

Angewandte Datenverarbeitung und Visualisierung (WiSe23/24)

WiSe23/24

Daniela Palleschi

2023-10-16

Inhaltsverzeichnis

Kursübersicht	5
Kursbeschreibung auf AGNES	5
Ziele des Kurses	5
Ressourcen	6
I. Kursübersicht	7
Syllabus	8
Erforderliche Software	9
R und RStudio	10
Pakete	10
RStudio Globale Optionen (optional)	11
tinyTex (optional)	12
II. Grundlagen	13
1. Einführung in R und RStudio	14
Heutige Ziele	14
Weitere Lektüre	14
1.1. Vorbereitung	14
1.2. RProjekt	15
1.3. R in RStudio	16
1.4. Reproduzierbarkeit	19
1.5. Rechnen in R	23
1.6. Vektoren	26
1.7. Endergebnis	27
1.8. Session Info	28
1.9. Nächste Woche	28
2. Datenvisualisierung 1	31
Heutige Ziele	31

2.1. Datenrahmen	33
2.2. Lexical Decision Task (LDT)	34
2.3. lexdec Datensatz	36
2.4. Erstellen von Plots mit ggplot2	38
2.5. Entscheidung für ein Geom	51
2.6. Exercises	51
.	51
Session Info	52
3. Dynamic reports with Quarto	55
Lernziele	55
Lesungen	55
Wiederholung	55
Set-up	57
3.1. Quarto	58
3.2. Unsere erstes Quarto-Dokument	59
3.3. Codierung in Quarto	64
3.4. Plots in Quarto	66
3.5. Ausgabeformate	69
3.6. Extra: Reproduzierbarkeit in Quarto	71
4. Data Wrangling 1: Transformation	74
Wiederholung	74
Heutige Ziele	74
4.1. Voraussetzungen	75
4.2. Data Wrangling	76
4.3. Zeilen	78
4.4. Spalten	83
4.5. dplyr und ggplot2	89
Aufgaben	90
Session Info	91
5. Datenvisualisierung 2	93
Set-up	94
5.1. Datenvisualisierung	96
5.2. Visualisierung von Beziehungen	97
5.3. Bearbeitete Daten	105
5.4. Quarto Code Chunk Einstellungen	107
5.5. Plots speichern	108
5.6. Übungen	110
6. Bericht 1	112
6.1. Einrichtung	112

6.2. Data wrangling	113
6.3. Datenvisualisierung	114
6.4. Interpretation	115
III. Nächste Stufe	116
7. Einlesen von Daten	117
Lesungen	117
8. Deskriptive Statistik	118
.	118
Lesungen	118
9. Data Wrangling 2	119
.	119
Lesungen	119
10. Datenvisualisierung 2	120
.	120
Lesungen	120
Bericht 2	121
IV. Fortgeschrittene Themen	122
Literaturverzeichnis	123

Kursübersicht

Dies ist die Webseite der Lehrveranstaltung “Angewandte Datenverarbeitung und Visualisierung” an der Humboldt-Universität zu Berlin, Institut der deutschen Sprache und Linguistik für das Wintersemester 2023/24. Wenn Sie für den Kurs eingeschrieben sind, finden Sie alle relevanten Materialien auf dem Kurs Moodle [hier](#) (Moodle-Schlüssel wird in der Vorlesung bereitgestellt).

Jedes Kapitel entspricht einer Vorlesung von einer Woche. Vorerst werden die Materialien auf dieser Website im Bullet-Point-Format erscheinen und genau denselben Inhalt wie die Kursfolien enthalten. Ich plane, die Aufzählungspunkte später in Prosa umzuwandeln und die einzelnen Themen zu vertiefen.

Kursbeschreibung auf AGNES

Dies ist ein Einführungskurs in das Denken, Arbeiten und Kommunizieren mit / über sprachliche Daten. Der Kurs fokussiert sich auf praktische Anwendungen und die Vermittlung übertragbarer Fähigkeiten. In RStudio machen sich die Teilnehmenden mit der Programmiersprache R vertraut und entwickeln Fähigkeiten zur Erstellung und Vermittlung zusammenfassender Statistiken für den akademischen und beruflichen Kontext. Die Teilnehmenden lernen, Rohdaten zu laden und zu manipulieren, Tabellen mit deskriptiven Statistiken zu erstellen und die Daten angemessen visuell darzustellen. Am Ende des Kurses werden die Teilnehmenden ein besseres Verständnis dafür haben, wie man mit Daten umgeht und die Fähigkeiten besitzen, Ergebnisse klar zu kommunizieren. Studierende, die keinen eigenen Laptop zum Unterricht mitbringen können, setzen sich bitte so früh wie möglich mit der Dozentin in Verbindung, damit ein alternativer Laptop organisiert werden kann. Der Kurs wird auf Deutsch gehalten.

Ziele des Kurses

Das Hauptziel dieses Kurses ist es, die Kenntnisse und Fähigkeiten zu entwickeln, die für die Durchführung einer “Explorativen Datenanalyse (EDA)” erforderlich sind. EDA ist kein formaler Prozess mit spezifischen Regeln, sondern vielmehr “a state of mind” (Wickham et al., 2023, Kapitel 11). Das Wissen, das für die Durchführung einer EDA erforderlich ist, besteht

einfach darin, die Daten zu verstehen und ihre Struktur zu erforschen, um ein Verständnis für ihre Verteilung und Muster zu bekommen. Die für die Durchführung einer EDA erforderlichen Fähigkeiten sind spezifisch für die zur Durchführung der EDA verwendete Sprache, in unserem Fall R.

Ressourcen

Die meisten unserer Materialien basieren auf dem Buch “R for Data Science” von Hadley Wickham (2. Auflage), das Sie [hier](#) vollständig online einsehen können. Wo es möglich war, habe ich die in diesem Buch verwendeten Daten durch linguistische Datensätze ersetzt, damit Sie sich ein Bild davon machen können, wie Linguisten R verwenden könnten.

Einige andere Ressourcen, die wir von Zeit zu Zeit verwenden werden oder die Sie vielleicht selbst erkunden möchten, sind das E-book *Data visualisation using R, for researchers who don't use R* (Nordmann et al., 2022) und das Lehrbuch *Statistics for Linguists: An Introduction Using R* by Bodo Winter [Winter (2019); PDF erhältlich über das Grimm Zentrum].

Teil I.

Kursübersicht

Syllabus

Die vorgeschlagene Lektüre erleichtert die Arbeit mit dem Material für jede Woche. Die Lektüre umfasst Kapitel oder Abschnitte aus Nordmann et al. (2022) (web tutorial), Wickham et al. (2023) (E-book), and Winter (2019) (PDF verfügbar über die Grimm-Bibliothek).

v Reading from "WiSe23/24 - BA Datenverarbeitung syllabus".

v Range 'Sheet1'.

Warning: HTML tags found, and they will be removed.

* Set `options(gt.html_tag_check = FALSE)` to disable this check.

HTML tags found, and they will be removed.

* Set `options(gt.html_tag_check = FALSE)` to disable this check.

Woche	Datum	Thema	Vorbereitung
1	18.10.2023	Einführung in R und RStudio	R4DS - Ch 1 (Introduction)
2	25.10.2023	Data Viz 1: Verteilungen	R4DS - Ch 2 (Data visualization)
3	01.11.2023	Dynamische Berichte mit Quarto	R4DS - Ch 29 (Quarto)
4	08.11.2023	Wrangling 1: Umwandlung von Daten	R4DS - Ch 4 (Data transformation)
5	15.11.2023	Data Viz 2: Visualisierung von Beziehungen	R4DS - Ch 5 (Workflow visualization)
6	22.11.2023	Bericht 1	
7	29.11.2023	Daten einlesen	R4DS - Ch 8 (Data import)
8	06.12.2023	Wrangling 2: Tidying data	R4DS - Ch 6 (Data tidying)
9	13.12.2023	Deskriptive Statistik	Winter (2019) - Ch 3 (Descriptive statistics)
10	20.12.2023	Data Viz 3: Visualisierung von Zusammenfassungen	R4DS - Ch 2 (Data visualization)
Vorlesungsfrei	27.12.2023	NA	
Vorlesungsfrei	03.01.2024	NA	
11	10.01.2024	Bericht 2	
12	17.01.2024	Einführung in Base R	R4DS - Ch 28 (A field guide to base R)
13	24.01.2024	Regular expressions	R4DS - Ch 16 (Regular expressions)
14	31.01.2024	Data Viz 4: Kommunikation	R4DS - Ch 12 (Communication)
15	07.02.2024	Bericht 3	
16	14.02.2024	Offene Sitzung: Q&A	

Erforderliche Software

Dieses Dokument beschreibt die Schritte, die erforderlich sind, um unseren reproduzierbaren Arbeitsablauf für den Kurs ‘Angewandte Datenanalyse und -visualisierung’ einzurichten. **?@sec-R** gibt einen Überblick über die Installation von R, RStudio und der erforderlichen Pakete. Diese Schritte sind erforderlich. **?@sec-tinytex** beschreibt die Installation von TinyTex, das benötigt wird, um Dokumente im LaTeX-Stil (z.B. PDFs) in R darzustellen.

R und RStudio

Um an diesem Kurs teilnehmen zu können, müssen Sie R und RStudio installieren.

[R](#) ist eine statistische Programmiersprache, die für statistische Berechnungen und grafische Darstellungen verwendet wird. Am häufigsten wird sie zur Analyse und Visualisierung von Daten verwendet, beides werden wir in diesem Semester tun. [RStudio](#) ist eine IDE (integrierte Entwicklungsumgebung) für R und andere Sprachen. RStudio macht die Analyse und Visualisierung von Daten in R viel einfacher (glauben Sie mir, als ich mit R anfang, gab es kein RStudio!).

Sie müssen R herunterladen, bevor Sie RStudio herunterladen können.

1. [R herunterladen](#)
2. [RStudio herunterladen](#)

Pakete

R-Pakete, die im Comprehensive R Archive Network, allgemein bekannt als CRAN-Repository, verfügbar sind, können einfach mit dem Befehl `install.packages("packageName")` installiert werden. Einige Pakete, die wir brauchen werden, sind:

- `here` Paket (Müller, 2020)
- `tidyverse`-Paketfamilie (Wickham et al., 2019)
 - enthält automatisch Pakete, die wir brauchen, wie `dplyr` und `ggplot2`
- `languageR`-Paket (Baayen & Shafaei-Bajestan, 2019)

Um mehrere Pakete auf einmal herunterzuladen, verwenden Sie die ‘concatenate’-Funktion in `r(c())` innerhalb von `install.packages()`:

```
install.packages(c("here",  
                  "tidyverse",  
                  "pacman"))
```

RStudio Globale Optionen (optional)

Hier sind meine bevorzugten globalen Optionen (RStudio > Werkzeuge > Globale Optionen). Ich empfehle dringend, die Einstellungen für “Arbeitsbereich” und “R-Sitzungen” zu befolgen, um die Reproduzierbarkeit zu gewährleisten. Mit den anderen Einstellungen können Sie herumspielen, um herauszufinden, was Ihnen gefällt.

- Allgemein > Grundeinstellungen
 - **Arbeitsbereich** (für reproduzierbare Arbeitsabläufe!!!)
 - * Deaktivieren Sie das Kontrollkästchen “RData beim Starten in Arbeitsbereich wiederherstellen”.
 - * Arbeitsbereich beim Beenden in .RData speichern: ***Niemals***
 - **R-Sitzungen**
 - * Deaktivieren Sie das Kontrollkästchen “Zuvor geöffnete Quelldokumente beim Start wiederherstellen”.
- Code > Anzeige
 - Allgemein
 - * Leerzeichen anzeigen
 - * Scrollen über das Ende des Dokuments hinaus zulassen
 - * Ausgewählte Zeile hervorheben
- Erscheinungsbild
 - Editor-Thema: Kobalt

tinyTex (optional)

Im weiteren Verlauf des Kurses werden wir lernen, wie man verschiedene Ausgabeformate, einschließlich PDF, erzeugt. Um PDF-Dokumente mit LaTeX unter der Haube darstellen zu können, müssen wir [tinytex](#) installieren. Es gibt verschiedene Möglichkeiten, dies zu tun:

- Führen Sie folgendes im *Terminal* aus: `quarto install tinytex`
- oder in der Konsole: `tinytex::install_tinytex()`

Sie können diesen Schritt vorerst überspringen, falls Sie Probleme haben.

Teil II.

Grundlagen

1. Einführung in R und RStudio

Heutige Ziele

- R und RStudio installieren
- in der Lage sein, Zusatzpakete zu installieren
- in der Lage sein, Hilfe für Pakete und Funktionen zu erhalten
- in der Lage sein, Objekte in der Konsole zu erstellen

Weitere Lektüre

- Dieser Vortrag basiert lose auf Kapitel 1 - *Introduction* und Kapitel 3 - *Workflow Basics* von Wickham et al. (2023)
- dieser Kurs folgt mehr oder weniger diesem Buch
- wo möglich, ersetze ich die Datensätze im Buch durch linguistische Datenbeispiele

1.1. Vorbereitung

- hoffentlich haben Sie R und RStudio bereits installiert/aktualisiert
 - falls nicht: Versuchen Sie es mit [Posit Cloud](#) für heute [posit.cloud](#)
- Gehen Sie zum [Kurs GitHub](#) und laden Sie eine ZIP-Datei des Repositorys herunter
 - große grüne Schaltfläche ‘<> Code’ > ZIP herunterladen

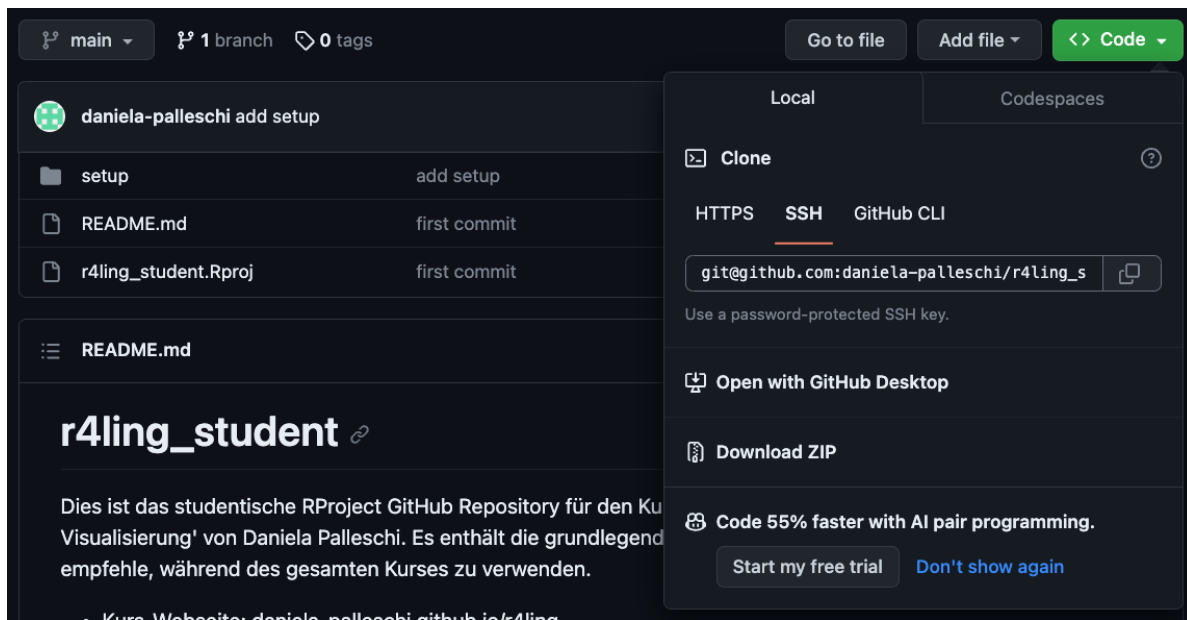


Abbildung 1.1.: Download GitHub repository

1.2. RProjekt

- Suchen Sie die ZIP-Datei, die Sie soeben heruntergeladen haben, auf Ihrem Computer und dekomprimieren Sie sie.
- Öffnen Sie den Ordner und navigieren Sie zu `r4ling_student.Rproj`, doppelklicken Sie darauf
- Sie sollten nun RStudio sehen, wie in [Abbildung 1.2](#)
- Jetzt können wir an unserem ersten Skript arbeiten

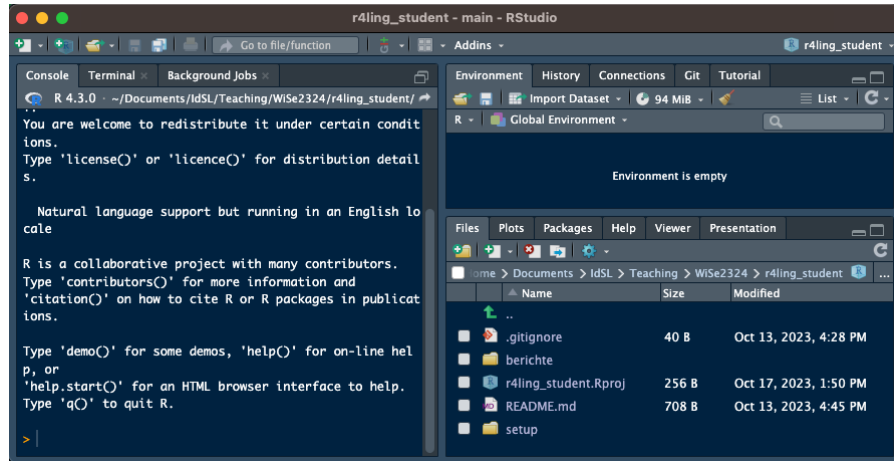


Abbildung 1.2.: Student RProject

⚠️ Warnung

Wichtig!!

Verschieben oder benennen Sie den Ordner **data/** nicht um! Sie müssen denselben Dateipfad zu den Datensätzen haben, um meinen Code in den nächsten Wochen nahtlos verwenden zu können.

1.3. R in RStudio

1. Öffnen Sie RStudio *immer* durch einen Doppelklick auf **r4ling_student.Rproj** (für diesen Kurs)
2. klicken Sie auf **File > New File > R Script**
 - sehen Sie nun vier Quadrate (statt 3 in [Abbildung 1.2](#)):
 - i. Texteditor - oben Links - wo wir unseren Code schreiben werden
 - ii. R-Konsole (EN: Console) - unten links - wo wir die Ausgabe unseres Codes und Warn-/Fehlermeldungen sehen werden
 - iii. Arbeitsumgebung (EN: Environment) - oben rechts - wo unsere Daten und Objekte nach dem Laden gespeichert werden
 - iv. Dateien und Grafikausgabe - unten links - wo wir unsere Dateien und die von uns erstellten Grafiken sehen oder Hilfe bekommen können

1.3.1. Erweiterungspakete

- R hat eine Reihe von nativen Funktionen und Datensätzen, auf die wir zugreifen können
 - ähnlich wie die Standard-Apps, die auf Ihrem Handy vorinstalliert sind
- Jeder kann Zusatzpakete für R erstellen, z.B.,
 - für Datenvisualisierung
 - Datenverarbeitung
- Dies ist ähnlich wie bei Handy-Apps, die von jedem erstellt und auf Ihr Gerät heruntergeladen werden können
 - aber Pakete sind *immer kostenlos*
- Es gibt 2 Schritte, um ein Paket zu verwenden:
 1. Installieren des Pakets (einmalig) mit `install.packages("Paket")`
 2. Laden Sie das Paket (zu Beginn jeder Sitzung) `library(Paket)`

1.3.1.1. Paket-Installation

- erfolgt mit der Funktion `install.packages()`
 - Sie machen dies nur einmal (wie das Herunterladen einer App)
- das Paket `tidyverse` ist sehr hilfreich für Datenverarbeitung und Visualisierung
 - Installieren wir es jetzt

Paket-Installation

- installieren Sie die Pakete `tidyverse` und `beepr`

```
install.packages("tidyverse")  
install.packages("beepr")
```

! Pakete in der Konsole installieren

Installieren Sie Pakete immer über die Konsole, nicht über ein Skript!
Sie können auch die Registerkarte “Pakete” in der unteren rechten Box verwenden (Pakete > Installieren)

1.3.1.2. tinytex

- wir brauchen auch LaTeX und `tinytex` (Xie, 2023), um PDF-Dokumente zu erstellen
- führen Sie diesen Code aus, um `tinytex` zu installieren

```
## run this in the console
install.packages("tinytex")
tinytex::install_tinytex()
```

- Sie müssen auch LaTeX installieren, wenn Sie es noch nicht haben: <https://www.latex-project.org/get/>

1.3.2. Laden eines Pakets

- die Funktion `library()` lädt ein Paket in Ihre Umgebung
- dies muss zu Beginn jeder Sitzung geschehen, um auf das entsprechende Paket zugreifen zu können

```
library(beep)
```

1.3.2.1. Verwendung einer Funktion

- Sobald Sie ein Paket geladen haben, können Sie auf dessen Funktionen zugreifen
- Zum Beispiel hat das Paket `beep` eine Funktion `beep()`, probieren wir sie aus

Listing 1.1 in der Konsole laufen

```
beep()
```

1.3.2.2. Funktionsargumente

- Argumente enthalten optionale Informationen, die an eine Funktion übergeben werden
 - Die Funktion `beep()` hat das Argument `sound`, das einen numerischen Wert von 1:11 annimmt.
 - Versuchen Sie, den folgenden Code mit anderen Zahlen auszuführen, was passiert?

Listing 1.2 in der Konsole laufen

```
beep(sound = 5)
```

Funktionsargumente

?help

Sie können mehr über eine Funktion (einschließlich ihrer verfügbaren Argumente) herausfinden, indem Sie ihren Namen nach einem Fragezeichen in die Konsole schreiben (z.B. `?beep`). Versuchen Sie, `?beep` auszuführen. Kannst du auf der Hilfeseite herausfinden, was du anstelle von `sound = 5` schreiben kannst, um denselben Ton zu erzeugen?

1.3.3. Aufgabe: Paket-Installation

Aufgabe

Wir brauchen auch das `here`-Paket. Installieren Sie dieses.
Nachdem Sie das Paket installiert haben, führen Sie den Befehl `here()` aus. Was geschieht?

1.4. Reproduzierbarkeit

- in diesem Kurs werden wir lernen, wie man *reproduzierbare Berichte* erstellt
 - Das bedeutet, dass unser Code später noch einmal ausgeführt werden kann und immer noch die gleichen Ergebnisse liefert
- wenn Ihre Arbeit reproduzierbar ist, können andere Leute (und Sie selbst) Ihre Arbeit verstehen und überprüfen
 - Für Kursaufgaben werden Sie Berichte sowie den Quellcode einreichen, die ich auf meinem Rechner ausführen können sollte

1.4.1. RStudio-Einstellungen

- wir wollen immer mit einem freien Arbeitsbereich in RStudio beginnen, um die Reproduzierbarkeit zu gewährleisten
 - Wir wollen auch niemals unseren Arbeitsbereich für später speichern
 - wir wollen nur unseren Code (und die Ausgabeberichte) speichern
- Gehen Sie zu Tools > Global Options
 - Deaktivieren Sie das Kontrollkästchen Restore .RData into workspace at startup
 - Setzen Sie Save workspace to .RData on exit: to Never

RStudio-Einstellungen

RStudio: Tools > Global Options:

- Restore .RData into workspace at startup
 - nein
- Save workspace to .RData on exit:
 - Never

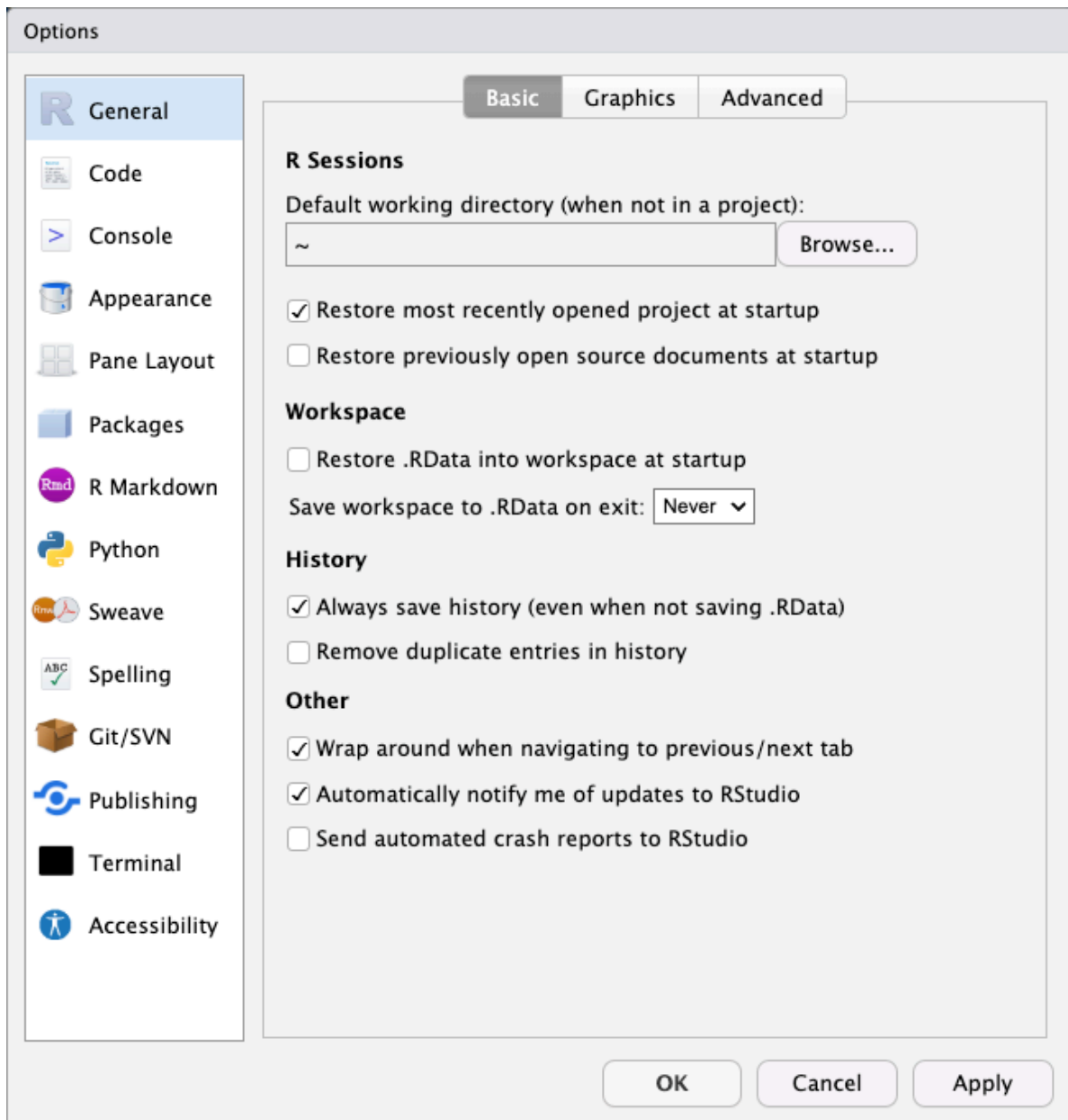


Abbildung 1.3.: Ihre ‘Global Options’ sollten wie folgt aussehen

RStudio-Einstellungen

- Klicken Sie auf **Appearance** (linke Spalte)

- Öffnen Sie die Optionen “Editor Theme” und wählen Sie ein Farbschema, das Ihnen gefällt
- Sie können auch die Schriftart/Schriftgröße ändern, wenn Sie dies wünschen

1.4.2. Aufgabe: neues R-Skript

Aufgabe

- in RStudio: File > New File > R Script
 - wenn sich oben links ein neues Fenster öffnet: “Datei > Speichern unter...”.
 - * speichern Sie es in Ihrem ‘notizen’ Ordner
 - schreiben Sie oben in das Skript: `## Angewandte Datenverarbeitung und Visualisierung - Woche 1 (17.04.2023)`

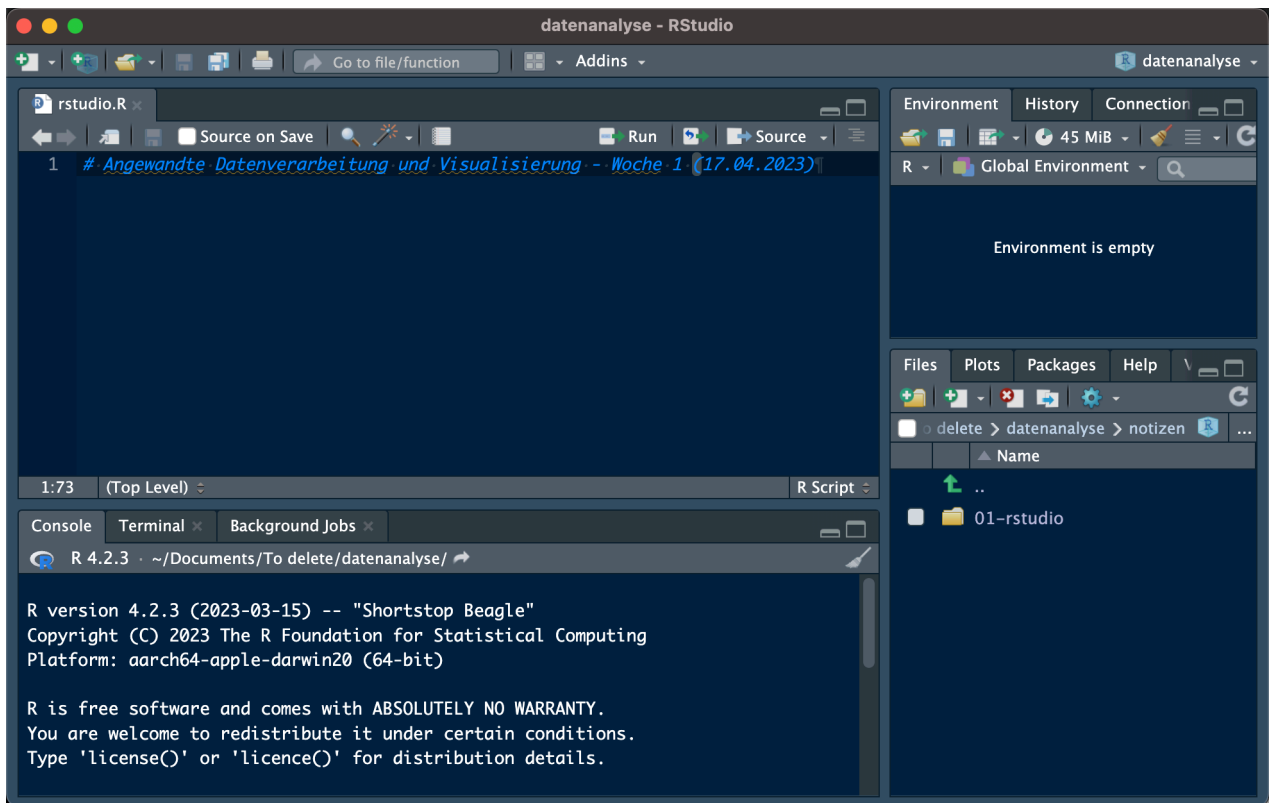


Abbildung 1.4.: Ihre Skript (oben links) sollten so aussehen

1.5. Rechnen in R

- können wir Berechnungen in R durchführen
- wir können addieren (+), subtrahieren (-), multiplizieren (*) und dividieren (/)

1.5.1. Aufgabe: Berechnungen

Aufgabe

1. Versuchen Sie, die folgenden Berechnungen in der Konsole auszuführen:

```
# Addition  
16+32
```

```
[1] 48
```

```
# Multiplikation  
16*32
```

```
[1] 512
```

```
# Subtraktion  
16-32
```

```
[1] -16
```

```
# Division  
16/32
```

```
[1] 0.5
```

2. schreiben Sie diese Berechnungen in Ihr Skript, und drücken Sie **Cmd/Strg-Enter**, um sie auszuführen

- Was passiert?

1.5.2. Kommentare

- Sie haben vielleicht bemerkt, dass in meinen Code-Blöcken z. B. `# Subtraktion` über dem Code stand
- R ignoriert jeden Text nach `#` (plus ein Leerzeichen)
- also können wir Kommentare nach `#` schreiben

```
# Kommentar zum folgenden Code  
16-32
```

[1] -16

- Wir können auch eine Abschnittsüberschrift erstellen, um unsere R-Skripte zu strukturieren, indem wir vier `#` nach einem Titel hinzufügen
- Die Struktur des Skripts kann dann durch Klicken auf die Schaltfläche “Gliederung” oberhalb des Skriptfensters angezeigt werden

```
# Rechnen mit R ####  
  
# Subtraction  
16-32
```

[1] -16

1.5.3. Objekte

- wir können auch Werte als Objekte/Variablen speichern, die in der Arbeitsumgebung gespeichert sind

```
x <- 16  
y <- 32
```

Assignment operator

Das Symbol `<-` ist ein sogenannter *assignment operator*. Es erstellt ein neues Objekt in Ihrer Arbeitsumgebung oder überschreibt ein vorhandenes Objekt mit demselben Namen. Es ist wie ein Pfeil, der sagt: “Nimm das, was rechts steht, und speichere es als den Objektnamen auf der linken Seite”.

1.5.4. Rechnen mit Funktionen

- es gibt auch eingebaute Funktionen für komplexere Berechnungen
- z.B., `mean()` (DE: Durchschnitt), `sum()` (DE: Summe)
- was passiert, wenn wir folgendes ausführen?

```
sum(6,10)
```

```
[1] 16
```

```
6+10
```

```
[1] 16
```

```
mean(6,10)
```

```
[1] 6
```

```
(6+10)/2
```

```
[1] 8
```

Rechnen mit Funktionen

- die Funktion `mean()` nimmt nur ein Argument an; alles andere wird ignoriert
 - das Komma in `6,10` listet 2 Argumente auf, also wird alles nach dem Komma ignoriert
- wenn wir mehr als ein Objekt in ein Argument einschließen wollen, müssen wir die “concatenate”-Funktion `c()` verwenden
 - “concatenate” bedeutet zusammenfügen oder kombinieren

```
mean(c(6,10))
```

```
[1] 8
```

Rechnen mit Funktionen

- Sie können auch benannte Objekte (d.h. die in Ihrer Arbeitsumgebung) verwenden, die einen numerischen Wert haben

Aufgabe: Rechnen mit Funktionen

1. Versuchen Sie, die Funktion `mean()` mit Ihren gespeicherten Variablen (`x` und `y`) als “verkettete” Argumente auszuführen
2. Machen Sie dasselbe mit der Funktion `sum()`. Was passiert, wenn Sie `c()` nicht verwenden?

1.6. Vektoren

- Vektoren sind eine Liste von Elementen desselben Typs (z. B. numerisch, Zeichenkette)
- wir können einen Vektor mit der Verkettungsfunktion `c()` erstellen
- Der folgende Code speichert in einem Objekt namens ‘vec’ einen Vektor aus mehreren Zahlen

```
# einen Vektor erstellen
vec <- c(171, 164, 186, 191)
```

- der folgende Code ruft das Objekt auf, das wir als ‘vec’ gespeichert haben, und gibt seinen Inhalt aus

```
# print vec
vec
```

```
[1] 171 164 186 191
```

1.6.1. Arithmetic mit Vektoren

- Grundlegende Arithmetik auf Vektoren wird auf jedes Element angewendet

```
# add 5 to vec
vec + 5
```

```
[1] 176 169 191 196
```

- können wir auch Funktionen auf Vektoren anwenden

```
# Summe von vec  
sum(vec)
```

```
[1] 712
```

```
# Mittelwert von vec  
mean(vec)
```

```
[1] 178
```

```
# Quadratwurzel aus vec  
sqrt(vec)
```

```
[1] 13.07670 12.80625 13.63818 13.82027
```

1.6.2. Ausgabe: Vektoren

Ausgabe

1. Erstelle einen Vektor namens **vec1**, der die Werte 12, 183, 56, 25 und 18 enthält
2. Erstellen Sie einen Vektor namens **vec2**, der die Werte 8, 5, 1, 6 und 8 enthält
3. Create a vector called **vec3** that contains the values 28, 54, 10, 13, 2, and 81
4. Finde die Summe von **vec1**.
5. Finde die Summe von **vec1** plus **vec2**. Wie unterscheidet sich das Ergebnis von dem, das Sie für **vec1** allein erhalten haben?
6. Was passiert, wenn du versuchst, die Summe von **vec1** und **vec3** zu finden?

1.7. Endergebnis

- Speichern Sie Ihr R-Skript (**File > Save**, oder **Cmd/Strg-S**)
- Sie sollten nun einen *RProject-Ordner* für diesen Kurs, der Folgendes enthält:
 - **r4ling_student.RProj**
 - einen Ordner namens **Daten**
 - einen Ordner namens **notes**, der Folgendes enthält + eine **.R**-Datei mit der heutigen Arbeit

- Sie wissen jetzt, wie man
 - *einfache Berechnungen* in R durchführen
 - *Objekte* in Ihrer Arbeitsumgebung zu speichern
 - einfache mathematische Berechnungen mit Ihren gespeicherten Objekten durchführen

1.8. Session Info

- Um die Reproduzierbarkeit zu verbessern, ist es nützlich, die Version von R, RStudio und die verwendeten Pakete zu verfolgen
 - Zu diesem Zweck können Sie die folgenden Befehle ausführen:

```
## R version
R.version.string
```

```
[1] "R version 4.3.0 (2023-04-21)"
```

```
## R version name
R.version$nickname
```

```
[1] "Already Tomorrow"
```

```
## RStudio version
RStudio.Version()$version
## RStudio version name
RStudio.Version()$release_name
```

```
## alle Paketversionen
sessionInfo()
```

1.9. Nächste Woche

vor nächster Woche, stellen Sie bitte sicher, dass Sie:

- R und RStudio installiert/aktualisiert haben
- die Pakete tidyverse und here installiert haben

- bitte stellen Sie sicher, dass Sie die Übungen des heutigen Kurses in Ihrem R-Skript durcharbeiten
- (optional) speichern Sie das Skript, und laden Sie es auf Moodle hoch, wenn Sie es auf Ihre 6 Skripte für die Teilnahme-LP anrechnen lassen möchten

Session Info

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

```
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;

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 utils datasets methods base

other attached packages:

[1] beepR_1.3 magick_2.7.4

loaded via a namespace (and not attached):

[1] digest_0.6.33 fastmap_1.1.1 xfun_0.39 magrittr_2.0.3
 [5] glue_1.6.2 stringr_1.5.0 audio_0.1-10 knitr_1.44
 [9] htmltools_0.5.5 png_0.1-8 rmarkdown_2.22 lifecycle_1.0.3
 [13] cli_3.6.1 compiler_4.3.0 rprojroot_2.0.3 here_1.0.1
 [17] rstudioapi_0.14 tools_4.3.0 evaluate_0.21 Rcpp_1.0.11
 [21] yaml_2.3.7 rlang_1.1.1 jsonlite_1.8.7 stringi_1.7.12

Literaturverzeichnis

- Baayen, R. H. (2008). *Analyzing Linguistic Data: A Practical Introduction to Statistics Using R*.
- Baayen, R. H., & Shafaei-Bajestan, E. (2019). *languageR: Analyzing Linguistic Data: A Practical Introduction to Statistics*. <https://CRAN.R-project.org/package=languageR>
- Müller, K. (2020). *here: A Simpler Way to Find Your Files*. <https://CRAN.R-project.org/package=here>
- Nordmann, E., & DeBruine, L. (2022). *Applied Data Skills*. Zenodo. <https://doi.org/10.5281/zenodo.6365078>
- 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., Averick, M., Bryan, J., Chang, W., McGowan, L. D., François, R., Grolemund, G., Hayes, A., Henry, L., Hester, J., Kuhn, M., Pedersen, T. L., Miller, E., Bache, S. M., Müller, K., Ooms, J., Robinson, D., Seidel, D. P., Spinu, V., ... Yutani, H. (2019). Welcome to the tidyverse. *Journal of Open Source Software*, 4(43), 1686. <https://doi.org/10.21105/joss.01686>
- Wickham, H., Çetinkaya-Rundel, M., & Grolemund, G. (2023). *R for Data Science* (2. Aufl.).
- Winter, B. (2019). Statistics for Linguists: An Introduction Using R. In *Statistics for Linguists: An Introduction Using R*. Routledge. <https://doi.org/10.4324/9781315165547>
- Xie, Y. (2023). *tinytex: Helper Functions to Install and Maintain TeX Live, and Compile LaTeX Documents*. <https://github.com/rstudio/tinytex>

2. Datenvisualisierung 1

Visualisierung von Verteilungen

Wiederholung

Letzte Woche haben wir...

- R und RStudio installiert
- unser erstes R-Skript erstellt
- einfache Arithmetik mit Objekten und Vektoren durchgeführt

Wiederholung

```
x <- c(1,2,3)
y <- sum(1,2,3)
```

- Was enthalten die Vektoren `x` und `y`?
- Das Objekt `x` enthält 1, 2, 3
- Das Objekt `y` enthält ‘ 6 ‘

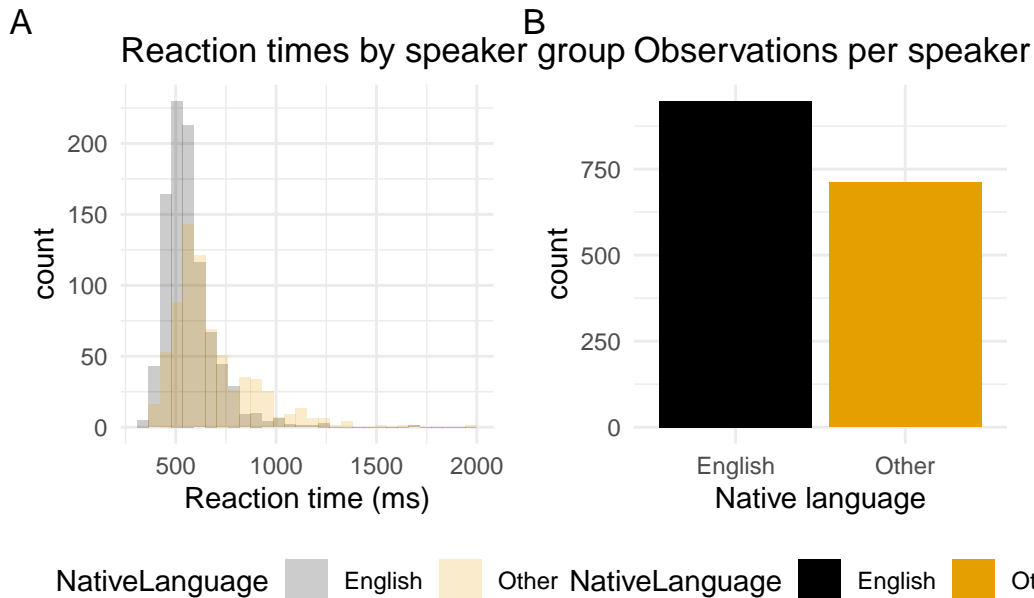
Heutige Ziele

Heute werden wir lernen...

- was Datenframes sind
- den Unterschied zwischen kategorialen und kontinuierlichen Daten
- wie man Diagramme mit `ggplot` erstellt
- die richtige Darstellung für unsere Daten auszuwählen

Endgültiges Ziel

- Unser heutiges Ziel ist es, die Daten wie folgt zu visualisieren
 - Das Diagramm zeigt die Verteilung (Anzahl) der Reaktionszeiten und der Muttersprache der Teilnehmer



Lust auf mehr?

- Kapitel 2 (Datenvisualisierung) in Wickham et al. (2023), bis zum Abschnitt 2.4
- Kapitel 3 (Datenvisualisierung) in Nordmann & DeBruine (2022)

Vorbereitung

In Ihrem RProject-Ordner...

- erstellen Sie einen neuen Ordner mit dem Namen `moodle`
 - Laden Sie die Moodle-Materialien von heute herunter und speichern Sie sie dort
- Erstellen Sie einen neuen Ordner in `notes` mit dem Namen `02-datenviz1`
- öffne ein neues `.R` Skript
 - speichere es in dem neuen Ordner

2.0.0.1. Pakete

- Pakete laden (und installieren)

- tidyverse
- languageR
- ggthemes
- patchwork

```
## in the CONSOLE: install packages if needed  
install.packages("tidyverse")  
install.packages("languageR")  
install.packages("ggthemes") ## for customising our plots  
install.packages("patchwork") ## plot layouts
```

```
## Pakete laden  
library(tidyverse)  
library(languageR)  
library(ggthemes)  
library(patchwork)
```

2.1. Datenrahmen

- Datenrahmen sind eine Sammlung von Variablen, wobei
 - jede Variable eine Spalte ist
 - jede Zeile eine einzelne Beobachtung/ein einzelner Datenpunkt ist
 - jede Zelle in einer Zeile verknüpft ist
- Datenrahmen sind genau wie Tabellenkalkulationen, aber rechteckig
- Verschiedene Wörter für Datenrahmen:
 - Datenrahmen
 - Datensatz
 - Tibble (im tidyverse)

2.1.1. Sprechen über Datensätze

- eine **Variable**: eine Menge, Qualität oder Eigenschaft, die man messen kann
- ein **Wert**: der Zustand einer Variablen, wenn man sie misst

- eine **Beobachtung**: eine Reihe von Messungen, die unter ähnlichen Bedingungen durchgeführt werden
 - enthält mehrere Werte, die jeweils mit einer Variablen verbunden sind
 - eine Beobachtung für eine einzelne Variable wird manchmal als *Datenpunkt* bezeichnet
- **Tabellendaten** sind eine Reihe von Werten, die jeweils mit einer Variablen und einer Beobachtung verbunden sind
 - Tabellarische Daten sind “tidy”, wenn jeder Wert in einer eigenen *Zelle*, jede Variable in einer eigenen Spalte und jede Beobachtung in einer eigenen Zeile steht

2.1.2. Kategoriale und kontinuierliche Variablen

- Wie wir die Verteilung einer Variablen darstellen, hängt davon ab, welche Art von Daten sie repräsentiert: *kategorisch* oder *numerisch*
- Eine Variable ist *kategorisch*, wenn sie eine kleine Menge von Werten annehmen kann, die sich in Gruppen zusammenfassen lassen
 - z. B. alt/jung, klein/groß, grammatikalisch/ungrammatikalisch, L1/L2-Sprecher
- eine Variable ist *numerisch* (d. h. quantitativ), wenn sie eine große Bandbreite an numerischen Werten annehmen kann
 - und es sinnvoll wäre, zu addieren, zu subtrahieren, den Mittelwert zu berechnen usw.
 - kann *kontinuierlich* sein (Dezimalpunkte sind sinnvoll, z. B. 1,5 cm)
 - oder *diskret* (Dezimalpunkte sind *nicht* sinnvoll, z. B. 1,5 Kinder sind nicht sinnvoll)
- wir erstellen verschiedene Diagramme, je nachdem, welche Art von Variablen wir visualisieren wollen

2.2. Lexical Decision Task (LDT)

- unser erster Datensatz enthält Daten aus einer lexikalischen Entscheidungsaufgabe
- Bei der LDT drücken die Teilnehmer eine Taste, um anzugeben, ob ein Wort ein echtes Wort oder ein Pseudowort ist.

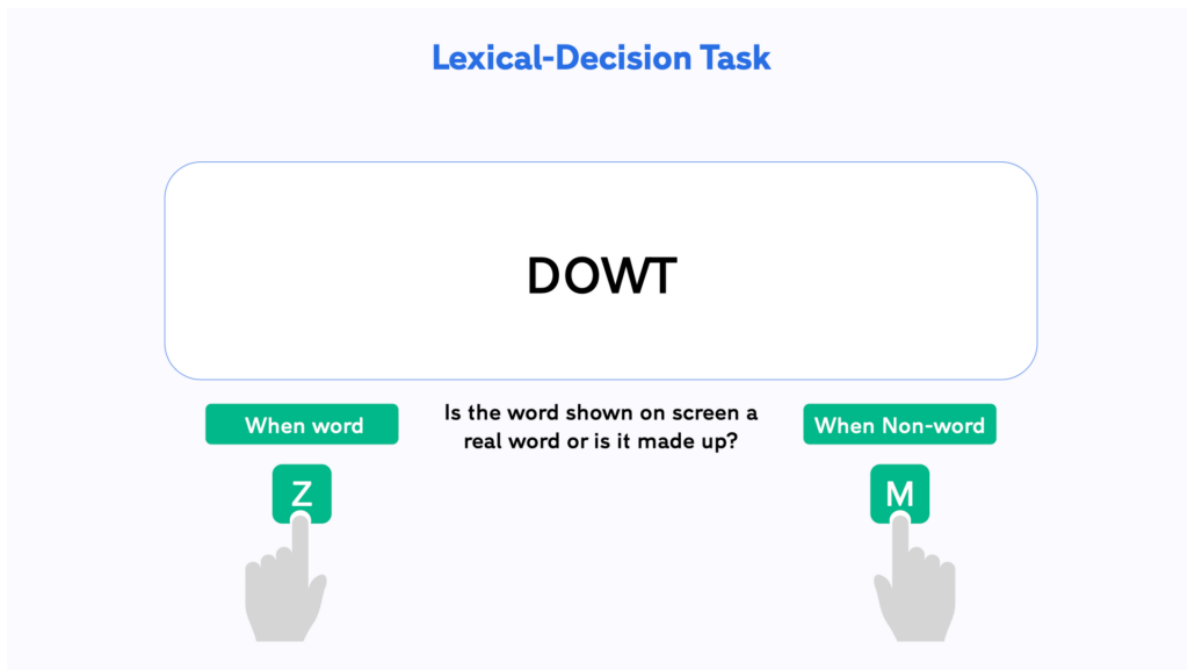


Abbildung 2.1.: Source: https://www.testable.org/wp-content/uploads/2022/11/Lexical_decision_task-1024x576.png

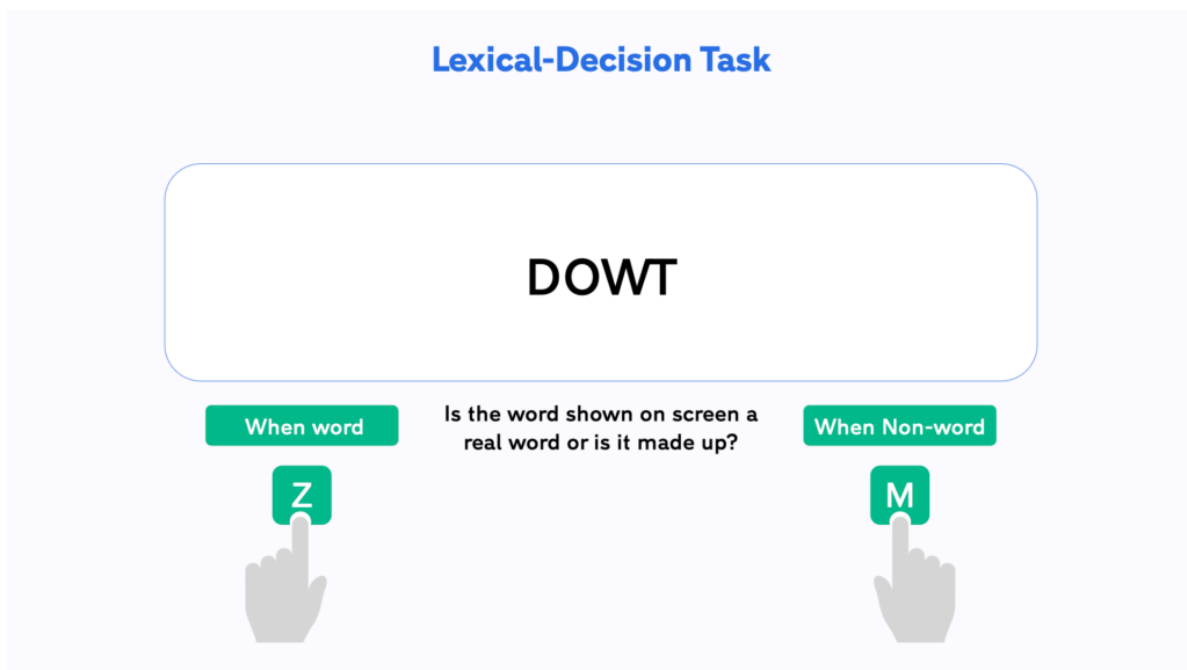


Tabelle 2.1.: Datenwörterbuch für `df_lexdec`: Lexikalische Entscheidungszeiten, die von 21 Probanden für 79 konkrete englische Substantive erhoben wurden, mit Variablen, die mit dem Subjekt oder dem Wort verknüpft sind.

Variable	Beschreibung
Subject	ein Faktor für die Probanden
RT	ein numerischer Vektor für die Reaktionszeit in Millisekunden
Trial	ein numerischer Vektor für den Rang des Versuchs in der Versuchsliste
Sex	ein Faktor mit den Ausprägungen F (weiblich) und M (männlich)
NativeLanguage	ein Faktor mit den Niveaus English und Other, der zwischen englischen Muttersprachlern und

2.2.1. LDT-Variablen

- Die üblichen Variablen, die in einem Experiment zur lexikalischen Entscheidungsaufgabe erhoben werden, sind:
 - Reaktionszeit
 - Genauigkeit (richtig/falsch)
 - Wortkategorie (z. B. real/pseudo, Nomen/Verb)
 - Worthäufigkeit
- Zusätzliche Variablen, die erhoben werden könnten, sind:
 - demografische Daten der Teilnehmer (z. B. Alter, L1/L2, Geschlecht)

2.3. lexdec Datensatz

- `languageR` ist ein Begleitpaket für das Lehrbuch Baayen (2008)
 - enthält linguistische Datensätze, z.B. `lexdec`.
- der `lexdec`-Datensatz enthält Daten für eine lexikalische Entscheidungsaufgabe im Englischen
 - wir werden mit Variablen wie Reaktionszeiten und Genauigkeit arbeiten

2.3.1. lexdec-Variablen

- eine Liste einiger der Variablen ist in Tabelle 2.1 enthalten

2.3.2. LDT-Forschungsfragen

- bevor wir ein Experiment durchführen, haben wir Forschungsfragen, die wir mit den Daten beantworten wollen
 - Wir werden uns heute mit der folgenden Frage beschäftigen:
 - * Unterscheiden sich die Reaktionszeiten zwischen Muttersprachlern und Nicht-Muttersprachlern?

2.3.3. Laden der Daten

- unsere Daten sind in dem Paket `lanaugeR` verfügbar, das wir bereits geladen haben
 - um die Daten zu drucken, geben Sie einfach den Namen des Datensatzes ein und führen Sie ihn aus
- Unten sehen wir nur ein paar Variablen, aber Sie sollten mehr in Ihrer Konsole sehen

```
lexdec
```

	Subject	RT	Trial	Sex	NativeLanguage	Correct	PrevType	PrevCorrect
1	A1	6.340359	23	F	English	correct	word	correct
2	A1	6.308098	27	F	English	correct	nonword	correct
3	A1	6.349139	29	F	English	correct	nonword	correct
4	A1	6.186209	30	F	English	correct	word	correct
5	A1	6.025866	32	F	English	correct	nonword	correct
6	A1	6.180017	33	F	English	correct	word	correct

- Wie viele Variablen haben wir? Beobachtungen?

2.3.3.1. Daten als Objekt speichern

- Um die Daten in unserer Umgebung zu speichern, müssen wir ihnen einen Namen zuweisen
 - Nennen wir es `df_lexdec`, was soviel bedeutet wie “Datenrahmen lexikalische Entscheidung”.

```
df_lexdec <- lexdec
```

- jetzt sehen wir es in unserem Environment
 - Doppelklicken Sie darauf, um es im Editorfenster zu sehen.

2.3.4. Relevante Variablen

- Zu den Variablen, die wir haben, gehören:
 1. **Subjekt**: Teilnehmer-ID
 2. **RT**: protokollierte Reaktionszeiten
 3. **NativeLanguage**: die Muttersprache des Teilnehmers
 4. **Word**: welches Wort präsentiert wurde
 5. **Class**: ob das Wort ein Tier oder eine Pflanze war

💡 Aufgabe 2.1: ?lexdec

Beispiel 2.1.

Um herauszufinden, wofür die anderen Variablen stehen, führen Sie ?lexdec in der Konsole aus.

2.4. Erstellen von Plots mit ggplot2

- das tidyverse ist eine Sammlung von Paketen, die das Aufräumen und die Visualisierung von Daten erleichtern
 - wenn wir tidyverse laden, wird diese Sammlung von Paketen automatisch geladen
- das ggplot2-Paket ist ein tidyverse-Paket, das Plots in Schichten aufbaut

ggplot2 Schichten

2.4.1. Ebene 1: leere Leinwand

- die erste Ebene mit der Funktion ggplot() ist wie eine leere Leinwand

```
ggplot(data = df_lexdec)
```

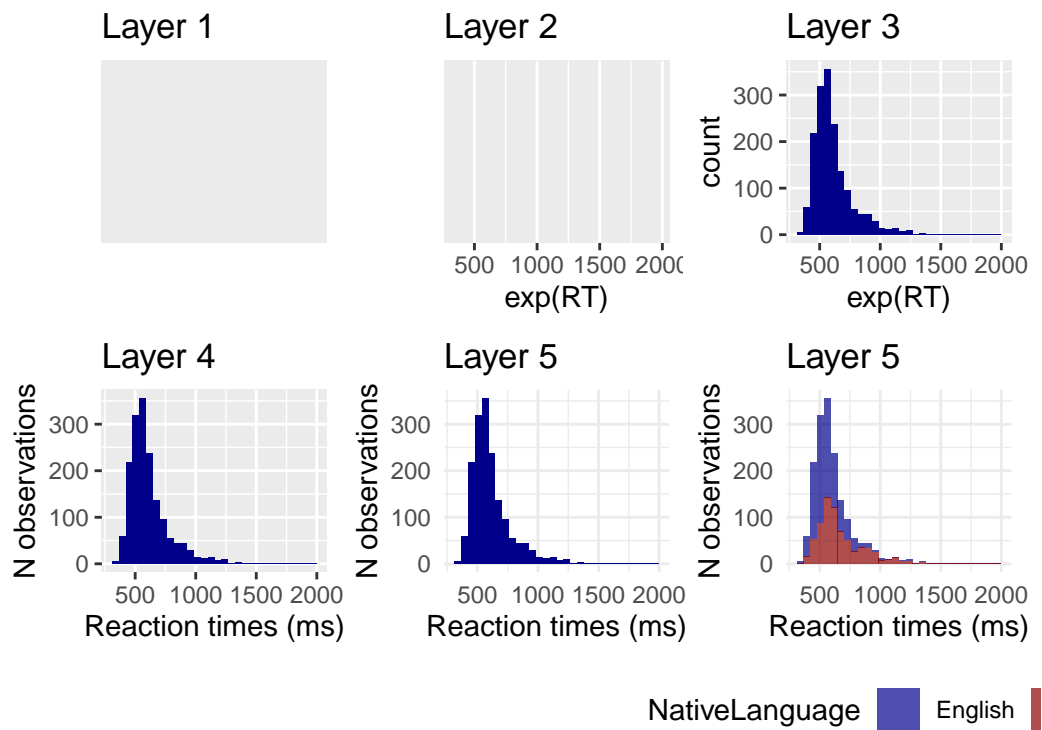
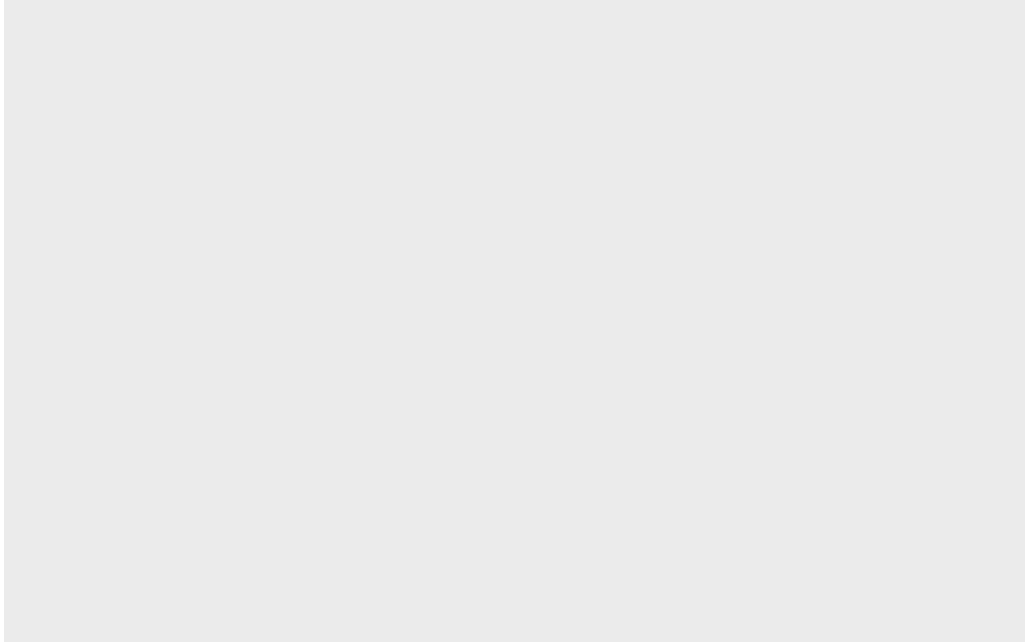


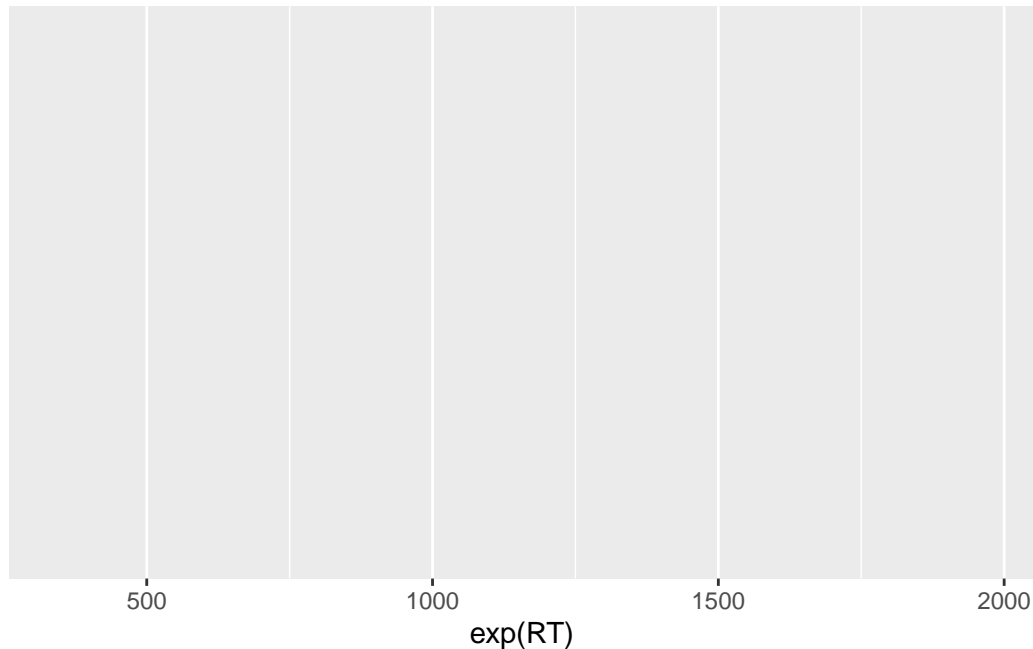
Abbildung 2.2.: Example of layers in a ggplot figure



2.4.2. Ebene 2: Ästhetik der Darstellung

- als nächstes teilen wir `ggplot()` mit, wie unsere Variablen visuell dargestellt werden sollen
 - Wir fügen das “+” am Ende unserer Codezeile ein und verwenden in einer neuen Codezeile die Funktion “`aes()`”, um unsere *Ästhetik* zu definieren.
- Unsere erste Ästhetik bildet die Reaktionszeiten (RT) auf der x-Achse ab (der untere Teil der Grafik)
 - wir wickeln die protokollierte RT in die Funktion `exp()` ein, um RTs in Millisekunden zu erhalten (aus Gründen, die wir nicht diskutieren werden)

```
ggplot(data = df_lexdec) +  
  aes(x = exp(RT))
```

💡 Aufgabe 2.2: Ästhetische Kartierung

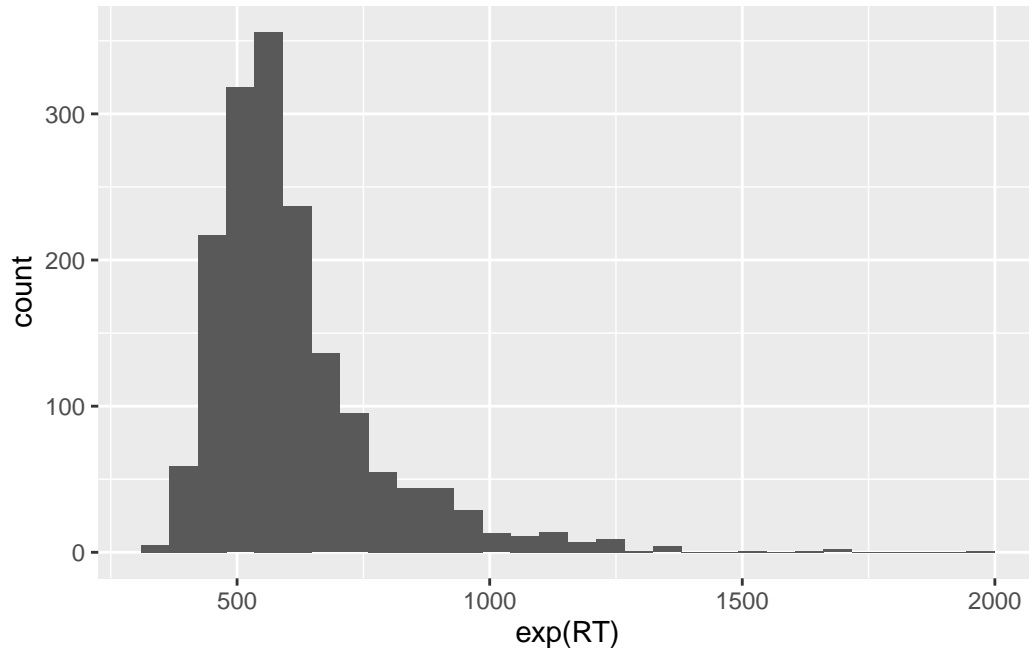
Beispiel 2.2.

Add the x-axis aesthetic.

2.4.3. Schicht 3: Hinzufügen von Beobachtungen

- wir sehen keine Beobachtungen (d.h. die Balken) in der Grafik, warum nicht?
 - wir haben `ggplot()` nicht gesagt, wie sie dargestellt werden sollen
- wir müssen ein **Geom** definieren: das *geometrische* Objekt, das ein Diagramm verwendet, um Daten darzustellen
 - in `ggplot2` beginnen die Geom-Funktionen mit `geom_`
 - wir beschreiben Diagramme oft in Bezug auf die Arten von Geomen, die sie verwenden, z.B. verwenden Balkendiagramme Balkengeome (`geom_bar()`), Liniendiagramme Liniengeome (`geom_line()`), Punktdiagramme ein Punktgeom (`geom_point()`), usw.
- Erzeugen wir unser Histogramm mit dem Geom `geom_histogram()`

```
ggplot(data = df_lexdec) +
  aes(x = exp(RT)) +
  geom_histogram()
```



i Hinweis

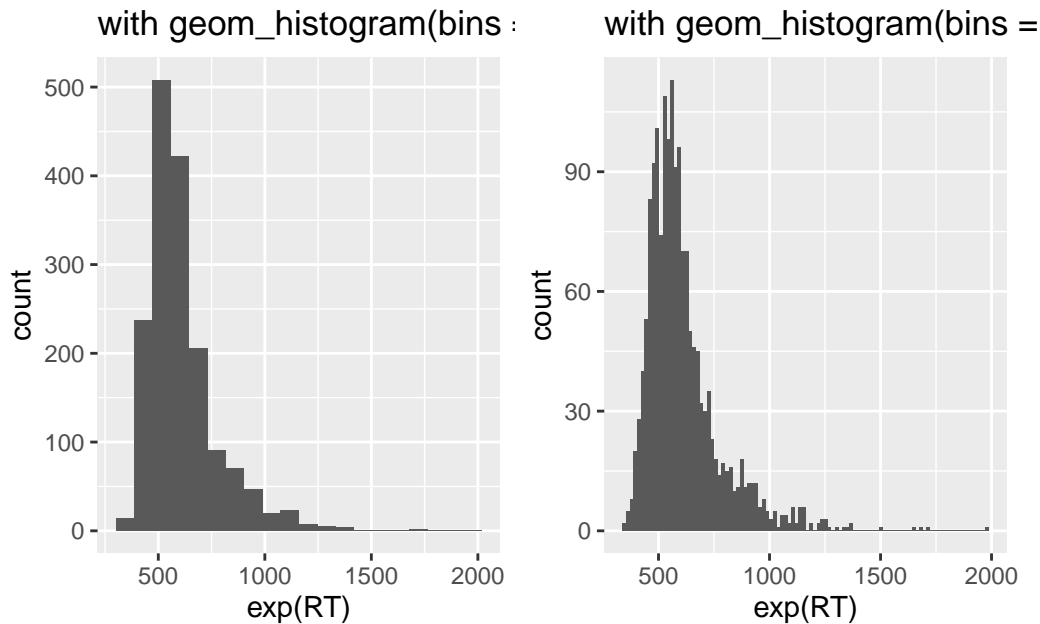
Wir erhielten die folgende Meldung, als wir `geom_point()` einschlossen:

`stat_bin()` mit `bins = 30`. Wählen Sie einen besseren Wert mit `binwidth`.

Dies sagt uns nur etwas über die Breite unserer Balken: jeder Balken repräsentiert einen Bereich möglicher Reaktionszeitwerte + `bins = 30` bedeutet einfach, dass es 30 Balken gibt, wir können dies ändern und mehr oder weniger Balken haben, indem wir z.B. `bins = 20` oder `bins = 100` in `geom_histogram()` einfügen

```
ggplot(
  data = df_lexdec,
  mapping = aes(x = exp(RT))
) +
  labs(title = "with geom_histogram(bins = 20)") +
  geom_histogram(bins = 20) +
```

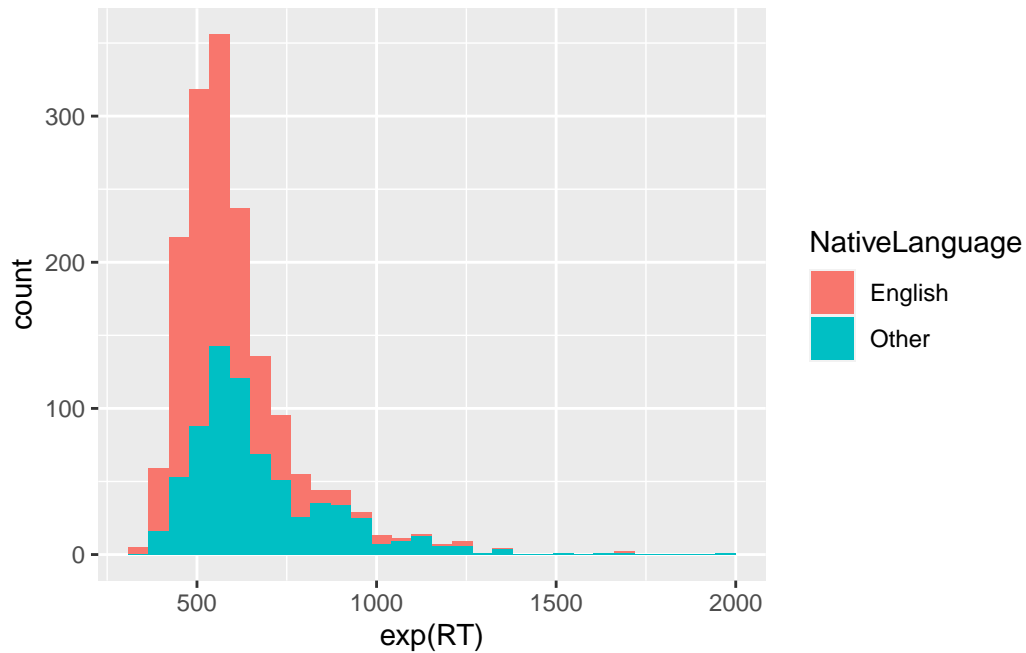
```
ggplot(
  data = df_lexdec,
  mapping = aes(x = exp(RT))
) +
  labs(title = "with geom_histogram(bins = 100)") +
  geom_histogram(bins = 100)
```



2.4.4. Hinzufügen von Ästhetik

- Es ist nützlich, die Verteilung der Reaktionszeiten im Allgemeinen zu sehen.
 - aber wir wollen normalerweise Gruppen vergleichen
 - z. B. Unterschiede zwischen Muttersprachlern und Nicht-Muttersprachlern oder zwischen verschiedenen Wortarten
- Wir haben auch die Muttersprache als Variable, wie könnten wir diese in unserem Diagramm visualisieren?

```
ggplot(
  data = df_lexdec,
  aes(x = exp(RT), fill = NativeLanguage)
) +
  geom_histogram()
```



- wir sehen die roten und die blauen Balken, aber ist das blaue Histogramm über das rote geschichtet?
– oder sind die roten Balken über den blauen Balken gestapelt?
- Es ist letzteres
– stellen wir es so ein, dass das blaue Histogramm über dem roten liegt

```
ggplot(
  data = df_lexdec,
  aes(x = exp(RT))
) +
  labs(title = "No grouping") +
  geom_histogram() +

ggplot(
  data = df_lexdec,
  aes(x = exp(RT), fill = NativeLanguage)
) +
  labs(title = "Stacked") +
  geom_histogram() +

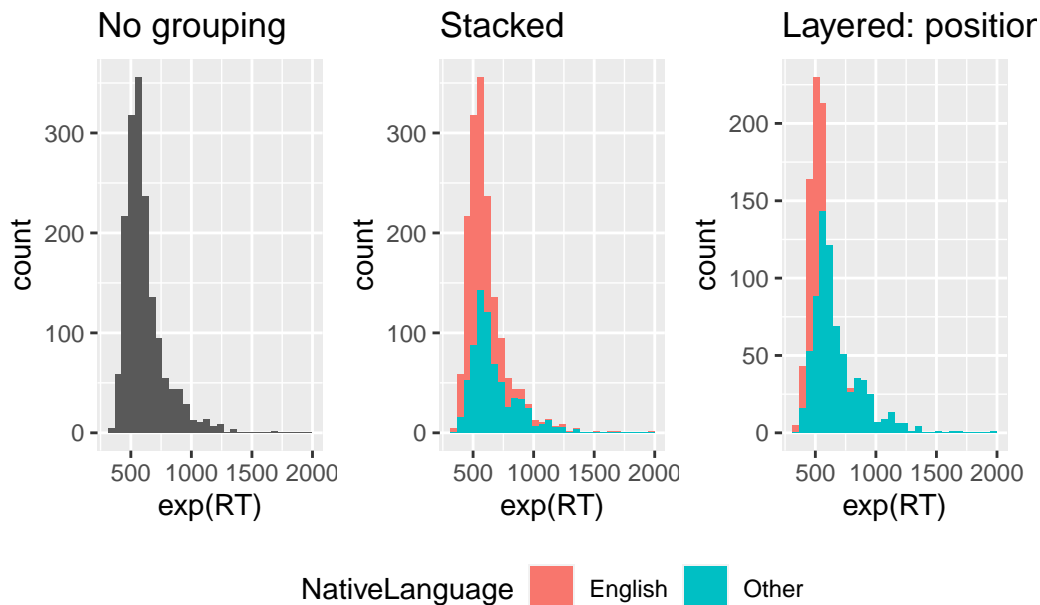
ggplot(
  data = df_lexdec,
```

```

aes(x = exp(RT), fill = NativeLanguage)
) +
labs(title = "Layered: position = \"identity\"",
geom_histogram(position = "identity") +

plot_layout(guides = "collect") & theme(legend.position = 'bottom')

```



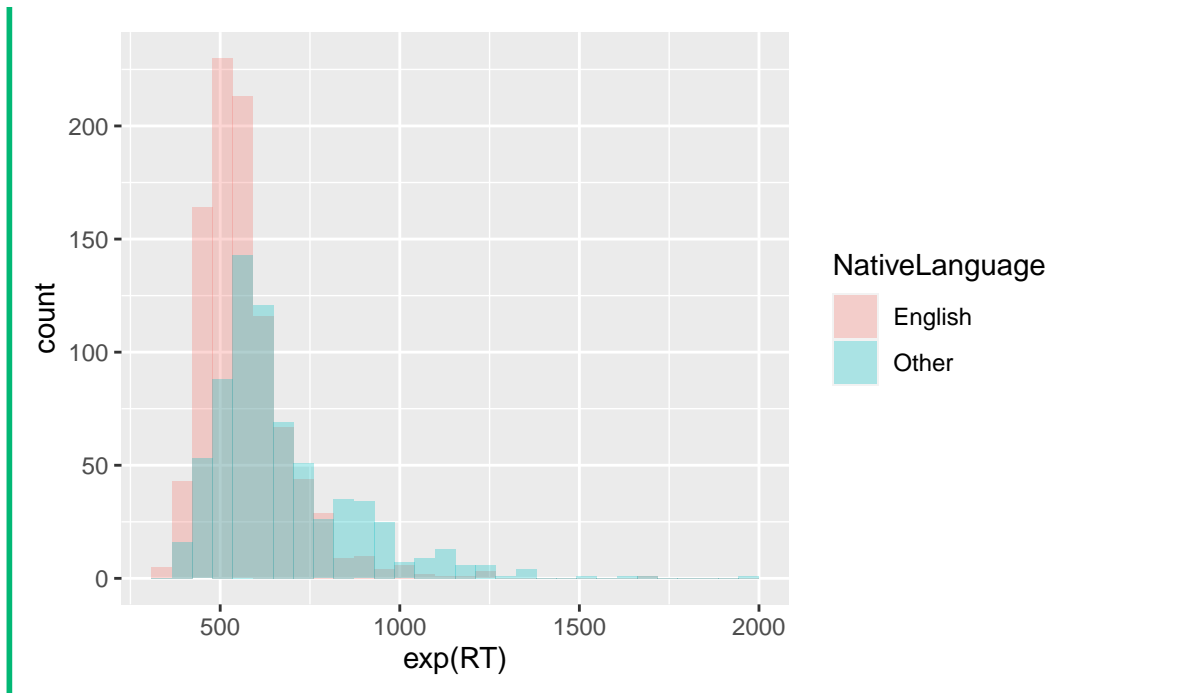
2.4.5. Globale und lokale Ästhetik

- in unserer endgültigen Darstellung ist die Farbe der Histogramme leicht transparent
 - Wir können dies steuern, indem wir das Argument `alpha = 0.3` zu `geom_histogram()` hinzufügen.
 - alpha kann jeden anderen Wert zwischen 0 und 1 annehmen.

💡 Aufgabe 2.3: Transparenz

Beispiel 2.3.

Spielen Sie mit der Transparenz des Histogramms `geom`. Wählen Sie den von Ihnen bevorzugten Alpha-Wert. Die Ausgabe sollte in etwa so aussehen:



2.4.6. Anpassen unseres Plots

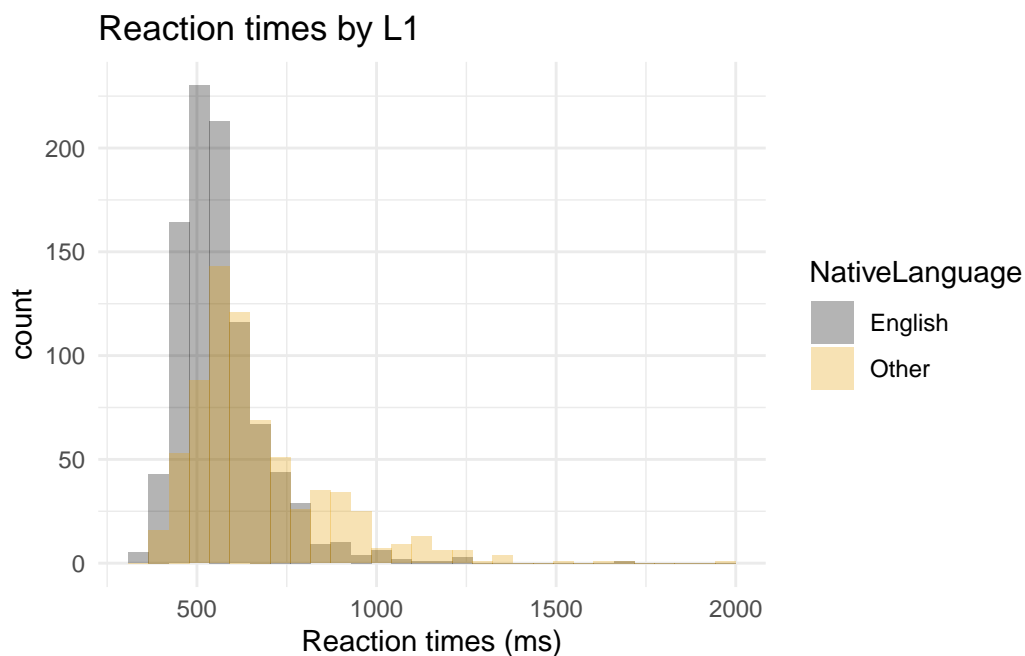
- wir können unsere Achsen- und Legendenbeschriftungen verbessern und auch Titel hinzufügen, indem wir die Funktion `labs()` verwenden
- Wir können auch die Funktion `scale_fill_colorblind()` aus dem Paket `ggthemes` verwenden.
 - dies erzeugt farbenblind-sichere Farben
- Wir werden auch die Funktion `theme_minimal()` aus dem Paket `ggplot2` verwenden; was bewirkt diese Funktion?
- Versuchen Sie, Ihrem Diagramm Folgendes hinzuzufügen
 - Ändern Sie die Beschriftungen entsprechend
 - und fügen Sie dem Code sinnvolle Kommentare mit `#` hinzu

```
labs(title = "Plot title",
     x = "x-axis label",
     y = "y-axis label") +
scale_fill_colourblind() +
theme_minimal()
```

2.4.7. Kommentar

- Der Code und die Darstellung sollten in etwa so aussehen:

```
## histogram of reaction times by native language
ggplot(data = df_lexdec) +
  aes(x = exp(RT), fill = NativeLanguage) + ## set aesthetics
  labs(title = "Reaction times by L1",
       x = "Reaction times (ms)") +
  geom_histogram(position = "identity", alpha = 0.3) +
  scale_fill_colorblind() + ## make fill colorblind friendly
  theme_minimal() ## set plot theme
```



2.4.8. Speichern von Plots

- Wir können Diagramme in unserer Umgebung speichern, genau wie wir Zahlen und Daten als Objekte speichern können.
 - Sie können Objekte beliebig benennen
 - aber es ist ratsam, den Namen sinnvoll zu gestalten (z.B. *nicht* fig1 oder xyz)
- Nennen wir diese Grafik `fig_lexdec_rt`, für “figure lexical decision task reaction times”.

💡 Aufgabe 2.4: Figur als Objekt speichern

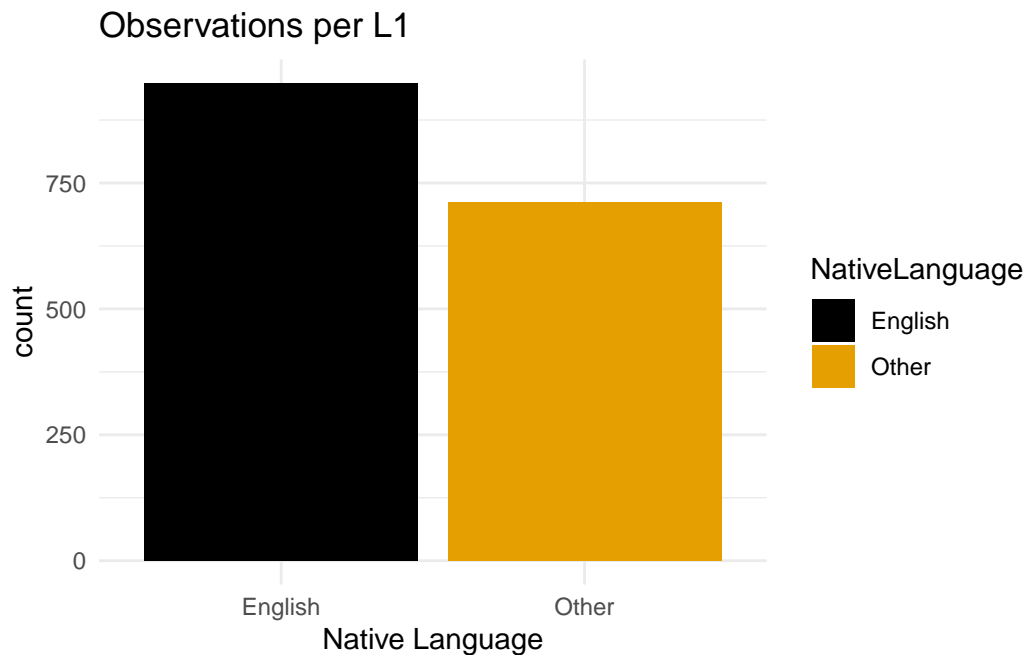
Beispiel 2.4.

1. Speichern Sie unsere endgültige Darstellung als Objekt mit dem Namen `fig_lexdec_rt`.

2.4.9. Balkendiagramme

1. Kopieren Sie den Code für Ihr Histogramm
2. Nehmen Sie die folgenden Änderungen vor, um unser Balkendiagramm darzustellen
 - Entfernen Sie die Namenszuweisung (`fig_lexdec_rt`)
 - auf der x-Achse wollen wir `NativeLanguage`
 - Ersetzen Sie `geom_histogram()` durch `geom_bar()`
 - Entfernen Sie die Argumente für das Histogramm (kein `position` oder `alpha`)
 - ändern Sie die Beschriftungen entsprechend
3. Speichern Sie das Diagramm als Objekt mit einem aussagekräftigen Namen (z.B. `fig_lexdec_l1`)

- sollte das Diagramm in etwa so aussehen:

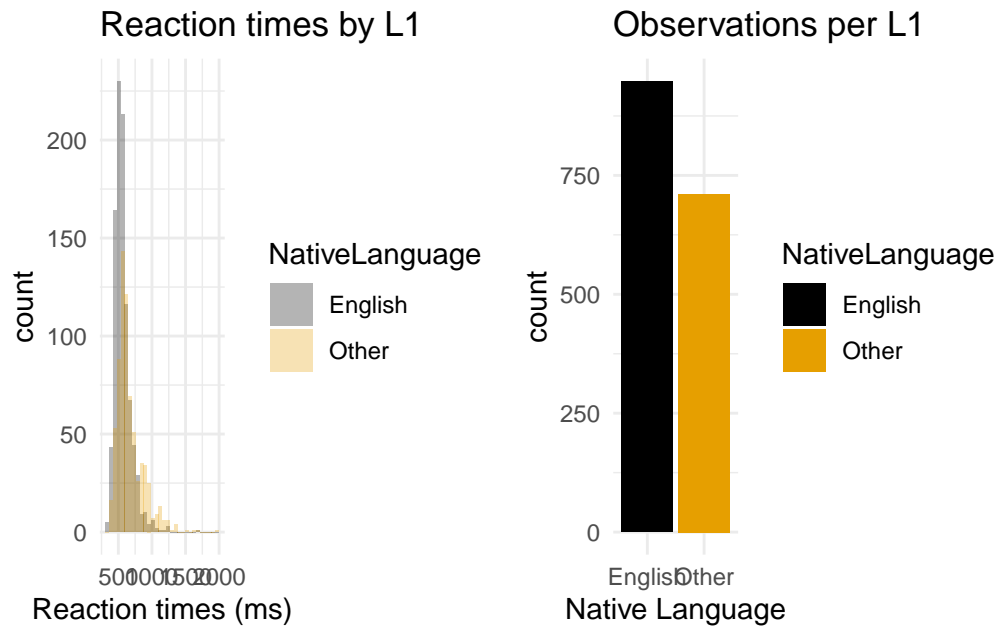


2.4.10. Kombinieren von Plots

- Ein Grund, Ihre Darstellung als Objekt zu speichern, ist, dass wir sie später aufrufen können
 - d.h. Sie können den Plot an einer Stelle in Ihrem Dokument erstellen, sich aber entscheiden, ihn erst im gerenderten Bericht weiter unten zu drucken
- ein weiterer Grund ist, dass wir mehrere Diagramme kombinieren können
 - Dies kann mit einer Vielzahl von Paketen geschehen
 - Versuchen wir es mit dem Paket **patchwork**
 - * Benutze `+` um zwei Plots nebeneinander zu verbinden
 - * oder `/`, um sie übereinander darzustellen

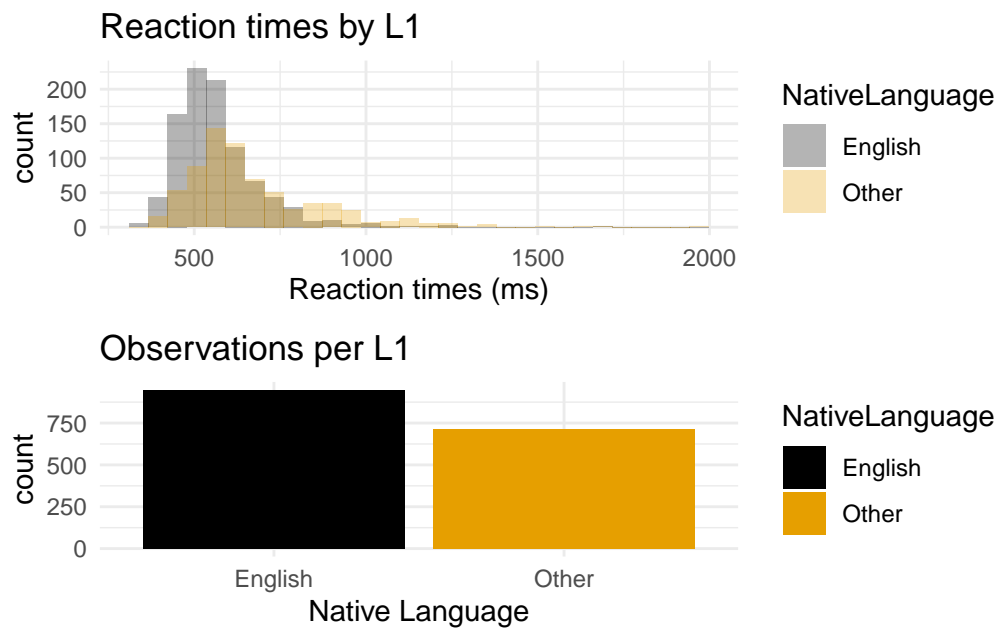
2.4.10.1. Kombinieren von Plots mit `+`

```
fig_lexdec_rt + fig_lexdec_l1
```



2.4.10.2. Kombinieren von Plots mit /

```
fig_lexdec_rt / fig_lexdec_l1
```



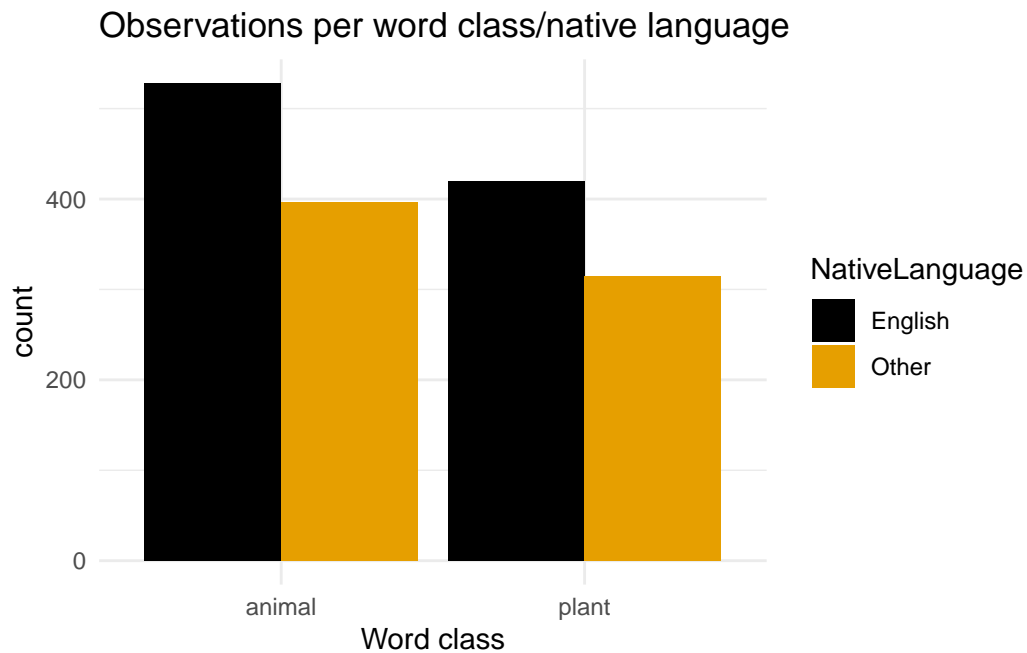
2.5. Entscheidung für ein Geom

- Warum verwenden wir ein Histogramm für die Reaktionszeit und ein Balkendiagramm für die Muttersprache?
- Um welche Arten von Variablen handelt es sich?
 - Reaktionszeit ist kontinuierlich
 - Muttersprache ist eine kategoriale Variable
- Wir verwenden Histogramme, um die Verteilungen von *kontinuierlichen* Variablen zu visualisieren.
- Wir verwenden Balkendiagramme, um Verteilungen von *kategorischen* Variablen zu visualisieren.
- Wenn wir wissen, was wir visualisieren wollen (z. B. Verteilungen) und welche Art von Variable wir haben (d. h. kontinuierlich, kategorial), können wir entscheiden, welche Art von Diagramm wir erstellen wollen.
- Oft ist es eine gute Idee, die Darstellung auf Papier zu zeichnen, bevor man in R beginnt (ich mache das auch oft).

2.6. Exercises

Diese Übungen sollten auch in Ihrem Skript enthalten sein, wenn Sie es auf Moodle hochladen. Das Durcharbeiten des Unterrichtsmaterials wird Sie auf diese Aufgaben vorbereiten.

1. Reproduzieren Sie unser Histogramm als *Dichte-Diagramm*, indem Sie `geom_histogram()` durch `geom_density()` ersetzen.
 - Was zeigt diese Art der Darstellung?
2. Erstellen Sie ein Balkendiagramm, das die Anzahl der Beobachtungen pro Wortklasse zeigt (Hinweis: Sie benötigen die Variable `Class` aus unserem Datensatz).
3. Drucken Sie Ihren Dichteplot und Ihren Klassen-Balkenplot übereinander mit Hilfe des `patchwork` Pakets
4. Reproduzieren Sie die folgenden Diagramme so genau wie möglich (Hinweis: Sie benötigen das Argument `position = "dodge"`):



Heutige Ziele

Heute haben wir gelernt...

- was Datenrahmen sind
- den Unterschied zwischen kategorialen und kontinuierlichen Daten
- wie man Diagramme mit `ggplot` erstellt
- die richtige Darstellung für unsere Daten auszuwählen

Session Info

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

```
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;
```

```
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  utils      datasets  methods    base
```

```
other attached packages:
```

```
[1] magick_2.7.4      kableExtra_1.3.4 knitr_1.44        patchwork_1.1.3
[5] ggthemes_4.2.4    languageR_1.5.0  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    xml2_1.3.4        stringi_1.7.12
[5] hms_1.1.3         digest_0.6.33     magrittr_2.0.3     evaluate_0.21
[9] grid_4.3.0        timechange_0.2.0  fastmap_1.1.1     rprojroot_2.0.3
[13] jsonlite_1.8.7    httr_1.4.6        rvest_1.0.3        fansi_1.0.4
[17] viridisLite_0.4.2 scales_1.2.1      cli_3.6.1          rlang_1.1.1
[21] munsell_0.5.0     withr_2.5.0       yaml_2.3.7         tools_4.3.0
[25] tzdb_0.4.0        colorspace_2.1-0  webshot_0.5.4     here_1.0.1
[29] pacman_0.5.1      vctrs_0.6.3       R6_2.5.1           lifecycle_1.0.3
[33] pkgconfig_2.0.3   pillar_1.9.0     gtable_0.3.4       Rcpp_1.0.11
[37] glue_1.6.2        systemfonts_1.0.4 xfun_0.39          tidyselect_1.2.0
[41] rstudioapi_0.14   farver_2.1.1     htmltools_0.5.5    svglite_2.1.1
[45] rmarkdown_2.22    labeling_0.4.3    compiler_4.3.0
```

Literaturverzeichnis

Baayen, R. H. (2008). *Analyzing Linguistic Data: A Practical Introduction to Statistics Using R*.

Baayen, R. H., & Shafaei-Bajestan, E. (2019). *languageR: Analyzing Linguistic Data: A Practical Introduction to Statistics*. <https://CRAN.R-project.org/package=languageR>

Müller, K. (2020). *here: A Simpler Way to Find Your Files*. <https://CRAN.R-project.org/package=here>

Nordmann, E., & DeBruine, L. (2022). *Applied Data Skills*. Zenodo. <https://doi.org/10.5281/zenodo.6365078>

- 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., Averick, M., Bryan, J., Chang, W., McGowan, L. D., François, R., Grolemund, G., Hayes, A., Henry, L., Hester, J., Kuhn, M., Pedersen, T. L., Miller, E., Bache, S. M., Müller, K., Ooms, J., Robinson, D., Seidel, D. P., Spinu, V., ... Yutani, H. (2019). Welcome to the tidyverse. *Journal of Open Source Software*, 4(43), 1686. <https://doi.org/10.21105/joss.01686>
- Wickham, H., Çetinkaya-Rundel, M., & Grolemund, G. (2023). *R for Data Science* (2. Aufl.).
- Winter, B. (2019). Statistics for Linguists: An Introduction Using R. In *Statistics for Linguists: An Introduction Using R*. Routledge. <https://doi.org/10.4324/9781315165547>
- Xie, Y. (2023). *tinytex: Helper Functions to Install and Maintain TeX Live, and Compile LaTeX Documents*. <https://github.com/rstudio/tinytex>

3. Dynamic reports with Quarto

Lernziele

- lernen, was dynamische Berichte sind
- unser eigenes Quarto-Dokument erstellen
- lernen, wie man ein Quarto-Dokument bearbeitet
- lernen, wie man Code in ein Quarto-Dokument einfügt
- ein Quarto-Dokument in verschiedenen Formaten wiedergeben

Lesungen

Die **Pflichtlektüre** zur Vorbereitung auf dieses Thema ist [Kap. 29 \(Quarto\)](#) und [Kap. 30 \(Quarto formats\)](#) in Wickham et al. (2023).

Eine **ergänzende Lektüre** ist [Ch. 2 \(Reproducible Workflows\)](#) in Nordmann & DeBruine (2022). Nordmann & DeBruine (2022) verwendet Rmarkdown-Skripte, während wir die nächste Generation verwenden werden: Quarto. Wir sollten in Quarto immer noch in der Lage sein, genau die gleichen Dinge zu tun, wie sie in Rmarkdown vorgeschlagen werden.

Wiederholung

Letzte Woche haben wir gelernt...

- was Datenrahmen sind
- den Unterschied zwischen kategorialen und kontinuierlichen Daten
- wie man Diagramme mit `ggplot` erstellt
- die richtige Darstellung für unsere Daten auszuwählen

Wiederholung: `ggplot()`

Sehen Sie sich diesen Code an. Was würde passieren, wenn wir ihn ausführen würden?

```
library(languageR)
library(tidyverse)
df_lexdec <- lexdec

fig_lexdec <-
  df_lexdec |>
  ggplot() +
  aes(x = RT, colour = Class) +
  geom_histogram(position = "identity", alpha = .5) +
  theme_bw()
```

Welche Darstellung in Abbildung 3.1 wird durch den folgenden Code erzeugt?

```
library(languageR)
library(tidyverse)
df_lexdec <- lexdec

fig_lexdec1 <-
  df_lexdec |>
  ggplot() +
  aes(x = RT, colour = Class) +
  geom_density(alpha = .5) +
  theme_bw()
```

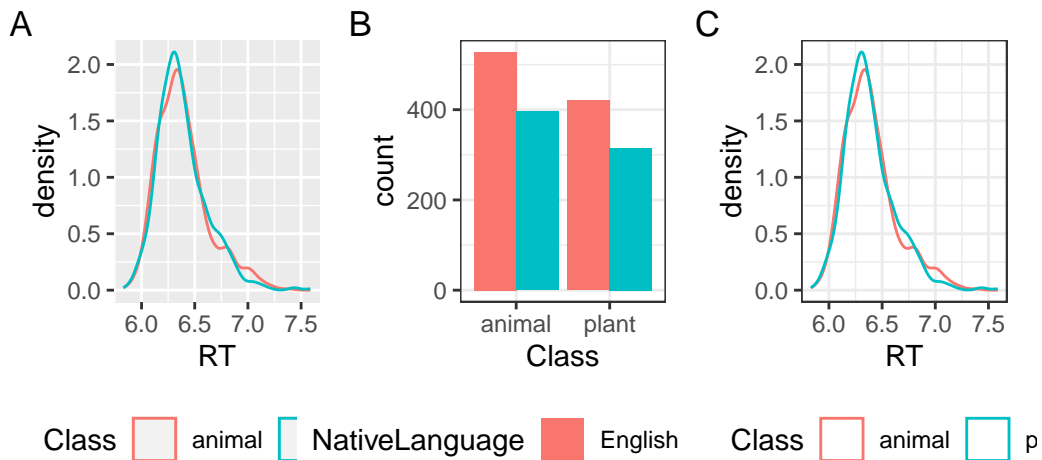


Abbildung 3.1.: Drei aus dem lexdec-Datensatz generierte Diagramme

Set-up

- wir müssen eine LaTeX-Distribution verwenden, um PDF-Dokumente mit Quarto zu erstellen
 - LaTeX ist ein Satzsystem
 - TinyTex ist eine eigene LaTeX-Distribution, mit der wir PDFs erstellen können.
 - Das Paket `tinytex` kann uns helfen, TinyTex zu installieren

Installation von LaTeX über tinytex

- Führen Sie den folgenden Code *in der Konsole* aus
- oder, wenn Sie ihn in einem Skript ausführen wollen, um zu dokumentieren, was Sie getan haben, kommentieren Sie ihn nach der Ausführung aus (d.h. fügen Sie ein `#` davor)

```
# run this in the console
install.packages("tinytex")
tinytex::install_tinytex()
```

Ordner für Woche 3

1. Fügen Sie einen Unterordner mit dem Namen `03-quarto` in `Notes` hinzu
2. Gehen Sie zu Moodle und speichern den Materialordner für '03 - Einführung in Quarto' in Ihrem `moodle` Ordner
3. Öffnen Sie das Dokument `_blatt.html` auf Ihrem Computer
 - Sehen Sie das Dokument an; Sie können oben rechts auf verschiedene Schaltflächen klicken. Probieren Sie es.

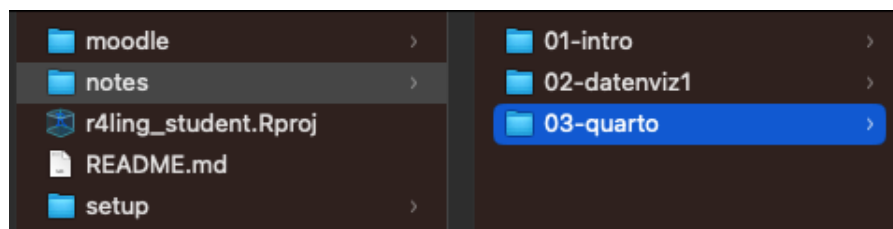


Abbildung 3.2.: Notes folder structure

3.1. Quarto


- [Quarto](#) ist ein Dateityp, der dynamische Berichte erstellt
- Quarto-Dokumente sehen genauso aus wie ihr Vorgänger, Rmarkdown

3.1.1. Dynamische Berichte

- diejenigen, die Text, Code, Codeausgabe enthalten
- Quarto bietet ein “unified authoring framework” für Data Science, das Ihren Text, Ihren Code und Ihre Code-Ausgabe einschließt (Wickham et al., 2023, Kap 29.1)
- Quarto wurde entwickelt, um auf drei Arten verwendet zu werden:
 1. Für die Kommunikation mit Entscheidungsträgern, die sich auf die Schlussfolgerungen und nicht auf den Code hinter der Analyse konzentrieren wollen.
 2. für die Zusammenarbeit mit anderen Datenwissenschaftlern (einschließlich Ihnen in der Zukunft!), die sich sowohl für Ihre Schlussfolgerungen als auch für die Art und Weise interessieren, wie Sie zu ihnen gekommen sind (d. h. für den Code).
 3. als eine Umgebung, in der Datenwissenschaft betrieben wird, als ein modernes Labornotizbuch, in dem wir nicht nur aufzeichnen können, was wir getan haben, sondern auch unsere Gedankengänge.

3.1.2. R v. Rmarkdown v. Quarto

- .R -Dateien enthalten nur (R-)Quellcode
- .Rmd *dynamische Berichte* mit
 - R-Code (und R-Pakete)
- .qmd *dynamische Berichte* (RStudio v2022.07 oder später) mit
 - R-Code (und R-Pakete)
 - Native Unterstützung für Python (und Jupyter-Notebooks)
 - Native Unterstützung für Julia

 Aufgabe 3.1: RStudio version

Beispiel 3.1.

1. Führen den folgenden Code in der Konsole aus: `RStudio.Version()$version`
 - wenn die ausgegebene Version 2022.07 oder höher ist, können Sie Quarto benutzen
 - wenn nicht:

2. Aktualisieren Sie RStudio: **Help > Check for updates**

3.1.3. Markdown

- .md-Dateien
- ein Klartext-Editor-Format, das
 - Formatierungselemente hinzufügt, die unabhängig von Gerät und Ausgabeformat sind (PDF, Word-Dokument, html...)
 - leicht zu lesen ist
- Markdown-Dokumente sind das Bindeglied zwischen unserem Quelldokument (.qmd) und unserer Ausgabe (z.B. PDF)

3.1.4. Folder structure

- jede .qmd sollte (normalerweise) in einem eigenen Ordner sein
 - d.h. es sollten nicht mehrere .qmd Dateien im selben Ordner sein
- dies ist nur mein Vorschlag, um die Ordner ordentlich und organisiert zu halten
 - d.h., es gibt keinen technischen Grund dafür (die Dokumente laufen auch dann, wenn sie sich alle im selben Ordner befinden)
- werfen wir einen Blick auf einige meiner früheren und aktuellen Projektordner

3.2. Unsere erstes Quarto-Dokument

- letzte Woche haben wir ein R-Skript erstellt, das wir über Moodle eingereicht haben
- wir werden nun unsere erste .qmd-Datei erstellen
- von nun an wird dies die Datei sein, die wir in Moodle einreichen (kein R-Skript)

Aufgabe 3.2: erste Quarto

Beispiel 3.2.

1. Erstellen Sie in Ihrem R-Projekt-Ordner, in dem ihr Ihre Kursunterlagen/Notizen aufbewahren, einen neuen Ordner für Woche 3
2. **File > New Document > Quarto Document**
 - Geben Sie ihm einen Titel wie “Quarto - Woche 3”
 - Deaktivieren Sie die Option “open with Visual Editor”.

3. Schauen das neue Skript an, um mehr über Quarto zu erfahren.
4. Klicken Sie auf die Schaltfläche “Render” am oberen Rand des Dokuments
 - Speichern Sie das Dokument in dem Ordner für Woche 3, den Sie gerade erstellt haben.
 - Was geschieht? Vergleichen die Ausgabe mit dem Quellcode des Dokuments.
5. Gehen Sie zurück zu Ihrem neuen Ordner 03-quarto
 - Was hat sich geändert?

3.2.1. Quarto-Grundlagen

- Quarto-Dokumente (wie Rmarkdown) enthalten drei wichtige Arten von Inhalten:
 1. den **YAML-Header**, der von `---` umgeben ist
 2. Text mit einer einfachen Formatierung oder Strukturierung wie `## Überschrift` oder `*Kursivschrift*`
 3. R-Code-Chunk, umgeben von ````{r}` `````

```
```{r}
#| code-line-numbers: false
Dies ist ein Code Chunk
1 + 1
```
```

[1] 2

3.2.2. YAML

- stand ursprünglich für *Yet Another Markup Language*
 - wurde aber in *YAML Ain't Markup Language* umbenannt, um den Zweck der Sprache als datenorientiert und nicht als Dokumentauszeichnung zu betonen (laut [Wikipedia](#))
- enthält alle Metainformationen zu Ihrem Dokument
 - z.B. Titel, Autorennamen
- auch Formatierungsinformationen
 - z.B. Typ der Ausgabedatei

- es gibt viele Möglichkeiten der Dokumentformatierung und -anpassung, die wir in diesem Kurs nicht behandeln werden
 - aber ich habe zum Beispiel viele YAML-Formatierungsoptionen im Quellcode meiner Folien

💡 Aufgabe 3.3: YAML

Beispiel 3.3.

1. Ändern Sie den Titel, wenn Sie das tun möchten.
2. Raten Sie, wie man einen “Untertitel” (EN: subtitle) hinzufügen könnte (Hinweis: es ist ähnlich wie beim Hinzufügen eines `title`)
3. Fügen Sie einen Autor hinzu, **Autor:** `"vorname nachname"` (siehe Beispiel unten)
4. Füge ein Inhaltsverzeichnis hinzu (EN: Table of Contents, `toc`), indem du **format** so änderst, dass es wie folgt aussieht:

```
---
title: "Quarto - Woche 3"
author: "Vorname Nachname"
format:
  html:
    toc: true
---
```

5. Rendern nun das Dokument. Sehen Sie Ihre Änderungen?

3.2.3. Strukturierung Ihres Dokuments

- wir können unser Dokument strukturieren mit
 - `##` Überschriften
 - `###` Zwischenüberschriften
 - `####` Unter-Zwischenüberschriften, usw.

```
---
title: "Quarto - Woche 3"
author: "Vorname Nachname"
format:
  html:
    toc: true
---
```

```
## Überschrift 1
```

Hier ist ein Text über das Thema, das mit dieser Überschrift verbunden ist.

```
## Überschrift 2
```

Hier ist ein weiterer Text zu einem anderen Thema.

```
### Unterüberschrift 2.1
```

Dies ist ein Text über das Unterthema.

Die Bedeutung der Formatierung

Zwischenüberschriften benötigen ein Leerzeichen nach dem letzten Hashtag (**##Zwischenüberschrift** anstelle von **##Zwischenüberschrift**), um als Überschrift gelesen zu werden. YAML erfordert außerdem einen sehr präzisen Satz. Da die Abstände in der YAML (und anderswo) so wichtig sind, möchte ich die Leerzeichen sehen und zählen können. Um dies zu tun, geht in RStudio:

- gehen zu Ihren Globalen Einstellungen (Werkzeuge > Globale Einstellungen)
- unter **Code** (linke Spalte) > **Display** (Tab), markieren das Kästchen > **Show whitespace character**

Aufgabe 3.4: Überschriften

Beispiel 3.4.

1. Kopieren den obigen Code (Überschriften und Unterüberschriften) und ersetzen den Text in der Quarto-Vorlage.
2. Ersetzen die erste Überschrift durch den Titel **Quarto**
 - Schreiben einen Text, der Quarto beschreibt, unter die Überschrift
3. Schreiben eine Unterüberschrift namens **YAML**
 - Schreiben einen Text, der die YAML-Struktur beschreibt, die wir besprochen haben
4. Erstellen eine Unterüberschrift mit dem Namen **Quarto-Struktur**.
 - Schreiben einige Notizen darüber, wie wir ein Quarto-Dokument strukturieren können (z.B. durch das Erstellen von Überschriften)

5. Finden Sie in RStudio die Schaltfläche **Outline** oben links im `.qmd` Text Editor Fenster

- Was sehen Sie, wenn Sie darauf klicken?

3.2.4. Textformatierung

- zum Formatieren von Text müssen wir die Markdown-Syntax verwenden

| Format | Markdown | Ausgabe |
|---------------|---|---|
| Kursivschrift | Dieser Text ist <code>*kursiv*</code> | Dieser Text ist <i>kursiv</i> |
| Fett | Dieser Text ist <code>**fett**</code> | Dieser Text ist fett |
| Subskription | Dieser Text ist <code>~tiefgestellt~</code> | Dieser Text ist _{tiefgestellt} |
| Hochgestellt | Dieser Text ist <code>^hochgestellt^</code> | Dieser Text ist ^{hochgestellt} |

3.2.5. Aufzählungen

- wir können Aufzählungslisten mit Bindestrichen erstellen.
 - Unteraufzählungen müssen eingerückt werden (drückt die Tabulatortaste)
- nummerierte Listen können durch einfaches Schreiben einer nummerierten Liste erstellt werden
 - Unteraufzählungen müssen in nummerierten Listen *doppelt* eingerückt werden

- dies ist ein Aufzählungszeichen

+ dies ist ein Unterpunkt

1. Dies ist ein nummerierter Punkt

a. dies ist ein unternummerierter Punkt (beachte den doppelten Einzug)

2. dies ist der zweite nummerierte Punkt

- dies ist ein Aufzählungszeichen

- dies ist ein Unterpunkt

1. Dies ist ein nummerierter Punkt

a. dies ist ein unternummerierter Punkt (beachte den doppelten Einzug)

2. dies ist der zweite nummerierte Punkt

💡 Aufgabe 3.5: Aufzählungen

Beispiel 3.5.

1. Fügen Ihrem `.qmd` Dokumententext eine Textformatierung hinzu.
2. Fügen eine Aufzählungsliste hinzu
3. Fügen eine nummerierte Liste hinzu
4. Rendern Sie das Dokument. Hat es geklappt?

3.3. Codierung in Quarto

- Der große Vorteil von dynamischen Berichten ist die Integration von Text und Code
- Vorletzte Woche haben wir gelernt, wie man einfache mathematische Berechnungen in R durchführt.
- wie würden wir R-Befehle in ein `.qmd`-Dokument einfügen?
 - Inline-Code (Code, der innerhalb einer Textzeile ausgeführt wird)
 - Code-Chunke (ein Code-Chunk, der nicht in Text enthalten ist)

3.3.1. Code-Chunks

- Code Chunks sind zwischen ````{r}` und ````` eingebettet.
- eine schöne Tastenkombination: `Cmd-Option-I` (Mac) oder `Strg-Alt-I` (PC)

```
```{r}
#| eval: false

Addition
4+6
```
```

- ihr könnt den Code in Ihrer RStudio-Sitzung ausführen, indem ihr:
 - auf das kleine grüne Dreieck oben rechts im Chunk klickt
 - die Tastenkombination `Cmd/Strg-Enter` verwendet, um eine einzelne Code-Zeile auszuführen (je nachdem, worauf der Cursor steht)
 - der Tastenkombination `Cmd/Strg-Shift-Enter` benutzt, um den gesamten Code-Chunk auszuführen (falls es mehrere Befehle innerhalb eines einzelnen Abschnitts gibt)

💡 Aufgabe 3.6: Code-Chunks

Beispiel 3.6.

1. Füge einen Code Chunk zu deiner `.qmd` Datei hinzu
 - Füge einige mathematische Operationen ein (Addition, Subtraktion, etc)
 - Fügt informative Anmerkungen zu Ihrem Code hinzu (z.B. `## Addition`)
2. Füge einen Text unter deinem Code-Chunk hinzu, der beschreibt, was der obige Code erreicht hat.
3. Rendern Sie das Dokument. Hat es geklappt?

i Erinnerung! Überschriften und Code-Anmerkungen

Denken Sie beim Schreiben von Notizen/bei der Bearbeitung von Übungen im Unterricht daran, informative Überschriften/Unterüberschriften zu erstellen! Auf diese Weise wird das Dokument strukturiert und übersichtlich, wenn ihr-in-der-Zukunft (oder ich) darauf zurückblickt.

Überschriften/Zwischenüberschriften strukturieren das gesamte Dokument. Code-Anmerkungen beschreiben, was bestimmte Teile des Codes bewirken (und warum). Beide beginnen mit einem Hashtag + Leerzeichen (`#`), aber Überschriften stehen außerhalb eines Codeabschnitts, während Codeanmerkungen innerhalb eines Codeabschnitts erscheinen.

Tipp: Klicken Sie auf die Schaltfläche “Outline” oben rechts im Texteditor-Fenster. Was zeigt sie an?

3.3.2. Code-Chunk-Optionen

- wir können die Ausführung von Code-Chunks steuern
- wir wollen nicht immer unseren Code in einem Bericht wiederholen
 - wir können dies in jedem Code-Chunk mit `##| echo: true` oder `false` steuern
- wir wollen nicht immer unseren Code in einem Bericht ausführen lassen
 - wir können dies in jedem Code-Chunk mit `##| eval: true` oder `false` steuern
- Dies würde wie folgt aussehen:

```
```${r}  
##| eval: true
```

```
Addition
4+6
```\n
```

[1] 10

- Wichtig ist, dass die Codechunk-Optionen:
 - mit `#|` beginnen, mit einem Leerzeichen dahinter und keinem Leerzeichen davor
 - direkt unter ````{r}` platziert werden

Aufgabe 3.7: `c()`

Beispiel 3.7.

1. Erinnern Sie sich, dass wir letzte Woche die Funktion `c()` (EN: concatenate) gesehen haben, die mehrere Werte kombiniert (z.B. `mean(c(3,4,25))` ergibt den Mittelwert von 3,4 und 25)
2. In einem Code-Stück: Erstellen sie ein Objekt, das eine Liste von Zahlen enthält (z.B. `Objektname <- c(...)`)
3. Berechnen Sie den Mittelwert dieser Zahlen, indem Sie nur den Objektnamen verwendet.
4. Speichern Sie den Mittelwert dieser Zahlen als ein Objekt
5. Rendern Sie das Dokument und seht sich den Abschnitt mit Ihrem Code-Chunk an.
 - Ändern Sie nun im Quellcode die Chunk-Einstellungen auf `echo: false` und rendern das Dokument. Was ändert sich?
 - Setzen nun `echo: true`, aber `eval: false`. Rendern das Dokument. Was ändert sich?

3.4. Plots in Quarto

- Ein großer Vorteil der gerenderten Quarto-Dokumente besteht darin, dass wir unsere Abbildungen zusammen mit den Textbeschreibungen anzeigen können
- Lassen Sie uns versuchen, eine Handlung von letzter Woche in unserem neuen Quarto-Dokument zu reproduzieren

3.4.1. Set-up

- unsere Pakete in einen Codechunk laden: `tidyverse`, `languageR`, und `ggthemes`

```

```{r}
Pakete laden
library(tidyverse)
library(languageR)
library(ggthemes)
```

```

- unsere Daten in einen separaten Codechunk laden (am besten ist es, einen einzigen Codechunk für einen einzigen Zweck zu verwenden)

```

```{r}
Daten laden
df_lexdec <- lexdec
```

```

3.4.2. Plots in Quarto

- Erstellen Sie jetzt einfach einen neuen Codechunk, der einen Code von letzter Woche enthält
- wir speichern es als Objekt mit dem Namen `fig_lexdec_hist`:

```

### histogram of reaction times by native language
ggplot(data = df_lexdec) +
  aes(x = exp(RT), fill = NativeLanguage) + ### set aesthetics
  geom_histogram(position = "identity", alpha = 0.3) +
  scale_fill_colorblind() + ### make fill colorblind friendly
  theme_minimal() ### set plot theme

```

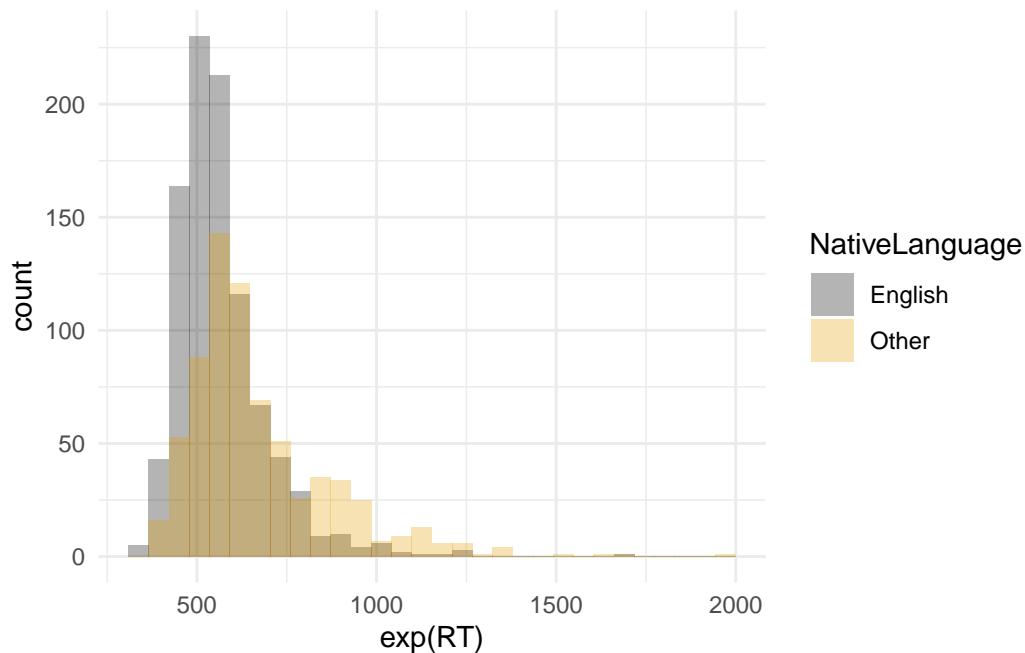


Abbildung 3.3.: Histogram of reactiontimes per native language from lexdec

3.4.3. Plots drucken

- Erinnern Sie sich an die letzte Woche: Wenn Sie einen Plot benennen, wird er nur gedruckt, wenn Sie den Namen des Objekts eingeben
- wenn Sie den Plot nicht als Objekt speichern, wird er gedruckt, wenn Sie den Code ausführen, der den Plot erzeugt
- Wenn Sie den Plot als Objekt speichern, wird er nicht gedruckt, wenn Sie den Code ausführen.
 - In diesem Fall müssen Sie den Objektnamen ausführen, um zu sehen, was unter diesem Namen gespeichert ist
 - Dies gilt für alle Arten von Objekten, nicht nur für Diagramme!

💡 Aufgabe 3.8: Plots in Quarto

Beispiel 3.8.

1. Einen neuen Codeabschnitt erstellen und das Balkendiagramm von letzter Woche erzeugen, aber als Objekt speichern
2. In einem separaten Codechunk nur den Objektnamen dieses Diagramms angeben
3. Rendern Sie das Dokument, um zu sehen, wo die Abbildung gedruckt wurde.

```
fig_lexdec_l1 <-
  ggplot(data = df_lexdec) +
    aes(x = NativeLanguage, fill = NativeLanguage) +
    ## add the geom:
    geom_bar() +
    scale_fill_colorblind() + ## add colourblind colours
    theme_minimal()
```

fig_lexdec_l1

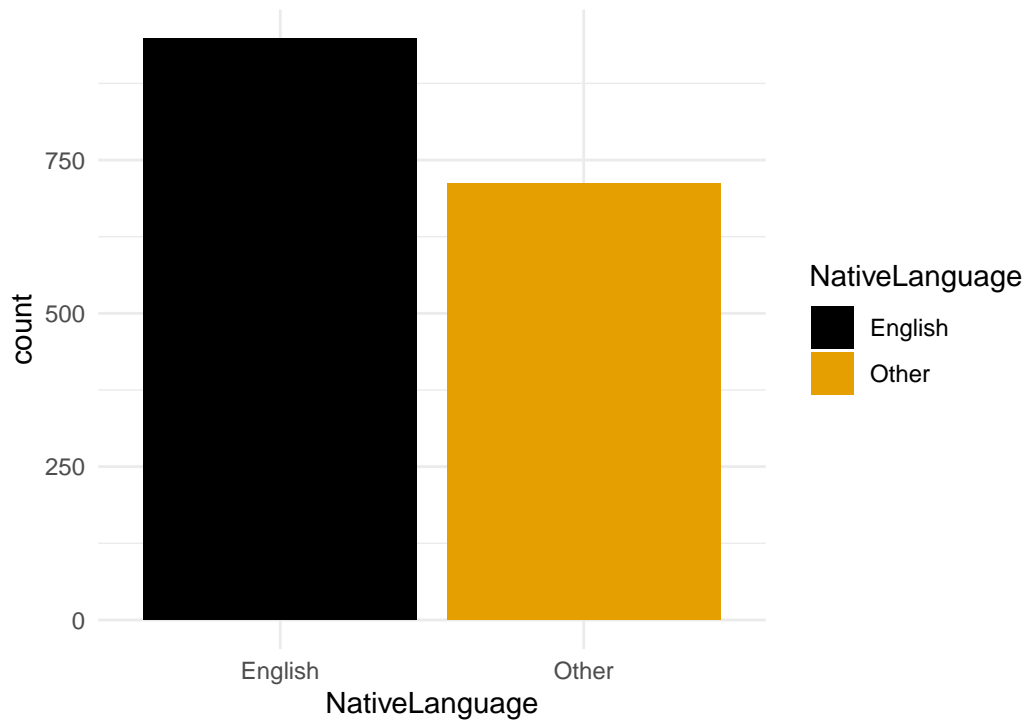


Abbildung 3.4.: Barplot of observations per native language

3.5. Ausgabeformate

- es gibt mehrere Ausgabeformate, die wahrscheinlich nützlichsten sind:
 - html (default)
 - pdf
 - revealjs (Folien)
 - docx

3.5.1. Ausgabeformate

- wenn wir das Dokument rendern:
 1. Quarto sendet die `.qmd`-Datei an **knitr** (ein R-Paket für dynamische Berichte mit R)
 2. **knitr** führt die Code-Chunke aus und erstellt ein neues `.md` Dokument mit Code und Ausgabe
 3. die `.md`-Datei wird von **pandoc** verarbeitet, das `.md`-Dateien in die fertige Datei konvertieren kann, mit vielen Ausgabeformaten



Abbildung 3.5.: Diagramm des Quarto-Workflows von `qmd`, zu `knitr`, zu `md`, zu `pandoc`, zur Ausgabe im PDF-, MS Word- oder HTML-Format. (Quelle: Wickham et al. (2023))

i Andere Verwendungen

Quarto kann für eine Vielzahl von Zwecken verwendet werden, wie z. B.:

- Websites/Blogs
- Notizen machen
- Dokumentieren von allem, was mit Code zu tun hat, um die Reproduzierbarkeit zu verbessern
 - Tipps zum Arbeitsablauf
 - Bearbeitung von `csv`-Dateien (z. B. Stimuluslisten)

💡 Aufgabe 3.9: Ausgabeformate

Beispiel 3.9.

1. Ersetzt `html` in der YAML durch `revealjs`. Rendert das Dokument.
 - Schauen Sie den Ordner für die Notizen dieser Woche an. Welche Dateien sieht?
2. Setzt nun `format` auf `pdf`. Rendert das Dokument.
 - Läuft es?
 - Versuche, `pdf` durch den Buchstaben `l` zu ersetzen. R schlägt eine Vervollständigung vor, welche ist es? Wähle sie aus und rendere das Dokument.

3. Setzt das Format wieder auf `html`. Rendert das Dokument.
4. Geht zurück zu Ihrem Ordner mit den Notizen dieser Woche. Welche Dateien sieht?
 - Ist die Ausgabe von `revealjs` dort?

Lernziele

Wir haben...

- gelernt, was dynamische Berichte sind
- unser eigenes Quarto-Dokument erstellt
- gelernt, wie man ein Quarto-Dokument bearbeitet
- gelernt, wie man Code in ein Quarto-Dokument einfügt
- ein Quarto-Dokument in verschiedenen Formaten wiedergibt

3.6. Extra: Reproduzierbarkeit in Quarto

- die Paketversionen mit `sessionInfo()` ausgeben
 - wenn ich ein neues Dokument beginne, ist eines der ersten Dinge, die ich tue, eine Kopfzeile `## Session Info` am unteren Ende hinzuzufügen, mit dem folgenden:

```
sessionInfo()
```

💡 Aufgabe 3.10: Session Info

Beispiel 3.10.

- fügt eine “Session Info” Abschnitt am Ende des Dokuments hin

Session Info

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

```
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;

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  utils      datasets  methods    base

other attached packages:
[1] ggthemes_4.2.4  magick_2.7.4    patchwork_1.1.3 lubridate_1.9.2
[5] forcats_1.0.0   stringr_1.5.0   dplyr_1.1.3     purrr_1.0.2
[9] readr_2.1.4     tidyr_1.3.0     tibble_3.2.1    ggplot2_3.4.3
[13] tidyverse_2.0.0 languageR_1.5.0

loaded via a namespace (and not attached):
[1] gt_0.9.0          utf8_1.2.3       generics_0.1.3   xml2_1.3.4
[5] stringi_1.7.12    hms_1.1.3        digest_0.6.33    magrittr_2.0.3
[9] evaluate_0.21     grid_4.3.0       timechange_0.2.0 fastmap_1.1.1
[13] rprojroot_2.0.3   jsonlite_1.8.7   fansi_1.0.4      scales_1.2.1
[17] cli_3.6.1         rlang_1.1.1      commonmark_1.9.0 munsell_0.5.0
[21] withr_2.5.0       yaml_2.3.7       tools_4.3.0      tzdb_0.4.0
[25] colorspace_2.1-0 here_1.0.1        png_0.1-8        vctrs_0.6.3
[29] R6_2.5.1          lifecycle_1.0.3  pkgconfig_2.0.3  pillar_1.9.0
[33] gtable_0.3.4      glue_1.6.2       Rcpp_1.0.11      xfun_0.39
[37] tidyselect_1.2.0  rstudioapi_0.14  knitr_1.44       farver_2.1.1
[41] htmltools_0.5.5   rmarkdown_2.22   labeling_0.4.3   compiler_4.3.0
[45] markdown_1.7

```

Literaturverzeichnis

Baayen, R. H. (2008). *Analyzing Linguistic Data: A Practical Introduction to Statistics Using R*.

- Baayen, R. H., & Shafaei-Bajestan, E. (2019). *languageR: Analyzing Linguistic Data: A Practical Introduction to Statistics*. <https://CRAN.R-project.org/package=languageR>
- Müller, K. (2020). *here: A Simpler Way to Find Your Files*. <https://CRAN.R-project.org/package=here>
- Nordmann, E., & DeBruine, L. (2022). *Applied Data Skills*. Zenodo. <https://doi.org/10.5281/zenodo.6365078>
- 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., Averick, M., Bryan, J., Chang, W., McGowan, L. D., François, R., Golemund, G., Hayes, A., Henry, L., Hester, J., Kuhn, M., Pedersen, T. L., Miller, E., Bache, S. M., Müller, K., Ooms, J., Robinson, D., Seidel, D. P., Spinu, V., ... Yutani, H. (2019). Welcome to the tidyverse. *Journal of Open Source Software*, 4(43), 1686. <https://doi.org/10.21105/joss.01686>
- Wickham, H., Çetinkaya-Rundel, M., & Golemund, G. (2023). *R for Data Science* (2. Aufl.).
- Winter, B. (2019). Statistics for Linguists: An Introduction Using R. In *Statistics for Linguists: An Introduction Using R*. Routledge. <https://doi.org/10.4324/9781315165547>
- Xie, Y. (2023). *tinytex: Helper Functions to Install and Maintain TeX Live, and Compile LaTeX Documents*. <https://github.com/rstudio/tinytex>

4. Data Wrangling 1: Transformation

Umwandlung von Daten

Lesungen

Die **Pflichtlektüre** zur Vorbereitung auf dieses Thema ist [Kap. 4 \(Data Transformation\)](#) in Wickham et al. (2023).

Eine **ergänzende Lektüre** ist [Ch. 9 \(Data Wrangling\)](#) in Nordmann & DeBruine (2022).

Wiederholung

Letzte Woche haben wir...

- gelernt, was dynamische Berichte sind
- unser eigenes Quarto-Dokument erstellt
- gelernt, wie man ein Quarto-Dokument bearbeitet
- gelernt, wie man Code in ein Quarto-Dokument einfügt
- ein Quarto-Dokument in verschiedenen Formaten wiedergibt

Heutige Ziele

Heute werden wir...

- lernen, wie man Daten mit dem Paket `dplyr` aus dem `tidyverse` verarbeitet
- lernen, wie man die `pipe` (`|>`) verwendet, um das Ergebnis einer Funktion in eine andere Funktion einzuspeisen
- Funktionen kennenlernen, die auf Zeilen operieren
- Funktionen kennenlernen, die mit Spalten arbeiten
- lernen, wie man `dplyr`-Funktionen mit Plots von `ggplot2` kombiniert

Lust auf mehr?

- [Kapital 4 \(Data transformation\)](#) in ([wickham_r_nodate?](#))
- [Kapital 9 \(Data wrangling\)](#) in Nordmann & DeBruine (2022)

4.1. Voraussetzungen

1. Frisches Quarto-Dokument

- Erstellen Sie ein neues Quarto-Dokument für den heutigen Unterricht
 - Datei > Neues Dokument > Quarto Dokument, mit dem Namen **04-wrangling**
- YAML einrichten: Titel, Ihr Name, ein **toc** hinzufügen

```
title: "Data wrangling"
subtitle: "Transforming data"
author: "Your name here"
lang: de
date: "11/08/2023"
format:
  html:
    toc: true
```

2. Pakete

- Die heutigen Pakete sind:
 - **tidyverse**: zum Verarbeiten (**dplyr**) und Plotten (**ggplot2**)
 - **languageR**: für linguistische Datensätze

```
library(tidyverse)
library(languageR)
```

3. Daten

- wir arbeiten wieder mit dem **lexdec**-Datensatz aus dem **languageR**-Paket (Baayen & Shafaei-Bajestan, 2019)
- wir speichern ihn als Objekt mit dem Namen **df_lexdec**
- wir wandeln auch die Variable **RT** um, so dass sie in Millisekunden angegeben wird (vorher war sie in log Millisekunden angegeben, aber machen Sie sich keine Gedanken darüber, was das bedeutet)
- und wir wählen 10 Variablen aus, die für uns heute relevant sind

```
df_lexdec <- lexdec |>
  mutate(RT = exp(RT)) |>
  select(Subject, RT, Trial, Sex, NativeLanguage, Correct, Word, Frequency, Class, Length)
```

4.2. Data Wrangling

- Im Englischen bezieht sich “wrangling” auf einen langen, schwierigen Prozess
 - z. B. treiben Cowboys ihre Rinder oder Herden zusammen (sammeln, sammeln ihre Tiere)
- Es gibt zwei Hauptbestandteile des Wrangling
 - Transformieren: Sortieren oder Erstellen neuer Variablen (was wir heute tun werden)
 - Aufräumen: Umformung oder Strukturierung Ihrer Daten (dies werden wir in einigen Wochen tun)
- Sowohl das Aufräumen als auch das Transformieren von Daten erfordern das Paket **dplyr** aus dem **tidyverse**.
 - **dplyr** Funktionen werden oft als Verben bezeichnet, weil sie etwas *tun*

Der Name **dplyr**

- Der Name **dplyr** kommt von einem früheren Paket, **plyr**, das dazu verwendet wird, Daten zu zerlegen, Funktionen darauf anzuwenden und zu kombinieren
 - Im Englischen klingt **plyr** wie das Wort für Zangen (“pliers”), die benutzt werden, um Dinge auseinander zu nehmen, wie das, was **plyr** mit Daten macht
 - das “d” in “dplyr” wurde hinzugefügt, weil das Paket speziell für die Arbeit mit Datenrahmen gedacht ist

4.2.1. **lexdec**

- der **lexdec**-Datensatz enthält Daten für eine lexikalische Entscheidungsaufgabe im Englischen
 - Schauen wir uns den Datensatz mit der Funktion **head()** an, die nur die ersten 6 Zeilen ausgibt
 - * hier geben wir die ersten 10 Zeilen aus

- In meinen Materialien verwende ich oft die Funktion “head()”, um zu vermeiden, dass der gesamte Datensatz in der Ausgabe gedruckt wird, aber Sie würden im Allgemeinen nicht “head()” verwenden wollen, wenn Sie Ihre Daten betrachten, sondern Ihren gesamten Datensatz betrachten wollen

Aufgabe 4.1: df_lexdec

Beispiel 4.1.

1. Betrachten Sie den Datensatz
 - wie viele Beobachtungen gibt es?
 - Wie viele Variablen gibt es?
2. Geben Sie den Datensatz in die Funktion `glimpse()` ein.
 - Was zeigt Ihnen das?
 - Wie sieht es im Vergleich zu dem aus, was Sie sehen, wenn Sie `summary()` verwenden?

4.2.2. dplyr-Grundlagen

- heute lernen wir einige der wichtigsten **dplyr**-Verben (Funktionen) kennen, mit denen wir die meisten unserer Datenmanipulationsprobleme lösen können
 - Ich verwende diese Verben mehrfach in wahrscheinlich jedem Analyseskript
- Die **dplyr**-Verben haben einige Dinge gemeinsam:
 1. das erste Argument ist immer ein Datenrahmen
 2. die folgenden Argumente beschreiben in der Regel die zu bearbeitenden Spalten, wobei der Variablenname (ohne Anführungszeichen) verwendet wird
 3. die Ausgabe ist immer ein neuer Datenrahmen
- Die Verben sind alle für eine Sache gut geeignet, so dass wir oft mehrere Verben auf einmal verwenden wollen.
 - Wir verwenden dazu die Pipe (`|>` oder `|>`)
 - Wir haben diese Pipe bereits gesehen, als wir einen Datenrahmen in `ggplot()` einspeisten.
 - wir können die Pipe als **und dann** lesen
- In dem folgenden Code identifizieren
 - den Datenrahmen
 - **dplyr**-Verben

– Variablennamen

- Kannst du versuchen, herauszulesen (zu erraten), was der folgende Code macht?

```
df_lexdec |>
  filter(Subject == "A1") |>
  select(Subject, Trial, RT, NativeLanguage, Word) |>
  relocate(NativeLanguage, .after = Trial)
```

Korrekte Syntax

.Beachten Sie, dass **A1** mit Anführungszeichen geschrieben wird, aber keiner der anderen Codes. Wenn wir ein Objekt (z.B. **df_lexdec**) oder seine Variablen (z.B. **Subject**) aufrufen, setzen wir sie nicht in Anführungszeichen. Wenn wir einen bestimmten *Wert* einer Variablen aufrufen, der nicht numerisch ist, müssen wir diesen Wert in Anführungszeichen setzen, weil die Subject ID **A1** ein Wert der Variablen **Subject** ist, müssen wir sie in Anführungszeichen setzen.

Versuchen Sie, die Anführungszeichen zu entfernen. Welche Fehlermeldung erhalten Sie? Versuchen Sie, einen Variablennamen in Anführungszeichen zu setzen, welche Fehlermeldung erhalten Sie?

Dies ist eine wichtige Übung, denn Sie werden oft feststellen, dass Ihr Code nicht läuft, aber die Lösung ist oft etwas so Einfaches wie fehlende oder zusätzliche Anführungszeichen oder Interpunktion.

4.3. Zeilen

- In aufgeräumten Daten stellen die Zeilen Beobachtungen dar.
- die wichtigsten Verben für Zeilen sind:
 - **filter()**: ändert, welche Zeilen vorhanden sind
 - **arrange()**: ändert die Reihenfolge der Zeilen
- Wir besprechen auch
 - **distinct()**: findet Zeilen mit unterschiedlichen Werten basierend auf einer Variablen (Spalte)

4.3.1. filter()

- ändert, welche Zeilen vorhanden sind, ohne ihre Reihenfolge zu ändern

- nimmt den Datenrahmen als erstes Argument
 - Die folgenden Argumente sind Bedingungen, die TRUE sein müssen, damit die Zeile erhalten bleibt
- findet alle Reaktionszeiten, die länger als 450 Millisekunden waren:

```
df_lexdec |>
  filter(RT > 450) |>
  head()
```

| | Subject | RT | Trial | Sex | NativeLanguage | Correct | Word | Frequency | Class |
|---|---------|----------|-------|-----|----------------|---------|------------|-----------|--------|
| 1 | A1 | 566.9998 | 23 | F | English | correct | owl | 4.859812 | animal |
| 2 | A1 | 548.9998 | 27 | F | English | correct | mole | 4.605170 | animal |
| 3 | A1 | 572.0000 | 29 | F | English | correct | cherry | 4.997212 | plant |
| 4 | A1 | 486.0002 | 30 | F | English | correct | pear | 4.727388 | plant |
| 6 | A1 | 483.0002 | 33 | F | English | correct | blackberry | 4.060443 | plant |
| 8 | A1 | 524.9999 | 38 | F | English | correct | squirrel | 4.709530 | animal |

| | Length |
|---|--------|
| 1 | 3 |
| 2 | 4 |
| 3 | 6 |
| 4 | 4 |
| 6 | 10 |
| 8 | 8 |

- Beachten Sie, dass wir den Wert der Reaktionszeit nicht in Anführungszeichen setzen, da er *numerisch* ist
- wenn Sie die gefilterten Daten speichern wollen, ist es in der Regel ratsam, sie unter einem *neuen* Objektnamen zu speichern
 - wenn Sie die vorgefilterte Version nicht überschreiben wollen, ist ein neuer Name erforderlich

```
df_lexdec_450 <-
  df_lexdec |>
  filter(RT > 450)
```

i Logische Operatoren

- Symbole, die zur Beschreibung einer logischen Bedingung verwendet werden
 - `==` ist *identisch* (`1 == 1`)

- `!=` ist nicht identisch (`1 != 2`)
- `>` ist größer als (`2 > 1`)
- `<` ist kleiner als (`1 < 2`)
- um Bedingungen zu kombinieren
 - `&` oder `,` *und auch* (für mehrere Bedingungen)
 - `|` *oder* (für mehrere Bedingungen)
- es gibt eine nette Abkürzung für die Kombination von `==` und `|: %in%`
 - behält Zeilen, in denen die Variable gleich einem der Werte auf der rechten Seite ist

4.3.1.1. `==` und `|`

```
df_lexdec |>
  filter(Trial == 30 | Trial == 23) |>
  head()
```

| | Subject | RT | Trial | Sex | NativeLanguage | Correct | Word | Frequency | Class |
|--------|---------|----------|-------|-----|----------------|---------|---------|-----------|--------|
| 1 | A1 | 566.9998 | 23 | F | English | correct | owl | 4.859812 | animal |
| 4 | A1 | 486.0002 | 30 | F | English | correct | pear | 4.727388 | plant |
| 475 | A2 | 561.0001 | 23 | M | English | correct | dog | 7.667626 | animal |
| 949 | C | 688.0001 | 23 | F | English | correct | vulture | 4.248495 | animal |
| 83 | D | 553.0000 | 30 | M | Other | correct | walnut | 4.499810 | plant |
| 317 | J | 824.0004 | 23 | F | Other | correct | beaver | 3.951244 | animal |
| Length | | | | | | | | | |
| 1 | 3 | | | | | | | | |
| 4 | 4 | | | | | | | | |
| 475 | 3 | | | | | | | | |
| 949 | 7 | | | | | | | | |
| 83 | 6 | | | | | | | | |
| 317 | 6 | | | | | | | | |

4.3.1.2. `%in%`

```
df_lexdec |>
  filter(Trial %in% c(30, 23)) |>
  head()
```

| | Subject | RT | Trial | Sex | NativeLanguage | Correct | Word | Frequency | Class |
|---|---------|----------|-------|-----|----------------|---------|------|-----------|--------|
| 1 | A1 | 566.9998 | 23 | F | English | correct | owl | 4.859812 | animal |

| | | | | | | | | |
|-----|----|----------|----|---|-----------------|---------|----------|--------|
| 4 | A1 | 486.0002 | 30 | F | English correct | pear | 4.727388 | plant |
| 475 | A2 | 561.0001 | 23 | M | English correct | dog | 7.667626 | animal |
| 949 | C | 688.0001 | 23 | F | English correct | vulture | 4.248495 | animal |
| 83 | D | 553.0000 | 30 | M | Other correct | walnut | 4.499810 | plant |
| 317 | J | 824.0004 | 23 | F | Other correct | beaver | 3.951244 | animal |

| | Length |
|-----|--------|
| 1 | 3 |
| 4 | 4 |
| 475 | 3 |
| 949 | 7 |
| 83 | 6 |
| 317 | 6 |

💡 Aufgabe 4.2: filter()

Beispiel 4.2.

1. Filtern Sie die Daten, um Zeilen aus Versuch 25 und Nicht-Muttersprachler (**andere**) einzuschließen.
2. Wie viele Zeilen gibt es?

4.3.2. arrange()

- ändert die Reihenfolge der Zeilen auf der Grundlage eines Wertes in einer oder mehreren Spalten

```
df_lexdec |>
  arrange(RT) |>
  head()
```

| | Subject | RT | Trial | Sex | NativeLanguage | Correct | Word | Frequency |
|------|---------|----------|-------|-----|----------------|-----------|---------|-----------|
| 542 | A2 | 340.0001 | 159 | M | English | incorrect | pig | 6.660575 |
| 815 | K | 347.9998 | 83 | F | English | incorrect | lemon | 5.631212 |
| 822 | K | 363.0001 | 99 | F | English | incorrect | potato | 6.461468 |
| 73 | A1 | 364.9999 | 174 | F | English | correct | chicken | 6.599870 |
| 524 | A2 | 365.9999 | 117 | M | English | correct | goose | 5.267858 |
| 1516 | I | 367.0001 | 51 | F | Other | correct | carrot | 4.976734 |

| | Class | Length |
|-----|--------|--------|
| 542 | animal | 3 |
| 815 | plant | 5 |

```
822 plant      6
73  animal     7
524 animal     5
1516 plant     6
```

- wenn Sie mehr als einen Spaltennamen verwenden, wird jede zusätzliche Spalte verwendet, um die Verbindung zwischen den Werten der vorangegangenen Spalten zu lösen

```
df_lexdec |>
  arrange(Length,Sex) |>
  head(10)
```

| | Subject | RT | Trial | Sex | NativeLanguage | Correct | Word | Frequency | Class |
|-----|---------|----------|-------|-----|----------------|-----------|------|-----------|--------|
| 1 | A1 | 566.9998 | 23 | F | English | correct | owl | 4.859812 | animal |
| 5 | A1 | 414.0000 | 32 | F | English | correct | dog | 7.667626 | animal |
| 15 | A1 | 556.9999 | 53 | F | English | correct | bee | 5.700444 | animal |
| 20 | A1 | 456.9998 | 61 | F | English | incorrect | bat | 5.918894 | animal |
| 31 | A1 | 581.9997 | 88 | F | English | correct | fox | 5.652489 | animal |
| 44 | A1 | 494.0002 | 113 | F | English | correct | pig | 6.660575 | animal |
| 62 | A1 | 467.9999 | 152 | F | English | correct | cat | 7.086738 | animal |
| 64 | A1 | 875.9999 | 157 | F | English | correct | ant | 5.347108 | animal |
| 719 | A3 | 607.0001 | 41 | F | Other | correct | ant | 5.347108 | animal |
| 720 | A3 | 562.0001 | 44 | F | Other | correct | pig | 6.660575 | animal |
| | Length | | | | | | | | |
| 1 | 3 | | | | | | | | |
| 5 | 3 | | | | | | | | |
| 15 | 3 | | | | | | | | |
| 20 | 3 | | | | | | | | |
| 31 | 3 | | | | | | | | |
| 44 | 3 | | | | | | | | |
| 62 | 3 | | | | | | | | |
| 64 | 3 | | | | | | | | |
| 719 | 3 | | | | | | | | |
| 720 | 3 | | | | | | | | |

- wir können `desc()` innerhalb von `arrange()` hinzufügen, um eine absteigende Reihenfolge (groß-klein) anstelle der standardmäßigen aufsteigenden Reihenfolge zu verwenden

```
df_lexdec |>
  arrange(desc(Length)) |>
```

`head()`

| | Subject | RT | Trial | Sex | NativeLanguage | Correct | Word | Frequency |
|-----|---------|----------|-------|-----|----------------|---------|------------|-----------|
| 6 | A1 | 483.0002 | 33 | F | English | correct | blackberry | 4.060443 |
| 7 | A1 | 417.9998 | 34 | F | English | correct | strawberry | 4.753590 |
| 69 | A1 | 540.9998 | 168 | F | English | correct | woodpecker | 2.890372 |
| 505 | A2 | 503.9999 | 87 | M | English | correct | woodpecker | 2.890372 |
| 516 | A2 | 400.9998 | 105 | M | English | correct | strawberry | 4.753590 |
| 518 | A2 | 517.0001 | 108 | M | English | correct | blackberry | 4.060443 |

| | Class | Length |
|-----|--------|--------|
| 6 | plant | 10 |
| 7 | plant | 10 |
| 69 | animal | 10 |
| 505 | animal | 10 |
| 516 | plant | 10 |
| 518 | plant | 10 |

Aufgabe 4.3: `arrange()`

Beispiel 4.3.

1. Filtere die Daten so, dass sie nur Beobachtungen der “Probanden” M1 und W2 enthalten, *und dann*
2. Ordnen Sie die Daten nach absteigender Reaktionszeit

4.4. Spalten

- In Tidy Data stellen die Spalten Variablen dar.
- die wichtigsten Verben für Spalten sind:
 - `rename()`: ändert die Namen der Spalten
 - `mutate()`: erzeugt neue Spalten, die von den vorhandenen Spalten abgeleitet werden
 - `select()`: ändert, welche Spalten vorhanden sind
 - `relocate()`: ändert die Position der Spalten

4.4.1. `rename()`

- Mit `rename()` können wir den Namen von Spalten ändern

- die Reihenfolge der Argumente ist `neuer_name = alter_name`
- Versuchen wir, einige der Variablennamen auf Deutsch zu ändern
 - Ich neige dazu, Variablennamen in Kleinbuchstaben zu schreiben, als Kodierungskonvention

```
## single variable
df_lexent <-
  df_lexdec |>
  rename(teilnehmer = Subject)

## or multiple variables at once
df_lexent <-
  df_lexdec |>
  rename(teilnehmer = Subject,
         rz_ms = RT,
         geschlecht = Sex,
         laenge = Length)
```

4.4.2. mutate()

- Mit `mutate()` werden neue Spalten aus vorhandenen Spalten erzeugt.
 - So können wir z.B. einfache Algebra mit den Werten in jeder Spalte durchführen

```
df_lexent |>
  mutate(
    rz_laenge = rz_ms / laenge,
  ) |>
  head()
```

| | teilnehmer | rz_ms | Trial | geschlecht | NativeLanguage | Correct | Word |
|---|------------|----------|-------|------------|----------------|---------|------------|
| 1 | A1 | 566.9998 | 23 | F | English | correct | owl |
| 2 | A1 | 548.9998 | 27 | F | English | correct | mole |
| 3 | A1 | 572.0000 | 29 | F | English | correct | cherry |
| 4 | A1 | 486.0002 | 30 | F | English | correct | pear |
| 5 | A1 | 414.0000 | 32 | F | English | correct | dog |
| 6 | A1 | 483.0002 | 33 | F | English | correct | blackberry |

| | Frequency | Class | laenge | rz_laenge |
|---|-----------|--------|--------|-----------|
| 1 | 4.859812 | animal | 3 | 188.99994 |
| 2 | 4.605170 | animal | 4 | 137.24994 |
| 3 | 4.997212 | plant | 6 | 95.33333 |

```

4 4.727388 plant      4 121.50005
5 7.667626 animal    3 138.00000
6 4.060443 plant     10 48.30002

```

- Mit `mutate()` werden diese neuen Spalten auf der rechten Seite des Datensatzes hinzugefügt.
 - Das macht es schwierig zu sehen, was passiert.
- um zu kontrollieren, wo die neue Spalte hinzugefügt wird, können wir `.before` oder `.after` verwenden

```

df_lexent |>
  mutate(
    rz_laenge = rz_ms / laenge,
    .after = rz_ms
  ) |>
  head()

```

| | teilnehmer | rz_ms | rz_laenge | Trial | geschlecht | NativeLanguage | Correct |
|---|------------|----------|-----------|-------|------------|----------------|---------|
| 1 | A1 | 566.9998 | 188.99994 | 23 | F | English | correct |
| 2 | A1 | 548.9998 | 137.24994 | 27 | F | English | correct |
| 3 | A1 | 572.0000 | 95.33333 | 29 | F | English | correct |
| 4 | A1 | 486.0002 | 121.50005 | 30 | F | English | correct |
| 5 | A1 | 414.0000 | 138.00000 | 32 | F | English | correct |
| 6 | A1 | 483.0002 | 48.30002 | 33 | F | English | correct |

| | | Word Frequency | Class | laenge |
|---|------------|----------------|--------|--------|
| 1 | owl | 4.859812 | animal | 3 |
| 2 | mole | 4.605170 | animal | 4 |
| 3 | cherry | 4.997212 | plant | 6 |
| 4 | pear | 4.727388 | plant | 4 |
| 5 | dog | 7.667626 | animal | 3 |
| 6 | blackberry | 4.060443 | plant | 10 |

! Rendernpause!

Nehmen Sie sich einen Moment Zeit, um Ihr Dokument zu rendern. Wird es gerendert? Können Sie das Dokument besser strukturieren? Z. B. durch Hinzufügen von mehr Überschriften, Text?

💡 Aufgabe 4.4: mutate()

Beispiel 4.4.

1. Create a new variable called `rz_s` in `df_lexent`:
 - equals `rz_ms` divided by 1000 (i.e., converts milliseconds to seconds)
 - appears after `rz_ms`
2. Render your document

4.4.3. select()

- `select()` fasst die Daten so zusammen, dass sie nur die gewünschten Spalten enthalten
- Spalten nach Namen auswählen

```
df_lexent |>
  select(teilnehmer, rz_ms, Word) |>
  head()
```

| | teilnehmer | rz_ms | Word |
|---|------------|----------|------------|
| 1 | A1 | 566.9998 | owl |
| 2 | A1 | 548.9998 | mole |
| 3 | A1 | 572.0000 | cherry |
| 4 | A1 | 486.0002 | pear |
| 5 | A1 | 414.0000 | dog |
| 6 | A1 | 483.0002 | blackberry |

- select alle Spalten zwischen `rz_ms` und `geschlecht`

```
df_lexent |>
  select(rz_ms:geschlecht) |>
  head()
```

| | rz_ms | rz_s | Trial | geschlecht |
|---|----------|-----------|-------|------------|
| 1 | 566.9998 | 0.5669998 | 23 | F |
| 2 | 548.9998 | 0.5489998 | 27 | F |
| 3 | 572.0000 | 0.5720000 | 29 | F |
| 4 | 486.0002 | 0.4860002 | 30 | F |
| 5 | 414.0000 | 0.4140000 | 32 | F |
| 6 | 483.0002 | 0.4830002 | 33 | F |

- alle Spalten außer `rz_s` auswählen (! wird als “nicht” gelesen)

```
df_lexent |>
  select(!rz_s) |>
  head()
```

| | teilnehmer | rz_ms | Trial | geschlecht | NativeLanguage | Correct | Word |
|---|------------|----------|-------|------------|----------------|---------|------------|
| 1 | A1 | 566.9998 | 23 | F | English | correct | owl |
| 2 | A1 | 548.9998 | 27 | F | English | correct | mole |
| 3 | A1 | 572.0000 | 29 | F | English | correct | cherry |
| 4 | A1 | 486.0002 | 30 | F | English | correct | pear |
| 5 | A1 | 414.0000 | 32 | F | English | correct | dog |
| 6 | A1 | 483.0002 | 33 | F | English | correct | blackberry |

| | Frequency | Class | laenge |
|---|-----------|--------|--------|
| 1 | 4.859812 | animal | 3 |
| 2 | 4.605170 | animal | 4 |
| 3 | 4.997212 | plant | 6 |
| 4 | 4.727388 | plant | 4 |
| 5 | 7.667626 | animal | 3 |
| 6 | 4.060443 | plant | 10 |

4.4.3.1. `select()`-Hilfsfunktionen

- einige Hilfsfunktionen, die das Leben bei der Arbeit mit `select()` erleichtern:
 - `starts_with("abc")`: wählt Spalten aus, die mit einer bestimmten Zeichenkette beginnen
 - `ends_with("xyz")`: wählt Spalten aus, die mit einer bestimmten Zeichenkette enden
 - `contains("ijk")`: wählt Spalten aus, die eine bestimmte Zeichenkette enthalten
 - `where(is.character)`: wählt Spalten aus, die einem logischen Kriterium entsprechen
 - * z.B. gibt die Funktion `is.character()` den Wert `TRUE` zurück, wenn eine Variable Zeichenketten enthält, nicht numerische Werte oder Kategorien

```
df_lexent |>
  select(starts_with("w")) |>
  head()
```

| | Word |
|---|------|
| 1 | owl |

```

2     mole
3     cherry
4     pear
5     dog
6 blackberry

```

```

df_lexent |>
  select(ends_with("er")) |>
  head()

```

```

teilnehmer
1      A1
2      A1
3      A1
4      A1
5      A1
6      A1

```

💡 Aufgabe 4.5: `select()`

Beispiel 4.5.

1. Drucke die Spalten in `df_lexent`, die mit “t” beginnen
2. Drucke die Spalten in `df_lexent`, die “ge” enthalten
3. Drucke die Spalten in `df_lexent`, die
 - mit mit “r” beginnen, und
 - mit “s” enden

4.4.4. `relocate()`

- `relocate()` verschiebt Variablen
 - standardmäßig werden sie nach vorne verschoben

```

df_lexent |> relocate(Trial) |>
  head()

```

```

Trial teilnehmer    rz_ms    rz_s geschlecht NativeLanguage Correct
1    23          A1 566.9998 0.5669998         F      English correct

```


| | | | | | | |
|---|----|----|----------|-----------|---|-----------------|
| 2 | 27 | A1 | 548.9998 | 0.5489998 | F | English correct |
| 3 | 29 | A1 | 572.0000 | 0.5720000 | F | English correct |
| 4 | 30 | A1 | 486.0002 | 0.4860002 | F | English correct |
| 5 | 32 | A1 | 414.0000 | 0.4140000 | F | English correct |
| 6 | 33 | A1 | 483.0002 | 0.4830002 | F | English correct |

| | Word | Frequency | Class | laenge |
|---|------------|-----------|--------|--------|
| 1 | owl | 4.859812 | animal | 3 |
| 2 | mole | 4.605170 | animal | 4 |
| 3 | cherry | 4.997212 | plant | 6 |
| 4 | pear | 4.727388 | plant | 4 |
| 5 | dog | 7.667626 | animal | 3 |
| 6 | blackberry | 4.060443 | plant | 10 |

- aber wir können auch `.before` oder `.after` verwenden, um eine Variable zu platzieren

```
df_lexent |>
  relocate(Trial, .after = teilnehmer) |>
  head()
```

| | teilnehmer | Trial | | rz_ms | rz_s | geschlecht | NativeLanguage | Correct |
|---|------------|-------|----|----------|-----------|------------|----------------|---------|
| 1 | | A1 | 23 | 566.9998 | 0.5669998 | F | English | correct |
| 2 | | A1 | 27 | 548.9998 | 0.5489998 | F | English | correct |
| 3 | | A1 | 29 | 572.0000 | 0.5720000 | F | English | correct |
| 4 | | A1 | 30 | 486.0002 | 0.4860002 | F | English | correct |
| 5 | | A1 | 32 | 414.0000 | 0.4140000 | F | English | correct |
| 6 | | A1 | 33 | 483.0002 | 0.4830002 | F | English | correct |

| | Word | Frequency | Class | laenge |
|---|------------|-----------|--------|--------|
| 1 | owl | 4.859812 | animal | 3 |
| 2 | mole | 4.605170 | animal | 4 |
| 3 | cherry | 4.997212 | plant | 6 |
| 4 | pear | 4.727388 | plant | 4 |
| 5 | dog | 7.667626 | animal | 3 |
| 6 | blackberry | 4.060443 | plant | 10 |

4.5. dplyr und ggplot2

- wir können einen Datensatz mit den `dplyr`-Verben ändern und diese Änderungen dann in `ggplot2` einspeisen
- Was wird der folgende Code ergeben?

```
df_lexent |>
  ## filter the data
  filter(rz_ms > 120,
         rz_ms > 500) |>
  ## plot the filtered data
  ggplot(aes(x = fct_infreq(Correct))) +
  geom_bar() +
  theme_minimal()
```

4.5.1. Pipe versus plus (|> vs. +)

- wichtig: wir können Pipes (|>) verwenden, um zusätzliche Verben/Funktionen mit dem Ergebnis einer vorherigen Codezeile auszuführen
 - Die Funktion `ggplot()` verwendet jedoch `+`, um neue *Ebenen* zur Darstellung hinzuzufügen

! Rendernpause!

Nehmen Sie sich einen Moment Zeit, um Ihr Dokument zu rendern. Wird es gerendert? Können Sie das Dokument besser strukturieren? Z. B. durch Hinzufügen von mehr Überschriften, Text?

Aufgaben

1. Drucken Sie in einer einzigen Pipeline `df_lexent`, wobei Sie nur die Spalten Reaktionszeiten (in Millisekunden), `NativeLanguage` und `Word` für Zeilen auswählen, die jede der folgenden Bedingungen erfüllen, sie in der Reihenfolge der Reaktionszeiten anordnen und so filtern, dass nur diese Zeilen berücksichtigt werden:
 - die Reaktionszeiten waren größer als 500ms *und* kleiner als 550ms
 - aus den Wörtern “pear”, “elephant” oder “tortoise” stammen
2. Sortiere (`arrange()`) `df_lexent` in absteigender Reihenfolge, um die Versuche mit den längsten Reaktionszeiten zu finden.
3. Speichern Sie in einer einzigen Pipeline ein neues Objekt namens `df_rz`, das `df_lexent` enthält, *und dann*:
 - Selektieren (`select()`) Sie die Variablen `Teilnehmer`, `NativeLanguage`, `Word`, `rz_s`, `laenge`, und `Frequency`

- Erstelle eine neue Variable `rz_s_laenge` (`mutate()`), die `rz_s` geteilt durch `laenge` ist
 - und wird vor `Laenge` gesetzt
- Benennen (`rename()`) Sie diese Variablen in Englisch um, so dass sie in Deutsch (und mit Kleinbuchstaben) sind.

Heutige Ziele

Heute haben wir gelernt...

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

Session Info

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

```
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;
```

```
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  utils      datasets  methods   base
```

other attached packages:

```
[1] languageR_1.5.0 lubridate_1.9.2 forcats_1.0.0  stringr_1.5.0
[5] dplyr_1.1.3      purrr_1.0.2     readr_2.1.4    tidyr_1.3.0
[9] tibble_3.2.1     ggplot2_3.4.3   tidyverse_2.0.0
```

loaded via a namespace (and not attached):

```
[1] gtable_0.3.4      jsonlite_1.8.7    compiler_4.3.0    tidyselect_1.2.0
[5] scales_1.2.1      yaml_2.3.7        fastmap_1.1.1     R6_2.5.1
[9] generics_0.1.3    knitr_1.44        munsell_0.5.0     pillar_1.9.0
[13] tzdb_0.4.0        rlang_1.1.1       utf8_1.2.3        stringi_1.7.12
[17] xfun_0.39         timechange_0.2.0  cli_3.6.1         withr_2.5.0
[21] magrittr_2.0.3    digest_0.6.33     grid_4.3.0        rstudioapi_0.14
[25] hms_1.1.3         lifecycle_1.0.3   vctrs_0.6.3       evaluate_0.21
[29] glue_1.6.2        fansi_1.0.4       colorspace_2.1-0  rmarkdown_2.22
[33] tools_4.3.0       pkgconfig_2.0.3   htmltools_0.5.5
```

Literaturverzeichnis

- Baayen, R. H. (2008). *Analyzing Linguistic Data: A Practical Introduction to Statistics Using R*.
- Baayen, R. H., & Shafaei-Bajestan, E. (2019). *languageR: Analyzing Linguistic Data: A Practical Introduction to Statistics*. <https://CRAN.R-project.org/package=languageR>
- Müller, K. (2020). *here: A Simpler Way to Find Your Files*. <https://CRAN.R-project.org/package=here>
- Nordmann, E., & DeBruine, L. (2022). *Applied Data Skills*. Zenodo. <https://doi.org/10.5281/zenodo.6365078>
- 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., Averick, M., Bryan, J., Chang, W., McGowan, L. D., François, R., Grolemund, G., Hayes, A., Henry, L., Hester, J., Kuhn, M., Pedersen, T. L., Miller, E., Bache, S. M., Müller, K., Ooms, J., Robinson, D., Seidel, D. P., Spinu, V., ... Yutani, H. (2019). Welcome to the tidyverse. *Journal of Open Source Software*, 4(43), 1686. <https://doi.org/10.21105/joss.01686>
- Wickham, H., Çetinkaya-Rundel, M., & Grolemund, G. (2023). *R for Data Science* (2. Aufl.).
- Winter, B. (2019). Statistics for Linguists: An Introduction Using R. In *Statistics for Linguists: An Introduction Using R*. Routledge. <https://doi.org/10.4324/9781315165547>
- Xie, Y. (2023). *tinytex: Helper Functions to Install and Maintain TeX Live, and Compile LaTeX Documents*. <https://github.com/rstudio/tinytex>

5. Datenvisualisierung 2

Visualisierung von Beziehungen

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).

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

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

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

```
library(tidyverse)
```

```
-- Attaching core tidyverse packages ----- tidyverse 2.0.0 --
v dplyr      1.1.3      v readr      2.1.4
v forcats    1.0.0      v stringr    1.5.0
v ggplot2    3.4.3      v tibble     3.2.1
v lubridate  1.9.2      v tidyr      1.3.0
v purrr      1.0.2
```

```
-- Conflicts ----- tidyverse_conflicts() --
```

```
x dplyr::filter() masks stats::filter()
```

```
x dplyr::lag()     masks stats::lag()
```

```
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become
```

```
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

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

```
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:

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*.

Tabelle 5.1.: 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 |

* 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?

5.1. 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)

5.1.1. 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.

5.2. 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
 - 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

5.2.1. Gruppierte kontinuierliche Variable

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

5.2.1.1. Gestapelt

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

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

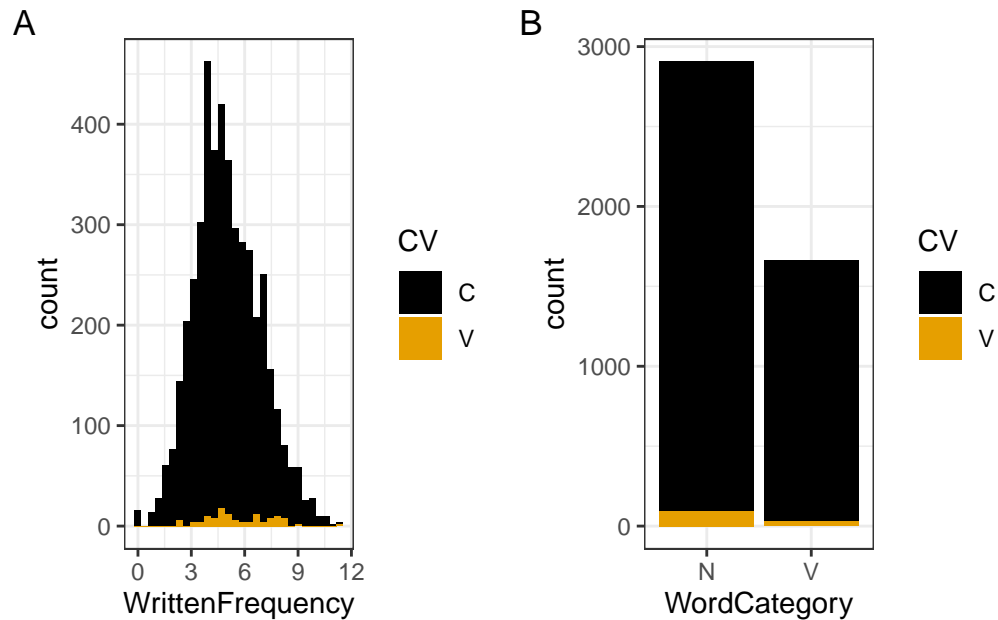


Abbildung 5.1.: Visualising relationships in distributions

5.2.1.2. Dodged (Ausgewiche)

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

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

5.2.2. 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?

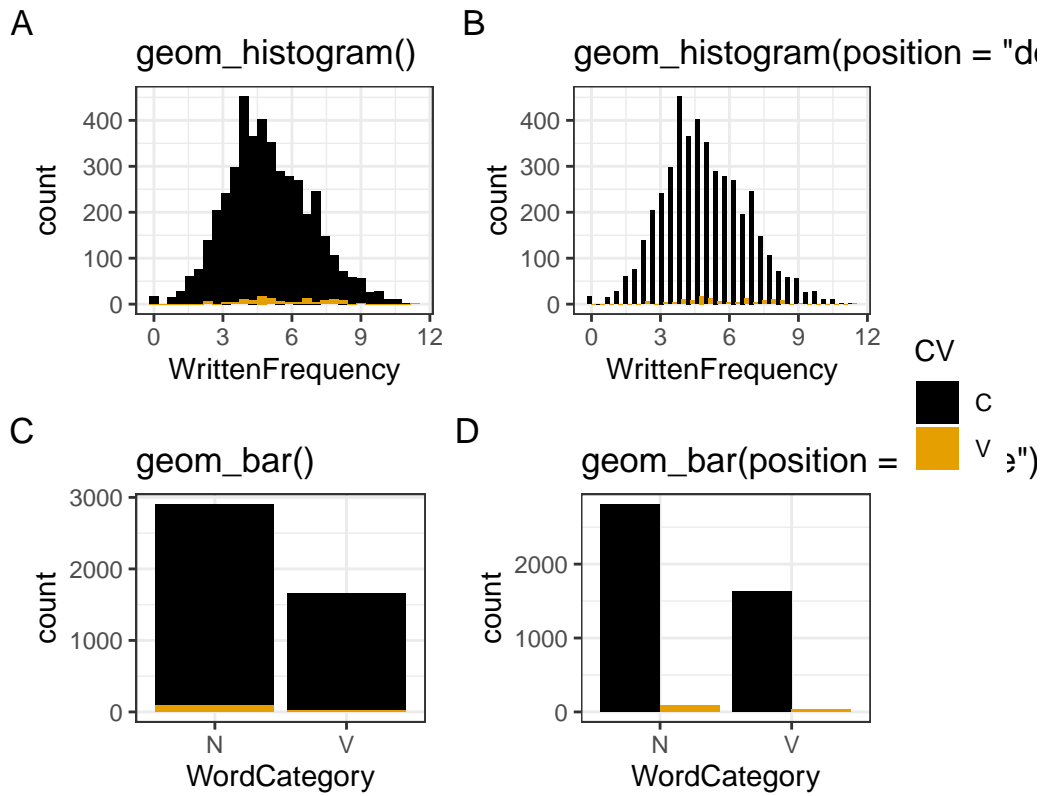
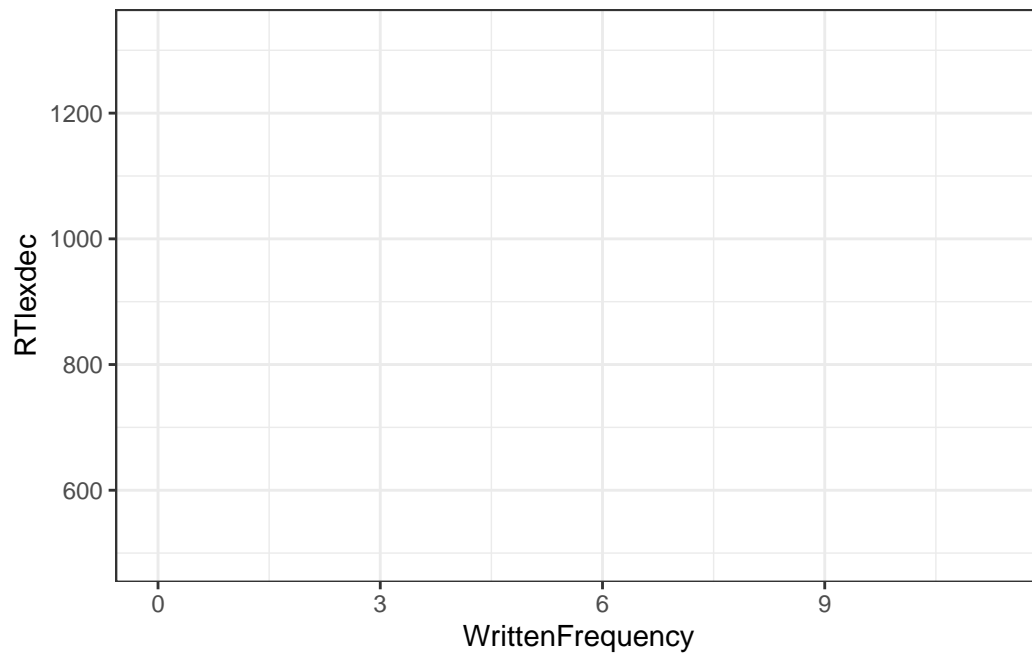
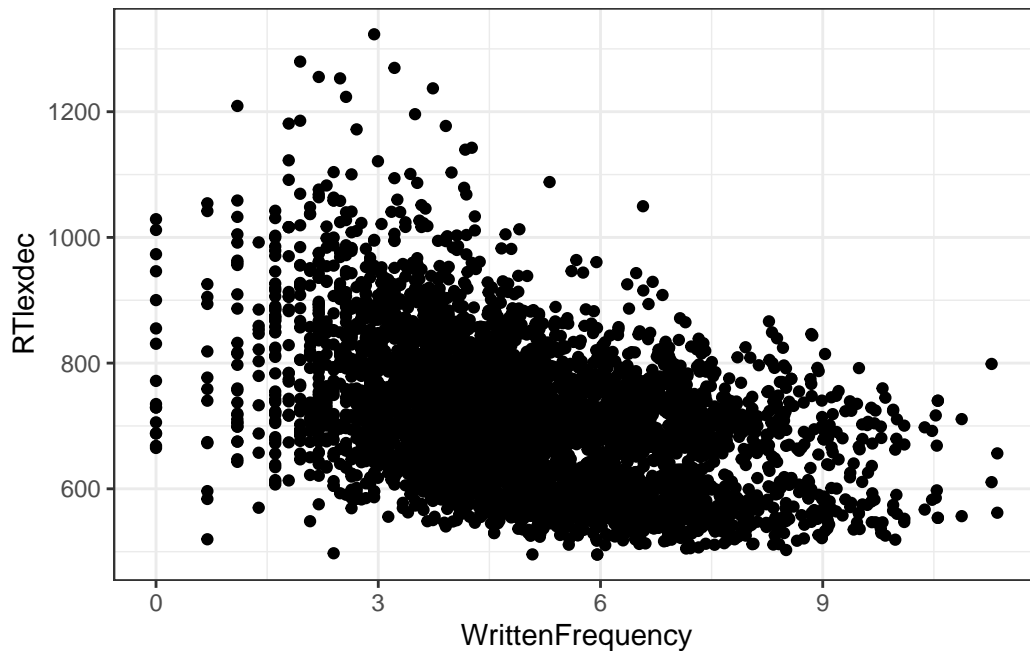


Abbildung 5.2.: Visualising relationships in distributions

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



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

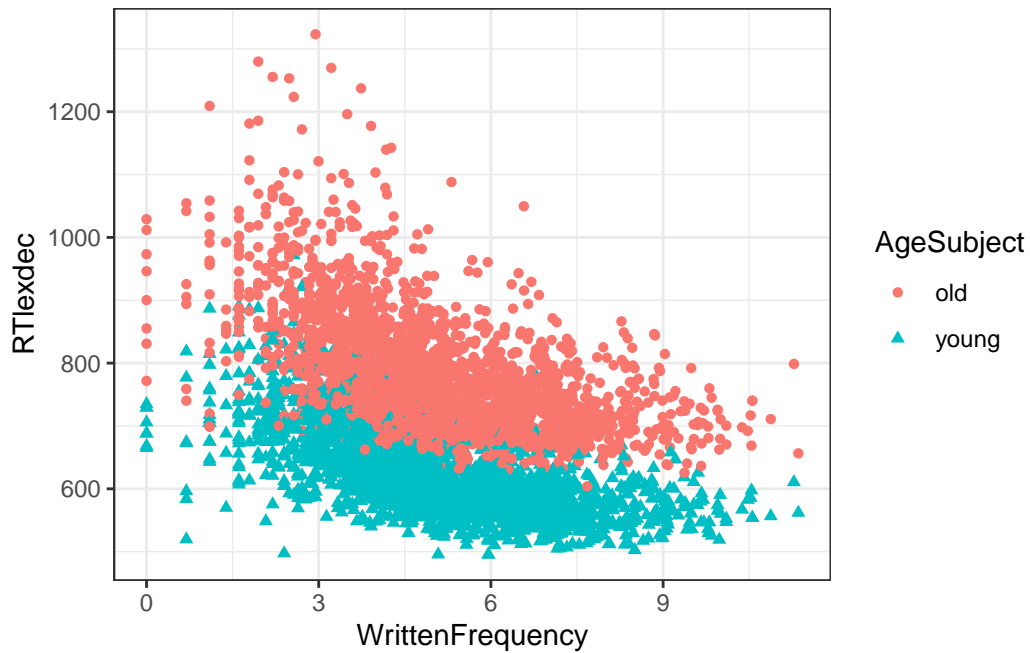


- Nehmen Sie sich einen Moment Zeit, um diese Grafik zu betrachten und eine Interpretation zu finden
 - Welchen Einfluss hat die Schrifthäufigkeit eines Wortes auf die Reaktionszeit bei einer lexikalischen Entscheidungsaufgabe?
 - Vervollständigen Sie den Satz: “Wörter mit einer höheren Worthäufigkeit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

5.2.3. Hinzufügen weiterer Variablen

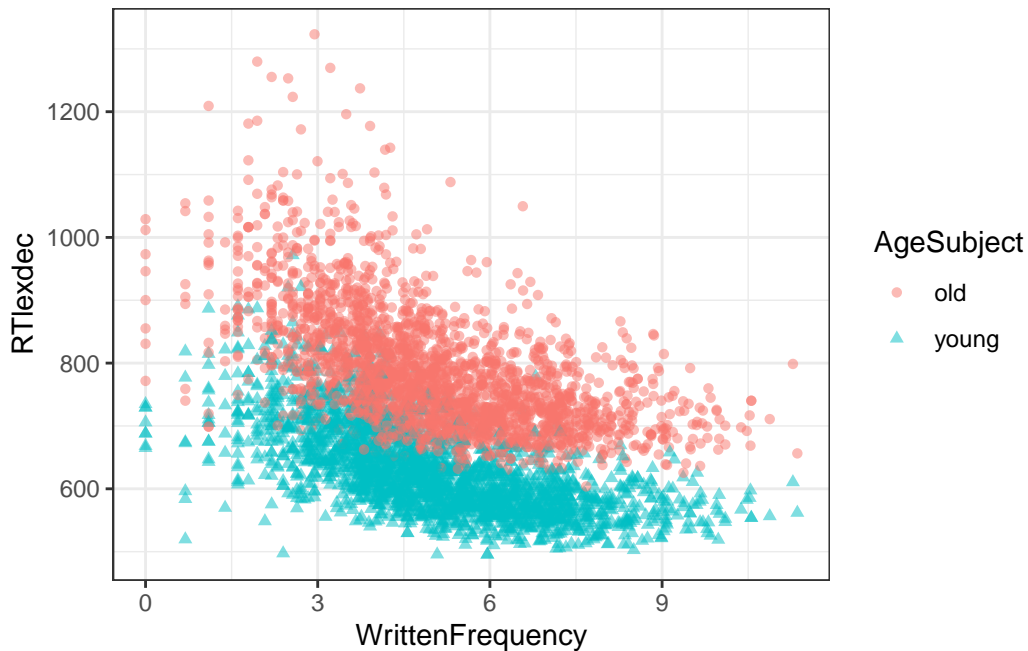
- Erinnern Sie sich daran, dass wir andere Ästhetiken wie `fill` oder `colour` verwenden können
 - für `geom_point()` ist es auch hilfreich, `shape` zu verwenden

```
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 5.1: Adding another variable

Beispiel 5.1.

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

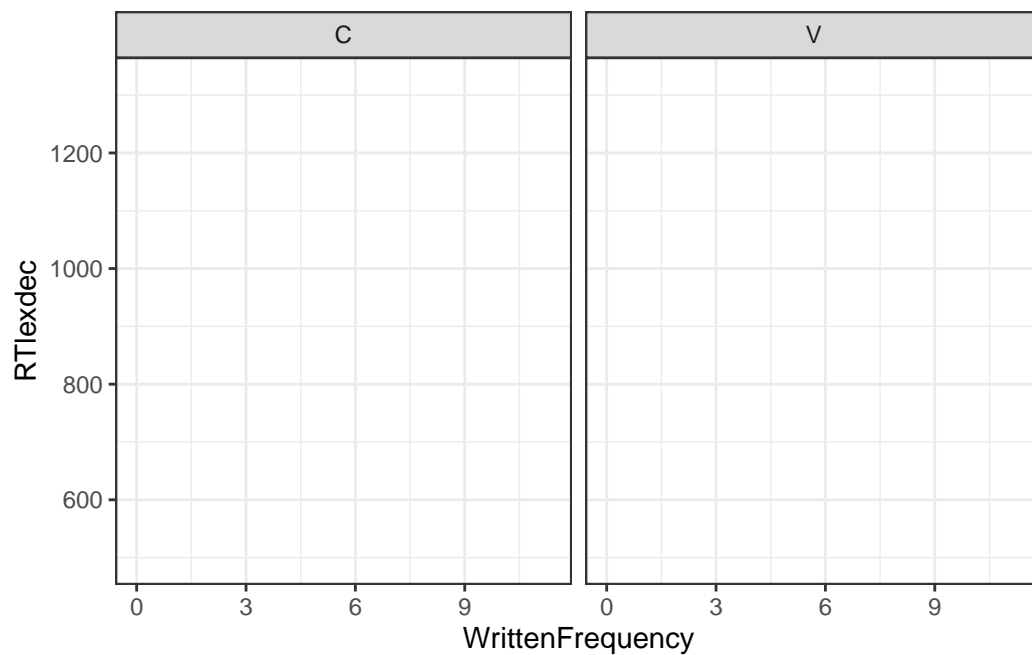
5.2.4. Facet grids

- 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)

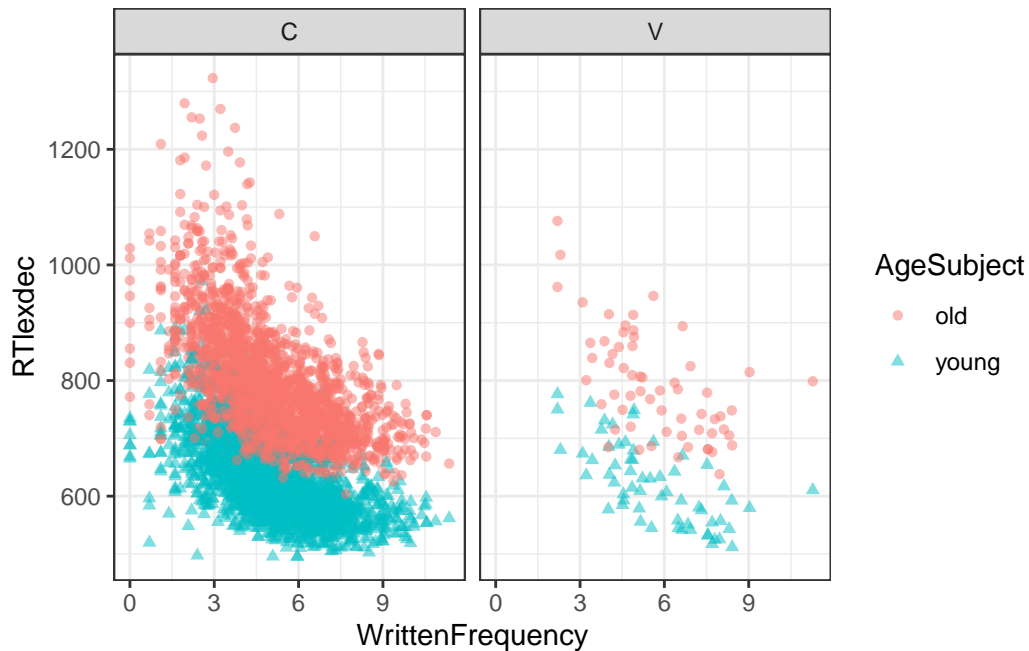
```



```

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)

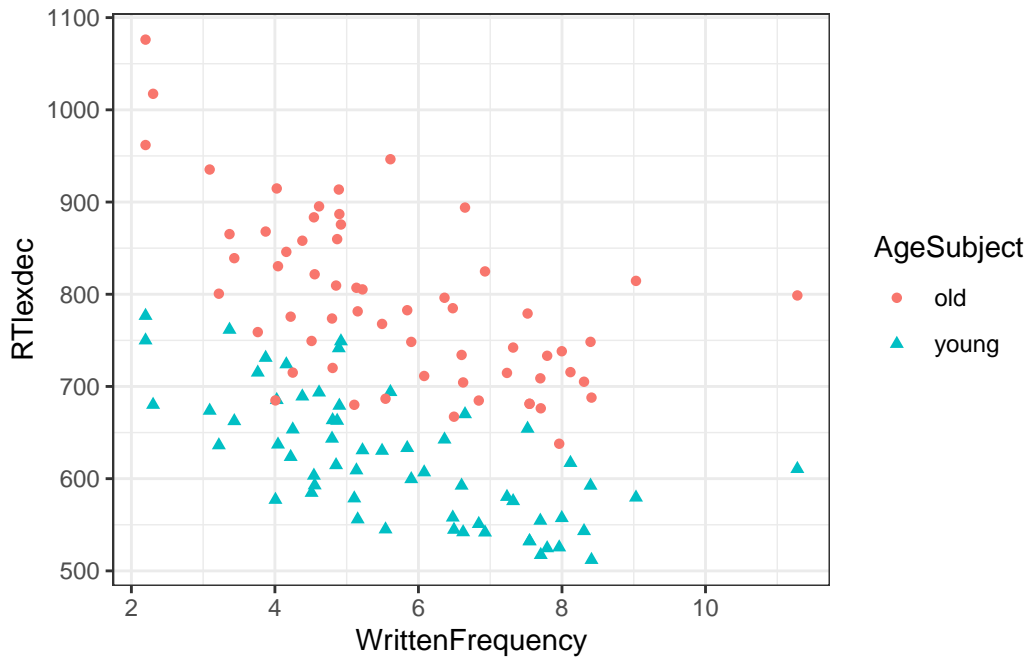
```

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 5.2: Plot-Anmerkung

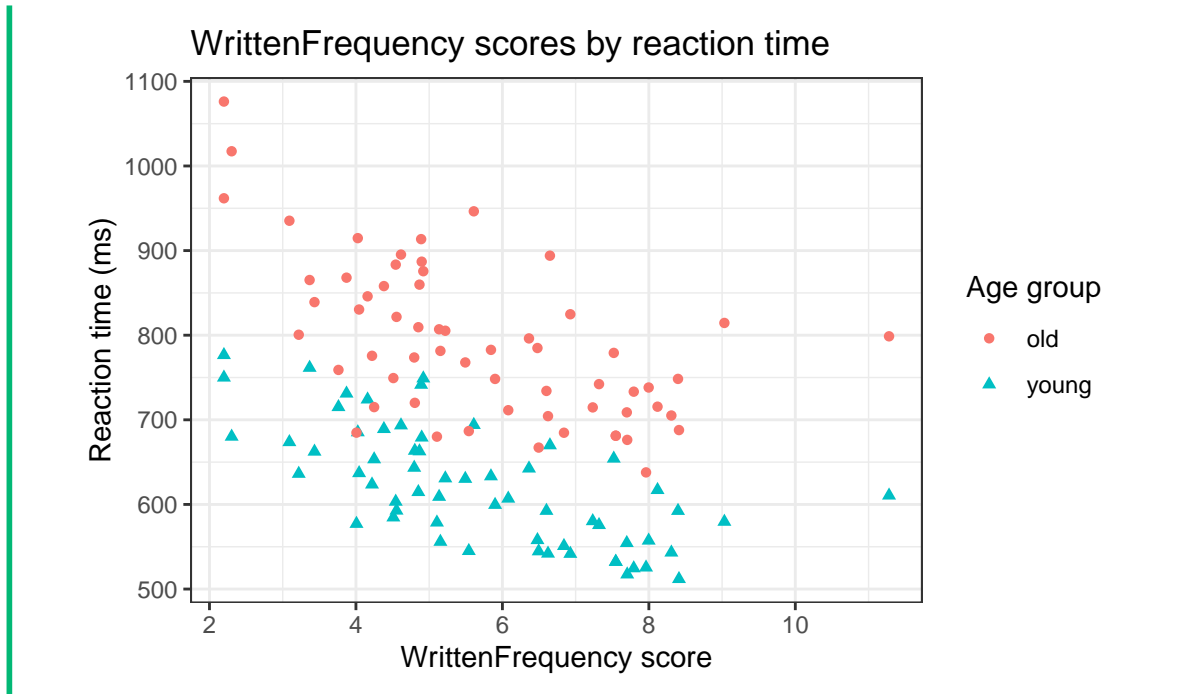
Beispiel 5.2.

- 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()
```

Tabelle 5.2.: 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? |



5.4. 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:

5.4.1. 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()
```
```

5.5. 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.

5.5.1. `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

5.5.1.1. 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 5.3: ggsave()

Beispiel 5.3.

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?

5.6. 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).

```
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;
```

```
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  utils      datasets  methods   base
```

```
other attached packages:
```

```
[1] kableExtra_1.3.4 knitr_1.44      languageR_1.5.0 ggthemes_4.2.4
[5] patchwork_1.1.3  lubridate_1.9.2  forcats_1.0.0   stringr_1.5.0
[9] dplyr_1.1.3      purrr_1.0.2      readr_2.1.4     tidyr_1.3.0
[13] 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  xml2_1.3.4      stringi_1.7.12
[5] hms_1.1.3       digest_0.6.33   magrittr_2.0.3   evaluate_0.21
[9] grid_4.3.0      timechange_0.2.0 fastmap_1.1.1    jsonlite_1.8.7
[13] httr_1.4.6      rvest_1.0.3     fansi_1.0.4      viridisLite_0.4.2
[17] scales_1.2.1    cli_3.6.1       rlang_1.1.1      munsell_0.5.0
[21] withr_2.5.0     yaml_2.3.7      tools_4.3.0      tzdb_0.4.0
[25] colorspace_2.1-0 webshot_0.5.4   pacman_0.5.1     vctrs_0.6.3
[29] R6_2.5.1        lifecycle_1.0.3 pkgconfig_2.0.3   pillar_1.9.0
[33] gtable_0.3.4    glue_1.6.2      systemfonts_1.0.4 xfun_0.39
[37] tidyselect_1.2.0 rstudioapi_0.14 farver_2.1.1      htmltools_0.5.5
[41] labeling_0.4.3  rmarkdown_2.22  svglite_2.1.1    compiler_4.3.0
```

Literaturverzeichnis

- Baayen, R. H. (2008). *Analyzing Linguistic Data: A Practical Introduction to Statistics Using R*.
- Baayen, R. H., & Shafaei-Bajestan, E. (2019). *languageR: Analyzing Linguistic Data: A Practical Introduction to Statistics*. <https://CRAN.R-project.org/package=languageR>
- Müller, K. (2020). *here: A Simpler Way to Find Your Files*. <https://CRAN.R-project.org/package=here>
- Nordmann, E., & DeBruine, L. (2022). *Applied Data Skills*. Zenodo. <https://doi.org/10.5281/zenodo.6365078>
- 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., Averick, M., Bryan, J., Chang, W., McGowan, L. D., François, R., Grolemund, G., Hayes, A., Henry, L., Hester, J., Kuhn, M., Pedersen, T. L., Miller, E., Bache, S. M., Müller, K., Ooms, J., Robinson, D., Seidel, D. P., Spinu, V., ... Yutani, H. (2019). Welcome to the tidyverse. *Journal of Open Source Software*, 4(43), 1686. <https://doi.org/10.21105/joss.01686>
- Wickham, H., Çetinkaya-Rundel, M., & Grolemund, G. (2023). *R for Data Science* (2. Aufl.).
- Winter, B. (2019). Statistics for Linguists: An Introduction Using R. In *Statistics for Linguists: An Introduction Using R*. Routledge. <https://doi.org/10.4324/9781315165547>
- Xie, Y. (2023). *tinytex: Helper Functions to Install and Maintain TeX Live, and Compile LaTeX Documents*. <https://github.com/rstudio/tinytex>

6. Bericht 1

Konsolidierung der neuen Kenntnisse

Dieser Bericht dient dazu, das bisher Gelernte zu wiederholen und zu festigen. Ihre Aufgaben umfassen das Laden von Paketen und Daten sowie eine leichte Datenverarbeitung (Kapitel 6.2). Außerdem werden Sie 4 Diagramme erstellen (Kapitel 6.3) und eine kurze Interpretation zu einem der Diagramme schreiben (Kapitel 6.4).

Ein Tipp: Ich empfehle Ihnen, Ihr Dokument häufig zu rendern, um eventuelle Fehler frühzeitig zu erkennen.

Sie müssen nur das Quarto-Skript einreichen, das auf meinem Rechner gerendert werden sollte (wenn es auf Ihrem gerendert wird, sollte es auch auf meinem gerendert werden).

6.1. Einrichtung

6.1.1. Quarto

Öffnen Sie ein neues Quarto-Skript und speichern Sie es als `nachname_vorname_bericht1.qmd`. Ändern Sie das YAML so, dass es einen:

- einen aussagekräftigen Titel
- Ihren Namen als **Autor**
- ein Inhaltsverzeichnis

Achten Sie darauf, Code Chunks, Prosa und Überschriften zu verwenden, um Ihre Aufgaben angemessen zu dokumentieren. Eine gute Faustregel ist, für jede (Unter-)Überschrift in diesem Dokument eine Überschrift hinzuzufügen.

6.1.2. Pakete

Laden Sie die Pakete `tidyverse` und `languageR` ein.

Tabelle 6.1.: ?(caption)

(a)

| Variable | Beschreibung |
|---------------------|--|
| Word | ein Faktor mit den Wörtern als Ebenen |
| Frequency | ein numerischer Vektor mit der absoluten Häufigkeit des Wortes im Spoken Dutch Corpus |
| Speaker | ein numerischer Vektor mit der absoluten Häufigkeit des Wortes im Spoken Dutch Corpus |
| Sex | ein Faktor mit den Lautsprechern als Ebenen |
| YearOfBirth | ein numerischer Vektor mit Geburtsjahren |
| DurationOfPrefix | ein numerischer Vektor mit der Dauer des Präfixes -ge in Sekunden. |
| SpeechRate | ein numerischer Vektor, der die Sprechgeschwindigkeit in Anzahl der Silben pro Sekunde kodiert |
| NumberSegmentsOnset | ein numerischer Vektor, der die Sprechgeschwindigkeit in Anzahl der Silben pro Sekunde kodiert |

6.1.3. Daten

Der Datensatz `durationsGe` aus dem `languageR`-Paket (Baayen & Shafaei-Bajestan, 2019) enthält Dauermessungen zur niederländischen Vorsilbe *ge*. Eine Beschreibung aller Variablen des Datensatzes findet sich in Tabelle 6.1. Ihre Aufgabe ist es:

1. Speichern Sie den Datensatz als Objekt `df_ge` in Ihrer Umgebung (dies kann auf die gleiche Weise geschehen wie bei allen Datensätzen, die wir bisher verwendet haben)
2. Drucken Sie die ersten 10 Zeilen des Datensatzes mit der Funktion “`head()`” aus.

6.2. Data wrangling

Hier werden Sie die `dplyr`-Verben aus Woche 4 verwenden. Denken Sie daran, dass Sie den Zuweisungsoperator (`<-`) nur verwenden müssen, wenn Sie die Änderungen, die Sie vornehmen, als Objekt in der Umgebung speichern wollen. Wenn Sie diese Änderungen nur ausdrucken wollen, brauchen Sie den Zuweisungsoperator nicht.

6.2.1. Subsetting

Drucken (aber nicht in Ihrer Umgebung speichern) Sie die Zeilen von `df_ge`, in denen `SpeechRate` über 9 liegt, nur mit den Spalten `word`, `speaker` und `SpeechRate`. Es sollten 5 Zeilen sein.

6.2.2. `mutate()`

Fügen Sie eine neue Variable hinzu, `duration_ms`, die `DauerVonPräfix` multipliziert mit 1000 (`DurationOfPrefix*1000`) entspricht. Dies entspricht der Dauer von `ge` in Millisekunden, statt in Sekunden. Stellen Sie sicher, dass Sie diese neue Variable in Ihrem Datenrahmen speichern (Hinweis: Sie müssen den Zuweisungsoperator `<-` und das `dplyr`-Verb `mutate()` verwenden).

6.2.3. Fehlersuche

Warum läuft dieser Code nicht? Es gibt zwei Probleme mit dem Code, identifizieren und beheben Sie sie.

```
## Troubleshooting
df_ge |>
  select(Frequency, word) +
  filter(YearOfBirth == 1978)
```

6.3. Datenvisualisierung

Verwenden Sie für alle Diagramme `labs(title = "...")`, um entsprechende Diagrammtitel hinzuzufügen.

Optional: Ändern Sie die `x` und `y` Achsenbeschriftungen, wenn Sie wollen, mit `labs(x = "...", y = "...")`. Vielleicht möchten Sie auch ein Thema hinzufügen (z.B. `theme_minimal()`).

6.3.1. Scatterplot

Erstellen Sie ein Streudiagramm mit `SpeechRate` (x-Achse) und `DurationOfPrefix` (y-Achse), mit `YearOfBirth` als Farbe (`colour`). Ändern Sie die Einstellungen für den Codechunk so, dass das Diagramm beim Rendern des Skripts *nicht* gedruckt wird, der Code aber schon. Tipp: Sie müssen `#| eval:` verwenden.

6.3.2. Facetten

Fügen Sie Facetten für **Sex** hinzu (denken Sie daran, die Tilde ~ einzufügen). Ändern Sie die Code-Chunk-Einstellungen so, dass die Darstellung gedruckt wird, wenn das Skript gerendert wird, aber der Code nicht (Sie benötigen `echo` anstelle von `eval`).

6.3.3. Reproduzieren eines Plots

Reproduzieren Sie die Abbildung 6.1 (es muss keine exakte Kopie sein, aber kommen Sie ihr so nahe wie möglich). Stellen Sie sicher, dass sowohl der Code als auch die Darstellung beim Rendern gedruckt werden. Hinweis: Sie müssen `filter()` sowohl für **Frequency** als auch für **Sex** verwenden. Ich würde mich darauf konzentrieren, zuerst das Diagramm zu erstellen und dann zu versuchen, die Daten zu filtern.

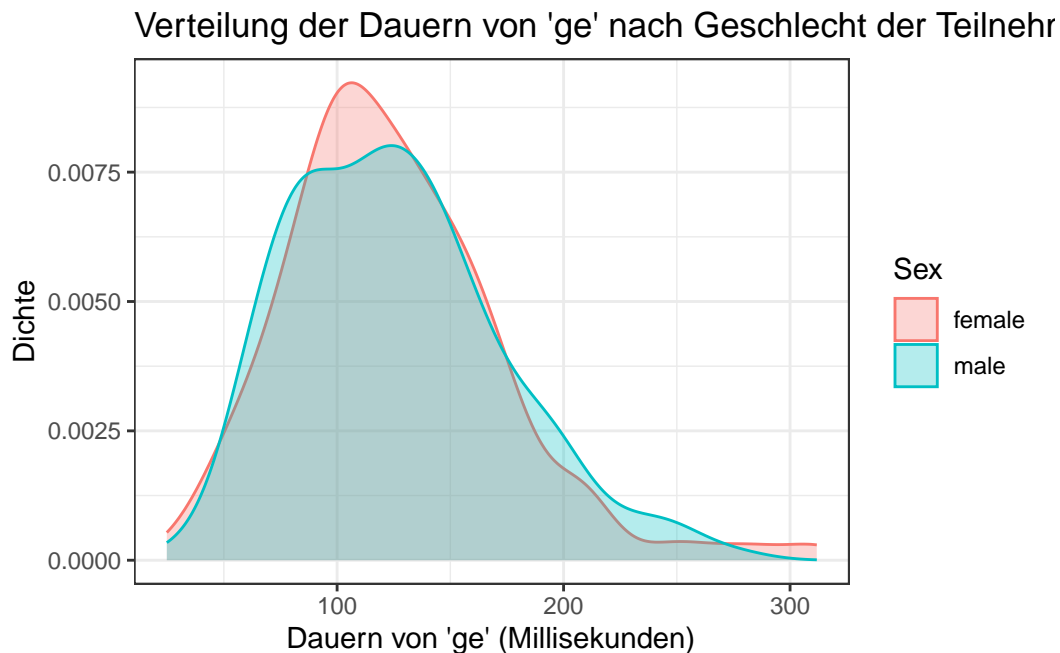


Abbildung 6.1.: Eine zu reproduzierende Figur

6.4. Interpretation

Beschreiben Sie die Beziehung zwischen den beiden Variablen, die Sie in Abbildung 6.1 sehen.

Teil III.

Nächste Stufe

7. Einlesen von Daten

Importieren von Datendateien

Die Materialien werden vor dem Unterricht zur Verfügung gestellt.

Lesungen

Die **Pflichtlektüre** zur Vorbereitung auf dieses Thema ist [Kap. 8 \(Data Import\)](#) in Wickham et al. (2023).

Eine **ergänzende Lektüre** ist [Ch. 4 \(Data Import\)](#) in Nordmann & DeBruine (2022).

8. Deskriptive Statistik

Maße der zentralen Tendenz und Streuung

Die Materialien werden vor dem Unterricht zur Verfügung gestellt.

Lesungen

Die **Pflichtlektüre** zur Vorbereitungen auf dieses Thema sind

1. Kap. 3, Abschnitt 3.4-3.9 (Descriptive statistics, models, and distributions) in Winter (2019) (online verfügbar über das [HU Grimm Zentrum](https://hu-berlin.hosted.exlibrisgroup.com/permalink/f/uig076/TN_cdi_askewsholts_vlebooks_9781351677431) unter https://hu-berlin.hosted.exlibrisgroup.com/permalink/f/uig076/TN_cdi_askewsholts_vlebooks_9781351677431).
2. [Bereich 4.5 \(Groups\)](#) in Kapitel 4 (Data Transformation) in ([wickham__tidyverse__2023?](#)).

9. Data Wrangling 2

Datenbereinigung (Data tidying)

Die Materialien werden vor dem Unterricht zur Verfügung gestellt.

Lesungen

Die **Pflichtlektüre** zur Vorbereitung auf dieses Thema ist [Kapital 6 \(Data tidying\)](#) in Wickham et al. (2023).

Eine **ergänzende Lektüre** ist [Kapital 8 \(Data tidying\)](#) in Nordmann & DeBruine (2022).

10. Datenvisualisierung 2

Visualisierung von Beziehungen

Die Materialien werden vor dem Unterricht zur Verfügung gestellt.

Lesungen

Die **Pflichtlektüre** zur Vorbereitung auf dieses Thema ist [Bereich 2.5 \(Visualising relationships\)](#) in Wickham et al. (2023).

Eine **ergänzende Lektüre** ist [Kapital 4 \(Representing summary statistics\)](#) in Nordmann et al. (2022).

Bericht 2

Teil IV.

Fortgeschrittene Themen

Literaturverzeichnis

- Baayen, R. H. (2008). *Analyzing Linguistic Data: A Practical Introduction to Statistics Using R*.
- Baayen, R. H., & Shafaei-Bajestan, E. (2019). *languageR: Analyzing Linguistic Data: A Practical Introduction to Statistics*. <https://CRAN.R-project.org/package=languageR>
- Müller, K. (2020). *here: A Simpler Way to Find Your Files*. <https://CRAN.R-project.org/package=here>
- Nordmann, E., & DeBruine, L. (2022). *Applied Data Skills*. Zenodo. <https://doi.org/10.5281/zenodo.6365078>
- 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., Averick, M., Bryan, J., Chang, W., McGowan, L. D., François, R., Golemund, G., Hayes, A., Henry, L., Hester, J., Kuhn, M., Pedersen, T. L., Miller, E., Bache, S. M., Müller, K., Ooms, J., Robinson, D., Seidel, D. P., Spinu, V., ... Yutani, H. (2019). Welcome to the tidyverse. *Journal of Open Source Software*, 4(43), 1686. <https://doi.org/10.21105/joss.01686>
- Wickham, H., Çetinkaya-Rundel, M., & Golemund, G. (2023). *R for Data Science* (2. Aufl.).
- Winter, B. (2019). Statistics for Linguists: An Introduction Using R. In *Statistics for Linguists: An Introduction Using R*. Routledge. <https://doi.org/10.4324/9781315165547>
- Xie, Y. (2023). *tinytex: Helper Functions to Install and Maintain TeX Live, and Compile LaTeX Documents*. <https://github.com/rstudio/tinytex>