Lab 10

Table of contents

Introduction to data preprocessing

Lab class

In Lab 8, we have had a closer look at PsychoPy output files. Now, it is time to start analysing these output files. This is where your statistics knowledge becomes relevant for the practicals: Using an example output file, today we will calculate means, medians and standard deviations (SDs).[1]

What SPSS needs and what we get from PsychoPy

Let's assume your aim is to find out if RTs on incongruent flanker trials are on average significantly slower than RTs on congruent flanker trials. To investigate this, you would have participants complete a number of trials from both conditions and run an inferential statistical test on the data. Remember that this is a within-subject design, as the same participants complete all levels of the IV, so for a parametric analysis this inferential test would be a paired-samples *t*-test.

What SPSS needs: One row per participant.

Think back to the data file for Statistics lecture 8 ("Comparing means - part 2"). In this file, one row corresponded to one participant, and for each participant you had two data points corresponding to the two conditions:

[1] If you can't remember how exactly these measures of central tendency and dispersion are calculated, you might want to return to your statistics lectures and look this up.

	No_Cloak	Cloak
1	3	4
2	1	3
3	5	6
4	4	6
5	6	8
6	4	5
7	6	5
8	2	4
9	0	2
10	5	5
11	4	7
12	5	5

This type of file is what SPSS needs to run a paired-samples *t*-test: **For a within-subjects design with two conditions, we need two data points for each participant, both in the same row.**

Accordingly, for the flanker task you would need one data point for congruent trials and one data point for incongruent trials for each participant.

What we typically get from PsychoPy: Many rows per participant.

If your flanker task had, say, 72 experimental trials (half congruent, half incongruent), there would be 36 rows per condition in your PsychoPy output file! **As SPSS expects one row per participant, we need a summary measure to represent the performance of each participant in both conditions** (i.e., a mean or median). The remainder of this chapter and the next chapter explain one approach to obtaining this summary measure.

How to get from PsychoPy output to SPSS input

The simplest approach for creating our summary measure would be to simply calculate the mean RT for all trials from a condition and participant. However, this approach has the potential shortcomings listed below.

Extreme RTs

First, there might be extremely fast as well as extremely slow responses. **Extremely fast responses** (say, faster than 100-150 ms) are likely **anticipatory responses**. That is, participants anticipated the appearance of the stimulus and pressed a response key before properly processing the stimulus.

Why 100-150 ms? The reason for this is that even in the macaque it takes about 70 ms on average for signals from the retina to arrive at the primary visual cortex (Lamme & Roelfsema, 2000). In a choice reaction time task, once these signals arrive, lower and higher order visual processing areas must process the object identity ("Which target is it?"). Once the target has been identified, the correct response must be identified (e.g., "H requires a left-hand response"). Finally, a motor signal must be sent to an effector (i.e., a finger must press down one of the response keys).

This is not to say that all of these processes occur strictly sequentially, but taking the various processing steps into account, it seems extremely unlikely that participants can produce valid responses (not just lucky guesses) before 100-150 ms (also see Whelan, 2008).

On the other hand, there might be **extremely slow responses**. These are likely due to **lapses of attention or external distractions**. Or, if your trials have infinite length, a participant might also have taken a break in the middle of your experiment! As these slow responses are not a direct consequence of the processing requirements of the task, an argument can be made for excluding them as well. The exact cut-off will be different for different tasks. For a straightforward flanker task run with healthy young participants, RTs which are longer than, say, 3 seconds might be considered extreme RTs.

Incorrect RTs

In addition, including **error trials** would bias our measure of central tendency:

- There is strong evidence that error trials in speeded reaction time tasks are faster than correct trials (e.g., Smith & Brewer, 1995).
- Errors also occur more frequently on incongruent trials than on congruent trials (e.g., Derrfuss et al., 2021).

Including error trials would underestimate mean RTs because errors are typically fast. This effect would be disproportionately large for incongruent trials, where errors are more common.

As a result, including RTs from incorrect trials would reduce the interference effect, making it less likely to detect a significant difference.[2]

Outlier RTs

Finally, there may be outliers. In the HHG framework, extreme RTs are defined in absolute terms, whereas **outlier RTs are relative to other RTs within the same condition for the same participant**.[3]

[2] In fact, we recently showed that a very similar issue has been a confound in many publications investigating post-error slowing (Derrfuss et al., 2021).

[3] Note that these terms are used inconsistently across the literature, so always define them explicitly.

There are several ways to identify and reject outliers: based on standard deviation, interquartile range, absolute median deviation, or by trimming (i.e., removing a fixed percentage of trials—e.g., the fastest and slowest 20%).

For our lab classes, we will focus on SD-based outlier rejection—a pragmatic choice. This method has a drawback: outliers inflate the SD, so extreme values can mask other outliers. However, SDs are easy to compute and sufficient for illustrating the concept. Moreover, a recent study (Berger & Kiefer, 2021) suggests that SD-based methods are relatively unbiased. That said, this conclusion is based on simulated data, and it remains unclear how well those simulations reflect real-world outliers. Furthermore, another study argues that outlier rejection may sometimes do more harm than good (Miller, 2023).

In short, the topic is still debated. Our goal is to ensure you understand how to apply SD-based outlier rejection so you can use it when appropriate.

Data preprocessing with Excel

Lab class

This chapter explains how to apply the preprocessing steps described in the previous chapter in Excel for one participant. That is, for this participant, we will:

- Remove extreme RTs.
- Remove incorrect trials.
- Remove outlier RTs.

Then, we will calculate measures of central tendency for this participant. Please note that not all researchers will apply all of these steps. However, you should be aware of the possible processing steps and know how to implement them using Excel. In addition, we will also calculate accuracies.

The processing steps, explained in detail below, are the following:

- Step 1: Convert reaction times to milliseconds.
- Step 2: Calculate overall accuracy.
- Step 3: Remove trials with extreme RTs.
- Step 4: Calculate condition-specific accuracies.
- Step 5: Calculate condition-specific mean RTs (before outlier removal).
- Step 6: Calculate SDs and thresholds for outlier removal.
- Step 7: Calculate condition-specific mean RTs (after outlier removal).

Please note that there are other ways to achieve the same aims in Excel. We have opted for an approach that is relatively verbose and keeps the formulas as simple as possible.

Preprocessing Shiny app

In our view, it's often helpful to visualise the effects of these preprocessing steps. Therefore, we have created an interactive app for visualising the effects of preprocessing choices. The aim of this app is to allow you to clearly see the effects of preprocessing choices on means and SDs for individual conditions for individual participants. You can test the app using this example PsychoPy output file.

A partially worked example

Please download the partially worked example below. This is an output file from a single participant who previously completed a letter flanker task (note that this experiment also included a flanker distance manipulation which we will ignore for our analysis).

Click here to download the partially worked example.

Please note that the following modifications were made to the original output file:

- The file was converted to an Excel `.xlsx` file.
- Some columns not relevant for the current analysis were removed.
- Rows corresponding to the practice trials were removed.
- Columns that were part of the original output file are highlighted in green.
- Columns added to the output file are highlighted in yellow.
- Analysis results are in cells highlighted in blue.

The most relevant columns for our purposes are:

- `congruency`: whether the trial was congruent (e.g., "HHHHH") or incongruent (e.g., "SSHSS")
- `response.corr`: the accuracy of the response (0 = incorrect, 1 = correct)
- `response.rt`: the response time (in seconds)

In the example file, we have already done all the calculations for the congruent trials. We will walk you through the formulas common to all trials and those specific to the congruent trials.

Self-study

Remove practice trials

The following short video demonstrates how to adjust column widths, remove practice trials, and remove unnecessary columns (note that the video has no sound).

This section contains content which is not available in the PDF version. Please visit the online version to see it.

Step 1: Convert reaction times to milliseconds

Aim: Convert RTs to ms.

Why do we do this?: This is done for convenience only. Most of us find it easier to use integers as opposed to decimals. Also, the numbers are shorter when using ms (e.g., 527 ms vs. 0.527 s).

In the partially worked example, we created a new column `RTms` and added the following formula to the first cell:

```
=H2*1000
```

That is, we simply multiplied the RT in seconds by 1,000 to obtain milliseconds. This formula was then copied downwards by dragging the fill handle (alternatively, you can also double-click on the fill handle).

Step 2: Calculate the overall accuracy

Aim: Calculate overall accuracy, taking into account all trials.

Why do we do this?: Sanity check of the data. If the overall accuracy is close to chance, you should probably exclude this participant.

> Chance performance
>
> What is chance performance? Chance performance is the average accuracy you would achieve by pressing response keys randomly. For a task with two response alternatives, chance performance would on average be 50%. That is, if you were to press keys randomly, you would on average still be correct 50% of the time. For a task with four response alternatives, chance performance would on average be 25%.

This is how to calculate accuracy:

$$Overall\ accuracy = \frac{Number\ of\ correct\ trials}{Total\ number\ of\ trials}$$

We used `COUNTIF` and `COUNT` to calculate the overall accuracy in cell `U5` in the partially worked example. As the accuracy data are in cells `G2` to `G73`, this is what the formula should look like:

```
=ROUND((COUNTIF(G2:G73, 1)/COUNT(G2:G73))*100,1)
```

In words:

- Count all the correct trials in cells `G2` to `G73` (i.e., the ones where the value is 1).
- Count all trials in cells `G2` to `G73`.
- Multiply by 100 to get a percentage (optional).
- Round the result to 1 decimal place (optional).

Without `ROUND`:

```
=(COUNTIF(G2:G73, 1)/COUNT(G2:G73))*100
```

Without percentage calculation:

```
=COUNTIF(G2:G73, 1)/COUNT(G2:G73)
```

Step 3: Remove trials with extreme RTs

Aim: Remove trials that are either extremely fast (anticipatory responses) or extremely slow (attentional lapses or distractions).

Why do we do this?: See Section .

In the partially worked example, we removed trials with extreme RTs from our column with the accuracy information. To this aim, we created a column called `accNoExtremes`. We added the following formula to the first cell and copied it downwards:

```
=IF(AND(I2>=150, I2<=3000), G2, "")
```

In words:

- If the RT is slower than 150 ms (`I2>=150`) *and*
- ...if the RT is faster than 3 seconds (`I2<=3000`)
- ...then copy the accuracy information to the new column.

Second, we removed trials with extreme RTs from our column with the RT information. To this aim, we created a column called `rtNoExtremes`. We added the following formula to the first cell and copied it downwards:

```
=IF(AND(I2>=150, I2<=3000), I2, "")
```

In words:

- If the RT is slower 150 ms (`I2>=150`) *and*
- ...if the RT is faster than 3 seconds (`I2<=3000`)
- ...then copy the RT information to the new column.

Finally, in cell `U8` we calculated the percentage of extreme RTs:

```
=ROUND((COUNTIF(J2:J73,"")/ROWS(J2:J73))*100,1)
```

In words:

- Count the empty cells in the range J2 to J73 (`COUNTIF(J2:J73,"")`).
- Divide this number by the number of rows in the range J2 to J73 (`ROWS(J2:J73)`).
- Optional: Multiply by 100 and round to one decimal place.

Formula without percentage calculation and rounding:

```
=COUNTIF(J2:J73,"")/ROWS(J2:J73)
```

Step 4: Calculate condition-specific accuracies

Aim: Calculate the accuracies for all experimental conditions (ignoring extreme RTs).

Why do we do this?: We would like to know if our experimental conditions had an influence on accuracy.

In the partially worked example, we created two new columns just for congruent and incongruent trial accuracies called `conAccNoExtremes` and `inconAccNoExtremes`, respectively.

Formula for column `conAccNoExtremes`: `=IF(D2="con", J2, "")`

Using this new column, we calculated the accuracy in congruent trials in cell `U12` with the following formula:

```
=ROUND((COUNTIF(L2:L73, 1)/COUNT(L2:L73))*100, 1)
```

In words:

- Count the cells in the range `L2` to `L73` that are equal to 1 (`(COUNTIF(L2:L73, 1)`).
- Count the cells in the range `L2` to `L73` that are any number (i.e., 0 or 1 in our case) (`(COUNT(L2:L73)`).
- Divide the first count by the second count (`/`).
- Optional: Multiply by 100 and round to one decimal place.

Step 5: Calculate condition-specific mean RTs (before outlier removal)

Aim: Calculate condition-specific mean RTs after removing incorrect trials.

Why do we do this?: Calculating mean RTs before removing outliers is a necessary step if one is using an SD-based approach for outlier rejection. We also remove incorrect trials as there is good evidence that these are typically faster than correct trials.

In the partially worked example, we created two new columns just for congruent and incongruent RTs called `conRTNoExtremesCorr` and `inconRTNoExtremesCorr`, respectively.

Formula for column `conRTNoExtremesCorr`: `=IF(AND(D2="con", J2=1), K2, "")`

In words:

- If the trial is congruent *and*
- ...if the trial is correct
- ...then copy the RT to the new cell.

We then copied the formula downwards.

Using this new column, we then calculated the mean RT in congruent trials in cell `U17` with the following formula:

```
=ROUND(AVERAGE(N2:N73), 0)
```

In words:

- Average trials in range N2 to N73.
- Round to 0 decimal places.

Step 6: Calculate SDs and thresholds for outlier removal

Aim: Calculate condition-specific mean RTs after removing outliers based on standard deviations (SDs).

Why do we do this?: Calculating SDs and SD-based outlier rejection thresholds is a necessary step if one is using an SD-based approach for outlier rejection.

Frequently, RTs located 2 or more SDs away from the mean are considered outliers (you might also find 2.5 SDs or 3 SDs in the literature). To illustrate this approach: Imagine a participant's mean RT is 1,000 ms and their SD is 200 ms. If you apply the 2 SDs rule, you would remove all RTs below 600 ms and above 1,400 ms from the analysis.

As a first step, we need to calculate the SDs of our two trial types. In the partially worked example, we used the following formula in cell `U21` for congruent trials:

```
=ROUND(STDEV.S(N2:N73), 0)
```

We then determined the lower and upper thresholds for outlier rejection for both conditions. The basic idea is:

- Lower threshold: `Mean - 2 SDs`
- Upper threshold: `Mean + 2 SDs`

It is important to use the correct mean and SD for this. That is, for example, thresholds for congruent trials need to use the congruent mean and the congruent SD. We used the following formulas in cells `U25` and `U26`, respectively:

- Lower congruent threshold: `=U17-2*U21`
- Upper congruent threshold: `=U17+2*U21`

Step 7: Calculate condition-specific mean RTs (after outlier removal)

Aim: Calculate condition-specific mean RTs after removing outliers based on standard deviations (SDs).

Why do we do this?: The result of this step is what we will input into SPSS to find out if our experimental conditions had an influence on RTs.

In our partially worked example, we've added two more columns for this, `conRTFinal` and `inconRTFinal`.

We then added the following formula for congruent trials to cell `P2` and copied it downwards:

```
=IF(AND(N2>$U$25, N2<$U$26), N2, "")
```

In words:

- If the RT is above the lower threshold (`N2>$U$25`) *and*
  - Note that we use an absolute cell reference here to make sure the reference to cell `U25` does not change!
- ...if the RT is below the upper threshold (`N2<$U$26`)

- ...then copy the RT to the new cell.

Based on the information in `conRTFinal`, we can then calculate the final mean RT for congruent trials in cell `U32`:

```
=ROUND(AVERAGE(P2:P73),0)
```

Note that this mean fulfils the following criteria:

- It is not influenced by extreme RTs.
- It is not influenced by outlier RTs.
- It is not influenced by RTs in incorrect trials.

We also counted how many congruent trials were removed as outliers in cell `U35`:

```
=COUNTIF(N2:N73, "<"&U25) + COUNTIF(N2:N73, ">"&U26)
```

In words:

- Count all RTs below the threshold `=COUNTIF(N2:N73, "<"&U25)`.
  - `<` means less than.
  - `&` means combine ("concatenate") what comes before and after the `&`.
  - `U25` is the lower threshold.
- Count all RTs above the threshold.
- Add both counts (`+`).

Calculate medians

In addition, we are going to calculate the medians. The median is insensitive to outliers. Therefore, we don't need to reject outliers (and we therefore used column `N` in the partially worked example). We calculated the median in cell `U39`:

The formula for congruent trials: `=ROUND(MEDIAN(N2:N73), 0)`

In words:

- Calculate the median for RTs in the range `N2:N73`.
- Round to 0 decimal places.

Comparison of means with and without outlier removal and medians

Let's compare the different values:

Condition	Mean (in ms) before outlier removal	Mean (in ms) after outlier removal	Median (in ms)
Congruent RT	484	475	456

Please note that mean before outlier removal $>$ mean after outlier removal $>$ median. This is a typical finding. RT distributions tend to be positively skewed (i.e., skewed towards larger values). The median is insensitive to these values. The mean after outlier removal will be influenced by some of the slower RTs (the ones that were not removed as outliers), and thus it will typically be greater than the median. The mean without outlier removal is influenced by all data points, so it will typically have the largest value.

Evaluation

After completing these steps we have the mean RTs and accuracies for one participant. In terms of the data file for SPSS, we have completed one row. Clearly this is not the most efficient way to do this. It is slow and writing formulas in Excel tends to be non-intuitive thanks to the awkward Excel syntax. We could have used R, but at present the School has asked us not to use R (which is a bit ironic given that I have written the HHG using R Markdown). Alternatively, we could have used a Python tool I have written, but that requires the installation of additional software.

On the plus side, as mentioned previously, knowing how to use Excel formulas *is* a really useful skill! Please note though that you will not need to remember these formulas for any of the upcoming assessments. There will also be no time limit for the upcoming assessments involving Excel. Therefore, you will have sufficient time to look these formulas up.

Data preprocessing with R

Optional self-study

In this chapter, we will show you how to apply the same preprocessing steps in R. You do not need to work through the R code if you're not interested in how R works. We only show this for the students who are. If you'd like to see the R code, click on the "Code" drop-down buttons. Note that the results are identical to those obtained with Excel.

```
library(tidyverse)

# read csv file
data <- read_csv("./assets/letter_flanker.csv")

# remove practice trials and create ms RT column
data <- data %>%
  filter(trials.thisN >= 0) %>%  # only keep main trials loop
  mutate(rt_ms = response.rt * 1000) # create new column with RTs in ms

# calculate overall accuracy (including extreme trials)
overall_acc = summarise(data, (sum(response.corr == 1) / (sum(response.corr == 1 | res
overall_acc = unlist(round(overall_acc * 100, digits = 1))  # multiply by 100, round t

# filter out extremes
data_no_extr <- data %>%
  filter(rt_ms >= 150, rt_ms <= 3000)

# calculate percentage of extreme trials
nr_extr = nrow(data) - nrow(data_no_extr)  # nr of rows in data minus nr of rows in da
percent_extr = round((nr_extr/nrow(data)) * 100, digits = 1)

# calculate accuracies per condition (including outliers)
condition_acc <- data_no_extr %>%
  group_by(congruency) %>%
  summarise(accuracy = (sum(response.corr == 1)/(sum(response.corr == 1 | response.cor
  mutate(accuracy = round(accuracy * 100, digits = 1))

# keep only correct trials
data_no_extr_only_corr <- data_no_extr %>%
  filter(response.corr == 1)

# calculate mean RTs per condition without outlier removal and median RTs
mean_rts_with_outl <- data_no_extr_only_corr %>%
  group_by(congruency) %>%
```

```
  summarise(mean_w_outl = round(mean(rt_ms, na.rm=TRUE), digits = 0),
            median = round(median(rt_ms, na.rm=TRUE), digits = 0)
  )

# calculate thresholds for outlier removal
outl_thresh <- data_no_extr_only_corr %>%
  group_by(congruency) %>%
  summarise(lthresh = mean(rt_ms, na.rm=TRUE) - (2. * sd(rt_ms, na.rm=TRUE)),
            uthresh = mean(rt_ms, na.rm=TRUE) + (2. * sd(rt_ms, na.rm=TRUE))
  )

# add the outlier removal thresholds to the tibble
# con trials will get con thresholds, incon trials incon thresholds
# this is done to allow the filtering operation in the next step
#old: data_no_extr_only_corr_with_thr <- full_join(data_no_extr_only_corr, outl_thresh
data_no_extr_only_corr_with_thr <- full_join(data_no_extr_only_corr, outl_thresh)

# remove outliers using the thresholds
data_no_extr_only_corr_no_outl <- data_no_extr_only_corr_with_thr %>%
  filter(rt_ms >= lthresh, rt_ms <= uthresh) # keep trials between +/- 2 SDs

# calculate mean RTs per condition with outlier removal
mean_rts_no_outl <- data_no_extr_only_corr_no_outl %>%
  group_by(congruency) %>%
  summarise(mean_no_outl = round(mean(rt_ms, na.rm=TRUE), digits = 0))

# combine results not distinguishing between con and incon trials
resultsOverall <- as_tibble(cbind(overall_acc, percent_extr))
print(resultsOverall)
```

```
# A tibble: 1 x 2
  overall_acc percent_extr
        <dbl>        <dbl>
1        93.1          2.8
```

```
# combine results into one tibble
results <- list(condition_acc, mean_rts_with_outl, mean_rts_no_outl) %>%
  reduce(left_join) %>%  # join list of tibbles
  relocate(median, .after = last_col())  # move median column to the right
print(results)
```

```
# A tibble: 2 x 5
  congruency accuracy mean_w_outl mean_no_outl median
  <chr>         <dbl>       <dbl>        <dbl>  <dbl>
1 con            97.2         484          475    456
2 incon          88.2         572          531    504
```

One of the advantages of using R is that it has a powerful visualisation package called ggplot2. Using ggplot2, the content and layout of plots can be controlled using code. This is far

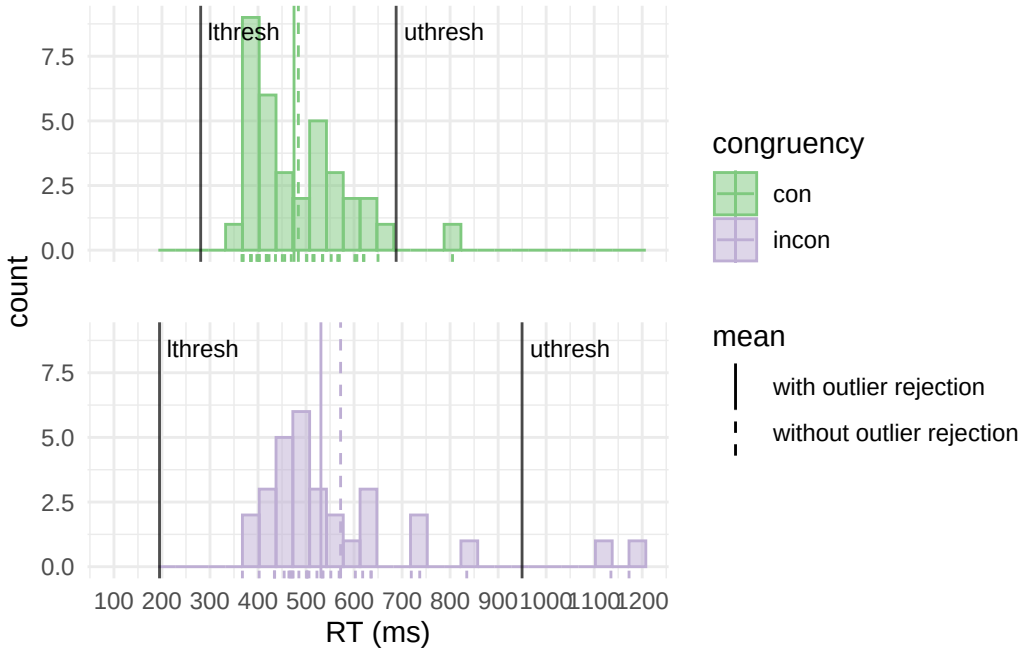
superior to visualisations in Excel, which typically have very limited options and frequently require manual interventions.

Let's plot our RT distributions using ggplot2. Note how the visualisations make it intuitively clear how the RTs are distributed, where the lower and upper outlier rejection thresholds are and how outlier rejection affects the condition means.

```
library(ggplot2)
library(tidyverse)

# rearrange the thresholds tibble (necessary for plotting with geom_vline)
outl_thresh2 <- outl_thresh %>%
  pivot_longer(
    cols = c("lthresh", "uthresh"),
    names_to = "threshType",
    values_to = "thresh"
  )

# plot the RT distributions
ggplot() +
  geom_histogram(data = data_no_extr_only_corr, aes(x = rt_ms, fill = congruency, colo
                 position = "identity", alpha = 0.5, binwidth = 35) +  # add histogram
  scale_color_brewer(palette="Accent") +  # change outline colours
  scale_fill_brewer(palette="Accent") +   # change fill colours
  geom_rug(data = data_no_extr_only_corr, aes(x = rt_ms, colour = congruency)) +  # ad
  geom_vline(data = mean_rts_with_outl, aes(xintercept = mean_w_outl, colour = congrue
                                            linetype = "without outlier rejection")) +
  geom_vline(data = mean_rts_no_outl, aes(xintercept = mean_no_outl, colour = congruen
                                          linetype = "with outlier rejection")) +  # a
  scale_linetype_manual(name = "mean", values = c("with outlier rejection" = "solid",
                                                  "without outlier rejection" = "dashe
  geom_vline(data = outl_thresh2, aes(xintercept = thresh), colour = "black", alpha =
  geom_text(data = outl_thresh2, aes(x = thresh, label = threshType),
            y = Inf, vjust = 2, hjust = -0.1, size = 3) +  # label outlier thresholds
  facet_wrap(vars(congruency), ncol = 1) +  # plot separate graphs for con and incon
  coord_cartesian(xlim = c(100, 1200)) +  # see https://github.com/tidyverse/ggplot2/i
  scale_x_continuous(name="RT (ms)", breaks = seq(from = 100, to = 1200, by = 100)) +
  theme_minimal() +
  theme(strip.text = element_blank(), panel.spacing.y = unit(.8, "cm"))  # remove head
```



Let's see what happens if we use the median absolute deviation (MAD) for outlier rejection.

```
library(tidyverse)

# read csv file
data <- read_csv("./assets/letter_flanker.csv")

# remove practice trials and create ms RT column
data <- data %>%
  filter(trials.thisN >= 0) %>%  # only keep main trials loop
  mutate(rt_ms = response.rt * 1000) # create new column with RTs in ms

# calculate overall accuracy (including extreme trials)
overall_acc = summarise(data, (sum(response.corr == 1) / (sum(response.corr == 1 | res
overall_acc = unlist(round(overall_acc * 100, digits = 1))  # multiply by 100, round t

# filter out extremes
data_no_extr <- data %>%
  filter(rt_ms >= 150, rt_ms <= 3000)

# calculate percentage of extreme trials
nr_extr = nrow(data) - nrow(data_no_extr)  # nr of rows in data minus nr of rows in da
percent_extr = round((nr_extr/nrow(data)) * 100, digits = 1)

# calculate accuracies per condition (including outliers)
condition_acc <- data_no_extr %>%
  group_by(congruency) %>%
  summarise(accuracy = (sum(response.corr == 1)/(sum(response.corr == 1 | response.cor
  mutate(accuracy = round(accuracy * 100, digits = 1))
```

```r
# keep only correct trials
data_no_extr_only_corr <- data_no_extr %>%
  filter(response.corr == 1)

# calculate mean RTs per condition without outlier removal and median RTs
mean_rts_with_outl <- data_no_extr_only_corr %>%
  group_by(congruency) %>%
  summarise(mean_w_outl = round(mean(rt_ms, na.rm=TRUE), digits = 0),
            median = round(median(rt_ms, na.rm=TRUE), digits = 0)
  )

# calculate thresholds for outlier removal
outl_thresh <- data_no_extr_only_corr %>%
  group_by(congruency) %>%
  summarise(lthresh = mean(rt_ms, na.rm=TRUE) - (2.5 * mad(rt_ms, na.rm=TRUE)),  # see
            uthresh = mean(rt_ms, na.rm=TRUE) + (2.5 * mad(rt_ms, na.rm=TRUE))
  )

# add the outlier removal thresholds to the tibble
# con trials will get con thresholds, incon trials incon thresholds
# this is done to allow the filtering operation in the next step
data_no_extr_only_corr_with_thr <- full_join(data_no_extr_only_corr, outl_thresh)

# remove outliers using the thresholds
data_no_extr_only_corr_no_outl <- data_no_extr_only_corr_with_thr %>%
  filter(rt_ms >= lthresh, rt_ms <= uthresh) # keep trials between +/- 2 SDs

# calculate mean RTs per condition with outlier removal
mean_rts_no_outl <- data_no_extr_only_corr_no_outl %>%
  group_by(congruency) %>%
  summarise(mean_no_outl = round(mean(rt_ms, na.rm=TRUE), digits = 0))

# combine results not distinguishing between con and incon trials
resultsOverall <- as_tibble(cbind(overall_acc, percent_extr))
print(resultsOverall)
```

```
# A tibble: 1 x 2
  overall_acc percent_extr
        <dbl>        <dbl>
1        93.1          2.8
```

```r
# combine results into one tibble
results <- list(condition_acc, mean_rts_with_outl, mean_rts_no_outl) %>%
  reduce(left_join) %>%  # join list of tibbles
  relocate(median, .after = last_col())  # move median column to the right
print(results)
```

```
# A tibble: 2 x 5
  congruency accuracy mean_w_outl mean_no_outl median
  <chr>         <dbl>       <dbl>        <dbl>  <dbl>
```

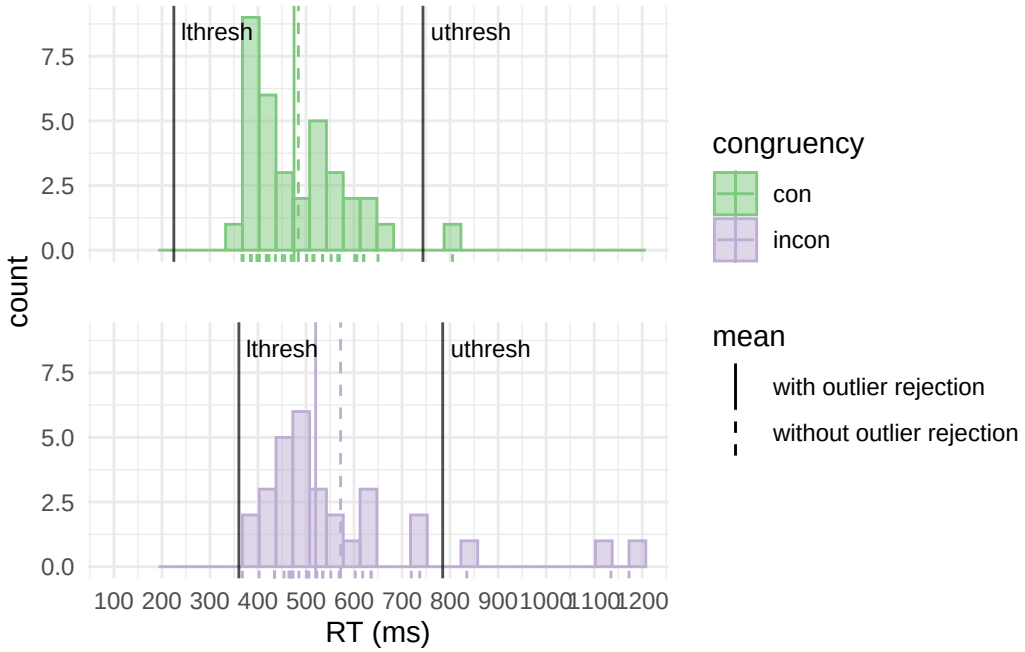
```
1 con            97.2         484          475    456
2 incon          88.2         572          520    504
```

Note that the thresholds for incongruent trials are now much closer together and an additional trial is considered an outlier. This illustrates a problem with the SD-based approach: The outliers themselves increase the SD. Thus outliers might not be detected because there might be even more pronounced outliers that mask them. The MAD is not sensitive to outliers, thus does not suffer from this problem.

```
library(ggplot2)

# rearrange the thresholds tibble (necessary for plotting with geom_vline)
outl_thresh2 <- outl_thresh %>%
  pivot_longer(
    cols = c("lthresh", "uthresh"),
    names_to = "threshType",
    values_to = "thresh"
  )

# plot the RT distributions
ggplot() +
  geom_histogram(data = data_no_extr_only_corr, aes(x = rt_ms, fill = congruency, colo
                 position = "identity", alpha = 0.5, binwidth = 35) +  # add histogram
  scale_color_brewer(palette="Accent") +  # change outline colours
  scale_fill_brewer(palette="Accent") +   # change fill colours
  geom_rug(data = data_no_extr_only_corr, aes(x = rt_ms, colour = congruency)) +  # ad
  geom_vline(data = mean_rts_with_outl, aes(xintercept = mean_w_outl, colour = congrue
                                            linetype = "without outlier rejection")) +
  geom_vline(data = mean_rts_no_outl, aes(xintercept = mean_no_outl, colour = congruen
                                          linetype = "with outlier rejection")) +  # a
  scale_linetype_manual(name = "mean", values = c("with outlier rejection" = "solid",
                                                  "without outlier rejection" = "dashe
  geom_vline(data = outl_thresh2, aes(xintercept = thresh), colour = "black", alpha =
  geom_text(data = outl_thresh2, aes(x = thresh, label = threshType),
            y = Inf, vjust = 2, hjust = -0.1, size = 3) +  # label outlier thresholds
  facet_wrap(vars(congruency), ncol = 1) +  # plot separate graphs for con and incon
  coord_cartesian(xlim = c(100, 1200)) +  # see https://github.com/tidyverse/ggplot2/i
  scale_x_continuous(name="RT (ms)", breaks = seq(from = 100, to = 1200, by = 100)) +
  theme_minimal() +
  theme(strip.text = element_blank(), panel.spacing.y = unit(.8, "cm"))  # remove head
```


lthresh
uthresh
7.5
5.0
2.5
0.0
count
lthresh
uthresh
7.5
5.0
2.5
0.0
100 200 300 400 500 600 700 800 900 1000 1100 1200
RT (ms)
congruency
con
incon
mean
with outlier rejection
without outlier rejection

Explore, apply, reflect

Lab class

Your task is to complete the calculations for the incongruent trials. Note that the partially worked example already includes all the columns you need—you will only need to add the formulas. In addition, note that you don't need to reinvent the wheel—simply copy and paste the formulas from the congruent trials and adapt them for the incongruent trials. How to copy and paste Excel formulas was described in **?@sec-copy-formulas**.

Show/hide results for incongruent trials

These are the results for incongruent trials:

- Accuracy: 88.2%
- Mean before outlier removal: 572 ms
- SD: 189 ms
- Mean after outlier removal: 531 ms
- Median: 504 ms

References

Berger, A., & Kiefer, M. (2021). Comparison of different response time outlier exclusion methods: A simulation study. *Frontiers in Psychology*, *12*, e675558. https://doi.org/10.3389/fpsyg.2021.675558

Derrfuss, J., Danielmeier, C., Klein, T. A., Fischer, A. G., & Ullsperger, M. (2021). Unbiased post-error slowing in interference tasks: A confound and a simple solution. *Behavior Research Methods*, *54*(3), 1416–1427. https://doi.org/10.3758/s13428-021-01673-8

Lamme, V. A. F., & Roelfsema, P. R. (2000). The distinct modes of vision offered by feedforward and recurrent processing. *Trends in Neurosciences*, *23*(11), 571–579. https://doi.org/10.1016/s0166-2236(00)01657-x

Miller, J. (2023). Outlier exclusion procedures for reaction time analysis: The cures are generally worse than the disease. *Journal of Experimental Psychology: General*, *152*(11), 3189–3217. https://doi.org/10.1037/xge0001450

Smith, G. A., & Brewer, N. (1995). Slowness and age: Speed-accuracy mechanisms. *Psychology and Aging*, *10*(2), 238–247. https://doi.org/10.1037//0882-7974.10.2.238

Whelan, R. (2008). Effective analysis of reaction time data. *The Psychological Record*, *58*(3), 475–482. https://doi.org/10.1007/BF03395630