Algorithms and programming in Python and R for Data Science

Introduction

The main focus of the project assignment is the ability to analyze and visualize data with Python and R.
The data model consists of three different tables, *top1000movies.csv, top1000movies.tsv* and *Actors.tsv*:

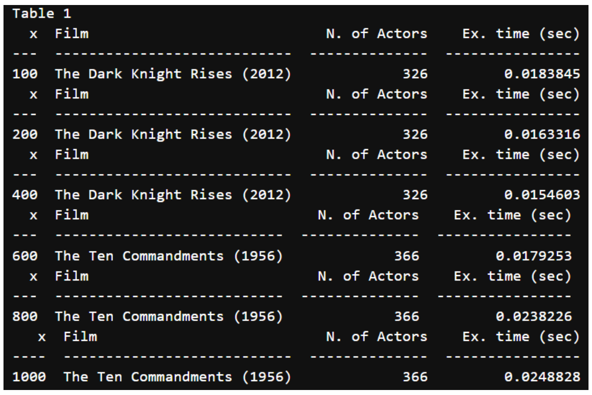
• *Top1000movies.tsv* contains the list of the top 1000 films classified in IMDB (Internet Movies DataBase). Each row is contains the film ranking position and the film title.
• *Top1000movies.csv* contains the details of the first 1000 movies in IMDB ranking, for each row we have Title, Genre, Director and other informations.
• For each line in *Actors.tsv* we have the name of one actor, the title of his film and the year of the film.
The project is made of two parts, in the first phase we have several questions to answer with a Python script, in the second part we use R to analyze the output of the Python script and the movies data.

Python Section

The Python section foresee that for a variable x with a set in {100, 200, 400, 600, 800, 1000}, where x is the first x movies in Top1000movies.tsv, the answer to these questions:

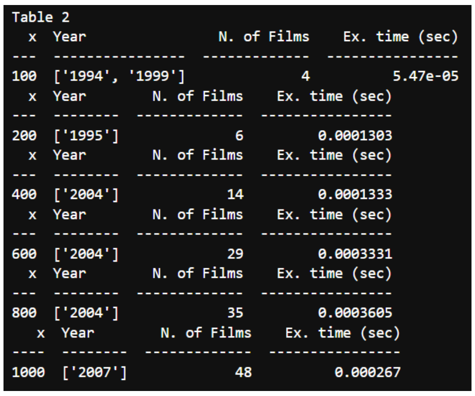
- Which film has the higher number of actors?
- What is the year in which there are more films?
- Which actor has made the more films?
- Median of the number of films made in a year
- Median of the number of films made by an actor
- Who is the actor who has collaborated with the more actors?
- Which actor couple has made the more films together?

The main techniques used were the construction of dictionaries on which to iterate, *for* loops, the definition of ad hoc functions, the use of *if* conditions and nested loops.
The modules used were *Timeit* for the calculation of the execution time, *Tabulate* for the visualization of the table in the printing phase and csv (only the Import part) for the creation of csv files in the output phase.
The films that have the more actors of all for the selected x values are:



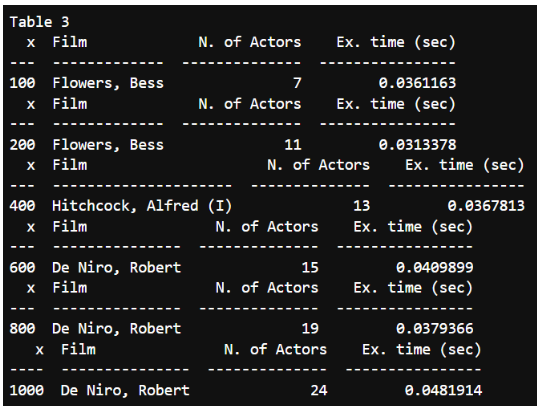
```
Table 1
  x  Film                           N. of Actors    Ex. time (sec)
---  ---------------------------  --------------  ----------------
100  The Dark Knight Rises (2012)            326         0.0183845
  x  Film                           N. of Actors    Ex. time (sec)
---  ---------------------------  --------------  ----------------
200  The Dark Knight Rises (2012)            326         0.0163316
  x  Film                           N. of Actors    Ex. time (sec)
---  ---------------------------  --------------  ----------------
400  The Dark Knight Rises (2012)            326         0.0154603
  x  Film                          N. of Actors    Ex. time (sec)
---  --------------------------  --------------  ----------------
600  The Ten Commandments (1956)            366         0.0179253
  x  Film                          N. of Actors    Ex. time (sec)
---  --------------------------  --------------  ----------------
800  The Ten Commandments (1956)            366         0.0238226
   x  Film                          N. of Actors    Ex. time (sec)
----  --------------------------  --------------  ----------------
1000  The Ten Commandments (1956)            366         0.0248828
```

In the table above it is possible to observe that, for the first three iterations of x the film with more actors, the answer is *The Dark Knight Rises (2012)* while for x greater than or equal to 600 the film is *The Ten Commandments (1956)*. Looking at the execution time we can see that for each iteration the time is around 0.02 sec. At first, a list was created, with extreme upper x, containing the first x films in terms of ranking, then through the use of a dictionary containing the films, it was iterated and populated with the film of each actor in the A*ctor.tsv* file through a for loop. The last step is the printing of the result and the creation of the relative csv.



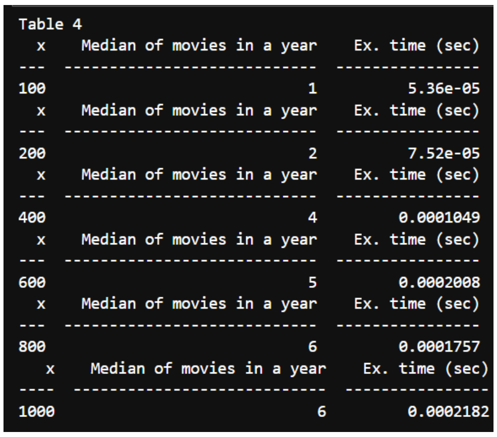
```
Table 2
  x  Year                N. of Films    Ex. time (sec)
---  ----------------  -------------  ----------------
100  ['1994', '1999']              4          5.47e-05
  x  Year        N. of Films    Ex. time (sec)
---  --------  -------------  ----------------
200  ['1995']              6         0.0001303
  x  Year        N. of Films    Ex. time (sec)
---  --------  -------------  ----------------
400  ['2004']             14         0.0001333
  x  Year        N. of Films    Ex. time (sec)
---  --------  -------------  ----------------
600  ['2004']             29         0.0003331
  x  Year        N. of Films    Ex. time (sec)
---  --------  -------------  ----------------
800  ['2004']             35         0.0003605
   x  Year        N. of Films    Ex. time (sec)
----  --------  -------------  ----------------
1000  ['2007']             48          0.000267
```

The second point, required to manipulate the data to get only the year of the film from the *topmovies1000* file instead of the full name. In the main part, the procedure was similar to the first question. Through the creation of ad hoc dictionaries and I used a for loop in order to obtain the year with the greatest number of films. We can see how the execution times have improved. Without considering x = 100, which has by far the shortest execution time, the rest are around 0.003 seconds. This case, like the previous one, has a linear complexity.

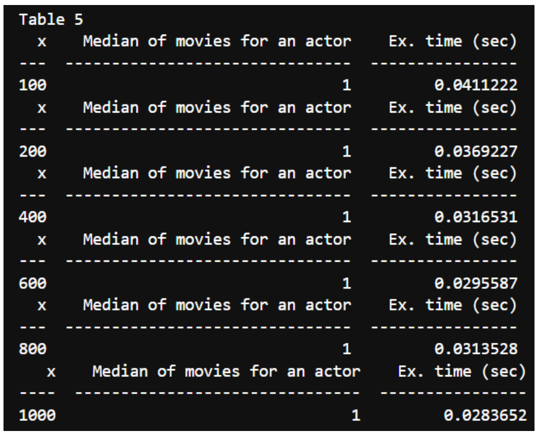


```
Table 3
  x  Film             N. of Actors    Ex. time (sec)
---  ------------    -------------   ---------------
100  Flowers, Bess               7         0.0361163
  x  Film             N. of Actors    Ex. time (sec)
---  ------------    -------------   ---------------
200  Flowers, Bess              11         0.0313378
  x  Film                     N. of Actors    Ex. time (sec)
---  --------------------    -------------   ---------------
400  Hitchcock, Alfred (I)              13         0.0367813
  x  Film               N. of Actors    Ex. time (sec)
---  --------------    -------------   ---------------
600  De Niro, Robert              15         0.0409899
  x  Film               N. of Actors    Ex. time (sec)
---  --------------    -------------   ---------------
800  De Niro, Robert              19         0.0379366
   x  Film               N. of Actors    Ex. time (sec)
----  --------------    -------------   ---------------
1000  De Niro, Robert              24         0.0481914
```

To answer the third question, I created an ad hoc dictionary that has as keys the names of the actors on which to iterate through a for loop. The execution times increase, reaching a peak of 0.05 for x = 1000. This may be due to the dictionary, because the number of actors is higher than the number of films or years.

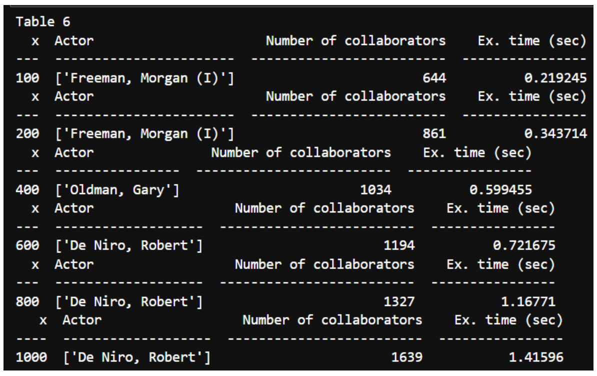


```
Table 4
  x    Median of movies in a year    Ex. time (sec)
---  ----------------------------   ---------------
100                             1          5.36e-05
  x    Median of movies in a year    Ex. time (sec)
---  ----------------------------   ---------------
200                             2          7.52e-05
  x    Median of movies in a year    Ex. time (sec)
---  ----------------------------   ---------------
400                             4         0.0001049
  x    Median of movies in a year    Ex. time (sec)
---  ----------------------------   ---------------
600                             5         0.0002008
  x    Median of movies in a year    Ex. time (sec)
---  ----------------------------   ---------------
800                             6         0.0001757
   x    Median of movies in a year    Ex. time (sec)
----  ----------------------------   ---------------
1000                             6         0.0002182
```



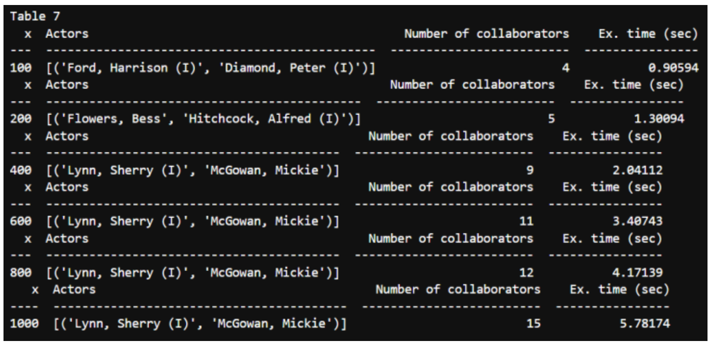
```
Table 5
  x    Median of movies for an actor    Ex. time (sec)
---  -------------------------------   ---------------
100                                1         0.0411222
  x    Median of movies for an actor    Ex. time (sec)
---  -------------------------------   ---------------
200                                1         0.0369227
  x    Median of movies for an actor    Ex. time (sec)
---  -------------------------------   ---------------
400                                1         0.0316531
  x    Median of movies for an actor    Ex. time (sec)
---  -------------------------------   ---------------
600                                1         0.0295587
  x    Median of movies for an actor    Ex. time (sec)
---  -------------------------------   ---------------
800                                1         0.0313528
   x    Median of movies for an actor    Ex. time (sec)
----  -------------------------------   ---------------
1000                                1         0.0283652
```

The fourth and fifth questions involved calculating the median of films in a year and the median of films per actor.
No library was used for the calculation of the median but I used an *if* command and a *len* command. Table 4, iterating over the dictionary with the years, resulted much faster than Table 5 which uses the dictionary of actors.
We can see how, as x increases, the median of films grows until 6 while for the median of films per actor remains stable at 1.



```
Table 6
  x  Actor                      Number of collaborators    Ex. time (sec)
---  ----------------------    ------------------------   ---------------
100  ['Freeman, Morgan (I)']                        644          0.219245
  x  Actor                      Number of collaborators    Ex. time (sec)
---  ----------------------    ------------------------   ---------------
200  ['Freeman, Morgan (I)']                        861          0.343714
  x  Actor               Number of collaborators    Ex. time (sec)
---  ---------------    ------------------------   ---------------
400  ['Oldman, Gary']                       1034          0.599455
  x  Actor                  Number of collaborators    Ex. time (sec)
---  ------------------    ------------------------   ---------------
600  ['De Niro, Robert']                       1194          0.721675
  x  Actor                  Number of collaborators    Ex. time (sec)
---  ------------------    ------------------------   ---------------
800  ['De Niro, Robert']                       1327           1.16771
   x  Actor                  Number of collaborators    Ex. time (sec)
----  ------------------    ------------------------   ---------------
1000  ['De Niro, Robert']                       1639           1.41596
```

For the sixth question, I made a function that involves the creation of two dictionaries, *movie2actors* and *actors2movie*. The first dictionary assigns the number of actors to the movie; the second dictionary instead, does the reverse process. Both dictionaries are parameterized respect to x.
A third dictionary is created with the numbers of collaboration between the actors, thanks to the use of the *movie2actors* and *actors2movie* dictionaries.
It is possible to observe an increase in processing times as x increases, this behavior is predictable since the dictionaries are dynamic respect to x.
Furthermore, using nested cycles, the calculation complexity is no longer linear. The execution time for x = 1000 is approximately five times that of x = 100.



```
Table 7
  x  Actors                                          Number of collaborators    Ex. time (sec)
---  ------------------------------------------  -----------------------  ----------------
100  [('Ford, Harrison (I)', 'Diamond, Peter (I)')]                        4           0.90594
  x  Actors                                        Number of collaborators    Ex. time (sec)
---  ------------------------------------------  -----------------------  ----------------
200  [('Flowers, Bess', 'Hitchcock, Alfred (I)')]                        5           1.30094
  x  Actors                                      Number of collaborators    Ex. time (sec)
---  ---------------------------------------  -----------------------  ----------------
400  [('Lynn, Sherry (I)', 'McGowan, Mickie')]                        9           2.04112
  x  Actors                                      Number of collaborators    Ex. time (sec)
---  ---------------------------------------  -----------------------  ----------------
600  [('Lynn, Sherry (I)', 'McGowan, Mickie')]                       11           3.40743
  x  Actors                                      Number of collaborators    Ex. time (sec)
---  ---------------------------------------  -----------------------  ----------------
800  [('Lynn, Sherry (I)', 'McGowan, Mickie')]                       12           4.17139
   x  Actors                                      Number of collaborators    Ex. time (sec)
----  ---------------------------------------  -----------------------  ----------------
1000  [('Lynn, Sherry (I)', 'McGowan, Mickie')]                       15           5.78174
```

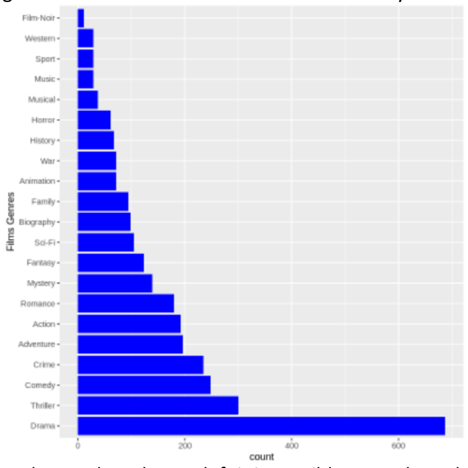
Table7 is the most time consuming. The time of execution equals approximately 65% of the total program time.
The first steps are similar to the function of Table6 which provides for the creation of three dictionaries: *movie2actors*, *actors2movie* and *collab*. The additional step involves the creation of a new dictionary that associates to each couple of actors that worked together, the number of films in which they where both part of the cast.
The execution of the entire program takes between 20 and 30 seconds, in the last execution, dated 01/06/2022, it took 25 seconds.

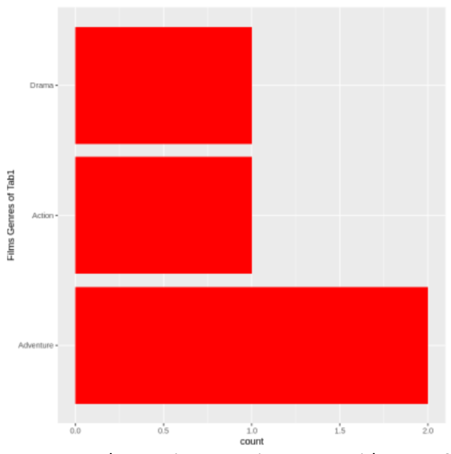
R Section

The R part involved the following requests:

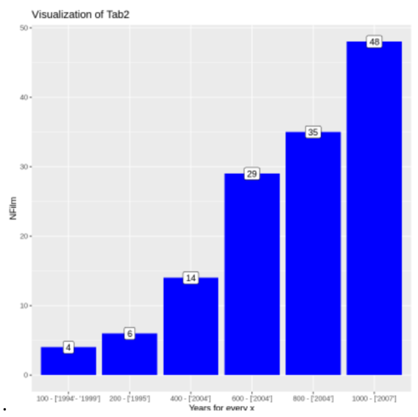
- Create a graphic about the distribution of all the movies that appear in the *top1000movies.csv* by genre. Repeat only for the films that appear in table 1.
- Graphically represent tables 2, 4 and 5.
- For each actor in tables 3 and 6, represent the distribution of films by genre.
- For each pair of actors X and Y in Table 7, graph the distribution of films by genre separately for actor X and actor Y.

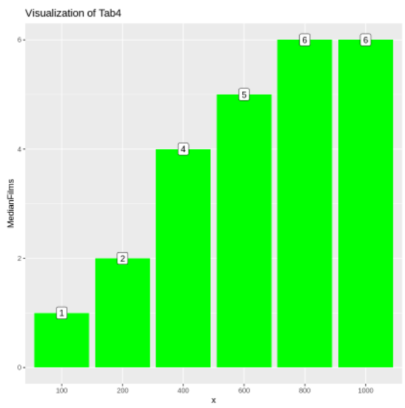
I used three libraries, *Stringr* for the manipulation of the dataframes, *ggplot2* for the creation of the graphs and *readr* for read the tsv files.
The most complex operation, was the split of the genres column, formed in the following way: "Genre1, Genre2, Genre3". To solve this problem I used *strsplit* to divide the genres through a separator and then through the *lapply* and the *replace* methods. I created 8 columns, each one enclosed a genre (populated with NA in the case of genres less than 8). Subsequently, through the *stack* and the *subset* commands, I have enclosed all the genres of the films taken into consideration by eliminating the NA values in a single column then represented through the ggplot library.

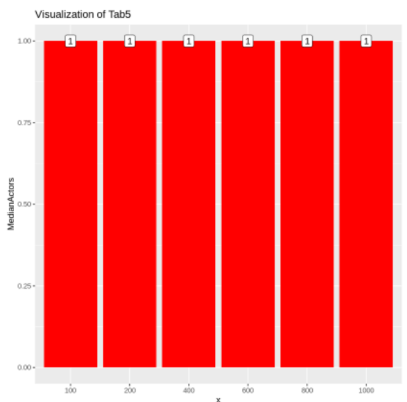




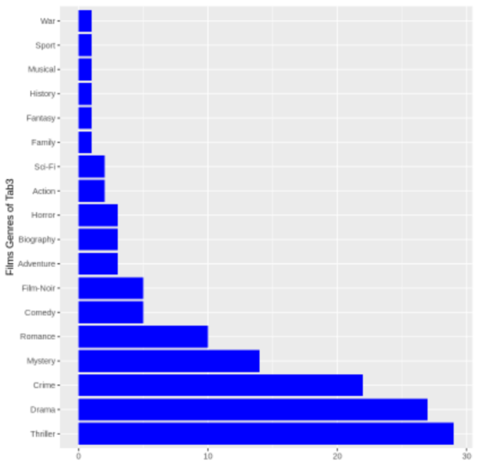
In the graph at the top left it is possible to see how the most popular movie genres is Drama, with over 700 occurrences, followed by Thriller, Comedy, Crime and Adventure, which are in the 200-300 range. For the genres of table 1 the Adventure genre is the Mode.

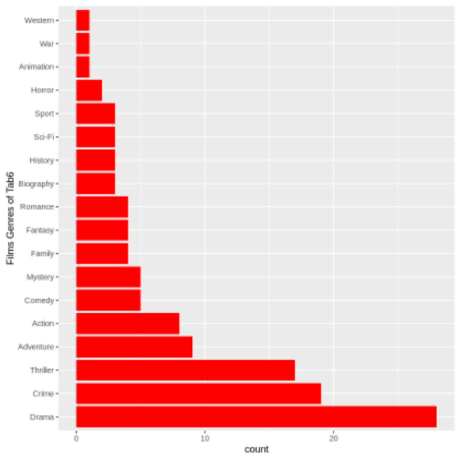




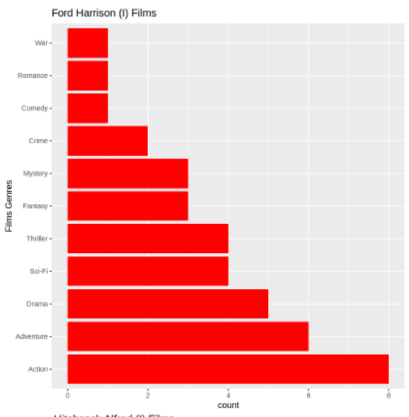


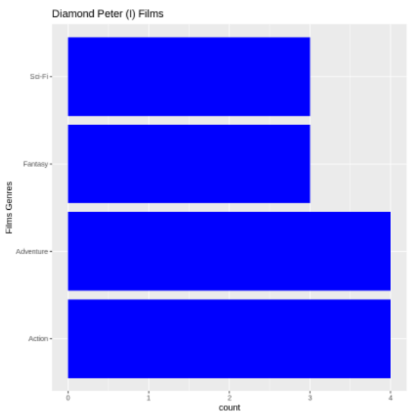
.
While in the graph relating to Table 4 it is possible to notice a growth of the median with the increase of films, which is predictable, Table 5 has a constant median equal to 1, since most of the actors have made only one film.

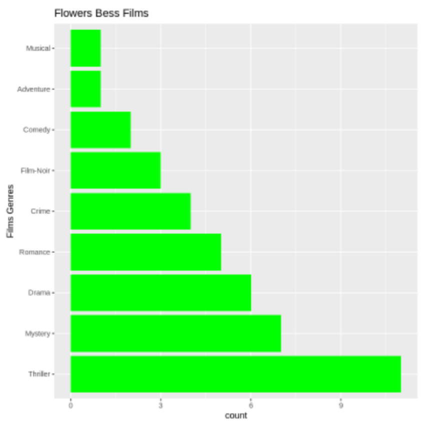


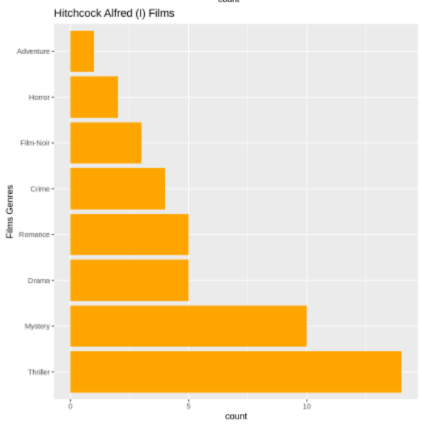


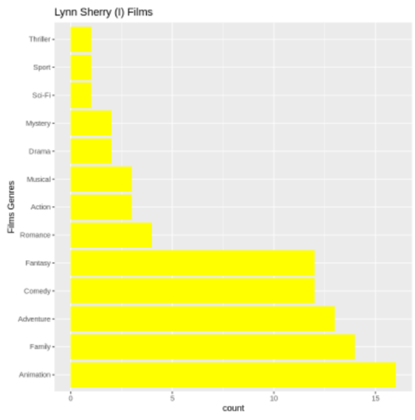
In the graph of table 3 (films by the actors Flowers, De Niro, Hitchcock), the distribution of genres is a good representation of the total distribution of films because it presents Thriller, Drama and Crime among the most frequent genres. In the graph of table 6 (film by the actors Morgan, De Niro, Oldman) we have a distribution similar to graph 3 with the exception of the Romance genre which reduces its frequency because Flowers is no longer presented.

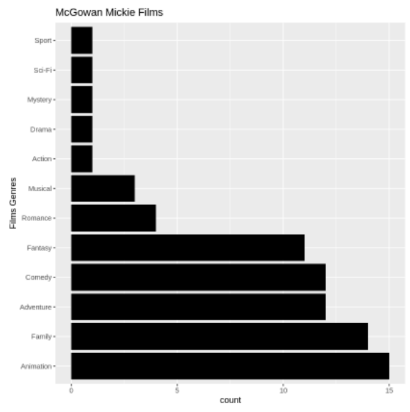












A further precaution to create these graphics was to format the name of the films in the same way between actor.tsv and top1000movies.csv by deleting the year from the name and the special symbols. To do this, we used the *str_replace_all* command to remove special symbols and *substring* to remove years. By observing the graphs it is possible to see how the distribution of genres is similar for the actors who have carried out multiple collaborations between them. This pattern is more visible for the Flower Bess-Hitchcock Alftred and Lynn Sherry-McGowan Mickie couples.