Data Tidying

Data Tidying

Daniela Palleschi

2023-06-06

Inhaltsverzeichnis

Wiederholung

Last week we learned how to…

- Code-Chunk-Optionen verwenden
- formatierte Tabellen drucken
- Abbildungsunterschriften hinzufügen
- die Größe von Abbildungen kontrolliern
- Querverweise erstellem

Last week's exercises

- I had some typos in my instructions :(
  - `include: false` should have been `message: false`
  - `fig-out: 6` should have been `fig-width: 6`
- there were some common issues
  - e.g., setting global options
  - `label` formatting
  - missing cross-references

Heutige Ziele

Today we will…

- learn about wide versus long data
- make wide data longer
- make long data wider

Lust auf mehr?

- Ch. 6 (Data tidying) in Wickham et al. (o. J.)
- Ch. 8 (Data Tidying) in Nordmann & DeBruine (2022)

1 Einrichtung

- packages: `tidyverse`, `here`

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

- data (in `daten` folder):
  - `table1.csv`
  - `df_billboard.csv` (neu)
  - `cms_patient_experience.csv` (neu)

```
df_tb <- read_csv(here("daten", "table1.csv"))
df_billboard <- read_csv(here("daten", "df_billboard.csv"))
df_cms <- read_csv(here("daten", "cms_patient_experience.csv"))
```

2 Tidy workflow

- we've learned how to import (`readr::read_csv`), transform (`dplyr` package), and visualise (`ggplot` package) data
- so far we've always worked with tidy data, so we haven't needed to perform the 'tidy' step

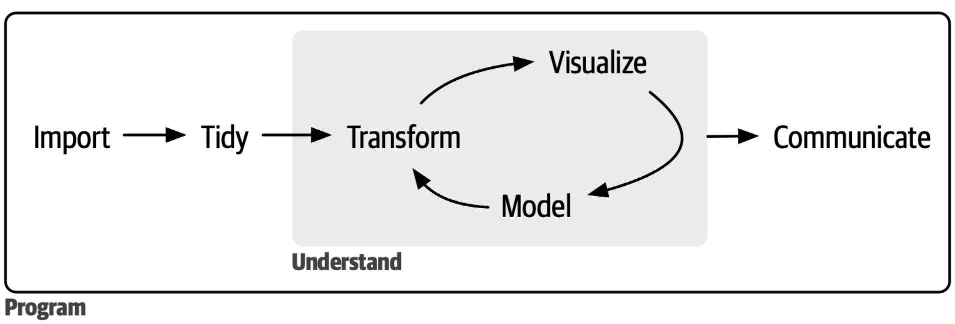

Abbildung 1: Image source: Wickham et al. (o. J.) (all rights reserved)

3 Tidy data

- the same data can be representing multiple ways

Tabelle 1: Tabelle 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

- the datasets below all show the same values of four variables: `country`, `year`, `popuplation`, and number of tuberculosis `cases`
  - each dataset organises the values differently
- which is easiest to read?

Three rules for tidy data:

1. Each variable is a column, each column is a variable
2. Each observation is a row, each row is an observation
3. Each value is a cell, each cell is a single value

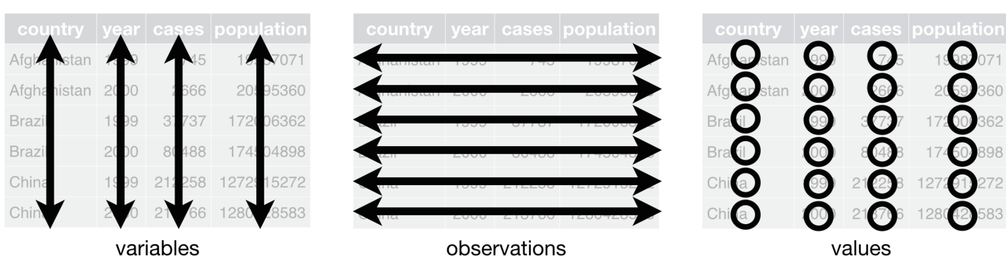

Abbildung 2: Image source: Wickham et al. (o. J.) (all rights reserved)

- what counts as an observations or a variable is often dependent on the task at hand

Tabelle 2: Tabelle 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: Tabelle 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 Why tidy data?

> "**Happy families** are all alike; every **unhappy family** is unhappy in its own way."
> — Leo Tolstoy

> "**Tidy datasets** are all alike, but every **untidy dataset** is untidy in its own way."
> — Hadley Wickham

- data tidying requires some work up front, but is helpful in the long run
- once you have tidy data, you'll spend less time trying to get your data in the right shape to do what you want

---

There are two main advantages to working with tidy data:

1. working with a consistent data structure allows us to adopt conventions
    - since tidy data is the generally agreed upon data structure, conventions are built on the assumption of this structure
    - so tools have an underlying uniformitivity
2. R's vectorised nature can shine
    - most built-in R funtions work with *vector values*
    - all packages in the `tidyverse` are designed to work with tidy data

Vectors

Vectors are the most basic data object type in R. A vector contains data of the same type, and is essentially a list. You can create a vector using the `c()` function, for example.

```
vector1 <- c(2, 3, 4, 6, 7)
vector2 <- c(2, 3, 4, 6, "c")
```

`vector1` will contain numeric values, because all elements are numbers. `vector2` will contain all character values (i.e., text), because there is a singlular unambiguous character element (`"c"`). So, R reads all elements as character type. We can create a dataframe from vectors of the same length using the `tibble()` function, for example.

```
tibble(vector1,vector2)
```

```
# A tibble: 5 x 2
  vector1 vector2
    <dbl> <chr>
```

```
1         2 2
2         3 3
3         4 4
4         6 6
5         7 c
```

- most data "in the wild" is untidy
  - data is often first organised for some goal other than analysis
    - usually this goal is data entry: we want to make it easy to document our observations first
  - most people aren't familiar with the principles of tidy data, only after spending *a lot* of time with data does it become obvious why tidy data is necessary
- this means most *real* analyses will require at least some tidying

Aufgabe 3.1: Tidy data

Beispiel 3.1.

1. Go back to Tables 1-3. For each table, describe what each observation and each column represents.
2. Sketch out the process you'd use to calculate the rate for table1. You will need just one verb that:
   - creates a new variable (call it `rate`) that contains:
     - the number of TB `cases` per country per year, divided by
     - the matching `population` per country per year,
     - multiply by `10000`
   - hint: Which `dplyr` verb creates new variables? (Look back at week 5)
3. Look at tables 2 and 3. Would it have been as easy to calculate `rate` with these data structures?

4 Data tidying

- re-shaping
  - e.g., from wide to long data
- outcome:

  - each column = a variable
  - each row = an observation

4.1 Data tidying with the `tidyverse`

- with `tidyr` package
  - `pivot_longer()`: make wide data longer
  - `pivot_wider()`: make long data wider
- we often need to convert between these formats to do different types of summaries or visualisation

---

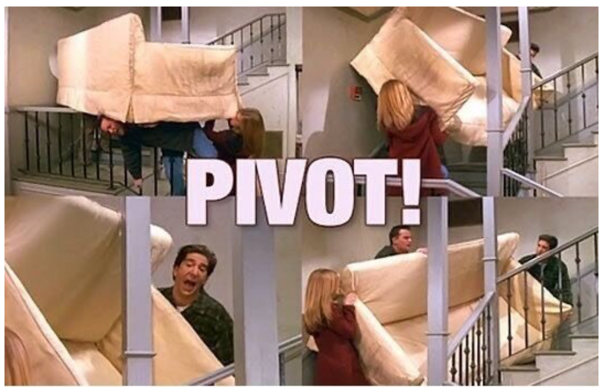

Abbildung 3: die berühmteste Verwendung des Wortes Pivot (zumindest für Millenials)

5 Wide versus long data

- Wide data: all of the observations about one thing are in the same row
  - wide data is *usually* not tidy
- long data: each observation is on a separate row
  - long data is *usually* tidy

Tabelle 4: df_billboard rank of songs in the year 2000

artist	track	date_entered	wk1	wk2	wk3	wk4	wk5	wk6	wk7	wk8
2 Pac	Baby Don't Cry (Keep...	2000-02-26	87	82	72	77	87	94	99	NA
2Ge+her	The Hardest Part Of ...	2000-09-02	91	87	92	NA	NA	NA	NA	NA
3 Doors Down	Kryptonite	2000-04-08	81	70	68	67	66	57	54	53
3 Doors Down	Loser	2000-10-21	76	76	72	69	67	65	55	59
504 Boyz	Wobble Wobble	2000-04-15	57	34	25	17	17	31	36	49
98^0	Give Me Just One Nig...	2000-08-19	51	39	34	26	26	19	2	2

6 Lengthening data: `df_billboard`

- in the `billboard.csv` dataset
  - each row is a song
  - the first three columns are variables that describe the song (`artist`, `track`, `date_entered`)
  - we have 76 columns (`wk1-wk76`) that describe the rank of the song that week
    - the columns names are one variable (the `week`), and the cell values are another variable (the `rank`)

```
df_billboard %>%
  head() %>%
  knitr::kable() %>%
  kableExtra::kable_styling()
```

- is this data in Tabelle 4 tidy?
- is this data too wide or too long?
- how might we tidy this data?

6.1 `pivot_longer()`

- `pivot_longer()`
  - converts a wide data table to a longer format by converting the headers from specified columns into the values of new columns
  - and combining the values of those columns into a new condensed column

```
df_billboard_tidy <- df_billboard %>%
  pivot_longer(
    cols = starts_with("wk"),
```

Tabelle 5: A pivoted version of `df_billboard` (first 10 rows)

artist	track	date_entered	week	rank
2 Pac	Baby Don't Cry (Keep...	2000-02-26	wk1	87
2 Pac	Baby Don't Cry (Keep...	2000-02-26	wk2	82
2 Pac	Baby Don't Cry (Keep...	2000-02-26	wk3	72
2 Pac	Baby Don't Cry (Keep...	2000-02-26	wk4	77
2 Pac	Baby Don't Cry (Keep...	2000-02-26	wk5	87
2 Pac	Baby Don't Cry (Keep...	2000-02-26	wk6	94
2 Pac	Baby Don't Cry (Keep...	2000-02-26	wk7	99
2 Pac	Baby Don't Cry (Keep...	2000-02-26	wk8	NA
2 Pac	Baby Don't Cry (Keep...	2000-02-26	wk9	NA
2 Pac	Baby Don't Cry (Keep...	2000-02-26	wk10	NA

```
    names_to = "week",
    values_to = "rank"
  )
```

```
df_billboard_tidy %>%
  head(n = 10) %>%
  knitr::kable() %>%
  kableExtra::kable_styling(font_size = 20)
```

- `col =` specifies which columns need to be pivoted (i.e., which are *not* variables)
  - takes the same syntax as `select()`, so we could use e.g., `starts_with("")`
- `names_to =` names the variable stored in the current column names, here it is `week`
- `values_to =` names the variable stored int he cell values, which we name `rank`
- N.B., we had to wrap `week` and `rank` with quotation marks because they aren't variable names *yet*

6.1.1 Removing missing values (`NA`s)

- we have a few `NA` (missing) values; this happened when a song was not in the top 100

Tabelle 6: Tidy `df_billboard` data with dropped `NA` values

artist	track	date_entered	week	rank
2 Pac	Baby Don't Cry (Keep...	2000-02-26	wk1	87
2 Pac	Baby Don't Cry (Keep...	2000-02-26	wk2	82
2 Pac	Baby Don't Cry (Keep...	2000-02-26	wk3	72
2 Pac	Baby Don't Cry (Keep...	2000-02-26	wk4	77
2 Pac	Baby Don't Cry (Keep...	2000-02-26	wk5	87
2 Pac	Baby Don't Cry (Keep...	2000-02-26	wk6	94
2 Pac	Baby Don't Cry (Keep...	2000-02-26	wk7	99
2Ge+her	The Hardest Part Of ...	2000-09-02	wk1	91
2Ge+her	The Hardest Part Of ...	2000-09-02	wk2	87
2Ge+her	The Hardest Part Of ...	2000-09-02	wk3	92

  - the `values_drop_na = TRUE/FALSE` argument drops pivoted rows don't have a value for the new variable

```
df_billboard_tidy <- df_billboard %>%
  pivot_longer(
    cols = starts_with("wk"),
    names_to = "week",
    values_to = "rank",
    values_drop_na = TRUE
  )
```

- how many rows fewer do we have now?

6.1.2 Parsing numbers

- the data is now tidy, but `week` still contains `wk` in the values
- the `readr` package has a handy function: `parse_number()` extracts the first number from a string, ignoring all other text
  - how could we use `parse_number()` to *alter the variable* `week`?

Tabelle 7: Tidy `df_billboard` data with dropped `NA` values

artist	track	date_entered	week	rank
2 Pac	Baby Don't Cry (Keep...	2000-02-26	1	87
2 Pac	Baby Don't Cry (Keep...	2000-02-26	2	82
2 Pac	Baby Don't Cry (Keep...	2000-02-26	3	72
2 Pac	Baby Don't Cry (Keep...	2000-02-26	4	77
2 Pac	Baby Don't Cry (Keep...	2000-02-26	5	87
2 Pac	Baby Don't Cry (Keep...	2000-02-26	6	94

```
df_billboard_tidy <- df_billboard_tidy %>%
  mutate(week = parse_number(week))
```

6.1.3 Working with dates

- the `tidyverse` also has a package that makes working with dates easier: `lubridate`
  - the variable `date_entered` has the format year-month-day (`ymd`)
  - we can use `lubridate` verbs to extract the `year`, `month`, and `day`

```
df_billboard_tidy <- df_billboard_tidy %>%
  mutate(
    month = month(date_entered, label = F),
    day = day(ymd(date_entered)),
    year = year(ymd(date_entered))
  )
```

6.1.4 Plotting our tidy data

- now we have the `week` data in one variable and the `rank` data in another variable
- let's try to plot billboard ranks by week

- what is the most number of weeks a song was in the top 100?
- why is 100 at the bottom and 0 at the top?
- why is the bottom right quarter empty? What does this mean?

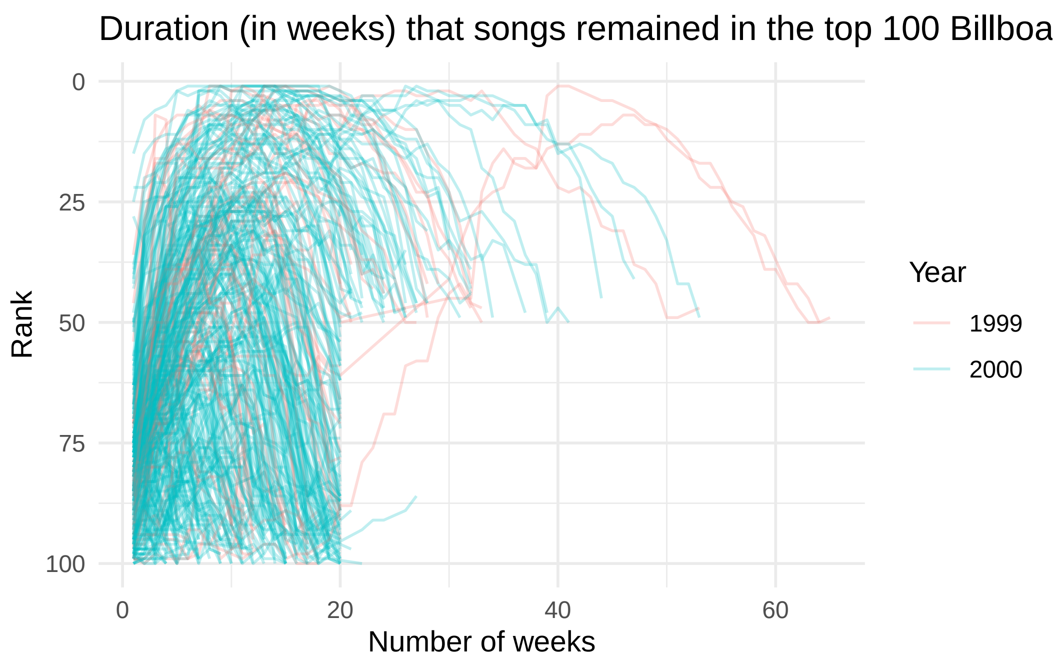

Abbildung 4: Billboard ranks by number of weeks for songs that were in the top 100 in the year 2000

Aufgabe 3.1: Tidy data

Beispiel 6.1.

1. Recreate Abbildung 4.

7 Widening data: `df_cms`

- `pivot_wider()` make datasets *wider* by increasing columns and reducing rows
  - this helps when one observation is spread across multiple rows
  - this type of data isn't very common in the wild, but is pretty common in governmental data

---

- the `cms_patient_experience.csv` dataset contains data about patient experiences fom the Centers of Medicare and Medicaid services

org_pac_id	org_nm	measure_cd
0446157747	USC CARE MEDICAL GROUP INC	CAHPS_GI
0446157747	USC CARE MEDICAL GROUP INC	CAHPS_GI
0446157747	USC CARE MEDICAL GROUP INC	CAHPS_GI
0446157747	USC CARE MEDICAL GROUP INC	CAHPS_GI
0446157747	USC CARE MEDICAL GROUP INC	CAHPS_GI
0446157747	USC CARE MEDICAL GROUP INC	CAHPS_GI

- the core unit being studied is an organisation (stored in `org_pac_id` and `org_nm`), but each organisation (i.e., observation) takes up 6 rows
    - one row for each measure (i.e., variable)
    - so we want these variables to be represented in columns

7.1 `pivot_wider()`

- takes long data and makes it wider
- takes a few arguments:
    - `id_cols`: identifying columns
    - `names_from`: what should we call the new column containing the previous column names?
    - `names_prefix`: prefix for the new column names
    - `values_from`: new column values

- how can we, for each `org`,
    - take the six values from `measure_cd` and create six new variable `names from` them,
    - and take the `values from` `prf_rate`?
- the result should look like Tabelle 8

```
cms_patient_experience %>%
  pivot_wider(
    id_cols = starts_with("org"),
    names_from = measure_cd,
```

Tabelle 8: Wider `df_cms` data (measure variable to columns with rating as values)

org_pac_id	org_nm
0446157747	USC CARE MEDICAL GROUP INC
0446162697	ASSOCIATION OF UNIVERSITY PHYSICIANS
0547164295	BEAVER MEDICAL GROUP PC
0749333730	CAPE PHYSICIANS ASSOCIATES PA
0840104360	ALLIANCE PHYSICIANS INC
0840109864	REX HOSPITAL INC

Tabelle 9: Original `cms_patient_experience.csv` format

org_pac_id	org_nm	measure_cd
0446157747	USC CARE MEDICAL GROUP INC	CAHPS_GI
0446157747	USC CARE MEDICAL GROUP INC	CAHPS_GI
0446157747	USC CARE MEDICAL GROUP INC	CAHPS_GI
0446157747	USC CARE MEDICAL GROUP INC	CAHPS_GI
0446157747	USC CARE MEDICAL GROUP INC	CAHPS_GI
0446157747	USC CARE MEDICAL GROUP INC	CAHPS_GI

```
    values_from = prf_rate
  ) %>%
  head() %>%
  knitr::kable() %>%
  kableExtra::kable_styling(font_size = 20)
```

- Tabelle 9 shows the first 6 rows of the original dataset
- Tabelle 10 shows the first 6 rows of the widened dataset
- how is the data from Tabelle 9 represented in Tabelle 10?

Tabelle 10: Widened `cms_patient_experience.csv` format

org_pac_id	org_nm
0446157747	USC CARE MEDICAL GROUP INC
0446162697	ASSOCIATION OF UNIVERSITY PHYSICIANS
0547164295	BEAVER MEDICAL GROUP PC
0749333730	CAPE PHYSICIANS ASSOCIATES PA
0840104360	ALLIANCE PHYSICIANS INC
0840109864	REX HOSPITAL INC

8 Aufgabe

1. Browse the tracks. Choose a song you like (or a song at random), and then
    - filter the data to include only this song, and then
    - create a line plot of the song's time on the Top 100 Billboard
    - refer to the plot and describe the song's trend (you'd want to look at the song's data for this)

An example: *Abbildung 5 shows the trend of 'Say My Name' by Destiny's child, which entered the charts on Dec. 25th, 1999 (`date_entered`) at number 83 (`rank` for `wk1`), and stayed on the top 100 for 32 weeks (`max(week)`).*

2. Load the dataset `biondo_etal_2021_tidy.csv` (subset of data from Biondo et al. (2022)) and save it as `df_biondo`
    - print a formatted table (using `knitr::kable()`) with a `label` and caption of the `head()` of the data
3. Lengthen the data so that the columns `rt` and `tt` are in one column:
    - the variable `names` should go `to` a new variable called `measure`
    - the variable `values` should go `to` a new variable `ms` (for milliseconds)
    - save the lengthened data as `df_biondo_long`
    - print a formatted table (using `knitr::kable()`) with a `label` and caption
4. Widen `df_biondo_long` so that the columns `rt` and `tt` are back in their own columns
    - the `id_cols` should be `subj` and `item`
    - the variable `names` should come `from` `measure`

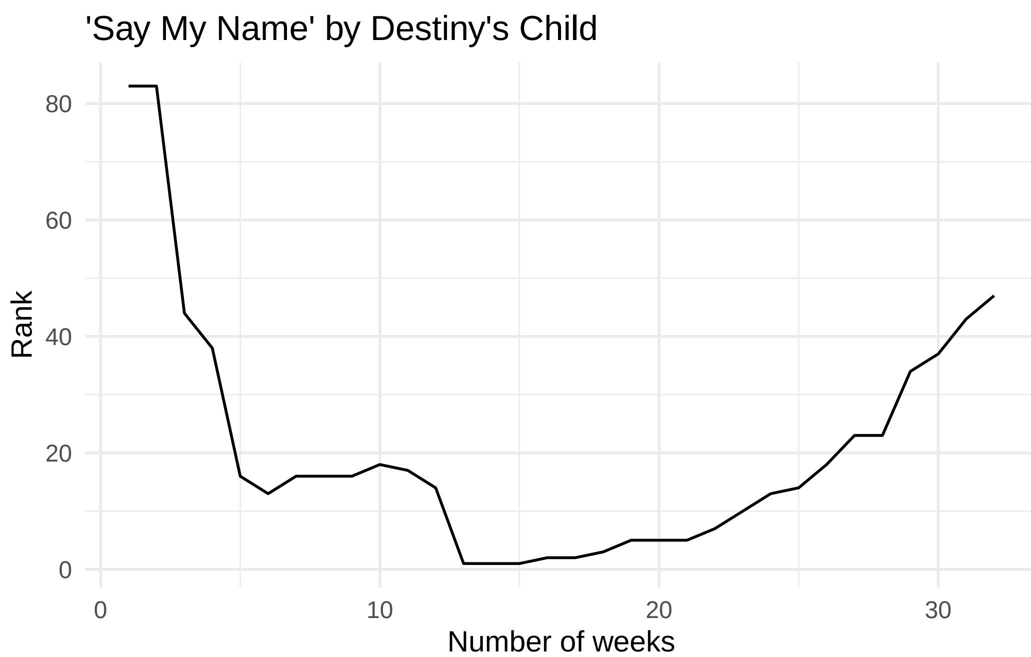

Abbildung 5: Example linegraph for 'Say My Name' by Destiny's Child

- the variable `values` should come `from ms` (for milliseconds)
- save the lengthened data as `df_biondo_wide`
- print a formatted table (using `knitr::kable()`) with a `label` and caption
- tip: `df_biondo_wide` should be the exact same as `df_biondo`

Heutige Ziele

Heute haben wir…

- learn about wide versus long data
- make wide data longer
- make long data wider
- review the `dplyr` verbs from week 3

Session Info

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

```r
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] here_1.0.1      lubridate_1.9.2 forcats_1.0.0   stringr_1.5.0
 [5] dplyr_1.1.2     purrr_1.0.1     readr_2.1.4     tidyr_1.3.0
 [9] tibble_3.2.1    ggplot2_3.4.2   tidyverse_2.0.0

loaded via a namespace (and not attached):
 [1] utf8_1.2.3            generics_0.1.3        xml2_1.3.4
 [4] stringi_1.7.12        hms_1.1.3             digest_0.6.31
 [7] magrittr_2.0.3        evaluate_0.21         grid_4.3.0
[10] timechange_0.2.0      fastmap_1.1.1         rprojroot_2.0.3
[13] jsonlite_1.8.4        httr_1.4.6            rvest_1.0.3
[16] fansi_1.0.4           viridisLite_0.4.2     scales_1.2.1
[19] cli_3.6.1             rlang_1.1.1           crayon_1.5.2
[22] bit64_4.0.5           munsell_0.5.0         withr_2.5.0
[25] yaml_2.3.7            tools_4.3.0           parallel_4.3.0
[28] tzdb_0.4.0            colorspace_2.1-0      webshot_0.5.4
[31] pacman_0.5.1          kableExtra_1.3.4.9000 png_0.1-8
[34] vctrs_0.6.2           R6_2.5.1              lifecycle_1.0.3
[37] magick_2.7.4          bit_4.0.5             vroom_1.6.3
[40] pkgconfig_2.0.3       pillar_1.9.0          gtable_0.3.3
[43] glue_1.6.2            Rcpp_1.0.10           systemfonts_1.0.4
[46] xfun_0.39             tidyselect_1.2.0      rstudioapi_0.14
[49] knitr_1.42            farver_2.1.1          htmltools_0.5.5
```

```
[52] labeling_0.4.2       svglite_2.1.1        rmarkdown_2.21
[55] compiler_4.3.0
```

Literaturverzeichnis

Biondo, N., Soilemezidi, M., & Mancini, S. (2022). Yesterday Is History, Tomorrow Is a Mystery: An Eye-Tracking Investigation of the Processing of Past and Future Time Reference during Sentence Reading. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *48*(7), 1001–1018. https://doi.org/10.1037/xlm0001053

Nordmann, E., & DeBruine, L. (2022). *Applied Data Skills* (Version 2.0). Zenodo. https://doi.org/10.5281/zenodo.6365078

Wickham, H., Çetinkaya-Rundel, M., & Grolemund, G. (o. J.). *R for Data Science* (2. Aufl.). https://r4ds.hadley.nz/