



Ein Übungsbuch für R-Einsteiger:innen und Fortgeschrittene

Prof. Dr. Jörg große Schlarmann

Version vom 01.07.2024

Lizenz



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

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

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

Inhaltsverzeichnis

Lizenz	i
Inhaltsverzeichnis	ii
Einleitung	1
I. Aufgaben	2
1. Aufgaben für Einsteiger:innen	3
1.1. Objekte in R	3
1.1.1. Aufgabe 1.1.1 Vektoren	3
1.1.2. Aufgabe 1.1.2 Zufallsvektoren	4
1.1.3. Aufgabe 1.1.3 Krankenhausaufenthalte	4
1.1.4. Aufgabe 1.1.4 Größe und Gewicht	5
1.1.5. Aufgabe 1.1.5 ordinale Faktoren	5
1.1.6. Aufgabe 1.1.6 Hogwarts-Kurse	6
1.1.7. Aufgabe 1.1.7 Datentabelle	7
1.1.8. Aufgabe 1.1.8 Zusatzpaket	8
1.1.9. Aufgabe 1.1.9 Daten laden	8
1.2. Deskriptive Statistik	9
1.2.1. Aufgabe 1.2.1 Serumcholesterin	9
1.2.2. Aufgabe 1.2.2 Gewichtsreduktion	10
1.2.3. Aufgabe 1.2.3 Anscombe-Quartett	11
1.2.4. Aufgabe 1.2.4 Kinder und Wohnräume	11
1.2.5. Aufgabe 1.2.5 Kinder und Geschwister	12
2. Aufgaben für geübte User:innen	13
2.1. Objekte in R	13
2.1.1. Aufgabe 2.1.1 Hogwarts-Kurse	13
2.1.2. Aufgabe 2.1.2 Aufnahme und Entlassung	14
2.2. Datensätze auswerten	15
2.2.1. Aufgabe 2.2.1 Aufnahme und Entlassung	15
3. Aufgaben für fortgeschrittene Anwender:innen	16
3.1. Objekte in R	16
3.1.1. Aufgabe 3.1.1 Hogwarts-Kurse	16
3.2. Datensätze auswerten	16
3.2.1. Aufgabe 3.2.1 Kurse	16

II. Lösungswege	17
4. Lösungswege zu den Aufgaben für Einsteiger:innen	18
4.1. Lösungen zu Objekten in R	18
4.1.1. Lösung zur Aufgabe 1.1.1 Vektoren	18
4.1.2. Lösung zur Aufgabe 1.1.2 Zufallsvektoren	19
4.1.3. Lösung zur Aufgabe 1.1.3 Krankenhausaufenthalte	20
4.1.4. Lösung zur Aufgabe 1.1.4 Größe und Gewicht	22
4.1.5. Lösung zur Aufgabe 1.1.5 ordinale Faktoren	24
4.1.6. Lösung zur Aufgabe 1.1.6 Hogwarts-Kurse	26
4.1.7. Lösung zur Aufgabe 1.1.7 Datentabelle	30
4.1.8. Lösung zur Aufgabe 1.1.8 Zusatzpaket	34
4.1.9. Lösung zur Aufgabe 1.1.9 Daten laden	36
4.2. Lösungen zur deskriptiven Statistik	40
4.2.1. Lösung zur Aufgabe 1.2.1 Serumcholesterin	40
4.2.2. Lösung zur Aufgabe 1.2.2 Gewichtsreduktion	45
4.2.3. Lösung zur Aufgabe 1.2.3 Anscombe-Quartett	53
4.2.4. Lösung zur Aufgabe 1.2.4 Kinder und Wohnräume	59
4.2.5. Lösung zur Aufgabe 1.2.5 Kinder und Geschwister	62
5. Lösungswege zu den Aufgaben für geübte User:innen	66
5.1. Lösungen zu Objekten in R	66
5.1.1. Lösung zur Aufgabe 2.1.1 Hogwarts-Kurse	66
5.1.2. Lösung zur Aufgabe 2.1.2 Aufnahme und Entlassung	69
5.2. Lösungen zu den Datensatzauswertungen	72
5.2.1. Lösung zur Aufgabe 2.2.1 Aufnahme und Entlassung	72
6. Lösungswege zu den Aufgaben für fortgeschrittene Anwender:innen	95
6.1. Lösungen zu Objekten in R	95
6.1.1. Lösung zur Aufgabe 3.1.1 Hogwarts-Kurse	95
6.2. Lösungen zu den Datensatzauswertungen	95
6.2.1. Lösung zur Aufgabe 3.2.1 Hogwarts-Kurse	95
Literaturverzeichnis	96
Credits	97

Einleitung

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

Willkommen im trainingslageR!

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

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

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

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

Teil I.

Aufgaben

1. Aufgaben für Einsteiger:innen

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


1.1. Objekte in R

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

1.1.1. Aufgabe 1.1.1 Vektoren

i

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

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

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



Lösung siehe Abschnitt 4.1.1

1.1.2. Aufgabe 1.1.2 Zufallsvektoren



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



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

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



Lösung siehe Abschnitt 4.1.2

1.1.3. Aufgabe 1.1.3 Krankenhausaufenthalte



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

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

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



Lösung siehe Abschnitt 4.1.3

1.1.4. Aufgabe 1.1.4 Größe und Gewicht


i Von 10 Personen wurden folgende Körpergrößen in Meter gemessen:

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

... sowie folgende Gewichte in Gramm:

78500	110100	97500	69200	82500
71500	81500	87200	75500	65500

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


 Lösung siehe Abschnitt 4.1.4

1.1.5. Aufgabe 1.1.5 ordinale Faktoren

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

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

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


 Lösung siehe Abschnitt 4.1.5

1.1.6. Aufgabe 1.1.6 Hogwarts-Kurse

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

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

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

 Lösung siehe Abschnitt 4.1.6

1.1.7. Aufgabe 1.1.7 Datentabelle

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

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


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

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

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


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

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

 Lösung siehe Abschnitt 4.1.7


1.1.8. Aufgabe 1.1.8 Zusatzpaket

- i** Das Zusatzpaket `jgsbook` enthält Funktionen und Datensätze aus dem freien Buch von große Schlar-mann (2024b).
- a) Installieren Sie das Zusatzpaket `jgsbook` mit allen Abhängigkeiten.
 - b) Welche Datensätze sind in dem Paket enthalten?
 - c) Speichern Sie den Datensatz `pf8` aus dem `jgsbook` in das Objekt `df`. Welche Variablen sind im Datensatz enthalten?
 - d) Rufen Sie Dokumentation für das `jgsbook`-Paket auf.
 - e) Wenden Sie die Funktion `freqTable()` aus dem Paket `jgsbook` auf die Variable `df$Kinder` an, **ohne** das Paket vorher per `library()` zu aktivieren.

 Lösung siehe Abschnitt 4.1.8

1.1.9. Aufgabe 1.1.9 Daten laden

- i** Laden Sie die folgenden Datensatz jeweils in ein R-Objekt und passen Sie die Datenklassen der Variablen entsprechend des Skalenniveaus an.
- a) <https://www.produnis.de/R/data/Datentabelle.txt>
 - b) <https://www.produnis.de/R/data/anwesenheitnoten.csv>
 - c) <https://www.produnis.de/R/data/alteDaten-kurz.sav>
 - d) <https://www.produnis.de/R/data/Testdatumdaten.xlsx>

 Lösung siehe Abschnitt 4.1.9

1.2. Deskriptive Statistik

In diesem Abschnitt üben Sie typische Funktionen und Arbeitsfolgen zur deskriptiven Auswertung der Daten.

1.2.1. Aufgabe 1.2.1 Serumcholesterin

i Ein Internist misst bei 20 seiner Patienten folgende Serumcholesterinspiegel in mmol/l

4,5 4,9 7,3 5,2 5,8 6,2 5,0 5,6 6,4 7,6
5,4 4,4 6,6 5,3 5,7 4,7 8,2 6,7 4,8 5,9

- Überführen Sie die Daten in ein Datenframe mit der Variable `chol`.
- Klassieren Sie die Serumcholesterinwerte nach folgendem Schema:
 - 4,0 bis 4,9;
 - 5,0 bis 5,9;
 -mmol/l
- Erstellen Sie eine ausreichend beschriftete Häufigkeitstabelle mit nicht kumulierten und kumulierten absoluten und relativen Häufigkeiten für die Häufigkeiten in den zuvor erstellten Serumcholesterinklassen.
- Bestimmen Sie bitte folgende Kenngrößen:
 - Median arithmetisches Mittel Spannweite
 - Varianz und Standardabweichung
 - Minimum 10. Perzentil 1. Quartil 3. Quartil 90. Perzentil Maximum
 - Interquartilsabstand
- Erstellen Sie einen Boxplot der Werte.
- Stellen Sie die in a) aufgelisteten absoluten nicht kumulierten Häufigkeiten als Histogramm dar.
- Welche Form hat die Verteilung?

💡 Lösung siehe Abschnitt 4.2.1

1.2.2. Aufgabe 1.2.2 Gewichtsreduktion

- i** Zu einer Gruppe von 20 Teilnehmern an einem Kurs zur Gewichtsreduktion liegen Ihnen die Angaben zu Alter [Jahren] und Geschlecht [1: männlich; 2: weiblich] vor.

Alter: 4 7 8 9 11 12 13 14 15 16 16 20 20 22 25 26 26 28 29 34
Geschlecht: 1 2 2 2 1 1 2 2 2 1 1 2 2 2 1 0 2 1 2 0


- a) Übertragen Sie die Daten in ein R-Datenframe.
b) Geben Sie der Variable “Geschlecht” die Werte

 'männlich' (statt 1)
 'weiblich' (statt 2)
 'divers' (statt 0)

- c) Klassieren Sie das Alter der Probanden nach folgendem Schema:

 0-5 6-10
 11-15 16-20
 21-25 26-30
 31-35

- d) Bestimmen Sie folgende Stichprobenkennzahlen für das Merkmal ‘Alter’:
- Minimum 5. Perzentil 1. Quartil Median Mittelwert
 - 3. Quartil 95. Perzentil Maximum Interquartilsabstand
- e) Zeichnen Sie ein Histogramm und ein Balkendiagramm für die nicht kumulierten absoluten Häufigkeiten zur Anzahl der Studienteilnehmer in den zuvor gebildeten Altersklassen.
f) Erstellen Sie eine Kontingenztafel zur gleichzeitigen Darstellung der beiden Merkmale Altersgruppe und Geschlecht.
g) Stellen Sie die Häufigkeitsverteilung der beiden Merkmale Altersgruppe und Geschlecht in einer geeigneten Graphik dar.


 Lösung siehe Abschnitt 4.2.2

1.2.3. Aufgabe 1.2.3 Anscombe-Quartett

i Das Anscombe-Quartett ist ein bekannter Datensatz in der Statistik. Lesen Sie sich zunächst den Wikipedia-Artikel durch, 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.
- Schreiben Sie die 4 Anscombe-Datensätze (`x1` bis `x4` und `y1` bis `y4`) in 4 neue Datenframes mit den Namen `Anscombe1` bis `Anscombe4`. Die enthaltenen Spalten sollten jeweils `x` und `y` heissen.
- Führen Sie für jedes Datenframe die Berechnungen von Anscombe durch (Mittelwert, Varianz, Korrelation und lineare Regression), wobei Sie Ihre Ergebnisse auf 2 Stellen runden sollen.
- Erzeugen Sie die 4 Anscombe-Diagramme (Punktwolke und Regressionsgerade) mit der `plot()`-Funktion, und hübschen Sie die Plots mit etwas Farbe auf.
- Erzeugen Sie die 4 Anscombe-Diagramme mittels `ggplot()`, wobei alle 4 Diagramme mit einem Plotaufruf erzeugt werden sollen. Dies geht am einfachsten, wenn der Datensatz im Tidy-Data-Format (long table) vorliegt.


 Lösung siehe Abschnitt 4.2.3

1.2.4. Aufgabe 1.2.4 Kinder und Wohnräume

i Man befragt 5 Ehepaare, bei denen beide Partner zwischen 20 und 40 Jahre alt sind, nach der Anzahl der im Haushalt lebenden Kinder (X) und nach der Anzahl der Wohnräume der Wohnung (Y). Die Antworten lauten:

Ehepaar	1	2	3	4	5
Anzahl Kinder im Haushalt (X)	0	2	3	0	1
Anzahl der Wohnräume (Y)	1	4	3	2	3

- Berechnen Sie den Korrelationskoeffizienten r
- Berechnen Sie die Regressionsgerade und erstellen Sie die Graphik dazu!


 Lösung siehe Abschnitt 4.2.4

1.2.5. Aufgabe 1.2.5 Kinder und Geschwister

- i** Man befragt 5 verheiratete Personen im Alter von mindestens 50 Jahren nach der Anzahl ihrer eigenen Kinder (X) und nach der Anzahl ihrer Geschwister (Y). Die Antworten lauten:

Person	1	2	3	4	5
Anzahl eigener Kinder (X)	1	0	3	2	1
Anzahl eigener Geschwister (Y)	0	1	4	1	2

- Berechnen Sie den Korrelationskoeffizienten r
- Berechnen Sie die Gleichung der Regressionsgeraden und erstellen Sie die Graphik dazu!
- Was geschieht mit r und mit der Regressionsgeraden, falls Sie die Angaben der 3. Person streichen und dann die Auswertung wiederholen?

 Lösung siehe Abschnitt 4.2.5

2. Aufgaben für geübte User:innen


2.1. Objekte in R

2.1.1. Aufgabe 2.1.1 Hogwarts-Kurse

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


 Tabelle 1

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

 Tabelle 2:


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

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

 Lösung siehe Abschnitt 5.1.1

2.1.2. Aufgabe 2.1.2 Aufnahme und Entlassung

- i** Im Datensatz `Krankenhaus.RData`¹ sind die Aufnahme- und Entlassungsdaten von Patienten eines Krankenhauses enthalten, die an einer bestimmten Krankheit leiden.
- a) Laden Sie den Datensatz `Krankenhaus.RData` in Ihre R-Session.
 - b) Ein Variablenname enthält einen Tippfehler. Reparieren Sie auch die Datenklassen der Variablen. Entfernen Sie alle Einträge mit ungültigen Zeitstempeln.
 - c) Erstellen Sie die neue Variable `Liegedauer`, welche die Aufenthaltsdauer in Tagen beinhaltet.
 - d) Über welchen Zeitraum wurden die Daten erhoben?
 - e) Klassieren Sie die Daten der Aufnahme mit einer neuen Variable `Kalenderjahr`.
 - f) Klassieren Sie die Daten der Entlassung je mit einer neuen Variable `Wochentag` und `Monat`.


 Lösung siehe Abschnitt 5.1.2

¹ siehe <https://www.produnis.de/R/data/Krankenhaus.RData>

2.2. Datensätze auswerten

2.2.1. Aufgabe 2.2.1 Aufnahme und Entlassung

- i** Im Datensatz `Krankenhaus.RData`² sind die Aufnahme- und Entlassungsdaten von Patienten eines Krankenhauses enthalten, die an einer bestimmten Krankheit leiden.
- a) Laden Sie den Datensatz `Krankenhaus.RData` in Ihre R-Session, korrigieren Sie den Tippfehler der Variable `ALter`, reparieren Sie die Datenklassen der Variablen und entfernen Sie alle Einträge mit ungültigen Zeitstempeln.
 - b) Plotten Sie die absoluten Häufigkeiten der Aufnahmen und Entlassungen pro Kalendertag. Was fällt Ihnen auf?
 - c) Plotten Sie die durchschnittlichen absoluten Häufigkeiten an täglichen Aufnahmen und Entlassungen pro Wochentag. Was fällt Ihnen auf?
 - d) Plotten Sie die durchschnittlichen absoluten Häufigkeiten an täglichen Aufnahmen und Entlassungen pro Monat sowie die absoluten Häufigkeiten pro Tagesstunde.
 - e) Erstellen Sie ein Poissonregressionsmodell für die Anzahl der täglichen Aufnahmen erklärt durch den Wochentag. Ist das Modell überdispersioniert? Wieviele Aufnahmen sind an einem Dienstag und an einem Sonntag zu erwarten?
 - f) Fügen Sie den Monat als weiteren Prädiktor hinzu. Wird das Modell dadurch besser? Wieviele Aufnahmen sind an einem Donnerstag im Mai zu erwarten, und wieviele im September?
 - g) Wie groß ist die Wahrscheinlichkeit, dass an einem Mittwoch im Mai 10 Patienten aufgenommen werden?
 - h) Wie groß ist die Wahrscheinlichkeit, dass an einem Mittwoch im Mai zwischen 4 und 7 Patienten aufgenommen werden?
 - i) Wie groß ist die Wahrscheinlichkeit, dass an einem Montag im Januar maximal 2 Patienten aufgenommen werden?
 - j) Erzeugen Sie ein Histogramm des Alters der Probanden. Was fällt Ihnen auf? Korrigieren Sie wenn nötig die Daten. Ist das Alter der Probanden normalverteilt?
 - k) Stellen Sie das Alter der Männern und Frauen tabellarisch und graphisch dar. Unterscheidet sich das Alter der Probanden zwischen Männern und Frauen?
 - l) Ist der Unterschied signifikant?
 - m) Ab welchem Alter sind 10% der Männer älter als dieser Wert?
 - n) Ab welchem Alter sind 80% der Frauen jünger als dieser Wert?
 - o) Wie groß ist die mittlere Liegedauer in Tagen? Stellen Sie die Liegedauer mittels Kennwerten sowie graphisch dar. Was fällt Ihnen auf?
 - p) Wie viel Prozent der Patienten haben eine Liegedauer von mehr als 7 Tagen?
 - q) Unterscheiden sich Männer und Frauen hinsichtlich der Liegedauer? Stellen Sie den Unterschied ebenfalls tabellarisch und graphisch dar.
 - r) Ist der Unterschied der Liegedauer zwischen Männern und Frauen signifikant?

 Lösung siehe Abschnitt 5.2.1

²siehe <https://www.produnis.de/R/data/Krankenhaus.RData>

3. Aufgaben für fortgeschrittene Anwender:innen

3.1. Objekte in R

3.1.1. Aufgabe 3.1.1 Hogwarts-Kurse

i

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



Lösung siehe Abschnitt 6.1.1

3.2. Datensätze auswerten

3.2.1. Aufgabe 3.2.1 Kurse

i

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



Lösung siehe Abschnitt 6.2.1

Teil II.

Lösungswege

4. Lösungswege zu den Aufgaben für Einsteiger:innen

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

4.1. Lösungen zu Objekten in R

4.1.1. Lösung zur Aufgabe 1.1.1 Vektoren

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

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

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

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

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

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

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

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

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

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

4.1.2. Lösung zur Aufgabe 1.1.2 Zufallsvektoren

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

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

```
[1] 219 322 332 16 411 471 366 73 359 477 414 362 376 392 363 446 68 325
[19] 18 279 439 77 32 55 393 334 403 84 482 25 288 250 151 379 255 413
[37] 335 453 259 162 306 20 207 159 386 460 180 217 252 75 157 317 338 247
[55] 324 405 300 375 44 404 198 307 231 313 177 341 226 351 449 24 368 244
[73] 53 263 94 190 406 223 242 142 65 367 158 450 330 224 305 469 500 184
[91] 276 402 444 118 350 188 51 22 101 35 76 239 106 134 345 164 126 353
[109] 256 456 67 195 298 410 248 149 339 390 296 369 163 50 34 495 148 494
[127] 243 3 343 287 491 461 431 128 418 138 227 415 119 365 112 135 221 271
[145] 445 49 486 435 315 204 290 108 110 206 487 213 381 165 197 268 459 354
[163] 297 6 92 245 265 498 103 205 70 37 154 179 311 39 286 370 382 105
[181] 291 289 212 199 270 480 170 13 257 331 230 401 201 46 425 186 131 133
[199] 100 452
```

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

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

```
[1] 401 400 189 3 395 474 89 74 465 67 340 292 99 356 397 224 461 310
[19] 267 101 73 162 121 35 13 153 94 231 411 86 409 372 407 444 285 500
[37] 329 399 370 185 471 89 216 156 115 121 460 382 320 287 141 364 396 196
[55] 412 141 27 200 375 113 22 177 146 156 492 319 244 33 45 101 63 281
[73] 181 362 363 102 102 50 478 426 405 469 181 129 275 10 484 347 456 457
[91] 361 306 232 56 430 148 287 6 209 39 417 195 129 15 436 378 66 91
```

```
[109] 110 392 175 471 452 217 258 394 437 97 109 21 378 126 414 118 103 334
[127] 326 356 257 325 256 131 407 102 308 240 29 467 488 194 364 236 422 184
[145] 43 59 282 222 342 404 498 228 170 460 350 291 308 250 232 353 33 153
[163] 363 345 16 272 226 138 417 470 418 64 356 176 185 11 55 279 268 452
[181] 93 321 75 297 394 270 477 193 103 337 238 279 303 264 64 3 383 114
[199] 420 387
```

4.1.3. Lösung zur Aufgabe 1.1.3 Krankenhausaufenthalte

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

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

#anschauen
KHAufenthalte

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

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

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

#anschauen
KHAufenthalte

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

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

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

#anschauen
KHAufenthalte

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


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

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

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

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

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

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

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

4.1.4. Lösung zur Aufgabe 1.1.4 Größe und Gewicht

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

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

```
# anzeigen  
Groesse
```

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

```
Gewicht
```

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

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

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

```
# anzeigen  
Kilogramm
```

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

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

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

```
# anzeigen  
BMI
```

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

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

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

# anzeigen
df
```

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

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

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

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

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

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

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

4.1.5. Lösung zur Aufgabe 1.1.5 ordinale Faktoren

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

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

```
# anzeigen
Monate
```

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

Wir können uns aber auch ein bisschen Schreibarbeit ersparen.

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

```
# anzeigen
Monate
```

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

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

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

# anzeigen
Schulnoten
```

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

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

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

# anzeigen
f
```

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

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

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

# anzeigen
f

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

4.1.6. Lösung zur Aufgabe 1.1.6 Hogwarts-Kurse

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

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

# anzeigen
Kurse
```

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

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

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

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

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

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

```
[1] 13
```

Es sind 13 Kurse in der Liste.

💡 c) Erstellen Sie für jedes Haus ein eigenes Datenframe

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

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

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

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

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

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

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

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

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

# anzeigen
Hogwarts$Kurs
```

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

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

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

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

	Haus	Kurs
9	Ravenclaw	Arithmantik
10	Ravenclaw	Astronomie

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

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

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

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

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

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

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

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

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

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

# anzeigen
sorted
```

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

4.1.7. Lösung zur Aufgabe 1.1.7 Datentabelle

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

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

# anzeigen
chol
```

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

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

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

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

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

# anzeigen
chol
```

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

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

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

# zusammenfügen
chol <- rbind(chol, neu)

# anzeigen
chol
```

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

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

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

# anzeigen
chol
```

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

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

Ein Klassierung kann auf mehrere Weisen erfolgen.

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

# anzeigen
chol
```

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

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

Alternativ kann die cut()-Funktion verwendet werden.

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

# anzeigen
chol
```

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

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

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

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

# anzeigen
male
```

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

	Adipositas
2	Adipositas
3	Normalgewicht
6	Adipositas
7	Übergewicht

4.1.8. Lösung zur Aufgabe 1.1.8 Zusatzpaket

- 💡 a) Installieren Sie das Zusatzpaket jgsbook mit allen Abhängigkeiten

```
# installiere inkl Abhängigkeiten
install.packages("jgsbook", dependencies = TRUE)
```

- 💡 b) Welche Datensätze sind in dem Paket enthalten?

Der folgende Befehl öffnet einen neuen Tab in RStudio:

```
# Zeige die enthaltenen Datensätze graphisch an
data(package = "jgsbook")
```

Für die Ausgabe auf der Konsole können wir so vorgehen.

```
# Zeige die enthaltenen Datensätze auf Konsole
a <- data(package = "jgsbook")
as.data.frame(a$results[, 3:4])
```

	Item	Title
1	Faktorenbogen	Datatable of the Faktorenbogen Example for factor analysis
2	MarioANOVA	Datatable of the SuperMario Example for Friedman-ANOVA
3	Messwiederholung	Datatable of the Messwiederholung Example for ANOVA
4	Pflegeberufe	Matrix of Pflegeberufe by Isfort et al. 2018
5	epa	Datatable of the epa Example
6	mma	Dataset of a work sampling study
7	nw (Nachtwachen)	Dataset of the German Nachtwachen study with labelled variables
8	nw_labelled (nw)	Dataset of the German Nachtwachen study with labelled variables
9	ordinalSample (OrdinalSample)	Datatable of an Ordinal Sample
10	pf8	Dataset of the PF8 example.

💡 c) Speichern Sie den Datensatz pf8 aus dem jgsbook in das Objekt df. Welche Variablen sind im Datensatz enthalten?

```
df <- jgsbook::pf8
```

```
# anzeigen  
str(df)
```

```
'data.frame': 731 obs. of 16 variables:  
 $ Standort      : Factor w/ 5 levels "Rheine","Münster",...: 2 2 2 2 2 2 2 2 2 2 ...  
 $ Alter         : int  18 67 60 61 24 21 59 56 82 52 ...  
 $ Geschlecht    : Factor w/ 3 levels "männlich","weiblich",...: 2 2 2 1 1 2 2 2 1 2 ...  
 $ Größe        : int  172 165 175 182 173 177 168 156 184 166 ...  
 $ Gewicht       : num  69 67 NA 90 68 60 80 60 NA 60 ...  
 $ Bildung       : Factor w/ 7 levels "keinen","Hauptschule",...: 6 3 7 3 6 6 3 4 3 5 ...  
 $ Beruf         : Factor w/ 104 levels "", "Produktionsleiter",...: 46 81 22 13 93 93 6 69 4 ...  
 $ Familienstand : Factor w/ 6 levels "ledig","Partnerschaft",...: 2 4 2 1 1 2 3 4 3 3 ...  
 $ Kinder        : int   0 0 0 0 0 0 0 2 0 1 ...  
 $ Wohnort       : Factor w/ 2 levels "städtisch","ländlich": 2 2 1 2 1 1 2 1 1 1 ...  
 $ Rauchen       : Factor w/ 2 levels "nein","ja": 1 1 1 1 1 2 1 2 1 1 ...  
 $ SportHäufig   : num   NA 2 2 4 4 1 2 1 1 2 ...  
 $ SportMinuten  : num   NA 60 45 120 60 60 45 90 NA 45 ...  
 $ SportWie      : Factor w/ 3 levels "Allein","Gruppe",...: 1 2 3 1 1 2 2 3 2 1 ...  
 $ SportWarum    : Factor w/ 8 levels "0","Vorbeugung",...: 6 2 2 4 3 4 2 2 4 2 ...  
 $ LebenZufrieden: num   5 7 7 2 9 8 5 8 10 8 ...
```

💡 d) Rufen Sie Dokumentation für das jgsbook-Paket auf.

```
help(package = "jgsbook")
```

💡 e) Wenden Sie die Funktion freqTable() aus dem Paket jgsbook auf die Variable df\$Kinder an, ohne das Paket vorher per library() zu aktivieren.

```
# Funktion aufrufen ohne Paket zu laden  
jgsbook::freqTable(df$Kinder)
```

	Wert	Haeufig	Hkum	Relativ	Rkum
1	0	563	563	77.02	77.02
2	1	81	644	11.08	88.10
3	2	60	704	8.21	96.31
4	3	21	725	2.87	99.18
5	4	1	726	0.14	99.32
6	5	1	727	0.14	99.46

4.1.9. Lösung zur Aufgabe 1.1.9 Daten laden

💡 a) Datentabelle.txt

```
# Lese Daten ein
a <- read.table("https://www.produnis.de/R/data/Datentabelle.txt", header=TRUE)
```

```
# Datenklassen anschauen
str(a)
```

```
'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
```

```
# Geschlecht anpassen
a$Geschlecht <- factor(a$Geschlecht)
```

```
# anschaeun
str(a)
```

```
'data.frame':  10 obs. of  4 variables:
 $ Geschlecht: Factor w/ 2 levels "m","w": 1 2 2 1 1 2 2 2 1 1
 $ 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
```

💡 b) anwesenheitnoten.csv

In der Datei werden Dezimalstellen mit “,” und Feldtrenner mit “;” angegeben. Entsprechend lautet der Aufruf von `read.table()`:

```
# Lese Daten ein
b <- read.table("https://www.produnis.de/R/data/anwesenheitnoten.csv",
                header=TRUE, dec=",", sep=";")
```

```
# Datenklassen anschauen
str(b)
```

```
'data.frame':  27 obs. of  2 variables:
 $ Anwesenheit: num  90.9 40.9 100 81.8 0 100 27.3 27.3 54.5 54.5 ...
 $ Note       : num   1.7 5 1 NA 5 2 3.7 5 3.3 NA ...
```

Alle Datenklassen sind korrekt (numerisch).

💡 c) alteDaten.sav

Dateien mit Endung .sav stammen von SPSS.

```
# Lese Daten ein
c <- haven::read_sav("https://www.produnis.de/R/data/alteDaten-kurz.sav")
```

```
# Datenklassen anschauen
str(c)
```

```
tibble [13 x 6] (S3: tbl_df/tbl/data.frame)
 $ v1      : dbl+lbl [1:13]  4,  4,  4,  4,  3,  2,  2,  2,  4,  3, NA, NA, NA
 ..@ label      : chr "Blue Question"
 ..@ format.spss: chr "F1.0"
 ..@ labels     : Named num [1:5] 1 2 3 4 9
 .. ..- attr(*, "names")= chr [1:5] "strongly disagree" "disagree" "agree" "strongly agree"
 $ v2      : dbl+lbl [1:13]  4,  4,  4,  3,  3,  3,  1,  3,  4,  3, NA, NA, NA
 ..@ label      : chr "Red Question"
 ..@ format.spss: chr "F1.0"
 ..@ labels     : Named num [1:5] 1 2 3 4 9
 .. ..- attr(*, "names")= chr [1:5] "strongly disagree" "disagree" "agree" "strongly agree"
 $ v3      : dbl+lbl [1:13]  4,  4,  4,  3,  2,  3,  3,  3,  4,  3, NA, NA, NA
 ..@ label      : chr "Green Question"
 ..@ format.spss: chr "F1.0"
 ..@ labels     : Named num [1:5] 1 2 3 4 9
 .. ..- attr(*, "names")= chr [1:5] "strongly disagree" "disagree" "agree" "strongly agree"
 $ v4      : dbl+lbl [1:13] NA, NA, NA,  4,  2,  2,  2,  3, NA,  4, NA, NA, NA
 ..@ label      : chr "Yellow Question"
 ..@ format.spss: chr "F1.0"
 ..@ labels     : Named num [1:5] 1 2 3 4 9
 .. ..- attr(*, "names")= chr [1:5] "strongly disagree" "disagree" "agree" "strongly agree"
 $ v5      : dbl+lbl [1:13] NA, NA, NA,  4,  4,  4,  1,  3, NA,  4, NA, NA, NA
 ..@ label      : chr "Pink Question"
 ..@ format.spss: chr "F1.0"
 ..@ labels     : Named num [1:5] 1 2 3 4 9
 .. ..- attr(*, "names")= chr [1:5] "strongly disagree" "disagree" "agree" "strongly agree"
 $ thetruth: dbl+lbl [1:13]  4,  4,  4,  4,  4,  4,  4,  3,  1,  4, NA, NA, NA
 ..@ label      : chr "Schalke steigt nächstes Jahr in die dritte Liga ab"
 ..@ format.spss: chr "F1.0"
 ..@ labels     : Named num [1:5] 1 2 3 4 9
 .. ..- attr(*, "names")= chr [1:5] "strongly disagree" "disagree" "agree" "strongly agree"
```

```
# Variable
c$v1
```

```
<labelled<double>[13]>: Blue Question
 [1]  4  4  4  4  3  2  2  2  4  3 NA NA NA
```

Labels:

value	label
1	strongly disagree
2	disagree
3	agree
4	strongly agree
9	no answer

Die Daten sind gelabelt und scheinen ordinalskaliert zu sein.

```
# labels aufschreiben
c.labels <- c("strongly disagree", "disagree", "agree", "strongly agree")
```

```
# Variablen in ordinale Faktoren umwandeln
c$v1 <- factor(c$v1, ordered=TRUE, levels=c(1:4))
c$v2 <- factor(c$v2, ordered=TRUE, levels=c(1:4))
c$v3 <- factor(c$v3, ordered=TRUE, levels=c(1:4))
c$v4 <- factor(c$v4, ordered=TRUE, levels=c(1:4))
c$v5 <- factor(c$v5, ordered=TRUE, levels=c(1:4))
c$thetruth <- factor(c$v1, ordered=TRUE, levels=c(1:4))
```

```
# Levelnamen ändern
levels(c$v1) <- c.labels
levels(c$v2) <- c.labels
levels(c$v3) <- c.labels
levels(c$v4) <- c.labels
levels(c$v5) <- c.labels
levels(c$thetruth) <- c.labels
```

```
# anschauen
str(c)
```

```
tibble [13 x 6] (S3: tbl_df/tbl/data.frame)
 $ v1      : Ord.factor w/ 4 levels "strongly disagree"<...: 4 4 4 4 3 2 2 2 4 3 ...
 $ v2      : Ord.factor w/ 4 levels "strongly disagree"<...: 4 4 4 3 3 3 1 3 4 3 ...
 $ v3      : Ord.factor w/ 4 levels "strongly disagree"<...: 4 4 4 3 2 3 3 3 4 3 ...
 $ v4      : Ord.factor w/ 4 levels "strongly disagree"<...: NA NA NA 4 2 2 2 3 NA 4 ...
 $ v5      : Ord.factor w/ 4 levels "strongly disagree"<...: NA NA NA 4 4 4 1 3 NA 4 ...
 $ thetruth: Ord.factor w/ 4 levels "strongly disagree"<...: 4 4 4 4 3 2 2 2 4 3 ...
```

d) Testdatumdaten.xlsx

Dateien mit Endung .sav stammen von SPSS.

```
# Lese Daten ein
d <- openxlsx::read.xlsx("https://www.produnis.de/R/data/Testdatumdaten.xlsx")
```

```
# Datenklassen anschauen
```

```
str(d)
```

```
tibble [38 x 4] (S3: tbl_df/tbl/data.frame)
```

```
$ Vorname      : chr [1:38] "Anima" "Annika" "Farhad" "Michèle" ...
```

```
$ Geschlecht   : chr [1:38] "weiblich" "weiblich" "männlich" "weiblich" ...
```

```
$ Geburtstag    : chr [1:38] "25.02.2001" "19.10.1995" "10.11.1999" "23.08.1993" ...
```

```
$ Lieblingsfarbe: chr [1:38] "blau" "grün" "gelb" "blau" ...
```

```
# Datenklassen anpassen
```

```
d$Vorname <- factor(d$Vorname)
```

```
d$Geschlecht <- factor(d$Geschlecht)
```

```
d$Lieblingsfarbe <- factor(d$Lieblingsfarbe)
```

```
# Zeitformat
```

```
d$Geburtstag <- lubridate::dmy(d$Geburtstag, tz="CET")
```

```
# anschauen
```

```
str(d)
```

```
tibble [38 x 4] (S3: tbl_df/tbl/data.frame)
```

```
$ Vorname      : Factor w/ 38 levels "Alexander","Anima",...: 2 4 13 30 38 35 10 3 5 15 ...
```

```
$ Geschlecht    : Factor w/ 2 levels "männlich","weiblich": 2 2 1 2 2 2 2 2 2 1 ...
```

```
$ Geburtstag    : POSIXct[1:38], format: "2001-02-25" "1995-10-19" ...
```

```
$ Lieblingsfarbe: Factor w/ 4 levels "blau","gelb",...: 1 3 2 1 3 3 1 4 4 2 ...
```

4.2. Lösungen zur deskriptiven Statistik

4.2.1. Lösung zur Aufgabe 1.2.1 Serumcholesterin

💡 a) Überführen Sie die Daten in ein Datenframe mit der Variable chol.

```
# gebe die Werte ein
serumchol <- c(4.5, 4.9, 7.3, 5.2, 5.8, 6.2, 5.0, 5.6, 6.4, 7.6,
              5.4, 4.4, 6.6, 5.3, 5.7, 4.7, 8.2, 6.7, 4.8, 5.9)

# erstelle Datenframe
df <- data.frame(chol= serumchol)
```

💡 b) Klassieren Sie die Serumcholesterinwerte.

Die Klassierung erfolgt entweder “von Hand”:

```
# erstelle Werteklassen in eigener Variablenspalte
df$cholklass[df$chol < 5] <- "4.0-4.9" # alle Werte kleiner 5
df$cholklass[df$chol < 6 & df$chol > 4.9] <- "5.0-5.9" # Werte kleiner 6 und größer 4
df$cholklass[df$chol < 7 & df$chol > 5.9] <- "6.0-6.9" # Werte kleiner 7 und größer 5
df$cholklass[df$chol < 8 & df$chol > 6.9] <- "7.0-7.9" # Werte kleiner 8 und größer 6
df$cholklass[df$chol < 9 & df$chol > 7.9] <- "8.0-8.9" # Werte kleiner 9 und größer 7
df$cholklass <- factor(df$cholklass, ordered=T)
# neue Variable anschauen
df$cholklass
```

```
[1] 4.0-4.9 4.0-4.9 7.0-7.9 5.0-5.9 5.0-5.9 6.0-6.9 5.0-5.9 5.0-5.9 6.0-6.9
[10] 7.0-7.9 5.0-5.9 4.0-4.9 6.0-6.9 5.0-5.9 5.0-5.9 4.0-4.9 8.0-8.9 6.0-6.9
[19] 4.0-4.9 5.0-5.9
Levels: 4.0-4.9 < 5.0-5.9 < 6.0-6.9 < 7.0-7.9 < 8.0-8.9
```

...oder mittels cut().

```
df$cholklass2 <- cut(df$chol, breaks=c(4:9),
                    right=FALSE,
                    ordered_result = TRUE)

# anzeigen
df
```

	chol	cholklass	cholklass2
1	4.5	4.0-4.9	[4,5)
2	4.9	4.0-4.9	[4,5)
3	7.3	7.0-7.9	[7,8)
4	5.2	5.0-5.9	[5,6)
5	5.8	5.0-5.9	[5,6)
6	6.2	6.0-6.9	[6,7)

7	5.0	5.0-5.9	[5,6)
8	5.6	5.0-5.9	[5,6)
9	6.4	6.0-6.9	[6,7)
10	7.6	7.0-7.9	[7,8)
11	5.4	5.0-5.9	[5,6)
12	4.4	4.0-4.9	[4,5)
13	6.6	6.0-6.9	[6,7)
14	5.3	5.0-5.9	[5,6)
15	5.7	5.0-5.9	[5,6)
16	4.7	4.0-4.9	[4,5)
17	8.2	8.0-8.9	[8,9)
18	6.7	6.0-6.9	[6,7)
19	4.8	4.0-4.9	[4,5)
20	5.9	5.0-5.9	[5,6)

💡 c) Erstellen Sie eine ausreichend beschriftete Häufigkeitstabelle mit nicht kumulierten und kumulierten absoluten und relativen Häufigkeiten für die Häufigkeiten in den zuvor erstellten Serumcholesterinklassen.

```
# erzeuge eine Häufigkeitstabelle
jgsbook::freqTable(df$cholklass)
```

	Wert	Haeufig	Hkum	Relativ	Rkum
1	4.0-4.9	5	5	25	25
2	5.0-5.9	8	13	40	65
3	6.0-6.9	4	17	20	85
4	7.0-7.9	2	19	10	95
5	8.0-8.9	1	20	5	100

💡 d) Bestimmen Sie bitte folgende Kenngrößen

```
# allgemein:
summary(df$chol)
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
4.400	4.975	5.650	5.810	6.450	8.200

```
# speziell
psych::describe(df$chol,
  IQR=TRUE,
  skew=FALSE,
  quant = c(.10, 0.25, 0.75, .90)
)
```

	vars	n	mean	sd	median	min	max	range	se	IQR	Q0.1	Q0.25	Q0.75	Q0.9
X1	1	20	5.81	1.06	5.65	4.4	8.2	3.8	0.24	1.48	4.68	4.97	6.45	7.33

```
# Was fehlt noch:  
# Varianz  
var(df$chol)
```

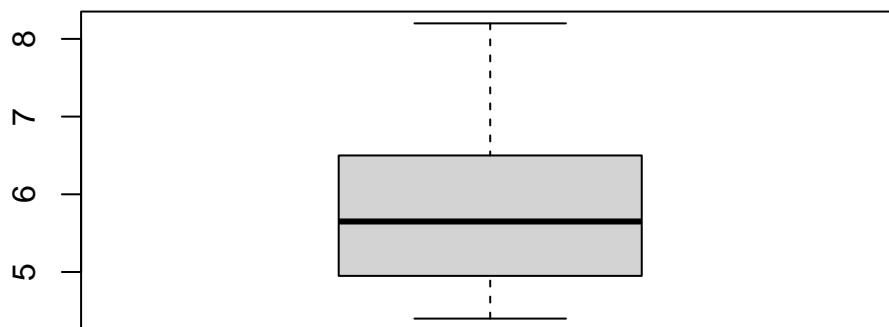
```
[1] 1.124105
```

```
# und Median  
median(df$chol)
```

```
[1] 5.65
```

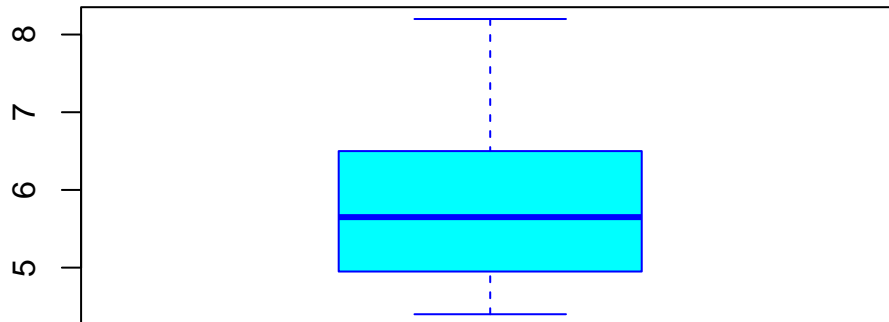
💡 e) Erstellen Sie einen Boxplot der Werte.

```
boxplot(df$chol)
```



```
# hübscher  
boxplot(df$chol, main="Serumcholesterinspiegel in mmol/l",  
        col="cyan", border="blue")
```

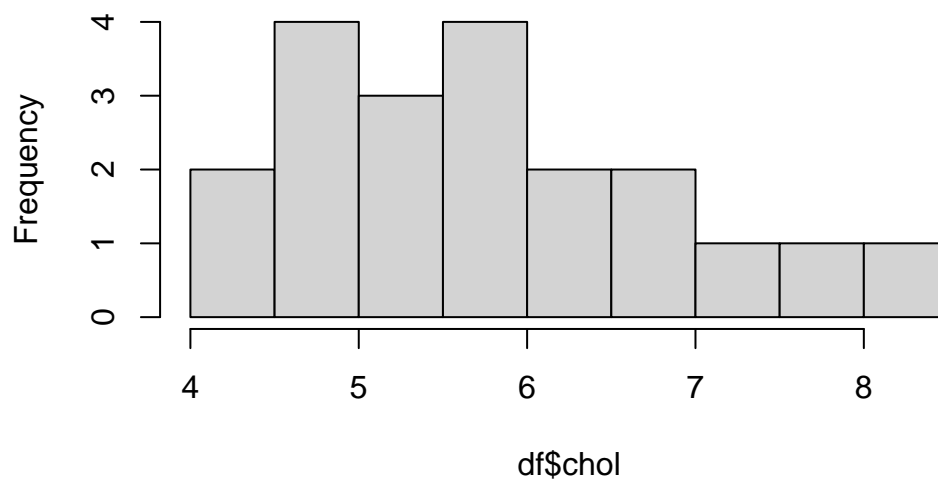
Serumcholesterinspiegel in mmol/l



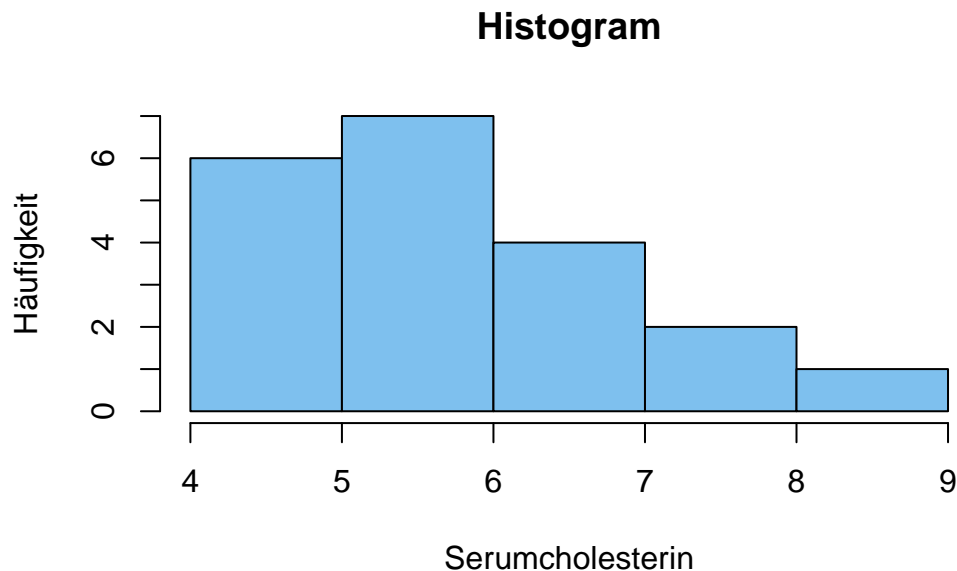
💡 f) Stellen Sie die in b) aufgelisteten absoluten nicht kumulierten Häufigkeiten als Histogramm dar.

```
# Histogram  
hist(df$chol)
```

Histogram of df\$chol



```
# Histogram mit 5 "breaks und etwas hübscher
hist(df$chol, breaks=5,
     col="skyblue2",
     main="Histogram",
     xlab="Serumcholesterin",
     ylab="Häufigkeit")
```



💡 g) Welche Form hat die Verteilung?

```
# "Schiefe" berechnen
psych::skew(df$chol)
```

```
[1] 0.6286707
```

```
# "Spitzigkeit" berechnen
psych::kurtosi(df$chol)
```

```
[1] -0.6340307
```

Die Skewness ist positiv, d.h. die Verteilung ist linksgipflig (aka rechtschief).

Die Kurtosis von -0,63 zeigt an, dass die Daten flacher und breiter als eine Normalverteilung sind.

4.2.2. Lösung zur Aufgabe 1.2.2 Gewichtsreduktion

💡 a) Überführen Sie die Daten in ein Datenframe.

```
alter <- c(4, 7, 8, 9, 11, 12, 13, 14, 15, 16, 16, 20, 20,
          22, 25, 26, 26, 28, 29, 34)
geschlecht <- c(1, 2, 2, 2, 1, 1, 2, 2, 2, 1, 1, 2, 2, 2, 1, 0, 2, 1, 2, 0)

# Erzeuge Datenframe
df <- data.frame(alter, geschlecht)
```

💡 b) Kodieren Sie der Variable "Geschlecht" um

```
# wandle "Geschlecht"-Einträge um
df$geschlecht[df$geschlecht == "0"] <- "divers"
df$geschlecht[df$geschlecht == "1"] <- "männlich"
df$geschlecht[df$geschlecht == "2"] <- "weiblich"

# wandle in factor() um
df$geschlecht <- factor(df$geschlecht)
```

💡 c) Klassieren Sie das Alter der Probanden

Die Altersklassierung erfolgt entweder "von Hand"...

```
# klassiere die Daten in eigener Spaltenvariable
df$alterk[df$alter < 6] <- "0-5" # alle Werte kleiner 6
df$alterk[df$alter < 11 & df$alter > 5] <- "6-10" # Werte kleiner 11 und größer 5
df$alterk[df$alter < 16 & df$alter > 10] <- "11-15" # Werte kleiner 16 und größer 10
df$alterk[df$alter < 21 & df$alter > 15] <- "16-20" # Werte kleiner 21 und größer 15
df$alterk[df$alter < 26 & df$alter > 20] <- "21-25" # Werte kleiner 26 und größer 20
df$alterk[df$alter < 31 & df$alter > 25] <- "26-30" # Werte kleiner 31 und größer 25
df$alterk[df$alter > 30] <- "31-35" # Werte größer 30 werden zu "31-35"

# ordinaler Faktor der Werteklassen
df$alterk <- factor(df$alterk,
                   levels=c("0-5", "6-10", "11-15", "16-20", "21-25",
                             "26-30", "31-35"),
                   ordered=TRUE)
```

... oder per cut()-Funktion.

```
df$alterk2 <- cut(df$alter, breaks = seq(0,35,by=5),
                 ordered=TRUE)

#anzeigen
df
```

```
alter geschlecht alterk alterk2
```

1	4	männlich	0-5	(0,5]
2	7	weiblich	6-10	(5,10]
3	8	weiblich	6-10	(5,10]
4	9	weiblich	6-10	(5,10]
5	11	männlich	11-15	(10,15]
6	12	männlich	11-15	(10,15]
7	13	weiblich	11-15	(10,15]
8	14	weiblich	11-15	(10,15]
9	15	weiblich	11-15	(10,15]
10	16	männlich	16-20	(15,20]
11	16	männlich	16-20	(15,20]
12	20	weiblich	16-20	(15,20]
13	20	weiblich	16-20	(15,20]
14	22	weiblich	21-25	(20,25]
15	25	männlich	21-25	(20,25]
16	26	divers	26-30	(25,30]
17	26	weiblich	26-30	(25,30]
18	28	männlich	26-30	(25,30]
19	29	weiblich	26-30	(25,30]
20	34	divers	31-35	(30,35]

💡 d) Bestimmen Sie folgende Stichprobenkennzahlen für das Merkmal 'Alter'.

```
# allgemein
summary(df$alter)
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
4.00	11.75	16.00	17.75	25.25	34.00

```
# Minimum
min(df$alter)
```

```
[1] 4
```

```
# Perzentile und Quartile
quantile(df$alter, probs = c(0.05, 0.25, 0.75, 0.95))
```

5%	25%	75%	95%
6.85	11.75	25.25	29.25

```
# Perzentile und Quartile
# mit SPSS-Rechenmethode (type=6)
quantile(df$alter, probs = c(0.05, 0.25, 0.75, 0.95), type=6)
```

5%	25%	75%	95%
4.15	11.25	25.75	33.75

```
# Median
median(df$alter)
```

```
[1] 16
```

```
# ar.Mittel
mean(df$alter)
```

```
[1] 17.75
```

```
# Maximum
max(df$alter)
```

```
[1] 34
```

```
# Interquartilsabstand
IQR(df$alter)
```

```
[1] 13.5
```

```
# Interquartilsabstand
# SPSS-Rechenmethode (type=6)
IQR(df$alter, type=6)
```

```
[1] 14.5
```

Berechne (fast) alles auf einmal:

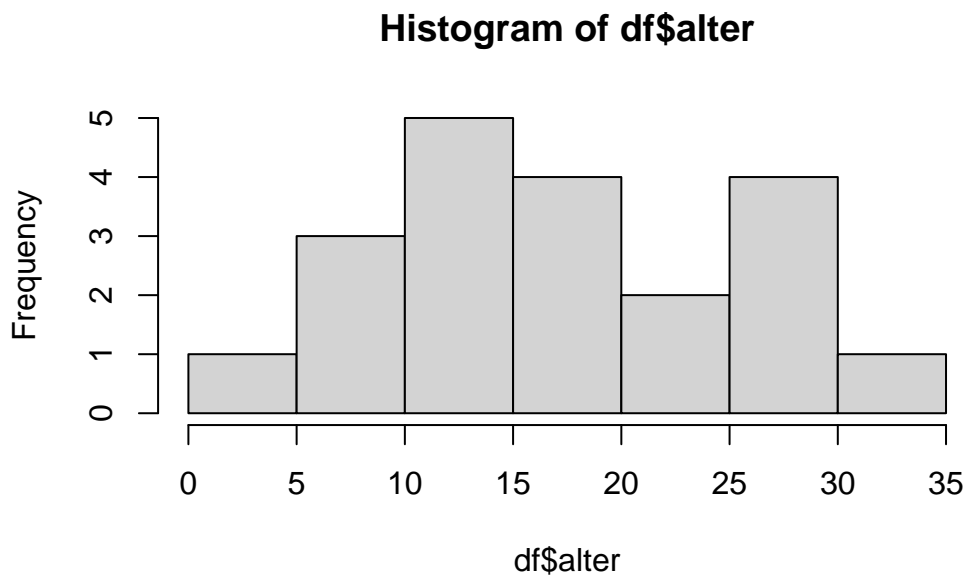
```
# oder einfach
psych::describe(df$alter,
  quant = c(0.05, 0.25, 0.75, 0.95),
  skew=FALSE,
  IQR=TRUE)
```

	vars	n	mean	sd	median	min	max	range	se	IQR	Q0.05	Q0.25	Q0.75	Q0.95
X1	1	20	17.75	8.33	16	4	34	30	1.86	13.5	6.85	11.75	25.25	29.25

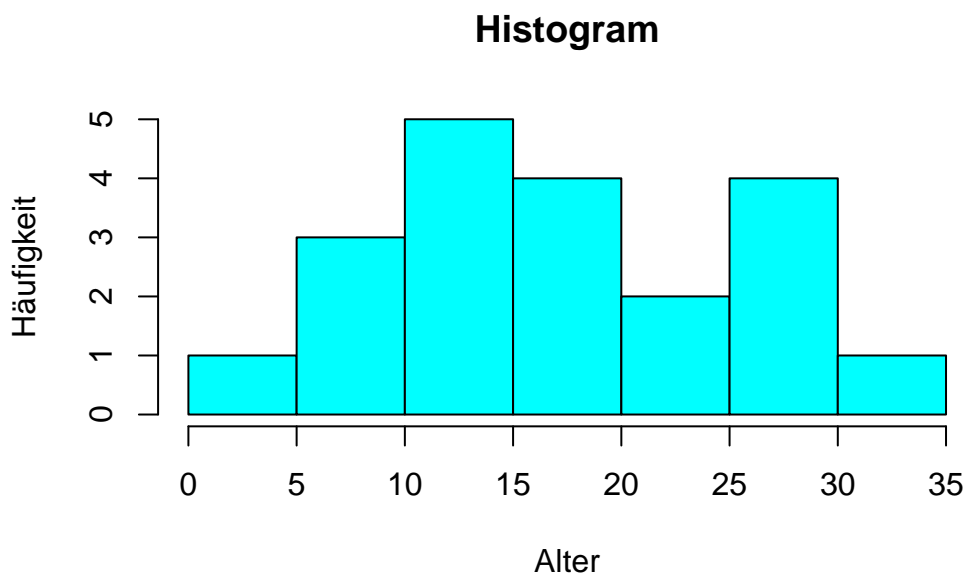
💡 e) Zeichnen Sie ein Histogramm und ein Balkendiagramm für die nicht kumulierten absoluten Häufigkeiten zur Anzahl der Studienteilnehmer in den zuvor gebildeten Altersklassen.

Die Funktion `hist()` kann nur metrische Daten verarbeiten. Daher nehmen wir die Variable “alter” (und nicht “alterk”) und stellen die Abstände auf 5 (Jahre).

```
# Histogramm geht mit R-base hist() nur bei metrischen Daten!!
# Die Werteklassen können per "breaks"-Parameter angegeben werden.
hist(df$alter)
```



```
# etwas hübscher
hist(df$alter, breaks = 5, col="cyan",
      main="Histogram",
      xlab="Alter",
      ylab="Häufigkeit") # 5er-Schritte
```



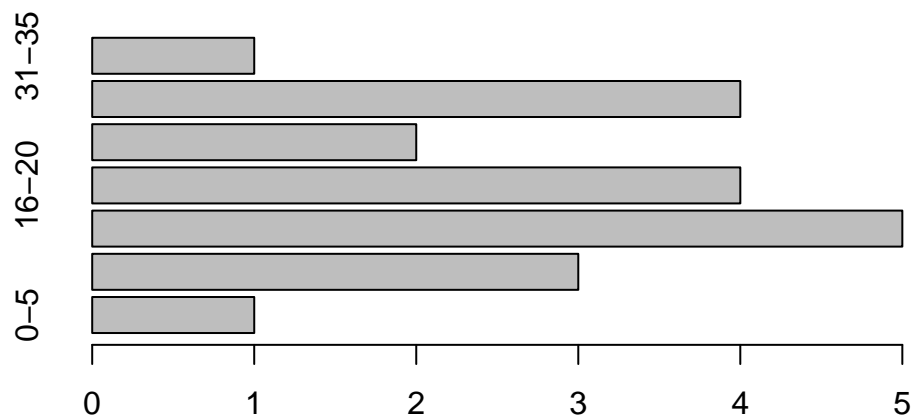
Für das Balkendiagramm nutzen wir die Funktion `table()` auf die Variable "alterk".

```
# Häufigkeitstabelle von "alterk"
table(df$alterk)
```

0-5	6-10	11-15	16-20	21-25	26-30	31-35
1	3	5	4	2	4	1

```
# Balkendiagramm
```

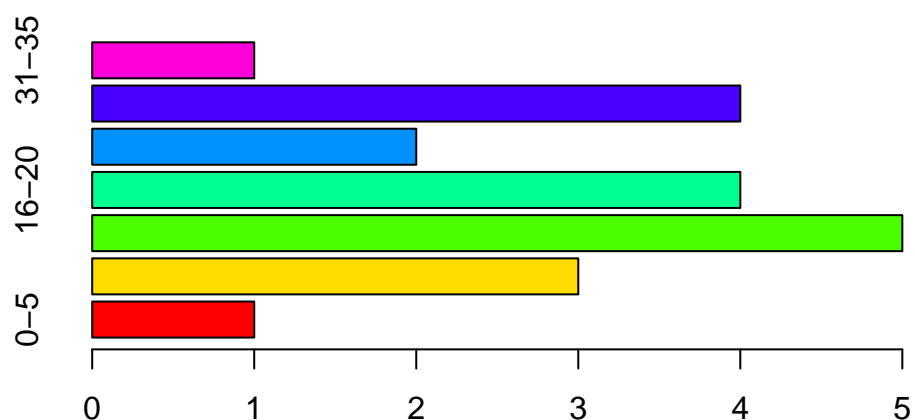
```
barplot(table(df$alterk), horiz = TRUE)
```



```
# etwas hübscher
```

```
barplot(table(df$alterk), horiz = TRUE,
        col=rainbow(7),
        main="Häufigkeiten der Altersklassen")
```

Häufigkeiten der Altersklassen



💡 f) Erstellen Sie eine Kontingenztafel zur gleichzeitigen Darstellung der beiden Merkmale Altersgruppe und Geschlecht.

```
# Kontingenztafel
# entweder mit table()
table(df$alterk, df$geschlecht)
```

	divers	männlich	weiblich
0-5	0	1	0
6-10	0	0	3
11-15	0	2	3
16-20	0	2	2
21-25	0	1	1
26-30	1	1	2
31-35	1	0	0

```
# oder mit xtabs()
xtabs(~df$alterk+df$geschlecht)
```

	df\$geschlecht		
df\$alterk	divers	männlich	weiblich
0-5	0	1	0
6-10	0	0	3
11-15	0	2	3
16-20	0	2	2
21-25	0	1	1
26-30	1	1	2
31-35	1	0	0

```
# in Dezimal-Prozent
prop.table(table(df$alterk, df$geschlecht))
```

	divers	männlich	weiblich
0-5	0.00	0.05	0.00
6-10	0.00	0.00	0.15
11-15	0.00	0.10	0.15
16-20	0.00	0.10	0.10
21-25	0.00	0.05	0.05
26-30	0.05	0.05	0.10
31-35	0.05	0.00	0.00

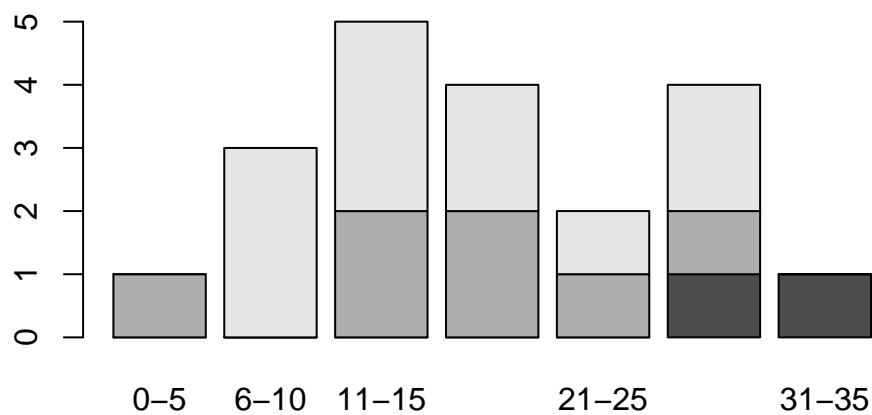
```
# in Prozent
prop.table(table(df$alterk, df$geschlecht))*100
```

	divers	männlich	weiblich
0-5	0	5	0
6-10	0	0	15
11-15	0	10	15
16-20	0	10	10
21-25	0	5	5
26-30	5	5	10
31-35	5	0	0

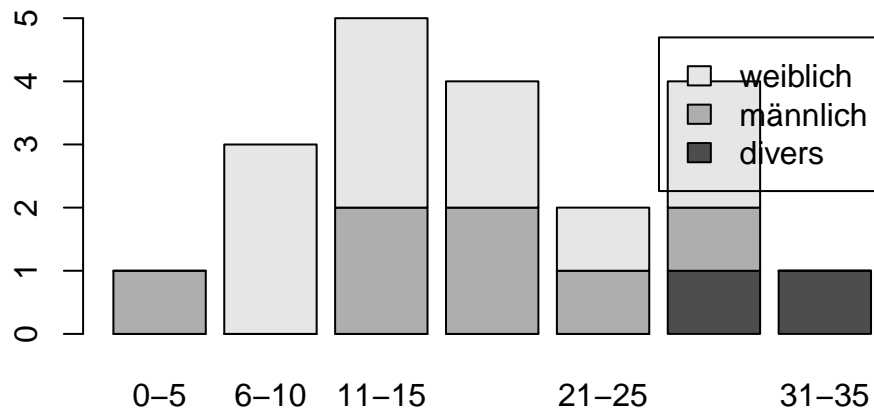
💡 g) Stellen Sie die Häufigkeitsverteilung der beiden Merkmale Altersgruppe und Geschlecht in einer geeigneten Graphik dar.

Geeignet ist ein geschichtetes Barplot.

```
# Barplot
barplot(table(df$geschlecht, df$alterk))
```

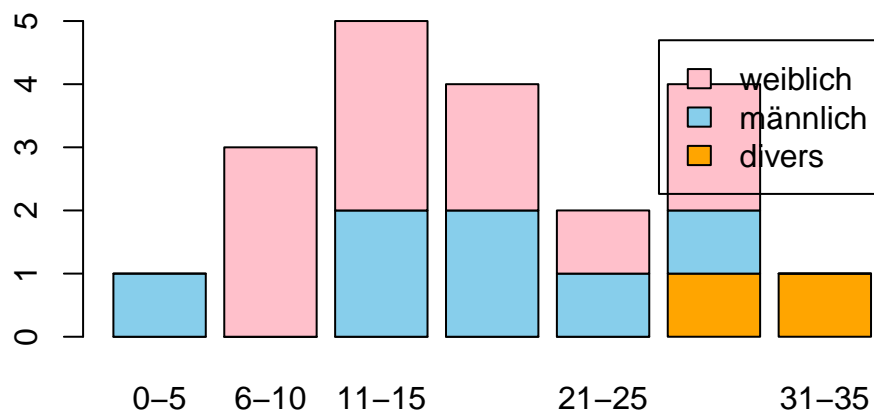


```
# mit Legendenbox
barplot(table(df$geschlecht, df$alterk), legend.text = levels(df$geschlecht))
```

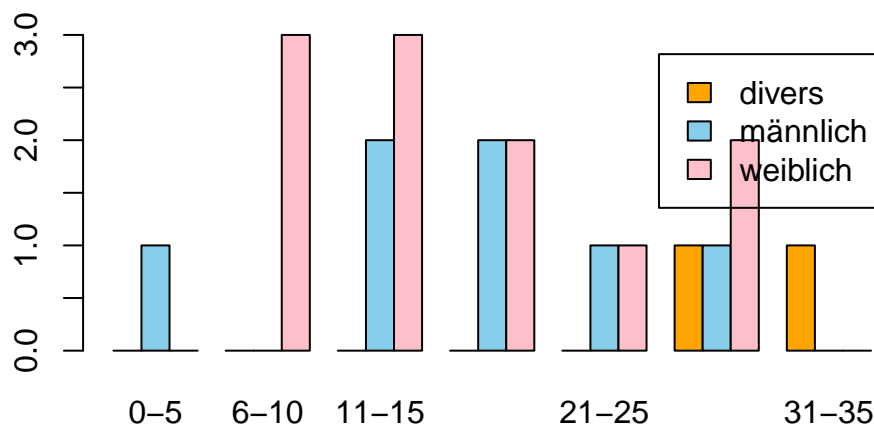


```
# hübscher
barplot(table(df$geschlecht, df$alterk),
        legend.text = levels(df$geschlecht),
        main="Altersklassen nach Geschlecht",
        col=c("orange", "skyblue", "pink")
)
```

Altersklassen nach Geschlecht




```
# variante "beside=TRUE"
barplot(table(df$geschlecht, df$alterk),
        legend.text = levels(df$geschlecht),
        beside = TRUE,
        col=c("orange", "skyblue", "pink")
        )
```



4.2.3. Lösung zur Aufgabe 1.2.3 Anscombe-Quartett

💡 a) Laden Sie den Datensatz anscombe in Ihre R-Session.

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

```
# anschauen
str(anscombe)
```

```
'data.frame':  11 obs. of  8 variables:
 $ x1: num  10 8 13 9 11 14 6 4 12 7 ...
 $ x2: num  10 8 13 9 11 14 6 4 12 7 ...
 $ x3: num  10 8 13 9 11 14 6 4 12 7 ...
 $ x4: num   8 8 8 8 8 8 8 19 8 8 ...
 $ y1: num  8.04 6.95 7.58 8.81 8.33 ...
 $ y2: num  9.14 8.14 8.74 8.77 9.26 8.1 6.13 3.1 9.13 7.26 ...
 $ y3: num  7.46 6.77 12.74 7.11 7.81 ...
 $ y4: num  6.58 5.76 7.71 8.84 8.47 7.04 5.25 12.5 5.56 7.91 ...
```

💡 b) Schreiben Sie die 4 Anscombe-Datensätze (x1 bis x4 und y1 bis y4) in 4 neue Datenframes mit den Namen Anscombe1 bis Anscombe4. Die enthaltenen Spalten sollten jeweils x und y heissen.

```
Anscombe1 <- data.frame(x=anscombe$x1, y=anscombe$y1)
Anscombe2 <- data.frame(x=anscombe$x2, y=anscombe$y2)
Anscombe3 <- data.frame(x=anscombe$x3, y=anscombe$y3)
Anscombe4 <- data.frame(x=anscombe$x4, y=anscombe$y4)
```

💡 c) Führen Sie für jedes Datenframe die Berechnungen von Anscombe durch (Mittelwert, Varianz, Korrelation und lineare Regression), wobei Sie Ihre Ergebnisse auf 2 Stellen runden sollen.

```
### Datensatz Anscombe1
# Mittelwert für x, gerundet auf 2 Stellen
round(mean(Anscombe1$x), 2)
```

```
[1] 9
```

```
# Varianz für x
round(var(Anscombe1$x), 2)
```

```
[1] 11
```

```
# Mittelwert für y
round(mean(Anscombe1$y), 2)
```

```
[1] 7.5
```

```
# Varianz für y
round(var(Anscombe1$y), 2)
```

```
[1] 4.13
```

```
# Korrelationskoeffizient
round(cor(Anscombe1$x, Anscombe1$y), 2)
```

```
[1] 0.82
```

```
# Regression
fit <- lm(Anscombe1$y ~ Anscombe1$x)
round(fit$coefficients, 2)
```

```
(Intercept) Anscombe1$x
          3.0          0.5
```

```
### Datensatz Anscombe2
# Mittelwert für x, gerundet auf 2 Stellen
round(mean(Anscombe2$x), 2)
```

```
[1] 9
```

```
# Varianz für x  
round(var(Anscombe2$x), 2)
```

```
[1] 11
```

```
# Mittelwert für y  
round(mean(Anscombe2$y), 2)
```

```
[1] 7.5
```

```
# Varianz für y  
round(var(Anscombe2$y), 2)
```

```
[1] 4.13
```

```
# Korrelationskoeffizient  
round(cor(Anscombe2$x, Anscombe2$y), 2)
```

```
[1] 0.82
```

```
# Regression  
fit <- lm(Anscombe2$y ~ Anscombe2$x)  
round(fit$coefficients, 2)
```

```
(Intercept) Anscombe2$x  
3.0 0.5
```

```
### Datensatz Anscombe3  
# Mittelwert für x, gerundet auf 2 Stellen  
round(mean(Anscombe3$x), 2)
```

```
[1] 9
```

```
# Varianz für x  
round(var(Anscombe3$x), 2)
```

```
[1] 11
```

```
# Mittelwert für y  
round(mean(Anscombe3$y), 2)
```

```
[1] 7.5
```

```
# Varianz für y  
round(var(Anscombe3$y), 2)
```

```
[1] 4.12
```

```
# Korrelationskoeffizient  
round(cor(Anscombe3$x, Anscombe3$y), 2)
```

```
[1] 0.82
```

```
# Regression  
fit <- lm(Anscombe3$y ~ Anscombe3$x)  
round(fit$coefficients, 2)
```

```
(Intercept) Anscombe3$x  
3.0 0.5
```

```
### Datensatz Anscombe4  
# Mittelwert für x, gerundet auf 2 Stellen  
round(mean(Anscombe4$x), 2)
```

```
[1] 9
```

```
# Varianz für x  
round(var(Anscombe4$x), 2)
```

```
[1] 11
```

```
# Mittelwert für y  
round(mean(Anscombe4$y), 2)
```

```
[1] 7.5
```

```
# Varianz für y  
round(var(Anscombe4$y), 2)
```

```
[1] 4.12
```

```
# Korrelationskoeffizient  
round(cor(Anscombe4$x, Anscombe4$y), 2)
```

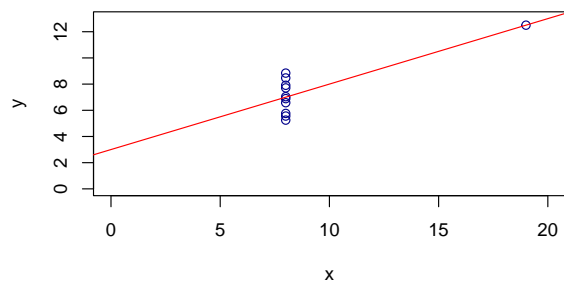
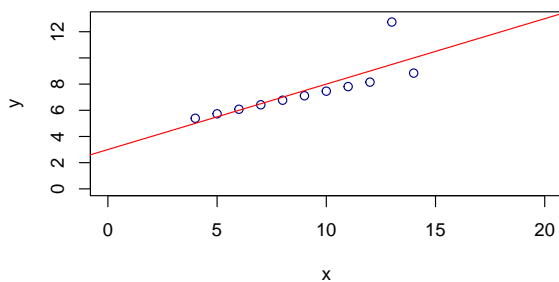
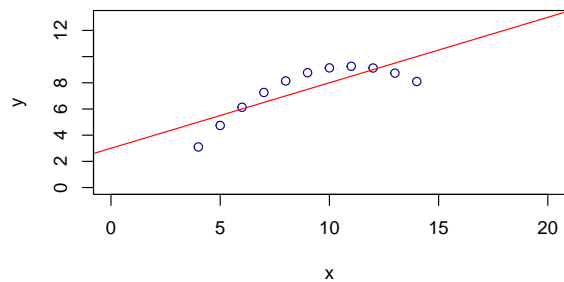
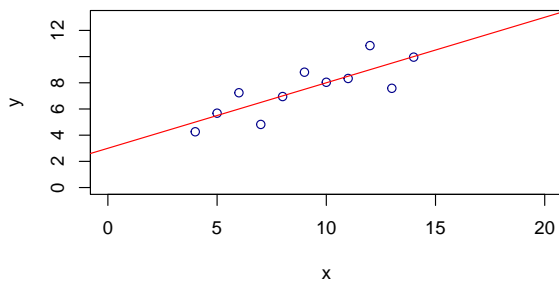
```
[1] 0.82
```

```
# Regression  
fit <- lm(Anscombe4$y ~ Anscombe4$x)  
round(fit$coefficients, 2)
```

```
(Intercept) Anscombe4$x  
3.0 0.5
```

💡 d) Erzeugen Sie die 4 Anscombe-Diagramme (Punktwolke und Regressionsgerade) mit der `plot()`-Funktion, und hübschen Sie die Plots mit etwas Farbe auf.

```
# Datensatz Anscombe1
plot(Anscombe1$x, Anscombe1$y,
     xlim = c(0,20),  xlab="x",
     ylim = c(0,13),  ylab="y",
     col="darkblue")
abline(lm(Anscombe1$y ~ Anscombe1$x), col="red")
# Datensatz Anscombe2
plot(Anscombe2$x, Anscombe2$y,
     xlim = c(0,20),  xlab="x",
     ylim = c(0,13),  ylab="y",
     col="darkblue")
abline(lm(Anscombe2$y ~ Anscombe2$x), col="red")
# Datensatz Anscombe3
plot(Anscombe3$x, Anscombe3$y,
     xlim = c(0,20),  xlab="x",
     ylim = c(0,13),  ylab="y",
     col="darkblue")
abline(lm(Anscombe3$y ~ Anscombe3$x), col="red")
# Datensatz Anscombe4
plot(Anscombe4$x, Anscombe4$y,
     xlim = c(0,20),  xlab="x",
     ylim = c(0,13),  ylab="y",
     col="darkblue")
abline(lm(Anscombe4$y ~ Anscombe4$x), col="red")
```



💡 e) Erzeugen Sie die 4 Anscombe-Diagramme mittels `ggplot()`, wobei alle 4 Diagramme mit einem Plotaufruf erzeugt werden sollen. Dies geht am einfachsten, wenn der Datensatz im Tidy-Data-Format (long table) vorliegt.

```
## Tidy-Longtable erzeugen
# Gruppen separieren
Anscombe1 <- data.frame(x=anscombe$x1, y=anscombe$y1, Gruppe="Anscombe1")
Anscombe2 <- data.frame(x=anscombe$x2, y=anscombe$y2, Gruppe="Anscombe2")
Anscombe3 <- data.frame(x=anscombe$x3, y=anscombe$y3, Gruppe="Anscombe3")
Anscombe4 <- data.frame(x=anscombe$x4, y=anscombe$y4, Gruppe="Anscombe4")

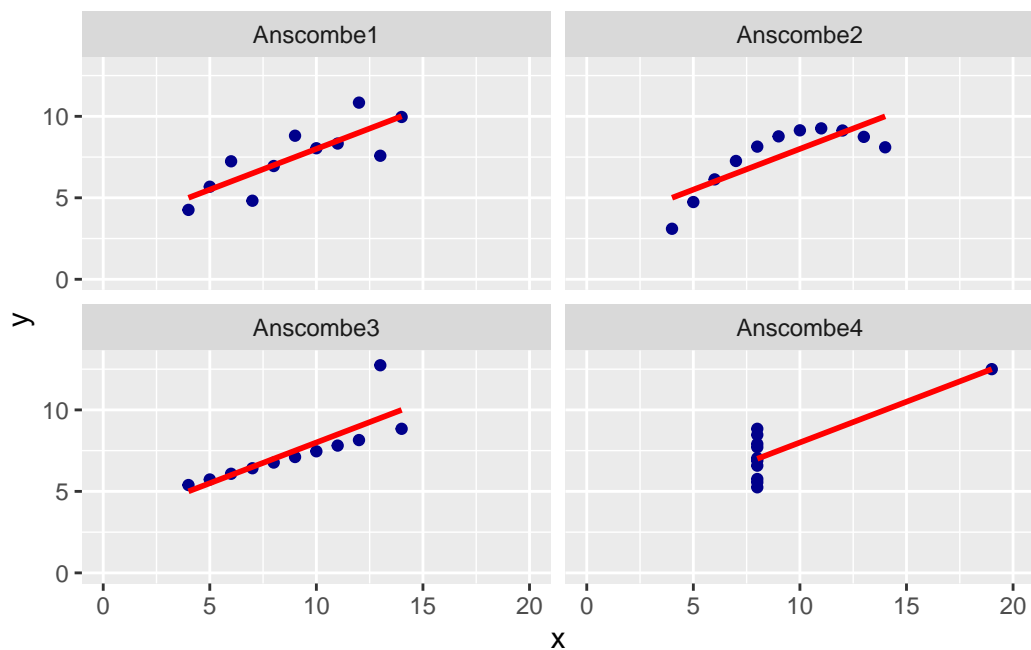
# alles zusammenfügen
df <- rbind(Anscombe1, Anscombe2, Anscombe3, Anscombe4)

# anschauen
str(df)
```

```
'data.frame':  44 obs. of  3 variables:
 $ x      : num  10  8 13  9 11 14  6  4 12  7 ...
 $ y      : num  8.04 6.95 7.58 8.81 8.33 ...
 $ Gruppe: chr   "Anscombe1" "Anscombe1" "Anscombe1" "Anscombe1" ...
```

```
# plotten
library(ggplot2)
ggplot(df) +
  aes(x=x, y=y) +
  xlim(0,20) +
  ylim(0,13) +
  geom_point(color="darkblue")+
  geom_smooth(method="lm", color="red", se=FALSE) +
  facet_wrap(~ Gruppe)
```

`geom_smooth()` using formula = 'y ~ x'



4.2.4. Lösung zur Aufgabe 1.2.4 Kinder und Wohnräume

💡 a) Berechnen Sie den Korrelationskoeffizienten r

```
df <- data.frame(ehepaar = c(1:5),
                 kinder   = c(0, 2, 3, 0, 1),
                 raeume    = c(1, 4, 3, 2, 3))
```

```
# Korrelation nach Pearson
cor(df$kinder, df$raeume)
```

```
[1] 0.7399401
```

💡 b) Berechnen Sie die Regressionsgerade und erstellen Sie die Graphik dazu!

```
# regressionsmodelle immer in variable speichern
fit <- lm(raeume~kinder, data=df)

# Modellübersicht
summary(fit)
```

Call:

```
lm(formula = raeume ~ kinder, data = df)
```

Residuals:

```
      1      2      3      4      5
-0.8235  0.8824 -0.7647  0.1765  0.5294
```

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	1.8235	0.5683	3.209	0.049 *
kinder	0.6471	0.3396	1.905	0.153

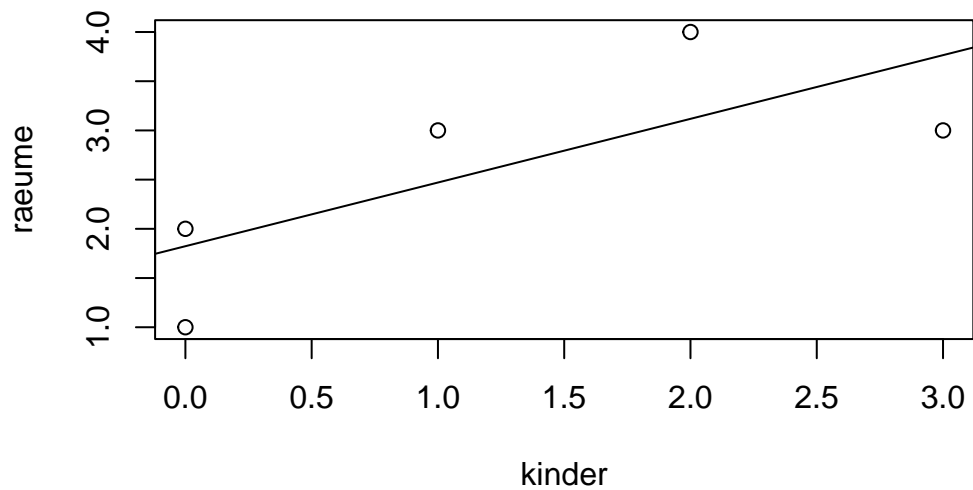
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.8856 on 3 degrees of freedom

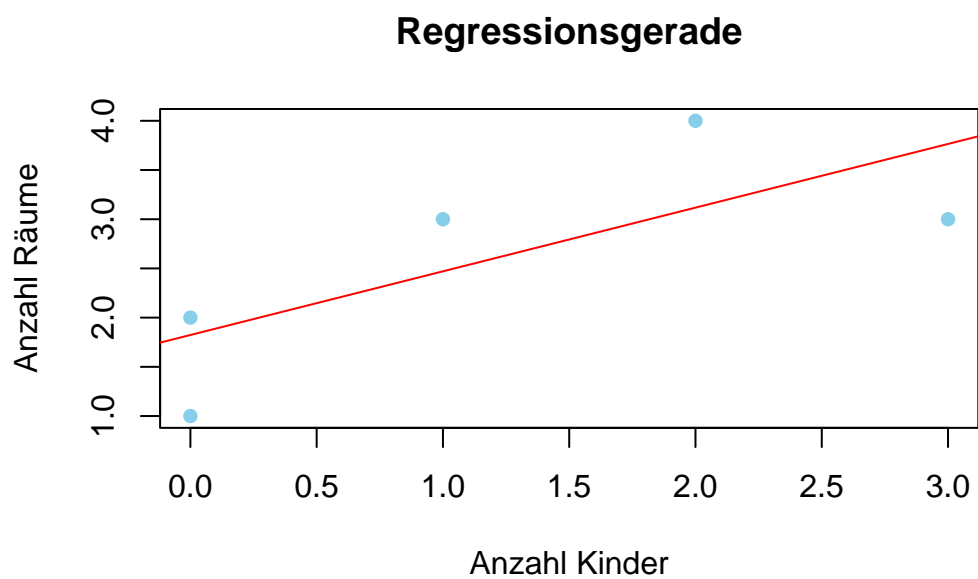
Multiple R-squared: 0.5475, Adjusted R-squared: 0.3967

F-statistic: 3.63 on 1 and 3 DF, p-value: 0.1528

```
# regressionsmodell plotten
plot(raeume~kinder, data=df) # Punktwolke
abline(fit)                  # Regressionsgerade hinzufügen
```

```
# etwas hübscher
plot(raeume~kinder, data=df,
     col="skyblue",
     pch=16,
     main="Regressionsgerade",
     xlab="Anzahl Kinder",
     ylab="Anzahl Räume")
abline(fit, col="red")
```



4.2.5. Lösung zur Aufgabe 1.2.5 Kinder und Geschwister

💡 a) Berechnen Sie den Korrelationskoeffizienten r

```
df <- data.frame(person = c(1:5),
                  kinder  = c(1, 0, 3, 2, 1),
                  geschwister = c(0, 1, 4, 1, 2))

# Korrelation
cor(df$kinder, df$geschwister)
```

```
[1] 0.6939779
```

💡 b) Berechnen Sie die Gleichung der Regressionsgeraden und erstellen Sie die Graphik dazu!

```
# regressionsmodelle immer in variable speichern
fit <- lm(geschwister~kinder, data=df)

# Modellübersicht
summary(fit)
```

Call:

```
lm(formula = geschwister ~ kinder, data = df)
```

Residuals:

1	2	3	4	5
-1.2308	0.6923	0.9231	-1.1538	0.7692

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	0.3077	0.9577	0.321	0.769
kinder	0.9231	0.5529	1.669	0.194

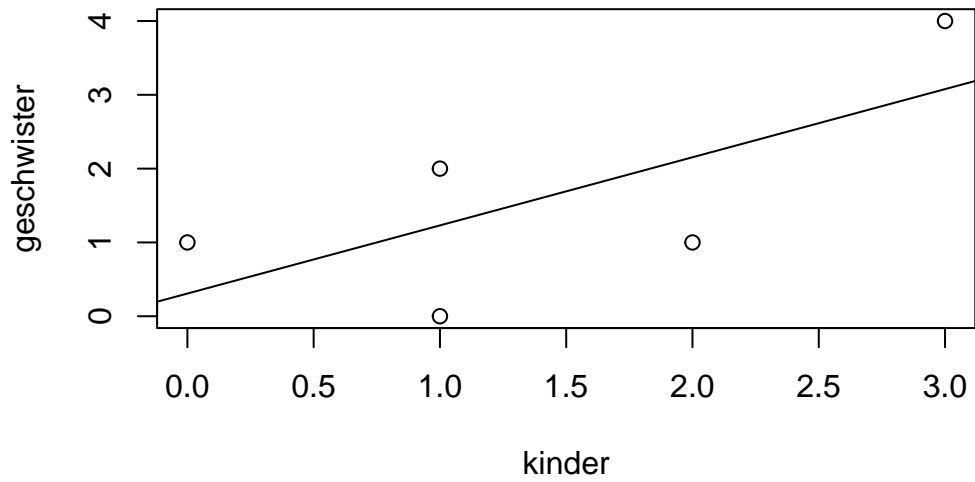
Residual standard error: 1.261 on 3 degrees of freedom

Multiple R-squared: 0.4816, Adjusted R-squared: 0.3088

F-statistic: 2.787 on 1 and 3 DF, p-value: 0.1936

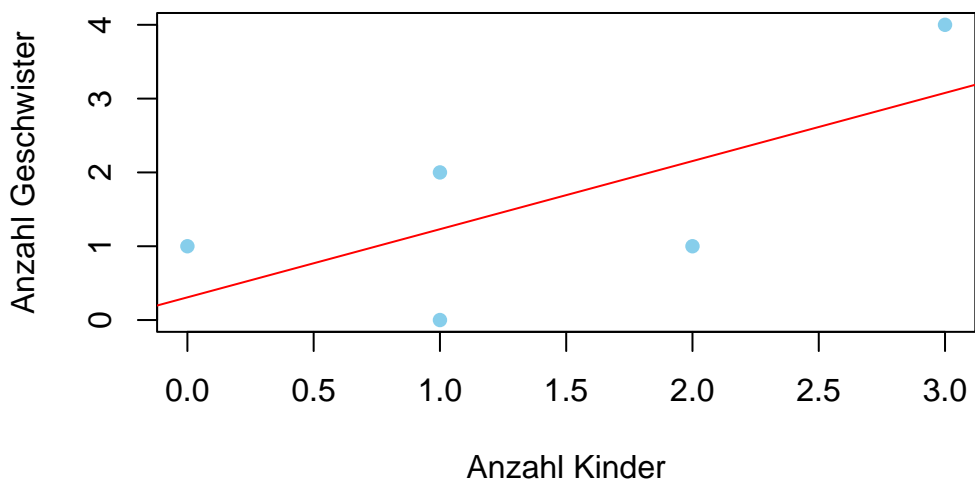
Die Gleichung der Regressionsgeraden lautet $y = 0,3077 + 0,9231 \cdot x$.

```
# regressionsmodell plotten
plot(geschwister~kinder, data=df) # Punktwolke
abline(fit)                        # Regressionsgerade hinzufügen
```



```
# etwas hübscher
plot(geschwister~kinder, data=df,
     col="skyblue",
     pch=16,
     main="Regressionsgerade",
     xlab="Anzahl Kinder",
     ylab="Anzahl Geschwister")
abline(fit, col="red")
```

Regressionsgerade



💡 c) Was geschieht mit r und mit der Regressionsgeraden, falls Sie die Angaben der 3. Person streichen und dann die Auswertung wiederholen?

```
# dritte Person streichen
df <- df[-3,]

# Korrelation
cor(df$kinder, df$geschwister)
```

```
[1] 0
```

```
# Regression
fit <- lm(geschwister~kinder, data=df)

# Modellübersicht
summary(fit)
```

Call:

```
lm(formula = geschwister ~ kinder, data = df)
```

Residuals:

```
      1      2      4      5
-1.000e+00  0.000e+00  5.551e-17  1.000e+00
```

Coefficients:

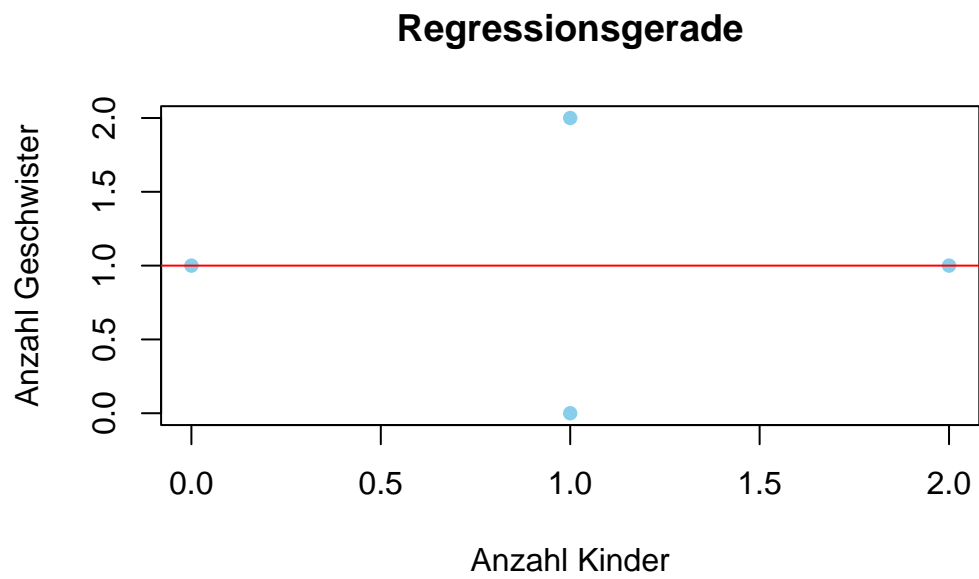
	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	1.000e+00	8.660e-01	1.155	0.368
kinder	-3.925e-17	7.071e-01	0.000	1.000

Residual standard error: 1 on 2 degrees of freedom

Multiple R-squared: 2.465e-32, Adjusted R-squared: -0.5

F-statistic: 4.93e-32 on 1 and 2 DF, p-value: 1

```
# etwas hübscher
plot(geschwister~kinder, data=df,
     col="skyblue",
     pch=16,
     main="Regressionsgerade",
     xlab="Anzahl Kinder",
     ylab="Anzahl Geschwister")
abline(fit, col="red")
```



Wenn die 3. Person aus dem Datensatz entfernt wird, kann kein Zusammenhang zwischen `geschwister` und `kinder` gezeigt werden ($r=0$). Die Regressionsgerade verläuft parallel zur X-Achse, so dass Y für jedes X gleich ist.

5. Lösungswege zu den Aufgaben für geübte User:innen

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

5.1. Lösungen zu Objekten in R

5.1.1. Lösung zur Aufgabe 2.1.1 Hogwarts-Kurse

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

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

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

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

```
tab2
```

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

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

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

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

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

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

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

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

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

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

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

# anzeigen
Hogwarts
```

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

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

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

# anzeigen
Kurse
```

```
# A tibble: 16 x 2
  Haus      Kurs
  <chr>    <fct>
1 Hufflepuff Kräuterkunde
2 Slytherin Zaubertränke
3 Gryffindor Verteidigung gegen die dunklen Künste
4 Ravenclaw Arithmantik
5 Hufflepuff Pflege magischer Geschöpfe
6 Slytherin Zauberkunst
7 Gryffindor Zauberkunst
8 Ravenclaw Astronomie
9 Hufflepuff Geisterkunde
10 Slytherin Dunkle Künste
11 Gryffindor Verwandlung
```



```

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

```

5.1.2. Lösung zur Aufgabe 2.1.2 Aufnahme und Entlassung

💡 a) Laden Sie den Datensatz Krankenhaus.RData in Ihre R-Session.

```

# Lese Daten ein
load("https://www.produnis.de/R/data/Krankenhaus.RData")

```

```

# anschauen
str(St.Gott.Hospital)

```

```

tibble [6,383 x 4] (S3: tbl_df/tbl/data.frame)
 $ Geschlecht: chr [1:6383] "m" "w" "m" "m" ...
 $ Alter      : num [1:6383] 65 75 76 82 71 71 57 82 61 84 ...
 $ Aufnahme   : chr [1:6383] "201509000000" "201510000000" "201606050000" "201606051914" ...
 $ Entlassung: chr [1:6383] "201509000000" "201510000000" "201606052359" "201606061300" ...

```

💡 b) Ein Variablenname enthält einen Tippfehler. Reparieren Sie auch die Datenklassen der Variablen. Entfernen Sie alle Einträge mit ungültigen Zeitstempeln.

```

# Variable ALTER korrigieren
library(dplyr)
kh <- St.Gott.Hospital %>%
  select(Geschlecht, Alter = ALter, Aufnahme, Entlassung)

```

```

# Datenklassen anpassen
# Geschlecht als Faktor
kh$Geschlecht <- factor(kh$Geschlecht)

```

```

# Erzeuge POSIX Zeitobjekte
# CET = Europäische Zeit
library(lubridate)
kh$Aufnahme <- ymd_hm(kh$Aufnahme, tz="CET")
kh$Entlassung <- ymd_hm(kh$Entlassung, tz="CET")

```

```

# anzeigen
str(kh)

```

```

tibble [6,383 x 4] (S3: tbl_df/tbl/data.frame)
 $ Geschlecht: Factor w/ 2 levels "m","w": 1 2 1 1 2 2 1 2 2 2 ...
 $ Alter      : num [1:6383] 65 75 76 82 71 71 57 82 61 84 ...

```

```
$ Aufnahme : POSIXct[1:6383], format: NA NA ...
$ Entlassung: POSIXct[1:6383], format: NA NA ...
```

Durch die Umwandlung der Aufnahme- und Entlassungsdaten sind die Datenreihen mit fehlerhaften oder unvollständigen Zeitstempeln in NAs umgewandelt worden.

```
kh <- kh %>%
  drop_na(Aufnahme, Entlassung)

# anschauen
glimpse(kh)
```

```
Rows: 6,251
Columns: 4
$ Geschlecht <fct> m, m, w, w, m, w, w, w, m, w, w, w, w, m, w, m, m, w, m, m,~
$ Alter      <dbl> 76, 82, 71, 71, 57, 82, 61, 84, 88, 74, 92, 73, 88, 86, 76,~
$ Aufnahme   <dtm> 2016-06-05 00:00:00, 2016-06-05 19:14:00, 2016-06-06 13:39~
$ Entlassung <dtm> 2016-06-05 23:59:00, 2016-06-06 13:00:00, 2016-06-14 13:30~
```

💡 c) Erstellen Sie die neue Variable Liegedauer, welche die Aufenthaltsdauer in Tagen beinhaltet.

```
# Liegedauer berechnen
# entweder...
kh$Liegedauer <- as_date(kh$Entlassung) - as_date(kh$Aufnahme)

# ...oder
kh$Liegedauer <- ceiling(difftime(kh$Entlassung, kh$Aufnahme, units="days"))

# anzeigen
head(kh$Liegedauer)
```

```
Time differences in days
[1] 1 1 8 14 14 22
```

```
str(kh)
```

```
tibble [6,251 x 5] (S3: tbl_df/tbl/data.frame)
 $ Geschlecht: Factor w/ 2 levels "m","w": 1 1 2 2 1 2 2 2 1 2 ...
 $ Alter      : num [1:6251] 76 82 71 71 57 82 61 84 88 74 ...
 $ Aufnahme   : POSIXct[1:6251], format: "2016-06-05 00:00:00" "2016-06-05 19:14:00" ...
 $ Entlassung : POSIXct[1:6251], format: "2016-06-05 23:59:00" "2016-06-06 13:00:00" ...
 $ Liegedauer : 'difftime' num [1:6251] 1 1 8 14 ...
 .. attr(*, "units")= chr "days"
```

💡 d) Über welchen Zeitraum wurden die Daten erhoben?

```
erste <- min(kh$Aufnahme, na.rm=TRUE)
letzte <- max(kh$Entlassung, na.rm=TRUE)

# Zeitspanne in Tagen
as_date(letzte) - as_date(erste)
```

Time difference of 2284 days

```
# Zeitspanne in Wochen
difftime(letzte, erste, units="weeks")
```

Time difference of 326.3253 weeks

```
# Zeitspanne in Jahren
as.numeric(as_date(letzte) - as_date(erste)) / 365
```

```
[1] 6.257534
```

💡 e) Klassieren Sie die Daten der Aufnahme in einer neuen Variable Kalenderjahr.

```
# cut ausprobieren
a <- cut.POSIXt(kh$Aufnahme, breaks="years")
head(a)
```

```
[1] 2016-01-01 2016-01-01 2016-01-01 2016-01-01 2016-01-01 2016-01-01
7 Levels: 2015-01-01 2016-01-01 2017-01-01 2018-01-01 ... 2021-01-01
```

```
# lubridate::year() ist einfacher
a <- year(kh$Aufnahme)
head(a)
```

```
[1] 2016 2016 2016 2016 2016 2016
```

```
# in neue Variable schreiben
kh$Kalenderjahr <- year(kh$Aufnahme)
```

```
# anschauen
glimpse(kh)
```

Rows: 6,251

Columns: 6

```
$ Geschlecht <fct> m, m, w, w, m, w, w, w, m, w, w, w, w, m, w, m, m, w, m, ~
$ Alter      <dbl> 76, 82, 71, 71, 57, 82, 61, 84, 88, 74, 92, 73, 88, 86, 7~
$ Aufnahme  <dtm> 2016-06-05 00:00:00, 2016-06-05 19:14:00, 2016-06-06 13:~
```

```
$ Entlassung    <dtm> 2016-06-05 23:59:00, 2016-06-06 13:00:00, 2016-06-14 13:~  
$ Liegedauer    <drtn> 1 days, 1 days, 8 days, 14 days, 14 days, 22 days, 3 day~  
$ Kalenderjahr  <dbl> 2016, 2016, 2016, 2016, 2016, 2016, 2016, 2016, 2016, 201~
```

💡 f) Klassieren Sie die Daten der Entlassung je mit einer neuen Variable Wochentag und Monat.

```
# Wochentag  
kh$Wochentag <- wday(kh$Entlassung, label=TRUE)  
  
# Monat  
kh$Monat <- month(kh$Entlassung, label=TRUE)  
  
# anschauen  
glimpse(kh)
```

```
Rows: 6,251  
Columns: 8  
$ Geschlecht    <fct> m, m, w, w, m, w, w, w, m, w, w, w, w, m, w, m, m, w, m, ~  
$ Alter         <dbl> 76, 82, 71, 71, 57, 82, 61, 84, 88, 74, 92, 73, 88, 86, 7~  
$ Aufnahme      <dtm> 2016-06-05 00:00:00, 2016-06-05 19:14:00, 2016-06-06 13:~  
$ Entlassung    <dtm> 2016-06-05 23:59:00, 2016-06-06 13:00:00, 2016-06-14 13:~  
$ Liegedauer    <drtn> 1 days, 1 days, 8 days, 14 days, 14 days, 22 days, 3 day~  
$ Kalenderjahr  <dbl> 2016, 2016, 2016, 2016, 2016, 2016, 2016, 2016, 2016, 201~  
$ Wochentag     <ord> So, Mo, Di, Di, Mo, Di, Do, Mi, Mi, Mi, Fr, Fr, Mo, Do, D~  
$ Monat        <ord> Jun, Jun, Jun, Jul, Jun, Jun, Jun, Jun, Aug, Jul, Jun, Ju~
```

5.2. Lösungen zu den Datensatzauswertungen

5.2.1. Lösung zur Aufgabe 2.2.1 Aufnahme und Entlassung

💡 a) Laden Sie den Datensatz Krankenhaus.RData in Ihre R-Session, korrigieren Sie den Tippfehler der Variable ALter, reparieren Sie die Datenklassen der Variablen und entfernen Sie alle Einträge mit ungültigen Zeitstempeln.

```
# Lese Daten ein  
load("https://www.produnis.de/R/data/Krankenhaus.RData")
```

```

library(dplyr)
library(lubridate)
# repariere Typo und Datenklassen und
# entferne NAs
kh <- St.Gott.Hospital %>%
  select(Geschlecht, Alter = ALter, Aufnahme, Entlassung) %>%
  mutate(Geschlecht = factor(Geschlecht),
         Aufnahme = ymd_hm(Aufnahme, tz="CET"),
         Entlassung = ymd_hm(Entlassung, tz="CET")
         ) %>%
  drop_na(Aufnahme, Entlassung)

# anzeigen
glimpse(kh)

```

```

Rows: 6,251
Columns: 4
$ Geschlecht <fct> m, m, w, w, m, w, w, w, m, w, w, w, w, m, w, m, m, w, m, m,~
$ Alter      <dbl> 76, 82, 71, 71, 57, 82, 61, 84, 88, 74, 92, 73, 88, 86, 76,~
$ Aufnahme   <dtm> 2016-06-05 00:00:00, 2016-06-05 19:14:00, 2016-06-06 13:39~
$ Entlassung <dtm> 2016-06-05 23:59:00, 2016-06-06 13:00:00, 2016-06-14 13:30~

```

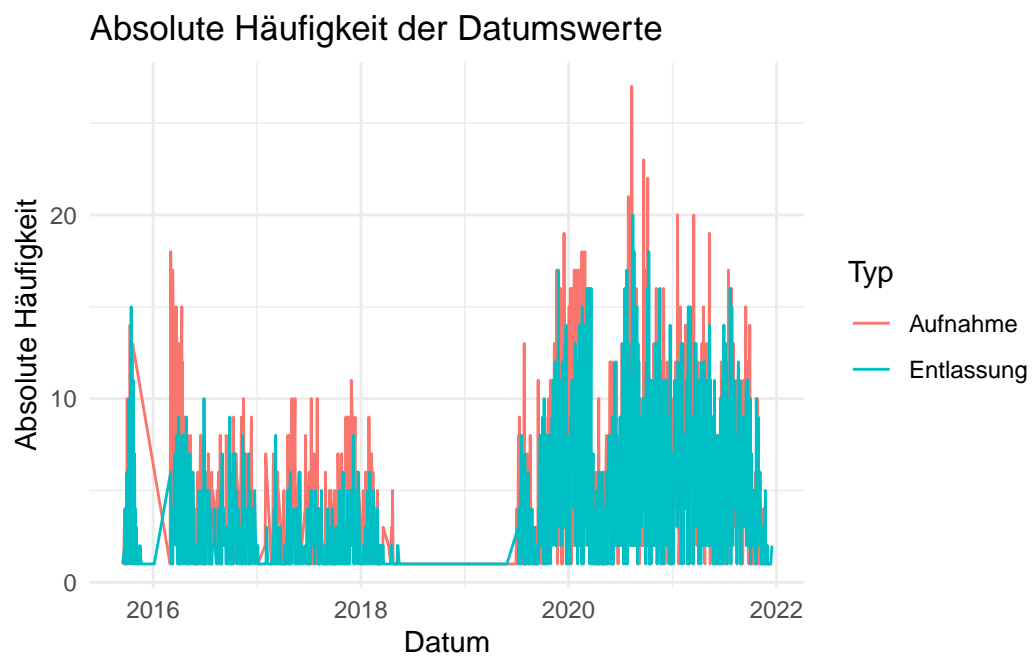
💡 b) Plotten Sie die absoluten Häufigkeiten der Aufnahmen und Entlassungen pro Kalendertag. Was fällt Ihnen auf?

```
library(ggplot2)
# Hilfsdatenframe mit Anzahl Aufnahmen pro Tag
Aufnahmen <- kh %>%
  group_by(as_date(Aufnahme)) %>%
  summarise(freq = n()) %>%
  # Spalten umbenennen
  select(Datum = `as_date(Aufnahme)`, freq) %>%
  # Variable "Typ" hinzufügen
  mutate(Typ="Aufnahme")

# Hilfsdatenframe mit Anzahl Entlassungen pro Tag
Entlassungen <- kh %>%
  group_by(as_date(Entlassung)) %>%
  summarise(freq = n()) %>%
  select(Datum = `as_date(Entlassung)`, freq) %>%
  mutate(Typ="Entlassung")

# Zusammenführen
df <- rbind(Aufnahmen, Entlassungen)

# Plotten
ggplot(df, aes(x=Datum, y=freq)) +
  geom_line(aes(color=Typ)) +
  labs(title = "Absolute Häufigkeit der Datumswerte",
       x = "Datum",
       y = "Absolute Häufigkeit") +
  theme_minimal()
```



Es fällt auf, dass für das Jahr 2019 keine Daten zur Verfügung stehen.

💡 c) Plotten Sie die durchschnittlichen (arithmetisches Mittel) absoluten Häufigkeiten an täglichen Aufnahmen und Entlassungen pro Wochentag. Was fällt Ihnen auf?

```
# nochmal Hilfsdatenframe mit Anzahl Aufnahmen pro Tag
Aufnahmen <- kh %>%
  group_by(as_date(Aufnahme)) %>%
  summarise(freq = n()) %>%
  # Spalten umbenennen
  select(Datum = `as_date(Aufnahme)`, freq) %>%
  # Variable "Typ" hinzufügen
  mutate(Typ = "Aufnahme",
         # Wochentag hinzufügen
         Tag = wday(Datum, label=TRUE))

# Hilfsdatenframe mit Anzahl Entlassungen pro Tag
Entlassungen <- kh %>%
  group_by(as_date(Entlassung)) %>%
  summarise(freq = n()) %>%
  select(Datum = `as_date(Entlassung)`, freq) %>%
  mutate(Typ = "Entlassung",
         # Wochentag hinzufügen
         Tag = wday(Datum, label=TRUE))

# zusammenführen
Wochentage <- rbind(Aufnahmen, Entlassungen)

# absolute Häufigkeiten anzeigen
table(Wochentage$Typ, Wochentage$Tag)
```

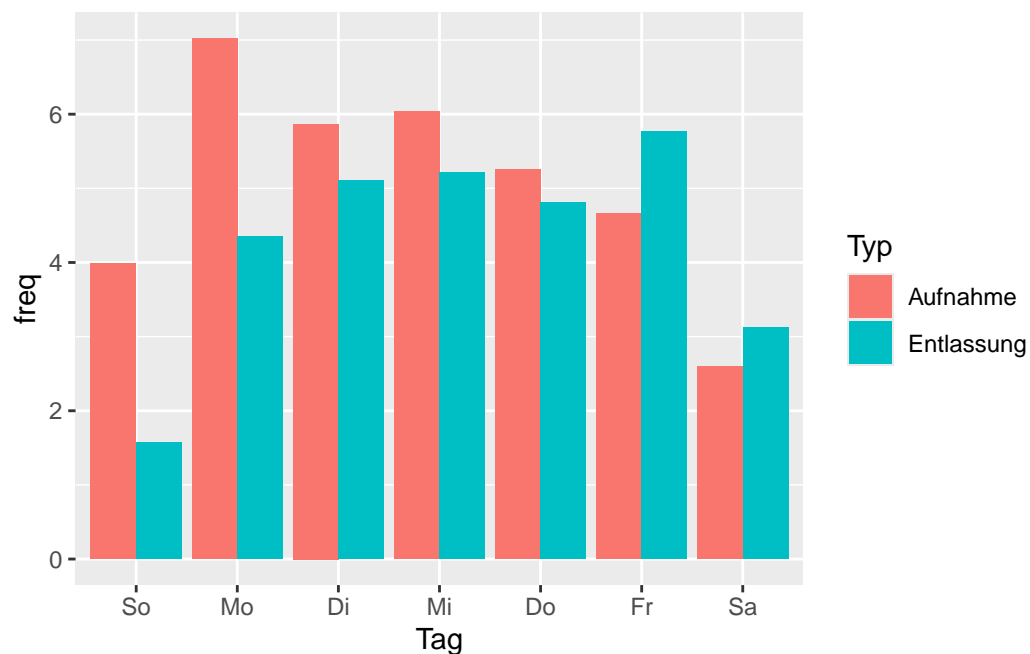
	So	Mo	Di	Mi	Do	Fr	Sa
Aufnahme	165	195	205	192	169	137	130
Entlassung	107	213	219	220	222	222	174

```
# durchschnittliche Häufigkeiten
Wochentage %>%
  group_by(Typ, Tag) %>%
  summarise(Mean = mean(freq))
```

```
# A tibble: 14 x 3
# Groups:   Typ [2]
  Typ      Tag      Mean
  <chr>    <ord> <dbl>
1 Aufnahme So      3.98
2 Aufnahme Mo      7.02
3 Aufnahme Di      5.87
4 Aufnahme Mi      6.04
```


5	Aufnahme	Do	5.25
6	Aufnahme	Fr	4.66
7	Aufnahme	Sa	2.6
8	Entlassung	So	1.57
9	Entlassung	Mo	4.35
10	Entlassung	Di	5.10
11	Entlassung	Mi	5.21
12	Entlassung	Do	4.81
13	Entlassung	Fr	5.77
14	Entlassung	Sa	3.13

```
# durchschnittliche (arith.) Häufigkeiten
ggplot(Wochentage, aes(x=Tag, y=freq, fill=Typ)) +
  stat_summary(fun=mean, geom="bar", position="dodge")
```



An Sonn- und Montag gibt es deutlich mehr Aufnahmen als Entlassungen.

💡 d) Plotten Sie die durchschnittlichen absoluten Häufigkeiten an täglichen Aufnahmen und Entlassungen pro Monat sowie die absoluten Häufigkeiten pro Tagesstunde.

```
# nochmal Hilfsdatenframe mit Anzahl Aufnahmen pro Monat
Aufnahmen <- kh %>%
  group_by(as_date(Aufnahme)) %>%
  summarise(freq = n()) %>%
  # Spalten umbenennen
  select(Datum = `as_date(Aufnahme)`, freq) %>%
  # Variable "Typ" hinzufügen
  mutate(Typ = "Aufnahme",
         # Monat hinzufügen
         Monat= month(Datum, label=TRUE))

# Hilfsdatenframe mit Anzahl Entlassungen pro Tag
Entlassungen <- kh %>%
  group_by(as_date(Entlassung)) %>%
  summarise(freq = n()) %>%
  select(Datum = `as_date(Entlassung)`, freq) %>%
  mutate(Typ = "Entlassung",
         # Monate hinzufügen
         Monat= month(Datum, label=TRUE))

# zusammenführen
Monate <- rbind(Aufnahmen, Entlassungen)

# absolute Häufigkeiten anzeigen
table(Monate$Typ, Monate$Monat)
```

	Jan	Feb	Mär	Apr	Mai	Jun	Jul	Aug	Sep	Okt	Nov	Dez
Aufnahme	82	73	86	95	93	87	121	108	124	141	103	80
Entlassung	77	84	121	108	108	97	125	128	132	163	128	106

```
# durchschnittliche Häufigkeiten
Monate %>%
  group_by(Typ, Monat) %>%
  summarise(Median = median(freq))
```

```
# A tibble: 24 x 3
# Groups:   Typ [2]
  Typ      Monat Median
<chr>    <ord> <dbl>
1 Aufnahme Jan      5
2 Aufnahme Feb      5
3 Aufnahme Mär      6
4 Aufnahme Apr      3
```

```

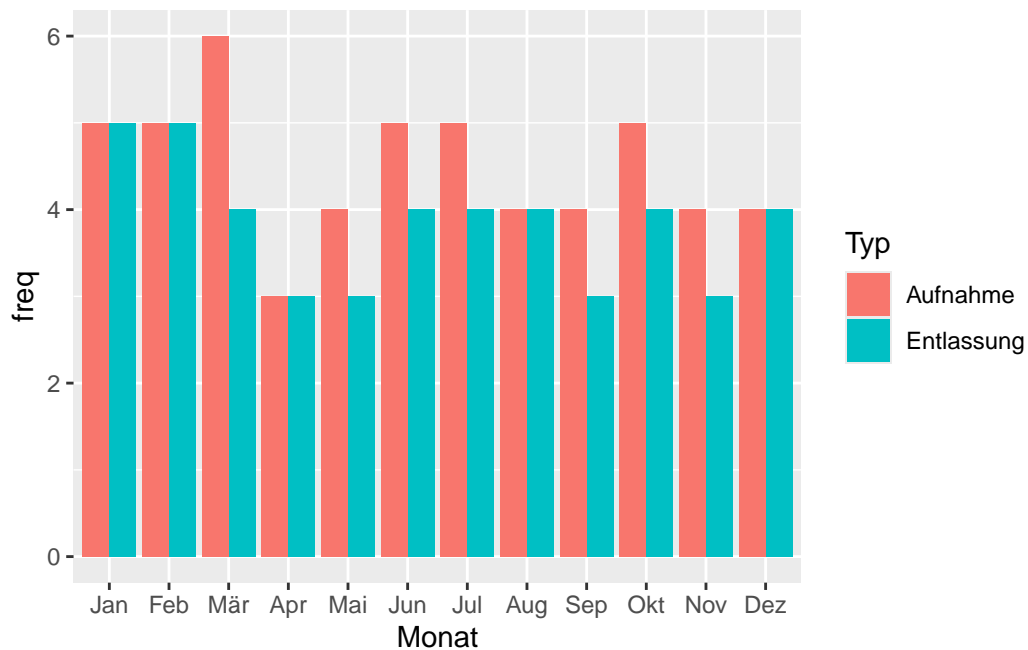
5 Aufnahme Mai          4
6 Aufnahme Jun          5
7 Aufnahme Jul           5
8 Aufnahme Aug           4
9 Aufnahme Sep           4
10 Aufnahme Okt          5
# i 14 more rows

```

```

# durchschnittliche (Median) Häufigkeiten
ggplot(Monate, aes(x=Monat, y=freq, fill=Typ)) +
  stat_summary(fun=median, geom="bar", position="dodge")

```



Wiederholen wir nun den Vorgang für die Häufigkeiten pro Tagesstunde.

```

# nochmal Hilfsdatenframe mit Anzahl Aufnahmen pro Tagesstunde
kh$Aufnahmestunde <- hour(kh$Aufnahme)
kh$Entlassungstunde <- hour(kh$Entlassung)

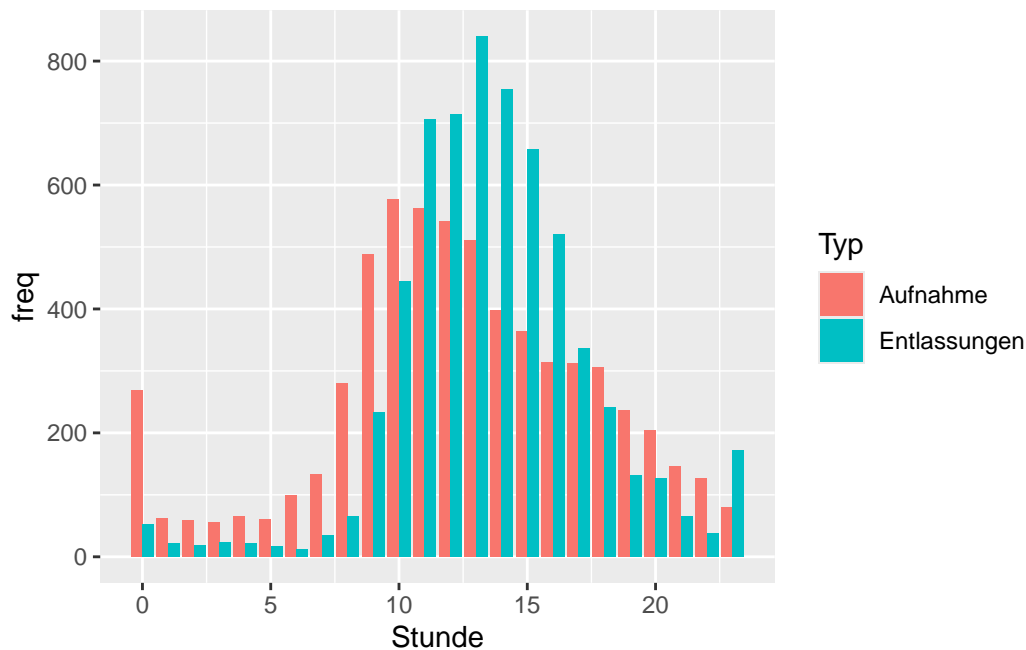
Aufnahmen <- kh %>%
  group_by(Aufnahmestunde) %>%
  summarise(freq = n()) %>%
  # Variable "Typ" hinzufügen
  mutate(Typ = "Aufnahme") %>%
  select(Stunde = Aufnahmestunde, freq, Typ)

# Hilfsdatenframe mit Anzahl Entlassungen pro Tagesstunde
Entlassungen <- kh %>%
  group_by(Entlassungstunde) %>%
  summarise(freq = n()) %>%
  # Variable "Typ" hinzufügen
  mutate(Typ = "Entlassungen") %>%
  select(Stunde = Entlassungstunde, freq, Typ)

# zusammenführen
Stunden <- rbind(Aufnahmen, Entlassungen)

# absolute Häufigkeiten pro Tagesstunde
ggplot(Stunden, aes(x=Stunde, y=freq, fill=Typ)) +
  geom_col(position="dodge")

```



💡 e) Erstellen Sie ein Poissonregressionsmodell für die Anzahl der täglichen Aufnahmen erklärt durch den Wochentag. Ist das Modell überdispersioniert? Wieviele Aufnahmen sind an einem Dienstag und an einem Sonntag zu erwarten?

```
# nur Aufnahmen
dfA <- subset(Wochentage, Typ=="Aufnahme")

# "Tag" für Poisson vorbereiten
# ordered entfernen
dfA$Tag <- factor(dfA$Tag, ordered=FALSE)
# Montag als Basiswert
dfA$Tag <- relevel(dfA$Tag, "Mo")

# Poisson-Modell erstellen
fit <- glm(freq ~ Tag, data=dfA, family = poisson)

# Zusammenfassung des Modells
summary(fit)
```

Call:

```
glm(formula = freq ~ Tag, family = poisson, data = dfA)
```

Coefficients:

	Estimate	Std. Error	z value	Pr(> z)
(Intercept)	1.94884	0.02703	72.107	< 2e-16 ***
TagSo	-0.56710	0.04746	-11.949	< 2e-16 ***
TagDi	-0.17927	0.03952	-4.536	5.72e-06 ***
TagMi	-0.15102	0.03992	-3.783	0.000155 ***
TagDo	-0.29089	0.04310	-6.749	1.49e-11 ***
TagFr	-0.41048	0.04794	-8.563	< 2e-16 ***
TagSa	-0.99332	0.06074	-16.354	< 2e-16 ***

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

(Dispersion parameter for poisson family taken to be 1)

Null deviance: 2994.4 on 1192 degrees of freedom
Residual deviance: 2574.6 on 1186 degrees of freedom
AIC: 6501.8

Number of Fisher Scoring iterations: 5

```
# alternative Zusammenfassung
sjPlot::tab_model(fit)
```

freq

Predictors	Incidence Rate Ratios	CI	p
(Intercept)	7.02	6.66 – 7.40	<0.001
Tag [So]	0.57	0.52 – 0.62	<0.001
Tag [Di]	0.84	0.77 – 0.90	<0.001
Tag [Mi]	0.86	0.80 – 0.93	<0.001
Tag [Do]	0.75	0.69 – 0.81	<0.001
Tag [Fr]	0.66	0.60 – 0.73	<0.001
Tag [Sa]	0.37	0.33 – 0.42	<0.001
Observations	1193		
R ² Nagelkerke	0.323		

Testen wir, ob das Modell überdispersioniert ist.

```
AER::dispersiontest(fit, trafo=1)
```

Overdispersion test

```
data: fit
z = 10.82, p-value < 2.2e-16
alternative hypothesis: true alpha is greater than 0
sample estimates:
  alpha
1.273968
```

Der Test ist signifikant, d.h. das Modell ist überdispersioniert. Wir müssen das Modell daher anpassen:

```
fit <- glm(freq ~ Tag, data=dfA, family = quasipoisson)
summary(fit)
```

Call:

```
glm(formula = freq ~ Tag, family = quasipoisson, data = dfA)
```

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	1.94884	0.04088	47.677	< 2e-16 ***
TagSo	-0.56710	0.07178	-7.900	6.30e-15 ***
TagDi	-0.17927	0.05977	-2.999	0.00276 **
TagMi	-0.15102	0.06037	-2.502	0.01250 *
TagDo	-0.29089	0.06519	-4.462	8.88e-06 ***
TagFr	-0.41048	0.07250	-5.662	1.88e-08 ***
TagSa	-0.99332	0.09186	-10.813	< 2e-16 ***

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

(Dispersion parameter for quasipoisson family taken to be 2.28739)

```
Null deviance: 2994.4 on 1192 degrees of freedom
Residual deviance: 2574.6 on 1186 degrees of freedom
AIC: NA
```

Number of Fisher Scoring iterations: 5

Mit dem neuen Modell können nun die Vorhersagen erfolgen.

```
# Vorhersage Dienstag
predict(fit, list(Tag="Di"), type = "response")
```

```
1
5.868293
```

```
# Vorhersage Sonntag
predict(fit, list(Tag="So"), type = "response")
```

```
1
3.981818
```

💡 f) Fügen Sie den Monat als weiteren Prädiktor hinzu. Wird das Modell dadurch besser? Wieviele Aufnahmen sind an einem Donnerstag im Mai zu erwarten, und wieviele im September?

```
dfA$Monat <- month(dfA$Datum, label=TRUE)
dfA$Monat <- factor(dfA$Monat, ordered=FALSE)
dfA$Monat <- relevel(dfA$Monat, "Jan")

fit <- glm(freq ~ Tag + Monat, data=dfA, family="poisson")
summary(fit)
```

Call:

```
glm(formula = freq ~ Tag + Monat, family = "poisson", data = dfA)
```

Coefficients:

	Estimate	Std. Error	z value	Pr(> z)
(Intercept)	2.13203	0.05032	42.368	< 2e-16 ***
TagSo	-0.56984	0.04749	-11.998	< 2e-16 ***
TagDi	-0.17175	0.03955	-4.342	1.41e-05 ***
TagMi	-0.14825	0.03995	-3.711	0.000207 ***
TagDo	-0.28742	0.04316	-6.660	2.74e-11 ***
TagFr	-0.41400	0.04798	-8.629	< 2e-16 ***
TagSa	-0.98855	0.06079	-16.263	< 2e-16 ***
MonatFeb	0.02741	0.06389	0.429	0.667963
MonatMär	-0.01210	0.06150	-0.197	0.844042
MonatApr	-0.29136	0.06485	-4.492	7.04e-06 ***
MonatMai	-0.32501	0.06576	-4.942	7.72e-07 ***
MonatJun	-0.26052	0.06548	-3.979	6.93e-05 ***
MonatJul	-0.14788	0.05923	-2.497	0.012536 *
MonatAug	-0.19857	0.06124	-3.243	0.001184 **
MonatSep	-0.29288	0.06052	-4.839	1.30e-06 ***
MonatOkt	-0.21975	0.05802	-3.788	0.000152 ***
MonatNov	-0.18356	0.06151	-2.984	0.002842 **
MonatDez	-0.29117	0.06761	-4.307	1.66e-05 ***

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

(Dispersion parameter for poisson family taken to be 1)


```
Null deviance: 2994.4 on 1192 degrees of freedom
Residual deviance: 2493.2 on 1175 degrees of freedom
AIC: 6442.3
```

Number of Fisher Scoring iterations: 5

Das Modell hat einen größeren AIC-Wert als das alte.
Testen wir, ob das Modell überdispersioniert ist.

```
AER::dispersiontest(fit, trafo=1)
```

Overdispersion test

```
data: fit
z = 10.534, p-value < 2.2e-16
alternative hypothesis: true alpha is greater than 0
sample estimates:
alpha
1.185659
```

Der Test ist signifikant, d.h. das Modell **ist** überdispersioniert. Wir müssen das Modell anpassen.

```
fit <- glm(freq ~ Tag + Monat, data=dfA, family = quasipoisson)
summary(fit)
```

Call:

```
glm(formula = freq ~ Tag + Monat, family = quasipoisson, data = dfA)
```

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	2.13203	0.07498	28.436	< 2e-16	***
TagSo	-0.56984	0.07076	-8.053	1.97e-15	***
TagDi	-0.17175	0.05893	-2.915	0.003630	**
TagMi	-0.14825	0.05952	-2.491	0.012892	*
TagDo	-0.28742	0.06430	-4.470	8.59e-06	***
TagFr	-0.41400	0.07148	-5.791	8.95e-09	***
TagSa	-0.98855	0.09057	-10.915	< 2e-16	***
MonatFeb	0.02741	0.09519	0.288	0.773478	
MonatMär	-0.01210	0.09163	-0.132	0.894978	
MonatApr	-0.29136	0.09663	-3.015	0.002623	**
MonatMai	-0.32501	0.09798	-3.317	0.000937	***
MonatJun	-0.26052	0.09756	-2.670	0.007681	**
MonatJul	-0.14788	0.08825	-1.676	0.094064	.
MonatAug	-0.19857	0.09124	-2.176	0.029728	*
MonatSep	-0.29288	0.09017	-3.248	0.001195	**
MonatOkt	-0.21975	0.08645	-2.542	0.011147	*
MonatNov	-0.18356	0.09164	-2.003	0.045410	*

```

MonatDez      -0.29117      0.10073    -2.891 0.003916 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for quasipoisson family taken to be 2.219922)

Null deviance: 2994.4  on 1192  degrees of freedom
Residual deviance: 2493.2  on 1175  degrees of freedom
AIC: NA

Number of Fisher Scoring iterations: 5

```

Mit dem neuen Modell können wir nun die Vorhersagen treffen.

```

# Vorhersagen
predict(fit, list(Tag="Do", Monat="Mai"), type = "response")

```

```

1
4.570387

```

```

predict(fit, list(Tag="Do", Monat="Sep"), type = "response")

```

```

1
4.719636

```

💡 g) Wie groß ist die Wahrscheinlichkeit, dass an einem Mittwoch im Mai 10 Patienten aufgenommen werden?

```

# Schätzen der mittleren Häufigkeit
mu <- predict(fit, list(Tag="Mi", Monat="Mai"), type = "response")

# Wahrscheinlichkeit für 10 Aufnahmen berechnen
dpois(10, lambda = mu)

```

```

[1] 0.02306207

```

Die Wahrscheinlichkeit liegt bei 2,3%.

💡 h) Wie groß ist die Wahrscheinlichkeit, dass an einem Mittwoch im Mai zwischen 4 und 7 Patienten aufgenommen werden?

```
# Schätzen der mittleren Häufigkeit
mu <- predict(fit, list(Tag="Mi", Monat="Mai"), type = "response")

# Wahrscheinlichkeit für 4 bis 7 Aufnahmen berechnen
# entweder
ppois(7, lambda=mu) - ppois(3, lambda=mu)
```

```
[1] 0.607611
```

```
# oder
sum(dpois(4:7, lambda=mu))
```

```
[1] 0.607611
```

Die Wahrscheinlichkeit liegt bei 60,76%.

💡 i) Wie groß ist die Wahrscheinlichkeit, dass an einem Montag im Januar maximal 2 Patienten aufgenommen werden?

```
# Schätzen der mittleren Häufigkeit
mu <- predict(fit, list(Tag="Mo", Monat="Jan"), type = "response")

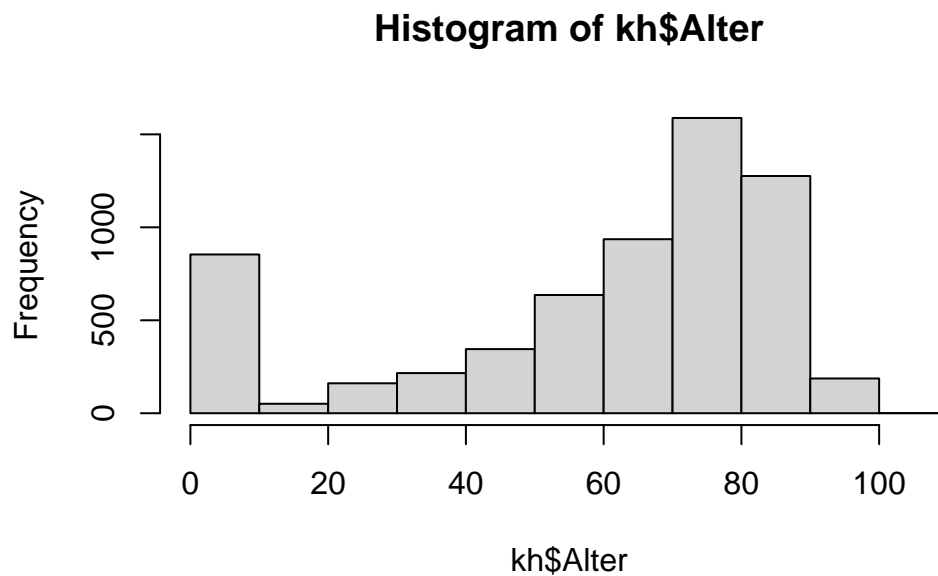
# Wahrscheinlichkeit für maximal 2 Aufnahmen berechnen
ppois(2, lambda = mu)
```

```
[1] 0.009796846
```

Die Wahrscheinlichkeit liegt bei 0,98%.

💡 j) Erzeugen Sie ein Histogramm des Alters der Probanden. Was fällt Ihnen auf? Korrigieren Sie wenn nötig die Daten. Ist das Alter der Probanden normalverteilt?

```
# Histogramm mit Rbase
hist(kh$Alter)
```



```
# Wahrscheinlichkeit für maximal 2 Aufnahmen berechnen  
ppois(2, lambda = mu)
```

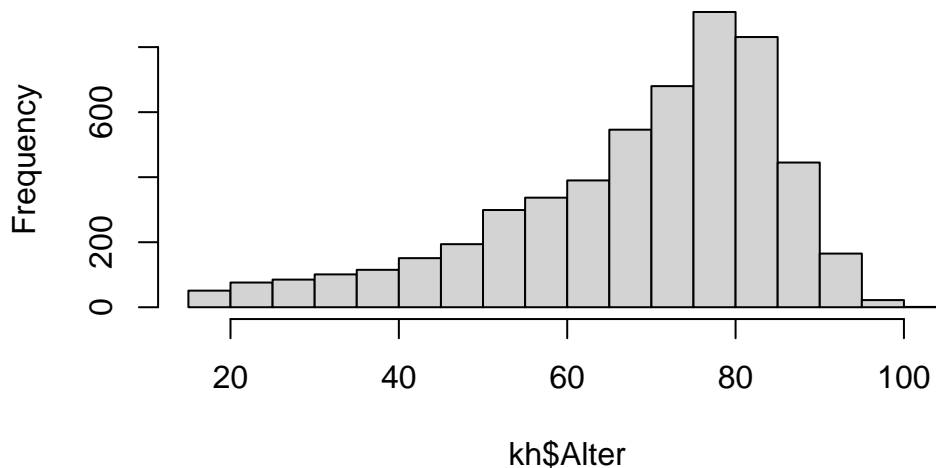
```
[1] 0.009796846
```

Es fällt auf, dass es viele Probanden mit Alter=0 gibt. Diese sollten in NA umgewandelt werden.

```
kh$Alter[kh$Alter==0] <- NA
```

```
# Histogramm wiederholen  
hist(kh$Alter)
```

Histogram of kh\$Alter



```
# Teste, ob Alter normalverteilt ist
ks.test(kh$Alter, "pnorm")
```

Warning in ks.test.default(kh\$Alter, "pnorm"): ties should not be present for the one-sample Kolmogorov-Smirnov test

Asymptotic one-sample Kolmogorov-Smirnov test

```
data: kh$Alter
D = 1, p-value < 2.2e-16
alternative hypothesis: two-sided
```

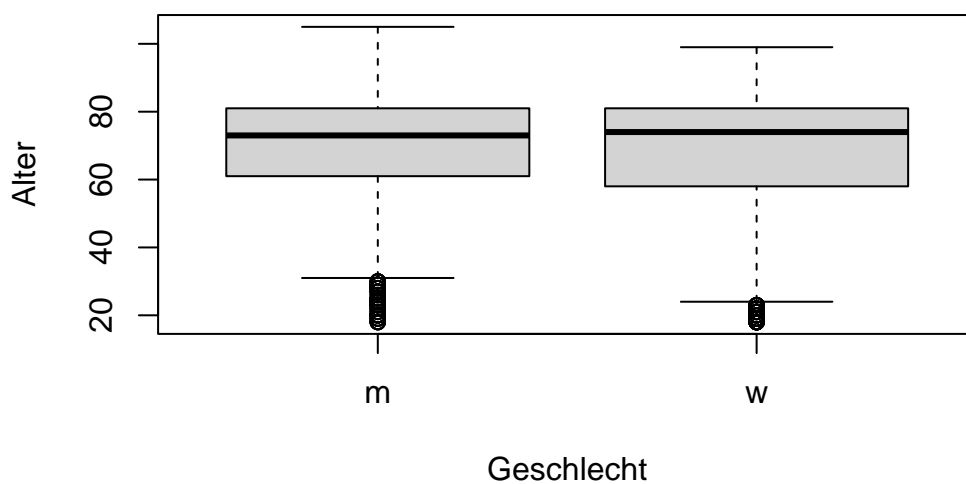
Der Test ist signifikant, das heisst, es liegt **keine** Normalverteilung vor.

💡 k) Stellen Sie das Alter der Männern und Frauen tabellarisch und graphisch dar. Unterscheidet sich das Alter der Probanden zwischen Männern und Frauen?

```
# Tabellarisch
kh %>%
  group_by(Geschlecht) %>%
  drop_na(Alter) %>%
  summarise(Min = min(Alter),
            Q1 = quantile(Alter, probs=0.25, type=6),
            Median = median(Alter),
            Mittel = mean(Alter),
            Q3 = quantile(Alter, probs=0.75, type=6),
            Max = max(Alter))
```

```
# A tibble: 2 x 7
  Geschlecht   Min    Q1 Median Mittel   Q3   Max
  <fct>       <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1 m           18    61    73   69.4    81   105
2 w           18    58    74   68.5    81    99
```

```
# graphisch
boxplot(Alter ~ Geschlecht, data=kh)
```



Männer und Frauen unterscheiden sich nicht hinsichtlich des Alters.

💡 1) Ist der Unterschied signifikant?

```
# subsets vorbereiten
m <- subset(kh, Geschlecht=="m")
w <- subset(kh, Geschlecht=="w")

# keine Normalverteilung = kein t.Test
wilcox.test(m$Alter, w$Alter)
```

Wilcoxon rank sum test with continuity correction

```
data: m$Alter and w$Alter
W = 3621860, p-value = 0.74
alternative hypothesis: true location shift is not equal to 0
```

Der Test ist nicht signifikant, es liegt kein Unterschied vor.

💡 m) Ab welchem Alter sind 10% der Männer älter als dieser Wert?

```
# nur Männer
m <- subset(kh, Geschlecht=="m")
# beim 90. Perzentil liegen 10% der Werte darüber
quantile(m$Alter, 0.9, na.rm=TRUE, type=6)
```

90%
86

Es sind 10% der Männer älter als 86 Jahre.

💡 n) Ab welchem Alter sind 80% der Frauen jünger als dieser Wert?

```
# nur Frauen
w <- subset(kh, Geschlecht=="w")
# beim 90. Perzentil liegen 10% der Werte darüber
quantile(w$Alter, 0.8, na.rm=TRUE, type=6)
```

80%
83

Es sind 80% der Frauen jünger als 83 Jahre.

💡 o) Wie groß ist die mittlere Liegedauer in Tagen? Stellen Sie die Liegedauer mittels Kennwerten sowie graphisch dar. Was fällt Ihnen auf?

```
# Liegedauer berechnen
kh$Liegedauer <- as_date(kh$Entlassung) - as_date(kh$Aufnahme)
# mittlere Liegedauer, Median
mean(kh$Liegedauer)
```

Time difference of 8.582627 days

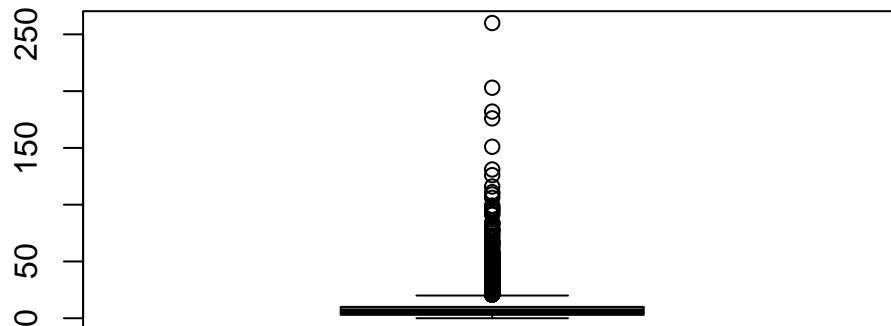
```
# mittlere Liegedauer, Median
median(kh$Liegedauer)
```

Time difference of 6 days

```
# Tabellarische Darstellung
summary(as.numeric(kh$Liegedauer))
```

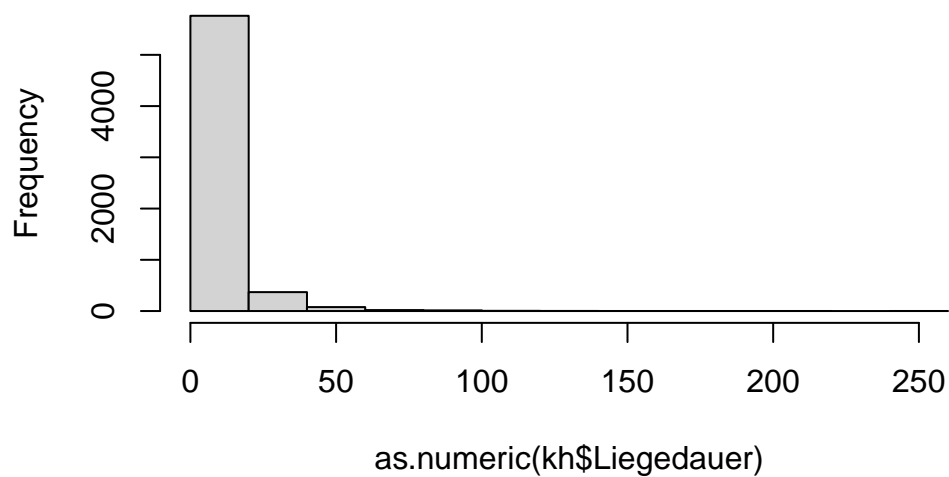
Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
0.000	3.000	6.000	8.583	10.000	260.000

```
# graphische Darstellung
boxplot(kh$Liegedauer)
```



```
hist(as.numeric(kh$Liegedauer))
```

Histogram of as.numeric(kh\$Liegedauer)



Es fällt auf, dass sehr viele Ausreißer enthalten sind.

💡 p) Wie viel Prozent der Patienten haben eine Liegedauer von mehr als 7 Tagen?

```
sum(kh$Liegedauer > 7) / length(kh$Liegedauer)
```

```
[1] 0.3722604
```


Im Datensatz haben 37,23 % der Patienten eine Liegedauer von mehr als 7 Tagen.

💡 q) Unterscheiden sich Männer und Frauen hinsichtlich der Liegedauer? Stellen Sie den Unterschied ebenfalls tabellarisch und graphisch dar.

```
# Tabellarische Darstellung
```

```
kh %>%
```

```
  group_by(Geschlecht) %>%
```

```
  summarise(Min = min(Liegedauer),
```

```
            Q1 = quantile(Liegedauer, probs=0.25, type=6),
```

```
            Median = median(Liegedauer),
```

```
            Mittel = mean(Liegedauer),
```

```
            Q3 = quantile(Liegedauer, probs=0.75, type=6),
```

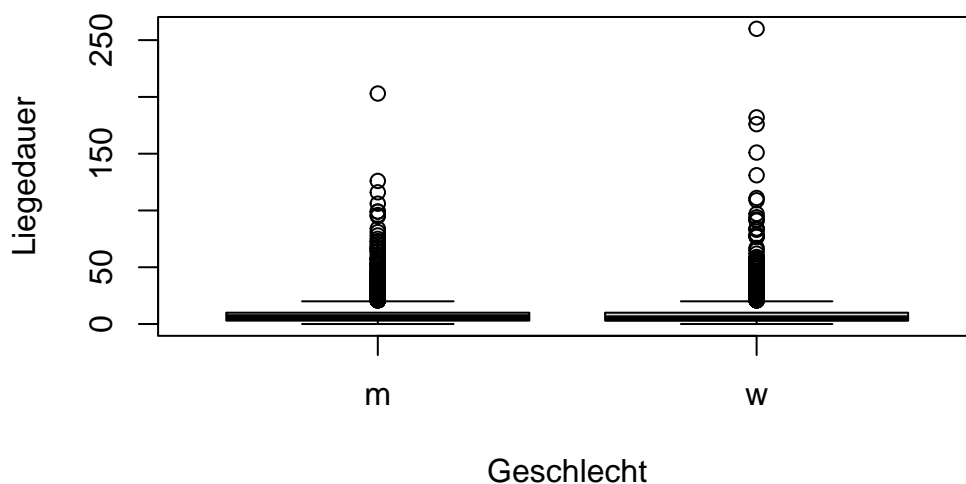
```
            Max = max(Liegedauer))
```

```
# A tibble: 2 x 7
```

	Geschlecht	Min	Q1	Median	Mittel	Q3	Max
	<fct>	<drtn>	<drtn>	<drtn>	<drtn>	<drtn>	<drtn>
1	m	0 days	3 days	6 days	8.684497 days	10 days	203 days
2	w	0 days	3 days	5 days	8.480984 days	10 days	260 days

```
# graphische Darstellung
```

```
boxplot(Liegedauer ~ Geschlecht, data=kh)
```



Es ist kein Unterschied erkennbar.

💡 r) Ist der Unterschied der Liegedauer zwischen Männern und Frauen signifikant?

```
# Teste auf Normalverteilung
ks.test(kh$Liegedauer, "pnorm")
```

Asymptotic one-sample Kolmogorov-Smirnov test

```
data: kh$Liegedauer
D = 0.84543 days, p-value < 2.2e-16
alternative hypothesis: two-sided
```

Der Test ist signifikant, d.h. es liegt **keine** Normalverteilung vor. Als Signifikanztest ist daher der Mann-Whitney-U-Test durchzuführen

```
# Vorbereitung
kh$Liegedauer <- as.numeric(kh$Liegedauer)
m <- subset(kh, Geschlecht=="m")
w <- subset(kh, Geschlecht=="w")

# Mann-Whitney-U-Test
wilcox.test(w$Liegedauer, m$Liegedauer)
```

Wilcoxon rank sum test with continuity correction

```
data: w$Liegedauer and m$Liegedauer
W = 4624670, p-value = 0.0002638
alternative hypothesis: true location shift is not equal to 0
```

Das Ergebnis ist signifikant. Es scheint doch einen Unterschied zwischen Männern und Frauen zu geben.

6. Lösungswege zu den Aufgaben für fortgeschrittene Anwender:innen

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

6.1. Lösungen zu Objekten in R

6.1.1. Lösung zur Aufgabe 3.1.1 Hogwarts-Kurse

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

```
library(tibble)
```

6.2. Lösungen zu den Datensatzauswertungen

6.2.1. Lösung zur Aufgabe 3.2.1 Hogwarts-Kurse

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

```
library(tibble)
```

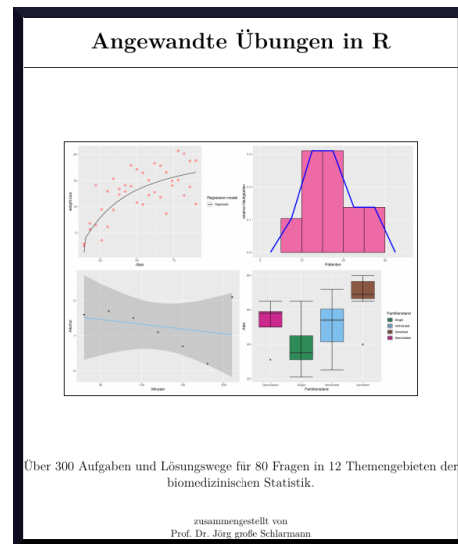
Literaturverzeichnis

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

Credits



(a) große Schlarmann (2024b)



(a) große Schlarmann (2024a)

Prof. Dr. Jörg große Schlarmann

Hochschule Niederrhein, Krefeld

joerg.grosseschlarmann@hs-niederrhein.de

<https://www.produnis.de/R>

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