STA141 Assignment 5

Zhen Zhang

November 30, 2015

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.

```
library(RSQLite)
```

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

```
library(plyr)
# read in two databases
con = dbConnect(SQLite(), 'lean_imdbpy.db')
con2 = dbConnect(SQLite(), 'lean_imdbpy_2010_idx.db')
```

1

```
# number of movies

# In this part I will create two new tables, which I will use afterwards all the time.
# One is title_movie, which is the subset of title only containing movies. The other is
# name_actor, which is also a subset of name only containing actors. Since questions
# afterwards will keep focused on movies and actors, I will create them as temporary
# tables.

# create a new title table that only have movies, and give the table name title_movie
# NOTE: THIS WILL BE RUN FOR ONLY ONE TIME SINCE IT WILL CREATE A NEW TABLE

# dbGetQuery(con, '
#             CREATE TABLE title_movie AS
#             SELECT DISTINCT title.*
#             FROM title JOIN kind_type kt ON title.kind_id = kt.id
#             WHERE kt.kind = "movie"
#             ')

dbGetQuery(con, '
           SELECT COUNT(DISTINCT title_movie.id) movie_count
           FROM title_movie
           ')
```

```
##   movie_count
## 1      878800
```

```
# number of actors

# create a new name table that only have actors, and give the table name name_actor
# Also one table

# dbGetQuery(con, '
#            CREATE TABLE name_actor AS
#            SELECT DISTINCT name.*
#            FROM cast_info ci JOIN name ON ci.person_id = name.id
#               JOIN role_type rt on ci.role_id = rt.id
#            WHERE rt.role IN ("actor", "actress")
#            ')


dbGetQuery(con, '
           SELECT count(name_actor.id) actor_count
           FROM name_actor
           ')
```

```
##   actor_count
## 1     3492018
```

2

```
# year span
dbGetQuery(con, '
           SELECT MAX(production_year) year_max, MIN(production_year) year_min
           FROM title
           ')
```

```
##   year_max year_min
## 1     2025     1874
```

3

```
# get the count of each gender
each_gender <-
  dbGetQuery(con, '
             SELECT COUNT(*) gender_count, gender
             FROM name_actor
             GROUP BY gender;
             ')

# get the total count
total_number_gender <-
  dbGetQuery(con, '
             SELECT COUNT(*)
             FROM name_actor
             ')
```

```
# propotionalized
each_gender$gender_count <- each_gender$gender_count / total_number_gender[[1]]

# return result
each_gender
```

```
##   gender_count gender
## 1    0.3537021      f
## 2    0.6462979      m
```

4

```
# get the count of type
each_kind <-
  dbGetQuery(con, '
             SELECT COUNT(*) type_count, kt.kind
             FROM title JOIN kind_type kt ON title.kind_id = kt.id
             GROUP BY kt.kind
             ')

# get the total number
total_number_kind <-
  dbGetQuery(con, '
             SELECT COUNT(*)
             FROM title
             ')

# proportionalize
each_kind$type_count <- each_kind$type_count / total_number_kind[[1]]

# print result
each_kind
```

```
##    type_count        kind
## 1 0.635583712     episode
## 2 0.249111894       movie
## 3 0.034126175    tv movie
## 4 0.035273371   tv series
## 5 0.004341033  video game
## 6 0.041563815 video movie
```

5

```
# get the number of genres
dbGetQuery(con, '
           SELECT COUNT(*) genre_count
           FROM (
              SELECT mi.info
              FROM info_type it JOIN movie_info mi ON it.id = mi.info_type_id
```

```
                  JOIN title_movie tm ON mi.movie_id = tm.id
              WHERE it.info = "genres"
              GROUP BY mi.info
           )
           ')
```

```
##   genre_count
## 1          28
```

```
# get all genres
dbGetQuery(con, '
           SELECT mi.info
           FROM info_type it INNER JOIN movie_info mi ON it.id = mi.info_type_id
              INNER JOIN title_movie tm ON mi.movie_id = tm.id
           WHERE it.info = "genres"
           GROUP BY mi.info
           ')
```

```
##           info
## 1       Action
## 2        Adult
## 3    Adventure
## 4    Animation
## 5    Biography
## 6       Comedy
## 7        Crime
## 8  Documentary
## 9        Drama
## 10      Family
## 11     Fantasy
## 12   Film-Noir
## 13   Game-Show
## 14     History
## 15      Horror
## 16       Music
## 17     Musical
## 18     Mystery
## 19        News
## 20  Reality-TV
## 21     Romance
## 22      Sci-Fi
## 23       Short
## 24       Sport
## 25   Talk-Show
## 26    Thriller
## 27         War
## 28     Western
```

```
# What if we want genres from all movies and tvs and others?
dbGetQuery(con, '
           SELECT mi.info
           FROM info_type it INNER JOIN movie_info mi ON it.id = mi.info_type_id
```

```
              INNER JOIN title tm ON mi.movie_id = tm.id
           WHERE it.info = "genres"
           GROUP BY mi.info
           ')
```

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

6

```
# the 10 most common genres of movies, showing the number of movies in each of
# these genres
top_10_genres <-
  dbGetQuery(con, '
             SELECT mi.info, COUNT(mi.info) genre_count
             FROM info_type it INNER JOIN movie_info mi ON it.id = mi.info_type_id
                INNER JOIN title_movie tm ON mi.movie_id = tm.id
             WHERE it.info = "genres"
             GROUP BY mi.info
             ORDER BY genre_count DESC
             LIMIT 10
             ')

top_10_genres
```

```
##            info genre_count
## 1         Short      470488
## 2         Drama      269898
## 3        Comedy      180315
## 4   Documentary      145018
## 5       Romance       52324
## 6      Thriller       51961
## 7        Action       45077
## 8        Horror       38620
## 9     Animation       38461
## 10        Crime       33010
```

7

```
# all movies with the keyword 'space'
movie_space <-
  dbGetQuery(con, '
             SELECT tm.id, tm.title
             FROM title_movie tm INNER JOIN movie_keyword mk ON tm.id = mk.movie_id
                INNER JOIN keyword kw ON mk.keyword_id = kw.id
             WHERE kw.keyword = "space"
             ')

head(movie_space)
```

```
##        id                        title
## 1 2365979          002 operazione Luna
## 2 2367917               12 to the Moon
## 3 2371167    20 Million Miles to Earth
## 4 2371436        2001: A Space Odyssey
## 5 2371922                         2010
## 6 2376022 4: Rise of the Silver Surfer
```

```
summary(movie_space)
```

```
##        id             title
##  Min.   :2365979   Length:401
##  1st Qu.:2668393   Class :character
##  Median :3004151   Mode  :character
##  Mean   :2969881
##  3rd Qu.:3241566
##  Max.   :3521673
```

```
# count
dbGetQuery(con, '
           SELECT COUNT(*) space_count
           FROM title_movie tm INNER JOIN movie_keyword mk ON tm.id = mk.movie_id
              INNER JOIN keyword kw ON mk.keyword_id = kw.id
           WHERE kw.keyword = "space"
           ')
```

```
##   space_count
## 1         401
```

```
# years
movie_space_year <-
  dbGetQuery(con, '
             SELECT DISTINCT tm.production_year
             FROM title_movie tm INNER JOIN movie_keyword mk ON tm.id = mk.movie_id
                INNER JOIN keyword kw ON mk.keyword_id = kw.id
             WHERE kw.keyword = "space" AND tm.production_year IS NOT NULL
             ORDER BY tm.production_year
             ')

movie_space_year$production_year
```

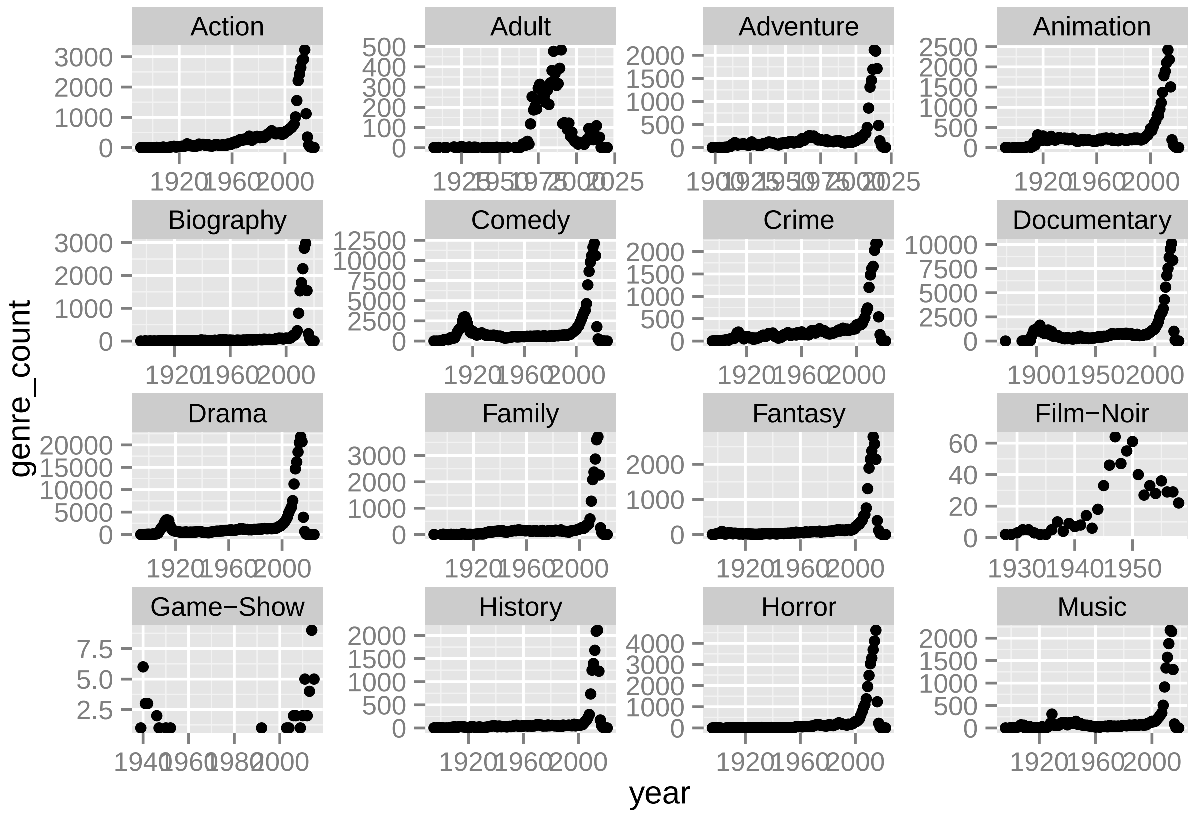
```
##  [1] 1911 1918 1922 1925 1930 1946 1947 1950 1951 1953 1954 1955 1956 1957
## [15] 1958 1959 1960 1961 1962 1964 1965 1966 1967 1968 1969 1970 1971 1972
## [29] 1973 1974 1975 1977 1978 1979 1980 1981 1982 1983 1984 1985 1986 1987
## [43] 1988 1989 1990 1991 1992 1993 1994 1996 1997 1998 1999 2000 2001 2002
## [57] 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014 2015 2016
## [71] 2017 2018
```

```
# top n actors with each movie

movie_space_top_actor_each <- function(n) {
  # This is the function that gives the result of top 5 movies with n movies. So if
  # you want the data for the first 10 movies, just give the argument n = 10, and this
  # is what I do afterwards.

  top_actor <- lapply(movie_space[['id']][1:n], function(x) {
                dbGetQuery(con, paste('
                           SELECT na.id, na.name, ci.nr_order
                           FROM title_movie tm JOIN cast_info ci ON tm.id = ci.movie_id
                              JOIN name_actor na ON na.id = ci.person_id
                           WHERE tm.id =', x, '
                           AND ci.nr_order IN (1, 2, 3, 4, 5, 6)
                           GROUP BY na.id
                           ORDER BY ci.nr_order
                           LIMIT 5'
                           ))
              })

  names(top_actor) <- movie_space[['name']][1:n]
  top_actor
}

movie_space_top_actor_total <- movie_space_top_actor_each(10)

movie_space_top_actor_total
```

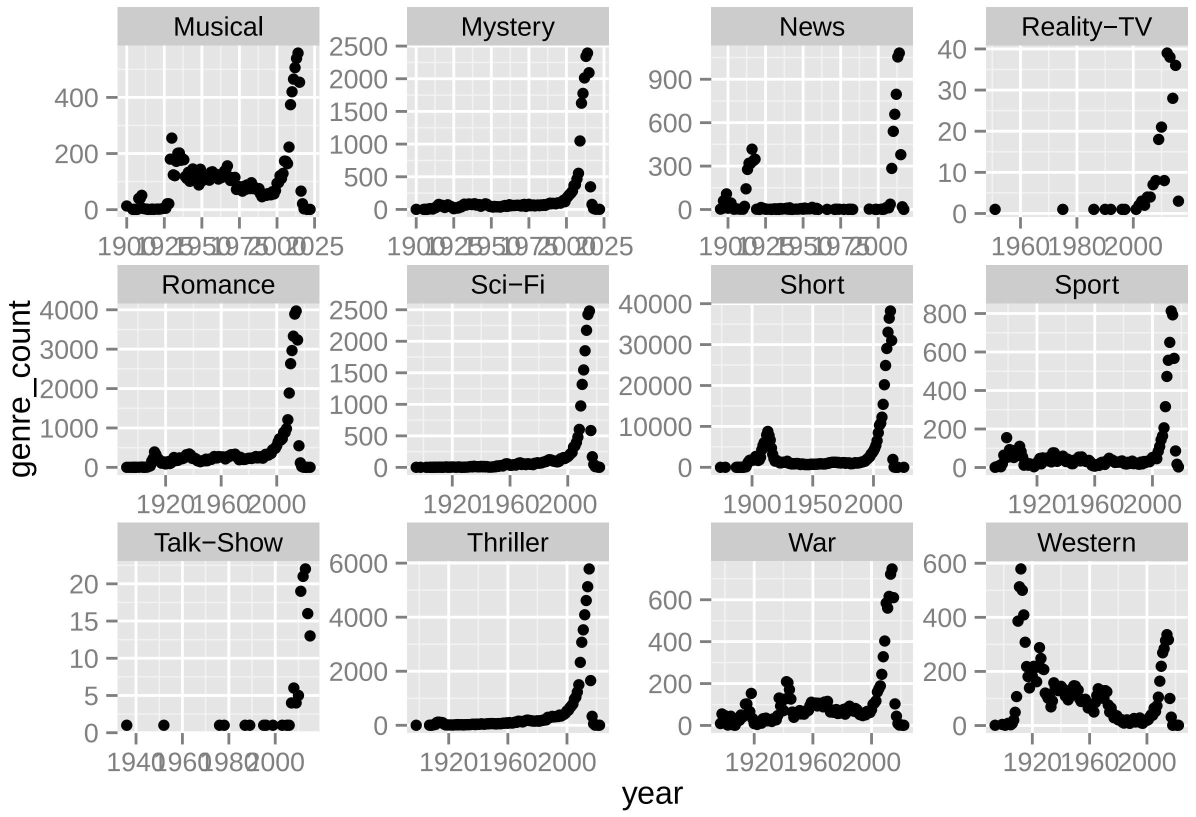
```
## [[1]]
##        id              name nr_order
## 1  661113   Franchi, Franco        1
## 2  935665 Ingrassia, Ciccio        2
## 3 3172528   Randall, Mónica        3
## 4 3291555       Sini, Linda        4
## 5 3286328      Silva, María        5
##
## [[2]]
##        id            name nr_order
## 1  374630      Clark, Ken        1
## 2 2845023     Kobi, Michi        2
## 3  402504     Conway, Tom        3
## 4  506884 Dexter, Anthony        4
## 5 2164166   Wengraf, John        5
##
## [[3]]
##        id              name nr_order
## 1  899083   Hopper, William        1
## 2 2843284 Knight, Charlotte        1
## 3 3357735      Taylor, Joan        2
## 4 1633148     Puglia, Frank        3
## 5 2243618     Zaremba, John        4
##
## [[4]]
##        id               name nr_order
## 1  550078       Dullea, Keir        1
## 2 1195795     Lockwood, Gary        2
## 3 1974787 Sylvester, William        3
## 4 1694544    Richter, Daniel        4
## 5 1740095  Rossiter, Leonard        5
##
## [[5]]
##        id          name nr_order
## 1  924468  Hyams, Peter        1
## 2 1803971 Scheider, Roy        1
## 3 1189509 Lithgow, John        2
## 4 3016212 Mirren, Helen        3
## 5  111419  Balaban, Bob        4
##
## [[6]]
##        id           name nr_order
## 1  782884 Gruffudd, Ioan        1
## 2 2060367   Turman, John        1
## 3 1155168      Lee, Stan        2
## 4 2275575  Alba, Jessica        2
## 5  599152   Evans, Chris        3
##
## [[7]]
## [1] id       name     nr_order
## <0 rows> (or 0-length row.names)
##
## [[8]]
##        id                   name nr_order
## 1 2122907 von Zeddelmann, Moritz        1
## 2 3098065    Osterloh, Dolly-Ann        2
## 3  425764           Cree, Steven        3
## 4 1440847          Nallon, Steve        4
## 5  539062           Doyle, Jamie        5
##
## [[9]]
## [1] id       name     nr_order
## <0 rows> (or 0-length row.names)
##
## [[10]]
## [1] id       name     nr_order
## <0 rows> (or 0-length row.names)
```

8

```
# The number of genres over time
genre_year <-
  dbGetQuery(con, '
                  SELECT mi.info genre, tm.production_year year, COUNT(mi.info) genre_count
                  FROM info_type it JOIN movie_info mi ON it.id = mi.info_type_id
                      JOIN title_movie tm ON mi.movie_id = tm.id
                  WHERE it.info = "genres" AND
                      tm.production_year IS NOT NULL AND
                      genre IS NOT NULL
                  GROUP BY mi.info, tm.production_year
                  ')

library(ggplot2)

# here I control the observations to make separate geners on 2 different plots
ggplot(genre_year[1:1778, ], aes(year, genre_count)) +
  geom_point() +
  facet_wrap(~ genre, scales = 'free')
```



```r
ggplot(genre_year[1779:dim(genre_year)[1], ], aes(year, genre_count)) +
  geom_point() +
  facet_wrap(~ genre, scales = 'free')
```



```
# From the figure I can see that, most of the movies increase the number after year 2000
# rapidly, except movie genre western and war.
```

9-12 are using small dataset

```
# To be consistent, I will also create two tables as before, which are movies and actors
# THIS WILL BE DONE ONLY ONCE

# dbGetQuery(con2, '
#             CREATE TABLE title_movie2 AS
#             SELECT DISTINCT title2.*
#             FROM title2 JOIN kind_type kt ON title2.kind_id = kt.id
#             WHERE kt.kind = "movie"
#             ')
#
# dbGetQuery(con2, '
#             CREATE TABLE name_actor2 AS
#             SELECT DISTINCT name2.*
#             FROM cast_info2 ci JOIN name2 ON ci.person_id = name2.id
#             JOIN role_type rt on ci.role_id = rt.id
#             WHERE rt.role IN ("actor", "actress")
#             ')
```

```r
# Here I will first get all tables I need from sql and then join them

# name_actor
data_na <-
  dbGetQuery(con2, '
                  SELECT *
                  FROM name_actor2 na
                  ')
names(data_na) <- paste0('na.',names(data_na))

# title_movie
data_tm <-
  dbGetQuery(con2, '
            SELECT *
            FROM title_movie2 tm
            ')
names(data_tm) <- paste0('tm.',names(data_tm))

# cast_info
data_ci <-
  dbGetQuery(con2, '
            SELECT *
            FROM cast_info2 ci
            ')
names(data_ci) <- paste0('ci.',names(data_ci))

# inner join them, use package dplyr
library(dplyr)

data_na_tm_ci <-
  inner_join(
    inner_join(
      data_na, data_ci, by = c('na.id' = 'ci.person_id')
      ),
    data_tm, by = c('ci.movie_id' = 'tm.id')
    )

# Then when I am using R, I can just select columns in data_na_tm_ci
```

9

```r
# Top 20 actors in movies

# SQL approach
dbGetQuery(con2, '
           SELECT na.id, na.name, COUNT(*) actor_count
           FROM cast_info2 ci JOIN name_actor2 na ON ci.person_id = na.id
              JOIN title_movie2 tm ON tm.id = ci.movie_id
           GROUP BY na.id
           ORDER BY actor_count DESC
           LIMIT 20
           ')
```

```
##          id                    name actor_count
## 1  1234086          MacLeod, Kevin        1508
## 2   568207            Edward, Noah         625
## 3  1705106           Rivers, Scott         529
## 4   636012 Fischbach, Mark Edward         443
## 5   919340              Hunter, G.         381
## 6  1228623           Macaroni, Sam         380
## 7  1735527            Rosen, Larry         340
## 8  1459666           Newton, Brett         339
## 9  1549168             Pasha, Omer         332
## 10 1217855             Lund, Tyler         329
## 11 1887456           Sloan, Lee A.         311
## 12  430358            Cross, Logan         294
## 13 2228101       Yeriomin, Nikolay         267
## 14 1819953          Sciolè, Flavio         246
## 15 1479028      Notarile, Chris R.         244
## 16 1366507           Mills, Travis         239
## 17     686              A., Sergey         236
## 18 2156581             Weecks, Dan         228
## 19 1488023        O'Connor, George         225
## 20 1700138        Ringgaard, Peter         214
```

```
# R approach

# extract data
actors <- data_na_tm_ci[, c('na.id', 'na.name')]
names(actors) <- c('id', 'name')

# use ddply to first split data by id and name, then count the number
actors_count <- ddply(actors, .(id, name), nrow)

# see the top 20
head(actors_count[order(actors_count$V1, decreasing = T), ], n = 20)
```

```
##              id                   name   V1
## 412384 1234086         MacLeod, Kevin 1508
## 191576  568207           Edward, Noah  625
## 571846 1705106          Rivers, Scott  529
## 214117  636012 Fischbach, Mark Edward  443
## 309021  919340             Hunter, G.  381
## 410604 1228623          Macaroni, Sam  380
## 582261 1735527           Rosen, Larry  340
## 489611 1459666          Newton, Brett  339
## 519791 1549168            Pasha, Omer  332
## 407229 1217855            Lund, Tyler  329
## 633089 1887456          Sloan, Lee A.  311
## 145976  430358           Cross, Logan  294
## 745514 2228101      Yeriomin, Nikolay  267
## 610069 1819953         Sciolè, Flavio  246
## 496066 1479028     Notarile, Chris R.  244
## 458089 1366507          Mills, Travis  239
## 231        686             A., Sergey  236
## 721032 2156581            Weecks, Dan  228
## 499050 1488023       O'Connor, George  225
## 570129 1700138       Ringgaard, Peter  214
```

10

```
# top billing

# SQL approach
billing_top_10 <-
  dbGetQuery(con2, '
             SELECT na.id, na.name, COUNT(na.id) actor_billing_count,
                MIN(tm.production_year) min_year, MAX(tm.production_year) max_year
             FROM name_actor2 na JOIN cast_info2 ci on na.id = ci.person_id
                JOIN title_movie2 tm ON tm.id = ci.movie_id
             WHERE ci.nr_order IN (1, 2, 3)
             GROUP BY na.id
             ORDER BY COUNT(na.id) DESC
             LIMIT 10
             ')

billing_top_10
```

```
##         id                      name actor_billing_count min_year max_year
## 1  1204854            Lorente, Txema                 106     2010     2015
## 2  1708783             Roberts, Eric                  75     2010     2016
## 3  1881637             Sizemore, Tom                  48     2010     2016
## 4  1488023           O'Connor, George                 46     2014     2015
## 5  2046271              Trejo, Danny                  43     2010     2015
## 6   292687 Calderón, Emilio Janhunen                 38     2011     2015
## 7   257734            Brown, Shannon                  37     2011     2016
## 8  1302470          Mazak, Kasey Ryne                 37     2010     2015
## 9  1237239           Madsen, Michael                 35     2010     2016
## 10 1381086                  Mohanlal                 34     2010     2015
```

```
# R approach

# extract data
billing_data <- data_na_tm_ci[, c('na.id', 'na.name', 'tm.production_year', 'ci.nr_order')]
names(billing_data) <- c('id', 'name', 'production_year', 'nr_order')

# subset by only nr_order of 1, 2, 3, and for columns, drop the nr_order
billing_top_data <- billing_data[billing_data$nr_order %in% c(1, 2, 3), -4]

# use ddply to split id and name, and count the number and the corresponding year span
billing_top_data_count <- ddply(billing_top_data, .(id, name), nrow)

# get the top 10 movies
billing_top_data_count_head <-
  head(billing_top_data_count[order(billing_top_data_count$V1, decreasing = T), ], n = 10)
# get the max and min year
billing_top_data_count_head$max_year <-
  sapply(1:nrow(billing_top_data_count_head), function(i) {
    max(billing_data[billing_data$id == billing_top_data_count_head[i, 'id'], 'production_year'])
  })
billing_top_data_count_head$min_year <-
  sapply(1:nrow(billing_top_data_count_head), function(i) {
    min(billing_data[billing_data$id == billing_top_data_count_head[i, 'id'], 'production_year'])
  })

billing_top_data_count_head
```

```
##            id                    name  V1 max_year min_year
## 44049 1204854          Lorente, Txema 106     2015     2010
## 62242 1708783          Roberts, Eric  75     2016     2010
## 68400 1881637          Sizemore, Tom  48     2016     2010
## 54410 1488023        O'Connor, George  46     2015     2012
## 74104 2046271            Trejo, Danny  43     2016     2010
## 11146  292687 Calderón, Emilio Janhunen  38     2016     2010
## 9825   257734          Brown, Shannon  37     2016     2011
## 47551 1302470        Mazak, Kasey Ryne  37     2016     2010
## 45243 1237239         Madsen, Michael  35     2016     2010
## 50531 1381086                Mohanlal  34     2015     2010
```

11

```
# SQL approach

top_10_actors_within_year <-
  dbGetQuery(con2, '
             SELECT na.id, na.name, COUNT(*) actor_count, tm.production_year
             FROM title_movie2 tm JOIN cast_info2 ci ON tm.id = ci.movie_id
                JOIN name_actor2 na ON ci.person_id = na.id
             GROUP BY tm.production_year, na.id
             ORDER BY actor_count DESC
             LIMIT 10
             ')

top_10_actors_movies <-
  lapply(1:10, function(i) {
    dbGetQuery(con2, paste('
               SELECT DISTINCT tm.id, tm.title
               FROM title_movie2 tm JOIN cast_info2 ci ON tm.id = ci.movie_id
                   JOIN name_actor2 na ON ci.person_id = na.id
               WHERE tm.production_year =', top_10_actors_within_year[i, 'production_year'], '
                   AND na.id = ', top_10_actors_within_year[i, 'id'], '
               LIMIT 5
               '))
  })

names(top_10_actors_movies) <- paste(top_10_actors_within_year[['name']], 'at year',
                                     top_10_actors_within_year[['production_year']])

top_10_actors_movies
```

```
## $`Edward, Noah at year 2014`
##        id                      title
## 1 2383050 A Date with Snout the Wall
## 2 2408139                   Adore Me
## 3 2425579             Amazing Friend
## 4 2435644                      Angel
## 5 2441078           Anything for You
##
## $`MacLeod, Kevin at year 2013`
##        id                                       title
## 1 2366193                             1 Last Question
## 2 2368818 17 Minutes in Texas: The Zombie Apocalypse
## 3 2370981                                 2 to Tangle
## 4 2373180                                    25 Years
## 5 2373515                                          2D
##
## $`MacLeod, Kevin at year 2012`
##        id               title
## 1 2379726 A Bed of Butterflies
## 2 2383622   A Dead Man's Money
## 3 2384741      A Family Dinner
## 4 2393156   A Perilous Journey
## 5 2404014         Abracadabra!
##
## $`MacLeod, Kevin at year 2014`
##        id           title
## 1 2365937      0 Feet Away
## 2 2372807  24 and Counting
## 3 2374933        37 Fallen
## 4 2376711      500 Grammaa
## 5 2382820 A Cyberpunk Tale
##
## $`Edward, Noah at year 2013`
##        id                  title
## 1 2445257 Aritistic Discrepencies
## 2 2452053                  Asylum
## 3 2453852                  Atsuya
## 4 2466473              Ballerinas
## 5 2473283         BBQ Best Friend
##
## $`MacLeod, Kevin at year 2011`
##        id                                   title
## 1 2367712                                   11:38
## 2 2372645                                      22
## 3 2374161             3 X Harder: My Man's and 'Em
## 4 2374398                     30 Second Exorcism
## 5 2381849 A Classic Tale: From Flabby to Fantastic
##
## $`Ringgaard, Peter at year 2010`
##        id                             title
## 1 2415524         Al Khalifa Family Montage
## 2 2464608      Bahrain Ambient Film: Adhari
## 3 2464609 Bahrain Ambient Film: Calligraphy
## 4 2464610   Bahrain Ambient Film: Formula 1
## 5 2464611       Bahrain Ambient Film: Islam
```

```
##
## $`Fischbach, Mark Edward at year 2015`
##        id                 title
## 1 2380254     A Box Full of Joy
## 2 2512174          Brighter Day
## 3 2527156         Can Your Pet?
## 4 2539480    Changing the World
## 5 2632813 Don't Shit Your Pants
##
## $`MacLeod, Kevin at year 2015`
##        id                                              title
## 1 2376386 5 Schritte zur Freiheit - Every day the same dream
## 2 2377974                                           72 Lies
## 3 2379185                                               9pm
## 4 2381689                                 A Christmas Story
## 5 2384003                           A Dish Best Served Cold
##
## $`Newton, Brett at year 2014`
##        id                      title
## 1 2383050 A Date with Snout the Wall
## 2 2408139                   Adore Me
## 3 2425579             Amazing Friend
## 4 2441078           Anything for You
## 5 2452289                At Her Word
```

```r
# R approach

# extract data
top_10_actors_within_year_data <- data_na_tm_ci[, c('na.id', 'na.name', 'tm.production_year')]
names(top_10_actors_within_year_data) <- c('id', 'name', 'production_year')

top_10_actors_within_year_data_count <-
  ddply(top_10_actors_within_year_data, .(id, name, production_year), nrow)

top_10_actors_within_year_data_count_result <-
  head(top_10_actors_within_year_data_count[order(top_10_actors_within_year_data_count$V1, decreasing =

# I only need to compare it with the initial result of top 10 actors, say
# top_10_actors_within_year, since if top_10_actors_within_year is the same as
# top_10_actors_within_year_data_count_result, then they will select the same movies of top
# 10 from sql.

top_10_actors_within_year_data_count_result
```

```
##             id                  name production_year  V1
## 46093   568207          Edward, Noah            2014 340
## 4595   1234086        MacLeod, Kevin            2013 305
## 39706  1234086        MacLeod, Kevin            2012 291
## 3644   1234086        MacLeod, Kevin            2014 282
## 162260  568207          Edward, Noah            2013 271
## 8846   1234086        MacLeod, Kevin            2011 243
## 106230 1700138       Ringgaard, Peter            2010 210
## 29092  1234086        MacLeod, Kevin            2015 200
## 41196   636012 Fischbach, Mark Edward            2015 200
## 46094  1459666          Newton, Brett           2014 199
```

```
top_10_actors_within_year
```

```
##         id                  name actor_count production_year
## 1   568207          Edward, Noah         340            2014
## 2  1234086         MacLeod, Kevin         305            2013
## 3  1234086         MacLeod, Kevin         291            2012
## 4  1234086         MacLeod, Kevin         282            2014
## 5   568207          Edward, Noah         271            2013
## 6  1234086         MacLeod, Kevin         243            2011
## 7  1700138       Ringgaard, Peter         210            2010
## 8   636012 Fischbach, Mark Edward         200            2015
## 9  1234086         MacLeod, Kevin         200            2015
## 10 1459666          Newton, Brett         199            2014
```

```
# they are identical
```

12

```
# 10 actors that have the most aliases

# SQL Approach
dbGetQuery(con2, '
           SELECT na.id, na.name, COUNT(*) alias_count
           FROM name_actor2 na JOIN aka_name2 an ON na.id = an.person_id
           GROUP BY na.id
           ORDER BY alias_count DESC
           LIMIT 10')
```

```
##         id                 name alias_count
## 1   662453        Franco, Jesús          78
## 2  1796694      Savage, Herschel          53
## 3  1869225          Silvera, Joey          42
## 4   373754      Clark, Christoph          38
## 5  1098131 Kronos, Donald Arthur          37
## 6  1792238      Sarno, Joseph W.          36
## 7  3213694           Rose, Sasha          32
## 8   969854           Jeremy, Ron          31
## 9   728227         Gillis, Jamie          30
## 10 2540989     DiAngelo, Natalli          30
```

```
# R Approach

# extract data
name_with_alias <- data_na_tm_ci[, c('na.id', 'na.name', 'an.name')]
names(top_10_actors_within_year_data) <- c('id', 'name', 'alias')

name_with_alias_count <- ddply(name_with_alias, .(id, name), nrow)
head(name_with_alias_count[order(name_with_alias_count$V1, decreasing = T), ], n = 10)
```

```
##             id                name V1
## 32080   662453        Franco, Jesús 78
## 85278  1796694      Savage, Herschel 53
## 88693  1869225         Silvera, Joey 42
## 18024   373754      Clark, Christoph 38
## 52474  1098131 Kronos, Donald Arthur 37
## 85088  1792238      Sarno, Joseph W. 36
## 150004 3213694           Rose, Sasha 32
## 46486   969854          Jeremy, Ron 31
## 35129   728227         Gillis, Jamie 30
## 119799 2540989     DiAngelo, Natalli 30
```

13

```r
# In this question, my approach is as follows:

# Since the main idea is to find the mapping between movie and actor, so it is a good method
# create a temporary table that map the movie name and id to the actor name and id. Since I
# have already created the temporary table in the beginning of this homework, so what I need
# to do now is to use the cast_info table to join them together. I give the final table a
# name of movie_actor.

# THIS SHOULD BE DONE ONLY ONCE
dbGetQuery(con, '
           CREATE TABLE movie_actor AS
           SELECT DISTINCT tm.id movie_id, tm.title movie_title,
           na.id actor_id, na.name actor_name
           FROM title_movie tm JOIN cast_info ci ON tm.id = ci.movie_id
           JOIN name_actor na ON ci.person_id = na.id
           ')

# The function vector_to_sql_id is a tranformation from dataframe-like id vector to sql-like
# id vector. For instance, we can get a vector of 1, 2, 3 in R, but in sql, what we need
# is (1, 2, 3) and nothing else. So this function calls the paste(paste0) function twice to
# combine all the elements together.
vector_to_sql_id <- function(vector) {
  paste0('(', paste(vector, collapse = ','), ')')
}

# First I get the dataframe for actors movies count:

movie_actor_count <-
  dbGetQuery(con, '
             SELECT actor_id, actor_name, COUNT(*) actor_count
             FROM movie_actor
             GROUP BY actor_id
             ')

# Now it is time to select which actor we are going to plot its movie network. Before this,
# I need to mention that, the number of actors are increasing in a very rapid speed. To
# avoid the large number of final set of actors, I will select a very small number of movies
# with respect to the initial actor,  and also a very small number of first set of actors
# related to the initial actor. But it is very computationally expensive, so I only select
# the actor with only 20 movies and minimize the number of first set of actors in the first
# 100 such actors.

# Note here, the words initial, first and second I refer to previously and afterwards are:
# The initial stands for the initial actor I select; The first represents the first set
# of movies related to this initial actor and the first set of actors related to the first
# set of actors; The second set represents the second set of movies related to the first
# set of actors and the second set of actors related to the second set of movies. This
# convention of naming will also be used afterwards when calculating the correspondent
# dataframes and vectors.

# calculate is such a function that calculates the first 100 first set of actor numbers
calculate_first_actor_count <- function(id) {
  # This function first calculates the first set of movies, then calculate the count
  # of first set of actors.

  # calculate the first set of movies
  first_movie <-
    dbGetQuery(con, paste('
                          SELECT movie_id, movie_title
                          FROM movie_actor
                          WHERE actor_id =', id
    ))

  # give the count of first set of actors
  first_actor_counts <-
    dbGetQuery(con, paste('
                          SELECT COUNT(DISTINCT actor_id)
                          FROM movie_actor
                          WHERE movie_id IN', vector_to_sql_id(first_movie[['movie_id']])
    ))

  # get the count
  first_actor_counts[1,1]
}

# call the calculate_first_actor_count function to calculate the first set actors count for
# the first 100 initial actors
actor_compare <- sapply(movie_actor_count[movie_actor_count$actor_count == 20, ][1:100, ][['actor_id']]

# The value for the 90th value is the smallest . So to avoid too large vertices and edges,
# I will select that value. (The table is too large, and 90th value is my observation, And
# I don't attach the table here)
movie_actor_count[movie_actor_count$actor_count == 20, ][90, ]
```

```
##       actor_id    actor_name actor_count
## 54634    76422 Armenta, Mark          20
```

```r
# So I will select actor id 76422

# Now I will calculate the second set of movies and second set of actors following the
# logic of initial actor, first set of movies, first set of actors, second set of movies,
# second set of actors.
```

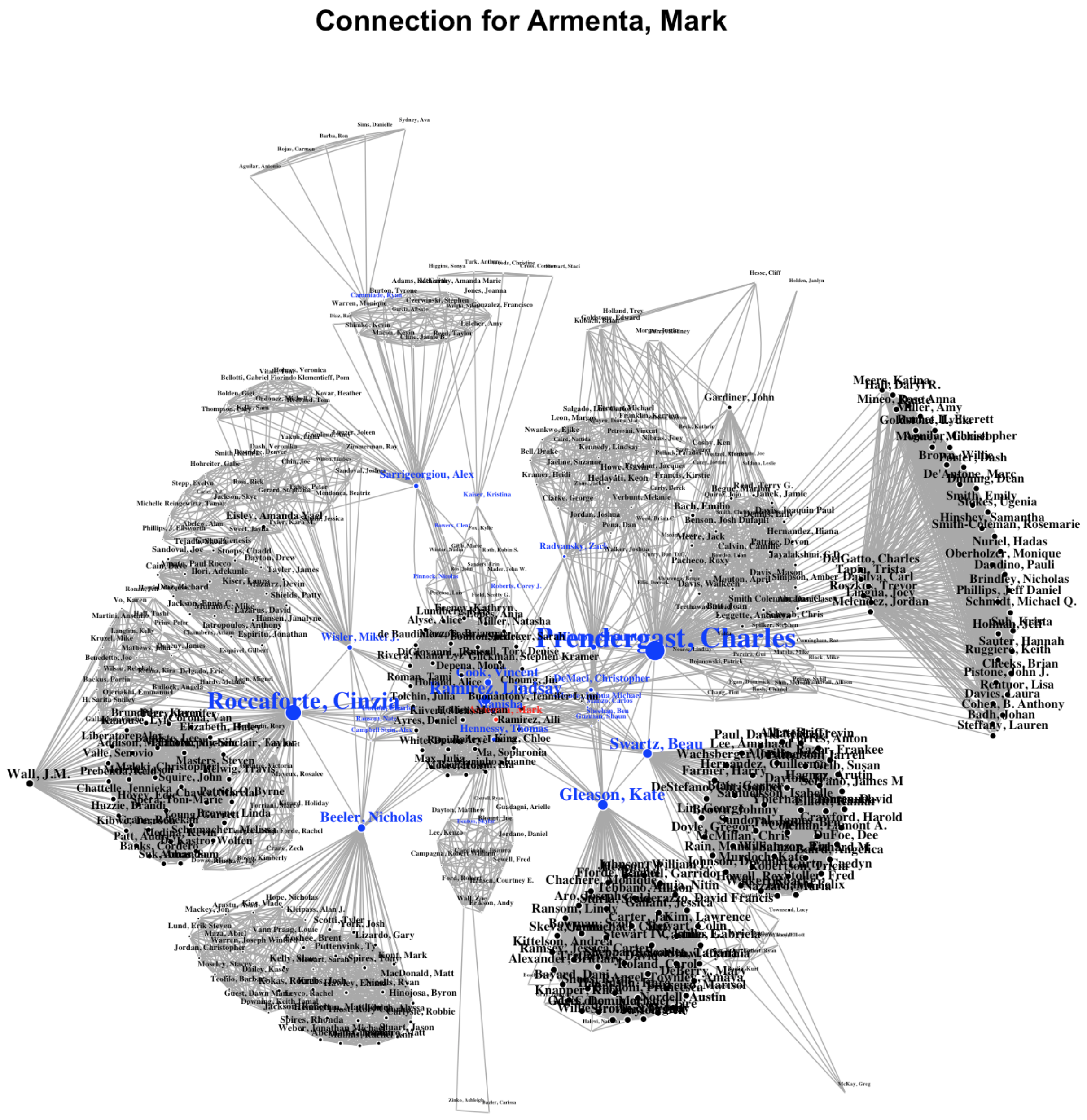
```
# give the initial actor value
initial_actor <- 76422

# calculate first set of movies, based on initial actor
first_movie <-
  dbGetQuery(con, paste('
                        SELECT DISTINCT movie_id, movie_title
                        FROM movie_actor
                        WHERE actor_id =', initial_actor
  ))

# calculate first set of actors, based on first set of movies
first_actor <-
  dbGetQuery(con, paste('
                        SELECT DISTINCT actor_id, actor_name
                        FROM movie_actor
                        WHERE movie_id IN', vector_to_sql_id(first_movie[['movie_id']])
  ))

# calculate second set of movies, based on first set of actors
second_movie <-
  dbGetQuery(con, paste('
                        SELECT DISTINCT movie_id, movie_title
                        FROM movie_actor
                        WHERE actor_id IN', vector_to_sql_id(first_actor[['actor_id']])
  ))

# calculate second set of actors, based on second set of movies
second_actor <-
  dbGetQuery(con, paste('
                        SELECT DISTINCT actor_id, actor_name
                        FROM movie_actor
                        WHERE movie_id IN', vector_to_sql_id(second_movie[['movie_id']])
  ))

# Now I have second hand of movies, then what I should do now is to just extract all the
# actors in each movie and create a link between them, whenever they emerge in the same
# movie. After that, I will combine all the links from different movies together. I know
# there will be some duplications, then I use the ddply function from plyr package to
# remove the rebundancies and give any replicated values weight equal to the number of
# replications.

create_connection_single <- function(name_A, name_B) {
  # First, for each actor A and B, create a link between them

  data.frame(first = name_A, second = name_B)
}

create_connection_movie <- function(id_movie) {
  # Second, for each movie, create all links between all actors, and then combine them together
  # extract relevent actor id by movie id
  id_actors <- dbGetQuery(con, paste('
                                   SELECT DISTINCT actor_id, actor_name
                                   FROM movie_actor
                                   WHERE movie_id =', id_movie
  ))

  # get all actor id
  id_actors <- id_actors[['actor_id']]

  # get the number of all actors
  N <- length(id_actors)

  # I will do two loops here, the inner loop and the outer loop. Based on these two lapply,
  # I will create a link for each combination of actors
  outer_result <- lapply(1:N, function(i) {
    inner_result <- lapply(i:N, function(j) {
      if (i!= j) create_connection_single(id_actors[i], id_actors[j])
    })
    do.call(rbind, inner_result)
  })

  # combine the results into a big dataframe
  final_result <- do.call(rbind, outer_result)

  # return result
  final_result
}

create_connection_movies <- function(second_movie) {
  # Third, based on a list of movies, I will call create_connection_movie to get the
  # dataframe for each movie, and then combine the dataframes together to make the
  # final dataframe. Of course I use ddply to remove duplicates and counts the
  # duplications as a new variable, weight.

  # get movie vector
  id_movies <- second_movie[['movie_id']]

  # call create_connection_movie to get a list of dataframes of links related to
  # each movie
  each_movie_list <- lapply(id_movies, create_connection_movie)

  # combine them together
  movie_total_connection <- do.call(rbind, each_movie_list)

  # use ddply to remove duplicates
  movie_unique_connection <- ddply(movie_total_connection, .(first, second), nrow)
  names(movie_unique_connection)[3] <- 'weight'

  # return value
  movie_unique_connection
}
```

```r
# Call the create_connection_movies with the input we get previously, second_movie
connection_data <- create_connection_movies(second_movie)

# The last part is to plot the results out. Since the dataframe I get now is particularly
# designed for this use, no further transformation or processing is required,
# except for I need to convert it to the object class of this package. Then call the plot
# function with proper parameters, everything is done.

library(igraph)
```

```
##
## Attaching package: 'igraph'
##
## The following objects are masked from 'package:stats':
##
##     decompose, spectrum
##
## The following object is masked from 'package:base':
##
##     union
```

```r
# convert to graph.data.frame
connection_network <- graph.data.frame(connection_data, directed = F)

# give different groups of actor different color. Since we have three groups: initial actor,
# first set of actors and second set of actors, I will give them different colors: red, blue
# and green respectively.
V(connection_network)$color <-
  ifelse(V(connection_network)$name %in% initial_actor, 'red',
         ifelse(V(connection_network)$name %in% first_actor[['actor_id']], 'blue', 'black'))

# get the name of each actors by their id
V(connection_network)$new_name <-
  dbGetQuery(con, paste('
                        SELECT name.name
                        FROM name
                        WHERE name.id IN', vector_to_sql_id(V(connection_network)$name)
  ))
V(connection_network)$new_name <- V(connection_network)$new_name[[1]]

# other configurations, including their degree, size, label size, vertex color and so on

# get the degree
V(connection_network)$degree <- degree(connection_network)
# set the node size
V(connection_network)$size <- V(connection_network)$degree / 30
# set the label font size
V(connection_network)$label.cex <- 1.0 * V(connection_network)$degree /
  max(V(connection_network)$degree) + 0.2
# set the label color
V(connection_network)$label.color <-
  ifelse(V(connection_network)$name %in% initial_actor, 'red',
         ifelse(V(connection_network)$name %in% first_actor[['actor_id']], 'blue', 'black'))
```

```r
# plot results out
plot(connection_network,
     # the layout is chosen to best represent the data
     main = 'Connection for Armenta, Mark',
     layout=layout.kamada.kawai,
     vertex.label.dist = 0.1,
     vertex.frame.color = 'white',
     vertex.label.font = 2,
     vertex.label = V(connection_network)$new_name
)
```



```r
# For different colors, I note it here: red is for the initial node, blue is the first set of
# actors, black is the second set of actors.
```

14

```r
# This is a subquery, since I need the movie stars, I will first select the actors for movies
# with nr_order less or equal than 5. Then use these actors to get the final result by
# join it to the other tables
dbGetQuery(con, '
           SELECT title.id, title.title, COUNT(DISTINCT ci.person_id) actor_count
           FROM title JOIN cast_info ci on title.id = ci.movie_id
              JOIN (SELECT DISTINCT ci.person_id mvnr_id
                    FROM title_movie tm JOIN cast_info ci ON tm.id = ci.movie_id
                    WHERE ci.nr_order IN (1, 2, 3, 4, 5)
                    ) mvnr ON ci.person_id = mvnr.mvnr_id
           WHERE ci.role_id IN (1,2)
              AND title.kind_id = 2
           GROUP BY ci.movie_id
           ORDER BY actor_count DESC
           LIMIT 10
           ')
```

```
##         id                  title actor_count
## 1   729678       General Hospital         576
## 2  1404883        One Life to Live         419
## 3   449619       Days of Our Lives         394
## 4   122527          Another World         361
## 5  1941002       The Guiding Light         343
## 6    76799         All My Children         272
## 7   147610      As the World Turns         267
## 8  1970321 The Laurel and Hardy Show         253
## 9  1917967       The Edge of Night         231
## 10 1572213     Retrosexual: The 80's         200
```