JazzAnalysis-Final-Copy1

July 14, 2019

```
In [1]: import pandas as pd
        import os
        import numpy as np
        import seaborn as sns
        import matplotlib.pyplot as plt
        import sklearn
        from sklearn import linear_model
        from sklearn import metrics
        from sklearn.model_selection import train_test_split
```

1 Introduction: Jazz

Jazz has been a major influence for various artists and even genres since its peak in the 1920s known as the Jazz age. Within every music genre there are various subgenres however, it seemed appropriate to analyze the overarching genre and proceed to break it down.

Our major motivation is to find what audio features make songs more or less popular according to spotify's popularity rating.

2 Data Wrangling

```
In [2]: ## Read in data and limit to Jazz songs
        totalData = pd.read_csv('SpotifyFeatures.csv')
        jazzData = totalData[totalData['genre'] == 'Jazz']
        jazzData.head()
```

```
Out[2]:        genre         artist_name           track_name                track_id  \
        177882  Jazz    Kelsea Ballerini         Miss Me More  5NfJGBAL9mgFPRQxKJmiX2
        177883  Jazz  Earth, Wind & Fire            September  5nNmj1cLH3r4aA4XDJ2bgY
        177884  Jazz      Leslie Odom Jr.  Alexander Hamilton  4TTV7EcfroSLWzXRY6gLv6
        177885  Jazz          Etta James              At Last  4Hhv2vrOTy89HFRcjU3QOx
        177886  Jazz      Leslie Odom Jr.          Wait for It  7EqpEBPOohgk7NnKvBGFWo

                popularity  acousticness  danceability  duration_ms  energy  \
        177882          74         0.014         0.643       192840   0.720
        177883          79         0.114         0.697       214827   0.809
        177884          72         0.524         0.609       236738   0.435
```

```
177885           74         0.707         0.171       182400   0.330
177886           69         0.125         0.561       193750   0.474

        instrumentalness key  liveness  loudness   mode  speechiness    tempo  \
177882          0.000000   D    0.0834    -7.146  Major       0.0527   96.028
177883          0.000521   A    0.1830    -8.197  Major       0.0302  125.941
177884          0.000000   B    0.1180    -7.862  Minor       0.2840  131.998
177885          0.003810   F    0.3020    -9.699  Major       0.0329  174.431
177886          0.000005  F#    0.0922    -9.638  Major       0.1550   86.897

       time_signature  valence
177882            4/4    0.491
177883            4/4    0.980
177884            4/4    0.563
177885            3/4    0.315
177886            4/4    0.513
```

In [3]:
```python
## Find basic information on Jazz dataset
def main():
    print("Number of observations :: ", len(jazzData.index))
    print("Number of columns :: ", len(jazzData.columns))
    print("Headers :: ", jazzData.columns.values)

if __name__ == "__main__":
    main()
```

```
Number of observations ::  9441
Number of columns ::  18
Headers ::  ['genre' 'artist_name' 'track_name' 'track_id' 'popularity' 'acousticness'
 'danceability' 'duration_ms' 'energy' 'instrumentalness' 'key' 'liveness'
 'loudness' 'mode' 'speechiness' 'tempo' 'time_signature' 'valence']
```

Given the above variety variables, we decided to only use numeric data. We then found that the duration of the songs was causing the variance to sky rocket so we decided to get rid of that feature and prodeed with our analysis. Additionally, given the large dataset, we will only analyze the songs with popularity higher than the median popularity.

In [4]:
```python
## Extracting our desired subset
features = ['popularity', 'acousticness', 'danceability', 'energy', ## Defining the de
            'instrumentalness', 'liveness', 'loudness', 'speechiness',
            'tempo', 'valence']
jazzFeatures = jazzData[features] ## Extracting features from the dataset

## Limit data to popularity > median
median = np.median(jazzFeatures['popularity'])
jazzLimited = jazzFeatures[jazzFeatures['popularity'] > median]

## Find basic information of final Jazz dataset
```

```
def main():
    print("Number of observations :: ", len(jazzLimited.index))
    print("Number of columns :: ", len(jazzLimited.columns))
    print("Headers :: ", jazzLimited.columns.values)

if __name__ == "__main__":
    main()

## Drop popularity from dataset for pca analysis later on
pcaData = jazzLimited.drop('popularity', 1)
```

```
Number of observations ::  4667
Number of columns ::  10
Headers ::  ['popularity' 'acousticness' 'danceability' 'energy' 'instrumentalness'
 'liveness' 'loudness' 'speechiness' 'tempo' 'valence']
```

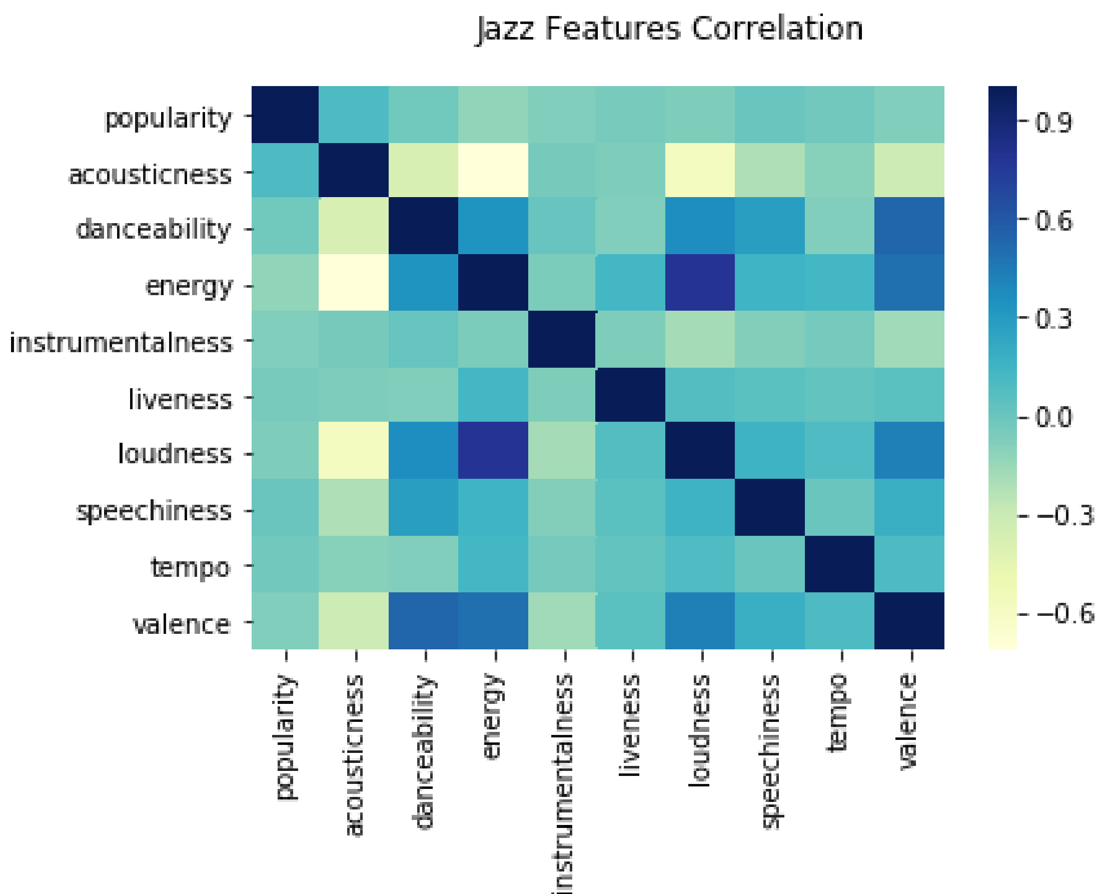
3 Genre Analysis: Jazz

To produce clear well centered plots, we'll normalize the features, plot scatterplots and then explore the actual matrix breakdown to see if there's any localized variance we can analyze. Additionally, we wanted to see if we a heatmap could give us some direction for our research.

```
In [5]: ## Normalizing all the features
        n = jazzLimited.shape[0]
        jazzMeans = np.mean(jazzLimited,0)
        normJazz = (jazzLimited-jazzMeans)/np.sqrt(n)
```

```
In [6]: jazzCorr = normJazz.corr()
        plt.suptitle("Jazz Features Correlation")
        sns.heatmap(jazzCorr, cmap="YlGnBu")
```

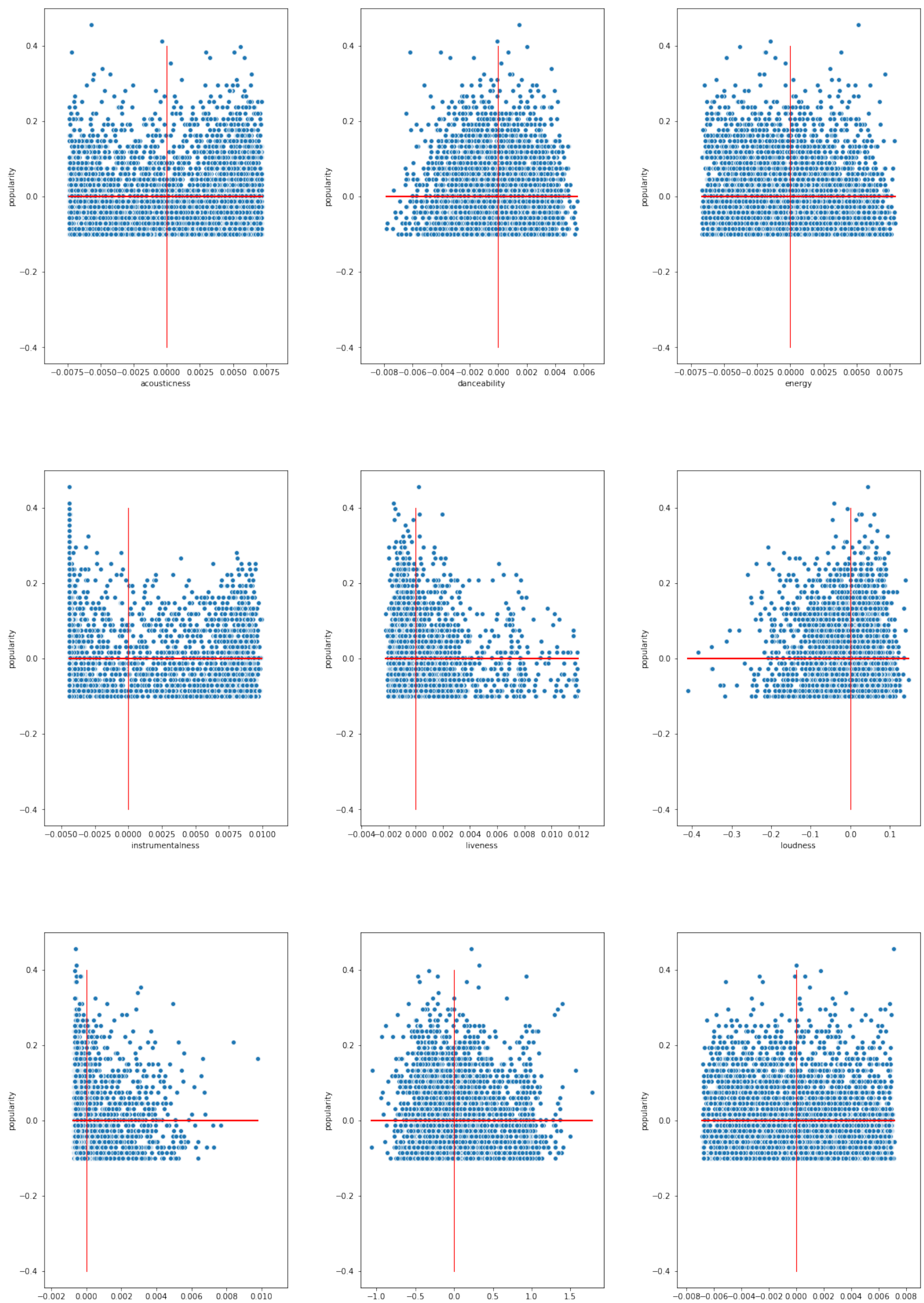
```
Out[6]: <matplotlib.axes._subplots.AxesSubplot at 0x7f9f72caa7b8>
```



Unfortunately, it doesn't seem like popularity is correlated with any specific features. Instead, we'll examine the scatterplots and try to find some trends.

```
In [7]: ## PLOT NORMALIZED FEATURES
        plt.figure(figsize=(20, 30))
        plt.suptitle("Scatter Matrix of Song Features")
        plt.subplots_adjust(wspace=0.3, hspace=0.3)
        for i in range(len(features)-1):
            plt.subplot(3, 3, i+1)
            sns.scatterplot(normJazz.iloc[:,i+1],normJazz.iloc[:,0])
            plt.plot([0, 0], [-0.4, 0.4], linewidth=1, color = 'r')
            plt.plot([normJazz.iloc[:,i+1].min(), normJazz.iloc[:,i+1].max()], [0, 0], linewid
            plt.xlabel(features[i+1])
            plt.ylabel(features[0])
```

Scatter Matrix of Song Features



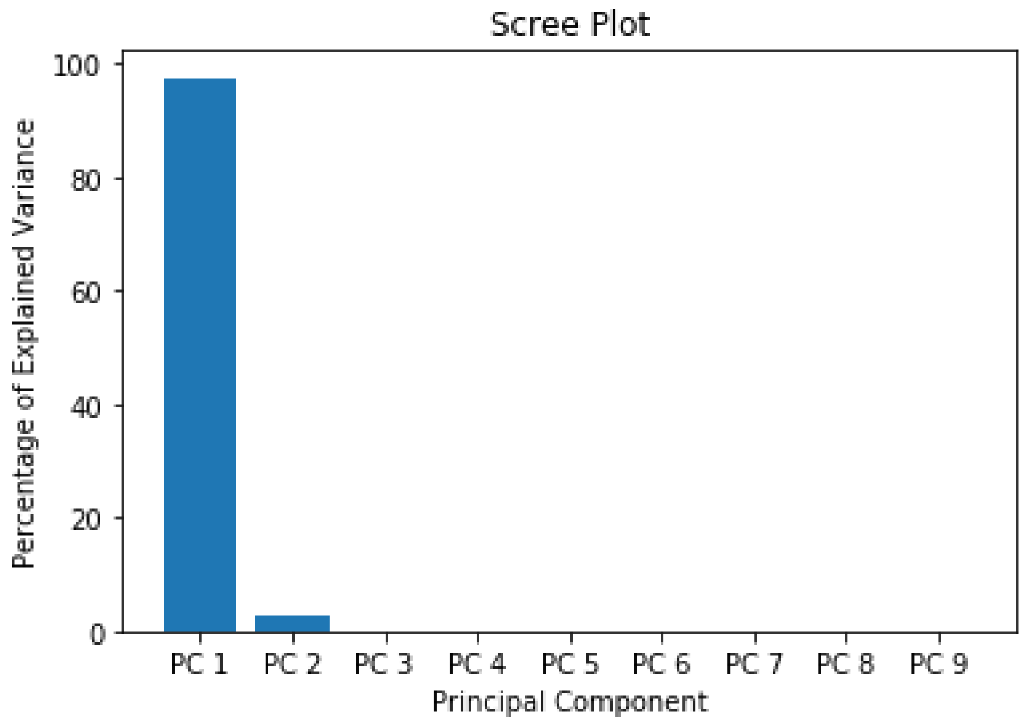
Trends: 1. Danceability and tempo seem normally distributed, with the peak popularity centered about the mean. 2. Energy, liveness and speechiness show a negative correlation with popularity. 3. Instrumentalness is least popular when it's at it's mean. In other words, the further a songs varies from mean instrumentalness the more popular it's likely to be.

3.1 PCA

While these observations provide some direction, they're more subjective and don't provide much concrete trends or measurements. Since most of us can only visualize two or three dimensions, so the most logical next-step for our 10 dimensional data was reducing its dimensions. This makes the data easier to interpret and speeds up these computations. Dimensionality reduction also helps us find appropriate functions of the predictors that correspond to interesting features, get rid of unwanted predictors, and removes contamination from measurement noise. We proceeded using Principal Component Analysis (PCA) as this was a great way to quantify and pinpoint the variation within our dataset. PCA is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components. These new principal components act as new axes, which help reduce the dimensionality of our dataset without loss of information. To better interpret these components we used loading plots of our new axes and examined what variables were the most heavily weighted as a means of better understanding our dataset.

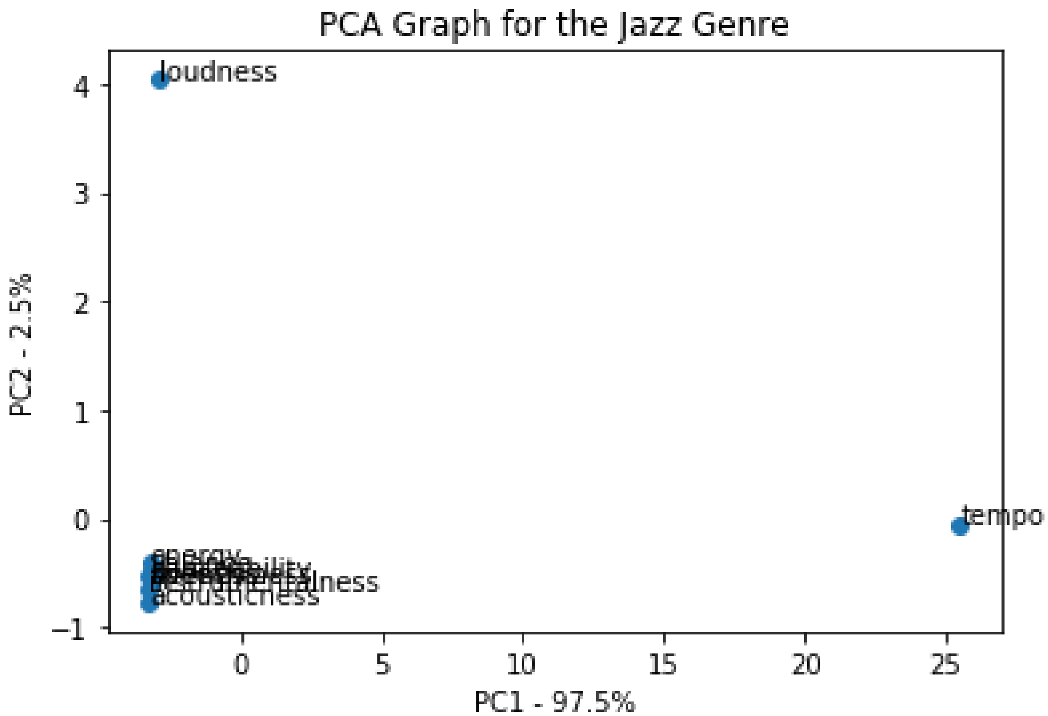
```
In [33]: from sklearn.decomposition import PCA
         from sklearn import preprocessing
```

```
In [34]: ## PCA w/o popularity
         pcaDataNorm = normJazz.drop('popularity', 1)

         pca = PCA()
         pca.fit(pcaDataNorm.T)
         pca_data = pca.transform(pcaDataNorm.T)

         per_var = np.round(pca.explained_variance_ratio_*100, decimals = 1)
         labels = ['PC 1', 'PC 2', 'PC 3', 'PC 4', 'PC 5', 'PC 6', 'PC 7', 'PC 8','PC 9']

         plt.bar(x = range(1,len(per_var)+1), height = per_var, tick_label = labels)
         plt.ylabel('Percentage of Explained Variance')
         plt.xlabel('Principal Component')
         plt.title("Scree Plot")
         plt.show()
```



We see here that all the variation in our data set can be explained by the first two prinicpal components. We'll take a look into the weight of the features within these components to better intrpret our results.

```
In [35]: pcaFeatures = ['acousticness', 'danceability', 'energy', ## Defining the desired song
                        'instrumentalness', 'liveness', 'loudness', 'speechiness',
                        'tempo', 'valence']
         pca_df = pd.DataFrame(pca_data, index = pcaFeatures, columns = labels)

         plt.scatter(pca_df['PC 1'],pca_df['PC 2'])
         plt.title('PCA Graph for the Jazz Genre')
         plt.xlabel('PC1 - {0}%'.format(per_var[0]))
         plt.ylabel('PC2 - {0}%'.format(per_var[1]))

         for sample in pca_df.index:
             plt.annotate(sample, (pca_df['PC 1'].loc[sample],pca_df['PC 2'].loc[sample]))

         plt.show()
```



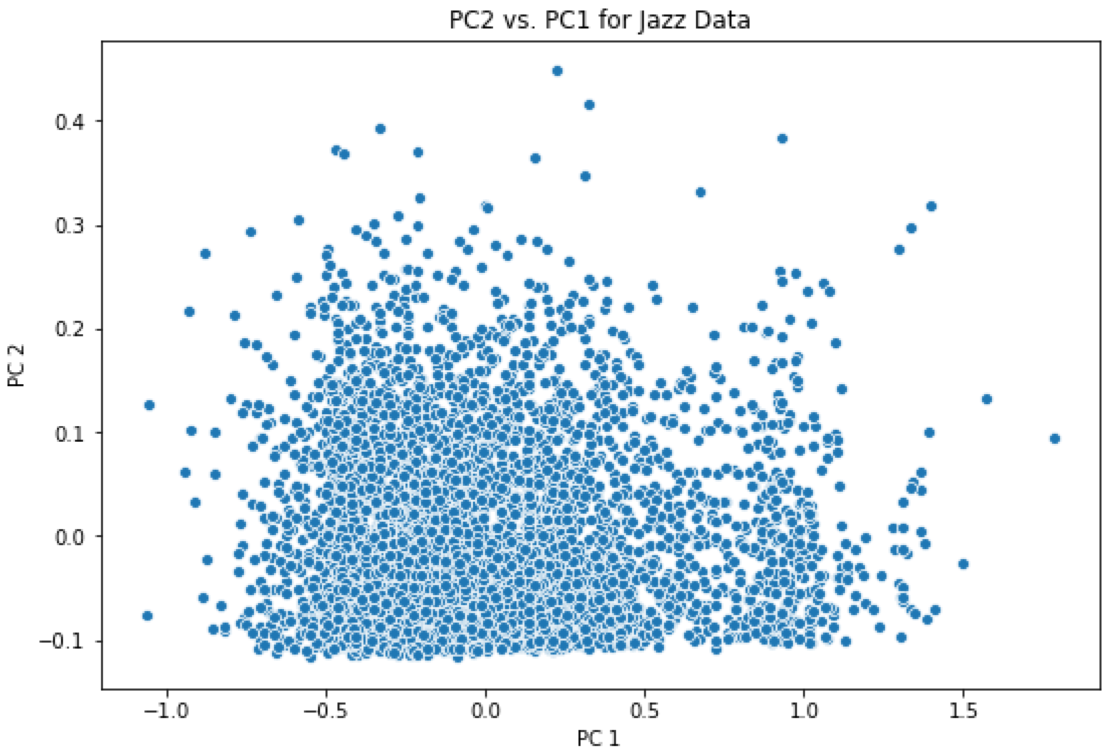
Here we see that tempo and loudness are our most heavily weighted features, which implies that the variation in popularity is heavily influenced by these song features. Since we utilized different functions to the ones we used in class. I thought it would be a good idea to go through these different functions and see if we can add any information to our findings.

```
In [19]: ## Getting matrices
         u,s,vt = np.linalg.svd(normJazz, full_matrices = False)
         u.shape, s, vt.shape

Out[19]: ((4667, 10),
          array([28.74949571,  5.98518129,  4.61904795,  0.37501179,  0.29082762,
                  0.2304098 ,  0.15048692,  0.1271072 ,  0.10301697,  0.07268919]),
          (10, 10))
```

```
In [20]: ## Computing total variance by summing squared singular values
         total_variance = np.sum(s**2)
         print("total_variance: {:.3f} should approximately equal the sum of feature variances
                .format(total_variance, np.sum(np.var(pcaData, axis=0))))

total_variance: 884.025 should approximately equal the sum of feature variances: 848.399
```

```
In [21]: ##2d shit
         jazz_2d = normJazz@np.transpose(vt)[:,0:2]
```

In [22]:
```
plt.figure(figsize=(9, 6))
plt.title("PC2 vs. PC1 for Jazz Data")
sns.scatterplot(jazz_2d.iloc[:, 0], jazz_2d.iloc[:, 1])
plt.xlabel('PC 1')
plt.ylabel('PC 2');
```

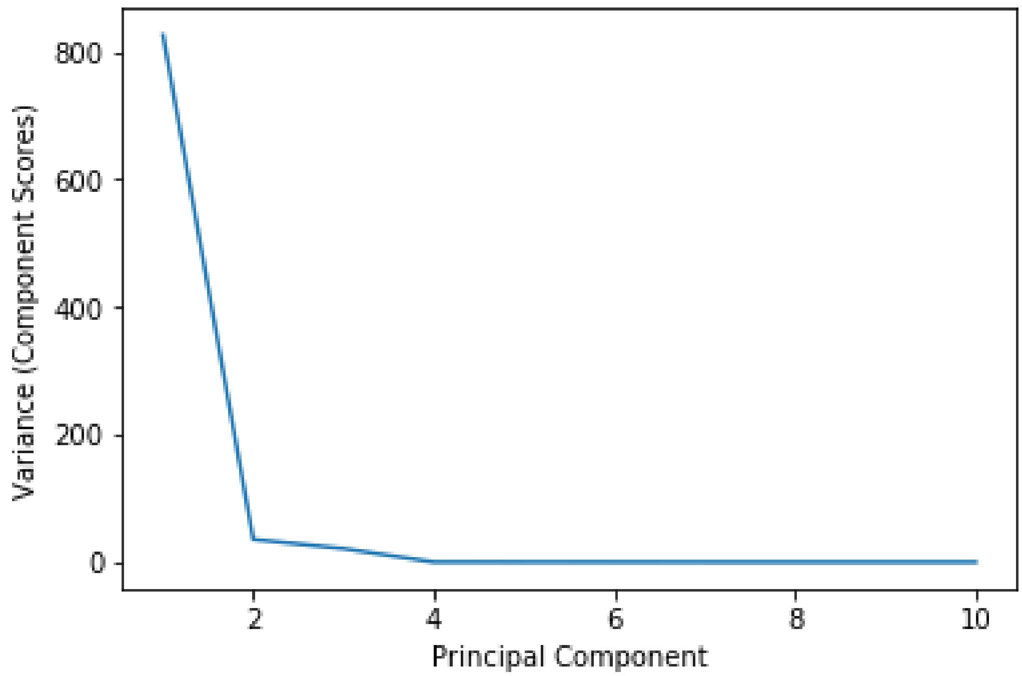


In [23]:
```
## Finding how much variance is explained by our 2 principal components
two_dim_variance = (s[0]**2 + s[1]**2)/total_variance
print("Our first 2 principal components explain ",
      100*round(two_dim_variance, 5),
      "% of the variance in our data.")
```

```
Our first 2 principal components explain  97.54899999999999 % of the variance in our data.
```

In [24]:
```
plt.plot(np.array([1,2,3,4,5,6,7,8,9,10]),(s**2))
plt.xlabel("Principal Component")
plt.ylabel("Variance (Component Scores)")
```

Out[24]:
```
Text(0, 0.5, 'Variance (Component Scores)')
```



The results are consistent and since we found the “defining” features for Jazz songs, we’ll try to expand out scope and see if other genre’s tempo and loudness are correlated with their popularity.

3.2 All Genre Analysis

In [36]:
```
totalData = pd.DataFrame(totalData)

## Create df topSongs with top 50% of songs based on popularity from each genre
genres = totalData['genre'].unique() ## Define all genres

## Creates df topSongs w/ top 50% songs based on popularity from the frist genre
index = (totalData[totalData['genre'] == genres[0]]['popularity']) > np.median(totalDa
topSongs = pd.DataFrame(totalData[totalData['genre'] == genres[0]][index])

## Loop through each genre
for i in range(len(genres)-1):
    a = (totalData[totalData['genre'] == genres[i+1]]['popularity']) > np.median(tota
    temp = pd.DataFrame(totalData[totalData['genre'] == genres[i+1]][a])
    topSongs = topSongs.append(temp, ignore_index=True)
```

In [37]:
```
## Find median data for each genre from our limited Dataset
groupedData = topSongs.groupby('genre').median()
groupedData.sort_values('popularity', ascending = False)
```

Out[37]:
```
                    popularity  acousticness  danceability  duration_ms  energy  \
genre
```

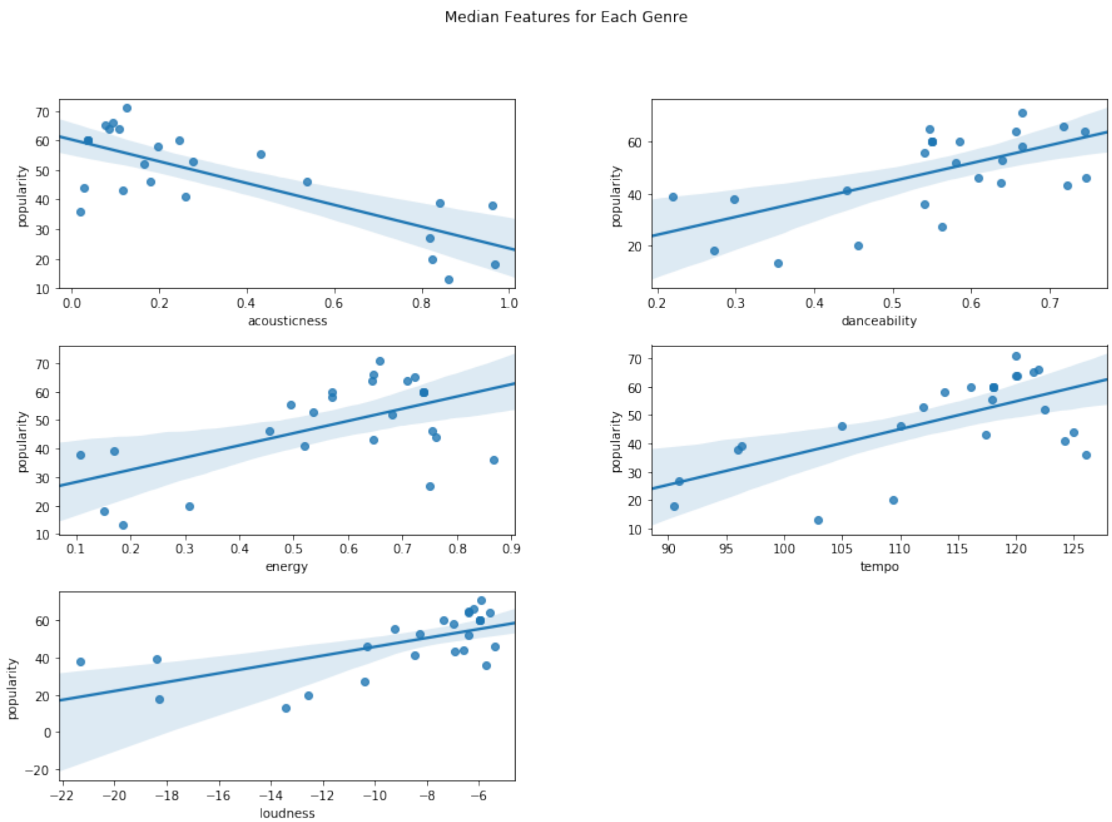
Pop	71.0	0.12700	0.6650	214910.0	0.658
Rap	66.0	0.09330	0.7170	216035.0	0.646
Rock	65.0	0.07620	0.5470	226120.0	0.723
Dance	64.0	0.08450	0.6570	215827.0	0.709
Hip-Hop	64.0	0.10800	0.7440	216700.0	0.644
Anime	60.0	0.03630	0.5500	225973.0	0.738
Blues	60.0	0.03630	0.5500	225973.0	0.738
Childrens Music	60.0	0.03630	0.5500	225973.0	0.738
Indie	60.0	0.24700	0.5850	216497.0	0.570
Alternative	60.0	0.03630	0.5500	225973.0	0.738
R&B	58.0	0.19600	0.6650	221266.0	0.570
Folk	55.5	0.43100	0.5400	227084.5	0.494
Soul	53.0	0.27800	0.6400	224166.5	0.536
Country	52.0	0.16500	0.5800	212013.0	0.680
Jazz	46.0	0.53900	0.6090	244867.0	0.454
Reggaeton	46.0	0.18000	0.7470	221064.5	0.754
Electronic	44.0	0.02730	0.6380	249750.0	0.761
Reggae	43.0	0.11600	0.7220	228467.0	0.646
World	41.0	0.26000	0.4420	288415.0	0.519
Soundtrack	39.0	0.84100	0.2200	175233.5	0.169
Classical	38.0	0.96100	0.2970	259173.0	0.108
Ska	36.0	0.01975	0.5400	193445.5	0.868
Comedy	27.0	0.81700	0.5620	206107.0	0.750
Movie	20.0	0.82400	0.4550	181806.5	0.308
Opera	18.0	0.96600	0.2715	230900.0	0.151
A Capella	13.0	0.86100	0.3530	185800.0	0.185

	instrumentalness	liveness	loudness	speechiness	tempo \
genre					
Pop	0.000000	0.122	-5.9460	0.06000	119.9700
Rap	0.000000	0.131	-6.2060	0.14400	121.9775
Rock	0.000029	0.123	-6.4060	0.03920	121.4540
Dance	0.000000	0.125	-5.5730	0.05260	120.0130
Hip-Hop	0.000000	0.129	-6.4035	0.17400	120.1370
Anime	0.000058	0.131	-5.9670	0.05030	118.0700
Blues	0.000058	0.131	-5.9670	0.05030	118.0700
Childrens Music	0.000058	0.131	-5.9670	0.05030	118.0700
Indie	0.000075	0.116	-7.3830	0.04190	116.0790
Alternative	0.000058	0.131	-5.9670	0.05030	118.0700
R&B	0.000000	0.120	-6.9935	0.07345	113.7895
Folk	0.000267	0.114	-9.2470	0.03530	117.9675
Soul	0.000069	0.118	-8.2900	0.04835	111.9905
Country	0.000000	0.124	-6.3950	0.03500	122.5220
Jazz	0.027100	0.114	-10.3250	0.04380	105.0010
Reggaeton	0.000000	0.134	-5.4200	0.09070	110.0445
Electronic	0.157000	0.124	-6.6090	0.05560	124.9990
Reggae	0.000005	0.117	-6.9310	0.07130	117.4250
World	0.000040	0.120	-8.4540	0.03570	124.2200

```
Soundtrack              0.883000     0.109  -18.3735      0.03990   96.2985
Classical               0.843000     0.108  -21.3200      0.04290   96.0150
Ska                     0.000038     0.162   -5.7225      0.06330  126.0195
Comedy                  0.000000     0.772  -10.3990      0.93000   90.9010
Movie                   0.000032     0.136  -12.5680      0.04150  109.4500
Opera                   0.014000     0.128  -18.3025      0.04520   90.4870
A Capella               0.000001     0.114  -13.4510      0.03450  102.9010

                 valence
genre
Pop               0.4810
Rap               0.4440
Rock              0.5180
Dance             0.5190
Hip-Hop           0.4815
Anime             0.4530
Blues             0.4530
Childrens Music   0.4530
Indie             0.4010
Alternative       0.4530
R&B               0.4410
Folk              0.4030
Soul              0.4540
Country           0.5390
Jazz              0.4980
Reggaeton         0.6780
Electronic        0.3640
Reggae            0.7120
World             0.2200
Soundtrack        0.0593
Classical         0.1330
Ska               0.6880
Comedy            0.3790
Movie             0.3310
Opera             0.1250
A Capella         0.2390
```

In [41]:
```python
cols = groupedData.columns
label = ['popularity', 'acousticness', 'danceability', 'energy', 'tempo', 'loudness']
target = groupedData[['popularity', 'acousticness', 'danceability', 'energy', 'tempo'

## Plot
plt.figure(figsize=(15, 10))
plt.suptitle("Median Features for Each Genre")
plt.subplots_adjust(wspace=0.3, hspace=0.3)
for i in range(5):
    plt.subplot(3, 2, i+1)
    sns.regplot(target.iloc[:, i+1], target.iloc[:, 0])
```

```
        plt.xlabel(label[i+1])
        plt.ylabel(label[0])
```



Our regression plots for all genres are somewhat consistent with our PCA findings, however they do seem to be skewed heavily by outliers. This leads us to believe that even though we cn find some correlation with popularity features and song features, there are no definitive sounds which make a song popular. If that were the case then we would see no variation at all and all sonds would have the same metrics for their audio features.

3.3 Jazz Artist Analysis

Finally, we'll examine specfic artists within the jazz genre to see if the subgenre trend will become more apparent.

```
In [50]: ## Miles Davis, Earth, Wind and Fire, Louis Armstrong
         artists = ['Earth, Wind & Fire', 'Miles Davis', 'Louis Armstrong']
         ewf = totalData[totalData['artist_name'] == artists[0]][features] # Earth, wind and f
         milesDavis = totalData[totalData['artist_name'] == artists[1]][features] # Miles Davi
         armstrong = totalData[totalData['artist_name'] == artists[2]][features] # Louis Amrst
```

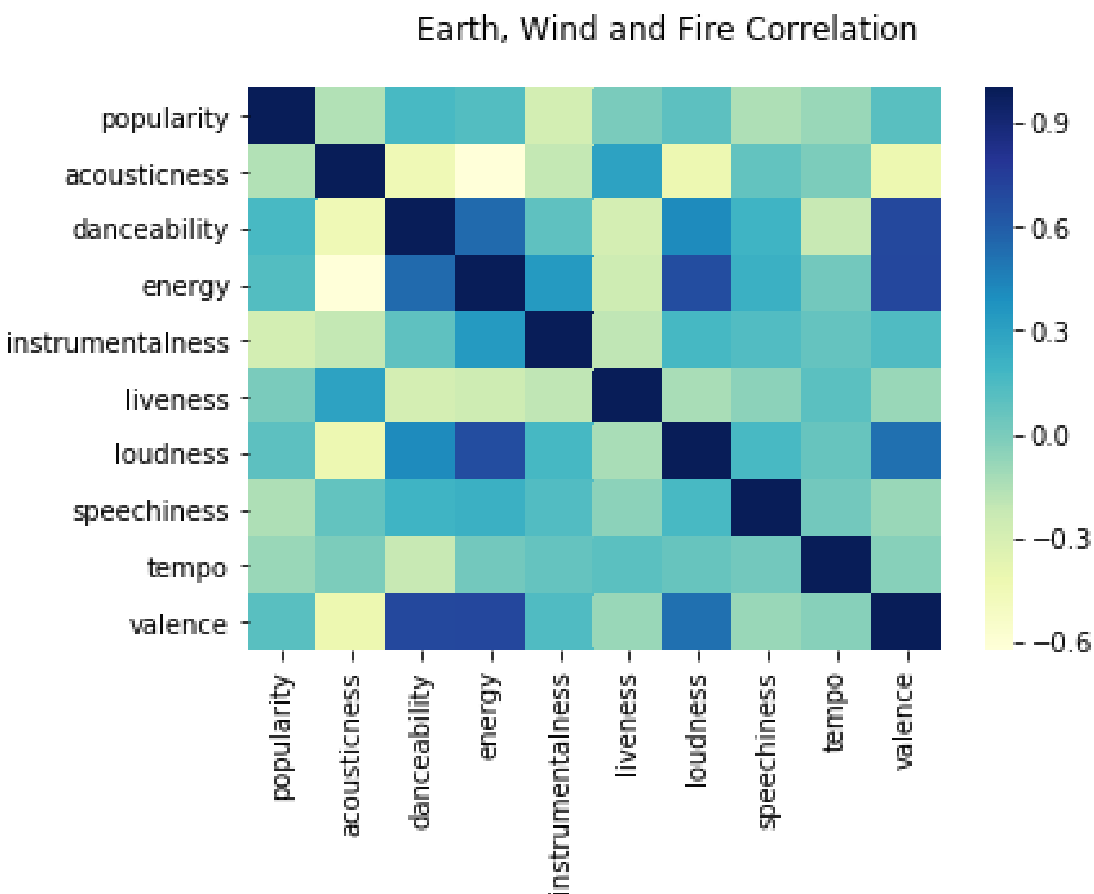
3.3.1 Earth, Wind & Fire

```
In [58]: ## Normalizing all the features
         n = ewf.shape[0]
```

```
ewfMeans = np.mean(ewf,0)
ewfNorm = (ewf-ewfMeans)/np.sqrt(n)
```

In [117]:
```
## Earth, Wind and Fire songs
ewfCorr = ewfNorm.corr()
plt.suptitle("Earth, Wind and Fire Correlation")
sns.heatmap(ewfCorr, cmap="YlGnBu")
```

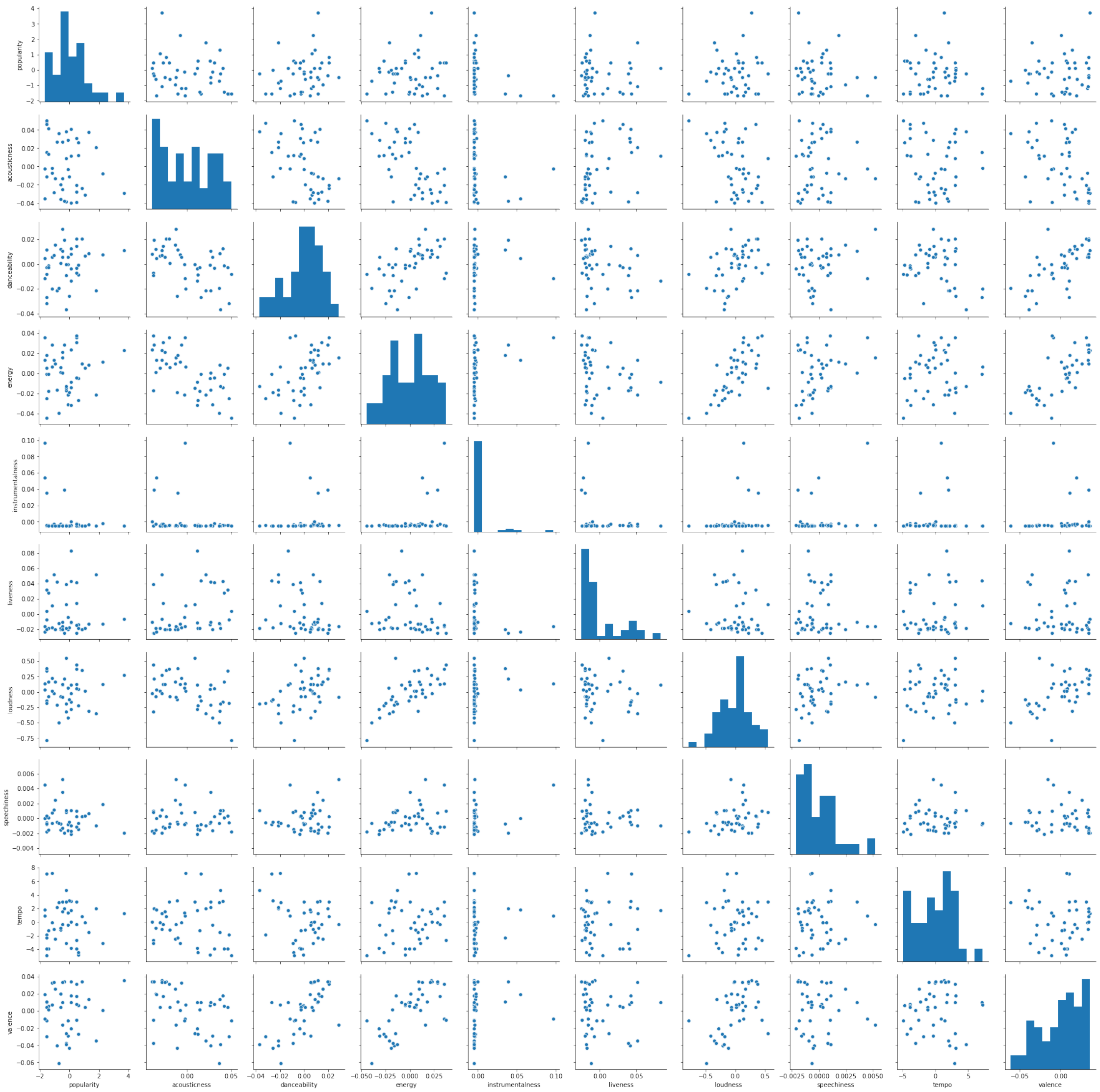
Out[117]: <matplotlib.axes._subplots.AxesSubplot at 0x7fec22cbda58>



We see here that there are much stronger correlations with specfic artists. This follows the Simpson's paradox and shows how the lack of trends and correlations are caused by generalizations and incorrect groupings. Genre's are not specific enough in categorizing songs to allow us to see a clear trend in our data.

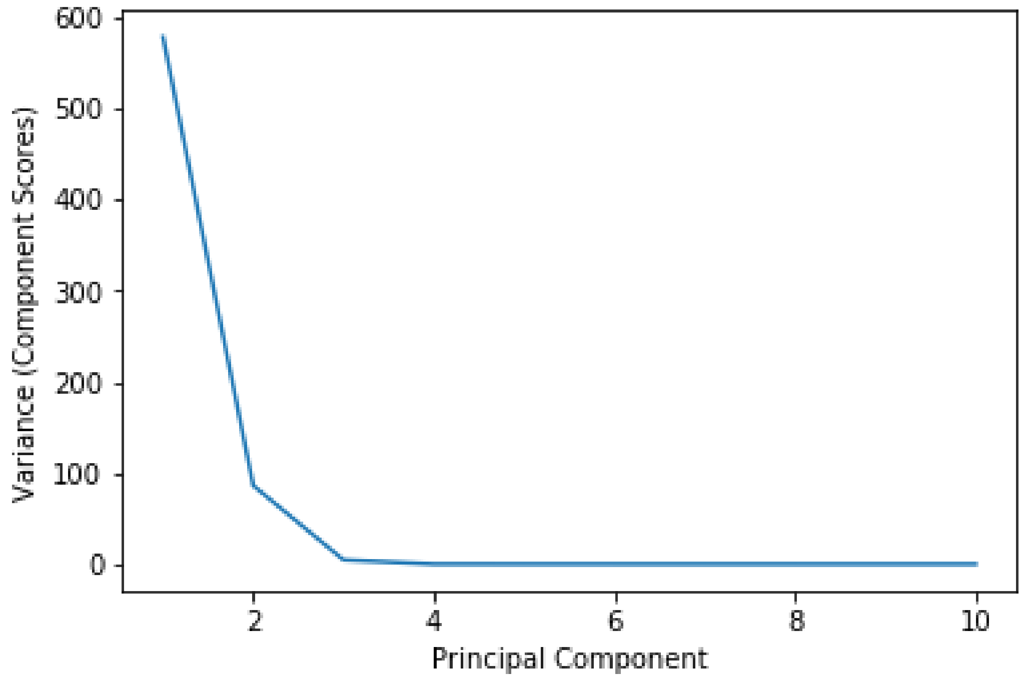
In [60]:
```
sns.pairplot(ewfNorm)
```

Out[60]: <seaborn.axisgrid.PairGrid at 0x7fec38496be0>




```
In [88]: ## Getting matrices
         u,s,vt = np.linalg.svd(ewfNorm, full_matrices = False)
         plt.plot(np.array([1,2,3,4,5,6,7,8,9,10]),(s**2))
         plt.xlabel("Principal Component")
         plt.ylabel("Variance (Component Scores)")

         ## Finding how much variance is explained by our 2 principal components
         total_variance = np.sum(s**2)
         two_dim_variance = (s[0]**2 + s[1]**2)/total_variance
         print("Our first 2 principal components explain ",
               100*round(two_dim_variance, 5),
               "% of the variance in our data.")

         print(s)
```

```
Our first 2 principal components explain  99.27 % of the variance in our data.
[2.40430639e+01 9.28962146e+00 2.17022303e+00 2.69748069e-01
 2.10515078e-01 1.67959313e-01 1.24341445e-01 8.33559731e-02
 7.20369614e-02 1.02401925e-02]
```

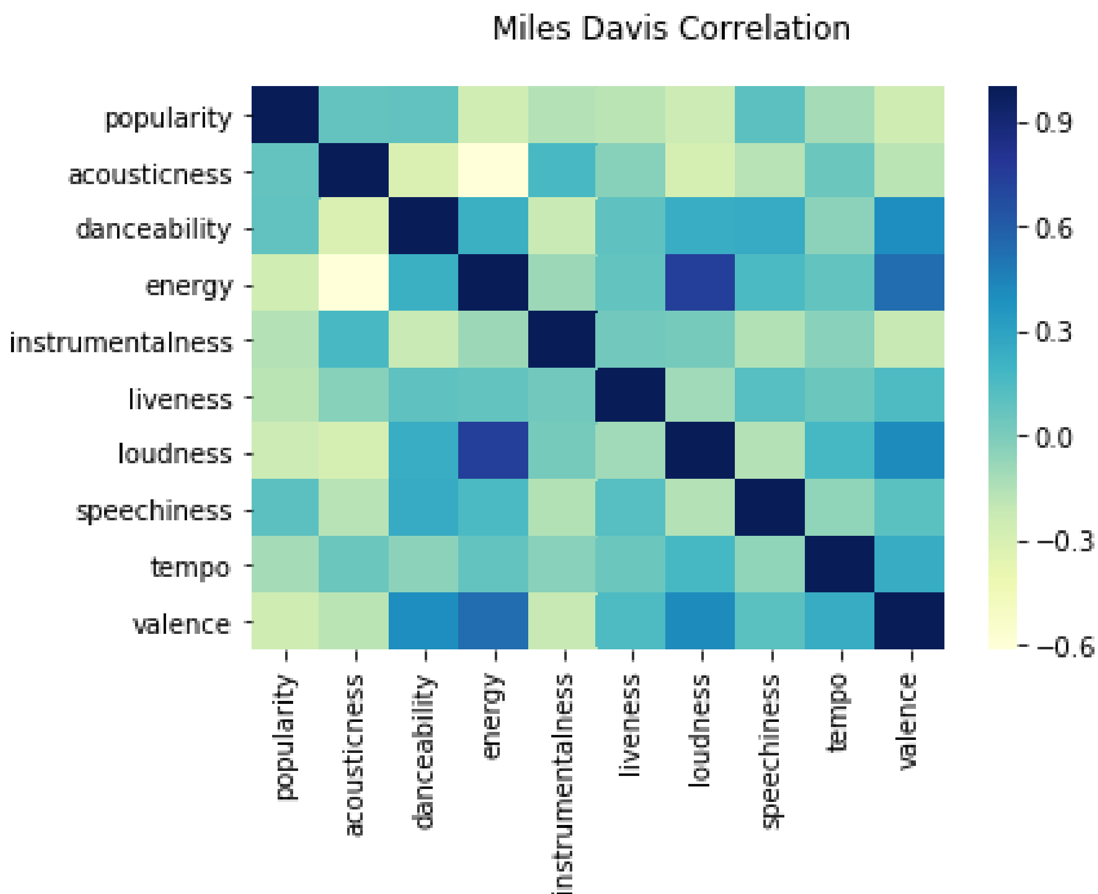


3.3.2 Miles Davis

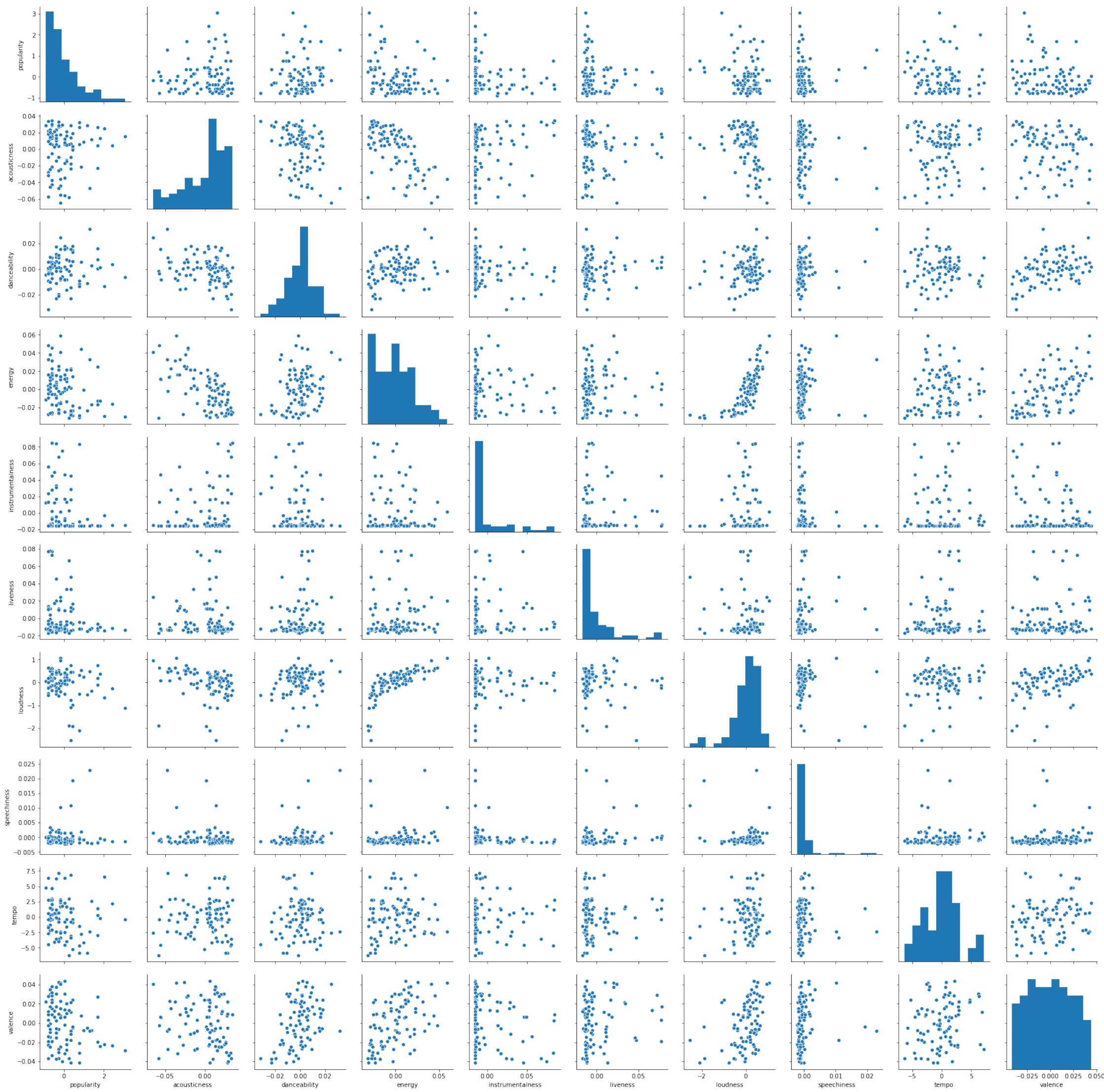
```
In [67]: ## Normalizing all the features
         n = milesDavis.shape[0]
         davisMeans = np.mean(milesDavis,0)
         davisNorm = (milesDavis-davisMeans)/np.sqrt(n)
```

```
In [114]: ## Miles Davis Songs
          davisCorr = davisNorm.corr()
          plt.suptitle("Miles Davis Correlation", )
          sns.heatmap(davisCorr, cmap="YlGnBu")
          sns.pairplot(davisNorm)
```

```
Out[114]: <seaborn.axisgrid.PairGrid at 0x7fec251da6d8>
```

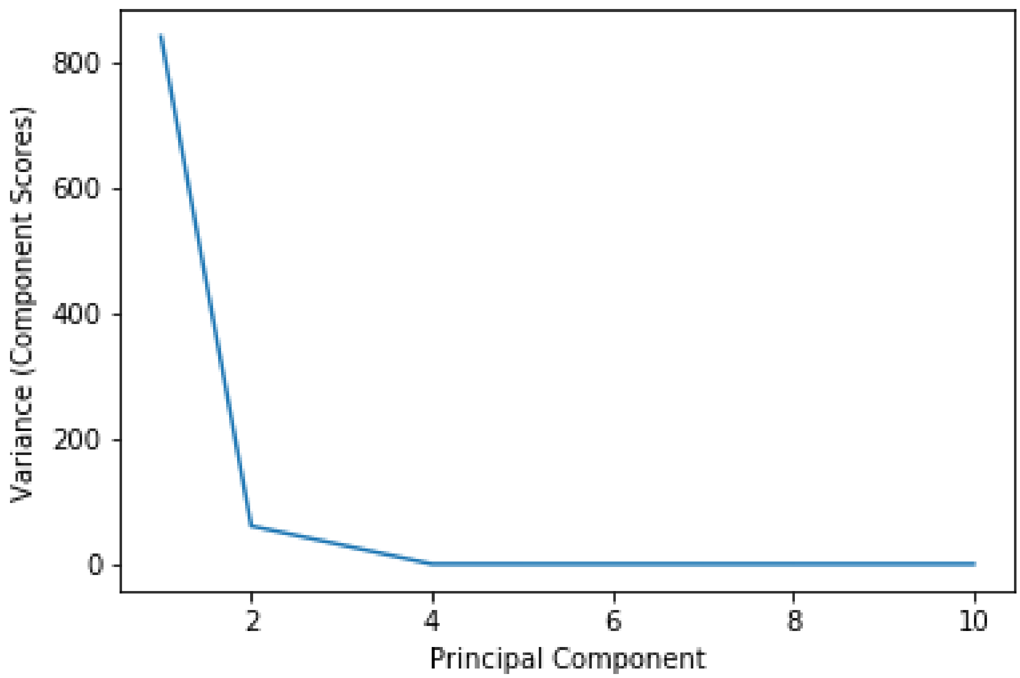


Miles Davis Correlation
popularity
acousticness
danceability
energy
instrumentalness
liveness
loudness
speechiness
tempo
valence
0.9
0.6
0.3
0.0
−0.3
−0.6



In [71]:
```python
## Getting matrices
u,s,vt = np.linalg.svd(davisNorm, full_matrices = False)
plt.plot(np.array([1,2,3,4,5,6,7,8,9,10]),(s**2))
plt.xlabel("Principal Component")
plt.ylabel("Variance (Component Scores)")

## Finding how much variance is explained by our 2 principal components
total_variance = np.sum(s**2)
two_dim_variance = (s[0]**2 + s[1]**2)/total_variance
print("Our first 2 principal components explain ",
      100*round(two_dim_variance, 5),
      "% of the variance in our data.")
```

```
Our first 2 principal components explain  96.69 % of the variance in our data.
```



4 Remarks

Tempo and Loudness seem to be the most important features in determining song popularity, however there is no definitive audi makeup that would make a song popular. Additionally, we saw Simpson's paradox in action and were able to conclude that the variability of song features within genres is extremely high and that subgenres or artists are a much more effective way of classifying or clustering music data.