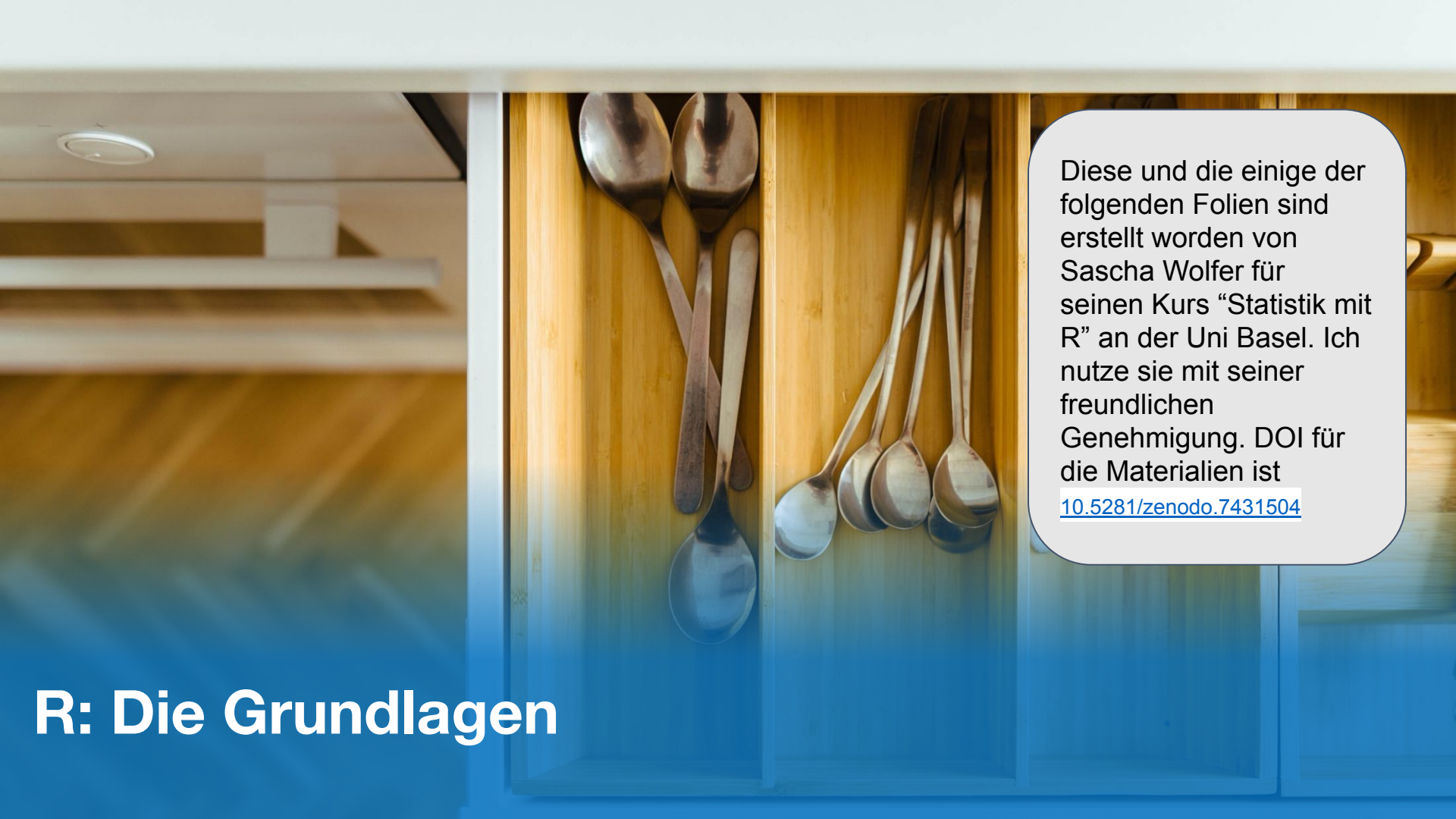
Einführung in die Statistik

Praktische Übung – Jürgen Hermes – IDH – SoSe 2023

R – Die Grundlagen

- R und RStudio-Setup
- Einfache Datentypen in R
- **Komplexe Datentypen in R**
- **Indizierung**
- Funktionen und Prädikate
- Zusatzpakete
- Daten einlesen und speichern
- Daten visualisieren



R: Die Grundlagen

Diese und die einige der folgenden Folien sind erstellt worden von Sascha Wolfer für seinen Kurs “Statistik mit R” an der Uni Basel. Ich nutze sie mit seiner freundlichen Genehmigung. DOI für die Materialien ist 10.5281/zenodo.7431504

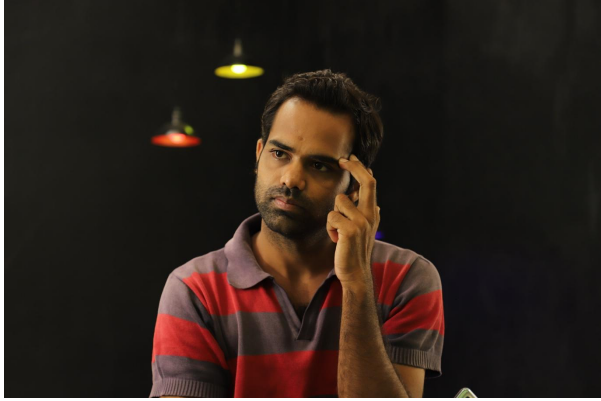
Komplexe Datentypen

- **Vektoren**
- **Dataframes**
- **Matrizen**
- **Listen**

Vektoren

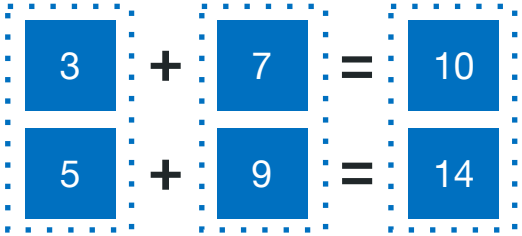
- Vektoren sind Reihen von Elementen **eines** Typs.
  - Numerische Vektoren
  - Character-Vektoren
  - Logische Vektoren
  - + Faktor-Vektoren
- Vektoren sind **geordnet**, die Reihenfolge ist wichtig.

Vektoren



- Erstellen Sie Vektoren mit `c()` – *construct*
  - `c("a", "b", 3)` → `[1] "a" "b" "3"`
- Andere Funktionen/Operatoren geben ebenfalls Vektoren zurück.
  - `5:9` → `[1] 5 6 7 8 9`
  - `seq(5, 8, 0.5)` → `[1] 5.0 5.5 6.0 6.5 7.0 7.5 8.0`
  - `rep(T, 5)` → `[1] TRUE TRUE TRUE TRUE TRUE`
- Sie können Vektoren miteinander verrechnen.
  - `c(3, 5) + c(7, 9)` → `[1] 10 14`

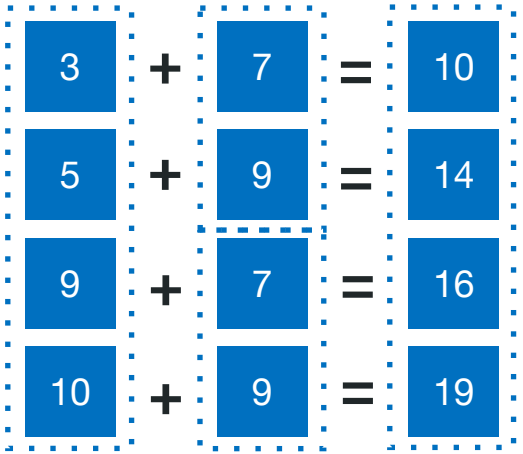
Recycling



`c(3, 5) + c(7, 9)`

Vektoren werden – falls nötig – *recyclet*. Sie werden also so lange wiederholt, wie es nötig ist.

`c(3, 5, 9, 10) + c(7, 9)`



Recycling: Achtung!

Das ist keine Fehlermeldung!

```
> c(3,5,9,10) + c(7,9,10)
[1] 10 14 19 17
Warnmeldung:
In c(3, 5, 9, 10) + c(7, 9, 10) : Länge des längeren Objektes
        ist kein Vielfaches der Länge des kürzeren Objektes
```

- R möchte Vektoren gerne komplett recyclen.
  - Hier kann der kürzere Vektor aber nur zum Teil recyclet werden.
- Wenn das geschieht, wird eine Warnmeldung ausgegeben.
  - Bei **Warn**meldungen wird trotzdem ein Ergebnis zurückgegeben, bei **Fehler**meldungen nicht.

Faktor-Vektoren

Um zu verstehen, was Faktor-Vektoren sind und wozu sie da sind, benötigen wir Kenntnisse über **Skalenniveaus**.

Skalenniveaus

Verhältnisskala

Intervallskala

Metrische Skala

Ordinalskala

Nominalskala

Nominalskala

- Notiert werden die **Gleich- und Verschiedenheit** von Elementen.
  - Keine Rangfolge
  - Keine Abstände
- Möglich: Auszählen von Häufigkeiten
- Beispiele: Wort-Type, Erstsprache, Korpusquelle
- Sonderfall: **Binäre** Skala
  (richtig/falsch, vorhanden/nicht vorhanden)

Ordinalskala

- **Zusätzlich** ablesbar: Rangfolge von Elementen
  - Keine Abstände
- Wir können zusätzlich alle Analysen/Maßzahlen anwenden, die auf der Rangfolge beruhen (z.B. **Median**).
- Beispiele: Ränge beim Militär (General > Oberst > Feldwebel > Gefreiter),
  Beurteilungen (sehr unzufrieden < unzufrieden < zufrieden < sehr z.)
  Schulnoten (sehr gut > gut > ... > ungenügend)

Intervallskala

- **Zusätzlich** können wir ablesen, wie weit Datenpunkte auf der Skala voneinander entfernt sind.
  - Kein absoluter Nullpunkt ➔ keine Verhältnisse
- Wir können alle relevanten statistischen Verfahren anwenden.
- Beispiele: IQ, Temperatur in Celsius

Verhältnisskala

- Einziger Unterschied zur Intervallskala: Absoluter Nullpunkt
- **Zusätzlich** können wir Aussagen über Verhältnisse machen.
  - "X ist doppelt so schwer wie Y."
- Unterschied in der Praxis (fast) nie relevant, daher Zusammenfassung von Intervall- und Verhältnisskala als **metrische Skala**
- Beispiel: Gewicht, Reaktionszeit, Temperatur in Kelvin

Nominalskala / Ordinalskala / Metrische Skala?

- Länge des Weges von der Haustür zur Uni
- Sprachkenntnisse (Niveau des europäischen Referenzrahmens)
- Studienfachkombination im BA
- Gewicht des Rucksacks, den Sie mit sich tragen
- Zeit, die für das Lösen einer Programmieraufgabe benötigt wird
- Zufriedenheit mit einem bestimmten Seminar
- Anteil des Studiums an der wöchentlichen Lebenszeit

Jetzt aber: Faktor-Vektoren

```
> factor(c("A", "A", "B", "C", "C", "C"))
[1] A A B C C C
Levels: A B C
```

- Faktorvektoren werden in R dazu verwendet, **nominal-** und **ordinalskalierte** Daten zu notieren.
- Ordinal: Sonderfall *ordered factors* – erstellen mit `ordered(x)`
  - *Ordered factors* braucht man sehr selten.
- *Levels*: Alle Stufen eines Faktors

Übung E03U03 - Faktor-Vektoren: Achtung!

- Weisen Sie einer Variable den folgenden numerischen Vektor zu:
  - `[42, -1, 8, 99]`
- Wandeln Sie die Variable in einen Faktor-Vektor um. Sie benötigen dazu die Funktion `as.factor()`. Geben Sie die Variable aus.
- Wandeln Sie die Variable mit `as.numeric()` wieder zurück in einen numerischen Vektor um.
- Geben Sie die Variable wieder aus. Was fällt Ihnen auf?

Begriffe

Vektoren

Faktor-Vektoren

Recycling

Warnmeldung

Fehlermeldung

Skalenniveau

Nominalskala

Ordinalskala

Metrische Skala

Binäre Skala

Vektoren: Zusammenfassung

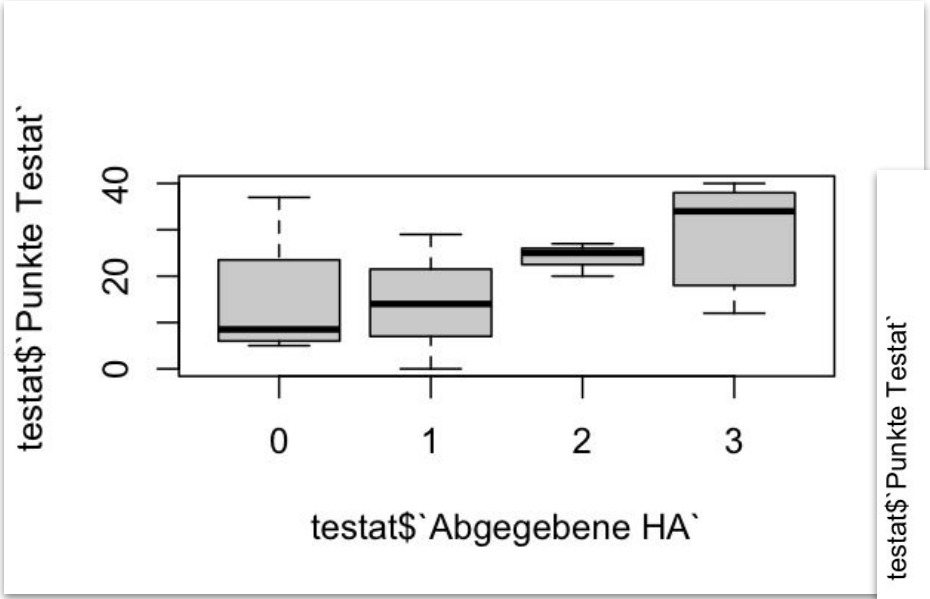
- Vektoren sind ein komplexer Datentyp.
- Vektoren sind geordnete Reihen **eines** elementaren Datentyps.
- Vektoren werden Element für Element miteinander verrechnet.
  - u.U. werden Vektoren dabei recyclet.
  - Nebenbei: Wir müssen unterscheiden zwischen **Fehlermeldungen** und **Warnmeldungen**.
- Faktor-Vektoren kodieren nominal- und ordinalskalierte Variablen und haben Stufen (*levels*).

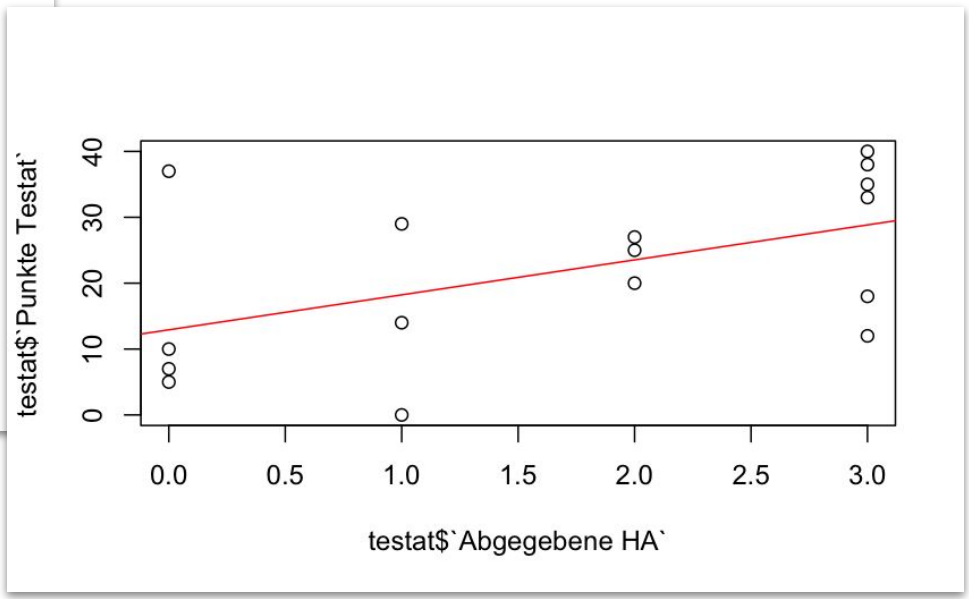
Fragen?

Hausaufgabe 1 (schon seit letzter Woche online)

- Die beste Übung, sich auf das Zwischentestat vorzubereiten, ist die Bearbeitung der Hausaufgabe (Stat_HA_1.R) in ILIAS. Die Aufgaben sind dem R-Kurs vom HPI entnommen. Dies habe ich aus dem Grund gemacht, als dass Sie so auch die dortige Umgebung nutzen können, um Ihre Lösungen zu überprüfen (CodeOcean).
- Die Abgabe wird nicht kontrolliert, Sie sind aber herzlich eingeladen, ihr R-Script auch in ILIAS einzureichen.
- Die Studienleistung wird vergeben aufgrund der Ergebnisse in den Testaten. Sollten Sie *eines* der Testate entschuldigt (Attest!) versäumen, *könnte* ggfs. die regelmäßige, rechtzeitige, vollständige und korrekte Abgabe der HA in die Bewertung mit einbezogen werden statt ein Nachtestat anzusetzen. [Hinweis: Das ist kein Automatismus!]

Hausaufgaben und Testat-Ergebnisse





Dataframes

Dataframes kann man als nebeneinander gestapelte Vektoren begreifen.

- Datentabellen: Ein Fall pro Zeile
  - Fall: Versuchsdurchlauf, Wort, Studierender, ...
- In den Spalten stehen Informationen zu jedem Fall.

leicht modifizierte Version (Jürgen Hermes)

Dataframes

- Dataframes sind für alle wichtigen Operationen der Datenanalyse der Ausgangspunkt.
  - Gruppenvergleiche
  - Zusammenhänge
  - Aggregationen
  - Visualisieren
  - ...
- Man kann Dataframes mit `data.frame()` erstellen, meist sind sie aber Ergebnisse von **Einleseoperationen**.
- Mit `head(<dataframe>)` werden die ersten 6 Zeilen eines Dataframes ausgegeben.

Matrizen

- Matrizen (Singular: Matrix) sind **mehrdimensionale Vektoren**.
  - Auch eine Matrix kann *nur einen Datentyp* enthalten!
- Zweidimensionale Matrizen haben auch Zeilen und Spalten.
  - Matrizen können aber beliebig viele Dimensionen haben.

```
> matrix(1:15, ncol = 3)
     [,1] [,2] [,3]
[1,]    1    6   11
[2,]    2    7   12
[3,]    3    8   13
[4,]    4    9   14
[5,]    5   10   15
```

```
> matrix(1:15, ncol = 4)
```

Wird ausgeführt, wirft aber eine Warnung, weil 15 nicht durch 4 teilbar ist.

Listen

- Listen sind geordnete Sequenzen von beliebigen Datentypen.
  - Im Gegensatz zu Vektoren können Listen auch komplexe Datentypen enthalten. Listen können also auch wiederum Listen enthalten.

```
> list(5:10, data.frame(A = 1:3, B = c("a", "b", "c")), rep(T,
4))
[[1]]
[1]  5  6  7  8  9 10

[[2]]
  A B
1 1 a
2 2 b
3 3 c

[[3]]
[1] TRUE TRUE TRUE TRUE
```

1. Element: Numerischer Vektor

2. Element: Zweispaltiger Dataframe

3. Element: Vektor mit Wahrheitswerten

Indizierung

Indizierung

- Indizierung = Zugriff; wir greifen auf komplexe Datentypen zu, um Teile von ihnen zu extrahieren oder zu verändern.
- Vektoren: `[n]`
  - `vec[2]`: Zweites Element des Vektors `vec`
  - `vec[3:5]`: Drittes bis fünftes Element von `vec`
- Dataframes: `[<Zeile n>, <Spalte m>]` oder `$<Spaltenname>`
  - `df[5,]`: Fünfte Zeile von Dataframe `df`
  - `df[, 4]`: Vierte Spalte von Dataframe `df`
  - `df[2, 3]`: Element in der zweiten Zeile in der dritten Spalte
  - `df$wort`: Komplette Spalte `wort` (= Vektor)

Indizierung

- Wir können auch über **Namen** zugreifen (wie in `df$wort`):
  - `df[4, "freq"]`: Der vierte Eintrag in der Spalte `freq`
- Wir können außerdem über **Wahrheitswerte** zugreifen (das wird später nochmal wichtig!):

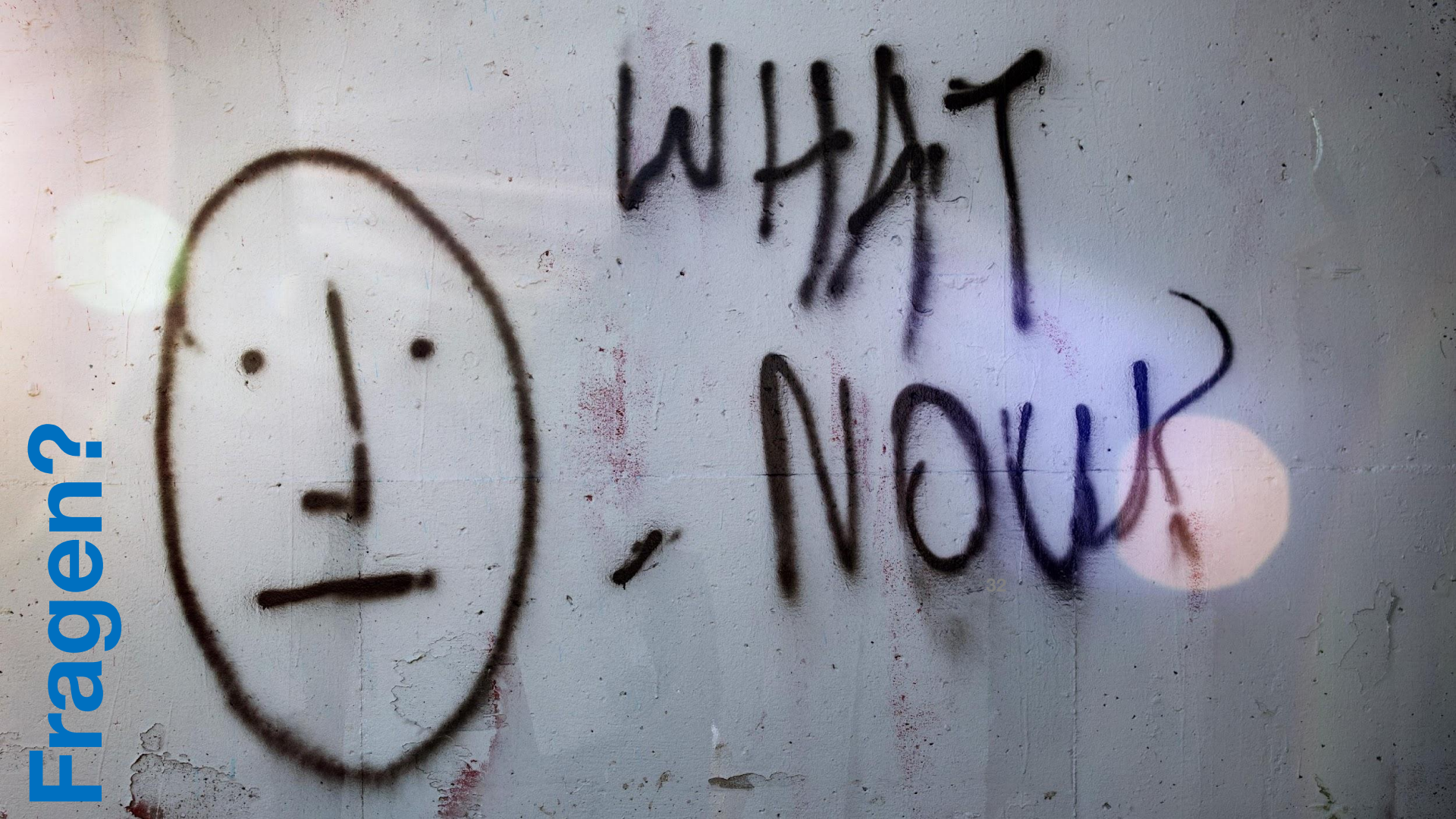
```
> vec <- c("das", "ist", "ein", "vektor")
> vec[c(T, T, F, T)]
[1] "das"    "ist"    "vektor"
```

Sortieren = Indizierung

- `order(<Vektor>)` gibt die Rangfolge der Elemente in einem Vektor zurück.
  - Bei Zeichenketten: alphabetisch, bei Zahlen: von klein nach groß
  - `order(c("D", "B", "C", "A"))` → `[1] 4 2 3 1`
  - `order(c("B", "C", "A"))` → `[1] 3 1 2`
  - Zurückgegeben wird ein Vector, der die Reihenfolge der Indizes enthält!
- Deshalb können wir Dataframes sortiert ausgeben, wenn wir `order()` verwenden, um auf Zeilen zuzugreifen.
  - `df` aufsteigend nach Spalte `freq` sortiert ausgeben: `df[order(df$freq),]`
  - Absteigend sortieren: `order(..., decreasing = T)`

Zusammenfassung

- **Vektoren**, **Dataframes**, Listen und Matrizen sind die wichtigsten komplexen Datentypen.
  - Dataframes kann man sich vorstellen als nebeneinander gestellte Vektoren.
  - Matrizen sind mehrdimensionale Vektoren.
  - Listen können alle anderen Datentypen enthalten, auch Listen selbst.
- Wir können auf diese zugreifen (sie indizieren) über
  - Zahlen (= Stellen/Indizes): `vec[3]` oder `df[3,]`
  - Namen: `df$wort` oder `df[5, "wort"]`
  - Wahrheitswerte: `vec[c(T, F)]`



Fragen?

Übung

- Erstellen Sie fünf Vektoren:
  - Vektor `user` mit den Werten `["km", "smv", "sw", "al"]`
  - Vektor `posts` `[18948, 11314, 2440, 14610]`
  - Vektor `followers` `[3584, 3609, 719, 2543]`
  - Vektor `follows` `[1374, 548, 877, 1059]`
  - Vektor `face.in.profile` `[T, F, F, T]`
- Kombinieren Sie die Vektoren in einem Dataframe `mastodonData`.
- Extrahieren Sie die erste Zeile.
- Extrahieren Sie die zweite und dritte Spalte.
- Extrahieren Sie den Wert `877`.
- Extrahieren Sie die User, die ein Bild von sich im Profil anzeigen.
- Sortieren Sie den Dataframe nach der Spalte `followers`.

Hausaufgabe 2

- Die beste Übung, sich auf das Zwischentestat vorzubereiten, ist die Bearbeitung der Hausaufgabe (Stat_HA_2.R) in ILIAS. Die Aufgaben sind dieses Mal nur zum Teil dem R-Kurs vom HPI entnommen.
- Die Abgabe wird nicht kontrolliert, Sie sind aber herzlich eingeladen, ihr R-Script auch in ILIAS einzureichen.
- Die Studienleistung wird vergeben aufgrund der Ergebnisse in den Testaten. Sollten Sie *eines* der Testate entschuldigt (Attest!) versäumen, *könnte* ggfs. die regelmäßige, rechtzeitige, vollständige und korrekte Abgabe der HA in die Bewertung mit einbezogen werden statt ein Nachtestat anzusetzen. [Hinweis: Das ist kein Automatismus!]