





neatStats: An R package for a neat pipeline from raw data to reportable statistics in psychological science

Gáspár Lukács^a  

^aDepartment of Cognition, Emotion, and Methods in Psychology, University of Vienna

Abstract ■ Performing the entire transition from raw data to reportable statistics can pose difficulties: it takes time, it allows various mistakes (that may or may not go unnoticed), and there are no general guidelines on how to proceed with this task. One particularly useful tool for this transition is the R programming language. However, how to use R for this is not trivial, especially for novices. The present paper serves as a step-by-step yet fast tutorial on how to make all the steps from raw data files to all the statistics normally needed in a conventional psychological experiment (including ANOVA and t-tests). At the same time, it also introduces the R package *neatStats*, which was created for the very purpose of making these steps as easy and straightforward as possible.

Keywords ■ data processing, raw data, analysis, statistics, reporting. **Tools** ■ R.

 gaspar.lukacs@univie.ac.at

 [10.20982/tqmp.17.1.p007](https://doi.org/10.20982/tqmp.17.1.p007)

Acting Editor ■
Cheng-Ta Yang
(Department of Psychology, National Cheng Kung University)

Reviewers
■ Two anonymous reviewers.

Introduction

As a rule, for each empirical experiment in psychological science, the collected raw data needs to be transformed into aggregated results (typically per examined subject or item), and the statistics to be reported need to be extracted from these aggregated results. Trivial as this may sound, to properly perform the entire necessary process for this is not always straightforward or easy. Indeed, it would make for an interesting study to assess the ratio of people who can go through all steps of transforming raw data, obtained from any given experiment, using their software of preference, into reportable statistics without making any mistake. In my own experience from overseeing the work of others (or vice versa), mistakes during this process happen more often than not, regardless of the career stage of the user. Even more worrying, as long as there is no clear error displayed by the given software and the obtained results seem sensible (or desirable), such mistakes can go unnoticed, and the mistaken results can get published. While the interpretation of statistics in themselves has received more and more attention and scrutiny in view of the so-called "replication crisis" in the recent years, the much blunter possibility of outright miscalculation is rarely mentioned – despite the baffling implications of related sporadic assessments (e.g., Brown, Kaiser, & Allison, 2018; Nui-

jten, Hartgerink, van Assen, Epskamp, & Wicherts, 2016).

Using programming languages instead of software operated primarily via graphical user interface is often suggested to facilitate data processing and statistical calculation (e.g. Rouder, Haaf, & Snyder, 2019). Once a code is written, it can be used and reused as many times as needed, literally with the press of a button. The automated processing prevents basic human mistakes that can occur during, for example, repetitive copy-pasting of large numbers of data points. Unlike in the case of manual data processing (e.g., in Microsoft Excel), incorrect data handling (missing data, wrong format, etc.) often result in explicit error or warning messages when the code is executed. The code can be written already for pilot test results, which can verify the integrity of both the obtained experimental data and the code itself. The same code can then be used for the completed (or ongoing) experiment. The parts of the analysis procedure can be easily double-checked step by step (typically: each line of the code), and corrections of mistakes (as well as any required modification) can be easily implemented at any given step, with the rest of the code intact. The code and the history of any changes can be conveniently recorded and reviewed via version control systems (e.g. *git*, GitHub). The output can be customized to a reportable format that can be easily copied into a manuscript (a convenience not to be under-



estimated when dealing with large numbers of statistics or with manuscript revisions). The code can be shared to be verified by colleagues, and even following publication by anyone interested, and often it can also be reused in future studies with relatively small modifications.

The R language (R Core Team, 2019) is particularly popular in data science and in academia. It is completely free, and there are abundant solutions for all sorts of specialized statistical tests with thousands of R packages available to expand the core language. However, R has a steep learning curve, which poses great difficulties to novices, and consequently a discouragement from choosing this otherwise excellent statistical tool. Also, while R does throw error or warning messages in many necessary cases, it will still allow various logical mistakes. One particular problem is the specific data arrangement required by many statistical R functions (let alone their different handling of given value types). For example, for each given set of within-subject factors in the test to be performed, the data usually needs to be transformed to one row per observation ("long format") – instead of always keeping a single row for all within-subject observations per subject (or item; "wide format") as it would be logical and readable, and as it is, consequently, implemented in most known statistical tools with user interface.

The present paper describes, via a relatively simple example, a pipelined procedure from raw data to reportable statistics including every output normally needed for scientific publication.¹ The single R package used for the entire code, *neatStats*, was created for the very purpose of making this pipeline as easy and straightforward as possible.² The resulting code is concise and readable, which lowers the probability of mistakes, while also making the task less burdensome – not only for the person who performs the original analysis, but also for those who try to understand and replicate it. This tutorial at the same time also serves as a fast introduction to statistics in R, to anyone with a minimal familiarity with R, or even just knowing another programming language. It should give a working knowledge of how to get all the relevant statistics out of raw data obtained in typical experimental designs, without leafing through thick books.

To note, I do not intend to address in detail the best practices related to specific statistics, but rather to show the basic steps of data analysis in a simple and focused

manner, with students in mind as primary readership. Having understood the general process, users may add any number of additional tests and modify it for different designs with just a little bit of creativity.

Material

The data used for the example is available via osf.io/49sq5/ (under "example_data"). The data was created artificially to simulate results from a hypothetical experiment. Each file represents the results of a participant who completed a response time task with positive and negative words (e.g., "happy", "peaceful" or "terror", "evil") displayed in green or in red, classifying each word with key presses according to valence (positive vs. negative). The main hypothesis is that positive words displayed in red and negative words displayed in green have slower responses (higher response times; RTs) as compared to positive words displayed in green and negative words displayed in red (Valence × Color interaction; e.g., Schietecat, Lakens, IJsselstein, & De Kort, 2018). Error rates (ERs) should follow a similar pattern (i.e., more incorrect responses are expected in cases where slower responses are expected). However, for about half of the participants, the green and red words were presented in separate blocks: all words shown first in green, and then again all words in red, or vice versa (all in red, then all in green). For these participants, color effects on valence should be absent (due to no polarity; Kawai, Lukács, & Ansorge, 2020; Proctor & Cho, 2006): no Valence × Color interaction expected in this group. Hence the three-way Valence × Color × Group interaction should be significant.

Each participant's data is given as a simple text file with .txt extension, whose file name contains the experiment title "expsim_color_valence", the given condition ("separate" or "mixed"), and the subject number (1-180), hence, for example, "expsim_color_valence_mixed_1.txt."

Each file contains the following data columns:

subject_num: Subject number.

condition: Contains "mixed" for the task with red and green colored words randomly mixed, and "separate" for the task with the two colors separated into two blocks.

rt: RTs, in ms, of the response to each stimulus. ("NA" for too slow responses.)

response: Contains "correct" for correct key pressed, "incorrect" for wrong key pressed, or "tooslow" for waiting too

¹The present tutorial describes one way of handling data, but of course this can be done in many different ways, and with the help of different R packages (or software). For example, one popular package that could replace many of the data transformation functions presented below is *tidyverse* (Wickham et al., 2019), which many find intuitive and helpful – however, some others (including the present author) find it rather cumbersome. In the end, everyone can follow and use their own preferred analysis steps, software packages, and general software (not excluding graphical interface-based ones that can also be excellent in their own way, see e.g. JASP Team, 2020). Regardless of personal preferences, at the very least the present tutorial demonstrates the general workflow of data processing, analysis, and reporting in R.

²The underlying statistical data is in many cases obtained via various other packages, which are given due credit in the documentation, detailed in the description of each given function (Calhoun, 2016; Kelley, 2019; Lawrence, 2016; Makowski, Ben-Shachar, & Lüdtke, 2019; Morey & Rouder, 2018; Robin et al., 2011; Wickham, 2016).



Figure 1 ■ Raw data example headers and rows.

subject_num	condition	rt	response	color	valence	age	gender
46	mixed	527.5664	correct	red	positive	21	2
46	mixed	815.0902	incorrect	red	positive	21	2
46	mixed	541.0515	correct	green	positive	21	2
46	mixed	NA	tooslow	red	positive	21	2
46	mixed	NA	tooslow	green	negative	21	2
46	mixed	503.9195	correct	green	positive	21	2

long with the response.

color: Color of each given presented stimulus ("green" or "red").

valence: Color of each given presented stimulus ("positive" or "negative").

age: Age of the participant.

gender: Gender of the participant (1, for male, or 2, for female).

The values for **subject_num**, **condition**, **age**, and **gender** are of course constant for each participant, i.e., have the same value in every row. Some additional information normally recorded – e.g., trial number, stimulus shown – are omitted here for simplicity since they are not relevant for the main statistics in any case.

The headers and six selected rows from file "exp-sim_color_valence_mixed_46.txt" are shown in Figure 1 as an example.

The script that is presented below step by step and described in detail, is also available with only brief in-line comments via osf.io/49sq5/ ("example_analysis.R"), which may also be used as a template for the analysis of any similar experiment. The full documentation of neatStats, as well as its source code, is available via github.com/gasparl/neatstats – and will be kept up-to-date for future versions. Via the same link, a number of hyperlinks are listed to related tutorials sources (e.g., short introductions to R for beginners).

Processing the Data

First install (if not yet installed) and load the library.

```
install.packages('neatStats')
library('neatStats')
```

Then set, as current working directory, the path to the folder that contains the data files. In RStudio (RStudio Team, 2015), the `path_neat()` function returns the path of the script file from which this function is executed. This

function takes a single argument, which will simply be appended to the path. For example, if all the data files are placed into a folder, named "example_data", next to the analysis script, then `path_neat('example_data')` will return the full path to that folder. Therefore, we can set the current working directory as follows.

```
setwd(path_neat('example_data'))
```

When using R via anything other than RStudio, the full path has to be given manually: for example, as³

```
setwd('C:/research/example_data')
```

Preliminary Inspection

First, we can read in and inspect all data together. Reading in all data at once takes time and therefore this is only for a first inspection, and should normally be omitted (along with the validation described below in this section) when rerunning modifications of the entire analysis.

Below, the `read_dir()` function merges the data of all files in the working directory with "txt" extension,⁴ and assigns it to `all_data`. (Since the data files contain headers, we have to specify `header = TRUE`, otherwise the returned data would contain the headers as the first row of data. It is advisable to set the `stringsAsFactors` and `fill` parameters to the given values as a precaution, although it should not matter in the present data; enter `?read_dir` and `?read.table` for details.)

```
all_data = read_dir(
  pattern = '\\.txt$',
  header = TRUE,
  stringsAsFactors = FALSE,
  fill = TRUE
)
```

First, to inspect the content and types of the data, we run

³To note, `path_neat()` is the only function in the `neatStats` package, as well as in the present paper, that requires RStudio.

⁴The pattern to detect txt extension is the regular expression `'\\.txt$'` (see `?base::regex`). Since all result file names start with "exp-sim_color_valence_", we could also include that in the pattern to make sure that no other text files that might be in this directory are collected: `'^expsim_color_valence_\\.txt$'`. Here, since no other text files are in the example folder, this is omitted for brevity.

**Listing 1** ■ Output of `str(all_data)`

```
'data.frame': 18000 obs. of 8 variables:
 $ subject\_num: int 1 1 1 1 1 1 1 1 1 1 ...
 $ condition : chr "mixed" "mixed" "mixed" "mixed" ...
 $ rt : num 133 509 NA 370.4 40.8 ...
 $ response : chr "correct" "correct" "tooslow" "correct" ...
 $ color : chr "green" "red" "green" "red" ...
 $ valence : chr "negative" "positive" "negative" "positive" ...
 $ age : int 23 23 23 23 23 23 23 23 23 ...
 $ gender : int 1 1 1 1 1 1 1 1 1 1 ...
```

Listing 2 ■ Output of `initial_peek_neat` on the raw data

```
green, negative:
  n   mean ci_low ci_upp  sd  median quantl_1st quantl_3rd fence_low fence_upp min max na
4435 586 581 591 165 583 476 697 -186 1359 7 1182 552
red, positive:
  n   mean ci_low ci_upp  sd  median quantl_1st quantl_3rd fence_low fence_upp min max na
4696 544 539 549 160 545 436 651 -209 1296 12 1112 488
green, positive:
  n   mean ci_low ci_upp  sd  median quantl_1st quantl_3rd fence_low fence_upp min max na
4322 527 522 532 158 526 417 634 -235 1287 4 1105 395
red, negative:
  n   mean ci_low ci_upp  sd  median quantl_1st quantl_3rd fence_low fence_upp min max na
4547 562 557 567 156 562 456 667 -175 1298 12 1071 573
```

`str(all_data)`. This returns the output shown in Listing 1.

Everything seems to be in order. We can now take a look at the data using the `peek_neat()` function. This function, by default, allows a general "peek" at any chosen variable (per group, if specified).⁵ Here, our variable of interest is the `rt` (response time) variable. We can already group this by the main factors **color** (red or green) and **valence** (positive or negative).

```
peek_neat(
  dat = all_data,
  values = 'rt',
  group_by = c('color', 'valence')
  round_to = 1
)
```

This returns, first, the output given in Listing 2 printed to the console.

One particular information we gain is that there are no far outliers: The `min` and `max` values are well within Tukey's far fences (`fence_low`, `fence_upp`; at 3 interquartile range [IQR] distance [see `?IQR`], which can be changed to the more conventional 1.5 IQR via the

`iqr_times` parameter). As always, see `?peek_neat` for details.

Second, the function also returns box plots per group, including overlaid violin plots (that indicate both density distribution and sample size via their width; Figure 2).

The box plots indicate a reasonable range, 0-1200 ms, for RTs (except perhaps for those below 150 ms, which can be excluded later on), the violin plots indicate roughly normal distribution for all data, and the outliers relatively close to the whiskers reconfirm that there are no far outliers. Both the violin plot widths and the `n` values show that trial numbers with valid (not NA) RT are roughly equal per condition. For a further check, the same function could also be executed grouping by response types (correct or incorrect responses) as `peek_neat(all_data, 'rt', 'response')` (Figure 3); or by all these factors (i.e., `c('color', 'valence', 'response')`). These will indicate that incorrect responses are relatively few, and that they are roughly equal in all conditions.

Data Preparation

We start the main analysis by extracting the necessary data from each file. To collect all file names in the directory, we

⁵Much more thorough and complex automatic data exploration can be performed via, for example, the DataExplorer (Cui, 2020) or the dataMaid (Petersen & Ekström, 2019) packages — as may be needed for more complex data or extensive exploratory analyses. The `psych` package also has various useful features: For example, the `psych::describe()` function provides skewness and kurtosis values. For data validation and cleaning in specific, the `validate` package may be useful.



Figure 2 ■ Box (and violin) plots, per valence and color, created using the peek_neat() function. The image was saved using RStudio's Exports option in the Plots panel.

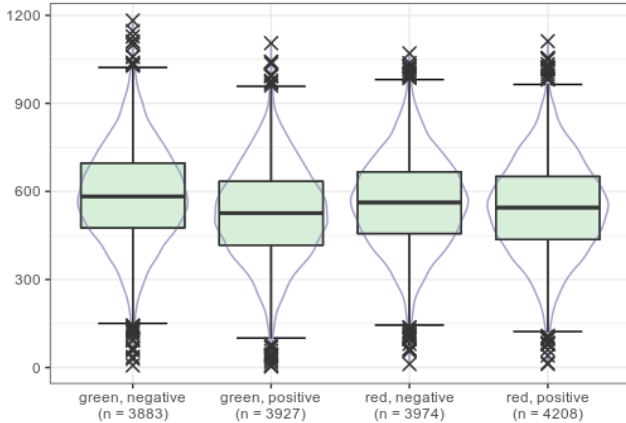
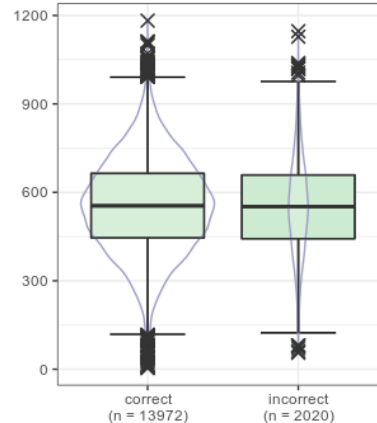


Figure 3 ■ Box (and violin) plots, per response type.



can use the list.files() function.

```
filenames=list.files(pattern='\\.txt$')
```

Now that we have the list of all the file names (in the filenames variable), we can loop through it, and, for each file name, read in the data from the corresponding file and extract the data that we need. The data from the participants will be merged together in one data frame, named subjects_merged, which will contain in each of its rows the extracted data of a single participant. Namely (Figure 4): condition, gender, age, overall ER, and mean (aggregated) RTs and ERs for each stimulus type.

The full code for this loop is given in Appendix 1. The detailed explanation follows here.

The enum() function prepends numbering to the file names (1 for first, 2 for second, etc.; this can be disabled via the enumerate parameter) merely for display, and, more importantly, it indicates a newly initiated loop for the rbind_loop function (see later).

The cat(file_name, fill = TRUE) line just prints the present file name to the console, to let us know which file is currently being processed. This is especially useful when the script is stopped due to an error: In that case, based on the last printed file name, we immediately know which file caused the error.

Then the read.table() function reads in the data of the current file (with the same arguments as for read_dir() before).

```
subject_data = read.table(
  file_name[2],
  header = TRUE,
  stringsAsFactors = FALSE,
  fill = TRUE )
```

Next, as a quick check to ensure the basic integrity of the data, I always verify that the data contains the expected number of rows (i.e., the number of trials in the given experiment). Otherwise the script is stopped and the text "unexpected trial number" is printed. (This could also be done per trial type, etc.)

```
if (nrow(subject_data) != 100) {
  stop('unexpected trial number: ',
       nrow(subject_data))
}
```

We can then get the mean RTs and ERs, per each stimulus type, using aggr_neat(). Here there are four stimulus types: (1) positive words in green, (2) positive words in red, (3) negative words in green, and (4) negative words in red. These four combinations can be obtained by grouping by the color and valence columns. For RTs, we need to get the mean of the values from the rt column (for each stimulus type). The method is mean by default, I write it out explicitly only for clarity. Without giving prefix, the default output names for item types would be, for example, green_negative, red_positive, etc. (automatically derived from the group_by arguments). To clarify that this is our RT measure, we can add 'rt' as a prefix, so that the item type names will be, for example, rt_green_negative, rt_red_positive. Finally, since we are only interested in the RTs of correct responses, we filter for response == 'correct', and, since RTs below 150 ms are probably just accidental (as such fast reactions are extremely unlikely), we also filter for rt > 150.

```
rts = aggr_neat(
  subject_data,
```

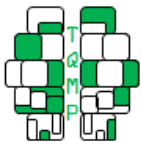


Figure 4 ■ Headers and six example rows (hence six different subjects) from the final aggregated data for statistical analysis.

subject_id	condition	gender	age	er_overall	rt_green_negative	rt_red_negative	rt_green_positive
88	mixed	1	24	0.10112360	583.7821	510.3457	473.4392
91	mixed	2	30	0.12643678	632.1917	568.1301	564.4493
92	mixed	2	23	0.17582418	577.0081	470.2068	501.9121
93	mixed	1	20	0.07692308	647.2663	598.7781	536.0131
100	separate	1	21	0.05555556	615.9090	590.8130	593.2610
102	separate	1	24	0.12359551	577.1626	550.8355	562.6747

rt_red_positive	er_green_negative	er_red_negative	er_green_positive	er_red_positive
531.0374	0.00000000	0.12121212	0.16666667	0.11111111
655.1975	0.14285714	0.13333333	0.05263158	0.16000000
544.8771	0.13043478	0.18181818	0.09090909	0.25000000
680.8611	0.05555556	0.09090909	0.07407407	0.08333333
613.7351	0.15789474	0.00000000	0.08000000	0.00000000
564.4707	0.08695652	0.14814815	0.14285714	0.12000000

```
rt,
group_by = c('color', 'valence'),
method = mean,
prefix = 'rt',
filt = (rt>150 & response=='correct')
)
```

The procedure is similar for ERs, except that the `aggr_neat()` has a special method for getting ratios of specific values (which are otherwise not straightforward with a single function): Whenever the argument for the method parameter is a string (i.e., text in quotation marks; character mode), the ratio of occurrences of the given string (in this case, the text 'incorrect') in the specified column (here: `response`) is returned (for each stimulus type). Since we usually do not want to include too slow responses in the calculation of ERs, we can filter for `response %in% c('correct', 'incorrect')`, and thereby get the ratio of the number of incorrect responses as compared to the number of correct and incorrect responses.⁶

```
ers = aggr_neat(
  subject_data,
  response,
  group_by = c('color', 'valence'),
  method = 'incorrect',
  prefix = 'er',
  filt = (response %in% c('correct', 'incorrect'))
)
```

)

The `aggr_neat()` function returns the value names (e.g., `rt_green_negative`) and values as columns (with column names **aggr_group** and **aggr_value**). The RT and ER values will eventually be transformed and merged into a single line together with the other subject information.

We would also like to get the overall ER (regardless of stimulus type), because we want to exclude participants with generally very high ER. (They may not have been paying attention or had undisclosed vision problems, etc.) For this too, we can use the `aggr_neat()` function, but in this case omitting the `group_by` argument, and appending, at the end, `$aggr_value`, in order to access the single value returned under this column.⁷

```
er_overall = aggr_neat(subject_data,
  response,
  method = 'incorrect',
  filt = (response %in% c('correct', 'incorrect'))
)$aggr_value
```

Finally, we also want the subject number, condition, age, and gender. These latter variables are constant in their respective columns, so we might as well take them from any row, for example the first row (e.g., for subject number: `subject_data$subject_num[1]`).

We can now use the `rbind_loop()` function that initiates a given data frame (here: `subjects_merged`) at the first cycle of the loop (internally detected via `enum()`)

⁶In this case the same could be achieved by a filter `response != 'tooslow'`, but perhaps that is less clear.

⁷To note, the same value can also be quite easily obtained without `aggr_neat()`, by writing, for example, `nrow(subject_data[subject_data$response == 'incorrect',]) / nrow(subject_data[subject_data$response %in% c('correct', 'incorrect'),])`. But, again, using `aggr_neat()` might be clearer.



and adds a new row in each cycle by "intelligently" merging all provided data and transforming them into a single row (see `?rbind_loop` for details).⁸

```
rbind_loop(  
  subjects_merged,  
  subject_id = subject_data$subject_num[1],  
  condition = subject_data$condition[1],  
  gender = subject_data$gender[1],  
  age = subject_data$age[1],  
  er_overall = er_overall,  
  rts,  
  ers  
)
```

When running the full for loop, the above described steps are repeated for each data file. After the loop has been run, the `subjects_merged` data frame is ready for statistical analysis. It might be worth noting that, while there is a `subject_id` column in this `subjects_merged` data frame, this is merely to keep track of records, but none of the statistical functions below require it: Each participant is represented by a single row in the data frame, hence no additional identifier is needed.

Debugging

An important advantage of looping through files⁹ is that it makes debugging and step-by-step double-checks extremely easy: As soon as an error occurs, the looping stops, and therefore we know that the file whose name was printed last caused it, and we can also immediately open/print any of the variables within the loop to find the precise cause. Relatedly, to inspect a single file, either to gain information on that specific file (e.g., due to an error) or to test or double-check the pre-processing steps, we can assign the single file name to the `file_name` variable (e.g., as `file_name = c(0, 'expsim_color_valence_mixed_1.txt')`), and then proceed to execute the subsequent lines one by one and check the corresponding results.

Similarly, by having a single subject's data within each loop, it is also easier to apply more complex data transformation and extraction and inspect the resulting data step-by-step (as we do not need to bother with grouping per subject for each step).

Statistics

At this point we might want to list column names, using `str(subjects_merged)`, for a quick overview of the

⁸If some of the data may be discrepant for some participants (e.g., some of the participants are tested with blue and yellow colors too), the matching columns are automatically paired (by name) and missing data is filled with NA values (Wickham, 2011).

⁹Instead of using, for example, `lapply` from base R or `group_by` from `tidyverse`.

¹⁰This assumes a priori (e.g., preregistered) exclusion criterion. Otherwise, to define exclusion criteria based on the observed data, one could for example look at the range and distribution of error rates using, for example, `peek_neat(subjects_merged, values = 'er_overall', f_plot = plot_neat)`.

¹¹The reasonable outlier exclusion criteria varies case by case and depends on the distribution of the given data, but, generally speaking, it is Tukey's far outliers (3 IQR distance from the IQR) that are considered extreme and may bias outcomes. Tukey's mere "regular" outliers (1.5 IQR distance from

content as well as for the convenient copy-pasting for subsequent use in statistical functions.

Exclusions and Inspection

Before any statistical tests, we exclude subjects with overall ER not smaller than 20% (i.e., for the analysis we only include subjects with an ER of less than 20%),¹⁰ using the `excl_neat()` function. (Whatever your research might be, you may have similar exclusion criteria, such as minimum test scores or attention check failures, that could be handled analogously at this point.)

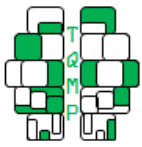
```
data_final = excl_neat(subjects_merged,  
  er_overall < 0.20,  
  group_by = 'condition')
```

This also automatically calculates and prints the number of exclusions and number of remaining participants per condition, showing three exclusions in the "mixed" condition, and two in the "separate" condition, leaving 87 and 89, respectively.

Then we can again use `peek_neat()` to get a summary and plots of the (now aggregated, per-subject) data, per condition (*mixed* or *separate*). Here I use a different plotting function argument (`plot_neat()`) to produce histograms, and I collapse all included RT variables into individual means (i.e., each subject will have one mean RT value included instead of four). The `collapse = mean` could simply be removed (or set to `NULL`) to print summaries and depict plots for each of the four RT variables per condition (here omitted for brevity).

```
peek_neat(  
  data_final,  
  values = c(  
    'rt_green_negative',  
    'rt_red_negative',  
    'rt_green_positive',  
    'rt_red_positive'  
  ),  
  group_by = 'condition',  
  collapse = mean,  
  f_plot = plot_neat  
)
```

The output is given in Listing 3. The plots (Figure 5) indicate that the observations (individual RT means) in each

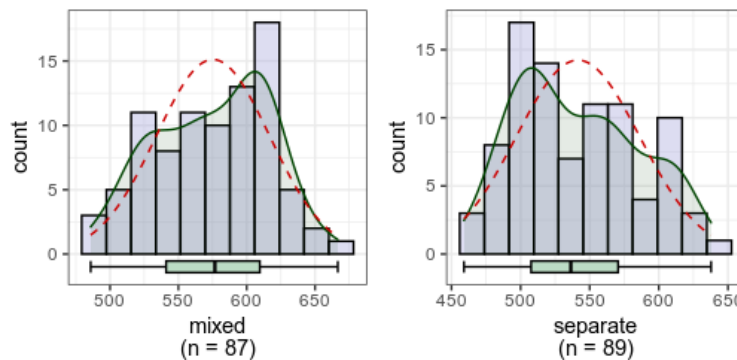


Listing 3 ■ Output peek_neat on the final data

```

mixed:
  n   mean  ci_low  ci_upper  sd   median  quantl_1st  quantl_3rd  fence_low  fence_upper  min  max  na
87  575    567    584     42    577    541        610        335       815        486 667  0
separate:
  n   mean  ci_low  ci_upper  sd   median  quantl_1st  quantl_3rd  fence_low  fence_upper  min  max  na
89  542    532    551     45    536    508        571        319       759        459 638  0
  
```

Figure 5 ■ Histograms, densities, horizontal box plots, per (between-subject) condition. (Red dashed line shows theoretical normally distributed density with the mean and standard deviation of the empirical data.)



group are within a reasonable range and no obvious outliers are present.¹¹

Demographics

Moving on to the first statistics to be reported, `dems_neat()` gives the average age and the number of males (or percentage, if so set), using (automatically) the age and gender columns.¹²

```
dems_neat(data_final, group_by = 'condition')
```

The output is:

```

Group <mixed>: 87 subjects (age=24±.23.8, 49 male)
Group <separate>: 89 subjects (age=24±.83.3, 45
  male)
  
```

Significance Testing

The main test is an analysis of variance (ANOVA) for the interaction Valence (positive vs. negative) × Color (green vs. red) × Group (separate vs. mixed). Since each participant may have several independent variables of interest

(in case of a within-subject design such as in this example), all variables (columns of numeric values) to be included in the test are given using their column names (as strings) in a string vector (or, in case of no within-subject factors, as a single string element), as argument for the values parameter. To determine which within-subject factors we want to contrast in the ANOVA (using the given values), there is a `within_ids` parameter that accepts a list as argument. In this list, the name of each element is the chosen display name for each factor; in this case "color" and "valence" (but we could use any other names as well). Each element must contain a vector of strings that are used to identify which of the independent variable names (given as values) belong to which factor. For example, the Color factor is given as `color = c('green', 'red')`. Using the given strings 'green' and 'red', the given value names 'rt_green_negative' and 'rt_green_positive' will be automatically identified as 'green' (since they contain the string 'green'), while the values 'rt_red_negative' and

the IQR), which are typically indicated as outliers in box plots (points above and below the whiskers), are usually acceptable. In any case, if we suspect that certain outliers may influence the outcomes (of the following statistical tests), we could compare the outcomes between when excluding them versus when keeping them. If we find any notable difference, this should be reported when writing up the results.

¹²The function automatically identifies the columns **age** and **gender** (or, alternatively, **sex**) in the data frame. Alternatively, appropriate column names may be provided via the `gender_col` and `age_col` optional parameters. Males and females are identified as "male" and "female" (or their abbreviations, e.g. "m" and "f"), or via the conventional number designations 1 (male) and 2 (female). There is no missing (NA) age or gender values in the example data – otherwise the missing numbers would be displayed as well.



Figure 6 ■ Factorial plot of means created using the `anova_neat()` function, with 95% CI in error bars (as per default).

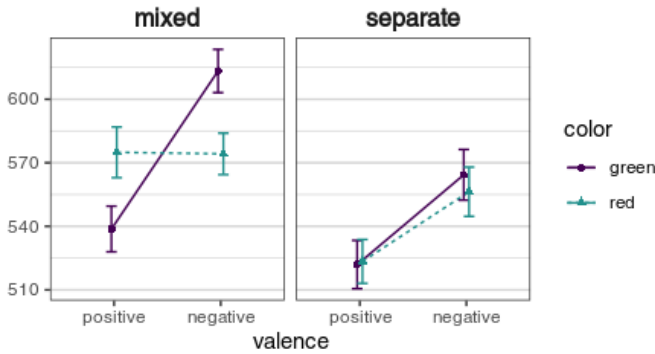
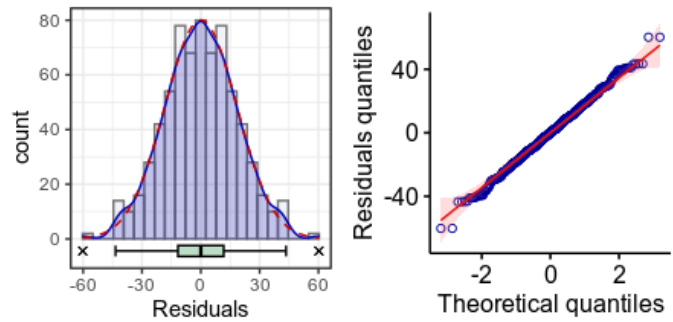


Figure 7 ■ Density (left panel) and Q-Q (right panel) plots for the ANOVA residuals. In both plots, the empirical data in blue is to be compared to the red lines depicting theoretical normally distributed values. (The light red area in the Q-Q plot indicates the 95% CI of the theoretical values).



'rt_red_positive' will be identified as 'red' (since they contain the string 'red').¹³ The between subject variables can simply be assigned to the `between_vars` parameter as a string vector, or, in case of only one between-subject factor (as in this example), as a single string element.

In addition, here I specify adding factorial plot (`plot_means`),¹⁴ normality tests (`norm_tests`), plots to visually assess normality (`norm_plots`), and variance descriptive statistics and tests (`var_tests`).

```
anova_neat (
  data_final,
  values = c(
    'rt_green_negative',
    'rt_green_positive',
    'rt_red_negative',
    'rt_red_positive'
  ),
  within_ids = list(
    color = c('green', 'red'),
    valence = c('positive', 'negative')
  ),
  between_vars = 'condition',
  plot_means = TRUE,
  norm_tests = 'all',
  norm_plots = TRUE,
  var_tests = TRUE
)
```

)

The factorial plot of means shows the expected pattern of outcomes (interaction in case of *mixed* condition, but not in case of the *separate* condition; Figure 6).¹⁵

The plots for the visual inspection of normality depict the pooled residuals in a density plot (with background histogram and boxplot at bottom; left panel in Figure 7), and a Q-Q plot ("quantile-quantile" plot; right panel in Figure 7). Both demonstrate very clear normal distribution, hence fulfilling the assumption of normality.¹⁶

The printed output consists of three parts. First, the "Normality of the Residuals", consisting of four normality tests (see `?neat_anova` for details). Since the results of these tests depend not only on the distribution but also on the sample size, visual inspection of the plots (see above) is preferable. Nonetheless, a significant normality test may serve as a reminder to double-check the plots. Here, none of the tests are statistically significant.

¹³Incorrect or ambiguous input returns an error message pointing out the given issue.

¹⁴This is implemented via the `plot_neat` function. Several features are customizable; see `?plot_neat` (to which arguments can also be passed via `anova_neat`). For example, to illustrate variation instead of certainty, we can display, with the error bars, the SDs of the means by adding the `eb_method = sd`. The main method could be replaced as well, for example, by setting `method = median`, to get medians instead of means, to control for outliers or skewness. (The corresponding error bars could be median absolute deviations; `eb_method = mad`.)

¹⁵When both factorial and normality plots are created, the former will be displayed first, and the latter will be displayed immediately after it. To move back to the first in RStudio, click on the "Previous plot" (left arrow) button in the plot panel.

¹⁶If the residuals do not indicate normal distributions, it does not in itself invalidate the ANOVA results: Several hyperlinks are provided at <https://github.com/gasparl/neatstats> for more details. You may also improve the model, for example, by removing outliers or trimming the data. Automatic trimming as well as robust ANOVA using medians can be performed via the `WRS2` package (Mair & Wilcox, 2020). Alternatively, linear modelling provides more flexible options (Bates, M'achler, Bolker, & Walker, 2015).



Listing 4 ■ anova_neat equality of variance

```
rt_green_negative:
  mixed: n = 87, SD = 48.62; separate: n = 89, SD = 57.54.
  Brown-Forsythe: F(1,174) = 1.77, p = .186; -FlignerKilleen: X2(1) = 1.953, p = .162.
rt_green_positive:
  mixed: n = 87, SD = 51.31; separate: n = 89, SD = 54.74.
  Brown-Forsythe: F(1,174) = 0.09, p = .766; -FlignerKilleen: X2(1) = 0.047, p = .828.
rt_red_negative:
  mixed: n = 87, SD = 46.57; separate: n = 89, SD = 55.96.
  Brown-Forsythe: F(1,174) = 2.20, p = .140; -FlignerKilleen: X2(1) = 2.220, p = .136.
rt_red_positive:
  mixed: n = 87, SD = 56.86; separate: n = 89, SD = 49.81.
  Brown-Forsythe: F(1,174) = 1.56, p = .214; -FlignerKilleen: X2(1) = 1.947, p = .163.
```

Listing 5 ■ anova_neat three-way results

```
F(1,174) = 29507.56, p < .001, ηp2 = .994, 90% CI [.993, .995], ηG2 = .991. ((Intercept))
F(1,174) = 27.03, p < .001, ηp2 = .134, 90% CI [.065, .213], ηG2 = .094. (condition)
F(1,174) = 0.78, p = .379, ηp2 = .004, 90% CI [0, .035], ηG2 = .001. (color)
F(1,174) = 223.33, p < .001, ηp2 = .562, 90% CI [.483, .622], ηG2 = .112. (valence)
F(1,174) = 0.11, p = .736, ηp2 = .001, 90% CI [0, .019], ηG2 < .001. (color × condition)
F(1,174) = 0.02, p = .888, ηp2 < .001, 90% CI [0, .009], ηG2 < .001. (condition × valence)
F(1,174) = 56.89, p < .001, ηp2 = .246, 90% CI [.159, .330], ηG2 = .038. (color × valence)
F(1,174) = 34.92, p < .001, ηp2 = .167, 90% CI [.090, .248], ηG2 = .024. (color × condition × valence)
```

```
Shapiro-Wilk test: W = 1.00, p = .702
D'Agostino-Pearson test: K2 = 0.23, p = .891
Anderson-Darling test: A2 = 0.18, p = .917
Jarque-Bera test: JB = 0.12, p = .940
```

Second, "Equality of Variances" (see Listing 4) that show sample sizes (*n*) and *SDs* per group, for each within-subject level combination. We can see that (a) the sample sizes per group are close to equal, and (b) the variances (*SDs*) are of similar magnitude in the two groups within each within-subject level combination; hence the outcomes will not be substantially biased.¹⁷ Two unequal variances significance tests are also shown for completeness (for each combination), although these are generally not very reliable – in any case, none of them is statistically significant.

Finally, the *F* statistics and related effect sizes are shown in Listing 5.

Without going into details, the three-way interaction is significant as expected. To note, the statistics are as close as possible to reportable format, but italics, subscripts, and superscripts are not well supported as console outputs – hence these have to be adjusted when preparing a manuscript (e.g., η_p^2 and η_G^2).

The ANOVA could be repeated for ERs by simply replacing the starting characters *rt_* with *er_* in the four vari-

able names for the values parameter. Similarly, all the tests below would be the same for ERs – i.e., only the variable input would need to be changed – but these are omitted here for brevity.

We follow up with two separate ANOVAs. (Normality testing is omitted here for brevity: Both residuals are normally distributed. Equal variances are not applicable since the only between-subject factor is dropped.) The first, below, is to show the absence of Color × Valence interaction in the "separate" condition.

```
anova_neat (
  data_final[data_final$condition == 'separate'],
  values = c(
    'rt_green_negative',
    'rt_green_positive',
    'rt_red_negative',
    'rt_red_positive'
  ),
  within_ids = list(
    color = c('green', 'red'),
    valence = c('positive', 'negative')
  ),
  bf_added = TRUE
)
```

Here Bayes Factors (BFs) are added as a complementary test to provide potential evidence for equivalence.¹⁸ While the rest of the numbers will always be identical for

¹⁷There is no established rule for an exact decision on whether the equal variances assumption is violated. Nonetheless, in case of a suspected violation, if the ANOVA contains only between-subject factors, one can simply use "hc3" correction (see the `white.adjust` parameter in `?anova_neat`). Alternatively, in case of a single between-subject factor ("one-way ANOVA"), Welch-correction can be used too (and is in fact applied by default, in `anova_neat`). In case of a mixed ANOVA, robust methods or linear modelling may be used (see the previous Footnote).

¹⁸Inclusion BFs based on matched models, calculated via `bayestestR` (Makowski et al., 2019).



Listing 6 ■ `anova_neat` two-way results ("separate" condition)

```
F(1,88) = 13114.19, p < .001,  $\eta^2$  = .993, 90% CI [.991, .995],  $\eta^2$  = .990. ((Intercept))
F(1,88) = 0.72, p = .398,  $\eta^2$  = .008, 90% CI [0, .064],  $\eta^2$  = .001, BF01 = 6.25. (color)
F(1,88) = 117.52, p < .001,  $\eta^2$  = .572, 90% CI [.456, .650],  $\eta^2$  = .107, BF10 = 2.15 × 1016. (valence)
F(1,88) = 1.27, p = .262,  $\eta^2$  = .014, 90% CI [0, .079],  $\eta^2$  = .002, BF01 = 3.14. (color × valence)
```

Listing 7 ■ `anova_neat` two-way results ("mixed" condition)

```
F(1,86) = 16701.63, p < .001,  $\eta^2$  = .995, 90% CI [.993, .996],  $\eta^2$  = .992. ((Intercept))
F(1,86) = 0.15, p = .699,  $\eta^2$  = .002, 90% CI [0, .041],  $\eta^2$  < .001, BF01 = 8.11. (color)
F(1,86) = 106.02, p < .001,  $\eta^2$  = .552, 90% CI [.432, .635],  $\eta^2$  = .117, BF10 = 4.15 × 1012. (valence)
F(1,86) = 106.33, p < .001,  $\eta^2$  = .553, 90% CI [.433, .635],  $\eta^2$  = .121, BF10 = 7.01 × 1017. (color × valence)
```

the same data, the BFs can vary slightly (typically only in fractional digits) due to its inherent random sampling process. My specific output is given in Listing 6.

As expected, no significant interaction. The *BF* for the interaction is also just large enough to be labeled as substantial evidence for equivalence.¹⁹ *BFs* supporting equivalence (i.e., below 1) are always displayed as inverse, hence all *BFs* displayed are above 1, and support for equivalence is indicated by the numbers 01, such as in BF01 (as opposed to BF10, for *BF* supporting difference). When assigning the `anova_neat()` function (e.g., `my_results = anova_neat(...)`), it will return a list that contains, among other things, the exact values of the statistics for each effect, including unconverted *BFs*.

Now, the second follow-up ANOVA, to show the presence of significant Color × Valence interaction in the "mixed" condition. My output is given in Listing 7:

```
anova_neat(
  data_final[data_final$condition == '
    mixed', ],
  values = c(
    'rt_green_negative',
    'rt_green_positive',
    'rt_red_negative',
    'rt_red_positive'
  ),
  within_ids = list(
    color = c('green', 'red'),
    valence = c('positive', 'negative')
```

```
),
  bf_added = TRUE
)
```

The interaction is significant as expected. To explore this interaction in the mixed condition, we could perform various t-tests,²⁰ but perhaps what is interesting here is to check whether there is a significant difference between red and green in case of either positive or negative words. First, for convenience, we create a new data frame with only mixed condition.

```
subjects_mx = excl_neat(data_final, condition == '
  mixed')
```

Now the paired t-test²¹ for red versus green for positive words, with normality tests and plots for normality inspection.

```
t_neat(
  subjects_mx$rt_green_negative,
  subjects_mx$rt_red_negative,
  pair = TRUE,
  norm_tests = 'all',
  norm_plots = TRUE,
  bf_added = TRUE)
```

Q-Q plots (Figure 8) and density plots (Figure 9) are created for both variables as well as for the pairwise difference, and a scatter plot for the relation of the two variables. The normality assumption concerns the difference only (and hence the normality significance tests are con-

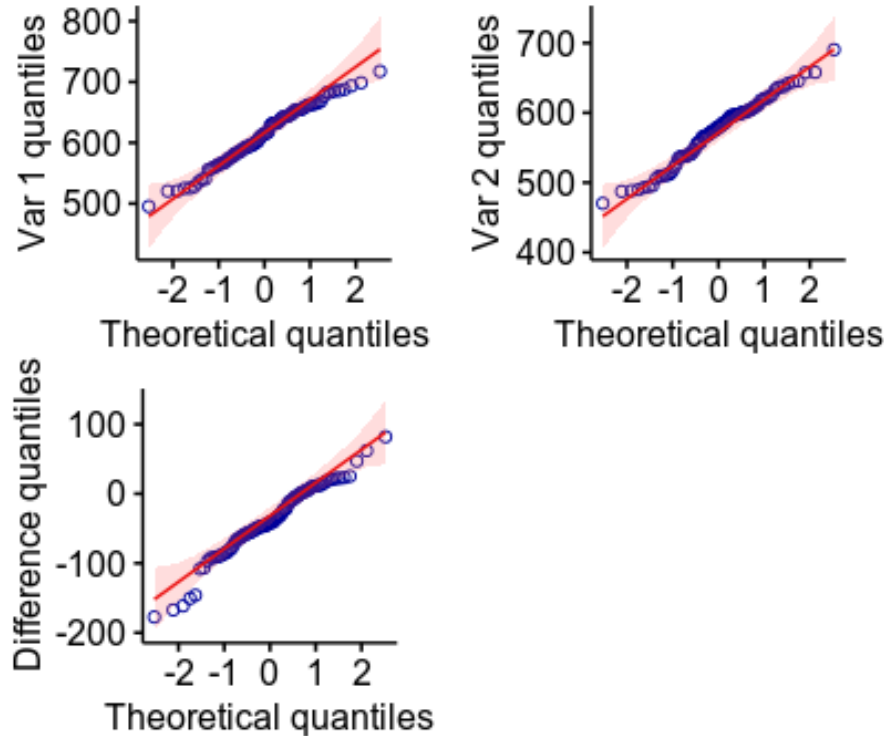
¹⁹Anyone unconvinced may use the "data_generation_code.R" file from osf.io/49sq5/ to "take more participants" and rerun the test with the increased sample size.

²⁰Typically, a statistically significant interaction of two factors should be followed up by testing simple effects (i.e., testing the effect of one factor at each level of the other factor), while a nonsignificant interaction should be followed up by examining only the main effect of each of the two factors (with the effect of one factor being averaged across the levels of the other factor). For a lucid paper on the details and on further possibilities, see Wei, Carroll, Harden, and Wu (2012). For multiple testing issues and solutions in ANOVAs, see Cramer et al. (2016); for more in general, see e.g. Bender and Lange (2001); note that conventional *p* value corrections can be easily implemented in base R (see `?p.adjust`). As a brief general recommendation: (a) ideally, the choice on decomposition and follow-up tests (and their corrections) should be determined (and preregistered) prior to any analysis (and data collection), and (b) complex designs (e.g., more than three factors and/or several levels per factor) are not only very difficult to interpret but also strongly diminish statistical power and therefore should be avoided.

²¹Generally complies with (and several features are based on) the recommendations of Lakens (2015).



Figure 8 ■ Q-Q plots for paired t-test. *Var1* is the first variable given in the function, *Var2* is the second. The difference values are calculated as *Var2* minus *Var1* (per observation).



ducted only for the difference values).

The plots seem to indicate roughly normally distributed values, although not very close to the theoretical values. The normality tests are also short of but near significance (at an alpha of .05).

Shapiro-Wilk test: $W = 0.97$, $p = .059$
D'Agostino-Pearson test: $K2 = 4.38$, $p = .112$
Anderson-Darling test: $A2 = 0.68$, $p = .071$
Jarque-Bera test: $JB = 3.63$, $p = .163$

The t-test statistics are:

Pearson correlation: $r(85) = .454$, 95% CI [.269, .606], $p < .001$.
Descriptives: $\pm MSD = 613 \pm .3148.62$ vs. $574 \pm .2046.57$
(raw mean difference: 39.11, 95% CI [28.50, 49.72])
 $t(86) = 7.33$, $p < .001$, $d = 0.79$, 95% CI [0.54, 1.02], $BF_{10} = 8.06 \times 10^7$.

Since the evidence for the difference is very strong ($p < .001$, and also $BF_{10} = 2.45 \times 10^7$), it is unlikely that the significant difference is only due to (minor) violations of the normality assumption. Nonetheless, the uncertainty regarding normality can be addressed

using Wilcoxon signed-rank test (including for *BFs*; van Doorn, Ly, Marsman, & Wagenmakers, 2020), by adding `nonparametric = TRUE` to the function.

```
t_neat (
  subjects_mx$rt_green_negative,
  subjects_mx$rt_red_negative,
  pair = TRUE,
  nonparametric = TRUE,
  bf_added = TRUE
)
```

The frequentist inference again strongly supports the difference between the two variables, though the *BF* is indeterminate. (A one-sided test expecting the first variable to be larger – as it would be reasonable in this specific experiment – could be specified as `greater = '1'`, and would provide substantial evidence for the difference in this direction with *BF* too.)

Spearman's rank correlation: $r_s = .456$, 95% CI [.272, .608], $p < .001$.
Descriptives: $M \pm SD = 613.31 \pm 48.62$ vs. 574.20 ± 46.57 (raw mean difference: 39.11, 95% CI [26.51, 47.70])

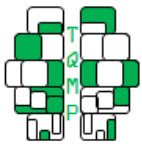
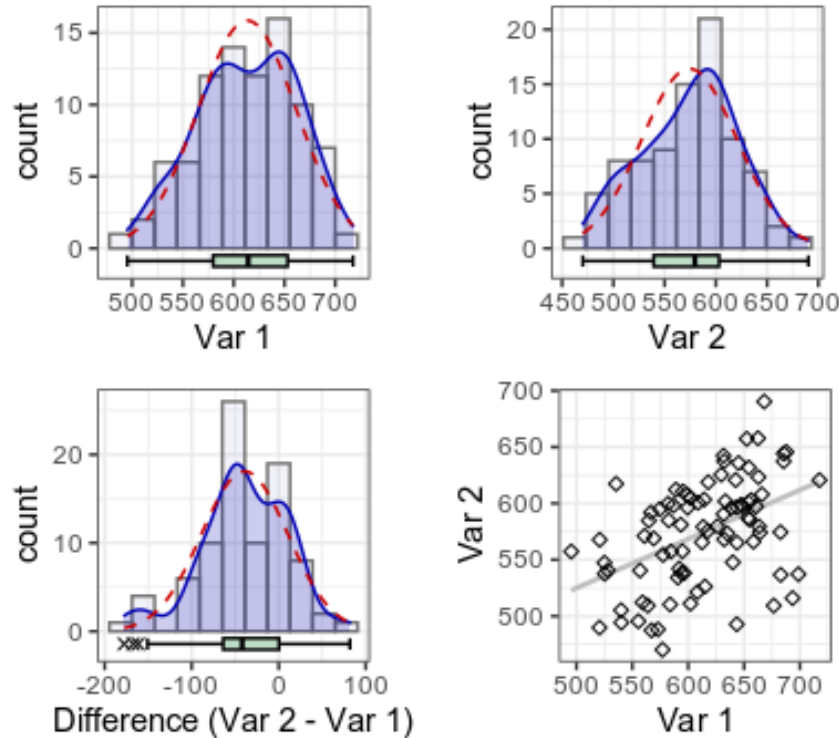


Figure 9 ■ Density plots (with background histogram and box plot at the bottom) and scatter plot for paired t-test. Var1 is the first variable given in the function, Var2 is the second.



W = 3327.00, p < .001, d = 0.79, 95% CI [0.54, 1.02], BF10 = 2.23.

The same t-tests could also be run for negative words by just replacing _positive with _negative for both variables.

Tables

All left to do is create a table to show means and SDs as customary, as per Appendix 2.

This will produce a table with the headers and data seen in Table 1. This table was actually obtained by setting the additional parameter to_clipboard = TRUE, which copies the table to the system clipboard with plain format. This can be copied into e.g. Microsoft Excel, and from that to Microsoft Word, which creates the table. (Unfortunately, Microsoft Word does not produce a table when directly pasting tab-separated plain text. The situation is similar in WPS, LibreOffice, etc.)

Conclusion

In the described example, we could conveniently gather all statistics needed for reporting from this (hypothetical) experiment. Arguably the process is fairly easy to adjust for similar usage in at least a large portion of quantitative psychology experiments. Much of the logic presented here is applicable even if the design is very different. For example, the data might need to be aggregated per presented item instead of per subject: In that case, all data may be read in at once (see read_dir) and then aggregated per item – and even so, other relevant information from files per subject can be extracted as presented above. An example script for such a specific case, with brief inline comments, is available via osf.io/49sq5/ ("example_2").

While the neatStats package aims to provide all necessary basic statistics and includes a number of convenience functions (Table 2), it of course does not by any means preclude or discourage the use of any other packages – in particular those needed for more advanced or

²²Other R packages like apaTables and papaja also offer publication-ready styled tables, but require different prior data transformation etc.

²³The table layout here is of course formatted for publication, but the names and numbers are verbatim.

²⁴See also the write_clip function of the clipr package for smartly copying all kinds of objects to the system clipboard.

**Table 1** ■ Means and SDs for individual RT means

aggr_group	rt_green_negative	rt_green_positive	rt_red_negative	rt_red_positive
mixed	613±49	539±51	574±47	575±57
separate	564±58	522±55	556±56	523±50

Note. Note. Means and SDs (in the format of M±SD) for individual RT means per stimulus type.

more unique statistical tests and computations.

Authors' note

Gáspár Lukács is a recipient of a DOC Fellowship of the Austrian Academy of Sciences at the Department of Cognition, Emotion, and Methods in Psychology at the University of Vienna. Thanks to Claudia "Kiki" Kawai and Eva Specker for their very helpful reviews of the first draft of the manuscript. Earlier versions of this tutorial were used as supplementary course material at the University of Vienna. The `neatStats` R package originates in (and was inspired by) a Cohen's d "helper" function provided by Bennett Kleinberg (https://github.com/benaaron188/r_helper_functions).

References

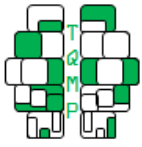
- Altman, D. G., & Bland, J. M. (2011). How to obtain the confidence interval from a p value. *BMJ*, *343*, d2090–d2090. doi:10.1136/bmj.d2090
- Bates, D., M'achler, M., Bolker, B., & Walker, S. (2015). Fitting linear mixed-effects models using `lme4`. *Journal of Statistical Software*, *67*(1), 11–11. doi:10.18637/jss.v067.i01
- Bender, R., & Lange, S. (2001). Adjusting for multiple testing—when and how? *Journal of Clinical Epidemiology*, *54*(4), 343–349. doi:10.1016/S0895-4356(00)00314-0
- Brown, A. W., Kaiser, K. A., & Allison, D. B. (2018). Issues with data and analyses: Errors, underlying themes, and potential solutions. *Proceedings of the National Academy of Sciences*, *115*(11), 2563–2570. doi:10.1073/pnas.1708279115
- Calhoun, P. (2016). `Exact`: Unconditional exact test [r package] (Version 1.7). Retrieved from <https://CRAN.R-project.org/package=Exact>
- Cramer, A. O. J., van Ravenzwaaij, D., Matzke, D., Steingrover, H., Wetzels, R., Grasman, R. P. P. P., ... Wagenmakers, E.-J. (2016). Hidden multiplicity in exploratory multiway anova: Prevalence and remedies. *Psychonomic Bulletin & Review*, *23*(2), 640–647. doi:10.3758/s13423-015-0913-5
- Cui, B. (2020). `Dataexplorer`: Automate data exploration and treatment (Version 8.0.1). Retrieved from <https://CRAN.R-project.org/package=DataExplorer>
- JASP Team. (2020). `Jasp` (Version 0.12.2). Retrieved from <https://jasp-stats.org/>
- Kawai, C., Lukács, G., & Ansorge, U. (2020). Polarities influence implicit associations between colour and emotion. *Acta Psychologica*, *209*, 103–143. doi:10.1016/j.actpsy.2020.103143
- Kelley, K. (2019). `Mbess`: The `mbess` r package [r package]. Version 4.5.1. Retrieved from <https://CRAN.R-project.org/package=MBESS>
- Lakens, D. (2015). The perfect t-test [r package] (Version 1.0.0). Retrieved from <https://github.com/Lakens/perfect-t-test>
- Lawrence, M. A. (2016). `Ez`: Easy analysis and visualization of factorial experiments [r package] (Version 4.4-0). Retrieved from <https://CRAN.R-project.org/package=ez>
- Mair, P., & Wilcox, R. (2020). Robust statistical methods in `r` using the `wrs2` package. *Behavior Research Methods*, *52*(2), 464–488. doi:10.3758/s13428-019-01246-w
- Makowski, D., Ben-Shachar, M., & Lüdtke, D. (2019). `Bayestestr`: Describing effects and their uncertainty, existence and significance within the Bayesian framework. *Journal of Open Source Software*, *4*(40), 1541–1545. doi:10.21105/joss.01541
- Morey, R. D., & Rouder, J. N. (2018). `Bayesfactor`: Computation of Bayes factors for common designs [r package] (Version 0.9.2). Retrieved from <https://CRAN.R-project.org/package=BayesFactor>
- Nuijten, M. B., Hartgerink, C. H. J., van Assen, M. A. L. M., Epskamp, S., & Wicherts, J. M. (2016). The prevalence of statistical reporting errors in psychology (1985–2013). *Behavior Research Methods*, *48*(4), 1205–1226. doi:10.3758/s13428-015-0664-2
- Petersen, A. H., & Ekstrøm, C. T. (2019). `Datamaid`: Your assistant for documenting supervised data quality screening in `r`. *Journal of Statistical Software*, *90*(6), 1–38. doi:10.18637/jss.v090.i06
- Proctor, R. W., & Cho, Y. S. (2006). Polarity correspondence: A general principle for performance of speeded binary classification tasks. *Psychological Bulletin*, *132*(3), 416–442. doi:10.1037/0033-2909.132.3.416
- R Core Team. (2019). *R: A language and environment for statistical computing*. Vienna, Austria: R Foundation for Statistical Computing. Retrieved from <https://www.R-project.org/>

**Table 2** ■ Overview of selected `neatStats` functions

Function	Description
<code>aggr_neat</code>	Returns aggregated values per group for a given variable.
<code>anova_neat</code>	ANOVA F-test results with appropriate Welch's and epsilon corrections where applicable (unless specified otherwise), including partial eta squared effect sizes with CIs, generalized eta squared, and inclusion Bayes factor based on matched models (<i>BFs</i>)
<code>ci_from_p</code>	Calculates approximate CI for any given difference, based on the difference value and the <i>p</i> value (Altman & Bland, 2011).
<code>corr_neat</code>	Parametric and nonparametric correlation results including CI and correlation <i>BF</i> .
<code>dems_neat</code>	Prints participant count, age mean and SD, and gender ratio (numbers or percentages) per group, from given data frame.
<code>excl_neat</code>	Filters data frame by rows and prints the numbers of excluded rows and remaining rows.
<code>peek_neat</code>	Prints and returns cursory summaries and creates plots per group.
<code>plot_neat</code>	Creates line or bar plots for factorial designs, or otherwise descriptive dispersion plots (histogram, density, box plots) for a continuous variable.
<code>props_neat</code>	Unconditional exact test results for the comparison of two independent proportions, including CI, Cohen's <i>h</i> (and its CI), and corresponding independent multinomial contingency table <i>BF</i> .
<code>read_dir</code>	Reads data files from any given directory and merges their content into a single data frame.
<code>roc_neat</code>	DeLong's comparison of two areas under the receiver operating characteristic curves, including CI.
<code>table_neat</code>	Creates a descriptive table, using <code>aggr_neat</code> functions as arguments.
<code>t_neat</code>	Welch's t-test results including CI, Cohen's <i>d</i> (and its CI), and <i>BF</i> . Wilcoxon equivalents for non-parametric test.

Note. Note. The table presents some of the functions available in `neatStats` (v1.5.1). See the official documentation for all functions and all details.

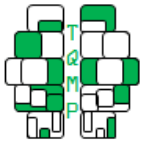
- Robin, X., Turck, N., Hainard, A., Tiberti, N., Lisacek, F., Sanchez, J.-C., & Müller, M. (2011). Proc: An open-source package for R and S+ to analyze and compare roc curves. *BMC Bioinformatics*, 12(1), 77–83. doi:10.1186/1471-2105-12-77
- Rouder, J. N., Haaf, J. M., & Snyder, H. K. (2019). Minimizing mistakes in psychological science. *Advances in Methods and Practices in Psychological Science*, 2(1), 3–11. doi:10.1177/2515245918801915
- RStudio Team. (2015). *Rstudio: Integrated development for R*. Inc: RStudio. Retrieved from <http://www.rstudio.com/>
- Schietecat, A. C., Lakens, D., IJsselstein, W. A., & De Kort, Y. A. W. (2018). Predicting context-dependent cross-modal associations with dimension-specific polarity attributions. *Part*, 4(1), 21–41. doi:10.1525/collabra.126
- van Doorn, J., Ly, A., Marsman, M., & Wagenmakers, E.-J. (2020). Bayesian rank-based hypothesis testing for the rank sum test, the signed rank test, and Spearman's rho. *Journal of Applied Statistics*, 45, 1–23. doi:10.1080/02664763.2019.1709053
- Wei, J., Carroll, R. J., Harden, K. K., & Wu, G. (2012). Comparisons of treatment means when factors do not interact in two-factorial studies. *Amino Acids*, 42(5), 2031–2035. doi:10.1007/s00726-011-0924-0
- Wickham, H. (2011). The split-apply-combine strategy for data analysis. *Journal of Statistical Software*, 40(1), 44–99. doi:10.18637/jss.v040.i01
- Wickham, H. (2016). *Ggplot2: Elegant graphics for data analysis (second edition)*. Berlin: Springer.
- Wickham, H., Averick, M., Bryan, J., Chang, W., McGowan, L., François, R., ... Yutani, H. (2019). Welcome to the tidyverse. *Journal of Open Source Software*, 4(43), 1686–1699. doi:10.21105/joss.01686

**Appendix 1: Code to obtain `subject_merged`**

```
for (file_name in enum(filenamees)) {
  cat(file_name, fill = TRUE)
  subject_data = read.table(
    file_name[2],
    stringsAsFactors = FALSE,
    fill = TRUE,
    header = TRUE
  )
  if (nrow(subject_data) != 100) {
    stop('unexpected trial number: ', nrow(subject_data))
  }
  rts = aggr_neat(
    subject_data,
    rt,
    group_by = c('color', 'valence'),
    method = mean,
    prefix = 'rt',
    filt = (rt > 150 & response == 'correct')
  )
  ers = aggr_neat(
    subject_data,
    response,
    group_by = c('color', 'valence'),
    method = 'incorrect',
    prefix = 'er',
    filt = (response %in% c('correct', 'incorrect'))
  )
  er_overall = aggr_neat(subject_data,
    response,
    method = 'incorrect',
    filt = (response %in% c('correct', 'incorrect')))$aggr_value
  rbind_loop(
    subjects_merged,
    subject_id = subject_data$subject_num[1],
    condition = subject_data$condition[1],
    gender = subject_data$gender[1],
    age = subject_data$age[1],
    er_overall = er_overall,
    rts,
    ers
  )
}
```

Appendix 2: The `table_neat` function

```
table_neat(
  list(
    aggr_neat(subjects_merged, rt_green_negative, round_to = 0),
    aggr_neat(subjects_merged, rt_green_positive, round_to = 0),
    aggr_neat(subjects_merged, rt_red_negative, round_to = 0),
```

```
    agr_neat(subjects_merged, rt_red_positive, round_to = 0)
  ),
  group\_by = 'condition'
)
```

Citation

Lukács, G. (2021). neatstats: An R package for a neat pipeline from raw data to reportable statistics in psychological science. *The Quantitative Methods for Psychology*, 17(1), 7–23. doi:[10.20982/tqmp.17.1.p007](https://doi.org/10.20982/tqmp.17.1.p007)

Copyright © 2021, Lukács. This is an open-access article distributed under the terms of the Creative Commons Attribution License (CC BY). The use, distribution or reproduction in other forums is permitted, provided the original author(s) or licensor are credited and that the original publication in this journal is cited, in accordance with accepted academic practice. No use, distribution or reproduction is permitted which does not comply with these terms.

Received: 20/05/2020 ~ Accepted: 12/02/2021