Deskriptive Statistik

Maße der zentralen Tendenz und Streuung

Daniela Palleschi

2023-12-05

Inhaltsverzeichnis

Learning objects

Today we will learn…

- about measures of central tendency (mean, median, mode)
- about measures of dispersion (range, standard deviation)
- how to use the `summarise()` function from `dplyr`
- how to produce summaries `.by` group

Readings

The required readings for this topic are:

1. Ch. 3, Sections 3.4-3.9 (*Descriptive statistics, models, and distributions*) in Winter (2019) (available online for students/employees of the HU Berlin via the HU Grimm Zentrum.
2. Section 4.5 (Groups) in Ch. 4 (*Data Transformation*) in Wickham et al. (2023).

1 Set-up

1.1 Clear Environment

- *always* start a new script with a clear R environment
  - no objects stored in the Environment
  - no packages loaded
- click `Session > Restart R` to start with a fresh environment
  - or the keyboard shortcut `Cmd/Ctrl+Strg+0`

1.2 Packages

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

1.3 Load data

- two datasets today:
  - `groesse_geburtstag_ws2324.csv`: a slightly changed `groesse_geburtstag` dataset from last week
  - `languageR_english.csv`: condensed version of `english` dataset from the `languageR` package
- if you don't have these data already, download them from Moodle

```
df_groesse <- read_csv(here("daten", "groesse_geburtstag_ws2324.csv"))
```

```
df_eng <- read_csv(here("daten", "languageR_english.csv")) |>
  clean_names() |>
  # fix some wonky variable names:
  rename(rt_lexdec = r_tlexdec,
         rt_naming = r_tnaming)
```

2 Deskriptive statistics

- quantitatively describe the central tendency, variability, and distribution of data
  - also called summary statistics
- e.g., range of values (minimum, maximum), the mean value, and the standard deviation

2.1 Number of observations (n)

- not a statistic, but is important information
  - more data (higher n) = more evidence
  - less data (lower n) = may not be generalisable to the broader population
- `nrow()`: get number of observations in a dataset

```
nrow(df_groesse)
```

```
[1] 9
```

2.2 Measures of central tendency

- quantitavely describe the centre of our data
  - the mean, median, and mode

2.2.1 Mean (μ)

- the `mean`, or average: the sum of all values divided by the number of values (as in Equation 1)

$$\mu = \frac{sum\ of\ values}{n} \tag{1}$$

- let's take our heights from last week: (`171, 168, 182, 190, 170, 163, 164, 167, 189`)
- we can calculate the mean by hand

```
171+ 168+ 182+ 190+ 170+ 163+ 164+ 167+ 189 / 9
```

```
[1] 1396
```

- but a mean of 1396 cm doesn't make sense
- we can fix the equation above by wrapping the heights with parantheses (`()`) before dividing by n

```
(171+ 168+ 182+ 190+ 170+ 163+ 164+ 167+ 189) / 9
```

```
[1] 173.7778
```

- we can also save the results of an equation as an object
  - or multiple values as a vector (a list of values of the same class)

```
# save heights as a vector
heights <- c(171, 168, 182, 190, 170, 163, 164, 167, 189)
```

- we could then use the functions `sum()` and `length()` to compute the mean

```
# divide the sum of heights by the n of heights
sum(heights)/length(heights)
```

```
[1] 173.7778
```

- or simply use the `mean()` function.

```
# or use the mean() function
mean(heights)
```

```
[1] 173.7778
```

- we can run the `mean()` function on a variable in a data frame by using the `$` operator (`dataframe$variable`).

```
mean(df_groesse$groesse)
```

```
[1] 173.6667
```

2.2.2 Median

- the value in the middle of the dataset
- if you line up your data in order of value, half of the data lie below the median, and half above it

2.2.2.1 Median in R

- we can use the `sort()` function and count which is the middle value:

```
sort(df_groesse$groesse)
```

```
[1] 163 164 167 167 170 171 182 189 190
```

- we could alternatively just use the function `median()`

```
median(df_groesse$groesse)
```

```
[1] 170
```

2.2.3 The median is robust, the mean is not

- the median is not affected by outliers/extreme values (unlike the mean)
  - what happens to the mean and median when we change our tallest height (190cm) to be the height of the current tallest person in the world: 251 cm?

```
df_groesste <-
  df_groesse |>
  mutate(groesse = ifelse(groesse == 190, 251, groesse))
```

```
sort(df_groesste$groesse)
```

```
[1] 163 164 167 167 170 171 182 189 251
```

```
median(df_groesste$groesse)
```

```
[1] 170
```

```
mean(df_groesste$groesse)
```

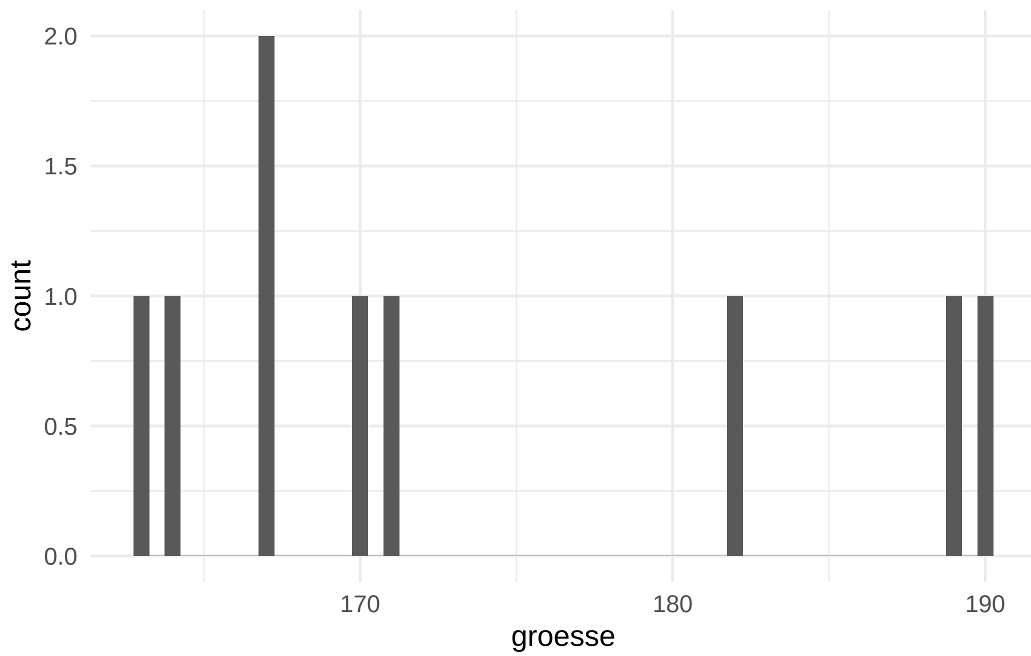
```
[1] 180.4444
```

- the median is often reported instead of the mean for data that has heavy skews to more extreme values, such as when reporting incomes in a population

2.2.4 Mode

- the value that occurs the most in a data set
- no R function to determine the `mode`
  - but we can visualise it, e.g., with a histogram or a density plot

```
df_groesse |>
  ggplot(aes(x = groesse)) +
  geom_histogram(binwidth = .5) +
  theme_minimal()
```



2.3 Measures of dispersion

- describe the spread of data points
  - tell us something about how the data whole is distributed

2.3.1 Range

- can refer to the highest (maximum) and lowest (minimum) values
  - or the difference between highest and lowest value

- `max()` and `min()`: print the highest and lowest values

```r
max(heights)
```

```
[1] 190
```

```r
min(heights)
```

```
[1] 163
```

- or use the `range()` function

```r
range(heights)
```

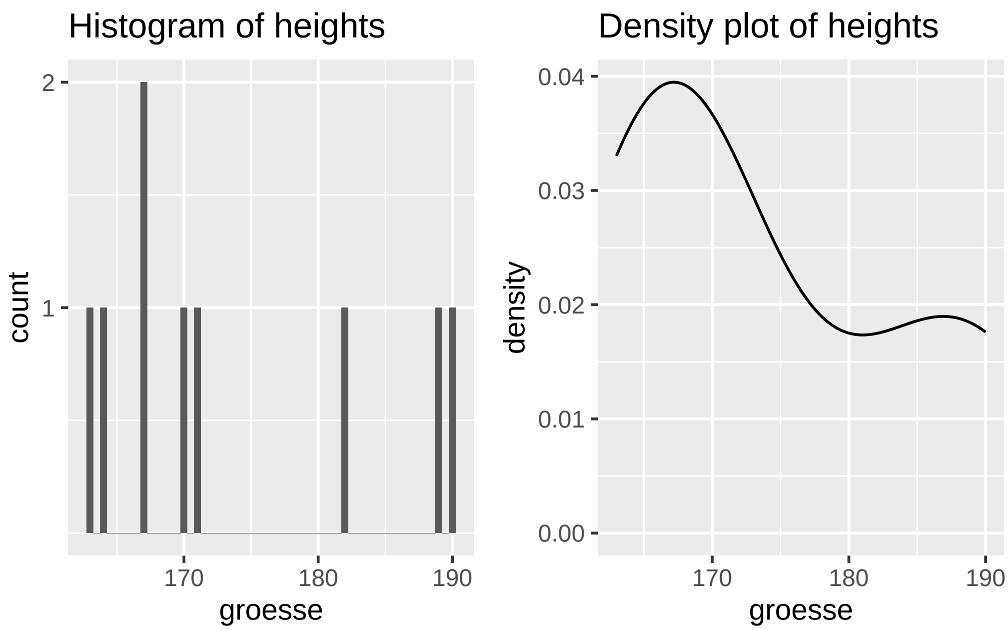
```
[1] 163 190
```

- we can get the difference between these values by subtracting the minimum value from the maximum value

```r
max(heights) - min(heights)
```

```
[1] 27
```

- In a histogram or density plot: the lowest and heights values on the x-axis



2.3.2 Standard deviation (`sd` or σ)

- a measure of how dispersed data is *in relation to the mean*
  - a low standard deviation means data are clustered around the mean (i.e., there is less spread)
  - a high standard deviation means data are more spread out
- standard deviation is very often reported whenever mean is reported

---

- Standard deviation (`sd`) = the square root ($\sqrt{}$ or `sqrt()` in R) of the sum of squared value deviations from the mean ($(x - \mu)^2$) divided by the number of observations minus 1 ($n - 1$)
  - given in Equation 2

$$\sigma = \sqrt{\frac{(x_1 - \mu)^2 + (x_2 - \mu)^2 + ... + (x_N - \mu)^2}{N - 1}} \tag{2}$$

- this looks intimidating, but we can calcuate standard deviation in R using the `sd()` function

```
sd(heights)
```

```
[1] 10.46157
```

- we can calculate standard deviation by hand if we know:
  - the value of each observation
  - the mean of these values
  - the number of observations

$$\sigma_{heights} = \sqrt{\frac{(height_1 - \mu)^2 + (height_2 - \mu)^2 + ...(heights_N - \mu)^2}{N - 1}} \tag{3}$$

- For example, in a vector with 3 observations (`3,5,9`), our values (x) are:

```
values <- c(3,5,16)
values
```

```
[1]  3  5 16
```

- adding these to Equation 2 we get Equation 4

$$\sigma_{values} = \sqrt{\frac{(3 - \mu)^2 + (5 - \mu)^2 + (16 - \mu)^2}{N - 1}} \tag{4}$$

- our mean (μ) is:

```
mean(values)
```

```
[1] 8
```

- adding these to Equation 4, we get Equation 5.

$$\sigma_{values} = \sqrt{\frac{(3-8)^2 + (5-8)^2 + (16-8)^2}{N-1}} \tag{5}$$

- the number of observations (n) is:

```
length(values)
```

```
[1] 3
```

- adding these to Equation 5 we get Equation 6.

$$\sigma_{values} = \sqrt{\frac{(3-8)^2 + (5-8)^2 + (16-8)^2}{3-1}} \tag{6}$$

- carrying out the remaining operations we get Equations 8 through 2:

$$\begin{aligned}
\sigma_{values} &= \sqrt{\frac{(-5)^2 + (-3)^2 + (8)^2}{3-1}} && (7) \\
& && (8) \\
&= \sqrt{\frac{25+9+64}{3-1}} && (9) \\
&= \sqrt{\frac{98}{2}} && (10) \\
&= \sqrt{49} && (11) \\
&= 7 && (12)
\end{aligned}$$

- check our work:

```
sd(values)
```

```
[1] 7
```

3 Summary statistics with R

- the `dplyr` package from the `tidyverse` has some helpful functions to produce summary statistics
- let's now use the `df_eng` dataset to learn about these `dplyr` verbs.

3.1 `dplyr::summarise`

- `summarise()` function (`dplyr`) computes summaries of data
  - but we have to tell it *what* to compute, and for which variable(s)
- for example, the `n()` function produces the number of observations (only when used inside `summarise()` or `mutate()`)

```
df_eng |>
  summarise(N = n())
```

```
# A tibble: 1 x 1
      N
  <int>
1  4568
```

- we can also run multiple computations at once
  - let's also generate the mean and standard deviation of the lexical decision task (`rt_lexdec`, in milliseconds)

```
df_eng |>
  summarise(mean_lexdec = mean(rt_lexdec, na.rm=T),
            sd_lexdec = sd(rt_lexdec, na.rm = T),
            N = n())
```

```
# A tibble: 1 x 3
  mean_lexdec sd_lexdec     N
        <dbl>     <dbl> <int>
1        708.      115.  4568
```

Missing values

- calculations aren't possible if there are missing values
  - the variable `rt_naming` has a missing value
  - the `mean()` function does not work with missing values

```
df_eng |>
  summarise(mean_naming = mean(rt_naming))
```

```
# A tibble: 1 x 1
  mean_naming
        <dbl>
1          NA
```

- we can remove them using the verb `drop_na()`

```
df_eng |>
  drop_na() |>
  summarise(mean_naming = mean(rt_naming))
```

```
# A tibble: 1 x 1
  mean_naming
        <dbl>
1        566.
```

4 Grouping variables

- we usually want to *compare* certain groups
  - e.g., comparing `groesse` between L1 speaker groups

4.1 `.by =`

- the `.by =` argument in `summarise()` computes our calculations on groups within a categorical variable

```
df_eng |>
  drop_na() |>
```

```r
  summarise(mean_lexdec = mean(rt_lexdec),
            sd_lexdec = sd(rt_lexdec),
            N = n(),
            .by = age_subject) |>
  arrange(mean_lexdec)
```

```
# A tibble: 2 x 4
  age_subject mean_lexdec sd_lexdec     N
  <chr>             <dbl>     <dbl> <int>
1 young              630.      69.1  2283
2 old                787.      96.2  2284
```

4.2 Group by multiple variables

- we can also group by multiple variables
  - for this we need `concatenate` (`c()`)
- we'll filter to only have a couple of carriers, just so our output isn't too long

---

```r
df_eng |>
  drop_na() |>
  summarise(mean_lexdec = mean(rt_lexdec),
            sd_lexdec = sd(rt_lexdec),
            N = n(),
            .by = c(age_subject, word_category)) |>
  arrange(age_subject)
```

```
# A tibble: 4 x 5
  age_subject word_category mean_lexdec sd_lexdec     N
  <chr>       <chr>               <dbl>     <dbl> <int>
1 old         N                    790.     101.   1452
2 old         V                    780.      86.5   832
3 young       N                    633.      70.8  1451
4 young       V                    623.      65.7   832
```

Tabelle 1: Summary stats of Anscombe's quratet datasets

dataset	mean_x	mean_y
Dataset 1	9	7.5
Dataset 2	9	7.5
Dataset 3	9	7.5
Dataset 4	9	7.5

5 Anscombe's Quartet

- Francis Anscombe constructed 4 datasets in 1973 to illustrate the importance of visualising data before analysing it and building a model
- these four plots represent 4 datasets that all have nearly identical mean and standard deviation, but very different distributions

---

5.1 DatasaurRus

- datasauRus package (Davies et al., 2022) contains some more datasets that have similar `mean` and `sd`, but different distributions
  - given in Tabelle 2

```
pacman::p_load("datasauRus")
```

- but when we plot them, they all look very different (Abbildung 2)!

- so, ***always plot your data***
  - don't just look at the descriptive stats!!
- both are very important to understanding your data
- next week we'll see how to plot our summary statistics

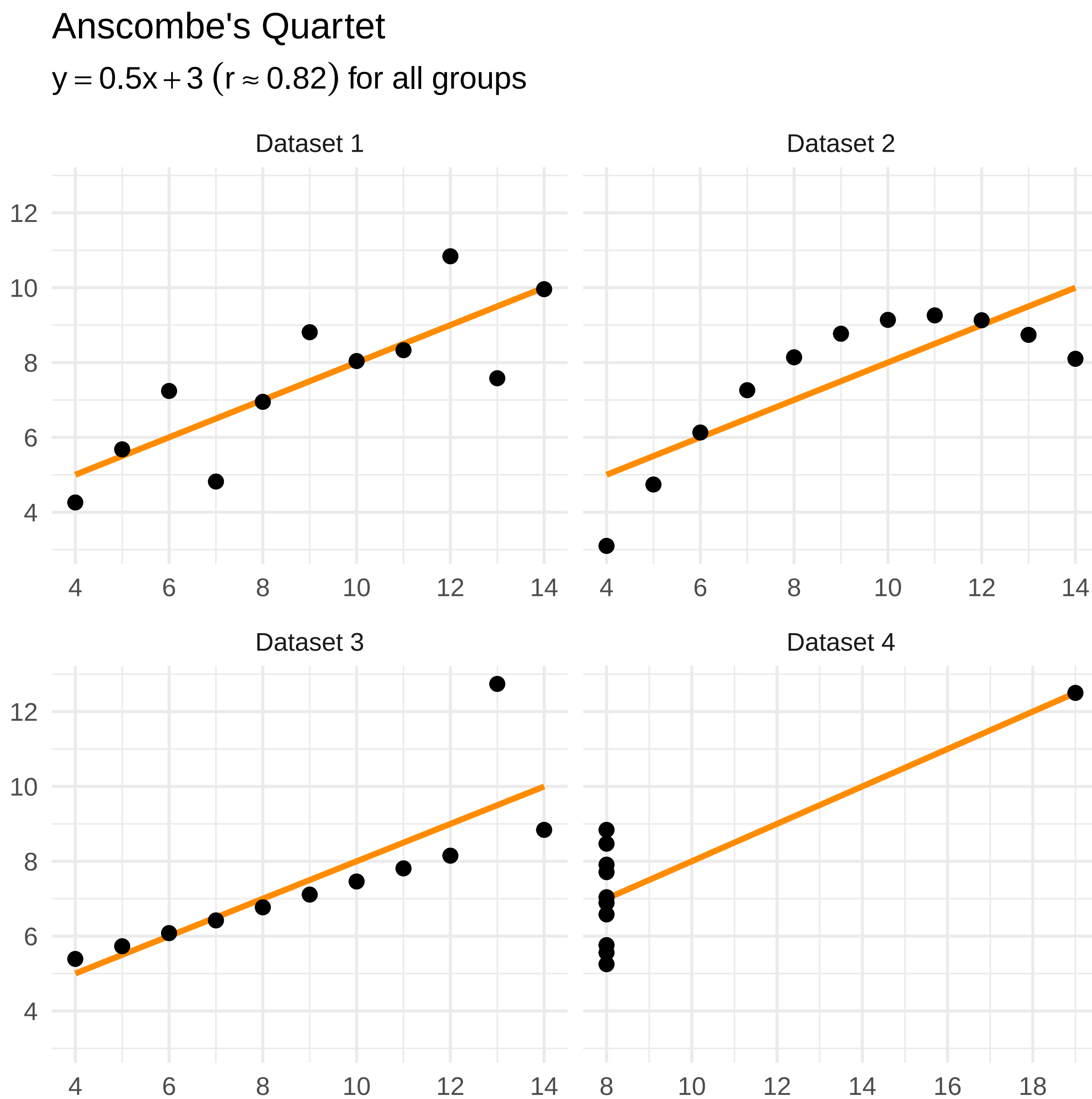


Abbildung 1: Plots of Anscombe's quratet distributions

Tabelle 2: Summary stats of datasauRus datasets

dataset	mean_x	mean_y	std_dev_x	std_dev_y	corr_x
away	54.27	47.83	16.77	26.94	-0
bullseye	54.27	47.83	16.77	26.94	-0
circle	54.27	47.84	16.76	26.93	-0
dino	54.26	47.83	16.77	26.94	-0
dots	54.26	47.84	16.77	26.93	-0
h_lines	54.26	47.83	16.77	26.94	-0
high_lines	54.27	47.84	16.77	26.94	-0
slant_down	54.27	47.84	16.77	26.94	-0
slant_up	54.27	47.83	16.77	26.94	-0
star	54.27	47.84	16.77	26.93	-0
v_lines	54.27	47.84	16.77	26.94	-0
wide_lines	54.27	47.83	16.77	26.94	-0
x_shape	54.26	47.84	16.77	26.93	-0

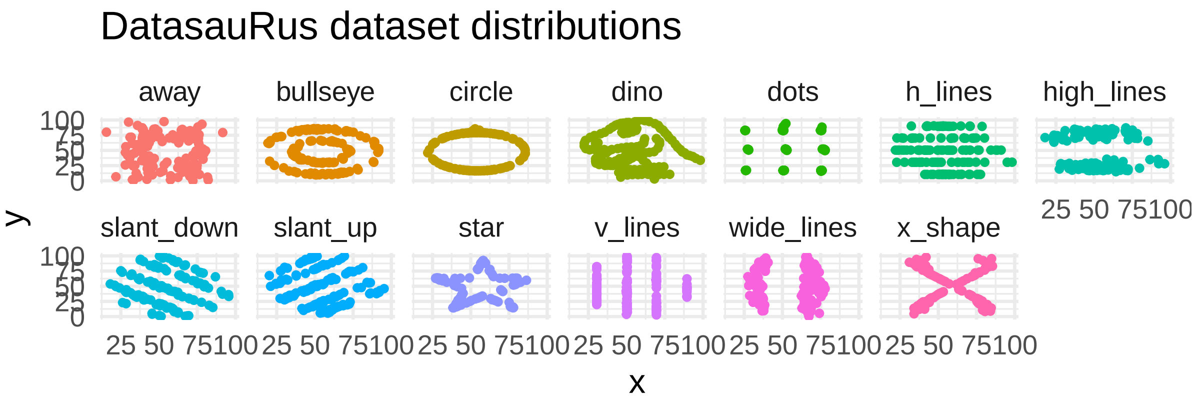


Abbildung 2: Plots of datasauRus dataset distributions

Learning objectives

Today we learned...

- about measures of central tendency
- about measures of dispersion
- how to use the `summarise()` function from `dplyr`
- how to produce summaries `.by` group

6 Aufgaben

1. Calculate the standard deviation of the values `152, 19, 1398, 67, 2111` without using the function `sd()`
   - show your work. The following R syntax might be useful (depending on how you decide to do it):
     - `c()`
     - `mean()`
     - `x^2` calculates the square of a value (here, `x`)
     - `sqrt()` calculates the square root
     - `length()` produces the number of observations in a vector

---

2. Use the function `sd()` to print the standard deviation of the values above. Did you get it right?
3. Using `summarise`, print the mean, standard deviation, and number of observations for `dep_delay`.
   - Hint: do you need to remove missing values (`NA`s)?
4. Do the same, but add the `.by()` argument to find the departure delay (`dep_delay`) per month
   - `arrange()` the output by the mean departure delay

Session Info

Created with R version 4.3.0 (2023-04-21) (Already Tomorrow) and RStudioversion 2023.9.0.463 (Desert Sunflower).

```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;  I

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] datasauRus_0.1.6 patchwork_1.1.3  janitor_2.2.0    here_1.0.1
 [5] lubridate_1.9.2  forcats_1.0.0    stringr_1.5.0    dplyr_1.1.3
 [9] purrr_1.0.2      readr_2.1.4      tidyr_1.3.0      tibble_3.2.1
[13] ggplot2_3.4.3    tidyverse_2.0.0

loaded via a namespace (and not attached):
 [1] gtable_0.3.4      xfun_0.39         lattice_0.21-8    tzdb_0.4.0
 [5] vctrs_0.6.3       tools_4.3.0       generics_0.1.3    parallel_4.3.0
 [9] fansi_1.0.4       pacman_0.5.1      pkgconfig_2.0.3   Matrix_1.5-4
[13] webshot_0.5.4     lifecycle_1.0.3   compiler_4.3.0    farver_2.1.1
[17] munsell_0.5.0     snakecase_0.11.0  htmltools_0.5.5   yaml_2.3.7
[21] pillar_1.9.0      crayon_1.5.2      nlme_3.1-162      tidyselect_1.2.0
[25] rvest_1.0.3       digest_0.6.33     stringi_1.7.12    labeling_0.4.3
[29] splines_4.3.0     rprojroot_2.0.3   fastmap_1.1.1     grid_4.3.0
[33] colorspace_2.1-0  cli_3.6.1         magrittr_2.0.3    utf8_1.2.3
[37] withr_2.5.0       scales_1.2.1      bit64_4.0.5       timechange_0.2.0
[41] rmarkdown_2.22    httr_1.4.6        bit_4.0.5         hms_1.1.3
[45] kableExtra_1.3.4  evaluate_0.21     knitr_1.44        viridisLite_0.4.2
[49] mgcv_1.8-42       rlang_1.1.1       glue_1.6.2        xml2_1.3.4
[53] svglite_2.1.1     rstudioapi_0.14   vroom_1.6.3       jsonlite_1.8.7
[57] R6_2.5.1          systemfonts_1.0.4
```

Literaturverzeichnis

Davies, R., Locke, S., & D'Agostino McGowan, L. (2022). *datasauRus: Datasets from the Datasaurus Dozen*. https://CRAN.R-project.org/package=datasauRus

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