Data Transformation

Working with rows and columns

Daniela Palleschi

2024-07-09

Inhaltsverzeichnis

Wiederholung

Letze Woche haben wir…

- gelernt, wie man einen neuen Datensatz in Augenschein nimmt
- gelernt, wie man verschiedene Datentypen importiert
- gelernt, wie man Daten von Hand eingibt
- einen neuen Datensatz visualisiert

Heutige Ziele

Today we will…

- learn how to wrangle data using the `dplyr` package from the `tidyverse`
- learn to use uses the `pipe` (`|>`) to feed the result of one function into another function
- learn about functions that operate on rows
- learn about functions that operate on columns
- learn how to combine `dplyr` functions with plots from `ggplot2`

Lust auf mehr?

- Ch. 4 in Wickham et al. (o. J.)
- Ch. 9 in Nordmann & DeBruine (2022)

1 Pre-requisites

1. Fresh Quarto document
    - create a new Quarto document for today's class
        - File > New Document > Quarto Document, named something like `04-wrangling`
    - set up the YAML: title, your name, add a `toc`

```
title: "Data wrangling"
subtitle: "Transforming data"
author: "Your name here"
lang: de
date: "`r Sys.Date()`"
format:
  html:
    toc: true
```

2. Packages

   - today's packages are:
     - `tidyverse`: for wrangling (`dplyr`) and plotting (`ggplot2`)
     - `languageR`: for linguistic datasets

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

3. Data

   - we're working again with the `lexdec` dataset from the `languageR` package (**languageR-package?**)
   - store it as an object with the name `df_lexdec`
   - we also transform the `RT` variable so that it is in milliseconds (it was previously in log milliseconds, but don't worry about understanding what that means)
   - and we choose 10 variables that are relevant for us today

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

2 Data Wrangling

- in English, wrangling refers to a long, difficult process
  - e.g., cowboys wrangle their cattle or herd (gather, collect their animals)
- there are two major parts of wrangling
  - transforming: sorting or creating new variables (what we'll do today)
  - tidying: reshaping or structuring your data (we'll do this in a few weeks)
- both data tidying and transforming require the `dplyr` package from the `tidyverse`
  - `dplyr` functions are often referred to as verbs, because they *do* something

The `dplyr` name

- the `dplyr` name comes from a previous package, `plyr` which is used to split apart, apply functions to, and combine data
  - in English, `plyr` sounds like the word for pliers ("pliers"), which are used to pry things apart, like what `plyr` does with data
  - the `d` in `dplyr` was added because the package is specifically for working with data frames

2.1 `lexdec`

- the `lexdec` dataset contains data for a lexical decision task in English
  - let's take a look at the dataset using the `head()` function, which just prints the first 6 rows
    - here we're telling it to print the first 10 rows
- in my materials I often use the `head()` function to avoid printing the whole dataset in the output, but you wouldn't generally want to use `head()` when looking at your data, you'd want to look at your whole dataset

Aufgabe 2.1: `df_lexdec`

Beispiel 2.1.

1. Look at the dataset
   - how many observations are there?
   - how many variables are there?
2. Feed the dataset into the `glimpse()` function
   - what does this show you?
   - how does it compare to what you see when you use `summary()`?

2.2 `dplyr` basics

- today we'll learn some of the primary `dplyr` verbs (functions) that allow us to solve the majority of our data manipulation challenges
  - I use these verbs multiple times in probably every analysis script
- `dplyr` verbs have some things in common:

1. the first argument is always a data frame
2. the following arguments typically describe which columns to be operated on, using the variable name (without quotation marks)
3. the output is always a new dataframe

- the verbs all do one thing well, so we often want to use multiple verbs at once
  - we use the pipe to do this (`|>` or `|>`)
  - we've already seen this pipe when we feed a dataframe into `ggplot()`
  - we can read the pipe as `and then`
- in the following code, identify
  - the data frame
  - `dplyr` verbs
  - variable names
- can you try to read out (guess) what the following code does?

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

- note that `A1` is written with quotation marks, but none of the other code is
  - when calling on an object (e.g., `df_lexdec`) or its variables (e.g., `Subject`), we do not wrap them in quotation marks
  - when we are calling on a certain *value* of a variable that is not numerical, we must wrap this value in quotation marks
  - because the Subject ID `A1` is a value of the variable `Subject`, we must use quotation marks around it
- try removing the quotation marks, what error message do you get?
- try adding quotation marks around a variable name, what error message do you get?
  - this is an important exercise, because you will often find your code will not run, but the solution is often something as simple as missing or extra quotation marks or punctuation

3 Rows

- in tidy data, rows represent observations
- the most important verbs for rows are:

    - `filter()`: changes which rows are present
    - `arrange()`: changes the order of rows

3.1 `filter()`

- changes which rows are present without changing their order
- takes the dataframe as first argument
    - following arguments are conditions that must be `TRUE` to keep the row
- find all reaction times that were longer than 450 milliseconds:

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

```
  Subject       RT Trial Sex NativeLanguage Correct       Word Frequency  Class
1      A1 566.9998    23   F        English correct        owl  4.859812 animal
2      A1 548.9998    27   F        English correct       mole  4.605170 animal
3      A1 572.0000    29   F        English correct     cherry  4.997212  plant
4      A1 486.0002    30   F        English correct       pear  4.727388  plant
6      A1 483.0002    33   F        English correct blackberry  4.060443  plant
8      A1 524.9999    38   F        English correct   squirrel  4.709530 animal
  Length
1      3
2      4
3      6
4      4
6     10
8      8
```

- notice that we don't put the reaction time value in quotation marks, because it is *numerical*
- if you want to save the filtered data, it's usually wise to save it with a *new* object name
    - unless you want to overwrite the pre-filtered version, a new name is necessary

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

Logical operators

- symbols used to describe a logical condition
- `==` *is idential* (`1 == 1`)
- `!=` *is not identical* (`1 != 2`)
- `>` *is greater than* (`2 > 1`)
- `<` *is less than* (`1 < 2`)
- to combine conditions
  - `&` or `,` *and also* (for multiple conditions)
  - `|` *or* (for multiple conditions)
- there's a nice shortcut for cominbing `==` and `|`: `%in%`
  - keeps rows where the variable equals one of the values on the right

3.1.1 `==` and `|`

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

```
     Subject        RT Trial Sex NativeLanguage   Correct     Word Frequency
1         A1  566.9998    23   F        English   correct      owl  4.859812
4         A1  486.0002    30   F        English   correct     pear  4.727388
475       A2  561.0001    23   M        English   correct      dog  7.667626
949        C  688.0001    23   F        English   correct  vulture  4.248495
83         D  553.0000    30   M          Other   correct  walnut  4.499810
317        J  824.0004    23   F          Other   correct   beaver  3.951244
320        J  568.9998    30   F          Other   correct   carrot  4.976734
791        K  407.9999    23   F        English   correct      owl  4.859812
793        K  459.9998    30   F        English   correct  vulture  4.248495
1581      M2  941.9997    23   F          Other incorrect  paprika  2.484907
1585      M2  628.9998    30   F          Other   correct   donkey  5.541264
159        P 1103.0000    23   F          Other incorrect    moose  2.708050
1345      R1  483.0002    30   F        English   correct      ant  5.347108
1112      R2  601.0000    30   M        English   correct    snake  6.120297
1268      R3  422.9999    30   M        English   correct      dog  7.667626
558       T1  576.9998    30   F        English   correct broccoli  2.833213
1423       V 1013.9998    23   F          Other incorrect    stork  3.044522
241        Z  640.9997    30   M          Other   correct    squid  3.970292
      Class Length
1    animal      3
4     plant      4
```

```
475   animal      3
949   animal      7
83     plant      6
317   animal      6
320    plant      6
791   animal      3
793   animal      7
1581   plant      7
1585 animal       6
159   animal      5
1345 animal       3
1112 animal       5
1268 animal       3
558    plant      8
1423 animal       5
241   animal      5
```

3.1.2 %in%

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

```
     Subject        RT Trial Sex NativeLanguage   Correct     Word Frequency
1         A1  566.9998    23   F        English   correct      owl  4.859812
4         A1  486.0002    30   F        English   correct     pear  4.727388
475       A2  561.0001    23   M        English   correct      dog  7.667626
949        C  688.0001    23   F        English   correct  vulture  4.248495
83         D  553.0000    30   M          Other   correct   walnut  4.499810
317        J  824.0004    23   F          Other   correct   beaver  3.951244
320        J  568.9998    30   F          Other   correct   carrot  4.976734
791        K  407.9999    23   F        English   correct      owl  4.859812
793        K  459.9998    30   F        English   correct  vulture  4.248495
1581      M2  941.9997    23   F          Other incorrect  paprika  2.484907
1585      M2  628.9998    30   F          Other   correct   donkey  5.541264
159        P 1103.0000    23   F          Other incorrect    moose  2.708050
1345      R1  483.0002    30   F        English   correct      ant  5.347108
1112      R2  601.0000    30   M        English   correct    snake  6.120297
1268      R3  422.9999    30   M        English   correct      dog  7.667626
558       T1  576.9998    30   F        English   correct broccoli  2.833213
1423       V 1013.9998    23   F          Other incorrect    stork  3.044522
241        Z  640.9997    30   M          Other   correct    squid  3.970292
```

```
       Class Length
1     animal      3
4      plant      4
475   animal      3
949   animal      7
83     plant      6
317   animal      6
320    plant      6
791   animal      3
793   animal      7
1581   plant      7
1585  animal      6
159   animal      5
1345  animal      3
1112  animal      5
1268  animal      3
558    plant      8
1423  animal      5
241   animal      5
```

Aufgabe 3.1: `filter()`

Beispiel 3.1.

1. Filter the data to include rows from Trial 25 and non-native English speakers (`other`)
2. How many rows are there?

3.2 `arrange()`

- changes the order of the rows based on a value in a column(s)

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

```
    Subject       RT Trial Sex NativeLanguage   Correct    Word Frequency
542      A2 340.0001   159   M        English incorrect     pig  6.660575
815       K 347.9998    83   F        English incorrect   lemon  5.631212
822       K 363.0001    99   F        English incorrect  potato  6.461468
73       A1 364.9999   174   F        English   correct chicken  6.599870
```

```
524        A2 365.9999   117    M         English    correct    goose  5.267858
1516        I 367.0001    51    F           Other    correct   carrot  4.976734
      Class Length
542  animal      3
815   plant      5
822   plant      6
73   animal      7
524  animal      5
1516  plant      6
```

- if you use more than one column name, each additional column will be used to break ties between values of the preceding columns

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

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

- we can add `desc()` inside `arrange()` to use descending order (big-to-small) instead of the default ascending order

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

```
    Subject       RT Trial Sex NativeLanguage Correct       Word Frequency
6        A1 483.0002    33   F        English correct blackberry  4.060443
7        A1 417.9998    34   F        English correct strawberry  4.753590
69       A1 540.9998   168   F        English correct woodpecker  2.890372
505      A2 503.9999    87   M        English correct woodpecker  2.890372
516      A2 400.9998   105   M        English correct strawberry  4.753590
518      A2 517.0001   108   M        English correct blackberry  4.060443
     Class Length
6    plant     10
7    plant     10
69  animal     10
505 animal     10
516  plant     10
518  plant     10
```

Aufgabe 3.2: `arrange()`

Beispiel 3.2.

1. Filter the data to include observations from only the `Subject` M1 and W2, *and then*
2. Arrange the data by descending reaction times

4 Columns

- in tidy data, columns represent variables
- the most important verbs for columns are:
  - `rename()`: changes the names of the columns
  - `mutate()`: creates new columns that are derived from the existing columns
  - `select()`: changes which columns are present
  - `relocate()`: changes the positions of the columns

4.1 `rename()`

- `rename()` lets us change the name of columns
  - the order of the arguments is `new_name` = `old_name`
- let's try changing some of the variable names to German
  - I tend to create variable names with lower case, as a coding convention

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

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

4.2 `mutate()`

- `mutate()` creates new columns from existing columns
  - e.g., we can perform basic algebra on the values in each column

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

```
  teilnehmer    rz_ms Trial geschlect NativeLanguage Correct       Word
1         A1 566.9998    23         F        English correct        owl
2         A1 548.9998    27         F        English correct       mole
3         A1 572.0000    29         F        English correct     cherry
4         A1 486.0002    30         F        English correct       pear
5         A1 414.0000    32         F        English correct        dog
6         A1 483.0002    33         F        English correct blackberry
  Frequency  Class laenge rz_laenge
```

```
1  4.859812 animal      3 188.99994
2  4.605170 animal      4 137.24994
3  4.997212  plant      6  95.33333
4  4.727388  plant      4 121.50005
5  7.667626 animal      3 138.00000
6  4.060443  plant     10  48.30002
```

- `mutate()` adds these new columns to the right of your dataset
  - this makes it difficult to see what's happening
- to control where the new column is added, we can use `.before` or `.after`

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

```
  teilnehmer    rz_ms rz_laenge Trial geschlect NativeLanguage Correct
1         A1 566.9998 188.99994    23         F        English correct
2         A1 548.9998 137.24994    27         F        English correct
3         A1 572.0000  95.33333    29         F        English correct
4         A1 486.0002 121.50005    30         F        English correct
5         A1 414.0000 138.00000    32         F        English correct
6         A1 483.0002  48.30002    33         F        English correct
        Word Frequency  Class laenge
1        owl  4.859812 animal      3
2       mole  4.605170 animal      4
3     cherry  4.997212  plant      6
4       pear  4.727388  plant      4
5        dog  7.667626 animal      3
6 blackberry  4.060443  plant     10
```

4.3 Exercise

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

4.4 `select()`

- `select()` subsets the data to include only the columns you want
- select columns by name

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

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

- select all columns between `rz_ms` and `geschlecht`

```r
df_lexent |>
  select(rz_ms:geschlect) |>
  head()
```

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

- select all columns except `rz_s` (`!` is read as "not")

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

```
  teilnehmer    rz_ms Trial geschlect NativeLanguage Correct       Word
1         A1 566.9998    23         F        English correct        owl
2         A1 548.9998    27         F        English correct       mole
3         A1 572.0000    29         F        English correct     cherry
```

```
4          A1 486.0002    30          F        English correct       pear
5          A1 414.0000    32          F        English correct        dog
6          A1 483.0002    33          F        English correct blackberry
  Frequency  Class laenge
1  4.859812 animal      3
2  4.605170 animal      4
3  4.997212  plant      6
4  4.727388  plant      4
5  7.667626 animal      3
6  4.060443  plant     10
```

4.5 `select()` helper functions

- some helper functions that make life easier when working with `select()`:
  - `starts_with("abc")`: selects columns that begin with a certain string of characters
  - `ends_with("xyz")`: selects columns that end with a certain string of characters
  - `contains("ijk")`: selects columns that contain a certain string of characters
  - `where(is.character)`: selects columns that match a logical criteria
    * e.g., the function `is.character()` returns the value `TRUE` when a variable contains character strings, not numerical values or categories

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

```
        Word
1        owl
2       mole
3     cherry
4       pear
5        dog
6 blackberry
```

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

```
  teilnehmer
1         A1
2         A1
```

```
3          A1
4          A1
5          A1
6          A1
```

Aufgabe 4.1: `select()`

Beispiel 4.1.

1. Print the columns in `df_lexent` that begin with "t"
2. Print the columns in `df_lexent` that contain "ge"
3. Print the columns in `df_lexent` that
    - begin with begin with "r", and
    - end with "s"

4.6 `relocate()`

- `relocate()` moves variables around
    - by default, it moves them to the front

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

```
  Trial teilnehmer    rz_ms      rz_s geschlect NativeLanguage Correct
1    23         A1 566.9998 0.5669998         F        English correct
2    27         A1 548.9998 0.5489998         F        English correct
3    29         A1 572.0000 0.5720000         F        English correct
4    30         A1 486.0002 0.4860002         F        English correct
5    32         A1 414.0000 0.4140000         F        English correct
6    33         A1 483.0002 0.4830002         F        English correct
        Word Frequency  Class laenge
1        owl  4.859812 animal      3
2       mole  4.605170 animal      4
3     cherry  4.997212  plant      6
4       pear  4.727388  plant      4
5        dog  7.667626 animal      3
6 blackberry  4.060443  plant     10
```

- but we can also use `.before` or `.after` to place a variable

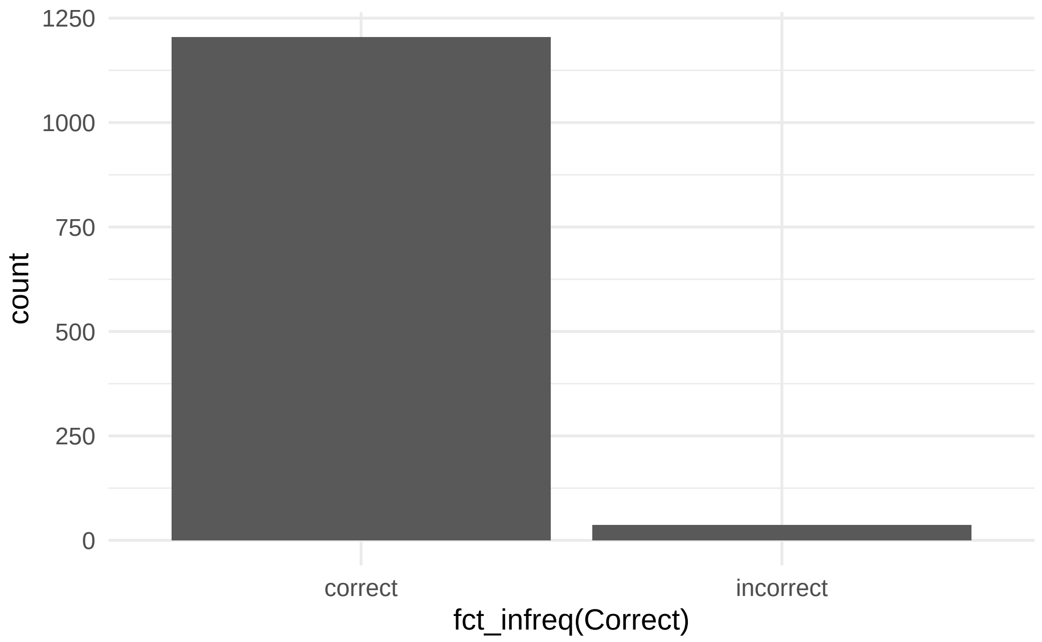
```
df_lexent |>
  relocate(Trial, .after = teilnehmer) |>
  head()
```

```
  teilnehmer Trial    rz_ms      rz_s geschlect NativeLanguage Correct
1         A1    23 566.9998 0.5669998         F        English correct
2         A1    27 548.9998 0.5489998         F        English correct
3         A1    29 572.0000 0.5720000         F        English correct
4         A1    30 486.0002 0.4860002         F        English correct
5         A1    32 414.0000 0.4140000         F        English correct
6         A1    33 483.0002 0.4830002         F        English correct
        Word Frequency  Class laenge
1        owl  4.859812 animal      3
2       mole  4.605170 animal      4
3     cherry  4.997212  plant      6
4       pear  4.727388  plant      4
5        dog  7.667626 animal      3
6 blackberry  4.060443  plant     10
```

5 dplyr and ggplot2

- we can change a dataset using the `dplyr` verbs, and then feed these changes into `ggplot2`
- what will the following code produce?

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



- important: we can use pipes (`|>`) to perform additional verbs/functions
  - but the `ggplot()` function uses `+` to add new *layers* to the plot

5.1 Exercises

1. In a single pipeline, print `df_lexent` where you `select` only the reaction times (in milliseconds), NativeLanguage, and Word columns for rows that meet each of the following conditions, `arrange` them in order of reaction times, and `filter` them to include only:
   - reaction times were greater than 500ms *and* less than 550ms
   - were from the words "pear", "elephant", or "tortoise"
2. Sort `df_lexent` in descending order to find the trials with longest reaction times.
3. In a single pipeline, store a new object called `df_rz` which contains `df_lexent`, *and then*:
   - select the variables `teilnehmer`, `NativeLanguage`, `Word`, `rz_s`, `laenge`, and `Frequency`
   - create a new variable `rz_s_laenge`, that is `rz_s` divided by `laenge`
     - and is placed before `laenge`
   - rename these variables in English so that they are in German (and with lower case)

Heutige Ziele

Today we learned…

- how to wrangle data using the `dplyr` package from the `tidyverse`
- learn to use uses the `pipe` (`|>`) to feed the result of one function into another function
- about functions that operate on rows
- about functions that operate on columns
- how to combine `dplyr` functions with plots from `ggplot2`

Session Info

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

```r
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] languageR_1.5.0 lubridate_1.9.3 forcats_1.0.0   stringr_1.5.1
 [5] dplyr_1.1.4     purrr_1.0.2     readr_2.1.5     tidyr_1.3.1
 [9] tibble_3.2.1    ggplot2_3.5.1   tidyverse_2.0.0

loaded via a namespace (and not attached):
 [1] gtable_0.3.5      jsonlite_1.8.8    compiler_4.4.0    renv_1.0.7
 [5] tinytex_0.50      tidyselect_1.2.1  scales_1.3.0      yaml_2.3.8
 [9] fastmap_1.1.1     R6_2.5.1          labeling_0.4.3    generics_0.1.3
[13] knitr_1.46        munsell_0.5.1     pillar_1.9.0      tzdb_0.4.0
[17] rlang_1.1.3       utf8_1.2.4        stringi_1.8.3     xfun_0.43
[21] timechange_0.3.0  cli_3.6.2         withr_3.0.0       magrittr_2.0.3
[25] digest_0.6.35     grid_4.4.0        rstudioapi_0.16.0 hms_1.1.3
[29] lifecycle_1.0.4   vctrs_0.6.5       evaluate_0.23     glue_1.7.0
[33] farver_2.1.1      fansi_1.0.6       colorspace_2.1-0  rmarkdown_2.26
[37] tools_4.4.0       pkgconfig_2.0.3   htmltools_0.5.8.1
```

Literaturverzeichnis

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/