R Notebook for naturalness/acceptability judgment experiments

Masha Onoeva

Table of contents

Info

Hi! This R notebook describes the steps that are required for the analysis of naturalness/acceptability judgment linguistic experiments. I use the experiment that I did with Radek Šimík testing Russian negated polar questions in different contexts as the example data (see Onoeva and Šimík 2023). We had several sub-experiments, here I report on one because code is the same for all of them. The experiment was run on LRex.

Design

The goal of the experiment was to find out how natural are various negated polar questions in different contexts. There were three independent variables, each had two manipulations, so the design was 2 x 2 x 2. There variables were:

1. verb: V1 li and V2
2. indefinite: ni and nibud
3. context: neutral and negative

Participants had to assess questions in different context on the scale from 1 very unnatural to 7 very natural. It was the dependent variable.

Within-items, between subjects?

Files

The csv file with raw results is available in this repo (perhaps I can also load a spreadsheet with all conditions that I used for LRex?). There are also two files with the script – qmd and md. The first one is a Quarto RMarkdown script from RStudio, the second one is a pretty version for GitHub (it is easier to follow online).

Loading and cleaning data

```
library(tidyverse) # THE package, it contains ggplot2, tidyr, dplyr, readr etc.
library(gt) # for pretty markdown tables, needed for notebook rendering
```

There is an option to download a version without abandoned trials from LRex and I load it here. Then I'm setting the working directory and loading data. I have two ways here:

- standard local set up

```
# standard way of setting the directory locally on your machine
setwd("/Users/maria.onoeva/Desktop/new_folder/GitHub/stat-repo")

# loading all data
all_df <-
  read_delim("data/queslav_neg_mo_RESULTS_2023-03-06-0953_noaband.csv", ";",
                                                escape_double = FALSE,
                                                trim_ws = TRUE,
                                                show_col_types = FALSE)
```

- this one is more convenient for sharing online and with someone

```
library(here) # sets the dir
# since I load it online, it's more convenient to do it via "here" package
# loading all data
all_df <-
  read_delim(here("data", # showing path to the folder with the file
                  "queslav_neg_mo_RESULTS_2023-03-06-0953_noaband.csv"), ";",
                                              escape_double = FALSE,
                                              trim_ws = TRUE,
                                              show_col_types = FALSE)
# I thank Masha Razguliaeva for the tip!
```

Then I remove the unnecessary example items.

```
# removing example items
main_df <- all_df %>%
  filter(materials != "1_examples")
```

Fillers and unreliable participants

You can see the number of participants on LRex but just to double-check it I'll run a couple of lines here as well.

```
# counts all participants
main_df %>%
  distinct(participant) %>%
  summarize(total_part = n())
```

```
# A tibble: 1 x 1
  total_part
       <int>
1         95
```

```
# summarizes items for all participants
main_df %>%
  group_by(participant) %>%
  summarize(items = n())
```

```
# A tibble: 95 x 2
   participant items
         <dbl> <int>
 1           1    82
 2           2    82
 3           3    82
 4           4    82
 5           5    82
 6           6    82
 7           7    82
 8           8    82
 9           9    82
10          10    82
# i 85 more rows
```

The next step is to extract all filler items. These items are used to test reliability of the participants. Here I have 10 such items, people had to assess 3 items as 'bad' or 1-4 and the rest as 'good' 5-7.

```
# creating a new df with the filler items only
fillers_only <- main_df %>%
  filter(materials == "f9_filler")

# creating a new column for checking if fillers are good or not
fillers_only$filler_answer <- 0
fillers_only$filler_answer <- as.numeric(fillers_only$filler_answer)

# rename filler items: the first three items were bad, the rest were good
fillers_only$condition[fillers_only$item %in% c("1", "2", "3")] <- 'bad'
fillers_only$condition[fillers_only$condition != "bad"] <- 'good'
```

Now I'm assigning '1' to filler_answer column if these filler items were assessed correctly. In other words, if bad fillers have 1-4 and good 5-7. If these conditions are not met, the value remains '0'.

```
# bad fillers
fillers_only$filler_answer[which(grepl('bad', fillers_only$condition) &
                                 grepl('1|2|3', fillers_only$rating1))] <- 1

# good fillers
fillers_only$filler_answer[which(grepl('good', fillers_only$condition) &
                                 grepl('5|6|7', fillers_only$rating1))] <- 1
```

So now I have the column which tells me how each participant assessed each filler item. I can measure their reliability simply by counting how good they were in the fillers. In the table below, the first participant has mean '1', so they have assessed all fillers correctly as I expected them. If mean is lower than 80 %, a participant is unreliable.

```
filler_results <- fillers_only %>%
  group_by(participant) %>%
  summarize(Mean = mean(filler_answer, na.rm=TRUE))

filler_results
```

```
# A tibble: 95 x 2
   participant  Mean
         <dbl> <dbl>
 1           1   1
 2           2   0.9
 3           3   0.8
 4           4   0.6
 5           5   0.6
 6           6   0.6
 7           7   0.9
 8           8   0.7
 9           9   0.9
10          10   0.9
# i 85 more rows
```

```
# how in general the participants went through fillers
mean(filler_results$Mean)
```

```
[1] 0.844
```

Now I need to find unreliable participants. This is done quickly, just to find the people who have means lower than 0.8. This is quite a high threshold but 68 participants is still fine.

```
unreliable_participants <- filler_results %>%
  filter(Mean < 0.8) # I have 27 unreliable participants
```

Then I remove unreliable participants from reliable ones.

```
fillers_only_reliable <- anti_join(filler_results, unreliable_participants,
                           by = "participant")

# testing by applying mean to the reliable df
mean(fillers_only_reliable$Mean)
```

```
[1] 0.924
```

Data sets

In this experiment, we had one big experiment and several smaller, see summary of the materials column.

```
main_df %>%
  group_by(materials) %>%
  summarise()
```

```
# A tibble: 10 x 1
   materials
   <chr>
 1 e1_main
 2 f1_nibud
 3 f2_razve_pos
 4 f3_razve_neg
 5 f4_slucajno_neg
 6 f5_slucajno_pos
 7 f6_ctoli_neg
 8 f7_ctoli_pos
 9 f8_repetitive
10 f9_filler
```

I'm going to separate them into several data frames. But first, it's necessary to remove the filler items and unreliable participants.

```
main_df1 <- main_df %>%
  filter(materials != "f9_filler")

main_df1 %>%
  group_by(materials) %>%
  summarise() # no filler in the summary
```

```
# A tibble: 9 x 1
  materials
  <chr>
1 e1_main
2 f1_nibud
3 f2_razve_pos
4 f3_razve_neg
5 f4_slucajno_neg
6 f5_slucajno_pos
7 f6_ctoli_neg
8 f7_ctoli_pos
9 f8_repetitive
```

Removing unreliable participants and checking the number.

```
main_df2 <- anti_join(main_df1, unreliable_participants,
                          by = "participant")

main_df2 %>%
  distinct(participant) %>%
  summarize(total_part = n())
```

```
# A tibble: 1 x 1
  total_part
       <int>
1         68
```

Now I need to create a separate df for each materials set, they are stored in split_main_df1. It can be done using filter() but I try here group_split(). I can access each group later.

```
split_main_df1 <- main_df2 %>% group_split(materials)
```

Descriptive stat

The design for this experiment was 2 x 2 x 2.

verb	context	indefinite
V1 li	neutral	ni
V2	negative	nibud

The conditions were coded as letters in the spreadsheet for LRex, so first, I assign new comprehensible conditions, so it's easier to read the results. Not sure if it can be done in a more sophisticated way (ChatGPT says otherwise :unamused: :expressionless:).

```
# accessing the first experiment from the groups
e1_df <- split_main_df1[[1]]

# creating a new column for the first variable 'verb' and recoding
# to the readable form 4 conditions were V1 li, 4 -- V2
e1_df$verb <- 0
e1_df$verb[e1_df$condition %in% c("a", "c", "e", "g")] <- "V1 li"
e1_df$verb[e1_df$verb != "V1 li"] <- "V2"

# the same as above for the second variable 'context'
e1_df$context <- 0
e1_df$context[e1_df$condition %in% c("a", "b", "c", "d")] <- "neutral"
e1_df$context[e1_df$context != "neutral"] <- "negative"

# the same as above for the third variable 'indefinite'
e1_df$indef <- 0
e1_df$indef[e1_df$condition %in% c("a", "b", "e", "f")] <- "ni"
e1_df$indef[e1_df$indef != "ni"] <- "nibud"

# ChatGPT version: idk case_when function
# e1_df <- e1_df %>%
#   mutate(
#     indef = case_when(
#       condition %in% c("a", "b", "e", "f") ~ "ni",
#       TRUE ~ "nibud"
#     )
#   )
```

Table

There are two ways how to look at my data: check how they are similar and how they are different. For the first, I need measures of central tendency – mode, mean, median, for the second variability values – range, variance, standard deviation.

```
# creating one GIGA condition
# not necessary though, group_by() works just fine, but I'll need it further
e1_df <- e1_df %>%
 mutate(condition1 = paste(condition, context, verb, indef))
```

```r
library(DescTools) # for Mode()
e1_df$rating1 <- as.numeric(e1_df$rating1)

raw_summary <- e1_df %>%
  dplyr::group_by(indef, verb, context) %>%
  dplyr::summarize(Mode = Mode(rating1),
            Median = median(rating1),
            Mean = mean(rating1),
            Range = paste(range(rating1), collapse = "-"),
            Variance = var(rating1),
            SD = sd(rating1), # sd = sqrt(var(rating1))
            )

as_raw_html(raw_summary %>% gt(groupname_col = 'indef',
                               rowname_col = 'context') %>%
  cols_label(verb = 'Verb'))
```

	Verb	Mode	Median	Mean	Range	Variance	SD
ni							
negative	V1 li	1	2	2.76	1-7	3.56	1.89
neutral	V1 li	1	3	3.33	1-7	4.85	2.20
negative	V2	7	4	4.17	1-7	4.48	2.12
neutral	V2	7	5	4.36	1-7	4.93	2.22
nibud							
negative	V1 li	7	6	5.01	1-7	3.90	1.98
neutral	V1 li	7	6	5.89	1-7	2.10	1.45
negative	V2	7	5	4.85	1-7	3.65	1.91
neutral	V2	7	6	5.28	1-7	3.07	1.75

- **Mode** is the most popular ~~dude~~ number in the set. It's usually not a very useful value but here why not :grin:
- **Median** is a true central tendency value as it's in the middle but it's necessary to order the values first. It's resilient to outliers which can be good and bad at the same time.
- **Mean** is also known as average. It's like a parent who loves their kids equally, or ideal socialism, it shows the sum of all values divided by their number, so if everybody should get the same, they get mean.
- **Range** is min and max values, not super telling here but can be useful with different data.
- For **variance** mean is required. To calculate that one, each observed value has to be compared to the mean, then this difference must be squared (because it can be negative), after the sum of all of these squared differences should be divided by the number of

observed values (I'm aware about n and n-1 stuff, but there is no space for that). So it's average for squared differences from the mean. It's possible to do from median, I guess, but here it's calculated from mean.

- **Standard deviation (SD)** is the easiest, it's a square root from variance. Since differences from the mean were squared in the previous step, one needs to 'unsquare' that result. Perhaps my SDs are too high in some cases but this is what I'm going to investigate in my dissertation (Masha Razguliaeva's comment: for naturalness judgments SD might be higher than for grammaticality).

Stacked bar plot

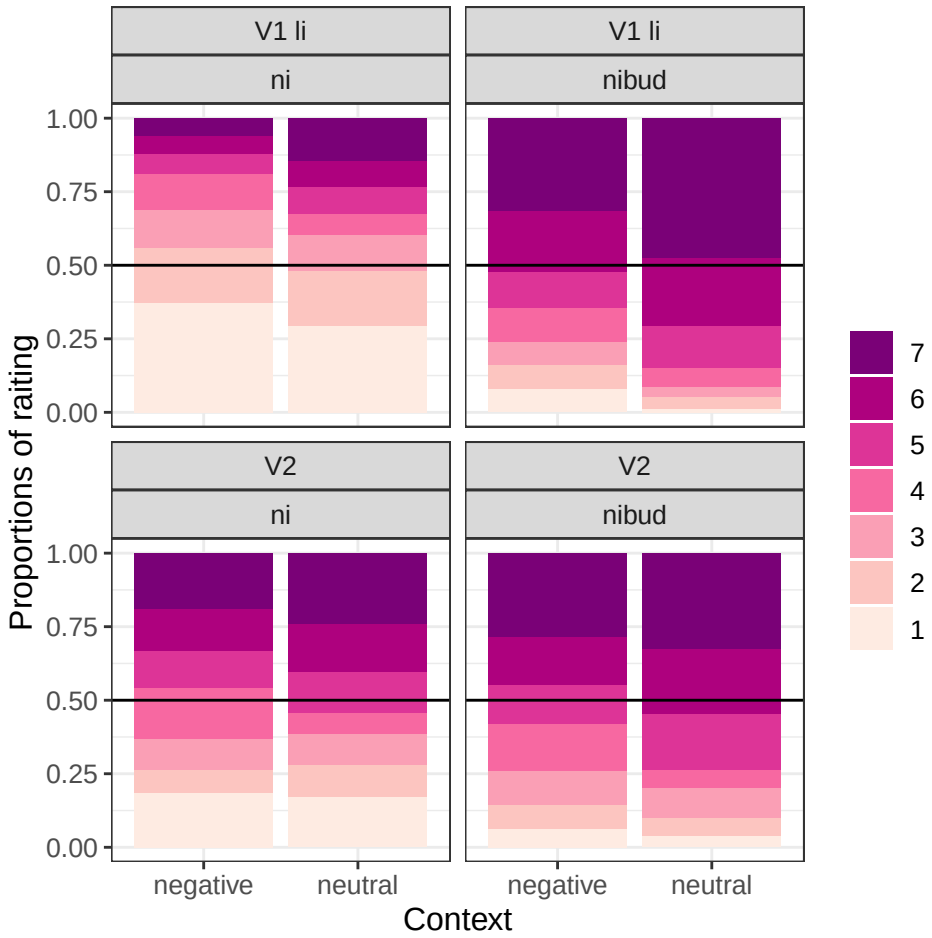
The table is cool but the next step is to plot the results. Before I do that I need to refactor and relevel ratings, so they are displayed properly (not upside-down).

```
# have to make as factor, otherwise error
e1_df$rating1 <- as.factor(e1_df$rating1)

# re-leveling ratings (I have this code from Anička Staňková)
e1_df_relevel <- e1_df %>%
  mutate(rating1 = fct_relevel(rating1,"7","6","5","4","3","2","1"))

# re-leveling verbs so they are showed differently in the plot
e1_df_relevel1 <- e1_df_relevel %>%
  mutate(verb = fct_relevel(verb,"V1 li", "V2"))
```

I have commented out some lines for the plot but they are mostly cosmetics that change size of text, etc. These might be useful for specific cases but here I don't need them.

```
e1_main_plot <- ggplot(e1_df_relevel1, aes(fill=rating1, x=context)) +
    geom_bar(position = "fill") +
    geom_hline(aes(yintercept=0.5), size=0.5) +
    facet_wrap(~verb+indef) +
  # coloring
    theme_bw() +
    scale_fill_brewer(palette = "RdPu", direction=-1) +
    theme(legend.position = "right",
          text = element_text(size = 12),
          # legend.text = element_text(size=20),
          # legend.key.size = unit(1, 'cm'),
          legend.title = element_blank())+
          # axis.text = element_text(size = 25),
```

```
          # axis.title = element_text(size = 25),
# axis.title.y = element_text(margin = margin(t = 0, r = 20, b = 0, l = 0)),
# axis.title.x = element_text(margin = margin(t = 20, r = 0, b = 0, l = 0))) +
# ggtitle("Stacked bar plot E1 (68 participants)") +
    xlab("Context") +
    ylab("Proportions of raiting")

e1_main_plot
```



On the x axis, there are contexts, on the y axis – proportions of ratings. The darkness of the bars indicates naturalness (dark means more natural). The black line that strikes through the plots is median.

It is also possible to save the plots using this code:

```
ggsave(e1_main_plot, file="e1_main1.eps",
       width = 35, height = 37, units = "cm", device="eps")
ggsave(e1_main_plot, file="e1_main1_pdf.pdf",
       width = 20, height = 20, units = "cm", device="pdf")
```

Interaction plot

The next step is to create an interaction plot. First, I do the calculations and then plot the results.

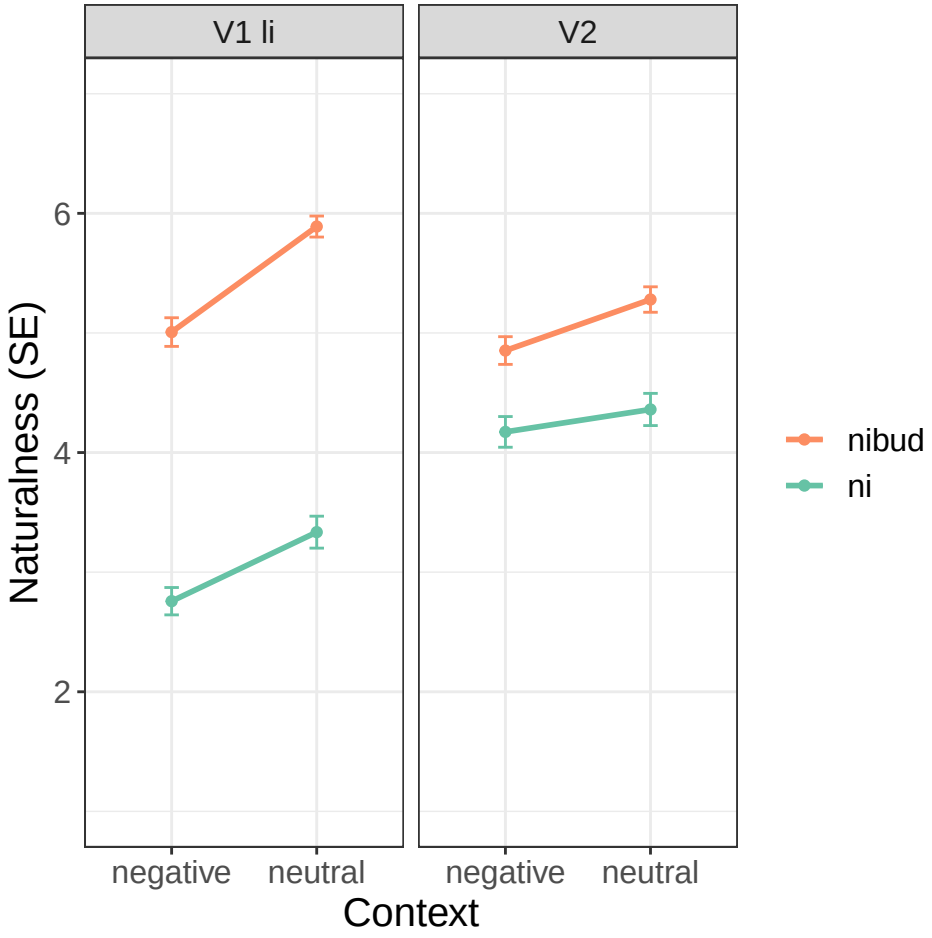
:exclamation: I use here the results **before re-leveling**.

```
# This code is based on Radek Šimík's code.

library(Rmisc) # for summarySE, needed just here

# I load the df to inter_df
inter_df <- e1_df

# changing rating1 to numeric
inter_df$rating1 <- as.numeric(inter_df$rating1)

# calculating interactions
tab_inter <- summarySE(inter_df, measurevar="rating1",
                       groupvars = c("context", "verb", "indef"))
```

The plot code might look crazy, but I've commented things out and most of the lines are cosmetics.

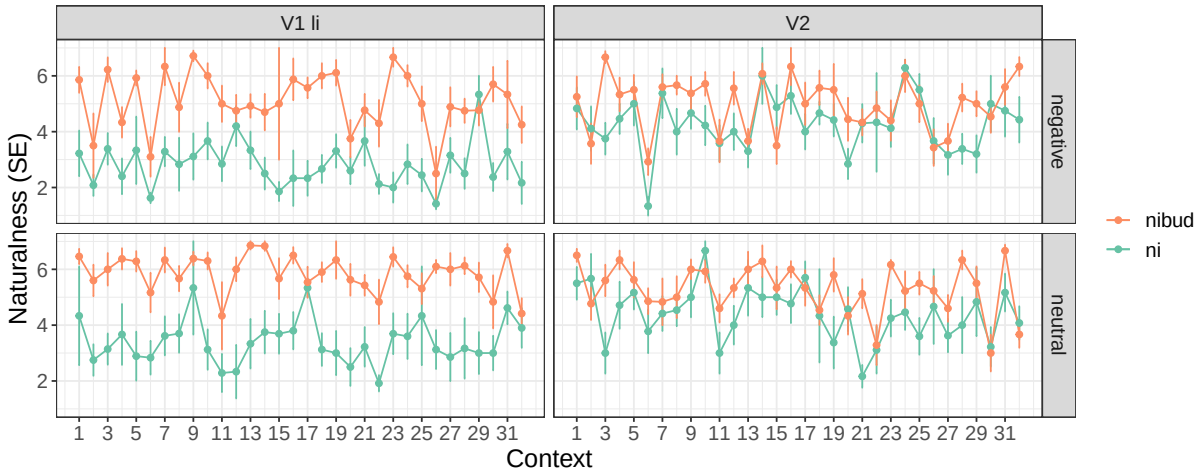
```
# plotting
inter_plot <- ggplot(tab_inter, aes(x=context, y=rating1,
                                    colour=indef, group=indef)) +
    geom_errorbar(aes(ymin=rating1-se, ymax=rating1+se), width=.1) +
    facet_wrap(~verb) +
    theme_bw() +
    geom_line(size = 1) +
    theme(
        text = element_text(size = 15),
        # legend.text = element_text(size=30),
        # legend.key.size = unit(1, 'cm'),
        legend.title=element_blank())+
```

```
        # legend.position = c(0.8, 0.15),
        # axis.text = element_text(size = 25),
        # axis.title = element_text(size = 25),
# axis.title.y = element_text(margin = margin(t = 0, r = 20, b = 0, l = 0)),
# axis.title.x = element_text(margin = margin(t = 20, r = 0, b = 0, l = 0))) +
    geom_point() +
    xlab("Context") +
    ylab("Naturalness (SE)") +
    coord_cartesian(ylim = c(1, 7)) +
    #scale_y_continuous(breaks = pretty_breaks(4)) +
    guides(colour = guide_legend(reverse=TRUE))  +
    scale_color_brewer(palette = "Set2")

inter_plot
```



The same plot as above but for each item. (Perhaps I can use conditions as color?)

```r
tab_inter_items <- summarySE(inter_df, measurevar="rating1",
                         groupvars = c("item", "context", "verb", "indef"))

# plotting
inter_plot_items <- ggplot(tab_inter_items, aes(x=item,y=rating1,
                                                colour=indef, group=indef)) +
    geom_errorbar(aes(ymin=rating1-se, ymax=rating1+se), width=.1) +
    #facet_wrap(~verb+context) +
    facet_grid(vars(verb),  rows = vars(context)) +
    theme_bw() +
    geom_line() +
    theme(
        text = element_text(size = 15),
        # legend.text = element_text(size=30),
        # legend.key.size = unit(1, 'cm'),
        legend.title=element_blank())+
        # legend.position = c(0.8, 0.15),
        # axis.text = element_text(size = 25),
        # axis.title = element_text(size = 25),
# axis.title.y = element_text(margin = margin(t = 0, r = 20, b = 0, l = 0)),
# axis.title.x = element_text(margin = margin(t = 20, r = 0, b = 0, l = 0))) +
    geom_point() +
    xlab("Context") +
    ylab("Naturalness (SE)") +
    coord_cartesian(ylim = c(1, 7)) +
    #scale_y_continuous(breaks = pretty_breaks(4)) +
    guides(colour = guide_legend(reverse=TRUE))  +
    scale_color_brewer(palette = "Set2") +
    scale_x_continuous(breaks=seq(1, 32, by = 2))

inter_plot_items
```



Inferential stat

The purpose of inferential statistics is to determine whether the experimental results were not produced by chance. In other words, there is a chance that our results are like this by a mere accident and the interactions between the variables are just coincidences. But using some smart tests one can eliminate this possibility. Inferential stat also helps to clearly articulate interactions which are found in the sample. Various models are used for this, so stay tuned.

I begin with the most basic basics and continue further.

Standard error

Unlike SD, SE is an inferential statistic in this case, so it can only be estimated. In the ideal case if one knows the population mean, SE can be calculated. But there is no mean for all Russian speakers, so the input values for SE are replaced with the sample mean and thus the sample SD.

In the table below, there are means from the descriptive table together with Standard Deviations (SD), Standard Errors (SE) and Relative SEs. SEs were already present in the interaction plot, it's represented as those little bars around the dots. There is an option to plot SD but those plots look crazy.

```
e1_df$rating1 <- as.numeric(e1_df$rating1)

# using dplyr:: here as the functions interfere with plyr
raw_summary1 <- e1_df %>%
  dplyr::group_by(indef, verb, context) %>%
  dplyr::summarize(Mean = mean(rating1),
```

```
            SD = sd(rating1), # sd = sqrt(var(rating1))
            SE = sd(rating1)/sqrt(length(rating1)),
            # RSD = sd(rating1)/mean(rating1) * 100,
            # Conf = mean(rating1)/sd(rating1),
            RSE = sd(rating1)/sqrt(length(rating1))/mean(rating1) * 100
            )

as_raw_html(raw_summary1 %>% gt(groupname_col = 'indef',
                                rowname_col = 'context') %>%
  cols_label(verb = 'Verb',
             RSE = 'RSE (%)'))
```

	Verb	Mean	SD	SE	RSE (%)
ni					
negative	V1 li	2.76	1.89	0.1144	4.15
neutral	V1 li	3.33	2.20	0.1335	4.00
negative	V2	4.17	2.12	0.1283	3.07
neutral	V2	4.36	2.22	0.1346	3.09
nibud					
negative	V1 li	5.01	1.98	0.1198	2.39
neutral	V1 li	5.89	1.45	0.0878	1.49
negative	V2	4.85	1.91	0.1158	2.39
neutral	V2	5.28	1.75	0.1063	2.01

SE is supposed to tell how close my sample mean is to the population mean. In other words, if I would do this experiment a couple more times, new possible means could differ this SE much from each other. The formula for SE is pretty easy, n is a number of observations,

$$SE = \frac{SD}{\sqrt{n}}$$

But is it too big or I'm fine? RSE indicates that. It represents the size of SE relative to the mean, so as percentage. They say if RSE is below 10 %, then SE is relatively small as they are here.

t-test

Student's t-test is unlikely the best test for this experiment but it's calculated from SE and mean, so why not to have it? :wink:

The formula for t-value is the following, looks complicated, meh.

$$t = \frac{\bar{x} - \mu}{SE}$$

But if one rewrites it, it becomes super clear:

$$t \times SE = \bar{x} - \mu$$

So t-value says how much my sample mean or x-bar is far from the population mean or mu in SEs. If I need many SEs, the difference is big, if little, the difference is not big.

Since I have only one sample and no population mean, I need to find mu to compare it with, shall it be 0. So now there are two hypotheses:

1. the sample mean = 0
2. the sample mean is significantly different from 0

I'm going to use here a one sample t-test because I have one sample. There are also paired samples t-test and independent samples t-test. For the first, it's required to measure one sample two times, for the second, one needs at least two different samples, then their differences are compared.

t-test can be run like this for one condition. The output says that it is unlikely that the observed mean was due to chance as t = 25 and p < 2e-16.

```
cond_1 <- e1_df %>% filter(condition1 == 'a neutral V1 li ni')
t.test(cond_1$rating1)
```

```
    One Sample t-test

data:  cond_1$rating1
t = 25, df = 271, p-value <2e-16
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 3.07 3.60
sample estimates:
mean of x
     3.33
```

Or it can be done for all 8 conditions at once. All t-values are huge, up to t = 67 and ps < 0.05.

```
library(broom)

# packing all conditions into a vector
conditions <- e1_df %>%
  dplyr::distinct(condition1) %>%
  pull(condition1)

# applying t.test function to each condition and creating a table
t_test_results <- map_df(conditions, function(cond) {
  result <- e1_df %>%
    filter(condition1 == cond) %>%
    pull(rating1) %>%
    t.test() %>%
    tidy()

  result$Condition <- cond
  return(result)
})

as_raw_html(t_test_results %>%
              gt() %>%
              cols_move_to_start(columns = Condition) %>%
              cols_label(estimate = 'Mean',
                         statistic = 't-value',
                         p.value = 'p-value'))
```

Condition	Mean	t-value	p-value	parameter	conf.low	conf.high	method	a
c neutral V1 li nibud	5.89	67.1	9.12e-171	271	5.72	6.06	One Sample t-test	tw
d neutral V2 nibud	5.28	49.7	3.99e-138	271	5.07	5.49	One Sample t-test	tw
e negative V1 li ni	2.76	24.1	2.16e-69	271	2.53	2.98	One Sample t-test	tw
f negative V2 ni	4.17	32.5	1.43e-95	271	3.92	4.43	One Sample t-test	tw
g negative V1 li nibud	5.01	41.8	3.70e-120	271	4.77	5.24	One Sample t-test	tw
h negative V2 nibud	4.85	41.9	1.90e-120	271	4.62	5.08	One Sample t-test	tw
a neutral V1 li ni	3.33	25.0	3.10e-72	271	3.07	3.60	One Sample t-test	tw
b neutral V2 ni	4.36	32.4	3.31e-95	271	4.10	4.63	One Sample t-test	tw

Two-tailed and one-tailed?

It depends on predictions. If I predict that my difference will be somehow direct towards positive or negative t, I use one-tailed test. Two-tailed is used when one just looks for some difference and doesn't care about positive or negative direction. Two-tailed is more conservative though which is good for the type I error exclusion. In principle, here I assume that my results should be bigger than 0, so I can use one-tailed test but I don't really care that much, so whatevs.

Why isn't t-test good for this experiment?

One of the reasons is that I have a factorial design, so there are multiple conditions that have to be compared to each other. t-test compares mean with 0. It says if the sample mean is significantly different and if it is or isn't obtained by chance. But I want to know what influence manipulations have on the dependent variable. I can pair conditions and run it but then it's ANOVA which is the next test. And it feels a bit wrong when I compare it with 0, maybe it's more correct to compare it with 1 'completely unacceptable' or 7 'completely acceptable'?

Another reason is that my sample data are not that independent as t-test requires them to be. I have to account for variability in participants and in items. There are possibilities how to account for it for t-test (aggregate all) but it's better to do a linear mixed model.

ANOVA

(All taken from this web.)

ANOVA (Analysis of Variance) suits a bit better for the experiment but it's still not quite there yet. With t-test I compared two things only, now I can do it with all my conditions. ANOVA determines whether the means I have in my conditions are significantly different from each other. My hypotheses for ANOVA are:

- H0: mean(a) = mean(b) = mean(c) = mean(d) = mean(e) = mean(f) = mean(g) = mean(e)
- H1: **at least one** of them is different

One of the requirements for this test is to have a continuous quantitative dependent variable. I have a Lickert scale, it's an ordinal scale and those are considered categorical but let's pretend that it's continuous. As for the independent variables, they have to be qualitative with at least two levels.

Other requirements are normality and equality of variances. For the first one, with this sample size distribution should not be necessary normal, but the second requirement has to be checked. Levene's test suits for this which is run below. p-value is pretty small, so the null hypothesis

that the variances are equal is rejected. I need to consider for that in the model and use Welch ANOVA.

```
e1_df$condition = as.factor(e1_df$condition)
car::leveneTest(rating1~condition, e1_df)
```

```
Levene's Test for Homogeneity of Variance (center = median)
        Df F value Pr(>F)
group    7    14.8 <2e-16 ***
      2168
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Welch ANOVA is present below. p-value is small, so we reject H0 that all groups are equal. But it only means that at least one is different but which one? And what is the relation between other variables?

```
oneway.test(rating1~condition,
            data = e1_df,
            var.equal = FALSE) # Welch ANOVA as variences aren't equal
```

```
    One-way analysis of means (not assuming equal variances)

data:  rating1 and condition
F = 90, num df = 7, denom df = 927, p-value <2e-16
```

Linear model and linear mixed model

Cumulative Link Mixed Model

I'll come back with more :v: :sparkles:

```
library(modelsummary)
library(lmerTest)
library(ordinal)
library(gtsummary)

e1_df$rating1 = as.factor(e1_df$rating1)
```

```
stat_E1 <- clmm(rating1 ~ verb * indef * context +
  (1 | participant) + (1 | item),
  contrasts = list(verb="contr.sum",indef="contr.sum", context="contr.sum"),
  data=e1_df)

summary(stat_E1)
```