

Datenvisualisierung 2

Visualisierung von Beziehungen

Daniela Palleschi

Di. den 21.05.2024

Inhaltsverzeichnis

Lesungen	3
Einrichtung	3
Pakete	3
ggplot theme	4
Data	4
english dataset	4
Hypotheses	5
1 Datenvisualisierung	5
1.1 Visualisierung von Verteilungen	5
2 Visualisierung von Beziehungen	6
2.1 Gruppierte kontinuierliche Variable	6
2.1.1 Gestapelt	6
2.1.2 Dodged (Ausgewiche)	7
2.2 Zwei kontinuierliche Variablen	7
2.3 Hinzufügen weiterer Variablen	10
2.4 Facet grids	12
.	12
.	13
3 Bearbeitete Daten	14
4 Quarto Code Chunk Einstellungen	16
4.1 Verwendung von Code-Bausteinen	16

5	Plots speichern	17
5.1	ggsave()	17
5.1.1	ggsave() optionale Argumente	17
6	Hausaufgaben	18

Wiederholung

Letzte Woche haben wir gelernt...

- wie man Daten mit dem Paket **dplyr** aus dem **tidyverse** verarbeitet
- gelernt, wie man die **pipe** (**|>**) verwendet, um das Ergebnis einer Funktion in eine andere Funktion einzuspeisen
- über Funktionen, die auf Zeilen operieren
 - **filter()**, **arrange()**
- über Funktionen, die auf Spalten operieren
 - **rename()**, **mutate()**, **select()**, **relocate()**
- wie man **dplyr**-Funktionen mit Plots von **ggplot2** kombiniert

Wiederholung

- Was verbinden Pipes? **|>**
 - **dplyr** Verben/Funktionen
- Was verbinden Pluszeichen? **+**
 - **ggplot** Schichten

```
data |>
  mutate(new_variable = height_cm/age) |>
  filter(new_variable > 5) |>
  ggplot() +
  aes(x = age, y = height) +
  geom_point()
```

Lernziele

Heute werden wir lernen...

- wie man zwei oder mehr Variablen darstellt
 - mit Ästhetik und mit Facettenrastern
- wie man Codechunk-Optionen verwendet
- wie man Plots als Dateien speichert

Lesungen

- Kurs-Website: [Kap. 5 - Datenvisualisierung 2](#)
- [Kap. 2 \(Datenvisualisierung\)](#) aus [Abschnitt 2.5](#) in Wickham et al. (2023).
- [Ch. 3 \(Data visualtion\)](#) in Nordmann & DeBruine (2022).

Einrichtung

Pakete

```
library(tidyverse)
library(patchwork)
library(ggthemes)
library(languageR)
```

- tidyverse Familie von Paketen
 - ggplot2 für Diagramme
 - dplyr für die Datenverarbeitung
- ggthemes für farbenblindenfreundliche Farbpaletten
- patchwork für Plot-Layouts
- languageR für linguistische Datensätze

ggplot theme

- wir können unser `ggplot`-Thema global festlegen
 - alle Zahlen mit dem von uns gewählten Thema gedruckt werden
- Im Folgenden verwende ich `theme_bw()`
 - das unter anderem den Plot-Hintergrund weiß (anstelle von grau) macht

```
theme_set(theme_bw())
```

- dieser Schritt ist *optional*

Data

Wir verwenden den **english**-Datensatz aus dem Baayen & Shafaei-Bajestan (2019).

- enthält Daten aus einer lexikalischen Entscheidungsaufgabe in Englisch
- Die logarithmisch transformierten Reaktionszeiten werden zurücktransformiert, so dass sie in Millisekunden angegeben werden
 - Wir verwenden dazu die Funktion `exp()`

```
df_english <-  
  english |>  
  mutate(RTlexdec = exp(RTlexdec),  
         RTnaming = exp(RTnaming))
```

english dataset

Unsere Variablen von Interesse sind:

Tabelle 1: english dataset variables of interest

variable	description
RTlexdec	Reaktionszeiten für eine visuelle lexikalische Entscheidung (Millisekunden)
RTnaming	Reaktionszeiten für den Beginn einer verbalen Wortbenennungsaufgabe (Millisekunden)
WrittenFrequency	numerischer Vektor mit der logarithmischen Häufigkeit in der lexikalischen Datenbank von
Wort	ein Faktor mit 2284 Wörtern
AgeSubject	ein Faktor mit der Altersgruppe des Probanden als Level: jung versus alt
WordCategory	ein Faktor mit den Wortkategorien N (Substantiv) und V (Verb) als Ebenen

CV	Faktor, der angibt, ob das Anfangsphonem des Wortes ein Konsonant (C) oder ein Vokal
CorrectLexdec	numerischer Vektor mit dem Anteil der Probanden, die das Item bei der lexikalischen En

Hypothesen

- Welche Arten von Hypothesen könnten Sie für solche Daten aufstellen?
 - Unsere Reaktionszeitdaten (`RTnaming`, `RTlexdec`) sind unsere *Messvariablen*.
 - * d.h. was wir messen
 - Alle anderen Variablen sind mögliche *Vorhersagevariablen* (*Prädiktoren*)
 - * d.h. wir könnten vorhersagen, dass ihr Wert unsere Messvariablen beeinflussen würde
- Welche Auswirkung (wenn überhaupt) könnte zum Beispiel die Worthäufigkeit auf die Reaktionszeiten bei lexikalischen Entscheidungsaufgaben haben? auf die Benennungszeiten?
 - Wie sieht es mit Unterschieden in den Reaktionszeiten zwischen jüngeren und älteren Teilnehmern aus?
- Welchen Effekt (wenn überhaupt) könnte die Wortkategorie auf die Reaktionszeiten haben?

1 Datenvisualisierung

- Die Visualisierung unserer Daten hilft uns, die Beziehung zwischen den Variablen zu veranschaulichen, um eine Geschichte zu erzählen.
- In der Regel visualisieren wir Variablen, für die wir eine bestimmte Hypothese haben: Prädiktor- und Messvariable(n)

1.1 Visualisierung von Verteilungen

- Histogramme, Dichtediagramme und Balkendiagramme für Zählwerte visualisieren die *Verteilung* von Beobachtungen
 - Sie geben Aufschluss darüber, wie oft wir bestimmte Werte einer Variablen beobachtet haben.
 - In der Regel tun wir dies, um ein Gefühl dafür zu bekommen, wie unsere Daten aussehen
 - * Was ist der Bereich unserer Daten, der Modus, die Gesamtverteilung der Werte?

💡 Aufgabe: Beziehungen visualisieren

1. Erstellen Sie ein Diagramm, das die Verteilung der Häufigkeit der geschriebenen Wörter visualisiert.
2. Erstellen Sie ein Diagramm, das die Verteilung von Substantiven und Verben visualisiert.

2 Visualisierung von Beziehungen

- Um Beziehungen zwischen Variablen zu visualisieren, müssen wir mindestens zwei Variablen auf die Ästhetik eines Diagramms abbilden
- Wir haben dies bereits getan, indem wir Farbe oder Füllung einer kategorischen Variable zugeordnet haben, während wir
 - eine kontinuierliche Variable auf die x-Achse für Histogramme/Dichte-Diagramme, oder
 - eine kategoriale Variable auf die y-Achse für ein Balkendiagramm

💡 Aufgabe: Visualisierung von Beziehungen in Verteilungen

1. Fügen Sie den soeben erstellten Diagrammen eine weitere Ästhetik hinzu, um sie darzustellen:
 - die Verteilung der WritenFrequency-Werte für Wörter mit Anfangskonsonanten und Vokalen
 - die Verteilung der Substantive und Verben für Wörter mit Anfangskonsonanten und Vokalen

2.1 Gruppierte kontinuierliche Variable

- Unsere Histogramme, Dichtediagramme und Balkendiagramme zeigen die Verteilung der Werte einer *kontinuierlichen* Variable nach verschiedenen Stufen einer *kategorischen* Variable

2.1.1 Gestapelt

- Beachten Sie, dass diese Kategorien standardmäßig übereinander gestapelt sind.

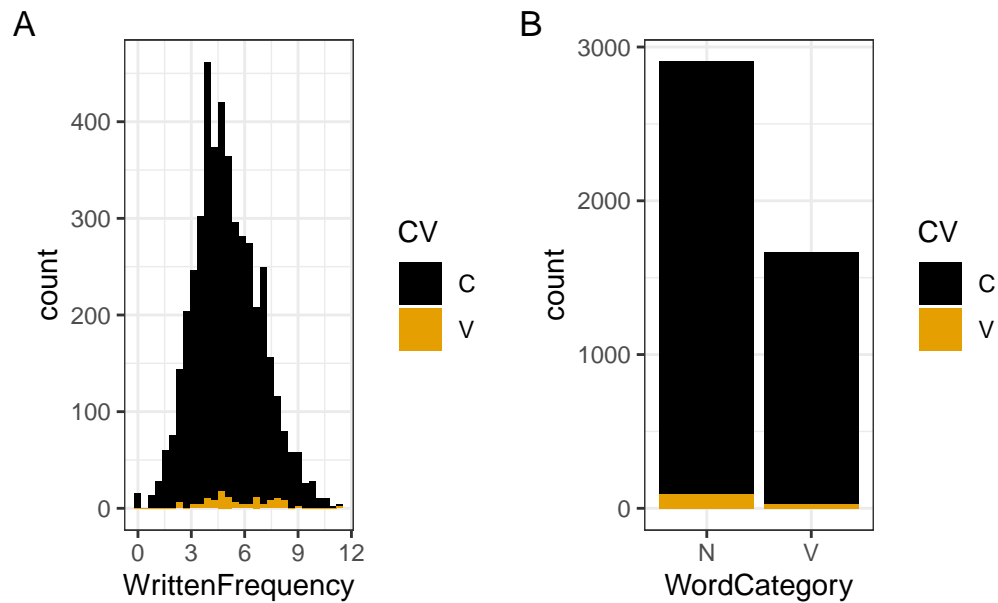


Abbildung 1: Visualising relationships in distributions

2.1.2 Dodged (Ausgewiche)

- aber dass wir sie nebeneinander haben können, indem wir `identity` auf `dodge` setzen
 - Ich finde, dass dies für Balkenplots nützlicher ist

2.2 Zwei kontinuierliche Variablen

- Wir wollen oft die Auswirkungen einer kontinuierlichen Variable auf eine andere sehen.
- In unserem Datensatz `english` haben wir zum Beispiel die Variablen `WrittenFrequency` und `RTlexdec`
 - Welche Art von Beziehung werden diese beiden Variablen Ihrer Meinung nach haben?
 - Denken Sie z.B., dass Wörter mit einer niedrigeren `WrittenFrequency` in einer lexikalischen Entscheidungsaufgabe tendenziell längere oder kürzere Reaktionszeiten haben werden?
 - Wie könnte man sich eine solche Beziehung vorstellen?

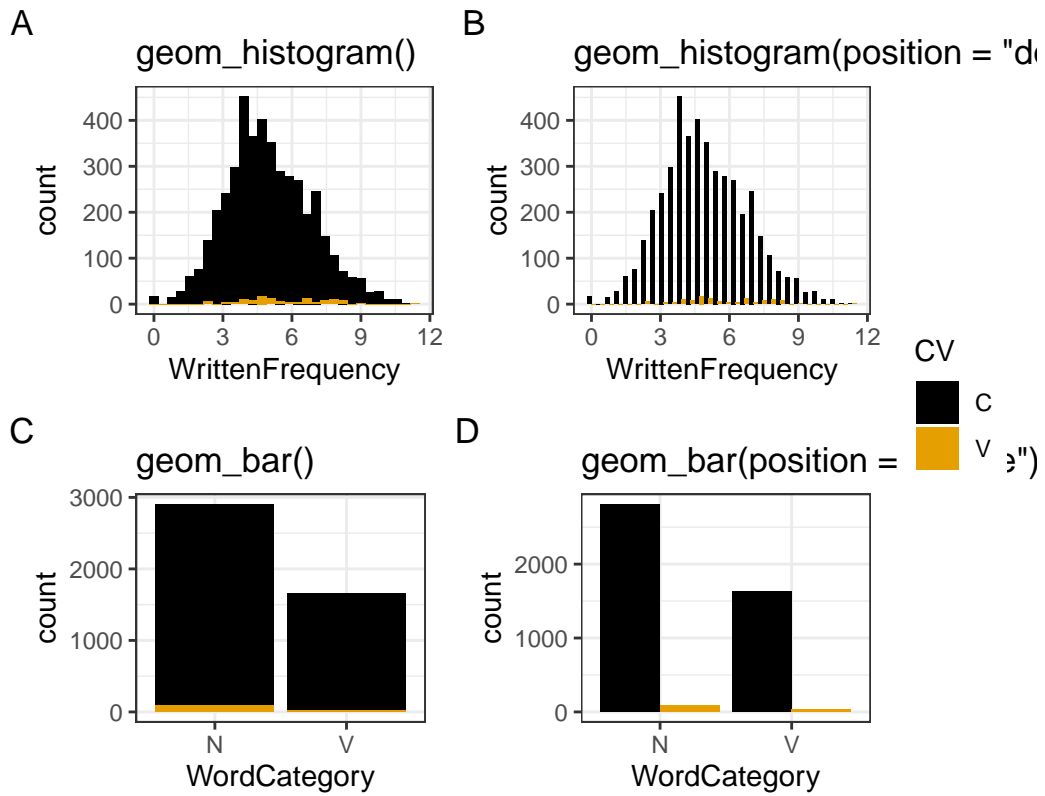
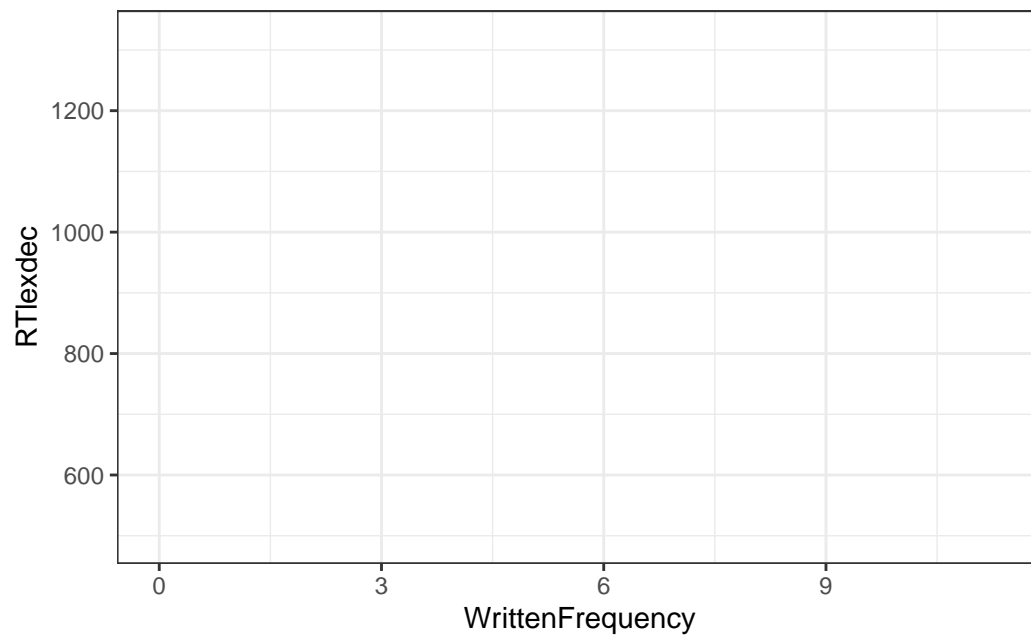
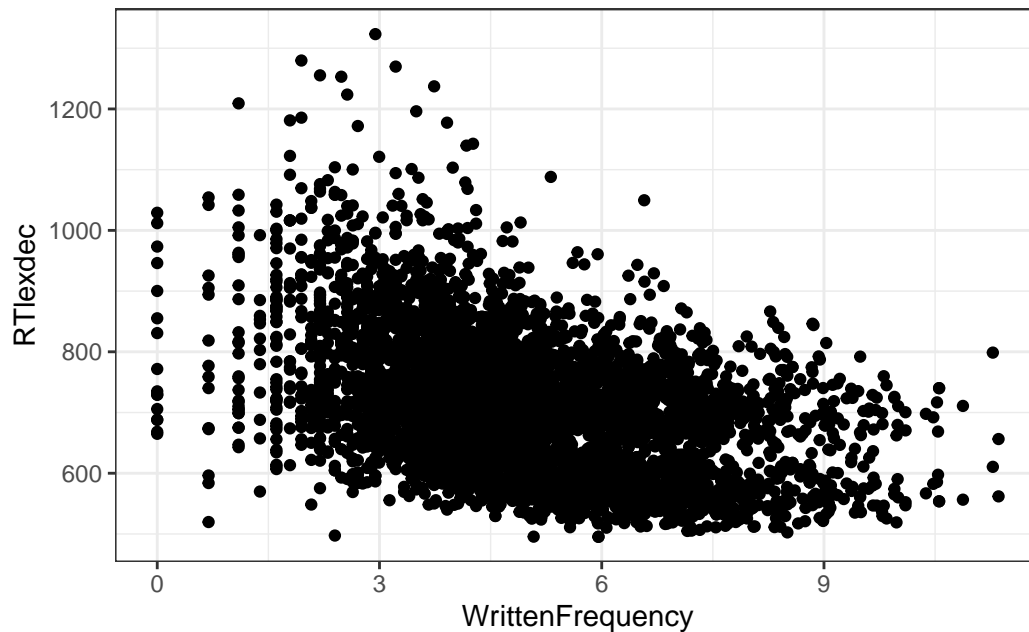


Abbildung 2: Visualising relationships in distributions


```
# + geom_  
df_english |>  
  ggplot() +  
  aes(x = WrittenFrequency, y = RTlexdec)
```



```
df_english |>  
  ggplot() +  
  aes(x = WrittenFrequency, y = RTlexdec) +  
  geom_point()
```

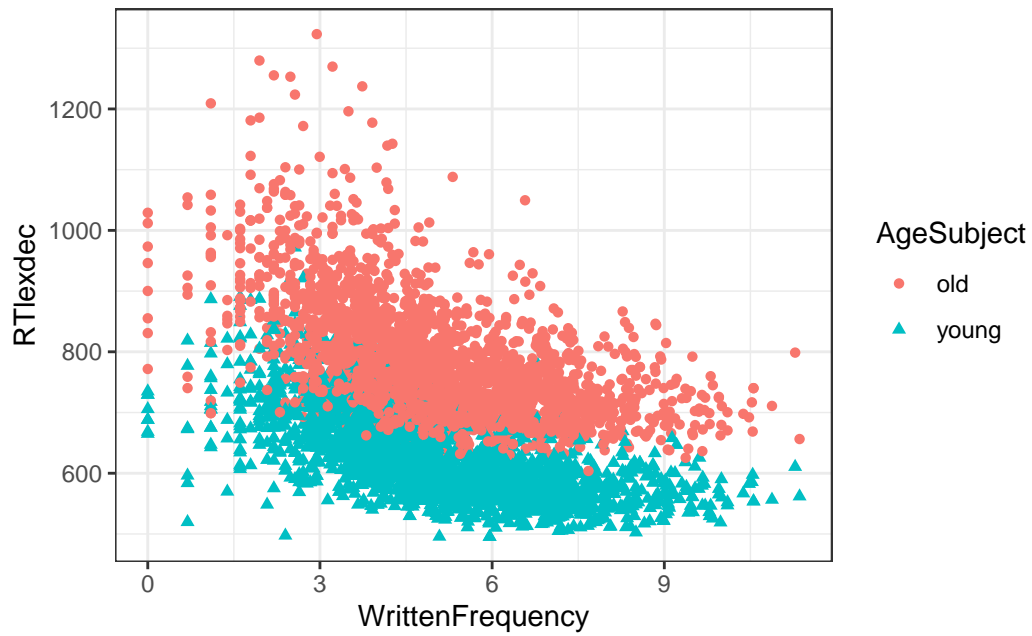


- Nehmen Sie sich einen Moment Zeit, um diese Grafik zu betrachten und eine Interpretation zu finden
 - Welchen Einfluss hat die Schrifthäufigkeit eines Wortes auf die Reaktionszeit bei einer lexikalischen Entscheidungsaufgabe?
 - Vervollständigen Sie den Satz: “Wörter mit einer höheren Worthäufigkeit lösten _____ Reaktionszeiten aus”
- Wo gab es mehr Variation in den Reaktionszeiten? Wo gab es weniger Variation?

2.3 Hinzufügen weiterer Variablen

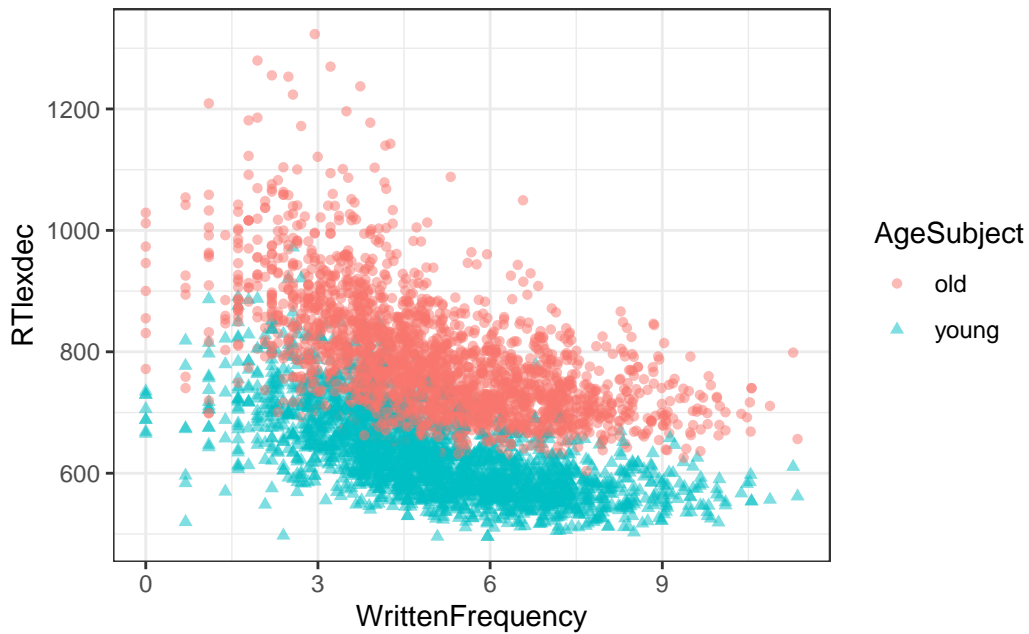
- Erinnern Sie sich daran, dass wir andere Ästhetiken wie `fill` oder `colour` verwenden können
 - für `geom_point()` ist es auch hilfreich, `shape` zu verwenden

```
df_english |>
  ggplot() +
  aes(x = WrittenFrequency, y = RTlexdec,
      colour = AgeSubject,
      shape = AgeSubject) +
  geom_point()
```



- In der Mitte des Diagramms gibt es viele Überschneidungen.
 - Wie können wir die Deckkraft der Punkte ändern?

```
df_english |>
  ggplot() +
  aes(x = WrittenFrequency, y = RTlexdec,
      colour = AgeSubject,
      shape = AgeSubject) +
  geom_point(alpha = .5)
```



- den Zusammenhang zwischen Altersgruppe und Reaktionszeit beschreiben

💡 Aufgabe 2.1: Hinzufügen einer weiteren Variablen

Beispiel 2.1.

Wie könnten Sie eine vierte Variable in die obige Darstellung einfügen? Versuchen Sie, CV hinzuzufügen. Ergibt die Darstellung immer noch eine klare Geschichte?

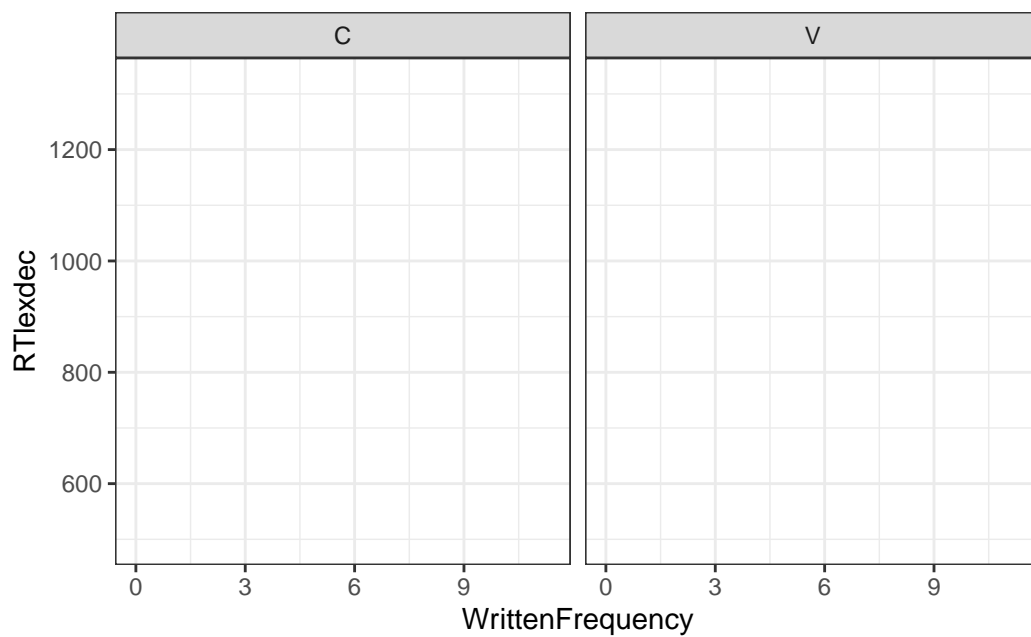
2.4 Facet grids

- Wenn Sie mehr als drei Variablen darstellen wollen, ist es im Allgemeinen eine gute Idee, kategoriale Variablen in *Facetten* aufzuteilen
 - Facetten sind Teilplots, die Teilmengen der Daten anzeigen
- wir können `facet_wrap()` verwenden, das eine Formel als Argument annimmt
 - Diese Formel enthält `~` und den Namen einer kategorialen Variable, z. B. `~CV`

```

1 # + geom_?
2 df_english |>
3   ggplot() +
4     aes(x = WrittenFrequency, y = RTlexdec,
5         colour = AgeSubject,
6         shape = AgeSubject) +
7     facet_wrap(~CV)

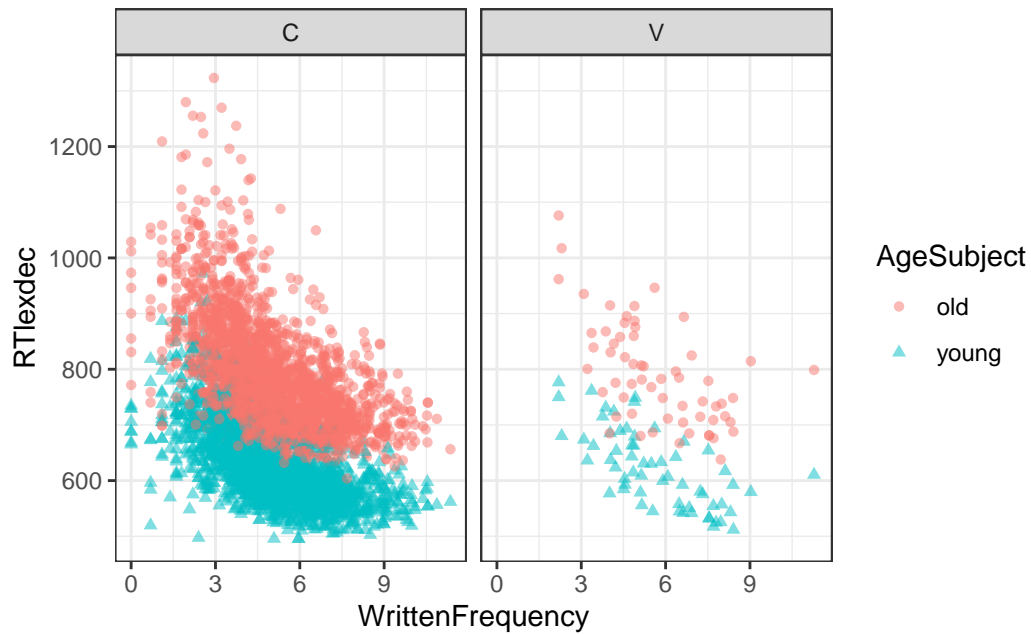
```



```

1 df_english |>
2   ggplot() +
3     aes(x = WrittenFrequency, y = RTlexdec,
4         colour = AgeSubject,
5         shape = AgeSubject) +
6     facet_wrap(~CV) +
7     geom_point(alpha = .5)

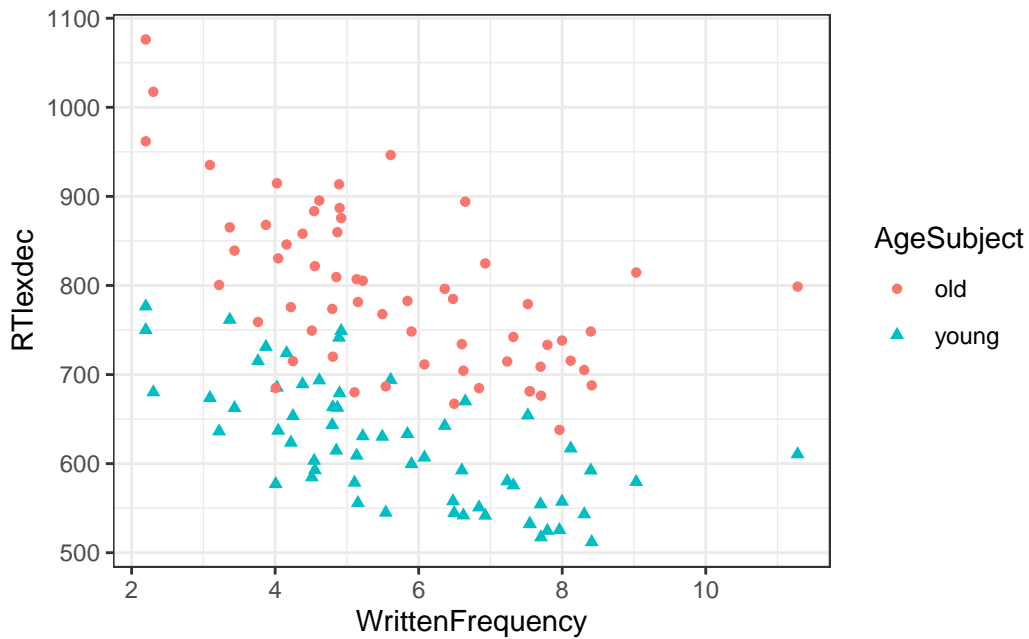
```



3 Bearbeitete Daten

- Wir können unsere Daten auch bearbeiten, bevor wir sie in `ggplot()` eingeben
 - Dies ist nützlich, wenn wir keine permanenten Änderungen an den Daten vornehmen wollen, sondern nur eine Teilmenge der Daten darstellen wollen
- Vielleicht wollen wir nur die Wörter betrachten, die mit einem Vokal beginnen
 - Wie könnten wir das mit einem `dplyr`-Verb machen?

```
df_english |>
  filter(CV == "V") |>
  ggplot() +
  aes(x = WrittenFrequency, y = RTlexdec,
      colour = AgeSubject,
      shape = AgeSubject) +
  geom_point()
```

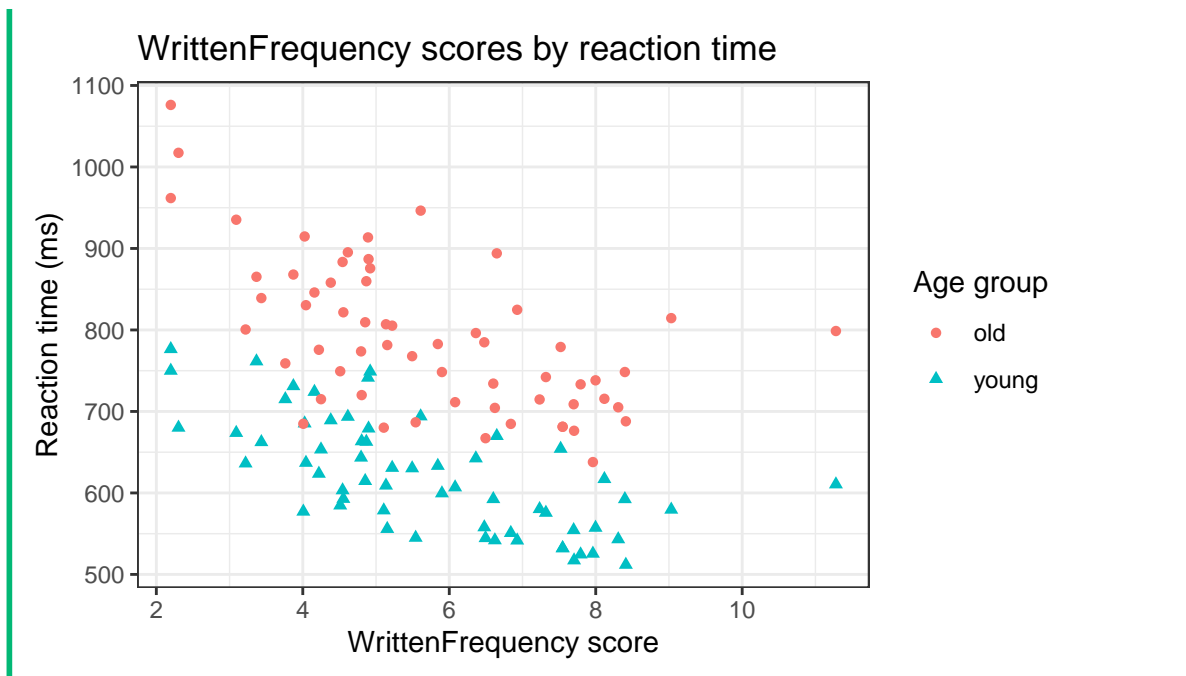


💡 Aufgabe 3.1: Plot-Anmerkung

Beispiel 3.1.

- Vergessen Sie nicht, Ihre Diagramme mit nützlichen Beschriftungen zu versehen, um dem Leser die Interpretation des Diagramms zu erleichtern
- Fügen wir einen Titel und Beschriftungen für die x- und y-Achse hinzu

```
df_english |>
  filter(CV == "V") |>
  ggplot() +
  aes(x = WrittenFrequency, y = RTlexdec,
      colour = AgeSubject,
      shape = AgeSubject) +
  labs(title = "WrittenFrequency scores by reaction time",
       x = "WrittenFrequency score",
       y = "Reaction time (ms)",
       colour = "Age group",
       shape = "Age group") +
  geom_point()
```



4 Quarto Code Chunk Einstellungen

- lange Codeabschnitte können zu sehr unübersichtlichen Ausgabedokumenten führen
- normalerweise ist nur die Darstellung für den Leser wichtig, nicht der Code, der sie erzeugt hat
- wir können die Darstellung und Auswertung von Code Chunks durch Code Chunk Optionen steuern
 - diese beginnen mit `#|`
 - und befinden sich direkt unter ````{r}````
- wichtige Code-Chunk-Optionen:

Tabelle 2: Most common chunk options

option	values	function
<code>##124;##124; echo:</code>	true/false	should this code chunk be printed when rendering?
<code>##124;##124; eval:</code>	true/false	should this code chunk be run when rendering?

4.1 Verwendung von Code-Bausteinen

- warum sehen wir das Ergebnis dieser Darstellung nicht?


```

```{r}
#| eval: false
df_english |>
 ggplot() +
 aes(x = RTlexdec, y = RTnaming,
 colour = AgeSubject,
 shape = AgeSubject) +
 geom_point()
```

```

5 Plots speichern

- oft wollen wir unsere Plots in einem Dokument verwenden, das nicht in RStudio erstellt wurde
 - zum Beispiel in einer Dissertation oder einem in LaTeX geschriebenen Papier
- um dies zu tun, müssen wir unsere Zahlen als einen akzeptierten Dateityp laden, wie **jpeg** oder **png**
- Das können wir mit der Funktion **ggsave()** machen.
- Können Sie erraten, welche Arten von Argumenten **ggsave()** benötigt, um unsere Diagramme zu speichern? Einige sind erforderlich, einige sind optional.

5.1 ggsave()

Als Minimum benötigt **ggsave()** Argumente:

1. den Namen des Plots in Ihrer Umgebung, den Sie speichern möchten
2. den Dateinamen, unter dem Sie Ihre Darstellung speichern möchten
 - Es ist eine gute Idee, einen Ordner zu erstellen, in dem Sie Ihre Plots speichern, und den Dateipfad in den Namen aufzunehmen

5.1.1 ggsave() optionale Argumente

- einige optionale Argumente sind:
 - **width** = wie breit soll der Plot in cm, mm, Zoll oder Pixel sein?
 - **height** = wie hoch soll der gespeicherte Plot in cm, mm, Zoll oder Pixel sein?

- `dpi` = gewünschte Auflösung (numerisch, oder eine Reihe von Strings: “retina” = 320, “print” = 300 oder “screen” = 72)

Warnung

Setzen Sie Code-Chunks, die Dateien auf Ihrem Rechner speichern, *immer* auf `eval: false!!!`. Andernfalls wird jedes Mal, wenn Sie Ihr Skript ausführen, die Datei lokal neu geschrieben.

Aufgabe 5.1: `ggsave()`

Beispiel 5.1.

1. Kopieren Sie den unten stehenden Code in einen Codechunk und führen Sie ihn aus. Schauen Sie sich Ihre “Files”-Tab an, was hat sich geändert?

```
```{r}
#| eval: false
ggsave(
 # required:
 "figures/04-dataviz2/fig_lexdec_rt.png",
 plot = fig_lexdec_rt,
 # optional:
 width = 2000,
 height = 1000,
 units = "px",
 scale = 1,
 dpi = "print")
```
```

2. Versuchen Sie, mit dem Maßstab und den dpi zu spielen. Was ändert sich?
3. Versuchen Sie, die Werte für Einheiten, Breite und Höhe zu ändern. Was ändert sich?

6 Hausaufgaben

[Anhang 5](#) auf der Website des Kurses.

Session Info

Hergestellt mit R version 4.4.0 (2024-04-24) (Puppy Cup) und RStudioversion 2023.9.0.463 (Desert Sunflower).

```
sessionInfo()
```

```
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;
```

```
locale:
```

```
[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
```

```
time zone: Europe/Berlin
```

```
tzcode source: internal
```

```
attached base packages:
```

```
[1] stats      graphics  grDevices datasets  utils      methods    base
```

```
other attached packages:
```

```
[1] kableExtra_1.4.0 knitr_1.46      languageR_1.5.0 ggthemes_5.1.0
[5] patchwork_1.2.0  lubridate_1.9.3 forcats_1.0.0   stringr_1.5.1
[9] dplyr_1.1.4      purrr_1.0.2     readr_2.1.5     tidyr_1.3.1
[13] 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      xml2_1.3.6
[5] stringi_1.8.3   hms_1.1.3       digest_0.6.35   magrittr_2.0.3
[9] evaluate_0.23   grid_4.4.0      timechange_0.3.0 fastmap_1.1.1
[13] jsonlite_1.8.8  tinytex_0.50    fansi_1.0.6     viridisLite_0.4.2
[17] scales_1.3.0    cli_3.6.2       rlang_1.1.3     munsell_0.5.1
[21] withr_3.0.0     yaml_2.3.8      tools_4.4.0     tzdb_0.4.0
[25] colorspace_2.1-0 pacman_0.5.1     vctrs_0.6.5     R6_2.5.1
[29] lifecycle_1.0.4 pkgconfig_2.0.3 pillar_1.9.0    gtable_0.3.5
[33] glue_1.7.0      systemfonts_1.0.6 xfun_0.43       tidyrselect_1.2.1
[37] rstudioapi_0.16.0 farver_2.1.1     htmltools_0.5.8.1 labeling_0.4.3
[41] rmarkdown_2.26  svglite_2.1.3   compiler_4.4.0
```

Literaturverzeichnis

- Baayen, R. H., & Shafaei-Bajestan, E. (2019). *languageR: Analyzing Linguistic Data: A Practical Introduction to Statistics*. <https://CRAN.R-project.org/package=languageR>
- Nordmann, E., & DeBruine, L. (2022). *Applied Data Skills*. Zenodo. <https://doi.org/10.5281/zenodo.6365078>
- Wickham, H., Çetinkaya-Rundel, M., & Grolemund, G. (2023). *R for Data Science* (2. Aufl.).