SYSTEM AND METHODS TO DETERMINE ME/CFS & LONG COVID DISEASE SEVERITY USING WEARABLE SENSOR & SURVEY DATA

by

Yifei Sun

yifei.sun@utah.edu

A Senior Thesis Submitted to the Faculty of
The University of Utah
In Partial Fulfillment of the Requirements for the

Degree in Bachelor of Science
in
Computer Science

Approved:

DocuSigned by:
Shad Roundy 5/1/2023
2185C518D40241A...

Shad Roundy
Thesis Faculty Supervisor

DocuSigned by:
5/1/2023
BFF0CD1900814C1...

Tucker Hermans
Thesis Faculty Supervisor

DocuSigned by:
5/2/2023
AAE08049C9E14A0...

H. James de St. Germain
Director of Undergraduate Studies

DocuSigned by:
Mary Hall 5/8/2023
C9AFA8C98FCF4A6...

Mary Hall
Director, School of Computing

May 2023

Abstract

Myalgic encephalomyelitis/chronic fatigue syndrome (ME/CFS) is a debilitating disease with high probability of misdiagnosis and significant unmet medical needs that affects as many as 2.5 million people in the U.S. and causes enormous burden for patients, their caregivers, the healthcare system and society. Between 84 to 91 percent of ME/CFS patients are not yet diagnosed [6, 19], and at least one-quarter of ME/CFS patients are house- or bedbound at some point in their lives [12, 13]. The impact of ME/CFS to the U.S. economy, is about \$17 to \$24 billion in medical bills and lost income from lost household and labor force productivity per year [7, 13].

Current widely used diagnosis methods of ME/CFS and other diseases with similar clinical symptoms like Long COVID [6, 21] are highly dependent on patients' self reporting [4, 5] and standardized survey, which are not optimal for medical diagnosis. In a joint study with The Bateman Horne Center (BHC) [1], we designed and developed a system prototype that was able to stably collect terabytes of inertial measurement unit (IMU) time-series data, and analyzed multiple candidate parameters derived from them that could be used as reliable biomarkers for ME/CFS and other diseases with similar clinical symptoms.

Utilizing our system prototype, MetaProcessor, we conducted grouped t-tests on data collected from the EndoPAT study group (55 recruited, 51 participated, 30 ME/CFS, 15 Long COVID, 6 healthy control) to evaluate the predictive power of Upright Position Time (UpTime), Hours of Upright Activity (HUA), and Steps/Day. Through statistical analysis, we were able to assert the following for ME/CFS versus healthy control: 1. UpTime yielded a low p-value of 0.00004, indicating a significant difference between the groups and demonstrating its potential as a reliable measure for differentiating ME/CFS from healthy control populations. 2. HUA had a p-value of less than 0.00004, suggesting it could also serve as a useful measure for distinguishing ME/CFS from healthy control groups. 3. Steps/Day, x-axis and y-axis, had p-values of 0.01059 and 0.08665, respectively, indicating that step count may be relevant for differentiating ME/CFS individuals from healthy controls, but step count alone may not be sufficient to reliably distinguish between these groups. In a linear regression analysis, we found a moderately positive correlation between UpTime and HUA with $r^2 = 0.68$. Overall, we can confidently conclude that UpTime is a superior overall predictor due to its objective nature and the lowest p-values observed across all groups.

[1] BHC is a non-profit research clinic specialized in the diagnosis and treatment of ME/CFS, fibromyalgia, post-viral syndromes, and related comorbidities.

Table of Contents

List of Figures

List of Tables

1 Introduction

In this chapter, we will briefly discuss the motivation of our work, ME/CFS and Long COVID's symptoms and impact, definition of keywords, and discussion of prior studies.

1.1 Motivation

ME/CFS, a disease for which the cause has yet to be found, affects an estimated 836,000 to 2,500,000 people in the U.S. [7, 13]. About 90% of people with ME/CFS has not been diagnosed [7, 13]. As a long-term illness, it affects patients' daily activities in various ways causing inconveniences and dramatically lowering their quality of life. Due to ME/CFS's large patient population, affecting people working in various fields, it costs the U.S. economy about \$17 to \$24 billion in medical bills and lost income from lost household and labor force productivity per year [7, 13].

Biomarkers, is a contraction of "biological marker", refers to a wide range of medical signs that can be objectively measured and reliably reproduced from outside the patient. These medical signs are distinguished from medical symptoms, which are restricted to the indications of health or illness that patients perceive themselves [20]. A digital biomarker is a type of biomarker that is measured using digital tools, such as smartphones, wearable devices, and other digital sensors. Digital biomarkers can be obtained in a non-invasive and continuous manner, providing a wealth of data on an individual's health status and disease progression.

Long COVID, or Post-COVID Conditions, is a group of health conditions that might display symptoms on some people who have been infected with SARS-CoV-2 or its variants. Studies reported the symptomatology between ME/CFS and Long COVID shares many overlaps [6, 21], so we hypothesized the digital biomarker(s) used in ME/CFS study may show similar results in diagnosing and measuring the severity of Long COVID, and we decided to include Long COVID in our study.

Currently, diagnosis and treatment are challenging because there are no specific biomarkers and tests for ME/CFS. Widely used diagnosis methods of the disease (and similar symptoms in Long COVID patients [6, 21]) are not based on a statistically proven biomarker, but rather, they are based on patients' self-description of symptoms, questionnaires, and clinical observations [4]. Similarly, in a recent CDC report, their suggested diagnosis method is also largely based on patients' self reporting [5]. Thus, well-defined and reliable biomarkers are needed for a more objective diagnosis and to more accurately measure clinically relevant and meaningful outcomes of treatment for ME/CFS.

As above mentioned, ME/CFS has a large patient population, relatively high misdiagnosis rate and low successful diagnosis ratio. It's a non-trivial problem with huge economical impact of which we have limited understanding. With the COVID-19 pandemic still raging, there might be an increase in Long COVID case numbers as time passes. However, FDA has not approved any physical or pharmaceutical treatment for ME/CFS due to the lack of validated efficacy endpoints [4]. This lack prohibits drug manufactures showing evidence to FDA of the new treatment's effectiveness. Thus, in this study, to address the above mentioned issues, we will present a system prototype that provides a comprehensive data pipeline for collecting raw IMU data from clinical side to investigator side and analysis of multiple relevant digital biomarkers.

1.2 Background

Although the medical community still has a relatively limited understanding of ME/CFS, its has recently attracted more attention and research, leading to a consensus of the disease's core symptoms:

1. fatigue in response to physical exertion and post-exertional malaise (PEM),
2. unrefreshing sleep,
3. cognitive impairment, and
4. orthostatic intolerance (OI) [4].

The symptoms of individuals with ME/CFS vary greatly in both severity and type, with many manifestations beyond the core symptoms [17]. This variability presents a significant challenge in assessing the effectiveness of treatments, resulting in the disease's poor test-retest reliability [14], thus, high misdiagnosis rate.

In a pilot study, investigators found there might be connection between ME/CFS severity and orthostatic intolerance level [8, 15]. OI refers to onset of symptoms which occur when standing upright (or to be considered in a upright posture). These symptoms can be alleviated by reclining [15]. Symptoms include dizziness, headaches, weakness, and nausea if they stay in prolonged upright posture [15].

Prior studies at the BHC and clinical experience with over 1,000 ME/CFS patients have indicated that patients' disease severity and degree of physical impairment reflected by the level of OI can be gauged by Hours of Upright Activity (HUA, Figure 1.1) which is defined as time spent with feet on the floor over a 24-hour period. Researchers at BHC observed severely ill ME/CFS patients reported 0 to 4 hours with their feet on the floor while moderately ill patients reported having their feet on the floor for 5 to 8 hours. Patients with less than 4 HUA had significantly worse orthostatic intolerance symptoms ($p < 0.001$) and significantly greater interference with walking and standing ($p < 0.001$) compared to age and sex matched

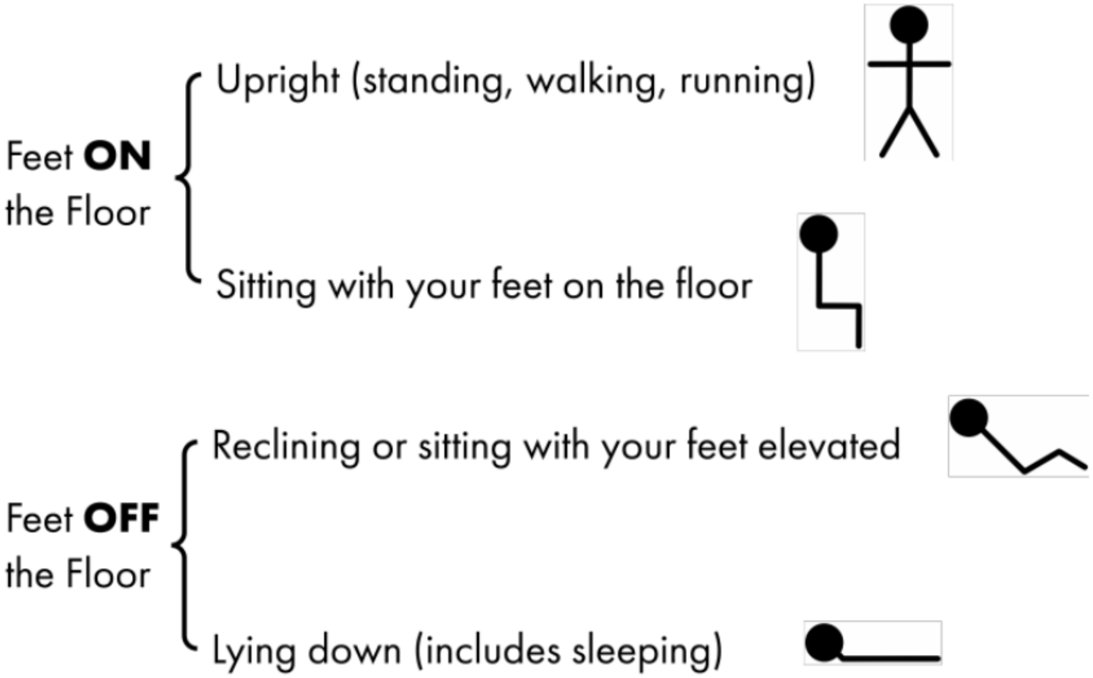
Feet ON the Floor
Upright (standing, walking, running)
Sitting with your feet on the floor
Feet OFF the Floor
Reclining or sitting with your feet elevated
Lying down (includes sleeping)

Figure 1.1: Definition of "upright activity": postures where the angle between study target's calf and an imaginary vertical line tangent to study target's kneecap is less than a certain critical point (Palombo [15], Figure 2.1).

healthy controls [9]. Although promising, HUA is based on pateients' self-report and, thus, may not be seen as reliable. These observations inspired us to develop an accurate and objective method to quantify impaired physical function by measuring upright activity, which we will refer to as Upright Position Time or UpTime (Figure 1.2) [15, 16].

UpTime is a good candidate digital biomarker in representing patients' orthostatic intolerance level with computation cost feasible for mainstream consumer level computers. The calculation involves several steps. First, raw inertial measurement unit (IMU) data is collected from the lower leg of subject participants. The following steps include pre-processing the raw IMU (merging raw output from sensors, unit conversions, etc.), filtering the data to remove high-frequency noise, and using a state-space Kalman filter to estimate the orientation of the leg in 3D space. Once the orientation estimate has been obtained, the program determines whether the leg is in an upright position or not. This is done by calculating the roll and pitch angles from the orientation estimate and checking whether the roll angle is within a critical range. The roll and pitch angles are calculated using quaternions, which is a way of representing rotations in 3D space. The quaternion representation of a rotation can be converted to Euler angles, which are commonly used to represent rotations in 3D graphics and robotics. If the roll angle is less than the critical angle, the leg is considered to be in

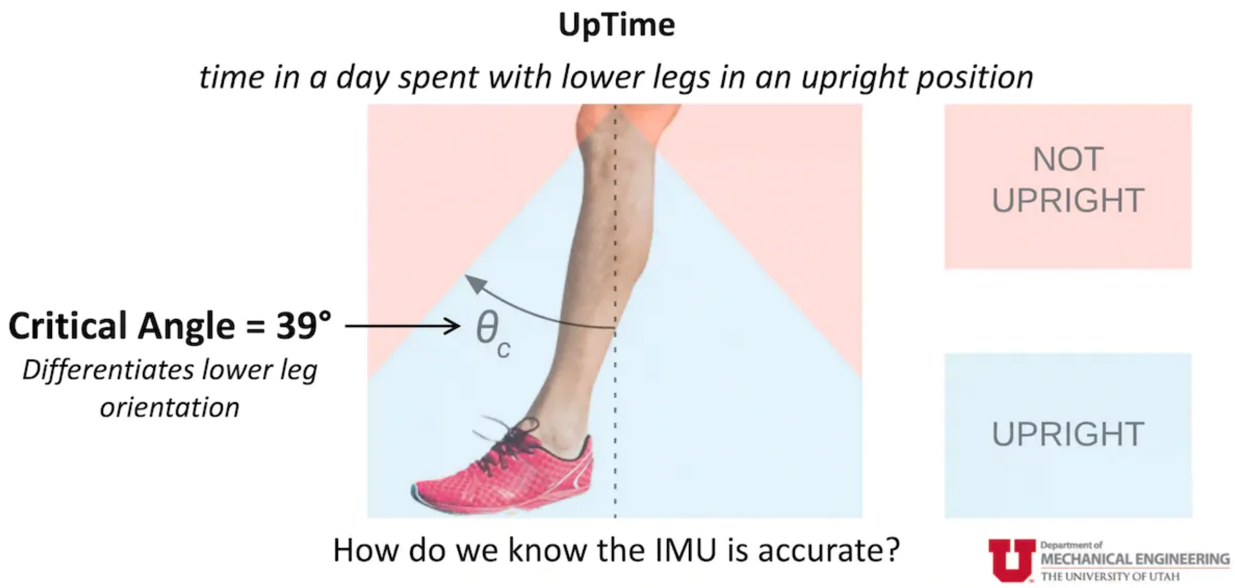
UpTime
time in a day spent with lower legs in an upright position
NOT UPRIGHT
Critical Angle = 39°
Differentiates lower leg orientation
θ_c
UPRIGHT
How do we know the IMU is accurate?
Department of MECHANICAL ENGINEERING THE UNIVERSITY OF UTAH

Figure 1.2: UpTime definition: The percentage of time a person's lower leg angle is less than a certain critical angle (39 degress in our study) over a pre-defined period of time (i.e., a day). Image reproduced with permission from Turner Palombo.

an upright position. Then, the upright percentage is then calculated by dividing the total time spent in an upright position by the total recording time [15]. [1]

To collect UpTime, we used external hardware to assist us. In a pilot study, researchers used Shimmer sensors to collect raw IMU data and then to extract UpTime percentage. In their study protocol, they required participants to wear a Shimmer IMU on lateral side of each lower leg, approximately two inches above the malleolus. In our study, we decided to use MbientLab's MetaMotionS (MMS), and put only 1 MMS on the outer side of patients' lower leg. (Analysis of the data from the pilot study indicated that there was no significant difference in UpTime values calculated from a single as opposed to both legs.) Figure 1.3 details the differences of our UpTime extraction workflow for the current study.

From the pilot study, investigators used Shimmer sensors for the collection of study participants' movement data, and they found the battery life of Shimmer sensors does not meet their expectation with significantly larger form factor and heavier weight comparing with MMS while only last around 3 days and MMS IMUs last around 7 days. Also, MbientLab provides comprehensive programming interfaces including low level C++ libraries to interact with the hardware and different bindings in other programming languages (Python, JavaScript, Swift) for all MetaWear series sensors, providing us more flexibility with stronger firmware customizability.

[1] A more comprehensive study about UpTime, including mathematical modeling and choice of critical angle, can be found in Turner Palombo's master's thesis.

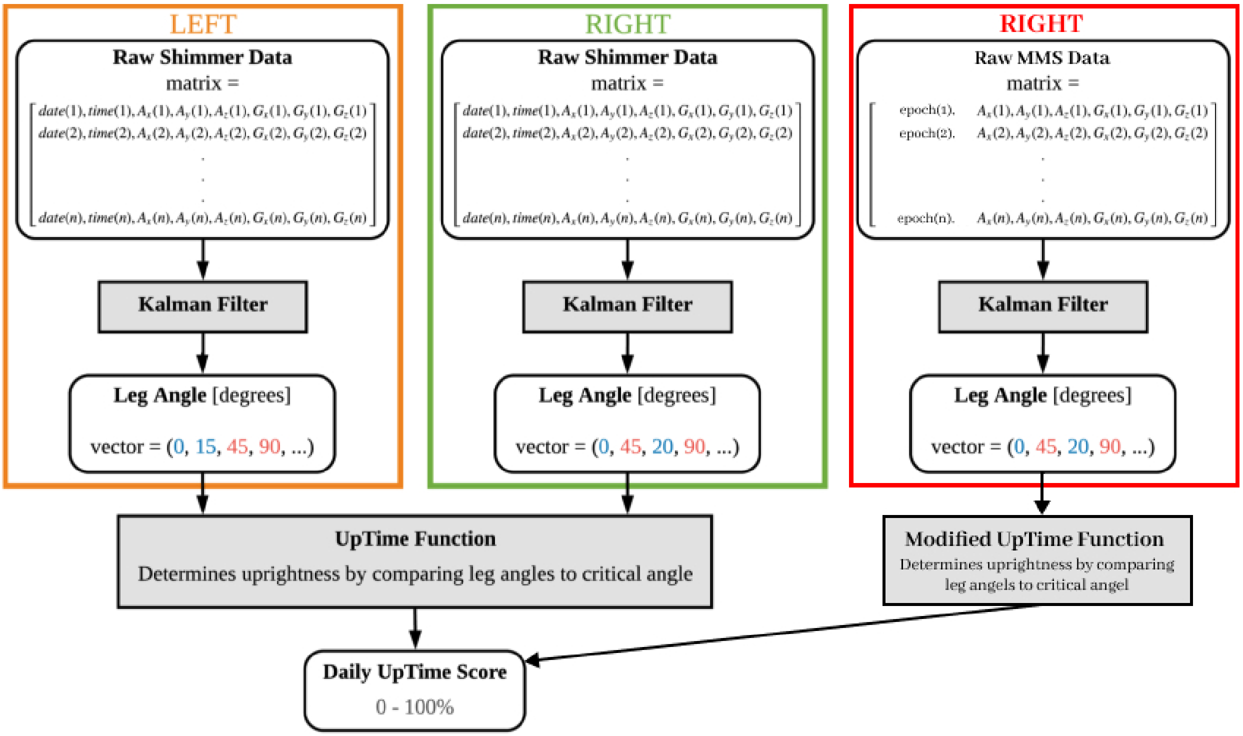
LEFT
Raw Shimmer Data
matrix =
$date(1), time(1), A_x(1), A_y(1), A_z(1), G_x(1), G_y(1), G_z(1)$
$date(2), time(2), A_x(2), A_y(2), A_z(2), G_x(2), G_y(2), G_z(2)$
$date(n), time(n), A_x(n), A_y(n), A_z(n), G_x(n), G_y(n), G_z(n)$
Kalman Filter
Leg Angle [degrees]
vector = (0, 15, 45, 90, ...)
RIGHT
Raw Shimmer Data
matrix =
$date(1), time(1), A_x(1), A_y(1), A_z(1), G_x(1), G_y(1), G_z(1)$
$date(2), time(2), A_x(2), A_y(2), A_z(2), G_x(2), G_y(2), G_z(2)$
$date(n), time(n), A_x(n), A_y(n), A_z(n), G_x(n), G_y(n), G_z(n)$
Kalman Filter
Leg Angle [degrees]
vector = (0, 45, 20, 90, ...)
RIGHT
Raw MMS Data
matrix =
epoch(1), $A_x(1), A_y(1), A_z(1), G_x(1), G_y(1), G_z(1)$
epoch(2), $A_x(2), A_y(2), A_z(2), G_x(2), G_y(2), G_z(2)$
epoch(n), $A_x(n), A_y(n), A_z(n), G_x(n), G_y(n), G_z(n)$
Kalman Filter
Leg Angle [degrees]
vector = (0, 45, 20, 90, ...)
UpTime Function
Determines uprightness by comparing leg angles to critical angle
Modified UpTime Function
Determines uprightness by comparing leg angels to critical angel
Daily UpTime Score
0 - 100%

Figure 1.3: Left: UpTime calculation workflow with Shimmer (Palombo [15], Figure 3.5, Appendix E). Right: Modified UpTime calculation workflow with MMS (Appendix A).

1.3 Study Design

The initial study undertaken by Turner Palombo showed a correlation between UpTime and ME/CFS disease severity and verified the UpTime algorithm accuracy [15]. However, this result is preliminary as the study that showed the correlation only had 15 subjects. Therefore, we collaborated with the BHC on a larger study with 51 participants which has a ME/CFS cohort, a Long COVID cohort, and a healthy control cohort. We collected 1 week worth of accelerometer and gyroscope data on each of the study participants. In addition standardized questionnaires/surveys were collected including RAND36, OISA, and OIDAS.

Medical professionals at the BHC are trained to use our custom system, MetaProcessor, which will be discussed in the following chapter, and they ensure the MMS device is placed correctly for each participant. In Figure 1.4, the “F” (forward) corresponds to the x-axis of the Bosch BMI160 accelerometer embedded in the MMS, and “V” (vertical) corresponds to the y-axis. If worn correctly, x-axis would point to forward walking direction, y-axis would point towards wearer’s knee, z-axis would point left towards wearer’s left ankle.

Each MMS device was placed in a soft, elastic band, then further secured on study participants’ right

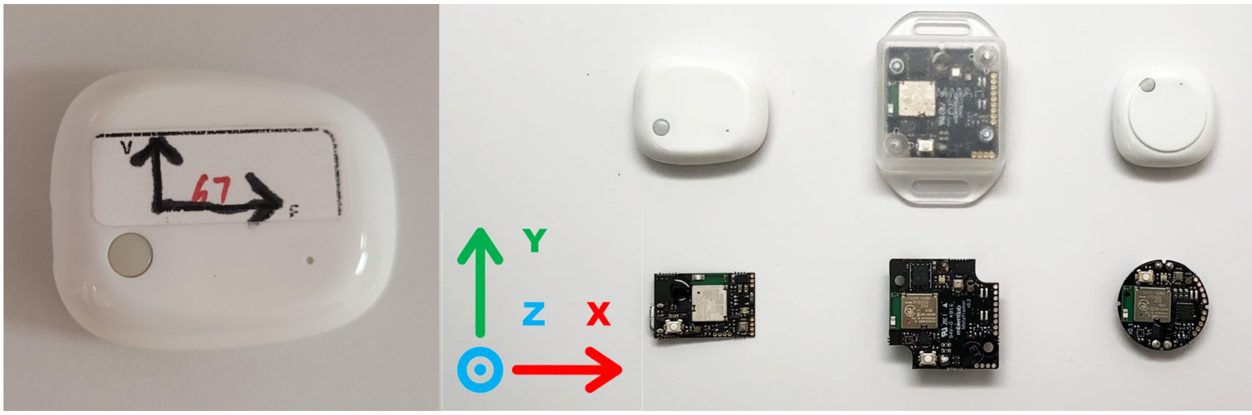

Figure 1.4: Left: one of our MMS in use, `f` means "forward", `v` means "vertical", during deployment, MMS devices are worn on study participants' lower right leg, on the outer side of ankle, with the label facing outward. Right: official Bosch BMI160 6-axis IMU orientation labeling from MbientLab, image source: `https://mbientlab.com/tutorials/Orientation.html`.

ankle, on the outside of their leg. Study participants were instructed not to take off the MMS device anytime unless they are showering or bathing. When showering or bathing, the participant needs to place the sensor in an orientation with the y-axis pointing upward if they were showering (so that the UpTime Algorithm would record "upright") or with the y-axis pointing horizontally if they were bathing (so that the UpTime Algorithm would record "not upright"). Before they leave the BHC, they are given a postage paid envelope to mail back the MMS at end of the data collection period (at least 7 continuous days).

In the following chapters of this thesis, we will detail:

1. our custom system, MetaProcessor, it's design, usage, and analytical data collected from the extended usage by The Bateman Horne Center,
2. multiple methods we used to analyze the data collected by MetaProcessor and results,
3. discussions of results,
4. finally, comments about the value of MetaProcessor and results of our study, and recommend directions for possible future work(s) in related fields.

2 System and Methods

In this chapter, we will cover our custom data collection system/tool, MetaProcessor, its design, choices made when designing the components with benchmarks, its usage, and analytical data collected from the extended usage by the BHC. After introducing MetaProcessor, we will further dig into the data collected using MetaProcessor, detailing the methodologies and possible contributions made to related fields.

2.1 System

The fundamental goal of this system is to reliably collect raw IMU data from MetaWear series sensors, securely transmit them to a intermediate location that's easily accessible for investigators, and computationally feasible for consumer electronics to extract relevant features (UpTime and other possible digital-biomarkers). We designed the whole system and MetaProcessor as a collection of tools that is understandable and usable by people without computer science background like medical professionals.

The system architecture overview is shown in Figure 2.1.

2.1.1 Hardware

The hardware used in our study (for data collection) includes: Raspberry Pi 3A units, Raspberry Pi 4B units, MbientLab MetaMotionS IMUs, and an Apple iPhone 7.

2.1.1.1 Requirement Analysis

When designing a data collection system, the first step is to analyze the requirements for the hardware components that will be used.

Raspberry Pi is a popular single-board computer that is widely used in the industry and the education sector. We selected the Raspberry Pi 3A and Raspberry Pi 4B models based on their processing power, memory, and cost-effectiveness. The Raspberry Pi 3A is a lower-end model, while the Raspberry Pi 4B is a more powerful model. The choice of which model to use depended on the specific requirements of the data collection task while both models are only meant to be used as an intermediate tool for downloading the data from the IMUs with some minimal sensor fusion tasks then upload the data to a cloud service provider.

The MbientLab MetaMotionS (MMS) is an IMU sensor that is specifically designed for wearable applications. It features a 6-axis sensor system that includes an accelerometer, gyroscope, and other sensors like magnetometer and ambient light sensors. The MMS is small, lightweight, and has a low power consumption, making it ideal for long-term wearables. We chose the MMS for our study due to its accuracy,

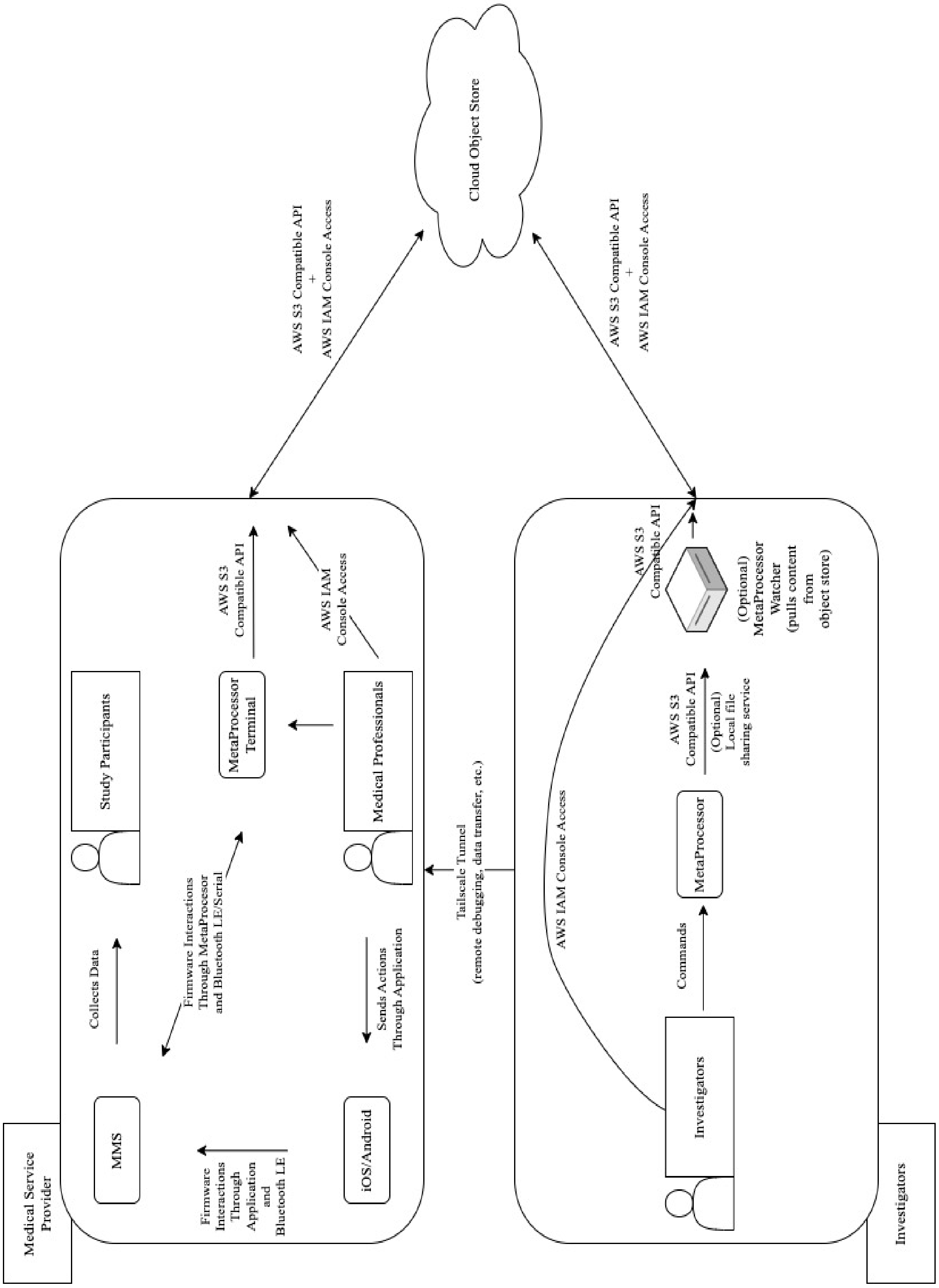

Figure 2.1: Overall system architecture. Study participants (SP) goes to research clinic, clinicians set up MMS using MetaBase, after SP mailed back MMS to clinic, clinicians download data from MMS, and the CLI compresses the data and send to object store, then investigators can perform analysis on collected data. Note: It is preferred to have the CLI running in a `tmux` session, so that investigators can use Tailscale Tunnel to remotely access `tty` session.

reliability, and ease of use. Also mentioned in Chapter 1, investigators were using Shimmer sensors before the start of our study. Their feedback was also taken into consideration: they found the battery life (3 days) of Shimmer sensors did not meet their expectation even with significantly larger form factor and heavier weight comparing with the MMS. In logging mode, the MMS can last approximately 7 days. Furthermore, MbientLab provides comprehensive programming interfaces including low level C++ libraries to interact with the hardware and different bindings in other programming languages (Python, JavaScript, Swift) for all MetaWear series sensors, providing us more flexibility with stronger firmware customizability.

The MMS device is configured with a smartphone. We chose to use an iPhone 7 in our study because we were planning to develop a phone application for MetaWear and the core developer was only familiar with app development using Swift. However, it's important to note that any other smartphone with Bluetooth Low Energy support could be used, if it is supported by MbientLab's MetaWear app [1]. Thus, the choice of using an iPhone 7 was primarily driven by our development team's familiarity with Swift and the availability of the required hardware.

2.1.1.2 System Configuration Analysis

When configuring the MMS sensors for our study, we needed to consider the sample rate, range settings, and other parameters that would affect the quality and accuracy of the data. Previous research indicates that human motion is confined to ultra-low frequencies, specifically those below 10 Hz [11], to prevent aliasing, our signal required a sample rate of at least twice this limit, or 20 Hz [1]. Nevertheless, gathering data at a higher sample rate than the minimum requirement would enhance filter performance [15]. In the preliminary study led by Turner Palombo, they were using the lowest sample rate supported by the Shimmer sensors (30Hz), and considering we will need sampling rate to be at least 20Hz, we decided to use the intermediate 25Hz. The Bosch BMI160 chip embedded in MMS, supports wide ranges of sampling rate and resolution, considering the nature of human and experience gained from previous studies, the accelerometer was set to an output range of ± 8 g, and the gyroscope was set to an output range of ± 1000 deg/sec.

To configure the MMS device for data collection, we utilized the MetaBase application, which is a powerful tool for configuring and managing various MbientLab devices. The MetaBase app is available for both iOS and Android devices, providing a wide range of compatibility options for users. To ensure consistency and accuracy in our data collection process, we chose to use an iPhone 7 for our configuration purposes. To begin the configuration process, we first instructed the Bateman Horne Center staff to reset the MMS device to its default settings. This was necessary to ensure that any previous configurations or data were cleared from the device before we began our data collection process. Once the device had been reset, we then instructed the

[1] GitHub: `https://github.com/mbientlab/metawear-swift-combine-sdk`.

staff to enable logging on the device, using the specific settings outlined in Table 2.1. These settings were carefully selected to ensure that we captured all relevant data points and minimized the potential for errors or discrepancies in our data. Finally, we instructed the staff to disable streaming on the device before it was given to study participants. This was done to prevent any potential interference or disruptions in the data collection process, as streaming can drain out the battery very quickly and an additional device is required to view the streaming log.

Features	**Settings**
Accelerometer	Enabled: 25Hz, ± 8 g
Ambient Light	Disabled
Gyroscope	Enabled: 25Hz, ± 1000 deg/sec
Magnetometer	Disabled
Pressure	Disabled
Temperature	Disabled

Table 2.1: MetaBase settings.

After the study participants had completed their data collection using the MMS device, they mailed the device back to BHC staff for further processing and analysis. To ensure that the collected data was accurate and complete, the staff utilized our custom tool called the MetaProcessor, which allowed them to seamlessly download and upload the data with attached metadata using only one command: `mp metawear download`, running on a Raspberry Pi 3 B+ we deployed at the BHC. The MetaProcessor is a versatile and user-friendly feature that is specifically designed to streamline the data processing and analysis process, making it easier for researchers and scientists to manage large amounts of data quickly and efficiently. With the MetaProcessor, the staff at the Bateman Horne Center were able to download the data collected by the MMS device directly onto their computer systems, where they could then analyze and process the data using a variety of different tools and software applications (detailed information in Section 2.1.2). In addition to the data itself, the MetaProcessor also allowed the staff to attach metadata to the collected data, providing important context and information about each data point. This metadata could include a wide range of different information, such as the date and time of the data collection, and any relevant notes or observations about the data itself. Overall, the use of the MetaProcessor was a critical component of the data collection and analysis process for The Bateman Horne Center staff, allowing them to efficiently manage and process large amounts of data with ease and accuracy.

During the early stages of development, specifically when the MMS firmware version was less than 1.70.0, we encountered a major hurdle in our data collection process. The issue was that serial download was not yet available, which meant that we had to solely rely on Bluetooth LE for data transfer. As a result, our data collection process was severely impacted, with an average download time of approximately 24.85 hours, based on a sample size of 20. This was due to the unpredictable nature of environmental radio disturbance, which caused download times to range from 17.53 hours to 40.18 hours per study participant. However, after we made the decision to switch to a updated firmware that had serial data transmission capability, we were able to significantly improve our data collection process. This was evidenced by the decrease in download time, which dropped down to 1.77 hours, based on a sample size of over 50. Furthermore, the range of download times also became much more manageable, ranging from 0.87 hours to 2.91 hours, with the exception of download sessions that encountered errors. As shown in Table 2.2, it's clear that this switch to the updated firmware with serial data transmission capability was a crucial step in optimizing our data collection process, and has allowed us to collect data in a more efficient and reliable manner.

Connection Type	**Average**	**Slowest**	**Fastest**
Bluetooth LE	24.85 hours ($n = 20$)	40.18 hours	17.51 hours
Serial	1.77 hours ($n > 50$)	2.91 hours	0.87 hours

Table 2.2: MMS data download speed comparison on different firmwares, switching from Bluetooth LE only firmware to serial enabled firmware resulted in 14.04x download speed increase.

During the initial stages of the development process, our team encountered several issues with the Raspberry Pis that we had deployed. These issues were identified through the monitoring platform Sentry.io, as well as reports from BHC staff members. To ensure that we could address these issues without the need for unnecessary travel and to facilitate faster project iteration, we leveraged Tailscale - a point-to-point VPN solution that comes with robust encryption and is supported by the open source WireGuard protocol. By using Tailscale Access Control, we were able to establish secure shell connections to the deployed Raspberry Pis, complete with two-factor authentication. This allowed us to perform necessary maintenance tasks remotely, while ensuring that data security was maintained at all times. A example of a possible usage of Tailscale Access Control List is presented in Figure 2.2.

2.1.1.3 Error Rate Analysis

Ensuring the stability of a data collection system is crucial for the accuracy and reliability of collected data (results are in Table 2.3). In our study, we evaluated the error rates of using the legacy method versus our custom data collection tool, MetaProcessor. The legacy method involves a human operator who manually controls the MetaBase app to start the data download process to a mobile device and then transfers the data to a dedicated storage location. On the other hand, MetaProcessor automates the data collection process and securely transmits the raw IMU data from MbientLab MetaMotionS sensors to an intermediate location.

Our error rate analysis revealed that using MetaProcessor resulted in significantly higher reliability compared to the legacy method. Specifically, we were able to achieve 5x reliability by using MetaProcessor,

```
{
  "groups": {
    "group:investigator": ["yifei.sun@utah.edu"]
  },
  "tagOwners": {
    "tag:testing": ["group:investigator"],
    "tag:production": ["group:investigator"]
  },
  "acls": [
    { "action": "accept", "src": ["group:investigator"], "dst": ["*:*"] }
  ],
  "tests": [
    {
      "src": "group:investigator",
      "accept": ["tag:testing:22", "tag:production:22"]
    }
  ],
  "ssh": [
    {
      "action": "accept",
      "src": ["group:investigator"],
      "dst": ["autogroup:self", "tag:testing", "tag:production"],
      "users": ["autogroup:nonroot", "root"]
    },
    {
      "action": "check",
      "src": ["autogroup:members"],
      "dst": ["autogroup:self"],
      "users": ["autogroup:nonroot", "root"]
    }
  ]
}
```

Figure 2.2: Example Tailscale ACL that mandates 2-factor authentication for SSH connections.

with an overall error rate of only 0.58%, compared to a 10% error rate for the legacy method. We collected 172 sessions using MetaProcessor, covering over 1,200 days worth of motion data, while we collected 20 sessions using the legacy method.

Although there were some instances of human error in using MetaProcessor (resulting in a 1.16% error rate), the overall error rate is expected to be below 1% for longer-term usages. This is a significant improvement from the legacy method, where human error was not a concern but the system error rate was much higher. Based on this analysis, we can conclude that MetaProcessor is a more reliable and efficient data collection system than the legacy method.

Data Collection Method	**System Error**	**Human Error**	**Total**
Legacy ($n = 20$)	2 (10%)	0 (0%)	2 (10%)
MetaProcessor ($n = 172$)	1 (0.58%)	2 (1.16%)	3 (1.74%)

Table 2.3: Error rate comparison: legacy method (2022-02-15 to 2022-04-25) v.s. MetaProcessor (2022-04-11 to 2023-03-07). We were able to achieve 5x reliability by using MetaProcessor comparing with using legacy method, and we expect the overall error rate to be below 1% for longer term usages.

2.1.2 Software: MetaProcessor

MetaProcessor [2] is a collection of tools built on top of MbientLab's MetaWear C++ and Python library, it provides a command line interface (CLI) with features to:

1. easy to use interactive CLI,
2. connect to MetaWear sensors and perform actions: download, reset, and other supported hardware level interactions via serial or Bluetooth LE connections,
3. manipulate data downloaded from MetaWear sensors: format conversion, sensor fusion, and compression,
4. manipulate dataframes generated from sensor fusion: time zone conversion, time stamp alignment, data interpolation,
5. calculate UpTime and extract other features using `tsfresh` [3],
6. configurable regular expression based batch processing,
7. interact with AWS S3 compatable API (including AWS S3, Cloudflare R2, Backblaze B2 Storage, etc.): list, read, write, and delete.

Overall, MetaProcessor is an efficient and highly configurable tool that provides a reliable and easy-to-use platform for collecting and analyzing motion data. Its components work together seamlessly, allowing users to easily perform various data collection and processing tasks with ease.

2.1.2.1 Components

Our data collection system, MetaProcessor, is designed with a modular architecture, meaning that each functionality of the system is separated into individual components. This approach was chosen for several reasons. Modular design allows for easier maintenance and updates. By breaking down the system into smaller components, each component can be updated independently of the others. This not only reduces the workload for developers, but it also minimizes the risk of breaking other parts of the system during updates. Additionally, components can be easily replaced or swapped out for other components if needed. Further, modular design promotes modular and reusable code. Each component can be designed as an independent module that can be used in other parts of the system or even in other projects entirely. This not only saves time in development, but it also improves code quality and reduces the likelihood of bugs. Modular components also enable greater flexibility and scalability. Since each component is designed to be

[2] GitHub: `https://github.com/metaprocessor/metaprocessor`, Documentation: `https://metaprocessor.org`.

independent, it can be easily scaled to accommodate changes in requirements or increased data volumes. New components can also be added to the system as needed without affecting the existing components, making it easier to adapt to new use cases or technologies.

1. **Preprocessor:** MetaProcessor's preprocessor is a powerful tool that facilitates the collection and processing of data from MetaWear devices. The preprocessor is designed to operate in two stages, each of which performs a distinct set of operations. The first stage, which is executed under the command `mp metawear`, includes a range of functions that allow for the seamless download of data from MetaWear devices. This stage also incorporates simple sensor fusion techniques, such as combining accelerometer and gyroscope dataframes, and format conversions to ensure that the data is in a suitable format for further analysis. The second stage of the preprocessor comprises operations that are executed outside of the `mp metawear` environment. These operations are not intended to be run on low-power single-board computers like the Raspberry Pi. Instead, they are designed to perform more complex tasks, such as missing data interpolations and sensor data time alignment. These operations play a crucial role in ensuring that the data collected from the MetaWear devices is accurate and reliable, which is essential for any subsequent analysis or modeling. It is worth noting that the preprocessor's two-stage design allows for greater flexibility and customization in the data collection and processing pipeline. Users can choose to execute only the first stage if they require basic data collection and formatting, or they can opt to run both stages to perform more advanced data processing tasks. This flexibility is particularly useful for researchers and practitioners who work with large datasets and need to tailor their data collection and processing workflows to their specific requirements.

   In order to utilize the raw IMU data obtained from MetaWear sensors, it is imperative that the data undergoes a preliminary stage of processing. This entails the retrieval of the data from the sensors and subsequent preprocessing. All the requisite operations are carried out through the invocation of library calls to the underlying MetaWear C++ library [3], which is accessed via Python C-bindings. Amongst the various operations that are performed, the most intricate involves the downloading of data from MMS devices. To accomplish this task, we have adapted the official example and utilized an event-driven approach to download data asynchronously from the Bosch BMI160's accelerometer and gyroscope. It is important to note that the accelerometer and gyroscope are treated as separate entities in the Bosch BMI160 driver implementation [4]. As a result, it is not feasible to extract both accelerometer and gyroscope data with matching Unix epoch simultaneously. To address this issue, we propose a solution whereby the

[3] GitHub: `https://github.com/mbientlab/metawear-sdk-cpp`.

[4] Bosch BMI160 Datasheet: `https://www.bosch-sensortec.com/media/boschsensortec/downloads/datasheets/bst-bmi160-ds000.pdf`, Bosch BMI160 Driver: `https://github.com/boschsensortec/bmi160_driver`.

6-axis IMU data is downloaded asynchronously in separate threads, as illustrated in Figure 2.3.

```python
# Example based on https://github.com/mbientlab/metawear-sdk-python/blob/master/examples/log_download.py

from pathlib import Path
from typing import Self, Any

from mbientlab.warble import *
from mbientlab.metawear import *
from mbientlab.metawear.cbindings import *


class Handler:
    def __init__(self: Self, signal: Any, dst: str) -> None:
        self.id = libmetawear.mbl_mw_anonymous_datasignal_get_identifier(signal)

        match self.id:
            case b"acceleration":
                print("accelerometer logger detected")
                self.sensor = "accelerometer"
            case b"angular-velocity":
                print("gyroscope logger detected")
                self.sensor = "gyroscope"
            case _:
                raise NotImplementedError(f"data handler for {self.id.decode()} is not implemented")

        self.file = None
        self.filename = Path.cwd()/dst/f"{self.sensor}.csv"
        self.data_handler_fn = FnVoid_VoidP_DataP(lambda ctx, ptr: self.write(ctx, ptr))


    def write(self: Self, ctx: Any, ptr: Any) -> None:
        df = parse_value(ptr)

        if self.file is None:
            print(f"writing {self.filename}")
            self.file = open(self.filename, "w")
            # csv style header
            self.file.write("epoch," + ",".join([f[0] for f in df._fields_]) + "\n")

        # get raw IMU data for all logger axis
        df = [str(getattr(df, f[0])) for f in df._fields_]
        self.file.write(str(ptr.contents.epoch) + "," + ",".join(df) + "\n")

    def __del__(self: Self) -> None:
        if self.file is not None:
            self.file.close()
```

Figure 2.3: Example asynchronous data handler for accelerometer and gyroscope.

The effective utilization of raw IMU data from MetaWear sensors largely depends on proper preprocessing and formatting of the collected data. To this end, we consider various file formats available for storing and processing the sensor data. We conducted benchmarking of popular file formats for storing series data, namely CSV, Feather, and Parquet, to evaluate their performance characteristics. Our experiments revealed that Parquet format is best suited to our needs, as it allows for faster data retrieval and compression compared to other file formats, as shown in Table 2.4. The columnar storage format of Parquet also enables efficient data filtering and aggregation, which makes it particularly suitable for sensor data analysis tasks. It is worth noting that the performance benefits of using Parquet may be dependent on the specific hardware and software configurations, as well as the type of sensor data being processed.

`aarch64-darwin`, **Apple M2 Max, 64GB RAM**			
Format	**Read**	**Write**	**Size**
CSV	3.14s (100%))	27.30s (100%)	716MB (100%)
Feather	0.16s (5.09%)	0.27s (0.99%)	285MB (39.86%)
Parquet	0.17s (5.41%)	1.38s (5.05%)	170MB (23.70%)

`x86_64-linux`, **Intel i9 9900K, 32GB RAM**			
Format	**Read**	**Write**	**Size**
CSV	9.37s (100%))	83.51s (100%)	716MB (100%)
Feather	0.69s (7.36%)	0.53s (0.63%)	285MB (39.86%)
Parquet	0.52s (5.55%)	3.05s (3.65%)	170MB (23.70%)

`aarch64-linux`, **ARM Cortex-A72 , 2GB RAM**			
Format	**Read**	**Write**	**Size**
CSV	23.06s (100%)	172.25s (100%)	716MB (100%)
Feather	7.81s (33.87%)	8.24s (4.78%)	285MB (39.86%)
Parquet	6.69s (29.01%)	9.31s (5.40%)	170MB (23.70%)

Table 2.4: Format benchmark result comparison (10 run average): benchmarked with a 716MB, 13,398,219 lines comma seperated value file.

However, it is important to mention that using Feather and Parquet formats can be slower when running the conversion process on a Raspberry Pi. This is due to the fact that the download operation needs to write the raw sensor data to CSV first before conversion, and the RAM on Raspberry Pis is very limited. As a result, our default setup is to directly write to CSV, compress both accelerometer and gyroscope data and the accompanying metadata into a zip file, and then upload the file to a cloud object store service provider. This approach optimizes storage space and enables easy and convenient access to the data for subsequent analysis and modeling tasks. In summary, it is very important to select an appropriate file format for storing and processing sensor data, taking into account factors such as data retrieval speed, compression efficiency, and analysis capabilities.

Once the data is preprocessed and compressed in the first stage, it is crucial to ensure that the data is securely stored and easily accessible for further analysis (more details in the Object Store). To achieve this, MetaProcessor can use cloud object store service provider that features AWS S3 compliant API. This allows us to easily upload and store the compressed data files in the cloud, where they can be easily accessed and processed by researchers. It is important to note that the compressed files will remain in the cloud unless a Watcher process (more details in Watcher) is initiated to download the files to a local storage and subsequently delete them.

In the second stage of preprocessing, the downloaded compressed files can undergo further processing to tailor the data to the specific requirements of the analysis or modeling task. One important operation in this stage is the conversion of file formats, as discussed earlier. Additionally, sensor data pre-processing can be employed to combine data from accelerometer and gyroscope and then provide missing data interpolations. While pre-processing step can be performed in the first stage, it is not recommended if the first stage is executed on a low-power single-board computer like the Raspberry Pi. Finally, data splitting is another operation in the second stage, which allows researchers to split the collected data into multiple segments for parallel processing or model training.

2. **Object Store:** As mentioned above, we need a secure, intermediate place to store and backup data collected from MetaWear sensors. As a part of data donwloading process, we will automatically upload compressed raw IMU data to a S3 compliant API service provider and then delete local copy after checksum verification. This approach ensures that the data is safely stored, even in the event of device failures or data corruption.

   To interact with these cloud object store service providers, MetaProcessor uses the Boto3 library[5]. Boto3 is the official Amazon Web Services (AWS) SDK for Python, which allows Python developers to write software that make use of services like Amazon S3, Amazon EC2, and others. Boto3 provides a high-level interface that abstracts the underlying API calls, making it easier to interact with various storage services. It also offers automatic retries and exponential backoff, ensuring that the upload and download operations are robust and resilient to network issues and service outages.

   To facilitate the interaction between MetaProcessor and the cloud object store service providers, we have implemented an Object Store helper object. This object provides methods for uploading, downloading, and deleting files, as well as listing objects in a specified bucket. The class also includes support for checksum verification to ensure data integrity during file transfers. The implementation is designed to be extensible and can be easily adapted to support additional cloud object store service providers as needed by providing configuration options (`$XDG_CONFIG_HOME/metaprocessor/config.toml` or `$HOME/.config/metaprocessor/config.toml`) avaible to end users with respect to XDG Base Directory Specification.

   In addition to providing a secure and reliable storage solution, the use of cloud object store services also enables efficient data sharing among researchers and practitioners. This is particularly useful in collaborative projects, where multiple users need to access and process the same dataset. To facilitate this process, MetaProcessor provides a sub-command that allows users to easily download the compressed

[5] Boto3: `https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/index.html`

data files from the cloud object store to their local machines (the Watcher process can also provide similar feature). The CLI also supports the deletion of files from the cloud object store after they have been successfully downloaded.

In this study, we have chosen AWS S3 as our primary cloud object store service provider. While there are indeed cheaper alternatives available in the market, the cost of using AWS S3 for our specific use case remained relatively low. After months of extensive usage, with approximately 40GB of compressed sensor data stored on S3, our monthly bill never exceeded $2. Given the benefits of using AWS S3, such as its robust infrastructure and extensive ecosystem, we believe that the slightly higher cost compared to other providers is justifiable and provides an excellent balance of affordability and quality for our data storage needs.

3. **Watcher:** The Watcher is a daemon process implemented using Python's daemon module. It is a optional sub-command that spawns a process that is responsible for monitoring the cloud object store for new data files uploaded by MetaProcessor. It requires users to supply relevant API configuration parameters, such as scan duration, API endpoint, API keys, bucket name, and a regex for file name detection. Users can configure the Watcher to delete files from the object store upon successful download. Once the timer goes off (as configured by the scan duration), the Watcher sends calls to the API endpoint to list files and proceeds to download them incrementally.

   In our study, we utilized a shared network storage and configured the Watcher to run on a machine with write access to the shared storage. Additionally, we set up the Watcher to delete files from the object store after successful downloads. This approach ensures that the data is readily available for analysis and processing while also minimizing storage costs by automatically deleting files from the cloud object store once they have been successfully downloaded.

4. **Parameter Calculation:**

   MetaProcessor includes several built-in subcommands for parameter calculation. These subcommands allow users to extract meaningful insights and features from the sensor data, making it easier to analyze and understand the collected information. The subcommands can extract following:

   (a) **UpTime:** The UpTime algorithm, derived from Turner Palombo's thesis, calculates the amount of time a subject spends upright from the sensor data [15]. We adapted the original MATLAB code to Python to make it compatible with the MetaProcessor workflow. As introduced in the Introduction, UpTime is an efficiently calculated parameter that can represent patient's orthostatic intolerance level. The parameter can be useful for understanding the overall activity levels and patterns of the subjects, providing valuable context for the sensor data analysis.

(b) **Steps/Day:** Another algorithm, Steps/Day, computes the daily step count for the subjects based on the accelerometer data. This algorithm was initially developed by an undergraduate researcher, McKenzie Hoggan, who previously worked on this project. The algorithm uses the local variance method to identify steps in the accelerometer data. The local variance method works by computing the local variance of the accelerometer data within a sliding window and detecting peaks that surpass a threshold value. These peaks correspond to the potential steps in the data. The algorithm then calculates the steps in both the x and y directions, outputting two step counts (based on x-axis and y-axis) for each input accelerometer data. Similar to UpTime, the original MATLAB code was modified and implemented in Python. The Steps/Day parameter offers insights into the daily physical activity levels of the subjects, and is suspected to also correlate with the orthostatic intolerance level.

(c) **Additional Parameters using `tsfresh`:** `tsfresh` is a powerful Python library designed for automatic time series feature extraction. It combines algorithms from various domains, such as statistics, time-series analysis, signal processing, and nonlinear dynamics, with a robust feature selection algorithm. By integrating `tsfresh` into MetaProcessor, we provide users with three options for feature extraction: minimal feature settings, efficient feature settings, and comprehensive feature settings [6]. This automated feature engineering process saves time and effort, allowing researchers to focus on other tasks. However, it is important to note that `tsfresh` is limited to CPU-based computation, which can result in lengthy processing times for large datasets. Users should be aware of this limitation and plan their feature extraction tasks accordingly.

In summary, the parameter calculation subcommands in MetaProcessor offer a wide range of options for extracting valuable information from the collected sensor data. By utilizing these subcommands, researchers can better understand the data and draw meaningful conclusions, ultimately enhancing the quality and efficiency of their research projects.

2.1.2.2 Conclusions

In conclusion, MetaProcessor is a powerful, open-source tool designed to facilitate the collection, processing, and analysis of raw IMU data from MetaWear devices (source code can be found at `https://github.com/metaprocessor/metaprocessor`, documentation can be found at `https://metaprocessor.org/`). The project has demonstrated its effectiveness by stably collecting over 1 terabyte of raw IMU data securely, with lower error rate compared to previous methods. MetaProcessor's user-friendly interface and streamlined

[6] The feature extraction settings are provided by `tsfresh`: `https://tsfresh.readthedocs.io/en/latest/api/tsfresh.feature_extraction.html#module-tsfresh.feature_extraction.settings`.

workflow make it accessible to users with varying levels of technical expertise, ultimately saving time and effort for medical professionals at BHC and investigators working with IMU data. MetaProcessor offers features like preprocessor, parameter calculation tools, secure data storage using cloud object store services, and working as a system daemon. These features, combined with the flexibility and customization options available in the data collection and processing pipeline, make MetaProcessor an invaluable tool for researchers and practitioners working with large IMU datasets. By providing an efficient, reliable, and user-friendly solution for managing IMU data, MetaProcessor has the potential to enhance the quality and efficiency of research projects in various fields, such as healthcare, human computer interaction, etc.

2.2 Methods

This section provides an exploration of the clinical study that employs MetaProcessor, including the setup, statistical analysis, and the various other methods utilized to analyze the study.

2.2.1 Clinical Study

For this project, we collaborated with the Bateman Horne Center on one of their ongoing ME/CFS and Long COVID clinical trials. 55 subjects were recruited (51 participated) for this study from three different cohorts: ME/CFS (30), Long COVID (15), and healthy controls (6). Together we refer to the study subjects as the EndoPAT study group.

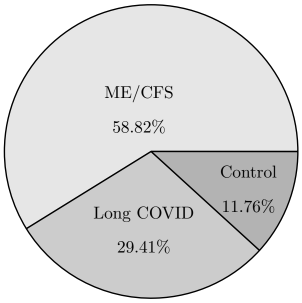

Figure 2.4: Dataset labels and percentages.

2.2.1.1 Study Protocol

Each subject in the study wore an MMS sensor on their outer side of right ankle for 1 week continuously. The only time the sensor was removed was during a bath or shower. The subjects were instructed to place the sensor in an orientation with the y-axis pointing upward if they were showering (so that the algorithm would record "upright") or with the y-axis pointing horizontally if they were bathing (so that the algorithm would record "not upright"). The sensors were configured as described above (refer to Hardware section), and data was processed utilizing a data collection station deployed at BHC (Figure 2.5).

The rationale for instructing participants to position the MMS device with the y-axis pointing upward during showering and horizontally during bathing is as follows: study participants are typically in an upright posture while showering, as it occurs during standing time, whereas bathing typically involves a reclined position. Consequently, the y-axis recordings of the MMS devices represent the lower leg's relative angle between the lower leg and an imaginary horizontal plane, thus accurately capturing posture differences

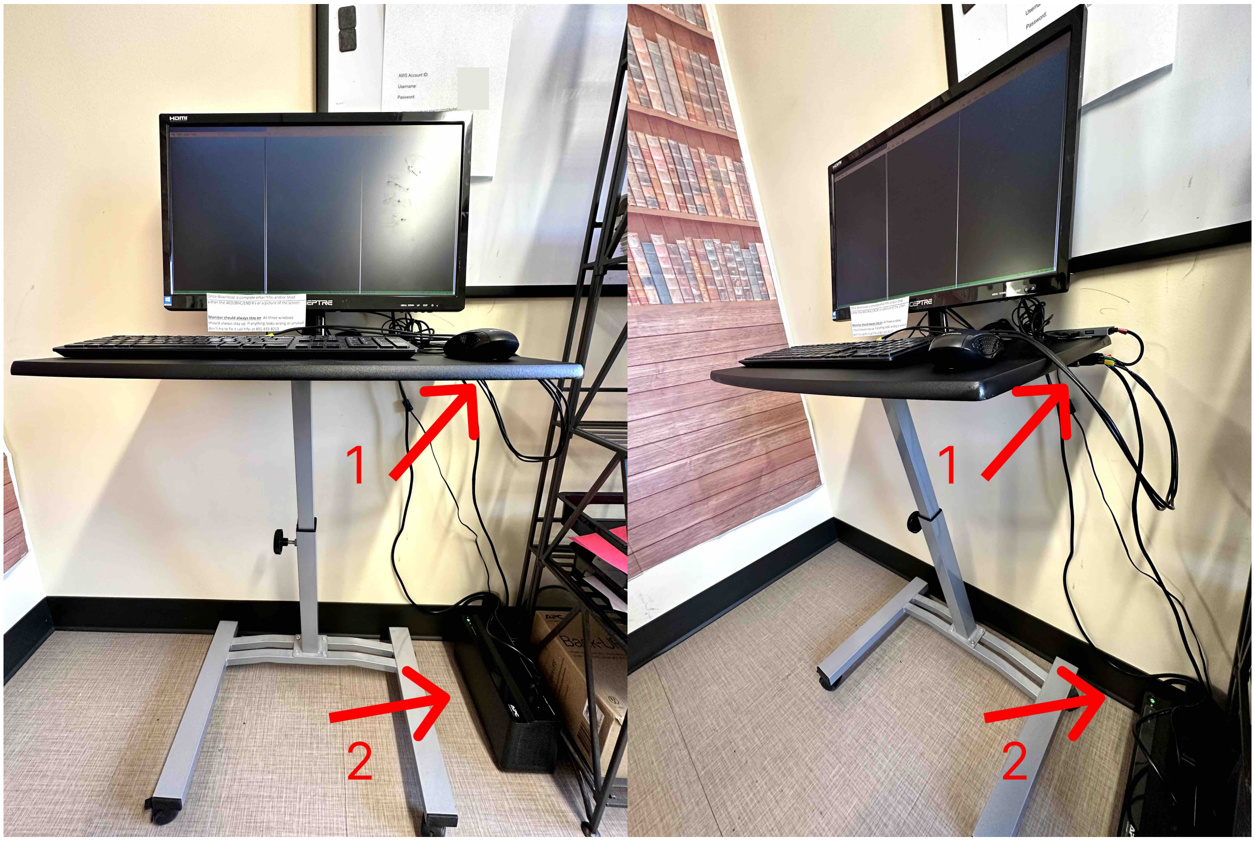

Figure 2.5: Data collection station deployed at BHC. 1. A Raspberry Pi 3 B+ unit taped under the desk. 2. A uninterruptible power supply. There are also three color coded USB Micro B cable plugged into the Raspberry Pi 3 B+ unit, providing charging and serial download functionality. Each of the three color coded USB cable corresponds to the `tmux` session name displayed on the monitor, clinicians can issue MetaProcessor commands into the corresponding `tmux` session, lowering error rate while enabling system administrators remote debugging and training capability.

between showering and bathing.

In addition, participants were asked to report the number of hours they spent upright each day.

At the end of the study, data from 3 subjects had to be discarded [7], leaving 48 subjects with valid data (Figure 2.4).

2.2.1.2 Dataset

Each data point collected by MetaProcessor was organized in the following structure (we used `.csv` in our study, but it can also be in `.parquet` or `.feature`):

- `{RedCap ID - BHC ID - Device ID}-Metadata.json`
- `{RedCap ID - BHC ID - Device ID}-Accelerometer.csv`

[7] Invalid entries: 1. Long COVID: END003 (setup error, accelerometer only), 2. ME/CFS: END006 (setup error, gyroscope only), 3. ME/CFS: END011 (bad firmware, the first downloading session with serial connection).

- {RedCap ID - BHC ID - Device ID}-Gyroscope.csv
- {RedCap ID - BHC ID - Device ID}-Preprocessed.csv
- Days/{RedCap ID - BHC ID - Device ID - Day Number}.csv

The files named *-Accelerometer.csv and *-Gyroscope.csv contain raw sensor data (direct output from MMS sensors), while the *-Metadata.json file contains session identifiers, start and end timestamps of the session, as well as any optional comment provided by clinicians or study participants. These metadata components supply the essential context required for subsequent analyses. The remaining two files/folders are generated through either manual or automatic means, contingent upon the configuration of the MetaProcessor, specifically its daemon process. These files represent the outcomes of the MetaProcessor's preliminary processing phase, which involves operations such as dataframe merging, timestamp alignment, and missing data interpolation, etc.

Each valid pre-processed entry (*-Preprocessed.csv) is a single comma separated value file with 7 columns (unix epoch, accelerometer x,y,z, gyroscope x,y,z) and about 12 million to 17 million rows (25Hz $\cdot$ 60s $\cdot$ 60min $\cdot$ 24h $\cdot$ 7days) in length.

Aside from the data collected by the MMS devices, each patient underwent a comprehensive evaluation during their initial visit, including vital signs measurements. They were also instructed to complete and submit various survey forms online, provided additional information about their health status. The vital measurements and survey data collected and are included in our dataset contains: DANA Brain Vital (DANA), Augmentation Index (AI), Augmentation Index Normalized to HR 75 bpm (AI75), Baseline Heart Rate (bpm) (BLHR), Natural Base Log of Reactive Hyperemia Index (LnRHI), Reactive Hyperemia Index (RHI), and Hours of Upright Activity (HUA).

The integration of MMS sensor data with these additional health metrics provided a comprehensive dataset for the study, allowing researchers to analyze various aspects of the participants' health and draw meaningful conclusions about the effectiveness of clinical trials. However, during the analysis, we did not conduct specific, in-depth evaluations of the collected survey data. This was due to above mentioned survey data was not the primary focus of our research, and we do not have valid hypothesis for collected survey data that could relate with the severity classification of ME/CFS.

2.2.1.3 Feature/Parameter Calculation

To further evaluate and analyze the health status of the study participants, the pre-processed data were utilized to calculate various parameters, including UpTime and Steps/Day. These calculations were initially performed using MATLAB, which were later converted to Python for integration with our existing data processing workflow (algorithm used for UpTime is included in Appendix A, and algorithm used for Steps/Day is included in Appendix B), MetaProcessor.

The primary objective of these calculations was to derive quantitative measures representing the subjects' physical activity levels that can infer the severity of orthostatic intolerance, then further reflect the severity of ME/CFS (Long COVID). Upon calculating the UpTime and Steps/Day results for each participant using pre-processed data, the output was stored in dataframes for further analysis. Additionally, we incorporated survey data provided by BHC, as well as clinical evaluation results obtained during each participant's visit to BHC. Integrating these data sources into a unified dataframe facilitated a comprehensive analysis of various health parameters in the context of the study.

The resulting dataframes, comprised of UpTime, Steps/Day, survey data, and clinical evaluation results, served as the foundation for subsequent statistical analyses. These analyses aimed to uncover patterns and relationships among the various parameters and their potential associations with the severity classification of ME/CFS or Long COVID. The outcomes of these statistical analyses are discussed in detail in the following sections and chapters of this thesis.

2.2.2 Statistical Analysis

Building upon the dataset compiled from the feature and parameter calculations, we used multiple statistical analysis methods to scrutinize the relationships among the various features/parameters. In the following sections, we will layout the analysis for three different cohorts, ME/CFS, Long COVID, and healthy control, for UpTime, HUA, and Steps/Day.

With our custom graphing tool, we can easily display the t-test results for the collected data. Although results for survey data do not form the central part of our analysis, they provide valuable supplementary information that can be useful for future research endeavors. In order to make this data accessible and to maintain transparency in our research process, we have included these graphs and t-test results in the Appendix C of this thesis. This additional information might prove beneficial for other researchers who are interested in exploring the relationships between various physiological and cognitive measures in the context of ME/CFS and Long COVID clinical trials.

2.2.2.1 UpTime

As an initial step in our analysis, we plotted the overall average UpTime across cohorts to observe visual patterns. In Figure 2.6, we noticed a substantially higher average on day 1 compared to the subsequent days. Consequently, we decided to focus our analysis on days 2 to 6, as these days include data from all participants while demonstrating a more consistent trend.

The variable n represents the number of participants who contributed IMU data for a specific day;

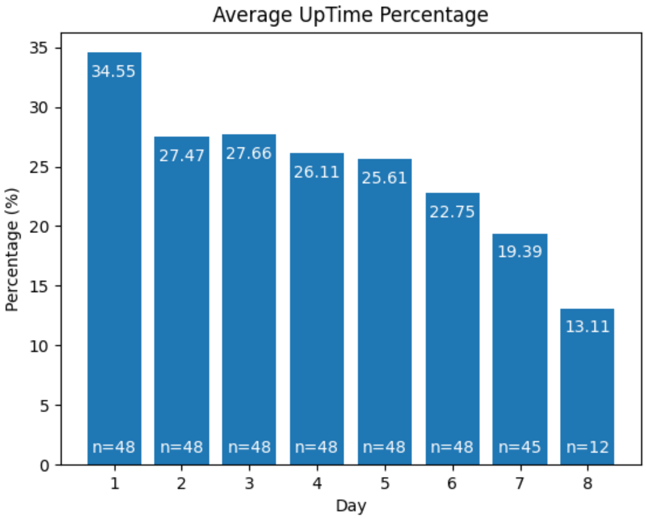

Figure 2.6: Average UpTime percentage per day accross cohorts.

however, it is crucial to note that this does not imply a full 24-hour data collection for each participant. For instance, on day 6, $n = 48$, while on day 7, $n = 45$. This discrepancy indicates that three participants concluded their participation in the study on day 6, and as a result, we do not possess complete 24-hour recordings for these individuals on that day.

As mentioned above, we observe a notably higher average on day 1 in comparison to subsequent days. We attribute this finding to the timing of the study, which typically commenced mid-day or in the afternoon. On day 1, participants arrived at the research clinic and were seated in an upright posture for an extended period, followed by traveling home, which also involved maintaining an upright posture (standing or sitting on public transport vehicle, or driving, or walking/jogging/running). Consequently, with an estimated recording time of fewer than 12 hours on day 1, a higher average UpTime is observed.

Given days 2 through 6 showed a more consistent trend, we used the UpTime data for each participant (originally spanning day 1 to day 6/7/8) and calculated the average from day 2 to day 6. This overall UpTime represents the participants' overall uprightness score throughout the study's duration. To further assess the statistical significance of the differences in UpTime values between the three cohorts, we performed grouped t-tests at 95% confidence level to differentiate the levels of severity of orthostatic intolerance as reflected by UpTime. By evaluating these results, we aimed to gain insights into the potential diagnostic value of UpTime as a digital-biomarker for orthostatic intolerance and related conditions, such as ME/CFS and Long COVID.

2.2.2.2 HUA

Figure 2.7 presents the daily average of HUA for all study participants across the different cohorts. During the initial analysis, we did not observe significant differences between the days compared to UpTime. However, to reduce unnecessary variability, we still chose to select days 2 to 6 and calculate the average HUA for each patient as an overall rating when performing grouped t-tests.

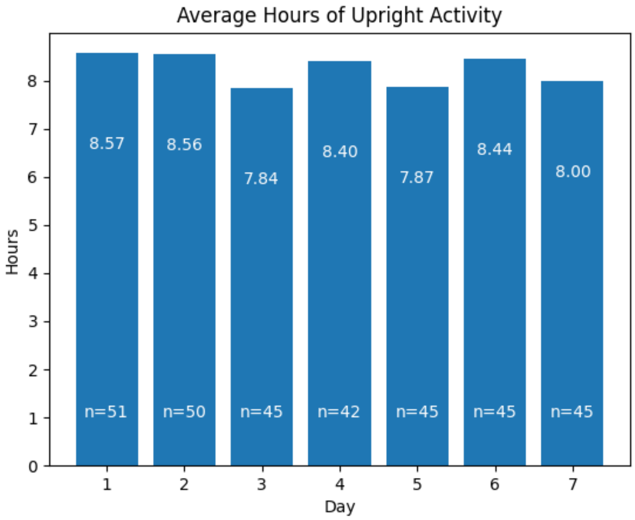

Figure 2.7: Average hours of upright activity per day across cohorts.

HUA as a self-reported measure, relies on patients logging into a secure portal and manually reporting their daily upright activity, this approach inherently introduces a degree of variability in the number of participants engaging in the survey throughout the study. Despite the consistent participation of all subjects on the first day, the number of participants fluctuates during the study's course. This variation may attributed to several factors, including the convenience of the data reporting process, patients' adherence to the protocol, or other external factors influencing individual engagement in the study.

Moreover, the daily average values derived from the HUA survey show a relatively smaller range of variation when compared with UpTime. The daily average HUA ranges from 7.84 hours to 8.57 hours, indicating a more stable pattern across the study period. This observation could be attributed to the subjective nature of self-reporting, which may not capture the subtle variations in patients' daily activities as effectively as objective sensor data.

2.2.2.3 Regression Analysis: HUA and UpTime

Given the similarities between HUA and UpTime as measures of orthostatic intolerance severity, conducting a linear regression analysis to explore the relationship between these two parameters is of great value.

Regression analysis allows us to directly compare HUA and UpTime values, enabling a more comprehensive understanding of their relationship and the implications of this relationship for patients with ME/CFS and Long COVID (results are in the upcoming chapter).

2.2.2.4 Steps/Day

We incorporated Steps/Day as a parameter in our study due to the ease of obtaining this data from readily available devices, such as smartphones and wearable technology. If Steps/Day is demonstrated to be a reliable digital-biomarker that can differentiate levels of OI severity and subsequently distinguish between ME/CFS cases, it could significantly simplify the integration of this feature into ongoing research efforts. This cost-effective approach would eliminate the need to invest in off-the-shelf IMU sensors, making it a more accessible and practical option for researchers and clinicians alike.

Moreover, there is reason to believe that Steps/Day may correlate with UpTime, as step count is intrinsically related to an individual's daily activity intensity, which could reflect their OI severity level. By exploring this potential relationship, we aim to assess the utility of Steps/Day as a proxy measure for OI and its value as a supplementary source of information in the context of ME/CFS and Long COVID research.

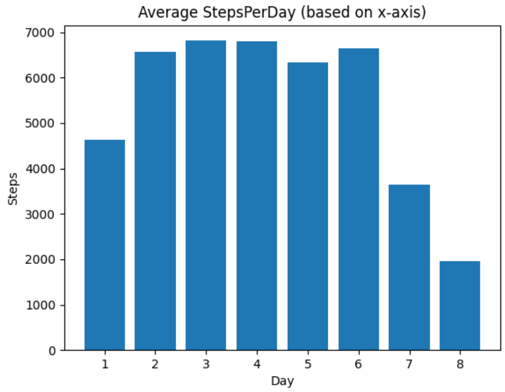

Figure 2.8: Average steps/day (calculated based on x-axis) across cohorts.

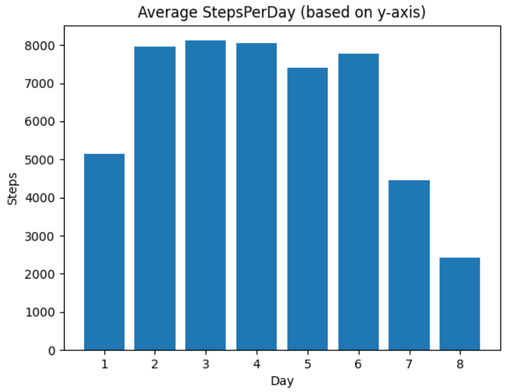

Figure 2.9: Average steps/day (calculated based on y-axis) across cohorts.

As shown in Figures 2.8 and 2.9, day 1 displays a markedly lower step count in comparison to other days, deviating from the observed trends for subsequent days. This observation supports the hypothesis that we previously discussed in the context of UpTime analysis, which posits that patients are likely to be sitting upright during their visits to the research clinic and traveling home in an upright posture. As a result, this leads to an inflated UpTime percentage on day 1 (refer to Figure 2.6).

In contrast, days 2 to 6 showcase a consistent straight line, which lends credibility to the accuracy and reliability of our data collection methods. Taking into account the rationale applied in the UpTime analysis, we chose to perform grouped t-tests on days 2 to 6 at 95% confidence level, as these days offer a more precise representation of patients' daily step counts and activity levels.

2.3 Conclusions

In this chapter, we presented a comprehensive overview of the custom data collection system prototype that was designed and developed to facilitate the collection, processing, and analysis of data from EndoPAT study, but extendable to any other study requires a data collection infrastructure that utilizes MbientLab's MetaMotion series IMU. We provided an in-depth examination of the design choices, benchmarks, usage, and analytical data obtained from the extended usage of MetaProcessor by the BHC. Furthermore, it set the stage for a more detailed investigation of the data collected using MetaProcessor and the potential contributions this information can make to the related fields.

In the System section, the chapter discussed the overall hardware choices and system configurations, including the IMU logging rate, resolution, and the significant increase in download speed achieved by using Bluetooth LE and Serial download (approximately 14.4 times faster). Additionally, the error rate analysis revealed that switching to MetaProcessor resulted in a 5-fold improvement in reliability, further emphasizing the value of out system prototype.

Moreover, in the Methods section, the chapter presented the recruitment information for the EndoPAT study: 55 subjects (30 ME/CFS, 15 Long COVID, and 6 healthy controls), with 51 ultimately participating in the research. The basic study protocol, data collection plan, processing plan, and the statistical analysis conducted on the pre-processed data were also thoroughly described, providing a solid foundation for interpreting the results and drawing meaningful conclusions from the study findings.

In the forthcoming chapter, Results and Discussions, we will delve deeper into the data collected using MetaProcessor, exploring the various relationships between variables and the potential implications of the findings for patients with ME/CFS, Long COVID, and healthy controls. Additionally, we will discuss the limitations of the current study, the potential avenues for future research (and exploration of machine learning methods utilized for initial exploration of our dataset), and the broader impact of the work presented in this thesis on the field of orthostatic intolerance and related conditions.

3 Results and Discussion

In this chapter, we will explain the results obtained from the data collected using MetaProcessor from EndoPAT study, examining the intricate relationships between variables and their potential implications for individuals with ME/CFS, Long COVID, and healthy controls. Furthermore, we will discuss the limitations of the present study, identifying potential avenues for future research, including the application of various machine learning methods as part of an exploratory pilot investigation. Lastly, we will offer some final thoughts and reflections on the broader impact and significance of this work in the field of ME/CFS, including similar medical conditions like Long COVID, connected by orthostatic intolerance and related clinically observed symptoms.

3.1 Results

The following section presents the key findings from the analysis of the data collected using MetaProcessor in the context of patients with ME/CFS, Long COVID, and healthy controls. We draw conclusions that contribute to our understanding of the severity of orthostatic intolerance, then further imply the details of the severity classification of ME/CFS and related medical conditions and its impact on patients with these conditions.

3.1.1 Integration of MetaProcessor

The integration of serial-download-capable firmware to MMS devices significantly improved our data download speed, achieving more than a 14-fold increase. The utilization of MetaProcessor yielded a considerable improvement in reliability compared to the legacy method. Specifically, we observed a 5x increase in reliability, with an overall error rate of only 0.58% (172 sessions) for MetaProcessor, compared to a 10% error rate (20 sessions) for the legacy method. These results demonstrate the effectiveness of MetaProcessor in enhancing data collection and processing efficiency.

3.1.2 Predictive Power of UpTime, HUA, and Steps/Day

In our pursuit to differentiate the levels of severity of orthostatic intolerance as reflected by UpTime, HUA, and Steps/Day, we used multiple t-tests to assess the statistical significance of the differences in UpTime values between three cohorts.

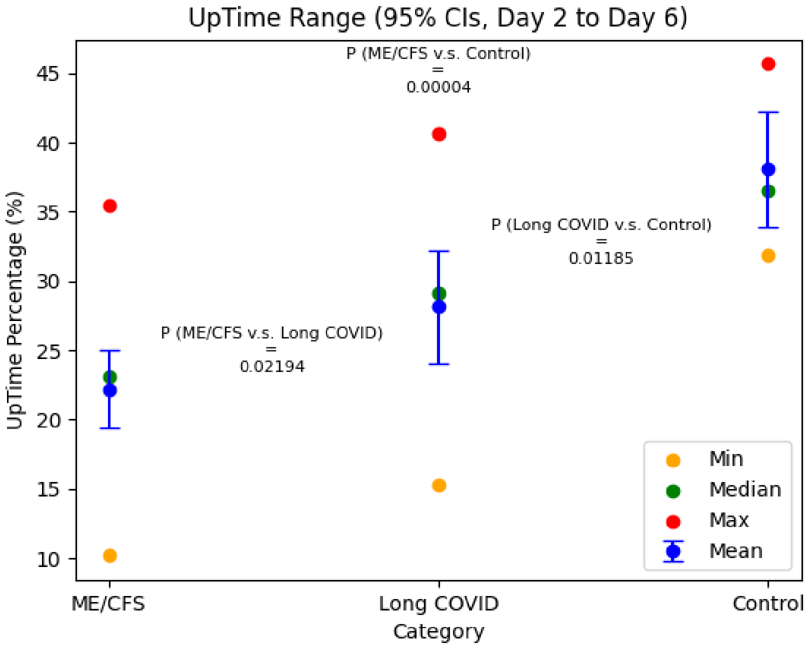

Figure 3.1: Grouped t-test performed at 95% confidence level for Upright Activity Time, ME/CFS vs. Long COVID vs. Control. In the legend, "Mean" is mean plus or minus a standard error.

3.1.2.1 UpTime

In Figure 3.1, at 95% confidence level, the t-test comparing the UpTime of the ME/CFS cohort and the control cohort yielded a highly significant p-value of 0.00004, suggesting a strong difference between the two groups. Similarly, the t-test comparing the UpTime of the Long COVID cohort and the control cohort produced a significant p-value of 0.01185, again highlighting a considerable difference between these groups. Lastly, when comparing the ME/CFS cohort to the Long COVID cohort, the t-test revealed a p-value of 0.02194, indicating a statistically significant distinction between these two cohorts as well.

In terms of the error bars representing the variability of the data, the ME/CFS cohort displayed a range from approximately 19% UpTime to about 25% UpTime, while the Long COVID cohort exhibited a range from about 25% UpTime to around 33% UpTime. The control cohort presented a wider range, extending from about 34% UpTime to roughly 43% UpTime.

3.1.2.2 HUA

Figure 3.2 presents the results of the t-tests performed at a 95% confidence level. The findings revealed statistically significant differences between each of the three cohorts. Specifically, the p-values obtained were as follows: p(ME/CFS vs Control) = 0.00000, p(Long COVID vs Control) = 0.00755, and p(ME/CFS vs Long COVID) = 0.04066. In terms of the HUA error bars, the ME/CFS cohort exhibited a range of

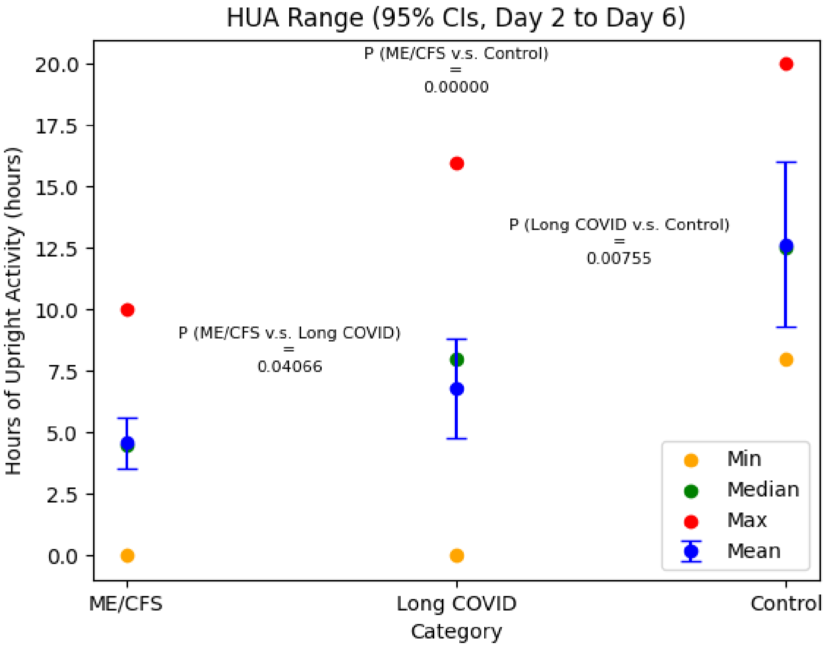

Figure 3.2: Grouped t-test performed at 95% confidence level for Hours of Upright Activity, ME/CFS vs. Long COVID vs. Control. In the legend, "Mean" is mean plus or minus a standard error.

approximately 3.75 to 6 HUA, the Long COVID cohort showed a range of about 4.75 to 9 HUA, and the healthy controls displayed a range of roughly 8.5 to 16.25 HUA.

3.1.2.3 Steps/Day

Despite the differences in the x and y axes, we expect that the data derived from both approaches will be consistent and should not interfere with our ability to draw meaningful conclusions. By comparing these data sources, we can further ensure the robustness of our findings and enhance our understanding of the relationship between step count and orthostatic intolerance severity in patients with ME/CFS and Long COVID. This comprehensive analysis will ultimately contribute to a more nuanced understanding of these conditions and inform the development of more effective, targeted treatment strategies.

In Figures 3.3 and 3.4, the t-test results reveal a statistically significant difference in step counts between the ME/CFS cohort and the healthy control group (x-axis: $p = 0.01059$, y-axis: $p = 0.08665$). These findings suggest that step count may indeed be relevant in differentiating between individuals with ME/CFS and healthy controls.

However, it is important to note that the error bars for the control group nearly completely overlap with those of both the Long COVID and ME/CFS groups. This overlap introduces an element of uncertainty in the distinction between these cohorts based solely on step count. Furthermore, although the p-value for

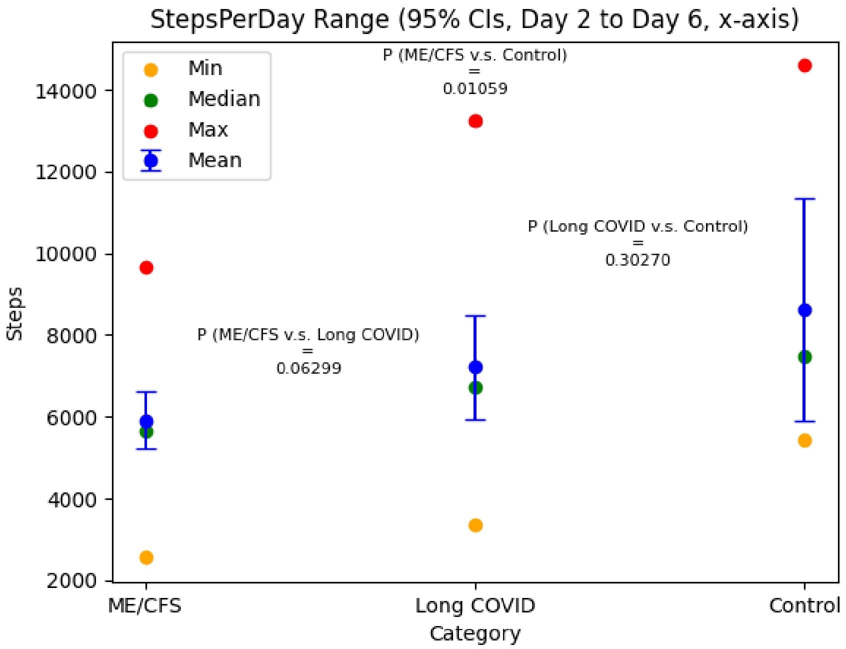

Figure 3.3: Grouped t-test performed at 95% confidence level for Steps/Day, x-axis, ME/CFS vs. Long COVID vs. Control. In the legend, "Mean" is mean plus or minus a standard error.

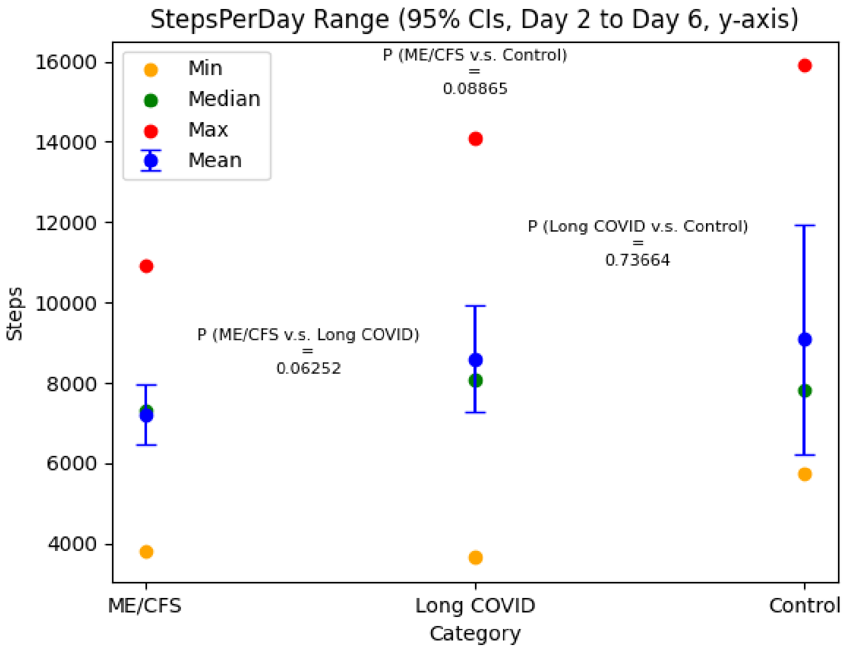

Figure 3.4: Grouped t-test performed at 95% confidence level for Steps/Day, y-axis, ME/CFS vs. Long COVID vs. Control. In the legend, "Mean" is mean plus or minus a standard error.

the comparison between the ME/CFS and Long COVID cohorts is relatively low, this does not necessarily imply that step count is a valid digital biomarker for distinguishing between these two groups.

3.1.2.4 Conclusions

In the comparison between ME/CFS and healthy control groups, we found the following:

- UpTime yielded a low p-value of 0.00004, indicating a significant difference between the groups, thus demonstrating its potential as a reliable measure for differentiating between ME/CFS and healthy control populations.

- HUA had a p-value of less than 0.00004, suggesting that it could also serve as a useful measure for distinguishing between ME/CFS and healthy control groups.

- Steps/Day, x-axis and y-axis, had p-values of 0.01059 and 0.08665, respectively, indicating that step count may be relevant for differentiating between individuals with ME/CFS and healthy controls. However, the overlap in error bars suggests that step count alone may not be sufficient to reliably distinguish between these groups.

In conclusion, our findings suggest that UpTime and HUA hold promise as potential digital biomarkers for differentiating between ME/CFS, Long COVID, and healthy control populations. However, the utility of step count as a sole measure to distinguish between these groups remains uncertain. Future research should explore the integration of multiple digital biomarkers and potential confounding factors to improve the accuracy and reliability of these measures in the context of orthostatic intolerance severity among patients with ME/CFS and Long COVID.

3.1.3 Relationship between UpTime and HUA

HUA and UpTime are both designed to quantify the duration of time patients spend in an upright position, yet they differ in their units of measurement. HUA is expressed in hours, while UpTime is represented as a percentage of total time. To enable a meaningful comparison and regression analysis, it is necessary to convert one of these measures to the same unit as the other. In this case, we can convert HUA to a percentage by dividing the HUA value by 24 hours.

Figures 3.5 and 3.6 demonstrate a positive relationship between HUA and UpTime, with a coefficient of determination (r^2) of 0.68. While this value is considered acceptable, it does not necessarily indicate a strong correlation between the two measures. This moderately positive relationship suggests that HUA and UpTime are related to some extent, reflecting consistency between self-reported experiences and objective measurements of orthostatic intolerance severity.

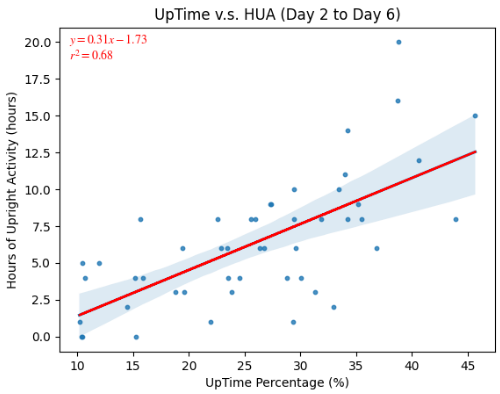

Figure 3.5: Linear regression analysis between UpTime and Hours of Upright Activity.

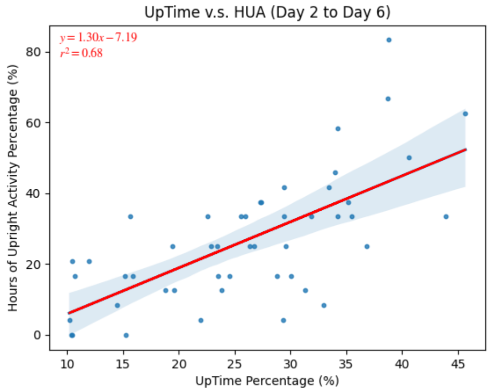

Figure 3.6: Linear regression analysis between UpTime and Hours of Upright Activity (HUA is converted into a percentage).

Interestingly, our analysis reveals that study participants are seemingly overestimating their HUA values in comparison to their corresponding UpTime data. For example, certain data points at 20% HUA were found to correspond to approximately 15% UpTime. However, at a closer look, there is a larger range of HUA values compared to UpTime, with the highest HUA reaching approximately 60% and the highest UpTime at about 45%. In contrast, the lowest HUA values range from around 2% to 8%, while the lowest UpTime is at 10%. Considering the slope of 1.3, this finding highlights the potential discrepancies between subjective reporting and objective measurements, emphasizing the importance of using both methods to gain a comprehensive understanding of patients' orthostatic intolerance.

In summary, the regression analysis reveals a moderately positive correlation between UpTime and HUA ($r^2 = 0.68$), indicating a consistent relationship between these measures. The trend of overestimating HUA values compared to corresponding UpTime data underscores the potential discrepancies between subjective reporting and objective measurements. These insights emphasize the importance of utilizing both methods to gain a comprehensive understanding of patients' orthostatic intolerance and inform treatment approaches for ME/CFS and Long COVID patients.

3.1.4 Long COVID Findings

Our analysis of the data related to Long COVID patients produced the following results:

- UpTime can be used to distinguish Control vs. Long COVID ($p < 0.019$) and ME/CFS vs. Long COVID ($p < 0.022$).

- HUA can be used to distinguish Control vs. Long COVID ($p < 0.0076$) and ME/CFS vs. Long COVID

($p < 0.041$).

Nonetheless, Steps/Day did not demonstrate sufficient statistical significance to be used as a predictor to differentiate between Control vs. Long COVID and ME/CFS vs. Long COVID. This finding further supports our recommendation to consider Steps/Day as a supplementary tool rather than a primary diagnostic indicator for these groups.

It is important to note that the root cause of Long COVID and ME/CFS is yet to be determined, and these two diseases might respond differently to the same treatment. Given the uncertainty surrounding the etiology of these conditions, we cannot make conclusive claims in the analysis of Long COVID data. Our findings should be interpreted with caution, and further research is needed to better understand the underlying mechanisms, severity classification and potential treatment strategies for both Long COVID and ME/CFS patients.

3.2 Discussions

In this section, we discuss the hardware considerations, parameter analysis, and implications of our findings, as well as the limitations and future directions of our study. Our aim is to provide a comprehensive understanding of the factors influencing our results, and to highlight potential areas for improvement and further research.

3.2.1 Hardware Considerations

While we have utilized a Raspberry Pi 3A for our data collection and processing, it is not necessary to use this specific single board computer. Any computer with the capability to run a Linux-based operating system and install the required MetaProcessor dependencies can be used. However, the capability of the CPU and size of RAM will affect the transcoding speed (i.e., converting from `csv` to `parquet` or `feature`). It is possible that the shell environment will terminate the process if MetaProcessor consumes too many system resources. Therefore, we recommend using a data collection station with at least 2GB of RAM and 16GB of storage to run MetaProcessor at clinicians' side. Although not required, we advise using `parquet` or `feature` for all use cases to accelerate compression time and data upload time to cloud object storage.

3.2.2 Parameter Analysis and Implications

Based on our findings, we can confidently assert the following:

- Patients with ME/CFS have significantly lower UpTime compared to healthy controls ($p < 0.00004$).

- Patients with ME/CFS have significantly lower HUA compared to healthy controls ($p < 0.00001$).

- Patients with ME/CFS have significantly lower step counts compared to healthy controls ($p < 0.1$).

Considering the results on the predictive power of UpTime, HUA, and Steps/Day, we can confidently conclude that UpTime is a superior overall predictor of orthostatic intolerance severity (and consequently, the severity of ME/CFS and Long COVID) compared to Steps/Day, with lower p-values across all groups. Although Steps/Day can be used as a statistically significant predictor for differentiating ME/CFS and healthy control groups (x-axis: $p = 0.01059$, y-axis: $p = 0.08665$), we recommend using it as a supplementary tool rather than a definitive diagnostic indicator without further analysis.

While HUA exhibited a slightly lower p-value, the difference of less than 10^{-4} is negligible. Moreover, UpTime, as an objective digital-biomarker, demonstrated a tighter error bar compared to HUA, which is a survey-derived feature.

3.2.3 Limitations and Future Directions

It is essential to consider that the root cause of Long COVID and ME/CFS is yet to be determined, and these two diseases might respond differently to the same treatment. Given the uncertainty surrounding the etiology of these conditions, we cannot make conclusive claims in the analysis of Long COVID data. Our findings should be interpreted with caution, and further research is needed to better understand the underlying mechanisms and potential treatment strategies for both Long COVID and ME/CFS patients.

Additionally, our study highlights the potential discrepancies between subjective reporting and objective measurements, emphasizing the importance of using both methods to gain a comprehensive understanding of patients' orthostatic intolerance. By examining these relationships and their implications, researchers can further refine their approaches to measuring and addressing orthostatic intolerance in patients with ME/CFS and Long COVID, ultimately contributing to improved treatment outcomes and patient experiences.

3.3 Initial Exploration and Future Work

In this thesis, we mainly discussed our approach of converting raw IMU data into basic parameters (HUA, UpTime, and Steps/Day) and apply statistical methods to them. Considering the size of the dataset, by no mean our analysis is complete. In this section, we will include the possible approaches of extending our analysis of basic parameters with more advanced machine learning methods in order get a better prediction of ME/CFS and Long COVID disease severity, during the trail, we considered multiple methods: feature extraction with `tsfresh` [3] (initial exploration), experimental ML approach with Transformers targeted to time-series classification [10] (future work), and Dynamic Movement Primitive (DMP) based motion analysis and recognition [18] (future work).

3.3.1 Feature Engineering with `tsfresh`

Feature extraction with `tsfresh` is a feature engineering method that uses time series data to extract meaningful features [3]. `tsfresh` is a Python package for automated time series feature extraction and selection based on the FeatuRe Extraction and Scalable Hypothesis testing (FRESH) [2] algorithm. This package is designed to extract and evaluate different time series features rapidly and automatically from time series datasets. It provides multiple series characterization methods that compute a large number of time series features. These features can be used to construct a design matrix that can be extended by additional univariate attributes and feature vectors from other types of time series. The design matrix can then be used for unsupervised machine learning, or supervised machine learning, in which statistically significant features can be selected with respect to the classification or regression problem at hand.

Even though `tsfresh` can extract a large number of features from the time series data automatically, we failed to utilize it to its full extent. `tsfresh` is benchmarked and showed good results with small datasets, meaning the row number of the testing datasets are relatively small (mostly less than 10 thousand rows), which may not be applicable to our EndoPAT study data as each of our data point averaged around 15 million rows, containing complex patterns and noises. Additionally, the feature extraction process with `tsfresh` requires a considerable amount of computational resources when running on larger datasets, which may not be feasible for our purpose. In our test runs, with the "basic feature" [1] settings (60 columns, 10 columns each for 3-axis accelerometer data and 3-axis gyroscope data), the calculation lasted around 1 minute per patient day, and for the "efficient feature" [2] settings (about 4700 columns) calculation lasted from 49 hours per patient day to 169 hours per patient day [3].

Using the "basic features" calculated with `tsfresh`, we classified the 60 output features into different groups: possible strong correlation, possible correlation, and theoretically not possible to have any correlation (Table 3.1. Our reasoning behind is due to the wearing orientation of MMS devices: forward corresponds to x-axis of IMU, vertical corresponds to y-axis of IMU, and left corresponds to z-axis of IMU.

The forward x-axis and left z-axis directions of the IMU, represented by A_x and A_z, are expected to demonstrate some correlation with disease severity when considering sum, medium, and RMS values. This is because these axes capture the horizontal and lateral movements during daily activities, which are likely to be affected in patients experiencing orthostatic intolerance, fatigue, and other symptoms common in ME/CFS and Long COVID.

On the other hand, the vertical direction, y-axis, represented by A_y, should not have a significant cor-

[1] Defined as `MinimalFCParameters` in `tsfresh`.

[2] Defined as `EfficientFCParameters` in `tsfresh`.

[3] The unit, "patient day" is defined as 24 hours worth of the study participants' accelerometer and gyroscope data, for example, if we collected 7 days of sensor data for 2 study participant, we'll have 14 "patient days" worth of data.

Feature	A_x	A_y	A_z	G_x	G_y	G_z
sum_values	R	N	R	N	R	N
median	R	N	R	N	R	N
mean	R	N	R	N	R	N
length	N	N	N	N	N	N
standard_deviation	R	N	R	N	R	N
variance	R	N	R	N	R	N
root_mean_square	S	N	S	R	S	R
maximum	N	N	N	N	N	N
absolute_maximum	N	N	N	N	N	N
minimum	N	N	N	N	N	N

Table 3.1: Analysis on tsfresh's MinimalFCParameters. $A_{x|y|z}$ and $G_{x|y|z}$ corresponds to accelerometer's and gyroscope's axis. "S" (in green) - might have strong correlation; "R" (in blue) - might have correlation; "N" (in black) - should not have any correlation.

relation with disease severity. This is because the y-axis mainly captures gravitational forces and is less informative in capturing the specific movements and postural changes that are relevant to these conditions.

G_y represents the angular velocity of the gyroscopic sensor, is expected to be a significant predictor of movement activity, particularly when considering RMS values. This is because the RMS values effectively capture the variations in movement intensity, which can be informative in assessing the overall activity levels of the patients.

Lastly, the variance for both acceleration and angular velocity should be predictive of activity levels. Higher variances in these measurements indicate a broader range of motion and more diverse movement patterns, which can be associated with higher activity levels in patients. Conversely, lower variances suggest reduced motion and a more limited range of activities, which can be indicative of lower activity levels commonly observed in patients with ME/CFS and Long COVID.

Feature	A_x	A_y	A_z	G_x	G_y	G_z
sum_values	0.041924	0.0	0.191563	0.000187	0.000523	0.0
median	0.015792	0.0	0.228202	0.647568	0.006879	0.000005
mean	0.041332	0.0	0.187355	0.00018	0.000533	0.0
length	0.000035	0.000035	0.000035	0.000035	0.000035	0.000035
standard_deviation	0.159109	0.000353	0.005491	0.027135	0.0	0.0
variance	0.110206	0.000389	0.004282	0.011215	0.0	0.0
root_mean_square	0.000826	0.0	0.000138	0.027227	0.0	0.0
maximum	0.836597	0.013301	0.690229	0.114135	0.07336	0.198528
absolute_maximum	0.020304	0.021652	0.412479	0.073017	0.202262	0.043988
minimum	0.002246	0.829734	0.412457	0.020094	0.333653	0.031144

Table 3.2: T-Tests for features gendered on tsfresh's MinimalFCParameters: Control vs. ME/CFS p-value table. "S" (in green) - might have strong correlation; "R" (in blue) - might have correlation; "N" (in black) - should not have any correlation.

Feature	A_x	A_y	A_z	G_x	G_y	G_z
sum_values	0.472635	0.00005	0.269436	0.099531	0.425181	0.000004
median	0.641273	0.000002	0.196691	0.957038	0.541774	0.001393
mean	0.46812	0.000073	0.269345	0.10009	0.428071	0.000005
length	0.000034	0.000034	0.000034	0.000034	0.000034	0.000034
standard_deviation	0.012679	0.097511	0.000537	0.024193	0.002971	0.003611
variance	0.020627	0.097818	0.000769	0.011676	0.002415	0.004402
root_mean_square	0.01226	0.000003	0.000904	0.024315	0.002977	0.003609
maximum	0.437505	0.028048	0.21497	0.802929	0.01724	0.89266
absolute_maximum	0.097325	0.037574	0.131244	0.828048	0.053146	0.697704
minimum	0.066436	0.934615	0.133253	0.775239	0.340476	0.363448

Table 3.3: T-Tests for features gendered on tsfresh's MinimalFCParameters: Control vs. Long COVID *p*-value table. "S" (in green) - might have strong correlation; "R" (in blue) - might have correlation; "N" (in black) - should not have any correlation.

Feature	A_x	A_y	A_z	G_x	G_y	G_z
sum_values	0.032805	0.010449	0.599469	0.073061	0.000482	0.365713
median	0.002744	0.00173	0.685771	0.642964	0.01127	0.921931
mean	0.03251	0.009921	0.589232	0.071567	0.000463	0.361612
length	0.589508	0.589508	0.589508	0.589508	0.589508	0.589508
standard_deviation	0.328658	0.000856	0.989138	0.756259	0.020668	0.003687
variance	0.494278	0.001406	0.723901	0.519591	0.012312	0.001817
root_mean_square	0.071836	0.000092	0.124049	0.756988	0.020627	0.003683
maximum	0.15072	0.910401	0.293532	0.011299	0.508641	0.155688
absolute_maximum	0.711243	0.776089	0.392233	0.004648	0.403723	0.046802
minimum	0.287907	0.688138	0.310116	0.004312	0.972233	0.171762

Table 3.4: T-Tests for features gendered on tsfresh's MinimalFCParameters: ME/CFS vs. Long COVID *p*-value table. "S" (in green) - might have strong correlation; "R" (in blue) - might have correlation; "N" (in black) - should not have any correlation.

Upon conducting grouped t-tests (Tables 3.2, 3.3, 3.4), we noticed promising results for most parameters marked as having possible correlation and possible strong correlation across all groups. However, further in-depth analysis is required to draw more concrete conclusions.

As tsfresh is not the primary focus of our research, we included all results of grouped t-tests in the tables and encourage interested readers to explore this area further for a more comprehensive understanding of the relationships between these features and the conditions under study. Future work in this area could potentially reveal additional insights into the utility of these features as potential biomarkers for disease severity and activity levels in patients with ME/CFS and Long COVID with overall lower computation cost.

3.3.2 DMP Based Pattern Recognition and Matching

DMP is a frequently used technique in robotics to generate trajectories: 1. Learn the input trajectory so that the motor system can reproduce the trajectory, 2. Adapt the trajectory to new goal(s). Trajectories in this context refer to the path that a robot's end effector, such as its arm or gripper, should follow over time to accomplish a task. The trajectory generated by a DMP is typically represented as a sequence of positions, velocities, and accelerations over time. In most cases, DMPs are generated in a demonstrative manner, for example, a human operator or another robot provides a reference trajectory for the robot to follow. Since the DMP models the trajectory as a non-linear dynamical system, it provides an adaptable and a flexible starting point which can be used in other dynamic systems for them to generate similar trajectories via reinforcement learning to achieve different goals or adapt to changes in the environment.

In *Dynamic Movement Primitives in Robotics: A Tutorial Survey*, the authors acknowledged the practicality of using DMP based approach to perform motion analysis and recognition [18]. DMP approach allows the user to generate movement trajectories by defining a set of attractor states and weights that modulate the movement of the system [4], such that this approach could be very effective in generating both periodic and non-periodic movements with varying complexities. These properties make it an attractive tool for motion analysis and recognition, as it can model and recognize a wide range of movement patterns.

However, ME/CFS is a complex and poorly understood medical condition that affects various systems in human bodies. Due to the diverse and variable nature of its symptoms, it is difficult to define a set of representative movement primitives that can be used to analyze and recognize the condition. Also, we are still unsure about whether ME/CFS or related medical syndromes will exhibit different symptoms on different individuals or not, this variability and inconsistency in human motor function/performance can make it challenging to develop accurate and reliable trajectory models. Furthermore, the lack of understanding of ME/CFS and the absence of objective diagnostic tests for the condition make it difficult to identify clear and consistent patterns of movement. Without better understanding of the underlying mechanisms of ME/CFS, it is challenging to develop accurate models of movement that can be used to analyze and recognize the condition with DMP.

In conclusion, despite the attempt, the use of the DMP may not be a feasible to analyze ME/CFS and its related medical syndromes. The limited understanding of ME/CFS, the lack of clinically proven representative movement primitives, the variability and inconsistency in human motor function/performance, and the lack of clear and consistent movement patterns all make it difficult to develop accurate models of movement.

3.3.3 Time-Series Data Classification with Transformer

Transformers, originally designed for language modeling, have emerged as an efficient alternative to recurrent neural networks in various sequential learning tasks [10]. However, they may not be ideally suited

[4] A very good short explainer (Patterns of Behavior in the System): `https://content.csbs.utah.edu/~butner/systems/DynamicalSystemsIntro.html`.

for our specific use case. Our dataset contains only 51 data points, which is insufficient for training a model with the desired performance. Furthermore, each data point in our dataset consists of millions of rows, posing significant challenges for processing and using as input for a Transformer model.

In our initial attempt to develop an encoder and decoder for a Transformer, we encountered limitations due to our current level of expertise. Nonetheless, given the recent trend in Generative Pre-trained Transformer (GPT), there is significant potential for creating a time-series specific model based on a similar architecture. Pursuing this direction in future work could yield substantial improvements in time-series data classification and analysis for orthostatic intolerance, ME/CFS, and Long COVID research.

3.3.4 Other Possible Approaches

During our two years of research experience in the time-series data processing field, we have identified some areas that warrant further exploration and development. In particular, we noticed a relative lack of contributions compared to other fields.

One potential approach is to address the limitations of consumer-level electronics, which often do not provide sufficient API access for developers to collect low-level IMU data and perform real-time analysis. Off-the-shelf IMUs are generally more expensive than their manufacturing cost, which makes the development of custom hardware tailored to our data collection needs an attractive alternative. Such custom hardware could include an IMU with a larger battery, increased storage capacity, and improved low-level API access.

Another area for improvement is in the feature extraction process. Given the size of our dataset, feature extraction can be cumbersome and inefficient. Developing a distributed time-series feature extraction tool, that could provide similar features to `tsfresh`, could prove beneficial. Additionally, exploring advanced machine learning techniques to overcome the limitations of small dataset size with unusually large data points may lead to more robust findings and improved analysis. By pursuing these future research directions, we aim to enhance the overall understanding and treatment of orthostatic intolerance, ME/CFS, and Long COVID.

3.4 Conclusions

The primary objective of our research was to design a system prototype for collecting IMU data and streamlining the data processing workflow, with the ultimate goal of using our proposed parameters/features to assess the severity of ME/CFS and Long COVID.

Utilizing MbientLab's MetaMotionS IMU and our system prototype, MetaProcessor, we analyzed data from the EndoPAT study and determined that UpTime is an objective digital biomarker with superior overall performance compared to HUA and Steps/Day. Upon further validation, UpTime could be confidently proposed to regulators as an objective clinical measure for disease severity classification in the context of ME/CFS and related medical conditions. Additionally, HUA and Steps/Day could serve as supplementary biomarkers to further enhance the accuracy of the assessment.

References

[1] Sanghee Cho et al. "Digital timing: sampling frequency, anti-aliasing filter and signal interpolation filter dependence on timing resolution". en. In: *Phys. Med. Biol.* 56.23 (Dec. 2011), pp. 7569–7583.

[2] Maximilian Christ, Andreas W. Kempa-Liehr, and Michael Feindt. "Distributed and parallel time series feature extraction for industrial big data applications". In: *CoRR* abs/1610.07717 (2016). arXiv: 1610.07717. URL: http://arxiv.org/abs/1610.07717.

[3] Maximilian Christ et al. "Time Series FeatuRe Extraction on basis of Scalable Hypothesis tests (tsfresh – A Python package)". In: *Neurocomputing* 307 (2018), pp. 72–77. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2018.03.067. URL: https://www.sciencedirect.com/science/article/pii/S0925231218304843.

[4] Committee On The Diagnostic Criteria For Myalgic Encephalomyelitis/Chronic Fatigue Syndrome, Board on the Health of Select Populations, and Institute of Medicine. *Beyond myalgic encephalomyelitis/chronic fatigue syndrome*. Washington, D.C., DC: National Academies Press, Mar. 2015.

[5] Centers for Disease Control and Prevention. *Diagnosis of ME/CFS*. Jan. 2021. URL: https://www.cdc.gov/me-cfs/symptoms-diagnosis/diagnosis.html.

[6] Centers for Disease Control and Prevention. *Long COVID or Post-COVID Conditions*. Sept. 2022. URL: https://www.cdc.gov/coronavirus/2019-ncov/long-term-effects/index.html.

[7] Centers for Disease Control and Prevention. *What is ME/CFS?* Jan. 2021. URL: https://www.cdc.gov/me-cfs/about/index.html.

[8] Jihyun Lee et al. "Clinically accessible tools for documenting the impact of orthostatic intolerance on symptoms and function in ME/CFS". en. In: *Work* 66.2 (2020), pp. 257–263.

[9] Jihyun Lee et al. "Clinically accessible tools for documenting the impact of orthostatic intolerance on symptoms and function in ME/CFS". en. In: *Work* 66.2 (2020), pp. 257–263.

[10] Yuxuan Liang et al. "TrajFormer: Efficient Trajectory Classification with Transformers". In: *Proceedings of the 31st ACM International Conference on Information; Knowledge Management*. CIKM '22. Atlanta, GA, USA: Association for Computing Machinery, 2022, pp. 1229–1237. ISBN: 9781450392365. DOI: 10.1145/3511808.3557481. URL: https://doi.org/10.1145/3511808.3557481.

[11] Huicong Liu et al. "A non-resonant rotational electromagnetic energy harvester for low-frequency and irregular human motion". en. In: *Appl. Phys. Lett.* 113.20 (Nov. 2018), p. 203901.

[12] Rebecca Marshall, Lorna Paul, and Les Wood. "The search for pain relief in people with chronic fatigue syndrome: a descriptive study". en. In: *Physiother. Theory Pract.* 27.5 (July 2011), pp. 373–383.

[13] Institute of Medicine. *Beyond Myalgic Encephalomyelitis/Chronic Fatigue Syndrome: Redefining an Illness*. Washington, DC: The National Academies Press, 2015. ISBN: 978-0-309-31689-7. DOI: `10.17226/19012`. URL: `https://pubmed.ncbi.nlm.nih.gov/25695122/`.

[14] Kyle W Murdock et al. "The utility of patient-reported outcome measures among patients with myalgic encephalomyelitis/chronic fatigue syndrome". en. In: *Qual. Life Res.* 26.4 (Apr. 2017), pp. 913–921.

[15] Turner Palombo. "Development of an Inertial Measurement-Based Assessment of Disease Severity in Chronic Fatigue Syndrome". MA thesis. University of Utah, 2020.

[16] Sue Pemberton and Diane L Cox. "Experiences of daily activity in chronic fatigue syndrome/myalgic encephalomyelitis (CFS/ME) and their implications for rehabilitation programmes". en. In: *Disabil. Rehabil.* 36.21 (2014), pp. 1790–1797.

[17] Judith B Prins, Gijs Bleijenberg, and Jos W M van der Meer. "Chronic fatigue syndrome and myalgic encephalomyelitis". en. In: *Lancet* 359.9318 (May 2002), p. 1699.

[18] Matteo Saveriano et al. *Dynamic Movement Primitives in Robotics: A Tutorial Survey*. 2021. DOI: `10.48550/ARXIV.2102.03861`. URL: `https://arxiv.org/abs/2102.03861`.

[19] Laura Solomon and William C Reeves. "Factors influencing the diagnosis of chronic fatigue syndrome". en. In: *Arch. Intern. Med.* 164.20 (Nov. 2004), pp. 2241–2245.

[20] Kyle Strimbu and Jorge A Tavel. "What are biomarkers?" en. In: *Curr. Opin. HIV AIDS* 5.6 (Nov. 2010), pp. 463–466.

[21] Timothy L Wong and Danielle J Weitzer. "Long COVID and myalgic encephalomyelitis/chronic fatigue syndrome (ME/CFS)-A systemic review and comparison of clinical presentation and symptomatology". In: *Medicina (Kaunas)* 57.5 (Apr. 2021). URL: `https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8145228/`.

A UpTime Algorithm (Python)

Following Python snippet is derived from the original upright position time algorithm written in MATLAB by Turner Palombo, it is Incorporated into MetaProcessor then extracted from the code base with extra comments, please refer to MetaProcessor's GitHub repository to see the usage in context.

```python
def uptime(filename: str) -> pd.DataFrame:
    start_time = datetime.now()
    df = read(filename)

    # initialize required dataframes
    uptime = pd.DataFrame(columns=["epoc (ms)", "upright (0/1)", "angle (deg)"])
    filtered = pd.DataFrame(columns=[
        "epoc (ms)",
        "x-axis (m/s^2)", "y-axis (m/s^2)", "z-axis (m/s^2)",
        "x-axis (rad/s)", "y-axis (rad/s)", "z-axis (rad/s)"
    ])

    # convert g to m/s^2
    df["x-axis (m/s^2)"] = df["x-axis (g)"] * constants.g
    df["y-axis (m/s^2)"] = df["y-axis (g)"] * constants.g
    df["z-axis (m/s^2)"] = df["z-axis (g)"] * constants.g
    # convert deg/s to rad/s
    df["x-axis (rad/s)"] = np.deg2rad(df["x-axis (deg/s)"])
    df["y-axis (rad/s)"] = np.deg2rad(df["y-axis (deg/s)"])
    df["z-axis (rad/s)"] = np.deg2rad(df["z-axis (deg/s)"])

    # time between two sample points
    dt = 1 / SAMPLE_RATE
    # 2nd order 10 Hz low-pass
    [b, a] = scipy.signal.butter(
        2,
        10 / (SAMPLE_RATE / 2),
        "low"
    )
    # timestamp
    uptime["epoc (ms)"] = df["epoc (ms)"]
    filtered["epoc (ms)"] = df["epoc (ms)"]

    # apply filter to each axis
    filtered["x-axis (m/s^2)"] = scipy.signal.lfilter(b, a, df["x-axis (m/s^2)"])
    filtered["y-axis (m/s^2)"] = scipy.signal.lfilter(b, a, df["y-axis (m/s^2)"])
    filtered["z-axis (m/s^2)"] = scipy.signal.lfilter(b, a, df["z-axis (m/s^2)"])
    filtered["x-axis (rad/s)"] = scipy.signal.lfilter(b, a, df["x-axis (rad/s)"])
    filtered["y-axis (rad/s)"] = scipy.signal.lfilter(b, a, df["y-axis (rad/s)"])
    filtered["z-axis (rad/s)"] = scipy.signal.lfilter(b, a, df["z-axis (rad/s)"])

    # trigonometric estimations of roll and pitch using raw accerometer data
    filtered["acce-phi-hat"] = np.arctan2(
        filtered["y-axis (m/s^2)"],
        np.sqrt(filtered["x-axis (m/s^2)"] ** 2 + filtered["z-axis (m/s^2)"] ** 2),
    )
    filtered["acce-theta-hat"] = np.arctan2(
        -filtered["x-axis (m/s^2)"],
        np.sqrt(filtered["y-axis (m/s^2)"] ** 2 + filtered["z-axis (m/s^2)"] ** 2),
    )

    # state-space form of kalman filter
    a = np.array([
        [1, -dt, 0,   0],
        [0,   1, 0,   0],
        [0,   0, 1, -dt],
        [0,   0, 0,   1],
    ])
    b = np.array([
        [dt, 0, 0,  0],
        [0,  0, dt, 0],
    ]).T
    c = np.array([
        [1, 0, 0, 0],
        [0, 0, 1, 0],
    ])

    # initial error covariance matrix (4 x 4 identity matrix init with 1)
    # large values = unsure if initial state is correct
    # small values = confident that initial guess is correct
    p = np.eye(4) * 1

    # proces covariance matrix (4 x 4 identity matrix init with 0.01)
    # large values = model is inaccurate
    # small values = model is accurate
    q = np.eye(4) * 0.01

    # measurement noise covariance matrix (2 x 2 identity matrix init with 10)
    # large values = greater sensor noise
    # small values = minimal sensor noise
    r = np.eye(2) * 10

    # initial value estimate (4 x 1 matrix)
    state_estimate = np.array([
        constants.pi / 2,
        0,
        0,
        0,
    ]).T

    # vector initialization
    phi        = np.zeros(len(filtered))
    bias_phi   = np.zeros(len(filtered))
    theta      = np.zeros(len(filtered))
    bias_theta = np.zeros(len(filtered))

    # kalman filter
    for i in range(len(filtered)):
        gryo_x = filtered["x-axis (rad/s)"][i]
        gryo_y = filtered["y-axis (rad/s)"][i]
        gryo_z = filtered["z-axis (rad/s)"][i]

        phi_hat   = phi[i - 1]
        theta_hat = theta[i - 1]

        # generate input vector
        phi_dot = gryo_x + \
            np.sin(phi_hat) * np.tan(theta_hat) * gryo_y + \
            np.cos(phi_hat) * np.tan(theta_hat) * gryo_z
        theta_dot = np.cos(phi_hat) * gryo_y - \
            np.sin(phi_hat) * gryo_z

        # predict state
        state_estimate = a @ state_estimate + \
            b @ np.array([
                phi_dot,
                theta_dot,
            ]).T

        # predict error covariance
```

```python
        p = a @ p @ a.T + q

        # update
        measurement = np.array([
            filtered["acce-phi-hat"][i],
            filtered["acce-theta-hat"][i],
        ]).T

        y_tilde = measurement - c @ state_estimate

        s = r + c @ p @ c.T
        k = p @ c.T @ np.linalg.inv(s)

        state_estimate = state_estimate + k @ y_tilde

        p = (np.eye(4) - k @ c) @ p

        phi[i]        = state_estimate[0]
        bias_phi[i]   = state_estimate[1]
        theta[i]      = state_estimate[2]
        bias_theta[i] = state_estimate[3]

    # convert phi and theta to a single angle output using quaternions
    roll  = constants.pi / 2 - phi
    pitch = theta

    qr = np.cos(roll / 2) * np.cos(pitch / 2)
    qi = np.sin(roll / 2) * np.cos(pitch / 2)
    qj = np.cos(roll / 2) * np.sin(pitch / 2)
    qk = np.sin(roll / 2) * np.sin(pitch / 2)

    # angle in radians
    angle = 2 * np.arctan2(
        np.sqrt(qi ** 2 + qj ** 2 + qk ** 2),
        qr,
    )
    # angle in degrees
    angle = np.degrees(angle)

    # insert angle into dataframe
    uptime["angle (deg)"] = angle

    # upright
    upright = np.zeros(len(uptime))
    for i in range(len(angle)):
        if CRITICAL_ANGLE > angle[i]:
            upright[i] = 1

    upright_percentage = np.sum(upright) / len(upright) * 100
    uptime["upright (0/1)"] = upright

    end_time = datetime.now()
    print(
        f"{filename}:\n"
        f"Upright percentage: {upright_percentage}%\n"
        f"Execution: {end_time - start_time}"
    )

    return uptime
```

B Steps/Day Algorithm (Python)

Following Python snippet is derived from the original step counter algorithm written in MATLAB by McKenzie Hoggan, it is Incorporated into MetaProcessor then extracted from the code base with extra comments, please refer to MetaProcessor's GitHub repository to see the usage in context.

```python
def read(filename: str) -> pd.DataFrame:
    df = pd.read_csv(filename, engine="pyarrow")
    if df.isnull().sum().sum() == 0:
        df = df.astype({
            "epoc (ms)": int,
            "x-axis (g)": float,
            "y-axis (g)": float,
            "z-axis (g)": float,
            "x-axis (deg/s)": float,
            "y-axis (deg/s)": float,
            "z-axis (deg/s)": float,
        })
    else:
        df = df.fillna(method="ffill").fillna(method="bfill")
        if df.isnull().sum().sum() != 0:
            raise ValueError(f"cannot fill NaN values in {filename}")
    return df


def steps_per_day(filename: str) -> int:
    start_time = datetime.now()
    df = read(filename)

    df = 9.81 * df.iloc[:, 1:4]
    logic = np.ones(len(df))
    steps = np.zeros(len(df))
    l = 1

    # x-axis
    acc = df.iloc[:, 0]
    # y-axis
    # acc = rawAcc.iloc[:, 1]
    w = 10
    steps = 0

    n = range(0, len(df))
    logic = logic[n]

    # compute local variance
    var = np.convolve(acc[n], np.ones((w,))/w, mode='valid')
    pks = np.array([])

    pks = np.array([])
    for i in range(len(var)):
        if (var[i] > 0.1):
            pks = np.append(pks, var[i])

    if (len(pks) != 0):
        thresh = min(pks)
        D1 = thresh*2
        D2 = D1/2
        B = np.zeros(len(var))

        for i in range(len(var)):
            if (var[i] >= thresh):
                B[i] = D1
            else:
                B[i] = D2

        F = np.zeros(len(B))

        F[0] = 1

        for i in range(1, len(B)):
            if (B[i-1] > B[i]):
                F[i] = -1
            elif (B[i-1] < B[i]):
                F[i] = 1

        # count up the steps
        if (F[0] == 1 and logic[i] == 1):
            steps = 1

        for i in range(1, len(F)):
            if (F[i] == 1):
                steps = steps + 1
            elif (F[i] == -1 and F[i-1] == -1 and logic[i] == 1):
                steps = steps + 1

    end_time = datetime.now()
    print(
        f"{filename}:\n"
        f"Steps: {steps}\n"
        f"Execution: {end_time - start_time}"
    )

    return steps
```

C T-Test Results for Survey Data Collected from EndoPAT Study

The figures presented in this appendix section were produced using standardized survey data collected from study participants during their visits to The Bateman Horne Center, as well as data collected through online forms. While we did not formulate specific hypotheses for each of the features depicted in these figures, we have the necessary tools and data to generate them. As such, we have included these figures for future reference.

Included features are: Augmentation Index (AI), Augmentation Index Normalized to HR 75 bpm (AI75), Baseline Heart Rate (bpm) (BLHR), Natural Base Log of Reactive Hyperemia Index (LnRHI), Reactive Hyperemia Index (RHI), and Hours of Upright Activity (HUA).

We would like to acknowledge the invaluable contribution of Dr. Suzanne Vernon, Research Director at The Bateman Horne Center, and the staff members at the center for providing us with this survey data used in this study. Their efforts in collecting and compiling this data have been instrumental in enabling us to conduct our research and draw meaningful conclusions.

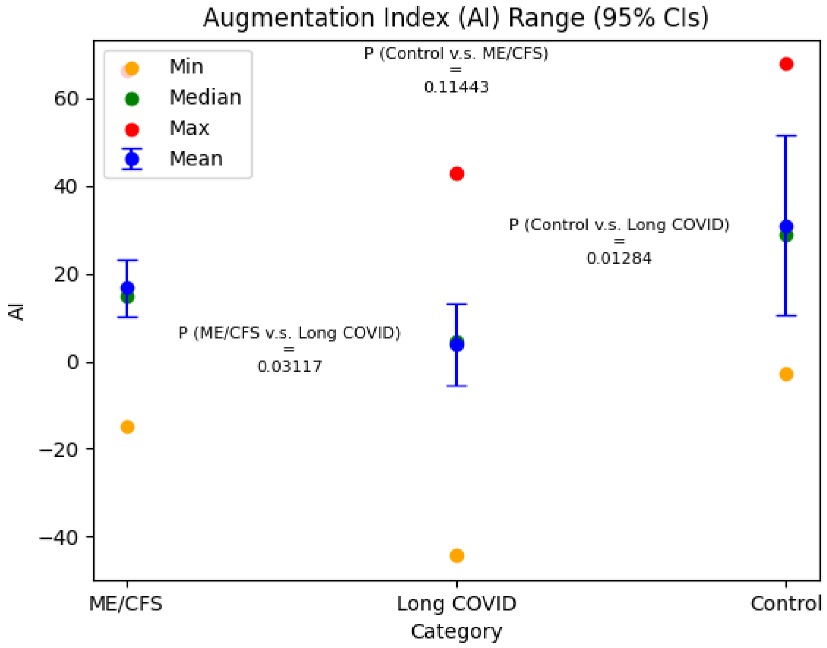

Figure C.1: Grouped t-test performed at 95% confidence level for Augmentation Index, ME/CFS vs. Long COVID vs. Control, 95% confidence interval.

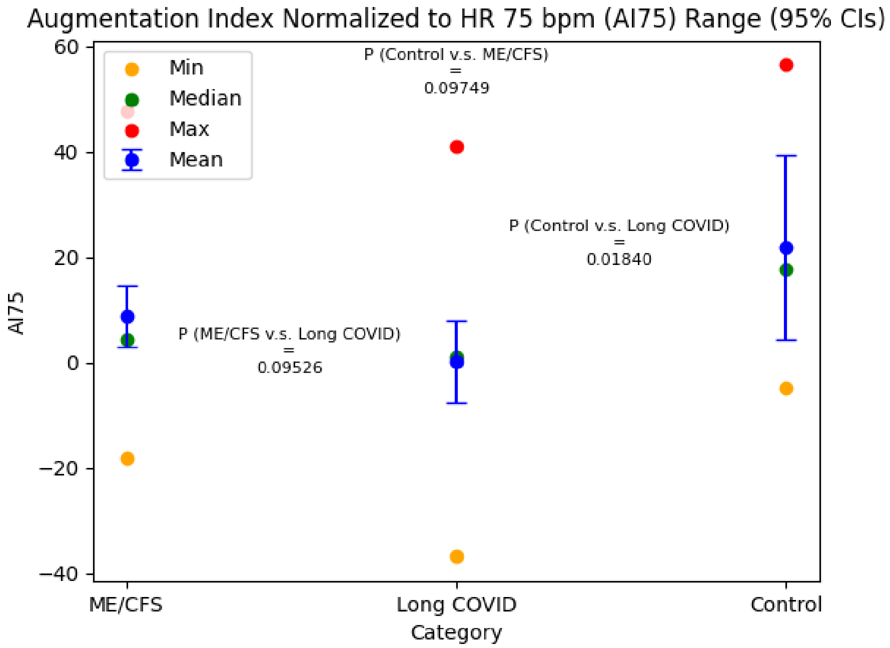

Figure C.2: Grouped t-test performed at 95% confidence level for Augmentation Index Normalized to HR 75 bpm, ME/CFS vs. Long COVID vs. Control. In the legend, “Mean” is mean plus or minus a standard error.

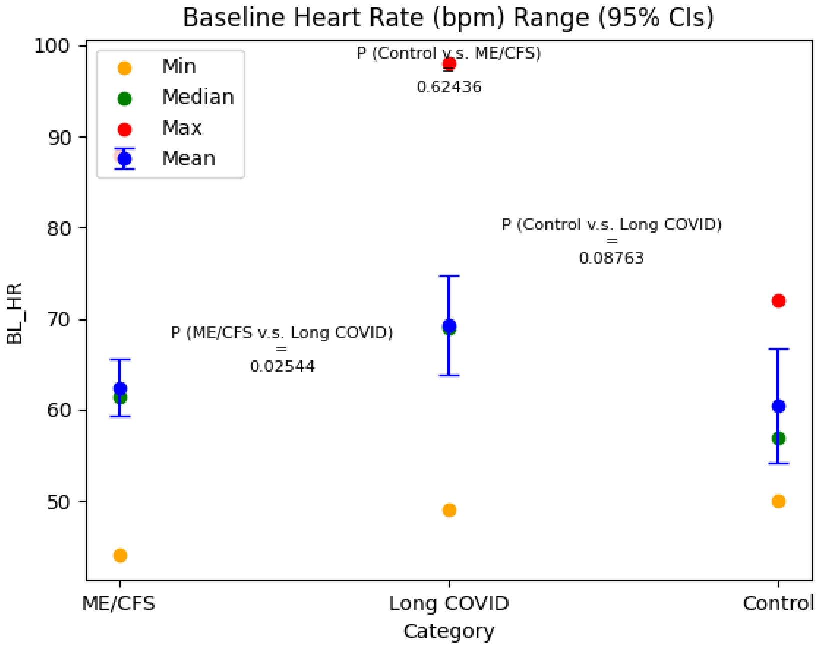

Figure C.3: Grouped t-test performed at 95% confidence level for Baseline Heart Rate (bpm), ME/CFS vs. Long COVID vs. Control. In the legend, "Mean" is mean plus or minus a standard error.

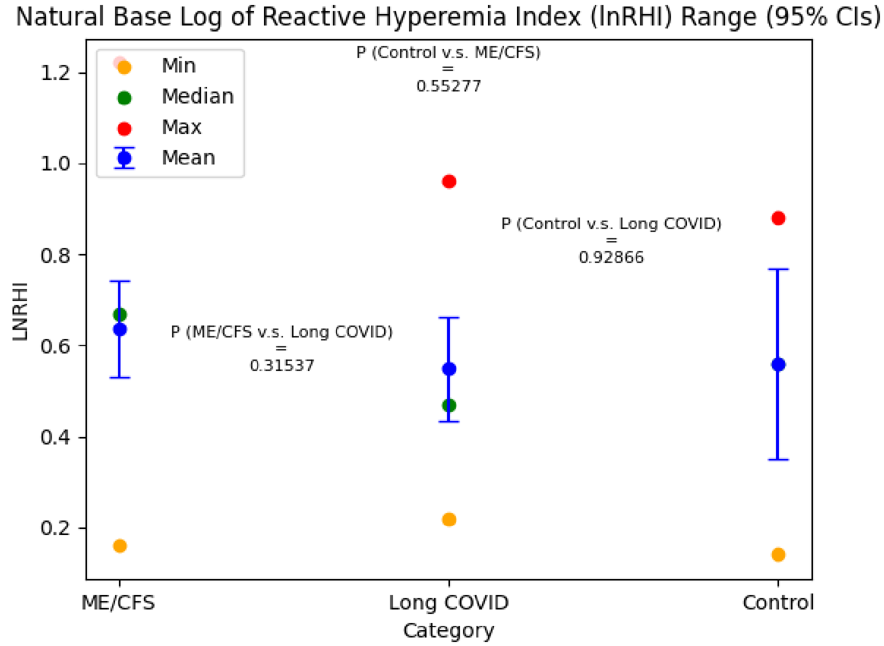

Figure C.4: Grouped t-test performed at 95% confidence level for Natural Base Log of Reactive Hyperemia Index, ME/CFS vs. Long COVID vs. Control. In the legend, "Mean" is mean plus or minus a standard error.

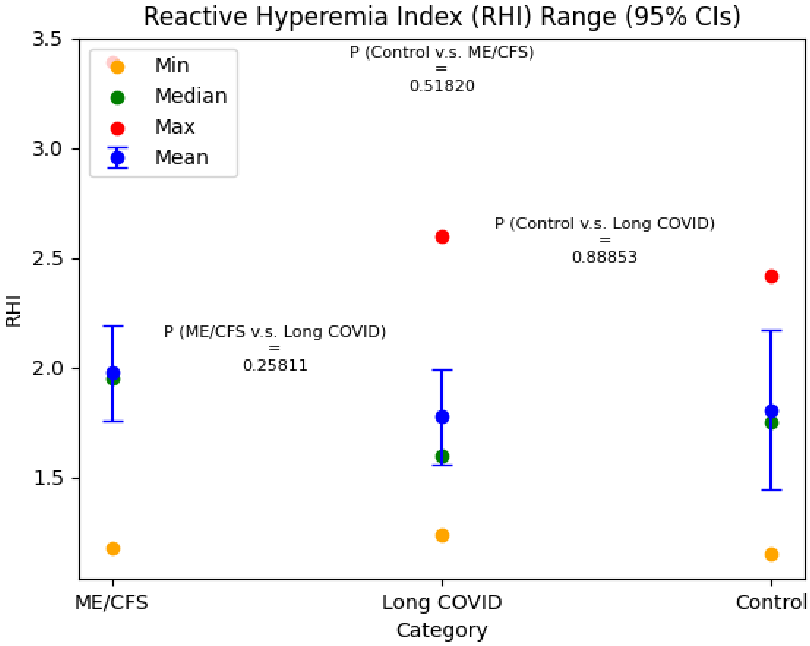

Figure C.5: Grouped t-test performed at 95% confidence level for Reactive Hyperemia Index, ME/CFS vs. Long COVID vs. Control. In the legend, “Mean” is mean plus or minus a standard error.