Original data

1m playlists along with tracks (songs)/albums/artists which are part of particular playlist. Data is in JSON format, split into 1000 files. After download, we adjusted code provided with the dataset to read, clean and reconcile the data.

Problem

Size. Loaded as-is, this dataset wouldn't fit into memory (~66m records). Names of songs / artists / playlists are duplicated which is a waste. Therefore, preprocessing utility was adjusted to split data into tables, adding ids for each entry: songs, artists, albums, playlists, tracks. Names were sanitized: brought to lower case, removed leading/trailing space, removed non-ASCII letter symbols, skipped entry if resulting string is empty (some names were all-hieroglyphs). This also allowed an easier analysis of tables (i.e. group by names). Even after the data normalization, track data wouldn't fit into a notebook with 8Gb of RAM. That, plus early results of EDA (~70% of track data belongs to playlists with just 1 follower), led to removing track data that has a single follower. Resulting dataframe fits into RAM.

Song details

('danceability', 'energy', 'key', 'loudness', 'mode', 'speechiness', 'acousticness', 'instrumentalness', 'liveness', 'valence', 'tempo', 'time_signature') are being downloaded through Spotify's REST API (using Spotify ID from playlist dataset to reconcile data). It's a slow and laborious process because authentication token must be re-taken every hour (manually) plus there's a rate limit (~4 songs per second). At this point, ~90% data has been downloaded.

After cleaning and preprocessing :

- Songs: 1389690. Not all songs are found in Spotify – hence post-join with playlist tracks entries which have N/A are dropped. Since this is a rare occurrence, we don't expect major impact on quality. Spotify URI is used to fetch song details (danceability, loudness, etc)
- Artists: 281893. Names are used to find match in similarity dataset but otherwise only having "same" artist (same id) seems to be important for playlists
- Albums: 547603. Same as artist: use for matching but otherwise track-album id is used
- Playlists: 984547; after dropping single-follower and top-follower (71643) playlists: 241799
- Tracks: 65138632; after dropping single-follower playlists: 19822736; after joining to song details 18660867 (preliminary - song details download is in-progress)

Playlist names

(17384 unique entries) don't look descriptive: although some provide context (e.g. decade-related '00s' repeated 219 times, '10' – 175; mood: 'zone' - 107 'zumba' – 342, 'zombie' - 25), many names look strange ('zzz' – 218 times). Until more promising modelling is done, no lexical matching is done on names.

Number of playlists by followers

Are heavily skewed by playlists with few followers, details below

Engineered features for playlists

Joining back track data with song details and representing playlist by its own data (number of tracks/albums/followers) and aggregating (depending on the field - e.g max mode) of song data per playlist, gives engineered features of the playlist (all fields range 0-1 for modelling) which allows modelling of playlists – e.g. clustering based on these features.

PCA

It was done on these features: 88% of variation is explained by 4 components. Since number of engineered features is not large, PCA is not used in modelling.

In [1]:
```
import sys
import json
import codecs
import datetime
import numpy as np
import pandas as pd
import string
DATA_DIR="./data/data"
```

In [2]:
```
df = pd.read_csv(DATA_DIR + '/preproc.csv.gz', compression='gzip')
dfAugSongs = pd.read_csv(DATA_DIR + '/full_aug_songs.csv.gz', compression='gzip
')
dfPlaylists = pd.read_csv(DATA_DIR + '/playlists.csv.gz', compression='gzip')
```

In [3]:
```
nfol = dfPlaylists.groupby('num_followers').agg(['count'])
```

In [4]:
```
nfol[nfol > 50].dropna()
```

Out[4]:

	id	name	collaborative	num_tracks	num_albums
	count	count	count	count	count
num_followers					
1	742663.0	742659.0	742663.0	742663.0	742663.0
2	147138.0	147137.0	147138.0	147138.0	147138.0
3	46178.0	46178.0	46178.0	46178.0	46178.0
4	19304.0	19304.0	19304.0	19304.0	19304.0
5	9669.0	9669.0	9669.0	9669.0	9669.0
6	5276.0	5276.0	5276.0	5276.0	5276.0
7	3256.0	3256.0	3256.0	3256.0	3256.0
8	2119.0	2119.0	2119.0	2119.0	2119.0
9	1494.0	1494.0	1494.0	1494.0	1494.0
10	998.0	998.0	998.0	998.0	998.0
11	820.0	820.0	820.0	820.0	820.0
12	626.0	626.0	626.0	626.0	626.0
13	468.0	468.0	468.0	468.0	468.0
14	356.0	356.0	356.0	356.0	356.0
15	325.0	325.0	325.0	325.0	325.0
16	285.0	285.0	285.0	285.0	285.0
17	232.0	232.0	232.0	232.0	232.0
18	206.0	206.0	206.0	206.0	206.0
19	161.0	161.0	161.0	161.0	161.0
20	137.0	137.0	137.0	137.0	137.0
21	117.0	117.0	117.0	117.0	117.0
22	124.0	124.0	124.0	124.0	124.0
23	106.0	106.0	106.0	106.0	106.0
24	95.0	95.0	95.0	95.0	95.0
25	66.0	66.0	66.0	66.0	66.0
26	69.0	69.0	69.0	69.0	69.0
27	73.0	73.0	73.0	73.0	73.0
28	70.0	70.0	70.0	70.0	70.0
29	71.0	71.0	71.0	71.0	71.0
31	64.0	64.0	64.0	64.0	64.0
32	60.0	60.0	60.0	60.0	60.0
36	65.0	65.0	65.0	65.0	65.0

In [5]:
```
dfNumFol = dfPlaylists[dfPlaylists.num_followers > 1] # remove min
dfNumFol = dfNumFol[dfNumFol.num_followers < 71643] # remove max as outlier
dfPlNumFol = pd.merge(df, dfNumFol, left_on='playlist_id', right_on='id', how='
left').dropna()
```

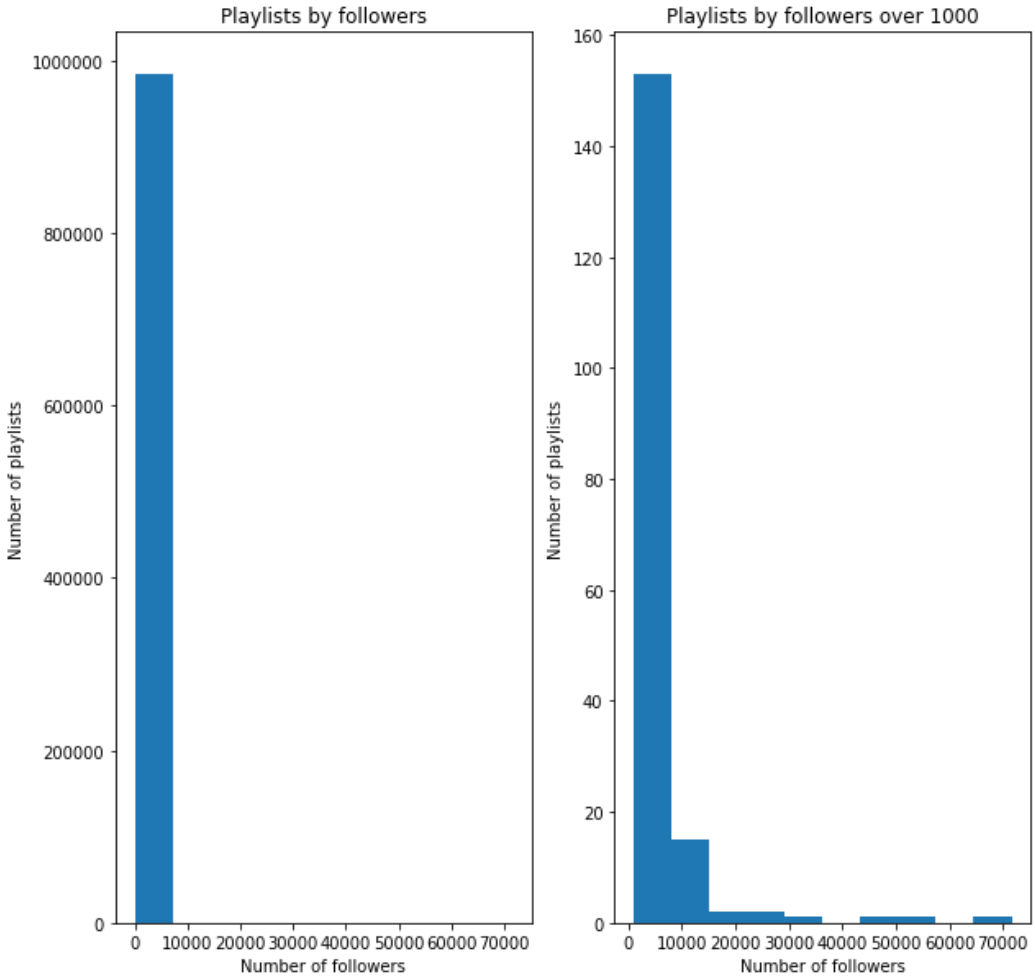
In [6]:
```
dfPlNumFol.drop(['id_y', 'name', 'num_followers', 'collaborative', 'num_tracks'
, 'num_albums'], axis=1, inplace=True)
```

In [7]:
```
dfPlNumFol.rename(columns={'id_x': 'pidpos_id'}, inplace=True)
```

In [8]:
```
dfPlNumFol.to_csv(DATA_DIR + '/pidpos.csv.gz', compression="gzip")
```

In [9]:
```
import matplotlib.pyplot as plt
plt.hist(dfPlaylists.num_followers);
```

In [10]:
```
fig, ax = plt.subplots(1, 2, figsize=(10, 10))
ax[0].hist(dfPlaylists[dfPlaylists.num_followers > 0].num_followers)
ax[0].set_title("Playlists by followers")
ax[0].set_xlabel('Number of followers')
ax[0].set_ylabel('Number of playlists')
ax[1].hist(dfPlaylists[dfPlaylists.num_followers > 1000].num_followers)
ax[1].set_title("Playlists by followers over 1000")
ax[1].set_xlabel('Number of followers')
ax[1].set_ylabel('Number of playlists');
```



In [11]:
```
dfSongs = pd.read_csv(DATA_DIR + '/orig_songs.csv.gz', compression='gzip')
dfSongs.head()
```

Out[11]:

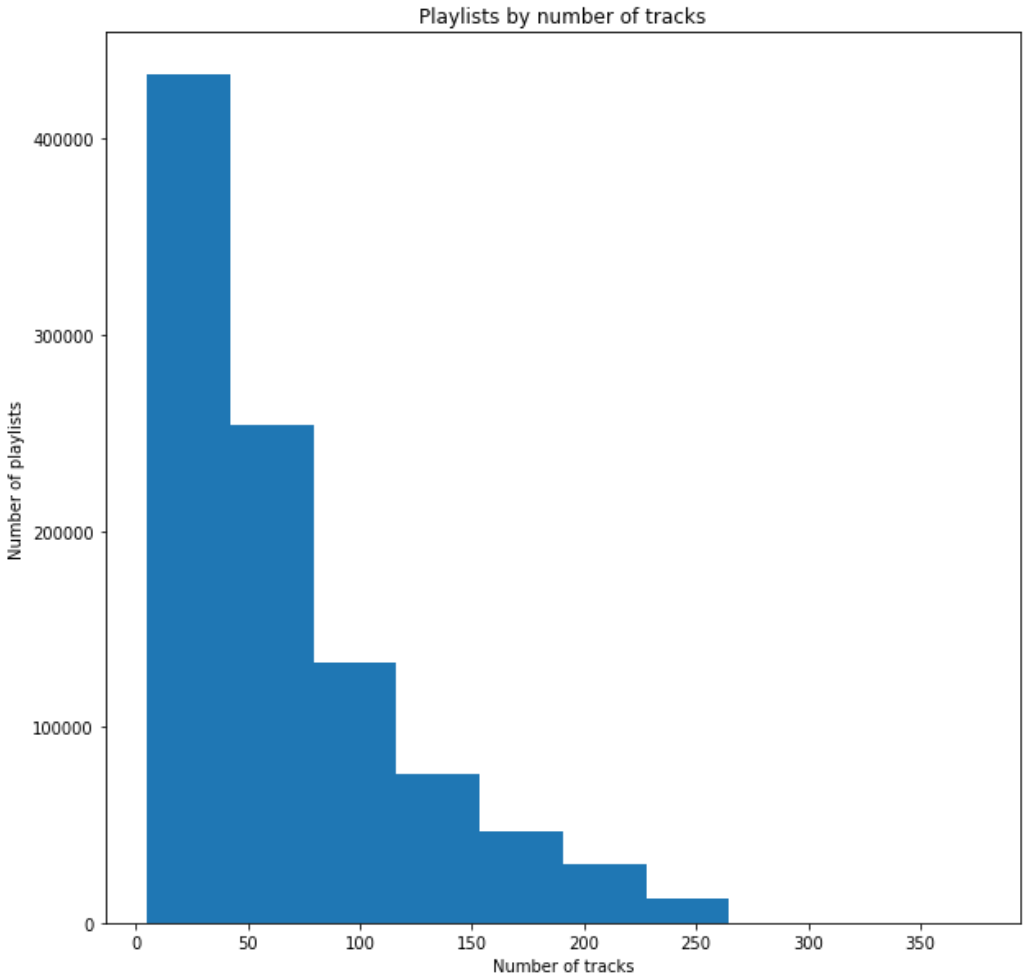
	name	id	uri	duration_ms
0	lose control feat ciara fat man scoop	0	0UaMYEvWZi0ZqiDOoHU3YI	226.863
1	toxic	1	6I9VzXrHxO9rA9A5euc8Ak	198.800
2	crazy in love	2	0WqIKmW4BTrj3eJFmnCKMv	235.933
3	rock your body	3	1AWQoqb9bSvzTjaLralEkT	267.266
4	it wasnt me	4	1Izr43nnXAijIGYnCT8M8H	227.600

In [13]:
```
dfAugSongs.describe()
```

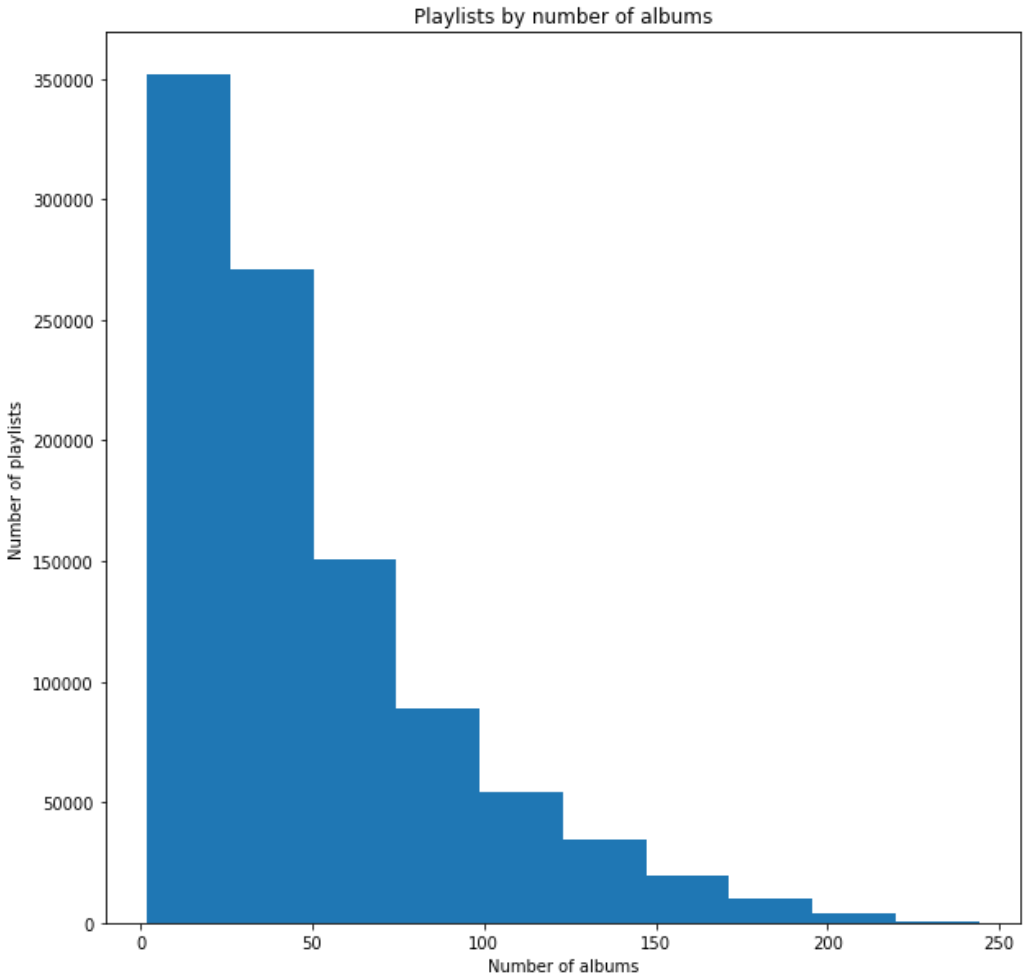
Out[13]:

	id	duration	danceability	energy	key	loudness
count	265512.000000	265512.000000	265512.000000	265512.000000	265512.000000	265512.000000
mean	132317.224099	244.971160	0.564075	0.618639	5.260873	-8.648893
std	76351.805012	128.179798	0.179457	0.249564	3.571320	5.021407
min	0.000000	0.000000	0.000000	0.000000	0.000000	-60.000000
25%	66383.750000	190.862000	0.447000	0.451000	2.000000	-10.486000
50%	132037.500000	227.040000	0.578000	0.660000	5.000000	-7.333500
75%	198417.250000	274.384500	0.697000	0.824000	8.000000	-5.335000
max	264795.000000	5823.660000	0.991000	1.000000	11.000000	3.792000

In [45]:
```
fig, ax = plt.subplots(1, 1, figsize=(10, 10))
ax.hist(dfPlaylists.num_tracks)
ax.set_title("Playlists by number of tracks")
ax.set_xlabel('Number of tracks')
ax.set_ylabel('Number of playlists');
```



In [46]:
```
fig, ax = plt.subplots(1, 1, figsize=(10, 10))
ax.hist(dfPlaylists.num_albums)
ax.set_title("Playlists by number of albums")
ax.set_xlabel('Number of albums')
ax.set_ylabel('Number of playlists');
```



In [16]:
```
dfPlTracks = pd.merge(df, dfNumFol, left_on='playlist_id', right_on='id', how='
left').dropna()
```

In [17]:
```
dfPlTracks.drop(['id_y', 'name', 'collaborative'], axis=1, inplace=True)
dfPlTracks.rename(columns={'id_x': 'pidpos_id'}, inplace=True)
```

In [18]:
```
dfPlTracks.head()
```

Out[18]:

	pidpos	pidpos_id	track_id	album_id	artist_id	playlist_id	num_followers	num_tracks	num_a
281	4-0	281	280	225	175	4	2.0	17.0	16.0
282	4-1	282	281	226	176	4	2.0	17.0	16.0
283	4-2	283	282	227	177	4	2.0	17.0	16.0
284	4-3	284	283	228	178	4	2.0	17.0	16.0
285	4-4	285	284	229	179	4	2.0	17.0	16.0

In [19]:
```
dfPlSongs = pd.merge(dfPlTracks, dfAugSongs, left_on='track_id', right_on='id',
how='left').dropna()
```

In [20]:
```
dfPlSongs.head()
```

Out[20]:

	pidpos	pidpos_id	track_id	album_id	artist_id	playlist_id	num_followers	num_tracks	num_alb
0	4-0	281	280	225	175	4	2.0	17.0	16.0
1	4-1	282	281	226	176	4	2.0	17.0	16.0
2	4-2	283	282	227	177	4	2.0	17.0	16.0
3	4-3	284	283	228	178	4	2.0	17.0	16.0
4	4-4	285	284	229	179	4	2.0	17.0	16.0

5 rows × 25 columns

In [21]:
```
dfPlSongs.columns
```

Out[21]:
```
Index(['pidpos', 'pidpos_id', 'track_id', 'album_id', 'artist_id',
       'playlist_id', 'num_followers', 'num_tracks', 'num_albums', 'name',
       'id', 'uri', 'duration', 'danceability', 'energy', 'key', 'loudness',
       'mode', 'speechiness', 'acousticness', 'instrumentalness', 'liveness',
       'valence', 'tempo', 'time_signature'],
      dtype='object')
```

In [22]:
```
dfPlSongs.drop(['id', 'name', 'uri'], axis=1, inplace=True)
```

In [23]:
```
dfPlGroupped = dfPlSongs.groupby('playlist_id')
```

In [24]:
```
# trust direct count of tracks, not the field - some songs haven't been downloa
ded yet
# key (pitch) is integer hence - max
# loudness, tempo, time_signature: 'max'
# mode: drop (not mean, why max / min ?)
dfPlSongsAgg = dfPlGroupped.agg({'track_id': 'count', 'num_followers' : 'first'
, 'num_tracks' : 'first',
                  'num_albums' : 'first', 'duration': 'mean',
                  'danceability': 'mean', 'energy': 'mean', 'key' : 'max', 'lou
dness': 'max',
       'speechiness': 'mean', 'acousticness': 'mean', 'instrumentalness': 'mean
', 'liveness': 'mean',
       'valence': 'mean', 'tempo': 'max', 'time_signature': 'max'}).rename(
    columns={'track_id': 'sum_num_tracks', 'duration': 'mean_duration', 'dancea
bility': 'mean_danceability', 'energy': 'mean_energy',
             'key' : 'max_key', 'loudness': 'max_loudness',
       'speechiness': 'mean_speechiness', 'acousticness': 'mean_acousticness',
        'instrumentalness': 'mean_instrumentalness', 'liveness': 'mean_liveness
',
       'valence': 'mean_valence', 'tempo': 'max_tempo', 'time_signature': 'max_
time_signature' })
```

In [25]:
```
dfPlSongsAgg.head()
```

Out[25]:

	sum_num_tracks	num_followers	num_tracks	num_albums	mean_duration	mean_dan
playlist_id						
4	17	2.0	17.0	16.0	255.016588	0.576765
8	46	2.0	46.0	37.0	216.280891	0.512370
10	72	2.0	72.0	60.0	229.388917	0.725931
20	14	3.0	14.0	9.0	171.202429	0.824429
22	42	2.0	42.0	39.0	215.148548	0.645238

In [26]:
```
# incomplete song data - still being downloaded
dfPlSongsAgg[dfPlSongsAgg.sum_num_tracks != dfPlSongsAgg.num_tracks].max_key.co
unt()
```

Out[26]: 152227

In [27]:
```
# normalize
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
dfPlSongsAgg[['sum_num_tracks', 'num_tracks', 'num_albums', 'mean_duration', 'm
ax_key',
              'max_loudness', 'max_tempo', 'max_time_signature']] = scaler.fit_
transform(
    dfPlSongsAgg[['sum_num_tracks', 'num_tracks', 'num_albums', 'mean_duration'
,
                                        'max_key', 'max_loudness', 'max_tempo',
'max_time_signature']])
```

In [28]:
```
dfPlSongsAgg.head()
```

Out[28]:

	sum_num_tracks	num_followers	num_tracks	num_albums	mean_duration	mean_dan
playlist_id						
4	0.053872	2.0	0.048980	0.058333	0.043293	0.576765
8	0.151515	2.0	0.167347	0.145833	0.034563	0.512370
10	0.239057	2.0	0.273469	0.241667	0.037517	0.725931
20	0.043771	3.0	0.036735	0.029167	0.024404	0.824429
22	0.138047	2.0	0.151020	0.154167	0.034308	0.645238

In [29]:
```
dfPlSongsAgg.to_csv(DATA_DIR + '/aug_playlists.csv.gz', compression="gzip")
```

In [30]:
```
from sklearn.decomposition import PCA
pca = PCA()
pca.fit(dfPlSongsAgg)
plSongsAgg_pca = pca.transform(dfPlSongsAgg)
```

In [31]:
```
plSongsAgg_pca
```

Out[31]:
```
array([[-5.26570221e+00, -4.00077050e-01, -5.97823891e-02, ...,
        -1.70195438e-02, -6.43652413e-03,  1.99573931e-03],
       [-5.26570176e+00, -2.13333439e-01, -7.34344757e-02, ...,
         1.95680783e-02, -6.91630848e-03, -6.18612150e-03],
       [-5.26570178e+00, -4.07861078e-02, -9.54723793e-02, ...,
         1.60570187e-02, -1.48854729e-02,  1.30105156e-03],
       ...,
       [-5.26569821e+00,  6.07251470e-01,  3.06131730e-01, ...,
        -1.13039556e-02,  3.52395122e-03,  3.75963882e-04],
       [-5.26570271e+00, -3.33009279e-01, -1.32031598e-01, ...,
         2.01985984e-02, -6.51052815e-02, -4.22438075e-03],
       [-5.26570097e+00, -2.69565443e-01,  2.24656246e-01, ...,
        -6.06234178e-02, -2.66207575e-02, -1.07814352e-03]])
```

In [32]:
```
sum_variance, component_count = 0, 0
while sum_variance < 0.85:
    sum_variance += pca.explained_variance_ratio_[component_count]
    component_count += 1

print('Number of Principal Components that explain >=85% of Variance: ', compon
ent_count)
print('Total Variance Explained by '+str(component_count)+' components:', str(s
um_variance*100)+'%')
```

```
Number of Principal Components that explain >=85% of Variance:  1
Total Variance Explained by 1 components: 99.99956072623758%
```

In [33]:
```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=10, random_state=0).fit(dfPlSongsAgg)
```

In [34]:
```
kmeans.labels_
```

Out[34]:
```
array([0, 0, 0, ..., 0, 0, 0], dtype=int32)
```

In [35]:
```
kmeans.cluster_centers_
```

Out[35]:
```
array([[2.56485005e-01, 3.51142654e+00, 3.15235927e-01, 2.51979118e-01,
        3.87335284e-02, 6.12080067e-01, 6.39715216e-01, 9.90645744e-01,
        8.49586489e-01, 9.45291764e-02, 2.34634695e-01, 5.79217407e-02,
        1.86836720e-01, 4.91607804e-01, 7.81176972e-01, 8.92775347e-01],
       [1.24579125e-01, 2.28010000e+04, 1.53061224e-01, 1.62500000e-01,
        3.47872841e-02, 5.90602953e-01, 6.29248811e-01, 1.00000000e+00,
        8.12799885e-01, 9.14648482e-02, 2.27811302e-01, 5.63848588e-02,
        2.03426128e-01, 5.54462961e-01, 7.54142088e-01, 8.00000000e-01],
       [2.27425773e-01, 1.06494545e+04, 3.23191095e-01, 1.33333333e-01,
        4.20223397e-02, 6.10645694e-01, 5.95658106e-01, 1.00000000e+00,
        8.31586737e-01, 9.37783396e-02, 2.64282487e-01, 1.20797576e-01,
        1.90890294e-01, 4.67734203e-01, 7.69939985e-01, 8.90909091e-01],
       [2.99663300e-01, 4.97305000e+04, 4.22448980e-01, 3.81250000e-01,
        4.03486842e-02, 5.50774350e-01, 6.23402677e-01, 1.00000000e+00,
        8.50707789e-01, 6.69327002e-02, 2.63845752e-01, 9.10937069e-02,
        1.77731158e-01, 4.85033631e-01, 7.83157445e-01, 8.00000000e-01],
       [2.37143363e-01, 1.81252632e+03, 3.05649839e-01, 2.24517544e-01,
        3.92306167e-02, 5.88960689e-01, 6.21113691e-01, 9.79904306e-01,
        8.40717451e-01, 8.15314535e-02, 2.81929259e-01, 6.45956108e-02,
        1.87169997e-01, 4.83522168e-01, 7.68919505e-01, 8.98947368e-01],
       [3.53054353e-01, 6.87428571e+03, 4.54227405e-01, 2.90178571e-01,
        4.01591818e-02, 5.77118000e-01, 6.38399155e-01, 9.93506494e-01,
        8.56750683e-01, 9.36443491e-02, 2.38562335e-01, 1.06845725e-01,
        1.92557294e-01, 4.62526088e-01, 8.13007205e-01, 9.42857143e-01],
       [2.01178451e-01, 1.46597500e+04, 2.37755102e-01, 1.45833333e-01,
        3.87751388e-02, 6.35418933e-01, 6.81821100e-01, 9.77272727e-01,
        8.39088348e-01, 1.10400991e-01, 1.73343638e-01, 2.55005509e-02,
        1.84883420e-01, 4.85866106e-01, 8.13477446e-01, 9.00000000e-01],
       [3.15151515e-01, 3.78733333e+03, 4.14557823e-01, 3.14444444e-01,
        3.93125253e-02, 5.85017415e-01, 6.36766049e-01, 9.93939394e-01,
        8.52098701e-01, 8.72829991e-02, 2.50847350e-01, 7.80740039e-02,
        1.87838387e-01, 4.49326894e-01, 7.92581023e-01, 9.06666667e-01],
       [2.52500312e-01, 5.44522222e+02, 3.41360544e-01, 2.37438272e-01,
        3.84172366e-02, 5.90198411e-01, 6.20607206e-01, 9.82828283e-01,
        8.45342189e-01, 9.06914819e-02, 2.68934431e-01, 8.30649808e-02,
        1.86253047e-01, 4.69576657e-01, 7.75703930e-01, 8.94074074e-01],
       [2.03703704e-01, 2.96845000e+04, 3.18367347e-01, 1.66666667e-01,
        2.95165475e-02, 5.94047619e-01, 5.85061596e-01, 1.00000000e+00,
        8.34379700e-01, 7.68080247e-02, 3.44960353e-01, 4.67019029e-02,
        1.88245635e-01, 4.70625309e-01, 7.46683426e-01, 9.00000000e-01]])
```

In [36]:
```
kmeans.predict([[0.053872, 0.000000, 0.048980, 0.058333, 0.444, 0.576765, 0.650
535, 0.818182, 0.812418,
                 0.041159, 0.177148, 0.081875, 0.166524, 0.99, 0.723059, 0.8
]])
```

Out[36]:
```
array([0], dtype=int32)
```

In [47]:
```
dfPlSongsAgg.num_tracks.count()
```

Out[47]:
```
241798
```

In [38]:
```
groupedPlNames = dfPlaylists.groupby('name', sort=False).count()
groupedPlNames.sort_index(ascending=False)
# little sense in playlist names
```

Out[38]:

name	id	num_followers	collaborative	num_tracks	num_albums
zzzzzzzzzz	6	6	6	6	6
zzzzzzzzz	13	13	13	13	13
zzzzzzzz	18	18	18	18	18
zzzzzzz	34	34	34	34	34
zzzzzz	55	55	55	55	55
zzzzz	57	57	57	57	57
zzzz	80	80	80	80	80
zzz	218	218	218	218	218
zz top	5	5	5	5	5
zz	24	24	24	24	24
zyzz	5	5	5	5	5
zydeco	54	54	54	54	54
zumba	342	342	342	342	342
zucchero	1	1	1	1	1
zouk	45	45	45	45	45
zoom	14	14	14	14	14
zoo	16	16	16	16	16
zoned	27	27	27	27	27
zone out	76	76	76	76	76
zone	107	107	107	107	107
zona ganjah	1	1	1	1	1
zomboy	4	4	4	4	4
zombies	64	64	64	64	64
zombie	25	25	25	25	25
zoe	56	56	56	56	56
zo	1	1	1	1	1
zion	36	36	36	36	36
zik	1	1	1	1	1
ziggy	12	12	12	12	12
zhu	6	6	6	6	6
...	...	...	...	...	...
10	175	175	175	175	175
1 playlist	29	29	29	29	29
1 5	1	1	1	1	1

In [42]:
```
df.pidpos.count()
```

Out[42]: 65138632

In [43]:
```
dfPlTracks.pidpos.count()
```

Out[43]: 19822697

In [44]:
```
dfPlSongs.pidpos.count()
```

Out[44]: 18660828