Datenvisualiserung 2

Visualisierung von Beziehungen

Daniela Palleschi
Humboldt-Universität zu Berlin

Mi. den 25.10.2023

Überblick

- Set-up
- Datenvisualisierung
- Visualisierung von Beziehungen
- Bearbeitete Daten
- Quarto Code Chunk Einstellungen
- Plots speichern
- Übungen

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

▶ Hinweis

Learning objectives

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

Die **Pflichtlektüre** zur Vorbereitung auf dieses Thema ist Kap. 2 (Datenvisualisierung) aus Abschnitt 2.5 in Wickham et al. (2023).

Eine **ergänzende Lektüre** ist Ch. 3 (Data visualtion) in Nordmann & DeBruine (2022).

Set-up

Packages

```
1 library(tidyverse)
2 library(patchwork)
3 library(ggthemes)
4 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

Ich habe mein bevorzugtes `ggplot`-Thema global festgelegt. Das bedeutet, dass nach dem Ausführen dieses Codes alle Diagramme dieses Thema verwenden werden.

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

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	type
RTlexdec	Reaktionszeiten für eine visuelle lexikalische Entscheidung (Millisekunden)	kontinuierlich
RTnaming	Reaktionszeiten für den Beginn einer verbalen Wortbenennungsaufgabe (Millisekunden)	kontinuierlich
WrittenFrequency	numerischer Vektor mit der logarithmischen Häufigkeit in der lexikalischen Datenbank von CELEX	kontinuierlich
Wort	ein Faktor mit 2284 Wörtern	kategorisch
AgeSubject	ein Faktor mit der Altersgruppe des Probanden als Level: jung versus alt	kategorisch
WordCategory	ein Faktor mit den Wortkategorien N (Substantiv) und V (Verb) als Ebenen	kategorisch
CV	Faktor, der angibt, ob das Anfangsphonem des Wortes ein Konsonant (C) oder ein Vokal (V) ist.	kategorisch
CorrectLexdec	numerischer Vektor mit dem Anteil der Probanden, die das Item bei der lexikalischen Entscheidung als Wort akzeptiert haben.	kontinuierlich

Hypotheses

- Welche Arten von Hypothesen könnten Sie für solche Daten aufstellen?
  - Unsere Reaktionszeitdaten sind unsere *Messvariablen*.
    - d.h. das, was wir messen
  - Alle anderen Variablen sind mögliche *Vorhersagevariablen*.
    - 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?

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)

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.

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
  - 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 WrittenFrequency-Werte für Wörter mit Anfangskonsonanten und Vokalen
   - die Verteilung der Substantive und Verben für Wörter mit Anfangskonsonanten und Vokalen

Gruppierte kontinuierliche Variable

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

Gestapelt

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

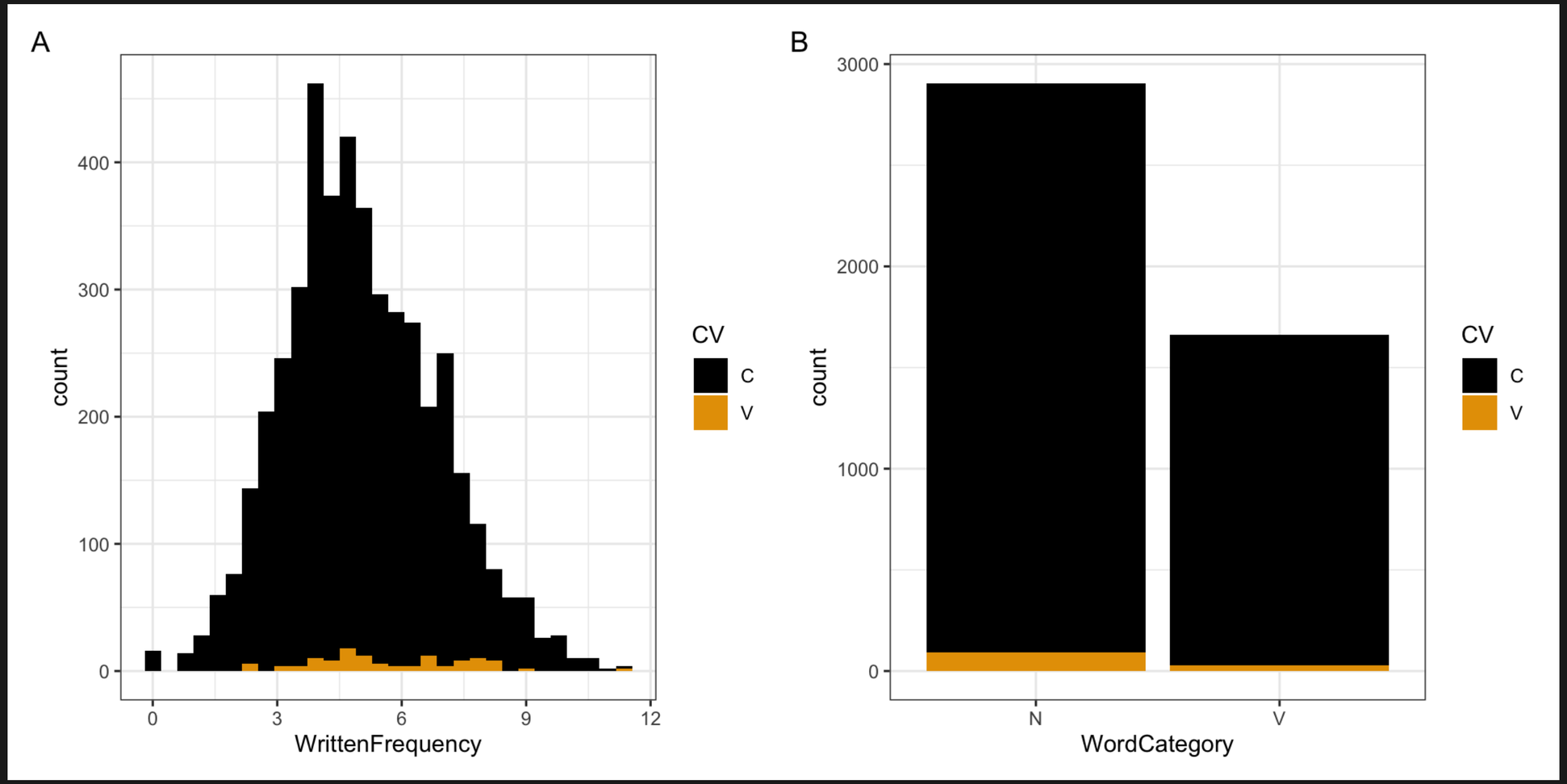


Abbildung 1: Visualising relationships in distributions

Dodged (Ausgewiche)

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

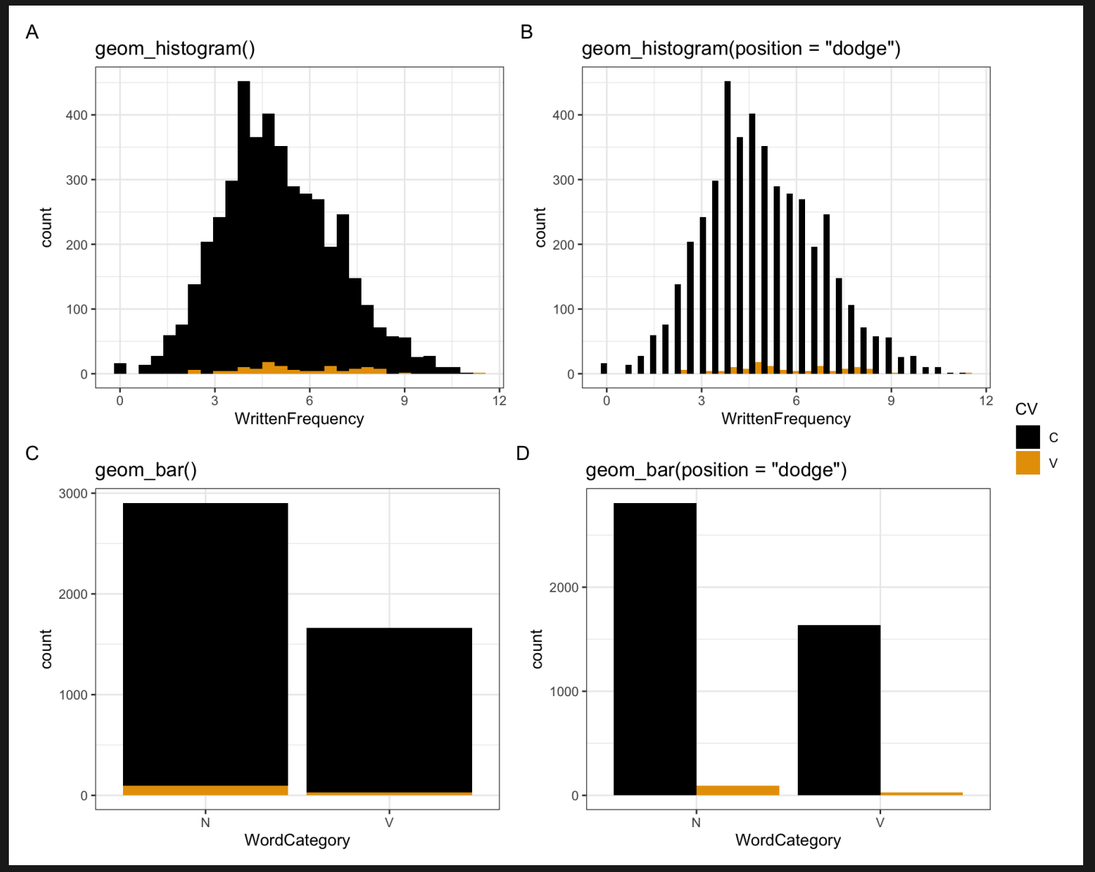
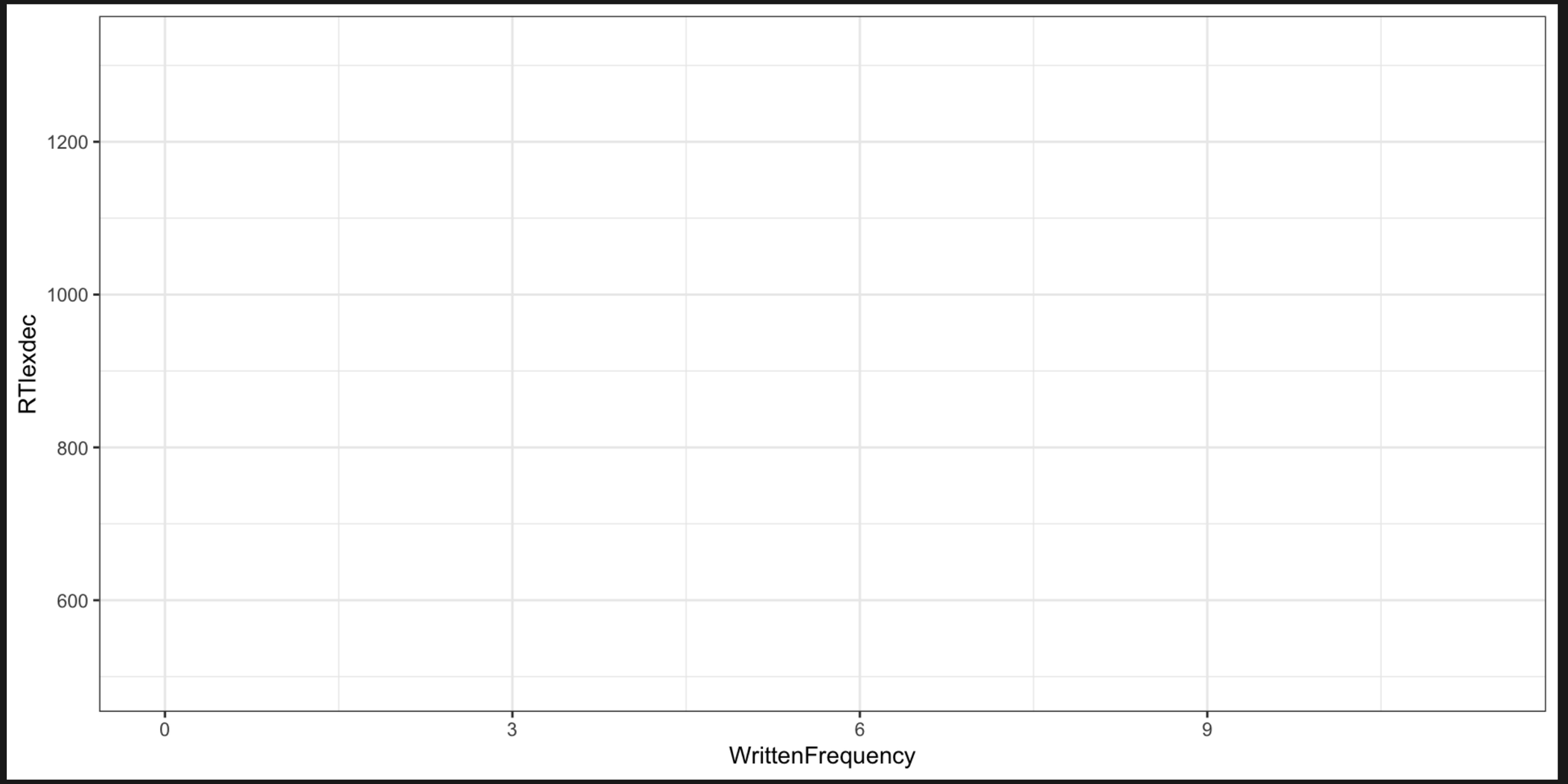


Abbildung 2: Visualising relationships in distributions

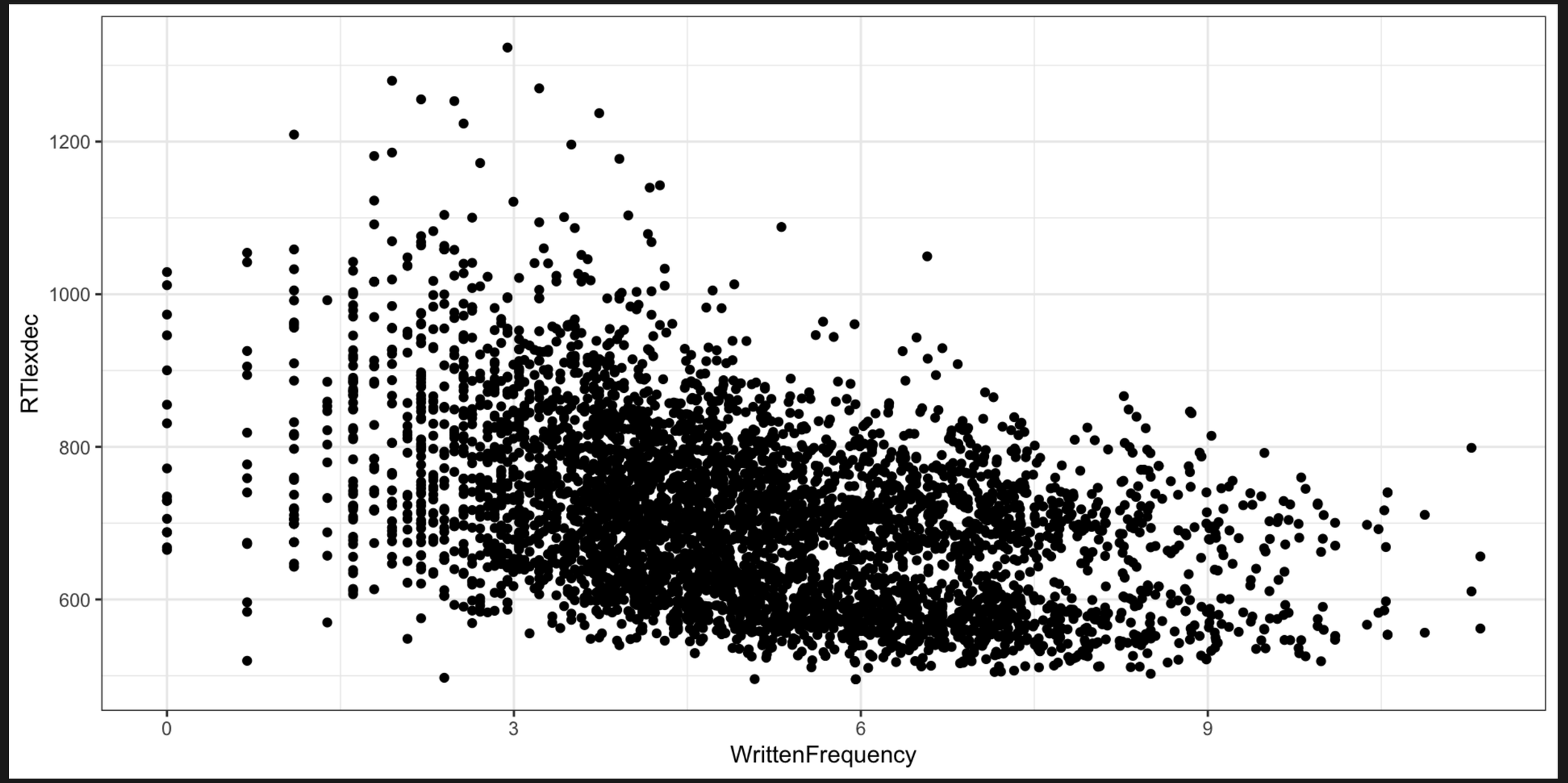
Zwei kontinuierliche Variablen

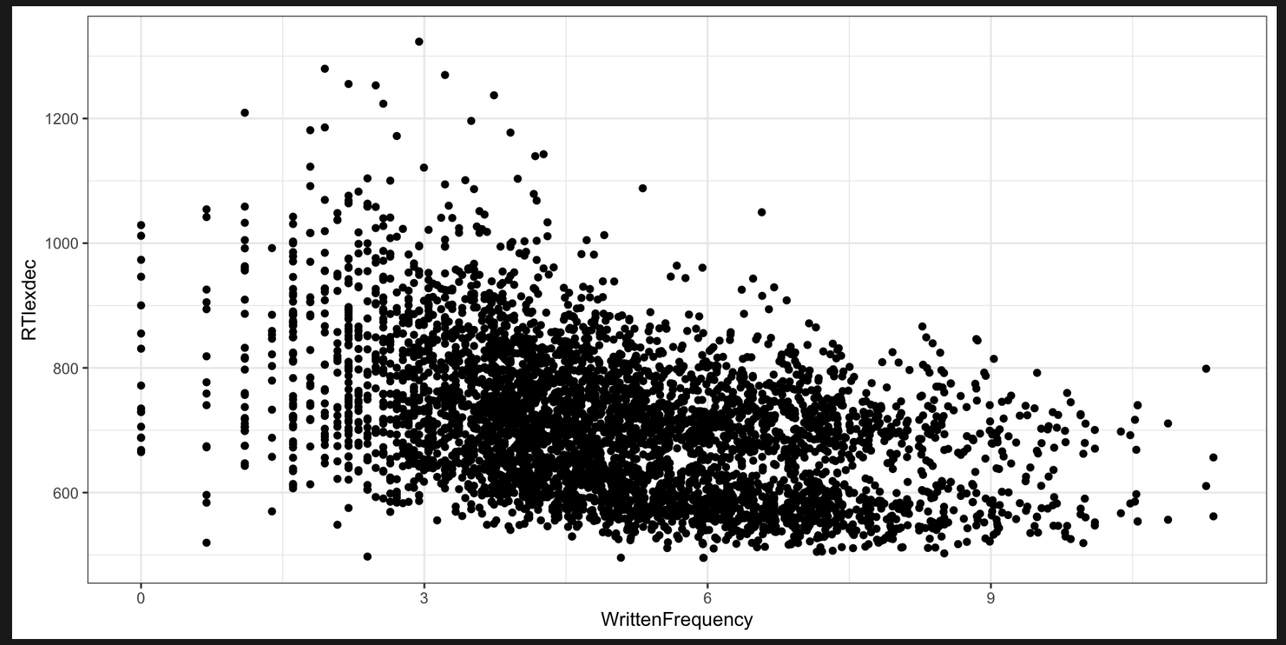
- Wir wollen oft die Auswirkungen einer kontinuierlichen Variable auf eine andere sehen.
- In unserem Datensatz `english` haben wir zum Beispiel die Variablen `WrittenFreuqency` und `RTlexdec`
  - Welche Art von Beziehung werden diese beiden Variablen Ihrer Meinung nach haben?
  - Denken Sie z.B., dass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

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



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



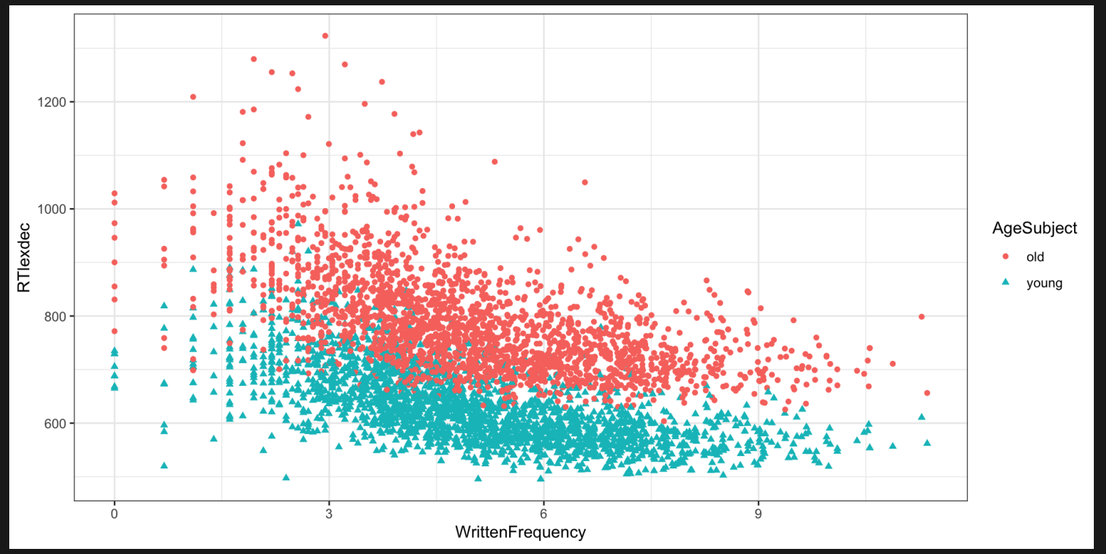


- Nehmen Sie sich einen Moment Zeit, um diese Grafik zu betrachten und eine Interpretation zu finden
  - Welchen Einfluss hat die Schrifthäufigkeit eines Wortes auf die Reaktionszeit bei einer lexikalischen Entscheidungsaufgabe?
  - Vervollständigen Sie den Satz: “Wörter mit einer höheren Worthäufigkeit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

Hinzufügen weiterer Variablen

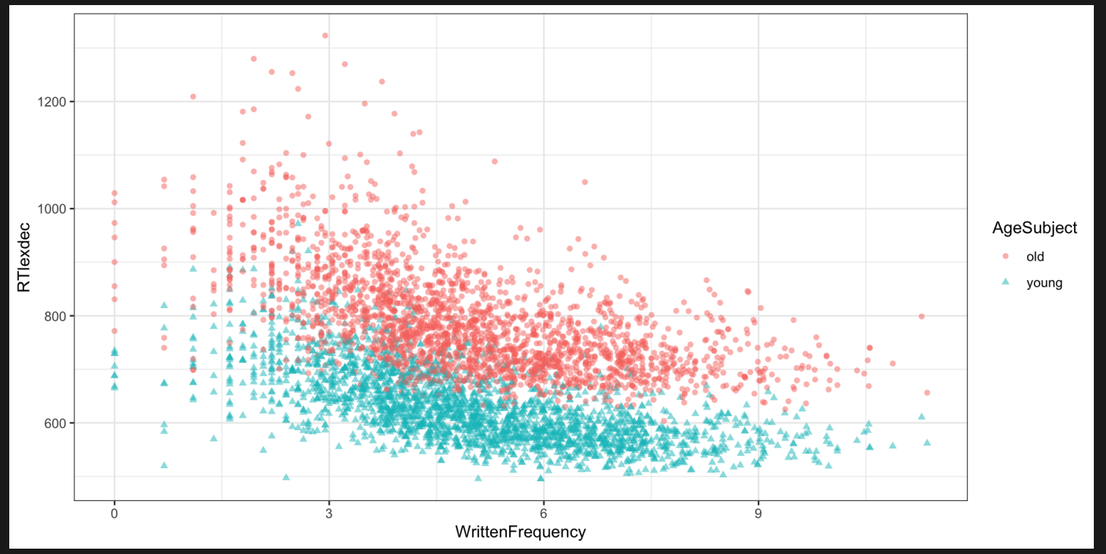
- Erinnern Sie sich daran, dass wir andere Ästhetiken wie `fill` oder `colour` verwenden können
  - für `geom_point()` ist es auch hilfreich, `shape` zu verwenden

```
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?

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



- den Zusammenhang zwischen Altersgruppe und Reaktionszeit beschreiben

Aufgabe

Aufgabe 1: `Adding another variable`

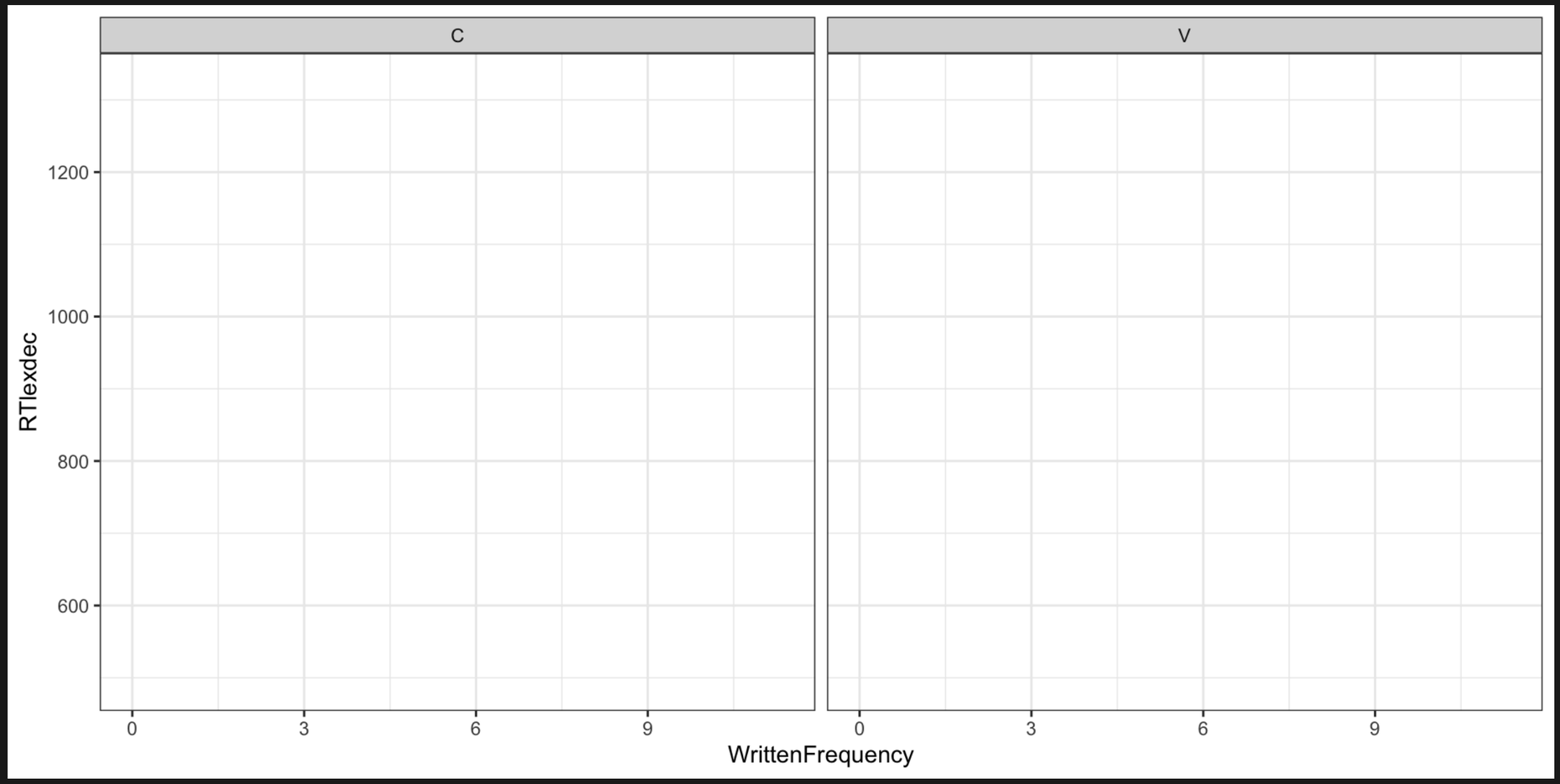
Beispiel 1

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

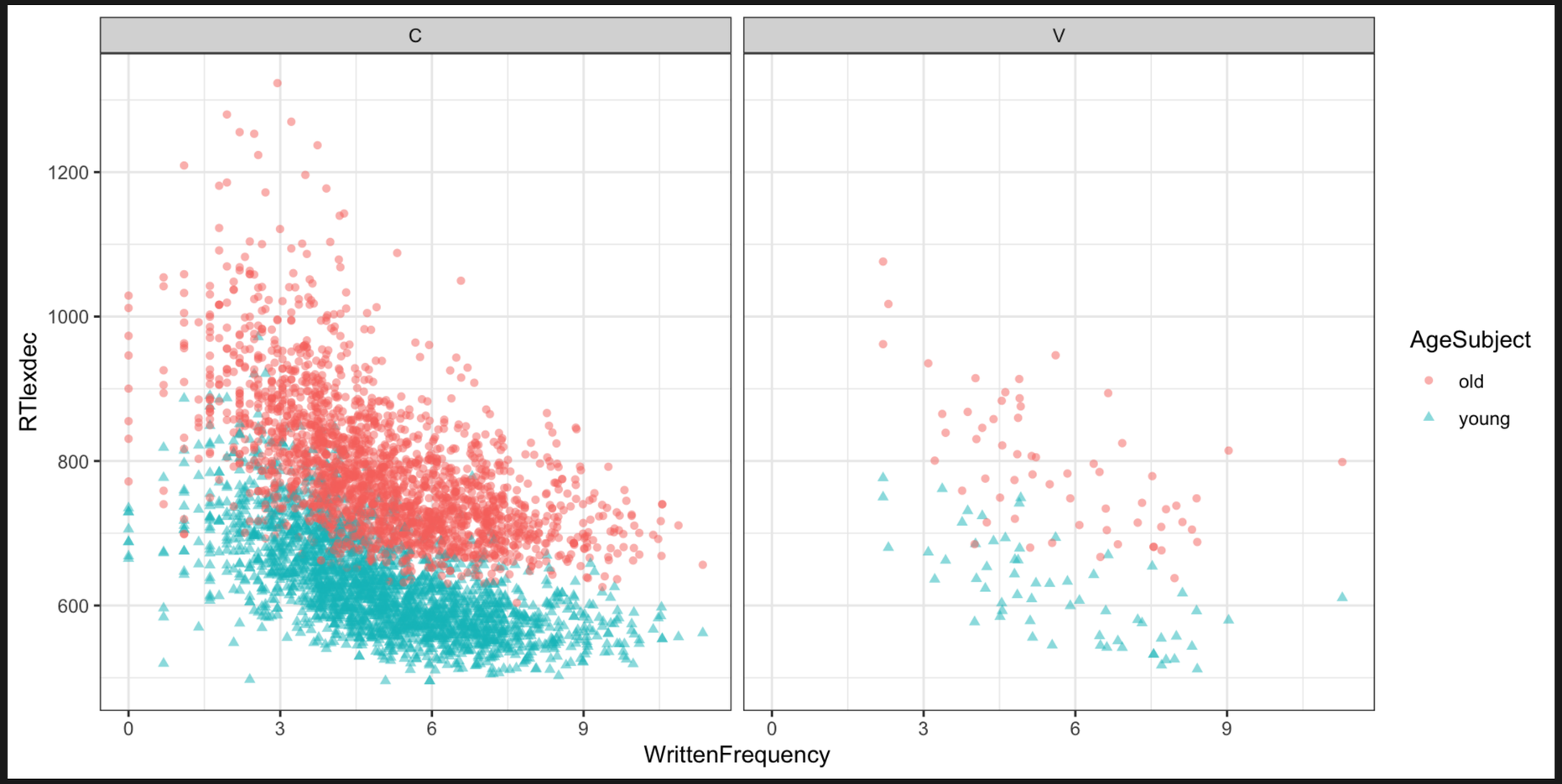
Facet grids

- Wenn Sie mehr als drei Variablen darstellen wollen, ist es im Allgemeinen eine gute Idee, kategorische 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`

```r
# + geom_?
df_english |>
  ggplot() +
  aes(x = WrittenFrequency, y = RTlexdec,
      colour = AgeSubject,
      shape = AgeSubject) +
  facet_wrap(~CV)
```



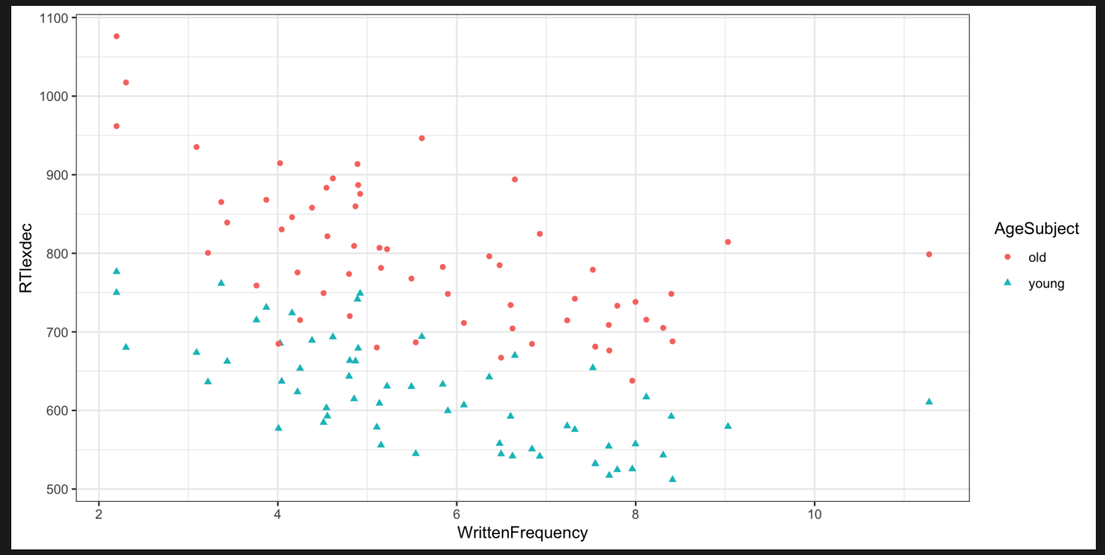
```r
df_english |>
  ggplot() +
  aes(x = WrittenFrequency, y = RTlexdec,
      colour = AgeSubject,
      shape = AgeSubject) +
  facet_wrap(~CV) +
  geom_point(alpha = .5)
```



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?

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



Aufgabe

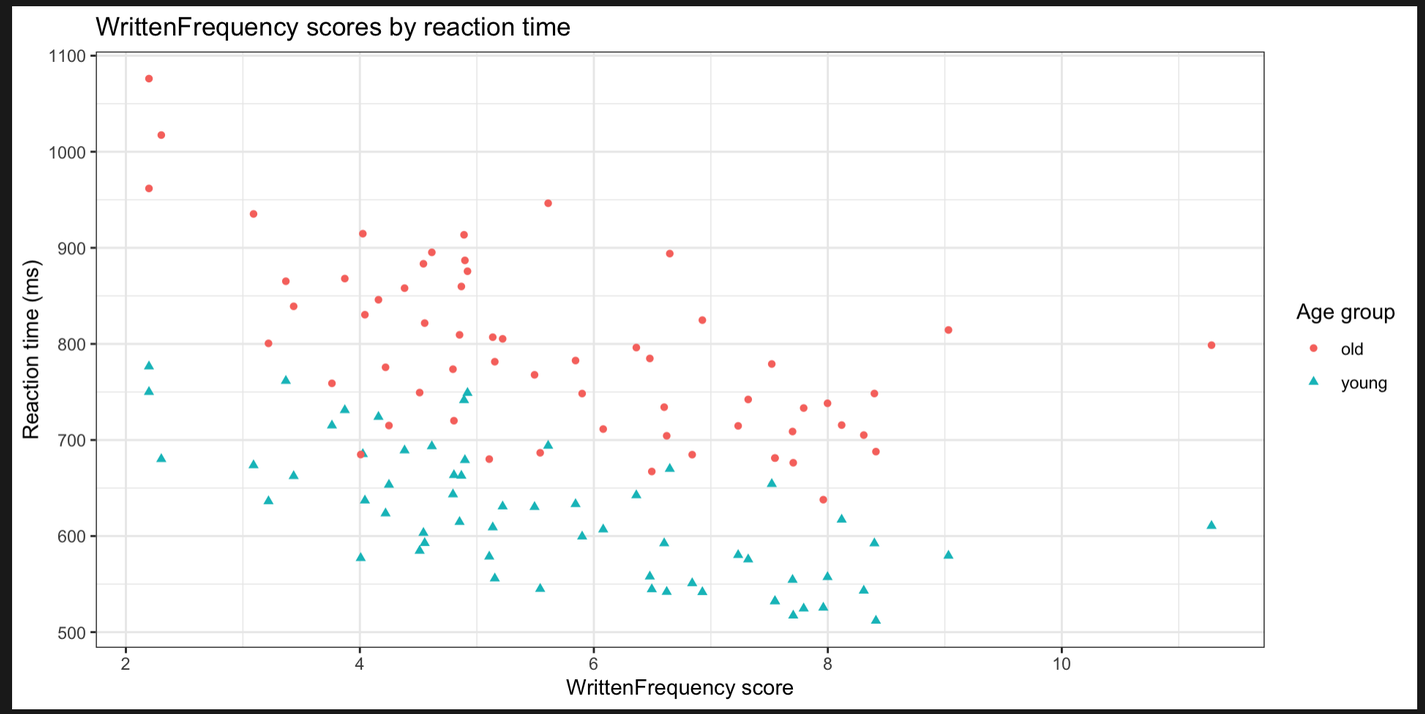
Aufgabe 2: Plot-Anmerkung

Beispiel 2

- 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()
```

Aufgabe


WrittenFrequency scores by reaction time
Reaction time (ms)
1100
1000
900
800
700
600
500
2
4
6
8
10
WrittenFrequency score
Age group
old
young

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
#| echo:	true/false	should this code chunk be printed when rendering?
#| eval:	true/false	should this code chunk be run when rendering?

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()
```
````

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.

`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

`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 gespeichert Plot in cm, mm, Zoll oder Pixel sein?
  - `dpi =` gewünschte Auflösung (numerisch, oder eine Reihe von Strings: “retina” = 320, “print” = 300 oder “screen” = 72)

`eval: false`

⚠ 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

Aufgabe 3: `ggsave()`

Beispiel 3

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?

Übungen

1. a. Zeichnen Sie abweichende Balkenplots von `AgeSubject` (x-Achse) nach `CV` (Facetten).

   b. Ändern Sie Ihre Code-Chunk-Optionen für den letzten Plot so, dass der Code, aber nicht der Plot, in der Ausgabe gedruckt wird.

2. a. Filtern Sie die Daten, um nur ältere Teilnehmer einzuschließen, und stellen Sie `RTlexdec` (x-Achse) durch `RTnaming` (y-Achse) dar. Übertragen Sie `CV` auf Farbe und Form. Fügen Sie geeignete Beschriftungen hinzu.

   b. Ändern Sie die Code-Chunk-Optionen für den letzten Plot so, dass der Plot, aber nicht der Code, in der Ausgabe gedruckt wird.

3. Speichern Sie den letzten Plot lokal und stellen Sie den Code Chunk so ein, dass er beim Rendern *nicht* ausgeführt wird.

Session Info

Hergestellt mit R version 4.3.0 (2023-04-21) (Already Tomorrow) und RStudioversion 2023.9.0.463 (Desert Sunflower).

```
1 sessionInfo()
```

```
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;  LAPACK version 3.11.0

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:
```

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.).