Data processing and analysis for: Affective, physiological, and attention restoration at a wooden desk: A pilot study

Dean Lipovac, InnoRenew CoE & University of Primorska
Mike Burnard, InnoRenew CoE & University of Primorska
Jure Žitnik, InnoRenew CoE & University of Primorska
firstname.lastname@innorenew.eu

October 29, 2021

Contents

Introduction

This analysis primarily investigates how people respond to the Mental Arithmetic Task (MAT) in terms of their affective states and physiological arousal, and how their cognitive performance changes between two task administrations. The analysis also checks whether affective, physiological, and cognitive responses differ between settings furnished with or without wood. We base our analysis on self-reported affective states, captured physiological (electrodermal and cardiovascular) activity, and results on the cognitive task (i.e., MAT).

Data dictionary

Participant and session data:

- p_id (numeric): participant identifier
- age (string): participant's age as age group
- gender (string): participant's gender
- start and end times of study phases (numeric; several variables): date and time of the measurement [yyyy-mm-dd hh:mm:ss]

Experimental conditions data:

- phase (string): period of the study
- condition (string): test setting - either wooden or white desktop

Affective states data:

- pleas%>% (numeric): score on the pleasure dimension of affect (from 1 - especially unpleasant to 9 - especially pleasant)
- arousal (numeric): score on the arousal dimension of affect (from 1 - especially inactivated to 9 - especially activated)

Cognitive performance data (Mental Arithmetic Task):

- n_responses (numeric) - number of total responses
- n_correct (numeric) - number of correct responses
- n_incorrect (numeric) - number of incorrect responses
- prop_correct (numeric) - proportion of correct responses

Electrodermal activity (EDA):

- p_id_timepoint (numeric) - timepoint of the measurement, starting with 1 for each participant
- eda_raw (numeric) - raw electrodermal activity (EDA) signal [S]
- eda_clean (numeric) - clean EDA signal [S]
- eda_tonic (numeric) - tonic EDA signal [S]
- eda_phasic (numeric) - phasic EDA signal [S]
- scr_onsets (numeric) - the point of onset of the skin conductance response (SCR) [0 or 1]
- scr_peaks (numeric) - the point of peak of the SCR [0 or 1]
- scr_height (numeric) - SCR amplitude of the signal including the tonic component [S]
- scr_amplitude (numeric) - SCR amplitude of the signal excluding the tonic component [S]
- scr_rise_time (numeric) - the rise time of the SCR [minutes]
- scr_recovery (numeric) - the point of recovery of the SCR [0 or 1]
- scr_recovery_time (numeric) - the recovery time of the SCR [minutes]

More information on EDA parameters is available at NeuroKit2 package webpage.

Cardiovascular activity:

- hr_bpm (numeric) - heart rate [beats per minute]
- rmssd (numeric) - root mean square of successive beat to beat interval differences [ms]
- other variables based on electrocardiography signal that are not analyzed here

Import and prepare data

Participant ID, session data, and affective states

We import two datasets and transform them to two tibbles: one includes participant and session data and the other includes affective states data.

```
# Participant ID and experimental
# condition
id_condition <- read_csv("data/id_condition.csv")
id_condition <- id_condition %>%
    filter(!is.na(p_id)) %>%
    mutate(p_id = as.numeric(p_id))

# Participant ID, demographics, session
# times, and affective states
id_session_affect <- read_csv("data/id_session_affect.csv")

## Remove unnecessary variables
id_session_affect <- id_session_affect %>%
    select(!contains(c("pilot", "complete",
```

```
        "restarts", "language", "timer")))

## Clean column names
colnames(id_session_affect) <- str_remove(colnames(id_session_affect),
    "session.|results.")

colnames(id_session_affect) <- make_clean_names(colnames(id_session_affect))

## Make 'Participant ID' numeric
id_session_affect$p_id <- as.numeric(id_session_affect$p_id)

# Merge the two datasets from above
id_session_affect <- left_join(id_session_affect,
    id_condition, by = "p_id")

## Clean column names
id_session_affect <- id_session_affect %>%
    rename(pleasure_baseline = pleasure...12,
        arousal_baseline = arousal...13,
        pleasure_task1 = pleasure...15, arousal_task1 = arousal...16,
        pleasure_recovery = pleasure...19,
        arousal_recovery = arousal...20)

# Create a separate tibble for
# participant ID and session data
session <- id_session_affect %>%
    select(!contains(c("pleasure", "arousal",
        "condition")))

# Create a separate tibble for
# affective states
affect <- id_session_affect %>%
    select(p_id, condition, contains(c("pleasure",
        "arousal")))

## Transform the tibble with affective
## states to long format
affect <- affect %>%
    pivot_longer(contains(c("pleasure", "arousal")),
        names_to = "affect_state", values_to = "affect_value")

## Create 'study phase' variable in the
## affective states tibble
affect <- affect %>%
    separate(affect_state, c("affect_state",
        "phase")) %>%
    relocate(phase, .after = condition) %>%
    arrange(p_id)

## Change 'study phase' variable to a
## factor
affect$phase <- factor(affect$phase, levels = c("baseline",
    "task1", "recovery"))
```

Summaries of physiological data

We import data on electrodermal and cardiovascular activity: the data includes summaries (means) of parameters for each participant at each study phase.

```
# Import data for all subjects and put
# them in one tibble
physio_smr <- list.files("data/physio_summary",
    full.names = TRUE) %>%
    map_dfr(function(x) read_delim(x, delim = "\t") %>%
        mutate(p_id = str_replace(basename(x),
            ".txt", "")))

# Take column names from the first row
colnames(physio_smr)[7:34] <- physio_smr[1,
    7:34]

# Clean the data
physio_smr <- physio_smr %>%
    select(-"Channel 6") %>%
    filter(HR_bpm != c("Mean", "BPM")) %>%
    clean_names() %>%
    rename(time = x1, gsr_us = gsr_u_s, phase = cmt_text,
        prrx = p_r_rx) %>%
    relocate(c(p_id, phase), .before = time)

# Change the class of certain variables
physio_smr <- physio_smr %>%
    mutate_all(funs(str_replace(., ",", "."))) %>%
    mutate_at(vars(-c(p_id, time, phase,
        analysis_start, analysis_end)), as.numeric) %>%
    mutate_at(vars(time, analysis_start,
        analysis_end), lubridate::hms) %>%
    mutate(p_id = str_remove_all(p_id, "[:punct:]|[:alpha:]"),
        p_id = as.numeric(p_id))

# Rename study phases
physio_smr$phase <- recode(physio_smr$phase,
    B = "baseline", T1 = "task1", R = "recovery",
    `R ` = "recovery", T2 = "task2", E = "end")

# Add participants' experimental
# condition
physio_smr <- left_join(physio_smr, id_condition,
    by = "p_id") %>%
    relocate(condition, .after = "p_id")

# Fill in missing values in the 'study
# phase' variable
physio_smr <- physio_smr %>%
    mutate(phase = if_else(phase == "end",
        "task2", phase))

# Manually fix an error in 'study
```

```
# phase'
physio_smr <- physio_smr %>%
    mutate(phase = if_else(p_id == 10 & time ==
        "17H 30M 23.209388S", "task1", phase))

# Change 'phase' to a factor
physio_smr$phase <- factor(physio_smr$phase,
    levels = c("baseline", "task1", "recovery",
        "task2"))

# Export data write_csv(physio_smr,
# 'data/physio_smr.csv')

# Create a tibble without physiological
# data at Task(2) phase
physio_smr_filt <- physio_smr %>%
    filter(phase != "task2")
```

Continuous physiological data

Here we import continuous physiological data, primarily so we can later process the raw EDA data to obtain additional EDA parameters.

```
# Import data for all subjects and put
# them in one tibble
physio_cont <- list.files("data/physio_continuous",
    full.names = TRUE) %>%
    map_dfr(function(x) read_delim(x, delim = "\t") %>%
        mutate(p_id = str_replace(basename(x),
            ".txt", "")))

# Clean the data
physio_cont <- physio_cont %>%
    select(1:5, p_id) %>%
    filter(HR_bpm != c("Mean", "BPM")) %>%
    clean_names() %>%
    rename(time = x1, gsr_us = gsr_u_s) %>%
    relocate(c(p_id), .before = time) %>%
    filter(!is.na(time))

# Change the class of certain variables
physio_cont <- physio_cont %>%
    mutate_all(funs(str_replace(., ",", "."))) %>%
    mutate_at(vars(hr_bpm:gsr_us), as.numeric) %>%
    mutate_at(vars(time), lubridate::hms) %>%
    mutate(p_id = str_remove(p_id, "S"),
        p_id = as.numeric(p_id))

# Add participants experimental
# condition
physio_cont <- left_join(physio_cont, id_condition,
    by = "p_id") %>%
```

```r
    relocate(condition, .after = "p_id")

# Export data write_csv(physio_cont,
# 'physio_cont.csv')

# Take only EDA data for further
# processing
eda <- physio_cont %>%
    select(p_id, condition, time, gsr_us) %>%
    filter(!is.na(gsr_us))

# Add time point to each data point
eda <- eda %>%
    group_by(p_id) %>%
    mutate(p_id_timepoint = 1:n(), .before = time)
```

Process EDA in neurokit2

Raw EDA data is processed with the Python NeuroKit2 package, step by step. We have more calculation methods to choose from at each step, but we stay with default methods.

```python
import neurokit2 as nk
import pandas as pd

# Create empty lists that will hold data from the for loop
eda_cleaned_all_py = []
eda_phasic_all_py = []
eda_peaks_all_py = []

# Run a for loop that will compute EDA parameters based on raw EDA data
#(separately for each participant)
for n in range(1,23):
  eda_p_id = r.eda[r.eda["p_id"] == n]
  eda_cleaned = nk.eda_clean(eda_p_id[["gsr_us"]], sampling_rate=16)
  eda_phasic = nk.eda_phasic(eda_cleaned, sampling_rate=16)
  eda_phasic_values = eda_phasic["EDA_Phasic"].values
  eda_peaks = nk.eda_peaks(eda_phasic_values)
  eda_cleaned_all_py.append(eda_cleaned)
  eda_phasic_all_py.append(eda_phasic)
  eda_peaks_all_py.append(eda_peaks)
```

Convert processed EDA to R's tibble

We convert the EDA data processed in Python/NeuroKit2 to R's tibble.

```r
# Convert the Python objects to R's
# lists
eda_cleaned_all_temp <- py$eda_cleaned_all_py
eda_phasic_all_temp <- py$eda_phasic_all_py
eda_peaks_all_temp <- py$eda_peaks_all_py
```

```r
# Convert lists to tibbles EDA Phasic
eda_phasic_all <- bind_rows(eda_phasic_all_temp,
    .id = "p_id")

## EDA Clean
eda_cleaned_all <- map_dfr(eda_cleaned_all_temp,
    as_tibble, .id = "p_id")

## EDA Peaks
eda_peaks_all <- tibble()

for (n in 1:22) {
    a <- eda_peaks_all_temp[[n]][[1]]
    a <- py_to_r(a)
    a <- a %>%
        add_column(p_id = n, .before = "SCR_Onsets")
    eda_peaks_all <- bind_rows(eda_peaks_all,
        a)
}

# Merge EDA data (Raw, Phasic, Peaks,
# Participant IDs) Merge EDA Phasic and
# EDA peaks data
eda_full <- bind_cols(eda, eda_cleaned_all,
    eda_phasic_all, eda_peaks_all)

## Fix ID columns and clean names
eda_full <- eda_full %>%
    rename(p_id = p_id...1, eda_raw = gsr_us,
        eda_clean = value) %>%
    select(-ends_with(c("6", "8", "11"))) %>%
    # Remove redundant ID columns
clean_names()

# Export data write_csv(eda_full,
# 'data/eda_full.csv')
```

Add time variable from summarised phyisological data to EDA data

We add the time variable to the EDA data from the previous dataset containing summaries of physiological data. We retain only the values for 5 minutes of the lowest physiological activation in baseline and recovery period. The periods of the lowest physiological activation are taken from the times available in the dataset containing summaries of physiological data; these times are based on the lowest values of the skin conductance level, as calculated by the software that captured physiological data from sensors.

```r
# Add the time variable to EDA data

## Create a tibble with time stamps for
## start and end of each study phase
physio_smr_temp <- physio_smr %>%
    pivot_longer(c(analysis_start, analysis_end),
        names_to = "start_end_period", values_to = "start_end_time") %>%
    select(p_id, condition, phase, start_end_period,
```

```
        start_end_time) %>%
    mutate(start_end_time = round(start_end_time,
        0))

## Manually correct a specific timing
## in the tibble created above (due to
## a bit of missing data that does not
## allow time stamps to match)
physio_smr_temp <- physio_smr_temp %>%
    mutate(start_end_time = if_else(p_id ==
        13 & start_end_time == "16H 32M 55S",
        "16H 32M 44S", as.character(start_end_time)),
        start_end_time = hms(start_end_time))

## Round time (to get matching times
## when joining two tibbles later)
eda_full <- eda_full %>%
    mutate(time = round(time, 0))

## Merge the tibble with time stamps
## with the tibble containing
## physiological data
eda_full_time <- left_join(eda_full, physio_smr_temp %>%
    select(-phase), by = c("p_id", "condition",
    time = "start_end_time")) %>%
    filter(!is.na(start_end_period))

# We have too many time matches, so
# we'll retain only the first match for
# each p_id/condition/phase
eda_full_time <- eda_full_time %>%
    distinct(p_id, condition, time, .keep_all = TRUE)

# Add study phase names (baseline,
# task, recovery)
eda_full_time <- eda_full_time %>%
    arrange(p_id, p_id_timepoint) %>%
    group_by(p_id) %>%
    mutate(phase = rep(c("baseline", "task1",
        "recovery", "task2"), each = 2),
        .after = condition)

# Add time stamps to the full dataset
eda_full_time_filt <- left_join(eda_full,
    eda_full_time %>%
        select(p_id, condition, phase, p_id_timepoint,
            start_end_period), by = c("p_id",
        "condition", "p_id_timepoint"))

# Only retain periods of interest in
# the full dataset
eda_full_time_filt <- eda_full_time_filt %>%
    fill(start_end_period, .direction = "down") %>%
```

```
    filter(start_end_period == "analysis_start") %>%
    fill(phase, .direction = "downup") %>%
    group_by(p_id, condition, phase)

# Remove/relocate certain variables
eda_full_time_filt <- eda_full_time_filt %>%
    select(-start_end_period) %>%
    relocate(phase, .after = condition)

# Make a tibble without the problematic
# data of subject #12 and without
# Task(2) data
eda_full_time_filt_no12 <- eda_full_time_filt %>%
    filter(phase != "task2", (p_id != 12))

# Make a tibble with data summaries
# Compute means for EDA Tonic and SCR
# Peaks
eda_full_time_filt_no12_smr <- eda_full_time_filt_no12 %>%
    group_by(p_id, condition, phase) %>%
    mutate(scr_peaks_perc = 100 * scr_peaks) %>%
    summarise(across(c(eda_tonic, scr_peaks_perc),
        mean, na.rm = TRUE)) %>%
    rename_with(~paste0("mean_", .), .cols = c(eda_tonic,
        scr_peaks_perc))

## SCR Amplitude: Average over only
## non-zero values of SCR Amplitude
eda_smr_scr_amp <- eda_full_time_filt_no12 %>%
    filter(scr_amplitude > 0) %>%
    group_by(p_id, condition, phase) %>%
    summarise(mean_scr_amplitude = mean(scr_amplitude))

## Merge the two above tibbles
eda_full_time_filt_no12_smr <- left_join(eda_full_time_filt_no12_smr,
    eda_smr_scr_amp)

## Replace NAs in Mean SCR Amplitude
## with Os (as they are Os and not NAs)
eda_full_time_filt_no12_smr <- eda_full_time_filt_no12_smr %>%
    mutate(mean_scr_amplitude = if_else(is.na(mean_scr_amplitude),
        0, mean_scr_amplitude))
```

Mental arithmetic task

We import and transform cognitive task data.

```
# Import data as a list
task_list <- list.files("data/task", full.names = TRUE)

# Read the data and create ID's from
# file names
```

```
task_raw <- task_list %>%
    map_dfr(function(x) read_csv(x) %>%
        mutate(p_id = str_replace(basename(x),
            ".csv", "")))

# Create separate variables
task_raw <- task_raw %>%
    separate(p_id, into = c("p_id", "phase"),
        "_") %>%
    mutate(p_id = as.numeric(p_id)) %>%
    arrange(p_id) %>%
    rename(number = ...1) %>%
    relocate(c(p_id, phase), .before = number)

#####

# Create a function that will assign
# '1' to correct responses
fill_ones <- function(df) {
    for (cl in 4:length(df)) {
        df[[cl]] <- as.numeric(df[[cl]])
        include_column <- if_else(sum(is.na(df[cl])) ==
            nrow(df) | sum(df[cl], na.rm = TRUE) ==
            1, FALSE, TRUE)
        if (include_column == TRUE) {
            zero <- which(df[, cl] == 0)
            before_zero <- zero - 1
            df[1:before_zero, cl] <- 1
        } else if (sum(df[cl], na.rm = TRUE) ==
            1) {
            df[cl] <- df %>%
                select(cl) %>%
                fill(1, .direction = "up")
            return(df)
        } else {
            return(df)
        }
    }
}

#####

# Fill certain sections with '1'
task_raw <- task_raw %>%
    group_by(p_id, phase) %>%
    group_map(~fill_ones(.x), keep = TRUE) %>%
    bind_rows()

# Transform to long format
task_raw_long <- task_raw %>%
    pivot_longer("1":"9", names_to = "trial",
        values_to = "correct")
```

```
# Create a summary of the raw data
task <- task_raw_long %>%
    group_by(p_id, phase) %>%
    summarise(n_responses = sum(!is.na(correct)),
        n_correct = sum(correct, na.rm = TRUE),
        n_incorrect = sum(correct == 0, na.rm = TRUE),
        prop_correct = n_correct/n_responses)

# Add experimental condition to the
# tibble
task <- left_join(task, id_condition, by = "p_id") %>%
    relocate(condition, .after = "p_id")

# Transform to longer format
task_long <- task %>%
    pivot_longer(n_responses:prop_correct,
        names_to = "parameter", values_to = "value")

# Retain only variables of interest
task_long_filt <- task_long %>%
    filter(parameter %in% c("n_responses",
        "prop_correct"))

# Filter out NAs
task_raw_long_filt <- task_raw_long %>%
    filter(!is.na(correct)) %>%
    left_join(., id_condition, by = "p_id") %>%
    relocate(condition, .after = "p_id")

# Export data write_csv(task_long_filt,
# 'task_long_filt.csv')
```

Explore data - Summary statistics and plots

Demographics

We calculate summary statistics for age and gender.

```
# Age
session %>%
    count(age)
```

```
## # A tibble: 4 x 2
##   age            n
##   <chr>      <int>
## 1 24 or less     6
## 2 25-34         13
## 3 35-44          1
## 4 45-54          2
```

```r
# Gender
session %>%
    count(gender)
```

```
## # A tibble: 2 x 2
##   gender     n
##   <chr>  <int>
## 1 female    18
## 2 male       4
```

Affective states

Overall affective states

We calculate summary statistics and create a plot displaying overall affective states scores throughout the study.

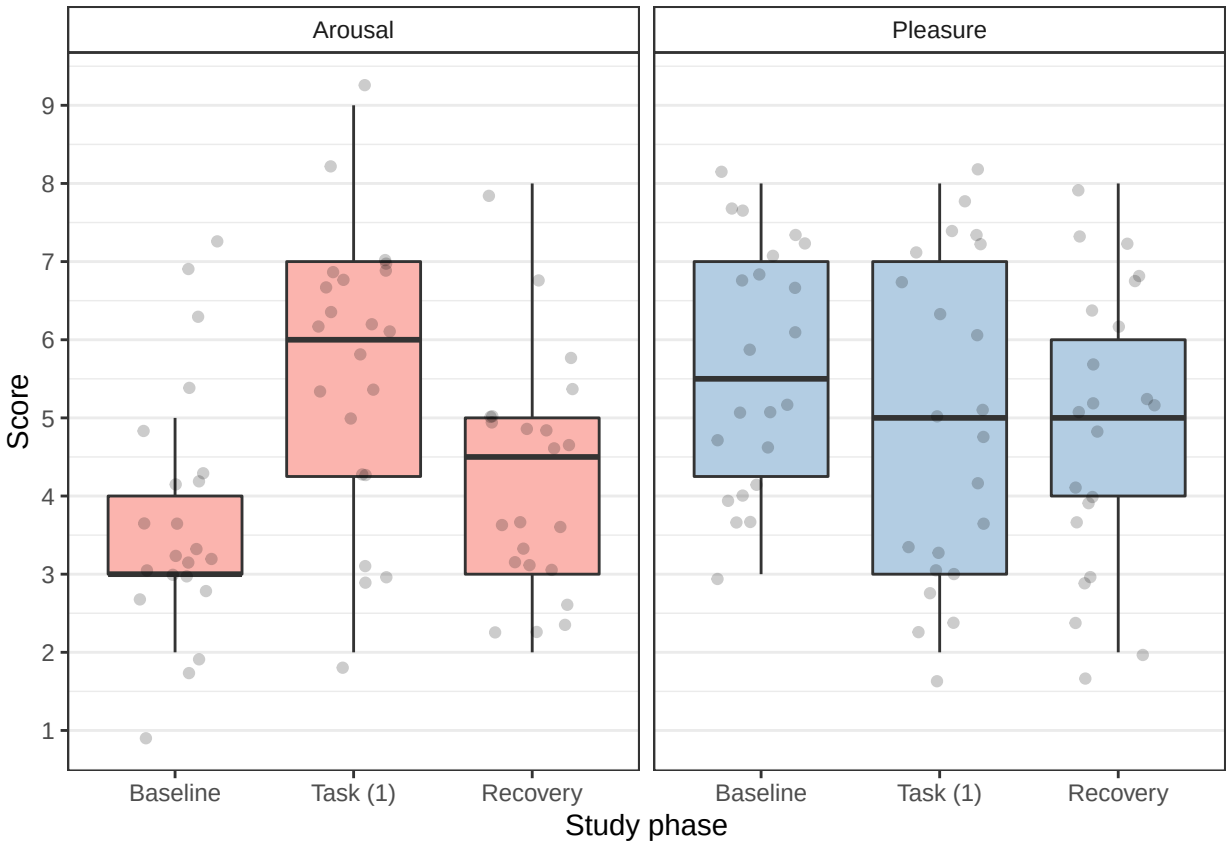
```r
#####

# Create a function that computes the
# mean and 95% confidence interval
mean_ci <- function(df, var, ...) {
    df %>%
        group_by(...) %>%
        get_summary_stats({
            {
                var
            }
        }, type = "mean_ci") %>%
        mutate(across(where(is.numeric),
            round, 2), ci95 = paste(mean -
            ci, mean + ci)) %>%
        select(-ci)
}

#####

# Arousal

## Create a tibble that includes only
## arousal data
affect_arousal <- affect %>%
    filter(affect_state == "arousal")

## Calcuate mean and 95% CIs for
## overall arousal scores
mean_ci(affect_arousal, affect_value)
```

```
## # A tibble: 1 x 4
##   variable         n  mean ci95
##   <chr>        <dbl> <dbl> <chr>
## 1 affect_value    66  4.53 4.09 4.97
```

```r
# Pleasure

## Create a tibble that includes only
## pleasure data
affect_pleasure <- affect %>%
    filter(affect_state == "pleasure")

## Calculate mean and 95% CIs for
## overall pleasure scores
mean_ci(affect_pleasure, affect_value)
```

```
## # A tibble: 1 x 4
##   variable         n  mean ci95
##   <chr>        <dbl> <dbl> <chr>
## 1 affect_value    66  5.15 4.7 5.6
```

```r
# Plot

## Create labels for the plot
affect_labels <- labeller(affect_state = c(arousal = "Arousal",
    pleasure = "Pleasure"))

## Boxplot - Arousal and pleasure
## through study phases
ggplot(affect, aes(phase, affect_value, fill = affect_state)) +
    geom_boxplot(outlier.shape = NA) + geom_jitter(width = 0.25,
    alpha = 0.2) + facet_wrap(~affect_state,
    labeller = affect_labels) + guides(fill = FALSE) +
    scale_y_continuous(breaks = 1:9) + scale_x_discrete(limits = c("baseline",
    "task1", "recovery"), labels = c("Baseline",
    "Task (1)", "Recovery")) + scale_fill_brewer(type = "qual",
    palette = 4) + labs(x = "Study phase",
    y = "Score") + theme(panel.grid.major.x = element_blank(),
    strip.background = element_rect(fill = "white"))
```



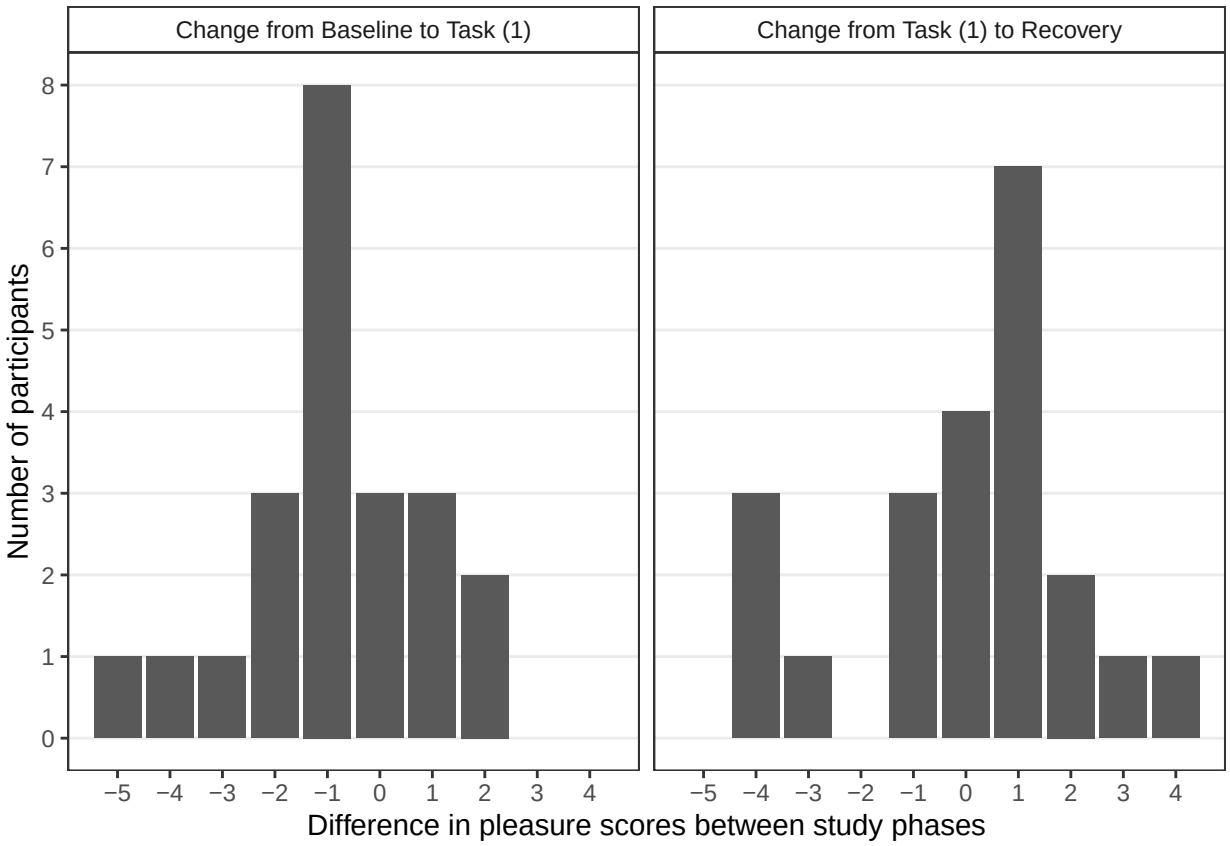

```
# Export plot
# ggsave('output/affect_plot.png',
# width = 15, height = 7.5, units =
# 'cm')
```

Changes in pleasure between study phases

We look at the changes between study phases in the affective state of pleasure within each participant.

```
# Create a tibble for the plot
affect_pleasure_plot_df <- affect_pleasure %>%
    pivot_wider(names_from = phase, values_from = affect_value) %>%
    mutate(dif_task1_baseline = task1 - baseline,
        dif_recovery_task1 = recovery - task1) %>%
    pivot_longer(c(dif_task1_baseline, dif_recovery_task1),
        names_to = "dif_phase", values_to = "dif_value") %>%
    mutate(dif_phase = factor(dif_phase,
        levels = c("dif_task1_baseline",
            "dif_recovery_task1")))

# Create labels for the plot
phase_names <- c(dif_task1_baseline = "Change from Baseline to Task (1)",
    dif_recovery_task1 = "Change from Task (1) to Recovery")

# Bar plot - Changes in pleasure within
```

```
# participants
ggplot(affect_pleasure_plot_df, aes(dif_value)) +
    facet_wrap(~dif_phase, labeller = as_labeller(phase_names)) +
    geom_bar() + labs(x = "Difference in pleasure scores between study phases",
    y = "Number of participants") + scale_x_continuous(breaks = -5:4) +
    scale_y_continuous(breaks = 0:8) + theme(panel.grid.major.x = element_blank(),
    panel.grid.minor.x = element_blank(),
    panel.grid.minor.y = element_blank(),
    strip.background = element_rect(fill = "white"))
```




```
# Export plot
# ggsave('output/pleasure_change.png',
# width = 15, height = 7.5, units =
# 'cm')
```

Participants with unusual vs usual patterns in pleasure scores

We identify six participants with unusual responses, for whom pleasure has increased or stayed the same at Task (1) and decreased or stayed the same at Recovery, in contrast with 16 participants with usual responses, for whom pleasure decreased at Task (1) and increased at Recovery.

```
# Identify participants who felt the
# same or more pleasure at Task 1 than
# at Baseline
```

```
dif_task1_baseline_unusual <- affect_pleasure_plot_df %>%
    filter(dif_phase == "dif_task1_baseline",
        dif_value >= 0) %>%
    pull(p_id)

# Identify participants who felt the
# same or less pleasure at Recovery
# than at Task (1)
dif_recovery_task1_unusual <- affect_pleasure_plot_df %>%
    filter(dif_phase == "dif_recovery_task1",
        dif_value <= 0) %>%
    pull(p_id)

# Identify participants for whom both
# of the above is true
both_dif_unusual <- dif_task1_baseline_unusual[dif_task1_baseline_unusual %in%
    dif_recovery_task1_unusual]

#####

# Create a function that fits the
# flextable to the page width when
# knitting to PDF
fit_flextable_to_page <- function(ft, pgwidth = 6) {
    ft_out <- ft %>%
        autofit()

    ft_out <- width(ft_out, width = dim(ft_out)$widths *
        pgwidth/(flextable_dim(ft_out)$widths))
    return(ft_out)
}

# Create a function that extract
# summary statistics

my_summary <- function(df, outcome_var, group_var1,
    group_var2, group_var3) {
    df <- df %>%
        mutate(response = if_else(p_id %in%
            both_dif_unusual, "Unusual (n = 6)",
            "Usual (n = 16)")) %>%
        group_by({
            {
                group_var1
            }
        }, {
            {
                group_var2
            }
        }, {
            {
                group_var3
            }
```

```
        }) %>%
        get_summary_stats({
            {
                outcome_var
            }
        }, type = "common") %>%
        # compute summary statistics
    select(-variable, -se, -ci) %>%
        # Remove unwanted variables
    mutate(across(where(is.numeric), round,
        2))  # Round numbers to two decimals

    df <- df %>%
        mutate(response = str_remove_all(response,
            "\\(n = 6\\)|\\(n = 16\\)"))

    colnames(df) <- str_to_sentence(colnames(df))

    df <- df %>%
        rename(IQR = Iqr, SD = Sd, n = N)

    df <- df %>%
        flextable() %>%
        autofit() %>%
        add_footer(Phase = "IQR = Interquartile range; SD = Standard deviation") %>%
        merge_at(j = 1:8, part = "footer")

    df <- fit_flextable_to_page(df)

    df
}

#####

# Pleasure for participants with
# unusual and usual responses

## Summary statistics
affect_pleasure %>%
    my_summary(outcome_var = affect_value,
        group_var1 = phase, group_var2 = response)  # %>%
```

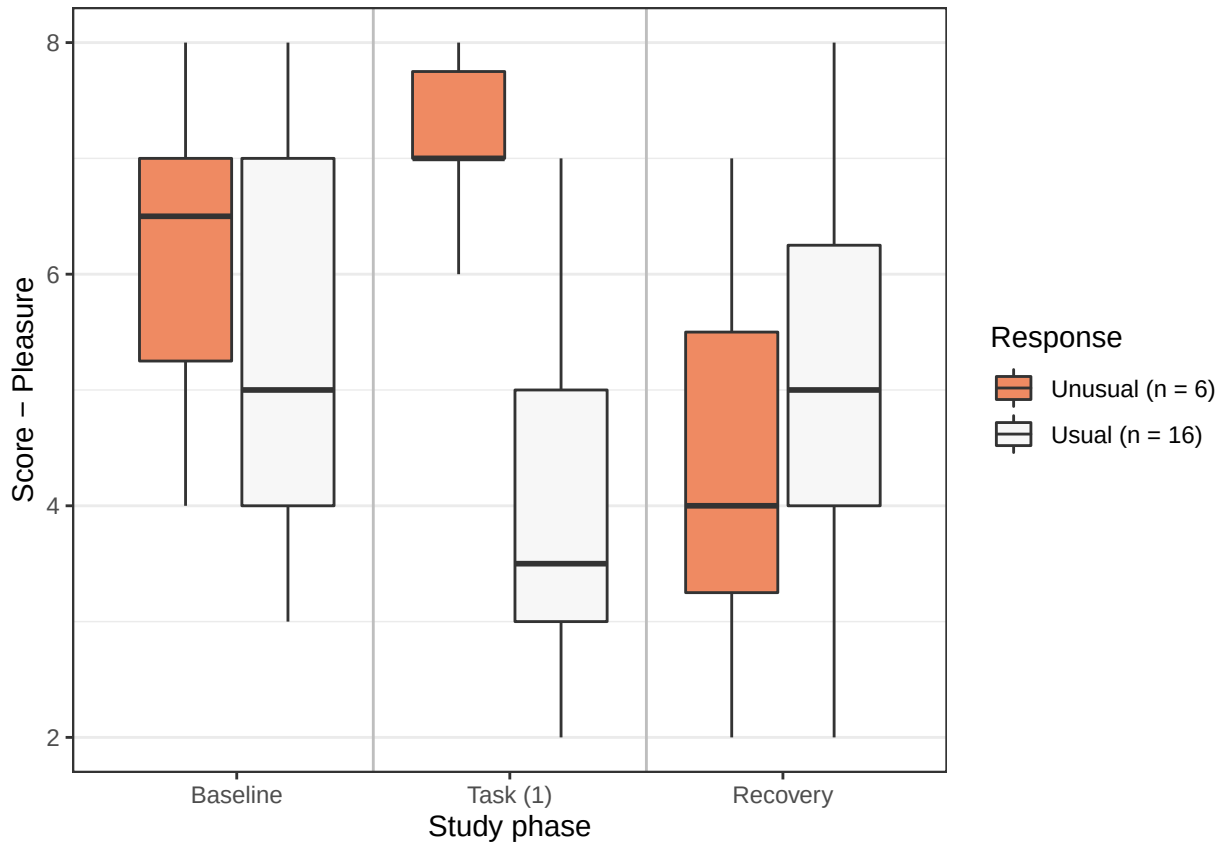
Phase	Response	n	Min	Max	Median	IQR	Mean	SD
baseline	Unusual	6	4	8	6.5	1.75	6.17	1.47
baseline	Usual	16	3	8	5.0	3.00	5.56	1.59
task1	Unusual	6	6	8	7.0	0.75	7.17	0.75
task1	Usual	16	2	7	3.5	2.00	4.00	1.67
recovery	Unusual	6	2	7	4.0	2.25	4.33	1.86

IQR = Interquartile range; SD = Standard deviation

Phase	Response	n	Min	Max	Median	IQR	Mean	SD
recovery	Usual	16	2	8	5.0	2.25	5.06	1.77

IQR = Interquartile range; SD = Standard deviation

```
# save_as_docx(path =
# 'output/Table_S1.docx')

## Plot
affect_pleasure %>%
    mutate(Response = if_else(p_id %in% both_dif_unusual,
        "Unusual (n = 6)", "Usual (n = 16)")) %>%
    ggplot(aes(phase, affect_value, fill = Response)) +
    geom_boxplot() + scale_x_discrete(limits = c("baseline",
    "task1", "recovery"), labels = c("Baseline",
    "Task (1)", "Recovery")) + scale_fill_brewer(type = "div",
    palette = 5) + labs(x = "Study phase",
    y = "Score - Pleasure") + theme(panel.grid.major.x = element_blank()) +
    geom_vline(xintercept = c(1.5, 2.5),
        color = "gray")
```




```
# Export plot
# ggsave('output/unusual_pleasure.png',
```

```
# width = 15, height = 7.5, units =
# 'cm')

# Arousal for participants with unusual
# and usual responses
```

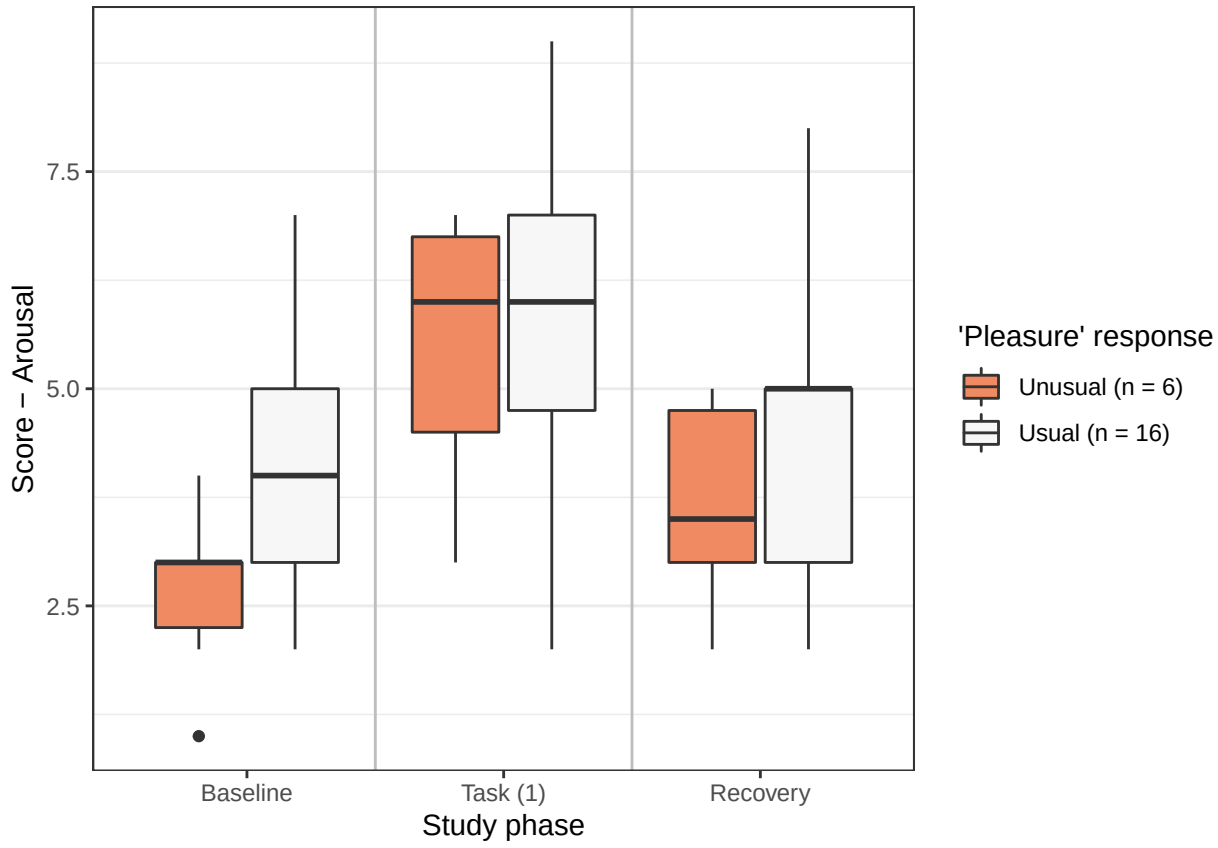
We compare subjective arousal scores between the usual and unusual (pleasure) responders across the study phases.

```
# Summary statistics
affect_arousal %>%
    my_summary(outcome_var = affect_value,
        group_var1 = phase, group_var2 = response)  # %>%
```

Phase	Response	n	Min	Max	Median	IQR	Mean	SD
baseline	Unusual	6	1	4	3.0	0.75	2.67	1.03
baseline	Usual	16	2	7	4.0	2.00	4.12	1.50
task1	Unusual	6	3	7	6.0	2.25	5.50	1.64
task1	Usual	16	2	9	6.0	2.25	5.62	1.93
recovery	Unusual	6	2	5	3.5	1.75	3.67	1.21
recovery	Usual	16	2	8	5.0	2.00	4.50	1.67

IQR = Interquartile range; SD = Standard deviation

```
# save_as_docx(path =
# 'output/Table_S2.docx')

# Plot - Comparison of usual and
# unusual (pleasure) responders on
# subjective arousal
affect_arousal %>%
    mutate(Response = if_else(p_id %in% both_dif_unusual,
        "Unusual (n = 6)", "Usual (n = 16)")) %>%
    ggplot(aes(phase, affect_value, fill = Response)) +
    geom_boxplot() + scale_x_discrete(limits = c("baseline",
    "task1", "recovery"), labels = c("Baseline",
    "Task (1)", "Recovery")) + scale_fill_brewer(type = "div",
    palette = 5) + labs(x = "Study phase",
    y = "Score - Arousal", fill = "'Pleasure' response") +
    theme(panel.grid.major.x = element_blank()) +
    geom_vline(xintercept = c(1.5, 2.5),
        color = "gray")
```




```
# Export plot
# ggsave('output/unusual_arousal.png',
# width = 15, height = 7.5, units =
# 'cm')

# Comparison of usual and unusual
# (pleasure) responders on subjective
# arousal with Wilcoxon tests

#####

## Create a function that will extract
## comparisons from the output
my_comparisons <- function(df, within = TRUE) {
    df <- df %>%
        mutate(p = as.character(p), across(where(is.numeric),
            round, 2), p = as.numeric(p),
            across(where(is.numeric), round,
                3), p.signif = case_when(p <
                0.001 ~ "***", p < 0.01 ~
                "**", p < 0.05 ~ "*", TRUE ~
                NA_character_), across(.cols = c(estimate,
                conf.low, conf.high), ~formatC(.x,
                digits = 2, format = "f")),
            estimate = paste0(estimate, " [",
                conf.low, ", ", conf.high,
```

```
                "]"), p = formatC(p, digits = 3,
                format = "f"), p = if_else(p ==
                "0.000", "< 0.001", p), p = paste0(p,
                p.signif), p = str_remove_all(p,
                "NA"))

    df <- df %>%
        mutate(across(.cols = matches("group|response"),
            ~str_remove_all(.x, "\\(n = 6\\)|\\(n = 16\\)")))

    df <- df %>%
        select(-.y., -conf.low, -conf.high,
            -method, -alternative) %>%
        select(matches("phase|parameter|response"),
            group1, group2, n1, n2, estimate,
            statistic, matches("df"), p) %>%
        # reorder variables
    rename(`Group 1` = group1, `Group 2` = group2,
        `Estimate [95% CI]` = estimate, Statistic = statistic)

    df <- df %>%
        flextable() %>%
        autofit() %>%
        add_footer(phase = "CI = confidence interval") %>%
        merge_at(j = 1:5, part = "footer")

    df <- fit_flextable_to_page(df)

    df
}

#####

## Comparisons
affect_arousal %>%
    mutate(Response = if_else(p_id %in% both_dif_unusual,
        "Unusual (n = 6)", "Usual (n = 16)")) %>%
    group_by(phase, affect_state) %>%
    wilcox_test(affect_value ~ Response,
        detailed = TRUE) %>%
    my_comparisons()  # %>%
```

phase	Group 1	Group 2	n1	n2	Estimate [95% CI]	Statistic	p
baseline	Unusual	Usual	6	16	-1.00 [-3.00, 0.00]	21.5	0.046*
task1	Unusual	Usual	6	16	-0.00 [-2.00, 2.00]	46.5	0.940
recovery	Unusual	Usual	6	16	-1.00 [-2.00, 1.00]	34.0	0.304

CI = confidence interval

```
# save_as_docx(path =
# 'output/Table_S3.docx')
```

Physiological activity

Cardiovascular activity

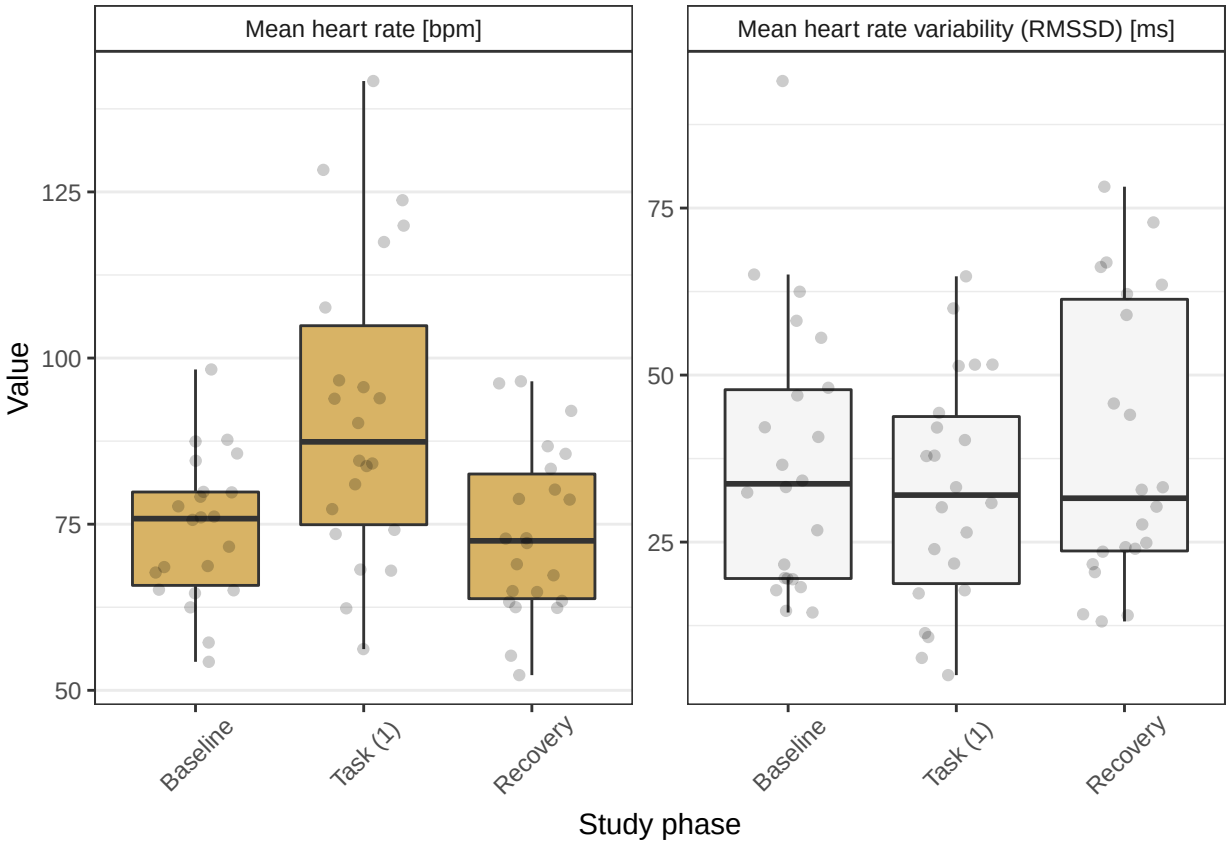
We plot changes in heart rate and heart rate variability throughout the study phases.

```
# Transform a selection of variables to
# the long format
physio_smr_filt_long <- physio_smr_filt %>%
    select(p_id, condition, phase, hr_bpm:gsr_us,
        rmssd) %>%
    pivot_longer(cols = c(hr_bpm, gsr_us,
        rmssd), names_to = "parameter", values_to = "parameter_value")

# Remove problematic subject 12 from
# EDA data for the plot
physio_smr_filt_no_12_long <- physio_smr_filt_long %>%
    filter(!(p_id == 12 & parameter == "gsr_us"))

# Compute means and CIs
physio_smr_filt_no_12_long %>%
    filter(parameter != "gsr_us") %>%
    mean_ci(parameter_value, group_by = parameter)
```

```
## # A tibble: 2 x 5
##   group_by variable            n  mean ci95
##   <chr>    <chr>           <dbl> <dbl> <chr>
## 1 hr_bpm   parameter_value    66  80.0 75.47 84.43
## 2 rmssd    parameter_value    66  36.4 31.59 41.23
```

```
## Create labels for the plot
cardio_labels <- labeller(parameter = c(hr_bpm = "Mean heart rate [bpm]",
    rmssd = "Mean heart rate variability (RMSSD) [ms]"))

############

# Physiological activity throughout the
# study

## Boxplot - Heart rate and heart rate
## variability
physio_smr_filt_no_12_long %>%
    filter(parameter != "gsr_us") %>%
    ggplot(aes(phase, parameter_value, fill = parameter)) +
    geom_boxplot(outlier.shape = NA) + geom_jitter(width = 0.25,
    alpha = 0.2) + facet_wrap(~parameter,
    scales = "free", labeller = cardio_labels) +
    guides(fill = FALSE) + scale_fill_brewer(type = "div",
    palette = 1) + scale_x_discrete(limits = c("baseline",
```

```
    "task1", "recovery"), labels = c("Baseline",
    "Task (1)", "Recovery")) + labs(x = "Study phase",
    y = "Value") + theme(axis.text.x = element_text(angle = 45,
    vjust = 0.6), panel.grid.major.x = element_blank(),
    strip.background = element_rect(fill = "white"))
```




```
# Export plot
# ggsave('output/cardio_plot.png',
# width = 15, height = 7.5, units =
# 'cm')
```

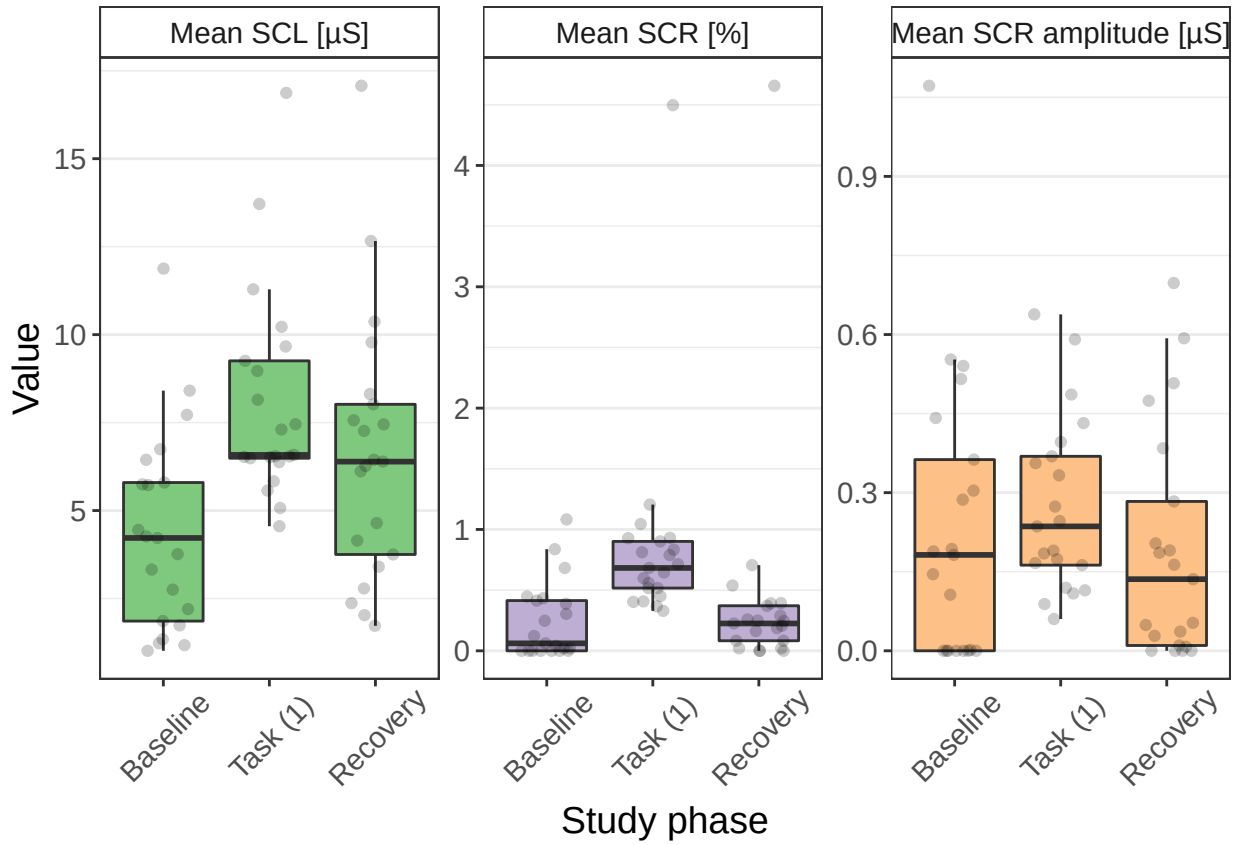
Electrodermal activity

We plot EDA parameters (skin conductance level, frequency of skin conductance responses, and the amplitude of skin conductance responses) throughout the study phases.

```
# Create a tibble in the long format
# for the plot
eda_plot <- eda_full_time_filt_no12_smr %>%
    pivot_longer(c(mean_eda_tonic, mean_scr_peaks_perc,
        mean_scr_amplitude), names_to = "eda_parameter",
        values_to = "eda_value")

# Compute means and CIs
mean_ci(eda_plot, eda_value, group_by = eda_parameter)
```

```
## # A tibble: 3 x 5
##   group_by            variable      n  mean ci95
##   <chr>               <chr>     <dbl> <dbl> <chr>
## 1 mean_eda_tonic      eda_value    63  6.35 5.46 7.24
## 2 mean_scr_amplitude  eda_value    63  0.23 0.17 0.29
## 3 mean_scr_peaks_perc eda_value    63  0.51 0.31 0.71
```

```
# Create an ordered factor for the plot
eda_plot$eda_parameter <- factor(eda_plot$eda_parameter,
    levels = c("mean_eda_tonic", "mean_scr_peaks_perc",
        "mean_scr_amplitude"))

## Create labels for the plot
eda_labels <- labeller(eda_parameter = c(mean_eda_tonic = "Mean SCL [µS]",
    mean_scr_peaks_perc = "Mean SCR [%]",
    mean_scr_amplitude = "Mean SCR amplitude [µS]"))

## Boxplot - EDA parameters
ggplot(data = eda_plot, aes(phase, eda_value,
    fill = eda_parameter)) + geom_boxplot(outlier.shape = NA) +
    geom_jitter(width = 0.25, alpha = 0.2) +
    facet_wrap(~eda_parameter, scales = "free",
        labeller = eda_labels) + guides(fill = FALSE) +
    scale_fill_brewer(type = "qual", palette = 1) +
    scale_x_discrete(limits = c("baseline",
        "task1", "recovery"), labels = c("Baseline",
        "Task (1)", "Recovery")) + labs(x = "Study phase",
    y = "Value") + theme(text = element_text(size = 14),
    axis.text.x = element_text(angle = 45,
        vjust = 0.6), panel.grid.major.x = element_blank(),
    strip.background = element_rect(fill = "white"))
```




```
# Export plot
# ggsave('output/eda_plot.png', width =
# 20, height = 12, units = 'cm')
```

Physiological activity of unusual vs usual 'pleasure' responders

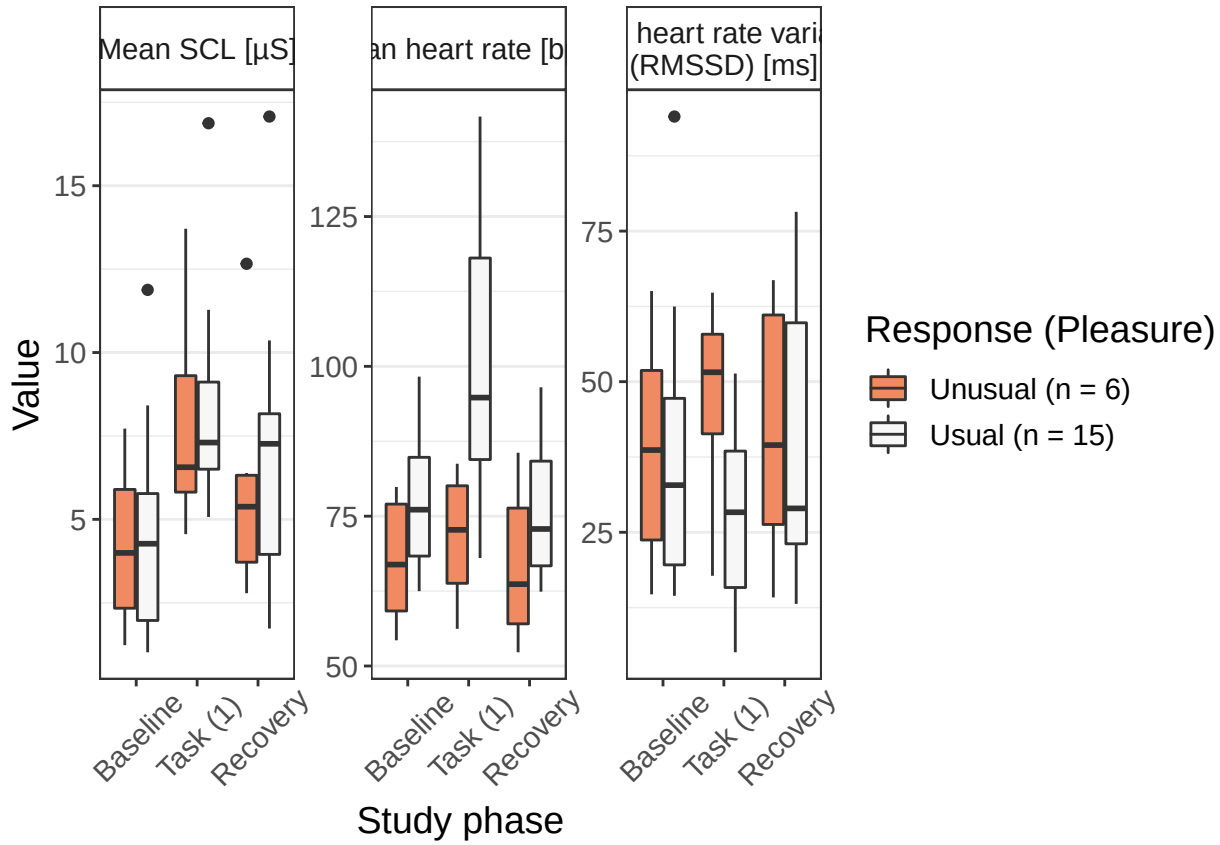
We compare physiological activity between participants with unusual and usual responses on the affective state of pleasure.

```
# Create labels for the plot
physio_labels <- labeller(parameter = c(gsr_us = "Mean SCL [µS]",
    hr_bpm = "Mean heart rate [bpm]", rmssd = "Mean heart rate variability\n(RMSSD) [ms]"))

## Comparison of usual and unusual
## (pleasure) responders on
## physiological activity

# Summary statistics
physio_smr_filt_no_12_long %>%
    my_summary(outcome_var = parameter_value,
        group_var1 = phase, group_var2 = response,
        group_var3 = parameter)  # %>%
```

Phase	Parameter	Response	n	Min	Max	Median	IQR	Mean	SD
baseline	gsr_us	Unusual	6	1.23	7.72	3.99	3.56	4.21	2.53
baseline	hr_bpm	Unusual	6	54.29	79.86	66.92	17.84	67.50	10.88
baseline	rmssd	Unusual	6	14.70	65.05	38.65	28.15	38.68	19.67
baseline	gsr_us	Usual	15	1.01	11.88	4.27	3.80	4.44	3.05
baseline	hr_bpm	Usual	16	62.49	98.28	76.09	16.48	76.77	10.03
baseline	rmssd	Usual	16	14.46	94.01	32.83	27.64	36.86	21.31
task1	gsr_us	Unusual	6	4.56	13.71	6.56	3.49	7.86	3.44
task1	hr_bpm	Unusual	6	56.20	83.75	72.72	16.27	71.46	10.98
task1	rmssd	Unusual	6	17.78	64.78	51.57	16.53	47.27	17.10
task1	gsr_us	Usual	15	5.07	16.87	7.30	2.61	8.15	2.93
task1	hr_bpm	Usual	16	68.01	141.68	94.79	33.63	99.59	21.52
task1	rmssd	Usual	16	5.08	51.36	28.33	22.64	27.17	14.08
recovery	gsr_us	Unusual	6	2.79	12.66	5.38	2.60	6.00	3.56
recovery	hr_bpm	Unusual	6	52.29	85.59	63.66	19.32	66.77	13.42
recovery	rmssd	Unusual	6	14.19	66.85	39.48	34.77	41.70	21.87
recovery	gsr_us	Usual	15	1.73	17.07	7.26	4.22	6.84	3.94
recovery	hr_bpm	Usual	16	62.40	96.50	72.86	17.47	76.29	11.70
recovery	rmssd	Usual	16	13.12	78.19	28.97	36.69	38.28	21.71

IQR = Interquartile range; SD = Standard deviation

```
# save_as_docx(path =
# 'output/Table_S4.docx')

# Plot
physio_smr_filt_no_12_long %>%
    mutate(`Response (Pleasure)` = if_else(p_id %in%
        both_dif_unusual, "Unusual (n = 6)",
        "Usual (n = 15)")) %>%
    ggplot(aes(phase, parameter_value, fill = `Response (Pleasure)`)) +
    geom_boxplot() + facet_wrap(~parameter,
    scales = "free", labeller = physio_labels) +
    scale_fill_brewer(type = "div", palette = 5) +
    scale_x_discrete(limits = c("baseline",
        "task1", "recovery"), labels = c("Baseline",
        "Task (1)", "Recovery")) + labs(x = "Study phase",
    y = "Value") + theme(text = element_text(size = 14),
    axis.text.x = element_text(angle = 45,
        vjust = 0.6), panel.grid.major.x = element_blank(),
    strip.background = element_rect(fill = "white"))
```




```
# Export plot
# ggsave('output/unusual_physio.png',
# width = 23.5, height = 12, units =
# 'cm')

# Comparison of usual and unusual
# (pleasure) responders on
# physiological activity with Wilcoxon
# tests
physio_smr_filt_no_12_long %>%
    mutate(Response = if_else(p_id %in% both_dif_unusual,
        "Unusual (n = 6)", "Usual (n = 16)")) %>%
    group_by(Response, parameter) %>%
    wilcox_test(parameter_value ~ phase,
        detailed = TRUE, paired = TRUE) %>%
    my_comparisons()  # %>%
```

parameter	Response	Group 1	Group 2	n1	n2	Estimate [95% CI]	Statistic	p
gsr_us	Unusual	baseline	task1	6	6	-3.52 [-5.99, -2.33]	0	0.031*
gsr_us	Unusual	baseline	recovery	6	6	-1.55 [-4.94, 0.07]	1	0.062
gsr_us	Unusual	task1	recovery	6	6	1.85 [0.43, 3.83]	21	0.031*

parameter	Response	Group 1	Group 2	n1	n2	Estimate [95% CI]	Statistic	p
hr_bpm	Unusual	baseline	task1	6	6	-3.14 [-12.13, 0.53]	1	0.062
hr_bpm	Unusual	baseline	recovery	6	6	1.75 [-5.72, 3.91]	14	0.563
hr_bpm	Unusual	task1	recovery	6	6	3.91 [-4.58, 14.75]	17	0.219
rmssd	Unusual	baseline	task1	6	6	-7.62 [-28.21, 13.48]	4	0.219
rmssd	Unusual	baseline	recovery	6	6	-3.18 [-10.61, 7.52]	5	0.313
rmssd	Unusual	task1	recovery	6	6	6.07 [-15.28, 19.04]	15	0.438
gsr_us	Usual	baseline	task1	15	15	-3.65 [-4.59, -2.84]	0	< 0.001***
gsr_us	Usual	baseline	recovery	15	15	-2.38 [-3.38, -1.32]	0	< 0.001***
gsr_us	Usual	task1	recovery	15	15	1.32 [0.29, 2.23]	102	0.015*
hr_bpm	Usual	baseline	task1	16	16	-22.61 [-31.69, -11.55]	0	< 0.001***
hr_bpm	Usual	baseline	recovery	16	16	0.45 [-1.29, 2.63]	79	0.597
hr_bpm	Usual	task1	recovery	16	16	22.94 [14.45, 31.56]	136	< 0.001***
rmssd	Usual	baseline	task1	16	16	9.06 [-0.18, 19.19]	105	0.058
rmssd	Usual	baseline	recovery	16	16	0.23 [-7.40, 4.31]	71	0.900
rmssd	Usual	task1	recovery	16	16	-10.37 [-19.23, -1.15]	26	0.029*

```
# save_as_docx(path = 'output/Table
# S5.docx')
```

Mental arithmetic task

We plot the results of Mental arithmetic task: number of total responses provided and the proportion of correct responses.

Overall results

```
# Compute mean and 95% confidence
# interval

## Total responses provided
task_long_filt_n_responses <- task_long_filt %>%
```

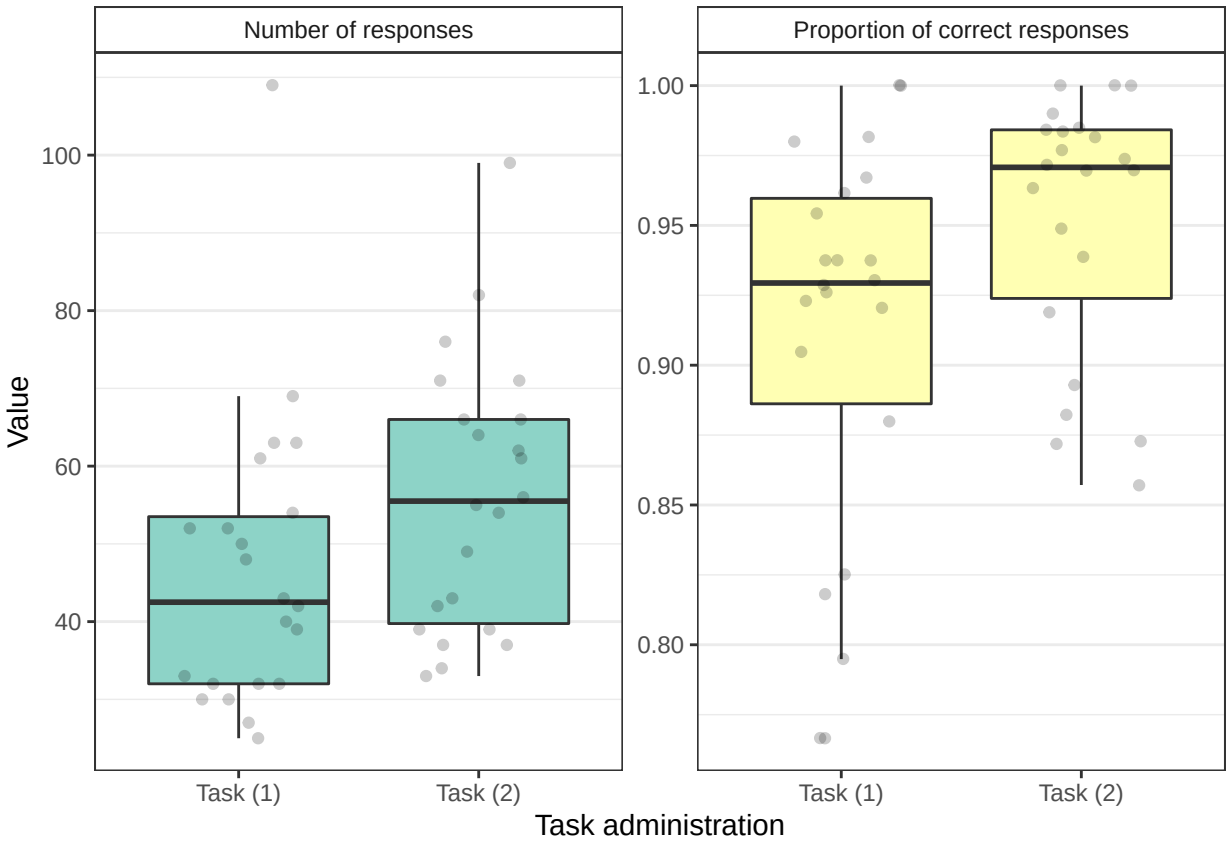
```
    filter(parameter == "n_responses") %>%
    ungroup()

mean_ci(task_long_filt_n_responses, value)
```

```
## # A tibble: 1 x 4
##   variable     n  mean ci95
##   <chr>    <dbl> <dbl> <chr>
## 1 value       44  51.4 45.72 57.1
```

```
## Number of correct responses
mean_ci(task_raw_long_filt, correct)
```

```
## # A tibble: 1 x 4
##   variable     n  mean ci95
##   <chr>    <dbl> <dbl> <chr>
## 1 correct   2262  0.94 0.93 0.95
```

```
# Plot

## Create labels for the plot
task_labels <- labeller(parameter = c(n_responses = "Number of responses",
    prop_correct = "Proportion of correct responses"))

## Boxplot - MAT results
ggplot(data = task_long_filt, aes(phase,
    value, fill = parameter)) + geom_boxplot(outlier.shape = NA) +
    geom_jitter(width = 0.25, alpha = 0.2) +
    facet_wrap(~parameter, scales = "free",
        labeller = task_labels) + guides(fill = FALSE) +
    scale_fill_brewer(type = "qual", palette = 8) +
    scale_x_discrete(limits = c("task1",
        "task2"), labels = c("Task (1)",
        "Task (2)")) + labs(x = "Task administration",
    y = "Value") + theme(panel.grid.major.x = element_blank(),
    strip.background = element_rect(fill = "white"))
```




```
# Export plot
# ggsave('output/task_plot.png', width
# = 15, height = 7.5, units = 'cm')
```

Comparison of participants with usual and unusual pleasure scores

We compare the MAT results between participants with usual and unusual pleasure scores.

```
# Comparison of usual and unusual
# (pleasure) responders on MAT results

## Summary statistics
task_long_filt %>%
    my_summary(outcome_var = value, group_var1 = phase,
        group_var2 = response, group_var3 = parameter)  # %>%
```

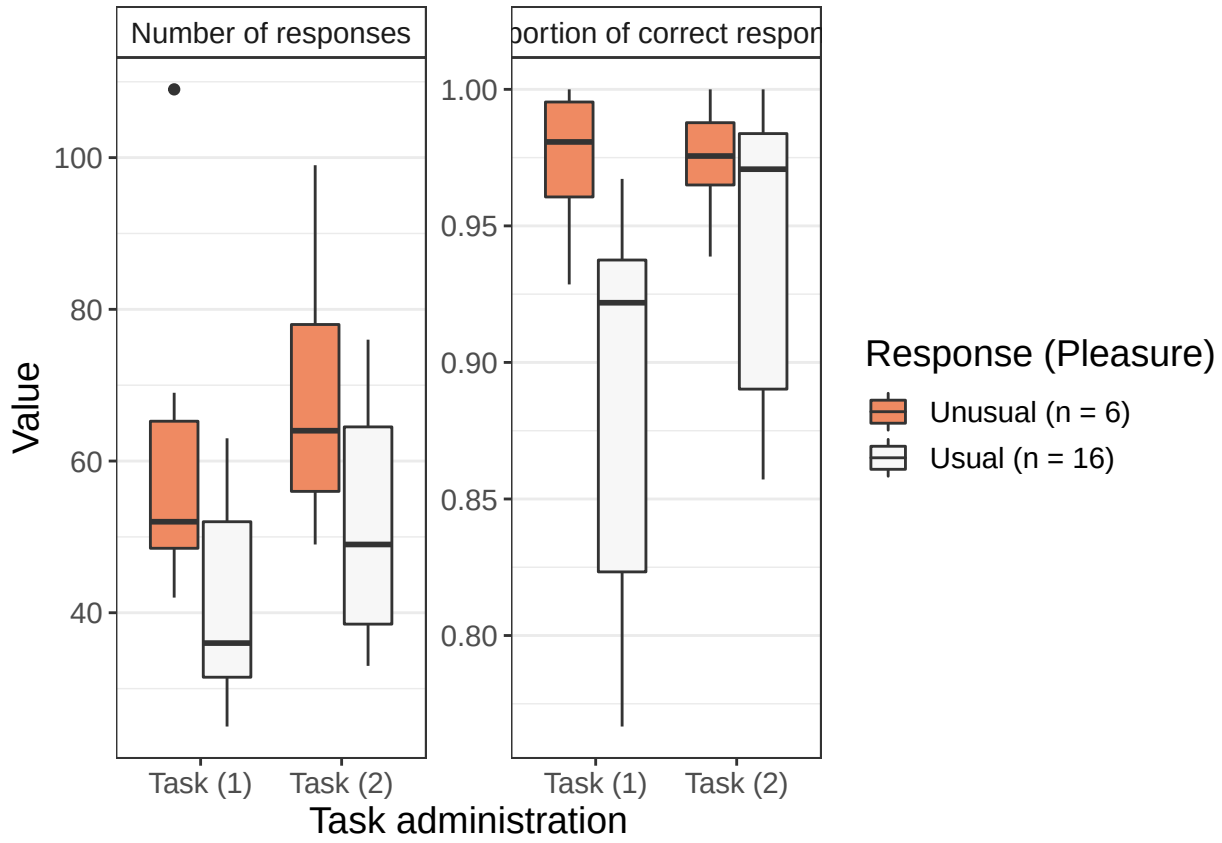
Phase	Parameter	Response	n	Min	Max	Median	IQR	Mean	SD
task1	n_responses	Unusual	6	42.00	109.00	52.00	16.75	62.00	24.75
task1	prop_correct	Unusual	6	0.93	1.00	0.98	0.04	0.97	0.03
task1	n_responses	Usual	16	25.00	63.00	36.00	20.50	40.88	13.22

IQR = Interquartile range; SD = Standard deviation

Phase	Parameter	Response	n	Min	Max	Median	IQR	Mean	SD
task1	prop_correct	Usual	16	0.77	0.97	0.92	0.11	0.89	0.07
task2	n_responses	Unusual	6	49.00	99.00	64.00	22.00	68.67	18.72
task2	prop_correct	Unusual	6	0.94	1.00	0.98	0.02	0.97	0.02
task2	n_responses	Usual	16	33.00	76.00	49.00	26.00	51.50	15.04
task2	prop_correct	Usual	16	0.86	1.00	0.97	0.09	0.94	0.05

IQR = Interquartile range; SD = Standard deviation

```
# save_as_docx(path =
# 'output/Table_S6.docx')

## Plot
task_long_filt %>%
    mutate(`Response (Pleasure)` = if_else(p_id %in%
        both_dif_unusual, "Unusual (n = 6)",
        "Usual (n = 16)")) %>%
    ggplot(aes(phase, value, fill = `Response (Pleasure)`)) +
    facet_wrap(~parameter, scales = "free",
        labeller = task_labels) + geom_boxplot() +
    scale_fill_brewer(type = "div", palette = 5) +
    scale_x_discrete(limits = c("task1",
        "task2"), labels = c("Task (1)",
        "Task (2)")) + labs(x = "Task administration",
    y = "Value") + theme(text = element_text(size = 14),
    panel.grid.major.x = element_blank(),
    strip.background = element_rect(fill = "white"))
```




```
# Export plot
# ggsave('output/unusual_mat.png',
# width = 20, height = 10, units =
# 'cm')

# Comparison of usual and unusual
# (pleasure) responders on MAT results
# with Wilcoxon tests
task_long_filt %>%
    mutate(Response = if_else(p_id %in% both_dif_unusual,
        "Unusual (n = 6)", "Usual (n = 16)")) %>%
    group_by(phase, parameter) %>%
    wilcox_test(value ~ Response, detailed = TRUE) %>%
    my_comparisons()
```

phase	parameter	Group 1	Group 2	n1	n2	Estimate [95% CI]	Statistic	p
task1	n_responses	Unusual	Usual	6	16	17.00 [2.00, 37.00]	77.0	0.035*
task1	prop_correct	Unusual	Usual	6	16	0.06 [0.03, 0.16]	88.0	0.004**
task2	n_responses	Unusual	Usual	6	16	16.87 [-4.00, 33.00]	71.5	0.090

CI = confidence interval

phase	parameter	Group 1	Group 2	n1	n2	Estimate [95% CI]	Statistic	p
task2	prop_correct	Unusual	Usual	6	16	0.02 [-0.01, 0.09]	60.5	0.376

CI = confidence interval

Modelling

We proceed with modeling using (generalized) linear mixed models. The modeling approach is similar across different outcomes. Where needed, we start with slight transformations of tibbles to prepare them for modeling, and then fit the model and compute model summary. Where relevant, we continue with post-hoc comparisons by computing estimated marginal means. Model diagnostics includes simulation and plots of residuals. The plots include results of calculations assessing model fit.

Affective states

We fit two linear mixed models to analyze affective states: we start with arousal scores as the dependent variable and continue with pleasure scores as the dependent variable.

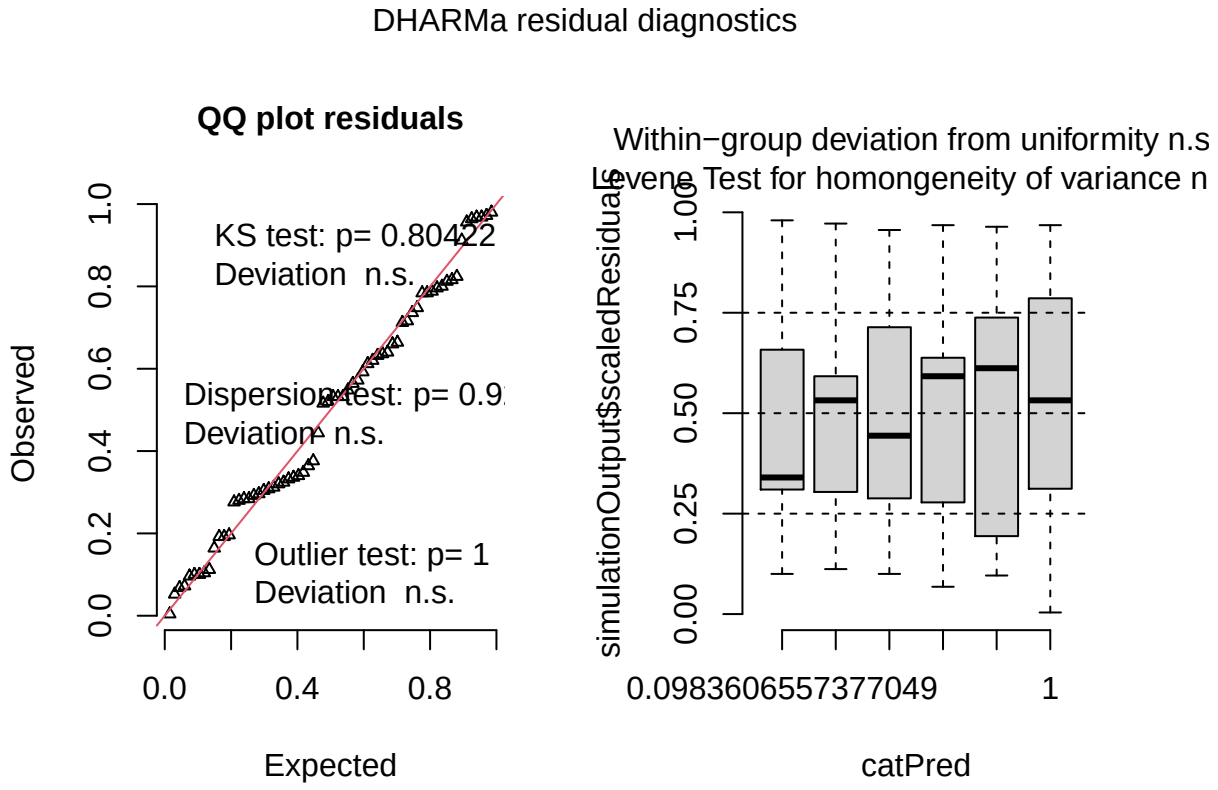
Arousal

```
#####

# Create a function that extracts data
# from the output of the model
tidy_lmer <- function(model, glmer = FALSE,
    exp = FALSE) {
    l <- tidy(model, conf.int = TRUE, exponentiate = exp,
        effects = "fixed")
    l <- l %>%
        mutate(p.value = if_else(p.value <
            0.001, "< 0.001", as.character(formatC(round(p.value,
            3), format = "f", digits = 3))))
    l <- l %>%
        mutate(across(where(is.numeric),
            round, 2), ci95 = paste(conf.low,
            "to", conf.high), .after = std.error) %>%
        select(-conf.low, -conf.high)
    if (!(glmer)) {
        l <- l %>%
            relocate(df, .before = statistic)
    } else {
        l
    }
    l
}

#####
```

```r
# Fit the linear mixed model
model_arousal <- lmer(affect_value ~ condition *
    phase + (1 | p_id), data = affect_arousal)

## Model summary
model_arousal_smr <- summary(model_arousal)

# Post-hoc comparisons
model_arousal_emmeans <- emmeans(model_arousal,
    specs = pairwise ~ phase)
confint(model_arousal_emmeans)
```

```
## $emmeans
##  phase    emmean    SE   df lower.CL upper.CL
##  baseline   3.73 0.356 35.2     3.01     4.45
##  task1      5.59 0.356 35.2     4.87     6.31
##  recovery   4.27 0.356 35.2     3.55     4.99
##
## Results are averaged over the levels of: condition
## Degrees-of-freedom method: kenward-roger
## Confidence level used: 0.95
##
## $contrasts
##  contrast            estimate    SE df lower.CL upper.CL
##  baseline - task1      -1.864 0.321 40   -2.644   -1.083
##  baseline - recovery   -0.545 0.321 40   -1.326    0.235
##  task1 - recovery       1.318 0.321 40    0.538    2.098
##
## Results are averaged over the levels of: condition
## Degrees-of-freedom method: kenward-roger
## Confidence level used: 0.95
## Conf-level adjustment: tukey method for comparing a family of 3 estimates
```

```r
# Model diagnostics

## Simulate residuals
model_arousal_residuals <- simulateResiduals(fittedModel = model_arousal,
    plot = F)

## Plot
plot(model_arousal_residuals)
```



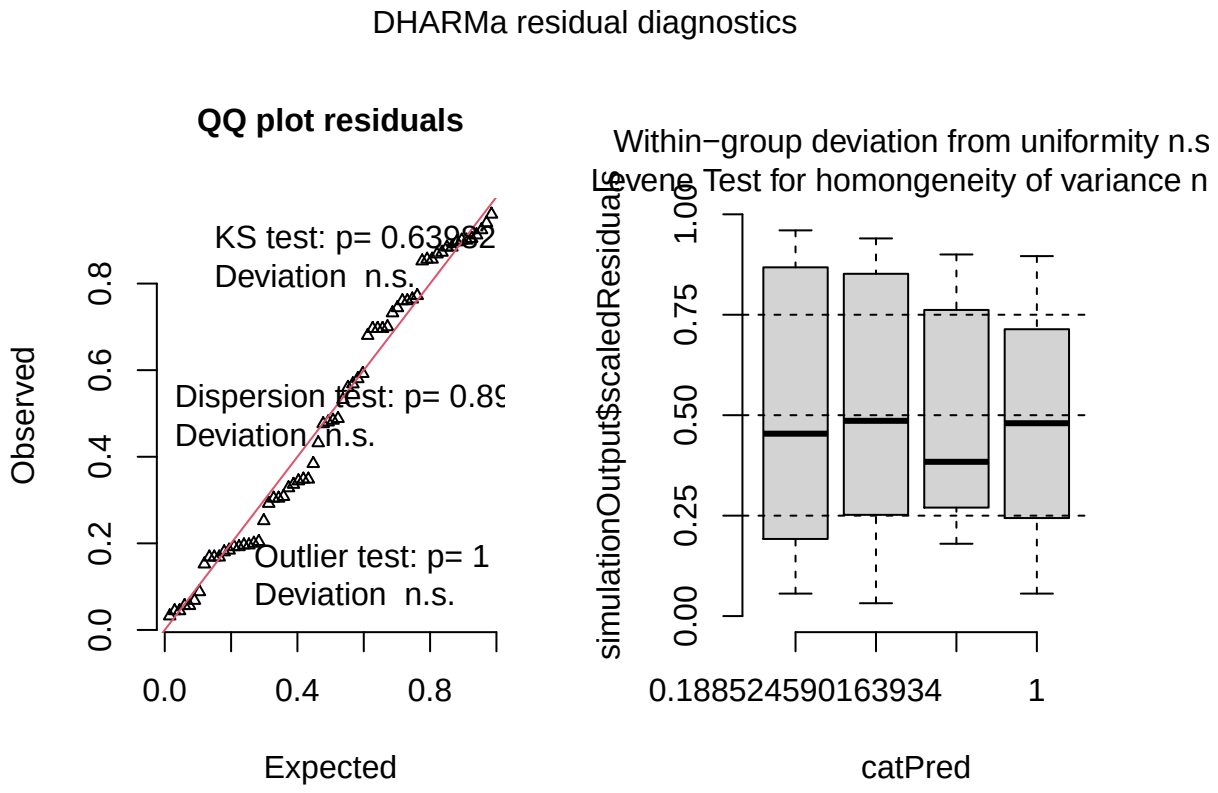

```
# Extract model output for publishing
tidy_lmer(model_arousal)
```

```
## # A tibble: 6 x 8
##   effect term                estimate std.error ci95         df statistic p.value
##   <chr>  <chr>                  <dbl>     <dbl> <chr>     <dbl>     <dbl> <chr>
## 1 fixed  (Intercept)             3.82      0.5  2.8 to 4~  35.2      7.59 < 0.001
## 2 fixed  conditionwood          -0.18      0.71 -1.63 to~  35.2     -0.26 0.800
## 3 fixed  phasetask1              2         0.45 1.08 to ~  40        4.41 < 0.001
## 4 fixed  phaserecovery           0.73      0.45 -0.19 to~  40        1.6  0.117
## 5 fixed  conditionwood:pha~     -0.27      0.64 -1.57 to~  40       -0.43 0.673
## 6 fixed  conditionwood:pha~     -0.36      0.64 -1.66 to~  40       -0.57 0.574
```

Pleasure

```
# Fit the linear mixed model
model_pleasure <- lmer(affect_value ~ condition *
    phase + (1 | p_id), data = affect_pleasure)

## Model summary
model_pleasure_smr <- summary(model_pleasure)

# Model diagnostics
```

```
## Simulate residuals
model_pleasure_residuals <- simulateResiduals(fittedModel = model_pleasure,
    plot = F)

## Plot
plot(model_pleasure_residuals)
```




```
# Extract model output for publishing
tidy_lmer(model_pleasure)
```

```
## # A tibble: 6 x 8
##   effect term                estimate std.error ci95         df statistic p.value
##   <chr>  <chr>                  <dbl>     <dbl> <chr>     <dbl>     <dbl> <chr>
## 1 fixed  (Intercept)             5.64      0.56 4.51 to ~  44.1      10.1  < 0.001
## 2 fixed  conditionwood           0.18      0.79 -1.4 to ~  44.1       0.23 0.818
## 3 fixed  phasetask1             -0.91      0.6  -2.12 to~  40        -1.52 0.136
## 4 fixed  phaserecovery          -0.91      0.6  -2.12 to~  40        -1.52 0.136
## 5 fixed  conditionwood:pha~      0.09      0.84 -1.61 to~  40         0.11 0.915
## 6 fixed  conditionwood:pha~      0.09      0.84 -1.61 to~  40         0.11 0.915
```

Mental Arithmetic Task

We again fit two models to analyze results on the Mental Arithmetic Task. The linear mixed model takes the number of total responses as the dependent variable, and the binomial (generalized) linear model takes

the correctness of responses (0 or 1) as the dependent variable.

Number of total responses

```
# Fit the linear mixed model
model_task_n_responses <- lmer(value ~ condition *
    phase + (1 | p_id), data = task_long_filt_n_responses)

## Model summary
summary(model_task_n_responses)
```

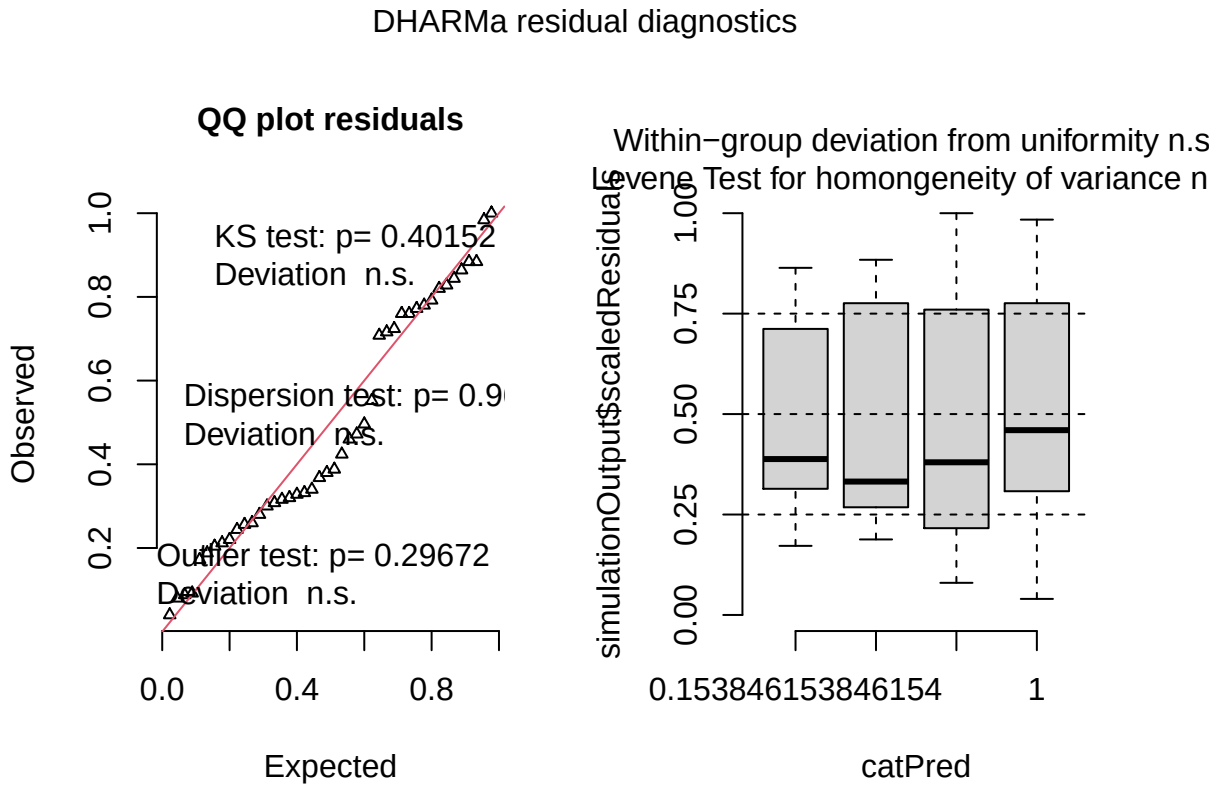
```
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: value ~ condition * phase + (1 | p_id)
##    Data: task_long_filt_n_responses
##
## REML criterion at convergence: 318
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -1.56964 -0.36513 -0.03833  0.38175  2.23366
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  p_id     (Intercept) 276.35   16.624
##  Residual              29.34    5.416
## Number of obs: 44, groups:  p_id, 22
##
## Fixed effects:
##                           Estimate Std. Error      df t value      Pr(>|t|)
## (Intercept)                 53.091      5.272  22.011  10.071 0.00000000106 ***
## conditionwood              -12.909      7.455  22.011  -1.732      0.097343 .
## phasetask2                   9.545      2.310  20.000   4.133      0.000515 ***
## conditionwood:phasetask2     0.000      3.266  20.000   0.000      1.000000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) cndtnw phsts2
## conditionwd -0.707
## phasetask2  -0.219  0.155
## cndtnwd:ph2  0.155 -0.219 -0.707
```

```
# Post-hoc comparisons
model_task_n_responses_emmeans <- emmeans(model_task_n_responses,
    specs = pairwise ~ phase, type = "response")
confint(model_task_n_responses_emmeans)
```

```
## $emmeans
##  phase emmean   SE df lower.CL upper.CL
##  task1   46.6 3.73 22     38.9     54.4
```

```
##  task2   56.2 3.73 22     48.5     63.9
##
## Results are averaged over the levels of: condition
## Degrees-of-freedom method: kenward-roger
## Confidence level used: 0.95
##
## $contrasts
##  contrast      estimate   SE df lower.CL upper.CL
##  task1 - task2    -9.55 1.63 20      -13    -6.14
##
## Results are averaged over the levels of: condition
## Degrees-of-freedom method: kenward-roger
## Confidence level used: 0.95
```

```r
# Model diagnostics

## Simulate residuals
model_task_n_responses_residuals <- simulateResiduals(fittedModel = model_task_n_responses,
    plot = F)

## Plot
plot(model_task_n_responses_residuals)
```




```r
# Extract model output for publishing
tidy_lmer(model_task_n_responses)
```

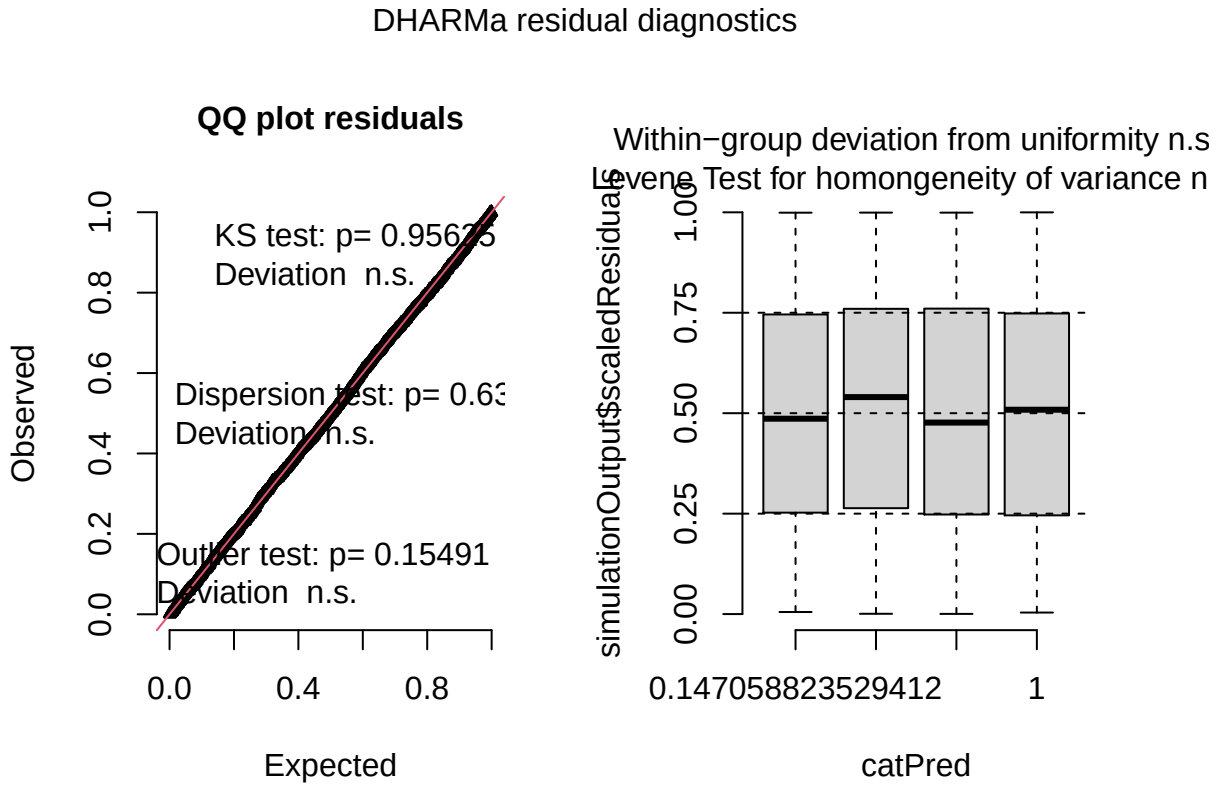
```
## # A tibble: 4 x 8
##   effect term                     estimate std.error ci95     df statistic p.value
##   <chr>  <chr>                       <dbl>     <dbl> <chr> <dbl>     <dbl> <chr>
## 1 fixed  (Intercept)                  53.1       5.27 42.1~  22.0     10.1  < 0.001
## 2 fixed  conditionwood               -12.9       7.46 -28.~  22.0     -1.73 0.097
## 3 fixed  phasetask2                    9.55      2.31 4.73~  20       4.13 < 0.001
## 4 fixed  conditionwood:phasetask2      0         3.27 -6.8~  20       0    1.000
```

Correct responses

```
# Fit the generalized linear mixed
# model
model_task_correct <- glmer(correct ~ condition *
    phase + (1 | p_id), data = task_raw_long_filt,
    family = "binomial")

## Model summary
model_task_correct_smr <- summary(model_task_correct)

# Post-hoc comparisons
model_task_correct_emmeans <- emmeans(model_task_correct,
    specs = pairwise ~ condition, type = "response")
confint(model_task_correct_emmeans)
```

```
## $emmeans
##  condition  prob     SE  df asymp.LCL asymp.UCL
##  white     0.946 0.0137 Inf     0.912     0.968
##  wood      0.953 0.0124 Inf     0.922     0.972
##
## Results are averaged over the levels of: phase
## Confidence level used: 0.95
## Intervals are back-transformed from the logit scale
##
## $contrasts
##  contrast     odds.ratio    SE  df asymp.LCL asymp.UCL
##  white / wood      0.862 0.333 Inf     0.404      1.84
##
## Results are averaged over the levels of: phase
## Confidence level used: 0.95
## Intervals are back-transformed from the log odds ratio scale
```

```
# Model diagnostics

## Simulate residuals
model_task_correct_simulated_residuals <- simulateResiduals(fittedModel = model_task_correct,
    plot = F)

## Plot
plot(model_task_correct_simulated_residuals)
```




```
# Extract model output for publishing
tidy_lmer(model_task_correct, glmer = TRUE,
    exp = TRUE)
```

```
## # A tibble: 4 x 7
##   effect term                      estimate std.error ci95      statistic p.value
##   <chr>  <chr>                        <dbl>     <dbl> <chr>         <dbl> <chr>
## 1 fixed  (Intercept)                  11.9       3.37 6.79 to ~      8.7  < 0.001
## 2 fixed  conditionwood                1.31      0.55 0.58 to ~      0.65 0.513
## 3 fixed  phasetask2                   2.21      0.54 1.38 to ~      3.28 0.001
## 4 fixed  conditionwood:phasetask2     0.78      0.29 0.37 to ~     -0.66 0.512
```

Electrodermal activity

We continue with the modeling that follows the same approach as shown and described above. In this case, three EDA parameters are dependent variables (skin conductance level, skin conductance responses, and the amplitude of skin conductance responses), leading to three instances of the linear mixed model.

```
# Skin conductance level

## Fit the linear mixed model
model_scl <- lmer(mean_eda_tonic ~ condition *
    phase + (1 | p_id), data = eda_full_time_filt_no12_smr)
```

```r
## Model summary
summary(model_scl)
```

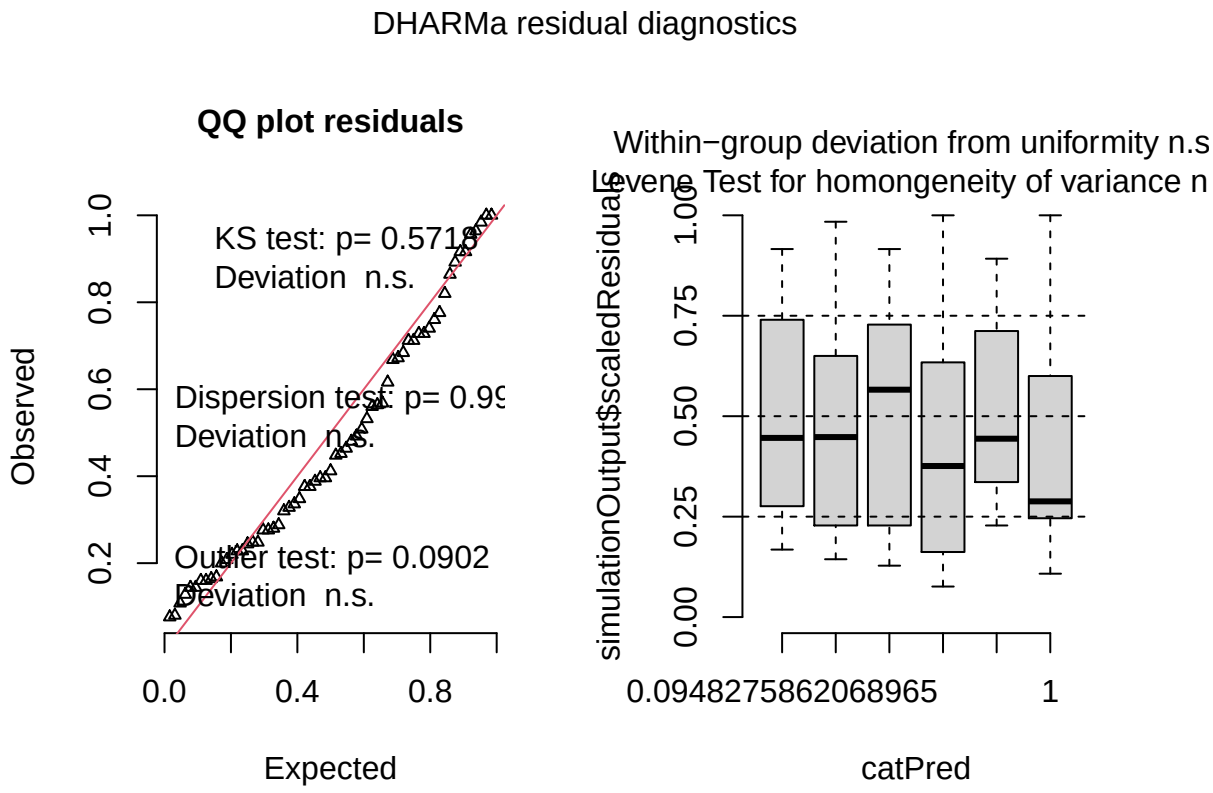
```
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: mean_eda_tonic ~ condition * phase + (1 | p_id)
##    Data: eda_full_time_filt_no12_smr
##
## REML criterion at convergence: 247.9
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -1.88886 -0.51984  0.00262  0.49896  1.83857
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  p_id     (Intercept) 9.461    3.076
##  Residual             1.224    1.107
## Number of obs: 63, groups:  p_id, 21
##
## Fixed effects:
##                              Estimate Std. Error      df t value       Pr(>|t|)
## (Intercept)                    4.6551     0.9856 22.1968   4.723       0.000101
## conditionwood                 -0.5972     1.4283 22.1968  -0.418       0.679869
## phaserecovery                  2.5981     0.4718 38.0000   5.506 0.000002705037
## phasetask1                     3.8545     0.4718 38.0000   8.170 0.000000000683
## conditionwood:phaserecovery   -0.7780     0.6837 38.0000  -1.138       0.262322
## conditionwood:phasetask1      -0.3239     0.6837 38.0000  -0.474       0.638355
##
## (Intercept)                  ***
## conditionwood
## phaserecovery                ***
## phasetask1                   ***
## conditionwood:phaserecovery
## conditionwood:phasetask1
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) cndtnw phsrcv phsts1 cndtn:
## conditionwd -0.690
## phaserecvry -0.239  0.165
## phasetask1  -0.239  0.165  0.500
## cndtnwd:phs  0.165 -0.239 -0.690 -0.345
## cndtnwd:ph1  0.165 -0.239 -0.345 -0.690  0.500
```

```r
# Post-hoc comparisons
model_scl_emmeans <- emmeans(model_scl, specs = pairwise ~
    phase, type = "response", interaction = TRUE)
confint(model_scl_emmeans)
```

```
## $emmeans
##  phase    emmean    SE   df lower.CL upper.CL
```

```
##  baseline   4.36 0.714 22.2     2.88     5.84
##  recovery   6.57 0.714 22.2     5.09     8.05
##  task1      8.05 0.714 22.2     6.57     9.53
##
## Results are averaged over the levels of: condition
## Degrees-of-freedom method: kenward-roger
## Confidence level used: 0.95
##
## $contrasts
##  phase_pairwise      estimate    SE df lower.CL upper.CL
##  baseline - recovery    -2.21 0.342 38    -2.90   -1.517
##  baseline - task1       -3.69 0.342 38    -4.38   -3.000
##  recovery - task1       -1.48 0.342 38    -2.18   -0.791
##
## Results are averaged over the levels of: condition
## Degrees-of-freedom method: kenward-roger
## Confidence level used: 0.95
```

```r
# Model diagnostics

## Simulate residuals
model_scl_residuals <- simulateResiduals(fittedModel = model_scl,
    plot = F)

## Plot
plot(model_scl_residuals)
```



```r
# Extract model output for publishing
tidy_lmer(model_scl)
```

```
## # A tibble: 6 x 8
##   effect term                estimate std.error ci95         df statistic p.value
##   <chr>  <chr>                  <dbl>     <dbl> <chr>     <dbl>     <dbl> <chr>
## 1 fixed  (Intercept)             4.66      0.99 2.61 to ~  22.2      4.72 < 0.001
## 2 fixed  conditionwood          -0.6       1.43 -3.56 to~  22.2     -0.42 0.680
## 3 fixed  phaserecovery           2.6       0.47 1.64 to ~  38        5.51 < 0.001
## 4 fixed  phasetask1              3.85      0.47 2.9 to 4~  38        8.17 < 0.001
## 5 fixed  conditionwood:pha~     -0.78      0.68 -2.16 to~  38       -1.14 0.262
## 6 fixed  conditionwood:pha~     -0.32      0.68 -1.71 to~  38       -0.47 0.638
```

```r
######

# Skin conductance responses

## Fit the linear mixed model Note that
## the outcome undergoes square root
## transformation to improve model fit
model_scr_peaks <- lmer(sqrt(mean_scr_peaks_perc) ~
    condition * phase + (1 | p_id), data = eda_full_time_filt_no12_smr)

## Model summary
summary(model_scr_peaks)
```

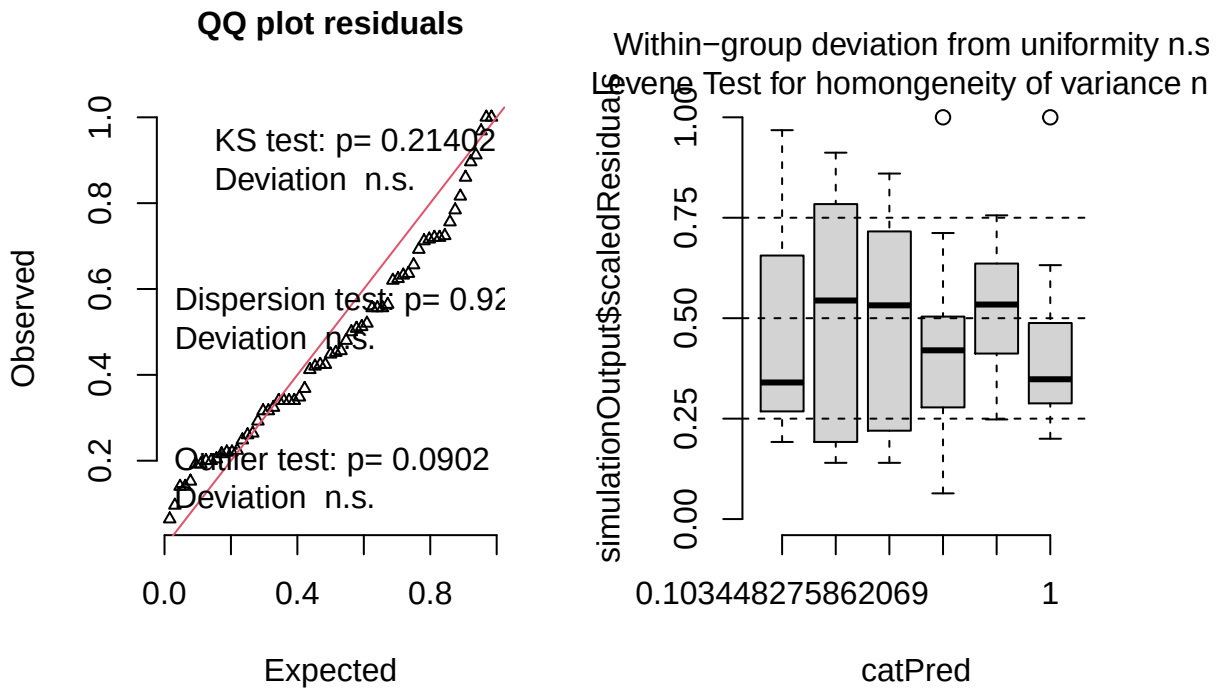
```
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: sqrt(mean_scr_peaks_perc) ~ condition * phase + (1 | p_id)
##    Data: eda_full_time_filt_no12_smr
##
## REML criterion at convergence: 39.7
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -1.81495 -0.46511 -0.07337  0.57155  2.75321
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  p_id     (Intercept) 0.10058  0.3172
##  Residual             0.04702  0.2168
## Number of obs: 63, groups:  p_id, 21
##
## Fixed effects:
##                             Estimate Std. Error       df t value    Pr(>|t|)
## (Intercept)                  0.32707    0.11584 29.55263   2.824     0.00842
## conditionwood                0.06436    0.16786 29.55263   0.383     0.70414
## phaserecovery                0.21465    0.09246 38.00000   2.322     0.02571
## phasetask1                   0.58887    0.09246 38.00000   6.369 0.000000178
## conditionwood:phaserecovery -0.17577    0.13399 38.00000  -1.312     0.19745
## conditionwood:phasetask1    -0.15000    0.13399 38.00000  -1.120     0.26993
##
## (Intercept)                 **
```

```
## conditionwood
## phaserecovery               *
## phasetask1                  ***
## conditionwood:phaserecovery
## conditionwood:phasetask1
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) cndtnw phsrcv phsts1 cndtn:
## conditionwd -0.690
## phaserecvry -0.399  0.275
## phasetask1  -0.399  0.275  0.500
## cndtnwd:phs  0.275 -0.399 -0.690 -0.345
## cndtnwd:ph1  0.275 -0.399 -0.345 -0.690  0.500
```

```
## Post-hoc comparisons Note that the
## outcome is back-transformed to
## original units
model_scr_peaks_emmeans <- emmeans(model_scr_peaks,
    specs = pairwise ~ phase, type = "response")
confint(model_scr_peaks_emmeans)
```

```
## $emmeans
##  phase    response     SE   df lower.CL upper.CL
##  baseline    0.129 0.0603 29.6   0.0352    0.282
##  recovery    0.236 0.0816 29.6   0.0989    0.432
##  task1       0.762 0.1466 29.6   0.4922    1.091
##
## Results are averaged over the levels of: condition
## Degrees-of-freedom method: kenward-roger
## Confidence level used: 0.95
## Intervals are back-transformed from the sqrt scale
##
## $contrasts
##  contrast            estimate    SE df lower.CL upper.CL
##  baseline - recovery   -0.127 0.067 38   -0.290   0.0366
##  baseline - task1      -0.514 0.067 38   -0.677  -0.3505
##  recovery - task1      -0.387 0.067 38   -0.550  -0.2237
##
## Results are averaged over the levels of: condition
## Note: contrasts are still on the sqrt scale
## Degrees-of-freedom method: kenward-roger
## Confidence level used: 0.95
## Conf-level adjustment: tukey method for comparing a family of 3 estimates
```

```
# Model diagnostics

## Simulate residuals
model_scr_peaks_residuals <- simulateResiduals(fittedModel = model_scr_peaks,
    plot = F)

## Plot
plot(model_scr_peaks_residuals)
```




```
# Extract model output for publishing
tidy_lmer(model_scr_peaks)
```

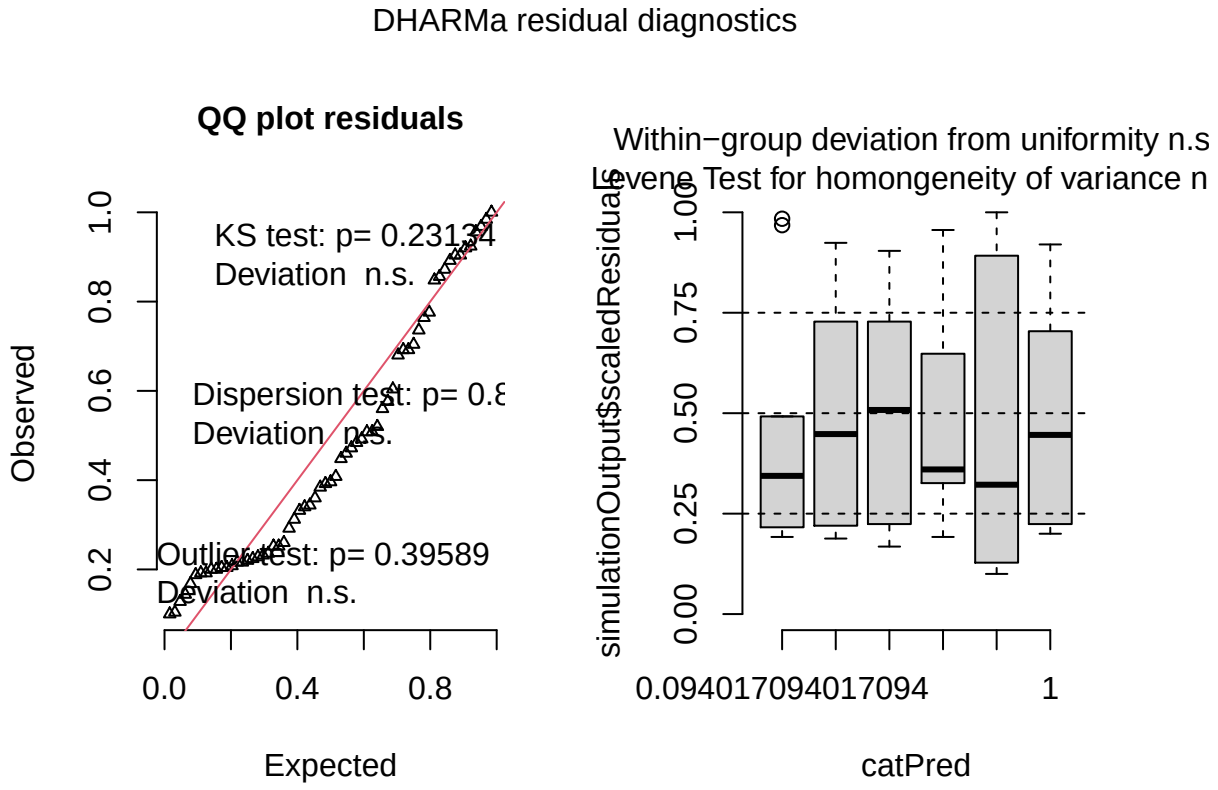
```
## # A tibble: 6 x 8
##   effect term                estimate std.error ci95         df statistic p.value
##   <chr>  <chr>                  <dbl>     <dbl> <chr>     <dbl>     <dbl> <chr>
## 1 fixed  (Intercept)             0.33      0.12 0.09 to ~  29.6      2.82 0.008
## 2 fixed  conditionwood           0.06      0.17 -0.28 to~  29.6      0.38 0.704
## 3 fixed  phaserecovery           0.21      0.09 0.03 to ~  38        2.32 0.026
## 4 fixed  phasetask1              0.59      0.09 0.4 to 0~  38        6.37 < 0.001
## 5 fixed  conditionwood:pha~     -0.18      0.13 -0.45 to~  38       -1.31 0.197
## 6 fixed  conditionwood:pha~     -0.15      0.13 -0.42 to~  38       -1.12 0.270
```

```
######

# Amplitude of skin conductance
# responses

## Fit the linear mixed model
model_scr_amplitude <- lmer(mean_scr_amplitude ~
    condition * phase + (1 | p_id), data = eda_full_time_filt_no12_smr)

## Model summary
summary(model_scr_amplitude)
```

```
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
```

```
## lmerModLmerTest]
## Formula: mean_scr_amplitude ~ condition * phase + (1 | p_id)
##    Data: eda_full_time_filt_no12_smr
##
## REML criterion at convergence: -3.7
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -1.8602 -0.4214 -0.1829  0.4011  2.5828
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  p_id     (Intercept) 0.02400  0.1549
##  Residual             0.02801  0.1674
## Number of obs: 63, groups:  p_id, 21
##
## Fixed effects:
##                              Estimate Std. Error        df t value Pr(>|t|)
## (Intercept)                  0.191533   0.068761 39.974089   2.785  0.00813 **
## conditionwood                0.087025   0.099644 39.974089   0.873  0.38768
## phaserecovery                0.002015   0.071361 38.000000   0.028  0.97762
## phasetask1                   0.070139   0.071361 38.000000   0.983  0.33188
## conditionwood:phaserecovery -0.093036   0.103411 38.000000  -0.900  0.37396
## conditionwood:phasetask1    -0.063910   0.103411 38.000000  -0.618  0.54025
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) cndtnw phsrcv phsts1 cndtn:
## conditionwd -0.690
## phaserecvry -0.519  0.358
## phasetask1  -0.519  0.358  0.500
## cndtnwd:phs  0.358 -0.519 -0.690 -0.345
## cndtnwd:ph1  0.358 -0.519 -0.345 -0.690  0.500
```

```r
# Post-hoc comparisons
model_scr_amplitude_emmeans <- emmeans(model_scr_amplitude,
    specs = pairwise ~ phase, type = "response",
    interaction = TRUE)

# Model diagnostics

## Simulate residuals
model_scr_amplitude_residuals <- simulateResiduals(fittedModel = model_scr_amplitude,
    plot = F)

## Plot
plot(model_scr_amplitude_residuals)
```




```
# Extract model output for publishing
tidy_lmer(model_scr_amplitude)
```

```
## # A tibble: 6 x 8
##   effect term                estimate std.error ci95         df statistic p.value
##   <chr>  <chr>                  <dbl>     <dbl> <chr>     <dbl>     <dbl> <chr>
## 1 fixed  (Intercept)             0.19      0.07 0.05 to ~  40.0      2.79 0.008
## 2 fixed  conditionwood           0.09      0.1  -0.11 to~  40.0      0.87 0.388
## 3 fixed  phaserecovery           0         0.07 -0.14 to~  38        0.03 0.978
## 4 fixed  phasetask1              0.07      0.07 -0.07 to~  38        0.98 0.332
## 5 fixed  conditionwood:pha~     -0.09      0.1  -0.3 to ~  38       -0.9  0.374
## 6 fixed  conditionwood:pha~     -0.06      0.1  -0.27 to~  38       -0.62 0.540
```

Cardiovascular activity

We repeat the process with cardiovascular activity, fitting two instances of the linear mixed model, with heart rate as dependent variable in one case and heart rate variability as the dependent variable in the other case.

```
# Heart rate

## Fit the linear mixed model
model_hr <- lmer(hr_bpm ~ condition * phase +
    (1 | p_id), data = physio_smr_filt)
```

```r
## Model summary
summary(model_hr)
```

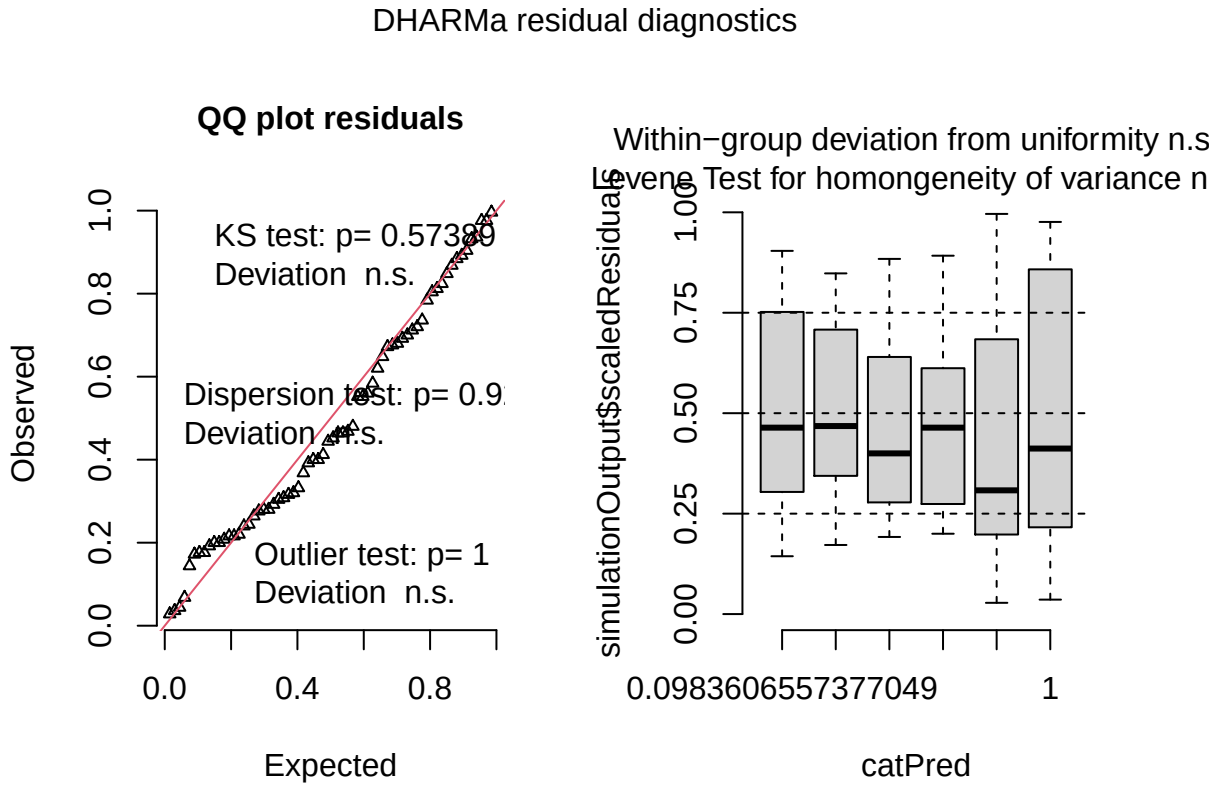
```
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: hr_bpm ~ condition * phase + (1 | p_id)
##    Data: physio_smr_filt
##
## REML criterion at convergence: 495.1
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -1.46579 -0.52058  0.04375  0.39494  2.88279
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  p_id     (Intercept) 174.33   13.203
##  Residual              94.36    9.714
## Number of obs: 66, groups:  p_id, 22
##
## Fixed effects:
##                             Estimate Std. Error      df t value
## (Intercept)                  71.5150     4.9422 32.5746  14.470
## conditionwood                 5.4510     6.9894 32.5746   0.780
## phasetask1                   16.4015     4.1420 40.0000   3.960
## phaserecovery                -0.7543     4.1420 40.0000  -0.182
## conditionwood:phasetask1      2.5512     5.8576 40.0000   0.436
## conditionwood:phaserecovery   0.4087     5.8576 40.0000   0.070
##                                        Pr(>|t|)
## (Intercept)                 0.000000000000000972 ***
## conditionwood                             0.4411
## phasetask1                                0.0003 ***
## phaserecovery                             0.8564
## conditionwood:phasetask1                  0.6655
## conditionwood:phaserecovery               0.9447
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) cndtnw phsts1 phsrcv cndt:1
## conditionwd -0.707
## phasetask1  -0.419  0.296
## phaserecvry -0.419  0.296  0.500
## cndtnwd:ph1  0.296 -0.419 -0.707 -0.354
## cndtnwd:phs  0.296 -0.419 -0.354 -0.707  0.500
```

```r
# Post-hoc comparisons
model_hr_emmeans <- emmeans(model_hr, specs = pairwise ~
    phase, type = "response")
confint(model_hr_emmeans)
```

```
## $emmeans
```

```
##  phase    emmean   SE   df lower.CL upper.CL
##  baseline   74.2 3.49 32.6     67.1     81.4
##  task1      91.9 3.49 32.6     84.8     99.0
##  recovery   73.7 3.49 32.6     66.6     80.8
##
## Results are averaged over the levels of: condition
## Degrees-of-freedom method: kenward-roger
## Confidence level used: 0.95
##
## $contrasts
##  contrast            estimate   SE df lower.CL upper.CL
##  baseline - task1      -17.68 2.93 40   -24.81   -10.55
##  baseline - recovery     0.55 2.93 40    -6.58     7.68
##  task1 - recovery       18.23 2.93 40    11.10    25.36
##
## Results are averaged over the levels of: condition
## Degrees-of-freedom method: kenward-roger
## Confidence level used: 0.95
## Conf-level adjustment: tukey method for comparing a family of 3 estimates
```

```r
# Model diagnostics

## Simulate residuals
model_hr_residuals <- simulateResiduals(fittedModel = model_hr,
    plot = F)

## Plot
plot(model_hr_residuals)
```




```
# Extract model output for publishing
tidy_lmer(model_hr)
```

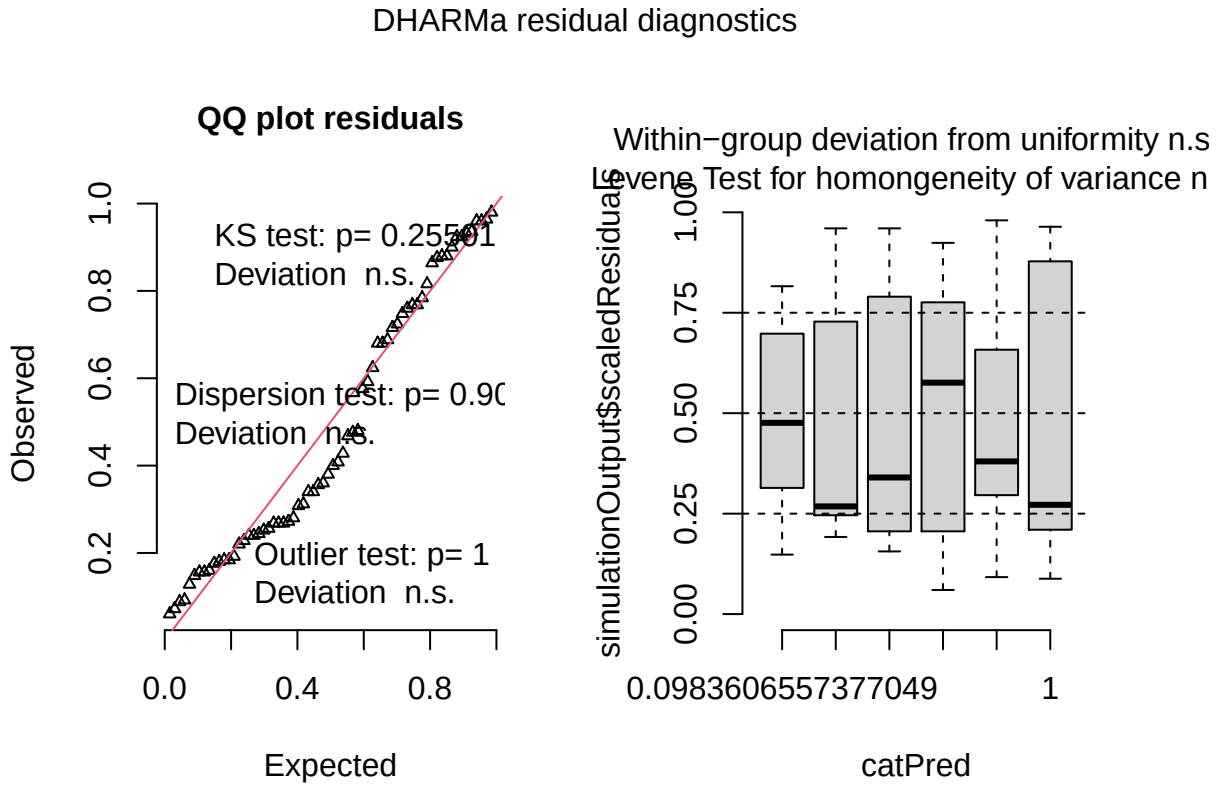
```
## # A tibble: 6 x 8
##   effect term              estimate std.error ci95          df statistic p.value
##   <chr>  <chr>                <dbl>     <dbl> <chr>      <dbl>     <dbl> <chr>
## 1 fixed  (Intercept)          71.5       4.94 61.45 to ~  32.6     14.5  < 0.001
## 2 fixed  conditionwood         5.45      6.99 -8.78 to ~  32.6      0.78 0.441
## 3 fixed  phasetask1           16.4       4.14 8.03 to 2~  40         3.96 < 0.001
## 4 fixed  phaserecovery        -0.75      4.14 -9.13 to ~  40        -0.18 0.856
## 5 fixed  conditionwood:ph~     2.55      5.86 -9.29 to ~  40         0.44 0.666
## 6 fixed  conditionwood:ph~     0.41      5.86 -11.43 to~  40         0.07 0.945
```

```
######

# Heart rate variability (RMSSD)

## Fit the linear mixed model
model_hrv <- lmer(rmssd ~ condition * phase +
    (1 | p_id), data = physio_smr_filt)

## Model summary
summary(model_hrv)
```

```
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
```

```
## lmerModLmerTest]
## Formula: rmssd ~ condition * phase + (1 | p_id)
##    Data: physio_smr_filt
##
## REML criterion at convergence: 516.8
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -1.6530 -0.4691 -0.1928  0.4688  2.3734
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  p_id     (Intercept) 233.0    15.26
##  Residual             139.4    11.81
## Number of obs: 66, groups:  p_id, 22
##
## Fixed effects:
##                              Estimate Std. Error        df t value     Pr(>|t|)
## (Intercept)                   43.47287    5.81842  33.65244   7.472 0.0000000121
## conditionwood                -12.23171    8.22849  33.65244  -1.487        0.146
## phasetask1                    -6.19915    5.03438  40.00000  -1.231        0.225
## phaserecovery                  1.89578    5.03438  40.00000   0.377        0.708
## conditionwood:phasetask1       2.98376    7.11969  40.00000   0.419        0.677
## conditionwood:phaserecovery   -0.07582    7.11969  40.00000  -0.011        0.992
##
## (Intercept)                 ***
## conditionwood
## phasetask1
## phaserecovery
## conditionwood:phasetask1
## conditionwood:phaserecovery
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) cndtnw phsts1 phsrcv cndt:1
## conditionwd -0.707
## phasetask1  -0.433  0.306
## phaserecvry -0.433  0.306  0.500
## cndtnwd:ph1  0.306 -0.433 -0.707 -0.354
## cndtnwd:phs  0.306 -0.433 -0.354 -0.707  0.500
```

```r
# Model diagnostics

## Simulate residuals
model_hrv_residuals <- simulateResiduals(fittedModel = model_hrv,
    plot = F)

## Plot
plot(model_hrv_residuals)
```




```
# Extract model output for publishing
tidy_lmer(model_hrv)
```

```
## # A tibble: 6 x 8
##   effect term                estimate std.error ci95          df statistic p.value
##   <chr>  <chr>                  <dbl>     <dbl> <chr>      <dbl>     <dbl> <chr>
## 1 fixed  (Intercept)            43.5       5.82 31.64 to ~  33.6      7.47 < 0.001
## 2 fixed  conditionwood         -12.2       8.23 -28.96 to~  33.6     -1.49 0.146
## 3 fixed  phasetask1             -6.2       5.03 -16.37 to~  40       -1.23 0.225
## 4 fixed  phaserecovery           1.9       5.03 -8.28 to ~  40        0.38 0.708
## 5 fixed  conditionwood:ph~       2.98      7.12 -11.41 to~  40        0.42 0.677
## 6 fixed  conditionwood:ph~      -0.08      7.12 -14.47 to~  40       -0.01 0.992
```

Environment

Information about the R environment where the analyses were conducted.

```
sessionInfo()
```

```
## R version 4.0.2 (2020-06-22)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 19043)
##
```

```
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=Greek_Greece.1253  LC_CTYPE=Greek_Greece.1253
## [3] LC_MONETARY=Greek_Greece.1253 LC_NUMERIC=C
## [5] LC_TIME=Greek_Greece.1253
## system code page: 65001
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] flextable_0.6.8   DHARMa_0.4.3      emmeans_1.6.2-1   broom.mixed_0.2.7
##  [5] lmerTest_3.1-3    lme4_1.1-27.1     Matrix_1.2-18     reticulate_1.20
##  [9] rstatix_0.7.0     janitor_2.1.0     lubridate_1.7.10  forcats_0.5.1
## [13] stringr_1.4.0     dplyr_1.0.7       purrr_0.3.4       readr_2.0.1
## [17] tidyr_1.1.3       tibble_3.1.3      ggplot2_3.3.5     tidyverse_1.3.1
##
## loaded via a namespace (and not attached):
##  [1] TH.data_1.0-10      minqa_1.2.4         colorspace_2.0-2
##  [4] ellipsis_0.3.2      rio_0.5.27          estimability_1.3
##  [7] snakecase_0.11.0    base64enc_0.1-3     fs_1.5.0
## [10] rstudioapi_0.13     farver_2.1.0        bit64_4.0.5
## [13] fansi_0.5.0         mvtnorm_1.1-2       xml2_1.3.2
## [16] codetools_0.2-16    splines_4.0.2       knitr_1.34
## [19] jsonlite_1.7.2      nloptr_1.2.2.2      pbkrtest_0.5.1
## [22] broom_0.7.9         dbplyr_2.1.1        png_0.1-7
## [25] compiler_4.0.2      httr_1.4.2          backports_1.2.1
## [28] assertthat_0.2.1    cli_3.0.1           formatR_1.11
## [31] htmltools_0.5.1.1   tools_4.0.2         coda_0.19-4
## [34] gtable_0.3.0        glue_1.4.2          rappdirs_0.3.3
## [37] Rcpp_1.0.7          carData_3.0-4       cellranger_1.1.0
## [40] vctrs_0.3.8         nlme_3.1-148        iterators_1.0.13
## [43] xfun_0.26           openxlsx_4.2.4      rvest_1.0.1
## [46] lifecycle_1.0.1     MASS_7.3-51.6       zoo_1.8-9
## [49] scales_1.1.1        vroom_1.5.4         hms_1.1.0
## [52] parallel_4.0.2      sandwich_3.0-1      RColorBrewer_1.1-2
## [55] yaml_2.2.1          curl_4.3.2          gdtools_0.2.3
## [58] stringi_1.7.3       highr_0.9           gap_1.2.3-1
## [61] foreach_1.5.1       boot_1.3-25         zip_2.2.0
## [64] rlang_0.4.10        pkgconfig_2.0.3     systemfonts_1.0.2
## [67] evaluate_0.14       lattice_0.20-41     labeling_0.4.2
## [70] bit_4.0.4           tidyselect_1.1.1    plyr_1.8.6
## [73] magrittr_2.0.1      R6_2.5.1            generics_0.1.0
## [76] multcomp_1.4-17     DBI_1.1.1           pillar_1.6.3
## [79] haven_2.4.3         foreign_0.8-81      withr_2.4.2
## [82] survival_3.1-12     abind_1.4-5         modelr_0.1.8
## [85] crayon_1.4.1        car_3.0-11          uuid_0.1-4
## [88] utf8_1.2.2          tzdb_0.1.2          rmarkdown_2.10
## [91] officer_0.4.0       grid_4.0.2          readxl_1.3.1
## [94] data.table_1.14.0   reprex_2.0.1        digest_0.6.27
## [97] xtable_1.8-4        numDeriv_2016.8-1.1 munsell_0.5.0
```