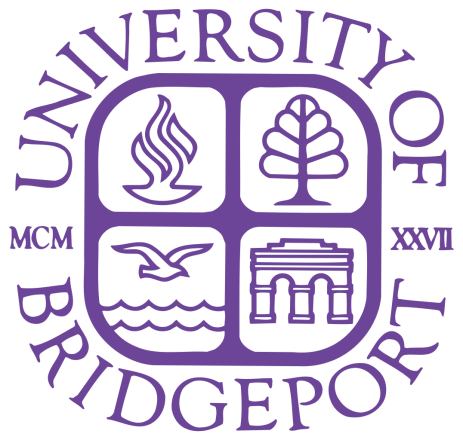
****

**CPEG-597C-MF(6200): Project III**

**Fall 2017 Computer Engineering Final Project**

**Heart Rate Analyzer using Python**

**Duhita S. Dharmadhikari**

**UB ID: 1022255**

[**ddharmad@my.bridgeport.edu**](mailto:ddharmad@my.bridgeport.edu)

**Adviser: Dr Miad Faezipour**

|  |  |
| --- | --- |
| **SR NO** | **TOPICS** |
| 1. | Introduction |
| 2. | Design and Implementation |
| 3. | Code |
| 4. | Future work |
| 5. | Conclusion |

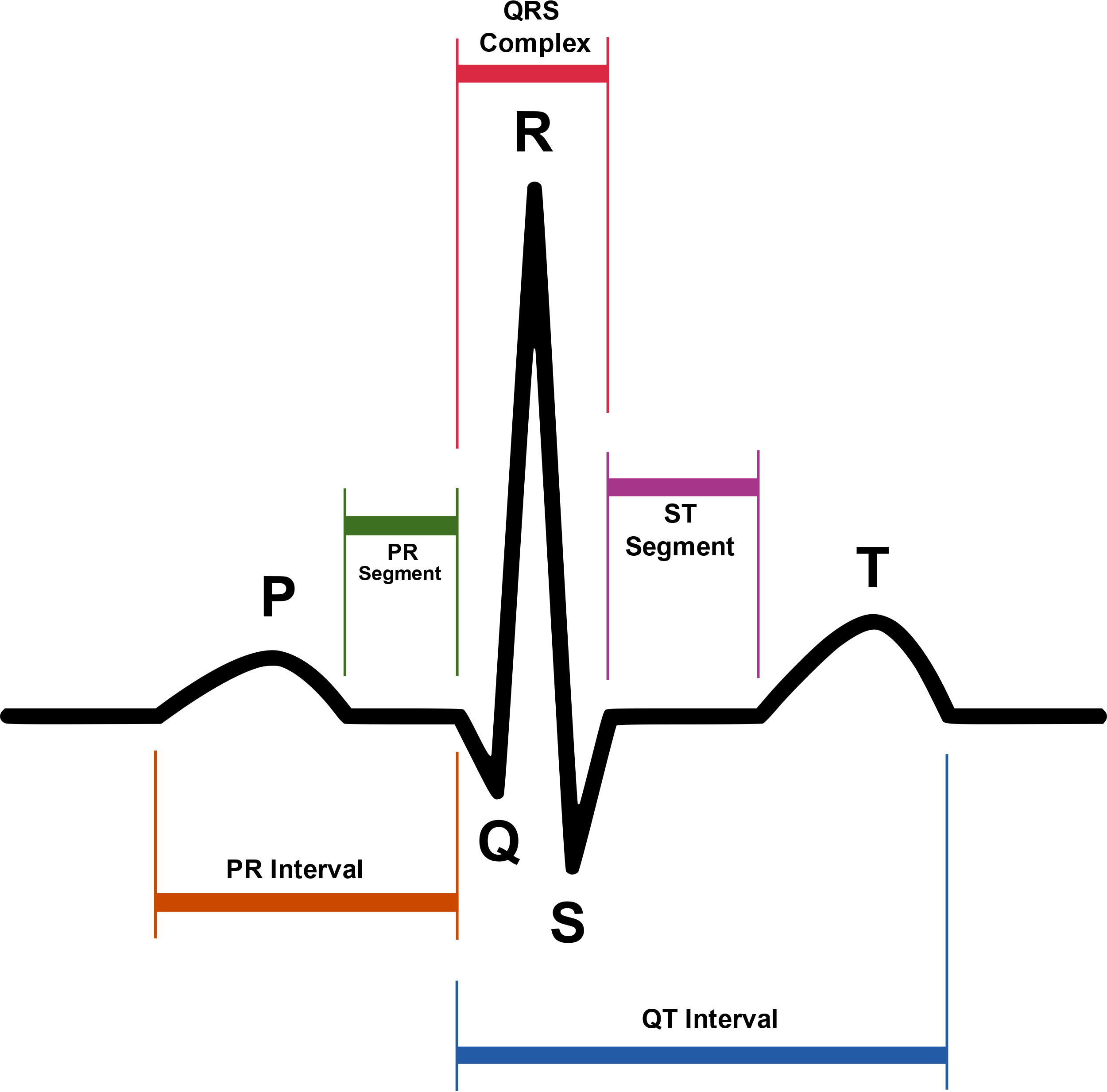
**TABLE OF CONTENTS**

**Introduction**

The heart rate of any given person can determine not only its beats per minute but also your physical health and fitness. With the help of the right technology, we will soon be able to determine the risk factors leading to the Heart attack or Myocardial Infraction and cause fatality. In order to help doctors and physicians understand the health of your heart better, I have created a program using Python to understand and analyze your heart beat (van Gent, 2016).

This program will take the Raw ECG data as the input and with careful consideration, it will produce the bpm, Time series measures such as IBI(Interbeat interval), SDSD(Standard deviation of an interval), RMSSD(Root mean square of the successive difference between adjacent RR intervals). pNNx. These measures are often called Heart Rate Variability measures (HRV).

Let's look at the heartbeat briefly:

****

The above figure represents a QRS complex. On the right side of the QRS complex is T wave, and on the left side of the QRS complex, there is P wave. The in-between segments are called as ST segment, QT interval, PR interval, PR segment. Any abnormal conduction in the heartbeat causes the QRS complex to widened. The time, amplitude and morphology of the QRS complex are used to diagnose severe diseases such as Cardiac arrhythmias, Ventricular hypertrophy, myocardial infarction and other disease states (al., 2018).

The EKG or ECG is obtained using a 12-lead ECG, ten electrodes. The P wave represents the depolarization of the atria. Whereas, QRS complex represents the depolarization of the Ventricles. The T wave represents the repolarization of the Ventricles. For every heartbeat there is an orderly progression of depolarization and so on.

**Design and Implementation**

**1.Libraries:**

We start by importing scientific libraries in our project module. In order to perform certain operations, we need to import those libraries. They include Pandas, NumPy, Scikitlearn, Math, Matplotlib.pyplot, datetime.

**2.Loading the data:**

We then start by loading the appropriate data, by appropriate I mean CSV (Comma Separated Values). Make sure that the data you gather is pre-processed as well as labelled data. To work on unlabeled data is out of the scope of this project.

We load the data file by using panda’s library “pd.read\_csv” function.

**3. Describing the data:**

In this step, we use various functions to extract the description, information and heads and tails of the data presented to us. It is important to do this step in order to get the hang of the data we are working with.

We use functions like “dataset.describe”, “dataset.info()”, “dataset.head()”, “dataset.hist()”

**4. Determining the RR Interval:**

This step is important to the project. Here we determine the RR interval and moving average of the heart rate, we calculate the peak list and list position in this step. We detect each peak and append the peak to the peak list.

**5. Calculating BPM, Displaying RR list and RR interval:**

After appending all the detected peaks to the peak list, we then calculate the BPM, RR list and RR interval

**6. Plotting:**

In this step we plot the ECG with moving average, detected peaks (in red), RR difference, RR square difference. We also display the time domain measures of IBI, SDNN, SDSD, RMSSD, Pnn50, Pnn20 using functions such as “np.std()”, “np.mean()”, “np.sqrt()” and accuracy.

**7. Reducing noise:**

We use Low pass Butter filter; it is ideally considered best filter to filter out the noise. Using the SciPy library we import Butter filter and input the

**Code**

import numpy as np

import pandas as pd

import math

import matplotlib.pyplot as plt

from datetime import datetime

measures = {}

dataset = pd.read\_csv(r"C:\Users\Dee1\PycharmProjects\Sklearn\data.csv")

hrw = 0.75

fs = 100

dataset.describe()

dataset.info()

dataset.head()

dataset.cumsum

plt.figure()

plt.show()

dataset.tail()

dataset.plot()

dataset.hist()

mov\_avg = dataset['hart'].rolling(int(hrw\*fs)).mean()

avg\_hr = (np.mean(dataset.hart))

mov\_avg = [avg\_hr if math.isnan(x) else x for x in mov\_avg]

# mov\_avg = [x\*1.2 for x in mov\_avg] #For now we raise the average by 20% to prevent the secondary heart contraction from interfering, in part 2 we will do this dynamically

dataset['hart\_rollingmean'] = mov\_avg #Append the moving average to the dataframe

#Mark regions of interest

window = []

peaklist = []

listpos = 0 #We use a counter to move over the different data columns

for datapoint in dataset.hart:

rollingmean = dataset.hart\_rollingmean[listpos] #Get local mean

if (datapoint <= rollingmean) and (len(window) <= 1): #If no detectable R-complex activity -> do nothing

listpos += 1

elif (datapoint > rollingmean): #If signal comes above local mean, mark ROI

window.append(datapoint)

print("datapoints are :", datapoint)

listpos += 1

else: #If signal drops below local mean -> determine highest point

maximum = max(window)

beatposition = listpos - len(window) + (window.index(max(window))) #Notate the position of the point on the X-axis

peaklist.append(beatposition)#Add detected peak to list

print("beat position is : ", beatposition)

window = [] #Clear marked ROI

listpos += 1

# CALCULATE BPM: DISPLAY RR LIST, RR INTERVAL, AVG BPM

rr\_list = []

cnt = 0

while (cnt < (len(peaklist)-1)):

rr\_interval = (peaklist[cnt+1] - peaklist[cnt]) #Calculate distance between beats in # of samples

ms\_dist = ((rr\_interval / fs) \* 1000.0) #Convert sample distances to ms distances

rr\_list.append(ms\_dist) #Append to list

cnt += 1

bpm = 60000 / np.mean(rr\_list) #60000 ms (1 minute) / average R-R interval of signal

print("rr list is : ", rr\_list)

print("rr intervals are : ", rr\_interval)

print("Average Heart Beat is: %.01f" %bpm)

# PLOTTING :

ybeat = [dataset.hart[x] for x in peaklist] #Get the y-value of all peaks for plotting purposes

plt.title("Detected peaks in signal")

plt.xlim(0,2500)

plt.plot(dataset.hart, alpha=0.5, color='blue', label = "raw signal")

# PLOTTING MOVING AVERAGE:

plt.plot(mov\_avg, color ='green', label = "moving average")

# PLOTTING DETECTED PEAKS:

plt.scatter(peaklist, ybeat, color='red', label = "average : %.1f BPM"%bpm)

plt.legend(loc= 4, framealpha= 0.6)

plt.show()

# CALCULATING THE RR DIFFERENCE AND RR SQUARE DIFFERENCE:

rr\_diff = []

rr\_sqdiff = []

cnt = 1 #Use counter to iterate over RR\_list

while (cnt < (len(rr\_list)-1)): #Keep going as long as there are R-R intervals

rr\_diff.append(abs(rr\_list[cnt] - rr\_list[cnt+1])) #Calculate absolute difference between successive R-R interval

rr\_sqdiff.append(math.pow(rr\_list[cnt] - rr\_list[cnt+1], 2)) #Calculate squared difference

cnt += 1

print("rr difference is : ",rr\_diff)

print("rr square difference is : ", rr\_sqdiff)

# CALCULATING ALL TIME DOMAIN MEASURES:

ibi = np.mean(rr\_list) #Take the mean of RR\_list to get the mean Inter Beat Interval

print("IBI:", ibi,"ms")

sdnn = np.std(rr\_list) #Take standard deviation of all R-R intervals

print("SDNN:", sdnn,"ms")

sdsd = np.std(rr\_diff) #Take standard deviation of the differences between all subsequent R-R intervals

print("SDSD:", sdsd,"ms")

rmssd = np.sqrt(np.mean(rr\_sqdiff)) #Take root of the mean of the list of squared differences

print("RMSSD:", rmssd,"ms")

nn20 = [x for x in rr\_diff if (x>20)] #First create a list of all values over 20, 50

nn50 = [x for x in rr\_diff if (x>50)]

pnn20 = float(len(nn20)) / float(len(rr\_diff)) #Calculate the proportion of NN20, NN50 intervals to all intervals

pnn50 = float(len(nn50)) / float(len(rr\_diff)) #Note the use of float(), because we don't want Python to think we want an int() and round the proportion to 0 or 1

Accuracy = (100 - 7.8) #100 - (real heart beat - bpm)

print("pNN20: ", pnn20,"%")

print("pNN50: ", pnn50,"%")

print("Accuracy: ", Accuracy, "%")

# CALCULATING

from scipy.signal import butter,lfilter

def butter\_lowpass(cutoff, fs, order=5):

nyq = 0.5 \* fs #Nyquist frequeny is half the sampling frequency

normal\_cutoff = cutoff / nyq

b,a = butter(order, normal\_cutoff, btype='low', analog=False)

return b,a

def butter\_lowpass\_filter(data, cutoff, fs, order):

b, a = butter\_lowpass(cutoff, fs, order=order)

y = lfilter(b, a, data)

return y

dataset = pd.read\_csv("data.csv")

dataset = dataset[600:12000].reset\_index(drop=True) #For visibility take a subselection of the entire signal from samples 6000 - 12000 (00:01:00 - 00:02:00)

filtered = butter\_lowpass\_filter(dataset.hart, 2.5, 100.0, 5)#filter the signal with a cutoff at 2.5Hz and a 5th order Butterworth filter

#Plot it

plt.subplot(211)

plt.plot(dataset.hart, color='Blue', alpha=0.5, label='Original Signal')

plt.legend(loc=4)

plt.subplot(212)

plt.plot(filtered, color='Red', label='Filtered Signal')

plt.ylim(200,800) #limit filtered signal to have same y-axis as original (filter response starts at 0 so otherwise the plot will be scaled)

plt.legend(loc=4)

plt.show()

**Future Work**

For future enhancements, I would design a model using Deep learning. Train it , evaluate it and test my model. Using Python libraries such as TensorFlow, Keras, Scikit learn I would design an intelligent system which will be able to predict the HRV measures and provide a very detailed analysis of the health of a human being.

**Conclusion**

Using python and its scientific libraries we are able to predict the HRV measures about 92.2%. The aim of the project was to determine the HRV analysis as accurate as possible and make this as easy as possible for future generation to continue and work on the project.

# References

al., A. L. (2018, Jan 10). Retrieved from https://www.ncbi.nlm.nih.gov: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5805987/

van Gent, P. (2016). *Analyzing a Discrete Heart Rate Signal Using Python*. Retrieved from http://www.paulvangent.com: http://www.paulvangent.com/2016/03/15/analyzing-a-discrete-heart-rate-signal-using-python-part1-/