

table traineR

Workouts für {data.table}



zusammengestellt von
Prof. Dr. Jörg große Schlarmann



Version vom 26.10.2024

Lizenz

Willkommen beim Table Training!

In diesem Buch sind zahlreiche Übungen zur freien Statistiksoftware R enthalten, die sich ausschließlich mit dem Paket `{data.table}` befassen.



Dieses Script ist unter der Creative Commons BY-NC-SA 4.0¹ 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:

- **👤 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.
- **🚫 Nicht kommerziell** — Sie dürfen das Material nicht für kommerzielle Zwecke nutzen.
- **🔄 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): “table traineR. Workouts für `{data.table}`”, Hochschule Niederrhein, <https://www.produnis.de/tabletrainer/>

```
@book{grSchl_tabletrainer,  
  author = {{große Schlarmann}, Jörg},  
  title = {{table traineR}. Workouts für \{data.table\}},  
  year = {2024},  
  publisher = {Hochschule Niederrhein},  
  address = {Krefeld},  
  copyright = {CC BY-NC-SA 4.0},  
  url = {https://www.produnis.de/tabletrainer/},  
  language = {de},  
}
```

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

Inhaltsverzeichnis

Lizenz	i
Inhaltsverzeichnis	ii
1. Einleitung	1
1.1. Vorbereitungen	1
2. Kurze Einführung in {data.table}	3
2.1. Installation	3
2.2. Modify-in-Place	3
2.3. Grundlegende Syntax	4
2.4. Daten einlesen	4
2.5. Daten speichern	5
2.6. Fälle filtern mit i	6
2.7. Fälle sortieren mit i	7
2.8. Daten verarbeiten mit j	8
2.9. Daten bearbeiten mit j	9
2.10. data.table kopieren	10
2.11. pipen	12
2.12. Ergebnisse gruppieren mit by	13
2.13. Weitere Funktionen aus dem data.table Paket	15
2.13.1. Einzigartige bestimmen mit uniqueN	15
2.13.2. Anzahl der Fälle mit .N	15
2.13.3. Lange Tabelle erzeugen mit melt()	16
2.13.4. Breite Tabelle erzeugen mit dcast()	18
2.13.5. Subset of Data (.SD)	19
2.14. Cheat Sheet	21
I. Aufgaben	22
3. Aufgaben	23
3.1. Aufgabe 3.1 Größe und Gewicht	23
3.2. Aufgabe 3.2 Datentabelle	24
3.3. Aufgabe 3.3 Big Five	25
3.4. Aufgabe 3.4 Rolling Stone Magazine	26
3.5. Aufgabe 3.5 Taylor Swift	27
3.6. Aufgabe 3.6 Anscombe-Quartett	27
3.7. Aufgabe 3.7 Neugeborene	28
3.8. Aufgabe 3.8 Verteidigung gegen die dunklen Künste	29
3.9. Aufgabe 3.9 Hogwarts Hauspunkte	29
3.10. Aufgabe 3.10 Lungenkapazität	30

II. Lösungswege	31
4. Lösungswege	32
4.1. Lösung zur Aufgabe 3.1 Größe und Gewicht	32
4.2. Lösung zur Aufgabe 3.2 Datentabelle	35
4.3. Lösung zur Aufgabe 3.3 Big Five	39
4.4. Lösung zur Aufgabe 3.4 Rolling Stone Magazine	41
4.5. Lösung zur Aufgabe 3.5 Taylor Swift	51
4.6. Lösung zur Aufgabe 3.6 Anscombe-Quartett	60
4.7. Lösung zur Aufgabe 3.7 Neugeborene	61
4.8. Lösung zur Aufgabe 3.8 Verteidigung gegen die dunklen Künste	67
4.9. Lösung zur Aufgabe 3.9 Hogwarts Hauspunkte	69
4.10. Lösung zur Aufgabe 3.10 Lungenkapazität	72
Literaturverzeichnis	78
Credits	79

1. 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 zum Table Training!

In diesem Buch sind zahlreiche Übungen zur freien Statistiksoftware R enthalten, die sich ausschließlich mit dem Paket `{data.table}` befassen. Dafür werden Grundkenntnisse in R und RStudio vorausgesetzt.

Sollten Sie neu in R sein, ist diese Aufgabensammlung wahrscheinlich nicht für Sie geeignet. Viel mehr könnte Ihnen das freie Nachschlagewerk von große Schlarmann (2024b) einen niederschweligen Einstieg in R ermöglichen.



Wenn Sie schon erste Schritte in R und dem `tidyverse` gegangen sind, können Sie nun versuchen, Ihre Lösungsstrategien mit `{data.table}` umzusetzen. Lassen Sie sich nicht entmutigen, R hat eine steile Lernkurve, und nur durch Übung kommen Sie weiter. Diese Sammlung möchte Sie auf Ihrem Weg begleiten und Sie befähigen, *typische* Aufgaben in R sicher mit `{data.table}` zu meistern.

Der Quelltext dieses Buchs ist bei GitHub verfügbar, siehe <https://github.com/produnis/tabletrainer>.

- Eine aktuelle epub-Version finden Sie unter:
<https://www.produnis.de/tabletrainer/tabletraineR.epub>
- Eine aktuelle PDF-Version finden Sie unter:
<https://www.produnis.de/tabletrainer/tabletraineR.pdf>
- Kritik und Diskussion sind per Mastodon möglich:
<https://mastodon.social/@rbuch>

1.1. Vorbereitungen

Falls Sie **RStudio** verwenden, legen Sie sich ein eigenes Projekt `tabletraineR` für die Übungen an.

Dies hat den Vorteil, dass alle Dateien in einem Ordner gesammelt vorhanden sind. Um Ihre Datensätze von Ihren Scriptdateien zu separieren, empfiehlt es sich, einen `data`-Ordner anzulegen, in welchem alle Datensätze gespeichert werden können (Abbildung 1.1).

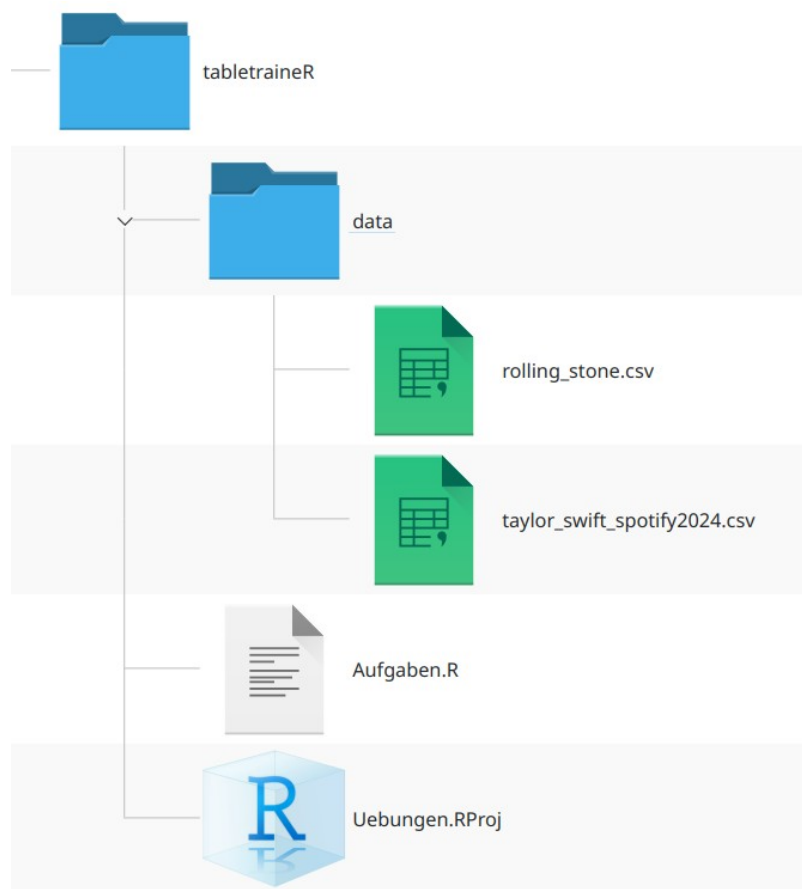
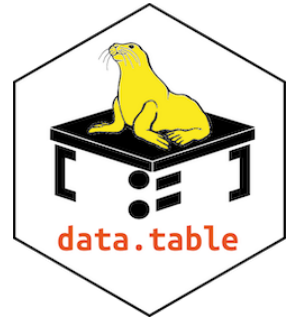


Abbildung 1.1.: beispielhafte Ordnerstruktur

2. Kurze Einführung in {data.table}

Neben dem tidyverse steht mit `data.table` ein weiterer R-Dialekt zur Verfügung, der sich immer größerer Beliebtheit erfreut. Im Kern sind `data.tables` verbesserte Versionen von `data.frames`, die schneller und speichereffizienter arbeiten und mit einer prägnanteren Syntax manipuliert werden können. Das Paket stellt außerdem eine Reihe zusätzlicher Funktionen zum Lesen und Schreiben von tabellarischen Dateien, zum Umformen von Daten zwischen langen und breiten Formaten und zum Verbinden von Datensätzen zur Verfügung.



2.1. Installation

Alle Funktionen sind über das Paket `data.table` implementiert, welches wie gewohnt installiert und aktiviert werden kann.

```
# installiere data.table
install.packages("data.table", dependencies=TRUE)
```

```
# data.table aktivieren
library(data.table)
```

2.2. Modify-in-Place

Der größte Unterschied besteht darin, dass `data.table` die *Modify-in-Place*-Methode verwendet. Das klassische R und auch das Tidyverse verwenden die *Copy-on-Modify*-Methode, welche besagt, dass bei der Manipulation eines Objektes das Ergebnis in einem neuen Objekt gespeichert wird.

```
# klassisches "Copy-on-Modify"
meine.daten %>%
  mutate(Neu = Alt*10)
```

Bei oben stehendem Code wird das Objekt `meine.daten` nicht verändert. Das Ergebnis der `mutate()`-Funktion wird als neues Objekt ausgegeben. Dieses neue Objekt ist eine Kopie der Ursprungsdaten `meine.daten`, an welcher die Veränderungen vorgenommen werden.

Mit `data.table` wird der Ansatz *Modify-in-Place* verfolgt.

```
# Modify-in-Place
meine.daten[, Neu := Alt*10]
```

Der oben stehende Code erzeugt keine Kopie von `meine.daten`. Vielmehr wird das Objekt `meine.daten` **direkt** verändert. Im klassischen R entspricht diese Vorgehensweise dem Code

```
meine.daten$Neu <- meine.daten$Alt*10
```

Durch *Modify-in-Place* wird `data.table` sehr effizient, wenn größere Datenmengen verarbeitet werden sollen. Es kann jedoch auch dazu führen, dass der Code schwieriger zu verstehen ist und überraschende Ergebnisse liefert (insbesondere, wenn ein `data.table` innerhalb einer Funktion modifiziert wird).

2.3. Grundlegende Syntax

Die generelle Syntax von `data.table` lautet

```
dt[i, j, by]
```

wobei

- `dt` ein `data.table`-Objekt ist.
- `i` zum Filtern und für `join`-Funktionen genutzt wird.
- `j` zum Manipulieren, Transformieren und Zusammenfassen der Daten verwendet wird.
- `by` zum Gruppieren genutzt wird.

Man kann die Syntax lesen als:

„In diesen Zeilen, mache dies, gruppiert nach jenem“.

2.4. Daten einlesen

Der erste Schritt der meisten Datenanalysen besteht darin, Daten in den Speicher zu laden. Wir können die Funktion `data.table::fread()` verwenden (das `f` steht für *fast* (schnell)), um reguläre, durch Trennzeichen getrennte Dateien wie `txt`- oder `csv`-Dateien zu lesen. Diese Funktion ist nicht nur schnell, sondern erkennt automatisch das Trennzeichen und errät die Klasse jeder Spalte sowie die Anzahl der Zeilen in der Datei.

```
# Daten einlesen mit fread()
dt <- fread("data/Befragung22.csv")

# anschauen
str(dt)
```

```
Classes 'data.table' and 'data.frame': 37 obs. of 6 variables:
 $ alter      : int  20 28 41 34 26 38 28 21 27 26 ...
 $ geschlecht: chr   "weiblich" "weiblich" "männlich" "weiblich" ...
 $ stifte     : int  12 7 1 13 18 25 29 1 2 5 ...
 $ geburtsort: chr   "Düren" "Neuss" "Bonn" "Düsseldorf" ...
 $ fahrzeit   : int  1 45 60 25 15 50 40 60 60 40 ...
 $ podcast    : chr   "selten" "selten" "selten" "oft" ...
 - attr(*, ".internal.selfref")=<externalptr>
```

Das Objekt `dt` gehört sowohl zur Klasse `data.frame` als auch zu der neuen Klasse `data.table`.

Die Daten können auch direkt über eine URL eingelesen werden.


```
# lade per URL
dt <- fread("https://www.produnis.de/tabletrainer/data/Befragung22.csv")
```

Liegen die Daten bereits als `data.frame` vor, können sie per `as.data.table()` umgewandelt werden.

```
# lade klassisches Datenframe
df <- read.table("https://www.produnis.de/tabletrainer/data/Datentabelle.txt",
                header=TRUE)

# wandle in data.table um
dt2 <- as.data.table(df)
```

```
Classes 'data.table' and 'data.frame':  10 obs. of  4 variables:
 $ Geschlecht: chr  "m" "w" "w" "m" ...
 $ Alter      : int  28 18 25 29 21 19 27 26 31 22
 $ Gewicht    : int  80 55 74 101 84 74 65 56 88 78
 $ Groesse    : int  170 174 183 190 185 178 169 163 189 184
 - attr(*, ".internal.selfref")=<externalptr>
```

Sollen die Daten von Hand eingegeben werden, wird die Funktion `data.table()` verwendet.

```
# erzeuge von Hand
dt3 <- data.table(x = 1:10,
                 y = 11:20,
                 z = factor(rep(c("foo", "bar"), 5))
                 )

# anschauen
str(dt3)
```

```
Classes 'data.table' and 'data.frame':  10 obs. of  3 variables:
 $ x: int  1 2 3 4 5 6 7 8 9 10
 $ y: int  11 12 13 14 15 16 17 18 19 20
 $ z: Factor w/ 2 levels "bar","foo": 2 1 2 1 2 1 2 1 2 1
 - attr(*, ".internal.selfref")=<externalptr>
```

2.5. Daten speichern

Mit der Funktion `fwrite()` können `data.tables` (aber auch `data.frames`) in eine Datei gespeichert werden. Sie funktioniert ähnlich wie `write.csv`, ist aber wesentlich schneller. Wird kein Dateiname angegeben, erfolgt die Ausgabe in der Konsole. So kann überprüft werden, was in die Datei geschrieben würde.

```
# schreibe Objekt "dt2" in die Konsole
fwrite(dt2)
```

```
Geschlecht,Alter,Gewicht,Groesse
m,28,80,170
w,18,55,174
w,25,74,183
```

```
m,29,101,190
m,21,84,185
w,19,74,178
w,27,65,169
w,26,56,163
m,31,88,189
m,22,78,184
```

```
# schreibe Objekt "dt" in datei "dt.csv"
fwrite(dt2, "dt2.csv")
# schreibe Objekt "dt" in datei "dt.txt"
fwrite(dt2, "dt2.txt")
```

2.6. Fälle filtern mit `i`

Wir erinnern uns, dass die allgemeine Syntax `dt[i, j, by]` lautet. Über den Parameter `i` können die Daten gefiltert werden, so dass nur bestimmte Fälle berücksichtigt werden. Beispielsweise könnten wir im Objekt `dt` nur solche Fälle auswählen, bei denen das Alter größer als 30 ist.

```
dt[alter > 30]
```

	alter	geschlecht	stifte	geburtsort	fahrzeit	podcast
	<int>	<char>	<int>	<char>	<int>	<char>
1:	41	männlich	1	Bonn	60	selten
2:	34	weiblich	13	Düsseldorf	25	oft
3:	38	weiblich	25	Dinslaken	50	oft
4:	38	männlich	5	Donezk	57	manchmal
5:	31	weiblich	16	Charkov Ukraine	135	oft
6:	36	weiblich	1	Rybnik	90	manchmal
7:	45	männlich	1	Gelsenkirchen	85	oft

Dies ist Vergleichbar mit dem klassischen R-Aufruf

```
# klassischer R-Befehl
dt[dt$alter > 30]
```

	alter	geschlecht	stifte	geburtsort	fahrzeit	podcast
	<int>	<char>	<int>	<char>	<int>	<char>
1:	41	männlich	1	Bonn	60	selten
2:	34	weiblich	13	Düsseldorf	25	oft
3:	38	weiblich	25	Dinslaken	50	oft
4:	38	männlich	5	Donezk	57	manchmal
5:	31	weiblich	16	Charkov Ukraine	135	oft
6:	36	weiblich	1	Rybnik	90	manchmal
7:	45	männlich	1	Gelsenkirchen	85	oft

Da alle Ausdrücke in `i` im Kontext der `data.table` ausgewertet werden, müssen wir den (eventuell sehr langen) Namen des Objektes nicht erneut eingeben. Dies ist vor allem bei längeren Ausdrücken sehr bequem.

```
# erzeuge langen Objektnamen
langer.Objekt.name <- dt
```

Der klassische R-Aufruf

```
# klassischer R-Befehl
langer.Objekt.name[langer.Objekt.name$alter > 25 &
                    langer.Objekt.name$geschlecht=="männlich" |
                    langer.Objekt.name$stifte > 30]
```

	alter	geschlecht	stifte	geburtsort	fahrzeit	podcast
	<int>	<char>	<int>	<char>	<int>	<char>
1:	41	männlich	1	Bonn	60	selten
2:	26	männlich	5	Düsseldorf	40	nie
3:	38	männlich	5	Donezk	57	manchmal
4:	20	weiblich	32	Wesel	89	nie
5:	45	männlich	1	Gelsenkirchen	85	oft

verkürzt sich auf

```
langer.Objekt.name[alter > 25 & geschlecht=="männlich" | stifte > 30]
```

	alter	geschlecht	stifte	geburtsort	fahrzeit	podcast
	<int>	<char>	<int>	<char>	<int>	<char>
1:	41	männlich	1	Bonn	60	selten
2:	26	männlich	5	Düsseldorf	40	nie
3:	38	männlich	5	Donezk	57	manchmal
4:	20	weiblich	32	Wesel	89	nie
5:	45	männlich	1	Gelsenkirchen	85	oft

2.7. Fälle sortieren mit i

Dem Parameter `i` können auch Funktionen übergeben werden. So lassen sich die Daten beispielsweise über die `order()`-Funktion sortieren.

```
# nehme anderen (kürzeren) Datensatz zur Demonstration
dt2[order(Alter)]
```

	Geschlecht	Alter	Gewicht	Groesse
	<char>	<int>	<int>	<int>
1:	w	18	55	174
2:	w	19	74	178
3:	m	21	84	185
4:	m	22	78	184
5:	w	25	74	183
6:	w	26	56	163
7:	w	27	65	169
8:	m	28	80	170
9:	m	29	101	190
10:	m	31	88	189

```
# absteigend
dt2[order(Gewicht, decreasing = TRUE)]
```

	Geschlecht	Alter	Gewicht	Groesse
	<char>	<int>	<int>	<int>
1:	m	29	101	190
2:	m	31	88	189
3:	m	21	84	185
4:	m	28	80	170
5:	m	22	78	184
6:	w	25	74	183
7:	w	19	74	178
8:	w	27	65	169
9:	w	26	56	163
10:	w	18	55	174

2.8. Daten verarbeiten mit j

Nachdem der Datensatz mittels `i` eventuell vorsortiert und -gefiltert wurde, erfolgen die eigentlichen Operationen über den Parameter `j`. So können wir den Mittelwert des Alters der Probanden wie folgt bestimmen:

```
# Mittelwert des Alters
dt[, mean(alter)]
```

```
[1] 25.2973
```

```
# Mittelwert des Alters der Männer
dt[geschlecht == "männlich", mean(alter)]
```

```
[1] 29
```

Innerhalb von `j` kann jede Funktion verwendet werden. So könnten wir überprüfen, ob die Variablen `fahrzeit` und `alter` miteinander korrelieren (ja, das ist quatsch).

```
# korrelieren alter und fahrzeit?
dt[, cor(alter, fahrzeit)]
```

```
[1] 0.1504465
```

Es können auch mehrere Funktionen angewendet werden. Hierfür müssen diese per `list()` an den Parameter `j` übergeben werden. Auf diese Weise könnten wir Median, Mittelwert und Standardabweichung des Alters der Probanden bestimmen.

```
# mehrere Funktionen per list()
dt[, list(Median = median(alter),
          Mittelw = mean(alter),
          Stdabw = sd(alter))]
```

```

Median Mittelw Stdabw
<int> <num> <num>
1: 22 25.2973 6.765373

```

Da der Parameter `j` immer eine Liste erwartet, kann die Funktion `list()` mit einem Punkt abgekürzt werden.

```

# geht auch mit "."
dt[, .(Median = median(alter),
      Mittelw = mean(alter),
      Stdabw = sd(alter),
      InterquA = IQR(alter))]

```

```

Median Mittelw Stdabw InterquA
<int> <num> <num> <num>
1: 22 25.2973 6.765373 6

```

2.9. Daten bearbeiten mit `j`

Über den Parameter `j` können die Daten auch manipuliert werden, ähnlich wie bei der `mutate()`-Funktion des `Tidyverse`. Eine neue Variable kann über die Zeichenkette `:=` definiert werden (dem so genannten *Walrus Operator* (Walross-Operator), der so heisst, weil die Zeichenfolge `:=` an die Stoßzähne eines Walrosses erinnert. Das Logo des `data.table`-Pakets zeigt eine Robbe, was zur humorvollen Verbindung beigetragen hat).

Mit folgendem Aufruf erzeugen wir eine neue Variable `FahrzeitH`, welche die `fahrzeit` in Stunden beinhalten soll.

```

# FahrzeitH in Stunden
dt[, FahrzeitH := fahrzeit/60]

# anzeigen
str(dt)

```

```

Classes 'data.table' and 'data.frame': 37 obs. of 7 variables:
 $ alter      : int 20 28 41 34 26 38 28 21 27 26 ...
 $ geschlecht: chr "weiblich" "weiblich" "männlich" "weiblich" ...
 $ stifte     : int 12 7 1 13 18 25 29 1 2 5 ...
 $ geburtsort: chr "Düren" "Neuss" "Bonn" "Düsseldorf" ...
 $ fahrzeit   : int 1 45 60 25 15 50 40 60 60 40 ...
 $ podcast    : chr "selten" "selten" "selten" "oft" ...
 $ FahrzeitH  : num 0.0167 0.75 1 0.4167 0.25 ...
- attr(*, ".internal.selfref")=externalptr>
- attr(*, "index")= int(0)
..- attr(*, "__geschlecht")= int [1:37] 3 8 10 11 13 31 33 34 37 1 ...

```

So können wir auch mittels der `cut()`-Funktion die Daten klassieren, zum Beispiel das Alter:

```

dt[, alterK := cut(alter, breaks=c(0,20,25,30,40,50),
                  ordered=TRUE)]

# anzeigen
str(dt)

```

```
Classes 'data.table' and 'data.frame': 37 obs. of 8 variables:
 $ alter      : int  20 28 41 34 26 38 28 21 27 26 ...
 $ geschlecht: chr   "weiblich" "weiblich" "männlich" "weiblich" ...
 $ stifte     : int  12 7 1 13 18 25 29 1 2 5 ...
 $ geburtsort: chr   "Düren" "Neuss" "Bonn" "Düsseldorf" ...
 $ fahrzeit   : int   1 45 60 25 15 50 40 60 60 40 ...
 $ podcast    : chr   "selten" "selten" "selten" "oft" ...
 $ FahrzeitH  : num   0.0167 0.75 1 0.4167 0.25 ...
 $ alterK     : Ord.factor w/ 5 levels "(0,20]"<"(20,25]"<...: 1 3 5 4 3 4 3 2 3 3 ...
- attr(*, ".internal.selfref")=<externalptr>
- attr(*, "index")= int(0)
..- attr(*, "__geschlecht")= int [1:37] 3 8 10 11 13 31 33 34 37 1 ...
```

Pro Aufruf kann der Walross-Operator nur einmal verwendet werden. Sollen mehrere Variablen verändert oder hinzugefügt werden, steht die `let()`-Funktion bereit. Innerhalb von `let()` werden wie gewohnt *einfache* Gleichheitszeichen verwendet.

```
# mehrere Manipulationen per let()
dt[, let(geschlecht = factor(geschlecht),
        geburtsort = factor(geburtsort),
        podcast = factor(podcast, ordered=TRUE,
                          levels=c("nie", "selten", "manchmal",
                                    "oft", "immer")))]

# anzeigen
str(dt)
```

```
Classes 'data.table' and 'data.frame': 37 obs. of 8 variables:
 $ alter      : int  20 28 41 34 26 38 28 21 27 26 ...
 $ geschlecht: Factor w/ 2 levels "männlich","weiblich": 2 2 1 2 2 2 2 1 2 1 ...
 $ stifte     : int  12 7 1 13 18 25 29 1 2 5 ...
 $ geburtsort: Factor w/ 26 levels "Bagdad","Bonn",...: 9 21 2 10 8 5 21 7 18 10 ...
 $ fahrzeit   : int   1 45 60 25 15 50 40 60 60 40 ...
 $ podcast    : Ord.factor w/ 5 levels "nie"<"selten"<...: 2 2 2 4 NA 4 4 3 1 1 ...
 $ FahrzeitH  : num   0.0167 0.75 1 0.4167 0.25 ...
 $ alterK     : Ord.factor w/ 5 levels "(0,20]"<"(20,25]"<...: 1 3 5 4 3 4 3 2 3 3 ...
- attr(*, ".internal.selfref")=<externalptr>
- attr(*, "index")= int(0)
```

Die Änderungen wurden direkt im Objekt `dt` gespeichert.

2.10. data.table kopieren

Eine weitere wesentliche Eigenschaft von `data.table`-Objekten besteht darin, dass man sie gesondert kopieren muss. Wir eine `data.table` auf klassischem Wege in ein neues Objekt “kopiert”, so erfolgt keine echte Kopie, sondern lediglich ein *symbolischer Link* auf das ursprüngliche Objekt.

```
# weise dt einem neuen Objekt zu
neu <- dt

str(neu)
```

```
Classes 'data.table' and 'data.frame': 37 obs. of 8 variables:
 $ alter      : int  20 28 41 34 26 38 28 21 27 26 ...
 $ geschlecht: Factor w/ 2 levels "männlich","weiblich": 2 2 1 2 2 2 2 1 2 1 ...
 $ stifte     : int  12 7 1 13 18 25 29 1 2 5 ...
 $ geburtsort: Factor w/ 26 levels "Bagdad","Bonn",...: 9 21 2 10 8 5 21 7 18 10 ...
 $ fahrzeit   : int   1 45 60 25 15 50 40 60 60 40 ...
 $ podcast    : Ord.factor w/ 5 levels "nie"<"selten"<...: 2 2 2 4 NA 4 4 3 1 1 ...
 $ FahrzeitH  : num   0.0167 0.75 1 0.4167 0.25 ...
 $ alterK     : Ord.factor w/ 5 levels "(0,20]"<"(20,25]"<...: 1 3 5 4 3 4 3 2 3 3 ...
 - attr(*, ".internal.selfref")=<externalptr>
 - attr(*, "index")= int(0)
```

Wir haben das Objekt `dt` nur *scheinbar* in das neue Objekt `neu` kopiert. Wenn wir Änderungen am Objekt `neu` vornehmen, so sind diese auch im Objekt `dt` präsent, weil eben **nicht** kopiert, sondern nur ein *Verweis* erstellt wurde.

```
# erstelle neue Variable in "neu"
neu[, kuckuck := fahrzeit * stifte]

# die neue Variable ist auch in "dt" enthalten
str(dt)
```

```
Classes 'data.table' and 'data.frame': 37 obs. of 9 variables:
 $ alter      : int  20 28 41 34 26 38 28 21 27 26 ...
 $ geschlecht: Factor w/ 2 levels "männlich","weiblich": 2 2 1 2 2 2 2 1 2 1 ...
 $ stifte     : int  12 7 1 13 18 25 29 1 2 5 ...
 $ geburtsort: Factor w/ 26 levels "Bagdad","Bonn",...: 9 21 2 10 8 5 21 7 18 10 ...
 $ fahrzeit   : int   1 45 60 25 15 50 40 60 60 40 ...
 $ podcast    : Ord.factor w/ 5 levels "nie"<"selten"<...: 2 2 2 4 NA 4 4 3 1 1 ...
 $ FahrzeitH  : num   0.0167 0.75 1 0.4167 0.25 ...
 $ alterK     : Ord.factor w/ 5 levels "(0,20]"<"(20,25]"<...: 1 3 5 4 3 4 3 2 3 3 ...
 $ kuckuck    : int  12 315 60 325 270 1250 1160 60 120 200 ...
 - attr(*, ".internal.selfref")=<externalptr>
 - attr(*, "index")= int(0)
```

! Dies ist ein häufiger fataler Anfängerfehler, der zum Datenverlust führen kann!

Um das Objekt tatsächlich zu kopieren, muss die Funktion `copy()` verwendet werden.

```
# kopieren dt2 nach neu2
neu2 <- copy(dt2)

# anzeigen
str(neu2)
```

```
Classes 'data.table' and 'data.frame': 10 obs. of 4 variables:
 $ Geschlecht: chr  "m" "w" "w" "m" ...
 $ Alter     : int  28 18 25 29 21 19 27 26 31 22
 $ Gewicht   : int  80 55 74 101 84 74 65 56 88 78
 $ Groesse   : int  170 174 183 190 185 178 169 163 189 184
 - attr(*, ".internal.selfref")=<externalptr>
```

```
# manipulieren
neu2[, Kuckuck := Groesse/Gewicht]

# dt2 ist unverändert
str(dt2)
```

```
Classes 'data.table' and 'data.frame': 10 obs. of 4 variables:
 $ Geschlecht: chr "m" "w" "w" "m" ...
 $ Alter : int 28 18 25 29 21 19 27 26 31 22
 $ Gewicht : int 80 55 74 101 84 74 65 56 88 78
 $ Groesse : int 170 174 183 190 185 178 169 163 189 184
 - attr(*, ".internal.selfref")=<externalptr>
```

2.11. pipen

Innerhalb von `data.table` kann auch die Pipe verwendet werden. Wird die R-Base-Pipe `|>` verwendet, kann mittels Unterstrich `_` auf den weitergeleiteten Datenstrom zugegriffen werden. Bei der Tidyverse-Pipe (eigentlich von `magrittr`) mit der Zeichenfolge `%>%` muss ein Punkt `.` verwendet werden.

Folgende Aufrufe filtern das Geschlecht und pipen den Datenstrom weiter. Anschließend wird nach `alter` sortiert.

```
# Daten pipen mit R_Base
dt2[Geschlecht=="m"] |>
  _[order(Alter)]
```

	Geschlecht	Alter	Gewicht	Groesse
	<char>	<int>	<int>	<int>
1:	m	21	84	185
2:	m	22	78	184
3:	m	28	80	170
4:	m	29	101	190
5:	m	31	88	189

```
# Daten pipen mit magrittr
dt2[Geschlecht=="m"] %>%
  .[order(Alter)]
```

	Geschlecht	Alter	Gewicht	Groesse
	<char>	<int>	<int>	<int>
1:	m	21	84	185
2:	m	22	78	184
3:	m	28	80	170
4:	m	29	101	190
5:	m	31	88	189

Oder wir erstellen ein lineares Modell und pipen es an die `summary()`-Funktion weiter.


```
dt2[, lm(Gewicht ~ Groesse)] |>
  summary()
```

Call:

```
lm(formula = Gewicht ~ Groesse)
```

Residuals:

Min	1Q	Median	3Q	Max
-14.9024	-3.4756	-0.3902	1.0915	15.0732

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	-146.5366	58.3503	-2.511	0.03630 *
Groesse	1.2439	0.3265	3.810	0.00516 **

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 8.992 on 8 degrees of freedom

Multiple R-squared: 0.6447, Adjusted R-squared: 0.6003

F-statistic: 14.51 on 1 and 8 DF, p-value: 0.005164

Wir können den Ausdruck aber auch direkt in die `summary()`-Funktion schreiben.

```
summary(dt2[, lm(Gewicht ~ Groesse)])
```

Call:

```
lm(formula = Gewicht ~ Groesse)
```

Residuals:

Min	1Q	Median	3Q	Max
-14.9024	-3.4756	-0.3902	1.0915	15.0732

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	-146.5366	58.3503	-2.511	0.03630 *
Groesse	1.2439	0.3265	3.810	0.00516 **

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 8.992 on 8 degrees of freedom

Multiple R-squared: 0.6447, Adjusted R-squared: 0.6003

F-statistic: 14.51 on 1 and 8 DF, p-value: 0.005164

2.12. Ergebnisse gruppieren mit `by`

Über den Parameter `by` können die Ergebnisse gruppiert werden.

```
# gruppiert nach Geschlecht
dt[, .(Median = median(alter),
      Mittelw = mean(alter),
      Stdabw = sd(alter)),
  by = geschlecht]
```

	geschlecht	Median	Mittelw	Stdabw
	<fctr>	<num>	<num>	<num>
1:	weiblich	21.5	24.10714	5.251732
2:	männlich	25.0	29.00000	9.617692

Die Ausgabe kann gepipet und weiterverarbeitet werden. In folgendem Beispiel berechnen wir den Variationskoeffizienten (sd/\bar{x}) aus den gruppierten Ergebnissen.

```
dt[, .(Median = median(alter),
      Mittelw = mean(alter),
      Stdabw = sd(alter)),
  by = geschlecht] |>
# berechnen
_[, VK := Stdabw / Mittelw] |>
# anzeigen
_[]
```

	geschlecht	Median	Mittelw	Stdabw	VK
	<fctr>	<num>	<num>	<num>	<num>
1:	weiblich	21.5	24.10714	5.251732	0.2178496
2:	männlich	25.0	29.00000	9.617692	0.3316446

Bitte beachten Sie, dass wir in diesem Beispiel die Anzeige der Endergebnisse mittels `|> _[]` erzwingen mussten. Dies ist notwendig, wenn per `by` gruppierte Ergebnisse weiter manipuliert werden sollen. `Data.table` speichert Änderungen durch `:=` immer direkt im Objekt, wobei keine Ausgabe der Daten erfolgt. Im vorliegenden Fall von `VK := Stdabw / Mittelw` ist diese Speicherung jedoch nicht möglich (ausgegeben wird ja eh nichts), da sich das Endergebnis nicht mehr auf das ursprüngliche Objekt `dt` bezieht. In diesem Fall ist es (sogar) möglich und üblich, das Ergebnis wie gewohnt in einem neuen Objekt zu *speichern*, ohne dass dabei ein symbolischer Link angelegt wird.

```
neu3 <- dt[, .(Median = median(alter),
              Mittelw = mean(alter),
              Stdabw = sd(alter)),
  by = geschlecht] |>
_[, VK := Stdabw / Mittelw] |>
_[]

# anzeigen
neu3
```

	geschlecht	Median	Mittelw	Stdabw	VK
	<fctr>	<num>	<num>	<num>	<num>
1:	weiblich	21.5	24.10714	5.251732	0.2178496
2:	männlich	25.0	29.00000	9.617692	0.3316446

Wir können den letzten Pipevorgang abkürzen, indem wir einfach eckige Klammern [] an unseren Aufruf anhängen.

```
neu4 <- dt[, .(Median = median(alter),
              Mittelw = mean(alter),
              Stdabw = sd(alter)),
            by = geschlecht] |>
  _[, VK := Stdabw / Mittelw][]

# anzeigen
neu4
```

	geschlecht	Median	Mittelw	Stdabw	VK
	<fctr>	<num>	<num>	<num>	<num>
1:	weiblich	21.5	24.10714	5.251732	0.2178496
2:	männlich	25.0	29.00000	9.617692	0.3316446

2.13. Weitere Funktionen aus dem data.table Paket

Das Paket data.table bringt zahlreiche eigene Funktionen mit, um typische Aufgabenstellungen effizienter bearbeiten zu können.

2.13.1. Einzigartige bestimmen mit uniqueN

Um zum Beispiel die Anzahl verschiedener Städte innerhalb der Variable geburtsort zu bestimmen, können wir auf die paketeigene Funktion uniqueN() zurückgreifen:

```
# wieviele unterschiedliche Städte sind in "geburtsort"?
dt[, uniqueN(geburtsort)]
```

```
[1] 26
```

2.13.2. Anzahl der Fälle mit .N

Mit der Funktion .N kann die Anzahl der Fälle ermittelt werden.

```
dt[, .(Anzahl = .N),
     by = geschlecht]
```

	geschlecht	Anzahl
	<fctr>	<int>
1:	weiblich	28
2:	männlich	9

Mit Hilfe von nrow() können so prozentuale Anteile berechnet werden.

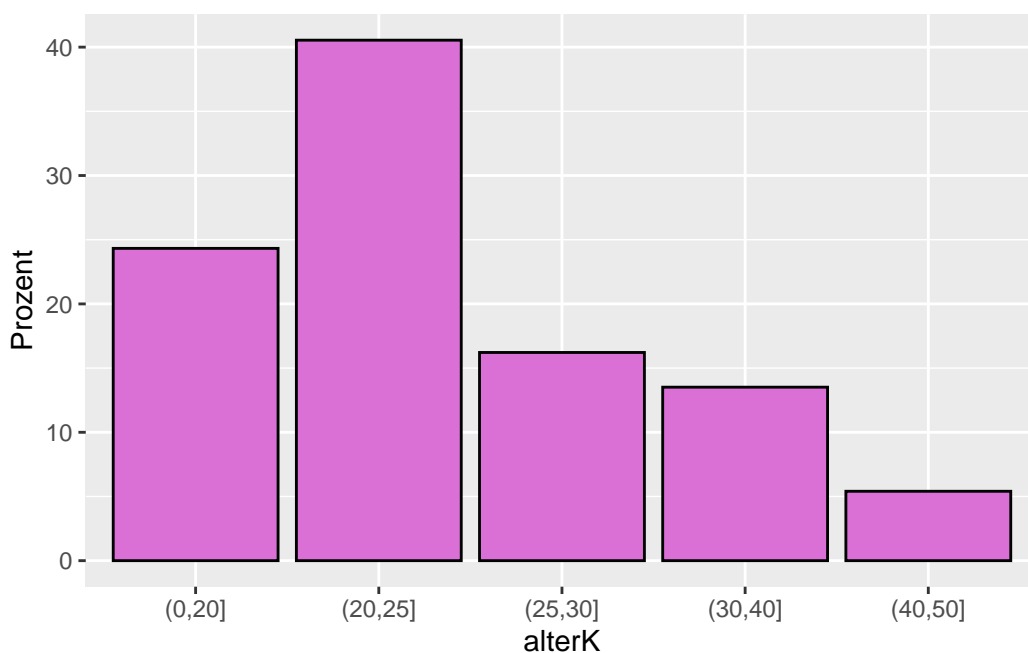
```
dt[, .(Anzahl = .N,
       Prozent = .N/nrow(dt)*100),
     by = alterK]
```

	alterK	Anzahl	Prozent
	<ord>	<int>	<num>
1:	(0,20]	9	24.324324
2:	(25,30]	6	16.216216
3:	(40,50]	2	5.405405
4:	(30,40]	5	13.513514
5:	(20,25]	15	40.540541

Die Ergebnisse können an `ggplot()` weitergereicht werden.

```
# ggplot
library(ggplot2)

dt[, .(Anzahl = .N,
       Prozent = .N/nrow(dt)*100),
     by = alterK] |>
ggplot(aes(x=alterK, y=Prozent)) +
geom_col(color="black", fill="orchid")
```



2.13.3. Lange Tabelle erzeugen mit `melt()`

Mit der Funktion `melt()` können breite Tabellen in lange (tidy) umgewandelt werden, ähnlich wie mit `dplyr::pivot_longer()`. Zur Demonstration verwenden wir die Pfl egetabelle von Isfort (2018)

```
# lade Testdaten
load("https://www.produnis.de/tabletrainer/data/Pflegeberufe.RData")
```

	1999	2001	2003	2005	2007	2009	2011	2013
Krankenpflegeassistenz	16624	19061	19478	21537	27731	36481	46517	54371
Altenpflegehilfe	55770	52710	49727	45776	48326	47903	47978	48363
Kinderkrankenpflege	47779	48203	48822	48519	49080	49307	48291	48937

```

Krankenpflege      430983 436767 444783 449355 457322 465446 468192 472580
Altenpflege        109161 124879 141965 158817 178902 194195 208304 227154
                  2015
Krankenpflegeassistenz 64127
Altenpflegehilfe    49507
Kinderkrankenpflege 48913
Krankenpflege       476416
Altenpflege         246412

```

Die Tabelle ist nicht *tidy* und liegt im breiten Format vor. Ausserdem ist sie von der Klasse `matrix`.

```

# wandeln um in data.table
pf <- as.data.table(Pflegeberufe, keep.rownames = "Berufsgruppe")

# anzeigen
pf

```

```

      Berufsgruppe  1999  2001  2003  2005  2007  2009  2011
      <char> <num> <num> <num> <num> <num> <num> <num>
1: Krankenpflegeassistenz 16624 19061 19478 21537 27731 36481 46517
2:      Altenpflegehilfe 55770 52710 49727 45776 48326 47903 47978
3:      Kinderkrankenpflege 47779 48203 48822 48519 49080 49307 48291
4:      Krankenpflege 430983 436767 444783 449355 457322 465446 468192
5:      Altenpflege 109161 124879 141965 158817 178902 194195 208304
      2013  2015
      <num> <num>
1: 54371 64127
2: 48363 49507
3: 48937 48913
4: 472580 476416
5: 227154 246412

```

Mittels `melt()` transformieren wir `pf` in eine lange (*tidy*) Tabelle. Dabei übergeben wir dem Parameter

- `id.vars` alle Variablen, welche "Identifikatoren" beinhalten. Damit sind alle Spalten gemeint, die keine konkreten Messwerte enthalten, sondern weitere *bezeichnende* Kennwerte. Klassischer Weise sind dies vor allem die **Zeilen**namen, in unserem Falle also `Berufsgruppe`. Es können mehrere `id.vars` mittels `c()` aneinander gereiht werden.
- `measure.vars` alle Spalten, welche die eigentlichen Messwerte enthalten, in unserem Falle 1999:2015 (alles außer `Berufsgruppe`). Wird dieser Parameter leer gelassen, nimmt `data.table` automatisch alle Spalten, die keine `id.vars` sind.
- `variable.name` den Name der neuen Spalte, in welche die Bezeichnungen der `measure.vars` überführt werden sollen, in unserem Fall `Jahr`.
- `value.name` den Name der neuen Spalte, in welche die Werte der `measure.vars` überführt werden sollen, in unserem Fall `Anzahl`.

Da wir alle Spalten außer `Berufsgruppe` *melten* wollen, kann der Parameter `measure.vars` weggelassen werden.

```

# pf mit melt() tidy machen
pf_tidy <- melt(pf, id.vars = "Berufsgruppe",
               variable.name = "Jahr",
               value.name = "Anzahl")

```

```
# anschauen
head(pf_tidy)
```

	Berufsgruppe	Jahr	Anzahl
	<char>	<fctr>	<num>
1:	Krankenpflegeassistenz	1999	16624
2:	Altenpflegehilfe	1999	55770
3:	Kinderkrankenpflege	1999	47779
4:	Krankenpflege	1999	430983
5:	Altenpflege	1999	109161
6:	Krankenpflegeassistenz	2001	19061

2.13.4. Breite Tabelle erzeugen mit dcast()

Mittels `dcast()` können lange Tabellen wieder in breite Tabellen transformiert werden, so wie bei `dplyr::pivot_wider()`.

Der Aufruf folgt der Semantik:

```
dcast(Bezeichner ~ Spaltenname, value.var = "Wertename")
```

wobei

- `Bezeichner` die Spalten der `id.vars` meint.
- `Spaltenname` die Spalte mit der `variable.name` meint.
- `value.var` den Namen der Spalte meint, welche die konkreten Messwerte enthält. Diese muss in Anführungszeichen angegeben werden. Wird dieser Parameter weggelassen, versucht `data.table` die korrekte Spalte zu erraten (was einfach ist, wenn nur noch eine Spalte übrig bleibt).

```
# wandle pf_tidy mit dcast() in breite Tabelle
pf_wide <- dcast(pf_tidy, Berufsgruppe ~ Jahr,
                 value.var = "Anzahl")

# anschauen
head(pf_wide)
```

Key: <Berufsgruppe>

	Berufsgruppe	1999	2001	2003	2005	2007	2009	2011
	<char>	<num>	<num>	<num>	<num>	<num>	<num>	<num>
1:	Altenpflege	109161	124879	141965	158817	178902	194195	208304
2:	Altenpflegehilfe	55770	52710	49727	45776	48326	47903	47978
3:	Kinderkrankenpflege	47779	48203	48822	48519	49080	49307	48291
4:	Krankenpflege	430983	436767	444783	449355	457322	465446	468192
5:	Krankenpflegeassistenz	16624	19061	19478	21537	27731	36481	46517
	2013	2015						
	<num>	<num>						
1:		227154	246412					
2:		48363	49507					
3:		48937	48913					
4:		472580	476416					
5:		54371	64127					

2.13.5. Subset of Data (.SD)

Subset of Data wird in der `data.table`-Syntax verwendet, um eine Teilmenge der Daten in einer speziellen Umgebung zu referenzieren. Die Funktion hierfür heisst `.SD` und enthält eine Auswahl an Spalten der `data.table`, die weiterverarbeitet werden können, z. B. durch Berechnungen oder Transformationen.

`.SD` ist besonders nützlich, wenn Sie Berechnungen oder Transformationen nur auf bestimmte Spalten anwenden möchten, während die anderen Spalten beibehalten werden sollen.

Schauen wir uns nochmal unser Objekt `pf` an.

```
str(pf)
```

```
Classes 'data.table' and 'data.frame':  5 obs. of  10 variables:
 $ Berufsgruppe: chr  "Krankenpflegeassistenz" "Altenpflegehilfe" "Kinderkrankenpflege" "Krankenp
 $ 1999          : num  16624 55770 47779 430983 109161
 $ 2001          : num  19061 52710 48203 436767 124879
 $ 2003          : num  19478 49727 48822 444783 141965
 $ 2005          : num  21537 45776 48519 449355 158817
 $ 2007          : num  27731 48326 49080 457322 178902
 $ 2009          : num  36481 47903 49307 465446 194195
 $ 2011          : num  46517 47978 48291 468192 208304
 $ 2013          : num  54371 48363 48937 472580 227154
 $ 2015          : num  64127 49507 48913 476416 246412
 - attr(*, ".internal.selfref")=<externalptr>
```

Nun verwenden wir `.SD`, um für jede Spalte den Mittelwert zu berechnen.

```
pf[, lapply(.SD, mean)]
```

```
   Berufsgruppe    1999    2001    2003    2005    2007    2009    2011
      <num>      <num>  <num>  <num>  <num>  <num>  <num>
1:      NA 132063.4 136324 140955 144800.8 152272.2 158666.4 163856.4
   2013    2015
      <num>  <num>
1: 170281 177075
```

Der Aufruf `lapply(.SD, mean)` wendet die Funktion `mean` auf jede Spalte in `.SD` an.

2.13.5.1. Verwendung von `.SDcols`

`.SDcols` ist ein optionaler Parameter, mit dem die Spalten, die in `.SD` enthalten sind, gezielt ausgewählt werden können.

Schauen wir uns das Objekt `dt` an.

```
str(dt)
```

Classes 'data.table' and 'data.frame': 37 obs. of 9 variables:

```
$ alter      : int  20 28 41 34 26 38 28 21 27 26 ...
$ geschlecht: Factor w/ 2 levels "männlich","weiblich": 2 2 1 2 2 2 2 1 2 1 ...
$ stifte     : int  12 7 1 13 18 25 29 1 2 5 ...
$ geburtsort: Factor w/ 26 levels "Bagdad","Bonn",...: 9 21 2 10 8 5 21 7 18 10 ...
$ fahrzeit   : int  1 45 60 25 15 50 40 60 60 40 ...
$ podcast    : Ord.factor w/ 5 levels "nie"<"selten"<...: 2 2 2 4 NA 4 4 3 1 1 ...
$ FahrzeitH  : num  0.0167 0.75 1 0.4167 0.25 ...
$ alterK     : Ord.factor w/ 5 levels "(0,20]"<"(20,25]"<...: 1 3 5 4 3 4 3 2 3 3 ...
$ kuckuck    : int  12 315 60 325 270 1250 1160 60 120 200 ...
- attr(*, ".internal.selfref")=<externalptr>
- attr(*, "index")= int(0)
```

Wir können nun den Median für alle mindestens ordinalskalierten Variablen (alter, stifte, fahrzeit) berechnen, indem wir diese Variablen per .SDcols angeben.

```
# Wähle die Spalten aus, die verwendet werden sollen
dt[, lapply(.SD, median), .SDcols=c("alter", "stifte", "fahrzeit")]
```

```
      alter stifte fahrzeit
      <int> <int>   <int>
1:      22      8      60
```

Dies klappt auch mit Gruppierungen. Wir berechnen die Werte erneut, diesmal aber getrennt nach geschlecht.

```
dt[, lapply(.SD, median),
    by=geschlecht, .SDcols=c("alter", "stifte", "fahrzeit")]
```

```
      geschlecht alter stifte fahrzeit
      <fctr> <num>   <num>   <num>
1: weiblich  21.5    12      60
2: männlich  25.0     5      57
```

Auf diese Weise können Spalten auch transformiert werden. Angenommen, wir möchten die Werte für alter und stifte verdoppeln, dann lautet der Aufruf:

```
# Verdopple alter und stifte
dt[, (c("alter", "stifte")) := lapply(.SD, function(x) x*2),
    .SDcols = c("alter", "stifte")]

head(dt)
```

```
      alter geschlecht stifte geburtsort fahrzeit podcast FahrzeitH alterK
      <num>   <fctr>   <num>   <fctr>   <int>   <ord>       <num>   <ord>
1:     40 weiblich    24     Düren      1 selten 0.01666667 (0,20]
2:     56 weiblich    14     Neuss     45 selten 0.75000000 (25,30]
3:     82 männlich     2      Bonn     60 selten 1.00000000 (40,50]
4:     68 weiblich    26 Düsseldorf 25 oft 0.41666667 (30,40]
5:     52 weiblich    36 Duisburg  15 <NA> 0.25000000 (25,30]
6:     76 weiblich    50 Dinslaken  50 oft 0.83333333 (30,40]
```



```
kuckuck
<int>
1:      12
2:     315
3:      60
4:     325
5:     270
6:    1250
```

2.14. Cheat Sheet

Auf GitHub ist ein schöner Cheat-Sheet für `data.table` vorhanden. Das PDF können Sie unter <https://raw.githubusercontent.com/rstudio/cheatsheets/master/datatable.pdf> herunterladen.

Teil I.

Aufgaben

3. Aufgaben

Schön, dass Sie Ihre `data.table`-Fähigkeiten überprüfen und festigen möchten. Bleiben Sie am Ball, Sie schaffen das!

3.1. Aufgabe 3.1 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 eine `data.table` mit den Variablen `Groesse` und `Gewicht`.
- Rechnen Sie das Gewicht um in Kilogramm, und speichern Sie Ihr Ergebnis in der neuen Variable `Kilogramm`.
- Lassen Sie die Daten von Proband 4, 7 und 9 ausgeben.
- Lassen Sie die Daten der Probanden ausgeben, deren Gewicht größer ist als 80kg.
- Lassen Sie die Daten der Probanden ausgeben, die größer als 1,7m sind und leichter als 85kg.
- Speichern Sie Ihr `data.table`-Objekt in die Datei `groegew.csv`. Lassen Sie sich dabei zunächst anzeigen, was in die Datei geschrieben werden wird.

💡 Lösung siehe Abschnitt 4.1

3.2. Aufgabe 3.2 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 eine `data.table` mit dem Namen `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.
 g) Speichern Sie die Objekte `chol` und `male` als Textdatei auf Ihre Festplatte. Lassen Sie sich dabei jeweils zuvor anzeigen, welcher Inhalt in die Textdatei geschrieben werden wird.

 Lösung siehe Abschnitt 4.2

3.3. Aufgabe 3.3 Big Five

i `{data.table}` ist vor allem bei großen Datensätzen beliebt, da es schneller ist als *herkömmliches* R. Die Datei `big_five_scores.csv` enthält Daten von 307.313 Probanden aus 236 Ländern zu den *Big Five* der Persönlichkeitspsychologie, siehe [https://de.wikipedia.org/wiki/Big_Five_\(Psychologie\)](https://de.wikipedia.org/wiki/Big_Five_(Psychologie)). Die Datei liegt als ZIP-Paket unter https://www.produnis.de/tabletrainer/data/big_five.zip. Laden Sie die ZIP Datei herunter, und entpacken Sie `big_five_scores.csv` in Ihren Projektordner (bzw. dort in den `data`-Ordner).

! Der Datensatz beinhaltet folgende Variablen

- `case_id`: Eindeutige ID der Person, zu der die Ergebnisse gehören
- `country`: Herkunftsland der Person
- `age`: Alter der Person
- `sex`: biologisches Geschlecht der Person. 1 = männlich, 2 = weiblich

Die restlichen Spalten sind die Punktzahlen der Person von 0 bis 1 für jede ihrer fünf großen Persönlichkeitsmerkmale:

- `Agreeableness` (Verträglichkeit)
- `Extraversion` (Extraversion)
- `Openness` (Offenheit)
- `Conscientiousness` (Gewissenhaftigkeit)
- `Neuroticism` (Neurotizismus).


- a) Lesen Sie den Datensatz `big_five_scores.csv` als `data.table` in Ihre R-Session und machen Sie sich mit dem Datensatz vertraut.
- b) Ändern Sie die Geschlechtskodierung, so dass `männlich` und `weiblich` verwendet werden.
- c) Passen Sie das Skalenniveau der Variablen an.

💡 Lösung siehe Abschnitt 4.3

3.4. Aufgabe 3.4 Rolling Stone Magazine

i Die Datei `rolling_stone.csv` enthält die *500 Greatest Albums of All Time* Listen des Rolling Stone Magazines aus den Jahren 2003, 2012 und 2020. Der Datensatz stammt aus dem *Tidy Tuesday* Projekt (2022) vom 07.05.2024 und kann auch unter https://www.produis.de/tabletrainer/data/rolling_stone.csv heruntergeladen werden. Die Aufgaben sind inspiriert von Corrales & Campitelli (2024).


- a) Laden Sie die Datei `rolling_stone.csv` als `data.table` in Ihre R-Session und machen Sie sich mit dem Datensatz vertraut.
- b) Passen Sie das Skalenniveau der Variablen an.
- c) Welche sind die Nummer 1 Alben der Jahre 2003, 2012 und 2020?
- d) Ist Ihre Lieblingsband in der Liste?
- e) Welche weiblichen Bands haben mehr als 3 Mitglieder?
- f) Welche Band hat die meisten Alben im Datensatz?
- g) Prüfen Sie per Korrelationsverfahren, ob die Beliebtheit bei Spotify (`spotify_popularity`) mit der Liste von 2020 übereinstimmt.
- h) Welchen durchschnittlichen Rang erzielen Alben des Genres "Electronic" in den Jahren 2003, 2012 und 2020?
- i) Berechnen Sie das arithmetische Mittel und den Median des Albenrankings für jedes Genre im Jahr 2020. Wieviele Alben sind pro Genre enthalten? Sortieren Sie die Ausgabe einmal absteigend nach dem Median, und einmal aufsteigend nach `genre`.
- j) Manche Künstler haben es in jede der 3 Listen geschafft. Wie groß ist die Anzahl an Bands, die in jeder der 3 Listen vertreten sind, wieviele Alben haben in jeder der 3 Listen eine Platzierung, welche Alben sind in jeder der 3 Listen auf dem selben Platz, welche haben sich kontinuierlich verbessert, welche kontinuierlich verschlechtert?
- k) Erzeugen Sie eine neue Variable `soloband`, in welcher festgehalten wird, ob es sich um eine(n) Solo-künstler(in) handelt (`solo`), oder um eine Band (`band`).
- l) Der Datensatz liegt als `wide.table` vor, da die Rankings für 2003, 2012 und 2020 als Variablen nebeneinander stehen. Wandeln Sie den Datensatz in eine `long.table` (Tidy Data) um, so dass die Rankingangaben in den Variablen `Rang` und `Rangjahr` angegeben sind.
- m) Plotten Sie mittels `ggplot()` die Rangveränderungen von 2003 bis 2020 für solche Alben, die sich kontinuierlich verschlechter haben. Was fällt Ihnen auf?

 Lösung siehe Abschnitt 4.4

3.5. Aufgabe 3.5 Taylor Swift

i Im Datensatz `taylor_swift_spotify2024.csv` (siehe https://www.produnis.de/tabletrainer/data/taylor_swift_spotify2024.csv) sind Daten von Taylor Swift bei Spotify enthalten (Stand 2024).

- Laden Sie den Datensatz `taylor_swift_spotify2024.csv` als `data.table` in Ihre R-Session. Nennen Sie Ihr Objekt dabei `ts` und verschaffen Sie sich mittels `str()` einen Überblick über die enthaltenen Daten.
- Wenn nötig, korrigieren Sie das Skalenniveau (nominal, ordinal, metrisch) der Variablen innerhalb des Datensatzes.
- Erstellen Sie eine neue Variable `sekunden`, welche die Songlängen in Sekunden enthält
- Wie lang dauern die Songs im Durchschnitt? Bei welcher Songlänge liegt der Median?
- Welcher Song ist laut Datensatz der populärste, welcher der längste, und welcher der langsamste insgesamt? Stellen Sie anschließend die Werte pro Album dar.
- Welches Album hat die meisten Songs, und welches hat die wenigsten Songs?
- Plotten Sie die Anzahl der Tracks pro Album als Punkt-Liniendiagramm, wobei das Datum auf der X-Achse, und die Trackanzahl auf der Y-Achse dargestellt werden.


 Lösung siehe Abschnitt 4.5

3.6. Aufgabe 3.6 Anscombe-Quartett

i Das Anscombe-Quartett ist ein bekannter Datensatz in der Statistik, der von Francis Anscombe (1973) erdacht wurde, siehe <https://de.wikipedia.org/wiki/Anscombe-Quartett>.

Der dazugehörige Datensatz ist in der R-Standardinstallation bereits implementiert und heisst `anscombe`.


- Laden Sie den Datensatz `anscombe` in Ihre R-Session und überführen Sie ihn in ein `data.table` Objekt mit dem Namen `ac`.
- Die Daten liegen als *breite Tabelle* (wide tabel) vor. Überführen Sie sie ins *long table* (tidy) Format, so dass Ihre `data.table` aus den Spalten `x`, `y`, und `Gruppe` besteht.
- Berechnen Sie für jede Gruppe die Mittelwerte, Standardabweichungen, Korrelations- und Regressionskoeffizienten von `x` und `y`, wobei Sie Ihre Ergebnisse auf 2 Stellen runden sollen.
- Erzeugen Sie mittels `ggplot()` eine Punktwolke mit Regressionsgeraden für jede Gruppe, wobei alle 4 Diagramme mit einem Plotaufruf erzeugt werden sollen.

 Lösung siehe Abschnitt 4.6

3.7. Aufgabe 3.7 Neugeborene

i Der Datensatz `neonates` von `rk.Teaching`¹, auch verfügbar unter <https://www.produnis.de/tabletrainer/data/neonates.RData>, enthält Informationen über eine Stichprobe von 320 Neugeborenen, die im Laufe eines Jahres nach normaler Schwangerschaftsdauer geboren wurden.

- a) Überführen Sie die Daten in ein `data.table`-Objekt mit dem Namen `ng`.
- b) Die Variabel `apgar1` enthält die APGAR-Scores nach 1 Minute. Wenn ein Score von 3 oder weniger anzeigt, dass das Neugeborene in einem kritischen Zustand ist, wie viel Prozent der Neugeborenen in der Stichprobe sind dann in einem kritischen Zustand?
- c) Erstellen Sie die Häufigkeitstabelle des Geburtsgewichts der Neugeborenen, indem Sie die Daten in Klassen mit einer Breite von 0,5 kg von 2 bis 4,5 kg einteilen. Welches Intervall enthält die meisten Neugeborenen?
- d) Vergleichen Sie die Häufigkeitsverteilung des APGAR-Scores nach 1 Minute für Mütter unter 20 Jahren und für Mütter über 20 Jahren. Welche Gruppe hat mehr Neugeborene in kritischem Zustand?
- e) Vergleichen Sie die relative Häufigkeitsverteilung des Geburtsgewichts der Neugeborenen, je nachdem, ob die Mutter während der Schwangerschaft geraucht hat (`smoke`) oder nicht. Wenn ein Gewicht unter 2,5 kg als niedriges Gewicht gilt, welche Gruppe hat einen höheren Prozentsatz an Neugeborenen mit niedrigem Gewicht?
- f) Berechnen Sie die Prävalenz von Neugeborenen mit niedrigem Gewicht für Mütter, die vor der Schwangerschaft geraucht haben (`smoke.before`), und den Nichtraucherinnen.
- g) Berechnen Sie die Odds Ratio eines niedrigen Geburtsgewichts des Neugeborenen, wenn die Mutter während der Schwangerschaft raucht, im Vergleich dazu, wenn die Mutter nicht raucht.
- h) Erstellen Sie das Balkendiagramm der kumulierten relativen Häufigkeit des APGAR-Scores nach 1 Minute. Unter welchem Wert liegen die Hälfte der Neugeborenen?
- i) Vergleichen Sie die Balkendiagramme der relativen Häufigkeitsverteilungen des APGAR-Scores nach 1 Minute, je nachdem, ob die Mutter während der Schwangerschaft geraucht hat oder nicht. Welche Schlussfolgerungen können gezogen werden?
- j) Berechnen Sie Median, Durchschnitt und Standardabweichung für die APGAR-Scores nach 1 und nach 5 Minuten jeweils für die Kinder von Müttern, die vor der Schwangerschaft geraucht haben, und den Nichtraucherinnen. Geben Sie auch die Anzahl an Fällen (N) an. Bewerten Sie die Ergebnisse.
- k) Ist der Unterschied der APGAR-Scores aus Aufgabe j) signifikant?


 Lösung siehe Abschnitt 4.7

¹<https://github.com/rkward-community/rk.Teaching>

3.8. Aufgabe 3.8 Verteidigung gegen die dunklen Künste

i In den Textdateien `VgddK_Lupin.txt`, `VgddK_Moody.txt` und `VgddK_Umbridge.txt` (verfügbar unter <https://www.produnis.de/tabletrainer/data/>) sind die Leistungspunkte (von 1 bis 10) von 25 Schüler:innen aus Hogwarts im Fach *Verteidigung gegen die dunklen Künste* enthalten, wobei diese Punkte jeweils für die Professoren Lupin, Moody und Umbridge erhoben wurde.


- Laden Sie die Textdateien als `data.table` in Ihre R-Session, und führe Sie diese zu einem einzelnen `data.table`-Objekt mit dem Namen `hp` zusammen.
- Erstellen Sie mit `ggplot()` ein Diagramm, welches die Leistungspunkte als Boxplots für jeden Professor darstellt. Hierfür bietet es sich an, die Daten ins `long table` Format zu überführen.

 Lösung siehe Abschnitt 4.8

3.9. Aufgabe 3.9 Hogwarts Hauspunkte

i In der Datei `PotterHauspunkte.RData` (verfügbar unter <https://www.produnis.de/tabletrainer/data/PotterHauspunkte.RData>) sind die Hauspunkte erfasst, die 3.273 Schüler in ihrem ersten, dritten und fünften Schuljahr an Hogwarts für ihr Haus gesammelt haben.

- Laden Sie den Datensatz von <https://www.produnis.de/tabletrainer/data/PotterHauspunkte.RData> in Ihre R-Session, und überführen Sie ihn in eine `data.table` mit dem Namen `pp`.
- Berechnen Sie Median, Mittelwert und Standardabweichung für die Hauspunkte insgesamt, und jeweils für jedes Haus und Jahr gesondert.
- Plotten Sie die Punkte als Boxplots in Abhängigkeit zum Schuljahr, und dann in Abhängigkeit zum Haus.


 Lösung siehe Abschnitt 4.9

3.10. Aufgabe 3.10 Lungenkapazität

i Tager et al. (1983) haben die Auswirkungen des mütterlichen Zigarettenrauchens auf die Lungenfunktion in einer Kohorte von Kindern und Jugendlichen untersucht, die über einen Zeitraum von sieben Jahren prospektiv beobachtet wurden. Dabei wurde auch erfasst, ob die Kinder selbst rauchen oder nicht. Die dazugehörigen Daten stehen unter anderem im GLMsData-Zusatzpaket unter dem Namen `lungcap` zur Verfügung. Im Datensatz beschreibt `FEV` das forcierte expiratorische Volumen in Litern, ein Maß für die Lungenkapazität. Die Variable `Ht` beschreibt die Körpergröße der Probanden in Zoll. Ob die Kinder selbst auch rauchen, ist in der Variable `Smoke` erfasst.

- Laden Sie den Datensatz `lungcap` als `data.table` mit dem Namen `lc` in Ihre R-Session
- Erzeugen Sie eine neue Variable Körpergröße, welche die Körpergröße in Zentimetern enthält (1 Zoll = 2,54cm)
- Ändern Sie die Kodierung der Variable `Smoke`, so dass statt 0 “nein”, und statt 1 “ja” enthalten ist. Passen Sie dabei auch das Skalenniveau an.
- Plotten Sie nebeneinander die Boxplots der Lungenkapazität nichtrauchenden und rauchenden Kindern. Legt das Diagramm einen Zusammenhang nahe?
- Führen Sie einen Signifikanztest durch, um zu überprüfen, ob sich die Lungenkapazitäten in Abhängigkeit zu `Smoke` unterscheidet.
- Erzeugen Sie eine Punktwolke des Lungenvolumens und des Alters, sowie des Lungenvolumens und der Körpergröße. Legen die Diagramme einen Zusammenhang nahe?
- Welches Regressionsmodell ist am besten geeignet, um `FEV` erklärt durch `Alter` zu bestimmen, und welches ist am besten geeignet, um `FEV` erklärt durch Körpergröße zu bestimmen?
- Berechnen Sie das Modell, welches `FEV` am besten erklärt.
- Plotten Sie eine Punktwolke, mit `FEV` auf der Y-Achse, und dem besten Prädiktor auf der X-Achse. Färben Sie die Daten mittels der Variable `Smoke`. Fügen Sie anschließend Ihre Modelllinie dem Plot hinzu.
- Fügen Sie `Smoke`, `Age` und `Gender` als weitere Prädiktor dem Modell hinzu. Hat Rauchen einen Einfluss auf `FEV`?

Weitere Informationen zur Auswertungsstrategie finden sich bei Kahn (2005).

 Lösung siehe Abschnitt 4.10

Teil II.

Lösungswege

4. Lösungswege

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

💡 Cheat Sheet

Auf GitHub ist ein schöner Cheat-Sheet für `data.table` vorhanden, der bei der Lösung der Aufgaben hilfreich sein könnte. Das PDF können Sie unter <https://raw.githubusercontent.com/rstudio/cheatsheets/master/datatable.pdf> herunterladen.

4.1. Lösung zur Aufgabe 3.1 Größe und Gewicht

💡 a) Überführen Sie die Daten in eine `data.table` mit den Variablen `Groesse` und `Gewicht`.

```
# Paket aktivieren
library(data.table)

# Überführe in eine data.table
dt <- data.table(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
dt
```

	Groesse	Gewicht
	<num>	<num>
1:	1.68	78500
2:	1.87	110100
3:	1.95	97500
4:	1.74	69200
5:	1.80	82500
6:	1.75	71500
7:	1.59	81500
8:	1.77	87200
9:	1.82	75500
10:	1.74	65500

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

```
# Umrechnen in Kilogramm
dt[, Kilogramm := Gewicht / 1000]

# anzeigen
dt
```

	Groesse	Gewicht	Kilogramm
	<num>	<num>	<num>
1:	1.68	78500	78.5
2:	1.87	110100	110.1
3:	1.95	97500	97.5
4:	1.74	69200	69.2
5:	1.80	82500	82.5
6:	1.75	71500	71.5
7:	1.59	81500	81.5
8:	1.77	87200	87.2
9:	1.82	75500	75.5
10:	1.74	65500	65.5

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

```
# Ausgabe der Daten von Proband 4, 7 und 9
dt[c(4, 7, 9)]
```

	Groesse	Gewicht	Kilogramm
	<num>	<num>	<num>
1:	1.74	69200	69.2
2:	1.59	81500	81.5
3:	1.82	75500	75.5

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

```
dt[Kilogramm > 80]
```

	Groesse	Gewicht	Kilogramm
	<num>	<num>	<num>
1:	1.87	110100	110.1
2:	1.95	97500	97.5
3:	1.80	82500	82.5
4:	1.59	81500	81.5
5:	1.77	87200	87.2

💡 e) Lassen Sie die Daten der Probanden ausgeben, die größer als 1,7m sind und leichter als 85kg

```
dt[Groesse > 1.7 & Kilogramm < 85]
```

	Groesse	Gewicht	Kilogramm
	<num>	<num>	<num>
1:	1.74	69200	69.2
2:	1.80	82500	82.5
3:	1.75	71500	71.5
4:	1.82	75500	75.5
5:	1.74	65500	65.5

💡 f) Speichern Sie Ihr data.table-Objekt in die Datei groegew.csv. Lassen Sie sich dabei zunächst anzeigen, was in die Datei geschrieben werden wird.

```
# zeige, was in die Datei gespeichert würde
fwrite(dt)
```

```
Groesse,Gewicht,Kilogramm
1.68,78500,78.5
1.87,110100,110.1
1.95,97500,97.5
1.74,69200,69.2
1.8,82500,82.5
1.75,71500,71.5
1.59,81500,81.5
1.77,87200,87.2
1.82,75500,75.5
1.74,65500,65.5
```

```
# speichere in Datei groegew.csv
fwrite(dt, "groegew.csv")
```

4.2. Lösung zur Aufgabe 3.2 Datentabelle

💡 a) Übertragen Sie die Daten in eine `data.table` mit dem Namen `chol`.

```
# übertrage die Daten
chol <- data.table(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
	<char>	<char>	<num>	<num>	<num>
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

```
# Alter der Probanden
alter <- c(18, 32, 24, 35, 46, 68)

# Neue Spalte 'Alter'
chol[, Alter := alter]

# Spalte 'Alter' zwischen `Name` und `Geschlecht`
setcolororder(chol, c("Name", "Alter", "Geschlecht", "Gewicht",
                     "Größe", "Cholesterol"))

# Ausgabe der data.table
chol
```

	Name	Alter	Geschlecht	Gewicht	Größe	Cholesterol
	<char>	<num>	<char>	<num>	<num>	<num>
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

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

# mit rbind zusammenbringen
chol <- rbind(chol, neu)

# anzeigen
chol
```

	Name	Alter	Geschlecht	Gewicht	Größe	Cholesterol
	<char>	<num>	<char>	<num>	<num>	<num>
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 berechnen
chol[, BMI := Gewicht / (Größe / 100)^2]

# anzeigen
chol
```

	Name	Alter	Geschlecht	Gewicht	Größe	Cholesterol	BMI
	<char>	<num>	<char>	<num>	<num>	<num>	<num>
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 wie folgt klassieren

Hierzu können wir entweder die `fifelse()`-Funktion nutzen...

```
# Klassifizieren mit fifelse
chol[, Adipositas := fifelse(BMI < 18.5, "Untergewicht",
                             fifelse(BMI >= 18.5 & BMI < 24.5, "Normalgewicht",
                             fifelse(BMI >= 24.5 & BMI <= 30, "Übergewicht", "Adipositas")))]

# anzeigen
chol
```

	Name	Alter	Geschlecht	Gewicht	Größe	Cholesterol	BMI
	<char>	<num>	<char>	<num>	<num>	<num>	<num>
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
	<char>
1:	Übergewicht
2:	Adipositas
3:	Normalgewicht
4:	Normalgewicht
5:	Normalgewicht
6:	Adipositas
7:	Übergewicht

...oder mittels `cut()`.

```
# Klassifizieren mit cut()
chol[, Adipositas := cut(BMI,
                          breaks = c(-Inf, 18.5, 24.5, 30, Inf),
                          labels = c("Untergewicht", "Normalgewicht",
                          "Übergewicht", "Adipositas"))]

# anzeigen
chol
```

	Name	Alter	Geschlecht	Gewicht	Größe	Cholesterol	BMI
	<char>	<num>	<char>	<num>	<num>	<num>	<num>
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
	<fctr>

```

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.

```
male <- chol[Geschlecht == "M"]
```

```
# anzeigen
male
```

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

	Adipositas
	<fctr>
1:	Adipositas
2:	Normalgewicht
3:	Adipositas
4:	Übergewicht

💡 g) Speichern Sie die Objekte `chol` und `male` als Textdatei auf Ihre Festplatte. Lassen Sie sich dabei jeweils zuvor anzeigen, welcher Inhalt in die Textdatei geschrieben werden wird.

```
# zeige, was in chol.txt gespeichert würde
fwrite(chol)
```

```

Name,Alter,Geschlecht,Gewicht,Größe,Cholesterol,BMI,Adipositas
Anna Tomie,18,W,85,179,182,26.528510346119,Übergewicht
Bud Zillus,32,M,115,173,232,38.4242707741655,Adipositas
Dieter Mietenplage,24,M,79,181,191,24.1140380330271,Normalgewicht
Hella Scheinwerfer,35,W,60,170,200,20.7612456747405,Normalgewicht
Inge Danken,46,W,57,158,148,22.8328793462586,Normalgewicht
Jason Zufall,68,M,96,174,249,31.7082837891399,Adipositas
Mitch Mackes,44,M,92,178,220,29.0367377856331,Übergewicht

```

```
# zeige, was in male.txt gespeichert würde
fwrite(male)
```

```
Name,Alter,Geschlecht,Gewicht,Größe,Cholesterol,BMI,Adipositas
```

```
Bud Zillus,32,M,115,173,232,38.4242707741655,Adipositas
Dieter Mietenplage,24,M,79,181,191,24.1140380330271,Normalgewicht
Jason Zufall,68,M,96,174,249,31.7082837891399,Adipositas
Mitch Mackes,44,M,92,178,220,29.0367377856331,Übergewicht
```

```
# speichere in Datei chol.txt
fwrite(chol, "chol.txt")

# speichere in Datei male.txt
fwrite(male, "male.txt")
```

4.3. Lösung zur Aufgabe 3.3 Big Five

💡 a) Lesen Sie den Datensatz `big_five_scores.csv` als `data.table` in Ihre R-Session.

Zunächst laden wir die Datei von https://www.produnis.de/tabletrainer/big_five.zip herunter und legen sie im `data`-Ordner (siehe Abschnitt 1.1) ab.

Wenn Sie dies bereits getan haben und die Datei `big_five_scores.csv` bereits entpackt ist, lautet der Befehl zum Einlesen der Daten:

```
# lese Daten ein
big5 <- fread("data/big_five_scores.csv")
```

Wir können die Datei aber auch direkt in R herunterladen und entpacken.

```
# File herunterladen und im "data" Ordner speichern
download.file("https://www.produnis.de/tabletrainer/data/big_five.zip",
             destfile = "data/testbig_five.zip")

# entpacken in temporäres Verzeichnis
unzip("data/big_five.zip",
     files = "big_five_scores.csv",
     exdir = tempdir())

# speichere Pfad auf temporäre Datei
pfad <- file.path(tempdir(), "big_five_scores.csv")
```

Der Befehl zum Einlesen aus der temporären Datei lautet

```
fread(pfad)
```

Falls Sie das unzip Programm auf Ihrem PC installiert haben, können Sie direkt auf das ZIP-Paket zugreifen

```
fread(cmd = 'unzip -p data/big_five.zip big_five_scores.csv')
```

Wenn das ZIP-Paket nur eine Datei enthält, muss dessen Dateiname nicht extra angegeben werden.


```
fread(cmd = 'unzip -p data/big_five.zip')
```

Mit dem Datensatz vertraut machen:

```
str(big5)
```

Classes 'data.table' and 'data.frame': 307313 obs. of 9 variables:

```
$ case_id      : int  1 3 4 5 6 7 8 9 10 11 ...
$ country      : chr  "South Afri" "UK" "USA" "UK" ...
$ age          : int  24 24 36 19 17 17 28 28 18 17 ...
$ sex          : int  1 2 2 1 1 1 2 2 1 1 ...
$ agreeable_score : num  0.753 0.733 0.88 0.69 0.6 ...
$ extraversion_score : num  0.497 0.68 0.77 0.617 0.713 ...
$ openness_score  : num  0.803 0.787 0.86 0.717 0.647 ...
$ conscientiousness_score: num  0.887 0.747 0.897 0.637 0.633 ...
$ neuroticism_score : num  0.427 0.59 0.297 0.563 0.513 ...
- attr(*, ".internal.selfref")=<externalptr>
```

 b) Ändern Sie die Geschlechtskodierung, so dass männlich und weiblich verwendet werden.

```
# Ändere die Geschlechtskodierung
big5[, sex := fifelse(sex == 1, "männlich", "weiblich")]

str(big5)
```

Classes 'data.table' and 'data.frame': 307313 obs. of 9 variables:

```
$ case_id      : int  1 3 4 5 6 7 8 9 10 11 ...
$ country      : chr  "South Afri" "UK" "USA" "UK" ...
$ age          : int  24 24 36 19 17 17 28 28 18 17 ...
$ sex          : chr  "männlich" "weiblich" "weiblich" "männlich" ...
$ agreeable_score : num  0.753 0.733 0.88 0.69 0.6 ...
$ extraversion_score : num  0.497 0.68 0.77 0.617 0.713 ...
$ openness_score  : num  0.803 0.787 0.86 0.717 0.647 ...
$ conscientiousness_score: num  0.887 0.747 0.897 0.637 0.633 ...
$ neuroticism_score : num  0.427 0.59 0.297 0.563 0.513 ...
- attr(*, ".internal.selfref")=<externalptr>
```

💡 c) Passen Sie das Skalenniveau der Variablen an.

```
# case_id und country sind nominale Variablen
big5[, let(case_id = factor(case_id),
           country = factor(country),
           sex      = factor(sex)
        )]
```

```
str(big5)
```

Classes 'data.table' and 'data.frame': 307313 obs. of 9 variables:

```
$ case_id      : Factor w/ 307313 levels "1","3","4","5",...: 1 2 3 4 5 6 7 8 9 10 ...
$ country      : Factor w/ 236 levels "", "Afghanista",...: 190 216 220 216 216 220 22
$ age          : int   24 24 36 19 17 17 28 28 18 17 ...
$ sex          : Factor w/ 2 levels "männlich", "weiblich": 1 2 2 1 1 1 2 2 1 1 ...
$ agreeable_score : num  0.753 0.733 0.88 0.69 0.6 ...
$ extraversion_score : num  0.497 0.68 0.77 0.617 0.713 ...
$ openness_score   : num  0.803 0.787 0.86 0.717 0.647 ...
$ conscientiousness_score : num  0.887 0.747 0.897 0.637 0.633 ...
$ neuroticism_score : num  0.427 0.59 0.297 0.563 0.513 ...
- attr(*, ".internal.selfref")=<externalptr>
```

4.4. Lösung zur Aufgabe 3.4 Rolling Stone Magazine

💡 a) Laden Sie die Datei `rolling_stone.csv` als `data.table` in Ihre R-Session und machen Sie sich mit dem Datensatz vertraut.

```
# falls schon gedownloadet
rs <- fread("data/rolling_stone.csv")

# per URL einlesen
rs <- fread("https://www.produnis.de/tabletrainer/data/rolling_stone.csv")
```

```
# anschauen
str(rs)
```


Classes 'data.table' and 'data.frame': 691 obs. of 21 variables:

```
$ sort_name      : chr  "Sinatra, Frank" "Diddley, Bo" "Presley, Elvis" "Sinatra, Fra
$ clean_name     : chr  "Frank Sinatra" "Bo Diddley" "Elvis Presley" "Frank Sinatra"
$ album          : chr  "In the Wee Small Hours" "Bo Diddley / Go Bo Diddley" "Elvis
$ rank_2003      : int   100 214 55 306 50 NA NA 421 NA 12 ...
$ rank_2012      : int   101 216 56 308 50 NA 451 420 NA 12 ...
$ rank_2020      : int   282 455 332 NA 227 32 33 NA 68 31 ...
$ differential   : int   -182 -241 -277 -195 -177 469 468 -80 433 -19 ...
$ release_year   : int   1955 1955 1956 1956 1957 2016 2006 1957 1985 1959 ...
$ genre          : chr  "Big Band/Jazz" "Rock n' Roll/Rhythm & Blues" "Rock n' Roll/R
$ type           : chr  "Studio" "Studio" "Studio" "Studio" ...
```

```

$ weeks_on_billboard      : int   14 NA 100 NA 5 87 173 NA 27 NA ...
$ peak_billboard_position : int    2 201 1 2 13 1 2 201 30 201 ...
$ spotify_popularity      : int   48 50 58 62 64 73 67 47 75 52 ...
$ spotify_url             : chr    "spotify:album:3GmwKB1tgPZgXeRJZSm9WX" "spotify:album:1cbtDEwxCjMhg1b490gNBR" "7GXP50hYyP
$ artist_member_count     : int    1 1 1 1 1 1 1 4 1 1 ...
$ artist_gender           : chr    "Male" "Male" "Male" "Male" ...
$ artist_birth_year_sum   : int   1915 1928 1935 1915 1932 1981 1983 7752 1958 1926 ...
$ debut_album_release_year : int   1946 1955 1956 1946 1957 2003 2003 1957 1978 1951 ...
$ ave_age_at_top_500      : num   40 27 21 41 25 35 23 19 27 33 ...
$ years_between           : int    9 0 0 10 0 13 3 0 7 8 ...
$ album_id               : chr    "3GmwKB1tgPZgXeRJZSm9WX" "1cbtDEwxCjMhg1b490gNBR" "7GXP50hYyP
- attr(*, ".internal.selfref")=<externalptr>

```

 b) Passen Sie das Skalenniveau der Variablen an.

Es gibt einige kategoriale Variablen im Datensatz.

```

rs[, let(sort_name = factor(sort_name),
      clean_name = factor(clean_name),
      album = factor(album),
      genre = factor(genre),
      type = factor(type),
      artist_gender = factor(artist_gender),
      album_id = factor(album_id),
      spotify_url = factor(spotify_url)
    )]

# anschauen
str(rs)

```

Classes 'data.table' and 'data.frame': 691 obs. of 21 variables:

```

$ sort_name      : Factor w/ 391 levels "2Pac","50 Cent",...: 315 92 268 315 189 25 37
$ clean_name     : Factor w/ 386 levels "2Pac","50 Cent",...: 114 40 100 114 185 27 15
$ album         : Factor w/ 685 levels "\"\"Love and Theft\"\"\"",...: 293 101 197 503
$ rank_2003      : int   100 214 55 306 50 NA NA 421 NA 12 ...
$ rank_2012      : int   101 216 56 308 50 NA 451 420 NA 12 ...
$ rank_2020      : int   282 455 332 NA 227 32 33 NA 68 31 ...
$ differential   : int  -182 -241 -277 -195 -177 469 468 -80 433 -19 ...
$ release_year   : int   1955 1955 1956 1956 1957 2016 2006 1957 1985 1959 ...
$ genre         : Factor w/ 17 levels "", "Afrobeat",...: 3 15 15 3 1 1 17 15 1 3 ...
$ type          : Factor w/ 5 levels "Compilation",...: 5 5 5 5 5 5 5 5 5 5 ...
$ weeks_on_billboard : int   14 NA 100 NA 5 87 173 NA 27 NA ...
$ peak_billboard_position : int    2 201 1 2 13 1 2 201 30 201 ...
$ spotify_popularity : int   48 50 58 62 64 73 67 47 75 52 ...
$ spotify_url     : Factor w/ 656 levels "", "6Mj0v3BeIjmht2ymtRih3s",...: 280 105 623 3
$ artist_member_count : int    1 1 1 1 1 1 1 4 1 1 ...
$ artist_gender    : Factor w/ 4 levels "", "Female", "Male",...: 3 3 3 3 3 2 2 3 2 3 ...
$ artist_birth_year_sum : int   1915 1928 1935 1915 1932 1981 1983 7752 1958 1926 ...
$ debut_album_release_year : int   1946 1955 1956 1946 1957 2003 2003 1957 1978 1951 ...
$ ave_age_at_top_500 : num   40 27 21 41 25 35 23 19 27 33 ...

```

```
$ years_between      : int  9 0 0 10 0 13 3 0 7 8 ...
$ album_id           : Factor w/ 691 levels "01TG7V0g4F90jXv3a1yCgA",...: 278 103 621 364
- attr(*, ".internal.selfref")=<externalptr>
```

💡 c) Welche sind die Nummer 1 Alben der Jahre 2003, 2012 und 2020?

```
rs[rank_2003 == 1 | rank_2012 == 1 | rank_2020 == 1,
  .(clean_name, album, rank_2003, rank_2012, rank_2020)
]
```

	clean_name <fctr>	album <fctr>	rank_2003 <int>	rank_2012 <int>
1:	The Beatles	Sgt. Pepper's Lonely Hearts Club Band	1	1
2:	Marvin Gaye	What's Going On	6	6

	rank_2020 <int>
1:	24
2:	1

💡 d) Ist Ihre Lieblingsband in der Liste?

Angenommen, meine Lieblingsband sei *Faith No More*.

```
rs[clean_name == "Faith No More"]
```

Empty data.table (0 rows and 21 cols): sort_name, clean_name, album, rank_2003, rank_2012, rank_2020

Die ist leider nicht enthalten. Versuchen wir es mit *Eminem*.

```
rs[clean_name == "Eminem", .(clean_name, album)]
```

	clean_name <fctr>	album <fctr>
1:	Eminem	The Slim Shady LP
2:	Eminem	The Marshall Mathers LP
3:	Eminem	The Eminem Show

💡 e) Welche weiblichen Bands haben mehr als 3 Mitglieder?

```
rs[artist_gender == "Female" & artist_member_count > 3, .(sort_name, clean_name)]
```

	sort_name <fctr>	clean_name <fctr>
1:	Destiny's Child	Destiny's Child
2:	Raincoats	The Raincoats

3:	Go Gos	The Go-Go's
4:	Ross, Diana & the Supremes	The Supremes

💡 f) Welche Band hat die meisten Alben im Datensatz?

```
# zähle die Bands und sortiere absteigend
# und zeige nur die ersten 5 Reihen an
rs[, .N, by = clean_name] |>
  _[order(-N)][1:5]
```

	clean_name	N
	<fctr>	<int>
1:	Bob Dylan	11
2:	The Beatles	11
3:	Rolling Stones	10
4:	Bruce Springsteen	9
5:	The Who	7

Bob Dylan und The Beatles haben jeweils 11 Alben in den Listen

💡 g) Prüfen Sie per Korrelationsverfahren, ob die Beliebtheit bei Spotify (spotify_popularity) mit der Liste von 2020 übereinstimmt.

```
# Achtung, es sind NAs enthalten
rs[, cor(spotify_popularity, rank_2020, use="complete.obs")]
```

```
[1] -0.2215204
```

Es gibt einen geringen negativen Zusammenhang. Da beim Ranking ein *geringer* Wert für ein *gutes* Ranking steht, ist es auch nicht verwunderlich, dass der Zusammenhang negativ ist. Bei Spotify bedeutet ein *hoher* Wert ein gutes Ranking. Dennoch ist der Zusammenhang eher schwach.

💡 h) Welchen durchschnittlichen Rang erzielen Alben des Genres "Electronic" in den Jahren 2003, 2012 und 2020?

```
# Achtung, es sind NAs enthalten
rs[genre == "Electronic", .(d2003 = mean(rank_2003, na.rm=TRUE),
                                     d2012 = mean(rank_2012, na.rm=TRUE),
                                     d2020 = mean(rank_2020, na.rm=TRUE)
                               )]
```

	d2003	d2012	d2020
	<num>	<num>	<num>
1:	376.4286	362.1667	298.3636

- 💡 i) Berechnen Sie das arithmetische Mittel und den Median des Albenrankings für jedes Genre im Jahr 2020. Wieviele Alben sind pro Genre enthalten?

```
# Achtung, es sind NAs enthalten
rs[, .(mean = mean(rank_2020, na.rm=TRUE),
  median = as.numeric(median(rank_2020, na.rm=TRUE)),
  N = sum(!is.na(rank_2020))), by = genre]
```

	genre	mean	median	N
	<fctr>	<num>	<num>	<int>
1:	Big Band/Jazz	251.2000	268.0	10
2:	Rock n' Roll/Rhythm & Blues	243.2222	263.0	9
3:		254.6273	265.0	110
4:	Soul/Gospel/R&B	264.7846	275.0	65
5:	Hip-Hop/Rap	204.7544	192.0	57
6:	Blues/Blues Rock	267.6786	313.0	28
7:	Country/Folk/Country Rock/Folk Rock	223.3750	211.0	40
8:	Indie/Alternative Rock	259.8085	276.0	47
9:	Punk/Post-Punk/New Wave/Power Pop	253.8246	260.0	57
10:	Electronic	298.3636	241.0	11
11:	Funk/Disco	280.3158	360.0	19
12:	Latin	446.5000	471.0	6
13:	Hard Rock/Metal	215.6111	227.5	18
14:	Singer-Songwriter/Heartland Rock	231.8667	251.0	15
15:	Blues/Blues R0ck	259.0000	259.0	1
16:	Reggae	155.4000	140.0	5
17:	Afrobeat	433.5000	433.5	2

Haben Sie den 15. Eintrag Blues/Blues R0ck bemerkt? Es ist ein Tippfehler im Datensatz enthalten.

```
# korrigiere den Tippfehler
rs[genre == "Blues/Blues R0ck", genre := "Blues/Blues Rock"]
```

Jetzt sortieren wir wie gewünscht einmal absteigend nach dem Median, und einmal aufsteigend nach genre.

```
# sortiert nach Median
rs[, .(mean = mean(rank_2020, na.rm=TRUE),
  median = as.numeric(median(rank_2020, na.rm=TRUE)),
  N = sum(!is.na(rank_2020))), by = genre] |>
  _[order(median, decreasing = TRUE)]
```

	genre	mean	median	N
	<fctr>	<num>	<num>	<int>
1:	Latin	446.5000	471.0	6
2:	Afrobeat	433.5000	433.5	2
3:	Funk/Disco	280.3158	360.0	19
4:	Blues/Blues Rock	267.3793	299.0	29
5:	Indie/Alternative Rock	259.8085	276.0	47
6:	Soul/Gospel/R&B	264.7846	275.0	65
7:	Big Band/Jazz	251.2000	268.0	10
8:		254.6273	265.0	110
9:	Rock n' Roll/Rhythm & Blues	243.2222	263.0	9

```

10: Punk/Post-Punk/New Wave/Power Pop 253.8246 260.0 57
11: Singer-Songwriter/Heartland Rock 231.8667 251.0 15
12: Electronic 298.3636 241.0 11
13: Hard Rock/Metal 215.6111 227.5 18
14: Country/Folk/Country Rock/Folk Rock 223.3750 211.0 40
15: Hip-Hop/Rap 204.7544 192.0 57
16: Reggae 155.4000 140.0 5

```

```

# sortiert nach Genre
rs[, .(mean = mean(rank_2020, na.rm=TRUE),
  median = as.numeric(median(rank_2020, na.rm=TRUE)),
  N = sum(!is.na(rank_2020))), by = genre] |>
  _[order(genre)]

```

	genre	mean	median	N
	<fctr>	<num>	<num>	<int>
1:		254.6273	265.0	110
2:	Afrobeat	433.5000	433.5	2
3:	Big Band/Jazz	251.2000	268.0	10
4:	Blues/Blues Rock	267.3793	299.0	29
5:	Country/Folk/Country Rock/Folk Rock	223.3750	211.0	40
6:	Electronic	298.3636	241.0	11
7:	Funk/Disco	280.3158	360.0	19
8:	Hard Rock/Metal	215.6111	227.5	18
9:	Hip-Hop/Rap	204.7544	192.0	57
10:	Indie/Alternative Rock	259.8085	276.0	47
11:	Latin	446.5000	471.0	6
12:	Punk/Post-Punk/New Wave/Power Pop	253.8246	260.0	57
13:	Reggae	155.4000	140.0	5
14:	Rock n' Roll/Rhythm & Blues	243.2222	263.0	9
15:	Singer-Songwriter/Heartland Rock	231.8667	251.0	15
16:	Soul/Gospel/R&B	264.7846	275.0	65

💡 j) Manche Künstler haben es in jede der 3 Listen geschafft. Wie groß ist die Anzahl an Bands, die in jeder der 3 Listen vertreten sind, wieviele Alben haben in jeder der 3 Listen eine Platzierung, welche Alben sind in jeder der 3 Listen auf dem selben Platz, welche haben sich kontinuierlich verbessert, welche kontinuierlich verschlechtert?

```

# Anzahl der Bands, die in jeder der 3 Listen sind
rs[!is.na(rank_2003) & !is.na(rank_2012) & !is.na(rank_2020),
  uniqueN(clean_name)]

```

```
[1] 205
```

```

# Anzahl der Alben, die in jeder der 3 Listen sind
rs[!is.na(rank_2003) & !is.na(rank_2012) & !is.na(rank_2020),
  uniqueN(album)]

```

```
[1] 317
```

```
# Alben mit dem selben Ranking
rs[rank_2003 == rank_2012 & rank_2012 == rank_2020,
  .(clean_name, album, rank_2003, rank_2012, rank_2020)]
```

	clean_name	album	rank_2003	rank_2012	rank_2020
	<fctr>	<fctr>	<int>	<int>	<int>
1:	The Beach Boys	Pet Sounds	2	2	2

```
# Alben, die sich kontinuierlich verbessert haben
# (verbessert heisst, dass das Ranking kleiner wird)
rs[rank_2003 > rank_2012 & rank_2012 > rank_2020,
  .(clean_name, album, rank_2003, rank_2012, rank_2020)] |>
  _[order(clean_name)] |>
# zeige nur die ersten 10, um hier Platz zu sparen
head(10)
```

	clean_name	album	rank_2003
	<fctr>	<fctr>	<int>
1:	A Tribe Called Quest	The Low End Theory	154
2:	Big Star	Third/Sister Lovers	456
3:	Brian Eno	Here Come the Warm Jets	436
4:	Brian Eno	Another Green World	433
5:	Bruce Springsteen	Darkness on the Edge of Town	151
6:	Coldplay	A Rush of Blood to the Head	473
7:	Creedence Clearwater Revival	Willy and the Poor Boys	392
8:	Cyndi Lauper	She's So Unusual	494
9:	D'Angelo	Voodoo	488
10:	DEVO Q: Are We Not Men? A: We Are Devo!		447

	rank_2012	rank_2020
	<int>	<int>
1:	153	43
2:	449	285
3:	432	308
4:	429	338
5:	150	91
6:	466	324
7:	309	193
8:	487	184
9:	481	28
10:	442	252

```
# Alben, die sich kontinuierlich verschlechtert haben
# (verschlechtert heisst, dass das Ranking größer wird)
rs[rank_2003 < rank_2012 & rank_2012 < rank_2020,
  .(clean_name, album, rank_2003, rank_2012, rank_2020)] |>
  _[order(clean_name)] |>
# zeige nur die ersten 10, um hier Platz zu sparen
head(10)
```

clean_name	album
<fctr>	<fctr>

1:	AC/DC	Back in Black
2:	Al Green	I'm Still in Love With You
3:	Al Green	Call Me
4:	Beck	Odelay
5:	Bee Gees	Saturday Night Fever: The Original Movie Sound Track
6:	Billy Joel	The Stranger
7:	Black Sabbath	Paranoid
8:	Black Sabbath	Black Sabbath
9:	Bo Diddley	Bo Diddley / Go Bo Diddley
10:	Bob Dylan	John Wesley Harding

	rank_2003	rank_2012	rank_2020
	<int>	<int>	<int>
1:	73	77	84
2:	285	286	306
3:	289	290	427
4:	305	306	424
5:	131	132	163
6:	67	70	169
7:	130	131	139
8:	241	243	355
9:	214	216	455
10:	301	303	337

💡 k) Erzeugen Sie eine neue Variable `soloband`, in welcher festgehalten wird, ob es sich um eine(n) Solokünstler(in) handelt (`solo`), oder um eine Band (`band`). Sind Solokünstlerinnen besser platziert als Solokünstler? Sind Bands besser platziert als Solokünstler(innen)?

```
# Solo oder Band?
rs[, soloband := fifelse(artist_member_count > 1, "band", "solo")]

# Künstlerinnen besser als Künstler?
rs[soloband=="solo", .(mean03 = mean(rank_2003, na.rm=TRUE),
                                mean12 = mean(rank_2012, na.rm=TRUE),
                                mean20 = mean(rank_2020, na.rm=TRUE)
                                ), by = artist_gender]
```

	artist_gender	mean03	mean12	mean20
	<fctr>	<num>	<num>	<num>
1:	Male	234.3276	234.3314	245.3488
2:	Female	257.2424	262.2121	260.1176

Männliche Solokünstler sind in allen 3 Listen durchschnittlich besser gerankt als weibliche.

```
# Solo besser als Band?
rs[, .(mean03 = mean(rank_2003, na.rm=TRUE),
        mean12 = mean(rank_2012, na.rm=TRUE),
        mean20 = mean(rank_2020, na.rm=TRUE)
        ), by = soloband]
```

soloband	mean03	mean12	mean20
----------	--------	--------	--------

	<char>	<num>	<num>	<num>
1:	solo	237.9807	238.7548	250.2335
2:	band	258.7820	257.8223	249.1203
3:	<NA>	300.5000	318.8000	451.0000

In den Jahren 2003 und 2012 waren Solokünstler besser gerankt als Bands, in 2020 waren Bands leicht besser.

💡 1) Der Datensatz liegt als `wide.table` vor, da die Rankings für 2003, 2012 und 2020 als Variablen nebeneinander stehen. Wandeln Sie den Datensatz in eine `long.table` (Tidy Data) um, so dass die Rankingangaben in den Variablen `Rang` und `Rangjahr` angegeben sind.

```
# long table mit melt()
long_rs <- melt(rs,
  measure.vars = c("rank_2003", "rank_2012", "rank_2020"),
  variable.name = "Rangjahr",
  value.name = "Rang")

# anzeigen
head(long_rs)
```

	sort_name	clean_name	album	differential
	<fctr>	<fctr>	<fctr>	<int>
1:	Sinatra, Frank	Frank Sinatra	In the Wee Small Hours	-182
2:	Diddley, Bo	Bo Diddley Bo Diddley / Go Bo Diddley		-241
3:	Presley, Elvis	Elvis Presley	Elvis Presley	-277
4:	Sinatra, Frank	Frank Sinatra Songs for Swingin' Lovers!		-195
5:	Little Richard	Little Richard	Here's Little Richard	-177
6:	Beyonce	Beyonce	Lemonade	469
	release_year	genre	type	weeks_on_billboard
	<int>	<fctr>	<fctr>	<int>
1:	1955	Big Band/Jazz	Studio	14
2:	1955	Rock n' Roll/Rhythm & Blues	Studio	NA
3:	1956	Rock n' Roll/Rhythm & Blues	Studio	100
4:	1956	Big Band/Jazz	Studio	NA
5:	1957		Studio	5
6:	2016		Studio	87
	peak_billboard_position	spotify_popularity		
	<int>	<int>		
1:	2	48		
2:	201	50		
3:	1	58		
4:	2	62		
5:	13	64		
6:	1	73		
	spotify_url	artist_member_count	artist_gender	
	<fctr>	<int>	<fctr>	
1:	spotify:album:3GmwKB1tgPZgXeRJJZSm9WX	1	Male	
2:	spotify:album:1cbtDEwxCjMhg1b490gNBR	1	Male	
3:	spotify:album:7GXP50hYyPVLmcVf09Iqin	1	Male	
4:	spotify:album:4kca7vXd1Wo5GE2DMafvMc	1	Male	

```

5: spotify:album:18tV6PLXYvVjsd0VkoS7M8          1          Male
6: spotify:album:7dK54iZu0xXFarGhXwEXfF          1          Female
  artist_birth_year_sum debut_album_release_year ave_age_at_top_500
      <int>                <int>                <num>
1:          1915                1946                40
2:          1928                1955                27
3:          1935                1956                21
4:          1915                1946                41
5:          1932                1957                25
6:          1981                2003                35
  years_between      album_id soloband  Rangjahr  Rang
      <int>          <fctr>   <char>   <fctr> <int>
1:          9 3GmwKB1tgPZgXeRJJZSm9WX      solo rank_2003    100
2:          0 1cbtDEwxCjMhg1b490gNBR      solo rank_2003    214
3:          0 7GXP50hYyPVLmcVf09Iqin      solo rank_2003     55
4:         10 4kca7vXd1Wo5GE2DMafvMc      solo rank_2003    306
5:          0 18tV6PLXYvVjsd0VkoS7M8      solo rank_2003     50
6:         13 7dK54iZu0xXFarGhXwEXfF      solo rank_2003     NA

```

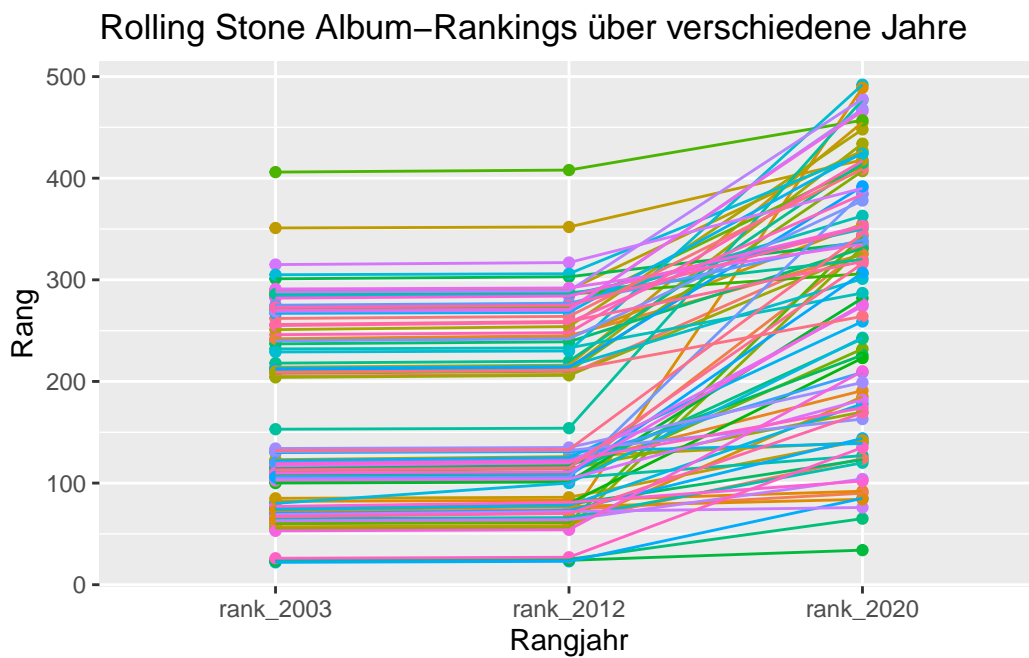
🔗 m) Plotten Sie mittels `ggplot()` die Rangveränderungen von 2003 bis 2020 für solche Alben, die sich kontinuierlich verschlechtert haben. Was fällt Ihnen auf?

```

# ggplot() aktivieren
library(ggplot2)

# Daten vorsortieren
rs[rank_2003 < rank_2012 & rank_2012 < rank_2020,
  .(album, rank_2003, rank_2012, rank_2020)] |>
# in long-table überführen
melt( , measure.vars = c("rank_2003", "rank_2012", "rank_2020"),
      variable.name = "Rangjahr",
      value.name = "Rang") |>
# ggplot
ggplot(aes(x=Rangjahr, y=Rang, color=album, group=album)) +
  geom_point() +
  geom_line() +
  ggtitle("Rolling Stone Album-Rankings über verschiedene Jahre") +
  theme(legend.position = "none")

```



Es fällt auf, dass sich die Rankings vor allem von 2012 nach 2020 deutlich verschlechtert haben.

4.5. Lösung zur Aufgabe 3.5 Taylor Swift

💡 a) Laden Sie den Datensatz `taylor_swift_spotify2024.csv` als `data.table` in Ihre R-Session. Nennen Sie Ihr Objekt dabei `ts` und verschaffen Sie sich mittels `str()` einen Überblick über die enthaltenen Daten.

```
# falls schon gedownloadet
ts <- fread("data/taylor_swift_spotify2024.csv")

# per URL einlesen
ts <- fread("https://www.produnis.de/tabletrainer/data/taylor_swift_spotify2024.csv")

# anschauen
str(ts)
```

```
Classes 'data.table' and 'data.frame': 582 obs. of 18 variables:
 $ V1      : int  0 1 2 3 4 5 6 7 8 9 ...
 $ name    : chr  "Fortnight (feat. Post Malone)" "The Tortured Poets Department" "My E
 $ album   : chr  "THE TORTURED POETS DEPARTMENT: THE ANTHOLOGY" "THE TORTURED POETS DE
 $ release_date : IDate, format: "2024-04-19" "2024-04-19" ...
 $ track_number : int  1 2 3 4 5 6 7 8 9 10 ...
 $ id      : chr  "6dODwocEuGzHAavXqTbwHv" "4PdLaGZubp4lghChqp8erB" "7uGYWMwRy24dm7RUDD
 $ uri     : chr  "spotify:track:6dODwocEuGzHAavXqTbwHv" "spotify:track:4PdLaGZubp4lghC
 $ acousticness : num  0.502 0.0483 0.137 0.56 0.73 0.384 0.624 0.178 0.607 0.315 ...
 $ danceability : num  0.504 0.604 0.596 0.541 0.423 0.521 0.33 0.533 0.626 0.606 ...
 $ energy      : num  0.386 0.428 0.563 0.366 0.533 0.72 0.483 0.573 0.428 0.338 ...
 $ instrumentalness: num  1.53e-05 0.00 0.00 1.00e-06 2.64e-03 0.00 0.00 0.00 0.00 0.00 ...
```

```

$ liveness      : num  0.0961 0.126 0.302 0.0946 0.0816 0.135 0.111 0.309 0.0921 0.106 ...
$ loudness      : num -10.98 -8.44 -7.36 -10.41 -11.39 ...
$ speechiness   : num  0.0308 0.0255 0.0269 0.0748 0.322 0.104 0.0399 0.138 0.0261 0.048 ...
$ tempo         : num  192 110.3 97.1 159.7 160.2 ...
$ valence       : num  0.281 0.292 0.481 0.168 0.248 0.438 0.34 0.398 0.487 0.238 ...
$ popularity    : int   82 79 80 82 80 81 78 79 82 81 ...
$ duration_ms   : int  228965 293048 203801 261228 262974 340428 210789 215463 254365 334084
- attr(*, ".internal.selfref")=<externalptr>

```

💡 b) Wenn nötig, korrigieren Sie das Skalenniveau (nominal, ordinal, metrisch) der Variablen innerhalb des Datensatzes.

```

ts[, let(name = factor(name),
        album = factor(album),
        id    = factor(id),
        uri   = factor(uri)
      )]

```

```

# anzeigen
str(ts)

```

Classes 'data.table' and 'data.frame': 582 obs. of 18 variables:

```

$ V1          : int  0 1 2 3 4 5 6 7 8 9 ...
$ name        : Factor w/ 363 levels "...Ready For It?","",...: 108 319 209 80 256 44 109 101
$ album       : Factor w/ 29 levels "1989","1989 (Deluxe)",...: 29 29 29 29 29 29 29 29 29
$ release_date : IDate, format: "2024-04-19" "2024-04-19" ...
$ track_number : int   1 2 3 4 5 6 7 8 9 10 ...
$ id          : Factor w/ 582 levels "00vJzaoxM3Eja1doBUhX0P",...: 463 343 578 116 580 349
$ uri         : Factor w/ 582 levels "spotify:track:00vJzaoxM3Eja1doBUhX0P",...: 463 343 578
$ acousticness : num  0.502 0.0483 0.137 0.56 0.73 0.384 0.624 0.178 0.607 0.315 ...
$ danceability : num  0.504 0.604 0.596 0.541 0.423 0.521 0.33 0.533 0.626 0.606 ...
$ energy       : num  0.386 0.428 0.563 0.366 0.533 0.72 0.483 0.573 0.428 0.338 ...
$ instrumentalness: num  1.53e-05 0.00 0.00 1.00e-06 2.64e-03 0.00 0.00 0.00 0.00 0.00 ...
$ liveness     : num  0.0961 0.126 0.302 0.0946 0.0816 0.135 0.111 0.309 0.0921 0.106 ...
$ loudness     : num -10.98 -8.44 -7.36 -10.41 -11.39 ...
$ speechiness  : num  0.0308 0.0255 0.0269 0.0748 0.322 0.104 0.0399 0.138 0.0261 0.048 ...
$ tempo        : num  192 110.3 97.1 159.7 160.2 ...
$ valence      : num  0.281 0.292 0.481 0.168 0.248 0.438 0.34 0.398 0.487 0.238 ...
$ popularity   : int   82 79 80 82 80 81 78 79 82 81 ...
$ duration_ms  : int  228965 293048 203801 261228 262974 340428 210789 215463 254365 334084
- attr(*, ".internal.selfref")=<externalptr>

```

💡 c) Erstellen Sie eine neue Variable sekunden, welche die Songlängen in Sekunden enthält

```

ts[, sekunden := duration_ms/1000]

```


💡 d) Wie lang dauern die Songs im Durchschnitt? Bei welcher Songlänge liegt der Median?

```
ts[, .(Durchschnitt = mean(sekunden),
      Median = median(sekunden)
    )]
```

```
Durchschnitt  Median
      <num>    <num>
1:      240.0112 235.433
```

💡 e) Welcher Song ist laut Datensatz der populärste, welcher der längste, und welcher der langsamste? Stellen Sie anschließend die Werte pro Album dar.

```
# populärster Song insgesamt
ts[popularity == max(popularity), name]
```

```
[1] Cruel Summer
363 Levels: ...Ready For It? ... You're On Your Own, Kid
```

```
# populärster Song pro Album
ts[, .SD[which.max(popularity)], by=album, .SDcols=c("name", "popularity")] |>
  _[order(popularity, decreasing=TRUE)]
```

```

                                                    album
                                                    <fctr>
1:                                                    Lover
2:                                     THE TORTURED POETS DEPARTMENT
3:                                                    Midnights
4:                                                    reputation
5:                                                    folklore
6:                                     THE TORTURED POETS DEPARTMENT: THE ANTHOLOGY
7:                                                    Fearless (Taylor's Version)
8:                                                    1989 (Deluxe)
9:                                                    1989 (Taylor's Version)
10:                                                    Speak Now (Taylor's Version)
11:                                                    Red (Taylor's Version)
12:                                     reputation Stadium Tour Surprise Song Playlist
13:                                                    evermore
14:                                                    1989
15:                                     1989 (Taylor's Version) [Deluxe]
16:                                                    evermore (deluxe version)
17:                                     Midnights (The Til Dawn Edition)
18:                                                    Speak Now
19:                                     Speak Now (Deluxe Package)
20:                                                    Midnights (3am Edition)
21:                                                    folklore (deluxe version)
22:                                     Taylor Swift (Deluxe Edition)
23:                                                    Red
24:                                                    Red (Deluxe Edition)
```

```

25:                                     Fearless (Platinum Edition)
26: folklore: the long pond studio sessions (from the Disney+ special) [deluxe edition]
27:                                     Speak Now World Tour Live
28:                                     Fearless (International Version)
29:                                     Live From Clear Channel Stripped 2008
                                     album

```

```

                                     name
                                     <fctr>
1:                                     Cruel Summer
2:                                     Fortnight (feat. Post Malone)
3:                                     Anti-Hero
4:                                     Don't Blame Me
5:                                     cardigan
6:                                     Fortnight (feat. Post Malone)
7:                                     You Belong With Me (Taylor's Version)
8:                                     Blank Space
9:                                     Blank Space (Taylor's Version)
10:                                    Enchanted (Taylor's Version)
11: All Too Well (10 Minute Version) (Taylor's Version) (From The Vault)
12:                                    I Don't Wanna Live Forever (Fifty Shades Darker)
13:                                    willow
14:                                    Wildest Dreams
15:                                    Bad Blood (feat. Kendrick Lamar) (Taylor's Version)
16:                                    right where you left me - bonus track
17:                                    Snow On The Beach (feat. More Lana Del Rey)
18:                                    Enchanted
19:                                    Enchanted
20:                                    The Great War
21:                                    the 1
22:                                    Our Song
23:                                    I Knew You Were Trouble.
24:                                    I Knew You Were Trouble.
25:                                    You Belong With Me
26:                                    my tears ricochet - the long pond studio sessions
27: Back To December/Apoloigize/You're Not Sorry - Live/2011/Medley
28:                                    Love Story
29: Beautiful Eyes - Live From Clear Channel Stripped 2008

```

```

                                     name
popularity
<int>
1:          93
2:          91
3:          85
4:          85
5:          84
6:          82
7:          81
8:          81
9:          80
10:         80
11:         80

```

```

12:      79
13:      77
14:      75
15:      74
16:      74
17:      73
18:      71
19:      71
20:      69
21:      68
22:      64
23:      61
24:      59
25:      57
26:      55
27:      47
28:      46
29:      38

```

popularity

```

# längster Song insgesamt
ts[sekunden == max(sekunden), name]

```

```

[1] All Too Well (10 Minute Version) (Taylor's Version) (From The Vault)
363 Levels: ...Ready For It? ... You're On Your Own, Kid

```

```

# längster Song pro Album
ts[, .SD[which.max(sekunden)], by=album, .SDcols=c("name", "sekunden")] |>
  _[order(sekunden, decreasing=TRUE)]

```

```

album
<fctr>
1:      Red (Taylor's Version)
2:      Speak Now (Taylor's Version)
3:      Speak Now World Tour Live
4:      Speak Now
5:      Speak Now (Deluxe Package)
6:      reputation Stadium Tour Surprise Song Playlist
7:      THE TORTURED POETS DEPARTMENT: THE ANTHOLOGY
8:      THE TORTURED POETS DEPARTMENT
9:      Red (Deluxe Edition)
10:      Red
11:      evermore (deluxe version)
12:      evermore
13:      Fearless (Taylor's Version)
14:      Fearless (Platinum Edition)
15: folklore: the long pond studio sessions (from the Disney+ special) [deluxe edition]
16:      folklore
17:      folklore (deluxe version)
18:      Fearless (International Version)
19:      Lover

```

```

20:                                     1989 (Taylor's Version) [Deluxe]
21:                                     1989 (Taylor's Version)
22:                                     1989 (Deluxe)
23:                                     1989
24:                                     Midnights (The Til Dawn Edition)
25:                                     Midnights (3am Edition)
26:                                     Live From Clear Channel Stripped 2008
27:                                     Midnights
28:                                     Taylor Swift (Deluxe Edition)
29:                                     reputation
                                     album

```

```

                                     name
                                     <fctr>
1: All Too Well (10 Minute Version) (Taylor's Version) (From The Vault)
2:                                     Dear John (Taylor's Version)
3:                                     Dear John - Live/2011
4:                                     Dear John
5:                                     Dear John
6:                                     Enchanted
7:                                     But Daddy I Love Him
8:                                     But Daddy I Love Him
9:                                     All Too Well
10:                                    All Too Well
11:                                    happiness
12:                                    happiness
13:                                    Untouchable (Taylor's Version)
14:                                    Untouchable
15: my tears ricochet - the long pond studio sessions
16:                                    betty
17:                                    betty
18:                                    Fifteen
19:                                    Daylight
20: Say Don't Go (Taylor's Version) (From The Vault)
21: Say Don't Go (Taylor's Version) (From The Vault)
22:                                    Clean
23:                                    Clean
24:                                    Would've, Could've, Should've
25:                                    Would've, Could've, Should've
26: Change - Live From Clear Channel Stripped 2008
27: Snow On The Beach (feat. Lana Del Rey)
28: Tied Together with a Smile
29: End Game
                                     name

```

sekunden

<num>

```

1: 613.026
2: 405.906
3: 404.680
4: 403.933
5: 403.887
6: 353.253

```

```

7: 340.428
8: 340.428
9: 329.160
10: 327.893
11: 315.146
12: 315.146
13: 312.107
14: 311.040
15: 295.173
16: 294.521
17: 294.520
18: 294.306
19: 293.453
20: 279.833
21: 279.833
22: 271.000
23: 271.000
24: 260.361
25: 260.361
26: 258.487
27: 256.124
28: 248.106
29: 244.826
    sekunden

```

```

# langsamster Song insgesamt
ts[tempo == min(tempo), name]

```

```

[1] this is me trying - the long pond studio sessions
363 Levels: ...Ready For It? ... You're On Your Own, Kid

```

```

# langsamster Song pro Album
ts[, .SD[which.min(tempo)], by=album, .SDcols=c("name", "tempo")] |>
  _[order(tempo)]

```

```

album
<fctr>
1: folklore: the long pond studio sessions (from the Disney+ special) [deluxe edition]
2:                                                                 Lover
3:                               THE TORTURED POETS DEPARTMENT: THE ANTHOLOGY
4:                                                                 1989
5: reputation Stadium Tour Surprise Song Playlist
6:                               Red (Taylor's Version)
7:                               Taylor Swift (Deluxe Edition)
8:                               evermore (deluxe version)
9:                               evermore
10: reputation
11: folklore
12: folklore (deluxe version)
13:                               Red (Deluxe Edition)
14:                               Red

```

15: Speak Now World Tour Live
 16: Speak Now (Taylor's Version)
 17: 1989 (Taylor's Version) [Deluxe]
 18: 1989 (Taylor's Version)
 19: THE TORTURED POETS DEPARTMENT
 20: 1989 (Deluxe)
 21: Midnights (The Til Dawn Edition)
 22: Midnights (3am Edition)
 23: Midnights
 24: Fearless (Platinum Edition)
 25: Fearless (Taylor's Version)
 26: Speak Now (Deluxe Package)
 27: Speak Now
 28: Fearless (International Version)
 29: Live From Clear Channel Stripped 2008
 album

	name	tempo
	<fctr>	<num>
1:	this is me trying - the long pond studio sessions	68.097
2:	Lover	68.534
3:	Chloe or Sam or Sophia or Marcus	70.266
4:	This Love	71.981
5:	Breathe	73.849
6:	Better Man (Taylor's Version) (From The Vault)	73.942
7:	Mary's Song (Oh My My My)	74.900
8:	tolerate it	74.952
9:	tolerate it	74.952
10:	So It Goes...	74.957
11:	exile (feat. Bon Iver)	75.602
12:	exile (feat. Bon Iver)	75.938
13:	I Knew You Were Trouble.	77.019
14:	I Knew You Were Trouble.	77.019
15:	Ours - Live/2011	77.769
16:	When Emma Falls in Love (Taylor's Version) (From The Vault)	77.879
17:	"Slut!" (Taylor's Version) (From The Vault)	77.978
18:	"Slut!" (Taylor's Version) (From The Vault)	77.978
19:	loml	78.539
20:	I Know Places - Voice Memo	78.828
21:	Vigilante Shit	79.964
22:	Vigilante Shit	79.964
23:	Vigilante Shit	79.964
24:	Jump Then Fall	79.991
25:	Bye Bye Baby (Taylor's Version) (From The Vault)	80.132
26:	Haunted - Acoustic Version	80.858
27:	Enchanted	81.975
28:	White Horse	92.710
29:	Fearless - Live From Clear Channel Stripped 2008	96.601
	name	tempo

💡 f) Welches Album hat die meisten Songs, und welches hat die wenigsten Songs?

```
# die meisten Songs
ts[ , .N, by=album][order(-N)][1]
```

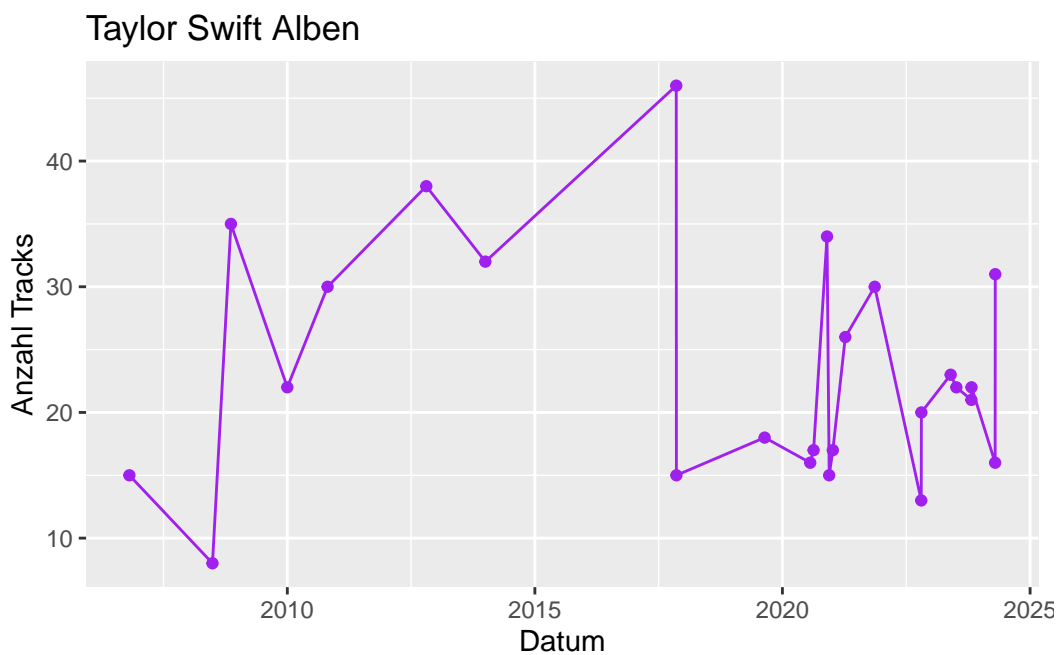
```
              album      N
              <fctr> <int>
1: reputation Stadium Tour Surprise Song Playlist    46
```

```
# die wenigsten Songs
ts[ , .N, by=album][order(N)][1]
```

```
              album      N
              <fctr> <int>
1: Live From Clear Channel Stripped 2008           8
```

💡 g) Plotten Sie die Anzahl der Tracks pro Album als Punkt-Liniendiagramm, wobei das Datum auf der X-Achse, und die Trackanzahl auf der Y-Achse dargestellt werden.

```
# Zähle Tracks pro Album
ts[ , .N, by=release_date] |>
  # ggplot
  ggplot(aes(x = release_date, y = N)) +
    geom_line(color = "purple") +
    geom_point(color = "purple") +
    labs(x = "Datum",
         y = "Anzahl Tracks",
         title = "Taylor Swift Alben")
```



4.6. Lösung zur Aufgabe 3.6 Anscombe-Quartett

- 💡 a) Laden Sie den Datensatz `anscombe` in Ihre R-Session und überführen Sie ihn in ein `data.table` Objekt mit dem Namen `ac`.

```
# aktiviere Datensatz
data("anscombe")

# überführe in data.table "ac"
ac <- as.data.table(anscombe)
```

- 💡 b) Die Daten liegen als *breite Tabelle* (wide tabel) vor. Überführen Sie sie ins *long table* (tidy) Format, so dass Ihre `data.table` aus den Spalten `x`, `y`, und `Gruppe` besteht.

Hierfür benutzen wir `patterns()`, um mittels *regular expression* alle Spalten auszuwählen, deren Namen mit `x` oder `y` anfangen.

```
# tidy data Format
lac <- melt(ac,
  measure.vars = patterns("^x", "^y"),
  value.name = c("x", "y"),
  variable.name = "Gruppe"
)
```

- 💡 c) Berechnen Sie für jede Gruppe die Mittelwerte, Standardabweichungen, Korrelations- und Regressionskoeffizienten von `x` und `y`, wobei Sie Ihre Ergebnisse auf 2 Stellen runden sollen.

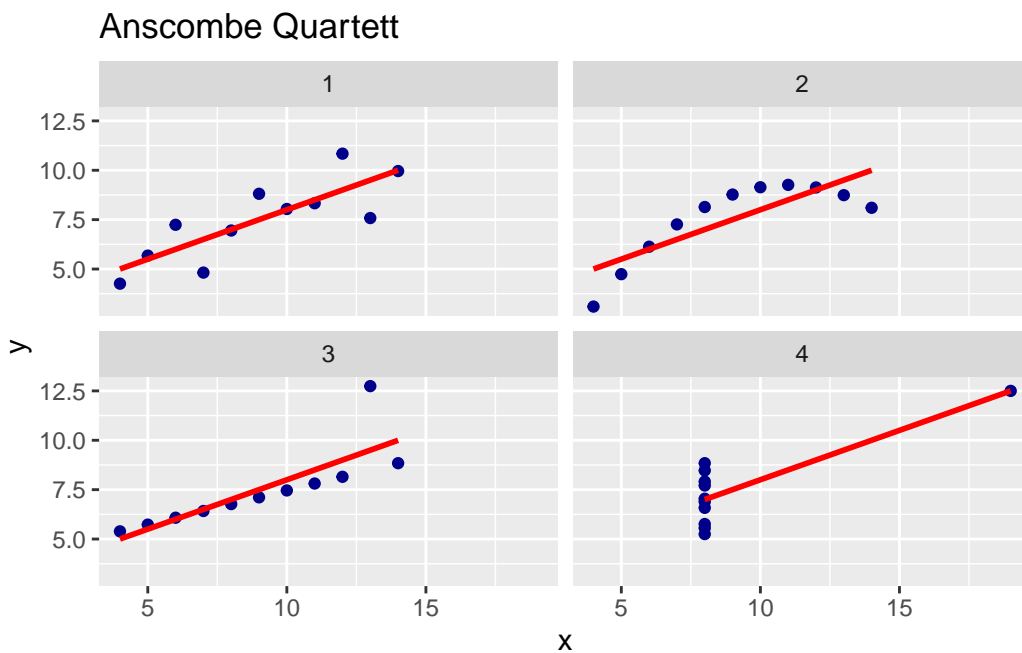
```
# berechne Anscombe Werte
lac[, .(MittelX = round(mean(x), 2),
  StdabwX = round(sd(x), 2),
  MittelY = round(mean(y), 2),
  StdabwY = round(sd(y), 2),
  Korrela = round(cor(x, y), 2),
  Regress = round(lm(y~x)$coefficients[2], 2)
), by = Gruppe]
```

	Gruppe	MittelX	StdabwX	MittelY	StdabwY	Korrela	Regress
	<fctr>	<num>	<num>	<num>	<num>	<num>	<num>
1:	1	9	3.32	7.5	2.03	0.82	0.5
2:	2	9	3.32	7.5	2.03	0.82	0.5
3:	3	9	3.32	7.5	2.03	0.82	0.5
4:	4	9	3.32	7.5	2.03	0.82	0.5

🔥 Jede Gruppe liefert die selben Werte!

💡 d) Erzeugen Sie mittels `ggplot()` eine Punktwolke mit Regressionsgeraden für jede Gruppe, wobei alle 4 Diagramme mit einem Plotaufwurf erzeugt werden sollen.

```
# plotten
ggplot(lac, aes(x=x, y=y)) +
  geom_point(color="darkblue")+
  geom_smooth(method="lm", color="red", se=FALSE) +
  facet_wrap(~Gruppe) +
  ggtitle("Anscombe Quartett")
```



🔥 Jede Gruppe liefert zwar die selben Kennwert, aber die Diagramme sehen nicht nur vollständig unterschiedlich aus, sie legen auch nahe, dass Ausreißer oder Messfehler die Ergebnisse verzerrt haben.

4.7. Lösung zur Aufgabe 3.7 Neugeborene

💡 a) Überführen Sie die Daten in ein `data.table`-Objekt mit dem Namen `ng`.

```
# Lade Daten
load("https://www.produnis.de/tabletrainer/data/neonates.RData")

# überführe in data.table
ng <- as.data.table(neonates)

# anschauen
summary(ng)
```

weight gender age smoke cigarettes

```

Min.    :2.021   male   :157   greater than 20:218   No :220   Min.    : 0.000
1st Qu.:2.794   female:163   less than 20    :102   Yes:100   1st Qu.: 0.000
Median :3.030                                     Median : 0.000
Mean    :3.026                                     Mean    : 3.891
3rd Qu.:3.267                                     3rd Qu.: 8.250
Max.    :4.182                                     Max.    :22.000

smoke.before      apgar1      apgar5
No :185           Min.    :2.000   Min.    : 2.000
Yes:135           1st Qu.:5.000   1st Qu.: 5.000
                Median :6.000   Median : 6.000
                Mean    :5.628   Mean    : 6.213
                3rd Qu.:6.000   3rd Qu.: 7.000
                Max.    :9.000   Max.    :10.000

```

💡 b) Die Variabel `apgar1` enthält die APGAR-Scores nach 1 Minute. Wenn ein Score von 3 oder weniger anzeigt, dass das Neugeborene in einem kritischen Zustand ist, wie viel Prozent der Neugeborenen in der Stichprobe sind dann in einem kritischen Zustand?

```
ng[, .(Prozent = 100 * mean(apgar1 <= 3))]
```

Prozent

<num>

1: 7.8125

7,8125% der Neugeborenen sind in einem kritischen Zustand.

💡 c) Erstellen Sie die Häufigkeitstabelle des Geburtsgewichts der Neugeborenen, indem Sie die Daten in Klassen mit einer Breite von 0,5 kg von 2 bis 4,5 kg einteilen. Welches Intervall enthält die meisten Neugeborenen?

```

ng[, gewichtK := cut(weight,
                     breaks=seq(2, 4.5, by=0.5),
                     right=FALSE)] |>
  _[, jgsbook::freqTable(gewichtK)]

```

	Wert	Haeufig	Hkum	Relativ	Rkum
1	[2,2.5)	22	22	6.88	6.88
2	[2.5,3)	127	149	39.69	46.57
3	[3,3.5)	146	295	45.62	92.19
4	[3.5,4)	24	319	7.50	99.69
5	[4,4.5)	1	320	0.31	100.00

💡 d) Vergleichen Sie die Häufigkeitsverteilung des APGAR-Scores nach 1 Minute für Mütter unter 20 Jahren und für Mütter über 20 Jahren. Welche Gruppe hat mehr Neugeborene in kritischem Zustand?

```
# Jünger als 20
ng[age=="less than 20", jgsbook::freqTable(apgar1)]
```

	Wert	Haeufig	Hkum	Relativ	Rkum
1	2	2	2	1.96	1.96
2	3	11	13	10.78	12.74
3	4	16	29	15.69	28.43
4	5	28	57	27.45	55.88
5	6	28	85	27.45	83.33
6	7	12	97	11.76	95.09
7	8	4	101	3.92	99.01
8	9	1	102	0.98	99.99

```
# Älter als 20
ng[age=="greater than 20", jgsbook::freqTable(apgar1)]
```

	Wert	Haeufig	Hkum	Relativ	Rkum
1	2	2	2	0.92	0.92
2	3	10	12	4.59	5.51
3	4	22	34	10.09	15.60
4	5	53	87	24.31	39.91
5	6	69	156	31.65	71.56
6	7	34	190	15.60	87.16
7	8	24	214	11.01	98.17
8	9	4	218	1.83	100.00

In der Gruppe der unter-20-jährigen liegt der Prozentsatz an Neugeborenen mit APGAR-Werten kleiner-gleich 3 bei 12,74%. In der Gruppe der über-20-jährigen liegt der Prozentwert bei 5,51%. Es tritt also in der Gruppe der jüngeren Mütter häufiger auf.

💡 e) Vergleichen Sie die relative Häufigkeitsverteilung des Geburtsgewichts der Neugeborenen, je nachdem, ob die Mutter während der Schwangerschaft geraucht hat (smoke) oder nicht. Wenn ein Gewicht unter 2,5 kg als niedriges Gewicht gilt, welche Gruppe hat einen höheren Prozentsatz an Neugeborenen mit niedrigem Gewicht?

```
# Prozentsatz Geburtsgewicht kleiner 2,5kg
ng[, .(Prozent = 100 * mean(weight<2.5)), by=smoke]
```

	smoke	Prozent
	<fctr>	<num>
1:	No	2.272727
2:	Yes	17.000000

In der Gruppe der Nichtraucherinnen trat ein Geburtsgewicht kleiner 2,5kg in 2,27% der Fälle auf. Bei den Raucherinnen waren es 17%.

💡 f) Berechnen Sie die Prävalenz von Neugeborenen mit niedrigem Gewicht für Mütter, die vor der Schwangerschaft geraucht haben (`smoke.before`), und den Nichtraucherinnen.

```
# Prozentsatz Geburtsgewicht kleiner 2,5kg
ng[, .(Prozent = 100 * mean(weight<2.5)), by=smoke.before]
```

	smoke.before	Prozent
	<fctr>	<num>
1:	No	1.081081
2:	Yes	14.814815

Die Prävalenz beträgt unter den Nichtraucherinnen 1,08% und unter den Raucherinnen 14,81%.

💡 g) Berechnen Sie die Odds Ratio eines niedrigen Geburtsgewichts des Neugeborenen, wenn die Mutter während der Schwangerschaft raucht, im Vergleich dazu, wenn die Mutter nicht raucht.

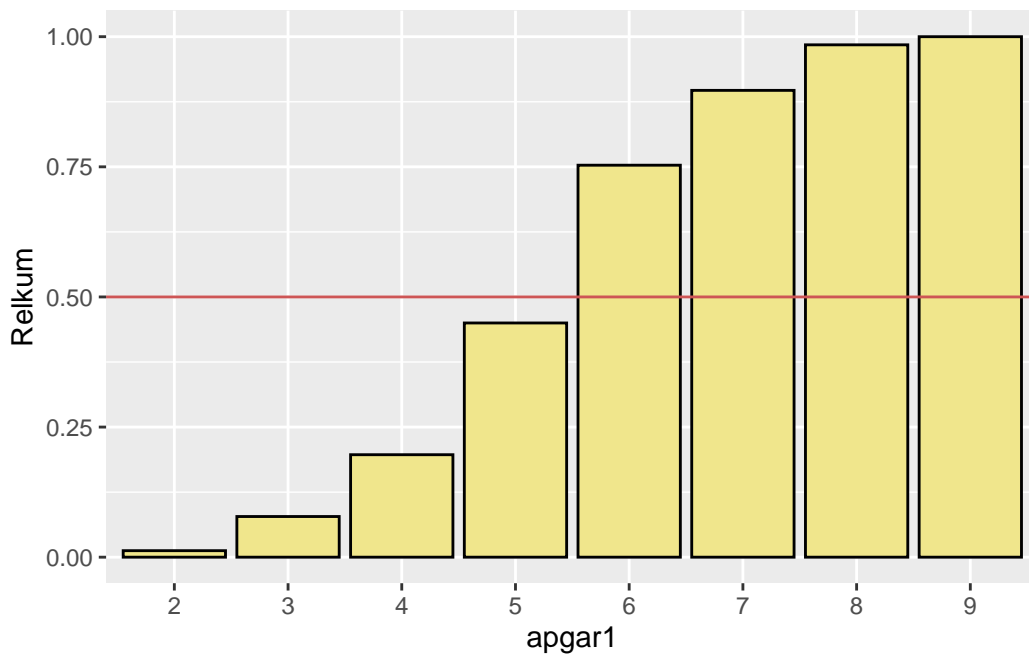
```
# Kreuztabelle erzeugen
t <- dcast(ng, smoke ~ (weight < 2.5), fun=length, value.var="weight")
# in Matrix überführen
t <- as.matrix(t, rownames=TRUE)
# Odds Ratio
epitools::oddsratio(t)$measure
```

		NA		
odds ratio with 95% C.I.	estimate	lower	upper	
	No	1.000000	NA	NA
	Yes	8.558258	3.236616	27.28525

Raucherinnen haben ein 8-fach höheres Risiko ein Kind mit niedrigem Gewicht zugebären als Nichtraucherinnen.

💡 h) Erstellen Sie das Balkendiagramm der kumulierten relativen Häufigkeit des APGAR-Scores nach 1 Minute. Unter welchem Wert liegen die Hälfte der Neugeborenen?

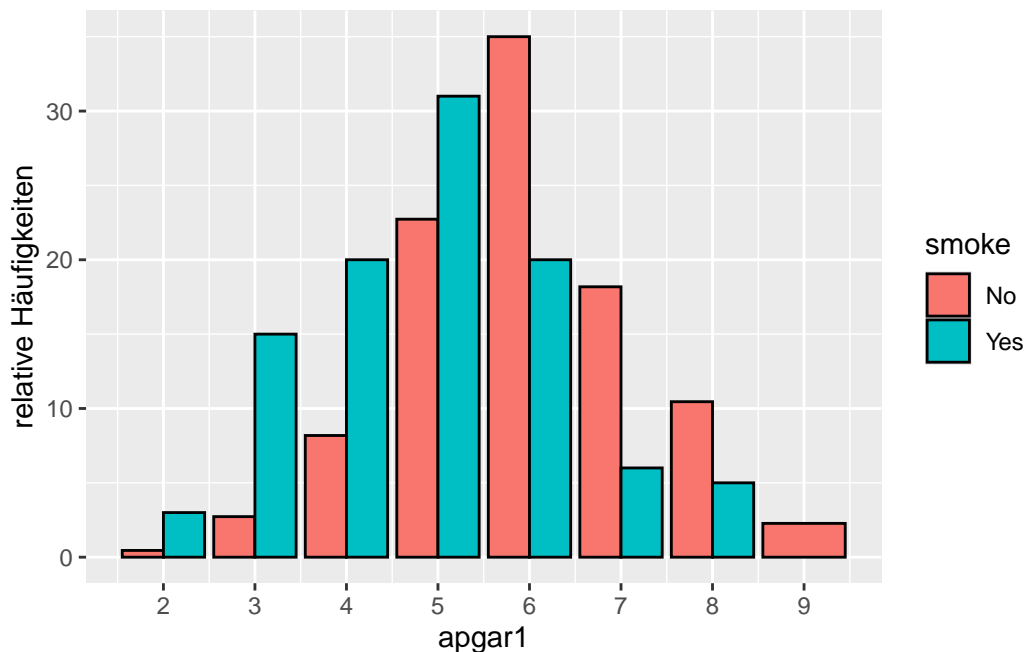
```
# relative Häufigkeitstabelle
ng[, .(prop.table(table(apgar1))) |>
# kumulieren
_[, Relkum := cumsum(N)][] |>
# an ggplot() senden
ggplot(aes(x=apgar1, y=Relkum)) +
  geom_bar(stat="identity", col="black", fill="khaki") +
  geom_hline(yintercept = 0.5, col="indianred3")
```



Die Hälfte der Werte liegen unter 6.

💡 i) Vergleichen Sie die Balkendiagramme der relativen Häufigkeitsverteilungen des APGAR-Scores nach 1 Minute, je nachdem, ob die Mutter während der Schwangerschaft geraucht hat oder nicht. Welche Schlussfolgerungen können gezogen werden?

```
# relative Häufigkeitstabelle
ng[, apgar1, by=smoke] |>
  # an ggplot() senden
  ggplot(aes(x=apgar1, y=after_stat(prop)*100, fill=smoke)) +
    geom_bar(col="black", position = "dodge") +
    scale_x_continuous(breaks = c(1:9)) +
    ylab("relative Häufigkeiten")
```



Die Kinder der Raucherinnen haben geringere APGAR-Werte. Kein Kind von Raucherinnen erreicht Wert 9.

💡 j) Berechnen Sie Median, Durchschnitt und Standardabweichung für die APGAR-Scores nach 1 und nach 5 Minuten jeweils für die Kinder von Müttern, die vor der Schwangerschaft geraucht haben, und den Nichtraucherinnen. Geben Sie auch die Anzahl an Fällen (N) an. Bewerten Sie die Ergebnisse.

```
# wähle benötigte Variablen aus
ng[, .(smoke.before, apgar1, apgar5)] |>
  # überführe in long table (tidy data)
  melt(id.vars= "smoke.before",
        measure.vars = c("apgar1", "apgar5"),
        variable.name = "Test",
        value.name = "APGAR_Score")|>
  # berechne die Kennwerte pro Gruppe
  _, .(Median = median(APGAR_Score),
        Mittel = mean(APGAR_Score),
        StdAbw = sd(APGAR_Score),
        N = .N
        ), by = c("Test", "smoke.before")]
```

Test	smoke.before	Median	Mittel	StdAbw	N
<fctr>	<fctr>	<num>	<num>	<num>	<int>
1: apgar1	No	6	6.054054	1.241185	185
2: apgar1	Yes	5	5.044444	1.449996	135
3: apgar5	No	7	6.616216	1.284843	185
4: apgar5	Yes	6	5.659259	1.546122	135

Kinder von Frauen, die vor der Schwangerschaft geraucht haben, haben zu beiden Messzeitpunkten niedrigere APGAR-Scores und eine höhere Streuung der Werte.

💡 k) Ist der Unterschied der APGAR-Scores aus Aufgabe j) signifikant?

Testen wir zunächst auf Normalverteilung

```
# Teste apgar1 auf Normalverteilung
ng[, .(p = shapiro.test(apgar1)$p.value), by="smoke.before"]
```

	smoke.before	p
	<fctr>	<num>
1:	No	8.733447e-07
2:	Yes	5.514593e-05

```
# Teste apgar5 auf Normalverteilung
ng[, .(p = shapiro.test(apgar5)$p.value), by="smoke.before"]
```

	smoke.before	p
	<fctr>	<num>
1:	No	2.449301e-07
2:	Yes	3.197790e-04

Alle Tests sind signifikant, d.h. es liegt **keine** Normalverteilung vor. Wir dürfen also nicht den t-Test verwenden, sondern müssen den Mann-Whitney-U-Test anwenden.

```
# Signifikanztest für APGAR1
ng[, wilcox.test(apgar1 ~ smoke.before)$p.value]
```

```
[1] 3.02217e-10
```

```
# Signifikanztest für APGAR5
ng[, wilcox.test(apgar5 ~ smoke.before)$p.value]
```

```
[1] 1.759924e-08
```

Beide Ergebnisse sind signifikant. Das bedeutet, dass der Unterschied in den APGAR-Scores zwischen Nicht-raucherinnen und Müttern, die vor der Schwangerschaft geraucht haben, signifikant ist.

4.8. Lösung zur Aufgabe 3.8 Verteidigung gegen die dunklen Künste

💡 a) Laden Sie die Textdateien als `data.table` in Ihre R-Session, und führe Sie diese zu einem einzelnen `data.table`-Objekt mit dem Namen `hp` zusammen.

```
# Lade Daten
lupin <- fread("https://www.produnis.de/tabletrainer/data/VgddK_Lupin.txt")
moody <- fread("https://www.produnis.de/tabletrainer/data/VgddK_Moody.txt")
umbri <- fread("https://www.produnis.de/tabletrainer/data/VgddK_Umbridge.txt")
```

```
# führe zu einem Objekt "hp" zusammen
hp <- lupin[mood, on=.(Schüler=Schüler)][umbri, on=.(Schüler=Schüler)]

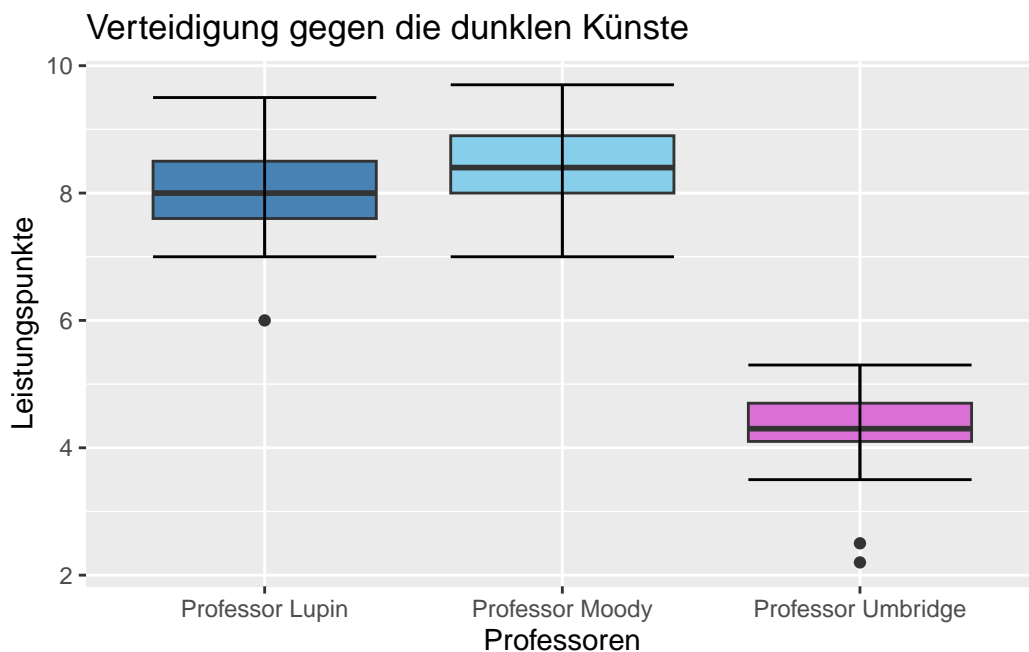
# anschauen
str(hp)
```

Classes 'data.table' and 'data.frame': 25 obs. of 4 variables:

```
$ Schüler      : chr  "Harry Potter" "Hermine Granger" "Ron Weasley" "Neville Longbottom"
$ Professor Lupin : num  8.5 9.5 7.5 6 8 8.2 7.8 7 8.5 7.7 ...
$ Professor Moody : num  9 9.7 8 7 8.5 8.8 8.2 7.5 8.9 8.2 ...
$ Professor Umbridge: num  2.5 5 2.2 3.5 4.8 4.7 4 3.8 4.6 4.2 ...
- attr(*, ".internal.selfref")=<externalptr>
```

💡 b) Erstellen Sie mit `ggplot()` ein Diagramm, welches die Leistungspunkte als Boxplots für jeden Professor darstellt. Hierfür bietet es sich an, die Daten ins long table Format zu überführen.

```
# Daten ins long-table-tidy-Format bringen
melt(hp,
      id.vars = "Schüler",
      variable.name = "Professoren",
      value.name = "Leistungspunkte") |>
# mit ggplot() plotten
ggplot(aes(x=Professoren, y=Leistungspunkte)) +
  geom_boxplot(fill=c("steelblue", "skyblue", "orchid")) +
  stat_boxplot(geom="errorbar") +
  ggtitle("Verteidigung gegen die dunklen Künste")
```



4.9. Lösung zur Aufgabe 3.9 Hogwarts Hauspunkte

- 💡 a) Laden Sie den Datensatz in Ihre R-Session, und überführen Sie ihn in eine `data.table` mit dem Namen `pp`.

```
# Lade Daten
load("https://www.produnis.de/tabletrainer/data/PotterHauspunkte.RData")

# überführe in data.table
pp <- as.data.table(PotterPunkte)

# anschauen
str(pp)
```

```
Classes 'data.table' and 'data.frame': 3273 obs. of 4 variables:
 $ Haus : Factor w/ 4 levels "Gryffindor","Hufflepuff",...: 1 3 1 3 3 1 2 4 3 3 ...
 $ Jahr1: num 52 48 49 50 72 42 5 25 30 49 ...
 $ Jahr3: num 58 70 59 62 74 86 38 54 58 70 ...
 $ Jahr5: num 73 86 75 72 83 77 45 81 77 91 ...
- attr(*, ".internal.selfref")=<externalptr>
```

- 💡 b) Berechnen Sie Median, Mittelwert und Standardabweichung für die Hauspunkte insgesamt, und jeweils für jedes Haus und Jahr gesondert.

```
# Werte insgesamt
pp[, .(Median = median(c(Jahr1, Jahr3, Jahr5)),
      Mittel = mean(c(Jahr1, Jahr3, Jahr5)),
      Stdabw = sd(c(Jahr1, Jahr3, Jahr5))
    )]
```

```
      Median      Mittel      Stdabw
      <num>      <num>      <num>
1:      57 57.56788 20.75463
```

```
# Werte für Haus und Schuljahr
melt(pp,
      id.vars = "Haus",
      measure.vars = c("Jahr1", "Jahr3", "Jahr5"),
      variable.name = "Schuljahr",
      value.name = "Hauspunkte") |>
_[, .(Median = median(Hauspunkte),
      Mittel = mean(Hauspunkte),
      Stdabw = sd(Hauspunkte)), by = c("Haus", "Schuljahr")]
```

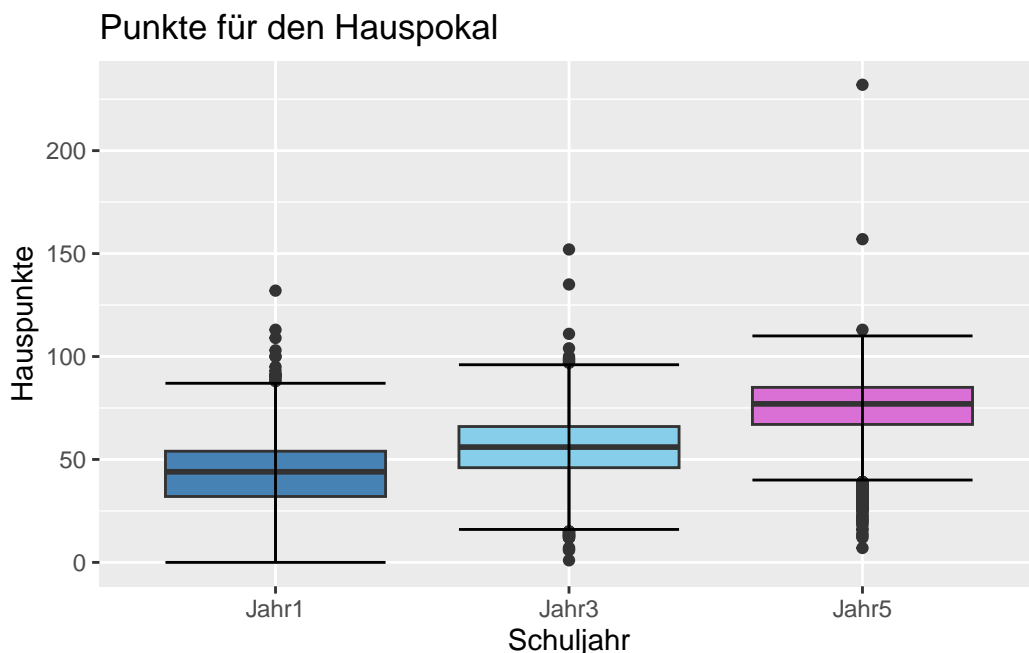
```
      Haus Schuljahr Median      Mittel      Stdabw
      <fctr>      <fctr> <num>      <num>      <num>
1: Gryffindor      Jahr1    44 44.12576 11.609269
2: Ravenclaw      Jahr1    51 50.41860 13.008144
```

3: Hufflepuff	Jahr1	33	35.60562	21.915475
4: Slytherin	Jahr1	41	40.22494	16.580231
5: Gryffindor	Jahr3	57	57.84615	12.826595
6: Ravenclaw	Jahr3	65	65.20808	13.089933
7: Hufflepuff	Jahr3	45	45.17582	14.781732
8: Slytherin	Jahr3	55	54.80562	13.750892
9: Gryffindor	Jahr5	79	79.37729	11.757972
10: Ravenclaw	Jahr5	85	85.23501	8.721373
11: Hufflepuff	Jahr5	56	55.47619	14.694455
12: Slytherin	Jahr5	78	77.38386	8.102576

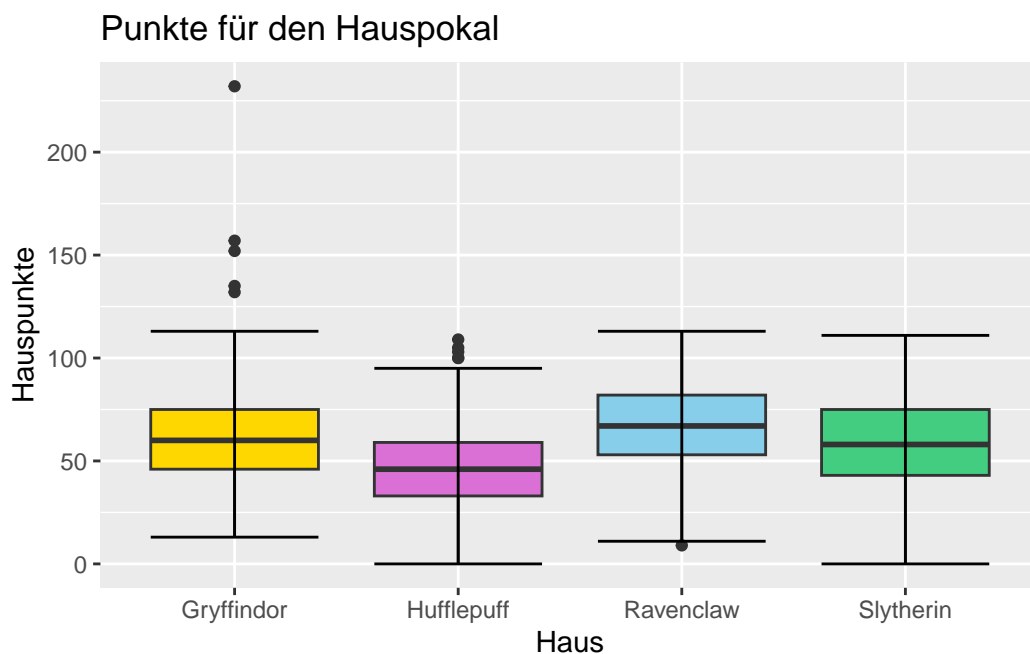
🔗 c) Plotten Sie die Punkte als Boxplots in Abhängigkeit zum Schuljahr, und dann in Abhängigkeit zum Haus. Verknüpfen Sie abschließend diese Bedingungen mittels `facet.wrap()`.

```
# Daten ins long-table-tidy-Format bringen
long_pp <- melt(pp,
  id.vars = "Haus",
  measure.vars = c("Jahr1", "Jahr3", "Jahr5"),
  variable.name = "Schuljahr",
  value.name = "Hauspunkte")

# plotte Punkte pro Schuljahr
ggplot(long_pp, aes(x=Schuljahr, y=Hauspunkte)) +
  geom_boxplot(fill=c("steelblue", "skyblue", "orchid")) +
  stat_boxplot(geom="errorbar") +
  ggtitle("Punkte für den Hauspokal")
```

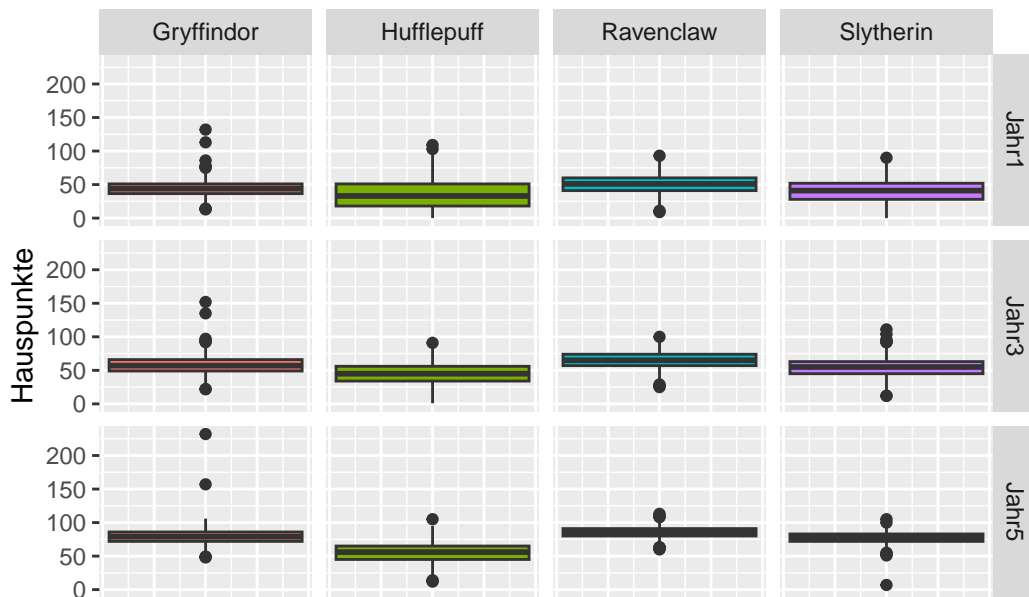


```
# plotte Punkte pro Haus
ggplot(long_pp, aes(x=Haus, y=Hauspunkte)) +
  geom_boxplot(fill=c("gold", "orchid", "skyblue", "seagreen3")) +
  stat_boxplot(geom="errorbar") +
  ggtitle("Punkte für den Hauspokal")
```



```
# Kombination von beiden Bedingungen
ggplot(long_pp, aes(y=Hauspunkte, fill=Haus)) +
  geom_boxplot() +
  theme(axis.text.x=element_blank(),
        axis.ticks.x=element_blank(),
        legend.position = "none") +
  facet_grid(Schuljahr~Haus) +
  ggtitle("Punkte für den Hauspokal")
```

Punkte für den Hauspokal



4.10. Lösung zur Aufgabe 3.10 Lungenkapazität

💡 a) Laden Sie den Datensatz lungcap als `data.table` mit dem Namen `lc` in Ihre R-Session

```
# Lade Daten
library(GLMsData)
data("lungcap")

# überführe in data.table
lc <- as.data.table(lungcap)

# anschauen
str(lc)
```

```
Classes 'data.table' and 'data.frame': 654 obs. of 5 variables:
 $ Age : int 3 4 4 4 4 4 4 5 5 5 ...
 $ FEV : num 1.072 0.839 1.102 1.389 1.577 ...
 $ Ht : num 46 48 48 48 49 49 50 46.5 49 49 ...
 $ Gender: Factor w/ 2 levels "F","M": 1 1 1 1 1 1 1 1 1 1 ...
 $ Smoke : int 0 0 0 0 0 0 0 0 0 0 ...
 - attr(*, ".internal.selfref")=<externalptr>
```

💡 b) Erzeugen Sie eine neue Variable Körpergröße, welche die Körpergröße in Zentimetern enthält (1 Zoll = 2,54cm)

```
lc[, Körpergröße := Ht*2.54 ]
```

💡 c) Ändern Sie die Kodierung der Variable Smoke, so dass statt 0 “nein”, und statt 1 “ja” enthalten ist. Passen Sie dabei auch das Skalenniveau an.

```
# ändere Kodierung von Smoke
lc[, Smoke := ifelse(Smoke == 1, "ja", "nein")]

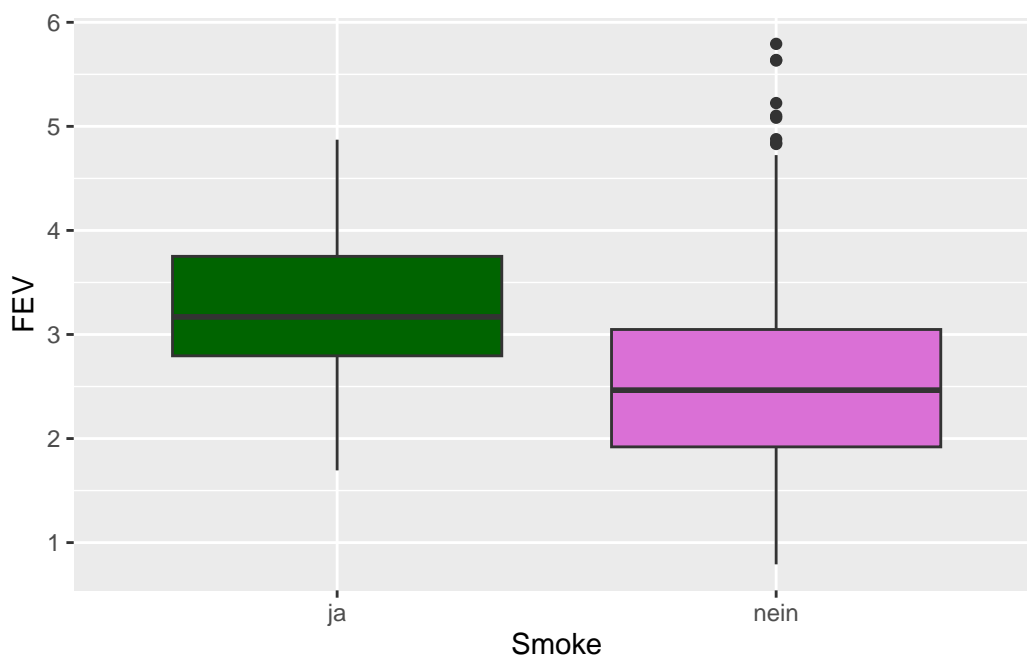
# passe Skalenniveau an
lc[, Smoke := factor(Smoke)]
```

Das geht auch in einer Zeile:

```
# ändere Kodierung und setze Factor in einem Rutsch
lc[, Smoke := factor(ifelse(Smoke == 1, "ja", "nein"))]
```

💡 d) Plotten Sie nebeneinander die Boxplots der Lungenkapazität nichtrauchenden und rauchenden Kindern. Legt das Diagramm einen Zusammenhang nahe?

```
ggplot(lc, aes(x=Smoke, y=FEV)) +
  geom_boxplot(fill=c("darkgreen", "orchid"))
```



Es scheint, als ob rauchende Kinder eine größere Lungenkapazität hätten.

💡 e) Führen Sie einen Signifikanztest durch, um zu überprüfen, ob sich die Lungenkapazitäten in Abhängigkeit zu Smoke unterscheiden.

Zunächst prüfen wir, ob die Daten in FEV normalverteilt sind.

```
lc[, .(p = shapiro.test(FEV)$p.value),
      by = Smoke]
```

	Smoke	p
	<fctr>	<num>
1:	nein	6.711569e-11
2:	ja	1.427488e-01

Der Test ist signifikant, d.h. FEV ist nicht normalverteilt. Wir müssen daher den Mann-Whitney-U-Test verwenden.

```
lc[, wilcox.test(FEV ~ Smoke, alternative = "greater")$p.value]
```

```
[1] 2.035427e-11
```

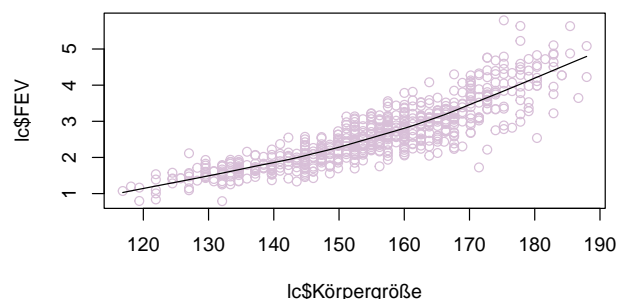
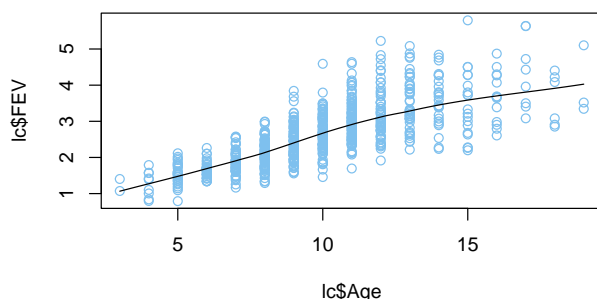
Der Test ist signifikant. Die Raucher haben eine größere Lungenkapazität als Nichtraucher.

⚠ Das sollte Sie erstmal verwundern!

Raucher haben die *besseren* Lungen?

💡 f) Erzeugen Sie eine Punktwolke des Lungenvolumens und des Alters, sowie des Lungenvolumens und der Körpergröße. Legen die Diagramme einen Zusammenhang nahe?

```
scatter.smooth(lc$Age, lc$FEV, col="skyblue2")
scatter.smooth(lc$Körpergröße, lc$FEV, col="thistle")
```



Es scheint einen linearen Zusammenhang zwischen dem Alter und der Lungenkapazität sowie zwischen der Körpergröße und der Lungenkapazität zu geben.

g) Welches Regressionsmodell ist am besten geeignet, um FEV erklärt durch Alter zu bestimmen, und welches ist am besten geeignet, um FEV erklärt durch Körpergröße zu bestimmen?

```
# Modelle für FEV~Age
lc[, jgsbook::compare.lm(FEV, Age)]
```

	Modell	R.square
7	potenz	0.6308534
4	exponentiell	0.5957878
3	kubisch	0.5925193
6	sigmoidal	0.5902058
2	quadratisch	0.5840171
1	linear	0.5722302
5	logarithmisch	0.5701891

```
# Modelle für FEV~Age
lc[, jgsbook::compare.lm(FEV, Körpergröße)]
```

	Modell	R.square
4	exponentiell	0.7956073
7	potenz	0.7944652
6	sigmoidal	0.7879391
3	kubisch	0.7741673
2	quadratisch	0.7740993
1	linear	0.7536584
5	logarithmisch	0.7370097

Für FEV erklärt durch Alter ist ein Potenzmodell am besten geeignet. Für FEV erklärt durch Körpergröße ist es ein exponentielles Modell. Dabei ist R^2 mit 0,79 größer als beim Potenzmodell des Alters (0,63).

Die Lungenkapazität wird am besten durch die Körpergröße erklärt.

h) Berechnen Sie das Modell, welches FEV am besten erklärt.

```
# bestimme exponentielles Modell
summary(lc[, lm(log(FEV) ~ Körpergröße)])
```

Call:

```
lm(formula = log(FEV) ~ Körpergröße)
```

Residuals:

	Min	1Q	Median	3Q	Max
	-0.70208	-0.08986	0.01190	0.09337	0.43174

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	-2.2713118	0.0635310	-35.75	<2e-16 ***
Körpergröße	0.0205193	0.0004073	50.38	<2e-16 ***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1508 on 652 degrees of freedom

Multiple R-squared: 0.7956, Adjusted R-squared: 0.7953

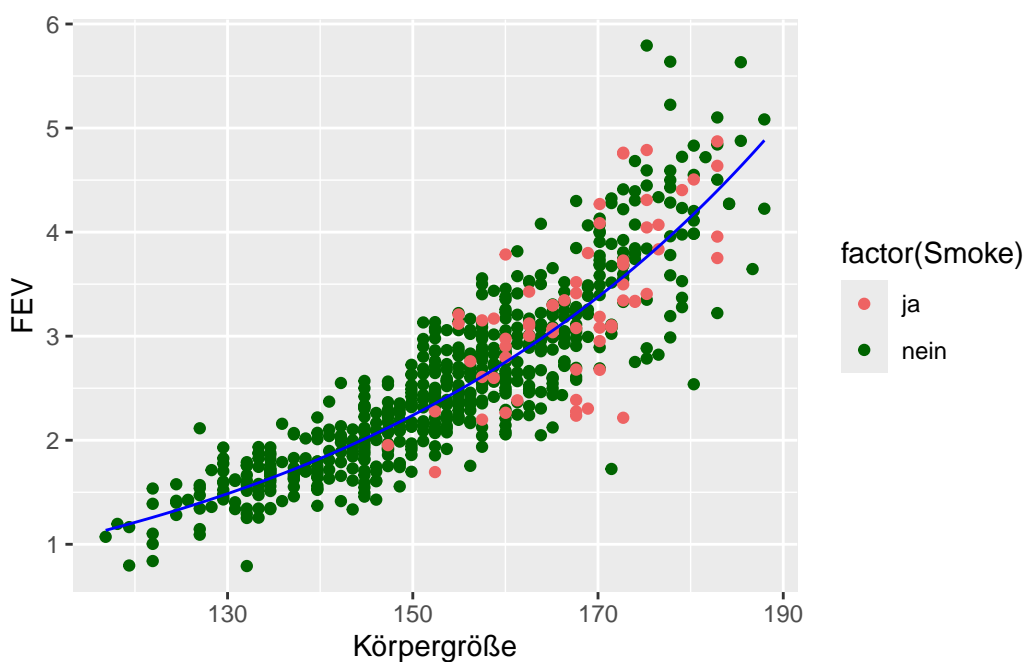
F-statistic: 2538 on 1 and 652 DF, p-value: < 2.2e-16

💡 i) Plotten Sie eine Punktwolke, mit FEV auf der Y-Achse, und dem besten Prädiktor auf der X-Achse. Färben Sie die Daten mittels der Variable Smoke. Fügen Sie anschließend Ihre Modelllinie dem Plot hinzu.

Der beste Prädiktor ist Körpergröße.

```
# speichere Modellvorhersagen in helper Objekt
helper <- lc[, jgsbook::compare.lm(FEV, Körpergröße, predict=TRUE)]

# plotte
ggplot(lc, aes(x=Körpergröße, y=FEV)) +
  geom_point(aes(color=factor(Smoke))) +
  scale_color_manual(values=c("indianred2", "darkgreen")) +
  geom_line(data=helper, aes(x=pred.x, y=expo), color="blue")
```



💡 j) Fügen Sie Smoke, Age und Gender als weitere Prädiktor dem Modell hinzu. Hat Rauchen einen Einfluss auf FEV?

```
# exponentielles Modell um "Smoke", "Age" und "Gender" erweitern
fit <- lc[, lm(log(FEV) ~ Körpergröße + Age + Gender + Smoke)]
summary(fit)
```


Call:

```
lm(formula = log(FEV) ~ Körpergröße + Age + Gender + Smoke)
```

Residuals:

Min	1Q	Median	3Q	Max
-0.63278	-0.08657	0.01146	0.09540	0.40701

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	-1.990066	0.081888	-24.302	< 2e-16 ***
Körpergröße	0.016849	0.000661	25.489	< 2e-16 ***
Age	0.023387	0.003348	6.984	7.1e-12 ***
GenderM	0.029319	0.011719	2.502	0.0126 *
Smokein	0.046067	0.020910	2.203	0.0279 *

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1455 on 649 degrees of freedom

Multiple R-squared: 0.8106, Adjusted R-squared: 0.8095

F-statistic: 694.6 on 4 and 649 DF, p-value: < 2.2e-16

Alle Prädiktoren sind signifikant. Der Beitrag von Smoke ist negativ. Dies spricht dafür, dass Rauchen die Lungenkapazität verschlechtert.

```
# Modelle vergleichen
fit0 <- lc[, lm(log(FEV) ~ Körpergröße)]
# R^2 vergleichen
summary(fit0)$r.squared - summary(fit)$r.squared
```

```
[1] -0.01503201
```

Durch Hinzunahme der Prädiktoren verbessert sich R^2 , aber nur minimal.

zusammengefasst

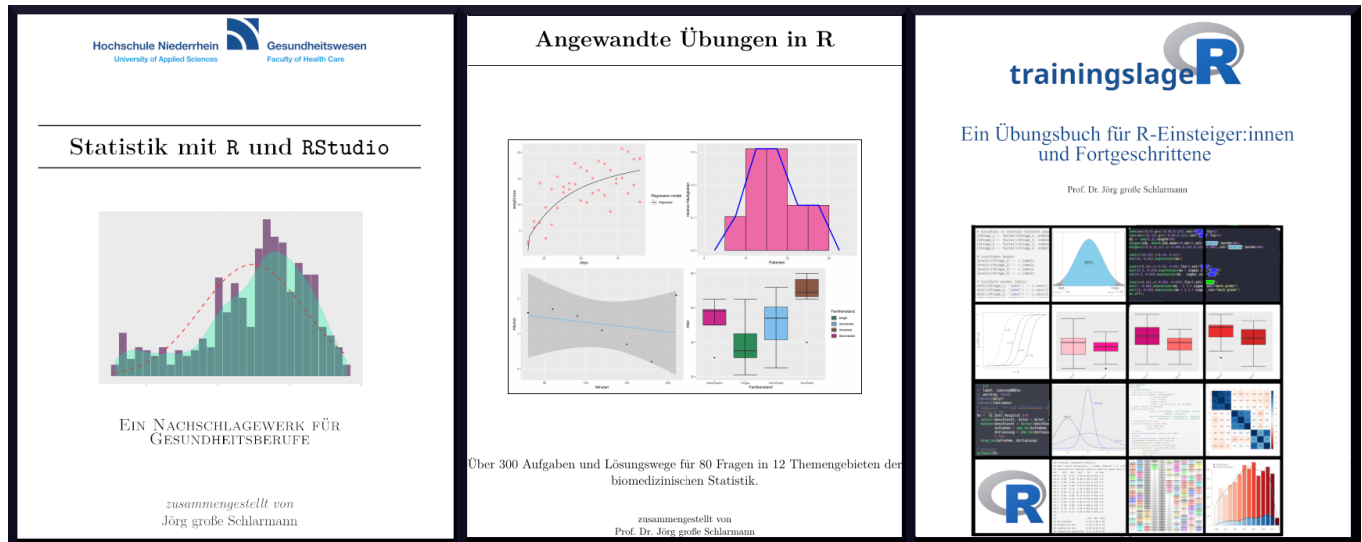
Es scheint nur auf den ersten Blick so, als hätten rauchende Kinder ein *besseres* Lungenvolumen als nicht-rauchende Kinder. Das liegt daran, dass

- das Lungenvolumen maßgeblich von der Körpergröße abhängt, und
- die jüngeren Kinder eher nicht rauchen, sondern die älteren. Diese haben dann aber auch einen größeren Körper, und somit auch ein größeres Lungenvolumen als die *kleinen* Nichtraucher.

Literaturverzeichnis

- Anscombe, F. J. (1973). Graphs in Statistical Analysis. *American Statistician*, 27(1), 17–21. <https://doi.org/10.1080/00031305.1973.10478966>
- Corrales, P., & Campitelli, E. (2024). *Introduction to data.table*. Workshop at useR!2024 Conference. <https://paocorales.github.io/intro-datatable/material.html>
- große Schlarmann, J. (2024a). *Angewandte Übungen in R*. Hochschule Niederrhein. 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>
- große Schlarmann, J. (2024c). *trainingslageR. Ein Übungsbuch für R-Einsteiger*innen und Fortgeschrittene*. Hochschule Niederrhein. <https://www.produnis.de/trainingslager>
- Isfort, M., Rottländer, R., Weidner, F., Gehlen, D., Hylla, J., & Tucman, D. (2018). *Pflege-Thermometer 2018 - Eine bundesweite Befragung von Führungskräften zur Situation der Pflege und Patientenversorgung in der stationären Langzeitpflege in Deutschland*. Deutsches Institut für angewandte Pflegeforschung e.V. (DIP).
- Kahn, M. (2005). An Exhilarant Problem for Teaching Statistics. *Journal of Statistics Education*, 13(2), 6. <https://doi.org/10.1080/10691898.2005.11910559>
- Mock, T. (2022). *Tidy Tuesday: A weekly data project aimed at the R ecosystem*. <https://github.com/rfordatascience/tyduesday>
- R Core Team. (2023). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing. <https://www.R-project.org/>
- Tager, I. B., Weiss, S. T., Muñoz, A., Rosner, B., & Speizer, F. E. (1983). Longitudinal study of the effects of maternal smoking on pulmonary function in children. *The New England Journal of Medicine*, 309(12), 699–703. <https://doi.org/10.1056/NEJM198309223091204>
- 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)

(a) große Schlarmann (2024c)



Prof. Dr. Jörg große Schlarmann, BScN, MScN, RN

Hochschule Niederrhein, Krefeld

joerg.grosseschlarmann@hs-niederrhein.de

<https://www.github.com/produnis/tabletrainer>