1_DataExploration

August 22, 2019

https://www.kaggle.com/kunstmord/exploring-the-songs # Data Exploration

[1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

[2]:
```
train = pd.read_csv('input/train.csv')
songs = pd.read_csv('input/songs.csv')
test = pd.read_csv('input/test.csv')
```

[3]:
```
print('Statistics From the Dataset: ')
songs_in_train_and_test = np.intersect1d(train['song_id'].unique(),␣
 ↪test['song_id'].unique())
print('Count of Unique Songs in Training Set: ',train['song_id'].nunique())
print('Count of Unique Songs in Testing Set: ',test['song_id'].nunique())
print('Songs that dont appear in Training Set: ',(test['song_id'].nunique() -␣
 ↪songs_in_train_and_test.shape[0]))
print('Percentage: ',(test['song_id'].nunique() - songs_in_train_and_test.
 ↪shape[0]) / test['song_id'].nunique())

print('Users Statistics: ')
users_in_train_and_test = np.intersect1d(train['msno'].unique(), test['msno'].
 ↪unique())
print('Count of Users in Training Set: ',train['msno'].nunique())
print('Count of Users in Testing Set: ',test['msno'].nunique())
print('Users that dont appear in Training Set: ',(test['msno'].nunique() -␣
 ↪users_in_train_and_test.shape[0]))
print('Percentage: ',(test['msno'].nunique() - users_in_train_and_test.
 ↪shape[0]) / test['msno'].nunique())

train_merged = train.merge(songs[['song_id', 'artist_name',␣
 ↪'genre_ids','language']], on='song_id')
test_merged = test.merge(songs[['song_id', 'artist_name',␣
 ↪'genre_ids','language']], on='song_id')

print('Artists Statistics: ')
```

```
artists_in_train_and_test = np.intersect1d(train_merged['artist_name'].
 ↪unique(),test_merged['artist_name'].unique())
print('Count of Artists in Training Set: ',train_merged['artist_name'].
 ↪nunique())
print('Count of Artists in Testing Set: ', test_merged['artist_name'].nunique())
print('Artists that dont appear in Training Set: ',(test_merged['artist_name'].
 ↪nunique() - artists_in_train_and_test.shape[0]))
print('Percentage: ',(test_merged['artist_name'].nunique() -␣
 ↪artists_in_train_and_test.shape[0]) / test_merged['artist_name'].nunique())

print('Language Statistics: ')
langs_in_train_and_test = np.intersect1d(train_merged['language'].
 ↪unique(),test_merged['language'].unique())
print('Number of Languages Present in Training Set: ',train_merged['language'].
 ↪nunique())
print('Number of Languages Present in Testing Set: ', test_merged['language'].
 ↪nunique())
print('Languages that dont appear in Training Set: ',(test_merged['language'].
 ↪nunique() - langs_in_train_and_test.shape[0]))
print('Percentage: ',(test_merged['language'].nunique() -␣
 ↪langs_in_train_and_test.shape[0]) / test_merged['language'].nunique())

print('Genre Statistics: ')
genres_in_train_and_test = np.intersect1d(train_merged['genre_ids'].apply(str).
 ↪unique(),test_merged['genre_ids'].apply(str).unique())
print('Number of Genres Present in Training Set: ',train_merged['genre_ids'].
 ↪nunique())
print('Number of Genres Present in Testing Set: ', test_merged['genre_ids'].
 ↪nunique())
print('Genres that dont appear in Traning Set: ',(test_merged['genre_ids'].
 ↪nunique() - genres_in_train_and_test.shape[0]))
print('Percentage: ',(test_merged['genre_ids'].nunique() -␣
 ↪genres_in_train_and_test.shape[0]) / test_merged['genre_ids'].nunique())
```

```
Statistics From the Dataset:
Count of Unique Songs in Training Set:  359966
Count of Unique Songs in Testing Set:  224753
Songs that dont appear in Training Set:  59873
Percentage:  0.2663946643648806
Users Statistics:
Count of Users in Training Set:  30755
Count of Users in Testing Set:  25131
Users that dont appear in Training Set:  3648
Percentage:  0.14515936492777845
Artists Statistics:
Count of Artists in Training Set:  40582
Count of Artists in Testing Set:  27563
Artists that dont appear in Training Set:  5790
Percentage:  0.21006421652214927
Language Statistics:
Number of Languages Present in Training Set:  10
Number of Languages Present in Testing Set:  10
Languages that dont appear in Training Set:  0
Percentage:  0.0
Genre Statistics:
Number of Genres Present in Training Set:  572
Number of Genres Present in Testing Set:  501
Genres that dont appear in Traning Set:  35
Percentage:  0.06986027944111776
```

[4]:

```
listen_log = train[['msno','song_id','target']].merge(songs,on='song_id')
listen_log_groupby = listen_log[['song_id', 'target']].groupby(['song_id']).
 ↪agg(['mean','count'])
listen_log_groupby.reset_index(inplace=True)
listen_log_groupby.columns = list(map(''.join, listen_log_groupby.columns.
 ↪values))
listen_log_groupby.columns = ['song_id', 'repeat_play_chance', 'plays']

song_data = listen_log_groupby.merge(songs, on='song_id')
song_data['repeat_events'] = song_data['repeat_play_chance'] *␣
 ↪song_data['plays']
```

[5]:

```
song_data.head()
```

[5]:

```
                                        song_id  repeat_play_chance  plays  \
0  +++2AEoM0d8iZTdbnAjUm35bnGKGMXdZJSv4rrWK6JQ=                 0.0      1
1  ++/ACCkEN/+VtgrJxEqeRgRmV4y8pcarDJ9T/yRAi1E=                 0.0      2
2  ++/1JNswCU+za2pYB0cWIbGL5UzWIKtfweX20+GImZA=                 0.0      3
3  ++4/NK5qpbTZWln/6UmykB8cLfRTCCj8E36IKZVzBjM=                 0.0      1
4  ++4Ihbdp0juQ9ldp9DysOL1WTLHIiawg7cnBTn55I/k=                 0.0      1

   song_length genre_ids          artist_name  \
0       223921       921
1       271302       465    Variété Française
2       221413   786|947
3       142471       465  It's Christmas Time
4       169970      2122    Wynton Kelly Trio

                                            composer  \
0                                Chackrit Muckkanaso
1                                                NaN
2                                                NaN
3  Arranged By| Felix Mendelssohn| Gordon Jenkins...
4                                                NaN
```

```
                        lyricist  language  repeat_events
0  Tadakorn; Narongvit Techatanawat      45.0            0.0
1                              NaN      52.0            0.0
2                              NaN      -1.0            0.0
3                              NaN      52.0            0.0
4                              NaN      -1.0            0.0
```

0.1 Relationship between Number of Plays and Repeatability

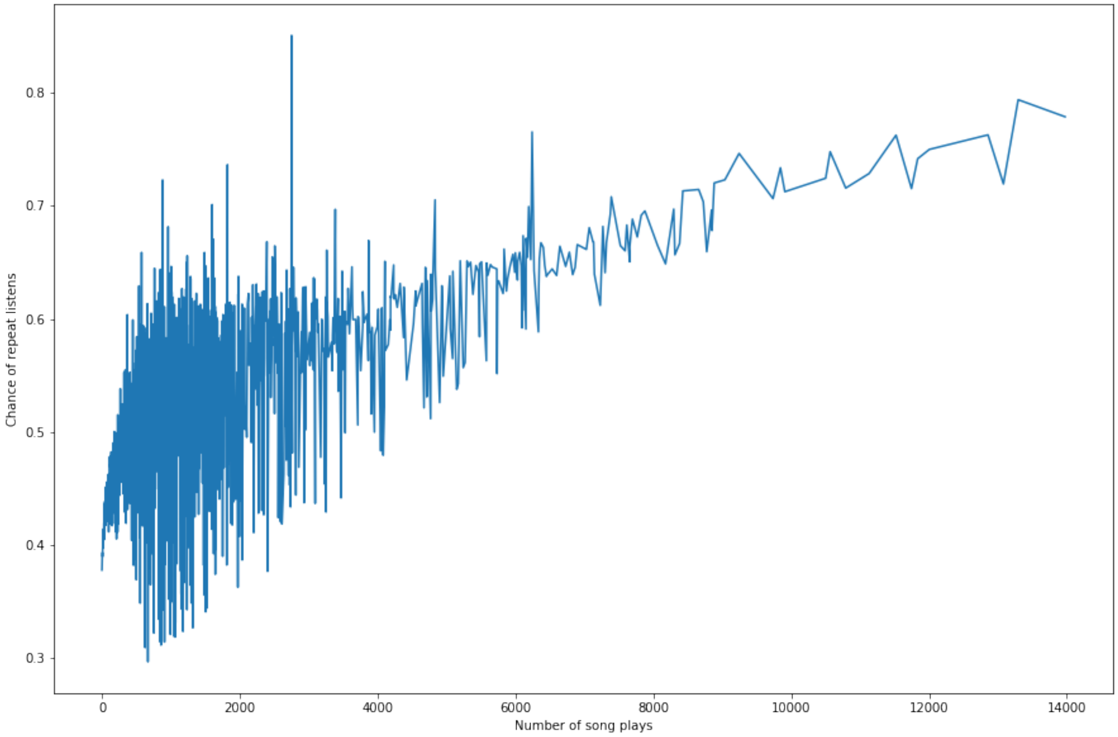
[6]:
```
song_data['plays'].max()
```

[6]:
```
13973
```

[7]:
```
number_of_plays = []
repeat_chance = []

for i in range(1,song_data['plays'].max()+1):
    plays_i = song_data[song_data['plays']==i]
    count = plays_i['plays'].sum()
    if count > 0:
        number_of_plays.append(i)
        repeat_chance.append(plays_i['repeat_events'].sum() / count)
```

[8]:
```
f,axarray = plt.subplots(1,1,figsize=(15,10))
plt.xlabel('Number of song plays')
plt.ylabel('Chance of repeat listens')
plt.plot(number_of_plays, repeat_chance)
```

[8]:
```
[<matplotlib.lines.Line2D at 0x7f9ed2c20630>]
```



[9]:
```
def count_vals(x):
    if type(x) != str:
        return 1
    else:
        return 1 + x.count('|')
```

[10]:
```
song_data['number_of_genres'] = song_data['genre_ids'].apply(count_vals)
song_data['number_of_composers'] = song_data['composer'].apply(count_vals)
song_data['number_of_lyricists'] = song_data['lyricist'].apply(count_vals)
```

[11]:
```
song_data
```

[11]:
```
                                            song_id  repeat_play_chance  \
0      +++2AEoM0d8iZTdbnAjUm35bnGKGMXdZJSv4rrWK6JQ=            0.000000
1      ++/ACCkEN/+VtgrJxEqeRgRmV4y8pcarDJ9T/yRAi1E=            0.000000
2      ++/lJNswCU+za2pYB0cWIbGL5UzWIKtfweX20+GImZA=            0.000000
3      ++4/NK5qpbTZWln/6UmykB8cLfRTCCj8E36IKZVzBjM=            0.000000
4      ++4Ihbdp0juQ9ldp9DysOL1WTLHIiawg7cnBTn55I/k=            0.000000
5      ++6SwJ+aXGV4LLqJmgEogoeEC0DxEdyus0MzD3iuveA=            0.000000
6      ++732ZgaVBo177j83D3Iht3ZeHUctfXg/y47RKvmc3k=            0.000000
7      ++7GdTgp8zbQLYOki7hVPEOHpu+KLZClsGrGiEuL2uI=            0.407407
8      ++8KD5dwLpXTteprbInWnhBQRkYQjmQPiFQLS3bVRLM=            0.000000
9      ++8TsjXZyHVfns0LTmZ+EdTlVY29HQSBxhzRrlSsAqk=            1.000000
10     ++8qtPGXpHX4yK4fUeS5dP+Cb1TcfLA3TE2b+nN/SBA=            1.000000
11     ++9CllWTafshZc7T8X7cvNfUxgDe0WYrJ3TOen026j4=            0.750000
```

```
12      ++9STZwz5v1kTToOlmexz9ZsuoghLuwY5lCQpa//lRs=            1.000000
13      ++A2uqzDg/vUWu0x/cBalbddrVIJXggKKgQCkEnXmkI=            0.428571
14      ++AGwKa7KU0dFAx2MC0nKJaT/jCuE5AqUPcAR2egNDA=            0.000000
15      ++BSW6MczXLSGS5ozEFtuBeJ7sXSIqreldhQAmCInQE=            0.000000
16      ++BW3oO/EHKpfAk08rmYFbAWce6sDyL1f4+xhDJibTQ=            0.000000
17      ++CfKs1t1wU1t0q0UxCdRqGoDpToqgMPmYytklaqo9o=            0.428571
18      ++CnoGMowrYqDI2eQM3aNJMIsxPNx1LD7u8ShTGwAQQ=            0.666667
19      ++D5M7t3luxBwz402CarhgCwYfe3f0b+fE62fY5GA1s=            0.000000
20      ++EBTkZ77PSeSnVQ72CHesRb3907hLqwlRGEZzBNkhs=            0.000000
21      ++EP5+TFokXDxXEhKSyUnGr04r4VYWYnuSJwvWXic/M=            0.000000
22      ++EfwIEFB45OM9YDJlf1QMexyzs7kz2gbum80xJUAvw=            0.000000
23      ++Fww4ED+PbNgOy39UImp1OW8zSukMAOrTiikz2pnig=            0.000000
24      ++FzY6zSfsFwM4DJFirHadkNwepv6qpBgji+6GcJ2/c=            0.500000
25      ++GbLWZvA7LtAQycHNI0/XvyUufZthW9oIf51oyML3k=            0.000000
26      ++GcbohyTNWPf/pXFF9vZ9kesiOeUTwUJraxklStgKM=            1.000000
27      ++HlV9O/nnF8fmlWOgsrJSxO+rx75nHWuE6O3ykpVS8=            1.000000
28      ++IrpeciSQ6NWOp78CLvSLjcJVwBecNHnYzvOrxFAPE=            0.666667
29      ++JIM2H/Tcwm7a4UfY7rETSUzvGp5NUacOz5HxxyNTA=            0.500000
...                                              ...                 ...
359884  zzk/ahisto9RQgKAEBzzSkkpRYrMtUpH7fbePHyoAkY=            0.333333
359885  zzkUULnSf8GbnEsoW/P0dn+SARAN8O8O3auSEgu46Tw=            0.000000
359886  zzkr/ROrk66bqF08yM4kLY5tzfScMlXn2eBuzqXaV0s=            0.000000
359887  zzl4HMzSbgXfWJkj40A6yBOO5iUZAlbbhzP35nIF3e8=            0.416667
359888  zzl6XyGVdTB1Y7Kl9HtgQGaxPnvJualy9ISWGLmUKx4=            1.000000
359889  zzlSz6h5ejz71YBPBkXwbT+7a8jBocKfOnwLKlYn964=            1.000000
359890  zzmOv2OhJ6xWV76JFCLgmWH2/A1t9Fa0Xp2EFvigeU0=            0.500000
359891  zzmTK18lH3b2dZoc+8bkAHI5+VTQNwPeCxiN1MzOeDY=            0.000000
359892  zzmgweXAi6AbNucpiw7g2z/N8JD1feCclDVo976Nqm4=            1.000000
359893  zznm672i3DzAJ85w1Nk1gJv4QbB1OiXxnjr8ky+jI8M=            0.000000
359894  zznrbAaaf5GakiY11hxh5n+ofefNOsQ/TiQP91bBMtI=            0.000000
359895  zznuHa14iW1mMrCEQUMY2SgRxR9xiJ/gW2dKB+KeAsM=            1.000000
359896  zznxMV0cjB7aGaS+fEiuh2sd/SekUGP/iSh4LxBwD4M=            0.565217
359897  zzoE7+U/Ss/ulhUz8anOZFeUwrWXFnyRzXnYbkaK58U=            0.800000
359898  zzp6TYFxcRbuX0CwcMJiqMl8OW9kmoJvIU8DAfz5xs8=            1.000000
359899  zzq7TWRHVtxuY4/v3/8hj04b+KxSZmv4+VN1tGKkC70=            0.000000
359900  zzq7gRJExPGAVxMDTqtADr2Kv213mUtFJ2EmQhkp5EE=            1.000000
359901  zzqCJ/xvRiG6GBv5YGXp/mDN2h3K/68gt6t5byTjcpY=            0.666667
359902  zzqFnGENPxlCveTVr6bIJvuNERE/HtXK6zDTFRfrHp0=            1.000000
359903  zzqFu2/dTaFzWZ7YdB7SN0aILlpbVoonyeULaLBW4mE=            0.326923
359904  zzszihw3XMMQTHFYM56VImzZMZke1RqZop3jHfIzkDA=            1.000000
359905  zztzrWbaiNpznnZFTyWfc56Xsyd1sXcRtih3kJ13OFs=            0.142857
359906  zzu3LS+/DuIiG2KkZCuU6goVDbT3dyy57R03yfJLjkU=            1.000000
359907  zzuRe+6ax33MGabaCk1ThVqCfXtTtm1ASvd92F4VgQY=            0.000000
359908  zzubyBL9pJJy9AZkG2ZYlVG+dQBmPvxVW8jitp4b8MA=            0.000000
359909  zzvfk6Np17ieMkvG9CQNxdDYQENCVLXuYx5VWrNhCvg=            0.500000
359910  zzwCHrZc0TezilVRRTbsiWY6ORNpBq1Zv8GGtHT2xlk=            0.250000
359911  zzwePOl2yF8NLVL7ZfTU2CmtlwnjSruAOpcYiod9iHs=            0.000000
```

```
359912  zzxGtSW9L/V3sRNKZbkOgziHtftZ9/oNvETOTY/QHr8=           1.000000
359913  zzzkIR9d2ggpCr5ofvGZu0JFdjScoIxMgcV3dR4yKh0=           0.000000

        plays  song_length    genre_ids  \
0           1       223921          921
1           2       271302          465
2           3       221413      786|947
3           1       142471          465
4           1       169970         2122
5           2       230086          465
6           1       228362          465
7          54       267075          458
8           2       122416          958
9           1       304405         2022
10          1       227184         1273
11          4       222632          423
12          1       270837         1609
13          7       224757          465
14          1       201920          465
15          1       186514         2122
16          1       308314          958
17         42       224574          465
18          3       214691          465
19          1       298840          NaN
20          1       214552         1259
21          2       458422          465
22          1       158066     1572|275
23          1       102060          465
24          2       230410          465
25          1       202710         2022
26          1       191285          465
27          1       225047         1609
28          3       269281          465
29          6       401705          873
...       ...          ...          ...
359884      9       188151      921|465
359885      2       192156          458
359886      1       174811          921
359887     48       219585          359
359888      1       299630  139|125|109
359889      1       233048          465
359890      2       288624          921
359891      1       282448         2022
359892      1       219585      444|465
359893      3       176013          545
359894      1       172106          958
359895      1       241023         2022
```

```
359896     23       274808         2022
359897      5       237725          465
359898      1       352966         1152
359899      1       282679         2022
359900      1       281240          958
359901      3       200306         1609
359902      1       318392         2122
359903     52       289320          465
359904      1       587859         1609
359905      7       242755          139
359906      1       419909          958
359907     12       223242          444
359908      1       187884      786|947
359909      4       199505         1259
359910      4       144096          458
359911      1       350458          465
359912      2       198344          465
359913      1       217547          940

                                    artist_name  \
0
1                             Variété Française
2
3                           It's Christmas Time
4                             Wynton Kelly Trio
5
6                               Various Artists
7                                (Cindy Yen)
8                                 Simone Kermes
9                                    Mago de Oz
10                                         U180
11                             Enrique Iglesias
12                              Various Artists
13
14                              Various Artists
15                            ECHO MUSIC
16                          Sir Roger Norrington
17                                       JUNIEL
18                                 Kenny Rogers
19                                 DEAN FUJIOKA
20                                      C Losta
21                                  Celine Dion
22                                     Basement
23           Singin In The Rain - Original Cast
24                               Britney Spears
25                                       Seacat
26                                   Will Young
```

```
27                              Various Artists
28                                  Rod Stewart
29                             Hillsong Worship
...                                         ...
359884
359885                        (JIANG Yaojia)
359886                          Various Artists
359887                                    Kaleo
359888                              Glenn Lewis
359889                          Jahméne Douglas
359890                                   Cagnet
359891                         Crimson Massacre
359892                               RECORDBELL
359893                 The Phantom of the opera
359894  Berliner Philharmoniker| Claudio Abbado
359895                          Various Artists
359896                              Joonil Jung
359897                              Anne Murray
359898
359899                               Bloc Party
359900                           Andrea Bocelli
359901
359902                            Erroll Garner
359903                             Mariah Carey
359904               M.A.N.D.Y. vs. Adultnapper
359905                                   Jaheim
359906                     Ludwig Van Beethoven
359907                                       IU
359908
359909                        Soulja Boy Tellem
359910                      (Monkey Pilot)
359911                                Tom Waits
359912                              Pat Monahan
359913                                      Spa

                                                 composer  \
0                                    Chackkrit Muckkanaso
1                                                     NaN
2                                                     NaN
3       Arranged By| Felix Mendelssohn| Gordon Jenkins...
4                                                     NaN
5                                                     NaN
6                                              Jack White
7                                                Cindy
8                                         Antonio Vivaldi
9                                                     NaN
10                                            Soundzimage
```

```
11      Enrique Iglesias| Ray El Ingeniero Casillas|...
12                                                   NaN
13
14                                             Eric Kwok
15      Alessandro Sgreccia| Pierfrancesco Bazzoffi| G...
16                                     Felix Mendelssohn
17                           TWO FACE |  (Lee Sang Ho)
18                                                   NaN
19                                                   NaN
20                                                   NaN
21                                          Jim Steinman
22                                                   NaN
23                                                   NaN
24                       Britney Spears| A. Stamatelatos
25                                                   NaN
26                                                   NaN
27                               L. Shipstad| Z. Mahmoud
28                                                   NaN
29                                                   NaN
...                                                  ...
359884
359885
359886                                               NaN
359887                                               NaN
359888                                               NaN
359889  David Guetta| Sia Furler| Giorgio H. Tuinfort|...
359890                                            Cagnet
359891                                               NaN
359892                                       Josung Gang
359893                                        Tim Sutton
359894                                               NaN
359895                                               NaN
359896                                       Joonil Jung
359897                                     Randy Goodrum
359898
359899                                               NaN
359900         Gian Pietro Felisatti| Malise| Gloria Nuti
359901                                               NaN
359902                       Harold Arlen| Johnny Mercer
359903                                               NaN
359904                                               NaN
359905                                               NaN
359906                              Ludwig van Beethoven
359907                                               NaN
359908                                               NaN
359909                                  D. Way| B. Green
359910
```

```
359911                                                NaN
359912                                                NaN
359913                                                NaN

                                                 lyricist  language  \
0                        Tadakorn; Narongvit Techatanawat      45.0
1                                                     NaN      52.0
2                                                     NaN      -1.0
3                                                     NaN      52.0
4                                                     NaN      -1.0
5                                                     NaN       3.0
6                              Jon Athan/Ian Gari VCowtan      52.0
7                                                |        3.0
8                                                     NaN      -1.0
9                                                     NaN      52.0
10                                                    NaN      52.0
11                                                    NaN      52.0
12                                                    NaN      52.0
13                                                         10.0
14                                                    NaN      24.0
15                                                    NaN      52.0
16                                                    NaN      -1.0
17                                       (Han Sung Ho)      31.0
18                                                    NaN      52.0
19                                                    NaN      17.0
20                                                    NaN      52.0
21                                           Jim Steinman      52.0
22                                                    NaN      52.0
23                                                    NaN      52.0
24                        Britney Spears| A. Stamatelatos      52.0
25                                                    NaN      52.0
26                                                    NaN      52.0
27                                                    NaN      52.0
28                                                    NaN      52.0
29                                                    NaN      52.0
...                                                   ...       ...
359884                                                      3.0
359885                                                     3.0
359886                                                NaN      52.0
359887                                                NaN      52.0
359888                                                NaN      -1.0
359889  David Guetta| Sia Furler| Giorgio H. Tuinfort|...      52.0
359890                                                NaN      52.0
359891                                                NaN      52.0
359892                                        Josung Gang      31.0
359893                                                NaN      -1.0
359894                                                NaN      -1.0
```

```
359895                                                NaN      52.0
359896                                        Joonil Jung      31.0
359897                                                NaN      52.0
359898                                                      -1.0
359899                                                NaN      52.0
359900                                                NaN      52.0
359901                                                NaN       3.0
359902                                                NaN      52.0
359903                                                NaN      52.0
359904                                                NaN      -1.0
359905                                                NaN      -1.0
359906                                                NaN      -1.0
359907                                                NaN      31.0
359908                                                NaN      -1.0
359909                                                NaN      52.0
359910                                                NaN       3.0
359911                                                NaN      52.0
359912                                                NaN      52.0
359913                                                NaN      -1.0

        repeat_events  number_of_genres  number_of_composers  \
0                 0.0                 1                    1
1                 0.0                 1                    1
2                 0.0                 2                    1
3                 0.0                 1                    4
4                 0.0                 1                    1
5                 0.0                 1                    1
6                 0.0                 1                    1
7                22.0                 1                    1
8                 0.0                 1                    1
9                 1.0                 1                    1
10                1.0                 1                    1
11                3.0                 1                    4
12                1.0                 1                    1
13                3.0                 1                    1
14                0.0                 1                    1
15                0.0                 1                    4
16                0.0                 1                    1
17               18.0                 1                    2
18                2.0                 1                    1
19                0.0                 1                    1
20                0.0                 1                    1
21                0.0                 1                    1
22                0.0                 2                    1
23                0.0                 1                    1
24                1.0                 1                    2
25                0.0                 1                    1
```

```
26                1.0                 1                    1
27                1.0                 1                    2
28                2.0                 1                    1
29                3.0                 1                    1
...               ...               ...                  ...
359884            3.0                 2                    1
359885            0.0                 1                    1
359886            0.0                 1                    1
359887           20.0                 1                    1
359888            1.0                 3                    1
359889            1.0                 1                    4
359890            1.0                 1                    1
359891            0.0                 1                    1
359892            1.0                 2                    1
359893            0.0                 1                    1
359894            0.0                 1                    1
359895            1.0                 1                    1
359896           13.0                 1                    1
359897            4.0                 1                    1
359898            1.0                 1                    1
359899            0.0                 1                    1
359900            1.0                 1                    3
359901            2.0                 1                    1
359902            1.0                 1                    2
359903           17.0                 1                    1
359904            1.0                 1                    1
359905            1.0                 1                    1
359906            1.0                 1                    1
359907            0.0                 1                    1
359908            0.0                 2                    1
359909            2.0                 1                    2
359910            1.0                 1                    1
359911            0.0                 1                    1
359912            2.0                 1                    1
359913            0.0                 1                    1

        number_of_lyricists
0                         1
1                         1
2                         1
3                         1
4                         1
5                         1
6                         1
7                         2
8                         1
9                         1
```

```
10                        1
11                        1
12                        1
13                        1
14                        1
15                        1
16                        1
17                        1
18                        1
19                        1
20                        1
21                        1
22                        1
23                        1
24                        2
25                        1
26                        1
27                        1
28                        1
29                        1
...                     ...
359884                    1
359885                    1
359886                    1
359887                    1
359888                    1
359889                    4
359890                    1
359891                    1
359892                    1
359893                    1
359894                    1
359895                    1
359896                    1
359897                    1
359898                    1
359899                    1
359900                    1
359901                    1
359902                    1
359903                    1
359904                    1
359905                    1
359906                    1
359907                    1
359908                    1
359909                    1
```

```
359910                       1
359911                       1
359912                       1
359913                       1

[359914 rows x 13 columns]
```

[12]:

```
n_genres_max = song_data['number_of_genres'].max()
n_composers_max = song_data['number_of_composers'].max()
n_lyricists_max = song_data['number_of_lyricists'].max()

print(n_genres_max, n_composers_max, n_lyricists_max)
```

```
8 51 23
```

0.2 Relationship between number of Genre, Composer and Lyricist to the Chance of Repeating a Song

[13]:

```
x_genres = list(range(1,n_genres_max+1))
x_composers = list(range(1,n_composers_max+1))
x_lyricists = list(range(1,n_lyricists_max+1))

y_genres = [song_data[song_data['number_of_genres'] == x].shape[0] for x in␣
 ↪x_genres]
y_composers = [song_data[song_data['number_of_composers'] == x].shape[0] for x␣
 ↪in x_composers]
y_lyricists = [song_data[song_data['number_of_lyricists'] == x].shape[0] for x␣
 ↪in x_lyricists]

empty_ids = [i for i, y in enumerate(y_composers) if y == 0]
x_composers_fixed = [x_composers[i] for i in range(0,n_composers_max) if i not␣
 ↪in empty_ids]
y_composers_fixed = [y_composers[i-1] for i in x_composers_fixed]

empty_ids = [i for i, y in enumerate(y_lyricists) if y == 0]
x_lyricists_fixed = [x_lyricists[i] for i in range(0,n_lyricists_max) if i not␣
 ↪in empty_ids]
y_lyricists_fixed = [y_lyricists[i-1] for i in x_lyricists_fixed]

y_repeat_chance_g = []
y_plays_g = []

for i in range(1,n_genres_max+1):
    genres_i = song_data[song_data['number_of_genres']==i]
    count = genres_i['plays'].sum()
    y_repeat_chance_g.append(genres_i['repeat_events'].sum() / count)
    y_plays_g.append(count)
```

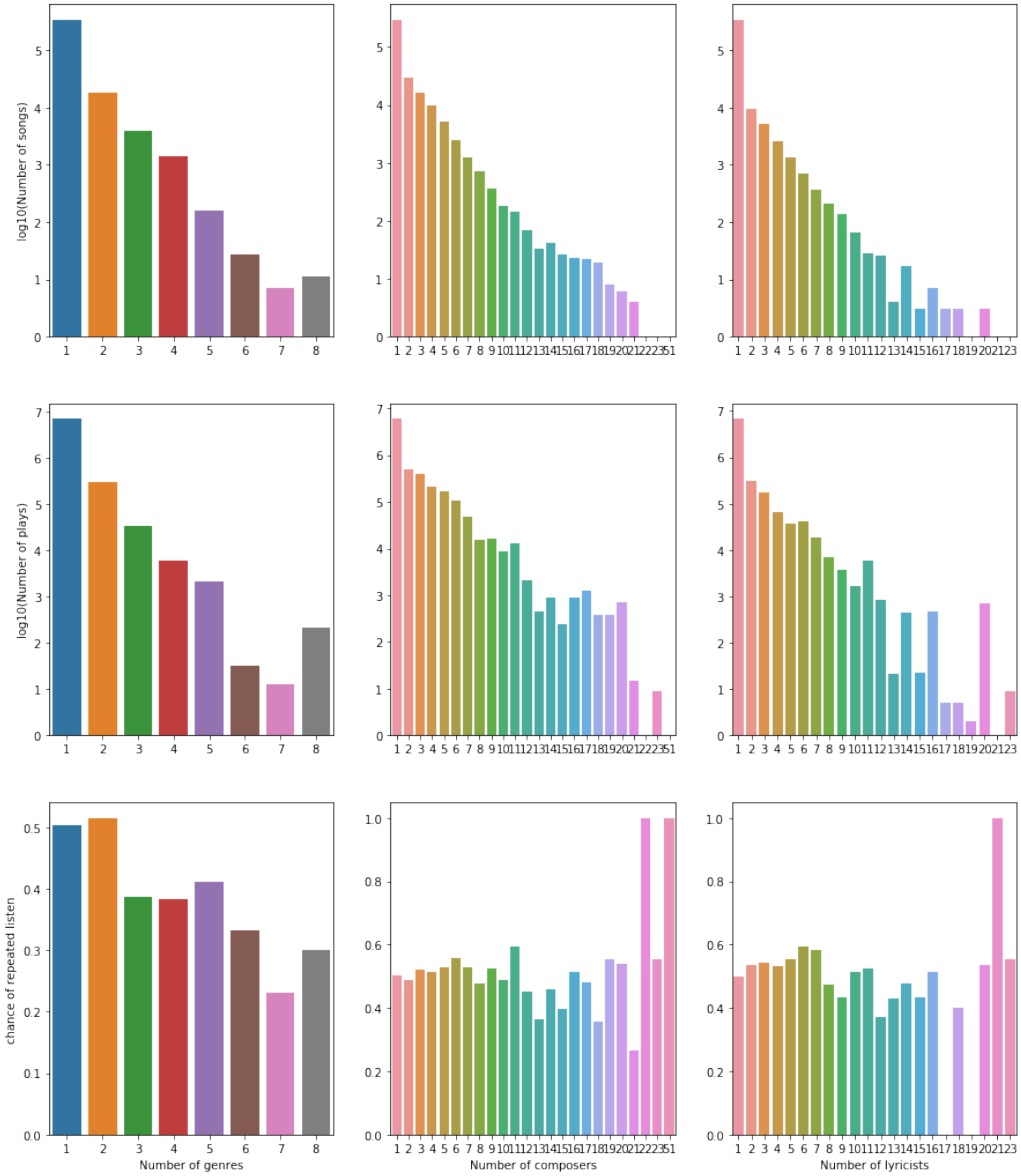
```
y_repeat_chance_c = []
y_plays_c = []

for i in x_composers_fixed:
    composers_i = song_data[song_data['number_of_composers']==i]
    count = composers_i['plays'].sum()
    y_repeat_chance_c.append(composers_i['repeat_events'].sum() / count)
    y_plays_c.append(count)

y_repeat_chance_l = []
y_plays_l = []

for i in x_lyricists_fixed:
    lyricists_i = song_data[song_data['number_of_lyricists']==i]
    count = lyricists_i['plays'].sum()
    y_repeat_chance_l.append(lyricists_i['repeat_events'].sum() / count)
    y_plays_l.append(count)
```

[14]:

```
fig = plt.figure(figsize=(15, 18))

ax331 = plt.subplot(3,3,1)
sns.barplot(x=x_genres,y=np.log10(y_genres))
ax331.set_ylabel('log10(Number of songs)')
ax334 = plt.subplot(3,3,4)
sns.barplot(x=x_genres,y=np.log10(y_plays_g))
ax334.set_ylabel('log10(Number of plays)')
ax337 = plt.subplot(3,3,7)
sns.barplot(x=x_genres,y=y_repeat_chance_g)
ax337.set_xlabel('Number of genres')
ax337.set_ylabel('chance of repeated listen')

plt.subplot(3,3,2)
sns.barplot(x=x_composers_fixed,y=np.log10(y_composers_fixed))
plt.subplot(3,3,5)
sns.barplot(x=x_composers_fixed,y=np.log10(y_plays_c))
ax338 = plt.subplot(3,3,8)
sns.barplot(x=x_composers_fixed,y=y_repeat_chance_c)
ax338.set_xlabel('Number of composers')


plt.subplot(3,3,3)
sns.barplot(x=x_lyricists_fixed,y=np.log10(y_lyricists_fixed))
plt.subplot(3,3,6)
sns.barplot(x=x_lyricists_fixed,y=np.log10(y_plays_l))
ax339 = plt.subplot(3,3,9)
sns.barplot(x=x_lyricists_fixed,y=y_repeat_chance_l)
```

```
ax339.set_xlabel('Number of lyricists')
```

```
[14]: Text(0.5, 0, 'Number of lyricists')
```



0.3 Relationship between Language and Number of Songs, Number of Plays and Chance of Repeat

[15]:

```
languages = song_data['language'].unique()
print(languages,languages.shape[0])

language_count = []
language_plays = []
language_repeat_chance = []

for l in languages:
    if not np.isnan(l):
        songs_with_language = song_data[song_data['language']==l]
        count = songs_with_language['plays'].sum()
        language_repeat_chance.append(songs_with_language['repeat_events'].
 ↪sum() / count)
        language_count.append(songs_with_language.shape[0])
        language_plays.append(count)
    else:
        songs_with_language = song_data[pd.isnull(song_data['language'])]
        count = songs_with_language['plays'].sum()
        language_repeat_chance.append(songs_with_language['repeat_events'].
 ↪sum() / count)
        language_count.append(songs_with_language.shape[0])
        language_plays.append(count)

languages[10] = -100
```

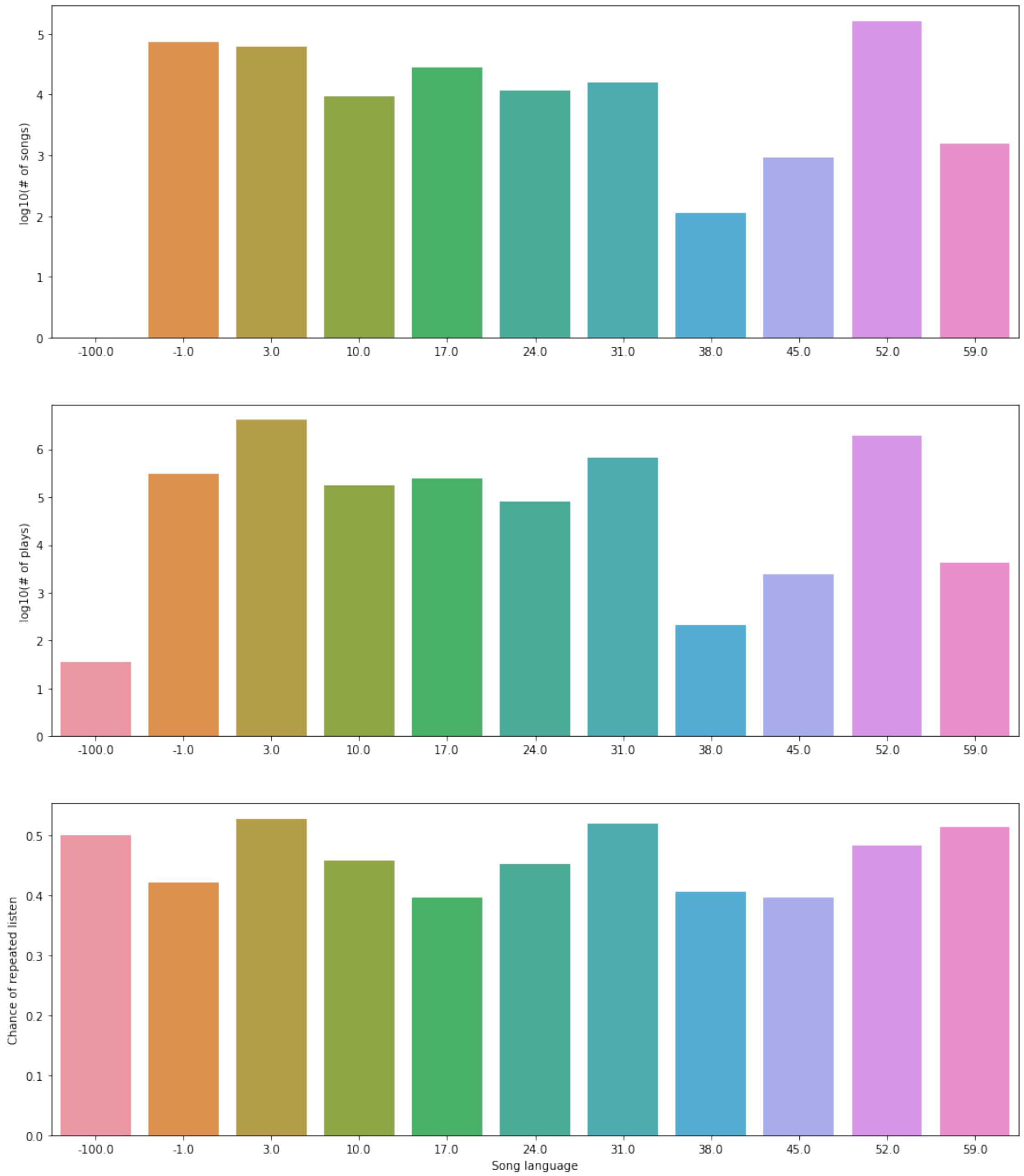
```
[45. 52. -1.  3. 10. 24. 31. 17. 59. 38. nan] 11
```

[16]:

```
fig = plt.figure(figsize=(15, 18))

ax1 = plt.subplot(3,1,1)
sns.barplot(x=languages,y=np.log10(language_count))
ax1.set_ylabel('log10(# of songs)')
ax2 = plt.subplot(3,1,2)
sns.barplot(x=languages,y=np.log10(language_plays))
ax2.set_ylabel('log10(# of plays)')
ax3 = plt.subplot(3,1,3)
sns.barplot(x=languages,y=language_repeat_chance)
ax3.set_ylabel('Chance of repeated listen')
ax3.set_xlabel('Song language')
```

[16]:

```
Text(0.5, 0, 'Song language')
```



0.4 Exploring the Length of the Songs

[17]:

```
min_song_length_sec = song_data['song_length'].min() / 1000  # the data is in msec
max_song_length_sec = song_data['song_length'].max() / 1000
print(min_song_length_sec, max_song_length_sec)
```

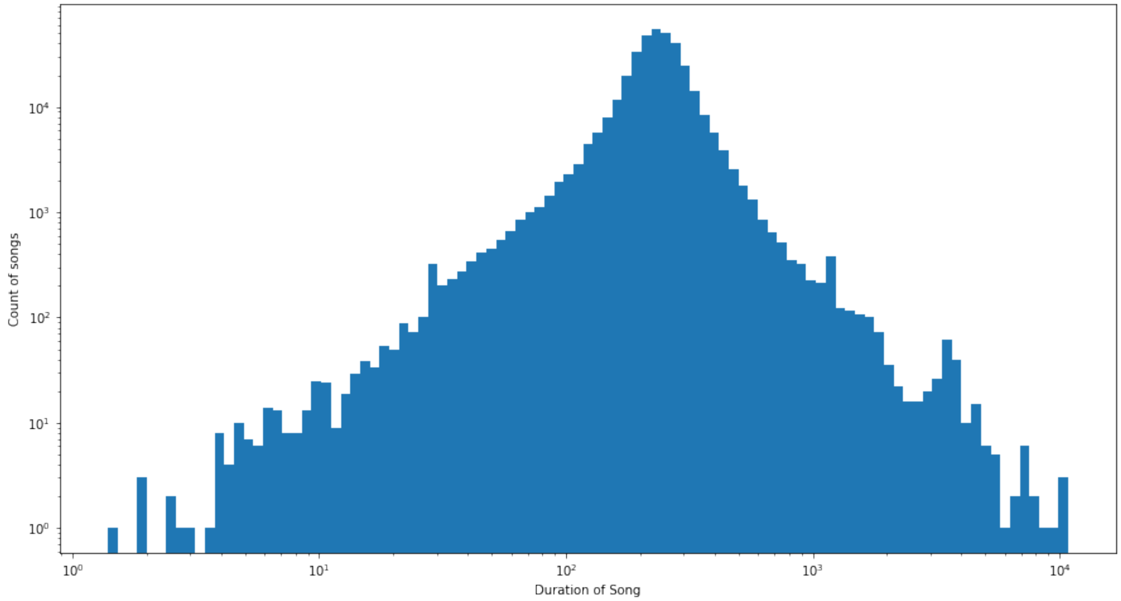
```
1.393 10851.706
```

[18]:

```
#min_length_song = song_data.iloc[song_data['song_length'].idxmin()]
#max_length_song = song_data.iloc[song_data['song_length'].idxmax()]
#print(min_length_song[['artist_name', 'composer', 'lyricist',␣
 ↪'number_of_composers','number_of_lyricists', 'song_length',␣
 ↪'repeat_play_chance']], '\n')
#print(max_length_song[['artist_name', 'composer', 'lyricist',␣
 ↪'number_of_composers','number_of_lyricists', 'song_length',␣
 ↪'repeat_play_chance']])
```

[19]:

```
plt.figure(figsize=(15,8))
length_bins = np.logspace(np.log10(min_song_length_sec),np.
 ↪log10(max_song_length_sec+1),100)
sns.distplot(song_data['song_length']/1000, bins=length_bins,␣
 ↪kde=False,hist_kws={"alpha": 1})
plt.xlabel('Duration of Song')
plt.ylabel('Count of songs')
plt.yscale('log')
plt.xscale('log')
```



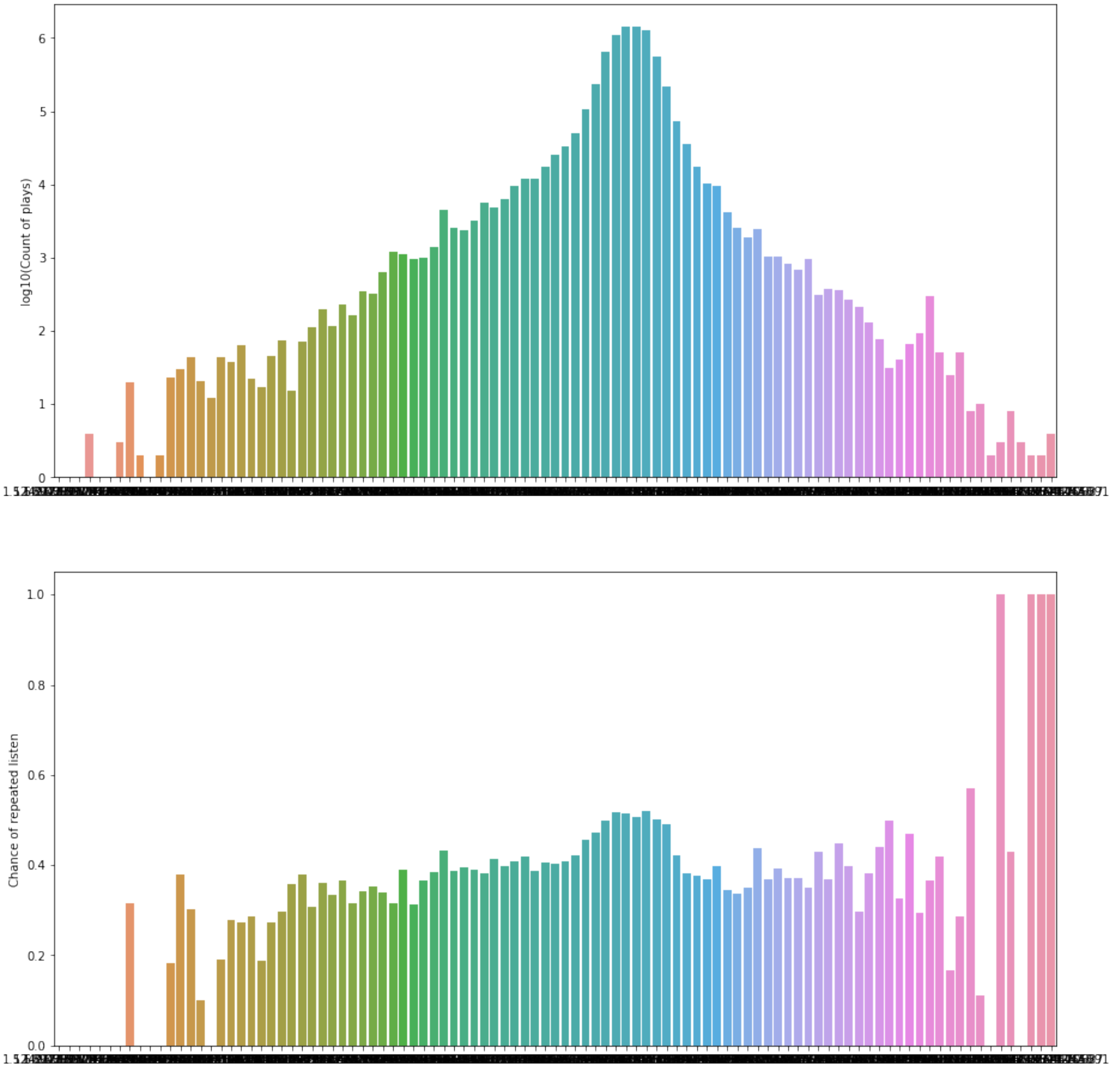
0.5 Relationship between Length of the Songs and Repeatability

[20]:

```
time_labels = list(range(length_bins.shape[0]-1))
song_data['time_cuts'] = pd.cut(song_data['song_length']/1000,␣
 ↪bins=length_bins, labels=time_labels)

y_repeat_chance_tc = []
y_plays_tc = []
```

```
y_rel_plays = []
for i in time_labels:
    timecut_i = song_data[song_data['time_cuts']==i]
    count = timecut_i['plays'].sum()
    y_plays_tc.append(count)
    if count != 0:
        y_repeat_chance_tc.append(timecut_i['repeat_events'].sum() / count)
        y_rel_plays.append(count / timecut_i.shape[0])
    else:
        y_repeat_chance_tc.append(0)
        y_rel_plays.append(0)

fig = plt.figure(figsize=(15, 16))

y_plays_tc = [yptc + 1 for yptc in y_plays_tc]

ax211 = plt.subplot(2,1,1)
sns.barplot(x=length_bins[time_labels],y=np.log10(y_plays_tc))
ax211.set_ylabel('log10(Count of plays)')

ax212 = plt.subplot(2,1,2)
sns.barplot(x=length_bins[time_labels],y=y_repeat_chance_tc)
ax212.set_ylabel('Chance of repeated listen')
```

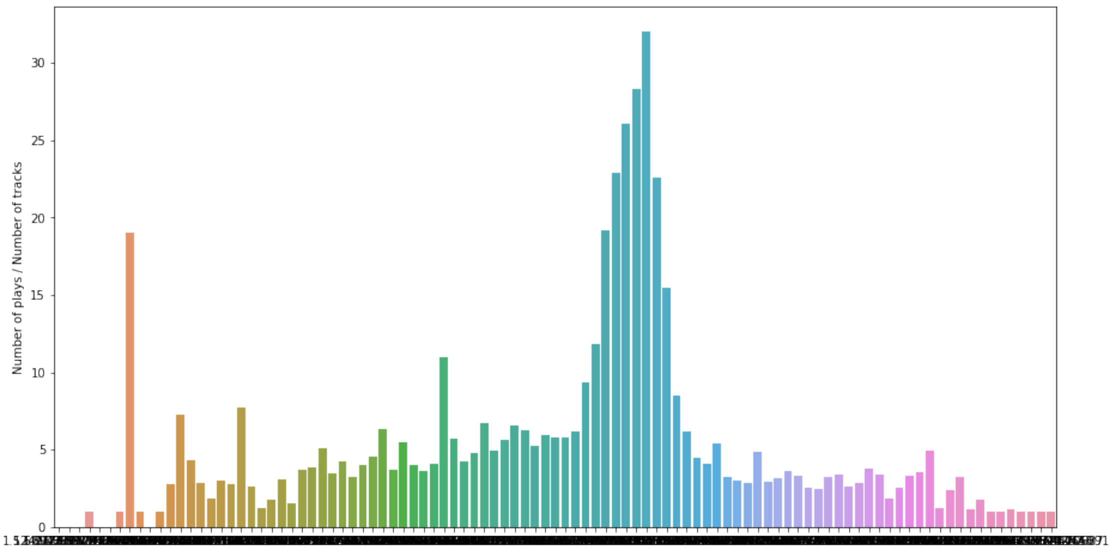
```
[20]: Text(0, 0.5, 'Chance of repeated listen')
```



0.6 Relationship between Number of Tracks and Number of Plays

[21]:
```
fig = plt.figure(figsize=(15, 8))

ax111 = plt.subplot(1,1,1)
sns.barplot(x=length_bins[time_labels],y=y_rel_plays)
ax111.set_ylabel('Number of plays / Number of tracks')
```

[21]:
```
Text(0, 0.5, 'Number of plays / Number of tracks')
```

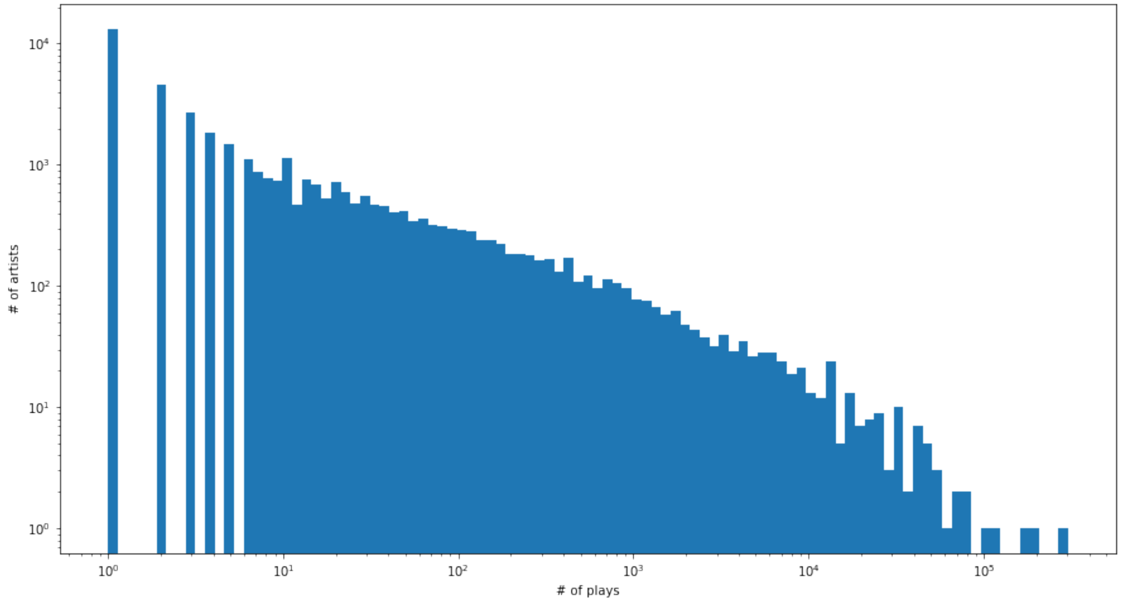


0.7 Relationship Between Number of Artists and Number of Plays

[22]:
```
artist_groupby = song_data[['artist_name', 'plays']].groupby(['artist_name'])
artist_plays = artist_groupby['plays'].agg(['sum'])
artist_plays.reset_index(inplace=True)

min_plays = artist_plays['sum'].min()
max_plays = artist_plays['sum'].max()
print(min_plays, max_plays)
```

```
1 303616
```

[23]:
```
plt.figure(figsize=(15,8))
play_bins = np.logspace(np.log10(min_plays),np.log10(max_plays+1),100)
# track_bins = np.linspace(1,max_tracks+1,100)
sns.distplot(artist_plays['sum'], bins=play_bins, kde=False,
             hist_kws={"alpha": 1})
plt.xlabel('# of plays')
plt.ylabel('# of artists')
plt.yscale('log')
plt.xscale('log')
```



0.8 Relationship Between Number of Artists and Chance of Repeatability

[24]:

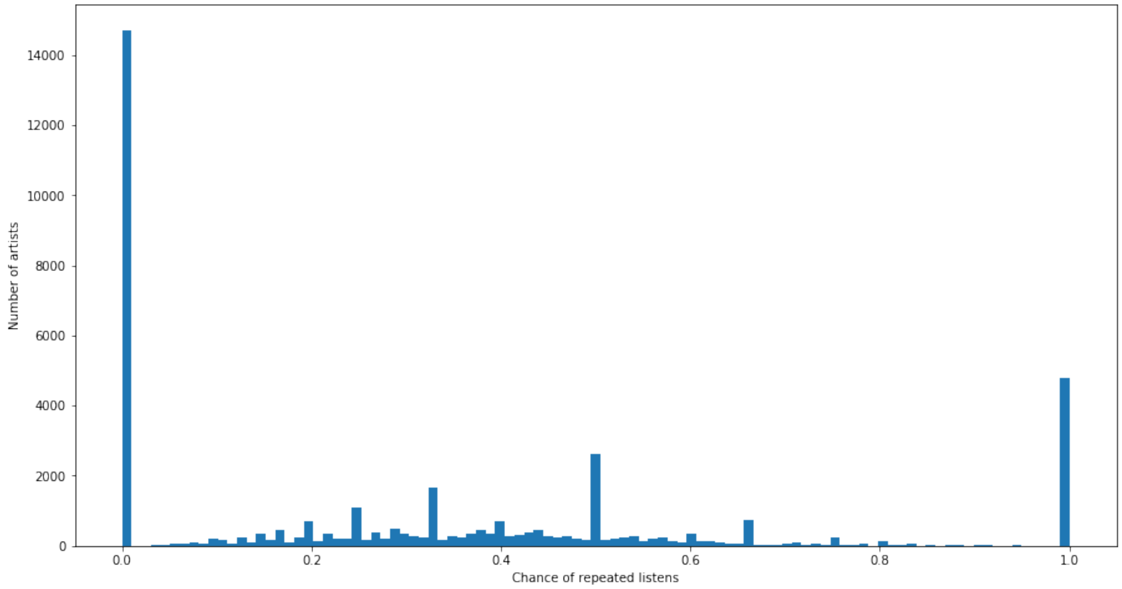
```python
artist_replgroupby = song_data[['artist_name', 'plays', 'repeat_events']].groupby(['artist_name'])
artist_replgroupby = artist_replgroupby['plays', 'repeat_events'].agg(['sum', 'count'])
artist_replgroupby.reset_index(inplace=True)
artist_replgroupby.columns = list(map(''.join, artist_replgroupby.columns.values))
artist_replgroupby.drop(['repeat_eventscount'], axis=1, inplace=True)
artist_replgroupby.columns = ['artist', 'plays', 'tracks', 'repeat_events']
artist_replgroupby['repeat_play_chance'] = artist_replgroupby['repeat_events'] / artist_replgroupby['plays']
```

[25]:

```python
plt.figure(figsize=(15,8))
chance_bins = np.linspace(0,1,100)
sns.distplot(artist_replgroupby['repeat_play_chance'], bins=chance_bins, kde=False,hist_kws={"alpha": 1})
plt.xlabel('Chance of repeated listens')
plt.ylabel('Number of artists')
#plt.yscale('log')
#plt.xscale('log')
```

[25]:

```
Text(0, 0.5, 'Number of artists')
```



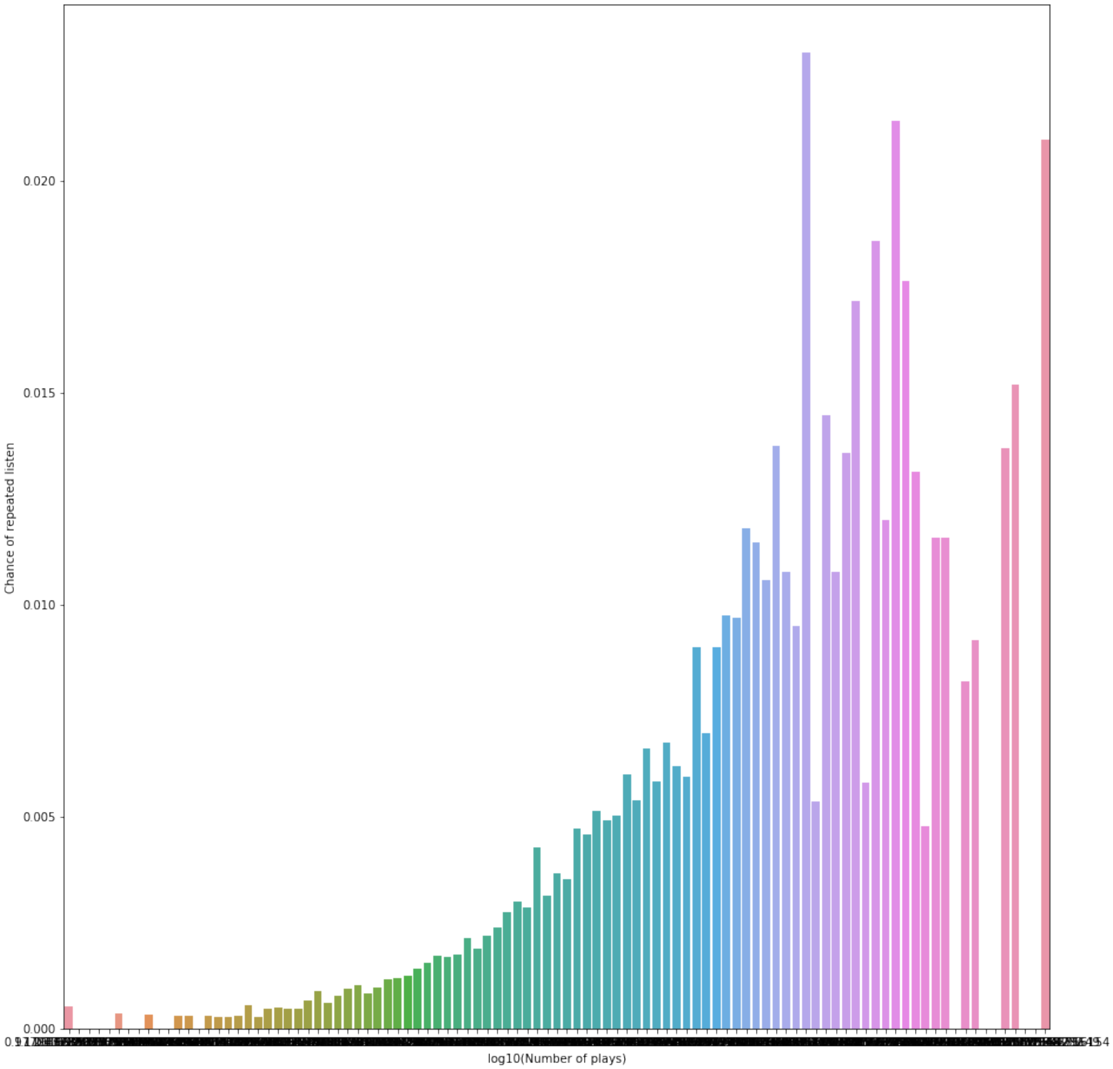
0.9 Relationship Between Number of Plays and Chance of Repeatibility

[26]:
```
artist_replgroupby['plays'].max()
```

[26]:
```
303616
```

[27]:
```
play_bins = np.logspace(-0.01, np.log10(artist_replgroupby['plays'].max()), 100)
play_labels = list(range(play_bins.shape[0]-1))
artist_replgroupby['play_cuts'] = pd.cut(artist_replgroupby['plays'],
                                         bins=play_bins, labels=play_labels)

y_repeat_chance_p = []
y_plays_p = []
for i in play_labels:
    playcut_i = artist_replgroupby[artist_replgroupby['play_cuts']==i]
    count = artist_replgroupby['plays'].sum()
    y_plays_p.append(count)
    if count != 0:
        y_repeat_chance_p.append(playcut_i['repeat_events'].sum() / count)
    else:
        y_repeat_chance_p.append(0)

fig = plt.figure(figsize=(15, 16))

ax111 = plt.subplot(1,1,1)
sns.barplot(x=play_bins[play_labels],y=y_repeat_chance_p)
ax111.set_xlabel('log10(Number of plays)')
ax111.set_ylabel('Chance of repeated listen')
```

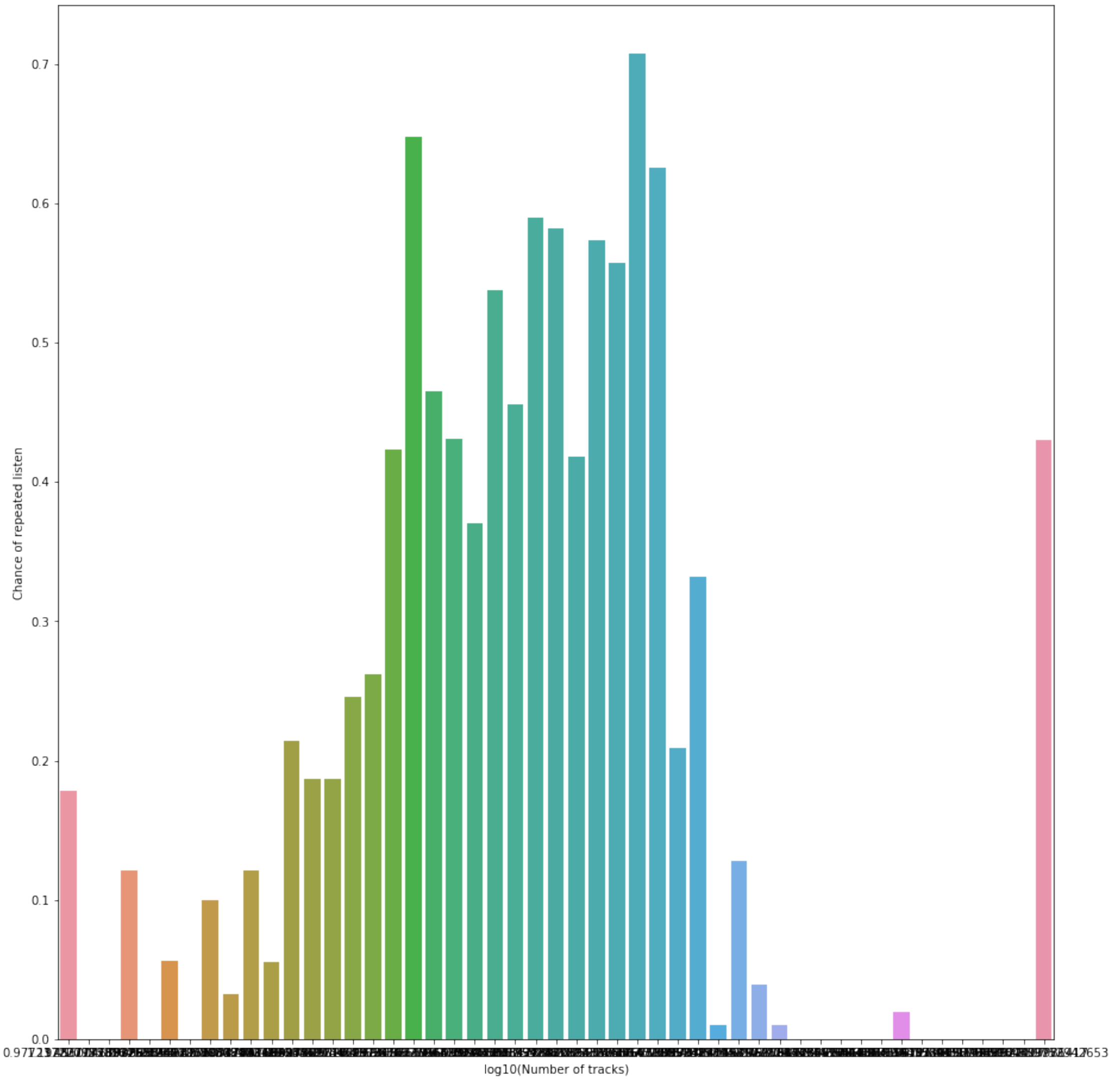
[27]:
```
Text(0, 0.5, 'Chance of repeated listen')
```



0.10 Relationship Between Number of Tracks and Chance of Repeatability

[28]:
```
track_bins = np.logspace(-0.01, np.log10(artist_replgroupby['tracks'].max()),␣
 ↪50)
track_labels = list(range(track_bins.shape[0]-1))
artist_replgroupby['track_cuts'] = pd.
 ↪cut(artist_replgroupby['tracks'],bins=track_bins, labels=track_labels)

y_repeat_chance_t = []
y_tracks_t = []
for i in track_labels:
    trackcut_i = artist_replgroupby[artist_replgroupby['track_cuts']==i]
    count = artist_replgroupby['tracks'].sum()
    y_tracks_t.append(count)
    if count != 0:
        y_repeat_chance_t.append(trackcut_i['repeat_events'].sum() / count)
    else:
        y_repeat_chance_t.append(0)

fig = plt.figure(figsize=(15, 16))

ax111 = plt.subplot(1,1,1)
sns.barplot(x=track_bins[track_labels],y=y_repeat_chance_t)
ax111.set_xlabel('log10(Number of tracks)')
ax111.set_ylabel('Chance of repeated listen')
```

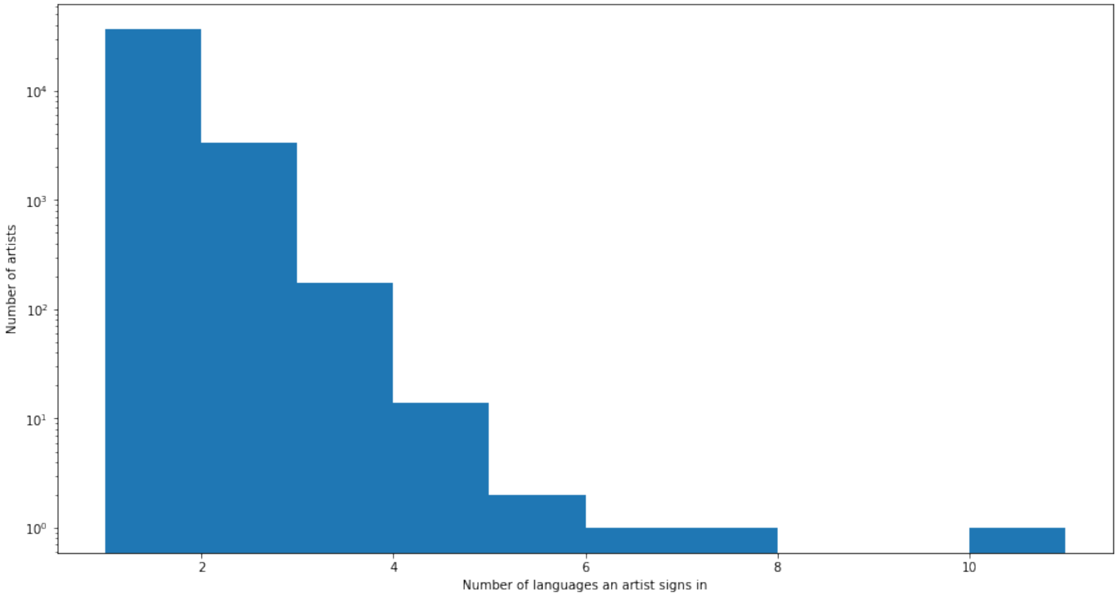
[28]:
```
Text(0, 0.5, 'Chance of repeated listen')
```



0.11 Relationship Between Number of Languages and Number of Artists

[29]:
```
artist_langgroupby = song_data[['artist_name',  'language']].
 ↪groupby(['artist_name'])
artist_langgroupby = artist_langgroupby.agg({"language": pd.Series.nunique})
artist_langgroupby.reset_index(inplace=True)
artist_langgroupby.columns = list(map(''.join, artist_langgroupby.columns.
 ↪values))
artist_langgroupby.columns = ['artist', 'language']

artist_repl_lang = artist_replgroupby.merge(artist_langgroupby, on='artist')
```

[30]:
```
plt.figure(figsize=(15,8))
chance_bins = np.linspace(1,artist_repl_lang['language'].max()+1,11)
sns.distplot(artist_repl_lang['language'], bins=chance_bins,␣
 ↪kde=False,hist_kws={"alpha": 1})
plt.xlabel('Number of languages an artist signs in')
plt.ylabel('Number of artists')
plt.yscale('log')
```



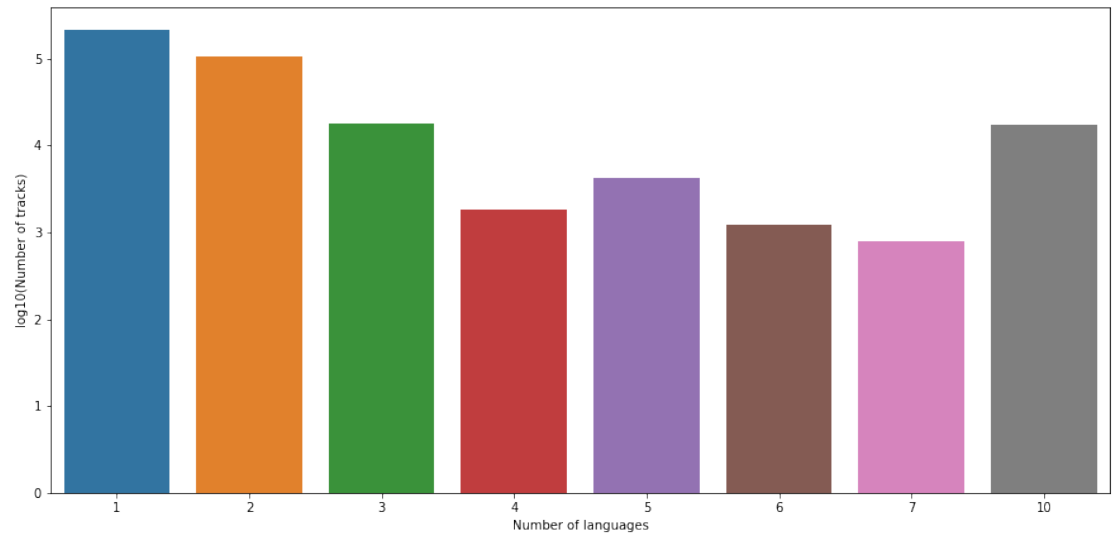
0.12 Relationship Between Number of Languages and Number of Tracks, Number of Plays and Change of Repeatability

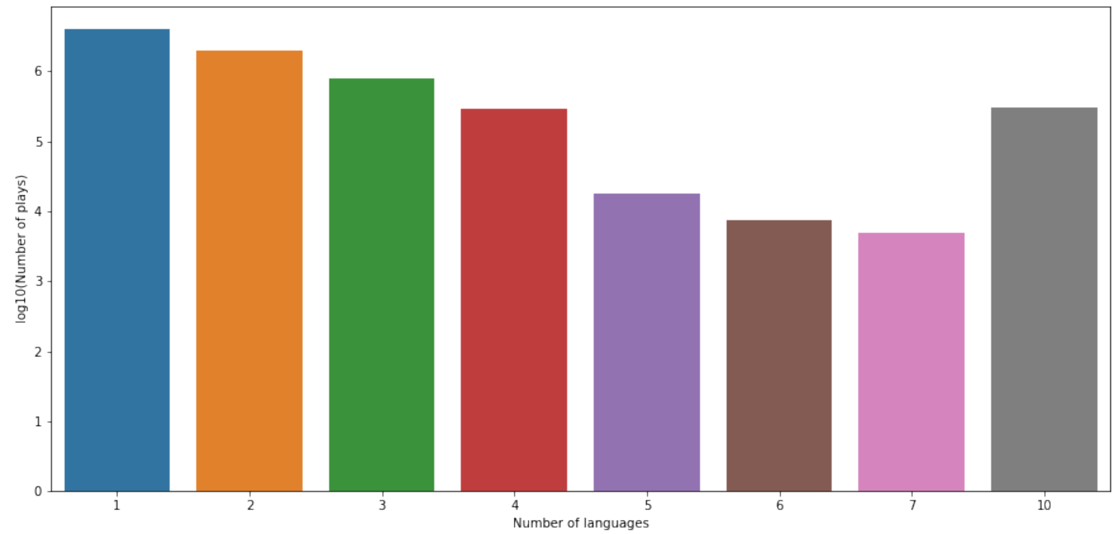
[31]:

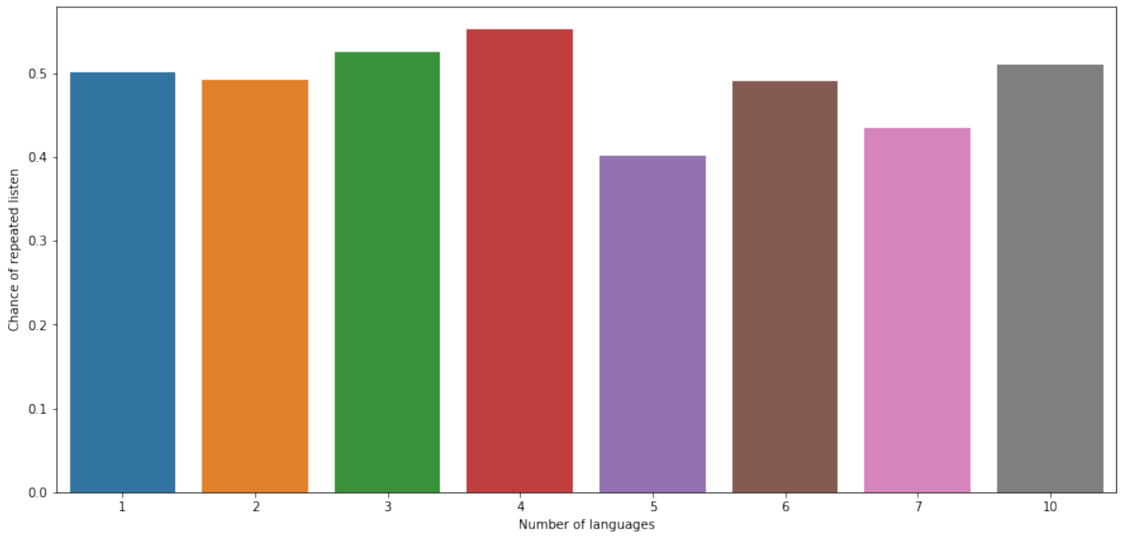
```
y_repeat_chance_l = []
y_plays_l = []
y_tracks_l = []

max_l = int(artist_repl_lang['language'].max())
l_list = []

for i in range(1,max_l+1):
    arlang = artist_repl_lang[artist_repl_lang['language']==i]
    count = arlang['plays'].sum()
    if count != 0:
        y_tracks_l.append(arlang['tracks'].sum())
        y_plays_l.append(count)
        l_list.append(i)
        y_repeat_chance_l.append(arlang['repeat_events'].sum() / count)

fig = plt.figure(figsize=(15, 24))

ax311 = plt.subplot(3,1,1)
sns.barplot(x=l_list,y=np.log10(y_tracks_l))
ax311.set_xlabel('Number of languages')
ax311.set_ylabel('log10(Number of tracks)')

ax312 = plt.subplot(3,1,2)
sns.barplot(x=l_list,y=np.log10(y_plays_l))
ax312.set_xlabel('Number of languages')
ax312.set_ylabel('log10(Number of plays)')

ax313 = plt.subplot(3,1,3)
sns.barplot(x=l_list,y=y_repeat_chance_l)
ax313.set_xlabel('Number of languages')
ax313.set_ylabel('Chance of repeated listen')
```

[31]:

```
Text(0, 0.5, 'Chance of repeated listen')
```


log10(Number of tracks)
0
1
2
3
4
5
1
2
3
4
5
6
7
10
Number of languages


log10(Number of plays)
0
1
2
3
4
5
6
1
2
3
4
5
6
7
10
Number of languages


Chance of repeated listen
0.0
0.1
0.2
0.3
0.4
0.5
1
2
3
4
5
6
7
10
Number of languages

0.13 Exploring the Genre Information in the DataSet

[32]:
```
def split_genres(x, n):
    # n is the number of the genre
    if type(x) != str:
        if n == 1:
            if not np.isnan(x):
                return int(x)
            else:
                return x
    else:
        if x.count('|') >= n-1:
            return int(x.split('|')[n-1])
```

[33]:
```
max_genres = song_data['number_of_genres'].max()

for i in range(1,max_genres+1):
    sp_g = lambda x: split_genres(x, i)
    song_data['genre_'+str(i)] = song_data['genre_ids'].apply(sp_g)

n_genres = set()

for i in range(1,max_genres+1):
    n_genres.update(song_data['genre_'+str(i)][song_data['genre_'+str(i)].
 ↪notnull()].unique().tolist())
```

[34]:
```
len(n_genres), song_data['genre_ids'].isnull().sum()
```

[34]:
```
(166, 7233)
```

[35]:
```
genres_plays = [0] * (len(n_genres) + 1)
genres_tracks = [0] * (len(n_genres) + 1)
genres_replays = [0] * (len(n_genres) + 1)

for i in range(1,max_genres+1):
    notnull_data = song_data[song_data['genre_'+str(i)].notnull()]
    for j, k in enumerate(n_genres):
        jk_sdata = notnull_data[notnull_data['genre_'+str(i)] == k]
        genres_plays[j] += jk_sdata['plays'].sum()
        genres_tracks[j] += jk_sdata['plays'].shape[0]
        genres_replays[j] += jk_sdata['repeat_events'].sum()

null_genre_data = song_data[song_data['genre_1'].isnull()]
genres_plays[len(n_genres)] = null_genre_data['plays'].sum()
genres_tracks[len(n_genres)] = null_genre_data['plays'].shape[0]
genres_replays[len(n_genres)] = null_genre_data['repeat_events'].sum()
```

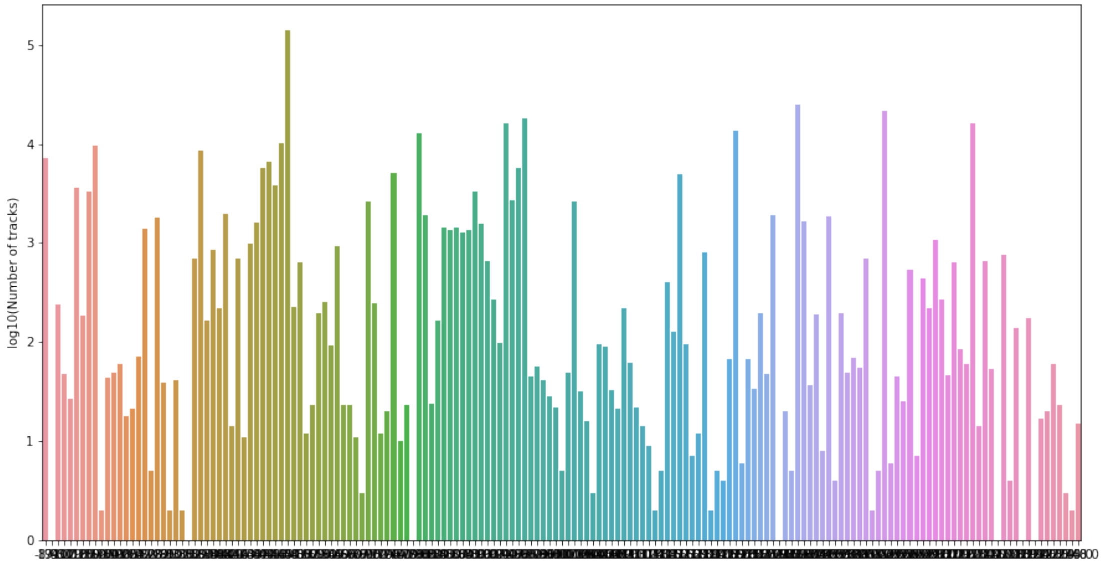
```
genres_rel_plays = [x/y for x, y in zip(genres_plays, genres_tracks)]
genres_repl_chance = [x/y for x, y in zip(genres_replays, genres_plays)]
```

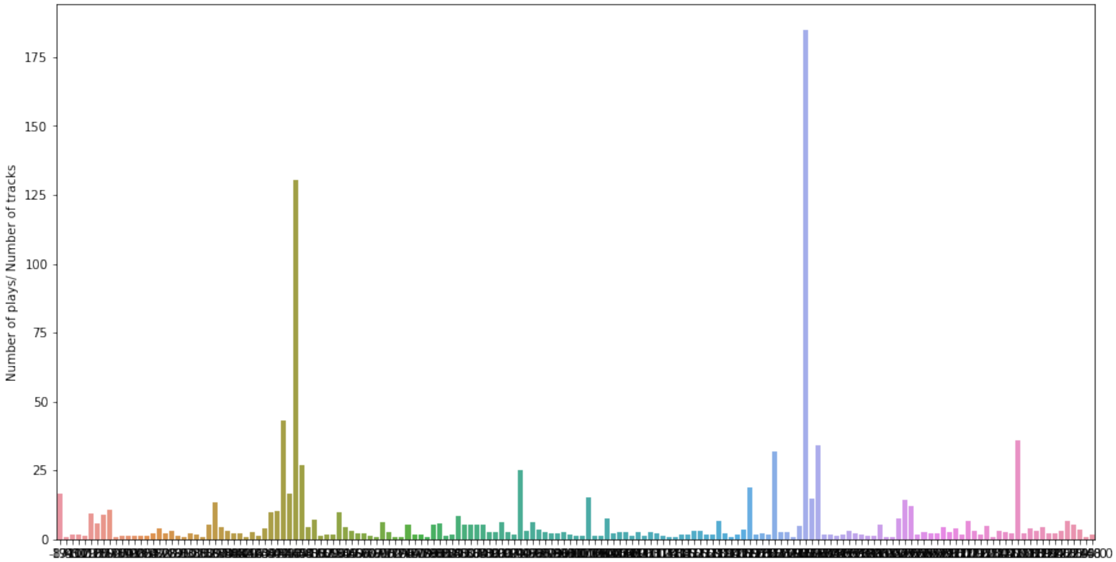
[36]:

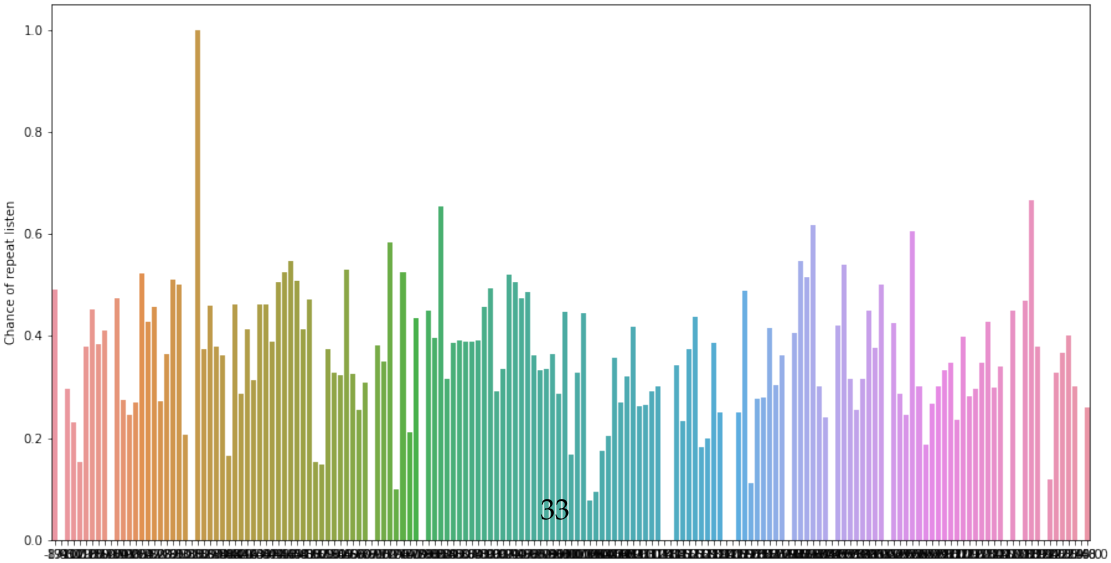
```
n_g_l = [x for x in n_genres]
n_g_l.append(-1)

fig = plt.figure(figsize=(15, 27))

ax411 = plt.subplot(3,1,1)
sns.barplot(x=n_g_l,y=np.log10(genres_tracks))
ax411.set_ylabel('log10(Number of tracks)')


ax413 = plt.subplot(3,1,2)
sns.barplot(x=n_g_l,y=genres_rel_plays)
ax413.set_ylabel('Number of plays/ Number of tracks')

ax414 = plt.subplot(3,1,3)
sns.barplot(x=n_g_l,y=genres_repl_chance)
ax414.set_ylabel('Chance of repeat listen')
```

[36]:

```
Text(0, 0.5, 'Chance of repeat listen')
```


log10(Number of tracks)
0
1
2
3
4
5


Number of plays/ Number of tracks
0
25
50
75
100
125
150
175

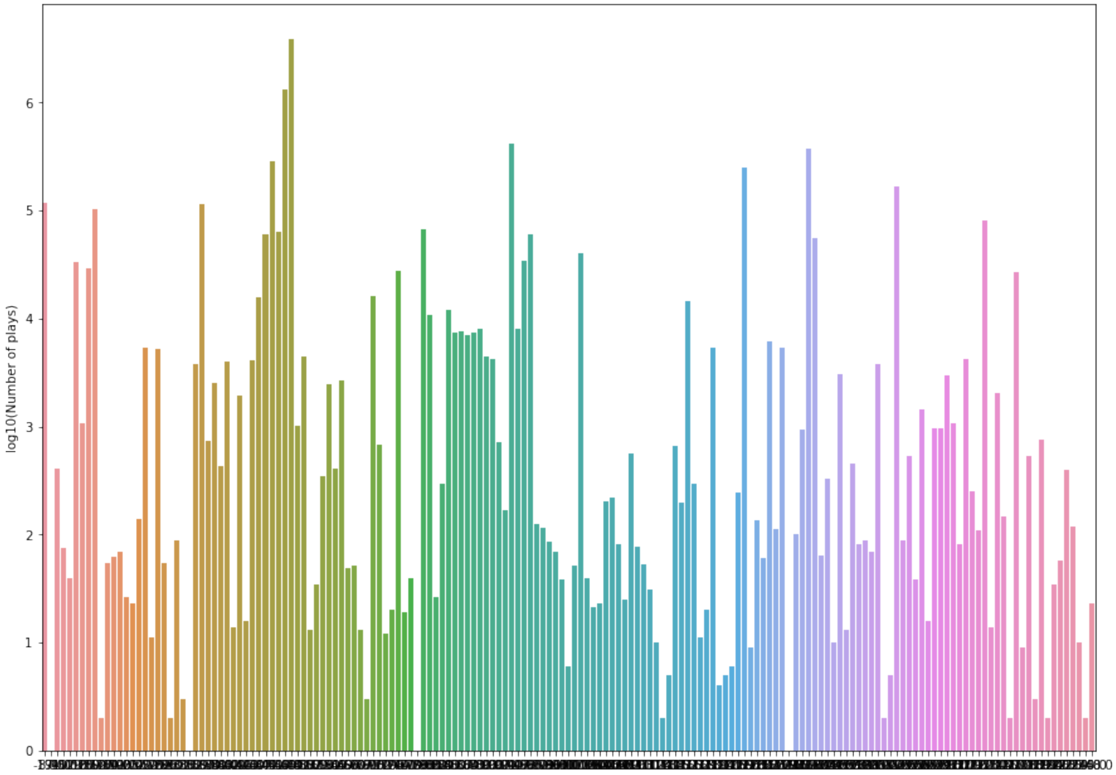

Chance of repeat listen
0.0
0.2
0.4
0.6
0.8
1.0

[37]:

```
fig = plt.figure(figsize=(15, 24))

ax412 = plt.subplot(2,1,1)
sns.barplot(x=n_g_l,y=np.log10(genres_plays))
ax412.set_ylabel('log10(Number of plays)')
```

[37]:

```
Text(0, 0.5, 'log10(Number of plays)')
```



0.14 Comparing Genre Information in Training and Testing Datasets

[38]:

```
test_merged['number_of_genres'] = test_merged['genre_ids'].apply(count_vals)
number_of_genres_test = test_merged['number_of_genres'].max()
print(number_of_genres_test)


for i in range(1,number_of_genres_test+1):
    sp_g = lambda x: split_genres(x, i)
    test_merged['genre_'+str(i)] = test_merged['genre_ids'].apply(sp_g)

n_genres_test = set()
```

```
for i in range(1,max_genres+1):
    n_genres_test.
 ↪update(test_merged['genre_'+str(i)][test_merged['genre_'+str(i)].notnull()].
 ↪unique().tolist())
print(len(n_genres_test))
```

```
8
162
```

[39]:

```
c = 0
in_test_not_in_train = []
for g in n_genres_test:
    if g not in n_genres:
        c += 1
        in_test_not_in_train.append(g)
print(c, in_test_not_in_train)
```

```
6 [1061.0, 2045.0, 1089.0, 166.0, 765.0, 303.0]
```

[40]:

```
song_genres_test = []
song_genres_artist = []
for g in in_test_not_in_train:
    tmp = 0
    for i in range(1,number_of_genres_test+1):
        tmp_filtered = test_merged[test_merged['genre_'+str(i)]==g]
        tmp += tmp_filtered.shape[0]
        for stt_artist in tmp_filtered['artist_name']:
            song_genres_artist.append(stt_artist)
    song_genres_test.append(tmp)
print(song_genres_test, sum(song_genres_test))
print(set(song_genres_artist))
```

```
[2, 2, 1, 10, 2, 1] 18
{'Sebastiano Serafini', 'Lea Salonga', 'Paul Simon', 'Fabrice Millischer',
'', 'Lovi', 'goldenage ()'}
```