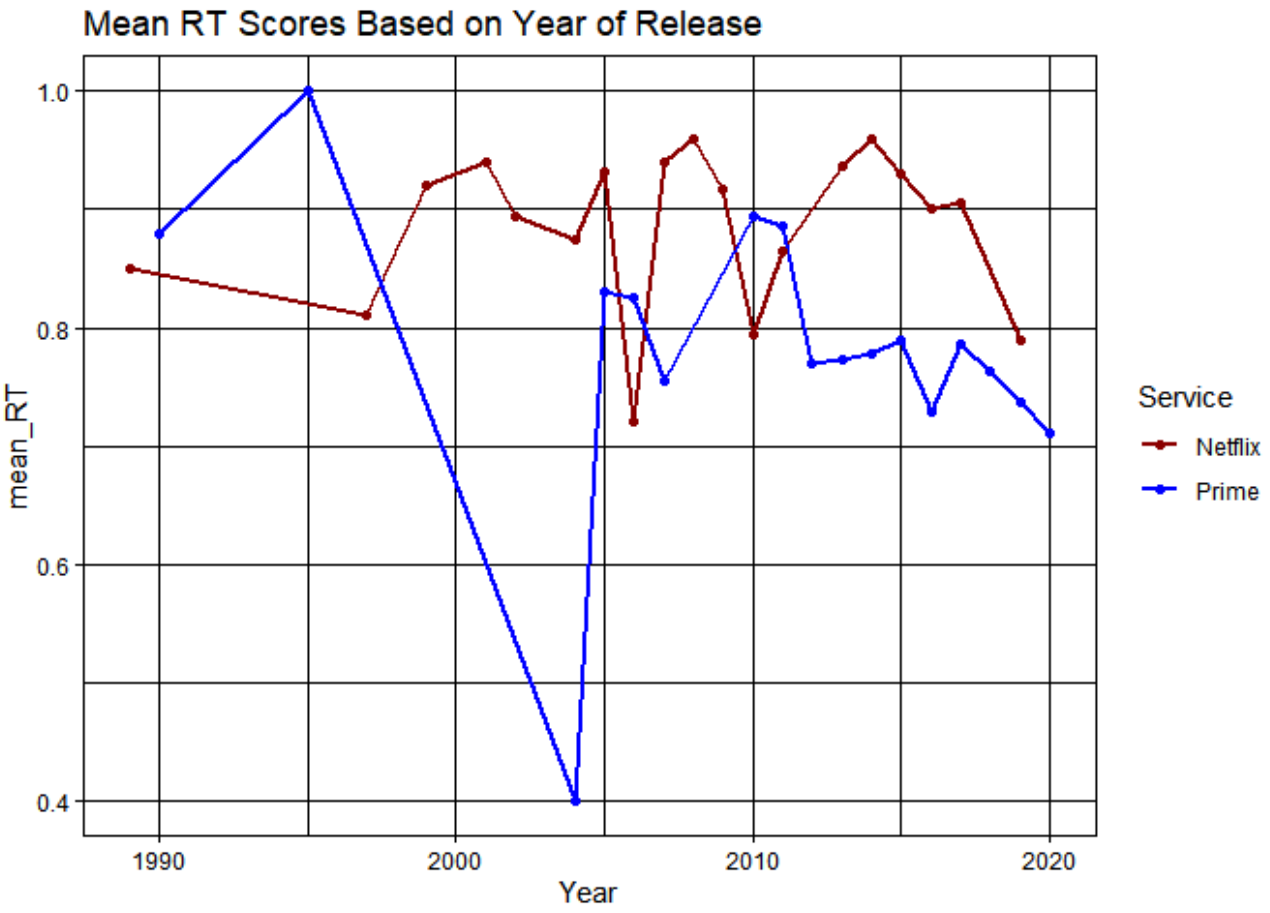
This heatmap is largely uninformative due to the low number of numeric variables in this dataset. Interestingly the ratio correlates more strongly with the RT scores than the IMDB

ratings. This is likely due to the larger degree of variation in the RT scores more heavily affecting the ratio, or it could simply be due to the fact that the RT scores are the numerator of the ratio.

```
fulldata%>%
# Group by the categorical variables to be depicted/explored
  group_by(Year,Service)%>%
# Use the summarize function to output means for each chosen categorical variable
  summarize(mean_RT=mean(RT, na.rm=T))%>%
# Drop Missing values to avoid gaps or errors in the data
  drop_na()%>%
# Use ggplot to plot the desired variables
  ggplot(aes(x=Year,y=mean_RT,color=Service))+
# Adjust the size to increase visibility
  geom_line(size=1,stat = "summary")+
# Set a theme to customize the background
  theme_linedraw() +
# Choose a title
  ggtitle("Mean RT Scores Based on Year of Release") +
# Add points to make individual observations more visible
  geom_point() +
  scale_color_manual(values=c("dark red","blue"))
```

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

## No summary function supplied, defaulting to `mean_se()`
```



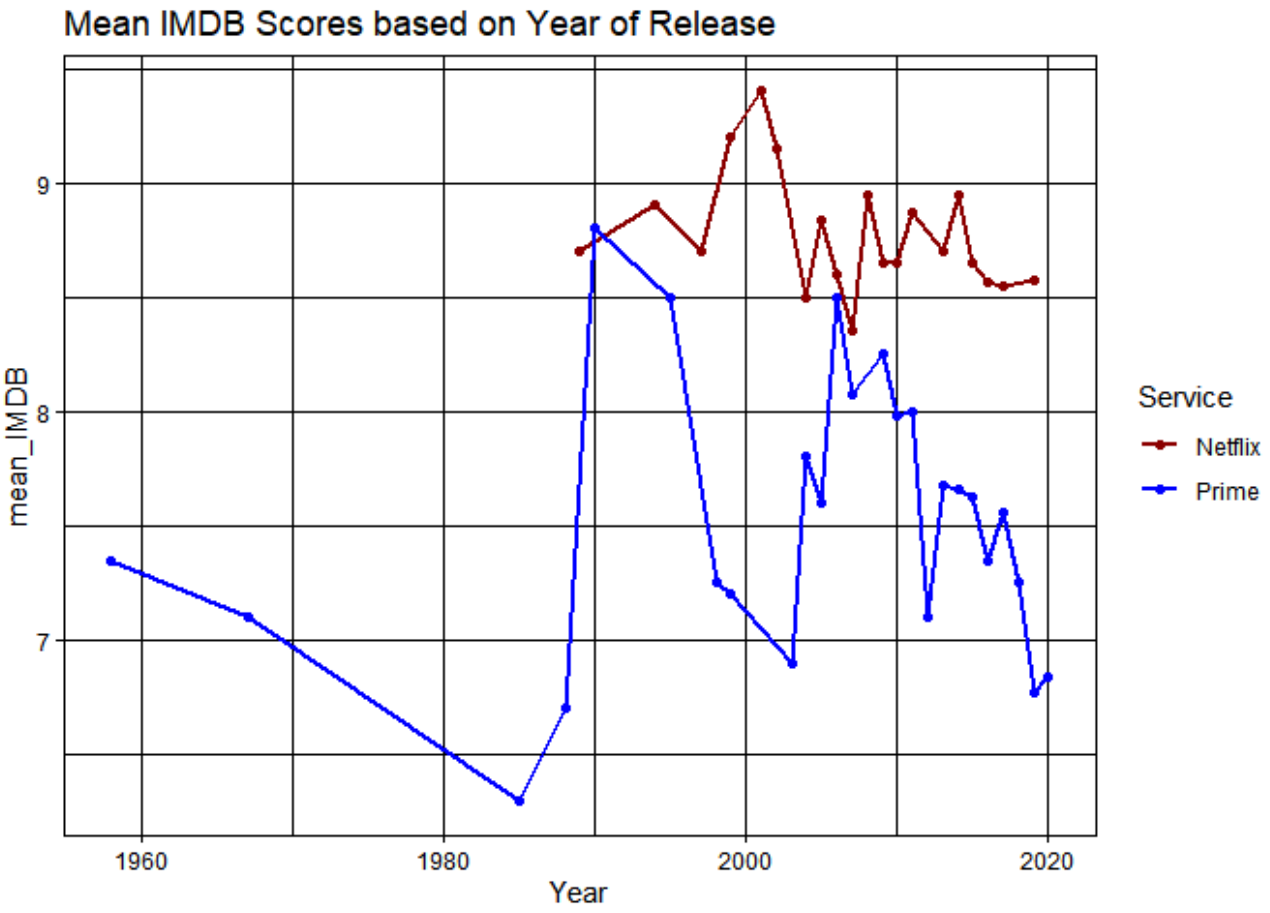
This line graph depicts the average Rotten Tomatoes score for each platform based on the year of the shows original release. Netflix has a more consistent output over time, which suggests that they search for content that has high audience appeal. Earlier analysis showed that there was a greater variation in the Prime Video dataset, so it is likely that this visualization is being skewed negatively for RT scores for the Prime platform due to imperfect sampling.

```
fulldata%>%
# Group by the categorical variables to be depicted/explored
  group_by(Year,Service)%>%
# Use the summarize function to output means for each chosen categorical variable
  summarize(mean_IMDB=mean(IMDB_Rating, na.rm=T))%>%
# Drop Missing values to avoid gaps or errors in the data
  drop_na()%>%
# Use ggplot to plot the desired variables
  ggplot(aes(x=Year,y=mean_IMDB,color=Service))+
# Adjust the size to increase visibility
  geom_line(size=1,stat = "summary")+
# Set a theme to customize the background
  theme_linedraw() +
# Choose a title
  ggtitle("Mean IMDB Scores based on Year of Release") +
# Add points to make individual observations more visible
```

```
  geom_point() +
  scale_color_manual(values=c("dark red","blue"))
```

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

```
## No summary function supplied, defaulting to `mean_se()`
```



By creating a similar graph for the IMDB variable, a direct comparison between the two numeric variables can be made. It is clear that IMDB scores trend significantly lower in the Prime data as compared to the Netflix data. Prime seems to hold a much higher quantity of content from a larger range of time, but the quality of this content is less consistent. Rotten Tomatoes scores seem to have a greater degree of variation in general as the lowest scores are much lower, while the highest are much greater.

Kmeans

```
# Use the function kmeans to find 3 clusters
kmeans1 <- fulldata %>%
```

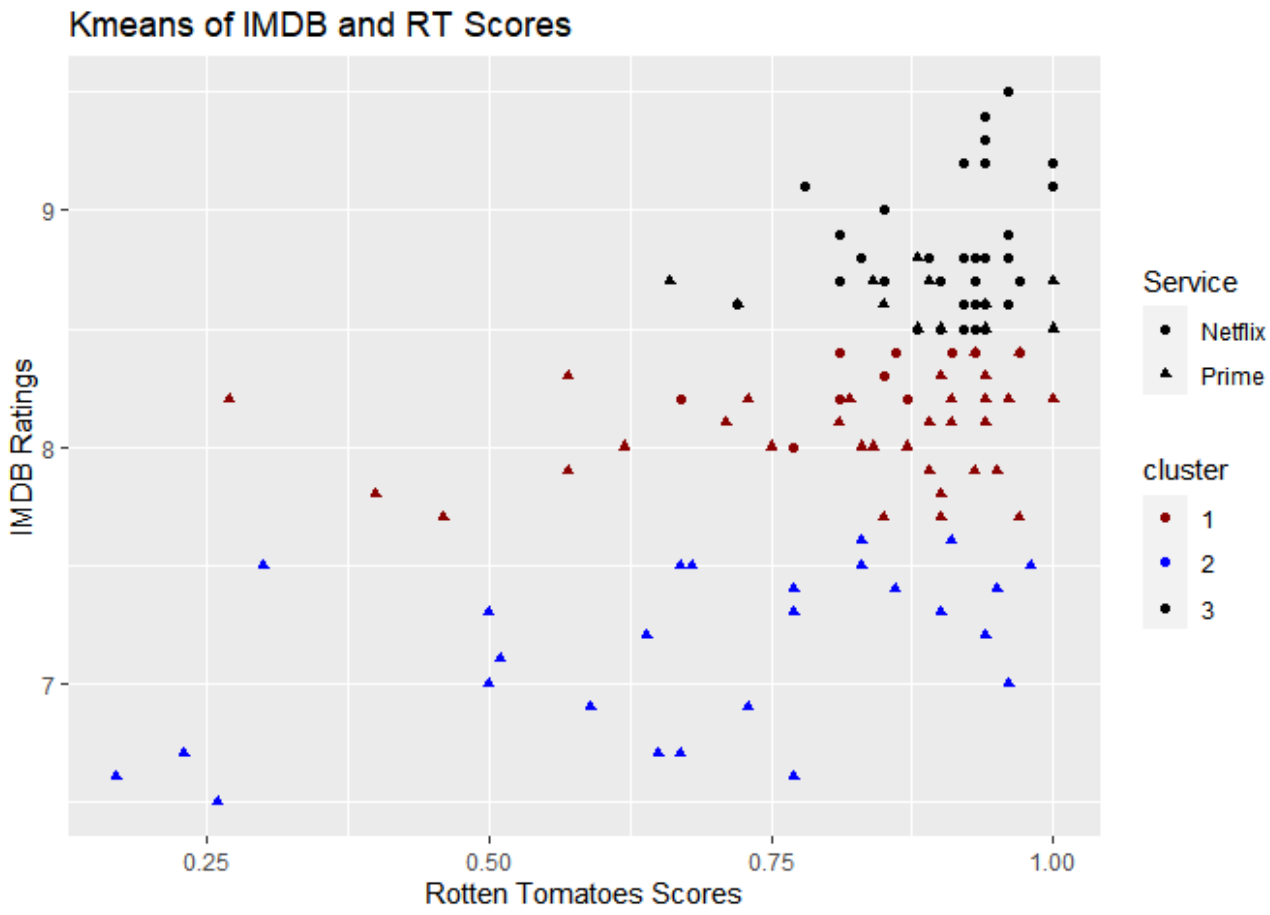
```
# Select the numeric variables being tested
  select(RT,IMDB_Rating)%>%
# Drop NA values to avoid errors
  drop_na()%>%
# Use the kmeans function to calculate the data
  kmeans(3)
kmeans1
```

```
## K-means clustering with 3 clusters of sizes 42, 27, 46
##
## Cluster means:
##          RT IMDB_Rating
## 1 0.8185714    8.107143
## 2 0.6814815    7.162963
## 3 0.9015217    8.776087
##
## Clustering vector:
##   [1] 3 3 3 3 3 3 3 3 3 3 1 3 3 3 3 3 1 3 3 3 3 3 3 1 1 3 3 1 1 3 3 3 3 3 3 3 1
##  [38] 3 3 3 1 1 1 3 1 3 3 2 1 3 2 2 1 2 1 1 2 3 2 1 3 1 3 1 3 1 1 1 1 2 2 2 1 1
##  [75] 2 1 1 3 2 2 1 1 2 1 3 1 1 2 1 2 1 2 1 1 2 2 1 2 1 1 2 2 2 1 2 3 3 2 1 2 1
## [112] 1 2 3 2
##
## Within cluster sum of squares by cluster:
## [1] 3.029771 4.627704 3.276489
##  (between_SS / total_SS =  80.5 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```
# Save cluster assignment as a column in your dataset
kmeansclust <- fulldata %>%
# Select the variables used for the clustering as well as any categorical variable th
  select(RT,IMDB_Rating,Service)%>%
# Remove missing values
  drop_na() %>%
# Add the cluster data from kmean1 as a factor to the data
  mutate(cluster=as.factor(kmeans1$cluster))

# Make a plot of data colored by final cluster assignment
kmeansclust %>%
# Use color to depict the cluster, and shape to depict the other categorical variable
  ggplot(aes(RT,IMDB_Rating,color = cluster,shape=Service)) +
# A scatterplot is the clearest visual to depict cluster data
  geom_point() +
# Add titles and labels
  ggtitle("Kmeans of IMDB and RT Scores") +
```

```
  ylab("IMDB Ratings")+
  xlab("Rotten Tomatoes Scores")+
# Edit colors
  scale_color_manual(values=c("dark red","blue","black"))
```



The kmeans cluster visual exhibits the generally positive correlation between the two numeric variables, but it is clear that low IMDB ratings can still acheive relatively high Rotten Tomatoes scores. Cluster 3 is entirely composed of Prime Video shows, and occupies the lower IMDB values, but the full range of RT scores. The second cluster is predominantly Prime Video shows with a handful of Netflix shows as well, with similar trends to cluster 3 regarding the range of RT scores but about a full point greater IMDB ratings. The first cluster is predominantly Netflix shows with a handful of Prime shows as well, with high values in both numeric variables.