

Evaluation Randomisiert-Kontrollierter Studien und Experimente mit R

Eyeballing und Manipulation der Studiendaten

Prof. Dr. David Ebert & Mathias Harrer

Graduiertenseminar TUM-FGZ

Psychology & Digital Mental Health Care, Technische Universität München

Eyeballing

Nach dem Import der Daten ist es sinnvoll, sich einen **Überblick über die Daten** zu verschaffen.

→ Eine Möglichkeit dazu ist die Funktion `glimpse` aus dem package `tidyverse`:

```
library(tidyverse)
glimpse(data)
## Rows: 264
## Columns: 34
## $ id          <chr> "stress_gui_002", "stress_gui_14~
## $ group       <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,~
## $ sex         <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
## ...
```

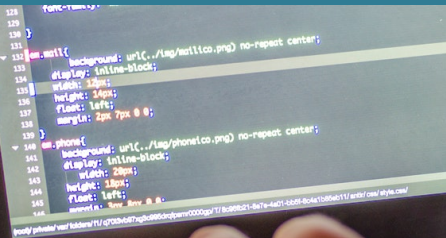
(Harrer et al., 2021, Kap. 2.5)

Vorteile von glimpse

- ✓ Anzahl der Beobachtungen/ Personen und Spalten/ Variablen dargestellt
- ✓ Übersicht über inkludierte Variablen
- ✓ Übersicht über zugeordnete Objektklassen
- ✓ Einblick in die Ausprägungen der ersten Personen im Datensatz auf den einzelnen

Praxis-Teil:

Eyeballing der Studiendaten



Manipulation von Studiendaten

***Data Wrangling* bezeichnet den essentiellen Prozess, in dem Daten für weitere Analysen nutzbar gemacht werden**

- Data Scientists verbringen z.B. wahrscheinlich die meiste Zeit damit, raw “untidy” data in “tidy” data umzuwandeln.
- Exzellente Werkzeuge dafür bietet das package tidyverse

(Harrer et al., 2021, Kap. 2.5)

Beim Datenimport werden Variablen teilweise falsch klassifiziert. Um die Daten nutzbar zu machen, müssen die Variablen zunächst den richtigen Klassen zugeordnet werden

Dazu eignen sich die Funktionen `as.numeric()`, `as.character()`, `as.factor()` und `as.logical()` aus dem R base package:


```
data$group <- as.factor(data$group)
class(data$group)
## [1] "factor"
```

⇒ Hierbei wird die betreffende Variable in die Funktion eingespeist und der output durch den Zuweisungsoperator `<-` wieder in der originalen Variable gespeichert.

(Harrer et al., 2021, Kap. 2.5)

Praxis-Teil:

Korrekte Klassenzuordnung



```
128 }
129
130 }
131
132 .mail{
133     background: url(../img/mailico.png) no-repeat center;
134     display: inline-block;
135     width: 120px;
136     height: 140px;
137     float: left;
138     margin: 2px 7px 0 0;
139 }
140 .phone{
141     background: url(../img/phoneico.png) no-repeat center;
142     display: inline-block;
143     width: 280px;
144     height: 120px;
145     float: left;
146     margin: 2px 7px 0 0;
147 }
```

Es gibt mehrere Wege, um in R Daten zu extrahieren. Z.B.:

1. Mithilfe des Dollarzeichen-Operators (schon besprochen)
2. Über eckige Klammern []
3. Über die Funktion `filter()` aus dem package `tidyverse`

(Harrer et al., 2021, Kap. 2.5)

2. Über eckige Klammern

- Generelle Form: `data.frame[rows, columns]`
- Als Argumente können sowohl die Nummer der Reihe oder Spalte als auch der Name der Reihe oder Spalte genutzt werden

```
data[2,3]
## [1] 0
data[2, "group"]
## [1] 0
```

→ Wird das Argument vor oder nach dem Komma leer gelassen, werden ganze Zeilen oder Spalten extrahiert werden.

(Harrer et al., 2021, Kap. 2.5)

2. Über eckige Klammern

concatenate bzw. c() Funktion

Diese Funktion bindet zwei oder mehr Worte oder Zahlen zusammen.

Bsp.: `c(1,5,3)`; `c("blau", "grün")`

Kombination aus `c()` Funktion und eckigen Klammern:

```
data[,c("pss.0", "pss.1", "pss.2")]
```

```
##      pss.0 pss.1 pss.2
## 1      25     15     21
## 2      22     18     24
## 3      25     NA     22
## ...
```

(Harrer et al., 2021, Kap. 2.5)

3. Über die Funktion `filter()`

- Generelle Form: `filter(data, Filterlogik)`
- Auch hierüber können Daten nach Variablennamen gefiltert werden
- Besonders an dieser Funktion ist allerdings, dass **Daten einfach basierend auf Zeilenwerten extrahiert** werden können

```
filter(data, age >= 40)
```

→ Dieser Befehl extrahiert z.B. einen Datensatz, in dem alle Personen mindestens 40 Jahre alt sind. Auch alle anderen Vergleichsoperatoren wie `<`; `==` etc. können hier angewandt werden.

(Harrer et al., 2021, Kap. 2.5)

Pipes haben zwei große Vorteile:

- Funktionen können auf ein Objekt angewandt werden, **ohne dass das Objekt in den Funktionen jeweils nochmal benannt werden muss**
- Mit pipes können **mehrere Funktionen aneinandergekettet** werden


```
data %>%  
  filter(age < 41) %>%  
  pull(pss.0) %>%  
  mean()  
## [1] 25.73469
```

pull Funktion

Die pull Funktion ist ein Äquivalent zum Dollarzeichen-Operator, die aber in Pipes genutzt werden kann. Die Funktion "zieht" eine Variable aus dem Datensatz und gibt sie weiter an die nächste Funktion.

(Harrer et al., 2021, Kap. 2.5)

Praxis-Teil:



```
128 }
129
130 }
131
132 .mail{
133     background: url(../img/mailico.png) no-repeat center;
134     display: inline-block;
135     width: 120px;
136     height: 140px;
137     float: left;
138     margin: 2px 7px 0 0;
139 }
140
141 .phone{
142     background: url(../img/phoneico.png) no-repeat center;
143     display: inline-block;
144     width: 20px;
145     height: 130px;
146     float: left;
147     margin: 2px 7px 0 0;
148 }
```

Referenzen

Harrer, M., Cuijpers, P., A, F. T., & Ebert, D. D. (2021). *Doing meta-analysis with R: A hands-on guide* (1st ed.). Chapman & Hall/CRC Press.