IST 707 – Text Analytics

Project Report on

Song Lyrics Text mining

SUBMITTED BY

Jieqing Hu | jhu125@syr.edu
Sumeet Santani | ssantani@syr.edu
Kshitij Sankesara | kssankes@syr.edu

Index

‘

1. Introduction

The ever growing amount of music available on the internet calls for intelligent tools for browsing and searching music databases. Music recommendation and retrieval systems can aid users in finding music that is relevant to them. This typically requires automatic music analysis, e.g., classification according to genre, content or artist and song similarity.
In this project, there are the lyrics for 57650 songs. We would use text mining approaches such as clustering of the words with similar meanings, predicting artist by the song or emotional analysis for the song.

2. Data Description

The dataset is available on kaggle(https://www.kaggle.com/mousehead/songlyrics). It contains 4 columns and 57650 rows of data. It has 44824 songs, sung by different artists. Lyrics are in separate column "Lyrics", which has to be used for text mining and sentiment analysis. The following is the description for each column.

1. Artist (target variable)

2 Song Name

3. Link to a webpage with the song (for reference). This is to be concatenated with http://www.lyricsfreak.com to form a real URL.

4. Lyrics of the song

Overview of the dataset:

```
data.head()
```

	artist	song	link	text
0	ABBA	Ahe's My Kind Of Girl	/a/abba/ahes+my+kind+of+girl_20598417.html	Look at her face, it's a wonderful face \nAnd...
1	ABBA	Andante, Andante	/a/abba/andante+andante_20002708.html	Take it easy with me, please \nTouch me gentl...
2	ABBA	As Good As New	/a/abba/as+good+as+new_20003033.html	I'll never know why I had to go \nWhy I had t...
3	ABBA	Bang	/a/abba/bang_20598415.html	Making somebody happy is a question of give an...
4	ABBA	Bang-A-Boomerang	/a/abba/bang+a+boomerang_20002668.html	Making somebody happy is a question of give an...

3. Data Preprocessing and Preparation

3.1 Data Cleaning

We removed stop words, converted all words to lowercase and did the stemming. We also removed unnecessary character in the lyrics column.

The following is the corpus and Word Cloud after data cleaning:

```
corpus[:2]
```

```
['look face wonder face mean someth special look way smile see lucki one fellow kind girl make feel fine could ever b
eliev could mine kind girl without blue ever leav could could go walk park hold squeez hand go walk hour talk thing p
lan kind girl make feel fine could ever believ could mine kind girl without blue ever leav could could',
 'take easi pleas touch gentli like summer even breez take time make slow andant andant let feel grow make finger sof
t light let bodi velvet night touch soul know andant andant go slowli music music song song music song play time time
make strong play caus make strong make sing make sound make sing make andant andant tread lightli ground andant andan
t oh pleas let shimmer eye like feel thousand butterfli pleas talk go play andant andant let float away music music s
ong song music song play time time make strong play caus make strong make sing make sound make sing make andant andan
t tread lightli ground andant andant oh pleas let make sing make sound make sing make andant andant tread lightli gro
und andant andant oh pleas let andant andant oh pleas let']
```

3.2 Prepare Vectorization

First, we used bigram CountVectorizer, and set the min_df = 2, max_df = 0.95.

```
tf_vectorizer = CountVectorizer(ngram_range=(2,2),max_df=0.95, min_df= 2, max_features= no_features, stop_words='englis
tf = tf_vectorizer.fit_transform(corpus)
tf_feature_names = tf_vectorizer.get_feature_names()
```

```
print(list(tf_vectorizer.vocabulary_.items())[:10])
```

```
[('look way', 497), ('kind girl', 347), ('girl make', 240), ('make feel', 566), ('feel fine', 204), ('time make', 86
1), ('let feel', 409), ('light let', 438), ('music song', 601), ('time time', 873)]
```

Then, we used trigram CountVectorizer and kept the parameters the same as the bigram one.

```
tf_vectorizer_tri = CountVectorizer(ngram_range=(3,3),max_df=0.95, min_df= 2, max_features= no_features, stop_words='er
tf = tf_vectorizer_tri.fit_transform(corpus)
print(list(tf_vectorizer_tri.vocabulary_.items())[:10])
```

```
[('make feel fine', 503), ('pleas let make', 633), ('thought love end', 838), ('make somebodi happi', 512), ('somebod
i happi question', 764), ('happi question learn', 355), ('question learn come', 651), ('come break everi', 126), ('br
eak everi smile', 79), ('everi smile everi', 209)]
```

We also tried unigram, bigram TfidfVectorizer and trigram TfidfVectorizer with the same parameters.

```
# covert words into TFIDF metrics
tfidf = TfidfVectorizer(max_df=0.95, min_df= 2, max_features= no_features, stop_words='english')
X_text = tfidf.fit_transform(corpus)
print(list(tfidf.vocabulary_.items())[:10])
```

```
[('look', 507), ('face', 281), ('wonder', 978), ('mean', 532), ('someth', 796), ('special', 809), ('way', 951), ('smi
le', 788), ('lucki', 520), ('kind', 457)]
```

```
# covert words into TFIDF metrics
tfidf = TfidfVectorizer(ngram_range=(2,2), max_df=0.95, min_df= 2, max_features= no_features, stop_words='english')
X_text = tfidf.fit_transform(corpus)
print(list(tfidf.vocabulary_.items())[:10])
```

```
[('look way', 497), ('kind girl', 347), ('girl make', 240), ('make feel', 566), ('feel fine', 204), ('time make', 86
1), ('let feel', 409), ('light let', 438), ('music song', 601), ('time time', 873)]
```

```
tfidf_tri = TfidfVectorizer(ngram_range=(3,3), max_df=0.95, min_df= 2, max_features= no_features, stop_words='english')
X_text = tfidf_tri.fit_transform(corpus)
print(list(tfidf_tri.vocabulary_.items())[:10])
```

```
[('make feel fine', 503), ('pleas let make', 633), ('thought love end', 838), ('make somebodi happi', 512), ('somebod
i happi question', 764), ('happi question learn', 355), ('question learn come', 651), ('come break everi', 126), ('br
eak everi smile', 79), ('everi smile everi', 209)]
```

4. Objective and Business Questions

1.) What are the songs that are frequently played together?

2.) What are the most common sentiments of a particular artist?

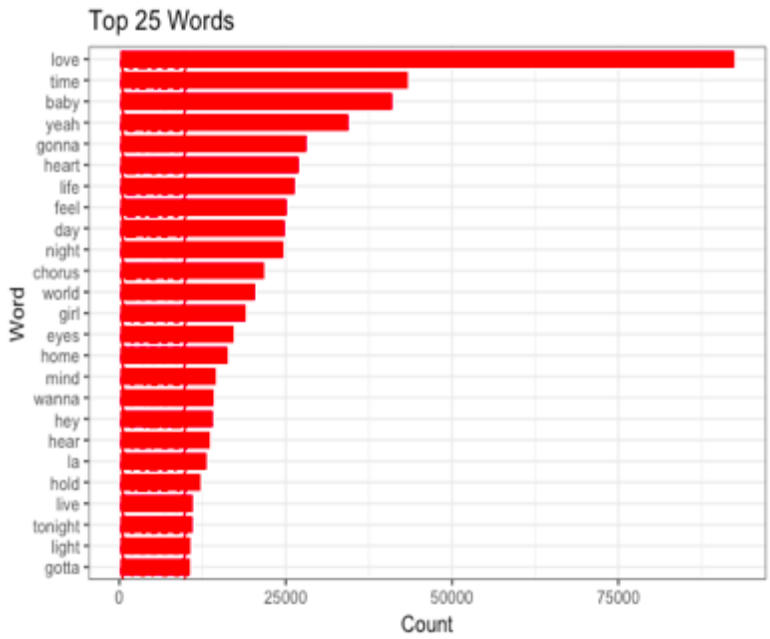
3.) Based on certain sentiments, predict the artist name

5. Text Analysis and Exploration

5.1 Data Exploration

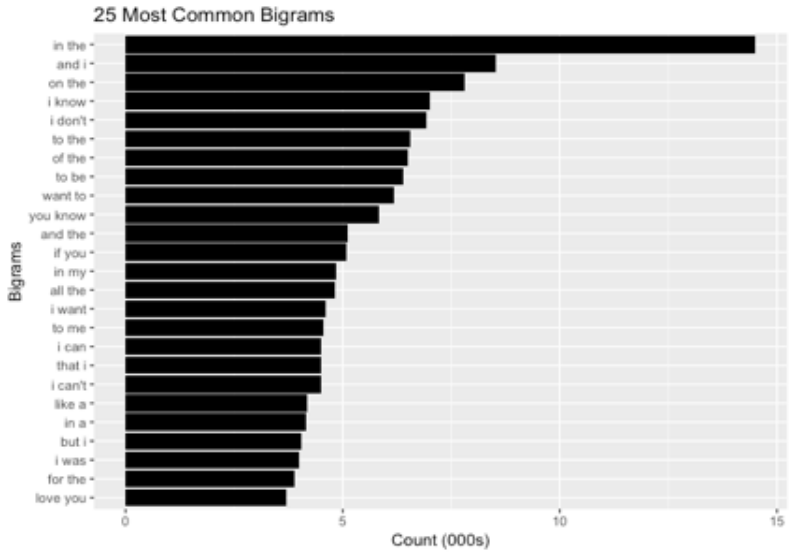
In order to explore the dataset and to get deeper insights from the data, we have made use of visual representations. Visual data analysis is one of the most important steps for data exploration. For our song dataset, we have created visualizations like wordclouds, top bigrams and trigrams, top words showing various emotions, etc. Below are the visualizations to know the data.

5.1.1 Top 25 Words:



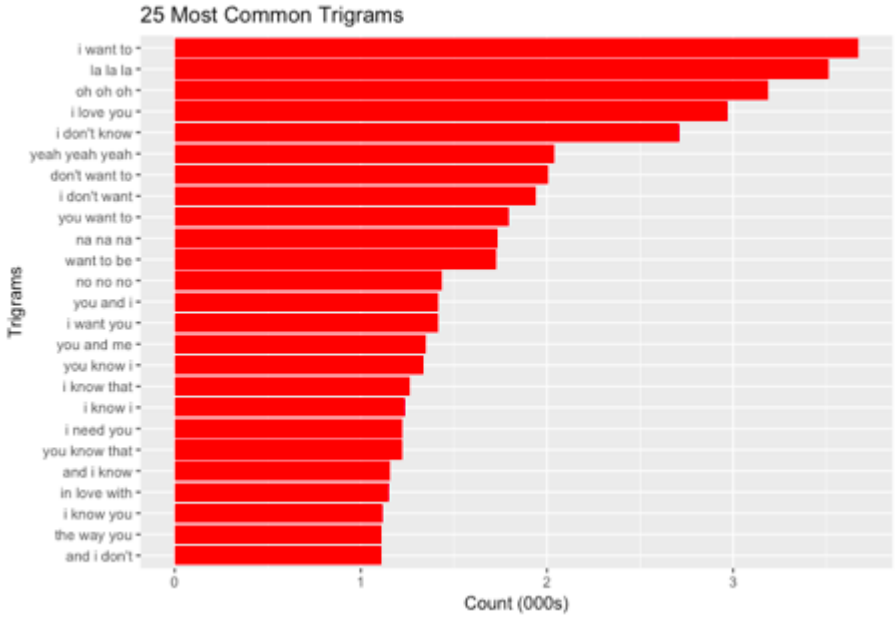
Above is the graph depicting the top words used in the lyrics of all the songs. Some of the words most frequently used are love, time, baby and heart. Surprisingly, the count for word Love is more than double the second word used i.e. Time. The count of word Love is almost 100k, whereas Time is used around 40k times.

5.1.2 Top 25 Bigrams:



The above graph shows us the common bigram which are present in the lyrics of most of the songs. ‘in the’ bigram is most commonly used. Around 15k times, the bigram ‘in the’ Is used. The next bigram which is used for the most number of times is ‘and i’.

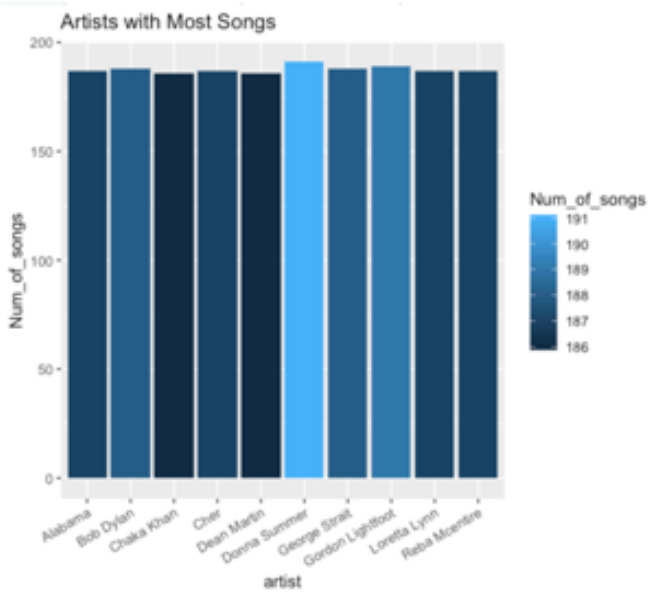
5.1.3 Top 25 Trigrams:



The above graph depicts the most commonly used Trigrams. All the common trigrams completely makes sense if we relate it to the type of songs which is generally being made.

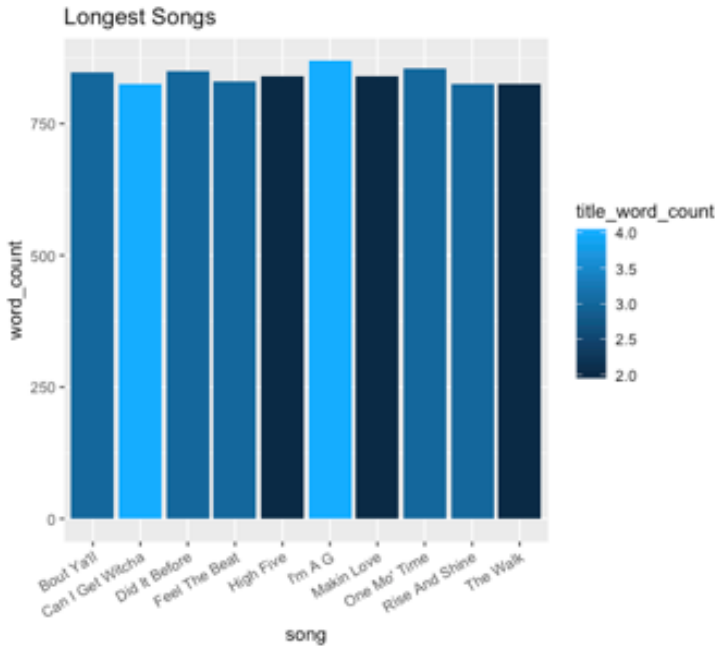
5.1.4 Top 10 Artists:

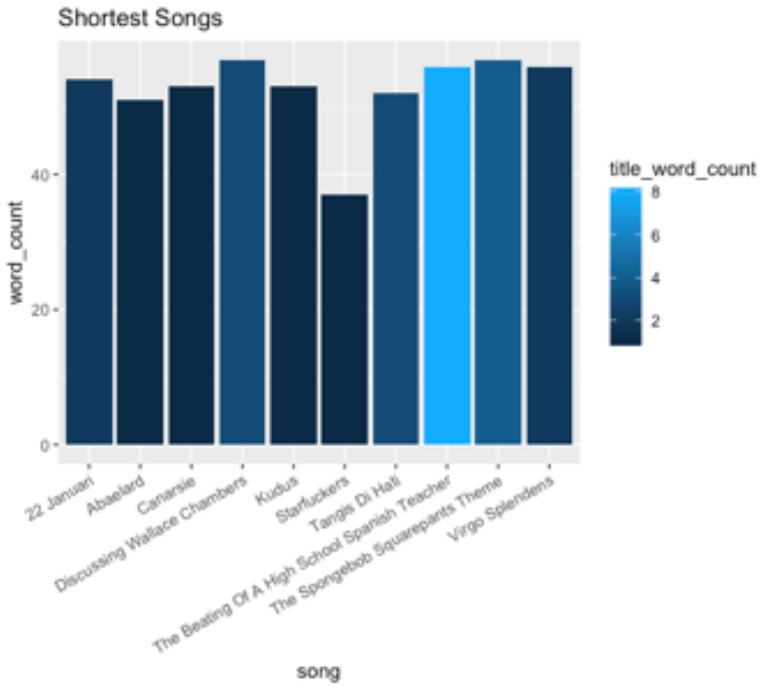
The below graph shows us the top 10 artists based on their number of songs. Most of the top artists has around 190 songs. Donna Summer is the artists which has the highest number of songs. For our further analysis, we can give more important to the artists with more number of songs compared to the artists with less songs.



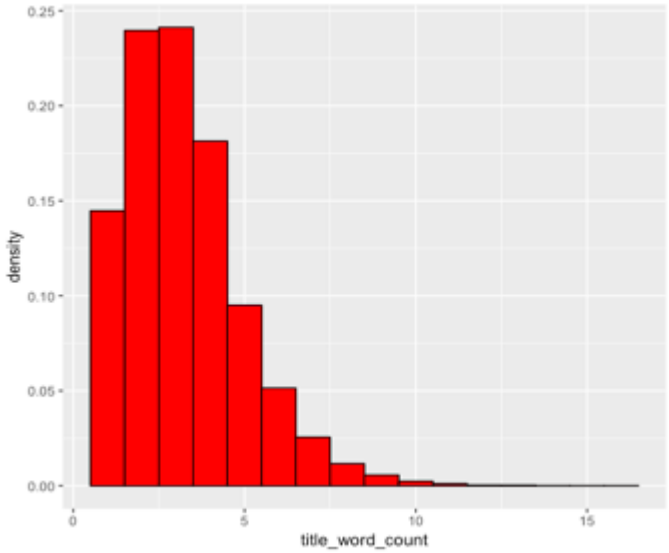
5.1.5 Longest and Shortest Songs:

The below graphs show us the largest and the smallest 10 songs based on their word count.





5.1.6 Title of the Songs:



The above histogram gives us the frequency of words of title. It shows that 2-4 words are most common for a song title.

Below is the wordcloud of the words used for song titles.

5.2 K-Means Document Clustering Analysis

Here we use K-Means clustering to classify the songs with similar lyrics. We choose unigram TfidfVectorizer, and set number of components as 5 and number of clustering as 5. We got the following result for each clustering.

	Cluster	0	1	2	3	4
0	1	0.172257	-0.041473	-0.036976	0.013546	0.028909
1	1	0.078635	-0.021322	0.005658	0.015955	-0.026425
2	1	0.110175	0.034562	-0.006568	-0.004240	0.021709
3	1	0.074691	0.072166	0.004439	0.018011	-0.007521
4	1	0.088499	0.081845	0.003083	0.019414	-0.008087
5	1	0.116077	-0.057334	0.036872	0.028376	0.024519
6	1	0.080927	-0.021834	-0.043804	0.029576	0.003876
7	1	0.101585	-0.013704	-0.028963	0.051995	0.003842
8	0	0.215759	-0.055545	-0.015665	0.004909	0.069152
9	3	0.125300	-0.012774	0.133946	0.121723	-0.012729
10	0	0.258626	-0.050132	-0.022877	0.014880	0.022128
11	1	0.108489	-0.043096	-0.025048	0.042275	0.045995
12	1	0.082959	-0.001084	-0.027988	-0.013653	-0.032177
13	0	0.177021	-0.059540	0.040180	-0.047952	0.035658
14	1	0.093798	-0.005144	0.014029	-0.019166	0.043206
15	1	0.083683	-0.023821	0.048619	0.022255	-0.001930
16	1	0.104532	0.001258	-0.051042	0.041705	0.021611
17	1	0.118766	-0.048631	-0.036339	0.032921	0.056865
18	1	0.040800	-0.012564	-0.023097	0.026143	0.010567
19	1	0.004872	-0.003785	0.004661	0.002384	0.002059
20	1	0.121058	-0.041355	-0.052145	0.035178	0.016255
21	1	0.119350	-0.007743	-0.060675	0.055257	0.038180
22	1	0.054795	-0.023775	-0.017642	0.018665	0.019283
23	1	0.098104	-0.039617	0.007305	-0.017888	0.069988
24	4	0.299562	0.111078	-0.008116	-0.034434	0.080533
25	1	0.127903	-0.013046	-0.039775	0.097901	0.008654
26	1	0.091468	0.003603	-0.019453	0.011470	0.030085
27	1	0.121366	-0.040373	-0.049968	0.049879	0.002032
28	4	0.183786	0.114883	-0.045210	0.007175	0.011152
29	1	0.116696	-0.030499	-0.038043	0.025372	0.020301

```
[57650 rows x 6 columns]
         Cluster
Cluster
0          11624
1          35818
2           3056
3           2328
4           4824
```

Next, we continue to use K-Means for further analysis, and we would like to predict the artist based on lyrics.

We selected ‘artist’ column as labels and got 643 clusters. Setting k-means clustering = random and max iteration number to 300. Then we got the following result.

```
k = 643
km = KMeans(n_clusters=k, algorithm='auto', init='random', n_init= 10, random_state=0, verbose=False)
km.fit(vecs)
```

```
KMeans(algorithm='auto', copy_x=True, init='random', max_iter=300,
    n_clusters=643, n_init=10, n_jobs=None, precompute_distances='auto',
    random_state=0, tol=0.0001, verbose=False)
```

```
HMD_subset_labels = data['artist'].values
print(km.cluster_centers_)
cm2 = pd.crosstab(HMD_subset_labels, km.labels_)
print(cm2)
```

```
[[0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]]
col_0                     0    1    2    3    4    5    6    7    8    9    \
row_0
'n Sync                   0    0    0    0    0    0    0    0    0   68
ABBA                      0    0    0    1    0    0    0    0    0   94
Ace Of Base               0    0    0    0    0    0    0    0    0   66
Adam Sandler              0    0    0    0    0    0    0    0    0   65
Adele                     0    0    0    0    0    0    0    0    0   45
Aerosmith                 0    0    0    0    0    0    0    0    0  153
Air Supply                0    0    0    0    0    0    0    0    0  139
Aiza Seguerra             0    0    0    0    0    0    0    0    0   21
Alabama                   0    0    0    0    0    0    0    0    0  147
Alan Parsons Project      0    0    0    0    0    0    0    0    0   93
```

```
col_0                     ...  633  634  635  636  637  638  639  640  641  \
row_0                     ...
'n Sync                   ...    0    0    0    0    0    0    0    0    0
ABBA                      ...    1    0    0    0    0    0    0    0    0
Ace Of Base               ...    0    0    1    0    0    0    0    0    0
Adam Sandler              ...    0    0    0    0    0    0    0    0    0
Adele                     ...    0    0    0    0    0    0    0    0    0
Aerosmith                 ...    0    0    0    0    1    0    0    0    0
Air Supply                ...    0    0    0    0    0    0    0    0    0
Aiza Seguerra             ...    0    0    0    1    0    0    0    0    1
Alabama                   ...    0    0    3    1    0    0    0    0    0
Alan Parsons Project      ...    0    0    0    0    0    0    0    0    0
Aled Jones                ...    0    0    0    0    0    0    0    0    0
Alice Cooper              ...    0    0    1    0    0    0    0    0    0
Alice In Chains           ...    0    0    0    0    0    0    0    0    0
Alison Krauss             ...    0    0    0    1    0    0    0    0    0
Allman Brothers Band      ...    1    0    0    0    0    0    0    0    0
Alphaville                ...    0    0    0    0    0    0    0    0    0
America                   ...    0    0    0    1    0    0    0    0    0
```

Zac Brown Band	0
Zakk Wylde	0
Zao	0
Zayn Malik	0
Zazie	0
Zebra	0
Zebrahead	0
Zed	0
Zero 7	0
Zeromancer	0
Ziggy Marley	0
Zoe	0
Zoegirl	0
Zornik	0
Zox	0
Zucchero	0
Zwan	0

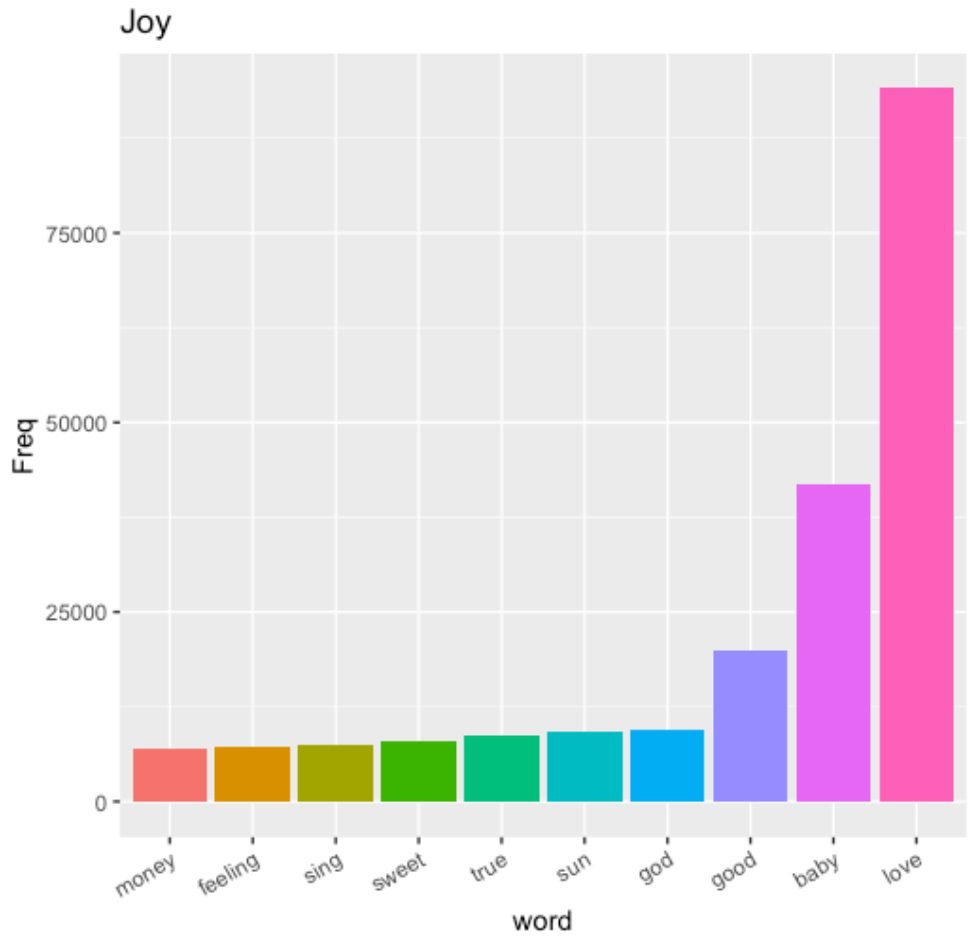
[643 rows x 643 columns]

Sentiment Analysis:

We have performed sentiment analysis for all the songs. We have made use of the NRC Sentiment Lexicon for our analysis on the lyrics. We chose five sentiments from the NRC Lexicon. The chosen five sentiments are Joy, Fear, Sadness, Trust and Anger. We started by analyzing the top words which are present in the lyrics dataset by their respective sentiment.

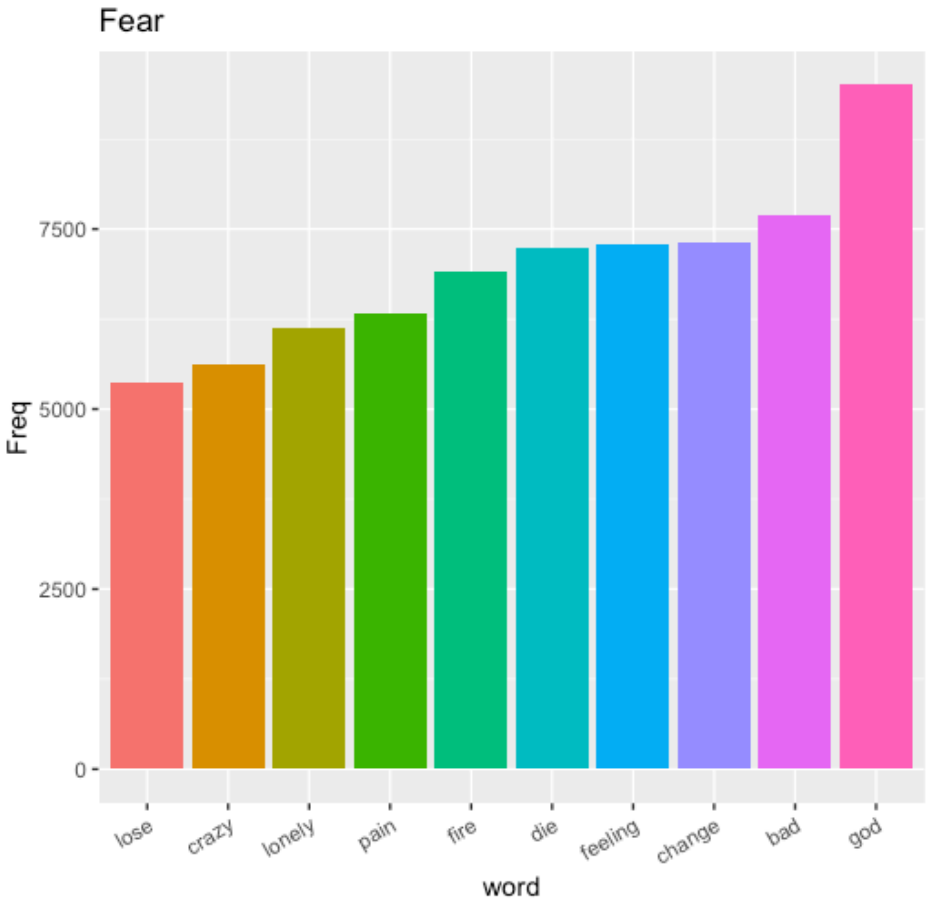
1. Joy – The top words based on their frequency which depicts the emotion Joy are love, baby, good, god, etc. Below are the words with their respective frequency.

	word	Freq
641	love	94070
640	baby	41890
639	good	19884
638	god	9503
637	sun	9087
636	true	8731
635	sweet	7868
634	sing	7369
633	feeling	7288
632	money	7038



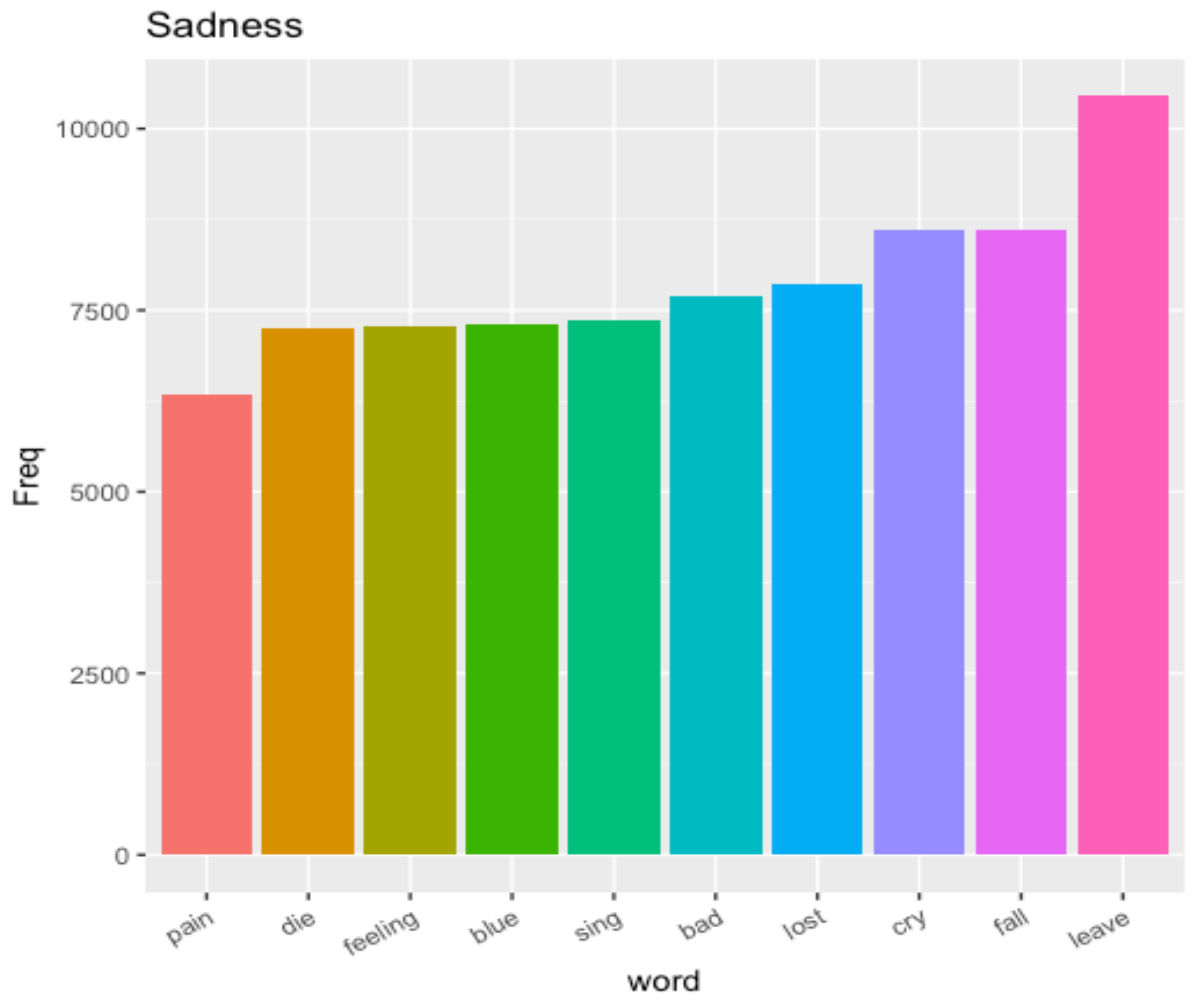
2. Fear – The words depicting the Fear emotion are god, bad, change, feeling, etc. The word god is most frequently used in the whole dataset. Words like pain, lonely, crazy are also used. The frequency of the top words depicting fear ranges from 5000 to 10000.

	word	Freq
1331	god	9503
1330	bad	7699
1329	change	7321
1328	feeling	7288
1327	die	7243
1326	fire	6906
1325	pain	6327
1324	lonely	6117
1323	crazy	5620
1322	lose	5355



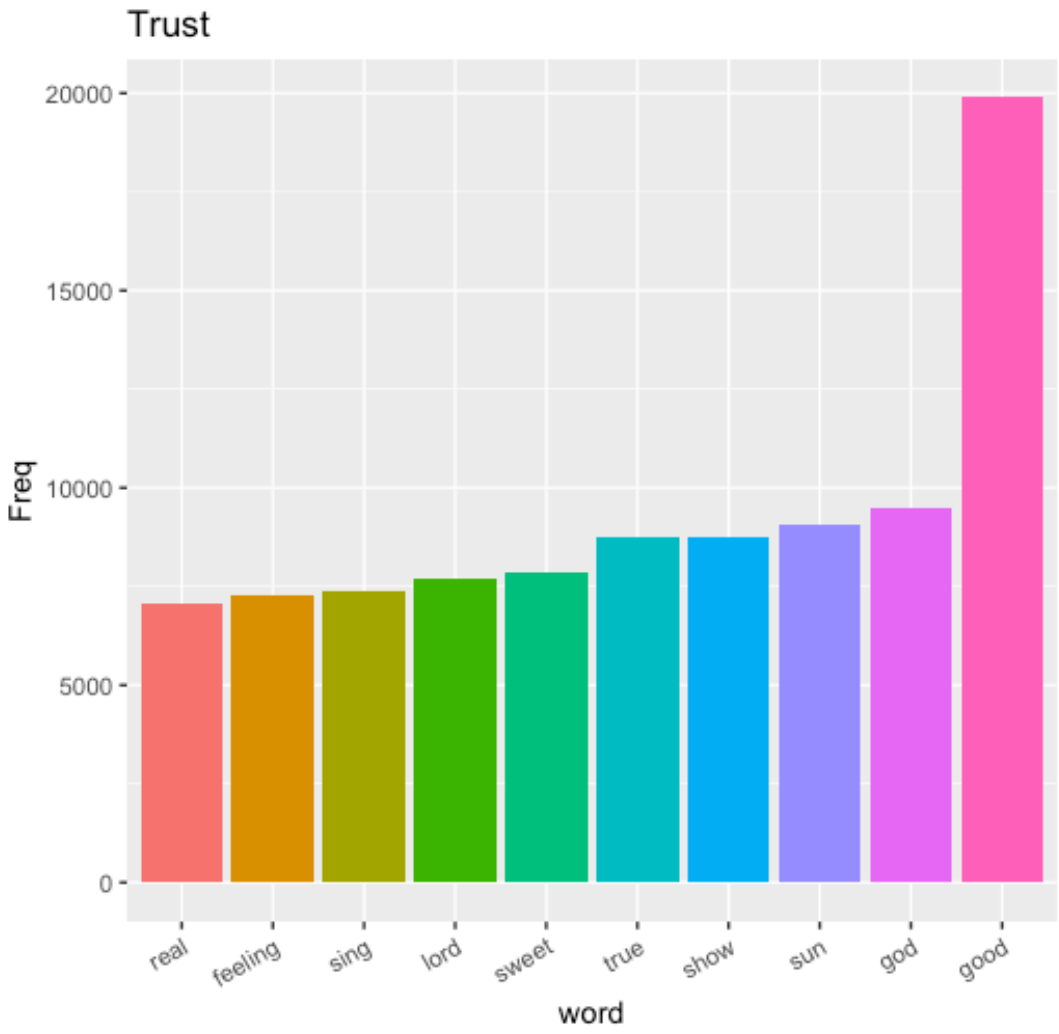
3. Sadness – As expected, words like leave, fall, cry consists of a large part of words showing sad sentiment. The word 'leave' is most commonly used.

	word	Freq
1064	leave	10446
1063	fall	8602
1062	cry	8595
1061	lost	7868
1060	bad	7699
1059	sing	7369
1058	blue	7296
1057	feeling	7288
1056	die	7243
1055	pain	6327



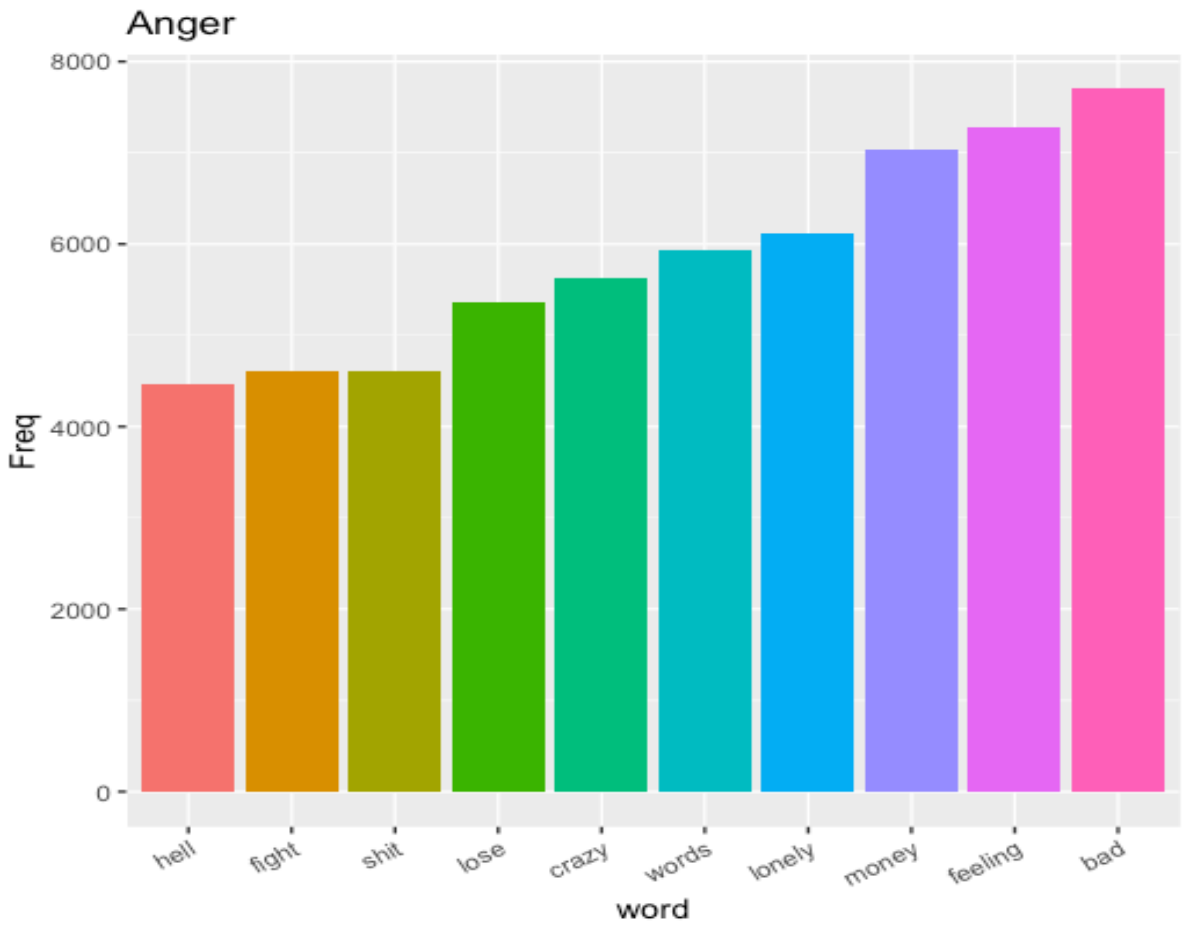
4. Trust – The below words show the Trust sentiment. These are the words used in the dataset most commonly for depicting Trust.

	word	Freq
1061	good	19884
1060	god	9503
1059	sun	9087
1058	show	8756
1057	true	8731
1056	sweet	7868
1055	lord	7672
1054	sing	7369
1053	feeling	7288
1052	real	7061



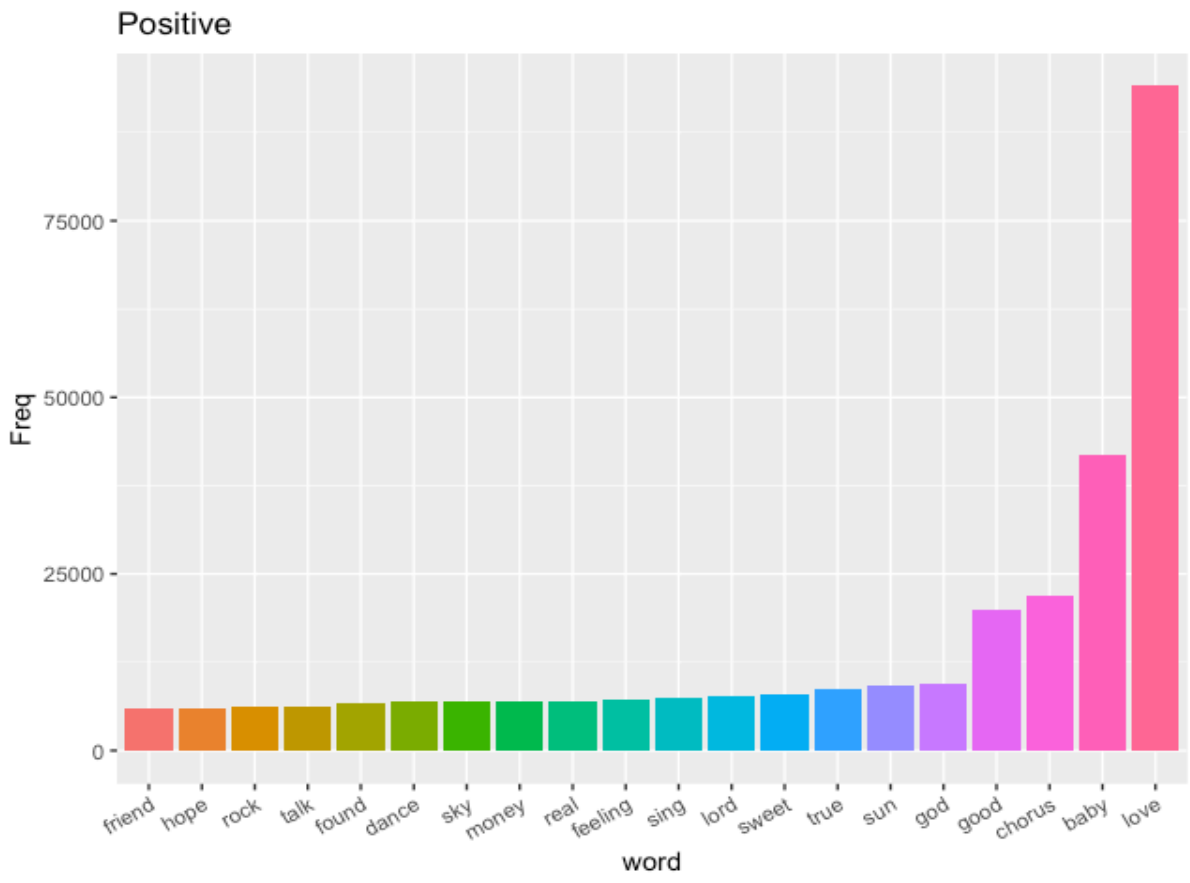
5. Anger – Words like shit, fight, hell are used a lot to depict anger. The frequency of these words ranges from 4k to 8k.

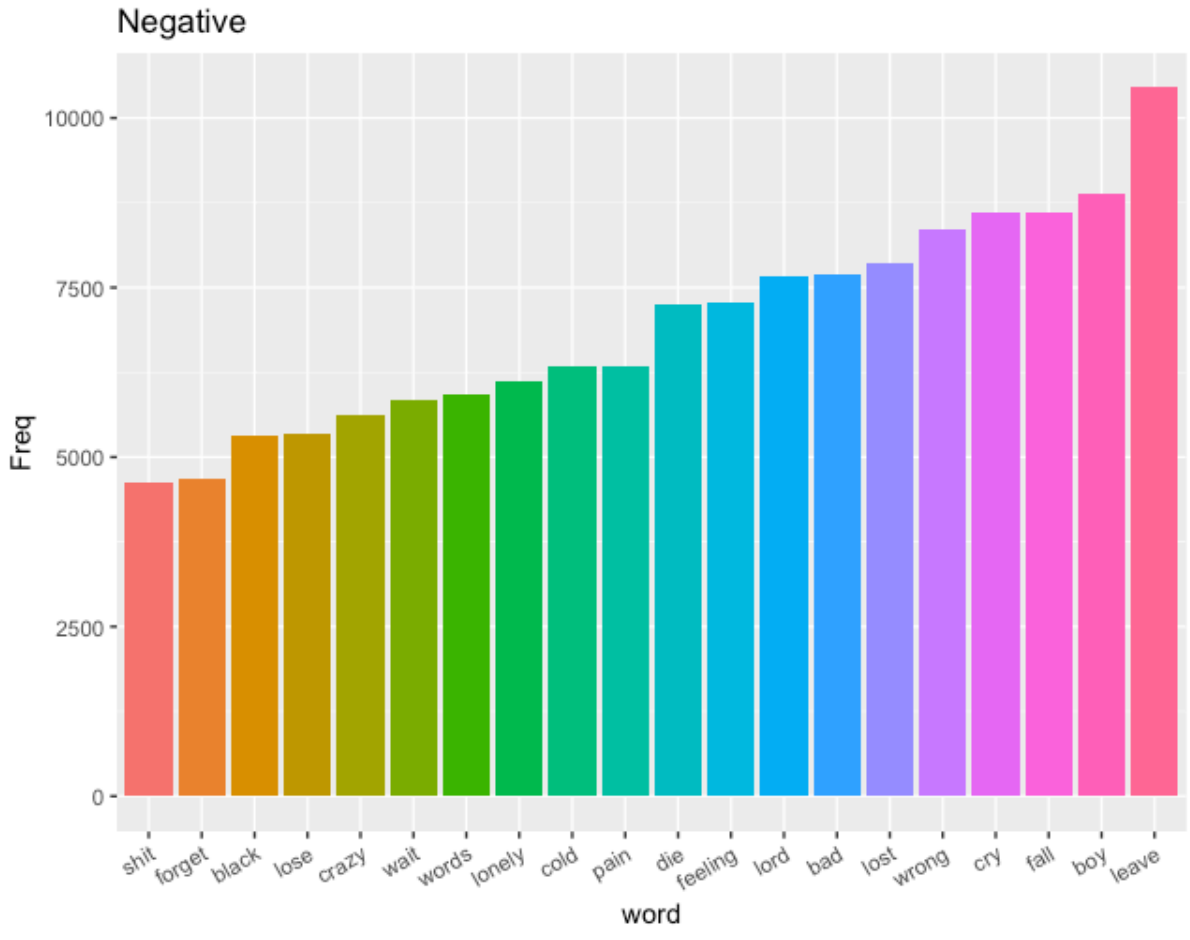
	word	Freq
1110	bad	7699
1109	feeling	7288
1108	money	7038
1107	lonely	6117
1106	words	5925
1105	crazy	5620
1104	lose	5355
1103	shit	4612
1102	fight	4600
1101	hell	4466



Positive & Negative:

We have classified words as Positive & Negative words by using the NRC Lexicon. Below are the graphs of words depictive positive and negative feeling using the NRC. Love and baby are mostly used as positive words whereas leave is used for showing negative emotion.

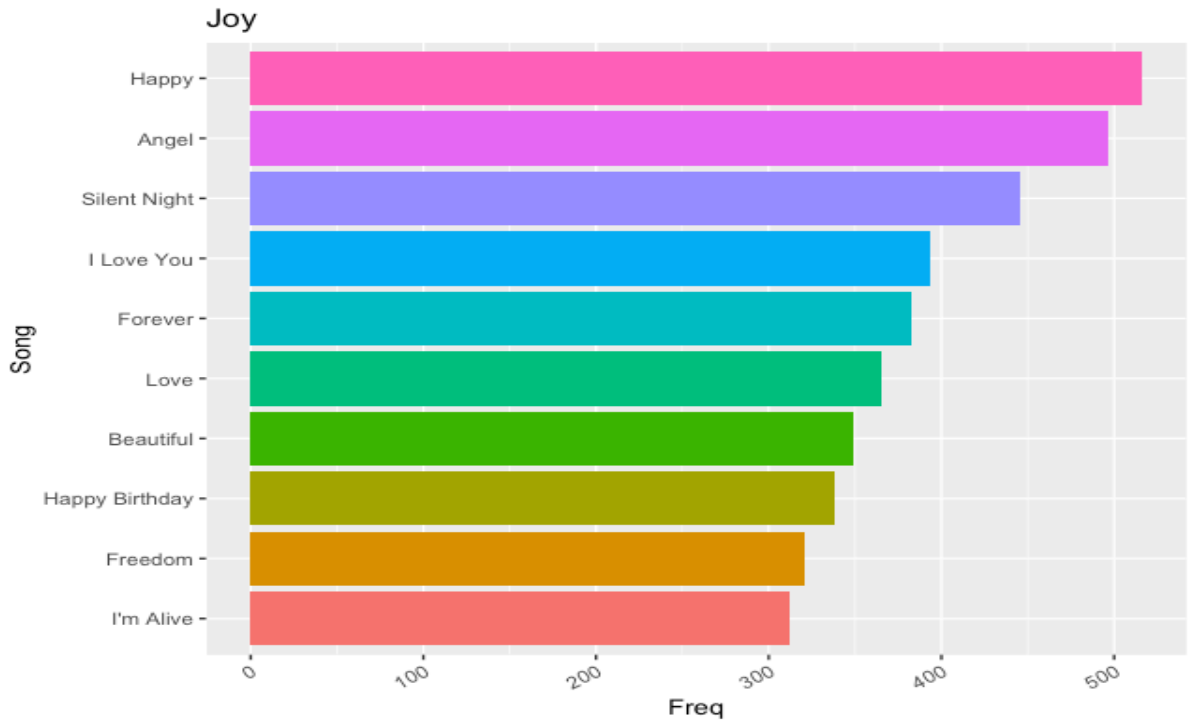

Positive
Freq
75000
50000
25000
0
friend
hope
rock
talk
found
dance
sky
money
real
feeling
sing
lord
sweet
true
sun
god
good
chorus
baby
love
word

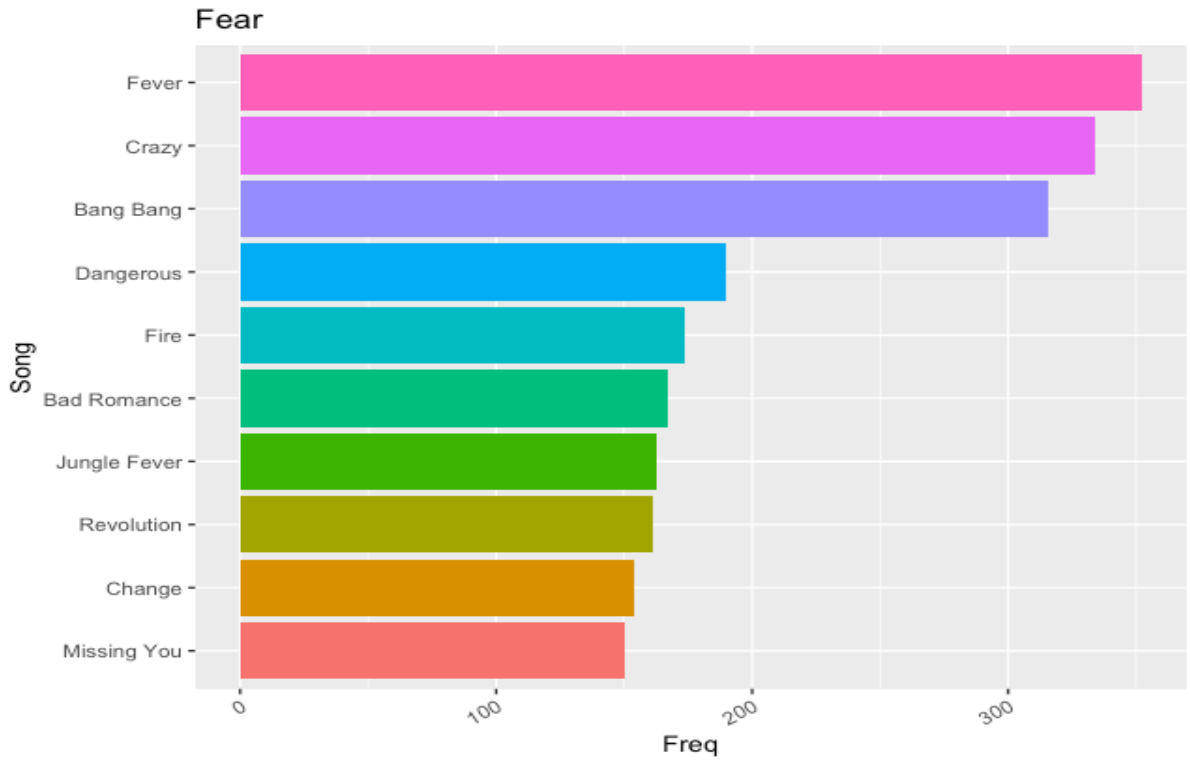

Negative
Freq
10000
7500
5000
2500
0
shit
forget
black
lose
crazy
wait
words
lonely
cold
pain
die
feeling
lord
bad
lost
wrong
cry
fall
boy
leave
word

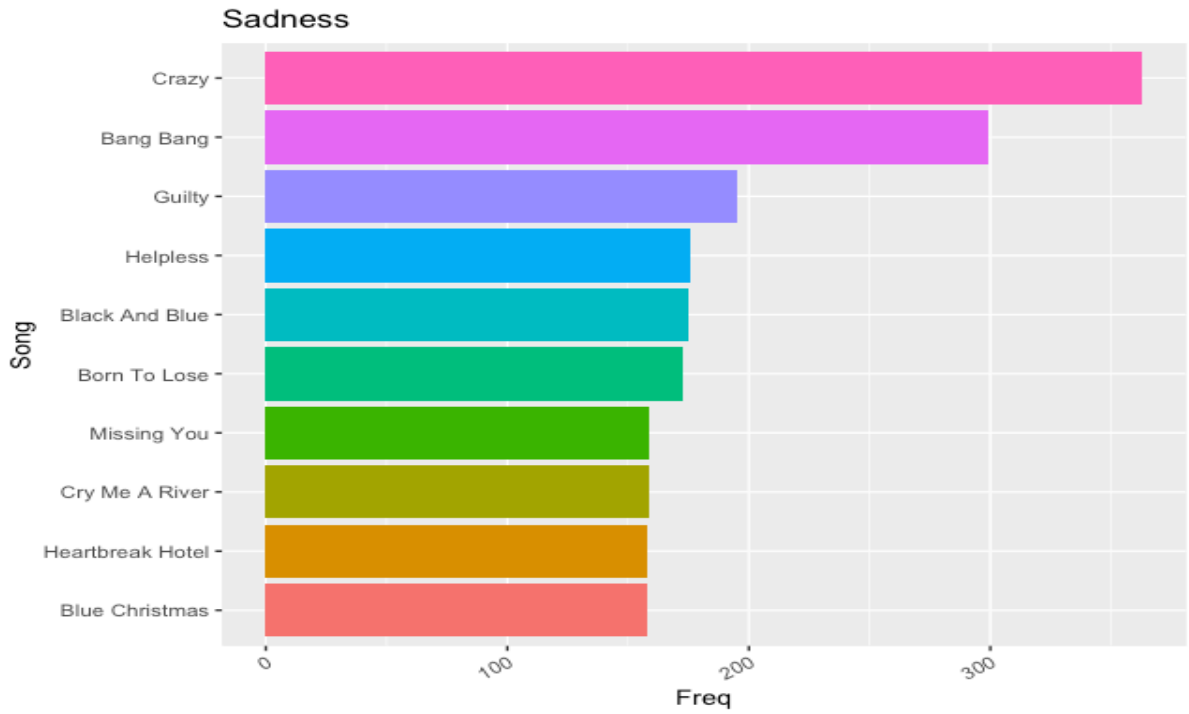
Songs showing Emotions:

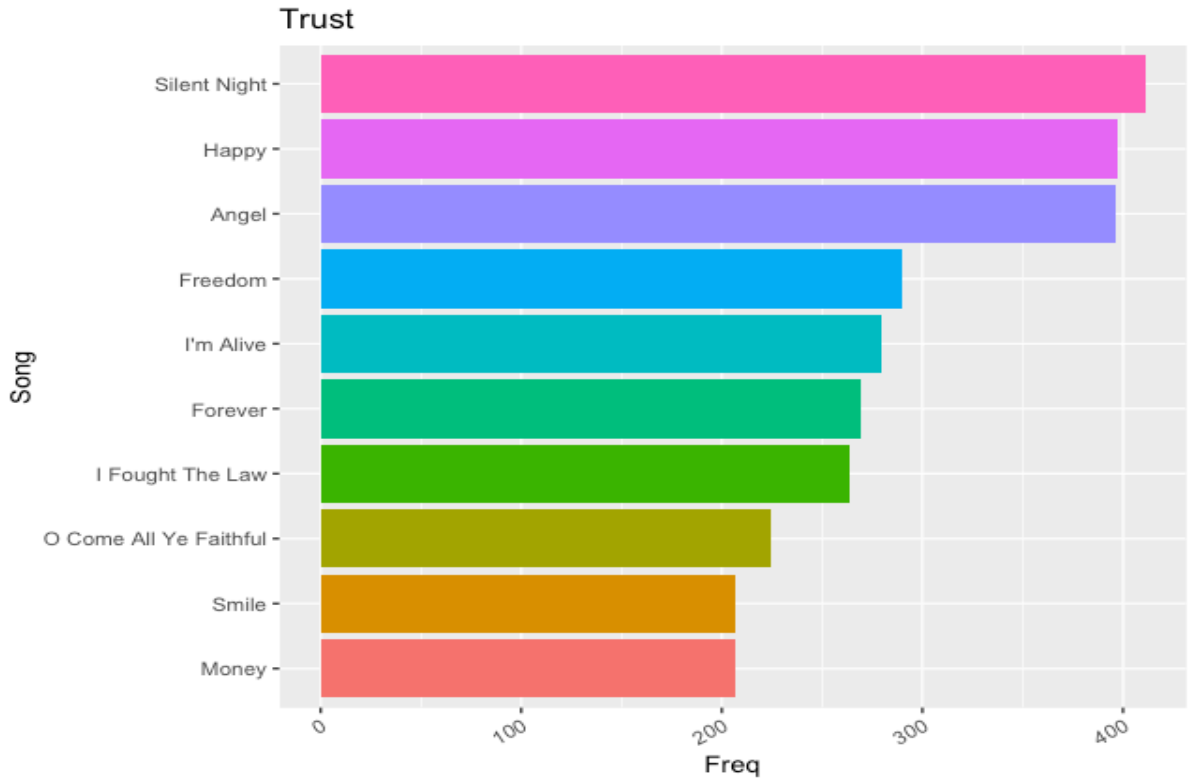
Later, we have found out songs which shows emotions. All the words of each song are used for the analysis. We then find the sentiment of each word. Then the song is assigned to each emotion based on the sentiment of all the words.

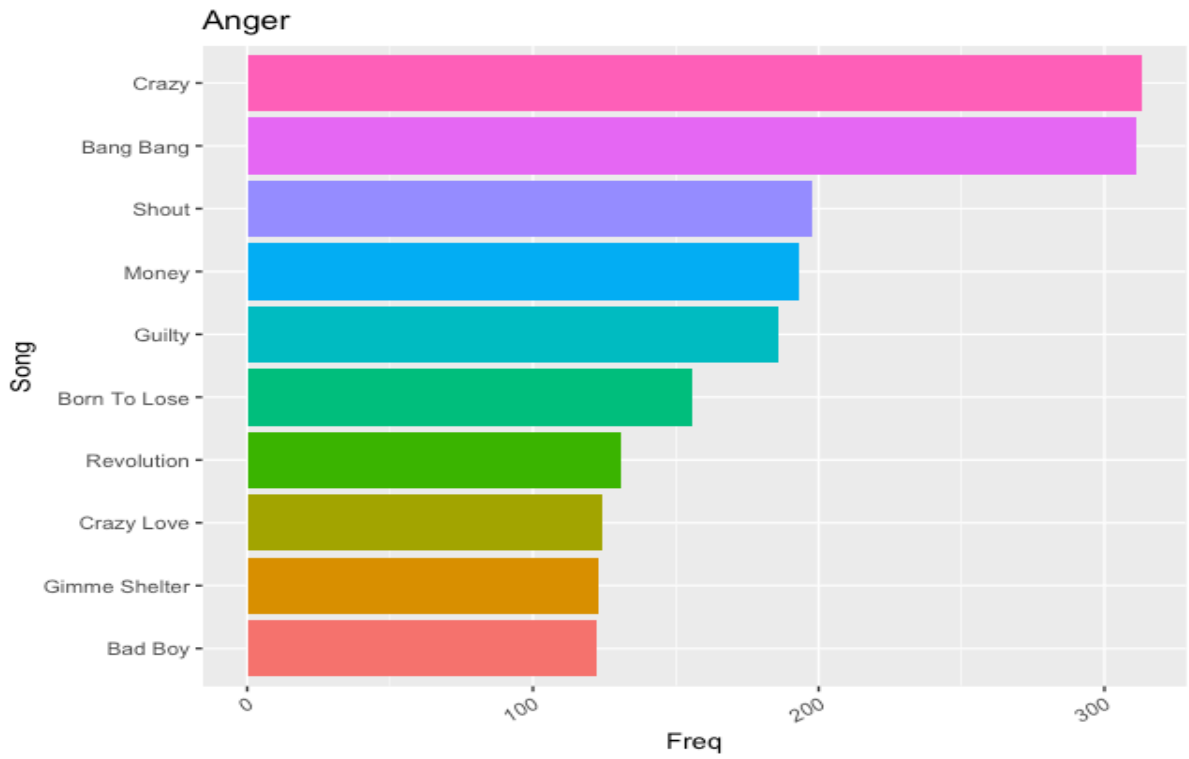
Songs which shows each emotion are shown below. The frequency gives us the frequency of words which depicts the emotions.





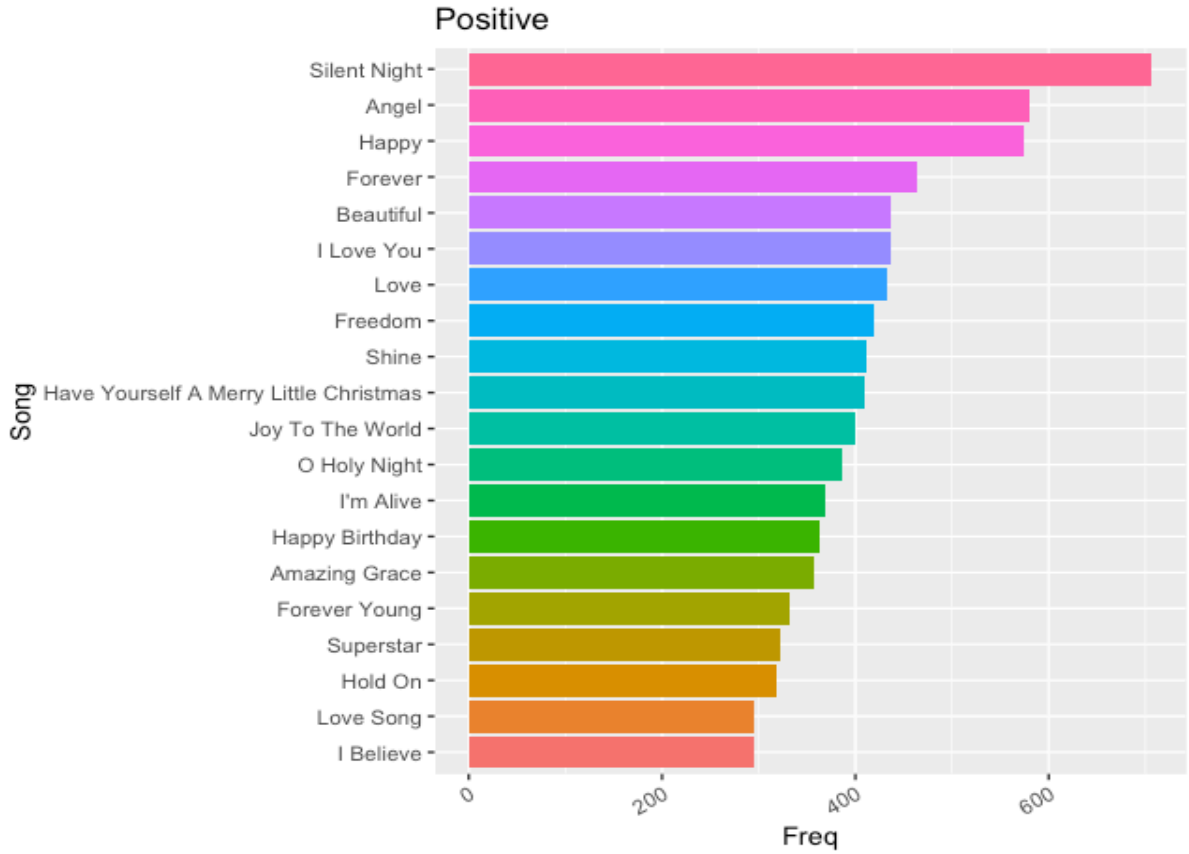

Sadness
Crazy
Bang Bang
Guilty
Helpless
Black And Blue
Born To Lose
Missing You
Cry Me A River
Heartbreak Hotel
Blue Christmas
Song
0
100
200
300
Freq

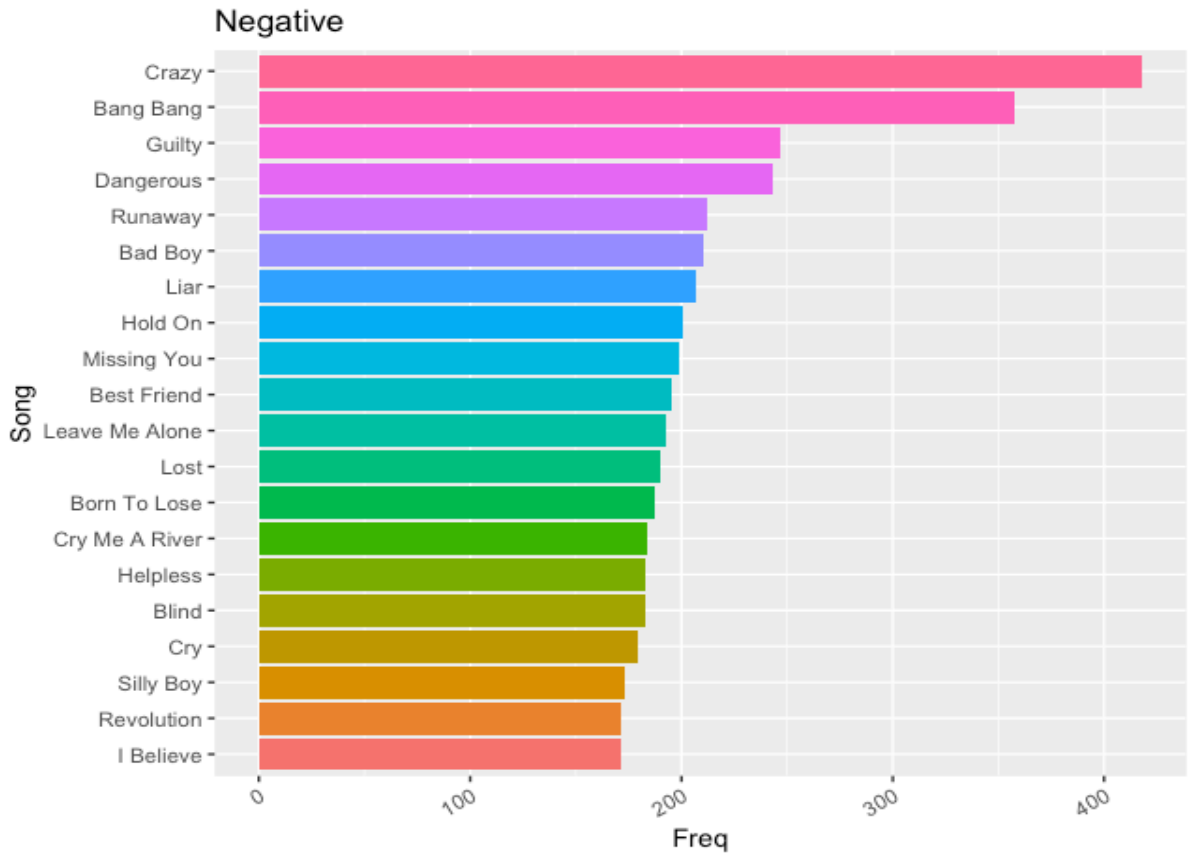

Trust
Silent Night
Happy
Angel
Freedom
I'm Alive
Forever
I Fought The Law
O Come All Ye Faithful
Smile
Money
Song
0
100
200
300
400
Freq



Positive & Negative Songs:

We have also shown songs with positive and negative emotion.





Further, we can also show artists and their preferred emotions based on all their songs. From that, we can predict the genre preferred by each artist. We can also recommend songs based on the emotions preferred by a user.

References:

Kuznetsov, S. 55000+ Song Lyrics. (2017). Kaggle.com. *Retrieved from* https://www.kaggle.com/mousehead/songlyrics