Universitetet i Stavanger

**DET TEKNISK-NATURVITENSKAPELIGE FAKULTET**

# BACHELOROPPGAVE

| | |
|---|---|
| Studieprogram/spesialisering:<br>Bachelor i Datateknologi | Vårsemesteret, 2021<br><br>Åpen / Konfidensiell |
| Forfatter: Kristoffer Skjæveland Larsen, Aleksander Odland Fattnes | |
| Fagansvarlig:<br>Veileder(e): Tomasz Wiktorski | |
| Tittel på bachelorpppgaven:<br>Engelsk tittel: Automated data alignment during sports events | |
| Studiepoeng: | |
| Emneord: | Sidetall: 72<br>+ vedlegg/annet: .....<br>Stavanger, 15. may, 2021<br>dato/år |

**Abstract**

We develop an algorithm to automate data alignment by precisely identifying stops in data collected from NEEDED 2018. Our approach make suggestions for a corrected timestamp and create a reports which are used for final verification purposes. These reports also facilitate manual corrections where applicable.

# Table of Contents

# List of Figures

# 1 Introduction

## 1.1 NEEDED 2018

This project is based on data from the NEEDED 2018 study collected during Nordsjørittet. NEEDED is a cooperative project which focuses on health and exercises [22]. Participants wore Garmin FR935 smartwatches for data collection. They were instructed to stop at four specific points during the race, and press a button when they did. These button presses, or laps, were often insufficient to precisely identify the stops. The data set we were given consisted of 59 participants. There were some differences between these participants in term of what data was stored. We shall return to this in later chapters.

## 1.2 Project Description

It is possible to align data manually, but this can be a time consuming task. While smaller sets of data might be suitable for manual alignment, the same can not be said for when the data sets expand; as the sheer number of data increases, the time investment required will increase until it becomes prohibitively high. As a result, manual alignment lacks the scalability required for large data sets. In this project we will therefore develop an algorithm that aligns data automatically. We will also generate reports for verification and fine tuning purposes. Our goal is to minimize the human time requirement for aligning data by shifting humans responsibility from manual alignment to verification and occasional fine tuning where needed.

The smartwatches used in the NEEDED 2018 study, stores data using the FIT protocol. This protocol lacks documentation in some key areas. A side-goal of our project is therefore to investigate FIT as thoroughly as possible. Our hope is that this can serve as a reference for other projects.

The code for our project will be made available at:
*https://github.com/krizzlybjorn/datbac*

# 2 Theory

Throughout this chapter the tools and libraries used in this project will be investigated. The chapter will start off by explaining the FIT protocol, where

most of the focus will be on the different messages which was encountered and their respective fields. This section was focused on for the reason of fully understanding every aspect of data which is used and stored. Combined with the message sequence, it will give an insight in the FIT protocol. The ReportLab and the mathematics behind the used algorithm, will also be explained. Our project was developed using the programming language Python in Jupyter Notebooks, using the pandas library for data analysis/management and matplotlib for creating plots. This will not be further addressed in this chapter, since it's of low importance.

## 2.1 The FIT protocol

The Garmin FR935 used in NEEDED 2018, store the collected data in FIT files. FIT is an acronym for "The Flexible and Interoperable Data Transfer". Following [3], the FIT protocol is designed for storing and sharing data collected from sport, fitness and health devices. This data is broadcast in real time as FIT messages, using low power ANT+ and Bluetooth protocols. FIT messages are stored in a FIT file, which acts as a container, or file structure, for data storage. Individual FIT messages consists of various data fields, and their values, depending on the type of FIT message. FIT is a proprietary protocol, as opposed to open-source. There is a general lack of documentation regarding FIT, especially where data fields in FIT messages are concerned. In this chapter, we will therefore first explain the FIT protocol before investigate the various FIT messages, and their contents, in detail to better understand FIT. While not directly related to our project, we hope that our investigation of FIT messages can be useful as a reference for other purposes.

### 2.1.1 FIT file types

There are 20 different types of FIT files [8], with the three most common being activity, workout and course [4]. The latter of these contains a series of messages that define a course. Such files are primarily used for recreating a course with course builders like garmin connect course creator [5]. A workout file contains instructions for performing a structured activity. As the name implies, such files are used in order to guide a user through their workout [6]. The activity file is the most common type of FIT file. These files are used for collecting data from various sources, such as wearable devices or cycling

computers. Activity files were used exclusively in our project and as such will be the focus in this chapter.

### 2.1.2 Message sequence

According to [7], there are usually one of two patterns the message sequence in the activity file follows. The most common pattern is the summary last sequence. This sequence happens when the messages are written to the file as they occur during the recording of the activity. In this sequence the summary messages are written to the file at the end of the time interval that they represent. The other message sequence is the summary first message pattern. In this pattern, all of the summary messages are grouped together at the beginning of the activity. Both patterns start with the required messages file id, and optional message device info.

In our case the message sequence for each participant was the summary last sequence. The sequence shown in figure 1 represents the message sequence which was similar for all participants. The number in array, represents the number of messages in this position. One to n, means that there were at least one of this type of message but could be followed up by more of the same message type right after. An example of this is device info, where the number of messages will differ by the equipment used. There were different equipment used in the course. According to [1], all participants was equipped with a chest-strap which measured the beat-to-beat HRV data. 40 of the participants was also equipped with power meters. Some participants will therefore not have the power and accumulated power values, while all participants will have the HRV data. Record messages are usually single messages that HRV messages interleave with. So there will almost always be a cycle between record- and HRV messages, when the activity is active. This is however not always the case. There are times where there are sent multiple record messages after each other. There are never sent multiple HRV messages after each other when the activity is active. When there's a button press there will be a lap message followed up by a unknown message, before the last HRV message. The data sequence will then continue from record message. This will create a cycle between record, HRV and unknown 216 messages. When the participants stops the whole event, the event timer stop message gets stored. There will be no more record messages after this.

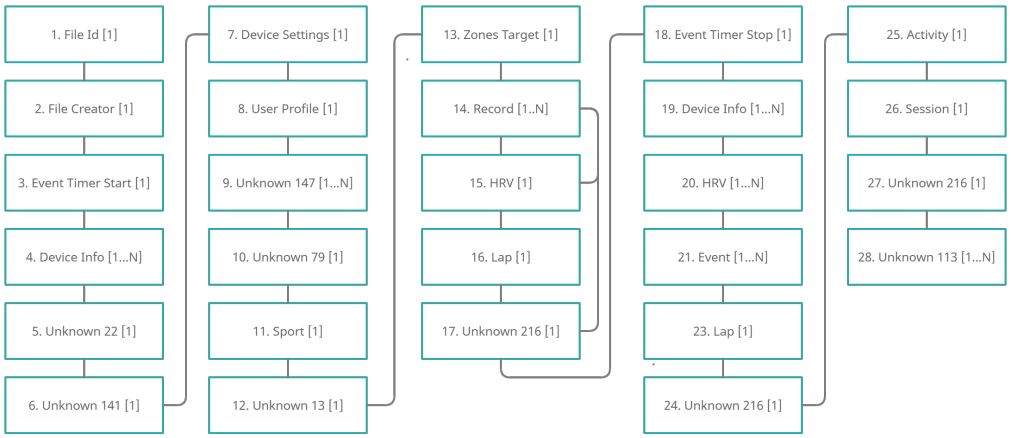


Figure 1: Message sequence

There are some messages which some participants have in their FIT file, and other participants doesn't have. There are also some messages which doesn't occur at a specific position. Both of these occurrences are documented in the table 1. The meaning of these unknown messages and their respective fields are documented in table 16. The meaning of the HR message is documented in the table 11.

| Name | Position and amount |
|---|---|
| Unknown 104 | This message came at an interval of 601 messages, with some exceptions. All the participants which participated in the NEEDED had this message. |
| Unknown 140 | This message came at the end of the activity. Not all participants which participated in the NEEDED had this message, however if it did occur it would only occur once. It would then occur either right after the last event message or right before the second last unknown 22. |
| Unknown 22 | This message would sometimes occur throughout the activity. Not all participants which participated in the NEEDED had this message. It occurred at random intervals, and the amount varied. |
| Unknown 233 | This message would occur throughout the activity. Not all participants which participated in the NEEDED had this message. The amount varied. |

| | |
|---|---|
| HR | This message came after the last unknown 113 message. This message only occurred for some participants. The amount varied. |

Table 1: Unknown sequence

### 2.1.3 Required messages

There are some messages which are required to be included in every activity file. As seen in figure 1, there are many of the same messages occurring at the same position. Many of these messages are not required. They are just common messages in activity file. In this section all the required messages, and their fields which occurred will be explained.

**File Id** All FIT files must contain a single file id message. It's also expected to be the first message in the file. The required fields for this message are type, manufacturer and time created.

| Found fields | Explanation |
|---|---|
| Garmin Product | This field contains information about which garmin product was used. This field depends on whether the manufacturer field is set to garmin or not. If manufacturer field was not set to garmin, this field would be product. |
| Manufacturer | The manufacturer of the device or platform that created the file. In our case it's set to garmin since it's the manufacturer for the garmin devices which was used in NEEDED. |
| Number | This is the file identifier. For all the participants which participated in the NEEDED, this field value for this message type was set to 0. |
| Serial number | The serial number is a unique identifier for the product used. This serial number is identical to the number found in the device info message, table 7. |
| Time created | This is the date and time for when the file was created. |
| Type | The type field identifies the intent of the FIT file. For all the participants which participated in the NEEDED, this field was set to activity for this message type. |

| | |
|---|---|
| Unknown 7 | This field is not known, and is set to none for all participants which participated in the NEEDED for this message type. |

Table 2: File Id position and amount

**Activity** All activity FIT files must also contain a single activity message. Required fields in the activity message are the local timestamp and session count. Since most of the FIT files are recorded in real time, and the session count are unknown until the end, the activity message will often be the last message.

| Found fields | Explanation |
|---|---|
| Event | This field informs that something notable occurred at the point of recording. This field is set to activity for all the participants which participated in the NEEDED, for this message type. This indicates that something notable happened to the activity. |
| Event Group | Related events have the same event group value. This is set to none for all participants which participated in the NEEDED, for this message type. This indicates that there were no other related events. Combining this field with the event field, it informs that it's only the activity something happened to. |
| Event Type | The event type is associated with the event and event group. This is set to stop for all participants which participated in the NEEDED, for this message type. This indicates that only the activity stopped. |
| Local timestamp | This field contains information about the local timestamp. The time offset field in device setting message, table 8, informs the difference in seconds between this field and the timestamp. For all the participants which participated in the NEEDED, this field was stored in central european summer time. |
| Number of sessions | This indicates how many session messages there were. In a multi-sport activity there will be multiple sessions, which will result in this field to increase. |

| | |
|---|---|
| Timestamp | This is the timestamp stored in universal time coordinated. This timestamp will therefore be two hours behind the local timestamp. |
| Total timer time | Total timer time is the start time from the activity started to it ended. It's stored in seconds with milliseconds. |
| Type | This field is explained in file id message, table 2. For all the participants which participated in the NEEDED, this field was set to manual for this message type. |
| Unknown 7 | This field is not known, and is set to none for all the participants which participated in the NEEDED. |

Table 3: Activity position and amount

**Session** There are required at least one session message for each activity. The sessions message is a summary message type. The required fields for session messages are start time, total elapsed time, total timer time, and timestamp. Those are also required fields for all summary messages.

| Found fields | Explanation |
|---|---|
| Average Cadence | This is the average cadence stored in rounds per minute. If the sport is running, the cadence is stored in steps per minute. The cadence value which the average cadence is calculated from, is stored in the record messages. Table 6. |
| Average Cadence Position | This is the average cadence position. |
| Average Combined Pedal Smoothness | This is the combined values for both average right- and left pedal smoothness. Right- and left pedal smoothness is described later in this table. |
| Average Fractional Cadence | This contains information about the average fractional cadence during the session. It's the decimals of the average cadence. The fractional cadence value, which the average fractional cadence is calculated from, is stored in the record messages. Table 6. |
| Average Heart Rate | This is the average heart rate value. The heart rate value, which the average heart rate is calculated from, is stored in the record messages. Table 6. |

| | |
|---|---|
| Average Left Platform Center Offset | This is the average left platform center offset. The platform is the pedal, and it's value describes the offset from the center stored in millimeters. |
| Average Left Pedal Smoothness | The pedal smoothness is calculated by dividing the average power by the peak power. This value is then shown as percentage. The average of this value is then calculated [20]. Pedal smoothness is a measurement of how evenly a rider is applying the force on the pedal for each pedal stroke. The left pedal smoothness value, which the average left pedal smoothness is calculated from, is stored in the record messages. Table 6. |
| Average Left Power Phase | It's the average of the left pedal power produced, stored in degrees. |
| Average Left Power Phase Peak | This is the peak of the pedal power produced on the left, stored in degrees. |
| Average Left Torque Effectiveness | The torque effectiveness is calculated through the pedals, by analyzing the forward and backward torque applied to the crank over each revolution [19]. The average of this value is then calculated for the left torque. It's stored as percent. The left torque effectiveness value, which the average left torque effectiveness is calculated from, is stored in the record messages. Table 6. |
| Average Power | This is the average power. It's stored in watts. The power value, which the average power is calculated from, is stored in the record messages. Table 6. |
| Average Power Position | This is the average power position. |
| Average Right Platform Center Offset | It's the same as average left platform center offset that was mentioned earlier in the table. |
| Average Right Pedal Smoothness | It's the same as average left pedal smoothness that was mentioned earlier in the table. The right pedal smoothness value, which the average right pedal smoothness is calculated from, is stored in the record messages. Table 6. |

| | |
|---|---|
| Average Right Power Phase | It's the same as average left power phase that was mentioned earlier in the table. |
| Average Right Power Phase Peak | It's the same as average left power phase peak that was mentioned earlier in the table. |
| Average Right Torque Effectiveness | It's the same as average left torque effectiveness that was mentioned earlier in the table. The right torque effectiveness value, which the average right torque effectiveness is calculated from, is stored in the record messages. Table 6. |
| Average Speed | This is the average speed of the participant through the course. It's stored as unsigned 16 bit meters per second. The speed value, which the average speed is calculated from, is stored in the record messages. Table 6. |
| Average Stance Time | This seems to be the average stance time, when doing a running activity. This is stored in milliseconds. A stance is the period when the foot is in contact with the ground. |
| Average Stance Time Balance | This seems to be the average stance time balance shown on each foot. It seems to be the same as left right balance, where the sum should be 100 percent. It's stored as percent. Seems to be for running activity. |
| Average Stance Time Percent | This seems to be the percentage of the average stance time, when doing a running activity. Stored as percent. |
| Average Step Length | This seems to be the average step length, found by all the steps taken in a running activity. This is stored as millimeter. |
| Average Stroke Count | Seems to be used for swimming activities. Where this value is the average of strokes. |
| Average Stroke Distance | Seems to be used for swimming activities. Where this value is the average stroke distance. Stored in meters. |
| Average Temperature | This is the average temperature which was recorded throughout the session. The temperature value, which the average temperature is calculated from, is stored in the record messages. Table 6. |
| Average Vertical Oscillation | This is the average amount that the torso moves vertically with each step, while running. Used in running activities [16]. Stored in centimeters. |

| | |
|---|---|
| Average Vertical Ratio | Vertical ratio value is a measurement of vertical oscillation, divided by the stride length, and expressed as a percent. This is the average of that value.[17] Is used for running activity. |
| Enhanced Average Speed | This is the same as average speed for most cases. It's stored as unsigned 32 bit integer. The enhanced speed value, which the enhanced average speed is calculated from, is stored in the record messages. Table 6. |
| Enhanced Max Speed | This is the same as max speed for most cases. It's stored as unsigned 32 bit integer meters per second. |
| Event | This field was also found in the activity message, table 3. Here the event value was set to lap for all participants which participated in the NEEDED for this message type. |
| Event Group | This field was also found in activity message, table 3, and the event group value was set to none for all participants which participated in the NEEDED for this message type. |
| Event Type | This field was also found in activity message, table 3, and the event type value was set to stop for all participants which participated in the NEEDED for this message type. |
| First Lap Index | This is the first lap index value. It's the same value as the first message index in lap message, table 5 |
| Intensity Factor | Intensity factor informs how difficult the course was in relation to the individuals overall fitness. It's calculated by dividing the normalized power by the functional threshold power. This value usually ranges between 0.75 and 1.15, where 1.15 being the most difficult [14]. |
| Left Right Balance | This seems to be the percent of the right and left pedal power. Which should be equal to 100 percent [15]. It's the same value as the left right balance in record messages, table 6 |
| Max Cadence | This is the maximum cadence that happened during the session. It's the same value as the cadence in record messages, table 6. |
| Max Cadence Position | This is the max cadence position. |

| | |
|---|---|
| Max Fractional Cadence | This is the maximum fractional cadence. It's the maximum of the fractional cadence which is found in record messages, table 6. |
| Max Heart Rate | This is the max heart rate value. The heart rate value, which the max heart rate is calculated from, is stored in the record messages. Table 6. |
| Max Power | This is the max power of the participant through the course. It's stored in watts. The power value, which the max power is calculated from, is stored in the record messages. Table 6. |
| Max Power Position | This is the max power position. |
| Max Speed | This is the maximum speed of the participant through the course. It's stored as unsigned 16 bit meters per second. The speed value, which the max speed is calculated from, is stored in the record messages. Table 6. |
| Max Temperature | This is the max temperature. The temperature value, which the max temperature is calculated from, is stored in the record messages. Table 6. |
| Message Index | This is the index value of the message. If there's only one message the index will be one. If there are more it will increment. |
| Northeast Corner Latitude | This is the northeast corner latitude. |
| Northeast Corner Longitude | This is the northeast corner longitude. |
| Normalized Power | Normalized power is a power averaging method. It's as power, stored in watts. Normalized power takes changes in resistance, for example changes in wind, terrain and elevation [12] in considering. The power value, which the normalized power is calculated from, is stored in the record messages. Table 6. |
| Number Active Lengths | Seems to be used for swimming. Where one length is the pool length. |
| Number Of Laps | The number of laps. This can be found through the number of lap messages. 5. |

| | |
|---|---|
| Pool Length | Seems to be used for swimming activities. Where this value is the pool length. |
| Pool Length Unit | This is used for swimming sport. The user of the device can enter the pool size before the activity starts in meters or yards. |
| Sport | This indicates what type of sport this activity is. There are different options to choose. Examples are running, swimming, basketball, soccer and cycling. For all the participants which participated in the needed the sport was set to cycling. This field is also found in sport message, 13. |
| Stand Count | This seems to be the number of times standing still, when doing a running activity. |
| Start Position Latitude | This is the start position latitude recorded when the activity started. The position latitude value, which the start position latitude is calculated from, is stored in the record messages. Table 6. |
| Start Position Longitude: | This is the start position longitude recorded when the activity started. The position longitude value, which the start position longitude is calculated from, is stored in the record messages. Table 6. |
| Start Time | This is the start time of the session. It has the same format with date and time as timestamp. It's the same as the timestamp for the first record message, 6. |
| Sub Sport | This indicates more specific how the sport is executed. Examples are indoor cycling and treadmill. This is set to generic for all participants except one, for the participants which participated in the NEEDED. This field is also found in sport message, 13. |
| Southwest Corner Latitude | This is the southwest corner latitude. |
| Southwest Corner Longitude | This is the southwest corner longitude. |
| Swim Stroke | This is used for swimming sport. A stroke is counted every time the arm wearing the device completes a full cycle. |

| | |
|---|---|
| Threshold Power | This is the threshold power. |
| Timestamp | This is stored the same way as in table 3. |
| Time Standing | This is the timed amount the participant is standing on the pedal. This is based on the measurements of the radial and tangential forces applied to the pedals [13]. It's stored in seconds. |
| Total Anaerobic Training Effect | This value is linked with the total training effect, which is mentioned later in this table. |
| Total Ascent | It's the total ascent of the participant. It's stored in meters. Total ascent and total descent are affected by the environmental factors, like air pressure and weather.[11] |
| Total Calories | This is the total calories which was burned. It's value is stored in kcal. |
| Total Cycles | This is the number of cycles the wheel had throughout this session. |
| Total Descent | It's the total descent of the participant. It's stored in meters. |
| Total Distance | This is the total distance traveled in meters. The distance value, which the total distance is calculated from, is stored in the record messages. Table 6. |
| Total Elapsed Time | It indicates the time from start to finish. It's the same as total timer time in activity, since participants which participated in the NEEDED only had one session. |
| Total Fractional Cycles | This is the total fractional cycles. It's the decimals of the total cycles. |
| Total Timer Time | This is the the time from start to finish. This value will be the same as total elapsed time when there's only one session. |
| Total Training Effect | This value contains information about how the activity effects the users aerobic and anaerobic fitness. It bases on the user profile information, heart rate, duration and intensity of activity. This value will be a number between 0.0 and 5.0, where 0 is low effect. [18] |
| Total Work | This is the calculated work volume, and it depends on the users weight, gender and other personal parameters. Stored in joules. |

| | |
|---|---|
| Training Stress Score | This is a way to measure how much stress is put on the body during the course. It's calculated through normalized power, intensity factor and duration [10]. Which is mentioned earlier. |
| Trigger | This is set to activity end for all participants in session. Trigger field is associated with the event group. |

Table 4: Session field and description

There are a lot of fields, that describes information about the activity. Field unknown 38, looks like is the first lap end position latitude. These are the exact same value. Unknown 39 field which comes right after looks like is the first lap end position longitude. Unknown 110 field is set to Sykling or Bike for all participants which participated in the NEEDED. It seems to be the name field in sport messages, 13.

There are some unknown fields which is hard to figure out, since it's values are either only none for each participant, or only zero for each participant. There are usually four cases found. The first case is that every participants session message contains this field and it's set to zero, which is the case for field unknown 81 and unknown 106 fields. The second case is that every participants session message contains this field and it's set to none, which is the case for unknown 78, unknown 33, unknown 79, unknown 80, unknown 107, unknown 108, unknown 109. The third case which is that not every participants sessions message contains this field, but if it does it's set to zero. This is the case for unknown 151 and unknown 152. And the last case, which is that not every participants session message contains this field, but if it does it's set to none. Which is the case for unknown 157, unknown 158, unknown 153, unknown 154. Unknown 138 is not in the mentioned cases above. Unknown 138 contains a list of two values, and the meaning behind this field could not be found. Unknown 150 field seems to be the min temperature.

**Lap** Lap message represents laps or intervals within the session. Lap message like session message is a summary message type. Therefore the required fields are the same here. An activity file should at least contain one lap message for every session message. These laps can be marked manually or automatically by the device. The sum of the total elapsed time and distance

values for all lap messages should be equal to the correspondent session message. The last lap message will have the field lap trigger set as session end. This message will be stored at the end of the activity, and it's triggered by an event message with the field event type set as stop all.

| Found fields | Explanation |
|---|---|
| Average Cadence | This is explained in session message, table 4. |
| Average Cadence Position | This is explained in session message, table 4. |
| Average Combined Pedal Smoothness | This is explained in session message, table 4. |
| Average Fractional Cadence | This is explained in session message, table 4. |
| Average Heart Rate | This is explained in session message, table 4. |
| Average Left Platform Center Offset | This is explained in session message, table 4. |
| Average Left Pedal Smoothness | This is explained in session message, table 4. |
| Average Left Power Phase | This is explained in session message, table 4. |
| Average Left Power Phase Peak | This is explained in session message, table 4. |
| Average Left Torque Effectiveness | This is explained in session message, table 4. |
| Average Power | This is explained in session message, table 4. |
| Average Power Position | This is explained in session message, table 4. |
| Average Right Platform Center Offset | This is explained in session message, table 4. |
| Average Right Pedal Smoothness | This is explained in session message, table 4. |
| Average Right Power Phase | This is explained in session message, table 4. |

| | |
|---|---|
| Average Right Power Phase Peak | This is explained in session message, table 4. |
| Average Right Torque Effectiveness | This is explained in session message, table 4. |
| Average Speed | This is explained in session message, table 4. |
| Average Stance Time | This is explained in session message, table 4. |
| Average Stance Time Balance | This is explained in session message, table 4. |
| Average Stance Time Percent | This is explained in session message, table 4. |
| Average Step Length | This is explained in session message, table 4. |
| Average Stroke Distance | This is explained in session message, table 4. |
| Average Temperature | This is explained in session message, table 4. |
| Average Vertical Oscillation | This is explained in session message, table 4. |
| Average Vertical Ratio | This is explained in session message, table 4. |
| End Position Latitude: | This is the end position latitude. It's stored in the same format as start position latitude. The position latitude value, which the end position latitude is taken from, is the last record messages. Table 6. |
| End Position Longitude: | This is the end position longitude. It's stored in the same format as start position longitude. The position longitude, which the end position longitude is taken from, is the last record messages. Table 6. |
| Enhanced Average Speed | This is explained in session message, table 4. |
| Enhanced Max Speed | This is explained in session message, table 4. |
| Event | This is explained in session message, table 4. |
| Event Group | This is explained in session message, table 4. |

| | |
|---|---|
| Event Type | This is explained in session message, table 4. |
| First Length Index | Seems to be used for swimming. Where this is the first length index. See number of active lengths field in session message, table 4. |
| Intensity | This is the intensity. |
| Lap Trigger | This is explained in session message, table 4. |
| Left Right Balance | This is explained in session message, table 4. |
| Max Cadence | This is explained in session message, table 4. |
| Max Cadence Position | This is explained in session message, table 4. |
| Max Fractional Cadence | This is explained in session message, table 4. |
| Max Heart Rate | This is explained in session message, table 4. |
| Max Power | This is explained in session message, table 4. |
| Max Power Position | This is explained in session message, table 4. |
| Max Speed | This is explained in session message, table 4. |
| Max Temperature | This is explained in session message, table 4. |
| Message Index | This is explained in session message, table 4. |
| Normalized Power | This is explained in session message, table 4. |
| Number Active Lengths | This is explained in session message, table 4. |
| Number Lengths | Seems to be used for swimming, where this is the number of lengths. |
| Sport | This is explained in session message, table 4. |
| Stand Count | This is explained in session message, table 4. |
| Start Position Latitude | This is explained in session message, table 4. |
| Start Position Longitude: | This is explained in session message, table 4 |
| Start Time | This is explained in session message, table 4. |
| Sub Sport | This is explained in session message, table 4. |
| Swim Stroke | This is explained in session message, table 4. |
| Time Standing | This is explained in session message, table 4 |
| Timestamp | This is explained in session message, table 4. |
| Total Ascent | This is explained in session message, table 4. |

| | |
|---|---|
| Total Calories | This is explained in session message, table 4. |
| Total Cycles | This is explained in session message, table 4 |
| Total Descent | This is explained in session message, table 4. |
| Total Distance | This is explained in session message, table 4 |
| Total Elapsed Time | This is explained in session message, table 4 |
| Total Fat Calories | This is the total fat calories burned. |
| Total Fractional Cycles | This is explained in session message, table 4. |
| Total Timer Time | This is explained in session message, table 4 |
| Total Work | This is explained in session message, table 4 |
| Workout Step Index | This is the index value for the workout step. |

Table 5: Lap field and description

Unknown 27 is the northeast corner latitude, while unknown 28 is the northeast corner longitude. Unknown 29 is the southwest corner latitude, while unknown 30 is the southwest corner longitude. Unknown 124 looks like the minimum temperature. Unknown 70, unknown 72, unknown 73 unknown 90, unknown 96, unknown 97, is set to none for all participants which participated in the NEEDED. Unknown 125, unknown 126, is none for those who have it. The meaning behind these unknowns couldn't be figured out.

**Record** Record contain timestamps usually with a one second difference, even though devices may store data at a lower rate. Timestamp and at least one other value is required for each record message. There are some records which have a larger difference than one second.

| Found fields | Explanation |
|---|---|
| Accumulated Power | This is the power, which have been accumulated. |
| Altitude | This is the altitude. |
| Cadence | This is the cadence. |
| Distance | This is the distance. |

| | |
|---|---|
| Enhanced Altitude | This is the altitude. The meaning of enhanced is described in session message, table 4. |
| Enhanced Speed | This is the speed. The meaning of enhanced is described in session message, 4. |
| Fractional Cadence | This is the fractional cadence, which is the stored decimals from cadence. More information about cadence can be found in table 4. |
| Heart Rate | This is the heart rate. |
| Left Pedal Smoothness | This is the left pedal smoothness. More information about left pedal smoothness can be found in table 4. |
| Left Right Balance | This is the left right balance. More information about left right balance can be found in table 4. |
| Left Torque Effectiveness | This is the left torque effectiveness. More information about left torque effectiveness can be found in table 4. |
| Position Latitude | This is the position latitude. |
| Position Longitude | This is the position longitude. |
| Power | This is the power. |
| Right Pedal Smoothness | This is the right pedal smoothness. More information about right pedal smoothness can be found in table 4. |
| Right Torque Effectiveness | This is the right torque effectiveness. More information about right torque effectiveness can be found in table 4. |
| Speed | This is the speed which is recorded at this record time. |
| Temperature | This is the temperature which is stored at this moment. |
| Timestamp | This is explained in session message, table 4. |

Table 6: Record field and description

There where three different unknown fields which was found in record messages. Unknown 87, seems to either be a number or none. Unknown 88 have two different values, which either was three hundred or one hundred. The meaning behind these two unknowns couldn't be found. Unknown 90 was the last unknown message found in record messages, and it seems to be the performance condition.

### 2.1.4 Optional messages

This is additional messages which some activity files have contained. These messages depends on multiple factors. It depends on features used on a device, whether the user was performing a workout or following a course, or what sensors were in use.

**Device Info** Device info contains information about the device and other hardware accessories or sensors that have been used. The serial number is a ten digit number, which we have chosen not to show. This message seems to contain several data-points about the device begin used.

| Found fields | Explanation |
|---|---|
| Ant Device Number | This is the ant device number. |
| Ant Network | This is the ant network. Examples of networks found for the participants which participated in the NEEDED was ant plus and ant-fs. |
| Ant Transmission Type | This is the ant transmission type. |
| Antplus Device Type | This field contains information about what type the antplus device is. An example found for the participants which participated in the NEEDED was the heart rate. |
| Battery Status | This is the battery status. Example of values found for the participants which participated in the needed was "ok", "good" and "low". |
| Battery Voltage | It's the battery voltage. |
| Cumulative Operating Time | This is the duration of operating time. |
| Device Index | This field is used to identify who created the message. |
| Device Type | If this field is set, there will be no antplus device type field. It contains information same as antplus device type field. |
| Garmin Product | This is used when manufacturer is set as garmin. More information can be found at 2. |
| Hardware Version | This is the hardware version. |
| Manufacturer | This is the manufacturer. More information can be found at 2. |

| | |
|---|---|
| Product | This is used when manufacturer is not set as garmin. |
| Sensor Position | This is the sensor position. |
| Serial Number | This is the serial number. More information can be found at 2. |
| Software Version | This is the software version. |
| Source Type | This is the source type for the device. Examples of values found for the participants which participated in the NEEDED was antplus and local. |
| Timestamp | This is stored the same way as in table 4. |

Table 7: Device Info field and description

There were many unknown fields found. Field 15, seems to be received packets which went through. Field 16 seems to be received packets which didn't go through. Field 24 seems to be the ant id. Field 8, 17 23 is set to none for all participants. Field 31 is a field which not all participants have, and its an integer value or none. All participants have field 13 and 9 and its an integer value or none. All participants also have field 29, which consist of a list of six values. Field 30 is also a field all participants have. This is set to either 1 or none. The meaning of these fields couldn't be figured out.

**Device Settings** There are one device setting message for each participant. It stores a lot of information about the settings which the device have. There are however, more unknown fields than known fields which makes it hard to figure out all the field meanings.

| Found fields | Explanation |
|---|---|
| Active Time Zone | This is the value for what the active time zone is used. |
| Activity Tracker Enabled | This can be set as true or false. The activity tracker will track various features such as daily step counts, distance traveled and calories burned [21]. |
| Auto Activity Detect | The device will auto detect the activity, and store this as a number. |
| Auto Synchronize Minimum Steps | The device will do an auto synchronize after this amount of steps have happened during the activity. |
| Auto Synchronize Minimum Time | The device will do an auto synchronize after this amount of time have happened during the activity. |

| | |
|---|---|
| Backlight Mode | There are various options for setting the backlight of the device screen. Some backlight mode options are auto brightness, smart notifications and off. |
| Display Orientation | Some devices with dual-orientation can be used either vertically or horizontally. |
| Lactate Threshold Autodetect Enabled | Either set to true or false. The lactate treshold is the point where the muscles starts to rapidly fatigue. |
| Mounting Side | Its either set to left or right. It depends on where the device is mounted. |
| Move Alert Enabled | Either set to true or false. This reminds the user to move, when being inactive for a period of time. |
| Number Of Screens | In case of multiple screens this value will differ. |
| Time Mode | This is the time mode which is chosen. For the participants which participated in the NEEDED there were found two values which was either 12 hours or 24 hours. |
| Time Offset | This is the time offset. In the case for the participants which participated in the NEEDED, this value was set to 7200. This if further described in 3. |
| Time Zone Offset | This is the time zone offset. In the case for the participants which participated in the NEEDED, this value was set to 0. |
| UTC Offset | This is the UTC offset. In the case for the participants which participated in the NEEDED, this value was set to 0. |

Table 8: Device Settings field and description

Unknown field 144 seems to store information about whether or not true up is enabled. There are a lot of enum unknown fields, which is hard to figure the meaning behind. These unknown fields are 3, 9, 10, 11, 22, 26, 28, 34, 35, 38, 41, 42, 43, 44, 45, 48, 52, 53, 54, 63,64,65, 66, 67, 68, 69, 75, 81, 82, 83, 84, 85, 87, 101, 104, 107, 108, 109, 110, 111, 112, 126, 127, 128, 133 and 138. Some other fields which is not enum, and couldn't find the unknown meaning behind were field 8, 49, 13, 14, 15, 97, 124 and 125. Field 8 was set to none for all participants. Field 49 was either set to none or 1 for each participant. Field 13 was set to eight for all participants. Field 14 was set

to either one hundred, twenty or seventy for all participants. Field 15 was set to fifty for all participants. Field 97 was set to either one or three for all participants. Field 124 was set to either twenty or sixty for all participants. Field 125 was either eight or sixty for all participants.

**Event** Event messages are used to indicate that something notable occurred, during the recording of an activity. It can in example indicate the start and stop of the timer or if the user went away from a course they were following. These are just two examples. The only unknown field which we noticed for some participants was the unknown field 15. This field was set to none for those participant who had this field.

| Found fields | Explanation |
|---|---|
| Data | The data field is a dynamic field. Which means the event field depends on the value of this field. |
| Event | This is described in the activity message, table **??**. |
| Event Group | This is described in the activity message, table **??**. |
| Timer Trigger | This indicates what will happen to the event. |
| Timestamp | This is stored the same way as in table 4. |

Table 9: Event field and description

**File Creator** This message contains the software version and hardware version. If this message exists there's only one message. Comes at the beginning of the file.

| Found fields | Explanation |
|---|---|
| Hardware Version | This is the hardware version. |
| Software Version | This is the software version. |

Table 10: File Creator field and description

There's also a unknown field which is unknown 2. Unknown 2 is the only unknown field which is found in this message for the participants which participated in the NEEDED. Not all participants have this unknown field, but for those who have it it's set to none. The meaning of this field is not known.

**HR** The HR messages contain compressed heart rate samples. HR messages are commonly used by heart rate monitors, like an arm band or chest strap, that use a store and forward data recording method. The data will be stored in the heart rate monitor and later sent to the Garmin device.

There are two hr messages which is stored. One of these hr messages looks like a summary hr message. The event timestamp in the summary message seems to be all the event timestamp values found in the other hr messages.

| Found fields | Explanation |
|---|---|
| Event Timestamp | This seems to be the timestamp of the heart rate, shown in seconds. There will be many event timestamp fields, and it will increment by the last event timestamp. |
| Event Timestamp 12 | Could not find the meaning behind this field. |
| Filtered Beats Per Minute | Could not find the meaning behind this field. |
| Fractional Timestamp | This is the decimal values of the event timestamp. |
| Timestamp | This is stored the same way as in table 4. |

Table 11: HR field and description

The only unknown fields which is in the HR messages, are the unknown 251. This field is set to a list where the first value is zero and the second is undefined. Not all messages have this field.

**HRV** Time is the time in seconds between heartbeats. Messages are meant to be sent every second, hence there is no timestamp and the possibility for five values. It's recorded as an array of RR intervals. They are interleaved with record and event messages in chronological order.

| Found fields | Explanation |
|---|---|
| Time | Time in seconds between heartbeats. |

Table 12: HRV field and description

There are no unknown fields in this message. The only field which can

appear in this message is the time field.

**sport** Contains information about the sport, subsport and name. This message comes at the start of the file.

| Found fields | Explanation |
|---|---|
| Name | This seems to be the name of the sport, the user has given. |
| Sport | Tells which type of sport this. |
| Subsport | The subsport will define further how the sport was done. |

Table 13: Sport field and description

The unknown fields 5, 6, 11 and 13 is an enum which the meaning behind couldn't be figured out. The fields 4, 10 and 12 is unknown. Field 12 is set to none for all messages. Field 4 is set to sixty-one for all messages. Field 10 is a list containing four values. These fields couldn't be figured out either.

**User Profile** This message contains information about the user at the time of recording the activity. There should only be one user profile message for an activity. This message can occur at any time in the file.

| Found fields | Explanation |
|---|---|
| Activity Class | This number is between 0 and 100. It contains a measurement about how often, and how for how long the user trains in a given week. This value is then used when calculating the calories burned during an activity. |
| Distance Setting | This contains information about which measurements is used. |
| Elevation Setting | This contains information about which measurements is used. |
| Gender | This is information about whether the user is is female or male. |
| Height | This is the height of the user. It value depends on the value of the height setting field. |
| Height Setting | This contains information about which measurements is used. |
| Heart Rate Setting | This contains information about the heart rate setting. This can be beats per minute, max or reserve. |

| | |
|---|---|
| Language | This contains information about which language the user have chosen on the device. |
| Position Setting | This contains information about which measurements is used. This is for latitude and longitude. |
| Resting Heart Rate | This contains information about the resting heart rate, which is then used for heart rate zones calculation. |
| Sleep Time | This contains information about the sleep time duration which was set. |
| Speed Setting | This contains information of which measurements is used. |
| Temperature Setting | This contains information of which measurements is used. |
| User Running Step Length | This contains information about the step length of the user when running. |
| User Walking Step Length | This contains information about the step length of the user when walking. |
| Wake Time | This field contains information about the wake time duration which was set. |
| Weight | This contains information about the users weight. This field depends on the value of the weight setting. |
| Weight Setting | This contains information of which measurements is used. |

Table 14: User Profile field and description

The unknown field 37 seems to be functional threshold speed. Field 24 seems to be the birth year. Field 33 varies between 800 to 1600, seems to be VO2Max related. Field 41 seems to be the last lactate threshold update in date time. Couldn't find the meaning behind the unknown fields 34 ,35 ,36, 38 and 42. Field 34, 35, 42 is a number and all participants messages have these fields. Field 36 is either set to zero or twenty, and all messages have this field. Field 38 is set to none for all messages. Field 43 is an enum.

**Zones Target** There should only be stored maximum one zone target message for each activity file. This message can occur at any time in the file. Some sport zone target values are calculated according to user parameters. Some parameters which are used are the maximum or threshold heart rate or power values.

The zones target are used in training zones, and it gives the user a set

intensity at which they should be working to during an activity. This will make the user be able to push their limits during intervals. There are five training zones, which will increase in difficulty. These zones can be set by garmin or the user manually.

| Found fields | Explanation |
|---|---|
| Functional Treshold Power | This is the highest power level the user can maintain for an hour without growing fatigued. It uses the heart rate and power data to calculate. |
| Heart Rate Calculation Type | This stores information about how the heart rate should be calculated in the heart rate zone. There are three different values which can be chosen. It's the percent of the max heart rate, percent heart rate resting and custom. |
| Max Heart Rate | This is calculated through subtracting the age from 220 for males and 226 for females. The max heart rate can also be manually entered. |
| Power Calculation Type | This stores information about how the power should be calculated in the power zone. There are two different values which can be chose. It's the percent of the functional threshold power or custom. |
| Threshold Heart Rate | The threshold heart rate is calculated through the zone percentage of the max heart rate. |

Table 15: Zones Target field and description

The unknown field 254 seems to be the message index. The unknown fields 9, 10, 11 and 12 is an enumerator, and the meaning behind these fields couldn't be found.

**unknown** There are both unknown messages and unknown fields in messages. They are identified the name unknown followed up by a number. The reason for unknowns messages and fields are because they are undocumented. This can be because of new features which has not been released yet. Since it's undocumented the user doesn't know what this stored data is, which makes it hard for the user to know what data is collected about them. The unknown messages which we have found, and the unknown fields correspondent to these unknown messages are shown in the table.

| Unknown name | Explanation |
|---|---|
| unknown 104 | As previously mentioned this message seems to come at an interval of 601 messages, with some exceptions. This message contains information about the battery. There were not found information about unknown field 1, which is set to none for all participants which participated in the NEEDED. The unknown field 253 is the classic timestamp. The unknown field 0 contains information about the voltage. The unknown field 2 is the battery percent. The unknown field 3 seems to be the current milliampere. |
| unknown 113 | This message always comes after the activity ended. It contains the personal records. The unknown field 0 is the longest distance. The unknown field 1 is the sport. The unknown field 2 is the distance. The unknown field 3 is the duration, however if longest distance is 1 then the unknown field 3 is distance. The unknown field 4 is start time. The unknown field 5 is an enumerator for whether it's a new record or not. |
| unknown 13 | Comes after the sport message and before the zones target message. This message have a lot of unknown fields. |
| unknown 140 | This message comes somewhere at the end of the activity. Many of the fields seems to be physiological metrics. Unknown fields which the meaning behind couldn't be figured out was field 2,3,5 and 6. The unknown field 0 is the minimum heart rate. The unknown field 1 is the maximum heart rate. The unknown field 4 is the aerobic training effect. The unknown field 7 is the MET max. which is metabolic equivalent of task. |
| unknown 141 | This message comes at the start of the activity. The unknown field 0 is an enumerator for whether the data is cached or not. The unknown field 1 is interval start time. The unknown field 2 is the interval end time. Couldn't find the meaning behind the unknown field 3,4 and 5. The unknown Field 253 is the classic timestamp. |

| | |
|---|---|
| unknown 147 | This message comes at the start of the activity. It's the sensor settings. There are a lot of unknown fields which is not listed here. The unknown field 0 is the ant id. The unknown field 2 is the name. The unknown field 11 is the calibration factor. The unknown field 21 is the wheel size in millimeters. The unknown field 254 is the message index. |
| unknown 216 | This message will come throughout the whole activity. For all participants there is one of these before session and one after session at end of activity. This message contains only unknowns fields, and seems to contain heart rate zones information. The fields which the meaning behind is not found is 5,9,14 and 15. The unknown field 253 is the classic timestamp. The unknown field 0 is the type, where it's set to 19 for laps and 18 for sessions. The unknown field 1 is the lap index. The unknown field 2 is an array of unsigned 32 bit integers and appears to be the total beats spent in each hr zone. The unknown field 6 is an array of unsigned 8 bit integers and it's the pre-calculated HR bpm zones. The unknown field 10 is the HR method used to make the zones, Where one is the maxhr method, two is the heartrate reserve method and three is the lactate threshold hr. The unknown field 11 is the pre-calculated max HR. The unknown field 12 is the user minimum HR selected. The unknown field 13 is the autodetected lactate threshold HR |
| unknown 22 | This message comes throughout the activity. It seems to be which device is used as metering source. The unknown field 0 is speed. The unknown field 1 is distance. The unknown field 2 is cadence. The unknown field 3 is elevation. The unknown field 4 is heart rate. The unknown field 5 is the mode. The unknown field 6 is the power. The unknown field 14 is the calories. The unknown field 253 is the classic timestamp. |

| | |
|---|---|
| unknown 79 | This message comes at the start of the activity. Only one message. This message contains user data. There are many unknown fields which is not mentioned here. The unknown field 253 is classic timestamp. The unknown field 0 is the MET max, calculated by VO2max divided by 3,5. The unknown field 1 is the age. The unknown field 2 is the height. The unknown field 3 is the weight. The unknown field 4 is the gender. The unknown field 5 is the activity class. The unknown field 6 is the max hr. The unknown field 8 is the recovery time in hours. The unknown field 10 is the average resting heart rate. The unknown field 11 is the running lactate threshold heart rate. The unknown field 12 is the functional threshold power. The unknown field 13 is the functional threshold speed. |
| unknown 233 | Some participants have a lot of this message throughout their activity, others have none of this message and some have few of this message. This message contains only one field, which is field 2. This contains a list of 4 values. The meaning behind this unknown field couldn't be found. |

Table 16: Unknown messages

## 2.2 ReportLab

We use ReportLab in order to generate a report for each participant. ReportLab is an open-source library used for creating PDF documents. According to [23], the library implements three layers;

- A graphics canvas API that 'draws' PDF pages
- A charts and widgets library for creating reusable data graphics.
- A page layout engine - PLATYPUS ("Page Layout and TYPography Using Scripts") - which builds documents from elements such as headlines, paragraphs, fonts, tables and vector graphics.

We will explain the first and last layers by exploring the general steps of generating a PDF document from start to finish. The second layer will be ignored, as we do not use ReportLab to create graphical elements.

In order to generate a report, we must start by initializing a *document*. To differentiate this from the output, the PDF document, we will refer to the latter as the *report*. The document acts as the container for our report. It contains rules which define the appearance of the report, such as page size, as well as the *canvases*. A canvas is a defined are on which one can add elements, such as images, text or tables. We add elements by specifying exact coordinates on the canvas. The terminology for adding an element to the canvas, is to *draw* the element. This can be thought of as a metaphor for a physical canvas and the act of drawing on said canvas. If we were to draw outside the boundaries of a physical canvas, we would be waving a pencil in the air and our drawing would not be visible. Similarly, if we draw on the digital canvas with coordinates outside the boundaries of the canvas, the drawing will not be visible. We must therefore choose coordinates carefully to suit our purpose. When we are done drawing, we save the document, which gives us the report. This layer of the library is useful for creating static reports. By specifying exact coordinates, we are free to draw elements wherever we want on the canvas. This approach offers us a high level of precision, but it lacks flexibility. If different reports use different elements as a basis, we need to specify new coordinates for every report. If we attempt to use the same coordinates, we might end up with overlapping elements. To overcome this limitation, we must use the PLATYPUS layout engine.

The PLATYPUS engine create reports using *flowables*. A flowable is an abstraction of a drawable object. Instead of drawing the object directly on the canvas, we append them to a list, which is then used to build the document. When we build the document, PLATYPUS automatically draws all flowables in the list in order. PLATYPUS also supports the usage of certain functions within the list. These functions allow for passing instructions to PLATYPUS, such as forcing a page break or switching between page layouts. The latter is requires defining a *page template* and attaching it to the document container. A page template is a set of specifications which define the layout of a page. A page template consists of a unique ID, a page size and a *frame*. Frames are areas of a page on which PLATYPUS can place flowables. In the same manner that flowables are an abstration of drawable object, frames can be considered as an abstraction of a canvas.

## 2.3 Chauvenet's criterion

Our algorithm will use the average distance of every lap as part of the alignment process. For a variety of reasons, which we will discuss in a later chapter, we want to exclude certain participants from our calculations. One possible method for rejecting data, is using Chauvenet's criterion. From [2], we find that Chauvenet's criterion is a test based on finding a probability band centered on the mean of a normal distribution. If a data point, or measurement, is outside the probability band, the measurement will be considered an outlier and can be removed from the set. In this chapter we will explain the mathematics behind Chauvenet's criterion by following and expanding upon [2, pages 165-170].

Consider $N$ measurements $x_1, ..., x_N$ which follow a normal distribution. One of of these measurements, $x_{sus}$, is very different from the others and we consider removing it from our set of measurements. Following the approach for Chauvenet's criterion, we start by finding the mean and standard deviation of our measurements.

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$$
$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_1 - \bar{x})^2}$$

We can now calculate the number of standard deviations $x_{sus}$ deviates from the mean $\bar{x}$.

$$t_{sus} = \frac{|x_{sus} - \bar{x}|}{\sigma}$$

Next, we need to find the expected number of measurements that would deviate from the mean by $t_{sus}$ given a total of $N$ measurements.

$$\begin{aligned} n &= N \times Prob(\text{outside } t_{sus}\sigma) \\ &= N \times (100\% - Prob(\text{within } t_{sus}\sigma)) \end{aligned}$$

Due to our assumption that the measurements follow a normal distribution, we can calculate the probability of a measurement being within the constraint with the following integral [2, page 136]

$$Prob(\text{within } t\sigma) = \frac{1}{\sqrt{2\pi}} \int_{-t}^{t} e^{-z^2/2}\, dz$$

We now have every formula needed to calculate $n$. Chauvenet's criterion states that a measurement $x_{sus}$ can be rejected if $n < \frac{1}{2}$. After rejecting a measurement, we must recalculate $\bar{x}$ and $\sigma$. We might find that there is now another $x_{sus}$ that seems to be outlier. According to [2], we must not apply the criterion a second time. Instead, we must modify our test to allow for rejection of multiple measurements in a single application of Chauvenet's criterion. This is done by calculating the number of standard deviations a measurement deviates from the mean, $t$, for every measurement.

$$t_i = \frac{|x_i - \bar{x}|}{\sigma}$$

These $t_i$ variables must then be sorted so that they are ordered from lowest to highest. We can then calculate the probability $n_i$ corresponding to each measurement. Starting with the lowest value of $t$, we test if this measurement can be rejected. If $n_i < a$, the measurement can be rejected together with every measurement with a greater $t$ value. The criterion $a$ depends on the number of measurements that will be rejected. In our example with a single rejection, we used $a = \frac{1}{2}$. When rejecting multiple values we can use $a = j\frac{1}{2}$, where $j$ is the number of measurements that would be rejected if the criterion is met.

Rejecting data is controversial, as is Chauvenet's criterion. Taylor writes "One reason many scientists are uncomfortable with Chauvenet's criterion is that the choice of one-half as the boundary of rejection (in the condition that $n < \frac{1}{2}$) is arbitrary" [2, page 169]. Expanding the test to include multiple measurements is therefore more controversial; "The objections to Chauvenet's criterion are even greater if two or more measurements are suspect, but the test *can* be extended to this situation [...]" [2, page 170]. So as to not misrepresent our source, it must be noted that the source only expand the test explicitly to account for two measurements. The source only implies the general relation in passing when stating the boundary "that is,

*two times* one-half" page 168. While our source states that the test *can* be expanded, it must be noted that Taylor only expands the test explicitly for rejecting exactly 2 measurements. We will discuss and justify our usage of Chauvenet's criterion in subsecuent chapters.

# 3 Implementation

In this chapter we will walk through our algorithm and approach for aligning data. We begin by examining how we import FIT messages from FIT files and store information in dataframes. Next we look at the alignment process and explain how we account for various scenarios. Finally, we explore how we generate PDF reports using the PLATYPUS engine from reportlab.

## 3.1 Import

In this chapter, we will explain how we store data from FIT files in pandas dataframes. Our implementation is only slightly modified from code which we received from our supervisor.

There are two types of FIT messages we are interested in, record messages and lap messages. Record messages contain data the data collected from our participant that is of interest. This includes timestamp, power, speed, distance, heart rate and so on. A full description of the contents of record messages are available in the table 6. The lap messages appear in the fit file when the participant presses the lap button. The only data of interest for our purposes, is the timestamp as this tells us the exact moment a participant pressed the button. We want to create a dataframe for each participant which contains data from record messages and the timestamp when the participant pressed the lap button. Furthermore, we want to use timestamp as the index for the dataframe. Naturally, this means that we cannot have two rows with identical timestamps. As a consequence, we cannot store all lap and record messages, as that would lead to duplicate timestamps. In order to circumvent this problem, we introduce a column in our dataframe which we shall call lap. We also initialize a global variable which we shall call lap. When we parse the FIT file, lap messages will increment the global lap variable, while record messages are stored in the dataframe alongside the variable. We can demonstrate this with pseudocode.

For our purposes, the only data we need from lap messages are the timestamps as this represents the moment a participant pressed the lap button. By defining a global variable to represent the current lap, and incrementing the variable when we parse a lap message, we can store this variable when the next record message is parsed. As illustrated by pseudocode below, lap messages update the variable, while record messages store data in the dataframe. From figure 1, we see that lap messages occur after record messages. We also know that record messages occur every second. This means that when a lap message is parsed, and the global variable is incremented, the variable will not be stored before the subsequent record message. All laps will therefore be stored exactly one second after the participants actually pressed the button.

```
FUNCTION:
global variable lap = 0
for every message:
    if message.name = record:
        Store message AND lap in Dataframe
    endif
    if message.name = lap:
        lap += 1
    endif
endfor
RETURN Dataframe
```

There is one obvious problem with this approach which we must keep in mind when working with the participant dataframe; from figure 1, we see that lap messages occur after record messages. We also know that record messages occur every second. This means that when a lap message is parsed, and the global variable is incremented, the variable will not be stored before the subsequent record message. All laps will therefore be stored exactly one second after the participants actually pressed the button.

After importing all data into dataframes, we concatenate all participants into a single dataframe. When working with a participant, we simply retrieve the participant dataframe from the concatenated dataframe and remove extraneous rows. We explore concatenation, and why this is performed, in later chapters.

## 3.2 Algorithm

Simplified, our algorithm can be broken down into the 3 general steps.

1. Obtain 4 laps, regardless of how many the participant had originally.
2. Identify a stop near each lap.
3. Fix laps without stops, if possible.

### 3.2.1 Distance Analysis

When inspecting laps individually, there are two broad categories in which they can be placed. Either the lap is slightly incorrect or it is completely incorrect. If the lap is only slightly incorrect, we can use the lap as a basis for the alignment algorithm. For the latter category, the registered lap is too erroneous to be useful. It can therefore not be used directly. While there could be a number of explanations for why a lap would fall into the second category, but the main explanation appears to be due to human error. As shown in table 17, there are 15 out of a total of 59 participants who pressed the lap button an incorrect number of times. For our algorithm to be able to determine which laps to use, and which to discard, we must be able to differentiate laps into the mentioned categories. We solve this by comparing the a value at the lap against an expected value for that lap.

| Registered laps | Number of participants |
|---|---|
| 3 | 4 |
| 4 | 44 |
| 5 | 6 |
| 6 | 3 |
| 8 | 1 |
| 20 | 1 |

Table 17: Number of registered laps

For this comparison purpose, we use distance. This is because all the participants have distance data, which is not the case for other data. When using the distance data, it was clear that there were some GPS drift. According to [9], the GPS location accuracy of the Garmin fitness devices is around

3 meters 95 percent of the time. This means that if the device is recording the GPS every second, the location recorded would be within a radius of 3 meters from the actual location. If standing still and the person do not pause an activity, the device can record up to 180 meters in 1 minute. This GPS drift can heavily impact the credibility of the distance data. There are also other factors for the distance variance through using the GPS. It can be environmental factors, such as interruption in the signal path of a GPS satellite to a device. If we assume that the rate of GPS drift is approximately constant for an individual participant, with different constants for various participants, we would conclude that the total distance between participants would vary greatly. As a consequence, the distance values for specific laps would also vary. In order to determine accurate expected laps, we could therefore use the relative distance as opposed to the absolute. By looking at distance of each lap as a percentage of the total lap, we create the following density plots.

We can create density plots to illustrate at what distance the participants registered a lap.

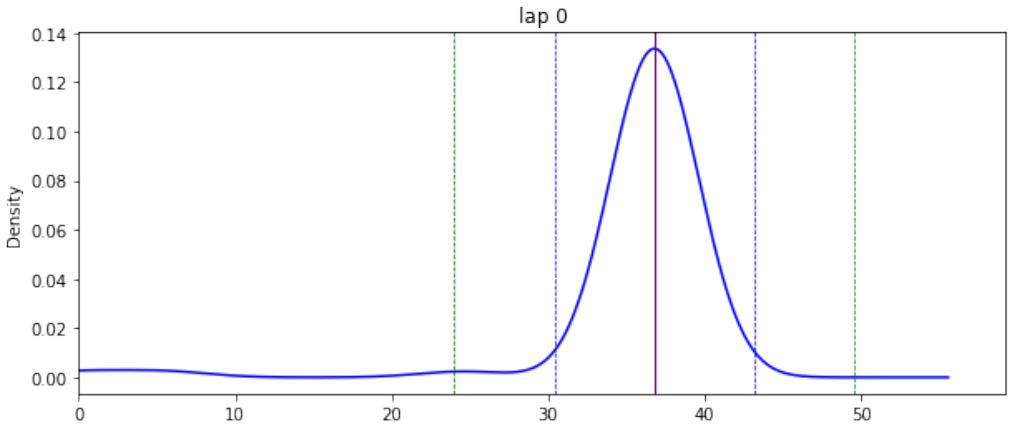


Figure 2: lap 0 - count values

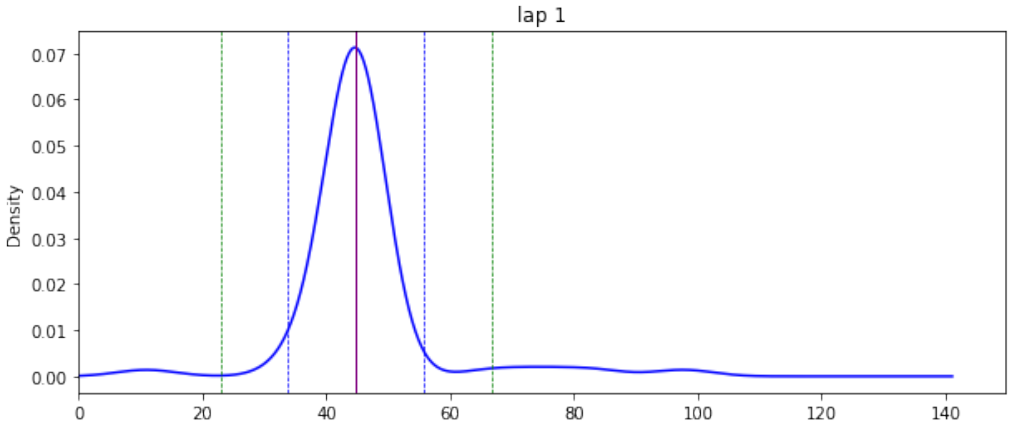


Figure 3: lap 1 - count values

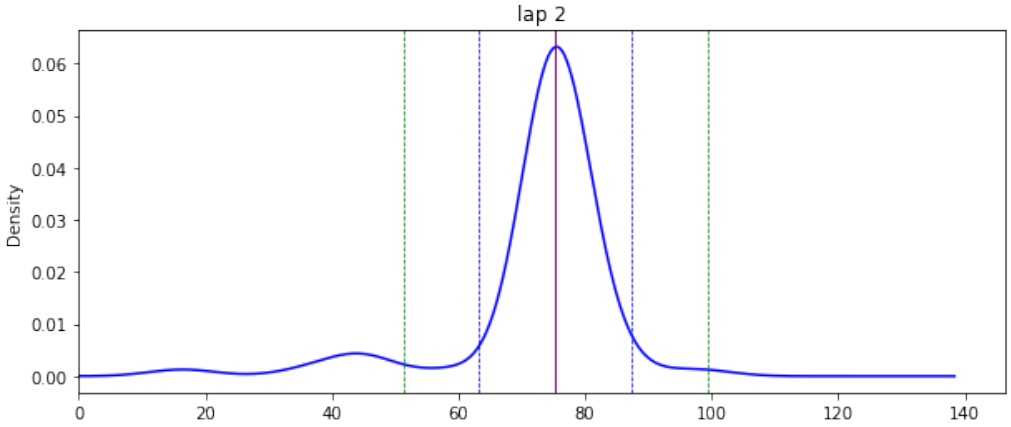


Figure 4: lap 2 - count values

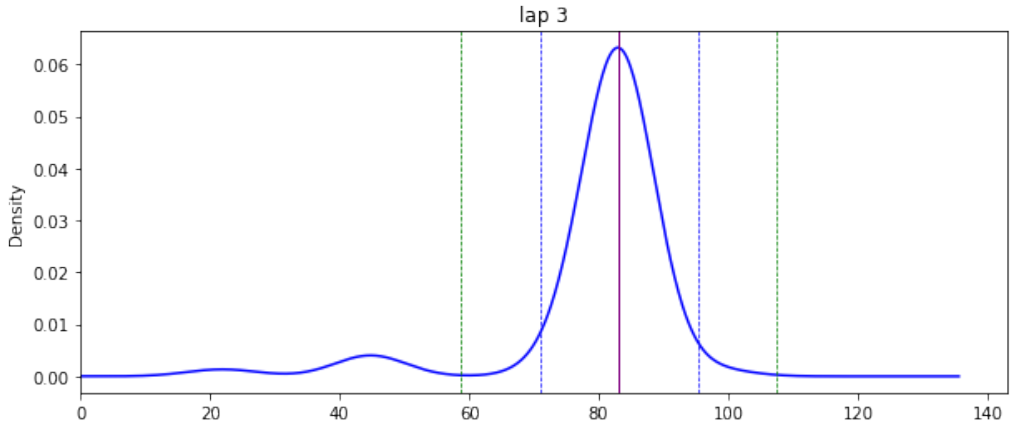


Figure 5: lap 3 - count values

We now have an expected percentage of the total distance, for each individual lap. This is used to calculate the expected distance value at laps for participants, by comparing it to the participant's total distance. We therefore have everything we need to start finding laps.

### 3.2.2 Finding laps

We start by retrieving both the distances and timestamps at the moments our participant pressed the lap button. This will give us two lists which represent the original, or registered, laps. We do this by searching the dataframe for indexes where the lap value is different from the previous value. These indexes are then stored in a list. As we recall, our import method causes all of these timestamps to be delayed. To mend the timestamps, we need to shift them backwards by exactly 1 second. This is done using a pandas timedelta. A timedelta is an absolute time duration, as opposed to a specific time [24]. By defining a timedelta as 1 second, we can subtract the timedelta from the retrieved timestamps. This results in a list of timestamps which matches the original lap messages precisely. Since we use timestamp as index in our dataframe, we can use these timestamps to retrieve the distance for each lap. We now have a list containing the timestamp, and another list with the distance, for all original laps. These original laps are stored so that they can be used when we generate the report for our participant.

The length of the lists will be determined by the number of times our participant pressed the lap button. For participants with extraneous laps,

we need a way to determine which laps to use and which to discard, For a participant with too few laps, we must determine which of the laps are missing. As an example, consider a participant who forgot the press the lap button at the start of stop. Such a participant will have three registered laps, numbered $0, 1, 2$, but we can not be certain which of the laps are missing. We solve this by comparing the expected laps against all registered laps. We will illustrate our approach with an example based on one of our participants;

| Lap | Distance | Expected Distance |
|---|---|---|
| 0 | 33690 | 33634 |
| 1 | 33691 | 40914 |
| 2 | 33692 | 68865 |
| 3 | 40990 | 75984 |
| 4 | 68961 | |
| 5 | 76068 | |

Table 18: Participant with too many laps

Consider a participant with 6 original laps as shown in table 18. From the table, it is immediately evident that something happened at laps $0, 1$ and 2. Whether due to human error, a device malfunction or any other reason, the first three laps are nearly identical. As a result of the error, there are three laps registered in rapid succession. To determine which original laps matches which expected lap, we calculate a value, $c = |lapDist_i - expDist_j|$, where $lapDist$ are the registered lap distances and $expDist$ are the expected distances. The value, $c$, is the absolute value of the expected lap distance subtracted from the registered lap distance. In other words, $c$ is a measurement of distance between a registered lap and an expected lap. We calculate all values of $c$ in table 19 for our example.

| | Expected Lap 0 | Expected Lap 1 | Expected Lap 2 | Expected Lap 3 |
|---|---|---|---|---|
| Registered Lap 0 | 56 | 7224 | 35175 | 42294 |
| Registered Lap 1 | 57 | 7223 | 35174 | 42293 |
| Registered Lap 2 | 58 | 7222 | 35173 | 42292 |
| Registered Lap 3 | 7356 | 76 | 27875 | 34994 |
| Registered Lap 4 | 35327 | 28047 | 96 | 7023 |
| Registered Lap 5 | 42434 | 35154 | 7203 | 84 |

Table 19: "Closeness" calculation example

For each expected lap, we choose the original lap with the lowest value of $c$. After choosing the best matching four laps among the original laps, we the remaining laps will be discarded. In our example, laps 1 and 2 will therefore be discarded, while laps $0, 3, 4, 5$ will be used. By matching laps to expected laps as shown, we end up with exactly four laps when the participant had extraneous button presses. If the participant had too few laps, we mend the situation by matching laps as shown in order to identify which laps are missing. The missing laps will then simply use the expected distance. We illustrate this by modifying our previous example

| Lap | Distance | Expected Distance |
|---|---|---|
| 0 | 33690 | 33634 |
| 1 | 68961 | 40914 |
| 2 | 76068 | 68865 |
| 3 | | 75984 |

Table 20: Participant with too few laps

For this participant, the laps $0, 1, 2$ will be used for $0, 2, 3$ respectively. From table 20, it is immediately evident that none of the registered laps match the expected lap 1. It is therefore evident that our missing lap was lap 1, which will be set to be equal to the expected distance of that lap.

When this approach is implemented in code, we could handle every scenario in the same method. We require two loops, one outer loop for the correct number of laps, and an inner loop for the registered laps. Instead of storing all values for $c$ and then choosing the lowest value, we could store only the lowest value of $c$ along with the original lap. As we loop over all registered laps, if we find a $c$ value that is lower than the previous, we update $c$ and the lap. By declaring the $c$ value before the inner loop, we would have a restraint on which registered lap values would be used. In other words, for a registered lap to be used, the lap must be closer to the expected value than the already initialized value of $c$. We illustrate this approach with the following pseudocode.

```
function:
list = empty list
for each ExpectedLap:
    closeness = StandardDeviationOfLap

    for each RegisteredLaps:
        if Abs(RegisteredLaps - ExpectedLap) < closeness:
            closeness = Abs(RegisteredLaps - ExpectedLap)
            lap = RegisteredLaps
    endfor

    if lap is defined:
        add lap to list
        endif
    if lap is not defined:
        add ExpectedLap to list
        endif
endfor
RETURN list
```

In this simplified code snippet, we see that we have to *CurrentLap* would be the current iteration of the registered laps in inner loop, and *ExpectedLap* is the expected lap as iterated over the outer loop. When the inner loop completes all iterations, the *lap* variable will contain the registered lap which is the best match for the current expected lap. If this variable is not initialized, it means that there are no registered laps that match the expected lap. In this case, we use the expected lap. Whichever we end up using, we append the lap to a list and return the list with the outer loop is complete. This gives us our list of four laps, regardless of how many time the participant pressed the button. It also allows us to check the laps of participants who pressed the button a correct number of times; if a participant registered four laps, but one or more of these are too far away from the expected value to be useful, we can discard that lap and instead use the expected lap.

### 3.2.3 Stop Detection

We now have a list of distance values for exactly four laps. For our purposes, we require timestamps. If we attempt to extract timestamps by searching

the dataframe for indexes matching our distances, we have two possible outcomes. If the distance value exists in the dataframe, we can retrieve the corresponding timestamp. This will always be the case if our list of distances is identical to the original list. The other possibility is that the distance value does not exist in the dataframe and no timestamp is retrieved. In order to find a timestamp corresponding to a distance value, we create a temporary dataframe which is constrained to only include rows where the distance value is less or equal to our distance. By finding the maximum index value in this dataframe, we find our desired timestamp. This is done for all laps, which gives us a list of four timestamps.

To find whether or not a lap is near a stop, we need to search a range around our timestamp. With a timedelta, we can subtract and add to the timestamp in order to obtain two new timestamps. These new timestamps define the start and end of the range in which we attempt to identify a stop. Choosing a good value for the timedelta is important as it controls the size of our range. If the range is too narrow, we run the risk of missing out on accurately identifying a stop. Conversely, if the range is too broad, we run the risk of basing our alignment on false positives. In our project, we found it beneficial to use two timedeltas. This allowed us to define the start of the range as closer to the timestamp than the end of the range. We settled on 15 seconds before, and 30 seconds after.

Slicing our dataframe with the two timestamps, we obtain all data rows between these timestamps. These rows are stored in a temporary dataframe which is used to identify the beginning of the stop. Our algorithm search the dataframe with progressively inaccurate methods. In other words, the algorithm becomes gradually more lenient in the approach used to detect stops. Initially, the algorithm searches for timestamps where both speed and power are less than or equal to a threshold. This threshold can be set arbitrarily small, including to zero. Using a threshold allows for leniency in the event of random noise in the data. As an example, a speed of 0.01 can be regarded as virtually zero. If the algorithm were to only search for speed and power equal to 0, we would risk missing actual stops in the event that there were fluctuations in the data. If the algorithm detects a stop, the timestamp is stored and we continue with the next lap. If no timestamps were found with these boundaries, the algorithm will try again with the same speed condition, but an altered condition for power data. The algorithm search for rows where speed is less than or equal to our threshold, while power is null for all rows of the dataframe. As we have seen earlier, there are participants without

power data. When stored in our dataframe, which contains a column for power data, the column will have a value of null for all rows. By specifically declaring that power must be null in all rows, we differentiate participants without power data from participants with power data.

If both preceding attempts have failed, we make a final attempt where we ignore power. As both of the previous attempts failed, if we are successful now, it would mean we have a participant with speed less than or equal to the threshold, and a power greater than the threshold. We will discuss in later chapters why this was necessary. If we are still unsuccessful in identifying any stops, we flag the lap as not containing any laps. We then move on to the next lap and repeat this whole process for every lap.

## 3.3 Reports

We generate a report for each participant in order to facilitate verification and quality control. We want the reports to include relevant graphs, a table and a text paragraph. We also want the participant ID as a header on every page, and a footer with both a page number and the total number of pages.

### 3.3.1 Data

For the graphs, we want a plot of the entire race for the participant and individual lap plots. The full plot should include vertical lines to mark where the participant pressed the lap button. We create this plot by using the participant dataframe and the original timestamps, with matplotlib. For the individual lap plots, we want to plot a range around the aligned timestamp together with a vertical line on the aligned timestamp. With timedeltas, we define a range which is used to retrieve the relevant rows from the participant dataframe. We then need to plot these rows of data, alongside the aligned timestamp, to obtain our lap plot. This is repeated for all laps.

The table is stored as a CSV file containing all original timestamps, their respective distances and the aligned timestamps. We also include an empty column for final adjustments. This allows adjustments to be made directly on the report when printed on a piece of paper.

The text paragraph is used to give us insight into how the algorithm processed the participant in a human-readable fashion. This gives us a quick overview of the alignment process. We do this by creating text-files for various scenarios. For example, if a participant had a lap for which the

algorithm is unable to find any stops, we write the lap number to a file. When generating a report for this participant, we read the file and create a text string. Using our example, assuming the lap in question was lap 1, the text string would read "The following lap(s) have problems that could not be solved automatically and manual correction is likely necessary: 1". As is evident by the specificity of the string, every scenario must be parsed by a unique function for us to obtain the correct string variable. We achieve this by looping over every text-file for a participant, and passing the files to a switch function.

```
FUNCTION (file):
if file = case_1:
    return function_1(file)
end if
if file = case_2:
    return function_2(file)
end if
```

As we note from the pseudocode, the switch returns a function based on the file. These functions return the appropriate strings for the specific files. We use 5 different functions for reading files, which we will discuss in later chapters. As some participants have multiple files, all strings returned by the functions include a line break at the beginning of the string. When multiple strings are joined together, this will increases readability.

### 3.3.2 Page template

As we have all the data necessary for the report, we can attempt to build the document. If we do this, we would be faced with a problem; the full plot contains a large amount of data - if it is to be readable, it must be as large as possible. A normal A4 page is too narrow for this plot to be read easily, if we want to study the details. By rotating the page, such that the orientation is in landscape, we could maximize the available horizontal space. We therefore want to use a landscape layout for the first page. The remaining pages, however, we still want to utilize a portrait layout. PLATYPUS can solve this using page templates.

We start by creating page templates. We need two templates, one for landscape and one for portrait. Each of these templates require a frame, a

defined page size, and a unique ID. For our purposes, the page sizes will be A4 and landscape A4. The frames have to be set such that they correspond to the page sizes. The portrait frame can be defined with the same measurements as an A4 sheet of paper, as can the landscape frame, only with the reverse order. Because an A4 piece of paper measures $210mm \times 297mm$ [25], the landscape frame will be defined as $297mm \times 210mm$. After creating the templates, we store them in a list. In order to use the templates, they must be attached to the document container. This is necessary if we are to use them as we cannot use page templates that are not stored in the document container.

As mentioned in previous chapters, PLATYPUS handles switching between page templates through the list of elements. By appending a function, "NextPageTemplate()", with a page template ID as argument to our list, PLATYPUS will apply the page template selection on the following page. For our purposes, we want to use the landscape layout for the first page. As PLATYPUS cannot alter the layout of the current page, we must be certain that the landscape template is used for the initial page. There are two things we need to do in order to achieve this; we must initialize the document container with the list as an argument, and we must make sure that the landscape template is the first entry in the list. This causes the document to default to the desired page layout for the first page.

### 3.3.3 Canvasmaker

We generated reports with the PLATYPUS engine, as this allowed us to use flowables. Footers and headers, however, are normally statically placed element. These are therefore ill-suited for PLATYPUS to place dynamically. We solve this with a *canvasmaker* when we are building the document. A canvasmaker, as the name implies, is an optional argument in the build method which specifies the canvases in the document. It allows us to extend the default canvas class and thereby change every canvas in the document. With a canvasmaker, we can gain access to the canvas despite working with PLATYPUS. This facilitates the creation of a header and footer, by treating them as static object to be drawn on every page. Our header is the ID of the participant, and can therefore be drawn with this method. Our footer, on the other hand, is more complicated due to the total page number being unknown until the build process is complete. From [26], we find that this complication can be circumvented by storing partial pages. To explain this

approach in detail, we must consider *how* a document is built.

When a document is built, it is built one page at a time. From the perspective of canvases, this means that we draw on a canvas until completion. When we are finished with the first canvas, we save it and move on to the next. When there are no more elements left to draw, we save the final canvas, and finish by saving the document. By modifying the canvas class, we can change the method used for saving a canvas. This allows us to store partial canvases in a dictionary. When there are no more elements to draw, we access the dictionary and complete the canvases before the document is saved. At this point, we know the total number of canvases, and therefore the page number.

With this approach implemented, we generate reports for all participants.

# 4 Evaluation

## 4.1 Import verification

It was important to make sure that the data stored in the dataframe was accurate with the data in the messages. The importer was therefore verified by plotting various plots from the dataframe and compare them with the message plot. Before plotting the dataframe it was necessary to locate and remove the data before and after the correct end and start point for each participant. This is necessary because of the concatenation which was done earlier. To verify the data, we looked at some key points.

### 4.1.1 First 600 values

The first six hundred record messages was plotted and compared to the first six hundred dataframe rows. This information is interesting to look at, because of record message verification after the import. The first plot was created by using the record messages, and the second plot was made by the importer dataframe. The expected outcome was similar plots for most participants, and some differences for participants with missing data. For most of the participants the graphs is the same, and the tick values are the same which indicates that the importer behaved as expected with the record messages. Other participants have different tick values, and different graphs which is caused by the concatenation in the importer. If there are missing data the record message plot will not be in the ten minutes range. Listed

under are a good example of this happening. Can also see the tick difference in the last tick for accumulated power, power and distance. When plotting with the dataframe there are some null values at the start of the plot. These will not be plotted, while in the raw plot it will connect the values. Heartrate, timestamp and participant id is not shown because of sensitive data.

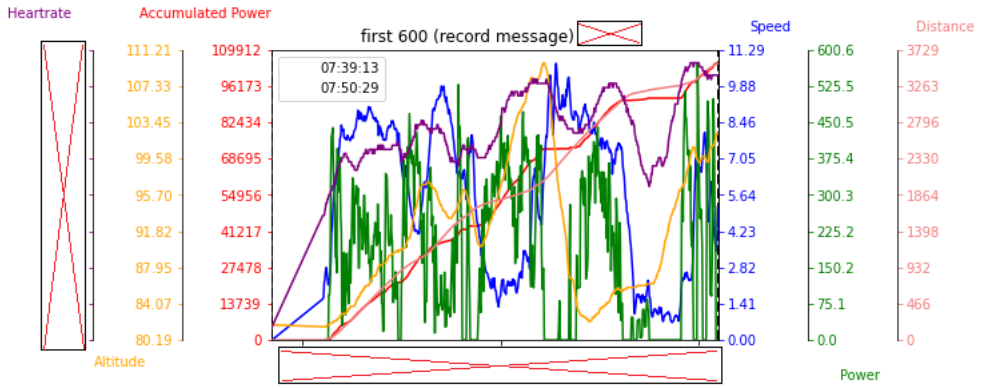


Figure 6: First 600 - record messages

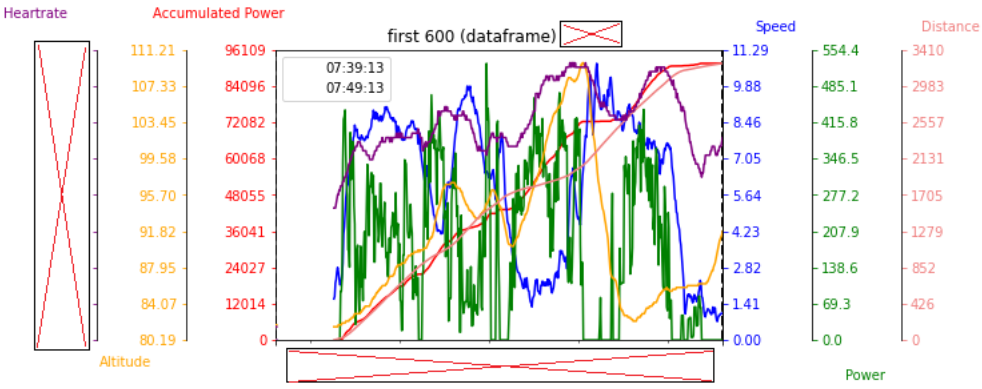


Figure 7: first 600 - dataframe

### 4.1.2 First 10 minutes

The first ten minutes at the start was plotted. This information is interesting to look at, because of record message verification after the import. The data used in both plots should be the same, except when there are missing record

message data. This should however make no difference in the plots, since the dataframe only will have null values. If there are some missing data from the record messages, the dataframe will have a larger set because of the concatenation. The start and end point should be the same for both plots, and the tick values should also be the same.

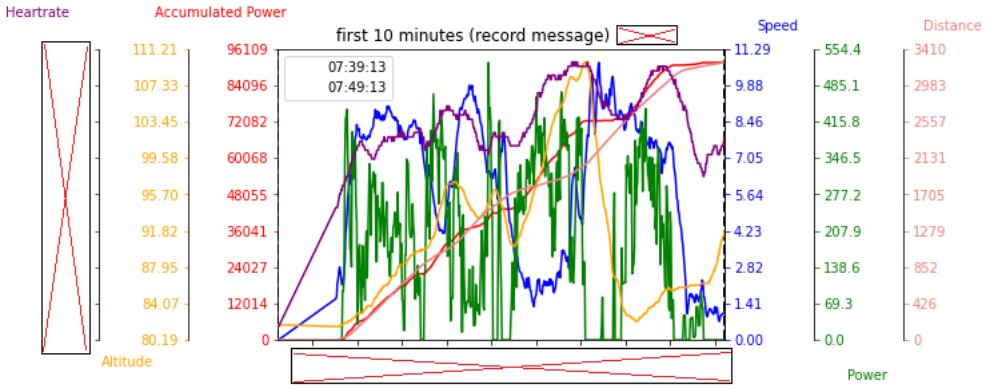


Figure 8: First 10 minutes - record messages

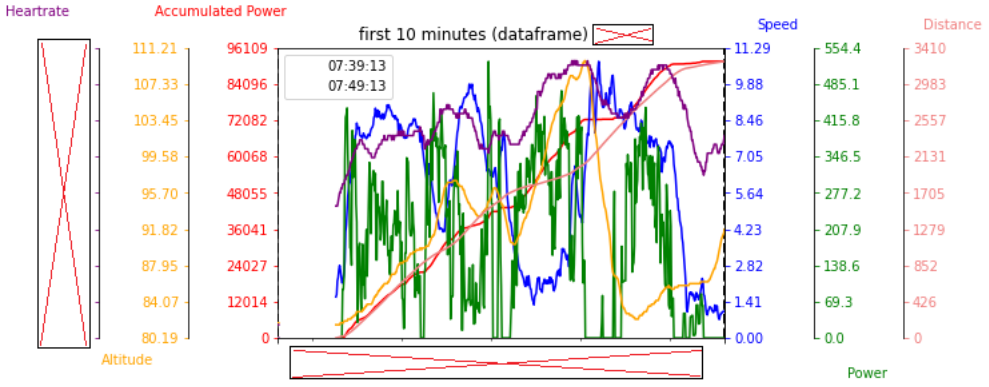


Figure 9: first 10 minutes - dataframe

### 4.1.3 Laps

The lap messages after the import was verified by plotting the lap changes which was stored in the lap messages. The closest record messages around the lap messages were also plotted to further validate the record messages.

Therefore we chose to plot the record messages thirty seconds before and after each lap message. This might not be the case for the record and lap message plot, if there is some missing data. Some participants have pressed the lap button at the wrong time. Depending on the time of this button press, it may not be possible to plot thirty seconds before and after. This value would then be set at the button press. Since there also might be missing data in the record message, the same problems as mentioned above can happen. If the missing data in the record message is at the timestamp of the lap message, it will not be possible to plot at this point. There is a one second difference for where the lap change is registered. This was noticed early on in the project, but admittedly almost forgotten, as this one second delay is mostly inconsequential. As shown in our implementation, solving this is trivial using a pandas timedelta.

All participants had an extra lap message which was purposely not stored in the dataframe, because of the lack of record data before this message. This last lap message is also not plotted in the lap and record plot, since there's no record messages stored around this timestamp. Inspecting the message in isolation, however, reveals that it is caused by the end of the the race and is a summary message. It therefore does not represent a lap button press made by our participants.

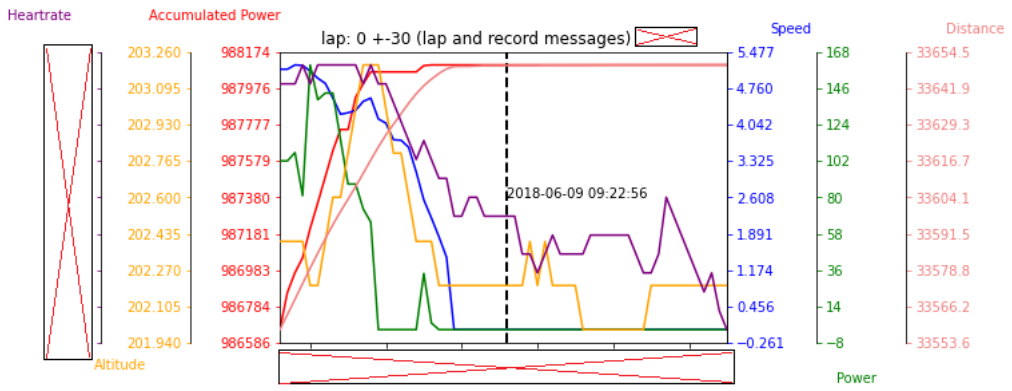


Figure 10: First lap - record- and lap messages

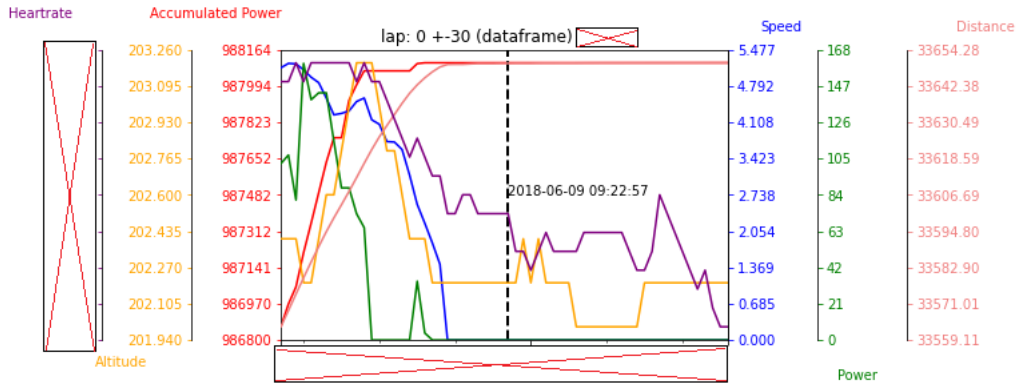


Figure 11: first lap - dataframe

### 4.1.4 Missed messages

Record messages are meant to be sent every second. This is important, as we use the timestamp as index in our dataframe. This is however not always the case; some participants seem to be missing messages. In order to investigate this issue, we must count how many messages a participant is missing. By subtracting the minimum timestamp from the maximum, we obtain a timedelta which tells us how long a participant spent on the race. After converting the timedelta into seconds, we can compare this to the number of rows in the paricipants dataframe. In table 21, we show how many messages are missing, and how many participants were missing exactly that many messages.

| Number of participants | Missed messages |
|---|---|
| 1 | 1 |
| 4 | 2 |
| 1 | 6 |
| 1 | 17 |
| 1 | 217 |
| 1 | 835 |
| 1 | 2681 |

Table 21: Missing messages

As shown in the table, we see that a total of 10 participants had missing

messages. There were a single participant missing 1 message, 4 participants missing 2 messages, and so on. While most of the participants with missing messages were only missing a few, we see that there are 3 participants missing more than 200 messages. It is important to keep in mind that each missing messages corresponds to a whole second of lost data. If a participant is missing more than 200 messages, this corresponds to over three minutes being lost.

Our initial concern when noticing this problem, was whether or not it was caused by importing the messages into our dataframe. If our import was causing the problem, we would have to rethink our approach. To investigate this, we count the number of record messages and compare them to the already counted number of rows. As shown in table 22, participants with missing messages have the same amount of in rows as compared to messages in FIT file.

| Participant | Rows in dataframe | Messages in FIT file |
|---|---|---|
| A | 24309 | 24309 |
| B | 15789 | 15789 |
| C | 11350 | 11350 |
| D | 12857 | 12857 |
| E | 20129 | 20129 |
| F | 16391 | 16391 |
| G | 16348 | 16348 |
| H | 13836 | 13836 |
| I | 15443 | 15443 |
| J | 13208 | 13208 |

Table 22: Missing messages: row and message comparison

This means that the issue is not caused by our implementation, but rather is a problem with our data. Because we use timestamps as index, we will get errors in our code when we attempt to use a timestamp which is missing. To resolve this problem, we concatenate all participant dataframes into a single concatenated dataframe. As a simplified example, we consider three participants that have been concatenated into a single dataframe, as show in table 23.

| | Participant 1 | | | Participant 2 | | | Participant 3 | | |
|---|---|---|---|---|---|---|---|---|---|
| | type | lap | ... | type | lap | ... | type | lap | ... |
| timestamp | | | | | | | | | |
| 2018-06-09 07:39:01 | NaN | NaN | ... | 1.0 | 0.0 | ... | NaN | NaN | ... |
| 2018-06-09 07:39:02 | NaN | NaN | ... | 1.0 | 0.0 | ... | NaN | NaN | ... |
| 2018-06-09 07:44:02 | NaN | NaN | ... | NaN | NaN | ... | 1.0 | 0.0 | ... |
| 2018-06-09 07:44:03 | NaN | NaN | ... | NaN | NaN | ... | 1.0 | 0.0 | ... |
| 2018-06-09 07:59:01 | 1.0 | 0.0 | ... | NaN | NaN | ... | NaN | NaN | ... |
| 2018-06-09 07:59:02 | 1.0 | 0.0 | ... | NaN | NaN | ... | NaN | NaN | ... |

Table 23: Concatenated dataframe

From this concatenated dataframe, we can retrieve the individual participant dataframes. The retreived dataframe will retain the number of rows from the concatenated, as shown in table 24. If we remove all rows where all columns have a null value, we end up with a participant dataframe identical to the one we had before concatenation. From this, we take the first and last timestamps. This gives us the total duration for our participant. If we now retrieve the participant dataframe from the concatenated datafram *again*, using these timestamps as lower and upper bounds, we obtain a participant dataframe without missing rows. These "restored" rows do not contain any data, but allow us to retain the integrity of our index.

| | Participant 1 | | | Participant 2 | | | Participant 3 | | |
|---|---|---|---|---|---|---|---|---|---|
| | type | lap | ... | type | lap | ... | type | lap | ... |
| timestamp | | | | | | | | | |
| 2018-06-09 07:39:01 | NaN | NaN | ... | 1.0 | 0.0 | ... | NaN | NaN | ... |
| 2018-06-09 07:39:02 | NaN | NaN | ... | 1.0 | 0.0 | ... | NaN | NaN | ... |
| 2018-06-09 07:44:02 | NaN | NaN | ... | NaN | NaN | ... | 1.0 | 0.0 | ... |
| 2018-06-09 07:44:03 | NaN | NaN | ... | NaN | NaN | ... | 1.0 | 0.0 | ... |
| 2018-06-09 07:59:01 | 1.0 | 0.0 | ... | NaN | NaN | ... | NaN | NaN | ... |
| 2018-06-09 07:59:02 | 1.0 | 0.0 | ... | NaN | NaN | ... | NaN | NaN | ... |

Table 24: Concatenated dataframe

To reiterate, in order to make sure a participant dataframe is without missing rows, we perform these steps.

1. Create participant dataframes.
2. Store all participant dataframes in a concatenated dataframe.
3. Retrieve a temporary participant dataframe.
4. Remove all rows where all columns have a null value from the temporary dataframe.
5. Find the first and last timestamp indexes from the temporary dataframe.
6. Retrieve the participant dataframe using timestamps, from the concatenated dataframe.

### 4.1.5 Missing data

In table 21, we saw that a few participants were missing a large amount of data. We examine this closer by using the two worst offenders as examples.

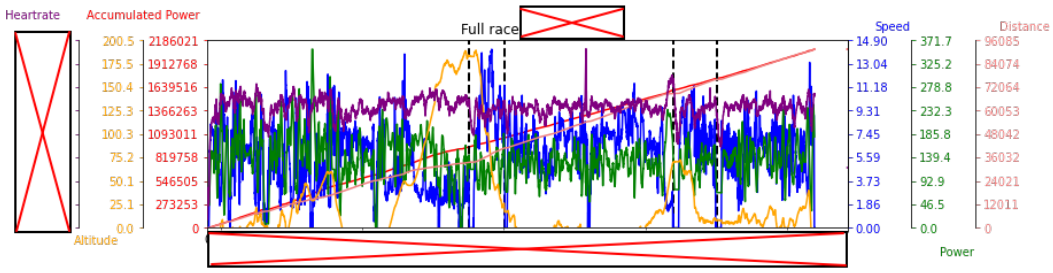


Figure 12: 832 missing messages

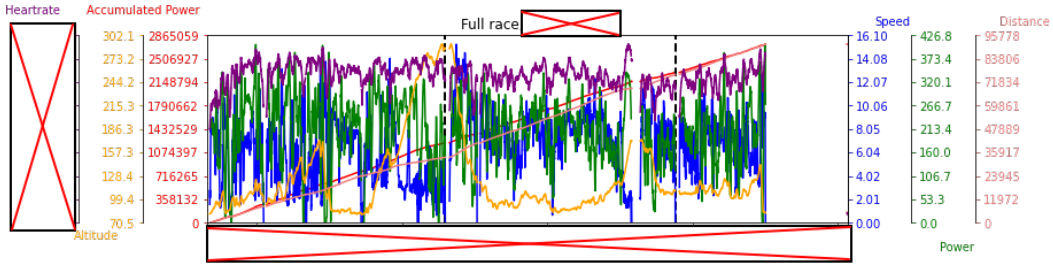


Figure 13: 2681 missing messages

It is immediately clear from figures 12 and 13 that most of the missing messages appear at the end of the plot. When this happens, we can still align stops, as the race itself is intact. For the participant with 2681 missing messages, however, we see there is a gap in the graph. This happens to coincide with a probable stop location, which means this stop cannot be aligned. With the participant's report in hand, where the plot is larger, we see that this is the case for every stop - this participant is missing data at every expected plot. The entire alignment procedure fails for this participant due to the lack of data.

An observant reader might notice an apparent discrepancy between figure 13 and table 17; in the figure, registered laps are marked with a black vertical line. There appears to be two such lines. In the table, where we show the number of registered laps for all participants, there are no participants counted as having two laps. We believe this to only be a visual problem with this particular participant, as they appeared to have more registered laps.

## 4.2 Algorithm

### 4.2.1 Distance analysis and Chauvenet

Initially, we used an absolute distance value as the expected lap value. Due to the variation in total distance, this would sometimes lead to poor choices of the initial suggested timestamp and therefore poor alignment. One of our biggest problems was identifying stops for participants with only three lap button presses. We noticed that participants with a correct number of laps, mostly pressed the button at the same distance. In order to improve the average distance, we applied Chauvenet's criterion. While we have stated plainly the controversial nature of this criterion, we must keep in mind the purpose for which it is used; we are simply trying to find a distance value which is likely to yield a stop. Using lap 2 as an example, if we look at the distances of every participant with a correct number of laps, we find that they mostly press the button at the same distance value. This is illustrated in figure 14.

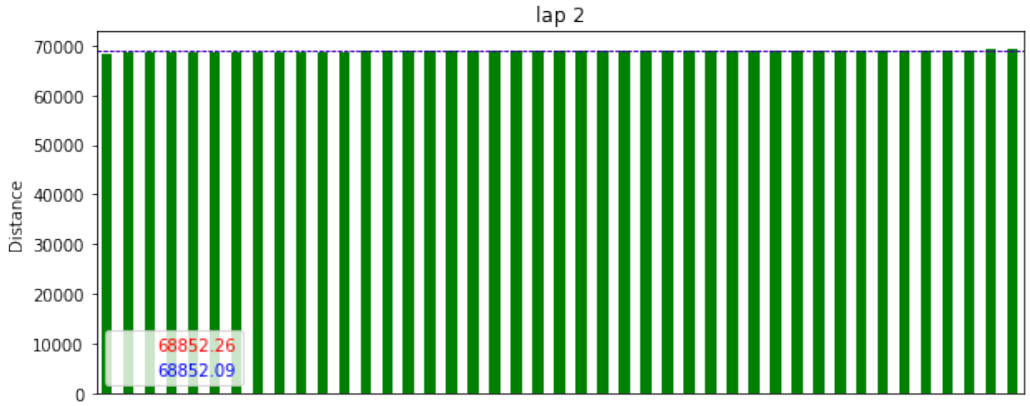


Figure 14: lap 2 - correct number of participants

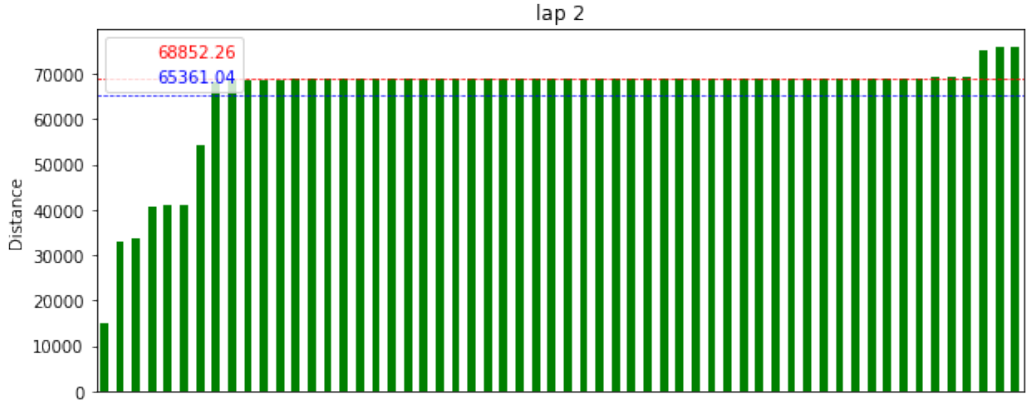


Figure 15: lap 2 - all participants

In figure 15, we see the distances of all participants for lap 2. Naturally there is more variance, but we notice how many participants have approximately the same value. The mean distance, denoted by the blue line, is however off-center from these participants. By using Chauvenet's criterion, we obtain a new mean distance, denoted in red, which better fits these participants. We therefore judged Chauvenet's criterion, while controversial, worthwhile for our purposes.

It was further suggested to us to attempt to use a relative distance, instead of absolute distance, to combat the issue of varying total distances. If, as

previously stated, we assume the rate of GPS drift to be approximately constant, this would result in better values, as we simply circumvent the problem.

### 4.2.2 Ignoring Power

We explained how our algorithm searches for stops using boundaries for speed and power data. In all cases, speed needed to be less than or equal to a threshold. Power, however, was initially less than or equal to a threshold. If this failed, power had to be null for all values. In the event that this also fails to identify a stop, we ignored power and only used speed. We will explore our reasoning

If neither of the two first methods detected a stop, we are left with two options. Either there are no stops within the rows of the dataframe, or our algorithm has failed to find it thus far. Assuming the latter option to be true, there is a stop which we have not yet found and we must consider explanations as to why the algorithm has failed thus far. We must conclude that one of the following statements are true.

- Speed is greater than the threshold.
- Power is greater than the threshold.
- Both are greater than the threshold.

Ignoring the possibility that the threshold is incorrectly set, we can consider the physical implications of each statement in order to judge their plausibility. If both speed and power is greater than the threshold, the participant is moving and we therefore do not have a stop. If speed is greater than the threshold while power is not, the participant is moving without pedaling the bicycle. The participant is in all likelihood either rolling down a hill, or simply stopped pedalling temporarily while momentum continues to move the bicycle forwards. In either case, the participant is moving and we therefore do not have a stop. If power is greater than the threshold while speed is not, the power device is registering power, but the participant is at a stand still. Depending on how the device collects the power data, this might occur if a participant is maintaining pressure on the pedals while engaging the brakes. It might also occur due to faulty hardware. Regardless of the mechanism

which causes the elevated power data, this is the only statement which can be true given our assumptions.

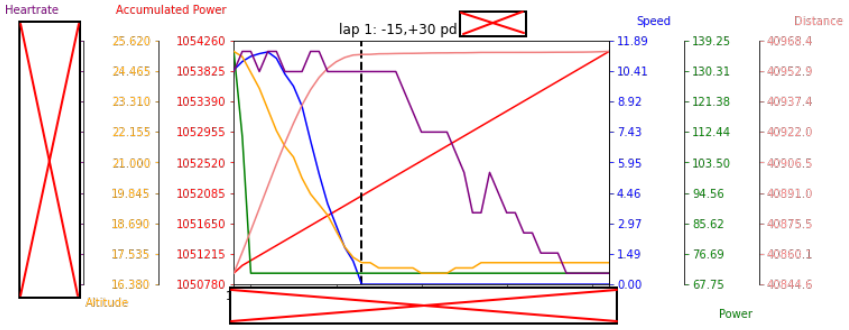


Figure 16: Apparent stop with speed data

From figure 4.2.2, we note this exact scenario occurring. We can clearly see that the distance value stops growing and becomes a horizontal line. While altitude changes slightly after the aligned timestamp, we consider this within reason. We can also note how the heart rate of the participant is decreasing steadily. This is another strong indication that the participant is not moving. The only piece of data that indicate otherwise, is the power data. While we do not have access to the equipment our participants used, and therefore are unable to given an exact explanation for this behaviour, we believe we have explored sufficiently plausible explanations as to determine that this participant have faulty power data.

## 4.3 Reports

### 4.3.1 Maximizing the full plot

As mentioned previously, if the entire report was in a portrait layout, the full plot was sometimes hard to read. We were suggested to try and rotate the image in order to give it as much horizontal space as possible. This could be achieved by creating a new flowable class which extends the Image flowable class. All that is required of this class is a wrap method which returns height and width of the image, and a draw method. The wrap method simply calls the wrap method from the Image class, but returns the height and width variables in reverse order. The draw method utilize an inbuilt rotate function to rotate the image before drawing it.

Our full plot was now an appropriate size and readable, but it caused a new problem with no easy solution. PLATYPUS build from the list of elements from top to bottom of a page. As such, it will place the element after our rotated image below the image. Our table would therefore be on the next page after the plot. This left a huge portion of our first page unused, as there was free space next to the plot. While it might have been possible to overcome this by creating two columns on the page is such a way that the plot would be in one column and the table in the other, this would be awkward to read. The plot would be rotated, but the table would not, meaning that two elements on the same page have different orientations. While rotating the an Image flowable was simple, we found that rotating a table was not as straight forward as rotating an image. While investigating ways to rotate the table, we realized that what we were actually trying to achieve, was for the whole page to be rotated. Instead of rotating the elements, and creating multiple columns, we should simply change the page template to be landscape.

### 4.3.2 Text

After realizing that the first page of the reports usually had some unused space, we decided that this could be a good option for including text. The text could give us quick insight into the alignment process and immediately point out potential problems. By storing the information we want to capture in text-files, we could simply create files at certain stages of our implementation. This approach made it easy to implement more scenarios in our code. The approach of using a dictionary switch for selecting the function to parse a file, meant that we could condense code and avoid if-statements. In other words, while currently sufficient for our specific project, the specific scenarios, and the corresponding text created for them, was designed for ease of future feature implementation. We currently have 5 scenarios for creating text.

1. Incorrect number of original laps
2. Total distance implies participant did not complete race
3. Which original laps were used as a basis for which lap
4. Aligned lap has a distance more than 2 standard deviations away from expected value

5. Lap could not be aligned automatically

Scenario 1 is self explanatory.

Scenario 2 occurs when the participant have a total distance which is more than 2 standard deviations from the mean. Only a single participant, whom we shall inspect closer to investigate the issue, were a match for scenario 2. While full plots are inherently hard to read at this size, as previously discussed, we can still see that the participant have anomalous speed data from figure 4.3.2. The distance also flattens out into a nearly horizontal line. Judging by the heart rate and altitude, however, the participant appears to have completed the race; they have altitude data that appear to match that of other participants, and the heart rate appears to indicate physical exertion. As the altitude and heart rate data appear to be complete, we can assume that the anomalous situation is not due to an import error - if it was, we would expect all data to be affected as everything is important from the same record messages. We can only conclude that this participant must have has some sort of hardware, or sensor, malfunction.

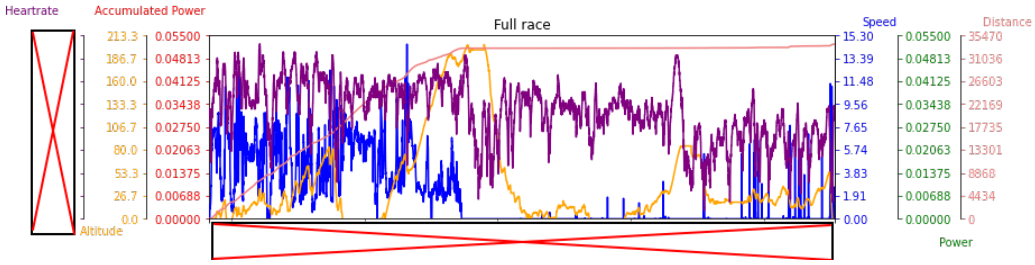


Figure 17: Scenario 2: Full plot of participant

The third scenario tells us which original timestamps were used for which laps as a basis for stop detection. While this information can be deduced from the table by matching original timestamps to aligned, it is helpful to be able to read the information plainly - especially when a participant have an abnormally large amount of original button presses.

Scenario 4 occurs whenever the aligned lap is more than 2 standard deviations away from the expected value. While merely indicative of a problem with the alignment, as this is not sufficient to ascertain, this text immediately points out which laps require closer inspection.

The final scenario appears in reports where one or more laps have complete failed alignment.

### 4.3.3 Plots and reports

A significant amount of time was spend during our project to fine-tune our plots to be as useful and readable as possible. As a concrete example, our initial full plots were unreadable due to the power data being too erratic. By plotting these values with a rolling average, we were able to smooth the power graph and make more readable plots. Our reports were sent to people connected to NEEDED by our supervisor. We received feedback which complimented our graphs as useful and pretty. Our lap plots show 15 seconds before the aligned timestamp and 30 seconds after. We were told that this was useful, as the period before the start contain useful information for their purposes. They also commented on the aligned timestamp and said it seemed to be aligned correctly. We received no other comments or feedback, but rather a request for additional features and plots. We find it encouraging that the feedback requested additional features, as it implies our reports are useful for more than verifying the alignment. We will return to the specific requests in closing of our thesis.

## 4.4 Success rate

While judging the success rate of our project is inherently a subjective exercise, we have looked at every the reports of every participant in order to ascertain whether or not we would adjust the automatic alignment. In table 25, we have placed every lap in one of 5 categories. These categories are based on how many seconds we believe the lap must be adjusted. A total of 9 laps are marked as non-applicable. This is because one participant only made 3 stops, one participant was missing too much data as previously discussed, and one participant had problems with speed and distance data as shown in figure 4.3.2.

| Deviation | Number of laps |
|---|---|
| 0-1 s | 168 |
| 2-4 s | 12 |
| 5-9 s | 5 |
| 10-14 s | 8 |
| > 14 s | 34 |
| NA | 9 |

Table 25: Deviation after automatic adjustment (subjective)

While inspecting every lap, we initially placed them in a list without categories. If we felt that a lap needed no adjustment, we counted it as 0s. As a test, we inspected and counted every lap again. Unsurprisingly, the results were slightly different. The problem is the subjective nature of determining the fine-tuning. In figure 4.4, we see an arguably well-aligned stop. Speed is close to zero, and distance appears unchanging. We do, however, notice that if the alignment was placed a second later, speed would reach zero. On the other hand, speed is very low, and one might even argue that if the alignment was placed a second earlier, we would still be within a tolerable level of speed. The rate of change for speed also changes at that moment, which might indicate that the participant is applying more breaks as they come to a complete stop. For this reason, it seemed more natural, as an attempt at objectivity, to place the laps in categories.

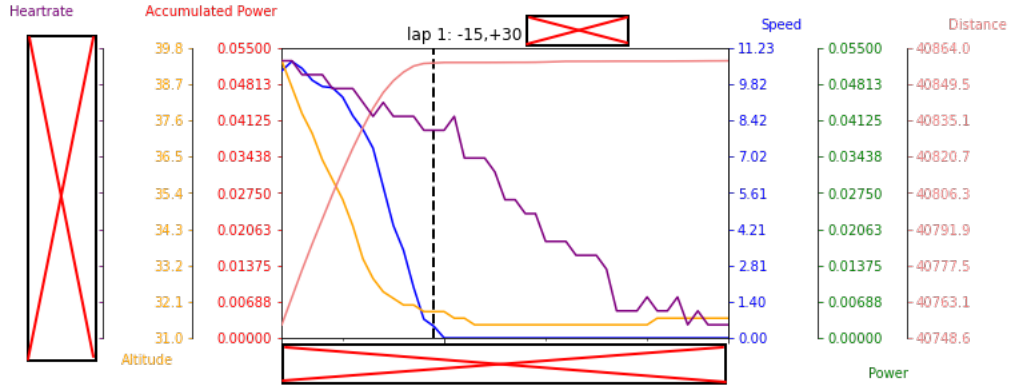


Figure 18: Aligned lap

All of this is to say that an accurate success rate is hard, if not impossible,

to calculate. We must also keep in mind that our goal for the project was alignment in and of itself; we do not use the data ourselves *after* alignment, and therefore cannot judge the quality of the alignment by its usefulness in further projects.

Despite all of this, with our table in mind, we believe we are justified in claiming some level of success; we have at the very least developed tools that simplify the manual process of aligning data. At best, if we consider every lap in the category of zero to one second as a success, we can calculate our success rate as $\frac{168}{236} \times 100 \approx 71\%$.

# 5 Conclusion and Future Work

## 5.1 Conclusion

Our goal was to make an algorithm that aligns data by precisely identifying stops. As a step in the alignment process, we generated reports for verification and fine-tuning purposes. We have shown how our algorithm attempt to find precise stops and how the reports not only visually represents data for inspection, but also guides the reader to potentially important information about the alignment process itself. We are therefore able to both perform automatic alignment on our data, but also lessen the burden of making manual corrections and verifications.

## 5.2 Further verify import

The FIT files are stored as binary. The library used in this algorithm to convert the FIT files to readable messages was the fitparse library. The fitparse is a python library to parse ANT/Garmin FIT files. Recently it's been frequently updated. It would have been a benefit to verify the parsing from the FIT files to messages. This could be done by parsing the Garmin FIT files without the library. Could then compare all the fitparse messages with the self implemented parsings.

## 5.3 Examining actual stops

When our algorithm fails to find an actual stop in a range around our suggested distance, we currently have no way to remedy the situation and resort

to simply flagging the stop in the report. A solution to this might be to instead approach the problem from the other end. When no stops are found around a certain point, instead of focusing on the point, we could look at the actual stops. For every participant, we could look at the ranges in which they appear to stop. By creating a dataframe for each stop in the entire race, we could examine the stops one by one. We could then match these actual stops to proposed stops based on distance analysis and button presses. Where distance analysis currently fails us, we could use the stop which best matches the expected stop. Furthermore, if this proposed approach store timedeltas corresponding to the duration of each stop, we could use this to identify false positives; if a stop is below a certain threshold in duration, it could be discarded. This would allow our algorithm to not just focus on distance, but also duration of stops. This would solve a hypothetical problem; let us consider a participant who stops slightly too early. They press the button and stop, but realize their mistake and start cycling a bit more. They then stop at the correct location without pressing the button. If the distance between the initial stop and the actual stop is short enough, our algorithm might choose the initial, slightly incorrect, stop. By examining duration as well as distance, we could find that the initial stop was too brief and discard it, thus focusing on the actual stop. While we have not identified this problem amongst any participants, implementing the suggested approach would solve this problem, as well as the aforementioned problem.

## 5.4 Reports

For our purposes, reports are used as a final manual correction in stop detection. Based on feedback we have received, these reports can also be used for other purposes. We received feedback both on the usefulness of the plots and that our aligned timestamp seemed appropriate. The feedback also expressed a desire for more plots and slight modifications which would be useful in other circumstances. This included vertical lines which indicates median heart rate for various groups of people, as well as plots that compare the speed, power and heart rate for every lap a participant had. This would be especially useful when comparing the heart rate at the different stops.

The text paragraph in our reports could also easily be modified to inform readers of more specific situations. To use the feedback as an example, the report could inform the reader if the heart rate of the participant is outside a defined range around the median for their group. In this case, we would

simply compare the participant's heart rate at a desired point with a criteria. If the criteria is met, we write a file which is then read by our report generator. Our report generator would then need a function which defines the string which is to be displayed in our text portion of the report. This can be done with an arbitrary amount of situations based only on the wishes of the end user.

# References

[1] T. Wiktorski, M. F. Bjørkavoll-Bergseth and S. Ørn, *Data-centered Analysis of Physiological Metrics During a Strenuous Physical Exercise*, n.d.

[2] John R. Taylor, *An introduction to Error Analysis 2nd Edition*, University Science Books, 1997

[3] *Flexible and Interoperable Data Transfer (FIT) Protocol*, https://developer.garmin.com/fit/protocol/, (Accessed May 1, 2021)

[4] *Common FIT File Types*, https://developer.garmin.com/fit/file-types/, (Accessed May 7, 2021)

[5] *FIT Course Files*, https://developer.garmin.com/fit/file-types/course/, (Accessed May 7, 2021)

[6] *Workout File*, https://developer.garmin.com/fit/file-types/workout/, (Accessed May 7, 2021)

[7] *Activity File*, https://developer.garmin.com/fit/file-types/activity/, (Accessed May 7, 2021)

[8] *Profile.csv*, https://developer.garmin.com/fit/download/, (Accessed May 7, 2021)

[9] *GPS Drift and Environmental Factors*, https://support.garmin.com/nb-NO/?faq=z0n0KE1XVF0Pe4Su8QiZgA, (Accessed May 5, 2021)

[10] *What Is the Seated/Standing Position Metric and How Is It Measured by Garmin Power Meters?*, https://support.garmin.com/en-US/?faq=UV4ObLXDaE979tyrI6t0VA, (Accessed May 12, 2021)

[11] *Difference in Total Ascent and Total Descent*, https://support.garmin.com/en-US/?faq=fQOZqCDNUr9izx2Tz9g1k7, (Accessed May 12, 2021)

[12] *What Is Normalized Power and How Is It Calculated on My Garmin Device?*, https://support.garmin.com/en-US/?faq=8r4llV3DFK5jc13BOHion5, (Accessed May 12, 2021)

[13] *What is Training Stress Score?*, https://support.garmin.com/en-US/?faq=9EOIDzMcjx7kFqTJUGqHj5, (Accessed May 12, 2021)

[14] *What Is Intensity Factor on My Garmin Fitness Device?*, https://support.garmin.com/en-US/?faq=lpxtpg6jEh6Hcdxzxcffe5, (Accessed May 12, 2021)

[15] *How is Balance Calculated With Garmin Power Meter Pedals?*, https://support.garmin.com/en-US/?faq=pzTWzYwC6z9ZMqPC432Qj8, (Accessed May 12, 2021)

[16] *What is the Vertical Oscillation Measurement on Garmin Fitness Watches?*, https://support.garmin.com/en-US/?faq=BxJKzDA0bG4PpkPwwiK2d8, (Accessed May 12, 2021)

[17] *What is Vertical Ratio Measurement on Garmin Fitness Watches?*, https://support.garmin.com/en-US/?faq=KNC51xctfF5vL2DrDVVio9, (Accessed May 12, 2021)

[18] *About Training Effect*, https://www8.garmin.com/manuals/webhelp/forerunner935/EN-US/GUID-7275629E-743A-4658-A284-C84F42A66AE5.html, (Accessed May 12, 2021)

[19] *What is Torque Effectiveness and How Is It Calculated by Garmin Power Meter Pedals?*, https://support.garmin.com/en-US/?faq=y2aLFo6Clu1PNvHK6eTm98, (Accessed May 12, 2021)

[20] *Pedal Smoothness and how it is Calculated by Vector Power Meter Pedals*, https://support.garmin.com/nb-NO/?faq=G41ZZf19cK4OhEM7XVkrc8, (Accessed May 12, 2021)

[21] *Activity Tracking*, https://www8.garmin.com/manuals/webhelp/forerunner935/EN-US/GUID-7A6BAD22-ACC7-451C-B40A-7D92B22A10AF.html, (Accessed May 14, 2021)

[22] *Kort om NEEDED*, http://www.nordsjorittet.no/needed/vis/Kort-om-NEEDED, (Accessed May 10, 2021)

[23] *Open Source*, https://www.reportlab.com/dev/opensource/, (Accessed May 12, 2021)

[24] *Time series / date functionality*, https://pandas.pydata.org/pandas-docs/stable/user_guide/timeseries.html, (Accessed May 13, 2021)

[25] *A4 in millimeters (cm)*, https://a-size.com/a4-paper-size/, (Accessed Apr. 17, 2021)

[26] *Improved ReportLab recipe for "page x of y" (Python recipe)*, https://code.activestate.com/recipes/576832/, (Accessed Apr. 17, 2021)