Datenvisualisierung 4

Multi-part plots and customisations

Daniela Palleschi

2024-01-31

Inhaltsverzeichnis

Lesungen 1

Learning objectives 1

Set-up 2

1 Dodged density plots 3

2 Positioning errorbar plots 7

3 Customisations 10

4 Aufgaben 14

Session Info 14

Lesungen

For further reading and practice on this topic, I suggest reading Section 11.5 (Communication: Themes) in Wickham et al. (2023), and Chapter 4 (Representing summary statistics) in Nordmann et al. (2022).

Learning objectives

In this section we will learn to

- how to build multi-part plots
- how to customise our plots to better communicate our data

Set-up

Packages

Today, we're loading our relevant tidyverse packages directly: dplyr and ggplot. These are the only To aid us in loading in our data, we're also loading the here package, and the janitor package which is useful for tidying up our data (e.g., the clean_names() function). To customise our plots we're also using the ggthemes and patchwork packages. The former helps us produce plots that are colour-blind friendly, while the latter allows us to print multiple plots together.

```
pacman::p_load(tidyverse,
               here,
               janitor,
               ggthemes,
               patchwork,
               gghalves
               )
```

Data

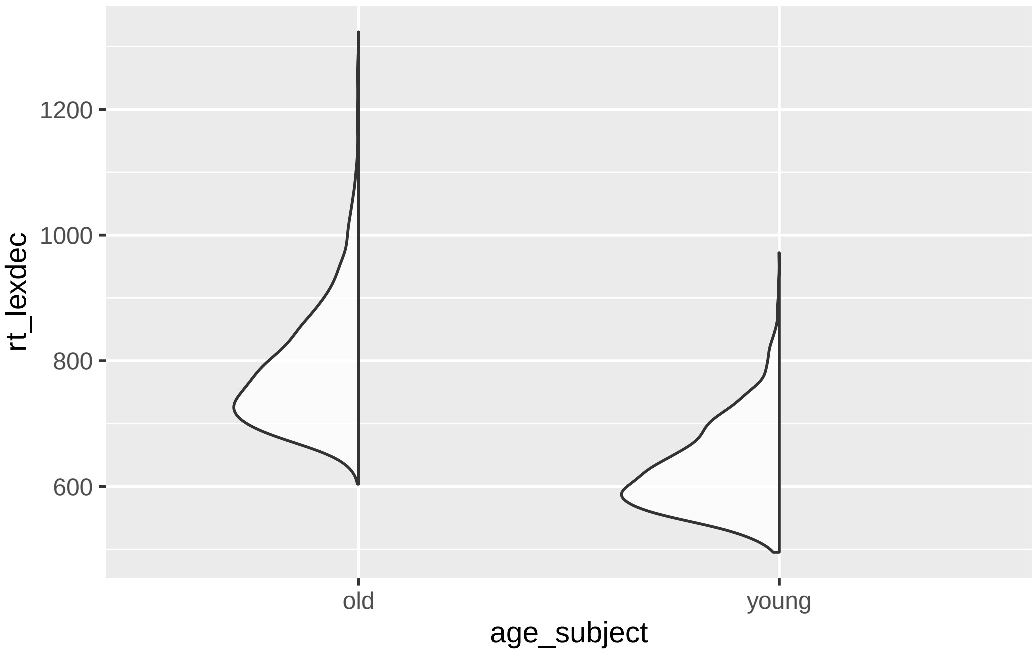
We're again working with our slightly altered version of the english dataset from the languageR package.

```
df_eng <- read_csv(
  here(
    "daten",
    "languageR_english.csv"
  )
) |>
  clean_names() |>
  rename(
    rt_lexdec = r_tlexdec,
    rt_naming = r_tnaming
  )
```

1 Dodged density plots

We can produce density plots mapped along a catgorical variable by using `geom_half_violin()` from the `gghalves` package.

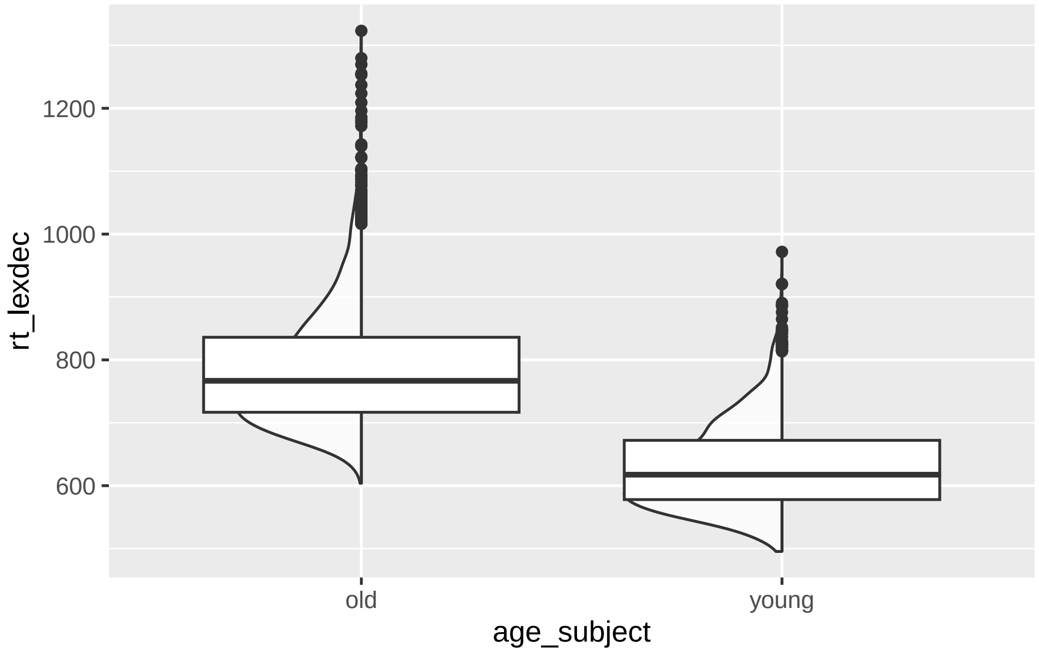
```
df_eng %>%
  ggplot() +
  aes(x = age_subject, y = rt_lexdec) +
  geom_half_violin(alpha = .8)
```



1.1 Adding a boxplot

We can also add another geom to add more information to the plot. Let's add a boxplot.

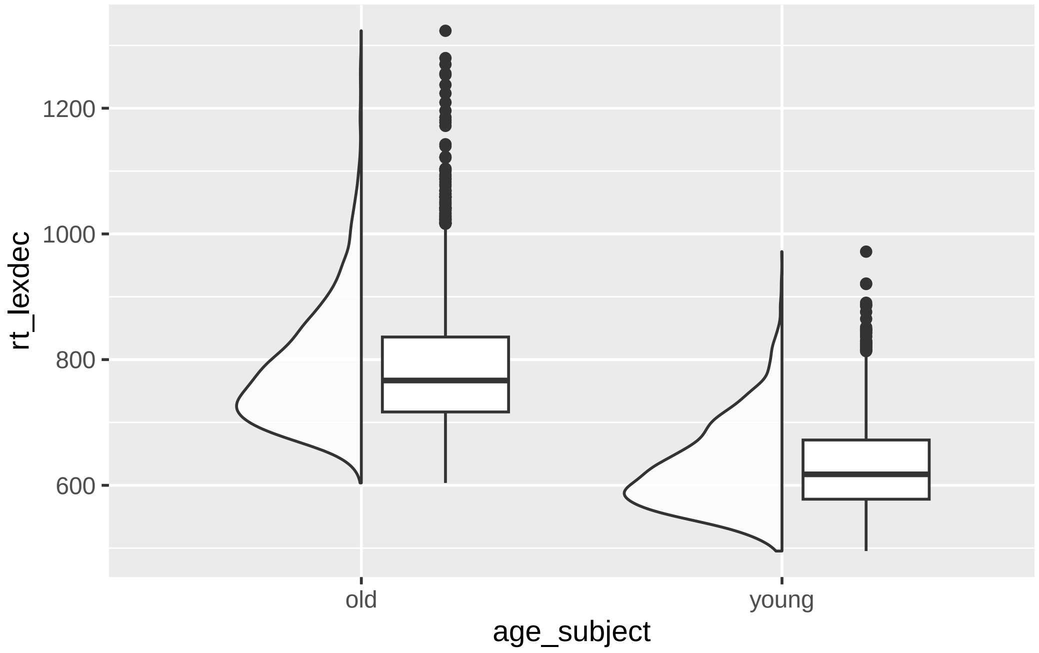
```
df_eng %>%
  ggplot() +
  aes(x = age_subject, y = rt_lexdec) +
  geom_half_violin(alpha = .8) +
  geom_boxplot()
```



1.2 `position_nudge()`

Maybe we want to move the boxplot so that it's not overtop of the density plots, and so that it's not quite as wide. We can do this by setting `position` to `position_nudge()`, and `width` to some value smaller than `.75`, which is the default width.

```r
df_eng %>%
  ggplot() +
  aes(x = age_subject, y = rt_lexdec) +
  geom_half_violin(alpha = .8) +
  geom_boxplot(width = .3, # make less wide
               position = position_nudge(x=0.2) # nudge 0.2 along the x-axis
               )
```



1.3 `position_jitter()` for scatterplots

This is from a family of options that allow us to alter the position of geoms. For example, Abbildung 1 A and B both show the exact same data, but Abbildung 1 B includes `position = position_jitter(0.2)` to move overlapping points. This way we get a good idea of how many observations there were across reaction times (y-axis).

```
df_eng |>
  ggplot() +
  aes(x = age_subject, y = rt_lexdec) +
  geom_point() +
  labs(title = "geom_point()") +
df_eng |>
  ggplot() +
  aes(x = age_subject, y = rt_lexdec) +
  geom_point(position = position_jitter(0.2),
             alpha = 0.2)+
  labs(title = "geom_point(position = position_jitter(0.2))") +

  plot_annotation(tag_levels = "A")
```

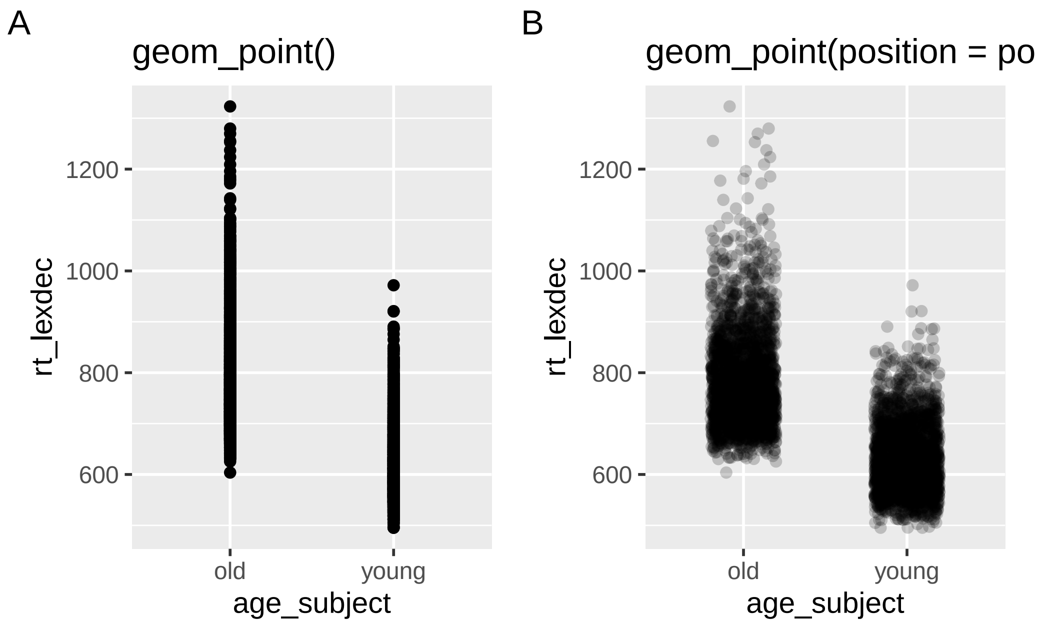


Abbildung 1: Plotting points along a categorical variable without (A) and with (B) position = position_jitter(0.2). Plot B also includes alpha = 0.2

1.4 Combining all three

If we put all of these plots together, we get a Abbildung 2.

```
fig_no_colour <-
  df_eng %>%
  ggplot() +
  aes(x = age_subject, y = rt_lexdec) +
  geom_point(position = position_jitter(0.2),
             alpha = 0.2) +
  geom_half_violin() +
  geom_boxplot(
    outlier.shape = NA,
                 width = .3,
                 position = position_nudge(x=0.2))
```

```
fig_no_colour
```

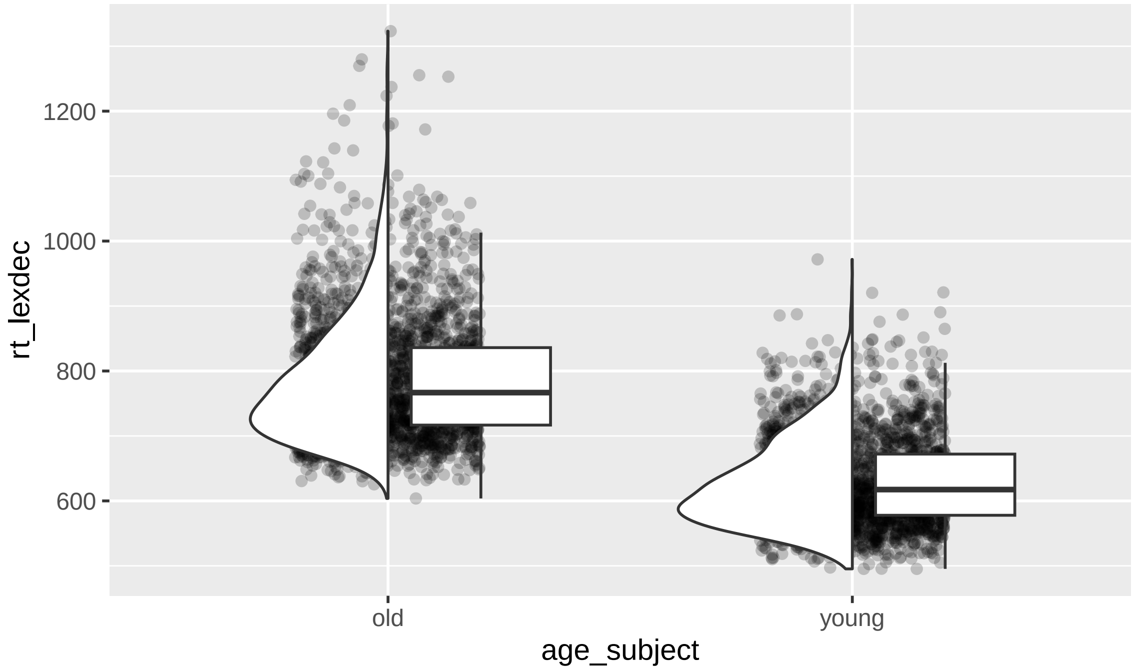


Abbildung 2: Violin plot: a mirrored density plot

2 Positioning errorbar plots

In the second report, you produced errorbar plots, but the errorbars were overlapping.

2.1 `pivot_longer() |> summarise()`

Let's reproduce something similar using the `english` dataset. First, we'll use `pivot_longer()` to lengthen our data, then we'll create a summary of reaction times for the lexical decision task and naming task per age group.

```r
sum_eng <-
  df_eng |>
  pivot_longer(
    cols = c(rt_lexdec, rt_naming),
    names_to = "task",
    values_to = "rt"
  ) |>
  summarise(
    mean = mean(rt, na.rm = T),
    sd = sd(rt, na.rm = T),
    .by = c(age_subject, task)
  ) |>
  mutate(age_subject = factor(age_subject, levels = c("young", "old")))
```

2.2 Overlapping errorbars

If we create an errorbar plot of this data, we get Abbildung 3.

```
sum_eng |>
  ggplot() +
  aes(x = age_subject, y = mean, colour = task, shape = task) +
  geom_point() +
  geom_errorbar(aes(ymin = mean-sd, ymax = mean+sd))
```

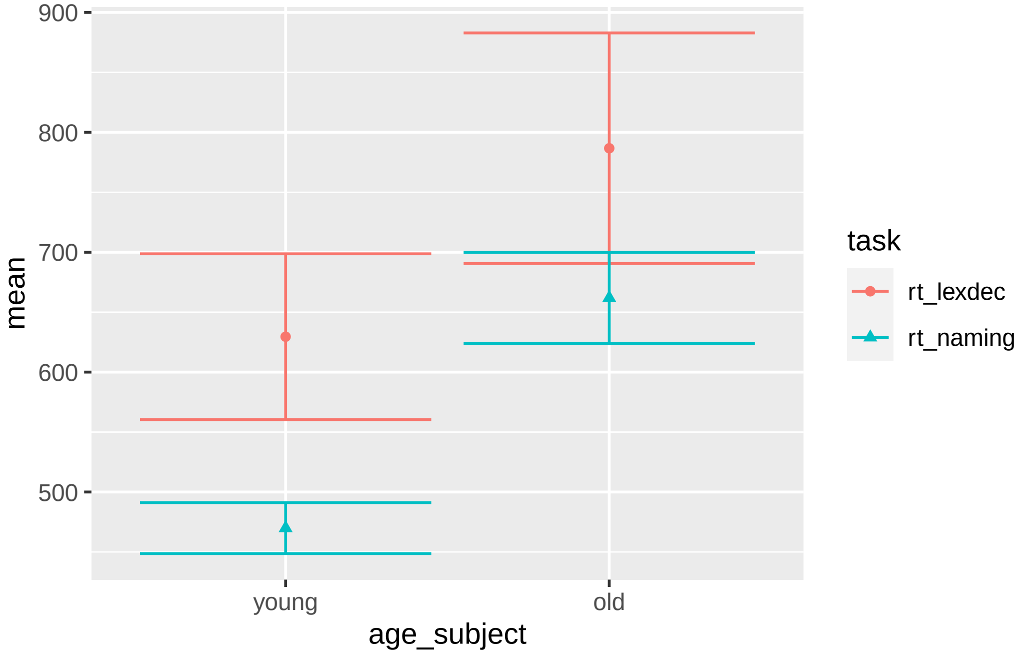


Abbildung 3: Overlapping errorbar plot

2.3 `position_dodge()`

We can add `position = position_dodge(0.2)` to force the errorbars to not overlap. We'll also adjust their `width` so they're not so wide (any value lower than 0.75).

```r
sum_eng |>
  ggplot() +
  aes(x = age_subject, y = mean, colour = task, shape = task) +
  geom_point() +
  geom_errorbar(aes(ymin = mean-sd, ymax = mean+sd),
                position = position_dodge(0.2),
                width = 0.2)
```

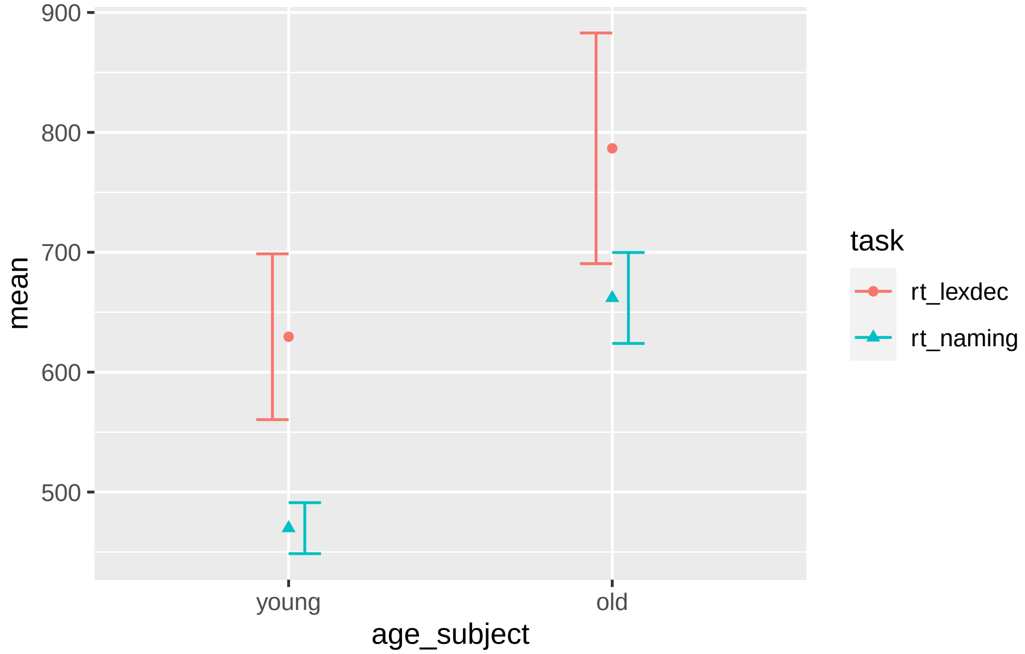


Abbildung 4: Overlapping errorbar plot

2.4 dodging all relevant geoms

But now we've left the points behind. We need to also dodge the points, so we add `position_dodge()` to `geom_point()`, making sure to use the same value as we did with `geom_errorbar()`.

```r
sum_eng |>
  ggplot() +
  aes(x = age_subject, y = mean, colour = task, shape = task) +
  geom_point(position = position_dodge(0.2)) +
  geom_errorbar(aes(ymin = mean-sd, ymax = mean+sd),
                position = position_dodge(0.2),
                width = 0.2)
```

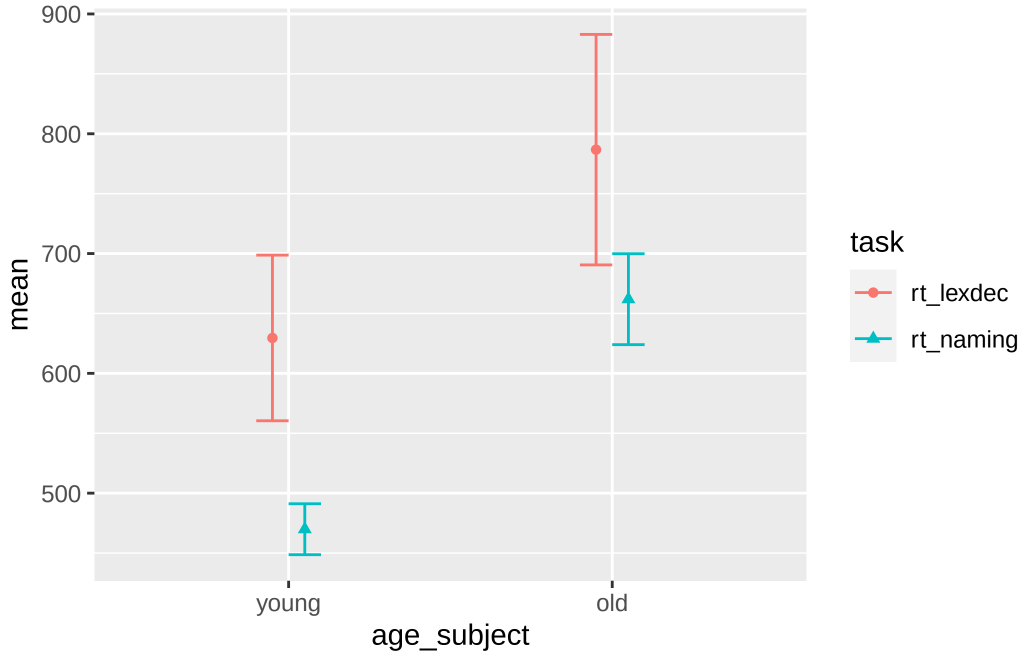


Abbildung 5: Overlapping errorbar plot

3 Customisations

What customisations do you see in the plots below?

```
fig_dens_colour <-
  df_eng %>%
  ggplot(aes(x = age_subject, y = rt_lexdec, )) +
  geom_point(
    color = "grey",
    position = position_jitter(0.2),
                 alpha = 0.2) +
  geom_half_violin(
    aes(fill = age_subject)) +
  geom_boxplot(
    outlier.shape = NA,
    aes(color = age_subject),
               width = .3,
               position = position_nudge(x=0.2)) +
  labs(title = "Violin plot",
       x = "Age group",
       y = "LDT reaction time (ms)",
    fill = "Age group") +
  scale_color_colorblind() +
  scale_fill_colorblind() +
  theme_minimal() +
  theme(legend.position = "none")

fig_point_colour <-
  df_eng %>%
  ggplot(aes(x = age_subject, y = rt_lexdec, )) +
  geom_point(
    aes(color = age_subject),
    position = position_jitter(0.2),
                 alpha = 0.2) +
  geom_half_violin() +
  geom_boxplot(
    outlier.shape = NA,
    # aes(color = age_subject),
               width = .3,
               position = position_nudge(x=0.2)) +
  labs(title = "Violin plot",
       x = "Age group",
       y = "LDT reaction time (ms)",
    fill = "Age group") +
  scale_color_colorblind() +
  scale_fill_colorblind() +
  theme_minimal() +
  theme(legend.position = "none")

fig_default <-
  sum_eng %>%
  ggplot(aes(x = age_subject, y = mean,
             colour = task, shape = task)) +
  geom_point() +
  geom_errorbar(aes(ymin=mean-sd,ymax=mean+sd))

fig_custom <-
sum_eng %>%
  mutate(task = fct_recode(task,
                           "LDT" = "rt_lexdec",
                           "Naming" = "rt_naming"),
  age_subject = fct_recode(age_subject,
                           "Young" = "young",
                           "Old" = "old")) |>
  ggplot(aes(x = age_subject, y = mean,
             colour = task, shape = task)) +
  geom_point(position = position_dodge(0.3),
             size = 3) +
  geom_errorbar(aes(ymin=mean-sd,ymax=mean+sd),
                position = position_dodge(0.3),
                width = .3) +
  geom_line(aes(group = task,
                linetype = task),
                position = position_dodge(0.3)) +
  theme_minimal() +
  labs(
    title = "Reaction times per group and task",
    x = "Age group",
    y = "Reaction time (ms)",
    colour = "Task",
    shape = "Task",
    linetype = "Task"
  ) +
  theme(axis.title = element_text(size = 12,
                                  face = "bold"),
        plot.title = element_text(size = 14),
        legend.title = element_text(face = "bold"))
```

3.1 Default themes

Firstly, `theme_minimal()` was added to each plot to customise the general look. There are a variety of custom themes to try, like `theme_bw()` or `theme_classic()`. Try them out.

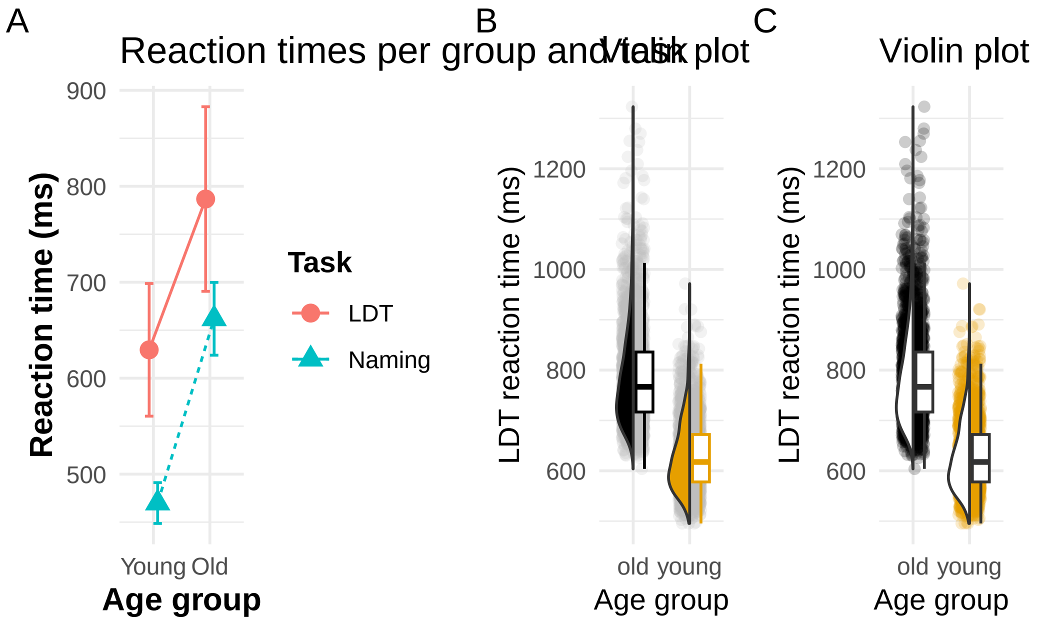
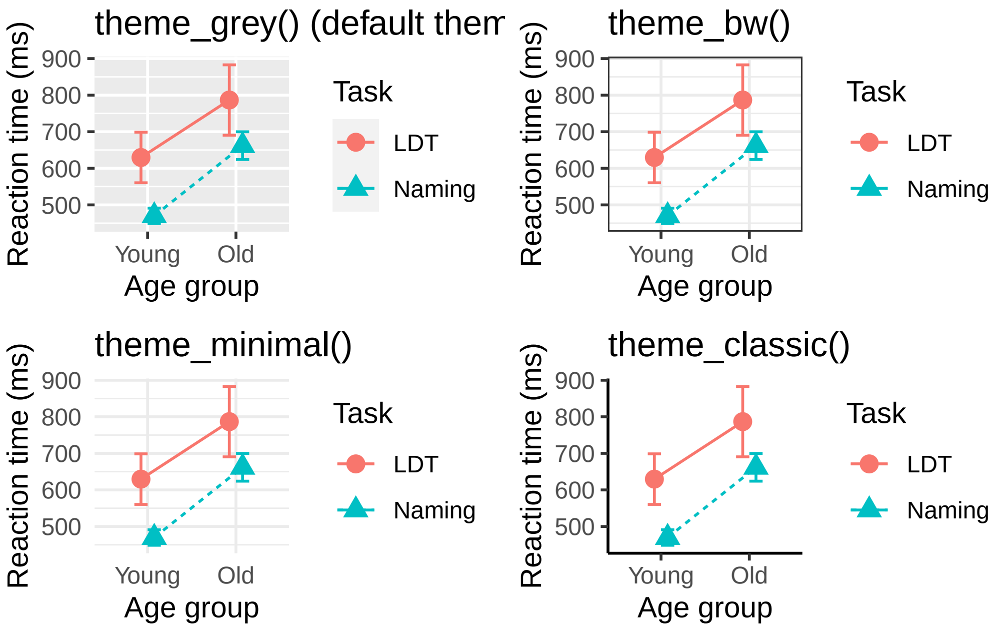


Abbildung 6: Customised plots to facilitation data communication.



3.2 theme()

We can also control individual components of theme by adding customisations with `theme()`. For example we see in Abbildung 6 A the axis titles are bolded. This was achieved by adding `theme(axis.title = element_text(face = "bold))`, where `axis.title =` indicates we want to make a change to the axis titles, `element_text()` indicates it's their text that we want to change, and `face = "bold"` indicates we want to make the text bold. The same was done for `legend.title =` to make the legend title bold.

Heutige Ziele

Heute haben wir gelernt, wie man…

- use `facet_wrap()` to plot more than three variables
- visualise summary statistics
- create multi-part plots

4 Aufgaben

1. Multi-part plot. Produce Abbildung 5 and Abbildung 2 for `rt_naming` (instead of `rt_lexdec`). Print the plots side-by-side using `patchwork`.
2. Customisations. Add customisations to the two plots by choosing a default theme, followed by `theme()` with adjustments for the axis titles, legend title, and plot title. You can change `face`, `size`, `family` (i.e., font).

Session Info

Hergestellt mit R version 4.3.0 (2023-04-21) (Already Tomorrow) und RStudioversion 2023.3.0.386 (Cherry Blossom).

```r
print(sessionInfo(),locale = F)
```

```
R version 4.3.0 (2023-04-21)
Platform: aarch64-apple-darwin20 (64-bit)
Running under: macOS Ventura 13.2.1

Matrix products: default
```

```
BLAS:   /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/lib/libRblas.0.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/lib/libRlapack.dylib;

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
 [1] magick_2.7.4    gghalves_0.1.4  patchwork_1.1.3 ggthemes_4.2.4
 [5] janitor_2.2.0   here_1.0.1      lubridate_1.9.2 forcats_1.0.0
 [9] stringr_1.5.0   dplyr_1.1.3     purrr_1.0.2     readr_2.1.4
[13] tidyr_1.3.0     tibble_3.2.1    ggplot2_3.4.3   tidyverse_2.0.0

loaded via a namespace (and not attached):
 [1] utf8_1.2.3       generics_0.1.3   stringi_1.7.12   hms_1.1.3
 [5] digest_0.6.33    magrittr_2.0.3   evaluate_0.21    grid_4.3.0
 [9] timechange_0.2.0 fastmap_1.1.1    rprojroot_2.0.3  jsonlite_1.8.7
[13] fansi_1.0.4      scales_1.2.1     cli_3.6.1        crayon_1.5.2
[17] rlang_1.1.1      bit64_4.0.5      munsell_0.5.0    withr_2.5.0
[21] yaml_2.3.7       parallel_4.3.0   tools_4.3.0      tzdb_0.4.0
[25] colorspace_2.1-0 pacman_0.5.1     vctrs_0.6.3      R6_2.5.1
[29] lifecycle_1.0.3  snakecase_0.11.0 bit_4.0.5        vroom_1.6.3
[33] pkgconfig_2.0.3  pillar_1.9.0     gtable_0.3.4     glue_1.6.2
[37] Rcpp_1.0.11      xfun_0.39        tidyselect_1.2.0 rstudioapi_0.14
[41] knitr_1.44       farver_2.1.1     htmltools_0.5.5  labeling_0.4.3
[45] rmarkdown_2.22   compiler_4.3.0
```

Literaturverzeichnis

Nordmann, E., McAleer, P., Toivo, W., Paterson, H., & DeBruine, L. M. (2022). Data Visualization Using R for Researchers Who Do Not Use R. *Advances in Methods and Practices in Psychological Science*, *5*(2), 251524592210746. https://doi.org/10.1177/25152459221074654

Wickham, H., Çetinkaya-Rundel, M., & Grolemund, G. (2023). *R for Data Science* (2. Aufl.).