Quantitative Methoden mit R & R-Studio

Sommersemester 2026 | Sitzung 3

Maura Kratz

Willkommen zurück!

I) Vorbereitung	01: Einführung in R und R-Studio	02: Projekte und Datenimport	03: Daten aufbereiten	04: Daten zusammenführen
II) Deskriptiv	05: Kannwerte & deren Visualisierung	06: Häufigkeiten & deren Visualisierung	07: The Grammar of Graphics	08: Korrelation & deskriptive Regression
III) Inferenz	09: Einfache und multiple lineare Regression (OLS)	10: Regression Diagnostics	11: Logistische Regression (GLM)	12: Varianzanalysen (ANOVA)
13: Wahlthema (Faktoranalyse, Indexbildung, ...) / Research Notes				
14: Wahlthema (Faktoranalyse, Indexbildung, ...) / Research Notes				

Recap

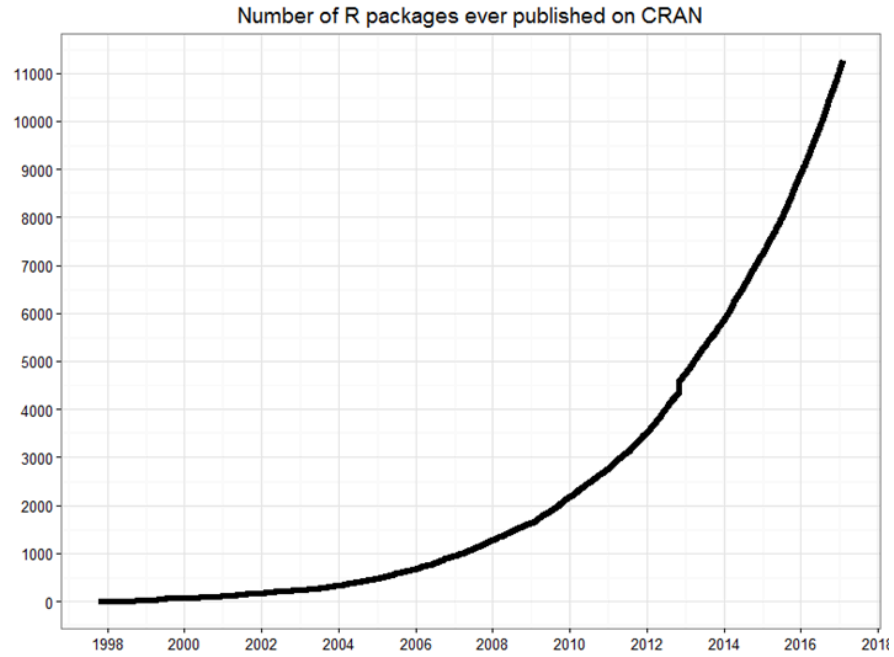
1. Projektordner anlegen und darin arbeiten ✓
2. tidy code und tidy Kommentare schreiben ✓
3. Hilfe suchen ✓
4. Daten einlesen mit dem *rio*-Paket ✓
5. Daten überblicken mit dem *summarytools*-Paket ✓

Was heute ansteht:

- Check-In
- Besprechung der Übung 1
- Was sind Pakete?
- Tidyverse
- Data Cleaning & Wrangling

Pakete

- Einige Funktionen sind bereits in base R enthalten
- Für andere müssen zusätzliche Pakete geladen werden
- Pakete sind Bündel unterschiedlicher Funktionen
- Base R reicht oft nur für sehr basale Operationen und Visualisierungen, für alles andere werden Pakete benötigt



summarytools-Paket

Funktion	Beschreibung	Geeignet für
`dfSummary()`	Übersicht über alle Variablen (Klasse, fehlende Werte, Verteilungen, etc.)	Schnelle Gesamtübersicht eines Datensatzes
`freq()`	Häufigkeitstabellen	Einfache Verteilungen kategorialer Variablen
`descr()`	Deskriptive Statistik (Mittelwert, Median, SD, etc.)	Basisanalyse numerischer Variablen
`ctable()`	Kreuztabellen mit Prozenten und optionalen Tests (z. B.	Zusammenhang zwischen zweikategorialen

summarytools-Paket

Abhängigkeiten:

- `summarytools` greift auf ein anderes Paket namens `cairo` zu
- `cairo` nutzt für die Erstellung von .html-Dateien eine Grafikbibliothek, die in Windows integriert ist, in macOS aber irgendwann ersetzt wurde
- macOS benötigt also ein Programm (XQuartz), das die Grafikbibliothek bereitstellt
- um das Programm manuell nachzuinstallieren:

1. XQuartz herunterladen: xquartz.org
2. RStudio öffnen und das Skript erneut ausführen

↩ Feedback zu Übung 1

- naming convention: uebung_01_name.R (+ gern auch im Dokument selbst angeben)
- Code immer ausführen! Das ist die beste Prüfung, ob er korrekt ist.
- achtet darauf alle benötigten Pakete zu laden (mein Fehler)
- nutzt lieber relative als absolute Pfade
- `install.packages(" ")` vs `library( )` : eins hat Anführungszeichen, das andere nicht
- Aufgabe 4c) zu Labels fiel schwer

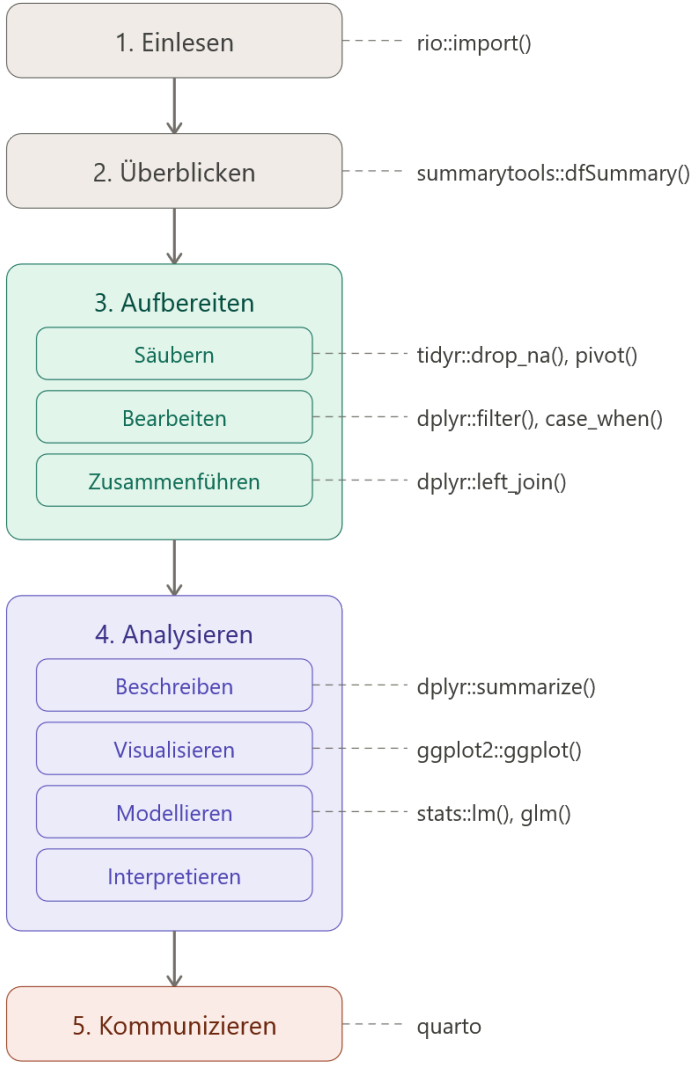
Important

- kursfremde Syntax fällt (negativ) auf!
- leere Übungen oder Kursskripte abzugeben ebenfalls

Übungen als Voraussetzung für die aktive und regelm. Teilnahme

- Es gibt rund 10 Übungen, davon bis zu 1 “Joker”
- nach jeder Abgabe notiere ich:
  - grün = pünktlich und bestanden;
  - gelb = mit Mängeln und/ oder unpünktlich,
  - rot = nicht bestanden oder nicht eingereicht
- am Ende des Semesters bekommen alle, die 9-10 mal grünes Licht bekommen haben (und die übrigen Bestandteile wie Lektüre und mündl. Teilnahme erfüllt haben) ihre aktive und regelmäßige Teilnahme bestätigt
- alles andere sind Einzelfallprüfungen. (Wie oft? Wurde eine Begründung angegeben? Wurde eine Ausgleichsleistung angeboten? etc.)

Datenanalyse-Workflow



Das *tidyverse*-Paketbündel

tidyverse-Logik

Base R

```
df <- read.csv("daten.csv")
df2 <- df[df$jahr == 2025, ]
mittel <- mean(df2$stimmen, na.rm = TRUE)
```

- Viele Zwischenobjekte
- schnell unlesbar durch verschachtelte [ ] bei mehreren Schritten

tidyverse

```
df <- rio::import("daten.csv")
mittel <- df %>%
  dplyr::filter(jahr == 2025) %>%
  dplyr::summarise(
    m = mean(stimmen, na.rm = TRUE)
  )
```

- Ein Objekt, eine Pipeline
- Jede Zeile = ein Schritt
- Schritte lesen sich von oben nach unten wie eine Anleitung: “nimm df, dann filtere, dann berechne.”

Pipes

- Zwei Optionen: Base R pipe: `|>` ODER magrittr pipe: `%>%`
- Die Default-Pipe kann über *Global Options* eingestellt werden
- Zum Einfügen: eintippen oder `Strg + Shift + M`
- Stilregeln:
  - Immer Leerzeichen vor der Pipe
  - Immer neue Zeile danach

```
btw_2025_ergebnisse %>%
  dplyr::filter(jahr == 2025) %>%
  dplyr::select(partei, zweitstimmen)
```

Hands On - Data Wrangling

Datentransformation mit *dplyr*

Befehl	Beschreibung	Bsp
`rename()`	Spaltennamen ändern	`df %>% rename(neu = alt)`
`select()`	Variablen auswählen	`df %>% select(var1, var2)`
`slice()`	Zeilen nach Position auswählen	`df %>% slice(1:10)`
`filter()`	Zeilen nach Werten filtern	`df %>% filter(jahr == 2025)`
`relocate()`	Spaltenreihenfolge ändern	`df %>% relocate(var1, .before = var2)`

Datentransformation mit mit *dplyr*

Befehl	Beschreibung	Bsp
`mutate()`	Neue Variablen hinzufügen	`df %>% mutate(anteil = stimmen / sum(stimmen))`
`summarise()`	Mehrere Werte zusammenfassen	`df %>% summarise(mean(stimmen))`
`count()`	Häufigkeiten zählen	`df %>% count(partei)`
`group_by()`	Operationen gruppenweise ausführen	`df %>% group_by(partei) %>% summarise(mean(stimmen))`

Datentransformation mit mit *tidyr*

Befehl	Beschreibung	Bsp
`pivot_longer()`	Breites Format in langes umwandeln	`df %>% pivot_longer(cols = -id, names_to = "var", values_to = "val")`
`pivot_wider()`	Langes Format in breites umwandeln	`df %>% pivot_wider(names_from = var, values_from = val)`

Änderungen überprüfen - 3 Optionen

1. In der Console ansehen

- Sinnvoll zum Testen, kann aber schnell unübersichtlich werden bei großen Datensätzen
- Änderungen werden nirgends gespeichert

```
btw_2025_ergebnisse %>%
  dplyr::select(gebietsart, gebietsnummer, gruppenname, stimme, prozent, gewahlt) %>%
  head(10)
```

```
   gebietsart gebietsnummer     gruppenname stimme   prozent gewahlt
1        Bund            99 Wahlberechtigte     NA        NA
2        Bund            99        Wählende     NA 82.512200
3        Bund            99       Ungültige      1  0.847738
4        Bund            99       Ungültige      2  0.559080
5        Bund            99         Gültige      1 99.152262
6        Bund            99         Gültige      2 99.440920
7        Bund            99             SPD      1 20.071417
8        Bund            99             SPD      2 16.413301
```

Änderungen überprüfen - 3 Optionen

2. “Nur kurz schauen”

- Daten werden in einem neuen Tab geöffnet.
- Umgeht das Unübersichtlichkeitsproblem.
- Nervt aber ggf. später, wenn das Skript von oben bis unten ausgeführt wird (“Source”-button).

```
btw_2025_ergebnisse %>%
  dplyr::filter(gebietsart != "Einzelbewerber/Wählergruppe") %>%
  View()
```

Änderungen überprüfen - 3 Optionen

3. Speichern oder Überschreiben

- viele unterschiedliche Objekte zu speichern kann unübersichtlich werden
- aber einmal überschriebene Daten schwieriger zurückzuholen

```
btw_2025_ergebnisse_be <-  btw_2025_ergebnisse %>%
  dplyr::filter(gebietsname == "Berlin")
```

```
btw_2025_ergebnisse <- btw_2025_ergebnisse %>%
  dplyr::rename(
    uberg_gebietsart = ueg_gebietsart, # neuer_name = alter_name
    uberg_gebietsnr = ueg_gebietsnummer)
```

Long vs. Wide

Long-Format ist sinnvoll für:

- ggplot2 erwartet fast immer long format
- group_by() + summarise()
- Regressionsmodelle

Wide-Format ist sinnvoll für:

- Korrelationsmatrizen (eine Zeile pro Beobachtungseinheit)
- Vergleichstabellen (Erst- vs. Zweitstimme nebeneinander)
- Export nach Excel

Minute Cards

Bitte füllt die Minute Cards für die heutige Sitzung aus. Das sollt enicht länger als 3 Minuten dauern. Vielen Dank für eure Mitarbeit!

Vielen Dank und bis kommenden Dienstag!

Übung 2 zu “Data Wrangling” bis spätestens Sonntagabend!