# STATS 3DA3

**Homework Assignment 1**

Lingyun Huang (400237999)

2024-01-25

## Question 1

We consider the open the surface weather dataset at noaa-gsod.csv

(a) The `gust` variable is Maximum wind gust reported for the day in knots to tenths. The `visib` variable is Mean visibility for the day in miles to tenths.

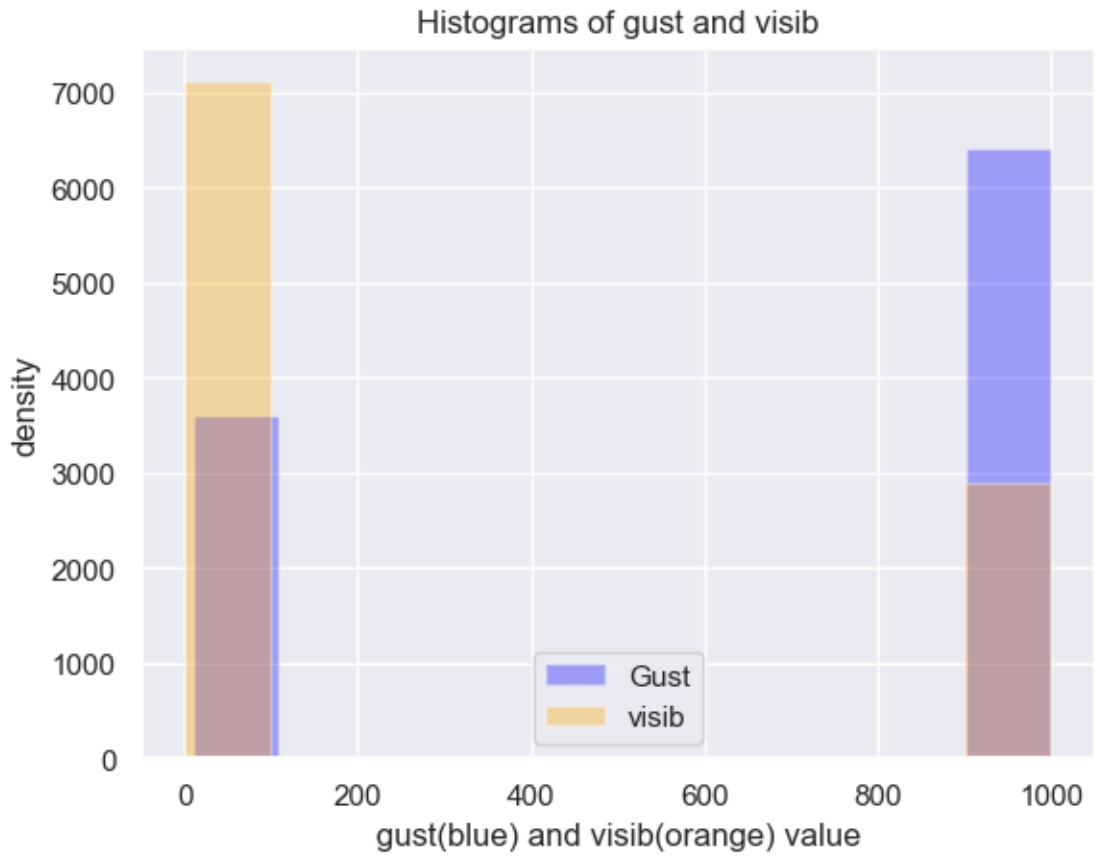
gust = Maximum wind gust reported for the day in knots to tenths. Missing = 999.9

visib = Mean visibility for the day in miles to tenths. Missing = 999.9

```
# make historgram of two variables that are continuous.
# code....
import pandas as pd
import matplotlib.pyplot as plt
import seaborn; seaborn.set()
import numpy as np
import seaborn as sns
df = pd.read_csv("https://gist.githubusercontent.com/krisrs1128/3845514e2d5eef57ec3271ea20fdcd
df
```

| | stn | wban | year | mo | da | temp | count_temp | dewp | count_dewp | slp | ... | flag_min |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 726115 | 54740 | 2019 | 1 | 24 | 34.4 | 24 | 9999.9 | 0 | 1004.5 | ... | * |
| 1 | 994430 | 99999 | 2019 | 3 | 15 | 76.6 | 10 | 9999.9 | 0 | 1019.9 | ... | * |
| 2 | 997996 | 99999 | 2019 | 1 | 23 | 35.9 | 17 | 9999.9 | 0 | 1009.8 | ... | * |
| 3 | 917540 | 99999 | 2019 | 7 | 21 | 82.5 | 11 | 9999.9 | 0 | 1009.1 | ... | NaN |
| 4 | 917540 | 99999 | 2019 | 8 | 1 | 82.4 | 12 | 9999.9 | 0 | 1011.0 | ... | NaN |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 9995 | 720407 | 462 | 2019 | 8 | 22 | 84.2 | 17 | 71.8 | 12 | 1012.9 | ... | * |
| 9996 | 171200 | 99999 | 2019 | 5 | 28 | 85.5 | 12 | 50.7 | 12 | 1010.5 | ... | NaN |
| 9997 | 150250 | 99999 | 2019 | 3 | 13 | 30.8 | 12 | 23.3 | 12 | 1019.6 | ... | * |
| 9998 | 722910 | 93116 | 2019 | 8 | 12 | 64.7 | 12 | 56.2 | 12 | 1014.8 | ... | * |
| 9999 | 868080 | 99999 | 2019 | 1 | 8 | 84.3 | 12 | 72.3 | 12 | 1012.4 | ... | NaN |

```
# plt.hist()
gust = np.array(df['gust'])
visib = np.array(df['visib'])
plt.hist(gust,color = 'blue', alpha = 1/3, label='Gust')
plt.hist(visib, color = 'orange', alpha = 1/3, label='visib')
plt.title('Histograms of gust and visib')
plt.xlabel('gust(blue) and visib(orange) value')
plt.ylabel('density')
plt.legend()
plt.show()
```




I have noticed that some values for the variables 'gust' and 'visib' are not normal, as they appear to be extremely large. From the introduction of dataset, they are supposed to be missing values.

(b) Next, we identify missing values in the the `gust` and `visib` variables using for loop and replacing them with explicit NaN missing values

for `gust`, missing value = 999.9

for `visib`, missing value = 999.9

```
# code for identifying the missing values
for i in range(len(gust)):
    if gust[i] == 999.9:
        gust[i] = np.nan #replace missing value with Nan

gust
```

```
array([17.1,  nan,  nan, ...,  9.7, 24.1, 18.1])
```

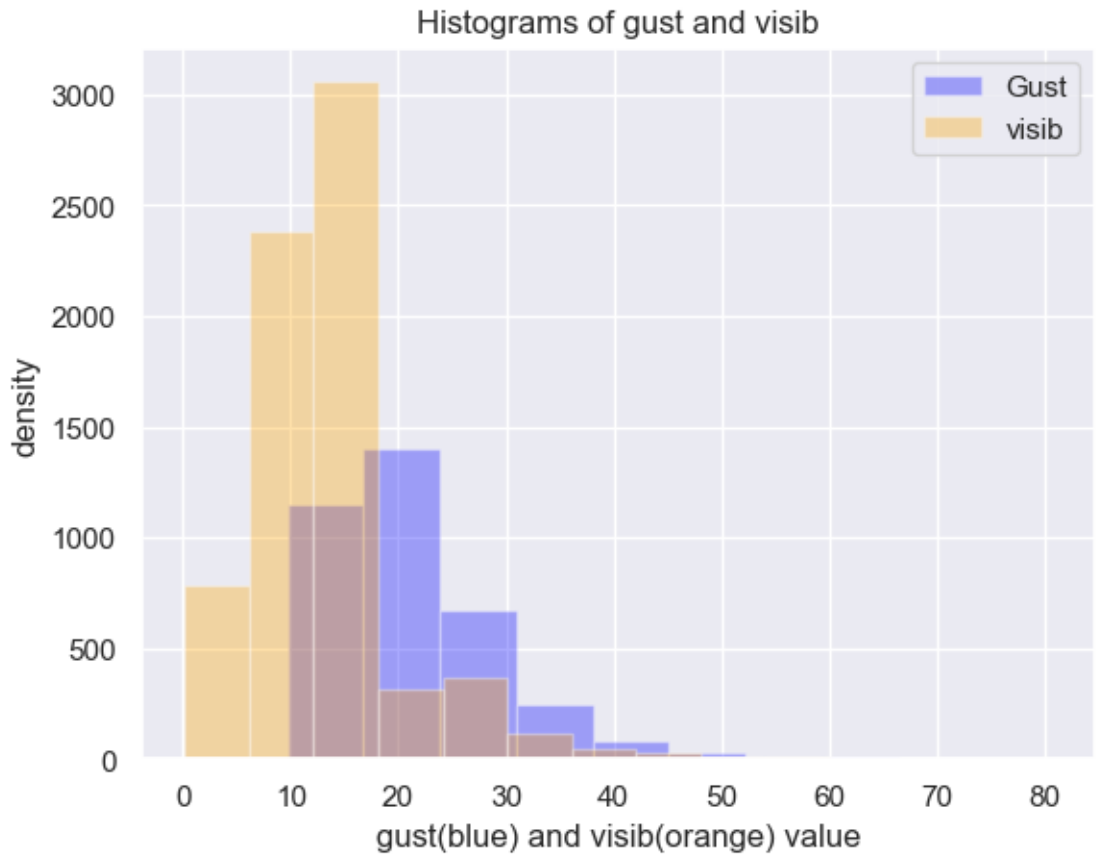
```
# code for identifying the missing values
for i in range(len(visib)):
    if visib[i] == 999.9:
        visib[i] = np.nan #replace missing value with Nan

visib
```

```
array([3. , nan, nan, ..., 3.2, 5.8, nan])
```

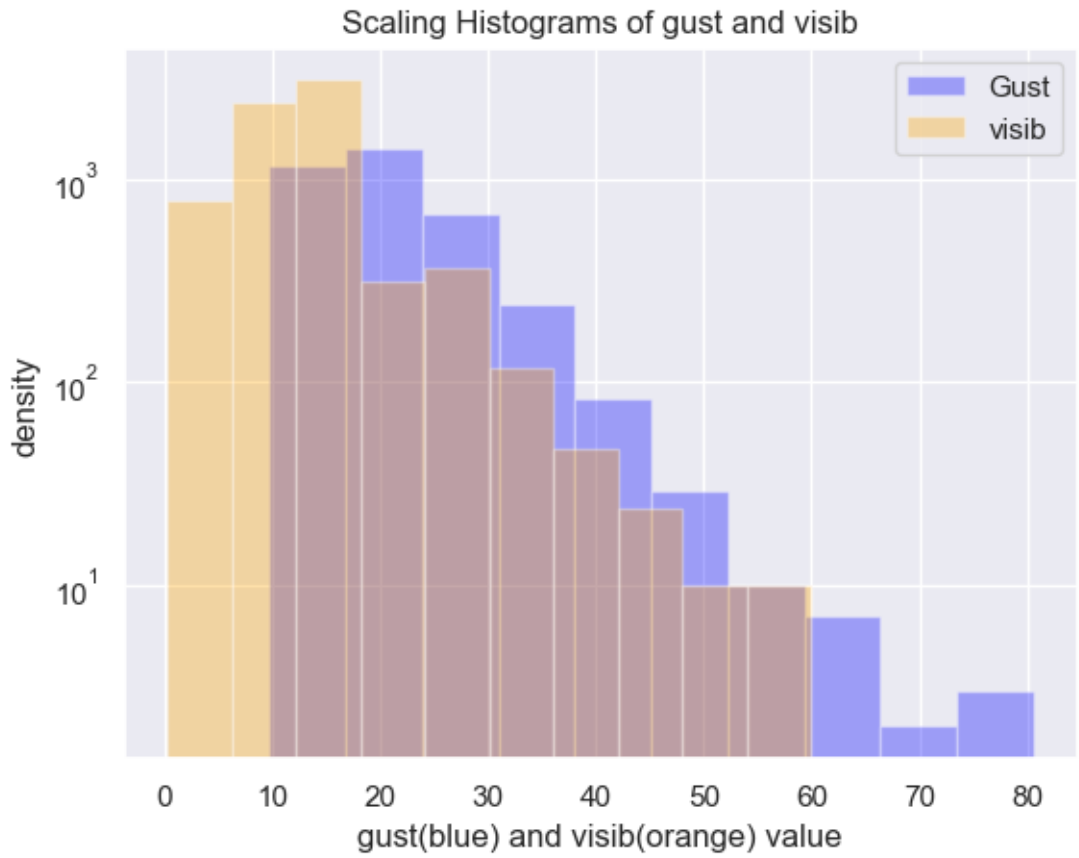
(c) The histograms of the `gust` and `visib` variables again, ignoring any missing values.

```
# code for histogram (I didn't ask it, but students must choose
# an approporate scaling for the y-axis)
plt.hist(gust,color = 'blue',alpha=1/3, label='Gust')
plt.hist(visib, color = 'orange',alpha=1/3, label='visib')
plt.title('Histograms of gust and visib')
plt.xlabel('gust(blue) and visib(orange) value')
plt.legend()
plt.ylabel('density');
#plt.yscale('log')
```




scaling the histogram

```
plt.hist(gust,color = 'blue',alpha=1/3, label='Gust')
plt.hist(visib, color = 'orange',alpha=1/3, label='visib')
plt.title('Scaling Histograms of gust and visib')
plt.xlabel('gust(blue) and visib(orange) value')
plt.ylabel('density')
plt.legend()
plt.yscale('log')
```




The histograms depict the distribution of the `gust` and `visib` variables in the context of the surface weather dataset.

The histrogram shows that `gust` variable, is representing Maximum wind gust reported for the day, is around 10 to 80 knots. The `visib` variable, is representing Mean visibility for the day, shows range of value is around 0 to 60 miles.

The shape of these two variables are both left-skewed distribution, which is really obevious under y-axis scarling histrograms.

The peak of `gust` occurs at around 20 knots, which indicates that a significant number of days in the surface weather dataset experienced a Maximum wind gust of approximately 20 knots. The peak of `visib` occurs at around 10 to 20 miles, which reveals that a significant number of days in the surface weather dataset exhibit Mean visibility values concentrated in the 10 to 20 miles range.

## Question 2

We consider Spotify Tracks DB dataset from Kaggle for the following analysis.

(a) We find the dimension of the Spotify dataset.

```
# code
import pandas as pd
spotify = pd.read_csv('/Users/lingyunhuang/Desktop/3da_files/SpotifyFeatures.csv')
spotify
```

| | genre | artist_name | track_name | track_id |
|---|---|---|---|---|
| 0 | Movie | Henri Salvador | C'est beau de faire un Show | 0BRjO6ga9RKCKjfDqeFgW |
| 1 | Movie | Martin & les fées | Perdu d'avance (par Gad Elmaleh) | 0BjC1NfoEOOusryehmNud |
| 2 | Movie | Joseph Williams | Don't Let Me Be Lonely Tonight | 0CoSDzoNIKCRs124s9uTV |
| 3 | Movie | Henri Salvador | Dis-moi Monsieur Gordon Cooper | 0Gc6TVm52BwZD07Ki6tIv |
| 4 | Movie | Fabien Nataf | Ouverture | 0IuslXpMROHdEPvSl1fTQ |
| ... | ... | ... | ... | ... |
| 232720 | Soul | Slave | Son Of Slide | 2XGLdVl7lGeq8ksM6Al7jT |
| 232721 | Soul | Jr Thomas & The Volcanos | Burning Fire | 1qWZdkBl4UVPj9lK6HuuF |
| 232722 | Soul | Muddy Waters | (I'm Your) Hoochie Coochie Man | 2ziWXUmQLrXTiYjCg2fZ2 |
| 232723 | Soul | R.LUM.R | With My Words | 6EFsue2YbIG4Qkq8Zr9Rir |
| 232724 | Soul | Mint Condition | You Don't Have To Hurt No More | 34XO9RwPMKjbvRry54Qz |

```
spotify.shape
```

```
(232725, 18)
```

There are 232,725 observations (rows) and 18 variables (columns).

(b) `track_id` should be unique to each observation. Verify this condition and comment on the results. Duplicates could be included in the dataset if tracks were different in genre only. If there is any duplicate, how many duplicated `track_id`. Drop the duplicate tracks from the dataset for further analysis.

```
#b
# Use is.unique to check duplicated track_id
if spotify['track_id'].is_unique == False:
    duplicated = spotify['track_id'].duplicated().sum() # Use duplicated and sum() to find the
    print('number of duplicated track_id is ' + str(duplicated))
    spotify = spotify.drop_duplicates(['track_id']) # Use drop_duplicates() to drop the duplica
    print('after drop the duplicate track_id, the number of unique track_id is ' + str(len(spot
elif spotify['track_id'].is_unique == True:
    print('all track_id is unique')
```

```
number of duplicated track_id is 55951
after drop the duplicate track_id, the number of unique track_id is 176774
```

(c) For each `track_id`, there is `genre`, `artist_name`, `track_name`, `popularity`, and some features of the track `acousticness`, `danceability`, `duration in milliseconds`, `energy`, `instrumentalness`, `key`, `liveliness`, `loudness`, `mode`,`speechiness`, `tempo`, `time_signature` and `valence`. Comment on the type of the above variables.

```
#c
spotify.dtypes
```

```
genre               object
artist_name         object
track_name          object
track_id            object
popularity           int64
acousticness       float64
danceability       float64
duration_ms          int64
energy             float64
instrumentalness   float64
key                 object
```

```
liveness            float64
loudness            float64
mode                 object
speechiness         float64
tempo               float64
time_signature       object
valence             float64
dtype: object
```

comment: `genre`, `artist_name`, `track_id`, `key`, `mode`, `time_signature` are object data type, which means they are Categorical Variables.
`popularity` and `duration_ms` are int64 data type, they are Integers; `acousticness`, `danceability`, `energy`, `instrumentalness`, `liveness`, `loudness`, `speechiness`, `tempo`, `valence` are float64 data type, they are Float. For all Float and Integer data type, they are all Numerical Variables.

(d) How many different `genres` are in the data?

```
#d
diff_genres = spotify['genre'].nunique()
print('the number of different genres are ' + str(diff_genres))
```

```
the number of different genres are 27
```

(e) Compute the average popularity of each genre and report the five most popular genres. Choose all the tracks related to the five most popular genres. Then, use this subset of tracks for the questions (f) - (h).

```
ave_pop = spotify.groupby('genre').agg(
    popularity = ('popularity', 'mean')
)
print(ave_pop.sort_values(by = 'popularity'))
```

| genre | popularity |
|---|---|
| Children's Music | 4.245650 |
| A Capella | 9.302521 |
| Movie | 12.146629 |
| Opera | 13.335628 |
| Comedy | 21.320240 |
| Anime | 24.256184 |
| Ska | 27.441063 |
| Classical | 29.320170 |
| Soundtrack | 33.681958 |
| Blues | 33.683851 |
| World | 34.416029 |
| Reggae | 35.390123 |
| Reggaeton | 36.480290 |
| Electronic | 37.592305 |
| Jazz | 39.885060 |
| Soul | 43.491874 |
| Country | 44.248679 |
| R&B | 48.463665 |
| Folk | 49.671347 |
| Alternative | 50.257944 |
| Children’s Music | 52.302465 |
| Indie | 53.528933 |
| Dance | 57.351541 |
| Hip-Hop | 58.516660 |
| Rock | 58.767849 |
| Rap | 59.515797 |
| Pop | 67.064957 |

5 most popular: Pop, Rap, Rock, Hip-hop, Dance

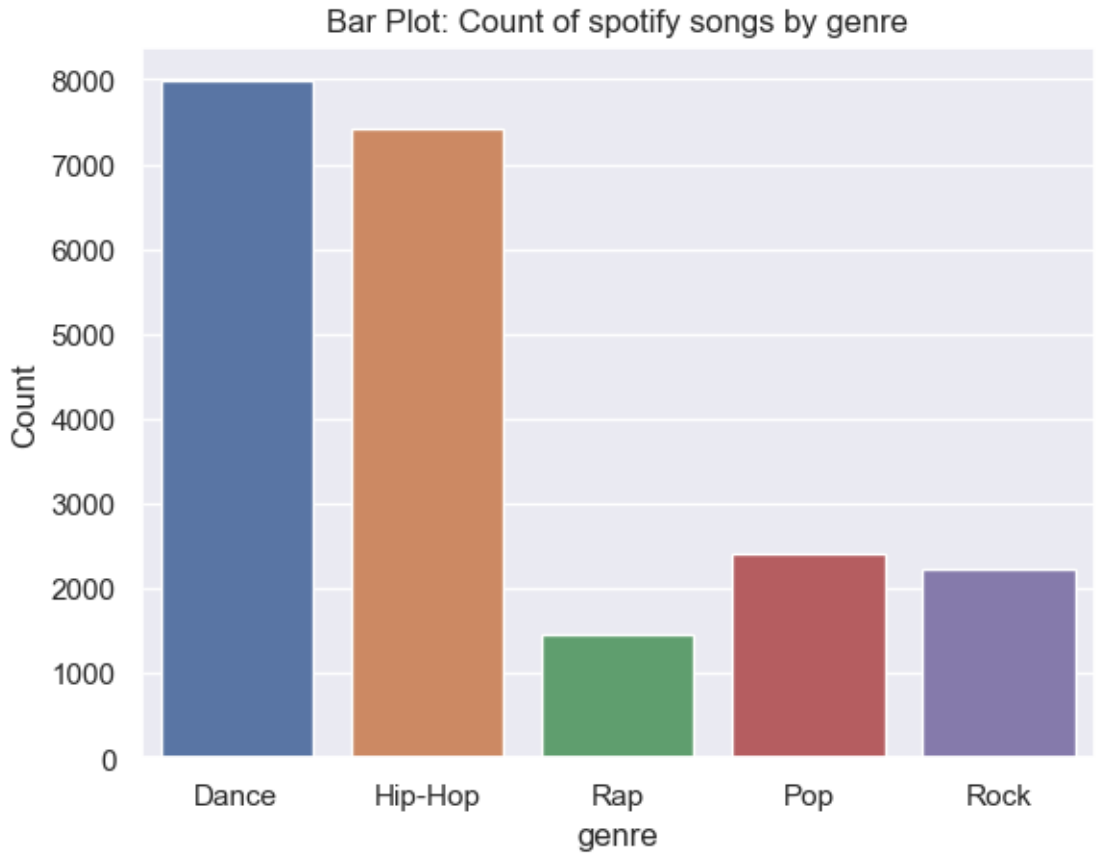
```
#create a subset
subset_5pop = spotify[(spotify['genre'] == 'Pop') |
                      (spotify['genre'] == 'Rap') |
                      (spotify['genre'] == 'Rock') |
                      (spotify['genre'] == 'Hip-Hop') |
                      (spotify['genre'] == 'Dance')
                     ]
subset_5pop
```

| | genre | artist_name | track_name | track_id |
|---|---|---|---|---|
| 9026 | Dance | Ariana Grande | break up with your girlfriend, i'm bored | 4kV4N9D1iKVxx1KL |
| 9027 | Dance | Ariana Grande | 7 rings | 14msK75pk3pA33pzP |
| 9028 | Dance | Halsey | Without Me | 5p7ujcrUXASCNwRa |
| 9029 | Dance | Ariana Grande | needy | 1TEL6MlSSVLSdhO |
| 9030 | Dance | Ariana Grande | NASA | 4uTvPEr01pjTbZgl7j |
| ... | ... | ... | ... | ... |
| 226488 | Rock | Fleetwood Mac | Little Lies | 08o75xMKmGrKny6 |
| 226493 | Rock | The Doors | Five To One | 5FlBGGwGuqYmqr0 |
| 226497 | Rock | Black Sabbath | The Wizard - Remastered Version | 6sjTzevtstOxOMsFsy |
| 226501 | Rock | The Beatles | I'm Happy Just To Dance With You - Remastered ... | 0gd50I2gKioJ59C827 |
| 226503 | Rock | The Beatles | No Reply - Remastered 2009 | 4ltC6PrqkTtpcRNi5l |

(f) Explore the distribution of `genre` using the appropriate visualization method and interpret the plot.

```
# Hint:
# 1. What is the data type of genre?
# 2. Choose the appropriate visualization method.
# 3. Interpret the plot. For example, which genre mostly appears in the dataset?
#plt.hist(subset_5pop['genre'])
#plt.title('distribution of genre')
#plt.xlabel('genre types')
```

```
#plt.ylabel('number')
sns.countplot(
    data=subset_5pop,
    x = 'genre'
)
plt.xlabel("genre")
plt.ylabel("Count")
plt.title("Bar Plot: Count of spotify songs by genre")
plt.show()
```




Data type of `genre` si object. From the plot, `Dance` mostly appears in the dataset.

(g) Explore the association between `genre` and `popularity` using the appropriate visualization method and interpret the plot. The plot must account for the different number of tracks in each `genre`.

```
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
```

```
num_track = subset_5pop['genre'].value_counts()
num_track.reset_index()
```

| | genre | count |
|---|---|---|
| 0 | Dance | 7982 |
| 1 | Hip-Hop | 7413 |
| 2 | Pop | 2417 |
| 3 | Rock | 2227 |
| 4 | Rap | 1456 |

```
relative_frequency = num_track/len(subset_5pop)
relative_frequency.reset_index()
```

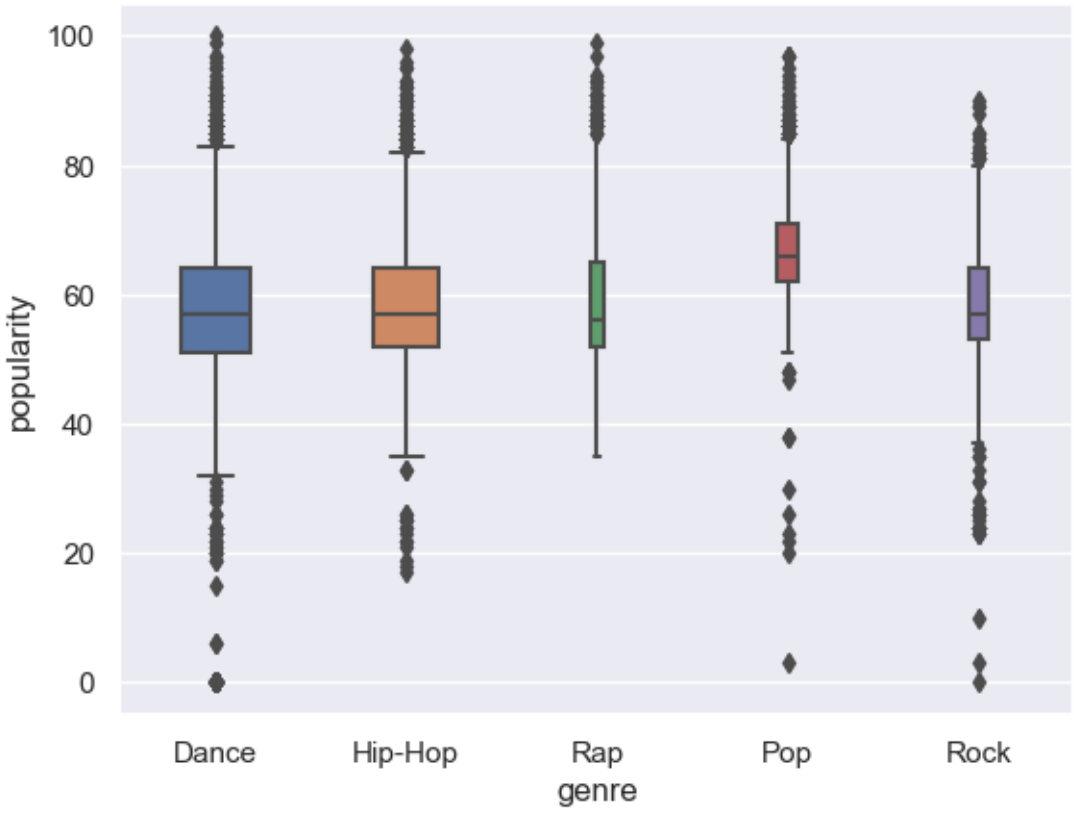
| | genre | count |
|---|---|---|
| 0 | Dance | 0.371342 |
| 1 | Hip-Hop | 0.344871 |
| 2 | Pop | 0.112445 |
| 3 | Rock | 0.103605 |
| 4 | Rap | 0.067737 |

```
unique_classes = subset_5pop['genre'].unique()
print(unique_classes)
print(len(unique_classes))
```

```
['Dance' 'Hip-Hop' 'Rap' 'Pop' 'Rock']
5
```

```
#for i in enumerate(unique_classes):
    #subset = subset_5pop[subset_5pop['genre'] == i[1]]
    #sns.boxplot(
        #data=subset,
        #x=subset_5pop['genre'],
        #y='popularity',
        #width=relative_frequency[i[1]]
        #)
```

```
unique_classes = subset_5pop['genre'].unique()
for i in unique_classes:
    subset = subset_5pop[subset_5pop['genre'] == i]
    sns.boxplot(
        data=subset,
        x=subset_5pop['genre'],
        y='popularity',
        width=relative_frequency[i],
    )
```




`Dance` appears mostly within the top 5 genres, then `Dance` has the largest relative frequency, therefore in the plot it has the widest width.

`Rap` appears least within top 5 genres, therefore it has the finest width.

From the boxplot, it is clear that `Pop` has the highest average popularity, which means `Pop` is the most popular `genre`.

Also, `Rap` has the lowest average popularity, which means `Rap` is the least popular `genre`.

(h) Explore the relationship between `acousticness` and `popularity`. Use an appropriate visualization method to avoid overplotting for this large dataset. Interpret the association between `acousticness` and `popularity`.

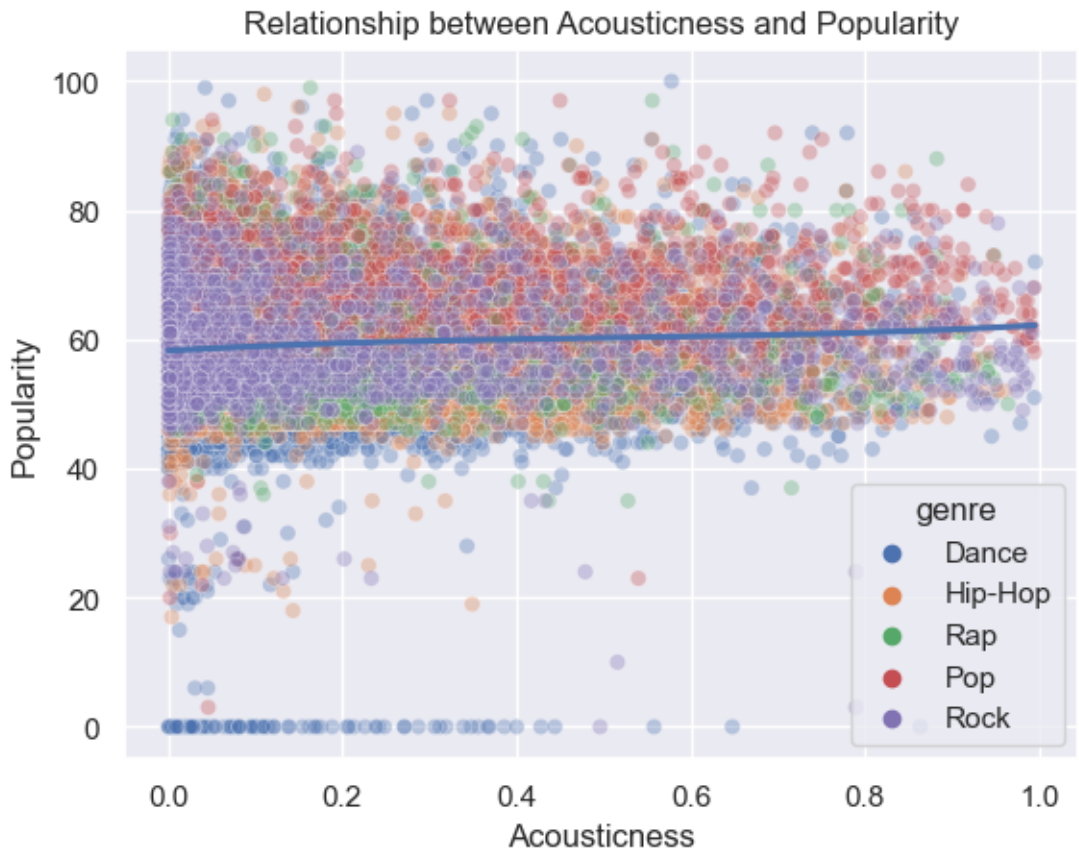
First let's use scatterplot and adjust alpha to see if it is overplotting.

If it is overplotting, we will use 2d histogram to solve it.

```
sns.scatterplot(data=subset_5pop, x='acousticness', y='popularity', hue='genre', alpha=1/3)
sns.regplot(data=subset_5pop, x='acousticness', y='popularity', scatter=False, ci=None, order =
plt.xlabel('Acousticness')
```

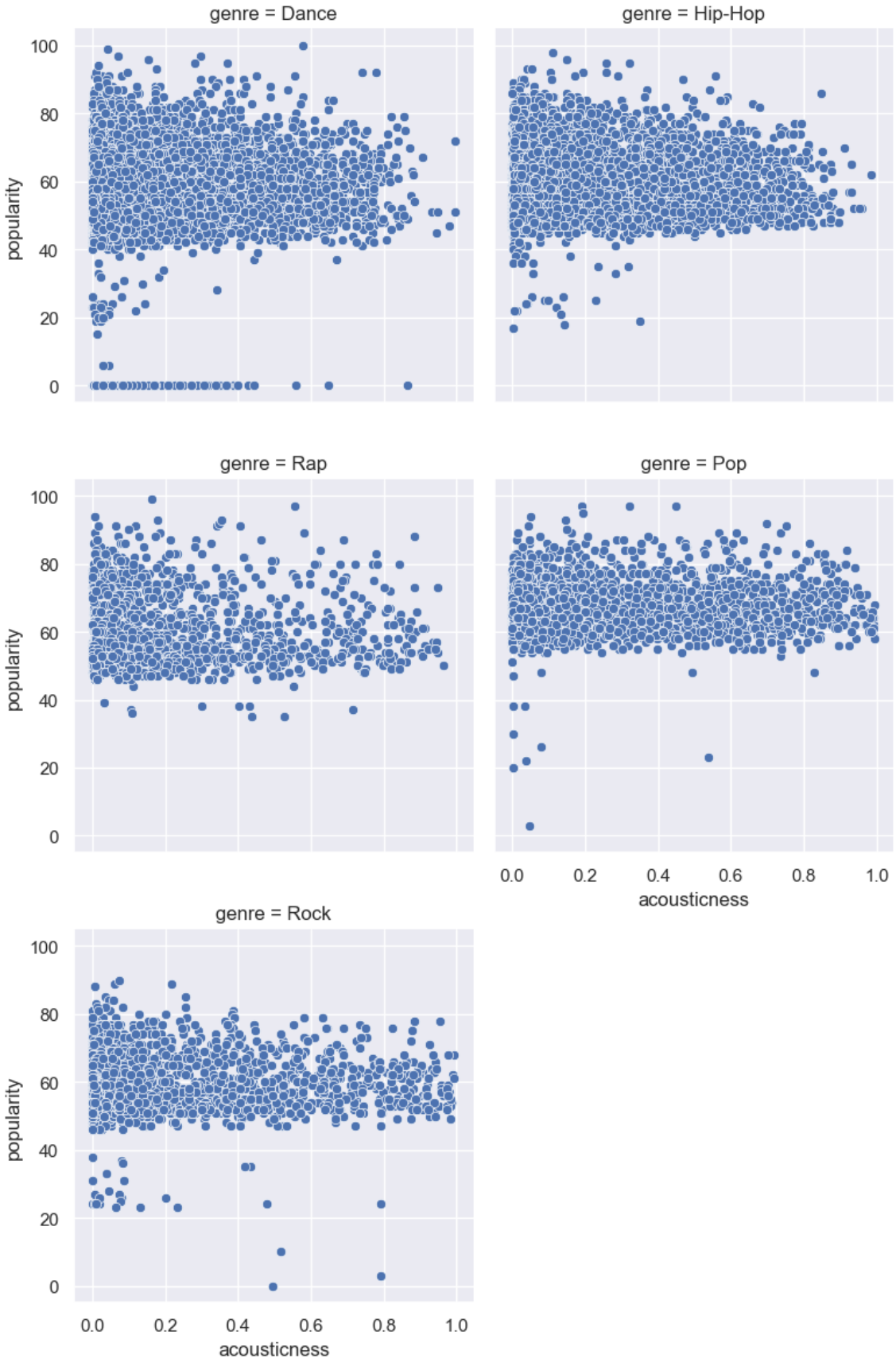
```
plt.ylabel('Popularity')
plt.title('Relationship between Acousticness and Popularity')
plt.show()


#2 d
```




```
g = sns.FacetGrid(
    subset_5pop,
    col="genre",# column facet variable
    col_wrap=2,
    height=4,
    sharex=True,
    sharey=True
    )
```

```
g.map(
    sns.scatterplot,
    'acousticness',
    'popularity'
    )
```

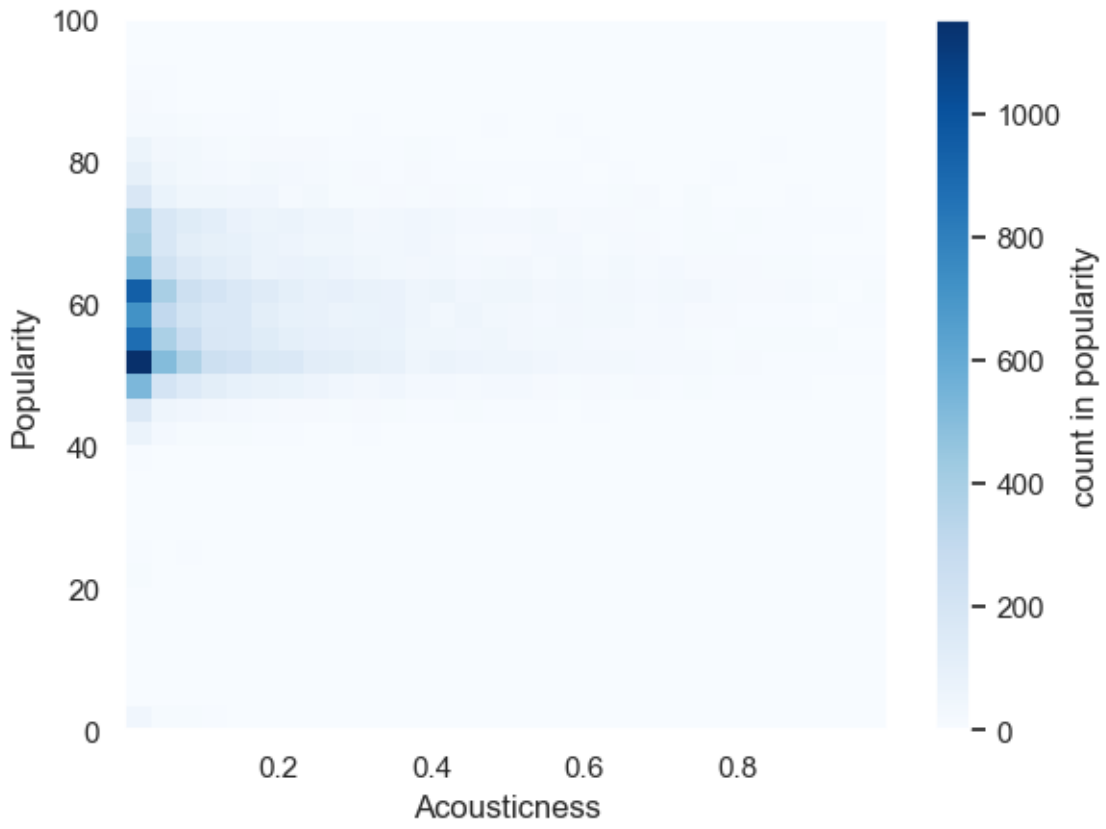

genre = Dance
genre = Hip-Hop
genre = Rap
genre = Pop
genre = Rock
popularity
acousticness
0
20
40
60
80
100
0.0
0.2
0.4
0.6
0.8
1.0

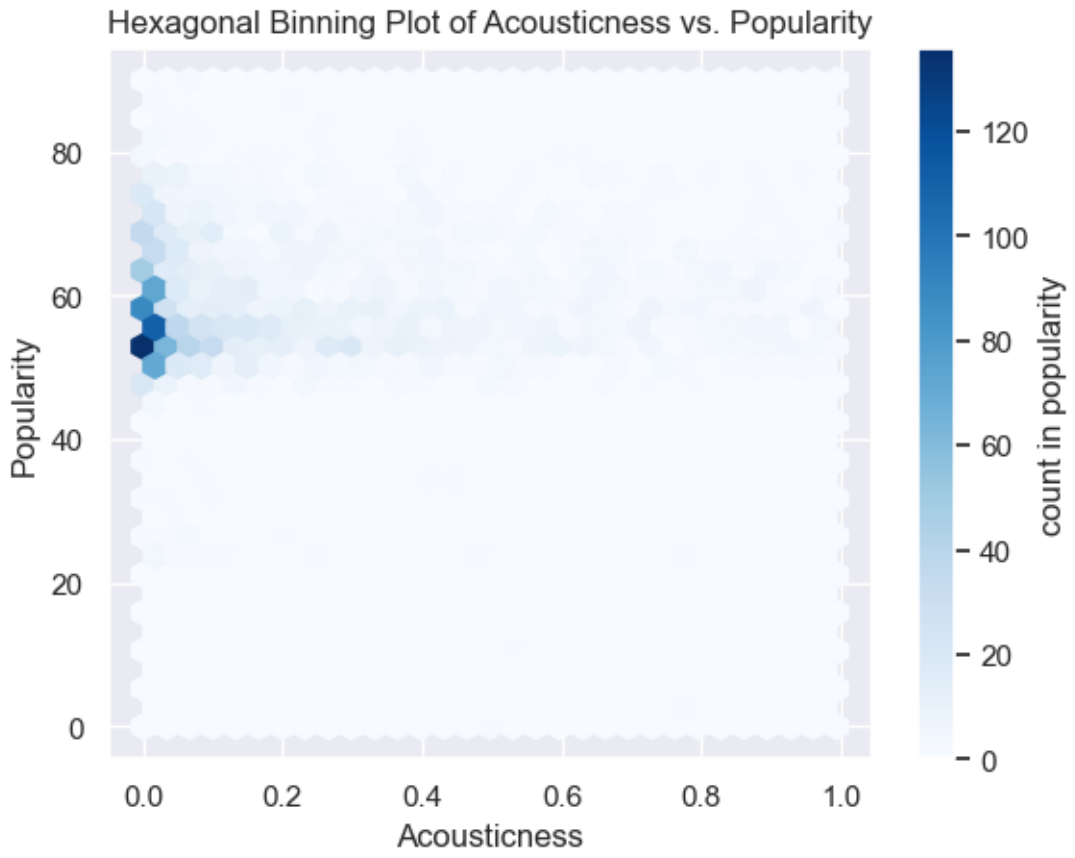
However these plots are not good.

To aviod overplotting for the large dataset, we use 2d histogram.

```
plt.hist2d(
    subset_5pop['acousticness'],
    subset_5pop['popularity'],
    bins=30,
    cmap='Blues'
    )
plt.colorbar(label='count in popularity')
plt.xlabel('Acousticness')
plt.ylabel('Popularity')
plt.show()

plt.hexbin(
    subset['acousticness'],
    subset['popularity'],
    gridsize=30,
    cmap='Blues'
    )
plt.colorbar(label='count in popularity')
plt.xlabel('Acousticness')
plt.ylabel('Popularity')
plt.title('Hexagonal Binning Plot of Acousticness vs. Popularity')
plt.show()
```


100
80
60
40
20
0
Popularity
0.2
0.4
0.6
0.8
Acousticness
1000
800
600
400
200
0
count in popularity




It appears that the `Popularity` does not change with the `Acousticness` increases; however, there are too few data with a high `Acousticness` to conclusively determine this trend. Hexbin plot shows there are too few `Acousticness` with $>.2$

## 3. Helper's name.

Xiangdong Wang

We discuss about the error when we were trying to plot the boxplot, the length of dataset was not fit the length of requency.

## References

- Lecture notebooks by Pratheepa Jeganathan
  - 01_tips_python_notebook_quarto
  - 01_data_viz_I
  - 01_data_viz_II