Spotify Music Analysis

Travis Bostick
Connor Hertz
Adam Johnson
Joe Kell

Agenda

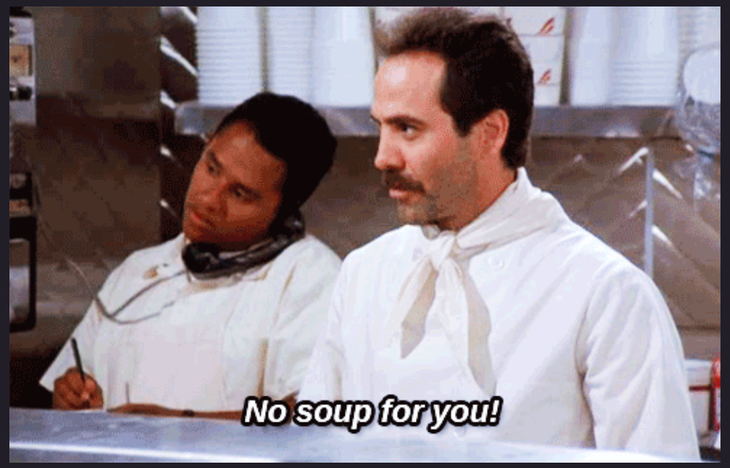
- Motivation
- About the data
- Audio Features in relation to country metrics
- Audio Features over time
- Impact on streams when artists die
- Additional questions and difficulties

Motivation

When looking at the Spotify data the thing that jumped out was the Audio Features. We focused heavily on analyzing those and how they correlate over time and region.

Looking at historic chart data on Spotify made us think of artists with streaming spikes after death.

About the Data



Spotify

With data from Spotify, we had access to the Audio Features for each track

Kaggle

This is where we found the data sets for the spotify audio features and country information

Beautiful Soup

Spotify Charts allows us to see which songs are popular in each region, so we scraped that data

```python
#Import Dependencies
import requests
from bs4 import BeautifulSoup
import csv
```

```python
#The purpose of this code is to pull a list of possible dates for spotify charts us in 2017.
date_codes=[]
date_names=[]
Original_url="https://spotifycharts.com/regional/us/daily/latest"
list_response=requests.get(Original_url)
soup=BeautifulSoup(list_response.text, "lxml")
#I can add to this later to change from daily or choose another date.
dropdowns=soup.find_all('div', class_='responsive-select')
dates=dropdowns[2].find_all('li')
for date in dates:
    date_codes.append(date['data-value'])
    date_names.append(date.text)
```

```python
#This is to limit the list of dates to 2017
for i in range(len(date_codes)-1,-1,-1):
    if date_codes[i][:4]!="2017":
        date_codes.pop(i)
        date_names.pop(i)
```

```python
#Open CSV
outPath="Resources/2017-daily-us-charts.csv"
with open(outPath, 'w', newline='') as csvfile:
    # Initialize csv.writer
    csvwriter = csv.writer(csvfile, delimiter=',')
    #Creating a header for the data
    csvwriter.writerow(["Date","Position","Track Name","Artist","Streams","ID"])

#Looping through the region codes and using them in the url to scrape the csv
    for Num,date in enumerate(date_codes):
        Current_Date=date_names[Num]
        url=f"https://spotifycharts.com/regional/us/daily/{date}/download"
        CSVresponse=requests.get(url)
        Cleaned_Response=str(CSVresponse.content).split(chr(92)+"n")
        Cleaned_Response.pop(len(Cleaned_Response)-1)
        Cleaned_Response.pop(1)
        Cleaned_Response.pop(0)
        reader= csv.reader(Cleaned_Response,delimiter=",")
        for row in reader:
            CleanRow=[Current_Date] + row
            CleanRow[5]=CleanRow[5][31:]
            csvwriter.writerow(CleanRow)
```

Spotify | Charts

DOWNLOAD TO CSV

TOP 200 VIRAL 50

Filter by GLOBAL DAILY 11/03/2020

		TRACK	STREAMS
	1	▲ **Dakiti** by Bad Bunny, Jhay Cortez	5,624,715
	2	▼ **positions** by Ariana Grande	5,604,245
	3	– **Mood (feat. iann dior)** by 24kGoldn	4,685,467
	4	– **Lemonade (feat. Gunna, Don Toliver & NAV)** by Internet Money	4,059,514
	5	▲ **Lonely (with benny blanco)** by Justin Bieber	3,474,677
	6	▲ **Dynamite** by BTS	3,331,271
	7	– **What You Know Bout Love** by Pop Smoke	3,329,334
	8	▼ **34+35** by Ariana Grande	3,293,347
	9	– **Hawái** by Maluma	2,958,768
	10	– **WAP (feat. Megan Thee Stallion)** by Cardi B	2,950,278

```python
#Import Dependencies
import requests
from bs4 import BeautifulSoup
import csv
```

```python
#The purpose of this code is to pull a list of possible dates for spotify charts us in 2017.
date_codes=[]
date_names=[]
Original_url="https://spotifycharts.com/regional/us/daily/latest"
list_response=requests.get(Original_url)
soup=BeautifulSoup(list_response.text, "lxml")
#I can add to this later to change from daily or choose another date.
dropdowns=soup.find_all('div', class_='responsive-select')
dates=dropdowns[2].find_all('li')
for date in dates:
    date_codes.append(date['data-value'])
    date_names.append(date.text)
```

```python
#This is to limit the list of dates to 2017
for i in range(len(date_codes)-1,-1,-1):
    if date_codes[i][:4]!="2017":
        date_codes.pop(i)
        date_names.pop(i)
```

```python
#Open CSV
outPath="Resources/2017-daily-us-charts.csv"
with open(outPath, 'w', newline='') as csvfile:
    # Initialize csv.writer
    csvwriter = csv.writer(csvfile, delimiter=',')
    #Creating a header for the data
    csvwriter.writerow(["Date","Position","Track Name","Artist","Streams","ID"])

#Looping through the region codes and using them in the url to scrape the csv
    for Num,date in enumerate(date_codes):
        Current_Date=date_names[Num]
        url=f"https://spotifycharts.com/regional/us/daily/{date}/download"
        CSVresponse=requests.get(url)
        Cleaned_Response=str(CSVresponse.content).split(chr(92)+"n")
        Cleaned_Response.pop(len(Cleaned_Response)-1)
        Cleaned_Response.pop(1)
        Cleaned_Response.pop(0)
        reader= csv.reader(Cleaned_Response,delimiter=",")
        for row in reader:
            CleanRow=[Current_Date] + row
            CleanRow[5]=CleanRow[5][31:]
            csvwriter.writerow(CleanRow)
```

```
# Dependencies
import pandas as pd
```

```
# Not every CSV requires an encoding, but be aware this can come up
US_Daily_2017_df = pd.read_csv("Resources/2017-daily-us-charts.csv")
```

```
# Show just the header
US_Daily_2017_df.head()
```

	Date	Position	Track Name	Artist	Streams	ID
0	12/31/2017	1	rockstar	Post Malone	1530210	7wGoVu4Dady5GV0Sv4UIsx
1	12/31/2017	2	No Limit	G-Eazy	1016584	2DQ1ITjI0YoLFzuADN1ZBW
2	12/31/2017	3	Bartier Cardi (feat. 21 Savage)	Cardi B	950886	75FDPwaULRdYDn4StFN2rT
3	12/31/2017	4	I Fall Apart	Post Malone	939975	75ZvA4QfFiZvzhj2xkaWAh
4	12/31/2017	5	Gucci Gang	Lil Pump	884672	43ZyHQITOjhciSUUNPVRHc

How have Audio Features changed over time?

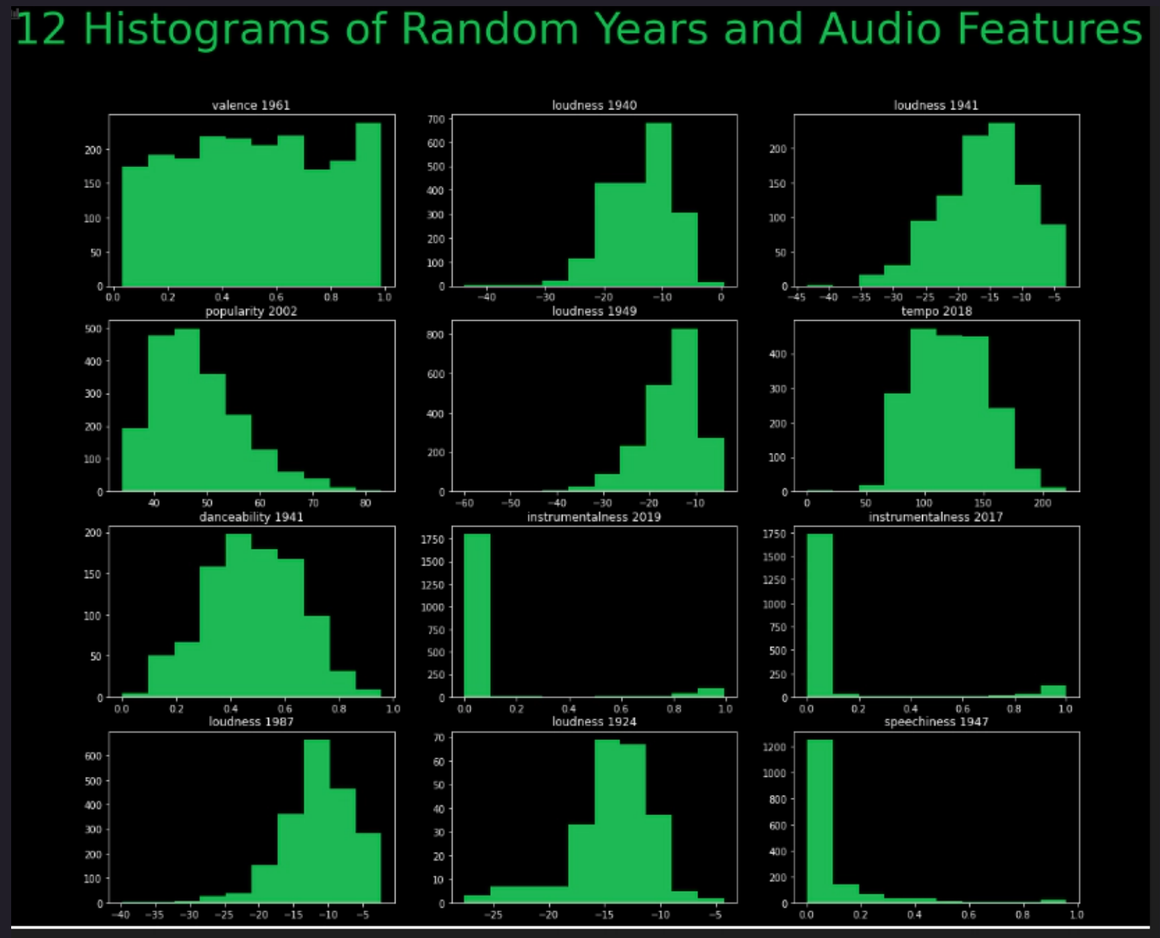
Audio Features

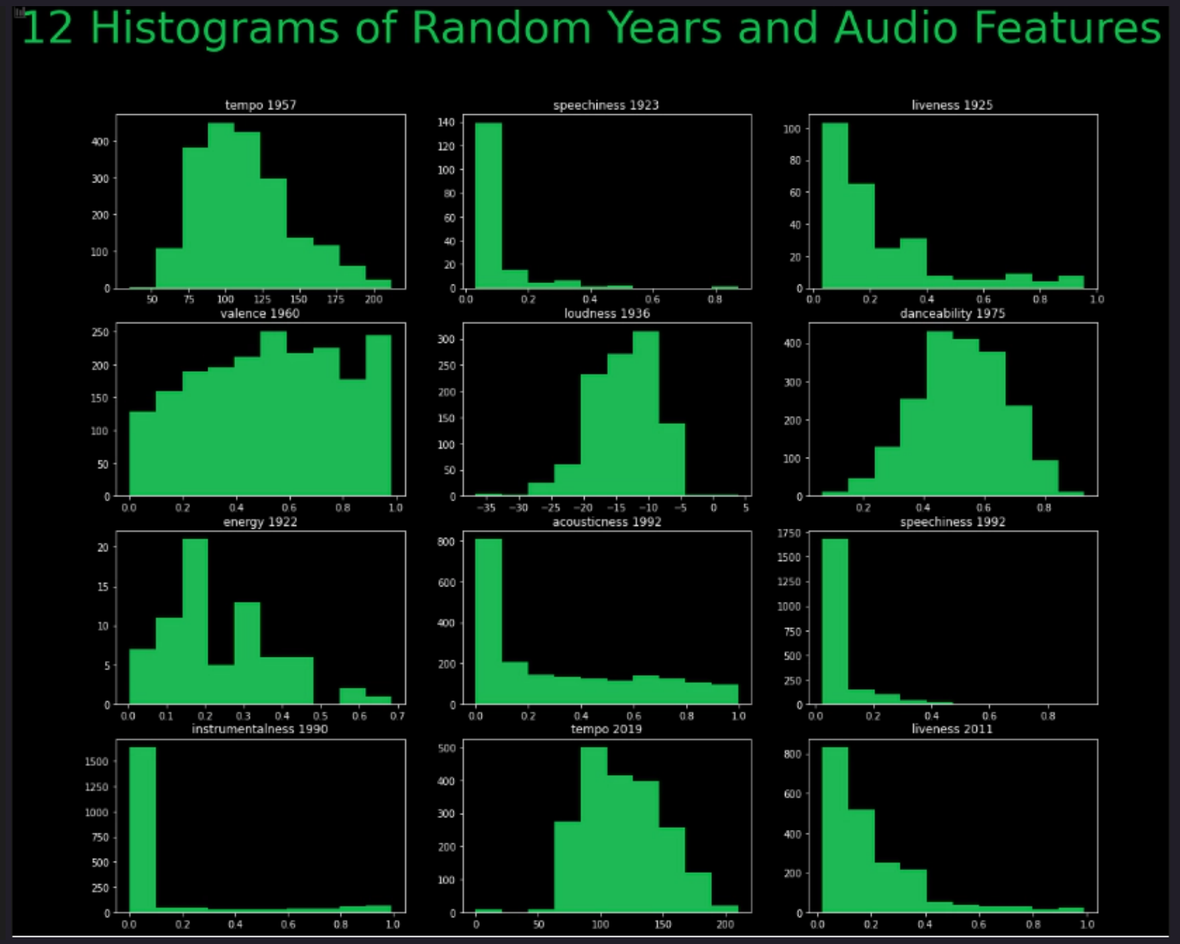
- *acousticness*
- *danceability*
- *energy*
- *instrumentalness*
- *liveness*
- *loudness*
- *speechiness*
- *valence*
- *tempo*

```
#creates a list of each year
years_list=kaggle_spot_df['year'].unique().tolist()

#list of each metric
metric_list=['acousticness','danceability','energy','instrumentalness','liveness',
'loudness', 'popularity','speechiness','tempo','valence']

##### creates 12 histograms for random years and audio features
fig_h=plt.figure(figsize=(18,15),facecolor='black')
fig_h.suptitle('12 Histograms of Random Years and Audio Features',color='#1DB954',
fontsize=45)

for t in range(1,13):

    rando_year=random.choice(years_list)
    rando_metric=random.choice(metric_list)
    ax_h=fig_h.add_subplot(4,3,t)
    ax_h.spines['bottom'].set_color('white')
    ax_h.spines['top'].set_color('white')
    ax_h.spines['right'].set_color('white')
    ax_h.spines['left'].set_color('white')
    ax_h.tick_params(axis='x', colors='white')
    ax_h.tick_params(axis='y', colors='white')
    ax_h.yaxis.label.set_color('white')
    ax_h.xaxis.label.set_color('white')
    ax_h.title.set_color('white')
    ax_h.patch.set_alpha(0)
    plt.title(str(rando_metric)+' '+str(rando_year))
    histy=kaggle_spot_df.loc[kaggle_spot_df['year']==rando_year,:][rando_metric]
    plt.hist(histy,color='#1DB954')
plt.show()
```

Reviewing histograms of random years and audio features to check distribution

If the histogram appears normally distributed, we use mean for the Audio Feature. Otherwise we use median.





```
### samples 10 random years for each audio features generating the P value for each...
# ... finds the average pvalue for each auido feature to determine if the audio feature is normally distributed

## adding the year for easier graphing in next cell
mean_graph_metric_list=['year']
median_graph_metric_list=['year']

alpha = 1e-3
# alpha= 0.05

# loop through a list of each audio feature(or metric)
for metric in metric_list:
    p_values_list=[]

    #we're gonna take this number of samples for each feature
    for a in range(10):
        #make a random year
        rando_year=random.choice(years_list)

        #run the normal test on a random year and the current audiot feature
        k2, p =stats.normaltest(kaggle_spot_df.loc[kaggle_spot_df['year']==rando_year,:][metric])

        p_values_list.append(p)

    #average of p values samples for current metric
    p_val_avg=sum(p_values_list)/len(p_values_list)

    # is it normally distributed or not? put the audio feature in the right list accordingly
    if p_val_avg < alpha:
        median_graph_metric_list.append(metric)
    else:
        mean_graph_metric_list.append(metric)
```

Samples 10 random years for each audio feature, tests the data to see if it's normally distributed

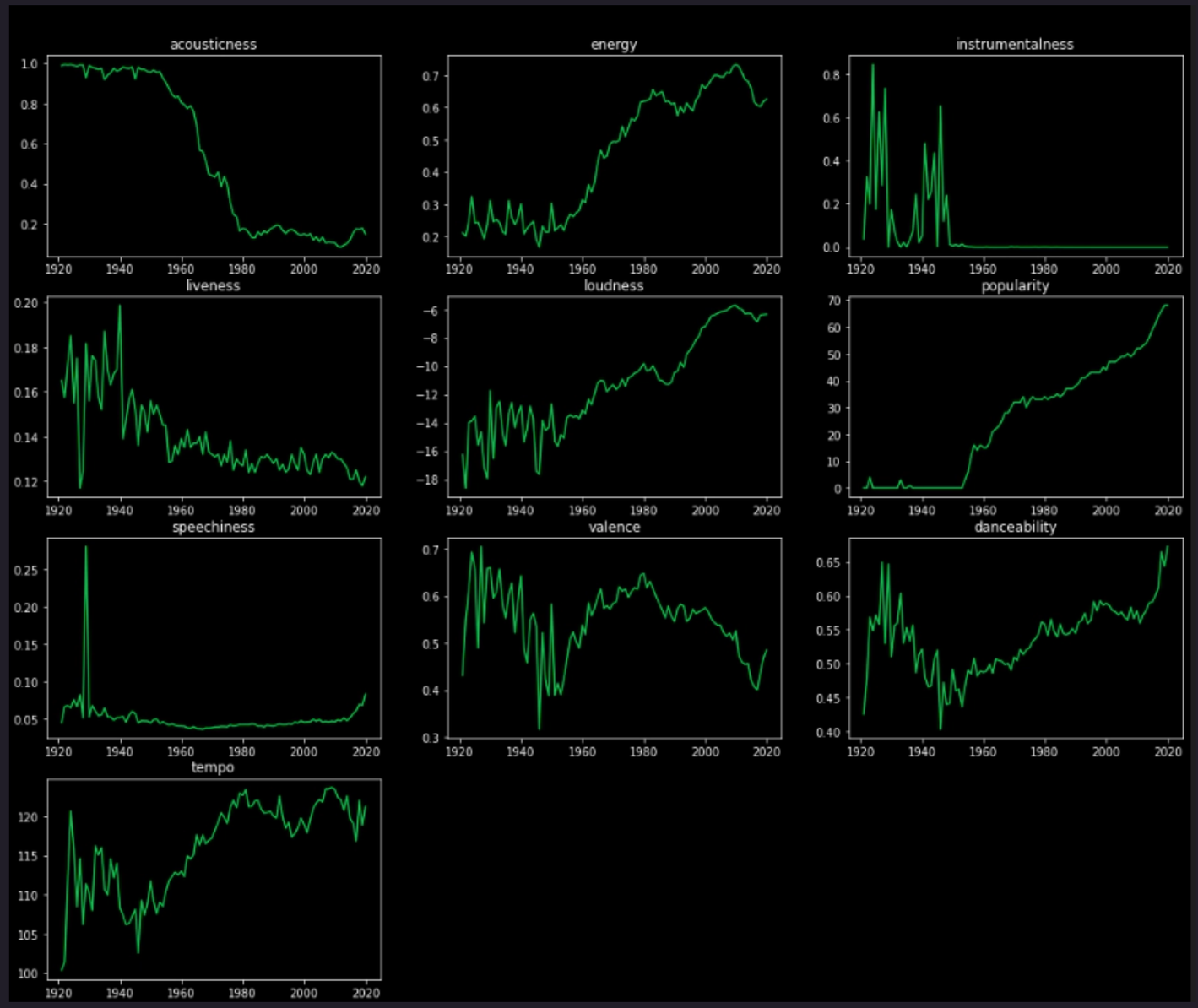
Creates a Dataframe with the annual mean or median for each audio feature

```
### creates a dataframe with the annual mean or median

## importing this from pandas to address DataError issue
from pandas.core.groupby.groupby import DataError
## added try block since mean list is empty sometimes
try:
    #dataframe of median metrics to be graphed
    median_annual_metrics_df=kaggle_spot_df[median_graph_metric_list].groupby('year').median()
    #dataframe of mean metrics to be graphed
    mean_annual_metrics_df=kaggle_spot_df[mean_graph_metric_list].groupby('year').mean()
    #merges the median dataframe and the mean dataframe together
    annual_metrics_df=pd.merge(median_annual_metrics_df,mean_annual_metrics_df,on='year')
except :
    print('one of these lists were empty')
    annual_metrics_df=kaggle_spot_df[['year','acousticness','danceability','energy','instrumentalness',
'liveness','loudness', 'popularity','speechiness','tempo','valence']].groupby('year').median()
```

```python
#graphs the average of each feature over time
def over_time_graphy(df):
    #turns the metric names into a list
    spot_metric_names=df.columns.to_list()
    ### plots graphs 4 by 3 with spotify colors
    fig = plt.figure(figsize=(18,15),facecolor='black')

    fig.suptitle('Audio Features Over Time',fontsize=50,color='#1DB954')
    for i in range(1,11):
        ax=fig.add_subplot(4,3,i)
        ax.spines['bottom'].set_color('white')
        ax.spines['top'].set_color('white')
        ax.spines['right'].set_color('white')
        ax.spines['left'].set_color('white')
        ax.tick_params(axis='x', colors='white')
        ax.tick_params(axis='y', colors='white')
        ax.yaxis.label.set_color('white')
        ax.xaxis.label.set_color('white')
        ax.title.set_color('white')
        ax.patch.set_alpha(0)
        x=df.index
        y=df.iloc[:,i-1]
        plt.plot(x,y,c='#1DB954')
        plt.title(spot_metric_names[i-1])
    return plt.show()
over_time_graphy(annual_metrics_df)
```

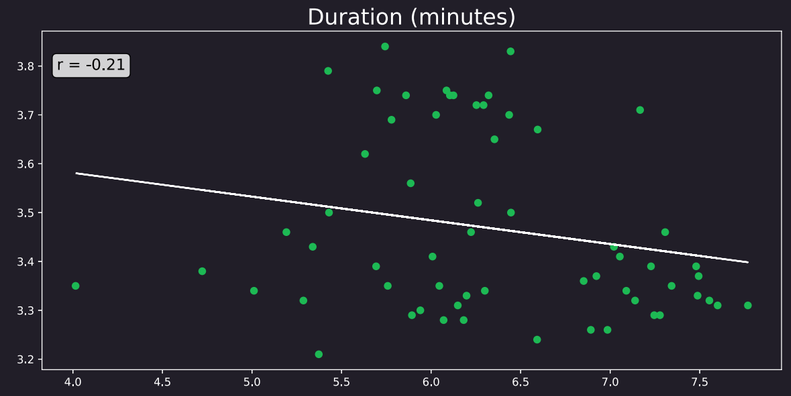
Graphing the Dataframe with Spotify Styling

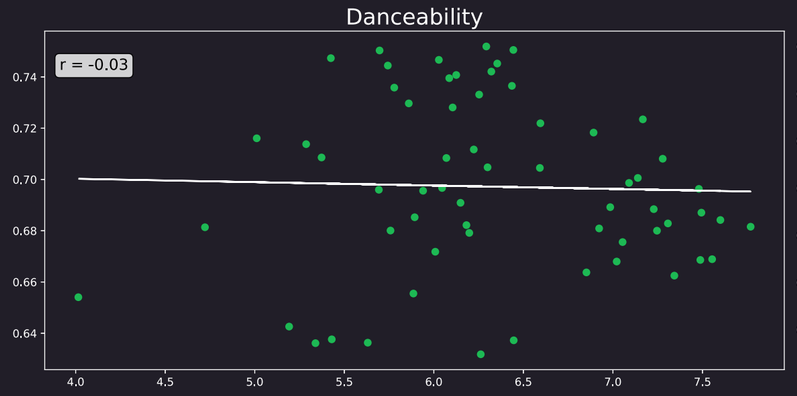
Audio Features Over Time

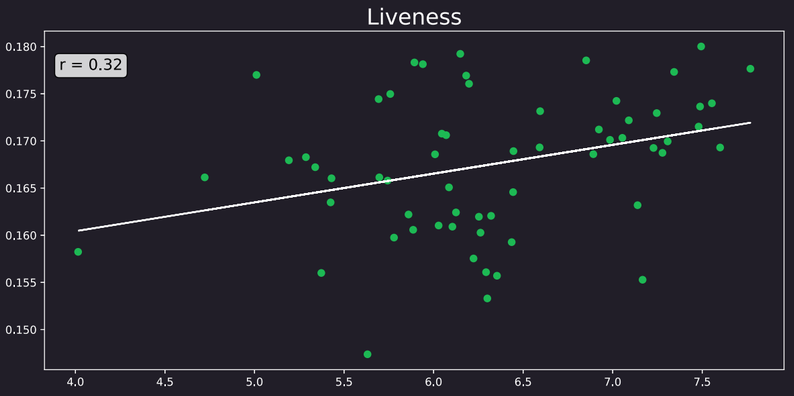


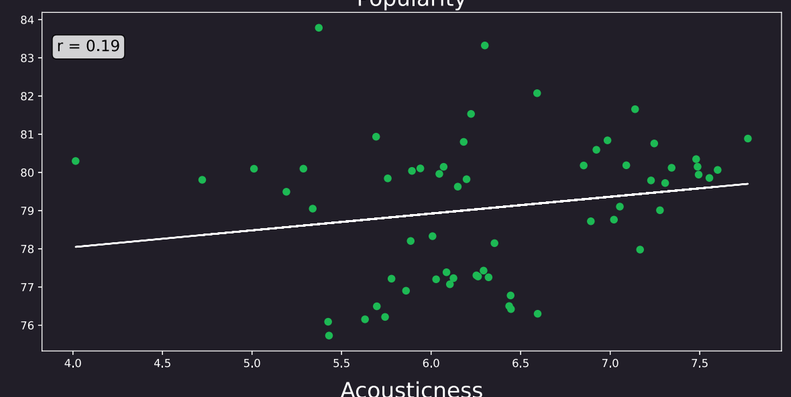
In 2019, do Audio Features of charting songs correlate to a country's Happiness Score?

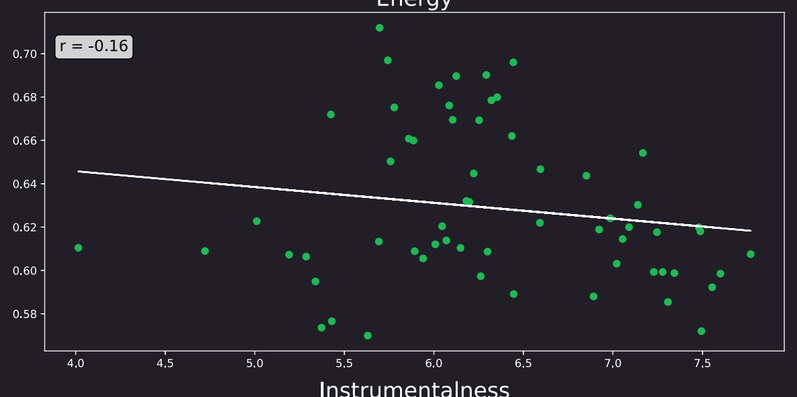
Audio Feature by Country Happiness Score

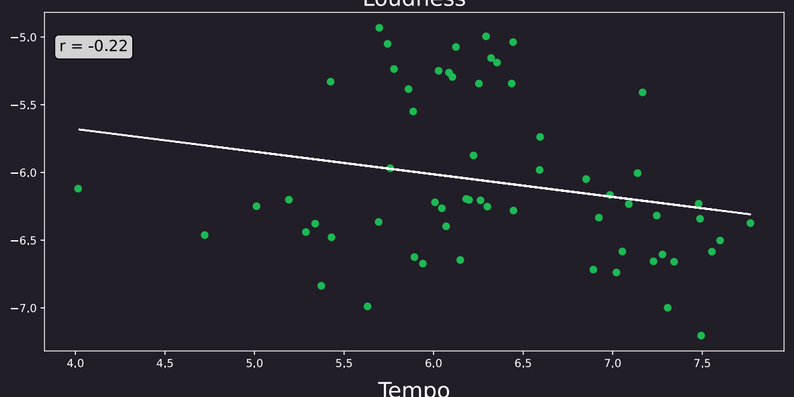


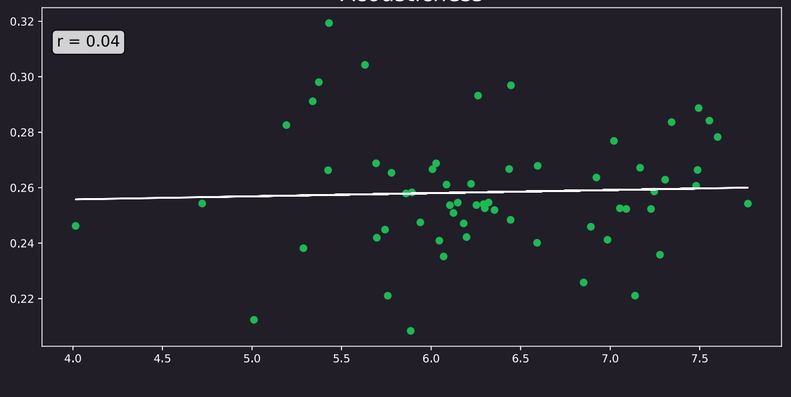


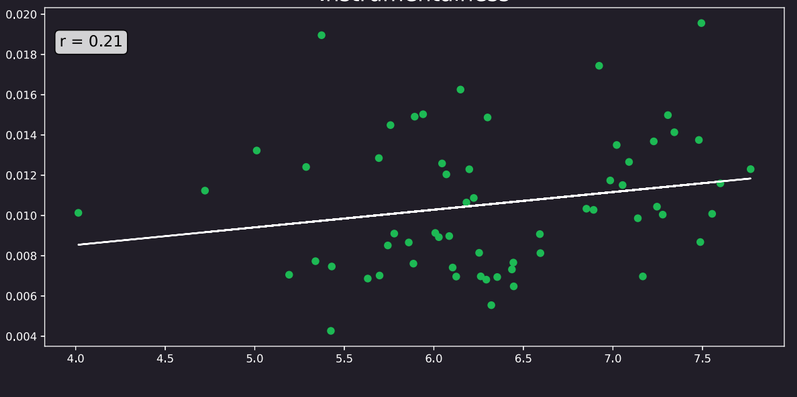


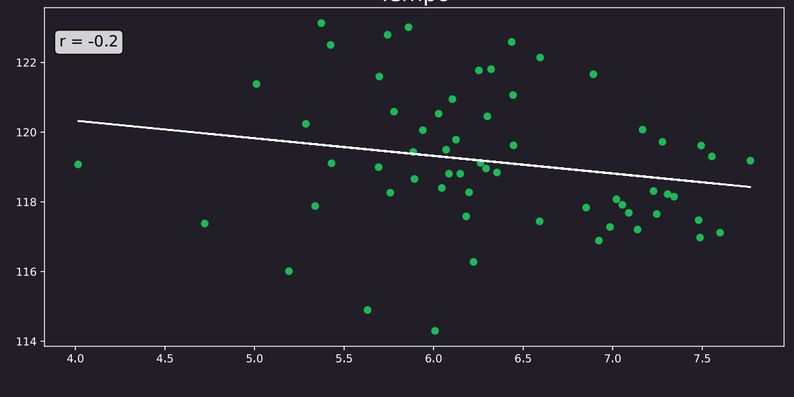








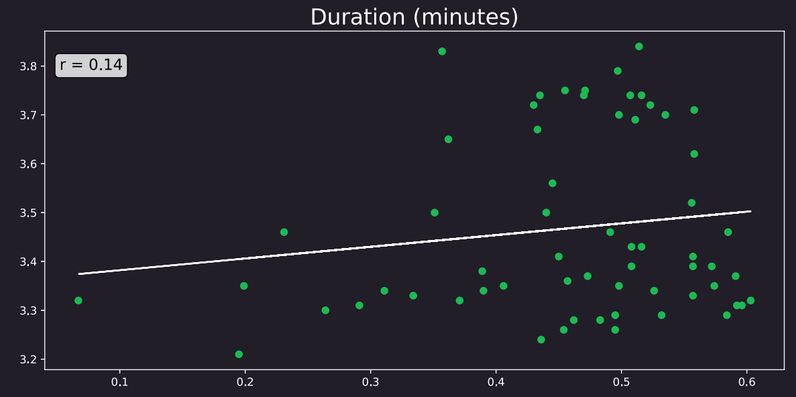


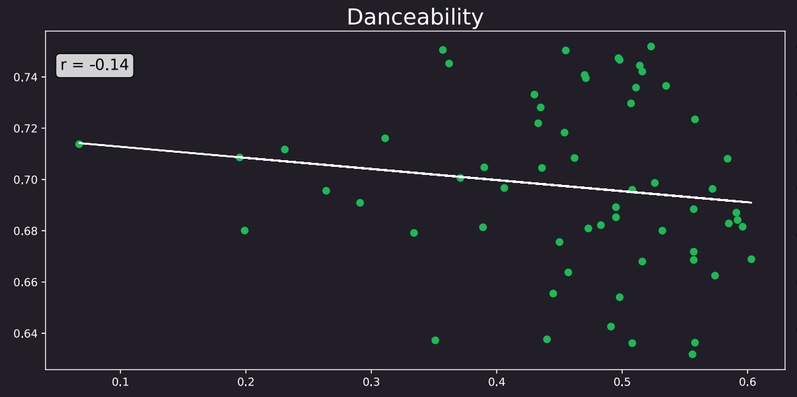


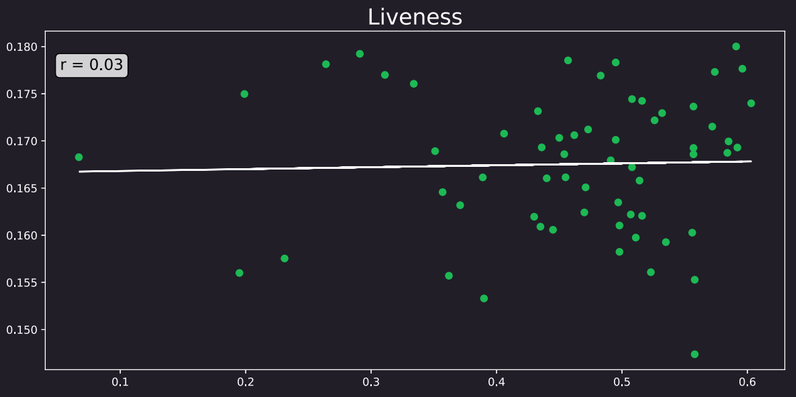
*based 2019 survey done by the Sustainable Development Solutions Network

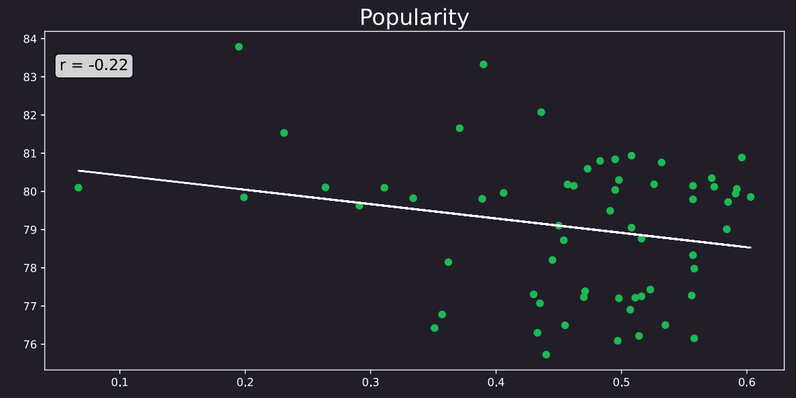
In 2019, do Audio Features of charting songs correlate to a country's Freedom to Make Life Choices Score?

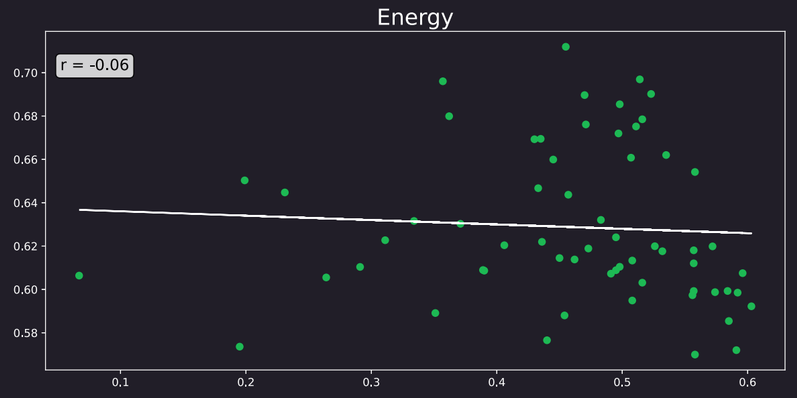
Audio Feature by Freedom to Make Life Choices Score

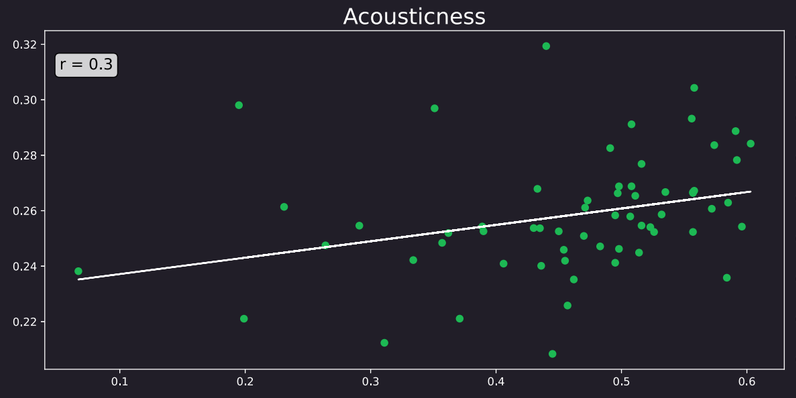


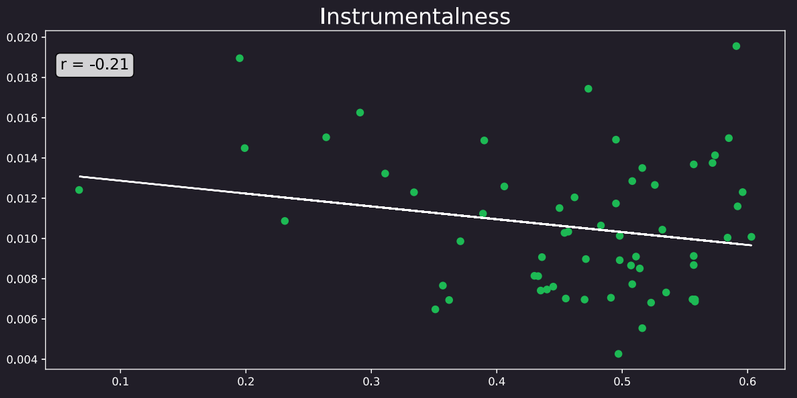


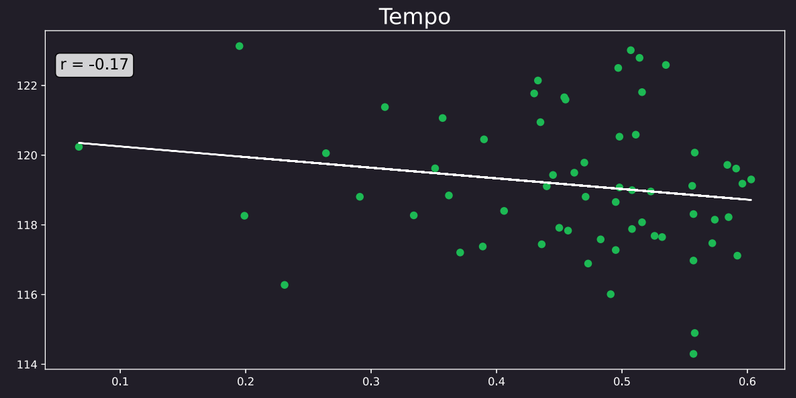








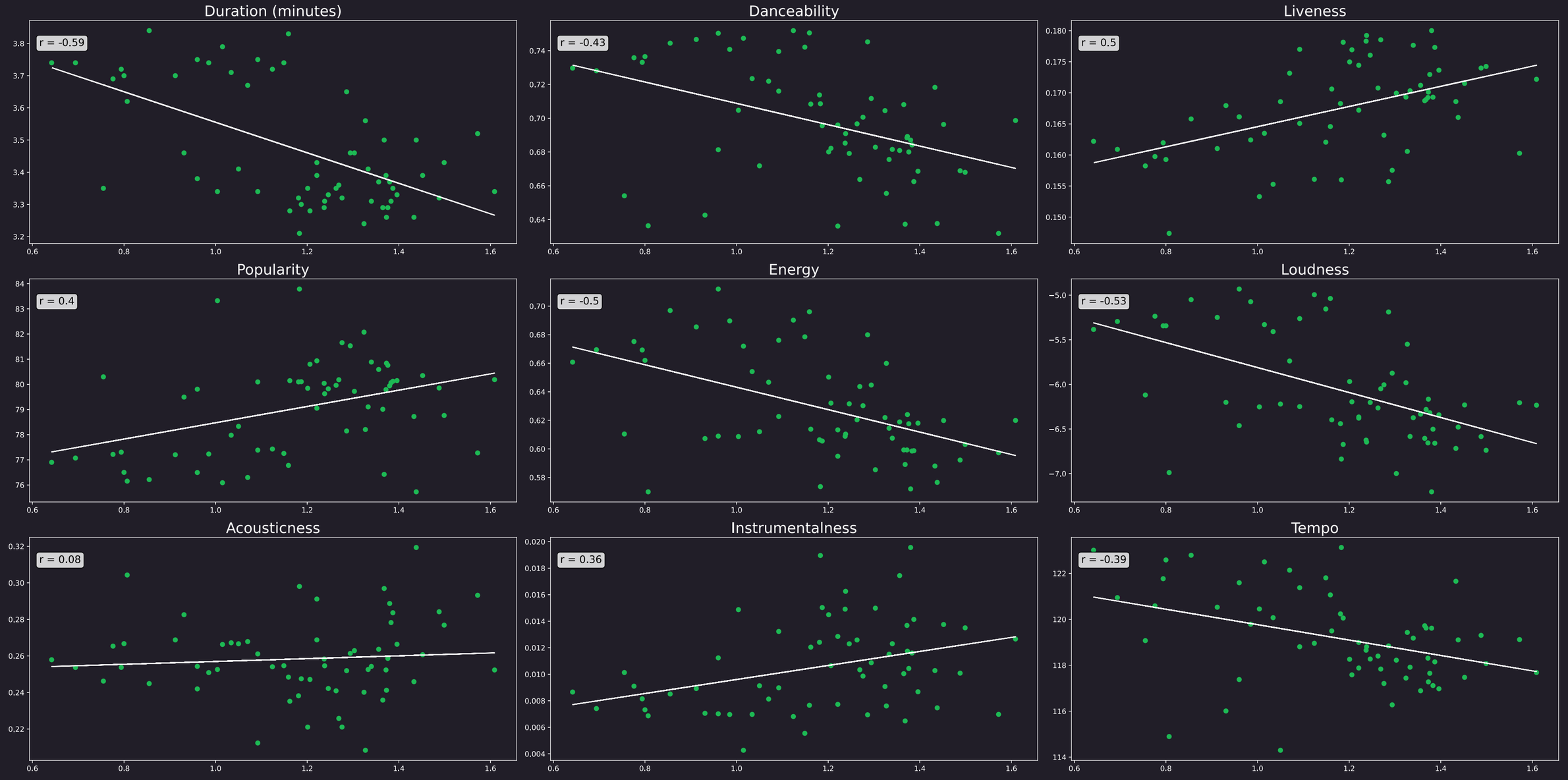




*based 2019 survey done by the Sustainable Development Solutions Network

In 2019, do Audio Features of charting songs correlate to a country's GDP per Capita?

Audio Feature by GDP Per Capita



*based 2019 survey done by the Sustainable Development Solutions Network

```
fig, axs = plt.subplots(3, 3)

for f in audio_features:
    x_values = df_region_metrics.happiness_score
    y_values = df_region_metrics[f]

    slope, intercept, r, p, stderr = stats.linregress(x_values, y_values)
    regress_values = (slope * x_values) + intercept

    axs[y][x].scatter(x_values, y_values, c='#1DB954')
    axs[y][x].plot(x_values,regress_values, color='white')
    axs[y][x].set_title(audio_features[f], fontdict={'fontsize':20})

    textstr = f'r = {round(r, 2)}'
    props = dict(boxstyle='round', facecolor='white', alpha=0.8)
    axs[y][x].text(0.02, 0.92, textstr,
                   transform=axs[y][x].transAxes,
                   fontsize=14,
                   verticalalignment='top',
                   bbox=props)

    axs[y][x].spines['bottom'].set_color('white')
    axs[y][x].spines['top'].set_color('white')
    axs[y][x].spines['right'].set_color('white')
    axs[y][x].spines['left'].set_color('white')
    axs[y][x].tick_params(axis='x', colors='white')
    axs[y][x].tick_params(axis='y', colors='white')
    axs[y][x].yaxis.label.set_color('white')
    axs[y][x].xaxis.label.set_color('white')
    axs[y][x].title.set_color('white')
    axs[y][x].patch.set_alpha(0)
```

Creating the subplots

- Loop through each audio feature
- Perform linear regression calculation
- Plot scatter points and regression line
- Display R Value
- Format colors of the axes and figure

What impact does the death of a Musician have on their streaming listenership?

```
#Merge the two csv files for formatting
merged_df = pd.merge(Deaths_df, US_Daily_2017_df, how="left", on=["Artist", "Artist"])
merged_df
```

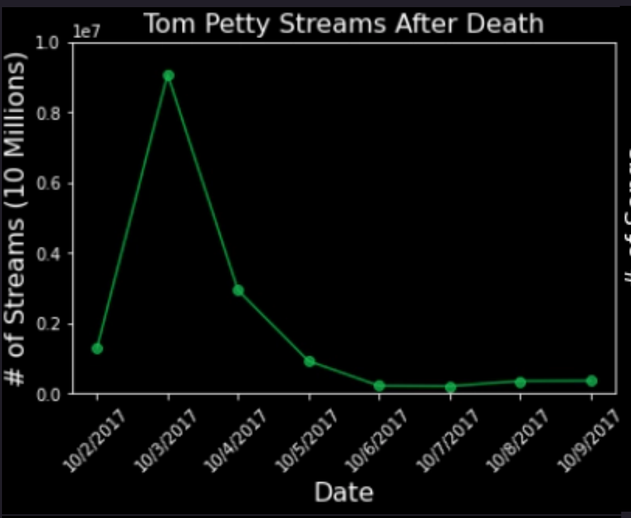
	Artist	Name	Age	Death	Date	Position	Track Name	Streams	ID
0	Manic Hispanic	Mike Gaborno	51	1/4/2017	NaN	NaN	NaN	NaN	NaN
1	Fanny Adams	Johnny Dick	73	1/6/2017	NaN	NaN	NaN	NaN	NaN
2	The Contours	Sylvester Potts	78	1/6/2017	NaN	NaN	NaN	NaN	NaN
3	Sons of Hawaii	Eddie Kamae	89	1/7/2017	NaN	NaN	NaN	NaN	NaN
4	Czerwone Gitary	Jerzy Kossela	74	1/7/2017	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...
437	Sanctuary	Warrel Dane	56	12/13/2017	NaN	NaN	NaN	NaN	NaN
438	Michael Prophet	Michael Prophet	60	12/16/2017	NaN	NaN	NaN	NaN	NaN
439	Richard Dobson	Richard Dobson	75	12/16/2017	NaN	NaN	NaN	NaN	NaN
440	Kevin Mahogany	Kevin Mahogany	59	12/17/2017	NaN	NaN	NaN	NaN	NaN
441	Shinee	Kim Jong-hyun	27	12/18/2017	NaN	NaN	NaN	NaN	NaN

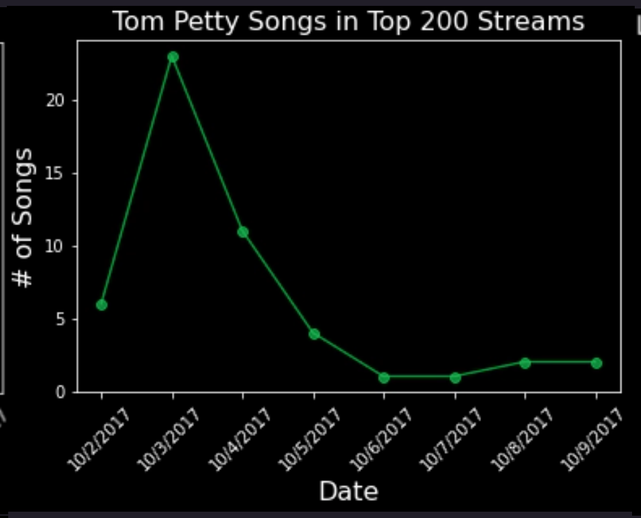
442 rows × 9 columns

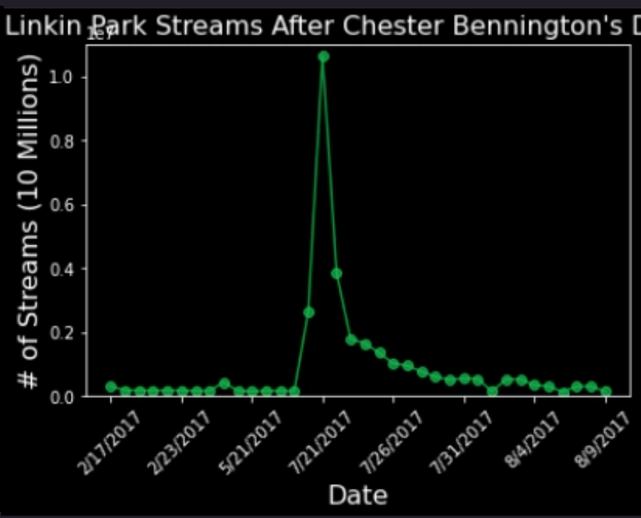
```
#Drop rows with inapplicable data
drop_df = merged_df.dropna()
drop_df
```

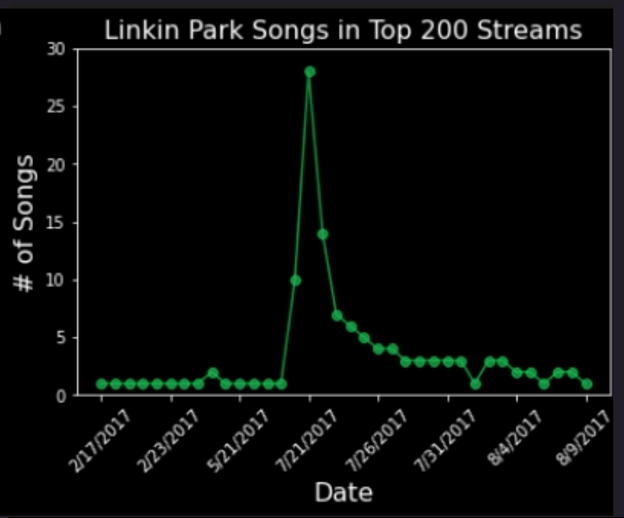
```
linkinpark = chester_bennington_df.groupby("Date")
linkinpark_streams = pd.DataFrame(linkinpark["Streams"].sum())
linkinparks = chester_bennington_df.groupby("Date")
linkinparks_streams = pd.DataFrame(linkinpark["Streams"].count())

linkinpark_chart = linkinpark_streams.plot(kind='line', marker="o", ylim=(0, 11000000)
alpha=0.75, rot=45, legend=False, color="#1DB954")
linkinpark_chart.set_title("Linkin Park Streams After Chester Bennington's Death",
fontsize=16)
linkinpark_chart.set_xlabel("Date", fontsize=16)
linkinpark_chart.set_ylabel("# of Streams (10 Millions)", fontsize=16)

linkinparks_chart = linkinparks_streams.plot(kind='line', marker="o", ylim=(0, 30),
alpha=0.75, rot=45,
legend=False, color="#1DB954")
linkinparks_chart.set_title("Linkin Park Songs in Top 200 Streams", fontsize=16)
linkinparks_chart.set_xlabel("Date", fontsize=16)
linkinparks_chart.set_ylabel("# of Songs", fontsize=16)

plt.style.use('dark_background')
plt.show()
```

Examining the Trends in Spotify Streams Following the Death of a Musician

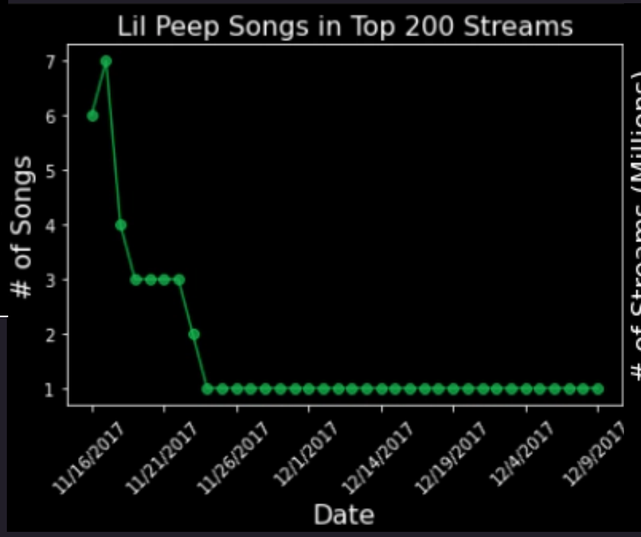


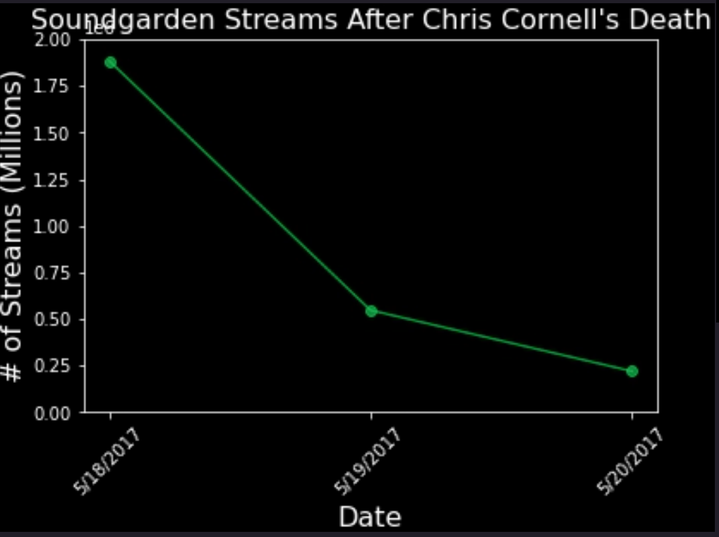


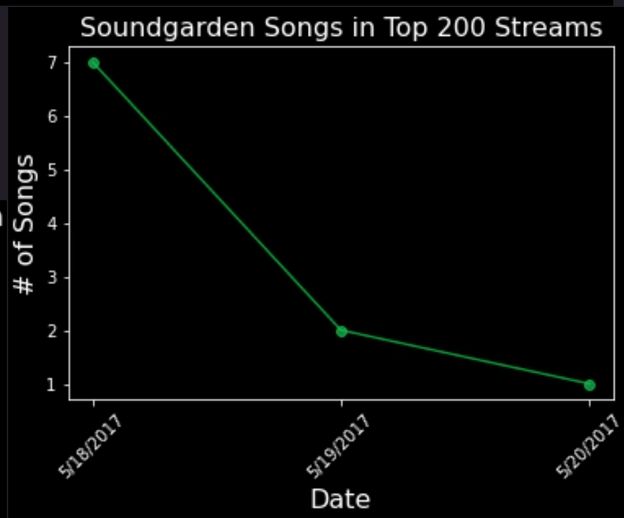




Four musicians showed significant data in the Spotify top 200 charts following their deaths.







```
chuck_berry_df = drop_columns_df.loc[drop_columns_df["Name"] == "Chuck Berry",:]
chuck_berry_df.head(12)
```

	Artist	Name	Death	Date	Track Name	Streams
55	Chuck Berry	Chuck Berry	3/18/2017	3/19/2017	Johnny B. Goode	166987.0
56	Chuck Berry	Chuck Berry	3/18/2017	11/24/2017	Run Rudolph Run - Single Version	177622.0
57	Chuck Berry	Chuck Berry	3/18/2017	11/25/2017	Run Rudolph Run - Single Version	197509.0
58	Chuck Berry	Chuck Berry	3/18/2017	11/26/2017	Run Rudolph Run - Single Version	213637.0
59	Chuck Berry	Chuck Berry	3/18/2017	11/27/2017	Run Rudolph Run - Single Version	215543.0
60	Chuck Berry	Chuck Berry	3/18/2017	11/28/2017	Run Rudolph Run - Single Version	204565.0
61	Chuck Berry	Chuck Berry	3/18/2017	11/29/2017	Run Rudolph Run - Single Version	191076.0
62	Chuck Berry	Chuck Berry	3/18/2017	11/30/2017	Run Rudolph Run - Single Version	211611.0
63	Chuck Berry	Chuck Berry	3/18/2017	12/1/2017	Run Rudolph Run - Single Version	225067.0
64	Chuck Berry	Chuck Berry	3/18/2017	12/2/2017	Run Rudolph Run - Single Version	259523.0
65	Chuck Berry	Chuck Berry	3/18/2017	12/3/2017	Run Rudolph Run - Single Version	259595.0
66	Chuck Berry	Chuck Berry	3/18/2017	12/4/2017	Run Rudolph Run - Single Version	264246.0

```
malcolm_young_df = drop_columns_df.loc[drop_columns_df["Name"] == "Malcolm Young",:]
malcolm_young_df
```

	Artist	Name	Death	Date	Track Name	Streams
417	AC/DC	Malcolm Young	11/18/2017	1/1/2017	You Shook Me All Night Long	178791.0
418	AC/DC	Malcolm Young	11/18/2017	10/31/2017	Highway to Hell	343613.0

Additional Questions:

- What songs show up on Spotify on certain days of the year/seasons.
- What is the significance in listenership following the death of a musician based on age or legacy?
- In which ways are the Spotify songs skewed? Are they more likely to include one song or genre over another?
- If we used the Billboard API how would correlating songs look over time?

Difficulties:

- Billboard charts API had a high fee and would not have a unique ID that we could use to merge with Spotify data
- Spotify Charts data could only be exported one day or week at a time so we scraped the data using Beautiful Soup
- Spotify only lists the number of streams for songs in the top 200, thus the total number of streams reflect only those particular tracks and not an artist's complete streaming total of all Spotify songs.

Data Sources

1. Spotify Audio Features: https://www.kaggle.com/yamaerenay/spotify-dataset-19212020-160k-tracks
2. Spotify Charts: https://spotifycharts.com/regional
3. World Metrics: https://www.kaggle.com/unsdsn/world-happiness
4. 2017 Musician Deaths: https://en.wikipedia.org/wiki/List_of_2017_deaths_in_rock_and_roll

Questions?