Datenvisualisierung 4

Mehrteilige Plots und Anpassungen

Daniela Palleschi

2024-06-25

Inhaltsverzeichnis

Lesungen

Für weitere Lektüre und Übungen zu diesem Thema empfehle ich die Lektüre von Abschnitt 11.5 (Kommunikation: Themen) in Wickham et al. (2023) und Kapitel 4 (Darstellung von zusammenfassenden Statistiken) in Nordmann et al. (2022).

Lernziele

In diesem Abschnitt werden wir lernen

- mehrteilige Diagramme zu erstellen
- die Position von Geomen anzupassen
- unsere Diagramme für eine bessere Datenkommunikation anzupassen

Einrichten

Pakete

Heute laden wir unsere relevanten `tidyverse`-Pakete direkt: `dplyr` und `ggplot`. Dies sind die einzigen Pakete, die uns beim Laden unserer Daten helfen. Wir laden auch das `here`-Paket und das `janitor`-Paket, das für das Aufräumen unserer Daten nützlich ist (z.B. die Funktion `clean_names()`). Um unsere Diagramme anzupassen, verwenden wir auch die Pakete `ggthemes` und `patchwork`. Ersteres hilft uns bei der Erstellung von farbenblindenfreundlichen Plots, während letzteres uns erlaubt, mehrere Plots zusammen zu drucken. Wir brauchen auch ein neues Paket: `gghalves`.

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

Daten

Wir arbeiten wieder mit unserer leicht veränderten Version des `english`-Datensatzes aus dem Paket `languageR`.

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

1 Ausweichende Dichteplots

Wir können Dichteplots entlang einer katgorischen Variable erstellen, indem wir `geom_half_violin()` aus dem Paket `gghalves` verwenden.

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

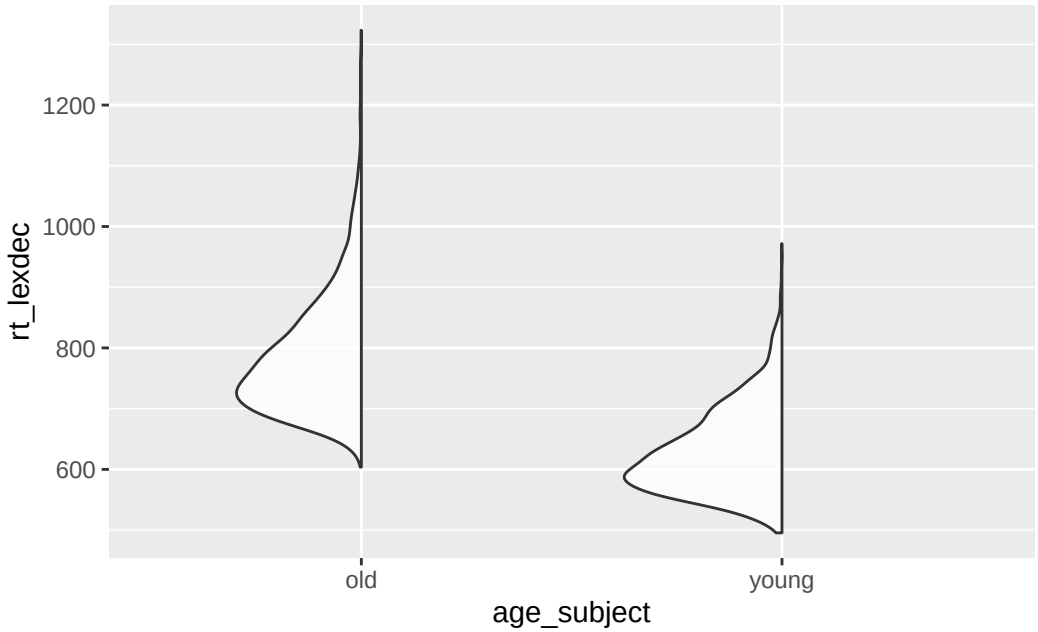

Abbildung 1: Dodged density plots with `gghalves::geom_half_violin()`

1.1 Hinzufügen eines Boxplots

Wir können auch ein weiteres Geom hinzufügen, um dem Diagramm weitere Informationen hinzuzufügen. Fügen wir einen Boxplot hinzu.

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

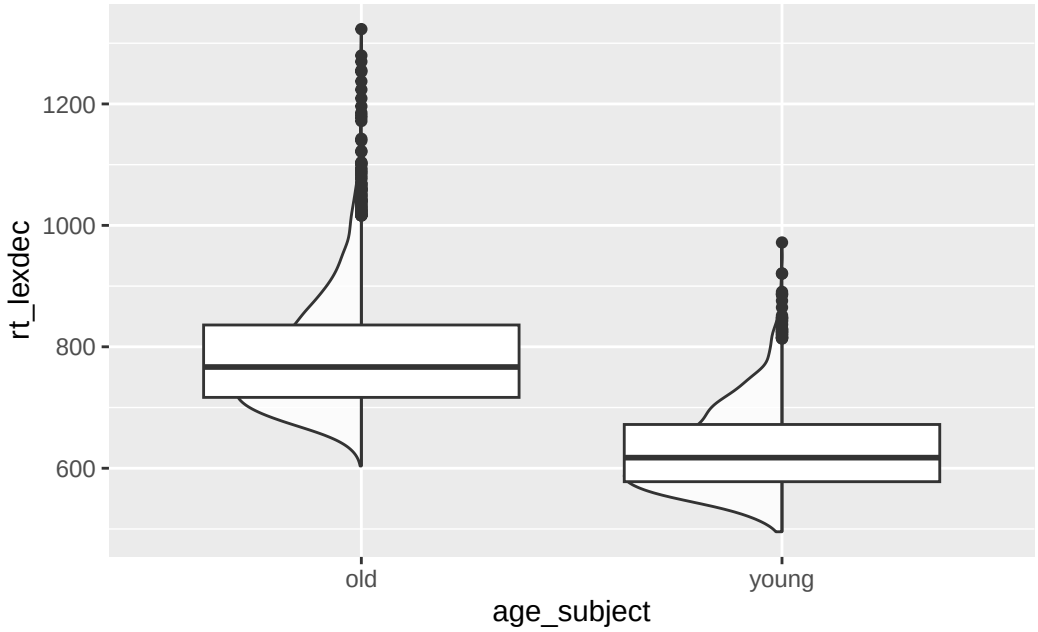

Abbildung 2: Boxplot on top of our dodged density plot

1.2 `position_nudge()`

Vielleicht wollen wir den Boxplot so verschieben, dass er nicht über den Dichteplots liegt und nicht ganz so breit ist. Wir können dies tun, indem wir `position` auf `position_nudge()` und `width` auf einen Wert kleiner als `.75` setzen, was die Standardbreite ist.

```
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)
               )
```

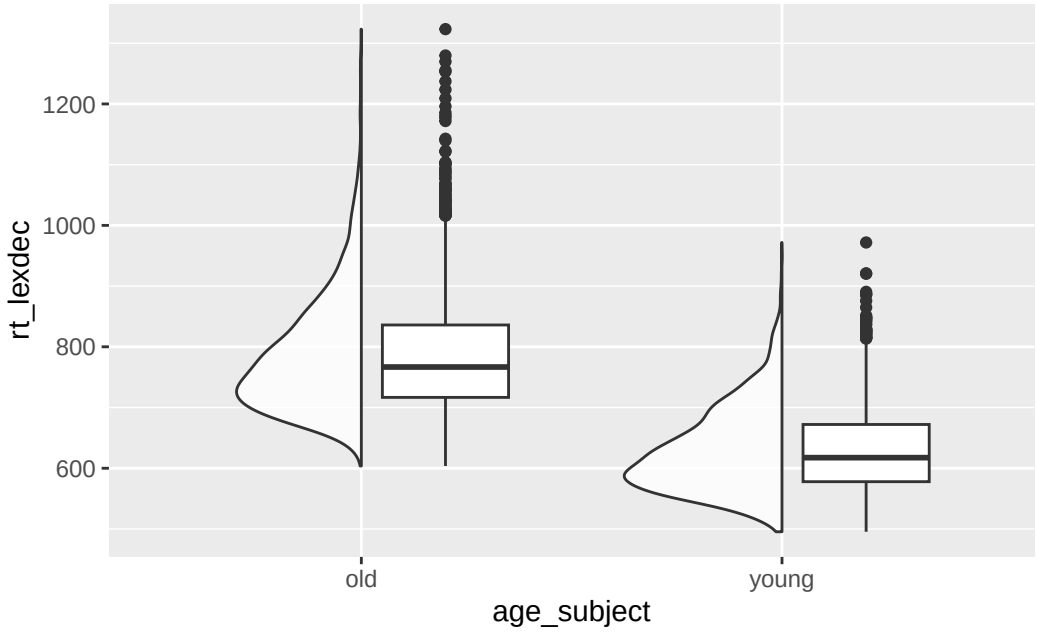

Abbildung 3: Boxplot on top of our dodged density plot

1.3 `position_jitter()` für Scatterplots

Dies gehört zu einer Familie von Optionen, mit denen man die Position von Geomen verändern kann. Zum Beispiel zeigen Abbildung 4 A und B beide genau die gleichen Daten, aber Abbildung 4 B enthält `position = position_jitter(0.2)`, um überlappende Punkte zu verschieben. Auf diese Weise erhalten wir eine gute Vorstellung davon, wie viele Beobachtungen es über die Reaktionszeiten hinweg gab (y-Achse).

```
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")
```

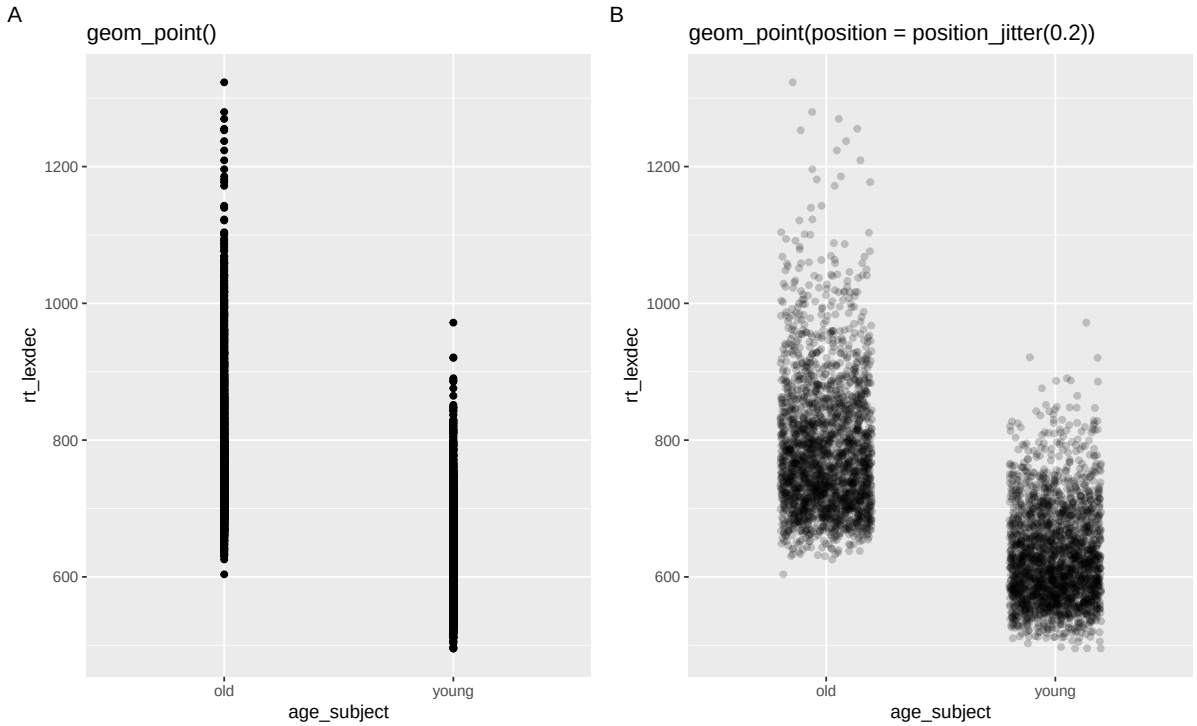

Abbildung 4: Plotting points along a categorical variable without (A) and with (B) position = position_jitter(0.2). Plot B also includes alpha = 0.2

1.4 Kombiniert alle drei

Wenn wir alle diese Diagramme zusammenfügen, erhalten wir eine Abbildung 5.

```
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))
```

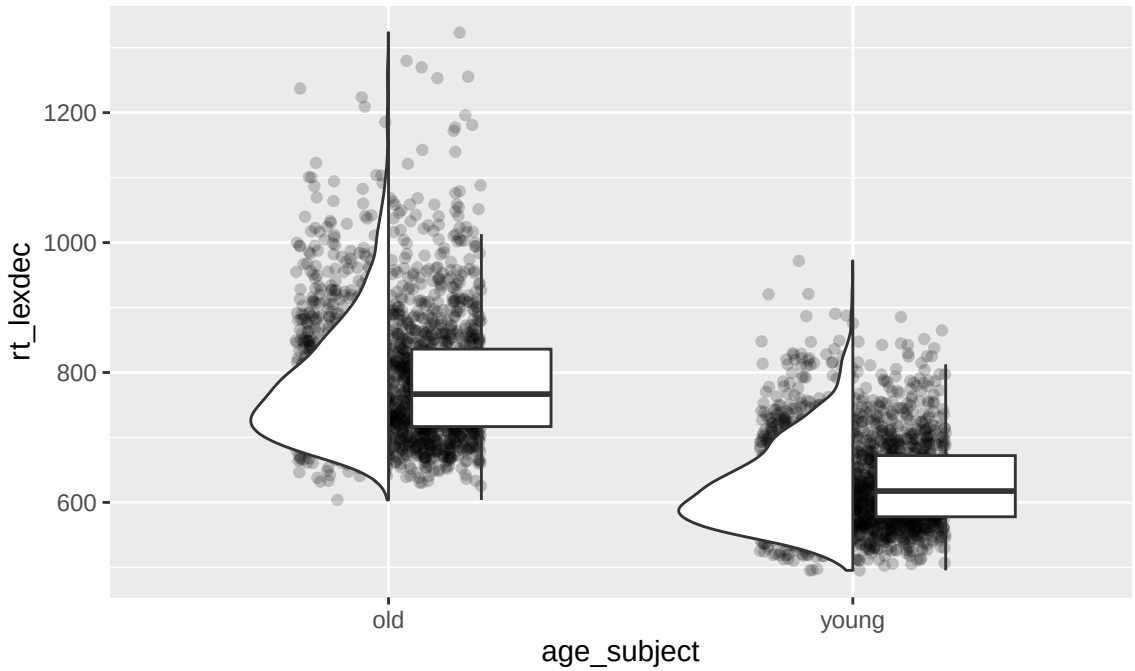

Abbildung 5: Density, boxplot, scatterplot

2 Positionierung von Fehlerbalkenplots

Im zweiten Bericht haben Sie Fehlerbalkenplots erstellt, aber die Fehlerbalken haben sich überschnitten.

2.1 `pivot_longer() |> summarise()`

Lassen Sie uns etwas Ähnliches mit dem Datensatz "Englisch" nachstellen. Zuerst werden wir `pivot_longer()` verwenden, um unsere Daten zu verlängern, dann erstellen wir eine Zusammenfassung der Reaktionszeiten für die lexikalische Entscheidungsaufgabe und die Benennungsaufgabe pro Altersgruppe.

```
sum_eng <-
  df_eng |>
  pivot_longer(
    cols = c(rt_lexdec, rt_naming),
    names_to = "task",
    values_to = "rt"
  ) |>
  summarise(
```

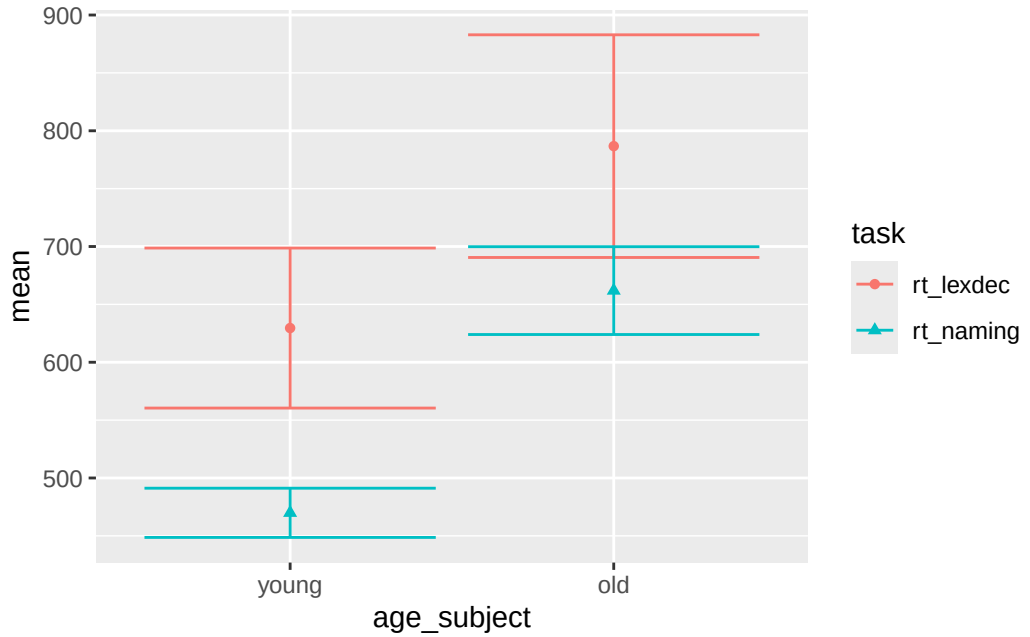

Abbildung 6: Overlapping errorbars

```
    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 Überlappende Fehlerbalken

Wenn wir für diese Daten ein Fehlerbalken-Diagramm erstellen, erhalten wir Abbildung 7.

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

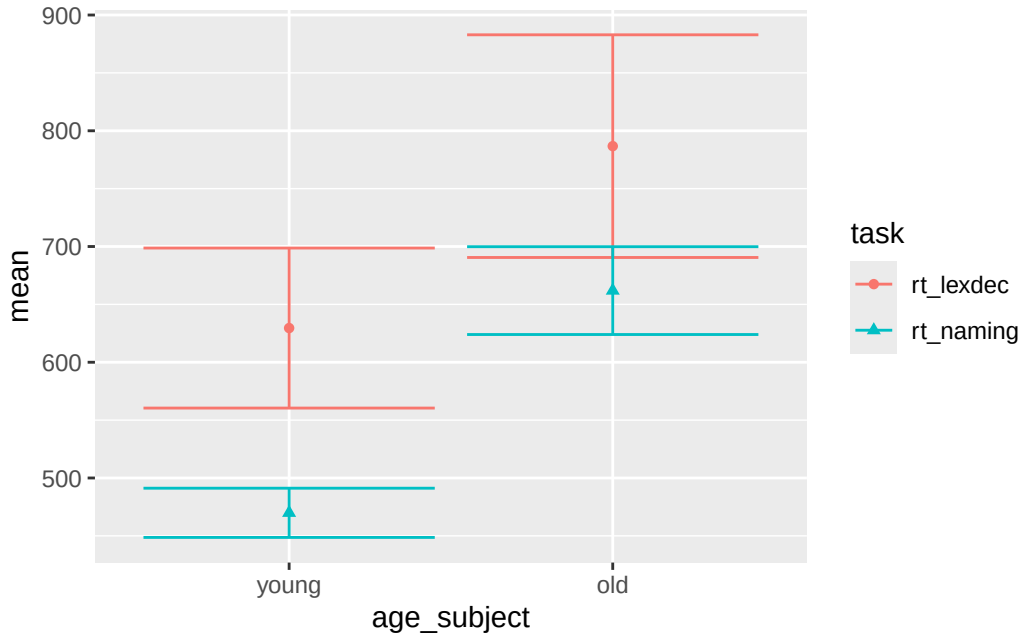

Abbildung 7: Overlapping errorbar plot

2.3 `position_dodge()`

Wir können `position = position_dodge(0.2)` hinzufügen, damit sich die Fehlerbalken nicht überlappen. Wir werden auch ihre `width` anpassen, damit sie nicht so breit sind (jeder Wert unter 0.75).

```
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)
```

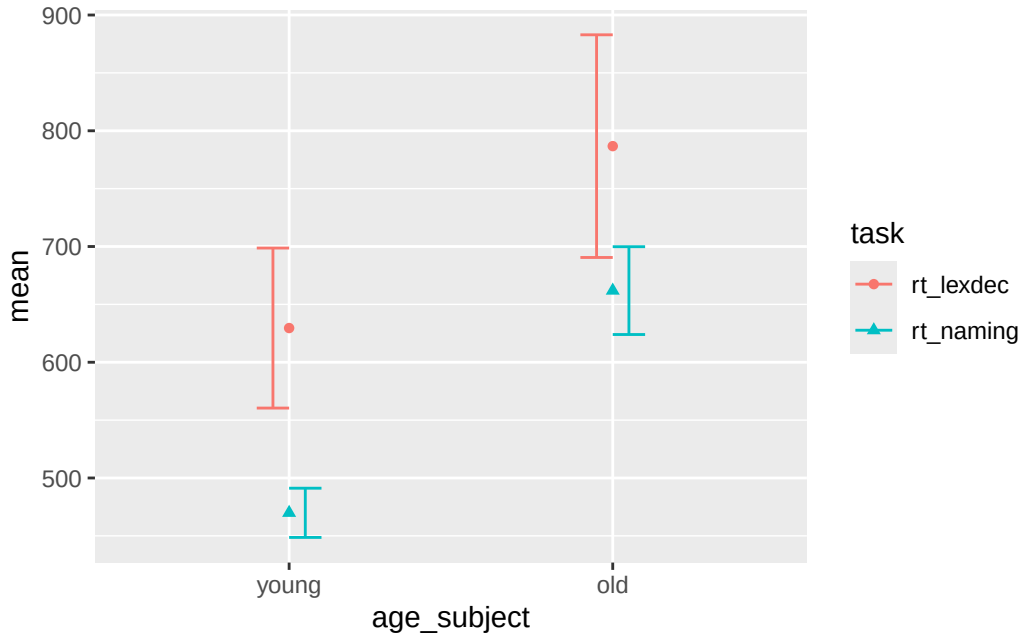

Abbildung 8: Overlapping errorbar plot

2.4 Ausweichen vor allen relevanten Geomen

Aber jetzt haben wir die Punkte hinter uns gelassen. Wir müssen auch den Punkten ausweichen, also fügen wir `position_dodge()` zu `geom_point()` hinzu und stellen sicher, dass wir den gleichen Wert wie bei `geom_errorbar()` verwenden.

```
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)
```

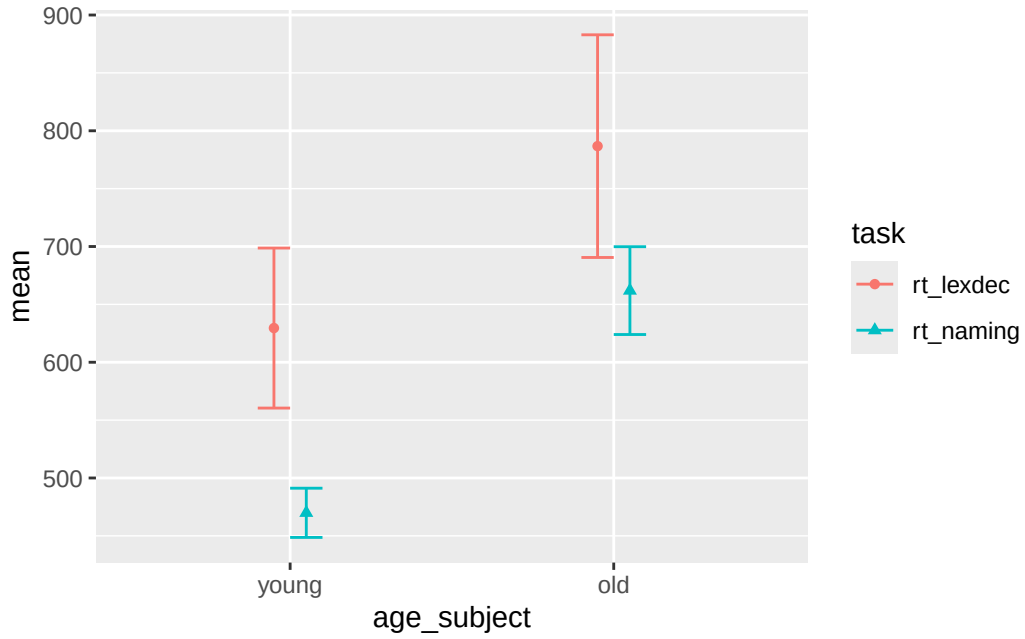

Abbildung 9: Overlapping errorbar plot

3 Anpassungen

Welche Anpassungen sehen Sie in den Diagrammen in Abbildung 10?

```
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 = "Distribution of reaction times",
       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 = "Distribution of reaction times",
       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 Standardthemen

Zunächst wurde `theme_minimal()` zu jedem Plot hinzugefügt, um das allgemeine Aussehen anzupassen. Es gibt eine Vielzahl von benutzerdefinierten Themen, wie `theme_bw()` oder `theme_classic()`. Probieren Sie sie aus.

3.2 theme()

Wir können auch einzelne Komponenten des Themas steuern, indem wir Anpassungen mit `theme()` hinzufügen. Zum Beispiel sehen wir in Abbildung 10 A, dass die Achsentitel fett gedruckt sind. Dies wurde durch Hinzufügen von `theme(axis.title = element_text(face = "bold))` erreicht, wobei `axis.title =` anzeigt, dass wir eine Änderung an den Achsentiteln vornehmen wollen, `element_text()` zeigt an, dass es ihr Text ist, den wir ändern wollen, und `face = "bold"` zeigt an, dass wir den Text fett machen wollen. Dasselbe wurde für `legend.title =` gemacht, um den Titel der Legende fett zu machen.

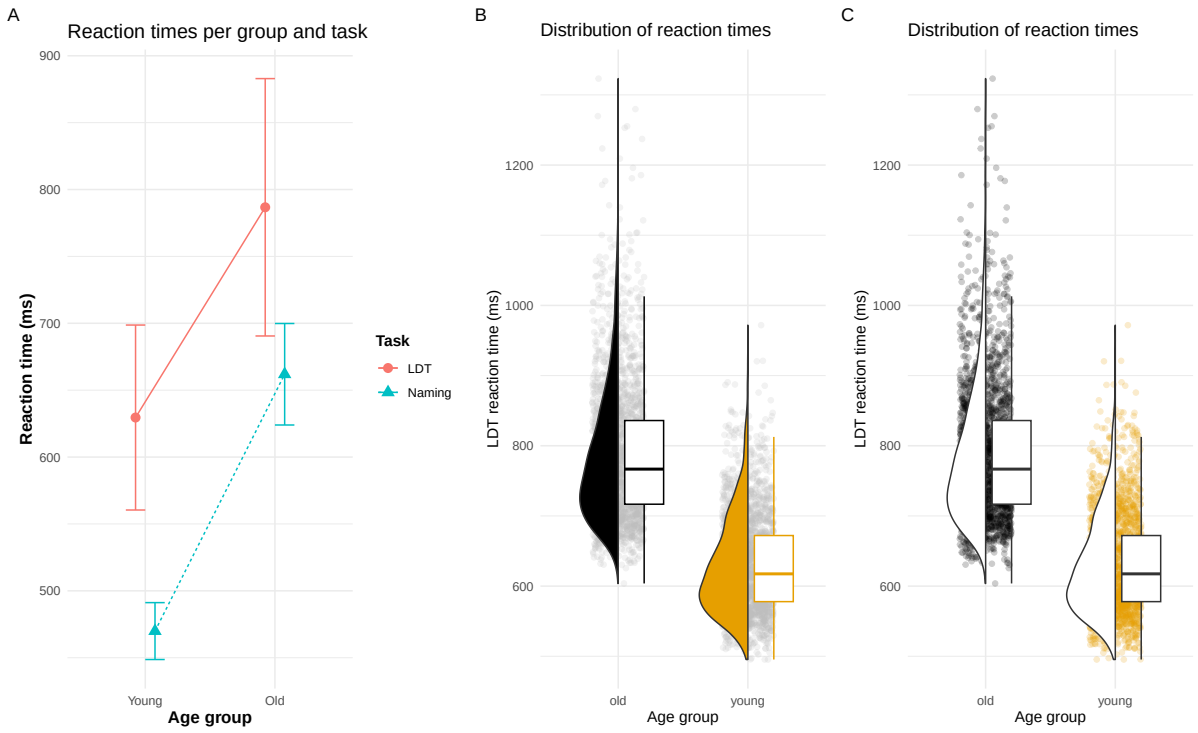

Abbildung 10: Customised plots to facilitation data communication.

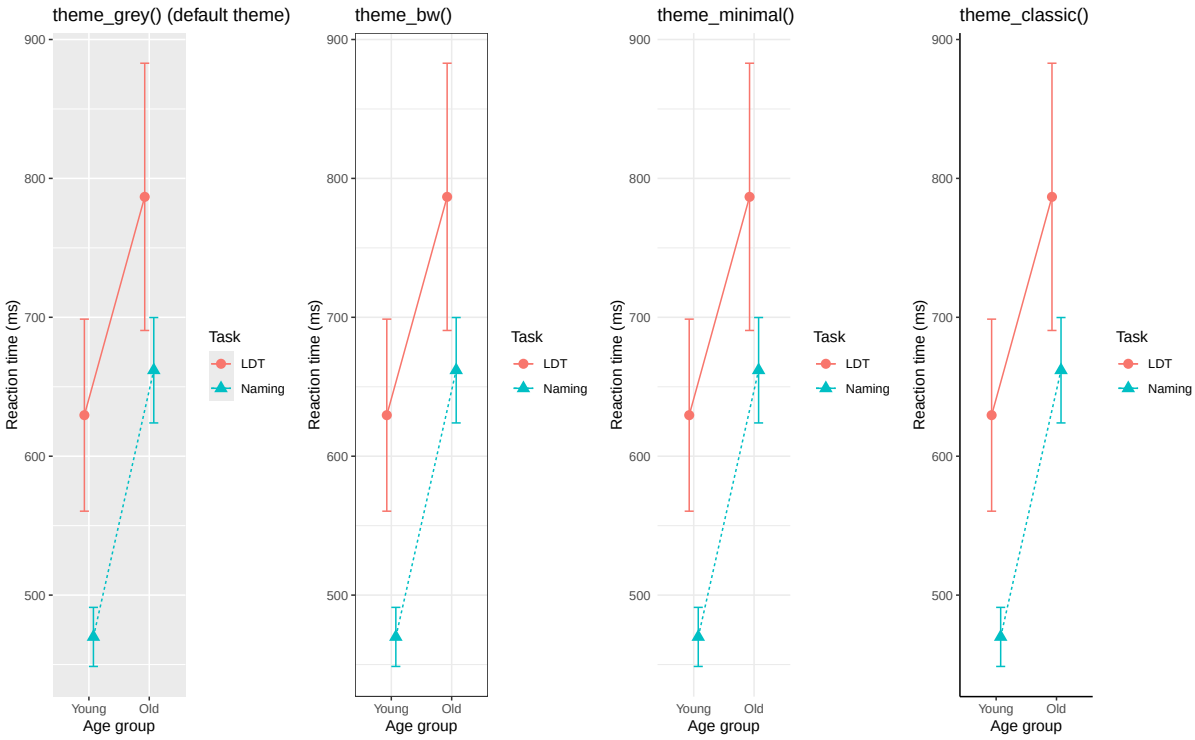

Abbildung 11: Preset themes

```
fig_no_colour + theme_minimal() +
  theme(
    axis.title = element_text(face = "italic")
    )
```

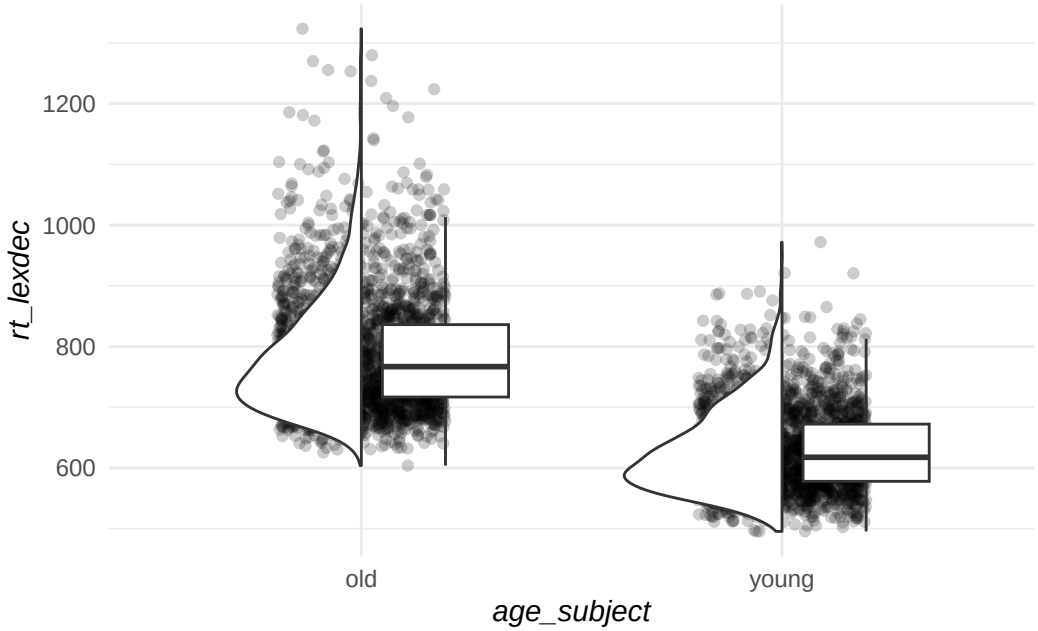

Abbildung 12: Using theme()

Heutige Ziele

Wir haben gelernt, wie man…

- mehrteilige Plots erstellen
- die Position von Geomen anzupassen
- unsere Plots für eine bessere Datenkommunikation anpassen

4 Aufgaben

1. Mehrteilige Darstellung. Erzeugen von Abbildung 9 und Abbildung 5 für `rt_naming` (anstelle von `rt_lexdec`). Drucken der Plots nebeneinander mit `patchwork`.

2. Beschriftungen. Benutzen Sie `labs()`, um Beschriftungen für den Titel, die x- und y-Achse und für die von Ihnen verwendete Ästhetik (Form, Farbe, etc.) hinzuzufügen, die in einer Legende resultieren. Dies sollte damit enden, dass der Titel der Legende auch einen eigenen Namen erhält.

3. Anpassungen. Fügen Sie den beiden Diagrammen Anpassungen hinzu, indem Sie ein Standardthema wählen, gefolgt von `theme()` mit Anpassungen für die Achsentitel, den Legendentitel und den Diagrammtitel. Sie können `face`, `size`, `colour`, `family` (d.h. Schriftart) ändern. Sie können `?theme` in der Konsole eingeben oder googeln, um einige Ideen zu bekommen. Wenn Sie sich nicht kreativ fühlen, versuchen Sie einfach, eine der Anpassungen zu replizieren, die Sie in Abbildung 10 sehen

Session Info

Hergestellt mit R version 4.4.0 (2024-04-24) (Puppy Cup) und RStudioversion 2023.3.0.386 (Cherry Blossom).

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

```
R version 4.4.0 (2024-04-24)
Platform: aarch64-apple-darwin20
Running under: macOS Ventura 13.2.1

Matrix products: default
BLAS:   /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libRblas.0.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libRlapack.dylib;  L

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

other attached packages:
 [1] magick_2.8.3    gghalves_0.1.4  patchwork_1.2.0 ggthemes_5.1.0
 [5] janitor_2.2.0   here_1.0.1      lubridate_1.9.3 forcats_1.0.0
 [9] stringr_1.5.1   dplyr_1.1.4     purrr_1.0.2     readr_2.1.5
[13] tidyr_1.3.1     tibble_3.2.1    ggplot2_3.5.1   tidyverse_2.0.0

loaded via a namespace (and not attached):
 [1] utf8_1.2.4        generics_0.1.3    renv_1.0.7        stringi_1.8.3
 [5] hms_1.1.3         digest_0.6.35     magrittr_2.0.3    evaluate_0.23
 [9] grid_4.4.0        timechange_0.3.0  fastmap_1.1.1     rprojroot_2.0.4
[13] jsonlite_1.8.8    tinytex_0.50      fansi_1.0.6       scales_1.3.0
```

```
[17] cli_3.6.2          crayon_1.5.2       rlang_1.1.3        bit64_4.0.5
[21] munsell_0.5.1      withr_3.0.0        yaml_2.3.8         parallel_4.4.0
[25] tools_4.4.0        tzdb_0.4.0         colorspace_2.1-0   pacman_0.5.1
[29] vctrs_0.6.5        R6_2.5.1           lifecycle_1.0.4    snakecase_0.11.1
[33] bit_4.0.5          vroom_1.6.5        pkgconfig_2.0.3    pillar_1.9.0
[37] gtable_0.3.5       glue_1.7.0         Rcpp_1.0.12        xfun_0.43
[41] tidyselect_1.2.1   rstudioapi_0.16.0  knitr_1.46         farver_2.1.1
[45] htmltools_0.5.8.1  labeling_0.4.3     rmarkdown_2.26     compiler_4.4.0
```

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.).