Project Report

Music Data Analysis

Peizhou Guo, pg1534

Xiaojie Zha, xz1776

Yang Shi, ys2843

11th May, 2017

Introduction

In this project, we used several datasets, by joining different datasets together, we may get the source data for further analysing of a specific purpose. The datasets we used for this project is listed below, including the name of dataset and the link to reach the dataset:

1. MSD(Million songs dataset)

https://labrosa.ee.columbia.edu/millionsong/

Subset 1: SecondHandSongs dataset

https://labrosa.ee.columbia.edu/millionsong/secondhand

Subset 2: musiXmatch dataset (lyric)

https://labrosa.ee.columbia.edu/millionsong/musixmatch

Subset 3: tagtraum genre annotations

http://www.tagtraum.com/msd_genre_datasets.html

Subset 4: The Echo Nest Taste

https://labrosa.ee.columbia.edu/millionsong/tasteprofile

2. Yahoo! Music User Ratings of Musical Artists

http://webscope.sandbox.yahoo.com/catalog.php?datatype=r

3. NRC Word-Emotion

http://saifmohammad.com/WebPages/NRC-Emotion-Lexicon.htm

4. Google Trends API

https://github.com/GeneralMills/pytrends

5. Amazon Review

http://jmcauley.ucsd.edu/data/amazon/links.html

In this project, we use different tools learned from this class to achieve different goals:

1. HPC

We use HPC service provided by NYU as the server to process our data. The Hadoop environment has already installed on dumbo in HPC.

2. Apache Pig

Apache Pig is a high-level platform for creating programs that run on Apache Hadoop, it's a main tool we used for this project.

3. Apache Spark

Apache Spark is an open-source cluster-computing framework. In this project, we used spark scala and spark SQL to for some analysis.

4. Apache Zeppelin

Apache Zeppelin is a web-based notebook that enables interactive data analytics. In Zeppelin, we can make beautiful data-driven, interactive and collaborative documents with SQL, Scala and more languages. In this project, we use Zeppelin as a environment to generate and to present our statistical analysing result.

5. R

R is an open source programming language and software environment for statistical computing and graphics. We use R and some R API to generate some fancy diagram and Location presentation on Google Map.

6. Python

Python is a widely used high-level programming language for general-purpose programming. In this project, we use python as a tool to do transfer and some other jobs for some relatively small dataset, to simplify the dataset for easier analysing.

Our goals:

1. Let people know more about the music they are listening

With the fast development of electronic device and record equipment, more and more people can record their own songs and publish them. Since different people tend to listen different music genre, the increasing of artists in different genre brings more people start to listen music. In this project, we use big data technic to show the feature of music based on music genre.

2. Give people better recommendation

Many people choose to use music app like Last.fm, Spotify, SoundCloud to explore the music world. In these application, a recommending system is provided. Usually, the system will give user a personal list each day based on user's listening history. In this project, we do some job to find those artists or albums which are most popular for each music genre, as a part of a recommending system. In the future, we will do the research around the pattern of user, try to find some common patterns or features for certain group of people, for recommending them music more precisely.

3. Show the trends of music development

Nowadays, music industry is a growing and prosperity field which has huge profit for investing. There are thousands of albums issued every year. We want to show the trends of music industry development by analysing million songs dataset. Our result, the graph of the number of albums issued per year since 1920 reflects the market growth of music industry.

4. Analyze which music genre has higher business value

We try to figure out a best solution for artist, investors to get more profit in music industry. We find out that different music genre has different business value and some certain genre is much more popular than others and can also make artist more popular. We compare the top 5 popular genre by play count and search heat from google and we find the most profitable genre of all time.

The trend of music development

Brief introduction:

In this part, our goal is to find the relation between popularity of songs, artist, genre and time. By dig into the trends of music industry development, genre influence on popularity of songs and artists and search heat of artist and genre according to time, we find the genre which has the most business value, the trends of albums selling and search heat of an artist etc.

In this part, MSD and google trends api are used as main source of research. Pig is used as the tool for analysing the dataset.

Preparing work:

1. Data from google trends was downloaded by queries using api
2. MSD dataset, train_triplet dataset(play count of each song by user)
3. All the data are uploaded to http://babar.es.its.nyu.edu/.

Implementation with pig

First, to prove the music industry is a profitable and fast-growing market. We shows the trends of music industry development by the number of album issued per year. We did this by grouping the album ID and year and output the count of the distinct tuples, and order the result by year. Then we can see clearly the number of albums issued from 1920 till now according to year.

```
songs_2 = GROUP songs BY AlbumID;

songs_3 = FOREACH songs_2 {

      top_rec = LIMIT songs 1;

      GENERATE FLATTEN(top_rec) as (AlbumID, Year);

};

songs_4 = GROUP songs_3 BY Year;

songs_5 = FOREACH songs_4 GENERATE FLATTEN(group) as Year, COUNT(songs_3.AlbumID);
```

Then, to find the genre with most business value, we group the million songs by their genre to output the total number of songs in each genre, and find out the popular genres of all time. The result is ordered by number in desc. Now we know the top five popular genre which has most business value. The next step we are going to analyze top 5 genre one by one.

```
song_tag_join = join song_tag by TrackID, songs by TrackId;

song_tag_id = foreach song_tag_join generate SongID, AlbumName, ArtistName,Title,Year,tag;

song_count_1 = GROUP songPlay_count BY SongID;

song_count = FOREACH song_count_1 GENERATE FLATTEN(group) as SongID, SUM(songPlay_count.counts)
as counts;

songAndcount = JOIN song_count by SongID, song_tag_id by SongID;

genreCount = GROUP songAndcount BY tag;
```

```
genreCount2 = FOREACH genreCount GENERATE FLATTEN(group) as tag, SUM(songAndcount.counts) as
counts;

genreCount3 = ORDER genreCount2 BY counts;
```

We analyze each of the top 5 popular genre individually. First we group MSD with user play count record and then filter the result by each genre. By add the play count together, we can get the total play count of a genre according to year.

```
song_count_1 = GROUP songPlay_count BY SongID;

song_count = FOREACH song_count_1 GENERATE FLATTEN(group) as SongID, SUM(songPlay_count.counts)
as counts;

songAndcount = JOIN song_count by SongID, song;_tag_id by SongID;

genreCount = GROUP songAndcount BY (Year,tag);

genreCount2 = FOREACH genreCount GENERATE FLATTEN(group) AS (Year, tag),
SUM(songAndcount.counts) as counts;

genreCount3 = FILTER genreCount2 BY tag == 'Country';

genreCount4 = ORDER genreCount3 BY Year;
```

To find out the effect of music genre on an artist, we calculated a rank according to total song play count of artist. This rank can reflect the popularity of artist which then compared to google trends result

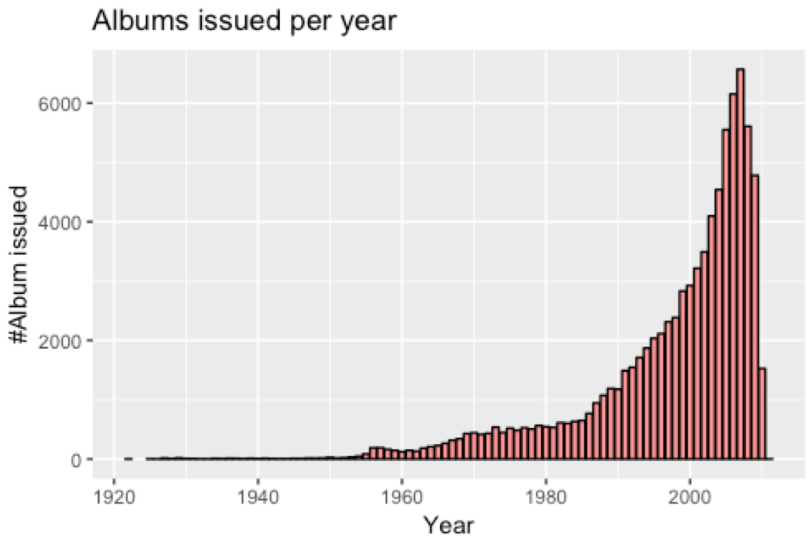
```
song_count_1 = GROUP songPlay_count BY SongID;

song_count = FOREACH song_count_1 GENERATE FLATTEN(group) as SongID,COUNT(songPlay_count.track)
as counts;

songAndcount = JOIN song_count by SongID, songs by SongID;

song_count = FOREACH songAndcount GENERATE ArtistID, ArtistName,counts;

song_count2 = ORDER song_count BY counts;

song_artist = GROUP song_count2 BY ArtistName;

song_artist2 =FOREACH song_artist GENERATE FLATTEN(group) as ArtistName,
SUM(song_count2.counts) as counts;

song_artist3 = ORDER song_artist2 BY counts DESC;

song_output = LIMIT song_artist3 10;
```

Generate graph using R:

For Album issued per year,

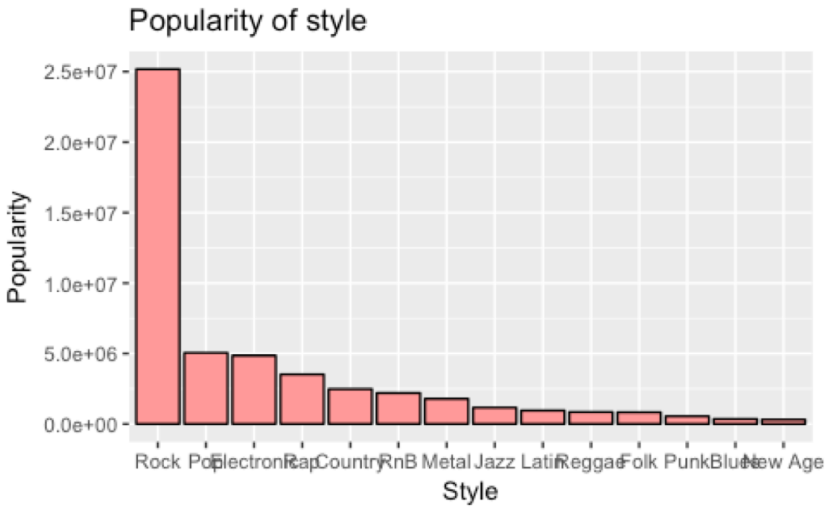
```
q <- ggplot(album,aes(year,count)) + geom_bar(stat = "identity",fill="#FF9999", colour="black")
+ labs(x = "Year", y = "#Album issued",title = "Albums issued per year")
```

print(q)



For the play count of different genre,

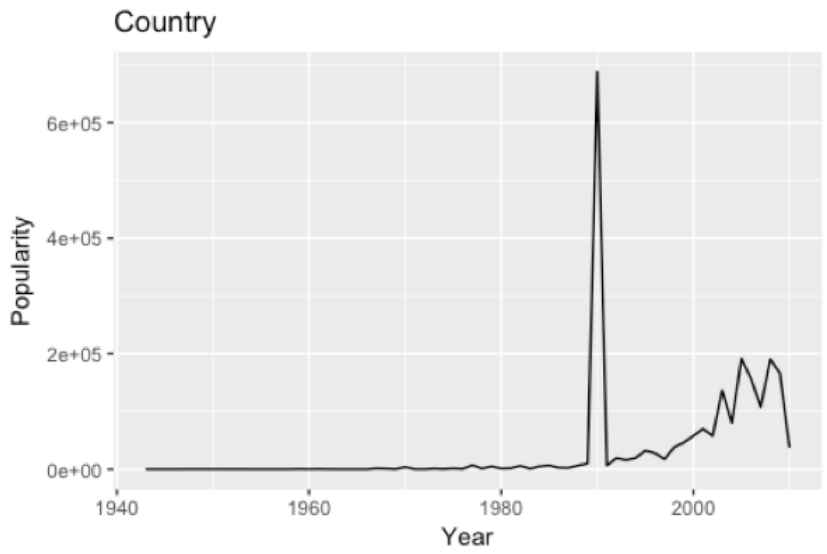
```
q2 <- ggplot(tag_count, aes(x = reorder(tag,-count),y=count)) +

  geom_bar(stat = "identity",fill="#FF9999", colour="black") + labs(x = "Style", y =
"Popularity",title="Popularity of style")

print(q2)
```



For the play count of each of 5 genres according to year,

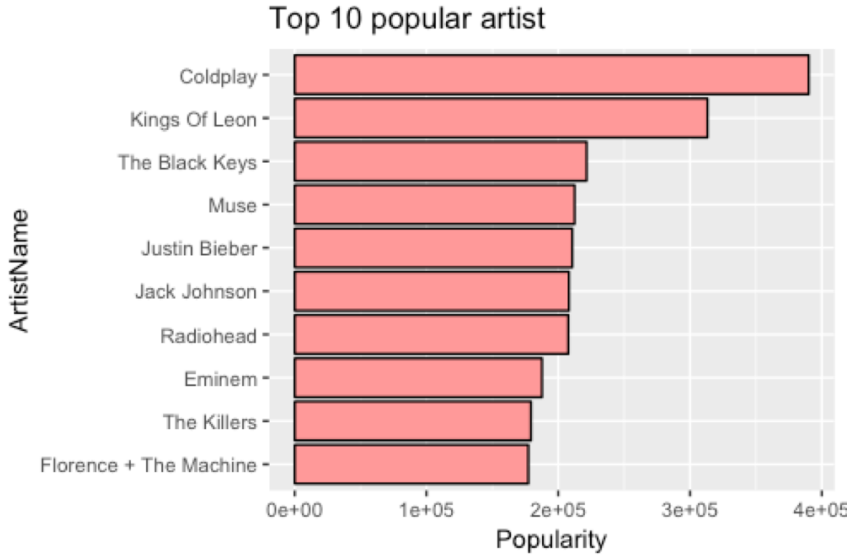
```
q7 = ggplot(country, aes(year, count)) + ggtitle("Country")+geom_line(stat = "identity")+
labs(x = "Year", y = "Popularity")
```

print(q7)



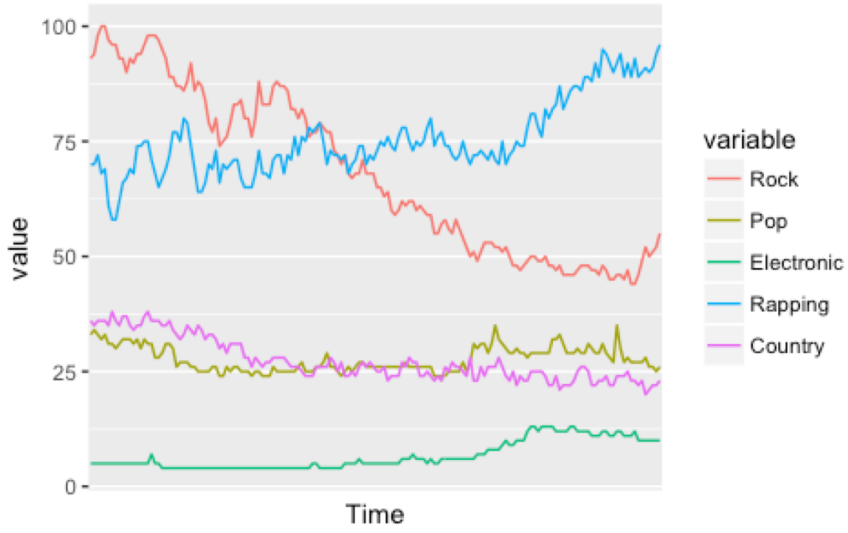
For the top 10 artist according to total play count of their all songs

```
q8 <- ggplot(artist,aes(reorder(artist,count),count))  + ggtitle("Top 10 popular artist")

q8 + geom_bar(stat="identity",fill="#FF9999", colour="black")+ coord_flip() + labs(x =
"ArtistName", y = "Popularity")
```



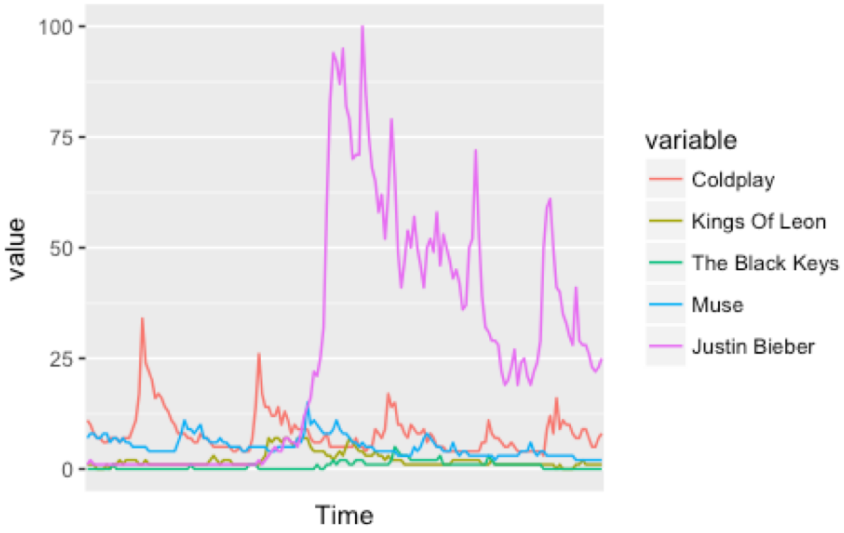
For the search heat of 5 genre,

```
genre <- melt(genre, id.vars=c("Month"))

top5artist <- melt(top5artist, id.vars=c("Month"))

q9 <- ggplot(genre,aes(Month,value,group = variable, color = variable)) + geom_line(stat =
"identity")+ scale_x_discrete(name = "Time", breaks=1)

print(q9)
```



For search hear of 5 artist,

```
q10 <- ggplot(top5artist,aes(Month,value,group = variable, color = variable)) + geom_line(stat
= "identity")+ scale_x_discrete(name = "Time", breaks=1)

print(q10)
```



Emotion, words and genre

Brief introduction:

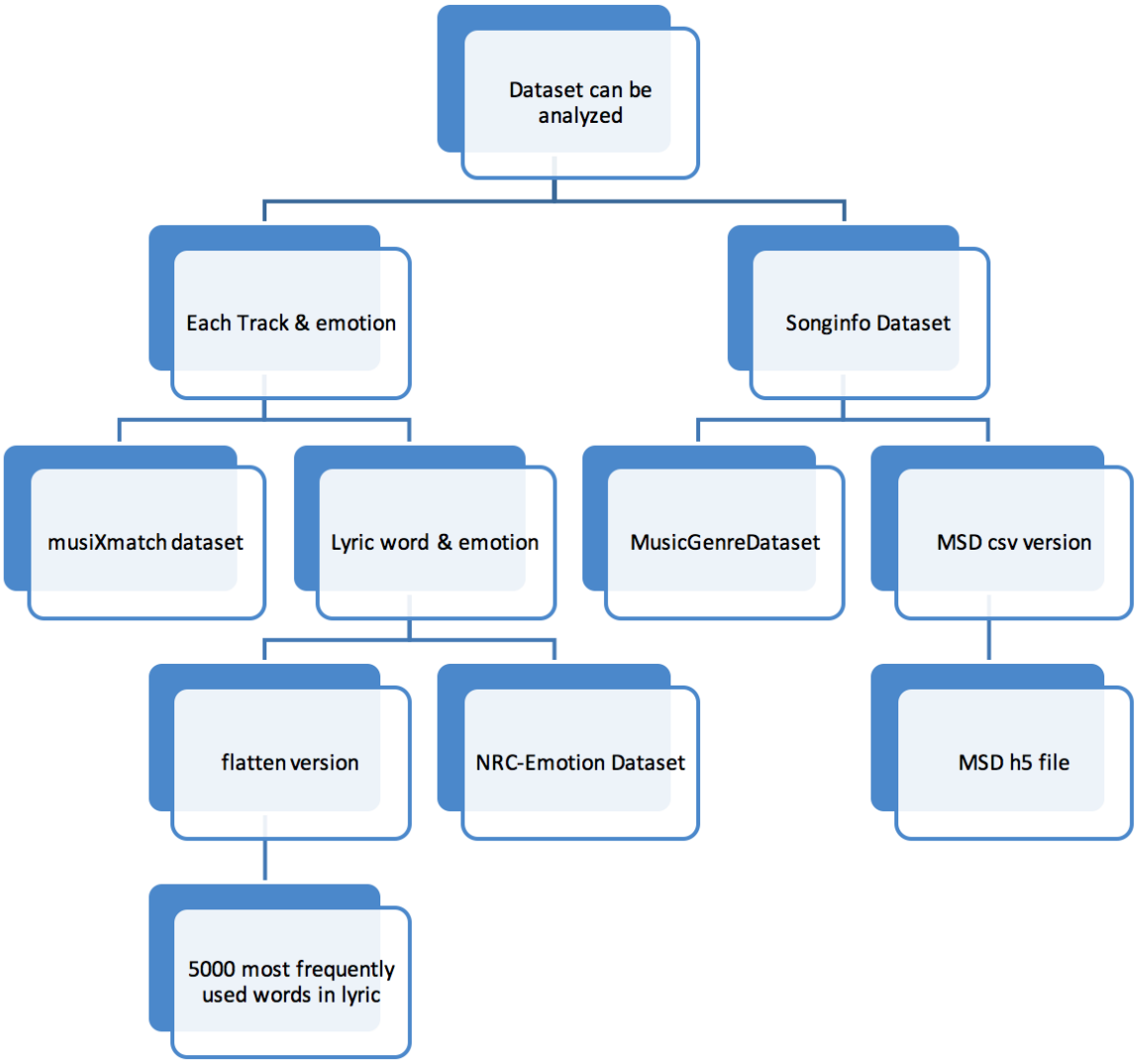
In this part, we use musiXmatch dataset, tagtraum genre annotations and NRC Word-Emotion dataset to analyse the emotion shown by each music genre.

musiXmatch is a dataset to count the most frequent 5000 words for every song in MSD dataset. In this dataset, the first part is a list of most frequent 5000 words among all songs, and the second part is the song id and a count for each word. For the second part of this dataset, the format for each song is : *(TID,MXMID, word1Index:cnt, word2Index:cnt,....,wordnNndex:cnt).*

tagtraum genre annotations is a dataset to tag which music belongs to which genre. For the original dataset we extract from MSD, there is no genre tag, so we use this dataset to further strengthen the MSD data with a genre attribute.

NRC Word-Emotion a list of English words and their associations with eight basic emotions (anger, fear, anticipation, trust, surprise, sadness, joy, and disgust) and two sentiments (negative and positive). The annotations were manually done by crowdsourcing. In this project, we use this dataset to join with the lyric dataset for analysing the emotion of songs.

Preparing work:



To get the dataset which can be analysed in this part, we combine the data from several different datasets.

First, we picked out the 5000 most frequently used word in lyric dataset, wrote a python script to fatten the words in a format we want: [index, word]. Then we join NRC-Emotion Dataset with these 5000 words using word, to ignore those words which don't appear in the emotion dataset. For this part, the goal is analysing emotion, so those "meaningless" words are negligible.

Then, we join the dataset we got from previous step with musiXmatch dataset using the index of words, to get a flatten dataset with format [songid, word: count, emotion]. Using this dataset, we can know the emotion ration for each song.

But now, we still cannot connect emotion with genre. So, for another of source data, we use the MSD dataset which we extracted from MSD h5 file to join with MusicGenreDataset. Since some songs in MSD do not have a genre classification, we can reduce the size of MSD dataset in this step.

In the end, we can finally join this two datasets together and do the rest of analysing relationship between genre and emotion. We calculate the score for each genre by count the corresponding word appearance in each genre.

Implementation:

(1) Join the genre and MSD dataset.

.....

```
songinfo = FOREACH song GENERATE SongID, TrackID, AlbumID, AlbumName, ArtistID, ArtistName,
ArtistLatitude, ArtistLongitude, ArtistLocation, Duration, KeySignature, Tempo, Title, Year;

genre = FOREACH geninfo GENERATE genre, track_id;

fullinfo  = JOIN songinfo BY TrackID, genre BY track_id;
```

.....

(2) Join lyric and NRC emotion dataset, ignore those useless words.

.....

```
LE = JOIN lyric by wordIndex, emotion by wordIn;

LE1 = FOREACH LE GENERATE lyric::TrackID as TrackID, lyric::MxMTrackID as MxMTrackID,
lyric::wordIndex as wordIndex, lyric::Cout as Cout, emotion::word as word, emotion::emotion as
emotion;
```

.....

(3) Join the musiXmatch dataset with the lyric word-emotion list.

```
Emo = LOAD 'NRC-Emotion-Lexicon-Wordlevel-v0.92.txt' USING PigStorage('  ') as (Eword:
chararray, emotion: chararray, yes: int);

Emot = FILTER Emo BY (yes == 1);

W = LOAD 'mxm_dataset_word.txt' USING PigStorage(',') as (index: int, word: chararray);

WE = JOIN Emot BY Eword, W BY word;

WEoutput = FOREACH WE GENERATE Emot::Eword, Emot::emotion;

Word = LOAD 'mxm_dataset_word.txt' USING PigStorage(',') as (index: int, word: chararray);

Res = JOIN WE by Emot::Eword, Word by word;

Resu = FOREACH Res GENERATE WE::W::index, WE::W::word, WE::Emot::emotion;
```

```
Resul = ORDER Resu by WE::W::index;

store Resul into 'lyricWordEmo' USING PigStorage(',');
```

(4) Join the previous dataset with the MSD dataset and genre dataset.

……

```
songinfo = FOREACH song GENERATE SongID, TrackID, AlbumID, AlbumName, ArtistID, ArtistName,
ArtistLatitude, ArtistLongitude, ArtistLocation, Duration, KeySignature, Tempo, Title, Year;

LE = LOAD 'TrackWordEmotion/part-r-00000' USING PigStorage(',') as (TrackID: chararray,
MxMTrackID: chararray, wordIndex: int, Cout: int, word: chararray, emotion: chararray);

LE1 = GROUP LE by (TrackID, emotion);

LE2 = FOREACH LE1 GENERATE group.TrackID as TrackID, group.emotion as emotion, SUM(LE.Cout) as
EmoCoun;

Gen = JOIN song by TrackID, LE2 by TrackID;

Genr = GROUP Gen by (genre, emotion);

Genre = FOREACH Genr GENERATE group.genre as genre, group.emotion, SUM(Gen.LE2::EmoCoun) as
EmoCoun;

Result = Order Genre by genre, EmoCoun;

store Result into '/genreSummary' USING PigStorage(',');
```

(5) We also count the top frequent words for each genre.

……

```
WT = GROUP W by (TrackID, word);

WT1 = FOREACH WT GENERATE group.TrackID as TrackID, group.word as word, SUM(W.Cout) as Cout;

Res = JOIN song by TrackID, WT1 by TrackID;

Resu = GROUP Res by (genre, word);

Resul = FOREACH Resu GENERATE group.genre as genre, group.word, SUM(Res.WT1::Cout) as Cout;

Result = Order Resul by genre, Cout DESC;
```

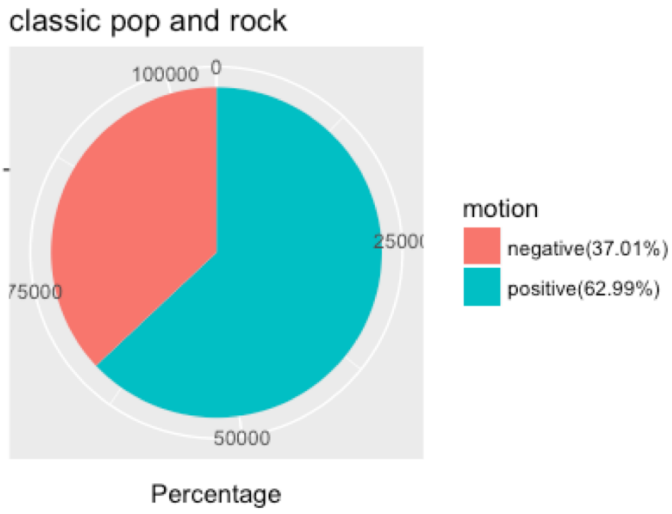
……

Statistic data and analyzation:

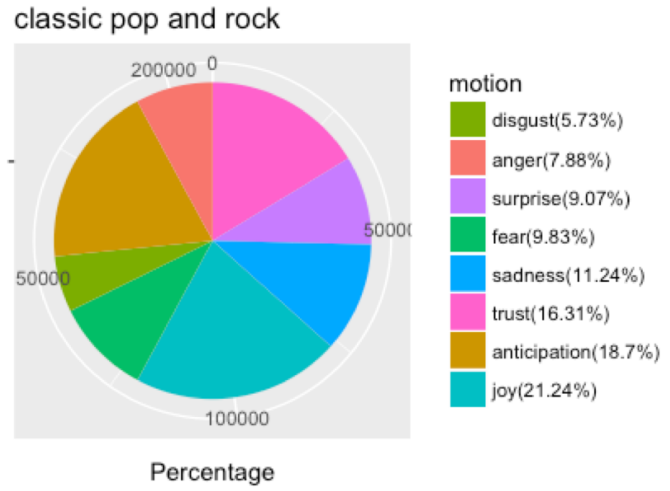
For each genre, we use R and Zeppelin to generate the diagram, including Positive & Negative ratio, emotion ratio and top frequent word diagram.

classic pop and rock

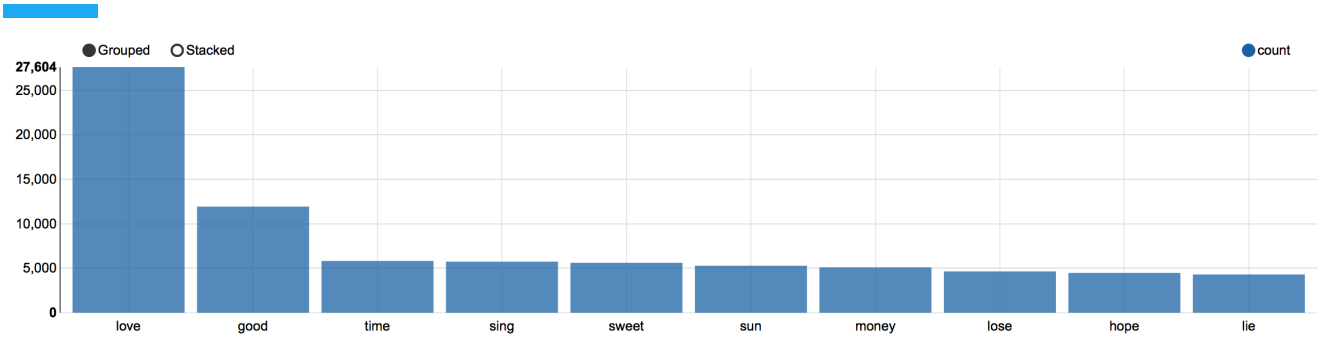
Positive & Negative:



Emotion ratio:



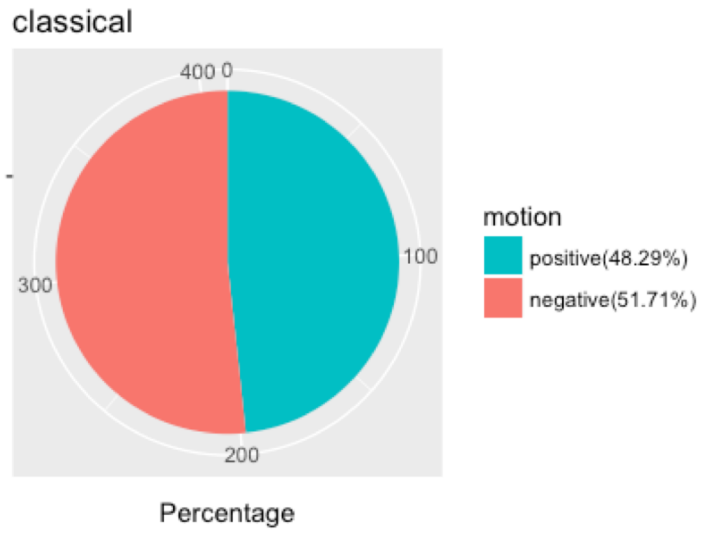
Top 10 Frequent word:



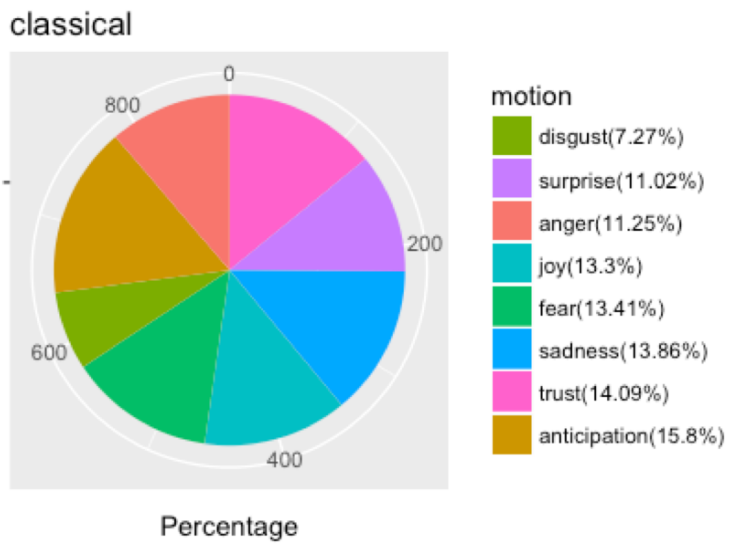
Analysis: We can see in the pie chart, classic pop and rock is a very positive music, and the main emotions are joy, anticipation and trust. So the top 10 words for this genre music (love, good, time, sing, sweet, sun, money, lose hope, lie) are basically represent its emotion. And the hot topics for classic pop and rock are time, money hope and lie, which may be the cause of this emotion ratio.

classical:

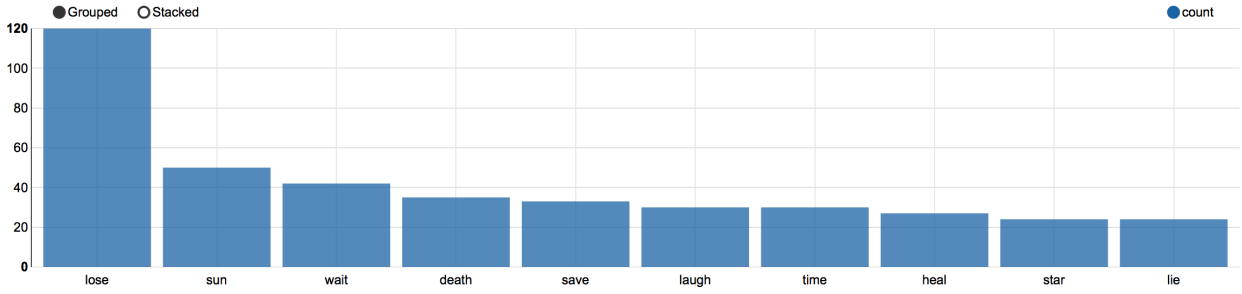
Positive & Negative:



Emotion ratio:



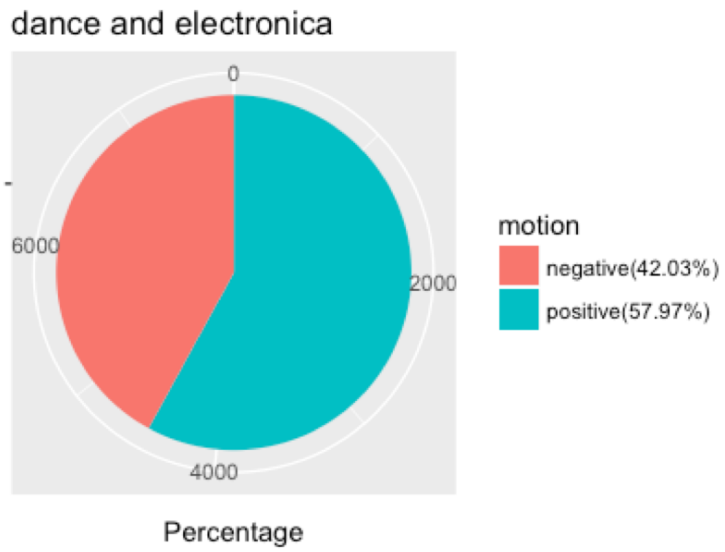
Top 10 Frequent word:



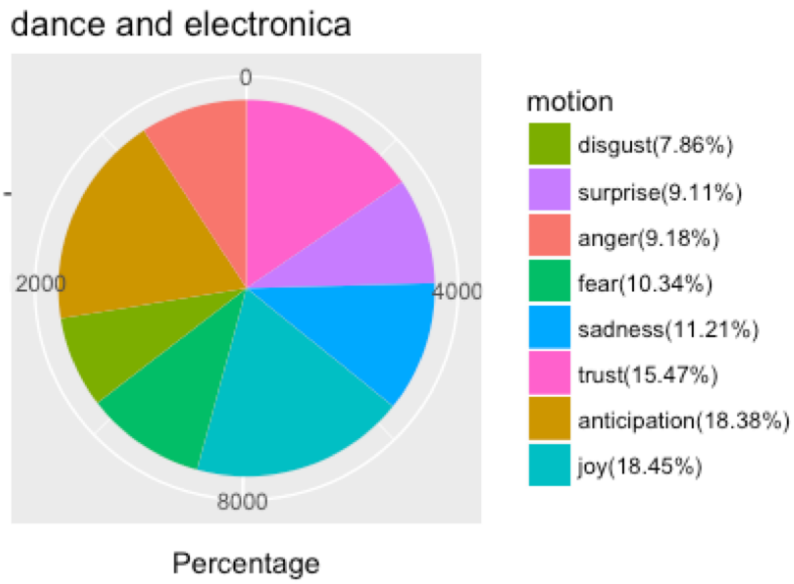
Analysis: We can see in the pie chart, classical is a negative music, but the negative and positive ratio are almost same. The main emotions are anticipation, trust and sadness. So the top 10 words for this genre music (lose, sun, wait, death, save, laugh, time, head, star, lie) are basically represent its emotion. Those negative words like lose, wait, death, lie could be the main source of sadness and fear come from.

dance and electronica:

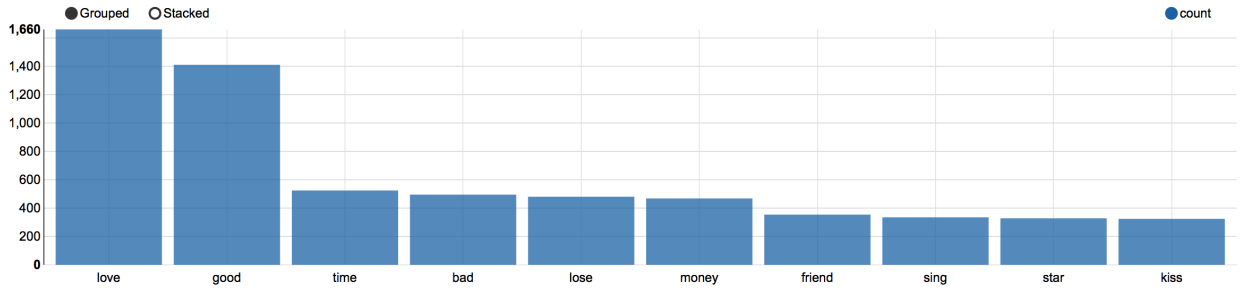
Positive & Negative:



Emotion ratio:



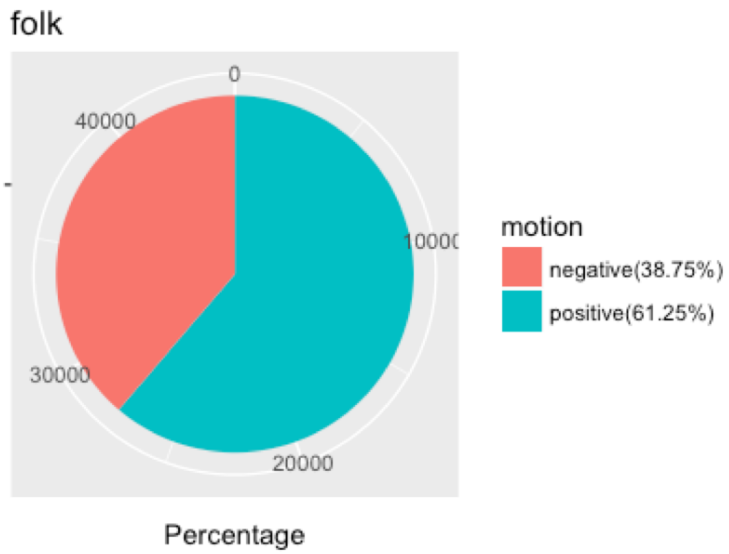
Top 10 Frequent word:



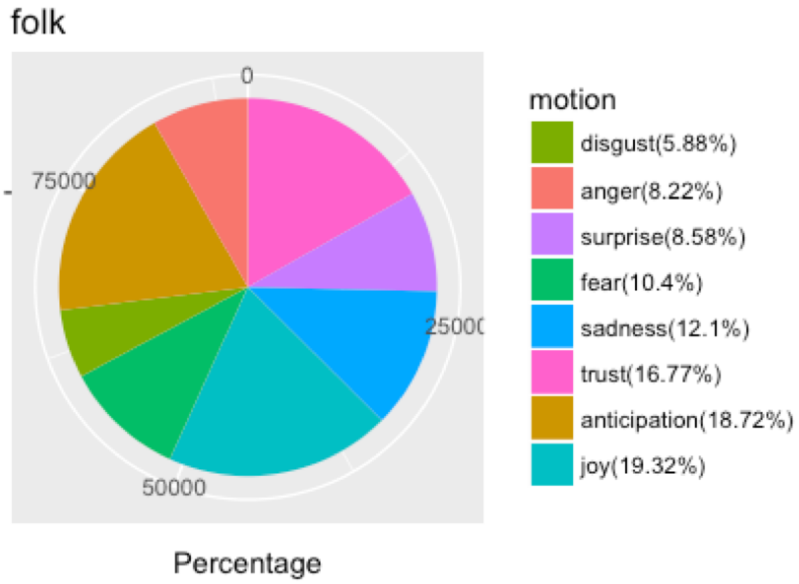
Analysis: We can see in the pie chart, dance and electronica is a positive music. The main emotions are joy, anticipation and trust. So the top 10 words for this genre music (love, good, time, bad, lose, money, friend, sing, star and kiss) are basically represent its emotion. It's not hard to see this is an energetic music, with some good feelings like love. Also, some warm words are inside the top 10 words: friend, sing, star and kiss, which represents the active and positive feelings this kind of songs want to express.

folk:

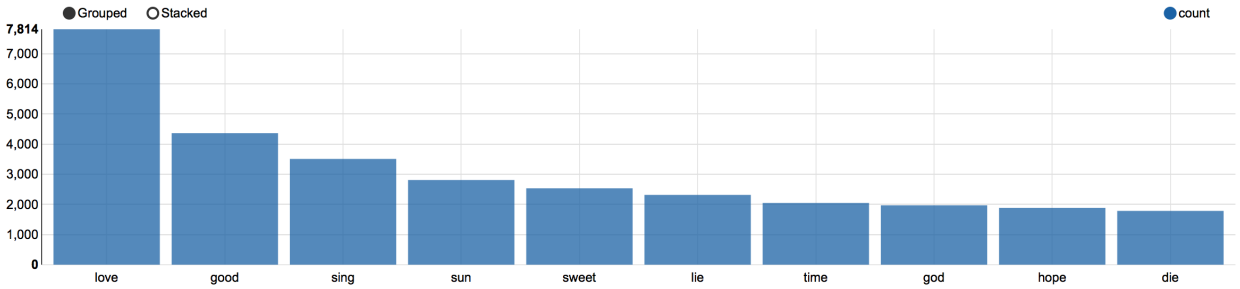
Positive & Negative:



Emotion ratio:



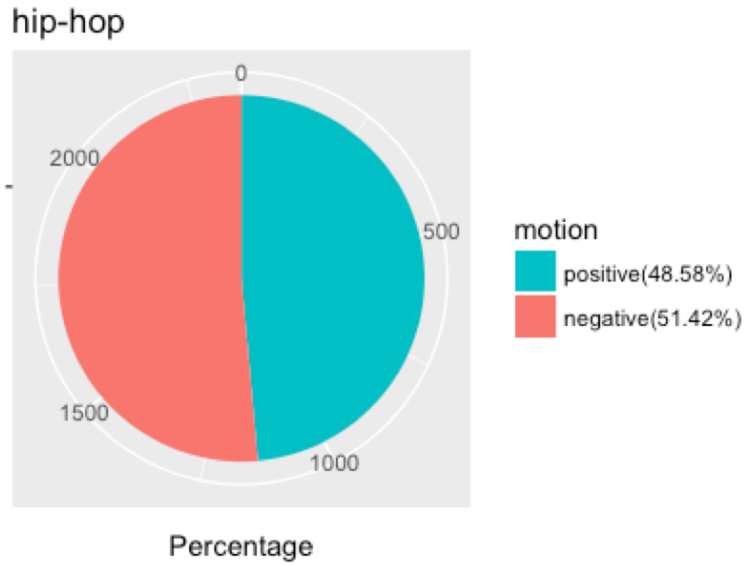
Top 10 Frequent word:



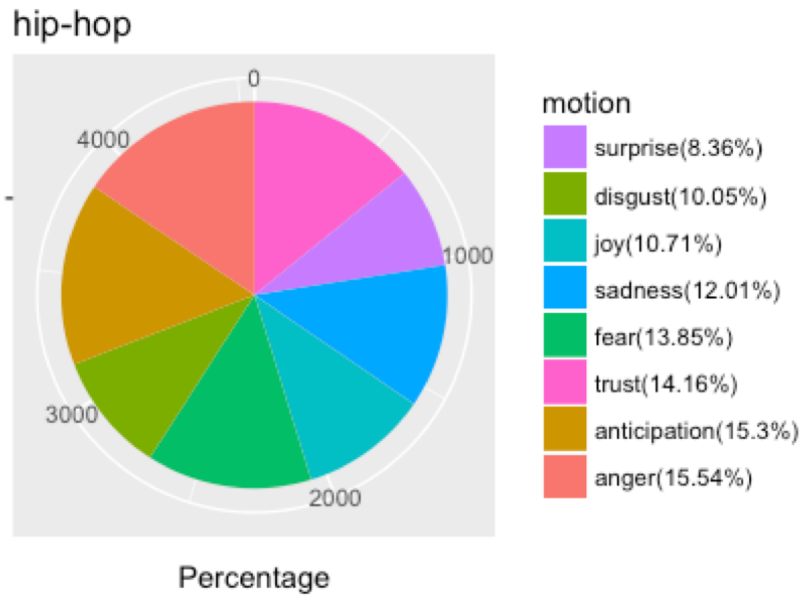
Analysis: We can see in the pie chart, folk is a very positive music. The main emotions are joy, anticipation and trust. So the top 10 words for this genre music (love, good, sing, sun, sweet, lie, time, god, hope, die) are basically represent its emotion. It seems like most of folk music are talking about some sweet love stories, though there are some lies in the stories, but the time and good sunshine will finally heal the past.

hip-hop:

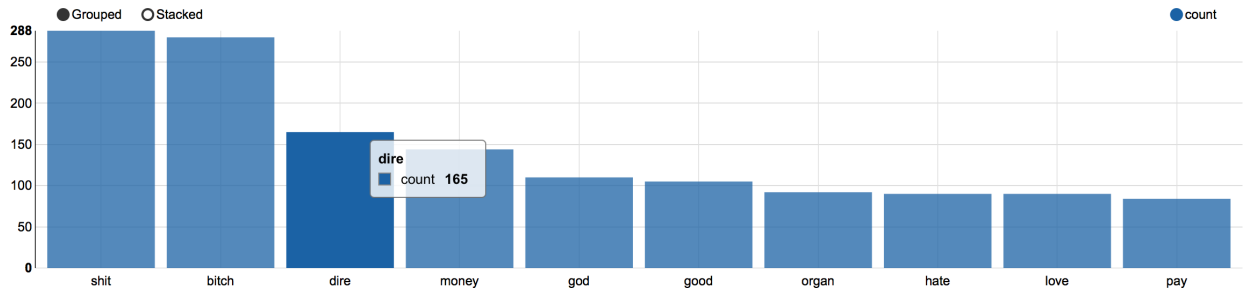
Positive & Negative:



Emotion ratio:



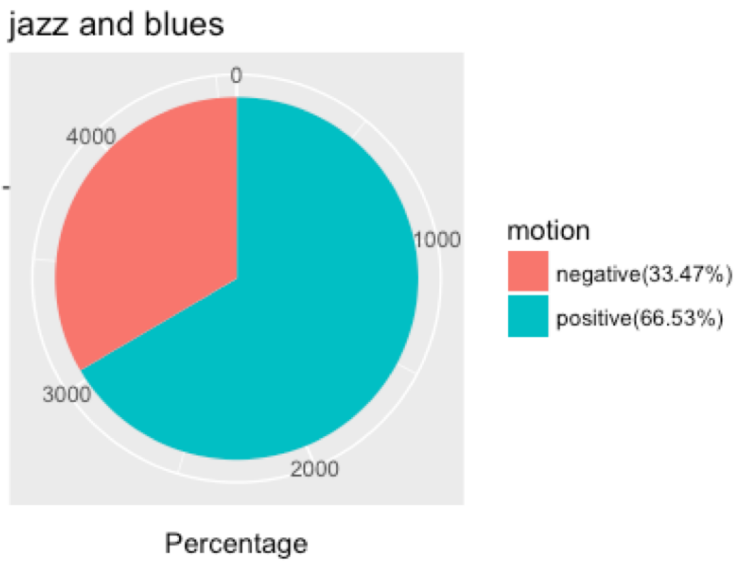
Top 10 Frequent word:



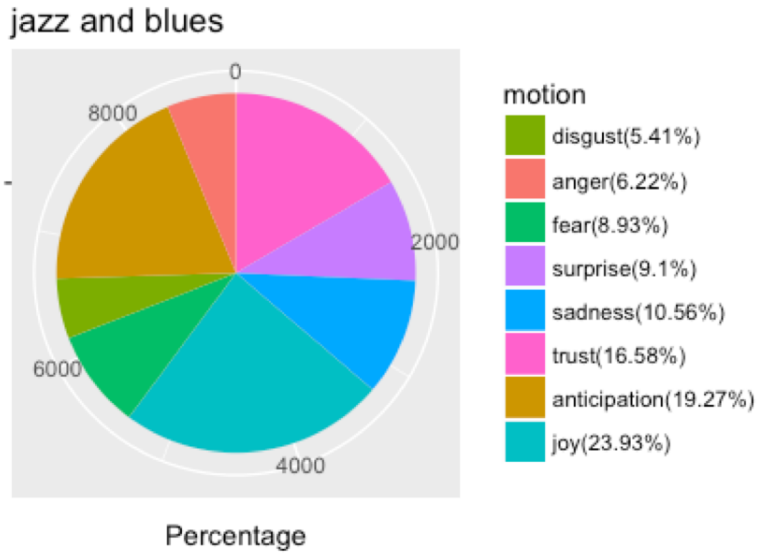
Analysis: We can see in the pie chart, hip-hop is a slightly negative music. The main emotions are anger, anticipation and trust. So the top 10 words for this genre music (shit, bitch, dire, money, god, good, organ, hate, love and pay) express a more negative feeling than what it showed in the first chart. Among this top 10 words, there are several words express anger feeling (shit, bitch, hate and pay). It seems like hip-hop music talks a lot about thug life.

jazz and blues:

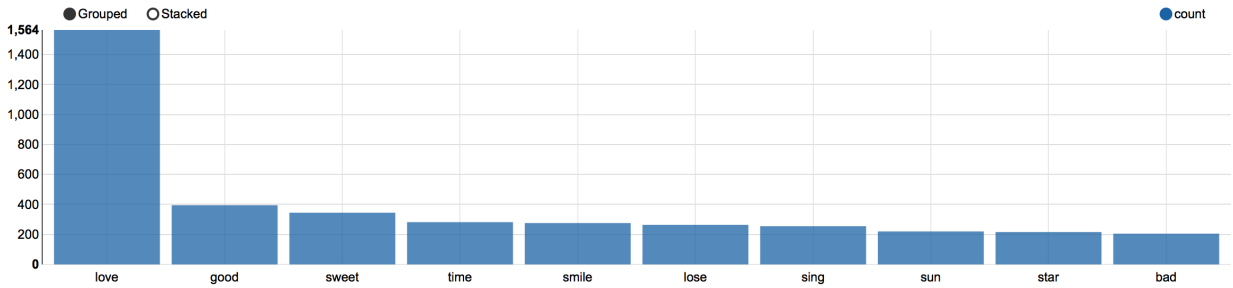
Positive & Negative:



Emotion ratio:



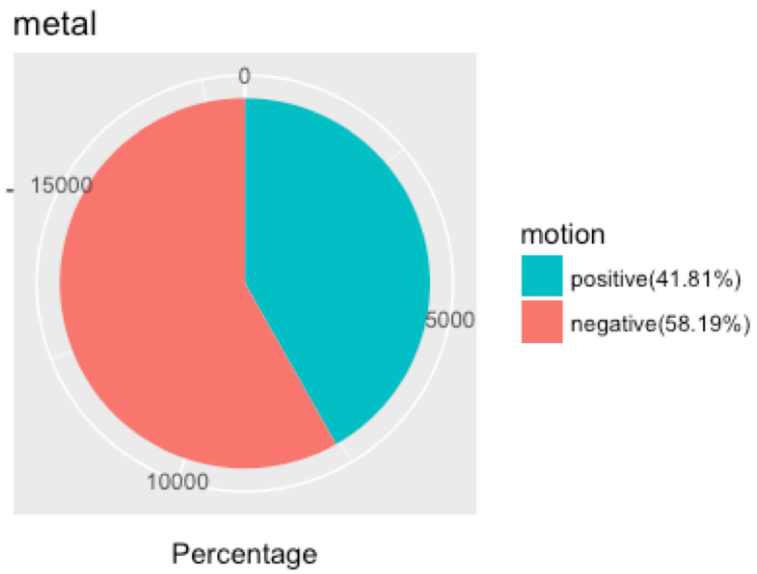
Top 10 Frequent word:



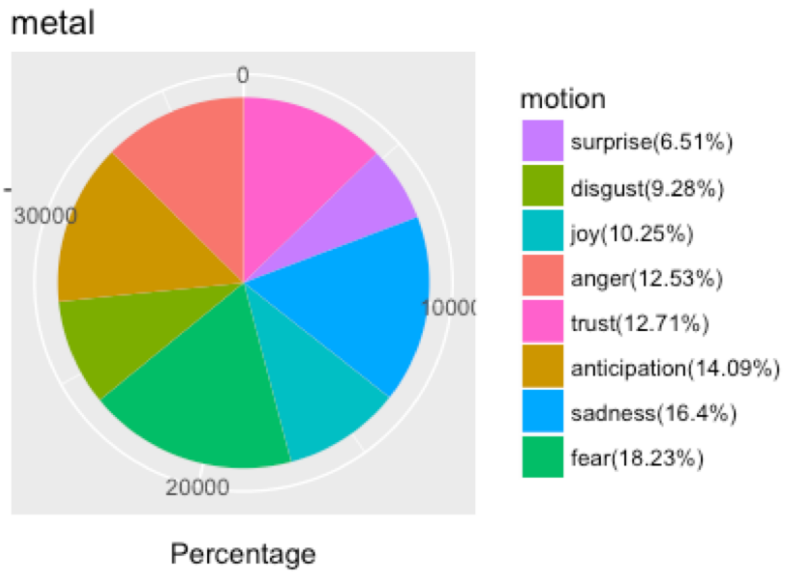
Analysis: We can see in the pie chart, jazz and blues is a happy music. The main emotions are joy, anticipation and trust. So the top 10 words for this genre music (love, good, sweet, time, simile, lose, sing, sun, star and bad) are basically represent its emotion. We can see that most topics are about love, brings a good and sweet feeling.

metal:

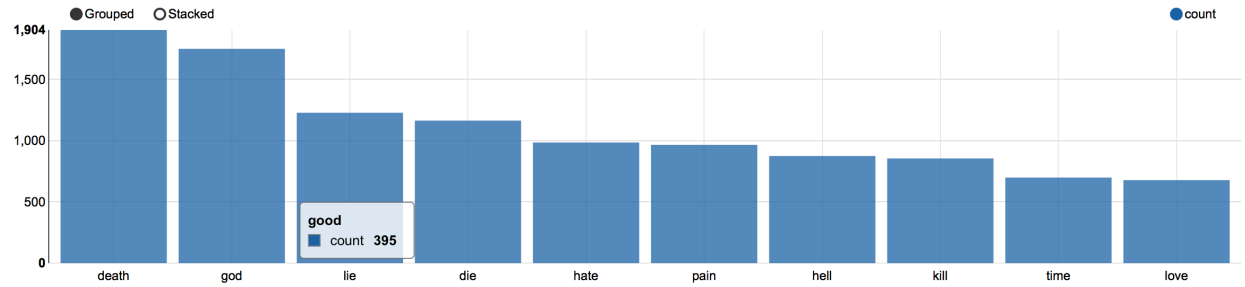
Positive & Negative:



Emotion ratio:



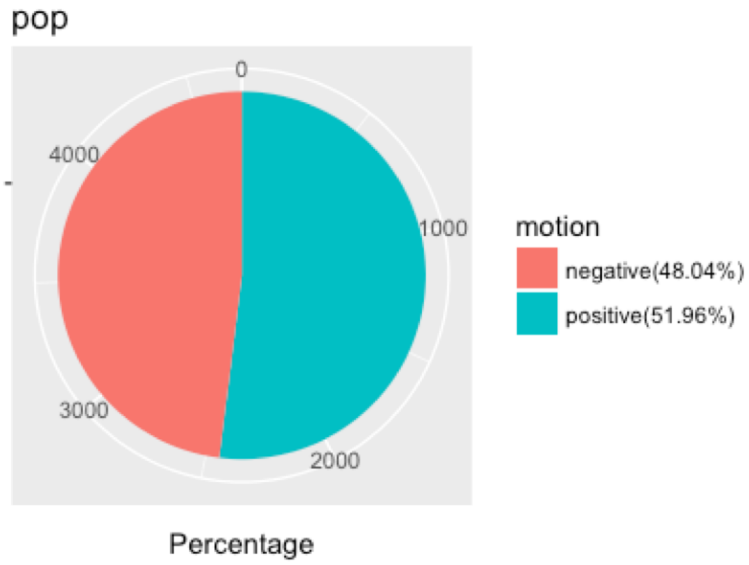
Top 10 Frequent word:



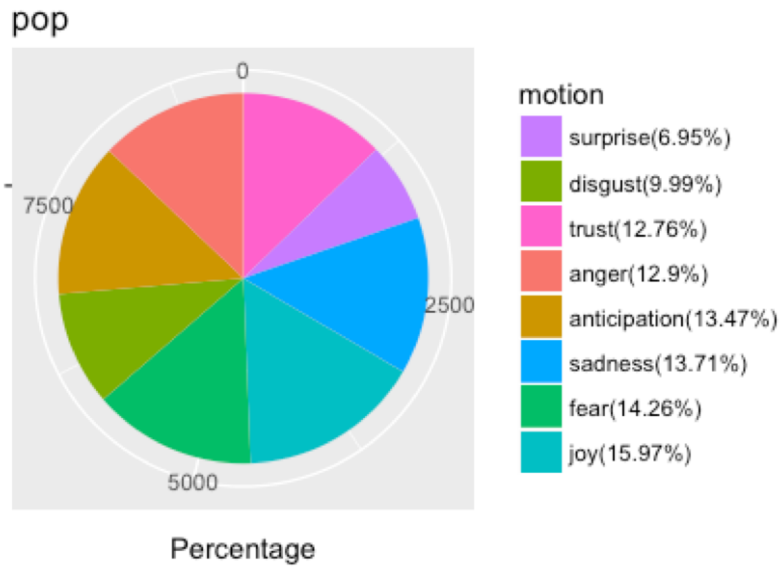
Analysis: We can see in the pie chart, metal is the most negative music among these genre. The main ear, sadness and anticipation. And the top 10 words for this genre music (death, god, lie, die, hate, pain, hell, kill, time and love) are basically represent its emotion. People do feel sad and fear about death and pain, but they also count on what they trust when feeling powerless. In the top 10 words for metal, we can see that metal music talks a lot about topics like faith and believe, and also love and death.

pop:

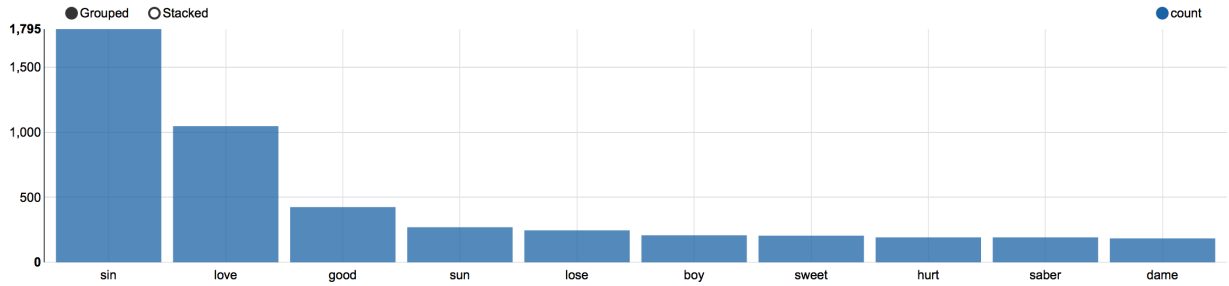
Positive & Negative:



Emotion ratio:



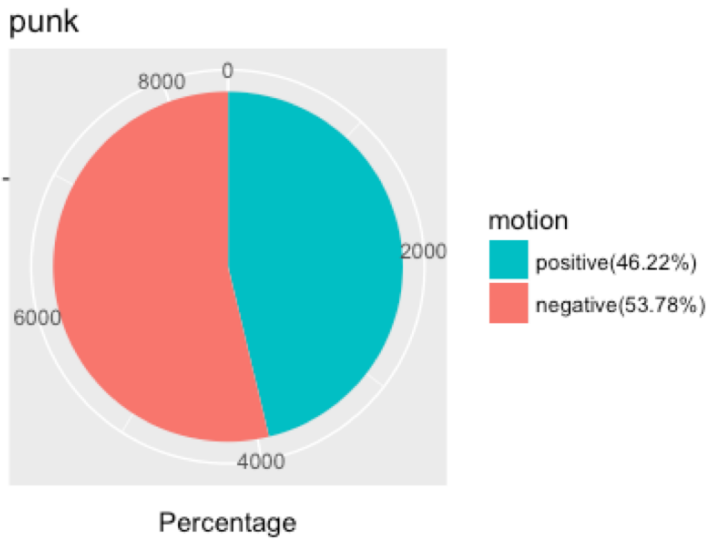
Top 10 Frequent word:



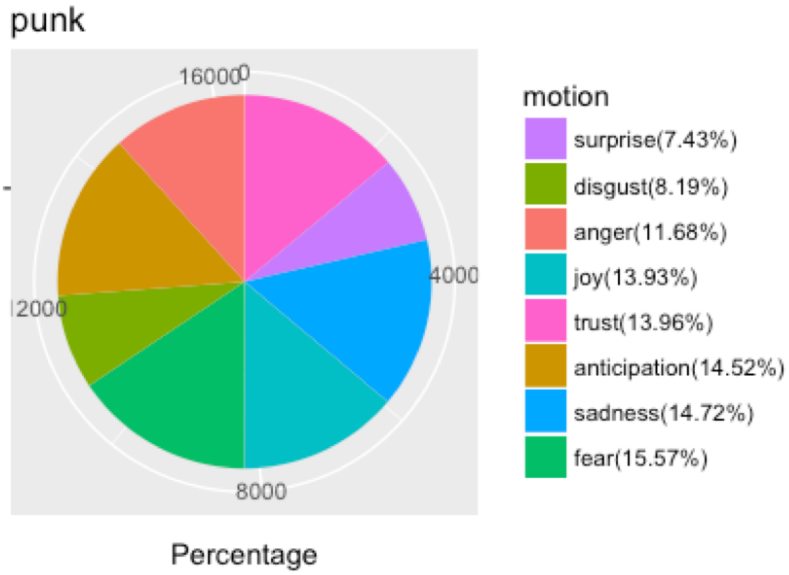
Analysis: We can see in the pie chart, pop is a slightly positive music. The main emotions are joy, fear and sadness. It is weird that 2 of the top 3 emotions are negative word. And the top 10 words for this genre music are sin, love, good, sun, lose, boy, sweet, hurt, saber and dame. So sin might be the most source of the fear emotion.

Punk:

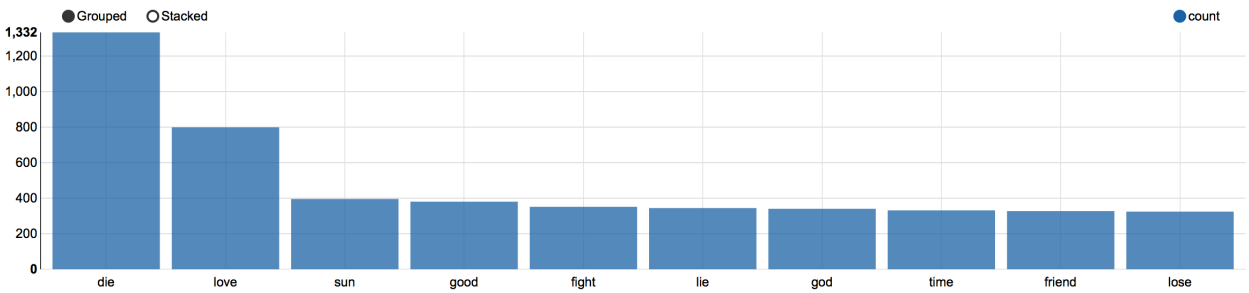
Positive & Negative:



Emotion ratio:



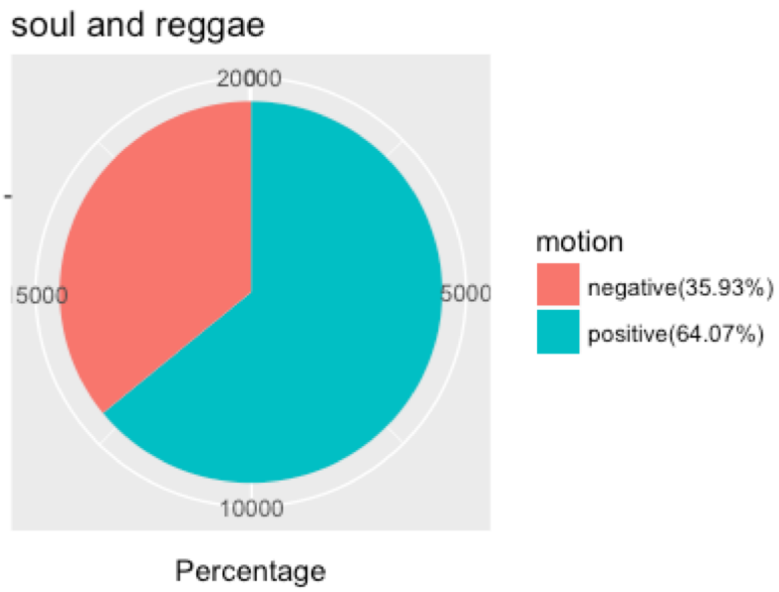
Top 10 Frequent word:



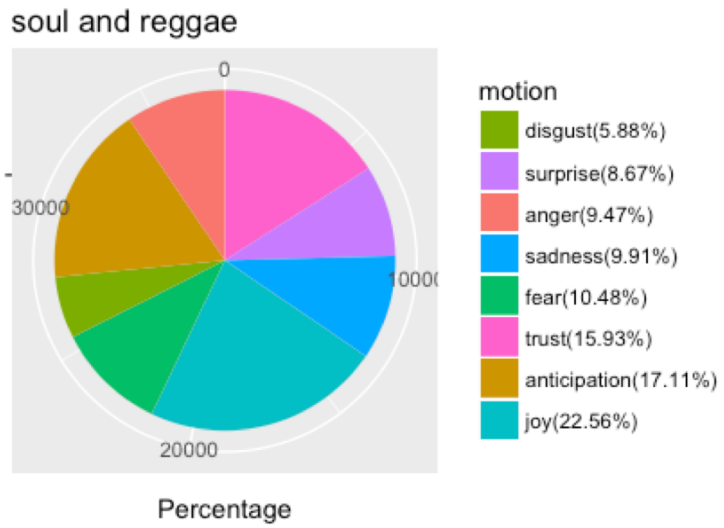
Analysis: We can see in the pie chart, punk is a negative music. The main emotions are fear, sadness and anticipation. So the top 10 words for this genre music (die, love, sun, good, fight, lie, god, time, friend and lose) are basically represent its emotion. It's obvious that punk is an angry music, several words in the top 10 word list express a feeling of impatience and cynical.

soul and reggae:

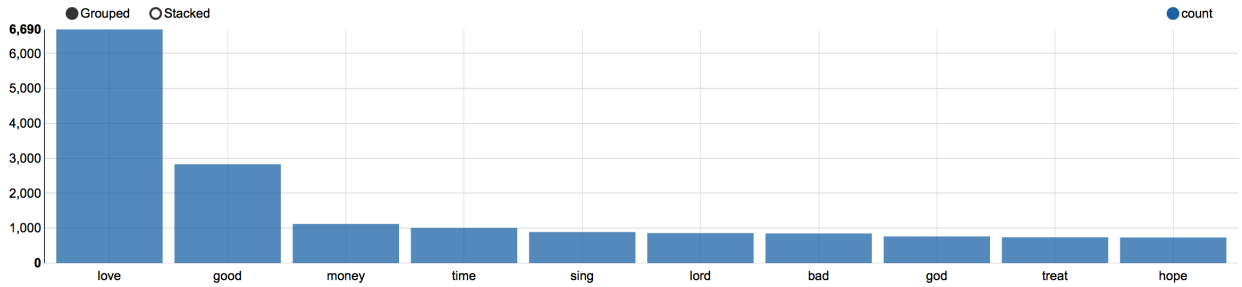
Positive & Negative:



Emotion ratio:



Top 10 Frequent word:



Analysis: We can see in the pie chart, soul and reggae is a very positive music genre. The main emotions are joy, anticipation and trust. And the top 10 words for this genre music (love, good, money, time, sing, lord, bad, god, treat and hope) are basically represent its emotion. Soul and reggae music talks a lot about delightful topic like love, money, time, lord and hope, shows a feeling of peace.

Geography and music

Brief introduction:

We believe that to some extent, music has relationship with its origin place. This is why we started to analyze the connection between music and geography. This part has two focuses: 1. The location mentioned in lyrics and reviews; 2. The location of different kind of artists.

Preparing work:

1. Collect the databases:

   Amazon Music Reviews: With reviews of the music, we can analyse the words that comes from it.

   MusiXmatch dataset: It has a lyric words appearance ranking Top 5000 subset.

   Artists Location Dataset: It has coordinate of the artists.

   Location Names: States names are used.

   Pop Star names: Got from Wikipedia
2. Set up dumbo or other environment to run Pig. We used pig via cloudera.
3. Set up environment for R using Rstudio. Load map library and load the U.S. map.

```
library(ggmap)

map<-get_map(location='united states', zoom=4, maptype = "terrain",
source='google',color='color')
```

Implementation:

Programming with pig:

For this part, we did some basic analysis to the existed databases using pig via cloudera.

Locations in lyrics

With the lyric word ranking database, we are able to find the state name related word count result.

```
inputs1= load './mxm_dataset_word.txt' USING PigStorage(',') as (ranking: int, word:
chararray);
inputs2 = load './FullStateName1.txt' USING PigStorage(',') as (word: chararray);
joiner = JOIN inputs1 by word, inputs2 by word;
STORE joiner INTO './Music_of_the_life/1_words_in_lyrics/' USING PigStorage('\t');
```

Locations in reviews

Convert all the reviews into plain texts.

```
mreviews = FOREACH A GENERATE (chararray)meta#'reviewerID' as revID, (chararray)meta#'asin' as
productasin, (chararray)meta#'reviewText' as revText, (chararray)meta#'summary' as summary;
texts = FOREACH mreviews GENERATE revText;
STORE texts INTO './Music_of_the_life/review_wordcount_pre' USING PigStorage('\t');
```

For the counted words, they are saved in 4 files. The following code combined them together.

```
inputs1 = load './Music_of_the_life/review_wordcount_pre/part-m-00000' USING PigStorage(',')
as (line: chararray);
inputs2 = load './Music_of_the_life/review_wordcount_pre/part-m-00001' USING PigStorage(',')
as (line: chararray);
inputs3 = load './Music_of_the_life/review_wordcount_pre/part-m-00002' USING PigStorage(',')
as (line: chararray);
inputs4 = load './Music_of_the_life/review_wordcount_pre/part-m-00003' USING PigStorage(',')
as (line: chararray);
u = UNION inputs1, inputs2, inputs3, inputs4;
words = FOREACH u GENERATE FLATTEN(TOKENIZE(line)) as word;
grouped = GROUP words BY word;
wordcount = FOREACH grouped GENERATE group, COUNT(words);
```

With the counted words, we are able to find the state name related word count result.

```
locations = load './statename' USING PigStorage(',') as (name: chararray);
joiner = JOIN words by word, locations by name;
```

Popstar locations on map

With the artist coordination database and the list of pop stars' names, we are able to get the locations of the popstars.

```
inputs1 = load 'artistcord.txt' USING PigStorage('\t') as (id: chararray, long: float, lat:
float, name: chararray, city: chararray);
inputs2 = load 'popstarnames.txt'  USING PigStorage(',') as (name: chararray);
joiner = JOIN artistcord by name, popstarnames by name;
…
```

Artists of different genres on map

With the artist coordination database and the csv file which has the artist names and his/her genres, we are able to generate the table that has artists names, locations and genres.

```
inputs1 = load 'artistcord.txt' USING PigStorage('\t') as (id: chararray, lon: float, lat:
float, name: chararray, city: chararray);
inputs2 = load 'msd_genre_dataset.csv' USING org.apache.pig.piggybank.storage.CSVExcelStorage()
as (genre: chararray, track_id: chararray, artist_name: chararray, title: chararray);
grouper = GROUP inputs2 by (genre, artist_name);
flater = FOREACH grouper GENERATE FLATTEN (group) as (genre, artist_name), COUNT(inputs2);
joiner = JOIN inputs1 by name, flater by artist_name;
…
```

Generate graph using R:

With the above process, we are able to get the processed data without title. The titles can be added to the database manually. After that, we already have the data of the following data in files:
“statenameCordinate”: state names, mentioned times, location in longitude and latitude.
“stateNameInLyrics.csv”: state names, ratings, locations in longitude and latitude.
“artistCordinate.csv” : artist names, locations in longitude and latitude
“popstarCord.txt”: artist names, locations in longitude and latitude
“artistCordGenre.txt” : artist names, artist genre, locations in longitude and latitude
With those data files, we are able to put information onto maps.

For the state names appeared in the music reviews:

```
ReviewStatesCord <- read.table('statenameCordinate', header = TRUE)

ggmap(map) + geom_point(data=dfcord, aes(x=lat, y=lon, size=4*sqrt(psize)), color="red",
alpha=0.5)+scale_size(range = c(5, 20)) + ggtitle("10 States mentioned most in the music
reviews")
```

For the state names appeared in the lyric word ranking list:

```
statescord <- read.table('stateNameInLyrics.csv', header = TRUE, fill=TRUE, sep=',')

ggmap(map) + geom_point(data=statescord, aes(x=lat, y=lon,  size=sqrt(5000 - no)),
color="orange", alpha=0.5) + ggtitle("Mentioned States Top 5") +scale_size(range = c(10, 30))
```

For artists:

```
artistsCord <- read.table('artistCordinate.csv', header = TRUE, fill=TRUE, sep=',')

ggmap(map) + geom_point(data=artistscord, aes(x=latitude, y=longtitude), size=1, color="blue",
alpha=0.5) + ggtitle("Artists Locations")
```

For popstars:

```
popstarCord <- read.table('popstarCord.txt', header = TRUE, fill=TRUE, sep='\t')

ggmap(map) + geom_point(data=artistscord, aes(x=latitude, y=longtitude), size=1, color="blue",
alpha=0.5) + geom_point(data=popstarCord, aes(x=lat, y=long), size=2, color="green", alpha=0.8)
```

For different genre artists:

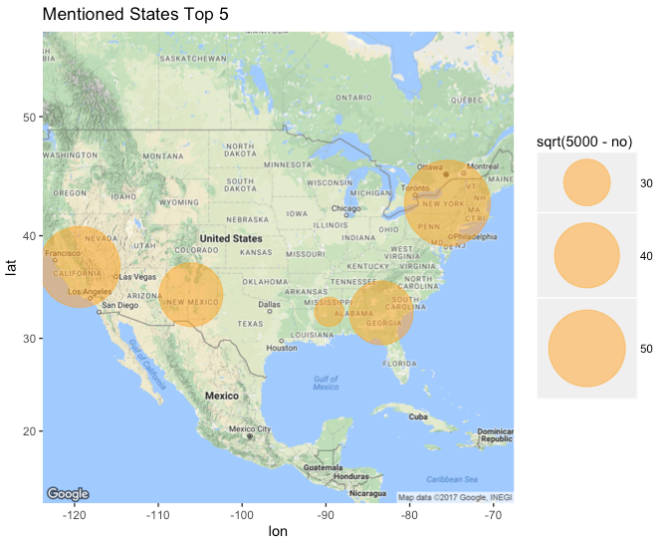
```
genreCord <- read.table('artistCordGenre.txt', header = TRUE, fill=TRUE, sep='\t')

ggmap(map) + geom_point(data=genreCord, aes(x=lat, y=lon, color=genre), size=15, alpha=0.8) +
ggtitle("Location of artists in different genre")
```

To check the corresponding map info within New York City:

```
map<-get_map(location='new york', zoom=9, maptype = "terrain", source='google',color='color')
```

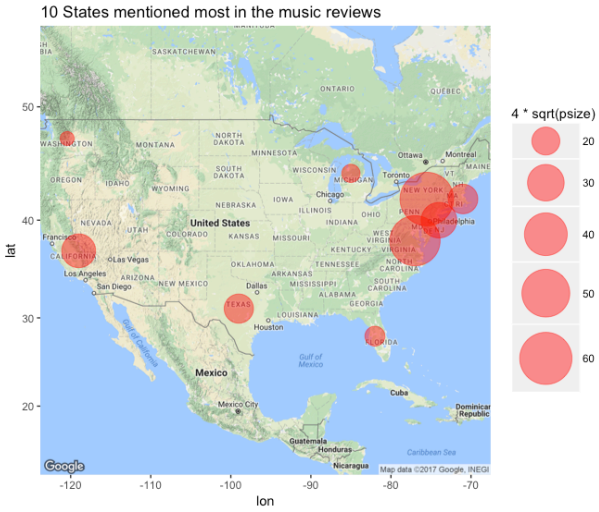
Statistic data and analyzation:

5 top states mentioned in the lyrics:



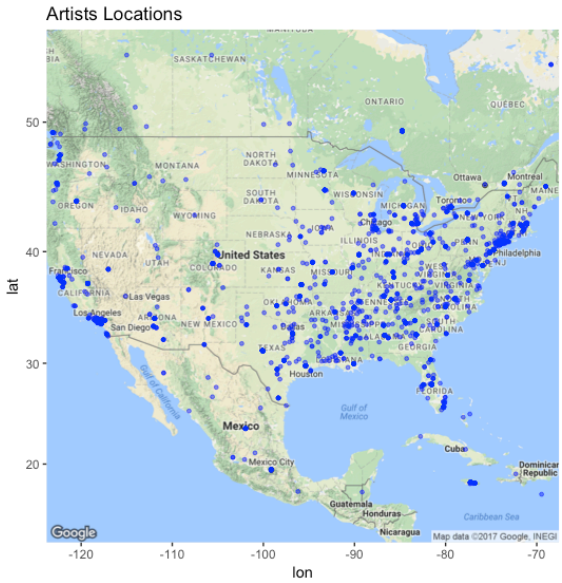
From the result graph we can see that: Not only big cities are mentioned in the lyrics! Mississippi and Georgia are also ranked in the top.

10 top states mentioned in the reviews:



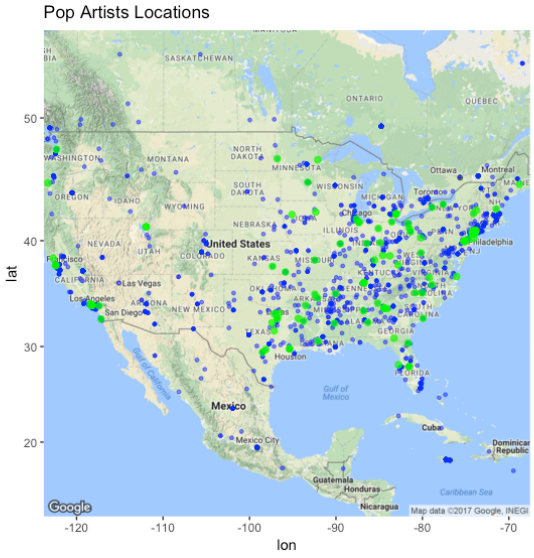
From the result graph we can see that: Shows that the content of music is more related with these places.

Artists locations:



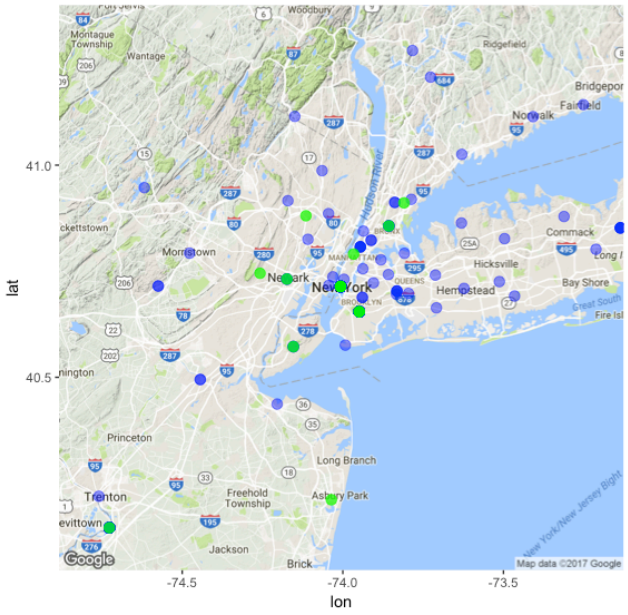
From the result graph we can see that: The artists location is proportional to the population of the states.

Popstar locations on top of all artists: (Blue: all artists. Green: popstars)

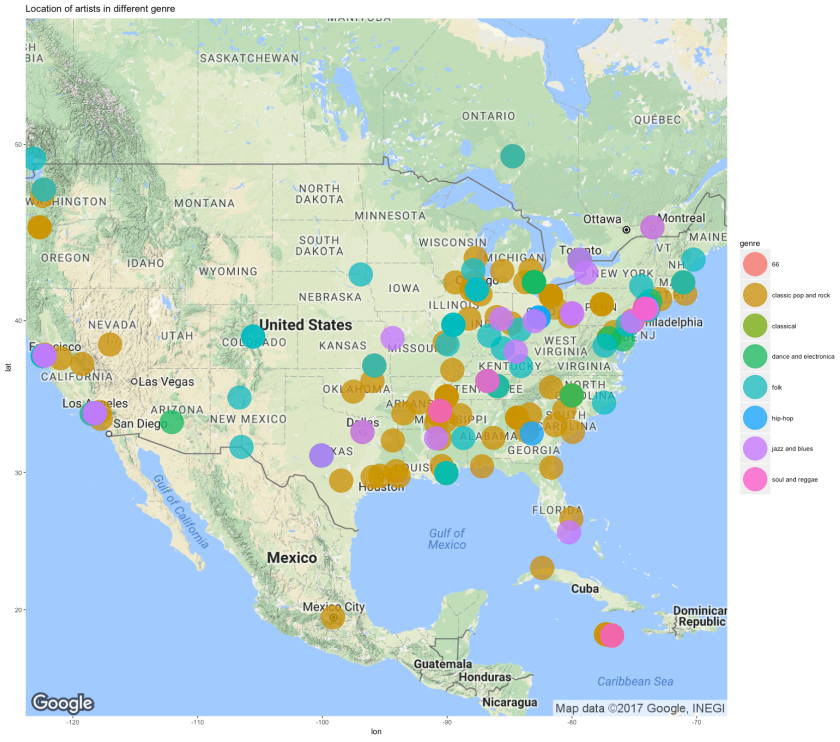


From the result graph we can see that: The pop stars are more close to the sea.

Popstars in New York:



Artists of different genres:



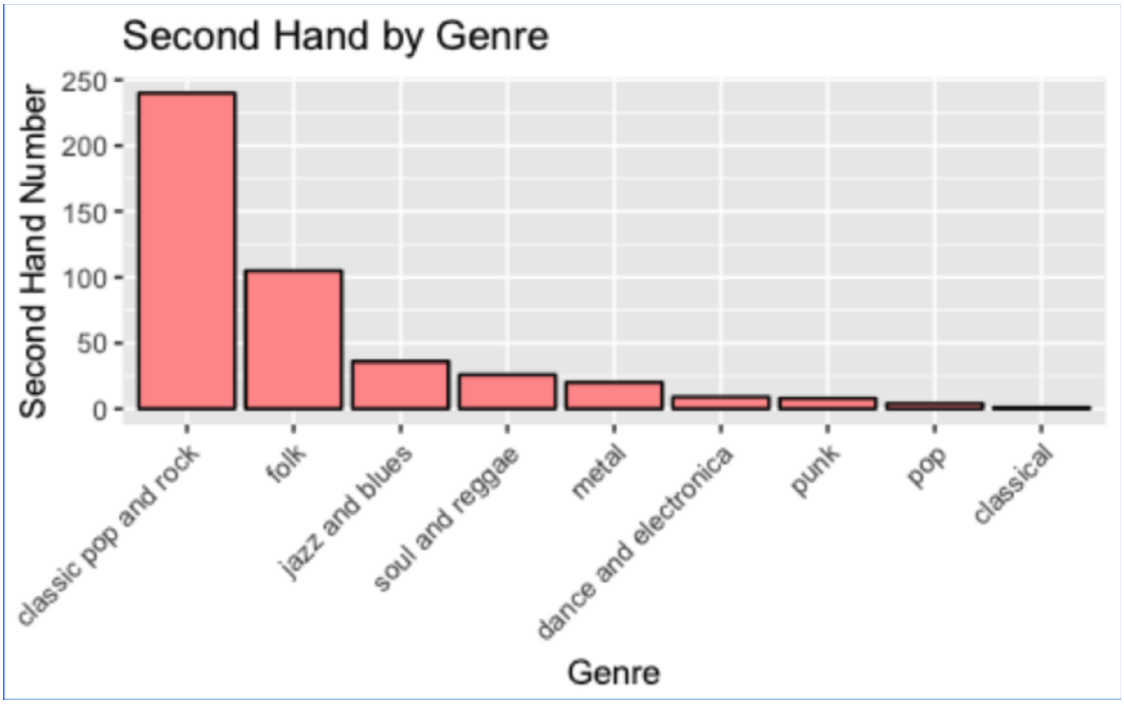
From the result graph we can see that: Some genre is more close to the sea, and some are more likely to be located in the inner part of the country.

Some interesting facts about music

1. About covered songs:

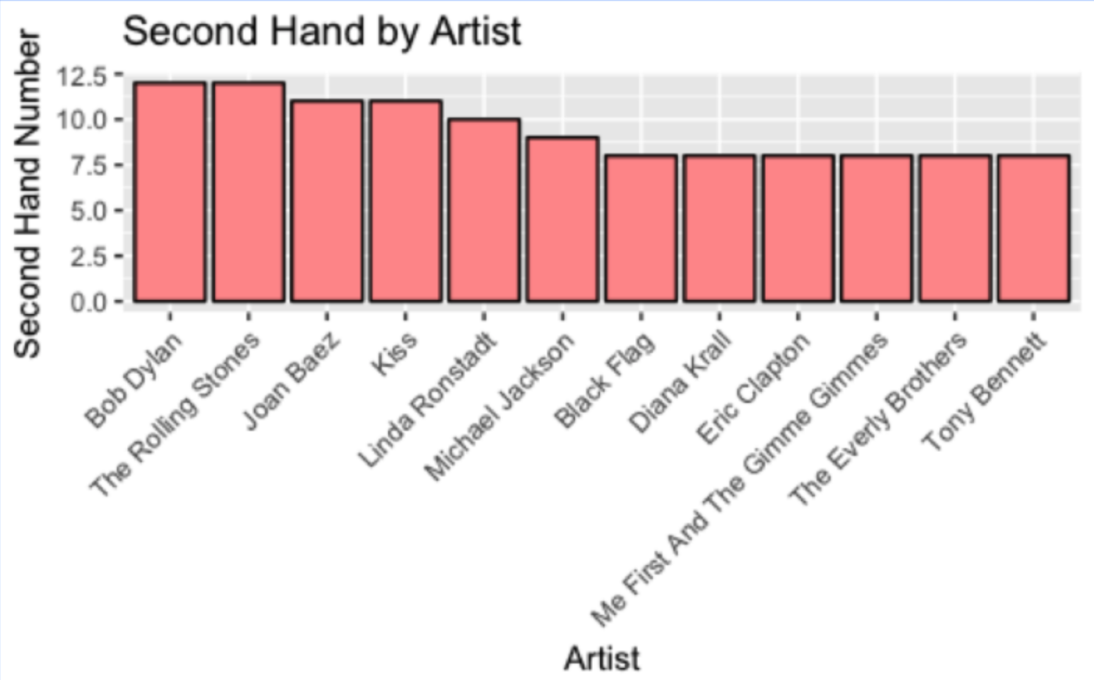
(1) Covered songs rank by genre：

The diagram below is the statistical result of the second hand songs dataset. In this diagram, we can see that the music genre which is most likely to be covered is classic pop and rock, which means that those songs under this classification has more music can be relatively easy to be covered, which make this genre of music more popular in karaoke or corresponding music applications.



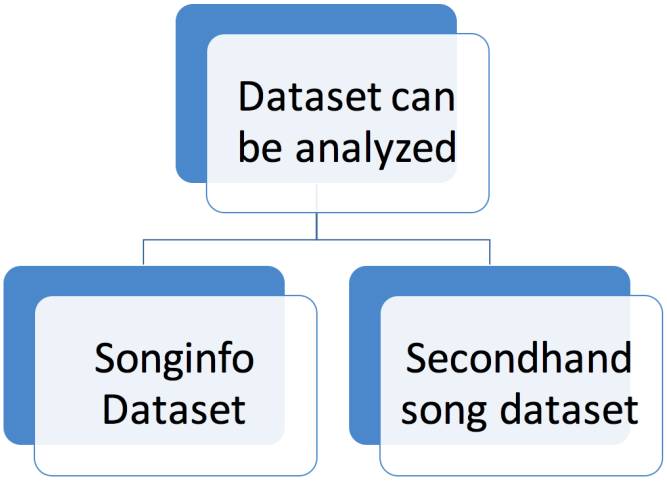
(2) covered songs rank by artists：

Same as the previous part for covered songs, the artists listed below are the top 10 artists whose songs are most likely to be covered by other artists. Which means karaoke or corresponding application may import more songs from these artists into their music database, which may bring them more users.



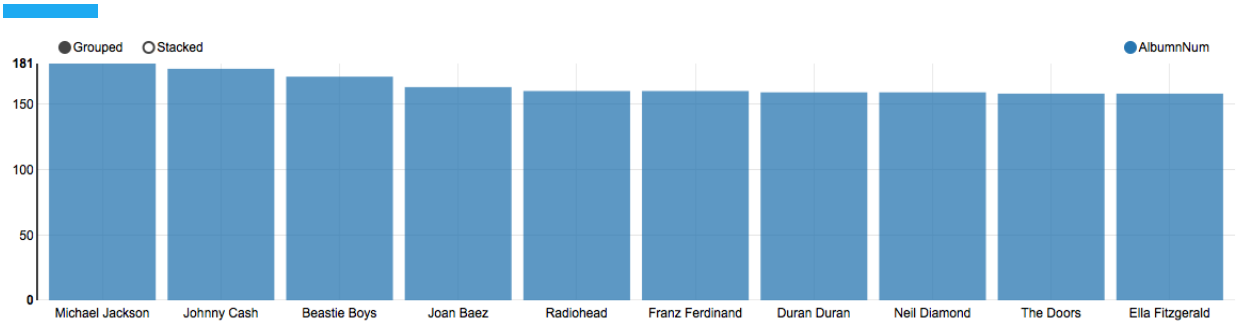
(3) Dataset we used：

We use the Songinfo dataset introduced in the previous chapter, join with the Secondhand song dataset using song id, generate the dataset which can be used to analyzed in this part.



2. About album publish:

The diagram below is generated using Apache Zeppelin, lists the top 10 artists by the number of albums they published.

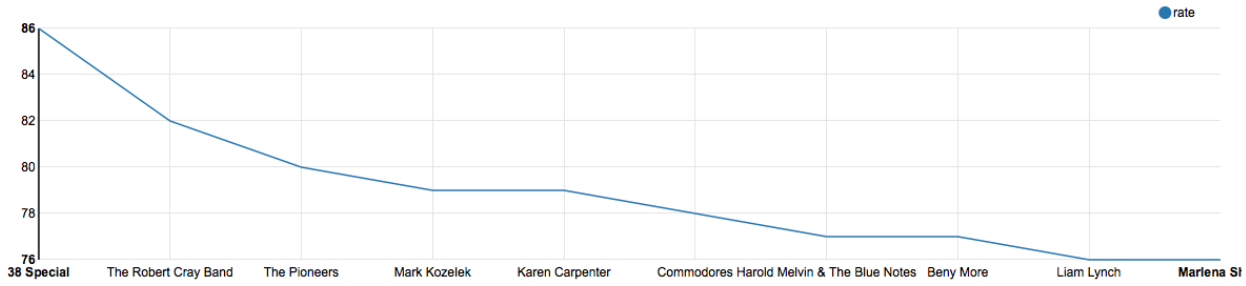


3. Artist's' rate from users:

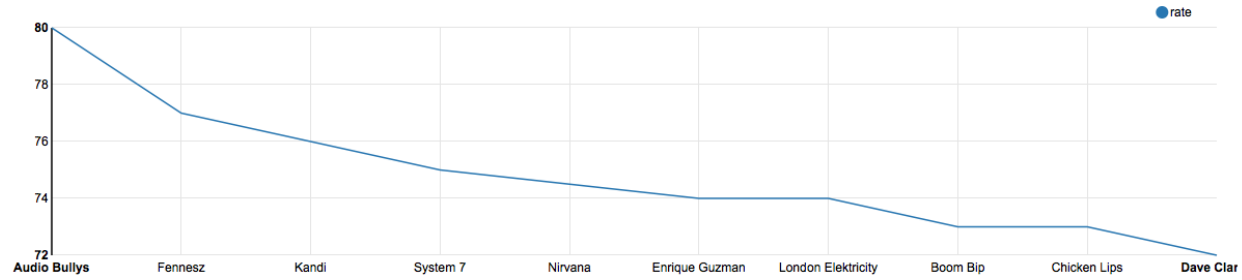
In this part, we use Zeppelin to do some query and find out the user rate for each genre of music. In the diagram below, we picked out the top 10 artists for each genre who has the highest rate. Using this statistical result and output dataset, we can do some recommendation for those users who love specific genre of music.

In this part of project, we use echo nest user taste dataset join with the songinfo dataset to generate the final dataset which can be used to analysed.

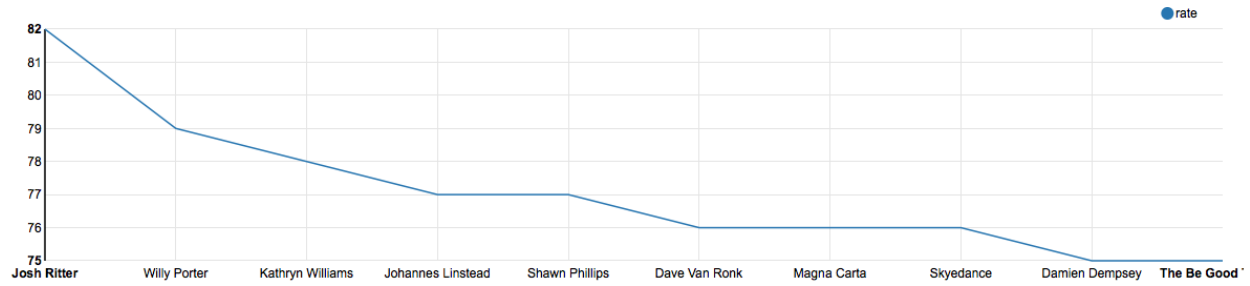
(1) classic pop and rock



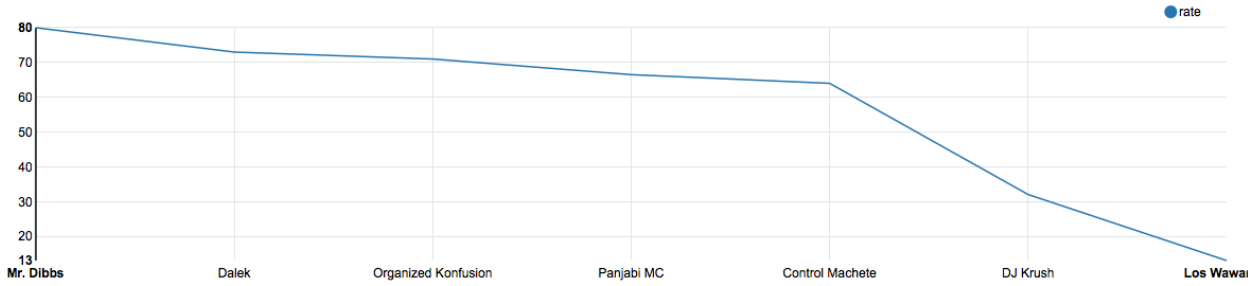
(2) dance and electronica



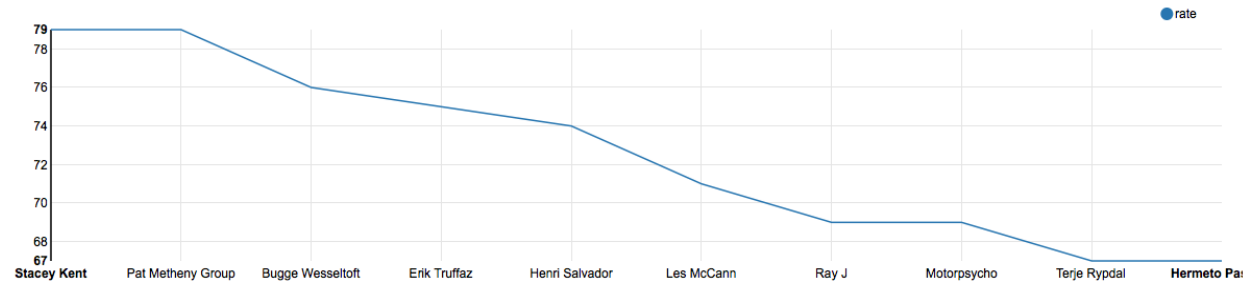
(3) Folk



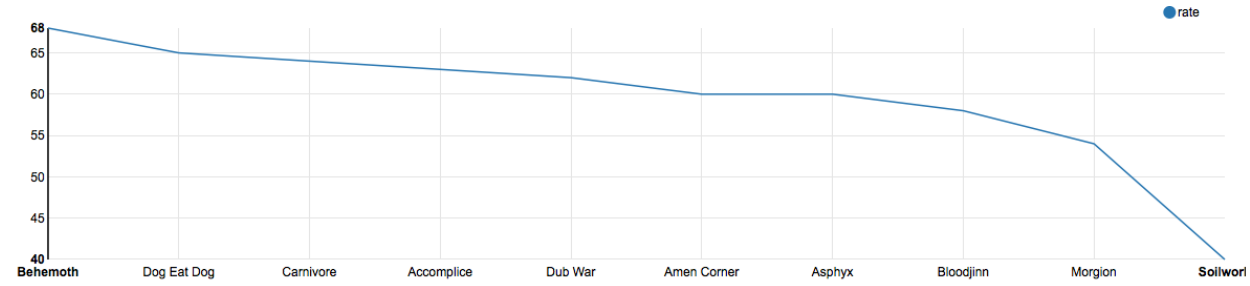
(4) Hiphop



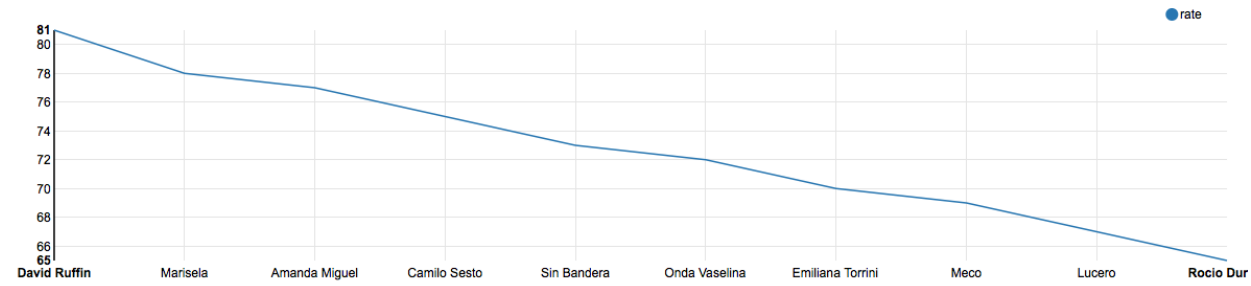
(5) jazz and blues



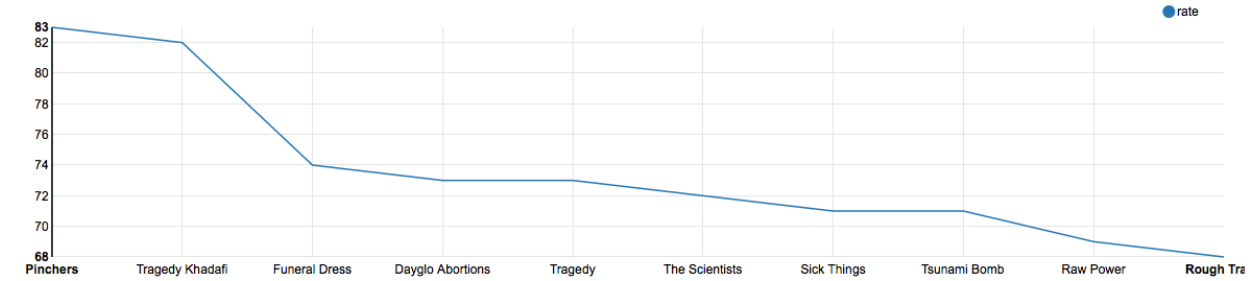
(6) Metal



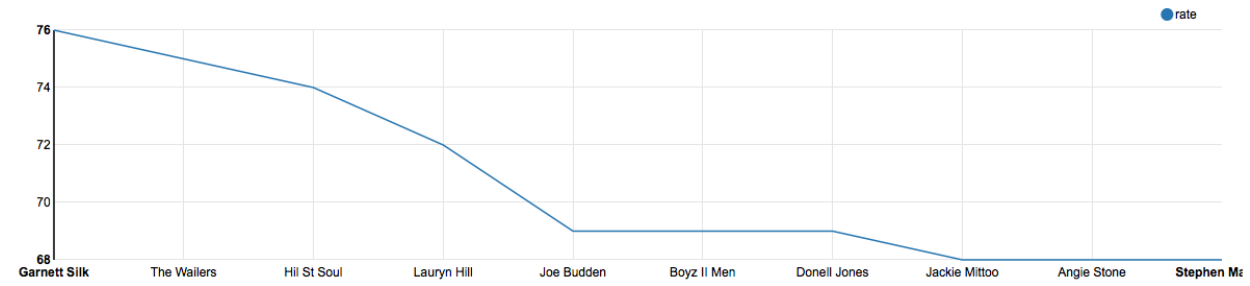
(7) Pop



(8)Punk

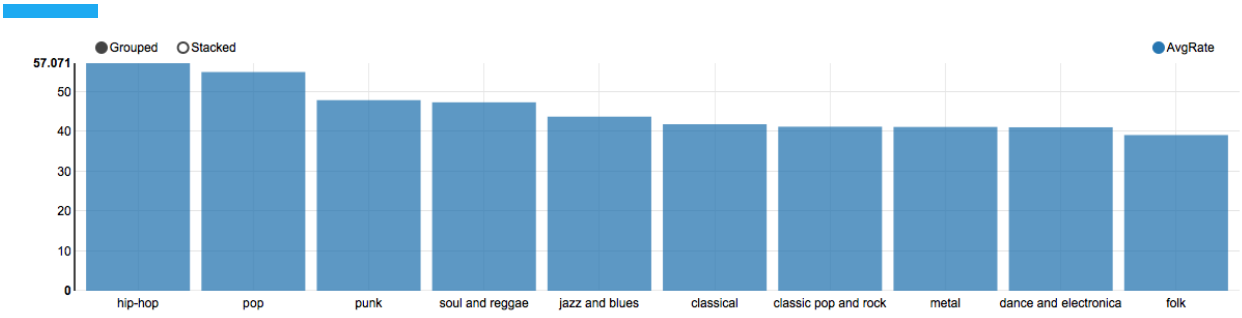


(9) soul and reggae



4. Average rate for every genre:

The diagram below is the statistical result of each genre. Using the same dataset like the previous part, we calculate the average rate for each genre, found out that the people tend to give hip-hop music higher score, then pop, and next punk. Among all music genres, folk has the lowest average score, probably the folk music itself is too undramatic.



Problems we met and solutions

1. Problem: The musiXmatch dataset has a word list which contains 5000 words in a single line, making joining them with other database really hard.

Solution: Since 5000 word is not a big file, i wrote a python script to flatten these 5000 words with a format [index, word]. Using the flatten version of this dataset, we can easily join it with another dataset using the index of each word.

2. Problem: The total size of MSD is 500 G and it is in .h5 format which is not possible for us the use.

Solution: We used python to extract the data we need. As there is MSD dataset available on HPC, we uploaded the python code and ran it on HPC. it took 12 hours to extract the total dataset to a 160 MB csv file.

3. Problem: When dealing with the map database in R, there is always an extra line, which results in reading data frame failure.

Solution: We tried adding parameter to ignore empty lines, and accept only certain lines, which all failed. Finally we solved it by using excel to delete the empty lines to the right and save it again.

Summary

In this project, we did music trend analysis, emotion analysis, lyric word analysis, location analysis, user rate analysis, secondhand songs analysis and album number analysis. After we chose music as our topic, we spend few days on looking for datasets. Except the famous MSD dataset, we try to find more datasets which might seem like unuseful but can help us to do some interesting analysis. The cross-dataset analysis is the most interesting part in this project. When we find a new dataset, we will think what can

we use this dataset to do, and list all possible analysis we can do. After that, we try our best to analyze as much as possible.

During this project, we have some challenges, some of them stuck our progress for several days, but we luckily found the way to solve them. Also, during the project, we practice those tools, languages and environments we learned in the class, by using them in hands-on work, we are more clear about what big data is, and what big data is capable of.

The reason we choose music as our topic is because we all love music and want to dig more about what we listen everyday. Using the dataset we have, we can still analyse more, for example, find the pattern of users, predict which songs this user might like. Though work like this may use more time on algorithm study, but we won't stop our exploring.

If interested in our project, please visit:
https://github.com/ys2843/NYU_Million_Song_Dataset_Analysis

Some code and output files are uploaded under this repository.