hearts

November 11, 2021

[ ]:
```
import pandas as pd
import matplotlib.pyplot as plt
from scipy.io import wavfile
from pathlib import Path
import numpy as np
from scipy import fftpack
from sklearn.preprocessing import normalize
from scipy import signal
```

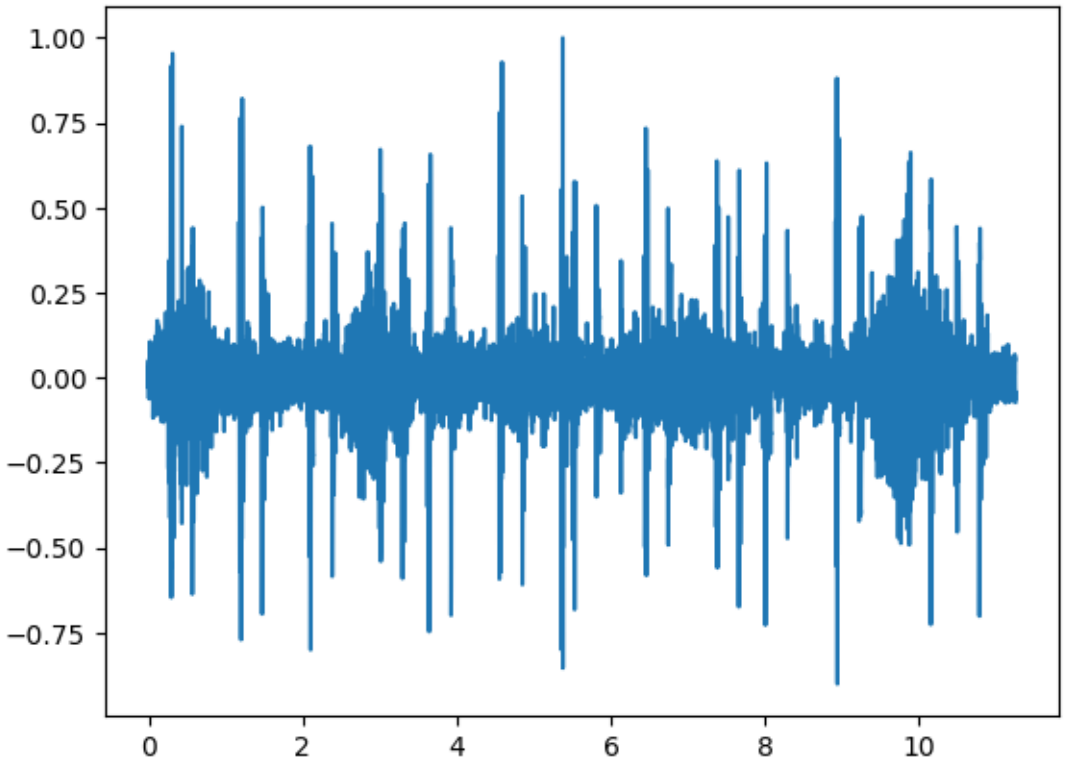
[ ]:
```
# Load wavefile
def readwav(file:str):
    filepath = Path(file).absolute()
    samplerate, data = wavfile.read((filepath))
    print(f"number of channels = {data.shape}")
    length = data.shape[0] / samplerate
    print(f"length = {length}s")
    return length,data,samplerate
```

[ ]:
```
# file="normal__201105011626.wav"
file="normal__140_1306519735121_A.wav"
timings = pd.read_csv("./heartbeats/set_a_timing.csv")
length,heart,samplerate = readwav(f"./heartbeats/wav files/{file}")
maxheart = np.amax(heart)
norm_heart = heart/maxheart
```

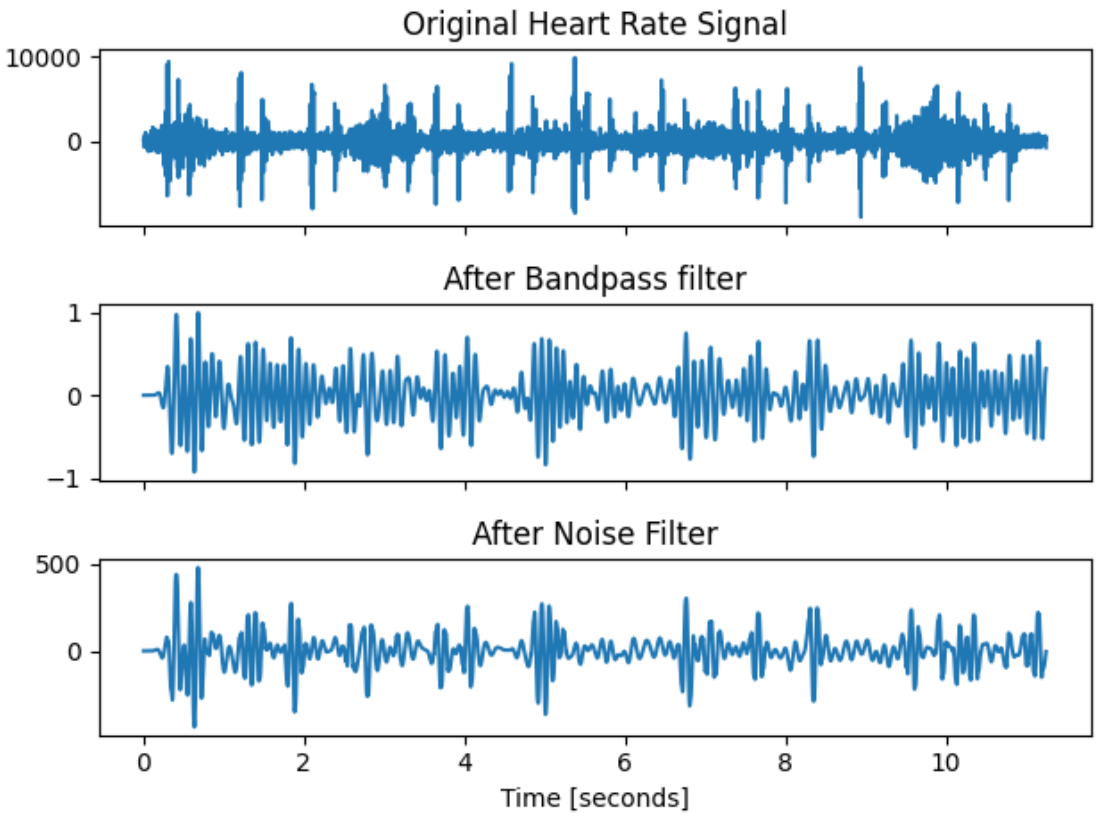
```
number of channels = (45000,)
length = 11.25s
```

[ ]:
```
t0 = 1/samplerate

t = np.arange(0,length,1/samplerate)
# t_step = t[1]-t[0]
plt.plot(t,norm_heart)
plt.show()
```

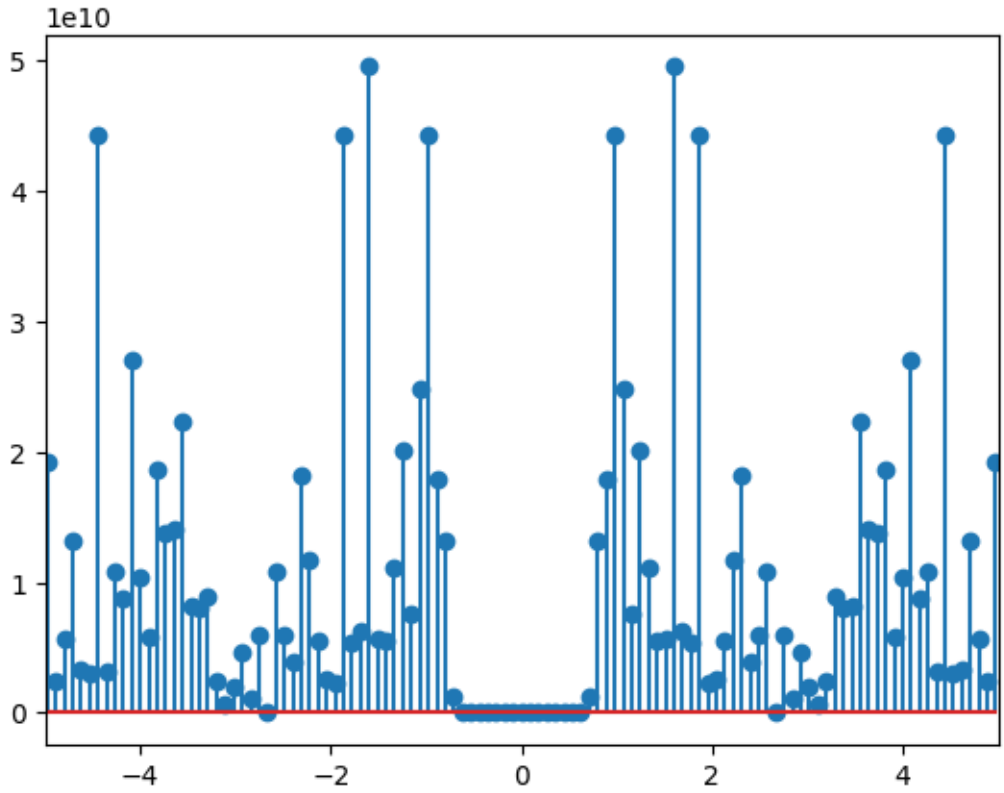


[ ]:
```
t0 = 1/samplerate
t = np.arange(0,length,1/samplerate)
sig = heart
fig, (ax1, ax2,ax3) = plt.subplots(3, 1, sharex=True)
ax1.plot(t, sig)
ax1.set_title('Original Heart Rate Signal')
sos = signal.butter(20, [.2,3], 'bp', fs=1000, output='sos')
filtered_heart = signal.sosfilt(sos, sig)
norm_heart = filtered_heart/np.amax(filtered_heart)
ax2.plot(t, norm_heart)
ax2.set_title('After Bandpass filter')
## Removing noise
norm_heart = signal.signaltools.wiener(filtered_heart,300)
ax3.plot(t, norm_heart)
ax3.set_title('After Noise Filter')
ax3.set_xlabel('Time [seconds]')
plt.tight_layout()
plt.show()
```

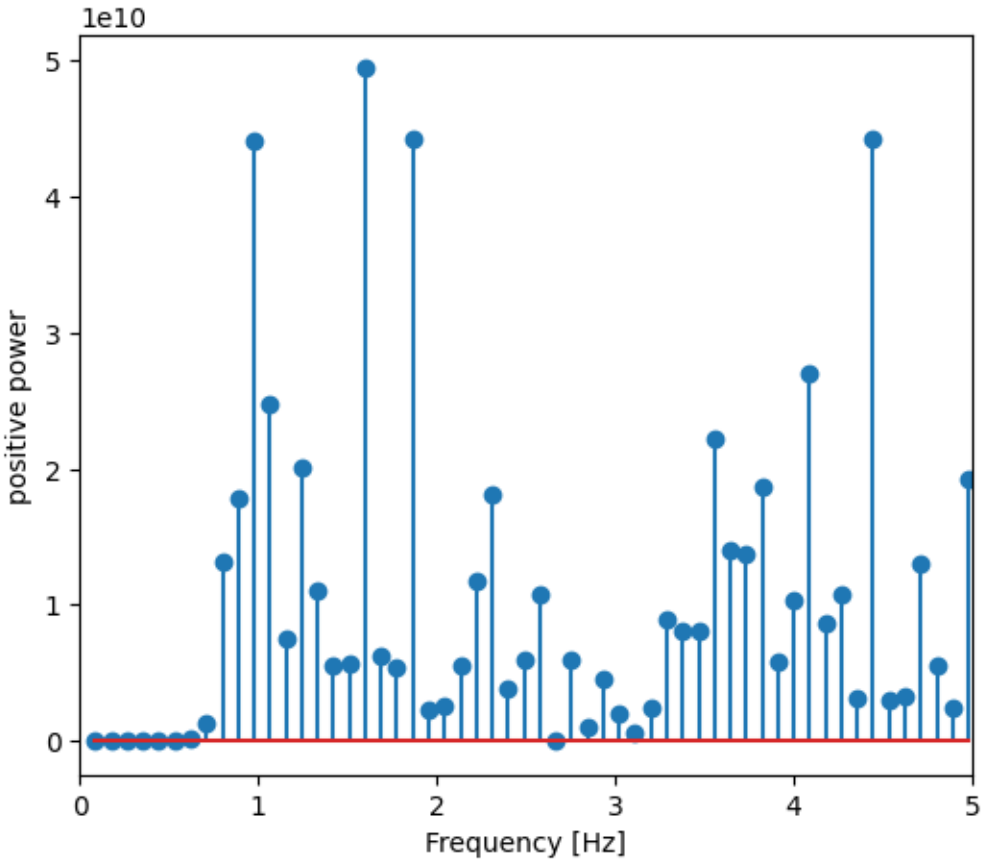


[ ]:
```
fft = fftpack.fft(norm_heart)
freqs = fftpack.fftfreq(norm_heart.size,d=t0)
getPower = lambda fft: np.abs(fft)**2
power = getPower(fft)
```

[ ]:
```
freqmask = np.abs(freqs) < 5
freqs = freqs[freqmask]
power = power[freqmask]
plt.stem(freqs,power)
plt.xlim(-5,5)
plt.show()
```



[ ]:
```
pos_mask = np.where((freqs > 0))
freqs = freqs[pos_mask]
pos_power = power[pos_mask]
maxfreq = freqs[pos_power.argmax()]
plt.figure(figsize=(6, 5))
plt.stem(freqs, pos_power)
plt.xlabel('Frequency [Hz]')
plt.ylabel('positive power')
plt.xlim(0,5)
plt.show()
```



[ ]:
```
strongest = freqs[pos_power > np.mean(pos_power) + np.std(pos_power)]
bpm = np.mean(strongest)*60
print(bpm)
```

```
140.44444444444446
```

[ ]:
```
np.mean(power)
```

[ ]:
```
10358375860.942425
```

[ ]:
```

```