Analysis of the quality of electrodermal activity and heart rate data recorded in daily life over a period of one week with an E4 wristband

N.M. Enewoldsen

s1481436

Enschede, June 2016

Bachelor Thesis

Faculty of Behavioral Science

Human Factors & Engineering Psychology

1st Supervisor: Dr. Matthijs Noordzij

2nd Supervisor: Dr. Marcel Pieterse

3rd Supervisor: MSc. Erika van Lier

UNIVERSITY OF TWENTE.

Table of Contents

Abstract

Wearable technology allows the monitoring of several psychophysiological responses, like electrodermal activity (EDA) and heart rate (HR), in real-time and in daily life over a period of days and weeks. This paper serves as an introduction to a new approach which tries to warn alcoholics based on their psychophysiological responses, in order to prevent relapse. However, the first step is to determine the quality of the recorded data which is discussed in the present paper regarding the amount of data which is artifact-free and can be used for further analysis. An experiment was conducted with eight participants with a heavy drinking behavior over the period of one week. They wore the E4 wristband from Empatica which is capable of measuring EDA, HR, inter-beat interval (IBI) and blood volume pulse (BVP) among others. Several analyses were applied including standard analyses via Matlab, as well as self-programmed analysis-programs. The results state a high quality of the EDA data, with around 90% of the data detected as clean signal and is thus suitable for a further analysis including drawing reliable conclusions from it. Regarding the HR, only 2% of the data was determined as possible artifacts, in contrast to the IBI data where only 15% of the possible maximum values could be measured. This great difference can be explained by the BVP measurements which are used to compute the HR and the IBI, whereby the HR is always computed or estimated and the IBI only on the basis of sufficient measurements. However, it is still possible to use the HR data, but it is important only to use fractions of the data and never base a decision solely on the HR or IBI data.

Keywords: *wearable sensor, quality analysis, electrodermal activity (EDA), heart rate (HR)*

Introduction

Through the extensive progress in technology in the recent years, it is possible these days, to transform a stationary measurement tool into a small, mobile device: a wearable system. These systems come in different shapes and forms and with different sensors. They are for example available in forms of wristbands (Poh, Swenson & Picard, 2010), watches (Poon, Wong & Zhang, 2006) or t-shirts (Gu et al., 2009). Wearable technologies offer a practical solution, to monitor a person over a longer period of time with a certain quality of the gathered data (Bonato, 2003) and of course outside a medical facility (Picard & Healey, 1997). These wearable devices are mostly non-obtrusive devices that allow measurements of various types of psychophysiological responses, like for example the heart rate or the activity of the sweat glands through measurements of the skin conductance (Bonato, 2003). Through long-term measurements of individuals in their home environment with multiple sensors, it is possible to describe the state of the user in a broader way, due to the increase in sensors and the different environment, and with a higher reliability than with short-term measurements in the laboratory (Ouwerkerk et al., 2013; Scanaill et al., 2006). Furthermore, it is possible to monitor the physical, as well as the emotional state of a user automatically, simultaneously and continuously during normal daily activities at home or outdoor in real-time (Bonato, 2003; Fletcher et al., 2011). In comparison, Poh, Swenson and Picard (2010) suggested only six years ago that there is a need for a sensor that „not only is low cost, compact, and unobtrusive, but also comfortable to wear and non-stigmatizing to the user“ (p.2). Six years later, these sensors are available. An example of such a device that is able to measure changes in the psychophysiological responses through a set of sensors is the E4 wristband from Empatica, a wrist-worn and wireless sensor which will be discussed later in more detail. These wearable devices provide several new opportunities, enabling the user of such a device to share the information with a doctor or a physician to treat problems, or help the user to make a decision based on the gathered data (Picard & Healey, 1997). Nevertheless, the E4 is not a medical device, but more a tool which can be used for individual treatment or research. In comparison to a stationary recording of a medical device like an electroencephalogram (ECG), the E4 cannot keep up with the high quality of the ECG. However, the wearable biosensors provide the opportunity to learn and gain more insight into the affective patterns, thus the psychophysiological responses of humans in a natural environment. Measuring these physiological parameters in daily life can, for example, provide useful information for health informatics which can then be used to detect, prevent or treat diseases (Wilard et al., 2011; Zheng et al., 2014; Cook, Togni, Schaub, Wenaweser & Hess, 2006). In order to make these reliable statements and decisions based on the recorded data, it is important to determine the quality of the data first. In particular, the quality of the recorded data is affected by artifacts which decrease the quality of the data if not removed. The different sources of artifacts and their specific implications for the quality of the data are described in more detail later in this paper. The aim of this study is to determine the amount of data which has a high quality and can thus be used for further analyses in order to make such decisions, with the focus on two different psychophysiological responses: the heart rate and the conductivity of the skin.

In the last few years, there has been a growing interest in these psychophysiological responses of humans in combination with alcohol, in particular, the effects of alcohol on these responses, whether consumed or just perceived. Thereby, a new approach was introduced which focuses on the psychophysiological responses related to relapse and craving, in order to gain more insight into these processes. Alcoholics often fail to recognize the early relapse precursors which make it impossible for them to apply their learned relapse prevention techniques (Tiffany, 1999). In order to solve this problem, or at least to support the identification of these relapse precursors, it was proposed to warn alcoholics based on their measured psychophysiological responses which could inform the user directly about psychophysiological responses which are related to the relapse precursors. Although several studies have indicated that these variables are strongly correlated, little attention has been given to the actual quality of the gathered data and especially the data which is recorded through a wearable device in the real-world over a longer period of time. The objective of this paper is thus to determine the quality of such a wearable biosensor, specifically for the electrodermal activity and the heart rate measured with the E4 wristband over a longer period. This includes their underlying concepts and components which will be discussed in more detail in the following.

In general, these measured physiological states of arousal are coordinated by a balance of activity of the two subsystems of the autonomic nervous system (ANS), the parasympathetic (PNS) and the sympathetic nervous system (SNS) (Poh, Swenson & Picard, 2010). Thereby, the PNS is responsible for the restoration and conservation of bodily energy, whereas the SNS increases the metabolic output to deal with external challenges and mobilizes the body system during activity. In preparation for motor action, the sympathetic arousal elevates the heart rate, blood pressure, sweating and redirects blood toward skeletal muscles, lungs, the heart and the brain, which are also activated when the body is in a fight or flight state (Poh, Swenson & Picard, 2010).

Electrodermal Activity

Electrodermal activity (also called: galvanic skin response; EDA) reflects generalized changes in the state of arousal, whereby these can be caused by emotional, cognitive or physical stimulation (Picard, 2009). One way of measuring EDA is through measuring the skin conductance (SC) which is widely used in psychophysiology as an expression of physiological or psychological arousal because of its connection with the SNS (Chen et al., 2015). It is measured by using an exosomatic method, thus through applying an external low voltage to the skin which then measures the SC (Boucsein, 2012; Fowles et al., 1981). Thereby, the SC measures the SNS which controls sudomotor innervation, the primary source for changes in SC (Chen et al., 2015; Lykken & Venables, 1971). In particular, SC can be described by two components: the skin conductance level (SCL) and the skin conductance response (SCR). The SCL describes slowly varying tonic activity and variations in EDA are more in the order of minutes (Benedek & Kaernbach, 2010). In contrast, the SCR characterizes fast varying phasic activity which can reflect a stimulus-specific response or, without an external event, a non-specific response. These changes in EDA are more in the order of seconds and can be coupled with clearly identifiable external events which arise in a predefined window, mostly some seconds after the stimulus onset (Benedek & Kaernbach, 2010; Kappeler-Setz, Gravenhorst, Schumm, Arnrich & Tröster, 2013). In general, there is a wide range of differences in SCLs and SCRs between individuals, but the basic principle is that with increased sweating, the SCL will also increase (Lykken & Venables, 1971). Thereby, the user does not have to subjectively experience sweating, in order for a sensor to detect differences in the SC because the resistance decreases as sweat rises in a gland, although the sweat may not attain the surface of the skin (Stern, Ray & Quigley, 2001). Van Dooren, de Vries and Janssen (2012) compared 16 different SC measurement locations and found out that the fingers, foot, and forehead offer the highest SC. Although the wrist is not under the top three, it is still present and measurable, but certainly a bit weaker.

In general, physiological signals can be thought of „as a finite time series for purposes of representation and analysis“ (Cacioppo & Tassinary, 1990, p.17). The amplitude variations over time form a waveform which characterizes the psychophysiological responses in a specific time domain (Cacioppo & Tassinary, 1990). The quality of the amplitude of the SC and the SCR, thus the quality of the gathered EDA depends on the following factors: the density of the sweat glands in the chosen skin area, the degree of the psychophysiological activity in this area and the size of the skin that has contact with the electrodes (Lykken & Venables, 1971). In order to identify the quality of a SCR, it is necessary to define its components first. A SCR consists of four main components: a latency, an amplitude, a rise time and a half recovery time (Ishchenko & Shev'ev, 1989; Setz, Arnich, Schumm, La Marca & Tröster, 2010). The latency refers to the time between the onset of the stimulus and the start of a SCR which is typically about 1 to 3 seconds (Dawson, Schell & Filion, 2007; Kappeler-Setz et al., 2013). In contrast to event-related SCRs, non-specific SCRs occur roughly 1 to 3 times per minute. The rise time describes the time between the onset of the SCR and its peak amplitude which also takes about 1 to 3 seconds. In order to be identified as a SCR, the deflection has to pass a certain threshold which is usually around 0.04µS, 0.03µS or 0.01µS and is known as the minimum amplitude threshold. Deflections below this threshold criteria are not counted as SCRs. The amplitude refers to the difference between the conductivity at the onset which is also referred as the baseline, and the peak whereby a phasic increase in conduction occurs which is around 0.2µS and 1.0µS (Dawson, Schell & Filion, 2007). These variations of the SCR's amplitude are due to differences in individual tonic levels of SC. Typically, the SCRs usually follow a characteristic pattern of an initial, relatively steep rise with a short peak and a relatively slower return to the baseline.

The quality of the data is affected by artifacts which have to be detected and removed. In total, there are two main sources of artifacts: artifacts derived through the procedure itself and artifacts derived from motion, so-called motion artifacts (Chen et al., 2015). Regarding the recording procedure, it is necessary to emphasize that also incorrect use of the technology and incomplete knowledge of the technology can cause a reduction in quality and interference (Cacioppo & Tassinary, 1990). This can cause physiological signals that appear to be mute regarding the psychological or behavioral response because the important information is masked by the parameters or the statistical analysis executed on these parameters. However, they can be avoided through an accurate control of the measurement technique which includes, for example, knowing how to wear a sensor properly or how to preprocess the data correctly, in order to be able to perform an analysis (Chen et al., 2015). Motion artifacts are generated through body or skin movements below the electrodes or below the skin and are mostly characterized by a high, overlaying frequency in the recorded data. In a study of Poh, Swenson and Picard (2010) where they were able to perform one of the first long-term EDA measurements, they observed motion artifacts only when the electrode-skin interface was disturbed, an external pressure was applied to the electrodes or the position of the electrodes was readjusted. In contrast to laboratory conditions, the natural environment provides many factors like for example diet or physical activity which have an influence on the recording and have a greater risk to obscure the measured psychophysiological responses (Picard & Healey, 1997).

Heart Rate

The cardiovascular system (CVS) is an organ system that controls the blood flow through the body (Mandryk, Inkpen & Calvert, 2006). The activity of this system can be described and measured by the heart rate variability (HRV), the heart rate (HR), the inter-beat interval (IBI), the blood pressure and the blood volume pulse (BVP). The HR responds to physiological perturbations like for example stress which are mediated by the PNS and the SNS (Appel, Berger, Saul, Smith & Cohen, 1989). According to Furlan et al. (1990), the greatest change of HR is between day and night, indicated by a sympathetic predominance during the day and a parasympathetic predominance during night. The E4 wristband which is used in the present study can record the BVP, through which the IBI and finally the HR can be derived. Thereby, a photoplethysmographic (PPG) sensor is used which „involves a light source to emit light into tissue and a photo-detector to collect light reflected from or transmitted through the tissue“ (Zheng et al., 2014, p. 1540; Fusco, Locatelli, Onorati, Durelli & Santambrogio, 2015). This method is widely used for HR measurements, whereby an LED is used to detect the amount of blood which is flowing through the vessels (Picard & Healey, 1997). The pulsatile component of the PPG sensor is synchronous with the beating heart and can, therefore, be used to determine the HR (Allen, 2007). Throughout the whole recording, the sensing unit has to have direct contact with the skin which indicates that it is susceptible to artifacts, especially motion artifacts which have a great negative influence on the quality of the data (Zheng et al., 2014). Furthermore, the PPG sensor, as well as the sensor which detects the EDA, are sensitive to the correct placement of the sensor, movements and cardiac arrhythmia which influence the rate parameter and can reduce its reliability (Allen, 2007; Picard & Healey, 1997). The measurements of the PPG sensor are further influenced by factors as emotional states, ambient temperature and fitness level, and responds and re-stabilizes only slowly to them (Picard & Healey, 1997; Tapia et al., 2007).

Psychophysiological Responses to Alcohol

Much research investigated the effects of alcohol on the psychophysiological responses whereby the main focus was on the time before consuming alcohol and while consuming alcohol. Regarding the time before alcohol was consumed, previous studies focused on the measurements of psychophysiological responses to alcohol cue exposure. Studies where alcoholics and non-alcoholics were exposed to alcohol cues (high-risk images & alcohol beverages), demonstrated an increase in HR, SCL, pulse rate, blood pressure, skin temperature and the self-reported desire to drink alcohol by alcoholics, in contrast to the exposure to a non-alcoholic beverage or in comparison with the control groups (Carter & Tiffany, 1999; Kaplan, Meyer & Stroebel, 1983; Payne et al., 1992; Sinha et al., 2003; Sinha et al., 2009). Even the information that the participants would receive alcohol was associated with a higher SCR, although they did not receive alcohol at this point (Laberg, 1986). Besides the exposure to external alcohol-related cues, internal alcohol-related cues were also found to increase drug craving and physiological arousal (Carter & Tiffany, 1999). Research focusing on the monitoring of psychophysiological responses while drinking alcohol can be summarized by the findings of Laberg and Ellertsen (1987) and Tiffany, Carter and Singleton (2000) who documented a significant increase in HR and SC parameters. However, psychophysiological responses while consuming alcohol have been scarcely investigated, especially over a longer period of time and without specific alcohol-cues.

Based on the new approach described earlier, the aim of this study is to provide more insight on the quality of the EDA and HR data gathered by the E4 wristband. Thereby, the main interest is the determination of the amount of data which can be used for further analysis and therefore is artifact-free. Furthermore, the amount of SCRs derived through the EDA signal is analyzed through different analyses and compared regarding the time before the consumption of alcohol and the time while alcohol is consumed. In particular, the following two research questions were formulated. How much (percentage) of the recorded psychophysiological data (heart rate and skin conductance) with the E4 wristband is free of artifacts and can be used for further analysis? And how many SCRs are detected in the recorded data with the E4 wristband, before and while consuming alcohol in the real-world?

Method

Participants

In total, eight participants between 19 and 24 years (M = 21.5, SD = 1.77) took part in the experiment over a period of two weeks, whereby four participants participated per week. In the first week, three men and one woman participated and in the second week, four women attended. They were all undergraduate students at a university and were selected through convenience sampling. Participants had to meet one of the following conditions which were based on the current dutch guidelines regarding the use of alcohol, in order to discriminate between moderate and heavier drinking behavior (Health Council of the Netherlands, 2015): men had either to drink on average 21 or more glasses of alcohol per week or had to drink six or more glasses on four or more different occasions per month, whereas women had to drink either 14 or more glasses of alcohol per week or drink six or more glasses on two or more occasions per month. Furthermore, they had to have a minimal age of 18 and had to be capable of understanding and speaking the dutch language. It was also requested that the participants did not take part in any current intervention aiming at reducing the drinking behavior of alcohol, as well as no other addiction to drugs (alcohol and nicotine addiction were allowed). Therefore, an ethical request was proposed to the ethical commission of the University of Twente and was approved. For the participants, an informed consent was provided where they confirmed their participation and accepted the procedural implications through a signature (see Appendix E).

Materials

A questionnaire about the alcohol use was used to identify if a participant met one of the criteria's regarding their alcohol-drinking behavior (see Appendix A). The first part of the questionnaire consists of questions about their drinking behavior of the last half year, whereby participants had seven to eight different potential answer categories. By the second part, participants had to answer yes or no to the DSM-V criteria for alcohol abuse.

The biosensor used in the present study was the E4 wristband from Empatica. The E4 is a wearable device designed for continuous, real-time data acquisition in daily life. The device is able to measure several different psychophysiological responses of the body. In particular, the E4 has 4 sensors: PPG sensor, EDA sensor, 3-axis accelerometer and a temperature sensor. The focus of the present study was the SC and the HR which were measured through an EDA sensor and a PPG sensor. Thereby, the PPG sensor measured the BVP from which the IBI and the HR were derived and the EDA sensor measured, as explained earlier, the arousal of the SNS. The sensor also includes an internal real-time clock and a function to set event markers which were used to indicate the beginning of the alcohol consumption by the participants. Furthermore, the E4 is able to work for over 36 hours with a capacity of over 60 hours of data storage. The data is stored in the internal memory of the E4 and can be downloaded via USB through the software Empatica Manager. Empatica Connect, a web-application, is then needed to download the raw data in CSV-format. Further information over the sensor can be found on the website of Empatica (www.empatica.com/e4-wristband).

Design

In order to analyze the data gathered through the E4 wristband regarding their quality, the EDA and HR were recorded whereby the HR consists of measurements of the BVP, the IBI, and the actual HR. An observational design was used, where participants were observed indirectly in their daily life through a biosensor. The independent variable thereby was the drinking behavior of the participants and the dependent variable were the psychophysiological responses of the bodies.

Procedure

The data-acquisition started with an instructions-meeting for the participants (see Appendix B for time-schedule). In the beginning, participants were welcomed and thanked for their participation which was followed by a short explanation of the different topics of the instruction day. This included the order of the topics, as well as the note that there was always extra room planned in for questions on each topic. The aim of the study and its function as explorative research in this new approach to prevent relapse were explained in detail. The participants were then asked to fill in the questionnaire over their alcohol behavior and the DSM-V criteria's (see Appendix A) which was followed by the download of the empatica manager. The installation of the empatica manager was done individually whereby each participant received their own individual and anonymous access data in order to log-in at the empatica manager. Thereby, the accounts for each participant were created before the instruction day to save time and to have direct access to the accounts without asking participants for their personal access data. Each account and email address were named anonymously, in order to protect the data and the accounts of the participants. An explanation of how to use the sensor, how to upload the data and general information about the sensor followed and of course participants received this information in the form of a handout (see Appendix C). After this, the week-plan was discussed together with the participants (see Appendix D). During the week, they had to wear the E4 every day for a period of one week, except at night when they were sleeping. Furthermore, they had to charge the E4 every night and upload the recorded data at least every two days. They were also asked to mark the beginning of the consumption of alcohol through setting an event marker, in order to detect this event later. Finally, the informed consents were handed out and the participants were asked to sign them (see Appendix E). It was clearly explained that all data of the participants were used confidential and anonymous and that participants could stop anytime with the experiment. Then appointments were made to give back the sensor and participants were thanked again for the participation.

Data Analysis

For the data analysis, many different software and programs were used whereby several of them were written by the researcher self. These programs were written in Python and can be used for the preparation and transformation of the data for analyses, as well as for different aspects of the analyses. The source-code with a more detailed description of the program itself and the potential use of it can be found in the appendix (see Appendix F - O).

In a first step, the files of the E4 were downloaded and renamed, in order to merge them for further analysis. Therefore, a program was written that merges the individual files of the sessions into one big file per participant (see Appendix F). The program also provides information on the start-time, end-time, and duration of the individual files within the merged file and is compatible with every recorded data-type of the E4 wristband. For the analysis of the EDA data, several analyses were conducted. First, a trough-to-peak (TTP) analysis was conducted with Ledalab, a program for Matlab which is also recommended on the website of Empatica for the signal processing and data analysis („Recommended tools“, 2015). Therefore, the data had to be down-sampled from 4 Hz to 1 Hz. A program was written which analyzes the output of the TTP analysis and calculates SCRs per minute (see Appendix G). For the visualization of the frequency of SCRs per minute, another program was written which deals with the plotting of the data (see Appendix H). Furthermore, a continuous decomposition analysis (CDA) was conducted, also with Ledalab. This method extracts the phasic information of the EDA signal and provides information on the signal characteristics (Benedek & Kaernbach, 2010). To increase the temporal precision, the data is deconvolved by the response shape and finally decomposed into its phasic and tonic components. Ledalab provides information over the amount of SCRs and the SCR-onset. For the CDA, the data was down-sampled from 4 Hz to 1 Hz. Then, the program which computes the SCRs per minute was used, in order to compare the results with the TTP analysis (see Appendix G) which was followed by the plotting of the frequency of SCRs per minute (see Appendix H). The final analysis in Ledalab was the event-related analysis, whereby the event markers were used which indicated the moment when the participants began to consume alcohol. The markers were computed in order to analyze the minute before the start of the alcohol consumption and the first minute of alcohol consumption. Furthermore, 10 pseudo-random events per participant were generated to compare the results of the minute before and the first minute of consuming alcohol. The event-related analysis was carried out with measures of the TTP analysis, as well as with the CDA which made it possible to compute an average amount of SCRs per minute for both analyses.

After the analysis in Ledalab, an additional EDA analysis was conducted. In particular, the online version of the EDA-Explorer was used to perform an automatic artifact and peak detection (Taylor et al., 2015). Since the online version can only deal with sessions with a maximal length of six to seven hours, a program was written that splits the merged files into equal files with a maximal length of seven hours (see Appendix I). The artifact detection filtered the data into three different categories: clean data, questionable data and signals which were detected as noise. Therefore, the temperature and accelerator data were also taken into account. A program was written in order to merge the result files of both analyses into one file per participant for further analysis (see Appendix J). For each of the analyses of the EDA-Explorer (peak- & artifact-detection), programs were written to analyze and summarize the result files (see Appendix K & Appendix L). By the noise analysis, this included the distribution of the signals over the three categories and their percentages whereby for the peak analysis, averages, minimums and maximums were computed for the essential parts of the EDA signal, like the amplitude or the rise-time. Again, this included the calculation of the SCRs per minute for the peak analysis. In the following step, the amount of SCRs per minute was compared between the three EDA analyses which dealt with the analysis of the SCRs.

For the analysis of the HR data regarding their quality, the HR data, the IBI data, and the BVP data were analyzed. For the HR data, a program was written that analyzes the distribution of the data and is able to detect irregularities within the data (see Appendix M). This includes the detection of unrealistic values of the heartbeat measurements over 200 and under 40 beats per minute, as well as the detection of signal sections where the HR changes over three beats per minute in a second. A program with the same purpose, the detection of possible artifacts and the representation of the distribution was written for the IBI data (see Appendix N). Besides the HR and the IBI data, the measurements of the BVP were analyzed because the HR and the IBI data is computed out of the measurements of the PPG sensor which primarily measures the BVP. A program was written to show the distribution of the file with an additional detection of outliers (see Appendix O). Finally, figures of the HR, IBI and

BVP data files per participant were computed with Matlab in order to look more closely at the values and sections the programs provided.

Results

Electrodermal Activity

Noise analysis. A noise analysis was conducted with the web-application of the EDA-Explorer which took also the temperature and the accelerator measurements of the E4 into account, in order to detect artifacts. Therefore, the data was categorized in three categories: clean signals, questionable signals, and noise. Figure 1 shows a visualization of an excerpt from the noise analysis, whereby detected artifacts are labeled red and questionable signals gray.

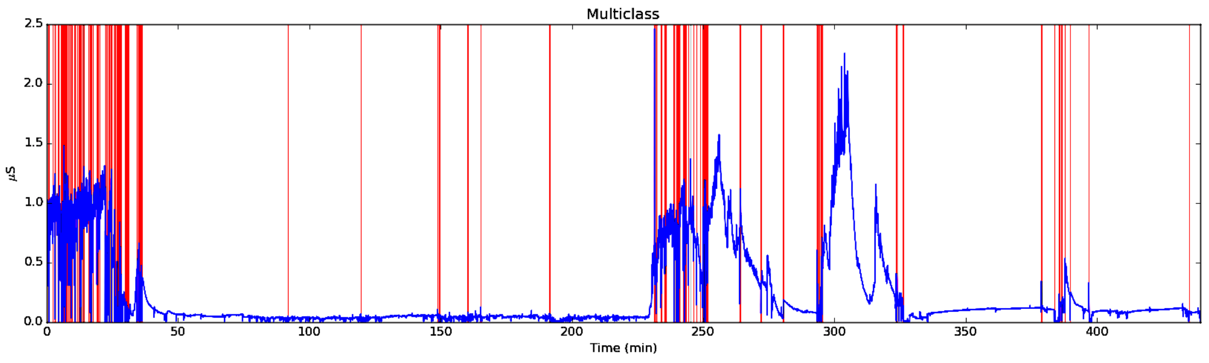


Figure 1. Visualization of an excerpt from the noise analysis conducted with the web-application of the EDA-Explorer.

Overall, around 90% of the EDA signals were clean (M = 90.12%, SD = 5.43%), whereby the percentages range from 76% up to 94%. In contrast, around 8% of the EDA signals were detected as artifacts (M = 8.16%, SD = 4.85%) and around 2% as questionable signals (M = 1.73%, SD = 1.59%). The worst quality of the EDA data was found by participant 3, by whom a bit less than one-quarter of the data either was detected as noise or as a questionable signal. Regarding the other participants, a maximum of 12% was found with regard to the artifact detection. A figure with the precise percentages per participant can be found in the appendix (see Appendix P).

Trough-to-peak analysis. The trough-to-peak (TTP) analysis was conducted, as described in the data analysis section, with a sample rate of 1 Hz and a minimum amplitude threshold of 0.01µS. The files which were used for the analysis contained recorded EDA data of 35 hours to 121 hours, varying per participant (M = 79h, SD = 25h).

In order to compare the results of the TTP analysis with the other analyses, like the continuous decomposition analysis, the average amount of SCRs per minute was computed. Altogether, the average amount of SCRs per minute over the eight participants was almost five SCRs per minute (M = 4.96, SD 4.37) with a mean amplitude of the SCRs of 0.1548µS. By a comparison between the participants, participant 8 was found to be an outlier with regard to the length of the recording, as well as to the average amount of SCRs per minute. In particular, an average of two SCRs per minute was found by participant 8 who had a total recording duration of a bit more than 35 hours.

Figure 2 illustrates the distribution of the frequency of the SCRs per minute, found through the TTP analysis. Thereby, the frequency reflects the total amount of minutes per certain amount of SCRs per minute. The distribution is positively skewed, showing a peak at zero SCRs per minute with a continuous decrease up to 18 SCRs per minute.

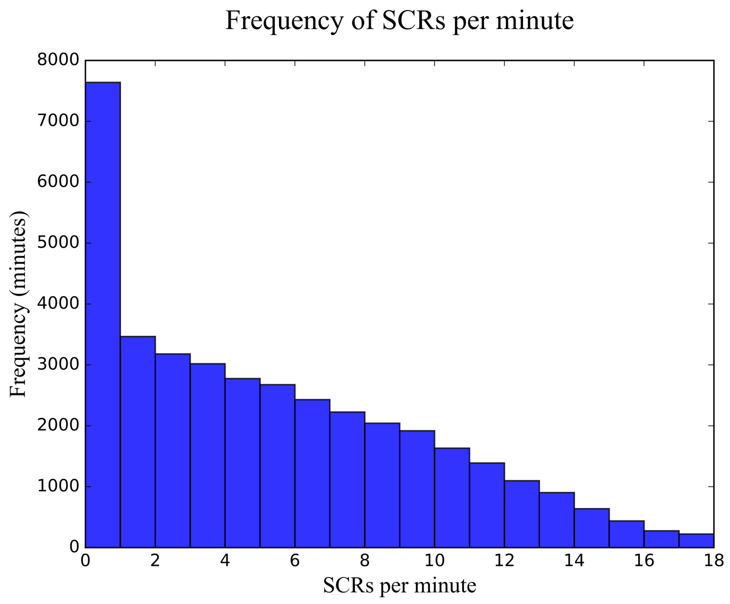


Figure 2. The graph shows the total amount of minutes per certain amount of SCRs per minute, gathered through the TTP analysis.

Peak analysis. The peak analysis was conducted with the web-application of the EDA-Explorer. Thereby, a minimum amplitude threshold of 0.01µS was used with a maximum rise-time of four seconds and an offset-value of 0.8s. Instead of downsampling, in order to preprocess the data, the EDA-Explorer provided a low-pass Butterworth filter which was mandatory to use. It was chosen for a filter frequency of 1 Hz and a filter order of 6. The average amount of SCRs per minute, found through the peak analysis, were close to four SCRs per minute (M = 3.86, SD = 1.16) with a mean amplitude of 0.1521µS. In comparison, the outcome of both peak analyses is almost in the same range, although the TTP analysis included downsampling and the peak analysis included a low-pass filter. The amount of SCRs per minute is one SCR per minute higher for the TTP analysis conducted with Ledalab. With regard to the average amplitude, there is only a difference of 0.0027µS. Although the TTP analysis is also a peak analysis, the different names are used in order to distinguish between the two analyses without naming the programs with which they were conducted. Furthermore, the peak analysis computed several variables of the essential parts of the EDA signal, like for example the rise-time or the decay-time which can be found as averages per participant in the appendix (see Appendix Q). In addition, there are also the peak-extraction-settings which were used for the peak analysis including a description of the variables (see Appendix Q).

Continuous decomposition analysis. The continuous decomposition analysis (CDA) was conducted using Ledalab. As by the TTP analysis, the data was down-sampled to 1 Hz. For the extraction of the SCRs after the CDA, a minimum amplitude threshold of 0.01µS was set. Furthermore, the average amount of SCRs per minute was computed, in order to compare the results with the two peak analyses. The average amount of SCRs per minute for the CDA was almost 7 SCRs per minute (M = 6.99, SD = 1.86). Figure 3 demonstrates the distribution of the frequency of the SCRs per minute, gathered through the CDA. As in Figure 2, the highest amount of SCRs per minute is at zero SCRs per minute, followed by a sharp decline at one SCRs per minute. Furthermore, there is a slight increase by 20 to 24 SCRs per minute which is only clearly visible in the right graph. In comparison with the figure 2, the distribution of the SCRs per minute of the CDA is more stable and stays at the same level whereas the graph in figure two has a steady decline.

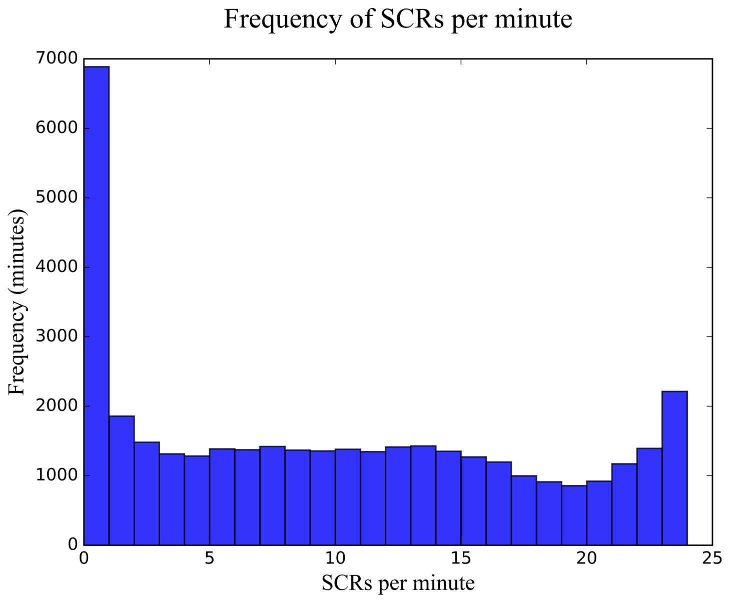


Figure 3. The graph shows the total amount of minutes per certain amount of SCRs per minute, gathered through the CDA analysis.

Table 1 shows the total amount of detected SCRs and the average amount of SCRs per minute for each of the three analyses: peak analysis, TTP analysis and CDA for a direct comparison. The CDA has thereby the highest SCRs per minute with almost 7 SCRs per minute on average, ranging from 4.2 up to 9.3 SCRs per minute. In contrast, the two peak analyses had on average 4 SCRs per minute, respectively 5 SCRs per minute, ranging from 1.4 to 4.7 SCRs per minute for the peak analysis and ranging from 2.2 to 5.6 SCRs per minute for the TTP analysis. In addition, table 1 clearly shows that participant 8 can be labeled as an outlier. In particular, participant 8 has an average amount of SCRs per minute which is twice as low as the second-lowest value regarding all three analyses.

Event-related analysis. Beside the TTP analysis and the CDA, an event-related analysis was carried out in Ledalab. Therefore, the event markers which indicate the beginning of consuming alcohol were transformed and imported in Ledalab. As mentioned in the data analysis section, the markers which only indicated the beginning of alcohol consumption were computed, in order to analyze the minute before the participants started drinking alcohol and the first minute of consuming alcohol. Furthermore, pseudo-random events per participant were generated to compare the results between the three different conditions. A minimum amplitude threshold of 0.01µS with a response window of 1 minute was used. It was chosen for a response window of one minute because, on the one hand, they

Table 1. *Amount of detected SCRs and average amount of SCRs per minute for the three analysis: peak analysis, TTP analysis & CDA*

	Peak-Analysis		TTP-Analysis		CDA	
	total	SCRs/min	total	SCRs/min	total	SCRs/min
Participant 1	13666	3.9	18154	5.2	29795	8.5
Participant 2	25014	4.9	27808	5.5	43480	8.6
Participant 3	21534	4.2	27517	5.4	41509	8.1
Participant 4	16190	3.1	20582	4.0	53542	5.2
Participant 5	33741	4.6	40832	5.6	85999	5.9
Participant 6	20844	4.7	24996	5.6	41699	9.3
Participant 7	21460	4.1	23793	4.45	64938	6.1
Participant 8	2883	1.4	4942	2.2	8968	4.2
mean	3.9 SCRs/min		5.0 SCRs/min		7.0 SCRs/min	

Note: Each analysis was carried out with a sample rate of 1 Hz. The exported SCRs had a minimum amplitude threshold of 0.01µS. All values presented in this table are rounded up to one decimal after the comma.

just indicated the begin of drinking and to be sure that they actually drank something, a response window of several seconds would be too short. On the other hand, this way enabled the computation of the average amount of SCRs per minute and therefore the comparison with the results of the individual analyses, as well as with the event-related analysis. Table 2 shows the average, minimum and maximum SCRs per minute for the three conditions: before, while and random, with measures of the TTP analysis and the CDA. A detailed table with values of each participant for the three conditions can be found in the appendix (see Appendix R).

As table 2 presents, the average amount of SCRs per minute for the minute before drinking alcohol and the first minute of drinking alcohol is almost the same. For the event-related analysis with the TTP-measures these were 7.9 and 8 SCRs per minute, whereas, for the event-related analysis with the CDA-measures, the average amount is 11 and 11.4 SCRs per minute. In comparison with the random events, the average amount of the SCRs per minute is almost 3 SCRs per minute higher for the conditions of before and while alcohol consumption. Furthermore, the comparison with the total average of SCRs per minute gathered through the analyses carried out earlier, reveals the same effect of increase. The total average amounts of SCRs per minute are actually almost 1 SCR per minute smaller than the values computed from the pseudo-random markers. The increase in the comparison of the total average of SCRs per minute and the alcohol conditions is then by almost 4 SCRs per minute regarding both analyses which is almost twice as high as the total average amount of SCRs per minute. However, only a small difference can be seen regarding the minute before and the first minute of drinking alcohol. For the TTP analysis this difference was found to be 0.1 SCRs per minute, whereby for the CDA, the difference was a bit greater with a difference of 0.4 SCRs per minute.

Table 2. *Total average, minimum and maximum SCRs per minute of the event-related analysis with TTP and CDA measures for the minute before the participants started drinking, the first minute of consuming alcohol and for 10 individual random minutes per participant*

			TTP			CDA	
	mean events	mean	min	max	mean	min	max
before	7.6	7.9 SCRs/ min	0 SCRs/ min	21 SCRs/ min	11.0 SCRs/ min	0 SCRs/ min	24 SCRs/ min
while	7.6	8.0 SCRs/ min	0 SCRs/ min	19 SCRs/ min	11.4 SCRs/ min	0 SCRs/ min	24 SCRs/ min
rando m	10	5.6 SCRs/ min	0 SCRs/ min	20 SCRs/ min	8.5 SCRs/min	0 SCRs/ min	24 SCRs/ min

Note: Each analysis was carried out with a sample rate of 1 Hz whereby the event-related SCRs had a minimum amplitude threshold of 0.01µS and a response-window of 1 minute. All values presented in this table are rounded up to one decimal after the comma.

Heart Rate

For the analysis of the HR data, three different data-types were analyzed: HR, IBI, and BVP. Thereby, the HR data, as well as the IBI data is calculated from the BVP. If values are missing or are found to be wrong, the algorithm of the E4 estimates these values for the HR data, although these were not actually measured and exclude these values from the IBI data. Therefore, each of the three data types will shortly be analyzed regarding their distribution and possible artifacts.

Heart rate. By the HR data, almost 80% of the measurements are between 60bpm and 100bpm (M = 79.67, SD = 6.22). The HR data consisted also of a few values which could probably artifacts. However, these values account only for less than two percent of the data. For example, 0.03% of the values are over 200bpm and therefore unrealistic measurements.

Inter-beat interval. By the IBI data, around 98% is ranged from 0.5s to 1.4s whereby the remaining two percent of the data has values smaller than 0.5s. Instead of missing values or unrealistic values, there are whole IBI data sets which were empty. Actually, there were 10 empty files, accounting for almost six hours of recording, although these were most of all small files (M = 34.4min, SD = 69.2min). In total, these files correspond to 1% of the total recording length. However, inspecting all of the IBI files made clear that the majority of the IBI files had very few measurements, in comparison to the other data-types. In order to get the approximate percentage of actual IBI measurements, the maximum of possible IBI measurements was compared with the actual measurements. An analysis of the total measurements which included all data files except the IBI file revealed that the total duration of the recordings was about 2281000s, varying per sensor. For reasons of simplicity, the total duration of 2281000s is used for the further analysis. The average value of the IBI was 0.76s which suggests a theoretical maximum of over three million IBI measurements. In fact, only 443126 measurements were detected which would represent a percentage of less than 15%, compared to the theoretically maximum of measurements.

Blood volume pulse. Regarding the BVP, around 80% of measurements are between -100 and 100. Thereby, the other 20% are almost equal larger and smaller than the majority, with values greater than 300, respectively smaller than -300. When the data is plotted, interesting irregularities can be seen. The upper graph of figure 4 shows the BVP of participant 3 where three to four gaps within the data can be seen. By a closer look at these gaps, it can be seen that the BVP has values around +100 there. In particular, these gaps in the BVP measurements account in total for over a whole day of measurements. These values combined with the graph beneath the BVP graph in figure 4 which shows the HR data, support the assumption that these gaps are actually artifacts. By the HR data, no gaps can be seen, but therefore, it is clearly visible where the values were estimated. At the location where the gaps are in the BVP file, noticeable high values in the HR file ranging from 100 up to

200bpm can be seen. These overestimated values are furthermore solely present in the HR data where the gaps are located in the BVP data.

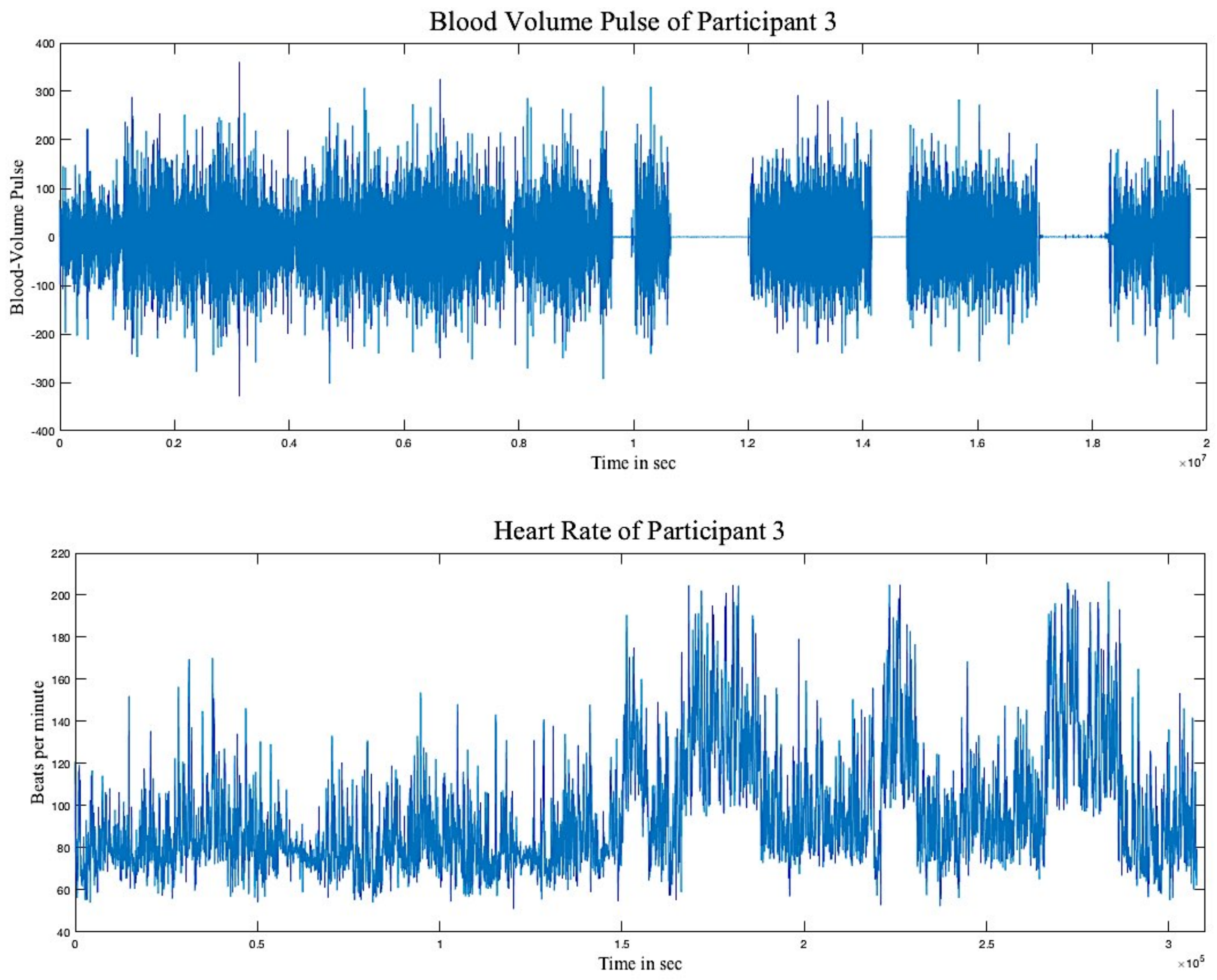


Figure 4. The upper graph shows the BVP measurements and the graph below the HR measurements of a whole week of Participant 3, and shows a good example of the value estimation for the HR data even though there is no actual measured BVP data.

Discussion

The main aim of the present study was to determine the amount of data, recorded over a week through the E4, which is free of artifacts and can be used for further analysis. The focus was thereby on the EDA and the HR data which can be divided into EDA data, HR data, IBI data and BVP data. The results demonstrate a relatively high quality of the gathered data which will be discussed in the following in more detail.

Summarized, around 90% of the EDA data was detected as a clean signal with a consistent amount of SCRs per minute. Regarding the peak analyses and the CDA, the average amount of SCRs per minute ranges from 4 and 5 SCRs per minute by the peak analyses up to almost 7 SCRs per minute by the CDA. Specifically, it was also possible to detect a difference in the amount of SCRs per minute regarding the minute before and the first minute of drinking which shows that there is an increase of almost 3 to 4 SCRs per minute in comparison with the rest of the recorded data.

Concerning the analysis of the HR data, around 2% of the measurements of the HR and IBI data were identified as unrealistic or possible artifacts. The rest of the HR and IBI data were in an acceptable range of values. However, there was only a small amount of IBI data which only accounts for less than 15% of the possible maximum of IBI measurements. Regarding the BVP data, around 80% of the data was in an acceptable range. Despite this high amount of clean data, artifacts could be identified through a visualization of the data.

Electrodermal Activity

For the EDA data, the amount of artifact-free data was clearly shown through the noise analysis which stated that on average, 90% of the data was clean and can thus be used for further analysis. Furthermore, the analysis was able to detect around 8% of the recorded data as artifacts which is relatively less, considering that the two electrodes require permanent contact with the skin, in order to measure the SC correctly, and are therefore very sensible to motion. These findings demonstrate a more than good quality of the EDA data, especially in the combination with long-term measurements. Although no explicit percentages of artifact-free EDA recordings were named in the literature, Poh, Swenson and Picard (2010) showed that it is possible to continuously measure EDA during daily activity with relatively artifact-free recordings. However, they did not use the E4 but built their own wrist-worn, wearable device with an EDA sensor.

The two peak analyses including the CDA gave a deeper insight into the amount of SCRs per minute, as well as into the differences between those analyses. In particular, the two peak analyses, the TTP analysis, and the peak analysis, had a difference of 1 SCR per minute regarding the average amount of SCRs per minute. However, 4 SCRs per minute for the peak analysis and 5 SCRs per minute for the TTP analysis are quite close, as well as the average amplitude which only differed three decimals behind the comma. These discrepancies can be explained by the different preprocessing of the data. Although both analyses are peak analyses and both analyses had the same minimum amplitude threshold, the data was down-sampled from 4 Hz to 1 Hz by the TTP analysis and smoothed by a low-pass Butterworth filter by the peak analysis („Documentation“, n.d.; „Information and FAQs“, n.d.). Furthermore, by each analysis, there were different variables which could be determined by the analysis. For example, the TTP analysis in Ledalab enabled the user to down-sample the data and to set a minimum amplitude threshold. In contrast, by the peak analysis from the EDA-Explorer, it was possible to configure the low-pass filter and additional the maximal decay and rise time plus the minimum amplitude threshold. There were also differences regarding the output which also focused on different aspects of the EDA signal. The TTP analysis provided a list of SCRs whereas the peak analyses provided a list with several essential variables of the SCR („Documentation“, n.d.; „Information and FAQs“, n.d.). Despite their different approaches, both analyses were able to demonstrate an adequate analysis of the SCRs and can be used for the analysis of the phasic component of the SCR. For a more specific comparison between the different analyses, the calculation of the specific correlations would have given more insight into the strength of the relationship. This way, it could be determined if the different analyses actually are proportional equal, thus differentiating by, for example, one SCR per minute for each participant which would be reflected through a strong and positive correlation.

The CDA revealed a higher amount of SCRs per minute, as well as a different distribution of the SCRs per minute then gathered through the TTP analysis. Overall, these differences between the CDA and the peak analyses are no cause for concern and are due to the different approaches of these different analyses. As explained briefly in the data analysis section, the standard peak analysis quantifies the amplitude of a given SCR through using the local minimum and the local maximum. This method, although it is a common method, can easily cause an underestimation of the amplitude with regard to a misattribution of the response window. The CDA, in contrast, uses a decomposition approach where the phasic driver of the recorded data is extracted which leads to a separation of the tonic and the phasic data and allows an unbiased estimation of the SCR-magnitude (Benedek & Kaernbach, 2010). According to Benedek and Kaernbach (2010), the EDA scores and results are more sensitive when they are based on a decomposition method, instead of, or as in this study, in addition to the standard TTP analysis. This explains the differences between these analyses, especially the greater amount of SCRs found through the CDA and the differences in the distribution of the SCRs per minute. Through the high sensitivity of the CDA, this analysis is able to detect the SCRs which are undetected by the TTP analysis, leading to a higher amount of detected SCRs per minute. The effect is an equal distribution with fewer minutes containing just one or two SCRs.

In one of their studies, Benedek and Kaernbach (2010) name explicit percentages with regard to the average reduction of the SCR-amplitude caused by an underestimation of the SCR-amplitudes. Benedek and Kaernbach (2010) described that the previously reported range of 15 to 17% underestimation could increase up to 40%, whereby the CDA was not found to be vulnerable to this underestimation. Except for participant 8, this statement is supported by the present study where the average amplitude per participant, gathered through a CDA, differs on average 32% from the average amplitude of a TTP analysis, assuming an underestimation of the amplitudes by the TTP analysis. This would implicate that for the further analyses in this approach, which are explained later in more detail, analyses like the CDA should be preferred over the peak analyses because it provides more accurate and valid information over the phasic component of the EDA signal (Barry, Feldmann, Gordon, Cocker & Rennie, 1993). However, it is not yet possible to conduct the CDA in real-time which is required, in order to inform alcoholics over psychophysiological changes which are related to the relapse precursors. Nevertheless, the results suggest that a development of a CDA which could be carried out in real-time would be a great improvement, in contrast to the peak analyses.

The event-related analysis showed an increase in the average amount of SCRs per minute for the minute before participants started drinking, as well as for the first minute of drinking. In particular, the amount of SCRs per minute increased by 3 SCRs per minute in comparison with the pseudo-random generated events. By a comparison with the total average amount of SCRs per minute, this effect was even greater with an increase of 4 SCRs per minute. These findings of the event-related analysis match the findings in the literature regarding alcohol in combination with the measurement of the psychophysiological responses of the body. As described in the introduction, there are several studies which report changes in the psychophysiological responses when an alcohol-related cue is perceived, before alcohol was actually consumed, as well as while the consumption of alcohol (Carter & Tiffany, 1999; Kaplan, Meyer & Stroebel, 1983; Laberg, 1986; Laberg & Ellertsen, 1987; Payne et al., 1992; Sinha et al., 2003; Sinha et al., 2009; Tiffany, Carter & Singleton, 2000). Although the response window was not that long in duration, it already showed an increase in the SCRs per minute. Unfortunately, it is only possible to relate this to the beginning of the consumption of alcohol, as well as for a specified time before because the markers which indicate the beginning of the consumption of alcohol, were the only references to an explicit action throughout the whole week. However, when comparing the minute before drinking with the first minute of consuming alcohol, only a small difference of 0.1 SCRs per minute for the TTP analysis and 0.4 SCRs per minute for the CDA could be found. These findings imply that there is only a small difference in the amount of SCRs between the actual drinking of alcohol and the intention to consume which is an important implication for future research. As it will be described in more detail at the end of this section, the next step of research in this approach is the actual detection of some sort of craving in the data which also requires the discrimination of the actual drinking and the desire to drink. The findings of the present study demonstrate that this will be more difficult than a rough discrimination between not drinking alcohol and drinking alcohol and requires a sensitive approach in order to able to detect these differences. Although the use of peak analyses did not have a limiting or negative effect in the present study, it should be clear that for some conditions, like the detection of craving in the EDA recordings, the traditional method of the peak analysis might not be sufficient (Barry, Feldmann, Gordon, Cocker & Rennie, 1993).

Heart Rate

The results of the analyses showed a moderate to good quality with only two percents of the HR and IBI data detected as unrealistic values and probable artifacts. However, by the IBI data only around 15% of the possible total measurements were actually measured. Regarding the BVP data, over 80% of the data was in an acceptable range, excluding artifacts which could only be detected through the visualization. For the adequate interpretation of the results, an integration of the underlying algorithms and their way of processing the data is necessary because the results can be deceiving otherwise. As explained earlier, the PPG sensor is used to compute the HR and the IBI. In particular, the PPG sensor uses green light for measuring the heartbeats and red light for the detection of motion which enables the sensor to detect and remove motion artifacts, not only purely related to global movements like movements of the device, but also artifacts related to local movements such as movements under the skin („How is IBI.csv obtained?“, 2015; „What should I know“, 2016). Two algorithms are responsible for the computations of the BVP, the HR, and the IBI scores, as well as for the estimation of HR values and the exclusion of detected artifacts regarding the IBI data. This has the advantage that the E4 is more robust to motion and motion artifacts, but has the disadvantage that it is not clear which of the measurements were actually measured and which measurements were estimated or computed.

With regard to the IBI data, the processing of the algorithm explains the empty files and the small amount of measured IBI data which were excluded due to insufficient measurements of the PPG sensor. At their website, Empatica states that by studies with strong movements which were defined as occurring more than 30% of the time, there is a high probability that the sensor is not able to compute enough reliable IBI data, but that it should be possible to compute the HR („What should I know“, 2016). In particular, they state that the software „aims to detect ‚every‘ heartbeat“ which could lead to useless data in cases of extreme motions. Although this can be seen as a limitation of the quality, the software was able to provide enough HR measurements, even though a certain amount was rather estimated than computed. This is supported by Fukushima, Kawanaka, Bhuiyan and Oguri (2012) who demonstrated an estimation of the HR using a wrist-worn PPG sensor by a subject who was actually running. They were able to achieve a higher degree of usability compared with the existing methods using ECG, whereby they removed artifacts using the accelerator data. In contrast, the E4 uses the secondary light source which is more accurately by irregular movements, in comparison with the approach of Fukushima et al. (2012) (Garbarino, Lai, Bender, Picard & Tognetti, 2014). A direct comparison between the two artifact removal approaches was made by Hernandez, McDuff and Picard (2015) who conducted a validation and a real-life study regarding the estimation of the HR, derived from the light-based BVP measurement of the E3 and the motion sensor of the Samsung Galaxy Gear. The results indicated that the measurements of the light-based PPG sensor were better than the measurements of the motion sensor due to a smaller amount of artifacts, in contrast to the real-life study where the combination of all the sensors yielded the best results. Finally, the findings of Empatica are consistent with the present study with regard to the validity of the HR („Have you done comparative“, 2015). Thereby, measurements of the predecessor model, the E3, were compared with measurements of the Biopac ECG, whereby an error rate of 2% was found by the E3 which reflects miss-detected peaks in every form. In particular, these two percents were also found in the present study, whereby these two percents also reflect unrealistic values and possible artifacts. Overall, these studies provide evidence that it is possible to estimate the HR based on the PPG sensor accurately. However, the problem regarding the algorithm are not the artifacts, but more the uncertainty that it is impossible to differ between estimated and computed values which could be unsatisfactory in some cases, especially when the consequences of certain decisions can have a negative impact on the life of the user which in the case of an early-warning system for alcoholics could be the case.

In contrast to the HR data, the BVP data contained several artifacts which consisted of outliers and „gaps“ and which were shown through the visualization (see Results, Figure 4). These artifacts can be explained by Hayes and Smith (1998) who state that recorded artifacts result in an amplitude of the PPG waveform which is either resting at a random value or reaching a maximum or a minimum value. In the HR data, the artifact is reflected by a high frequency which can increase up to about 170bpm under noisy conditions („Have you done comparative“, 2015). Furthermore, Hayes and Smith (1998) state that a PPG signal which consists of 10% or fewer artifacts represents a clean signal, while an amount of around 20% artifacts contains visible signs of corruption. Specifically, the BVP data in the present study consisted of around 20% of unrealistic values plus the artifacts which were detected through the visualization. It can thus be concluded that although the BVP data contains more than 20% artifacts, it is possible to interpret and use the BVP measurements for further analysis, at least for the computation of the IBI and the HR data. However, the same advices and implications regarding the appropriate use of the HR and IBI data apply also for the BVP data.

The strength of the present study was the used wearable device, the E4 wristband. With this biosensor, it was possible to measure several autonomic variables in real-time and over a longer period. In comparison with other commercial wearable devices, the E4 is able to capture the most different psychophysiological responses and to provide access to the raw data, in order to analyze it (del Pozo et al., 2014). All the other available devices have either fewer sensors or not the possibility to access the raw data which only allows the user to view the recorded data from the graphical user interface, provided by the manufacturer. According to Empatica, the E4 wristband is also the only wearable sensor which combines the EDA and the PPG sensor in the same device („Empatica E4“, 2014). Furthermore, several studies, including the present one, have demonstrated the high quality and validity of the E4 measurements (Fukushima et al., 2012; Garbarino et al., 2014; „Have you done comparative“, 2015; Hernandez, McDuff & Picard, 2015).

In addition, the present study was able to use the wearable sensor over a period of one week which allowed a good imitation of the possible artifacts which could reduce the quality of recordings in daily life. However, regarding the HR, the algorithms used to compute the HR and IBI data allowed only a more superficial analysis of the quality because the number of actual measurements could not be determined. Nevertheless, as described earlier, it is possible to demonstrate a highly accurate HR estimation based on a PPG sensor (Fukushima et al., 2012; Garbarino et al., 2014; Hernandez, McDuff & Picard, 2015.

A possible limitation of the present study were the participants. These were, in fact, no alcoholics, but students with a high to severe drinking behavior. For the most part of the study, this fact was not that important because the main focus was the quality of the gathered data, whereby it was not considerably who the participants were. However, findings of Laberg (1986) who studied the psychophysiological responses to alcohol stimuli in severely, moderately, and non-dependent drinkers, suggest differences regarding the autonomic responses which can have implications for the quality of the data. In particular, Laberg (1986) demonstrated differences between the severely dependent and the moderately dependent group, whereby the severely dependent group displayed a greater SCR and a stronger effect regarding HR, before alcohol was consumed, as well as while the consumption of alcohol. These differences could be reflected in the autonomic responses of alcoholics in from of a higher measured amount of SCRs or an increase in the SCRs per minute regarding the minute before consuming alcohol and the first minute of drinking.

Despite these limitations, the present study showed that the EDA data recorded with the E4 has a really high quality regarding the amount of clean data, as well as the amount of SCRs per minute. This quality allows an accurate analysis of the EDA data as a whole, as well as an analysis of a specific time period. Furthermore, these results can be used to make reliable statements and conclusions. However, it is not clear if this quality is sufficient in order to warn alcoholics, based on these measurements, to prevent them from relapse. As the event-related analysis demonstrated, a sensitive approach is required, in order to be able to distinguish between the actual drinking and the desire to drink based on the recordings.

Regarding the HR, it is possible to use the recorded information for analyses. Thereby, it is necessary that decisions are not solely based on the HR data and that if possible not the whole HR data is taken into account. This minimizes the probability to make wrong decisions or statements and contribute to a more reliable result. In practice, the HR data can, for example, be used to compute the resting HR or to use fractions of the data to provide additional information. This could then be integrated into the analysis of the other sensors, in order to get a broader picture of the situation.

On the basis of the promising findings presented in this paper, future research requires the participation of real alcoholics. In particular, the next stage of research includes the actual detection of craving in the recorded psychophysiological data in some form which implies, as described earlier, the discrimination of actual drinking behavior and the desire to drink. This difference was found to be very small, concluding that a normal peak analysis could probably be not sufficient for the detection of this discrimination. However, the analysis of the EDA data has to be carried out in real-time, in order to warn alcoholics while they are wearing the wearable device. Therefore, it is advised to consider all the different types of autonomic responses, to be able to make this difference clearer. In the case that neither the peak analysis nor the integration of the other psychophysiological responses can lead to a clear detection of craving, it is advised to develop other analyses methods, like for example the CDA, so that it is possible to perform them in real-time. Furthermore, it is necessary to perform this research with real alcoholics because they are most likely to display extreme forms of craving when confronted with alcohol which probably result in a more extreme difference with regard to the psychophysiological responses. It is advised to introduce some kind of control group to compare the recordings regarding the detection of craving, in order to identify and sharpen the related representation of craving in the EDA signal. Besides the control group, more indicators of the activities of the participants are required, especially regarding craving which can then be analyzed as an event-related response and can provide more reliable and explicit results.

References

Allen, J. (2007). Photoplethysmography and its application in clinical physiological measurement. *Physiological Measurement*, Vol. 28(3), doi: 10.1088/0967-3334/28/3/R01

Appel, M.L., Berger, R.D., Saul, J.P., Smith, J.M., & Cohen, R.J. (1989). Beat to beat variability in cardiovascular variables: Noise or music? *Journal of the American College of Cardiology*, Vol. 14(5), 1139-1148, doi: doi: 10.1016/0735-1097(89)90408-7

Barry, R.J., Feldmann, S., Gordon, E., Cocker, K.I., & Rennie, C. (1993). Elicitation and habituation of the electrodermal orienting response in a short interstimulus interval paradigm. *International Journal of Psychophysiology*, Vol. 15(3), 247-253, doi: 10.1016/0167-8760(93)90008-D

Benedek, M, & Kearnbach, C. (2010). A continuous measure of phasic electrodermal activity. *Journal of Neuroscience Methods*, Vol. 190(1), 80-91, doi: 0.1016/j.jneumeth.2010.04.028

Bonato, P. (2003). Wearable sensors/systems and their impact on biomedical engineering. *IEEE Engineering in Medicine and Biology Magazine*, Vol. 22(3), 18-20, doi: 10.1109/MEMB.2003.1213622

Bouscein, W. (2012). *Electrodermal activity*. New York: Springer Science + Business Media.

Cacioppo, J.T., & Tassinary, L.G. (1990). Inferring psychological significance from physiological signals. *American Psychologist*, Vol. 45(1), 16-28, doi: 10.1037/0003-066X.45.1.16

Carter, B.L., & Tiffany, S.T. (1999). Meta-analysis of cue-reactivity in addiction research. *Addiction*, Vol. 94(3), 327-340, doi: 10.1046/j.1360-0443.1999.9433273.x

Chen, W., Jaques, N., Taylor, S., Sano, A., Fedor, S., & Picard, R.W. (2015, August 25-29). *Wavelet-based motion artifact removal for electrodermal activity*. Paper presented at the Engineering in Medicine and Biology Society, 2015 37th Annual International Conference of the IEEE, Milan. doi: 10.1109/EMBC.2015.7319814

Cook, S., Togni, M., Schaub, M.C., Wenaweser, P., & Hess, O.M. (2006). High heart rate: a cardiovascular risk factor? *European Heart Journal*, Vol. 26, 2387-2393, doi: 10.1093/eurheartj/ehl259

Dawson, M.E., Schell, A.M., & Filion, D.L. (2007). The Electrodermal System. In Cacioppo, J.T., Tassinary, L.G., & Berntson, G. (3rd Ed.), *Handbook of Psychophysiology* (159-181). Cambridge: University Press.

del Pozo, G.B., Vázquesz, I.M., Ávila, C.S., Casanova, J.G., Gómez, M.A., & Morales, L.G. (2014, October). D4.1 State of the Art - Wearable Sensors. Retrieved from http://www.daphne-fp7.eu/sites/default/files/D4.1%20State%20of%20the%20Art%20Wearable%20Sensors%20.pdf

Documentation. (n.d.). Retrieved from http://www.ledalab.de/documentation.htm#References

Empatica E4. (2014, November 19). Retrieved from http://box.empatica.com/documentation/20141119_E4_TechSpecs.pdf

Fletcher, R.R., Tam, S., Omojola, O., Redemske, R., Fedor, S., & Moshoka, J.M. (2011, May 23-26). *Mobile application and wearable sensors for use in cognitive behavioral therapy for drug addiction and PTSD*. Paper presented at the Pervasive Computing Technologies for Healthcare (PervasiveHealth), 2011 5th International Conference, Dublin. doi: 10.4108/icst.pervasivehealth.2011.246105

Fowles, D.C., Christie, M.J., Edelberg, R., Grins, W.W., Lykken, D.T., & Venables, P.H. (1981). Publication recommendations for electrodermal measurements. *Psychophysiology*, Vol. 18(3), 232-239, doi: 10.1111/j.1469-8986.1981.tb03024.x

Furlan, R., Guzzetti, S., Crivellaro, W., Dass, S., Tinelli, M., Baselli, G., Cerutti, S., Lombardi, F., Pagani, M., & Malliani, A. (1990). Continuous 24-Hour Assessment of the Neural Regulation of Systemic Arterial Pressure and RR Variabilities in Ambulant Subjects. *Circulation*, Vol. 81, 537-547, doi: 10.1161/01.CIR.81.2.537

Garbarino, M., Lai, M., Bender, D., Picard, R.W., & Tognetti, S. (2014, November 3-5). *Empatica E3 - A wearable wireless multi-sensor device for real-time computerized biofeedback and data acquisition*. Paper presented at the 4th International Conference on Wireless Mobile Comunication and Healthcare (Mobihealth), Athens. doi: 10.1109/MOBIHEALTH.2014.7015904

Gu, W.B., Poon, C.C.Y., Sy, M.Y., Leung, H.K., Liang, Y.P., & Zhang, Y.T. (2009, June 3-5). *A h-Shirt-Based Body Sensor Network for Cuffless Calibration and Estimation of Arterial Blood Pressure*. Paper presented at the Wearable and Implantable Body Sensor Networks, 6th Internation Workshop, Berkeley. doi: 10.1109/BSN.2009.23

Have you done comparative studies or validation of the heart rate obtained from the E3? (2015, September 8). Retrieved from https://support.empatica.com/hc/en-us/articles/200293658-Have-you-done-comparative-studies-or-validation-of-the-heart-rate-obtained-from-the-E3-

Hayes, M.J., & Smith, P.R. (1999, January 15). *Quantitive evaluation of photoplethysmographic artefact reduction for pulse oximetry*. Paper presented at the EUROPTO Conference on Medical Sensors and Fiber Optic Sensors IV, Sweden. doi: 10.1117/12.336924

Hernandez, J., McDuff, D., & Picard, R.W. (2015, May 20-23). BioWatch: estimation of heart and breathing rates from wrist motions. Paper presented at the Proceedings of the 9th International Conference on Pervasive Computing Technology for Healthcare, PervasiveHealth'15, Istanbul. doi: 10.4108/icst.pervasivehealth.2015.259064

How is IBI.csv obtained? (2015, November 9). Retrieved from https://support.empatica.com/hc/en-us/articles/201912319-How-is-IBI-csv-obtained-

Information and FAQs. (n.d.). Retrieved from http://eda-explorer.media.mit.edu/info/

Ishchenko, A.N., & Shev'ev, P.P. (1989). Automated complex for multiparameter analysis of the galvanic skin response signal. *Biomedical Engineering*, Vol. 23(3), 113-117, doi: 10.1007/BF00562429

Kaplan, R.F., Meyer, R.E., & Stroebel, C.F. (1983). Alcohol Dependence and Responsively to an Ethanol Stimulus as Predictors of Alcohol Consumption. *Addiction*, Vol. 78(3), 259-267, doi:10.1111/j.1360-0443.1983.tb02510.x

Kappeler-Setz, C., Gravenhorst, F., Schumm, J., Arnrich, B., & Tröster, G. (2013). Towards long term monitoring of electrodermal activity in daily life. *Personal and Ubiquitous Computing*, Vol. 17(2), 261-271, doi: 10.1007/s00779-011-0463-4

Laberg, J.C. (1986). Alcohol and Expectancy: subjective, psychophysiological and behavioral responses to alcohol stimuli in severely, moderately and non-dependent drinkers. *Addiction*, Vol. 81(6), 797-808, doi: 10.1111/j.1360-0443.1986.tb00407.x

Laberg, J.C., & Ellertsen, B. (1987). Psychophysiological Indicators of Craving in Alcoholics: Effects of cue exposure. *Addiction*, Vol. 82(12), 1341-1348, doi: 10.1111/j.1360-0443.1987.tb00437.x

Lykken, D.T., & Venables, P.H. (1971). Direct measurement of skin conductance: A proposal for standardization. *Psychophysiology*, Vol. 8(5), 656-672, doi: 10.1111/j. 1469-8986.1971.tb00501.x

Mandryk, R.L., Inkuben, K.M., & Calvert, T.W. (2006). Using psychophysiological techniques to measure user experience with entertainment technologies. *Behavior & Information Technology*, Vol. 25(2), 141-158, doi: 10.1080/01449290500331156

Ouwerkerk, M., Dandine, P., Bolio, D., Kocielnik, R., Mercurio, J., Huijgen, H., & Westerink, J. (2013, November 1-3). *Wireless multi sensor bracelet with discreet feedback*. Paper presented at Proceedings of the 4th Conference on Wireless Health (WH'13), Baltimore. doi: 10.1145/2534088.2534104

Payne, T.J., Rychtarik, R.G., Rapport, N.B., Smith, P.O., Etscheidt, M., Brown, T.A., & Johnson, C.A. (1992). Reactivity to alcohol-relevant beverage and imaginal cues in alcoholics. *Addictive Behaviors*, Vol. 17(3), 209-217, doi: 10.1016/0306-4603(92)90026-R

Picard, R.W. (2009). Future affective technology for autism and emotion communication. *Philosophical Transactions of the Royal Society*, Vol. 364, 3575-3584, doi: 10.1098/ rstb. 2009.0143

Picard, R.W., & Healey, J. (1997). Affective wearables. *Personal Technologies*, Vol. 1(4), 231-240, doi: 10.1007/BF01682026

Poon, C.C.Y., Wong, Y.M., & Zhang, Y.T. (2006. July). *M-Health: The Development of Cuff-less and Wearable Blood Pressure Meters for Use in Body Sensor Networks*. Paper presented at Life Science Systems and Applications Workshop, Bethesda. doi: 10.1109/LSSA.2006.250377

Poh, M.-Z., Swenson, N.C., & Picard, R.W. (2010). A Wearable Sensor for Unobtrusive, Long-Term Assessment of Electrodermal Activity. *IEEE Transactions on Biomedical Engineering*, Vol. 57(5), 1243-1252, doi: 10.1109/TBME.2009.2038487

Recommended tools for signal processing and data analysis. (2015, November 9). Retrieved from https://support.empatica.com/hc/en-us/articles/202872739-Recommended-tools-for-signal-processing-and-data-analysis

Scanaill, C.N., Carew, S., Barralon, P., Nouri, N., Lyons, D., & Lyons, G.M. (2006). A Review of Approaches to Mobility Telemonitoring of the Elderly in Their Living Environment. *Annals of Biomedical Engineering*, Vol. 34(4), 547-563, doi: 10.1007/s10439-005-9068-2

Setz, C., Arnich, B., Schumm, J., La Marca, R., & Tröster, G. (2010). Disciminating Stress From Cognitive Load Using a Wearable EDA Device. *IEEE Transactions on Information Technology in Biomedicine*, Vol. 14(2), 410-417, doi: 10.1109/TITB.2009.2036164

Sinha, R., Fox, H.C., Hong, K.A., Bergquist, K., Bhagwagar, Z., & Siedlarz, K.M. (2009). Enhanced Negative Emotion and Alcohol Craving, and Altered Physiological Responses Following Stress and Cue Exposure in Alcohol Dependent Individuals. *Neuropsychopharmacology*, Vol. 34, 1198-1208, doi: 10.1038/npp.2008.78

Sinha, R., Talih, M., Malison, R., Cooney, N., Anderson, G.M., & Kreek, M.J. (2003). Hypothalamic-pituitary-adrenal axis and sympatho-adreno-medullary responses during stress-induced and drug cue-induced cocaine craving states. *Psychopharmacology*, Vol. 170(1), 62-72, doi: 10.1007/s00213-003-1525-8

Stern, R.M., Ray, W.J., & Quigley, K.S. (2001). *Psychophysiological Recording*. New York: Oxford University Press.

Tapia, E.M, Intille, S.S., Haskell, W., Larson, K., Wright, J., King, A., & Friedman, R. (2007, October 11-13). *Real-Time Recognition of Physical Activities and Their Intensities Using Wireless Accelerometers and a Heart Rate Monitor*. Paper presented at Wearable Computers, 2007 11th IEEE International Symposium, Boston. doi: 10.1109/ISWC.2007.4373774

Taylor, S., Jaques, N., Chen, W., Fedor, S., Sano, A., & Picard, R.W. (2015, August 25-29). *Automatic identification of artifacts in electrodermal activity data*. Paper presented at the Engineering in Medicine and Biology Society, 2015 37th Annual International Conference of the IEEE, Milan. doi: 10.1109/EMBC.2015.7318762

Tiffany, S.T. (1999). Cognitive concepts of craving. *Alcohol Research and Health*, Vol. 23(3), 215-224

Tiffany, S.T., Carter, B.L., & Singleton, E.G. (2000). Challenges in the manipulation, assessment and interpretation of craving relevant variables. *Addiction*, Vol. 95(8s2), 177-187, doi: 10.1046/j.1360-0443.95.8s2.7.x

van Dooren, M., de Vries, J.J.G., & Janssen, J.H. (2012). Emotional sweating across the body: Comparing 16 different skin conductance measurement locations. *Physiology & Behavior*, Vol. 106(2), 209-304, doi: 10.1016/j.physbeh.2012.01.020

What should I know to use the PPG/IBI data in my experiment? (2016, April 27). Retrieved from https://support.empatica.com/hc/en-us/articles/203621335-What-should-I-know-to-use-the-PPG-IBI-data-in-my-experiment-

Wiard, R.M., Inan, O.T., Argyres, B., Temadi, M., Kovacs, G.T.A., & Giovangrandi, L. (2011). Automatic detection of motion artifacts in the ballistocardiogram measured on a modified bathroom scale. *Medical & Biological Engineering & Computing*, Vol. 49(2), 213-220, doi: 10.1007/s11517-010-0722-y

Zheng, Y., Ding, X., Poon, C.C., Lo, B.P., Zhang, H., Zhou, X., Yang, G., Zhao, N., & Zhang, Y. (2014). Unobtrusive Sensing and Wearable Devices for Health Informatics. *IEEE Transactions on Biomedical Engineering*, Vol. 61(5), 1538-1554, doi: 10.1109/TBME. 2014.2309951

Appendix

Appendix A - Questionnaire about Alcohol Use

Instucties: De vragen gaan over het afgelopen half jaar

Ik ben:

- Man
- Vrouw

Leeftijd:

Hoe veel glazen alcohol drink je gemiddeld per week?

- 0-4
- 4-8
- 8-12
- 12-14
- 14-18
- 18-22
- 22-24
- 24 of meer

Hoe vaak drink je gemiddeld per week?

- 0-1
- 1-2
- 2-3
- 3-4
- 4-5
- 5-6
- 6-7

Hoe vaak per maand drink je gemiddeld?

- 0-1
- 1-2
- 2-3
- 3-4
- 4-5
- 5-6
- 6 keer of meer

Hoe vaak per maand drink je meer dan 6 glazen alcohol?

- 0-1
- 1-2
- 2-3
- 3-4
- 4-5
- 5-6
- 6 keer of vaker

Instructies: Antwoord met "ja" of "nee"

In the past year, have you:

1. Had times when you ended up drinking more, or longer, than you intended?
2. More than once wanted to cut down or stop drinking, or tried to, but couldn't?
3. Spent a lot of time drinking? Or being sick or getting over other aftereffects?
4. Wanted a drink so badly you couldn't think of anything else?
5. Found that drinking—or being sick from drinking—often interfered with taking care of your home or family? Or caused job troubles? Or school problems?
6. Continued to drink even though it was causing trouble with your family or friends?
7. Given up or cut back on activities that were important or interesting to you, or gave you pleasure, in order to drink?

8. More than once gotten into situations while or after drinking that increased your chances of getting hurt (such as driving, swimming, using machinery, walking in a dangerous area, or having unsafe sex)?
9. Continued to drink even though it was making you feel depressed or anxious or adding to another health problem? Or after having had a memory blackout?
10. Had to drink much more than you once did to get the effect you want? Or found that your usual number of drinks had much less effect than before?
11. Found that when the effects of alcohol were wearing off, you had withdrawal symptoms, such as trouble sleeping, shakiness, restlessness, nausea, sweating, a racing heart, or a seizure? Or sensed things that were not there?

Appendix B - Instruction-Plan

Tijd in minuten	**Actie**
5 minuten	Proefpersonen welkom heten en bedanken dat ze meedoen aan het onderzoek
5 minuten	Korte uitleg over het verloop van de sessie: volgorde van de stappen en uitleggen dat we eerst bij elke stap iets gaan vertellen en dat na elke stap tijd is voor vragen
	Downloaden Empatica Manager en tijdens het downloaden
5 minuten	Uitleggen wat de bedoeling van het onderzoek is, zoals het in het formulier van de ethische toetsing staat
5 minuten	Vragen over het doel van het onderzoek?
1 minuut	Vragenlijst drinkgedrag
3 minuten	Vragenlijst DSM-V
5 minuten	Vragen over de vragenlijst
	Installatie Empatica Manager + toegangsgegevens en tijdens de installatie
10 minuten	Instructies sensor en formulier sensor
10 minuten	Vragen sensor
5 minuten	Uitleg weekoverzicht
3 minuten	Informed Consent laten tekenen
5 minuten	Afspraak maken voor het inleveren sensor en interview
10 minuten	Tijd voor vragen en nog een keer bedanken voor het meedoen

Appendix C - Sensor Instructions

Algemene instructies:

De E4 sensor is een draagbare biosensor die fysiologische reacties van de lichaam kan meten. De E4 komt met een USB-kabel en een adopter welk nodig is om de sensor op te laden. Om de data up te loaden is er een software nodig, namelijk de empatica manager. (ik had bedacht dat het misschien het handigst is om van te voren al een account aan te maken voor iedereen en dan tijdens de instructiedag hun deze gegevens te geven. Dit zou voor de proefpersoon makkelijker maken omdat zij of hij dan al de gegevens heeft en voor ons als onderzoekers omdat wij dan direct per empatica connect de upgeloade data kunnen inzien.)

De sensor is niet watervast, dus voor het handen wassen, douchen enz. moet de sensor worden afgenomen.

Tijdens het slapen hoeft de sensor niet gedragen te worden.

De sensor moet elke avond worden opgeladen. Dit kan doormiddels hem in de stopcontact te stoppen of aan je laptop aan te sluiten.

Download en installatie van empatica manager:

Om de data van de E4 up te loaden, is de empatica manager nodig (de volgende stappen moeten *alleen één keer* uitgevoerd worden):

1. installeer de empatica manager (https://www.empatica.com/empatica-manager-download)
2. sluit de empatica manager
3. koppel de E4 via USB aan je computer
4. verwijder de E4
5. start de empatica manager software en log in met jouw gegevens
6. koppel de E4 aan je computer

Uploaden van de data:

De empatica manager wordt gebruikt om de data up te loaden. Darvoor moet je de software starten en je inloggen. De upload gebeurd dan automatisch en wordt bevestigd door een kleine berichtje. Deze procedure moet *alle twee dagen* gebeuren.

Gebruik van de E4:

Om de sensor te starten moet je 1-2 seconden de button drukken. De E4 gaat automatisch in de record-mode welke zichtbaar wordt door een groene licht op de sensor. Bovendien is het mogelijk om markers te zetten door de button even kort te drukken (maar laat op dat je het niet te lang drukt en de sensor uitmaakt). Door de marker te zetten is het later mogelijk om de deze tijdpunt weer terug te vinden. Door de button 2 seconden te drukken, kan je de E4 uit schakelen.

Appendix D - Week-Plan

Dag 2 - 8

1. Sensor omdoen
2. Meetmoment 1 om 11:30 : De participant krijgt een alert dat er een vragenlijst aankomt en vervolgens de vragenlijst.
3. Meetmoment 2 om 15:30: De participant krijgt een alert dat er een vragenlijst aankomt en vervolgens de vragenlijst
4. Meetmoment 3 om 19:30: De participant krijgt een alert dat er een vragenlijst aankomt en vervolgens de vragenlijst
5. Meetmoment 4 om 24:00: De participant krijgt een alert dat er een vragenlijst aankomt en vervolgens de vragenlijst
6. Uploaden van de data
7. Opladen van de sensor

Dag 9

1. Interview
2. Sensor inleveren
3. Vragen?

Appendix E - Informed Consent

Titel onderzoek: Alcohol in daily life

Verantwoordelijke onderzoeker: Niklas Enewoldsen

In te vullen door de deelnemer

Ik verklaar op een voor mij duidelijke wijze te zijn ingelicht over de aard, methode, doel en de risico's en belasting van het onderzoek. Ik weet dat de gegevens en resultaten van het onderzoek alleen anoniem en vertrouwelijk aan derden bekend gemaakt zullen worden. Mijn vragen zijn naar tevredenheid beantwoord.

Ik begrijp dat de recorded fysiologische data en de bewerking daarvan uitsluitend voor analyse en/of wetenschappelijke presentaties zal worden gebruikt.

Ik stem geheel vrijwillig in met deelname aan dit onderzoek. Ik behoud me daarbij het recht voor om op elk moment zonder opgaaf van redenen mijn deelname aan dit onderzoek te beëindigen.

Naam deelnemer: ……………………………………………………………………

Datum: …………………… Handtekening deelnemer: ……………………………….

In te vullen door de uitvoerende onderzoeker

Ik heb een mondelinge en schriftelijke toelichting gegeven op het onderzoek. Ik zal resterende vragen over het onderzoek naar vermogen beantwoorden. De deelnemer zal van een eventuele voortijdige beëindiging van deelname aan dit onderzoek geen nadelige gevolgen ondervinden.

Naam deelnemer: ……………………………………………………………………

Datum: …………………… Handtekening onderzoeker: ……………………………..

Appendix F - Program: Merge CSV-Files (E4 Data)

Program Description. The program is suitable for the preprocessing of the data gathered through the E4 wristband, more specifically for the merging of the session files per data type. By the E4 wristband, it is only possible to download the measured data through the web-application „Empatica Connect“. This application only allows the user to download each session separately which can easily turn in a lot of files which have to be merged some way for further analysis. To use this program to its fully potential, here are a few steps you have to make in order to use the program properly:

1. The program was written in python, thus make sure you have python installed on your operating system.
2. First, you have to rename the files. In each session-folder, the files are named the same. This program was designed to merge only files of the same data-type, for example EDA-files. Therefore, you have to rename the files in the following order: first the type of file, then the number of the participant followed by the number of the file (with two dots between), for example „EDA_1.1.csv“. If you have four EDA-files you want to merge, you name them ranging from „EDA_1.1.csv“ to „EDA_1.4.csv“. (All this is also explained briefly in the beginning of the program)
3. Make sure that all the files you want to merge are in the same directory.
4. Now you can start the program.
5. Type in the file path where the files are located you want to merge (e.g. /Users/ene/ Documents/).
6. Type in the number of participant (e.g. 1, 2, 3, …).
7. Type in the type of data you want to merge (e.g. EDA, ACC, HR, IBI, …).
8. The program will then begin to merge the files and will create two new files: the merged file and a file containing the start-times, end-times and durations of the individual files within the merged file (with the original timestamp + a computation in seconds). This is useful information which is probably needed later for the analysis, especially if you want to integrate the event-markers into your analysis. The program will communicate at the end how the new files are named (these can be found in the same directory, where the files you wanted to merge are in).

```python
# -*- coding: utf-8 -*-
"""
Created on Sun May 22 01:26:00 2016

@author: ene
"""

import os
import time

print("--------------------------------------------------------------------------------")
print("------------------------------- MERGE CSV FILES --------------------------------")
print("-------- with an extra file including starttime, endtime & duration -----------")
print("--------------------------------------------------------------------------------")
print(" ")
print("All files are needed in the following format:")
print("example: EDA_1.1.csv")
print(" ")
print("First the type of data (EDA, HR, ACC, ...)")
print("Followed by the number of the participant and the number of the file")
print(" ")
print("Thus if you have 4 files of EDA-data of participant 1,")
print("the files range from EDA_1.1.csv to EDA_1.4")
print(" ")

time.sleep(2)

filepath = raw_input("Please type in the filepath of the folder where the files are located (e.g. /Users/ene/Documents/):   ")

os.chdir(filepath)

participant  = raw_input("Type in the number of the Participant (e.g. 1, 2, 3, ...):   ")
total_files  = raw_input("Type in the amount of files you want to merge (e.g. 1, 2, 3, ...):   ")
type_of_data = raw_input("Type in the type of data you want to merge (e.g. EDA, ACC, HR, IBI, ...):   ")

time_point   = 0
system_state = "Merging files"
loading      = "."

outputfile1 = open("p_" + str(participant) + "_total" + str(type_of_data) + ".csv", "a")
outputfile2 = open("p_" + str(participant) + "_" + str(type_of_data) + "_starttime_endtime_duration.txt", "a")

print(" ")
print("--------------------------------------------------------------------------------")
print(" ")
print(system_state)
print(" ")

for num in range(1, int(total_files) + 1):
    f = open(str(type_of_data) + "_" + str(participant) + "." + str(num) + ".csv")

    print loading,

    if os.stat(str(type_of_data) + "_" + str(participant) + "." + str(num) + ".csv").st_size == 0:
        print("file " + str(num) + " of Participant " + str(participant) + " is empty")

        outputfile2.write("File: " + str(type_of_data) + "_" + str(participant) + "." + str(num) + "\n")
        outputfile2.write("\n")
        outputfile2.write("File: " + str(type_of_data) + "_" + str(participant) + "." + str(num) + " is empty." + "\n")
        outputfile2.write("\n")

        f.close()
    else:
        outputfile2.write("File: " + str(type_of_data) + "_" + str(participant) + "." + str(num) + "\n")

        # Beginn TimeStamp
        line1 = f.next()
        outputfile2.write("Beginn(TimeStamp): " + str(line1))

        # Beginn innerhalb des Files
        outputfile2.write("Beginn: " + str(time_point) + " sec" + "\n")

        # Dauer des Files
        row_counter = sum(1 for row in f)
        total_rows = row_counter - 1
        duration = float(total_rows) / float(4)
        outputfile2.write("Dauer: " + str(duration) + " sec" + "\n")

        # Ende des Files
        time_point = float(time_point) + float(duration)
        outputfile2.write("Ende: " + str(time_point) + " sec" + "\n")

        # freie Zeile
        outputfile2.write("\n")

        f.close()

        f = open(str(type_of_data) + "_" + str(participant) + "." + str(num) + ".csv")

        f.next()
        f.next()
        for line in f:
            outputfile1.write(line)

        f.close()

outputfile1.close()
outputfile2.close()

print(" ")
print("--------------------------------------------------------------------------------")
print(" ")
print("A document with the name: p_" + str(participant) + "_total" + str(type_of_data) + ".csv was created.")
print(" ")
print("A document with the name: p_" + str(participant) + "_" + str(type_of_data) + "_starttime_endtime_duration.txt was created.")
print(" ")
print("--------------------------------------------------------------------------------")
print("-------------------- programmed by Niklas Enewoldsen ---------------------------")
print("--------------------------------------------------------------------------------")
```

Appendix G - Program: SCR Analyzer (TTP & CDA)

Program Description. The program was designed to summarize the results of the TTP and CDA analysis which were conducted with Ledalab. In particular, this program is able to summarize the SCR-lists which can be exported in Ledalab and consist of the onset time of the SCR and the amplitude. The SCR Analyzer generates an output-file where for each participant the following variables are computed:

- the total amount of SCRs
- the average of the amplitudes
- the standard deviation of the amplitudes
- the length of the recording in minutes
- the average of SCRs per minute
- the standard deviation of SCRs per minute
- maximum of the SCRs per minute
- minimum of the SCRs per minute

In order to be able to summarize the SCR-lists, some variables have to configured in the source code. First, the file-path has to be set whereby it is important that if you want to summarize more than one file, they all have to be in the same directory. This is followed by the total amount of files and the file type which is either „CDA“ or „TTP“. The program is able to process just one of the two file types at a time, but for as many participants as you want. Then the total-time in seconds of the files you want to merge has to be set in the variable, whereby it is started by the index one (!! leave the zero as first item in the list !!). Furthermore, the total-time in seconds has to be in the same chronologically order as the files you want to summarize, starting with participant one and then ranging up to the last one. Finally, the files have to be named in the following way (it is proposed to work with the txt output of Ledalab):

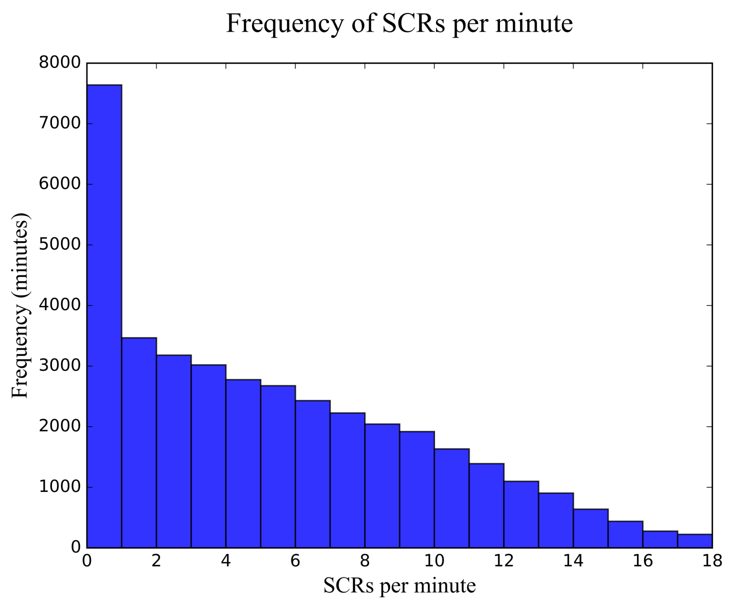
p_1_SCRlist_CDA.txt or p_1_SCRlist_TTP.txt

Thus first „p“ followed by the number of the participant plus „SCRlist“ combined with the type of file. Notice that if you want to summarize more than one file per time, the participant numbers have to start from one and should be consistently.

```python
# -*- coding: utf-8 -*-
"""
Created on Thu May 26 18:18:41 2016

@author: ene
"""

import os
import csv
import numpy as np
from bisect import bisect_left

os.chdir("/Users/ene/Documents/Ledalab")

scr_ons  = []
scr_amp  = []
interval = []
counter  = []

last = 0
total_time = [0, 210876, 303701, 307815, 309969, 436403, 267795, 317667, 127244]

total_files  = 4
type_of_file = "CDA"

for num in range(1, total_files + 1):
    with open("/Users/ene/Documents/Ledalab/p_" + str(num) + "_SCRlist_" + str(type_of_file)
     + ".txt", "r") as f:
        f.next()
        reader = csv.reader(f, delimiter = "\t")
        for onset, amp in reader:
            scr_ons.append(onset)
            scr_amp.append(amp)

    # clean lists from spaces & convert to numpy
    scr_ons = [x.strip(" ") for x in scr_ons]
    scr_amp = [x.strip(" ") for x in scr_amp]

    scr_np_ons = np.array(map(float , scr_ons))
    scr_np_amp = np.array(map(float , scr_amp))

    # compute mean & std
    length_amp = len(scr_amp)
    length_ons = len(scr_ons)

    mean_amp = np.mean(scr_np_amp)
    std_amp  = np.std(scr_np_amp)

    # create list with all the minutes
    total_list = range(0, total_time[num] + 60)

    for i in total_list:
        if i % 60 == 0:
            interval.append(i)

    # count SCRs per minute
    scr_ons = map(float , scr_ons)

    for range_max in interval:
        i = bisect_left(scr_ons, range_max, last)
        counter.append(i - last)
        last = i

    # mean, std, min & max for SCR/Minute
    counter_np   = np.array(counter)
    mean_scr_min = np.mean(counter_np)
    std_scr_min  = np.std(counter_np)
    min_scr_min  = min(counter)
    max_scr_min  = max(counter)

    length = len(interval)

    # write in outputfile
    outputfile = open("/Users/ene/Documents/Ledalab/total_" + str(type_of_file) + "
    _SCRlist_statistics.txt", "a")

    outputfile.write("Participant " + str(num) + " - " + str(type_of_file))
    outputfile.write("\n")
    outputfile.write("In total, the file consisted " + str(length_amp) + " SCRs." + "\n")
    outputfile.write("Mean of the amplitudes: " + str(mean_amp) + "\n")
    outputfile.write("Standard Deviation of the amplitudes: " + str(std_amp) + "\n")
    outputfile.write("\n")
    outputfile.write("In total, the file shows " + str(length) + " minutes of measurements."
     + "\n")
    outputfile.write("Mean of SCRs per minute: " + str(mean_scr_min) + "\n")
    outputfile.write("Standard Deviation of SCRs per minute: " + str(std_scr_min) + "\n")
    outputfile.write("Minimum of SCRs per minute: " + str(min_scr_min) + "\n")
    outputfile.write("Maximum of SCRs per minute: " + str(max_scr_min) + "\n")
    outputfile.write("\n")
    outputfile.write("\n")

    # clear variables & lists
    del scr_ons[:]
    del scr_amp[:]
    del interval[:]
    del counter[:]
    del total_list[:]

    last = 0

outputfile.close()
```

Appendix H - Program: SCRs per Minute Graph-Designer

Program Description. The program was designed to design graphs which show the amount of SCRs per minute gathered through the TTP and CDA analysis which were conducted with Ledalab. In particular, this program is able to compute the SCRs per minute and transform these values into a histogram. The SCRs per Minute Graph-Designer thereby can take up to eight participants and data-sets into account with one type of data at a time, thus TTP or CDA. This an example of a histogram plotted by the present program:



In order to be able to get these graphs, some variables have to configured in the source code. First, the file-path has to be set whereby it is important that if you want to summarize more than one file, they all have to be in the same directory. This is followed by the total amount of files and the file type which is either „CDA“ or „TTP“. The program is able to process just one of the two file types at a time, but for as many participants as you want. Then the total-time in seconds of the files you want to merge has to be set in the variable, whereby it is started by the index one (!! leave the zero as first item in the list !!). Furthermore, the total-time in seconds has to be in the same chronologically order as the files you want to summarize, starting with participant one and then ranging up to the last one. Finally, the files have to be named in the following way (it is proposed to work with the txt output of Ledalab):

p_1_SCRlist_CDA.txt or p_1_SCRlist_TTP.txt

Thus first „p“ followed by the number of the participant plus „SCRlist“ combined with the type of file. Notice that if you want to summarize more than one file per time, the participant numbers have to start from one and should be consistently.

For a more specific configuration of the histograms, you can change the names for the axis, the title, as well as all the other possible settings for histograms. Furthermore, the program prints the mean SCRs per minute, with the standard deviation, the maximum and the minimum.

```python
# -*- coding: utf-8 -*-
"""
Created on Thu May 26 21:29:32 2016

@author: ene
"""

import os
import csv
import numpy as np
from bisect import bisect_left
import matplotlib.pyplot as plt

os.chdir("/Users/ene/Documents/Ledalab/TTP/")

scr_ons  = []
scr_amp  = []
interval = []
counter1 = []
counter2 = []
counter3 = []
counter4 = []
counter5 = []
counter6 = []
counter7 = []
counter8 = []

total_counter = []
total_counter_mean = []

last = 0
total_time = [0, 210876, 303701, 307815, 309969, 436403, 267795, 317667, 127244]

total_files  = 8
type_of_file = "TTP"

for num in range(1, total_files + 1):
    with open("/Users/ene/Documents/Ledalab/TTP/p_" + str(num) + "_SCRlist_" + str(
    type_of_file) + ".txt", "r") as f:
        f.next()
        reader = csv.reader(f, delimiter = "\t")
        for onset, amp in reader:
            scr_ons.append(onset)
            scr_amp.append(amp)

    # clean lists from spaces & convert to numpy
    scr_ons = [x.strip(" ") for x in scr_ons]
    scr_amp = [x.strip(" ") for x in scr_amp]

    scr_np_ons = np.array(map(float, scr_ons))
    scr_np_amp = np.array(map(float, scr_amp))

    # compute mean & std
    length_amp = len(scr_amp)
    length_ons = len(scr_ons)

    mean_amp = np.mean(scr_np_amp)
    std_amp  = np.std(scr_np_amp)

    scr_ons = map(float, scr_ons)

    total_list = range(0, total_time[num] + 60)

    for i in total_list:
        if i % 60 == 0:
            interval.append(i)

    # count SCRs per minute
    if num == 1:
        for range_max in interval:
            i = bisect_left(scr_ons, range_max, last)
            counter1.append(i - last)
            last = i

    elif num == 2:
        for range_max in interval:
            i = bisect_left(scr_ons, range_max, last)
            counter2.append(i - last)
            last = i

    elif num == 3:
        for range_max in interval:
            i = bisect_left(scr_ons, range_max, last)
            counter3.append(i - last)
            last = i

    elif num == 4:
        for range_max in interval:
            i = bisect_left(scr_ons, range_max, last)
            counter4.append(i - last)
            last = i

    elif num == 5:
        for range_max in interval:
            i = bisect_left(scr_ons, range_max, last)
            counter5.append(i - last)
            last = i

    elif num == 6:
        for range_max in interval:
            i = bisect_left(scr_ons, range_max, last)
            counter6.append(i - last)
            last = i

    elif num == 7:
        for range_max in interval:
            i = bisect_left(scr_ons, range_max, last)
            counter7.append(i - last)
            last = i

    elif num == 8:
        for range_max in interval:
            i = bisect_left(scr_ons, range_max, last)
            counter8.append(i - last)
            last = i

    # clear lists
    del total_list[:]
    del scr_ons[:]
    del scr_amp[:]
    del interval[:]
    last = 0

# create figure
total_counter_without_mean = counter1 + counter2 + counter3 + counter4 + counter5 + counter6
     + counter7 + counter8
```

```python

total_counter_without_mean_np = np.array(total_counter_without_mean)

x_axis = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18]
y_axis = total_counter_without_mean_np

plt.hist(y_axis, bins = x_axis, alpha = 0.8)
plt.xlabel("SCRs per minute", fontname = "Times New Roman", size = "16")
plt.ylabel("Frequency (minutes)", fontname = "Times New Roman", size = "16")
title = plt.title("Frequency of SCRs per minute", fontname = "Times New Roman", size = "20")
title.set_position([.5, 1.05])
plt.savefig("/Users/ene/Documents/Ledalab/TTP/total_SCRs_per_minute_" + str(type_of_file) +
    ".png", format = "png", dpi = 1200)
plt.show()

# mean, std, min & max for SCR/Minute
mean_scr_min = np.mean(total_counter_without_mean_np)
std_scr_min  = np.std(total_counter_without_mean_np)
min_scr_min  = min(total_counter_without_mean_np)
max_scr_min  = max(total_counter_without_mean_np)

print("Mean SCRs per minute: " + str(mean_scr_min))
print("Standard Deviation of SCRs per minute: " + str(std_scr_min))
print("Minimum SCRs per minute: " + str(min_scr_min))
print("Maximum SCRs per minute: " + str(max_scr_min))
```

Appendix I - Program: Split CSV-Files (E4 Data)

Program Description. The program is suitable for the preprocessing of the data gathered through the E4 wristband, more specifically for the splitting of big data-files or merged files. By several analysis, it is not possible to upload or process recorded data from a week or even a day. This program helps you to split your data-files in up to 16 equal sized files with the right header. To use this program to its fully potential, here are a few steps you have to make in order to use the program properly:

1. The program was written in python, thus make sure you have python installed on your operating system.
2. If you have not merged your files before with the CSV-Merge program, you probably have to rename the files. This program was designed to split only one file at a time. Therefore, you have to rename the file in the following order: first „p_" plus the number of the participant, followed by „_total" plus the type of the file, for example „p_1_totalEDA.csv". (All this is also explained briefly in the beginning of the program)
3. Make sure that the file you want to split is in the right directory.
4. Now you can start the program.
5. Type in the file path where the files are located you want to merge (e.g. /Users/ene/Documents/).
6. Type in the number of participant (e.g. 1, 2, 3, …).
7. Type in the type of data you want to merge (EDA, ACC or TEMP).
8. Type in the number of files you want to split the file in (it is possible to split the file up into 16 equal files).
9. The program will then begin to split the files and will create eight new files: the eight equal parts of the merged file. The program will communicate at the end how the new files are named (these can be found in the same directory, where the files you wanted to merge are in). Furthermore, you get displayed how many measurements the big data-file contained and how many measurements each new file contains.

```python
# -*- coding: utf-8 -*-
"""
Created on Sun May 22 04:54:30 2016

@author: ene
"""

import os
import time

print("------------------------------------------------------------------------------")
print("----------------------------- SPLIT CSV FILES --------------------------------")
print("------------------------------ in even files ---------------------------------")
print("------------------------------------------------------------------------------")
print(" ")
print("The file is needed in the following format:")
print("example: p_1_totalEDA.csv")
print(" ")
print("First the number of the participant")
print("Followed by the type of data (EDA, HR, ACC, ...)")
print(" ")
print("This is the same name-format of the 'csv-merge-program'.")
print("It is possible to split EDA, ACC & TEMP data.")
print(" ")

time.sleep(2)

filepath = raw_input("Please type in the filepath of the folder where the file is located (e.g. /Users/ene/Documents/):   ")

os.chdir(filepath)

participant  = raw_input("Type in the number of the Participant (e.g. 1, 2, 3, ...):   ")
type_of_file = raw_input("Type in the type of data you want to merge (e.g. EDA, ACC or TEMP):   ")
print("Type in the number of files you want to split the file in")
total_files  = int(raw_input("(it is possible to split the file up into 16 equal files):   "))

counter = 0

file1_counter = 0
file2_counter = 0
file3_counter = 0
file4_counter = 0
file5_counter = 0
file6_counter = 0
file7_counter = 0
file8_counter = 0
file9_counter  = 0
file10_counter = 0
file11_counter = 0
file12_counter = 0
file13_counter = 0
file14_counter = 0
file15_counter = 0
file16_counter = 0

print(" ")
print("------------------------------------------------------------------------------")
print(" ")
print("Splitting Files...")
print(" ")
```

```

amount = range(1, total_files + 1)

header1 = "1460368715.000000"
header2 = "4.000000"

if type_of_file == "ACC":
    header1 = "1460368715.000000, 1460368715.000000, 1460368715.000000"
    header2 = "32.000000, 32.000000, 32.000000"

# create outputfiles
outputfile1 = open("p_" + str(participant) + "_" + str(type_of_file) + "_" + str(amount[0])
    + "." + str(total_files) + ".csv", "a")
if total_files >= 2:
    outputfile2 = open("p_" + str(participant) + "_" + str(type_of_file) + "_" + str(amount
    [1]) + "." + str(total_files) + ".csv", "a")
if total_files >= 3:
    outputfile3 = open("p_" + str(participant) + "_" + str(type_of_file) + "_" + str(amount
    [2]) + "." + str(total_files) + ".csv", "a")
if total_files >= 4:
    outputfile4 = open("p_" + str(participant) + "_" + str(type_of_file) + "_" + str(amount
    [3]) + "." + str(total_files) + ".csv", "a")
if total_files >= 5:
    outputfile5 = open("p_" + str(participant) + "_" + str(type_of_file) + "_" + str(amount
    [4]) + "." + str(total_files) + ".csv", "a")
if total_files >= 6:
    outputfile6 = open("p_" + str(participant) + "_" + str(type_of_file) + "_" + str(amount
    [5]) + "." + str(total_files) + ".csv", "a")
if total_files >= 7:
    outputfile7 = open("p_" + str(participant) + "_" + str(type_of_file) + "_" + str(amount
    [6]) + "." + str(total_files) + ".csv", "a")
if total_files >= 8:
    outputfile8 = open("p_" + str(participant) + "_" + str(type_of_file) + "_" + str(amount
    [7]) + "." + str(total_files) + ".csv", "a")
if total_files >= 9:
    outputfile9  = open("p_" + str(participant) + "_" + str(type_of_file) + "_" + str(amount
    [8]) + "." + str(total_files) + ".csv", "a")
if total_files >= 10:
    outputfile10 = open("p_" + str(participant) + "_" + str(type_of_file) + "_" + str(amount
    [9]) + "." + str(total_files) + ".csv", "a")
if total_files >= 11:
    outputfile11 = open("p_" + str(participant) + "_" + str(type_of_file) + "_" + str(amount
    [10]) + "." + str(total_files) + ".csv", "a")
if total_files >= 12:
    outputfile12 = open("p_" + str(participant) + "_" + str(type_of_file) + "_" + str(amount
    [11]) + "." + str(total_files) + ".csv", "a")
if total_files >= 13:
    outputfile13 = open("p_" + str(participant) + "_" + str(type_of_file) + "_" + str(amount
    [12]) + "." + str(total_files) + ".csv", "a")
if total_files >= 14:
    outputfile14 = open("p_" + str(participant) + "_" + str(type_of_file) + "_" + str(amount
    [13]) + "." + str(total_files) + ".csv", "a")
if total_files >= 15:
    outputfile15 = open("p_" + str(participant) + "_" + str(type_of_file) + "_" + str(amount
    [14]) + "." + str(total_files) + ".csv", "a")
if total_files >= 16:
    outputfile16 = open("p_" + str(participant) + "_" + str(type_of_file) + "_" + str(amount
    [15]) + "." + str(total_files) + ".csv", "a")

# insert the choosen header
outputfile1.write(header1)
outputfile1.write("\n")
outputfile1.write(header2)
```

```
outputfile1.write("\n")
if total_files >= 2:
    outputfile2.write(header1)
    outputfile2.write("\n")
    outputfile2.write(header2)
    outputfile2.write("\n")
if total_files >= 3:
    outputfile3.write(header1)
    outputfile3.write("\n")
    outputfile3.write(header2)
    outputfile3.write("\n")
if total_files >= 4:
    outputfile4.write(header1)
    outputfile4.write("\n")
    outputfile4.write(header2)
    outputfile4.write("\n")
if total_files >= 5:
    outputfile5.write(header1)
    outputfile5.write("\n")
    outputfile5.write(header2)
    outputfile5.write("\n")
if total_files >= 6:
    outputfile6.write(header1)
    outputfile6.write("\n")
    outputfile6.write(header2)
    outputfile6.write("\n")
if total_files >= 7:
    outputfile7.write(header1)
    outputfile7.write("\n")
    outputfile7.write(header2)
    outputfile7.write("\n")
if total_files >= 8:
    outputfile8.write(header1)
    outputfile8.write("\n")
    outputfile8.write(header2)
    outputfile8.write("\n")
if total_files >= 9:
    outputfile9.write(header1)
    outputfile9.write("\n")
    outputfile9.write(header2)
    outputfile9.write("\n")
if total_files >= 10:
    outputfile10.write(header1)
    outputfile10.write("\n")
    outputfile10.write(header2)
    outputfile10.write("\n")
if total_files >= 11:
    outputfile11.write(header1)
    outputfile11.write("\n")
    outputfile11.write(header2)
    outputfile11.write("\n")
if total_files >= 12:
    outputfile12.write(header1)
    outputfile12.write("\n")
    outputfile12.write(header2)
    outputfile12.write("\n")
if total_files >= 13:
    outputfile13.write(header1)
    outputfile13.write("\n")
    outputfile13.write(header2)
    outputfile13.write("\n")
if total_files >= 14:
    outputfile14.write(header1)
    outputfile14.write("\n")
    outputfile14.write(header2)
    outputfile14.write("\n")
if total_files >= 15:
    outputfile15.write(header1)
    outputfile15.write("\n")
    outputfile15.write(header2)
    outputfile15.write("\n")
if total_files >= 16:
    outputfile16.write(header1)
    outputfile16.write("\n")
    outputfile16.write(header2)
    outputfile16.write("\n")

f = open("p_" + str(participant) + "_total" + str(type_of_file) + ".csv")

row_count = sum(1 for row in f)
x         = row_count / int(total_files)

one    = x
two    = x * 2
three  = x * 3
four   = x * 4
five   = x * 5
six    = x * 6
seven  = x * 7
eight  = x * 8
nine   = x * 9
ten    = x * 10
eleven = x * 11
twelve = x * 12
thirdt = x * 13
fourth = x * 14
fifth  = x * 15
sixth  = x * 16

f.close()

f = open("p_" + str(participant) + "_total" + str(type_of_file) + ".csv")

# split the file
for line in f:
    if counter <= one:
        outputfile1.write(line)
        counter = counter + 1
        file1_counter = file1_counter + 1
    if total_files >= 2:
        if counter > one and counter <= two:
            outputfile2.write(line)
            counter = counter + 1
            file2_counter = file2_counter + 1
    if total_files >= 3:
        if counter > two and counter <= three:
            outputfile3.write(line)
            counter = counter + 1
            file3_counter = file3_counter + 1
    if total_files >= 4:
        if counter > three and counter <= four:
            outputfile4.write(line)
            counter = counter + 1
            file4_counter = file4_counter + 1
    if total_files >= 5:
        if counter > four and counter <= five:
            outputfile5.write(line)
            counter = counter + 1
            file5_counter = file5_counter + 1
    if total_files >= 6:
        if counter > five and counter <= six:
            outputfile6.write(line)
            counter = counter + 1
            file6_counter = file6_counter + 1
    if total_files >= 7:
        if counter > six and counter <= seven:
            outputfile7.write(line)
            counter = counter + 1
            file7_counter = file7_counter + 1
    if total_files >= 8:
        if counter > seven and counter <= eight:
            outputfile8.write(line)
            counter = counter + 1
            file8_counter = file8_counter + 1
    if total_files >= 9:
        if counter > eight and counter <= nine:
            outputfile9.write(line)
            counter = counter + 1
            file9_counter = file9_counter + 1
    if total_files >= 10:
        if counter > nine and counter <= ten:
            outputfile10.write(line)
            counter = counter + 1
            file10_counter = file10_counter + 1
    if total_files >= 11:
        if counter > ten and counter <= eleven:
            outputfile11.write(line)
            counter = counter + 1
            file11_counter = file11_counter + 1
    if total_files >= 12:
        if counter > eleven and counter <= twelve:
            outputfile12.write(line)
            counter = counter + 1
            file12_counter = file12_counter + 1
    if total_files >= 13:
        if counter > twelve and counter <= thirdt:
            outputfile13.write(line)
            counter = counter + 1
            file13_counter = file13_counter + 1
    if total_files >= 14:
        if counter > thirdt and counter <= fourth:
            outputfile14.write(line)
            counter = counter + 1
            file14_counter = file14_counter + 1
    if total_files >= 15:
        if counter > fourth and counter <= fifth:
            outputfile15.write(line)
            counter = counter + 1
            file15_counter = file15_counter + 1
    if total_files >= 16:
        if counter > fifth and counter <= sixth:
            outputfile16.write(line)
            counter = counter + 1
            file16_counter = file16_counter + 1

# close all files
f.close()
outputfile1.close()
if total_files >= 2:
    outputfile2.close()
if total_files >= 3:
    outputfile3.close()
if total_files >= 4:
    outputfile4.close()
if total_files >= 5:
    outputfile5.close()
if total_files >= 6:
    outputfile6.close()
if total_files >= 7:
    outputfile7.close()
if total_files >= 8:
    outputfile8.close()
if total_files >= 9:
    outputfile9.close()
if total_files >= 10:
    outputfile10.close()
if total_files >= 11:
    outputfile11.close()
if total_files >= 12:
    outputfile12.close()
if total_files >= 13:
    outputfile13.close()
if total_files >= 14:
    outputfile14.close()
if total_files >= 15:
    outputfile15.close()
if total_files >= 16:
    outputfile16.close()

print(" ")
print("--------------------------------------------------------------------------------------------")
print(" ")
print(str(total_files) + " documents with the names: p_" + str(participant) + "_" + str(type_of_file) + "_X/" + str(total_files) + ".csv were created.")
print(" ")
print("The original file contained: " + str(row_count) + " measurements.")
print(" ")
print("The following shows the amount of measurement per file:")
print(" ")
print(str(type_of_file) + " File 1: " + str(file1_counter))
if total_files >= 2:
    print(str(type_of_file) + " File 2: " + str(file2_counter))
if total_files >= 3:
    print(str(type_of_file) + " File 3: " + str(file3_counter))
if total_files >= 4:
    print(str(type_of_file) + " File 4: " + str(file4_counter))
if total_files >= 5:
    print(str(type_of_file) + " File 5: " + str(file5_counter))
if total_files >= 6:
    print(str(type_of_file) + " File 6: " + str(file6_counter))
if total_files >= 7:
    print(str(type_of_file) + " File 7: " + str(file7_counter))
if total_files >= 8:
    print(str(type_of_file) + " File 8: " + str(file8_counter))
if total_files >= 9:
    print(str(type_of_file) + " File 9: " + str(file9_counter))
if total_files >= 10:
    print(str(type_of_file) + " File 10: " + str(file10_counter))
if total_files >= 11:
    print(str(type_of_file) + " File 11: " + str(file11_counter))
if total_files >= 12:
    print(str(type_of_file) + " File 12: " + str(file12_counter))
```

```
if total_files >= 13:
    print(str(type_of_file) + " File 13: " + str(file13_counter))
if total_files >= 14:
    print(str(type_of_file) + " File 14: " + str(file14_counter))
if total_files >= 15:
    print(str(type_of_file) + " File 15: " + str(file15_counter))
if total_files >= 16:
    print(str(type_of_file) + " File 16: " + str(file16_counter))
print(" ")
print("Furthermore, each file contains a header with a Timestapm and the Samplerate.")
print(" ")
print("------------------------------------------------------------------------------------------------")
print("------------------------ programmed by Niklas Enewoldsen -------------------------------")
print("------------------------------------------------------------------------------------------------")
```

Appendix J - Program: Merge CSV-Files (EDA-Explorer)

Program Description. The program is suitable for the processing of the result-files of the peak and artifact detection, more specifically for the merging of several individual result-files. By the EDA-Explorer it is only possible to upload a file with maximum seven hours of recorded data. This program helps you to merge the analyzed, but still splitted result-files. To use this program to its fully potential, here are a few steps you have to make in order to use the program properly:

1. The program was written in python, thus make sure you have python installed on your operating system.
2. If you have not splitted your files before with the CSV-Split program, you probably have to rename the files. This program was designed to merge as many peak or artifact detection result-files as you want, but only one at a time. Therefore, you have to rename the file in the following order: for example if you have four noise-files you want to merge, you name them: „p_1_(1.4)_NoiseLabels_Multiclass.csv", „p_1_(2.4)_NoiseLabels_Multiclass.csv", „p_1_(3.4)_NoiseLabels_Multiclass.csv", „p_1_(4.4)_NoiseLabels_Multiclass.csv". For peak-files the same, but with. „p_1_(1.4)_PeakFeatures_0.01.csv" (All this is also explained briefly in the beginning of the program)
3. Make sure that the files you want to split are in the right directory or change the directory.
4. Change the four variables in the code with changing the first part with either ‚peak' or ‚noise'. The variables are the following (as an example for noise): noise_name, noise_header, noise_analysis, noise_analysis.
5. Set the number of the participant
6. Set the total-file size.
7. Now you can start the program.
8. The program will then begin to merge the files and will create a new file. The program will communicate at the end how the new files is named (these can be found in the same directory, where the files you wanted to merge are in).

```python
# -*- coding: utf-8 -*-
"""
Created on Sun May 22 06:30:01 2016

@author: ene
"""

# PEAK OR NOISE?
# in case: change the 4 variables

import os

os.chdir("/Users/ene/Documents/Ledalab")

participant  = 8
total_files  = 8

peak_analysis  = ")_PeakFeatures.csv"
noise_analysis = ")_NoiseLabels_Multiclass.csv"

peak_header  = ",EDA,rise_time,max_deriv,amp,decay_time,SCR_width,AUC"
noise_header = "EpochNum,StartTime,EndTime,MulticlassLabels"

peak_name  = "_total_SCRs_eda_explorer.csv"
noise_name = "_total_noise_eda_explorer.csv"

outputfile = open("p_" + str(participant) + str(peak_name), "a")

outputfile.write(str(peak_header))
outputfile.write("\n")

for num in range(1, total_files + 1):
    f = open("p_" + str(participant) + "_(" + str(num) + "." + str(total_files) + str(
    peak_analysis))
    if os.stat("p_" + str(participant) + "_(" + str(num) + "." + str(total_files) + str(
    peak_analysis)).st_size == 0:
        print("file " + str(num) + " of Participant " + str(participant) + " is empty")
        f.close()
    else:
        f.next()
        for line in f:
            outputfile.write(line)
        f.close()

outputfile.close()
```

Appendix K - Program: Analysis Noise-Detection (EDA-Explorer)

Program Description. The program is suitable for the analysis of „noise-files“ which were generated via the EDA-Explorer web-application. It categorizes the data into the three used categories: clean signals, noise signals and questionable signals. At the end you get one file consisting of the amount of signals per category with the associated percentage per participant. To use this program to its fully potential, here are a few steps you have to make in order to use the program properly:

1. The program was written in python, thus make sure you have python installed on your operating system.
2. First, you have to rename the files (if you did not used the program that merges the result-files EDA-Explorer), for example: „p_1_total_noise_eda_explorer.csv“. If you have 4 result-files of four different participants, the filenames will range from „p_1_total_noise_eda_explorer.csv“ to „p_1_total_noise_eda_explorer.csv“.
3. Make sure that all the files you want to analyze are in the same directory.
4. Now you can start the program.
5. Type in the file path where the files are located you want to merge (e.g. /Users/ene/Documents/).
6. Type in the amount of files you want to analyze (e.g. 1, 2, 3, …).
7. The program will then begin to analyze the files and will create a new file which consist of the amount of signals per category per participant. Furthermore, the associated percentage of the amount of the signals per category will be provided. The program will communicate at the end how the new file is named (these can be found in the same directory, where the files you wanted to analyze are in).

```python
# -*- coding: utf-8 -*-
"""
Created on Tue May 24 11:55:12 2016

@author: ene
"""

import os
import csv
import time

print(" ")
print(" ")
print("--------------------------------------------------------------------------------")
print("------------------------------- NOISE ANALYSIS ---------------------------------")
print("--------------------------------------------------------------------------------")
print(" ")
print("All files are needed in the following format:")
print("example: p_1_total_noise_eda_explorer.csv")
print(" ")
print("First the 'p_' plus the participant number")
print("Followed by '_total_noise_eda_explorer.csv'")
print(" ")
print("Thus if you have 4 result-files of four different participants,")
print("the files range from p_1_total_noise_eda_explorer.csv to p_4_total_noise_eda_explorer.csv")
print(" ")

time.sleep(2)

print("--------------------------------------------------------------------------------")
print(" ")

# input
filepath = raw_input("Please type in the filepath of the folder where the files are located (e.g. /Users/ene/Documents/):   ")

print(" ")

os.chdir(filepath)

amount_of_files = raw_input("Type in the number of files you want to analyze (e.g. 1, 2, 3, ...):   ")

counter_clean   = 0
counter_noise   = 0
counter_quest   = 0
counter_length  = 0

# opens file and saves it in dictionary
for num in range(1, int(amount_of_files) + 1):
    f = open("p_" + str(num) + "_total_noise_eda_explorer.csv", "r")
    reader = csv.DictReader(f)
    data = {}
    for row in reader:
        for header, value in row.items():
            try:
                data[header].append(value)
            except KeyError:
                data[header] = [value]

    f.close()

    # transform data to single lists
    en  = data["EpochNum"]
    st  = data["StartTime"]
    et  = data["EndTime"]
    mcl = data["MulticlassLabels"]

    # categorize between clean, noise & questionable
    for i in mcl:
        counter_length += 1
        if float(i) == 1.0:
            counter_clean += 1
        elif float(i) == 0:
            counter_quest += 1
        elif float(i) == -1.0:
            counter_noise += 1

    # compute percentage
    one_percent = float(counter_length) / 100.00
    perc_clean  = "%.2f" % (float(counter_clean) / float(one_percent))
    perc_noise  = "%.2f" % (float(counter_noise) / float(one_percent))
    perc_quest  = "%.2f" % (float(counter_quest) / float(one_percent))

    # write in file
    if amount_of_files == 1:
        outputfile = open("p_1_noise_analysis.txt", "a")
    else:
        outputfile = open("p_1_to_" + str(amount_of_files) +  "_noise_analysis.txt", "a")
    outputfile.write("Participant " + str(num) + " - Noise Analysis")
    outputfile.write("\n")
    outputfile.write("\n")

    outputfile.write("In total, the file consists of " + str(counter_length) + " measurements.")
    outputfile.write("\n")
    outputfile.write("\n")

    outputfile.write("Clean Signal" + "\n")
    outputfile.write("(is labeled through a 1)"  + "\n")
    outputfile.write("In total, " + str(counter_clean) + " measurements are clean." + "\n")
    outputfile.write("This equals " + str(perc_clean) + " %." + "\n")
    outputfile.write("\n")

    outputfile.write("Noise Signal" + "\n")
    outputfile.write("(is labeled through a -1)"  + "\n")
    outputfile.write("In total, " + str(counter_noise) + " measurements are noise." + "\n")
    outputfile.write("This equals " + str(perc_noise) + " %." + "\n")
    outputfile.write("\n")

    outputfile.write("Questionable Signal" + "\n")
    outputfile.write("(is labeled through a 0)"  + "\n")
    outputfile.write("In total, " + str(counter_quest) + " measurements are questionable." + "\n")
    outputfile.write("This equals " + str(perc_quest) + " %." + "\n")
    outputfile.write("\n")
    outputfile.write("\n")
    outputfile.write("\n")

    # reset content of variables & lists
    counter_clean   = 0
    counter_noise   = 0
    counter_quest   = 0
    counter_length  = 0

    del en[:]
    del st[:]
    del et[:]
    del mcl[:]

outputfile.close()

print(" ")
print("--------------------------------------------------------------------------------")
print(" ")
if amount_of_files == 1:
    print("A document with the name: p_1_noise_analysis.csv was created.")
else:
    print("A document with the name: p_1_to_" + str(amount_of_files) + "_noise_analysis.csv was created.")
print(" ")
print("--------------------------------------------------------------------------------")
print("-------------------- programmed by Niklas Enewoldsen ---------------------------")
print("--------------------------------------------------------------------------------")
```

Appendix L - Program: Analysis Peak-Detection (EDA-Explorer)

Program Description. The program was designed to summarize the results of the peak-detection analysis which were conducted with web-application of the EDA-Explorer. In particular, this program is able to summarize the SCR-lists which consist of several important EDA-variables. The SCR Analyzer generates an output-file where the following variables are computed and provided with an average, maximum and minimum value:

- the EDA amplitude at the peak
- rise-time
- maximum-derivative
- amplitude
- decay-time
- SCR-width
- area under the curve

The output-file also contains a short description per variable. In order to be able to summarize the SCR-lists, some variables have to configured in the source code. First, the file-path has to be set which is followed by the participant number. Note that this program is only able to process one participant at a time. Each time the program is used, it creates a new output-file with the participant number in the title. However, if there is already an existing file of a certain participant, the output is written in the file which already existed. Finally, the files have to be named in the following way (it is proposed to work with the txt output of Ledalab):

p_1_total_SCRs_eda_explorer.csv

Thus first „p“ followed by the number of the participant plus „SCRlist“ combined with the type of file. If you used the program to split and merge the CSV-files for the analysis with the EDA-Explorer, the files should already be named correctly.

```python
# -*- coding: utf-8 -*-
"""
Created on Sun May 22 06:42:48 2016

@author: ene
"""

import os
import csv

os.chdir("/Users/ene/Documents/Ledalab")

participant = 8

counter_eda = 0
counter_rt  = 0
counter_md  = 0
counter_amp = 0
counter_dt  = 0
counter_sw  = 0
counter_auc = 0

missing_dt  = 0
missing_sw  = 0
missing_auc = 0

# opens the file in universal line ending mode and creates dictionary
with open("p_" + str(participant) + "_total_SCRs_eda_explorer.csv", "r") as infile:
    reader = csv.DictReader(infile)
    data = {}
    for row in reader:
        for header, value in row.items():
            try:
                data[header].append(value)
            except KeyError:
                data[header] = [value]

eda = data["EDA"]
rt  = data["rise_time"]
md  = data["max_deriv"]
amp = data["amp"]
dt  = data["decay_time"]
sw  = data["SCR_width"]
auc = data["AUC"]

summer_eda = sum(float(x) for x in eda)
summer_rt  = sum(float(x) for x in rt)
summer_md  = sum(float(x) for x in md)
summer_amp = sum(float(x) for x in amp)
summer_dt  = 0
summer_sw  = 0
summer_auc = 0

# count sum of different variables
for i in eda:
    counter_eda += 1

for i in rt:
    counter_rt += 1

for i in md:
    counter_md += 1

```

```python
for i in amp:
    counter_amp += 1

for i in dt:
    counter_dt += 1
    if type(i) == str:
        try:
            summer_dt += float(i)
        except ValueError:
            missing_dt += 1
        else:
            summer_dt += float(i)
    else:
        summer_dt += float(i)

for i in sw:
    counter_sw += 1
    if type(i) == str:
        try:
            summer_sw += float(i)
        except ValueError:
            missing_dt += 1
        else:
            summer_sw += float(i)
    else:
        summer_sw += float(i)

for i in auc:
    counter_auc += 1
    if type(i) == str:
        try:
            summer_auc += float(i)
        except ValueError:
            missing_auc += 1
        else:
            summer_auc += float(i)
    else:
        summer_auc += float(i)

# compute average, min & max
average_eda = float(summer_eda) / float(counter_eda)
average_rt  = float(summer_rt)  / float(counter_rt)
average_md  = float(summer_md)  / float(counter_md)
average_amp = float(summer_amp) / float(counter_amp)
average_dt  = float(int(summer_dt)) / float(counter_dt - missing_dt)
average_sw  = float(summer_sw) / float(counter_sw - missing_sw)
average_auc = float(summer_auc) / float(counter_auc - missing_auc)

min_eda = min(eda)
min_rt  = min(rt)
min_md  = min(md)
min_amp = min(amp)
min_dt  = min(dt)
min_sw  = min(sw)
min_auc = min(auc)

max_eda = max(eda)
max_rt  = max(rt)
max_md  = max(md)
max_amp = max(amp)
max_dt  = max(dt)
max_sw  = max(sw)
max_auc = max(auc)
```

```

# write in outputfile
outputfile = open("p_" + str(participant) + "_SCR_analysis.txt", "a")
outputfile.write("Participant " + str(participant) + " - SCR Analysis")
outputfile.write("\n")
outputfile.write("\n")

outputfile.write("EDA" + "\n")
outputfile.write("Description: The EDA amplitude at the apex (peak), in uSiemens."  + "\n")
outputfile.write("Average: " + str(average_eda) + " uS"  + "\n")
outputfile.write("Minimum: " + str(min_eda) + " uS" + "\n")
outputfile.write("Maximum: " + str(max_eda) + " uS" + "\n")
outputfile.write("\n")

outputfile.write("Rise-time" + "\n")
outputfile.write("Description: The time, in seconds, it takes for the SCR to rise from the
    start of the SCR to the apex" + "\n")
outputfile.write("The start of the SCR is computed by going backwards from the apex of the
    peak to the point where" + "\n")
outputfile.write("derivative is less than 1% of its maximum value." + "\n")
outputfile.write("Average: " + str(average_rt) + " s" + "\n")
outputfile.write("Minimum: " + str(min_rt) + " s" + "\n")
outputfile.write("Maximum: " + str(max_rt) + " s" + "\n")
outputfile.write("\n")

outputfile.write("Maximum-derivative" + "\n")
outputfile.write("Description: Maximum derivative of SCR, in in uSiemens per second." + "\n"
    )
outputfile.write("Average: " + str(average_md) + " uS/s" + "\n")
outputfile.write("Minimum: " + str(min_md) + " uS/s" + "\n")
outputfile.write("Maximum: " + str(max_md) + " uS/s" + "\n")
outputfile.write("\n")

outputfile.write("Amplitude" + "\n")
outputfile.write("Description: Amplitude of peak; that is [amp = (EDA at apex) - (EDA at
    start of the SCR)], in uSiemens." + "\n")
outputfile.write("Average: " + str(average_amp) + " uS" + "\n")
outputfile.write("Minimum: " + str(min_amp) + " uS" + "\n")
outputfile.write("Maximum: " + str(max_amp) + " uS" + "\n")
outputfile.write("\n")

outputfile.write("Decay-time" + "\n")
outputfile.write("Description: The time, in seconds, that it takes for the SCR to decay to
    50% of its amplitude." + "\n")
outputfile.write("Note that this is blank if an SCR doesnt decay to 50% before another
    peak starts or before the maximum decay time is reached." + "\n")
outputfile.write("Average: " + str(average_dt) + " s" + "\n")
outputfile.write("Minimum: " + str(min_dt) + "\n")
outputfile.write("Maximum: " + str(max_dt) + "\n")
outputfile.write("\n")

outputfile.write("SCR-width" + "\n")
outputfile.write("Description: The time in seconds between the 50% of the amplitude on the
    incline side of the" + "\n")
outputfile.write("peak to 50% of the amplitude on the decline side of the SCR. Note that
    this is blank if a Decay_time wasnt computed." + "\n")
outputfile.write("Average: " + str(average_sw) + " s" + "\n")
outputfile.write("Minimum: " + str(min_sw) + "\n")
outputfile.write("Maximum: " + str(max_sw) + "\n")
outputfile.write("\n")

outputfile.write("Area under the curve" + "\n")
outputfile.write("Description: Area under the Curve; approximated by multiplying the
    Amplitude by the" + "\n")
outputfile.write("SCR_width. Note that this is blank if a Decay_time wasnt computed." + "
    \n")
outputfile.write("Average: " + str(average_auc) + " s" + "\n")
outputfile.write("Minimum: " + str(min_auc) + "\n")
outputfile.write("Maximum: " + str(max_auc) + "\n")
outputfile.write("\n")

outputfile.close()
```

Appendix M - Program: Distribution & Artifact Detection of HR Data

Program Description. The program is suitable for the analysis of heart-rate data regarding its distribution and possible outliners. Thereby, heart-rate data is categorized in over eight different categories and the amount of measurements in a category will be displayed, as well as the associated percentage. Furthermore, the program is specialized to detect value pairs which have an increase more than 5 bpm in a second and provide the user with a figure of the total heart-rate data per participant. To use this program to its fully potential, here are a few steps you have to make in order to use the program properly:

1. The program was written in python, thus make sure you have python installed on your operating system.
2. First, you have to rename the files (if you did not use the program which is specialized in merging csv-files of the E4 wristband). The program was written in order to analyze the total files of a participant which means that for every participant there is just one file. Therefore, you have to rename the files in the following order: first the „p“ plus the number of the participant, then the word total combined with the type of file, for example „p_1_totalHR.csv“. If you have four HR-files containing all the HR-data from four different participants, you name them ranging from „p_1_totalHR.csv“ to „p_4_totalHR.csv“. (All this is also explained briefly in the beginning of the program)
3. Make sure that all the files you want to merge are in the same directory.
4. Now you can start the program.
5. Type in the file path where the files are located you want to analyze (e.g. /Users/ene/Documents/).
6. Type in the total number of files or respectively the number of participants (e.g. 1, 2, 3, …).
7. The program will then begin to analyze the files and will create one new file where the amount of signals per category, as well as the associated percentage summarized. This includes how many value-pairs have a difference of more than 5 bpm in a second. Furthermore, a figure per participant is plotted and saved. The program will communicate at the end how the new files are named (these can be found in the same directory, where the files you wanted to merge are in).

```
# -*- coding: utf-8 -*-
"""
Created on Tue May 24 16:46:14 2016

@author: ene
"""

import os
import time
import numpy as np
import matplotlib.pyplot as plt

print(" ")
print(" ")
print("-----------------------------------------------------------------------------------------")
print("----------------------------------- HEART RATE ------------------------------------------")
print("------------------------- distribution & possible artifacts -----------------------------")
print("-----------------------------------------------------------------------------------------")
print(" ")
print("All files are needed in the following format:")
print("example: p_1_totalHR.csv")
print(" ")
print("First the 'p_' plus the participant number")
print("Followed by '_totalHR.csv'")
print(" ")
print("Thus if you have 4 result-files of four different participants,")
print("the files range from p_1_totalHR.csv to p_4_totalHR.csv")
print(" ")
print("-----------------------------------------------------------------------------------------")

time.sleep(2)

filepath = raw_input("Please type in the filepath of the folder where the files are located
    (e.g. /Users/ene/Documents/):   ")

print(" ")

os.chdir(filepath)

total_files = raw_input("Type in the number of files you want to analyze or respectively the
     number of participants (e.g. 1, 2, 3, ...):   ")

print(" ")
print("-----------------------------------------------------------------------------------------")
print(" ")

excluded_over_200   = 0
excluded_under_40   = 0
excluded_unknown    = 0

counter_40_to_49    = 0
counter_50_to_59    = 0
counter_60_to_100   = 0
counter_101_to_120  = 0
counter_121_to_150  = 0
counter_151_to_200  = 0

counter_bigger_30   = 0
counter_bigger_50   = 0

index_counter       = 0
length_counter      = 0
```

```
total_over_200   = []
total_under_40   = []
total_unknown    = []
total_40_to_49   = []
total_50_to_59   = []
total_60_to_100  = []
total_101_to_120 = []
total_121_to_150 = []
total_151_to_200 = []
total_bigger_30  = []
total_bigger_50  = []

total_files = 8

for num in range(1, int(total_files) + 1):
    f = open("p_" + str(num) + "_totalHR.csv")
    data_list  = []
    bigger_30  = {}
    bigger_50  = {}

    # saves file in list & already categorization
    for line in f:
        data_list.append(line)

        length_counter += 1

        line = int(float(line))

        if line > 200:
            excluded_over_200 += 1
        elif line < 40:
            excluded_under_40 += 1
        elif line >= 40 and line < 50:
            counter_40_to_49 += 1
        elif line >= 50 and line < 60:
            counter_50_to_59 += 1
        elif line >= 60 and line <= 100:
            counter_60_to_100 += 1
        elif line > 100 and line <= 120:
            counter_101_to_120 += 1
        elif line > 120 and line <= 150:
            counter_121_to_150 += 1
        elif line > 150 and line <= 200:
            counter_151_to_200 += 1
        else:
            excluded_unknown += 1

    # counts & writes value-pairs with difference more than 3bpm / 5bpm in a second
    for x, y in zip(data_list, data_list[1:]):
        index_counter += 1
        if float(x) - float(y) > 3.0 or float(y) - float(x) < -3.0:
            counter_bigger_30 += 1
            bigger_30.update({str(index_counter):[x, y]})
            if float(x) - float(y) > 5.0 or float(y) - float(x) < -5.0:
                counter_bigger_50 += 1
                bigger_50.update({str(index_counter):[x, y]})

    # compute precentage
    one_percent     = float(length_counter) / 100.00
    perc_over_200   = "%.2f" % (float(excluded_over_200)  / float(one_percent))
    perc_under_40   = "%.2f" % (float(excluded_under_40)  / float(one_percent))
    perc_unknown    = "%.2f" % (float(excluded_unknown)   / float(one_percent))
    perc_40_to_49   = "%.2f" % (float(counter_40_to_49)   / float(one_percent))
    perc_50_to_59   = "%.2f" % (float(counter_50_to_59)   / float(one_percent))
    perc_60_to_100  = "%.2f" % (float(counter_60_to_100)  / float(one_percent))
    perc_101_to_120 = "%.2f" % (float(counter_101_to_120) / float(one_percent))
    perc_121_to_150 = "%.2f" % (float(counter_121_to_150) / float(one_percent))
    perc_151_to_200 = "%.2f" % (float(counter_151_to_200) / float(one_percent))
    perc_bigger_30  = "%.2f" % (float(counter_bigger_30)  / float(one_percent))
    perc_bigger_50  = "%.2f" % (float(counter_bigger_50)  / float(one_percent))

    # for total percentages
    total_over_200.append(perc_over_200)
    total_under_40.append(perc_under_40)
    total_unknown.append(perc_unknown)
    total_40_to_49.append(perc_40_to_49)
    total_50_to_59.append(perc_50_to_59)
    total_60_to_100.append(perc_60_to_100)
    total_101_to_120.append(perc_101_to_120)
    total_121_to_150.append(perc_121_to_150)
    total_151_to_200.append(perc_151_to_200)
    total_bigger_30.append(perc_bigger_30)
    total_bigger_50.append(perc_bigger_50)

    # write in file
    if total_files == 1:
        outputfile = open("p_1_HR_total_distribution_of_data.txt", "a")
    else:
        outputfile = open("p_1_to_" + str(total_files) + "_HR_total_distribution_of_data.txt
", "a")

    outputfile.write("\n")
    outputfile.write("Participant " + str(num) + "\n" + "\n")

    outputfile.write("Excluded over 200 : " + str(excluded_over_200) + "       percent: " +
str(perc_over_200) + "\n")
    outputfile.write("Excluded under 40 : " + str(excluded_under_40) + "       percent: " +
str(perc_under_40) + "\n")
    outputfile.write("Excluded unknown  : " + str(excluded_unknown) + "       percent: " +
str(perc_unknown) + "\n" + "\n")

    outputfile.write("Counter_40_to_49  : " + str(counter_40_to_49) + "       percent: " +
str(perc_40_to_49) + "\n")
    outputfile.write("Counter_50_to_59  : " + str(counter_50_to_59) + "    percent: " + str(
perc_50_to_59) + "\n")
    outputfile.write("Counter_60_to_100 : " + str(counter_60_to_100) + "  percent: " + str(
perc_60_to_100) + "\n")
    outputfile.write("Counter_101_to_120: " + str(counter_101_to_120) + "   percent: " + str
(perc_101_to_120) + "\n")
    outputfile.write("Counter_121_to_150: " + str(counter_121_to_150) + "    percent: " +
str(perc_121_to_150) + "\n")
    outputfile.write("Counter_151_to_200: " + str(counter_151_to_200) + "     percent: " +
str(perc_151_to_200) + "\n" + "\n")

    outputfile.write("There were " + str(counter_bigger_30) + " (" + str(perc_bigger_30) + "
%) values with a difference of more than 3 in a second." + "\n")
    outputfile.write("From these, " + str(counter_bigger_50) + " (" + str(perc_bigger_50) +
"%) values had a difference which is bigger than 5 in a second." + "\n")
    outputfile.write("The following value-pairs are measured 1 second after another" + "\n")
    outputfile.write("and have at least a difference from more than 5 bpm: " + "\n")

    for x, y in bigger_50.items():
        outputfile.write(str(x) + " - " + str(y))
        outputfile.write("\n")

    outputfile.write("\n")

    # creating a figure
    length = len(data_list)
    x_axis = range(length)
    y_axis = data_list

    fig = plt.figure()
    plt.plot(x_axis, y_axis)
    plt.xlabel("time (s)")
    plt.ylabel("Heart rate (bpm)")
    plt.title("HR (bpm) of Participant " + str(num))
    plt.savefig("HR_of_Participant " + str(num))
    plt.show()
    plt.close(fig)

    f.close()

    # reset variables & dicts
    excluded_over_200   = 0
    excluded_under_40   = 0
    excluded_unknown    = 0

    counter_40_to_49    = 0
    counter_50_to_59    = 0
    counter_60_to_100   = 0
    counter_101_to_120  = 0
    counter_121_to_150  = 0
    counter_151_to_200  = 0

    counter_bigger_30   = 0
    counter_bigger_50   = 0

    index_counter       = 0
    length_counter      = 0

    del data_list[:]
    bigger_30.clear()
    bigger_50.clear()

# create numpy arrays + computation of mean and std
np_total_over_200   = np.array(map(float, total_over_200))
np_total_under_40   = np.array(map(float, total_under_40))
np_total_unknown    = np.array(map(float, total_unknown))
np_total_40_to_49   = np.array(map(float, total_40_to_49))
np_total_50_to_59   = np.array(map(float, total_50_to_59))
np_total_60_to_100  = np.array(map(float, total_60_to_100))
np_total_101_to_120 = np.array(map(float, total_101_to_120))
np_total_121_to_150 = np.array(map(float, total_121_to_150))
np_total_151_to_200 = np.array(map(float, total_151_to_200))
np_total_bigger_30  = np.array(map(float, total_bigger_30))
np_total_bigger_50  = np.array(map(float, total_bigger_50))

mean_total_over_200   = np.mean(np_total_over_200)
mean_total_under_40   = np.mean(np_total_under_40)
mean_total_unknown    = np.mean(np_total_unknown)
mean_total_40_to_49   = np.mean(np_total_40_to_49)
mean_total_50_to_59   = np.mean(np_total_50_to_59)
mean_total_60_to_100  = np.mean(np_total_60_to_100)
mean_total_101_to_120 = np.mean(np_total_101_to_120)
mean_total_121_to_150 = np.mean(np_total_121_to_150)
mean_total_151_to_200 = np.mean(np_total_151_to_200)
mean_total_bigger_30  = np.mean(np_total_bigger_30)
mean_total_bigger_50  = np.mean(np_total_bigger_50)
```

```python

std_total_over_200   = np.std(np_total_over_200)
std_total_under_40   = np.std(np_total_under_40)
std_total_unknown    = np.std(np_total_unknown)
std_total_40_to_49   = np.std(np_total_40_to_49)
std_total_50_to_59   = np.std(np_total_50_to_59)
std_total_60_to_100  = np.std(np_total_60_to_100)
std_total_101_to_120 = np.std(np_total_101_to_120)
std_total_121_to_150 = np.std(np_total_121_to_150)
std_total_151_to_200 = np.std(np_total_151_to_200)
std_total_bigger_30  = np.std(np_total_bigger_30)
std_total_bigger_50  = np.std(np_total_bigger_50)

# write in outputfile
outputfile.write("\n")
outputfile.write("Total Percentages" + "\n" + "\n")

outputfile.write("Excluded over 200 : mean:" + str(mean_total_over_200) + "        std: " + str(std_total_over_200) + "\n")
outputfile.write("Excluded under 40 : mean:" + str(mean_total_under_40) + "        std: " + str(std_total_under_40) + "\n")
outputfile.write("Excluded unknown  : mean:" + str(mean_total_unknown) + "        std: " + str(std_total_unknown) + "\n" + "\n")

outputfile.write("Counter_40_to_49  : mean:" + str(mean_total_40_to_49) + "        std: " + str(std_total_40_to_49) + "\n")
outputfile.write("Counter_50_to_59  : mean:" + str(mean_total_50_to_59) + "    std: " + str(std_total_50_to_59) + "\n")
outputfile.write("Counter_60_to_100 : mean:" + str(mean_total_60_to_100) + "  std: " + str(std_total_60_to_100) + "\n")
outputfile.write("Counter_101_to_120: mean:" + str(mean_total_101_to_120) + "   std: " + str(std_total_101_to_120) + "\n")
outputfile.write("Counter_121_to_150: mean:" + str(mean_total_121_to_150) + "    std: " + str(std_total_121_to_150) + "\n")
outputfile.write("Counter_151_to_200: mean:" + str(mean_total_151_to_200) + "     std: " + str(std_total_151_to_200) + "\n" + "\n")

outputfile.write("Counter_bigger_30 : mean:" + str(mean_total_bigger_30) + "   std:" + str(std_total_bigger_30) + "\n")
outputfile.write("Counter_bigger_50 : mean:" + str(mean_total_bigger_50) + "   std:" + str(std_total_bigger_50) + "\n")

outputfile.close()

print(" ")
print("--------------------------------------------------------------------------------")
print(" ")
if total_files == 1:
    print("A document with the name: p_1_HR_total_distribution_of_data.txt was created.")
    print(" ")
    print("Furthermore, a figure with the name: HR_of_Participant1.png was created.")
else:
    print("A document with the name: p_1_to_" + str(total_files) + "_HR_total_distribution_of_data.txt was created.")
    print(" ")
    print("Furthermore, " + str(total_files) + " figures with names ranging from: HR_of_Participant1.png to HR_of_Participant" + str(total_files) + ".png were created.")
print(" ")
print("--------------------------------------------------------------------------------")
print("-------------------- programmed by Niklas Enewoldsen ---------------------------")
print("--------------------------------------------------------------------------------")
```

Appendix N - Program: Distribution & Artifact Detection of IBI Data

Program Description. The program is suitable for the analysis of inter-beat-interval data regarding its distribution and possible outliners. Thereby, inter-beat-interval data is categorized in over ten different categories per participant the percentage of these categories is displayed. Furthermore, there is a total statistic section at the end where the average amount of measurements in a category will be displayed, as well as the associated standard deviation. To use this program to its fully potential, here are a few steps you have to make in order to use the program properly:

1. The program was written in python, thus make sure you have python installed on your operating system.
2. First, you have to rename the files. The program was written in order to analyze the total files of a participant which means that for every participant there is just one file. Therefore, you have to rename the files in the following order: first the „p“ plus the number of the participant, then the word total combined with the type of file, for example „p_1_totalIBI.csv“. If you have four IBI files containing all the IBI data from four different participants, you name them ranging from „p_1_totalIBI.csv“ to „p_4_totalIBI.csv“.
3. Set the file-path and make sure that all the files you want to merge are in the same directory.
4. Set the amount of participants. Note that regarding the name of the files, these have to begin with one and have to chronologically increase.

The program will then begin to analyze the files and will create one new file with the name „p_1_to_x_IBI_distribution.txt“ which consists of the amount of signals per category, as well as the associated percentage. In particular, for each participant the percentage of each category is displayed which is followed by the total average and standard deviation per category. This will provide you with a good overview over the distribution of the inter-beat-interval data.

```python
# -*- coding: utf-8 -*-
"""
Created on Sun May 22 18:57:38 2016

@author: ene
"""

import os
import numpy as np

os.chdir("/Users/ene/Documents/Ledalab")

participant = 8

counter_under_00 = 0
counter_00_to_02 = 0
counter_02_to_04 = 0
counter_04_to_05 = 0
counter_05_to_06 = 0
counter_06_to_07 = 0
counter_07_to_10 = 0
counter_10_to_13 = 0
counter_13_to_14 = 0
counter_14_to_15 = 0
counter_over_15  = 0

total_under_00 = []
total_00_to_02 = []
total_02_to_04 = []
total_04_to_05 = []
total_05_to_06 = []
total_06_to_07 = []
total_07_to_10 = []
total_10_to_13 = []
total_13_to_14 = []
total_14_to_15 = []
total_over_15  = []

counter_unknown  = 0
unknown_values   = []

counter_length  = 0

total_files = 8

for num in range(1, total_files + 1):
    b = np.genfromtxt(r"/Users/ene/Documents/Ledalab/p_" + str(num) + "_totalIBI.csv",
    delimiter = ",", names = True, dtype = None)

    ibi = b["IBI"]

    for i in ibi:
        counter_length += 1
        if float(i) >= 1.5:
            counter_over_15 += 1
        elif float(i) <= 0:
            counter_under_00 += 1
        elif float(i) > 0 and float(i) < 0.2:
            counter_00_to_02 += 1
        elif float(i) >= 0.2 and float(i) < 0.4:
            counter_02_to_04 += 1
        elif float(i) >= 0.4 and float(i) <= 0.5:
            counter_04_to_05 += 1
        elif float(i) <= 0.6 and float(i) > 0.5:
            counter_05_to_06 += 1
        elif float(i) >= 1.4 and float(i) < 1.5:
            counter_14_to_15 += 1
        elif float(i) <= 0.7 and float(i) > 0.6:
            counter_06_to_07 += 1
        elif float(i) >= 1.3 and float(i) < 1.4:
            counter_13_to_14 += 1
        elif float(i) <= 1.0 and float(i) > 0.7:
            counter_07_to_10 += 1
        elif float(i) >= 1.0 and float(i) < 1.3:
            counter_10_to_13 += 1
        else:
            counter_unknown += 1
            unknown_values.append(i)

    # compute percentages
    one_percent   = float(counter_length) / 100.00
    perc_under_00 = "%.2f" % (float(counter_under_00) / float(one_percent))
    perc_00_to_02 = "%.2f" % (float(counter_00_to_02) / float(one_percent))
    perc_02_to_04 = "%.2f" % (float(counter_02_to_04) / float(one_percent))
    perc_04_to_05 = "%.2f" % (float(counter_04_to_05) / float(one_percent))
    perc_05_to_06 = "%.2f" % (float(counter_05_to_06) / float(one_percent))
    perc_06_to_07 = "%.2f" % (float(counter_06_to_07) / float(one_percent))
    perc_07_to_10 = "%.2f" % (float(counter_07_to_10) / float(one_percent))
    perc_10_to_13 = "%.2f" % (float(counter_10_to_13) / float(one_percent))
    perc_13_to_14 = "%.2f" % (float(counter_13_to_14) / float(one_percent))
    perc_14_to_15 = "%.2f" % (float(counter_14_to_15) / float(one_percent))
    perc_over_15  = "%.2f" % (float(counter_over_15)  / float(one_percent))

    total_under_00.append(perc_under_00)
    total_00_to_02.append(perc_00_to_02)
    total_02_to_04.append(perc_02_to_04)
    total_04_to_05.append(perc_04_to_05)
    total_05_to_06.append(perc_05_to_06)
    total_06_to_07.append(perc_06_to_07)
    total_07_to_10.append(perc_07_to_10)
    total_10_to_13.append(perc_10_to_13)
    total_13_to_14.append(perc_13_to_14)
    total_14_to_15.append(perc_14_to_15)
    total_over_15.append(perc_over_15)

    if total_files == 8:
        outputfile = open("p_1_to_8_IBI_distribution.txt", "a")
    else:
        outputfile = open("p_1_to_" + str(num) + "_IBI_distribution.txt", "a")

    # write in outputfile
    outputfile.write("Participant " + str(num) + " - IBI Distribution")
    outputfile.write("\n")
    outputfile.write("\n")

    outputfile.write("In total, the file consists of " + str(counter_length) + " measurements.")
    outputfile.write("Several files were empty, thus they are not included in this overview.")
    outputfile.write("Check the Starttime, Endtime & Duration File to see which files were empty.")
    outputfile.write("\n")
    outputfile.write("\n")

    outputfile.write("      Under 0.0: " + str(counter_under_00) + " percent: " + str(perc_under_00) + "\n")
    outputfile.write("From 0.0 to 0.2: " + str(counter_00_to_02) + " percent: " + str(
    perc_00_to_02) + "\n")
    outputfile.write("From 0.2 to 0.4: " + str(counter_02_to_04) + " percent: " + str(
    perc_02_to_04) + "\n")
    outputfile.write("From 0.4 to 0.5: " + str(counter_04_to_05) + " percent: " + str(
    perc_04_to_05) + "\n")
    outputfile.write("From 0.5 to 0.6: " + str(counter_05_to_06) + " percent: " + str(
    perc_05_to_06) + "\n")
    outputfile.write("From 0.6 to 0.7: " + str(counter_06_to_07) + " percent: " + str(
    perc_06_to_07) + "\n")
    outputfile.write("From 0.7 to 1.0: " + str(counter_07_to_10) + " percent: " + str(
    perc_07_to_10) + "\n")
    outputfile.write("From 1.0 to 1.3: " + str(counter_10_to_13) + " percent: " + str(
    perc_10_to_13) + "\n")
    outputfile.write("From 1.3 to 1.4: " + str(counter_13_to_14) + " percent: " + str(
    perc_13_to_14) + "\n")
    outputfile.write("From 1.4 to 1.5: " + str(counter_14_to_15) + " percent: " + str(
    perc_14_to_15) + "\n")
    outputfile.write("       Over 1.5: " + str(counter_over_15)  + " percent: " + str(
    perc_over_15) + "\n")
    outputfile.write("\n")

    # clean variables & lists
    ibi = []

    counter_length   = 0

    counter_under_00 = 0
    counter_00_to_02 = 0
    counter_02_to_04 = 0
    counter_04_to_05 = 0
    counter_05_to_06 = 0
    counter_06_to_07 = 0
    counter_07_to_10 = 0
    counter_10_to_13 = 0
    counter_13_to_14 = 0
    counter_14_to_15 = 0
    counter_over_15  = 0

# compute total mean & std
np_total_under_00 = np.array(map(float, total_under_00))
np_total_00_to_02 = np.array(map(float, total_00_to_02))
np_total_02_to_04 = np.array(map(float, total_02_to_04))
np_total_04_to_05 = np.array(map(float, total_04_to_05))
np_total_05_to_06 = np.array(map(float, total_05_to_06))
np_total_06_to_07 = np.array(map(float, total_06_to_07))
np_total_07_to_10 = np.array(map(float, total_07_to_10))
np_total_10_to_13 = np.array(map(float, total_10_to_13))
np_total_13_to_14 = np.array(map(float, total_13_to_14))
np_total_14_to_15 = np.array(map(float, total_14_to_15))
np_total_over_15  = np.array(map(float, total_over_15))

mean_total_under_00 = np.mean(np_total_under_00)
mean_total_00_to_02 = np.mean(np_total_00_to_02)
mean_total_02_to_04 = np.mean(np_total_02_to_04)
mean_total_04_to_05 = np.mean(np_total_04_to_05)
mean_total_05_to_06 = np.mean(np_total_05_to_06)
mean_total_06_to_07 = np.mean(np_total_06_to_07)
mean_total_07_to_10 = np.mean(np_total_07_to_10)
mean_total_10_to_13 = np.mean(np_total_10_to_13)
mean_total_13_to_14 = np.mean(np_total_13_to_14)
mean_total_14_to_15 = np.mean(np_total_14_to_15)
mean_total_over_15  = np.mean(np_total_over_15)
```

```

std_total_under_00 = np.std(np_total_under_00)
std_total_00_to_02 = np.std(np_total_00_to_02)
std_total_02_to_04 = np.std(np_total_02_to_04)
std_total_04_to_05 = np.std(np_total_04_to_05)
std_total_05_to_06 = np.std(np_total_05_to_06)
std_total_06_to_07 = np.std(np_total_06_to_07)
std_total_07_to_10 = np.std(np_total_07_to_10)
std_total_10_to_13 = np.std(np_total_10_to_13)
std_total_13_to_14 = np.std(np_total_13_to_14)
std_total_14_to_15 = np.std(np_total_14_to_15)
std_total_over_15  = np.std(np_total_over_15)

outputfile.write("Total Statistics:" + "\n" + "\n")
outputfile.write("      Under 0.0: mean:" + str(mean_total_under_00) + " std: " + str(
    std_total_under_00) + "\n")
outputfile.write("From 0.0 to 0.2: mean:" + str(mean_total_00_to_02) + " std: " + str(
    std_total_00_to_02) + "\n")
outputfile.write("From 0.2 to 0.4: mean:" + str(mean_total_02_to_04) + " std: " + str(
    std_total_02_to_04) + "\n")
outputfile.write("From 0.4 to 0.5: mean:" + str(mean_total_04_to_05) + " std: " + str(
    std_total_04_to_05) + "\n")
outputfile.write("From 0.5 to 0.6: mean:" + str(mean_total_05_to_06) + " std: " + str(
    std_total_05_to_06) + "\n")
outputfile.write("From 0.6 to 0.7: mean:" + str(mean_total_06_to_07) + " std: " + str(
    std_total_06_to_07) + "\n")
outputfile.write("From 0.7 to 1.0: mean:" + str(mean_total_07_to_10) + " std: " + str(
    std_total_07_to_10) + "\n")
outputfile.write("From 1.0 to 1.3: mean:" + str(mean_total_10_to_13) + " std: " + str(
    std_total_10_to_13) + "\n")
outputfile.write("From 1.3 to 1.4: mean:" + str(mean_total_13_to_14) + " std: " + str(
    std_total_13_to_14) + "\n")
outputfile.write("From 1.4 to 1.5: mean:" + str(mean_total_14_to_15) + " std: " + str(
    std_total_14_to_15) + "\n")
outputfile.write("       Over 1.5: mean:" + str(mean_total_over_15)  + " std: " + str(
    std_total_over_15) + "\n")
outputfile.write("\n")

outputfile.close()
```

Appendix O - Program: Distribution & Artifact Detection of BVP Data

Program Description. The program is suitable for the analysis of blood-volume pulse data regarding its distribution and possible outliners. Thereby, blood-volume pulse data is categorized in over ten different categories per participant the percentage of these categories is displayed. Furthermore, there is a total statistic section at the end where the average amount of measurements in a category will be displayed, as well as the associated standard deviation. The program also provides the user with a figure per participant where the whole recording of the blood-volume pulse data per participant can be seen. To use this program to its fully potential, here are a few steps you have to make in order to use the program properly:

1. The program was written in python, thus make sure you have python installed on your operating system.
2. First, you have to rename the files. The program was written in order to analyze the total files of a participant which means that for every participant there is just one file. Therefore, you have to rename the files in the following order: first the „p“ plus the number of the participant, then the word total combined with the type of file, for example „p_1_totalBVP.csv“. If you have four BVP files containing all the IBI data from four different participants, you name them ranging from „p_1_totalBVP.csv“ to „p_4_totalBVP.csv“.
3. Set the file-path and make sure that all the files you want to merge are in the same directory.
4. Set the amount of participants. Note that regarding the name of the files, these have to begin with one and have to chronologically increase.

The program will then begin to analyze the files and will create one new file with the name „p_1_to_x_IBI_distribution.txt“ which consists of the amount of signals per category, as well as the associated percentage. In particular, for each participant the percentage of each category is displayed which is followed by the total average and standard deviation per category. This will provide you with a good overview over the distribution of the inter-beat-interval data. Furthermore, for a graph for each participant is provided and saved as a png in the same directory as the output-file.

```python
# -*- coding: utf-8 -*-
"""
Created on Sun May 22 11:37:41 2016

@author: ene
"""

import os
import numpy as np
import matplotlib.pyplot as plt

os.chdir("/Users/ene/Documents/Ledalab")

excluded_over_200    = 0
excluded_under_m200  = 0
excluded_over_300    = 0
excluded_under_m300  = 0
excluded_unknown     = 0

counter_m200_to_m100 = 0
counter_m100_to_m50  = 0
counter_m50_to_0     = 0
counter_0_to_50      = 0
counter_50_to_100    = 0
counter_100_to_200   = 0

total_over_200     = []
total_under_m200   = []
total_over_300     = []
total_under_m300   = []
total_m200_to_m100 = []
total_m100_to_m50  = []
total_m50_to_0     = []
total_0_to_50      = []
total_50_to_100    = []
total_100_to_200   = []
total_unknown      = []

length_counter = 0

total_files = 8

for num in range(1, total_files + 1):
    if total_files == 1:
        outputfile = open("p_1_BVP_total_distribution_of_data.txt", "a")
    else:
        outputfile = open("p_1_to_" + str(total_files) + "_BVP_total_distribution_of_data.
    txt", "a")

    f = open("p_" + str(num) + "_totalBVP.csv")
    data_list  = []
    unknown_values = []

    for line in f:
        data_list.append(line)

    f.close()

    for line in data_list:
        length_counter += 1

        if float(line) >= 300:
            excluded_over_300 += 1
        elif float(line) <= -300:
            excluded_under_m300 += 1
        elif float(line) <= -200 and float(line) > -300:
            excluded_under_m200 += 1
        elif float(line) >= 200 and float(line) < 300:
            excluded_over_200 += 1
        elif float(line) <= -100 and float(line) > -200:
            counter_m200_to_m100 += 1
        elif float(line) >= 100 and float(line) < 200:
            counter_100_to_200 += 1
        elif float(line) <= -50 and float(line) > -100:
            counter_m100_to_m50 += 1
        elif float(line) >= 50 and float(line) < 100:
            counter_50_to_100 += 1
        elif float(line) < 0 and float(line) > -50:
            counter_m50_to_0 += 1
        elif float(line) >= 0 and float(line) < 50:
            counter_0_to_50 += 1
        else:
            excluded_unknown += 1
            unknown_values.append(line)

    # compute precentage
    one_percent       = float(length_counter) / 100.00
    perc_over_300     = "%.2f" % (float(excluded_over_300)    / float(one_percent))
    perc_under_m300   = "%.2f" % (float(excluded_under_m300)  / float(one_percent))
    perc_unknown      = "%.2f" % (float(excluded_unknown)     / float(one_percent))
    perc_under_m200   = "%.2f" % (float(excluded_under_m200)  / float(one_percent))
    perc_m200_to_m100 = "%.2f" % (float(counter_m200_to_m100) / float(one_percent))
    perc_m100_to_m50  = "%.2f" % (float(counter_m100_to_m50)  / float(one_percent))
    perc_m50_to_0     = "%.2f" % (float(counter_m50_to_0)     / float(one_percent))
    perc_0_to_50      = "%.2f" % (float(counter_0_to_50)      / float(one_percent))
    perc_50_to_100    = "%.2f" % (float(counter_50_to_100)    / float(one_percent))
    perc_100_to_200   = "%.2f" % (float(counter_100_to_200)   / float(one_percent))
    perc_over_200     = "%.2f" % (float(excluded_over_200)    / float(one_percent))

    # for total computation of mean & std
    total_over_200.append(perc_over_200)
    total_under_m200.append(perc_under_m200)
    total_over_300.append(perc_over_300)
    total_under_m300.append(perc_under_m300)
    total_m200_to_m100.append(perc_m200_to_m100)
    total_m100_to_m50.append(perc_m100_to_m50)
    total_m50_to_0.append(perc_m50_to_0)
    total_0_to_50.append(perc_0_to_50)
    total_50_to_100.append(perc_50_to_100)
    total_100_to_200.append(perc_100_to_200)
    total_unknown.append(perc_unknown)

    # write in outputfile
    outputfile.write("\n")
    outputfile.write("Participant " + str(num) + "\n" + "\n")

    outputfile.write("Excluded under -300: " + str(excluded_under_m300) + "  percent: " + str(perc_under_m300) + "\n")
    outputfile.write("From -300 to -200: " + str(excluded_under_m200) + "  percent: " + str(perc_under_m200) + "\n")
    outputfile.write("From -200 to -100: " + str(counter_m200_to_m100) + "  percent: " + str(perc_m200_to_m100) + "\n")
    outputfile.write("From -100 to -50: " + str(counter_m100_to_m50) + "  percent: " + str(perc_m100_to_m50) + "\n")
    outputfile.write("From -50 to +0: " + str(counter_m50_to_0) + "  percent: " + str(perc_m50_to_0) + "\n")
    outputfile.write("From +0 to +50: " + str(counter_0_to_50) + "  percent: " + str(
    perc_0_to_50) + "\n")
    outputfile.write("From +50 to +100: " + str(counter_50_to_100) + "  percent: " + str(
    perc_50_to_100) + "\n")
    outputfile.write("From +100 to +200: " + str(counter_100_to_200) + "  percent: " + str(
    perc_100_to_200) + "\n")
    outputfile.write("From +200 to +300: " + str(excluded_over_200) + "  percent: " + str(
    perc_over_200) + "\n")
    outputfile.write("Excluded over +300: " + str(excluded_over_300) + "  percent: " + str(
    perc_over_300) + "\n")
    outputfile.write("Excluded unknown values: " + str(excluded_unknown) + "\n" + "\n")
    outputfile.write("The following values are unknown and probably artifacts: " + "\n")

    outputfile.write("\n")
    outputfile.write("\n")

    for x in unknown_values:
        outputfile.write(str(x))
        outputfile.write("\n")

    # plot figure
    length = len(data_list)
    x_axis = range(length)
    y_axis = data_list

    plt.plot(x_axis, y_axis)
    plt.xlabel("time (s)")
    plt.ylabel("bloodvolume pulse")
    plt.title("BVP of Participant " + str(num))
    plt.savefig("BVP_of_Participant " + str(num))
    plt.show()

    # clear variables
    excluded_over_200    = 0
    excluded_under_m200  = 0
    excluded_over_300    = 0
    excluded_under_m300  = 0
    excluded_unknown     = 0

    counter_m200_to_m100 = 0
    counter_m100_to_m50  = 0
    counter_m50_to_0     = 0
    counter_0_to_50      = 0
    counter_50_to_100    = 0
    counter_100_to_200   = 0

    length_counter = 0

    del data_list[:]
    del unknown_values[:]

# compute total mean & std
np_total_over_200     = np.array(map(float, total_over_200))
np_total_under_m200   = np.array(map(float, total_under_m200))
np_total_over_300     = np.array(map(float, total_over_300))
np_total_under_m300   = np.array(map(float, total_over_300))
np_total_m200_to_m100 = np.array(map(float, total_m200_to_m100))
np_total_m100_to_m50  = np.array(map(float, total_m100_to_m50))
np_total_m50_to_0     = np.array(map(float, total_m50_to_0))
np_total_0_to_50      = np.array(map(float, total_0_to_50))
np_total_50_to_100    = np.array(map(float, total_50_to_100))
np_total_100_to_200   = np.array(map(float, total_100_to_200))
np_total_unknown      = np.array(map(float, total_unknown))
```

```

mean_total_over_200      = np.mean(np_total_over_200)
mean_total_under_m200    = np.mean(np_total_under_m200)
mean_total_over_300      = np.mean(np_total_over_300)
mean_total_under_m300    = np.mean(np_total_under_m300)
mean_total_m200_to_m100 = np.mean(np_total_m200_to_m100)
mean_total_m100_to_m50  = np.mean(np_total_m100_to_m50)
mean_total_m50_to_0      = np.mean(np_total_m50_to_0)
mean_total_0_to_50       = np.mean(np_total_0_to_50)
mean_total_50_to_100     = np.mean(np_total_50_to_100)
mean_total_100_to_200    = np.mean(np_total_100_to_200)
mean_total_unknown       = np.mean(np_total_unknown)

std_total_over_200      = np.std(np_total_over_200)
std_total_under_m200    = np.std(np_total_under_m200)
std_total_over_300      = np.std(np_total_over_300)
std_total_under_m300    = np.std(np_total_under_m300)
std_total_m200_to_m100 = np.std(np_total_m200_to_m100)
std_total_m100_to_m50  = np.std(np_total_m100_to_m50)
std_total_m50_to_0      = np.std(np_total_m50_to_0)
std_total_0_to_50       = np.std(np_total_0_to_50)
std_total_50_to_100     = np.std(np_total_50_to_100)
std_total_100_to_200    = np.std(np_total_100_to_200)
std_total_unknown       = np.std(np_total_unknown)

outputfile.write("\n")
outputfile.write("Total percentages (mean & std)" + "\n" + "\n")
outputfile.write("Excluded under -300: mean: " + str(mean_total_under_m300) + "  std: " +
    str(std_total_under_m300) + "\n")
outputfile.write("From -300 to -200: mean: " + str(mean_total_under_m200) + "  std: " + str(
    std_total_under_m200) + "\n")
outputfile.write("From -200 to -100: mean: " + str(mean_total_m200_to_m100) + "  std: " +
    str(std_total_m200_to_m100) + "\n")
outputfile.write("From -100 to -50: mean: " + str(mean_total_m100_to_m50) + "  std: " + str(
    std_total_m100_to_m50) + "\n")
outputfile.write("From -50 to +0: mean: " + str(mean_total_m50_to_0) + "  std: " + str(
    std_total_m50_to_0) + "\n")
outputfile.write("From +0 to +50: mean: " + str(mean_total_0_to_50) + "  std: " + str(
    std_total_0_to_50) + "\n")
outputfile.write("From +50 to +100: mean: " + str(mean_total_50_to_100) + "  std: " + str(
    std_total_50_to_100) + "\n")
outputfile.write("From +100 to +200: mean: " + str(counter_100_to_200) + "  std: " + str(
    std_total_100_to_200) + "\n")
outputfile.write("From +200 to +300: mean: " + str(mean_total_over_200) + "  std: " + str(
    std_total_over_200) + "\n")
outputfile.write("Excluded over +300: mean: " + str(mean_total_over_300) + "  std: " + str(
    std_total_over_300) + "\n")
outputfile.write("Excluded unknown values: mean: " + str(mean_total_unknown) + "  std: " +
    str(std_total_unknown) + "\n" + "\n")

outputfile.close()
```

Appendix P - Noise-Analysis: Percentages per Participant

The table shows the percentages per participant for each category: clean signals, questionable signals and noise

	Clean		Noise		Questionable		Total
	n	%	n	%	n	%	
Participant 1	37243	88.30	4628	10.97	305	0.72	42176
Participant 2	57230	94.22	1338	2.20	2176	3.58	60744
Participant 3	47236	76.72	11415	18.54	2921	4.74	61572
Participant 4	57466	92.68	3551	5.73	987	1.59	62004
Participant 5	80319	92.02	6450	7.39	511	0.59	87280
Participant 6	48055	89.71	5379	10.04	134	0.25	53568
Participant 7	59877	94,24	2173	3.42	1490	2.34	63540
Participant 8	23.683	93.04	1770	6.95	2	0.01	25455

Appendix Q - Peak-Analysis: Averages & Settings of essential EDA-variables

The table shows the mean values of the following EDA-variables gathered through the peak-analysis: EDA, Rise-time, Maximum-derivative, Amplitude, Decay-time, SCR-width & Area under the curve

	EDA (µS)	Rise-time (s)	Maximum-derivative (µS/s)	Amplitude (µS)	Decay-time (s)	SCR-width (s)	Area under the curve (s)
P1	0.4219	1.3708	0.1188	0.0815	3.0176	2.3084	0.2796
P2	1.4622	1.4165	0.1462	0.1087	3.1038	2.6668	0.4270
P3	3.9040	1.3790	0.5803	0.4324	2.1756	2.3357	1.3130
P4	1.9302	1.4761	0.2706	0.1978	3.2214	2.6082	0.7397
P5	0.8569	1.4113	0.1635	0.1146	2.7368	2.4873	0.4180
P6	0.5922	1.3725	0.1577	0.1023	2.6390	2.4722	0.3368
P7	1.3137	1.4687	0.1548	0.1142	3.0414	2.6936	0.4155
P8	0.1086	1.1766	0.1268	0.0653	2.1197	2.1410	0.1709

Note: The peak-analysis was conducted with the web-application of the EDA-Explorer. Further explanation and the actual settings of the variables used by the analysis can be found the table below. All values presented in this table are rounded up to four decimal after the comma.

The table contains the peak-extraction-settings which were used for the peak-analysis plus a description of the variables

	Value	Description
Minimum Amplitude	0.01	The minimum amplitude of potential SCR must reach in order to be counted as an SCR.
Offset	0.8	The number of seconds for which the derivative must be positive before a peak and the number of seconds for which the derivative must be negative after a peak.
Filter Frequency	1.0	The filter frequency is the cutoff frequency (in Hz) of this filter (use of Butterworth lowpass filter).
Filter Order	6	The filter order is the number of poles/zeros in the filter (use of Butterworth lowpass filter). The higher the order the steeper the cutoff on the filter; however, higher orders take longer to compute.
Max Rise Time	4	The maximum number of seconds before the apex of a peak that is the "start" of the peak.
Max Decay Time	4	The maximum number of seconds after the apex of a peak that is the "rec.t/2" of the peak, 50% of amp.

Appendix R - Event-related Analysis: SCRs/min per Participant & Condition

The table contains the average SCRs per minute per participant of the event-related analysis with TTP- and CDA-measures for the minute before participants started drinking, the first minute of drinking and random events

	Before		While		Random	
	TTP	CDA	TTP	CDA	TTP	CDA
Participant 1	3.0	11.0	6.0	6.0	4.5	9.2
Participant 2	6.8	11.3	6.8	10.3	6.4	10.8
Participant 3	6.7	9.6	6.5	8.9	4.1	6.9
Participant 4	10.3	13,4	9.4	14.4	5.7	8.7
Participant 5	11.3	13.7	11.7	15.2	9.3	11.8
Participant 6	7.8	10.8	7.8	12.7	5.4	9.3
Participant 7	4.3	7.0	5.0	8.4	8.3	10.5
Participant 8	4.5	8.8	4.7	7.9	1.0	1.1

Note: Each analysis was carried out with a sample-rate of 1 Hz. The event-related SCRs had a minimum amplitude threshold of 0.01µS and a response-window of 1 minute. All values presented in this table are rounded up to one decimal after the comma.