miditime-New York State

June 7, 2020

PART 1: MAKING MIDI for different Cities

[1]:
```
import pandas as pd
```

[2]:
```
time=pd.read_csv('data/time_series_data.csv')
others=pd.read_csv('data/other_information.csv')
location=pd.read_csv('data/location.csv')
```

[19]:
```
# find new york cities other than new york (4):
SA=time[time['Combined_Key'].str.match('Saratoga')].iloc[0,:]
AL=time[time['Combined_Key'].str.match('Albany')].iloc[0,:]
WY=time[time['Combined_Key'].str.match('Wyoming')].iloc[0,:]
ES=time[time['Combined_Key'].str.match('Essex')].iloc[0,:]
```

[20]:
```
#Using daily increase as New York example point out that daily numbers are␣
 ↪better in sound
SA_day=SA[1:].diff().fillna(0)
AL_day=AL[1:].diff().fillna(0)
WY_day=WY[1:].diff().fillna(0)
ES_day=ES[1:].diff().fillna(0)
```

[21]:
```
#basic range:
low_SA=min(SA_day)
high_SA=max(SA_day)
low_AL=min(AL_day)
high_AL=max(AL_day)
low_WY=min(WY_day)
high_WY=max(WY_day)
low_ES=min(ES_day)
high_ES=max(ES_day)
```

[22]:
```
from miditime.miditime import MIDITime
```

[23]:
```
#generate midi file with different base octave based on their geological␣
 ↪location, cities in a norther position have higher base octave
midi1 = MIDITime(120, 'results/NYmix/WY.mid', 120, 5, 1)
midi2 = MIDITime(120, 'results/NYmix/AL.mid', 120, 4, 1)
midi3 = MIDITime(120, 'results/NYmix/SA.mid', 120, 6, 1)
midi4 = MIDITime(120, 'results/NYmix/ES.mid', 120, 7, 1)
```

[24]:

```python
#different mag_to_pitch function
def mag_to_pitch_tuned(magnitude, low, high):
    # Where does this data point sit in the domain of your data? (I.E. the min␣
 ↪magnitude is 3, the max in 5.6). In this case the optional 'True' means the␣
 ↪scale is reversed, so the highest value will return the lowest percentage.
    scale_pct = midi1.linear_scale_pct(low, high, magnitude)

    # Another option: Linear scale, reverse order
    # scale_pct = mymidi.linear_scale_pct(3, 5.7, magnitude, True)

    # Another option: Logarithmic scale, reverse order
    #scale_pct = mymidi.log_scale_pct(low, high, magnitude, True)

    # Pick a range of notes. This allows you to play in a key.
    c_major = ['C', 'D', 'E', 'F', 'G', 'A', 'B']
    #Find the note that matches your data point

    note = midi1.scale_to_note(scale_pct, c_major)

    #Translate that note to a MIDI pitch
    midi_pitch = midi1.note_to_midi_pitch(note)

    return midi_pitch
```

[25]:

```python
def mag_to_pitch_tuned_Nmin(magnitude, low, high):
    # Where does this data point sit in the domain of your data? (I.E. the min␣
 ↪magnitude is 3, the max in 5.6). In this case the optional 'True' means the␣
 ↪scale is reversed, so the highest value will return the lowest percentage.
    scale_pct = midi2.linear_scale_pct(low, high, magnitude)

    # Another option: Linear scale, reverse order
    # scale_pct = mymidi.linear_scale_pct(3, 5.7, magnitude, True)

    # Another option: Logarithmic scale, reverse order
    #scale_pct = mymidi.log_scale_pct(low, high, magnitude, True)

    # Pick a range of notes. This allows you to play in a key.
    c_Nmin = ['C', 'D', 'Eb', 'F', 'G', 'Ab', 'Bb']
    #Find the note that matches your data point

    note = midi2.scale_to_note(scale_pct, c_Nmin)

    #Translate that note to a MIDI pitch
    midi_pitch = midi2.note_to_midi_pitch(note)

    return midi_pitch
```

[26]:

```
def mag_to_pitch_tuned_Hmin(magnitude, low, high):
    # Where does this data point sit in the domain of your data? (I.E. the min␣
 ↪magnitude is 3, the max in 5.6). In this case the optional 'True' means the␣
 ↪scale is reversed, so the highest value will return the lowest percentage.
    scale_pct = midi3.linear_scale_pct(low, high, magnitude)

    # Another option: Linear scale, reverse order
    # scale_pct = mymidi.linear_scale_pct(3, 5.7, magnitude, True)

    # Another option: Logarithmic scale, reverse order
    #scale_pct = mymidi.log_scale_pct(low, high, magnitude, True)

    # Pick a range of notes. This allows you to play in a key.
    c_Hmin = ['C', 'D', 'Eb', 'F', 'G', 'Ab', 'B']
    #Find the note that matches your data point

    note = midi3.scale_to_note(scale_pct, c_Hmin)

    #Translate that note to a MIDI pitch
    midi_pitch = midi3.note_to_midi_pitch(note)

    return midi_pitch
```

[27]:

```
def mag_to_pitch_tuned_Mmin(magnitude, low, high):
    # Where does this data point sit in the domain of your data? (I.E. the min␣
 ↪magnitude is 3, the max in 5.6). In this case the optional 'True' means the␣
 ↪scale is reversed, so the highest value will return the lowest percentage.
    scale_pct = midi4.linear_scale_pct(low, high, magnitude)

    # Another option: Linear scale, reverse order
    # scale_pct = mymidi.linear_scale_pct(3, 5.7, magnitude, True)

    # Another option: Logarithmic scale, reverse order
    #scale_pct = mymidi.log_scale_pct(low, high, magnitude, True)

    # Pick a range of notes. This allows you to play in a key.
    c_Mmin = ['C', 'D', 'Eb', 'F', 'G', 'A', 'B']
    #Find the note that matches your data point

    note = midi4.scale_to_note(scale_pct, c_Mmin)

    #Translate that note to a MIDI pitch
    midi_pitch = midi4.note_to_midi_pitch(note)

    return midi_pitch
```

[28]:

```
# WY, C major:
data1=WY_day.copy()
```

```
data1.index=list(range(len(data1)))
data1=data1.to_dict()
data1=[{'beat': d, "overall": data1[d]} for d in data1.keys()]
my_data_timed1 = [{'beat': midi1.beat(d['beat']), 'magnitude': d['overall']} for d in data1]
note_list1 = []
counter=0
for d in my_data_timed1:
    try:
        note_list1.append([
            d['beat'],
            mag_to_pitch_tuned(d['magnitude'], low_WY, high_WY),
            100,  # velocity
            1  # duration, in beats
        ])
        counter +=1
    except Exception as e:
        print(d['beat'])
        print(counter)
        print(e)
        # some notes may do not match
midi1.add_track(note_list1)
midi1.save_midi()
```

```
62 0.0 1 100
62 0.66 1 100
62 1.31 1 100
62 1.97 1 100
62 2.63 1 100
62 3.29 1 100
62 3.94 1 100
62 4.6 1 100
62 5.26 1 100
62 5.91 1 100
62 6.57 1 100
62 7.23 1 100
62 7.89 1 100
62 8.54 1 100
62 9.2 1 100
62 9.86 1 100
62 10.51 1 100
62 11.17 1 100
62 11.83 1 100
62 12.48 1 100
62 13.14 1 100
62 13.8 1 100
62 14.46 1 100
```

```
62 15.11 1 100
62 15.77 1 100
62 16.43 1 100
62 17.08 1 100
62 17.74 1 100
62 18.4 1 100
62 19.06 1 100
62 19.71 1 100
62 20.37 1 100
62 21.03 1 100
62 21.68 1 100
62 22.34 1 100
62 23.0 1 100
62 23.66 1 100
62 24.31 1 100
62 24.97 1 100
62 25.63 1 100
62 26.28 1 100
62 26.94 1 100
62 27.6 1 100
62 28.25 1 100
62 28.91 1 100
62 29.57 1 100
62 30.23 1 100
62 30.88 1 100
62 31.54 1 100
62 32.2 1 100
62 32.85 1 100
62 33.51 1 100
62 34.17 1 100
62 34.83 1 100
62 35.48 1 100
62 36.14 1 100
62 36.8 1 100
62 37.45 1 100
62 38.11 1 100
62 38.77 1 100
62 39.43 1 100
62 40.08 1 100
64 40.74 1 100
62 41.4 1 100
64 42.05 1 100
62 42.71 1 100
62 43.37 1 100
62 44.02 1 100
62 44.68 1 100
62 45.34 1 100
64 46.0 1 100
```

```
64 46.65 1 100
62 47.31 1 100
64 47.97 1 100
64 48.62 1 100
62 49.28 1 100
62 49.94 1 100
64 50.6 1 100
64 51.25 1 100
62 51.91 1 100
62 52.57 1 100
62 53.22 1 100
62 53.88 1 100
62 54.54 1 100
69 55.2 1 100
62 55.85 1 100
64 56.51 1 100
62 57.17 1 100
62 57.82 1 100
62 58.48 1 100
62 59.14 1 100
62 59.79 1 100
60 60.45 1 100
64 61.11 1 100
71 61.77 1 100
64 62.42 1 100
64 63.08 1 100
62 63.74 1 100
64 64.39 1 100
64 65.05 1 100
62 65.71 1 100
62 66.37 1 100
62 67.02 1 100
62 67.68 1 100
62 68.34 1 100
62 68.99 1 100
62 69.65 1 100
64 70.31 1 100
64 70.97 1 100
64 71.62 1 100
62 72.28 1 100
62 72.94 1 100
62 73.59 1 100
62 74.25 1 100
62 74.91 1 100
62 75.56 1 100
62 76.22 1 100
62 76.88 1 100
62 77.54 1 100
```

```
62 78.19 1 100
62 78.85 1 100
62 79.51 1 100
62 80.16 1 100
62 80.82 1 100
62 81.48 1 100
62 82.14 1 100
62 82.79 1 100
62 83.45 1 100
62 84.11 1 100
64 84.76 1 100
64 85.42 1 100
62 86.08 1 100
62 86.74 1 100
62 87.39 1 100
62 88.05 1 100
```

[29]:

```python
# AL, C Natural minor:
data2=AL_day.copy()
data2.index=list(range(len(data2)))
data2=data2.to_dict()
data2=[{'beat': d, "overall": data2[d]} for d in data2.keys()]
my_data_timed2 = [{'beat': midi2.beat(d['beat']), 'magnitude': d['overall']}␣
 ↪for d in data2]
note_list2 = []
counter=0
for d in my_data_timed2:
    try:
        note_list2.append([
            d['beat'],
            mag_to_pitch_tuned_Nmin(d['magnitude'], low_AL, high_AL),
            100,  # velocity
            1  # duration, in beats
        ])
        counter +=1
    except Exception as e:
        print(d['beat'])
        print(counter)
        print(e)
        # some notes may do not match
midi2.add_track(note_list2)
midi2.save_midi()
```

```
48 0.0 1 100
48 0.66 1 100
48 1.31 1 100
48 1.97 1 100
```

```
48 2.63 1 100
48 3.29 1 100
48 3.94 1 100
48 4.6 1 100
48 5.26 1 100
48 5.91 1 100
48 6.57 1 100
48 7.23 1 100
48 7.89 1 100
48 8.54 1 100
48 9.2 1 100
48 9.86 1 100
48 10.51 1 100
48 11.17 1 100
48 11.83 1 100
48 12.48 1 100
48 13.14 1 100
48 13.8 1 100
48 14.46 1 100
48 15.11 1 100
48 15.77 1 100
48 16.43 1 100
48 17.08 1 100
48 17.74 1 100
48 18.4 1 100
48 19.06 1 100
48 19.71 1 100
48 20.37 1 100
48 21.03 1 100
48 21.68 1 100
48 22.34 1 100
48 23.0 1 100
48 23.66 1 100
48 24.31 1 100
48 24.97 1 100
48 25.63 1 100
48 26.28 1 100
48 26.94 1 100
48 27.6 1 100
48 28.25 1 100
48 28.91 1 100
48 29.57 1 100
48 30.23 1 100
48 30.88 1 100
48 31.54 1 100
48 32.2 1 100
48 32.85 1 100
48 33.51 1 100
```

```
48 34.17 1 100
48 34.83 1 100
48 35.48 1 100
48 36.14 1 100
48 36.8 1 100
48 37.45 1 100
50 38.11 1 100
50 38.77 1 100
51 39.43 1 100
48 40.08 1 100
50 40.74 1 100
48 41.4 1 100
50 42.05 1 100
50 42.71 1 100
48 43.37 1 100
48 44.02 1 100
48 44.68 1 100
48 45.34 1 100
50 46.0 1 100
48 46.65 1 100
50 47.31 1 100
50 47.97 1 100
48 48.62 1 100
50 49.28 1 100
50 49.94 1 100
48 50.6 1 100
51 51.25 1 100
53 51.91 1 100
50 52.57 1 100
51 53.22 1 100
50 53.88 1 100
51 54.54 1 100
48 55.2 1 100
51 55.85 1 100
51 56.51 1 100
51 57.17 1 100
51 57.82 1 100
48 58.48 1 100
50 59.14 1 100
51 59.79 1 100
50 60.45 1 100
53 61.11 1 100
58 61.77 1 100
53 62.42 1 100
51 63.08 1 100
51 63.74 1 100
55 64.39 1 100
58 65.05 1 100
```

```
51 65.71 1 100
51 66.37 1 100
50 67.02 1 100
51 67.68 1 100
48 68.34 1 100
50 68.99 1 100
50 69.65 1 100
51 70.31 1 100
51 70.97 1 100
50 71.62 1 100
50 72.28 1 100
50 72.94 1 100
50 73.59 1 100
50 74.25 1 100
51 74.91 1 100
51 75.56 1 100
48 76.22 1 100
51 76.88 1 100
53 77.54 1 100
50 78.19 1 100
50 78.85 1 100
55 79.51 1 100
50 80.16 1 100
48 80.82 1 100
48 81.48 1 100
48 82.14 1 100
48 82.79 1 100
50 83.45 1 100
50 84.11 1 100
48 84.76 1 100
50 85.42 1 100
50 86.08 1 100
50 86.74 1 100
50 87.39 1 100
48 88.05 1 100
```

[30]:

```python
# SA, C Harmonic minor:
data3=SA_day.copy()
data3.index=list(range(len(data3)))
data3=data3.to_dict()
data3=[{'beat': d, "overall": data3[d]} for d in data3.keys()]
my_data_timed3 = [{'beat': midi3.beat(d['beat']), 'magnitude': d['overall']} for d in data3]
note_list3 = []
counter=0
for d in my_data_timed3:
    try:
        note_list3.append([
            d['beat'],
            mag_to_pitch_tuned_Hmin(d['magnitude'], low_SA, high_SA),
            100,  # velocity
            1  # duration, in beats
        ])
        counter +=1
    except Exception as e:
        print(d['beat'])
        print(counter)
        print(e)
        # some notes may do not match
midi3.add_track(note_list3)
midi3.save_midi()
```

```
74 0.0 1 100
74 0.66 1 100
74 1.31 1 100
74 1.97 1 100
74 2.63 1 100
74 3.29 1 100
74 3.94 1 100
74 4.6 1 100
74 5.26 1 100
74 5.91 1 100
74 6.57 1 100
74 7.23 1 100
74 7.89 1 100
74 8.54 1 100
74 9.2 1 100
74 9.86 1 100
74 10.51 1 100
74 11.17 1 100
74 11.83 1 100
74 12.48 1 100
74 13.14 1 100
74 13.8 1 100
74 14.46 1 100
74 15.11 1 100
74 15.77 1 100
74 16.43 1 100
74 17.08 1 100
74 17.74 1 100
74 18.4 1 100
74 19.06 1 100
74 19.71 1 100
74 20.37 1 100
```

```
74 21.03 1 100
74 21.68 1 100
74 22.34 1 100
74 23.0 1 100
74 23.66 1 100
74 24.31 1 100
74 24.97 1 100
74 25.63 1 100
74 26.28 1 100
74 26.94 1 100
74 27.6 1 100
74 28.25 1 100
74 28.91 1 100
74 29.57 1 100
74 30.23 1 100
74 30.88 1 100
74 31.54 1 100
74 32.2 1 100
74 32.85 1 100
74 33.51 1 100
74 34.17 1 100
74 34.83 1 100
74 35.48 1 100
75 36.14 1 100
75 36.8 1 100
74 37.45 1 100
75 38.11 1 100
77 38.77 1 100
74 39.43 1 100
74 40.08 1 100
77 40.74 1 100
75 41.4 1 100
75 42.05 1 100
75 42.71 1 100
77 43.37 1 100
75 44.02 1 100
75 44.68 1 100
74 45.34 1 100
77 46.0 1 100
75 46.65 1 100
75 47.31 1 100
74 47.97 1 100
75 48.62 1 100
74 49.28 1 100
74 49.94 1 100
75 50.6 1 100
75 51.25 1 100
75 51.91 1 100
```

```
75 52.57 1 100
75 53.22 1 100
75 53.88 1 100
79 54.54 1 100
74 55.2 1 100
75 55.85 1 100
75 56.51 1 100
75 57.17 1 100
75 57.82 1 100
74 58.48 1 100
77 59.14 1 100
75 59.79 1 100
72 60.45 1 100
77 61.11 1 100
83 61.77 1 100
75 62.42 1 100
75 63.08 1 100
75 63.74 1 100
74 64.39 1 100
75 65.05 1 100
75 65.71 1 100
74 66.37 1 100
74 67.02 1 100
75 67.68 1 100
74 68.34 1 100
74 68.99 1 100
74 69.65 1 100
77 70.31 1 100
75 70.97 1 100
74 71.62 1 100
74 72.28 1 100
75 72.94 1 100
74 73.59 1 100
75 74.25 1 100
75 74.91 1 100
74 75.56 1 100
75 76.22 1 100
75 76.88 1 100
74 77.54 1 100
74 78.19 1 100
74 78.85 1 100
75 79.51 1 100
75 80.16 1 100
75 80.82 1 100
75 81.48 1 100
74 82.14 1 100
74 82.79 1 100
74 83.45 1 100
```

```
75 84.11 1 100
75 84.76 1 100
74 85.42 1 100
74 86.08 1 100
74 86.74 1 100
75 87.39 1 100
75 88.05 1 100
```

[31]:

```
# ES, C Melodic minor:
data4=ES_day.copy()
data4.index=list(range(len(data4)))
data4=data4.to_dict()
data4=[{'beat': d, "overall": data4[d]} for d in data4.keys()]
my_data_timed4 = [{'beat': midi4.beat(d['beat']), 'magnitude': d['overall']}␣
 ↪for d in data4]
note_list4= []
counter=0
for d in my_data_timed4:
    try:
        note_list4.append([
            d['beat'],
            mag_to_pitch_tuned_Mmin(d['magnitude'], low_ES, high_ES),
            100,  # velocity
            1  # duration, in beats
        ])
        counter +=1
    except Exception as e:
        print(d['beat'])
        print(counter)
        print(e)
        # some notes may do not match
midi4.add_track(note_list4)
midi4.save_midi()
```

```
84 0.0 1 100
84 0.66 1 100
84 1.31 1 100
84 1.97 1 100
84 2.63 1 100
84 3.29 1 100
84 3.94 1 100
84 4.6 1 100
84 5.26 1 100
84 5.91 1 100
84 6.57 1 100
84 7.23 1 100
84 7.89 1 100
```

```
84 8.54 1 100
84 9.2 1 100
84 9.86 1 100
84 10.51 1 100
84 11.17 1 100
84 11.83 1 100
84 12.48 1 100
84 13.14 1 100
84 13.8 1 100
84 14.46 1 100
84 15.11 1 100
84 15.77 1 100
84 16.43 1 100
84 17.08 1 100
84 17.74 1 100
84 18.4 1 100
84 19.06 1 100
84 19.71 1 100
84 20.37 1 100
84 21.03 1 100
84 21.68 1 100
84 22.34 1 100
84 23.0 1 100
84 23.66 1 100
84 24.31 1 100
84 24.97 1 100
84 25.63 1 100
84 26.28 1 100
84 26.94 1 100
84 27.6 1 100
84 28.25 1 100
84 28.91 1 100
84 29.57 1 100
84 30.23 1 100
84 30.88 1 100
84 31.54 1 100
84 32.2 1 100
84 32.85 1 100
84 33.51 1 100
84 34.17 1 100
84 34.83 1 100
84 35.48 1 100
84 36.14 1 100
84 36.8 1 100
84 37.45 1 100
84 38.11 1 100
84 38.77 1 100
84 39.43 1 100
```

```
84 40.08 1 100
84 40.74 1 100
84 41.4 1 100
84 42.05 1 100
86 42.71 1 100
86 43.37 1 100
86 44.02 1 100
84 44.68 1 100
86 45.34 1 100
86 46.0 1 100
86 46.65 1 100
87 47.31 1 100
86 47.97 1 100
86 48.62 1 100
86 49.28 1 100
87 49.94 1 100
87 50.6 1 100
87 51.25 1 100
89 51.91 1 100
87 52.57 1 100
89 53.22 1 100
87 53.88 1 100
87 54.54 1 100
89 55.2 1 100
89 55.85 1 100
89 56.51 1 100
89 57.17 1 100
87 57.82 1 100
84 58.48 1 100
91 59.14 1 100
87 59.79 1 100
91 60.45 1 100
95 61.11 1 100
91 61.77 1 100
89 62.42 1 100
87 63.08 1 100
87 63.74 1 100
91 64.39 1 100
89 65.05 1 100
89 65.71 1 100
89 66.37 1 100
87 67.02 1 100
87 67.68 1 100
87 68.34 1 100
91 68.99 1 100
87 69.65 1 100
91 70.31 1 100
87 70.97 1 100
```

```
86 71.62 1 100
84 72.28 1 100
86 72.94 1 100
86 73.59 1 100
87 74.25 1 100
87 74.91 1 100
87 75.56 1 100
86 76.22 1 100
86 76.88 1 100
86 77.54 1 100
86 78.19 1 100
86 78.85 1 100
86 79.51 1 100
86 80.16 1 100
86 80.82 1 100
86 81.48 1 100
86 82.14 1 100
86 82.79 1 100
86 83.45 1 100
86 84.11 1 100
86 84.76 1 100
86 85.42 1 100
93 86.08 1 100
84 86.74 1 100
84 87.39 1 100
86 88.05 1 100
```

PART 2: Simple MASH UP

[32]:

```python
from mido import MidiFile
```

[33]:

```python
WY_mid=MidiFile('results/NYmix/WY.mid')
AL_mid=MidiFile('results/NYmix/AL.mid')
SA_mid=MidiFile('results/NYmix/SA.mid')
ES_mid=MidiFile('results/NYmix/ES.mid')
```

[34]:

```python
WY_mid.tracks.extend(AL_mid.tracks)
WY_mid.tracks.extend(SA_mid.tracks)
WY_mid.tracks.extend(ES_mid.tracks)
WY_mid.save('results/NYmix.mid')
```

[ ]:

```python

```