Tori Grosskopf and Nathan Fritz
12/11/2023

Part 1: Data Collection and Raw Plots

Data was collected very carefully to ensure that the data would be high quality and uncomplicated. To do this, the ECG sensor and Arduino were set up to be stationary. The tasks for each subset were selected to be as stationary as possible while producing the desired effect on the data. All data was collected with the patient (male 20) shirtless and with all wires and Arduino components as still as possible to reduce the amount of artifacts. Noise and artifacts within the ECG data can come from multiple courses, for example movement in the wires during data collection will cause peaks that could be seen as a heartbeat, or voltage that comes from the computer outlet that is powering the Arduino can show low amounts of noise.

5 minutes of sitting at rest: Data was collected while the user sat in a chair and meditated for 5 minutes straight. The patient practiced square breathing, 5 second inhale, 5 second hold, 5 second exhale, 5 second hold and repeated. This was chosen to keep the user in the most rested state possible.

5 minutes of Relaxing Activity: Data was collected while the patient stood and slowly moved around the floor on two feet while watching a history video on YouTube. This was chosen because it is a relaxing activity to do but is not done completely at rest. Quality data was collected by taking care to prevent the wires and Arduino system from moving around.

5 minutes of Mentally Stressful Activity: Data was collected while the user sat and played Fortnite Battle Royale, which is a highly stressful video game. The patient hot dropped Tilted Towers, known as the most stressful part of the game. Quality data was collected by keeping the wires and Arduino system as still as possible.

5 minutes of Physically Stressful Activity: Data was collected while the patient squatted continuously with body weight for 5 minutes straight to cause physical stress and increase heart rate. To ensure the best data collection, the Arduino was kept stationary, and the same body motion was attempted to be replicated with each squat to reduce the type and frequency of noise due to wire motion.

Two functions were written in order to plot the raw data. The first function, load_graph_data(file_name, subplot_value, title), reads in a data file and creates a subplot so that it can be used for plotting multiple data sets on the same figure. The function also converts Arduino units to voltage units. This function can be used for any set of data recorded by an Arduino with electrodes. The second function, concatenated_graph_data(data_tuple, subplot_value), concatenates the raw data and plots the result. It can be used for any amount of ECG data sets provided that they are placed into a tuple.

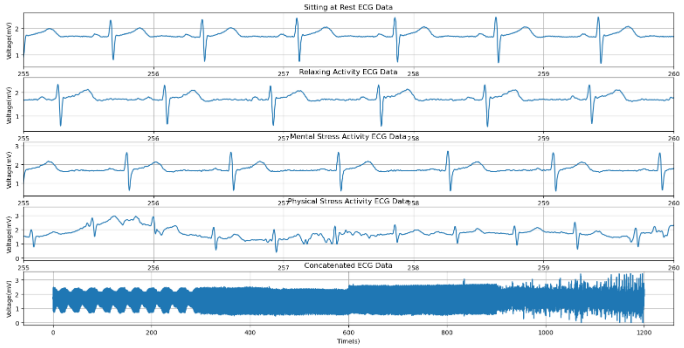

Fig. 1: Plot of raw collected data for each activity and the concatenation of that data

As can be seen in fig. 1, high quality data was achieved. In subplots 1, 2, and 3, clear ECG data and heart beats can be seen with little to no noise. In subplot 4 there was the most movement as the patient was physically active which caused noise.

Part 2: Filtering Data

For the collected data, a Butterworth filter was chosen (functions from the SciPy library). The choice of a Butterworth filter was made after researching methods that are commonly applied for ECG data filtration and concluding from multiple sources that it was recommended and how to use it[1].

The user expectation for the filter was that it would likely eliminate some of the noise, however it was not expected to function perfectly based on past user experiences with filter creation. This assumption was correct, as several adjustments to the filter parameters had to be made until a satisfactory amount of noise had been removed.

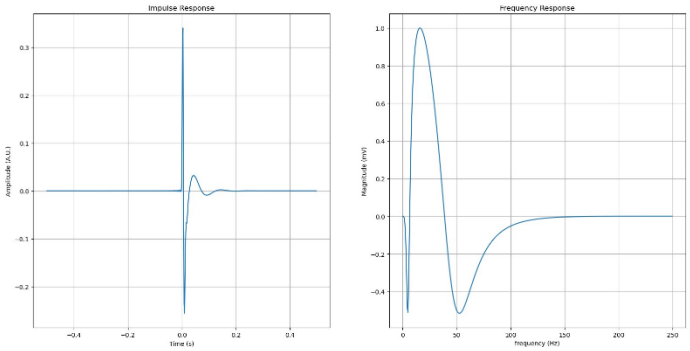

Fig 2: Impulse and frequency response of filter

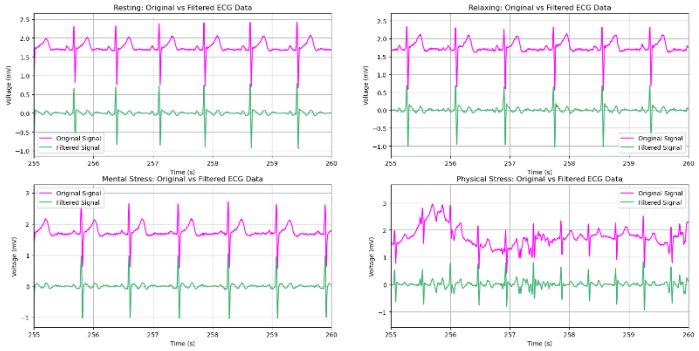

Fig 3: Comparative plots of raw data and filtered data

Several functions were created for this part of the project. The first one, create_filter_data(data, low_cutoff, high_cutoff, fs, order), is a flexible function where any combination of raw data, cutoff frequencies, sampling frequency and filter order can be inputted to produce a filtered signal. The filtered signal is predicted to be centralized around 0 on the y-axis, while decreasing overall noise, and maintingn the ECG signal for analysis. The plot in figure 3 was representative of user expectations. Producing the filtered data seen above was a trial-and-error process with different cutoff frequency and order values. The second function written, plot_frequency_response(numerator_polynomials, denominator_polynomials, fs, subplot_value), was written to plot the filter's frequency response. It takes inputs returned by the create_filter_data, as well as a sampling frequency and a subplot value for

displaying the frequency response next to the impulse response. This function will work with any input values provided that they are compatible with the freqz function from the SciPy library. The next function used in this part, plot_impulse_response(computed_frequency, frequency_response, subplot_value) relies on data returned from the plot_frequency_response function to plot the impulse response . It also takes an input for a subplot value so that the impulse response can be displayed in tandem with the frequency response. This function is dependent on outputs from plot_frequency_response. The final function written for part 2, plot_comparison(time, data, filtered_data, subplot_value, title), creates a plot of the various raw ECG data sets and their filtered counterparts. It can be used for any set of raw and filtered data. As can be seen in figure 3 the filter worked how we anticipated, decreasing noise, centralizing the data at 0 on the y-axis, while maintaining the ECG signal components. This allows further analysis of the ECG data for autonomic nervous system.

Part 3: Detect Heartbeats

To detect heartbeats within the data sets, the find_peaks function from the SciPy library was imported. This method was used as it is straightforward and minimizes the quantity of code that needs to be written. Using three optional inputs in find_peaks(), threshold(0), height(0.22), and distance(170), which were found by looking at data found in figure 3 as well as some trial and error, all peaks for even the noisiest data(physical stress) was found.

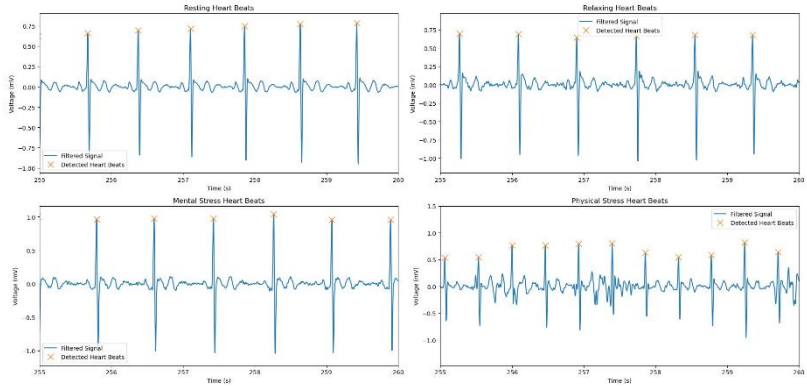

Fig 4: Filtered ECG data with markers identifying when heartbeats occurred

One function, plot_heart_beats(filtered_data, subplot_value, fs, title), was created to detect heartbeats and plot markers for when those beats occurred. The function is designed to produce a comparative plot for heartbeats across multiple data sets; data can be passed into it any number of times.

The function worked well in terms of detecting and returning the times at which heartbeats occurred. However, identifying the correct threshold for the data was a difficult trial-and-error process. Several times, the function would miss some of the heartbeats present within the data. While the code will work for any ECG data set, this could impact the accuracy of beat detection when different data is used. But it is believed that all heart beats were found accurately. The only data set that could possibly have some data is the physical stress heart beat subplot (4), which had the most noise so it is possible that there are miss identified heart beats.

Tori Grosskopf and Nathan Fritz
12/11/2023

Part 4: Determine HRV

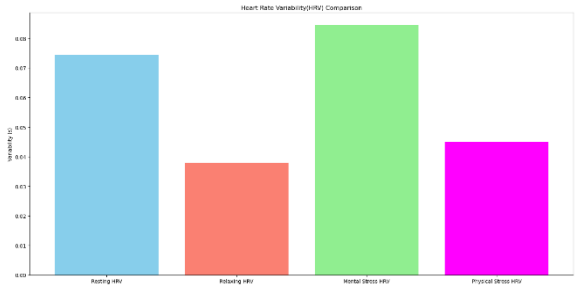

Fig 5: Bar graph displaying a comparison of heart rate variability for each data set

It was expected that the resting and relaxing activity data sets would have the highest heart rate variability, and the mentally and physically stressful activity data sets would have the lowest heart rate variability. The plots did not fully reflect this assumption. The mentally stressful activity data had the highest heart rate variability, followed by the resting activity, then the physically stressful activity, and finally the relaxing activity had the lowest heart rate variability. Although these are not perfect, the values of variability are low (<0.09) which means that heart beats were effectively found to then calculate the heart rate variability.

Two functions were created in this section, although technically one of them was done in preparation for plotting the power spectra (part 5). The first function, determine_hrv(filtered_data, fs, peaks), loops through the array of peaks for each data set and calculates the differences of time between when each heartbeat occurred, known as the inter-beat intervals (IBIs). The function then calculates the standard deviation of the IBIs, returning the heart-rate variability for each data set. This function will work for any set of ECG data provided the data has been filtered and the peaks have been determined. The second function, def determine_interpolated_timecourse(beat_intervals, time, peaks), uses linear interpolation with the array of beat intervals to return a new array of estimated beat intervals that are evenly spaced. This function will work with the beat intervals and peaks of any ECG data set, and the dt constant (representing samples per second) may be modified as needed.

Part 5

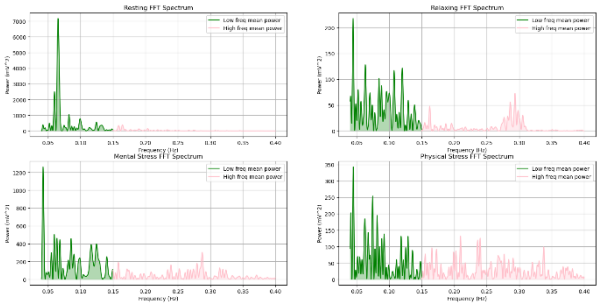

Fig 6: Displays the plotted frequency power spectra of each data set

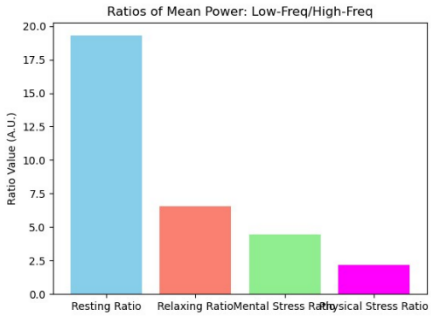

Fig 7: Bar graph displaying a comparison of the LF/HF ratios for each data set

The expectation for the data was that the lowest stress activity (resting) would have the highest ratio of low frequency power to high frequency, and the highest stress activity (exercise) would have the lowest ratio. This expectation was due to the assumption that the resting activity did not trigger the sympathetic nervous system (SNS), whereas the highest stress activity activated the body to move away from "rest and digest" (ANS) mode to "fight or flight" mode. The calculated LF/HF ratio values matched this hypothesis as can be seen in figure 7 with the resting ratio having the highest ratio value and physical stress having the lowest. This proves that resting and relaxing were using the parasympathetic nervous system(rest and digest) while mental and physical stress were using the sympathetic nervous system(fight or flight) of the autonomic nervous system.

One function, calculate_and_plot_mean_powers(interpolated_timecourse, dt, subplot_value, title), was written for this portion of data analysis. It takes the previously interpolated time course, and performs a Fourier transform in order to determine the high and low frequency bands and then plot those values as seen in figure 6. This function will work for any array that represents an interpolated time course and can produce multiple comparative subplots. The frequencies calculated in figure 6 were directly used to calculate the LF/HF frequency ratios that are plotted in figure 7, showing the accuracy of the mean powers, and functions.

Part 6

The data analysis supports the position that the LF/HF ratio reflects stress levels. This support is demonstrated in figure 7. The least stressful activity (resting meditation) has the highest LF/HF ratio, and the most stressful activity (continuous exercise) has the lowest LF/HF ratio. This data was not professionally recorded in research setting that limited user error. Artifacts were used within the final calculations, that could have thrown off the LF/HF calculations which are a direct indicator of what part of the autonomic nervous system is being used. However, at each step in the process, data was collected with the least number of artifacts possible, and beats were detected accurately showing overall effective data analysis. Additionally, the data was only recorded for a few minutes over a short period of time, which may not be a solid indicator of stress levels. Given more time and resources, it is likely that results would provide a clearer conclusion. The data collection could be spaced out – for example, only recording one activity per day and increasing the time for recording from five minutes to an hour. A more consistent Arudino set up could be created to mitigate movement of wires and noise. Additionally, with more time, multiple methods of data filtration could be tested with the intention of producing the best result, instead of focusing on just one method and adjusting parameters via trial-and-error.

Work Cited

[1] (PDF) filtering of ECG signal using Butterworth filter and its feature ..., https://www.researchgate.net/publication/267842421_Filtering_of_ECG_signal_using_Butterworth_Filter_and_its_feature_extraction (accessed Dec. 11, 2023).