Validation of a Wrist-Worn Photoplethysmographic Sensor (E4): Comparison of Finger and Wrist

N.M. Enewoldsen | s1481436

Enschede, September 2021

Master Thesis

Human Factors & Engineering Psychology

Faculty of Behavioural, Management & Social Sciences

1^{st} Supervisor: Prof. Dr. J.M.C. Schraagen

2^{nd} Supervisor: Dr. R.H.J. van der Lubbe

UNIVERSITY OF TWENTE.

Table of Contents

Abstract

Background: The fast rise of wearable technology has led to the development of photoplethysmography (PPG) sensors which enable the monitoring of physical and emotional states (e.g., stress) of users in real-time and in daily life. Thereby, detection and management of stress is a key component of the research, due to the detrimental negative effects of stress on health. However, studies have shown the sensors' vulnerability towards artifacts, especially motion artifacts, which emphasize the need for accurate validity assessments. The present study implements the validity assessment protocol of van Lier et al. (2019), whereby the wrist-worn PPG sensor of the Empatica E4 is validated by comparing it to a PPG sensor at the fingertip which reflects the reference device (RD).

Methods: A modified version of the Sing-A-Song-Stress task (SSST) and the Stroop task were used as social and cognitive stressor, in order to induce different intensities of stress. Beside the recording of the physiological arousal, the State Trait Anxiety Inventory (STAI) was used to measure the subjective stress experience. The analysis was performed at the signal, parameter and event level, enabling a comprehensive and standardized validation of the E4.

Results: The successful induction of subjective stress for both tasks could not be measured by the physiological measurements. On the one hand, the strict quality inclusion criteria for the analysis resulted in the exclusion of around 40% of the data, while on the other hand, the stressors were found to be not strong enough to induce detectable physiological arousal. Furthermore, at the signal level of the PPG, no relationship was found between the measurements of the two sensors, which is supported by the differences in signal, demonstrated by the analysis at the parameter level.

Conclusions: This study questions the validity of the E4 by showing that the validity is lower than described in the literature. Nevertheless, previous research has shown that the E4 can be used for strong and long-lasting stressors, as well as for averages of HR over a longer period of time. Moreover, the validity assessment protocol showed its potential, by enabling the less time-consuming validation with a PPG sensor as RD instead of an electrocardiogram, consequently, making the protocol more available and less restricting with regard to participants and setting.

Keywords: Wearable Sensor, Photoplethysmography, Heart Rate, Stress, Validity

List of Abbreviations

ANS ....Autonomous Nervous System

BP ....Blood Pressure
BS1 ....Baseline (first)
BS2 ....Baseline (second)
BVP ....Blood Volume Pulse

CVS ....Cardiovascular System

ECG ....Electrocardiogram

HR ....Heart Rate
HRV ....Heart Rate Variability

PNS ....Parasympathetic Nervous System
PPG ....Photoplethysmography
PPI ....Peak-to-Peak Interval

RC1 ....Recovery Phase (first)
RC2 ....Recovery Phase (second)
RD ....Reference Device
RMSSD ....Root Mean Square of Successive Differences

SDNN ....Standard Deviation of Peak-to-Peak Interval
SNS ....Sympathetic Nervous System
SQI ....Signal Quality Index
SSST ....Sing-a-Song Stress Task
ST1 ....Stress Task (first; modified. SSST)
ST2 ....Stress Task (second; Stroop Task)
STAI ....State Trait Anxiety Inventory

UT ....University of Twente

Introduction

Wearable sensors are becoming more widely available and cheaper, offering new opportunities in healthcare by enabling the monitoring of the physical, as well as physiological changes related with the emotional state in daily life and in real-time (Bonato, 2003; Elgendi, 2012; Fletcher et al., 2011; Picard & Healey, 1997; Ragot et al., 2017). These devices are available in many different shapes and forms, equipped with different combinations of sensors (Gu et al., 2009; Poh et al., 2010; Poon et al., 2006). By measuring these physiological parameters in daily life, health informatics can use this information to detect, prevent or treat diseases (Cook et al., 2006; Wilard et al, 2011; Zheng et al., 2014).

Thereby, the measurement of mental stress has become a key component which is associated with physiological and psychological reactions (Chrousos, 2009). Specifically, "stress can be conceptualized as the response of the Autonomic Nervous System (ANS) to various stimuli" (Nath et al., 2020, p.1) including physical, psychological and environmental stimuli (Selvaraj, 2015). There are two types of stress responses of the ANS, namely, the physiological response, which is the effect of stress by physiological activities (e.g., increased heart rate, increased cortisol level, increased sweating), as well as the psychological response which reflects the stress perceived by the individual (e.g., mental stress, emotional stress, perceptual stress) (Nath et al., 2020). In the short term, the activation of stress responses is adaptive, but can become maladaptive if stress responses are repeatedly provoked (Kemeny, 2003). These maladaptive physiological and psychological stress responses can result in various disorders (e.g., hypertension, cardiovascular diseases, anxiety disorders) (Chrousos, 2009; Yoo & Lee, 2011; Zangróniz et al., 2018). Therefore, detection and management of mental stress is essential, in order to prevent those negative effects of mental stress and reduce health risks (Mitsuhashi et al., 2019; Nath et al., 2020; Yoo & Lee, 2011). For the effective detection of stress, various vital signs which were shown to be sensitive to changes in mental stress, are measured by wearables (Abbod et al., 2011; Mitsuhashi et al., 2019; Singh et al., 2011; Yoo & Lee, 2011; Zangróniz et al., 2018).

The pursuit of practical and valid methods to measure stress is reflected by the adoption of multi-sensor technology and the integration of reflective photoplethysmography (PPG) sensors which describe the "optical detection of pulsatile blood volume changes in the vascular bed under the PPG sensor" (Greve et al., 2012, p.1; Elgendi, 2012). Specifically, the PPG sensor measures changes in tissue and blood volume by emitting light into tissue and detect optical variations, in form of light reflected from or transmitted through the tissue (Fusco et al., 2015; Santos et al., 2012; Zhang et al., 2019; Zheng et al., 2014). Compared to conventional plethysmograph methods, the reflective PPG is a non-invasive method, enabling a more practical and user-friendly experience, as well as the use of those sensors in an ambulatory setting.

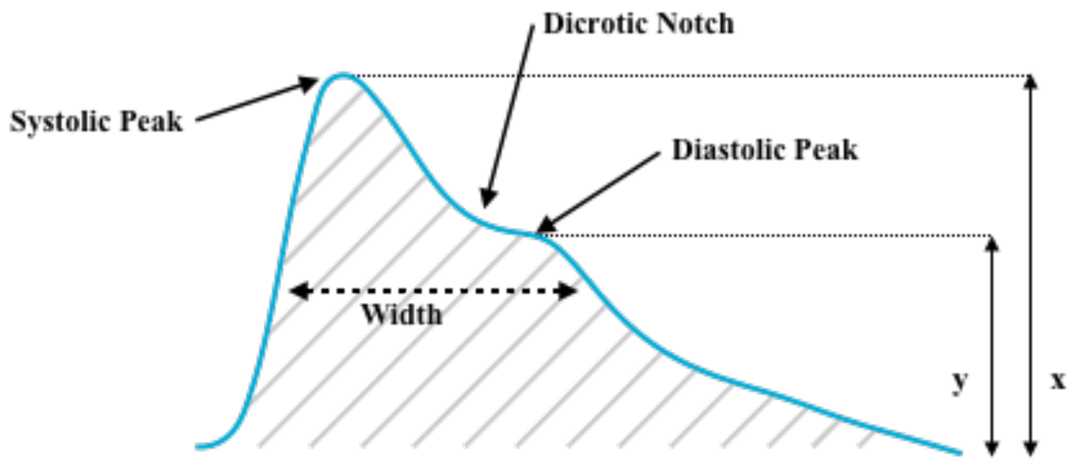
Although studies have investigated the PPG signal and demonstrated satisfactory data quality, high quality could only be achieved in the absence of excessive hand movements or during resting phases (McCarthy et al., 2016; Pietilä et al., 2017). Specifically, the quality of the recorded data is susceptible to contamination, so-called artifacts, which have to be removed, in order to obtain a high quality. With regard to the PPG sensor, their sensitivity to the correct placement of the sensor, movements (physical exercise) and cardiac arrhythmia can influence the rate parameter and reduce its reliability if these artifacts are not removed properly (Allen, 2007; Mullan et al., 2015; Picard & Healey, 1997; van Dooren et al., 2012; Wander & Morrison, 2014; Zheng et al., 2014). Even "the properties of the subject's skin at measurement, including the individual skin structure, the blood oxygen saturation, blood flow rate, skin temperature and the measuring environment" (Elgendi, 2012, p.16) can have a negative impact on the PPG signal. Furthermore, measurements of the PPG sensor are influenced by factors such as emotional states, ambient temperature and fitness level, and they respond and re-stabilize only slowly to them (Picard & Healey, 1997; Tapia et al., 2007). Therefore, several algorithms have been developed which try to remove noise as accurately as possible, whereby some even integrate additional recorded acceleration data as an indicator of the actual movements of the user (Mullan et al., 2015).

Summarized, the susceptibility of wearable PPG sensors towards artifacts and the rapid development of new alternatives emphasize the need for systematic validation of those sensors (Goldsack et al., 2020; Jo et al., 2016; Munos et al., 2016; Shcherbina et al., 2017). Therefore, the measurement of the PPG sensor of interest is compared with the "true" measurement of an already established measurement device, also known as reference device (RD). In particular, the detection of physiological changes due to external stressor levels, measured by the RD and the device of interest, enable a comprehensive validation on multiple levels. The physiological changes, also called events, can be examined by evaluating and comparing the ability of both sensors to significantly detect those specific events (van Lier et al., 2019). Furthermore, the parameter level focusses on whether the similar physiological parameters can be extracted from the recordings of the new device, compared to the recordings of the RD. Finally, the signal level reflects the extent to which the device of interest and the RD produce similar signals which can be determined by assessing the recorded raw data.

Among the measured signals of the PPG is the heart rate variability (HRV), a "popular non-invasive marker of the autonomic nervous system and is widely used to assess cardiac health" (Jeyhani et al., 2015, p.1; Elgendi et al., 2010). HRV reflects the variance of time between consecutive heartbeats (i.e., beat-to-beat variation, also known as inter-beat interval or peak to peak interval; PPI) and provides an indirect insight into the cardiovascular system (CVS) which controls the blood flow through the body, and which can be characterized by measurable parameters such as heart rate (HR), peak to peak interval (PPI), blood pressure (BP) and blood volume pulse (BVP) (Akhter et al., 2016; Mandryk et al., 2006). These cardiovascular variables are fluctuating on a beat-to-beat basis and are coordinated by balanced activity of the two branches of the ANS, the parasympathetic nervous system (PNS) which is responsible for the restoration and conservation of bodily energy, and the sympathetic nervous system (SNS) which increases metabolic output to deal with external challenges and mobilizes the body system during activities, as well as stressful situations (Akther et al., 2016; Appel et al., 1989; Poh et al., 2010). Thus, in preparation for motor action, the sympathetic arousal elevates the HR, BP, sweating and redirects blood toward skeletal muscles, lungs, the heart and the brain, which are also activated in stressful situations when the body is in a fight or flight state (acute stress response) (Yoo & Lee, 2011). By measuring BVP which corresponds to changes in blood flow and HR, the PPG sensor enables the measurement of cardiovascular changes in the ANS caused by stress, as well as the computation of HRV parameter like PPI and HR. This is due to the pulsatile component of the PPG signal which is synchronous with

Figure 1

A typical PPG waveform with its characteristic parameters



Note. The x reflects the amplitude of the systolic peak and y the amplitude of the diastolic peak.

the fluctuations of the beating heart and thus visible in the rise and fall of the PPG signal (Allen, 2007).

The waveform of the PPG signal can be divided into two phases (see Figure 1), the anacrotic phase which is primarily concerned with systole reflection, and the catacrotic phase which describes the diastole and wave reflections (Elgendi, 2012). Beside the parameters shown in Figure 1 (e.g., the systolic amplitude and the pulse width), important features of the PPG signal include the pulse area which reflects the total area under the PPG curve (indicated by the grey lines in Figure 1) and the peak to peak interval (PPI).

With regard to validity assessment of PPG sensors, most research is conducted by comparing the electrocardiogram (ECG), also known as the golden standard of HRV, with the PPG recordings. Whereas RR intervals are a prominent feature of the ECG signal which describes the distance between two successive R peaks, the PPG signal has PPIs which represent the time between the top of the peaks in the BVP signal (Akhter et al., 2016). A great number of publications focused on the correlation between RRIs and PPIs, as well as the HRV parameters derived from PPG and ECG. The majority of these publications demonstrated a high correlation and good agreement with regard to PPG and ECG parameters in various conditions (Bolanos et al., 2006; Elgendi, 2012; Giardino et al., 2002; Greve et al., 2012; Jeyhani et al., 2015; Lin et al., 2014; Selvaraj et al., 2008). Gil et al. (2010) even suggested "the use of the PRV signal as an alternative measurement of the HRV signal during non-stationary conditions" (p.1288). However, the high correlation between PPG and ECG seems to decrease in conditions where participants have to exercise which is consistent with the sensitivity issue of PPG sensors (Lin et al., 2014). Only a few studies were unable to establish a statistical correlation (Jo et al., 2016; Parasnis et al., 2015).

Although these studies all examined the validity of PPG sensors, they often come to inconclusive and incomparable inferences, due to the use of different statistical methods (Sartor et al., 2018; Zaki et al., 2012), evaluation on different variable levels (Shcherbina et al., 2017) and the lack of decision criteria to determine the validity of those sensors (van Lier et al., 2019), indicating a lack of standardized validity assessments. Therefore, van Lier et al. (2019) introduced a standardized validity assessment protocol for physiological signals (e.g., electrodermal activity and cardiovascular activity) on three levels: the signal, parameter and event level. The signal level assesses the ability of a device to measure similar signals, compared with a reference device (RD), and whether these differences lie within an acceptable range with regard to their correlations (van Lier et al., 2019). The parameter level focusses on whether the device is able to produce the similar physiological parameters for each individual, compared to the RD, while the event level determines the ability to detect events reflected by physiological changes, due to external stressors.

In particular, van Lier et al. (2019) validated the PPG sensor of the E4 wearable by comparing it with an ECG. In contrast to extracted parameters of the PPG and ECG signal which can be correlated, the signal themselves have quite different characteristics, due to the different measurement techniques. Therefore, the comparison at the signal level was not performed since it would be impossible to determine which differences were introduced by the wearable and which was from using different techniques. In order to implement the validity assessment protocol and evaluate a PPG sensor at all three levels, a validation of such a PPG sensor has to be compared with a PPG sensor which is already validated and established. Subsequently, the PPG-based measurement of CVA at the fingertip is a conventional stress measurement approach in ambulatory settings and, therefore, a practical RD-alternative to the ECG (Nardelli et al., 2020). Due to the fact that the validity assessment protocol was only applied with an ECG as RD, the complete protocol has to be implemented with a PPG as RD, in order to be able to make a reliable statement about the validity of the assessed sensor. Moreover, this would not only demonstrate the ability of the assessment protocol to be used with RDs other than the ECG, but also enable a more efficient and cheaper way of validating sensors, due to the availability and easy usage of PPG sensors. The accessibility of PPG sensors in comparison with the ECG, combined with the standardized validity assessment protocol, could facilitate a more frequent and straightforward approach to validating sensors.

In this article, the validity of the wrist-worn E4 PPG sensor is assessed by applying the comprehensive validity assessment protocol of van Lier et al. (2019) and by comparing the E4 to an optical pulse sensor which measures PPG from the fingertip. Therefore, a thorough investigation at the signal, parameter and event level is performed. By inducing different intensities of stress, in the form of a social and a mental task, the sensors can be observed in various situations, enabling a comprehensive analysis.

Specifically, the social and mental stress tasks were found to be the most effective stressors, whereby the immediate affection of the psychophysiological responses by these stressors is guaranteed (Dickerson & Kemeny, 2004; Linden et al., 1997). On the one hand, a modified version of the Sing-a-Song Stress Test (SSST) is used to induce social stress in an ethical and straightforward approach (Brouwer & Hogervorst, 2014). The magnitude of increased stress levels by the original SSST are comparable to the Trier Social Stress Test (Kirschbaum et al., 1993). However, the SSST can be performed by one researcher in a shorter amount of time than the Trier Social Stress Test, which is an effective, but time-consuming task. Therefore, the SSST is chosen over the Trier Social Stress Test, due to its quick and feasible approach (van Lier et al., 2019). Moreover, Egilmez et al. (2017) investigated several stress-inducing activities, like a Math test and the Stroop Color-Word Test, and showed that singing induced the most stress in young adults. Nonetheless, studies investigating the effects of the SSST on HR indicated that the main increase in stress was observed during the preparation phase, before participants had to actually sing (Kudielka et al., 2004; von Dawans et al., 2011). In particular, Kudielka et al. (2004) demonstrated that the HR response pattern increased significantly during the preparation phase for children, younger adults and older adults. The significant increase in stress during the preparation phase could also be shown in subjective stress measurements (Häusser et al., 2012; von Dawans et al., 2011). Therefore, the SSST is modified, in order to be even more time efficient, and measure a fast increase induced by social stress, as well as a relatively abrupt decrease in form of relief. On the other hand, the Stroop color-word task is chosen as a demanding cognitive stressor (Choi et al., 2010; Egilmez et al., 2017; Manuck et al., 1996; Muldoon et al., 1992; Poh et al., 2010). Although van Lier et al. (2019) used a noise task, they recommended to use a stronger and longer stressor, due to the fact that they could not distinguish the responses measured by both sensors. In particular, the Stroop task is likely to increase the stress slowly, due to its permanent cognitive demand, followed by a slow decline, caused by its repetitiveness (Al'Absi et al., 1997; Kamarck & Lovallo, 2003). It was chosen to do the cognitive task last, due to its potential to provoke anger or frustration in participants which was shown to prolong the return to the baseline levels, whereby even 10 minutes of recovery were not enough to return to baseline levels (Baum & Contrada, 2010; Linden et al., 1997).

Methods

Participants

Twenty healthy participants (age: *M* = 22.9, *SD* = 2.49; gender: 65% female) participated in the experiment. The study protocol was approved by the Ethics Committee of the University of Twente (UT) and written consent was obtained from the participants beforehand. Recruitment was based on convenience sampling, involving approaching people on the campus of the UT and using the university internal SONA platform which provides undergraduate students with a full overview of the study, as well as the opportunity to sign up and choose their preferred timeslot. Thereby, participants had to be able to physically perform the exercises with minimal risk of health complications. In case of reported significant medical conditions, especially diseases which are related to the cardiovascular system, participants were excluded (see Appendix A). The exclusion criteria also included participants dealing with addictions or the consumption of drugs, medication or substances which have a direct or indirect impact on the CVS including alcohol (Quintana & Heathers, 2014). Although smoking cigarettes and drinking coffee was not an exclusion criterion, participants were asked beforehand to avoid consuming a few hours before the experiment. These criteria enabled a high reliability by minimizing factors which might have disrupted or distorted the recordings.

Materials

The experiment was programmed with Python 2.7 and PsychoPy 3 (Peirce et al., 2019) (see Appendix B). The interaction between participant and program was restricted to key input. This applied to the questionnaires, the tasks, as well as the instructions which were automatically presented on a 15" Windows 7 laptop. In addition, the program saved timestamps for every interaction, as well as starting points of important parts to separate files.

Empatica E4

The Empatica E4 (Empatica, Milan, Italy) is a wearable device designed for continuous, real-time data acquisition in daily life. Beside the PPG sensor, the device is able to measure several different psychophysiological responses of the body, like electrodermal activity, temperature and acceleration. The PPG sensor measures BVP with a sample rate of 64 Hz which is used in order to compute the HR and PPI. Furthermore, the sensor is equipped with an internal real-time clock, enabling recordings for over 36 hours with a capacity of over 60 hours of data storage. Specifically, the data is stored on 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. With regard to the assessment of cardiac activity, a validation study of the E4 demonstrated sufficiently precise PPG signals (McCarthy et al., 2016). Moreover, Ragot et al. (2017) concluded that the wearable devices, and especially the E4, "appear as accurate as laboratory sensors for emotion recognition" (p.20).

Shimmer3 GSR+

The Shimmer3 GSR+ unit (Shimmer Sensing, Dublin, Ireland) is a wireless biosensor designed for simultaneous and real-time measurements. In particular, the Shimmer3 is able to capture electrodermal activity and acceleration, as well as continuous heart rate via an optical pulse sensor. Therefore, the optical pulse sensor has to be attached to the GSR+ module via the 3.5 mm headphone port, providing a PPG signal from the finger with a sample rate of 128 Hz. The recorded data is stored locally on microSD and has to be imported with the Consensys Software which also allows for the transformation of PPG data into HR. In order to transfer the recordings to a computer, the sensor has to be connected with the docking station, which is also used for charging, as well as calibrating the sensor.

Short State-Trait Anxiety Inventory

The short version of the State-Trait Anxiety Inventory (STAI; Spielberger et al., 1983) consists of six items on a four-point Likert scale (see Table 1). Specifically, this questionnaire was used to determine the perceived stress of the participants before and after each task, in order to investigate the relationship between the subjective levels of stress and the recordings of the sensors. The original STAI is used to measure „the presence and severity of current symptoms of anxiety and a generalized propensity to be anxious“ (Julian, 2011, p.467), whereby one of the leading interests in the research domain is the investigation of its possible ability to indicate the presence of mental disorders (Groth-Marnat, 2003; Julian, 2011; Kvaal et al., 2005). Consequently, the short STAI consists of three anxiety-absent items and three anxiety-present items which were identified by Spielberger, one of the main developers of the original STAI, to be particularly sensitive to low and high stressors, respectively (Marteau & Bekker, 1992; Spielberger et al., 1983). According to Marteau and Bekker (1992) who developed and validated the short version of the STAI, this shortened questionnaire has a good reliability (reliability coefficient of 0.82) and validity which was replicated by Tluczek et al. (2009).

Table 1

Self-evaluation questionnaire (short STAI: Y-6 item) (Marteau & Bekker, 1992)

Items	Not at all	Somewhat	Moderately	Very much
1. I feel calm …	1	2	3	4
2. I am tense …	1	2	3	4
3. I feel upset …	1	2	3	4
4. I am relaxed …	1	2	3	4
5. I feel content …	1	2	3	4
6. I am worried …	1	2	3	4

Modified Sing-a-Song Stress Test

A modified version of the Sing-a-Song Stress Test (SSST) was used to induce social stress in an ethical and straightforward approach (Brouwer & Hogervorst, 2014). Due to studies demonstrating a significant increase in stress during the preparation phase, the modification included that participants only were shown the message to prepare to sing a song, but not actually had to sing a song (Kudielka et al., 2004; von Dawans et al., 2011). This was implemented, in order to measure a steep decline in stress through the relief of not having to sing a song which would be more gradually if participants should have to sing a song. This way, a steep increase, as well as steep decrease in stress could be measured. Participants were asked not to disclose the fact that they do not actually have to perform the song to other participants, in order to prevent someone from knowing this beforehand which would make the social stress induction ineffective.

During the task, participants got the instruction to think of a song which they later had to perform for 30 seconds. They had two minutes to think of a song which could have been any song they liked or knew. After the anticipation period, a message was displayed which requested the participants to prepare for singing ("Are you ready?") and press space to continue. Instead of the instruction to start singing, participants were informed that they do not have to sing, in order to be able to measure the relief in the form of a fast decrease of cardiovascular activity.

Stroop Color-Word Test

The Stroop Color-Word Interference Test is a demanding task designed to induce psychological stress and physiological arousal (Choi et al., 2010; Egilmez et al., 2017; Manuck et al., 1996; Muldoon et al., 1992; Poh et al., 2010). For each trial, one of three color words (red, green or blue) was displayed in one of those colors (red, green or blue), whereby both, word and color, were randomly selected. At the bottom of the screen, the three words were presented in their matched color and with their related key on the keyboard (Red: B, Green: N, Blue: M). Participants had to press the key that matched the color of the presented word. Furthermore, instructions were provided with a short test run which displayed feedback in the form of a green window for the correct key and a red window for pressing the wrong key. In total, participants had to execute 500 trials which took approximately eight minutes. The responses of those trials were neither registered, nor analyzed.

Design

For the present study, a within-subjects design was implemented, whereby all participants were equipped with both, the Shimmer3 and the Empatica E4, sensors. Thereby, all participants had to perform the modified SSST and the Stroop task, in order to induce different levels of stress (Dickerson & Kemeny, 2004; Linden et al., 1997).

Procedure

In the beginning of each session, participants were brought to a quiet room and sat in front of the laptop to acclimate while they were briefed by the Experimenter regarding the experiment. They were informed on how their data would be used, that participation was voluntary and that they could stop at any point during the experiment. It was emphasized that participants should move as little as possible, especially with the arm where both sensors would be attached to. Also, they were informed that the experiment would take approximately 40 minutes and that instructions were displayed on the screen. Afterwards, they were asked to read and sign the informed consent.

Both sensors were then attached to the non-dominant arm of the participant, so that they still could press keys with their dominant hand without producing a wide range of artifacts. In particular, the E4 was placed on the wrist, while the Shimmer3 was clipped at the index finger. Next, the program with the experiment was started and the participants demographics were noted. There was a baseline (BS1) and recovery phase (RC1) of 5 minutes before and after the modified SSST (ST1). With regard to the Stroop task, there was also a second baseline (BS2) before the Stroop task (ST2), followed by a recovery phase (RC2). Participants were instructed to sit quietly, move as little as possible and focus on their breathing. In addition, participants filled out the short STAI questionnaire before and after each task, as well as after both recovery periods. In order to increase the stress during the modified SSST, the researcher took a seat next to the participant. For the Stroop Word-Color Test, as well as the rest of the experiment, the researcher was in the room next to the participant. After completing the experiment, the researcher removed both sensors and thanked participants for their participation.

Data Analysis

STAI Questionnaire Analysis

STAI scores were calculated for each of the six-time measurements in 20 participants, as acute stress reference measurement. The statistical significance of the STAI scores between pre- and post-SSST, as well as pre- and post-Stroop task was determined by using a paired sample t-test. The STAI scores for each measurement point were normally distributed, as assessed by the Shapiro-Wilk Test ($W_{T1\ before\ SSST} = .929$, $p_{T1\ before\ SSST} = .150$; $W_{T2\ after\ SSST} = .952$, $p_{T2\ after\ SSST} = .394$; $W_{T3\ after\ Recovery} = .906$, $p_{T3\ after\ Recovery} = .054$; $W_{T4\ before\ Stroop} = .919$, $p_{T4\ before\ Stroop} = .095$; $W_{T5\ after\ Stroop} = .943$, $p_{T5\ after\ Stroop} = .269$; $W_{T6\ after\ Recovery} = .936$, $p_{T6\ after\ Recovery} = .205$). The graph was created with the statistical program R (R Core Team, 2020) and the associated script can be found in appendix C.

Data Quality Assessment

The recordings were visualized for each participant and phase after the completion of the experiment with the statistical program R (R Core Team, 2020), in order to get an overview with regard to the quality of the data (see Appendix D). Due to artifacts and the fluctuating stability of the PPG signal introduced most likely by motions and the sensitivity of the sensors itself, there were parts of participants' data which had to be removed because they contained too many artifacts to be interpretable (Elgendi, 2012). However, the quality had to be assessed carefully, in order to only remove parts with a low signal-to-noise ratio, while taking into account the limited amount of recorded data. Therefore, a signal quality measure in form of a signal quality index (SQI; Karlen et al., 2012; Orphanidou et al., 2014) was applied throughout the whole PPG signals and for each of the important phases (baselines, recovery periods, stress tests), ranging from 0 to 100 (van Lier et al., 2019). For the baselines, as well as the two main tasks, a signal quality of 70 was needed in order to be included in the further analyses. If a certain recording of one of the phases had less than 50% of usable data after the removal of artifacts, it was also excluded because there was not enough reliable data to make a reliable interpretation. In addition to the signal quality measure, the PPG signals should be checked for powerline interference artifacts which could be due „to the instrumentation amplifiers, the recording system picking up ambient electromagnetic signals and other artifacts" (Elgendi, 2012, p.17), high frequency artifacts, arrhythmia, motion as well as muscle artifacts and low amplitudes. In advance of the application of the signal quality index, the data of three participants were removed, due to unrealistic values measured by the E4. In particular, for each of the three participants the PPG signal dropped around zero for the majority of the recordings which made it not usable for the further analysis. The signal quality of the remaining 17 participants was on average 82.7 (*SD* = 10.9). This approach led to the removal of 37% of the data, whereby the complete datasets of further three participants were removed. Thereby, the majority of signal quality indices below 70 were found in measurements of the E4. In total, the data of 14 participants were used for further analysis.

Signal Comparison (Cross-correlation Function)

In order to perform a cross-correlation function on both PPG signals, the recordings of the Shimmer3 had to be sampled down from 128 Hz to 62.5 Hz. Therefore, a python script was written for the downsampling, as well as the computation of the cross-correlation coefficients across different time lags (see Appendix E). For each of the important phases, cross-correlation was performed, whereby correlations were considered very strong (>0.9), strong (0.7-0.9), moderate (0.5-0.7) or weak (<0.5), depending on the coefficient. However, only correlations of 0.8 or higher are considered acceptable (van Lier et al., 2019).

Parameter Comparison (Bland Altman Plot)

After downsampling, both PPG signals were filtered with low-pass and high-pass filters between 5 and 15 Hz. The PPIs were produced by calculating the duration between successive peak locations, whereby only PPIs with a length between 0.33 and 1.5 were included (van Lier et al., 2019). Next, the signal quality index (Orphanidou et al., 2014) was assigned to the PPIs, only when at least 50% of the data had an SQI of 80%. In order to compare both sensors on a parameter level, three commonly retrieved parameters of the time domain were calculated, namely the mean PPI which was converted into instantaneous HR, the SD PPI (SDNN) and the root mean square of the successive differences (RMSSD). Furthermore, the data was checked for normality and missing data. Lastly, the Bland-Altmann plots were created with the "BlandAltmanLeh" package (Lehnert, 2015) for the statistical program R (R Core Team, 2020) (see Appendix F). Thereby, limits of agreement for each parameter were not to exceed 10% of the plausible values (CI) for the parameters, in order to suggest that the parameter was validly measured. Additionally, the a priori defined boundaries for HR were ± 5 bpm, the SDNN ± 0.06 and RMSSD ± 0.07, respectively (van Lier et al., 2019).

Event Comparison (Error Bar Plot)

After downsampling the Shimmer3 data and filtering the recorded PPG signals, PPIs and HR were calculated. Furthermore, the data was checked for normality and missing data, before creating the event difference plots with R (R Core Team, 2020) (see Appendix G). In particular, a line plot with the mean and the standard error (SE) per task was created, whereby each individual is represented by a line, as well as a line plot with the differences between both sensors for each person with the mean and the SE per task. Thereby, it was important that within the same PPG sensor, the error bars should not overlap between baseline and task. In addition, the a priori defined boundaries which are the reference effects (difference between the baseline and stress task of the S3) of the HR were calculated and plotted. Specifically, the reference effects for the modified SSST and the Stroop task were ± 6.809 and ± 13.33, respectively.

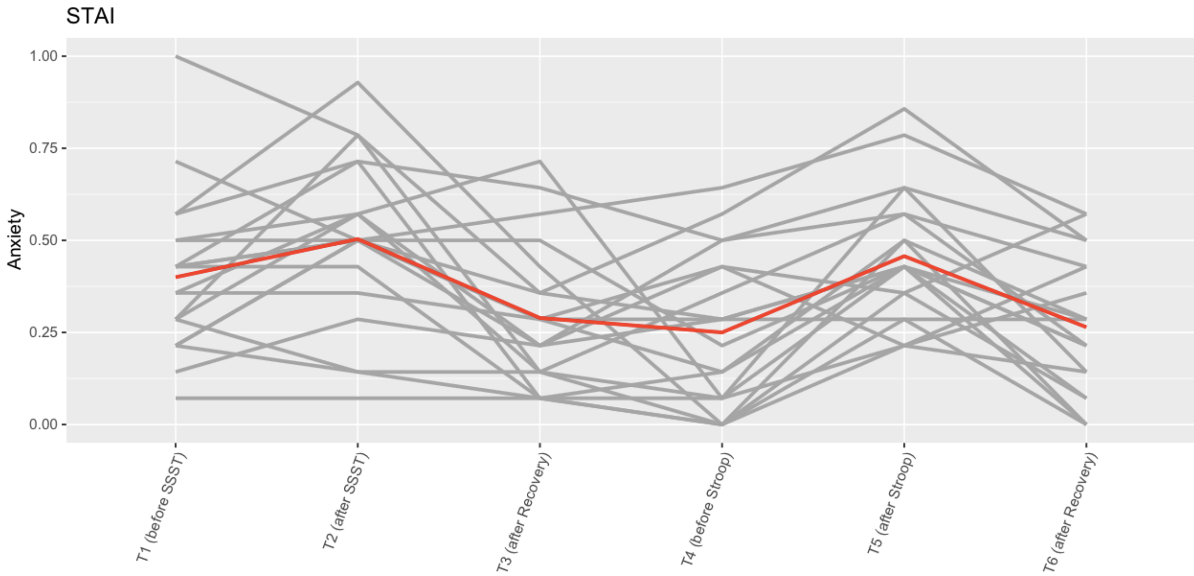
Results

STAI Questionnaire Analysis

A detailed overview of the total STAI scores per phase for each participant can be found in the appendix H. The paired sample t-test found a significant difference of the subjective stress measurement before and after the modified SSST ($t(19) = -2.398$, $p = .027$). The inspection of the means showed an increase in stress after the task ($M = 43.50$, $SD = 10.51$), compared to the measured stress before the modified SSST ($M = 38.66$, $SD = 9.94$). With regard to the Stroop task, there was also a significant difference present in the subjective stress measurement before and after the task ($t(19) = -5.040$, $p<.001$). In particular, an increase in stress was visible after the Stroop task ($M = 41.33$, $SD = 8.47$) compared to before the task ($M = 31.66$, $SD = 9.76$).

Figure 2

Normalized subjective measurements (total STAI scores)



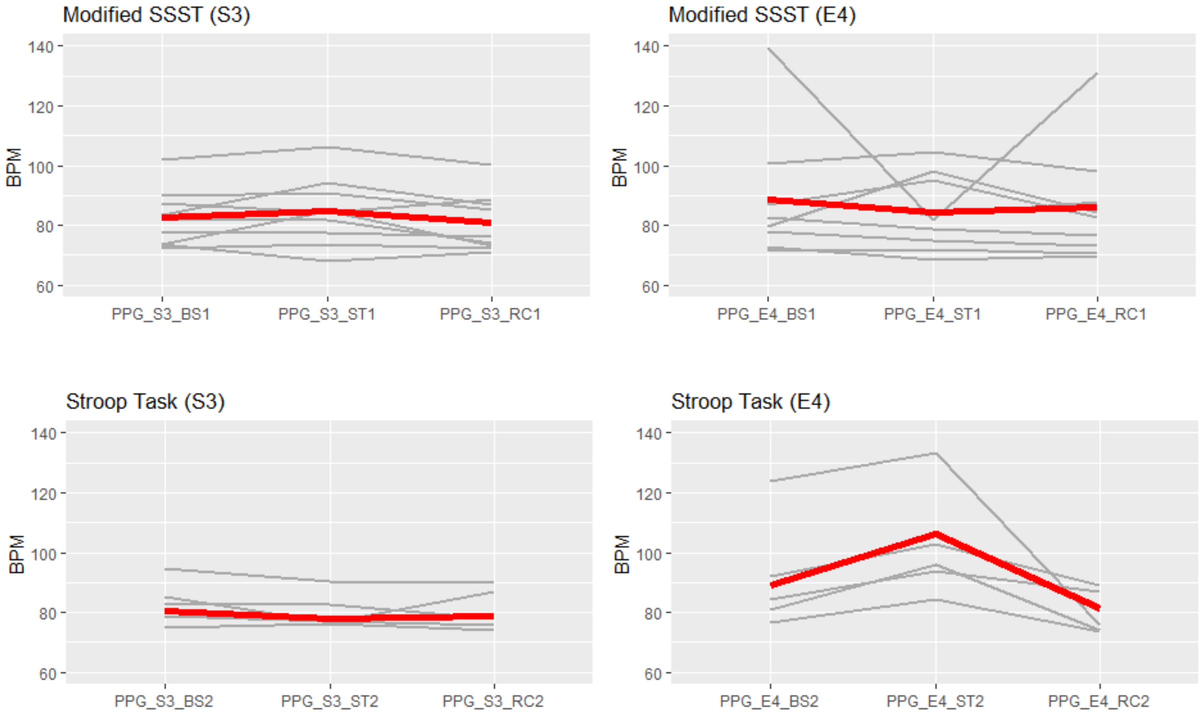
Note. The normalized subjective measurements (total STAI scores) are displayed for the relevant phases for each participant (grey lines) with the average (red line).

These results are clearly visible by the visualization of the subjective stress (see Figure 2). Thereby, a clear increase in the subjective stress can be seen in the measurement after the modified SSST, followed by a decrease during recovery and even lower stress levels after the second baseline. With regard to the Stroop task, the induction of stress had a more significant effect which is represented by a steeper increase in the average of subjective stress levels.

However, the inspection of the physiological data showed that the increase in stress could not be demonstrated by both sensors. In particular, the E4 was not able to measure the increase for the modified SSST, in fact the average showed a slight decrease in HR for ST1 (see Figure 3). In contrast, the S3 was able to measure a slight increase in HR for ST1. With regard to the Stroop task, the E4 measured on average an increase from more than 15 bpm for ST2 (see Figure 3). However, the S3 was not able to detect an increase and actually displayed a slight decrease in HR for ST2.

Figure 3

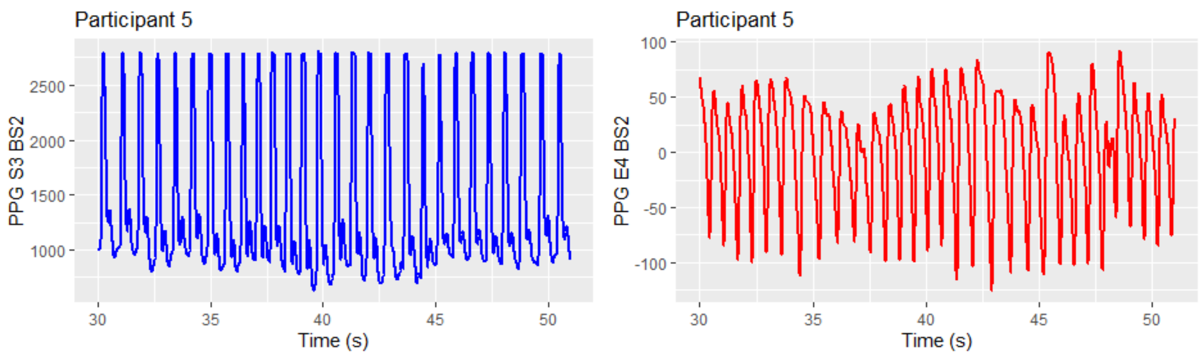
HR of the E4 and S3: modified SSST & Stroop task



Note. The measured HR of the E4 and S3 is displayed for each participant (grey lines), as well as the average (red line), with regard to the relevant phases of the modified SSST and Stroop task, respectively.

Figure 4

PPG signals of S3 and E4 of participant 5 during the second baseline



Signal Comparison

For all participants, low correlations for the measurements of the S3 and E4 were found, ranging from -0.086 to 0.097 with an average around zero (M = 0.004). The correlation coefficients of each participant for the PPG data of the E4 and the S3 can be seen in table 2.

The weak correlations can be illustrated in an example of the recorded PPG data curves of both sensors (see Figure 4). In the figure, the difference between both sensors can be seen clearly with the S3 displaying a much higher amplitude of the systolic peak, in comparison to the measured amplitude of the E4.

Table 2

Correlation Coefficients of E4 and S3 PPG data per Participant and phase

Participant	BS1	ST1	RC1	BS2	ST2	RC2
P1	-0.015	0.052	0.027	-0.006		-0.004
P2	0.042	0.025	-0.004	0.010	0.015	-0.023
P4	0.043	0.097	-0.003	0.070		-0.051
P5	0.021	0.041	-0.035	-0.037	-0.002	-0.038
P6	-0.021	-0.086	0.044	-0.042	0.024	
P7	0.035	-0.010	0.024	-0.030	-0.016	-0.044
P8		0.007				
P9	-0.042	-0.013	0.059	-0.021	0.066	
P12	-0.013	0.010		0.044		
P13				0.006		-0.058
P14	0.019		0.014	0.005	-0.023	
P15	0.063	0.091	-0.004	-0.033	-0.022	-0.022
P17	0.049		-0.005			0.046
P18	-0.030	-0.012	0.018	0.000	-0.045	0.022

Note. The correlation coefficients for E4 and S3 are displayed for the relevant phases (BS1 = first baseline; ST1 = modified SSST; RC1 = first recovery phase; BS2 = second baseline; ST2 = Stroop task; RC2 = second recovery).

Parameter Comparison

The Bland-Altman plots of the mean, SD, and the RMSSD of the PPI are displayed in Figure 5. For the HR, the majority of values are within the limits of agreement, with one participant exceeding theses limits, while still being within the 95% confidence level of the limits of agreement. More than half of the participants are also within the proposed boundaries of ± 5 bpm. Thereby, all values outside the boundaries are greater than zero, showing that for higher averages of HR for both sensors, the E4 measures much greater values for HR, resulting in a greater difference between the measured HR of E4 and S3. With regard to the SDNN and RMSSD of the PPIs, no values lie within the ± 0.07 boundary and ± 0.06 boundary, respectively. However, all the values of the RMSSD are within the limits of agreement, with lower means being more frequent in the 95% confidence interval of the mean difference. With regard to the SDNN Bland-Altman plot, all participants, except for one, lie within the limits of agreement, whereby the outlying participant is still within the 95% confidence interval of the limits of agreement.

Event Comparison

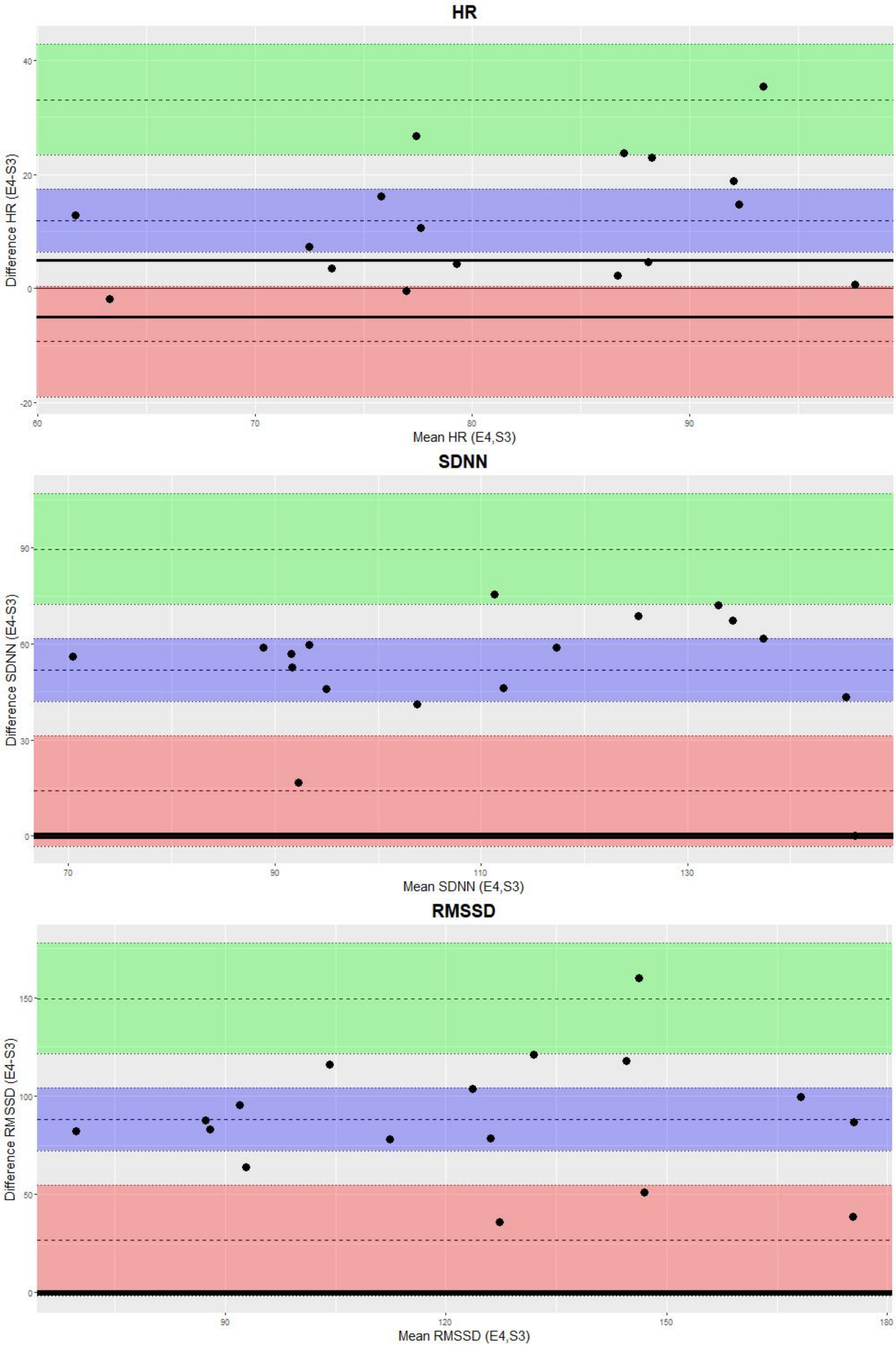
For the creation of the error bar plots, only a few participants could be used, due to the requirement of having high quality data for three consecutive phases (e.g., BS1, ST1, RC1). In order to include as many participants as possible for the event comparison, the experiment was split up into two parts, consisting of a baseline, either the modified SSST or Stroop task and the related recovery phase. In particular, nine participants were eligible for the first three phases, whereas only five participants were included for the last three phases (see Figure 6).

With the exception of one participant who showed differences in the HR of around 60 bpm for the baseline, as well as recovery phase of the modified SSST, all participants' data lie within the predefined boundaries. However, no clear pattern could be established. For some participants, the modified SSST resulted in a decrease of HR, while for others it increased. The average shows a slight decrease in HR for ST1. With regard to the error bars, only a few participants lie within the error bars which is mainly due to the outlier described previously, resulting in greater error bars for BS1 and RC1, in comparison to ST1.

A clear pattern was found for the Stroop task. Specifically, the HR increased after the BS2 and during ST2 and decreased afterwards (RC2). This trend was even present in the outlying participant who showed differences in HR between E4 and S3 of around 40 bpm at BS2 and 50 bpm at ST2. Most of the participants, as well as the error bar lie within the predefined boundaries at BS2. This is mainly due to the outlier who showed enormous differences in HR among the two sensors. The error bar for ST2 lies outside the defined boundaries, with two participants within the error bar. In contrast, all participants, as well as error bar, are within the boundaries at RC2, displaying differences of less than four bpm.

Figure 5

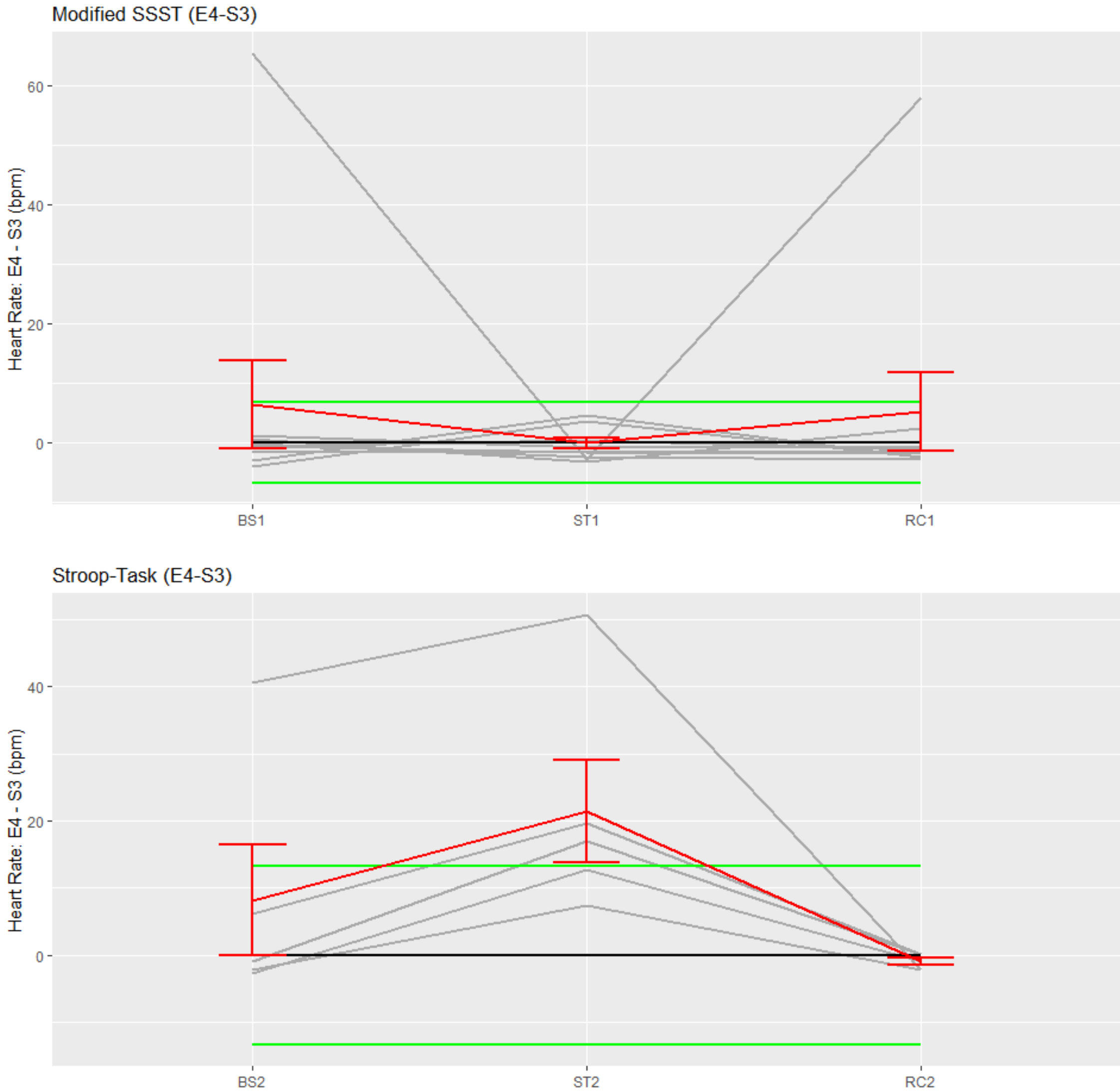
Bland-Altman plots for HR, SDNN and RMSSD



Note. The average of the E4 and S3 is displayed on the horizontal axis and the difference between the two sensors along the vertical axis for heart rate (HR), standard deviation of inter-beat interval (SDNN) and root mean square of successive differences (RMSSD). Each dot represents a participant. Thereby, the blue dotted line shows the mean difference with the purple colored area representing the 95% confidence interval of the mean difference. The red and green dotted lines display the limits of agreement with the colored area around them representing the 95% confidence interval for the limits of agreement. The two bold black lines are the a priori chosen acceptable boundaries.

Figure 6

Error bar plot for HR of each participant for the modified SSST and Stroop task



Note. The error bar plot displays differences in heart rate (HR) for each participant (grey lines) during the modified SSST and the Stroop task with the associated baseline and recovery phase. The red lines show the means and standard errors (SEs), while the black line is the zero axis. The green lines represent the a priori defined boundaries (size of the reference effect).

Discussion

By applying the validity assessment protocol developed by van Lier et al. (2019), the validity of the wrist-worn E4 PPG sensor was assessed by comparing it to the fingertip-worn Shimmer3 PPG sensor. Thereby, the PPG signal was investigated at the signal, parameter and event level by inducing different intensities of stress, induced by the modified SSST and the Stroop task.

The analysis of the subjective stress measurement found a significant difference before and after both stress tasks. In particular, there was an increase in stress for both tasks. Through the analysis at the signal level, no correlations could be found in the PPG measurements of the E4 and S3, implying a low signal level validity. At the parameter level, all participants were within the 95% confidence interval of the limits of agreement. More than half of the measurements were within the predefined boundaries for HR. However, for SDNN and RMSSD no values were within the boundaries. The error bar plot showed a clear pattern for the Stroop task, but not for the modified SSST. With the exception of one participant, all others were in the predefined boundaries with regard to the modified SSST. With regard to the Stroop task, the clear pattern showed an increase in HR differences at ST2, while most participants were within the boundaries.

Subjective Stress Measurement

The significant difference found by the paired sample t-test and the visualization of the average scores show a successful stress induction for the modified SSST, as well as the Stroop task. This indicates that although the increase in stress is not as steep as demonstrated by numerous other studies, both stress tasks led to a significant increase in subjective stress (Dickerson & Kemeny, 2004; Häusser et al., 2012; Kudielka et al., 2004; Linden et al., 1997; von Dawans et al., 2011). With regard to the Stroop task, the results are in line with the literature which found that the Stroop task was effective in inducing psychological stress (Choi et al., 2010; Egilmez et al., 2017; Manuck et al., 1996; Muldoon et al., 1992; Poh et al., 2010). Although the modified SSST showed a significant increase in subjective stress, it was slightly lower than one would expect based on previous research which identified the social stressor to be the most effective (Dickerson & Kemeny, 2004; Egilmez et al., 2017; Linden et al., 1997). One possible explanation for the lower increase of subjective stress by the modified SSST could be the modification itself. Thereby, participants only had to prepare to sing a song, without actually singing out loud, based on previous studies which demonstrated a significant increase of HR, as well as subjective stress measurements during the preparation phase alone (Häusser et al., 2012; van Dick & Mojzisch, 2012; von Drawans et al., 2011). However, the results of the present study indicate that this modified version of the SSST is not strong enough as a stressor to induce the anticipated amount of subjective stress, as only the singing part was left out for the modified SSST.

Signal Level

In this study, the E4 showed a low signal validity, indicating that there was no linear relationship between the measurements of the E4 and the S3 and, consequently, disabling an extensive comparison at the signal level. Although the criterion for validity at the signal level was .80, only cross-correlation coefficients around zero could be found, which for the most part do not reflect the findings of previous research. The majority of studies report good correlations between PPG signals recorded from finger and wrist, whereby the shape of the PPG signal from the finger was described as sharper and more reliable than the wrist (Keikhosravi & Zahedi, 2012; Nardelli et al., 2020; Tăuţan et al., 2015). Besides, Menghini et al. (2019) conducted a study validating the E4 and found similar weak correlations for HR, SDNN and RMSSD during the social stressor and the Stroop task. However, they achieved high correlations for seated and resting phases which are similar to the baseline and recovery phases of the present experiment.

In order to investigate whether the absence of correlations between the E4 and the S3 could be attributed to the data analysis, various methods and approaches were used. Specifically, different downsampling methods were used, namely interpolation, numpy's interpolation, FFT for the Fourier method and poly, provided by the *NeuroKit2* toolbox for python (Makowski et al., 2021). Those different downsampling methods were also tried with an upsampling approach, whereby the E4 was upsampled instead of downsampling the S3. Furthermore, it was controlled for different time lags. However, the individual downsampling methods, the upsampling approach, as well as the time lags yielded lower correlations or around the same correlations, but not higher correlations across all of the relevant phases. Finally, the timestamps which had to be computed for the E4 based on the starting timestamp were recalculated but showed no difference between the original and recalculated timestamps. It was also checked whether the timestamps and lengths of the relevant phases were matching between the two sensors. Overall, these investigations ruled out errors in the data analysis and indicated that the error could be attributed to a systematic error within the sensor, most likely the algorithm of the E4. In particular, the algorithm adjusts the PPG signal, in order to remove artifacts and will be discussed as a possible cause for the absence of correlations between the sensors in the following.

The absence of a linear relationship between the PPG signal of the E4 and the S3, can partially be explained by several differences with regard to the measurement approach. The S3 sensor contains a bright green LED and an ambient light sensor which provides a voltage that is converted by the Shimmer ADC to a 12-bit number that represents the PPG signal in micro Volt (Shimmer, 2016). On the other hand, the E4 sensor uses green and red light and combines them with an algorithm, in order to maximize detection of the pulse wave, according to the website of Empatica (Utilizing the PPG/BVP signal, 2020). Therefore, the measured light contains most information on the pulse wave during exposure to the green LED, whereas the measured light during exposure to the red light contains a reference level to cancel out motion artifacts. However, no specific information could be found regarding the artifact removal algorithm itself and how it selects and processes missing or corrupted data (Ollander et al., 2016). This could include limitations with regard to the validity, due to possible interferences and adjustments by the algorithm which are impossible to detect afterwards. Nevertheless, a linear relationship between the E4 and the S3 was expected. The absence of the linear relationship indicates that the results of the E4 should be interpreted cautiously, especially with regard to the validation of the sensor where accurate measurements are essential.

Furthermore, the difference in approach of obtaining the PPG signal implies different measurement units. Despite the different measurement units, a linear relationship was expected, due to the characteristic parameters of a PPG signal which are present in both signals. However, as Figure 4 indicates, the S3 measures more accurately than the E4 sensor. Specifically, the S3 measured in micro Volt which relates to voltage, whereas the E4 measured in nano Watt which relates to power. The S3 measured values around 1.000 to 2.000 micro Volt, in contrast to the E4 which measured values around the zero axis with positive and negative values in nano Watt. This difference of the measurement units is visualized in Figure 4, where especially the difference with regard to the amplitude of the systolic peak is clearly visible. In particular, this difference in amplitude can be explained by the different measurement sites and their effects on the pulse wave characteristics which attributes a more identifiable waveform characteristic to measurements at the finger, in contrast to recordings at the wrist (Hartmann et al., 2019; Rajala et al., 2018). Furthermore, this includes higher amplitudes of PPG signals at peripheral areas, especially signals recorded at the finger, which are due to larger vascular beds (Hartmann et al., 2019; Maeda et al., 2011; Rajala et al., 2018).

Beside the impact of the sensor placement on waveform characteristics, the measurement site influences the quality of the recorded data. Specifically, sensors at the fingertip were found, in prior research, to measure psychophysiological data more accurately, in contrast to sensors at the wrist which were found to have the highest error, although this was controlled for in the present study by minimizing the movements of the participant and by reminding the participant to move their non-dominant hand with the sensors as little as possible (Longmore et al., 2019; van Dooren et al., 2012). This is partly due to the higher probability of motion artifacts when measuring at the wrist because the sensors need a constant contact with the skin, in order to provide high quality recordings. Therefore, motions like twisting the wrist or just moving the arm around have the potential of creating motion artifacts. With regard to the correlations between both sensors, the validity assessment protocol states that “signals are not expected to have high cross correlation due to placement differences and usage of alternative techniques” (van Lier et al., 2019, p.19). Therefore, these findings do not necessarily imply invalidity of the sensor but emphasize the importance of the investigation of the parameter and event levels.

Parameter Level

The results of the parameter level analysis showed that more than half of the participants were in the predefined boundaries. This is slightly different from studies of Ollander et al. (2016), as well as of Zheng and Poon (2016) who showed that HR of the E4 could be validly determined within the boundaries. Although all values were within the 95% confidence interval of the limits of agreement, the Bland-Alman plot showed that greater averages of HR from E4 and S3 resulted in a greater difference in HR between both sensors. In particular, participants had a difference from up to 20 bpm, with one participant displaying a difference of almost 40 bpm. Therefore, the HR can only be determined validly for averages up to 80 bpm. For greater averages which likely will occur when inducing stress, the HR measurement of the E4 cannot be validly determined. With regard to the SDNN, as well as RMSSD, no values were within the predefined boundaries. This could be attributed to a systematic bias which is caused by the different ways of measuring the PPG signal or the need for a different way of analyses as the signals are in principle quite different, reflected by greater E4 values for HR, SDNN and RMSSD. The differences in the extracted parameters were shown to be large and led to values within the limits of agreement, while being outside the predefined boundaries. Therefore, the HR of the E4 can be determined validly only for lower HRs, whereas the SDNN and RMSSD cannot be determined validly, due to the large difference in the extracted parameters.

Event Level

In contrast to the subjective stress measurements, the physiological measurements could not replicate the increase in measured stress. The analysis at the event level showed no clear pattern for the modified SSST. However, except for one participant, all others were in the predefined boundaries. This outlying participant is the reason for the greater height of the error bars at BS1 and RC1, due to differences in HR of around 60 bpm. With regard to the Stroop task, the error bar plot showed that the values for BS2 and RC2 were within the boundaries, whereas most participants were outside the predefined boundaries for ST2. Taking into account figure 3 which displays the HR per participant for each sensor and task, allows for a more detailed discussion. In particular, the E4 was not able to detect an increase in HR for the modified SSST, whereas for the Stroop task, the S3 did not measure an increase. Therefore, the results at the event level are inconclusive for both, the modified SSST and the Stroop task. Furthermore, it can be concluded that both tasks were not strong enough stressors to elevate the HR and to see differences in the physiological arousal.

This is in contrast to the literature which found social and cognitive stressors to be the most effective in inducing stress at the HR level (Choi et al., 2010; Egilmez et al., 2017; Manuck et al., 1996; Muldoon et al., 1992; Poh et al., 2010). With regard to the modified SSST, studies demonstrated that the HR response pattern significantly increased during the preparation phase (Kudielka et al., 2004; von Dawans et al., 2011). However, the responses measured by the E4 could not be distinguished from each other. This could be due to the short stressor and the absence of the singing. Several studies found the wrist to be only sensitive to larger stressors (Ollander et al., 2016; van Dooren et al., 2012). With regard to the Stroop task, the responses measured by the S3 could not be distinguished from each other. A possible explanation could be that participants moved too much during the task, although the instructions were to only use the dominant hand and hold the non-dominant hand with the sensors as still as possible, as the S3 was found to be extremely sensitive toward physical exercise (Longmore et al., 2019). Another possible explanation for the inconclusive results could be that the difficult level of the Stroop task did not change. This way, the induced stress decreased rather quickly, due to a habituation effect. By implementing an adaptation algorithm which decreases response times, the Stroop task could be made gradually more difficult even for participants who performed well during the first phases of the task, as well as create a steady stress inducement which then can be evaluated by examining time on task effects.

With regard to the subjective measurements which could not be replicated by the physiological measurements, the findings are supported by a study of Campbell and Ehlert (2012) who investigated the relationship between the physiological responses induced by a social stressor such as the SSST and subjective experience and found non-existent to moderate correlations. However, this only would explain the results of the modified SSST and not of the Stroop task, indicating that both tasks are not strong enough stressors.

Beside the fact that the stressors were not strong enough, there was too much data loss of the E4. Specifically, three participants were removed completely, due to unrealistic values measured by the E4. After applying the SQI, 37% of the data was removed which led to the removal of another three participants, resulting in only 14 participants for the analysis. The removal included each individual phase which did not have an SQI of at least 70. For the event level analysis, participants were required to have high quality data for three consecutive phases (e.g., BS1, ST1, RC1), whereby such a phase had to be excluded if an individual phase was removed due to insufficient quality. In case BS1 of a participant was removed due to its SQI of 40, all three consecutive phases had to be removed for both sensors which resulted in only nine participants for the first three phases and five participants for the last three phases. This led to the exclusion of S3 data which were otherwise applicable for the analysis of the modified SSST, as well as the Stroop task. With regard to the data loss, it is possible that the algorithm of the E4 which removes artifacts could have played a role, due to the fact that it is not possible to detect where the algorithm adjusted measurements or even how it works (Ollander et al., 2016). Nevertheless, the data loss of the E4 is supported by Longmore et al. (2019) who found that the E4 had the highest error overall, compared to other sensors. It can thus be concluded that the E4 sensor does not measure stress as accurately compared to the S3. Furthermore, the amount of excluded data of the E4 suggests that the algorithm which removes artifacts or the E4 itself is not as sensitive towards weak stressors as the S3 and therefore should not be used in validation studies or studies which do not use strong stressors.

Limitations

One limitation of the present study was the exclusion of data that did not meet the prescribed quality criteria. This limits the generalization of the findings with regard to the validity, especially with the goal being the validation of a biosensor. Almost 40% of the PPG data had to be excluded, whereby the sample size amplified the problem, resulting in only a few participants who could actually be included in the analysis. Although the S3 measured more accurately, many parts of the recorded S3 data had to be excluded, due to the low quality of the E4. Therefore, future research should investigate the influence of the E4's algorithm on the artifact removal and quality of the recorded PPG signal. Furthermore, it is suggested to examine different analysis methods for the signal level evaluation of the E4 and S3, due to the different waveforms and measurement units of the recordings. Based on the demonstrated low validity and inconclusive results, it is advised to deliberately consider the context in which to use the E4. Specifically, the E4 should only be used in ambulatory assessments for the investigation of strong and long-lasting stressors. A validated alternative should be used if the aim is to measure subtle variations in HR parameters or to make meaningful decisions regarding a person's health (e.g., medical- / treatment-decisions). For those reasons, it is suggested to apply the SQI beforehand in a pilot trial, in order to ensure that a sensor produces data with a certain quality. In case of the E4, the application of the SQI would have showed that a large part of the data did not meet the required quality standards for a validation study.

Another limitation was the physiological stress induction which was not successful and also contributed to the inconclusive results with regard to the event level, resulting in indistinguishable signals of the E4. With regard to the SSST, a shorter version of the SSST was used, in order to be more time efficient. However, as the present study showed, the stressor was not strong enough to induce enough stress in the participants. This issue can be addressed by using longer and more intensive social stressors (e.g., original SSST, Trier Social Stress Test), consequently, increasing the overall length of the experiment. With regard to the Stroop task, the level of difficulty should be increased over time by adapting the presentation time, as well as by introducing more or other words and colors, respectively. Further studies are needed, in order to investigate whether the reason for not detecting the physiological arousal is due to the sensitivity of the stressor or the sensor itself.

Conclusion

The application of the validity assessment protocol showed that the validity of the E4 is lower than described in the literature (McCarthy et al., 2016). Specifically, the successful induction of subjective stress for both the modified SSST and the Stroop task could not be replicated by the physiological measurements. Although the present study was unable to make a definitive statement about the validity of the E4, the approach of taking a PPG sensor as RD, as well as the validity assessment protocol itself showed its potential. Instead of an ECG, the PPG sensors are more available, less time-consuming and less restricting with regard to participants and setting. However, it is important for future research to find applicable PPG sensors as reference devices, with regard to artifact sensitivity, the sensor placement and the technical measurement approach of the sensor. Furthermore, stronger and more long-lasting stressors should be used, in order to detect significant increases in HR arousal and possible patterns. This way, it can be examined if the inability to accurately measure the stress induction is due to the stressor or the sensor itself. By implementing those recommendations, data loss to the extent of the present study should be prevented, enabling the extraction and analysis of parameters, as well as the determination of validity.

References

Abbod, M., Chiou, Y. R., Yang, S. H., Fan, S. Z., & Shieh, J. S. (2011). Developing a monitoring psychological stress index system via photoplethysmography. *Artificial Life and Robotics, 16*(3), 430-433, https://doi.org/10.1007/s10015-011-0976-y

Akhter, N., Tharewal, S., Kale, V., Bhalerao, A., & Kale, K. V. (2016). Heart-based biometrics and possible use of heart rate variability in biometric recognition systems. In R. Chaki, A. Cortesi, K. Saeed, & N. Chaki (eds), *Advanced Computing and Systems for Security. Advances in Intelligent Systems and Computing*, Vol. 395 (pp. 15-29). India: Springer.

Al'Absi, M., Bongard, S., Buchanan, T., Pincomb, G. A., Licinio, J., & Lovallo, W. R. (1997). Cardiovascular and neuroendocrine adjustment to public speaking and mental arithmetic stressors. *Psychophysiology*, *34*(3), 266-275.

Allen, J. (2007). Photoplethysmography and its application in clinical physiological measurement. *Physiological Measurement, 28*(3), 1-39, https://doi.org/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, 14*(5), 1139-1148, https://doi.org/10.1016/0735-1097(89)90408-7

Baum, A., & Contrada, R. (Eds.). (2010). *The handbook of stress science: Biology, psychology, and health*. New York: Springer Publishing Company.

Bolanos, M., Nazeran, H., & Haltiwanger, E. (2006, August). *Comparison of heart rate variability signal features derived from electrocardiography and photoplethysmography in healthy individuals*. Paper presented at the 28th IEEE EMBS Annual International Conference, New York City.

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

Brouwer, A.-M., & Hogervorst, M. A. (2014). A new paradigm to induce mental stress: The sing-a-song stress test (SSST). *Frontiers in Neuroscience, 8*, 224-231, https://doi.org/10.3389/fnins.2014.00224

Campbell, J., & Ehlert, U. (2012). Acute psychosocial stress: Does the emotional stress response correspond with physiological responses? *Psychoneuroendocrinology, 37*(8), 1111-1134, https://doi.org/10.1016/j.psyneuen.2011.12.010

Choi, J., Ahmed, B., & Gutierrez-Osuna, R. (2010). *Ambulatory stress monitoring with minimally-invasive wearable sensors* (Technical Report tamu-cs-tr-2010-11-1). Department of Computer Science and Engineering, Texas A&M University.

Chrousos, G. P. (2009). Stress and disorders of the stress system. *Nature Reviews Endocrinology, 5*(7), 374-381, https://doi.org/10.1038/nrendo.2009.106

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

Dickerson, S. S., & Kemeny, M. E. (2004). Acute stressors and cortisol responses: A theoretical integration and synthesis of laboratory research. *Psychological Bulletin*, *130*(3), 355-391, https://doi.org/10.1037/0033-2909.130.3.355

Egilmez, B., Poyraz, E., Zhou, W., Memik, G., Dinda, P., & Alshurafa, N. (2017, March 13-17). *UStress: Understanding college student subjective stress using writs-based passive sensing*. Paper presented at the 2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), 673-678, Kona, USA, https://doi.org/10.1109/PERCOMW.2017.7917644

Elgendi, M. (2012). On the analysis of fingertip photoplethysmogram signals. *Current Cardiology Reviews, 8*, 14-25, https://doi.org/10.2174/157340312801215782

Elgendi, M., Jonkman, M., & DeBoer, F. (2010, January). Heart rate variability and the acceleration plethysmogram signals measured at rest. In A. Fred, J. Filipe & H. Gamboa (eds.), *International Joint Conference on Biomedical Engineering Systems and Technologies* (pp. 266-277). Valencia, Spain: Springer.

Fletcher, R. R., Tam, S., Omojola, O., Redemske, R., Redor, 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, https://doi.org/10.4108/icst.pervasivehealth.2011.246105

Fusco, A., Locatelli, D., Onorati, F., Durelli, G. C., & Santambrogio, M. D. (2015, October). *On how to extract breathing rate from PPG signal using wearable devices*. In Biomedical Circuits and Systems Conference (BioCAS), 2015 IEEE (pp. 1-4). Atlanta, GA: IEEE.

Giardino, N. D., Lehrer, P. M., & Edelberg, R. (2002). Comparison of finger plethysmograph to ECG in the measurement of heart rate variability. *Psychophysiology, 39*, 246-253, https://doi.org/10.1017.S0048577201392168

Gil, E., Orini, M., Bailón, R., Vergara, J. M., Mainardi, L., & Laguna, P. (2010). Photoplethysmography pulse rate variability as a surrogate measurement of heart rate variability during non-stationary conditions. *Physiological Measurement, 31*, 1271-1290, https://doi.org/10.1088/0967-3334/31/9/015

Goldsack, J. C., Coravos, A., Bakker, J. P., Bent, B., Dowling, A. V., Fitzer-Attas, C., Godfrey, A., Godino, J. G., Gujar, N., Izmailova, E., Manta, C., Peterson, B., Vandendriessche, B., Wood, W. A., Wang, K. W., & Dunn, J. (2020). Verification, analytical validation, and clinical validation (V3): The foundation of determining fit-for-purpose for biometric monitoring technologies (BioMeTs). *npj digital Medicine*, *3*(1), 1-15, https://doi.org/10.1038/s41746-020-0260-4

Greve, M., Kviesis-Kipge, E., Rubenis, O., Rubenis, U., Mecnika, V., Grabovskis, A., & Marcinkevics, Z. (2012, April). *Comparison of pulse rate variability derived from digital photoplethysmography over the temporal artery with the heart rate variability derived from a polar heart rate monitor*. Paper presented at the 7th ESGCO, Kazimierz Dolny.

Groth-Marnat, G. (2003). *Handbook of psychological assessment (4th Edition)*. Hoboken, New Jersey: John Wiley & Sons Inc.

Gu, W. B., Poon, C. C. Y., Sy, M. Y., Leung, H. K., Liang, Y. P., & Zhang, Y. T. (2009, June 3-5). *A t-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 International Workshop, Berkeley, https://doi.org/10.1109/BSN.2009.23

Häusser, J. A., Kattenstroth, M., van Dick, R., & Mojzisch, A. (2012). “We” are not stressed: Social identity in groups buffers neuroendocrine stress reactions. *Journal of*

Experimental Social Psychology, 48(4), 973-977, https://doi.org/10.1016/j.jesp.2012.02.020

Hartmann, V., Liu, H., Chen, F., Qiu, Q., Hughes, S., & Zheng, D. (2019). Quantitative comparison of photoplethysmographic waveform characteristics: Effect of measurement site. *Frontiers in Physiology, 10*, 198-205, https://doi.org/10.3389/fphys.2019.00198

Jeyhani, V., Mahdiani, S., Peltokangas, M., & Vehkaoja, A. (2015). *Comparison of HRV parameters derived from photoplethysmography and electrocardiography signals*. Paper presented at the 37th IEEE Engineering in Medicine and Biology Society Conference, Milan, https://doi.org/10.1109/EMBC.2015.7319747

Jo, E., Lewis, K., Directo, D., Kim, M. J., & Dolezal, B. A. (2016). Validation of biofeedback wearables for photoplethysmographic heart rate tracking. *Journal of Sports Science and Medicine, 15*, 540-547.

Julian, L. J. (2011). Measures of anxiety - State-trait anxiety inventory (STAI), beck anxiety inventory (BAI), and hospital anxiety and depression scale-anxiety (HADS-A). *Arthritis Care & Research, 63*(11), 467-472, https://doi.org/10.1002/acr.20561

Kamarck, T. W., & Lovallo, W. R. (2003). Cardiovascular reactivity to psychological challenge: conceptual and measurement considerations. *Psychosomatic Medicine*, *65*(1), 9-21, https://doi.org/10.1097/01.psy.0000030390.34416.3e.

Karlen, W., Kobayashi, K., Ansermino, J. M., & Dumont, G. A. (2012). Photoplethysmogram signal quality estimation using repeated gaussian filters and cross-correlation. *Physiological measurement, 33*(10), 1617-1629, https://doi.org/10.1088/0967-3334/33/10/1617

Keikhosravi, A., & Zahedi, E. (2012, December). Effect of deep breath on the correlation between wrist and finger photoplethysmograms. In *2012 19th Iranian Conference of Biomedical Engineering (ICBME)* (pp. 135-138). IEEE, https://doi.org/10.1109/ICBME.2012.6519672

Kemeny, M. E. (2003). The psychobiology of stress. *Current directions in psychological science, 12*(4), 124-129, https://doi.org/10.1111/1467-8721.01246

Kirschbaum, C., Pirke, K. M., & Hellhammer, D. H. (1993). The 'trier social stress test'–A tool for investigating psychobiological stress responses in a laboratory setting. *Neuropsychobiology*, *28*(1-2), 76-81, https://doi.org/10.1159/000119004

Kudielka, B. M., Buske-Kirschbaum, A., Hellhammer, D. H., & Kirschbaum, C. (2004). Differential heart rate reactivity and recovery after psychosocial stress (TSST) in healthy children, younger adults, and elderly adults: The impact of age and gender. *International Journal of Behavioral Medicine, 11*(2), 116-121, https://doi.org/10.1207/s15327558ijbm1102_8

Kvaal, K., Ulstein, I., Nordhus, I. H., & Engedal, K. (2005). The spielberger state-trait anxiety inventory (STAI): The state scale in detecting mental disorders in geriatric patients. *International Journal of Geriatric Psychiatry, 20*, 629-634, https://doi.org/10.1002/gps.1330

Lehnert, B. (2015). *BlantAltmanLeh*. R package version 0.3.1, URL: https://CRAN.R-project.org/package=BlandAltmanLeh.

Lin, W.-H., Wu, D., Li, C., Zhang, H., & Zhang, Y.-T. (2014). Comparison of heart rate variability from PPG with that from ECG. In Y.-T. Zhang (eds), *The International Conference on Health Informatics. IFMBE Proceedings*, Vol. 42 (pp. 213-215). Cham: Springer, https://doi.org/10.1007/978-3-319-03005-0_54

Longmore, S. K., Lui, G. Y., Naik, G., Breen, P. P., Jalaludin, B., & Gargiulo, G. D. (2019). A comparison of reflective photoplethysmography for detection of heart rate, blood oxygen saturation, and respiration rate at various anatomical locations. *Sensors, 19*(8), 1874.

Maeda, Y., Sekine, M., & Tamura, T. (2011). Relationship between measurement site and motion artifacts in wearable reflected photoplethysmography. *Journal of Medical Yystems, 35*(5), 969-976, https://doi.org/10.1007/s10916-010-9505-0

Makowski, D., Pham, T., Lau, Z. J., Brammer, J. C., Lespinasse, F., Pham, H., Schölzel, C., & Chen, S. A. (2021). NeuroKit2: A python toolbox for neurophysiological signal processing. *Behavior Research Methods, 53*, 1689-1696, https://doi.org/10.3758/s13428-020-01516-y

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

Manuck, S. B., Polefrone, J. M., Terrell, D. F., Muldoon, M. F., Kasprowicz, A. L., Waldstein, S. R., Jennings, R. J., Malkoff, S. B., Marsland, A., & Graham, R. E. (1996). Absence of enhanced sympathoadrenal activity and behaviorally evoked cardiovascular reactivity among offspring of hypertensives. *American Journal of Hypertension, 9*(3), 248-255.

Marteau, T. M., & Bekker, H. (1992). The development of a six-item short-form of the state scale of the spielberger state-trait anxiety inventory (STAI). *British Journal of Clinical Psychology, 31*, 301-306.

McCarthy, C., Pradhan, N., Redpath, C., & Adler, A. (2016, May). Validation of the Empatica E4 wristband. In *Student Conference (ISC), 2016 IEEE EMBS International* (pp. 1-4). Ottawa: IEEE.

Menghini, L., Gianfranchi, E., Cellini, N., Patron, E., Tagliabue, M., & Sarlo, M. (2019). Stressing the accuracy: Wrist-worn wearable sensor validation over different conditions. *Psychophysiology, 56*(11), 13441-13456, https://doi.org/10.1111/psyp.13441

Mitsuhashi, R., Iuchi, K., Goto, T., Matsubara, A., Hirayama, T., Hashizume, H., & Tsumura, N. (2019, July). Video-based stress level measurement using imaging photoplethysmography. In *2019 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)* (pp. 90-95). IEEE, https://doi.org/10.1109/ICMEW.2019.0-105.

Muldoon, M. F., Bachen, E. A., Manuck, S. B., Waldstein, S. R., Bricker, P. L., & Bennett, J. A. (1992). Acute cholesterol responses to mental stress and change in posture. *Archives of Internal Medicine, 152*(4), 775-780.

Mullan, P., Kanzler, C. M., Lorch, B., Schroeder, L., Winkler, L., Laich, L., Riedel, F., Richer, R., Luckner, C., Leutheuser, H., Eskofier, B. M., & Pasluosta, C. (2015, August). Unobtrusive heart rate estimation during physical exercise using photoplethysmographic and acceleration data. In *Engineering in Medicine and Biology Society (EMBC), 2015 37th Annual International Conference of the IEEE* (pp. 6114-6117). Milan, Italy: IEEE.

Munos, B., Baker, P. C., Bot, B. M., Crouthamel, M., de Vries, G., Ferguson, I., Hixon, J. D., Malek, L. A., Mastrototaro, J. J., Misra, V., Ozcan, A., Sacks, L., & Wang, P. (2016). Mobile health: The power of wearables, sensors, and apps to transform clinical trials. *Annals of the New York Academy of Sciences*, *1375*(1), 3-18, https://doi.org/10.1111/nyas.13117

Nardelli, M., Vanello, N., Galperti, G., Greco, A., & Scilingo, E. P. (2020). Assessing the Quality of heart rate variability estimated from wrist and finger PPG: A novel approach based on cross-mapping method. *Sensors, 20*(11), 3156-3168, https://doi.org/10.3390/s20113156

Nath, R. K., Thapliyal, H., & Caban-Holt, A. (2020, January). Validating physiological stress detection model using cortisol as stress bio marker. In *2020 IEEE International Conference on Consumer Electronics (ICCE)* (pp. 1-5). IEEE.

Ollander, S., Godin, C., Campagne, A., & Charbonnier, S. (2016, October). A comparison of wearable and stationary sensors for stress detection. In *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)* (pp. 004362-004366). Budapest, Hungary: IEEE.

Orphanidou, C., Bonnici, T., Charlton, P., Clifton, D., Vallance, D., & Tarassenko, L. (2014). Signal-quality indices for the electrocardiogram and photoplethysmogram: Derivation and applications to wireless monitoring. *IEEE Journal of Biomedical and Health Informatics*, *19*(3), 832-838, https://doi.org/10.1109/JBHI.2014.2338351

Quintana, D. S., & Heathers, J. A. (2014). Considerations in the assessment of heart rate variability in biobehavioral research. *Frontiers in Psychology*, *5*, 78-87, https://doi.org/10.3389/fpsyg.2014.00805.

Parasnis, R., Pawar, A., & Manivannan, M. (2015, November). Multiscale entropy and poincare plot-based analysis of pulse rate variability and heart rate variability of ICU patients. In *International Conference on Intelligent Informatics and Biomedical Sciences (ICIIBMS) 2015* (pp. 290-295). Okinawa, Japan: IEEE.

Peirce, J. W., Gray, J. R., Simpson, S., MacAskill, M. R., Höchenberger, R., Sogo, H., Kastman, E., Lindeløv, J. (2019). PsychoPy2: Experiments in behavior made easy. *Behavior Research Methods, 51*(1), 195-203, https://doi.org/10.3758/s13428-018-01193-y

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

Pietilä, J., Mehrang, S., Tolonen, J., Helander, E., Jimison, H., Pavel, M., & Korhonen, I. (2017). Evaluation of the accuracy and reliability for photoplethysmography based heart rate and beat-to-beat detection during daily activities. In *EMBEC & NBC 2017* (pp. 145-148). Singapore: Springer.

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, 57*(5), 1243-1252, https://doi.org/10.1109/TBME.2009.2038487

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, https://doi.org/10.1109/LSSA.2006.250377

R Core Team (2020). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria. URL: https://www.R-project.org/.

Ragot, M., Martin, N., Em, S., Pallamin, N., & Diverrez, J.-M. (2017). Emotion recognition using physiological signals: Laboratory vs. wearable sensors. In T. Ahram & C. Falcao (eds.), *Advances in human factors in wearable technologies and game design*. AHFE 2017. Advances in Intelligent Systems and Computing*, 608*, 15-22. Los Angeles, California: Springer, https://doi.org/10.1007/978-3-319-60639-2_2

Rajala, S., Lindholm, H., & Taipalus, T. (2018). Comparison of photoplethysmogram measured from wrist and finger and the effect of measurement location on pulse arrival time. *Physiological Measurement, 39*(7), 075010-0750129, https://doi.org/10.1088/1361-6579/aac7ac

Santos, S. A., Venema, B., & Leonhardt, S. (2012). Accelerometer-assisted PPG measurement during physical exercise using the LAVIMO sensor system. *Acta Polytechnica*, *52*(5), 80-85, https://doi.org/10.14311/1646

Sartor, F., Papini, G., Cox, L. G. E., & Cleland, J. (2018). Methodological shortcomings of wrist-worn heart rate monitors validations. *Journal of Medical Internet Research*, *20*(7), 31-36, https://doi.org/10.2196/10108

Selvaraj, N. (2015, August). Psychological acute stress measurement using a wireless adhesive biosensor. In *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)* (pp. 3137-3140). IEEE.

Selvaraj, N., Jaryal, A., Santhosh, J., Deepak, K. K., & Anand, S. (2008). Assessment of heart rate variability derived from finger-tip photoplethysmography as compared to electrocardiography. *Journal of Medical Engineering & Technology, 32*(6), 479-484, https://doi.org/10.1080/03091900701781317

Shcherbina, A., Mattsson, C. M., Waggott, D., Salisbury, H., Christle, J. W., Hastie, T., Wheeler, M. T., & Ashley, E. A. (2017). Accuracy in wrist-worn, sensor-based measurements of heart rate and energy expenditure in a diverse cohort. *Journal of Personalized Medicine, 7*(2), 3-14, https://doi.org/10.3390/jpm7020003

Shimmer (2016). Optical pulse sensor user guide revision 1.6. Retrieved from https://www.shimmersensing.com/images/uploads/docs/Optical_Pulse_Sensor_User_Guide_rev1.6.pdf. Retrieved at 02.06.2021

Singh, R. R., Conjeti, S., & Banerjee, R. (2011, October). An approach for real-time stress-trend detection using physiological signals in wearable computing systems for automotive drivers. In *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)* (pp. 1477-1482). IEEE.

Spielberger, C. D., Gorsuch, R. L., Lushene, R., Vagg, P. R., & Jacobs, G. A. (1983). *Manual for the state-trait anxiety inventory*. Palo Alto, CA: Consulting Psychologists 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, https://doi.org/10.1109/ISWC.2007.4373774

Tăuţan, A. M., Young, A., Wentik, E., & Wieringa, F. (2015, August). Characterization and reduction of motion artifacts in photoplethysmographic signals from a wrist-worn device. In *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)* (pp. 6146-6149). Milan, Italy: IEEE.

Tluczek, A., Henriques, J. B., & Brown, R. L. (2009). Support for the reliability and validity of a six-item state anxiety scale derived from the state-trait anxiety inventory. *Journal of Nursing Measurement*, *17*(1), 19-28.

Utilizing the PPG/BVP signal. (23.01.2020). Retrieved from https://support.empatica.com/hc/en-us/articles/204954639-Utilizing-the-PPG-BVP-signal. Retrieved at 02.06.2021

van Dooren, M., de Vries, J. J. G. G. J., & Janssen, J. H. (2012). Emotional sweating across the body: Comparing 16 different skin conductance measurement locations. *Physiology and Behavior, 106*(2), 298-304, https://doi.org/10.1016/j.physbeh.2012.01.020

van Lier, H. G., Pieterse, M. E., Garde, A., Postel, M. G., de Haan, H. A., Vollenbroek-Hutten, M. M., Schraagen, J. M. & Noordzij, M. L. (2019). A standardized validity assessment protocol for physiological signals from wearable technology: Methodological underpinnings and an application to the E4 biosensor. *Behavior Research Methods, 52*, 607-629, https://doi.org/10.3758/s13428-019-01263-9

von Dawans, B., Kirschbaum, C., & Heinrichs, M. (2011). The trier social stress test for groups (TSST-G): A new research tool for controlled simultaneous social stress exposure in a group format. *Psychoneuroendocrinology, 36*(4), 514-522, https://doi.org/10.1016/j.psyneuen.2010.08.004

Wander, J. D., & Morris, D. (2014). A combined segmenting and non-segmenting approach to signal quality estimation for ambulatory photoplethysmography. *Physiological Measurement*, *35*(12), 2543-2561, https://doi.org/10.1088/0967-3334/35/12/2543

Wilard, 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, 49*(2), 213-220, https://doi.org/10.1007/s11517-010-0722-y

Yoo, K. S., & Lee, W. H. (2011, June). Mental stress assessment based on pulse photoplethysmography. In *2011 IEEE 15th International Symposium on Consumer Eletronics (ISCE)* (pp. 323-326). IEEE.

Zaki, R., Bulgiba, A., Ismail, R., & Ismail, N. A. (2012). Statistical methods used to test for agreement of medical instruments measuring continuous variables in method comparison studies: A systematic review. *PloS one*, *7*(5), e37908, https://doi.org/10.1371/journal.pone.0037908

Zangróniz, R., Martinez-Rodrigo, A., López, M. T., Pastor, J. M., & Fernández-Caballero, A. (2018). Estimation of mental distress from photoplethysmography. *Applied Sciences, 8*(1), 69-83, https://doi.org/10.3390/app8010069

Zhang, Y., Song, S., Vullings, R., Biswas, D., Simões-Capela, N., Van Helleputte, N., van Hoof, C., & Groenendaal, W. (2019). Motion artifact reduction for wrist-worn photoplethysmograph sensors based on different wavelengths. *Sensors*, *19*(3), 673-690, https://doi.org/10.3390/s19030673

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, 61*(5), 1538-1554, https://doi.org/10.1109/TBME. 2014.2309951

Zheng, Y., & Poon, C. C. Y. (2016, May). Wearable devices and their applications in surgical robot control and p-medicine. In *2016 IEEE 20th International Conference on Computer Supported Cooperative Work in Design (CSCWD)* (pp. 659-663). IEEE, https://doi.org/10.1109/CSCWD.2016.7566067

Appendix

Appendix A – Screening Questions

The following questions were asked, in order to examine if potential participants fulfill any of the exclusion criteria.

1. Do you currently suffer from any physical or psychological illness?

2. Do you have any preexisting heart or lung condition?
   For Example:
   a. cardiovascular disease
   b. pulmonary disease
   c. arrhythmias
   d. aorta disease
   e. coronary artery disease
   f. hypertension

3. Have you ever been told by a doctor, nurse, or other health professional that you have high blood pressure?

4. Do you currently take any prescribed medication (especially medication affecting the cardiovascular or nervous system)?

5. Do you have any substance addiction (e.g., alcohol, drugs, medications)?

Appendix B – Experiment Script (Psychopy)

```
#imports
import os
import sys
import csv
import time
import random

import numpy as np

import pygame
from pygame.locals import *
from pygame.compat import unichr_, unicode_

import psychopy.gui
import psychopy.visual
import psychopy.core
import psychopy.event

#set directory
os.chdir('/Users/ene/Documents/Python_Scripts')

#set GUI
gui = psychopy.gui.Dlg()

gui.addField("Subject ID:")

gui.show()

subj_id = gui.data[0]
print("Subject ID: " + subj_id)

#check for existing file
data_path = "p_" + subj_id + "_time_data.csv"

if os.path.exists(data_path):
    sys.exit("Data path " + data_path + " already exists!")

data_path = "p_" + subj_id + "_stai_data.csv"

if os.path.exists(data_path):
    sys.exit("Data path " + data_path + " already exists!")

data_path = "p_" + subj_id + "_stroop_data.csv"

if os.path.exists(data_path):
    sys.exit("Data path " + data_path + " already exists!")

responses = []

#variables
stai_1 = []
stai_2 = []
stai_3 = []
stai_4 = []
stai_5 = []
stai_6 = []

int_stai1 = []
int_stai2 = []
int_stai3 = []
int_stai4 = []
int_stai5 = []
int_stai6 = []
```

```
total_stai = []

timestamps = []

stroop = []

#set window
win = psychopy.visual.Window(size = [1000, 800], units = "pix", fullscr = False, color = [1,
      1, 1])

#INTRODUCTION WINDOW
intro_txt = psychopy.visual.TextStim(win = win, wrapWidth = 350, color = [-1, -1, -1])
intro_txt.text = """
Welcome to the Experiment! \n
Press space to begin.
"""

intro_txt.draw()
win.flip()

curr_time = time.time()
timestamps.append(curr_time)

psychopy.event.waitKeys(keyList=['space'])

#ACCLIMATION PHASE
acclimation_txt = psychopy.visual.TextStim(win = win, wrapWidth = 350, color = [-1, -1, -1])
acclimation_txt.text = """
ACCLIMATION PHASE \n
This phase will take around 5 minutes to acclimate you. \n
All the instructions of the experiment are shown on the screen. \n
Please move as little as possible during the experiment.
"""

acclimation_txt.draw()
win.flip()

curr_time = time.time()
timestamps.append(curr_time)

psychopy.core.wait(300) #300

acclimation_txt.text = """
Press space to continue.
"""

acclimation_txt.draw()
win.flip()

psychopy.event.waitKeys(keyList=['space'])

#BASELINE MEASUREMENT
baseline_txt = psychopy.visual.TextStim(win = win, wrapWidth = 350, color = [-1, -1, -1])
baseline_txt.text = """
BASELINE MEASUREMENT \n
This phase will take about 5 minutes. \n
Please move as little as possible.
"""

baseline_txt.draw()
win.flip()
```

```
curr_time = time.time()
timestamps.append(curr_time)
print(timestamps)

psychopy.core.wait(300) #300

baseline_txt.text = """
Press space to continue.
"""

baseline_txt.draw()
win.flip()

curr_time = time.time()
timestamps.append(curr_time)

psychopy.event.waitKeys(keyList=['space'])

#STAI QUESTIONNAIRE
stai_txt = psychopy.visual.TextStim(win = win, wrapWidth = 350, color = [-1, -1, -1])
stai_txt.text = """
Short State-Trait Anxiety Inventory \n
To select one of the answers, press key 1-4. \n
Press space to continue.
"""

stai_txt.draw()
win.flip()

curr_time = time.time()
timestamps.append(curr_time)

psychopy.event.waitKeys(keyList=['space'])

#item1
stai_item1_txt = psychopy.visual.TextStim(win = win, wrapWidth = 350, color = [-1, -1, -1])
stai_item1_txt.text = """
I feel calm ... \n \n
1. Not at all \n
2. Somwhat \n
3. Moderately \n
4. Very much \n
"""

stai_item1_txt.draw()
win.flip()

keys = psychopy.event.waitKeys(keyList=["1", "2", "3", "4"])
stai_1.append(keys[0])
print(keys)

#item2
stai_item2_txt = psychopy.visual.TextStim(win = win, wrapWidth = 350, color = [-1, -1, -1])
stai_item2_txt.text = """
I am tense ... \n \n
1. Not at all \n
2. Somwhat \n
3. Moderately \n
4. Very much \n
"""

stai_item2_txt.draw()
win.flip()
```

```
keys = psychopy.event.waitKeys(keyList=["1", "2", "3", "4"])
stai_1.append(keys[0])
print(keys)

#item3
stai_item3_txt = psychopy.visual.TextStim(win = win, wrapWidth = 350, color = [-1, -1, -1])
stai_item3_txt.text = """
I feel upset ... \n \n
1. Not at all \n
2. Somwhat \n
3. Moderately \n
4. Very much \n
"""

stai_item3_txt.draw()
win.flip()

keys = psychopy.event.waitKeys(keyList=["1", "2", "3", "4"])
stai_1.append(keys[0])
print(keys)

#item4
stai_item4_txt = psychopy.visual.TextStim(win = win, wrapWidth = 350, color = [-1, -1, -1])
stai_item4_txt.text = """
I am relaxed ... \n \n
1. Not at all \n
2. Somwhat \n
3. Moderately \n
4. Very much \n
"""

stai_item4_txt.draw()
win.flip()

keys = psychopy.event.waitKeys(keyList=["1", "2", "3", "4"])
stai_1.append(keys[0])
print(keys)

#item5
stai_item5_txt = psychopy.visual.TextStim(win = win, wrapWidth = 350, color = [-1, -1, -1])
stai_item5_txt.text = """
I feel content ... \n \n
1. Not at all \n
2. Somwhat \n
3. Moderately \n
4. Very much \n
"""

stai_item5_txt.draw()
win.flip()

keys = psychopy.event.waitKeys(keyList=["1", "2", "3", "4"])
stai_1.append(keys[0])
print(keys)

#item6
stai_item6_txt = psychopy.visual.TextStim(win = win, wrapWidth = 350, color = [-1, -1, -1])
stai_item6_txt.text = """
I am worried ... \n \n
1. Not at all \n
2. Somwhat \n
3. Moderately \n
```

```
4. Very much \n
"""

stai_item6_txt.draw()
win.flip()

keys = psychopy.event.waitKeys(keyList=["1", "2", "3", "4"])
stai_1.append(keys[0])
print(keys)
print(stai_1)

#ALLEGED SING-A-SONG STRESS TEST
sasst_txt = psychopy.visual.TextStim(win = win, wrapWidth = 350, color = [-1, -1, -1])
sasst_txt.text = """
SING-A-SONG STRESS TEST \n
For this task you have to first think of a song and then perform 30 seconds of that song. \n
Therefore, you have two minutes to think of a song. \n
Afterwards, you have to perform 30 seconds of that song. \n
Press space to begin.
"""

sasst_txt.draw()
win.flip()

curr_time = time.time()
timestamps.append(curr_time)

psychopy.event.waitKeys(keyList=['space'])

sasst_txt = psychopy.visual.TextStim(win = win, wrapWidth = 350, color = [-1, -1, -1])
sasst_txt.text = """
You have two minutes to think of a song. \n
This can be any song you like or know.
"""

sasst_txt.draw()
win.flip()

curr_time = time.time()
timestamps.append(curr_time)

psychopy.core.wait(120) #120

sasst_txt = psychopy.visual.TextStim(win = win, wrapWidth = 350, color = [-1, -1, -1])
sasst_txt.text = """
Are you ready? \n
Press space to sing the song.
"""

sasst_txt.draw()
win.flip()

curr_time = time.time()
timestamps.append(curr_time)

psychopy.event.waitKeys(keyList=['space'])

sasst_txt = psychopy.visual.TextStim(win = win, wrapWidth = 350, color = [-1, -1, -1])
sasst_txt.text = """
Good news for you! \n
You do not have perform the song. \n
Press space to continue.
"""
```

```
sasst_txt.draw()
win.flip()

curr_time = time.time()
timestamps.append(curr_time)

psychopy.event.waitKeys(keyList=['space'])

#STAI QUESTIONNAIRE
stai_txt = psychopy.visual.TextStim(win = win, wrapWidth = 350, color = [-1, -1, -1])
stai_txt.text = """
Short State-Trait Anxiety Inventory \n
To select one of the answers, press key 1-4. \n
Press space to continue.
"""

stai_txt.draw()
win.flip()

curr_time = time.time()
timestamps.append(curr_time)

psychopy.event.waitKeys(keyList=['space'])

#item1
stai_item1_txt = psychopy.visual.TextStim(win = win, wrapWidth = 350, color = [-1, -1, -1])
stai_item1_txt.text = """
I feel calm ... \n \n
1. Not at all \n
2. Somwhat \n
3. Moderately \n
4. Very much \n
"""

stai_item1_txt.draw()
win.flip()

keys = psychopy.event.waitKeys(keyList=["1", "2", "3", "4"])
stai_2.append(keys[0])
print(keys)

#item2
stai_item2_txt = psychopy.visual.TextStim(win = win, wrapWidth = 350, color = [-1, -1, -1])
stai_item2_txt.text = """
I am tense ... \n \n
1. Not at all \n
2. Somwhat \n
3. Moderately \n
4. Very much \n
"""

stai_item2_txt.draw()
win.flip()

keys = psychopy.event.waitKeys(keyList=["1", "2", "3", "4"])
stai_2.append(keys[0])
print(keys)

#item3
stai_item3_txt = psychopy.visual.TextStim(win = win, wrapWidth = 350, color = [-1, -1, -1])
stai_item3_txt.text = """
I feel upset ... \n \n
```

```
1. Not at all \n
2. Somwhat \n
3. Moderately \n
4. Very much \n
"""

stai_item3_txt.draw()
win.flip()

keys = psychopy.event.waitKeys(keyList=["1", "2", "3", "4"])
stai_2.append(keys[0])
print(keys)

#item4
stai_item4_txt = psychopy.visual.TextStim(win = win, wrapWidth = 350, color = [-1, -1, -1])
stai_item4_txt.text = """
I am relaxed ... \n \n
1. Not at all \n
2. Somwhat \n
3. Moderately \n
4. Very much \n
"""

stai_item4_txt.draw()
win.flip()

keys = psychopy.event.waitKeys(keyList=["1", "2", "3", "4"])
stai_2.append(keys[0])
print(keys)

#item5
stai_item5_txt = psychopy.visual.TextStim(win = win, wrapWidth = 350, color = [-1, -1, -1])
stai_item5_txt.text = """
I feel content ... \n \n
1. Not at all \n
2. Somwhat \n
3. Moderately \n
4. Very much \n
"""

stai_item5_txt.draw()
win.flip()

keys = psychopy.event.waitKeys(keyList=["1", "2", "3", "4"])
stai_2.append(keys[0])
print(keys)

#item6
stai_item6_txt = psychopy.visual.TextStim(win = win, wrapWidth = 350, color = [-1, -1, -1])
stai_item6_txt.text = """
I am worried ... \n \n
1. Not at all \n
2. Somwhat \n
3. Moderately \n
4. Very much \n
"""

stai_item6_txt.draw()
win.flip()

keys = psychopy.event.waitKeys(keyList=["1", "2", "3", "4"])
stai_2.append(keys[0])
print(keys)
```

```
print(stai_2)

#RECOVERY PHASE
recovery_txt = psychopy.visual.TextStim(win = win, wrapWidth = 350, color = [-1, -1, -1])
recovery_txt.text = """
RECOVERY PHASE \n
This phase will take about 5 minutes.
"""

recovery_txt.draw()
win.flip()

curr_time = time.time()
timestamps.append(curr_time)

psychopy.core.wait(300) #300

recovery_txt.text = """
Press space to continue.
"""

recovery_txt.draw()
win.flip()

psychopy.event.waitKeys(keyList=['space'])

#STAI QUESTIONNAIRE
stai_txt = psychopy.visual.TextStim(win = win, wrapWidth = 350, color = [-1, -1, -1])
stai_txt.text = """
Short State-Trait Anxiety Inventory \n
To select one of the answers, press key 1-4. \n
Press space to continue.
"""

stai_txt.draw()
win.flip()

curr_time = time.time()
timestamps.append(curr_time)

psychopy.event.waitKeys(keyList=['space'])

#item1
stai_item1_txt = psychopy.visual.TextStim(win = win, wrapWidth = 350, color = [-1, -1, -1])
stai_item1_txt.text = """
I feel calm ... \n \n
1. Not at all \n
2. Somwhat \n
3. Moderately \n
4. Very much \n
"""

stai_item1_txt.draw()
win.flip()

keys = psychopy.event.waitKeys(keyList=["1", "2", "3", "4"])
stai_3.append(keys[0])
print(keys)

#item2
stai_item2_txt = psychopy.visual.TextStim(win = win, wrapWidth = 350, color = [-1, -1, -1])
stai_item2_txt.text = """
I am tense ... \n \n
```

```
1. Not at all \n
2. Somwhat \n
3. Moderately \n
4. Very much \n
"""

stai_item2_txt.draw()
win.flip()

keys = psychopy.event.waitKeys(keyList=["1", "2", "3", "4"])
stai_3.append(keys[0])
print(keys)

#item3
stai_item3_txt = psychopy.visual.TextStim(win = win, wrapWidth = 350, color = [-1, -1, -1])
stai_item3_txt.text = """
I feel upset ... \n \n
1. Not at all \n
2. Somwhat \n
3. Moderately \n
4. Very much \n
"""

stai_item3_txt.draw()
win.flip()

keys = psychopy.event.waitKeys(keyList=["1", "2", "3", "4"])
stai_3.append(keys[0])
print(keys)

#item4
stai_item4_txt = psychopy.visual.TextStim(win = win, wrapWidth = 350, color = [-1, -1, -1])
stai_item4_txt.text = """
I am relaxed ... \n \n
1. Not at all \n
2. Somwhat \n
3. Moderately \n
4. Very much \n
"""

stai_item4_txt.draw()
win.flip()

keys = psychopy.event.waitKeys(keyList=["1", "2", "3", "4"])
stai_3.append(keys[0])
print(keys)

#item5
stai_item5_txt = psychopy.visual.TextStim(win = win, wrapWidth = 350, color = [-1, -1, -1])
stai_item5_txt.text = """
I feel content ... \n \n
1. Not at all \n
2. Somwhat \n
3. Moderately \n
4. Very much \n
"""

stai_item5_txt.draw()
win.flip()

keys = psychopy.event.waitKeys(keyList=["1", "2", "3", "4"])
stai_3.append(keys[0])
print(keys)
```

```
#item6
stai_item6_txt = psychopy.visual.TextStim(win = win, wrapWidth = 350, color = [-1, -1, -1])
stai_item6_txt.text = """
I am worried ... \n \n
1. Not at all \n
2. Somwhat \n
3. Moderately \n
4. Very much \n
"""

stai_item6_txt.draw()
win.flip()

keys = psychopy.event.waitKeys(keyList=["1", "2", "3", "4"])
stai_3.append(keys[0])
print(keys)
print(stai_3)

#BASELINE MEASUREMENT
baseline_txt = psychopy.visual.TextStim(win = win, wrapWidth = 350, color = [-1, -1, -1])
baseline_txt.text = """
BASELINE MEASUREMENT \n
This phase will take about 5 minutes. \n
Please move as little as possible.
"""

baseline_txt.draw()
win.flip()

curr_time = time.time()
timestamps.append(curr_time)

psychopy.core.wait(300) #300

baseline_txt.text = """
Press space to continue. \n
"""

curr_time = time.time()
timestamps.append(curr_time)

baseline_txt.draw()
win.flip()

psychopy.event.waitKeys(keyList=['space'])

#STAI QUESTIONNAIRE
stai_txt = psychopy.visual.TextStim(win = win, wrapWidth = 350, color = [-1, -1, -1])
stai_txt.text = """
Short State-Trait Anxiety Inventory \n
To select one of the answers, press key 1-4. \n
Press space to continue.
"""

stai_txt.draw()
win.flip()

curr_time = time.time()
timestamps.append(curr_time)

psychopy.event.waitKeys(keyList=['space'])
```

```
#item1
stai_item1_txt = psychopy.visual.TextStim(win = win, wrapWidth = 350, color = [-1, -1, -1])
stai_item1_txt.text = """
I feel calm ... \n \n
1. Not at all \n
2. Somwhat \n
3. Moderately \n
4. Very much \n
"""

stai_item1_txt.draw()
win.flip()

keys = psychopy.event.waitKeys(keyList=["1", "2", "3", "4"])
stai_4.append(keys[0])
print(keys)

#item2
stai_item2_txt = psychopy.visual.TextStim(win = win, wrapWidth = 350, color = [-1, -1, -1])
stai_item2_txt.text = """
I am tense ... \n \n
1. Not at all \n
2. Somwhat \n
3. Moderately \n
4. Very much \n
"""

stai_item2_txt.draw()
win.flip()

keys = psychopy.event.waitKeys(keyList=["1", "2", "3", "4"])
stai_4.append(keys[0])
print(keys)

#item3
stai_item3_txt = psychopy.visual.TextStim(win = win, wrapWidth = 350, color = [-1, -1, -1])
stai_item3_txt.text = """
I feel upset ... \n \n
1. Not at all \n
2. Somwhat \n
3. Moderately \n
4. Very much \n
"""

stai_item3_txt.draw()
win.flip()

keys = psychopy.event.waitKeys(keyList=["1", "2", "3", "4"])
stai_4.append(keys[0])
print(keys)

#item4
stai_item4_txt = psychopy.visual.TextStim(win = win, wrapWidth = 350, color = [-1, -1, -1])
stai_item4_txt.text = """
I am relaxed ... \n \n
1. Not at all \n
2. Somwhat \n
3. Moderately \n
4. Very much \n
"""

stai_item4_txt.draw()
win.flip()
```

```
keys = psychopy.event.waitKeys(keyList=["1", "2", "3", "4"])
stai_4.append(keys[0])
print(keys)

#item5
stai_item5_txt = psychopy.visual.TextStim(win = win, wrapWidth = 350, color = [-1, -1, -1])
stai_item5_txt.text = """
I feel content ... \n \n
1. Not at all \n
2. Somwhat \n
3. Moderately \n
4. Very much \n
"""

stai_item5_txt.draw()
win.flip()

keys = psychopy.event.waitKeys(keyList=["1", "2", "3", "4"])
stai_4.append(keys[0])
print(keys)

#item6
stai_item6_txt = psychopy.visual.TextStim(win = win, wrapWidth = 350, color = [-1, -1, -1])
stai_item6_txt.text = """
I am worried ... \n \n
1. Not at all \n
2. Somwhat \n
3. Moderately \n
4. Very much \n
"""

stai_item6_txt.draw()
win.flip()

keys = psychopy.event.waitKeys(keyList=["1", "2", "3", "4"])
stai_4.append(keys[0])
print(keys)
print(stai_4)

#STROOP-COLOR-WORD TEST
# Colors abd screen
col_white = (250, 250, 250)
col_black = (0, 0, 0)
col_gray = (220, 220, 220)
col_red = (250, 0, 0)
col_green = (0, 200, 0)
col_blue = (0, 0, 250)
col_yellow = (250,250,0)

BACKGR_COL = col_gray
SCREEN_SIZE = (700, 500)


# Experiment
n_trials = 500

WORDS    = ("red", "green", "blue")

COLORS   = {"red": col_red,
            "green": col_green,
            "blue": col_blue}

```

```
KEYS      = {"red": K_b,
             "green": K_n,
             "blue": K_m}

pygame.init()
pygame.display.set_mode(SCREEN_SIZE)
pygame.display.set_caption("Stroop Test")

screen = pygame.display.get_surface()
# screen.fill(BACKGR_COL)

font = pygame.font.Font(None, 80)
font_small = pygame.font.Font(None, 40)

curr_time = time.time()
timestamps.append(curr_time)

def main():

    STATE = "welcome"
    trial_number = 0

    while True:

        # refreshing the surface
        screen.fill(BACKGR_COL)

        # Event loop
        for event in pygame.event.get():

            # interactive transitionals
            if STATE == "welcome":
                if event.type == KEYDOWN and event.key == K_SPACE:
                    STATE = "prepare_trial"
                    print(STATE)
                    continue

            if STATE == "trial":
                if event.type == KEYDOWN and event.key in KEYS.values():
                    time_when_reacted = time.time()
                    this_reaction_time = time_when_reacted - time_when_presented
                    this_correctness = (event.key == KEYS[this_color])
                    if trial_number < n_trials:
                        STATE = "prepare_trial"
                    else:
                        STATE = "goodbye"
                    print(STATE)
                    continue


            if event.type == QUIT:
                STATE = "quit"
                print(STATE)
                break


        # automatic transitionals
        if STATE == "prepare_trial":
                trial_number = trial_number + 1
                this_word  = pick_color()
                this_color = pick_color()
                time_when_presented = time.time()
                STATE = "trial"
```

```
                print(STATE)



        # Presentitionals
        if STATE == "welcome":
            draw_welcome()
            draw_button(SCREEN_SIZE[0]*1/5, 450, "Red: B", col_red)
            draw_button(SCREEN_SIZE[0]*3/5, 450, "Green: N", col_green)
            draw_button(SCREEN_SIZE[0]*4/5, 450, "Blue: M", col_blue)

        if STATE == "trial":
            draw_stimulus(this_color, this_word)
            draw_button(SCREEN_SIZE[0]*1/5, 450, "Red: B", col_red)
            draw_button(SCREEN_SIZE[0]*3/5, 450, "Green: N", col_green)
            draw_button(SCREEN_SIZE[0]*4/5, 450, "Blue: M", col_blue)

        if STATE == "goodbye":
            draw_goodbye()

        if STATE == "quit":
            pygame.quit()
            break

        # Updating the display
        pygame.display.update()


# Function definitions

def pick_color():
    """ Return a random word.
    """
    random_number = random.randint(0,2)
    return WORDS[random_number]

def draw_button(xpos, ypos, label, color):
    text = font_small.render(label, True, color, BACKGR_COL)
    text_rectangle = text.get_rect()
    text_rectangle.center = (xpos, ypos)
    screen.blit(text, text_rectangle)

def draw_welcome():
    text_surface = font.render("STROOP Experiment", True, col_black, BACKGR_COL)
    text_rectangle = text_surface.get_rect()
    text_rectangle.center = (SCREEN_SIZE[0]/2.0,150)
    screen.blit(text_surface, text_rectangle)
    text_surface = font_small.render("Press Spacebar to continue", True, col_black, BACKGR_COL)
    text_rectangle = text_surface.get_rect()
    text_rectangle.center = (SCREEN_SIZE[0]/2.0,300)
    screen.blit(text_surface, text_rectangle)

def draw_stimulus(color, word):
    text_surface = font.render(word, True, COLORS[color], col_gray)
    text_rectangle = text_surface.get_rect()
    text_rectangle.center = (SCREEN_SIZE[0]/2.0,150)
    screen.blit(text_surface, text_rectangle)

def draw_goodbye():
    text_surface = font_small.render("END OF THE EXPERIMENT", True, col_black, BACKGR_COL)
    text_rectangle = text_surface.get_rect()
    text_rectangle.center = (SCREEN_SIZE[0]/2.0,150)
```

```
    screen.blit(text_surface, text_rectangle)
    text_surface = font_small.render("Close the application.", True, col_black, BACKGR_COL)
    text_rectangle = text_surface.get_rect()
    text_rectangle.center = (SCREEN_SIZE[0]/2.0,200)
    screen.blit(text_surface, text_rectangle)

main()

curr_time = time.time()
timestamps.append(curr_time)

#STAI QUESTIONNAIRE
stai_txt = psychopy.visual.TextStim(win = win, wrapWidth = 350, color = [-1, -1, -1])
stai_txt.text = """
Short State-Trait Anxiety Inventory \n
To select one of the answers, press key 1-4. \n
Press space to continue.
"""

stai_txt.draw()
win.flip()

curr_time = time.time()
timestamps.append(curr_time)

psychopy.event.waitKeys(keyList=['space'])

#item1
stai_item1_txt = psychopy.visual.TextStim(win = win, wrapWidth = 350, color = [-1, -1, -1])
stai_item1_txt.text = """
I feel calm ... \n \n
1. Not at all \n
2. Somwhat \n
3. Moderately \n
4. Very much \n
"""

stai_item1_txt.draw()
win.flip()

keys = psychopy.event.waitKeys(keyList=["1", "2", "3", "4"])
stai_5.append(keys[0])
print(keys)

#item2
stai_item2_txt = psychopy.visual.TextStim(win = win, wrapWidth = 350, color = [-1, -1, -1])
stai_item2_txt.text = """
I am tense ... \n \n
1. Not at all \n
2. Somwhat \n
3. Moderately \n
4. Very much \n
"""

stai_item2_txt.draw()
win.flip()

keys = psychopy.event.waitKeys(keyList=["1", "2", "3", "4"])
stai_5.append(keys[0])
print(keys)

#item3
stai_item3_txt = psychopy.visual.TextStim(win = win, wrapWidth = 350, color = [-1, -1, -1])
```

```
stai_item3_txt.text = """
I feel upset ... \n \n
1. Not at all \n
2. Somwhat \n
3. Moderately \n
4. Very much \n
"""

stai_item3_txt.draw()
win.flip()

keys = psychopy.event.waitKeys(keyList=["1", "2", "3", "4"])
stai_5.append(keys[0])
print(keys)

#item4
stai_item4_txt = psychopy.visual.TextStim(win = win, wrapWidth = 350, color = [-1, -1, -1])
stai_item4_txt.text = """
I am relaxed ... \n \n
1. Not at all \n
2. Somwhat \n
3. Moderately \n
4. Very much \n
"""

stai_item4_txt.draw()
win.flip()

keys = psychopy.event.waitKeys(keyList=["1", "2", "3", "4"])
stai_5.append(keys[0])
print(keys)

#item5
stai_item5_txt = psychopy.visual.TextStim(win = win, wrapWidth = 350, color = [-1, -1, -1])
stai_item5_txt.text = """
I feel content ... \n \n
1. Not at all \n
2. Somwhat \n
3. Moderately \n
4. Very much \n
"""

stai_item5_txt.draw()
win.flip()

keys = psychopy.event.waitKeys(keyList=["1", "2", "3", "4"])
stai_5.append(keys[0])
print(keys)

#item6
stai_item6_txt = psychopy.visual.TextStim(win = win, wrapWidth = 350, color = [-1, -1, -1])
stai_item6_txt.text = """
I am worried ... \n \n
1. Not at all \n
2. Somwhat \n
3. Moderately \n
4. Very much \n
"""

stai_item6_txt.draw()
win.flip()

keys = psychopy.event.waitKeys(keyList=["1", "2", "3", "4"])
```

```
stai_5.append(keys[0])
print(keys)
print(stai_5)

#RECOVERY PHASE
recovery_txt = psychopy.visual.TextStim(win = win, wrapWidth = 350, color = [-1, -1, -1])
recovery_txt.text = """
RECOVERY PHASE \n
This phase will take about 5 minutes.
"""

recovery_txt.draw()
win.flip()

curr_time = time.time()
timestamps.append(curr_time)

psychopy.core.wait(300) #300

recovery_txt.text = """
Press space to continue.
"""

recovery_txt.draw()
win.flip()

psychopy.event.waitKeys(keyList=['space'])

#STAI QUESTIONNAIRE
stai_txt = psychopy.visual.TextStim(win = win, wrapWidth = 350, color = [-1, -1, -1])
stai_txt.text = """
Short State-Trait Anxiety Inventory \n
To select one of the answers, press key 1-4. \n
Press space to continue.
"""

stai_txt.draw()
win.flip()

curr_time = time.time()
timestamps.append(curr_time)

psychopy.event.waitKeys(keyList=['space'])

#item1
stai_item1_txt = psychopy.visual.TextStim(win = win, wrapWidth = 350, color = [-1, -1, -1])
stai_item1_txt.text = """
I feel calm ... \n \n
1. Not at all \n
2. Somwhat \n
3. Moderately \n
4. Very much \n
"""

stai_item1_txt.draw()
win.flip()

keys = psychopy.event.waitKeys(keyList=["1", "2", "3", "4"])
stai_6.append(keys[0])
print(keys)

#item2
stai_item2_txt = psychopy.visual.TextStim(win = win, wrapWidth = 350, color = [-1, -1, -1])
```

```
stai_item2_txt.text = """
I am tense ... \n \n
1. Not at all \n
2. Somwhat \n
3. Moderately \n
4. Very much \n
"""

stai_item2_txt.draw()
win.flip()

keys = psychopy.event.waitKeys(keyList=["1", "2", "3", "4"])
stai_6.append(keys[0])
print(keys)

#item3
stai_item3_txt = psychopy.visual.TextStim(win = win, wrapWidth = 350, color = [-1, -1, -1])
stai_item3_txt.text = """
I feel upset ... \n \n
1. Not at all \n
2. Somwhat \n
3. Moderately \n
4. Very much \n
"""

stai_item3_txt.draw()
win.flip()

keys = psychopy.event.waitKeys(keyList=["1", "2", "3", "4"])
stai_6.append(keys[0])
print(keys)

#item4
stai_item4_txt = psychopy.visual.TextStim(win = win, wrapWidth = 350, color = [-1, -1, -1])
stai_item4_txt.text = """
I am relaxed ... \n \n
1. Not at all \n
2. Somwhat \n
3. Moderately \n
4. Very much \n
"""

stai_item4_txt.draw()
win.flip()

keys = psychopy.event.waitKeys(keyList=["1", "2", "3", "4"])
stai_6.append(keys[0])
print(keys)

#item5
stai_item5_txt = psychopy.visual.TextStim(win = win, wrapWidth = 350, color = [-1, -1, -1])
stai_item5_txt.text = """
I feel content ... \n \n
1. Not at all \n
2. Somwhat \n
3. Moderately \n
4. Very much \n
"""

stai_item5_txt.draw()
win.flip()

keys = psychopy.event.waitKeys(keyList=["1", "2", "3", "4"])
```

```
stai_6.append(keys[0])
print(keys)

#item6
stai_item6_txt = psychopy.visual.TextStim(win = win, wrapWidth = 350, color = [-1, -1, -1])
stai_item6_txt.text = """
I am worried ... \n \n
1. Not at all \n
2. Somwhat \n
3. Moderately \n
4. Very much \n
"""

stai_item6_txt.draw()
win.flip()

keys = psychopy.event.waitKeys(keyList=["1", "2", "3", "4"])
stai_6.append(keys[0])
print(keys)
print(stai_6)

curr_time = time.time()
timestamps.append(curr_time)

#CLOSING WINDOW
end_txt = psychopy.visual.TextStim(win = win, wrapWidth = 350, color = [-1, -1, -1])
end_txt.text = """
Thank you for participating! \n
You reached the end of the Experiment. \n
Press space to close the application.
"""

end_txt.draw()
win.flip()

curr_time = time.time()
timestamps.append(curr_time)

psychopy.event.waitKeys(keyList=['space'])

#save data to csv-files
np.savetxt("p_" + subj_id + "_time_data.csv", timestamps, delimiter=",")

for x in stai_1:
    a = int(x)
    int_stai1.append(a)

for x in stai_2:
    a = int(x)
    int_stai2.append(a)

for x in stai_3:
    a = int(x)
    int_stai3.append(a)

for x in stai_4:
    a = int(x)
    int_stai4.append(a)

for x in stai_5:
    a = int(x)
    int_stai5.append(a)
```

```
for x in stai_6:
    a = int(x)
    int_stai6.append(a)

np.savetxt("p_" + subj_id + "_stai1_data.csv", int_stai1, delimiter=",")
np.savetxt("p_" + subj_id + "_stai2_data.csv", int_stai2, delimiter=",")
np.savetxt("p_" + subj_id + "_stai3_data.csv", int_stai3, delimiter=",")
np.savetxt("p_" + subj_id + "_stai4_data.csv", int_stai4, delimiter=",")
np.savetxt("p_" + subj_id + "_stai5_data.csv", int_stai5, delimiter=",")
np.savetxt("p_" + subj_id + "_stai6_data.csv", int_stai6, delimiter=",")

win.close()
```

Appendix C – Normalized STAI-Scores Graphic (R-Script)

```r
rm(list = ls())
#########Task1
library(ggplot2)
library(grDevices)
install.packages("gdata")
library(gdata)
#perl <- gdata:::findPerl("perl")
stai <- read.xls("STAI_Normalized.xls", sheet=1)

quartz(width=10, height=5)
ggplot(stai, aes(x = factor(Phase, level = c('T1 (before SSST)', 'T2 (after SSST)', 'T3 (after Recovery)',
                                             'T4 (before Stroop)', 'T5 (after Stroop)', 'T6 (after Recovery)')), group=1)) +
  geom_line(aes(y=p1), size = 1, color= "darkgrey") +
  geom_line(aes(y=p2), size = 1, color= "darkgrey")+
  geom_line(aes(y=p3), size = 1, color= "darkgrey")+
  geom_line(aes(y=p4), size = 1, color= "darkgrey")+
  geom_line(aes(y=p5), size = 1, color= "darkgrey")+
  geom_line(aes(y=p6), size = 1, color= "darkgrey")+
  geom_line(aes(y=p7), size = 1, color= "darkgrey")+
  geom_line(aes(y=p8), size = 1, color= "darkgrey")+
  geom_line(aes(y=p9), size = 1, color= "darkgrey")+
  geom_line(aes(y=p10), size = 1, color= "darkgrey")+
  geom_line(aes(y=p11), size = 1, color= "darkgrey")+
  geom_line(aes(y=p12), size = 1, color= "darkgrey")+
  geom_line(aes(y=p13), size = 1, color= "darkgrey")+
  geom_line(aes(y=p14), size = 1, color= "darkgrey")+
  geom_line(aes(y=p15), size = 1, color= "darkgrey")+
  geom_line(aes(y=p16), size = 1, color= "darkgrey")+
  geom_line(aes(y=p17), size = 1, color= "darkgrey")+
  geom_line(aes(y=p18), size = 1, color= "darkgrey")+
  geom_line(aes(y=p19), size = 1, color= "darkgrey")+
  geom_line(aes(y=p20), size = 1, color= "darkgrey")+
  geom_line(aes(y=mean), size= 1, color = "red")+
  labs(title = "STAI", x="",y = "Anxiety")+
  theme(axis.text.x = element_text(angle = 70, hjust = 1))
```

Appendix D – Visual Inspection Graphs (R-Script)

```
rm(list = ls())
####################################PPG_BS1####################################

path <- "Masterarbeit/MASTERDATA/PPG/csv/"
files_e4 <- c(list.files(path)[grepl('PPG_E4_BS1',list.files(path))])
files_s3 <- c(list.files(path)[grepl('PPG_S3_BS1',list.files(path))])
name_e4 <- gsub("PPG_E4_BS1", "", gsub(".csv", "", files_e4))
name_s3 <- gsub("PPG_S3_BS1", "", gsub(".csv", "", files_s3))
for (u in 1:length(files_e4)){
  e4_bs1 <-read.csv(paste0(path,files_e4[u]), header=TRUE, sep = ",", quote = "\"'", dec = ".", fill = TRUE)

for (o in 1:length(files_s3)){
    s3_bs1 <-read.csv(paste0(path,files_s3[o]), header=TRUE, sep = ",", quote = "\"'", dec = ".", fill = TRUE)

  ###Zeit  dern
  e4_bs1$time <- as.POSIXct(as.numeric(e4_bs1$TIME)/1000, origin = '1970-01-01', tz = 'GMT')
  s3_bs1$time <- as.POSIXct(as.numeric(s3_bs1$TIME)/1000, origin = '1970-01-01', tz = 'GMT')

  ###plots erstellen
  jpeg(filename=paste0("Masterarbeit/MASTERDATA/PPG/Plots/BS1/","plot_ppg_bs1_s3", name_s3[o],".jpeg"), width = 1000, height = 500)
  plot(s3_bs1$time, s3_bs1$PPG_S3_BS1,type ="l", main = paste0("Plot PPG S3 BS1", name_s3[o]), xlab = "zeit", ylab = "s3_bs1")
  dev.off()

  jpeg(filename=paste0("Masterarbeit/MASTERDATA/PPG/Plots/BS1/","plot_ppg_bs1_e4", name_e4[u],".jpeg"), width = 1000, height = 500)
  plot(e4_bs1$time, e4_bs1$PPG_E4_BS1,type ="l",  main = paste0("Plot PPG E4 BS1", name_e4[u]), xlab = "zeit", ylab = "e4_bs1")
  dev.off()

save(file=paste0("Masterarbeit/MASTERDATA/PPG/S3/S3_BS1", name_s3[o],".RData"), s3_bs1,
    compress = T)
}#ende for files_s3

save(file=paste0("Masterarbeit/MASTERDATA/PPG/E4/E4_BS1", name_e4[u],".RData"), e4_bs1,
    compress = T)
}#ende for files_e4


rm(list = ls())
####################################PPG_ST1####################################

path <- "Masterarbeit/MASTERDATA/PPG/csv/"
files_e4 <- c(list.files(path)[grepl('PPG_E4_ST1',list.files(path))])
files_s3 <- c(list.files(path)[grepl('PPG_S3_ST1',list.files(path))])
name_e4 <- gsub("PPG_E4_ST1", "", gsub(".csv", "", files_e4))
name_s3 <- gsub("PPG_S3_ST1", "", gsub(".csv", "", files_s3))
for (u in 1:length(files_e4)){
  e4_st1 <-read.csv(paste0(path,files_e4[u]), header=TRUE, sep = ",", quote = "\"'", dec = ".", fill = TRUE)

  for (o in 1:length(files_s3)){
    s3_st1 <-read.csv(paste0(path,files_s3[o]), header=TRUE, sep = ",", quote = "\"'", dec = ".", fill = TRUE)

    ###Zeit  dern
    e4_st1$time <- as.POSIXct(as.numeric(e4_st1$TIME)/1000, origin = '1970-01-01', tz = 'GMT')
    s3_st1$time <- as.POSIXct(as.numeric(s3_st1$TIME)/1000, origin = '1970-01-01', tz = 'GMT')
```

```
    ###plots erstellen
    jpeg(filename=paste0("Masterarbeit/MASTERDATA/PPG/Plots/ST1/","plot_ppg_st1_s3", name_s3[o],".jpeg"), width = 1000, height = 500)
    plot(s3_st1$time, s3_st1$PPG_S3_ST1,type ="l", main = paste0("Plot PPG S3 ST1", name_s3[o]), xlab = "zeit", ylab = "s3_st1")
    dev.off()

    jpeg(filename=paste0("Masterarbeit/MASTERDATA/PPG/Plots/ST1/","plot_ppg_st1_e4", name_e4[u],".jpeg"), width = 1000, height = 500)
    plot(e4_st1$time, e4_st1$PPG_E4_ST1,type ="l",  main = paste0("Plot PPG E4 ST1", name_e4[u]), xlab = "zeit", ylab = "e4_st1")
    dev.off()

    save(file=paste0("Masterarbeit/MASTERDATA/PPG/S3/S3_ST1", name_s3[o],".RData"), s3_st1, compress = T)
  }#ende for files_s3
  save(file=paste0("Masterarbeit/MASTERDATA/PPG/E4/E4_ST1", name_e4[u],".RData"), e4_st1, compress = T)
}#ende for files_e4


rm(list = ls())
###################################PPG_RC1##################################
path <- "Masterarbeit/MASTERDATA/PPG/csv/"
files_e4 <- c(list.files(path)[grepl('PPG_E4_RC1',list.files(path))])
files_s3 <- c(list.files(path)[grepl('PPG_S3_RC1',list.files(path))])
name_e4 <- gsub("PPG_E4_RC1", "", gsub(".csv", "", files_e4))
name_s3 <- gsub("PPG_S3_RC1", "", gsub(".csv", "", files_s3))
for (u in 1:length(files_e4)){
  e4_rc1 <-read.csv(paste0(path,files_e4[u]), header=TRUE, sep = ",", quote = "\"'", dec = ".", fill = TRUE)

  for (o in 1:length(files_s3)){
    s3_rc1 <-read.csv(paste0(path,files_s3[o]), header=TRUE, sep = ",", quote = "\"'", dec = ".", fill = TRUE)

    ###Zeit  dern
    e4_rc1$time <- as.POSIXct(as.numeric(e4_rc1$TIME)/1000, origin = '1970-01-01', tz = 'GMT')
    s3_rc1$time <- as.POSIXct(as.numeric(s3_rc1$TIME)/1000, origin = '1970-01-01', tz = 'GMT')

    ###plots erstellen
    jpeg(filename=paste0("Masterarbeit/MASTERDATA/PPG/Plots/RC1/","plot_ppg_rc1_s3", name_s3[o],".jpeg"), width = 1000, height = 500)
    plot(s3_rc1$time, s3_rc1$PPG_S3_RC1,type ="l", main = paste0("Plot PPG S3 RC1", name_s3[o]), xlab = "zeit", ylab = "s3_rc1")
    dev.off()

    jpeg(filename=paste0("Masterarbeit/MASTERDATA/PPG/Plots/RC1/","plot_ppg_rc1_e4", name_e4[u],".jpeg"), width = 1000, height = 500)
    plot(e4_rc1$time, e4_rc1$PPG_E4_RC1,type ="l",  main = paste0("Plot PPG E4 RC1", name_e4[u]), xlab = "zeit", ylab = "e4_rc1")
    dev.off()

    save(file=paste0("Masterarbeit/MASTERDATA/PPG/S3/S3_RC1", name_s3[o],".RData"), s3_rc1, compress = T)
  }#ende for files_s3
  save(file=paste0("Masterarbeit/MASTERDATA/PPG/E4/E4_RC1", name_e4[u],".RData"), e4_rc1, compress = T)
}#ende for files_e4
```

```
rm(list = ls())
####################################PPG_BS2####################################
path <- "Masterarbeit/MASTERDATA/PPG/csv/"
files_e4 <- c(list.files(path)[grepl('PPG_E4_BS2',list.files(path))])
files_s3 <- c(list.files(path)[grepl('PPG_S3_BS2',list.files(path))])
name_e4 <- gsub("PPG_E4_BS2", "", gsub(".csv", "", files_e4))
name_s3 <- gsub("PPG_S3_BS2", "", gsub(".csv", "", files_s3))
for (u in 1:length(files_e4)){
  e4_bs2 <-read.csv(paste0(path,files_e4[u]), header=TRUE, sep = ",", quote = "\"'", dec = ".", fill = TRUE)

  for (o in 1:length(files_s3)){
    s3_bs2 <-read.csv(paste0(path,files_s3[o]), header=TRUE, sep = ",", quote = "\"'", dec = ".", fill = TRUE)

    ###Zeit  dern
    e4_bs2$time <- as.POSIXct(as.numeric(e4_bs2$TIME)/1000, origin = '1970-01-01', tz = 'GMT')
    s3_bs2$time <- as.POSIXct(as.numeric(s3_bs2$TIME)/1000, origin = '1970-01-01', tz = 'GMT')


    ###plots erstellen
    jpeg(filename=paste0("Masterarbeit/MASTERDATA/PPG/Plots/BS2/","plot_ppg_bs2_s3", name_s3[o],".jpeg"), width = 1000, height = 500)
    plot(s3_bs2$time, s3_bs2$PPG_S3_BS2,type ="l", main = paste0("Plot PPG S3 BS2", name_s3[o]), xlab = "zeit", ylab = "s3_bs2")
    dev.off()

    jpeg(filename=paste0("Masterarbeit/MASTERDATA/PPG/Plots/BS2/","plot_ppg_bs2_e4", name_e4[u],".jpeg"), width = 1000, height = 500)
    plot(e4_bs2$time, e4_bs2$PPG_E4_BS2,type ="l",  main = paste0("Plot PPG E4 BS2", name_e4[u]), xlab = "zeit", ylab = "e4_bs2")
    dev.off()

    save(file=paste0("Masterarbeit/MASTERDATA/PPG/S3/S3_BS2", name_s3[o],".RData"), s3_bs2, compress = T)
  }#ende for files_s3
  save(file=paste0("Masterarbeit/MASTERDATA/PPG/E4/E4_BS2", name_e4[u],".RData"), e4_bs2, compress = T)
}#ende for files_e4


rm(list = ls())
####################################PPG_RC1####################################
path <- "Masterarbeit/MASTERDATA/PPG/csv/"
files_e4 <- c(list.files(path)[grepl('PPG_E4_ST2',list.files(path))])
files_s3 <- c(list.files(path)[grepl('PPG_S3_ST2',list.files(path))])
name_e4 <- gsub("PPG_E4_ST2", "", gsub(".csv", "", files_e4))
name_s3 <- gsub("PPG_S3_ST2", "", gsub(".csv", "", files_s3))
for (u in 1:length(files_e4)){
  e4_st2 <-read.csv(paste0(path,files_e4[u]), header=TRUE, sep = ",", quote = "\"'", dec = ".", fill = TRUE)

  for (o in 1:length(files_s3)){
    s3_st2 <-read.csv(paste0(path,files_s3[o]), header=TRUE, sep = ",", quote = "\"'", dec = ".", fill = TRUE)

    ###Zeit  dern
    e4_st2$time <- as.POSIXct(as.numeric(e4_st2$TIME)/1000, origin = '1970-01-01', tz = 'GMT')
    s3_st2$time <- as.POSIXct(as.numeric(s3_st2$TIME)/1000, origin = '1970-01-01', tz = 'GMT
```

```
')

    ###plots erstellen
    jpeg(filename=paste0("Masterarbeit/MASTERDATA/PPG/Plots/ST2/","plot_ppg_st2_s3", name_s3[o],".jpeg"), width = 1000, height = 500)
    plot(s3_st2$time, s3_st2$PPG_S3_ST2,type ="l", main = paste0("Plot PPG S3 ST2", name_s3[o]), xlab = "zeit", ylab = "s3_st2")
    dev.off()

    jpeg(filename=paste0("Masterarbeit/MASTERDATA/PPG/Plots/ST2/","plot_ppg_st2_e4", name_e4[u],".jpeg"), width = 1000, height = 500)
    plot(e4_st2$time, e4_st2$PPG_E4_ST2,type ="l",  main = paste0("Plot PPG E4 ST2", name_e4[u]), xlab = "zeit", ylab = "e4_st2")
    dev.off()

    save(file=paste0("Masterarbeit/MASTERDATA/PPG/S3/S3_ST2", name_s3[o],".RData"), s3_st2, compress = T)
  }#ende for files_s3
  save(file=paste0("Masterarbeit/MASTERDATA/PPG/E4/E4_ST2", name_e4[u],".RData"), e4_st2, compress = T)
}#ende for files_e4



rm(list = ls())
##################################PPG_RC2##################################
path <- "Masterarbeit/MASTERDATA/PPG/csv/"
files_e4 <- c(list.files(path)[grepl('PPG_E4_RC2',list.files(path))])
files_s3 <- c(list.files(path)[grepl('PPG_S3_RC2',list.files(path))])
name_e4 <- gsub("PPG_E4_RC2", "", gsub(".csv", "", files_e4))
name_s3 <- gsub("PPG_S3_RC2", "", gsub(".csv", "", files_s3))
for (u in 1:length(files_e4)){
  e4_rc2 <-read.csv(paste0(path,files_e4[u]), header=TRUE, sep = ",", quote = "\"'", dec = ".", fill = TRUE)

  for (o in 1:length(files_s3)){
    s3_rc2 <-read.csv(paste0(path,files_s3[o]), header=TRUE, sep = ",", quote = "\"'", dec = ".", fill = TRUE)

    ###Zeit dern
    e4_rc2$time <- as.POSIXct(as.numeric(e4_rc2$TIME)/1000, origin = '1970-01-01', tz = 'GMT')
    s3_rc2$time <- as.POSIXct(as.numeric(s3_rc2$TIME)/1000, origin = '1970-01-01', tz = 'GMT')

    ###plots erstellen
    jpeg(filename=paste0("Masterarbeit/MASTERDATA/PPG/Plots/RC2/","plot_ppg_rc2_s3", name_s3[o],".jpeg"), width = 1000, height = 500)
    plot(s3_rc2$time, s3_rc2$PPG_S3_RC2,type ="l", main = paste0("Plot PPG S3 RC2", name_s3[o]), xlab = "zeit", ylab = "s3_rc2")
    dev.off()

    jpeg(filename=paste0("Masterarbeit/MASTERDATA/PPG/Plots/RC2/","plot_ppg_rc2_e4", name_e4[u],".jpeg"), width = 1000, height = 500)
    plot(e4_rc2$time, e4_rc2$PPG_E4_RC2,type ="l",  main = paste0("Plot PPG E4 RC2", name_e4[u]), xlab = "zeit", ylab = "e4_rc2")
    dev.off()

    save(file=paste0("Masterarbeit/MASTERDATA/PPG/S3/S3_RC2", name_s3[o],".RData"), s3_rc2, compress = T)
  }#ende for files_s3
  save(file=paste0("Masterarbeit/MASTERDATA/PPG/E4/E4_RC2", name_e4[u],".RData"), e4_rc2, compress = T)
```

```
}#ende for files_e4
```

Appendix E – Pre-processing & Extraction of Parameters (Python-Script)

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Fri May  8 21:48:01 2020

@author: Niklas Enewoldsen
"""

#IMPORTS
import os
import csv
import time
import datetime
import os.path

import numpy as np
import scipy as sp
import heartpy as hp
import matplotlib.pyplot as plt


#SET DIRECTORY
os.chdir('/Users/ene/Documents/Master')

#VARIABLES
time_s3_adj = []
ppg_s3_adj  = []
ibi_s3_adj  = []
hr_s3_adj   = []

bvp_e4_adj = []
ibi_e4_adj = []
ibt_e4_adj = []
hr_e4_adj  = []

time_units = []

timestamps_bvp = [0,
                  1558607821000, 1558611680000, 1558615230000, 1558618648000,
                  1558625271000, 1558687078000, 1558690199000, 1558693158000,
                  1558697905000, 1558700019000, 1558703299000, 1558706377000,
                  1558898720000, 1558901276000, 1558906390000, 1558949721000,
                  1558954011000, 1558956966000, 1558959780000]
timestamps_ibi = [0,
                  1558607821000, 1558611680000, 1558615230000, 1558618648000,
                  1558625271000, 1558687078000, 1558690199000, 1558693158000,
                  1558697905000, 1558700019000, 1558703299000, 1558706377000,
                  1558898720000, 1558901276000, 1558906390000, 1558949721000,
                  1558954011000, 1558956966000, 1558959780000]
timestamps_hr  = [0,
                  1558607831000, 1558611690000, 1558615240000, 1558618658000,
                  1558625281000, 1558687088000, 1558690209000, 1558693168000,
                  1558697915000, 1558700029000, 1558703309000, 1558706387000,
                  1558898730000, 1558901286000, 1558906400000, 1558949731000,
                  1558954021000, 1558956976000, 1558959790000]

start_time_bs1 = [0,
                  1558608133920, 1558611962237, 1558615527392, 1558618965540,
                  1558625002762, 1558687412138, 1558690509015, 1558693324066,
                  1558697942775, 1558700305016, 1558703615416, 1558706670308,
                  1558899026943, 1558901591781, 1558906663802, 1558949990710,
                  1558954321215, 1558956940916, 1558959756152] #03
start_time_st1 = [0,
```

```
                     1558608473183, 1558612349524, 1558615867258, 1558619321992,
                     1558625333711, 1558687760303, 1558690893903, 1558693659648,
                     1558698285324, 1558700634014, 1558704134017, 1558707003157,
                     1558899399654, 1558901936828, 1558907060205, 1558950373509,
                     1558954672545, 1558957301832, 1558960083546] #06
start_time_rc1 = [0,
                     1558608625747, 1558612566845, 1558616084645, 1558619497810,
                     1558625495543, 1558687925302, 1558691087404, 1558693839182,
                     1558698464908, 1558700804581, 1558704299796, 1558707218312,
                     1558899567267, 1558902138162, 1558907259441, 1558950578574,
                     1558954849976, 1558957488180, 1558960248177] #11
start_time_bs2 = [0,
                     1558608942642, 1558612884208, 1558616411459, 1558619824108,
                     1558625808373, 1558688250633, 1558691418268, 1558694169314,
                     1558698784139, 1558701132546, 1558704629527, 1558707537008,
                     1558899897594, 1558902472038, 1558907628608, 1558950917919,
                     1558955175853, 1558957822890, 1558960652160] #13
start_time_st2 = [0,
                     1558609271761, 1558613196952, 1558616746852, 1558620149699,
                     1558626120623, 1558688587293, 1558691754580, 1558694584715,
                     1558699099464, 1558701455166, 1558705001790, 1558707867963,
                     1558900217589, 1558902830863, 1558907964789, 1558951247997,
                     1558955490820, 1558958165820, 1558960965563] #16
start_time_rc2 = [0,
                     1558609629065, 1558613625461, 1558617265974, 1558620663565,
                     1558626439250, 1558689097064, 1558692195459, 1558695001310,
                     1558699535662, 1558701822095, 1558705536296, 1558708560058,
                     1558900612244, 1558903341259, 1558908414307, 1558951746772,
                     1558955914398, 1558958743752, 1558961349168] #19

end_time_bs1   = [0,
                     1558608434293, 1558612262266, 1558615827422, 1558619265572,
                     1558625302793, 1558687712166, 1558690809044, 1558693624133,
                     1558698242804, 1558700605074, 1558703915446, 1558706970346,
                     1558899326966, 1558901891807, 1558906963830, 1558950290737,
                     1558954621241, 1558957240943, 1558960056197] #04
end_time_st1   = [0,
                     1558608602094, 1558612542393, 1558616050208, 1558619463956,
                     1558625472374, 1558687897983, 1558691052200, 1558693805178,
                     1558698438988, 1558700783395, 1558704275144, 1558707198709,
                     1558899540882, 1558902077407, 1558907220936, 1558950544456,
                     1558954818590, 1558957449209, 1558960216890] #10
end_time_rc1   = [0,
                     1558608927557, 1558612868456, 1558616387540, 1558619802955,
                     1558625798038, 1558688232165, 1558691388498, 1558694140911,
                     1558698766553, 1558701106726, 1558704611859, 1558707521657,
                     1558899872423, 1558902441451, 1558907585255, 1558950880566,
                     1558955152484, 1558957792405, 1558960633344] #12
end_time_bs2   = [0,
                     1558609242679, 1558613184224, 1558616711475, 1558620124124,
                     1558626108389, 1558688550648, 1558691718284, 1558694469363,
                     1558699084154, 1558701432561, 1558704929543, 1558707837037,
                     1558900197613, 1558902772055, 1558907928647, 1558951217943,
                     1558955475869, 1558958122907, 1558960952177] #14
end_time_st2   = [0,
                     1558609606225, 1558613596766, 1558617238830, 1558620646297,
                     1558626419967, 1558689070121, 1558692166783, 1558694981943,
                     1558699516354, 1558701805601, 1558705516774, 1558708534361,
                     1558900588289, 1558903314134, 1558908388093, 1558951718944,
                     1558955883535, 1558958699976, 1558961327858] #17
end_time_rc2   = [0,
                     1558610040841, 1558613944259, 1558617571186, 1558620965693,
                     1558626743562, 1558689407144, 1558692497904, 1558695291144,
```

```
                  1558699838774, 1558702123290, 1558705842825, 1558708861870,
                  1558900914170, 1558903649906, 1558908756459, 1558952049213,
                  1558956218257, 1558959049410, 1558961650660] #20

ppg_s3_bs1  = []
ppg_s3_st1  = []
ppg_s3_rc1  = []
ppg_s3_bs2  = []
ppg_s3_st2  = []
ppg_s3_rc2  = []

ibi_s3_bs1  = []
ibi_s3_st1  = []
ibi_s3_rc1  = []
ibi_s3_bs2  = []
ibi_s3_st2  = []
ibi_s3_rc2  = []

hr_s3_bs1   = []
hr_s3_st1   = []
hr_s3_rc1   = []
hr_s3_bs2   = []
hr_s3_st2   = []
hr_s3_rc2   = []

bvp_e4_bs1  = []
bvp_e4_st1  = []
bvp_e4_rc1  = []
bvp_e4_bs2  = []
bvp_e4_st2  = []
bvp_e4_rc2  = []

ibi_e4_bs1  = []
ibi_e4_st1  = []
ibi_e4_rc1  = []
ibi_e4_bs2  = []
ibi_e4_st2  = []
ibi_e4_rc2  = []

hr_e4_bs1   = []
hr_e4_st1   = []
hr_e4_rc1   = []
hr_e4_bs2   = []
hr_e4_st2   = []
hr_e4_rc2   = []

tppg_s3_bs1 = []
tppg_s3_st1 = []
tppg_s3_rc1 = []
tppg_s3_bs2 = []
tppg_s3_st2 = []
tppg_s3_rc2 = []

tibi_s3_bs1 = []
tibi_s3_st1 = []
tibi_s3_rc1 = []
tibi_s3_bs2 = []
tibi_s3_st2 = []
tibi_s3_rc2 = []

thr_s3_bs1  = []
thr_s3_st1  = []
thr_s3_rc1  = []
```

```
thr_s3_bs2  = []
thr_s3_st2  = []
thr_s3_rc2  = []

tbvp_e4_bs1 = []
tbvp_e4_st1 = []
tbvp_e4_rc1 = []
tbvp_e4_bs2 = []
tbvp_e4_st2 = []
tbvp_e4_rc2 = []

tibi_e4_bs1 = []
tibi_e4_st1 = []
tibi_e4_rc1 = []
tibi_e4_bs2 = []
tibi_e4_st2 = []
tibi_e4_rc2 = []

thr_e4_bs1  = []
thr_e4_st1  = []
thr_e4_rc1  = []
thr_e4_bs2  = []
thr_e4_st2  = []
thr_e4_rc2  = []

subjects = []
bpm_e4   = []
bpm_s3   = []
ibi_e4   = []
ibi_s3   = []
rmssd_e4 = []
rmssd_s3 = []
sdnn_e4  = []
sdnn_s3  = []

zero_ibi_bs1 = False
zero_ibi_st1 = False
zero_ibi_rc1 = False
zero_ibi_bs2 = False
zero_ibi_st2 = False
zero_ibi_rc2 = False

zero_hr_bs1 = False
zero_hr_st1 = False
zero_hr_rc1 = False
zero_hr_bs2 = False
zero_hr_st2 = False
zero_hr_rc2 = False

corr_coef = []
corr_mode = []

modes          = ['Modes']
bpm            = ['Bpm']
breathingrate  = ['Breathingrate']
hf             = ['Hf']
hr_mad         = ['Hr_mad']
ibi            = ['Ibi']
lf             = ['Lf']
lf_hf          = ['Lf/Hf']
pnn20          = ['Pnn20']
pnn50          = ['Pnn50']
rmssd          = ['Rmssd']
```

```python
s               = ['S']
sd1             = ['Sd1']
sd1_sd2         = ['Sd1/Sd2']
sd2             = ['Sd2']
sdnn            = ['Sdnn']
sdsd            = ['Sdsd']

subjects = 19

#FUNCTIONS

def get_s3_units(list):
    unit = list[0]
    #print('Unit: ' + unit)

    time_units.append(unit)

    adj_list = np.delete(list, 0)

    return adj_list

def get_bvp_timestamp(list):
    timestamp = list[0]
    print('Timestamp: ' + str(timestamp))

    #timestamps_bvp.append(timestamp)

    adj_list = np.delete(list, 0)
    adj_list = np.delete(list, 0)

    return adj_list

def get_bvp_length(subj):
    data_path = 'p_' + str(subj) + '_BVP.csv'
    print('CurrentFile: ' + data_path)

    bvp_e4_raw = hp.get_data(data_path, column_name = 'BVP')
    bvp_e4_adj = get_bvp_timestamp(bvp_e4_raw)

    y = len(bvp_e4_adj)

    print(str(data_path) + ' Length: ' + str(y))

def downsampling_check(data_list, time_list):
    a = len(data_list)
    b = len(time_list)

    if a == b:
        print('Successful DownSampling: ' + str(a))
    else:
        print('Something went Wrong: ' + str(a) + '      ' + str(b))

def shorten_timestamps(old_list, new_list):
    for x in range(0, len(old_list)):
        y = old_list[x]
        z = int(str(y)[:10])
        #print(z)

        new_list.append(z)

    return new_list

def read_bvp_data(subj):
```

```
    data_path = 'p_' + str(subj) + '_BVP.csv'
    print('CurrentFile: ' + data_path)

    bvp_e4_raw = hp.get_data(data_path, column_name = 'BVP')
    bvp_e4_adj = get_bvp_timestamp(bvp_e4_raw)

    return bvp_e4_adj

def read_ibi_data(subj):
    data_path = 'p_' + str(subj) + '_IBI.csv'
    print('CurrentFile: ' + data_path)

    ibi_e4_adj = hp.get_data(data_path, column_name = "IBI")

    ibt_e4_adj = hp.get_data(data_path, column_name = "Time")

    return ibi_e4_adj, ibt_e4_adj

def read_hr_data(subj):
    data_path = 'p_' + str(subj) + '_HR.csv'
    print('CurrentFile: ' + data_path)

    hr_e4_raw = hp.get_data(data_path, column_name = "HR")
    hr_e4_adj = hr_e4_raw

    return hr_e4_adj

def create_bvp_time(subj, bvp_e4_adj, timestamps_bvp):
    length = len(bvp_e4_adj)
    act_length = length - 1

    bvp_time = []
    bvp_time.append(timestamps_bvp[subj])

    counter = timestamps_bvp[subj]

    for x in range(1, (act_length + 1)):
        counter += 16
        bvp_time.append(counter)

    if length == len(bvp_time):
        print('BVP_TIME CREATION SUCCESSFUL!')
        print(length)
        print(len(bvp_time))

    return bvp_time

def create_ibi_time(subj, ibi_e4_adj, timestamps_ibi):
    length = len(ibi_e4_adj)

    ibi_time = []
    counter = timestamps_ibi[subj]

    for x in range(0, length):
        y = counter + (ibi_e4_adj[x] * 1000)
        ibi_time.append(y)

    if length == len(ibi_time):
        print('IBI_TIME CREATION SUCCESSFUL!')
        print(length)
        print(len(ibi_time))

    return ibi_time
```

```python

def create_hr_time(subj, hr_e4_adj, timestamps_hr):
    length = len(hr_e4_adj)
    act_length = length - 1

    hr_time = []
    hr_time.append(timestamps_hr[subj])

    counter = timestamps_hr[subj]

    for x in range(1, (act_length + 1)):
        counter += 1000
        hr_time.append(counter)

    if length == len(hr_time):
        print('HR_TIME CREATION SUCCESSFUL!')
        print(length)
        print(len(hr_time))

    return hr_time

def read_s3_data(subj, time_s3_adj, ppg_s3_adj, ibi_s3_adj, hr_s3_adj):
    data_path = 'p_' + str(subj) + '_Shimmer.csv'
    print('CurrentFile: ' + data_path)

    time_s3_raw = hp.get_data(data_path, column_name = 'Shimmer_CD26_Timestamp_Unix_CAL')
    time_s3_adj = get_s3_units(time_s3_raw)
    #print(time_s3_adj)

    ppg_s3_raw = hp.get_data(data_path, column_name = 'Shimmer_CD26_PPG_A13_CAL')
    ppg_s3_adj = get_s3_units(ppg_s3_raw)
    #print(ppg_s3_adj)

    ibi_s3_raw = hp.get_data(data_path, column_name = 'Shimmer_CD26_PPG_IBI_CAL')
    ibi_s3_adj = get_s3_units(ibi_s3_raw)
    #print(ibi_s3_adj)

    hr_s3_raw = hp.get_data(data_path, column_name = 'Shimmer_CD26_PPGtoHR_CAL')
    hr_s3_adj = get_s3_units(hr_s3_raw)
    #print(hr_s3_adj)

    return time_s3_adj, ppg_s3_adj, ibi_s3_adj, hr_s3_adj

def split_up_s3_ppg(subj, tppg_s3_adj, ppg_s3_adj, ppg_s3_bs1, ppg_s3_st1, \
                    ppg_s3_rc1, ppg_s3_bs2, ppg_s3_st2, ppg_s3_rc2, tppg_s3_bs1, \
                    tppg_s3_st1, tppg_s3_rc1, tppg_s3_bs2, tppg_s3_st2, tppg_s3_rc2):
    ppg_s3_length = len(ppg_s3_adj)
    print(ppg_s3_length)

    for y in range(0, ppg_s3_length):
        i = float(tppg_s3_adj[y])

        #BS1
        if i >= float(start_time_bs1[subj]) and i <= float(end_time_bs1[subj]):
            ppg_s3_bs1.append(ppg_s3_adj[y])
            tppg_s3_bs1.append(tppg_s3_adj[y])

        #ST1
        if i >= float(start_time_st1[subj]) and i <= float(end_time_st1[subj]):
            ppg_s3_st1.append(ppg_s3_adj[y])
            tppg_s3_st1.append(tppg_s3_adj[y])

        #RC1
```

```
        if i >= float(start_time_rc1[subj]) and i <= float(end_time_rc1[subj]):
            ppg_s3_rc1.append(ppg_s3_adj[y])
            tppg_s3_rc1.append(tppg_s3_adj[y])

        #BS2
        if i >= float(start_time_bs2[subj]) and i <= float(end_time_bs2[subj]):
            ppg_s3_bs2.append(ppg_s3_adj[y])
            tppg_s3_bs2.append(tppg_s3_adj[y])

        #ST2
        if i >= float(start_time_st2[subj]) and i <= float(end_time_st2[subj]):
            ppg_s3_st2.append(ppg_s3_adj[y])
            tppg_s3_st2.append(tppg_s3_adj[y])

        #RC2
        if i >= float(start_time_rc2[subj]) and i <= float(end_time_rc2[subj]):
            ppg_s3_rc2.append(ppg_s3_adj[y])
            tppg_s3_rc2.append(tppg_s3_adj[y])

    print('SUBJECT: ' + str(subj))
    print('S3_PPG')
    print('TPPG_BS1_Length: ' + str(len(tppg_s3_bs1)))
    print('PPG_BS1_Length: ' + str(len(ppg_s3_bs1)))

    print('TPPG_ST1_Length: ' + str(len(tppg_s3_st1)))
    print('PPG_ST1_Length: ' + str(len(ppg_s3_st1)))

    print('TPPG_RC1_Length: ' + str(len(tppg_s3_rc1)))
    print('PPG_RC1_Length: ' + str(len(ppg_s3_rc1)))

    print('TPPG_BS2_Length: ' + str(len(tppg_s3_bs2)))
    print('PPG_BS2_Length: ' + str(len(ppg_s3_bs2)))

    print('TPPG_ST2_Length: ' + str(len(tppg_s3_st2)))
    print('PPG_ST2_Length: ' + str(len(ppg_s3_st2)))

    print('TPPG_RC2_Length: ' + str(len(tppg_s3_rc2)))
    print('PPG_RC2_Length: ' + str(len(ppg_s3_rc2)))

    return ppg_s3_bs1, ppg_s3_st1, ppg_s3_rc1, ppg_s3_bs2, ppg_s3_st2, ppg_s3_rc2,
    tppg_s3_bs1, tppg_s3_st1, tppg_s3_rc1, tppg_s3_bs2, tppg_s3_st2, tppg_s3_rc2

def split_up_s3_ibi(subj, tibi_s3_adj, ibi_s3_adj, ibi_s3_bs1, ibi_s3_st1, \
                    ibi_s3_rc1, ibi_s3_bs2, ibi_s3_st2, ibi_s3_rc2, tibi_s3_bs1, \
                    tibi_s3_st1, tibi_s3_rc1, tibi_s3_bs2, tibi_s3_st2, tibi_s3_rc2):
    ibi_s3_length = len(ibi_s3_adj)
    print(ibi_s3_length)

    for y in range(0, ibi_s3_length):
        i = float(tibi_s3_adj[y])

        #BS1
        if i >= float(start_time_bs1[subj]) and i <= float(end_time_bs1[subj]):
            ibi_s3_bs1.append(ibi_s3_adj[y])
            tibi_s3_bs1.append(tibi_s3_adj[y])

        #ST1
        if i >= float(start_time_st1[subj]) and i <= float(end_time_st1[subj]):
            ibi_s3_st1.append(ibi_s3_adj[y])
            tibi_s3_st1.append(tibi_s3_adj[y])

        #RC1
        if i >= float(start_time_rc1[subj]) and i <= float(end_time_rc1[subj]):
```

```
            ibi_s3_rc1.append(ibi_s3_adj[y])
            tibi_s3_rc1.append(tibi_s3_adj[y])

        #BS2
        if i >= float(start_time_bs2[subj]) and i <= float(end_time_bs2[subj]):
            ibi_s3_bs2.append(ibi_s3_adj[y])
            tibi_s3_bs2.append(tibi_s3_adj[y])

        #ST2
        if i >= float(start_time_st2[subj]) and i <= float(end_time_st2[subj]):
            ibi_s3_st2.append(ibi_s3_adj[y])
            tibi_s3_st2.append(tibi_s3_adj[y])

        #RC2
        if i >= float(start_time_rc2[subj]) and i <= float(end_time_rc2[subj]):
            ibi_s3_rc2.append(ibi_s3_adj[y])
            tibi_s3_rc2.append(tibi_s3_adj[y])

    print('SUBJECT: ' + str(subj))
    print('S3_IBI')
    print('TIBI_BS1_Length: ' + str(len(tibi_s3_bs1)))
    print('IBI_BS1_Length: ' + str(len(ibi_s3_bs1)))

    print('TIBI_ST1_Length: ' + str(len(tibi_s3_st1)))
    print('IBI_ST1_Length: ' + str(len(ibi_s3_st1)))

    print('TIBI_RC1_Length: ' + str(len(tibi_s3_rc1)))
    print('IBI_RC1_Length: ' + str(len(ibi_s3_rc1)))

    print('TIBI_BS2_Length: ' + str(len(tibi_s3_bs2)))
    print('IBI_BS2_Length: ' + str(len(ibi_s3_bs2)))

    print('TIBI_ST2_Length: ' + str(len(tibi_s3_st2)))
    print('IBI_ST2_Length: ' + str(len(ibi_s3_st2)))

    print('TIBI_RC2_Length: ' + str(len(tibi_s3_rc2)))
    print('IBI_RC2_Length: ' + str(len(ibi_s3_rc2)))

    return ibi_s3_bs1, ibi_s3_st1, ibi_s3_rc1, ibi_s3_bs2, ibi_s3_st2, ibi_s3_rc2,
    tibi_s3_bs1, tibi_s3_st1, tibi_s3_rc1, tibi_s3_bs2, tibi_s3_st2, tibi_s3_rc2

def split_up_s3_hr(subj, thr_s3_adj, hr_s3_adj, hr_s3_bs1, hr_s3_st1, hr_s3_rc1, \
                   hr_s3_bs2, hr_s3_st2, hr_s3_rc2, thr_s3_bs1, thr_s3_st1, \
                   thr_s3_rc1, thr_s3_bs2, thr_s3_st2, thr_s3_rc2):
    hr_s3_length = len(hr_s3_adj)
    print(hr_s3_length)

    for y in range(0, hr_s3_length):
        i = float(thr_s3_adj[y])

        #BS1
        if i >= float(start_time_bs1[subj]) and i <= float(end_time_bs1[subj]):
            hr_s3_bs1.append(hr_s3_adj[y])
            thr_s3_bs1.append(thr_s3_adj[y])

        #ST1
        if i >= float(start_time_st1[subj]) and i <= float(end_time_st1[subj]):
            hr_s3_st1.append(hr_s3_adj[y])
            thr_s3_st1.append(thr_s3_adj[y])

        #RC1
        if i >= float(start_time_rc1[subj]) and i <= float(end_time_rc1[subj]):
            hr_s3_rc1.append(hr_s3_adj[y])
```

```
                thr_s3_rc1.append(thr_s3_adj[y])

            #BS2
            if i >= float(start_time_bs2[subj]) and i <= float(end_time_bs2[subj]):
                hr_s3_bs2.append(hr_s3_adj[y])
                thr_s3_bs2.append(thr_s3_adj[y])

            #ST2
            if i >= float(start_time_st2[subj]) and i <= float(end_time_st2[subj]):
                hr_s3_st2.append(hr_s3_adj[y])
                thr_s3_st2.append(thr_s3_adj[y])

            #RC2
            if i >= float(start_time_rc2[subj]) and i <= float(end_time_rc2[subj]):
                hr_s3_rc2.append(hr_s3_adj[y])
                thr_s3_rc2.append(thr_s3_adj[y])

    print('SUBJECT: ' + str(subj))
    print('S3_HR')
    print('THR_BS1_Length: ' + str(len(thr_s3_bs1)))
    print('HR_BS1_Length: ' + str(len(hr_s3_bs1)))

    print('THR_ST1_Length: ' + str(len(thr_s3_st1)))
    print('HR_ST1_Length: ' + str(len(hr_s3_st1)))

    print('THR_RC1_Length: ' + str(len(thr_s3_rc1)))
    print('HR_RC1_Length: ' + str(len(hr_s3_rc1)))

    print('THR_BS2_Length: ' + str(len(thr_s3_bs2)))
    print('HR_BS2_Length: ' + str(len(hr_s3_bs2)))

    print('THR_ST2_Length: ' + str(len(thr_s3_st2)))
    print('HR_ST2_Length: ' + str(len(hr_s3_st2)))

    print('THR_RC2_Length: ' + str(len(thr_s3_rc2)))
    print('HR_RC2_Length: ' + str(len(hr_s3_rc2)))

    return hr_s3_bs1, hr_s3_st1, hr_s3_rc1, hr_s3_bs2, hr_s3_st2, hr_s3_rc2, thr_s3_bs1,
    thr_s3_st1, thr_s3_rc1, thr_s3_bs2, thr_s3_st2, thr_s3_rc2

def split_up_e4_bvp(subj, bvp_e4_adj, time_e4_bvp, bvp_e4_bs1, bvp_e4_st1, \
                    bvp_e4_rc1, bvp_e4_bs2, bvp_e4_st2, bvp_e4_rc2, tbvp_e4_bs1, \
                    tbvp_e4_st1, tbvp_e4_rc1, tbvp_e4_bs2, tbvp_e4_st2, tbvp_e4_rc2):
    print('SPLITTING OF E4 BVP')

    bvp_e4_length = len(bvp_e4_adj)
    tbvp_e4_length = len(time_e4_bvp)
    print(bvp_e4_length)
    print(tbvp_e4_length)

    for y in range(0, bvp_e4_length):
        i = float(time_e4_bvp[y])

        #BS1
        if i >= float(start_time_bs1[subj]) and i <= float(end_time_bs1[subj]):
            bvp_e4_bs1.append(bvp_e4_adj[y])
            tbvp_e4_bs1.append(time_e4_bvp[y])

        #ST1
        if i >= float(start_time_st1[subj]) and i <= float(end_time_st1[subj]):
            bvp_e4_st1.append(bvp_e4_adj[y])
            tbvp_e4_st1.append(time_e4_bvp[y])

```

```
        #RC1
        if i >= float(start_time_rc1[subj]) and i <= float(end_time_rc1[subj]):
            bvp_e4_rc1.append(bvp_e4_adj[y])
            tbvp_e4_rc1.append(time_e4_bvp[y])

        #BS2
        if i >= float(start_time_bs2[subj]) and i <= float(end_time_bs2[subj]):
            bvp_e4_bs2.append(bvp_e4_adj[y])
            tbvp_e4_bs2.append(time_e4_bvp[y])

        #ST2
        if i >= float(start_time_st2[subj]) and i <= float(end_time_st2[subj]):
            bvp_e4_st2.append(bvp_e4_adj[y])
            tbvp_e4_st2.append(time_e4_bvp[y])

        #RC2
        if i >= float(start_time_rc2[subj]) and i <= float(end_time_rc2[subj]):
            bvp_e4_rc2.append(bvp_e4_adj[y])
            tbvp_e4_rc2.append(time_e4_bvp[y])

    print('SUBJECT: ' + str(subj))
    print('E4_BVP')
    print('TIME_BS1_Length: ' + str(len(tbvp_e4_bs1)))
    print('BVP_BS1_Length: ' + str(len(bvp_e4_bs1)))

    print('TIME_ST1_Length: ' + str(len(tbvp_e4_st1)))
    print('BVP_ST1_Length: ' + str(len(bvp_e4_st1)))

    print('TIME_RC1_Length: ' + str(len(tbvp_e4_rc1)))
    print('BVP_RC1_Length: ' + str(len(bvp_e4_rc1)))

    print('TIME_BS2_Length: ' + str(len(tbvp_e4_bs2)))
    print('BVP_BS2_Length: ' + str(len(bvp_e4_bs2)))

    print('TIME_ST2_Length: ' + str(len(tbvp_e4_st2)))
    print('BVP_ST2_Length: ' + str(len(bvp_e4_st2)))

    print('TIME_RC2_Length: ' + str(len(tbvp_e4_rc2)))
    print('BVP_RC2_Length: ' + str(len(bvp_e4_rc2)))

    return bvp_e4_bs1, bvp_e4_st1, bvp_e4_rc1, bvp_e4_bs2, bvp_e4_st2, bvp_e4_rc2,
    tbvp_e4_bs1, tbvp_e4_st1, tbvp_e4_rc1, tbvp_e4_bs2, tbvp_e4_st2, tbvp_e4_rc2

def split_up_e4_ibi(subj, ibi_e4_adj, time_e4_ibi, ibi_e4_bs1, ibi_e4_st1, \
                    ibi_e4_rc1, ibi_e4_bs2, ibi_e4_st2, ibi_e4_rc2, tibi_e4_bs1, \
                        tibi_e4_st1, tibi_e4_rc1, tibi_e4_bs2, tibi_e4_st2, tibi_e4_rc2):
    ibi_e4_length = len(ibi_e4_adj)
    print(ibi_e4_length)

    for y in range(0, ibi_e4_length):
        i = float(time_e4_ibi[y])

        #BS1
        if i >= float(start_time_bs1[subj]) and i <= float(end_time_bs1[subj]):
            ibi_e4_bs1.append(ibi_e4_adj[y])
            tibi_e4_bs1.append(time_e4_ibi[y])

        #ST1
        if i >= float(start_time_st1[subj]) and i <= float(end_time_st1[subj]):
            ibi_e4_st1.append(ibi_e4_adj[y])
            tibi_e4_st1.append(time_e4_ibi[y])

        #RC1
```

```
        if i >= float(start_time_rc1[subj]) and i <= float(end_time_rc1[subj]):
            ibi_e4_rc1.append(ibi_e4_adj[y])
            tibi_e4_rc1.append(time_e4_ibi[y])

        #BS2
        if i >= float(start_time_bs2[subj]) and i <= float(end_time_bs2[subj]):
            ibi_e4_bs2.append(ibi_e4_adj[y])
            tibi_e4_bs2.append(time_e4_ibi[y])

        #ST2
        if i >= float(start_time_st2[subj]) and i <= float(end_time_st2[subj]):
            ibi_e4_st2.append(ibi_e4_adj[y])
            tibi_e4_st2.append(time_e4_ibi[y])

        #RC2
        if i >= float(start_time_rc2[subj]) and i <= float(end_time_rc2[subj]):
            ibi_e4_rc2.append(ibi_e4_adj[y])
            tibi_e4_rc2.append(time_e4_ibi[y])

    print('SUBJECT: ' + str(subj))
    print('E4_IBI')
    print('TIME_BS1_Length: ' + str(len(tibi_e4_bs1)))
    print('IBI_BS1_Length: ' + str(len(ibi_e4_bs1)))

    print('TIME_ST1_Length: ' + str(len(tibi_e4_st1)))
    print('IBI_ST1_Length: ' + str(len(ibi_e4_st1)))

    print('TIME_RC1_Length: ' + str(len(tibi_e4_rc1)))
    print('IBI_RC1_Length: ' + str(len(ibi_e4_rc1)))

    print('TIME_BS2_Length: ' + str(len(tibi_e4_bs2)))
    print('IBI_BS2_Length: ' + str(len(ibi_e4_bs2)))

    print('TIME_ST2_Length: ' + str(len(tibi_e4_st2)))
    print('IBI_ST2_Length: ' + str(len(ibi_e4_st2)))

    print('TIME_RC2_Length: ' + str(len(tibi_e4_rc2)))
    print('IBI_RC2_Length: ' + str(len(ibi_e4_rc2)))

    return ibi_e4_bs1, ibi_e4_st1, ibi_e4_rc1, ibi_e4_bs2, ibi_e4_st2, ibi_e4_rc2,
    tibi_e4_bs1, tibi_e4_st1, tibi_e4_rc1, tibi_e4_bs2, tibi_e4_st2, tibi_e4_rc2

def split_up_e4_hr(subj, hr_e4_adj, time_e4_hr, hr_e4_bs1, hr_e4_st1, hr_e4_rc1, \
                   hr_e4_bs2, hr_e4_st2, hr_e4_rc2, thr_e4_bs1, thr_e4_st1, \
                   thr_e4_rc1, thr_e4_bs2, thr_e4_st2, thr_e4_rc2):
    hr_e4_length = len(hr_e4_adj)
    print(hr_e4_length)

    for y in range(0, hr_e4_length):
        i = float(time_e4_hr[y])

        #BS1
        if i >= float(start_time_bs1[subj]) and i <= float(end_time_bs1[subj]):
            hr_e4_bs1.append(hr_e4_adj[y])
            thr_e4_bs1.append(time_e4_hr[y])

        #ST1
        if i >= float(start_time_st1[subj]) and i <= float(end_time_st1[subj]):
            hr_e4_st1.append(hr_e4_adj[y])
            thr_e4_st1.append(time_e4_hr[y])

        #RC1
        if i >= float(start_time_rc1[subj]) and i <= float(end_time_rc1[subj]):
```

```
            hr_e4_rc1.append(hr_e4_adj[y])
            thr_e4_rc1.append(time_e4_hr[y])

        #BS2
        if i >= float(start_time_bs2[subj]) and i <= float(end_time_bs2[subj]):
            hr_e4_bs2.append(hr_e4_adj[y])
            thr_e4_bs2.append(time_e4_hr[y])

        #ST2
        if i >= float(start_time_st2[subj]) and i <= float(end_time_st2[subj]):
            hr_e4_st2.append(hr_e4_adj[y])
            thr_e4_st2.append(time_e4_hr[y])

        #RC2
        if i >= float(start_time_rc2[subj]) and i <= float(end_time_rc2[subj]):
            hr_e4_rc2.append(hr_e4_adj[y])
            thr_e4_rc2.append(time_e4_hr[y])

    print('SUBJECT: ' + str(subj))
    print('E4_HR')
    print('TIME_BS1_Length: ' + str(len(thr_e4_bs1)))
    print('HR_BS1_Length: ' + str(len(hr_e4_bs1)))

    print('TIME_ST1_Length: ' + str(len(thr_e4_st1)))
    print('HR_ST1_Length: ' + str(len(hr_e4_st1)))

    print('TIME_RC1_Length: ' + str(len(thr_e4_rc1)))
    print('HR_RC1_Length: ' + str(len(hr_e4_rc1)))

    print('TIME_BS2_Length: ' + str(len(thr_e4_bs2)))
    print('HR_BS2_Length: ' + str(len(hr_e4_bs2)))

    print('TIME_ST2_Length: ' + str(len(thr_e4_st2)))
    print('HR_ST2_Length: ' + str(len(hr_e4_st2)))

    print('TIME_RC2_Length: ' + str(len(thr_e4_rc2)))
    print('HR_RC2_Length: ' + str(len(hr_e4_rc2)))

    return hr_e4_bs1, hr_e4_st1, hr_e4_rc1, hr_e4_bs2, hr_e4_st2, hr_e4_rc2, thr_e4_bs1,
    thr_e4_st1, thr_e4_rc1, thr_e4_bs2, thr_e4_st2, thr_e4_rc2

def cut_in_shape(subj, time_s3_adj, ppg_s3_adj, ibi_s3_adj, hr_s3_adj, bvp_e4_adj, \
                 ibi_e4_adj, hr_e4_adj, time_e4_bvp, time_e4_ibi, time_e4_hr):
    tppg_s3 = []
    tibi_s3 = []
    thr_s3  = []
    ppg_s3  = []
    ibi_s3  = []
    hr_s3   = []

    s3_length = len(time_s3_adj)
    print(s3_length)

    for y in range(0, s3_length):
        i = float(time_s3_adj[y])

        if i >= float(start_time_bs1[subj]) and i <= float(end_time_rc2[subj]):
            tppg_s3.append(time_s3_adj[y])
            tibi_s3.append(time_s3_adj[y])
            thr_s3.append(time_s3_adj[y])

            ppg_s3.append(ppg_s3_adj[y])
            ibi_s3.append(ibi_s3_adj[y])
```

```
            hr_s3.append(hr_s3_adj[y])

    bvp_e4  = []
    ibi_e4  = []
    hr_e4   = []
    tbvp_e4 = []
    tibi_e4 = []
    thr_e4  = []

    e4_bvp_length = len(bvp_e4_adj)
    print('BVP LENGTH: ' + str(e4_bvp_length))

    g_rest = []
    g_trest = []
    g_pos = []
    k_rest = []
    k_trest = []
    k_pos = []

    for y in range(0, e4_bvp_length):
        i = float(time_e4_bvp[y])

        if i >= float(start_time_bs1[subj]) and i <= float(end_time_rc2[subj]):
            tbvp_e4.append(time_e4_bvp[y])
            bvp_e4.append(bvp_e4_adj[y])
        if i < float(start_time_bs1[subj]):
            k_rest.append(bvp_e4_adj[y])
            k_trest.append(time_e4_bvp[y])
            k_pos.append(y)
        if i > float(end_time_rc2[subj]):
            g_rest.append(bvp_e4_adj[y])
            g_trest.append(time_e4_bvp[y])
            g_pos.append(y)

    print('G_REST: ' + str(len(g_rest)))
    print('G_TREST: ' + str(len(g_trest)))
    #print(g_pos)

    print('K_REST: ' + str(len(k_rest)))
    print('K_TREST: ' + str(len(k_trest)))
    #print(k_pos)


    e4_ibi_length = len(ibi_e4_adj)
    print(e4_ibi_length)

    for y in range(0, e4_ibi_length):
        i = float(time_e4_ibi[y])

        if i >= float(start_time_bs1[subj]) and i <= float(end_time_rc2[subj]):
            tibi_e4.append(time_e4_ibi[y])
            ibi_e4.append(ibi_e4_adj[y])

    e4_hr_length = len(hr_e4_adj)
    print(e4_hr_length)

    for y in range(0, e4_hr_length):
        i = float(time_e4_hr[y])

        if i >= float(start_time_bs1[subj]) and i <= float(end_time_rc2[subj]):
            thr_e4.append(time_e4_hr[y])
            hr_e4.append(hr_e4_adj[y])

```

```
    return tppg_s3, tibi_s3, thr_s3, ppg_s3, ibi_s3, hr_s3, bvp_e4, ibi_e4, hr_e4, tbvp_e4,
    tibi_e4, thr_e4

def downsampling_ppg(subj, tppg_s3_bs1, tppg_s3_st1, tppg_s3_rc1, tppg_s3_bs2, \
                     tppg_s3_st2, tppg_s3_rc2, bvp_e4_bs1, bvp_e4_st1, bvp_e4_rc1, \
                     bvp_e4_bs2, bvp_e4_st2, bvp_e4_rc2, ppg_s3_bs1, ppg_s3_st1, \
                     ppg_s3_rc1, ppg_s3_bs2, ppg_s3_st2, ppg_s3_rc2):
    bs1_length = len(bvp_e4_bs1)
    st1_length = len(bvp_e4_st1)
    rc1_length = len(bvp_e4_rc1)
    bs2_length = len(bvp_e4_bs2)
    st2_length = len(bvp_e4_st2)
    rc2_length = len(bvp_e4_rc2)

    bs1 = ppg_s3_bs1
    st1 = ppg_s3_st1
    rc1 = ppg_s3_rc1
    bs2 = ppg_s3_bs2
    st2 = ppg_s3_st2
    rc2 = ppg_s3_rc2

    tbs1 = np.asarray(tppg_s3_bs1, dtype = np.float64)
    tst1 = np.asarray(tppg_s3_st1, dtype = np.float64)
    trc1 = np.asarray(tppg_s3_rc1, dtype = np.float64)
    tbs2 = np.asarray(tppg_s3_bs2, dtype = np.float64)
    tst2 = np.asarray(tppg_s3_st2, dtype = np.float64)
    trc2 = np.asarray(tppg_s3_rc2, dtype = np.float64)

    ppg_s3_bs1 = sp.signal.resample(bs1, bs1_length, t = tbs1)
    ppg_s3_st1 = sp.signal.resample(st1, st1_length, t = tst1)
    ppg_s3_rc1 = sp.signal.resample(rc1, rc1_length, t = trc1)
    ppg_s3_bs2 = sp.signal.resample(bs2, bs2_length, t = tbs2)
    ppg_s3_st2 = sp.signal.resample(st2, st2_length, t = tst2)
    ppg_s3_rc2 = sp.signal.resample(rc2, rc2_length, t = trc2)

    print('SUBJECT: ' + str(subj))
    print('PPG_S3')
    downsampling_check(ppg_s3_bs1[0], bvp_e4_bs1)
    downsampling_check(ppg_s3_st1[0], bvp_e4_st1)
    downsampling_check(ppg_s3_rc1[0], bvp_e4_rc1)
    downsampling_check(ppg_s3_bs2[0], bvp_e4_bs2)
    downsampling_check(ppg_s3_st2[0], bvp_e4_st2)
    downsampling_check(ppg_s3_rc2[0], bvp_e4_rc2)
    print('TPPG_S3')
    downsampling_check(ppg_s3_bs1[1], bvp_e4_bs1)
    downsampling_check(ppg_s3_st1[1], bvp_e4_st1)
    downsampling_check(ppg_s3_rc1[1], bvp_e4_rc1)
    downsampling_check(ppg_s3_bs2[1], bvp_e4_bs2)
    downsampling_check(ppg_s3_st2[1], bvp_e4_st2)
    downsampling_check(ppg_s3_rc2[1], bvp_e4_rc2)

    return ppg_s3_bs1, ppg_s3_st1, ppg_s3_rc1, ppg_s3_bs2, ppg_s3_st2, ppg_s3_rc2

def downsampling_ibi(subj, zero_ibi_bs1, zero_ibi_st1, zero_ibi_rc1, zero_ibi_bs2, \
                     zero_ibi_st2, zero_ibi_rc2, tibi_s3_bs1, tibi_s3_st1, \
                     tibi_s3_rc1, tibi_s3_bs2, tibi_s3_st2, tibi_s3_rc2, \
                     ibi_e4_bs1, ibi_e4_st1, ibi_e4_rc1, ibi_e4_bs2, ibi_e4_st2, \
                     ibi_e4_rc2, ibi_s3_bs1, ibi_s3_st1, ibi_s3_rc1, ibi_s3_bs2, \
                     ibi_s3_st2, ibi_s3_rc2):
    bs1_length = len(ibi_e4_bs1)
    st1_length = len(ibi_e4_st1)
    rc1_length = len(ibi_e4_rc1)
    bs2_length = len(ibi_e4_bs2)
```

```
    st2_length = len(ibi_e4_st2)
    rc2_length = len(ibi_e4_rc2)

    tbs1 = np.asarray(tibi_s3_bs1, dtype = np.float64)
    tst1 = np.asarray(tibi_s3_st1, dtype = np.float64)
    trc1 = np.asarray(tibi_s3_rc1, dtype = np.float64)
    tbs2 = np.asarray(tibi_s3_bs2, dtype = np.float64)
    tst2 = np.asarray(tibi_s3_st2, dtype = np.float64)
    trc2 = np.asarray(tibi_s3_rc2, dtype = np.float64)

    print('SUBJECT: ' + str(subj))
    print('IBI_S3')

    if bs1_length <= 1:
        zero_ibi_bs1 = True
    else:
        bs1 = ibi_s3_bs1
        ibi_s3_bs1 = sp.signal.resample(bs1, bs1_length, t = tbs1)
        downsampling_check(ibi_s3_bs1[0], ibi_e4_bs1)
        downsampling_check(ibi_s3_bs1[1], ibi_e4_bs1)

    if st1_length <= 1:
        zero_ibi_st1 = True
    else:
        st1 = ibi_s3_st1
        ibi_s3_st1 = sp.signal.resample(st1, st1_length, t = tst1)
        downsampling_check(ibi_s3_st1[0], ibi_e4_st1)
        downsampling_check(ibi_s3_st1[1], ibi_e4_st1)

    if rc1_length <= 1:
        zero_ibi_rc1 = True
    else:
        rc1 = ibi_s3_rc1
        ibi_s3_rc1 = sp.signal.resample(rc1, rc1_length, t = trc1)
        downsampling_check(ibi_s3_rc1[0], ibi_e4_rc1)
        downsampling_check(ibi_s3_rc1[1], ibi_e4_rc1)

    if bs2_length <= 1:
        zero_ibi_bs2 = True
    else:
        bs2 = ibi_s3_bs2
        ibi_s3_bs2 = sp.signal.resample(bs2, bs2_length, t = tbs2)
        downsampling_check(ibi_s3_bs2[0], ibi_e4_bs2)
        downsampling_check(ibi_s3_bs2[1], ibi_e4_bs2)

    if st2_length <= 1:
        zero_ibi_st2 = True
    else:
        st2 = ibi_s3_st2
        ibi_s3_st2 = sp.signal.resample(st2, st2_length, t = tst2)
        downsampling_check(ibi_s3_st2[0], ibi_e4_st2)
        downsampling_check(ibi_s3_st2[1], ibi_e4_st2)

    if rc2_length <= 1:
        zero_ibi_rc2 = True
    else:
        rc2 = ibi_s3_rc2
        ibi_s3_rc2 = sp.signal.resample(rc2, rc2_length, t = trc2)
        downsampling_check(ibi_s3_rc2[0], ibi_e4_rc2)
        downsampling_check(ibi_s3_rc2[1], ibi_e4_rc2)

    return ibi_s3_bs1, ibi_s3_st1, ibi_s3_rc1, ibi_s3_bs2, ibi_s3_st2, ibi_s3_rc2,
    zero_ibi_bs1, zero_ibi_st1, zero_ibi_rc1, zero_ibi_bs2, zero_ibi_st2, zero_ibi_rc2
```

```
def downsampling_hr(subj, zero_hr_bs1, zero_hr_st1, zero_hr_rc1, zero_hr_bs2, \
                    zero_hr_st2, zero_hr_rc2, thr_s3_bs1, thr_s3_st1, thr_s3_rc1, \
                    thr_s3_bs2, thr_s3_st2, thr_s3_rc2, hr_e4_bs1, hr_e4_st1, \
                    hr_e4_rc1, hr_e4_bs2, hr_e4_st2, hr_e4_rc2, hr_s3_bs1, \
                    hr_s3_st1, hr_s3_rc1, hr_s3_bs2, hr_s3_st2, hr_s3_rc2):
    bs1_length = len(hr_e4_bs1)
    st1_length = len(hr_e4_st1)
    rc1_length = len(hr_e4_rc1)
    bs2_length = len(hr_e4_bs2)
    st2_length = len(hr_e4_st2)
    rc2_length = len(hr_e4_rc2)

    tbs1 = np.asarray(thr_s3_bs1, dtype = np.float64)
    tst1 = np.asarray(thr_s3_st1, dtype = np.float64)
    trc1 = np.asarray(thr_s3_rc1, dtype = np.float64)
    tbs2 = np.asarray(thr_s3_bs2, dtype = np.float64)
    tst2 = np.asarray(thr_s3_st2, dtype = np.float64)
    trc2 = np.asarray(thr_s3_rc2, dtype = np.float64)

    print('SUBJECT: ' + str(subj))
    print('HR_S3')

    if bs1_length <= 1:
        zero_hr_bs1 = True
    else:
        bs1 = hr_s3_bs1
        hr_s3_bs1 = sp.signal.resample(bs1, bs1_length, t = tbs1)
        downsampling_check(hr_s3_bs1[0], hr_e4_bs1)
        downsampling_check(hr_s3_bs1[1], hr_e4_bs1)

    if st1_length <= 1:
        zero_hr_st1 = True
    else:
        st1 = hr_s3_st1
        hr_s3_st1 = sp.signal.resample(st1, st1_length, t = tst1)
        downsampling_check(hr_s3_st1[0], hr_e4_st1)
        downsampling_check(hr_s3_st1[1], hr_e4_st1)

    if rc1_length <= 1:
        zero_hr_rc1 = True
    else:
        rc1 = hr_s3_rc1
        hr_s3_rc1 = sp.signal.resample(rc1, rc1_length, t = trc1)
        downsampling_check(hr_s3_rc1[0], hr_e4_rc1)
        downsampling_check(hr_s3_rc1[1], hr_e4_rc1)

    if bs2_length <= 1:
        zero_hr_bs2 = True
    else:
        bs2 = hr_s3_bs2
        hr_s3_bs2 = sp.signal.resample(bs2, bs2_length, t = tbs2)
        downsampling_check(hr_s3_bs2[0], hr_e4_bs2)
        downsampling_check(hr_s3_bs2[1], hr_e4_bs2)

    if st2_length <= 1:
        zero_hr_st2 = True
    else:
        st2 = hr_s3_st2
        hr_s3_st2 = sp.signal.resample(st2, st2_length, t = tst2)
        downsampling_check(hr_s3_st2[0], hr_e4_st2)
        downsampling_check(hr_s3_st2[1], hr_e4_st2)

```

```
    if rc2_length <= 1:
        zero_hr_rc2 = True
    else:
        rc2 = hr_s3_rc2
        hr_s3_rc2 = sp.signal.resample(rc2, rc2_length, t = trc2)
        downsampling_check(hr_s3_rc2[0], hr_e4_rc2)
        downsampling_check(hr_s3_rc2[1], hr_e4_rc2)

    return hr_s3_bs1, hr_s3_st1, hr_s3_rc1, hr_s3_bs2, hr_s3_st2, hr_s3_rc2, zero_hr_bs1, zero_hr_st1, zero_hr_rc1, zero_hr_bs2, zero_hr_st2, zero_hr_rc2

def create_plot(x_time, y_data, subj, cond):
    plt.plot(x_time, y_data)

    title = 'p_' + str(subj) + '_' + cond

    plt.title(title)
    plt.show()

def create_file(subj, mode, data, tdata):
    length = len(data)
    subject = []

    for x in range(0, length):
        subject.append(subj)

    rows = zip(subject, tdata, data)

    data_path = mode + '_p' + str(subj) + '.csv'

    with open(data_path, "w") as f:
        writer = csv.writer(f)
        writer.writerow(['SUBJECT', 'TIME', mode])

        for row in rows:
            writer.writerow(row)

    print('Successful FileCreation: ' + mode)

def create_corr_file(mode, data, tdata):
    rows = zip(tdata, data)

    data_path = mode + '.csv'

    with open(data_path, "w") as f:
        writer = csv.writer(f)
        writer.writerow(['MODE', mode])

        for row in rows:
            writer.writerow(row)

    print('Successful FileCreation: ' + mode)

def np_cross_correlation(subj, mode, xdata, ydata):
    title = 'p_' + str(subj) + '_' + mode

    corr = np.corrcoef(xdata, ydata)

    corr_coef.append(corr[0][1])
    corr_mode.append(title)

    print(title)
    print(corr[0][1])
```

```
def get_hr_s3_parameter(subj, mode, ppg_s3):
    tppg = np.asarray(ppg_s3[1], dtype = np.float64)

    fs = hp.get_samplerate_mstimer(tppg)

    working_data, measures = hp.process(ppg_s3[0], fs, calc_freq = False)

    modes.append(mode)
    bpm.append(measures['bpm'])
    breathingrate.append(measures['breathingrate'])
    #hf.append(measures['hf'])
    hr_mad.append(measures['hr_mad'])
    ibi.append(measures['ibi'])
    #lf.append(measures['lf'])
    #lf_hf.append(measures['lf/hf'])
    pnn20.append(measures['pnn20'])
    pnn50.append(measures['pnn50'])
    rmssd.append(measures['rmssd'])
    s.append(measures['s'])
    sd1.append(measures['sd1'])
    sd1_sd2.append(measures['sd1/sd2'])
    sd2.append(measures['sd2'])
    sdnn.append(measures['sdnn'])
    sdsd.append(measures['sdsd'])

    title = 'Heart Rate Signal Peak Detection ' + mode + '_p' + str(subj)

    plt.figure(figsize=(12,4))

    plot_object = hp.plotter(working_data, measures, show = False, title = title)

    save = mode + '_p' + str(subj) + ' Heart Rate Signal Peak Detection.jpg'

    plot_object.savefig(save)
    plot_object.show()

    print('Successful: p_' + str(subj) + '_' + mode)

def get_hr_e4_parameter(subj, mode, bvp_e4, tbvp_e4):
    tbvp = np.asarray(tbvp_e4, dtype = np.float64)
    bvp  = np.asarray(bvp_e4, dtype = np.float64)

    fs = hp.get_samplerate_mstimer(tbvp)

    working_data, measures = hp.process(bvp, fs, calc_freq = False)

    modes.append(mode)
    bpm.append(measures['bpm'])
    breathingrate.append(measures['breathingrate'])
    #hf.append(measures['hf'])
    hr_mad.append(measures['hr_mad'])
    ibi.append(measures['ibi'])
    #lf.append(measures['lf'])
    #lf_hf.append(measures['lf/hf'])
    pnn20.append(measures['pnn20'])
    pnn50.append(measures['pnn50'])
    rmssd.append(measures['rmssd'])
    s.append(measures['s'])
    sd1.append(measures['sd1'])
    sd1_sd2.append(measures['sd1/sd2'])
    sd2.append(measures['sd2'])
    sdnn.append(measures['sdnn'])
```

```
    sdsd.append(measures['sdsd'])

    title = 'Heart Rate Signal Peak Detection ' + mode + '_p' + str(subj)

    plt.figure(figsize=(12,4))

    plot_object = hp.plotter(working_data, measures, show = False, title = title)

    save = mode + '_p' + str(subj) + ' Heart Rate Signal Peak Detection.jpg'

    plot_object.savefig(save)
    plot_object.show()

    print('Successful: p_' + str(subj) + '_' + mode)

def save_hr_parameters(subj, modes, bpm, breathingrate, hr_mad, ibi, pnn20, \
                       pnn50, rmssd, s, sd1, sd1_sd2, sd2, sdnn, sdsd):
    rows = zip(modes, bpm, breathingrate, hr_mad, ibi, pnn20, pnn50, rmssd, s, \
               sd1, sd1_sd2, sd2, sdnn, sdsd)

    print(len(bpm))
    print(len(breathingrate))
    print(len(hr_mad))
    print(len(ibi))
    print(len(pnn20))
    print(len(pnn50))
    print(len(rmssd))

    data_path = 'PPG_p_' + str(subj) + '_Hear_Rate_Parameters.csv'

    with open(data_path, "w") as f:
        writer = csv.writer(f)

        for row in rows:
            writer.writerow(row)

    print('Successful FileCreation: ' + str(data_path))

def read_hr_parameters(subj, bpm_e4, bpm_s3, ibi_e4, ibi_s3, rmssd_e4, rmssd_s3, \
                       sdnn_e4, sdnn_s3):
    data_path = 'PPG_p_' + str(subj) + '_Hear_Rate_Parameters.csv'
    print('CurrentFile: ' + data_path)

    subjects.append(subj)

    bpm_raw = hp.get_data(data_path, column_name = 'Bpm')
    bpm_s3_mean = (bpm_raw[0] + bpm_raw[1] + bpm_raw[2] + \
                   bpm_raw[3] + bpm_raw[4] + bpm_raw[5]) / 6
    bpm_e4_mean = (bpm_raw[6] + bpm_raw[7] + bpm_raw[8] + \
                   bpm_raw[9] + bpm_raw[10] + bpm_raw[11]) / 6

    bpm_s3.append(bpm_s3_mean)
    bpm_e4.append(bpm_e4_mean)

    ibi_raw = hp.get_data(data_path, column_name = 'Ibi')
    ibi_s3_mean = (ibi_raw[0] + ibi_raw[1] + ibi_raw[2] + \
                   ibi_raw[3] + ibi_raw[4] + ibi_raw[5]) / 6
    ibi_e4_mean = (ibi_raw[6] + ibi_raw[7] + ibi_raw[8] + \
                   ibi_raw[9] + ibi_raw[10] + ibi_raw[11]) / 6

    ibi_s3.append(ibi_s3_mean)
    ibi_e4.append(ibi_e4_mean)
```

```
    rmssd_raw = hp.get_data(data_path, column_name = 'Rmssd')
    rmssd_s3_mean = (rmssd_raw[0] + rmssd_raw[1] + rmssd_raw[2] + \
                     rmssd_raw[3] + rmssd_raw[4] + rmssd_raw[5]) / 6
    rmssd_e4_mean = (rmssd_raw[6] + rmssd_raw[7] + rmssd_raw[8] + \
                     rmssd_raw[9] + rmssd_raw[10] + rmssd_raw[11]) / 6

    rmssd_s3.append(rmssd_s3_mean)
    rmssd_e4.append(rmssd_e4_mean)

    sdnn_raw = hp.get_data(data_path, column_name = 'Sdnn')
    sdnn_s3_mean = (sdnn_raw[0] + sdnn_raw[1] + sdnn_raw[2] + \
                    sdnn_raw[3] + sdnn_raw[4] + sdnn_raw[5]) / 6
    sdnn_e4_mean = (sdnn_raw[6] + sdnn_raw[7] + sdnn_raw[8] + \
                    sdnn_raw[9] + sdnn_raw[10] + sdnn_raw[11]) / 6

    sdnn_s3.append(sdnn_s3_mean)
    sdnn_e4.append(sdnn_e4_mean)

    return bpm_e4, bpm_s3, ibi_e4, ibi_s3, rmssd_e4, rmssd_s3, sdnn_e4, sdnn_s3

def create_parameter_file(subjects, bpm_e4, bpm_s3, ibi_e4, ibi_s3, rmssd_e4, \
                          rmssd_s3, sdnn_e4, sdnn_s3):
    rows = zip(subjects, bpm_e4, bpm_s3, ibi_e4, ibi_s3, rmssd_e4, rmssd_s3, \
               sdnn_e4, sdnn_s3)

    data_path = 'Summary_HR_Parameter.csv'

    with open(data_path, "w") as f:
        writer = csv.writer(f)
        writer.writerow(['Subject', 'bpm_e4', 'bpm_s3', 'ibi_e4', 'ibi_s3', 'rmssd_e4', 'rmssd_s3', 'sdnn_e4', 'sdnn_s3'])

        for row in rows:
            writer.writerow(row)

    print('Successful FileCreation: ' + str(data_path))

#MAIN PART
subj = 19

for subj in range(1, (subj + 1)):
    #IMPORT E4 DATA
    bvp_e4_adj = read_bvp_data(subj)

    ibi_e4_adj, ibt_e4_adj = read_ibi_data(subj)

    hr_e4_adj = read_hr_data(subj)


    #CREATE TIME LISTS FOR E4
    time_e4_bvp = create_bvp_time(subj, bvp_e4_adj, timestamps_bvp)

    time_e4_ibi = create_ibi_time(subj, ibt_e4_adj, timestamps_ibi)

    time_e4_hr = create_hr_time(subj, hr_e4_adj, timestamps_hr)


    #GET BVP LENGTH
    get_bvp_length(subj)


    #IMPORT S3 DATA
    time_s3_adj, ppg_s3_adj, ibi_s3_adj, hr_s3_adj = read_s3_data(subj, time_s3_adj,
```

```
ppg_s3_adj, ibi_s3_adj, hr_s3_adj)


#CUT RECORDINGS IN SHAPE
tppg_s3_adj, tibi_s3_adj, thr_s3_adj, ppg_s3_adj, ibi_s3_adj, hr_s3_adj, bvp_e4_adj,
ibi_e4_adj, hr_e4_adj, time_e4_bvp, time_e4_ibi, time_e4_hr = cut_in_shape(subj,
time_s3_adj, ppg_s3_adj, ibi_s3_adj, hr_s3_adj, bvp_e4_adj, ibi_e4_adj, hr_e4_adj,
time_e4_bvp, time_e4_ibi, time_e4_hr)


#SPLIT UP E4 DATA
bvp_e4_bs1, bvp_e4_st1, bvp_e4_rc1, bvp_e4_bs2, bvp_e4_st2, bvp_e4_rc2, tbvp_e4_bs1,
tbvp_e4_st1, tbvp_e4_rc1, tbvp_e4_bs2, tbvp_e4_st2, tbvp_e4_rc2 = split_up_e4_bvp(subj,
bvp_e4_adj, time_e4_bvp, bvp_e4_bs1, bvp_e4_st1, bvp_e4_rc1, bvp_e4_bs2, bvp_e4_st2,
bvp_e4_rc2, tbvp_e4_bs1, tbvp_e4_st1, tbvp_e4_rc1, tbvp_e4_bs2, tbvp_e4_st2, tbvp_e4_rc2
)

ibi_e4_bs1, ibi_e4_st1, ibi_e4_rc1, ibi_e4_bs2, ibi_e4_st2, ibi_e4_rc2, tibi_e4_bs1,
tibi_e4_st1, tibi_e4_rc1, tibi_e4_bs2, tibi_e4_st2, tibi_e4_rc2 = split_up_e4_ibi(subj,
ibi_e4_adj, time_e4_ibi, ibi_e4_bs1, ibi_e4_st1, ibi_e4_rc1, ibi_e4_bs2, ibi_e4_st2,
ibi_e4_rc2, tibi_e4_bs1, tibi_e4_st1, tibi_e4_rc1, tibi_e4_bs2, tibi_e4_st2, tibi_e4_rc2
)

hr_e4_bs1, hr_e4_st1, hr_e4_rc1, hr_e4_bs2, hr_e4_st2, hr_e4_rc2, hr_e4_bs1, thr_e4_st1,
 thr_e4_rc1, thr_e4_bs2, thr_e4_st2, thr_e4_rc2 = split_up_e4_hr(subj, hr_e4_adj,
time_e4_hr, hr_e4_bs1, hr_e4_st1, hr_e4_rc1, hr_e4_bs2, hr_e4_st2, hr_e4_rc2, thr_e4_bs1
, thr_e4_st1, thr_e4_rc1, thr_e4_bs2, thr_e4_st2, thr_e4_rc2)


#SPLIT UP S3 DATA
ppg_s3_bs1, ppg_s3_st1, ppg_s3_rc1, ppg_s3_bs2, ppg_s3_st2, ppg_s3_rc2, tppg_s3_bs1,
tppg_s3_st1, tppg_s3_rc1, tppg_s3_bs2, tppg_s3_st2, tppg_s3_rc2 = split_up_s3_ppg(subj,
tppg_s3_adj, ppg_s3_adj, ppg_s3_bs1, ppg_s3_st1, ppg_s3_rc1, ppg_s3_bs2, ppg_s3_st2,
ppg_s3_rc2, tppg_s3_bs1, tppg_s3_st1, tppg_s3_rc1, tppg_s3_bs2, tppg_s3_st2, tppg_s3_rc2
)

ibi_s3_bs1, ibi_s3_st1, ibi_s3_rc1, ibi_s3_bs2, ibi_s3_st2, ibi_s3_rc2, tibi_s3_bs1,
tibi_s3_st1, tibi_s3_rc1, tibi_s3_bs2, tibi_s3_st2, tibi_s3_rc2 = split_up_s3_ibi(subj,
tibi_s3_adj, ibi_s3_adj, ibi_s3_bs1, ibi_s3_st1, ibi_s3_rc1, ibi_s3_bs2, ibi_s3_st2,
ibi_s3_rc2, tibi_s3_bs1, tibi_s3_st1, tibi_s3_rc1, tibi_s3_bs2, tibi_s3_st2, tibi_s3_rc2
)

hr_s3_bs1, hr_s3_st1, hr_s3_rc1, hr_s3_bs2, hr_s3_st2, hr_s3_rc2, thr_s3_bs1, thr_s3_st1
, thr_s3_rc1, thr_s3_bs2, thr_s3_st2, thr_s3_rc2 = split_up_s3_hr(subj, thr_s3_adj,
hr_s3_adj, hr_s3_bs1, hr_s3_st1, hr_s3_rc1, hr_s3_bs2, hr_s3_st2, hr_s3_rc2, thr_s3_bs1,
 thr_s3_st1, thr_s3_rc1, thr_s3_bs2, thr_s3_st2, thr_s3_rc2)


#DOWNSAMPLE S3 DATA
ppg_s3_bs1, ppg_s3_st1, ppg_s3_rc1, ppg_s3_bs2, ppg_s3_st2, ppg_s3_rc2 =
downsampling_ppg(subj, tppg_s3_bs1, tppg_s3_st1, tppg_s3_rc1, tppg_s3_bs2, tppg_s3_st2,
tppg_s3_rc2, bvp_e4_bs1, bvp_e4_st1, bvp_e4_rc1, bvp_e4_bs2, bvp_e4_st2, bvp_e4_rc2,
ppg_s3_bs1, ppg_s3_st1, ppg_s3_rc1, ppg_s3_bs2, ppg_s3_st2, ppg_s3_rc2)

ibi_s3_bs1, ibi_s3_st1, ibi_s3_rc1, ibi_s3_bs2, ibi_s3_st2, ibi_s3_rc2, zero_ibi_bs1,
zero_ibi_st1, zero_ibi_rc1, zero_ibi_bs2, zero_ibi_st2, zero_ibi_rc2 = downsampling_ibi(
subj, zero_ibi_bs1, zero_ibi_st1, zero_ibi_rc1, zero_ibi_bs2, zero_ibi_st2, zero_ibi_rc2
, tibi_s3_bs1, tibi_s3_st1, tibi_s3_rc1, tibi_s3_bs2, tibi_s3_st2, tibi_s3_rc2,
ibi_e4_bs1, ibi_e4_st1, ibi_e4_rc1, ibi_e4_bs2, ibi_e4_st2, ibi_e4_rc2, ibi_s3_bs1,
ibi_s3_st1, ibi_s3_rc1, ibi_s3_bs2, ibi_s3_st2, ibi_s3_rc2)

hr_s3_bs1, hr_s3_st1, hr_s3_rc1, hr_s3_bs2, hr_s3_st2, hr_s3_rc2, zero_hr_bs1,
zero_hr_st1, zero_hr_rc1, zero_hr_bs2, zero_hr_st2, zero_hr_rc2 = downsampling_hr(subj,
```

```
zero_hr_bs1, zero_hr_st1, zero_hr_rc1, zero_hr_bs2, zero_hr_st2, zero_hr_rc2, thr_s3_bs1
, thr_s3_st1, thr_s3_rc1, thr_s3_bs2, thr_s3_st2, thr_s3_rc2, hr_e4_bs1, hr_e4_st1,
hr_e4_rc1, hr_e4_bs2, hr_e4_st2, hr_e4_rc2, hr_s3_bs1, hr_s3_st1, hr_s3_rc1, hr_s3_bs2,
hr_s3_st2, hr_s3_rc2)

    #bvp_e4_bs1, bvp_e4_st1, bvp_e4_rc1, bvp_e4_bs2, bvp_e4_st2, bvp_e4_rc2, tbvp_e4_bs1,
tbvp_e4_st1, tbvp_e4_rc1, tbvp_e4_bs2, tbvp_e4_st2, tbvp_e4_rc2 = upsampling_bvp(subj,
ppg_s3_bs1, ppg_s3_st1, ppg_s3_rc1, ppg_s3_bs2, ppg_s3_st2, ppg_s3_rc2, bvp_e4_bs1,
bvp_e4_st1, bvp_e4_rc1, bvp_e4_bs2, bvp_e4_st2, bvp_e4_rc2, tbvp_e4_bs1, tbvp_e4_st1,
tbvp_e4_rc1, tbvp_e4_bs2, tbvp_e4_st2, tbvp_e4_rc2)


    #HEARTPY
    if subj == 3 or subj == 16:
        print('Bad Quality')
    else:
        get_hr_s3_parameter(subj, 'PPG_S3_BS1', ppg_s3_bs1)
        get_hr_s3_parameter(subj, 'PPG_S3_ST1', ppg_s3_st1)
        get_hr_s3_parameter(subj, 'PPG_S3_RC1', ppg_s3_rc1)
        get_hr_s3_parameter(subj, 'PPG_S3_BS2', ppg_s3_bs2)
        get_hr_s3_parameter(subj, 'PPG_S3_ST2', ppg_s3_st2)
        get_hr_s3_parameter(subj, 'PPG_S3_RC2', ppg_s3_rc2)

        get_hr_e4_parameter(subj, 'PPG_E4_BS1', bvp_e4_bs1, tbvp_e4_bs1)
        get_hr_e4_parameter(subj, 'PPG_E4_ST1', bvp_e4_st1, tbvp_e4_st1)
        get_hr_e4_parameter(subj, 'PPG_E4_RC1', bvp_e4_rc1, tbvp_e4_rc1)
        get_hr_e4_parameter(subj, 'PPG_E4_BS2', bvp_e4_bs2, tbvp_e4_bs2)
        get_hr_e4_parameter(subj, 'PPG_E4_ST2', bvp_e4_st2, tbvp_e4_st2)
        get_hr_e4_parameter(subj, 'PPG_E4_RC2', bvp_e4_rc2, tbvp_e4_rc2)

        save_hr_parameters(subj, modes, bpm, breathingrate, hr_mad, ibi, pnn20, pnn50, rmssd
, s, sd1, sd1_sd2, sd2, sdnn, sdsd)

    #CREATE FILES
    create_file(subj, 'PPG_S3_BS1', ppg_s3_bs1[0], ppg_s3_bs1[1])
    create_file(subj, 'PPG_S3_ST1', ppg_s3_st1[0], ppg_s3_st1[1])
    create_file(subj, 'PPG_S3_RC1', ppg_s3_rc1[0], ppg_s3_rc1[1])
    create_file(subj, 'PPG_S3_BS2', ppg_s3_bs2[0], ppg_s3_bs2[1])
    create_file(subj, 'PPG_S3_ST2', ppg_s3_st2[0], ppg_s3_st2[1])
    create_file(subj, 'PPG_S3_RC2', ppg_s3_rc2[0], ppg_s3_rc2[1])

    create_file(subj, 'PPG_E4_BS1', bvp_e4_bs1, tbvp_e4_bs1)
    create_file(subj, 'PPG_E4_ST1', bvp_e4_st1, tbvp_e4_st1)
    create_file(subj, 'PPG_E4_RC1', bvp_e4_rc1, tbvp_e4_rc1)
    create_file(subj, 'PPG_E4_BS2', bvp_e4_bs2, tbvp_e4_bs2)
    create_file(subj, 'PPG_E4_ST2', bvp_e4_st2, tbvp_e4_st2)
    create_file(subj, 'PPG_E4_RC2', bvp_e4_rc2, tbvp_e4_rc2)

    if zero_ibi_bs1 == False:
        create_file(subj, 'IBI_S3_BS1', ibi_s3_bs1[0], ibi_s3_bs1[1])
        create_file(subj, 'IBI_E4_BS1', ibi_e4_bs1, tibi_e4_bs1)

    if zero_ibi_st1 == False:
        create_file(subj, 'IBI_S3_ST1', ibi_s3_st1[0], ibi_s3_st1[1])
        create_file(subj, 'IBI_E4_ST1', ibi_e4_st1, tibi_e4_st1)

    if zero_ibi_rc1 == False:
        create_file(subj, 'IBI_S3_RC1', ibi_s3_rc1[0], ibi_s3_rc1[1])
        create_file(subj, 'IBI_E4_RC1', ibi_e4_rc1, tibi_e4_rc1)

    if zero_ibi_bs2 == False:
        create_file(subj, 'IBI_S3_BS2', ibi_s3_bs2[0], ibi_s3_bs2[1])
        create_file(subj, 'IBI_E4_BS2', ibi_e4_bs2, tibi_e4_bs2)
```

```
    if zero_ibi_st2 == False:
        create_file(subj, 'IBI_S3_ST2', ibi_s3_st2[0], ibi_s3_st2[1])
        create_file(subj, 'IBI_E4_ST2', ibi_e4_st2, tibi_e4_st2)

    if zero_ibi_rc2 == False:
        create_file(subj, 'IBI_S3_RC2', ibi_s3_rc2[0], ibi_s3_rc2[1])
        create_file(subj, 'IBI_E4_RC2', ibi_e4_rc2, tibi_e4_rc2)


    if zero_hr_bs1 == False:
        create_file(subj, 'HR_S3_BS1', hr_s3_bs1[0], hr_s3_bs1[1])
        create_file(subj, 'HR_E4_BS1', hr_e4_bs1, thr_e4_bs1)

    if zero_hr_st1 == False:
        create_file(subj, 'HR_S3_ST1', hr_s3_st1[0], hr_s3_st1[1])
        create_file(subj, 'HR_E4_ST1', hr_e4_st1, thr_e4_st1)

    if zero_hr_rc1 == False:
        create_file(subj, 'HR_S3_RC1', hr_s3_rc1[0], hr_s3_rc1[1])
        create_file(subj, 'HR_E4_RC1', hr_e4_rc1, thr_e4_rc1)

    if zero_hr_bs2 == False:
        create_file(subj, 'HR_S3_BS2', hr_s3_bs2[0], hr_s3_bs2[1])
        create_file(subj, 'HR_E4_BS2', hr_e4_bs2, thr_e4_bs2)

    if zero_hr_st2 == False:
        create_file(subj, 'HR_S3_ST2', hr_s3_st2[0], hr_s3_st2[1])
        create_file(subj, 'HR_E4_ST2', hr_e4_st2, thr_e4_st2)

    if zero_hr_rc2 == False:
        create_file(subj, 'HR_S3_RC2', hr_s3_rc2[0], hr_s3_rc2[1])
        create_file(subj, 'HR_E4_RC2', hr_e4_rc2, thr_e4_rc2)


    #CROSSCORRELATIONS
    np_cross_correlation(subj, 'PPG_BS1', ppg_s3_bs1[0], bvp_e4_bs1)
    np_cross_correlation(subj, 'PPG_ST1', ppg_s3_st1[0], bvp_e4_st1)
    np_cross_correlation(subj, 'PPG_RC1', ppg_s3_rc1[0], bvp_e4_rc1)
    np_cross_correlation(subj, 'PPG_BS2', ppg_s3_bs2[0], bvp_e4_bs2)
    np_cross_correlation(subj, 'PPG_ST2', ppg_s3_st2[0], bvp_e4_st2)
    np_cross_correlation(subj, 'PPG_RC2', ppg_s3_rc2[0], bvp_e4_rc2)

    if zero_ibi_bs1 == False:
        np_cross_correlation(subj, 'IBI_BS1', ibi_s3_bs1[0], ibi_e4_bs1)
    if zero_ibi_st1 == False:
        np_cross_correlation(subj, 'IBI_ST1', ibi_s3_st1[0], ibi_e4_st1)
    if zero_ibi_rc1 == False:
        np_cross_correlation(subj, 'IBI_RC1', ibi_s3_rc1[0], ibi_e4_rc1)
    if zero_ibi_bs2 == False:
        np_cross_correlation(subj, 'IBI_BS2', ibi_s3_bs2[0], ibi_e4_bs2)
    if zero_ibi_st2 == False:
        np_cross_correlation(subj, 'IBI_ST2', ibi_s3_st2[0], ibi_e4_st2)
    if zero_ibi_rc2 == False:
        np_cross_correlation(subj, 'IBI_RC2', ibi_s3_rc2[0], ibi_e4_rc2)

    if zero_ibi_bs1 == False:
        np_cross_correlation(subj, 'HR_BS1', hr_s3_bs1[0], hr_e4_bs1)
    if zero_ibi_st1 == False:
        np_cross_correlation(subj, 'HR_ST1', hr_s3_st1[0], hr_e4_st1)
    if zero_ibi_rc1 == False:
        np_cross_correlation(subj, 'HR_RC1', hr_s3_rc1[0], hr_e4_rc1)
    if zero_ibi_bs2 == False:
```

```
        np_cross_correlation(subj, 'HR_BS2', hr_s3_bs2[0], hr_e4_bs2)
    if zero_ibi_st2 == False:
        np_cross_correlation(subj, 'HR_ST2', hr_s3_st2[0], hr_e4_st2)
    if zero_ibi_rc2 == False:
        np_cross_correlation(subj, 'HR_RC2', hr_s3_rc2[0], hr_e4_rc2)

    print(corr_coef)
    print(corr_mode)

    if subj == 19:
        create_corr_file('CorrCoef', corr_coef, corr_mode)

    #PARAMETER TRANSFORMATION
    if subj == 3 or subj == 16:
        print('No Files available')
    else:
        read_hr_parameters(subj, bpm_e4, bpm_s3, ibi_e4, ibi_s3, rmssd_e4, rmssd_s3, sdnn_e4
    , sdnn_s3)

    #CLEAN LISTS
    time_s3_adj = []
    ppg_s3_adj  = []
    ibi_s3_adj  = []
    hr_s3_adj   = []

    bvp_e4_adj = []
    ibi_e4_adj = []
    ibt_e4_adj = []
    hr_e4_adj  = []

    ppg_s3_bs1  = []
    ppg_s3_st1  = []
    ppg_s3_rc1  = []
    ppg_s3_bs2  = []
    ppg_s3_st2  = []
    ppg_s3_rc2  = []

    ibi_s3_bs1  = []
    ibi_s3_st1  = []
    ibi_s3_rc1  = []
    ibi_s3_bs2  = []
    ibi_s3_st2  = []
    ibi_s3_rc2  = []

    hr_s3_bs1   = []
    hr_s3_st1   = []
    hr_s3_rc1   = []
    hr_s3_bs2   = []
    hr_s3_st2   = []
    hr_s3_rc2   = []

    bvp_e4_bs1  = []
    bvp_e4_st1  = []
    bvp_e4_rc1  = []
    bvp_e4_bs2  = []
    bvp_e4_st2  = []
    bvp_e4_rc2  = []

    ibi_e4_bs1  = []
    ibi_e4_st1  = []
    ibi_e4_rc1  = []
    ibi_e4_bs2  = []
    ibi_e4_st2  = []
```

```
    ibi_e4_rc2  = []

    hr_e4_bs1   = []
    hr_e4_st1   = []
    hr_e4_rc1   = []
    hr_e4_bs2   = []
    hr_e4_st2   = []
    hr_e4_rc2   = []

    tppg_s3_bs1 = []
    tppg_s3_st1 = []
    tppg_s3_rc1 = []
    tppg_s3_bs2 = []
    tppg_s3_st2 = []
    tppg_s3_rc2 = []

    tibi_s3_bs1 = []
    tibi_s3_st1 = []
    tibi_s3_rc1 = []
    tibi_s3_bs2 = []
    tibi_s3_st2 = []
    tibi_s3_rc2 = []

    thr_s3_bs1  = []
    thr_s3_st1  = []
    thr_s3_rc1  = []
    thr_s3_bs2  = []
    thr_s3_st2  = []
    thr_s3_rc2  = []

    tbvp_e4_bs1 = []
    tbvp_e4_st1 = []
    tbvp_e4_rc1 = []
    tbvp_e4_bs2 = []
    tbvp_e4_st2 = []
    tbvp_e4_rc2 = []

    tibi_e4_bs1 = []
    tibi_e4_st1 = []
    tibi_e4_rc1 = []
    tibi_e4_bs2 = []
    tibi_e4_st2 = []
    tibi_e4_rc2 = []

    thr_e4_bs1  = []
    thr_e4_st1  = []
    thr_e4_rc1  = []
    thr_e4_bs2  = []
    thr_e4_st2  = []
    thr_e4_rc2  = []

    zero_ibi_bs1 = False
    zero_ibi_st1 = False
    zero_ibi_rc1 = False
    zero_ibi_bs2 = False
    zero_ibi_st2 = False
    zero_ibi_rc2 = False

    zero_hr_bs1 = False
    zero_hr_st1 = False
    zero_hr_rc1 = False
    zero_hr_bs2 = False
    zero_hr_st2 = False
```

```
    zero_hr_rc2 = False

    modes          = ['Modes']
    bpm            = ['Bpm']
    breathingrate  = ['Breathingrate']
    hf             = ['Hf']
    hr_mad         = ['Hr_mad']
    ibi            = ['Ibi']
    lf             = ['Lf']
    lf_hf          = ['Lf/Hf']
    pnn20          = ['Pnn20']
    pnn50          = ['Pnn50']
    rmssd          = ['Rmssd']
    s              = ['S']
    sd1            = ['Sd1']
    sd1_sd2        = ['Sd1/Sd2']
    sd2            = ['Sd2']
    sdnn           = ['Sdnn']
    sdsd           = ['Sdsd']

    print('ALL LISTS ARE CLEANED!')

create_parameter_file(subjects, bpm_e4, bpm_s3, ibi_e4, ibi_s3, rmssd_e4, rmssd_s3, sdnn_e4,
      sdnn_s3)
```

Appendix F – Bland-Altman Plot (R-Script)

```r
install.packages("blandr")
library(blandr)
summary <- read.csv("Summary_HR_Parameter.csv", header=TRUE, sep = ",", quote = "\"'", dec =
    ".", fill = TRUE)

statistics.results_hr <- blandr.statistics(summary$bpm_e4 , summary$bpm_s3 )

jpeg("Bland-Altman-Plot_HR_neu.jpeg", width = 1000, height = 500)
blandr.plot.ggplot(statistics.results_hr, method1name = "E4",
                   method2name = "S3",
                   ciDisplay = TRUE, ciShading = TRUE, normalLow = FALSE,
                   normalHigh = FALSE, overlapping = FALSE,
                   plotProportionalBias = FALSE,
                   plotProportionalBias.se = TRUE, assume.differences.are.normal = TRUE)+
  geom_hline(yintercept=5, size=1.1)+
  geom_hline(yintercept=-5, size=1.1)+
  geom_point(shape=16, size=4)+
  labs(title = "HR", x="Mean HR (E4,S3)",y = "Difference HR (E4-S3)")+
  theme(plot.title = element_text(size = 20, face = "bold"), axis.title.x=element_text(size
    =15),axis.title.y=element_text(size=15))
dev.off()

statistics.results_sdnn <- blandr.statistics(summary$sdnn_e4 , summary$sdnn_s3 )

jpeg("Bland-Altman-Plot_SDNN_neu.jpeg", width = 1000, height = 500)
blandr.plot.ggplot(statistics.results_sdnn, method1name = "E4",
                   method2name = "S3",
                   ciDisplay = TRUE, ciShading = TRUE, normalLow = FALSE,
                   normalHigh = FALSE, overlapping = FALSE,
                   plotProportionalBias = FALSE,
                   plotProportionalBias.se = TRUE, assume.differences.are.normal = TRUE)+
  geom_hline(yintercept=0.56, size=1.1)+
  geom_hline(yintercept=-0.56, size=1.1)+
  geom_point(shape=16, size=4)+
  labs(title = "SDNN", x="Mean SDNN (E4,S3)",y = "Difference SDNN (E4-S3)")+
  theme(plot.title = element_text(size = 20, face = "bold"), axis.title.x=element_text(size
    =15),axis.title.y=element_text(size=15))
dev.off()

statistics.results_rmssd <- blandr.statistics(summary$rmssd_e4 , summary$rmssd_s3 )

jpeg("Bland-Altman-Plot_RMSSD_neu.jpeg", width = 1000, height = 500)
blandr.plot.ggplot(statistics.results_rmssd, method1name = "E4",
                   method2name = "S3",
                   ciDisplay = TRUE, ciShading = TRUE, normalLow = FALSE,
                   normalHigh = FALSE, overlapping = FALSE,
                   plotProportionalBias = FALSE,
                   plotProportionalBias.se = TRUE, assume.differences.are.normal = TRUE)+
  geom_hline(yintercept=0.71, size=1.1)+
  geom_hline(yintercept=-0.71, size=1.1)+
  geom_point(shape=16, size=4)+
  labs(title = "RMSSD", x="Mean RMSSD (E4,S3)",y = "Difference RMSSD (E4-S3)")+
  theme(plot.title = element_text(size = 20, face = "bold"), axis.title.x=element_text(size
    =15),axis.title.y=element_text(size=15))
dev.off()
```

Appendix G – Error Bar Plot (R-Script)

```r
rm(list = ls())
#########Task1
library(ggplot2)
library(grDevices)
task1 <- read.csv("Summary_1st_Task.csv", header=TRUE, sep = ",", quote = "\"'", dec = ".", fill = TRUE)

zero<- c(0,0,0)
boun1<- c(6.809,6.809,6.809)
boun2<- c(-6.809,-6.809,-6.809)
data<- cbind(task1, zero, boun1, boun2)
windows(width=10, height=5)
   ggplot(data, aes(x =  factor(Phase, level = c('BS1', 'ST1', 'RC1')), group=1)) +
     geom_line(aes(y=p1), size = 1, color= "darkgrey") +
     geom_line(aes(y=p2), size = 1, color= "darkgrey")+
     geom_line(aes(y=p4), size = 1, color= "darkgrey")+
     geom_line(aes(y=p5), size = 1, color= "darkgrey")+
     geom_line(aes(y=p6), size = 1, color= "darkgrey")+
     geom_line(aes(y=p7), size = 1, color= "darkgrey")+
     geom_line(aes(y=p9), size = 1, color= "darkgrey")+
     geom_line(aes(y=p15), size = 1, color= "darkgrey")+
     geom_line(aes(y=p18), size = 1, color= "darkgrey")+
     geom_line(aes(y=zero), size = 1, color= "black")+
     geom_line(aes(y=boun1), size = 1, color= "green")+
     geom_line(aes(y=boun2), size = 1, color= "green")+
     geom_line(aes(y=mean), size= 1, color = "red")+
     geom_errorbar(aes(ymin = mean-se, ymax= mean + se),color="red", size=1, width=.2)+
     labs(title = "Modified SSST (E4-S3)", x="",y = "Heart Rate: E4 - S3 (bpm)")

#######Task2
rm(list = ls())
task2 <- read.csv("Summary_2nd_Task.csv", header=TRUE, sep = ",", quote = "\"'", dec = ".", fill = TRUE)

zero<- c(0,0,0)
boun1<- c(13.33,13.33,13.33)
boun2<- c(-13.33,-13.33,-13.33)
data<- cbind(task2, zero, boun1, boun2)
windows(width=10, height=5)
 ggplot(data, aes(x =  factor(Phase, level = c('BS2', 'ST2', 'RC2')), group=1)) +
     geom_line(aes(y=p2), size = 1, color= "darkgrey")+
     geom_line(aes(y=p5), size = 1, color= "darkgrey")+
     geom_line(aes(y=p7), size = 1, color= "darkgrey")+
     geom_line(aes(y=p15), size = 1, color= "darkgrey")+
     geom_line(aes(y=p18), size = 1, color= "darkgrey")+
     geom_line(aes(y=zero), size = 1, color= "black")+
     geom_line(aes(y=boun1), size = 1, color= "green")+
     geom_line(aes(y=boun2), size = 1, color= "green")+
     geom_line(aes(y=mean), size= 1, color = "red")+
     geom_errorbar(aes(ymin = mean-se, ymax= mean + se),color="red", size=1, width=.2)+
     labs(title = "Stroop-Task (E4-S3)", x="",y = "Heart Rate: E4 - S3 (bpm)")
```

Appendix H – Overview of mean STAI scores of participants per phase

Participant	T1 (before SSST)	T2 (after SSST)	T3 (after Recovery)	T4 (before Stroop)	T5 (after Stroop)	T6 (after Recovery)
1	40	53.33	23.33	26.66	40	20
2	23.33	23.33	23.33	23.33	30	26.66
3	46.66	53.33	50	43.33	46.66	40
4	33.33	56.66	26.66	36.66	46.66	30
5	30	43.33	33.33	40	30	36.66
6	30	26.66	23.33	20	43.33	30
7	43.33	43.33	33.33	26.66	43.33	33.33
8	40	43.33	30	43.33	50	43.33
9	53.33	43.33	46.66	50	56.66	46.66
10	46.66	63.33	40	20	30	40
11	40	40	23.33	20	36.66	23.33
12	36.66	36.66	33.33	33.33	33.33	33.33
13	66.66	56.66	36.66	46.66	60	43.33
14	30	43.33	43.33	30	40	30
15	26.66	33.33	30	33.33	40	33.33
16	43.33	46.66	53.33	23.33	50	26.66
17	33.33	46.66	26.66	23.33	30	23.33
18	33.33	26.66	26.66	20	33.33	20
19	36.66	46.66	30	40	36.66	46.66
20	40	43.33	36.66	33.33	40	30