1_DataExploration

September 21, 2019

1 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/training/source_data/train.csv')
songs = pd.read_csv('input/training/source_data/songs.csv')
test = pd.read_csv('input/training/source_data/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:  145262
Count of Unique Songs in Testing Set:  168114
Songs that dont appear in Training Set:  82287
Percentage:  0.4894714301010029
Users Statistics:
Count of Users in Training Set:  18933
Count of Users in Testing Set:  22185
Users that dont appear in Training Set:  6842
Percentage:  0.3084065810232139
Artists Statistics:
```

```
Count of Artists in Training Set:  19103
Count of Artists in Testing Set:  21954
Artists that dont appear in Training Set:  8663
Percentage:  0.39459779539036166
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:  412
Number of Genres Present in Testing Set:  457
Genres that dont appear in Traning Set:  85
Percentage:  0.18599562363238512
```

[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  ++7GdTgp8zbQLYOki7hVPEOHpu+KLZClsGrGiEuL2uI=            0.666667     12
1  ++9CllWTafshZc7T8X7cvNfUxgDe0WYrJ3TOen026j4=            1.000000      1
2  ++CfKs1t1wU1t0q0UxCdRqGoDpToqgMPmYytklaqo9o=            0.500000     10
3  ++CnoGMowrYqDI2eQM3aNJMIsxPNx1LD7u8ShTGwAQQ=            1.000000      1
4  ++EBTkZ77PSeSnVQ72CHesRb3907hLqwlRGEZzBNkhs=            0.000000      1

   song_length genre_ids       artist_name  \
0       267075       458    (Cindy Yen)
1       222632       423  Enrique Iglesias
2       224574       465            JUNIEL
3       214691       465      Kenny Rogers
4       214552      1259           C Losta

                                          composer           lyricist  \
0                                          Cindy           |
1  Enrique Iglesias| Ray El Ingeniero Casillas|...                NaN
2                      TWO FACE |  (Lee Sang Ho)   (Han Sung Ho)
3                                             NaN                NaN
```

```
4                                                    NaN                  NaN

   language  repeat_events
0       3.0            8.0
1      52.0            1.0
2      31.0            5.0
3      52.0            1.0
4      52.0            0.0
```

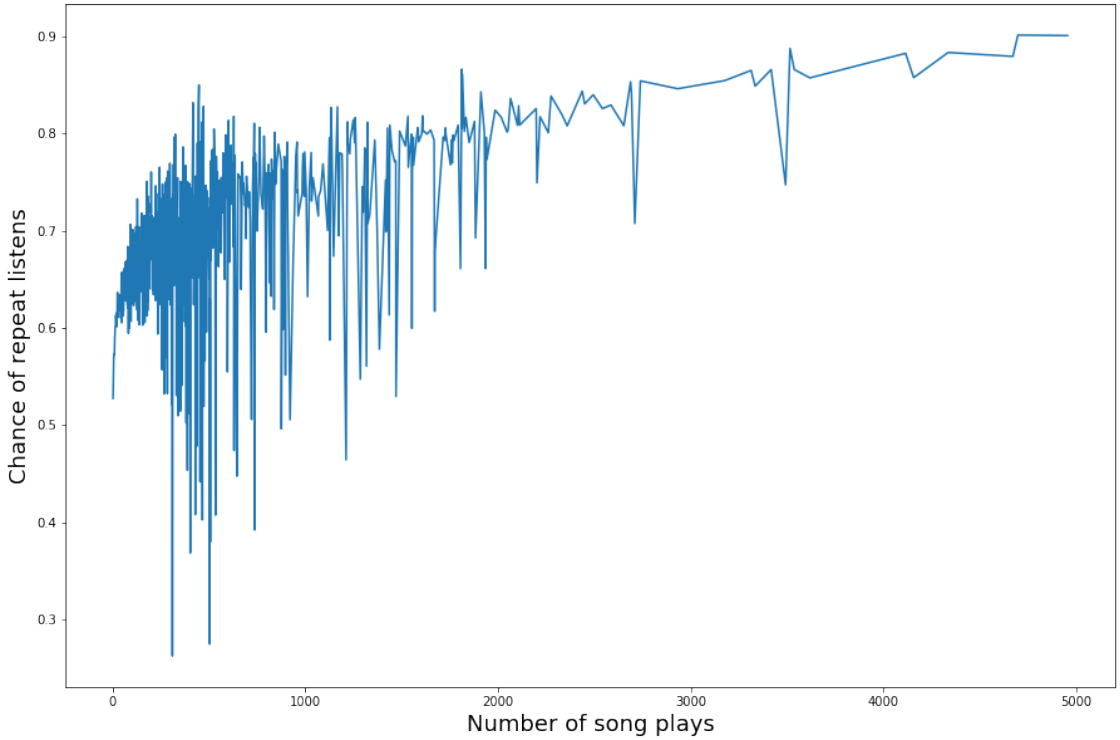
1.1 Relationship between Number of Plays and Repeatability

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

[6]:
```
4957
```

[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',fontsize=18)
plt.ylabel('Chance of repeat listens',fontsize=18)
plt.plot(number_of_plays, repeat_chance)
fig1 = plt.gcf()
fig1.savefig('ExplorationPlots/PlaysVsRepeatability.png', dpi=100,␣
 ↪bbox_inches='tight')
```



[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	++7GdTgp8zbQLYOki7hVPEOHpu+KLZClsGrGiEuL2uI=	0.666667	
1	++9CllWTafshZc7T8X7cvNfUxgDe0WYrJ3TOen026j4=	1.000000	
2	++CfKs1t1wU1t0q0UxCdRqGoDpToqgMPmYytklaqo9o=	0.500000	
3	++CnoGMowrYqDI2eQM3aNJMIsxPNx1LD7u8ShTGwAQQ=	1.000000	
4	++EBTkZ77PSeSnVQ72CHesRb3907hLqwlRGEZzBNkhs=	0.000000	
5	++EfwIEFB45OM9YDJlf1QMexyzs7kz2gbum80xJUAvw=	0.000000	
6	++HlV9O/nnF8fmlWOgsrJSxO+rx75nHWuE6O3ykpVS8=	1.000000	
7	++JnoK4BpUEfSsdS3ocjjffJIVvPrzlRqDn1qqdj1ZA=	1.000000	
8	++KIlFgmhgdnlqheXS+tSjDaTXtXMeO2qyztbFHOaRU=	1.000000	
9	++NP2gO99EW6w6sm4E1XP2Onx8FhbBxsK9fOvbPHwdO=	0.000000	
10	++OAO4xg27nDnqoaaxDZJK1sCG2VsesZTz+uR19iiKg=	1.000000	
11	++QYBUGIUXrSsYfBPDW4PN5V6wnlJzLYCwP+FS9gMnY=	0.500000	

12	++QfzyM/LiFaCuvkDFK/wJe13ZEMgTgAaVCcolo7nnY=	0.766497
13	++R6gkTXSRPJqjrIOmrl5GJJjcG/X5tEkOmNTz55Byg=	1.000000
14	++Sl6qP1lfLfC6c5aKEljKz3BcmIn6783TAzpd/vzaY=	1.000000
15	++UbBtvSfBUl2Um/2RiJNcvYnOR+fA1FaWWNracdOeI=	1.000000
16	++VOkgAg5i1aYtHoDzngmUUJFTUUCGrg4p+F5L1MEww=	1.000000
17	++VJq2+JeJByPWmj7Gf58ptVkIje+t9xeptpK3BARx4=	0.500000
18	++X36LmFngSdYDFPNeMi4DO8iYuVmZuBymO4pQUqPkg=	0.000000
19	++XsB7MZl/8xl/dO/QFmZ4TWIRDKhAQ5saiqmjznmi4=	0.000000
20	++YdDeXdEKHWLdWjDz6XoH1pS4021pZFA+aQxn1ivU4=	1.000000
21	++Zjtv16qG/odvO63kOCnu4SW5SGBnNYLFUCa/IZsPA=	0.333333
22	++cEaPOULLDm7ExxfT3B4HxihU1RMo6AexQOsNPqZLo=	0.000000
23	++chbCzz+7BYhLH5svRrj4AM37B1bTcC5O77wxODrVo=	0.750000
24	++dMcRr2gQsX2NO+kG7GqpHpAkOyZFY3GuBWzcsZLU4=	0.000000
25	++dP6kWVODms9Qky7m3TUPLzS6xLO+v2jYs9tkwKJHQ=	1.000000
26	++dfh2M5Vrc6WUb41WaJ6nj+EBOBaFzAzWW8gD8Ieak=	0.500000
27	++eOms2cYxLeFjxP/I5Gjl1M68g6/xMCK6/HY8WxBwg=	0.416667
28	++ezxSWRgPo7VimLbuChif2kuAZI5G6bmKuDdautYtw=	0.500000
29	++ggfo9NjqvKwCn6mDDDRz5rmRNVq/J4TEhtLcOP7jQ=	0.500000
...	...	...
145211	zzFIyaFsq3VFobOQlr2S/rxBe2vEO6AwGtcCrjGgYzQ=	0.500000
145212	zzO5/APo7dsGq8B90o00vFV7dS6GluNeIcsGCKCmzjc=	0.000000
145213	zzOuGsPPbTtL4yAr2rRhtlcxmyUfbIY4cC4lfsnyWbU=	0.000000
145214	zzQz2ZAmk4uhYQBWPHkDtdsmR7EU79sPGj+Tuvu//mo=	0.000000
145215	zzS51Dgte8PqJtSTJYhkcSp3tG6Nd20+DCtoyRvcBfE=	0.000000
145216	zzUGzKeFdZ2AGBNf5u37HP/obwtJSX049qqa53AF/Fg=	0.000000
145217	zzVH+lPqhEwgi7oPIyieoLLfx69R6AmWhDNqOAsot1c=	0.500000
145218	zzYKGLZb/5MT70wENhzz21mGhZv++s8izWfowQDcg8s=	1.000000
145219	zzYz/zNGrnGDxtjSE3So+2fckVHjo4TuHm3GQ2WMoow=	0.722222
145220	zzaVMc6q0P7Mh5xtcLuVpdfhr2yyFmM5BD/z1ME/TKE=	1.000000
145221	zzb/1Cbq02JEeNYseqSL4P8z38AOORVB7T+CQCZduuE=	0.000000
145222	zzbCo1jv8pvMwOb8RGlxHDN9CdQ3OVKaPEwhkbES9jY=	0.000000
145223	zzbGYYzMHO6WZFz+i94lfFJe1R5/+geFb0el7E4Vbvg=	0.000000
145224	zzczMi6KM2zWiFejIsJDDc9kYBfmSyDzIfwtUHPHvfg=	0.750000
145225	zzfOfT2K8/muudj+Ot23Ucq0Y+EeJp4Iv/qaToqE18E=	1.000000
145226	zzhKmqLpoOsaHH4fDOUn34qX6WRnkikEKy1p4Y5N5gc=	1.000000
145227	zzjKexepP2u/nB7Fx+ghAb+iMqepTVeno2ULP2hudao=	1.000000
145228	zzkr/ROrk66bqFO8yM4kLY5tzfScMlXn2eBuzqXaVOs=	0.000000
145229	zzl4HMzSbgXfWJkj40A6yBOO5iUZAlbbhzP35nIF3e8=	1.000000
145230	zzlSz6h5ejz71YBPBkXwbT+7a8jBocKfOnwLKlYn964=	1.000000
145231	zznuHa14iW1mMrCEQUMY2SgRxR9xiJ/gW2dKB+KeAsM=	1.000000
145232	zznxMVOcjB7aGaS+fEiuh2sd/SekUGP/iSh4LxBwD4M=	0.666667
145233	zzoE7+U/Ss/ulhUz8anOZFeUwrWXFnyRzXnYbkaK58U=	0.000000
145234	zzqFu2/dTaFzWZ7YdB7SNOaILlpbVoonyeULaLBW4mE=	0.800000
145235	zzszihw3XMMQTHFYM56VImzZMZke1RqZop3jHfIzkDA=	1.000000
145236	zztzrWbaiNpznnZFTyWfc56Xsyd1sXcRtih3kJ13OFs=	0.000000
145237	zzuRe+6ax33MGabaCk1ThVqCfXtTtm1ASvd92F4VgQY=	0.000000
145238	zzubyBL9pJJy9AZkG2ZYlVG+dQBmPvxVW8jitp4b8MA=	0.000000

```
145239  zzvfk6Np17ieMkvG9CQNxdDYQENCVLXuYx5VWrNhCvg=            1.000000
145240  zzwCHrZc0TezilVRRTbsiWY6ORNpBq1Zv8GGtHT2xlk=            0.000000

        plays  song_length        genre_ids  \
0          12       267075              458
1           1       222632              423
2          10       224574              465
3           1       214691              465
4           1       214552             1259
5           1       158066         1572|275
6           1       225047             1609
7           1       311588             1609
8           1       170527          786|947
9           1       161541          726|242
10          1       291224              465
11          2       498253              873
12        985       215016              444
13          1       212140              465
14          2       302088              437
15          1       246883              465
16          1       303229              465
17          2       290528              437
18          1       256522  864|857|850|843
19          1       452092              947
20          1       240001              465
21          3       282017              465
22          3       197903              465
23          4       272439              465
24          1       280079      139|125|109
25          1       254583             1609
26          2       210337              465
27         12       247222              465
28          2       205682              NaN
29          2       252540              139
...       ...          ...              ...
145211      2       327471              786
145212      1        44512              465
145213      1       228755              465
145214      1       413884              NaN
145215      1        85774              958
145216      1       442386             2079
145217      8       284421              465
145218      1       231247             2122
145219     18       222261              458
145220      1       193282              921
145221      2       206564              451
145222      1       195143              465
```

145223	1	279980	2122
145224	4	268329	465
145225	2	247710	465
145226	1	205728	1259
145227	1	204382	465|798
145228	1	174811	921
145229	1	219585	359
145230	1	233048	465
145231	1	241023	2022
145232	9	274808	2022
145233	1	237725	465
145234	5	289320	465
145235	1	587859	1609
145236	1	242755	139
145237	1	223242	444
145238	1	187884	786|947
145239	1	199505	1259
145240	2	144096	458

	artist_name \
0	(Cindy Yen)
1	Enrique Iglesias
2	JUNIEL
3	Kenny Rogers
4	C Losta
5	Basement
6	Various Artists
7	Tropical House
8	Crystal Romance
9	
10	Unsuku Ke ()
11	Tasha Cobbs
12	TWICE
13	
14	Kana Nishino ()
15	
16	Various Artists
17	INFINITE
18	ZAQ
19	C'est La Vie
20	Workout Remix Factory
21	()
22	B.A.D.
23	(Cyndi Chaw)
24	Omarion
25	United Cube
26	

```
27                                  S.H.E
28                   HENRY (Super Junior-M)
29                              Ray Charles
...                                     ...
145211
145212
145213                         Itsuki Hiroshi
145214                        Various Artists
145215                         Arcadi Volodos
145216  Progressive PsyTrance (Edition 2010)
145217                                   CS
145218               Merry Christmas in Jazz
145219                             (Show Lo)
145220                        Various Artists
145221                          (Faye Wong)
145222                         (LilAshes)
145223                       Le Jardin Secret
145224                        (Ye Qi Tian)
145225                                    2PM
145226                       Young Paperboyz
145227
145228                       Various Artists
145229                                 Kaleo
145230                       Jahméne Douglas
145231                       Various Artists
145232                           Joonil Jung
145233                           Anne Murray
145234                          Mariah Carey
145235           M.A.N.D.Y. vs. Adultnapper
145236                                Jaheim
145237                                   IU
145238
145239                     Soulja Boy Tellem
145240                   (Monkey Pilot)

                                                composer  \
0                                               Cindy
1       Enrique Iglesias| Ray El Ingeniero Casillas|...
2                           TWO FACE |  (Lee Sang Ho)
3                                                  NaN
4                                                  NaN
5                                                  NaN
6                               L. Shipstad| Z. Mahmoud
7                                                  NaN
8                                                  NaN
9                                         JOE HISAISHI
10                                        Keisuke Hama
```

```
11                                        William Reagan
12                                      Black Eyed Win 1
13
14                                      Yuichi Hayashida
15
16
17                                      Ryuichi Kawamura
18                                                  ZAQ
19                                             Honey B
20                                                 NaN
21
22                                                 NaN
23
24                                                 NaN
25                              Sang Hyuk Lim| Jae-Woo Seo
26                                                  /
27
28                                                 NaN
29                                           Don Gibson
...                                                ...
145211                                  Hoagy Carmichael
145212                                   Kim Chanu
145213
145214                                             NaN
145215                                             NaN
145216                                             NaN
145217
145218                                             NaN
145219  Claire Rodrigues| Fridolin Walcher| Christoph ...
145220                                             NaN
145221                                       Ming Huang
145222                                       Lil Ashes
145223                                             NaN
145224                                             NaN
145225                                             NaN
145226                                             NaN
145227
145228                                             NaN
145229                                             NaN
145230  David Guetta| Sia Furler| Giorgio H. Tuinfort|...
145231                                             NaN
145232                                     Joonil Jung
145233                                   Randy Goodrum
145234                                             NaN
145235                                             NaN
145236                                             NaN
145237                                             NaN
```

```
145238                                                NaN
145239                                 D. Way| B. Green
145240

                                                lyricist  language  \
0                                                       |       3.0
1                                                     NaN      52.0
2                                          (Han Sung Ho)      31.0
3                                                     NaN      52.0
4                                                     NaN      52.0
5                                                     NaN      52.0
6                                                     NaN      52.0
7                                                     NaN      52.0
8                                                     NaN      -1.0
9                                         JOE HISAISHI         -1.0
10                                              Yu Aku         17.0
11                                                    NaN      52.0
12                                                    NaN      31.0
13                                                           3.0
14                                        Kana Nishino         17.0
15                                                  10.0
16                                                           3.0
17                                                    NaN      17.0
18                                                    ZAQ      17.0
19                                                    NaN      -1.0
20                                                    NaN      52.0
21                                                           3.0
22                                                    NaN       3.0
23                                                           3.0
24                                                    NaN      52.0
25                                                    NaN      31.0
26                                                           3.0
27                                                           3.0
28                                                    NaN       3.0
29                                                    NaN      52.0
...                                                   ...       ...
145211                                                NaN      -1.0
145212                                                NaN      31.0
145213                                                      17.0
145214                                                NaN      52.0
145215                                                NaN      -1.0
145216                                                NaN      52.0
145217                                                       3.0
145218                                                NaN      52.0
145219                                                NaN       3.0
145220                                                NaN      52.0
145221                                                NaN       3.0
```

```
145222                                                    24.0
145223                                                     NaN      52.0
145224                                                     NaN      10.0
145225                                                     NaN      17.0
145226                                                     NaN      52.0
145227                                                    3.0
145228                                                     NaN      52.0
145229                                                     NaN      52.0
145230  David Guetta| Sia Furler| Giorgio H. Tuinfort|...      52.0
145231                                                     NaN      52.0
145232                                           Joonil Jung      31.0
145233                                                     NaN      52.0
145234                                                     NaN      52.0
145235                                                     NaN      -1.0
145236                                                     NaN      -1.0
145237                                                     NaN      31.0
145238                                                     NaN      -1.0
145239                                                     NaN      52.0
145240                                                     NaN       3.0

        repeat_events  number_of_genres  number_of_composers  \
0                 8.0                 1                    1
1                 1.0                 1                    4
2                 5.0                 1                    2
3                 1.0                 1                    1
4                 0.0                 1                    1
5                 0.0                 2                    1
6                 1.0                 1                    2
7                 1.0                 1                    1
8                 1.0                 2                    1
9                 0.0                 2                    1
10                1.0                 1                    1
11                1.0                 1                    1
12              755.0                 1                    1
13                1.0                 1                    1
14                2.0                 1                    1
15                1.0                 1                    1
16                1.0                 1                    1
17                1.0                 1                    1
18                0.0                 4                    1
19                0.0                 1                    1
20                1.0                 1                    1
21                1.0                 1                    1
22                0.0                 1                    1
23                3.0                 1                    1
24                0.0                 3                    1
25                1.0                 1                    2
```

26	1.0	1	1
27	5.0	1	1
28	1.0	1	1
29	1.0	1	1
...	...	...	...
145211	1.0	1	1
145212	0.0	1	1
145213	0.0	1	1
145214	0.0	1	1
145215	0.0	1	1
145216	0.0	1	1
145217	4.0	1	1
145218	1.0	1	1
145219	13.0	1	4
145220	1.0	1	1
145221	0.0	1	1
145222	0.0	1	1
145223	0.0	1	1
145224	3.0	1	1
145225	2.0	1	1
145226	1.0	1	1
145227	1.0	2	1
145228	0.0	1	1
145229	1.0	1	1
145230	1.0	1	4
145231	1.0	1	1
145232	6.0	1	1
145233	0.0	1	1
145234	4.0	1	1
145235	1.0	1	1
145236	0.0	1	1
145237	0.0	1	1
145238	0.0	2	1
145239	1.0	1	2
145240	0.0	1	1

	number_of_lyricists
0	2
1	1
2	1
3	1
4	1
5	1
6	1
7	1
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	1
25	1
26	1
27	1
28	1
29	1
...	...
145211	1
145212	1
145213	1
145214	1
145215	1
145216	1
145217	1
145218	1
145219	1
145220	1
145221	1
145222	1
145223	1
145224	1
145225	1
145226	1
145227	1
145228	1
145229	1
145230	4
145231	1
145232	1
145233	1
145234	1
145235	1
145236	1

```
145237                    1
145238                    1
145239                    1
145240                    1

[145241 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 23 23
```

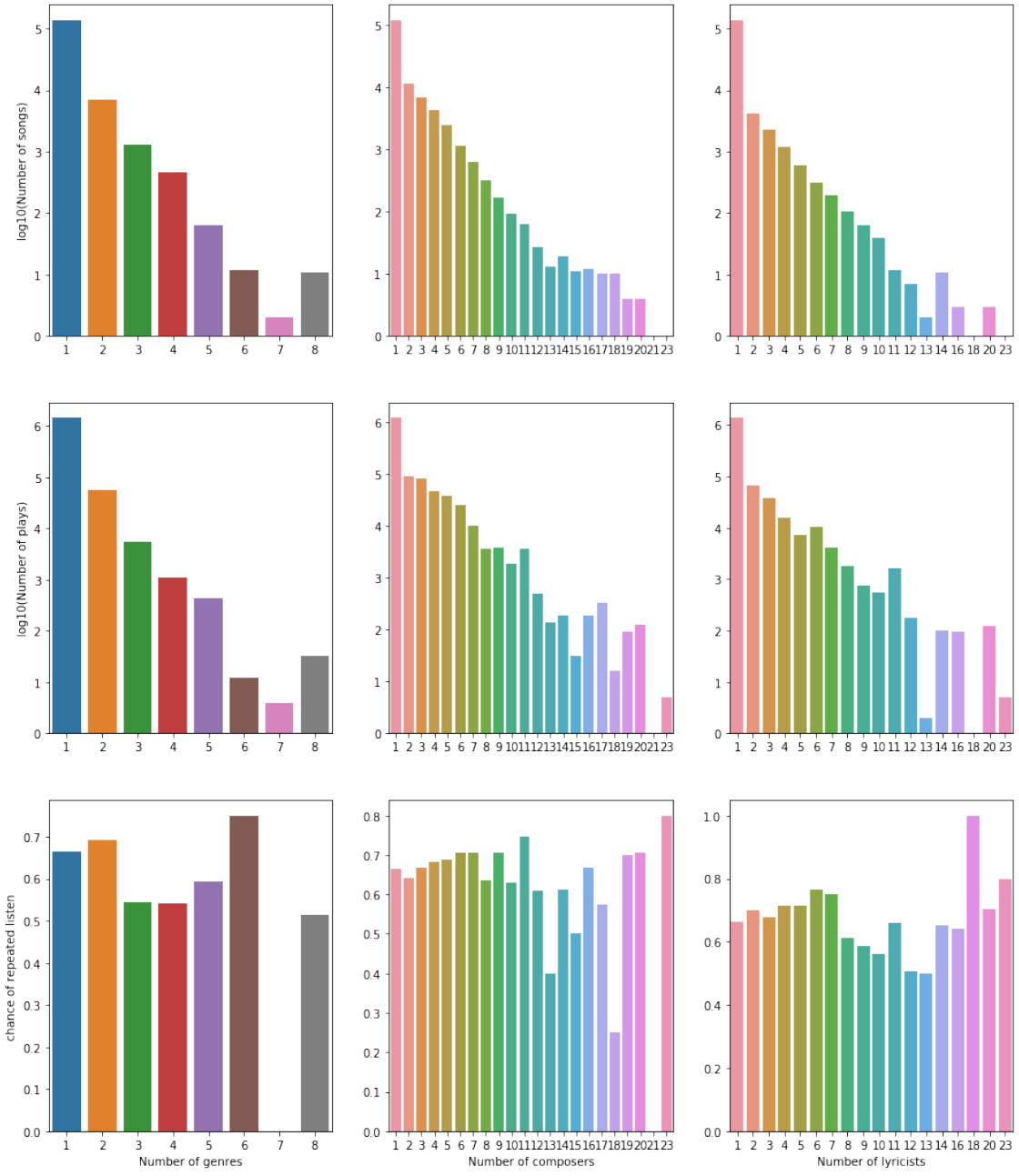
1.2 Relationship between number of Genre, Composer and Lyricist to the Chance of Repeating a Song

[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')
fig1 = plt.gcf()
fig1.savefig('ExplorationPlots/Genre,Composer,LyricistVsRepeatability.png',␣
 ↪dpi=100, bbox_inches='tight')
```

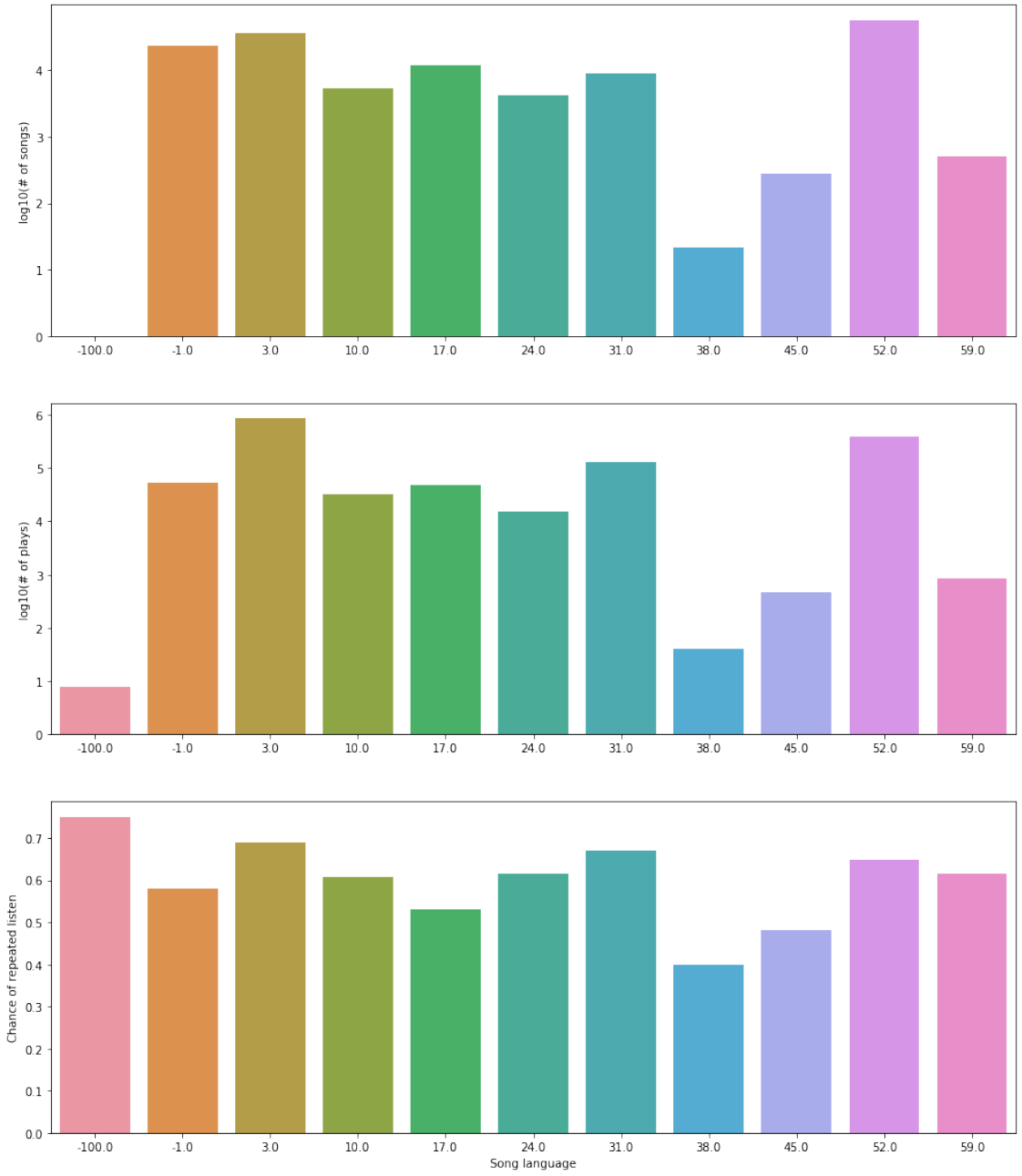


1.3 Relationship between Language and Number of Songs, Number of Plays and Chance of Repeat

[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
```

```
[ 3. 52. 31. -1. 17. 10. 24. 59. 45. 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')
fig1 = plt.gcf()
fig1.savefig('ExplorationPlots/Language,Songs,PlaysVsRepeatability.png',␣
 ↪dpi=100, bbox_inches='tight')
```



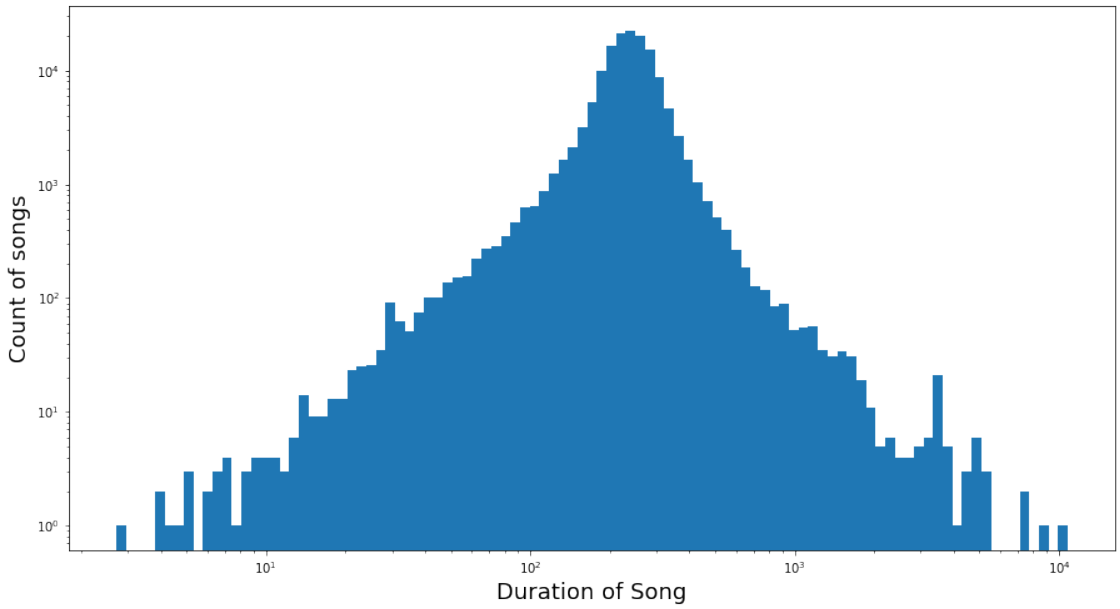
1.4 Exploring the Length of the Songs

[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)
```

```
2.716 10800.065
```

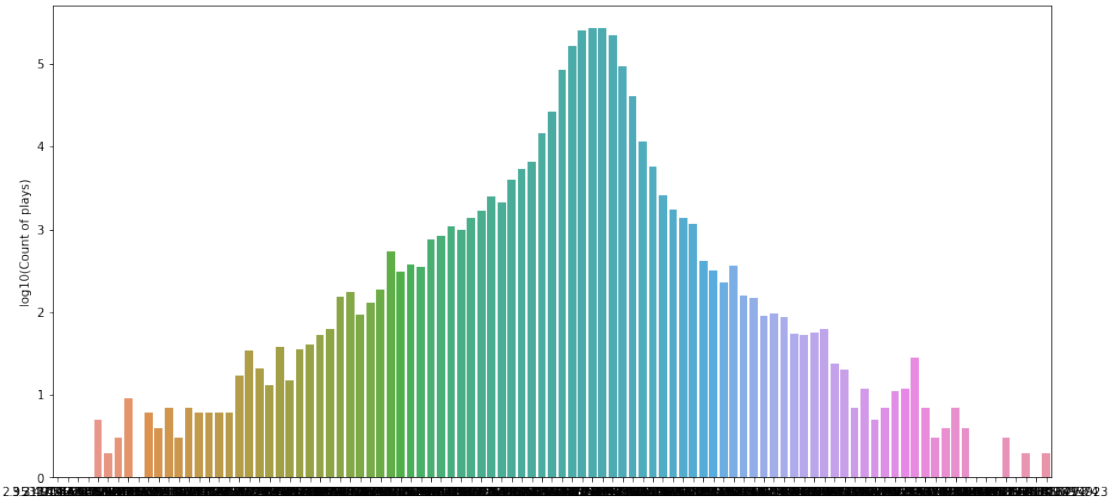
[18]:
```
#min_length_song = song_data.iloc[song_data['song_length'].idxmin()]
#max_length_song = song_data.iloc[song_data['song_length'].idxmax()]
#print(min_length_song[['artist_name', 'composer', 'lyricist',␣
 ↪'number_of_composers','number_of_lyricists', 'song_length',␣
 ↪'repeat_play_chance']], '\n')
#print(max_length_song[['artist_name', 'composer', 'lyricist',␣
 ↪'number_of_composers','number_of_lyricists', 'song_length',␣
 ↪'repeat_play_chance']])
```

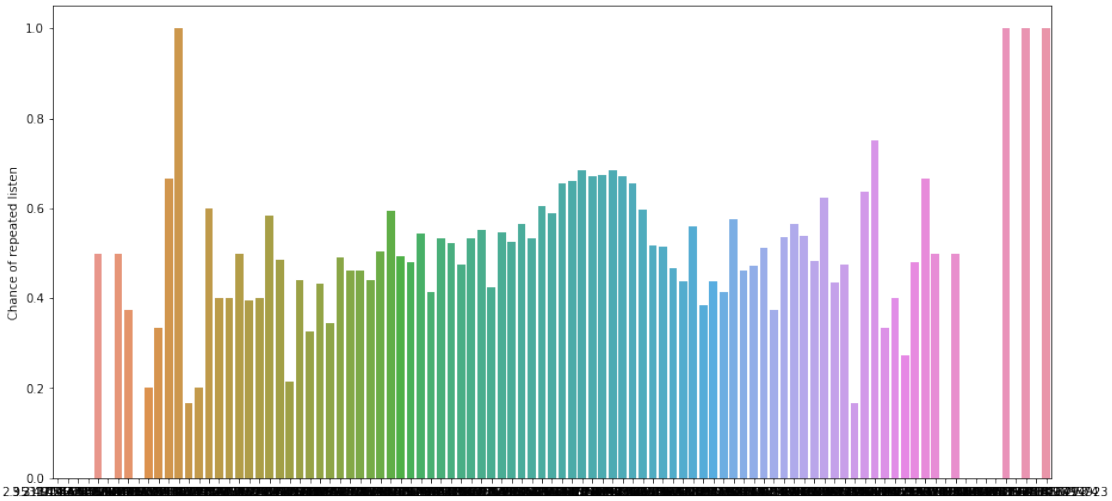
[19]:
```
plt.figure(figsize=(15,8))
length_bins = np.logspace(np.log10(min_song_length_sec),np.
 ↪log10(max_song_length_sec+1),100)
sns.distplot(song_data['song_length']/1000, bins=length_bins,␣
 ↪kde=False,hist_kws={"alpha": 1})
plt.xlabel('Duration of Song',fontsize=18)
plt.ylabel('Count of songs',fontsize=18)
plt.yscale('log')
plt.xscale('log')
fig1 = plt.gcf()
fig1.savefig('ExplorationPlots/CountVsDuration.png', dpi=100,␣
 ↪bbox_inches='tight')
```



1.5 Relationship between Length of the Songs and Repeatability

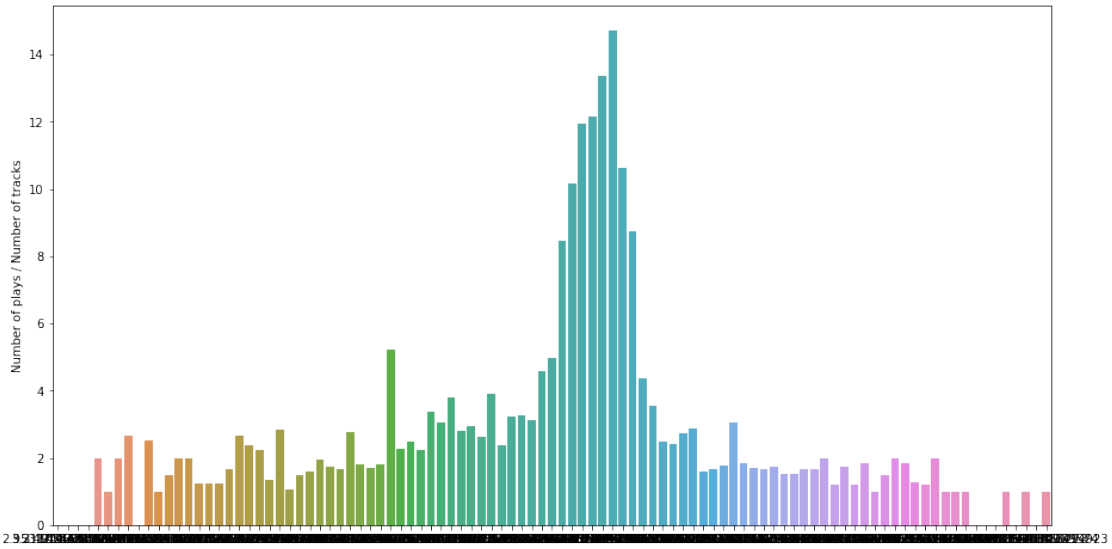
[20]:
```
time_labels = list(range(length_bins.shape[0]-1))
song_data['time_cuts'] = pd.cut(song_data['song_length']/1000,␣
 ↪bins=length_bins, labels=time_labels)

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')
fig1 = plt.gcf()
fig1.savefig('ExplorationPlots/CountVsRepeatability.png', dpi=100,␣
 ↪bbox_inches='tight')
```





1.6 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')
fig1 = plt.gcf()
fig1.savefig('ExplorationPlots/CountVsPlays.png', dpi=100, bbox_inches='tight')
```

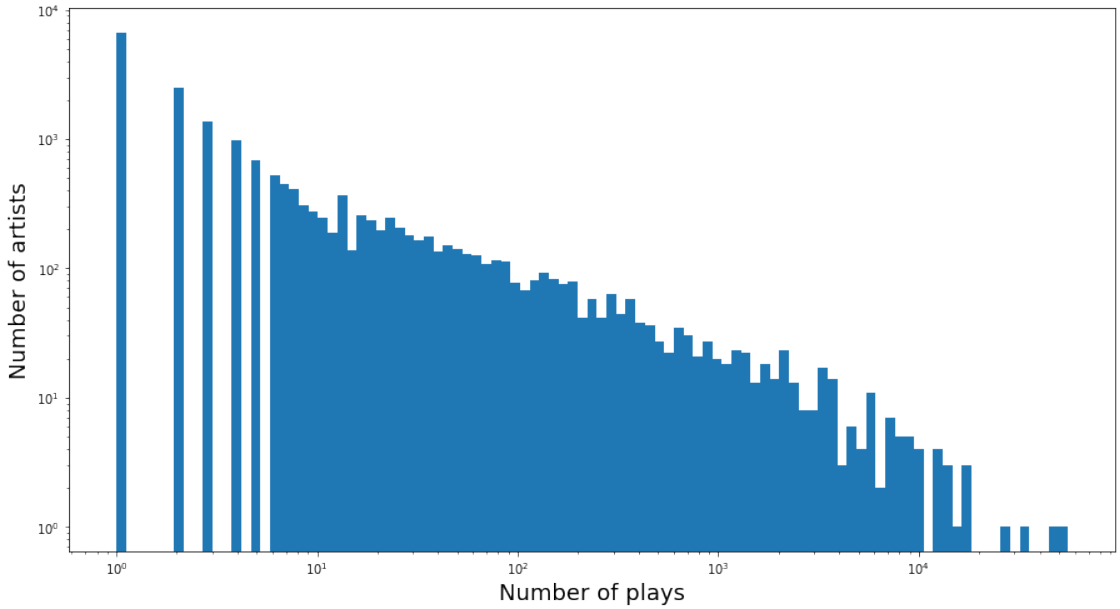


1.7 Relationship Between Number of Artists and Number of Plays

[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 55233
```

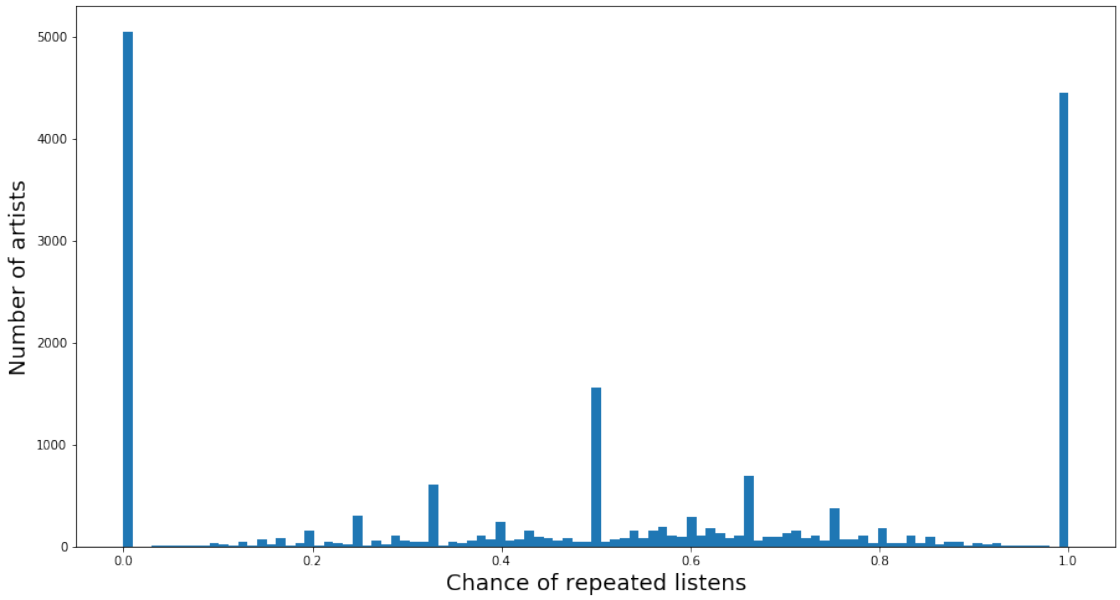
[23]:
```
plt.figure(figsize=(15,8))
play_bins = np.logspace(np.log10(min_plays),np.log10(max_plays+1),100)
# track_bins = np.linspace(1,max_tracks+1,100)
sns.distplot(artist_plays['sum'], bins=play_bins, kde=False,
             hist_kws={"alpha": 1})
plt.xlabel('Number of plays',fontsize=18)
plt.ylabel('Number of artists',fontsize=18)
plt.yscale('log')
plt.xscale('log')
fig1 = plt.gcf()
fig1.savefig('ExplorationPlots/ArtistsVsPlays.png', dpi=100,␣
 ↪bbox_inches='tight')
```



1.8 Relationship Between Number of Artists and Chance of Repeatability

[24]:
```
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]:
```
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',fontsize=18)
plt.ylabel('Number of artists',fontsize=18)
#plt.yscale('log')
#plt.xscale('log')
fig1 = plt.gcf()
fig1.savefig('ExplorationPlots/ArtistsVsRepeatability.png', dpi=100,␣
 ↪bbox_inches='tight')
```



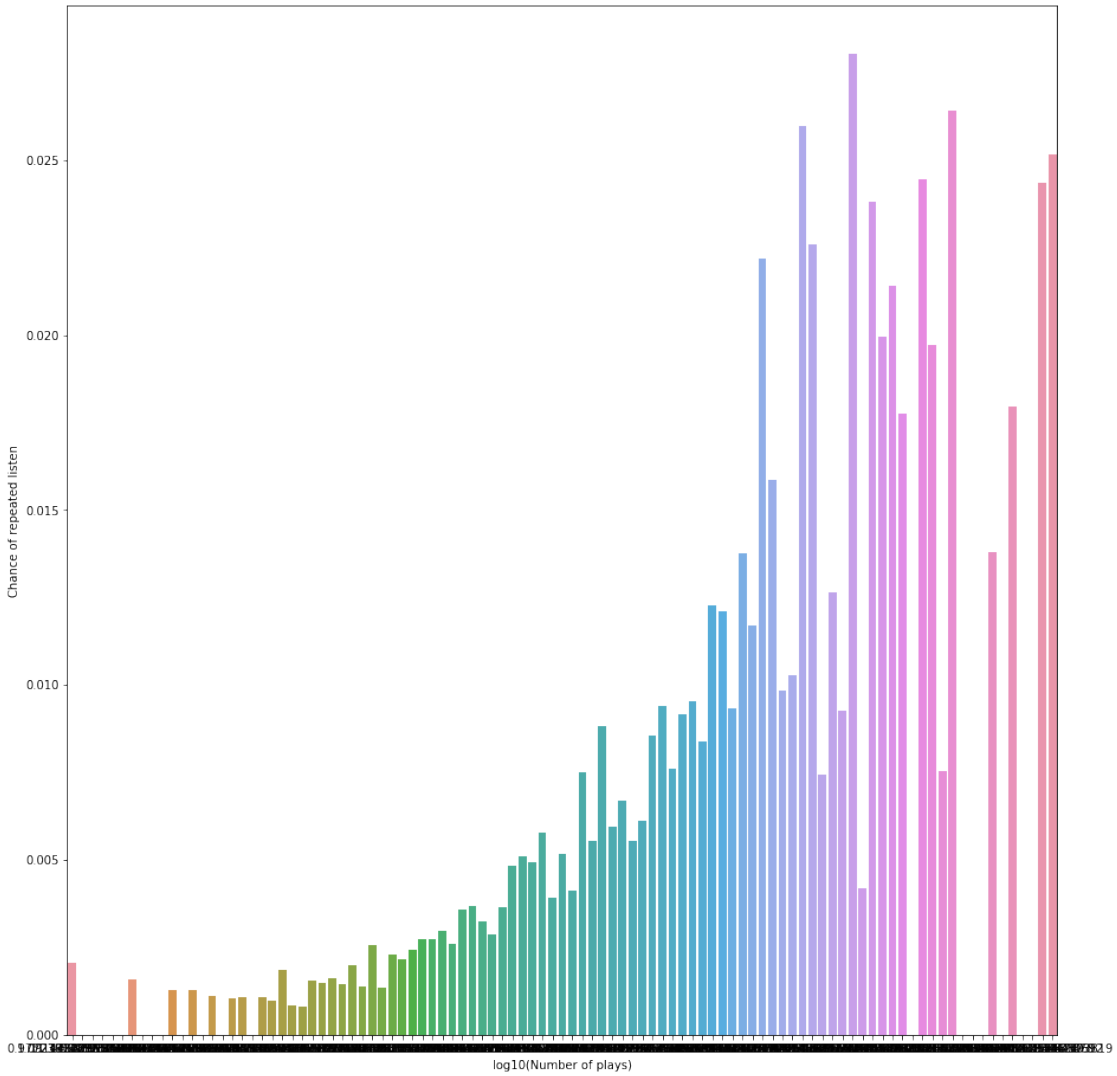
1.9 Relationship Between Number of Plays and Chance of Repeatibility

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

[26]: 55233

[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')
```

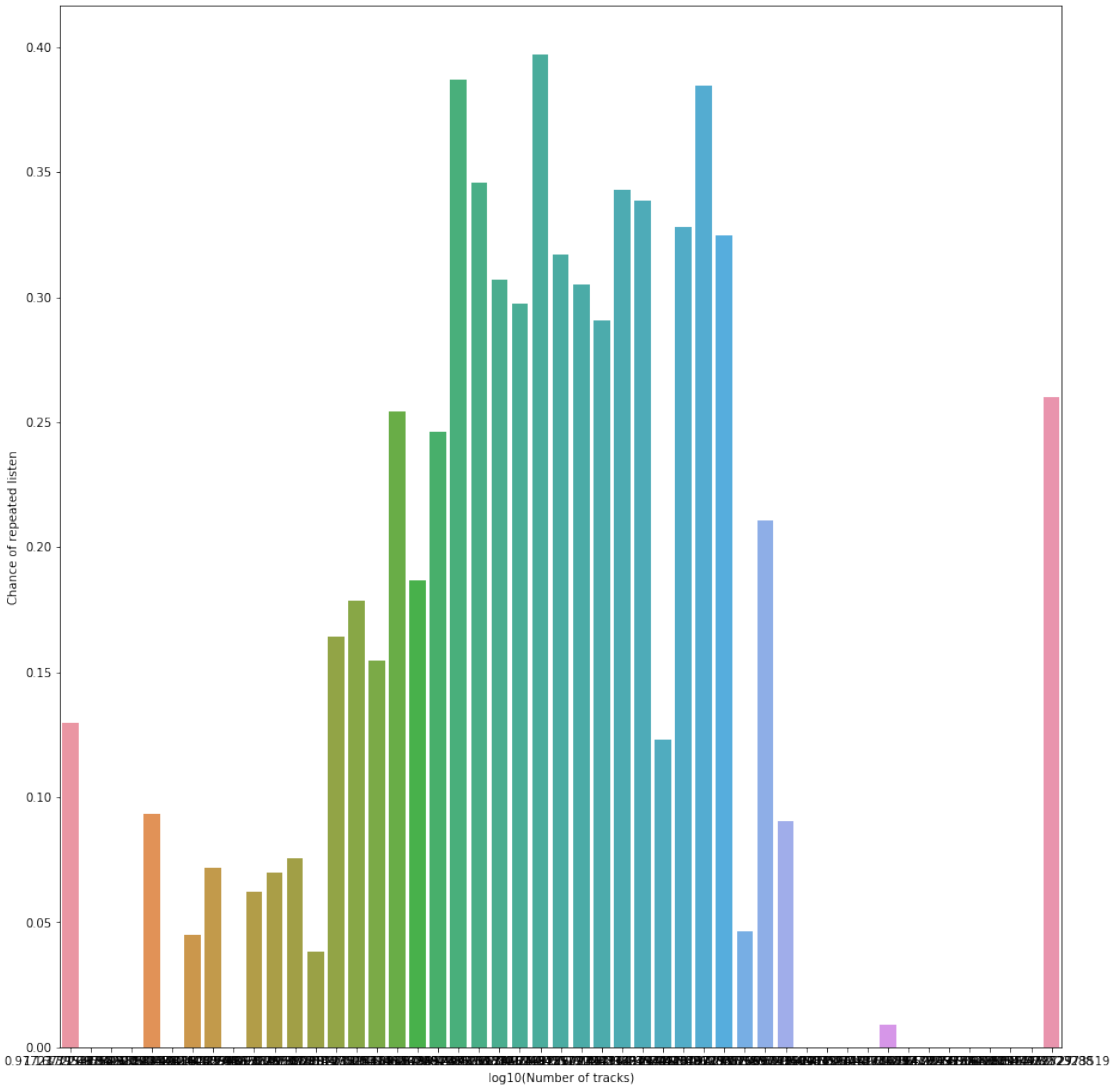
```
fig1 = plt.gcf()
fig1.savefig('ExplorationPlots/RepeatabilityVsPlays.png', dpi=100,␣
 ↪bbox_inches='tight')
```



1.10 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 = []
```

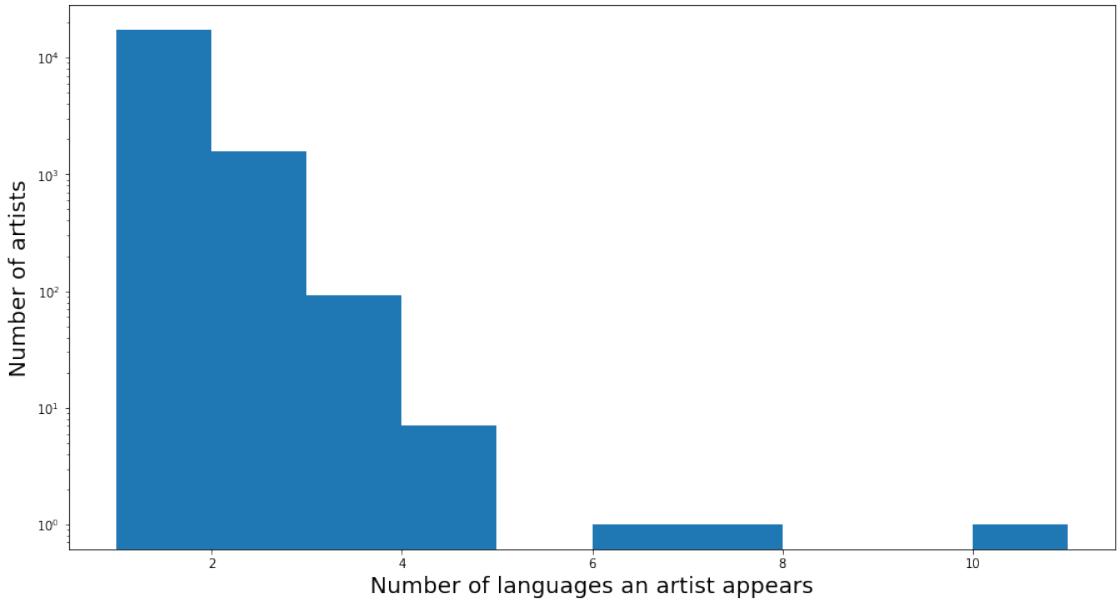
```
y_tracks_t = []
for i in track_labels:
    trackcut_i = artist_replgroupby[artist_replgroupby['track_cuts']==i]
    count = artist_replgroupby['tracks'].sum()
    y_tracks_t.append(count)
    if count != 0:
        y_repeat_chance_t.append(trackcut_i['repeat_events'].sum() / count)
    else:
        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')
fig1 = plt.gcf()
fig1.savefig('ExplorationPlots/RepeatabilityVsPlays.png', dpi=100,␣
 ↪bbox_inches='tight')
```



1.11 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')
```

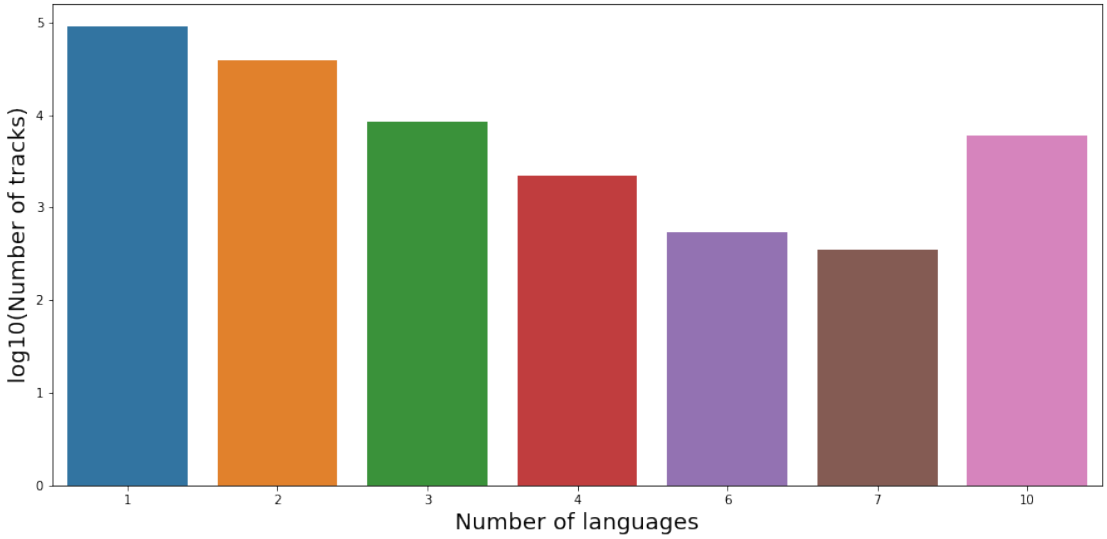
[30]:
```
plt.figure(figsize=(15,8))
chance_bins = np.linspace(1,artist_repl_lang['language'].max()+1,11)
sns.distplot(artist_repl_lang['language'], bins=chance_bins,␣
 ↪kde=False,hist_kws={"alpha": 1})
plt.xlabel('Number of languages an artist appears',fontsize=18)
plt.ylabel('Number of artists',fontsize=18)
plt.yscale('log')
fig1 = plt.gcf()
fig1.savefig('ExplorationPlots/ArtistsVsLanguages.png', dpi=100,␣
 ↪bbox_inches='tight')
```

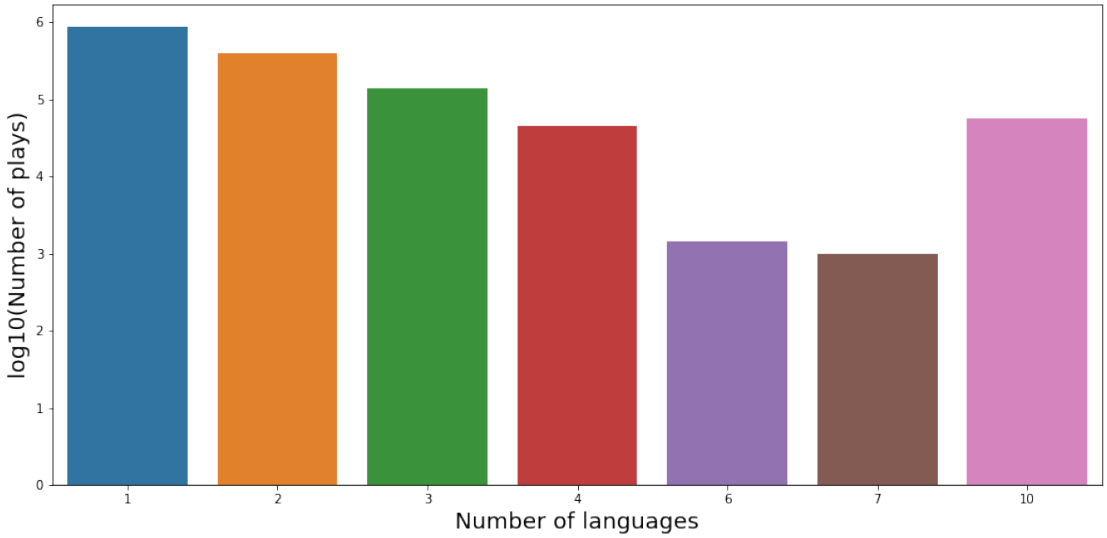


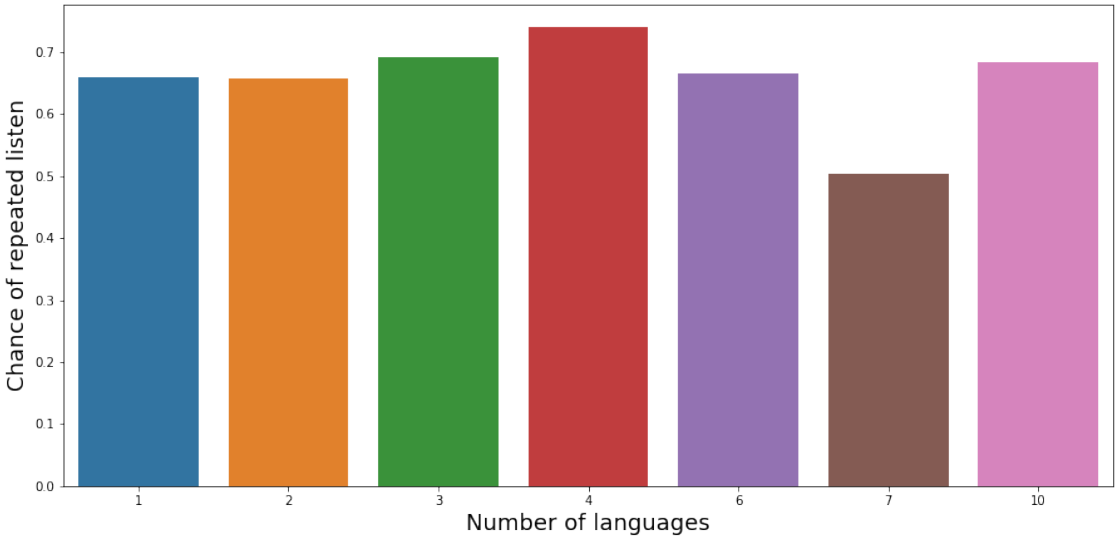
1.12 Relationship Between Number of Languages and Number of Tracks, Number of Plays and Change of Repeatability

[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',fontsize=18)
ax311.set_ylabel('log10(Number of tracks)',fontsize=18)

ax312 = plt.subplot(3,1,2)
sns.barplot(x=l_list,y=np.log10(y_plays_l))
ax312.set_xlabel('Number of languages',fontsize=18)
ax312.set_ylabel('log10(Number of plays)',fontsize=18)

ax313 = plt.subplot(3,1,3)
sns.barplot(x=l_list,y=y_repeat_chance_l)
ax313.set_xlabel('Number of languages',fontsize=18)
ax313.set_ylabel('Chance of repeated listen',fontsize=18)
fig1 = plt.gcf()
fig1.savefig('ExplorationPlots/Tracks,Plays,RepeatabilityVsLanguage.png',␣
 ↪dpi=100, bbox_inches='tight')
```


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


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


Chance of repeated listen
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
1
2
3
4
6
7
10
Number of languages

1.13 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())
```

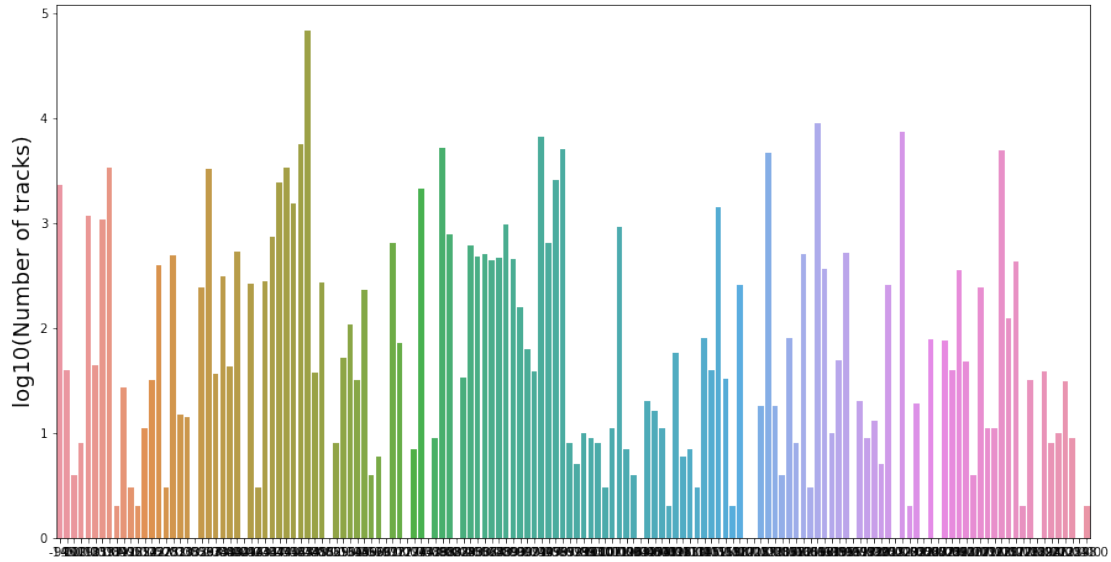
[34]:
```
len(n_genres), song_data['genre_ids'].isnull().sum()
```

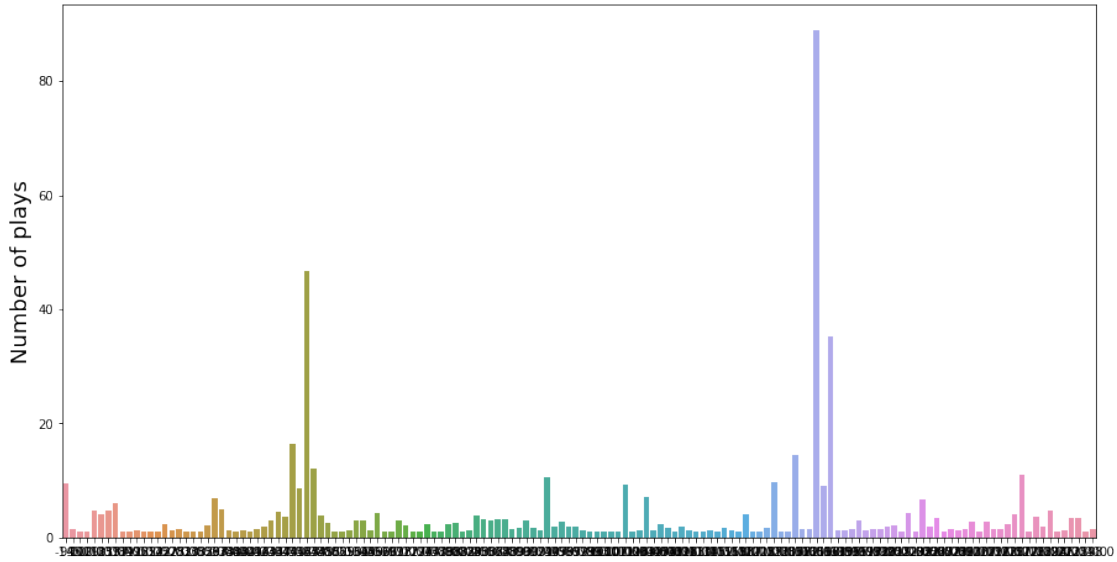
[34]:
```
(145, 2284)
```

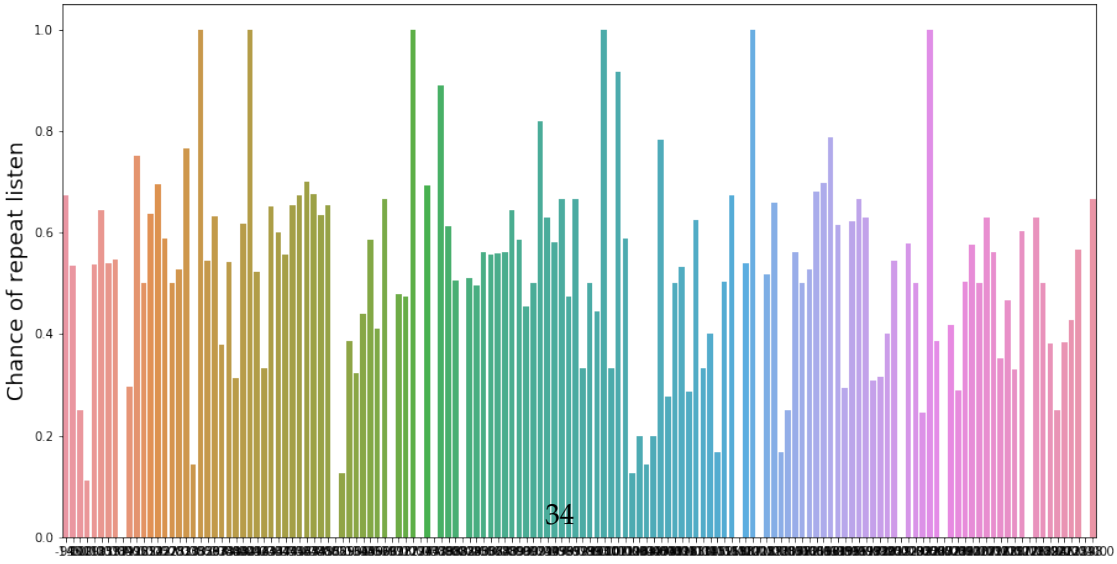
[35]:
```
genres_plays = [0] * (len(n_genres) + 1)
genres_tracks = [0] * (len(n_genres) + 1)
genres_replays = [0] * (len(n_genres) + 1)

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)]
```

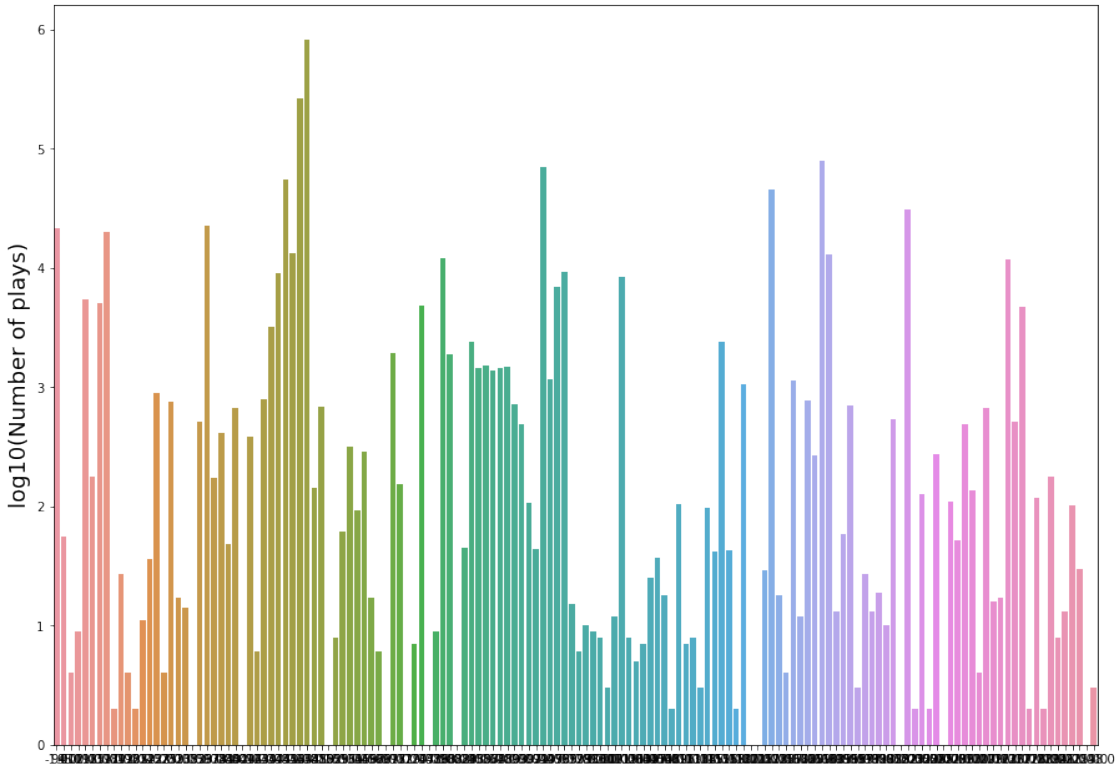
[36]:
```
n_g_l = [x for x in n_genres]
n_g_l.append(-1)

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)',fontsize=18)


ax413 = plt.subplot(3,1,2)
sns.barplot(x=n_g_l,y=genres_rel_plays)
ax413.set_ylabel('Number of plays',fontsize=18)

ax414 = plt.subplot(3,1,3)
sns.barplot(x=n_g_l,y=genres_repl_chance)
ax414.set_ylabel('Chance of repeat listen',fontsize=18)
fig1 = plt.gcf()
fig1.savefig('ExplorationPlots/Track,Plays,RepeatabilityVsGenre.png', dpi=100,␣
 ↪bbox_inches='tight')
```


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


Number of plays
0
20
40
60
80


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)',fontsize=18)
fig1 = plt.gcf()
fig1.savefig('ExplorationPlots/PlaysVsGenre.png', dpi=100, bbox_inches='tight')
```



1.14 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
157
```

[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)
```

```
16 [1061.0, 2045.0, 1598.0, 1089.0, 1117.0, 2144.0, 1162.0, 2192.0, 677.0,
166.0, 2245.0, 1266.0, 765.0, 303.0, 331.0, 1944.0]
```

[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))
```

```
[1, 2, 10, 1, 2, 12, 26, 1, 1, 10, 2, 2, 2, 1, 3, 12] 88
{'Leïla Martial', 'The Unspoken Rules', 'Spiral69', 'Rakim', 'TheOvertunes',
'Musikimia', 'Ville Ojanen', 'Los 3 Deos', 'Lovi', 'Wardruna', '', 'Billie
Holiday|Ella Fitzgerald|Nina Simone|Sarah Vaughan', 'Various Artists', 'Fabrice
Millischer', 'Sebastiano Serafini', 'Midnight Lamp  ', 'Klum Baumgartner|
Elektronik Kitchen Of Ideas', 'goldenage ()', 'Sarah Darling', 'Isyana
Sarasvati', 'Judika', 'Paul Simon', 'Youn Sun Nah', '', 'Paul Oakenfold'}
```