Angewandte Datenverarbeitung und Visualisierung (WiSe23/24)

WiSe23/24

Daniela Palleschi

2023-10-16

Inhaltsverzeichnis

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 sich 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 (Introducti
2	25.10.2023	Data Viz 1: Verteilungen	R4DS - Ch 2 (Data visu
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 tran
5	15.11.2023	Data Viz 2: Visualisierung von Beziehungen	R4DS - Ch 5 (Workflow:
6	22.11.2023	Bericht 1	
7	29.11.2023	Daten einlesen	R4DS - Ch 8 (Data imp
8	06.12.2023	Wrangling 2: Tidying data	R4DS - Ch 6 (Data tidy
9	13.12.2023	Deskriptive Statistik	Winter (2019) - Ch 3 (D
10	20.12.2023	Data Viz 3: Visualisierung von Zusammenfassungen	R4DS - Ch 2 (Data visu
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 g
13	24.01.2024	Regular expressions	R4DS - Ch 16 (Regular
14	31.01.2024	Data Viz 4: Kommunikation	R4DS - Ch 12 (Commun
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 anfing, 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

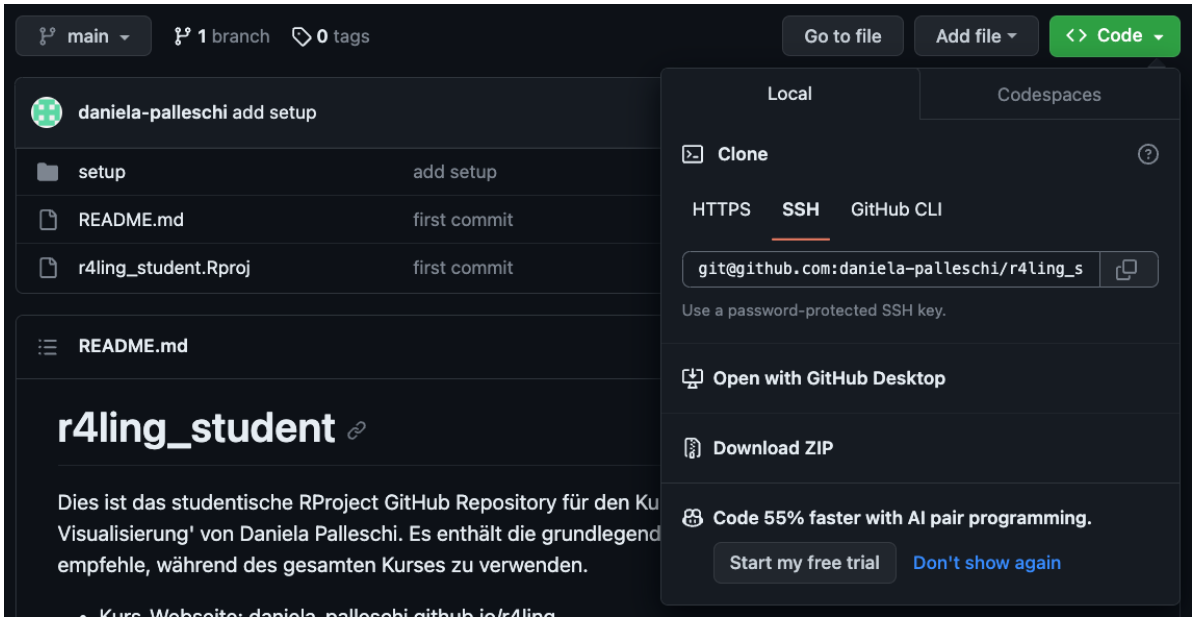


Abbildung 1.1.: Download GitHub repositiory

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

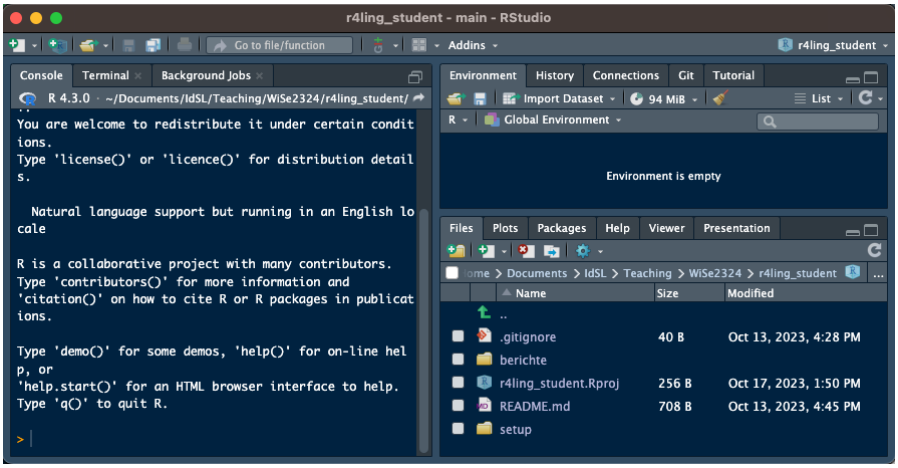


Abbildung 1.2.: Student RProject

> **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. Erweitungspakete

- 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(beepr)
```

1.3.2.1. Verwendung einer Funktion

- Sobald Sie ein Paket geladen haben, können Sie auf dessen Funktionen zugreifen
- Zum Beispiel hat das Paket `beepr` 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`

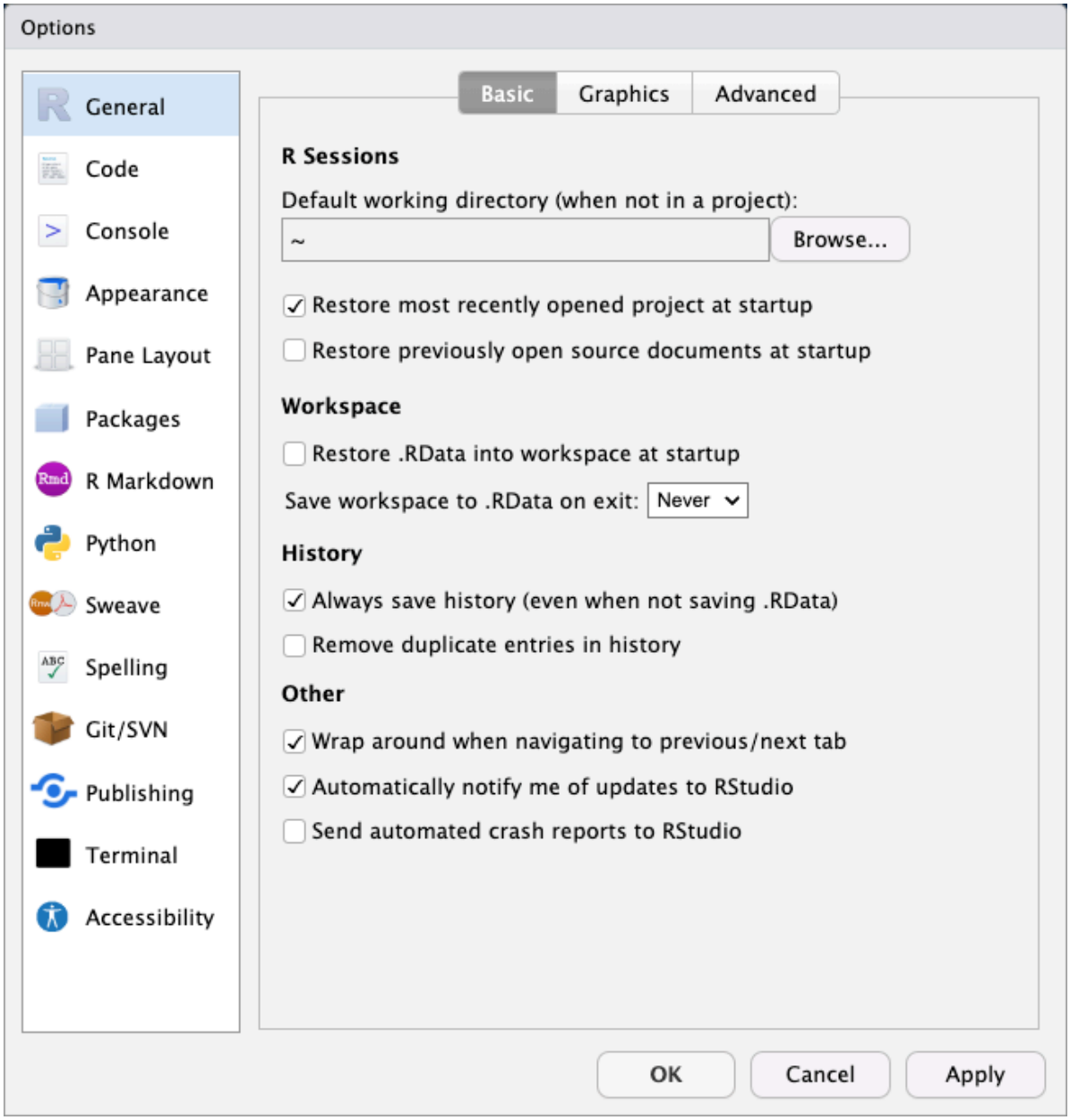


Abbildung 1.3.: Ihre 'Global Options' sollten wie folgt aussehen

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)`

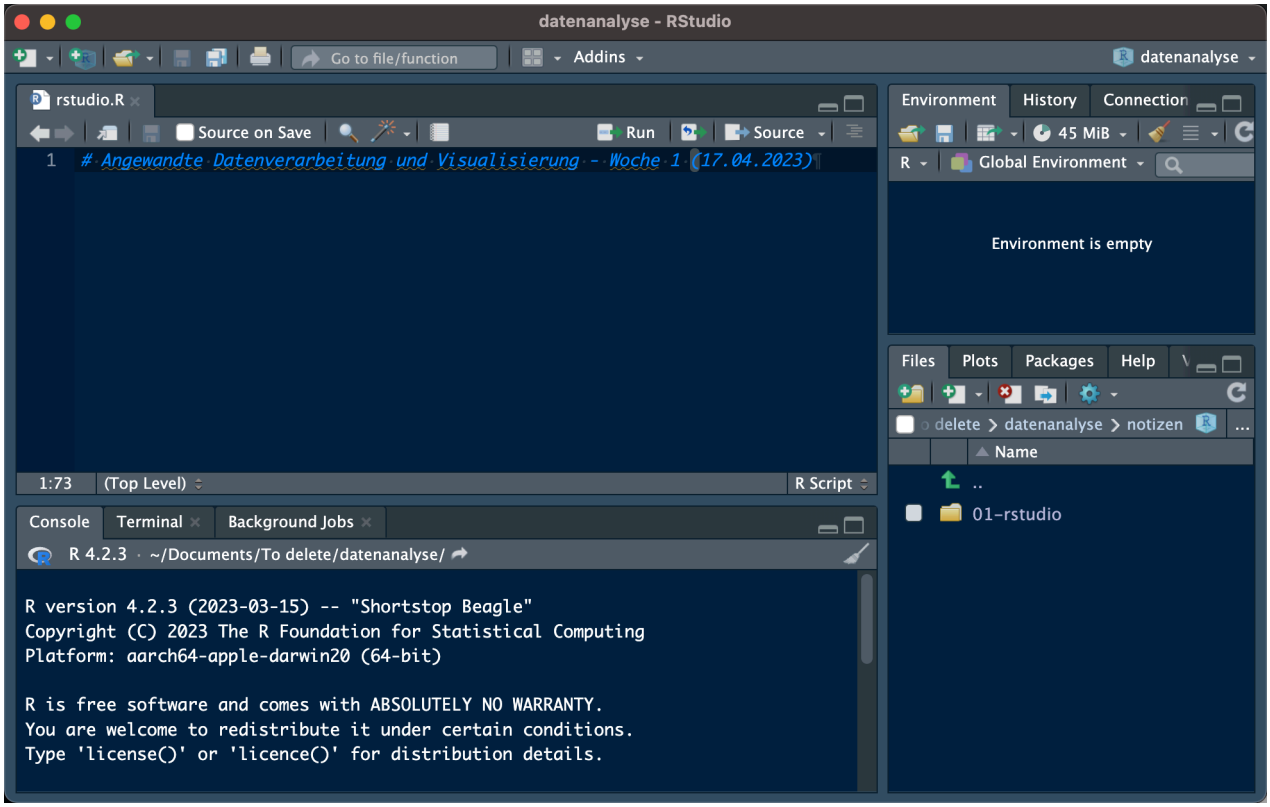


Abbildung 1.4.: Ihre Skript (oben links) sollten so aussehen

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

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

```
[1] 712
```

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

```
[1] 178
```

```r
# 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 Paketeversionen
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 Blosson).

```
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. Datenvisualiserung 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 “

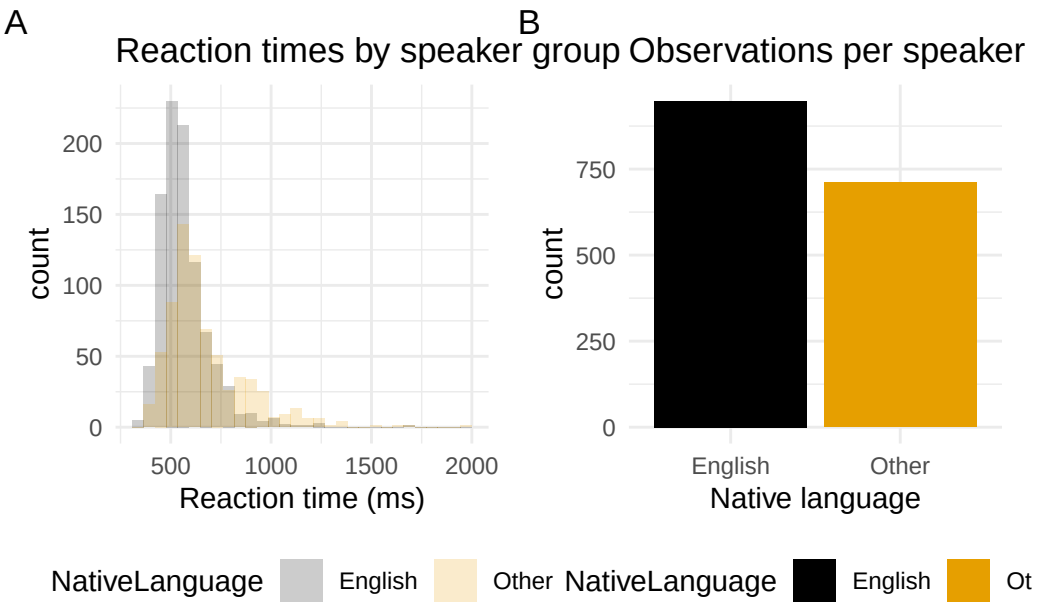
Heutige Ziele

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.

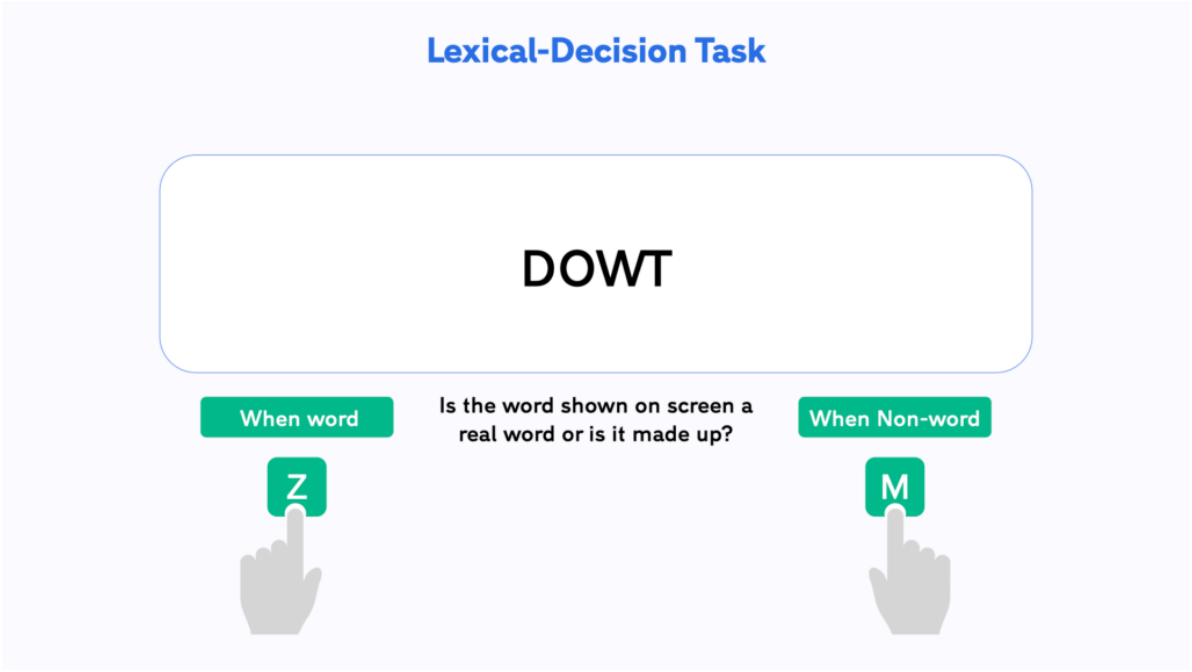


Abbildung 2.1.: Source: https://www.testable.org/wp-content/uploads/2022/11/Lexical_decision_task-1024x576.png

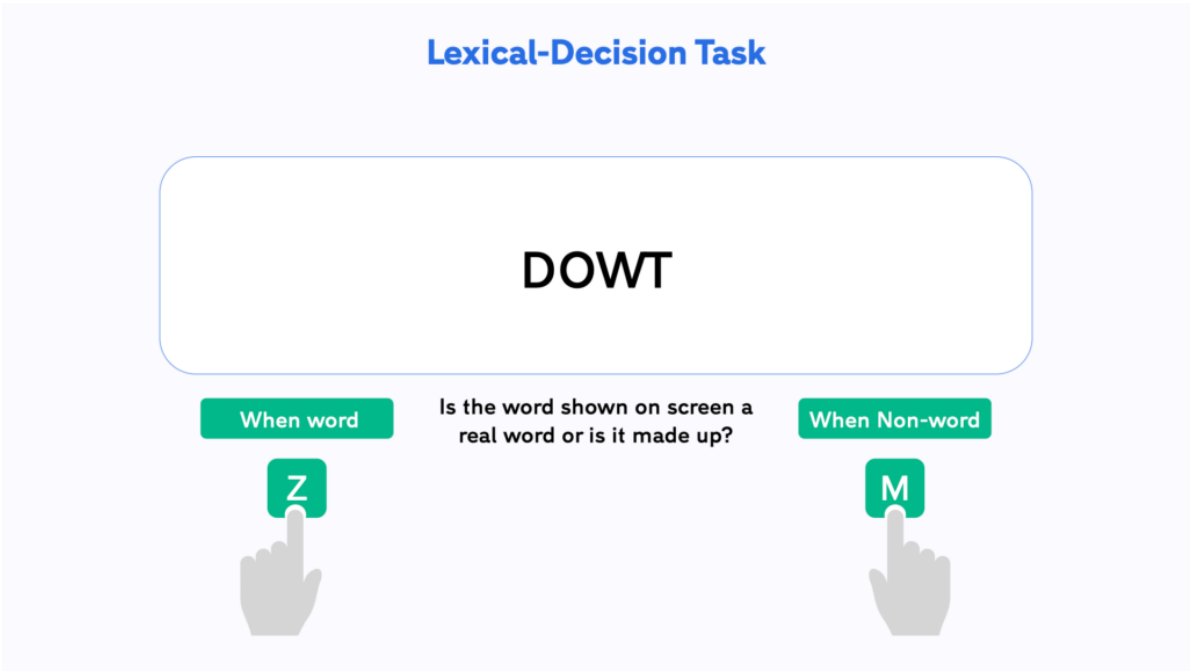


Tabelle 2.1.: Datenwörterbuch für `df_lexdec`: Lexikalische Entscheidungslatenzen, die von 21 Probanden für 79 konkrete englische Substantive erhoben wurden, mit Variablen, die mit dem Subjekt oder dem Wort verknüpft sind.

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

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 Enrivonment
  - 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)
```

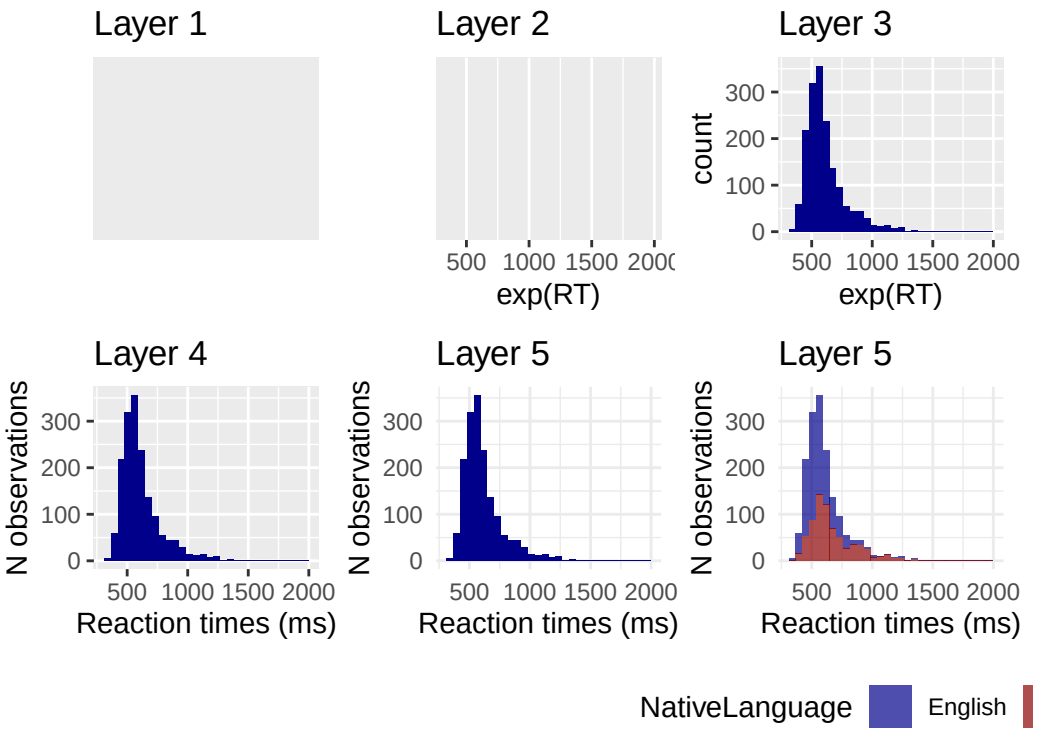
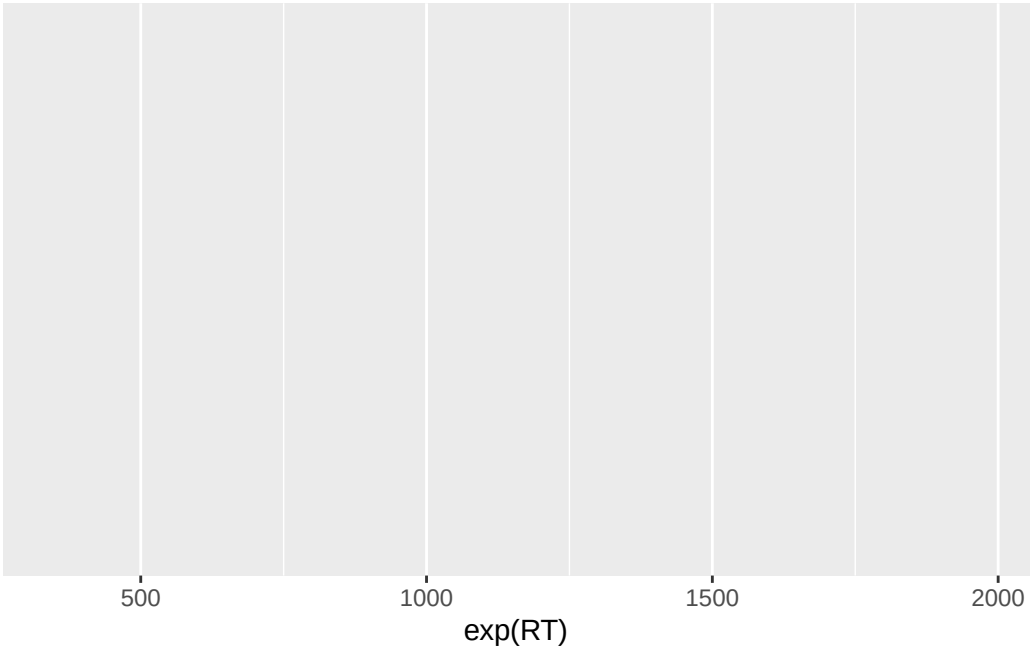


Abbildung 2.2.: Example of layers in a ggplot figure

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))
```



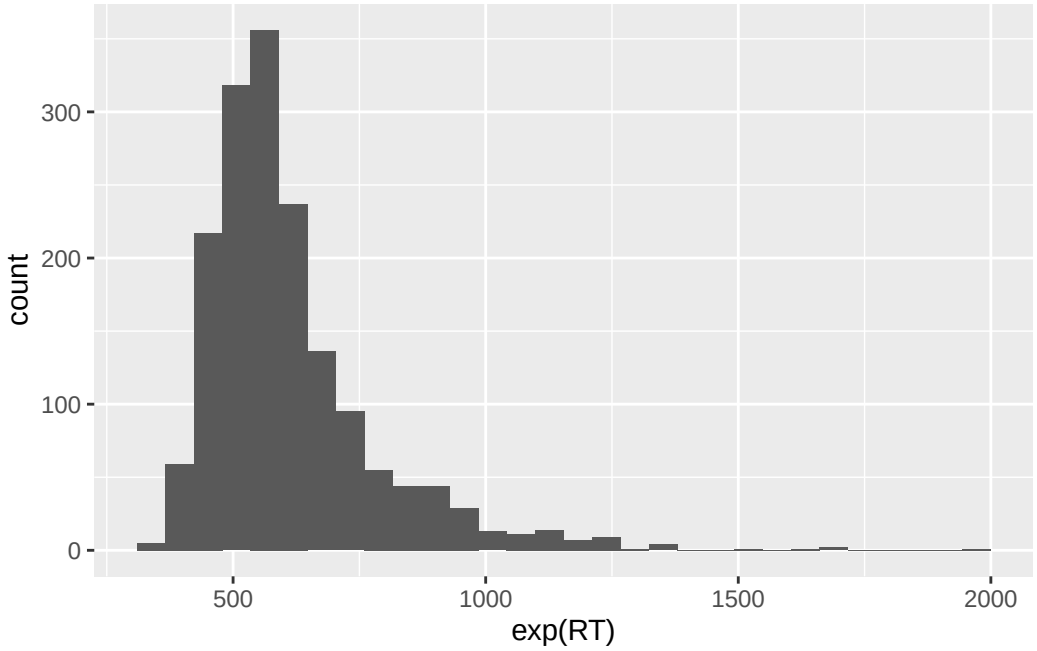
Aufgabe 2.2: Ästhetische Kartierung

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 *geom*etrische 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()`

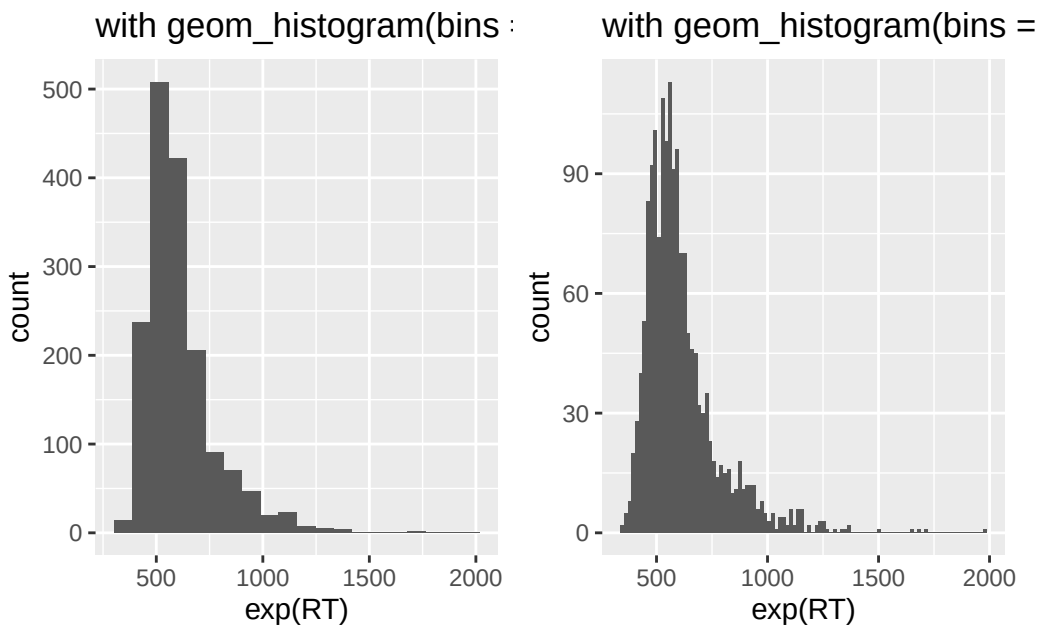
```r
ggplot(data = df_lexdec) +
  aes(x = exp(RT)) +
  geom_histogram()
```



> **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

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

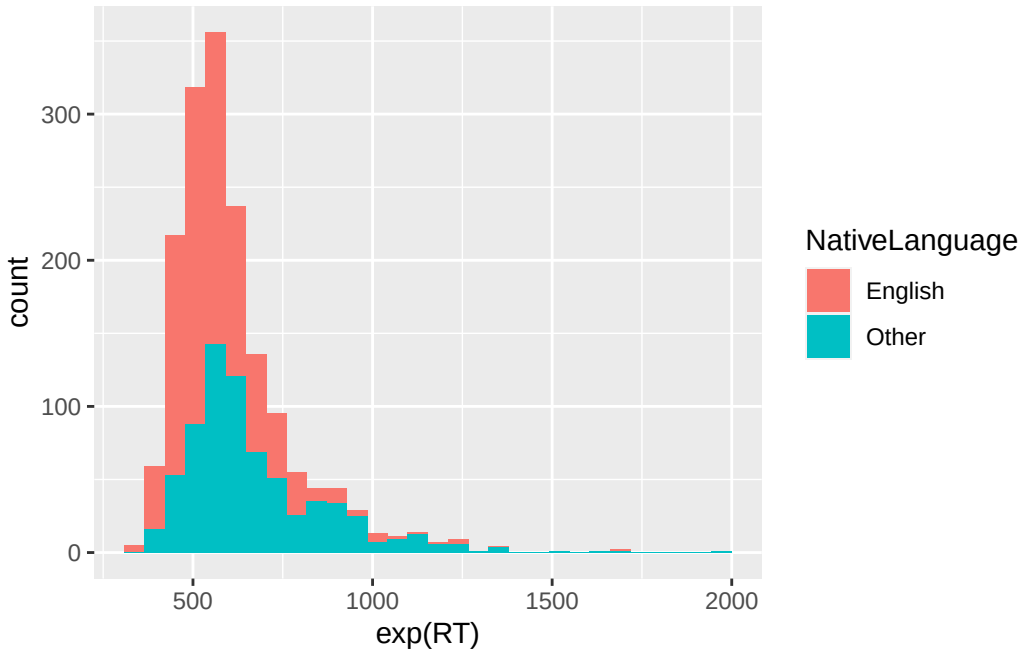
```
  ggplot(
  data = df_lexdec,
  mapping = aes(x = exp(RT))
) +
  labs(title = "with geom_histogram(bins = 100)") +
  geom_histogram(bins = 100)
```



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

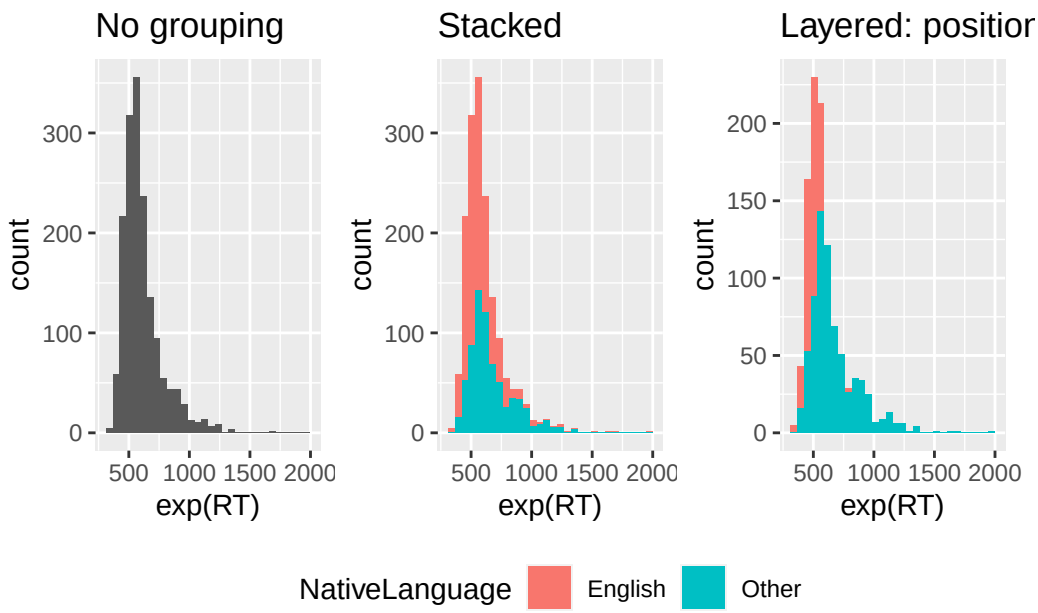
```
ggplot(
  data = df_lexdec,
  aes(x = exp(RT))
) +
  labs(title = "No grouping") +
  geom_histogram() +

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')
```

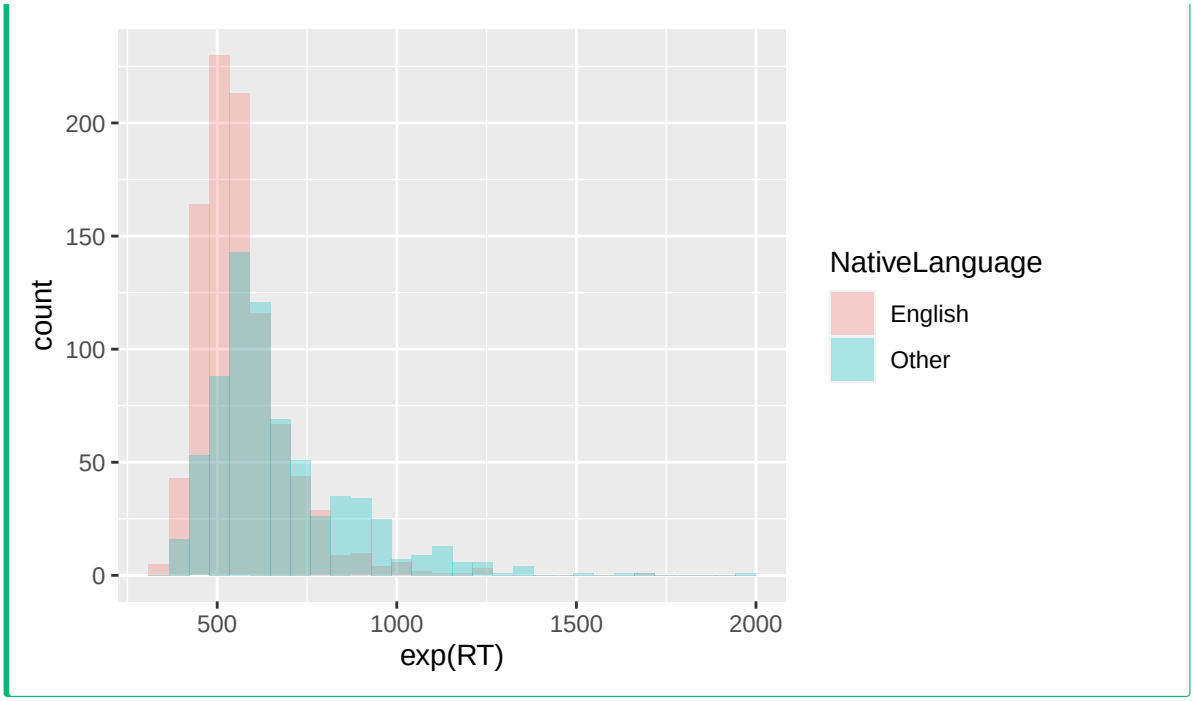


2.4.5. Globale und lokale Ästhetik

- 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

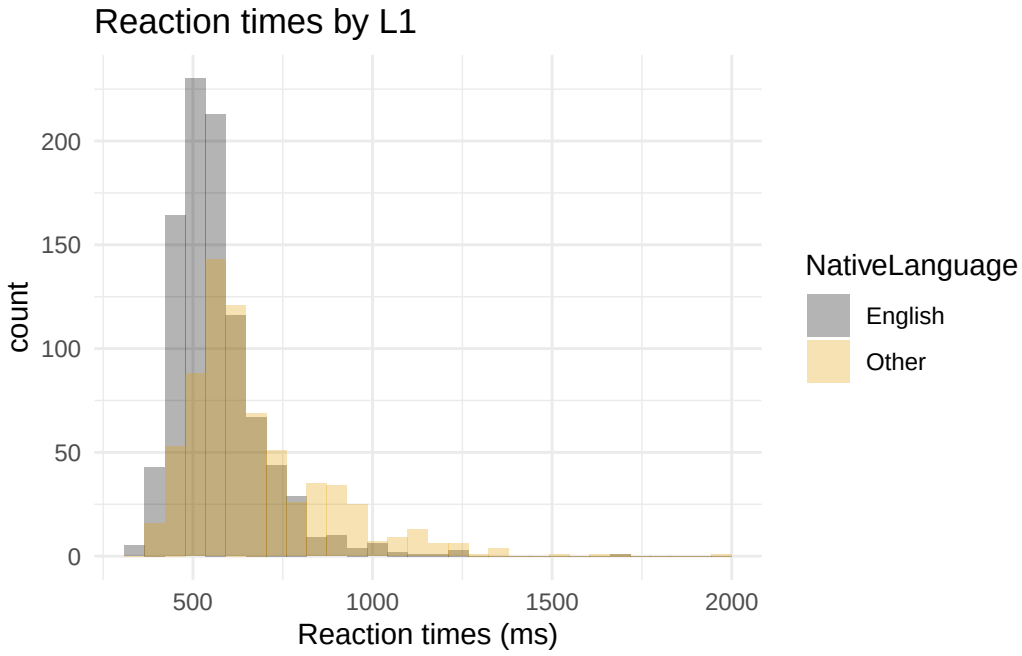
- wir können unsere Achsen- und Legendenbeschriftungen verbessern und auch Titel hinzufügen, indem wir die Funktion `labs()` verwenden
- Wir können auch die Funktion `scale_fill_colorblind()` aus dem Paket `ggthemes` verwenden.
  - dies erzeugt farbenblind-sichere Farben
- Wir werden auch die Funktion `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

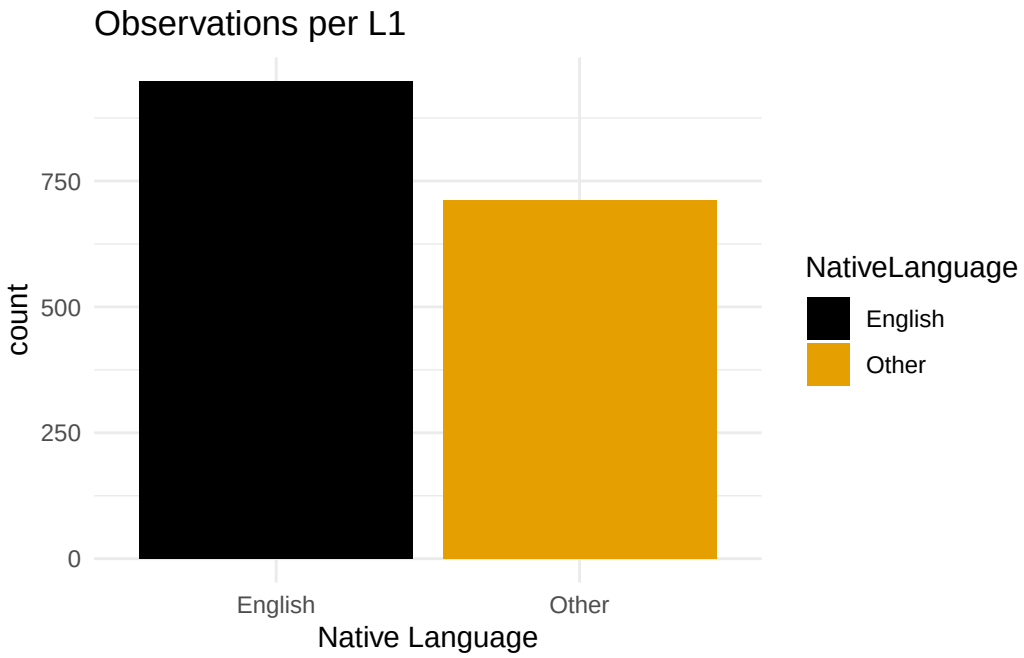
Beispiel 2.4.

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:

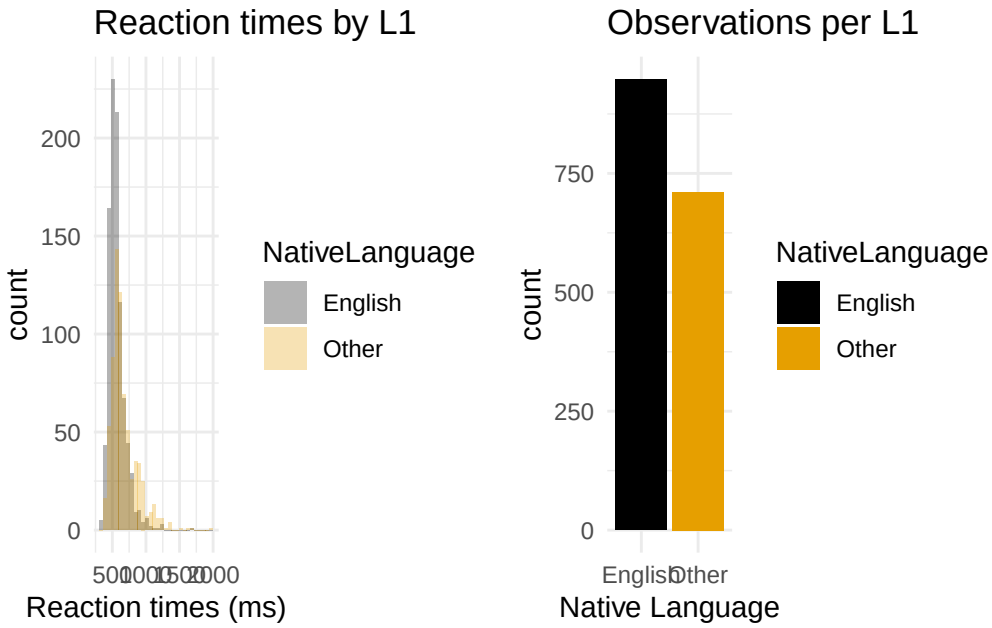


2.4.10. Kombinieren von Plots

- 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

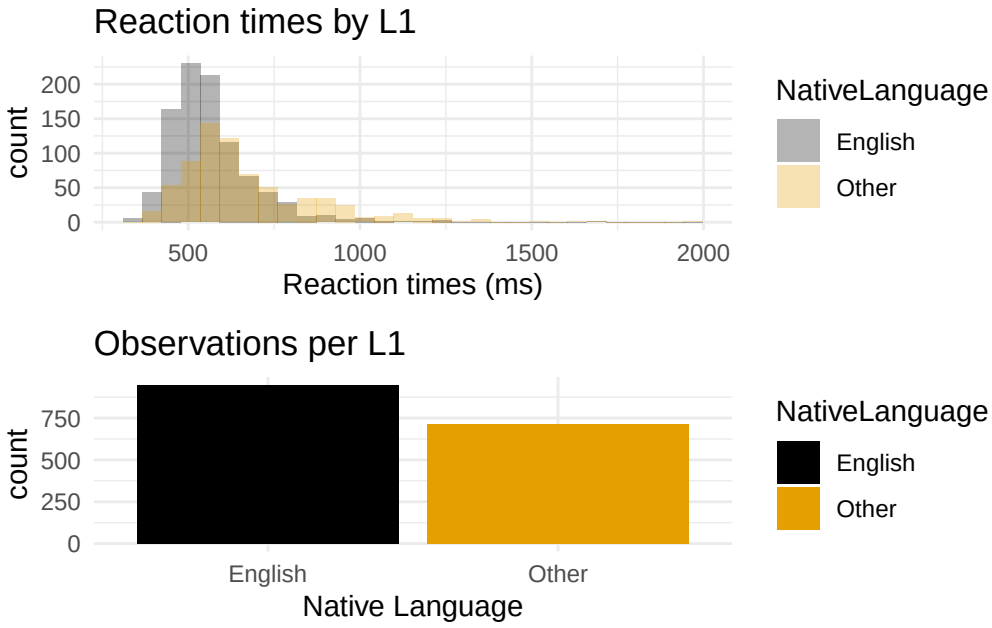
2.4.10.1. Kombinieren von Plots mit +

```
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

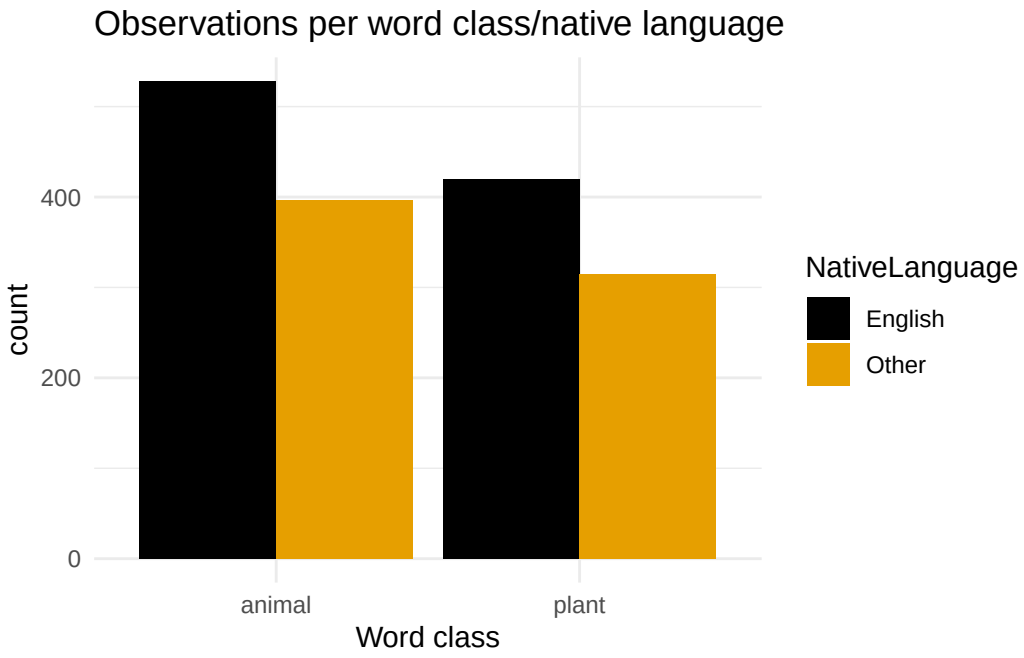
- Warum verwenden wir ein Histogramm für die Reaktionszeit und ein Balkendiagramm für die Muttersprache?
- Um welche Arten von Variablen handelt es sich?
  - Reaktionszeit ist kontinuierlich
  - Muttersprache ist eine kategoriale Variable
- Wir verwenden Histogramme, um die Verteilungen von *kontinuierlichen* Variablen zu visualisieren.
- Wir verwenden Balkendiagramme, um Verteilungen von *kateogrischen* 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;  L

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()
```

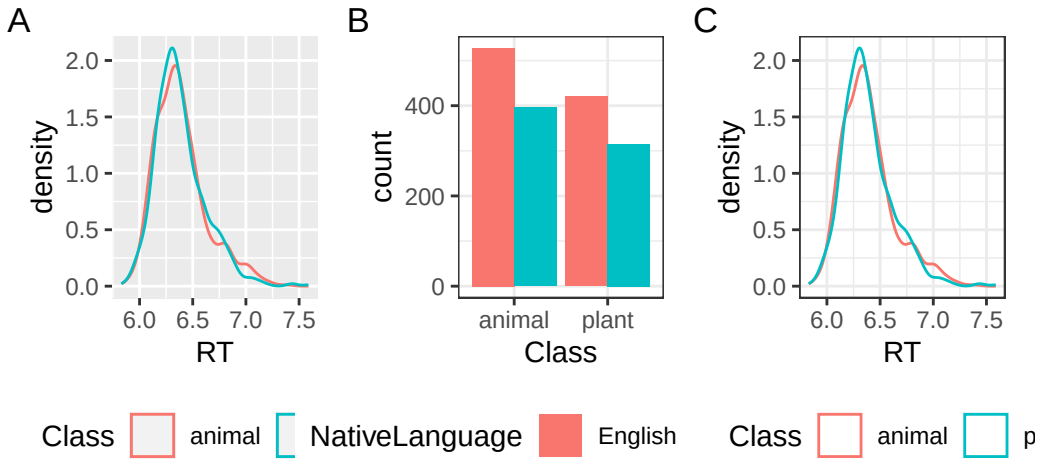


Abbildung 3.1.: Drei aus dem lexdec-Datensatz generierte Diagramme

Set-up

- wir müssen eine LaTeX-Distribution verwenden, um PDF-Dokumente mit Quarto zu erstellen
  - LaTeX ist ein Schriftsatzsystem
  - 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 Ihren Computer
   - Sehen Sie das Dokument an; Sie können oben rechts auf verschiedene Schaltflächen klicken. Probieren Sie es.

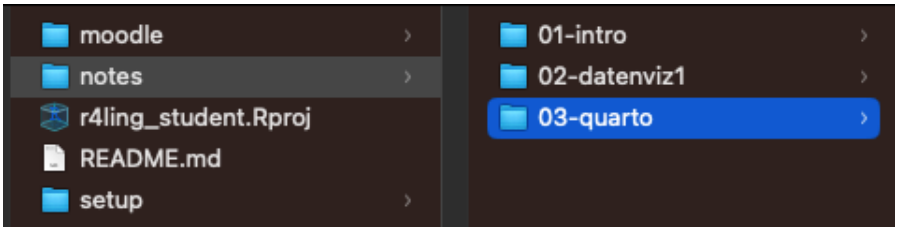


Abbildung 3.2.: Notes folder structure

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 geschiehen? 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, Autorenname
- 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 Schriftsatz. 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. Kopiern 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 sehent 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 *kursiv*`	Dieser Text ist *kursiv*
Fett	`Dieser Text ist **fett**`	Dieser Text ist **fett**
Subskription	`Dieser Text ist ~tiefgestellt~`	Dieser Text ist$_{\text{tiefgestellt}}$
Hochgestelt	`Dieser Text ist ^hochgestellt^`	Dieser Text ist $^{\text{hochgestellt}}$

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 (beachtt 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 (beachtt 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` verwendt, 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?

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-Chunken 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
```

```
[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 verwendt.
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
```

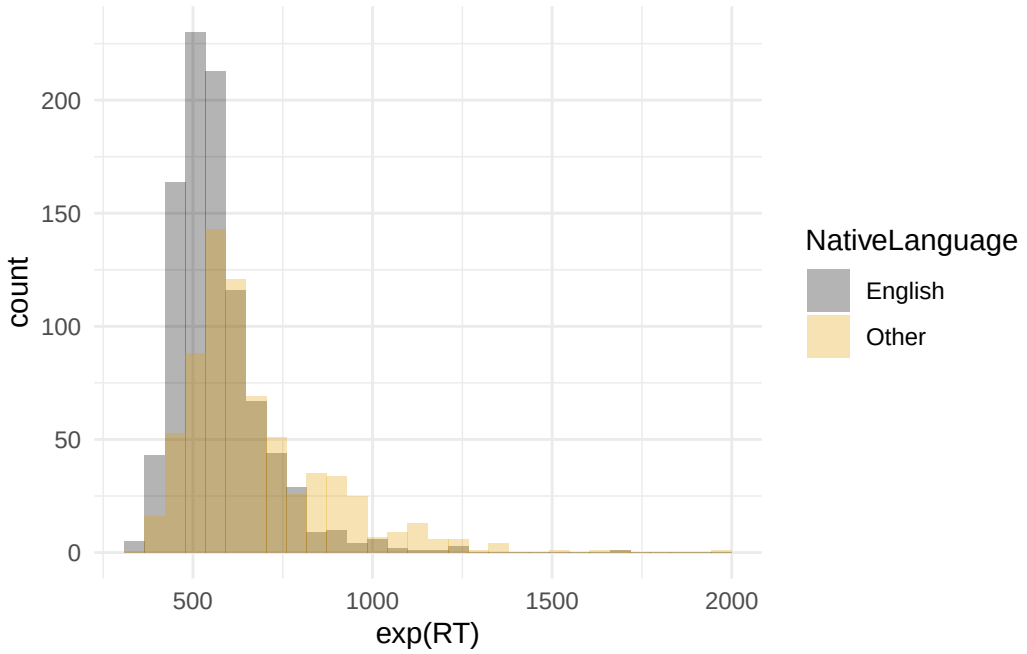


Abbildung 3.3.: Histogram of reactiontimes per native language from lexdec

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.

```r
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()
```

```r
fig_lexdec_l1
```

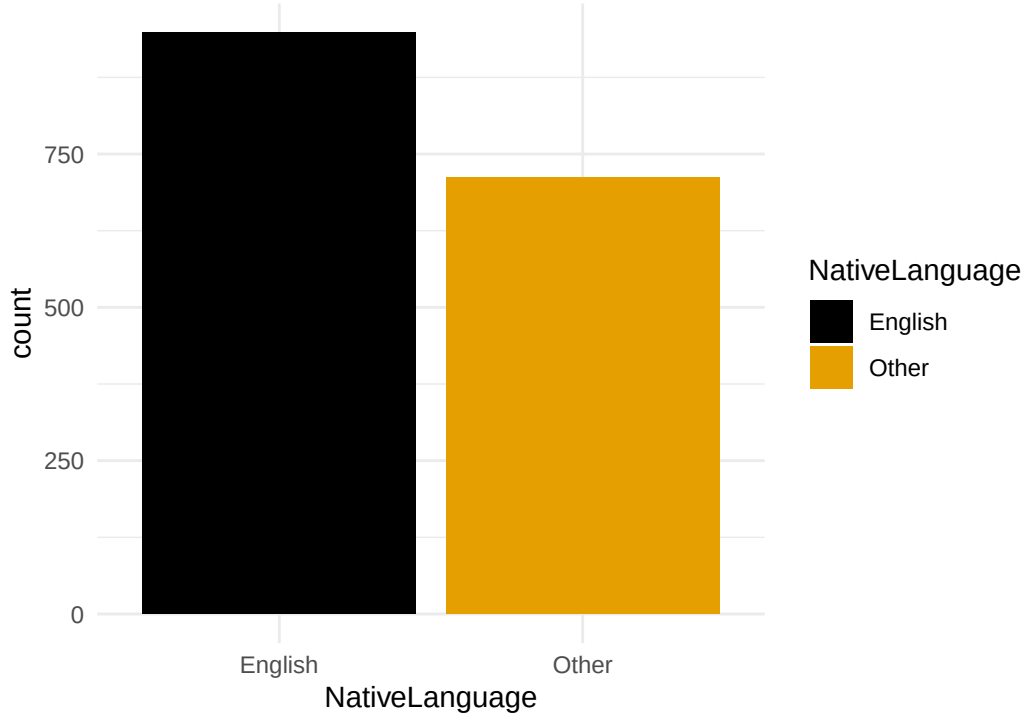


Abbildung 3.4.: Barplot of observations per native language

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 seht?
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 seht?
    - 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 wiedergebt

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;  L

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., 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

4. Data Wrangling 1: Transformation

Umwandlung von Daten

Die Materialien werden vor dem Unterricht zur Verfügung gestellt.

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).

5. Datenvisualisierung 2

Visualisierung von Beziehungen

Die Materialien werden vor dem Unterricht zur Verfügung gestellt.

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).

Bericht 1

Die Anweisungen werden am Tag der Vorlesung veröffentlicht.

Teil III.

Nächste Stufe

6. 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).

7. 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 unter https://hu-berlin.hosted.exlibrisgroup.com/permalink/f/uig076/TN_cdi_askewsholts_vlebooks_9781351677431).
2. Bereich 4.5 (Groups) in Kapital 4 (Data Transformation) in (**wickham_tidyverse_2023?**).

8. 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 relationsips) 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

Teil V.

Literaturverzeichnis

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