BACHELOR THESIS

ARTIFICIAL INTELLIGENCE

Radboud University 

Personalizing Digital Cognitive Behavioral Therapy for Insomnia Through Automation of Micro-Intervention Selection Based on Data from Wearable Technology; A Case Study on the Oura Ring

Author:
Sjors Weggeman
S4799771
sjors.weggeman@student.ru.nl

First supervisor:
dr. M. VASTENBURG
Social Sciences/BSI
martijn.vastenburg@ru.nl

Second supervisor:
dr. S. THILL
Artificial Intelligence
serge.thill@donders.ru.nl

June 16, 2021

Abstract

Because of the low supply and high demand for the treatment of insomnia, researchers have decided to provide Cognitive Behavioral Therapy through digital means because it has the potential to reach many people. However, due to the large intended userbase it is very hard to personalize the treatment program, meaning that patients have to follow the long, full program. This leads to low adherence rates and therefore less efficacy. One way to increase personalization in these programs on such a scale, is through the integration of wearables. Currently this data is being used as an automated substitution of the sleep diary, even though this data can be used to reach a much higher level of personalization. For example by providing short, directly actionable interventions, from the superset of Cognitive Behavioral Therapy for Insomnia, to the user, that specifically target the sleep problems of the user. This is a much more effective way of helping patients.

Contents

1 Introduction

Extensive research has shown that our sleep is directly related to our health (Imeri & Opp, 2009; Walker, 2017). A reduction in either the quality or the duration of our sleep is detrimental for both our physical- and our mental wellbeing (Fortier-Brochu et al., 2017; Espie et al., 2019). And even though everyone experiences a night of bad sleep now and then, for some people this occurs on multiple nights within a short period of time. The name for this problem is: `Insomnia'.

A common consequence of Insomnia is that the person suffering from it feels unrefreshed after the sleep and is possibly even impaired in their daytime functioning. Examples of such impairments include, but are not limited to: feeling tired, being unable to focus and being more irritable than usual (American Psychiatric Association, 2013).

Global trends show that currently, about ten to twenty percent of all adults worldwide experience chronic insomnia (Bhaskar et al., 2016, Liu et al., 2016; Kronholm et al., 2016) and about thirty-three percent of all adults worldwide experience occasional insomnia (van der Zweerde et al., 2019). The current COVID-19 pandemic and its consequences have only made this worse: People experienced more stress, lost their daily routine, and lost the social control that maintained it (Cellinen et al., 2020; Partinen et al., 2021).

Our well-being is in turn related to the rest of society, since tired or ill people cannot contribute as much to society as before. On the contrary, their treatment costs a lot of time and money, while in the meantime their functions are not being executed as good as they were, or maybe not even at all. This ramps up the total costs to up to ten times the costs of healthy people (Daley et al., 2009; Kessler et al., 2011). Therefore, it is in the best interest of everyone that sleeping problems are treated as soon and as fast as possible.

The golden standard treatment for insomnia is Cognitive Behavioural Therapy for insomnia (CBT-I) and while it works well, there is a large discrepancy between supply and demand. To resolve this discrepancy, researchers have been looking for a scalable way of providing CBT-I, such that they can provide it to everyone who needs it.

Since more than half of the entire population on earth has access to smartphones and internet (BroadbandSearch, 2021), the most obvious solution was to go digital. Accordingly, there are a variety of web and mobile applications now, providing digital CBT-I (dCBT-I).

The problem with dCBT-I however, is that it is extremely hard to personalize the therapy because these applications have a very large intended userbase. A consequence of this lack of personalization is that all users must work through the entire CBT-I program, while normally a therapist would be able to guide the patient into certain directions based on some questions. This takes much time and effort. Because, on average, humans are impatient of nature they tend to give up when they do not see results quickly (Xu et al., 2020), leading to low adherence rates when not repeatedly reminded of the progress they have made so far (Mohr et al., 2011).

Accordingly, some of the better applications have been trying to implement ways to personalize the CBT-I program, allowing for a more direct application of the information by limiting the treatment to relevant parts: micro-interventions. All apps that implemented this personalization used questionnaires, like has been done since the early days of CBT-I (Espie et al., 1989). These questionnaires need to be filled-in upon installing the application. Since this is considered burdensome by some people, developers went a step further by implementing wearable integration (Luik et al., 2019) in order to automate this process of filling in a sleep diary (Baron et al, 2021). Even though this is useful for some people (Luik et al., 2018; Baron et al., 2021), this data can be used to gain a lot more (Baron et al., 2021).

The wearable of choice for this study is the Oura-Ring. The data collected by the Oura ring consists of sleep and activity data such as movement, temperature and heart rate. Usually this data is collected through polysomnography, the golden standard method to diagnose sleep disorders. The wearable that performs the best when compared to polysomnography is the Oura ring (Kinnunen, 2016; Kang et al., 2017; de Zambotti et al., 2019; Moreno-Pino et al., 2019). Hence the research question: Can sleep and activity data collected through the Oura ring, be used to determine a selection of micro-interventions from the superset of CBT-I, that targets all the sleep-related problems indicated by the user?

I hypothesize that the micro-intervention framework makes it possible to create a collection of individually administrable, directly actionable interventions that target exactly those sleep-related problems of which the user indicates that they apply to his life, without any human being part of the selection process.

This makes the large amount of information on the treatment of insomnia much more accessible to a large group of patients by reducing the time needed to work through the information to a minimum. This directly translates to a reduction in time until results are achieved, which is an improvement of the treatment. That, combined with an increased engagement of the patients (Luik et al., 2018), might yield an increase in adherence rates (Baron et al., 2021), which would also be an improvement of the treatment. Lastly, this solution would greatly help reduce the number of insomnia patients, opening space in the schedules of trained CBT-I experts for those who need even more individualized therapeutic guidance. All this should be an incentive for further research into the integration of wearable technology into dCBT-I.

In the following chapters will be clarified if and how data collected through the Oura ring can be used to automatize the selection of personal micro-interventions of dCBT-I. First, in chapter 2, some prior knowledge will be provided as well as some frequently recurring abbreviations and their meanings. Then, in chapter 3, the related work will be discussed as well as how this research relates to it. In chapter 4 an explanation will be provided on how the problem could be tackled. The results of these methods will be provided in chapter 5 and the meaning of these findings will be discussed in chapter 6. In chapter 7 the conclusion about the hypothesis will be drawn and in chapter 8 and 9 statements will be made about the compliance to the ethical standards and the funding of this research. After that a list of references is included in chapter 10, followed by the appendices.

2 Preliminaries

2.1 Common Abbreviations

CBT - Cognitive Behavioral Therapy

CBT-I - Cognitive Behavioral Therapy for Insomnia

dCBT-I - digital Cognitive Behavioral Therapy for Insomnia

2.2 Definition of Insomnia by the DSM-5

According to the fifth edition of the Diagnostic and Statistical Manual of Mental Disorders, insomnia is characterized by one or more of the following symptoms: difficulty initiating sleep, difficulty maintaining sleep and early morning awakenings (American Psychiatric Association, 2013).

2.3 Types of insomnia

There are two types of insomnia: short-term and long-term. The short-term category comprises both transient and acute insomnia, whereas the long-term only covers chronic insomnia. Transient insomnia lasts a few days to a week, acute insomnia between a week and three months, and chronic insomnia longer than three months (American Psychiatric Association, 2013).

2.4 Wearable

“Wearable technology consists of things that can be worn, such as clothing or glasses, that contain computer technology or can connect to the internet[.]” (Cambridge University Press, n.d.)

2.5 Justification of the Oura ring

The Strategist, a network created by Vox Media that claims to be “an obsessively researched online shopping and product discovery guide” (Vox Media, n.d.), interviewed seven experts about the Oura ring. According to these experts, the major advantage the Oura ring has over other wearables is its size. Most wearables are quite large and hence people tend to be aware of wearing them. This makes it a burden to wear these wearables at night, a problem that the Oura ring does not have.

This advantage, combined with its great performance compared to other wearables (Kang et al., 2017; Moreno-Pino et al., 2019) and the golden standard way of determining whether someone has a sleep disorder, polysomnography (Kinnunen, 2016; de Zambotti et al., 2019), make it an interesting product for medical applications (Cheslaw, 2020).

2.6 Definition intervention Cambridge Dictionary

"The action of becoming intentionally involved in a difficult situation, in order to improve it or prevent it from getting worse[.]" (Cambridge University Press, n.d.)

2.7 Micro-interventions

The concept of micro-interventions was first introduced by Elefant et al. (2017). The idea behind this concept is that the patient can be provided a short, very specific intervention for a problem that can directly be applied or executed. This is intended to deliver fast results quickly, which in turn should increase adherence to the treatment.

3 Related Work

The idea of using CBT as an alternative for the pharmacologic treatment of insomnia has been around since the 1990's (Morin et al., 1999; Luik et al., 2019) and was administered through digital means for the first time in 2004 (Ström et al., 2004). Almost a decade later, there still had been conducted little to no additional research on this topic (Espie et al., 2012; Espie et al., 2013). Even though, halfway through that decade it was already known that there was a large discrepancy between the demand and supply of CBT-I (Espie, 2009). In order to resolve that discrepancy, they needed to scale up the treatment. However, it was also known that CBT-I was not scalable in its current form. Digitalization was the most logical and most promising way to scale CBT-I treatment (Espie, 2009).

It was at the end of this decade, in 2013, that the usage of wearable technological devices was first mentioned in the context of dCBT-I (Espie et al., 2013), hinting at the opportunities for future research. Yet it would take four more years before other researchers picked up on this. During this time the efficacy of dCBT-I would be proven for various treatment conditions (Thorndike et al., 2013; Zachariae et al., 2015; Luik et al., 2017, Espie et al., 2019), as well as the fact that insomnia often not only occurs together with other psychological disorders (Budhiraja et al., 2011; Thorndike et al., 2013), but can also cause them (Pigeon et al., 2017). This is an extra incentive for the treatment of insomnia. Furthermore, CBT became the golden standard for the treatment of insomnia (Qaseem et al., 2016; Riemann et al., 2017; Riemann et al., 2017), which produced a sudden boost in the amount of research in the area of CBT-I, including dCBT-I.

Questions about dCBT-I that remained unanswered during this time, however, were mostly concerning the optimal dosage of treatment (Espie et al., 2013; van der Zweerde at al., 2019), the best way to implement dCBT-I (Luik et al., 2017; van der Zweerde et al., 2019) and what the interpersonal differences are that affect treatment (Blanken et al., 2019; van der Zweerde et al., 2019). Some examples of what a dose might consist of are “the number of elements in the treatment, the number of sessions, [and] the amount of personal tailoring of treatment” (Espie et al., 2013).

Since the efficacy had already been proven and wearables were increasingly more capable and popular, the most logical next step was attempting to utilize the opportunities from the integration of wearables. So that is what researchers started looking into. Opportunities they found were the possibilities of: providing actionable interventions (Bianchi, 2017), personalizing the treatment (Luik et al., 2018; Jaiswal et al., 2019; Perez-Pozuelo et al., 2020; Baron et al., 2021), improving interventions, measuring sleep data population-wide (Jaiswal et al., 2019; Perez-Pozuelo et al. 2020), increasing adherence rates (Cajita et al., 2020), engaging patients more actively, improving communication between the patient and the CBT-I provider ánd improving the accessibility of CBT-I (Baron et al., 2021). These are opportunities that most related studies did not capitalize upon just yet (Baron et al., 2021).

The first opportunity: providing actionable interventions, has been researched in an area closely related to that of dCBT-I, namely in the treatment of depression. Depression is a mental disorder that often co-occurs with insomnia. The first-line treatment for depression is also CBT. Also, in this area the problems of scalability and adherence occur. Because CBT is very straightforward, people can easily apply the intervention themselves (Schröder et al., 2016), but since CBT is a combination of multiple therapies, the process of finding the right intervention costs a lot of effort and is very time consuming. This can be solved by splitting the CBT into smaller, directly actionable interventions: micro-interventions (Elefant et al., 2017). Since these micro-interventions take less time and provide faster results, people tend to adhere better to the treatment, partially solving that problem as well. Since the concept of micro-interventions seems to work well (Ayers et al., 2015; Bunge et al., 2016; Lokman et al., 2017) and are based on CBT, just like the golden standard treatment for insomnia, it seems that the use of micro-interventions in the treatment of insomnia is a logical next step.

As said before: there are still questions that remain unanswered within this area of research and the integration of wearable technology into dCBT-I programs brings a lot of opportunities that have not been capitalized on yet. Trying to capitalize on these opportunities might just help us get closer to finding the answer to these questions.

4 Methods

Research Question

Can sleep and activity data collected through the Oura ring be used to determine a selection of micro-interventions from the superset of CBT-I, that targets all the sleep-related problems indicated by the user?

In order to answer the research question, an overview of existing CBT-I interventions needs to be charted based on a literature search, and needs to be combined into a framework by a meta-synthesis of this literature. With the help of Jupyter Notebook, collected data will be analyzed with the purpose of determining for which micro-interventions from this framework it is possible to make an automated selection, using the data. The actual selection will then be determined based on relevance for the user. Therefore, the selection must be validated.

In order to improve clarity, each subsection of the following chapters will be divided into the following three categories.

4.1 Framework

The first step in this process is scouring existing literature and translating the findings into a framework through a meta-synthesis.

The literature search will be conducted covering several scientific literature databases as well as grey literature. Multiple databases have been used to prevent publication bias. For this research I used Google Scholar, PubMed and RUQuest, the latter of which is the online search engine of the Radboud University Nijmegen for scientific research.

During my preliminary research I used the same research databases and found that certain names occurred repeatedly: Morin, C.M., Espie, C.A., Luik, A.I. and Ritterband, L.M. Googling these names together with the term ‘researcher’ or ‘insomnia’ resulted in the confirmation that all these people are experts in the research area of sleep. Accordingly, their research provides a solid ground for further research into this topic.

Since this part of the research concerns Cognitive Behavioral Therapy for Insomnia only, that term will be the first search term. In this search I will then only look for papers authored or co-authored by these experts.

The time frame of this search will be from 1990 (Morin et al., 1999; Luik et al., 2019) until the present. Because this part of the research is conducted in March 2021, it might be incomplete. Important to mention is that the idea of using behavioral interventions to treat insomnia ranges further back, but only the combination of both cognitive and behavioral interventions are relevant for this research, hence literature that only mentions behavioral interventions is excluded.

If components of CBT-I interventions have been found but are not as small as possible yet, resulting in individually administrable, directly actionable interventions, then a secondary literature search is performed using the title of the smallest known component, combined with the word ‘component’, as search term. This process is repeated until micro-interventions have been reached for each branch.

All found components and subsidiary components are then combined into a tree structure.

4.1.1 Limited time

Because this research is part of a Bachelor thesis, there is a limited amount of time I can spend on this part, hence I decided to take the first ten entries of the literature I can find in the primary search. The components and subsidiary components shall then be extracted. Then for each component or subsidiary component which did not result in a micro-intervention yet, up to three pieces of literature are taken into account in the additional searches.

4.2 Data collection and analysis

The second step in this process is the collection and analysis of sleep data and sleep-related data collected using a wearable.

In order to automate the selection of interventions from the framework, data is needed from which can be determined whether a user is suffering from sleep problems and if so, which sleep problems. The wearable chosen for this study is the Oura ring, the justification of which can be found in the preliminary section (Chapter 2).

4.2.1 Data acquisition

For this research three subjects who owned and used an Oura ring were recruited through primary and secondary connections within my personal network. These subjects voluntarily provided the raw datasets collected using their Oura ring. They downloaded the raw data from the Oura cloud (Oura on the web, n.d.). This data was anonymized upon reception. Exclusion criteria were datasets smaller than a month, because the Oura ring has an adjustment period during which it calibrates itself to the user, finding reasonable baselines for measurements (Get started with Oura, n.d.). Because it has not been specified how long this adjustment period is, in this study we will assume that it is two weeks at most. Furthermore, because this is a general feasibility study, gender and age were not included in the exclusion criteria. Based on these criteria, none of the datasets have been excluded. All users gave their informed consent for the usage of their data, given that it would be anonymized and would be stored and treated in line with the General Data Protection Regulations (European Parliament and the Council of 27 April 2016, 2016)

4.2.2 Data analysis

The program chosen for the analysis of the data is Jupyter Notebook. This program has been chosen because it can run Python code, which is simple and versatile (Millman & Aivazis, 2011), and it enables me to structure the code in a readable fashion next to regular text descriptions.

The data analysis was performed on one dataset and the resulting functions were tested on the two other datasets.

As a first step in the data analysis the types of data measured by the Oura ring were explored, including when they are measured, and in which format they are stored. Most of this information is contained in the Oura API documentation for developers (Ōura Health Oy, n.d.), but in order to create a double dissociation this information was compared against the actual data.

In order to be able to visually inspect the data some preprocessing was necessary, because initially each value is stored on a separate line, which made it impossible to see the bigger picture. Therefore, the data needed to be rearranged into a format that does allow that.

4.3 Validation

The third step in this process is checking whether the automated micro-intervention selection consists of micro-interventions that are relevant to the subject.

A Google forms questionnaire was provided to the participants with fourteen questions after the data had been analyzed. This questionnaire can be found in Appendix A. The questionnaire is partially inspired by the Pittsburgh' Sleep Quality Index (PSQI) questionnaire (Buysse et al., 1989), which can be found in Appendix B. The reason for using Google forms is that it is free to use and is easily accessible for both the researcher and the participants on both stationary and mobile devices.

The questions in the questionnaire were either related to the subjects' personal data or to their sleep related behavior: The first question was used to link the results to the right dataset, whilst maintaining the subjects' anonymity. The second and third questions were about gender and age and were included for the purpose of determining generalizability of the results of this study. The fourth and fifth questions were used to determine the validity of the datasets in general. The next four questions have directly been taken from the PSQI questionnaire to gain insight into the subjects' perception of the problems. The four questions after that have been inspired by the PSQI and are used to validate the provided advice. The last question concerns the consent of the user to use the dataset provided as well as the answers given in the questionnaire.

5 Results

5.1 Framework

The details of the literature found in the literature search using the search term: 'Cognitive Behavioral Therapy for Insomnia' are presented in Table 1, as well as which interventions they mentioned to be a part of CBT-I. This table can be found in appendix C. Out of the ten entries only eight mentioned components. The details of the literature found in the second part of the literature search can be found in Table 2 in appendix D. These were three sources which have been found using the names where more information was necessary as search strings, combined with the term: 'components'.

A thorough meta-synthesis of the findings denoted in Table 1 and Table 2 has been combined into a tree-structured framework of forty-eight micro-interventions which can be found in Appendix E. Snapshots of certain aspects of the framework have been included as well in Appendices E - 1 to E - 3 to improve readability. The justification for the chosen subdivision has been provided below.

5.1.1 The main subdivision

In this framework the distinction has been made between the four most seen main interventions that CBT-I consists of, namely: Stimulus control, sleep restriction therapy, education about sleep, and cognitive therapy.

5.1.2 The first branch

Since the first part of treatment always consists of educating the patient and their families about the condition of the patient, the educational part comes first. In the treatment of mental illnesses and disorders this part is called psychoeducation. Concerning subsidiary components a lot of aspects are mentioned, but no concrete consensus can be found, therefore more research was necessary. The details of this additional research can be found in Table 2 in Appendix D.

5.1.3 The second branch

According to Morin & Espie (2003), the first CBT-I interventions that should be tried in a treatment program are sleep hygiene and relaxation therapy. In the literature however, there are many different perspectives on the subdivision of sleep hygiene. Take for example the paper by Lacks & Morin (1992). In this paper sleep hygiene is considered to be a part of lifestyle, whilst in the papers of Morin et al. and Espie from 1993, sleep hygiene only comprises substance intake, exercise and the environment. In order to unify the different perspectives of all papers I have decided to make the second branch a combination of sleep hygiene and lifestyle factors.

5.1.3.1 The first split

The subdivision of which subsidiary components or micro-interventions belong to which branch has been determined using logic. Lifestyle is clearly about how an individual lives his or her life. Aspects related to this are the amount of exercise they get and what they ingest such as caffeine, alcohol, tobacco, drugs, liquids, and diet, and in which quantities they ingest these things. These are all small components that yield short, directly actionable interventions, so for this branch the micro-interventions have been reached.

5.1.3.2 The second split

Most of these things exert influence on the sleep quality in a matter of hours, whilst for the sleep hygiene parts this is mostly in a matter of minutes or seconds. Therefore sleep hygiene consists of aspects that have an immediate effect on the sleep quality, such as the sleep schedule, the activities performed in and around the sleeping environment, as well as certain aspects of the sleep environment itself. According to the literature, the first two of these are generally considered to be stimulus control, hence the next subdivision was easy to make.

Aspects of the sleep environment found in the literature are related to comfortability, air quality, noise levels, light levels, room temperature and body temperature. Each of these aspects are micro-interventions.

5.1.3.2 The third split

As mentioned in the previous section, both the sleep schedule and the activities performed in and around the sleeping environment are considered to be stimulus control, so these two were the next branches.

The division of subsidiary components from stimulus control could be made based on whether the interventions directly target sleep or rather the activities related to or performed around sleep. For the most part, this subdivision was quite easy, but there was one exception: In the literature ‘getting out of bed when unable to sleep’ is often combined with ‘only go to bed when sleepy’, but these are in essence two micro-interventions that can easily be provided separately. This has the positive side-effect that the intervention can also be provided when someone is going to bed because it is night, even though they might not be sleepy yet. On the downside however, ‘only go to bed when sleepy’ fits in both the sleep schedule and the sleep activities category. I decided to place it in the sleep schedule category, because it is more directly related to sleep than to a specific action that can be performed.

Interventions that directly target sleep are: going to bed only when sleepy, rising at the same time every day, and not napping during the day. These are all micro-interventions.

All the other parts that do not directly target sleep, but only the activities revolving around sleep or related to sleep are: Only using the bed for sleeping or sex, getting out of bed when unable to sleep, relaxing before sleep and no worrying whilst in bed or in the bedroom.

Most of these things are already micro-interventions, except for relaxing before sleep. There are many ways in which a person can try to relax, some of which work good for one person, but not for another. Therefore, additional research had to be done for this branch too.

The found relaxation techniques were: Breath focus, body scan, guided imagery, mindfulness meditation, yoga, tai chi, and qigong, and repetitive prayer. Some of these aligned with what was already mentioned such as deep breathing. Others that were mentioned were taking regular breaks or walking through nature.

5.1.4 The third branch

The third branch is sleep restriction therapy. According to some of the literature this is a part of stimulus control or even more specific: of the sleep scheduling branch. Most literature name it as a separate intervention, which is why it became a separate branch. In the end it does not matter for administering the micro-interventions, because a patient can still receive all of them. It is just not very likely that a person needs to receive every micro-intervention.

5.1.5 The fourth branch

In the literature found during the primary search, one thing became clear about another branch, namely that it should start with: Cognitive. What should come after it is less clear: ‘Cognitive therapy’, ‘cognitive restructuring’, and ‘cognitive methods’ are just some examples of names that the literature provided for this branch of interventions. A consequence of this uncertainty overall is that hardly any article mentioned clear and concise subsidiary components. Therefore I decided to perform secondary research on this intervention.

Since many of the more recent articles mentioned cognitive restructuring instead of cognitive therapy, the name of the branch became cognitive restructuring. In additional research to the components of cognitive restructuring I found that Clark wrote a whole chapter about it in the Wiley Handbook of Cognitive Behavioral Therapy (Clarke, 2013). He named three core components: Collaborative empiricism, verbal interventions, and empirical hypothesis testing. The verbal interventions are: Evidence gathering, consequential analysis, cognitive bias identification, generate alternative, normalization, decatastrophizing, problem solving, imaginal exposure, distancing, reframing or perspective taking, reattribution, and positivity reorientation. Clarke mentioned that all three core components should be present in a cognitive restructuring intervention, but he also wrote that it would be some other type of intervention if not administered all together. This means that all these parts are micro-interventions. Paradoxical intention has been added to this list because it was often mentioned in the literature under the cognitive branch.

5.1.6 The fifth branch

The fifth branch is the biofeedback node in the centre. Biofeedback has been mentioned in the literature as a relaxation method, but also as a separate intervention. Since this project uses the data collected using a wearable in order to provide micro-interventions, which in essence is biofeedback, hence I have placed it in the middle between the sleepy hygiene and lifestyle education branch and the sleep restriction therapy branch, because these are the branches that contain the most (possibly) automatable micro-interventions.

5.2 Data collection and analysis

5.2.1 Data acquisition

Three datasets were collected from three male subjects with ages ranging between 23 and 48. The first dataset comprised 585 entries in the activity data, 2 in the restful periods data and 571 in the sleep data. This dataset was used to analyze most of the data and was used to create the automated micro-intervention selection.

The second dataset comprised 46 entries in the activity data, 2 in the restful periods data and 44 in the sleep data and the third dataset comprised 49 entries in the activity data, 26 in the restful periods data and 44 in the sleep data. These two datasets were used to test the automated micro-intervention selection and served as additional information to determine one of the automations, because a part of the data was deemed unreliable. More information about this will be provided in section 5.2.2. Lastly, the results of the automated selection on these two datasets have been used to validate the selection. More information about that will be provided in section 5.2.3.

5.2.2 Data analysis

In the following section the most important results of the data analysis will be highlighted. For the full details, please see Appendix F, the code has been highly documented and extensive explanations have been provided in between the code blocks.

After the first preprocessing it could be determined that most of the data corresponded to what the developer's documentation described. A small discrepancy was found however: According to the 'Data types' section in the documentation (Ōura Health Oy, n.d.), all durations are expressed in seconds. In the 'Activity' part of the documentation however, durations are expressed in minutes. Because the least accurate variable determines the accuracy when performing calculations, my answers are going to get as precise as minutes.

5.2.2.1 Preprocessing for analysis

After this primary preprocessing for visual inspection, I could start preprocessing the data for the analysis.

There were two time variables that were represented in a standard ISO 8601 notation, which made it impossible to perform calculations or comparisons to other variables. Therefore, these two variables had to be converted into workable time formats that could be compared: Minutes. Furthermore, there were dates in the data. A function to compare dates and calculate the difference between these dates existed already, but the dates had to be converted into another format, so I wrote a function for that too.

Using the conversion and comparison function I was able to compare the first and last date in certain sections of the data to check whether there were gaps in the data. For the dataset I was working on, this was the case. The logical next step was to determine where these gaps were located and whether they were contained in the part of the data that I was going to work with. In order to determine which part of the data I was going to work with I

performed a small literature search.

Normally, in order to diagnose a sleep disorder, sleep experts ask their patients to fill in a sleep diary. There are plenty of sleep diary formats and instructions on how to use them online, but they differ in lengths. Some formats and instructions denote one week as minimum, others two. In order to determine which one is the most trustworthy I looked up the impact factor of the sources of some formats, finding that the journal of the American Academy of Sleep Medicine was the most trustworthy with an impact factor of 3.586 and a five year impact of 3.952 in 2020 (Journal of Clinical Sleep Medicine, 2020). Accordingly, I decided to use the sleep diary format they had put online, which has a duration of two weeks. (American Academy of Sleep Medicine, 2017).

No gaps were found within the last two weeks of the data, meaning that any advice based upon this data would be representative of the subject's current sleeping pattern.

Lastly, since the data was longer than one and a half years ánd there had just been a daylight-saving-time switching date, I decided to check for patterns in the data based on daylight-saving-time switching dates, because it might have been of influence on the subject's current sleeping pattern. No pattern was found based on daylight-saving-time however.

5.2.2.2 The actual analysis

In the analysis I found that it was possible to automate six micro interventions from the framework I created, using the data and the guidelines for the interventions found in the literature. The automated micro-interventions are: Rise at the same time every day, no napping, get out of bed when unable to sleep, exercise regularly, reduce time in bed to hours slept, and increment that time by 15 to 30 minutes.

When implementing the function that detects whether the user went out of bed when awake for a prolonged period of time, I found that the categorization of the activity levels provided in the Oura API documentation did not correspond to the data: According to the documentation the medium activity level corresponded to walking, but when checking the activity spectra of days where the subject walked more than seven-thousand steps, the medium activity level was hardly ever reached. This raised a question about the accuracy of the categorization provided in the documentation. The boundaries of the medium activity level are age dependent, so that might have been the cause. Therefore I decided to only provide advice after inspecting that part of the data in the datasets of the two other subjects.

The data of the second subject seemed to support my earlier assumption about the accuracy of the categorization, because on the day where he walked the most, the activity level did not nearly reach the medium level as much as would be expected on a day with that many steps, as can be seen in figure 1. (This data can be found in detail in Appendix G). The data of the third subject was more accurate however, as can be seen in figure 2. (This data can be found in detail in appendix H.) This means that there are probably other factors that play a role in the accuracy of this categorization, but more about that will follow in the discussion.

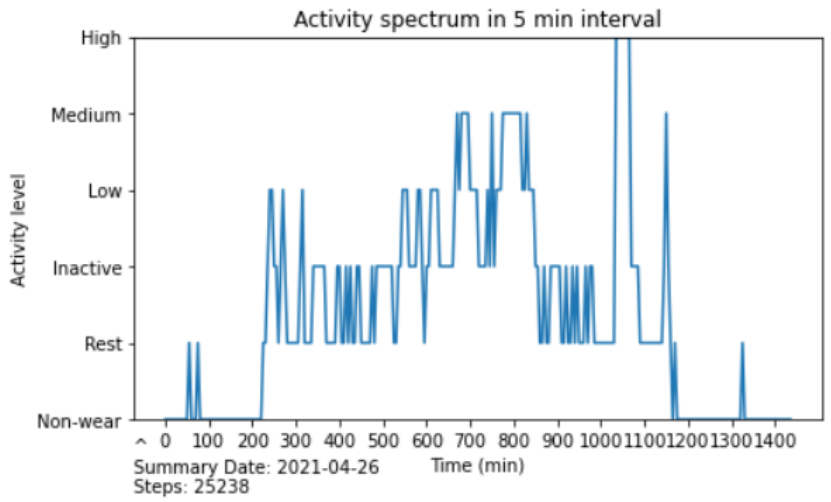


Figure 1: Activity spectrum of day with most walked steps by subject 2
Rest corresponds to sleeping or laying down, inactive to sitting, low to doing chores, medium to walking and high to exercising.

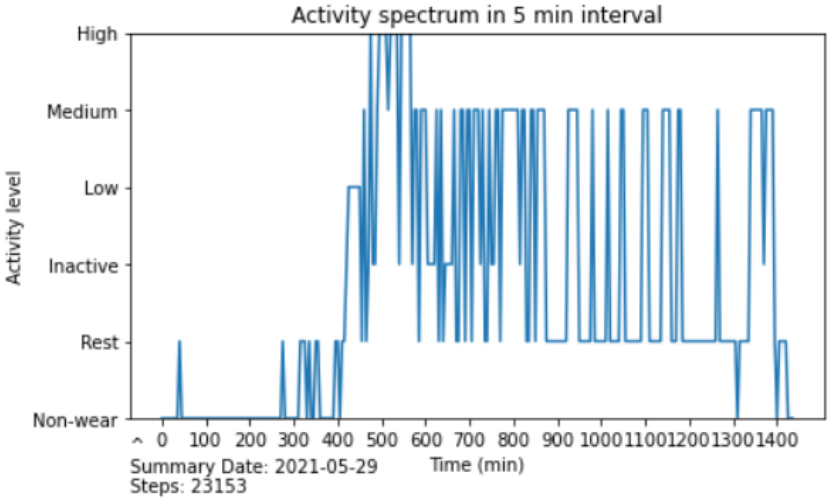


Figure 2: Activity spectrum of day with most walked steps by subject 3
Rest corresponds to sleeping or laying down, inactive to sitting, low to doing chores, medium to walking and high to exercising.

Furthermore, I also investigated the body temperature, because the Oura ring does measure that variable. The only problem is that it is stored as a deviation from the average night-time body temperature. While this might be useful to determine whether someone is getting ill, it makes it impossible to provide advice about the sleep environment, because if the user is too hot or too cold during the night, every night, then that temperature becomes the night-time average. In order to be sure that we could not make use of this data by combining it with the data from another variable, I decided to check for correlations between the temperature_deviation variable and other variables, but no correlation was found.

Lastly, I also inspected the relaxation techniques interventions, because the Oura ring contained heart rate and heart rate variability data, because both variables are influenced by the opposite: stress (Centre d'études sur le stress humain, 2007). It is not possible to use this data to provide advice, but it might be possible to provide evidence to the user about the effectiveness of relaxation techniques. Currently this cannot be done using the Oura ring however, because the Oura ring only collects these variables during sleep.

5.3 Validation

The automatically selected subset of micro-interventions have been compared against the answers provided to this questionnaire and have been put together into a table which can be found in Appendix I. From this data we can conclude that the automated selection is valid, because it corresponds to the data the user provided. From the six automated interventions, two are concerning sleep restriction therapy, which only has to be provided when the user wants to. The remaining four were correctly provided in cases where the subject also said it was applicable, and they were not provided in cases where the subject also said it was not applicable, for both subjects.

6 Discussion

6.1 Framework

Although this framework has been based upon limited literature, the framework has proven to be useful in automating the micro-intervention selection and therefore in personalizing dCBT-I. This framework can be used in future research and can be extended if new components are added to CBT-I.

6.2 Data collection and analysis

During the data collection I ran into corrupted JSON files, which contain all the interval data that is necessary to provide sensible advice. This data looked a lot like the image provided in Appendix J (jsonhelp, 2013). Trying to figure out other ways of accessing this data did not succeed. In the end I figured out that the data was corrupted due to some settings in my computer, which could be solved by installing the JSON file repair tool (Fix corrupt JSON Files, n.d.), which changes these settings.

6.3 Validation

The validation proves that it is possible to provide an automated personalized selection of interventions, meaning that this is a promising ground for further research. The validation could also have been performed on the first dataset, but that has not been done. Furthermore the pool of subjects could be extended to provide a wider array of subjects, because all subjects were male.

6.4 Future research

One possible direction for future research is trying to make an automated micro-intervention selection using other wearable devices. Another direction is looking into the efficacy of micro-interventions in CBT-I, both in general and when administered through automated means. Yet another topic could be to see whether the adherence rates to CBT-I treatments actually increase by personalizing the administration process and if so, how much. Furthermore the cost-effectivity of dCBT-I has not yet been determined, but the cost-effectivity of automated personalized dCBT-I is quite likely even higher. This should also be researched. Lastly, the framework provided in this research can be used to find an answer to the question: What is the optimal dosage of treatment for (d)CBT-I?

6.5 Conclusion

In conclusion: The micro-intervention framework makes it possible to create a collection of individually administrable, directly actionable interventions that target exactly those sleep-related problems of which the user indicates that they apply to his life, without any human being part of the selection process. Even though the actual automation rate is quite low using the Oura ring, any improvement in personalization is worth looking into,

because of the large intended userbase of dCBT-I applications. Any person helped with an automated personalization process takes some pressure off the underlying system for even more personal, face-to-face CBT-I, allowing patients with more severe insomnia to receive the help they need (Espie, 2009).

7 Compliance with Ethical Standards

7.1 Conflict of Interests Disclosure

Sjors Weggeman declares that he has no potential conflict of interest.

7.2 Human and Animal Rights

The author of this paper declares that he has not done any research with human or animal subjects, hence this research does not contain any studies done by the author that required approval of an ethics committee.

7.3 Confidentiality

The data used in this research falls under the General Data Protection Regulation (GDPR) of the European Union (European Parliament and the Council of 27 April 2016, 2016), meaning that it is treated with utmost care to ensure the privacy and confidentiality of the rightful owners of the data. Any data provided has been rid of any personal or identifiable information and has been stored securely. The data shall not be kept longer than necessary and shall not be shared without the explicit written consent of the rightful owner.

7.4 Informed Consent

All participants have been informed about the usage and treatment of their data and have all provided their informed consent.

8 Funding

This study did not receive any funding.

9 References

American Academy of Sleep Medicine. (2017). Two week sleep diary. National Institute for Health and Care Excellence, 1, 2. http://yoursleep.aasmnet.org/pdf/sleepdiary.pdf

American Psychiatric Association. (2013). *Diagnostic and Statistical Manual of Mental Disorders* (5th ed.). American Psychiatric Association. https://doi.org/10.1176/appi.books.9780890425596

Ayers, S., Fitzgerald, G., & Thompson, S. (2015). Brief Online Self-help Exercises for Postnatal Women to Improve Mood: A Pilot Study. *Maternal and Child Health Journal*, 19(11), 2375–2383. https://doi.org/10.1007/s10995-015-1755-5

Baglioni, C., Altena, E., Bjorvatn, B., Blom, K., Bothelius, K., Devoto, A., Espie, C. A., Frase, L., Gavriloff, D., Tuuliki, H., Hoflehner, A., Högl, B., Holzinger, B., Järnefelt, H., Jernelöv, S., Johann, A. F., Lombardo, C., Nissen, C., Palagini, L., … Riemann, D. (2020). The European Academy for Cognitive Behavioural Therapy for Insomnia: An initiative of the European Insomnia Network to promote implementation and dissemination of treatment. *Journal of Sleep Research*, 29(2), e12967. https://doi.org/10.1111/jsr.12967

Bhaskar, S., Hemavathy, D., & Prasad, S. (2016). Prevalence of chronic insomnia in adult patients and its correlation with medical comorbidities. *Journal of Family Medicine and Primary Care*, 5(4), 780. https://doi.org/10.4103/2249-4863.201153

Bianchi, M. T. (2018). Sleep devices: wearables and nearables, informational and interventional, consumer and clinical. *Metabolism: Clinical and Experimental*, 84, 99–108. https://doi.org/10.1016/j.metabol.2017.10.008

Blanken, T. F., Benjamins, J. S., Borsboom, D., Vermunt, J. K., Paquola, C., Ramautar, J., Dekker, K., Stoffers, D., Wassing, R., Wei, Y., & Van Someren, E. J. W. (2019). Insomnia disorder subtypes derived from life history and traits of affect and personality. *The Lancet Psychiatry*, 6(2), 151–163. https://doi.org/10.1016/S2215-0366(18)30464-4

BroadbandSearch. (2021). Key Internet Statistics to Know in 2021 (Including Mobile). https://www.broadbandsearch.net/blog/internet-statistics

Budhiraja, R., Roth, T., Hudgel, D. W., Budhiraja, P., & Drake, C. L. (2011). Prevalence and polysomnographic correlates of insomnia comorbid with medical disorders. *Sleep*, 34(7), 859–867. https://doi.org/10.5665/SLEEP.1114

Bunge, E. L., Williamson, R. E., Cano, M., Leykin, Y., & Muñoz, R. F. (2016). Mood management effects of brief unsupported internet interventions. *Internet Interventions*, 5, 36–43. https://doi.org/10.1016/j.invent.2016.06.001

Buysse, D.J., Reynolds, C.F., Monk, T.H., Berman, S.R., & Kupfer, D.J. (1989). The Pittsburgh Sleep Quality Index (PSQI): A new instrument for psychiatric research and practice. Psychiatry Research, 28(2), 193-213.

Cajita, M. I., Kline, C. E., Burke, L. E., Bigini, E. G., & Imes, C. C. (2020). Feasible but Not Yet Efficacious: a Scoping Review of Wearable Activity Monitors in Interventions Targeting Physical Activity, Sedentary Behavior, and Sleep. *Current Epidemiology Reports*. https://doi.org/10.1007/s40471-020-00251-4

Cambridge University Press. (n.d.). INTERVENTION | meaning in the Cambridge English Dictionary. Cambridge Dictionary. Retrieved June 18, 2021, from https://dictionary.cambridge.org/dictionary/english/intervention

Cambridge University Press. (n.d.). WEARABLE | definition in the Cambridge English Dictionary. Cambridge Dictionary. Retrieved June 18, 2021, from https://dictionary.cambridge.org/us/dictionary/english/wearable

Cellini, N., Canale, N., Mioni, G., & Costa, S. (2020). Changes in sleep pattern, sense of time and digital media use during COVID-19 lockdown in Italy. *Journal of Sleep Research*, 29(4). https://doi.org/10.1111/jsr.13074

Centre d'études sur le stress humain. (2007). How to measure stress in humans. 1–39. https://humanstress.ca/Documents/pdf/Mesures-physiologiques/CESH_howMesureStress-MB.pdf

Cheslaw, L. V. M. (2020). *Is Oura's $299 "Smart Ring" Just a Fancy Thermometer?* The Strategist. https://nymag.com/strategist/2020/07/oura-ring-review-2020.html

Clark, D. A. (2013). Cognitive Restructuring. The Wiley Handbook of Cognitive Behavioral Therapy, I, 1–22. https://doi.org/10.1002/9781118528563.wbcbt02

Corliss, J. (2016). Six Relaxation Techniques to Reduce Stress - Harvard Health. Harvard Health Publishing. https://www.health.harvard.edu/mind-and-mood/six-relaxation-techniques-to-reduce-stress

Cummings, N. A., & Cummings, J. L. (2008). Psychoeducation in Conjunction with Psychotherapy Practice. In Evidence-Based Adjunctive Treatments (pp. 41–59). Elsevier Inc. https://doi.org/10.1016/B978-012088520-6.50004-4

Daley, M., Morin, C. M., LeBlanc, M., Grégoire, J. P., & Savard, J. (2009). The economic burden of insomnia: Direct and indirect costs for individuals with

insomnia syndrome, insomnia symptoms, and good sleepers. *Sleep*, *32*(1), 55–64. https://doi.org/10.1016/s1073-5437(10)79480-8

Elefant, A. B., Contreras, O., Muñoz, R. F., Bunge, E. L., & Leykin, Y. (2017). Microinterventions produce immediate but not lasting benefits in mood and distress. *Internet Interventions*, 10(September), 17–22. https://doi.org/10.1016/j.invent.2017.08.004

Espie, C. A. (1993). Practical management of insomnia: Behavioural and cognitive techniques. In British Medical Journal (Vol. 306, Issue 6876, pp. 509–511).

Espie, C. A. (2009). "Stepped care": A health technology solution for delivering cognitive behavioral therapy as a first line insomnia treatment. In Sleep (Vol. 32, Issue 12, pp. 1549–1558). *Associated Professional Sleep Societies*, LLC. https://doi.org/10.1093/sleep/32.12.1549

Espie, C. A., Kyle, S. D., Williams, C., Ong, J. C., Douglas, N. J., Hames, P., & Brown, J. S. L. (2012). A randomized, placebo-controlled trial of online cognitive behavioral therapy for chronic insomnia disorder delivered via an automated media-rich web application. *Sleep*, 35(6), 769–781. https://doi.org/10.5665/sleep.1872

Espie, C. A., Hames, P., & McKinstry, B. (2013). Use of the internet and mobile media for delivery of cognitive behavioral insomnia therapy. *Sleep Medicine Clinics*, 8(3), 407–419. https://doi.org/10.1016/j.jsmc.2013.06.001

Espie, C. A., Emsley, R., Kyle, S. D., Gordon, C., Drake, C. L., Siriwardena, A. N., Cape, J., Ong, J. C., Sheaves, B., Foster, R., Freeman, D., Costa-Font, J., Marsden, A., & Luik, A. I. (2019). Effect of Digital Cognitive Behavioral Therapy for Insomnia on Health, Psychological Well-being, and Sleep-Related Quality of Life: A Randomized Clinical Trial. *JAMA Psychiatry*, 76(1), 21–30. https://doi.org/10.1001/jamapsychiatry.2018.2745

European Parliament and the Council of 27 April 2016. (2016). General Data Protection Regulation. OJ L 119, 59. https://doi.org/10.1017/9781780688459.021

Fix Corrupt JSON Files. (n.d.). Retrieved June 18, 2021, from https://www.fileerrors.com/corrupt-json-files.html

Fortier-Brochu, É., Beaulieu-Bonneau, S., Ivers, H., & Morin, C. M. (2010). Relations between sleep, fatigue, and health-related quality of life in individuals with insomnia. Journal of *Psychosomatic Research*, 69(5), 475–483. https://doi.org/10.1016/j.jpsychores.2010.05.005

Glazer Baron, K., Culnan, E., Duffecy, J., Berendson, M., Cheung Mason, I., Lattie, E., & Manalo, N. (2021). How are Consumer Sleep Technology Data Being Used to

Deliver Behavioral Sleep Medicine Interventions? A Systematic Review. *Behavioral Sleep Medicine*. https://doi.org/10.1080/15402002.2021.1898397

Imeri, L., & Opp, M. R. (2009). How (and why) the immune system makes us sleep. *Nature Reviews Neuroscience* 10(3), 199–210. https://doi.org/10.1038/nrn2576

Jaiswal, S. J., Topol, E. J., & Steinhubl, S. R. (2019). Digitising the way to better sleep health. *The Lancet*, 393(10172), 639. https://doi.org/10.1016/S0140-6736(19)30240-5

Journal of Clinical Sleep Medicine. (2020). In Journal of Clinical Sleep Medicine (Vol. 6, Issue 3). https://jcsm.aasm.org/info/about-jcsm

Jsonhelp. (2013). How to read/recover corrupt .json file : AskReddit. https://www.reddit.com/r/AskReddit/comments/wt8rm/how_to_readrecover_corrupt_json_file/

Kang, S. G., Kang, J. M., Ko, K. P., Park, S. C., Mariani, S., & Weng, J. (2017). Validity of a commercial wearable sleep tracker in adult insomnia disorder patients and good sleepers. *Journal of Psychosomatic Research*, *97*, 38–44. https://doi.org/10.1016/j.jpsychores.2017.03.009

Kessler, R. C., Berglund, P. A., Coulouvrat, C., Hajak, G., Roth, T., Shahly, V., Shillington, A. C., Stephenson, J. J., & Walsh, J. K. (2011). Insomnia and the performance of US workers: Results from the America Insomnia Survey. *Sleep*, *34*(9), 1161–1171. https://doi.org/10.5665/SLEEP.1230

Kinnunen Ouraring, H. (2016). Sleep Lab validation of a wellness ring in detecting sleep patterns based on photoplethysmogram, actigraphy and body temperature.

Kronholm, E., Partonen, T., Härmä, M., Hublin, C., Lallukka, T., Peltonen, M., & Laatikainen, T. (2016). Prevalence of insomnia-related symptoms continues to increase in the Finnish working-age population. *Journal of Sleep Research*, 25(4), 454–457. https://doi.org/10.1111/jsr.12398

Lacks, P., & Morin, C. M. (1992). Recent Advances in the Assessment and Treatment of Insomnia. Journal of Consulting and Clinical Psychology, 60(4), 586–594. https://doi.org/10.1037/0022-006X.60.4.586

Liu, Y., Wheaton, A. G., Chapman, D. P., Cunningham, T. J., Lu, H., & Croft, J. B. (2016). Prevalence of Healthy Sleep Duration among Adults — United States, 2014. *Morbidity and Mortality Weekly Report*, 65(6), 137–141. https://doi.org/10.15585/mmwr.mm6506a1

Lokman, S., Leone, S. S., Sommers-Spijkerman, M., Van Der Poel, A., Smit, F., & Boon, B. (2017). Complaint-directed mini-interventions for depressive complaints:

A randomized controlled trial of unguided web-based self-help interventions. *Journal of Medical Internet Research*, 19(1), e6581. https://doi.org/10.2196/jmir.6581

Luik, A. I., Kyle, S. D., & Espie, C. A. (2017). Digital Cognitive Behavioral Therapy (dCBT) for Insomnia: a State-of-the-Science Review. *Current Sleep Medicine Reports*, 3(2), 48–56. https://doi.org/10.1007/s40675-017-0065-4

Luik, A. I., Farias Machado, P., & Espie, C. A. (2018). Delivering digital cognitive behavioral therapy for insomnia at scale: does using a wearable device to estimate sleep influence therapy? *Npj Digital Medicine*, *1*(1), 1–6. https://doi.org/10.1038/s41746-017-0010-4

Luik, A. I., van der Zweerde, T., van Straten, A., & Lancee, J. (2019). Digital Delivery of Cognitive Behavioral Therapy for Insomnia. *Current Psychiatry Reports*, 21(7). https://doi.org/10.1007/s11920-019-1041-0

Millman, K. J., & Aivazis, M. (2011). Python for scientists and engineers. I*n Computing in Science and Engineering* (Vol. 13, Issue 2, pp. 9–12). https://doi.org/10.1109/MCSE.2011.36

Mohr, D. C., Cuijpers, P., & Lehman, K. (2011). Supportive accountability: A model for providing human support to enhance adherence to eHealth interventions. In *Journal of Medical Internet Research* (Vol. 13, Issue 1, p. e1602). JMIR Publications Inc. https://doi.org/10.2196/jmir.1602

Moreno-Pino, F., Porras-Segovia, A., López-Esteban, P., Artés, A., & Baca-García, E. (2019). Validation of fitbit charge 2 and fitbit alta hr against polysomnography for assessing sleep in adults with obstructive sleep apnea. *Journal of Clinical Sleep Medicine*, *15*(11), 1645–1653. https://doi.org/10.5664/jcsm.8032

Morin, C. M., Kowatch, R. A., Barry, T., & Walton, E. (1993). Cognitive-Behavior Therapy for Late-Life Insomnia. Journal of Consulting and Clinical Psychology, 61(1), 137–146. https://doi.org/10.1037/0022-006X.61.1.137

Morin, C. M., Hauri, P. J., Espie, C. A., Spielman, A. J., Buysse, D. J., & Bootzin, R. R. (1999). Nonpharmacologic treatment of chronic insomnia. *Sleep*, 22(8), 1134–1156. https://doi.org/10.1093/sleep/22.8.11344

Ōura Health Oy. (n.d.). API Documentation. Retrieved June 11, 2021, from https://cloud.ouraring.com/docs/

Ōura Health Oy. (n.d.). Get Started with Oura. The Pulse Blog. Retrieved June 11, 2021, from https://ouraring.com/blog/get-started/

Ōura Health Oy. (n.d.). Oura on the web. Retrieved June 11, 2021, from https://cloud.ouraring.com/profile

Partinen, M., Bjorvatn, B., Holzinger, B., Chung, F., Penzel, T., Espie, C. A., & Morin, C. M. (2021). Sleep and circadian problems during the coronavirus disease 2019 (COVID-19) pandemic: the International COVID-19 Sleep Study (ICOSS). *Journal of Sleep Research*, 30(1), e13206. https://doi.org/10.1111/jsr.13206

Perez-Pozuelo, I., Zhai, B., Palotti, J., Mall, R., Aupetit, M., Garcia-Gomez, J. M., Taheri, S., Guan, Y., & Fernandez-Luque, L. (2020). The future of sleep health: a data-driven revolution in sleep science and medicine. *Npj Digital Medicine*, 3(1), 1–15. https://doi.org/10.1038/s41746-020-0244-4

Pigeon, W. R., Bishop, T. M., & Krueger, K. M. (2017). Insomnia as a Precipitating Factor in New Onset Mental Illness: a Systematic Review of Recent Findings. In *Current Psychiatry Reports* (Vol. 19, Issue 8, pp. 1–11). Current Medicine Group LLC 1. https://doi.org/10.1007/s11920-017-0802-x

Qaseem, A., Kansagara, D., Forciea, M. A., Cooke, M., Denberg, T. D., Barry, M. J., Boyd, C., Chow, R. D., Fitterman, N., Harris, R. P., Humphrey, L. L., Manaker, S., McLean, R., Mir, T. P., Schünemann, H. J., Vijan, S., & Wilt, T. (2016). Management of chronic insomnia disorder in adults: A clinical practice guideline from the American college of physicians. *Annals of Internal Medicine*, 165(2), 125–133. https://doi.org/10.7326/M15-2175

Riemann, D., Baglioni, C., Bassetti, C., Bjorvatn, B., Dolenc Groselj, L., Ellis, J. G., Espie, C. A., Garcia-Borreguero, D., Gjerstad, M., Gonçalves, M., Hertenstein, E., Jansson-Fröjmark, M., Jennum, P. J., Leger, D., Nissen, C., Parrino, L., Paunio, T., Pevernagie, D., Verbraecken, J., … Spiegelhalder, K. (2017). European guideline for the diagnosis and treatment of insomnia. *Journal of Sleep Research*, 26(6), 675–700. https://doi.org/10.1111/jsr.12594

Riemann, D., Baum, E., Cohrs, S., Crönlein, T., Hajak, G., Hertenstein, E., Klose, P., Langhorst, J., Mayer, G., Nissen, C., Pollmächer, T., Rabstein, S., Schlarb, A., Sitter, H., Weeß, H.-G., Wetter, T., & Spiegelhalder, K. (2017). S3-Leitlinie Nicht erholsamer Schlaf/Schlafstörungen. *Somnologie*, 21(1), 2–44. https://doi.org/10.1007/s11818-016-0097-x

Schröder, J., Berger, T., Westermann, S., Klein, J. P., & Moritz, S. (2016). Internet interventions for depression: New developments. *Dialogues in Clinical Neuroscience*, 18(2), 203–212. https://doi.org/10.31887/dcns.2016.18.2/jschroeder

Ström, L., Pettersson, R., & Andersson, G. (2004). Internet-Based Treatment for Insomnia: A Controlled Evaluation. *Journal of Consulting and Clinical Psychology*, 72(1), 113–120. https://doi.org/10.1037/0022-006X.72.1.113

Thorndike, F. P., Ritterband, L. M., Gonder-Frederick, L. A., Lord, H. R., Ingersoll, K. S., & Morin, C. M. (2013). A randomized controlled trial of an internet intervention for adults with insomnia: Effects on comorbid psychological and fatigue symptoms. *Journal of Clinical Psychology*, 69(10), 1078–1093. https://doi.org/10.1002/jclp.22032

Vox Media. (n.d.). *The Strategist*. New York Magazine. Retrieved June 9, 2021, from http://mediakit.nymag.com/the-strategist/

Walker, M. (2017). *Why We Sleep: Unlocking the Power of Sleep and Dreams* (1st ed.). Scribner.

Xu, P., González-Vallejo, C., & Vincent, B. T. (2020). Waiting in intertemporal choice tasks affects discounting and subjective time perception. Journal of Experimental Psychology: General, 149(12), 2289–2313. https://doi.org/10.1037/xge0000771

Zachariae, R., Lyby, M. S., Ritterband, L. M., & O'Toole, M. S. Efficacy of internet-delivered cognitive-behavioral therapy for insomnia—a systematic review and meta-analysis of randomized controlled trials. *Sleep Med. Rev*. 30, 1–10 (2015).

de Zambotti, M., Rosas, L., Colrain, I. M., & Baker, F. C. (2019). The Sleep of the Ring: Comparison of the ŌURA Sleep Tracker Against Polysomnography. *Behavioral Sleep Medicine*, *17*(2), 124–136. https://doi.org/10.1080/15402002.2017.1300587

van der Zweerde, T., Lancee, J., Ida Luik, A., & van Straten, A. (2019). Internet-Delivered Cognitive Behavioral Therapy for Insomnia: Tailoring Cognitive Behavioral Therapy for Insomnia for Patients with Chronic Insomnia. *Sleep Medicine Clinics*, *14*(3), 301–315. https://doi.org/10.1016/j.jsmc.2019.04.002

Appendices

Appendix A

Question to link your answers to the right analysis whilst maintaining your anonymity

Question 1: What is the filename of the dataset you provided?
Answer: _____

Questions for research completeness

Question 2: What is your age?
Answer: _____

Question 3: What is your gender?
Answer: _____

Question 4: How often do you wear the Oura ring during a week?
Answer:
O Daily
O Every other day
O Infrequently
O Other, namely:_____

Question 5: At what time do you wear the Oura ring?
Answer:
O Only during the day
O Only at night
O Day and night
O Other, namely:_____

Questions for validation of the analysis

Question 6: During the past month, what time have you usually gone to bed at night?
Answer: _____

Question 7: During the past month, how long (in minutes) has it usually taken you to fall asleep each night?
Answer: _____

Question 8: During the past month, what time have you usually gotten up in the morning?
Answer: _____

Question 9: During the past month, how many hours of actual sleep did you get at night?
(This may be different than the number of hours you spent in bed)
Answer: _____

Question 10: During the past month, how many times have you been napping during the day?

Answer:
O Not during the past month
O Less than once a week
O Once or twice a week
O Three or more times a week

Question 11: During the past month, how many times have you woken up at Night for a prolonged period of time?
(Approximately 15 minutes or more)

Answer:
O Not during the past month
O Less than once a week
O Once or twice a week
O Three or more times a week

If the previous question has been answered with 'Not during the past month' skip to question 13

Question 12: Did you stay in bed during this period?

Answer: _____

Question 13: During the past month, how many times have you been walking for more than an hour or have you been exercising?

Answer:
O Not during the past month
O Less than once a week
O Once or twice a week
O Three or more times a week

Question for the use of this data

In order to be able to use this data and publish it in the research report I am required to ask for your consent. If you give consent you agree that this data as well as the dataset you provided may be used in this research project. Your data will be anonymized and will be treated according to the GDPR, it will be stored safely and will not be kept longer than necessary. Your data will not be shared without your explicit written consent, which will be asked separately when a request is made. If you do not give consent, please specify why. You are not obliged to provide a reason, but it would help me to improve this research.

Question 14: Do you give consent for the use of this data in the bachelor project?

Answer:
O I give consent
O I do **not** give consent, because: _____
_____

Appendix B

Subject's Initials_____ID#_____Date_____Time_____AM PM

PITTSBURGH SLEEP QUALITY INDEX

INSTRUCTIONS:
The following questions relate to your usual sleep habits during the past month only. Your answers should indicate the most accurate reply for the majority of days and nights in the past month. Please answer all questions.

1. During the past month, what time have you usually gone to bed at night?

   BED TIME _____

2. During the past month, how long (in minutes) has it usually taken you to fall asleep each night?

   NUMBER OF MINUTES _____

3. During the past month, what time have you usually gotten up in the morning?

   GETTING UP TIME _____

4. During the past month, how many hours of actual sleep did you get at night? (This may be different than the number of hours you spent in bed.)

   HOURS OF SLEEP PER NIGHT _____

For each of the remaining questions, check the one best response. Please answer all questions.

5. During the past month, how often have you had trouble sleeping because you . . .

a) Cannot get to sleep within 30 minutes

Not during the past month_____	Less than once a week_____	Once or twice a week_____	Three or more times a week_____

b) Wake up in the middle of the night or early morning

Not during the past month_____	Less than once a week_____	Once or twice a week_____	Three or more times a week_____

c) Have to get up to use the bathroom

Not during the past month_____	Less than once a week_____	Once or twice a week_____	Three or more times a week_____

d) Cannot breathe comfortably

Not during the past month_____ Less than once a week_____ Once or twice a week_____ Three or more times a week_____

e) Cough or snore loudly

Not during the past month_____ Less than once a week_____ Once or twice a week_____ Three or more times a week_____

f) Feel too cold

Not during the past month_____ Less than once a week_____ Once or twice a week_____ Three or more times a week_____

g) Feel too hot

Not during the past month_____ Less than once a week_____ Once or twice a week_____ Three or more times a week_____

h) Had bad dreams

Not during the past month_____ Less than once a week_____ Once or twice a week_____ Three or more times a week_____

i) Have pain

Not during the past month_____ Less than once a week_____ Once or twice a week_____ Three or more times a week_____

j) Other reason(s), please describe_____

_____

How often during the past month have you had trouble sleeping because of this?

Not during the past month_____ Less than once a week_____ Once or twice a week_____ Three or more times a week_____

6. During the past month, how would you rate your sleep quality overall?

Very good _____

Fairly good _____

Fairly bad _____

Very bad _____

7. During the past month, how often have you taken medicine to help you sleep (prescribed or "over the counter")?

Not during the past month_____ Less than once a week_____ Once or twice a week_____ Three or more times a week_____

8. During the past month, how often have you had trouble staying awake while driving, eating meals, or engaging in social activity?

Not during the past month_____ Less than once a week_____ Once or twice a week_____ Three or more times a week_____

9. During the past month, how much of a problem has it been for you to keep up enough enthusiasm to get things done?

No problem at all _____

Only a very slight problem _____

Somewhat of a problem _____

A very big problem _____

10. Do you have a bed partner or room mate?

No bed partner or room mate _____

Partner/room mate in other room _____

Partner in same room, but not same bed _____

Partner in same bed _____

If you have a room mate or bed partner, ask him/her how often in the past month you have had . . .

a) Loud snoring

Not during the past month_____ Less than once a week_____ Once or twice a week_____ Three or more times a week_____

b) Long pauses between breaths while asleep

Not during the past month_____ Less than once a week_____ Once or twice a week_____ Three or more times a week_____

c) Legs twitching or jerking while you sleep

Not during the past month_____ Less than once a week_____ Once or twice a week_____ Three or more times a week_____

d) Episodes of disorientation or confusion during sleep

Not during the past month_____ Less than once a week_____ Once or twice a week_____ Three or more times a week_____

e) Other restlessness while you sleep; please describe_____

_____

Not during the past month_____ Less than once a week_____ Once or twice a week_____ Three or more times a week_____

Buysse DJ, Reynolds CF, Monk TH, Berman SR, Kupfer DJ: Psychiatry Research, 28:193-213, 1989.

Appendix C

Table 1: CBT-I components mentioned in literature

Year	Author	Title	Interventions mentioned
1992	Lacks, P. & Morin, C.M.	Recent Advances in the Assessment and Treatment of Insomnia	Stimulus Control Relaxation-based interventions: - progressive relaxation/autogenic training - electromyographic biofeedback - meditation - imagery Paradoxical intention Cognitive restructuring Sleep restriction therapy Life-style factors: - Sleep hygiene
1993	Morin et al.	Cognitive-Behavior Therapy for Late-Life Insomnia	Stimulus control: - Go to bed only when sleepy - Use the bed and bedroom only for sleeping and sex - Get out of bed when unable to sleep and return to bed only when sleepy again - Arise at the same time every morning - Do not nap more than one hour a day and preferably before 3pm. Sleep restriction: - Initial reduction allowed time in bed - Weekly adjustment of the allowed time in bed Cognitive therapy: - Identification of dysfunctional beliefs - Confrontation and challenging of those thoughts - Cognitive restructuring Sleep hygiene: - Caffeine

			- Nicotine - Alcohol - Drugs - Exercise - Diet Education: - Sleep-wake patterns - How sleep changes over a lifetime
1993	Espie, C.A.	Practical Management of Insomnia: Behavioural and Cognitive Techniques	Education: - Sleep stages - Sleep functions - Sleep developmental changes Sleep hygiene: - Exercise - Diet - Caffeine - Alcohol - Environment Sleep scheduling Wind down: - Relaxation: - Relaxation techniques Cognitive methods

1993	Sloan et al.	The Nuts and Bolts of Behavioral Therapy for Insomnia	Sleep hygiene:
			- Sleep strategy:
			  - Rise at the same time every day
			  - Get out of bed when unable to sleep and only return to bed when sleepy again
			  - Explore naps: take them if you sleep good the night after, otherwise do not take naps
			  - Exercise regularly
			  - Wind down:
			  - Relaxation techniques
			  - Meditate
			  - Do not worry whilst in bed
			- Sleep environment:
			  - Noise
			  - Temperature
			  - Bed partner
			  - Bed

			- Substances:
			  - Drugs
			  - Fluids
			  - Avoid eating heavy meals before bedtime
			  - Caffeine
			  - Alcohol
			  - Smoking
			
			Stimulus control:
			- Only go to bed when sleepy
			- Do not use the bed for anything except sleep or sex
			- Get out of bed when unable to sleep and only return to bed when sleepy again
			- Rise at the same time every morning
			- Do not nap during the day
			
			Sleep restriction
			
			Cognitive therapy

1999	Morin et al.	Nonpharmacologic treatment of chronic insomnia	Stimulus control Progressive muscle relaxation Paradoxical intention Sleep restriction Biofeedback Multifaceted CBT
2003	Morin & Espie	Insomnia: A Clinical Guide to Assessment and Treatment	Sleepy hygiene: - Caffeine - Nicotine - Alcohol - Diet - Exercise - Noise - Room temperature - Body temperature - Air quality - Lighting - Mattress and pillows Relaxation therapy:

			- Bedtime wind-down
			- Relaxation training
			
			Sleep scheduling:
			- Restrict time spent in bed
			- Rise at the same time every day
			- Go to bed only when sleepy
			- Adjust the new schedule
			- Only use the bed for sleeping or sex
			- Do not take naps during the day
			
			Cognitive therapy:
			- Identifying dysfunctional beliefs
			- Changing dysfunctional beliefs:
			  - Keep realistic expectations
			  - Revise attributions about the causes of insomnia
			  - Do not blame the lack of sleep on the daytime impairments
			  - Decatastrophize
			  - Do not overemphasize sleep
			  - Try to accept the loss of sleep
			  - Do not try to sleep
			- Paradoxical intention
			- Cognitive control
			- Thought-blocking

2017	Riemann et al.	European Guideline for the Diagnosis and Treatment of Insomnia	Psychoeducation/Sleep hygiene:
			- Sleep hygiene rules:
			  - Sleep activities
			  - Lifestyle
			  - Environment
			- Basic information about normal sleep
			- Basic information about age-related changes in sleep patterns
			
			Relaxation training
			
			Stimulus control therapy
			
			Sleep restriction therapy
			
			Cognitive therapy

Appendix D

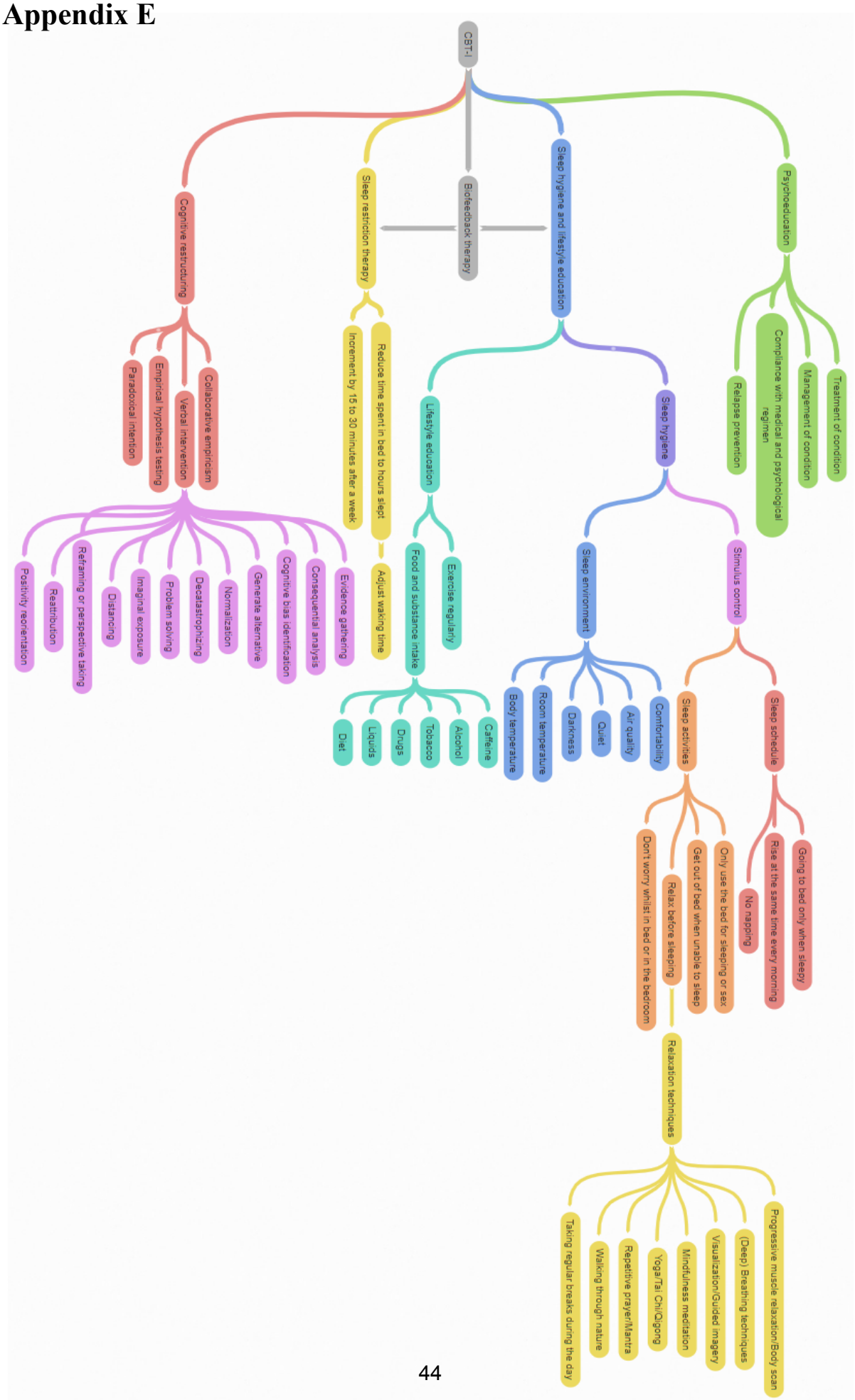
Table 2: CBT-I components mentioned in additional literature

Year	Author	Title	Interventions mentioned
2008	Cummings & Cummings	Psychoeducation in Conjunction with Psychotherapy Practice	Psychoeducation: All components are necessary otherwise it is not psychoeducation: - Treatment of the condition - Management of the condition - Compliance with the medical and psychological regimen - Relapse prevention

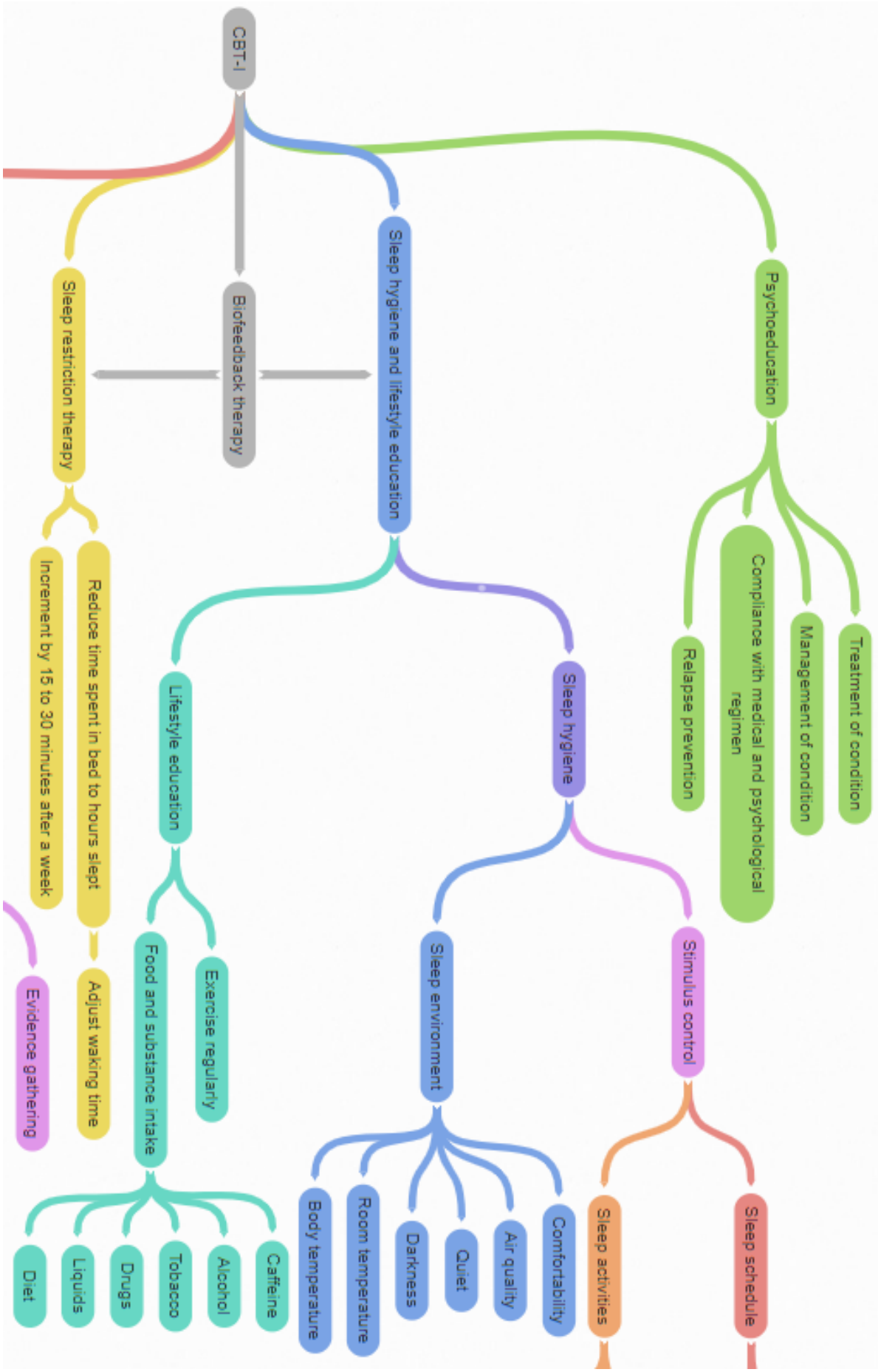
2013	Clark	Cognitive Restructuring	Cognitive restructuring:
			All components are necessary for an intervention to be cognitive restructuring intervention:
			- Collaborative empiricism
			- Verbal interventions
			  - Evidence gathering
			  - Consequential analysis
			  - Cognitive bias identification
			  - Generate alternative
			  - Normalization
			  - Decatastrophizing
			  - Problem solving
			  - Imaginal exposure
			  - Distancing
			  - Reframing or perspective taking
			  - Reattribution
			  - Positivity reorientation
			- Empirical hypothesis testing

2016	Corliss	Six Relaxation Techniques to Reduce Stress	Relaxation techniques: - Breath focus - Body scan - Guided imagery - Mindfulness meditation - Yoga, Tai Chi, Qigong - Repetitive prayer

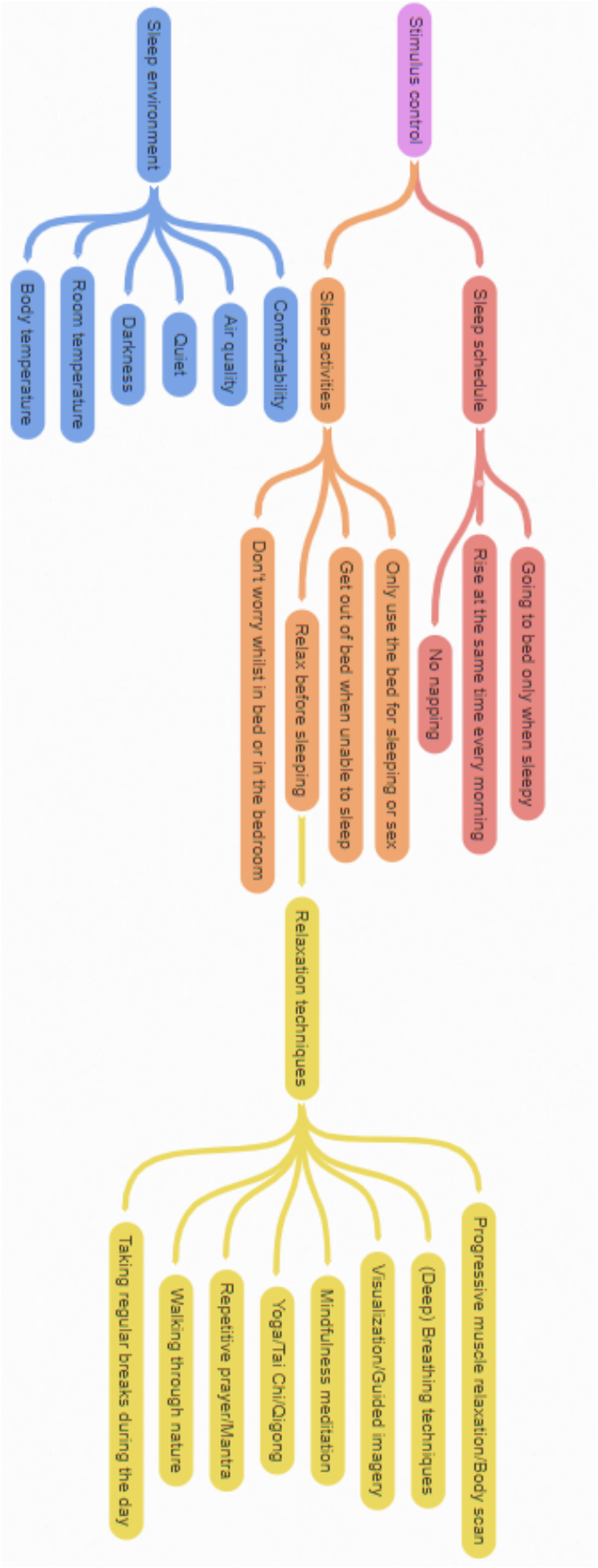
Appendix E


CBT-I
Cognitive restructuring
Paradoxical intention
Empirical hypothesis testing
Verbal intervention
Collaborative empiricism
Positivity reorientation
Reattribution
Reframing or perspective taking
Distancing
Imaginal exposure
Problem solving
Decatastrophizing
Normalization
Generate alternative
Cognitive bias identification
Consequential analysis
Evidence gathering
Sleep restriction therapy
Increment by 15 to 30 minutes after a week
Reduce time spent in bed to hours slept
Adjust waking time
Biofeedback therapy
Sleep hygiene and lifestyle education
Lifestyle education
Food and substance intake
Diet
Liquids
Drugs
Tobacco
Alcohol
Caffeine
Exercise regularly
Sleep hygiene
Sleep environment
Body temperature
Room temperature
Darkness
Quiet
Air quality
Comfortability
Stimulus control
Sleep activities
Don't worry whilst in bed or in the bedroom
Relax before sleeping
Get out of bed when unable to sleep
Only use the bed for sleeping or sex
Relaxation techniques
Taking regular breaks during the day
Walking through nature
Repetitive prayer/Mantra
Yoga/Tai Chi/Qigong
Mindfulness meditation
Visualization/Guided imagery
(Deep) Breathing techniques
Progressive muscle relaxation/Body scan
Sleep schedule
No napping
Rise at the same time every morning
Going to bed only when sleepy
Psychoeducation
Relapse prevention
Compliance with medical and psychological regimen
Management of condition
Treatment of condition

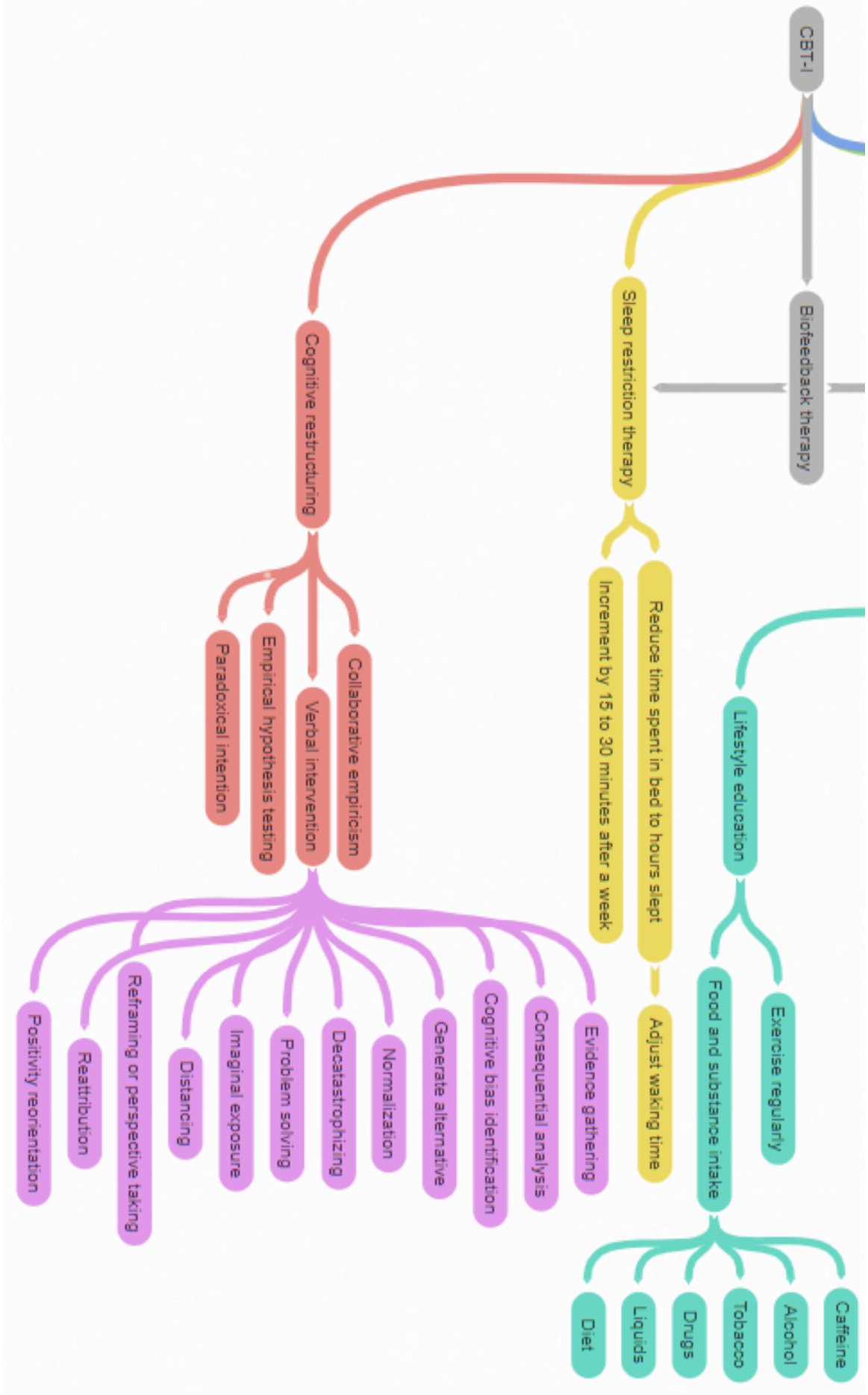
Appendix E - 1


CBT-I
Sleep restriction therapy
Biofeedback therapy
Sleep hygiene and lifestyle education
Psychoeducation
Increment by 15 to 30 minutes after a week
Reduce time spent in bed to hours slept
Lifestyle education
Sleep hygiene
Relapse prevention
Compliance with medical and psychological regimen
Management of condition
Treatment of condition
Evidence gathering
Adjust waking time
Food and substance intake
Exercise regularly
Sleep environment
Stimulus control
Diet
Liquids
Drugs
Tobacco
Alcohol
Caffeine
Body temperature
Room temperature
Darkness
Quiet
Air quality
Comfortability
Sleep activities
Sleep schedule

Appendix E - 2


Stimulus control
Sleep activities
Sleep schedule
Sleep environment
Comfortability
Air quality
Quiet
Darkness
Room temperature
Body temperature
Don't worry whilst in bed or in the bedroom
Relax before sleeping
Get out of bed when unable to sleep
Only use the bed for sleeping or sex
No napping
Rise at the same time every morning
Going to bed only when sleepy
Relaxation techniques
Taking regular breaks during the day
Walking through nature
Repetitive prayer/Mantra
Yoga/Tai Chi/Qigong
Mindfulness meditation
Visualization/Guided imagery
(Deep) Breathing techniques
Progressive muscle relaxation/Body scan

Appendix E - 3


CBT-I
Biofeedback therapy
Sleep restriction therapy
Reduce time spent in bed to hours slept
Adjust waking time
Increment by 15 to 30 minutes after a week
Cognitive restructuring
Collaborative empiricism
Verbal intervention
Empirical hypothesis testing
Paradoxical intention
Lifestyle education
Exercise regularly
Food and substance intake
Caffeine
Alcohol
Tobacco
Drugs
Liquids
Diet
Evidence gathering
Consequential analysis
Cognitive bias identification
Generate alternative
Normalization
Decatastrophizing
Problem solving
Imaginal exposure
Distancing
Reframing or perspective taking
Reattribution
Positivity reorientation

Appendix F

This code is part of the bachelor thesis project: Personalizing Digital Cognitive Behavioral Therapy for Insomnia Through Automation of Micro-Intervention Selection Based on Data from Wearable Techology; A Case Study on the Oura Ring

Author Information

Author: Sjors Weggeman\ Student number: s4799771\ University: Radboud University Nijmegen\ Email address: sjors.weggeman@student.ru.nl

Imports

In [1]:

```
#Importing the necessary packages
import json
import matplotlib
import matplotlib.pyplot as plt
import numpy as np
import re
import time

from datetime import date
from statsmodels import robust
```

Loading the data

In [2]:

```
#Opening the file
file=open('oura_try.json')

#Loading the file into a workable dataframe
data=json.load(file)
```

Inspecting the data

In [3]:

```
#Reformatting the data into a representation that looks like the Oura API documentation
(Ōura Health Oy, n.d.) and printing the first entry of every section of the data (bold)
```

```
for i in data:
        counter=-1
        print('\033[1m'+i+'\033[0m')
        for j in data[i]:
        counter+=1
        if counter==0:
        for k in j:
                print(k+': '+repr(j[k]))
        print('counter=',counter,'\n')
```

activity
average_met: 1.28125
cal_active: 303
cal_total: 2426
class_5min:
'000000000000000000000000000000000000000000000000000000000000000000000000
22232222333322322223333322333323222333320322322321222333223222222223321322
3444233222223442223332333332232222232122222222232222122222322342222212222222
1222322342222122111111111111111111111111111111111122111111111111'
daily_movement: 5588
day_end: '2019-08-16T03:59:59+02:00'
day_start: '2019-08-15T04:00:00+02:00'
high: 1
inactive: 500
inactivity_alerts: 0
low: 244
medium: 27
met_1min: [0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1,
0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1,
0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1,
0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1,
0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1,
0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1,
0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1,
0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1,
0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1,
0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1,
0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1,
0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1,
0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1,
0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1,
0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1,
0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1,
0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1,
0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 1.3, 1.6, 1.5, 1.9, 1.3, 1.2, 1.5, 1.2, 1.4,
1.2, 0.1, 0.1, 1.6, 3.3, 2, 2.1, 2.3, 3.2, 2, 1.4, 2.3, 1.4, 1.4, 1.2, 1.2, 1.4, 1.2, 1.3, 1.3, 1.2, 1.2,
1.2, 1.2, 1.2, 1.6, 1.2, 1.2, 1.6, 3, 2.6, 1.5, 2, 1.3, 1.2, 1.2, 1.3, 1.5, 1.2, 1.3, 3.2, 3.4, 2.9, 1.9,
1.3, 1.9, 3, 2.3, 2.7, 2.3, 1.3, 1.4, 1.4, 1.4, 1.2, 1.3, 1.2, 1.2, 1.3, 1.4, 1.9, 3.3, 2.7, 2, 1.2, 2.4,
1.3, 1.2, 1.3, 1.2, 1.2, 1.2, 1.2, 1.2, 1.2, 1.2, 1.3, 1.2, 1.2, 1.2, 1.2, 1.2, 1.2, 1.8, 1.3, 1.4, 1.1,

1.2, 3.9, 3, 1.4, 1.5, 1.8, 1.9, 2.5, 2.1, 2.3, 1.6, 1.5, 1.9, 1.6, 2.3, 1.7, 2, 1.5, 2.2, 1.5, 1.7, 1.5,
2.8, 2.5, 1.4, 1.4, 1.3, 1.3, 1.4, 1.4, 1.2, 1.4, 1.4, 1.4, 1.3, 1.5, 1.8, 2.1, 2.3, 3.2, 3.2, 1.4, 2.5,
1.8, 1.6, 1.3, 1.8, 1.9, 2, 1.5, 1.9, 1.3, 1.7, 1.3, 1.3, 1.7, 1.5, 1.2, 1.4, 1.3, 2.2, 2, 1.8, 1.7, 1.5,
1.3, 1.5, 1.2, 1.2, 1.2, 1.2, 1.2, 1.4, 1.2, 1.3, 1.3, 1.2, 1.2, 1.2, 1.6, 1.4, 1.3, 2.8, 2.1, 2, 1.5,
1.4, 1.3, 1.3, 1.2, 1.5, 2.1, 1.8, 1.6, 1.9, 4.5, 1.2, 0, 1.2, 1.1, 0.9, 0.9, 0.1, 0.1, 0.1, 0.1, 0.1,
1.6, 2, 1.7, 1.8, 1.8, 1.8, 1.8, 1.5, 1.2, 1.2, 1.5, 1.3, 1.2, 1.3, 1.3, 2.3, 2.2, 3.6, 2.4, 1.8, 4.2,
1.2, 1.7, 1.9, 1.3, 1.3, 1.3, 1.3, 1.3, 1.3, 1.3, 1.8, 2.4, 1.8, 1.6, 1.7, 1.5, 1.3, 1.3, 1.4, 1.9, 1.4,
1.4, 1.3, 1.2, 1.3, 1.3, 1.9, 1.3, 1.3, 1.3, 2.6, 1.4, 1.6, 1.2, 1.2, 1.7, 1.4, 1.2, 1.2, 1.2, 1.3, 1.4,
2.1, 2.4, 2.6, 1.3, 2.2, 2.7, 1.7, 2.3, 3.4, 2.9, 1.7, 1.2, 1.2, 1.2, 1.3, 1.2, 1.2, 1.3, 1.3, 1.3, 1.2,
1.3, 1.4, 4.9, 2.6, 1.3, 1.3, 4.1, 1.4, 2.1, 1.4, 1.2, 1.2, 1.4, 1.2, 1.2, 1.7, 1.6, 2.7, 0.1, 0.1, 0.1,
1.6, 1.2, 1.2, 1.2, 1.2, 1.2, 1.2, 1.2, 1.3, 1.2, 1.3, 1.4, 1.2, 1.2, 1.2, 1.2, 1.2, 1.2, 1, 1.2, 1.2,
1.4, 2.6, 2.1, 1.7, 2, 1.8, 3.3, 1.6, 1.2, 1.4, 1.3, 1.2, 1.2, 1.2, 1.2, 1.3, 1.2, 1.2, 1.3, 1.4, 1.5,
2.3, 2.7, 1.8, 1.6, 1.2, 1.3, 1.3, 1.3, 1.4, 1.2, 1.2, 1.2, 1.4, 1.3, 1.4, 1.3, 2.3, 2.4, 1.2, 1.2, 1.3,
1.4, 5.2, 5.1, 5.4, 4.7, 1.9, 2, 4, 5.2, 5.2, 4.9, 2.4, 1.2, 1.2, 1.2, 1.2, 1.6, 1.2, 1.2, 1.2, 1.2, 1.6,
3.2, 1.7, 2.1, 2.3, 1.7, 2.5, 1.7, 1.2, 1.2, 1.2, 0.9, 0.9, 0.9, 1, 1.8, 0.9, 0.1, 0.1, 1.2, 1, 1, 1.4,
1.3, 1.1, 1, 1.2, 1.3, 1.3, 2.3, 2, 2.5, 1.8, 1.4, 1.7, 4.8, 4.2, 5.7, 5.4, 8.3, 4.3, 2.1, 1.2, 1.2, 1.2,
1.2, 1.7, 1.7, 1.2, 1.2, 1.3, 1.2, 1.3, 1.2, 1.2, 1.2, 1.2, 1.2, 1.2, 1.2, 1.9, 3.4, 3.9, 2.9, 2.8, 4.1,
3.9, 3, 3.9, 4.2, 2.5, 1.7, 1.6, 1.3, 1.4, 1.3, 1.4, 1.7, 1.7, 1.5, 1.9, 1.5, 2.5, 3.2, 3, 1.9, 3.4, 2.5,
2.4, 2.7, 2.3, 1.6, 2.7, 2.3, 1.8, 1.8, 1.8, 2.7, 2.2, 2, 2.8, 1.3, 1.4, 1.5, 1.3, 1.4, 1.3, 1.6, 2.2,
1.6, 1.2, 1.2, 1.2, 1.3, 1.2, 2, 2.6, 2.5, 1.4, 1.3, 1.3, 1.3, 1.5, 1.4, 1.4, 1.5, 1.7, 1.2, 1.2, 1.3,
1.4, 1.3, 1.4, 2.1, 1.6, 1.5, 1.3, 1.2, 1.2, 1.2, 1.2, 1.8, 3, 2.9, 2.9, 2, 1.3, 1.2, 1.1, 1.2, 1.2, 1.1,
0.9, 0.9, 0.9, 1.1, 1, 0.9, 0, 0, 1.3, 1.3, 1.2, 1.2, 1.2, 1.4, 1.3, 1.3, 1.2, 1.2, 1.2, 1.2, 1.2, 1.2,
1.2, 1.2, 1.2, 1.2, 1.1, 1.1, 1.2, 1, 1.2, 0.9, 0.9, 0.9, 1.1, 1.3, 1.5, 1.4, 1.3, 1.2, 1.3, 1.2, 1.3,
1.5, 1.3, 2.2, 2.3, 2.6, 1.3, 1.5, 1.8, 1.3, 1.5, 1.7, 1.5, 1.3, 1.3, 1.2, 1.3, 1.4, 1.2, 1.4, 1.2, 1.2,
1.2, 1.2, 1.2, 1.2, 1.2, 0.9, 1, 0.9, 1.1, 0.9, 1, 0.9, 0.9, 1, 1.1, 2.8, 1.3, 0.9, 1.3, 1.5, 1.3, 1.2,
1.3, 1.4, 1.3, 1.4, 1.6, 1.2, 1.2, 1.3, 2, 1.6, 1.4, 1.4, 1.9, 1.7, 1.6, 1.6, 1.9, 2.6, 1.3, 1.5, 1.4,
1.3, 1.5, 1.2, 1.2, 1.2, 1.3, 1.2, 1.4, 1.7, 1.5, 1.7, 3.7, 5.5, 2.4, 1.8, 5.2, 4.8, 1.6, 1.3, 1.4, 1.5,
1.2, 1.6, 1.6, 1.2, 1.2, 1.3, 1.4, 1.2, 1.4, 1.4, 1.2, 1.2, 1.2, 1.3, 1.2, 1.2, 1.2, 1.2, 1.2, 1.2, 1.2,
1.2, 1.3, 1.2, 1.2, 0.9, 0.9, 0.9, 1, 0.9, 1.2, 0.9, 1, 1.1, 1.1, 1.7, 1.4, 2.9, 1.5, 1.7, 1.5, 1.6, 1.3,
1.4, 1.6, 1.3, 1.5, 1.3, 1.3, 1.2, 1.2, 1.2, 1.2, 1.2, 1.4, 1.3, 1.3, 1.2, 1.2, 1.2, 1.2, 1.3, 1.3, 1.3,
1.3, 1.3, 1.2, 1.2, 1.2, 1.2, 1.2, 1.4, 1.3, 1.3, 1.3, 1.3, 1.4, 1.9, 1.7, 1.4, 2.3, 1.9, 1.2, 1.2, 1.2,
1.2, 1.2, 1.2, 1.2, 1.2, 1.4, 1.5, 1.2, 2, 1.4, 1.2, 1.4, 2.9, 2.9, 4.9, 4.1, 1.2, 1.3, 1.4, 1.2, 1.2,
1.3, 1.2, 1.2, 1.2, 1.2, 1.2, 1.3, 1.2, 1.2, 1.2, 1.2, 1.3, 1.2, 1.2, 1.2, 1.2, 1.2, 1.2, 1.2, 1.2, 1.3,
1.2, 1.3, 1.2, 1.2, 1.2, 1.2, 1.2, 1.3, 1.2, 1.2, 1.1, 0.9, 0.9, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.1, 0.9, 0.9, 0.9, 0.9,
0.9, 0.9, 0.9, 0.9, 0.9, 0.9, 0.9, 0.9, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 1.1, 1, 0.9, 0.9, 0.9, 0.9, 0.9, 0.9, 0.9, 0.9, 0.9, 0.9, 1, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.1, 0.9, 0.9, 0.9, 0.9, 0.9, 1.2, 0.9, 0.9, 0.9, 0.9, 0.9, 0.9, 0, 0,
0, 0, 0, 0, 0, 1.2, 0.9, 1, 0.9, 1.1, 0.9, 0.9, 0.9, 0.9, 1.2, 0.9, 0.9, 0.9, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 1.1, 1, 0.9, 0.9, 0.9, 0.9, 0.9, 0.9, 1, 0.9, 1, 0.9, 0.9, 0, 1.4, 3.2, 1.2, 0.9, 0.9, 0.9, 0.9,
0.9, 0.9, 0.9, 0.9, 0.9, 0.9, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.1, 0.9, 1, 1, 0.9, 0.9,
0.9, 0.9, 0.9, 1.1, 0.9, 0.9, 0.9, 1.1, 0.9, 0.9, 0.9, 0.9, 0.9, 1.1, 1.3, 0.9, 0.9, 0.9, 0.9, 0.9, 0, 0,
0, 0, 0, 0, 0, 0, 1.1, 0.9, 0.9, 0.9, 0.9, 0.9, 0.9, 0.9, 0.9, 0.9, 0.9, 0.9, 0.9, 0, 0, 0, 0, 0, 0, 0, 0,
0]
met_min_high: 7
met_min_inactive: 6
met_min_low: 152
met_min_medium: 78

non_wear: 378
rest: 290
score: 91
score_meet_daily_targets: 78
score_move_every_hour: 100
score_recovery_time: 98
score_stay_active: 82
score_training_frequency: 100
score_training_volume: 99
steps: 7237
summary_date: '2019-08-15'
target_calories: 450
target_km: 9
target_miles: 5.5
timezone: 120
to_target_km: 2.9
to_target_miles: 1.7
total: 272
counter= 584

readiness
period_id: 0
score: 81
score_activity_balance: 100
score_previous_day: 64
score_previous_night: 70
score_recovery_index: 56
score_resting_hr: 100
score_sleep_balance: 80
score_temperature: 93
summary_date: '2019-08-16'
counter= 572

restful_periods
bedtime_end: '2020-08-12T16:50:02+02:00'
bedtime_start: '2020-08-12T16:37:02+02:00'
breath_average: 12.75
duration: 780
period_id: 1
summary_date: '2020-08-12'
timezone: 120
counter= 1

sleep
awake: 5550
bedtime_end: '2019-08-16T07:41:32+02:00'
bedtime_end_delta: 27692
bedtime_start: '2019-08-15T23:18:32+02:00'

bedtime_start_delta: -2488
breath_average: 12.75
deep: 5190
duration: 30180
efficiency: 82
hr_5min: [0, 57, 58, 58, 53, 52, 51, 51, 51, 51, 52, 55, 52, 50, 52, 52, 53, 58, 56, 56, 54, 51, 50, 51, 51, 51, 52, 51, 49, 50, 49, 52, 48, 47, 47, 49, 50, 47, 47, 0, 51, 49, 51, 50, 51, 50, 49, 48, 49, 48, 44, 45, 46, 47, 46, 45, 47, 46, 45, 44, 45, 45, 43, 44, 48, 46, 51, 48, 49, 54, 51, 51, 52, 51, 50, 52, 50, 0, 49, 50, 51, 52, 54, 48, 48, 48, 48, 48, 49, 49, 51, 53, 53, 52, 51, 46, 48, 48, 50, 46, 51]
hr_average: 52.375
hr_lowest: 43
hypnogram_5min:
'4444211111111211231111112222222222211224422322222222211222222222333444442224444432222222223332333344'
is_longest: 1
light: 15630
midpoint_at_delta: 12182
midpoint_time: 14670
onset_latency: 1080
period_id: 0
rem: 3810
restless: 42
rmssd: 61
rmssd_5min: [0, 40, 29, 27, 34, 46, 57, 50, 49, 53, 56, 56, 79, 67, 59, 68, 54, 41, 52, 40, 47, 54, 55, 55, 50, 49, 47, 67, 83, 67, 52, 68, 80, 84, 61, 49, 59, 78, 69, 0, 41, 55, 61, 66, 53, 58, 57, 50, 49, 61, 63, 68, 71, 60, 63, 75, 60, 65, 59, 78, 63, 82, 86, 96, 83, 80, 54, 78, 67, 40, 60, 53, 44, 52, 66, 58, 49, 0, 45, 42, 46, 39, 38, 52, 56, 55, 40, 43, 42, 60, 72, 63, 45, 47, 63, 89, 58, 73, 69, 86, 60]
score: 78
score_alignment: 88
score_deep: 96
score_disturbances: 67
score_efficiency: 76
score_latency: 91
score_rem: 60
score_total: 77
summary_date: '2019-08-16'
temperature_delta: -0.31
temperature_deviation: -0.31
timezone: 120
total: 24630
counter= 570

In [4]:

```
#Checking the length of the data
for i in data:
        print('The \'{}\' data contains {} entries.'.format(i, len(data[i])))
```
```
The 'activity' data contains 585 entries.
The 'readiness' data contains 573 entries.
The 'restful_periods' data contains 2 entries.
The 'sleep' data contains 571 entries.
```

Since there is a difference between the length of the sleep and activity data, I will need to perform some matching to connect the days to the right nights. This can be done based on 'summary_date', since there is one in both sections of the data.

Preprocessing

In the Oura API documentation (Ōura Health Oy, n.d.) under the header 'Data Types' it says that all variables that denote a duration are expressed in seconds. However, if you look under the header 'Activity' you will find that this is not true, because the variables: 'non-wear', 'rest', 'inactive', 'low', 'medium' and 'high', are all expressed in minutes. Since data is only as accurate as the least accurate variable, this means that it will be necessary to convert all the variables that are not expressed in minutes, to minutes.

The 'bedtime_start' and 'bedtime_end' variables are currently not in a workable format because they are presented using standard ISO 8601 notation, hence the digital time format is located in the centre of a string and will need to be extracted before we can use it. So, that is what will be done first to the 'bedtime_start' and 'bedtime_end' variables: extract the digital time formats and convert it to minutes.

In [5]:

```
def time_min_conversion(time_data):
        #This function is designed to extract the digital time from the 'bedtime_start' and
'bedtime_end' variables and convert it to minutes.
   #Accordingly, time_data must be of the form string and can only have values
'bedtime_start' and 'bedtime_end'.
        try:
        #Container for the times in minutes
        times_m=[]

        #Performing steps for every entry in the 'sleep' data
        for i in data['sleep']:
        #Extracting the time out of the concatenated data
        time=re.split(r"[-T+:]", i[time_data])[3:6]

        #Transforming the time to minutes
        if int(time[2])<30:
                time_m=int(time[0])*60+int(time[1])
        else:
                time_m=int(time[0])*60+int(time[1])+1

        #Storing the time in the containers
        times_m.append(time_m)
```

```
    return times_m
        except:
        print('An error occurred.')
```

In [6]:

```
#The actual conversion
sleeping_times_m=time_min_conversion('bedtime_start')
rising_times_m=time_min_conversion('bedtime_end')
```

As just mentioned, I will also have to compare dates. This can be done using the date function from datetime, which effectively transforms the 'summary_date' attribute into a variable. This function expects an input in the following format: (year, month, day), but it is stored in the format: year-month-day, therefore I will need to extract this data from the 'summary_date' attribute and put it into the right format:

In [7]:

```
def date_conversion(date2conv):
        #This function is designed to convert a date provided in string format
('year-month-day') to int format (year, month, day) so it can be used by the 'date' function
from datetime.
        try:
        #Extracting the date data from the string
        conv_date_str=re.split(r"[-]",date2conv)

        #Converting the string representations to integers
        conv_date_int=[int(i) for i in conv_date_str]

    return conv_date_int
        except:
        print('An error occurred.')
```

Now that we can convert a date from the dataset to the right datatype, we can compare them:

In [8]:

```
def date_difference(date1,date2):
        #This function is designed to calculate the number of days difference between two
given dates.
        #Hereby the earliest date needs to be provided first, otherwise a negative number will
be returned.
        try:
        #Converting the provided dates to comparable formats
        d1=date(date1[0], date1[1], date1[2])
        d2=date(date2[0], date2[1], date2[2])

    #Comparing the dates
        dif=d2-d1
```

```
    return dif.days
    except:
    print('An error occurred.')
```

With this function, we can determine the difference between the first entry in the dataset and the last entry in the dataset and compare this to the length of the dataset, to determine if any days are missing, and if so, how many:

In [9]:

```
dd=date_difference(date_conversion(data['sleep'][0]['summary_date']),
date_conversion(data['sleep'][-1]['summary_date']))+1
print('The difference between {} and {} (included) is {} days.\nThe dataset contains {} entries
in the \'sleep\' data.\nThis means that there are {} missing days in the \'sleep\' data,
assuming the dataset does not contain any duplicate dates.'
      .format(data['sleep'][0]['summary_date'], data['sleep'][-1]['summary_date'], dd,
len(data['sleep']), abs(len(data['sleep'])-dd)))
```

```
The difference between 2019-08-16 and 2021-04-13 (included) is 607 days.
The dataset contains 571 entries in the 'sleep' data.
This means that there are 36 missing days in the 'sleep' data, assuming the dataset does not
contain any duplicate dates.
```

In [10]:

```
def gaps():
    #This function is designed to find the number of gaps in the data and their respective
sizes, where a gap is one or more consecutive missing days.
    try:
    #Creating a counter for the number of gaps and a container for the sizes of the gaps
    counter=0
    diff=[]

    #Working through all the gaps
    for i in range(0,len(data['sleep'])-1):
    if date_difference(date_conversion(data['sleep'][i]['summary_date']),
                       date_conversion(data['sleep'][i+1]['summary_date']))>1:

        #Determining the size of the current gap
            diff.append(date_difference(date_conversion(data['sleep'][i]['summary_date']),
                               date_conversion(data['sleep'][i+1]['summary_date']))-1)

        #Determining the total number of gaps
            counter+=1

    return counter, diff
    except:
    print('An error occurred.')
```

In [11]:

```
def gaps_print():
        #This function is designed to print the results from the 'gaps' function in a nice format.
        try:
        c, d=gaps()
        if c==1:
        return print('There is {} gap in the data, comprising a total of {} days.'.format(c,
np.sum(d)))
        elif c>1:
        return print('There are {} gaps in the data, comprising a total of {} days.\nThe average
gap size is approximately {} days and the largest gap size is {} days.'.format(c, np.sum(d),
np.round(np.mean(d)), np.max(d)))
        else:
        return print('There are no gaps in the data.')
        except:
        print('An error occurred.')
```

In [12]:

```
#The actual printing
gaps_print()
There are 17 gaps in the data, comprising a total of 36 days.
The average gap size is approximately 2.0 days and the largest gap size is 16 days.
```

Since the number of gaps corresponds to the number we found earlier, we know for sure that there are no duplicate dates in the 'sleep' data, because otherwise the missing days and the number of days contained in the dataset would not have totalled to the number of days between the first and last day in the dataset.

Now, since not all data will be used from this large dataset, it is important to know which part of the data I am going to work with and whether this part contains any gaps. In order to determine the size of the part of the data that will be used, it is good practice to look at existing research and interventions.

In sleep research, a common component of an intervention is the use of a sleep diary (Morin & Espie, 2003). Most sleep therapists and physicians advise their patients with sleep problems to fill in a sleep diary over the course of a week or more, and then come back so they can draw a conclusion. On the internet these templates are very large in numbers. This would not be a problem if they were all the same, but that is not the case. Some templates denote a minimum duration of one week, others two weeks. Since the American Academy of Sleep Medicine is one of the most credible sources when it comes to sleep research (Journal of Clinical Sleep Medicine, 2020), I decided to base my research on the template they have put online (American Academy of Sleep Medicine, 2017). This template has a length of two weeks.

When a day is missing from the last two weeks of data, we will be working with data of the last fifteen days, with one missing day, instead of with data from the last fourteen days. Therefore, we need to determine a boundary for when this becomes too much. There is no boundary defined in the literature that specifies which range of days and nights is the most representative for your current sleep, hence I will now set that boundary to be: one day. This way I know the difference will not be too large, assuming the data only contains normal values. That means that there should be no more than one gap, and if there is a gap it should not be more than one day.

In [13]:

```
def last_gap_date():
        #This function has been designed to determine the first date after the last gap in the
data.
   try:
        dd=0
        last_gap_date=''

      #Working through all the gaps
        for i in range(0,len(data['sleep'])-1):
        dd=date_difference(date_conversion(data['sleep'][i]['summary_date']),
date_conversion(data['sleep'][i+1]['summary_date']))-1

        #Determining the first date after the last gap in the data
        if dd>0:
                last_gap_date=data['sleep'][i+1]['summary_date']

      return last_gap_date
        except:
        print('An error occurred.')
```

In [14]:

```
def pre_last_gap_date():
        #This function has been designed to determine the first date after the second to last
gap in the data.
        try:
        c,d=gaps()
        if c>1:
          dd=0
        pre_last_gap_date=''

        #Working through all the gaps
        for i in range(0,len(data['sleep'])-1):
                dd=date_difference(date_conversion(data['sleep'][i]['summary_date']),
date_conversion(data['sleep'][i+1]['summary_date']))-1
                #Determining the previous gap
                if dd>0:
                c-=1
                #Determining the first date after the pre-last gap in the data
                if c==0:
                pre_last_gap_date=data['sleep'][i+1]['summary_date']

        return pre_last_gap_date
        except:
        print('An error occurred.')
```

In [15]:

```
def safe_date_dif_last_gap_print():
        #This function has been designed to print whether it is safe to trust the advice based on this data, based on the size and location of the gaps with respect to the day of the last 'sleep' data-entry contained in the dataset.
        try:
        c,d=gaps()
        greatest_gap=np.max(d)
        if pre_last_gap_date():
        pldd=date_difference(date_conversion(pre_last_gap_date()), date_conversion(data['sleep'][-1]['summary_date']))+1
        else:
        pldd=15
        ldd=date_difference(date_conversion(last_gap_date()), date_conversion(data['sleep'][-1]['summary_date']))+1
        if pldd<=14:
        print('The missing data occurs within the last two weeks on more than one occasion. It is unwise to trust any advices based on this data.')
        elif pldd>14 and ldd<=14 and greatest_gap>1:
        print('The missing data occurs within the last two weeks on one occasion, but is larger than one day. It is unwise to trust any advices based on this data.')
        elif pldd>14 and ldd<=14 and greatest_gap<1:
        print('The missing data occurs within the last two weeks on one occasion, but is not larger than one day. It is safe to trust any advices based on this data.')
        else:
        print('The missing data does not occur within the last two weeks. It is safe to trust any advices based on this data.')
        except:
        print('An error occurred.')
```

In [16]:

```
#The actual printing
safe_date_dif_last_gap_print()
```

```
The missing data does not occur within the last two weeks. It is safe to trust any advice based on this data.
```

Checking for trends in the data based on Daylight Saving Time (DST)

Since this dataset comprises over one and a half year it might be worth checking if there are any trends over the year caused by DST, because this might have an influence on the sleep of the user and hence on the advice given. However, since there is a gap of over two weeks in the very first part of the data, I will only use the data after that large gap:

In [17]:

```
#Determining the difference between the day of the first 'sleep' data-entry and the date of the first day after the largest gap
dd=date_difference(date_conversion(data['sleep'][0]['summary_date']),
date_conversion('2019-09-05'))+1
print('The difference between {} and {} (included) is {}
days.'.format(data['sleep'][0]['summary_date'], '2019-09-05', dd))
```
The difference between 2019-08-16 and 2019-09-05 (included) is 21 days.

In order to visually inspect this I will need to plot the data:

In [18]:

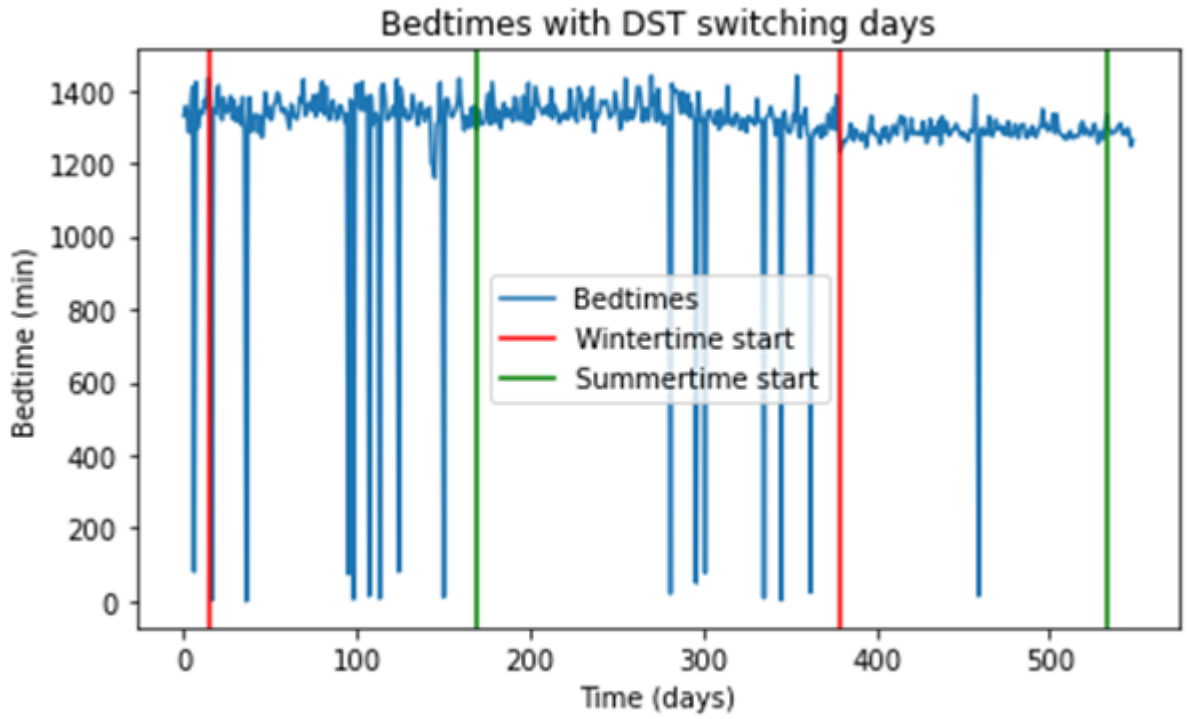
```
def plot(data_p1_x, data_p1_y, data_p2, title, axh, axv, col, spec_labs, xlab, xmajloc,
xticklabs, xlim, ylab, ymajloc, yticklabs, ylim, legend, figtext):
        #This function is designed to make it possible to plot a graph using just one line of
code.
    try:
        if col:
        colours=col
        linestyles=['solid']
        else:
        colours=['r', 'g', 'm']
        linestyles=['dotted', 'dashed', 'dashdot']

        fig,ax=plt.subplots(figsize=(7, 4))
        if not data_p1_y:
        ax.plot(data_p1_x)
        else:
        ax.plot(data_p1_x, data_p1_y)
        if(data_p2):
        ax.plot(data_p2)
        ax.set_title(title)
        if axh:
        for j in range(0,len(axh)):
                ax.axhline(axh[j], c=colours[j%len(colours)],
linestyle=linestyles[j%len(linestyles)])
        if axv:
        for k in range(0,len(axv)):
                ax.axvline(axv[k], c=colours[k%len(colours)])
        ax.set_xlabel(xlab)
        if spec_labs:
        if xmajloc:
                ax.xaxis.set_major_locator(matplotlib.ticker.FixedLocator(xmajloc))
                if xticklabs:
                ax.set_xticklabels(xticklabs)
        if ymajloc:
                ax.yaxis.set_major_locator(matplotlib.ticker.FixedLocator(ymajloc))
                if yticklabs:
```

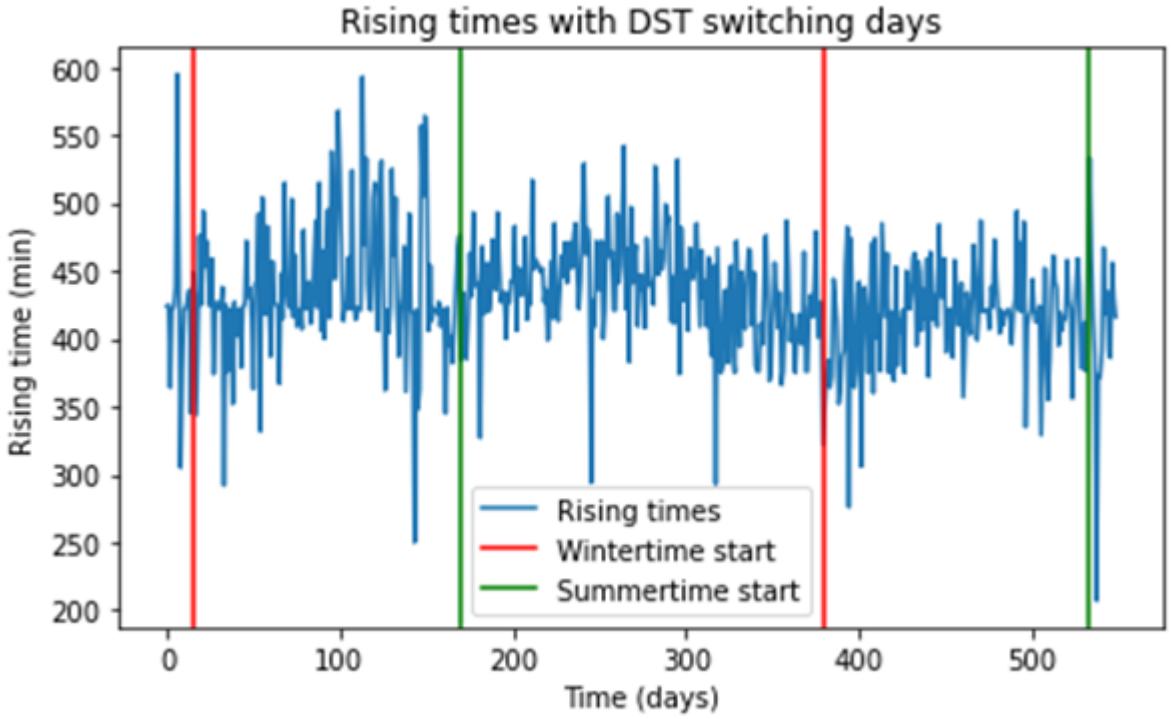
```
            ax.set_yticklabels(yticklabs)
    else:
    if xmajloc:
            ax.xaxis.set_major_locator(matplotlib.ticker.FixedLocator(xmajloc[0]))
            if xticklabs:
            ax.set_xticklabels(xticklabs[0])
    if ymajloc:
            ax.yaxis.set_major_locator(matplotlib.ticker.FixedLocator(ymajloc[0]))
            if yticklabs:
            ax.set_yticklabels(yticklabs[0])
    if xlim:
    ax.set_xlim(xlim)
    ax.set_ylabel(ylab)
    if ylim:
    ax.set_ylim(ylim)
    if legend:
    ax.legend(legend)
    if figtext:
    fig.text(figtext[0], figtext[1], figtext[2])
    plt.show()
    except:
    print('An error occurred.')
```

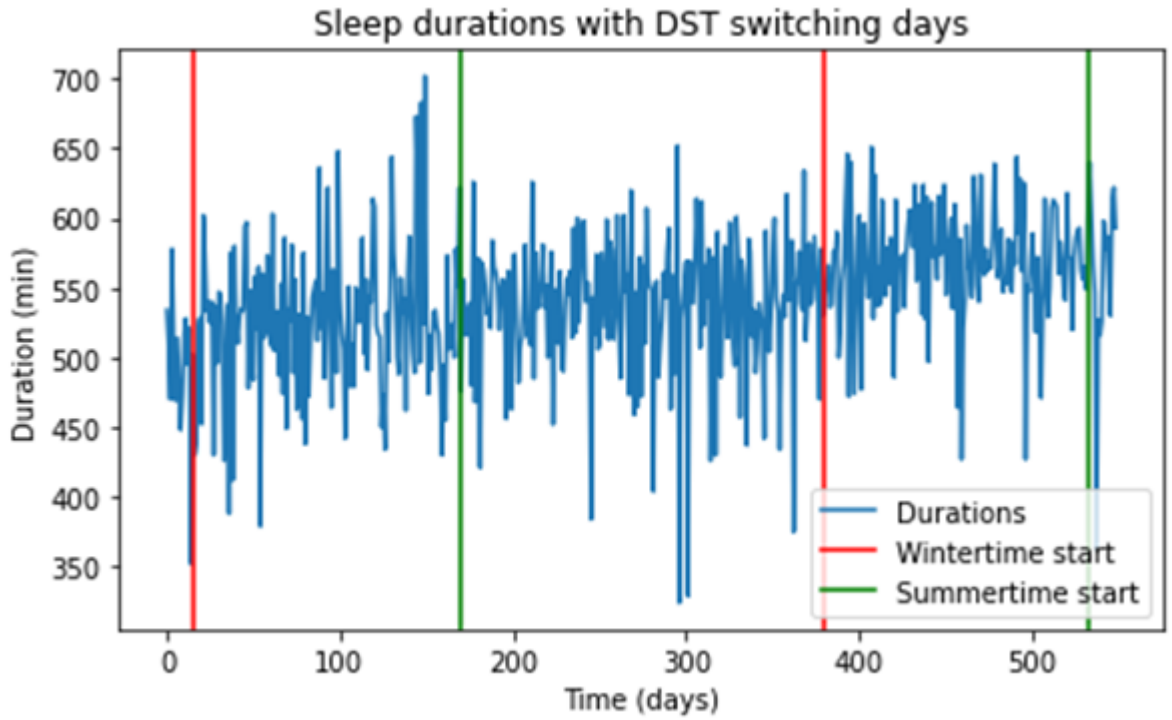
In [19]:

```
#Checking the date of the last day in the dataset
print('Last day in the dataset: {}'.format(data['sleep'][-1]['summary_date']))

#The following data is taken from Daylight Saving Time 2019 in the Netherlands (n.d.),
Daylight Saving Time 2020 in the Netherlands (n.d.) and Daylight Saving Time 2021 in the
Netherlands (n.d.), because the user lives in the Netherlands:
dst1='28-03-2021'
dst2='25-10-2020'
dst3='29-03-2020'
dst4='27-10-2019'

#Calculating the difference between the last day in the 'sleep' data and the DST switching
days
dd_dst1=date_difference(date_conversion('2021-03-28'),
date_conversion(data['sleep'][-1]['summary_date']))
dd_dst2=date_difference(date_conversion('2020-10-25'),
date_conversion(data['sleep'][-1]['summary_date']))
dd_dst3=date_difference(date_conversion('2020-03-29'),
date_conversion(data['sleep'][-1]['summary_date']))
dd_dst4=date_difference(date_conversion('2019-10-27'),
date_conversion(data['sleep'][-1]['summary_date']))
```

```
#Printing the DST switching days and their respective difference compared to the last day in the 'sleep' data
print('The summertime started at {}, which is {} days before the last day in the dataset.'.format(dst1, dd_dst1))
print('The wintertime started at {}, which is {} days before the last day in the dataset.'.format(dst2, dd_dst2))
print('The summertime started at {}, which is {} days before the last day in the dataset.'.format(dst3, dd_dst3))
print('The summertime started at {}, which is {} days before the last day in the dataset.'.format(dst4, dd_dst4))

#Plotting the sleeping times in minutes for the entire dataset and plotting the DST switching dates
plot(sleeping_times_m[21:-1], [], [], 'Bedtimes with DST switching days', [], [len(sleeping_times_m[21:-1])-dd_dst4, len(sleeping_times_m[21:-1])-dd_dst3, len(sleeping_times_m[21:-1])-dd_dst2, len(sleeping_times_m[21:-1])-dd_dst1], ['r','g'], False, 'Time (days)', [], [], [], 'Bedtime (min)', [], [], [], ['Bedtimes', 'Wintertime start', 'Summertime start'], [])

#Plotting the rising times in minutes for the entire dataset and plotting the DST switching dates
plot(rising_times_m[21:-1], [], [], 'Rising times with DST switching days', [], [len(rising_times_m[21:-1])-dd_dst4, len(rising_times_m[21:-1])-dd_dst3, len(rising_times_m[21:-1])-dd_dst2, len(rising_times_m[21:-1])-dd_dst1], ['r','g'], False, 'Time (days)', [], [], [], 'Rising time (min)', [], [], [], ['Rising times', 'Wintertime start', 'Summertime start'], [])

#Creating a list of sleep durations
durations_m=[]
for i in data['sleep']:
        durations_m.append(i['duration']/60)

#Plotting the sleep duration times in minutes for the entire dataset and plotting the DST switching dates
plot(durations_m[21:-1], [], [], 'Sleep durations with DST switching days', [], [len(durations_m[21:-1])-dd_dst4, len(durations_m[21:-1])-dd_dst3, len(durations_m[21:-1])-dd_dst2, len(durations_m[21:-1])-dd_dst1], ['r','g'], False, 'Time (days)', [], [], [], 'Duration (min)', [], [], [], ['Durations', 'Wintertime start', 'Summertime start'], [])
Last day in the dataset: 2021-04-13
The summertime started at 28-03-2021, which is 16 days before the last day in the dataset.
The wintertime started at 25-10-2020, which is 170 days before the last day in the dataset.
The summertime started at 29-03-2020, which is 380 days before the last day in the dataset.
The summertime started at 27-10-2019, which is 534 days before the last day in the dataset.
```


Bedtimes with DST switching days
Bedtime (min)
Time (days)
Bedtimes
Wintertime start
Summertime start

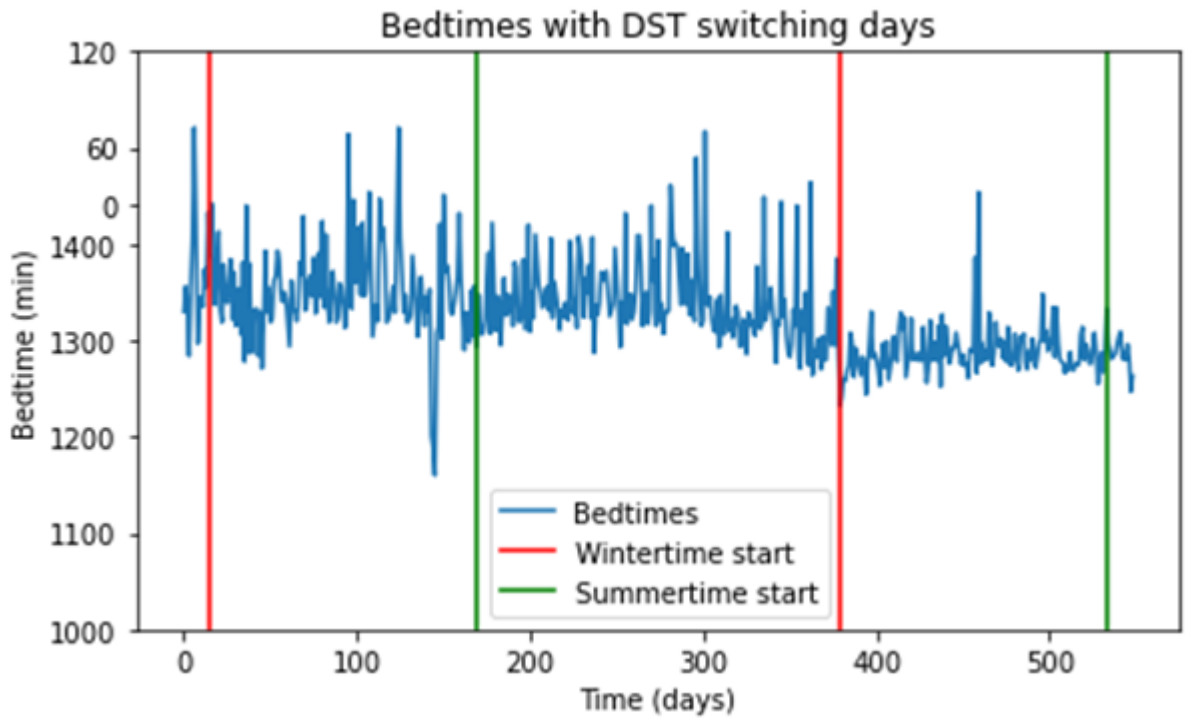

Rising times with DST switching days
Rising time (min)
Time (days)
Rising times
Wintertime start
Summertime start



The sleeping times data is currently not really visible due to the fact that some sleeping times are after midnight and therefore end up on the lower side of the spectrum. In order to improve visibility (and therefore my ability to detect a pattern in the data based on DST) I shall reshape the sleeping times data by putting these later bedtimes above the other data. In this reshaping I shall use 4am as a limit, because that is the time the Oura ring uses as well to differ between days, as can be seen in the Oura API documentation (Ōura Health Oy, n.d.) in the 'Activity' section under the information for the 'summary_date' attribute.

In [20]:

```
#Reshaping the sleeping_times_m data
r_sleeping_times_m=[]
for i in range(0,len(sleeping_times_m)):
        if sleeping_times_m[i]<(4*60):
        r_sleeping_times_m.append(sleeping_times_m[i]+(24*60))
        else:
        r_sleeping_times_m.append(sleeping_times_m[i])

#Plotting the reshaped sleeping times in minutes for the entire dataset and plotting the DST switching dates
plot(r_sleeping_times_m[21:-1], [], [], 'Bedtimes with DST switching days', [], [len(r_sleeping_times_m[21:-1])-dd_dst4, len(r_sleeping_times_m[21:-1])-dd_dst3, len(r_sleeping_times_m[21:-1])-dd_dst2, len(r_sleeping_times_m[21:-1])-dd_dst1], ['r','g'], True, 'Time (days)', [], [], [], 'Bedtime (min)', [1000, 1100, 1200, 1300, 1400, 1440, 1500, 1600, 1700, 1800], [1000, 1100, 1200, 1300, 1400, 0, 60, 120, 180, 240], [1000, 1600], ['Bedtimes', 'Wintertime start', 'Summertime start'], [])
```



No clear patterns emerge in bedtimes, rising times and sleep durations in this dataset based on DST. This is probably partially caused by the fact that this dataset comprises roughly one and a half year of data, which means that there are only two DST periods that we can compare. A positive trend can be seen in the sleep durations data, which means that the user has been getting more and more sleep over the past one and a half year, this is however, not related to DST.

The preprocessing has been done and no large trends have been found in the data that might be important to take into account during the data analysis, therefore we can get started on the actual analysis.

Basing Micro Interventions on Data

Wake at the same time every day

In order to determine the average wake time we need to know over which length of the data we need to take the average. During the preprocessing of the data we determine that we would use the last two weeks of data, based on the length of the sleep diary put online by the American Academy of Sleep Medicine (American Academy of Sleep Medicine, 2017).

Inspecting the data

When taking the average of a (part of the) dataset it is important to know the shape of the data distribution, because if the data is normally distributed and does not contain outliers, the mean is a very good representation of the average. If the data is skewed or contains outliers however, it is better to take the median instead. This can be best determined by plotting a histogram:

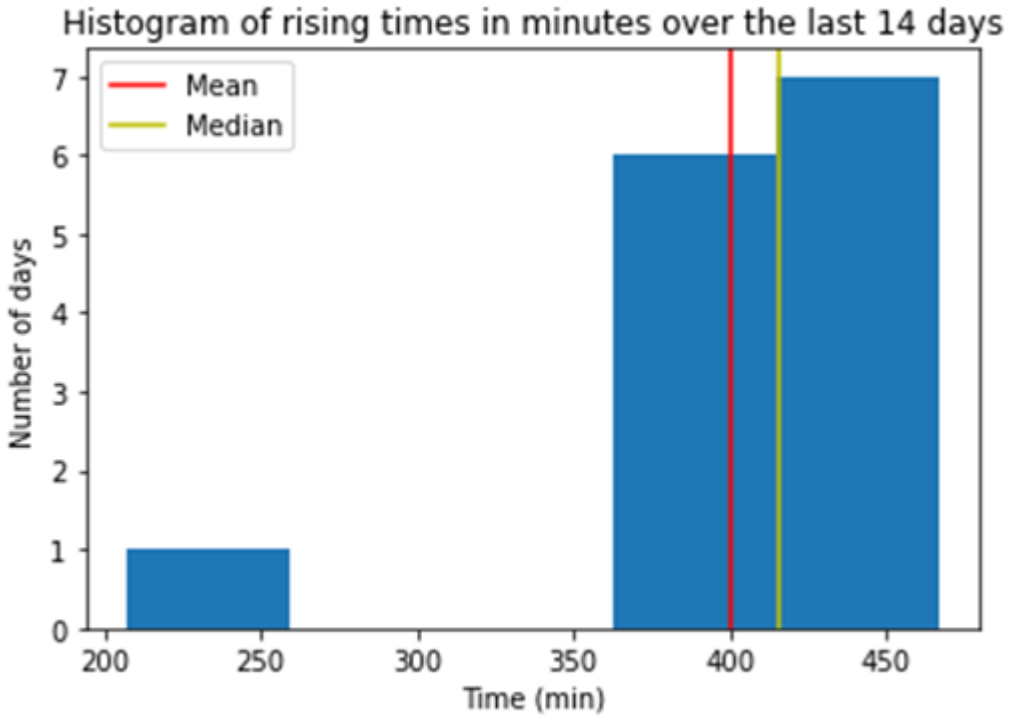
In [21]:

```
def histogram(data, bins, title, axv, xlab, ylab, legend):
```

```
        #This function is designed to make it possible to plot a histogram using just one line
of code.
        try:
        colours=['r','y']
        plt.hist(data,bins)
        plt.title(title)
        if axv:
        for i in range(0,len(axv)):
                plt.axvline(axv[i], c=colours[i])
        plt.xlabel(xlab)
        plt.ylabel(ylab)
        plt.legend(legend)
        plt.show()
        except:
        print('An error occurred.')
```

In [22]:

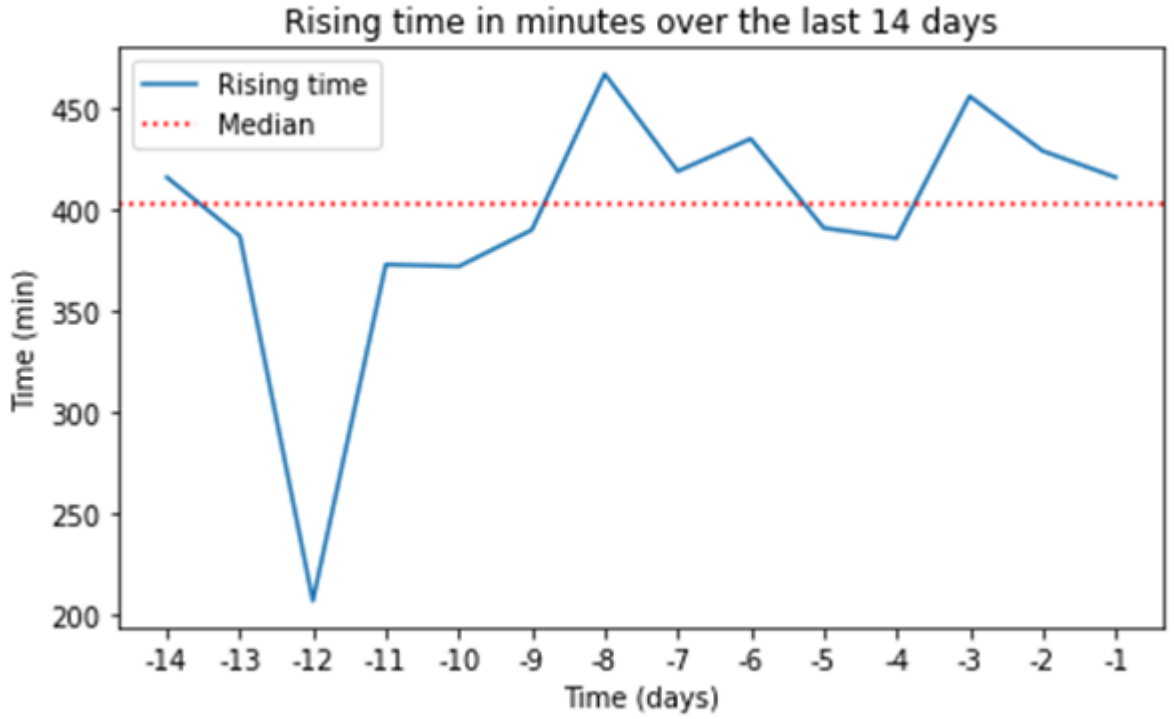
```
#Plotting a histogram of the data to see whether I should use the mean or the median
histogram(rising_times_m[-15:-1], 5, 'Histogram of rising times in minutes over the last 14
days', [np.mean(rising_times_m[-16:-1]), np.median(rising_times_m[-16:-1])], 'Time (min)',
'Number of days', ['Mean', 'Median'])
```



Since this data has an outlier on the left side of the histogram, the median is a better representative of the average.

In [23]:

#Plotting the rising times over the last two weeks and the average rising time in minutes
plot(rising_times_m[-15:-1], [], [], 'Rising time in minutes over the last 14 days',
[np.median(rising_times_m[-15:-1])], [], [], False, 'Time (days)', [np.arange(0,14)],
[np.arange(-14,0)], [], 'Time (min)', [], [], [], ['Rising time','Median'] ,[])



Let's compare this data to the data from the last month and the last week to determine if the two week boundary is justified:

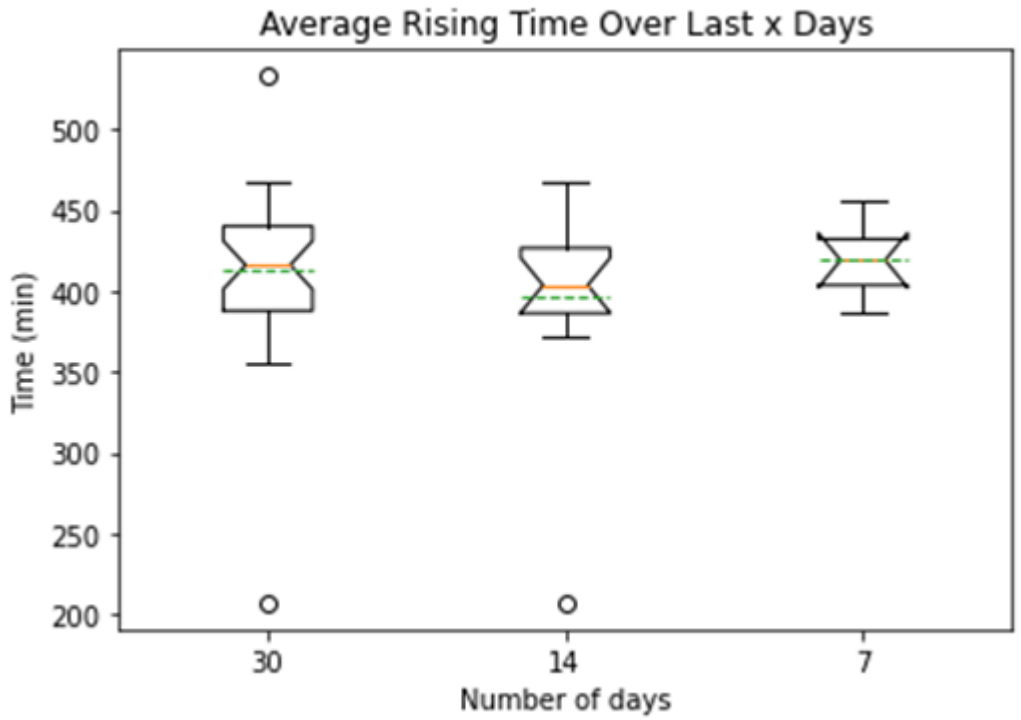
Plotting boxplots for comparison

In [24]:

```
def boxplots(data, labs, title, xlab, ylab):
    #This function is designed to make it possible to plot boxplots using just one line.
    plt.boxplot(data, labels=labs, notch=True, showmeans=True, meanline=True)
    plt.title(title)
    plt.xlabel(xlab)
    plt.ylabel(ylab)
    plt.show()
```

In [25]:

```
#Plotting the rising times in minutes as boxplots to create a better overview about
interquartile ranges, medians and confidence
#intervals
boxplots([rising_times_m[-31:-1], rising_times_m[-15:-1], rising_times_m[-8:-1]], ['30', '14',
'7'], 'Average Rising Time Over Last x Days', 'Number of days', 'Time (min)')
```



What can we determine from these boxplots?

- The data of the last month and the last two weeks contain outliers, whilst the data of the last week does not. (Denoted by the dots.)
- The mean of the data of the last two weeks lies a little lower than the means of the data of the last month and the data of the last week. (Denoted by the dotted green horizontal lines.)
- The median of the data of the last two weeks lies a little bit lower than the medians of the data of the last month and the data of the last week. (Denoted by the yellow horizontal lines.)
- The tapered lines going outwards from the medians are the 95% confidence intervals of the medians. When this concave area of a boxplot overlaps with the concavity of another boxplot, as is the case here for all boxplots, then the medians are not significantly different (Chambers et al., 1983).
- The minima and maxima of all data sets are quite similar, with the minimum of the data of the last month being slightly lower than the minimum of the data of the last two weeks, whilst their maxima are the same, and the minimum of the data of the last week is slightly higher and the maximum slightly lower than those of the data over the longer periods. (Denoted by the black horizontal lines.)

The fact that the mean from the data of the last two weeks lies a little lower than the mean from the data of the last week, makes sense, because the mean is known to be susceptible to outliers, which is the case for the data from the last two weeks and not for the data from the last week. For the data of the last month this is the same as for the data from the last two weeks, but it has an outlier on both sides, which cancel each other out almost perfectly, resulting in a mean that is almost the same as the median for the data of the last month.

Furthermore, the medians of the different partitions of the data are not significantly different, meaning that we could have used each of them for determining the ideal wake time. With a larger partition however, we do increase the risk of having gaps in the partition as well as the risk of missing out on changes in sleep behavior, which are clearly captured by the median of the data of the last two weeks. Therefore I will keep on using two weeks.

Determining the advice

In [26]:

```
def wake_time():
        #This function is designed to determine the ideal wake time in minutes.
        try:
        #Determining the ideal wake time in minutes
        wake_time=int(np.median(rising_times_m[-15:-1]))

      return wake_time
        except:
        print('An error occurred.')
```

In [27]:

```
def wake_time_print():
        #This function is designed to print the advice for the ideal wake time, with the ideal
wake time in digital format.
        try:
        return print('The advised rising time is: {}.\nThis advice can be rounded to the nearest
quarter if wished for.'.format(time.strftime("%H:%M", time.gmtime(wake_time()*60))))
        except:
      print('An error occurred.')
```

In [28]:

```
#The actual printing
wake_time_print()
```

```
The advised rising time is: 06:43.
This advice can be rounded to the nearest quarter if wished for.
```

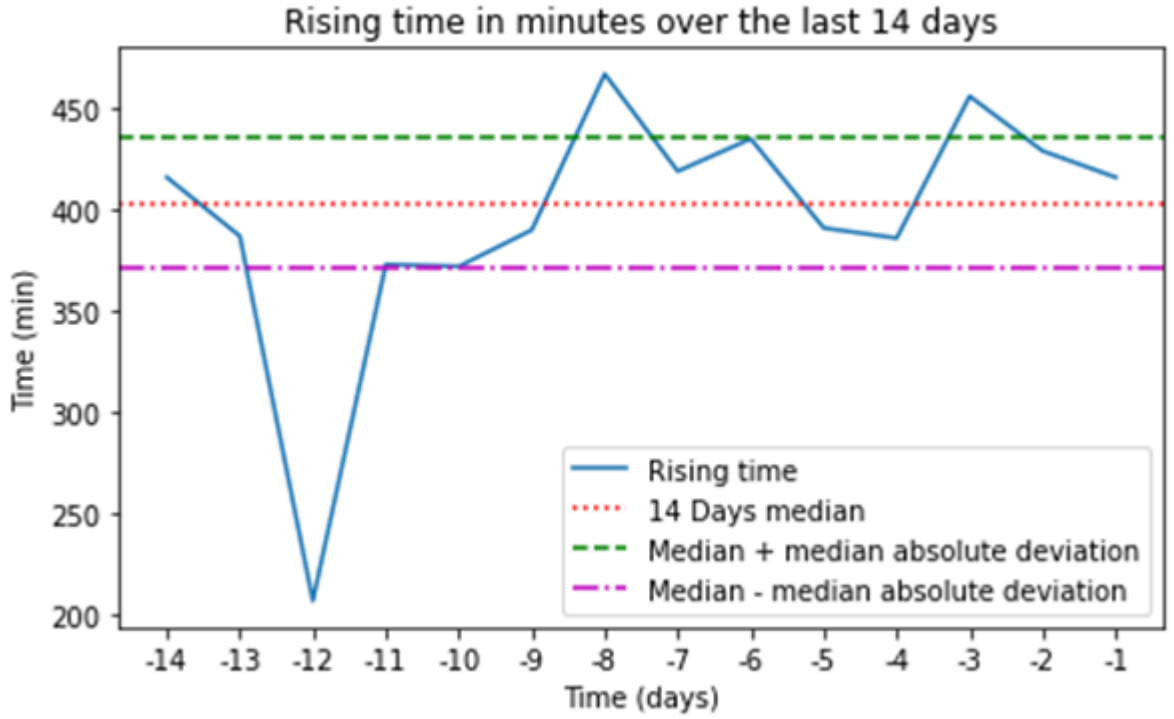
When should this advice be given?

To determine when this advice should be given, we should determine whether the user is waking up earlier or later than usual. In order to determine what is usual we should look at the standard deviation. However, the standard deviation is based on the mean, but we are using the median. The standard deviation of the median is the median absolute deviation, so let's plot these lines and see whether the user wakes up way earlier or later than usual given those boundaries.

In [29]:

```
#Plotting the standard deviation from the median
plot(rising_times_m[-15:-1], [], [], 'Rising time in minutes over the last 14 days',
[np.median(rising_times_m[-15:-1]),
np.median(rising_times_m[-15:-1])+robust.mad(rising_times_m[-15:-1]),
np.median(rising_times_m[-15:-1])-robust.mad(rising_times_m[-15:-1])], [], [], False, 'Time
```

(days)', [np.arange(0,14)], [np.arange(-14,0)], [], 'Time (min)', [], [], [], ['Rising time', '14 Days median', 'Median + median absolute deviation', 'Median - median absolute deviation'], [])



Sometimes the user wakes up earlier or later than usual indeed, so on these days it might be good to advice the user to wake up at the same time every day (again) and tell them which time that is.

In [30]:

```
def provide_wake_time():
        #This function is designed to determine whether the user should be advised to wake at the same time every day.
        try:
        rising_times_m14=[]

    #Converting the rising times to minutes for if that had not been done yet
        rising_times_m=time_min_conversion('bedtime_end')

    #Storing the rising times of the last two weeks
        for i in range(0, 14):
        rising_times_m14.append(rising_times_m[-15+i])

    #Determining the median of the rising times of the last two weeks
        med=np.median(rising_times_m14)

    #Determining the median absolute deviation for the rising times of the last two weeks

    mad=robust.mad(rising_times_m14)

    if rising_times_m14[-1]<(med-mad) or rising_times_m14[-1]>(med+mad):
        return True
```

```
        else:
        return False
        except:
        print('An error occurred.')
```

In [31]:

```
#Printing whether the advice should be provided to the user or not
print('The user should be adviced to wake up at the same time everyday:
{}'.format(provide_wake_time()))
```

```
The user should be adviced to wake up at the same time everyday: False
```

No Napping

The Oura ring has a built-in nap detection (Oura Team, 2021), where a nap is defined to need to pertain to the following requirements:

A nap needs takes between 15 minutes and 3 hours
The body needs to enter at least one sleep stage that isn't Awake

If the nap lasts longer than 3 hours it will be classified as a sleep period. This implementation has one one major disadvantage: The naps need to be confirmed by the user in the application in order for the naps to be registered. If not confirmed, they get stored as restfull periods, just like the naps that don't fulfill the requirements.

So in order to determine whether any naps have been taken we need to look at the restful periods as well as the sleeping periods. In the primary data inspection at the beginning of this document we have already determined that there are only two data entries in the 'restful_periods' data, so let's inspect them:

In [32]:

```
#Printing the 'restful_periods' data
for i in data['restful_periods']:
        print(i)
```

```
{'bedtime_end': '2020-08-12T16:50:02+02:00', 'bedtime_start':
'2020-08-12T16:37:02+02:00', 'breath_average': 12.75, 'duration': 780, 'period_id': 1,
'summary_date': '2020-08-12', 'timezone': 120}
{'bedtime_end': '2020-08-15T17:50:00+02:00', 'bedtime_start':
'2020-08-15T17:36:00+02:00', 'breath_average': 12.75, 'duration': 840, 'hr_average': 73.5,
'hr_lowest': 71, 'period_id': 1, 'rmssd': 33, 'summary_date': '2020-08-15', 'timezone': 120}
```

First of all, both these restful periods fall outside of the two weeks boundary, but even if they would not have, the duration of the first one is 780 seconds, which is equal to 13 minutes, and the duration of the second one is 840 seconds, which is equal to 14 minutes. Neither duration is long enough to be qualified as a nap.

Because naps longer than three hours are considered sleep periods, we also need to check that. The attribute 'period_id' in the 'sleep' data allocates a number to the sleep period within the same summary_date, according to the Oura API documentation (Ōura Health Oy, n.d.). Accordingly, the first

sleep period gets a period id of 0, because the first sleep period is always the one of the night before. Therefore, in order to discover naps, we have to find all periods with an id higher than 0.

There is one exception to this however, in which a period id of 1 is not a nap: It could occur that someone goes to sleep after midnight one day and then goes to sleep before midnight after waking up on that same day. This should not count as a nap, hence we have to inspect the data. First, I am going to take a look at the entire dataset, to see if there is a pattern in the period id's.

Inspecting the data

In [33]:

```
#Checking how many sleep periods have an id of 1 or more
counter=0
for i in data['sleep']:
        if i['period_id']>0:
        counter+=1
print('There are {} periods with an id of 1 or more.'.format(counter))
```

```
There are 15 periods with an id of 1 or more.
```

This is a small enough number to have a look at all of these periods and the periods preceding these periods. The latter might give an indication of why a sleep period could have gotten an id of 1 or more.

In [34]:

```
#Printing all the days with a sleep period of 1 or more, together with the date before for
control
for i in range(0, len(data['sleep'])):
        if data['sleep'][i]['period_id']>0:
        print('Sleep period id: {}, Date: {}, Bedtime start:
{}'.format(data['sleep'][i-1]['period_id'], data['sleep'][i-1]['summary_date'],
data['sleep'][i-1]['bedtime_start']))
        print('Sleep period id: {}, Date: {}, Bedtime start: {}\n'.format(data['sleep'][i]['period_id'],
data['sleep'][i]['summary_date'], data['sleep'][i]['bedtime_start']))
```

```
Sleep period id: 0, Date: 2019-08-16, Bedtime start: 2019-08-15T23:18:32+02:00
Sleep period id: 1, Date: 2019-08-18, Bedtime start: 2019-08-17T22:21:08+02:00

Sleep period id: 0, Date: 2019-09-07, Bedtime start: 2019-09-06T22:21:25+02:00
Sleep period id: 1, Date: 2019-09-08, Bedtime start: 2019-09-07T22:39:56+02:00

Sleep period id: 0, Date: 2019-09-28, Bedtime start: 2019-09-27T23:28:56+02:00
Sleep period id: 1, Date: 2019-09-29, Bedtime start: 2019-09-29T01:21:01+02:00

Sleep period id: 1, Date: 2019-09-29, Bedtime start: 2019-09-29T01:21:01+02:00
Sleep period id: 1, Date: 2019-09-30, Bedtime start: 2019-09-29T23:42:29+02:00

Sleep period id: 0, Date: 2019-10-30, Bedtime start: 2019-10-30T00:00:28+01:00
Sleep period id: 1, Date: 2019-10-31, Bedtime start: 2019-10-30T21:26:53+01:00
```

Sleep period id: 0, Date: 2020-01-12, Bedtime start: 2020-01-11T23:02:59+01:00
Sleep period id: 1, Date: 2020-01-13, Bedtime start: 2020-01-13T00:13:44+01:00

Sleep period id: 0, Date: 2020-03-30, Bedtime start: 2020-03-29T21:47:15+02:00
Sleep period id: 1, Date: 2020-03-31, Bedtime start: 2020-03-30T23:42:13+02:00

Sleep period id: 0, Date: 2020-05-23, Bedtime start: 2020-05-22T21:52:52+02:00
Sleep period id: 1, Date: 2020-05-25, Bedtime start: 2020-05-24T23:27:38+02:00

Sleep period id: 0, Date: 2020-06-29, Bedtime start: 2020-06-28T23:28:42+02:00
Sleep period id: 1, Date: 2020-06-30, Bedtime start: 2020-06-29T23:28:59+02:00

Sleep period id: 0, Date: 2020-07-18, Bedtime start: 2020-07-17T22:11:00+02:00
Sleep period id: 1, Date: 2020-07-19, Bedtime start: 2020-07-19T00:20:43+02:00

Sleep period id: 0, Date: 2020-08-02, Bedtime start: 2020-08-01T22:00:21+02:00
Sleep period id: 1, Date: 2020-08-03, Bedtime start: 2020-08-03T00:49:56+02:00

Sleep period id: 0, Date: 2020-09-20, Bedtime start: 2020-09-19T21:55:36+02:00
Sleep period id: 1, Date: 2020-09-21, Bedtime start: 2020-09-21T00:04:12+02:00

Sleep period id: 0, Date: 2020-09-29, Bedtime start: 2020-09-28T21:54:48+02:00
Sleep period id: 1, Date: 2020-09-30, Bedtime start: 2020-09-29T23:59:38+02:00

Sleep period id: 0, Date: 2020-10-07, Bedtime start: 2020-10-06T21:14:00+02:00
Sleep period id: 1, Date: 2020-10-08, Bedtime start: 2020-10-08T00:23:34+02:00

Sleep period id: 0, Date: 2020-10-22, Bedtime start: 2020-10-21T21:34:53+02:00
Sleep period id: 1, Date: 2020-10-23, Bedtime start: 2020-10-22T23:04:40+02:00

As can be seen in the data above, by looking at the period id's, the dates on which they occurred and the starting bedtimes associated with these sleep periods, there are four types of occassions in which a period gets the number one:

1. A day is skipped in the data, so either the user skipped a nights sleep or didn't wear the ring during that night, effectively creating a one day gap in the sleep data. (1st datapoint)
2. Randomly. (2nd datapoint)
3. The user went to bed before midnight on one day and then went to bed after midnight the day after, effectively creating a one day gap in the sleeping data. (3rd datapoint)
4. The user went to bed after midnight one day and then went to bed before midnight the day after, effectively going to sleep twice on the same day. (4th datapoint)

All other datapoints can be distributed under these four categories.

The fourth category was predicted, the others however, are unexpected. Given the fact that there are two types of occasions that create a gap in the data (first and third category) and both these occasions result in a period id of 1, I think it is safe to assume that this is an unintentional effect created by the Oura ring developers.

The random allocations of a number one to a period id could have been naps if it was not for the fact that the bedtime start is way too late.

Regardless of the cause of these seemingly faulty allocations, it is unwanted behavior for this research. Because of these unwanted behaviors, I decided to extend the period id check with a check for abnormal sleeping times:

In [35]:

```
#Checking for abnormal sleeping times
printed=0
for i in sleeping_times_m:
        if int(i)>=(7*60) and int(i)<(19*60):
        print(i)
        printed+=1
if not printed>0:
        print('No sleeping times found between 7am and 7pm.')
```

```
No sleeping times found between 7am and 7pm.
```

Besides that, no sleeping periods were found to have an id higher than 1, so it is safe to conclude that the user did not nap in the past one and a half years, whilst wearing the ring. For the determination of the advice however, just the last two weeks will be used again.

Determining the advice

To conclude whether no naps have been taken in the last two weeks, I need a system to check whether the date of the restful periods are within fourteen days of the last date in the dataset, which in an online application would be today or yesterday, depending on the time of applying.

In [36]:

```
def within_14_days_check(conv_date):
        #This function is designed to check if a date lies within 14 days of the day of the last
'sleep' data-entry contained in the dataset.
        try:
        #Getting the last date from the dataset
        ref_date=data['sleep'][-1]['summary_date']

        #Converting the dates for comparison
        conv_ref_date=date_conversion(ref_date)

      #Comparing the dates
        if date_difference(conv_date, conv_ref_date)>=0 and date_difference(conv_date,
conv_ref_date)<=14:
        return True
        else:
        return False
        except:
        print('An error occurred.')
```

Now that we have this system, the advice can be determined.

In [37]:

```
def no_napping():
        #This function is designed to determine whether any naps have been taken in the
last two weeks and returns a list of them.
        try:
        #Container for naps
        naps=[]

    #Converting sleeping times to minutes
        sleeping_times_m=time_min_conversion('bedtime_start')

    #Adding unconfirmed naps within the last two weeks
        for i in data['restful_periods']:
        if within_14_days_check(date_conversion(i['summary_date'])):
                if i['duration']>=(15*60) and i['duration']<(3*60*60):
                naps.append(i)

    #Adding confirmed naps within the last two weeks based on period id and unusual
sleeping times
        for j in range(0, len(data['sleep'])):
        if j>len(data['sleep'])-15:
                if data['sleep'][j]['period_id']>0 and sleeping_times_m[j]>=(7*60) and
sleeping_times_m[j]<(19*60):
                naps.append(data['sleep'][j])

    return naps
        except:
        print('An error occurred')
```

In [38]:

```
def no_napping_print():
        #This function is designed to print the advice for no napping
        try:
        #Checking if there are any naps and giving advise if any naps have been found
    if not len(no_napping())==0:
        return print('It looks like you have been taking naps lately. If you feel like your sleep
quality has suffered it might be better not to take naps anymore.\nThese are the naps:
{}'.format(no_napping()))
        else:
        return print('It looks like you have not been taking naps lately. This is healthy
behavior that is good for your sleep quality. Keep it up!')
        except:
        print('An error occurred')
```

In [39]:

```
#The actual printing
no_napping_print()
```
It looks like you have not been taking naps lately. This is healthy behavior that is good for your sleep quality. Keep it up!

When should this advice be given?

In [40]:

```
def provide_no_napping():
    #This function is designed to determine whether the user should be advised not to nap anymore.
    try:
    if not len(no_napping())==0:
    return True
    else:
    return False
    except:
    print('An error occurred.')
```

In [41]:

```
#Printing whether the advice should be provided to the user or not
print('The user should be advice not to take naps: {}'.format(provide_no_napping()))
```
The user should be advice not to take naps: False

Get out of bed when unable to sleep

For getting out of bed when unable to sleep we are going to look at three aspects:

1. The sleep onset latency: How long does the person stay awake before falling asleep after going to bed.\ 2. If and for how long the user woke up during the night and whether he stayed in bed or not.\ 3. How long the user was awake before going out of bed.

These aspects are all combined into a score: The sleep efficiency.

The most reliable way to determine whether the user stayed in bed or not is by self-report, so if an application is being created questions should be asked when it was detected that the user was awake for a prolonged period of time during the night. Then the user can specify his course of actions or the lack there off.

A little bit less reliable, but technological data collection method is the activity measurement. This data is collected in 5 minute intervals and has been separated into 5 classes:

0. Non-wear \ 1. Rest \ 2. Inactive \ 3. Low activity \ 4. Medium activity \ 5. High activity

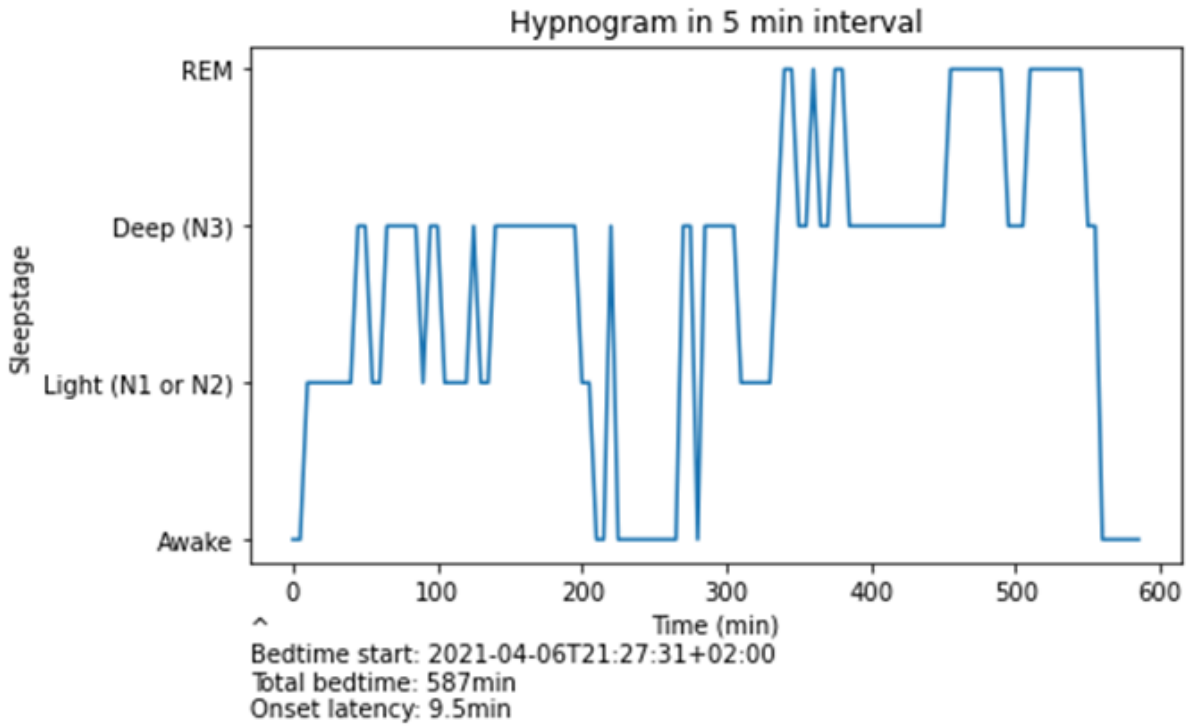
This data collection starts from 4am and lasts until 4am the next day, hence I'll have to combine the last bit of data from one day with the first bit of data from the next day to get the data for that night.

So first we have to find out when the user was awake, which is contained in the hypnograms:

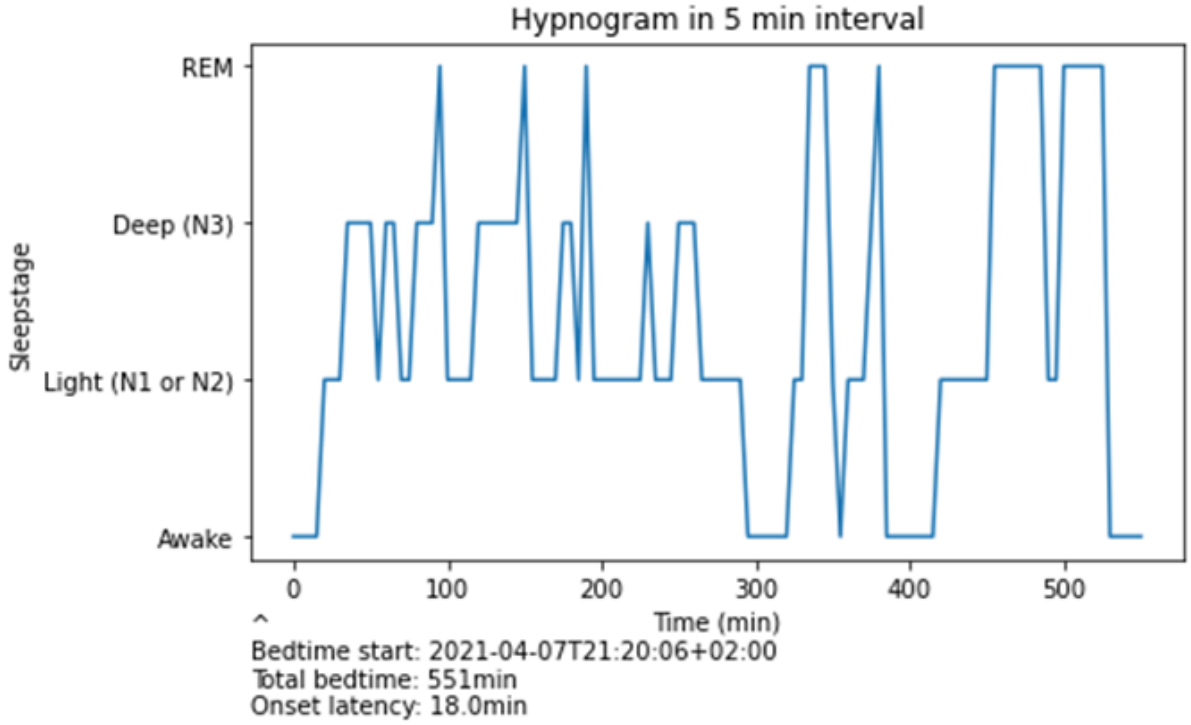
Inspecting the data

In [42]:

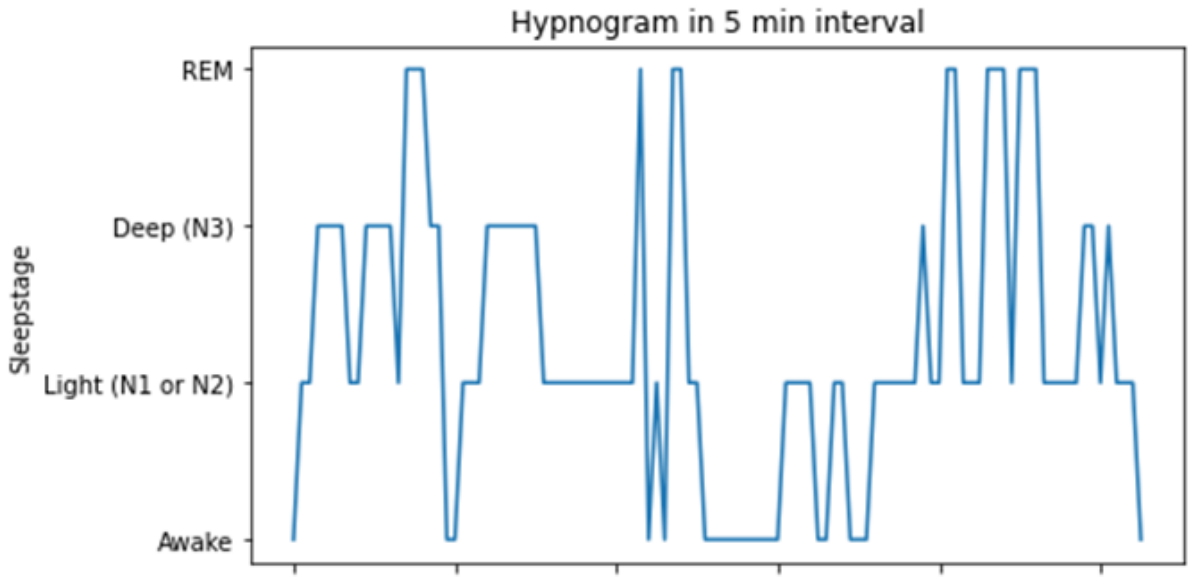
```
#Plotting the hypnograms for the last week
counter=-1
#Storing the hypnograms for later use
hgs5=[]
for i in data['sleep']:
        counter+=1
        hg5_conc=i['hypnogram_5min']
        hg5_split=[]
        if counter>(len(data['sleep'])-8):
        for j in hg5_conc:
        hg5_split.append(j)
        hgs5.append(hg5_split)
        plot(hg5_split, [], [], 'Hypnogram in 5 min interval', [], [], [], False, 'Time (min)',
[np.arange(0, 161, 20)],[np.arange(0, 801, 100)], [], 'Sleepstage', [np.arange(0, 4)], [['Awake',
'Light (N1 or N2)', 'Deep (N3)', 'REM']],[], [], [0.125, -0.1, '^\nBedtime start: {}\nTotal bedtime:
{}min\nOnset latency: {}min'.format(i['bedtime_start'], int(i['duration']/60),
i['onset_latency']/60)])
```



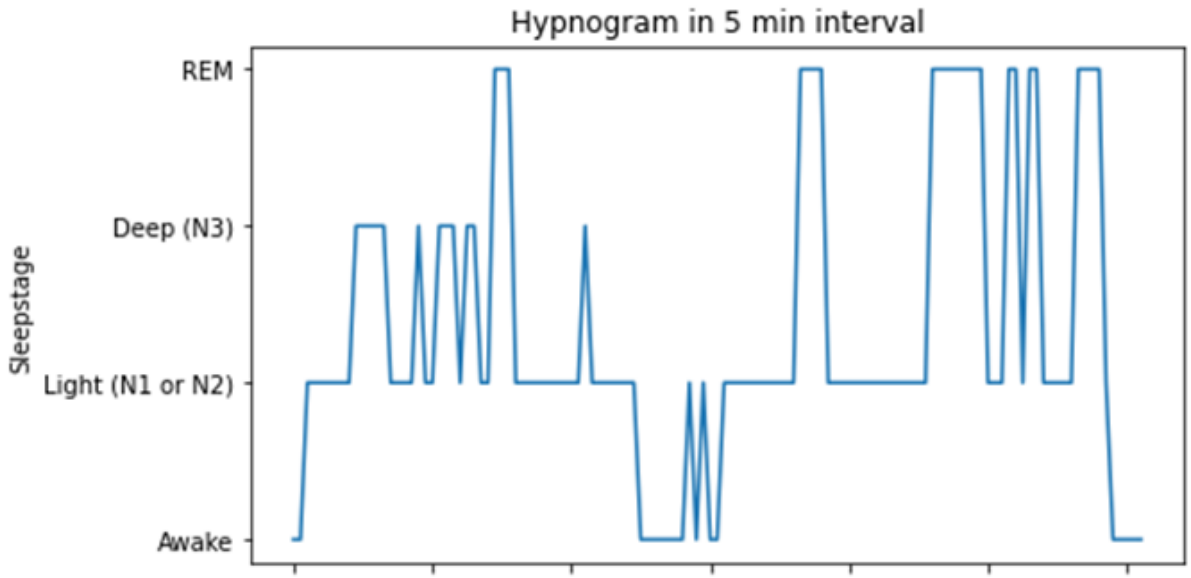
Bedtime start: 2021-04-06T21:27:31+02:00
Total bedtime: 587min
Onset latency: 9.5min



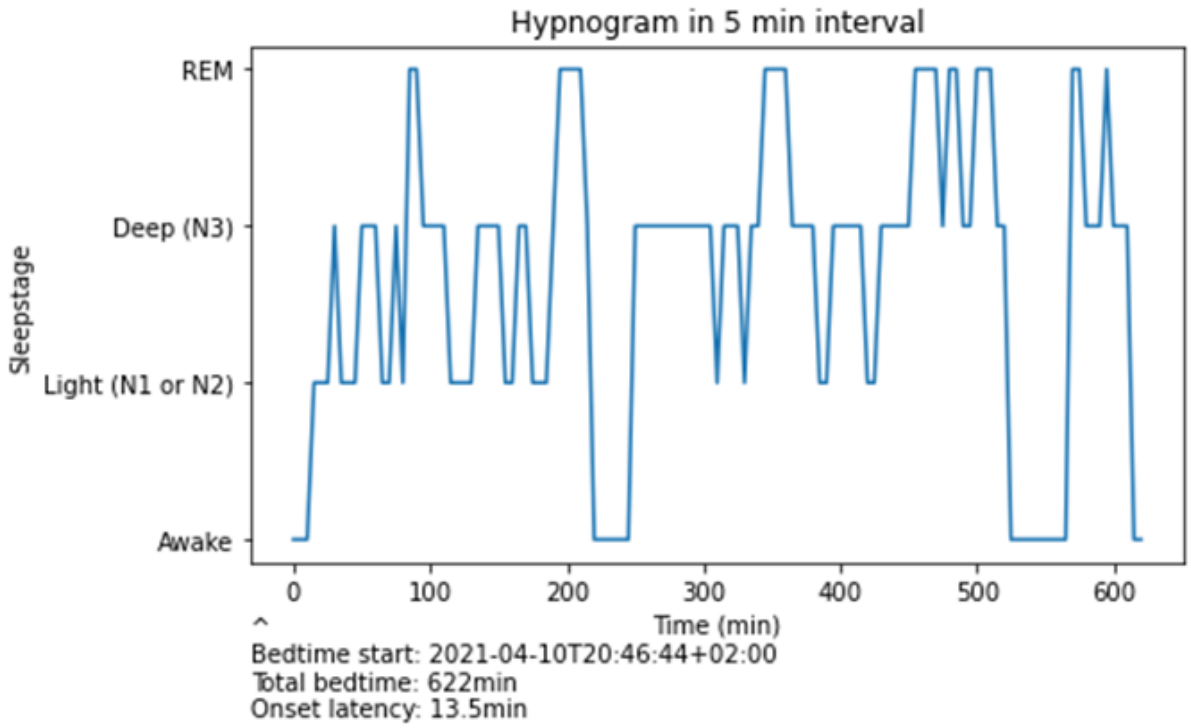
Bedtime start: 2021-04-07T21:20:06+02:00
Total bedtime: 551min
Onset latency: 18.0min



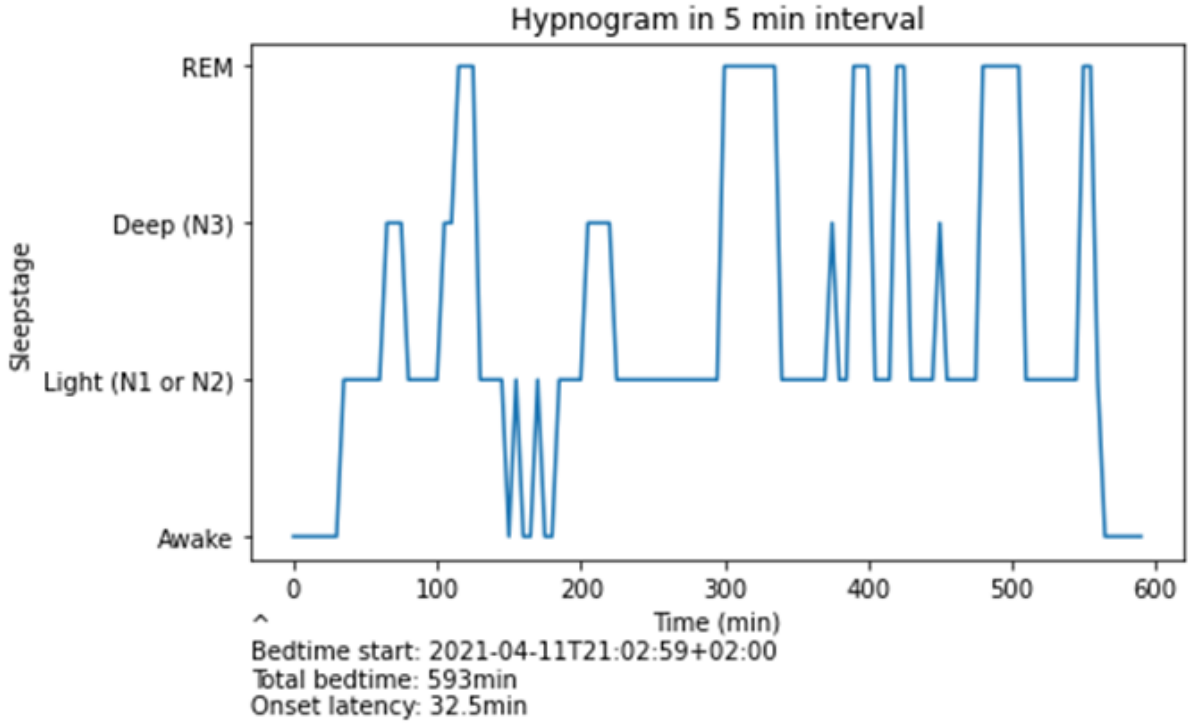
Bedtime start: 2021-04-08T21:36:04+02:00
Total bedtime: 530min
Onset latency: 6.0min



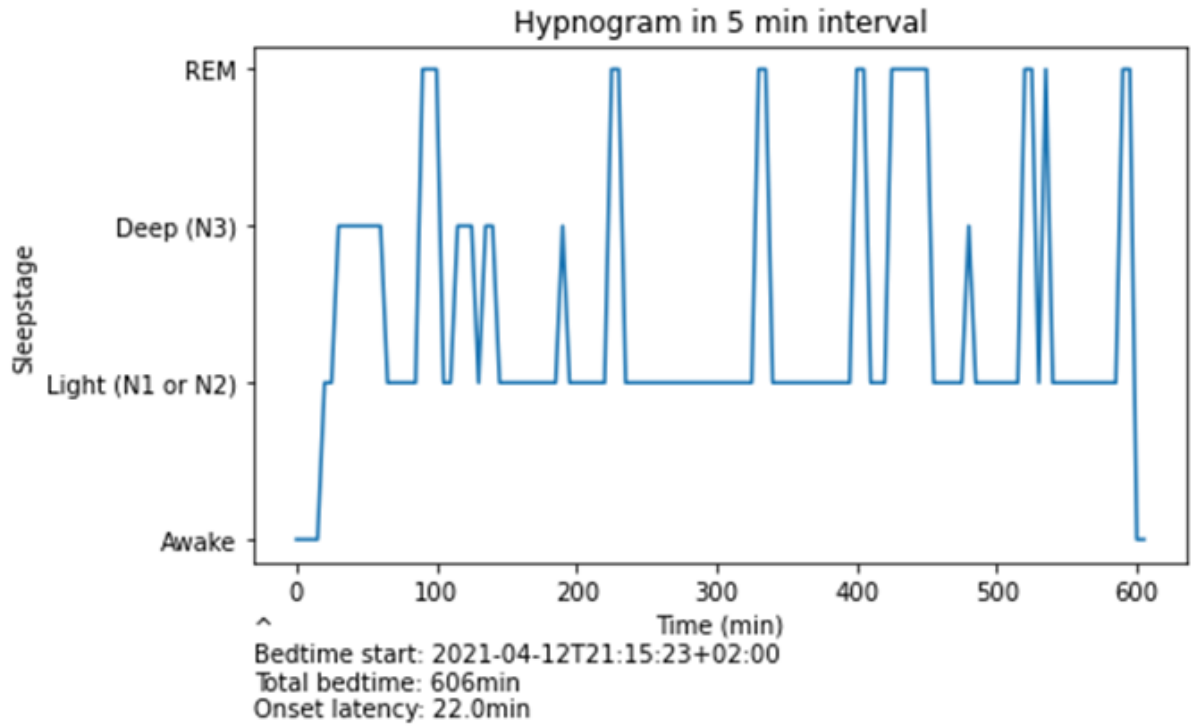
Bedtime start: 2021-04-09T21:21:30+02:00
Total bedtime: 614min
Onset latency: 11.0min



Bedtime start: 2021-04-10T20:46:44+02:00
Total bedtime: 622min
Onset latency: 13.5min



Bedtime start: 2021-04-11T21:02:59+02:00
Total bedtime: 593min
Onset latency: 32.5min



As can be seen in the Oura API documentation (Ōura Health Oy, n.d.), Oura defines a sleep onset latency (SOL) of less than 15 minutes to be good, 15 minutes as normal and more than 15 minutes problematic. Since we are using the Oura ring, this is the standard that I will be using as well.

For the last week in the dataset, the second night, sixth night and seventh night had problematic SOL's. The first five nights had one or more awakenings during the night that lasted longer than 15 minutes. The first, second, fourth and sixth night the user stayed in bed for longer than 15 minutes after waking up at the end of the night.

For the SOL it might be that the user was doing something that wasn't related to going to sleep. This is where sleep hygiene is important. We cannot know the cause of this except by self-report.

Inspecting full day activity spectra

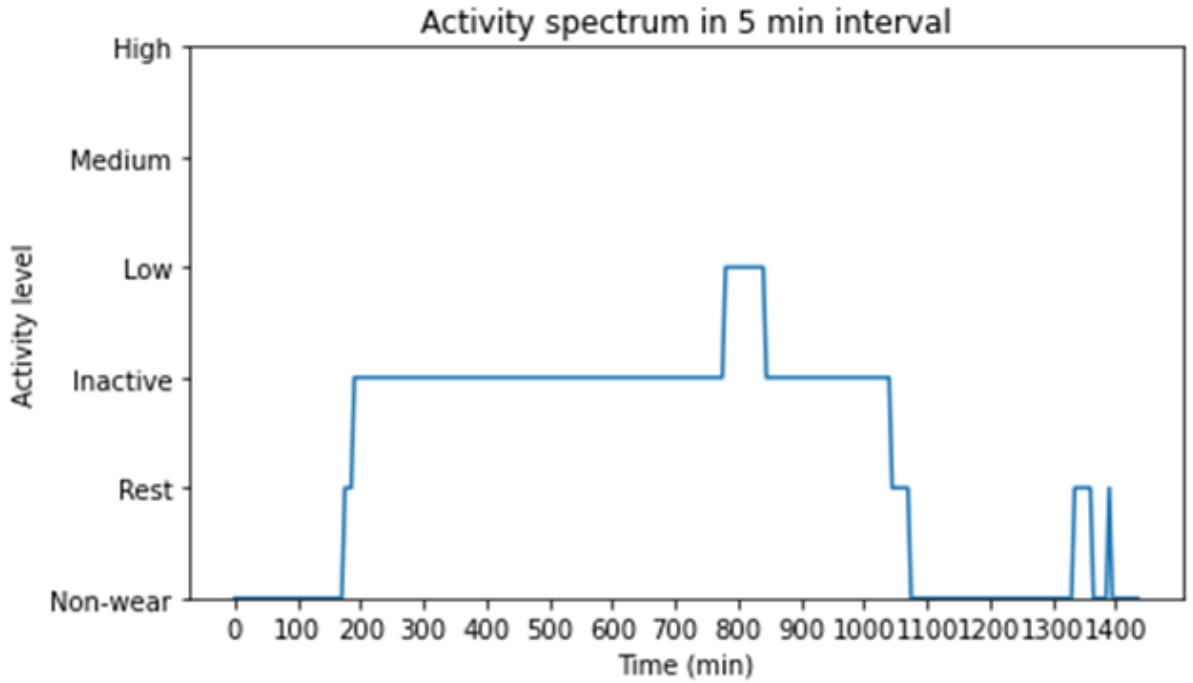
For the following activity spectra it is useful to understand the difference between the respective activity levels.\ According to the Oura API documentation (Ōura Health Oy, n.d.) the levels are defined as follows:

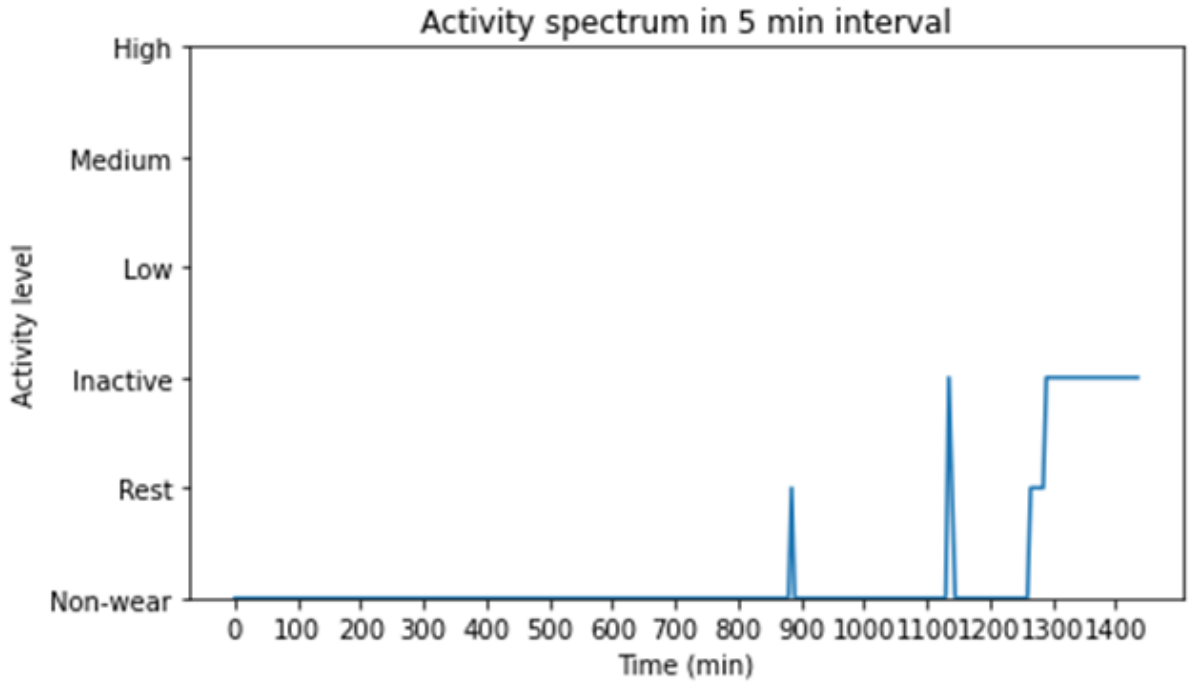
Metabolic-equivalent (MET) minutes express how much more energy the user is burning than if they were sitting still.

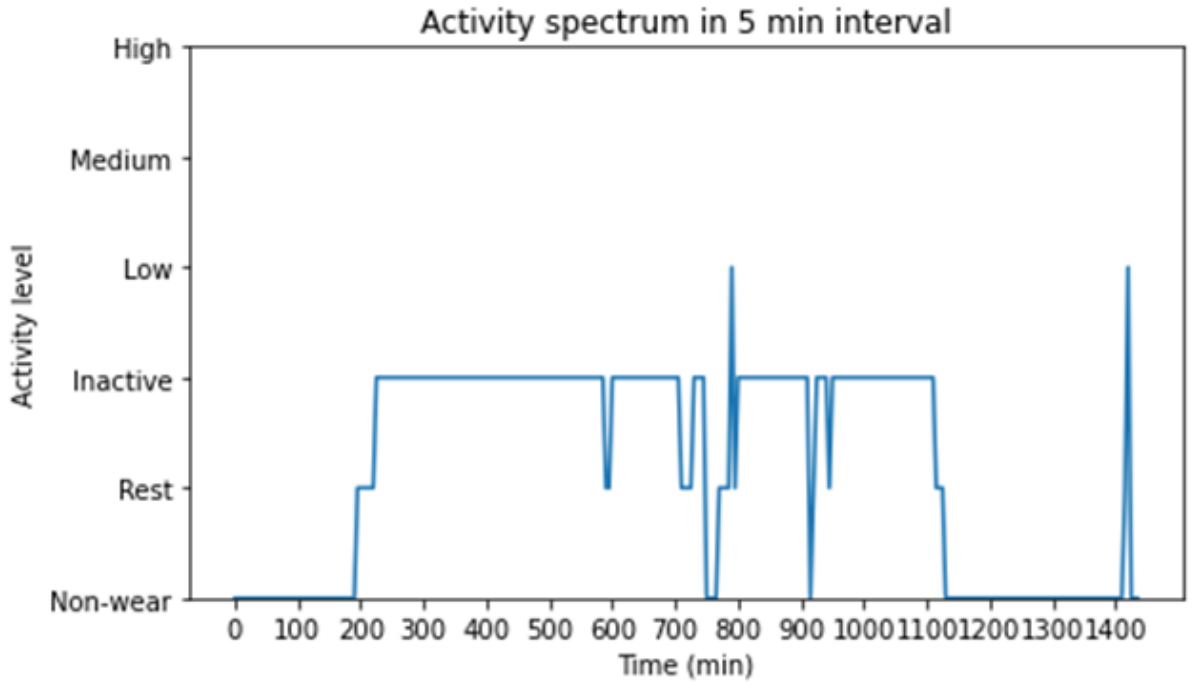
Non-wear: Not wearing the ring
Rest: Lying down or sleeping, corresponds to an average MET of <1.05.
Inactive: Sitting or standing still, corresponds to an average MET of 1.05 to 2.
Low: Low intensity activity, e.g. Household work, corresponds to an average MET of 2 to the age dependent limit.
Medium: Medium intensity activity, e.g. Walking, average MET level depends on age and gender.
High: High intensity activity, e.g. Running, average MET level depends on age and gender.

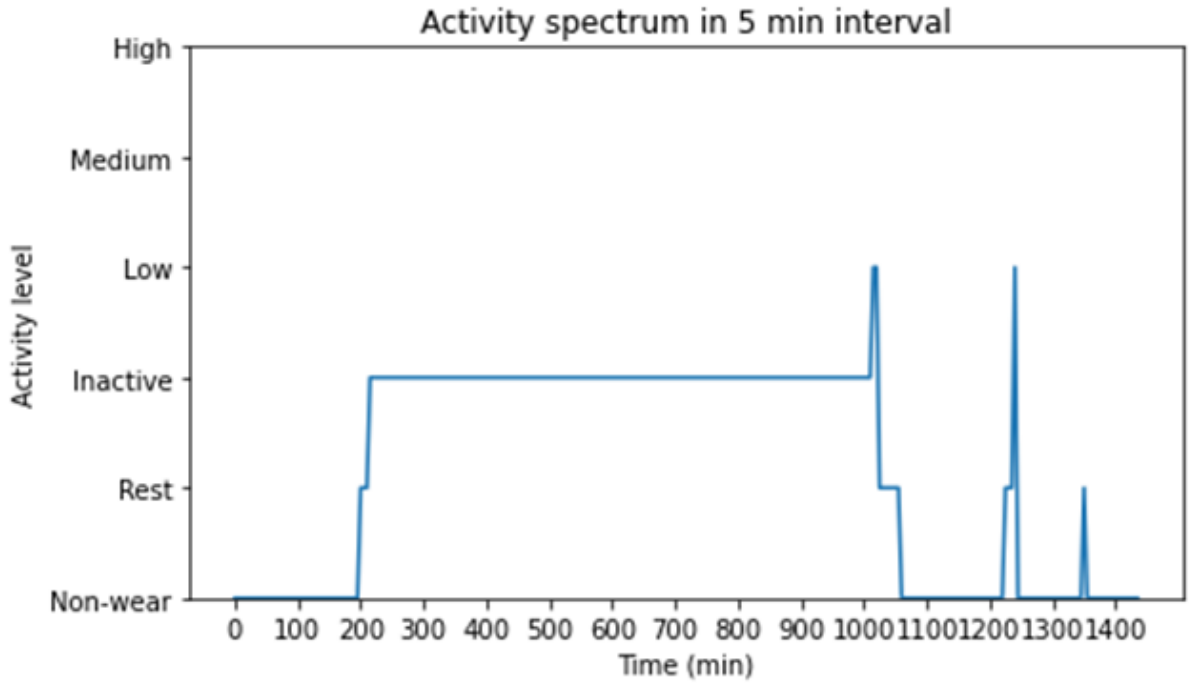
In [43]:

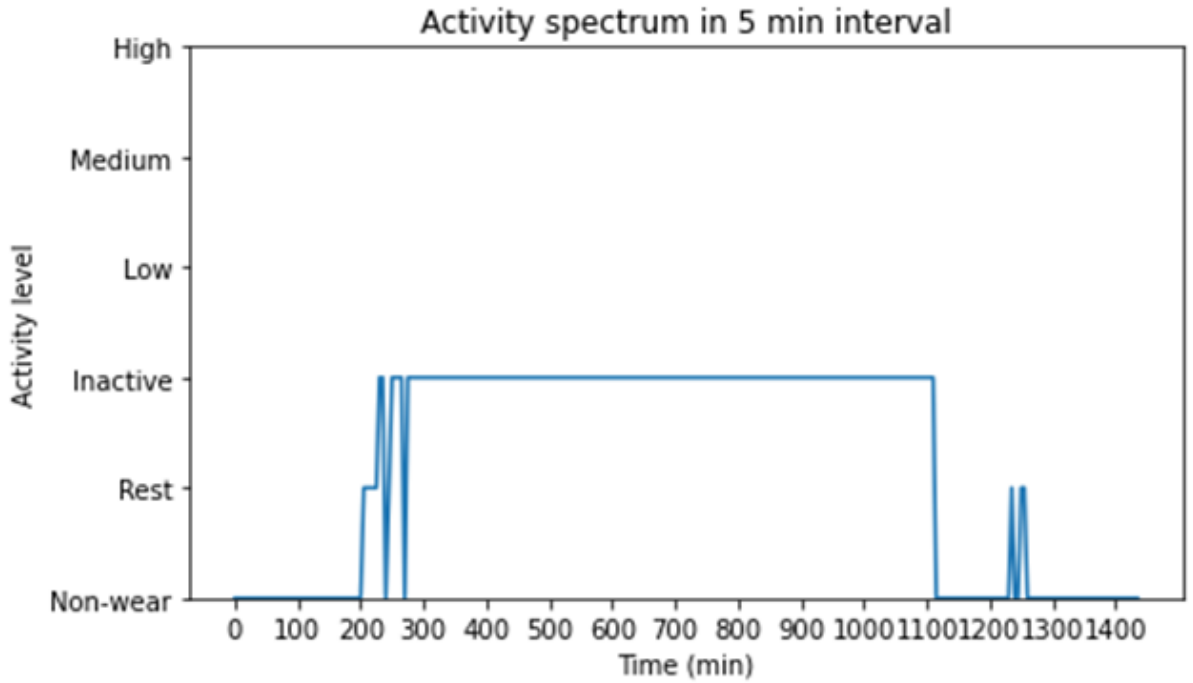
```
#Plot full activity spectra for some random datasamples, using a seed to ensure the samples
are always the same
np.random.seed(52)
samples=np.random.randint(0, len(data['activity']), 10)
for i in samples:
        temps=[]
        for j in data['activity'][i]['class_5min']:
        temps.append(j)
        plot(temps, [], [], 'Activity spectrum in 5 min interval', [], [], [], False, 'Time (min)',
[np.arange(0, 281, 20)],[np.arange(0, 1401, 100)], [], 'Activity level', [np.arange(0, 6)],
[['Non-wear', 'Rest', 'Inactive', 'Low', 'Medium', 'High']], [0, 5], [], [])
```

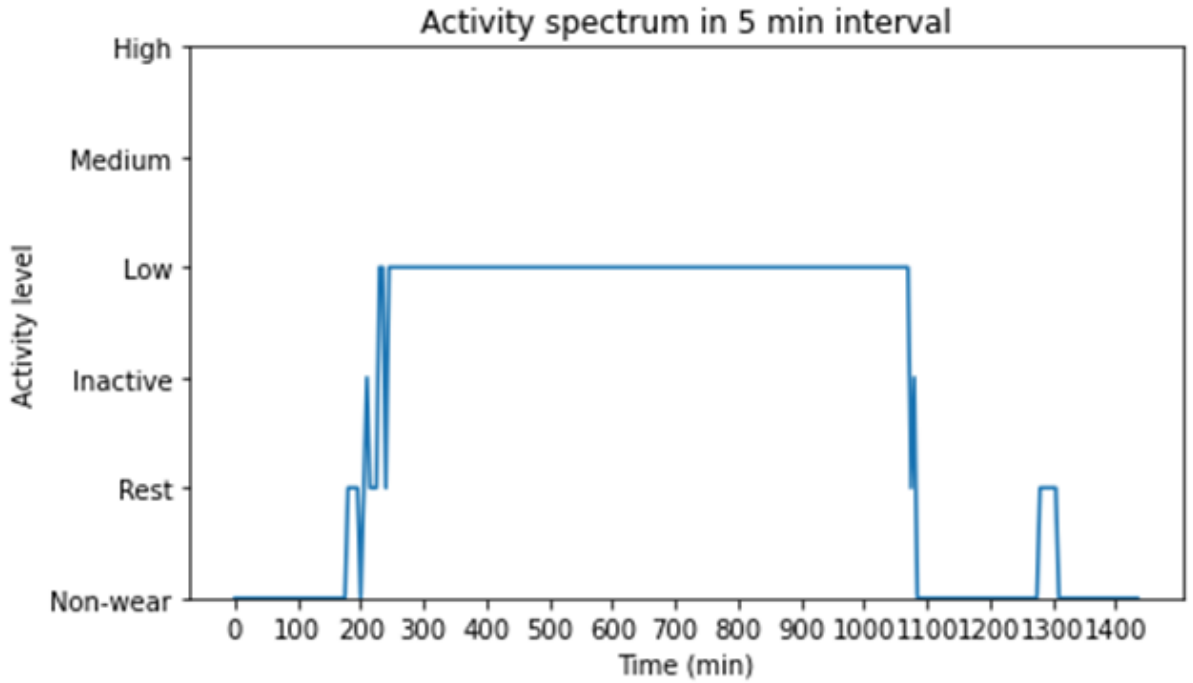

Activity spectrum in 5 min interval
High
Medium
Low
Inactive
Rest
Non-wear
Activity level
0 100 200 300 400 500 600 700 800 900 1000 1100 1200 1300 1400
Time (min)

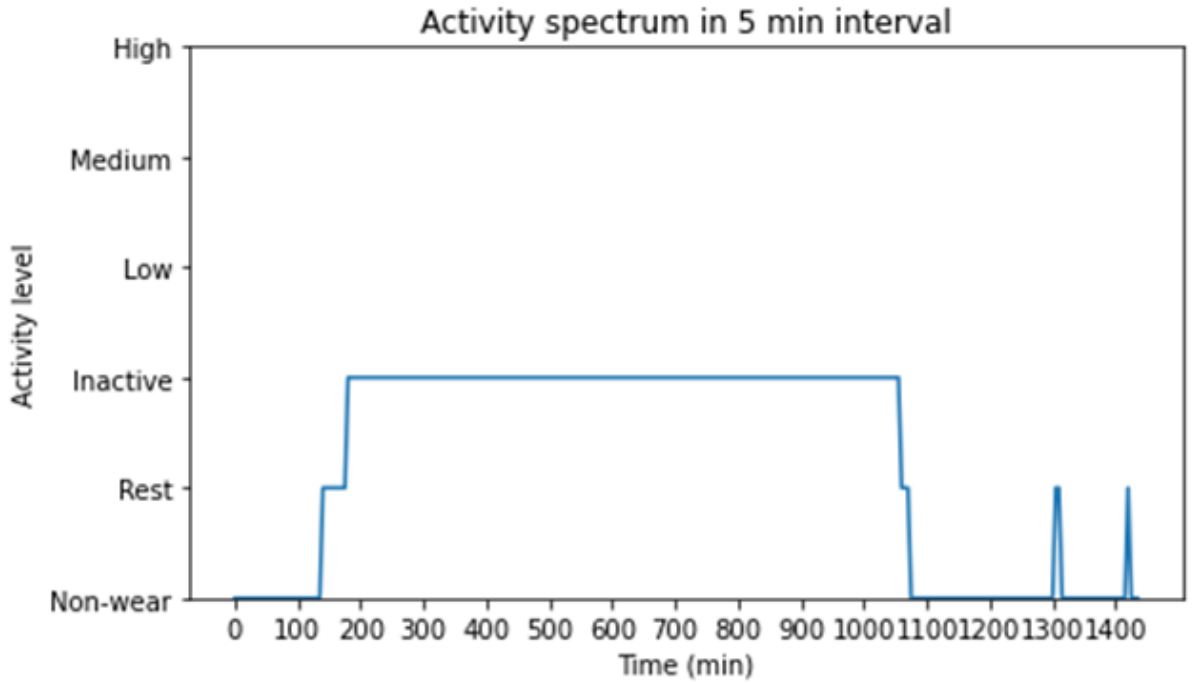

Activity spectrum in 5 min interval
High
Medium
Low
Inactive
Rest
Non-wear
Activity level
0 100 200 300 400 500 600 700 800 900 1000 1100 1200 1300 1400
Time (min)

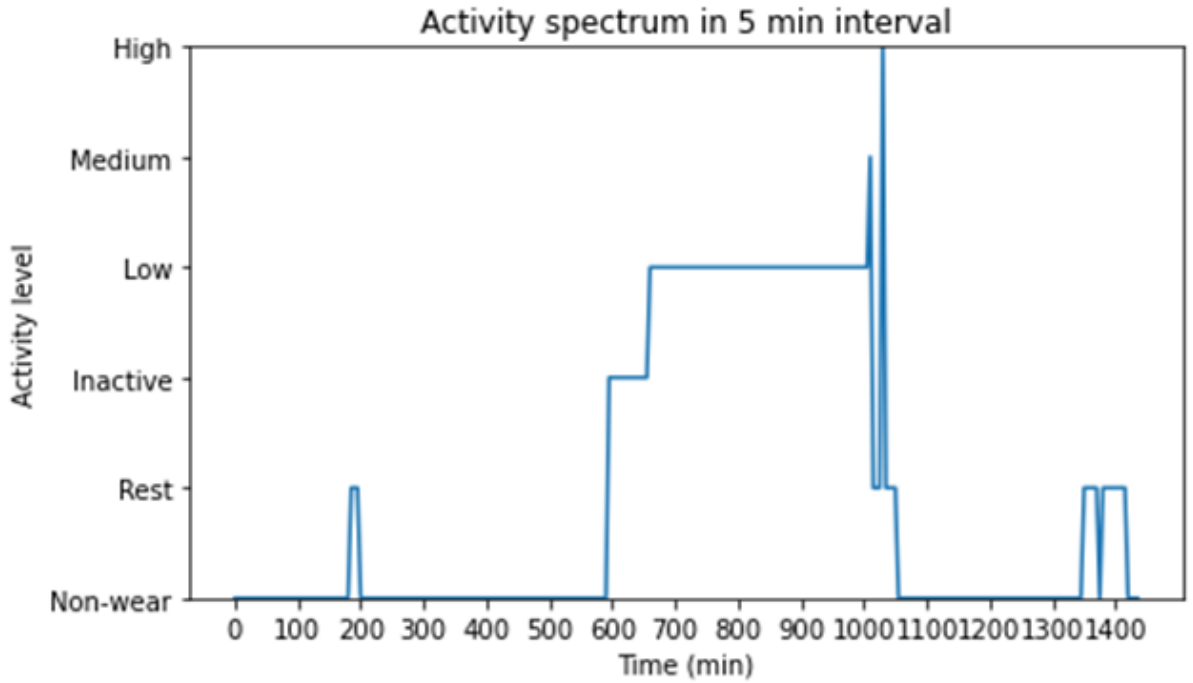

Activity spectrum in 5 min interval
High
Medium
Low
Inactive
Rest
Non-wear
Activity level
0 100 200 300 400 500 600 700 800 900 1000 1100 1200 1300 1400
Time (min)

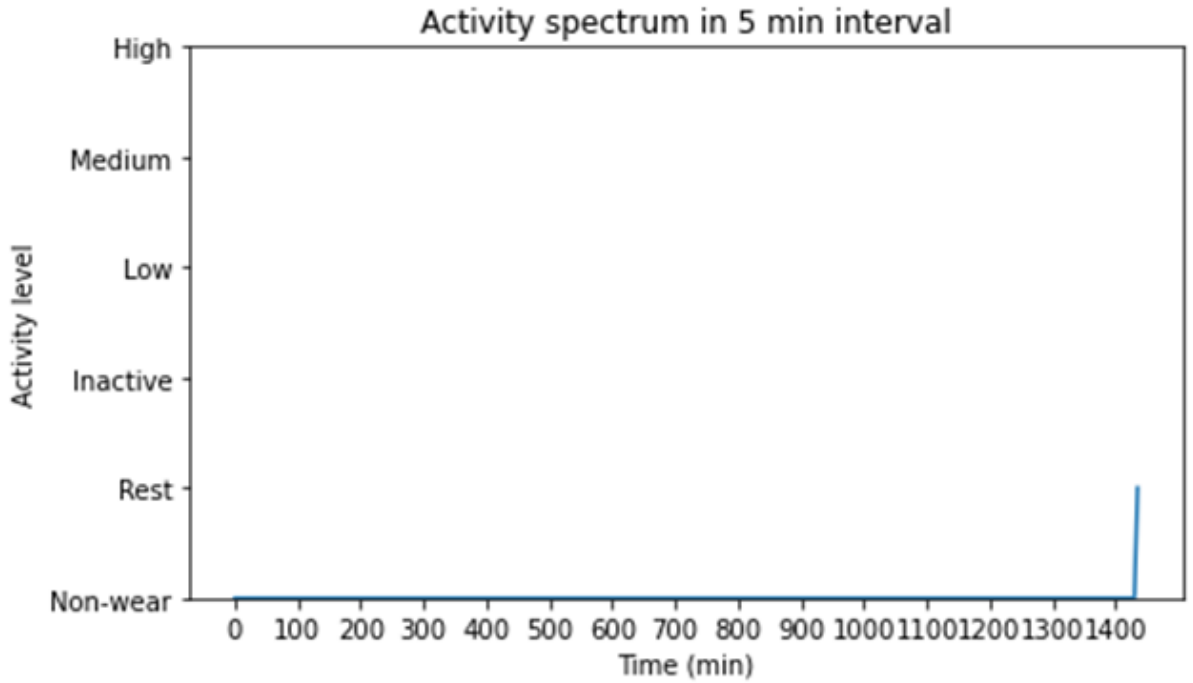

Activity spectrum in 5 min interval
High
Medium
Low
Inactive
Rest
Non-wear
Activity level
0 100 200 300 400 500 600 700 800 900 1000 1100 1200 1300 1400
Time (min)

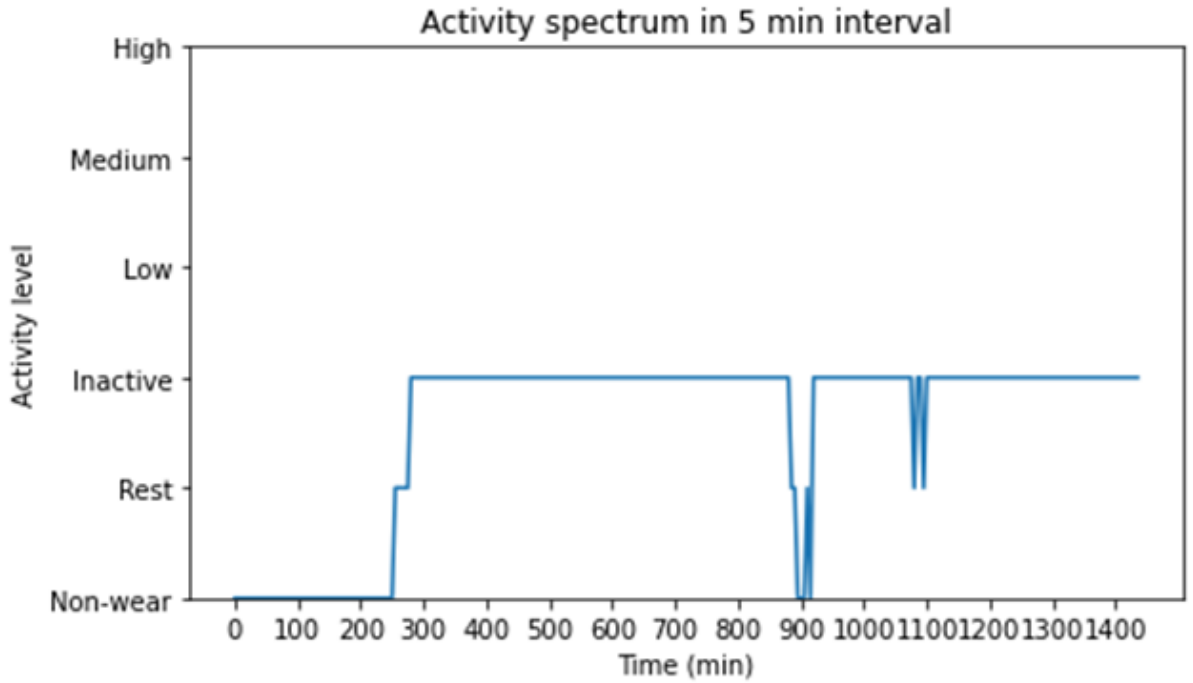

Activity spectrum in 5 min interval
High
Medium
Low
Inactive
Rest
Non-wear
Activity level
0 100 200 300 400 500 600 700 800 900 1000 1100 1200 1300 1400
Time (min)


Activity spectrum in 5 min interval
High
Medium
Low
Inactive
Rest
Non-wear
Activity level
0 100 200 300 400 500 600 700 800 900 1000 1100 1200 1300 1400
Time (min)


Activity spectrum in 5 min interval
High
Medium
Low
Inactive
Rest
Non-wear
Activity level
0 100 200 300 400 500 600 700 800 900 1000 1100 1200 1300 1400
Time (min)


Activity spectrum in 5 min interval
High
Medium
Low
Inactive
Rest
Non-wear
Activity level
0 100 200 300 400 500 600 700 800 900 1000 1100 1200 1300 1400
Time (min)





First of all, this data is surprising, due to the fact that the user hardly ever wears the Oura ring during the day, so where this data comes from is a mystery. Possibly though, this data is collected through another wearable or their phone, using another application that is linked to Apple Health, just like the Oura ring (Ōura Health Oy, n.d.).

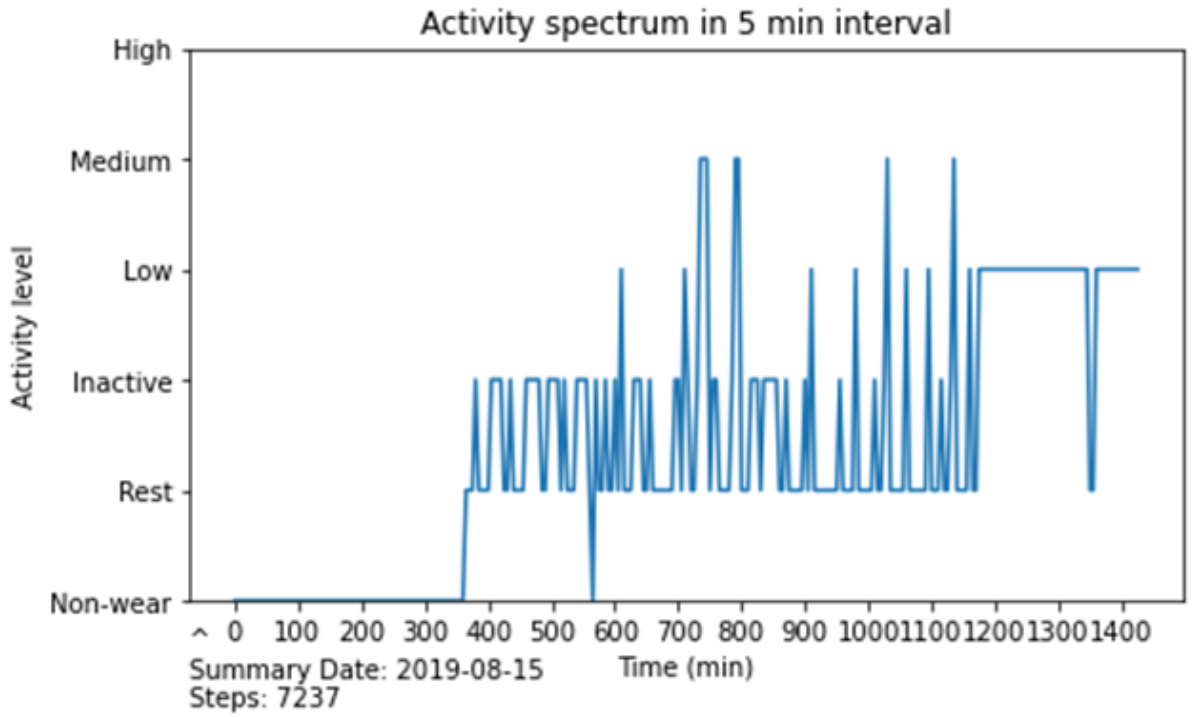
Using the data though, we can quickly see from the visualisation that the medium level is hardly ever reached and the High level even less. This makes the scale provided by Oura quite questionable, because the user is not disabled, meaning that the activity level associated with walking should be reached at least a few times during the day, which is not the case.

The Oura ring data has an attribute called: Steps, which contains the number of steps taken during a day. Possibly, looking at the activity spectra of a day where many steps have been taken can grant more insight in this matter.

In [44]:

```
steps=[]
for i in range(0, len(data['activity'])):
        steps.append(data['activity'][i]['steps'])

print('The most steps taken during a day is: {} steps.'.format(np.max(steps)))
temps=[]
for j in data['activity'][np.where(steps==np.max(steps))[0][0]]['class_5min']:
        temps.append(j)
plot(temps, [], [], 'Activity spectrum in 5 min interval', [], [], [], False, 'Time (min)',
[np.arange(0, 281, 20)], [np.arange(0, 1401, 100)], [], 'Activity level', [np.arange(0, 6)],
[['Non-wear', 'Rest', 'Inactive', 'Low', 'Medium', 'High']], [0, 5], [], [0.125, -0.02, '^\nSummary
Date: {}\nSteps:
{}'.format(data['activity'][np.where(steps==np.max(steps))[0][0]]['summary_date'],
data['activity'][np.where(steps==np.max(steps))[0][0]]['steps'])])
```

```
The most steps taken during a day is: 7237 steps.
```



Here we can see that even when the user walked quite some steps, the activity hardly ever gets to the medium level and sometimes not at all. Therefore, it is safe to assume that the attribution of Medium activity level to walking is unrealistic for this user, hence I can safely conclude that most of the walking activity done by this user is classified as Low level activity.

Aligning the activity spectra to the hypnograms

In [45]:

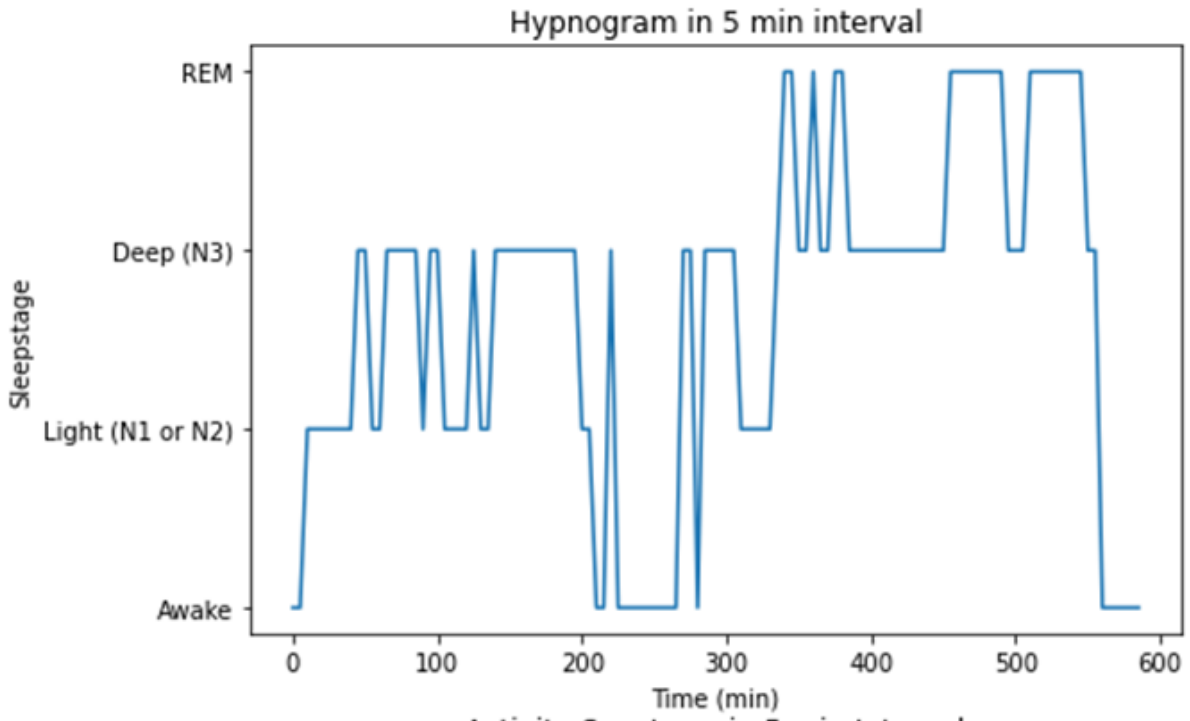
```
#Aligning the activity spectra for the nights during which the user was awake for more than
15 minutes in a row.
#Creating an empty container to store the combined activity class strings for each day
activity_strings=[]
for i in range(0,len(sleeping_times_m)):
        if i>=(len(data['sleep'])-8) and i<(len(data['sleep'])-1-2):
        #Creating an empty container to store the combined end and beginning of the activity
class strings
        int_ind=""

      #Calculate how much minutes from bedtime start to 4 am
        min_till_4am=((24*60)-sleeping_times_m[i]+(4*60))%1440
        #Divide in 5 minute intervals starting from the back
        last_intval_ind=int(np.floor(min_till_4am/5))

      #Matching the date from the sleep data to the date from the activity data
        match_ind=0
        for j in range(0, len(data['activity'])):
        if data['activity'][j]['summary_date']==data['sleep'][i]['summary_date']:
                match_ind=j

      #Check if one entry earlier is indeed one day earlier
        if(date_difference(date_conversion(data['activity'][match_ind-1]['summary_date']),
date_conversion(data['activity'][match_ind]['summary_date']))):

        #Add that part of the interval to the string
        int_ind+=(data['activity'][match_ind-1]['class_5min'][-last_intval_ind-1:-1])

        #Calculate how much minutes from 4 am to bedtime end
        min_from_4am=rising_times_m[i]-(4*60)
        #Divide in 5 minute intervals starting from the back
        first_intval_ind=int(np.floor(min_from_4am/5))

        #Add that part of the interval to the string
        int_ind+=(data['activity'][match_ind]['class_5min'][0:first_intval_ind])

        #Store the string in the container
        activity_strings.append(int_ind)

      else:
        print('Missing date in the activity data before the night to examine.')
```

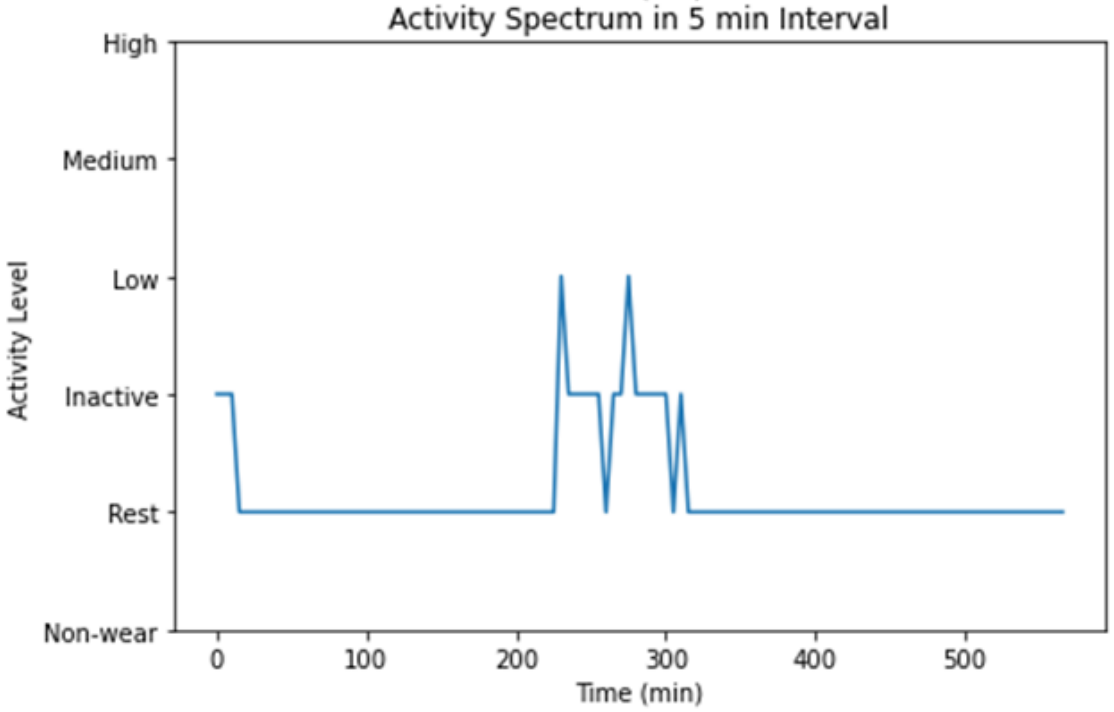
In [46]:

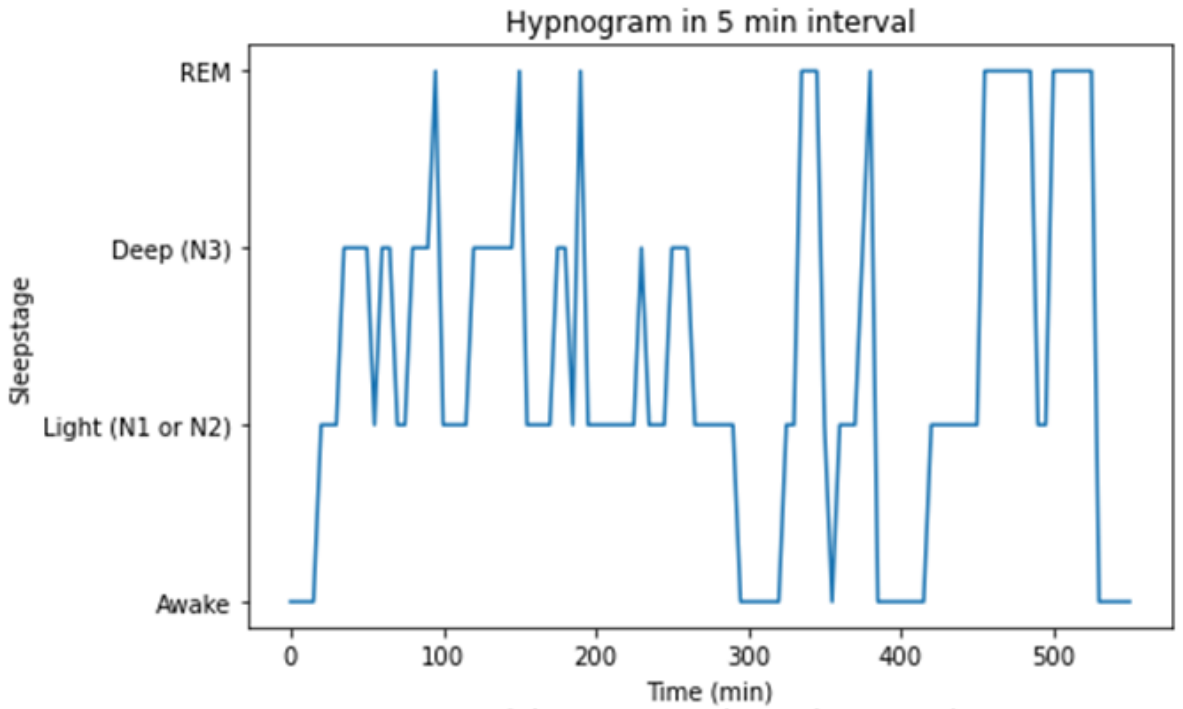
```
#Lets plot the activity spectra for the nights during which the user was awake for more than
15 minutes in a row.
```

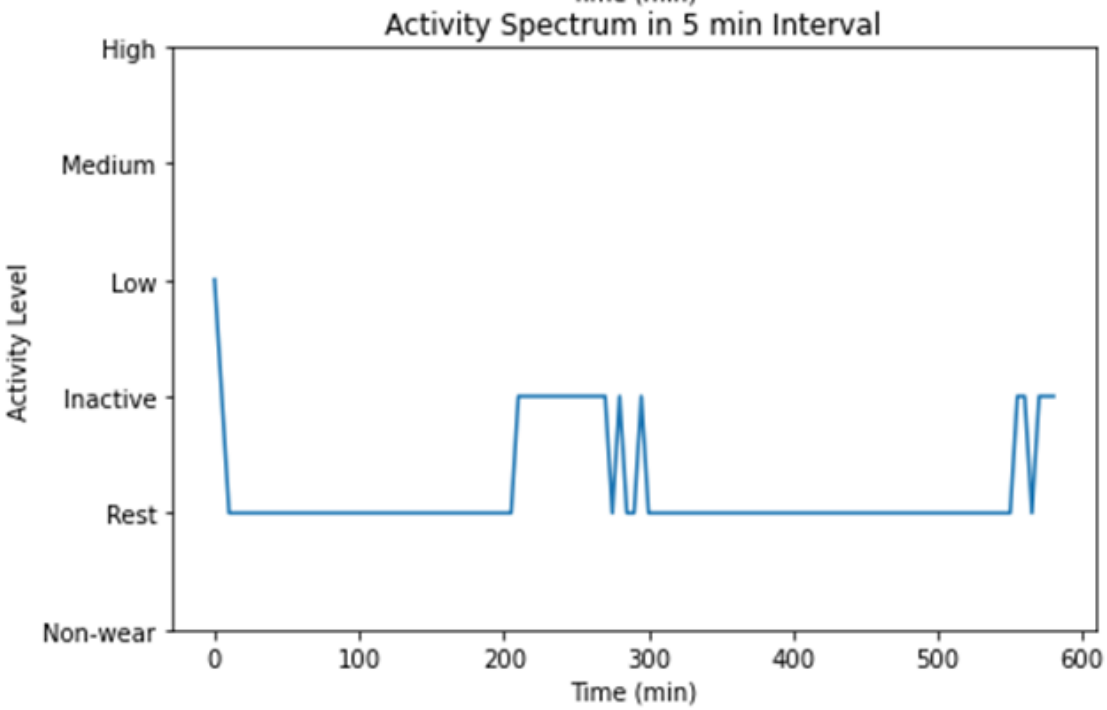
#Since I want to plot the hypnograms and the activity spectra right above each other for easy comparison I will not use the plotting function created earlier, as it does not allow such operations

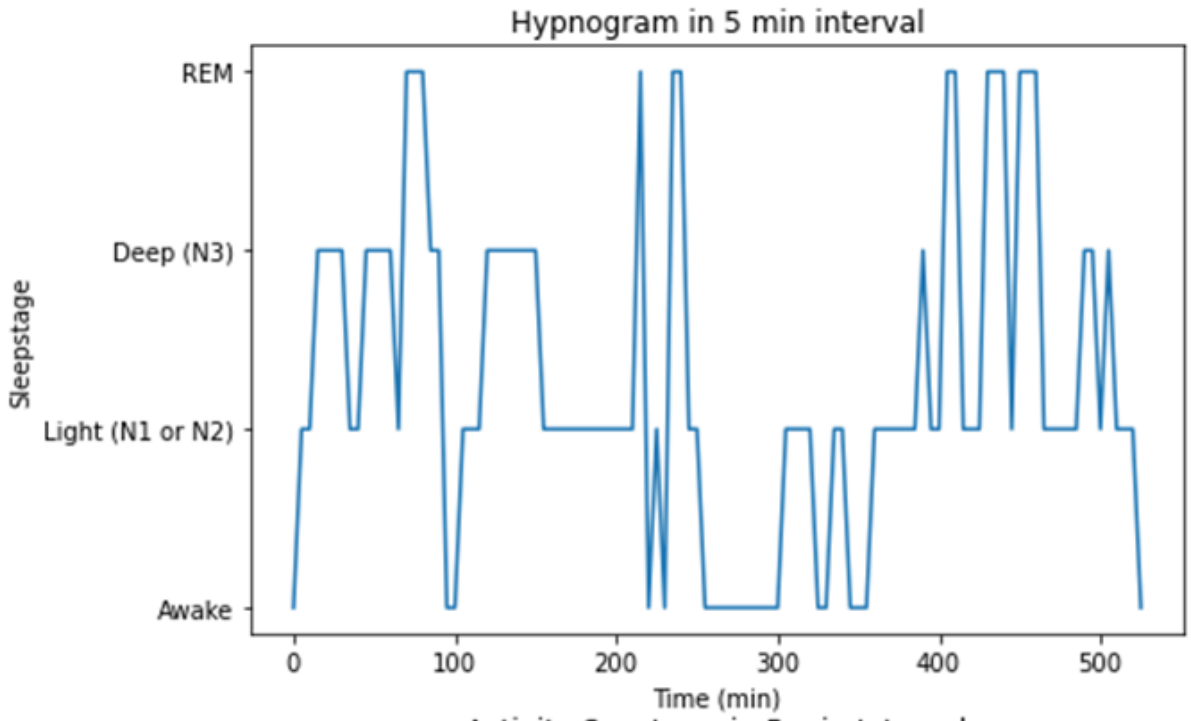
```
counter=-1

#Converting the activity string to a list of integers
for i in activity_strings:
        counter+=1
        as5_split=[]
        for j in range(0,len(i)):
        as5_split.append(int(i[j]))

   #Plotting the hypnograms for easy comparison
        fig, (ax1, ax2)=plt.subplots(2, 1, figsize=(7,10))
        ax1.plot(hgs5[counter])
        ax1.set_title('Hypnogram in 5 min interval')
        ax1.set_ylabel('Sleepstage')
        ax1.yaxis.set_major_locator(matplotlib.ticker.FixedLocator([0, 1, 2, 3]))
        ax1.set_yticklabels(['Awake', 'Light (N1 or N2)', 'Deep (N3)', 'REM'])
        ax1.set_xlabel('Time (min)\n')
        ax1.xaxis.set_major_locator(matplotlib.ticker.FixedLocator([0, 20, 40, 60, 80, 100,
120, 140, 160]))
        ax1.set_xticklabels(['0', '100', '200', '300', '400', '500', '600', '700', '800'])

        #Plotting the activity spectra
        ax2.plot(as5_split)
        ax2.set_title('Activity Spectrum in 5 min Interval')
        ax2.set_ylabel('Activity Level')
        ax2.yaxis.set_major_locator(matplotlib.ticker.FixedLocator([0, 1, 2, 3, 4, 5]))
        ax2.set_yticklabels(['Non-wear', 'Rest', 'Inactive', 'Low', 'Medium', 'High'])
        ax2.set_ylim([0, 5])
        ax2.set_xlabel('Time (min)')
        ax2.xaxis.set_major_locator(matplotlib.ticker.FixedLocator([0, 20, 40, 60, 80, 100,
120, 140, 160]))
        ax2.set_xticklabels(['0', '100', '200', '300', '400', '500', '600', '700', '800'])
        plt.show()
```

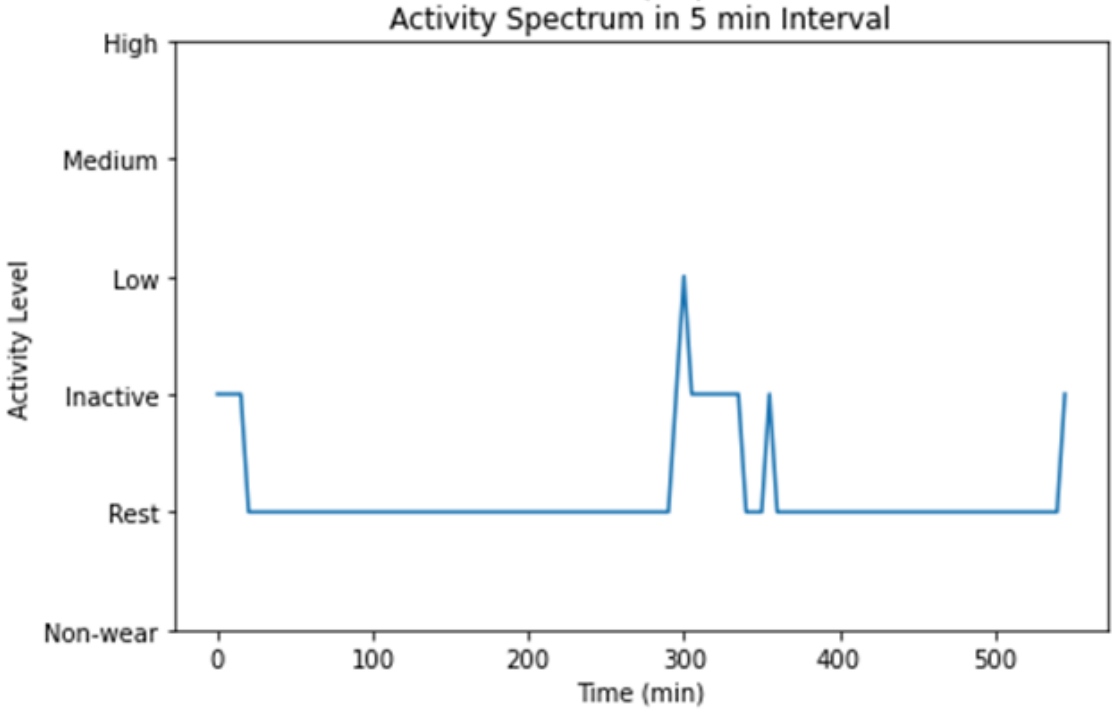

Hypnogram in 5 min interval
REM
Deep (N3)
Light (N1 or N2)
Awake
Sleepstage
0
100
200
300
400
500
600
Time (min)

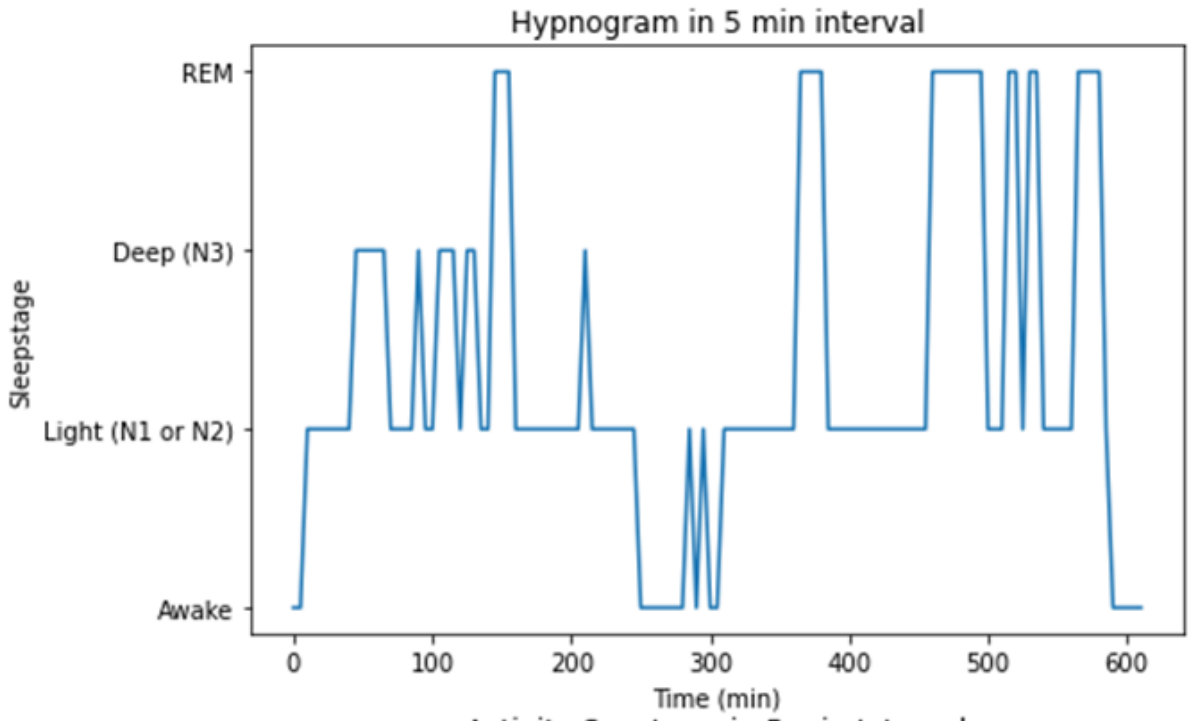

Activity Spectrum in 5 min Interval
High
Medium
Low
Inactive
Rest
Non-wear
Activity Level
0
100
200
300
400
500
Time (min)

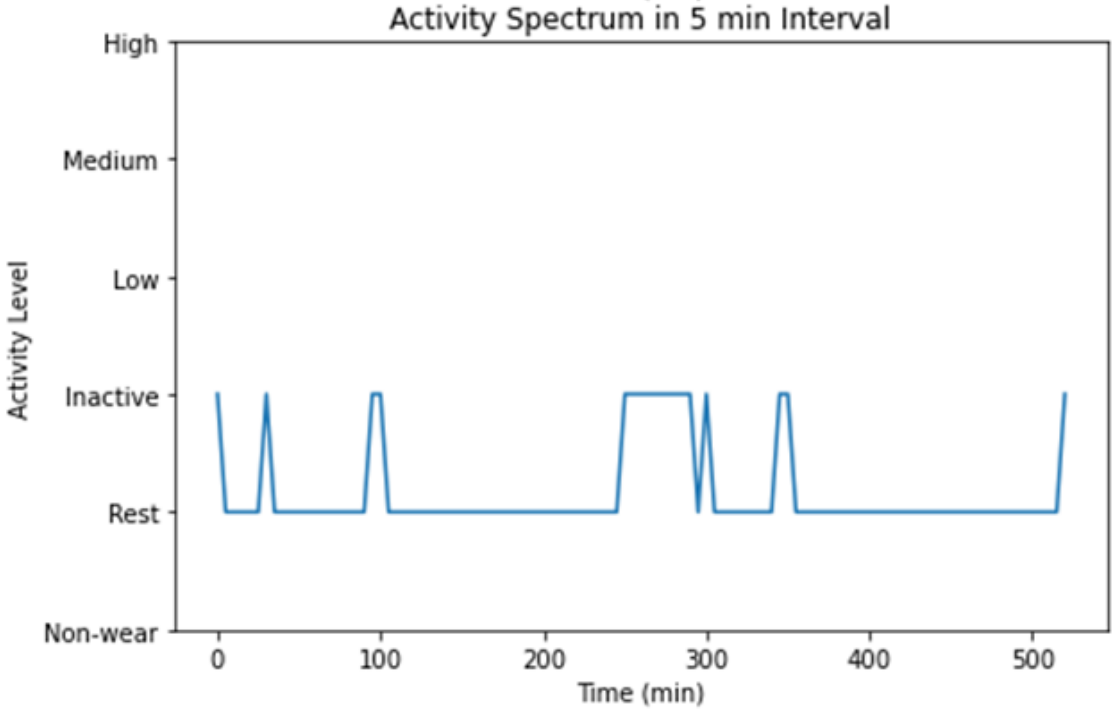

Hypnogram in 5 min interval
REM
Deep (N3)
Light (N1 or N2)
Awake
Sleepstage
0
100
200
300
400
500
Time (min)

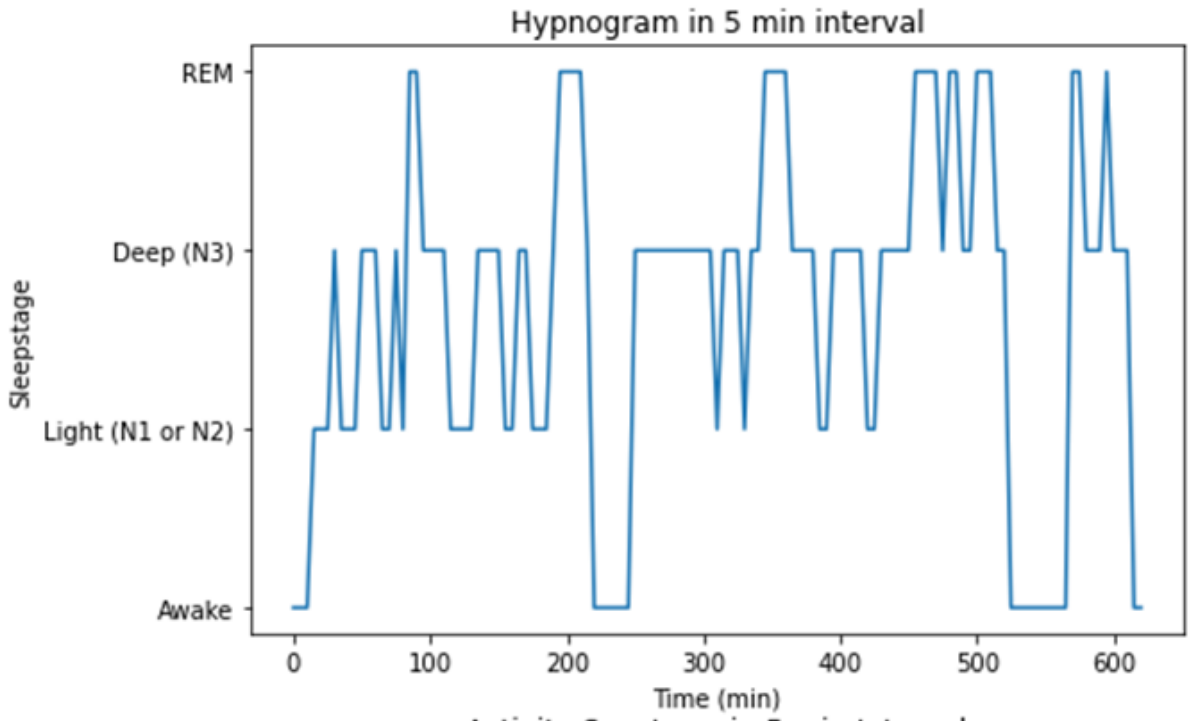

Activity Spectrum in 5 min Interval
High
Medium
Low
Inactive
Rest
Non-wear
Activity Level
0
100
200
300
400
500
600
Time (min)

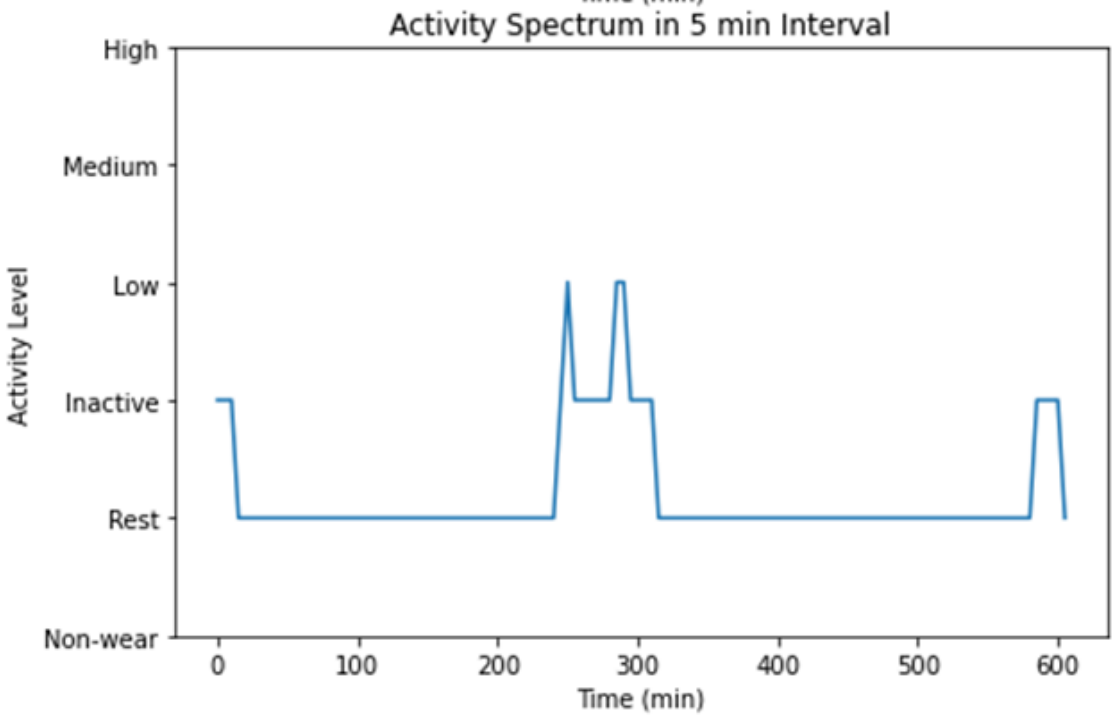

Hypnogram in 5 min interval
REM
Deep (N3)
Light (N1 or N2)
Awake
Sleepstage
0
100
200
300
400
500
Time (min)


Activity Spectrum in 5 min Interval
High
Medium
Low
Inactive
Rest
Non-wear
Activity Level
0
100
200
300
400
500
Time (min)


Hypnogram in 5 min interval
REM
Deep (N3)
Light (N1 or N2)
Awake
Sleepstage
0
100
200
300
400
500
600
Time (min)


Activity Spectrum in 5 min Interval
High
Medium
Low
Inactive
Rest
Non-wear
Activity Level
0
100
200
300
400
500
Time (min)





Since we needed more information from the 'activity' data, which was also recorded at a 5-minute interval (or less), the MET levels appeared to be a proper resource, since they are recorded at a 1-minute interval ('MET_1min'). The Oura API documentation (Õura Health Oy, n.d.) however, states that the MET level boundaries for the 'Low' level (and thus also for the levels above) are age related (See: 'Activity', 'class_5min', point 3).

I have emailed Oura ring about these proclaimed age-related boundaries and asked them if they could specify these boundaries. Even though I have asked this question two times, they have not clarified anything, meaning that it is not possible to use the 'MET_1min' data for the purpose of trying to draw a conclusion about the accuracy of the activity levels as specified in the 'class_5min' data.

Determining the advice

Based on this data alone we can only provide advice based upon the assumption that light activity corresponds to walking activity, which, according to the Oura API documentation, should be the medium level at least. Inquiries about these levels have not resulted in a clarification, making it unable for us to answer this question with certainty. Maybe the data from other users can provide more insight into this matter.

Relaxation techniques

With wearable technology we cannot measure directly whether someone is relaxed, but we can measure the oppossite, namely stress. The Center for Studies on Human Stress created a document in which they stated exactly which physiological aspects of stress can be measured (Centre d'études sur le stress humain, 2007). Of all measures described, the only one which can be provided by a wearable device is the blood pressure. And the problem with that is that it is not even actually measured, but only estimated based off some other measurements related to the heart rate and the heart rate variability, meaning their accuracy is limited (Caddy, 2021). There are however a lot of things that influence these variables; hence it is hard to provide an advice to start using relaxation techniques based on this data alone. It can be used however to provide evidence to the user of the effectiveness of relaxation techniques.

This is not possible using the Oura ring though, because the Oura ring only measures these variables during sleep, as can be read in the Oura API documentation in the sleep section, under the variables 'hr_5min' and 'rmssd_5min' (Ōura Health Oy, n.d.).

Another measure that can be used to detect stress, which is built in into some wearable devices, is skin conductance (Kim et al., 2019; Sawh, 2019). The Oura ring is not one of these devices though.

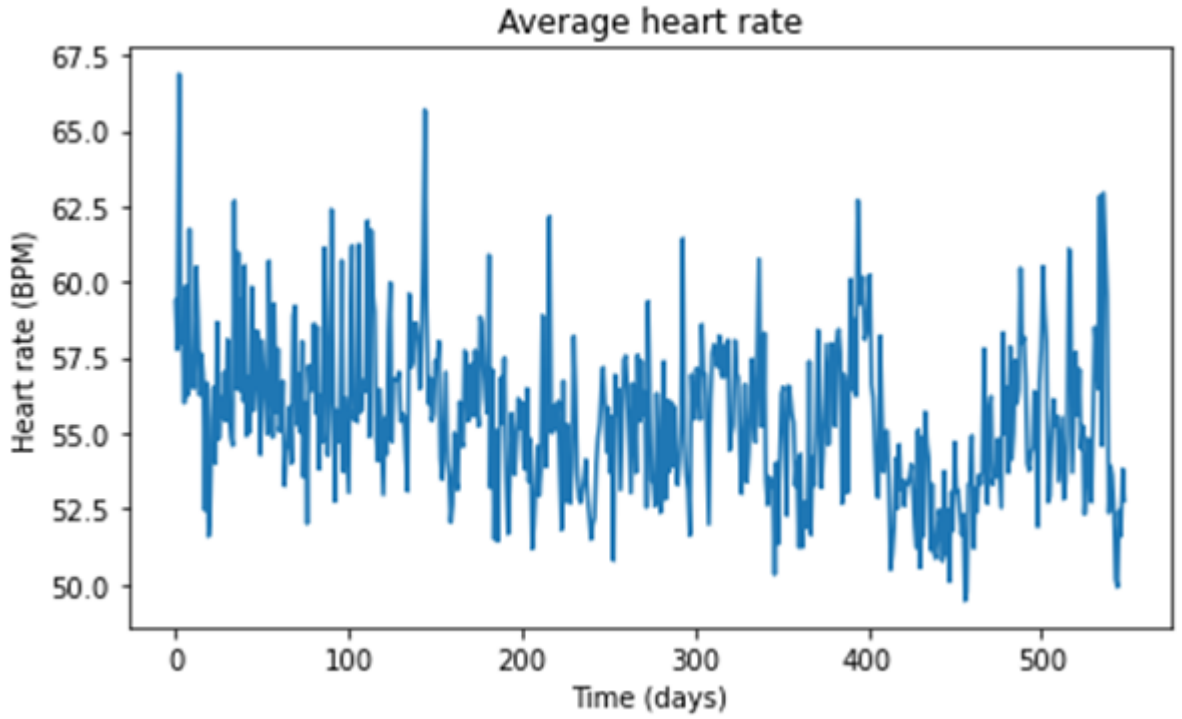
Let's have a look at the heart rate and the heart rate variability variables over time to see if we can detect a pattern from which it might be possible to draw any conclusions.

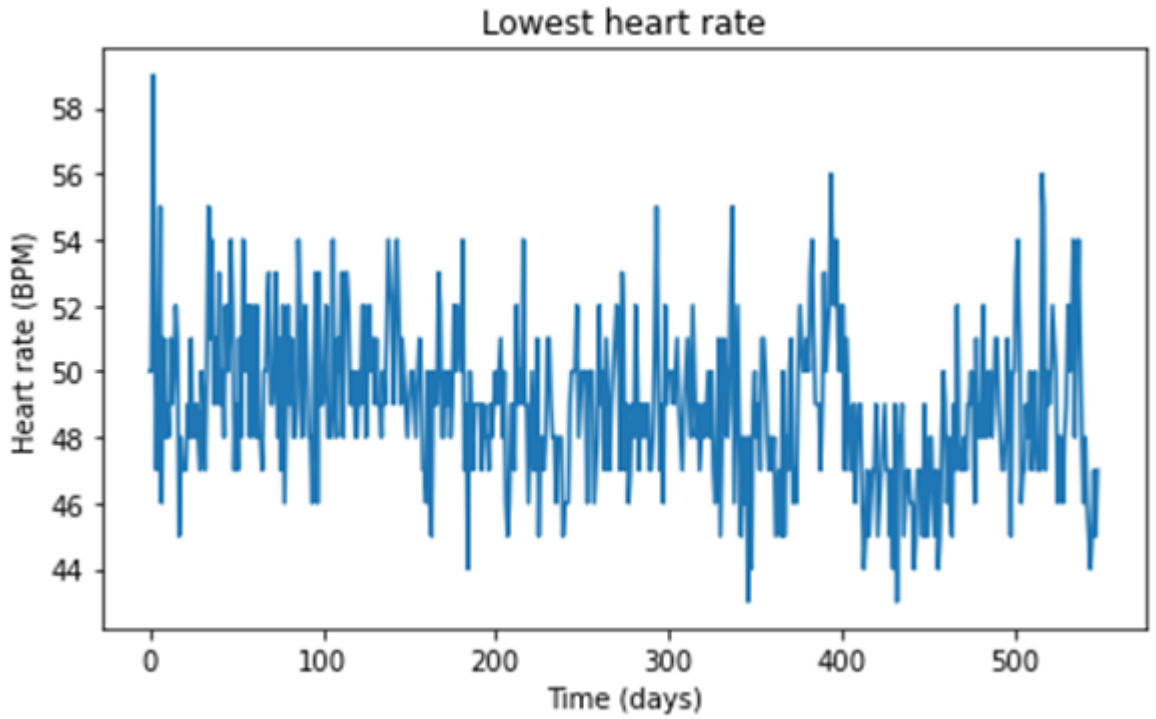
Inspecting the data

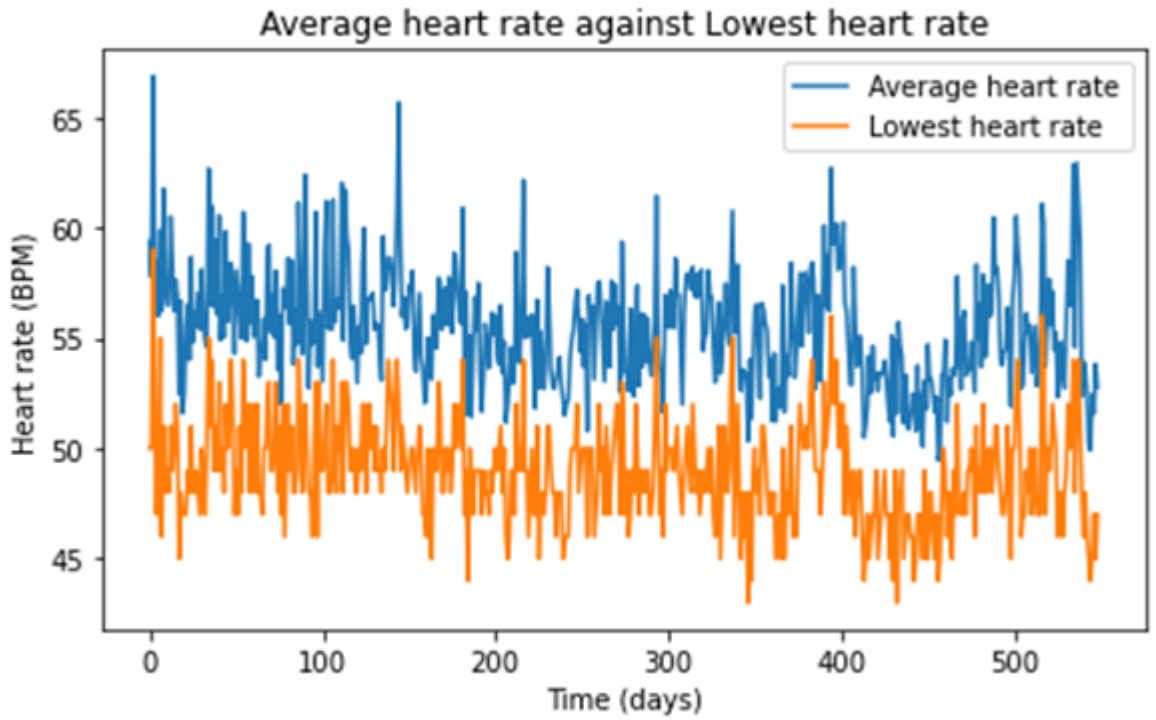
In [47]:

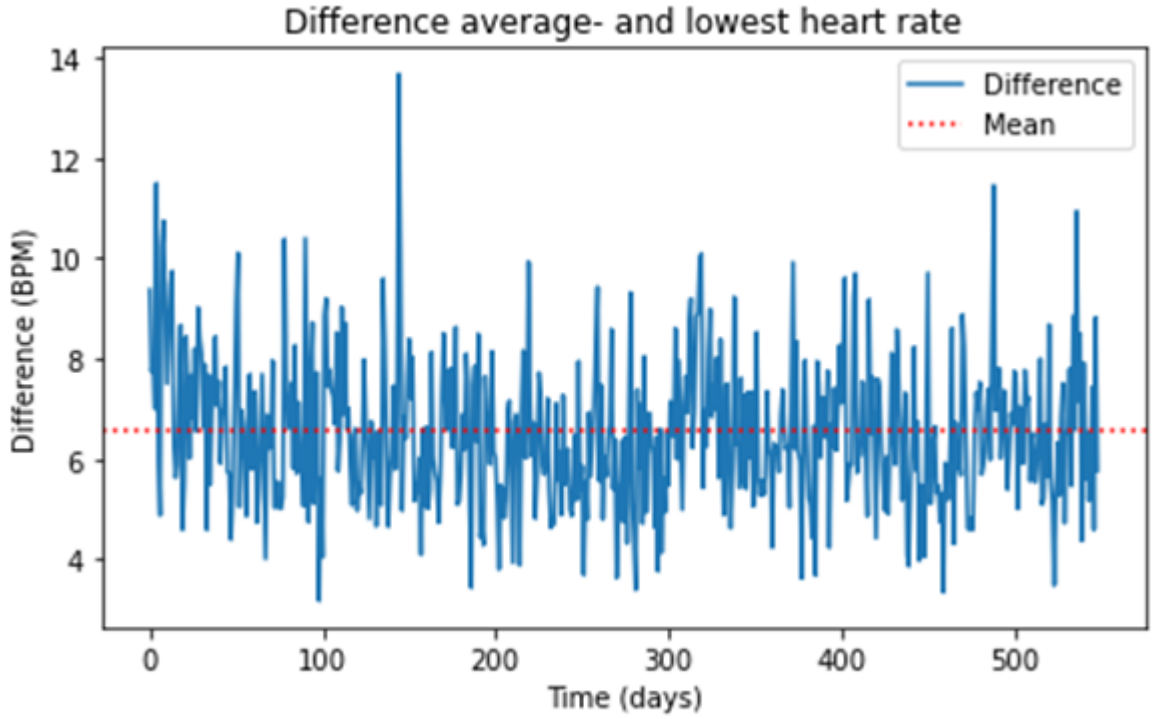
```
avg_hr=[]
l_hr=[]
rmssd=[]
for i in range(0, len(data['sleep'])):
        avg_hr.append(data['sleep'][i]['hr_average'])
        l_hr.append(data['sleep'][i]['hr_lowest'])
        rmssd.append(data['sleep'][i]['rmssd'])

plot(avg_hr[21:-1], [], [], 'Average heart rate', [], [], [], False, 'Time (days)', [], [], [], 'Heart rate (BPM)', [], [], [], [],[])
plot(l_hr[21:-1], [], [], 'Lowest heart rate', [], [], [], False, 'Time (days)', [], [], [], 'Heart rate (BPM)', [], [], [], [], [])
plot(avg_hr[21:-1], [], l_hr[21:-1], 'Average heart rate against Lowest heart rate', [], [], [], False, 'Time (days)', [], [], [], 'Heart rate (BPM)', [], [], [], ['Average heart rate', 'Lowest heart rate'], [])
```

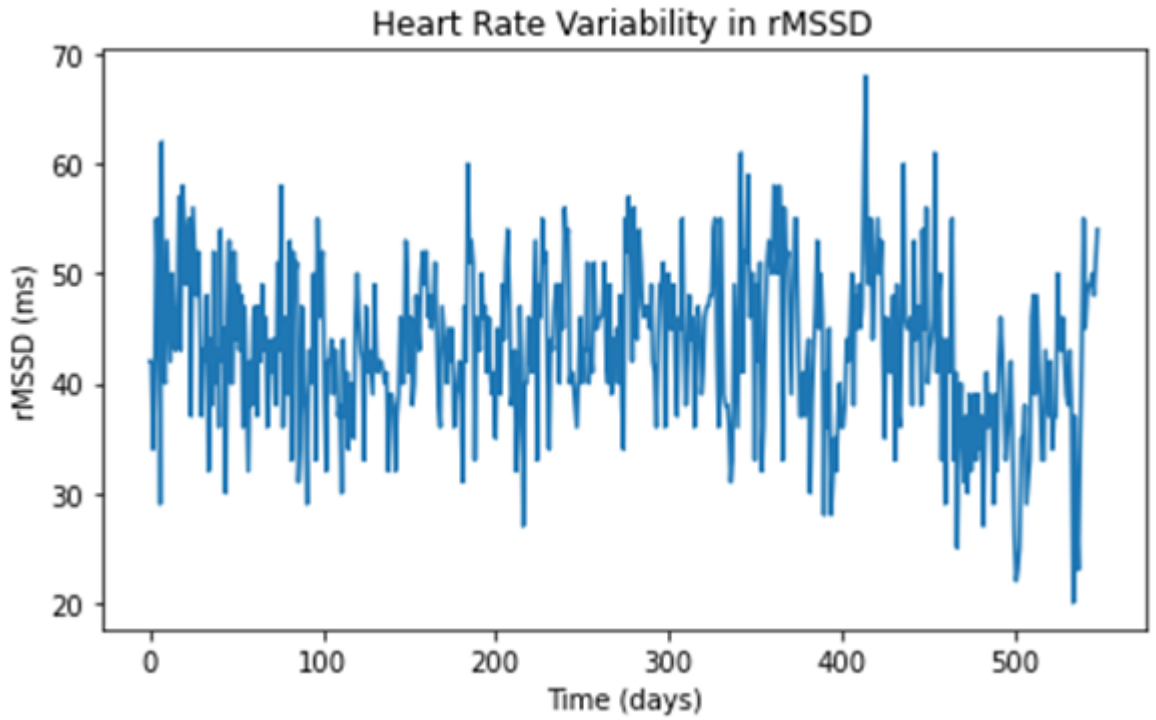
plot(np.subtract(avg_hr[21:-1], l_hr[21:-1]), [], [], 'Difference average- and lowest heart rate', [np.mean(np.subtract(avg_hr, l_hr))], [], [], False, 'Time (days)', [], [], [], 'Difference (BPM)', [], [], [], ['Difference', 'Mean'], [])
plot(rmssd[21:-1], [], [], 'Heart Rate Variability in rMSSD', [], [], [], False, 'Time (days)', [], [], [], 'rMSSD (ms)', [], [], [], [], [])






Average heart rate against Lowest heart rate
Average heart rate
Lowest heart rate
Heart rate (BPM)
65
60
55
50
45
0
100
200
300
400
500
Time (days)


Difference average- and lowest heart rate
Difference
Mean
Difference (BPM)
14
12
10
8
6
4
0
100
200
300
400
500
Time (days)



No clear pattern emerges.

Determining the advice

As already mentioned, before it currently seems to be impossible to provide an advice for the use of relaxation techniques using the Oura ring, or provide evidence of the effectiveness after the user has tried relaxation techniques, because the heart rate and heart rate variability are only measured during sleep.

There is a possibility that it evidence can be provided, given the fact that a small period right before sleep is measured too, the SOL, as we have seen in the hypnograms. Hence it might be possible to actually measure the effects of relaxation techniques on the heart rate and heart rate variability if these relaxation techniques are performed whilst laying in bed right before going to sleep. However, the effect would only be visible the day after when the sleep has been processed. Of all relaxation techniques in the framework, the following can be performed whilst laying in bed, right before sleep: Progressive Muscle Relaxation (PMR)/Body scan, (Deep) Breathing techniques, Visualization/Guided Imagery, Mindfulness meditation and/or Repetitive Prayer/Mantra. More research is needed to determine whether this would actually be recorded by the Oura ring though.

Another option that would allow the provision of this evidence, is by recording this data during the day. This feature can be requested at the Oura ring website.

Body temperature

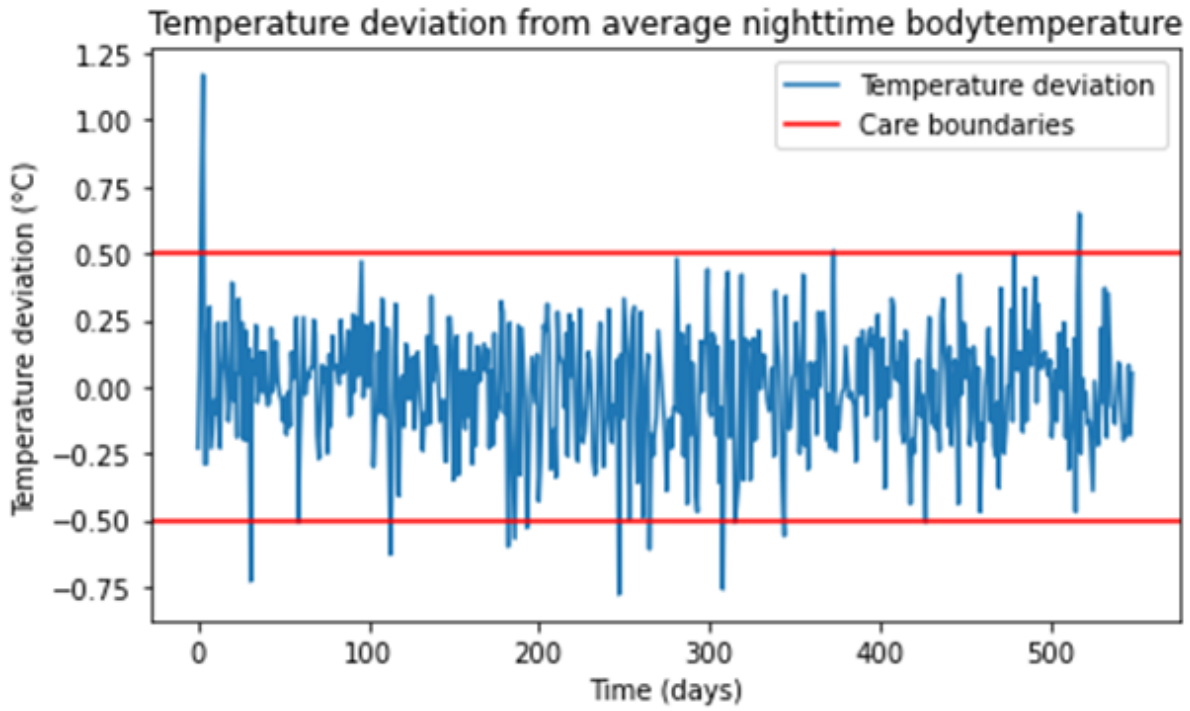
In their introduction to body temperature, the company behind the Oura ring claims that the body is in its most stable state during the night, which is why measurements of the body temperature taken during the night are the most reliable in terms of representing the true state of the body (Ōura Health Oy, n.d.). Furthermore, they say that any deviation of more than 0.5°C in either direction can indicate that your body is fighting something and accordingly recommend the user to take more time to rest than usual and to take care of themselves.

In the same article the sensor used to measure the body temperature is provided: a Negative Temperature Coefficient (NTC) sensor. This sensor takes measurements at an interval that is not specified and stores the average body temperature deviation per night under the 'Sleep' data under the 'Temperature_delta' and 'Temperature_deviation' variables. As was mentioned during the primary data inspection already, these variables store exactly the same data in exactly the same format, so for clarity and consistency I will just use the 'Temperature_delta' variable from now on.

Inspecting the data

In [48]:

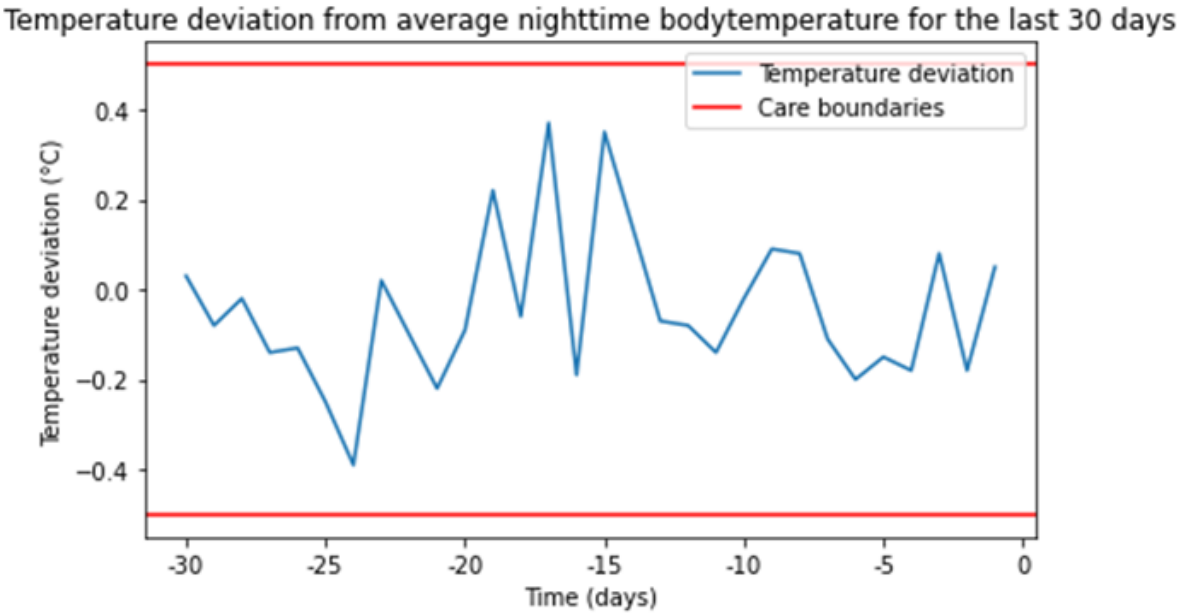
```
temp_delta=[]
for i in range(0, len(data['sleep'])):
        temp_delta.append(data['sleep'][i]['temperature_delta'])

plot(temp_delta[21:-1], [], [], 'Temperature deviation from average nighttime
bodytemperature', [0.5, -0.5], [], ['r'], False, 'Time (days)', [], [], [], 'Temperature deviation
(°C)', [], [], [], ['Temperature deviation', 'Care boundaries'], [])
```

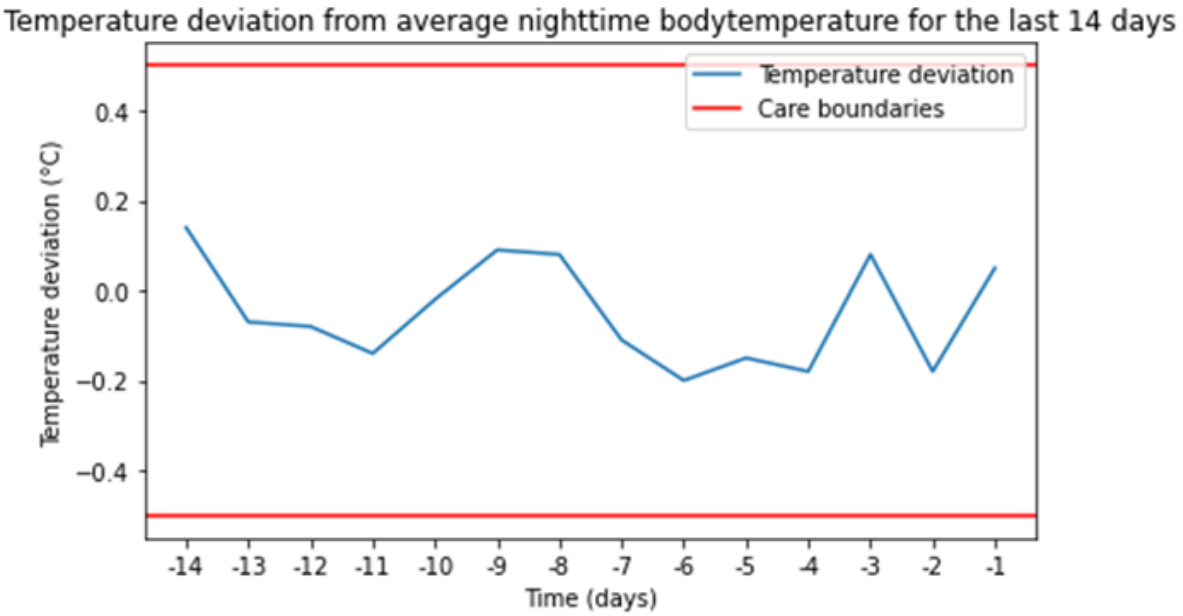


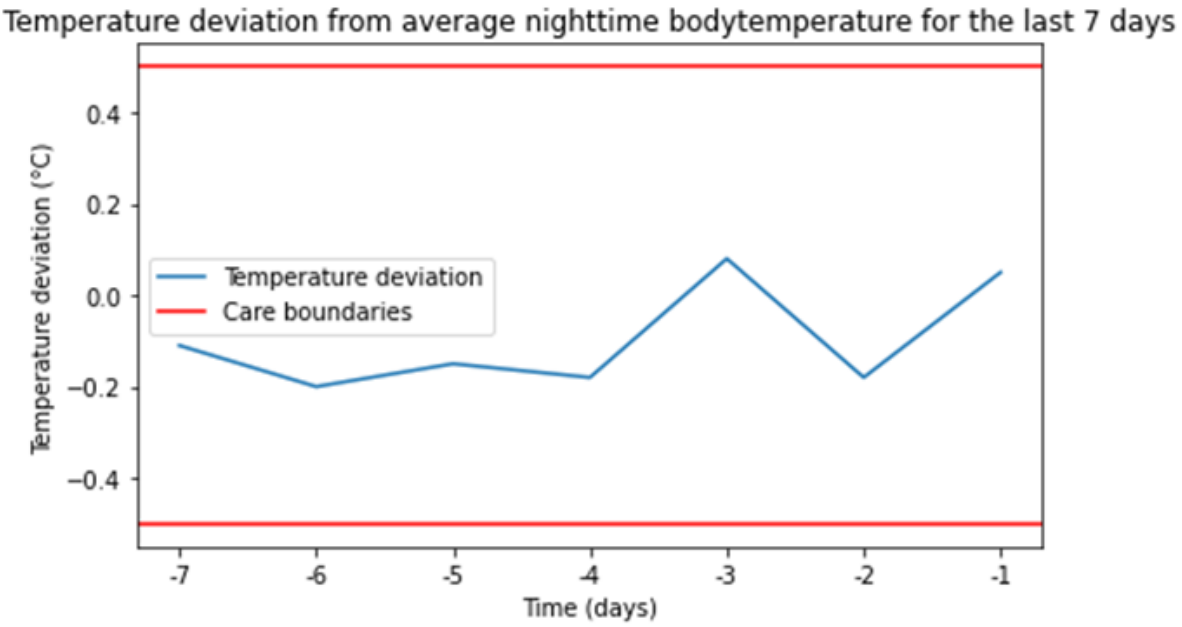
In [49]:

```
plot(temp_delta[-31:-1], [], [], 'Temperature deviation from average nighttime
bodytemperature for the last 30 days', [0.5, -0.5], [], ['r'], False, 'Time (days)', [np.arange(0,
31, 5)], [np.arange(-30, 1, 5)], [], 'Temperature deviation (°C)', [], [], [], ['Temperature
deviation', 'Care boundaries'], [])
plot(temp_delta[-15:-1], [], [], 'Temperature deviation from average nighttime
bodytemperature for the last 14 days', [0.5, -0.5], [], ['r'], False, 'Time (days)', [np.arange(0,
14)], [np.arange(-14, 0)], [], 'Temperature deviation (°C)', [], [], [], ['Temperature deviation',
'Care boundaries'], [])
plot(temp_delta[-8:-1], [], [], 'Temperature deviation from average nighttime bodytemperature
for the last 7 days', [0.5, -0.5], [], ['r'], False, 'Time (days)', [np.arange(0, 7)], [np.arange(-7,
```

0)], [], 'Temperature deviation (°C)', [], [], [], ['Temperature deviation', 'Care boundaries'], [])







Since the body temperature average is being calculated over the long term during the night, it is hard to draw conclusions based on this data. If the user is usually too warm or too cold during the night, it is still considered to be the average body temperature. If the user would consistently be warmer than average, it would be advisable to make some changes to the sleeping environment, such as sleeping under thinner bedsheets, stop wearing a (complete) pyjama or to try sleeping with the window open and vice versa if the users body temperature would be colder than average.

Maybe this data gets more informative when combined with other data, so I'm going to see if there is a correlation between the average body temperature deviation and any of the single value attributes. Probably the heart rate is correlated the strongest with the body temperature.

Inspecting the correlation coefficients

In [50]:

```
efficiency=[]
score_disturbances=[]
midpoint_time=[]
total=[]
duration=[]
awake=[]
light=[]
deep=[]
rem=[]
onset_latency=[]
restless=[]
breath_avg=[]

for i in range(22,len(data['sleep'])):
        efficiency.append(data['sleep'][i]['efficiency'])
        score_disturbances.append(data['sleep'][i]['score_disturbances'])
        midpoint_time.append(data['sleep'][i]['midpoint_time'])
```

```
        total.append(data['sleep'][i]['total'])
        duration.append(data['sleep'][i]['duration'])
        awake.append(data['sleep'][i]['awake'])
        light.append(data['sleep'][i]['light'])
        deep.append(data['sleep'][i]['deep'])
        rem.append(data['sleep'][i]['rem'])
    onset_latency.append(data['sleep'][i]['onset_latency'])
        restless.append(data['sleep'][i]['restless'])
        breath_avg.append(data['sleep'][i]['breath_average'])

print('Efficiency: {}'.format(np.corrcoef(temp_delta[21:-1], efficiency)[0][1]))
print('Score_disturbances: {}'.format(np.corrcoef(temp_delta[21:-1],
score_disturbances)[0][1]))
print('Midpoint_time: {}'.format(np.corrcoef(temp_delta[21:-1], midpoint_time)[0][1]))
print('Total: {}'.format(np.corrcoef(temp_delta[21:-1], total)[0][1]))
print('Duration: {}'.format(np.corrcoef(temp_delta[21:-1], duration)[0][1]))
print('Awake: {}'.format(np.corrcoef(temp_delta[21:-1], awake)[0][1]))
print('Light: {}'.format(np.corrcoef(temp_delta[21:-1], light)[0][1]))
print('Deep: {}'.format(np.corrcoef(temp_delta[21:-1], deep)[0][1]))
print('Rem: {}'.format(np.corrcoef(temp_delta[21:-1], rem)[0][1]))
print('Onset_latency: {}'.format(np.corrcoef(temp_delta[21:-1], onset_latency)[0][1]))
print('Restless: {}'.format(np.corrcoef(temp_delta[21:-1], restless)[0][1]))
print('Breath_average: {}'.format(np.corrcoef(temp_delta[21:-1], breath_avg)[0][1]))
print('Lowest_hr: {}'.format(np.corrcoef(temp_delta[21:-1], l_hr[21:-1])[0][1]))
print('Average_hr: {}'.format(np.corrcoef(temp_delta[21:-1], avg_hr[21:-1])[0][1]))
print('rMSSD: {}'.format(np.corrcoef(temp_delta[21:-1], rmssd[21:-1])[0][1]))
Efficiency: -0.0074070443029226295
Score_disturbances: 0.01269112698368937
Midpoint_time: 0.013459289740388397
Total: 0.010674388252648041
Duration: 0.014690136663381675
Awake: 0.009316875430696935
Light: 0.00875949417810744
Deep: 0.020509137927235393
Rem: -0.013143450694316171
Onset_latency: 0.018826127502503093
Restless: -0.03277985964158087
Breath_average: 0.01879335083096084
Lowest_hr: 0.15839256625544504
Average_hr: 0.181595050652787
rMSSD: -0.12077042756098029
```

None of the single value attributes correlate well with the temperature_delta variable. The lowest- and average heart rate variables correlate the most with the temperature_delta variables, with correlation values of 0.158 and 0.182, which is considered to be no relation because the r-values are less than 0.25.

Determining the advice

Since the body temperature data is stored as a deviation from the average night-time body temperature, ánd because there is no correlation between the data from this variable and the data from any other variable or attribute, it is impossible to draw a meaningful conclusion. This means that we cannot base a scientifically supported advice on this data to improve the sleep quality with respect to the body temperature.

Exercise regularly

The Oura ring measures the activity level through the three different sensors that measure heart- and respiration rate, body temperature and movement. Even though the accuracy of these sensors individually have been tested, no accuracy level can be found for any variables determined through combinations of measurements taken by these sensors, as is most likely the case for the activity levels.

In the data there are three variables that represent the activity the user gets:

1. Activity.score_training_frequency
2. Activity.daily_movement
3. Activity.steps

The first variable is an indirect measure, as it is a score attributed to a specific level of activity during the week. This score is maximal when the user has had more than 100 minutes of medium level or higher on at least four days in the past week, and 95 when the user has done so on at least three days in the past week.

The second variable expresses the daily physical activity as the number of meters that would have been walked, if all the energy put into the physical activity had been put into walking.

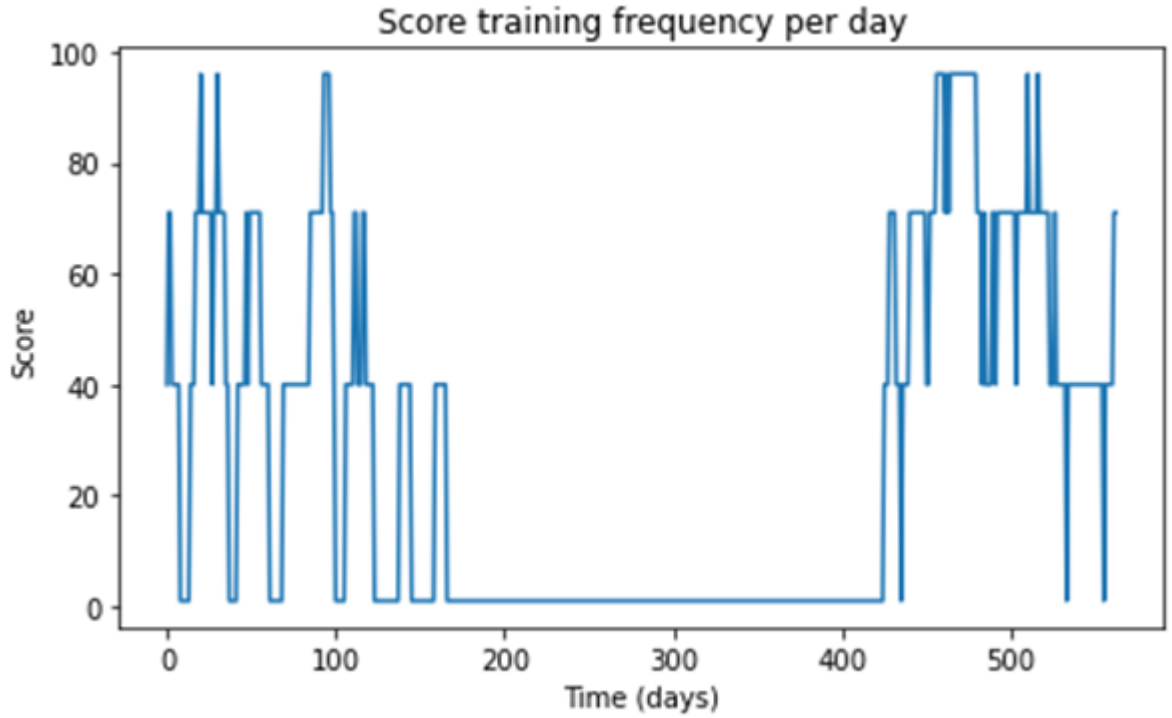
The last variable is the number of steps taken during the day.

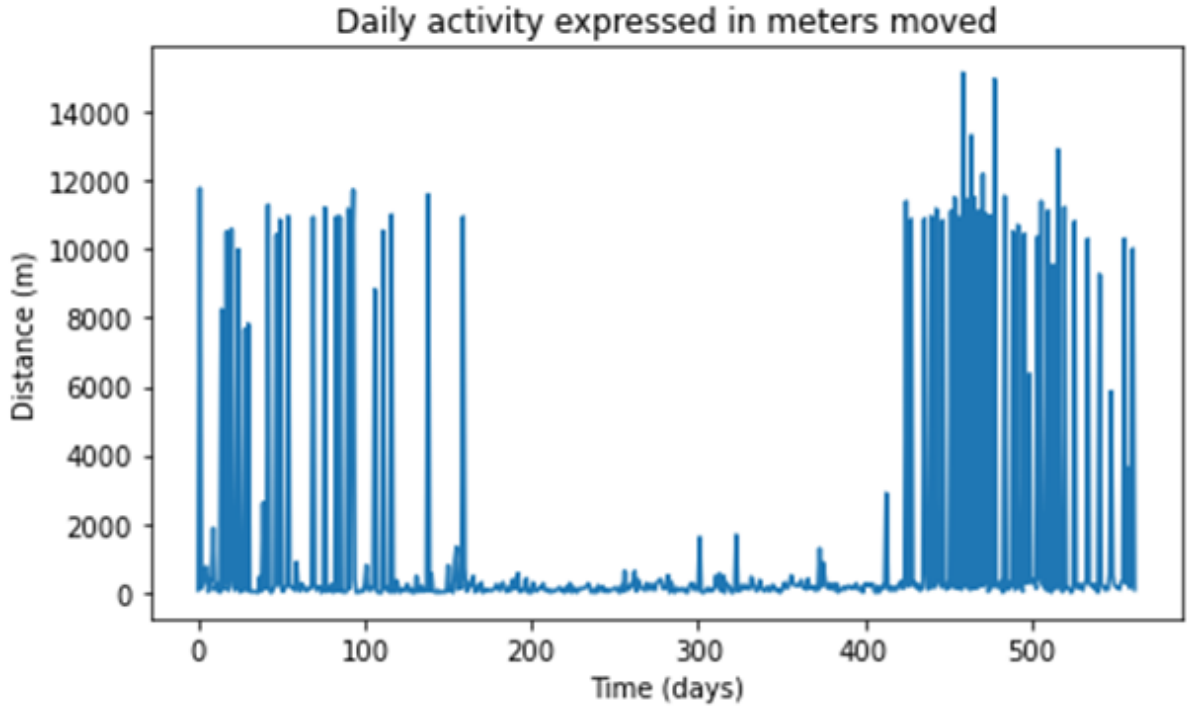
Inspecting the data

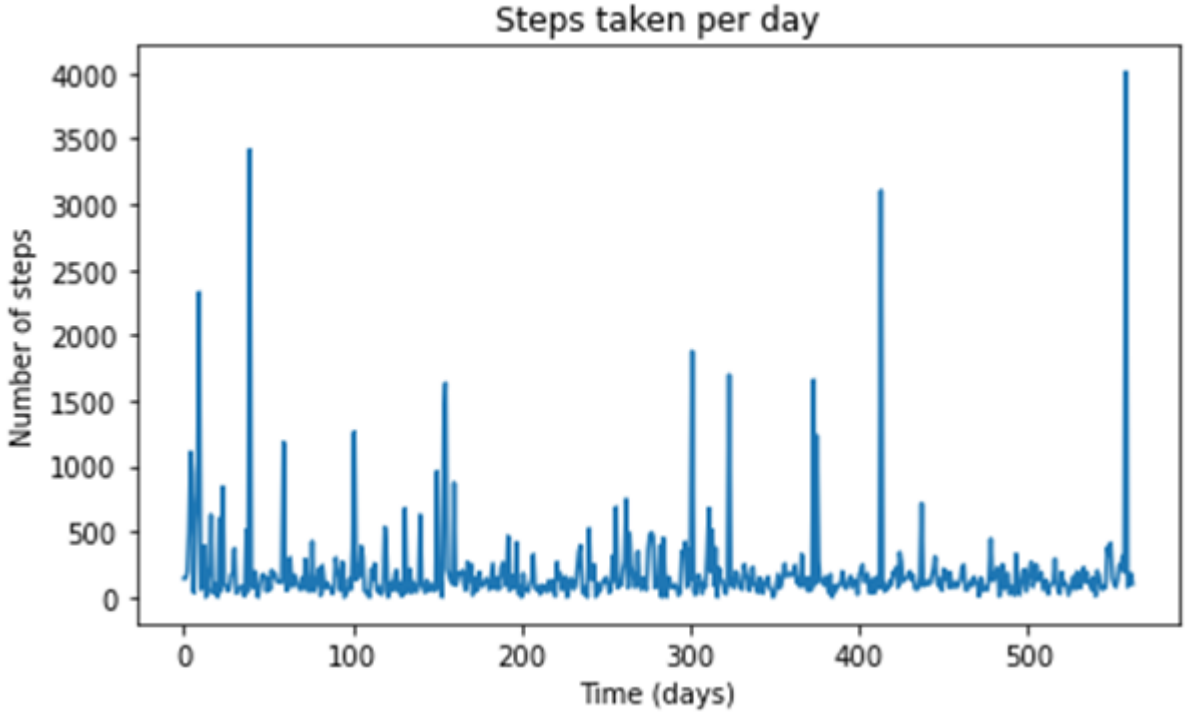
In [51]:

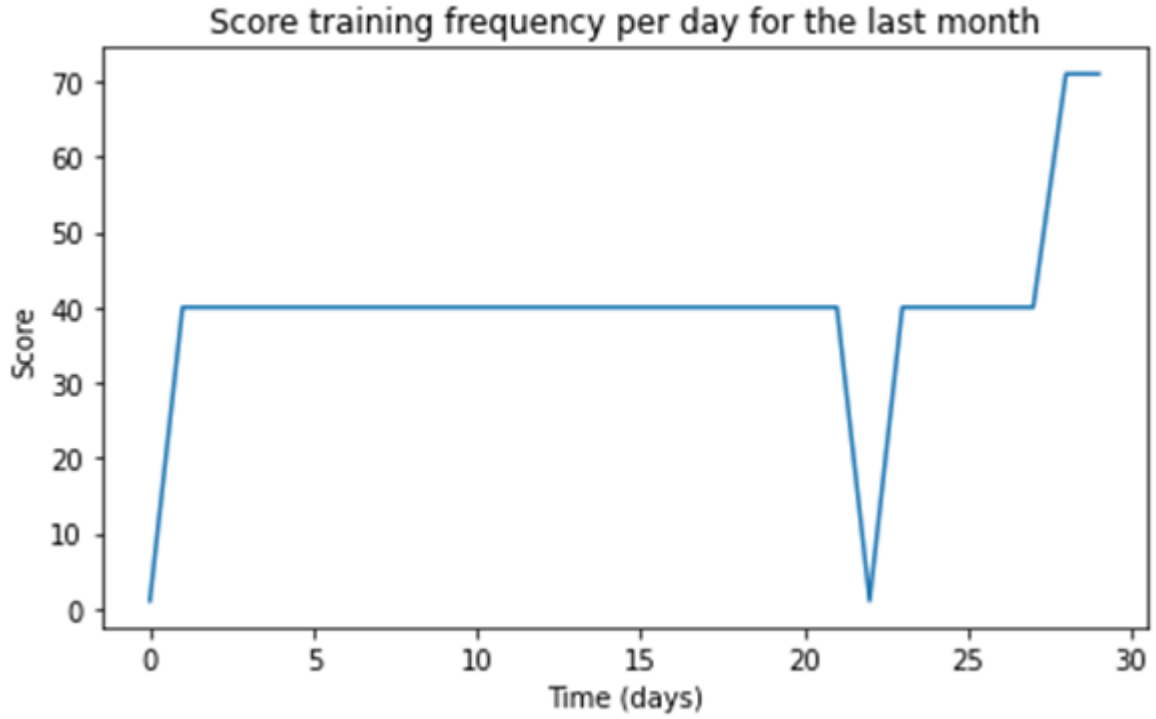
```
#Creating containers for the score training frequency per day and for the daily movement per day
stf=[]
dm=[]

#Filling the containers with all the training frequency scores and the all the daily movement
for i in range(0, len(data['activity'])):
        stf.append(data['activity'][i]['score_training_frequency'])
        dm.append(data['activity'][i]['daily_movement'])

#Plotting the training frequency scores, daily movement and number of steps (earlier created already) over the entire dataset, starting after the largest gap
plot(stf[21:-1], [], [], 'Score training frequency per day', [], [], [], False, 'Time (days)', [], [], [], 'Score', [], [], [], [], [])
plot(dm[21:-1], [], [], 'Daily activity expressed in meters moved', [], [], [], False, 'Time (days)', [], [], [], 'Distance (m)', [], [], [], [], [])
```

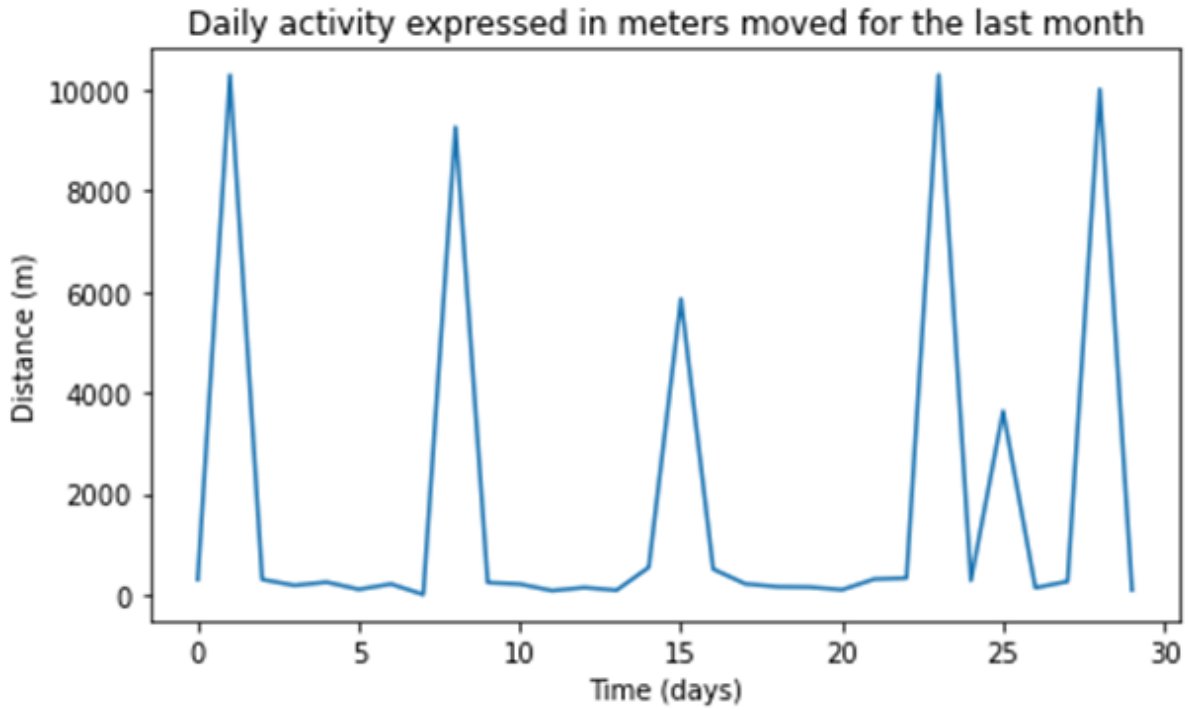
```
plot(steps[21:-1], [], [], 'Steps taken per day', [], [], [], False, 'Time (days)', [], [], [], 'Number of steps', [], [], [], [], [])

#Plotting the training frequency scores, daily movement and number of steps (earlier created already) over the last month
plot(stf[-31:-1], [], [], 'Score training frequency per day for the last month', [], [], [], False, 'Time (days)', [], [], [], 'Score', [], [], [], [], [])
plot(dm[-31:-1], [], [], 'Daily activity expressed in meters moved for the last month', [], [], [], False, 'Time (days)', [], [], [], 'Distance (m)', [], [], [], [], [])
plot(steps[-31:-1], [], [], 'Steps taken per day for the last month', [], [], [], False, 'Time (days)', [], [], [], 'Number of steps', [], [], [], [], [])
```

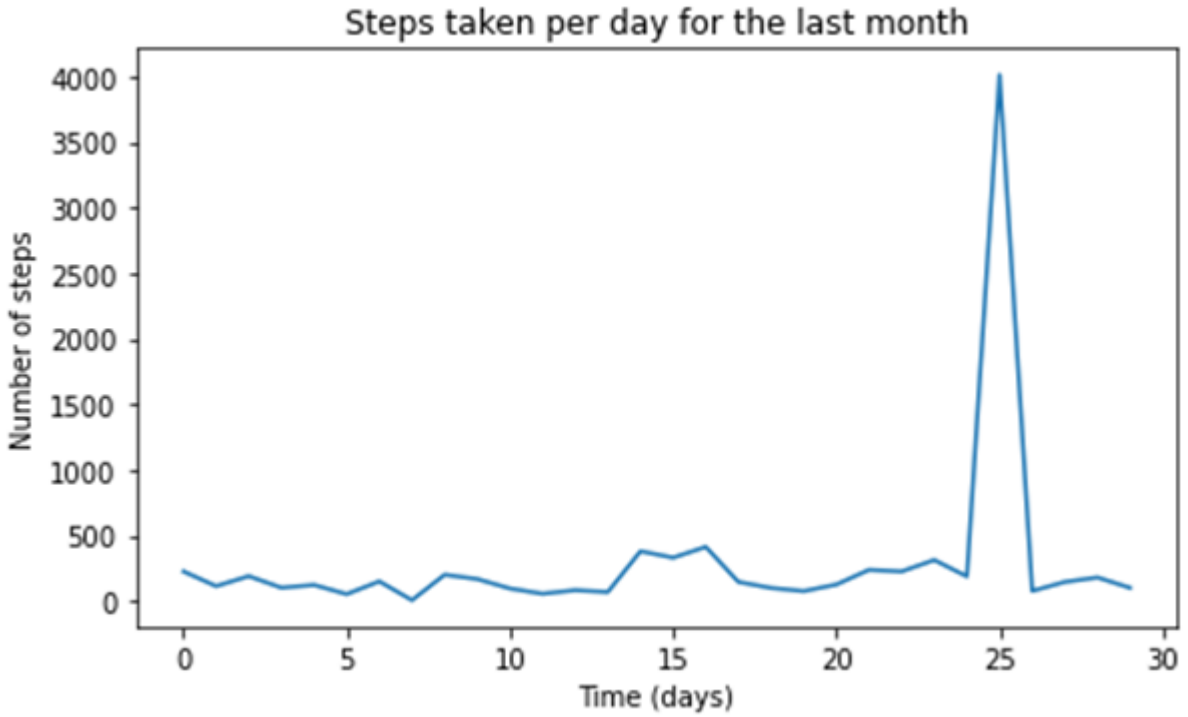

Score training frequency per day
Score
Time (days)
0
20
40
60
80
100
0
100
200
300
400
500


Daily activity expressed in meters moved
Distance (m)
Time (days)
0
2000
4000
6000
8000
10000
12000
14000
0
100
200
300
400
500


Steps taken per day
Number of steps
4000
3500
3000
2500
2000
1500
1000
500
0
0
100
200
300
400
500
Time (days)


Score training frequency per day for the last month
Score
70
60
50
40
30
20
10
0
0
5
10
15
20
25
30
Time (days)





First of all, I conclude from the 'score_training_frequency' data, that there are 5 levels: 100, 95, 70, 40 and 0. According to the Oura API documentation (Ōura Health Oy, n.d.) the first two levels correspond to 100 minutes of medium activity on at least 4 days in the past week and to 100 minutes of medium activity on 3 days in the past week. Given that there are 3 more levels, these levels probably correspond to at least 100 minutes of medium activity for 2 days, 1 day and 0 days in the past week.

Secondly, the number of steps is quite low according to Lee et al. (2019), this makes sense however, because the user usually does not wear the ring during the day.

Hence, I can conclude from this data visualisation that the user does not exercise on a regular basis **whilst wearing the ring**. Self-report can however provide additional data here to rectify any missing data due to not wearing the ring.

Determining the advice

Since the 'daily_movement' is an indirect variable, that is influenced by more than just the exercise someone gets and the number of steps they take during they day, combined with the fact that there is no scientific measurement of how much 'daily_movement' in meters someone from a specific age group should reach, I decided not to use it for determining the advice. For the number of steps, I decided to use the number associated with the lowest possible mortality rate achievable by walking (Lee et al., 2019).

In [52]:

```
def exercise_regularly():
    #This function is designed to determine the number of days the user exercised enough, where exercise is classified as either walking more than 7500 steps a day or being active on the medium level for more than 100 minutes on a certain amount of days per week.
    try:
    #Determining on how many days the user walked 7500 steps or more during the last week
    more_than_7500_steps=0
    for i in range(-8, -1):
    if data['activity'][i]['steps']>=7500:
        more_than_7500_steps+=1

    #Determining on how many days the user was active on the medium level for more than 100 minutes, or walked 7500 steps or more during the last week
    if (data['activity'][-1]['score_training_frequency']==100 and more_than_7500_steps==0) or (data['activity'][-1]['score_training_frequency']>=95 and data['activity'][-1]['score_training_frequency']<100 and more_than_7500_steps>0) or (data['activity'][-1]['score_training_frequency']>=70 and data['activity'][-1]['score_training_frequency']<95 and more_than_7500_steps>1) or (data['activity'][-1]['score_training_frequency']>=40 and data['activity'][-1]['score_training_frequency']<70 and more_than_7500_steps>2) or (more_than_7500_steps>3):
    return 4
    elif (data['activity'][-1]['score_training_frequency']>=95 and data['activity'][-1]['score_training_frequency']<100 and more_than_7500_steps==0) or (data['activity'][-1]['score_training_frequency']>=70 and data['activity'][-1]['score_training_frequency']<95 and more_than_7500_steps==1) or (data['activity'][-1]['score_training_frequency']>=40 and data['activity'][-1]['score_training_frequency']<70 and more_than_7500_steps==2) or (more_than_7500_steps==3):
    return 3
    elif (data['activity'][-1]['score_training_frequency']>=70 and data['activity'][-1]['score_training_frequency']<95 and more_than_7500_steps==0) or (data['activity'][-1]['score_training_frequency']>=40 and
```

```
data['activity'][-1]['score_training_frequency']<70 and more_than_7500_steps==1) or
(more_than_7500_steps==2):
        return 2
        elif (data['activity'][-1]['score_training_frequency']>=40 and
data['activity'][-1]['score_training_frequency']<70 and more_than_7500_steps==0) or
(more_than_7500_steps==1):
        return 1
        else:
        return 0
        except:
        print('An error occurred.')
```

In [53]:

```
def exercise_regularly_print():
        #This function is designed to print the advice for exercising regularly
        try:
        if exercise_regularly()==4:
        return print('It looks like you have been exercising enough lately. This is healthy
behavior that is good for your sleep quality. Keep it up!')
        elif exercise_regularly()==3:
        return print('It looks like you have been exercising enough lately, but there is room for
improvement. If you feel like your sleep quality has suffered, it might be better to exercise
more (regularly) or walk more than 7500 steps a day (more regularly). Keep it up!')
        else:
        return print('It looks like you have not been exercising enough lately. If you feel like
your sleep quality has suffered, it might be a good idea to start exercising more (regularly) or
start walking more than 7500 steps a day (more regularly).')
        except:
        print('An error occurred.')
```

In [54]:

```
#The actual printing
exercise_regularly_print()
```

```
It looks like you have not been exercising enough lately. If you feel like your sleep quality has
suffered, it might be a good idea to start exercising more (regularly) or start walking more
than 7500 steps a day (more regularly).
```

When should this advice be given?

Using the standard of the Oura ring, exercising medium activity for one hundred minutes on four days a week, a score of four should be the limit. If it is less, advice shall be given accordingly.

In [55]:

```
def provide_exercise_regularly():
        #This function is designed to determine whether the user should be advised to
exercise more regularly or not.
```

```
    try:
if exercise_regularly()<4:
    return True
    else:
    return False
    except:
    print('An error occurred.')
```

In [56]:

```
#Printing whether the advice should be provided to the user or not
print('The user should be advised to exercise more regularly:
{}'.format(provide_exercise_regularly()))
The user should be advised to exercise more regularly: True
```

Reduce time in bed to time slept

Kaiser Permanente, an American integrated care consortium, has provided sleep restriction therapy instructions (Kaiser Permanente, 2015). In these instructions they say that the maximum time that should be spent in bed should be determined by taking the average total time spent sleeping over the course of two weeks. Then, when the average has been determined, half an hour should be added.

In the 'sleep' data, there is a variable named 'total', which is the total time spent sleeping. This is the combination of all light-, deep- and REM sleep times added up together.

The second step in the instructions is to wake up at the same time every day, which we already determined earlier as part of the sleep hygiene.

According to the third step, all we have to do is add half an hour to the average total sleep time and subtract the total value from the waking time to get the time at which the user should go to bed.

As final warning the instructions also state that the time spent in bed should not be reduced to less than five and a half hours, so we should also check for that.

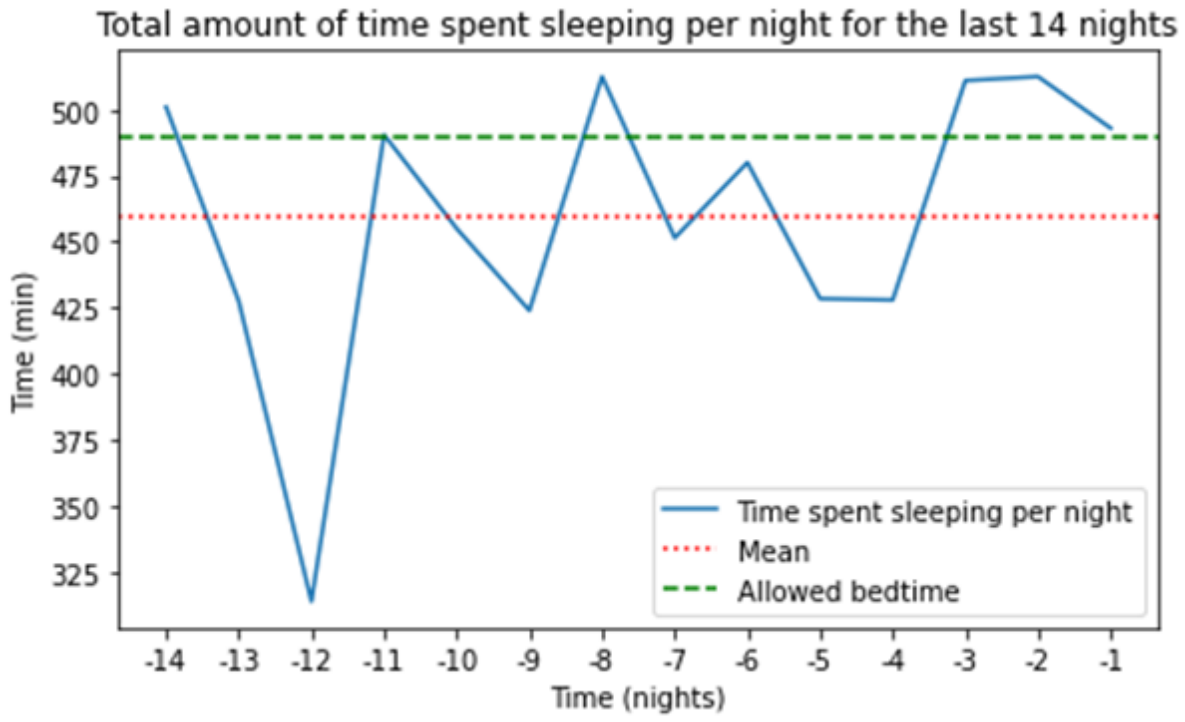
First, let's have a look at the total time slept during the last 14 days.

Inspecting the data

In [57]:

```
#Plotting the total amount of sleep per night in minutes for the last two weeks
plot(np.divide(total, 60)[-15:-1], [], [], 'Total amount of time spent sleeping per night for the
last 14 nights', [np.mean(np.divide(total, 60)[-15:-1]), np.mean(np.divide(total,
60)[-15:-1]+30)], [], [], False, 'Time (nights)', [np.arange(0, 14)],[np.arange(-14, 0)], [], 'Time
(min)', [], [], [], ['Time spent sleeping per night', 'Mean', 'Allowed bedtime'], [])

#Determining the average total sleep duration of the past two weeks in minutes
```

```
avg_total=int(np.mean(np.divide(total, 60)[-15:-1]))

#Adding half an hour to the average total sleep duration of the past two weeks
inc_avg_total=avg_total+30

#Printing the average sleep duration
print('The mean sleeping time per night over the last 14 days is: {}
minutes.'.format(avg_total))

#Ensuring the increased total average is not less than 5.5 hours (Kaiser Permanente, 2015;
Stanford Health Care, 2017)
if inc_avg_total>=(5.5*60):
        #printing the increased average sleep duration of the past two weeks
        print('The mean sleeping time of the last 14 days increased by 30 minutes is: {}
minutes.'.format(inc_avg_total))
else:
        #Setting the increased average total to 5.5 hours
        inc_avg_total=(5.5*60)
        #printing the increased average sleep duration of the past two weeks
        print('Since it is advised not to spend less than 5.5 hours in bed, your total average
bedtime is: {} minutes.'.format(inc_avg_total))
```




```
The mean sleeping time per night over the last 14 days is: 459 minutes.
The mean sleeping time of the last 14 days increased by 30 minutes is: 489 minutes.
```

To determine the ideal sleeping time, we should subtract this number from the ideal waking time determined earlier:

In [58]:

```
#Determining the sleep restriction bedtime
```

```
sleep_restriction_bedtime=wake_time()-inc_avg_total
print('The ideal bedtime for this user when applying sleep restriction therapy is:
{}.'.format(time.strftime("%H:%M", time.gmtime(sleep_restriction_bedtime*60))))
```
The ideal bedtime for this user when applying sleep restriction therapy is: 22:34.

Determining the advice

In [59]:

```
def sleep_restriction_bedtime():
        #This function is designed to determine the sleep restriction bedtime.
        try:
        #Creating a container for the total sleep duration per night of the last two weeks
        total=[]

    #Creating a list of all total sleep durations of the last two weeks
        for i in range(len(data['sleep'])-15, len(data['sleep'])-1):
        total.append(data['sleep'][i]['total'])

    #Taking the average total sleep duration of the last two weeks
        avg_total=int(np.mean(np.divide(total, 60)))

    #Adding half an hour to the average total sleep duration of the past two weeks
        inc_avg_total=avg_total+30

    #Ensuring the increased average total sleep duration of the past two weeks is 5.5 hours
at minimum
        if inc_avg_total<(5.5*60):
        inc_avg_total=(5.5*60)

    #Calculating the ideal bedtime based on the increased average total sleep duration
        sleep_restriction_bedtime=wake_time()-inc_avg_total

    return sleep_restriction_bedtime
        except:
        print('An error occurred.')
```

In [60]:

```
def sleep_restriction_bedtime_print():
        #This function is designed to print the advice for the sleep restriction bedtime.
        try:
        return print('The ideal sleeping time for this user when applying sleep restriction
therapy is at {}.'.format(time.strftime("%H:%M",
time.gmtime(sleep_restriction_bedtime()*60))))
        except:
        print('An error occurred.')
```

In [61]:

```
#The actual printing
sleep_restriction_bedtime_print()
```

The ideal sleeping time for this user when applying sleep restriction therapy is at 22:34.

When should this advice be given?

In [62]:

```
def provide_srbt():
        #This function is designed to determine whether the user should be advised to follow
sleep restriction or not.
        try:
        #Creating a container for the sleep efficiency per night of the last week
        efficiency=[]

      #Creating a list of the sleep efficiency per night for the last week
        for i in data['sleep'][-8:-1]:
        efficiency.append(i['efficiency'])

      #Determining the average sleep efficiency of last week
        avg_efficiency=np.mean(efficiency)

      if avg_efficiency<80 or avg_efficiency>85:
        return True
        else:
        return False
        except:
        print('An error occurred.')
```

In [63]:

```
#Printing whether the advice should be provided to the user or not
print('The user should be advised to follow sleep restriction therapy:
{}'.format(provide_srbt()))
```

The user should be advised to follow sleep restriction therapy: False

Increment time in bed in steps of 15 to 30 minutes

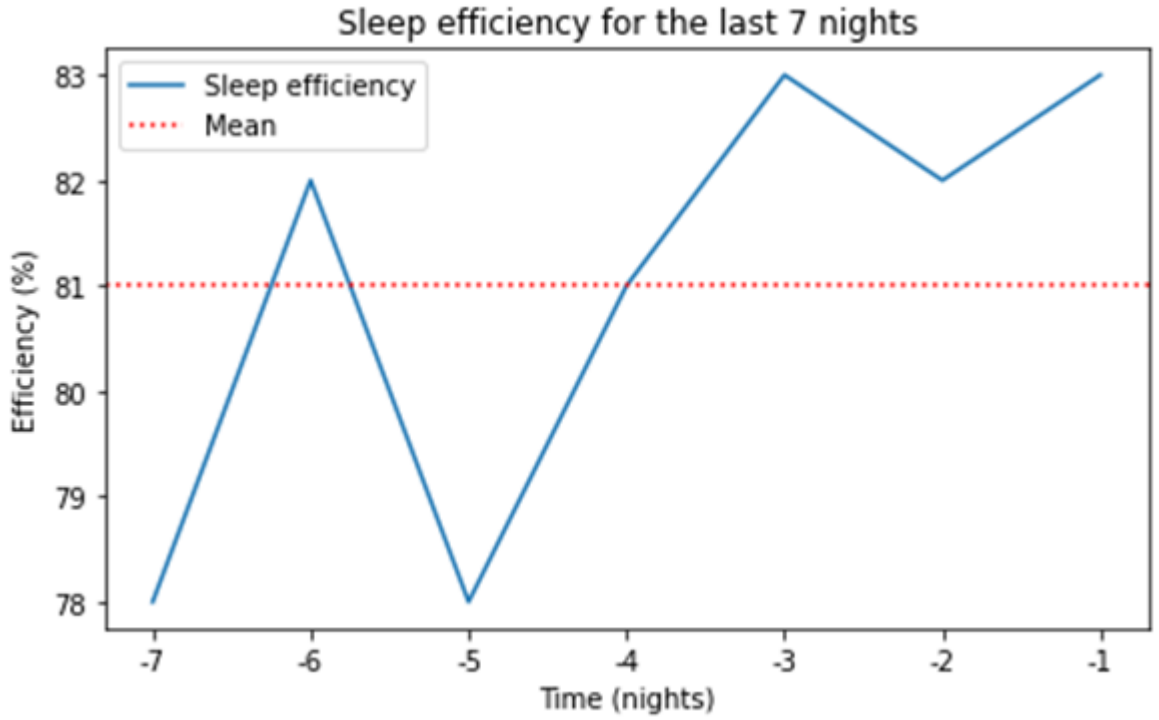
Incrementing the allowed bedtime is step 4 of the instructions. After the first two weeks the user is allowed to start incrementing or decreasing the allowed bedtime at the start of a week, based on how tired they feel during the day. However, when using measured data, we can automate this using some criterion.

According to Stanford Health Care (Stanford Health Care, 2017), an important criterion for the quality of sleep, and therefore also for sleep restriction, is the sleep efficiency. If the average sleep efficiency of a week is equal to 85% or more, the sleep time can be incremented by 15 to 30 minutes, according to the instructions (Kaiser Permanente, 2015). If the average sleep efficiency of a week is less than 80% however, the bedtime is reduced. If it is between these values, the allowed sleep time is good as it is.

Inspecting the data

In [64]:

```
#Plotting the sleep efficiency per night for the past week
plot(efficiency[-8:-1], [], [], 'Sleep efficiency for the last 7 nights', [np.mean(efficiency[-8:-1])],
[], [], False, 'Time (nights)', [np.arange(0, 7)], [np.arange(-7, 0)], [], 'Efficiency (%)', [], [], [],
['Sleep efficiency', 'Mean'], [])
```



Determining the advice

In [65]:

```
def new_sleep_restriction_bedtime():
        #This function is designed to determine the new sleep restriction bedtime.
        try:
        #Creating a container for the sleep efficiency per night of the last week
        efficiency=[]

    #Creating a list of the sleep efficiency per night for the last week
        for i in data['sleep'][-8:-1]:
        efficiency.append(i['efficiency'])

    #Determining the average sleep efficiency of last week
        avg_efficiency=np.mean(efficiency)

    #Printing the average sleep efficiency of last week
        print('The average sleep efficiency of last week is: {}.'.format(avg_efficiency))
```

```
        #Determining the new sleep restriction bedtimes based on the sleep efficiency of last
week
        if avg_efficiency<80:
        new_srbt1=sleep_restriction_bedtime()+15
        new_srbt2=sleep_restriction_bedtime()+30

        return new_srbt1, new_srbt2
        elif avg_efficiency>=80 and avg_efficiency<85:
        new_srbt1=sleep_restriction_bedtime()
        new_srbt2=sleep_restriction_bedtime()

        return new_srbt1, new_srbt2
        else:
        new_srbt1=sleep_restriction_bedtime()-30
        new_srbt2=sleep_restriction_bedtime()-15

        return new_srbt1, new_srbt2
    except:
        print('An error occurred.')
```

In [66]:

```
def new_sleep_restriction_bedtime_print():
        #This function is designed to print the advice for the new sleep restriction bedtime.
        try:
        nsrbt1, nsrbt2=new_sleep_restriction_bedtime()
        if nsrbt1==sleep_restriction_bedtime() and nsrbt2==sleep_restriction_bedtime():
        print('The user\'s allowed bedtime is good.\nThis is at
{}.'.format(time.strftime("%H:%M", time.gmtime(nsrbt1*60))))
        elif nsrbt1<sleep_restriction_bedtime() and nsrbt2<sleep_restriction_bedtime():
        print('The user\'s allowed bedtime may be increased by 15 to 30 minutes, depending
on how tired they feel during the day.\nThis is between {} and {} on the
clock.'.format(time.strftime("%H:%M", time.gmtime(nsrbt1*60)), time.strftime("%H:%M",
time.gmtime(nsrbt2*60))))
        else:
        return print('The user\'s allowed bedtime should be further restricted by 15 to 30
minutes, depending on how tired they feel.\nThis is between {} and {} on the
clock.'.format(time.strftime("%H:%M", time.gmtime(nsrbt1*60)), time.strftime("%H:%M",
time.gmtime(nsrbt2*60))))
        except:
        print('An error occurred.')
```

In [67]:

```
#The actual printing
new_sleep_restriction_bedtime_print()
```

```
The average sleep efficiency of last week is: 81.0.
The user's allowed bedtime is good.
```

This is at 22:34.

When should this advice be given?

This function should only be ran two weeks after initiating sleep restriction therapy for the user, and every week after that. This can be done but implementing that is outside the scope of this research.

In [68]:

```
def provide_nsrbt():
        #This function is designed to determine whether the user should be advised a new
sleep restriction bedtime or not.
        #This function should be ran two weeks after initiating sleep restriction therapy and
every week after, until the new sleep restriction bedtime stays the same for a longer period
of time (in the range of months).
        try:
        efficiency=[]

      #Creating a list of the sleep efficiency per night for the last week
        for i in data['sleep'][-8:-1]:
        efficiency.append(i['efficiency'])

      #Determining the average sleep efficiency of last week
        avg_efficiency=np.mean(efficiency)

      #Ensuring that the new sleep restriction bedtime cannot be less than 5.5 hours of sleep
        if (avg_efficiency<80 and
abs(wake_time()-sleep_restriction_bedtime())>=((5.5*60)+30)) or avg_efficiency>85:
        return True
        else:
        return False
        except:
        print('An error occurred.')
```

In [69]:

```
#Printing whether the advice should be provided to the user or not
print('The user should be advised a new sleep restriction bedtime:
{}'.format(provide_nsrbt()))
```

```
The user should be advised a new sleep restriction bedtime: False
```

Putting everything together

In [70]:

```
counter, _=gaps()
if counter>0:
        gaps_print()
        safe_date_dif_last_gap_print()
```

```
print('')
wake_time_print()
print('')
no_napping_print()
print('')
exercise_regularly_print()
print('')
sleep_restriction_bedtime_print()
print('')
new_sleep_restriction_bedtime_print()
```
```
There are 17 gaps in the data, comprising a total of 36 days.
The average gap size is approximately 2.0 days and the largest gap size is 16 days.
The missing data does not occur within the last two weeks. It is safe to trust any advice based on this data.

The advised rising time is: 06:43.
This advice can be rounded to the nearest quarter if wished for.

It looks like you have not been taking naps lately. This is healthy behavior that is good for your sleep quality. Keep it up!

It looks like you have not been exercising enough lately. If you feel like your sleep quality has suffered, it might be a good idea to start exercising more (regularly) or start walking more than 7500 steps a day (more regularly).

The ideal sleeping time for this user when applying sleep restriction therapy is at 22:34.

The average sleep efficiency of last week is: 81.0.
The user's allowed bedtime is good.
This is at 22:34.
```

Automating the selection of relevant micro-interventions

In [71]:

```
def relevant_selection():
    #This function is designed to provide a selection of micro-interventions that are relevant for the user
    try:
    #Determining if the advice provided are trustworthy or not
    counter, _=gaps()
    if counter>0:
    gaps_print()
    safe_date_dif_last_gap_print()
```

```
    print('')

#Determining which advice should be provided and which not
    #If yes the advice shall be printed
    if provide_wake_time():
    wake_time_print()
    if provide_no_napping():
    no_napping_print()
    if provide_exercise_regularly():
    exercise_regularly_print()
    if provide_srbt():
    sleep_restriction_bedtime_print()
    if provide_nsrbt():
    new_sleep_restriction_bedtime_print()

return print('')
    except:
    print('An error occurred')
```

In [72]:

```
#The actual selection
relevant_selection()
```

```
There are 17 gaps in the data, comprising a total of 36 days.
The average gap size is approximately 2.0 days, and the largest gap size is 16 days.
The missing data does not occur within the last two weeks. It is safe to trust any advice based on this data.

It looks like you have not been exercising enough lately. If you feel like your sleep quality has suffered, it might be a good idea to start exercising more (regularly) or start walking more than 7500 steps a day (more regularly).
```

Conclusion

For the following text, the micro-interventions have been linked to numbers, starting from the top in the framework working to the bottom, taking the end node of every branch as a micro-intervention.

The data collected through the Oura ring can provide added value for micro interventions: 6, 7, 25, 32 and 33 and in consultation with a psychologist also for micro interventions 35 and 47. This sums up to a guaranteed total of seven out of forty-eight micro-interventions, which equals to about 14.6%.

This is probably due to the fact that CBT-I largely consists of a few educational aspects (Psychoeducation, 1-4; Sleep schedule, 5; Sleep activities: 8, 16-18; Lifestyle education, 26-31: Total of 15 micro interventions) and some therapeutical aspects (Cognitive restructuring, 34, 36-46, 48: Total of thirteen micro-interventions), which sum up to a total of twenty-eight micro-interventions already, which is about

58.3%. From the remaining twenty micro interventions (42.7%), seven (14.6% out of total) can be accounted for according to this research, leading to a 35% gain ratio.

This means that there is quite some room for improvement. This concerns micro-interventions: 9-15 and 19-24. This is a total of thirteen micro interventions.

The first of these is the intervention 'get out of bed when unable to sleep'. For this intervention we determined that more information is necessary in order to draw a conclusion about the accuracy of the activity data, from which we would be able to conclude whether or not the user stayed in bed whilst being awake.

The next six are the relaxation techniques that can possibly be measured using the sensors of the Oura ring, but that currently only work at night. Five of these can hypothetically be recorded by the Oura ring when performed whilst in bed right before going to sleep. These are micro-interventions 10-13 and 15. More research is needed to determine this, however.

The other six are sleep environment micro-interventions. These can be separated into three subgroups: The 'comfortability' intervention, the 'body temperature' intervention and the rest. Comfortability can be measured indirectly through how much the user moves during sleep. However, there are a lot of other factors that also influence how much a person moves during their sleep. Furthermore, it is not guaranteed that the user also moves his or her hand or arm, at which the Oura ring is located. Therefore, the data that the advice of this micro-intervention would be based upon is circumstantial at best.

The body temperature is being measured by the Oura ring, but as a deviation from the average nighttime body temperature of the user. While this most certainly has some advantages with respect to drawing conclusions based on this data, it makes it impossible to give advice regarding the sleep environment, because when the user is too hot or too cold during his sleep, every night, then this will become the average nighttime body temperature.

For the other four micro interventions holds the same: these are not usually measured by wearables directly, but they can be measured in other ways, using additional technological devices that the user might already own (e.g. thermometer, electronic air filtering system, microphone, light sensor, etc.), combined with self-report.

So concludingly I'd like to say that there are seven micro-interventions (6, 7, 25, 32, 33, 35 & 47) out of fifteen micro interventions that can be administered digitally by using the data of the Oura ring (6, 7, 9-15, 24, 25, 32, 33, 35 and 47), with a possible increase to twenty micro-interventions (including 19-23) when combined with self-report and other electronical instruments, some of which may already be present in the house (for example, a smartphone contains a microphone and a light sensor).

References

American Academy of Sleep Medicine. (2017). Two week sleep diary. National Institute for Health and Care Excellence, 1, 2. http://yoursleep.aasmnet.org/pdf/sleepdiary.pdf

Caddy, B. (2021). Are wearables the future of blood pressure monitoring? TechRadar. https://www.techradar.com/news/are-wearables-the-future-of-blood-pressure-monitoring

Centre d’études sur le stress humain. (2007). How to measure stress in humans. 1–39. https://humanstress.ca/Documents/pdf/Mesures-physiologiques/CESH_howMesureStress-MB.pdf

Chambers, J. M., Cleveland, W. S., Kleiner, B., & Tukey, P. A. (1983). Notched Boxplots. Graphical Methods for Data Analysis (pp. 60–63). Wadsworth International Group. https://docs.google.com/viewer?a=v&pid=sites&srcid=ZGVmYXVsdGRvbWFpbnxkYXZpZHNzdGF0aXN0aWNzfGd4OjYxNTMzNTE4ZmY4Y2ZhNGQ

Daylight Saving Time 2019 in the Netherlands. (n.d.). Retrieved June 15, 2021, from https://www.timeanddate.com/time/change/netherlands?year=2019

Daylight Saving Time 2020 in the Netherlands. (n.d.). Retrieved June 15, 2021, from https://www.timeanddate.com/time/change/netherlands?year=2020

Daylight Saving Time 2021 in the Netherlands. (n.d.). Retrieved June 15, 2021, from https://www.timeanddate.com/time/change/netherlands?year=2021

Journal of Clinical Sleep Medicine. (2020). In Journal of Clinical Sleep Medicine (Vol. 6, Issue 3). https://jcsm.aasm.org/info/about-jcsm

Kaiser Permanente. (2015). What is Sleep Restriction Therapy? https://thrive.kaiserpermanente.org/care-near-you/northern-california/sanjose/wp-content/uploads/sites/7/2015/10/sleep-restriction-rev2_tcm28-557887.pdf

Kim, A. Y., Jang, E. H., Choi, K. W., Jeon, H. J., Byun, S., Sim, J. Y., Choi, J. H., & Yu, H. Y. (2019). Skin conductance responses in Major Depressive Disorder (MDD) under mental arithmetic stress. PLoS ONE, 14(4). https://doi.org/10.1371/journal.pone.0213140

Lee, I. M., Shiroma, E. J., Kamada, M., Bassett, D. R., Matthews, C. E., & Buring, J. E. (2019). Association of Step Volume and Intensity with All-Cause Mortality in Older Women. JAMA Internal Medicine, 179(8), 1105–1112. https://doi.org/10.1001/jamainternmed.2019.0899

Morin, C. M., & Espie, C. A. (2003). Insomnia A Clinical Guide to Assessment and Treatment. Kluwer Academic/Plenum Publishers.

Ōura Health Oy. (n.d.). An Introduction to Body Temperature. Oura Help. Retrieved June 17, 2021, from https://support.ouraring.com/hc/en-us/articles/360025587493-An-Introduction-to-Body-Temperature

Ōura Health Oy. (n.d.). API Documentation. Retrieved June 11, 2021,

from https://cloud.ouraring.com/docs/

Ōura Health Oy. (n.d.). Use Apple Health with Oura. Oura Help. Retrieved June 10, 2021, from https://support.ouraring.com/hc/en-us/articles/360025438734-Use-Apple-Health-with-Oura

Oura Team. (2021). Oura's Nap Detection Supports Diverse Sleeping Patterns. The Pulse Blog. https://ouraring.com/blog/nap_detection/

Sawh, M. (2019). Getting all emotional: Wearables that are trying to monitor how we feel. Wareable. https://www.wareable.com/wearable-tech/wearables-that-track-emotion-7278

Stanford Health Care. (2017). Sleep Restriction and CBTI. https://stanfordhealthcare.org/medical-treatments/c/cognitive-behavioral-therapy-insomnia/procedures/sleep-restriction.html

Appendix G

This code is part of the bachelor thesis project: Personalizing Digital Cognitive Behavioral Therapy for Insomnia Through Automation of Micro-Intervention Selection Based on Data from Wearable Techology; A Case Study on the Oura Ring

Author Information

Author: Sjors Weggeman\ Student number: s4799771\ University: Radboud University Nijmegen\ Email address: sjors.weggeman@student.ru.nl

Imports

In [1]:

```
#Importing the necessary packages
import json
import matplotlib
import matplotlib.pyplot as plt
import numpy as np
import re
import time

from datetime import date
from statsmodels import robust
```

Loading the data

In [2]:

```
#Opening the file
file=open('oura1.json')

#Loading the file into a workable dataframe
data=json.load(file)
```

Inspecting the data

In [3]:

```
#Checking the length of the data
for i in data:
```

```
        print('The \'{}\' data contains {} entries.'.format(i, len(data[i])))
```
```
The 'activity' data contains 46 entries.
The 'readiness' data contains 44 entries.
The 'restful_periods' data contains 2 entries.
The 'sleep' data contains 44 entries.
```

Preprocessing

In [4]:

```
def time_min_conversion(time_data):
        #This function is designed to extract the digital time from the 'bedtime_start' and
'bedtime_end' variables and convert it to minutes.
   #Accordingly, time_data must be of the form string and can only have values
'bedtime_start' and 'bedtime_end'.
        try:
        #Container for the times in minutes
        times_m=[]

        #Performing steps for every entry in the 'sleep' data
        for i in data['sleep']:
        #Extracting the time out of the concatenated data
        time=re.split(r"[-T+:]", i[time_data])[3:6]

        #Transforming the time to minutes
        if int(time[2])<30:
                time_m=int(time[0])*60+int(time[1])
        else:
                time_m=int(time[0])*60+int(time[1])+1

        #Storing the time in the containers
        times_m.append(time_m)

      return times_m
        except:
        print('An error occurred.')
```

In [5]:

```
#The actual conversion
sleeping_times_m=time_min_conversion('bedtime_start')
rising_times_m=time_min_conversion('bedtime_end')
```

In [6]:

```
def date_conversion(date2conv):
```

```
        #This function is designed to convert a date provided in string format
('year-month-day') to int format (year, month, day) so it can be used by the 'date' function
from datetime.
        try:
        #Extracting the date data from the string
        conv_date_str=re.split(r"[-]",date2conv)

        #Converting the string representations to integers
        conv_date_int=[int(i) for i in conv_date_str]

    return conv_date_int
        except:
        print('An error occurred.')
```

In [7]:

```
def date_difference(date1,date2):
        #This function is designed to calculate the number of days difference between two
given dates.
        #Hereby the earliest date needs to be provided first, otherwise a negative number will
be returned.
        try:
        #Converting the provided dates to comparable formats
        d1=date(date1[0], date1[1], date1[2])
        d2=date(date2[0], date2[1], date2[2])

    #Comparing the dates
        dif=d2-d1

    return dif.days
        except:
        print('An error occurred.')
```

In [8]:

```
dd=date_difference(date_conversion(data['sleep'][0]['summary_date']),
date_conversion(data['sleep'][-1]['summary_date']))+1
print('The difference between {} and {} (included) is {} days.\nThe dataset contains {} entries
in the \'sleep\' data.\nThis means that there are {} missing days in the \'sleep\' data,
assuming the dataset does not contain any duplicate dates.'
        .format(data['sleep'][0]['summary_date'], data['sleep'][-1]['summary_date'], dd,
len(data['sleep']), abs(len(data['sleep'])-dd)))
```

```
The difference between 2021-04-17 and 2021-05-31 (included) is 45 days.
The dataset contains 44 entries in the 'sleep' data.
This means that there are 1 missing days in the 'sleep' data, assuming the dataset does not
contain any duplicate dates.
```

In [9]:

```
def gaps():
        #This function is designed to find the number of gaps in the data and their respective
sizes, where a gap is one or more consecutive missing days.
        try:
        #Creating a counter for the number of gaps and a container for the sizes of the gaps
      counter=0
        diff=[]

      #Working through all the gaps
        for i in range(0,len(data['sleep'])-1):
        if date_difference(date_conversion(data['sleep'][i]['summary_date']),
                           date_conversion(data['sleep'][i+1]['summary_date']))>1:

            #Determining the size of the current gap
                diff.append(date_difference(date_conversion(data['sleep'][i]['summary_date']),
                                    date_conversion(data['sleep'][i+1]['summary_date']))-1)

            #Determining the total number of gaps
                counter+=1

      return counter, diff
        except:
        print('An error occurred.')
```

In [10]:

```
def gaps_print():
        #This function is designed to print the results from the 'gaps' function in a nice format.
        try:
        c, d=gaps()
        if c==1:
        return print('There is {} gap in the data, comprising a total of {} days.'.format(c,
np.sum(d)))
        elif c>1:
        return print('There are {} gaps in the data, comprising a total of {} days.\nThe average
gap size is approximately {} days and the largest gap size is {} days.'.format(c, np.sum(d),
np.round(np.mean(d)), np.max(d)))
        else:
        return print('There are no gaps in the data.')
        except:
        print('An error occurred.')
```

In [11]:

```
#The actual printing
gaps_print()
```

```
There is 1 gap in the data, comprising a total of 1 days.
```

In [12]:

```
def last_gap_date():
        #This function has been designed to determine the first date after the last gap in the
data.
        try:
        dd=0
        last_gap_date=''

      #Working through all the gaps
        for i in range(0,len(data['sleep'])-1):
        dd=date_difference(date_conversion(data['sleep'][i]['summary_date']),
date_conversion(data['sleep'][i+1]['summary_date']))-1

        #Determining the first date after the last gap in the data
        if dd>0:
                last_gap_date=data['sleep'][i+1]['summary_date']

      return last_gap_date
        except:
        print('An error occurred.')
```

In [13]:

```
def pre_last_gap_date():
        #This function has been designed to determine the first date after the second to last
gap in the data.
        try:
        c,d=gaps()
        if c>1:
        dd=0
        pre_last_gap_date=''

        #Working through all the gaps
        for i in range(0,len(data['sleep'])-1):
                dd=date_difference(date_conversion(data['sleep'][i]['summary_date']),
date_conversion(data['sleep'][i+1]['summary_date']))-1
                #Determining the previous gap
                if dd>0:
                c-=1
                #Determining the first date after the pre-last gap in the data
                if c==0:
                pre_last_gap_date=data['sleep'][i+1]['summary_date']

        return pre_last_gap_date
        except:
        print('An error occurred.')
```

In [14]:

```
def safe_date_dif_last_gap_print():
        #This function has been designed to print whether it is safe to trust the advices based on this data, based on the size and location of the gaps with respect to the day of the last 'sleep' data-entry contained in the dataset.
        try:
        c,d=gaps()
        greatest_gap=np.max(d)
         if pre_last_gap_date():
        pldd=date_difference(date_conversion(pre_last_gap_date()), date_conversion(data['sleep'][-1]['summary_date']))+1
        else:
        pldd=15
        ldd=date_difference(date_conversion(last_gap_date()), date_conversion(data['sleep'][-1]['summary_date']))+1
        if pldd<=14:
        print('The missing data occurs within the last two weeks on more than one occasion. It is unwise to trust any advices based on this data.')
        elif pldd>14 and ldd<=14 and greatest_gap>1:
        print('The missing data occurs within the last two weeks on one occasion, but is larger than one day. It is unwise to trust any advices based on this data.')
        elif pldd>14 and ldd<=14 and greatest_gap<1:
        print('The missing data occurs within the last two weeks on one occasion, but is not larger than one day. It is safe to trust any advices based on this data.')
        else:
        print('The missing data does not occur within the last two weeks. It is safe to trust any advices based on this data.')
        except:
        print('An error occurred.')
```

In [15]:

```
#The actual printing
safe_date_dif_last_gap_print()
```

```
The missing data does not occur within the last two weeks. It is safe to trust any advices based on this data.
```

Checking for trends in the data based on Daylight Saving Time (DST)

Since this dataset comprises little more than a month it is not worth checking if there are any trends caused by DST.

I will still need to plot data:

In [16]:

```
def plot(data_p1_x, data_p1_y, data_p2, title, axh, axv, col, spec_labs, xlab, xmajloc, xticklabs, xlim, ylab, ymajloc, yticklabs, ylim, legend, figtext):
```

```
        #This function is designed to make it possible to plot a graph using just one line of
code.
    try:
        if col:
        colours=col
        linestyles=['solid']
        else:
        colours=['r', 'g', 'm']
        linestyles=['dotted', 'dashed', 'dashdot']

        fig,ax=plt.subplots(figsize=(7, 4))
        if not data_p1_y:
        ax.plot(data_p1_x)
        else:
        ax.plot(data_p1_x, data_p1_y)
        if(data_p2):
        ax.plot(data_p2)
        ax.set_title(title)
        if axh:
        for j in range(0,len(axh)):
                ax.axhline(axh[j], c=colours[j%len(colours)],
linestyle=linestyles[j%len(linestyles)])
        if axv:
        for k in range(0,len(axv)):
                ax.axvline(axv[k], c=colours[k%len(colours)])
        ax.set_xlabel(xlab)
        if spec_labs:
        if xmajloc:
                ax.xaxis.set_major_locator(matplotlib.ticker.FixedLocator(xmajloc))
                if xticklabs:
                ax.set_xticklabels(xticklabs)
        if ymajloc:
                ax.yaxis.set_major_locator(matplotlib.ticker.FixedLocator(ymajloc))
                if yticklabs:
                ax.set_yticklabels(yticklabs)
         else:
        if xmajloc:
                ax.xaxis.set_major_locator(matplotlib.ticker.FixedLocator(xmajloc[0]))
                if xticklabs:
                ax.set_xticklabels(xticklabs[0])
        if ymajloc:
                ax.yaxis.set_major_locator(matplotlib.ticker.FixedLocator(ymajloc[0]))
                if yticklabs:
                ax.set_yticklabels(yticklabs[0])
        if xlim:
        ax.set_xlim(xlim)
        ax.set_ylabel(ylab)
        if ylim:
```

```
        ax.set_ylim(ylim)
        if legend:
        ax.legend(legend)
        if figtext:
        fig.text(figtext[0], figtext[1], figtext[2])
        plt.show()
        except:
        print('An error occurred.')
```

Basing Micro Interventions on Data

Wake at the same time every day

In [17]:

```
def wake_time():
        #This function is designed to determine the ideal wake time in minutes.
        try:
        #Determining the ideal wake time in minutes
        wake_time=int(np.median(rising_times_m[-15:-1]))

    return wake_time
        except:
        print('An error occurred.')
```

In [18]:

```
def wake_time_print():
        #This function is designed to print the advice for the ideal wake time, with the ideal
wake time in digital format.
        try:
        return print('The advised rising time is: {}.\nThis advice can be rounded to the nearest
quarter if wished for.'.format(time.strftime("%H:%M", time.gmtime(wake_time()*60))))
        except:
        print('An error occurred.')
```

In [19]:

```
def provide_wake_time():
        #This function is designed to determine whether the user should be advised to wake
at the same time every day.
        try:
        rising_times_m14=[]

    #Converting the rising times to minutes for if that had not been done yet
        rising_times_m=time_min_conversion('bedtime_end')

    #Storing the rising times of the last two weeks
```

```
        for i in range(0, 14):
        rising_times_m14.append(rising_times_m[-15+i])

    #Determining the median of the rising times of the last two weeks
        med=np.median(rising_times_m14)

    #Determining the median absolute deviation for the rising times of the last two weeks

    mad=robust.mad(rising_times_m14)

    if rising_times_m14[-1]<(med-mad) or rising_times_m14[-1]>(med+mad):
        return True
        else:
        return False
        except:
        print('An error occurred.')
```

No Napping

In [20]:

```
def within_14_days_check(conv_date):
        #This function is designed to check if a date lies within 14 days of the day of the last
'sleep' data-entry contained in the dataset.
        try:
        #Getting the last date from the dataset
        ref_date=data['sleep'][-1]['summary_date']

        #Converting the dates for comparison
        conv_ref_date=date_conversion(ref_date)

    #Comparing the dates
        if date_difference(conv_date, conv_ref_date)>=0 and date_difference(conv_date,
conv_ref_date)<=14:
        return True
        else:
        return False
        except:
        print('An error occurred.')
```

In [21]:

```
def no_napping():
        #This function is designed to determine whether any naps have been taken in the
last two weeks and returns a list of them.
        try:
        #Container for naps
        naps=[]
```

```
    #Converting sleeping times to minutes
        sleeping_times_m=time_min_conversion('bedtime_start')

    #Adding unconfirmed naps within the last two weeks
        for i in data['restful_periods']:
        if within_14_days_check(date_conversion(i['summary_date'])):
                if i['duration']>=(15*60) and i['duration']<(3*60*60):
                naps.append(i)

    #Adding confirmed naps within the last two weeks based on period id and unusual
sleeping times
        for j in range(0, len(data['sleep'])):
        if j>len(data['sleep'])-15:
                if data['sleep'][j]['period_id']>0 and sleeping_times_m[j]>=(7*60) and
sleeping_times_m[j]<(19*60):
                naps.append(data['sleep'][j])

    return naps
        except:
        print('An error occurred')
```

In [22]:

```
def no_napping_print():
        #This function is designed to print the advice for no napping
        try:
        #Checking if there are any naps and giving advise if any naps have been found
    if not len(no_napping())==0:
        return print('It looks like you have been taking naps lately. If you feel like your sleep
quality has suffered it might be better not to take naps anymore.\nThese are the naps:
{}'.format(no_napping()))
        else:
        return print('It looks like you have not been taking naps lately. This is healthy
behavior that is good for your sleep quality. Keep it up!')
        except:
        print('An error occurred')
```

In [23]:

```
def provide_no_napping():
        #This function is designed to determine whether the user should be advised not to
nap anymore.
        try:
        if not len(no_napping())==0:
        return True
        else:
        return False
```

```
except:
print('An error occurred.')
```

Get out of bed when unable to sleep

In the creation of this automation process it was found that more information was needed to determine whether the medium activity level determined by the Oura team indeed corresponds to walking level, or that it is more likely that the low activity corresponds to walking. The user of the first dataset only wore the ring at night, which might have influenced the accuracy of the activity data collection. Through which means this activity data was collected for him remains unknown, but it is most likely collected through another wearable and its corresponding application, which is connected to Apple Health, the same application the Oura ring and its corresponding application are connected to. Or it might just plainly have been recorded by Apple Health using the phone of the user. Regardless of the way through which the activity data was recorded, it remains unclear how it was recorded exactly and thus its accuracy remains unclear too. Therefore it is important to also have a look at the data from the other two users, who said to wear the ring day and night.

To remind you of what we were looking at:

For getting out of bed when unable to sleep we are going to look at three aspects:

1. The sleep onset latency: How long does the person stay awake before falling asleep after going to bed.
2. If and for how long the user woke up during the night and whether he stayed in bed or not.
3. How long the user was awake before going out of bed.

These aspects are all combined into a score: The sleep efficiency.

The most reliable way to determine whether the user stayed in bed or not is by self-report, so if an application is being created questions should be asked when it was detected that the user was awake for a prolonged period of time during the night. Then the user can specify his course of actions or the lack there off.

A little bit less reliable, but technological data collection method is the activity measurement. This data is collected in 5 minute intervals and has been separated into 5 classes:

0. Non-wear \ 1. Rest \ 2. Inactive \ 3. Low activity \ 4. Medium activity \ 5. High activity

This data collection starts from 4am and lasts until 4am the next day, hence I'll have to combine the last bit of data from one day with the first bit of data from the next day to get the data for that night.

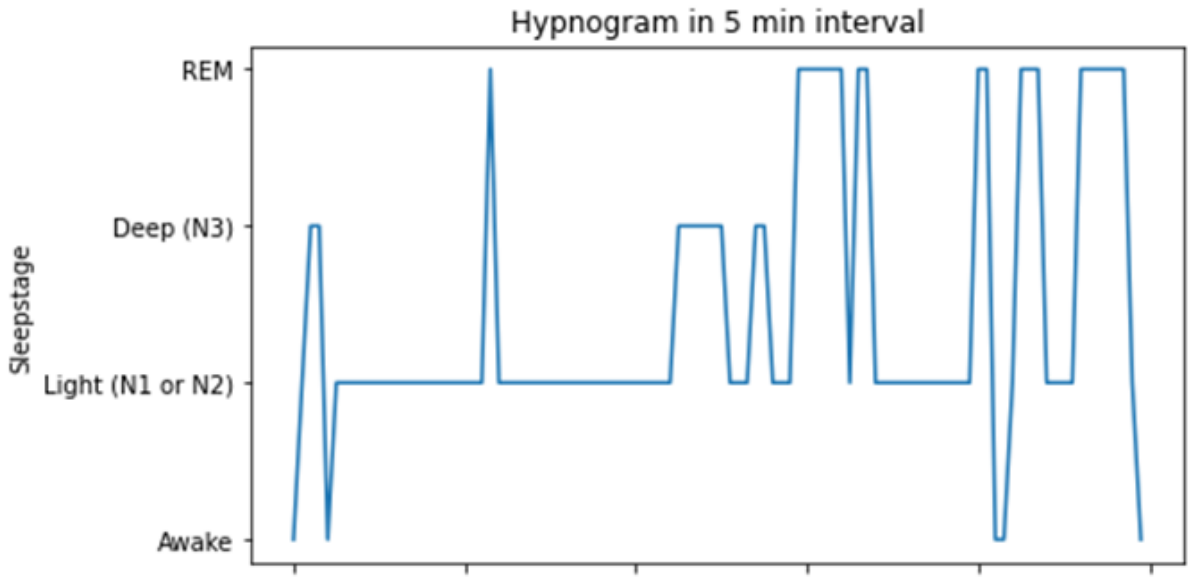
So first we have to find out when the user was awake, which is contained in the hypnograms:

Inspecting the data

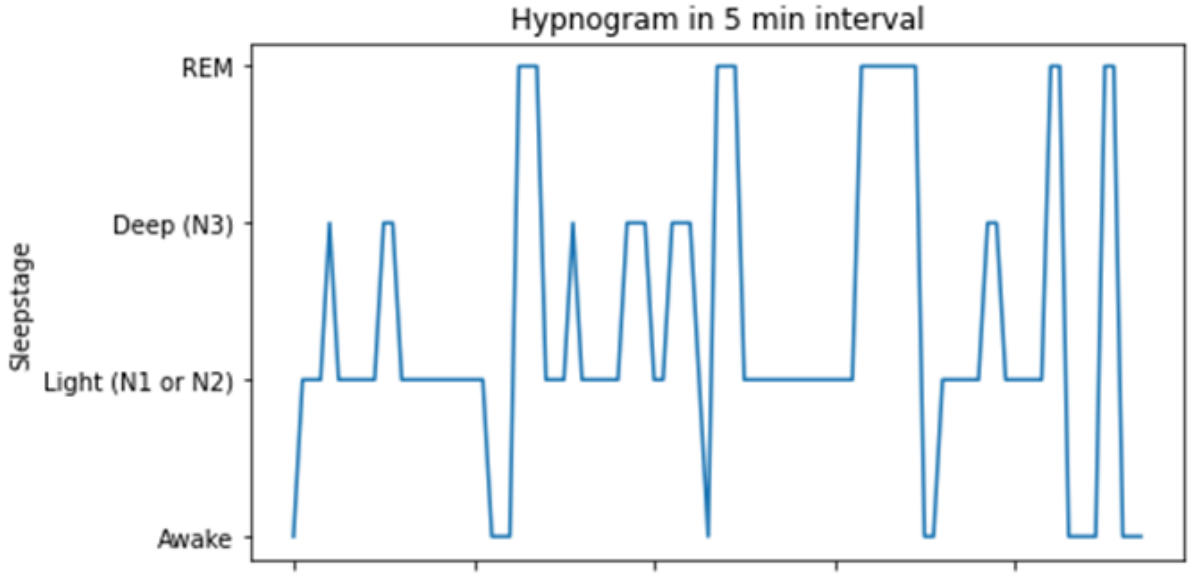
In [24]:

```
#Plotting the hypnograms for the last week
counter=-1
#Storing the hypnograms for later use
```

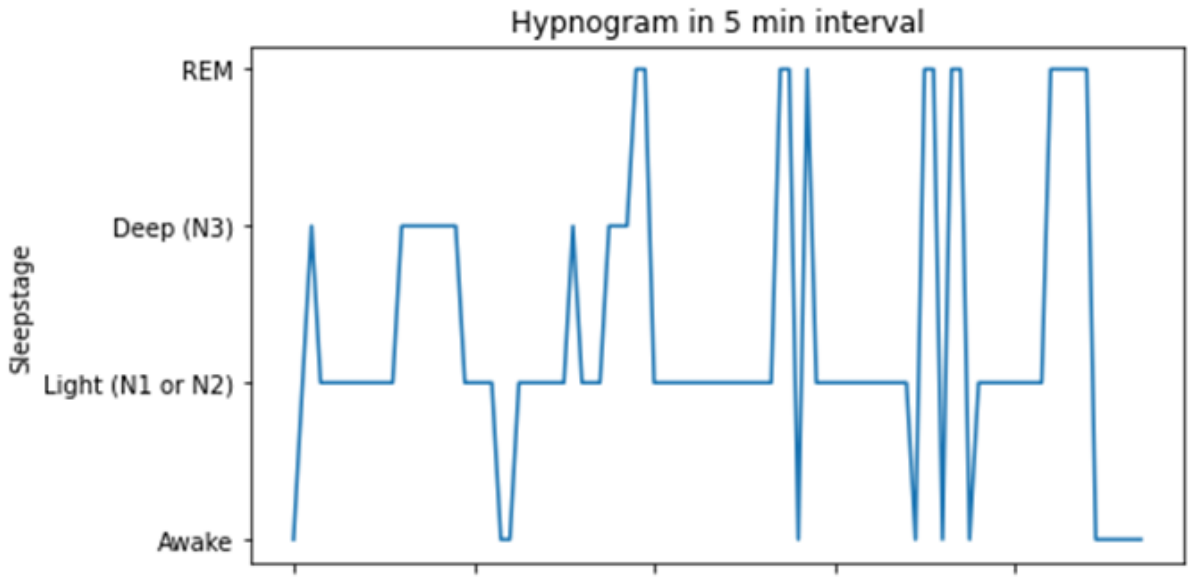
```
hgs5=[]
for i in data['sleep']:
        counter+=1
        hg5_conc=i['hypnogram_5min']
        hg5_split=[]
        if counter>(len(data['sleep'])-8):
        for j in hg5_conc:
        hg5_split.append(j)
        hgs5.append(hg5_split)
        plot(hg5_split, [], [], 'Hypnogram in 5 min interval', [], [], [], False, 'Time (min)',
[np.arange(0, 161, 20)],[np.arange(0, 801, 100)], [], 'Sleepstage', [np.arange(0, 4)], [['Awake',
'Light (N1 or N2)', 'Deep (N3)', 'REM']],[], [], [0.125, -0.1, '^\nBedtime start: {}\nTotal bedtime:
{}min\nOnset latency: {}min'.format(i['bedtime_start'], int(i['duration']/60),
i['onset_latency']/60)])
```



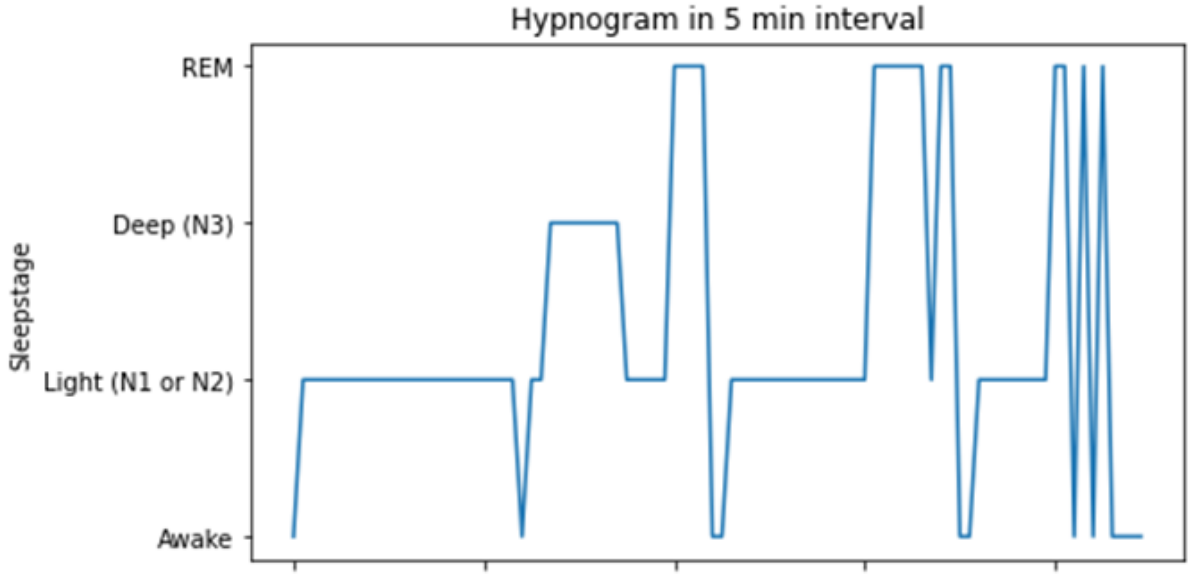
Bedtime start: 2021-05-24T23:04:20+02:00
Total bedtime: 499min
Onset latency: 3.5min



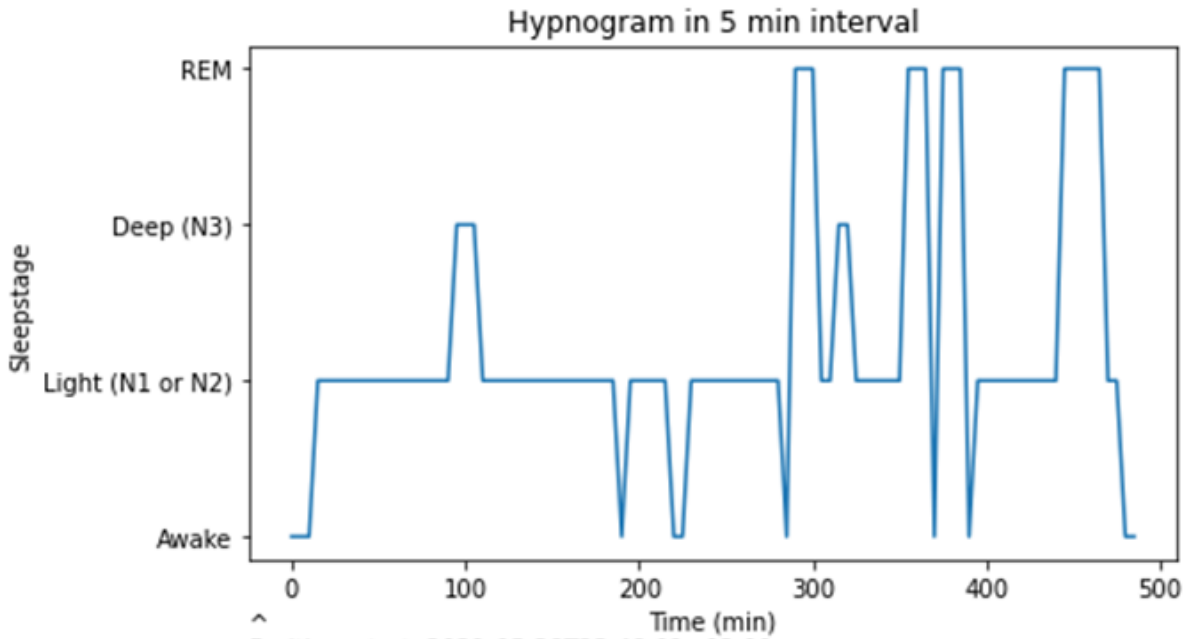
Bedtime start: 2021-05-25T23:14:33+02:00
Total bedtime: 472min
Onset latency: 4.0min



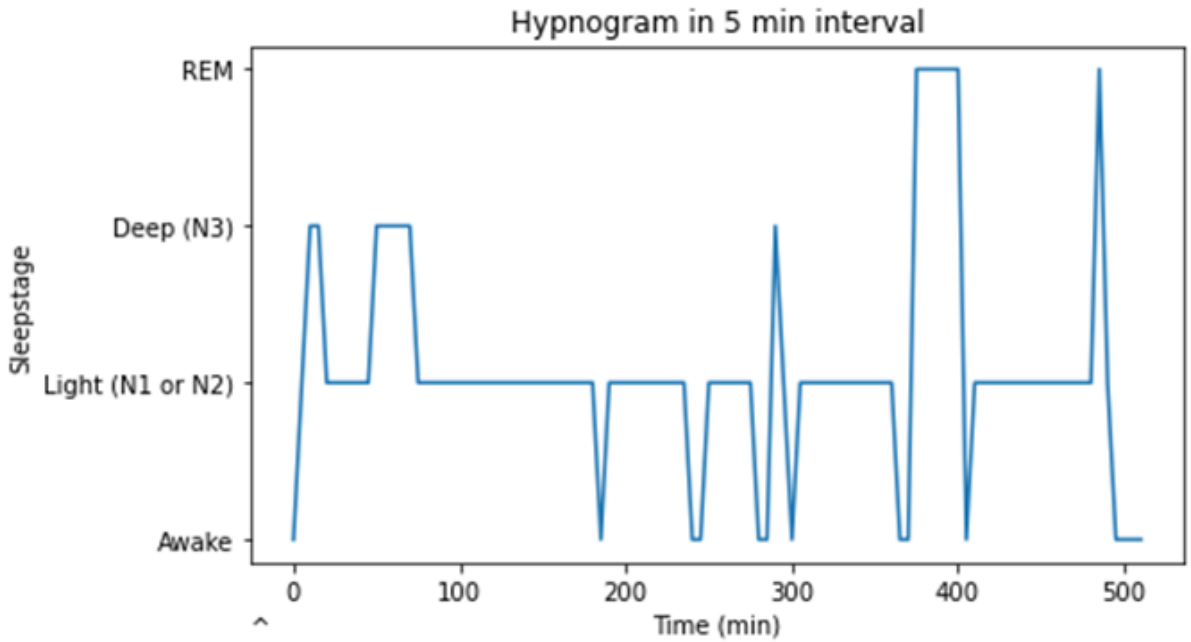
Bedtime start: 2021-05-26T23:45:39+02:00
Total bedtime: 472min
Onset latency: 6.5min



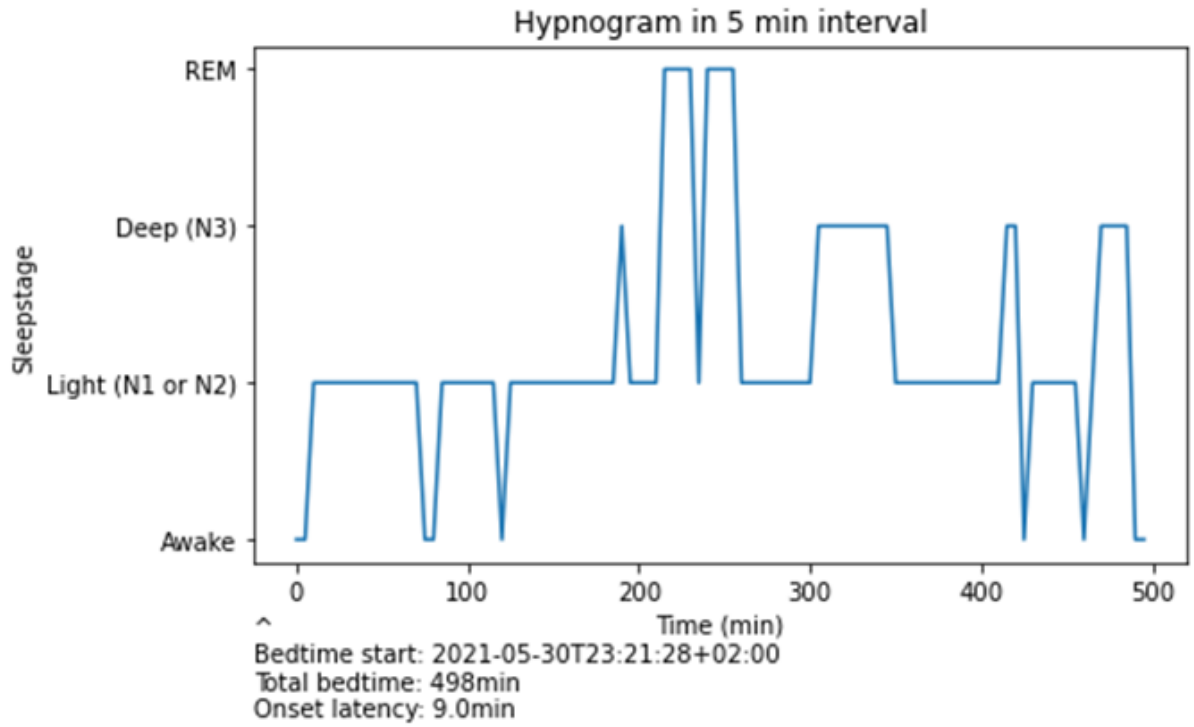
Bedtime start: 2021-05-28T00:14:32+02:00
Total bedtime: 446min
Onset latency: 4.5min



Bedtime start: 2021-05-28T23:48:01+02:00
Total bedtime: 487min
Onset latency: 15.0min



Bedtime start: 2021-05-29T23:55:30+02:00
Total bedtime: 512min
Onset latency: 5.0min



As can be seen in the Oura API documentation (Ōura Health Oy, n.d.), Oura defines a sleep onset latency (SOL) of less than 15 minutes to be good, 15 minutes as normal and more than 15 minutes problematic. Since we are using the Oura ring, this is the standard that I will be using as well.

For the last week in the dataset, this user only had good and normal SOL's. No nights contained any awakenings that lasted longer than 15 minutes. The third and sixth night the user stayed in bed for longer than 15 minutes after waking up at the end of the night.

Inspecting full day activity spectra

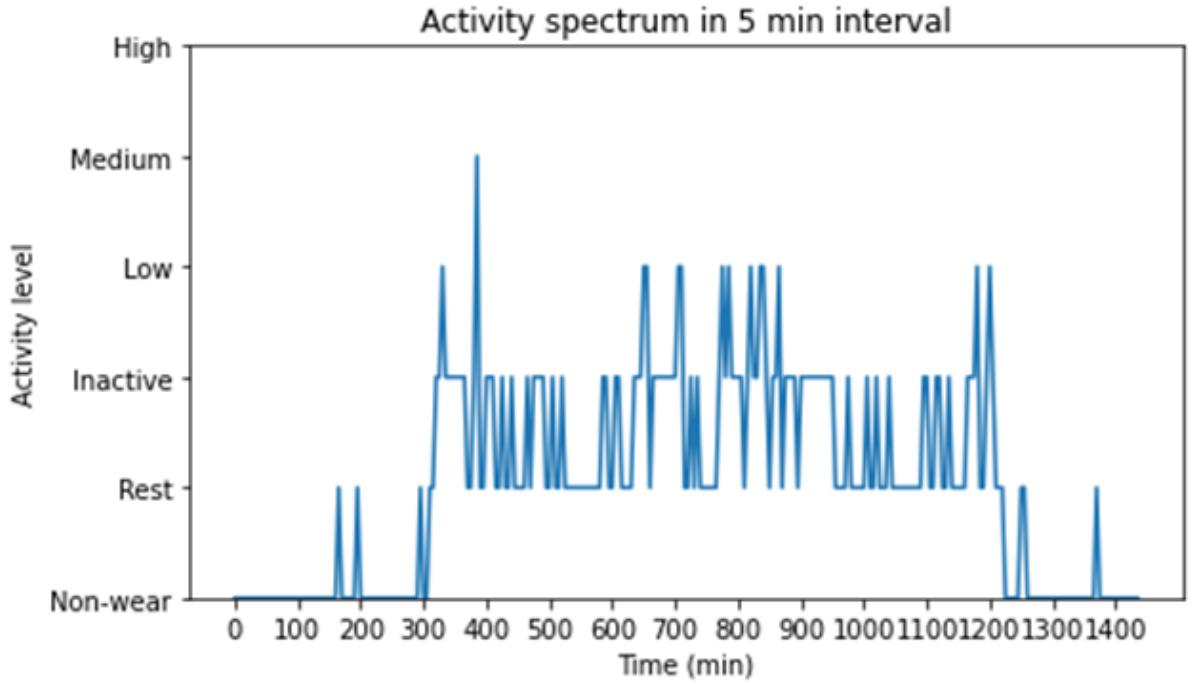
For the following activity spectra it is useful to understand the difference between the respective activity levels.\ According to the Oura API documentation (Ōura Health Oy, n.d.) the levels are defined as follows:

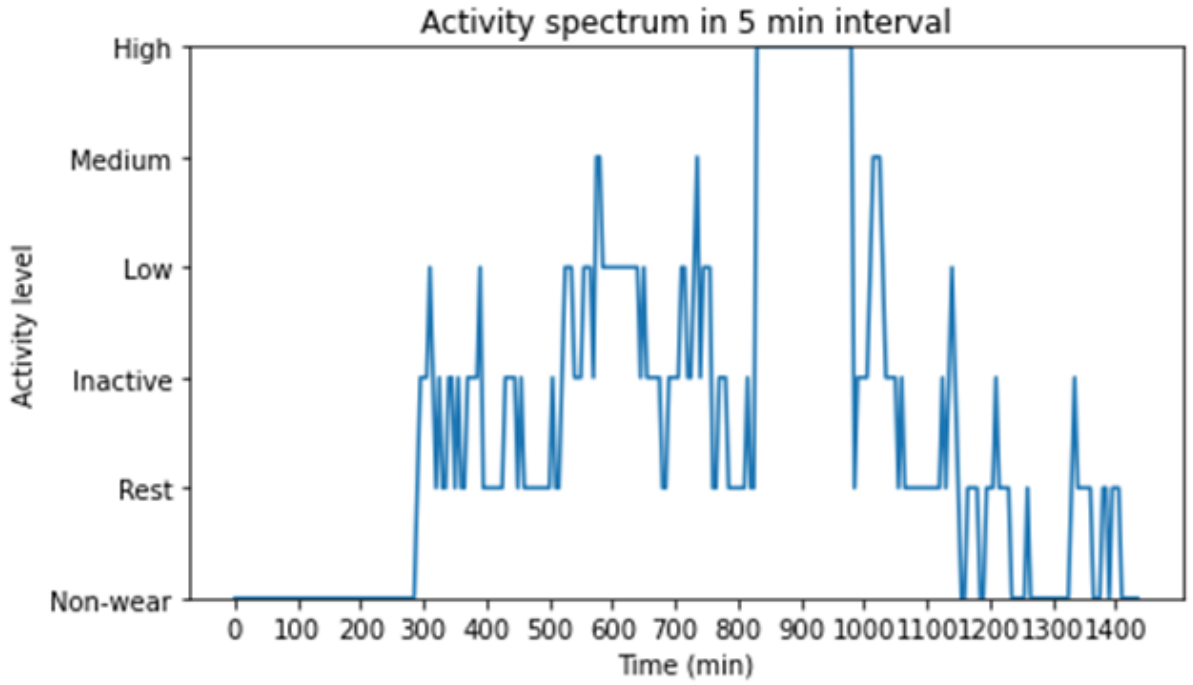
Metabolic-equivalent (MET) minutes express how much more energy the user is burning than if they were sitting still.

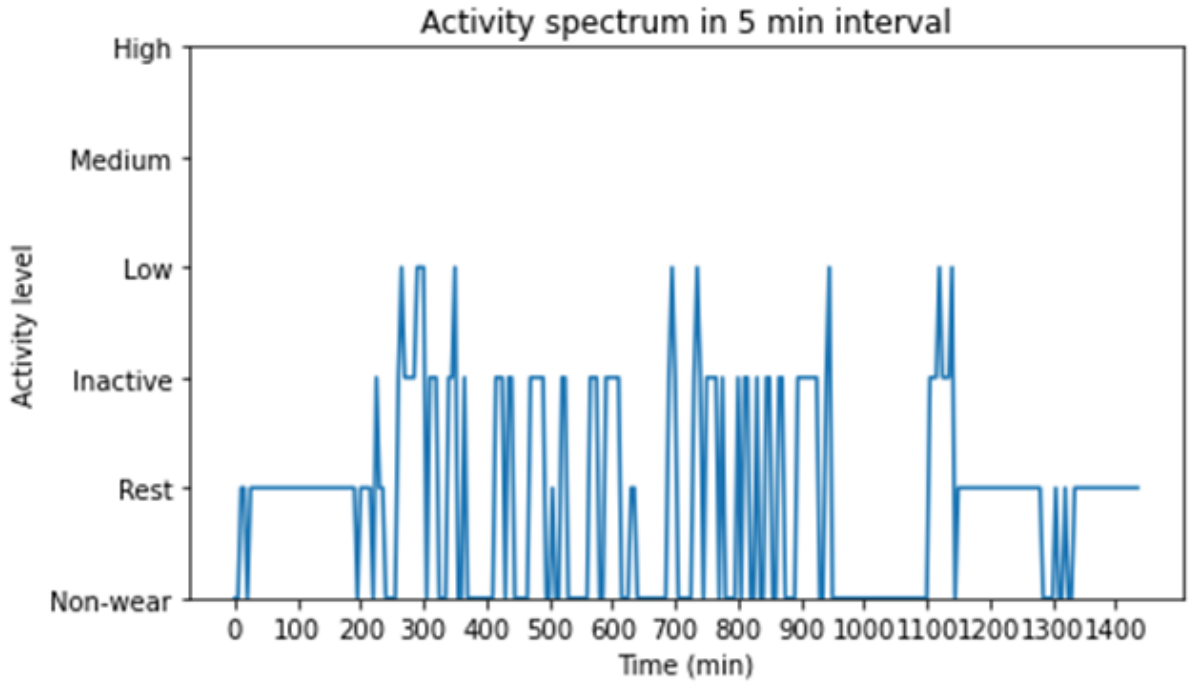
Non-wear: Not wearing the ring
Rest: Lying down or sleeping, corresponds to an average MET of <1.05.
Inactive: Sitting or standing still, corresponds to an average MET of 1.05 to 2.
Low: Low intensity activity, e.g. Household work, corresponds to an average MET of 2 to the age dependent limit.
Medium: Medium intensity activity, e.g. Walking, average MET level depends on age and gender.
High: High intensity activity, e.g. Running, average MET level depends on age and gender.

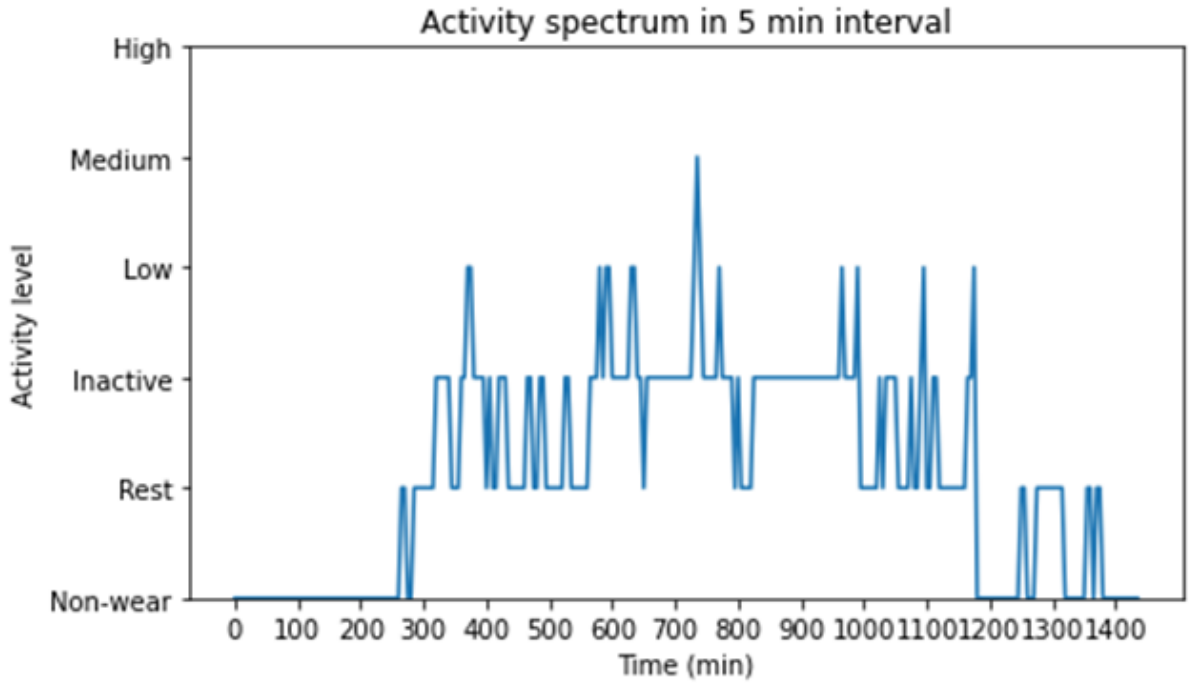
In [25]:

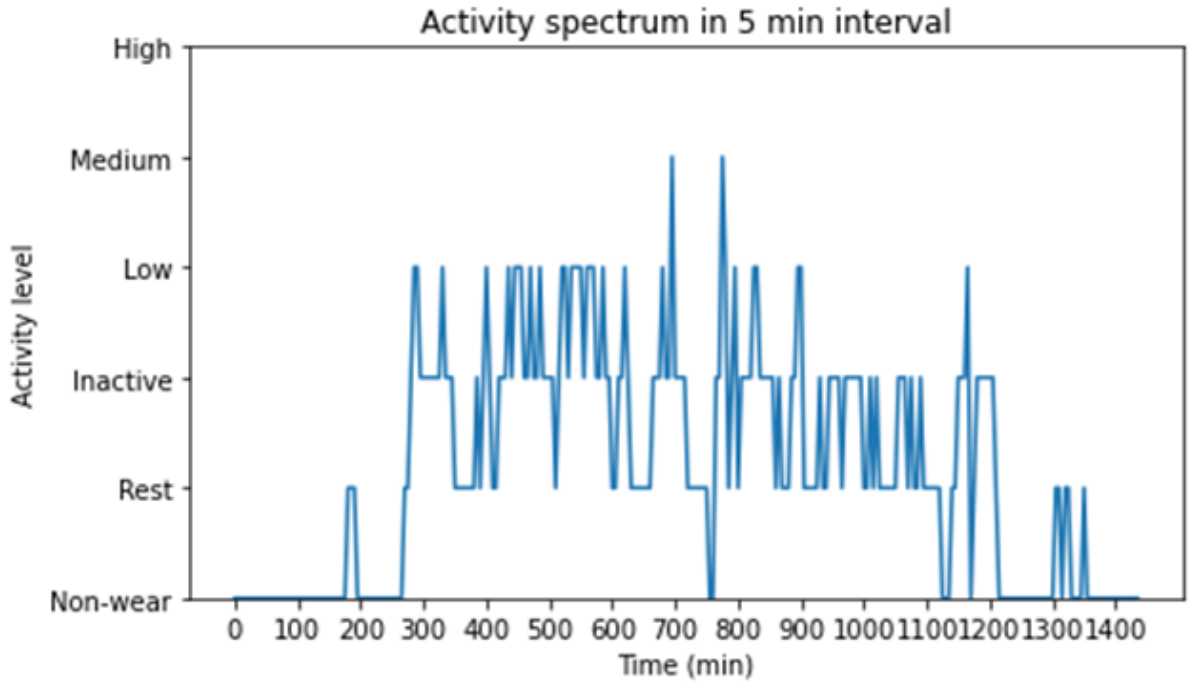
```
#Plot full activity spectra for some random datasamples, using a seed to ensure the samples are always the same
np.random.seed(52)
samples=np.random.randint(0, len(data['activity']), 10)
for i in samples:
        temps=[]
        for j in data['activity'][i]['class_5min']:
        temps.append(j)
        plot(temps, [], [], 'Activity spectrum in 5 min interval', [], [], [], False, 'Time (min)', [np.arange(0, 281, 20)],[np.arange(0, 1401, 100)], [], 'Activity level', [np.arange(0, 6)], [['Non-wear', 'Rest', 'Inactive', 'Low', 'Medium', 'High']], [0, 5], [], [])
```

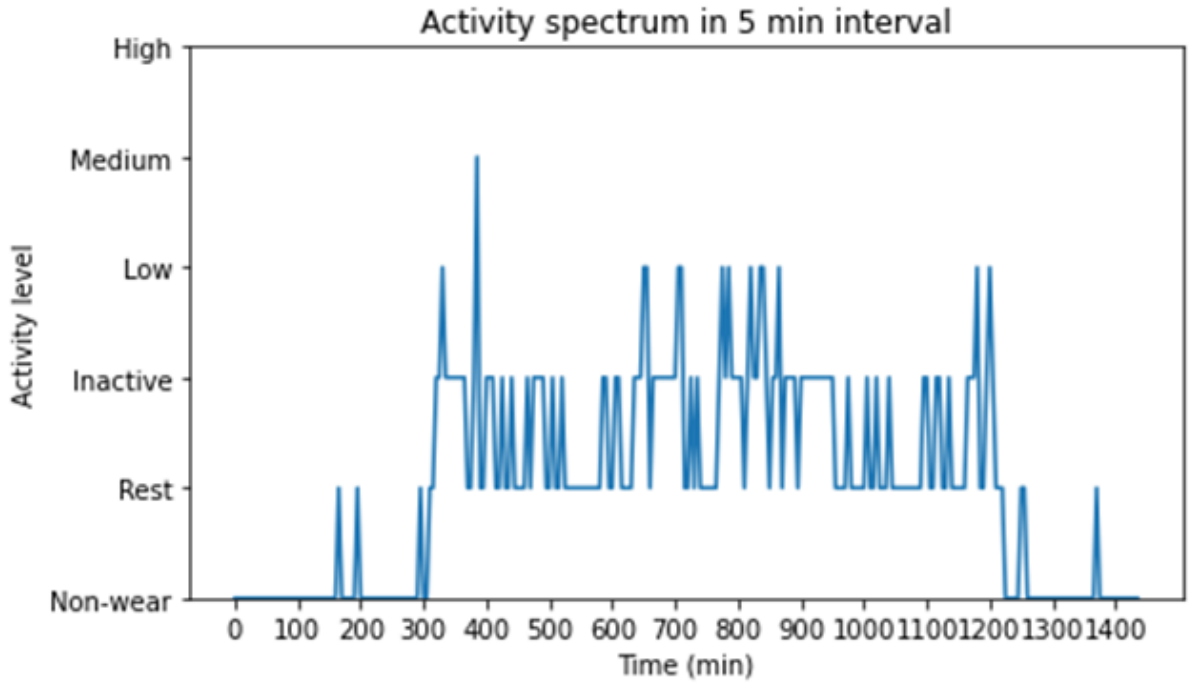


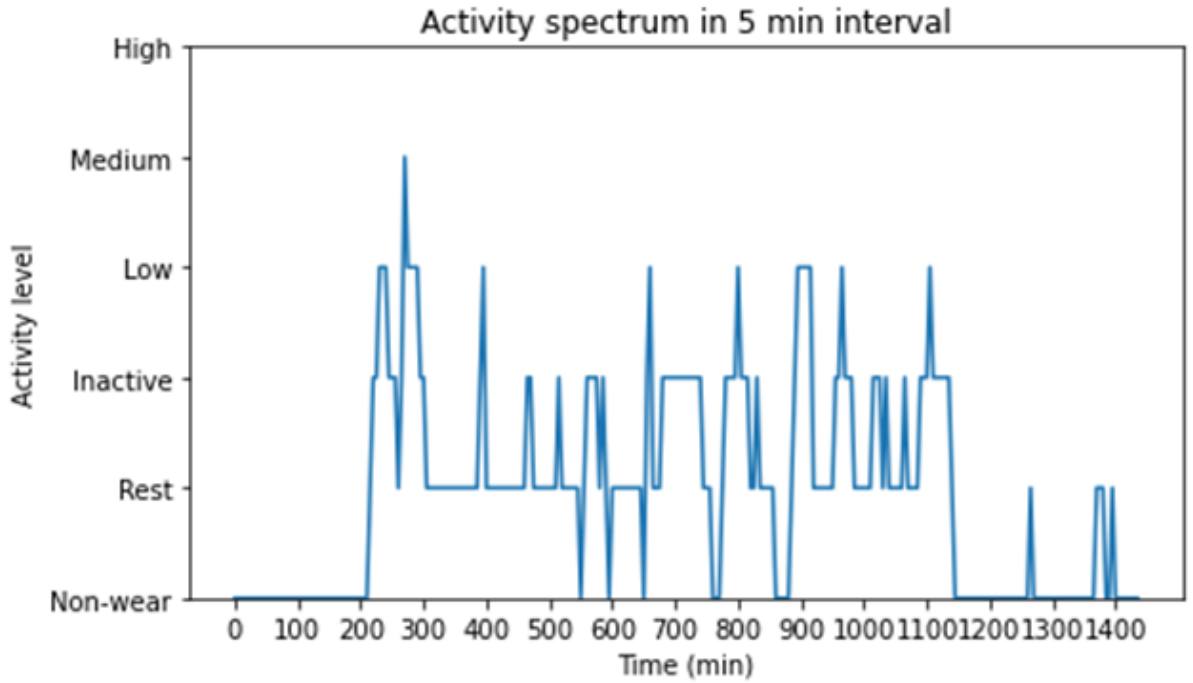


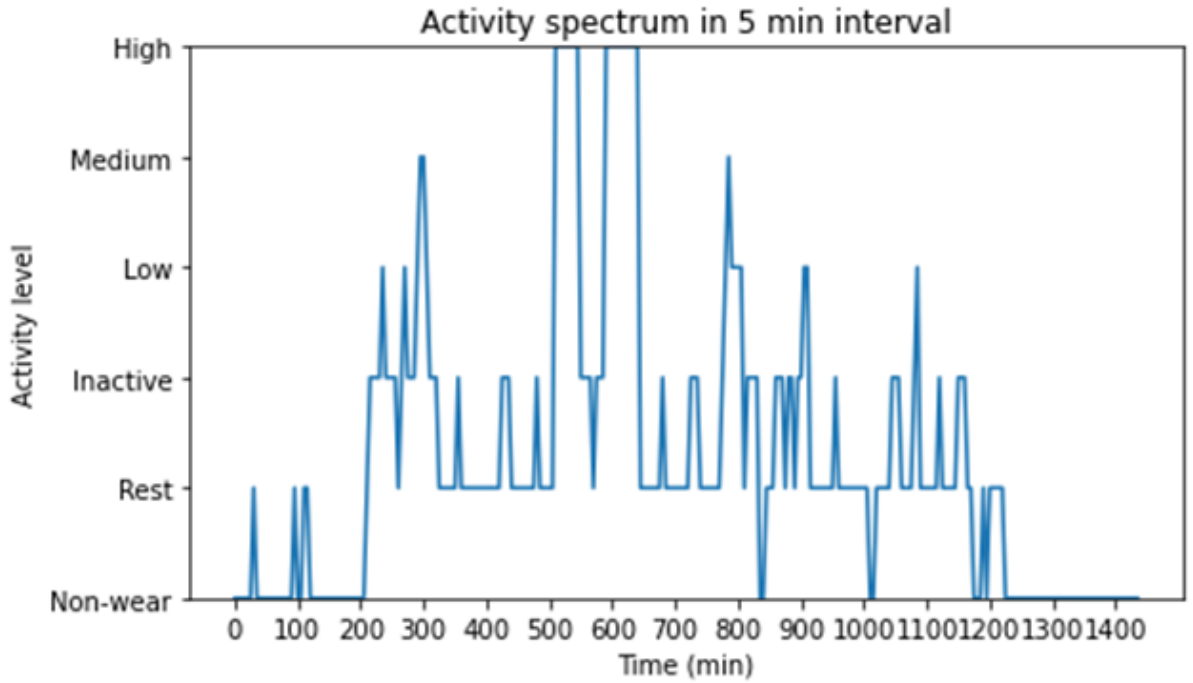

Activity spectrum in 5 min interval
High
Medium
Low
Inactive
Rest
Non-wear
Activity level
0 100 200 300 400 500 600 700 800 900 1000 1100 1200 1300 1400
Time (min)

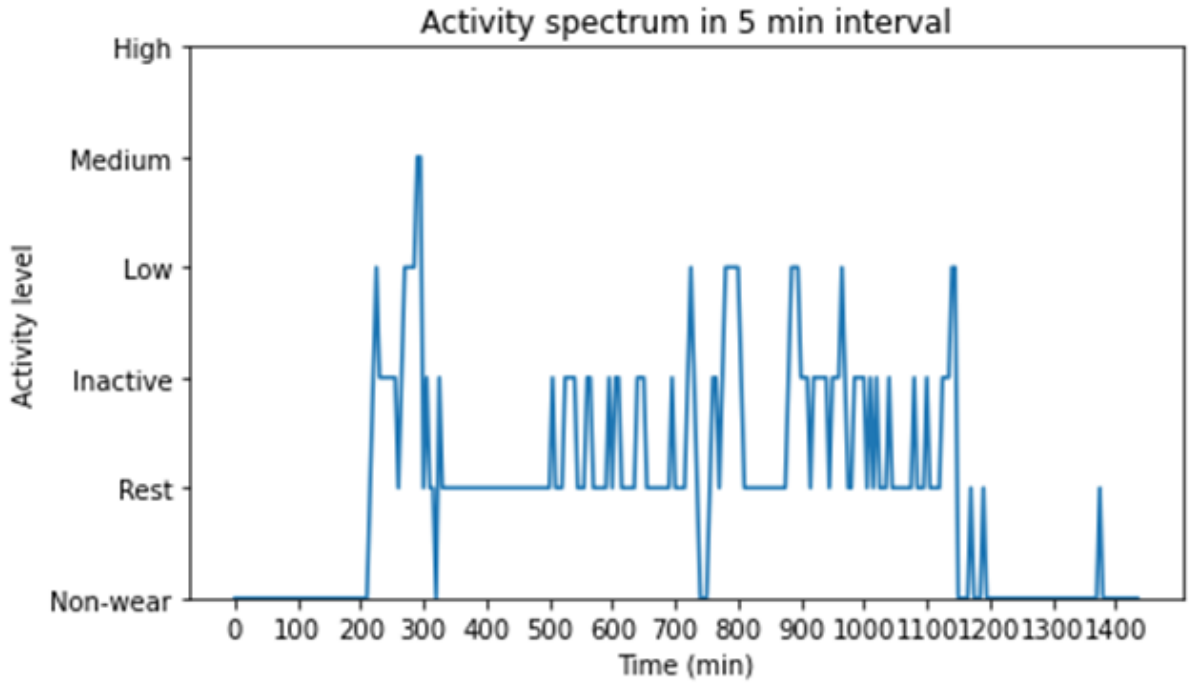

Activity spectrum in 5 min interval
High
Medium
Low
Inactive
Rest
Non-wear
Activity level
0 100 200 300 400 500 600 700 800 900 1000 1100 1200 1300 1400
Time (min)

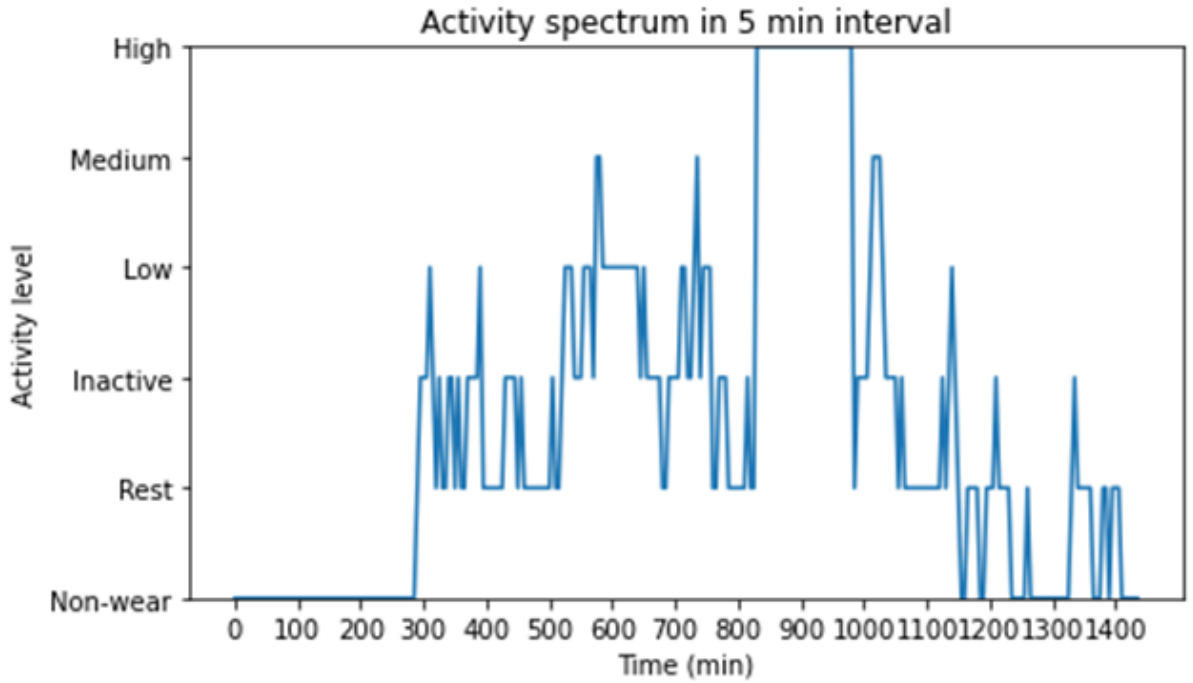

Activity spectrum in 5 min interval
High
Medium
Low
Inactive
Rest
Non-wear
Activity level
0 100 200 300 400 500 600 700 800 900 1000 1100 1200 1300 1400
Time (min)


Activity spectrum in 5 min interval
High
Medium
Low
Inactive
Rest
Non-wear
Activity level
0 100 200 300 400 500 600 700 800 900 1000 1100 1200 1300 1400
Time (min)


Activity spectrum in 5 min interval
High
Medium
Low
Inactive
Rest
Non-wear
Activity level
0 100 200 300 400 500 600 700 800 900 1000 1100 1200 1300 1400
Time (min)


Activity spectrum in 5 min interval
High
Medium
Low
Inactive
Rest
Non-wear
Activity level
0 100 200 300 400 500 600 700 800 900 1000 1100 1200 1300 1400
Time (min)





From this visualisation we can see that for this user the activity levels already reach the medium level and high level more often and also for prolonged periods of time, whereas this was not the case for the first subject.

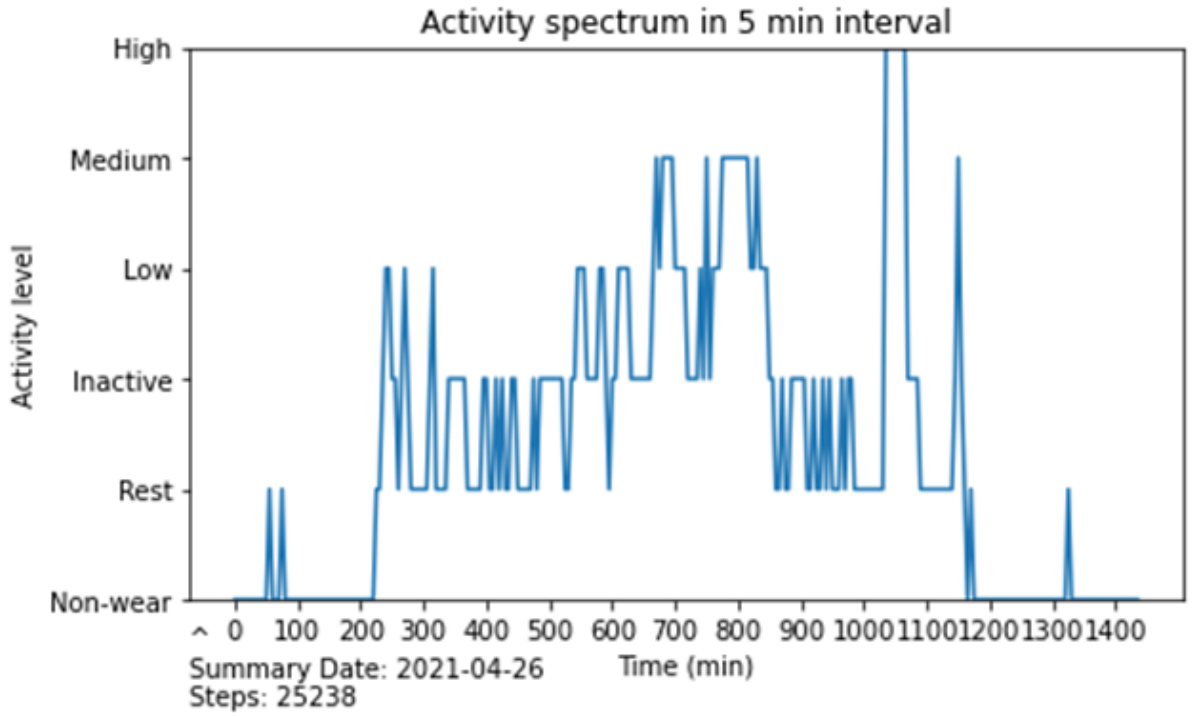
Let's also take a look at the steps like we did before:

In [26]:

```
steps=[]
for i in range(0, len(data['activity'])):
        steps.append(data['activity'][i]['steps'])
```

```
print('The most steps taken during a day is: {} steps.'.format(np.max(steps)))
temps=[]
for j in data['activity'][np.where(steps==np.max(steps))[0][0]]['class_5min']:
        temps.append(j)
plot(temps, [], [], 'Activity spectrum in 5 min interval', [], [], [], False, 'Time (min)',
[np.arange(0, 281, 20)], [np.arange(0, 1401, 100)], [], 'Activity level', [np.arange(0, 6)],
[['Non-wear', 'Rest', 'Inactive', 'Low', 'Medium', 'High']], [0, 5], [], [0.125, -0.02, '^\nSummary
Date: {}\nSteps:
{}'.format(data['activity'][np.where(steps==np.max(steps))[0][0]]['summary_date'],
data['activity'][np.where(steps==np.max(steps))[0][0]]['steps'])])
```

```
The most steps taken during a day is: 25238 steps.
```



Here we can see that even on the day where the user walked the most steps, which is really a lot of steps, the activity hardly ever gets to the Medium level. Therefore it is safe to assume that the attribution of Medium activity level to walking is unrealistic for this user too, making it more likely that most of the walking activity is classified as Low level activity, despite the fact that the Oura team classified it as chores. In order to be sure we will also check this for the last user.

Aligning the activity spectra to the hypnograms

In [27]:

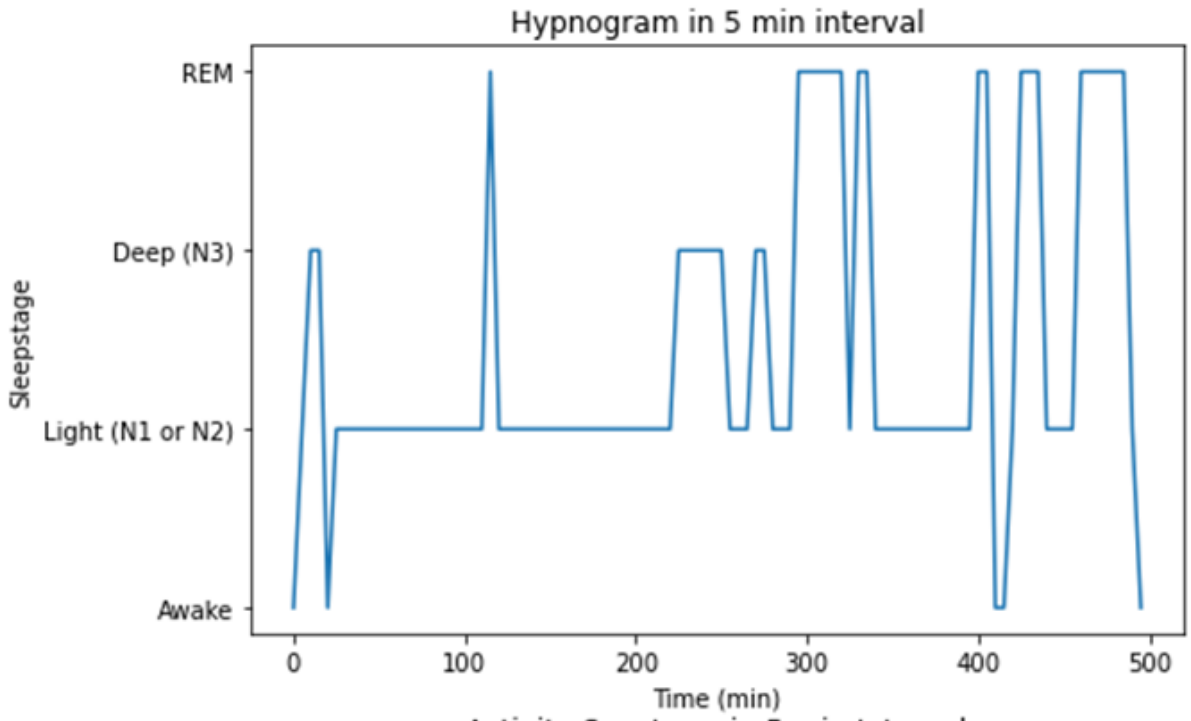
```
#Aligning the activity spectra for the nights during which the user was awake for more than
15 minutes in a row.
#Creating an empty container to store the combined activity class strings for each day
activity_strings=[]
for i in range(0,len(sleeping_times_m)):
        if i>=(len(data['sleep'])-8) and i<(len(data['sleep'])-1-2):
```

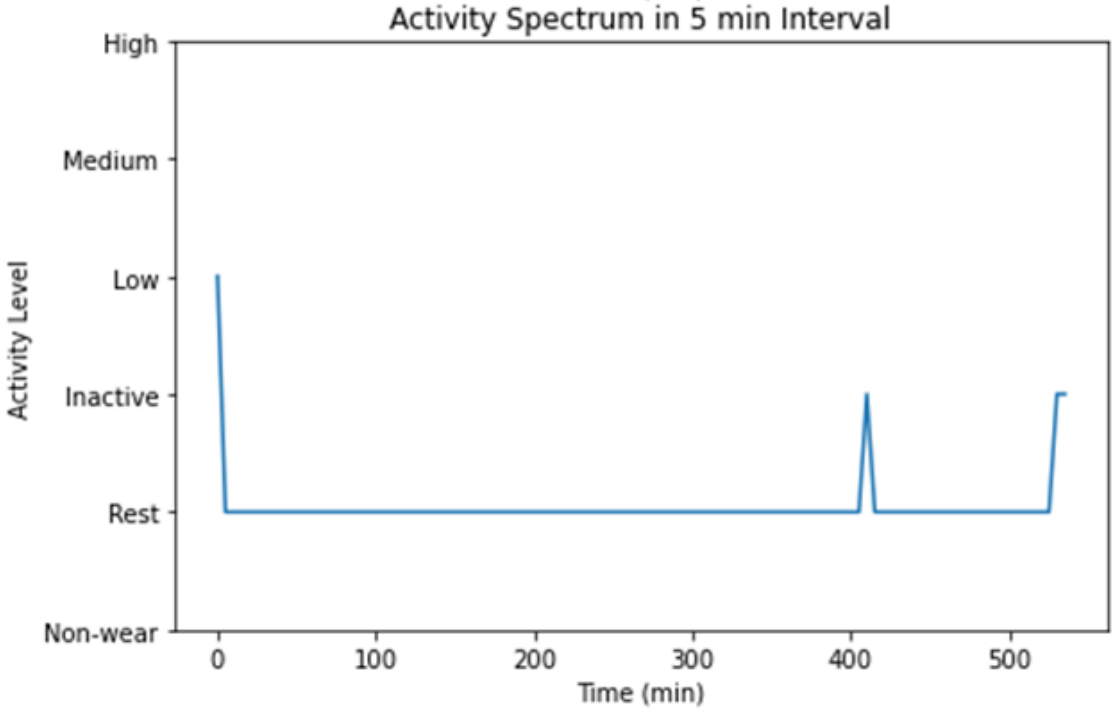
```
        #Creating an empty container to store the combined end and beginning of the activity
class strings
        int_ind=""

      #Calculate how much minutes from bedtime start to 4 am
        min_till_4am=((24*60)-sleeping_times_m[i]+(4*60))%1440
      #Divide in 5 minute intervals starting from the back
        last_intval_ind=int(np.floor(min_till_4am/5))

      #Matching the date from the sleep data to the date from the activity data
        match_ind=0
        for j in range(0, len(data['activity'])):
        if data['activity'][j]['summary_date']==data['sleep'][i]['summary_date']:
                match_ind=j

      #Check if one entry earlier is indeed one day earlier
        if(date_difference(date_conversion(data['activity'][match_ind-1]['summary_date']),
date_conversion(data['activity'][match_ind]['summary_date']))):

        #Add that part of the interval to the string
        int_ind+=(data['activity'][match_ind-1]['class_5min'][-last_intval_ind-1:-1])

        #Calculate how much minutes from 4 am to bedtime end
        min_from_4am=rising_times_m[i]-(4*60)
        #Divide in 5 minute intervals starting from the back
        first_intval_ind=int(np.floor(min_from_4am/5))

        #Add that part of the interval to the string
        int_ind+=(data['activity'][match_ind]['class_5min'][0:first_intval_ind])

        #Store the string in the container
        activity_strings.append(int_ind)

      else:
        print('Missing date in the activity data before the night to examine.')
```

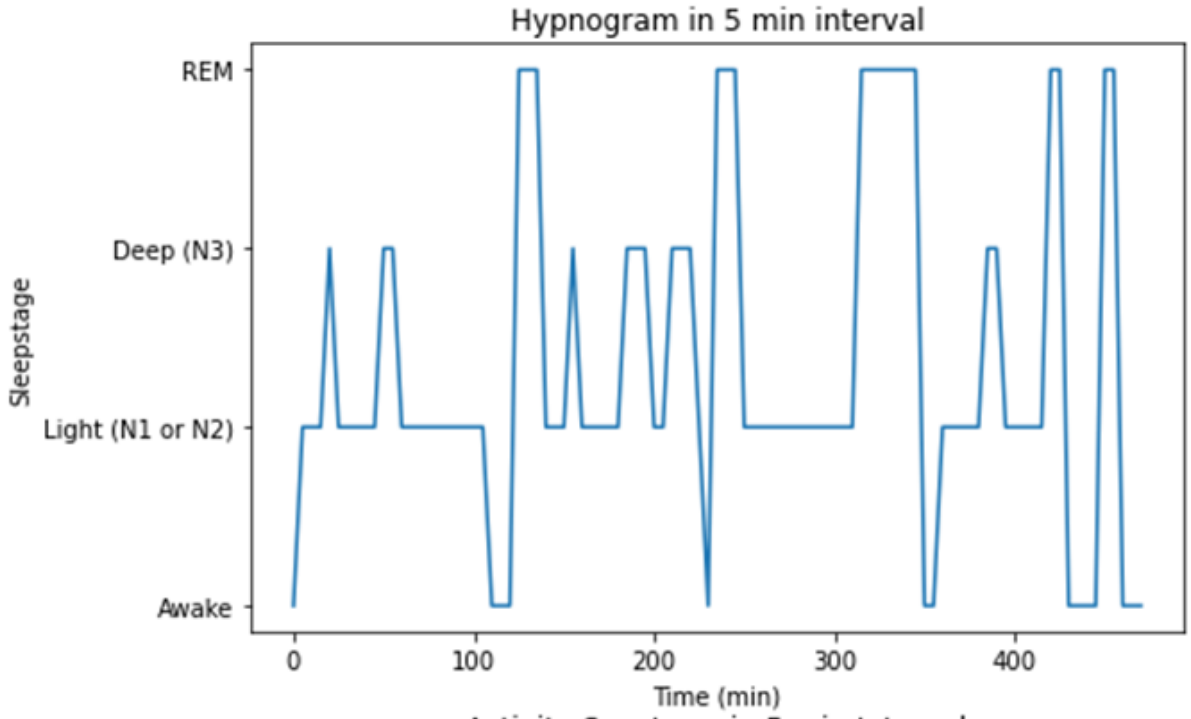
In [28]:

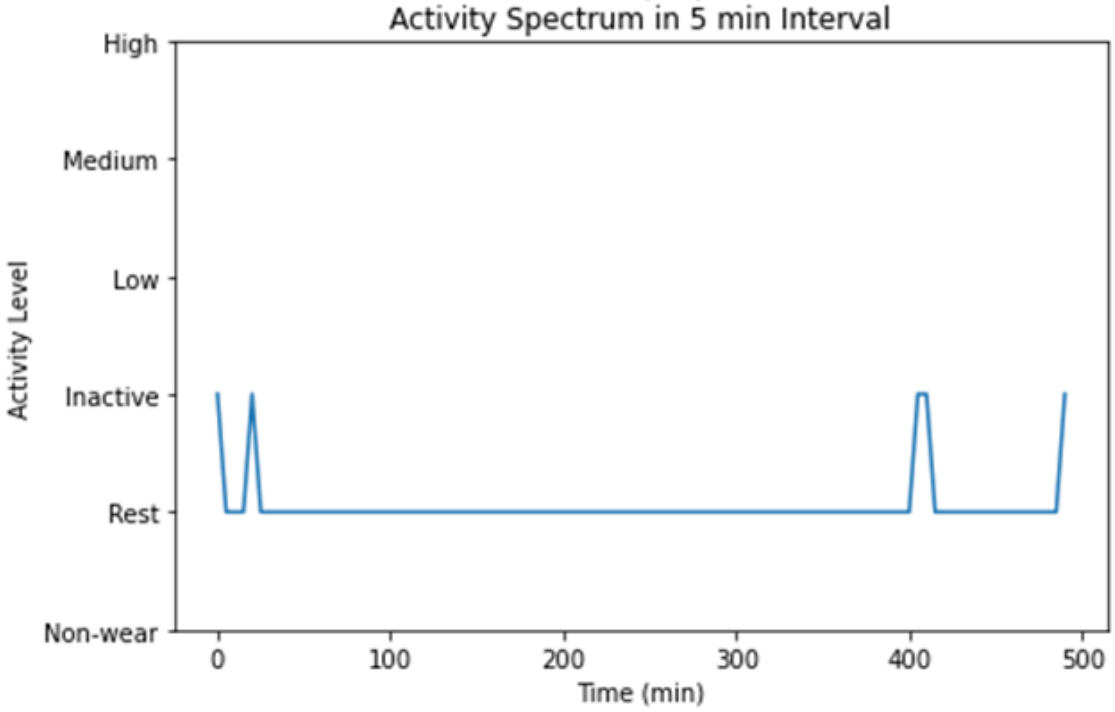
```
#Lets plot the activity spectra for the nights during which the user was awake for more than
15 minutes in a row.
#Since I want to plot the hypnograms and the activity spectra right above each other for easy
comparison I will not use the plotting function created earlier, as it does not allow such
operations

counter=-1

#Converting the activity string to a list of integers
for i in activity_strings:
```

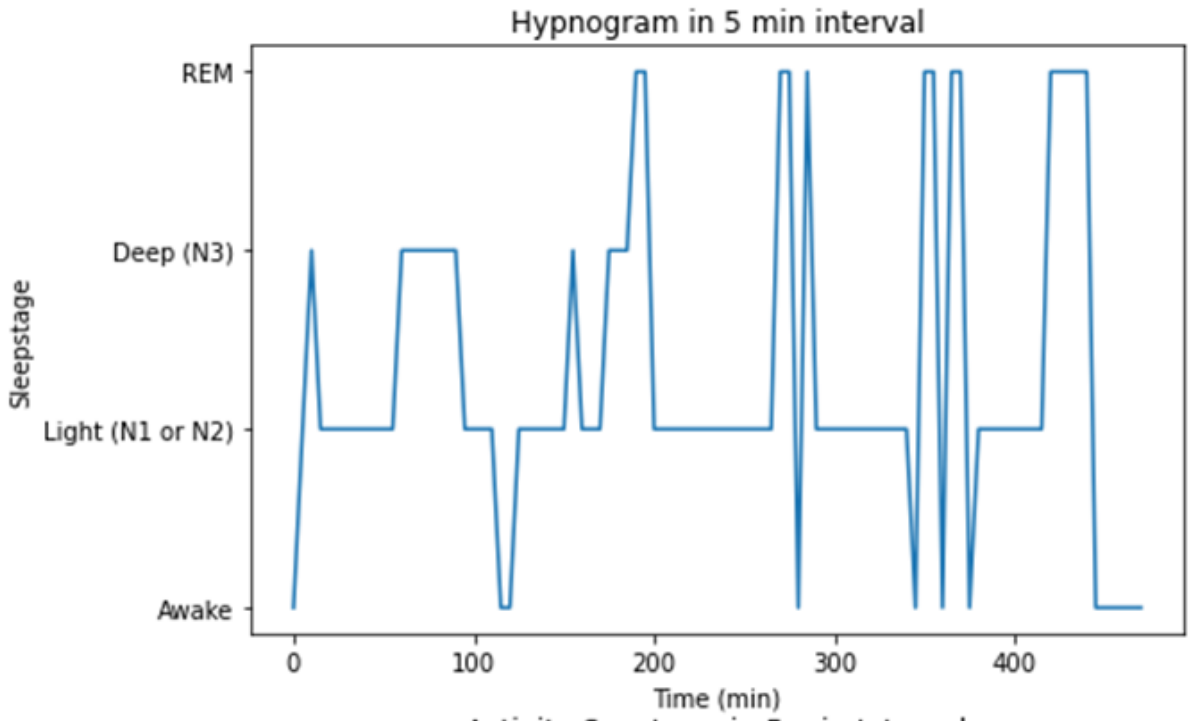
```
        counter+=1
        as5_split=[]
        for j in range(0,len(i)):
        as5_split.append(int(i[j]))

    #Plotting the hypnograms for easy comparison
        fig, (ax1, ax2)=plt.subplots(2, 1, figsize=(7,10))
        ax1.plot(hgs5[counter])
        ax1.set_title('Hypnogram in 5 min interval')
        ax1.set_ylabel('Sleepstage')
        ax1.yaxis.set_major_locator(matplotlib.ticker.FixedLocator([0, 1, 2, 3]))
        ax1.set_yticklabels(['Awake', 'Light (N1 or N2)', 'Deep (N3)', 'REM'])
        ax1.set_xlabel('Time (min)\n')
        ax1.xaxis.set_major_locator(matplotlib.ticker.FixedLocator([0, 20, 40, 60, 80, 100,
120, 140, 160]))
        ax1.set_xticklabels(['0', '100', '200', '300', '400', '500', '600', '700', '800'])

        #Plotting the activity spectra
        ax2.plot(as5_split)
        ax2.set_title('Activity Spectrum in 5 min Interval')
        ax2.set_ylabel('Activity Level')
        ax2.yaxis.set_major_locator(matplotlib.ticker.FixedLocator([0, 1, 2, 3, 4, 5]))
        ax2.set_yticklabels(['Non-wear', 'Rest', 'Inactive', 'Low', 'Medium', 'High'])
        ax2.set_ylim([0, 5])
        ax2.set_xlabel('Time (min)')
        ax2.xaxis.set_major_locator(matplotlib.ticker.FixedLocator([0, 20, 40, 60, 80, 100,
120, 140, 160]))
        ax2.set_xticklabels(['0', '100', '200', '300', '400', '500', '600', '700', '800'])
        plt.show()
```

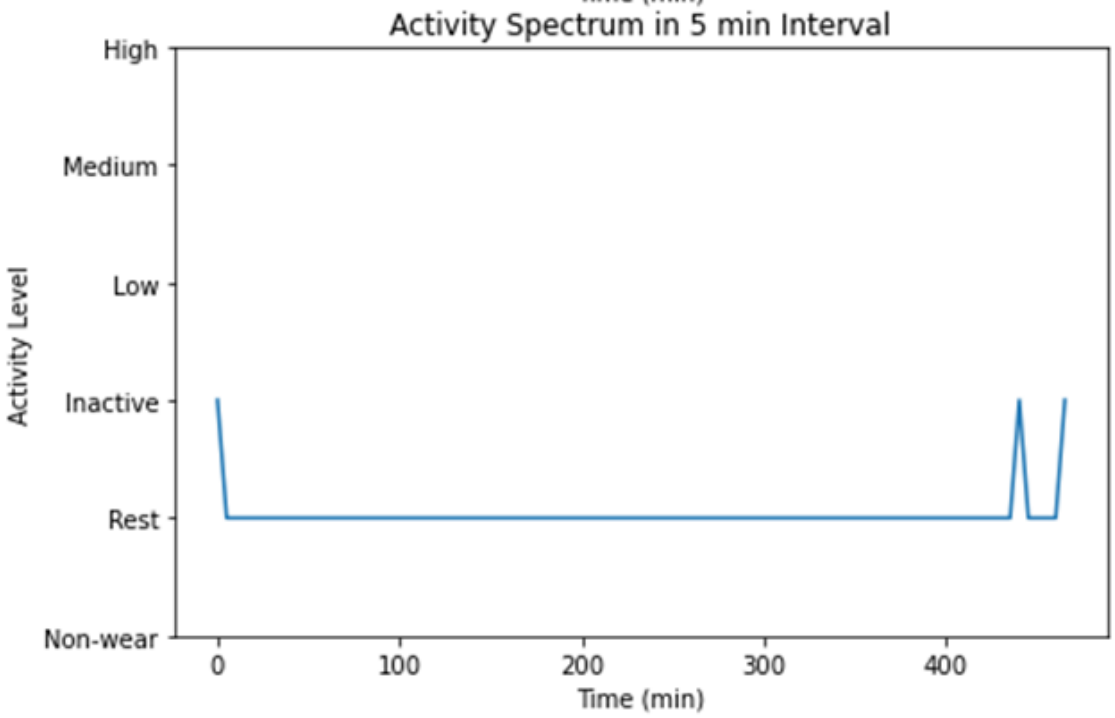

Hypnogram in 5 min interval
REM
Deep (N3)
Light (N1 or N2)
Awake
Sleepstage
0
100
200
300
400
500
Time (min)

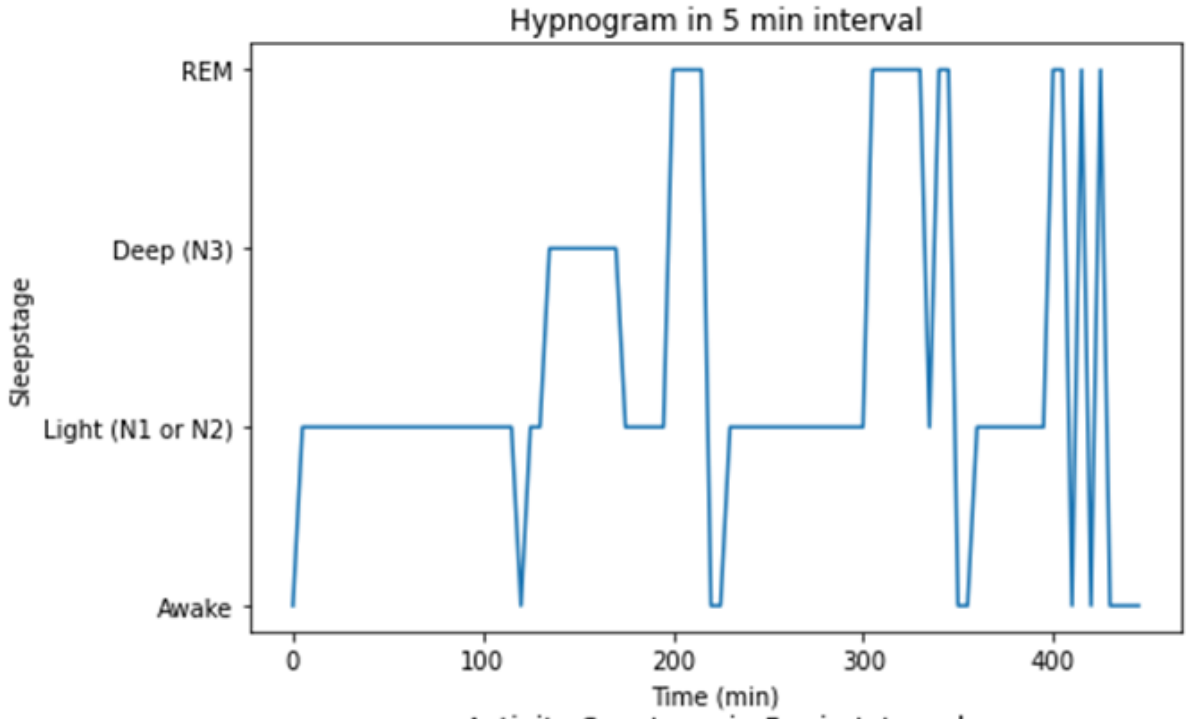

Activity Spectrum in 5 min Interval
High
Medium
Low
Inactive
Rest
Non-wear
Activity Level
0
100
200
300
400
500
Time (min)

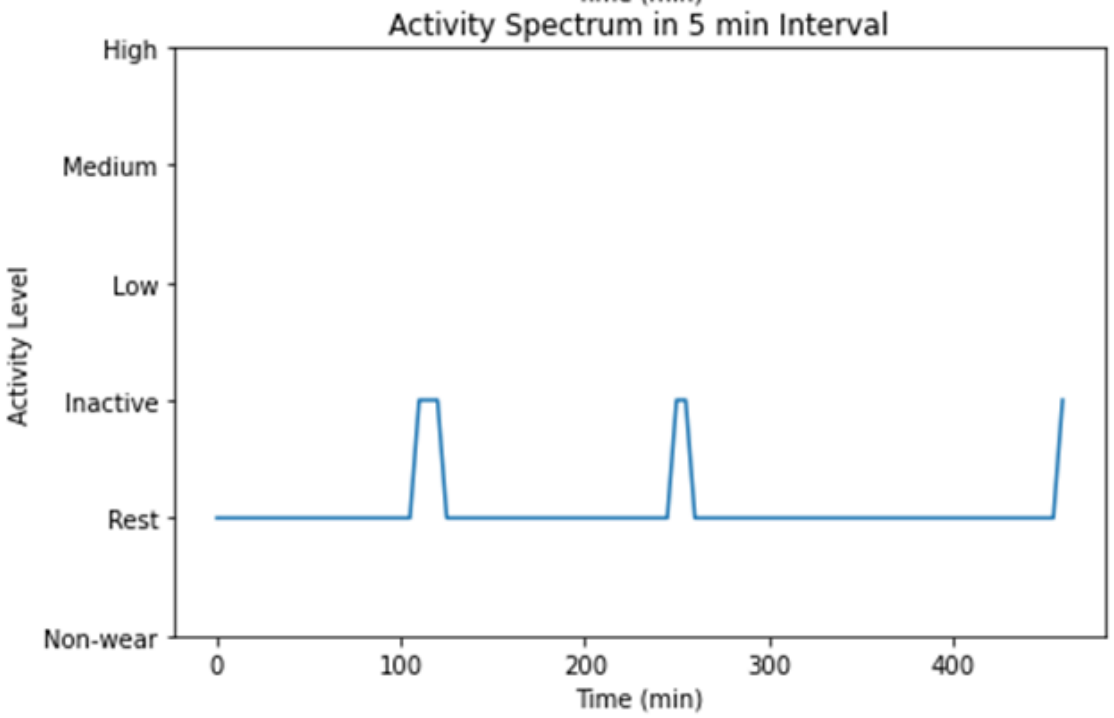

Hypnogram in 5 min interval
REM
Deep (N3)
Light (N1 or N2)
Awake
Sleepstage
0
100
200
300
400
Time (min)

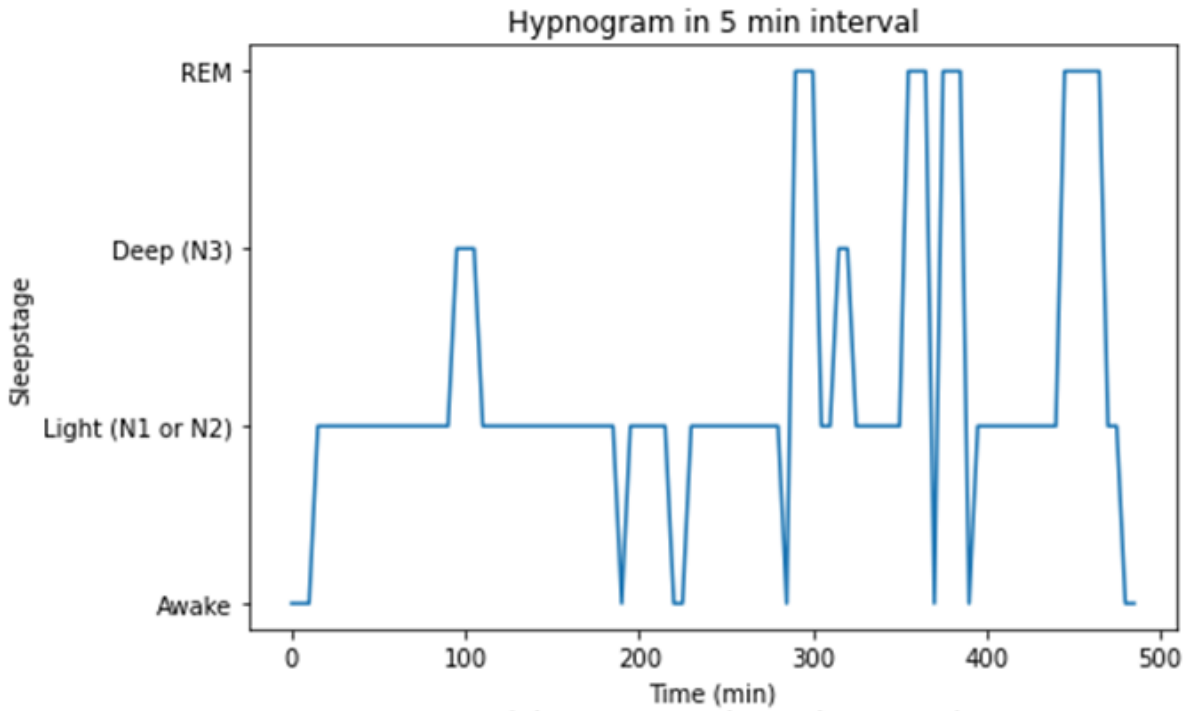

Activity Spectrum in 5 min Interval
High
Medium
Low
Inactive
Rest
Non-wear
Activity Level
0
100
200
300
400
500
Time (min)

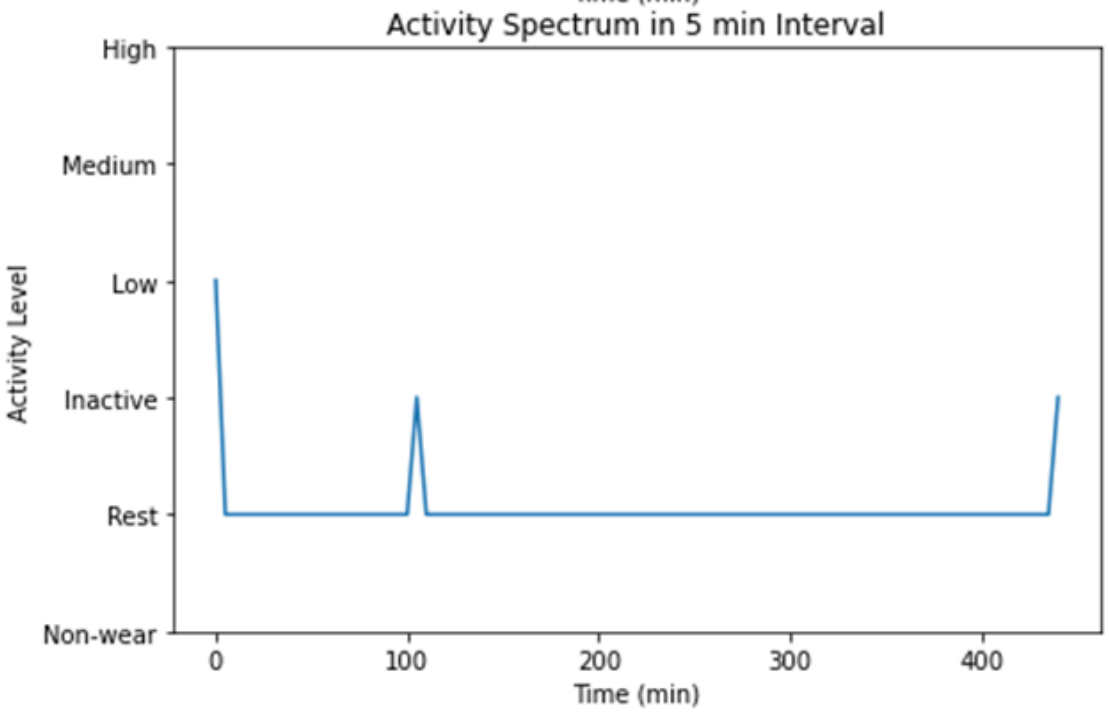

Hypnogram in 5 min interval
REM
Deep (N3)
Light (N1 or N2)
Awake
Sleepstage
0
100
200
300
400
Time (min)


Activity Spectrum in 5 min Interval
High
Medium
Low
Inactive
Rest
Non-wear
Activity Level
0
100
200
300
400
Time (min)


Hypnogram in 5 min interval
REM
Deep (N3)
Light (N1 or N2)
Awake
Sleepstage
0
100
200
300
400
Time (min)


Activity Spectrum in 5 min Interval
High
Medium
Low
Inactive
Rest
Non-wear
Activity Level
0
100
200
300
400
Time (min)





As can be seen from the hypnograms, sometimes when the user awakens during the night the activity goes or has gone from rest to inactive, but even the low level is never reached. Hence I can safely conclude that the user did not go out of bed during the night, which is good because they did not have to, since they were not awake for a prolonged period of time during the night.

Determining the advice

Based on the fortified assumption that the low activity level corresponds to walking activity indeed, this is how we can determine the advice:

In [29]:

```
def get_out_of_bed():
        #This function is designed to return the number of times the user stayed in bed whils
being awake for a prolonged period of time during the past two weeks.
        try:
        #Creating a counter for the number of times the user stayed in bed whilst being
awake for a prolonged period of time
    times=0
        #Creating an index counter
        ind=-1
        #Converting the sleeping times to minutes in case that has not been done yet
        sleeping_times_m=time_min_conversion('bedtime_start')

     #Checking the hypnograms of the last two weeks
        for i in data['sleep'][-15:-1]:
        #Keeping track of the respective index of the sleep data
        ind+=1
        #Checking for how many consecutive times the users sleep state is awake
        counter=0
        #Checking every value of the hypnogram
        for j in range(0, len(i['hypnogram_5min'])):
                #Incrementing the counter if the sleeping state is 'Awake'
                if int(i['hypnogram_5min'][j])==4:
                counter+=1
                #Resetting the counter if the consecutive streak is broken
                else:
                counter=0
                #If the user was awake for more than 15 consecutive minutes check activity
levels
                if counter>3:
                #Creating a container for the activity string
                activity_string=""

              #Matching the date from the sleep data to the date from the activity data
                match_ind=0

              for k in range(0, len(data['activity'])):
                if data['activity'][k]['summary_date']==data['sleep'][-15+ind]['summary_date']:
                        match_ind=k

                #Check if one entry earlier is indeed one day earlier

if(date_difference(date_conversion(data['activity'][match_ind-1]['summary_date']),
date_conversion(data['activity'][match_ind]['summary_date']))):
                int_ind=""
                #Calculate how much minutes from bedtime start to 4 am
                min_till_4am=((24*60)-sleeping_times_m[-15+ind]+(4*60))%1440
                #Divide in 5 minute intervals starting from the back
                last_intval_ind=int(np.floor(min_till_4am/5))
```

```
                #Add that part of the interval to the string
                int_ind+=(data['activity'][match_ind-1]['class_5min'][-last_intval_ind-1:-1])

                #Calculate how much minutes from 4 am to bedtime end
                min_from_4am=rising_times_m[-15+ind]-(4*60)
                #Divide in 5 minute intervals starting from the back
                first_intval_ind=int(np.floor(min_from_4am/5))

                #Add that part of the interval to the string
                int_ind+=(data['activity'][match_ind]['class_5min'][0:first_intval_ind])

                #Store the string in the container
                activity_string+=int_ind
                else:
                print('Missing date in the activity data before the night to examine.\nThis
means that the activity spectrum cannot be matched to the hypnogram.')

            #Check each value of the activity string within the range of the prolonged
awakeness
             for l in range((j+1)-4,(j+1)):
                #Since the activity string can be slightly longer due to rounding errors, we
need to check if the index does not exceed the length of the activity string
                if not l>=len(activity_string):
                        #If a value in the activity string within the range of prolonged
awakeness reaches the low activity level or higher return true
                        if int(activity_string[l])>=3:
                        times+=1
        return times
        except:
        print('An error occurred.')
```

In [30]:

```
def get_out_of_bed_print():
        try:
        t=get_out_of_bed()
        if not t==0:
        return print('It looks like you stayed in bed whilst being awake for a prolonged period
of time {} times within the last two weeks.\nThis is considered to be bad sleep hygiene, so
next time you lie awake for a prolonged period of time, get out of bed until you are tired
again.'.format(t))
        else:
        return print('It looks like you have not stayed in bed whilst being awake for a
prolonged period of time within the last two weeks.\nThis is considered to be good sleep
hygiene, keep it up!')
        except:
    print('An error occurred.')
```

When should this advice be given?

In [31]:

```
def provide_get_out_of_bed():
	#This function is designed to determine whether the user should be adviced to get out of bed when unable to sleep.
	#This includes not being able to fall asleep and not being able to continue to sleep, which includes the final awakening.
	try:
	t=get_out_of_bed()
	if not t==0:
	return True
	else:
	return False
	except:
	print('An error occurred.')
```

Exercise regularly

In [32]:

```
def exercise_regularly():
	#This function is designed to determine the number of days the user exercised enough, where exercise is classified as either walking more than 7500 steps a day or being active on the medium level for more than 100 minutes on a certain amount of days per week.
	try:
	#Determining on how many days the user walked 7500 steps or more during the last week
	more_than_7500_steps=0
	for i in range(-8, -1):
	if data['activity'][i]['steps']>=7500:
		more_than_7500_steps+=1

	#Determining on how many days the user was active on the medium level for more than 100 minutes, or walked 7500 steps or more during the last week
	if (data['activity'][-1]['score_training_frequency']==100 and more_than_7500_steps==0) or (data['activity'][-1]['score_training_frequency']>=95 and data['activity'][-1]['score_training_frequency']<100 and more_than_7500_steps>0) or (data['activity'][-1]['score_training_frequency']>=70 and data['activity'][-1]['score_training_frequency']<95 and more_than_7500_steps>1) or (data['activity'][-1]['score_training_frequency']>=40 and data['activity'][-1]['score_training_frequency']<70 and more_than_7500_steps>2) or (more_than_7500_steps>3):
	return 4
	elif (data['activity'][-1]['score_training_frequency']>=95 and data['activity'][-1]['score_training_frequency']<100 and more_than_7500_steps==0) or (data['activity'][-1]['score_training_frequency']>=70 and
```

```
data['activity'][-1]['score_training_frequency']<95 and more_than_7500_steps==1) or
(data['activity'][-1]['score_training_frequency']>=40 and
data['activity'][-1]['score_training_frequency']<70 and more_than_7500_steps==2) or
(more_than_7500_steps==3):
        return 3
        elif (data['activity'][-1]['score_training_frequency']>=70 and
data['activity'][-1]['score_training_frequency']<95 and more_than_7500_steps==0) or
(data['activity'][-1]['score_training_frequency']>=40 and
data['activity'][-1]['score_training_frequency']<70 and more_than_7500_steps==1) or
(more_than_7500_steps==2):
            return 2
        elif (data['activity'][-1]['score_training_frequency']>=40 and
data['activity'][-1]['score_training_frequency']<70 and more_than_7500_steps==0) or
(more_than_7500_steps==1):
        return 1
        else:
        return 0
        except:
        print('An error occurred.')
```

In [33]:

```
def exercise_regularly_print():
        #This function is designed to print the advice for exercising regularly
        try:
        if exercise_regularly()==4:
        return print('It looks like you have been exercising enough lately. This is healthy
behavior that is good for your sleep quality. Keep it up!')
        elif exercise_regularly()==3:
        return print('It looks like you have been exercising enough lately, but there is room for
improvement. If you feel like your sleep quality has suffered, it might be better to exercise
more (regularly) or walk more than 7500 steps a day (more regularly). Keep it up!')
        else:
        return print('It looks like you have not been exercising enough lately. If you feel like
your sleep quality has suffered, it might be a good idea to start exercising more (regularly) or
start walking more than 7500 steps a day (more regularly).')
        except:
        print('An error occurred.')
```

In [34]:

```
def provide_exercise_regularly():
        #This function is designed to determine whether the user should be advised to
exercise more regularly or not.
        try:
      if exercise_regularly()<4:
        return True
        else:
```

```
        return False
        except:
        print('An error occurred.')
```

Reduce time in bed to time slept

In [35]:

```
def sleep_restriction_bedtime():
        #This function is designed to determine the sleep restriction bedtime.
        try:
        #Creating a container for the total sleep duration per night of the last two weeks
        total=[]

    #Creating a list of all total sleep durations of the last two weeks
        for i in range(len(data['sleep'])-15, len(data['sleep'])-1):
        total.append(data['sleep'][i]['total'])

    #Taking the average total sleep duration of the last two weeks
        avg_total=int(np.mean(np.divide(total, 60)))

    #Adding half an hour to the average total sleep duration of the past two weeks
        inc_avg_total=avg_total+30

    #Ensuring the increased average total sleep duration of the past two weeks is 5.5 hours
at minimum
        if inc_avg_total<(5.5*60):
        inc_avg_total=(5.5*60)

    #Calculating the ideal bedtime based on the increased average total sleep duration
        sleep_restriction_bedtime=wake_time()-inc_avg_total

    return sleep_restriction_bedtime
        except:
        print('An error occurred.')
```

In [36]:

```
def sleep_restriction_bedtime_print():
        #This function is designed to print the advice for the sleep restriction bedtime.
        try:
        return print('The ideal sleeping time for this user when applying sleep restriction
therapy is at {}.'.format(time.strftime("%H:%M",
time.gmtime(sleep_restriction_bedtime()*60))))
        except:
        print('An error occurred.')
```

In [37]:

```
def provide_srbt():
        #This function is designed to determine whether the user should be advised to follow
sleep restriction or not.
        try:
        #Creating a container for the sleep efficiency per night of the last week
        efficiency=[]

      #Creating a list of the sleep efficiency per night for the last week
        for i in data['sleep'][-8:-1]:
        efficiency.append(i['efficiency'])

      #Determining the average sleep efficiency of last week
        avg_efficiency=np.mean(efficiency)

      if avg_efficiency<80 or avg_efficiency>85:
        return True
        else:
        return False
        except:
        print('An error occurred.')
```

Increment time in bed in steps of 15 to 30 minutes

In [38]:

```
def new_sleep_restriction_bedtime():
        #This function is designed to determine the new sleep restriction bedtime.
        try:
        #Creating a container for the sleep efficiency per night of the last week
        efficiency=[]

      #Creating a list of the sleep efficiency per night for the last week
        for i in data['sleep'][-8:-1]:
        efficiency.append(i['efficiency'])

      #Determining the average sleep efficiency of last week
        avg_efficiency=np.mean(efficiency)

      #Printing the average sleep efficiency of last week
        print('The average sleep efficiency of last week is: {}.'.format(avg_efficiency))

        #Determining the new sleep restriction bedtimes based on the sleep efficiency of last
week
        if avg_efficiency<80:
        new_srbt1=sleep_restriction_bedtime()+15
        new_srbt2=sleep_restriction_bedtime()+30
```

```
        return new_srbt1, new_srbt2
        elif avg_efficiency>=80 and avg_efficiency<85:
        new_srbt1=sleep_restriction_bedtime()
        new_srbt2=sleep_restriction_bedtime()

        return new_srbt1, new_srbt2
        else:
        new_srbt1=sleep_restriction_bedtime()-30
        new_srbt2=sleep_restriction_bedtime()-15

        return new_srbt1, new_srbt2
        except:
        print('An error occurred.')
```

In [39]:

```
def new_sleep_restriction_bedtime_print():
        #This function is designed to print the advice for the new sleep restriction bedtime.
        try:
        nsrbt1, nsrbt2=new_sleep_restriction_bedtime()
        if nsrbt1==sleep_restriction_bedtime() and nsrbt2==sleep_restriction_bedtime():
        print('The user\'s allowed bedtime is good.\nThis is at
{}.'.format(time.strftime("%H:%M", time.gmtime(nsrbt1*60))))
        elif nsrbt1<sleep_restriction_bedtime() and nsrbt2<sleep_restriction_bedtime():
        print('The user\'s allowed bedtime may be increased by 15 to 30 minutes, depending
on how tired they feel during the day.\nThis is between {} and {} on the
clock.'.format(time.strftime("%H:%M", time.gmtime(nsrbt1*60)), time.strftime("%H:%M",
time.gmtime(nsrbt2*60))))
        else:
        return print('The user\'s allowed bedtime should be further restricted by 15 to 30
minutes, depending on how tired they feel.\nThis is between {} and {} on the
clock.'.format(time.strftime("%H:%M", time.gmtime(nsrbt1*60)), time.strftime("%H:%M",
time.gmtime(nsrbt2*60))))
        except:
        print('An error occurred.')
```

In [40]:

```
def provide_nsrbt():
        #This function is designed to determine whether the user should be advised a new
sleep restriction bedtime or not.
        #This function should be ran two weeks after initiating sleep restriction therapy and
every week after, until the new sleep restriction bedtime stays the same for a longer period
of time (in the range of months).
        try:
        efficiency=[]
```

```
    #Creating a list of the sleep efficiency per night for the last week
        for i in data['sleep'][-8:-1]:
        efficiency.append(i['efficiency'])

    #Determining the average sleep efficiency of last week
        avg_efficiency=np.mean(efficiency)

    #Ensuring that the new sleep restriction bedtime cannot be less than 5.5 hours of sleep
        if (avg_efficiency<80 and
abs(wake_time()-sleep_restriction_bedtime())>=((5.5*60)+30)) or avg_efficiency>85:
        return True
        else:
        return False
        except:
        print('An error occurred.')
```

Putting everything together

In [41]:

```
counter, _=gaps()
if counter>0:
        gaps_print()
        safe_date_dif_last_gap_print()
print('')
wake_time_print()
print('')
no_napping_print()
print('')
get_out_of_bed_print()
print('')
exercise_regularly_print()
print('')
sleep_restriction_bedtime_print()
print('')
new_sleep_restriction_bedtime_print()
```

```
There is 1 gap in the data, comprising a total of 1 days.
The missing data does not occur within the last two weeks. It is safe to trust any advices based on this data.

The advised rising time is: 07:43.
This advice can be rounded to the nearest quarter if wished for.

It looks like you have not been taking naps lately. This is healthy behavior that is good for your sleep quality. Keep it up!
```

It looks like you have not stayed in bed whilst being awake for a prolonged period of time within the last two weeks.
This is considered to be good sleep hygiene, keep it up!

It looks like you have been exercising enough lately. This is healthy behavior that is good for your sleep quality. Keep it up!

The ideal sleeping time for this user when applying sleep restriction therapy is at 00:19.

The average sleep efficiency of last week is: 85.42857142857143.
The user's allowed bedtime may be increased by 15 to 30 minutes, depending on how tired they feel during the day.
This is between 23:49 and 00:04 on the clock.

Automating the selection of relevant micro-interventions

In [42]:

```
def relevant_selection():
        #This function is designed to provide a selection of micro-interventions that are relevant for the user
        try:
        #Determining if the advices provided are trustworthy or not
        counter, _=gaps()
        if counter>0:
        gaps_print()
        safe_date_dif_last_gap_print()
        print('')

      #Determining which advice should be provided and which not
        #If yes the advice shall be printed
        if provide_wake_time():
        wake_time_print()
        if provide_no_napping():
        no_napping_print()
        if provide_get_out_of_bed():
        get_out_of_bed_print()
        if provide_exercise_regularly():
        exercise_regularly_print()
        if provide_srbt():
        sleep_restriction_bedtime_print()
        if provide_nsrbt():
        new_sleep_restriction_bedtime_print()
```

```
    return print('')
    except:
    print('An error occurred')
```

In [43]:

```
#The actual selection
relevant_selection()
```

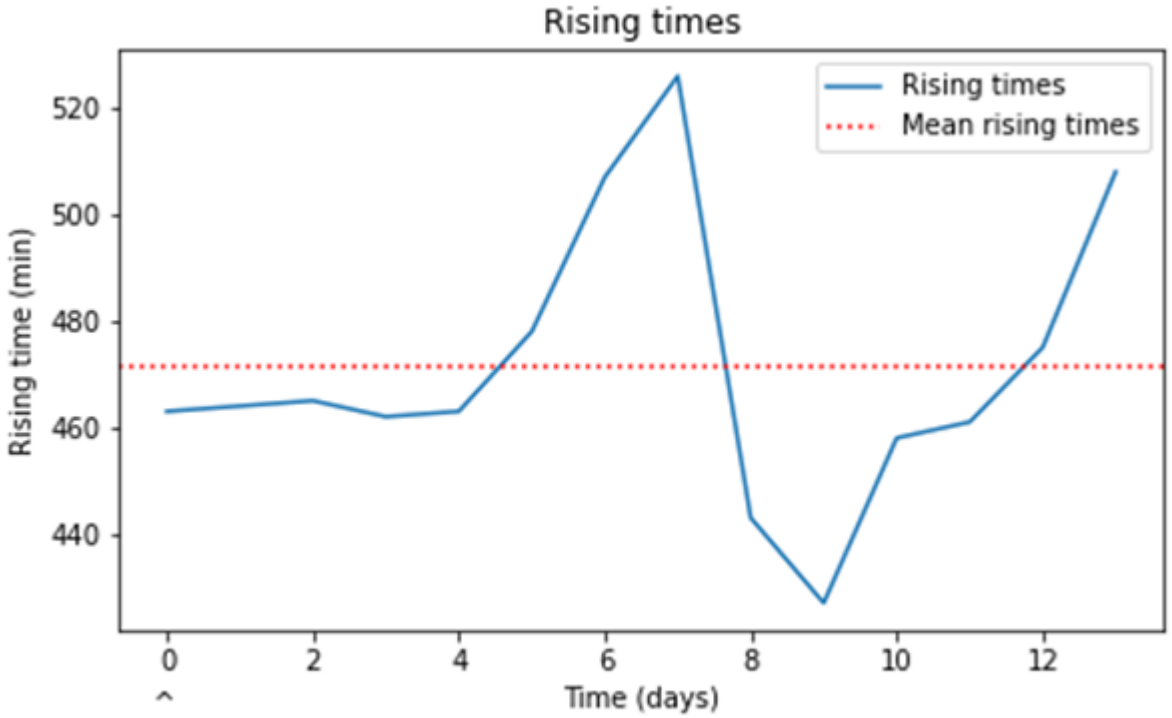
```
There is 1 gap in the data, comprising a total of 1 days.
The missing data does not occur within the last two weeks. It is safe to trust any advices based on this data.

The advised rising time is: 07:43.
This advice can be rounded to the nearest quarter if wished for.
The ideal sleeping time for this user when applying sleep restriction therapy is at 00:19.
The average sleep efficiency of last week is: 85.42857142857143.
The user's allowed bedtime may be increased by 15 to 30 minutes, depending on how tired they feel during the day.
This is between 23:49 and 00:04 on the clock.
```

Plotting the variables needed for validation

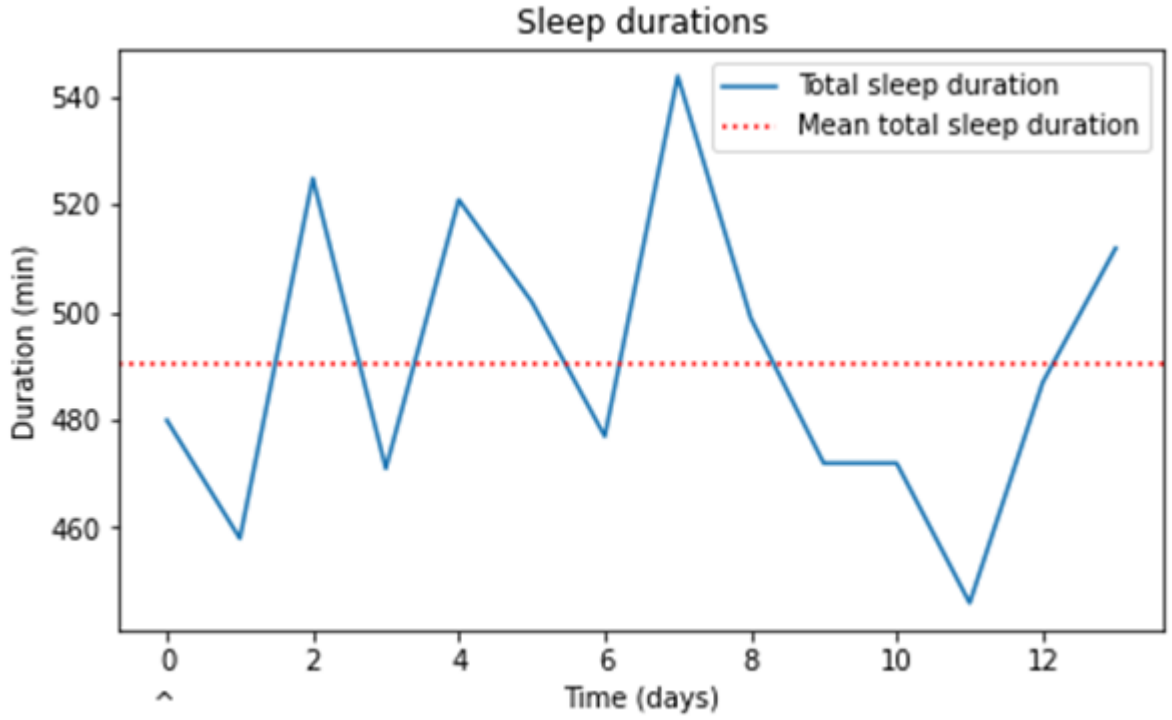
In [44]:

```
#Plotting the rising times in minutes for the entire dataset and plotting the DST switching dates
plot(rising_times_m[-15:-1], [], [], 'Rising times', [np.mean(rising_times_m[-15:-1])], [], [], False, 'Time (days)', [], [], [], 'Rising time (min)', [], [], [], ['Rising times', 'Mean rising times'], [0.15, -0.025, '^\nThe mean rising time is: {}.'.format(time.strftime("%H:%M", time.gmtime(np.mean(rising_times_m[-15:-1])*60)))])

#Creating a list of sleep durations
durations_m=[]
for i in data['sleep']:
    durations_m.append(i['duration']/60)

#Plotting the sleep duration times in minutes for the entire dataset and plotting the DST switching dates
plot(durations_m[-15:-1], [], [], 'Sleep durations', [np.mean(durations_m[-15:-1])], [], [], False, 'Time (days)', [], [], [], 'Duration (min)', [], [], [], ['Total sleep duration', 'Mean total sleep duration'], [0.15, -0.025, '^\nThe mean sleep duration is: {} minutes.'.format(time.strftime("%H:%M", time.gmtime(np.mean(durations_m[-15:-1])*60)))])

#Reshaping the sleeping_times_m data
r_sleeping_times_m=[]
for i in range(0,len(sleeping_times_m)):
```

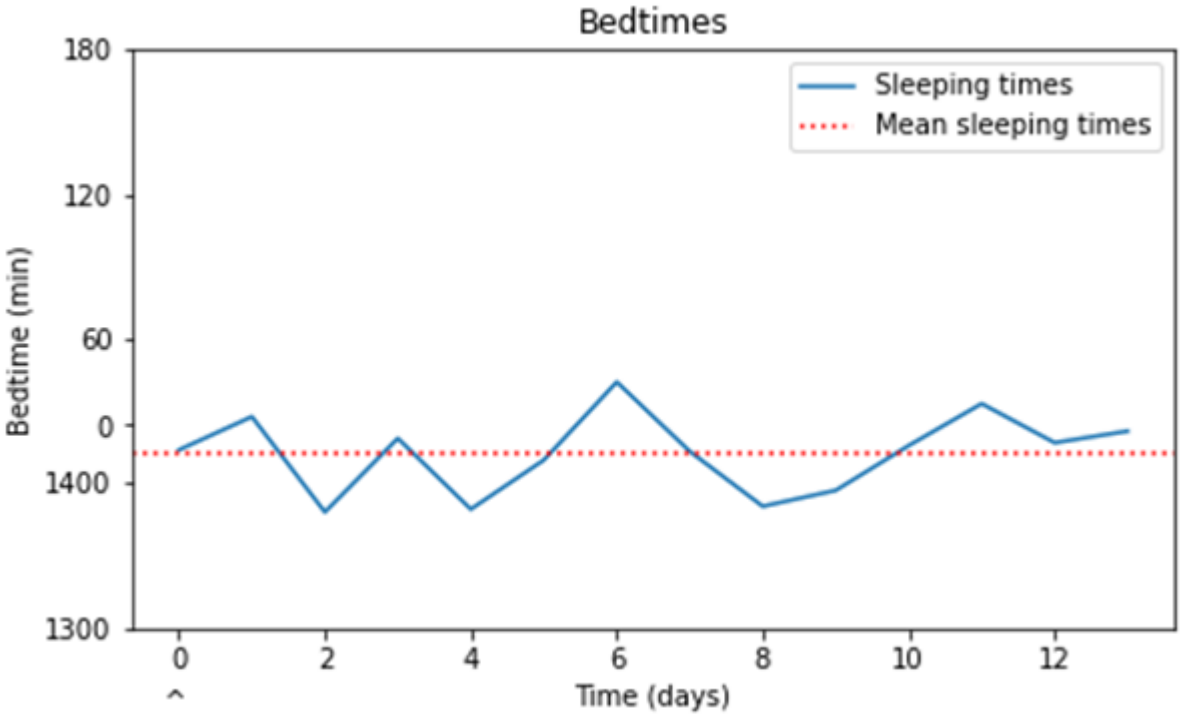
```
        if sleeping_times_m[i]<(4*60):
        r_sleeping_times_m.append(sleeping_times_m[i]+(24*60))
        else:
        r_sleeping_times_m.append(sleeping_times_m[i])

#Plotting the reshaped sleeping times in minutes for the entire dataset and plotting the DST
switching dates
plot(r_sleeping_times_m[-15:-1], [], [], 'Bedtimes', [np.mean(r_sleeping_times_m[-15:-1])], [],
[], True, 'Time (days)', [], [], [], 'Bedtime (min)', [1300, 1400, 1440, 1500, 1600, 1700, 1800],
[1300, 1400, 0, 60, 120, 180, 240], [1300, 1700], ['Sleeping times', 'Mean sleeping times'],
[0.15, -0.025, '^\nThe mean sleeping time is: {}.'.format(time.strftime("%H:%M",
time.gmtime(np.mean(r_sleeping_times_m[-15:-1])*60)))])
```



The mean rising time is: 07:51.



The mean sleep duration is: 08:10 minutes.



The mean sleeping time is: 23:41.

Appendix H

This code is part of the bachelor thesis project: Personalizing Digital Cognitive Behavioral Therapy for Insomnia Through Automation of Micro-Intervention Selection Based on Data from Wearable Techology; A Case Study on the Oura Ring

Author Information

Author: Sjors Weggeman\ Student number: s4799771\ University: Radboud University Nijmegen\ Email address: sjors.weggeman@student.ru.nl

Imports

In [1]:

```
#Importing the necessary packages
import json
import matplotlib
import matplotlib.pyplot as plt
import numpy as np
import re
import time

from datetime import date
from statsmodels import robust
```

Loading the data

In [2]:

```
#Opening the file
file=open('oura2.json')

#Loading the file into a workable dataframe
data=json.load(file)
```

Inspecting the data

In [3]:

```
#Checking the length of the data
for i in data:
```

```
        print('The \'{}\' data contains {} entries.'.format(i, len(data[i])))
```
```
The 'activity' data contains 49 entries.
The 'readiness' data contains 44 entries.
The 'restful_periods' data contains 26 entries.
The 'sleep' data contains 44 entries.
```

Preprocessing

In [4]:

```
def time_min_conversion(time_data):
        #This function is designed to extract the digital time from the 'bedtime_start' and
'bedtime_end' variables and convert it to minutes.
   #Accordingly, time_data must be of the form string and can only have values
'bedtime_start' and 'bedtime_end'.
        try:
        #Container for the times in minutes
        times_m=[]

        #Performing steps for every entry in the 'sleep' data
        for i in data['sleep']:
        #Extracting the time out of the concatenated data
        time=re.split(r"[-T+:]", i[time_data])[3:6]

        #Transforming the time to minutes
        if int(time[2])<30:
                time_m=int(time[0])*60+int(time[1])
        else:
                time_m=int(time[0])*60+int(time[1])+1

        #Storing the time in the containers
        times_m.append(time_m)

      return times_m
        except:
        print('An error occurred.')
```

In [5]:

```
#The actual conversion
sleeping_times_m=time_min_conversion('bedtime_start')
rising_times_m=time_min_conversion('bedtime_end')
```

In [6]:

```
def date_conversion(date2conv):
```

```
        #This function is designed to convert a date provided in string format
('year-month-day') to int format (year, month, day) so it can be used by the 'date' function
from datetime.
        try:
        #Extracting the date data from the string
        conv_date_str=re.split(r"[-]",date2conv)

        #Converting the string representations to integers
        conv_date_int=[int(i) for i in conv_date_str]

    return conv_date_int
        except:
        print('An error occurred.')
```

In [7]:

```
def date_difference(date1,date2):
        #This function is designed to calculate the number of days difference between two
given dates.
        #Hereby the earliest date needs to be provided first, otherwise a negative number will
be returned.
        try:
        #Converting the provided dates to comparable formats
        d1=date(date1[0], date1[1], date1[2])
        d2=date(date2[0], date2[1], date2[2])

    #Comparing the dates
        dif=d2-d1

    return dif.days
        except:
        print('An error occurred.')
```

In [8]:

```
dd=date_difference(date_conversion(data['sleep'][0]['summary_date']),
date_conversion(data['sleep'][-1]['summary_date']))+1
print('The difference between {} and {} (included) is {} days.\nThe dataset contains {} entries
in the \'sleep\' data.\nThis means that there are {} missing days in the \'sleep\' data,
assuming the dataset does not contain any duplicate dates.'
        .format(data['sleep'][0]['summary_date'], data['sleep'][-1]['summary_date'], dd,
len(data['sleep']), abs(len(data['sleep'])-dd)))
```

```
The difference between 2021-04-15 and 2021-05-31 (included) is 47 days.
The dataset contains 44 entries in the 'sleep' data.
This means that there are 3 missing days in the 'sleep' data, assuming the dataset does not
contain any duplicate dates.
```

In [9]:

```
def gaps():
        #This function is designed to find the number of gaps in the data and their respective
sizes, where a gap is one or more consecutive missing days.
        try:
        #Creating a counter for the number of gaps and a container for the sizes of the gaps
        counter=0
        diff=[]

      #Working through all the gaps
        for i in range(0,len(data['sleep'])-1):
        if date_difference(date_conversion(data['sleep'][i]['summary_date']),
                           date_conversion(data['sleep'][i+1]['summary_date']))>1:

            #Determining the size of the current gap
                diff.append(date_difference(date_conversion(data['sleep'][i]['summary_date']),
                                date_conversion(data['sleep'][i+1]['summary_date']))-1)

            #Determining the total number of gaps
                counter+=1

      return counter, diff
        except:
        print('An error occurred.')
```

In [10]:

```
def gaps_print():
        #This function is designed to print the results from the 'gaps' function in a nice format.
        try:
        c, d=gaps()
        if c==1:
        return print('There is {} gap in the data, comprising a total of {} days.'.format(c,
np.sum(d)))
        elif c>1:
        return print('There are {} gaps in the data, comprising a total of {} days.\nThe average
gap size is approximately {} days and the largest gap size is {} days.'.format(c, np.sum(d),
np.round(np.mean(d)), np.max(d)))
        else:
        return print('There are no gaps in the data.')
        except:
        print('An error occurred.')
```

In [11]:

```
#The actual printing
gaps_print()
```

```
There are 3 gaps in the data, comprising a total of 3 days.
The average gap size is approximately 1.0 days and the largest gap size is 1 days.
```

In [12]:

```
def last_gap_date():
        #This function has been designed to determine the first date after the last gap in the
data.
        try:
        dd=0
        last_gap_date=''

      #Working through all the gaps
        for i in range(0,len(data['sleep'])-1):
        dd=date_difference(date_conversion(data['sleep'][i]['summary_date']),
date_conversion(data['sleep'][i+1]['summary_date']))-1

        #Determining the first date after the last gap in the data
        if dd>0:
                last_gap_date=data['sleep'][i+1]['summary_date']

      return last_gap_date
        except:
        print('An error occurred.')
```

In [13]:

```
def pre_last_gap_date():
        #This function has been designed to determine the first date after the second to last
gap in the data.
        try:
        c,d=gaps()
        if c>1:
        dd=0
        pre_last_gap_date=''

        #Working through all the gaps
        for i in range(0,len(data['sleep'])-1):
                dd=date_difference(date_conversion(data['sleep'][i]['summary_date']),
date_conversion(data['sleep'][i+1]['summary_date']))-1
                #Determining the previous gap
                if dd>0:
                c-=1
                #Determining the first date after the pre-last gap in the data
                if c==0:
                pre_last_gap_date=data['sleep'][i+1]['summary_date']

        return pre_last_gap_date
        except:
        print('An error occurred.')
```

In [14]:

```
def safe_date_dif_last_gap_print():
        #This function has been designed to print whether it is safe to trust the advices based on this data, based on the size and location of the gaps with respect to the day of the last 'sleep' data-entry contained in the dataset.
        try:
        c,d=gaps()
        greatest_gap=np.max(d)
        if pre_last_gap_date():
        pldd=date_difference(date_conversion(pre_last_gap_date()), date_conversion(data['sleep'][-1]['summary_date']))+1
        else:
        pldd=15
        ldd=date_difference(date_conversion(last_gap_date()), date_conversion(data['sleep'][-1]['summary_date']))+1
        if pldd<=14:
        print('The missing data occurs within the last two weeks on more than one occasion. It is unwise to trust any advices based on this data.')
        elif pldd>14 and ldd<=14 and greatest_gap>1:
        print('The missing data occurs within the last two weeks on one occasion, but is larger than one day. It is unwise to trust any advices based on this data.')
        elif pldd>14 and ldd<=14 and greatest_gap<1:
        print('The missing data occurs within the last two weeks on one occasion, but is not larger than one day. It is safe to trust any advices based on this data.')
        else:
        print('The missing data does not occur within the last two weeks. It is safe to trust any advices based on this data.')
        except:
        print('An error occurred.')
```

In [15]:

```
#The actual printing
safe_date_dif_last_gap_print()
```

```
The missing data does not occur within the last two weeks. It is safe to trust any advices based on this data.
```

Checking for trends in the data based on Daylight Saving Time (DST)

Since this dataset comprises little more than a month it is not worth checking if there are any trends caused by DST.

I will still need to plot data:

In [16]:

```
def plot(data_p1_x, data_p1_y, data_p2, title, axh, axv, col, spec_labs, xlab, xmajloc,
xticklabs, xlim, ylab, ymajloc, yticklabs, ylim, legend, figtext):
        #This function is designed to make it possible to plot a graph using just one line of
code.
   try:
        if col:
        colours=col
        linestyles=['solid']
        else:
        colours=['r', 'g', 'm']
        linestyles=['dotted', 'dashed', 'dashdot']

        fig,ax=plt.subplots(figsize=(7, 4))
        if not data_p1_y:
        ax.plot(data_p1_x)
        else:
        ax.plot(data_p1_x, data_p1_y)
        if(data_p2):
        ax.plot(data_p2)
        ax.set_title(title)
        if axh:
        for j in range(0,len(axh)):
                ax.axhline(axh[j], c=colours[j%len(colours)],
linestyle=linestyles[j%len(linestyles)])
        if axv:
        for k in range(0,len(axv)):
                ax.axvline(axv[k], c=colours[k%len(colours)])
        ax.set_xlabel(xlab)
        if spec_labs:
        if xmajloc:
                ax.xaxis.set_major_locator(matplotlib.ticker.FixedLocator(xmajloc))
                if xticklabs:
                ax.set_xticklabels(xticklabs)
        if ymajloc:
                ax.yaxis.set_major_locator(matplotlib.ticker.FixedLocator(ymajloc))
                if yticklabs:
                ax.set_yticklabels(yticklabs)
         else:
        if xmajloc:
                ax.xaxis.set_major_locator(matplotlib.ticker.FixedLocator(xmajloc[0]))
                if xticklabs:
                ax.set_xticklabels(xticklabs[0])
        if ymajloc:
                ax.yaxis.set_major_locator(matplotlib.ticker.FixedLocator(ymajloc[0]))
                if yticklabs:
                ax.set_yticklabels(yticklabs[0])
        if xlim:
        ax.set_xlim(xlim)
```

```
        ax.set_ylabel(ylab)
        if ylim:
        ax.set_ylim(ylim)
        if legend:
        ax.legend(legend)
        if figtext:
        fig.text(figtext[0], figtext[1], figtext[2])
        plt.show()
        except:
        print('An error occurred.')
```

Basing Micro Interventions on Data

Wake at the same time every day

In [17]:

```
def wake_time():
        #This function is designed to determine the ideal wake time in minutes.
        try:
        #Determining the ideal wake time in minutes
        wake_time=int(np.median(rising_times_m[-15:-1]))

    return wake_time
        except:
        print('An error occurred.')
```

In [18]:

```
def wake_time_print():
        #This function is designed to print the advice for the ideal wake time, with the ideal
wake time in digital format.
        try:
        return print('The advised rising time is: {}.\nThis advice can be rounded to the nearest
quarter if wished for.'.format(time.strftime("%H:%M", time.gmtime(wake_time()*60))))
        except:
        print('An error occurred.')
```

In [19]:

```
def provide_wake_time():
        #This function is designed to determine whether the user should be advised to wake
at the same time every day.
        try:
        rising_times_m14=[]

    #Converting the rising times to minutes for if that had not been done yet
        rising_times_m=time_min_conversion('bedtime_end')
```

```
    #Storing the rising times of the last two weeks
      for i in range(0, 14):
      rising_times_m14.append(rising_times_m[-15+i])

    #Determining the median of the rising times of the last two weeks
      med=np.median(rising_times_m14)

    #Determining the median absolute deviation for the rising times of the last two weeks

    mad=robust.mad(rising_times_m14)

    if rising_times_m14[-1]<(med-mad) or rising_times_m14[-1]>(med+mad):
      return True
      else:
      return False
      except:
      print('An error occurred.')
```

No Napping

In [20]:

```
def within_14_days_check(conv_date):
      #This function is designed to check if a date lies within 14 days of the day of the last
'sleep' data-entry contained in the dataset.
      try:
      #Getting the last date from the dataset
      ref_date=data['sleep'][-1]['summary_date']

      #Converting the dates for comparison
      conv_ref_date=date_conversion(ref_date)

    #Comparing the dates
      if date_difference(conv_date, conv_ref_date)>=0 and date_difference(conv_date,
conv_ref_date)<=14:
      return True
      else:
      return False
      except:
      print('An error occurred.')
```

In [21]:

```
def no_napping():
      #This function is designed to determine whether any naps have been taken in the
last two weeks and returns a list of them.
      try:
```

```
        #Container for naps
        naps=[]

    #Converting sleeping times to minutes
        sleeping_times_m=time_min_conversion('bedtime_start')

    #Adding unconfirmed naps within the last two weeks
        for i in data['restful_periods']:
        if within_14_days_check(date_conversion(i['summary_date'])):
                if i['duration']>=(15*60) and i['duration']<(3*60*60):
                naps.append(i)

    #Adding confirmed naps within the last two weeks based on period id and unusual
sleeping times
        for j in range(0, len(data['sleep'])):
        if j>len(data['sleep'])-15:
                if data['sleep'][j]['period_id']>0 and sleeping_times_m[j]>=(7*60) and
sleeping_times_m[j]<(19*60):
                naps.append(data['sleep'][j])

    return naps
        except:
        print('An error occurred')
```

In [22]:

```
def no_napping_print():
        #This function is designed to print the advice for no napping
        try:
        #Checking if there are any naps and giving advise if any naps have been found
    if not len(no_napping())==0:
        return print('It looks like you have been taking naps lately. If you feel like your sleep
quality has suffered it might be better not to take naps anymore.\nThese are the naps:
{}'.format(no_napping()))
        else:
        return print('It looks like you have not been taking naps lately. This is healthy
behavior that is good for your sleep quality. Keep it up!')
        except:
        print('An error occurred')
```

In [23]:

```
def provide_no_napping():
        #This function is designed to determine whether the user should be advised not to
nap anymore.
        try:
        if not len(no_napping())==0:
        return True
```

```
    else:
return False
except:
print('An error occurred.')
```

Get out of bed when unable to sleep

In the creation of this automation process it was found that more information was needed to determine whether the medium activity level determined by the Oura team indeed corresponds to walking level, or that it is more likely that the low activity corresponds to walking. The user of the first dataset only wore the ring at night, which might have influenced the accuracy of the activity data collection. Through which means this activity data was collected for him remains unknown, but it is most likely collected through another wearable and its corresponding application, which is connected to Apple Health, the same application the Oura ring and its corresponding application are connected to. Or it might just plainly have been recorded by Apple Health using the phone of the user. Regardless of the way through which the activity data was recorded, it remains unclear how it was recorded exactly and thus its accuracy remains unclear too. Therefore it is important to also have a look at the data from the other two users, who said to wear the ring day and night.

To remind you of what we were looking at:

For getting out of bed when unable to sleep we are going to look at three aspects:

1. The sleep onset latency: How long does the person stay awake before falling asleep after going to bed.
2. If and for how long the user woke up during the night and whether he stayed in bed or not.
3. How long the user was awake before going out of bed.

These aspects are all combined into a score: The sleep efficiency.

The most reliable way to determine whether the user stayed in bed or not is by self-report, so if an application is being created questions should be asked when it was detected that the user was awake for a prolonged period of time during the night. Then the user can specify his course of actions or the lack there off.

A little bit less reliable, but technological data collection method is the activity measurement. This data is collected in 5 minute intervals and has been separated into 5 classes:

0. Non-wear \ 1. Rest \ 2. Inactive \ 3. Low activity \ 4. Medium activity \ 5. High activity

This data collection starts from 4am and lasts until 4am the next day, hence I'll have to combine the last bit of data from one day with the first bit of data from the next day to get the data for that night.

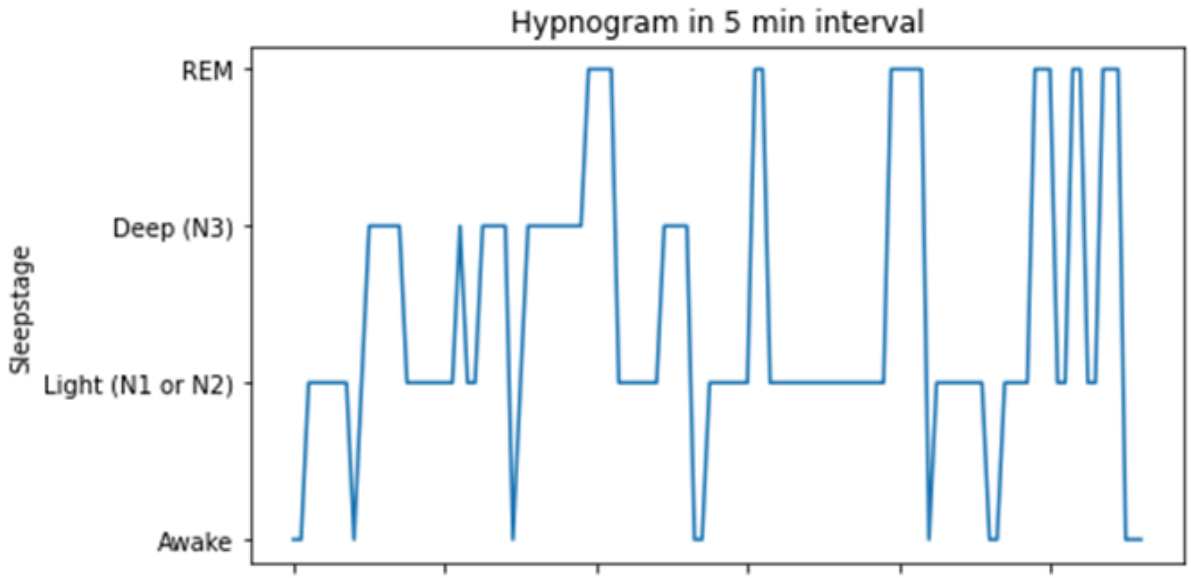
So first we have to find out when the user was awake, which is contained in the hypnograms:

Inspecting the data

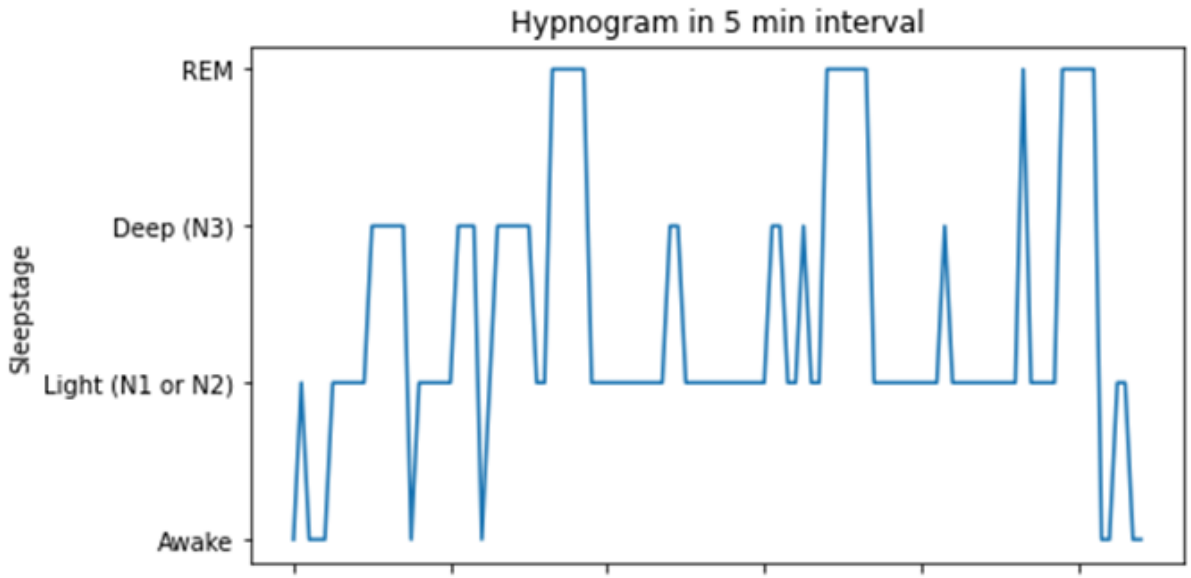
In [24]:

```
#Plotting the hypnograms for the last week
```

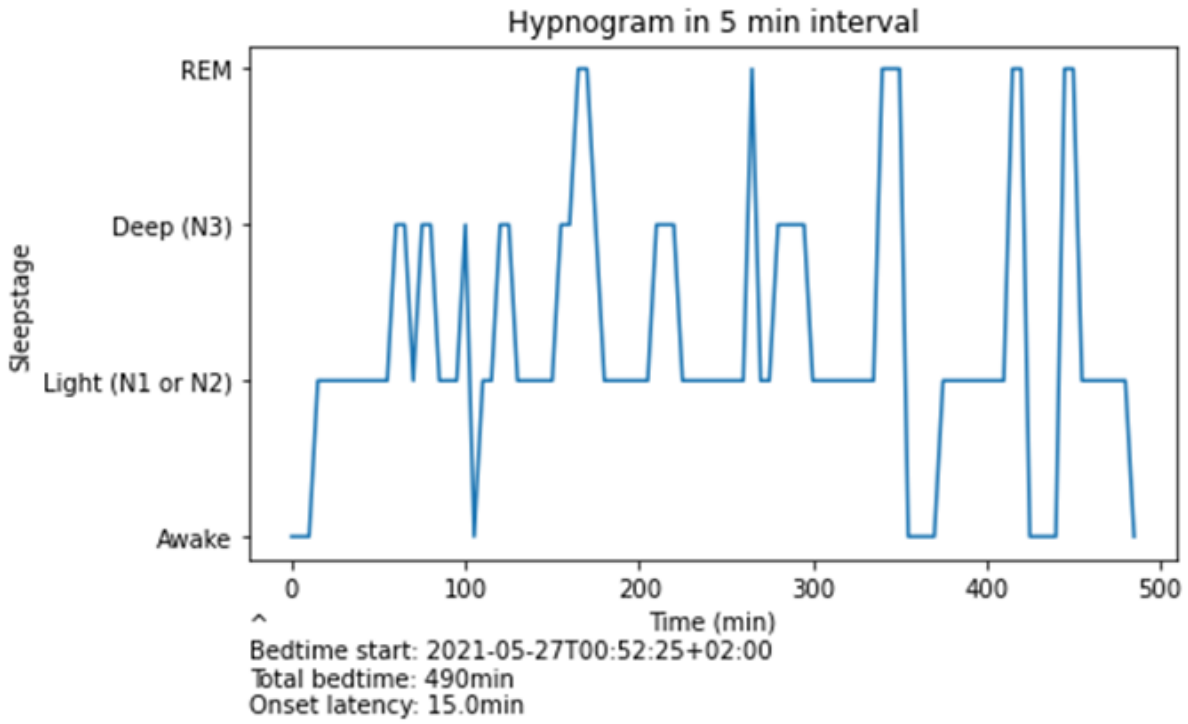
```
counter=-1
#Storing the hypnograms for later use
hgs5=[]
for i in data['sleep']:
        counter+=1
        hg5_conc=i['hypnogram_5min']
        hg5_split=[]
        if counter>(len(data['sleep'])-8):
        for j in hg5_conc:
        hg5_split.append(j)
        hgs5.append(hg5_split)
        plot(hg5_split, [], [], 'Hypnogram in 5 min interval', [], [], [], False, 'Time (min)',
[np.arange(0, 161, 20)],[np.arange(0, 801, 100)], [], 'Sleepstage', [np.arange(0, 4)], [['Awake',
'Light (N1 or N2)', 'Deep (N3)', 'REM']],[], [], [0.125, -0.1, '^\nBedtime start: {}\nTotal bedtime:
{}min\nOnset latency: {}min'.format(i['bedtime_start'], int(i['duration']/60),
i['onset_latency']/60)])
```



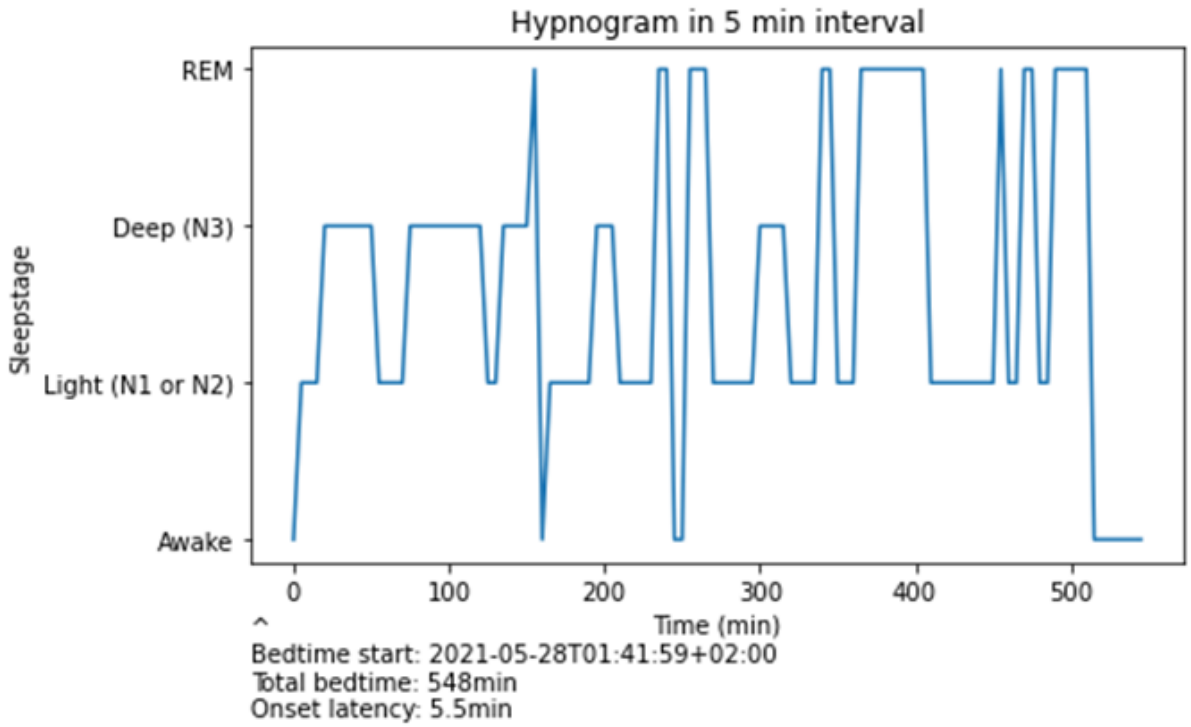
Bedtime start: 2021-05-24T23:33:00+02:00
Total bedtime: 564min
Onset latency: 13.5min



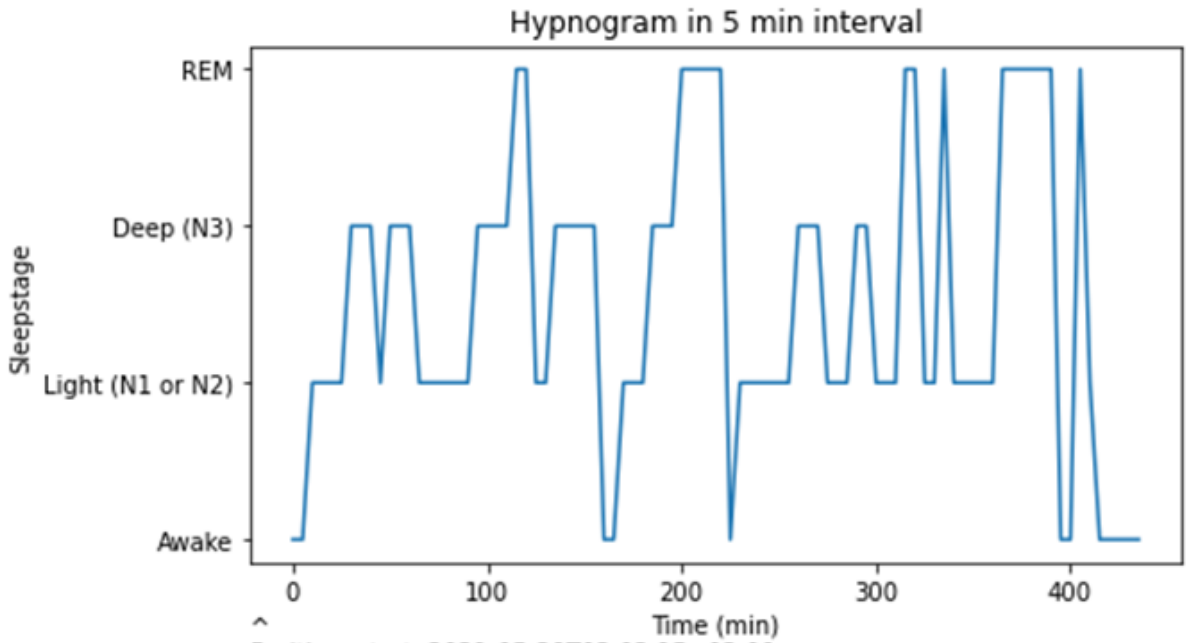
Bedtime start: 2021-05-26T00:07:47+02:00
Total bedtime: 542min
Onset latency: 26.5min



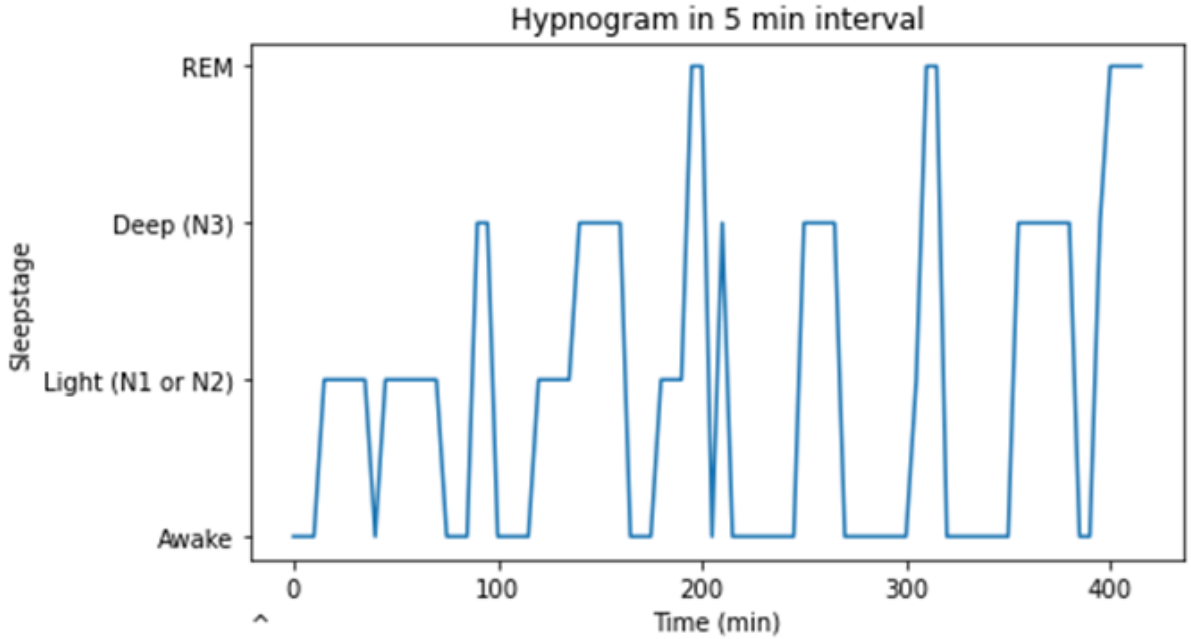
Bedtime start: 2021-05-27T00:52:25+02:00
Total bedtime: 490min
Onset latency: 15.0min



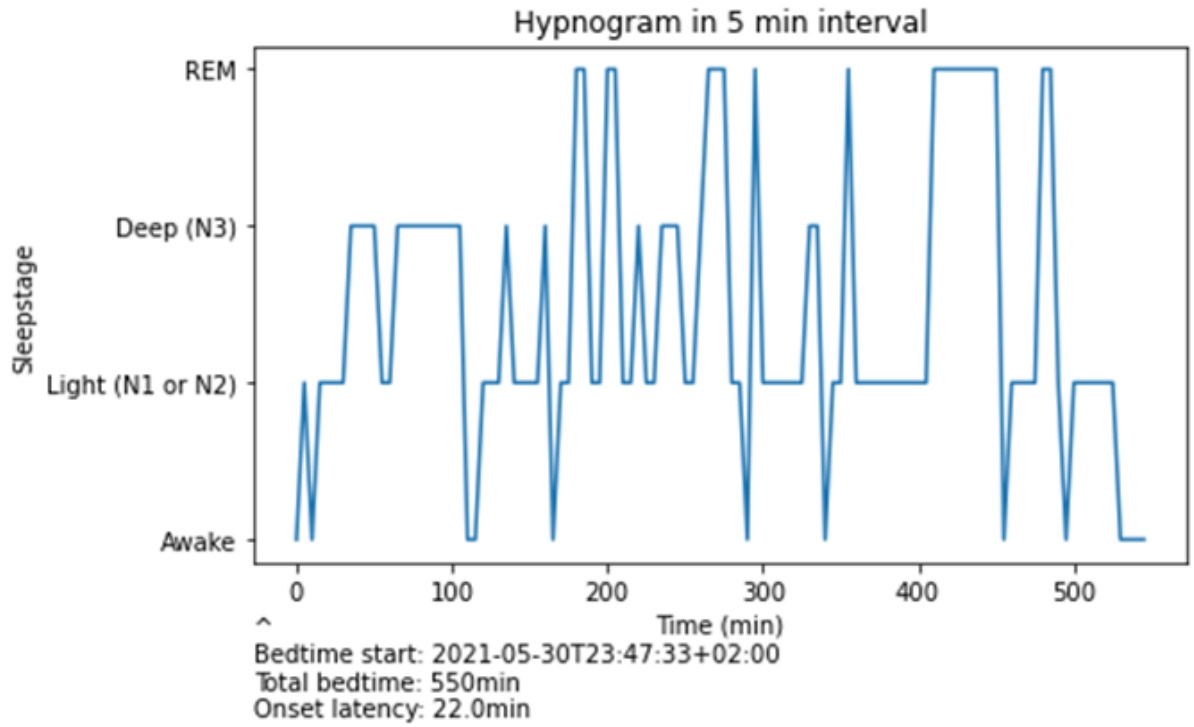
Bedtime start: 2021-05-28T01:41:59+02:00
Total bedtime: 548min
Onset latency: 5.5min



Bedtime start: 2021-05-29T02:02:15+02:00
Total bedtime: 439min
Onset latency: 8.0min



Bedtime start: 2021-05-30T03:45:00+02:00
Total bedtime: 416min
Onset latency: 2.5min



As can be seen in the Oura API documentation (Ōura Health Oy, n.d.), Oura defines a sleep onset latency (SOL) of less than 15 minutes to be good, 15 minutes as normal and more than 15 minutes problematic. Since we are using the Oura ring, this is the standard that I will be using as well.

For the last week in the dataset, this user mostly had good and normal SOL's, except for the second and seventh night. The third and the fifth night contained awakenings that lasted longer than 15 minutes. The fourth and fifth night the user stayed in bed for longer than 15 minutes after waking up at the end of the night.

Inspecting full day activity spectra

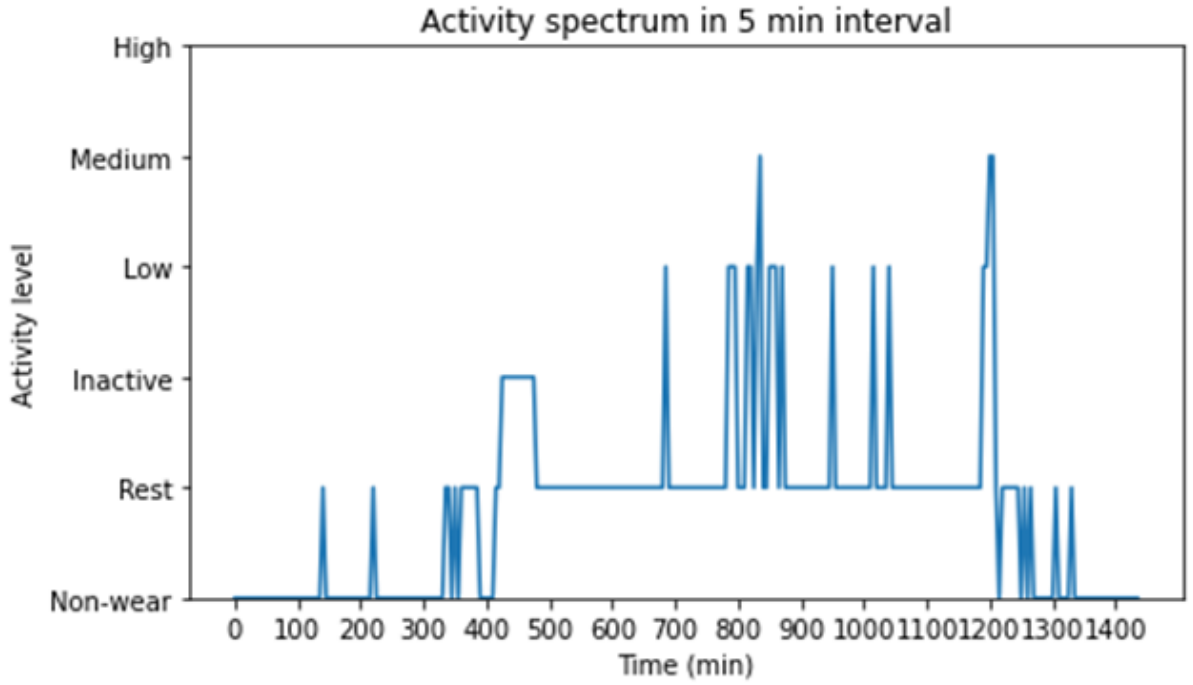
For the following activity spectra it is useful to understand the difference between the respective activity levels.\ According to the Oura API documentation (Ōura Health Oy, n.d.) the levels are defined as follows:

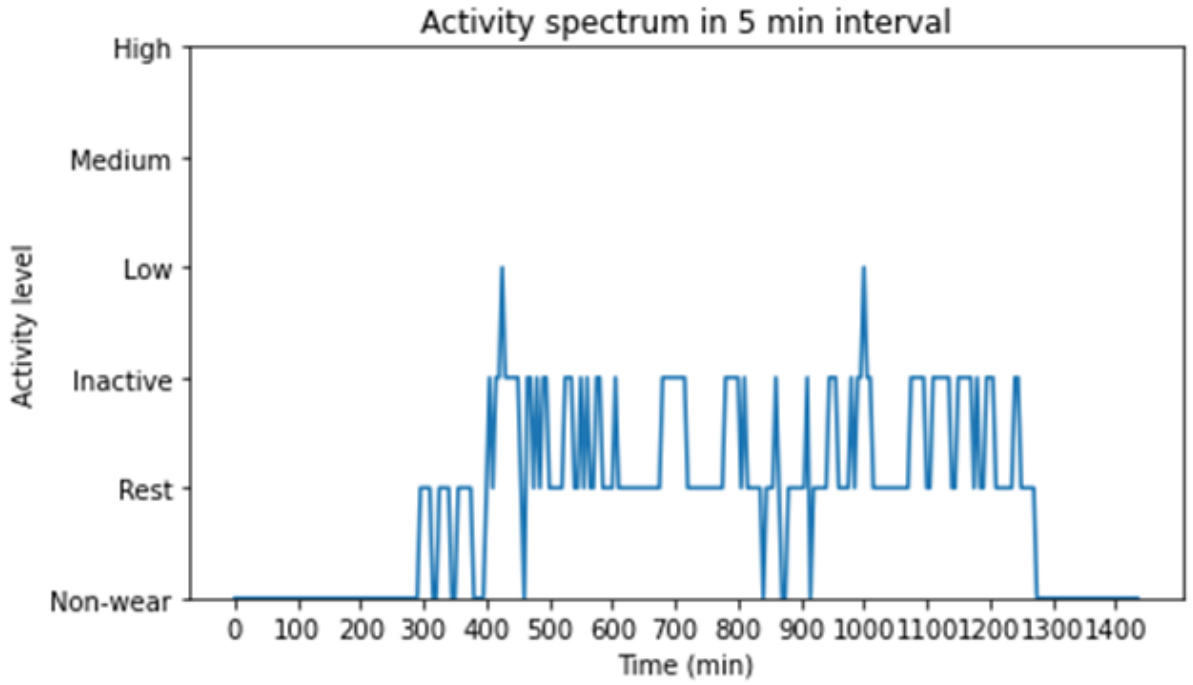
Metabolic-equivalent (MET) minutes express how much more energy the user is burning than if they were sitting still.

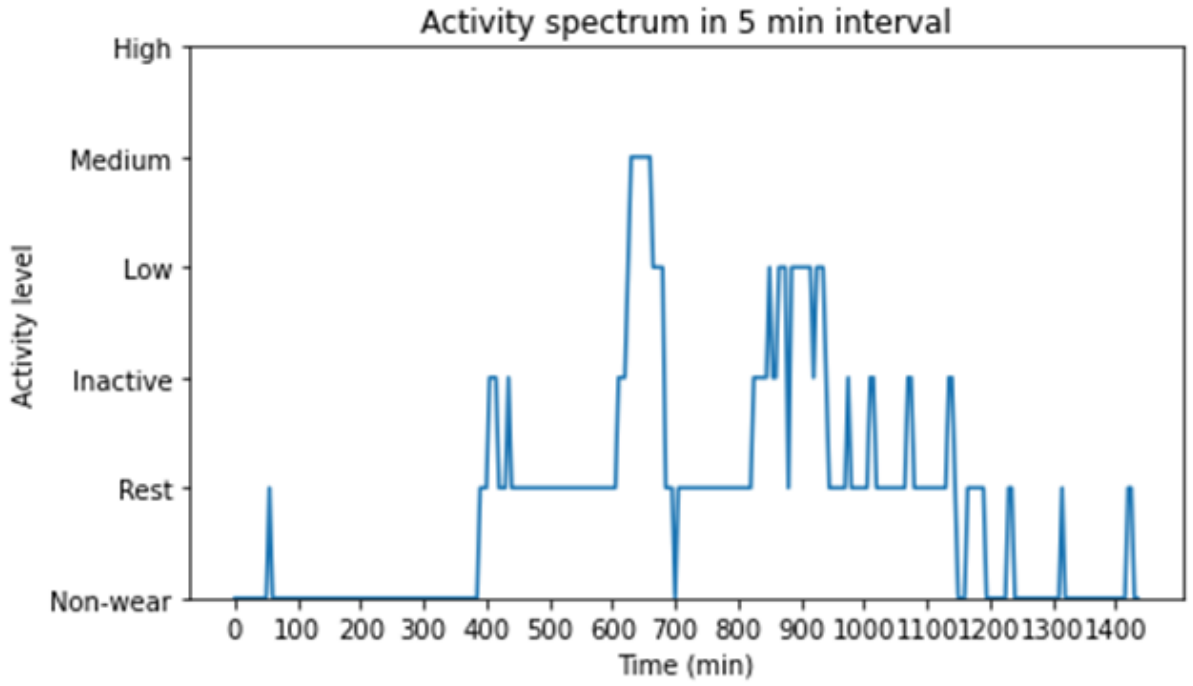
Non-wear: Not wearing the ring
Rest: Lying down or sleeping, corresponds to an average MET of <1.05.
Inactive: Sitting or standing still, corresponds to an average MET of 1.05 to 2.
Low: Low intensity activity, e.g. Household work, corresponds to an average MET of 2 to the age dependent limit.
Medium: Medium intensity activity, e.g. Walking, average MET level depends on age and gender.
High: High intensity activity, e.g. Running, average MET level depends on age and gender.

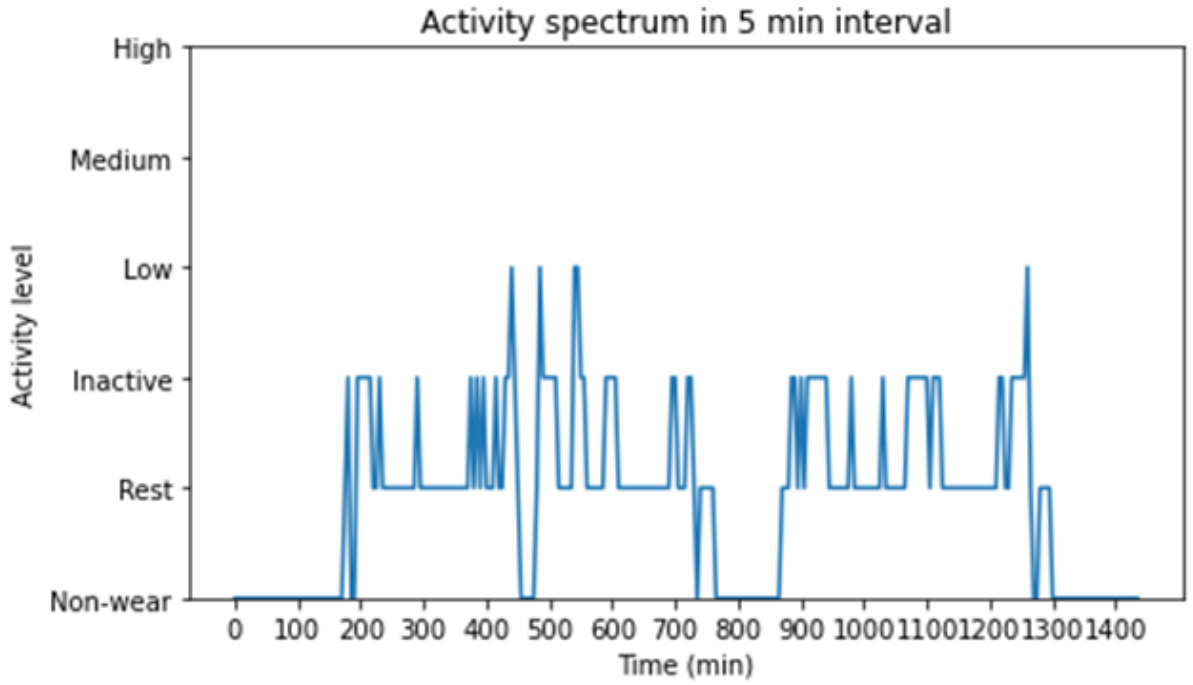
In [25]:

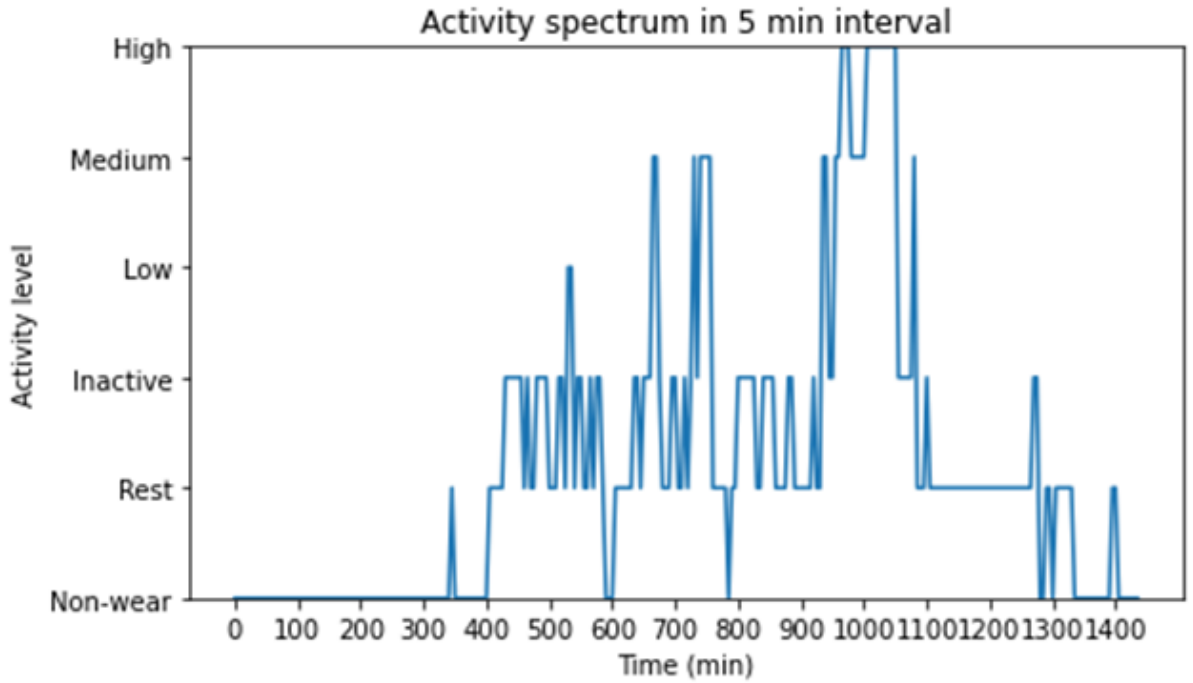
```
#Plot full activity spectra for some random datasamples, using a seed to ensure the samples are always the same
np.random.seed(52)
samples=np.random.randint(0, len(data['activity']), 10)
for i in samples:
        temps=[]
        for j in data['activity'][i]['class_5min']:
        temps.append(j)
        plot(temps, [], [], 'Activity spectrum in 5 min interval', [], [], [], False, 'Time (min)', [np.arange(0, 281, 20)],[np.arange(0, 1401, 100)], [], 'Activity level', [np.arange(0, 6)], [['Non-wear', 'Rest', 'Inactive', 'Low', 'Medium', 'High']], [0, 5], [], [])
```

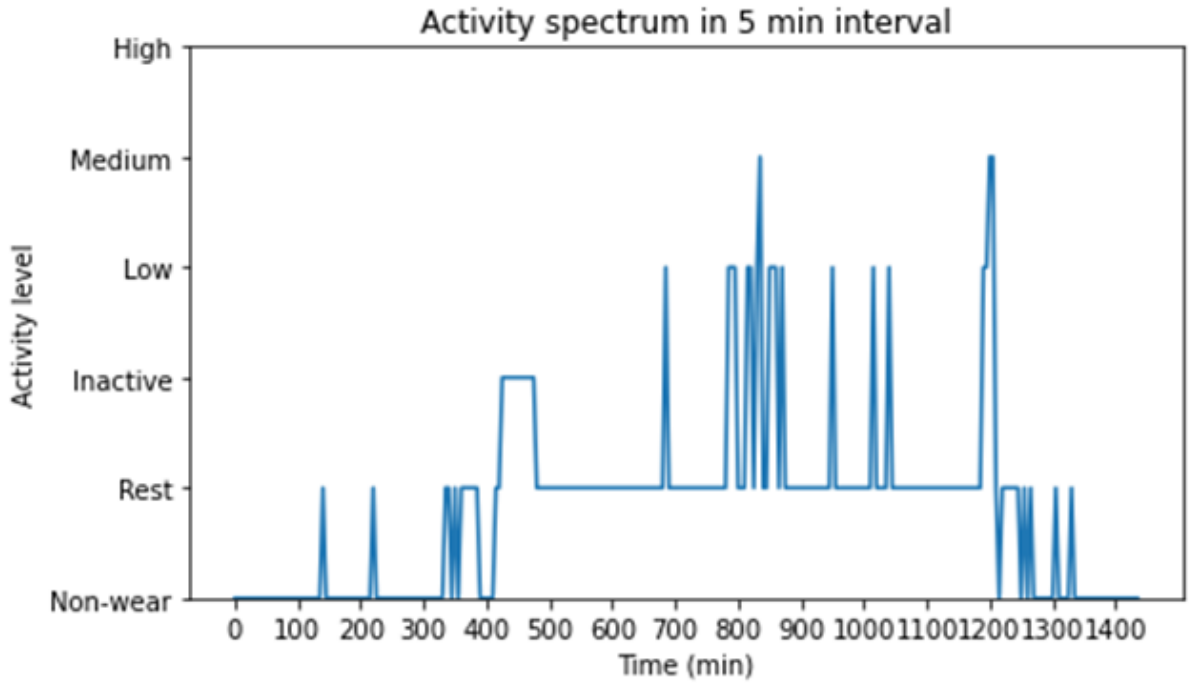


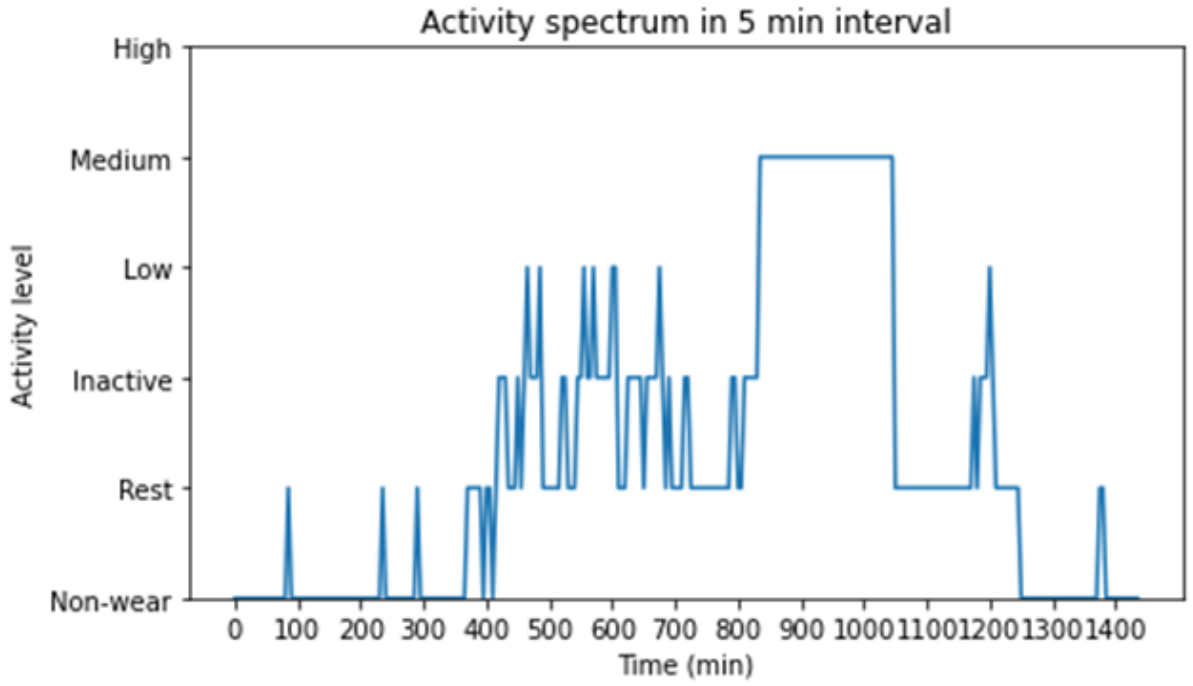


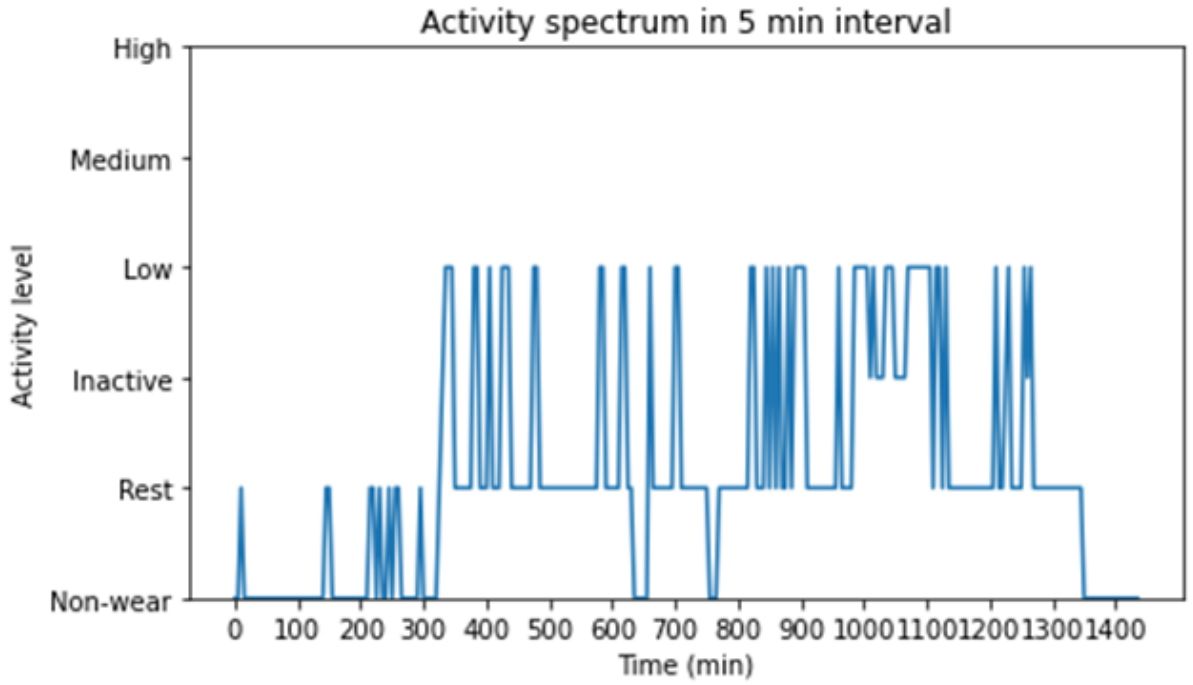

Activity spectrum in 5 min interval
High
Medium
Low
Inactive
Rest
Non-wear
Activity level
0 100 200 300 400 500 600 700 800 900 1000 1100 1200 1300 1400
Time (min)

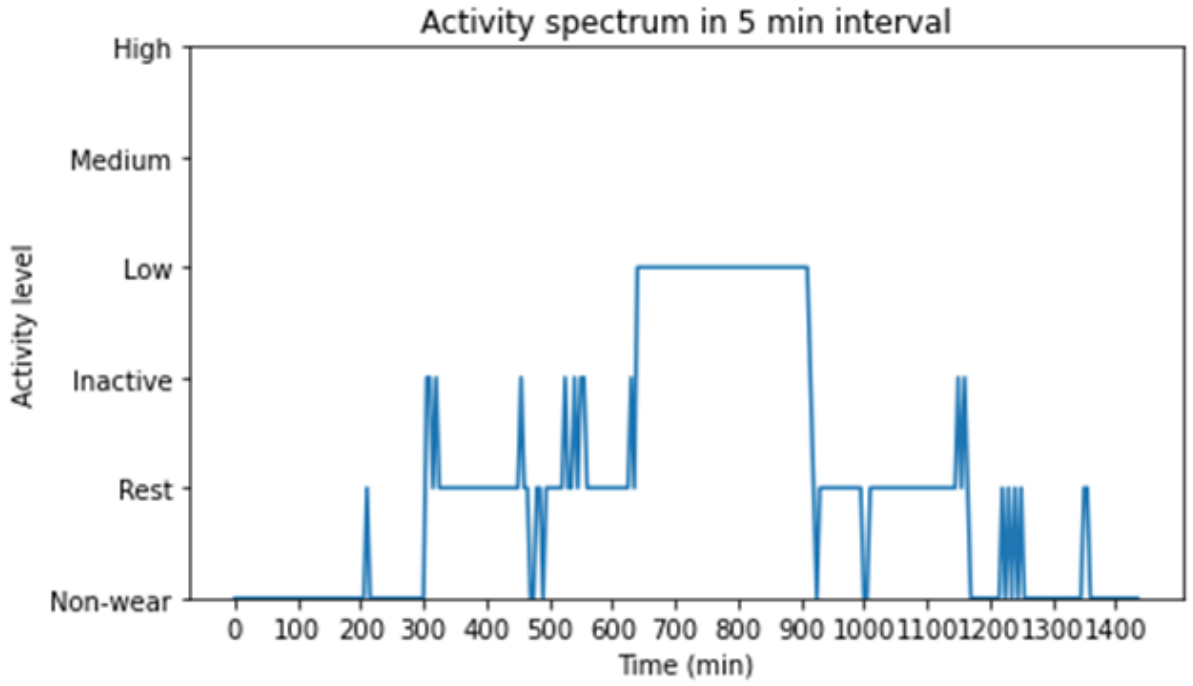

Activity spectrum in 5 min interval
High
Medium
Low
Inactive
Rest
Non-wear
Activity level
0 100 200 300 400 500 600 700 800 900 1000 1100 1200 1300 1400
Time (min)

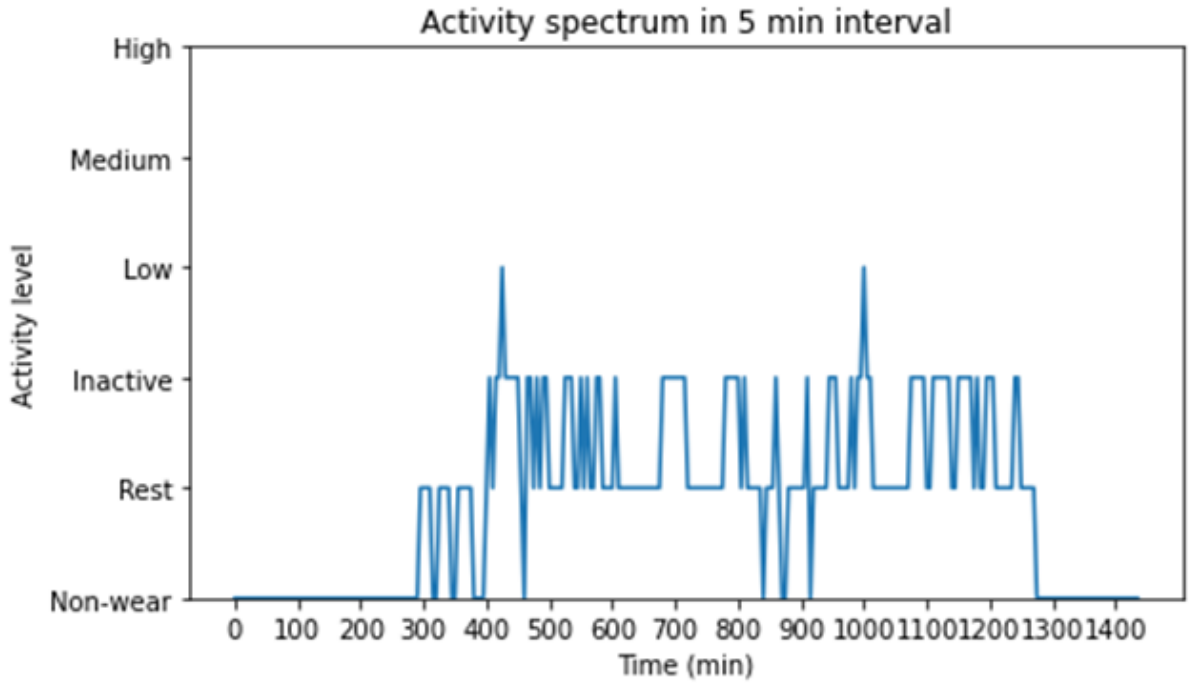

Activity spectrum in 5 min interval
High
Medium
Low
Inactive
Rest
Non-wear
Activity level
0 100 200 300 400 500 600 700 800 900 1000 1100 1200 1300 1400
Time (min)


Activity spectrum in 5 min interval
High
Medium
Low
Inactive
Rest
Non-wear
Activity level
0 100 200 300 400 500 600 700 800 900 1000 1100 1200 1300 1400
Time (min)


Activity spectrum in 5 min interval
High
Medium
Low
Inactive
Rest
Non-wear
Activity level
0 100 200 300 400 500 600 700 800 900 1000 1100 1200 1300 1400
Time (min)


Activity spectrum in 5 min interval
High
Medium
Low
Inactive
Rest
Non-wear
Activity level
0 100 200 300 400 500 600 700 800 900 1000 1100 1200 1300 1400
Time (min)





From this visualisation we can see that for this user the activity levels already reach the medium level and high level more often and also for prolonged periods of time, whereas this was not the case for the first subject, but it is less than the second subject.

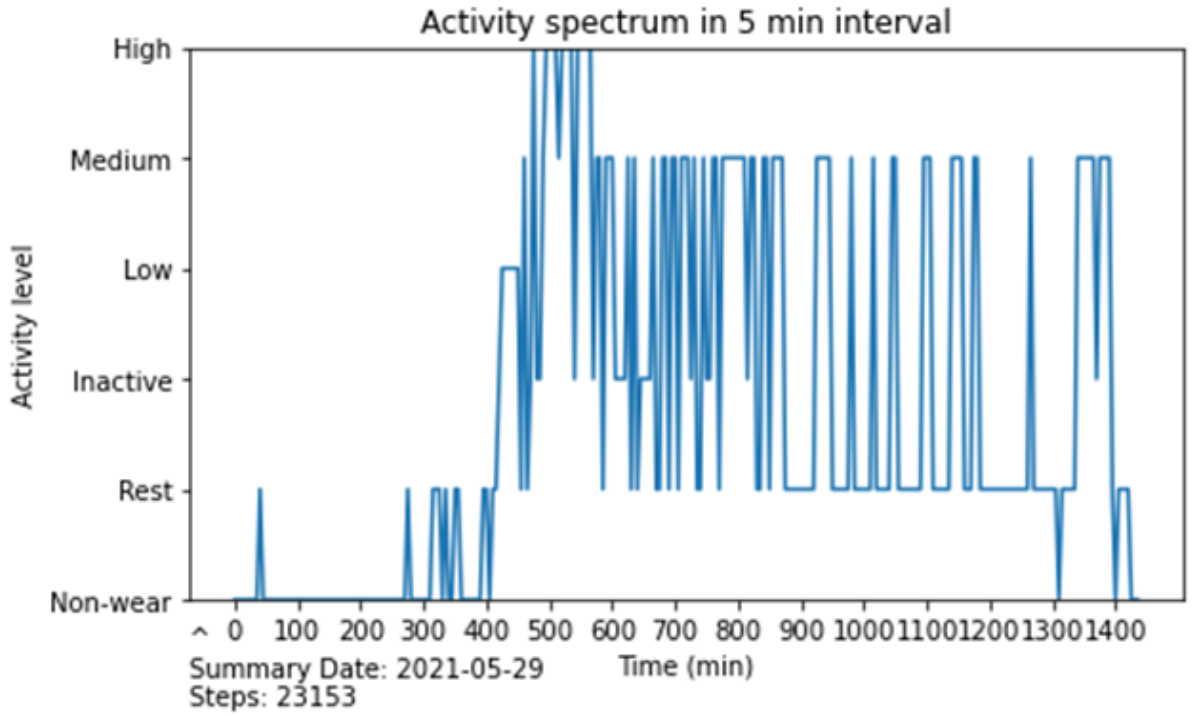
Let's also take a look at the steps like we did before:

In [26]:

```
steps=[]
for i in range(0, len(data['activity'])):
        steps.append(data['activity'][i]['steps'])
```

```
print('The most steps taken during a day is: {} steps.'.format(np.max(steps)))
temps=[]
for j in data['activity'][np.where(steps==np.max(steps))[0][0]]['class_5min']:
        temps.append(j)
plot(temps, [], [], 'Activity spectrum in 5 min interval', [], [], [], False, 'Time (min)',
[np.arange(0, 281, 20)], [np.arange(0, 1401, 100)], [], 'Activity level', [np.arange(0, 6)],
[['Non-wear', 'Rest', 'Inactive', 'Low', 'Medium', 'High']], [0, 5], [], [0.125, -0.02, '^\nSummary
Date: {}\nSteps:
{}'.format(data['activity'][np.where(steps==np.max(steps))[0][0]]['summary_date'],
data['activity'][np.where(steps==np.max(steps))[0][0]]['steps'])])
```

```
The most steps taken during a day is: 23153 steps.
```



Here we can see that even on the day where the user walked the most steps, which is really a lot of steps, the activity is mostly on the medium level and sometimes even on the high level. This corresponds much better to the classification designed by the Oura team. Accordingly, I have to nullify my earlier assumption about the accuracy of the activity data collected by the Oura ring. Possibly there are other factors that influence these activity measurements, such as age, weight, or stamina. This is possibly related to the age dependent limits which Oura did not provide, hence more research is necessary to determine what exactly the age dependent limits are in order to be able to provide the most accurate advice. Since the data for two users was inaccurate though, I believe that the current implementation is still the best for now, until the further research has been done. Due to the extensive documentation I have provided, I am sure it will be easy to adapt the program.

Aligning the activity spectra to the hypnograms

In [27]:

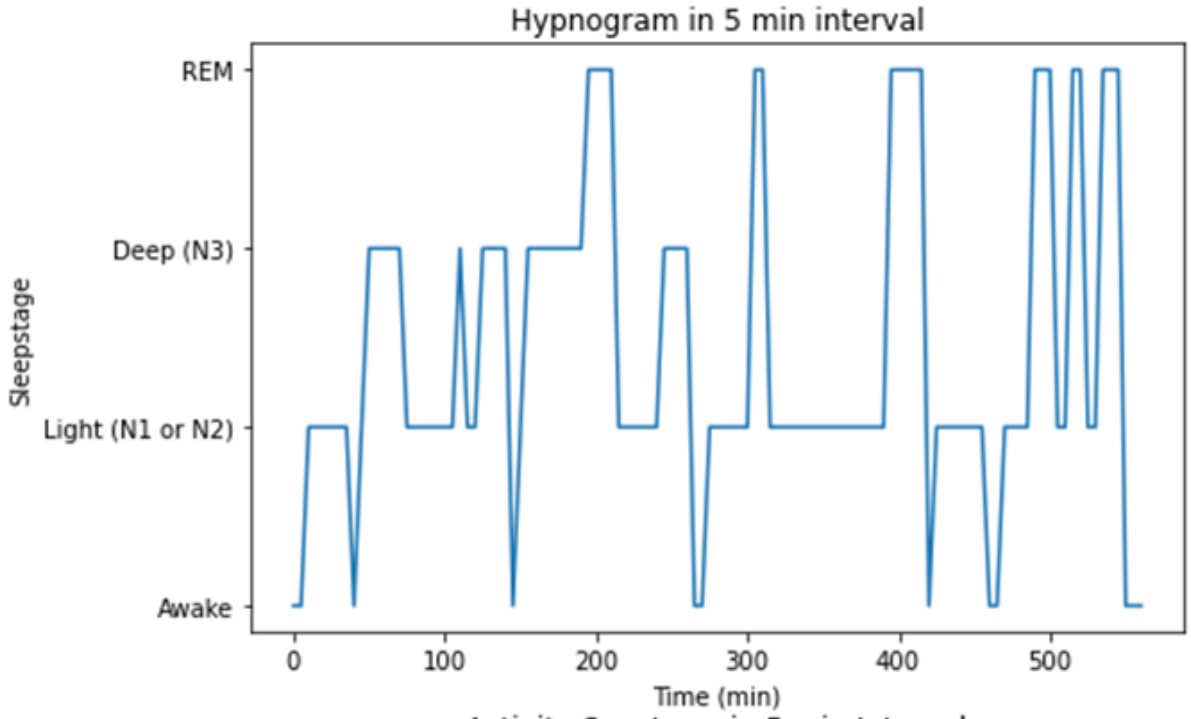
```
#Aligning the activity spectra for the nights during which the user was awake for more than
15 minutes in a row.
```

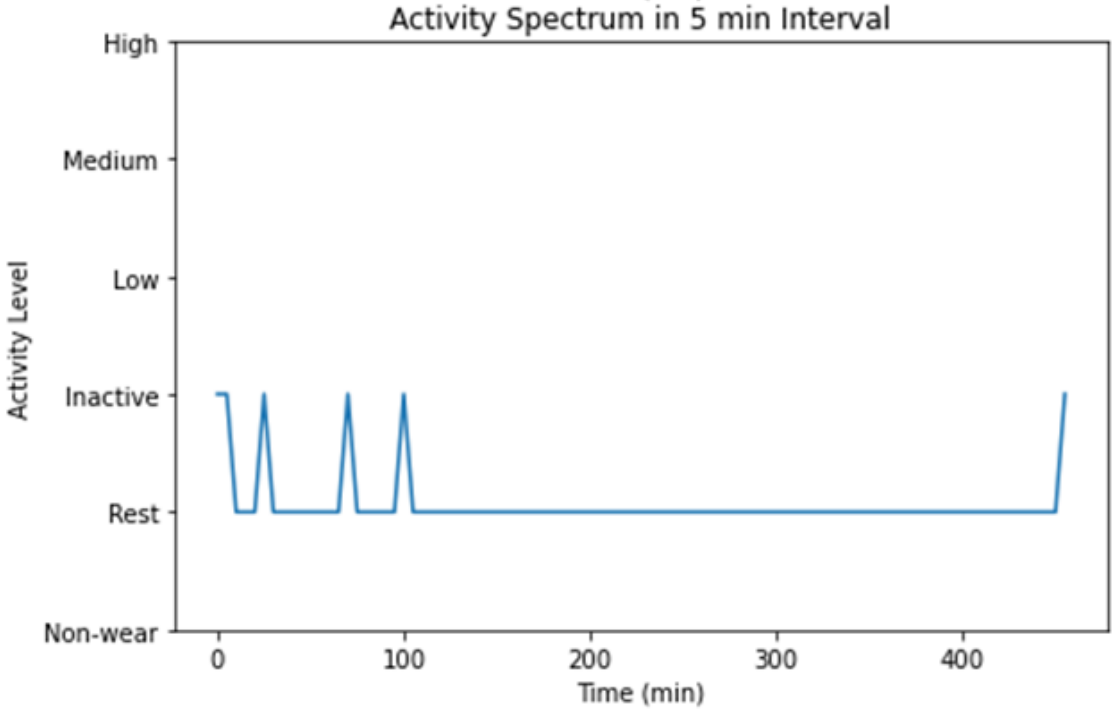
```
#Creating an empty container to store the combined activity class strings for each day
activity_strings=[]
for i in range(0,len(sleeping_times_m)):
        if i>=(len(data['sleep'])-8) and i<(len(data['sleep'])-1-2):
        #Creating an empty container to store the combined end and beginning of the activity
class strings
        int_ind=""

      #Calculate how much minutes from bedtime start to 4 am
        min_till_4am=((24*60)-sleeping_times_m[i]+(4*60))%1440
        #Divide in 5 minute intervals starting from the back
        last_intval_ind=int(np.floor(min_till_4am/5))

      #Matching the date from the sleep data to the date from the activity data
        match_ind=0
        for j in range(0, len(data['activity'])):
        if data['activity'][j]['summary_date']==data['sleep'][i]['summary_date']:
                match_ind=j

      #Check if one entry earlier is indeed one day earlier
        if(date_difference(date_conversion(data['activity'][match_ind-1]['summary_date']),
date_conversion(data['activity'][match_ind]['summary_date']))):

        #Add that part of the interval to the string
        int_ind+=(data['activity'][match_ind-1]['class_5min'][-last_intval_ind-1:-1])

        #Calculate how much minutes from 4 am to bedtime end
        min_from_4am=rising_times_m[i]-(4*60)
        #Divide in 5 minute intervals starting from the back
        first_intval_ind=int(np.floor(min_from_4am/5))

        #Add that part of the interval to the string
        int_ind+=(data['activity'][match_ind]['class_5min'][0:first_intval_ind])

        #Store the string in the container
        activity_strings.append(int_ind)

      else:
        print('Missing date in the activity data before the night to examine.')
```

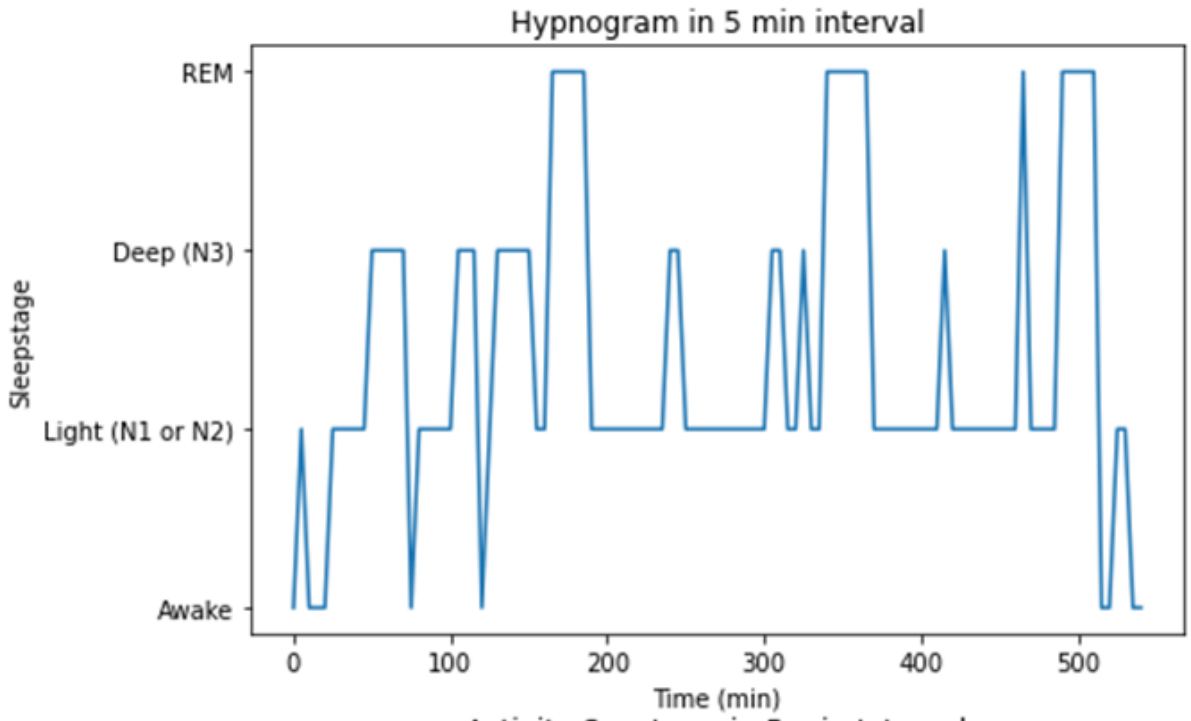
In [28]:

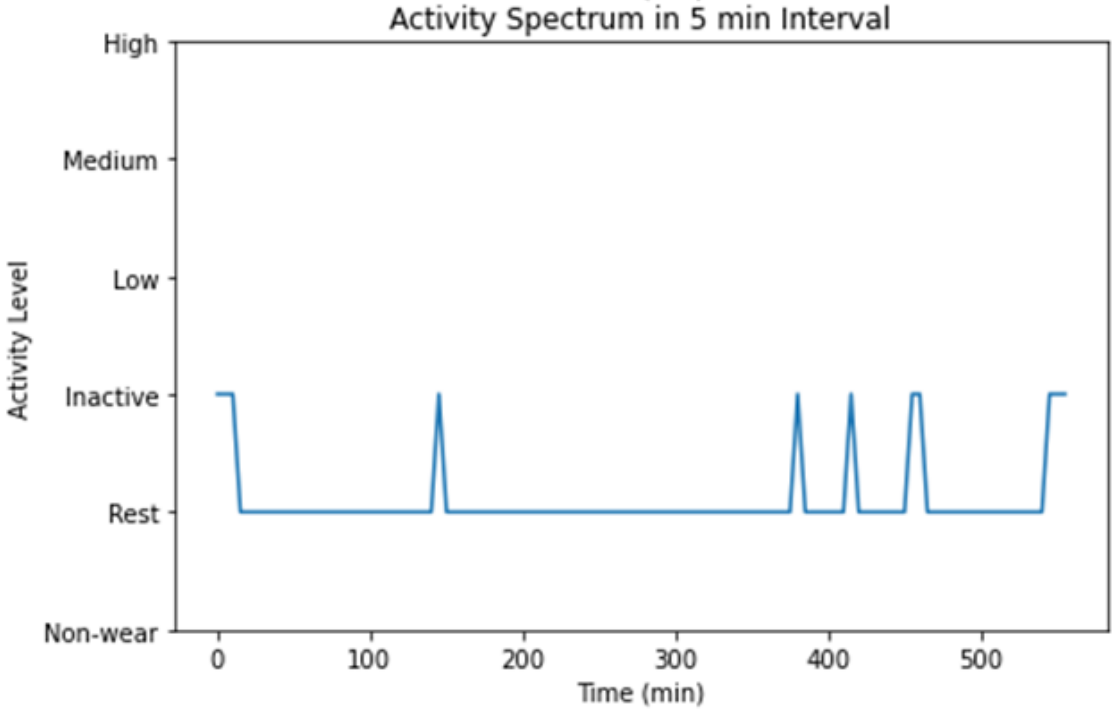
```
#Lets plot the activity spectra for the nights during which the user was awake for more than
15 minutes in a row.
#Since I want to plot the hypnograms and the activity spectra right above each other for easy
comparison I will not use the plotting function created earlier, as it does not allow such
operations
```

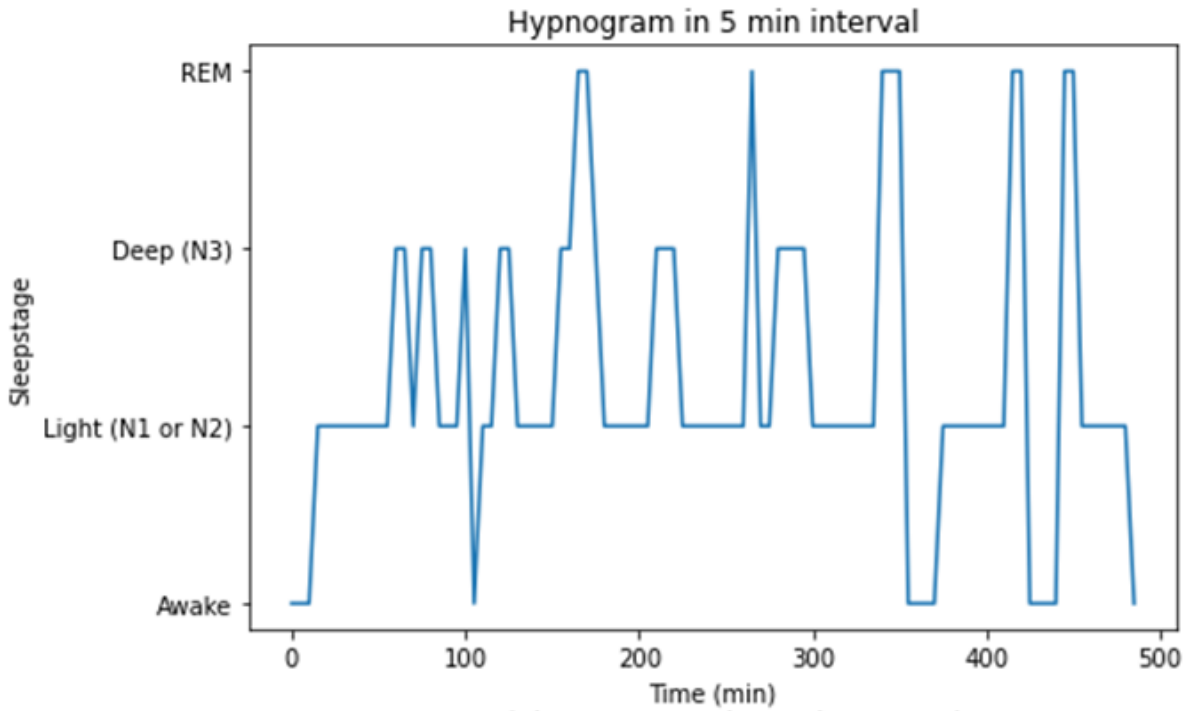
```
counter=-1

#Converting the activity string to a list of integers
for i in activity_strings:
        counter+=1
        as5_split=[]
        for j in range(0,len(i)):
        as5_split.append(int(i[j]))

   #Plotting the hypnograms for easy comparison
        fig, (ax1, ax2)=plt.subplots(2, 1, figsize=(7,10))
        ax1.plot(hgs5[counter])
        ax1.set_title('Hypnogram in 5 min interval')
        ax1.set_ylabel('Sleepstage')
        ax1.yaxis.set_major_locator(matplotlib.ticker.FixedLocator([0, 1, 2, 3]))
        ax1.set_yticklabels(['Awake', 'Light (N1 or N2)', 'Deep (N3)', 'REM'])
        ax1.set_xlabel('Time (min)\n')
        ax1.xaxis.set_major_locator(matplotlib.ticker.FixedLocator([0, 20, 40, 60, 80, 100,
120, 140, 160]))
        ax1.set_xticklabels(['0', '100', '200', '300', '400', '500', '600', '700', '800'])

        #Plotting the activity spectra
        ax2.plot(as5_split)
        ax2.set_title('Activity Spectrum in 5 min Interval')
        ax2.set_ylabel('Activity Level')
        ax2.yaxis.set_major_locator(matplotlib.ticker.FixedLocator([0, 1, 2, 3, 4, 5]))
        ax2.set_yticklabels(['Non-wear', 'Rest', 'Inactive', 'Low', 'Medium', 'High'])
        ax2.set_ylim([0, 5])
        ax2.set_xlabel('Time (min)')
        ax2.xaxis.set_major_locator(matplotlib.ticker.FixedLocator([0, 20, 40, 60, 80, 100,
120, 140, 160]))
        ax2.set_xticklabels(['0', '100', '200', '300', '400', '500', '600', '700', '800'])
        plt.show()
```

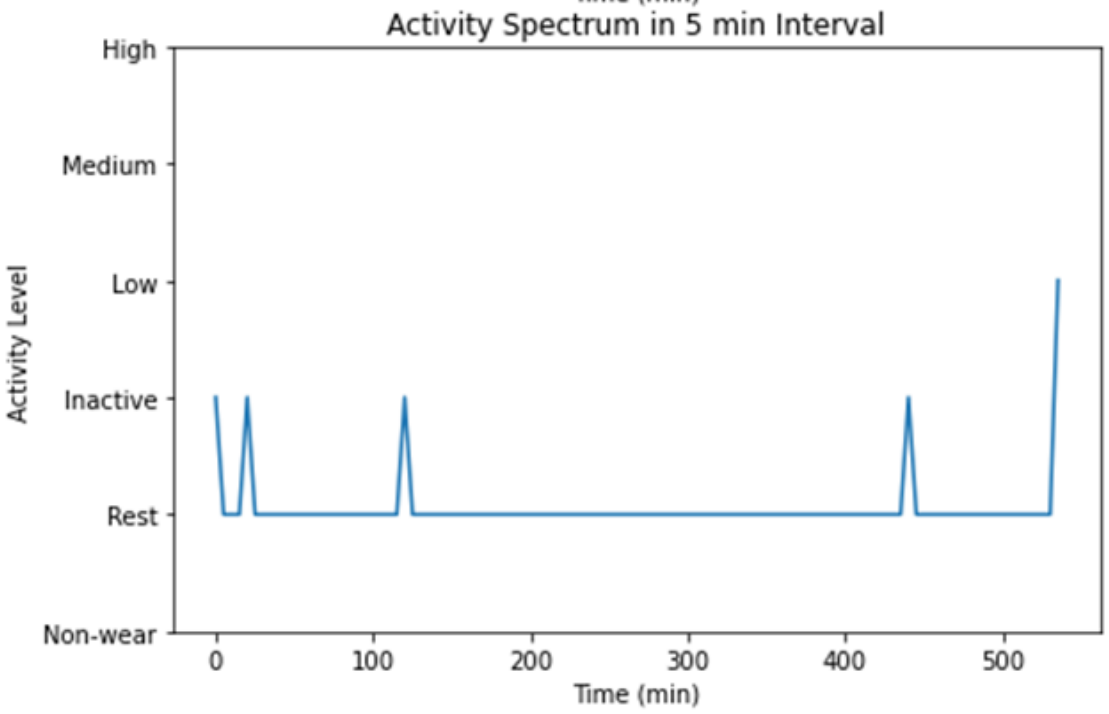

Hypnogram in 5 min interval
REM
Deep (N3)
Light (N1 or N2)
Awake
Sleepstage
0
100
200
300
400
500
Time (min)

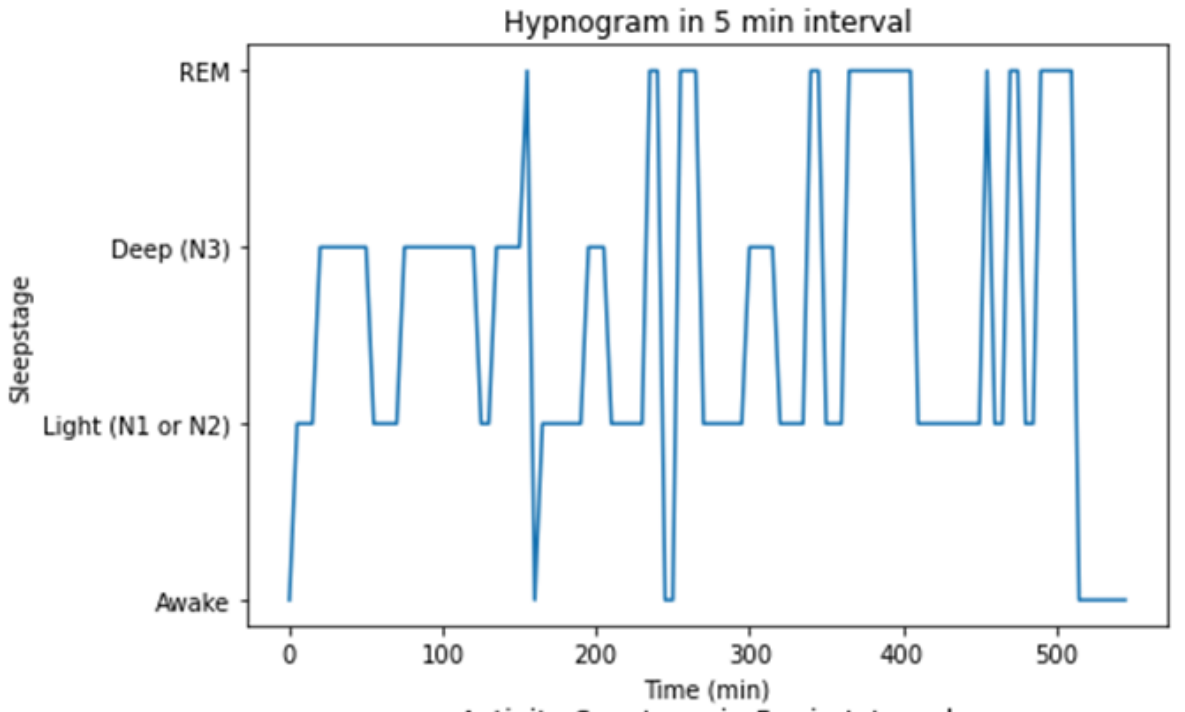

Activity Spectrum in 5 min Interval
High
Medium
Low
Inactive
Rest
Non-wear
Activity Level
0
100
200
300
400
Time (min)

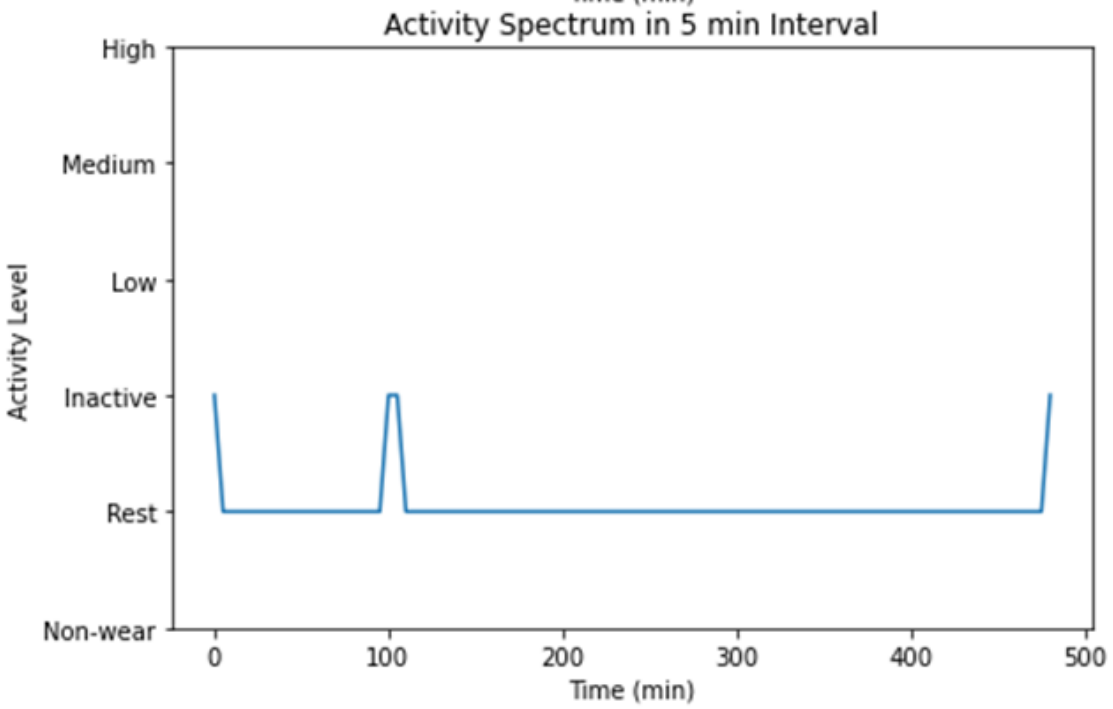

Hypnogram in 5 min interval
REM
Deep (N3)
Light (N1 or N2)
Awake
Sleepstage
0
100
200
300
400
500
Time (min)

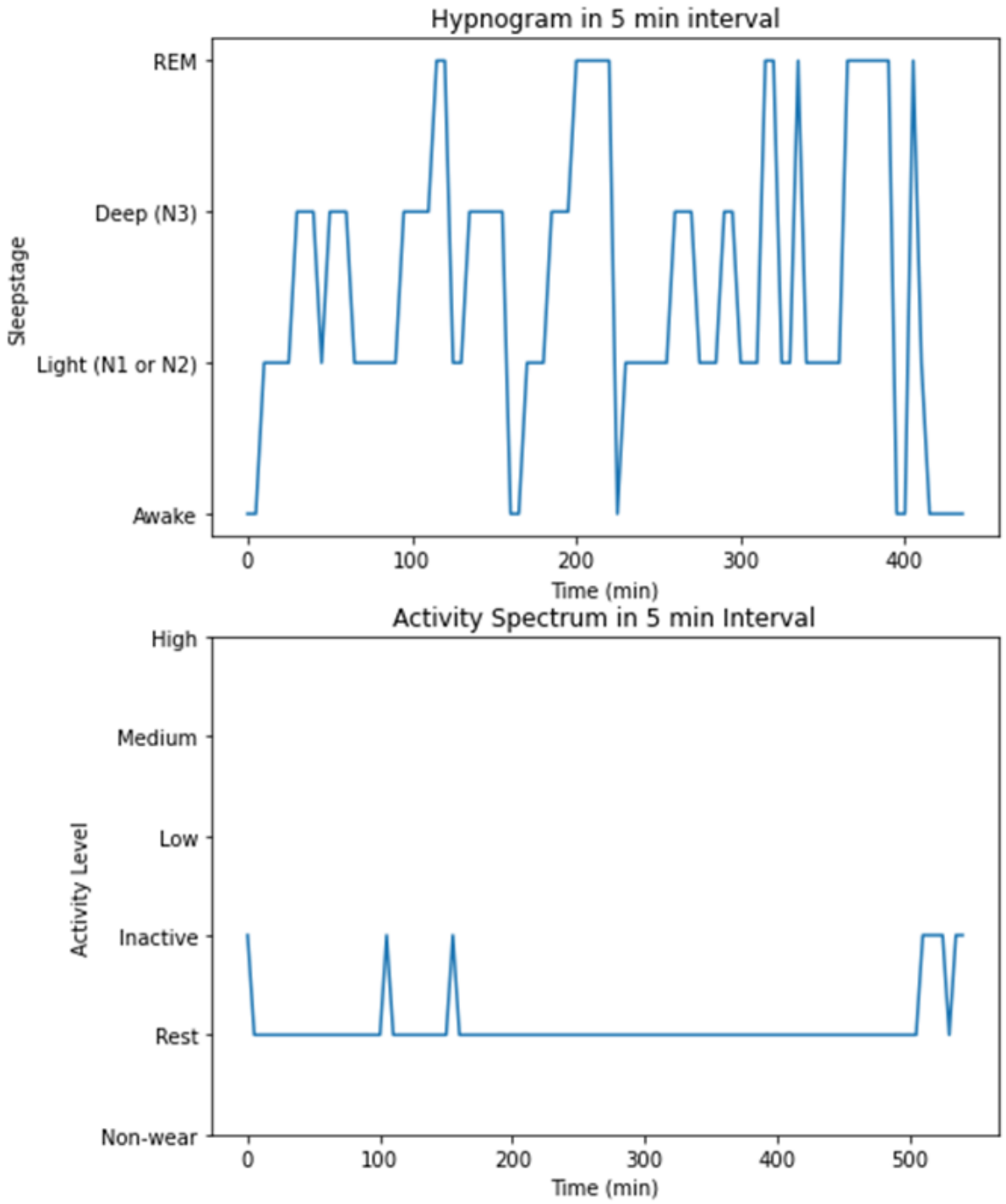

Activity Spectrum in 5 min Interval
High
Medium
Low
Inactive
Rest
Non-wear
Activity Level
0
100
200
300
400
500
Time (min)


Hypnogram in 5 min interval
REM
Deep (N3)
Light (N1 or N2)
Awake
Sleepstage
0
100
200
300
400
500
Time (min)


Activity Spectrum in 5 min Interval
High
Medium
Low
Inactive
Rest
Non-wear
Activity Level
0
100
200
300
400
500
Time (min)


Hypnogram in 5 min interval
REM
Deep (N3)
Light (N1 or N2)
Awake
Sleepstage
0
100
200
300
400
500
Time (min)


Activity Spectrum in 5 min Interval
High
Medium
Low
Inactive
Rest
Non-wear
Activity Level
0
100
200
300
400
500
Time (min)



As can be seen from the hypnograms, sometimes when the user awakens during the night the activity goes or has gone from rest to inactive, but even the low level is never reached. Hence I can safely conclude that the user did not go out of bed during the night, which is good because they did not have to, since they were not awake for a prolonged period of time during the night.

Determining the advice

Based on the fortified assumption that the low activity level corresponds to walking activity indeed, this is how we can determine the advice:

In [29]:

```
def get_out_of_bed():
        #This function is designed to return the number of times the user stayed in bed whils
being awake for a prolonged period of time during the past two weeks.
        try:
      #Creating a counter for the number of times the user stayed in bed whilst being awake
for a prolonged period of time
        times=0
        #Creating an index counter
        ind=-1
        #Converting the sleeping times to minutes in case that has not been done yet
        sleeping_times_m=time_min_conversion('bedtime_start')

      #Checking the hypnograms of the last two weeks
        for i in data['sleep'][-15:-1]:
        #Keeping track of the respective index of the sleep data
        ind+=1
        #Checking for how many consecutive times the users sleep state is awake
        counter=0
        #Checking every value of the hypnogram
        for j in range(0, len(i['hypnogram_5min'])):
                #Incrementing the counter if the sleeping state is 'Awake'
                if int(i['hypnogram_5min'][j])==4:
                counter+=1
                #Resetting the counter if the consecutive streak is broken
                else:
                counter=0
                #If the user was awake for more than 15 consecutive minutes check activity
levels
                if counter>3:
                #Creating a container for the activity string
               activity_string=""

              #Matching the date from the sleep data to the date from the activity data
                match_ind=0

              for k in range(0, len(data['activity'])):
                if data['activity'][k]['summary_date']==data['sleep'][-15+ind]['summary_date']:
                        match_ind=k

                #Check if one entry earlier is indeed one day earlier

if(date_difference(date_conversion(data['activity'][match_ind-1]['summary_date']),
date_conversion(data['activity'][match_ind]['summary_date']))):
                int_ind=""
                #Calculate how much minutes from bedtime start to 4 am
                min_till_4am=((24*60)-sleeping_times_m[-15+ind]+(4*60))%1440
                #Divide in 5 minute intervals starting from the back
                last_intval_ind=int(np.floor(min_till_4am/5))
```

```
                #Add that part of the interval to the string
                int_ind+=(data['activity'][match_ind-1]['class_5min'][-last_intval_ind-1:-1])

                #Calculate how much minutes from 4 am to bedtime end
                min_from_4am=rising_times_m[-15+ind]-(4*60)
                #Divide in 5 minute intervals starting from the back
                first_intval_ind=int(np.floor(min_from_4am/5))

                #Add that part of the interval to the string
                int_ind+=(data['activity'][match_ind]['class_5min'][0:first_intval_ind])

                #Store the string in the container
                activity_string+=int_ind
                else:
                print('Missing date in the activity data before the night to examine.\nThis
means that the activity spectrum cannot be matched to the hypnogram.')

            #Check each value of the activity string within the range of the prolonged
awakeness
                for l in range((j+1)-4,(j+1)):
                #Since the activity string can be slightly longer due to rounding errors, we
need to check if the index does not exceed the length of the activity string
                if not l>=len(activity_string):
                        #If a value in the activity string within the range of prolonged
awakeness reaches the low activity level or higher return true
                        if int(activity_string[l])>=3:
                        times+=1
        return times
        except:
        print('An error occurred.')
```

In [30]:

```
def get_out_of_bed_print():
        try:
        t=get_out_of_bed()
        if not t==0:
        return print('It looks like you stayed in bed whilst being awake for a prolonged period
of time {} times within the last two weeks.\nThis is considered to be bad sleep hygiene, so
next time you lie awake for a prolonged period of time, get out of bed until you are tired
again.'.format(t))
        else:
        return print('It looks like you have not stayed in bed whilst being awake for a
prolonged period of time within the last two weeks.\nThis is considered to be good sleep
hygiene, keep it up!')
        except:
        print('An error occurred.')
```

When should this advice be given?

In [31]:

```
def provide_get_out_of_bed():
        #This function is designed to determine whether the user should be adviced to get
out of bed when unable to sleep.
        #This includes not being able to fall asleep and not being able to continue to sleep,
which includes the final awakening.
        try:
        t=get_out_of_bed()
        if not t==0:
        return True
        else:
        return False
        except:
        print('An error occurred.')
```

Exercise regularly

In [32]:

```
def exercise_regularly():
        #This function is designed to determine the number of days the user exercised
enough, where exercise is classified as either walking more than 7500 steps a day or being
active on the medium level for more than 100 minutes on a certain amount of days per week.
    try:
        #Determining on how many days the user walked 7500 steps or more during the last
week
        more_than_7500_steps=0
        for i in range(-8, -1):
        if data['activity'][i]['steps']>=7500:
                more_than_7500_steps+=1

        #Determining on how many days the user was active on the medium level for more
than 100 minutes, or walked 7500 steps or more during the last week
        if (data['activity'][-1]['score_training_frequency']==100 and
more_than_7500_steps==0) or (data['activity'][-1]['score_training_frequency']>=95 and
data['activity'][-1]['score_training_frequency']<100 and more_than_7500_steps>0) or
(data['activity'][-1]['score_training_frequency']>=70 and
data['activity'][-1]['score_training_frequency']<95 and more_than_7500_steps>1) or
(data['activity'][-1]['score_training_frequency']>=40 and
data['activity'][-1]['score_training_frequency']<70 and more_than_7500_steps>2) or
(more_than_7500_steps>3):
        return 4
        elif (data['activity'][-1]['score_training_frequency']>=95 and
data['activity'][-1]['score_training_frequency']<100 and more_than_7500_steps==0) or
(data['activity'][-1]['score_training_frequency']>=70 and
```

```
data['activity'][-1]['score_training_frequency']<95 and more_than_7500_steps==1) or
(data['activity'][-1]['score_training_frequency']>=40 and
data['activity'][-1]['score_training_frequency']<70 and more_than_7500_steps==2) or
(more_than_7500_steps==3):
        return 3
    elif (data['activity'][-1]['score_training_frequency']>=70 and
data['activity'][-1]['score_training_frequency']<95 and more_than_7500_steps==0) or
(data['activity'][-1]['score_training_frequency']>=40 and
data['activity'][-1]['score_training_frequency']<70 and more_than_7500_steps==1) or
(more_than_7500_steps==2):
        return 2
        elif (data['activity'][-1]['score_training_frequency']>=40 and
data['activity'][-1]['score_training_frequency']<70 and more_than_7500_steps==0) or
(more_than_7500_steps==1):
        return 1
        else:
        return 0
        except:
        print('An error occurred.')
```

In [33]:

```
def exercise_regularly_print():
        #This function is designed to print the advice for exercising regularly
        try:
        if exercise_regularly()==4:
        return print('It looks like you have been exercising enough lately. This is healthy
behavior that is good for your sleep quality. Keep it up!')
        elif exercise_regularly()==3:
        return print('It looks like you have been exercising enough lately, but there is room for
improvement. If you feel like your sleep quality has suffered, it might be better to exercise
more (regularly) or walk more than 7500 steps a day (more regularly). Keep it up!')
        else:
        return print('It looks like you have not been exercising enough lately. If you feel like
your sleep quality has suffered, it might be a good idea to start exercising more (regularly) or
start walking more than 7500 steps a day (more regularly).')
        except:
        print('An error occurred.')
```

In [34]:

```
def provide_exercise_regularly():
        #This function is designed to determine whether the user should be advised to
exercise more regularly or not.
        try:
    if exercise_regularly()<4:
        return True
        else:
```

```
        return False
        except:
        print('An error occurred.')
```

Reduce time in bed to time slept

In [35]:

```
def sleep_restriction_bedtime():
        #This function is designed to determine the sleep restriction bedtime.
        try:
        #Creating a container for the total sleep duration per night of the last two weeks
        total=[]

      #Creating a list of all total sleep durations of the last two weeks
        for i in range(len(data['sleep'])-15, len(data['sleep'])-1):
        total.append(data['sleep'][i]['total'])

      #Taking the average total sleep duration of the last two weeks
        avg_total=int(np.mean(np.divide(total, 60)))

      #Adding half an hour to the average total sleep duration of the past two weeks
        inc_avg_total=avg_total+30

      #Ensuring the increased average total sleep duration of the past two weeks is 5.5 hours
at minimum
        if inc_avg_total<(5.5*60):
        inc_avg_total=(5.5*60)

      #Calculating the ideal bedtime based on the increased average total sleep duration
        sleep_restriction_bedtime=wake_time()-inc_avg_total

      return sleep_restriction_bedtime
        except:
        print('An error occurred.')
```

In [36]:

```
def sleep_restriction_bedtime_print():
        #This function is designed to print the advice for the sleep restriction bedtime.
        try:
        return print('The ideal sleeping time for this user when applying sleep restriction
therapy is at {}.'.format(time.strftime("%H:%M",
time.gmtime(sleep_restriction_bedtime()*60))))
        except:
        print('An error occurred.')
```

In [37]:

```
def provide_srbt():
        #This function is designed to determine whether the user should be advised to follow
sleep restriction or not.
        try:
        #Creating a container for the sleep efficiency per night of the last week
        efficiency=[]

      #Creating a list of the sleep efficiency per night for the last week
        for i in data['sleep'][-8:-1]:
        efficiency.append(i['efficiency'])

      #Determining the average sleep efficiency of last week
        avg_efficiency=np.mean(efficiency)

      if avg_efficiency<80 or avg_efficiency>85:
        return True
          else:
        return False
        except:
        print('An error occurred.')
```

Increment time in bed in steps of 15 to 30 minutes

In [38]:

```
def new_sleep_restriction_bedtime():
        #This function is designed to determine the new sleep restriction bedtime.
        try:
        #Creating a container for the sleep efficiency per night of the last week
        efficiency=[]

      #Creating a list of the sleep efficiency per night for the last week
        for i in data['sleep'][-8:-1]:
        efficiency.append(i['efficiency'])

      #Determining the average sleep efficiency of last week
        avg_efficiency=np.mean(efficiency)

      #Printing the average sleep efficiency of last week
        print('The average sleep efficiency of last week is: {}.'.format(avg_efficiency))

        #Determining the new sleep restriction bedtimes based on the sleep efficiency of last
week
        if avg_efficiency<80:
        new_srbt1=sleep_restriction_bedtime()+15
        new_srbt2=sleep_restriction_bedtime()+30
```

```
        return new_srbt1, new_srbt2
        elif avg_efficiency>=80 and avg_efficiency<85:
        new_srbt1=sleep_restriction_bedtime()
        new_srbt2=sleep_restriction_bedtime()

        return new_srbt1, new_srbt2
        else:
        new_srbt1=sleep_restriction_bedtime()-30
        new_srbt2=sleep_restriction_bedtime()-15

        return new_srbt1, new_srbt2
        except:
        print('An error occurred.')
```

In [39]:

```
def new_sleep_restriction_bedtime_print():
        #This function is designed to print the advice for the new sleep restriction bedtime.
        try:
        nsrbt1, nsrbt2=new_sleep_restriction_bedtime()
        if nsrbt1==sleep_restriction_bedtime() and nsrbt2==sleep_restriction_bedtime():
        print('The user\'s allowed bedtime is good.\nThis is at
{}.'.format(time.strftime("%H:%M", time.gmtime(nsrbt1*60))))
        elif nsrbt1<sleep_restriction_bedtime() and nsrbt2<sleep_restriction_bedtime():
        print('The user\'s allowed bedtime may be increased by 15 to 30 minutes, depending
on how tired they feel during the day.\nThis is between {} and {} on the
clock.'.format(time.strftime("%H:%M", time.gmtime(nsrbt1*60)), time.strftime("%H:%M",
time.gmtime(nsrbt2*60))))
        else:
        return print('The user\'s allowed bedtime should be further restricted by 15 to 30
minutes, depending on how tired they feel.\nThis is between {} and {} on the
clock.'.format(time.strftime("%H:%M", time.gmtime(nsrbt1*60)), time.strftime("%H:%M",
time.gmtime(nsrbt2*60))))
        except:
        print('An error occurred.')
```

In [40]:

```
def provide_nsrbt():
        #This function is designed to determine whether the user should be advised a new
sleep restriction bedtime or not.
        #This function should be ran two weeks after initiating sleep restriction therapy and
every week after, until the new sleep restriction bedtime stays the same for a longer period
of time (in the range of months).
        try:
        efficiency=[]

      #Creating a list of the sleep efficiency per night for the last week
```

```
        for i in data['sleep'][-8:-1]:
        efficiency.append(i['efficiency'])

    #Determining the average sleep efficiency of last week
        avg_efficiency=np.mean(efficiency)

    #Ensuring that the new sleep restriction bedtime cannot be less than 5.5 hours of sleep
        if (avg_efficiency<80 and
abs(wake_time()-sleep_restriction_bedtime())>=((5.5*60)+30)) or avg_efficiency>85:
        return True
        else:
        return False
        except:
        print('An error occurred.')
```

Putting everything together

In [41]:

```
counter, _=gaps()
if counter>0:
        gaps_print()
        safe_date_dif_last_gap_print()
print('')
wake_time_print()
print('')
no_napping_print()
print('')
get_out_of_bed_print()
print('')
exercise_regularly_print()
print('')
sleep_restriction_bedtime_print()
print('')
new_sleep_restriction_bedtime_print()
```

```
There are 3 gaps in the data, comprising a total of 3 days.
The average gap size is approximately 1.0 days and the largest gap size is 1 days.
The missing data does not occur within the last two weeks. It is safe to trust any advices
based on this data.

The advised rising time is: 09:47.
This advice can be rounded to the nearest quarter if wished for.

It looks like you have been taking naps lately. If you feel like your sleep quality has suffered it
might be better not to take naps anymore.
These are the naps: [{'bedtime_end': '2021-05-18T14:53:30+02:00', 'bedtime_start':
'2021-05-18T14:04:30+02:00', 'breath_average': 15.625, 'duration': 2940, 'hr_average':
```

48.83, 'hr_lowest': 48, 'period_id': 3, 'rmssd': 72, 'summary_date': '2021-05-18', 'timezone': 120}, {'bedtime_end': '2021-05-18T15:49:30+02:00', 'bedtime_start': '2021-05-18T15:23:30+02:00', 'breath_average': 15.625, 'duration': 1560, 'period_id': 4, 'summary_date': '2021-05-18', 'timezone': 120}, {'bedtime_end': '2021-05-19T14:46:38+02:00', 'bedtime_start': '2021-05-19T14:07:38+02:00', 'breath_average': 16.25, 'duration': 2340, 'hr_average': 57.6, 'hr_lowest': 56, 'period_id': 1, 'rmssd': 39, 'summary_date': '2021-05-19', 'timezone': 120}, {'bedtime_end': '2021-05-21T16:18:29+02:00', 'bedtime_start': '2021-05-21T15:51:29+02:00', 'breath_average': 16, 'duration': 1620, 'hr_average': 56.5, 'hr_lowest': 53, 'period_id': 1, 'summary_date': '2021-05-21', 'timezone': 120}, {'bedtime_end': '2021-05-21T17:46:29+02:00', 'bedtime_start': '2021-05-21T17:25:29+02:00', 'breath_average': 16, 'duration': 1260, 'hr_average': 52.25, 'hr_lowest': 50, 'period_id': 2, 'rmssd': 63, 'summary_date': '2021-05-21', 'timezone': 120}, {'bedtime_end': '2021-05-23T17:46:32+02:00', 'bedtime_start': '2021-05-23T17:28:32+02:00', 'breath_average': 16, 'duration': 1080, 'hr_average': 55.67, 'hr_lowest': 53, 'period_id': 1, 'rmssd': 68, 'summary_date': '2021-05-23', 'timezone': 120}, {'bedtime_end': '2021-05-24T17:50:56+02:00', 'bedtime_start': '2021-05-24T17:08:56+02:00', 'breath_average': 15.875, 'duration': 2520, 'hr_average': 49, 'hr_lowest': 49, 'period_id': 1, 'summary_date': '2021-05-24', 'timezone': 120}, {'bedtime_end': '2021-05-27T14:30:32+02:00', 'bedtime_start': '2021-05-27T13:56:32+02:00', 'breath_average': 15.625, 'duration': 2040, 'hr_average': 49.33, 'hr_lowest': 47, 'period_id': 1, 'rmssd': 66, 'summary_date': '2021-05-27', 'timezone': 120}, {'bedtime_end': '2021-05-27T15:49:31+02:00', 'bedtime_start': '2021-05-27T15:22:31+02:00', 'breath_average': 15.625, 'duration': 1620, 'hr_average': 52.75, 'hr_lowest': 51, 'period_id': 3, 'rmssd': 58, 'summary_date': '2021-05-27', 'timezone': 120}, {'bedtime_end': '2021-05-27T17:11:32+02:00', 'bedtime_start': '2021-05-27T16:02:32+02:00', 'breath_average': 15.625, 'duration': 4140, 'hr_average': 48.83, 'hr_lowest': 47, 'period_id': 4, 'rmssd': 57, 'summary_date': '2021-05-27', 'timezone': 120}]

It looks like you have not stayed in bed whilst being awake for a prolonged period of time within the last two weeks.
This is considered to be good sleep hygiene, keep it up!

It looks like you have been exercising enough lately. This is healthy behavior that is good for your sleep quality. Keep it up!

The ideal sleeping time for this user when applying sleep restriction therapy is at 02:04.

The average sleep efficiency of last week is: 85.14285714285714.
The user's allowed bedtime may be increased by 15 to 30 minutes, depending on how tired they feel during the day.
This is between 01:34 and 01:49 on the clock.

Automating the selection of relevant micro-interventions

In [42]:

```
def relevant_selection():
        #This function is designed to provide a selection of micro-interventions that are
relevant for the user
        try:
        #Determining if the advices provided are trustworthy or not
        counter, _=gaps()
        if counter>0:
        gaps_print()
        safe_date_dif_last_gap_print()
        print('')

      #Determining which advice should be provided and which not
        #If yes the advice shall be printed
        if provide_wake_time():
        wake_time_print()
        if provide_no_napping():
        no_napping_print()
        if provide_get_out_of_bed():
        get_out_of_bed_print()
        if provide_exercise_regularly():
        exercise_regularly_print()
        if provide_srbt():
        sleep_restriction_bedtime_print()
        if provide_nsrbt():
        new_sleep_restriction_bedtime_print()

      return print('')
   except:
        print('An error occurred')
```

In [43]:

```
#The actual selection
relevant_selection()
```

```
There are 3 gaps in the data, comprising a total of 3 days.
The average gap size is approximately 1.0 days and the largest gap size is 1 days.
The missing data does not occur within the last two weeks. It is safe to trust any advices
based on this data.

It looks like you have been taking naps lately. If you feel like your sleep quality has suffered it
might be better not to take naps anymore.
These are the naps: [{'bedtime_end': '2021-05-18T14:53:30+02:00', 'bedtime_start':
'2021-05-18T14:04:30+02:00', 'breath_average': 15.625, 'duration': 2940, 'hr_average':
48.83, 'hr_lowest': 48, 'period_id': 3, 'rmssd': 72, 'summary_date': '2021-05-18', 'timezone':
120}, {'bedtime_end': '2021-05-18T15:49:30+02:00', 'bedtime_start':
'2021-05-18T15:23:30+02:00', 'breath_average': 15.625, 'duration': 1560, 'period_id': 4,
```

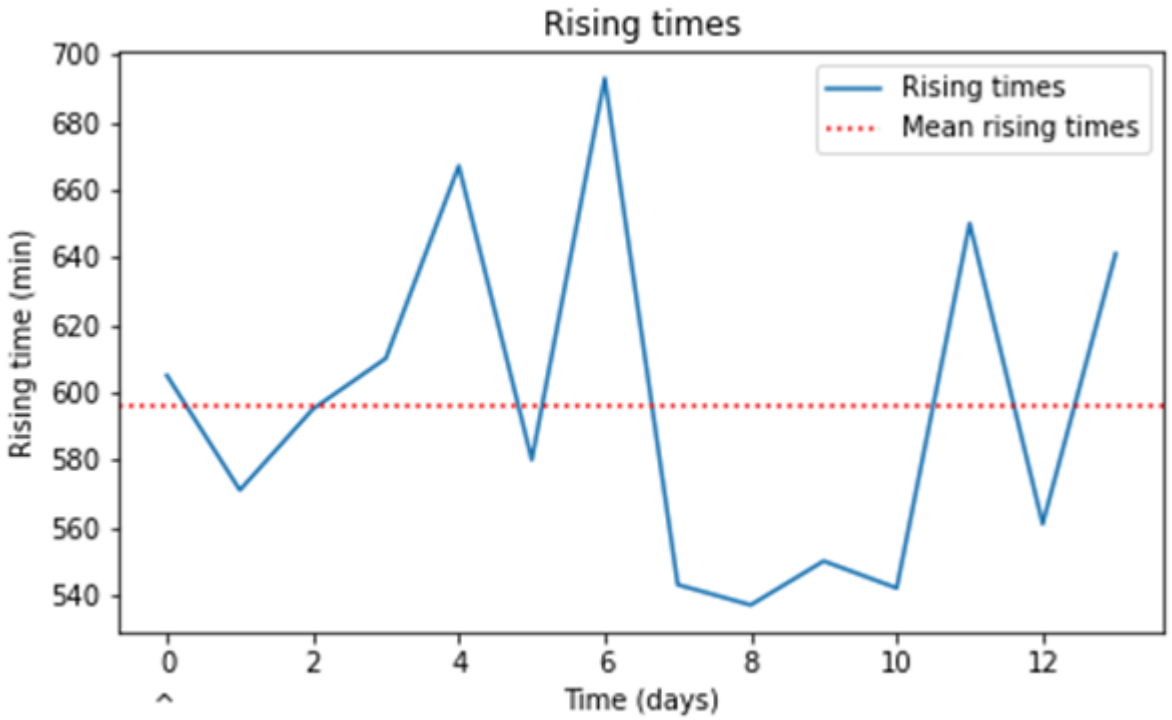
'summary_date': '2021-05-18', 'timezone': 120}, {'bedtime_end':
'2021-05-19T14:46:38+02:00', 'bedtime_start': '2021-05-19T14:07:38+02:00',
'breath_average': 16.25, 'duration': 2340, 'hr_average': 57.6, 'hr_lowest': 56, 'period_id': 1,
'rmssd': 39, 'summary_date': '2021-05-19', 'timezone': 120}, {'bedtime_end':
'2021-05-21T16:18:29+02:00', 'bedtime_start': '2021-05-21T15:51:29+02:00',
'breath_average': 16, 'duration': 1620, 'hr_average': 56.5, 'hr_lowest': 53, 'period_id': 1,
'summary_date': '2021-05-21', 'timezone': 120}, {'bedtime_end':
'2021-05-21T17:46:29+02:00', 'bedtime_start': '2021-05-21T17:25:29+02:00',
'breath_average': 16, 'duration': 1260, 'hr_average': 52.25, 'hr_lowest': 50, 'period_id': 2,
'rmssd': 63, 'summary_date': '2021-05-21', 'timezone': 120}, {'bedtime_end':
'2021-05-23T17:46:32+02:00', 'bedtime_start': '2021-05-23T17:28:32+02:00',
'breath_average': 16, 'duration': 1080, 'hr_average': 55.67, 'hr_lowest': 53, 'period_id': 1,
'rmssd': 68, 'summary_date': '2021-05-23', 'timezone': 120}, {'bedtime_end':
'2021-05-24T17:50:56+02:00', 'bedtime_start': '2021-05-24T17:08:56+02:00',
'breath_average': 15.875, 'duration': 2520, 'hr_average': 49, 'hr_lowest': 49, 'period_id': 1,
'summary_date': '2021-05-24', 'timezone': 120}, {'bedtime_end':
'2021-05-27T14:30:32+02:00', 'bedtime_start': '2021-05-27T13:56:32+02:00',
'breath_average': 15.625, 'duration': 2040, 'hr_average': 49.33, 'hr_lowest': 47, 'period_id': 1,
'rmssd': 66, 'summary_date': '2021-05-27', 'timezone': 120}, {'bedtime_end':
'2021-05-27T15:49:31+02:00', 'bedtime_start': '2021-05-27T15:22:31+02:00',
'breath_average': 15.625, 'duration': 1620, 'hr_average': 52.75, 'hr_lowest': 51, 'period_id': 3,
'rmssd': 58, 'summary_date': '2021-05-27', 'timezone': 120}, {'bedtime_end':
'2021-05-27T17:11:32+02:00', 'bedtime_start': '2021-05-27T16:02:32+02:00',
'breath_average': 15.625, 'duration': 4140, 'hr_average': 48.83, 'hr_lowest': 47, 'period_id': 4,
'rmssd': 57, 'summary_date': '2021-05-27', 'timezone': 120}]
The ideal sleeping time for this user when applying sleep restriction therapy is at 02:04.
The average sleep efficiency of last week is: 85.14285714285714.
The user's allowed bedtime may be increased by 15 to 30 minutes, depending on how tired they feel during the day.
This is between 01:34 and 01:49 on the clock.

Plotting the variables needed for validation

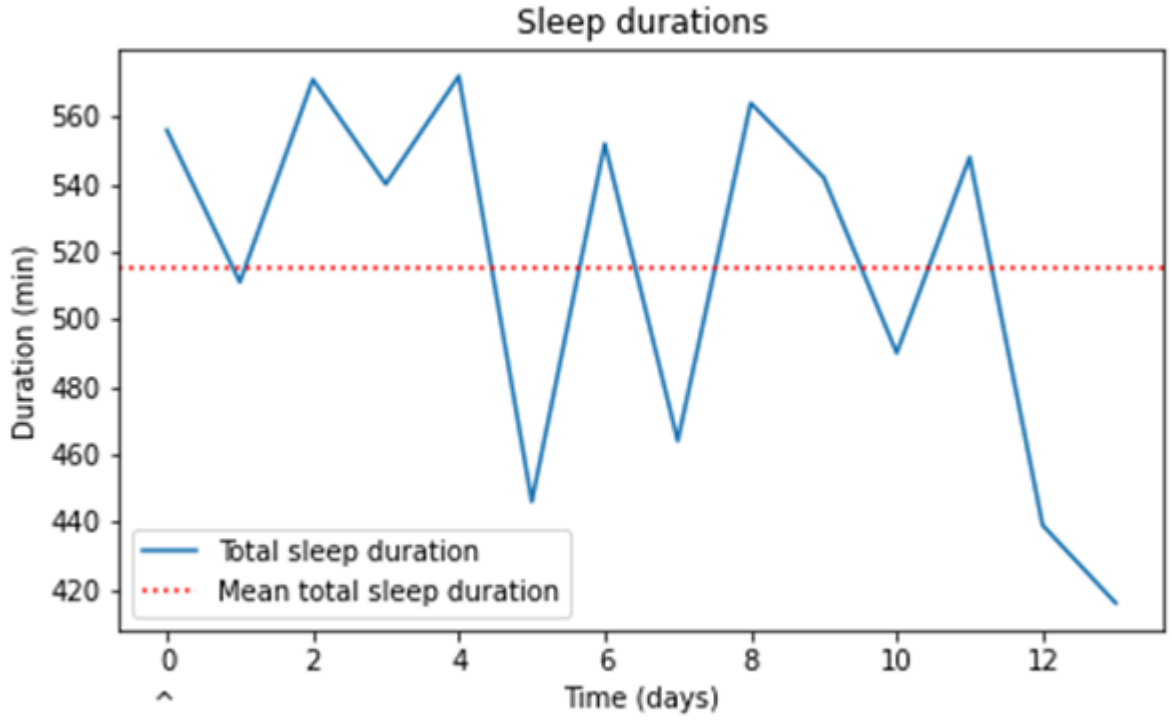
In [44]:

```
#Plotting the rising times in minutes for the entire dataset and plotting the DST switching
dates
plot(rising_times_m[-15:-1], [], [], 'Rising times', [np.mean(rising_times_m[-15:-1])], [], [],
False, 'Time (days)', [], [], [], 'Rising time (min)', [], [], [], ['Rising times', 'Mean rising times'],
[0.15, -0.025, '^\nThe mean rising time is: {}.'.format(time.strftime("%H:%M",
time.gmtime(np.mean(rising_times_m[-15:-1])*60)))])

#Creating a list of sleep durations
durations_m=[]
for i in data['sleep']:
```

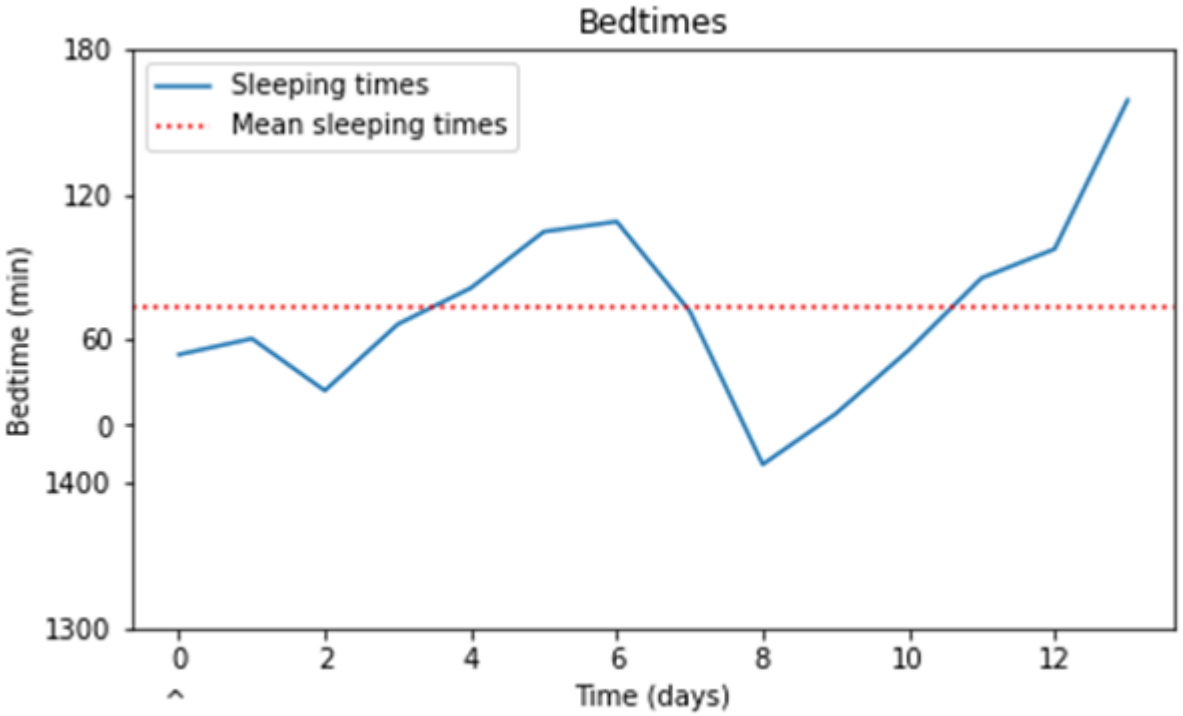
```
        durations_m.append(i['duration']/60)

#Plotting the sleep duration times in minutes for the entire dataset and plotting the DST
switching dates
plot(durations_m[-15:-1], [], [], 'Sleep durations', [np.mean(durations_m[-15:-1])], [], [], False,
'Time (days)', [], [], [], 'Duration (min)', [], [], [], ['Total sleep duration', 'Mean total sleep
duration'], [0.15, -0.025, '^\nThe mean sleep duration is: {}
minutes.'.format(time.strftime("%H:%M", time.gmtime(np.mean(durations_m[-15:-1])*60)))])

#Reshaping the sleeping_times_m data
r_sleeping_times_m=[]
for i in range(0,len(sleeping_times_m)):
        if sleeping_times_m[i]<(4*60):
        r_sleeping_times_m.append(sleeping_times_m[i]+(24*60))
        else:
        r_sleeping_times_m.append(sleeping_times_m[i])

#Plotting the reshaped sleeping times in minutes for the entire dataset and plotting the DST
switching dates
plot(r_sleeping_times_m[-15:-1], [], [], 'Bedtimes', [np.mean(r_sleeping_times_m[-15:-1])], [],
[], True, 'Time (days)', [], [], [], 'Bedtime (min)', [1300, 1400, 1440, 1500, 1600, 1700, 1800],
[1300, 1400, 0, 60, 120, 180, 240], [1300, 1700], ['Sleeping times', 'Mean sleeping times'],
[0.15, -0.025, '^\nThe mean sleeping time is: {}.'.format(time.strftime("%H:%M",
time.gmtime(np.mean(r_sleeping_times_m[-15:-1])*60)))])
```



The mean rising time is: 09:56.



The mean sleep duration is: 08:35 minutes.



The mean sleeping time is: 01:21.

Appendix I

Questions	Subject 1	Application 1	Subject 2	Application 2
Age	48	-	23	-
Gender	M	-	M	-
Wearing Oura ring (week)	Daily	-	Daily	-
Wearing Oura ring (Day)	Day and night	-	Day and night	-
Sleeping time	22:45	23:41	00:00	01:21
Sleep onset latency	20m	-	5-10m	-
Waking time	07:30	07:51	10:00	09:56
Advised waking time	-	07:43	-	09:47
Wake time advised	-	Yes	-	No
Total sleep	7h	8h 10m	8h	8h 35m
Number of naps	Not within the last month	Not within the last month	More than 3 times a week	More than 3 times a week
No napping advised	-	No	-	Yes
Stayed in bed during midnight awakenings	-	-	-	-
Exercising	More than 3 times a week	More than 3 times a week	More than 3 times a week	More than 3 times a week
Exercising advised	-	No	-	No
Sleep restriction bedtime	-	00:19	-	02:04
Adapted sleep restriction bedtime	-	23:49 - 00:04	-	01:24 - 01:39

* - = not applicable

Appendix J

□ □)IÔg□ q□ n(K²□ …µ□ g Vš# 6ƒ* □ù/ 'T9 Õ†J «<M äÚT ©□] rž^ ÷¨i În :Áu Á□| kƒ… 'œ□ Òã™ —í› ' ® –± ;LÀ j-Æ îC× ÈvØ œí ßcú C ÿ þ‰□□□K □ëF%□□ö+□Å¤.□j?2□(3□□'4□}ZG□"□J□¹-□· s□□ð{□Ñ □□ƒ*Š□æpŸ□]´□x½»□…□ Å□ÑÀÆ□ª□Ë□.{Ò□ ÁÕ□Y™Ú□_jÝ□ÔMß□mÈä□Ø- □]ë□□ &□□ ' □O 6□H&E□ÝÓJ□DàZ□,□a□F±d□ `h□¡□t□gã|□pg†□:Æˆ□6£□□v–˜□Š ›□zW¦□□Š®□VÅ²□¼z»□;žÁ□›□Ù□:*á□Ï×ã□„,ï□©>ó□™4□□À□□□>³ □ , □ZÅ□□ 3 □Ëà)□p"C□S¿]□%<_□´bf□.›j□1Ø□Úr□ >w□¥ýƒ□z□‰□ë¥‹□'Æ¤□□Ã²□y~À□.„Ê□—ÚÖ□,ëß□□êå□·Pé□öëí□, ú□□y□□BÇ □Ùû □üX'□Vœ6□/C8□§Q;□oô?□wA□)ÃR□iÊS□Âñf□\Ýt□¶□y□zÉ‹□MZš□š³¡□6Ó□ÃÝ±□ ß¼□:d¾□ □Æ□Ù□Ö□ÉîÞ□F£à□c â□Ü]å□n6î□•□ú□æ±ü□Y° □±‡ □‰ñ□□Áh□□¢0 □´õ#□œð3□□[>□□'X□d□□kóc□7±h□YÅk□Bál□_þs□H£…□‹