Data Wrangling 2

Data Tidying

Daniela Palleschi

2024-07-09

Inhaltsverzeichnis

Lernziele

Heute werden wir lernen...

- über breite versus lange Daten
- wie man breite Daten länger macht
- wie man lange Daten breiter macht

Ressourcen

Die vorgeschlagenen Ressourcen für dieses Thema sind

- Kurs-Website: Kapitel 9 (Data Wrangling 2)
- Kapitel 6 (Data Tidying) in Wickham et al. (2023)
- Kapitel 8 (Data Tidying) in Nordmann & DeBruine (2022)

1 Einrichtung

1.1 Pakete

```
pacman::p_load(tidyverse,
               here,
               janitor)
```

1.2 Daten

- Wir verwenden den Datensatz `languageR_english.csv` (im Ordner `daten`)

```
df_eng <- read_csv(here("daten", "languageR_english.csv")) |>
  clean_names() |>                                                    ①
  arrange(word) |>                                                    ②
  rename(                                                             ③
    rt_lexdec = r_tlexdec,                                            ④
    rt_naming = r_tnaming                                             ⑤
  ) |>
  select(age_subject, word, word_category, rt_lexdec, rt_naming)      ⑥
```

① Bereinigen (d.h. *tidy*) von Variablennamen (von `janitor`)

② Zeilen nach `wort` in ansteigender Reihenfolge anordnen (A-Z)
③ Variablen umbenennen…
④ `r_tlexdec` in `rt_lexdec` umbenennen
⑤ `r_tlexdec` in `rt_lexdec` umbenennen
⑥ nur die genannten Spalten behalten

2 'Tidy' Arbeitsablauf

- Abbildung 1 zeigt einen Überblick über den typischen Data-Science-Prozess
  - Wir importieren unsere Daten, bereinigen sie und durchlaufen dann einen Zyklus aus Umwandlung, Visualisierung und Modellierung, bevor wir schließlich unsere Ergebnisse kommunizieren.

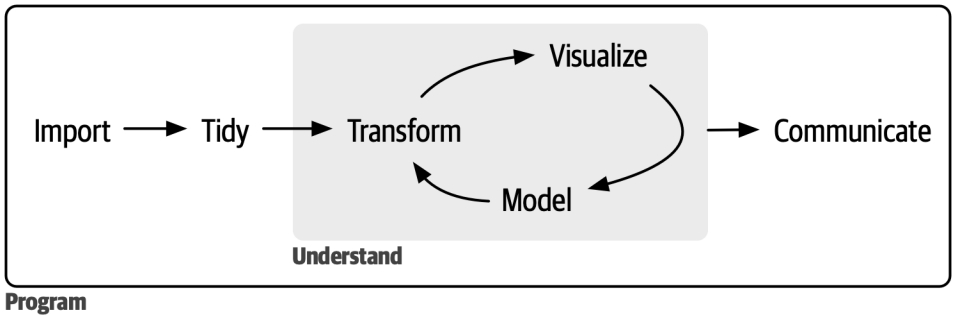

Abbildung 1: Image source: Wickham et al. (2023) (all rights reserved)

- Bisher haben wir gelernt, wie man
  - unsere Daten importieren (`readr::read_csv`)
  - Daten transformieren (Paket `dplyr`)
  - Daten zu visualisieren (Paket `ggplot`)
  - unsere Ergebnisse mit dynamischen Berichten zu kommunizieren (Quarto)
- aber wir haben bis jetzt nur aufgeräumte Daten gesehen
  - daher mussten wir den Schritt des "tidy" (Paket `tidyr`) noch nicht durchführen

3 'Tidy' Daten

- dieselben Daten können auf verschiedene Weise dargestellt werden
- Wir werden uns 3 Tabellen ansehen, die genau dieselben Daten in verschiedenen Formaten darstellen

- Die Tabellen zeigen die gleichen Werte von vier Variablen:
  - Land (`country`)
  - Jahr (`year`)
  - Bevölkerung (`population`)
  - Anzahl der Tuberkulosefälle (`cases`)
- Jeder Datensatz ordnet die Werte anders an
- überlegen Sie, welche Tabelle für Sie am einfachsten zu lesen ist

Tabelle 1: Version 1

country	year	cases	population
Afghanistan	1999	745	19987071
Afghanistan	2000	2666	20595360
Brazil	1999	37737	172006362
Brazil	2000	80488	174504898
China	1999	212258	1272915272
China	2000	213766	1280428583

Tabelle 2: Version 2

country	year	type	count
Afghanistan	1999	cases	745
Afghanistan	1999	population	19987071
Afghanistan	2000	cases	2666
Afghanistan	2000	population	20595360
Brazil	1999	cases	37737
Brazil	1999	population	172006362
Brazil	2000	cases	80488

Brazil	2000	population	174504898
China	1999	cases	212258
China	1999	population	1272915272
China	2000	cases	213766
China	2000	population	1280428583

Tabelle 3: Version 3

country	year	rate
Afghanistan	1999	745/19987071
Afghanistan	2000	2666/20595360
Brazil	1999	37737/172006362
Brazil	2000	80488/174504898
China	1999	212258/1272915272
China	2000	213766/1280428583

3.1 Regeln für 'tidy' Daten

- Wahrscheinlich ist Tabelle 1 für Sie am einfachsten zu lesen
  - sie folgt den drei Regeln für aufgeräumte Daten (visualisiert in Abbildung 2):

1. Jede Variable ist eine Spalte, jede Spalte ist eine Variable
2. Jede Beobachtung ist eine Zeile, jede Zeile ist eine Beobachtung
3. Jeder Wert ist eine Zelle, jede Zelle ist ein Einzelwert

3.2 Warum 'tidy' Daten?

> "**Glückliche Familien** sind alle gleich; jede **unglückliche Familie** ist auf ihre eigene Art unglücklich."
>
> — Leo Tolstoy

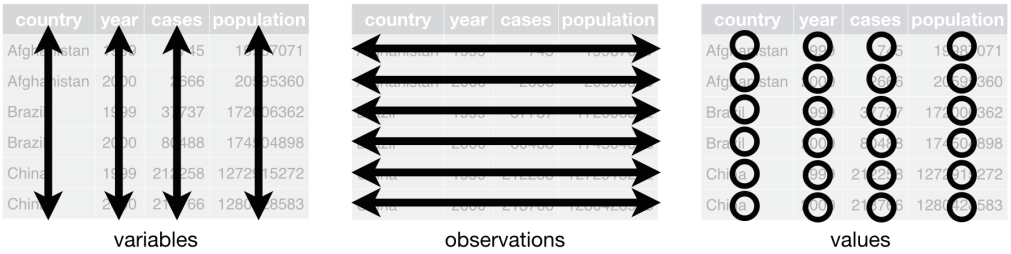

Abbildung 2: Image source: Wickham et al. (2023) (all rights reserved)

> "**'Tidy' Datensätze** sind alle gleich, aber jeder **'untidy' Datensatz** ist auf seine eigene Weise unordentlich."
>
> — Hadley Wickham

Die Arbeit mit aufgeräumten Daten hat zwei wesentliche Vorteile:

1. Die Arbeit mit einer konsistenten Datenstruktur ermöglicht es uns, Konventionen zu übernehmen.
    - Aufgeräumte Daten sind die allgemein vereinbarte Datenstruktur
    - Konventionen/Werkzeuge basieren auf der Annahme dieser Struktur
2. Die vektorisierte Natur von R kann glänzen
    - die meisten eingebauten R-Funktionen arbeiten mit *Vektorwerten* (und Spalten sind im Wesentlichen Vektoren)
    - Alle Pakete im `tidyverse` sind darauf ausgelegt, mit aufgeräumten Daten zu arbeiten (z.B. `ggplot2` und `dplyr`)

4 Daten bereinigen (tidying)

- Umwandlung breiter Daten in lange Daten und langer Daten in breite Daten (neben anderen Schritten)
    - Ergebnis: aufgeräumte Daten (normalerweise)

4.1 'Tidying' Daten mit dem `tidyverse`

- Das Paket `tidyr` (aus `tidyverse`) hat zwei nützliche Funktionen zum Transponieren unserer Daten:

- `pivot_longer()`: macht breite Daten länger
- `pivot_wider()`: lange Daten breiter machen

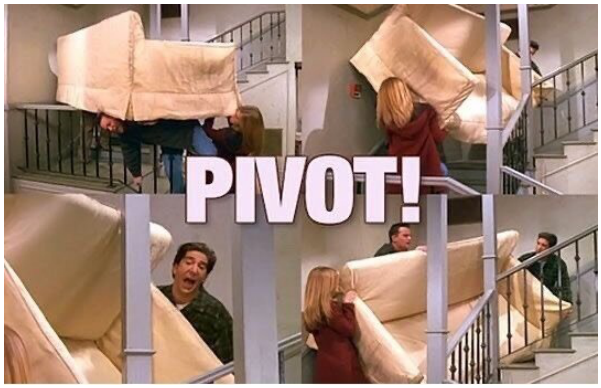

Abbildung 3: die berühmteste Verwendung des Wortes Pivot (zumindest für Millenials)

4.2 Breite versus lange Daten

- Wir müssen oft zwischen breiten und langen Datenformaten konvertieren, um verschiedene Arten von Zusammenfassungen oder Visualisierungen zu erstellen
- breite Daten: alle Beobachtungen zu einer Sache befinden sich in einer einzigen Zeile
  - ist *normalerweise* nicht aufgeräumt
- lange Daten: jede Beobachtung befindet sich in einer separaten Zeile
  - ist *normalerweise* aufgeräumt
- Beginnen wir mit dem typischsten Fall: Umwandlung breiter Daten in lange Daten

5 `pivot_longer()`

- im Datensatz `languageR_english.csv` (`df_eng`)
  - haben wir 4568 Beobachtungen (Zeilen)
  - Wir haben 5 Variablen (Spalten)
  - die Spalte `age_subject` gibt an, ob eine Beobachtung von einem Teilnehmer der Altersgruppe `old` oder `young` stammt
  - die Spalten `word` und `word_category` beschreiben Eigenschaften des Stimulus für eine bestimmte Beobachtung (d. h. das Wort)
  - die Spalte `rt_lexdec` enthält die Reaktionszeit für eine lexikalische Entscheidungsaufgabe

– die Spalte `rt_naming` enthält die Antwortzeit für eine Wortbenennungsaufgabe

Tabelle 4: df_eng

age_subject	word	word_category	rt_lexdec	rt_naming
young	ace	N	623.61	456.3
old	ace	N	775.67	607.8
young	act	V	617.10	445.8
old	act	V	715.52	639.7
young	add	V	575.70	467.8
old	add	V	742.19	605.4

- Sind diese Daten in Tabelle 4 aufgeräumt?
- Sind diese Daten zu breit oder zu lang?
- Wie können wir diese Daten länger machen?

5.1 Our goal

- wir wollen Abbildung 4 produzieren

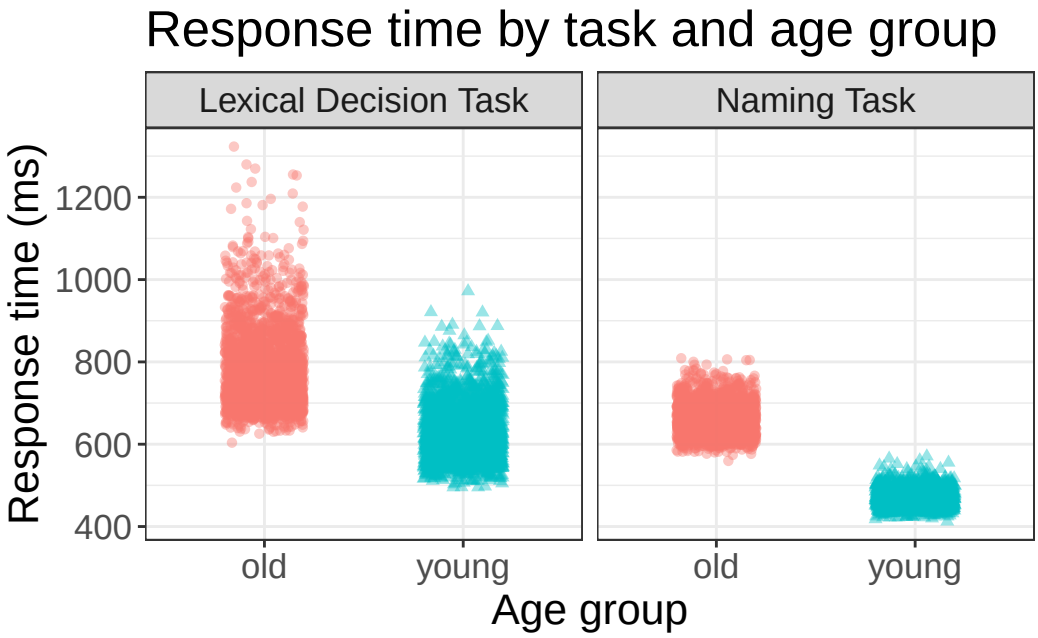

Abbildung 4: Our plot to be reproduced

- die beiden kontinuierlichen Variablen `rt_lexdec` und `rt_naming` erscheinen in Facetten

  - `facet_wrap()` nimmt eine *kategorische* Variable als Argument und erzeugt eine Facette für jede Kategorie

- wir brauchen also eine kategorische Variable, die die Ebenen `lexdec` und `naming` enthält
  - und eine *kontinuierliche* Variable, die die entsprechende Antwortzeit enthält

5.2 `pivot_longer()`

- Die Funktion `pivot_longer()` (von `tidyr`) konvertiert eine breite Datentabelle in ein längeres Format
  - wandelt die Namen der angegebenen Spalten in die Werte einer neuen kategorischen Spalte um
  - und kombiniert die Werte dieser Spalten in einer neuen Spalte

```
df_eng_long <-                                  ①
  df_eng %>%                                    ②
  pivot_longer(                                 ③
    cols = starts_with("rt_"),                  ④
    names_to = "response",                      ⑤
    values_to = "time"                          ⑥
  )
```

① Erstellen Sie ein neues Objekt namens `df_eng_long`, das…
② `df_eng`, und dann
③ mache es länger
④ indem du Spalten (`col =`) nimmst, die mit `rt_` beginnen
⑤ und eine Variable namens `response` erstellen, die die Namen aus `cols` enthält (`names_to =`)
⑥ und eine Variable namens `time` erstellen, die die Werte aus `cols` enthält (`values_to =`)

```
df_eng_long |> head()
```

```
# A tibble: 6 x 5
  age_subject word  word_category response   time
  <chr>       <chr> <chr>         <chr>     <dbl>
1 young       ace   N             rt_lexdec  624.
2 young       ace   N             rt_naming  456.
3 old         ace   N             rt_lexdec  776.
4 old         ace   N             rt_naming  608.
5 young       act   V             rt_lexdec  617.
6 young       act   V             rt_naming  446.
```

- Vergleichen wir die Beobachtungen für die Wörter `ace` und `act` in
  - `df_eng` (Tabelle 5)
  - `df_eng_longer` (Tabelle 6)

Tabelle 5: `df_eng`

age_subject	word	rt_lexdec	rt_naming
young	ace	623.61	456.3
old	ace	775.67	607.8
young	act	617.10	445.8
old	act	715.52	639.7

Tabelle 6: `df_eng |> pivot_longer(...)`

age_subject	word	response	time
young	ace	rt_lexdec	623.61
young	ace	rt_naming	456.30
old	ace	rt_lexdec	775.67
old	ace	rt_naming	607.80
young	act	rt_lexdec	617.10
young	act	rt_naming	445.80
old	act	rt_lexdec	715.52
old	act	rt_naming	639.70

- die beiden Tabellen enthalten genau die gleichen Informationen
  - 8 Werte für die Antwortzeit:
    - 4 für `rt_lexdec`
    - 4 für `rt_naming`
- Dies ist eine wichtige Erkenntnis: Wir haben keine Daten oder Beobachtungswerte geändert, sondern lediglich die Organisation der Datenpunkte neu strukturiert.

5.2.1 Plotten unserer 'tidy' Daten

- Versuchen wir nun, unser Diagramm zu erstellen:
  - `age_subject` auf der x-Achse
  - `time` auf der y-Achse
  - Kategorien `response` in Facetten

Abbildung 5: Response times per age group for the lexical decision task vs. naming task

Aufgabe 5.1: Tidy data

Beispiel 5.1.
Abbildung 5 neu erstellen.

6 `pivot_wider()`

- Es kommt häufiger vor, dass man seine Daten verlängern will (man nimmt Spalten und macht aus deren Werten neue Zeilen)
  - aber manchmal möchte man seine Daten auch verbreitern (man nimmt Zeilen und verwandelt ihre Werte in neue Spalten)

- Die `tidyr`-Funktion `pivot_wider()` macht Datensätze *breiter*, indem sie Spalten vergrößert und Zeilen reduziert.
  - Dies ist hilfreich, wenn eine Beobachtung über mehrere Zeilen verteilt ist.
- Lassen Sie uns versuchen, `df_eng` breiter zu machen
  - Wir könnten zum Beispiel eine einzige Zeile pro *Wort* haben
    * mit einer einzigen Variablen für die Antwort des `young` Probanden und die Antwort des `old` Probanden

6.1 `pivot_wider()`

- pivot wider nimmt ähnliche Argumente wie `pivot_longer()`, mit einigen leichten Unterschieden:
  - `id_cols` (optional): identifizierende Spalten (welche Spalten identifizieren jede Beobachtung eindeutig?)
  - `names_from`: wie soll die neue Spalte heißen, die die vorherigen Spaltennamen enthält (muss eine kategorische Variable sein)?
  - `names_prefix` (optional): Präfix für die neuen Spaltennamen (optional)
  - `values_von`: neue Spaltenwerte

6.2

- lassen Sie uns zwei neue Variablen erstellen, die ihre Namen von `age_subject` und ihre Werte von `rt_lexdec` übernehmen

```
df_eng_wide <-
  df_eng %>%
  select(-rt_naming) |>
  pivot_wider(
    names_from = age_subject,                    ①
    values_from = rt_lexdec,                     ②
    names_prefix = "lexdec_"                     ③
  )
```

① neue Spaltennamen unter Verwendung der Werte in `age_subject` erstellen
② Erstelle neue Beobachtungswerte aus `rt_lexdec`
③ Hinzufügen von `lexdec_` am Anfang der neuen Spaltennamen

6.3

- Vergleichen wir die Beobachtungen für die Wörter `ace` und `act` in
  - `df_eng` (Tabelle 5)
  - `df_eng_longer` (Tabelle 6)

Tabelle 7: `df_eng`

age_subject	word	word_category	rt_lexdec
young	ace	N	623.61
old	ace	N	775.67
young	act	V	617.10
old	act	V	715.52

Tabelle 8: `df_eng_wide`

word	word_category	lexdec_young	lexdec_old
ace	N	623.61	775.67
act	V	617.10	715.52

- Auch hier haben wir keine Daten oder Beobachtungswerte geändert, sondern lediglich die Anordnung der Datenpunkte neu strukturiert.

6.4 Eindeutige Werte

- Wir haben `rt_naming` entfernt, weil es auch einen eindeutigen Wert pro Wort pro Altersgruppe hat
- wir ändern nur die Breite und führen `NA`-Werte für `lexdec_young` für alte Themen und `NA`-Werte für `lexdec_old` für junge Themen ein
- Hätten wir sie nicht entfernt, sähen unsere ersten 6 Zeilen wie Tabelle 9 aus
  - Vergleichen Sie dies mit der Ausgabe in Tabelle 8, sehen Sie den Unterschied?

Tabelle 9: Wider data with missing values

word	word_category	rt_naming	lexdec_young	lexdec_old
ace	N	456.3	623.61	NA
ace	N	607.8	NA	775.67
act	V	445.8	617.10	NA
act	V	639.7	NA	715.52

Lernziele

Heute haben wir gelernt...

- über breite und lange Daten
- wie man breite Daten länger macht
- wie man lange Daten breiter macht

7 Hausaufgaben

Anhang 2 auf der Website des Kurses.

Session Info

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

```
sessionInfo()
```

```
R version 4.4.0 (2024-04-24)
Platform: aarch64-apple-darwin20
Running under: macOS Ventura 13.2.1

Matrix products: default
BLAS:   /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libRblas.0.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libRlapack.dylib;
```

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

time zone: Europe/Berlin
tzcode source: internal

attached base packages:
[1] stats     graphics  grDevices datasets  utils     methods   base

other attached packages:
 [1] janitor_2.2.0   here_1.0.1      lubridate_1.9.3 forcats_1.0.0
 [5] stringr_1.5.1   dplyr_1.1.4     purrr_1.0.2     readr_2.1.5
 [9] tidyr_1.3.1     tibble_3.2.1    ggplot2_3.5.1   tidyverse_2.0.0

loaded via a namespace (and not attached):
 [1] utf8_1.2.4        generics_0.1.3    renv_1.0.7        xml2_1.3.6
 [5] stringi_1.8.3     hms_1.1.3         digest_0.6.35     magrittr_2.0.3
 [9] evaluate_0.23     grid_4.4.0        timechange_0.3.0  fastmap_1.1.1
[13] rprojroot_2.0.4   jsonlite_1.8.8    tinytex_0.50      fansi_1.0.6
[17] viridisLite_0.4.2 scales_1.3.0      cli_3.6.2         rlang_1.1.3
[21] crayon_1.5.2      bit64_4.0.5       munsell_0.5.1     withr_3.0.0
[25] yaml_2.3.8        tools_4.4.0       parallel_4.4.0    tzdb_0.4.0
[29] colorspace_2.1-0  pacman_0.5.1      kableExtra_1.4.0  png_0.1-8
[33] vctrs_0.6.5       R6_2.5.1          magick_2.8.3      lifecycle_1.0.4
[37] snakecase_0.11.1  bit_4.0.5         vroom_1.6.5       pkgconfig_2.0.3
[41] pillar_1.9.0      gtable_0.3.5      Rcpp_1.0.12       glue_1.7.0
[45] systemfonts_1.0.6 xfun_0.43         tidyselect_1.2.1  rstudioapi_0.16.0
[49] knitr_1.46        farver_2.1.1      htmltools_0.5.8.1 labeling_0.4.3
[53] svglite_2.1.3     rmarkdown_2.26    compiler_4.4.0
```

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.).