



# Ein Übungsbuch für R-Einsteiger:innen und Fortgeschrittene

Prof. Dr. Jörg große Schlarmann

Version vom 27.06.2024

# Lizenz



Dieses Script ist unter der Creative Commons BY-NC-SA 4.0<sup>1</sup> lizenziert.

Sie dürfen:

- **Teilen** — das Material in jedwedem Format oder Medium vervielfältigen und weiterverbreiten.
- **Bearbeiten** — das Material remixen, verändern und darauf aufbauen.

Unter folgenden Bedingungen:

- **① Namensnennung** — Sie müssen angemessene Urheber- und Rechteangaben machen, einen Link zur Lizenz beifügen und angeben, ob Änderungen vorgenommen wurden. Diese Angaben dürfen in jeder angemessenen Art und Weise gemacht werden, allerdings nicht so, dass der Eindruck entsteht, der Lizenzgeber unterstütze gerade Sie oder Ihre Nutzung besonders.
- **⑤ Nicht kommerziell** — Sie dürfen das Material nicht für kommerzielle Zwecke nutzen.
- **③ Weitergabe unter gleichen Bedingungen** — Wenn Sie das Material remixen, verändern oder anderweitig direkt darauf aufbauen, dürfen Sie Ihre Beiträge nur unter derselben Lizenz wie das Original verbreiten.

**Keine weiteren Einschränkungen** — Sie dürfen keine zusätzlichen Klauseln oder technische Verfahren einsetzen, die anderen rechtlich irgendetwas untersagen, was die Lizenz erlaubt.

## 💡 Zitationsvorschlag

große Schlarmann, J (2024): “trainingslageR. Ein Übungsbuch für R-Einsteiger\*innen und Fortgeschrittene”, Hochschule Niederrhein, <https://www.produnis.de/R/trainingslager.html>

```
@book{grSchl_exeRueb,  
  author = {{große Schlarmann}, Jörg},  
  title = {{trainingslageR}. Ein Übungsbuch für R-Einsteiger*innen und Fortgeschrittene},  
  year = {2024},  
  publisher = {Hochschule Niederrhein},  
  address = {Krefeld},  
  copyright = {CC BY-NC-SA 4.0},  
  url = {https://www.produnis.de/R/trainingslager.html},  
  language = {de},  
}
```

<sup>1</sup> siehe <https://creativecommons.org/licenses/by-nc-sa/4.0/>

# Inhaltsverzeichnis

<b>Lizenz</b>	<b>i</b>
<b>Inhaltsverzeichnis</b>	<b>ii</b>
<b>Einleitung</b>	<b>1</b>
<b>I. Aufgaben</b>	<b>2</b>
<b>1. Aufgaben für Einsteiger:innen</b>	<b>3</b>
1.1. Objekte in R . . . . .	3
1.1.1. Aufgabe 1.1.1 Vektoren . . . . .	3
1.1.2. Aufgabe 1.1.2 Zufallsvektoren . . . . .	4
1.1.3. Aufgabe 1.1.3 Krankenhausaufenthalte . . . . .	4
1.1.4. Aufgabe 1.1.4 Größe und Gewicht . . . . .	5
1.1.5. Aufgabe 1.1.5 ordinale Faktoren . . . . .	5
1.1.6. Aufgabe 1.1.6 Hogwarts-Kurse . . . . .	6
1.1.7. Aufgabe 1.1.7 Datentabelle . . . . .	7
<b>2. Aufgaben für geübte User:innen</b>	<b>8</b>
2.1. Objekte in R . . . . .	8
2.1.1. Aufgabe 2.1.1 Hogwarts-Kurse . . . . .	8
<b>3. Aufgaben für fortgeschrittene Anwender:innen</b>	<b>9</b>
3.1. Objekte in R . . . . .	9
3.1.1. Aufgabe 3.1.1 Hogwarts-Kurse . . . . .	9
<b>II. Lösungswege</b>	<b>10</b>
<b>4. Lösungswege zu den Aufgaben für Einsteiger:innen</b>	<b>11</b>
4.1. Lösungen zu Objekten in R . . . . .	11
4.1.1. Lösung zur Aufgabe 1.1.1 Vektoren . . . . .	11
4.1.2. Lösung zur Aufgabe 1.1.2 Zufallsvektoren . . . . .	12
4.1.3. Lösung zur Aufgabe 1.1.3 Krankenhausaufenthalte . . . . .	13
4.1.4. Lösung zur Aufgabe 1.1.4 Größe und Gewicht . . . . .	15
4.1.5. Lösung zur Aufgabe 1.1.5 ordinale Faktoren . . . . .	17
4.1.6. Lösung zur Aufgabe 1.1.6 Hogwarts-Kurse . . . . .	19
4.1.7. Lösung zur Aufgabe 1.1.7 Datentabelle . . . . .	23
<b>5. Lösungswege zu den Aufgaben für geübte User:innen</b>	<b>27</b>
5.1. Lösungen zu Objekten in R . . . . .	27
5.1.1. Lösung zur Aufgabe 2.1.1 Hogwarts-Kurse . . . . .	27

<b>6. Lösungswege zu den Aufgaben für fortgeschrittene Anwender:innen</b>	<b>31</b>
6.1. Lösungen zu Objekten in R . . . . .	31
6.1.1. Lösung zur Aufgabe 3.1.1 Hogwarts-Kurse . . . . .	31
<b>Literaturverzeichnis</b>	<b>32</b>
<b>Credits</b>	<b>33</b>

# Einleitung

“You shouldn’t feel ashamed about your code - if it solves the problem, it’s perfect just the way it is. But also, it could always be better.” — **Hadley Wickham** at `rstudio::conf2019`

## Willkommen im trainingslageR!

In diesem Buch sind zahlreiche Übungen zur freien Statistiksoftware R enthalten. Für Ihre Lösungswege kann das freie Nachschlagewerk von große Schlarmann ([2024b](#)) hilfreich sein.

Lassen Sie sich nicht entmutigen, R hat eine steile Lernkurve.

Falls Sie nach diesen Übungen immer noch nicht genug haben, finden Sie weitere Aufgabenstellungen bei große Schlarmann ([2024a](#)).

**Teil I.**

**Aufgaben**

# 1. Aufgaben für Einsteiger:innen

Schön, dass Sie Ihre R-Fähigkeiten überprüfen möchten. Bleiben Sie am Ball, Sie schaffen das!


## 1.1. Objekte in R

In diesem Abschnitt üben Sie den Umgang mit R-Objekten wie Vektoren, Faktoren und Datenframes.

### 1.1.1. Aufgabe 1.1.1 Vektoren

**i**

- a) Erzeugen Sie mit möglichst wenig Aufwand einen Datenvektor aus den Zahlen 1 bis 100.
- b) Erzeugen Sie einen Datenvektor, der aus den Wörtern “Apfel”, “Birne” und “Postauto” besteht.
- c) Erzeugen Sie einen weiteren Datenvektor, in welchem die Wörter “Apfel”, “Birne” und “Postauto” 30 mal wiederholt werden.

 Schauen Sie sich die Hilfeseite zur Funktion `rep()` an, um Aufgabe c) besser lösen zu können

```
?rep()  
# oder  
help(rep)
```



Lösung siehe Abschnitt 4.1.1

### 1.1.2. Aufgabe 1.1.2 Zufallsvektoren



- Erzeugen Sie einen Datenvektor aus 200 zufälligen Zahlen zwischen 1 und 500, ohne dass eine Zahl doppelt vorkommt (sog. “ohne zurücklegen”).
- Erzeugen Sie einen weiteren Datenvektor mit ebenfalls 200 zufälligen Zahlen zwischen 1 und 500, wobei Zahlen nun doppelt vorkommen dürfen (sog. “mit zurücklegen”).



Schauen Sie sich die Hilfeseite zur Funktion `sample()` an, um die Aufgaben leichter lösen zu können.

```
?sample  
# oder  
help(sample)
```



Lösung siehe Abschnitt 4.1.2

### 1.1.3. Aufgabe 1.1.3 Krankenhausaufenthalte



Hundert zufällig ausgewählte Personen wurden befragt, wie oft sie im letzten Jahr im Krankenhaus stationär behandelt wurden. Die Antworten wurden wie folgt notiert:

```
1,0,0,3,1,5,1,2,2,0,1,0,5,2,1,0,1,0,0,4,0,1,1,3,0,  
1,1,1,3,1,0,1,4,2,0,3,1,1,7,2,0,2,1,3,0,0,0,0,6,1,  
1,2,1,0,1,0,3,0,1,3,0,5,2,1,0,2,4,0,1,1,3,0,1,2,1,  
1,1,1,2,2,0,3,0,1,0,1,0,0,0,5,0,4,1,2,2,7,1,3,1,5
```

- Überführen Sie die Daten in ein R-Objekt mit dem Namen `KHAufenthalte`.
- Entfernen Sie den ersten und den dritten Eintrag aus Ihrem R-Objekt.
- Fügen Sie die Werte 7 und 2 dem Objekt hinzu.
- Benennen Sie das Objekt in `hospital.stays` um.
- Unterteilen Sie die Krankenhausaufenthalte mit der `cut()`-Funktion in die Klassen
  - 0,
  - 1–2 und
  - mehr als 2 Aufenthalte.



Lösung siehe Abschnitt 4.1.3



### 1.1.4. Aufgabe 1.1.4 Größe und Gewicht


**i** Von 10 Personen wurden folgende Körpergrößen in Meter gemessen:

1,68	1,87	1,95	1,74	1,80
1,75	1,59	1,77	1,82	1,74

... sowie folgende Gewichte in Gramm:

78500	110100	97500	69200	82500
71500	81500	87200	75500	65500

- Überführen Sie die Daten in R-Objekte mit den Namen `Groesse` und `Gewicht`.
- Rechnen Sie das Gewicht um in Kilogramm, und speichern Sie Ihr Ergebnis in der Variable `Kilogramm`.
- Berechnen Sie den BMI ( $\text{kg/m}^2$ ) der Probanden und speichern Ihr Ergebnis in das Objekt `BMI` (Dabei könnten Ihnen die zuvor erstellten Variablen von Nutzen sein!).
- Fügen Sie die Objekte `Groesse`, `Gewicht` (aber in Kilogramm) und `BMI` zu einem Datenframe zusammen.
- Lassen Sie die Daten von Proband 4, 7 und 9 ausgeben.
- Lassen Sie die Daten der Probanden ausgeben, deren Gewicht größer ist als 80kg.

 Lösung siehe Abschnitt 4.1.4


### 1.1.5. Aufgabe 1.1.5 ordinale Faktoren

**i**

- Erstellen Sie die ordinale Variable `Monate`, in welcher die 12 ausgeschriebenen Monatsnamen in korrekter Levelreihenfolge enthalten sind.
- Erstellen Sie die ordinale Variable `Schulnoten`, in welcher die 6 ausgeschriebenen Schulnoten in korrekter Levelreihenfolge enthalten sind.
- Erzeugen Sie aus den folgenden Daten einen ordinalen Faktor mit korrekter Levelreihenfolge.

vielleicht, glaube nicht, nein, glaube nicht, ja, glaube schon, vielleicht, nein, glaube nicht, ja, ja, glaube schon, ja, ja, nein, glaube nicht, glaube schon, vielleicht, vielleicht, glaube nicht, vielleicht, glaube nicht, nein, glaube nicht, ja, glaube schon, vielleicht, nein, glaube nicht, ja, ja, glaube schon, ja, ja, nein, glaube nicht, glaube schon, vielleicht, vielleicht, glaube nicht

- Ändern Sie die Levelnamen in -2, -1, 0, 1, 2.


 Lösung siehe Abschnitt 4.1.5

### 1.1.6. Aufgabe 1.1.6 Hogwarts-Kurse

**i** In Hogwarts wurden jeweils die vier beliebtesten Kurse der Schüler pro Haus ermittelt.

Haus	Kurs
Gryffindor	Verteidigung gegen die dunklen Künste
Gryffindor	Zauberkunst
Gryffindor	Verwandlung
Gryffindor	Besenflugunterricht
Hufflepuff	Kräuterkunde
Hufflepuff	Pflege magischer Geschöpfe
Hufflepuff	Geschichte der Zauberei
Hufflepuff	Alte Runen
Ravenclaw	Arithmantik
Ravenclaw	Astronomie
Ravenclaw	Verwandlung
Ravenclaw	Verteidigung gegen die dunklen Künste
Slytherin	Zaubertränke
Slytherin	Zauberkunst
Slytherin	Dunkle Künste
Slytherin	Legilimentik

- Erstellen Sie das Datenframe `Kurse`, in welchem die Daten aus den Tabellenspalten `Haus` und `Kurs` enthalten sind.
- Wieviele Kurse haben es in die Auswahlliste geschafft?
- Erstellen Sie für jedes Haus ein eigenes Datenframe
- Wandeln Sie in jedem Haus-Datenframe die Variablen in Faktoren um.
- Fügen Sie die Haus-Datenframes zu einem einzigen Datenframe `Hogwarts` zusammen, in der Reihenfolge `Ravenclaw`, `Gryffindor`, `Syltherin` und `Hufflepuff`. Ändern Sie anschließend den Kurs *“Geschichte der Zauberei”* in *“Geisterkunde”* um.
- Sortieren Sie den Datensatz, so dass die Kurse in alphabetischer Reihenfolge angezeigt werden.
- Speichern Sie den so sortierten Datensatz in das Objekt `sorted`, und reparieren Sie die Zeilennummerierung von `sorted`.

 Lösung siehe Abschnitt 4.1.6

### 1.1.7. Aufgabe 1.1.7 Datentabelle

**i** Von 6 Probanden wurde der Cholesterolspiegel in mg/dl gemessen.

Name	Geschlecht	Gewicht	Größe	Cholesterol
Anna Tomie	W	85	179	182
Bud Zillus	M	115	173	232
Dieter Mietenplage	M	79	181	191
Hella Scheinwerfer	W	60	170	200
Inge Danken	W	57	158	148
Jason Zufall	M	96	174	249


- a) Übertragen Sie die Daten in das Datenframe chol.  
b) Erstellen Sie eine neue Variable Alter, die zwischen Name und Geschlecht liegt und folgende Daten beinhaltet:

Name	Alter
Anna Tomie	18
Bud Zillus	32
Dieter Mietenplage	24
Hella Scheinwerfer	35
Inge Danken	46
Jason Zufall	68

- c) Fügen Sie einen weiteren Fall mit folgenden Daten dem Datenframe hinzu

Name	Alter	Geschlecht	Gewicht	Größe	Cholesterol
Mitch Mackes	44	M	92	178	220

- d) Erzeugen Sie eine neue Variable BMI ( $BMI = \frac{kg}{m^2}$ ).  
e) Fügen Sie die Variable Adipositas hinzu, in welcher Sie die BMI-Werte wie folgt klassieren:
- weniger als 18,5 → Untergewicht
  - zwischen 18,5 und 24,5 → Normalgewicht
  - zwischen 24,5 und 30 → Übergewicht
  - größer als 30 → Adipositas
- f) Filtern Sie Ihren Datensatz, so dass Sie einen neuen Datensatz male erhalten, welcher nur die Daten der Männer beinhaltet.


 Lösung siehe Abschnitt 4.1.7

## 2. Aufgaben für geübte User:innen


### 2.1. Objekte in R

#### 2.1.1. Aufgabe 2.1.1 Hogwarts-Kurse

- i** In Hogwarts wurden jeweils die vier beliebtesten Kurse der Schüler pro Haus ermittelt. Die Ergebnisse liegen in 2 Tabellen vor.


 Tabelle 1

	Hufflepuff	Slytherin
	Kräuterkunde	Zaubertränke
Pflege magischer Geschöpfe		Zauberkunst
Geschichte der Zauberei		Dunkle Künste
	Alte Runen	Legilimentik

 Tabelle 2:

	Gryffindor	Ravenclaw
Verteidigung gegen die dunklen Künste		Arithmantik
	Zauberkunst	Astronomie
	Verwandlung	Verwandlung
Besenflugunterricht	Verteidigung gegen die dunklen Künste	

- Benutzen Sie die `tribble()`-Funktion, um die Daten in die Objekte `tab1` und `tab2` zu überführen.
- Fügen Sie `tab1` und `tab2` zu einem Objekt `Hogwarts` zusammen.
- Nutzen Sie die `mutate()`-Funktion, um die Datenklassen der Variablen anzupassen (Skalenniveau).
- Ändern Sie anschließend mit der `mutate()`-Funktion den Kurs “*Geschichte der Zauberei*” in “*Geisterkunde*” um.
- Die Daten liegen nicht im Tidy-Data-Format vor. Erzeugen Sie ein neues Objekt `Kurse` mit den Variablen `Haus` und `Kurs`.

 Lösung siehe Abschnitt 5.1.1

## 3. Aufgaben für fortgeschrittene Anwender:innen

### 3.1. Objekte in R

#### 3.1.1. Aufgabe 3.1.1 Hogwarts-Kurse

**i**

- a) Benutzen Sie die `tribble()`-Funktion, um die Daten in die Objekte `tab1` und `tab2` zu überführen.



Lösung siehe Abschnitt 6.1.1

**Teil II.**

**Lösungswege**

## 4. Lösungswege zu den Aufgaben für Einsteiger:innen

⚠ Gerade als Anfänger:in sollten Sie zumindest *versuchen*, die Aufgaben selbstständig zu lösen, bevor Sie sich die Lösungswege anschauen. Kopf hoch, Sie schaffen das!

### 4.1. Lösungen zu Objekten in R

#### 4.1.1. Lösung zur Aufgabe 1.1.1 Vektoren

💡 a) Erzeugen Sie mit möglichst wenig Aufwand einen Datenvektor aus den Zahlen 1 bis 100.

```
zahlen <- c(1:100)
#anschauen
zahlen
```

```
[1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18
[19] 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36
[37] 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54
[55] 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72
[73] 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90
[91] 91 92 93 94 95 96 97 98 99 100
```

💡 b) Erzeugen Sie einen Datenvektor, der aus den Wörtern “Apfel”, “Birne” und “Postauto” besteht.

```
worte <- c("Apfel", "Birne", "Postauto")
# anschauen
worte
```

```
[1] "Apfel"      "Birne"      "Postauto"
```

💡 c) Erzeugen Sie einen weiteren Datenvektor, in welchem die Wörter “Apfel”, “Birne” und “Postauto” 30 mal wiederholt werden.

```
# mit rep() 30mal "worte" wiederholen
worte30 <- rep(worte, 30)
# anschauen
worte30
```

```
[1] "Apfel"      "Birne"      "Postauto" "Apfel"      "Birne"      "Postauto"
```

```
[7] "Apfel"      "Birne"      "Postauto"   "Apfel"      "Birne"      "Postauto"
[13] "Apfel"      "Birne"      "Postauto"   "Apfel"      "Birne"      "Postauto"
[19] "Apfel"      "Birne"      "Postauto"   "Apfel"      "Birne"      "Postauto"
[25] "Apfel"      "Birne"      "Postauto"   "Apfel"      "Birne"      "Postauto"
[31] "Apfel"      "Birne"      "Postauto"   "Apfel"      "Birne"      "Postauto"
[37] "Apfel"      "Birne"      "Postauto"   "Apfel"      "Birne"      "Postauto"
[43] "Apfel"      "Birne"      "Postauto"   "Apfel"      "Birne"      "Postauto"
[49] "Apfel"      "Birne"      "Postauto"   "Apfel"      "Birne"      "Postauto"
[55] "Apfel"      "Birne"      "Postauto"   "Apfel"      "Birne"      "Postauto"
[61] "Apfel"      "Birne"      "Postauto"   "Apfel"      "Birne"      "Postauto"
[67] "Apfel"      "Birne"      "Postauto"   "Apfel"      "Birne"      "Postauto"
[73] "Apfel"      "Birne"      "Postauto"   "Apfel"      "Birne"      "Postauto"
[79] "Apfel"      "Birne"      "Postauto"   "Apfel"      "Birne"      "Postauto"
[85] "Apfel"      "Birne"      "Postauto"   "Apfel"      "Birne"      "Postauto"
```

#### 4.1.2. Lösung zur Aufgabe 1.1.2 Zufallsvektoren

💡 a) Erzeugen Sie einen Datenvektor aus 200 zufälligen Zahlen zwischen 1 und 500, ohne dass eine Zahl doppelt vorkommt (sog. "ohne zurücklegen").

```
sample(1:500, 200, replace = FALSE)
```

```
[1] 426 245 110 242 344 158 185 438 162 68 387 130 113 367 190 434 349 329
[19] 476 312 480 276 429 198 385 209 26 165 243 77 424 160 461 423 477 234
[37] 35 187 23 439 107 475 99 310 268 450 183 41 28 266 115 208 244 391
[55] 466 294 44 400 416 483 288 175 418 212 112 406 314 447 272 331 260 52
[73] 307 463 85 88 106 140 278 451 216 446 443 54 492 132 392 376 9 398
[91] 119 5 203 337 205 60 10 29 304 131 409 254 80 487 129 449 127 335
[109] 56 146 340 303 370 291 67 166 283 401 353 174 274 460 267 442 87 231
[127] 419 195 105 360 484 330 262 321 369 121 298 136 122 343 389 372 4 84
[145] 269 65 40 220 362 315 311 491 325 373 230 96 31 33 133 347 486 135
[163] 412 301 415 411 53 313 210 142 95 49 154 425 289 433 74 126 86 324
[181] 499 173 386 20 118 116 358 454 393 488 459 428 410 39 397 379 399 458
[199] 427 207
```

💡 b) Erzeugen Sie einen weiteren Datenvektor mit ebenfalls 200 zufälligen Zahlen zwischen 1 und 500, wobei Zahlen nun doppelt vorkommen dürfen (sog. "mit zurücklegen").

```
sample(1:500, 200, replace = TRUE)
```

```
[1] 255 389 433 404 417 435 172 55 58 100 380 195 301 451 444 250 368 328
[19] 27 174 54 318 411 23 313 81 17 13 476 149 68 460 190 237 66 110
[37] 271 385 408 83 338 346 168 120 200 178 12 259 215 485 290 141 70 320
[55] 208 259 54 48 158 19 256 133 282 13 128 108 410 494 42 396 390 130
[73] 153 43 100 187 26 437 346 207 65 474 413 381 8 241 139 272 346 11
[91] 48 102 359 47 306 316 180 209 61 468 104 233 376 465 160 406 405 255
```



```
[109] 250 138 190 413 419 85 186 18 317 156 162 232 161 329 226 489 364 473
[127] 69 421 319 108 329 400 254 98 108 16 376 287 496 467 491 298 183 253
[145] 116 341 24 336 247 315 368 197 222 459 482 303 57 72 294 250 143 444
[163] 458 467 13 208 79 336 298 293 33 214 58 412 107 138 9 373 156 209
[181] 268 146 269 321 355 464 388 132 488 465 395 348 78 128 24 335 232 480
[199] 438 435
```

#### 4.1.3. Lösung zur Aufgabe 1.1.3 Krankenhausaufenthalte

💡 a) Überführen Sie die Daten in ein R-Objekt mit dem Namen KHAufenthalte.

```
KHAufenthalte <- c(1,0,0,3,1,5,1,2,2,0,1,0,5,2,1,0,1,0,0,4,0,1,1,3,0,
1,1,1,3,1,0,1,4,2,0,3,1,1,7,2,0,2,1,3,0,0,0,0,6,1,
1,2,1,0,1,0,3,0,1,3,0,5,2,1,0,2,4,0,1,1,3,0,1,2,1,
1,1,1,2,2,0,3,0,1,0,1,0,0,0,5,0,4,1,2,2,7,1,3,1,5)

#anschauen
KHAufenthalte

[1] 1 0 0 3 1 5 1 2 2 0 1 0 5 2 1 0 1 0 0 4 0 1 1 3 0 1 1 1 3 1 0 1 4 2 0 3 1
[38] 1 7 2 0 2 1 3 0 0 0 0 6 1 1 2 1 0 1 0 3 0 1 3 0 5 2 1 0 2 4 0 1 1 3 0 1 2
[75] 1 1 1 1 2 2 0 3 0 1 0 1 0 0 0 5 0 4 1 2 2 7 1 3 1 5
```

💡 b) Entfernen Sie den ersten und den dritten Eintrag aus Ihrem R-Objekt.

```
# ersten und dritten Wert entfernen
KHAufenthalte <- KHAufenthalte[-c(1,3)]

#anschauen
KHAufenthalte

[1] 0 3 1 5 1 2 2 0 1 0 5 2 1 0 1 0 0 4 0 1 1 3 0 1 1 1 3 1 0 1 4 2 0 3 1 1 7 2
[39] 0 2 1 3 0 0 0 0 6 1 1 2 1 0 1 0 3 0 1 3 0 5 2 1 0 2 4 0 1 1 3 0 1 2 1 1 1 1
[77] 2 2 0 3 0 1 0 1 0 0 0 5 0 4 1 2 2 7 1 3 1 5
```

💡 c) Fügen Sie die Werte 7 und 2 dem Objekt hinzu.

```
# 7 und 2 hinzufügen
KHAufenthalte <- c(KHAufenthalte, 7, 2)

#anschauen
KHAufenthalte

[1] 0 3 1 5 1 2 2 0 1 0 5 2 1 0 1 0 0 4 0 1 1 3 0 1 1 1 3 1 0 1 4 2 0 3 1 1 7
[38] 2 0 2 1 3 0 0 0 0 6 1 1 2 1 0 1 0 3 0 1 3 0 5 2 1 0 2 4 0 1 1 3 0 1 2 1 1
[75] 1 1 2 2 0 3 0 1 0 1 0 0 0 5 0 4 1 2 2 7 1 3 1 5 7 2
```

💡 d) Benennen Sie das Objekt in `hospital.stays` um.

```
# umbenennen
hospital.stays <- KHAufenthalte
```

💡 e) Klassieren Sie mit der `cut()`-Funktion in die Klassen 0, 1 – 2 und > 2 Aufenthalte.

```
# cut
cut(hospital.stays, breaks=c(0,1,3,Inf), right=FALSE)
```

```
[1] [0,1) [3,Inf) [1,3) [3,Inf) [1,3) [1,3) [1,3) [1,3) [0,1) [1,3)
[10] [0,1) [3,Inf) [1,3) [1,3) [0,1) [1,3) [0,1) [0,1) [3,Inf)
[19] [0,1) [1,3) [1,3) [3,Inf) [0,1) [1,3) [1,3) [1,3) [3,Inf)
[28] [1,3) [0,1) [1,3) [3,Inf) [1,3) [0,1) [3,Inf) [1,3) [1,3)
[37] [3,Inf) [1,3) [0,1) [1,3) [1,3) [3,Inf) [0,1) [0,1) [0,1)
[46] [0,1) [3,Inf) [1,3) [1,3) [1,3) [1,3) [0,1) [1,3) [0,1)
[55] [3,Inf) [0,1) [1,3) [3,Inf) [0,1) [3,Inf) [1,3) [1,3) [0,1)
[64] [1,3) [3,Inf) [0,1) [1,3) [1,3) [3,Inf) [0,1) [1,3) [1,3)
[73] [1,3) [1,3) [1,3) [1,3) [1,3) [1,3) [0,1) [3,Inf) [0,1)
[82] [1,3) [0,1) [1,3) [0,1) [0,1) [0,1) [3,Inf) [0,1) [3,Inf)
[91] [1,3) [1,3) [1,3) [3,Inf) [1,3) [3,Inf) [1,3) [3,Inf) [3,Inf)
[100] [1,3)
Levels: [0,1) [1,3) [3,Inf)
```

```
# mit custom labels
cut(hospital.stays, breaks=c(0,1,3,Inf), right=FALSE,
    labels=c("0", "1-2", "mehr als 2"))
```

```
[1] 0      mehr als 2 1-2      mehr als 2 1-2      1-2
[7] 1-2    0          1-2      0          mehr als 2 1-2
[13] 1-2    0          1-2      0          0          mehr als 2
[19] 0      1-2      1-2      mehr als 2 0          1-2
[25] 1-2    1-2      mehr als 2 1-2      0          1-2
[31] mehr als 2 1-2    0          mehr als 2 1-2      1-2
[37] mehr als 2 1-2    0          1-2      1-2      mehr als 2
[43] 0      0          0          0          mehr als 2 1-2
[49] 1-2    1-2      1-2      0          1-2      0
[55] mehr als 2 0      1-2      mehr als 2 0          mehr als 2
[61] 1-2    1-2      0          1-2      mehr als 2 0
[67] 1-2    1-2      mehr als 2 0      1-2      1-2
[73] 1-2    1-2      1-2      1-2      1-2      1-2
[79] 0      mehr als 2 0      1-2      0          1-2
[85] 0      0          0          mehr als 2 0          mehr als 2
[91] 1-2    1-2      1-2      mehr als 2 1-2      mehr als 2
[97] 1-2    mehr als 2 mehr als 2 1-2
Levels: 0 1-2 mehr als 2
```

#### 4.1.4. Lösung zur Aufgabe 1.1.4 Größe und Gewicht

💡 a) Überführen Sie die Daten in R-Objekte mit den Namen Groesse und Gewicht.

```
Groesse <- c(1.68, 1.87, 1.95, 1.74, 1.80,  
            1.75, 1.59, 1.77, 1.82, 1.74)  
  
Gewicht <- c(78500, 110100, 97500, 69200, 82500,  
            71500, 81500, 87200, 75500, 65500)
```

```
# anzeigen  
Groesse
```

```
[1] 1.68 1.87 1.95 1.74 1.80 1.75 1.59 1.77 1.82 1.74
```

```
Gewicht
```

```
[1] 78500 110100 97500 69200 82500 71500 81500 87200 75500 65500
```

💡 b) Rechnen Sie das Gewicht um in Kilogramm, und speichern Sie Ihr Ergebnis in der Variable Kilogramm.

```
# Rechne Gramm in Kilogramm um  
Kilogramm <- Gewicht/1000
```

```
# anzeigen  
Kilogramm
```

```
[1] 78.5 110.1 97.5 69.2 82.5 71.5 81.5 87.2 75.5 65.5
```

💡 c) Berechnen Sie den BMI ( $\text{kg/m}^2$ ) der Probanden und speichern Ihr Ergebnis in das Objekt BMI.

```
# BMI berechnen  
BMI <- Kilogramm / (Groesse^2)
```

```
# anzeigen  
BMI
```

```
[1] 27.81321 31.48503 25.64103 22.85639 25.46296 23.34694 32.23765 27.83364  
[9] 22.79314 21.63430
```

💡 d) Fügen Sie die Objekte Groesse, Gewicht (aber in Kilogramm) und BMI zu einem Datenframe zusammen.

```
# Datenframe erzeugen
df <- data.frame(Groesse, Gewicht=Kilogramm, BMI)

# anzeigen
df
```

	Groesse	Gewicht	BMI
1	1.68	78.5	27.81321
2	1.87	110.1	31.48503
3	1.95	97.5	25.64103
4	1.74	69.2	22.85639
5	1.80	82.5	25.46296
6	1.75	71.5	23.34694
7	1.59	81.5	32.23765
8	1.77	87.2	27.83364
9	1.82	75.5	22.79314
10	1.74	65.5	21.63430

💡 e) Lassen Sie die Daten von Proband 4, 7 und 9 ausgeben.

```
df[c(4, 7, 9),]
```

	Groesse	Gewicht	BMI
4	1.74	69.2	22.85639
7	1.59	81.5	32.23765
9	1.82	75.5	22.79314

💡 f) Lassen Sie die Daten der Probanden ausgeben, deren Gewicht größer ist als 80kg.

```
df[df$Gewicht > 80 , ]
```

	Groesse	Gewicht	BMI
2	1.87	110.1	31.48503
3	1.95	97.5	25.64103
5	1.80	82.5	25.46296
7	1.59	81.5	32.23765
8	1.77	87.2	27.83364

#### 4.1.5. Lösung zur Aufgabe 1.1.5 ordinale Faktoren

💡 a) Erstellen Sie die ordinale Variable Monate, in welcher die 12 ausgeschriebenen Monatsnamen in korrekter Levelreihenfolge enthalten sind.

```
# ordinaler Faktor
Monate <- factor(c("Januar", "Februar", "März", "April", "Mai", "Juni",
                  "Juli", "August", "September", "Oktober", "November",
                  "Dezember"),
               levels= c("Januar", "Februar", "März", "April", "Mai",
                        "Juni", "Juli", "August", "September", "Oktober",
                        "November", "Dezember"),
               ordered=TRUE )
```

```
# anzeigen
Monate
```

```
[1] Januar    Februar    März       April      Mai        Juni       Juli
[8] August     September  Oktober    November   Dezember
12 Levels: Januar < Februar < März < April < Mai < Juni < Juli < ... < Dezember
```

Wir können uns aber auch ein bisschen Schreibarbeit ersparen.

```
# Hilfsvektor erzeugen
dummy <- c("Januar", "Februar", "März", "April", "Mai", "Juni", "Juli",
          "August", "September", "Oktober", "November", "Dezember")
# ordinaler Faktor
Monate <- factor(dummy, levels=dummy, ordered=TRUE)
```

```
# anzeigen
Monate
```

```
[1] Januar    Februar    März       April      Mai        Juni       Juli
[8] August     September  Oktober    November   Dezember
12 Levels: Januar < Februar < März < April < Mai < Juni < Juli < ... < Dezember
```

💡 b) Erstellen Sie die ordinale Variable Schulnoten, in welcher die 6 ausgeschriebenen Schulnoten in korrekter Levelreihenfolge enthalten sind.

```
# ordinaler Faktor
# Achten Sie auf die Reihenfolge der Schulnoten,
# wir müssen mit der schlechtesten anfangen.
Schulnoten <- c("ungenügend", "mangelhaft", "ausreichend", "befriedigend",
               "gut", "sehr gut")
Schulnoten <- factor(Schulnoten, levels=Schulnoten, ordered=TRUE)

# anzeigen
Schulnoten
```

```
[1] ungenügend   mangelhaft   ausreichend  befriedigend gut
[6] sehr gut
6 Levels: ungenügend < mangelhaft < ausreichend < befriedigend < ... < sehr gut
```

💡 c) Erzeugen Sie aus den folgenden Daten einen ordinalen Faktor mit korrekter Levelreihenfolge

```
# ordinaler Faktor
f <- factor(c("vielleicht", "glaube nicht", "nein", "glaube nicht",
             "ja", "glaube schon", "vielleicht", "nein", "glaube nicht",
             "ja", "ja", "glaube schon", "ja", "ja", "nein",
             "glaube nicht", "glaube schon", "vielleicht", "vielleicht",
             "glaube nicht", "vielleicht", "glaube nicht", "nein",
             "glaube nicht", "ja", "glaube schon", "vielleicht", "nein",
             "glaube nicht", "ja", "ja", "glaube schon", "ja", "ja",
             "nein", "glaube nicht", "glaube schon", "vielleicht",
             "vielleicht", "glaube nicht"),
           levels=c("nein", "glaube nicht", "vielleicht", "glaube schon", "ja"),
           ordered=TRUE)

# anzeigen
f
```

```
[1] vielleicht   glaube nicht nein           glaube nicht ja
[6] glaube schon vielleicht   nein           glaube nicht ja
[11] ja           glaube schon ja           ja           nein
[16] glaube nicht glaube schon vielleicht vielleicht   glaube nicht
[21] vielleicht   glaube nicht nein           glaube nicht ja
[26] glaube schon vielleicht   nein           glaube nicht ja
[31] ja           glaube schon ja           ja           nein
[36] glaube nicht glaube schon vielleicht vielleicht   glaube nicht
Levels: nein < glaube nicht < vielleicht < glaube schon < ja
```

💡 d) Ändern Sie die Levelnamen in -2, -1, 0, 1, 2.

```
# Levelnamen ändern
levels(f) <- c("-2", "-1", "0", "1", "2")

# anzeigen
f

[1] 0 -1 -2 -1 2 1 0 -2 -1 2 2 1 2 2 -2 -1 1 0 0 -1 0 -1 -2 -1 2
[26] 1 0 -2 -1 2 2 1 2 2 -2 -1 1 0 0 -1
Levels: -2 < -1 < 0 < 1 < 2
```

#### 4.1.6. Lösung zur Aufgabe 1.1.6 Hogwarts-Kurse

💡 a) Erstellen Sie das Datenframe Kurse, in welchem die Daten aus den Tabellenspalten Haus und Kurs enthalten sind.

```
# Daten übertragen
Kurse <- data.frame(
  Haus = c("Gryffindor", "Gryffindor", "Gryffindor", "Gryffindor",
           "Hufflepuff", "Hufflepuff", "Hufflepuff", "Hufflepuff",
           "Ravenclaw", "Ravenclaw", "Ravenclaw", "Ravenclaw",
           "Slytherin", "Slytherin", "Slytherin", "Slytherin"),
  Kurs = c("Verteidigung gegen die dunklen Künste", "Zauberkunst",
           "Verwandlung", "Besenflugunterricht",
           "Kräuterkunde", "Pflege magischer Geschöpfe",
           "Geschichte der Zauberei", "Alte Runen",
           "Arithmantik", "Astronomie",
           "Verwandlung", "Verteidigung gegen die dunklen Künste",
           "Zaubertränke", "Zauberkunst",
           "Dunkle Künste", "Legilimentik")
)
# anzeigen
Kurse
```

	Haus	Kurs
1	Gryffindor	Verteidigung gegen die dunklen Künste
2	Gryffindor	Zauberkunst
3	Gryffindor	Verwandlung
4	Gryffindor	Besenflugunterricht
5	Hufflepuff	Kräuterkunde
6	Hufflepuff	Pflege magischer Geschöpfe
7	Hufflepuff	Geschichte der Zauberei
8	Hufflepuff	Alte Runen
9	Ravenclaw	Arithmantik
10	Ravenclaw	Astronomie
11	Ravenclaw	Verwandlung

12	Ravenclaw	Verteidigung gegen die dunklen Künste
13	Slytherin	Zaubertränke
14	Slytherin	Zauberkunst
15	Slytherin	Dunkle Künste
16	Slytherin	Legilimentik

💡 b) Wieviele Kurse haben es in die Auswahlliste geschafft?

```
# unique()
unique(Kurse$Kurs)
```

```
[1] "Verteidigung gegen die dunklen Künste"
[2] "Zauberkunst"
[3] "Verwandlung"
[4] "Besenflugunterricht"
[5] "Kräuterkunde"
[6] "Pflege magischer Geschöpfe"
[7] "Geschichte der Zauberei"
[8] "Alte Runen"
[9] "Arithmantik"
[10] "Astronomie"
[11] "Zaubertränke"
[12] "Dunkle Künste"
[13] "Legilimentik"
```

```
length(unique(Kurse$Kurs))
```

```
[1] 13
```

Es sind 13 Kurse in der Liste.

💡 c) Erstellen Sie für jedes Haus ein eigenes Datenframe

```
# Subsets erstellen
gryffindor <- subset(Kurse, Haus=="Gryffindor")
hufflepuff <- subset(Kurse, Haus=="Hufflepuff")
ravenclaw <- subset(Kurse, Haus=="Ravenclaw")
slytherin <- subset(Kurse, Haus=="Slytherin")
```



💡 d) Wandeln Sie in jedem Haus-Datenframe die Variablen in Faktoren um.

```
# Subsets erstellen
gryffindor$Kurs <- factor(gryffindor$Kurs)
gryffindor$Haus <- factor(gryffindor$Haus)

hufflepuff$Kurs <- factor(hufflepuff$Kurs)
hufflepuff$Haus <- factor(hufflepuff$Haus)

ravenclaw$Kurs <- factor(ravenclaw$Kurs)
ravenclaw$Haus <- factor(ravenclaw$Haus)

slytherin$Kurs <- factor(slytherin$Kurs)
slytherin$Haus <- factor(slytherin$Haus)
```

💡 e) Fügen Sie die Haus-Datenframes zu einem einzigen Datenframe Hogwarts zusammen, in der Reihenfolge Ravenclaw, Gryffindor, Slytherin und Hufflepuff. Ändern Sie anschließend den Kurs “Geschichte der Zauberei” in “Geisterkunde” um.

```
# Zusammenführen
Hogwarts <- rbind(ravenclaw, gryffindor, slytherin, hufflepuff)

# Level ändern
levels(Hogwarts$Kurs)[levels(Hogwarts$Kurs)=="Geschichte der Zauberei"] <- "Geisterkunde"

# anzeigen
Hogwarts$Kurs
```

```
[1] Arithmantik          Astronomie
[3] Verwandlung          Verteidigung gegen die dunklen Künste
[5] Verteidigung gegen die dunklen Künste Zauberkunst
[7] Verwandlung          Besenflugunterricht
[9] Zaubertränke         Zauberkunst
[11] Dunkle Künste        Legilimentik
[13] Kräuterkunde         Pflege magischer Geschöpfe
[15] Geisterkunde         Alte Runen
13 Levels: Arithmantik Astronomie ... Pflege magischer Geschöpfe
```

💡 f) Sortieren Sie den Datensatz, so dass die Kurse in alphabetischer Reihenfolge angezeigt werden.

Wenn wir “einfach so” die `order()`-Funktion nutzen, erhalten wir eine falsche Ausgabe.

```
# wird nicht korrekt sortiert
Hogwarts[order(Hogwarts$Kurs),]
```

	Haus	Kurs
9	Ravenclaw	Arithmantik
10	Ravenclaw	Astronomie

12	Ravenclaw	Verteidigung gegen die dunklen Künste
1	Gryffindor	Verteidigung gegen die dunklen Künste
11	Ravenclaw	Verwandlung
3	Gryffindor	Verwandlung
4	Gryffindor	Besenflugunterricht
2	Gryffindor	Zauberkunst
14	Slytherin	Zauberkunst
15	Slytherin	Dunkle Künste
16	Slytherin	Legilimentik
13	Slytherin	Zaubertränke
8	Hufflepuff	Alte Runen
7	Hufflepuff	Geisterkunde
5	Hufflepuff	Kräuterkunde
6	Hufflepuff	Pflege magischer Geschöpfe

Das liegt daran, dass `Hogwarts$Kurs` als Factor vorliegt, und somit nach Levelreihenfolge sortiert wird.

```
# Datenklasse Factor
class(Hogwarts$Kurs)
```

```
[1] "factor"
```

Wir müssen daher die Funktion `as.character()` um die Variable wickeln, um eine alphabetische Sortierung zu erzwingen.

```
# jetzt klappt es
Hogwarts[order(as.character(Hogwarts$Kurs)),]
```

	Haus	Kurs
8	Hufflepuff	Alte Runen
9	Ravenclaw	Arithmantik
10	Ravenclaw	Astronomie
4	Gryffindor	Besenflugunterricht
15	Slytherin	Dunkle Künste
7	Hufflepuff	Geisterkunde
5	Hufflepuff	Kräuterkunde
16	Slytherin	Legilimentik
6	Hufflepuff	Pflege magischer Geschöpfe
12	Ravenclaw	Verteidigung gegen die dunklen Künste
1	Gryffindor	Verteidigung gegen die dunklen Künste
11	Ravenclaw	Verwandlung
3	Gryffindor	Verwandlung
2	Gryffindor	Zauberkunst
14	Slytherin	Zauberkunst
13	Slytherin	Zaubertränke

💡 g) Speichern Sie den so sortierten Datensatz in das Objekt `sorted`, und reparieren Sie die Zeilennummerierung von `sorted`.

```
# sortiert speichern
sorted <- Hogwarts[order(as.character(Hogwarts$Kurs)),]

# Zeilennummerierung reparieren
rownames(sorted) <- 1:length(sorted$Kurs)

# anzeigen
sorted
```

	Haus	Kurs
1	Hufflepuff	Alte Runen
2	Ravenclaw	Arithmantik
3	Ravenclaw	Astronomie
4	Gryffindor	Besenflugunterricht
5	Slytherin	Dunkle Künste
6	Hufflepuff	Geisterkunde
7	Hufflepuff	Kräuterkunde
8	Slytherin	Legilimentik
9	Hufflepuff	Pflege magischer Geschöpfe
10	Ravenclaw	Verteidigung gegen die dunklen Künste
11	Gryffindor	Verteidigung gegen die dunklen Künste
12	Ravenclaw	Verwandlung
13	Gryffindor	Verwandlung
14	Gryffindor	Zauberkunst
15	Slytherin	Zauberkunst
16	Slytherin	Zaubertränke

#### 4.1.7. Lösung zur Aufgabe 1.1.7 Datentabelle

💡 a) Übertragen Sie die Daten in das Datenframe `chol`.

```
# Daten übertragen
chol <- data.frame(Name = c("Anna Tomie", "Bud Zillus", "Dieter Mietenplage",
                           "Hella Scheinwerfer", "Inge Danken", "Jason Zufall"),
                  Geschlecht = c("W", "M", "M", "W", "W", "M"),
                  Gewicht = c(85, 115, 79, 60, 57, 96),
                  Größe = c(179, 173, 181, 170, 158, 174),
                  Cholesterol = c(182, 232, 191, 200, 148, 249)
                  )

# anzeigen
chol
```

	Name	Geschlecht	Gewicht	Größe	Cholesterol
1	Anna Tomie	W	85	179	182

2	Bud Zillus	M	115	173	232
3	Dieter Mietenplage	M	79	181	191
4	Hella Scheinwerfer	W	60	170	200
5	Inge Danken	W	57	158	148
6	Jason Zufall	M	96	174	249

💡 b) Erstellen Sie eine neue Variable Alter, die zwischen Name und Geschlecht liegt

```
# Daten übertragen
alter <- c(18, 32, 24, 35, 46, 68)

# zwischen Name und Geschlecht einfügen
chol <- data.frame(Name=chol$Name, Alter=alter, Geschlecht=chol$Geschlecht,
                   Gewicht=chol$Gewicht, Größe=chol$Größe,
                   Cholesterol=chol$Cholesterol)

# anzeigen
chol
```

	Name	Alter	Geschlecht	Gewicht	Größe	Cholesterol
1	Anna Tomie	18	W	85	179	182
2	Bud Zillus	32	M	115	173	232
3	Dieter Mietenplage	24	M	79	181	191
4	Hella Scheinwerfer	35	W	60	170	200
5	Inge Danken	46	W	57	158	148
6	Jason Zufall	68	M	96	174	249

💡 c) Fügen Sie einen weiteren Fall mit folgenden Daten dem Datenframe hinzu.

```
# Daten übertragen
neu <- data.frame(Name="Mitch Mackes", Alter=44, Geschlecht="M", Gewicht=92,
                  Größe=178, Cholesterol=220)

# zusammenfügen
chol <- rbind(chol, neu)

# anzeigen
chol
```

	Name	Alter	Geschlecht	Gewicht	Größe	Cholesterol
1	Anna Tomie	18	W	85	179	182
2	Bud Zillus	32	M	115	173	232
3	Dieter Mietenplage	24	M	79	181	191
4	Hella Scheinwerfer	35	W	60	170	200
5	Inge Danken	46	W	57	158	148
6	Jason Zufall	68	M	96	174	249
7	Mitch Mackes	44	M	92	178	220

💡 d) Erzeugen Sie eine neue Variable BMI ( $BMI = \frac{kg}{m^2}$ ).

```
# BMI hinzufügen
# Größe muss in Meter umgerechnet werden
chol$BMI <- chol$Gewicht / (chol$Größe/100)^2

# anzeigen
chol
```

	Name	Alter	Geschlecht	Gewicht	Größe	Cholesterol	BMI
1	Anna Tomie	18	W	85	179	182	26.52851
2	Bud Zillus	32	M	115	173	232	38.42427
3	Dieter Mietenplage	24	M	79	181	191	24.11404
4	Hella Scheinwerfer	35	W	60	170	200	20.76125
5	Inge Danken	46	W	57	158	148	22.83288
6	Jason Zufall	68	M	96	174	249	31.70828
7	Mitch Mackes	44	M	92	178	220	29.03674

💡 e) Fügen Sie die Variable Adipositas hinzu, in welcher Sie die BMI-Werte klassieren

Ein Klassierung kann auf mehrere Weisen erfolgen.

```
# bedingtes Referenzieren
chol$Adipositas[chol$BMI < 18.5] <- "Untergewicht"
chol$Adipositas[chol$BMI >= 18.5 & chol$BMI < 24.5] <- "Normalgewicht"
chol$Adipositas[chol$BMI >= 24.5 & chol$BMI < 30] <- "Übergewicht"
chol$Adipositas[chol$BMI >= 30] <- "Adipositas"

# anzeigen
chol
```

	Name	Alter	Geschlecht	Gewicht	Größe	Cholesterol	BMI
1	Anna Tomie	18	W	85	179	182	26.52851
2	Bud Zillus	32	M	115	173	232	38.42427
3	Dieter Mietenplage	24	M	79	181	191	24.11404
4	Hella Scheinwerfer	35	W	60	170	200	20.76125
5	Inge Danken	46	W	57	158	148	22.83288
6	Jason Zufall	68	M	96	174	249	31.70828
7	Mitch Mackes	44	M	92	178	220	29.03674

	Adipositas
1	Übergewicht
2	Adipositas
3	Normalgewicht
4	Normalgewicht
5	Normalgewicht
6	Adipositas
7	Übergewicht

Alternativ kann die cut()-Funktion verwendet werden.

```
# cut-Funktion
chol$Adipositas <- cut(chol$BMI, breaks = c(0, 18.5, 24.5, 30, Inf),
                      labels = c("Untergewicht", "Normalgewicht",
                                "Übergewicht", "Adipositas"),
                      right = FALSE)

# anzeigen
chol
```

	Name	Alter	Geschlecht	Gewicht	Größe	Cholesterol	BMI
1	Anna Tomie	18	W	85	179	182	26.52851
2	Bud Zillus	32	M	115	173	232	38.42427
3	Dieter Mietenplage	24	M	79	181	191	24.11404
4	Hella Scheinwerfer	35	W	60	170	200	20.76125
5	Inge Danken	46	W	57	158	148	22.83288
6	Jason Zufall	68	M	96	174	249	31.70828
7	Mitch Mackes	44	M	92	178	220	29.03674

	Adipositas
1	Übergewicht
2	Adipositas
3	Normalgewicht
4	Normalgewicht
5	Normalgewicht
6	Adipositas
7	Übergewicht

💡 f) Filtern Sie Ihren Datensatz, so dass Sie einen neuen Datensatz male erhalten, welcher nur die Daten der Männer beinhaltet.

```
# subset erzeugen
male <- subset(chol, Geschlecht=="M")

# anzeigen
male
```

	Name	Alter	Geschlecht	Gewicht	Größe	Cholesterol	BMI
2	Bud Zillus	32	M	115	173	232	38.42427
3	Dieter Mietenplage	24	M	79	181	191	24.11404
6	Jason Zufall	68	M	96	174	249	31.70828
7	Mitch Mackes	44	M	92	178	220	29.03674

	Adipositas
2	Adipositas
3	Normalgewicht
6	Adipositas
7	Übergewicht

## 5. Lösungswege zu den Aufgaben für geübte User:innen

**i** Wenn Ihr R-Code eleganter ist als die hier präsentierten Lösungswege, dann freuen Sie sich! Wenn Sie meinen, Ihr Code sei zu klobig und umständlich, dann Kopf hoch: wenn er tut, was er soll, dann ist er genau richtig.

### 5.1. Lösungen zu Objekten in R

#### 5.1.1. Lösung zur Aufgabe 2.1.1 Hogwarts-Kurse

**💡** a) Benutzen Sie die `tribble()`-Funktion, um die Daten in die Objekte `tab1` und `tab2` zu überführen.

```
library(tibble)
tab1 <- tribble(
  ~Hufflepuff,          ~Slytherin,
  "Kräuterkunde",      "Zaubertränke",
  "Pflege magischer Geschöpfe", "Zauberkunst",
  "Geschichte der Zauberei",  "Dunkle Künste",
  "Alte Runen",          "Legilimentik"
)

tab2 <- tribble(
  ~Gryffindor,          ~Ravenclaw,
  "Verteidigung gegen die dunklen Künste", "Arithmantik",
  "Zauberkunst",        "Astronomie",
  "Verwandlung",        "Verwandlung",
  "Besenflugunterricht", "Verteidigung gegen die dunklen Künste"
)
# anzeigen
tab1
```

```
# A tibble: 4 x 2
  Hufflepuff      Slytherin
  <chr>          <chr>
1 Kräuterkunde   Zaubertränke
2 Pflege magischer Geschöpfe Zauberkunst
3 Geschichte der Zauberei   Dunkle Künste
4 Alte Runen      Legilimentik
```

```
tab2
```

```
# A tibble: 4 x 2
  Gryffindor          Ravenclaw
  <chr>            <chr>
1 Verteidigung gegen die dunklen Künste Arithmantik
2 Zauberkunst        Astronomie
3 Verwandlung        Verwandlung
4 Besenflugunterricht Verteidigung gegen die dunklen Künste
```

💡 b) Fügen Sie tab1 und tab2 zu einem Objekt Hogwarts zusammen.

```
Hogwarts <- cbind(tab1, tab2)
```

```
# anzeigen
str(Hogwarts)
```

```
'data.frame':  4 obs. of  4 variables:
 $ Hufflepuff: chr  "Kräuterkunde" "Pflege magischer Geschöpfe" "Geschichte der Zauberei" "A
 $ Slytherin : chr  "Zaubertränke" "Zauberkunst" "Dunkle Künste" "Legilimentik"
 $ Gryffindor: chr  "Verteidigung gegen die dunklen Künste" "Zauberkunst" "Verwandlung" "Bes
 $ Ravenclaw : chr  "Arithmantik" "Astronomie" "Verwandlung" "Verteidigung gegen die dunklen
```

💡 c) Nutzen Sie die mutate()-Funktion, um die Datenklassen der Variablen anzupassen (Skalenniveau).

```
library(dplyr)
Hogwarts <- Hogwarts %>%
  mutate_if(is.character, as.factor)
```

```
# anzeigen
str(Hogwarts)
```

```
'data.frame':  4 obs. of  4 variables:
 $ Hufflepuff: Factor w/ 4 levels "Alte Runen","Geschichte der Zauberei",...: 3 4 2 1
 $ Slytherin : Factor w/ 4 levels "Dunkle Künste",...: 4 3 1 2
 $ Gryffindor: Factor w/ 4 levels "Besenflugunterricht",...: 2 4 3 1
 $ Ravenclaw : Factor w/ 4 levels "Arithmantik",...: 1 2 4 3
```



💡 d) Ändern Sie anschließend mit der `mutate()`-Funktion den Kurs “*Geschichte der Zauberei*” in “*Geisterkunde*” um.

```
library(dplyr)
library(forcats)
Hogwarts <- Hogwarts %>%
  mutate(Hufflepuff = fct_recode(Hufflepuff,
                                "Geisterkunde" = "Geschichte der Zauberei"))

# anzeigen
Hogwarts
```

	Hufflepuff	Slytherin	
1	Kräuterkunde	Zaubertränke	
2	Pflege magischer Geschöpfe	Zauberkunst	
3	Geisterkunde	Dunkle Künste	
4	Alte Runen	Legilimentik	
		Gryffindor	Ravenclaw
1	Verteidigung gegen die dunklen Künste		Arithmantik
2		Zauberkunst	Astronomie
3		Verwandlung	Verwandlung
4	Besenflugunterricht	Verteidigung gegen die dunklen Künste	

💡 e) Die Daten liegen nicht im Tidy-Data-Format vor. Erzeugen Sie ein neues Objekt `Kurse` mit den Variablen `Haus` und `Kurs`.

```
library(tidyr)
Kurse <- Hogwarts %>%
  pivot_longer(Hufflepuff:Ravenclaw,
               names_to = "Haus",
               values_to = "Kurs")

# anzeigen
Kurse
```

# A tibble: 16 x 2

	Haus	Kurs
	<chr>	<fct>
1	Hufflepuff	Kräuterkunde
2	Slytherin	Zaubertränke
3	Gryffindor	Verteidigung gegen die dunklen Künste
4	Ravenclaw	Arithmantik
5	Hufflepuff	Pflege magischer Geschöpfe
6	Slytherin	Zauberkunst
7	Gryffindor	Zauberkunst
8	Ravenclaw	Astronomie
9	Hufflepuff	Geisterkunde
10	Slytherin	Dunkle Künste
11	Gryffindor	Verwandlung

- 12 Ravenclaw Verwandlung
- 13 Hufflepuff Alte Runen
- 14 Slytherin Legilimentik
- 15 Gryffindor Besenflugunterricht
- 16 Ravenclaw Verteidigung gegen die dunklen Künste

## 6. Lösungswege zu den Aufgaben für fortgeschrittene Anwender:innen

- i** Wenn Ihr R-Code eleganter ist als die hier präsentierten Lösungswege, dann freuen Sie sich! Wenn Sie meinen, Ihr Code sei zu klobig und umständlich, dann Kopf hoch: wenn er tut, was er soll, dann ist er genau richtig.

### 6.1. Lösungen zu Objekten in R

#### 6.1.1. Lösung zur Aufgabe 3.1.1 Hogwarts-Kurse

- 💡** a) Benutzen Sie die `tribble()`-Funktion, um die Daten in die Objekte `tab1` und `tab2` zu überführen.

```
library(tibble)
```

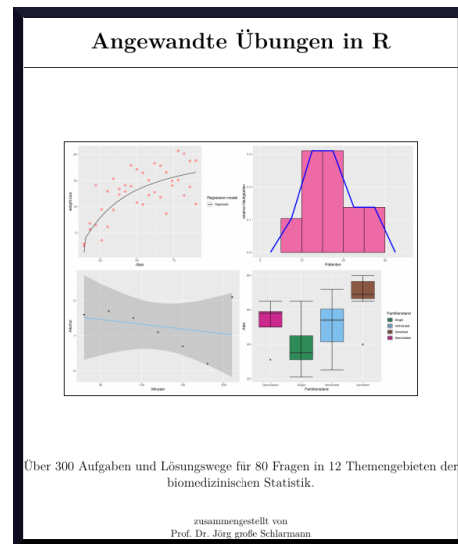
# Literaturverzeichnis

- große Schlarmann, J. (2024a). *Angewandte Übungen in R*. Hochschule Niederrhein. [https://github.com/produnis/angewandte\\_uebungen\\_in\\_R](https://github.com/produnis/angewandte_uebungen_in_R)
- große Schlarmann, J. (2024b). *Statistik mit R und RStudio - Ein Nachschlagewerk für Gesundheitsberufe*. Hochschule Niederrhein. <https://www.produnis.de/R>
- Mock, T. (2022). *Tidy Tuesday: A weekly data project aimed at the R ecosystem*. <https://github.com/rfordatascience/tidytuesday>
- R Core Team. (2023). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing. <https://www.R-project.org/>
- Walther, B. (2022). *Statistik mit R Schnelleinstieg*. MITP Verlags GmbH.
- Wickham, H., Çetinkaya-Rundel, M., & Grolemund, G. (2023). *R for Data Science*. O'Reilly Media. <https://r4ds.hadley.nz/>

# Credits



(a) große Schlarmann (2024b)



(a) große Schlarmann (2024a)

Prof. Dr. Jörg große Schlarmann  
Hochschule Niederrhein, Krefeld  
[joerg.grosseschlarmann@hs-niederrhein.de](mailto:joerg.grosseschlarmann@hs-niederrhein.de)  
<https://www.produnis.de/R>