Movie Data Analytics

Lab 3 - October 7, 2015

1. Overview

1.1 Group Members

- Siying Zhang (siying.zhang@pitt.edu);
- Chukun Xia(CHX26@pitt.edu);
- Abdulaziz Abdulrahman Almuzaini (aaa169@pitt.edu);
- Yue Li (YUL134@pitt.edu);
- Abhishek Mukherjee (ABM84@pitt.edu)

1.2 Data Source

- **Movie rating list:** http://boxnumbertwo.com/MovieData/ratings.list
- **Movie budget form:** http://boxnumbertwo.com/MovieData/budget.html

2. Introduction

Lab 3 have three parts, the first one is using python to store data of the movie rating list data into sqlite 3. The second part is similar to the part 1, but has the different data source. In these two parts, we have to stylised original data firstly, which will make a big difference to the later data analytics. The final part of this lab is using R to analyse data in two tables.

3. Detailed Explanation

3.1 Store rating data

The most complicated part of this python program is how to stylised the movie name in the list, because there are lots of special characters at the head of String or some other modifications at the end of it (E.g. "{xxxxx}"). We made a regulation of the movie name "movieName (Year)", which is also adopted by another data source.

We use six reglex (regular expression) to stylised movie names:

```
# remove special characters "2 or more characters, not A-Z a-z 0-9 { } \"
wordReplace1 = re.sub('[^A-Z^a-z^0-9^{^}^\(^\)^\"^ ]{2,}','',line[3])
#remove "{.*}" behind of the string (423431 results / 3376 match)
wordReplace2 = re.sub(' {.*}', '', wordReplace1)
# remove '(TV)' '(V)' () (422585 results)
wordReplace3 = re.sub(r' \(\D*\)', '', wordReplace2)
# change (2007/I) into (2007) (422303 results / 3572 match)
wordReplace4 = re.sub(r'\/.{1,3}\)',')', wordReplace3)
# remove the \" at the begining
wordReplace5 = re.sub(r'^\"','', wordReplace4)
# handle the last \" and the begining 'space'
wordReplace6 = wordReplace5.replace("\" "," ").lstrip()
```

There are still some rows with the same movie name. Fortunately, they are ordered by the ascending votes. So we choose the first row of them.

```
# store the value into dict, if there are movies with the same name then chose the first one (Max vot
if wordReplace6 not in movies:
    movies[wordReplace6] = {'rating': line[2], 'votes': line[1], 'year': wordReplace6[-5:-1]}
```

The way to store processed data into database is the same as lab2.

3.2 Store budgets data

Since this web page is in the form of HTML, we adopt "beautiful soup" to extract data.

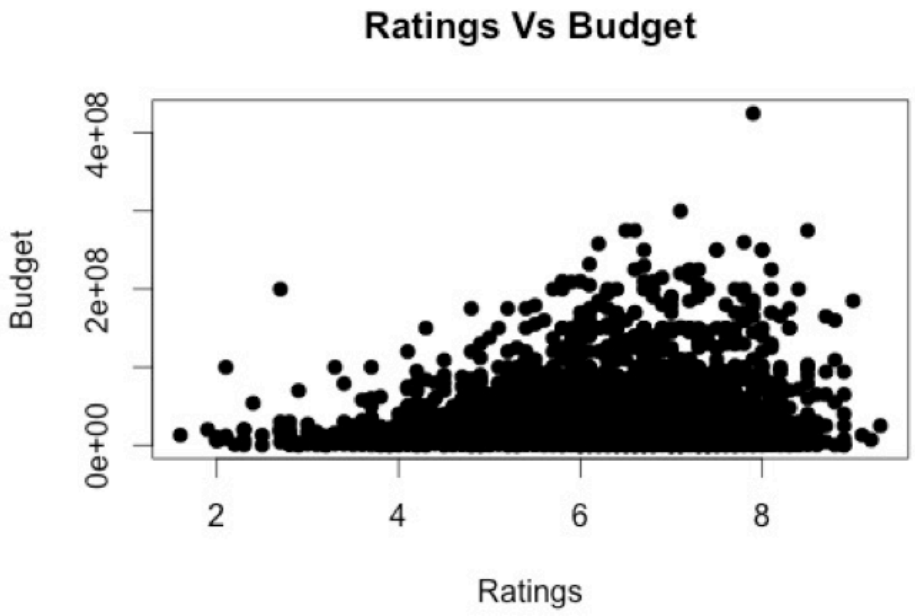
There are six attributes in this database, where year is extracted from the date column. All data is from the first table, so the loop is in soup('table')[0].

To extract the data from every column, we use findall('td')[x:y] to locate them. There are still some parts needing to be removed/replaced.

```
key = ''  # empty string to store the movies names
if len(row.findAll('td')) != 0:    ## store just the occupied table data
    for td in row.findAll('td')[2:3]:  # find the td for movies names which is in td 2
        key+=td.text    # store the text of that data cell - movies name

    for td in row.findAll('td')[1:2]:    # looping through td 1 "The date"
        key+=' ('+(str)(td.text)[-4:]+')'  ### store the years (2009)
        yearDic[key] = (str)(td.text)[-4:]
        if(td.text[1]=='/'):    ### store the months 5/10/2014 if its two digits  -> 5
            monthDic[key] = td.text[0]
        else:
            monthDic[key] = td.text[0:2]  # months 10/3/2013  -> 10

    for td in row.findAll('td')[3:4]:   ## looping through budget
        budgetValue = []   # empty list for elimentation
        budgetValue = td.text.replace(',','')  # replacing the comma with nothing
        budgetValue=budgetValue.replace('$','')  # removing $
        budgetDic[key]=(str)(budgetValue)  # store them in a dic using movies names as keys and make them String

    for td in row.findAll('td')[4:5]:   # looping through domestic colimns
        domesticValue=[]
        domesticValue = td.text.replace(',','')
        domesticValue=domesticValue.replace('$','')
        domesticDic[key]=(str)(domesticValue)   ## storing

    for td in row.findAll('td')[5:6]:  ## worldwide budget
        worldwideValue=[]
        worldwideValue = td.text.replace(',','')
        worldwideValue = worldwideValue.replace('$','')
        worldwideDic[key] = (str)(worldwideValue)   # stroing to dic
```

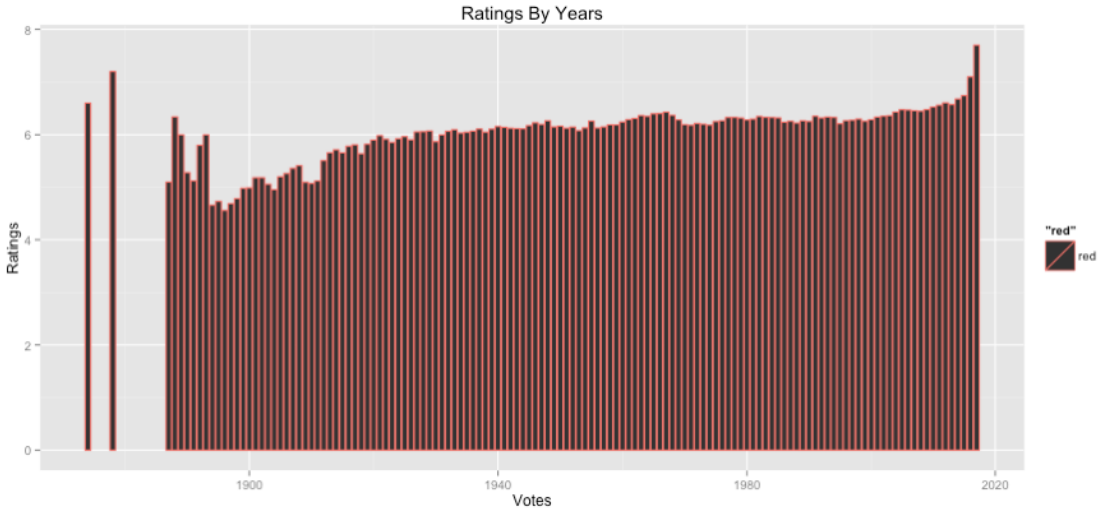
3.3 R part

3.3.1 Ratings Vs Budget



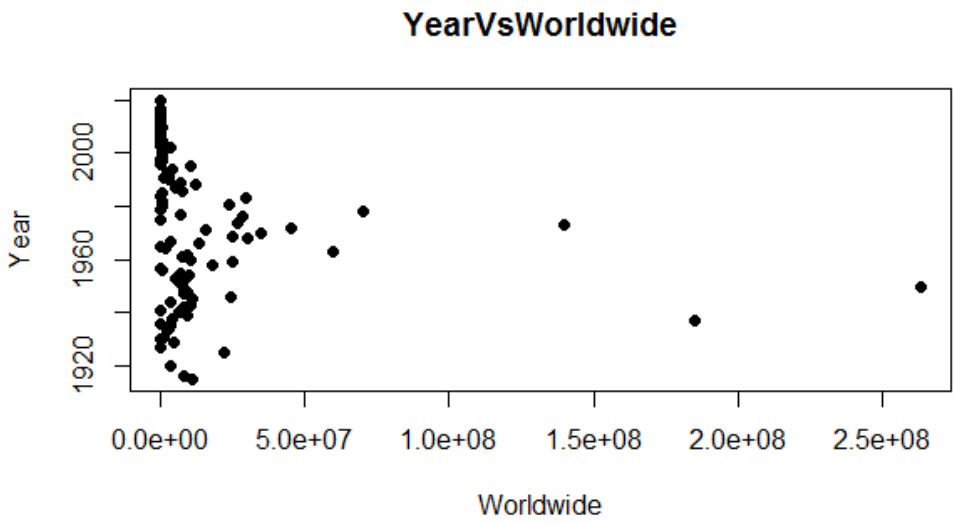
In this graph, we are able to determine the correlation between ratings and budget. It shows the budget increases as the ratings increases as well. It seems steady until 8.5 and then decreases significantly. Although there are some outliers in some points, these outliers are not taken into consideration to judge the whole graph. Some movies have less rating but have high budget. What we assume is that some movies are not popular as others in the Internet, so that could have led to less ratings because people might not had a chance to rate them.

3.3.2 Ratings By Years



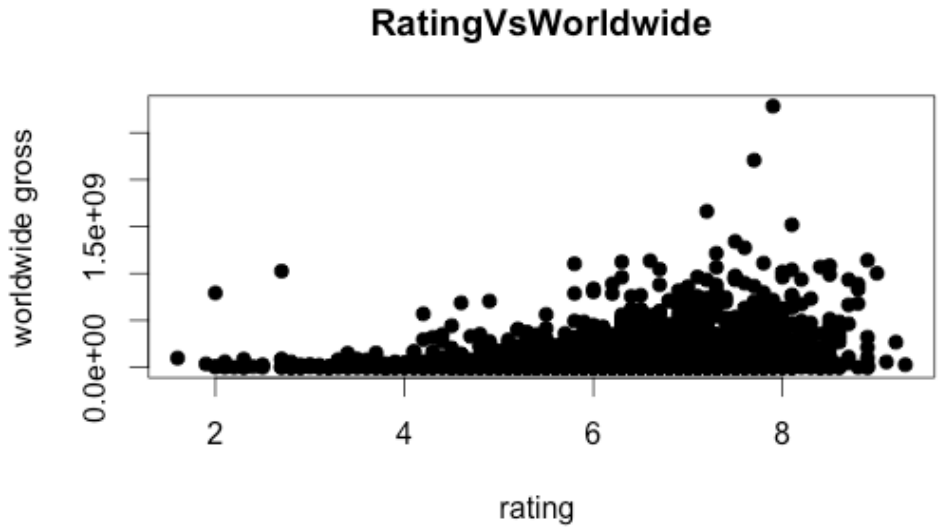
This graph is generated by qplot. It indicates the tendency of ratings over these years. As we can see, around 1880s, the ratings are high. But after that decade, ratings suddenly dropped. From 1900 to 2015, the ratings increase in general, which I think has two means, one people relaxed their rating standard, one movies quality became better .

3.3.3 Year Vs Worldwide



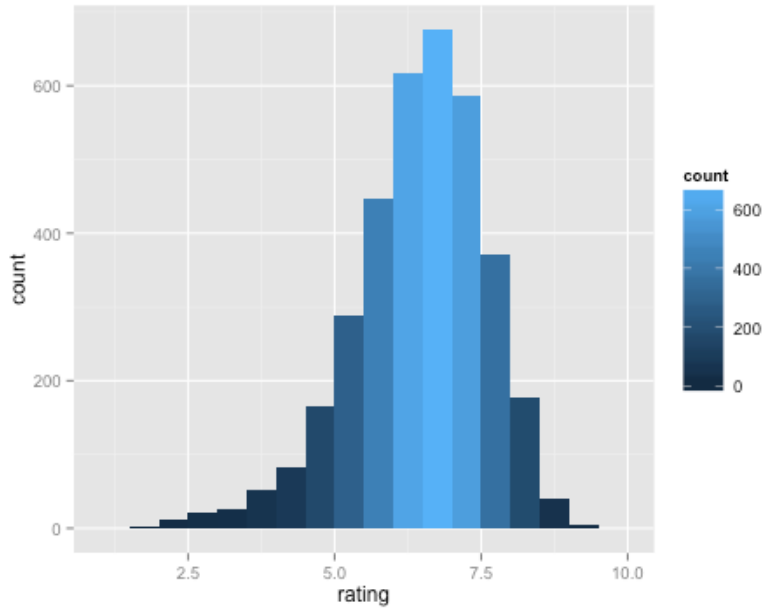
From this graph we can see the worldwide gross income from movies for the time period 1920 to 1960 showed a little increase. But then for the time period 1960 to 1980 the worldwide gross went up suddenly and remained quite high. Again after 1980 the worldwide gross decreased and it remained almost constant till the year 2020.

3.3.4 Ratings Vs Worldwide Gross



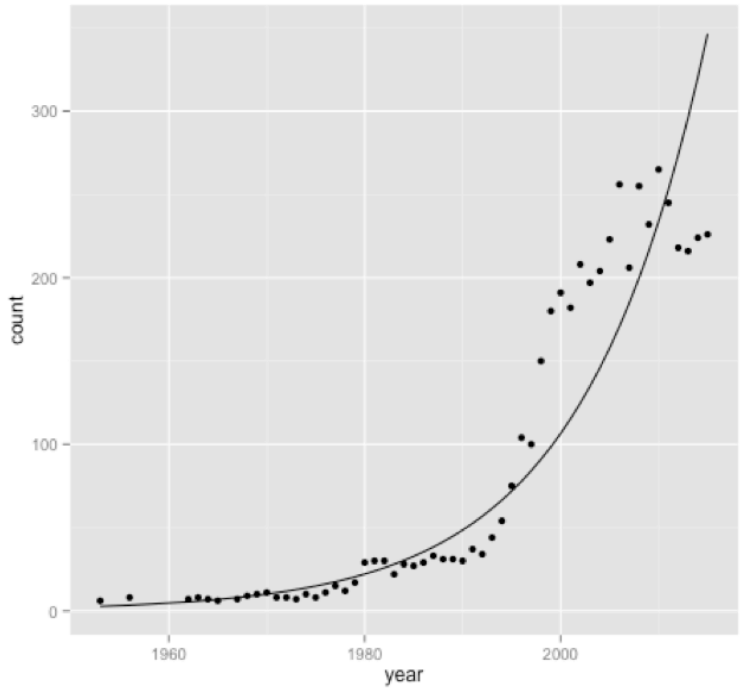
From the graph we can see the relation between rating and worldwide gross of each movie. Generally, movies with higher ratings have higher worldwide gross. People are willing to spend money on high reputation movies. But there are a few exceptions, there are some movies with low rating but have a relatively high worldwide gross, which can be explained that these movies have good advertisement or famous stars that attracting people at first, so they have a good box-office income.

3.3.5 Rating Counts



We use ggplot to make a graph showing the counting of ratings. From this graph, we can see that most ratings are around 7. We can assume that when people rate a movie 7 which means he feel it is normal, not bad but not that satisfied. So when the rating is above 7.5, that means the movie has good feedback, vice versa. From the graph we can also get another conclusion that there are more films not satisfying people.

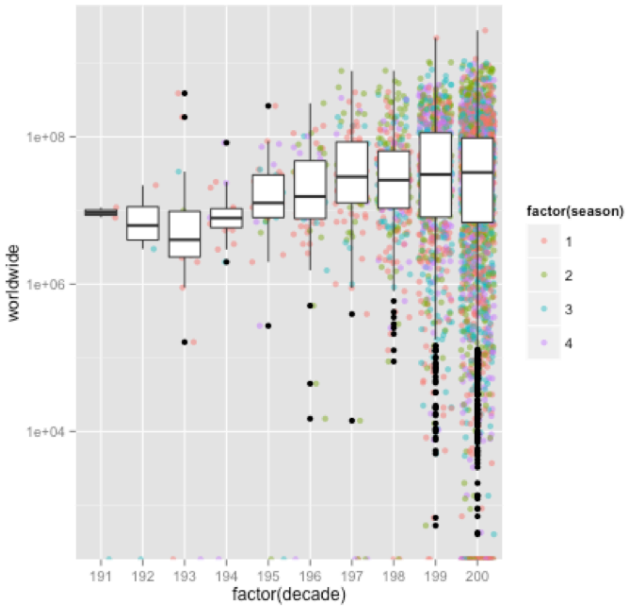
3.3.6 Number of movies By Year



This graph shows relation between the ***year*** and ***number of movie*** released in that year. They are roughly in exponential relation, which means the total number of movies increases exponentially yearly. The year after 2015 and the year in which there are less than 5 movies released are excluded

from the data frame since they are not representative, and the linear regression is significant statistically.

3.3.7 Average worldwide gross By Decades



This graph shows relation between ***average worldwide gross*** and ***decades***, as well as rough relation between ***average worldwide gross*** and different ***seasons*** in one decades. Firstly, the average worldwide gross increases among years, they are in linear relation. However, there are an obvious decrease in 1930's, it could be affected by The Great Depression. We can also see that the boxes are becoming bigger and bigger, which means the number of movies getting larger, their worldwide gross varies in a larger scale as well, which could be explained as the quality of movies varies in a lager and larger scale as well. There are more "better" movies and also more "worse" ones. Secondly, from the points behind the boxes we can see that green ones, which stand for movies released in the second season, are a little more likely to become the blockbuster.

4. Summary

It's the first time for us to get to know R, which is so useful in generating graph. Graph could visualise the relationship between different items. In R, plot can generate a normal graph, which is very clear although simple. qplot will generate a graph with more beautiful looks.

Beautiful Soup is really a useful tool to get the content in a html page. It could let us find exact tag easily.

Note

Because of the encoding problem, we translate unicode string to 8-bit string, which was a kind of stumbling block in the way to store data.

```
con = lite.connect(directoryForDB)
#use 8-bit strings instead of unicode string
con.text_factory = str
```