From chunking drills to Hallelujah:

Using new methods to train and evaluate complete piano beginners

Ruben Grasemann

Supervisors

Prof. Dr.-Ing. Willem Verwey

Dr. Martin Schmettow

Department of Cognitive Psychology and Ergonomics

Faculty of Behavioural Sciences

University of Twente

Enschede, 07.08.2018

Abstract

An experiment was conducted to gain insight into the use of chunking drills in the context of music and its benefits for the acquisition of piano playing skills. Completely music naive participants were trained to play a short musical piece by practicing one of two learning protocols: The control group (n=16) practiced the whole piece repeatedly, while the experimental group (n=15) engaged in chunking drills. The chunking drill protocol was developed by breaking down the piece into a total of 10 segments, each consisted of 3 to 8 stimuli. Music gaming software was used to bypass the prerequisite of reading musical notes. Performance on the test block, that followed the practice phase and consisted of the whole piece, was expected to be higher for participants of the experimental group. Audio recordings were quantified and analyzed by using the programming language R. Results showed significantly higher scores for the experimental group in both, number of correctly played notes and performance on a composite score, that takes pitch and timing errors into account. The present study supports previous research that claimed performance enhancement due to chunking drills. Moreover, an innovative method was developed that not only enables the observation of music naive participants but also illustrates new ways of precise response time measurement and efficient analysis in music research. Several research suggestions are proposed for further investigation into the acquisition of piano playing skill.

Keywords: music training, cognitive chunking, skill acquisition, motor learning, motor skills, sight-reading, piano skills, pitch, rhythm, music performance, performance estimation, response time measurement, analysis in R

Introduction

The topic of music is a broad field of research that attracted the attention of philosophers and psychologists as an interesting and complex form of art (Swanwick, 2002). Musical skill is considered a powerful and unique form of communication and the acquisition of musical skill is associated with the understanding of oneself and the ability to relate to others (QCA, 1999). Moreover, learning to play music is proposed to increase self-discipline, creativity, aesthetic sensitivity, and fulfilment (QCA, 1999). Accordingly, music education became a vital aspect of school curricula, established in recent guidelines like the American *Every child achieves act* (2015) and the *Lifelong Learning Programme* (2006), which serves as a guideline for European Union member states to set up their own music education systems. Despite the relevance of the topic, surprisingly little knowledge is applied when it comes to effective learning strategies in the context of musical skill acquisition (McPherson, 2005). Research suggests that music teachers rather pay attention to the performance of their students than on sufficiently emphasizing the development of task-appropriate strategies to aid their performance (McPherson, 2005). It was argued that better instructions on strategies would improve the efficiency of music education (McPherson, 2005; Miksza, 2007). This paper aims on investigating factors that aid the acquisition of music skill.

One of the most important factors of every musician's development is the time devoted to practice (Kohut, 1985, Sloboda, 1996; Lehmann & Ericsson, 1997; Smith, 2002). Miksza (2007) observed practice behavior and studied self-reported practice habits of 60 wind players to specify practice behaviors that are the best predictors of performance achievement. Strategies that improved musical skill were the *whole-part-whole strategy*, where the whole segment is played at first, then a smaller phrase within the piece is isolated and played separately, before the whole segment is played again. A similar approach is to repeatedly practice a specific segment of the piece: Isolating hard parts from the whole piece and practice them separately has proven to enhance performance especially in the beginning stages of playing a musical instrument (Miksza, 2007). These strategies can be applied by skipping directly to or just before critical musical sections of the étude. Another successful strategy used was *chaining*, where participants started off with a small segment of the piece and then systematically added segments before or after the segment they started with. Furthermore, the usage of a metronome is recommended for beginners to help them to develop rhythm skills and understanding of tempo. Moreover, Miksza (2007) found that participants improved performance when starting the practice by slowing the tempo down and, as learning goes by, accelerating accordingly until the original tempo is reached (Barry & McArthur, 1994). This was already recommended by Bach (1753) who further emphasized the importance of correct arm, hand, and fingers positioning.

Exploring the development of information processing and consequent movement production is an important topic within the field of cognitive research. Several studies investigated musicians and people who learn to play music, as playing an instrument can serve as an example of a complex, highly practiced task. Hatfield (2016) found that setting specific and hierarchical goals enhances participants' concentration, self-observation, and self-efficacy when it comes to instrumental practice and performance. Studies on attention argued that increasing the distance of the effect (the produced sound) from the action producing it (pressing down a specific key) through manipulation of the attentional focus, enhanced learning (McNevin et al., 2003; Duke et al., 2011). This was illustrated by Duke, Cash and Allen (2011) who found that the performance of advanced musicians on keyboard playing was most accurate when participants focused on the effects their movements produced rather than on the movements of their fingers, the piano keys, or the piano hammers themselves.

According to the *theory of event coding*, systematic interactions between perception, action planning and sensorimotor processing are produced by multi-layered networks of bindings, called *event files* (Hommel et al., 2001). Therefore, musicians form associations between a specific pitch, the corresponding written note, and the physical action that must be performed on an instrument, for example a specific arm and finger movement. Event files might also contain bindings for whole melodies. Development of these associations could explain why expert musicians are able to perform unfamiliar pieces without practicing first. This can be applied to learning programs by strategies such as listening and studying pieces beforehand (Barry, 1991).

Cognitive theories might not only explain why known strategies yield improved performance but could also serve as a starting point to develop new strategies on the acquisition of music skill. Research shows that execution of complex tasks can be illustrated by using a *hierarchical model* (Gallistel, 1980; Verwey et al., 2010). This idea was also transferred to the theory of music, where the highest hierarchy would be the whole piece, which consists of sub pieces until the lowest hierarchy, the single movement, is reached, e.g. pressing a piano key (Verwey, 2010; Palmer, 1997). The separation of a whole movement into subsequences reduces the memory load while executing a continuing performance (Bo and Seidler, 2009; Halford et al., 1998, Ericsson et al., 1980). A small number of individual motoric movements, that is bound in a fluid, uniform movement, is called a *motor chunk* (Halford et al., 1998; Pew, 1966; Verwey, 1996). Pike & Carter (2010) investigated sight-reading in piano playing, which refers to simultaneously reading and performing a piece of music that is new to the player. Sight-readers benefit from encoding separate pitches and rhythms into chunks of familiar chords and rhythmic patterns (Drake & Palmer, 2000, Gilman & Underwood, 2003). Pike & Carter (2010) developed two different exercises to minimize rhythm and pitch errors correspondingly, as these were found to be the most common mistakes made by piano sight-reading beginners (Gudmundsdottir, 2008). These short piano exercises were meant to be executed repeatedly and designed to encourage chunking of rhythm or pitch patterns. They were therefore called *chunking drills*. It was found that, regardless of modality, rhythm and continuity, performance scores of sight-reading beginners who practiced these chunking drills improved. It was suggested to further investigate strategies, such as *chunking*, that could assist students in meeting basic keyboard competencies efficiently and effectively (Pike & Carter, 2010).

When following the promising idea to further examine *chunking drills* in the context of music, a very similar task, the *discrete sequence production* (DSP) task can be found (for reviews, see Abrahamse et al., 2013; Verwey et al., 2015). During the DSP task, participants are asked to place their fingers on a keyboard and to execute a fixed series of 3-7 stimuli, similar to playing an arpeggio on a piano. During the practice phase, which includes 500 – 1000 repetitions per sequence, *motor chunks* develop by forming associations between successive response representations. When motor chunks develop, participants acquire the skill to “rapidly and accurately produce a sequence of movements with limited effort and/or attentional monitoring” (Abrahamse et al., 2013, p. 1). The *Dual Processor model* provides an explanation of the cognitive processes underlying discrete sequence production (Verwey, 2001; Verwey et al, 2010).

According to the *Dual processor model* participants go through three different stages of sequence execution (Abrahamse et al., 2013). At first, when unfamiliar sequences are presented, the stimulus (e.g. a note on the sheet) is processed by the *cognitive processor*. The *cognitive processor* then triggers the *motor processor* to produce the required response. This is called the *reaction mode*. During *associative mode*, the *cognitive processor* develops a weak sequence representation, but stimuli remain necessary. When this representation gets stronger, *motor* *chunks* develop, allowing the *chunking mode* (Abrahamse et al., 2013). The *chunks* are then loaded into the *motor buffer* before being executed. When the chunk is loaded into the *motor buffer*, the *cognitive processor* triggers the *motor processor* to read the *chunk* and execute it fairly autonomously as if it were a single response (Abrahamse et al., 2013, Verwey et al., 2015).

During the present study music naive participants were asked to practice a short beginners piano piece (a simplified version of Leonard Cohens' Hallelujah) by using Synthesia, a music video game and piano keyboard trainer, which allows users to play a MIDI keyboard in time to a MIDI file by following on-screen directions. Instead of displaying musical notes in the form of sheet music it displays a keyboard at the bottom of the screen and bars that continuously move down towards the keyboard from the top of the screen. The bars show which key must be pressed at what time and for how long. This is indicated by the key that the bar falls upon, the moment the bar hits the key and the length of the bar, correspondingly (see Fig. 1).

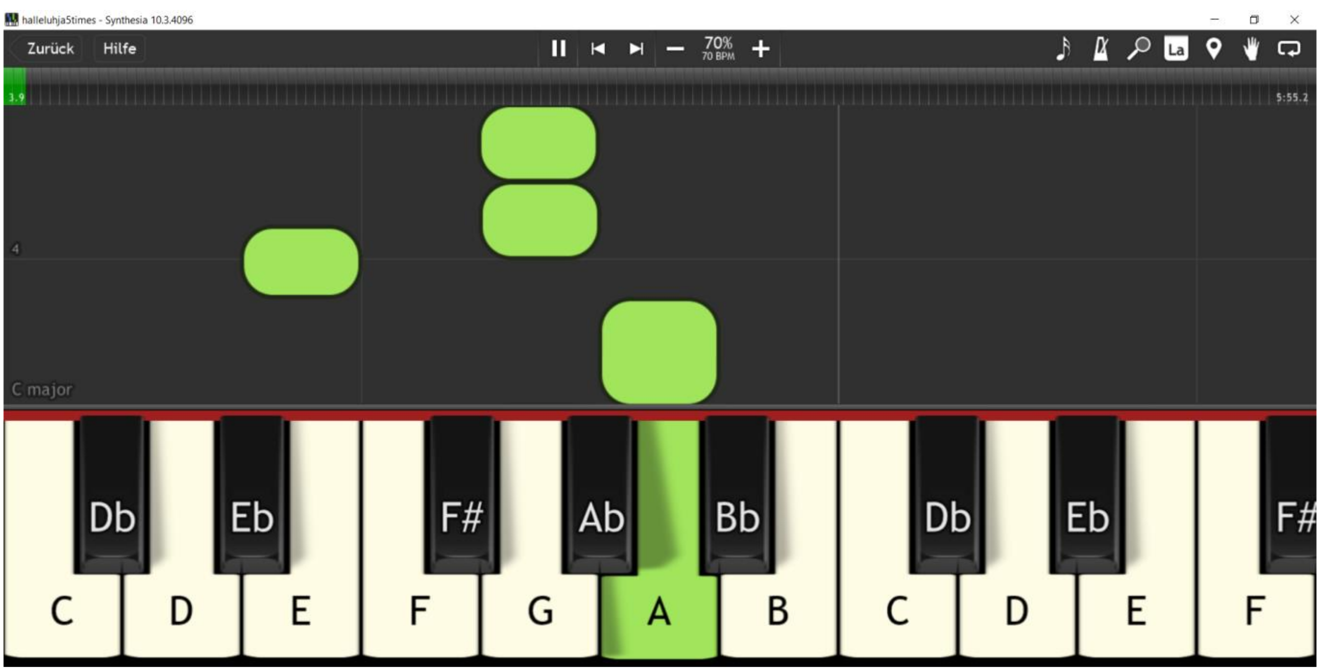


Fig. 1 Synthesia overview while playing Hallelujah

Therefore, the program allows to be used without any prior knowledge in reading sheet music, while still providing a setting similar to sight-reading, where players cannot focus on their hands or the keyboard as they would otherwise lose track of the notes that must be played. Moreover, previously-mentioned factors that are suggested to improve the acquisition of music skill can be taken into account, as Synthesia allows to use a metronome, and to adjust the tempo of the piece. The advantage of using a digital setup with Synthesia and a MIDI keyboard not only offers accessibility to music naive players but also enables accurate recordings of the participants keypresses and key releases so that they can be compared to the source file, that is read into Synthesia instead of sheet music. Previously cited research in the field of music generally uses participants familiar with playing instruments, as well as performance estimation either by specific programs or by professionals. In contrast to that, the present study sought to find a new method to compare measurement of response times to the source file quantitively.

To validate this new method, two different learning protocols were designed. While both protocols lasted 60 minutes, started off with a five-minute familiarization phase, and ended with a test phase to estimate the participants performance, the specific exercise protocols differed: the control protocol simply repeated the whole piece 25 times successively, whereas the experimental protocol used chunking drills to teach the piece. The drills were designed by splitting the piece into 10 segments, that consisted of 3 to 8 stimuli. To facilitate chunking, each segment was recognizable and meaningful within the whole piece as it consisted of a phrase, which can be explained as “the smallest musical unit that conveys a more or less complete musical thought” (White, 1994, p. 71). The chunking drills were presented in a randomized order. After the chunking drills were exercised, the control protocol repeated the complete piece five times in a row, so that participants could get used to the original order, in which the segments were assembled within the piece. The aim of this study is to answer the research question whether providing students with chunking drills would result in improved performance when compared to students who did not engage in chunking-drill exercises. Performance is estimated by comparing the number of correct keypresses and correctly played length of the notes, as well as the participants timing of keypress and length compared to the source file.

Method

Participants

Thirty-two students (20 female, M age = 20.88 years, SD = 2.79 years) from the University of Twente took part in this study in exchange for course credit. Informed consent was obtained from all individual participants included in the study. One participant was removed due to a high amount of keypress errors. The study had been approved by the Ethics Committee of the University of Twente and was performed in accordance with the ethical standards described in the Declaration of Helsinki.

Apparatus

The experiment was conducted in the piano keyboard training program Synthesia 10.3 running under Windows 10. Instructions and stimuli were presented on a 15.6″ TFT display, with a resolution of 1.920 × 1.080 pixels of an Acer Aspire V5 laptop computer. An AKAI LPK25 MIDI keyboard was used as input device (see Fig. 2). Participants used the MIDI keyboard keys to react to the stimuli. The instructor solely used the on-board keyboard and mouse to start the practice and test blocks. The room (2.25 × 2.25 × 3.50 m) was dimly lit with fluorescent light and fitted with a webcam for monitoring purposes.

Fig. 2 AKAI LPK25 MIDI keyboard used for the present study

The device driver “LoopBe1” was used as an internal MIDI device to transfer the Synthesia MIDI data to the program MidiEditor in real time. MidiEditor was used to record the keystrokes and to store the performance as MIDI files (see Fig. 3). A Rscript was written to extract response times from the MIDI files by using the ‘signal’ and the ‘tuneR’ library.

Task

At the beginning of the experiment, participants were instructed to place their right little, ring, middle, and index fingers on the keys corresponding to the notes F, E, D, C. They were asked not to use the left hand at any time. Synthesia displayed a keyboard with 25 keys at the bottom of the screen, corresponding to the LPK25 Midi keyboard keys. The program then directed which key had to be pressed by bars that moved down from the top of the screen to the displayed keyboard. The moment, the bar hit the on-screen keyboard key, the corresponding key on the LPK25 keyboard had to be pressed. The length of the bar indicated the duration of the keypress.

The first block was conducted to familiarize the participants with the keyboard keys and the music piece, the simplified version of Leonard Cohens' "Hallelujah". At first, participants were asked to listen to the full piece, which took two and a half minutes. Then the familiarization block was conducted in Synthesia. During this block, the whole piece passed through, 3 times in a row, and participants were asked to react to the bars by pressing the corresponding keys. However, performance was not measured, and the program continuously stopped the piece until the correct key was pressed. The experimenter observed the participants and advised and corrected the positioning of arm, hand, and fingers.

For the practice blocks 2-6, participants were randomly assigned to one of two conditions depending on the participant number. Even participant numbers (n=16) were assigned to the experimental condition while odd participant numbers (n=16) were assigned to the control condition. Over the course of all following 6 blocks Synthesia did not stop the piece when an incorrect key was pressed. Whenever participants pressed a false key, the corresponding key on the displayed keyboard was highlighted grey. The program continued with the next note of the given sequence. Correct keypresses resulted in a highlighted green of the key displayed (see the correctly played A-key in Fig. 1). An in-build metronome was used to aid participants in the timing. At the end of each block, a message informed the participant that the block had finished. During the break that followed each block, participants were encouraged to improve performance of the duration of the keypress and were reinstructed on correct arm, hand, and finger positioning if necessary.

For the experimental condition, the piece was split up in a total of 10 segments consisting of 3 to 8 elements, based on phrases within the piece. Blocks 2-5 were each composed of 2-3 segments of the piece (depending on the segment length) that where repeated for 5 minutes per block. During block 6, all 10 segments were assembled back into the correct order according to the piece, the full piece was then executed 5 times in a row. Participants in the experimental condition practiced each of the 10 segments 20 times and the full piece 8 times. Each block took 5 minutes to execute.

In the control condition, blocks 2-6 each consisted of the full piece, executed 5 times in a row per block. Across Blocks 1 to 6 participants in the control group performed 28 repetitions of the whole piece. Again, each block took 5 minutes to execute. The duration of the practice phase, as well as the number of total keypress repetitions was comparable between the two groups.

After the practice blocks, the same test phase, block 7, was conducted for both groups. During this block, participants played the full piece 3 times in a row, with 10-second pauses in-between. Performance of the test block was recorded and stored as a MIDI file, that also contained the response time data.

Procedure

At the start of the experiment, participants were asked to take a seat in front of the laptop. They were instructed that they had to learn a musical piece by using the Synthesia software and that response times would be measured. Participants were told that the experiment would last about one hour, they were asked to respond to the falling bars that were displayed in Synthesia by pressing the corresponding keys. Participants were told to focus not on the keyboard, but only on the screen, to press as little wrong keys as possible, and to release the key at the correct point in time. They were also told that they had a 3-minute break after each block. Furthermore, participants were informed that participation was voluntary, that no risks were involved in participating, that the data collection would be anonymous and that they were filmed for monitoring purposes. Participants then signed the informed consent form while the experimenter wrote down the number and name of the participant, the date and the time of the day into the logbook. After that, the experimenter started the program and instructed the participant about the correct arm, hand, and finger position. Before the experiment started, participants listened to the full piece.

The experiment consisted of 7 blocks in a single session, starting with the familiarization phase, which was the same for all participants. Then, participants were randomly assigned to either the experimental or the control group and practice blocks 2 to 6 were conducted. At the end of each practice block participants received feedback, displaying the amount of correct responses and errors. After completing the practice blocks, the test phase, block 7, was conducted, which was the same for both groups. After participants completed block 7, the experimenter wrote down events into the logbook that could have had an impact on the experiment and granted the credits.

Data analysis

Each recording of the test block was compared to the source Midi file (that Synthesia used to simulate the blocks) note by note. Whenever a note should be played, according to the source file, the Rscript checked, whether the participant pressed the correct key, corresponding to the note. To account for differences in the interpretation of the piece, correct keypresses were determined to be valid when the onset occurred within an interval margin of half a second before or after the note should have been played. Therefore, if participants played the desired note, but the keypress occurred earlier or later than 500ms of the moment it should have been played, the keypress was counted as 'keypress-error' (for an illustration of this +/- 500ms margin, see Fig. 3). Moreover, not more than one keypress per note was determined to be a 'correct keypress', this was introduced to account for situations in which participants pressed the correct key multiple times within the margin of 1 second. Whenever a note was marked as correct keypress, the program checked the duration of the keypress by recording the time of the 'key-release' and subtracting it from the timecode of the correct keypress. Again, the length of the played note had to be within a margin of +/- 500 milliseconds compared to the length of the corresponding note in the source file. If the latter was true, the note was marked as correct key-release. Therefore, this variable involved all instances in which participants pressed and then released the key at the correct time. Whenever, a key was pressed too short or too long, it was counted as key-release error.

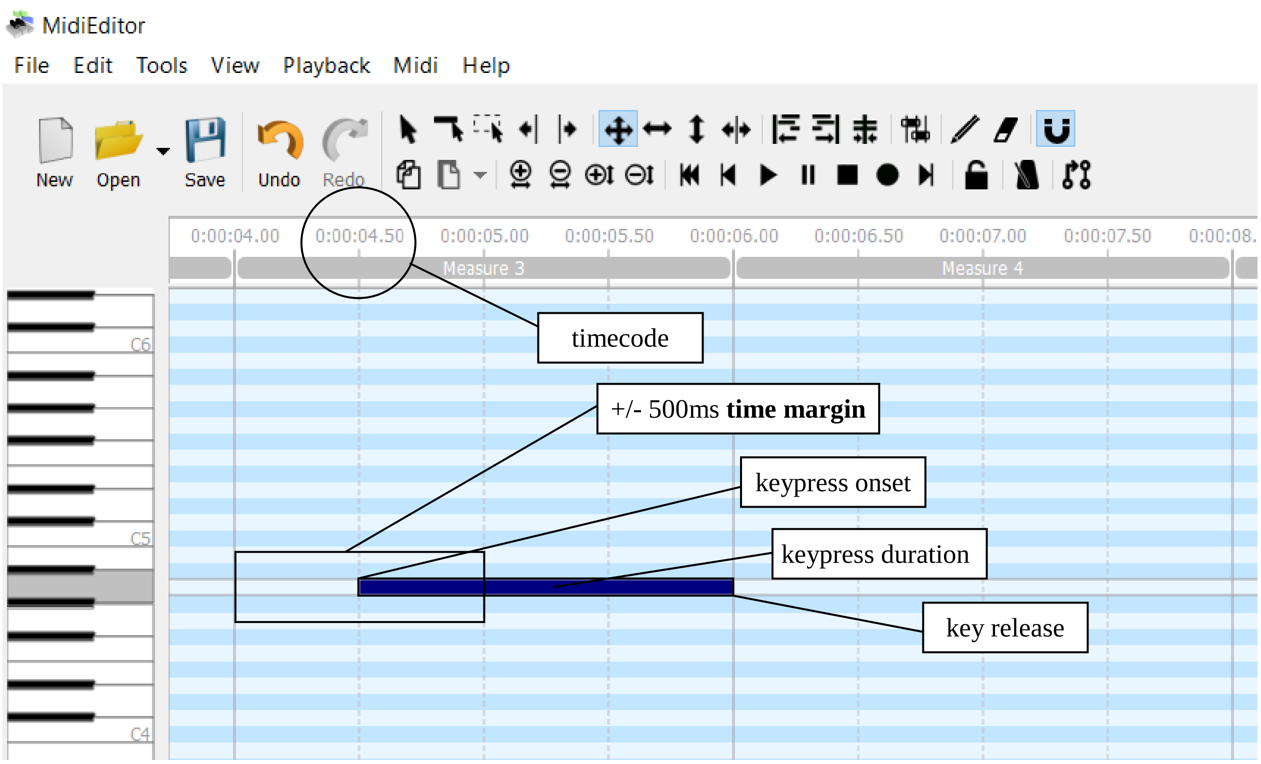


Fig. 3 Illustration of the 1 second time margin for keypress onset based on the depiction of the 8th keystroke of the source file, within the program MidiEditor. The corresponding note was 'A', the keypress started at 4.5 sec into track and lasted for 1.5 sec.

Previous research in the field of sight-reading, proposed a system for performance estimation by using a ‘composite score’. This composite score takes all performance factors, such as pitch and timing, into account (for reviews, see: Henry & Demorest, 1994; Demorest & May, 1995; Demorest, 1998). These models usually grant one full point for each correctly performed note (correct pitch and duration) and deduct one-half point for each timing error and one-half point for each pitch error. It is important to note however, that participants of studies using this scoring system were students that had received superior (highest) ratings in sight-singing contests (Demorest & May, 1995). Contrary to previous research, the present study consisted of participants without any prior musical knowledge. Therefore, it was predicted that keypresses that are both, correctly played regarding the pitch (note that must be played) as well as the timing (duration of the keypress) are scarce compared to previous research, while errors are assumed to be abundant. To take this into account, a modified version of the “performance estimation” was applied, that grants two points for correctly played notes (regarding pitch and duration) and deducts one-half point for each error in pitch and timing (see equation 1). Therefore, with a total of 312 notes that should have been played according to the source file, the hypothetical maximum score was 624.

$$\textit{composite score} = \textit{correctly played notes} * 2 - \textit{pitch error} * 0.5 - \textit{timing error} * 0.5 \qquad (1)$$

For deeper analysis of the performance of ‘keypress’ and ‘keypress duration’, two additional scores were introduced for each participant. Instead of counting errors, the absolute deviation in milliseconds was calculated between the moment of the keypress, and the moment the note should have been played, as well as the absolute duration of the keypress and the desired duration according to the source file. This was to account for the hypothetical scenario that two participants had similar scores in 'correct keypresses' and 'key duration' within the predetermined time margins, while one participant might perform closer to the desired timing of 'keypress' and 'duration'.

The previously mentioned scores for 'correct keypresses', 'wrong keypresses', 'correct key-releases', 'wrong key-releases', as well as the 'composite score' were checked for individuals that scored higher than the average plus three times the standard deviation of the corresponding scale. This excluded one participant of the experimental group due to a high number of 'keypress errors'. All scores did not deviate according to Shapiro-Wilks tests, so that independent samples t-tests were carried out for the previously mentioned five scores, as well as for the deviation scores of keypress and duration.

Additionally, each phrase of the piece was attributed with a level of difficulty. The difficulty was determined by the following system: easy phrases (including 174 notes) required no change in hand positioning, medium phrases (including 90 notes) required change in hand positioning and playing an *acciaccatura*[1], while hard phrases (including 48 notes) involved chords and thus required simultaneous keypresses, each varying in keypress duration. Three average scores (one per difficulty level) were calculated for each of the four following scores: the number of correct keypresses, the number of correct key-releases, the deviation in milliseconds of the keypress compared to the original file, as well as the deviation in milliseconds of the keypress-length compared to the original file. ANOVAs were applied to determine differences between groups among the three difficulty levels.

[1]a small grace note melodically adjacent to a principal note and played simultaneously with or immediately before it (Acciaccatura, 2018)

Results

Table 1 depict the average scores and standard deviations for the four performance measures correct keypresses, correct key releases, keypress errors and key release errors split by group. A Pearson correlation coefficient was computed to assess the relationships between the previously mentioned measures (see Table 2). There were positive correlations between the variables correct keypresses and correct key releases, r = 0.66, and between correct keypresses and key release errors, r = 0.64.

	Number of correct keypresses (SD)	**Number of keypress errors** (SD)	**Number of correct key releases** (SD)	**Number of key release errors** (SD)
Control Group	62.88 (5.8)	256.13 (17.42)	38.50 (4.03)	24.38 (3.88)
Experimental Group	63.93 (5.86)	255.27 (18.27)	42.13 (4.29)	21.80 (4.57)

Table 1. Average scores and standard deviations for correct keypresses, keypress errors, correct key releases, and key release errors, split by group.

	1	2	3	4
1. Correct keypresses	-			
2. Keypress errors	0.08	-		
3. Correct key releases	0.66	0.20	-	
4. Key release error	0.64	-0.10	-0.15	-

Table 2. Correlations table including the variables correct keypresses, keypress errors, correct key releases, and key release errors.

To estimate the differences between the control group and the experimental group in the test block, composite scores were analyzed using a Welch two-sample t-test. Results showed significant differences between the composite scores of the control group (M=-63.25, SD=11.40) and the experimental group (M=-54.27, SD=10.02) conditions; t (28.89)= -2.33, p = 0.03. This indicates, that participants in the experimental group yielded significantly higher performance for the composite score compared to the control group (see Fig. 4). Additionally, a two-sample t-test was carried out for correct key releases, as this variable contained correctly played notes by counting instances in which correct keypresses were released after the correct duration. Results showed significant difference between the control group (M=38.5, SD=4.03) and the experimental group (M=42.13, SD=4.29) conditions; t (28.53)= -2.43, p = 0.02. This indicates that participants of the experimental group performed significantly better regarding the correct duration of correct keypresses compared to the control group.

Moreover, two-sample t-tests were carried out for scores of correct keypresses, keypress errors and key release errors. No significant differences were obtained. However, boxplots indicate trends towards higher scores in correct keypresses and lower scores in wrong key releases for the experimental group (see Fig. 5).

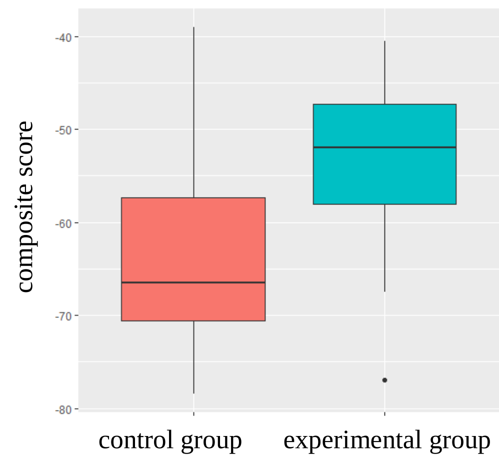


Fig. 4 Differences in the composite score between control group and experimental group.

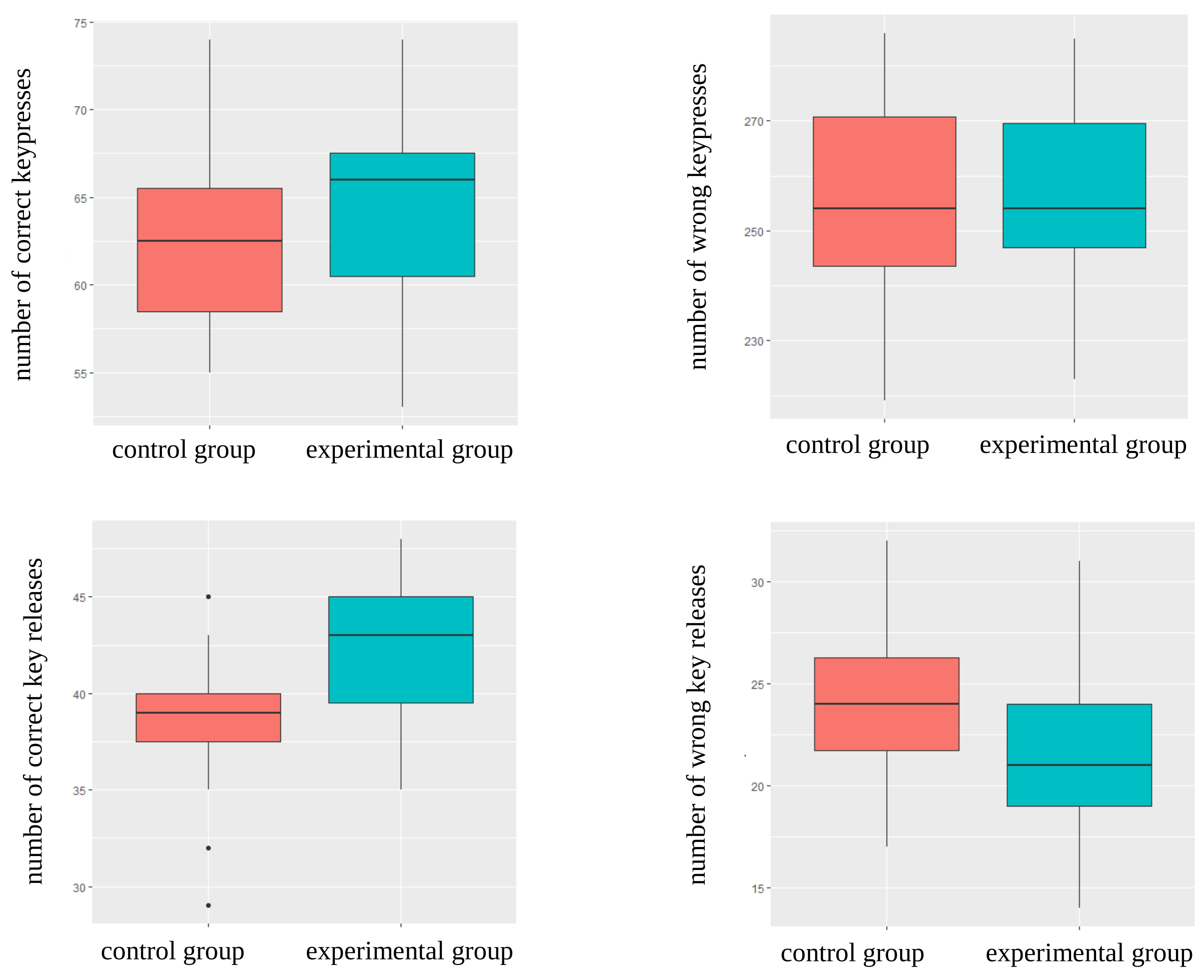


Fig. 5 Overview over the four predictive variables between control group and experimental group.

A two-sample t-test for differences among groups in the deviation of the moment the correct key was pressed, compared to the source file, showed no significant results; the same is true for the deviation in keypress length. Therefore, neither the performance of the correct keypress nor the length was significantly more accurate in one of the groups.

The scores on correct key-releases were analyzed using a 2 (control vs experimental) × 3 (difficulty level) two-way ANOVA. The three difficulty levels were easy (D1), medium (D2), and hard (D3). The multivariate result was non-significant for correct key-release, Pillai’s Trace = .20, F = 2.31, df = (27), p = .09 indicates no overall difference of the difficulty levels between the two groups. However, the univariate F-tests showed that there was significant difference between the groups for hard parts, D3, F = 5.42, p = .03, while differences for easy parts, D1, F = 3.75, p = .06 and medium parts, D2, F = 0.21, p = .07, although close to the significance level, remained non-significant.

Taken together, these results showed significantly higher performances in the experimental group regarding the execution of correctly pitched and timed notes as compared to the control group. This increase in performance was specifically observable in hard difficulty segments of the piece. No difference between the group was found regarding the number of wrong keypresses, as well as the accuracy of correct pitch and timing. Moreover, trends were observable that indicated higher scores in correct keypresses, and lower scores in the incorrect key-releases of the corresponding keypresses, although no significant results have been obtained for these measurements.

Discussion

The present study observed participants without any prior knowledge in playing instruments or reading musical notes by using a piano training software as well as response time measurement to examine whether chunking drills would benefit the acquisition of piano playing skills. Chunking exercises in general are commonly used in cognitive psychology research, for example to aid the development of automated movement sequences in the discrete sequence production task (Verwey et al., 1999). Moreover, Pike & Carter (2010) claim, that chunking drills improve performance of piano players on sight-reading. To test this claim, two learning protocols were used to answer the research question whether chunking would result in improved performance. Both protocols took 60 minutes to complete and involved an equal number of notes; they differed only in the design of the exercise: The control group practiced the full piece repeatedly, while the experimental group used chunking drills.

The present results suggest that, with the aid of Synthesia, music naive players were indeed capable of playing a simplified version of the piano piece ‘Hallelujah’ good enough, to allow for performance estimation and comparison. Significant differences between the two groups could be obtained: Participants of the experimental group scored significantly higher in both, correctly played notes as well as the composite score. This was especially true for the harder parts of the piece. Moreover, descriptive figures illustrated a trend towards more correct keypresses and less errors in the duration of keypresses for the experimental group. High correlations between the performance measures where only found between ‘correct keypresses’ and both, ‘correct key releases’ and ‘key release errors’, as the latter variables are based on ‘correct keypresses’.

Based on the observed performance improvements of the experimental group, it is suggested that the usage of modern gaming software like Synthesia combined with a MIDI keyboard is a promising method for the collection of data in the field of music research. This approach could allow for the study of music-naive participants, which would be cumbersome and immensely time consuming otherwise, as they lack the prerequisite of being able to read sheet music. Although these first observations must be further validated in future studies, Synthesia seems to provide an environment which is comparable to sight-reading exercises. An example for the further investigation of this approach could be the comparison of performance levels on recordings that are played by memory between players who previously practiced with sheet music and players who used Synthesia instead.

Furthermore, this study explored a new method of estimating piano performance based on MIDI file audio records, that are analyzed by R, a programming language that is widely used among statisticians and data analysts across various fields (Vance, 2009). Although MIDI audio records, obtained by digital pianos, are commonly used in music research, no common ground seems to be established when it comes to the performance estimation of these files. Two different approaches can be observed: On one hand, researchers use software to estimate player performances, on the other hand, manual evaluation is applied. Although specifically written programs, such as POCO (Honing, 1990) or FTAP (Finnley, 2001) exist to control musical experiments and to collect data in millisecond resolution, it seems that researchers rarely apply those methods. This might be due to the fact, that these open programs are relatively old, but no definitive answers could be found in the reviewed literature. If response times are measured, a tendency of using self-written software without providing specific information over the program can be observed (Duke et al. 2011). Contrary to the previous approach of using software, most commonly applied performance estimation still seems to be based on manual evaluation of the obtained MIDI audio recordings. This is executed by professionals which is described by Miksza (2007, p. 363) as “intensive judging duties” (for more examples, see: Demorest, 1998; McPherson, 2005, Pike, 2010). The present study opens new ways for response time measurement in music research as the used method bypasses additional programs and uses Rscript to analyze obtained MIDI files. As R is a frequently used, platform-independent, open, and accessible programming language, the usage of R seems to be a promising opportunity for musical research to establish a common ground for performance estimation. Packages such as the presently used ‘signal’ and ‘tuneR’ allow for the convenient collection and modification of huge quantities of data (see Appendix). Additionally, consequent in-depth analyses can be carried out in R, which makes further analysis software redundant.

However, the present study also comes with limitations in task, method, and data analysis that are discussed in the following paragraphs. First of all, the average amount of keypress errors was very high, with 255,7 on average, and stood in sharp contrast to an average of 63,4 correct keypresses. Although high error rates were expected previously, as observed participants were completely naive in the field of music, the scores might hint at a too high task difficulty. For further research, it is advised to choose a commonly used keyboard beginner exercise, rather than a popular musical piece, as the latter could be too difficult to execute for a beginner. The usage of a task that is friendlier towards complete beginners is expected to lead to less errors so that the usually applied performance estimation must not be modified (Henry and Demorest, 1994). Moreover, this could also facilitate the error discrimination due to a lower complexity. Additionally, it is suggested to set the task duration considerably longer than 60 minutes of practice.

A second limitation of the study is, that no significant differences could be obtained for ‘correct keypresses’ and ‘keypress errors’, as well as for ‘key release errors’ (see Table 1).

Although visible trends were in favor of the assumption that the experimental group would execute more ‘correct keypresses’, and would produce less ‘key release errors’, significant evidence is lacking (see Fig. 5). It is assumed, that visible trends would become significant if a task was chosen that is friendlier towards complete beginners and if the duration of the task was extended. For now, further research is needed to prove or refute this claim. However, these suggested improvements would not necessarily answer the question, why the amount of ‘keypress errors’ remained stagnant across groups. The current program is limited in this context as it does not discriminate between different types of ‘keypress errors’. Future programs should therefore differentiate between ‘timing keypress errors’ and ‘pitch keypress errors’ to allow for a better understanding of the mistakes, participants make.

In addition to the lack of confirmation for two of the three performance variables and the high error rates, the program is limited in how difficulty levels are treated. Whenever keypresses were correct in the timing of the key release, the difficulty of the corresponding note was assessed and added to a separate score (see Appendix). Due to the low overall score of correct key releases, the average scores for correctly played ‘easy’, ‘medium’, and ‘hard’ notes were relatively low. Although significant differences for hard parts could be obtained, there are more suitable approaches than merely comparing the difficulty level scores among groups. An interesting way to determine the impact of difficulty would be to define the difficulty level as predictor of the composite score. Moreover, interaction effects could give insight into how other variables such as age or gender could interact with the difficulty level on the estimated performance. However, before more valid analysis can be applied, the strategy of assigning difficulty levels to music segments must be reevaluated. The present study for example found no significant effect for medium parts, but a small trend towards worse performance of the experimental group. This could be explained by the fact, that those parts involved a change in hand positioning, although the difficulty of changing the hand positioning might be caused by independent factors such as hand size and not affected by the benefits of chunking strategies. To solve this issue, changing the task to established keyboard exercises is recommended for several reasons: Firstly, a popular piece, like the presently used one, often involves segments of varying and alternating degrees of difficulty, which can easily lead to wrong assessments of difficulty levels. Secondly, it is easier to determine the difficulty of traditional exercises, as these are commonly designed for a specific skill level. Lastly, the difficulty of well-established standard exercises can be more validly assessed based on previous research and theories.

Finally, it must be stated, that there are other approaches to analyze the data. The presently used Shapiro-Wilks tests, Welch t-tests and ANOVAs are grounded in statistical hypothesis testing and thus assuming, that the null hypothesis is only rejected if the resulting p-value is less than the selected probability threshold of, in this case, 5%. Null hypothesis significance testing (NHST) has been the topic of a continuing debate (for example, Nickerson, 2000, Branch, 2014). The core argument is, that while it can provide critical information, statistical significance does not automatically imply practical significance and correlation does not imply causation. It is thus stated, that casting doubt on the null hypothesis can be easily misunderstood as directly supporting the research hypothesis. Therefore, using p-values is claimed to be ineffective in ensuring the replicability of social sciences (Open Science Collaboration, 2015). Bayesian inference could be used to gain deeper insight into the data, as the research question is quantitative and explorative in nature. As an example, a Bayesian generalized linear regression model could be used to express the dependency between the predictors ‘control group’ and ‘experimental group’ and the outcome variable ‘correctly released keys’ by performing a parameter estimation using the method of Markov-Chain Monte-Carlo sampling (Muth et. al, 2018). The estimates would then express how much better the chunking drill is in terms of the average number of keys that participants of the experimental group correctly released more of, as compared to the control group.

In conclusion, the present study supports the hypothesis that chunking drills do indeed benefit the acquisition of piano playing skills, as participants of the experimental group scored higher in regards of correctly played notes as well as the composite score. However, further research is needed to validate the method and to illustrate a clearer picture of the benefits and boundaries it brings to the table. For now, the chunking-based software piano training seems to be a promising approach to develop practical applications for the piano skill acquisition, for example by providing real time feedback within the learning software so that players can efficiently target their weaknesses. Based on the present experiment, it is recommended to further investigate chunking methods in the context of musical skill acquisition as recent results seem promising and only few studies exist in this line of research. If the limitations of this study are tackled and deeper understanding of the effects of chunking methods is obtained, future research and traditional piano training methods could be combined to design adaptive training programs. Such programs could then constantly measure performances and use algorithms to continually change the difficulty of the task to address the unique needs of each learner. Further developing the present method by taking both, the previously mentioned limitations as well as the research suggestions into account could thus lead to an exciting, modern way to facilitate the acquisition of music skill in the future.

References

Abrahamse, E., Ruitenberg, M., De Kleine, E., & Verwey, W. (2013). Control of automated behavior: insights from the discrete sequence production task. *Frontiers in human neuroscience*, *7*, 1-16.

Acciaccatura (2018). *Collins English dictionary*. Glasgow: HarperCollins Publishers. Retrieved from https://www.collinsdictionary.com/

Bach, C. P. E. (1753). *Versuch über die wahre Art das Clavier zu spielen, mit Exempeln und achtzehn Probe-Stücken in sechs Sonaten*. Berlin, Author. Retrieved from: https://books.google.de/books?id=jYtZAAAAcAAJ

Barry, N. & Hallam, S. (2002) 'Practicing', in Parncutt, R., & McPherson, G. (Eds.). (2002). *The science and psychology of music performance: Creative strategies for teaching and learning*. Oxford University Press.

Barry, N. H., & McArthur, V. (1994). Teaching practice strategies in the music studio: A survey of applied music teachers. *Psychology of Music*, 22(1), 44-55.

Branch, M. (2014). Malignant side effects of null-hypothesis significance testing. *Theory & Psychology*, *24*(2), 256-277.

Drake, C., & Palmer, C. (2000). Skill acquisition in music performance: Relations between planning and temporal control. *Cognition*, 74, 1–32.

Duke, R. A., Cash, C. D., & Allen, S. E. (2011). Focus of attention affects performance of motor skills in music. *Journal of Research in Music Education*, *59*(1), 44-55.

Every Child Achieves Act (2015). Senate 1177, 114th Cong. Retrieved from http://www.help.senate.gov.

Finney, S. A. (2001). FTAP: A Linux-based program for tapping and music experiments. *Behavior Research Methods, Instruments, & Computers*, 33(1), 65-72.

Gallistel, C. (1980). *The organization of action: A new synthesis.* Hillsdale, NJ: Erlbaum.

Gilman, B., & Underwood, G. (2003). Restricting the field of view to investigate the perceptual span of pianists. *Visual Cognition*, 10(2), 201–232.

Gruson, L. M. (1988). Rehearsal skill and musical competence: Does practice make perfect? *Generative processes in music: The psychology of performance, improvisation, and composition*, 91-112.

Gudmundsdottir, H. (2008). *Development of pitch reading skills in young piano students.* Paper presented at the international conference of the International Society for Music Education, Bologna, Italy.

Halford, G., Wilson, W., & Phillips, S. (1998). Processing capacity defined by relational complexity: Implications for comparative, developmental, and cognitive psychology. *Behavioral and Brain Sciences, 21*(06), 803-831.

Hatfield, J. L., Halvari, H., & Lemyre, P. N. (2016). Instrumental practice in the contemporary music academy: A three-phase cycle of Self-Regulated Learning in music students. *Musicae Scientiae*, 1029864916658342.

Hommel, B., Müsseler, J., Aschersleben, G., Prinz, W. (2001) The Theory of Event Coding (TEC): A framework for perception and action planning. *Behavioral and Brain Sciences,24*(5), 849-878.

Honing, H. (1990). POCO: An Environment for Analysing, Modifying and Generating Expression in Music. In *ICMC*.

Kohut, D. L. (1985). *Musical performance: Learning theory and pedagogy*. Prentice Hall.

Lifelong Learning Program (2006). *European Union Programs Agency*. Retrieved from http://ec.europa.eu.

McNevin, N. H., Shea, C. H., & Wulf, G. (2003). Increasing the distance of an external focus of attention enhances learning. *Psychological research*, *67*(1), 22-29.

McPherson, G. E. (2005). From child to musician: Skill development during the beginning stages of learning an instrument. *Psychology of music*, *33*(1), 5-35.

Miksza, P. (2007). Effective practice an investigation of observed practice behaviors, self-reported practice habits, and the performance achievement of high school wind players. *Journal of Research in Music Education*, *55*(4), 359-375.

Muth, C., Oravecz, Z., & Gabry, J. (2018). User-friendly Bayesian regression modeling: A tutorial with rstanarm and shinystan. *The Quantitative Methods for Psychology*, *14*(2), 99-119.

Nickerson, R. S. (2000). Null hypothesis significance testing: a review of an old and continuing controversy. *Psychological methods*, *5*(2), 241.

Open Science Collaboration. (2015). Estimating the reproducibility of psychological science. *Science*, *349*(6251).

Palmer, C. (1997). Music performance. *Annual review of psychology*, *48*(1), 115-138.

Pew, R. W. (1966). Acquisition of hierarchical control over the temporal organization of a skill. *Journal of experimental psychology*, *71*(5), 764.

Pike, P. D., & Carter, R. (2010). Employing cognitive chunking techniques to enhance sight-reading performance of undergraduate group-piano students. *International Journal of Music Education*, *28*(3), 231-246.

Qualifications and Curriculum Authority (1999). *The review of the National Curriculum in England: The consultation materials*. London: QCA.

Swanwick, K. (2002). *A basis for music education*. Routledge.

Vance, A. (2009). "Data Analysts Captivated by R's Power". *New York Times*. Retrieved 2018-06-18.

Verwey, W. B. (1996). Buffer loading and chunking in sequential keypressing. *Journal of Experimental Psychology: Human Perception and Performance*, *22*(3), 544.

Verwey, W. B. (1999). Evidence for a multistage model of practice in a sequential movement task. *Journal of Experimental Psychology-Human Perception and Performance*, 25(6), 1693-1708.

Verwey, W. B., Abrahamse, E. L., & De Kleine, E. (2010). Cognitive processing in new and practiced discrete keying sequences. *Frontiers in psychology*, *1*, 32.

Verwey, W., Shea, C., Wright, D. (2015) A cognitive framework for explaining serial processing and sequence execution strategies. *Psychon Bull Rev*, *22*(1), 54–77.

Vidal, F., Meckler, C., & Hasbroucq, T. (2015). Basics for sensorimotor information processing: some implications for learning. *Frontiers in psychology*, *6*.

White, J. D. (1994). *Comprehensive musical analysis*. Scarecrow Press.

Appendix

The following code was written for RStudio to read the recorded MIDI-files, transform them into workable data and to describe them. They are compared to the original file to estimate the participants performance. Afterwards, the analysis is carried out. Important results are printed bold and marked in red. The first section installs the required packages and reads the libraries, used for this Rscript.

```
install.packages("C:/Users/Graceman/Desktop/R Project/signal_0.7-6.tar.gz", repos =
NULL, type = "source", lib="C:/Program Files/R/R-3.3.2/library")
install.packages("C:/Users/Graceman/Desktop/R Project/tuneR_1.3.2.tar.gz", repos =
NULL, type = "source", lib="C:/Program Files/R/R-3.3.2/library")

libs = c('readr','data.table','NLP','psych','signal','tuneR','nlme','knitr')
invisible(lapply(libs, library, character.only=TRUE, quietly=TRUE,
warn.conflicts=FALSE))
```

The following code section is used to create a Dataframe and to read in the data by looping through all MidiFiles. During this process a dynamic output message is presented to inform the user about the progress in percentage and the estimated time that is left.

```
CompleteDataSetL2PM<-
c("Filepath","ParticipantNumber","Group","CompleteCases","KeypressCorrect","KeypressE
rror","KeypressDeviationMS","KeyreleaseCorrect","KeyreleaseError","KeypressLengthDevi
ationMS","notecorrectEasy","notecorrectMedium","notecorrectHard","KeypressDeviationMS
Easy","KeypressDeviationMSMedium","KeypressDeviationMSHard","KeyreleaseCorrectEasy","
KeyreleaseCorrectMedium","KeyreleaseCorrectHard","KeypressLengthDeviationMSEasy","Key
pressLengthDeviationMSMedium","KeypressLengthDeviationMSHard")

# Loading Data from files
filelist <- dir("Midis", pattern=NULL, all.files=FALSE,full.names=FALSE)

# correct play, to compare the rest to
DataOriginalFile<-getMidiNotes(readMidi(paste("Midis/",filelist[1],sep = "")))
DataOriginalFile$time<-DataOriginalFile$time - DataOriginalFile$time[1]
DataOriginalFile$time<-DataOriginalFile$time/800*1.66666 #800 to convert miditime (of
GuitarPro) to realseconds
DataOriginalFile$length<-DataOriginalFile$length/800*1.66666 #800 to convert miditime
(of GuitarPro) to realseconds
#1.66 to convert the original play to slow play

difficultycats<-
c(1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,1,1,1,1,1,1,1,1,1
,1,1,1,1,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,2,2,2,2,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,3,3,3,3,3,3,3
,3,3,3,3,3,3,3,3,3,3,1,1,1,1,1,1,1,1,1,1,1,1,1,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,
2,2,2,2,2,2,2,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,2,2,2,2,1,1,1,1
,1,1,1,1,1,1,1,1,1,1,1,1,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,1,1,1,1,1,1,1,1,1,1,1,1,1,
2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
```

```
,1,1,1,1,1,1,1,1,1,1,1,2,2,2,2)
DataOriginalFile<-cbind(DataOriginalFile,difficultycats)

## Loop through all data
# Start the clock
start_time <- Sys.time()

loopcounter <-1
for (rowdata in 2: length(filelist)) {
  loopcounter <- loopcounter+1
  ## andere separatoren zu nutzen?! oder formate der datensets angleichen?!
  dataparticipantplay<-getMidiNotes(readMidi(paste("Midis/",filelist[loopcounter],sep
= "")))
  dataparticipantplay$time<-dataparticipantplay$time - dataparticipantplay$time[1]
  dataparticipantplay$time<-dataparticipantplay$time*0.002717 #0.002717 to convert
miditime (of Synthesia) to realseconds
  dataparticipantplay$length<-dataparticipantplay$length*0.002717 #0.002717 to
convert miditime (of Synthesia) to realseconds

  # Process Data into workable Dataset >>=
  notecorrect   <-0
  noteerror     <-0
  KeyreleaseCorrect <-0
  KeyreleaseError   <-0
  events        <-0
  KeypressDeviationMS        <-0
  KeypressLengthDeviationMS  <-0
  notecorrectondifficulty <-c(0,0,0)
  KeypressDeviationMSdifficulty        <-c(0,0,0)
  KeyreleaseCorrectdifficulty <-c(0,0,0)
  KeypressLengthDeviationKeypressDeviationMSiculty  <-c(0,0,0)

  # loop pro index of correct notes...
  for (rowloopcorrect in 1:nrow(DataOriginalFile)) {
    # create interval for accepting participant played notes
    # Determine interval in which a played note is accepted as correctly played
    intervalMargin <- 0.5   ## can be altered!
    intervalmitte   <- DataOriginalFile$time[rowloopcorrect]
    intervalminimum <- intervalmitte - intervalMargin
    intervalmaximum <- intervalmitte + intervalMargin

    # Determine interval in which a key release is accepted as correctly timed
    intervalMargin2 <- 0.5   ## can be altered!
    intervalmitte2   <- DataOriginalFile$length[rowloopcorrect]
    intervalminimum2 <- intervalmitte2 - intervalMargin2
    intervalmaximum2 <- intervalmitte2 + intervalMargin2

    # determine indeces of notes within correct interval
    indecescorrectnoteplayed<-which(dataparticipantplay$time>intervalminimum &
dataparticipantplay$time<intervalmaximum)
    # find where participant played correct note(s)
    for (rowrow in 1:length(indecescorrectnoteplayed)){

      ii<-indecescorrectnoteplayed[rowrow]
```

```
if(identical(DataOriginalFile$notename[rowloopcorrect],dataparticipantplay$notename[i
i])){
        notecorrect <- notecorrect + 1
        notecorrectondifficulty[difficultycats[rowloopcorrect]] <-
notecorrectondifficulty[difficultycats[rowloopcorrect]]+1
        KeypressDeviationMS <-
KeypressDeviationMS+round(abs(DataOriginalFile$time[rowloopcorrect]-
dataparticipantplay$time[ii]), digits = 2)

        KeypressDeviationMSdifficulty[difficultycats[rowloopcorrect]] <-
KeypressDeviationMSdifficulty[difficultycats[rowloopcorrect]]+
          round(abs(DataOriginalFile$time[rowloopcorrect]-
dataparticipantplay$time[ii]), digits = 2)


        if(dataparticipantplay$length[ii]>intervalminimum2 &
dataparticipantplay$length[rowloopcorrect]<intervalmaximum2){
          KeyreleaseCorrect <- KeyreleaseCorrect + 1
          KeyreleaseCorrectdifficulty[difficultycats[rowloopcorrect]] <-
KeyreleaseCorrectdifficulty[difficultycats[rowloopcorrect]]+1

          KeypressLengthDeviationMS<-
KeypressLengthDeviationMS+round(abs(DataOriginalFile$length[rowloopcorrect] -
dataparticipantplay$length[ii]), digits = 2)

KeypressLengthDeviationKeypressDeviationMSiculty[difficultycats[rowloopcorrect]]<-
KeypressLengthDeviationKeypressDeviationMSiculty[difficultycats[rowloopcorrect]]+roun
d(abs(DataOriginalFile$length[rowloopcorrect] - dataparticipantplay$length[ii]),
digits = 2)

        }else{
          KeyreleaseError <- KeyreleaseError + 1
        }

        break # ends the loop to prevent double scoring
      }
    }
  }
  ################ END Note/Error Loops

  noteerror<-sum(complete.cases(dataparticipantplay))-notecorrect
  cc <- sum(complete.cases(dataparticipantplay))
  # make Group variable
  filename<-paste(filelist[rowdata])
  firstsplit<-strsplit(filename,"_")
  firstsplit<-unlist(firstsplit)
  firstsplit[2]
  secondsplit<-strsplit(firstsplit[2],"\\.")
  secondsplit<-unlist(secondsplit)
  Group<-secondsplit[1]

  newrow<-c(paste("Midis/",filelist[rowdata]),rowdata-
1,Group,cc,notecorrect,noteerror,KeypressDeviationMS,KeyreleaseCorrect,KeyreleaseErro
r,KeypressLengthDeviationMS,notecorrectondifficulty[1],notecorrectondifficulty[2],not
ecorrectondifficulty[3],KeypressDeviationMSdifficulty[1],KeypressDeviationMSdifficult
```

```
y[2],KeypressDeviationMSdifficulty[3],KeyreleaseCorrectdifficulty[1],KeyreleaseCorrec
tdifficulty[2],KeyreleaseCorrectdifficulty[3],KeypressLengthDeviationKeypressDeviatio
nMSiculty[1],KeypressLengthDeviationKeypressDeviationMSiculty[2],KeypressLengthDeviat
ionKeypressDeviationMSiculty[3])


  # create new row ready for insertion into the dataset
  CompleteDataSetL2PM <- rbind(CompleteDataSetL2PM,newrow) # inserting...

  # Stop the clock
  end_time = Sys.time()
  time_diff       <- (round(as.numeric(difftime(time1 = end_time, time2 = start_time,
units = "secs")), 3))
  loops_left      <- length(filelist) - loopcounter
  avg_looptime    <- time_diff/loopcounter
  loop_duration   <- avg_looptime*length(filelist)
  time_left       <- round(loops_left*avg_looptime,digits=0)
  percentage_done <- 100+(round((loopcounter/length(filelist)-1)*100, digits=0))

  if(loopcounter>4)
  {
    cat('\r', paste('Midi Files are getting analyzed! Estimated time left:
',time_left, ' seconds. Progress:',percentage_done,'%'))
    flush.console()
  } else
  {
    cat('\r', paste('Time left is calculated. Progress:',percentage_done,' %'))
    flush.console()
  }

}# from for loop after loading correct play data

 Time left is calculated. Progress: 6  %
 Time left is calculated. Progress: 9  %
 Time left is calculated. Progress: 12  %
 Midi Files are getting analyzed! Estimated time left:  22  seconds. Progress: 15 %
 Midi Files are getting analyzed! Estimated time left:  22  seconds. Progress: 18 %
 Midi Files are getting analyzed! Estimated time left:  21  seconds. Progress: 21 %
 Midi Files are getting analyzed! Estimated time left:  21  seconds. Progress: 24 %
 Midi Files are getting analyzed! Estimated time left:  21  seconds. Progress: 27 %
 Midi Files are getting analyzed! Estimated time left:  20  seconds.
```

.
.
.
.

The calculation runs through

```
 Midi Files are getting analyzed! Estimated time left:  3  seconds. Progress: 91 %
 Midi Files are getting analyzed! Estimated time left:  2  seconds. Progress: 94 %
 Midi Files are getting analyzed! Estimated time left:  1  seconds. Progress: 97 %
 Midi Files are getting analyzed! Estimated time left:  0  seconds. Progress: 100 %
```

```
# complete data collection by making first row column names, then deleting first row
colnames(CompleteDataSetL2PM) = CompleteDataSetL2PM[1, ] # the first row will be the
header
CompleteDataSetL2PM = CompleteDataSetL2PM[-1, ]

  diff_time <- round(Sys.time() - start_time, 4)
  print(paste('Binding datarows to Dataframe accomplished! Total Duration:
',time_diff,' seconds!' ) )

## [1] "Binding datarows to Dataframe accomplished! Total Duration:  31.32  seconds!"

  print(CompleteDataSetL2PM)
```

The following output shows the first 6 rows of the print-command (in total 32 rows). The Dataset involves: the file path of the Midi file, participant number and group (where “c” stands for control group and “e” for experimental group), the total amount of recorded events, the number of correct and wrong keypresses, the total deviation in milliseconds between the moment of the keypress and the moment that it should have been pressed according to the original file, the number of correct and wrong key-releases, the deviation of milliseconds between the keypress-length and the length that it should have been pressed according to the original file, as well as the number of correct keypresses and key-releases per difficulty, and the deviations in milliseconds for both, keypress and length, per difficulty.

```
##        Filepath           ParticipantNumber Group CompleteCases
## newrow "Midis/ p1_c.mid"  "1"               "c"   "297"
## newrow "Midis/ p10_e.mid" "2"               "e"   "314"
## newrow "Midis/ p11_e.mid" "3"               "e"   "354"
## newrow "Midis/ p12_e.mid" "4"               "e"   "315"
## newrow "Midis/ p13_c.mid" "5"               "c"   "301"
## newrow "Midis/ p14_c.mid" "6"               "c"   "309"



KeypressCorrect KeypressError KeypressDeviationMS KeyreleaseCorrect
## newrow "56"            "241"         "14.32"             "32"
## newrow "67"            "247"         "16.39"             "46"
## newrow "69"            "285"         "16.01"             "40"
## newrow "63"            "252"         "14.76"             "43"
## newrow "61"            "240"         "13.36"             "38"
## newrow "65"            "244"         "16.07"             "45"

##        KeyreleaseError KeypressLengthDeviationMS KeypressCorrectEasy
## newrow "24"            "8.9"                     "33"
## newrow "21"            "11.83"                   "45"
## newrow "29"            "9.52"                    "44"
## newrow "20"            "12.62"                   "42"
## newrow "23"            "12.17"                   "39"
## newrow "20"            "18.38"                   "44"
```

```
##         KeypressCorrectMedium KeypressCorrectHard KeypressDeviationMSEasy
## newrow "13"                  "10"                "8.45"
## newrow "12"                  "10"                "10.39"
## newrow "13"                  "12"                "10.78"
## newrow "11"                  "10"                "9.09"
## newrow "11"                  "11"                "8.57"
## newrow "11"                  "10"                "10.94"

##         KeypressDeviationMSMedium KeypressDeviationMSHard
## newrow "3.27"                    "2.6"
## newrow "3.57"                    "2.43"
## newrow "3"                       "2.23"
## newrow "2.46"                    "3.21"
## newrow "2.81"                    "1.98"
## newrow "2.69"                    "2.44"

##         KeyreleaseCorrectEasy KeyreleaseCorrectMedium KeyreleaseCorrectHard
## newrow "21"                  "9"                     "2"
## newrow "32"                  "10"                    "4"
## newrow "27"                  "6"                     "7"
## newrow "30"                  "7"                     "6"
## newrow "27"                  "8"                     "3"
## newrow "32"                  "7"                     "6"

##         KeypressLengthDeviationMSEasy KeypressLengthDeviationMSMedium
## newrow "6.99"                        "1.8"
## newrow "6.82"                        "3.61"
## newrow "6.81"                        "1.98"
## newrow "7.34"                        "3.8"
## newrow "7.51"                        "4.49"
## newrow "11.35"                       "5.04"

##         KeypressLengthDeviationMSHard
## newrow "0.11"
## newrow "1.4"
## newrow "0.73"
## newrow "1.48"
## newrow "0.17"
## newrow "1.99"
```

This section prepares the data and excludes outliers.

```
#transforming data, excluding outliers
dfdata<-data.frame(CompleteDataSetL2PM)

dfdata$CompleteCases <- as.numeric(CompleteDataSetL2PM[,4])
dfdata$KeypressCorrect <- as.numeric(CompleteDataSetL2PM[,5])
dfdata$KeypressError <- as.numeric(CompleteDataSetL2PM[,6])
dfdata$KeypressDeviationMS <- as.numeric(CompleteDataSetL2PM[,7])
dfdata$KeyreleaseCorrect <- as.numeric(CompleteDataSetL2PM[,8])
dfdata$KeyreleaseError <- as.numeric(CompleteDataSetL2PM[,9])
dfdata$KeypressLengthDeviationMS <- as.numeric(CompleteDataSetL2PM[,10])
```

```
dfdata$notecorrectEasy <- as.numeric(CompleteDataSetL2PM[,11])
dfdata$notecorrectMedium <- as.numeric(CompleteDataSetL2PM[,12])
dfdata$notecorrectHard <- as.numeric(CompleteDataSetL2PM[,13])
dfdata$KeypressDeviationMSEasy <- as.numeric(CompleteDataSetL2PM[,14])
dfdata$KeypressDeviationMSMedium <- as.numeric(CompleteDataSetL2PM[,15])
dfdata$KeypressDeviationMSHard <- as.numeric(CompleteDataSetL2PM[,16])
dfdata$KeyreleaseCorrectEasy <- as.numeric(CompleteDataSetL2PM[,17])
dfdata$KeyreleaseCorrectMedium <- as.numeric(CompleteDataSetL2PM[,18])
dfdata$KeyreleaseCorrectHard <- as.numeric(CompleteDataSetL2PM[,19])
dfdata$KeypressLengthDeviationMSEasy <- as.numeric(CompleteDataSetL2PM[,20])
dfdata$KeypressLengthDeviationMSMedium <- as.numeric(CompleteDataSetL2PM[,21])
dfdata$KeypressLengthDeviationMSHard <- as.numeric(CompleteDataSetL2PM[,22])

#descriptives table
print(paste('Before exclusion of outliers:'))

## [1] "Before exclusion of outliers:"

describeBy(dfdata, Group="Group",digits=2)

##                                 vars  n   mean    sd median trimmed   mad
## Filepath*                          1 32  16.50  9.38  16.50   16.50 11.86
## ParticipantNumber*                 2 32  16.50  9.38  16.50   16.50 11.86
## Group*                             3 32   1.50  0.51   1.50    1.50  0.74
## CompleteCases                      4 32 322.56 27.02 318.50  320.04 19.27
## KeypressCorrect                    5 32  63.84  6.22  64.00   63.65  6.67
## KeypressError                      6 32 258.72 24.24 254.00  257.27 18.53
## KeypressDeviationMS                7 32  15.20  1.61  15.10   15.17  1.79
## KeyreleaseCorrect                  8 32  40.38  4.46  40.00   40.58  5.19
## KeyreleaseError                    9 32  23.47  4.70  23.00   23.27  4.45
## KeypressLengthDeviationMS         10 32  11.45  2.37  10.93   11.29  2.16
## notecorrectEasy                   11 32  41.44  3.86  42.00   41.58  4.45
## notecorrectMedium                 12 32  11.66  1.88  11.00   11.58  2.22
## notecorrectHard                   13 32  10.75  2.29  10.00   10.73  2.97
## KeypressDeviationMSEasy           14 32   9.53  1.05   9.57    9.56  1.31
## KeypressDeviationMSMedium         15 32   2.96  0.55   2.99    3.00  0.62
## KeypressDeviationMSHard           16 32   2.71  0.63   2.54    2.71  0.79
## KeyreleaseCorrectEasy             17 32  27.88  3.23  28.50   28.08  2.97
## KeyreleaseCorrectMedium           18 32   7.72  1.67   8.00    7.69  1.48
## KeyreleaseCorrectHard             19 32   4.78  2.21   4.50    4.69  2.22
## KeypressLengthDeviationMSEasy     20 32   7.33  1.60   7.21    7.21  1.22
## KeypressLengthDeviationMSMedium   21 32   2.96  1.06   2.77    2.92  0.85
## KeypressLengthDeviationMSHard     22 32   1.16  0.71   1.00    1.12  0.71

##                                    min    max  range  skew kurtosis   se
## Filepath*                         1.00  32.00  31.00  0.00    -1.31 1.66
## ParticipantNumber*                1.00  32.00  31.00  0.00    -1.31 1.66
## Group*                            1.00   2.00   1.00  0.00    -2.06 0.09
## CompleteCases                   279.00 430.00 151.00  1.81     5.41 4.78
## KeypressCorrect                  53.00  78.00  25.00  0.17    -0.67 1.10
## KeypressError                   219.00 352.00 133.00  1.54     4.54 4.28
## KeypressDeviationMS              11.76  18.66   6.90  0.07    -0.72 0.28
## KeyreleaseCorrect                29.00  48.00  19.00 -0.44    -0.35 0.79
## KeyreleaseError                  14.00  34.00  20.00  0.34    -0.51 0.83
## KeypressLengthDeviationMS         6.73  18.38  11.65  0.69     0.54 0.42
```

```
## notecorrectEasy                   33.00  48.00  15.00 -0.28    -0.91 0.68
## notecorrectMedium                  8.00  17.00   9.00  0.43     0.48 0.33
## notecorrectHard                    6.00  15.00   9.00  0.11    -0.75 0.40
## KeypressDeviationMSEasy            7.62  11.47   3.85 -0.08    -1.05 0.19
## KeypressDeviationMSMedium          1.74   3.80   2.06 -0.39    -0.64 0.10
## KeypressDeviationMSHard            1.46   3.79   2.33  0.07    -1.11 0.11
## KeyreleaseCorrectEasy             21.00  33.00  12.00 -0.53    -0.67 0.57
## KeyreleaseCorrectMedium            5.00  12.00   7.00  0.19    -0.27 0.30
## KeyreleaseCorrectHard              1.00  10.00   9.00  0.34    -0.81 0.39
## KeypressLengthDeviationMSEasy      4.72  11.35   6.63  0.65     0.14 0.28
## KeypressLengthDeviationMSMedium    0.90   5.25   4.35  0.38    -0.52 0.19
## KeypressLengthDeviationMSHard      0.11   3.05   2.94  0.58    -0.24 0.13

#exclude outliers
mean<-mean(dfdata$KeypressCorrect)
sd<-sd(dfdata$KeypressCorrect)
intervalmin <- mean - sd*3
intervalmax <- mean + sd*3
counter <- 1
amountofoutliers <-0

for(i in dfdata$KeypressCorrect){
  if(i < intervalmin | i > intervalmax){
    dfdata<-dfdata[-c(counter), ]
    amountofoutliers <- amountofoutliers + 1
  }
  counter <- counter+1
}

mean<-mean(dfdata$KeypressError)
sd<-sd(dfdata$KeypressError)
intervalmin <- mean - sd*3
intervalmax <- mean + sd*3
counter <- 1

for(i in dfdata$KeypressError){
  if(i < intervalmin | i > intervalmax){
   dfdata<-dfdata[-c(counter), ]
   amountofoutliers <- amountofoutliers + 1
  }
  counter <- counter+1
}

mean<-mean(dfdata$KeyreleaseCorrect)
sd<-sd(dfdata$KeyreleaseCorrect)
intervalmin <- mean - sd*3
intervalmax <- mean + sd*3
counter <- 1

for(i in dfdata$KeyreleaseCorrect){
  if(i < intervalmin | i > intervalmax){
    dfdata<-dfdata[-c(counter), ]
    amountofoutliers <- amountofoutliers + 1
  }
```

```
  counter <- counter+1
}

mean<-mean(dfdata$KeyreleaseError)
sd<-sd(dfdata$KeyreleaseError)
intervalmin <- mean - sd*3
intervalmax <- mean + sd*3
counter <- 1

for(i in dfdata$KeyreleaseError){
  if(i < intervalmin | i > intervalmax){
    dfdata<-dfdata[-c(counter), ]
    amountofoutliers <- amountofoutliers + 1
  }
  counter <- counter+1
}

#calculate composite score based on KeyreleaseCorrect (which stands for correctly
played notes as it involves correct pitch and timing)
dfdata$compositescore <- dfdata$KeyreleaseCorrect*2 - dfdata$KeypressError*0.5 -
dfdata$KeyreleaseError *0.5

#exclude outliers
mean<-mean(dfdata$compositescore)
sd<-sd(dfdata$compositescore)
intervalmin <- mean - sd*3
intervalmax <- mean + sd*3
counter <- 1

for(i in dfdata$compositescore){
  if(i < intervalmin | i > intervalmax){
    dfdata<-dfdata[-c(counter), ]
    amountofoutliers <- amountofoutliers + 1
  }
  counter <- counter+1
}
print(paste('Number of excluded Outliers:',amountofoutliers))

## [1] "Number of excluded Outliers: 1"
```

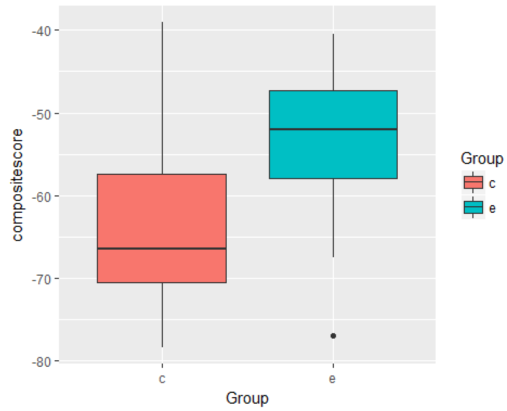
For the descriptive section, mean, standard deviation, median, minimum, and maximum values as well as the range of all relevant variables are calculated. Additionally, average scores and standard deviations for correct keypresses, correct key releases, keypress errors and key release errors were displayed, split by group.

```
describeBy(dfdata, dfdata$Group,digits=2)

##                                 vars  n   mean    sd median trimmed   mad
## CompleteCases                      4 31 319.10 18.90 318.00  318.84 19.27
## KeypressCorrect                    5 31  63.39  5.75  64.00   63.32  5.93
## KeypressError                      6 31 255.71 17.54 254.00  256.44 17.79
## KeypressDeviationMS                7 31  15.09  1.50  15.00   15.09  1.69
## KeyreleaseCorrect                  8 31  40.26  4.49  40.00   40.44  4.45
## KeyreleaseError                    9 31  23.13  4.36  23.00   22.96  4.45
## KeypressLengthDeviationMS         10 31  11.48  2.40  11.12   11.32  2.22
## notecorrectEasy                   11 31  41.23  3.73  42.00   41.40  4.45
## notecorrectMedium                 12 31  11.48  1.63  11.00   11.52  1.48
## notecorrectHard                   13 31  10.68  2.29  10.00   10.64  2.97
## KeypressDeviationMSEasy           14 31   9.47  1.01   9.56    9.51  1.23
## KeypressDeviationMSMedium         15 31   2.94  0.54   2.97    2.97  0.62
## KeypressDeviationMSHard           16 31   2.68  0.62   2.49    2.67  0.73
## KeyreleaseCorrectEasy             17 31  27.87  3.28  29.00   28.08  2.97
## KeyreleaseCorrectMedium           18 31   7.68  1.68   8.00    7.64  1.48
## KeyreleaseCorrectHard             19 31   4.71  2.21   4.00    4.60  2.97
## KeypressLengthDeviationMSEasy     20 31   7.35  1.61   7.34    7.25  1.13
## KeypressLengthDeviationMSMedium   21 31   2.95  1.08   2.77    2.91  0.86
## KeypressLengthDeviationMSHard     22 31   1.17  0.72   0.98    1.13  0.74
## compositescore                    23 31 -58.90 11.52 -58.00  -58.84 14.08


##                                    min    max range  skew kurtosis   se
## CompleteCases                   279.00 356.00 77.00  0.06    -0.59 3.39
## KeypressCorrect                  53.00  74.00 21.00 -0.02    -0.94 1.03
## KeypressError                   219.00 286.00 67.00 -0.15    -0.70 3.15
## KeypressDeviationMS              11.76  17.77  6.01 -0.09    -0.89 0.27
## KeyreleaseCorrect                29.00  48.00 19.00 -0.39    -0.38 0.81
## KeyreleaseError                  14.00  32.00 18.00  0.21    -0.60 0.78
## KeypressLengthDeviationMS         6.73  18.38 11.65  0.65     0.42 0.43
## notecorrectEasy                  33.00  47.00 14.00 -0.33    -0.93 0.67
## notecorrectMedium                 8.00  15.00  7.00 -0.15    -0.53 0.29
## notecorrectHard                   6.00  15.00  9.00  0.17    -0.69 0.41
## KeypressDeviationMSEasy           7.62  11.12  3.50 -0.14    -1.11 0.18
## KeypressDeviationMSMedium         1.74   3.80  2.06 -0.35    -0.62 0.10
## KeypressDeviationMSHard           1.46   3.79  2.33  0.12    -1.03 0.11
## KeyreleaseCorrectEasy            21.00  33.00 12.00 -0.52    -0.75 0.59
## KeyreleaseCorrectMedium           5.00  12.00  7.00  0.25    -0.24 0.30
## KeyreleaseCorrectHard             1.00  10.00  9.00  0.41    -0.72 0.40
## KeypressLengthDeviationMSEasy     4.72  11.35  6.63  0.60     0.05 0.29
## KeypressLengthDeviationMSMedium   0.90   5.25  4.35  0.39    -0.59 0.19
## KeypressLengthDeviationMSHard     0.11   3.05  2.94  0.56    -0.34 0.13
## compositescore                  -78.50 -39.00 39.50  0.02    -1.17 2.07

aggregate(dfdata$KeypressCorrect, by=list(dfdata$Group), FUN=mean)
aggregate(dfdata$KeypressCorrect, by=list(dfdata$Group), FUN=sd)

aggregate(dfdata$KeypressError, by=list(dfdata$Group), FUN=mean)
aggregate(dfdata$KeypressError, by=list(dfdata$Group), FUN=sd)

aggregate(dfdata$KeyreleaseCorrect, by=list(dfdata$Group), FUN=mean)
```

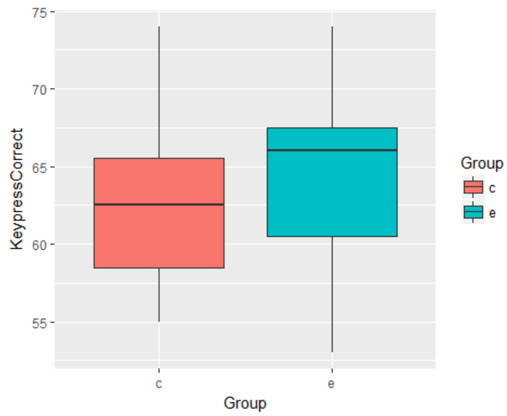
```
aggregate(dfdata$KeyreleaseCorrect, by=list(dfdata$Group), FUN=sd)

aggregate(dfdata$KeyreleaseError, by=list(dfdata$Group), FUN=mean)
aggregate(dfdata$KeyreleaseError, by=list(dfdata$Group), FUN=sd)

> aggregate(dfdata$KeypressCorrect, by=list(dfdata$Group), FUN=mean)
  Group.1        x
1       c 62.87500
2       e 63.93333
> aggregate(dfdata$KeypressCorrect, by=list(dfdata$Group), FUN=sd)
  Group.1        x
1       c 5.795113
2       e 5.861090
>
> aggregate(dfdata$KeypressError, by=list(dfdata$Group), FUN=mean)
  Group.1        x
1       c 256.1250
2       e 255.2667
> aggregate(dfdata$KeypressError, by=list(dfdata$Group), FUN=sd)
  Group.1        x
1       c 17.41599
2       e 18.26576
>
> aggregate(dfdata$KeyreleaseCorrect, by=list(dfdata$Group), FUN=mean)
  Group.1        x
1       c 38.50000
2       e 42.13333
> aggregate(dfdata$KeyreleaseCorrect, by=list(dfdata$Group), FUN=sd)
  Group.1        x
1       c 4.033196
2       e 4.290632
>
> aggregate(dfdata$KeyreleaseError, by=list(dfdata$Group), FUN=mean)
  Group.1      x
1       c 24.375
2       e 21.800
> aggregate(dfdata$KeyreleaseError, by=list(dfdata$Group), FUN=sd)
  Group.1        x
1       c 3.879433
2       e 4.570089
```

After that, boxplot diagrams are created to compare the two groups. Comparisons being made are between the number of correct and wrong keypresses, differences in the deviation in milliseconds between the moment of the keypress and the moment that it should have been pressed according to the original file, the number of correct and wrong key-releases, the deviation of milliseconds between the keypress-length and the length that it should have been pressed according to the original file, as well as differences in the composite score. To gain further insights, boxplots for key releases between the groups were analyzed for each level of difficulty.

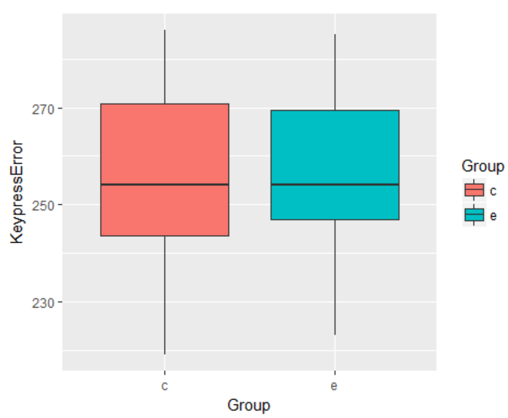
```
ggplot(data=dfdata, aes(x = Group, y=compositescore, fill=Group)) +
  geom_boxplot()
```



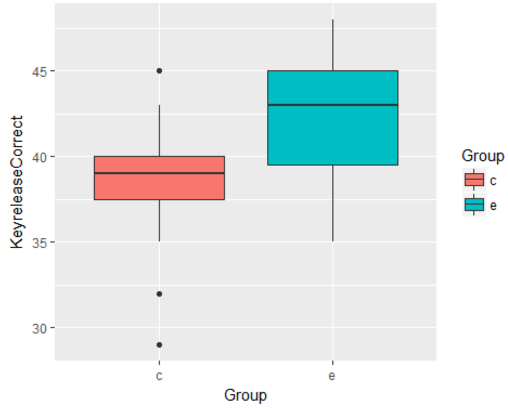

```
ggplot(data=dfdata, aes(x = Group, y=KeypressCorrect, fill=Group)) +
  geom_boxplot()
```



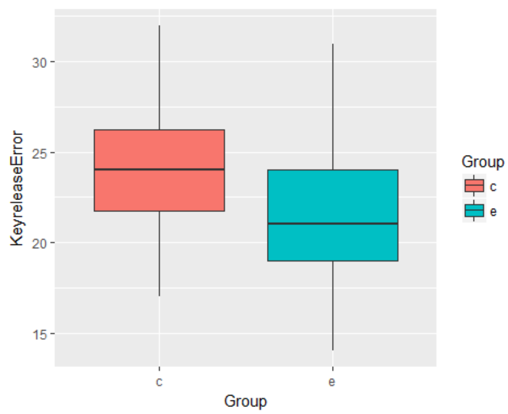

```
ggplot(data=dfdata, aes(x = Group, y=KeypressError, fill=Group)) +
  geom_boxplot()
```



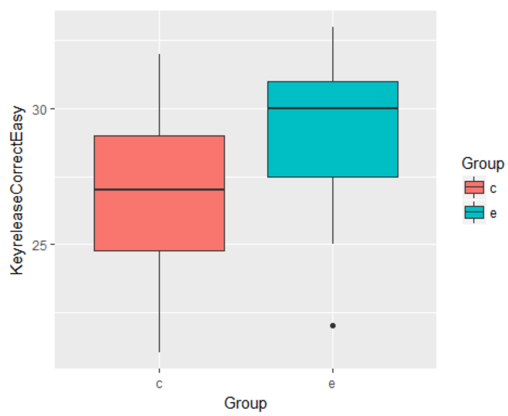

```
ggplot(data=dfdata, aes(x = Group, y=KeyreleaseCorrect, fill=Group)) +
  geom_boxplot()
```



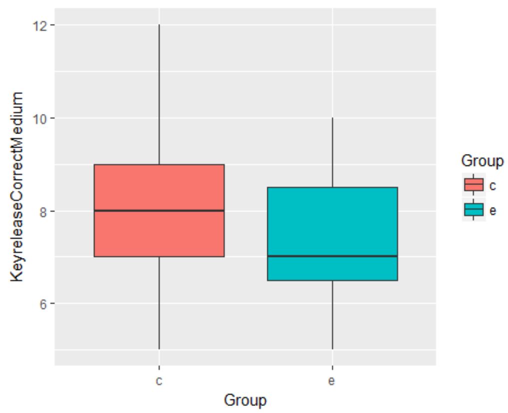

```r
ggplot(data=dfdata, aes(x = Group, y=KeyreleaseError, fill=Group)) +
  geom_boxplot()
```



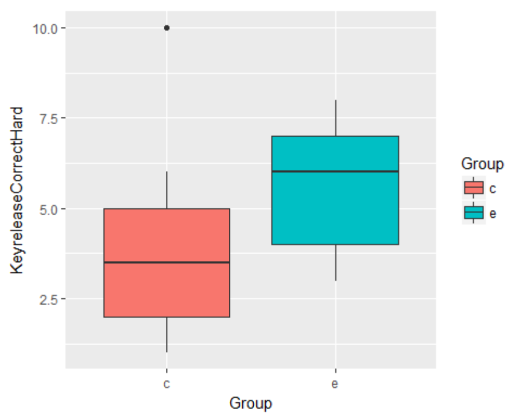

```r
ggplot(data=dfdata, aes(x = Group, y=KeyreleaseCorrectEasy, fill=Group)) +
  geom_boxplot()
```




```r
ggplot(data=dfdata, aes(x = Group, y=KeyreleaseCorrectMedium, fill=Group)) +
  geom_boxplot()
```




```
ggplot(data=dfdata, aes(x = Group, y=KeyreleaseCorrectHard, fill=Group)) +
  geom_boxplot()
```

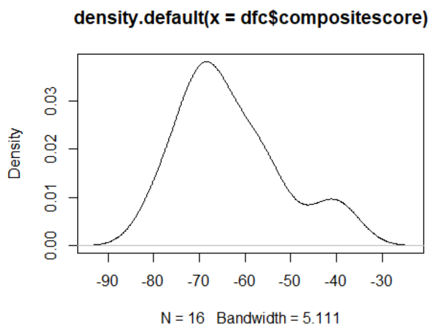


The Pearson correlation coefficient was calculated for the 4 predictive variables and the composite score

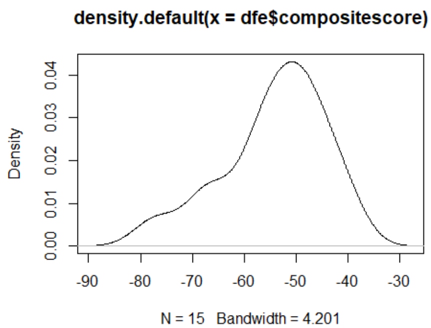
```
#create new dataframe and conduct pearsons correlations coefficient
cor_data <- dfdata[,c(5,6,8,9,23)]
round(cor(cor_data),2)

##                   KeypressCorrect KeypressError KeyreleaseCorrect
## KeypressCorrect              1.00          0.08              0.66
## KeypressError                0.08          1.00              0.20
## KeyreleaseCorrect            0.66          0.20              1.00
## KeyreleaseError              0.64         -0.10             -0.15
## compositescore               0.33         -0.59              0.65
##                   KeyreleaseError compositescore
## KeypressCorrect              0.64           0.33
## KeypressError               -0.10          -0.59
## KeyreleaseCorrect           -0.15           0.65
## KeyreleaseError              1.00          -0.23
## compositescore              -0.23           1.00
```

During this section Shapiro-Wilks tests are executed, and the program decides whether T-Tests or Wilcoxon Tests are used, depending on the distribution of the population.

```
# indexing via this! >>> CompleteDataSetL2PM["ParticipantNumber"]
splitdf <-split(dfdata, dfdata$Group, drop = TRUE)
dfc <-splitdf$c
dfe <-splitdf$e

#shapiro compositescore
result <- shapiro.test(as.numeric(paste(dfc$compositescore)))
result$p.value

## [1] 0.2154903

result <- shapiro.test(as.numeric(paste(dfe$compositescore)))
shapiro<-round(result$p.value, digits=2)

plot(density(dfc$compositescore))
```




```
plot(density(dfe$compositescore))
```



```
#test compositescore
if(shapiro>0.05)
{
  ttest<-t.test(as.numeric(paste(dfc$compositescore)),
as.numeric(paste(dfe$compositescore)), alternative = "two.sided")
  print(paste('Results for Compositescore: Shapiro estimated a normal distribution
(',shapiro,'), an independent samples t-test is carried out:'))
  print(ttest)
} else
{
 wilcox<-wilcox.test(as.numeric(paste(dfc$compositescore)),
as.numeric(paste(dfe$compositescore)), alternative = "two.sided")
  print(paste('Results for Compositescore: Shapiro estimated a non-parametric
distribution (',shapiro,'), Wilcoxon rank sum test is carried out:'))
    print(wilcox)
}
```

```
## [1] "Results for Compositescore: Shapiro estimated a normal distribution ( 0.4 ),
an independent samples t-test is carried out:"
##
##  Welch Two Sample t-test
##
## data:  as.numeric(paste(dfc$compositescore)) and
as.numeric(paste(dfe$compositescore))
## t = -2.3336, df = 28.888, p-value = 0.02679
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -16.858033  -1.108633
## sample estimates:
## mean of x mean of y
## -63.25000 -54.26667
```

```
#shapiro KeypressCorrect
result <- shapiro.test(as.numeric(paste(dfc$KeypressCorrect)))
result$p.value
```

```
## [1] 0.533027
```

```
result <- shapiro.test(as.numeric(paste(dfe$KeypressCorrect)))
shapiro<-round(result$p.value, digits=2)

#test KeypressCorrect
if(shapiro>0.05)
{
  ttest<-t.test(as.numeric(paste(dfc$KeypressCorrect)),
as.numeric(paste(dfe$KeypressCorrect)), alternative = "two.sided")
  print(paste('Results for Correct Keypresses: Shapiro estimated a normal
distribution (',shapiro,'), an independent samples t-test is carried out:'))
  print(ttest)
} else
{

wilcox<-wilcox.test(as.numeric(paste(dfc$KeypressCorrect)),
as.numeric(paste(dfe$KeypressCorrect)), alternative = "two.sided")
```

```
  print(paste('Results for Correct Keypresses: Shapiro estimated a non-parametric
distribution (',shapiro,'), Wilcoxon rank sum test is carried out:'))
    print(wilcox)
}

## [1] "Results for Correct Keypresses: Shapiro estimated a normal distribution (
0.32 ), an independent samples t-test is carried out:"
##
##  Welch Two Sample t-test
##
## data:  as.numeric(paste(dfc$KeypressCorrect)) and
as.numeric(paste(dfe$KeypressCorrect))
## t = -0.50517, df = 28.824, p-value = 0.6173
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -5.344270  3.227603

## sample estimates:
## mean of x mean of y
##  62.87500  63.93333

#shapiro KeypressError
result <- shapiro.test(as.numeric(paste(dfc$KeypressError)))
result$p.value

## [1] 0.8307235

result <- shapiro.test(as.numeric(paste(dfe$KeypressError)))
shapiro<-round(result$p.value, digits=2)

#test KeypressError
if(shapiro>0.05)
{
  ttest<-t.test(as.numeric(paste(dfc$KeypressError)),
as.numeric(paste(dfe$KeypressError)), alternative = "two.sided")
  print(paste('Results for Wrong Keypresss: Shapiro estimated a normal distribution
(',shapiro,'), an independent samples t-test is carried out:'))
  print(ttest)
} else
{
 wilcox<-wilcox.test(as.numeric(paste(dfc$KeypressError)),
as.numeric(paste(dfe$KeypressError)), alternative = "two.sided")
  print(paste('Results for Wrong Keypresss: Shapiro estimated a non-parametric
distribution (',shapiro,'), Wilcoxon rank sum test is carried out:'))
    print(wilcox)
}

## [1] "Results for Wrong Keypresss: Shapiro estimated a normal distribution ( 0.49
), an independent samples t-test is carried out:"
##
##  Welch Two Sample t-test
##
## data:  as.numeric(paste(dfc$KeypressError)) and
as.numeric(paste(dfe$KeypressError))
## t = 0.13372, df = 28.626, p-value = 0.8946
## alternative hypothesis: true difference in means is not equal to 0
```

```
## 95 percent confidence interval:
##  -12.27685  13.99352
## sample estimates:
## mean of x mean of y
##  256.1250  255.2667
```

```r
#shapiro KeyreleaseCorrect
result <- shapiro.test(as.numeric(paste(dfc$KeyreleaseCorrect)))
result$p.value
```

```
## [1] 0.2299438
```

```r
result <- shapiro.test(as.numeric(paste(dfe$KeyreleaseCorrect)))
shapiro<-round(result$p.value, digits=2)



#test KeyreleaseCorrect
if(shapiro>0.05)
{
  ttest<-t.test(as.numeric(paste(dfc$KeyreleaseCorrect)),
as.numeric(paste(dfe$KeyreleaseCorrect)), alternative = "two.sided")
  print(paste('Results for Correct Keyreleases: Shapiro estimated a normal
distribution (',shapiro,'), an independent samples t-test is carried out:'))
  print(ttest)
} else
{
 wilcox<-wilcox.test(as.numeric(paste(dfc$KeyreleaseCorrect)),
as.numeric(paste(dfe$KeyreleaseCorrect)), alternative = "two.sided")
  print(paste('Results for Correct Keyreleases: Shapiro estimated a non-parametric
distribution (',shapiro,'), Wilcoxon rank sum test is carried out:'))
    print(wilcox)
}
```

```
## [1] "Results for Correct Keyreleases: Shapiro estimated a normal distribution (
0.16 ), an independent samples t-test is carried out:"
##
##  Welch Two Sample t-test
##
## data:  as.numeric(paste(dfc$KeyreleaseCorrect)) and
as.numeric(paste(dfe$KeyreleaseCorrect))
## t = -2.4255, df = 28.529, p-value = 0.02186
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -6.6992593 -0.5674073
## sample estimates:
## mean of x mean of y
##  38.50000  42.13333
```

```r
#shapiro KeyreleaseError
result <- shapiro.test(as.numeric(paste(dfc$KeyreleaseError)))
result$p.value
```

```
## [1] 0.8840993
```

```
result <- shapiro.test(as.numeric(paste(dfe$KeyreleaseError)))
shapiro<-round(result$p.value, digits=2)

#test KeyreleaseError
if(shapiro>0.05)
{
  ttest<-t.test(as.numeric(paste(dfc$KeyreleaseError)),
as.numeric(paste(dfe$KeyreleaseError)), alternative = "two.sided")
  print(paste('Results for Wrong Keyreleases: Shapiro estimated a normal distribution
(',shapiro,'), an independent samples t-test is carried out:'))
  print(ttest)
} else
{
 wilcox<-wilcox.test(as.numeric(paste(dfc$KeyreleaseError)),
as.numeric(paste(dfe$KeyreleaseError)), alternative = "two.sided")
  print(paste('Results for Wrong Keyreleases: Shapiro estimated a non-parametric
distribution (',shapiro,'), Wilcoxon rank sum test is carried out:'))
    print(wilcox)
}

## [1] "Results for Wrong Keyreleases: Shapiro estimated a normal distribution ( 0.51
), an independent samples t-test is carried out:"
##
##  Welch Two Sample t-test
##
## data:  as.numeric(paste(dfc$KeyreleaseError)) and
as.numeric(paste(dfe$KeyreleaseError))
## t = 1.6859, df = 27.564, p-value = 0.1031
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.5560057  5.7060057
## sample estimates:
## mean of x mean of y
##    24.375    21.800

#shapiro KeypressDeviationMS
result <- shapiro.test(as.numeric(paste(dfc$KeypressDeviationMS)))
result$p.value

## [1] 0.9217665

result <- shapiro.test(as.numeric(paste(dfe$KeypressDeviationMS)))
shapiro<-round(result$p.value, digits=2)

#test KeypressDeviationMS
if(shapiro>0.05)
{
  ttest<-t.test(as.numeric(paste(dfc$KeypressDeviationMS)),
as.numeric(paste(dfe$KeypressDeviationMS)), alternative = "two.sided")
  print(paste('Results for Keypress Deviation: Shapiro estimated a normal
distribution (',shapiro,'), an independent samples t-test is carried out:'))
  print(ttest)
} else
{
```

```
wilcox<-wilcox.test(as.numeric(paste(dfc$KeypressDeviationMS)),
as.numeric(paste(dfe$KeypressDeviationMS)), alternative = "two.sided")
  print(paste('Results for Keypress Deviation: Shapiro estimated a non-parametric
distribution (',shapiro,'), Wilcoxon rank sum test is carried out:'))
    print(wilcox)
}

## [1] "Results for Correct Keypress Deviation: Shapiro estimated a normal
distribution ( 0.33 ), an independent samples t-test is carried out:"
##
##  Welch Two Sample t-test
##
## data:  as.numeric(paste(dfc$KeypressDeviationMS)) and
as.numeric(paste(dfe$KeypressDeviationMS))
## t = -1.1484, df = 27.788, p-value = 0.2606
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -1.7027731  0.4796898
## sample estimates:
## mean of x mean of y
##  14.79312  15.40467

#shapiro KeypressLengthDeviationMS
result <- shapiro.test(as.numeric(paste(dfc$KeypressLengthDeviationMS)))
result$p.value

## [1] 0.5430723

result <- shapiro.test(as.numeric(paste(dfe$KeypressLengthDeviationMS)))
shapiro<-round(result$p.value, digits=2)

#test KeypressLengthDeviationMS
if(shapiro>0.05)
{
  ttest<-t.test(as.numeric(paste(dfc$KeypressLengthDeviationMS)),
as.numeric(paste(dfe$KeypressLengthDeviationMS)), alternative = "two.sided")
  print(paste('Results for Length Deviation: Shapiro estimated a normal distribution
(',shapiro,'), an independent samples t-test is carried out:'))
  print(ttest)
} else
{
 wilcox<-wilcox.test(as.numeric(paste(dfc$KeypressLengthDeviationMS)),
as.numeric(paste(dfe$KeypressLengthDeviationMS)), alternative = "two.sided")
  print(paste('Results for Length Deviation: Shapiro estimated a non-parametric
distribution (',shapiro,'), Wilcoxon rank sum test is carried out:'))
    print(wilcox)
}

## [1] "Results for Length Deviation: Shapiro estimated a normal distribution ( 0.68
), an independent samples t-test is carried out:"
##
##  Welch Two Sample t-test
##
## data:  as.numeric(paste(dfc$KeypressLengthDeviationMS)) and
as.numeric(paste(dfe$KeypressLengthDeviationMS))
## t = 0.065245, df = 25.003, p-value = 0.9485
```

```
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -1.724428  1.837261
## sample estimates:
## mean of x mean of y
##  11.50375  11.44733
```

Finally, ANOVA analysis are carried out to analyze differences between the difficulty levels easy, medium, and hard.

```
Y <- cbind(dfdata$notecorrectEasy,dfdata$notecorrectMedium,dfdata$notecorrectHard)
resultsmanova=manova(Y ~ Group,data=dfdata)
summary(resultsmanova, test="Pillai")

##           Df   Pillai approx F num Df den Df Pr(>F)
## Group      1 0.010419 0.094758      3     27 0.9623
## Residuals 29

summary.aov(resultsmanova)

##  Response 1 :
##             Df Sum Sq Mean Sq F value Pr(>F)
## Group        1   4.07  4.0694  0.2855 0.5972
## Residuals   29 413.35 14.2534
##
##  Response 2 :
##             Df Sum Sq Mean Sq F value Pr(>F)
## Group        1  0.071  0.0711  0.0259 0.8733
## Residuals   29 79.671  2.7473
##
##  Response 3 :
##             Df  Sum Sq Mean Sq F value Pr(>F)
## Group        1   0.437  0.4367   0.081  0.778
## Residuals   29 156.337  5.3909

dfdata$notecorrectEasyperc <- dfdata$notecorrectEasy/174
dfdata$notecorrectMediumperc <- dfdata$notecorrectMedium/90
dfdata$notecorrectHardperc <- dfdata$notecorrectHard/48

Y <-
cbind(dfdata$notecorrectEasyperc,dfdata$notecorrectMediumperc,dfdata$notecorrectHardp
erc)
resultsmanova=manova(Y ~ Group,data=dfdata)
summary(resultsmanova, test="Pillai")

##           Df   Pillai approx F num Df den Df Pr(>F)
## Group      1 0.010419 0.094758      3     27 0.9623
## Residuals 29

summary.aov(resultsmanova)
```

```
##  Response 1 :
##              Df    Sum Sq    Mean Sq F value Pr(>F)
## Group         1 0.0001344 0.00013441  0.2855 0.5972
## Residuals    29 0.0136527 0.00047078
##
##  Response 2 :
##              Df    Sum Sq    Mean Sq F value Pr(>F)
## Group         1 0.0000088 0.00000878  0.0259 0.8733
## Residuals    29 0.0098359 0.00033917
##
##  Response 3 :
##              Df   Sum Sq    Mean Sq F value Pr(>F)
## Group         1 0.000190 0.00018954   0.081  0.778
## Residuals    29 0.067855 0.00233982
```

```
Y <-
cbind(dfdata$KeypressDeviationMSEasy,dfdata$KeypressDeviationMSMedium,dfdata$Keypress
DeviationMSHard)
resultsmanova=manova(Y ~ Group,data=dfdata)
summary(resultsmanova, test="Pillai")
```

```
##            Df   Pillai approx F num Df den Df Pr(>F)
## Group       1 0.043988   0.4141      3     27 0.7442
## Residuals 29
```

```
summary.aov(resultsmanova)
```

```
##  Response 1 :
##              Df  Sum Sq Mean Sq F value Pr(>F)
## Group         1  0.7567 0.75665  0.7366 0.3978
## Residuals    29 29.7905 1.02726
##
##  Response 2 :
##              Df Sum Sq Mean Sq F value Pr(>F)
## Group         1 0.1107 0.11071  0.3667 0.5495
## Residuals    29 8.7547 0.30189
##
##  Response 3 :
##              Df Sum Sq Mean Sq F value Pr(>F)
## Group         1  0.249 0.24898  0.6329 0.4327
## Residuals    29 11.408 0.39338
```

```
Y <-
cbind(dfdata$KeypressLengthDeviationMSEasy,dfdata$KeypressLengthDeviationMSMedium,dfd
ata$KeypressLengthDeviationMSHard)
resultsmanova=manova(Y ~ Group,data=dfdata)
summary(resultsmanova, test="Pillai")
```

```
##            Df  Pillai approx F num Df den Df Pr(>F)
## Group       1 0.14559   1.5335      3     27 0.2285
## Residuals 29
```

```
summary.aov(resultsmanova)
```

```
##  Response 1 :
##              Df Sum Sq Mean Sq F value Pr(>F)
```

```
## Group        1  0.139  0.1391  0.0516 0.8218
## Residuals   29 78.106  2.6933
##
##  Response 2 :
##              Df Sum Sq Mean Sq F value Pr(>F)
## Group         1  1.201  1.2006  1.0332 0.3178
## Residuals    29 33.698  1.1620
##
##  Response 3 :
##              Df  Sum Sq Mean Sq F value  Pr(>F)
## Group         1  1.7205 1.72052  3.6066  0.06754 .
## Residuals    29 13.8345 0.47705
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Y <-
cbind(dfdata$KeyreleaseCorrectEasy,dfdata$KeyreleaseCorrectMedium,dfdata$KeyreleaseCo
rrectHard)
resultsmanova=manova(Y ~ Group,data=dfdata)
summary(resultsmanova, test="Pillai")

##            Df  Pillai approx F num Df den Df  Pr(>F)
## Group       1 0.20414   2.3085      3     27  0.09896 .
## Residuals 29
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

summary.aov(resultsmanova)

##  Response 1 :
##              Df  Sum Sq Mean Sq F value  Pr(>F)
## Group         1  37.046  37.046  3.7507  0.06258 .
## Residuals    29 286.438   9.877
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##  Response 2 :
##              Df Sum Sq Mean Sq F value Pr(>F)
## Group         1  0.603 0.60336  0.2079 0.6518
## Residuals    29 84.171 2.90244
##
##  Response 3 :
##              Df  Sum Sq Mean Sq F value  Pr(>F)
## Group         1  23.037 23.0371  5.4161  0.02714 *
## Residuals    29 123.350  4.2534
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

aggregate(dfdata$KeyreleaseCorrectEasy, list(dfdata$Group), mean)

##   Group.1       x
## 1       c 26.8125
## 2       e 29.0000

aggregate(dfdata$KeyreleaseCorrectMedium, list(dfdata$Group), mean)
```

```
##   Group.1        x
## 1       c 7.812500
## 2       e 7.533333
```

```
aggregate(dfdata$KeyreleaseCorrectHard, list(dfdata$Group), mean)
```

```
##   Group.1     x
## 1       c 3.875
## 2       e 5.600
```