Arora: Python Package for Sleep Revolution

Anton Björn Mayböck Helgason,
Björgvin Ægir Elisson,
Eva Björg Nåbye,
Margrét Sól Aðalsteinsdóttir

Thesis of 12 ECTS credits submitted to the Department of Computer Science at Reykjavík University in partial fulfillment of the requirements for the degree of Bachelor of Science

May 13, 2022

Thesis Committee:

María Óskarsdóttir, Supervisor
Reykjavik University, Iceland,

Benedikt Hólm Þórðarson, Supervisor
Reykjavik University, Iceland,

Anna Sigríður Íslind, Examiner
Reykjavik University, Iceland,

This page was left intentionally blank

Arora: Python Package for Sleep Revolution

Anton Björn Mayböck Helgason,
Björgvin Ægir Elisson,
Eva Björg Nåbye,
Margrét Sól Aðalsteinsdóttir

May 13, 2022

Abstract

This report discusses a B.Sc. project done at Reykjavik University on sleep research, more specifically Arora, a Python package developed as a tool for sleep researchers to assist with data reading, preprocessing, feature extraction, and visualization. The package is thought of as a tool for the first steps in sleep data processing before any extensive data analysis or machine learning is done on the data. The report discusses the workflow of the project and the methodology used, along with how the development and testing of the package were conducted. The product is a solid foundation of a Python package but the last chapter examines how the future development and the missing functionality can be implemented. This project is done in collaboration with Sleep Revolution, a project funded by the European Union, focused on researching obstructive sleep apnea with cutting-edge methods and technology.

Arora: Python Pakki fyrir Svefnbyltinguna

Anton Björn Mayböck Helgason,
Björgvin Ægir Elisson,
Eva Björg Nåbye,
Margrét Sól Aðalsteinsdóttir

13. maí 2022

Útdráttur

Þessi skýrsla ræðir verkefni til Bakkalársgráðu við Háskólann í Reykjavík um svefnrannsóknir, nánar tiltekið Arora, Python pakka þróaðan sem verkfærakassi fyrir svefnrannsakendur til að aðstoða við lesningu gagna, forvinnslu þeirra, úrdrátt sérkenna, og myndbirtingu. Pakkinn er hugsaður sem tól fyrir fyrstu skrefin í úrvinnslu svefngagna áður en umfangsmikilli gagnagreiningu eða vélrænu gagnanámi er beitt á gögnin. Skýrslan talar um vinnuferli verkefnisins og aðferðafræðina sem var notuð, ásamt því hvernig þróunarferli pakkans og prófunum hans var háttað. Afurðin er áreiðanlegur grunnur af Python pakka en síðasti kaflinn rannsakar hvernig þróun pakkans til framtíðar getur verið háttað ásamt því hvernig virknin sem vantar getur verið innleidd. Þetta verkefni var gert í samstarfi við Svefnbyltinguna sem er rannsóknarverkefni fjármagnað af Evrópusambandinu, sem rannsakar kæfisvefn með háþróuðum aðferðum og tækni.

Acknowledgements

We would like to take some time to thank our wonderful instructors, María Óskarsdóttir and Benedikt Hólm Þórðarson, for being a helping hand throughout the project and offering words of encouragement when needed. This package and this project would not be where they are without them.

Additionally, we would like to thank Jacopo Piccini, Katrín Hera Gústafsdóttir, and Matias Rusanen for their code contribution to the package.

We would also like to thank everyone who graciously gave us their time and expertise in the process of developing Arora. Whether it was in testing, interviews, or in other ways.

Lastly, we are grateful for everyone who kept us sane over the course of the semester, you know who you are.

Contents

List of Figures

List of Tables

List of Abbreviations

Abbreviation	Full name
OSA	Obstructive sleep apnea
PSG	Polysomnography
EEG	Electroencephalogram
REM	Rapid Eye Movement
PET	Positron Emission Tomography
GUI	Graphical User Interface
PEP	Python Enhancement Proposal
EDF	European Data Format
ACF	Autocorrelation function
PE	Prediction Error
LP	Linear Prediction
SEM	Spectral Error Measure
RAM	Random Access Memory

Chapter 1

Introduction

1.1 Introduction

Sleep is vital to every person's health and plays a critical role in a person's life [3]. Sleep disorders and their effects can therefore have negative implications on one's life in numerous ways [4]. Disrupted sleep can, for example, have an impact on memory, attention, cognitive functions, and accuracy as well as drastically deteriorating one's mental health by increasing stress and anxiety [4]. In the last ten years or so, sleep journals and studies have grown tremendously, showcasing the effects of sleep disruption, such as OSA and sleep deprivation [4].

Sleep data is generally collected in PSGs, a combination of EEG signals, muscle activity, breathing, and eye movement measurements [5]. However, the signals gathered from these methods are difficult to decipher manually and only experienced and certified sleep technologists can do so [1]. Future sleep researchers are trying to get over that hurdle by using and developing automated methods, e.g. machine learning techniques, that could make data scoring more efficient and reliable [1].

Sleep Revolution is an international research and development project led by scientists at Reykjavík University that aims to revolutionize modern sleep research by implementing a new diagnostic and digital management paradigm. It focuses on sleep disorders, especially OSA and by incorporating machine learning techniques they plan to estimate the severity and treatment of the disorder, to improve health outcomes and quality of life.

In the past, sleep measurements have primarily been done in a hospital environment and the patient's sleep is usually only measured for one night [6]. This arrangement is not nearly good enough to gather sufficient data since it usually takes a person at the very least one night to get used to all of the tools that are used for the measurements as well as the unfamiliar environment [7]. With the tools that Sleep Revolution has accumulated, the patient can record their measurements in the safety of their homes and can wear the devices for multiple nights, making the data gathering much more reliable [6].

For our final assignment, we were tasked with creating a Python package called Arora which further helps this project by enabling researchers to work more efficiently with sleep data from different sources. The package is meant to be an end-to-end solution for doing analysis and machine learning with sleep data, from data import to model evaluation.

Our main focus was on data pre-processing such as importing the data, segmenting it, feature extraction, and visualization. The package makes sure to fit to pre-established standards of other Python libraries such as Pandas, SciPy, and Seaborn. This will in turn assist future research from Sleep Revolution by making the data standardized and thus more efficient and easier to work with. The package was tested by adding unit tests and with user testing.

The Arora package will help people working in Sleep Revolution by making their jobs easier. There has been an issue with code redundancy when working with the data provided by Sleep Revolution, where people all fix the same issue in their code in different ways without knowing that another person has already done so as well. Arora will make this issue a thing of the past with a consistent codebase that helps solve trivial problems for anyone who requires it. In turn, allowing members working on the project to manage their time more efficiently.

The package will also be well documented with an emphasis on readability so that people can use it regardless of their programming knowledge and skill. Therefore researchers working on the project from other fields can be more self-sufficient in their applications. The package will be structured so that it is possible to build more functionalities on top of the ones that already exist so if other problems arise when working through the data it is simple to add the solution to the package. The solution that Arora has to offer is necessary since there are no

other comparable options available. It will also serve as a foundation for future research inside Sleep Revolution.

This project report describes the process of developing the package from start to finish, with both the methods applied and their respective results. It will go over related works, where tools that are currently used for sleep research and data gathering are discussed more thoroughly. It also covers the next steps in the development of the package where future developers can see ideas for further implementation.

Accompanying this report is an operation manual that goes over the steps to import the package and helps with understanding how to use Arora. The documentation then serves as a user manual and can be found at `https://sleep-revolution-sleepy.readthedocs-hosted.com/en/latest/`.

Chapter 2

Related Works

2.1 Related Works

2.1.1 Tools for Sleep Research

In 1875, Caton was the first to record brain electrical activities in animals but in 1929 it was Berger who first studied the EEG of man and it has been the backbone of sleep studies and sleep medicine since then and is the origin of modern sleep research [5, 8].By examining EEG patterns that occur during sleep, classification of stages of sleep were defined, that in turn, created the foundation of studying human sleep, sleep disruptions, and discovering the significant relation between sleep and health [4]. Since 2005 the pace of research and discovery has increased rapidly and the number of peer-reviewed sleep journals has more than tripled but it still has a long way to go [4].

The EEG is now combined with many different techniques; monitoring body and brain temperature, changes in brain and blood chemistry, and changes in brain functions [8]. Researchers and clinical practices need to achieve the right balance of subjective and objective measures of sleep seeing that the future directions in clinical trials in sleep medicine will emphasize patient engagement as well as patient-centered outcomes[4]. The golden standard for objective assessment is PSG, which combines EEG signals for brain activity, muscle activity (electromyogram), and eye movement (electrooculogram) which differentiates between sleep stages [4, 8]. EEG channels can extend to up to 256 in humans, enabling detailed localization of changes across the brain, making it so valuable for sleep research and is the reason why it is still one of the most important tools in the industry [8].

Another type of tool frequently used is actigraphs, most often a wrist-worn activity monitor that records physical activities [8]. Actigraphy is a standard procedure to observe rhythms in humans who are entering an experiment, commonly worn 1-2 weeks before it where activity is accumulated over a predetermined epoch, or used to do a preliminary screen for sleep or circadian rhythm pathology in subjects [8]. Actigraphs help minimize recall bias and complement subjective measures of sleep but they still have drawbacks since wearable devices tend to overestimate time spent sleeping and currently are not capable of accurately measuring changes in sleep changes in an individual such as non-REM and REM [4]. Dr. Clete A. Kushida, a sleep specialist believes that in five to ten years these types of devices will most likely be able to accurately estimate sleep and sleep stages since the product cycle for new sleep tools are progressing very rapidly [4].

Body and brain temperatures are also standard in sleep studies and in PSG a recording of those temperatures at different locations is sometimes recorded simultaneously to determine heat-loss changes during sleep and wakefulness to obtain information in relations between sleep and thermoregulation [8]. PET scans have also shown to be successful in various sleep researches in humans but they are not a real time-measurement and are therefore harder to examine [8].

All of the tools mentioned above are valuable, non-invasive methods for examining sleep and the ones that are most commonly used. Other more invasive methods have been tested on animals but are not yet applicable to human testing [4, 8].

2.1.2 Sleep Data Analysis with Machine Learning

Several methods are used to extract information from sleep data known as data mining and analytics, preprocessing it and isolating the noise and errors and extracting useful variables, finding the patterns between measurements and events, summarizing the data to give an overview of the main aspects and visualizing the data to look for useful trends in it. The most commonly

used model for these tasks is the cross-industry data model [1]. It is split into six phases and this

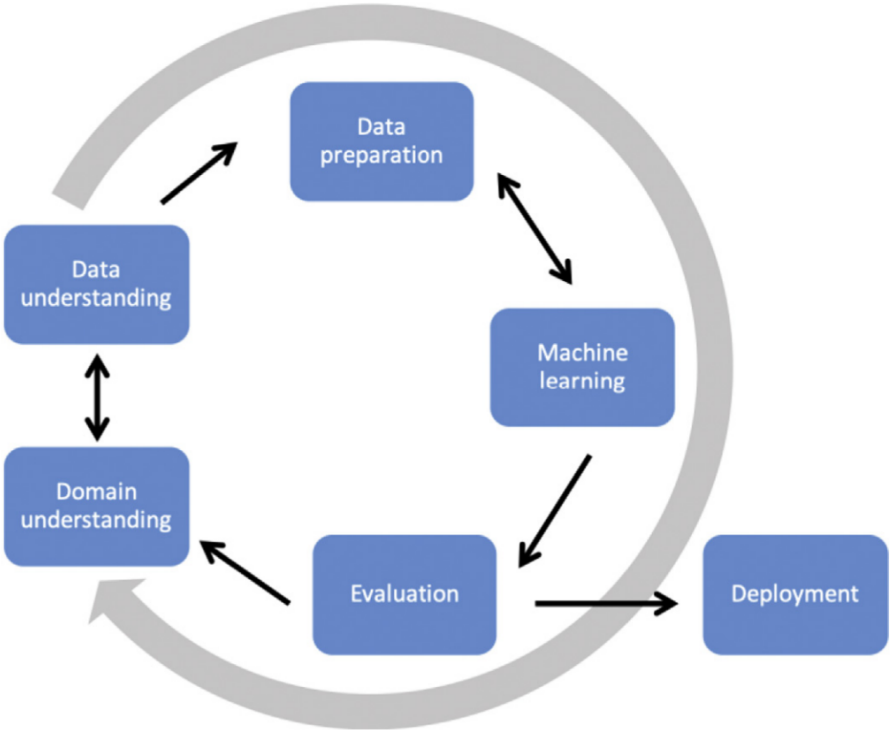


Figure 2.1: Picture of the cross-industry model [1, Fig. 4]

subsection will focus on the machine learning application in this model but machine learning is often considered to be the future of sleep measurements and data mining and the prevalence continues to grow. It is a prime contender to automatically analyze PSG data.

The three main aspects of machine learning are unsupervised learning, supervised learning, and reinforcement learning. Unsupervised learning in sleep analysis can be used to find different sleep patterns in the long-term data of one person or phenotypes in larger datasets and can for example identify sleep stages. Supervised learning uses labeled data, events, or class labels of the observation and can be used to find disruptions in sleep like OSA and predict future probabilities of having health-related problems. Reinforcement learning learns optimal behavior with an algorithm for a given task and uses trial and error to maximize the long-term reward [1].

There are, however, limitations to using machine learning in sleep measurements and research. The need for labeled data in supervised machine learning requires manual scoring that can only be accomplished with experienced sleep technologists or somonologists and takes a lot of time. Another limitation is the need for computational power but cloud solutions are becoming more readily available and can be trained in a reasonable time. This in turn may raise an issue in data security since the data collected is often sensitive and requires expert knowledge and training [1].

2.1.3 Python Packages for Sleep Researchers

There have been a few Python packages developed to help analyze sleep data, one of which is simply called Sleep. Sleep is a GUI for visualization and scoring of PSG sleep data designed to support and process copious amounts of data. It loads and reads raw EEG data from standard file formats as well as a range of other commercial data formats [9]. This package is built on top of another library called VisPy, a high-performance data visualization library [9]. The Sleep package is functional but Arora is more specified towards the Sleep Revolution project and allows for more control and customizability in what the package should contain.

Chapter 3

Project workflow

3.1 Project Workflow

3.1.1 Project Workflow

Throughout developing the Arora package, the team worked with the agile development methodology, where the project was split into sprints ranging from one to two weeks of work. There were meetings set every other day and on Fridays we met up with the instructors of the project where we discussed what had been accomplished during the week and the next steps of the development. The group also had regular meetings and online communications with other students working on their final assignments for Sleep Revolution as well since the projects often intertwined with one another. After each sprint, there was a meeting where the team went over the sprint retrospectively, what we accomplished, what went wrong, and how it is possible to prevent that for future sprints as well as planning our steps for the upcoming weeks. Tables 3.1 and 3.2 show the total hours worked by the team in each sprint and the total hours worked by each team member. Additionally, appendix D depicts a burndown chart showing the total time worked by the team vis-à-vis the total time estimated for the project.

In addition to the sprints and their management, the group developed a risk analysis, consisting of possible factors that could hinder progress over the semester. A detailed risk analysis can be seen in figure B.1. To get a clearer picture of the project as a whole, a list of requirements was cultivated which can be seen in detail in appendix A.

3.1.2 Hours Worked

Table 3.1: Total Hours worked for each sprint

Date	Sprint	Hours worked (total)
17.01-23.01	Sprint 0	48
24.01-06.02	Sprint 1	98
07.02-20.02	Sprint 2	85
21.02-06.03	Sprint 3	128
07.03-20.03	Sprint 4	105
21.03-03.04	Sprint 5	56
04.04-17.04	Sprint 6	280.5
18.04-01.05	Sprint 7	320.5
02.05-13.05	Sprint 8	355

Table 3.2: Total hours worked for each group member

Name	Hours Worked (total)
Anton	417
Björgvin	307.5
Eva	366
Margrét	359.5

Since this is a project involving a large number of individuals, many of whom have been working with the data we were meant to process for a while, we reached out to some of them for interviews and asked what struggles they have been experiencing whilst working on Sleep Revolution for a better understanding of what was expected of us and what was within the scope

of the project. The requirements for each sprint were written down in Trello, an interactive task board, and the group split them down, each group member getting different tasks to work on and helped to visualize what was to be done.

Over these 15 weeks, the work organization did not always go as planned, with meetings being canceled, members catching Covid-19, and other courses getting in the way of the development, the last three weeks went to finishing all the requirements that were yet to be done with everyone meeting every day for 8-12 hours.

3.1.2.1 Technical Environment

As previously mentioned in the report, Arora will be an end-to-end solution for processing, analysis, and machine learning with sleep data, programmed in Python, a high-level, object-oriented programming language. The code uses other pre-established Python libraries such as SciPy, Pandas, and NumPy. The reason for these pre-existing libraries being used is so that the group can use their time more efficiently without having to code menial functions that already exist and can therefore focus on other, more important tasks. The code follows the PEP 8 style guide [10] for better readability so that future developers can understand it sufficiently and keep working on it. For automatic analysis and visualization, the data will also be fit to the standards that are pre-established, such as Seaborn [11], a Python data visualization library based on matplotlib [12] which provides a high-level interface for drawing informative statistical graphics.

Access to the Sleep Revolution cluster was gained by using Salamander, a centralized cluster and a server that runs and contains all valuable information in a personal environment where the information was stored in EDF files. This cluster is used within Sleep Revolution since the sleep data contains personal information. In Salamander, JupyterLab was used to test various data. JupyterLab is a web-based interactive development environment for notebooks, code, and data that allows users to configure and arrange workflows in machine learning and data science. The code for Arora is stored in the Sleep Revolution repository on GitLab and Git is also used as the version control for the package.

3.1.3 Help we Received

Whilst working on the Arora package we received advice and help from several people involved in Sleep Revolution. Sleep Revolution provided us with a rudimentary version of the package, developed by an employee who had been interested in a solution like Arora. In addition, we got a chunk of code from a student who had contributed to Sleep Revolution before. The code was for their Master's assignment which, in turn, helped the group figure out how to start coding and how to implement it using DataFrames, a two-dimensional, tabular data structure in Python, that uses labeled axes. Finally, towards the end of our project, we got extra help and examples of code from another employee at Sleep Revolution which helped us greatly with understanding and implementing ways for Arora to prepare data for machine learning algorithms. We incorporated all of the code from these different sources in one way or another into the final version of Arora which will eventually also be improved on by Arora's successors.

Chapter 4

Methods

4.1 Methods

This chapter will cover the methodology used whilst developing the Arora package. It will go over the project workflow, the development of the Python package, the user interviews the group conducted to get a clearer vision of the scope of the package, and how the product was tested.

4.1.1 Developing the Arora Package

The Arora package was developed over four months, starting in January of 2022 and ending in May, the same year. Before any coding was started, a large chunk of time went into researching various sleep researches and talking with people involved in Sleep Revolution in some way to see what they expected Arora to look like at the end of this process. With that in mind, a mind map was made where the functions and subpackages were jotted down for better visualization of the package as a whole. The code is split into eight modules, as can be seen in section 5.1.1. Being able to correctly extract and process various sleep data from different EDF files and the segmentation were the biggest challenges when it came to this package. Later on, the team was split up into different groups where the focus was either on the code, the visualization, or the testing parts of the package, based on everyone's strengths and interests.

4.1.2 User Interviews

The user interviews were carried out as an informal chat about sleep data analysis. The group talked to nine people, affiliated with Sleep Revolution in various ways, from other students working on their final assignments with Sleep Revolution to sleep, researchers and technicians. The interviews started with an introduction of the package for those who were not already familiar with it. Then all interviewees were asked what difficulties they had stumbled upon in the past whilst using the EDF data files if they had done so and what they would like for the package to be able to do to make their jobs easier and help solve those issues. It was made clear that not all functions could be executed in the span of the project but everyone was encouraged to talk about whatever they wanted without worrying if it would be possible to implement it in the time span or if it were out of the scope of the project. Since Arora is built so that more code can be implemented on top of it, section 8.1 in this report discusses the plan that goes over all of the ideas that came up in the interviews that could not be carried out for future developers to use.

4.1.3 Testing the Package

4.1.3.1 User Tests

To test the package and make sure all of the features offered were clear and easy to work with, the team brought six individuals, some of whom were part of the interviews. The group soon realized, when preparing for the testing process, that the testing would be unconventional. Normally, when testing the usability of applications, the user would be asked to complete certain tasks and the time it took for the user to complete those tasks, along with whether or not they could complete them. This method of usability testing is widely used to cater to the user experience [13]. This kind of method would prove inapt to test a Python package since there is no clear happy path when using the package. The user should be able to use any given function at any given time without needing to use another function beforehand. With that in mind, the

testing process was set up so the users were given a hypothetical scenario, such as given an EDF file, what are the steps the user would take to extract the necessary data from the file? When going through their typical workflow, the documentation of the package would be shown to the user to exhibit that the functionality to execute the workflow is included in the package. In some instances, the actual code was shown to help the users understand the functionality better.

4.1.3.2 Unit Testing

Python comes with a built-in module, unittest, for unit testing which was perfect for testing the individual units of the package. In listing 4.1, a basic example of a test file is depicted. This format was used for all testing files to ensure the tests were modular and uniform. Necessary modules and functions were imported for appropriate tests and the unit tests were built to fit the function they were testing. Assertions used were typical to check if the length of the output was correct since many functions return an array. Type assertions were used in every unit test to guarantee the package returned the correct type of data. Some tests used assertions to check that the input and output of the function did not match to reiterate that the function should have manipulated the data in some way.

```
1  import unittest
2
3
4  class MyTestCase(unittest.TestCase):
5      def test_something(self):
6          self.assertEqual(True, False)  # add assertion here
7
8      def test_something_else(self):
9          self.assertEqual(True, False)  # add assertion here
10
11
12 if _name_ == '_main_':
13     unittest.main()
```

Listing 4.1: Example of a test file

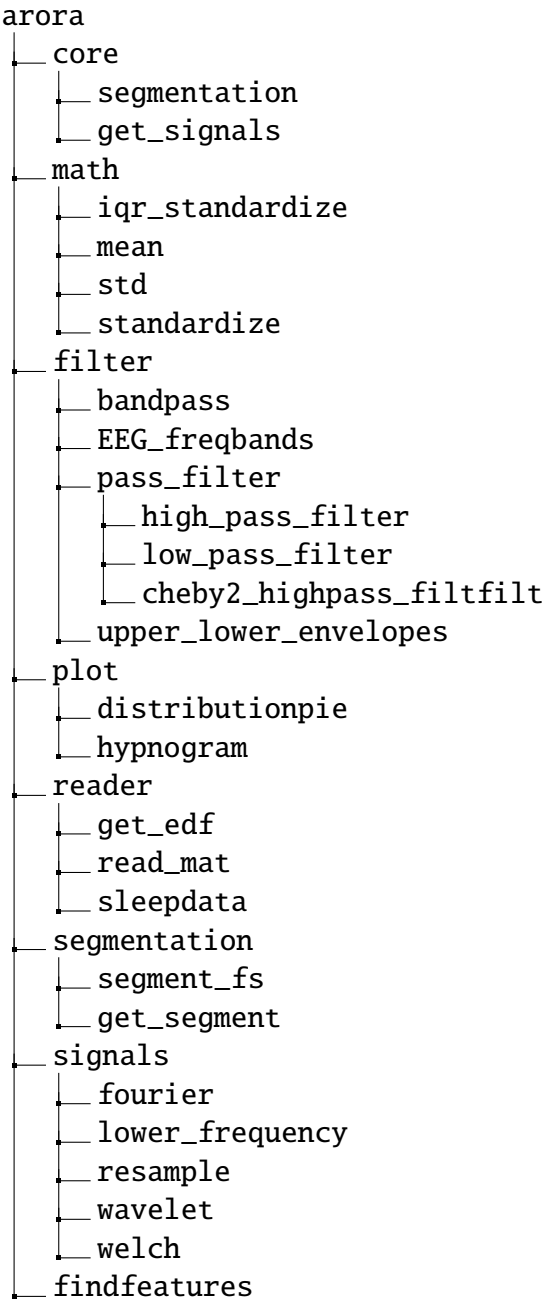
Chapter 5

Results

5.1 Results

5.1.1 The Arora Package

This section will go over how the code for Arora is set up, and the complete overview of the structure of the package is shown in the figure below.



The package, as of now, consists of the following folders; core, filter, math, plot, reader, segmentation, and signals, and each of these folders will be described briefly down below.

1. Core: The core folder includes the foundation functions that are to be used. The main functions in the core are; get signals and segmentation where the get signals function takes two parameters, an EDF file and a name of the signals that a user wants to find, and then it returns a list of all the signals that were requested. The segmentation function has five parameters; signals, a list of EEG signals, onset, the start of recording in seconds, freq, the sampling frequency of the EEG signals, duration, the end of recording in seconds, and time unit, that is the unit of time for the duration. All of the arguments are taken in and return a list of EEG signals and a pandas DataFrame that includes the beginning and end index of the sample.

2. Filter: This folder contains different filtering methods to filter through the EDF data. It is a pre-processing functionality and the files included in the folder are the following; bandpass, EEG freqbands, upper-lower signals, and filtering, all of which return the filtered data in an array.

3. Math: Math includes basic mathematical functionality of signals. The functions found in the folder are the following as of now; iqr standardize, mean val and std val.

4. Plot: Plot includes all the plotting and visualization functionality. It includes a hypnogram function and a distribution pie, both of which return charts of the signals. This module was done in collaboration with another student working on their B. Sc. thesis focused on visualizing sleep data.

5. Reader: The reader includes all functions and helper functions for reading, loading, and exporting data from different files and file types. Currently, the only type of file that the reader works with are EDF files but in Arora's future plans this will expand to other types of files such as MatLab files.

6. Segmentation: the segmentation folder includes the function get segment that segments the data of a given file. It returns the segmented data in an array that correlates to the indexes that are either pre-established or put in manually.

7. Signals: The signal folder includes pre-processing functionalities on the signal or some specific epoch. It includes the functions; fourier, segment fs, lower frequency, and welch psd, all of which return an array of the signals.

Figure 5.1 shows the package in a mind map, for easier understanding of the structure and the different modules.

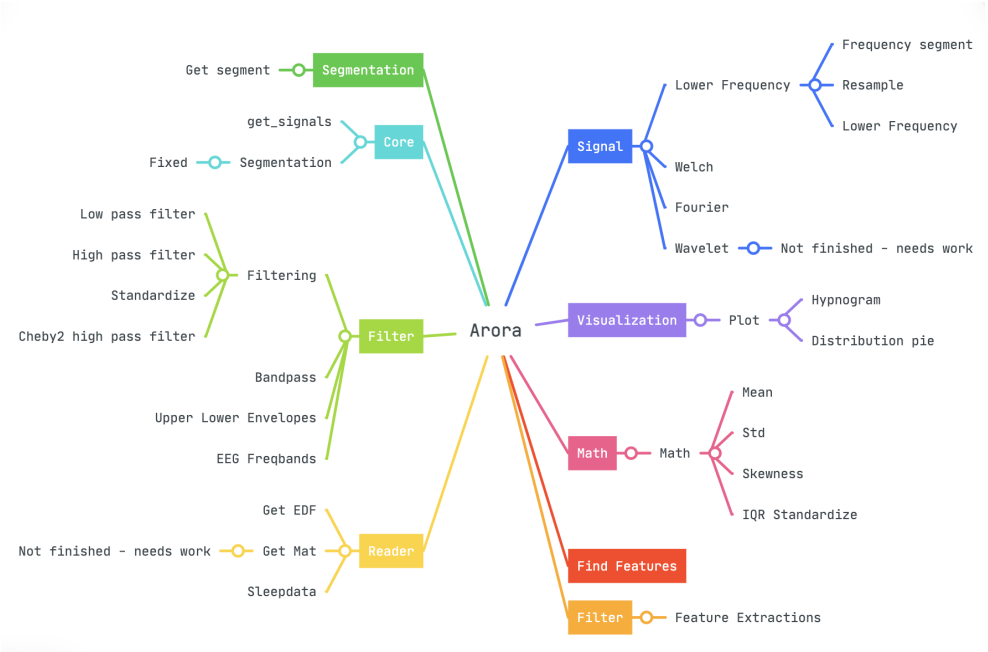


Figure 5.1: The Final Package

As previously stated, the interviewees in this phase were nine in total and each interview lasted for approximately 30 minutes. They are all connected to Sleep Revolution in one way or another and provided expert advice on what functionality the package could include based on their work in sleep research. Quality assurance of signals was a big issue that was raised but if the package had functions that could discard signals with bad quality it would remove the human error factor of manually scoring signals. Adaptive segmentation was another item that several interviewees brought up, where the package could automatically segment the data from sleep stage to sleep stage and label them correctly. Additionally, advanced filtering and visualization methods were discussed in the interviews to give the future users of the package better insight into the data. Eventually, it would be ideal for the package to be able to load and read data from different sources, such as EDF files, the Withings watches (.csv files), and MatLab files. In summarizing the user interviews, the structure of the package was rethought and in figure C.2 in Appendix C the package structure can be seen as it was imagined at this stage in the project.

5.1.2 User Testing

As mentioned before, the testers at the stage of the user testing were six in total and each evaluation took approximately 30 minutes, as the user interviews. The overall impression of the package was good. Some testers mentioned functionality they felt was missing but did not think was fundamental to have, like combining the data from multiple EDF files, reading MatLab files, and working with wavelets. The issue of adaptive segmentation and its potential was brought up again by several of the interviewees but they agreed that the fixed segmentation already implemented in the package looked beneficial. All testers agreed that the package served as a good foundation for a powerful tool in sleep research and the functionality that was implemented at the time of the testing process was similar to what they had been using individually.

5.1.3 Unit Testing

All unit tests that were made passed. Some adjustments had to be made when developing them since there were some inconsistencies based on which version of Python was running the tests but those inconveniences were minor. The coverage of the unit tests stands at 32% as this report is written which is not optimal but due to time constraints and other more crucial tasks, the unit testing could not be developed further.

5.1.4 Limitations

The development of Arora was not smooth sailing all the way through the process. Before starting the packet none of the team members had any experience in working with EDF files or sleep data and a large chunk of the time before the coding could even begin went into research. While doing the research and by talking to the instructors of the project numerous great ideas came about on which direction Arora could take, however, some of the ideas were severely underestimated in terms of difficulty and time.

Other functionalities had similar issues, sounding like something that could be done in a relatively short time in a simple manner but with more research on the subject matter, there was not enough time to accomplish all of the tasks that were set out in the beginning.

Another limitation was the EDF reader. Since EDF files are not commonly used in Python programming, there were no helper functions available to use that would make reading the data easier and the reader that was developed takes up a great deal of computational power and is therefore extremely slow. This limitation proved to be a hindrance again when developing the unit tests. The testing of functions that used the EDF reader failed in unison since the computational power was simply too great for the unit tests to pass. Therefore, all unit tests that tested functions that read files had to be cut out of the package. The solution came from another person working in Sleep Revolution, where EDF files could be changed to MatLab files that are easier to work with. In the end, this solution was not used in Arora since the data needed to test the Mat files was sent too late in the process and once again time was an issue when it came to developing the functionality.

Some of these limitations will be further discussed in section 8.1.

Chapter 6

Discussion

6.1 Discussion

The results that were discussed previously show that this product could not be tested like a typical program. Since the tests were more conversational and did not have a precise structure the comments and result accumulated could not be precisely measured. Interviewees were however excited for Arora and the process involved in the development. In partaking in the interviews, we realized that a lot of the researchers and general workers at Sleep Revolution had similar specifications for what they would want our Python package to be capable of. Since the package is an EDF reader and pre-processor, many of our participants expected the package to be able to read and load EDF files into their Python environment.

The product is not complete, with people being able to expand it later on, people showed positive reactions when shown the current package and when discussing its future. Some technical issues came up during the testing and therefore some changes had to be done to fix those bugs but all in all the user "tests" were successful.

This chapter will cover the collaboration between Arora and another final project that was also involved with the Sleep Revolution team, which was developing a DevOps program for the Arora package.

6.1.1 Working with DevOps

Arora is a final project worked for Sleep Revolution but the package is not the only assignment that was assigned over this semester. Another project was DevOps for Arora where a student worked on setting up a CI/CD and did automatic, continuous testing. The Arora team worked in closed proximities with the individual and as such had to make adjustments to be able to fit the standards used in the DevOps project.

The code structure had to undergo some changes to be able to work with the DevOps testing but all in all the collaboration went well. There were some issues when it came to communication where the Arora group had not finished the code needed for the DevOps individual to complete the task needed for progression in the DevOps part and some problems came up when testing the code with DevOps that had not come up for the Arora code before, but they were resolved and helped both teams achieve their respective tasks in the end.

Chapter 7

Conclusion

7.1 Conclusion

Sleep research is vital to understanding how sleep affects us and Sleep Revolution uses cutting-edge technology to research this phenomenon. The Arora package is an end-to-end solution that is a useful tool for sleep researchers, from reading files containing sleep data to processing said data and preparing it for further analysis. The foundation of the package was developed over a four-month period with help from experts in the field and tested in various ways. Halfway through the process, user interviews were conducted which provided useful insight into where the development of the package should head in the time frame of the project. Arora was unit tested as thoroughly as possible given the time constraint and the scope of the project and the coverage stands at 32% when this is written. The package is not finished in its entirety but a solid foundation has been built for future developers that want to help make the package the reliable sleep research tool it has the potential to become.

Chapter 8

Future Works

8.1 Future Works

Since the package has been created as a public source project and developed as such it will never be fully completed. This chapter will showcase the next steps that could be taken in the continuation of the development of Arora. It will also cover the research that has gone into finding out why these future functions were chosen, how they could be implemented, and why they would be valuable for the package.

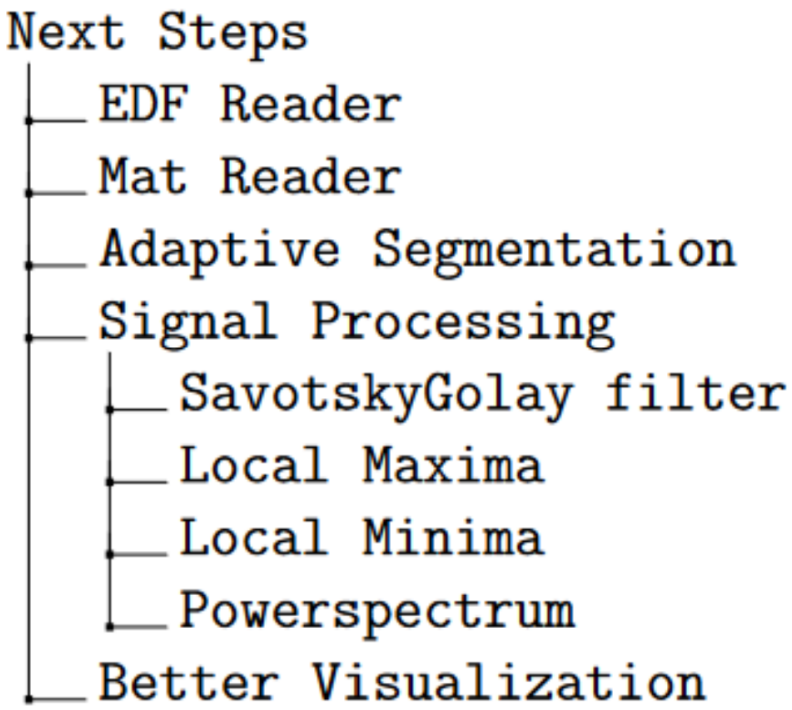


Figure 8.1: Picture of the roadmap

8.1.1 Adaptive Segmentation

In 1977 a paper titled *Adaptive segmentation of EEG records: a new approach to automatic EEG analysis* was published [2] which explained a new method in segmentation of sleep data. This new method was different from the previous way of segmenting sleep data so that instead of segmenting data based on some fixed increment, it would find sleep events and create a segment based on the changes in the events. This works by creating a moving window that is continuously compared to a set starting window and when the difference is higher than some threshold the segmentation will occur and a new starting window is chosen. This process then repeats itself until all the data has been segmented.

It was difficult to understand the process that Bodenstein and Praetorius used in their paper [2] but from what was gathered these were the steps required in implementation of adaptive segmentation:

1. Compute the signal-ACF of orders p for $s_{-N}...s_N$, where p is the filter length, s_t represents the time series data signal at time t, using the Durbin-Levinson recursive method[14].
2. Compute the PE $e_{-N}...e_N$ from the signals $s_{-N}...s_N$, where e_t is the error of the LP method for signal s_t at time t, then in accordance with $R(i) = \frac{1}{N} * \sum_{n=1}^{N-|i|} s_n s_{n+|i|}$ define a running short time ACF with $r(n;m) = \frac{1}{2N+1} * \sum_{k=-N}^{N=m} e_{n+k} e_{n+k+m}$,
3. Calculate $r(0;m)$ for $m = 0, ..., M$
4. Compute $r(n;,)$ according to the recursion method [2, Fig. 3b]
   a) $r(n;m) = r(n-1;m) + e_{n+N}e_{n+N-m} - e_{n-N-1}e_{n+m-N-1}$
   b) Define the SEM at time t $SEM_t = (\frac{r(0;0)}{r(t;0)} - 1)^2 + 2\sum_{k=1}^{M}(\frac{r(t;k)}{r(r;0)})^2$, where r(0;0) takes care of the fact that the signal may be of arbitrary power
5. Compute SEM_t and check if it is over the given threshold θ so that $SEM_t > \theta$
   a) If not then increase n by 1 and go back to step 4
   b) If it is true then a segment boundary has been detected at time n. Reset the procedure by
      i. Substituting (k - N) with (n + k) and start again with step 1

These would be the parameters as seen in table 8.1

Parameter	Description	Given value
ω	The lower cut-off frequency	1 Hz
Ω	The higher cut-off frequency	25 Hz
θ	The threshold for segmentation	0.5
N	The window length and the frequency of the signal	50
p	The filter length and the system predictor	6-8
SEM_{min}	The lowest SEM possible	0.04

Table 8.1: Parameters for adaptive segmentation

The idea that each epoch has to be 30 seconds is based on the old method where the sleep researchers used to read the data on paper and data corresponding to 30 seconds is all that fit on one sheet [15]. With the advent of computers and machine learning, this is no longer necessary and this also paves the way for a new way of implementing segmentation. By implementing this feature to the package, a more efficient method for segmentation would be added and researchers are given a new way to discern the data. This would also help in sleep staging but some epochs have multiple events happening at the same time and thus some information is lost. Furthermore, having a segmentation method that finds those events and segments them would help sleep researchers since it would show a clearer overview of what event happens at what time. It also reduces the overhead of each segmented signal as there could potentially be fewer epochs in the signal.

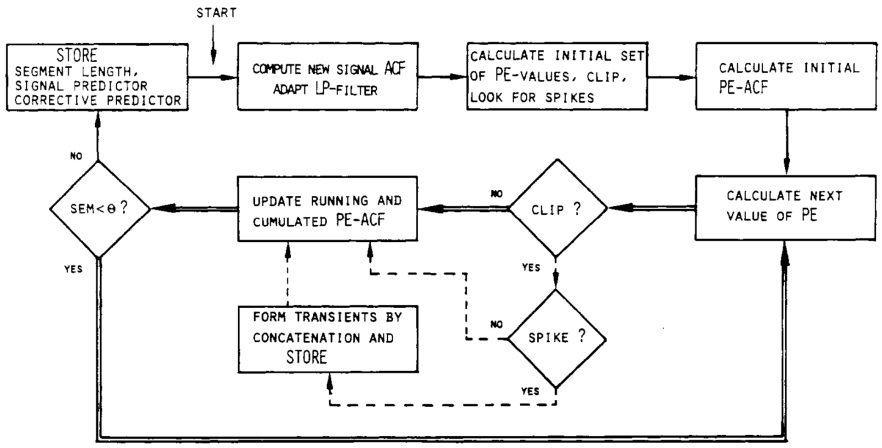


Figure 8.2: Diagram of the adaptive segmentation process [2, Fig. 5]

8.1.2 Improved EDF Reader

One problem we faced when using packages in Python that read EDF files was that they can often be unreliable. Values, such as sample frequency, would sometimes change between the EDF files. The file reader is also slow and computationally expensive and therefore some computers and clusters that were used, where all the data was located, had difficulties processing them because they could not withstand the RAM usage. For those reasons, a faster, more reliable, and inexpensive reader is needed for the future of Arora. A Ph.D. student from Sleep Revolution came up with a solution for a new, improved reader and has implemented an

EDF reader that uses Octave, Matlab files, and Python to create the EDF files. Unfortunately, due to time constraints, the Arora team was unable to add this function to the package but as the figure **??** shows, this task is the first on the list of functionalities that are to be implemented in the future. This will hopefully greatly improve the efficiency of the package and will allow users to have more control over how and what data should be returned. This also makes the overhead smaller when installing the package.

8.1.3 Reading Matlab Files

Another type of file that Sleep Revolution has been working with is Matlab files, more commonly referred to as mat files. These files can contain the same information as that of the EDF files and it could be more efficient to make a reader uses mat files instead of EDF files, as Arora does now. Mat files are far more reliable and common and are typically easier to work with so the efficiency of the reader would be better than it currently is with EDF files. This could also mean that it would be easier to debug the code when problems occur in the reader.

8.1.4 Signal processing

Signals are the data foundation of what Sleep Revolution is working with and it is important to have many unique features so that the user does not need to use multiple packages to work with the signal data. Some of the features that came to mind when designing the package, that were not implemented due to time constraints are the following;

1. One of the features that were discussed but was unable to create due to time constraints was a sort of smoothing function that would smooth the signal to remove the small oscillations that are in the signal. One common method that we found was the Savotzky-Golay filtering [16]. This filter would help with increasing the precision of the data and would not sacrifice signal tendencies since this filter removes unnecessary noise from the signal and makes it more uniform or *smoother*.

2. Another feature that would be helpful for signal processing is some kind of function that would find the local maxima and local minima of a signal. This would greatly help using the Savotzky-Golay filter as that is based on finding the local maxima of a signal [16].

3. Finding the amplitude between frequencies can also help with finding noise between signals as well as finding the sampling frequency of the signal. This can prove helpful and improve the user experience by no longer requiring the user to remember the sampling frequency of the signal. The solution to this can be creating some power-spectrum function that automates this work.

8.1.5 Testing and Improvements

When programming and creating a package it is important to have rigorous testing to see if the code that was programmed works or if there are any bugs in the code. Due to time constraints, all of the code was unable to be tested fully and features that we were able to test were not tested as extensively as would be ideal. Some functions could also not be tested due to the inefficiency of the EDF readers in Python. However, with the plans for an improved EDF reader in Arora, it would bypass this problem and allow for further testing of the functions that use the data from the EDF files.

There is no such thing as a *perfect code* when it comes to programming and with more time the code for the Arora package can be improved, and become more efficient and readable so that the next developers can continue the process already started. These improvements would also be to add more parameters to functions that have been implemented to give the users more freedom in deciding on the return values of specific functions. Lastly, some improvements could be put into existing functions to make them more efficient by removing unnecessary overhead and more validation of parameters could be put in place to streamline the process better.

8.1.6 Better Visualization

The importance of good graphs and visualization will not be overstated. Being able to accurately display data in a good and concise way benefits the user when dealing with large amounts of data as EDF files often contain. Plotly was brought up in the development process which is an interactive visualization graph that is hosted on a local server and allows the user far more freedom than visualization packages such as matplotlib. As the EDF files contain personal data a server would possibly be a bad idea but seeing as Plotly only creates a local server, available only on the developer's local environment that is not believed to pose a problem.

Chapter 9

Disclosure Statements

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 965417

Bibliography

[1] E. S. Arnardottir, A. S. Íslind, and M. Óskarsdóttir, "The future of sleep measurements: A review and perspective," *Sleep Medicine Clinics*, vol. 16, no. 3, pp. 447–464, 2021.

[2] G. Bodenstein and H. M. Praetorius, "Feature extraction from the electroencephalogram by adaptive segmentation," *Proceedings of the IEEE*, vol. 65, no. 5, pp. 642–652, 1977.

[3] G. Medic, M. Wille, and M. E. Hemels, "Short- and long-term health consequences of sleep disruption," *Nature and science of sleep*, vol. 9, pp. 151–161, 2017.

[4] S. L. Worley, "The extraordinary importance of sleep: The detrimental effects of inadequate sleep on health and public safety drive an explosion of sleep research," *P T: a peer-reviewed journal for formulary management*, vol. 43, no. 12, p. 758–763, 2018.

[5] J. W. Shepard, D. J. Buysse, A. L. Chesson, W. C. Dement, R. Goldberg, C. Guilleminault, C. D. Harris, C. Iber, E. Mignot, M. M. Mitler, K. E. Moore, B. A. Phillips, S. F. Quan, R. S. Rosenberg, T. Roth, H. S. Schmidt, M. H. Silber, J. K. Walsh, and D. P. White, "History of the development of sleep medicine in the united states," *Journal of clinical sleep medicine*, vol. 1, no. 1, p. 61–82, 2005.

[6] [Online]. Available: http://sleeprevolution.eu/en/about-sleep-revolution/

[7]

[8] T. Deboer, "Technologies of sleep research," *Cellular and Molecular Life Sciences*, vol. 64, no. 10, p. 1227–1235, 2007.

[9] E. Combrisson, R. Vallat, J.-B. Eichenlaub, C. O'Reilly, T. Lajnef, A. Guillot, P. M. Ruby, and K. Jerbi, "Sleep: An open-source python software for visualization, analysis, and staging of sleep data," *Frontiers in Neuroinformatics*, vol. 11, pp. 1662–5196, 2017.

[10] N. Coghlan, G. van Rossum, and B. Warsaw, "Pep 8 – style guide for python code," Available at https://peps.python.org/pep-0008/ (2013/08/01).

[11] M. L. Waskom, "seaborn: statistical data visualization," *Journal of Open Source Software*, vol. 6, no. 60, p. 3021, 2021. [Online]. Available: https://doi.org/10.21105/joss.03021

[12] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[13] K. Moran, "Usability testing 101," Available at https://www.nngroup.com/articles/usability-testing-101/ (2019/12/01).

[14] D. Xiao, F. Mo, Y. Zhang, M. Zhao, and L. Ma, "An extended levinson-durbin algorithm and its application in mixed excitation linear prediction," *Heliyon*, vol. 4, no. 11, p. e00948, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405844018331888

[15] L. Thomas, "History of sleep," May 2021. [Online]. Available: https://www.news-medical.net/health/History-of-Sleep.aspx

[16] R. W. Schafer, "What is a savitzky-golay filter?[lecture notes]," *IEEE Signal processing magazine*, vol. 28, no. 4, pp. 111–117, 2011.

Appendix A

Requirements

A.1 Requirements

Requirement	Importance	Guarantor
Error Handling	A	Magga
Read EDF	A	Anton
Fixed Segmentation	A	Magga
Score Vector	A	Anton
Filtering	A	Magga
Hypnogram	A	Eva
Mean	A	Anton
Std	A	Anton
High-pass filter	A	Björgvin
Low-pass filter	A	Björgvin
Bandpass	A	Björgvin
Resample	A	Eva
IQR standardize	A	Björgvin
Mat reader	B	Magga
Noise Graph	B	Eva
Wavelet	B	Eva
Powerspectrum	B	Anton
Skewness	B	Anton
Adaptive Segmentation	B	Anton/Magga
Combine EDF	B	Anton

Figure A.1: Part 1 of the requirements

A.2 User Stories

User storie no.	Title	Status
1	As a developer I want to read at least 3 sleep automation research papers to get a better understanding of the project	Finished
2	As a developer I want to set up a python project so I can get started on the package	Finished
3	As a developer I want to set up an environment on salamander to gain access to the Sleep Revolution database	Finished
4	As a developer I want to get access to the Sleep Revolution cluster so I can start working on SleePy	Finished
5	As a developer I want to set up a custom environment so I can continue to work with the data	Finished
6	As a developer I want to set up a git repository so others can get access to my code	Finished
7	As a user I want to be able to aquire basic info from the database so I can analyze it	Finished
8	As a developer I want to create a basic python package so that others can use it	Finished
9	As a developer I want to define the data format so that I can be more organazied with the data	Finished
10	As a developer I want to be able to get the program to read data from an edf file so I can analyze the information given	Finished
11	As a developer I want to implement two ways to load the data (variable and fixed) so that people can get the data that best applies	Unfinished
12	As a developer I want to implement 10 featurization methods foe sleep researchers so that they will use our package	Finished
13	As a developer I want to visualize all features so I can read the data more efficiently	Finished
14	As a developer I want to make a documentation on how to use the package so other users can understand it better	Finished
15	As a developer I want to document all the code so that others know how to use it	Finished
16	As a developer I want to make documentation on how the package works (what it takes in and what it returns) so that others will know how to use it	Finished
17	As a developer I want to implement Sleep Stage Automation so sleep researchers can use it to their advantage	Unfinished
18	As a devloper I want to implement unit tests of reading/loading data so the code runs efficiently	Finished
19	As a developer I want to implement unit tests of visualization so the code runs efficiently	Finished
20	As a developer I want to implement unit tests of feuturization so the code runs efficiently	Finished
21	As a developer I want to implement unit tests of the segmentation of the data so the code runs efficiently	Finished
22	As a developer I want to get data from an edf file ready for machine learning so ?	Finished

Figure A.2: Part 2 of the requirements

Appendix B

Risk Analysis

Description	Prevention	Response	Impact	Probability	Risk factor	Guarantor
A group member gets sick for a long period of time	Have a method of communication and planning so that if one of us is sick for a long time they can still work from home.	Other team members pick up the work of the sick team member.	2	2	4	Anton
Every group member gets Covid	Have a clear sprint schedule to reduce likelihood of backlog overload.	Stragetically manage necessary tasks until we have more time to work on less important tasks	5	1	5	Eva
The program processes the wrong information	Test all new code before we add it to the main code.	Test functions and fix any mistakes.	4	2	8	Björgvin
Salamander is unavailable	Keep all our code backed up in a seperate space.	Work on our code in the seperate space until salamander is back up and running.	5	1	5	Margrét Sól
Functions return the wrong value	Document and test each new function before adding it to the general codebase.	Test functions and fix any mistakes.	3	2	6	Björgvin
Problems with Sleep Revolution, little time to get help	Use team meetings (friday meetings with Sleep Revolution) to go over any questions and issues we might have.	Focus on the work that we are able to do without Sleep Revolution's help	3	1	3	Anton
Program returns the wrong information	Test all new code before we add it to the main code.	Test functions and fix any mistakes.	5	4	20	Margrét Sól
Computers shut down	Save all work immediately, preferably automatically.	Try to get them to turn on again. If that does not work find new computers to work on	2	2	4	Margrét Sól
The scope of the project is too large	Talk to the project owners to create an appropriate scope for this project	Talk to the product owners to lessen the scope of the project.	4	2	8	Anton
Team member unreachable	Good communication between members and tell others in the group when something comes up	Try to reach the member on all platforms, otherwise members try to do their part	2	5	10	Eva
A member of the team or the whole team are not able to access the database	Have an understanding of what the data looks like and how we can work with it. So that theoretically if we didn't have access to the database we could still add functionality to the code.	Talk to Sleep Revolution and focus on inner functionality until we get access to the database again.	5	1	5	Anton
Version control system fails	Establish clear rules on readability so that even without version control everyone is aware of the standard of code within the project.	Allot time later to go over any new code with the linter after it is back up and running	5	1	5	Eva
The project doesn't run	Test all new code before we add it to the main code.	Test functions and fix any mistakes.	5	3	15	Björgvin

Figure B.1: Risk Analysis for Arora

Appendix C

Mind Maps

C.1 The Initial Package

Figure C.1: Initial Mind Map

C.2 The Dream Package

Figure C.2: The Dream Package

Appendix D

Burndown chart

The following chart shows our real time performance vs. our planned overall time estimated for this project.

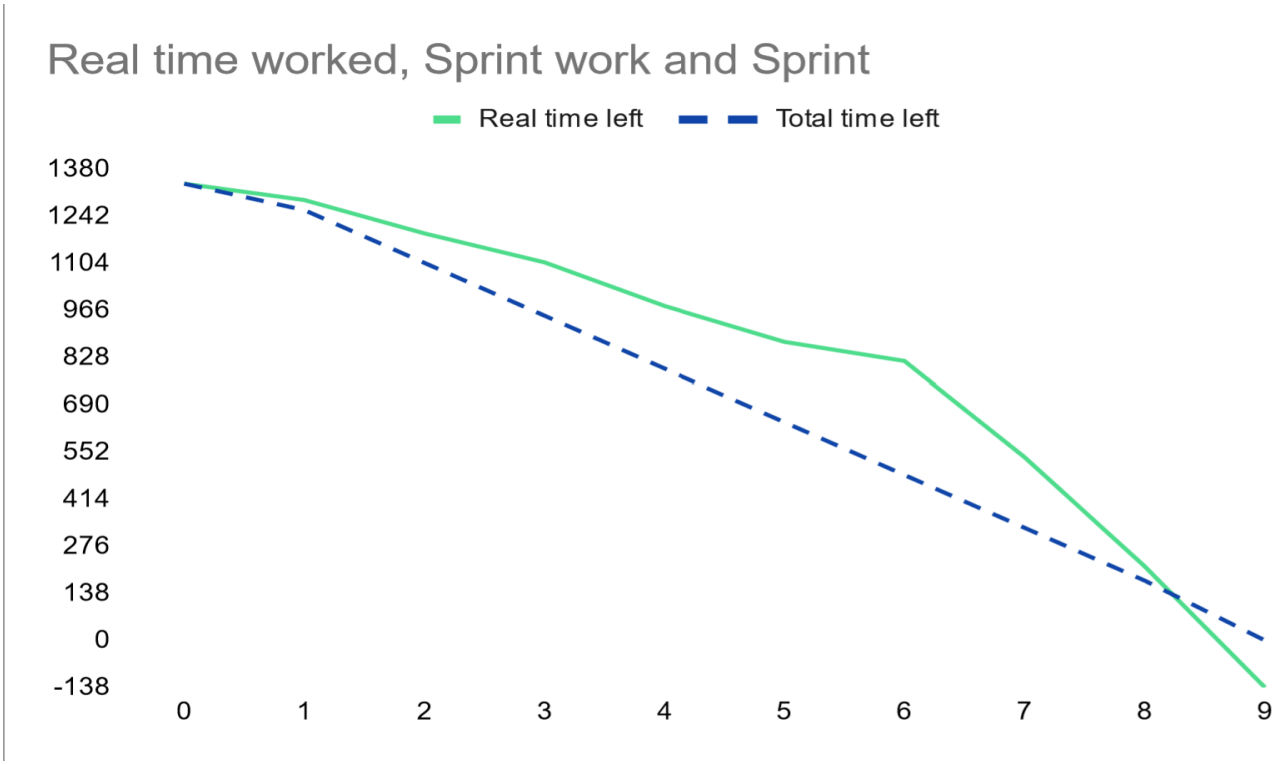


Figure D.1: Burndown chart for the total time of the project