DS-UA 111 Final Project

Spotify Data: The Relationship Between Songs' Valence and Danceability

---

Part 1: Pre-step

1. Describe briefly the question you would like to answer or the topic you would like to explore. Essentially, what do you hope to learn from your analysis?

I enjoy listening to music and sometimes use it to stimulate certain emotions, such as excitement or sadness. A song's mood is a pretty arbitrary quality, so I thought of it as something you could only describe subjectively, with the description of any one song's mood varying slightly from person to person. However, I recently found out that Spotify has been determining quantitative measures of several seemingly indefinite song attributes based on more objective qualities. The main measure I was interested in is the one that pertains to the song's mood - the valence score, which quantifies how positive a song is. I wondered if the valence of a song could be related to any of the other vague traits that were quantified by Spotify. With this project, I would like to investigate if the mood, or positivity, of a song (quantified as valence) is related to any of its other arbitrary qualities, at least as defined by Spotify.

Part 2: Data

2. Find a dataset that may help you explore at least some of these questions.

a. Describe where you found the data set.

I found this dataset on **Kaggle**, an online community of data scientists where people can find datasets, publish their own, and share their analyses of either their datasets or others'. The original dataset can be downloaded from this site (https://www.kaggle.com/nadintamer/top-spotify-tracks-of-2018).

b. Describe how you found it.

I first visited the links to possible datasets for the project that were shared in lecture. There were many interesting topics, but I found it difficult to find datasets with continuous variables. After I stumbled upon some articles with analyses of Spotify data and became interested in the variable called valence, I searched Kaggle (a website which I found through the lecture links) for datasets of Spotify songs and data that I might be able to use. I was hoping to use a dataset that contained data for more recent songs (such as top songs of 2019), but I chose this one because the continuous variables were more precise compared to that of other similar datasets made available on Kaggle.

c. Describe at least two variables in the dataset that are relevant to the analysis you described above.

The explanatory variable I chose is **valence**. Measured from 0.0 to 1.0, it describes the musical "positiveness" that a song conveys. A high valence means that the song is more positive, and vice versa. Valence is the quality which is most similar to the "mood" of a song that I am most interested in.

I considered several variables to use as the response variable - the one that I chose is **danceability**. This continuous variable describes how suitable a track is for dancing, based on factors such as tempo and beat strength. It is also measured from 0.0 to 1.0, with a high danceability value indicating that the song is more suitable for dancing, and vice versa. In general, danceability is a vague song quality that you typically would not describe with a number, which is the kind of quality I was hoping to explore.

Other potential response variables that I considered were energy, which describes the intensity of a song with a value between 0.0 and 1.0, and loudness measured in decibels. These are also variables that can be described in a subjective manner.

d. Describe the unit of observation (individual, city, etc.).

A single observation represents one **song**, specifically one of the 100 most popular songs on Spotify in 2018.

3. If you could change this dataset in one way to make it better for your analysis, what would that change be and how could it improve your analysis?

I would like to **increase the number of songs** contained in the dataset. There are only 100 observations in this dataset, reflecting a playlist of the top 100 songs on Spotify in the year of 2018. While 100 is not a small number of observations, it is not a very large one either, considering just how many songs are out there, as well as the capabilities of Python data organization to sort through much larger amounts of data in a short time. Simply increasing the number of songs would make the sample of songs contained in the dataset more representative of the population of all songs and thus the sample statistics more similar to that of the population, due to the Law of Large Numbers. This would then make it more likely that the results of my analysis and the conclusions I draw from it also apply to the population of all songs on Spotify or even in existence.

If the goal is to make the dataset contain a sample that is more representative of the population, I would also try to **randomize** the songs that are included **based on their genres**. The sample of songs in this dataset contains very specifically only the 100 most popular songs on Spotify in 2018, and many songs of a certain genre may have been popular during that one year. If the proportion of songs of a some genres is much greater than that of other genres, the sample will not be as representative of the population of all songs. By randomizing genres, the dataset will be more likely to be representative of all songs, thus also making it more likely that the conclusions made from analyzing this dataset can be extended to all songs.

4. Import the dataset into Jupyter using any method you like and show the first five observations. If you had to do any pre-work to get the data into an uploadable format please describe it briefly. (If you didn't, please say so as well.)

In [1]:
```
# Import all packages necessary for analysis
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import statsmodels.api as sm

# Import the dataset - uploaded to the same folder as this notebook
songs = pd.read_csv('top2018spotify.csv')

# Show the first five observations
songs.head()
```

Out[1]:

	id	name	artists	danceability	energy	key	loudness	mode
0	6DCZcSspjsKoFjzjrWoCd	God's Plan	Drake	0.754	0.449	7.0	-9.211	1.0
1	3ee8Jmje8o58CHK66QrVC	SAD!	XXXTENTACION	0.740	0.613	8.0	-4.880	1.0
2	0e7ipj03S05BNilyu5bRz	rockstar (feat. 21 Savage)	Post Malone	0.587	0.535	5.0	-6.090	0.0
3	3swc6WTsr7rl9DqQKQA55	Psycho (feat. Ty Dolla $ign)	Post Malone	0.739	0.559	8.0	-8.011	1.0
4	2G7V7zsVDxg1yRsu7Ew9R	In My Feelings	Drake	0.835	0.626	1.0	-5.833	1.0

I did not have to change anything to get the data into an uploadable format - the data I wanted to use was made available in the same .csv file.

Part 3: Initial Analysis

5. Conduct at least two different manipulations of your now-ready table that help you understand something of interest about the dataset (e.g., you might explore options like sort, shape, value counts, groupby, etc.). Why did you choose these two, and what have you learned? (Hint: You may need to do a bit work to get the data into a format that is usable for you – e.g., renaming columns, changing data types, etc. If any of this was necessary, show your code and briefly explain why you made these changes)

I firstly chose to create a new dataframe with a subset of the original data containing only the columns that represent my variables of interest (valence and danceability) and the columns for song name and artist. This allows me to focus on the variables that I would like to explore (since the dataset includes are many columns) without modifying the original dataset.

In [2]:
```
# The new dataframe is named after the variables of interest:
# val(ence) and dance(ability)
songs_val_dance = songs[['name','artists','valence','danceability']]
songs_val_dance.head()
```

Out[2]:

	name	artists	valence	danceability
0	God's Plan	Drake	0.357	0.754
1	SAD!	XXXTENTACION	0.473	0.740
2	rockstar (feat. 21 Savage)	Post Malone	0.140	0.587
3	Psycho (feat. Ty Dolla $ign)	Post Malone	0.439	0.739
4	In My Feelings	Drake	0.350	0.835

For my first manipulation, I used the `describe()` function for my to see the general range of the values in each of my variables of interest.

In [3]:
```
songs_val_dance['valence'].describe()
```

Out[3]:
```
count    100.000000
mean       0.484443
std        0.206145
min        0.079600
25%        0.341000
50%        0.470500
75%        0.641500
max        0.931000
Name: valence, dtype: float64
```

The songs in the dataset have valence values from 0.079600 to 0.931000. These values, along with the quartile values, indicate that there is a good range of positivity levels in the dataset. A low valence like the minimum value 0.079600 means the song is very negative, and a high valence like the maximum value 0.931000 can be attributed to a song that is very positive.

In [4]:
```
songs_val_dance['danceability'].describe()
```

Out[4]:
```
count    100.00000
mean       0.71646
std        0.13107
min        0.25800
25%        0.63550
50%        0.73300
75%        0.79825
max        0.96400
Name: danceability, dtype: float64
```

The songs in the dataset have danceability values from 0.25800 to 0.96400. The minimum value is fairly low, but the first quartile (25th percentile) is at a value of 0.63550, indicating that a majority of songs in the dataset are more suitable for dancing and fewer songs in the dataset are not as suitable for dancing.

The next manipulation I conducted involved the `sort_values()` function, which I used to observe the danceability values of the songs with the greatest and least valence values.

In [5]:
```
songs_val_dance.sort_values(by='valence', ascending=False)
```

Out[5]:

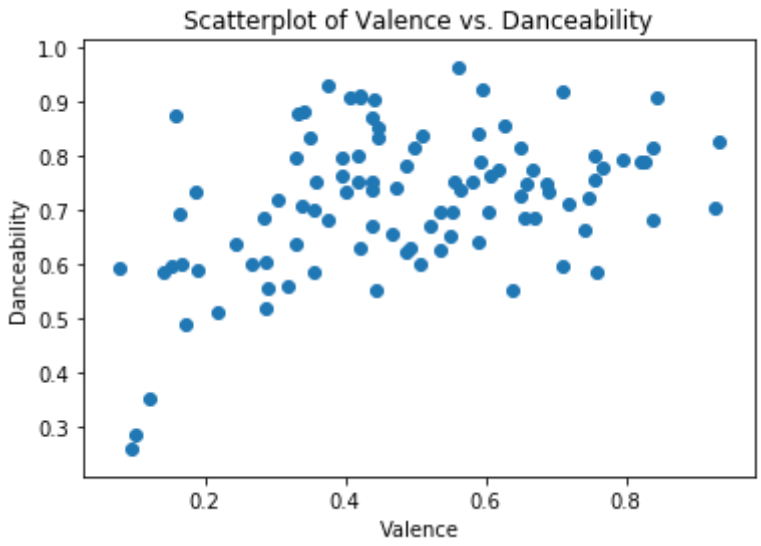
	name	artists	valence	danceability
25	Shape of You	Ed Sheeran	0.9310	0.825
46	Finesse (Remix) [feat. Cardi B]	Bruno Mars	0.9260	0.704
89	Bella	Wolfine	0.8440	0.909
66	D?jala que vuelva (feat. Manuel Turizo)	Piso 21	0.8390	0.681
78	Criminal	Natti Natasha	0.8390	0.814
...	...	...	...	...
2	rockstar (feat. 21 Savage)	Post Malone	0.1400	0.587
80	lovely (with Khalid)	Billie Eilish	0.1200	0.351
93	This Is Me	Keala Settle	0.1000	0.284
98	Dusk Till Dawn - Radio Edit	ZAYN	0.0967	0.258
33	Nevermind	Dennis Lloyd	0.0796	0.592

100 rows × 4 columns

The observations have been sorted based on valence values, in decreasing order. The songs with the five highest valence values have danceability values of at least 0.681, while the songs with the five lowest valence values have danceability value of at most 0.592. Interestingly, the song with the highest danceability among the five songs with the lowest valence is the one that has lowest valence.

6. Generate two different types of graphs of any kind that are useful to you to better understand what you're interested in. They don't need to be formatted particularly beautifully, but you do need to use two different types of graphs (e.g., a bar chart and a scatterplot) and explain what you hoped to understand, why you chose these graphs, and whether they're useful in improving your understanding.

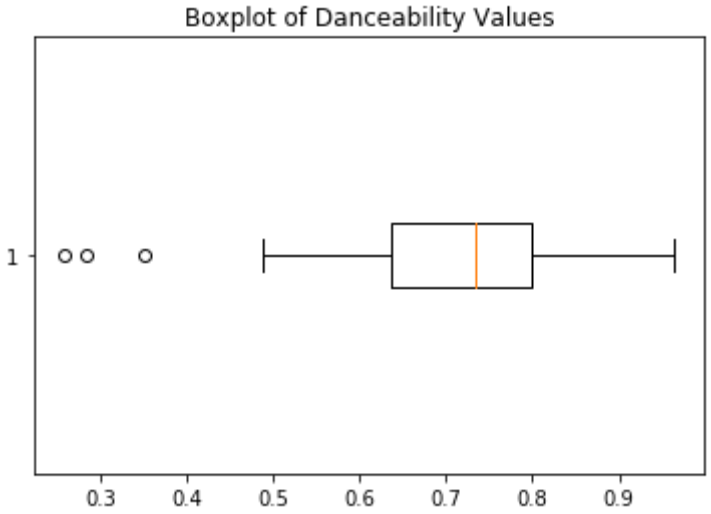
In [6]:
```
plt.scatter(songs_val_dance['valence'], songs_val_dance['danceability'])
plt.xlabel('Valence')
plt.ylabel('Danceability')
plt.title('Scatterplot of Valence vs. Danceability');
plt.show()
```



I used a **scatterplot** to see if there appears to be a relationship between valence and danceability. Roughly, there does appear to be a positive relationship, but it does not seem very strong. This helps me get closer to understanding what I want to learn from the analysis - whether or not there is a relationship between valence and danceability.

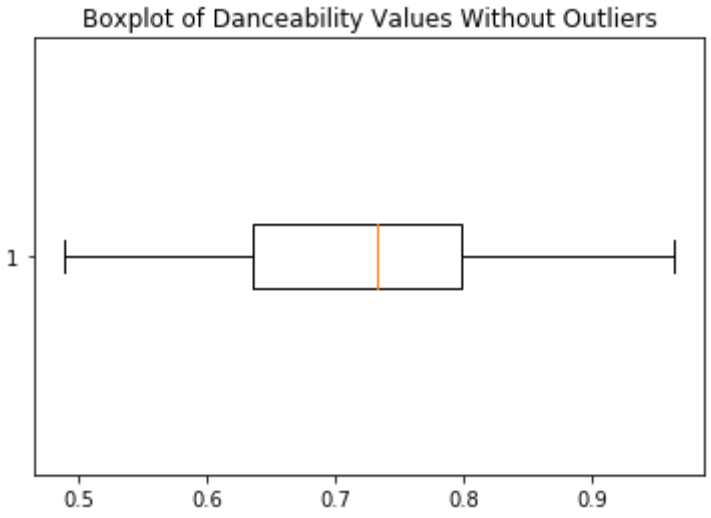
Additionally, the scatterplot shows, like the `describe()` function did for valence, that there is a good spread of valence values in this dataset. It also affirms what the `describe()` function showed for danceability - danceability values, for the most part, are higher than 0.5, and only a small number of songs have low danceability values that are less than 0.5. There appears to be three outliers, each with low valence and danceability values. If these outliers were eliminated, the relationship between valence and danceability would appear to be stronger.

In [7]:
```
plt.boxplot(songs_val_dance['danceability'], vert=False)
plt.title('Boxplot of Danceability Values');
```



I created a **boxplot** of the danceability values in the dataset to see if the three points in the scatterplot above are, in fact, outliers. The boxplot is helpful to my understanding, as it clearly confirms that the points are statistically considered outliers. The distribution of danceability values, not including the outliers, can be seen more clearly in the following boxplot:

In [8]:
```
plt.boxplot(songs_val_dance['danceability'], vert=False, showfliers=Fals
e)
plt.title('Boxplot of Danceability Values Without Outliers');
```



This boxplot shows that, without the outliers, the danceability values in the dataset are distributed pretty fairly within the upper half of possible danceability values (0.5 to 1.0). It shows, once again, that most songs in the dataset have higher danceability values, helping us to understand how representative the dataset may be of all songs. There are definitely songs out there that are not as suitable for dancing, so the top 100 Spotify songs in 2018 may not be the best sample to represent all Spotify songs. This is something to keep in mind as we continue with the analysis and begin to draw conclusions.

Part 4: Hypothesis Formation

7. What is your dependent variable and independent variable? Briefly describe how they are measured in this dataset. (Remember, they'll both need to be continuous variables.)

- My **dependent**, or response, variable is **danceability**. This is a measure of how suitable a song is for dancing, determined based on musical elements including the song's tempo, rhythm stability, beat strength, and overall regularity. Its values range between 0.0 to 1.0, and a high value means the song is more suitable for dancing.
- My **independent**, or explanatory, variable is **valence**. This is a measure of the musical "positiveness" that a song conveys. Its values range between 0.0 to 1.0, with a high valence meaning the song sounds more positive (as in happy or cheerful) and a low valence meaning it sounds more negative (as in sad, depressed, or angry). The measure was developed by Spotify using a set of agreed-upon ideas of what positive music sounds like.

8. Calculate the correlation coefficient between your two variables and interpret the result.

In [9]:
```
# Calculate and index correlation coefficient (r) using numpy
correlation_array = np.corrcoef(songs_val_dance['valence'], songs_val_da
nce['danceability'])
correlation = correlation_array[0][1]
print('The correlation coefficient between valence and danceability is',
correlation)
```

```
The correlation coefficient between valence and danceability is 0.41385
509151178385
```

The correlation coefficient is about **0.413855 or 41.3855%**. This value is quite low and is not close to 1, which would represent a perfectly linear positive relationship. The positive but low correlation indicates that the relationship between valence and danceability is positive (i.e. danceability increases as valence increases), but that the linear association between the two variables is fairly weak. Again, this value would likely increase and the linear relationship would be strengthened if outliers were removed.

9. Write out your regression model as an equation.

The regression model we will use is:

$$\text{danceability} = \beta_0 + \beta_1 * \text{valence}$$

where β_0 is the y-intercept of the regression line we are estimating (i.e. the expected danceability value if the valence value is 0.0) and β_1 is the slope of the regression line, or the regression coefficient (i.e. the expected average increase in the danceability value with each one-unit increase in the valence value).

10. Write out your null and alternative hypotheses.

- Null hypothesis (H_0): There is no linear relationship between valence and danceability. ($\beta_1 = 0$)
- Alternative hypothesis (H_a): There is a linear relationship between valence and danceability. ($\beta_1 \neq 0$)

Part 5: Regression Analysis

11. Estimate the regression equation you specified above and show the regression output.

In [10]:
```
# Use the statsmodels package to estimate the regression equation
x = songs_val_dance['valence']
y = songs_val_dance['danceability']

X = sm.add_constant(x) # Add an intercept to the indepedent variable
model = sm.OLS(y, X, missing = 'drop') # Construct a model
results = model.fit() # Fit the model

# Show the regression output
results.summary()
```

```
/opt/conda/envs/dsua-111/lib/python3.7/site-packages/numpy/core/fromnum
eric.py:2542: FutureWarning: Method .ptp is deprecated and will be remo
ved in a future version. Use numpy.ptp instead.
  return ptp(axis=axis, out=out, **kwargs)
```

Out[10]:

OLS Regression Results

Dep. Variable:	danceability	**R-squared:**	0.171
Model:	OLS	**Adj. R-squared:**	0.163
Method:	Least Squares	**F-statistic:**	20.25
Date:	Wed, 06 May 2020	**Prob (F-statistic):**	1.87e-05
Time:	17:22:56	**Log-Likelihood:**	71.204
No. Observations:	100	**AIC:**	-138.4
Df Residuals:	98	**BIC:**	-133.2
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.5890	0.031	19.148	0.000	0.528	0.650
valence	0.2631	0.058	4.500	0.000	0.147	0.379

Omnibus:	1.588	**Durbin-Watson:**	1.940
Prob(Omnibus):	0.452	**Jarque-Bera (JB):**	1.037
Skew:	-0.162	**Prob(JB):**	0.595
Kurtosis:	3.379	**Cond. No.**	6.06

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Using the results above and the regression model specified in question 9, the estimated regression equation is:

$$\text{danceability} = 0.5890 + 0.2631 * \text{valence}$$

12. What do the results in the regression output tell you? Interpret the coefficient, p-value, and confidence interval for your independent variable (you don't have to do the intercept) and the R^2.

According to the regression output results, the **regression coefficient** β_1, or the slope of the regression equation, is about **0.2631**. This means that with each one-unit increase in the valence value, we expect an average increase of 0.2631 in the danceability value.

The **p-value**, which is shown in the valence row under the P>|t| column in the regression output, is displayed to be 0.000. The p-value is so small that it is basically zero when rounded. The full p-value is:

In [11]:
```
p_value_sci = results.pvalues['valence']
p_value_round = '%f' % p_value_sci
print(p_value_sci, 'or', p_value_round)
```

```
1.8684859607637874e-05 or 0.000019
```

This means that there is approximately a **0.000019%** chance that we would see a test statistic as high as or higher than our test statistic β_1 (0.2631) by chance if the null hypothesis was true (i.e. if there was no linear relationship between valence and danceability).

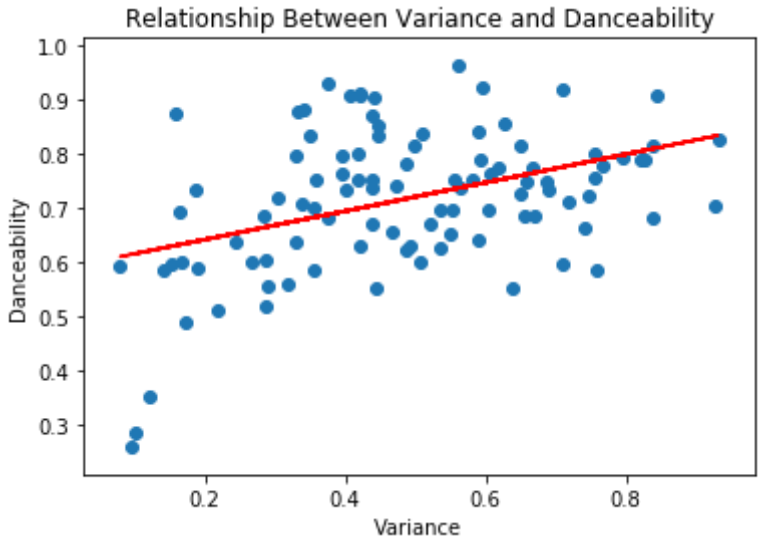
The **95% confidence interval** for the regression coefficient, which can be found in the valence row under the columns labeled [0.025 0.975] in the regression output, is **[0.147, 0.379]**. The 95% confidence interval means that if we sampled data for valence and danceability and infinite number of times ('n' times) , then 95% of those 'n' times, the true value of the regression coefficient for the population would be contained within the confidence interval. Interpreting the confidence interval found above using this definition, we can be 95% confident that the true value of the regression coefficient is contained within the interval from 0.147 and 0.379.

The R^2 value, found at the top right corner of the regression output, is shown to be **0.171**. This means that 17.1% of the variation in the danceability value can be explained by changes in the valence value. This is a low value, so if we do conclude that there is a linear relationship between valence and danceability, we will know that the relationship does not explain very much of the variation in the dependent variable (danceability).

We can graph the estimated regression equation over the scatterplot of the relationship between valence and danceability values:

In [12]:
```
# Create a scatterplot
plt.scatter(x,y)
plt.xlabel('Variance')
plt.ylabel('Danceability')
plt.title('Relationship Between Variance and Danceability')

# Draw the line for the regression equation
plt.plot(x, results.predict(X), color='red');
```



13. Which hypothesis do you reject and fail to reject, and why?

The p-value was found to be extremely low (0.000019). If the p-value cutoff is 5%, we can **reject the null hypothesis** since 0.000019 < 0.05. We can conclude that there is enough evidence to support the alternative hypothesis that there *is* a relationship between valence and danceability.

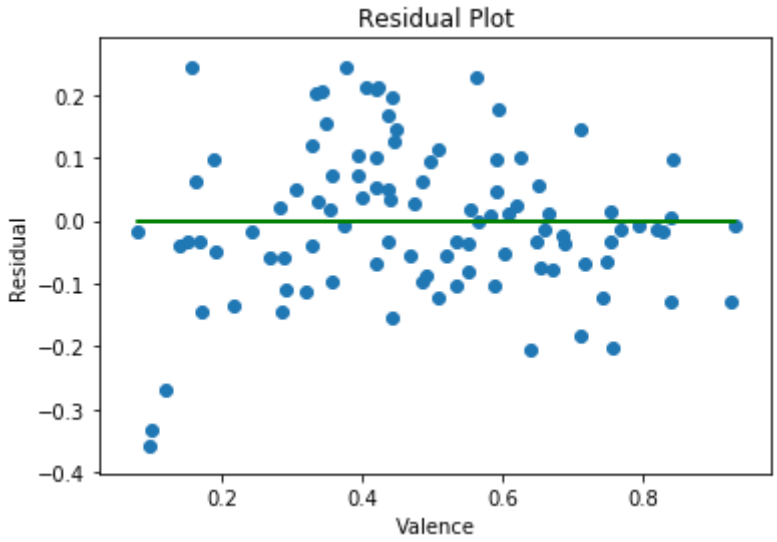
Using the 95% confidence interval also leads to the same conclusion. The null hypothesis states that the regression coefficient equals zero. Since 0 is not contained within the interval [0.147, 0.379], we can reject the null hypothesis.

14. Generate the residual plot and comment on any heteroskedasticity. What does this imply for your inference?

In [13]:
```
# Define residuals: observed - predicted
residuals = y - results.predict(X)

# Generate residual plot
plt.scatter(x, residuals)
plt.xlabel('Valence')
plt.ylabel('Residual')
plt.title('Residual Plot')

# Draw a line at y=0
plt.plot(x, [0]*len(x), color = 'green');
```



There appears to be no noticeable trend in the residual plot above, and the residuals seem to be scattered at random distances from the residual = 0 line. We can note that the outliers found in question 6 are once again noticeable here because they are the furthest away from the residual = 0 in the negative direction - the regression equation overestimates the danceability values for these observations. However, even with these outliers, there appears to be **little or no heteroskedasticity**, implying that the inference I made previously has not been weakened by a presence of changes in variance.

Part 6: Conclusions

15. What biases might be present in the sample itself that could be affecting the outcome? Discuss at least two sources of bias.

1. As previously mentioned, the songs in this dataset are not a random sample of songs on Spotify but specifically the top 100 songs on Spotify in 2018. The most popular songs, especially only during on particular year, are not necessarily representative of all songs on Spotify. Therefore the valence and danceability values found in this dataset/sample may also not be representative of that of all songs on Spotify, and the conclusion may not apply to songs on Spotify as a whole.
2. There may be a confounder related to the nature of the songs within this dataset. The songs are the *most popular* tracks on Spotify in 2018 - the songs' popularity may be related to them having some common quality unrelated to our variables of interest, such as energy or genre. Though we concluded that there is a linear relationship between valence and danceability, this does not mean one can necessarily be used to predict the other, as certain valence and danceability values may both be related to a different song attribute that we have not accounted for in the analysis.
3. In a way, there is a survivor bias in the sample itself, as the dataset excludes any songs that were not one of the 100 most popular tracks in 2018. The 101st most popular track in 2018 as well as the most popular song on Spotify in 2017 are not included.

16. Considering all the work you've done, including the regression output, the results of your hypothesis tests, and any biases present in the data, what conclusions, however tentative, can you draw from your analysis about the relationship between your two variables of interest?

The biases pertaining to the nature of the sample and the songs included in the dataset make it difficult to extend the solution to much beyond this sample. However, based on the regression output, especially the very low p-value, and the result of the hypothesis tests, we can conclude that there is a relationship between the variables valence and danceability, at the *very* least for these top 100 Spotify tracks of 2018. I do not want to assert causality, or that changes in valence cause or can be used to predict variation in danceability, due to the low R^2 value as well as the biases and weaknesses in my sample and analysis.

17. What is your analysis's greatest weakness? In other words, what are the best reasons to be cautious about what we can learn from it?

I have discussed the limitations that the sample itself presents for the applications of this analysis multiple times, such as in questions 3 and 15. Though I looked for and found a dataset that is very clean and that contains many potential variables to work with, I believe the greatest weakness of the analysis is still the dataset itself.

Relative to the population, this dataset actually contains a very small sample, given that there are probably millions of songs on Spotify and even more in existence. This makes it unlikely that it is representative of the population, as the Law of Large Numbers states that the larger the sample, the more similar to the population it is. The fact that the dataset was taken specifically from the top 100 songs in 2018 makes it even less likely that it is representative of the population; popular songs could be popular for certain reasons, and these reasons may not necessarily be orthogonal to the variables we have investigated.

The conclusion drawn from the hypothesis test is very strong, due to the small p-value that is close to zero. However, because of the size of the sample and where this dataset was drawn from (i.e. the specific set of observations that the dataset reflects), it is hard make any greater conclusion about the population or even a bigger sample based on the analysis of this sample. It is not unlikely that the results of this analysis gives a hint about the results when using a larger sample that is more representative of the population of all songs; the results may very well align. But we still must be very careful when extending what we have learned from this particular analysis to other analyses, if we really can at all.

In [ ]: