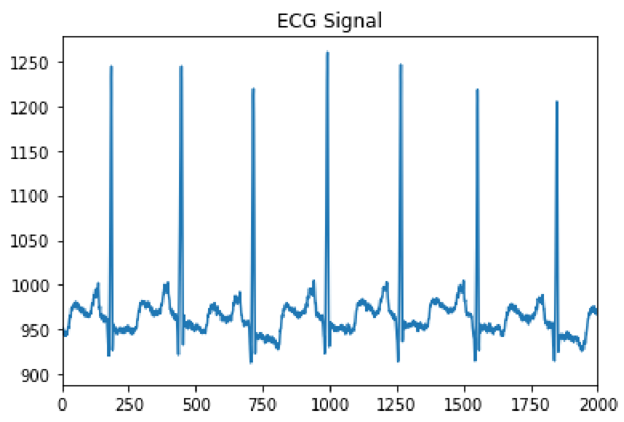
Heart Rate Monitor

1. Normal ECG

Data Source: https://data.mendeley.com/datasets/7dybx7wyfn/3

PART 1: Importing the dataset, required libraries and plotting the data.

```
#Importing all required libraries

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import math

import os

from scipy.io import loadmat

#Loading the .mat file. Change the path if running.

os.chdir("J:\\MTech Notes\\Biomedical devices\\Assignment 2\\MLII\\1 NSR")

x= loadmat('100m (0).mat')

print(x)

#Converting dict to dataframe for simplifying the calculation part

df=pd.DataFrame([x])

df=df.explode('val').explode('val')

df.reset_index(drop=True, inplace=True)

#displaying the plot of raw data

plt.title("ECG Signal")

plt.plot(df.val)

plt.xlim(0,2000)

plt.show()
```



PART 2: Filtering the data: Here as the signal is already clean and free of noise thus not using any filter and directly trying for peak detection.

PART 3: We are trying to find running average which is a series of averages of the subsets of main dataset.

#Define a window which is the subset of the data we are choosing. I took 0.25 after going through multiple values. Some values give flat curve which were ignored. 0.25 gave a similar curve to original curve and hence 0.25 chosen as window size. Sampling frequency is known.

```
hw=0.25

fs=360

moving_average=df['filtered_data'].rolling(int(hw*fs)).mean()
```

#Finding the average of whole dataset.

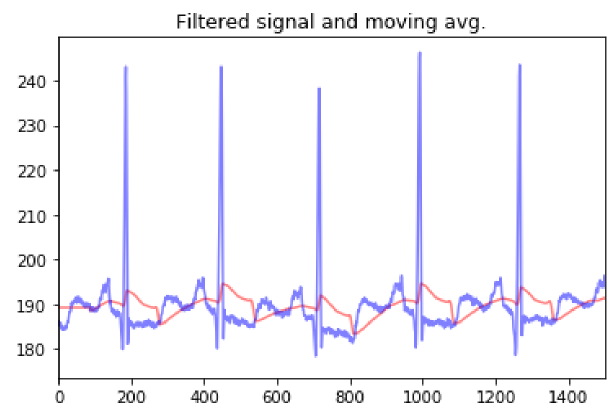
```
average=np.mean(df.filtered_data)
```

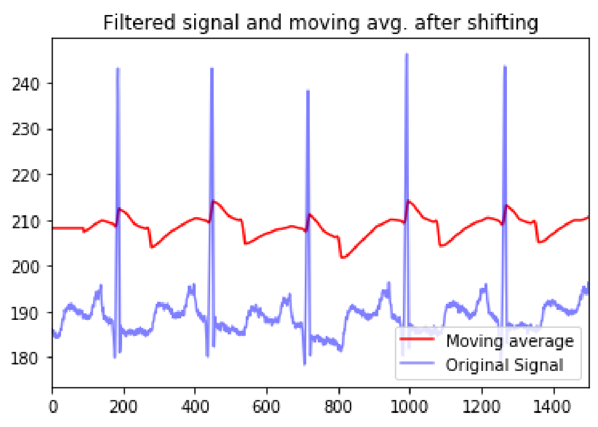
#In the start of running average values are NAN hence replacing such values with average of the whole dataset.

```
moving_average=[average if math.isnan(x) else x for x in moving_average]
```

#If we directly use the moving average as given in 1st figure for peak detection both R and other peaks above the moving average will be detected. Therefore, we will multiply with 1.1 to upscale and shift the average value so that it is above all peaks except R peaks for proper detection.

```
moving_average=[x*1.1 for x in moving_average]

df['roll_mean']=moving_average
```



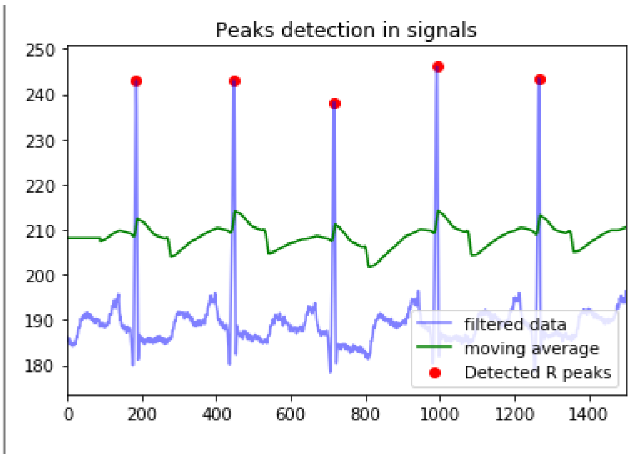


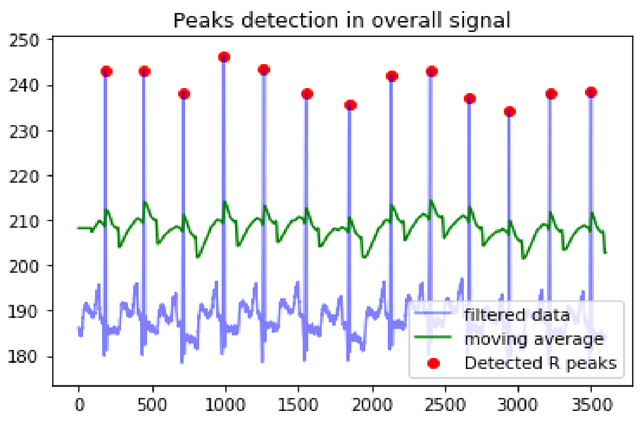
#Our goal is to find the peak in the region above the moving average and store the points to plot later.

```
w=[]

peak=[]

cnt=0

for x in df.filtered_data:

    x_mean=df.roll_mean[cnt]

    if(x <= x_mean) and (len(w) <=  1):
        cnt += 1            #when less than average counter just gets incremented without doing anything.
    elif(x>x_mean):
        w.append(x)      #append to list when greater
        cnt +=1
    else:
        maxi=max(w)
        pos=cnt-len(w)+w.index(max(w))   #calculating the position of peak point
        peak.append(pos)    #add the pos to peak list
        w=[]     #reset the value
        cnt += 1   #increment the counter
y_pos= [df.filtered_data[x] for x in peak]
```

#Plotting the detected peaks. Applying the x-limit for better view of graph

```
plt.title("Peaks detection in signals")
plt.xlim(0,1500)
plt.plot(df.filtered_data,alpha=0.5,color='blue')
plt.plot(moving_average,color='green')
plt.scatter(peak,y_pos,color='red')
plt.legend(loc=4)
plt.show()
```





#Calculate heart rate: Finding the R peak difference then dividing by sampling frequency and mul by 1000 to get bpm value.

```
rr_int_list=[]
cnt=0
while(cnt<len(peak)-1):
    rr_int=(peak[cnt+1]-peak[cnt])
    x=(rr_int/fs)*1000
    rr_int_list.append(x)

    cnt += 1

h_rate=60000/np.mean(rr_int_list)

print("Average heart rate of the person is: ", h_rate)
```

O/P: Average heart rate of the person is: 78.19004524886877

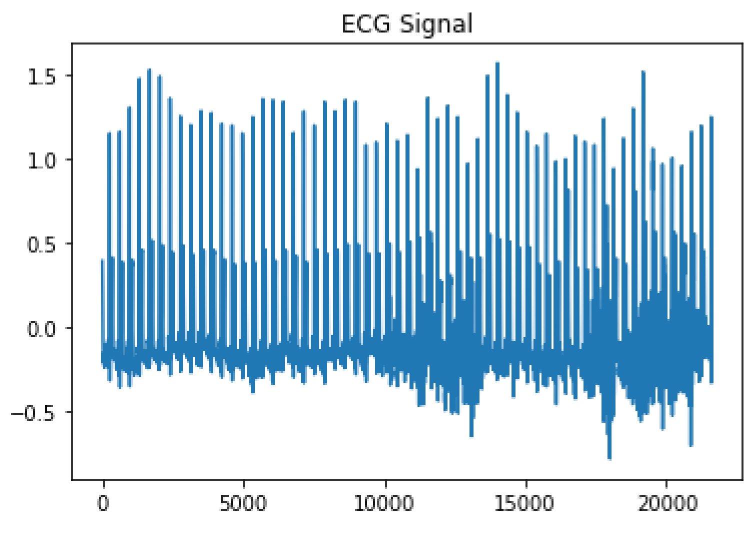
Outcome: All peaks detected correctly.

STRESS ECG

Data Source: https://physionet.org/content/stdb/1.0.0/

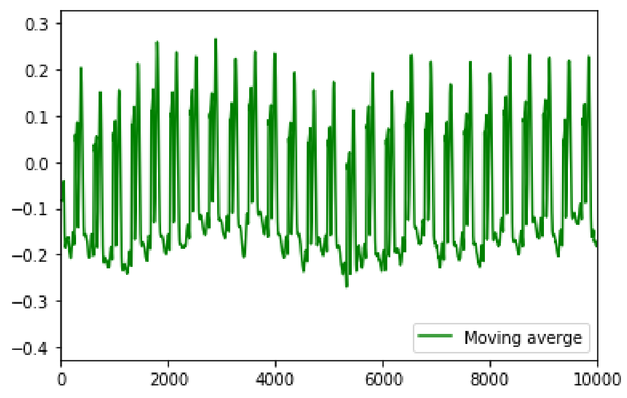
#Importing required libraries and loading csv data

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import math

import os

os.chdir("J:\\MTech Notes\\Biomedical devices\\Assignment 2")

data= pd.read_csv("stress_ecg_noise.csv")

df=data.iloc[0:50001,1:2]

plt.title("ECG Signal")

plt.plot(df['val'])

plt.show()
```



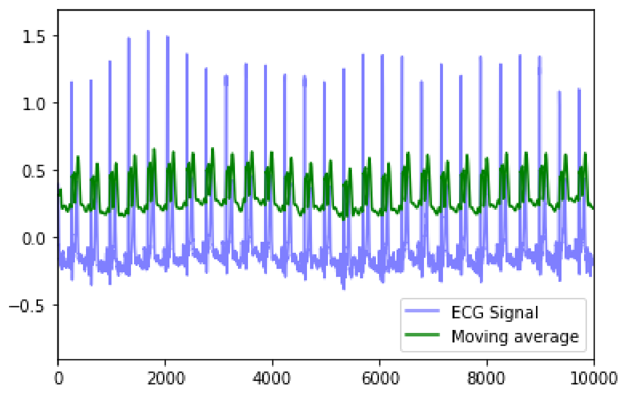
```
hw=0.15

fs=360

moving_average=df['val'].rolling(int(hw*fs)).mean()

average=np.mean(df.val)

moving_average=[average if math.isnan(x) else x for x in moving_average]

#moving_average=[x*8.1 for x in moving_average]

df['roll_mean']=moving_average

plt.plot(df.roll_mean,color='green',label='Moving averge')

plt.legend(loc=4)

plt.xlim(0,10000)
```

```
Out[135]: (0, 10000)
```



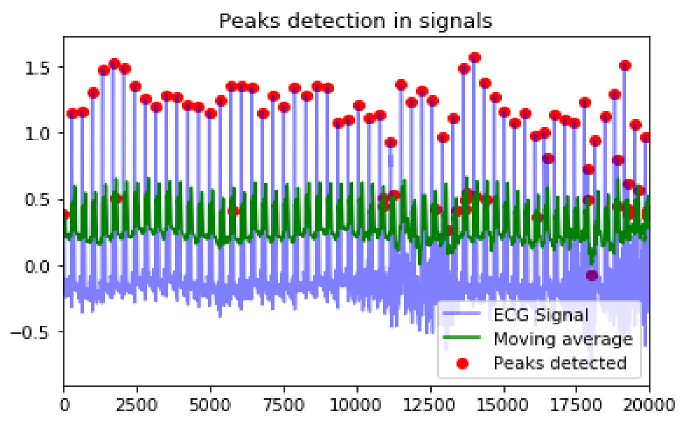
#Shifting above 0 axis

```
df['roll_mean']=df['roll_mean']+abs(df['roll_mean'].min())

abs(df['roll_mean'].max())

plt.plot(df.val,alpha=0.5,color='blue',label='ECG Signal')

plt.plot(df.roll_mean,color='green',label='Moving average')

plt.legend(loc=4)

plt.xlim(0,10000)
```



#Peak detection code same as before

```
w=[]
peak=[]
cnt=0
for x in df.val:
   x_mean=df.roll_mean[cnt]
   if(x <= x_mean) and (len(w) <=  1):
      cnt += 1
   elif(x > x_mean):
      w.append(x)
      cnt +=1
   else:
      maxi=max(w)
      pos=cnt-len(w)+w.index(max(w))
      peak.append(pos)
      w=[]
      cnt += 1
y_pos= [df.val[x] for x in peak]
plt.title("Peaks detection in signals")
plt.xlim(0,20000)
plt.plot(df.val,alpha=0.5,color='blue',label='ECG Signal')
plt.plot(df.roll_mean,color='green',label='Moving average')
plt.scatter(peak,y_pos,color='red',label='Peaks detected')
plt.legend(loc=4)
plt.show()
```

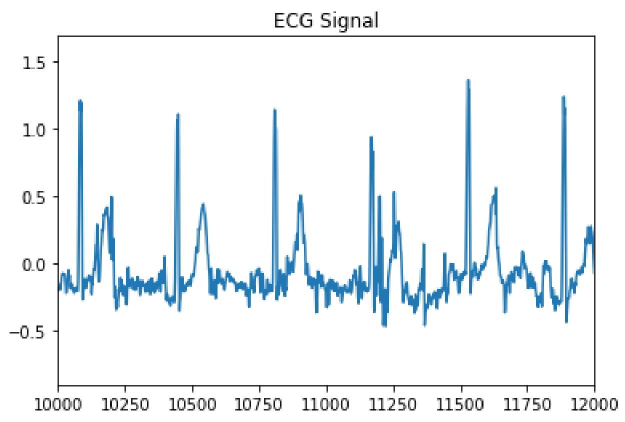


As shown in the above plot the code is detecting some incorrect peaks. Along with R peak its detecting T peaks also.

Hence, we try to improve the moving average and its position on the graph so that its above the T peak and below R peak.

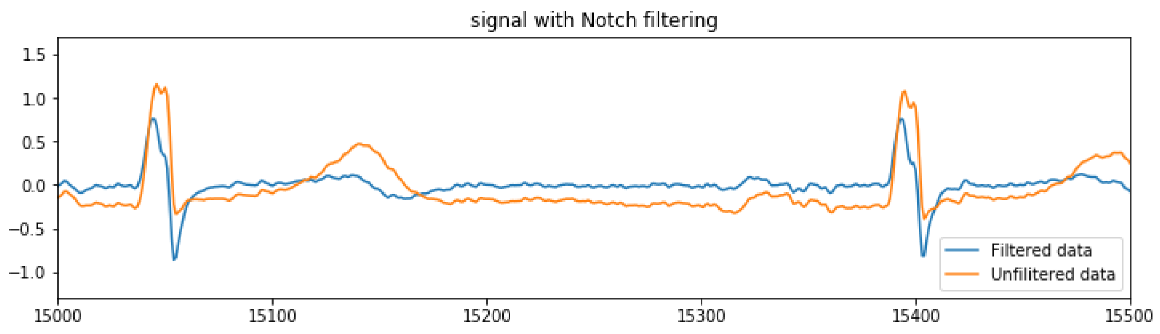
Improved code:

1. Introduction of notch filter to clear out the noise. Baseline Wander effect can be reduced using notch filter. As the data is noisy and one can see lot of interfering frequencies therefore notch filter will suit the best for this type of data. The frequency content of the baseline wander is in the range of 0.5 Hz. (ref: http://www.jscholaronline.org/articles/JBER/Signal-Processing.pdf)



```
from scipy.signal import iirnotch,lfilter

def filter_signal(data, cutoff, sample_rate, order=2, filtertype='notch',

              return_top = False):

  b, a = iirnotch(cutoff, Q = 0.005, fs = sample_rate)

  return b,a

def notch_filter(data, sample_rate, cutoff=0.05):

  b,a= filter_signal(data = data, cutoff = cutoff, sample_rate = sample_rate,

                  filtertype='notch')

  z = lfilter(b,a,data)
```

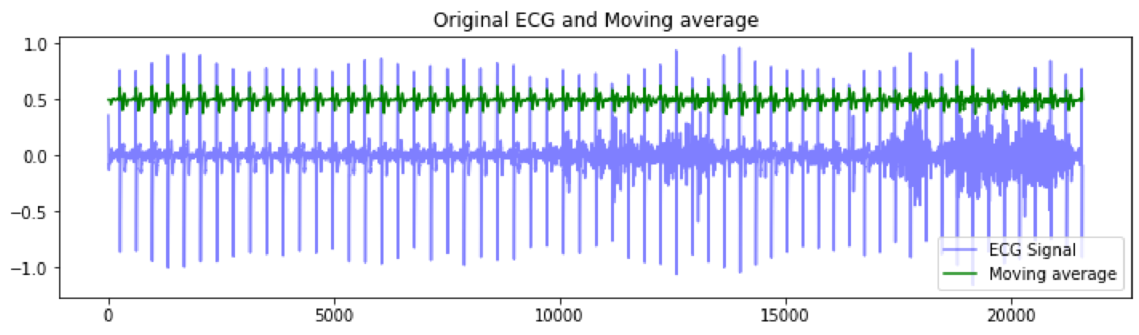
```
    return z
df['filtered_data'] = notch_filter(df.val, cutoff = 0.05, sample_rate = 360.0)


# Plotting the filtered data
plt.figure(figsize=(12,3))
plt.title(‘signal with Notch filtering’)
plt.plot(df.filtered_data,label='Filtered data')
plt.plot(df.val,label='Unfilitered data')
plt.legend(loc=4)
plt.xlim(15000,15500)
plt.show()
```



One can observe the difference between filtered and unfiltered data after filtering has been done.

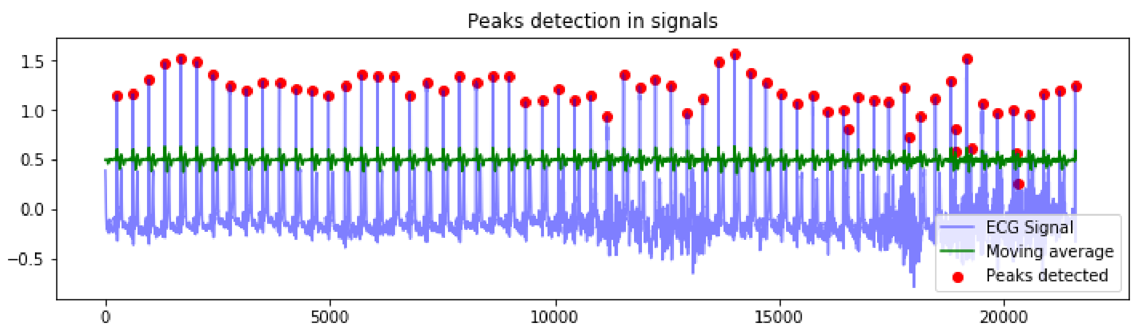
Using same code as above but changing the window to 0.15 which is giving better result.

```
#Same code as before
hw=0.15
fs=360
moving_average=df['filtered_data'].rolling(int(hw*fs)).mean()
average=np.mean(df.filtered_data)
moving_average=[average if math.isnan(x) else x for x in moving_average]
df['roll_mean']=moving_average
plt.plot(df.roll_mean,color='green',label='Moving averge')
plt.legend(loc=4)
plt.xlim(19000,20000)
#Shifting the moving average above by 0.5 to detect the peaks correctly.
df['roll_mean']=df['roll_mean']+0.5
plt.plot(df.filtered_data,alpha=0.5,color='blue',label='ECG Signal')
plt.plot(df.roll_mean,color='green',label='Moving average')
plt.legend(loc=4)
```



#Peak detection code same as before:

```
w=[]

peak=[]

cnt=0

for x in df.val:

   x_mean=df.roll_mean[cnt]

   if(x <= x_mean) and (len(w) <=  1):

      cnt += 1

   elif(x > x_mean):

      w.append(x)

      cnt +=1

   else:

      maxi=max(w)

      pos=cnt-len(w)+w.index(max(w))

      peak.append(pos)

      w=[]

      cnt += 1

y_pos= [df.val[x] for x in peak]

plt.figure(figsize=(12,3))

plt.title("Peaks detection in signals")

plt.plot(df.val,alpha=0.5,color='blue',label='ECG Signal')

plt.plot(df.roll_mean,color='green',label='Moving average')

plt.scatter(peak,y_pos,color='red',label='Peaks detected')

plt.legend(loc=4)

plt.show()
```



Outcome: We can see 2 incorrectly detected peaks (in the last segment).

Improvements that can be done:

1. Find a way to decide optimum value for window when calculating the running average. Here it was done using trial and error method.
2. Same goes with the position of moving average. As in case of stress ECG it is very easy for T peak to interrupt with R peak. Hence a way has to be devised to dynamically find the position of moving average and up to what scale it can be improved.
3. Filtering part can be improved by working on different types of noise data set.