Spencer Phillips

998887568

11/23/15

I did this assignment by myself and developed and wrote the code for each part by myself,

drawing only from class, section, Piazza posts and the Web. I did not use code from

a fellow student or a tutor or any other individual.

STA 141 Assignment 5

**Please note that this entire assignment was done using the first dataset that Duncan gave to the class unless stated otherwise.**

1. The number of actors in the database was found using the argument SELECT COUNT(idactors) FROM actors; and gave back a total of 3500167. The number of movies in the database is 1298737 found through this argument: SELECT COUNT(idmovies) FROM movies;.

2. The time period that the database covers says that the first movie was made in year 1. However, we know that this is impossible. I looked for all data before 1891 since the first movie was apparently made in 1888, there are a number of films that were made in 1888 so I am going to make the lower limit be 1888. Since IMDb also has spots saved for movies that are going to come out in the future, the max year for movies is at 2025. This makes the range of years be from 1888-2025.

3. In this dataset, all of the females are under the classification of NA while males are under the classification of 1. I took all of the NA female actors and put them into their own dataset and found that the length of this was 1238021 dividing this by 3500167 we get a proportion of 35.37%. Male actors are at 2262146 which is 64.63% of all the actors. As we can see there are drastically more male than female actors.

4. For this problem, I had to find out what the different type sets were for the movies dataset. For this I had to then look at what each of the types were and then test them to see what each means, there was NA, 1, 2, and 3. I found that 1 is the classification for movies (after first assuming it was for porn after looking at the first 25 of them…). Going at this, I found the number of movies was 147391 of the 1298737, this makes the proportion of movies to be 11.3% of all entries. The classification of 2 was video games at 1.2%, NA was TV events (like the Emmy’s or other awards shows for example) at 9.3% and 3 was TV shows at 78.1%

5. There are 32 different types of movie genres available. The descriptions are as follows: Documentary, Reality, Horror, Drama, Comedy, Musical, Talk, Mystery, News, Sport, Sci, Romance, Family, Short, Biography, Music, Game, Adventure, Crime, War, Fantasy, Thriller, Animation, Action, History, Adult, Western, Lifestyle, Film, Experimental, Commercial, and Erotica.

6. The above descriptions also can be given a number in a different dataset (from 1 to 32) therefore I am going to loop over all of those in order to get the number back. I did this by using: dbGetQuery(db, 'SELECT idgenres, COUNT(\*) FROM movies\_genres GROUP BY idgenres;'). The top ten genres were as follows, Comedy, Drama, Documentary, Reality, Family, Talk, Animation, Music, Romance, and Game. The following is the proportions of the data that was found: 

This gives us that comedies are the most prolific genre at 18.8%

7. Of the movies that have the word space as keyword, there are 483 different movies with the description of space. This included a range of 53 different years from 1949 to 2016 as follows:



I found this by using the long query that is here: dbGetQuery(db, 'SELECT DISTINCT year  
 FROM keywords, movies\_keywords, movies   
 WHERE keyword = "space"  
 AND keywords.idkeywords = movies\_keywords.idkeywords  
 AND movies.idmovies = movies\_keywords.idmovies')

There were also 3079 top 5 actors for all of the genres with the keyword space. There were a lot due to the fact that all of the 483 entries were TV shows which have casting changes throughout the season. It was found using this combination of many datasets: dbGetQuery(db, 'SELECT DISTINCT lname || ", " || fname   
 FROM keywords, movies\_keywords, movies, acted\_in, actors   
 WHERE keyword = "space"   
 AND keywords.idkeywords = movies\_keywords.idkeywords  
 AND movies.idmovies = movies\_keywords.idmovies  
 AND billing\_position < 6  
 AND movies.idmovies = acted\_in.idmovies  
 AND acted\_in.idactors = actors.idactors;')

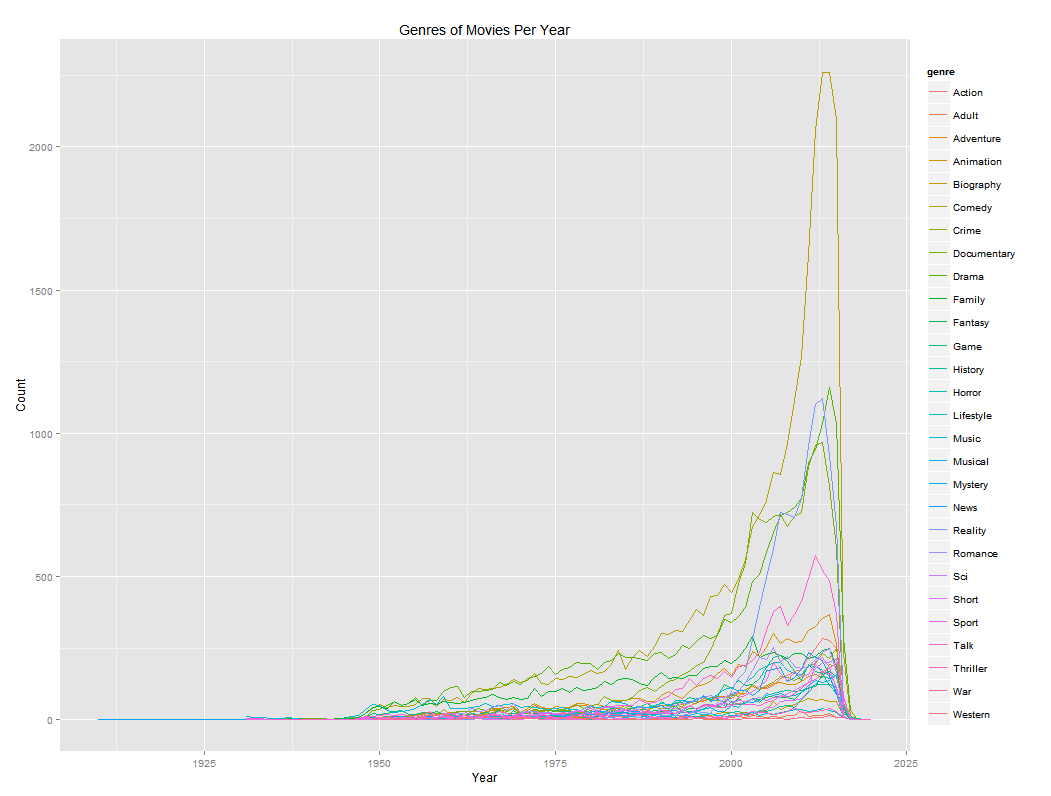
8. I first collected all of the years of movies and their corresponding genre. I then omitted all values that had NAs since they would be irrelevant in the current dataset. Using this SQL function: dbGetQuery(db, 'SELECT COUNT(movies.idmovies) AS count, year, genre

FROM genres, movies\_genres, movies

WHERE genres.idgenres = movies\_genres.idgenres

AND movies\_genres.idmovies = movies.idmovies GROUP BY genre, year')

We get the following graph after putting it through ggplot we get the following graph.  
This graph shows us that the comedy genre was a big hit in the 2010s, outperforming all other genres by a vast margin other than that, most movies usually are dramas with a huge spike in the Reality drama in the early 2000s, probably due to the rise in reality TV shows.



9. The following are the 20 actors in the most movies. Found through the following and then modified in R. (dbGetQuery(db, 'CREATE TEMP TABLE counts AS   
 SELECT lname, fname, COUNT(acted\_in.idactors) AS count   
 FROM acted\_in, actors  
 WHERE acted\_in.idactors = actors.idactors  
 GROUP BY acted\_in.idactors'))

This shows that Alex Trebek, the host of the show Jepoardy was the most prolific actor.

10. For picking the actors with the top billing and with the range of years was given out by the following code. dbGetQuery(db, 'CREATE TEMP TABLE bestcounts AS

SELECT lname, fname, COUNT(acted\_in.idactors) AS count, MIN(year), MAX(year)

FROM acted\_in, actors, movies

WHERE acted\_in.idactors = actors.idactors

AND movies.idmovies = acted\_in.idmovies

AND billing\_position < "4"

GROUP BY acted\_in.idactors')

dbGetQuery(db, 'SELECT \* FROM bestcounts ORDER BY count DESC LIMIT 10')

I was given the following table for the top 10 actors as follows

 Showing that Carol Vorderman was one of the top 10 high listing actors with a career from 1982 to 2013.

11. For this problem, I created a new table called num11 (since this is number 11, I am so clever) to create the sort of database that I would want to draw the data from. I also limited the data to just give movies (type = 1). The argument is as follows.

dbGetQuery(db, 'CREATE TEMP TABLE num11 AS

SELECT fname, lname, COUNT(acted\_in.idactors) AS count, year

FROM acted\_in, actors, movies

WHERE acted\_in.idactors = actors.idactors

AND movies.idmovies = acted\_in.idmovies

AND movies.type = 1

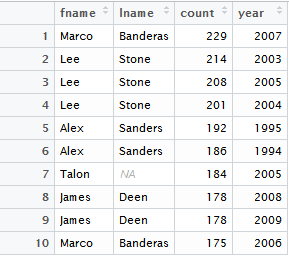
GROUP BY year, acted\_in.idactors')

I used this and then excluded movies with a year before 1878 since around that time the first movie was made and some values were mistakenly recorded for a time before that. Due to this it would cause errors in my data. I then ordered this by count and took the first 10 values of the data. dbGetQuery(db, 'SELECT DISTINCT \* FROM num11

WHERE year > "1878"

ORDER BY count DESC

LIMIT 10')

This gave me the following table. 

12. The Actors with the most aliases were found using: dbGetQuery(db, 'SELECT fname, lname, count(aka\_names.idactors) AS count   
 FROM aka\_names, actors   
 WHERE aka\_names.idactors = actors.idactors  
 GROUP BY aka\_names.idactors ORDER BY count DESC LIMIT 10')

The table is as follows:

This shows us that Jess Franco is the leader in aliases with a 76 with there being a large drop off after that as we go down the list of the 10 people.

13. For this problem I used the smaller dataset that Duncan gave us with the data starting in 2010 (non optimized). For this network I started with the actor Joseph Gordon Levitt. I found out his person\_id value using SQL and then made new tables using SQL that only included movies and not TV shows. I then moved these tables into R so I can modify them. EX:   
dbGetQuery(q13db,'CREATE TABLE title AS

SELECT \* FROM title2

WHERE kind\_id = 1;')

After moving the data into the R environment, I created a function to get movie ID numbers and another function to get the top 5 actor IDs from the movies (ranked on listing for the movie). For the second level, the loop took an incredibly long time to go through, so long that it was unable to be done due to the indefinite amount of time to get through it. I also attempted to make a network but was unsuccessful seeing as how I have no idea how to make it and things online are not very helpful at explaining things to me. I was able to find out that Joseph Gordon-Levitt was directly connected to 22 movies with 64 other top 5 actors who are then connected to 722 other movies, meaning this would lead to be a very large chart in the end. I suppose I will figure out how to actually make a network at a later point perhaps, for I did give it my best effort in the end...

The afore mentioned data was found using the following functions:   
getmovielist = function(actorid)

{movieIDs = castinfo[castinfo$person\_id == actorid,]

movieIDs = movieIDs[!duplicated((movieIDs$movie\_id)),]

final = movieIDs$movie\_id

return(final)}  
and: getactors = function(movieID)

{otheractors = castinfo[castinfo$movie\_id == movieID & castinfo$nr\_order < 5,]

theids = completeFun(otheractors, "nr\_order")

actors = theids$person\_id

return(actors)}