awm

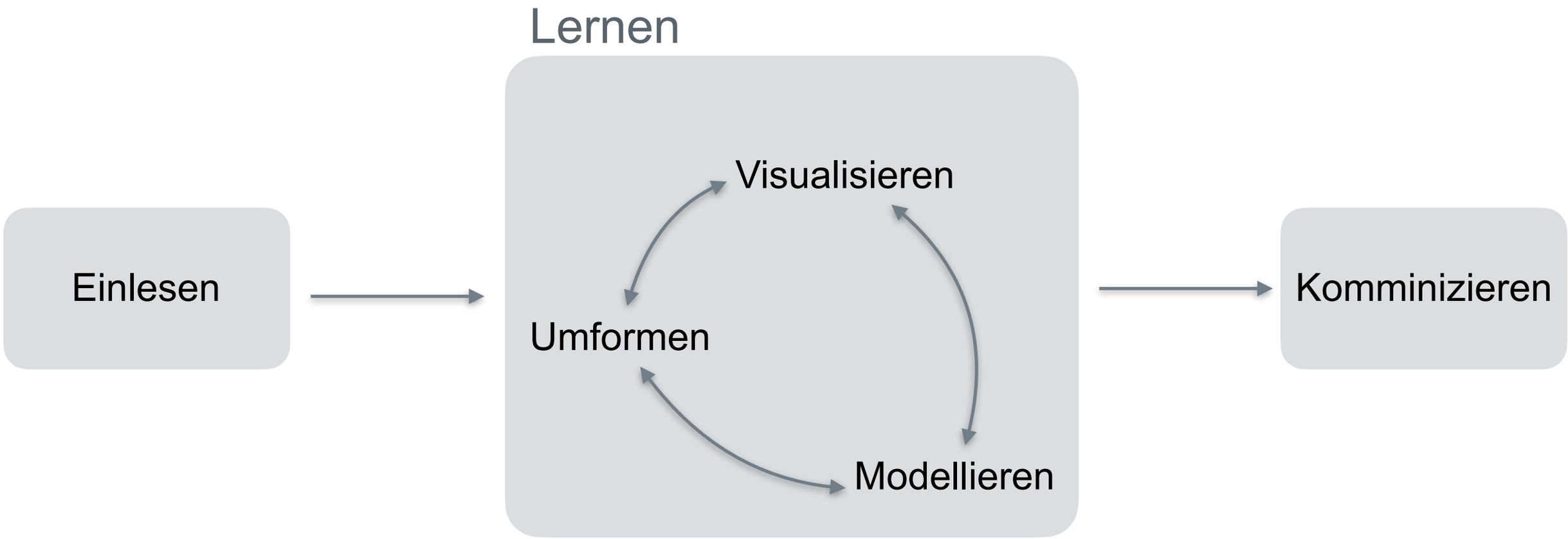
angewandte **w**irtschafts- und **m**edienpsychologie

Thema 3: Datenjudo

QM1, SoSe 22

HOCHSCHULE
ANSBACH

Prozess der Datenanalyse



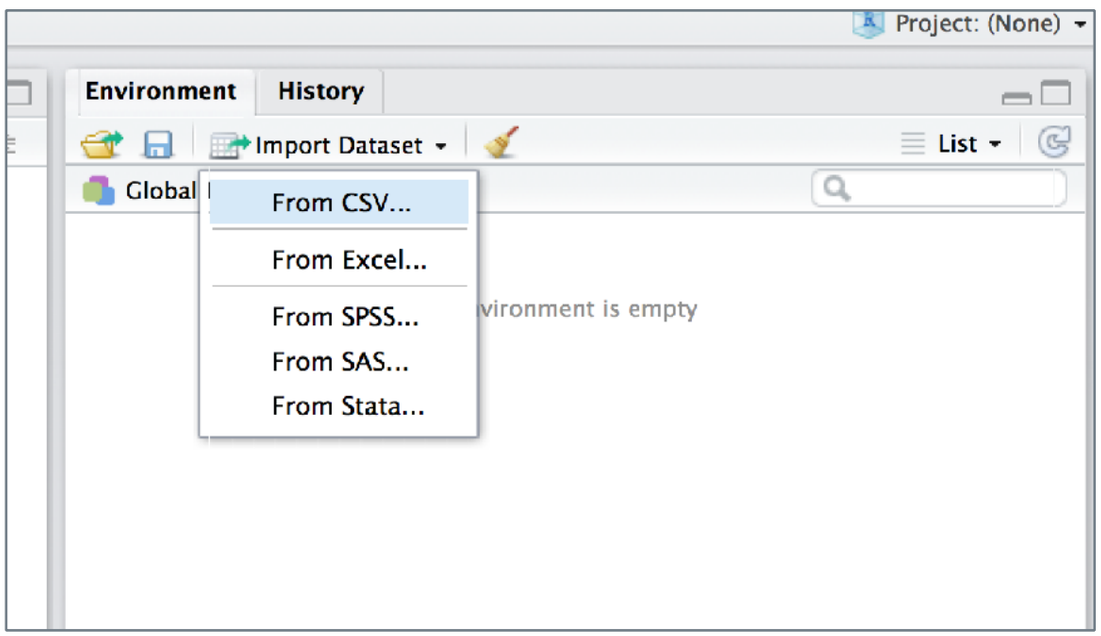

```
# ggf. vorher installieren nicht vergessen
library(tidyverse)
```

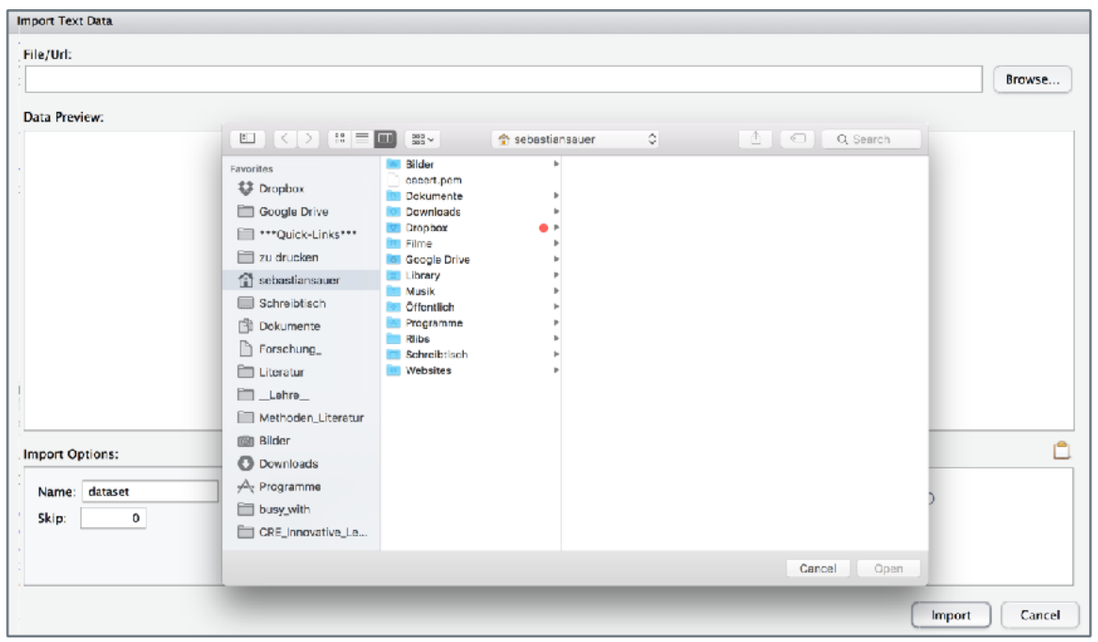
Daten in R importieren

Daten einlesen

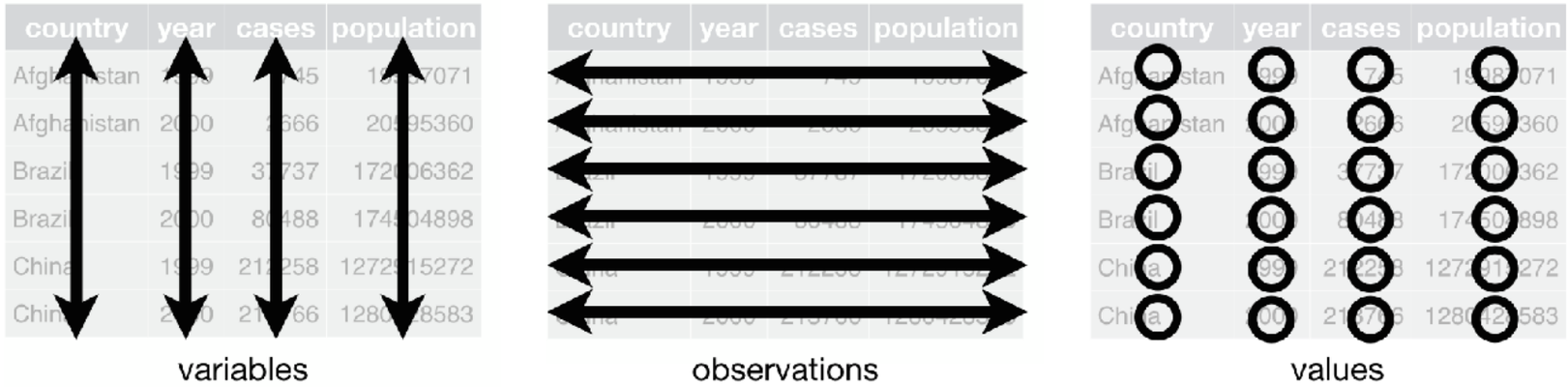
- CSV einlesen:
  `library(readr); my_df <- read_csv("my_file.csv")`
- XLS(X) einlesen:
  `library(readxl); my_df <- read_excel("my_file.xlsx")`
- Lesen Sie in MODAR, Kap. 6.1

RStudio:





Tidy data



[Quelle]

BEISPIEL

ID	Nam	Note	Note	Note
1	Anna	1	2	3
2	Berta	1	1	1
3	Carla	2	3	4
…	…	…	…	…

Die „Rechteckmatrix“ (Normalform) ist der Anfang Ihrer Analyse

- In **jeder Zeile ein Fall** (eine Beobachtungseinheit) und in **jeder Spalte eine Variable**
- Die **erste Spalte** sollte eine **laufende Nummer** (ID) sein
- Die **erste Zeile** sollte die **Variablennamen** enthalten (Spaltenköpfe)
- Verwenden Sie ***keine* Lehr**- oder **Sonderzeichen** (ä,ß,…) und **keine Punkte** in den **Spaltenköpfen**
- Vertiefung

- Verwenden Sie ***keine* Lehr**- oder **Sonderzeichen** (ä,ß,…) und **keine Punkte** in den **Spaltenköpfen**
- In jeder **Zelle** steht ein **Wert** (Zahl oder Text)
- **Fehlt** ein **Wert**, so sollte die entsprechenden Zelle **leer** bleiben
- Keine Leerzeilen
- Keine Farbmarkierungen o.ä.
- Am besten überall nur Standardzeichen (amerikanische Tastatur) verwendenΩVerwenden Sie keine Lehr- oder Sonderzeichen (ä,ß,…) und keine Punkte in den Spaltenköpfen
- In jeder Zelle steht ein Wert (Zahl oder Text)
- Fehlt ein Wert, so sollte die entsprechenden Zelle leer bleiben
- Keine Leerzeilen
- Keine Farbmarkierungen o.ä.
- Am besten überall nur Standardzeichen (amerikanische Tastatur) verwenden

Typische Operationen des Datenjudo

Zeilen filtern

Spalten filtern

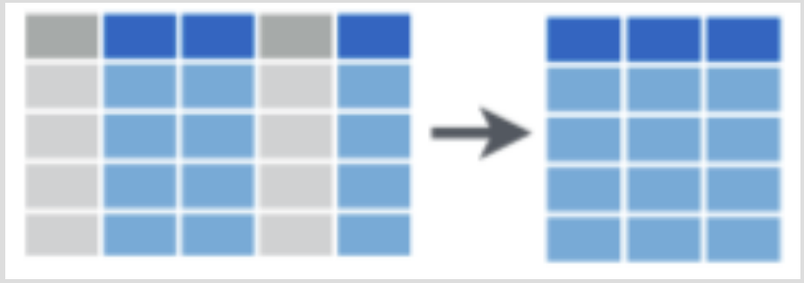
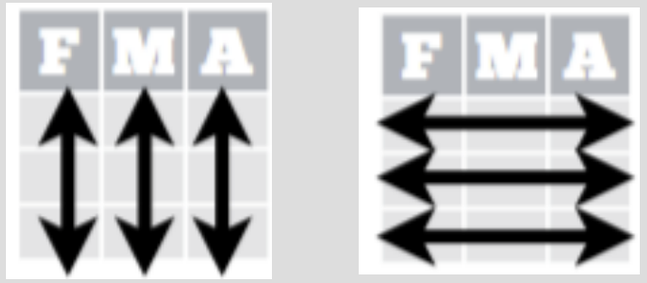
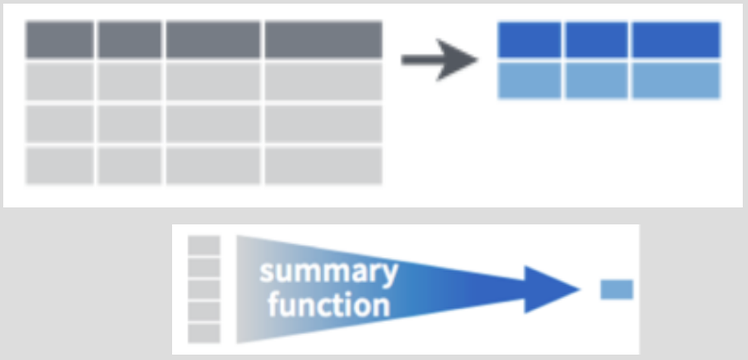


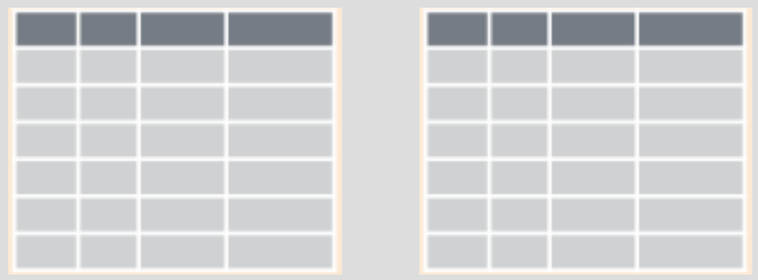
Tabelle normieren



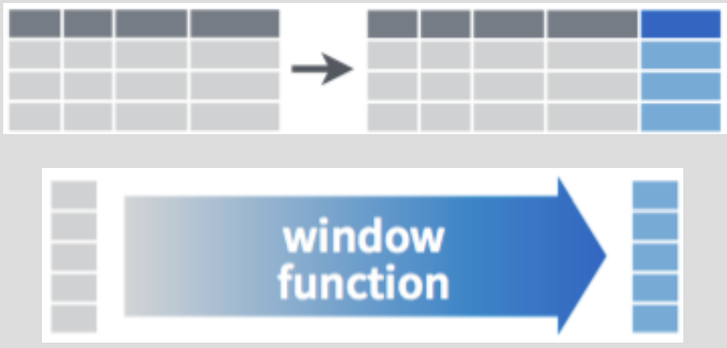
Zusammenfassen



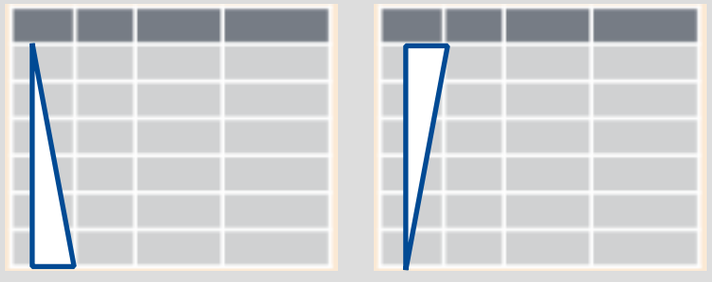
Gruppieren



Neue Spalte erstellen



Zeilen sortieren



Tabellen vereinen

sonstiges

…

Die Grammatik der Datenanalyse mit dplyr

- Zu den häufigsten Operationen in der praktischen Datenanalyse gehören:
- Zeilen filtern (z. B. nur Frauen)
- Zeilen sortieren (z. B. Studenten mit guten Noten in den oberen Zeilen)
- Spalten wählen (z. B. 100 weitere Produkte ausblenden)
- Spalten in eine Zahl zusammenfassen (z. B. Notenspiegel 1. Klausur)
- Datensätze nach Gruppen aufteilen (z. B. Analyse getrennt nach Standorten)
- Zeilen in eine Zahl zusammenfassen (z. B. Annas Durchschnittsnote)
- Werte aus einer Spalte verändern (z. B. Summe über alle Quartale/Items bilden).
- Das R-Paket dplyr bietet diese Operationen auf einfache aber flexible Weise.
- Laden Sie dplyr und werfen Sie einen Blick ("to glimpse") in den mtcars-Datensatz mit glimpse(mtcars).

Zeilen filtern – `filter`

Sinnbild

ID	Name	Note1
1	Anna	1
2	Anna	1
3	Berta	2
4	Carla	2
5	Carla	2

Filter mir die Anna!

ID	Name	Note1
1	Anna	1
2	Anna	1

Nur "Anna-Zeilen" sollen übrig bleiben.

Fallbeispiel

- Sie sind nur an den Noten eines bestimmten Studenten interessiert.
- Sie wollen die Umsätze nur eines Standorts berechnen.
- Sie wollen einen Analyse nur für die Männer Ihres Datensatzes erstellen.

Beschreibung

- Mit `filter` filtert man Zeilen (Beobachtungen/ Fälle).
- Alle Zeilen, die das Kriterium erfüllen, bleiben im Datensatz (werden gezeigt), die anderen werden (temporär) entfernt.
- Man kann nach mehreren Bedingungen filtern.

Syntax

- `mein_df <- filter(Daten, Kriterium)`
- `filter(tips, sex == "Female")`
- `filter(Affair, affairs > 0)`
- `filter(Affair, gender == "male", age > 35)  # Männer über 35`
- `filter(Datensatz, a == 1 | a == 2)  # Zeilen, in denen a=1 ist oder 2 ist`

Zeilen sortieren – `arrange`

Sinnbild

ID	Name	Note1
1	Anna	1
2	Anna	5
3	Berta	2
4	Carla	4
5	Carla	3

Gute Noten zuerst!

ID	Name	Note1
1	Anna	**1**
3	Berta	**2**
5	Carla	**3**
4	Carla	**4**
2	Anna	**5**

Fallbeispiel

- Sie wollen die Verkäufer mit den höchsten Umsätzen sehen.
- Sie wollen die Schüler mit den schlechtesten Noten kennen.
- Sie wollen die ersten Tage des Jahres ganz oben in der Tabelle stehen haben.

Beschreibung

- Mit `arrange` sortiert man Zeilen aufsteigend (oder absteigend).
- Man kann nach mehreren Kriterien sortieren; dann wird zuerst nach Kriterium 1 sortiert und dann jeder Wert von Kriterium 1 nach Kriterium 2
- Man kann auch alphabetisch sortieren.

Syntax

- `arrange(datensatz, Kriterium1)`
- `arrange(mtcars, hp)`
- `arrange(mtcars, -hp) #absteigend`
- `arrange(mtcars, cyl, am)`
  `# erst nach cyl, innerhalb jeder Gruppe von cyl nach am`

Spalten wählen – `select`

Sinnbild

ID	Name	N1	*N2*	*N3*
1	Anna	1	*2*	*3*
2	Berta	1	*1*	*1*
3	Carla	2	*3*	*4*
…	…	…	…	…

Nur ein paar Spalten interessieren mich!

ID	Name	N1
1	Anna	1
2	Berta	1
3	Carla	2
…	…	…

Fallbeispiel

- Ihr Datensatz hat 100 Spalten (für 100 Items), das ist unübersichtlich. Sie sind an den Etraversion-Items interessiert, die in Spalten 12, 13, 27 und 81 stehen. Andere Spalten sollen nicht gezeigt werden.

Beschreibung

- Mit `select` wählt man Spalten aus; nicht gewählte werden (temporär) gelöscht.
- Man kann mehrere Spalten auf einmal wählen.
- Man kann Spalten auf vielerlei Arten wählen (Details, mehr Details).

Syntax

- `mein_df <- select(daten, Spalte1)`
- `select(mtcars, hp, mpg)`
- `select(mtcars,-hp) #alle ohne hp`
- `select(mtcars, 1:3) # Spalten 1-3`
- `select(mtcars, mpg:disp) # dito`
- `select(extra, contains("i"))`

Spalten in eine Zahl zusammenfassen – `summarise`

Sinnbild

Note1
1
1
2
…

Will Mittelwert wissen!
(Spalte wird zu Zahl)

MW
2,4

Fallbeispiel

- Sie wollen die mittlere Anzahl an Affären wissen.
- Sie wollen wissen, was der größte Trinkgeldwert war.
- Sie wollen die Streuung der Umsätze ermitteln.

Beschreibung

- Mit `summarise` wird eine Spalte zu einer Zahl zusammengefasst.
- Die genaue Funktion der Zusammenfassung ist frei (MW, Md, min, max, SD, IQR, n, …)
- Jede Funktion, die aus einer Spalte *eine* Zahl macht, ist erlaubt.

Syntax

- `summarise(datensatz, Funktion1)`
- `summarise(mtcars, mean(hp))`
- `summarise(mtcars, MW_hp = mean(hp)) #Ergebnis kriegt Namen`
- ```
  summarise(Affair,
    biggest_halodrie = max(affair),
    halodrie_sd = sd(affair))
  ```
  ```

Nach Gruppen aufteilen – `group_by`

Sinnbild

ID	Name	Note	Fach
1	Anna	1	A
2	Berta	1	A
3	Carla	2	B
…	…	…	…

Noten nach Fächern aufteilen!

ID	Name	Note	Fach
1	Anna	1	**A**
2	Berta	1	**A**
…	…	…	**…**

ID	Name	Note	Fach
1	Anna	1	**B**
2	Berta	1	**B**
…	…	…	**…**

…

Fallbeispiel

- Sie möchten Noten von Studierenden nach Fächern vergleichen.
- Sie möchten Umsätze nach Produkten (3) und nach Standort (4) vergleichen (12 Gruppen).

Beschreibung

- Mit `group_by` teilt man den Datensatz in Untergruppen auf.
- *Folgende* Analysen werden dann automatisch jeweils pro Untergruppe getrennt durchgeführt.
- Z. B. würde der Befehl `mean` dann für jede der Gruppen durchgeführt.

Syntax

- `neuer_df <- group_by(daten, Gruppierung)`
- `neuer_df <- group_by(mtcars, am)`
- `group_by(mtcars, am, cyl)`
- `df_group <- group_by(mtcars, cyl)`
-
```
summarise(df_group,
mw_group = mean(hp))
```

Die Pfeife

```
fasse_zusammen(analysiere(bereite_auf(lade(meine_daten.csv))))
```

Introducing the "pipe"

%>%

```
meine_daten %>%
  lade %>%
  bereite_auf %>%
  analysiere %>%
  fasse_zusammen
```

Befehle verknüpfen mit der "Pfeife": %>%

Viele Analysen bestehen aus mehreren Schritten; z. B.

```
Affair2 <- select(Affair, affairs, sex, rating)
Affair3 <- filter(Affair2, affairs != 0)
Affair4 <- group_by(Affair3, gender)
Affair5 <- summarise(Affair4, affairs_MW = mean(affairs))
```

Nachteil ist, dass viele "Zwischenlager" (`Affair2`, `Affair3`,...) entstehen. Eine Alternative wäre, die Befehle ineinander zu verschachteln, was aber leicht unübersichtlich wird. In vielen Situationen ist es von Vorteil, die Befehle "hintereinander zu schalten":

Errisch mit dplyr:	Deutsche Übersetzung
`Affair %>%`	Nimm den Datensatz "Affair" UND DANN
`  select(affairs, gender, rating) %>%`	wähle diese Spalten: gender, rating UND DANN
`  filter(affairs != 0) %>%`	filtere die Zeilen, in denen "affairs" nicht 0 ist UND DANN
`  group_by(gender) %>%`	gruppiere nach Geschlecht UND DANN
`  summarise(affairs_mw = mean(affairs))`	fasse nach dem Mittelwert von "affairs" zusammen

Der R-Befehl "**%>%**"* (die "Pfeife"/ engl. "pipe") lässt sich übersetzen als "UND DANN".

*Tastaturkürzel für "%>%" bei RStudio: Cmd+Shift+M