STA141_Assignment5

Chad Pickering

Monday, November 30, 2015

Non-copying statement:

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.

Signed: ____

Resources: TutorialsPoint SQL tutorials, w3schools SQL tutorials, various YouTube SQL tutorials, Stack Overflow, Piazza forums, Office Hours, IMDB statistics and occasional references

```
library(RSQLite)
```

```
## Loading required package: DBI
```

```
imdb <- dbConnect(SQLite(),
                  "C:/Users/cpickering/Syncplicity Folders/ChadSync/STATISTICS/STA141/imdb_data")
```

Preface:
I used the **old (original) database** to retrieve all results. I define "movies" as any program, regardless of type, unless otherwise noted in my analysis or asked of me in the question. Despite the instructions, Duncan indicated on Piazza multiple times that we can define what we mean by "movies". This being said, realize that some of my answers may differ from a "master key" or a majority of other students' work; in my analysis for each question, I am clear about what I am defining, strategies I am using to query, and why some outputs have strange or anomalous characteristics. Most of these anomalies can be attributed to the fact that this database is flawed in many ways, which is why an alternative database was made available. I felt that the old database was more straightforward in terms of organization and content, and I wanted such an environment in which to learn SQL, the purpose of this assignment.

1. How many actors are there in the database? How many movies? The number of actors is the equivalent to the count of the rows in the actors table. A similar strategy is used for movies. Each ID is distinct already, so there is no need to count distinct IDs. To re-emphasize, I am considering all people in the actors table to be "actors" and all movies in the movies table to be "movies", despite the type (more on this in #4 analysis). When querying, I get 3500167 actors and 1298737 movies. These are quite a bit lower from the official statistics from IMDB - this original database has unknown faults, including exclusions and misclassifications. The integrity of the database is not my responsibility, so I will retrieve results with the methods I learned throughout this process.

```
dbGetQuery(imdb, "SELECT COUNT(*) AS total_actors FROM actors")
```

```
##   total_actors
## 1      3500167
```

```
dbGetQuery(imdb, "SELECT COUNT(*) AS total_movies FROM movies")
```

```
##   total_movies
## 1      1298737
```

2. What time period does the database cover? Year is contained within the movies table, so I queried for the minimum, maximum, and count of the years (meaning the amount of years reported - and most are). The minimum year is found to be 1 - upon researching this anomaly, I found that dozens, if not hundreds of titles are assigned a year less than the true minimum year for IMDB - 1874 - because they were either originally misclassified upon entry or scraped incorrectly. Most of these titles, interestingly, are either adult or independent films. The maximum is found to be 2025 - this is not a misclassificaton; some films are in pre-production or in planning stages as far as 10 years in advance of release.

```
dbGetQuery(imdb, "SELECT MIN(year)||'-'||MAX(year) AS year_range,
                COUNT(year) AS years_reported
           FROM movies")
```

```
##   year_range years_reported
## 1     1-2025        1276381
```

For reference, I also went ahead and explored the small subset of movies that have alternative titles because there are multiple international editions or because the names changed over time. In this subset, the range was within reason, 1924-2018, but this is by no means a random subset, so this range does not represent all of the data.

```
dbGetQuery(imdb, "SELECT MIN(year)||'-'||MAX(year) AS intntl_range,
                COUNT(year) AS subset_size
           FROM aka_titles")
```

```
##   intntl_range subset_size
## 1    1924-2018       27212
```

3. What proportion of the actors are female? male? The output of this question is the main reason why an alternative database was posted. The gender has two levels, "NA" and "1" - upon research using a large set of both male and female actors, I was able to conclude that the "1" level corresponds to males and the "NA" level corresponds to females. About 64.6% of actors are reported to be males and the remaining 35.4% are reported to be female. This makes sense because over the entire lifetime of film, males were overwhelmingly prominent in the industry for quite a while. This calculation required a nested query to count the total number of actors for the denominator, and also a float value of 100.0 in the numerator so the proportion would not be rounded to an integer value, either 1 or 0, respectively.

```
dbGetQuery(imdb, "SELECT gender,
                COUNT(*) * 100.0 / (SELECT COUNT(*) FROM actors) AS gender_percentage
           FROM actors
           GROUP BY gender")
```

```
##   gender gender_percentage
## 1     NA          35.37034
## 2      1          64.62966
```

4. What proportion of the entries in the movies table are actual movies and what proportion are television series, etc.? The type of the movie is of interest here; over 78% of the entries are of type "3", where the other three categories are not as prominent whatsoever. On Piazza (@1312), it was revealed that "NA" means "made for TV", "1" means "made for video or direct-to-video release", "2" means "video game", and "3" means "movie". However, after much research and testing individual titles, I was able to determine that these labels mean nothing. It is a light categorical "reference" point that may lead one to find what they are looking for, but these “types” are not consistent. This is why I made the decision to include any entry as a “movie”, so as to not exclude any movies that were classified in the categories not designated for strictly “movies”. Furthermore, there seems to be no “type” validation other than if a movie has a series ID, then that entry is a member of a “series”. There is no other method of validation in this specific database, so I feel that my inclusion of every entry as a “movie” is safe, and will still allow me to get at least semi-accurate results for most questions. If the goal of this assignment was to get extremely accurate results, I would be using the alternative database - it is clear that it contains much more comprehensive and accurate information (but it is huge and I was not able to attain it). However, the goal of this assignment is to learn SQL, and I feel that my methodology is sufficient.

```
dbGetQuery(imdb, "SELECT type, COUNT(*) AS total_type,
                COUNT(*) * 100.0 / (SELECT COUNT(*) FROM movies) AS type_percentage
           FROM movies
           GROUP BY type")
```

```
##   type total_type type_percentage
## 1   NA     121217        9.333452
## 2    1     147391       11.348795
## 3    2      15384        1.184535
## 4    3    1014745       78.133217
```

5. How many genres are there? What are their names/descriptions? There are 32 total genres in the database, containing a comprehensive list of possible genres. There are no alternative tables or columns that contain a “description” of a genre beyond the mere categorical title.

```
dbGetQuery(imdb, "SELECT COUNT(*) AS total_genres
           FROM genres")
```

```
##   total_genres
## 1           32
```

```
dbGetQuery(imdb, "SELECT genre AS genre_names
           FROM genres")
```

```
##     genre_names
## 1   Documentary
## 2       Reality
## 3        Horror
## 4         Drama
## 5        Comedy
## 6       Musical
## 7          Talk
## 8       Mystery
## 9          News
## 10        Sport
## 11          Sci
## 12      Romance
## 13       Family
## 14        Short
## 15    Biography
## 16        Music
## 17         Game
## 18    Adventure
## 19        Crime
## 20          War
## 21      Fantasy
## 22     Thriller
## 23    Animation
## 24       Action
## 25      History
## 26        Adult
## 27      Western
## 28    Lifestyle
## 29         Film
## 30 Experimental
## 31   Commercial
## 32      Erotica
```

6. List the 10 most common genres of movies, showing the number of movies in each of these genres. Another flaw in this original database is the fact that only a small percentage of entries actually have a genre despite the fact that it claims that about 78% of the entries are "movies". To be exact, 149335 entries in this database have been assigned a genre - this is likely not a random subset, so it does not represent the population of all movies. Out of these, the most common genre is comedy, with over 28000 movies; drama, documentary, and reality follow.

```
dbGetQuery(imdb, "SELECT genres.genre, COUNT(movies_genres.idgenres) AS genre_frequency
           FROM movies_genres, genres
           WHERE movies_genres.idgenres = genres.idgenres
           GROUP BY movies_genres.idgenres
           ORDER BY COUNT(movies_genres.idgenres) DESC
           LIMIT 10")
```

```
##          genre genre_frequency
## 1       Comedy           28152
## 2        Drama           20149
## 3  Documentary           14934
## 4      Reality           10360
## 5       Family            8915
## 6         Talk            7949
## 7    Animation            6797
## 8        Music            5222
## 9      Romance            4679
## 10        Game            4367
```

7. Find all movies with the keyword 'space'. How many are there? What are the years these were released? and who were the top 5 actors in each of these movies? I was extremely direct and searched for only those movies with the keyword 'space' - no 'spaceman' or 'spacesuit' or anything that would be grabbed with 'LIKE space%'. Upon looking at the titles of the movies with the keyword 'space', I found that some very popular titles were not included that one would expect to find, such as the 6 Star Wars movies, some of the Star Trek films, and some of the recent space-related films, like Interstellar. This comes back to the concept that this original database has issues in every corner. I also had to count the distinct titles or else I would be including repeated entries - if this were the case, I would find almost 500 titles with the keyword 'space'. Using DISTINCT, I find 273.

```
dbGetQuery(imdb, "SELECT COUNT(DISTINCT(movies.title)) AS freq_space
           FROM movies, movies_keywords, keywords
           WHERE movies_keywords.idmovies = movies.idmovies
           AND movies_keywords.idkeywords = keywords.idkeywords
           AND keywords.keyword = 'space'")
```

```
##   freq_space
## 1        273
```

For each year, then, I found the number of 'space' movies released. This means that I, by default, displayed all years that at least one space movie was released. All years with a count of zero are not displayed. I limited the output to the most recent 20 years for convenience.

```
dbGetQuery(imdb, "SELECT year, COUNT(DISTINCT(movies.title)) AS space_freq
           FROM movies, movies_keywords, keywords
           WHERE movies_keywords.idmovies = movies.idmovies
           AND movies_keywords.idkeywords = keywords.idkeywords
           AND keywords.keyword = 'space'
           GROUP BY year
           ORDER BY year DESC
           LIMIT 20")
```

```
##    year space_freq
## 1  2016          3
## 2  2015          2
## 3  2014         10
## 4  2013          6
## 5  2012          4
## 6  2011          6
## 7  2010          7
## 8  2009          8
## 9  2008          8
## 10 2007          5
## 11 2006          5
## 12 2005          6
## 13 2004         12
## 14 2003          6
## 15 2002          8
## 16 2001          8
## 17 2000          8
## 18 1999         13
## 19 1998         11
## 20 1997          4
```

For each of the distinct movies with the keyword 'space', I found the actors and actresses within the top 5 billing positions. This query sometimes picks up more than one actor per billing position because some of the "movies" extracted are TV shows/series with multiple episodes, and over the course of the TV show/series airing, several actors have been, for example, in the third billing position. So all will appear as such below. This also means that recurring actors can be labeled as more than one billing position. I ordered by the ID, and then established a secondary ordering scheme on the billing position within each ID. Then, if there are duplicate identical billing positions for one title, the names are automatically sorted by last name.

```
dbGetQuery(imdb, "SELECT DISTINCT(movies.idmovies) AS movie_id, movies.title,
              actors.lname||', '||actors.fname AS actor_name,
              acted_in.billing_position AS bill_posn
           FROM actors, acted_in, movies, movies_keywords, keywords
           WHERE billing_position <= 5
           AND actors.idactors = acted_in.idactors
           AND acted_in.idmovies = movies.idmovies
           AND movies.idmovies = movies_keywords.idmovies
           AND movies_keywords.idkeywords = keywords.idkeywords
           AND keywords.keyword = 'space'
           ORDER BY movies.idmovies ASC, acted_in.billing_position ASC
           LIMIT 30")
```

```
##    movie_id    title          actor_name bill_posn
## 1       197 Farscape        Browder, Ben         1
## 2       197 Farscape      Black, Claudia         2
## 3       197 Farscape       Hey, Virginia         3
## 4       197 Farscape     Simcoe, Anthony         3
## 5       197 Farscape        Edgley, Gigi         4
## 6       197 Farscape       Hey, Virginia         4
## 7       197 Farscape     Simcoe, Anthony         4
## 8       197 Farscape   Cook, Alyssa-Jane         5
## 9       197 Farscape        Edgley, Gigi         5
## 10      197 Farscape         Fox, Alison         5
## 11      197 Farscape          Mara, Mary         5
## 12      197 Farscape    Mendoza, Natalie         5
## 13      197 Farscape     Milliken, Angie         5
## 14      197 Farscape        Raison, Kate         5
## 15      197 Farscape    Szubanski, Magda         5
## 16      197 Farscape          Adam, John         5
## 17      197 Farscape       Clayton, John         5
## 18      197 Farscape de Montemas, Damian         5
## 19      197 Farscape      Getley, Adrian         5
## 20      197 Farscape       Goddard, Paul         5
## 21      197 Farscape      Haywood, Chris         5
## 22      197 Farscape        Leyden, Paul         5
## 23      197 Farscape        McCord, Kent         5
## 24      197 Farscape       Muldoon, Rhys         5
## 25      197 Farscape       Pygram, Wayne         5
## 26      680     Lexx       Downey, Brian         1
## 27      680     Lexx      Habermann, Eva         2
## 28      680     Lexx      Seeberg, Xenia         2
## 29      680     Lexx    McManus, Michael         3
## 30      680     Lexx     Aaltonen, Minna         4
```

8. Has the number of movies in each genre changed over time? Plot the overall number of movies in each year over time, and for each genre. The number of movies in each genre can be shown using multiple time series. The query to generate all year/genre combinations and corresponding counts for each is below.

```
num_genres_movies <- dbGetQuery(imdb, "SELECT year, genres.genre,
                                COUNT(genres.genre) AS genre_freq_per_year
```

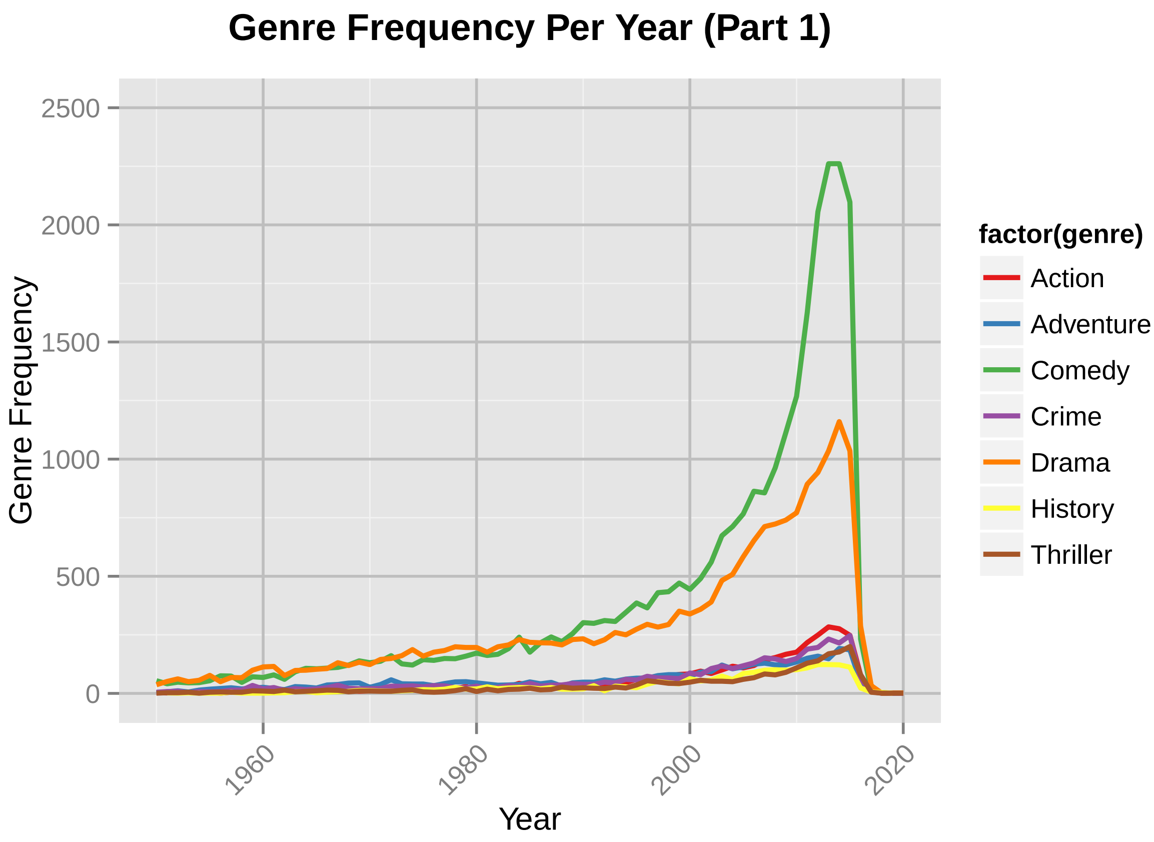
```
           FROM movies, movies_genres, genres
           WHERE movies_genres.idgenres = genres.idgenres
           AND movies.idmovies = movies_genres.idmovies
           AND year IS NOT NULL
           GROUP BY movies.year, genres.genre
           ORDER BY movies.year DESC")
```

The code below to create the subsets for my four plots showing seven time series each is **absolutely clear** as to what I am doing. While I know that this is "repeating myself" and that I could use a function and an sapply() to generate create these four lines of code, I find that effort rather unnecessary. Now, if I had 10 or more plots like this, I would certainly use the function and sapply(), but I feel that what I am doing here is not a problem.

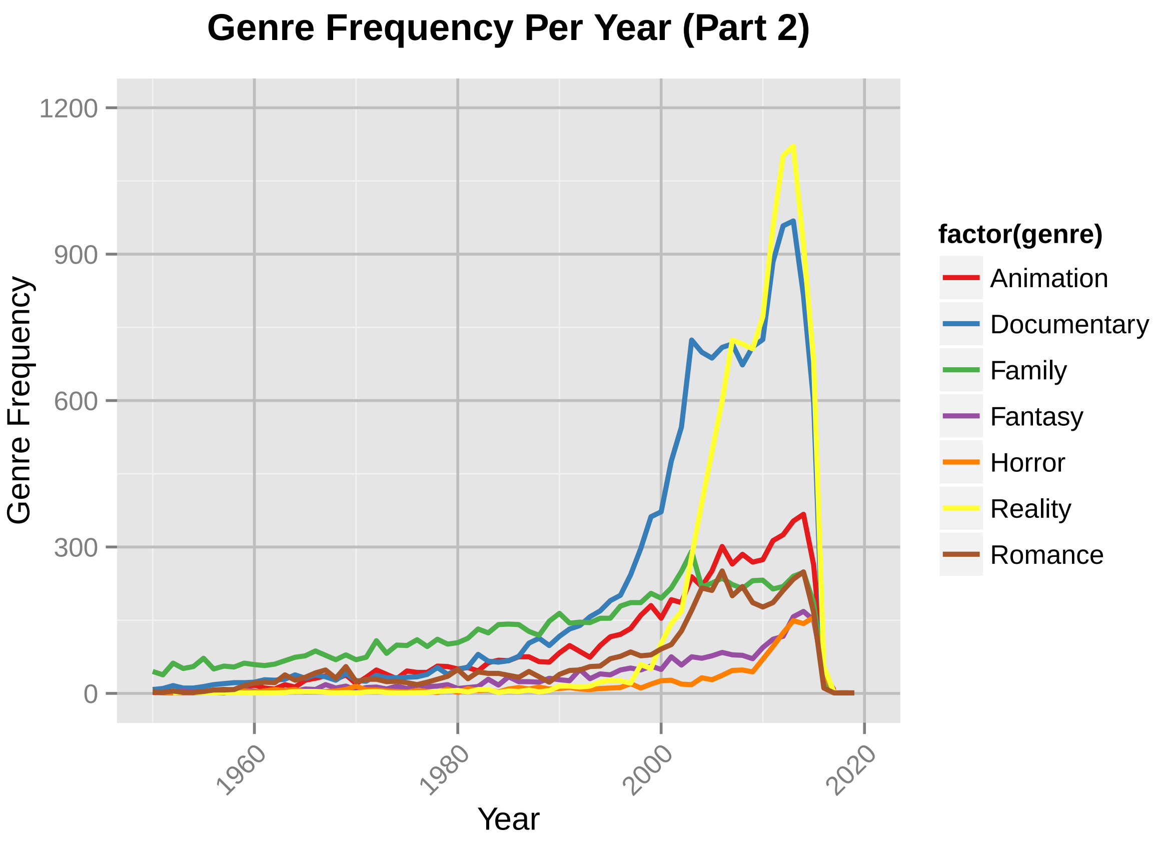
```
all_genres <- unique(num_genres_movies$genre)

genres_one <- num_genres_movies[num_genres_movies$genre %in% all_genres[1:7], ]

genres_two <- num_genres_movies[num_genres_movies$genre %in% all_genres[8:14], ]

genres_three <- num_genres_movies[num_genres_movies$genre %in% all_genres[15:21], ]

genres_four <- num_genres_movies[num_genres_movies$genre %in% all_genres[22:28], ]
```

The following four plots contain seven genres each in order to limit the number of plots (we would not want 28 plots, one for each genre), but also to increase cleanliness and understanding of the data (all 28 genres on one plot would be incomprehensible and essentially useless). For more popular genres like comedy, drama, reality, and documentary, the exponential rate at which the frequency increases is quite obvious; less frequent genres are less obvious visually, but still follow an approximately exponential increase in frequency. This is due to technology advancements, expansion in the market including independent films, and the demand of viewers/consumers. After 2015 (present day), the number of films per genre drops off rapidly because there are only a limited amount of films known to be in pre-production or in preparatory stages at the current time. I can think of one way to improve my set of plots, though: I could have ordered all genre's lifetime maximas in descending order, and grouped by seven from that order of genres. This way, the y limits would be fitting for each group, e.g. there would not be some time series lines very close to the x axis and some with dramatic movement up to the maximum y value; all time series with similar lifetime maximas would be grouped together. However, the reason why I kept them the way they are is because the colors may overlap because each genre's behavior over time is rather similar. I think the current display is sufficient; I just wanted to suggest an alternative idea.

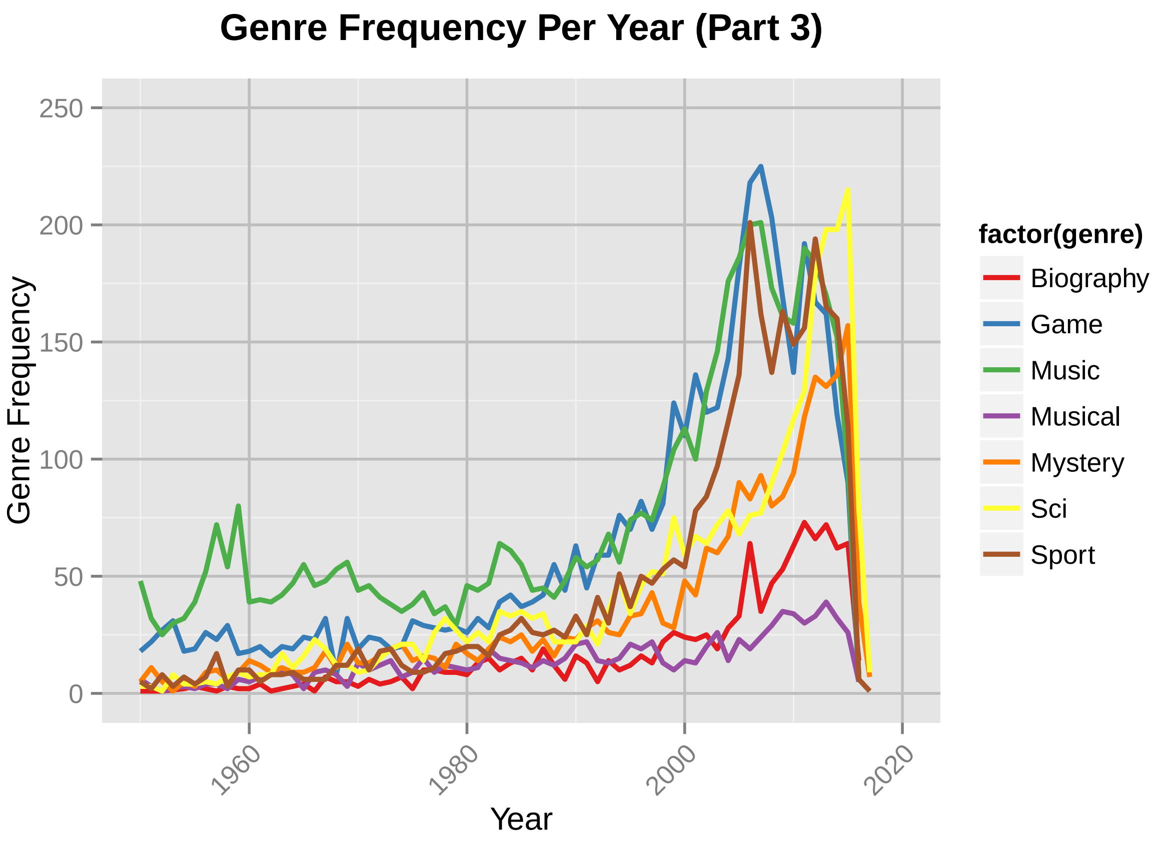
```
library(ggplot2)

ggplot(genres_one, aes(year, genre_freq_per_year, color = factor(genre)))+
  geom_line(size = 0.9)+
  scale_color_brewer(palette = "Set1")+
  labs(x = "Year", y = "Genre Frequency",
       title = "Genre Frequency Per Year (Part 1)")+
  theme(plot.title = element_text(size = 14, face = "bold", vjust = 2),
        panel.grid.major = element_line(colour = "gray", size = .5),
        axis.text.x = element_text(angle = 45, hjust = 1))+
  scale_x_continuous(limits = c(1950, 2020))+
  scale_y_continuous(limits = c(0, 2500), breaks = seq(0, 2500, 500))
```



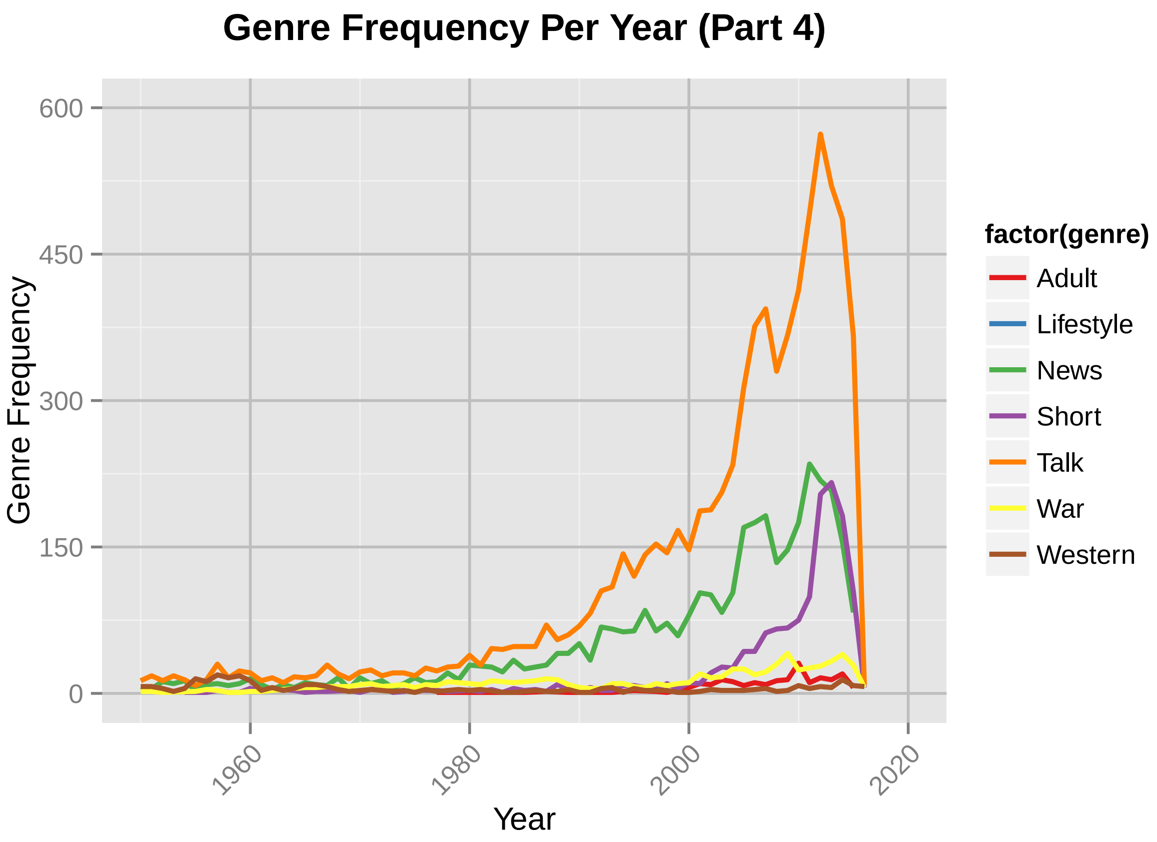
```
ggplot(genres_two, aes(year, genre_freq_per_year, color = factor(genre)))+
  geom_line(size = 0.9)+
  scale_color_brewer(palette = "Set1")+
  labs(x = "Year", y = "Genre Frequency",
       title = "Genre Frequency Per Year (Part 2)")+
  theme(plot.title = element_text(size = 14, face = "bold", vjust = 2),
        panel.grid.major = element_line(colour = "gray", size = .5),
        axis.text.x = element_text(angle = 45, hjust = 1))+
  scale_x_continuous(limits = c(1950, 2020))+
  scale_y_continuous(limits = c(0, 1200), breaks = seq(0, 1200, 300))
```



```
ggplot(genres_three, aes(year, genre_freq_per_year, color = factor(genre)))+
  geom_line(size = 0.9)+
  scale_color_brewer(palette = "Set1")+
  labs(x = "Year", y = "Genre Frequency",
       title = "Genre Frequency Per Year (Part 3)")+
  theme(plot.title = element_text(size = 14, face = "bold", vjust = 2),
        panel.grid.major = element_line(colour = "gray", size = .5),
        axis.text.x = element_text(angle = 45, hjust = 1))+
  scale_x_continuous(limits = c(1950, 2020))+
  scale_y_continuous(limits = c(0, 250), breaks = seq(0, 250, 50))
```



```
ggplot(genres_four, aes(year, genre_freq_per_year, color = factor(genre)))+
  geom_line(size = 0.9)+
  scale_color_brewer(palette = "Set1")+
  labs(x = "Year", y = "Genre Frequency",
      title = "Genre Frequency Per Year (Part 4)")+
  theme(plot.title = element_text(size = 14, face = "bold", vjust = 2),
       panel.grid.major = element_line(colour = "gray", size = .5),
       axis.text.x = element_text(angle = 45, hjust = 1))+
  scale_x_continuous(limits = c(1950, 2020))+
  scale_y_continuous(limits = c(0, 600), breaks = seq(0, 600, 150))
```



9. Who are the actors that have been in the most movies? List the top 20. Since I defined "movies" as any entry in the movies table in order to not exclude any actual movies, I searched for the top 20 actors in any movie of any type. I ended up getting almost exclusively talk show hosts and game show hosts. Some have thousands of appearances, although I do believe, with a quick sample query, that some of these are duplicates, or are just irrelevant. By actually searching IMDB online, I can confirm that some entries are repeated - this database is honestly not accurate whatsoever. If I had a machine at my disposal that could successfully download the larger more accurate database, I would do it in a heartbeat.

```r
dbGetQuery(imdb, "SELECT acted_in.idactors, COUNT(acted_in.idactors),
               actors.fname, actors.lname
          FROM actors, acted_in
          WHERE acted_in.idactors = actors.idactors
          GROUP BY actors.idactors
          ORDER BY COUNT(acted_in.idactors) DESC
          LIMIT 20")
```

```
##    idactors COUNT(acted_in.idactors)              fname      lname
## 1   3284305                     7259               Alex     Trebek
## 2   1963709                     7233             Johnny    Gilbert
## 3   1363105                     6898                Bob     Barker
## 4   3007716                     6278                Pat      Sajak
## 5   1191284                     6219              Vanna      White
## 6   1164869                     5740              Carol  Vorderman
## 7    863140                     5540             Janice Pennington
## 8   2401656                     5352                Jay       Leno
## 9   2745295                     4971             Johnny      Olson
## 10  2407136                     4740              David  Letterman
## 11  3413201                     4605            Richard   Whiteley
## 12  2727248                     4389 O'Donnell, Charlie       <NA>
## 13  3400320                     4272              Frank     Welker
## 14  3079126                     4215               Paul    Shaffer
## 15  2724638                     4040     O'Brien, Conan       <NA>
## 16  2955614                     4013                Rod      Roddy
## 17   509347                     3994             Helena     Isabel
## 18  1832366                     3914                Lus Esparteiro
## 19  1569250                     3803             Manuel     Cavaco
## 20   614277                     3759    Katherine Kelly       Lang
```

R Version: In lecture 11/24, Duncan said that we can use dbGetQuery() to join all of the tables we need for the R portion of 9-12. I chose only the columns that I needed because if I chose everything, the resulting data frame would be too big and I was not able to store it properly. This is all valid for my solutions in R for 9 through 12 - apologies if this is not what was wanted, but the data frames were just way too big. In addition, I was rushed through the R sections, and would have done a more complete job time permitting. I was able to create a sorted table to display the same results as the SQL query did. However, there is a very compelling difference between these two results. In the SQL result, the present in two of the last names (both having “O”’ interestingly - seems like the || operator cannot handle the apostrophe) is preserved. However, in the R result, the was incompatible and all names with an apostrophe, and therefore an in the last name are completely thrown out. I think this is because SQL is “literal” and R simply did not know what to do with such a syntax and ignored them. This means that in the R output, 18 of the 20 names that were in the SQL output are the same. However, two more names are added in at the end that had the next two most frequent IDs. I tried gsub(), but when the is removed in the data frame, nothing replaces it; the concatenation cannot be done consistently when some last name elements are missing.

```
q9_df <- dbGetQuery(imdb, "SELECT acted_in.idmovies AS idmovies,
                          acted_in.idseries AS idseries,
                          actors.idactors AS idactors, lname||', '||fname AS names
           FROM acted_in, actors
           WHERE acted_in.idactors = actors.idactors")

head(sort(table(na.omit(q9_df$names)), decreasing = TRUE), 20)
```

```
##
##           Trebek, Alex        Gilbert, Johnny            Barker, Bob
##                   7259                   7234                   6900
##             Sajak, Pat           White, Vanna       Vorderman, Carol
##                   6278                   6219                   5740
##     Pennington, Janice              Leno, Jay          Olson, Johnny
##                   5540                   5352                   4971
##       Letterman, David      Whiteley, Richard          Welker, Frank
##                   4740                   4605                   4272
##          Shaffer, Paul             Roddy, Rod         Isabel, Helena
##                   4215                   4013                   3994
##        Esparteiro, Lus         Cavaco, Manuel  Lang, Katherine Kelly
##                   3914                   3803                   3759
## Cerdeira, Antnio Pedro            Lima, Pedro
##                   3644                   3513
```

10. Who are the actors that have had the most number of movies with "top billing", i.e., billed as 1, 2 or 3? For each actor, also show the years these movies spanned? I restricted the query to only consider entries with a billing_position of 1, 2, or 3. Then, I counted these occurances per actor and limited this to the top 10. I used the handy || operator again to present the years these programs/movies spanned. Again, most of these people are talk show hosts because of my definition of "movie". I left the names separate so Conan O'Brien was properly represented.

```
dbGetQuery(imdb, "SELECT actors.fname, actors.lname,
                  COUNT(acted_in.billing_position IN (1,2,3)) AS num_programs,
                  MIN(year)||'-'||MAX(year) AS years_spanned
           FROM actors, acted_in, movies
           WHERE actors.idactors = acted_in.idactors
           AND acted_in.idmovies = movies.idmovies
           AND acted_in.billing_position IN (1, 2, 3)
           GROUP BY actors.idactors
           ORDER BY COUNT(acted_in.billing_position IN (1, 2, 3)) DESC
           LIMIT 10")
```

```
##              fname     lname num_programs years_spanned
## 1            Carol Vorderman         4655     1982-2013
## 2          Richard  Whiteley         4510     1982-2003
## 3              Jon   Stewart         2604     1987-2012
## 4              Edd      Hall         2236     1983-2007
## 5              Jay      Leno         2194     1962-2015
## 6        Alexandra Lencastre         2115     1990-2015
## 7   O'Brien, Conan      <NA>         1839     1993-2015
## 8         Fernanda   Serrano         1736     1995-2015
## 9            Craig  Ferguson         1625     1990-2016
## 10          Rogrio    Samora         1574     1987-2015
```

R Version: Just like in #9 or the following R examples, I get the same result excluding any name with an apostrophe, and including the next highest counts added in at the rear of the table. I subsetted all actors where their position in a film was in the top 3 billing positions as well.

```
q10_df <- dbGetQuery(imdb, "SELECT actors.idactors AS idactors,
                            lname||', '||fname AS names, year,
                            acted_in.billing_position AS billing_position,
                            movies.idmovies AS idmovies
           FROM movies, acted_in, actors
           WHERE acted_in.idactors = actors.idactors
           AND acted_in.idmovies = movies.idmovies")

top_bill_pos <- q10_df[q10_df$billing_position %in% 1:3,]

head(sort(table(na.omit(top_bill_pos$names)), decreasing = TRUE), 10)
```

```
## 
##     Vorderman, Carol    Whiteley, Richard         Stewart, Jon 
##                 4655                 4510                 2605 
##            Hall, Edd            Leno, Jay Lencastre, Alexandra 
##                 2236                 2194                 2115 
##    Serrano, Fernanda      Ferguson, Craig       Samora, Rogrio 
##                1736                 1625                 1574
##        Kimmel, Jimmy
##                1547
```

11. Who are the 10 actors that performed in the most movies within any given year? What are their names, the year they starred in these movies and the names of the movies? Because of my broad definition of "movie", I get more of the same game show and talk show hosts here as well. However, I am completely convinced that this query does not give me what I want it to give me. Beyond the fact that for programs with multiple episodes, IMDB defaults to the premiere year (in my experience), the counts for actor IDs and movie IDs with this same query are the same, which is rather curious. I am quite convinced that the function MID() may help in displaying all names of the movies, but I did not have time to implement that part of the query.

```
dbGetQuery(imdb, "SELECT movies.year, actors.lname||', '||actors.fname AS name,
           COUNT(movies.idmovies) as num_prgms
           FROM actors, acted_in, movies
           WHERE acted_in.idactors = actors.idactors
           AND acted_in.idmovies = movies.idmovies
           GROUP BY movies.year, actors.idactors
           ORDER BY COUNT(movies.idmovies) DESC
           LIMIT 10")
```

```
##    year               name num_prgms
## 1  1984       Trebek, Alex      7024
## 2  1984    Gilbert, Johnny      7021
## 3  1972        Barker, Bob      6719
## 4  1983         Sajak, Pat      6115
## 5  1983       White, Vanna      6090
## 6  1972 Pennington, Janice      5519
## 7  1982   Vorderman, Carol      4756
## 8  1992          Leno, Jay      4622
## 9  1983               <NA>      4232
## 10 1982  Whiteley, Richard      3985
```

R Version: This was supposed to split by year, grab the top 10 (currently limiting by 5 for time's sake), and display all counts WITH their corresponding names. Then, as a second step, consider all of these counts unconditional on year and grab the top 10. This did not end up happening, but I did get it to a place that outputs all top 10 counts per year. I am certainly not proud of this whatsoever, despite the fact that I used a function.

```
q11_df <- dbGetQuery(imdb, "SELECT lname||', '||fname AS names, year
           FROM movies, acted_in, actors
           WHERE acted_in.idactors = actors.idactors
           AND acted_in.idmovies = movies.idmovies")

split_years <- split(q11_df, q11_df$year)

num_actor <- function(i)
{
  year_i <- as.data.frame(split_years[i])
  head(sort(table(year_i), decreasing = TRUE), 5)
}
```

```
sapply(1:length(split_years), num_actor)
```

12. Who are the 10 actors that have the most aliases (i.e., see the aka_names table). This question is strategically similar to #9 in that I am finding the top counts for actors with alternative names. I want to take this opportunity to point out the issue again. In this question, I left the first name and last name separate, not connected with || - the reason being that Joe D'Amato's name contains an apostrophe, and if I were to format the names with the || operator, his 2nd place count of 69 would be completely omitted. I left it in this original state to compare with #9.

```
dbGetQuery(imdb, "SELECT actors.fname, actors.lname,
                COUNT(aka_names.idactors) AS num_altnames
           FROM aka_names, actors
           WHERE aka_names.idactors = actors.idactors
           GROUP BY actors.idactors
           ORDER BY COUNT(aka_names.idactors) DESC
           LIMIT 10")
```

```
##            fname    lname num_altnames
## 1           Jess   Franco           76
## 2  D'Amato, Joe     <NA>           69
## 3          Uschi   Digard           60
## 4       Herschel   Savage           52
## 5        Godfrey       Ho           49
## 6           Joey  Silvera           41
## 7         Zuzana  Presova           37
## 8      Christoph    Clark           37
## 9      Nathanael      Len           37
## 10        Sandra Kalerman           36
```

R Version: I created a table, much like #9, to display the same results as the SQL version. Because I had to use the || operator on the names to make it one column, Mr. D'Amato has been omitted (because of the apostrophe exception) and the 11th place actor is now in 10th place.

```
q12_df <- dbGetQuery(imdb, "SELECT aka_names.idactors AS idaka_actors,
                          lname||', '||fname AS names
                    FROM aka_names, actors
                    WHERE aka_names.idactors = actors.idactors")

head(sort(table(na.omit(q12_df$names)), decreasing = TRUE), 10)
```

```
## 
##     Franco, Jess    Digard, Uschi Savage, Herschel      Ho, Godfrey 
##               77               60               52               49 
##    Silvera, Joey Clark, Christoph   Len, Nathanael  Presova, Zuzana 
##               41               37               37               37 
## Kalerman, Sandra  Redgrave, Joana 
##               36               36
```

13. Networks: Pick a (lead) actor who has been in at least 20 movies. Find all of the other actors that have appeared in a movie with that person. For each of these, find all the people

they have appeared with. Use this to create a network/graph of who has appeared with who. Use the igraph or statnet packages to display this network. I picked a child actor that appears in 20 movies/episodes. After finding his unique actor ID, I used subqueries to find all of the actor IDs of those who have been in movies with him (the first degree), and then all of the actor IDs of those who have been in movies with those people in the first degree (the second degree). The second degree takes over a half hour to run, but both degrees are just lists of IDs. Optimally, I would display my results for igraph, but the output is just too crowded to even be comprehensible. I can summarize what igraph is doing, however: The central node/vertex is Dylan Kingwell in my case, and all of the edges ("branches") connect thousands of other nodes that represent the actors that he has acted with. If we went further into the second degree, we would replicate this process for every first degree node. There is an added feature to this: if two actors have worked in the same movie, there will be an edge to connect them as well, making the network a web of organizational chaos. I honestly don't find a visual use for this horribly crowded scenario - if there were fewer than 100 actors/people total in both degrees, I could see this being useful, but this is quite ridiculous.
I had to comment out the second degree so I could Knit the document in the interest of time. I am certain that there is a better method of graphically representing this data, such as adding counts to the degrees, and so forth.

```
id <- dbGetQuery(imdb, "SELECT idactors
           FROM actors
           WHERE actors.lname = 'Kingwell'
           AND actors.fname = 'Dylan'")

# First degree - find actor IDs for DK's movies (has 20 appearances, 11 unique IDs)
dbGetQuery(imdb, "SELECT DISTINCT(actors.idactors) AS idactors,
                actors.lname||', '||actors.fname AS name
           FROM actors, acted_in
           WHERE acted_in.idmovies IN (SELECT DISTINCT(idmovies)
                FROM acted_in
                WHERE idactors = '2297727')
           AND actors.idactors = acted_in.idactors")

# Second degree - find actor IDs for DK's movie's movies - takes >30 minutes to run
# dbGetQuery(imdb, "SELECT DISTINCT(actors.idactors) AS idactors,
#                      actors.lname||', '||actors.fname AS name
#            FROM actors, acted_in
#            WHERE acted_in.idmovies IN (SELECT DISTINCT(acted_in.idmovies) AS idmovies
#               FROM actors, acted_in
#               WHERE actors.idactors IN (SELECT DISTINCT(actors.idactors)
#                   FROM actors, acted_in
#                   WHERE acted_in.idmovies IN (SELECT DISTINCT(idmovies)
#                       FROM acted_in
#                       WHERE idactors = '2297727')
#                   AND actors.idactors = acted_in.idactors)
#               AND actors.idactors = acted_in.idactors)
#            AND acted_in.billing_position IN (1)")
```

Bonus question: What are the 10 television series that have the most number of movie stars appearing in the shows? In the way that I defined "movies" - which, by the way, I wish was not true; I wish I could define movies as truly "movies" (see preface and first 4 questions) - I am finding the 10 television series (or any program with a series ID) with the most distinct actor IDs. I do get results such as Law & Order and 60 Minutes, which have both been running for a very long while. Other results, especially the sheer number of distinct actor IDs per title, are questionable. I have found, though, as I assert in 13, that when an actor has only 30 or 40 films in the true IMDB online database, this database here tells me that he or she had been in upwards of 300. There are just too many anomalies in this database to even consider anything being accurate. I am just happy to have had this opportunity to learn SQL!

```
dbGetQuery(imdb, "SELECT movies.title, COUNT(DISTINCT(actors.idactors))
           FROM actors, acted_in, series, movies
           WHERE actors.idactors = acted_in.idactors
           AND acted_in.idseries = series.idseries
           AND acted_in.idseries IS NOT NULL
           AND series.idmovies = movies.idmovies
           GROUP BY movies.title
           ORDER BY COUNT(DISTINCT(actors.idactors)) DESC
           LIMIT 10")
```

```
##                                 title COUNT(DISTINCT(actors.idactors))
## 1                          The Bill                             9201
## 2                            Tatort                             8112
## 3                          Casualty                             6936
## 4                       Law & Order                             6844
## 5                           Doctors                             6553
## 6         NFL Monday Night Football                             6376
## 7                        60 Minutes                             6128
## 8  Law & Order: Special Victims Unit                             5549
## 9                         Biography                             5208
## 10                               ER                             4993
```