Datenimport

Einlesen lokaler Datendateien

Daniela Palleschi

Mi. den 29.11.2023

Inhaltsverzeichnis

Lektüre

- **Pflichtlektüre**: Kap. 8 (Datenimport) in Wickham et al. (2023)
- **Ergänzende Lektüre**: Kap. 4 (Datenimport) in Nordmann & DeBruine (2022)

Wiederholung

Bis jetzt haben wir gelernt, wie man…

- Quarto-Skripte für die reproduzierbare Datenanalyse zu verwenden
- eingebaute Datensätze zu laden
- Daten mit `dplyr`-Verben zu verarbeiten
- Verteilungen und Beziehungen zwischen verschiedenen Variablentypen zu visualisieren

Lernziele

Heute werden wir lernen, wie man:

- lokale Datendateien (`.csv`) erstellen und speichern
- lokale Datendateien mit dem Paket `readr` importieren
- mit fehlenden Werten umzugehen
- Variablen in Faktoren umwandeln

1 Einrichtung

1.1 Pakete mit `pacman`

- wir fangen an, das Paket `pacman` anstelle von `install.packages()` und `library` zu benutzen
  - die Funktion `p_load()` nimmt Paketnamen als Argumente
  - prüft dann, ob Sie das Paket installiert haben
    - wenn ja –> lädt das Paket (genau wie `library()`)
    - wenn nicht –> wird das Paket installiert und dann geladen (wie mit `install.packages()` + `library()`)
- dies erspart uns die individuelle Installation neuer Pakete

---

```r
# install new packages IN THE CONSOLE!
install.packages("pacman")
```

```r
# load packages
pacman::p_load(tidyverse, # wrangling
               janitor, # wrangling
               here # relative file paths
               )
```

- wir haben jetzt `tidyverse` geladen und die neuen Pakete `janitor` und `here` installiert und geladen
  - Um mehr über diese Pakete herauszufinden, geben Sie `?janitor` und `?here` in der Konsole ein.
- fügen Sie Ihrem Projektverzeichnis einen Ordner mit dem Namen `daten` hinzu (der *genau* gleich geschrieben ist).

RProjects

- Stellen Sie sicher, dass Sie in der Klasse RProject arbeiten!
- Falls nicht, folgen Sie der Übung auf der Kurs-Website hier

2 CSV: Comma separated value

- Es gibt viele verschiedene Dateitypen, die Daten annehmen können, z. B. .xlsx, .txt, .csv, .tsv.
- csv" ist der typischste Dateityp und steht für: Comma Separated Values.
- So sieht eine einfache CSV-Datei aus, wenn man sie als Rohtext betrachtet

```
Student ID,Full Name,favourite.food,mealPlan,AGE
1,Sunil Huffmann,Strawberry yoghurt,Lunch only,4
2,Barclay Lynn,French fries,Lunch only,5
3,Jayendra Lyne,N/A,Breakfast and lunch,7
4,Leon Rossini,Anchovies,Lunch only,
5,Chidiegwu Dunkel,Pizza,Breakfast and lunch,five
6,Güvenç Attila,Ice cream,Lunch only,6
```

- die erste Zeile (die "Kopfzeile") enthält die Spaltennamen
- die folgenden Zeilen enthalten die Daten
- Wie viele Variablen gibt es? Wie viele Beobachtungen?

2.1 'Tidy' Daten

- Sie wollen, dass Ihre Daten *aufgeräumt* sind
  - aufgeräumte Daten sind rechteckig, und:
  - jede Spalte steht für eine Variable
  - jede Zeile eine Beobachtung
  - jede Zelle ein Datenpunkt (**?@fig-tidy-data**)

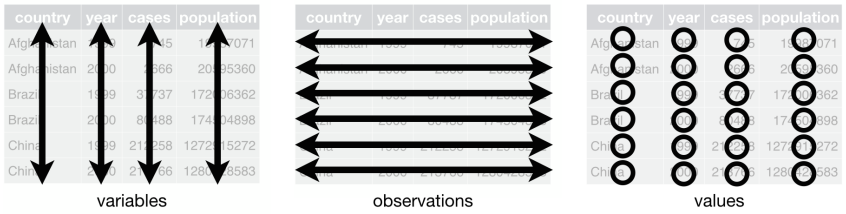

Abbildung 1: Source: Wickham et al. (2023) (all rights reserved)

2.2 Tabelle zu csv

- Lassen Sie uns einige Spielzeugdaten in einer Tabellenkalkulation sammeln, die wir dann als CSV-Datei speichern und in R laden werden
  - Klicken Sie hier, um zu einem bearbeitbaren Arbeitsblatt zu gelangen.
  - Geben Sie die relevanten Informationen über sich selbst ein, oder erfinden Sie einige Daten: den Namen eines Haustiers, das Sie haben/hatten, Größe, Geburtsmonat und -tag sowie Ihre erste Sprache. Wenn Sie kein Haustier haben, lassen Sie die Zelle leer.

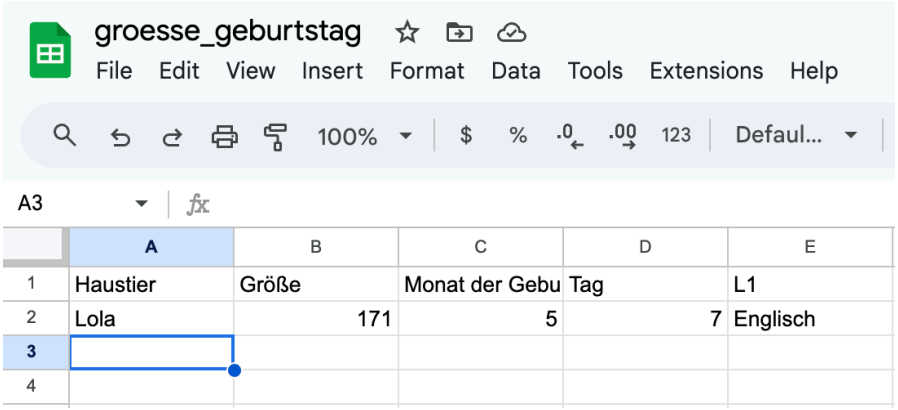

Abbildung 2: Our spreadsheet

2.3 CSV speichern

- Speichern Sie die Tabelle als `groesse_geburtstag.csv` auf Ihrem Computer, direkt in einem Ordner namens `daten` in unserem Projektverzeichnis

Aufgabe 2.1: `Saving a CSV`

Beispiel 2.1.

1. Erstellen Sie einen neuen Ordner mit dem Namen `daten` in Ihrem Projektordner (falls Sie das nicht schon getan haben).
2. Laden Sie das Google Sheet herunter und speichern Sie es in Ihrem `daten` Ordner als `groesse_geburtstag.csv`.
3. Gehen Sie zu Ihrem `daten`-Ordner und überprüfen Sie, ob die CSV-Datei dort ist.

3 Das Paket `readr`

- müssen wir nun *die Daten einlesen*
- wir müssen eine Funktion verwenden, die CSV-Daten liest, und angeben, *wo* sich die Daten in unserem RProject-Ordner befinden
- Das `readr`-Paket (Teil von `tidyverse`) kann die meisten Datentypen einlesen und hat mehrere Funktionen für verschiedene Datentypen

---

```
read_csv(here::here("daten", "groesse_geburtstag.csv"))
```

Tabelle 1: Data from the groesse_geburtstag.csv file as a table.

Haustier	Größe	Monat der Geburt	Tag	L1
Lola	171	5	7	Englisch
NA	168	11	26	Deutsch
N/A	182	4	15	Deutsch

Aufgabe 3.1: `readr`

Beispiel 3.1.

1. Importieren Sie den Datensatz "groesse_geburtstag.csv" und speichern Sie ihn als Objekt mit dem Namen `df_groesse`.
    - `df_` ist die Abkürzung für DataFrame; es ist eine gute Idee, ein Präfix vor Objektnamen zu verwenden, damit wir wissen, was jedes Objekt enthält
2. Beim Importieren von Daten mit `read_csv` werden einige Informationen in der Konsole ausgegeben. Was wird gedruckt?
3. Untersuche den Datensatz mit Funktionen wie `summary()` oder `head()`
4. Sehen Sie etwas Ungewöhnliches?

4 Das Paket `here`

- Woher weiß R genau, wo der Ordner `daten` zu finden ist?
- unser *Arbeitsverzeichnis* ist auf den Ort unseres RProjekts auf unserem Computer festgelegt
    - wann immer wir auf Daten in unserem RProjekt zugreifen wollen, sollten wir unseren Dateipfad in `here()` verschachteln
- um zu sehen, von wo aus `here()` startet, führen Sie `here()` aus

```
here()
```

```
[1] "/Users/danielapalleschi/Documents/IdSL/Teaching/WiSe2324/B.A./r4ling"
```

- Die Ausgabe wird auf allen Rechnern unterschiedlich aussehen, da sie relativ zu dem Ort ist, an dem wir unseren Projektordner abgelegt haben

5 Arbeiten mit Daten

5.1 Fehlende Werte

- Sie haben vielleicht einige `NA` oder `N/A` Werte bemerkt
    - `N/A` wurde als Text in einer unserer Beobachtungen geschrieben, und so liest R es als solches

Abbildung 3: Image source: Allison Horst (all rights reserved)

  - `NA` in R bezieht sich auf fehlende Daten ("Nicht verfügbar")
  - Echte fehlende Werte sind komplett leer, so dass `N/A` in unseren `df_groesse`-Daten nicht wirklich als fehlender Wert gelesen wird.
  - Um dies zu beheben, können wir das Argument `na =` für die Funktion `read_csv()` verwenden, das der Funktion `read_csv()` mitteilt, welche Werte sie mit fehlenden Werten gleichsetzen soll

```
# force "N/A" to missing values
df_groesse <- read_csv(here::here("daten", "groesse_geburtstag.csv"),
                       na = "N/A")
```

```
# print the head of the data set
head(df_groesse)
```

```
# A tibble: 3 x 5
  Haustier Größe `Monat der Geburt`   Tag L1
  <chr>    <dbl>              <dbl> <dbl> <chr>
1 "Lola"     171                  5     7 Englisch
2 ""         168                 11    26 Deutsch
3  <NA>      182                  4    15 Deutsch
```

- der Wert, der vorher `""` war, wird als `NA` gelesen
- aber was ist mit der leeren Zelle? Wir haben jetzt überschrieben, dass `read_csv()` leere Zellen als `NA` liest
  - Nun wollen wir `read_csv()` anweisen, *mehr als eine* Art von Eingabe als `NA` zu lesen, d.h. wir wollen es anweisen, `""` *und* `"N/A"` als `NA` zu lesen
  - Dazu verwenden wir unsere immer nützliche Verkettungsfunktion: `c()`

```
# force "N/A" and empty cells to missing values
df_groesse <- read_csv(here::here("daten", "groesse_geburtstag.csv"),
                       na = c("N/A",""))
```

```
# print the head of the data set
head(df_groesse)
```

```
# A tibble: 3 x 5
  Haustier Größe `Monat der Geburt`   Tag L1
  <chr>    <dbl>              <dbl> <dbl> <chr>
1 Lola       171                  5     7 Englisch
2 <NA>       168                 11    26 Deutsch
3 <NA>       182                  4    15 Deutsch
```

5.2 Spaltennamen

- Ein Spaltenname in unseren Daten ist von Backticks umgeben (z.B. `` `Monat der Geburt` ``)
  - Das liegt daran, dass er ein Leerzeichen enthält, das syntaktisch nicht gültig ist.
  - Eine schnelle Lösung ist die Funktion `clean_names()` aus dem Paket `janitor`, das wir bereits geladen haben

```
clean_names(df_groesse)
```

```
# A tibble: 3 x 5
  haustier grosse monat_der_geburt   tag l1
  <chr>     <dbl>            <dbl> <dbl> <chr>
1 Lola        171                5     7 Englisch
2 <NA>        168               11    26 Deutsch
3 <NA>        182                4    15 Deutsch
```

- Das sieht besser aus! Aber wenn Sie jetzt `head(df_groesse)` ausführen, sehen Sie dann die bereinigten Spaltennamen?
- Sie sollten nicht, denn wenn wir ein Objekt durch eine Funktion übergeben, wird das Objekt nicht 'aktualisiert'
  - Deshalb müssen wir das Objekt erneut mit dem Zuweisungsoperator `<-` zuweisen.

```
df_groesse <- janitor::clean_names(df_groesse)
```

5.3 Pipes

- Pipes werden am Ende eines Funktionsaufrufs platziert, wenn das Ergebnis dieser Funktion durch eine nachfolgende Funktion weitergegeben werden soll
  - Pipes können als "und dann…" gelesen werden

```
read_csv(here::here("daten", "groesse_geburtstag.csv")) |>
  head()
```

```
# A tibble: 3 x 5
  Haustier Größe `Monat der Geburt`   Tag L1
  <chr>    <dbl>              <dbl> <dbl> <chr>
1 Lola       171                  5     7 Englisch
2 <NA>       168                 11    26 Deutsch
3 N/A        182                  4    15 Deutsch
```

There are currently 2 pipes that can be used in R.

1. the `magrittr` package pipe: `%>%`
2. the new native R pipe: `|>`

- there aren't any major differences that are important for our current uses
- you can use the keyboard shortcut `Cmd/Ctrl + Shift/Strg + M` to produce a pipe

Aufgabe 5.1: pipes

Beispiel 5.1.

1. Laden Sie den Datensatz `groesse_geburtstag.csv` erneut mit festen `NA`s *und dann*
   - Benutzen Sie eine Pipe, um `clean_names()` für den Datensatz aufzurufen, *und dann*
   - rufen Sie die Funktion "head()" auf
   - Überprüfen Sie die Anzahl der Beobachtungen und Variablen, gibt es ein Problem?
2. Laden Sie den Datensatz `groesse_geburtstag.csv` erneut mit festen `NA`s, speichern Sie ihn als Objekt `df_groesse`, *und dann*
   - Verwenden Sie eine Pipe, um `clean_names()` auf den Datensatz anzuwenden.
3. Warum sollte man nicht eine Pipe und die Funktion "head()" verwenden, wenn man den Datensatz als Objekt speichert?

5.4 Variablentypen

- die wichtigsten Spaltentypen, die man kennen sollte, sind "numerisch" und "Faktor" (kategorisch)
- Faktoren enthalten *Kategorien* oder *Gruppen* von Daten, können aber manchmal *aussehen* wie `numerische` Daten
  - Unsere Spalte "Monat" enthält zum Beispiel Zahlen, aber sie könnte auch den Namen jedes Monats enthalten
  - Es ist sinnvoll, den Mittelwert einer "numerischen" Variable zu berechnen, aber nicht den eines "Faktors"
  - Es ist zum Beispiel sinnvoll, die durchschnittliche Körpergröße zu berechnen, aber nicht den durchschnittlichen Geburtsmonat

`as_factor()`

- Wir können die Funktion "as_factor()" verwenden, um einen Variablentyp in einen Faktor zu ändern.
- Wir können entweder die R-Basissyntax verwenden, um dies zu tun, indem wir ein `$` verwenden, um eine Spalte in einem Datenrahmen zu indizieren:

```
# mit base R
df_groesse$monat_der_geburt <- as_factor(df_groesse$monat_der_geburt)
```

- oder wir können die Syntax `tidyverse` und die Funktion `mutate()` verwenden

```
# mit tidyverse
df_groesse <-
  df_groesse |>
  mutate(monat_der_geburt = as_factor(monat_der_geburt))
```

6 Andere Dateitypen und Begrenzungszeichen

- `readr` hat weitere Funktionen, die ebenfalls einfach zu benutzen sind, man muss nur wissen, wann man welche benutzt
- `read_csv2()` liest Semikolon-getrennte csv-Dateien (`;`)
  - Dieser Dateityp ist in Ländern üblich, die `,` als Dezimaltrennzeichen verwenden (wie Deutschland)
- `read_tsv()` liest Tabulator-getrennte Dateien
- Die Funktion `read_delim()` liest Dateien mit beliebigen Trennzeichen ein.
  - sie versucht, das Trennzeichen zu erraten, es sei denn, Sie geben es mit dem Argument `delim =` an (z.B. `read_delim(groesse_geburtstag.csv, delim = ",")`)

Lernziele

Heute haben wir gelernt, wie man…

- lokale Datendateien mit dem Paket `readr` importiert
- fehlende Werte behandeln

- Variablen in Faktoren umwandeln

Lassen Sie uns nun dieses neue Wissen anwenden.

Hausaufgaben

Wir wollen nun üben, das Paket `readr` zu verwenden und unsere Daten zu verarbeiten.

`readr` Funktionen

1. Welche Funktion würden Sie verwenden, um eine Datei zu lesen, in der die Felder mit `|` getrennt sind?
2. Welche Argumente haben `read_csv()` und `read_tsv()` gemeinsam?
3. Welche Funktion(en) könnten Sie verwenden, um einen Datensatz mit einem Semikolon (`;`) als Trennzeichen einzulesen?

Daten-wrangling

Laden Sie die Datei `groesse_geburtstag.csv` erneut. Benutzen Sie Pipes, um auch die Funktion `clean_names` zu benutzen und um die folgenden Änderungen im Objekt `df_groesse` vorzunehmen:

1. Umwandlung der Variablen `l1` in einen Faktor.
2. Umbenennen von
    - `grosse` in `groesse`
    - `monat_der_geburt` in `geburtsmonat`

Plots

1. Erstellen Sie unter Verwendung unseres Datensatzes `df_groesse` ein Streudiagramm, das die Beziehung zwischen unserem Geburtsdatum und unseren Geburtstagen veranschaulicht (es macht keinen Sinn, dies zu vergleichen, aber das ist nur eine Übung). Legen Sie die Farbe und die Form so fest, dass sie "L1" entsprechen. Fügen Sie einen Plot-Titel hinzu.
2. Suchen Sie Ihren Geburtstag auf dem Diagramm.
3. Erstellen Sie ein Balkendiagramm, das die Anzahl der Beobachtungen pro `l1` zeigt.

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] magick_2.7.4    here_1.0.1      janitor_2.2.0   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

loaded via a namespace (and not attached):
 [1] utf8_1.2.3       generics_0.1.3   stringi_1.7.12   hms_1.1.3
 [5] digest_0.6.33    magrittr_2.0.3   evaluate_0.21    grid_4.3.0
 [9] timechange_0.2.0 fastmap_1.1.1    rprojroot_2.0.3  jsonlite_1.8.7
[13] fansi_1.0.4      scales_1.2.1     cli_3.6.1        rlang_1.1.1
[17] crayon_1.5.2     bit64_4.0.5      munsell_0.5.0    withr_2.5.0
[21] yaml_2.3.7       tools_4.3.0      parallel_4.3.0   tzdb_0.4.0
[25] colorspace_2.1-0 pacman_0.5.1     vctrs_0.6.3      R6_2.5.1
[29] lifecycle_1.0.3  snakecase_0.11.0 bit_4.0.5        vroom_1.6.3
[33] pkgconfig_2.0.3  pillar_1.9.0     gtable_0.3.4     Rcpp_1.0.11
[37] glue_1.6.2       xfun_0.39        tidyselect_1.2.0 rstudioapi_0.14
[41] knitr_1.44       htmltools_0.5.5  rmarkdown_2.22   compiler_4.3.0
```

Literaturverzeichnis

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

Wickham, H., Çetinkaya-Rundel, M., & Grolemund, G. (2023). *R for Data Science* (2. Aufl.).