**Penn State University**

**Project 1 Report**

**Faviana Troshani**

**CMPSC 463**

**Professor Yang**

**October 25, 2025**

**TABLE OF CONTENTS**

| **1** | **DESCRIPTION OF PROJECT** | **3** |
| --- | --- | --- |
| **2** | **INSTALLATION AND USAGE** | **4** |
| **3** | **STRUCTURE OF CODE** | **5** |
| **4** | **DESCRIPTION OF ALGORITHM** | **6** |
| **5** | **EXECUTION RESULTS AND DISCUSSION** | **8** |
| **6** | **CONCLUSIONS** | **10** |
|  |  |  |

**DESCRIPTION OF PROJECT**

For this project, we were required to design a system that performs unsupervised clustering of time-series extracted from PulseDB, using divide-and-conquer strategies and algorithmic reasoning. The goal was to group similar physiological signal segments and analyze their internal structure using closest pair algorithms and maximum subarray analysis. This project emphasizes how algorithmic design can drive meaningful segmentation and interpretation of biomedical time-series data.

The key components and algorithms that this project required are:

1. Divide-and-Conquer Clustering of Time Series Segments
2. Closest Pair of Time Series Within Clusters
3. Maximum Subarray Analysis (Kadane’s Algorithm)
4. PulseBD (for the segments)

**INSTALLATION AND USAGE**

Divide and Conquer:

Divide and Conquer computes pairwise similarity between all time series using correlation distance, splits the set recursively if the average distance exceeds a threshold, and each recursion refines clusters until internal similarity is high. The recursion divides data by average cohesion and each recursive split isolates similar signals.

Closest Pair of Time Series:

The closest pair of time series finds the two most similar segments inside each cluster and measures cluster cohesion. It helps validate whether a cluster is well-formed or not. This will also help us find the closest pair distance.

Maximum Subarray Analysis (Kadane’s Algorithm):

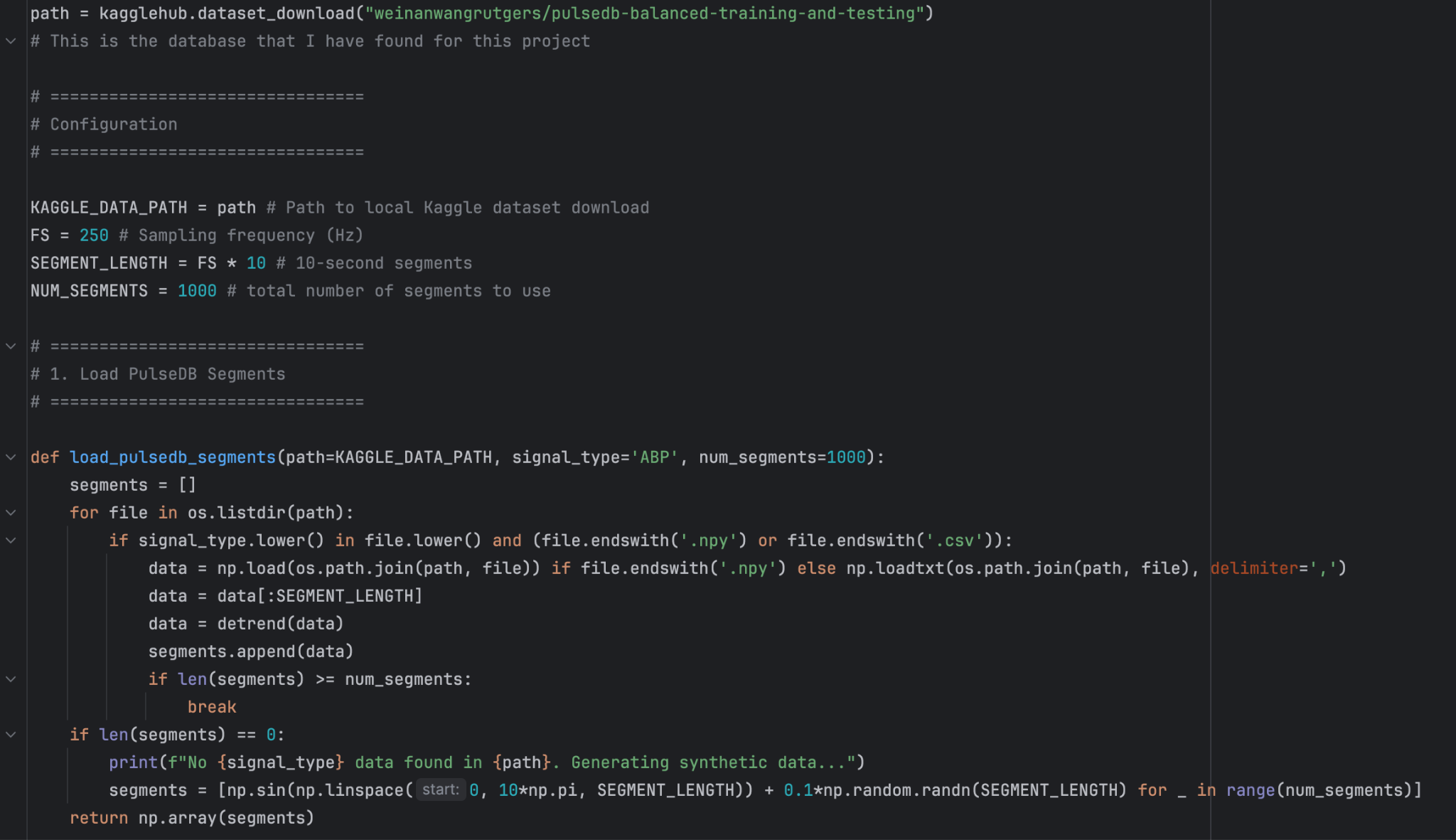
Kadane’s Algorithm identifies the most active or anomalous interval in each signal and efficiently finds the subarray with the maximum cumulative value. It also helps interpret why signals belong to a cluster. For our code, it helps find the start and end of the max activity interval and the max sum of the subarray.

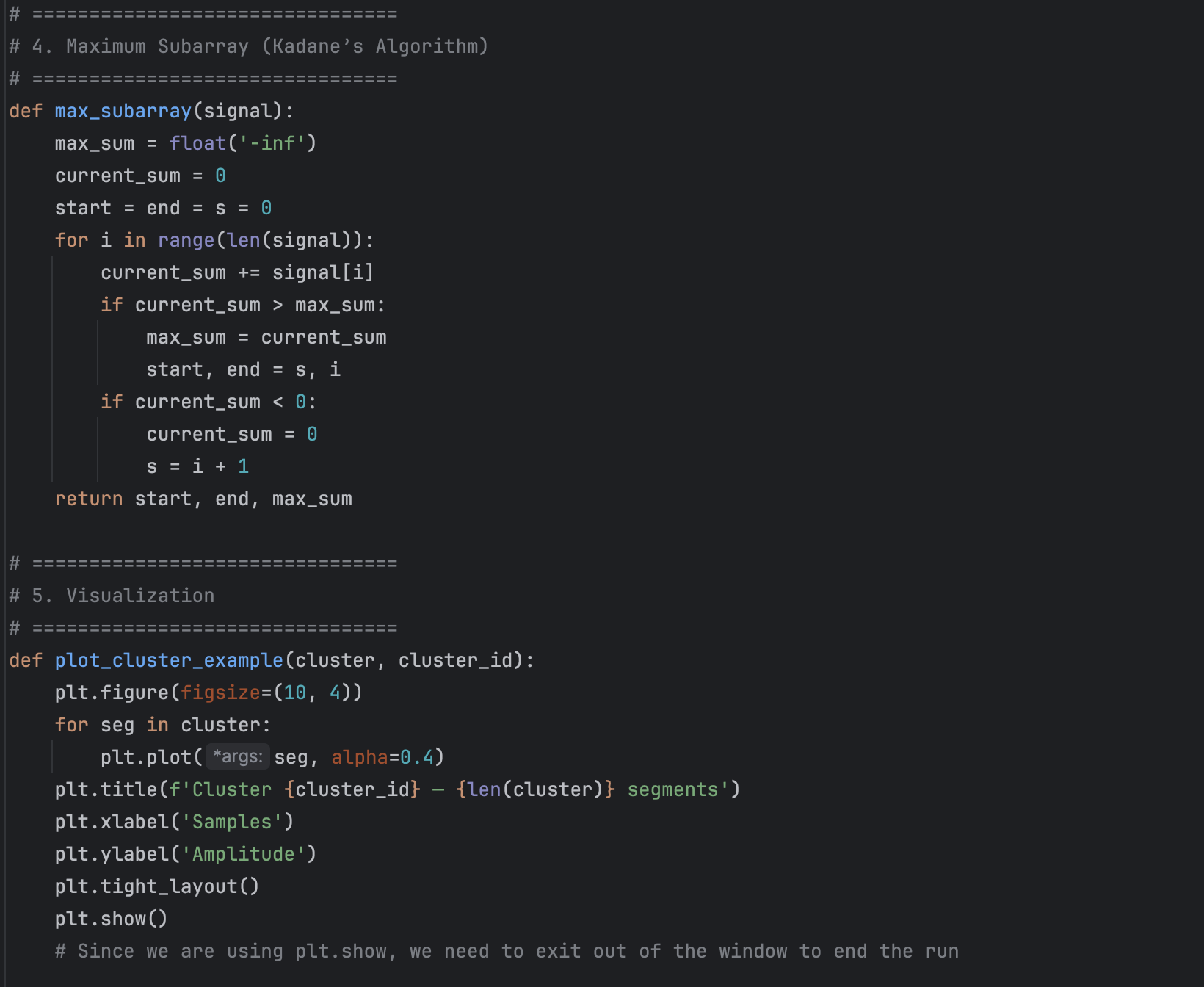
Visualization:

For the code, I use matplotlib to plot all time series in a cluster to properly visualize their similarities.

**STRUCTURE OF CODE**

Example Screenshots (the full code will be found in the Github):





**DESCRIPTION OF ALGORITHM**

With this project, the imports are extremely important. To help me complete the code, I imported os, numpy, matplotlib, scipy, and kagglehub (don’t forget to install them into your terminal). The PulseDB that I found required a hefty download size so I also included in the program to generate segments/clusters instead just in case someone would prefer to avoid the downloading process. I also added a download section to the database that I have found on kagglehub as well as a configuration section to easily update the data information.

For the first important section of the code, we have to load the PulseDB segments. To do this, I created a function that will take a path to the database that you are planning on using which can easily be changed in the configuration section. This is also where you implement the number of segments that you want, this case being 1,000. If the segments for the database are not found, numpy will instead generate new segments to be used for the code so as to not throw an error.

The next section is the divide-and-conquer clustering where I have made a function to find any similarities which is used in the recursive function I made where the base case simply returns the segments while the recursive part divides the clusters into sections many times until it either stops splitting when the clusters are similar or continues until the segments are stable. This is effective because it produces an unsupervised clustering without the need of machine learning. This function works as the divide-and-conquer key component needed in this project.

The closest pair algorithm is the next section of the code as well as the next key component of the project. This algorithm helps us find the two most similar signals (aka the pairs closest to one another) which is the core structure of the cluster as well as the degree of cohesion. It compares every pair of signals to each other and it’s important because finding the pairwise distance will indicate whether or not the cluster is strongly cohesive or will benefit from more recursive splitting/filtering.

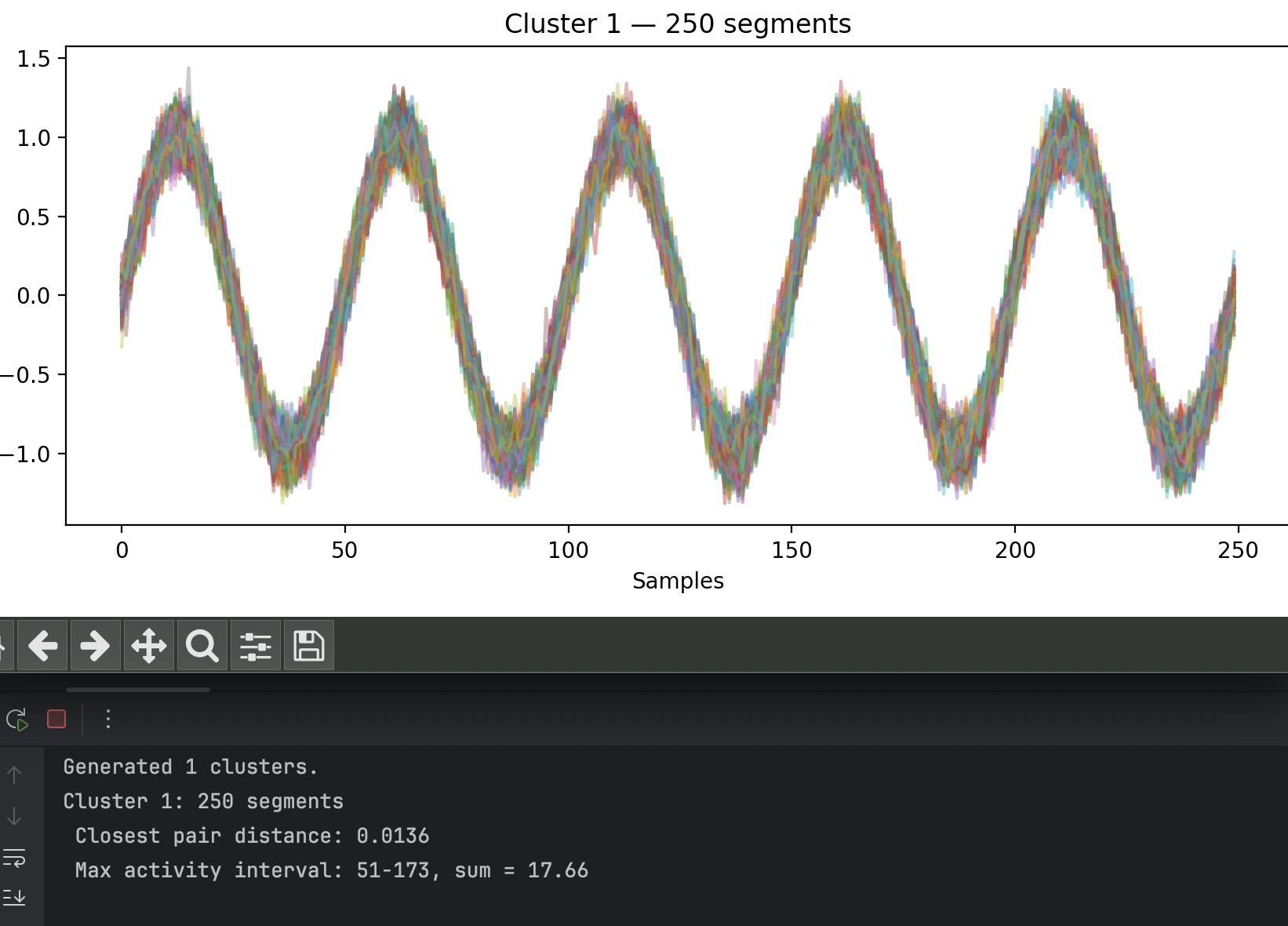
Kadane’s Algorithm here is used in our code to find the contiguous subarray with the largest possible sum in a time series. The function iterates through the signals, updating the max found so far and the max at the current position until the very end where we get the total max. It provides a local insight in each time-series segment and turns low-level algorithmic outputs into interpretable physiological evidence.

Next up is the visualization section which was created to more properly see what the segments look like as well as how they belong together. It displays/interprets the signal behavior and helps us decide whether the information we got from the previous steps are accurate or not as well as if the clustering matches together. Since this was created using matplotlib, it pops open a window for the visual that you need to exit out of in order to stop running the program.

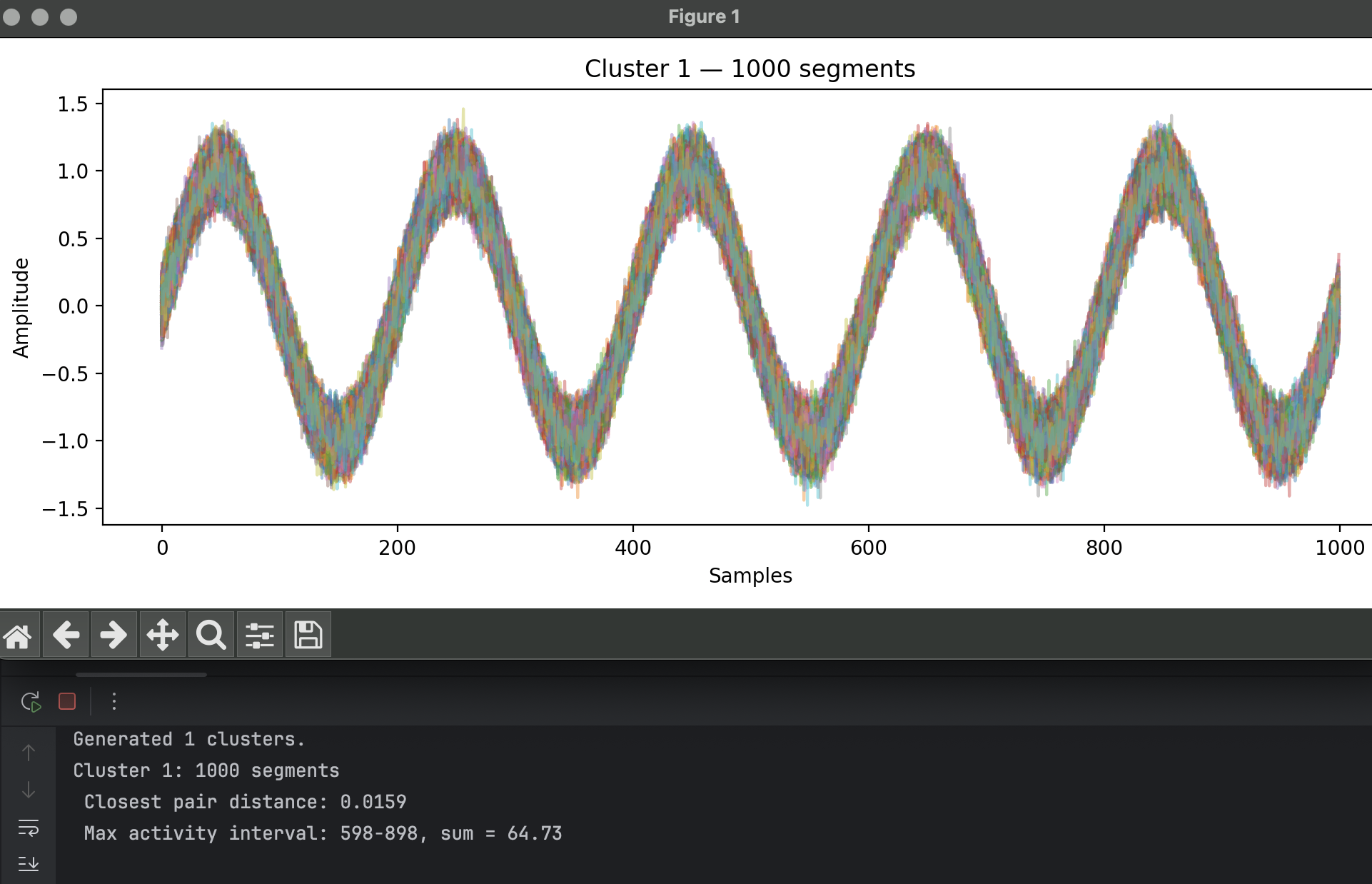
Finally is the main pipeline of the code where everything comes together and helps print out the closest pair length, max activity interval, and the sum for the cluster.

**EXECUTION RESULTS AND DISCUSSION**

**Toy Execution (250 segments):**



**Proper Execution (1000 segments):**



**Discussion:**

From the visual of our results, we are able to pick up on how the segments have a similar baseline aptitude because most of the segments line up together/have a similar shape to one another. We also notice that since the closest pair distance is less than 0.05, it is a highly cohesive cluster which indicates that it no longer needs to be split or filtered. Our results also show that this is regular cardiac activity.

**CONCLUSION**

As this is our first project in this course, I would have to say that it was by far the most time consuming assignment. Had I known just how long it would take, I would’ve done my best on completing and turning it in sooner. This especially includes the data I downloaded from Kaggle as it was a large file and took me around 30 minutes to fully download. My main takeaways from this project would include how divide and conquer work together very well with the closest pair algorithm as well as how I still need to learn more about Kadane’s Algorithm due to having more trouble on completing it compared to the other key components of the code. Overall, this project gave me the chance to expand my knowledge on clusters of time-series which will hopefully prove to be useful in the future as well as test my current skills of what I have learned so far in the course.