Abgabe Testat 1

Gruppe Nr. 1

Gruppenmitglieder

Name	**Matrikelnummer**
Nikola Munder	2212719
Tabea Wollenweber	3034261
Youssef Elwy	3037523

Veranstaltung:	DSR
Dozent:	Prof. Gumbel
Abgabedatum:	01.11.2026

Eigenes Histogramm

Eigenes Histogramm

Es soll ein eigenes Histogramm erzeugt werden. Der Dateiname für das Skript ist `myhistogram.R`.

Funktion `myhistogram`

Programmieren Sie in R die Funktion `myhistogram`, die als Parameter `x` einen Vektor aus Zahlen erhält. Die Zahlen werden in `n` Intervalle einsortiert, und es wird gezählt, wie oft eine Zahl in einem Intervall vorkommt. Der Rückgabewert ist eine Liste mit den Einträgen `borders`, die die $n + 1$ Intervallgrenzen enthalten und `counts`, der die Anzahlen enthält.

- Die n Intervalle sollen gleich groß sein (Δb), d.h. für die Intervallgrenzen $b_1, b_2, \ldots, b_{n+1}$ gilt $\frac{b_{i+1}-b_i}{n} = \Delta b$ für $i = 1, 2, \ldots n$.
- Die äußeren Grenzen b_1 und b_n sollen als optionale Parametern `min` und `max` an die Funktion übergeben werden. Werte aus `x`, die zu keinem Intervall gehören, sollen ignoriert werden. Es wird aber eine Warnung ausgegeben, die sagt, welche Zahlen außerhalb des Bereichs liegen.
- Eine Zahl z gehört zum i-ten Intervall, falls $b_i \leq z < b_{i+1}$ gilt.

Bis auf `x` sollen alle Parameter optional sein. Überlegen Sie sinnvolle Default-Werte.

Es ist natürlich **nicht** erlaubt, in der eigenen Funktion andere Funktionen zu nutzen, die ein Histogramm erzeugen.

Hier ein Beispiel:

```r
# Bitte mit echter Lösung in myhistogram.R ersetzen:
myhistogram = function(x, n, min, max) {
  if (missing(min)) min <- base::min(x)

  # sonst fällt max(x) wegen b_i <= z < b_{i+1} raus
  if (missing(max)) max <- base::max(x) + 10^-12

  if (missing(n)) n <- 10

  if (min >= max) {
    stop("min muss kleiner als max sein.")
  }

  # Äquidistante Aufteilung der Eingabe in n Intervalle mit n + 1 Grenzen
  borders <- seq(min, max, length.out = n + 1)
  counts <- rep(0, n)

  # Werte außerhalb des gewählten Intervalls ermitteln
  outside <- x[x < min | x >= max]

  if (length(outside) > 0) {
    warning(paste("Zahl(en) außerhalb Intervallgrenzen:",
                  paste(outside, collapse = ", ")))
  }
```

```
  for(value in x) {
    for (i in 1:n){
      # Linke Grenze inklusive, rechte exklusive
      # Deswegen wurde oben auf max 10^-12 addiert
      if (value >= borders[i] && value < borders[i+1])
        {
        counts[i] <- counts[i] + 1
        break
      }
    }
  }
  # Intervallgrenzen und absolute Häufigkeit zurückgeben
  list(borders=borders, counts = counts)
}
x = seq(-5, 6, by = 1 / 3)

l = myhistogram(x)
print(l$borders)
```

```
##  [1] -5.0 -3.9 -2.8 -1.7 -0.6  0.5  1.6  2.7  3.8  4.9  6.0
```

```
print(l$counts)
```

```
##  [1] 4 3 3 4 3 3 4 3 3 4
```

Beispieldaten

Hier zunächst zwei Beispiele.

Beispiel 1

Es wird eine Warnung ausgeben:

```
## Warning in myhistogram(x, n = 10, min = -5, max = 6): Zahl(en) außerhalb
## Intervallgrenzen: 6
```

```
x = seq(-5, 6, by = 1 / 3)
l = myhistogram(x, n = 10, min = -5, max = 6)
```

```
## Warning in myhistogram(x, n = 10, min = -5, max = 6): Zahl(en) außerhalb
## Intervallgrenzen: 6
```

```
solution = list(
  borders = c(-5.0, -3.9, -2.8, -1.7, -0.6, 0.5, 1.6, 2.7, 3.8, 4.9, 6.0),
  counts = c(4, 3, 3, 4, 3, 3, 4, 3, 3, 3)
)
print(l$borders)
```

```
##  [1] -5.0 -3.9 -2.8 -1.7 -0.6  0.5  1.6  2.7  3.8  4.9  6.0
```

```
print(l$counts)
```

```
##  [1] 4 3 3 4 3 3 4 3 3 3
```

Beispiel 2

```
x = seq(-5, 6, by = 1 / 3)
myhistogram(x, n = 5, min = -10, max = 10)
```

```
## $borders
## [1] -10  -6  -2   2   6  10
##
## $counts
## [1]  0  9 12 12  1
```

```
solution = list(borders = c(-10, -6, -2, 2, 6, 10),
                counts = c(0, 9, 12, 12, 1))
```

Beispiel 3

Ganze Zahlen mit Wiederholungen, Default-Werte für `n`, `min` und `max`

```
x = c(1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 5, 6, 7, 8, 9, 10)
l = myhistogram(x)
print(l$borders)
```

```
##  [1]  1.0  1.9  2.8  3.7  4.6  5.5  6.4  7.3  8.2  9.1 10.0
```

```
print(l$counts)
```

```
##  [1] 1 2 3 4 1 1 1 1 1 1
```

Beispiel 4

Werte unterhalb von `min` und oberhalb von `max` werden ignoriert und in der Warnung aufgelistet:

```
x = c(-3, -1, 0, 0.5, 1, 2.5, 4, 4.99, 5, 7)

# da max explizit gesetzt wird, fallen 5 und 7 aus den gültigen Grenzen raus
l = myhistogram(x, n = 5, min = 0, max = 5)
```

```
## Warning in myhistogram(x, n = 5, min = 0, max = 5): Zahl(en) außerhalb
## Intervallgrenzen: -3, -1, 5, 7
```

```
solution = list(borders = c(0, 1, 2, 3, 4, 5),
                counts = c(2, 1, 1, 0, 2))
print(l$borders)
```

```
## [1] 0 1 2 3 4 5
```

```
print(l$counts)
```

```
## [1] 2 1 1 0 2
```

Beispiel 5

Normalverteilte Zufallszahlen, deren Bereich vollständig innerhalb von `min` und `max` liegt (keine Warnung):

```
set.seed(42)
x = rnorm(100, mean = 0, sd = 1)
l = myhistogram(x, n = 8, min = -4, max = 4)
print(l$borders)
```

```
## [1] -4 -3 -2 -1  0  1  2  3  4
```

```r
print(l$counts)
```

```
## [1]  0  4 11 31 37 15  2  0
```

```r
print(sum(l$counts))  # muss 100 ergeben
```

```
## [1] 100
```

Barplot

Nutzen Sie Ihre Funktion `myhistogram` und erzeugen Sie einen Barplot mit *ggplot*. Die x-Achse zeigt dabei die Mitte des Intervalls und die y-Achse die Anzahl der Elemente in dieser Klasse.

Tipp: Der Parameter `stat` von `geom_bar` ist wichtig.

Vervollständigen Sie den Chunk. Die Kommentare sollen zu Anweisungen umgewandelt werden:

```r
library("ggplot2")
set.seed(1)
x = rnorm(0, 1, n = 1000)
h = myhistogram(x)
# letzte und erste Grenze entfernen sodass nebeneinanderliegende Grenzen in den Vektoren
# jeweils auf der gleichen Indexposition liegen. Man verschiebt die Grenzen also
bins = (h$borders[-1]+h$borders[-length(h$borders)])/2

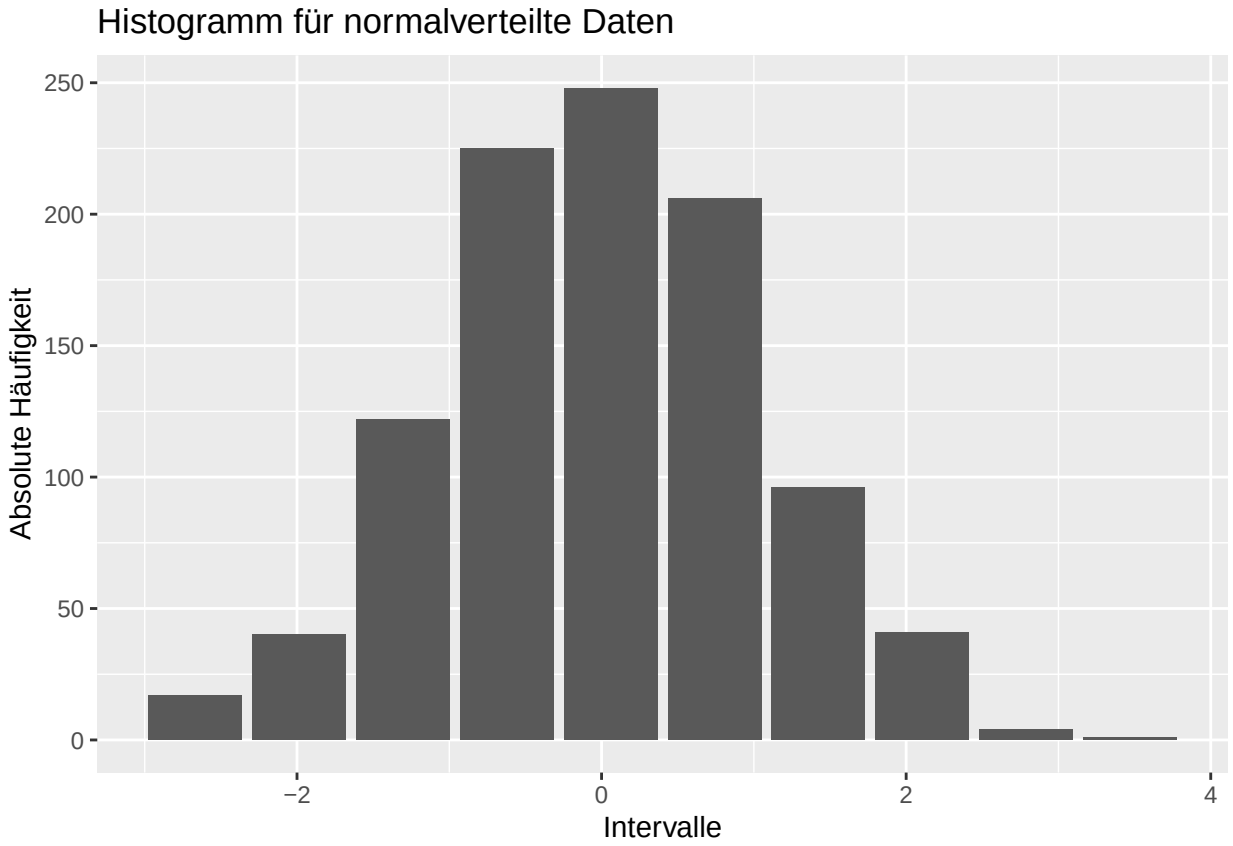
#print(h$borders[-1])
#print(h$borders[length(-h$borders)])

print(h$borders)
```

```
##  [1] -3.0080486 -2.3262161 -1.6443835 -0.9625510 -0.2807185  0.4011140
##  [7]  1.0829466  1.7647791  2.4466116  3.1284442  3.8102767
```

```r
# Damit kann man die relative statt der absoluten Häufigkeit anzeigen
count = h$counts #/sum(h$counts)

ggplot() +
 geom_bar(aes(x=bins, y=count), stat="identity")+
  labs(title="Histogramm für normalverteilte Daten", y="Absolute Häufigkeit", x="Intervalle")
```




```
print(h$counts)
```

```
##  [1]  17  40 122 225 248 206  96  41   4   1
```

Plot 2

Visualisierung von Datensätzen

In diesem Abschnitt sollen alle Graphiken mit *ggplot* und alle Tabellen mit `kable` erstellt werden.

Säuglingssterblichkeit und Wirtschaftsleistung

Nutzen Sie den Datensatz `carData::UN`. In der Hilfe finden Sie Hinweise, was dort gezeigt ist. Beachten Sie, dass für einige Länder Werte fehlen (`NA`); diese Länder sollen nicht berücksichtigt werden.

Wirtschaftsleistung vs. Säuglingssterblichkeit

Erzeugen Sie diese Graphik, indem Sie den nachfolgenden Chunk vervollständigen. Die gezeigten Länder sind Germany, United States, China, India, Afghanistan, Sierra Leone, Japan, Cuba.

Tipp: Sie dürfen (und sollen) weitere Libraries nutzen, wenn diese hilfreich sind.

```
# Ihre Lösung:
library(knitr) # für Darstellung in Tabellen

un = carData::UN # Datensatz laden
kable(un[1:20,]) # Datensatz als Tabelle darstellen, ersten 10 Reihen/Zeilen
```

	region	group	fertility	ppgdp	lifeExpF	pctUrban	infantMortality
Afghanistan	Asia	other	5.968	499.0	49.49	23	124.53500
Albania	Europe	other	1.525	3677.2	80.40	53	16.56100
Algeria	Africa	africa	2.142	4473.0	75.00	67	21.45800
American Samoa	NA	NA	NA	NA	NA	NA	11.29389
Angola	Africa	africa	5.135	4321.9	53.17	59	96.19100
Anguilla	Caribbean	other	2.000	13750.1	81.10	100	NA
Argentina	Latin Amer	other	2.172	9162.1	79.89	93	12.33700
Armenia	Asia	other	1.735	3030.7	77.33	64	24.27200
Aruba	Caribbean	other	1.671	22851.5	77.75	47	14.68700
Australia	Oceania	oecd	1.949	57118.9	84.27	89	4.45500
Austria	Europe	oecd	1.346	45158.8	83.55	68	3.71300
Azerbaijan	Asia	other	2.148	5637.6	73.66	52	37.56600
Bahamas	Caribbean	other	1.877	22461.6	78.85	84	14.13500
Bahrain	Asia	other	2.430	18184.1	76.06	89	6.66300
Bangladesh	Asia	other	2.157	670.4	70.23	29	41.78600
Barbados	Caribbean	other	1.575	14497.3	80.26	45	12.28400
Belarus	Europe	other	1.479	5702.0	76.37	75	6.49400
Belgium	Europe	oecd	1.835	43814.8	82.81	97	3.73900
Belize	Latin Amer	other	2.679	4495.8	77.81	53	16.20000
Benin	Africa	africa	5.078	741.1	58.66	42	76.67400

```r
# Enfernen der Zeilen, in denen NA Werte auftreten
un_cleaned = un[complete.cases(un), ] #


# Vergleichen, wie viele NA Werte herausgefallen sind
print(nrow(un))
```

```
## [1] 213
```

```r
print(nrow(un_cleaned))
```

```
## [1] 193
```

```r
library(ggplot2) # für Grafiken
library(ggrepel) # für die Ländernamen


# in der Grafik beschriftete Länder als Vektor definieren
markierte_laender = c(
  "Germany", "United States", "China", "India",
  "Afghanistan", "Sierra Leone", "Japan", "Cuba"
)

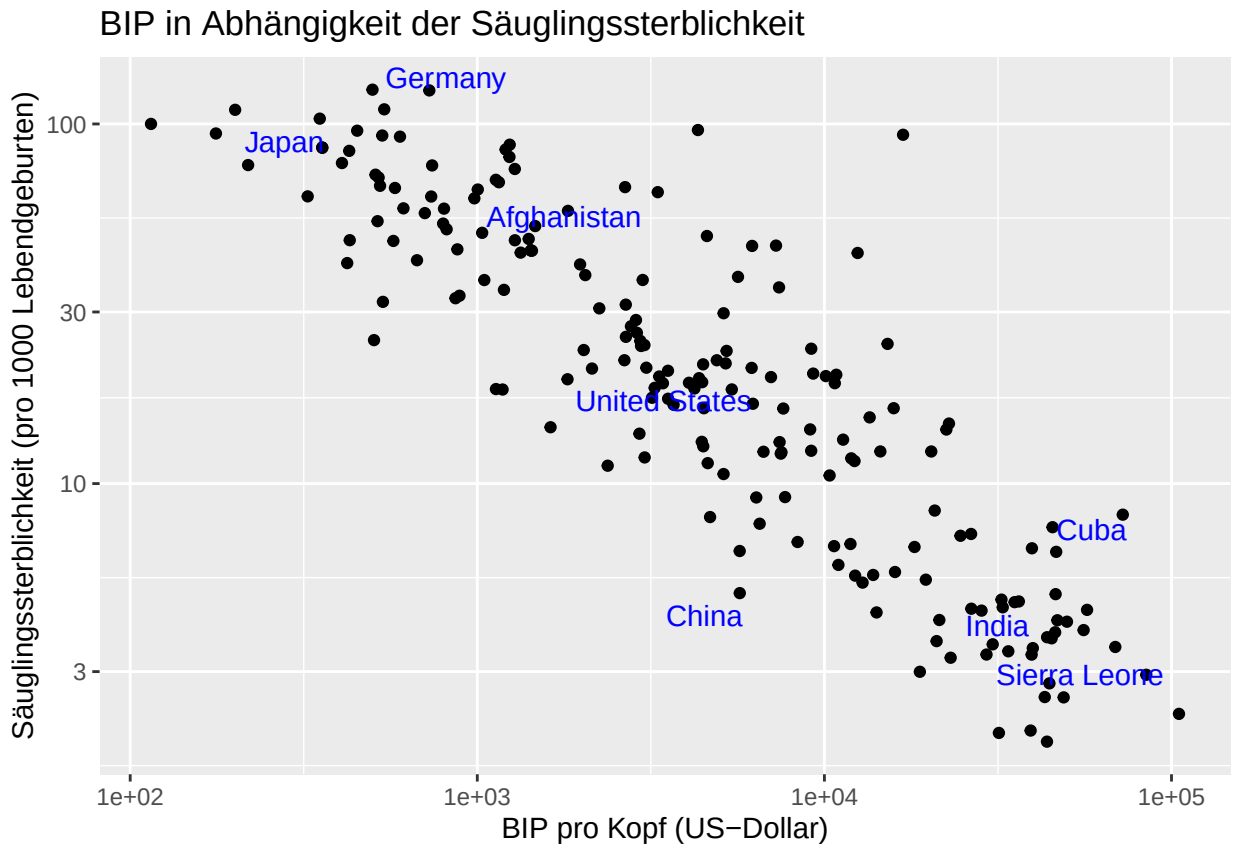
# Rückgabe df in dem nur Zeilen der markierten Länder vorkommen
# alle Spalten werden übernommen
labeldaten = un_cleaned[rownames(un_cleaned) %in% markierte_laender, ]

# gibt Liste mit TRUE und FALSE Werten zurück
print(rownames(un_cleaned) %in% markierte_laender)
```

```
##   [1]  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [25] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE
##  [37] FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
##  [49] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [61] FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [73] FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [85] FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [97] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [109] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [121] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [133] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [145] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE
## [157] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [169] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [181] FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [193] FALSE
```

```r
# eigentlichen Plot erzeugen
# df übergeben, x und y definieren
ggplot(un_cleaned, aes(x = ppgdp, y = infantMortality)) +
  geom_point() + # für Scatterplot/Punktdiagramm
  # Text in Grafik einfügen, "labeldaten" enthält nur Reihen der markierten Länder
  geom_text_repel(data = labeldaten, color="blue", aes(label = markierte_laender)) +
  scale_x_log10() + # log x Achse
  scale_y_log10() + # log y Achse
```

```
  labs(
    title= "BIP in Abhängigkeit der Säuglingssterblichkeit",
    x = "BIP pro Kopf (US-Dollar)",
    y = "Säuglingssterblichkeit (pro 1000 Lebendgeburten)"
  )
```



Säuglingssterblichkeit im Verhältnis zur Wirtschaftsleistung

Geben Sie diejenigen 10 Länder als Tabelle im Notebook aus, die das größte Verhältnis r von Säuglingssterblichkeit zu Wirtschaftsleistung haben. Dabei ist r die Säuglingssterblichkeit (pro 1000 Lebendgeburten) je 1000 US-Dollar BIP pro Kopf. Die Liste soll nach r absteigend sortiert sein und den Ländernamen und r enthalten.

Vervollständigen Sie diesen Chunk:

```
# Ihre Lösung:
library(dplyr) # für mutate
```

```
##
## Attache Paket: 'dplyr'
```

```
## Die folgenden Objekte sind maskiert von 'package:stats':
##
##     filter, lag
```

```
## Die folgenden Objekte sind maskiert von 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
# Funktion, die Relation berechnet
func_relation = function(ppgdp, infantMortality) ppgdp / infantMortality

# neue Spalte zum df hinzufügen mit mutate, die Relationswerte enthält
un_r = un_cleaned |> mutate(relation = func_relation(ppgdp, infantMortality))

werte = data.frame(
  Land = rownames(un_r),
  Relation = un_r$relation) # df mit "Land" und "Relation" erstellen

top10 = werte |>
  # sortierte Liste der 10 größten Werte zurückgeben
  slice_max(order_by = Relation, n=10)

kable(top10)
```

Land	Relation
Luxembourg	45913.24
Norway	28771.67
Singapore	22851.30
Switzerland	19607.23
Sweden	19224.14
Iceland	19094.80
Japan	16924.64
Finland	15990.55
Hong Kong	15707.65
Denmark	14264.23

Geben Sie nun – wie eben – diejenigen 10 Länder als Tabelle aus, die das **kleinste** Verhältnis r haben. Die Liste soll nach r absteigend sortiert sein.

Vervollständigen Sie diesen Chunk:

```r
# Ihre Lösung:
bottom10 = werte |>
  # diesmal die 10 kleinsten Relation Werte zurückgeben
  slice_min(order_by = Relation, n=10) |>
  # arrange in descending order, Erabsteigend nach Relation sortieren
  arrange(desc(Relation))

kable(bottom10)
```

Land	Relation
Guinea-Bissau	4.911763
Central African Republic	4.706570
Niger	4.168026
Malawi	4.152917
Afghanistan	4.006906
Sierra Leone	3.399414
Liberia	2.844391
Burundi	1.877066
Democratic Republic of the Congo	1.832348

Land	Relation
Somalia	1.147805

Kieferwachstum bei Kindern

Nutzen Sie den Datensatz `nlme::Orthodont`. In der Hilfe finden Sie Hinweise, was dort gezeigt ist.

Überblick über Verlauf bei allen Kindern

Plotten Sie im folgenden Chunk den Verlauf des Abstands zwischen Hypophyse und Fossa pterygopalatina (y-Achse) in Abhängigkeit vom Alter (x-Achse). Dies soll in einem Diagramm mit Unterdiagrammen erfolgen: ein Unterdiagramm zeigt den Verlauf für je ein Kind (Subject) und dessen Geschlecht (Male oder Female).

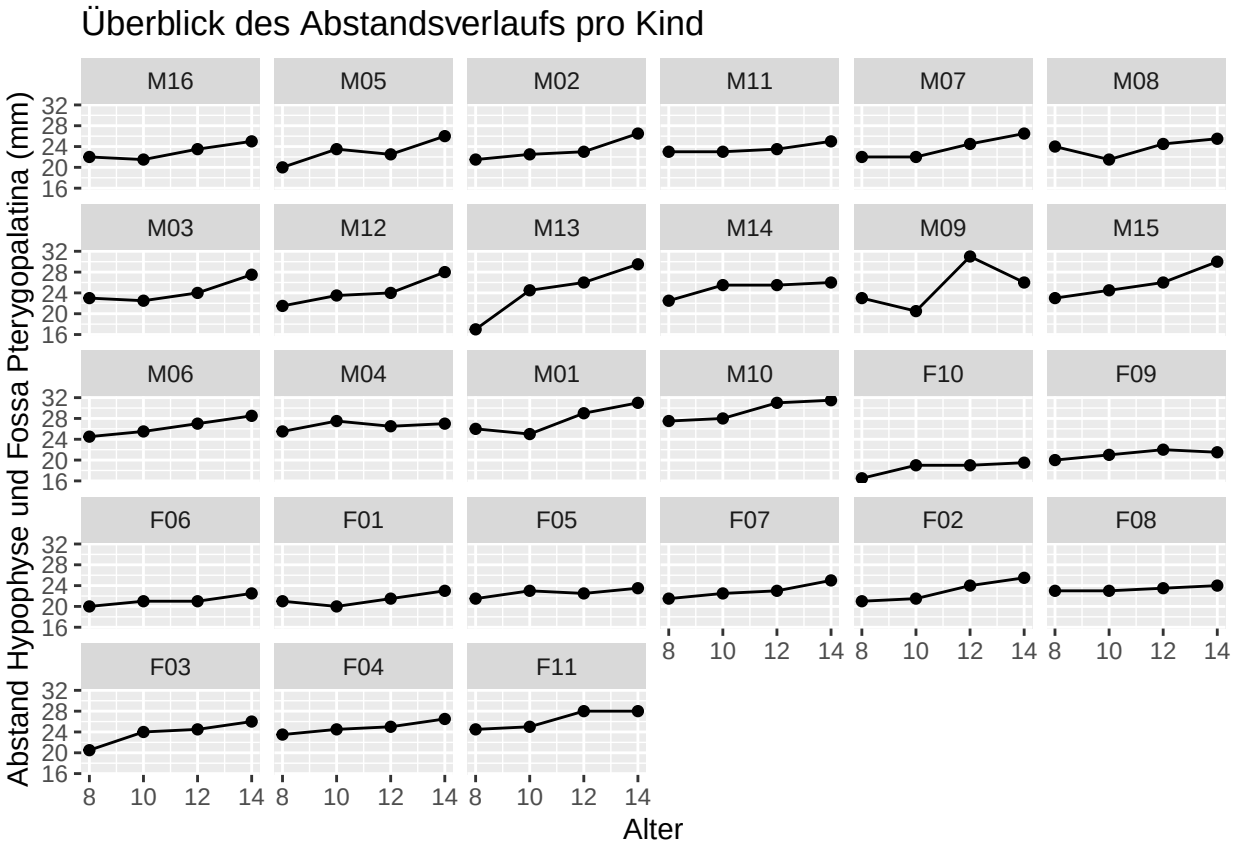
```
# Ihre Lösung:
orthodont = nlme::Orthodont # Datensatz einlesen

kable(orthodont[1:10, ]) # Datensatz als Tabelle anzeigen, die 10 ersten Zeilen
```

distance	age	Subject	Sex
26.0	8	M01	Male
25.0	10	M01	Male
29.0	12	M01	Male
31.0	14	M01	Male
21.5	8	M02	Male
22.5	10	M02	Male
23.0	12	M02	Male
26.5	14	M02	Male
23.0	8	M03	Male
22.5	10	M03	Male

```
ggplot(orthodont) +
  geom_line(aes(x=age, y=distance)) + # Liniendiagramm
  geom_point(aes(x=age, y=distance)) + # Punkte hinzufügen

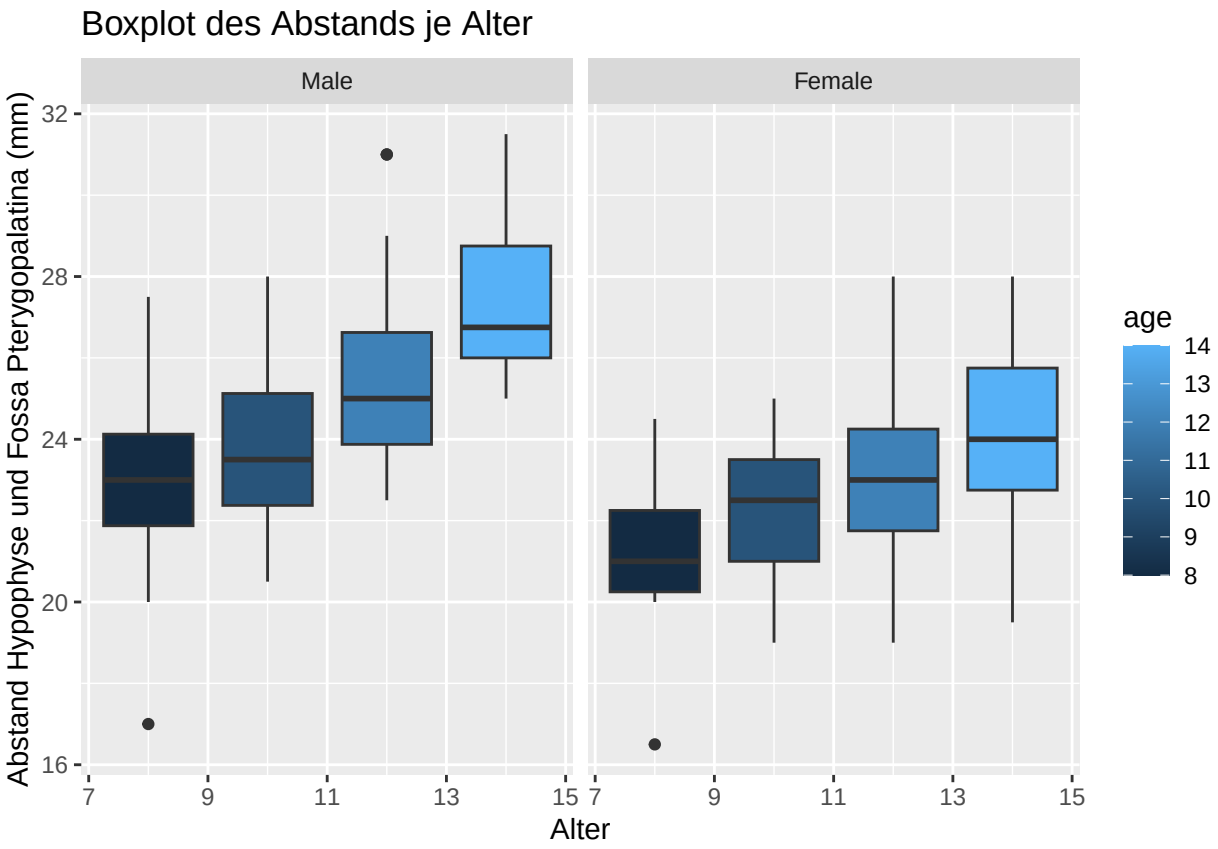
  # für jedes Subjekt ein Diagramm
  # M und F im Namen zeigt bereits, um welches Geschlecht es sich handelt
  facet_wrap(~ Subject) +
  labs(title="Überblick des Abstandsverlaufs pro Kind",
       x="Alter", y="Abstand Hypophyse und Fossa Pterygopalatina (mm)")
```



Boxplots des Abstands je Alter

Erzeugen Sie ein Diagramm, das in zwei Unterdiagrammen für die Jungen und die Mädchen Boxplots erstellt. Die Boxplots geben die Verteilung des Abstands je Alter an. In Anlehnung an das obige Diagramm sollen die Boxplots vertikal ausgerichtet sein.

```
# Ihre Lösung:
ggplot(orthodont) +
  # Boxplots für jedes Alter, fill für Farbe
  geom_boxplot(aes(x=age, y=distance, group=age, fill=age)) +
  facet_wrap(~ Sex) + # zwei Diagramme, eines für Male eines für Female
  labs(title="Boxplot des Abstands je Alter",
       x="Alter", y="Abstand Hypophyse und Fossa Pterygopalatina (mm)")
```


Boxplot des Abstands je Alter
Male
Female
Abstand Hypophyse und Fossa Pterygopalatina (mm)
32
28
24
20
16
7
9
11
13
15
Alter
age
14
13
12
11
10
9
8

3 Krebsfälle in Deutschland

3 Krebsfälle in Deutschland

Alle benötigten Libraries aktivieren

```
library(ggplot2)
library(dplyr)
```

```
##
## Attache Paket: 'dplyr'
```

```
## Die folgenden Objekte sind maskiert von 'package:stats':
##
##     filter, lag
```

```
## Die folgenden Objekte sind maskiert von 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(tidyr)
library(tibble)
```

3.1 Einlesen der Daten und Data Preprocessing

```
krebs_df = read.csv2(
  "Krebsdaten.csv",
  header = FALSE,
  skip = 1, # skips the first row
  stringsAsFactors = FALSE
  # FALSE because we will convert all data types manually later
)
```

```
head(krebs_df, 5)
```

```
##     V1                              V2       V3           V4      V5 V6
## 1 2004 Krebs gesamt (C00-C97 ohne C44) weiblich       0 - 44  69.417 NA
## 2 2004 Krebs gesamt (C00-C97 ohne C44) weiblich      45 - 54  92.221 NA
## 3 2004 Krebs gesamt (C00-C97 ohne C44) weiblich      55 - 64 149.642 NA
## 4 2004 Krebs gesamt (C00-C97 ohne C44) weiblich      65 - 74 174.556 NA
## 5 2004 Krebs gesamt (C00-C97 ohne C44) weiblich 75 und älter 188.884 NA
```

Letzte leere Spalte entfernen

```
krebs_df = krebs_df[, 1:5]
```

Spalten Namen benennen

```
colnames(krebs_df) = c(
  "Jahr",
  "Diagnose",
  "Geschlecht",
  "Altersgruppe",
  "Prävalenz"
)
```

```
head(krebs_df, 5)
```

```
##   Jahr                         Diagnose Geschlecht Altersgruppe Prävalenz
## 1 2004 Krebs gesamt (C00-C97 ohne C44)   weiblich       0 - 44    69.417
## 2 2004 Krebs gesamt (C00-C97 ohne C44)   weiblich      45 - 54    92.221
## 3 2004 Krebs gesamt (C00-C97 ohne C44)   weiblich      55 - 64   149.642
## 4 2004 Krebs gesamt (C00-C97 ohne C44)   weiblich      65 - 74   174.556
## 5 2004 Krebs gesamt (C00-C97 ohne C44)   weiblich 75 und älter   188.884
```

```
dim(krebs_df)
```

```
## [1] 6300    5
```

Data Frame in ein Tibble umwandeln

```
krebs_tib = as_tibble(krebs_df)

head(krebs_tib, 5)
```

```
## # A tibble: 5 x 5
##    Jahr Diagnose                         Geschlecht Altersgruppe Prävalenz
##   <int> <chr>                            <chr>      <chr>        <chr>
## 1  2004 Krebs gesamt (C00-C97 ohne C44) weiblich   0 - 44       69.417
## 2  2004 Krebs gesamt (C00-C97 ohne C44) weiblich   45 - 54      92.221
## 3  2004 Krebs gesamt (C00-C97 ohne C44) weiblich   55 - 64      149.642
## 4  2004 Krebs gesamt (C00-C97 ohne C44) weiblich   65 - 74      174.556
## 5  2004 Krebs gesamt (C00-C97 ohne C44) weiblich   75 und älter 188.884
```

Datentypen aller Spalten überprüfen

```
sapply(krebs_tib, class)
```

```
##         Jahr     Diagnose   Geschlecht Altersgruppe    Prävalenz
##    "integer"  "character"  "character"  "character"  "character"
```

Spalten vor der Data Type Conversion genauer analysieren

For each column, we should check all unique values so that we can see if there are any Null values, etc... For the `Prävalenz` column, we need to do some additional analysis steps.

1. Werte der Spalte `Jahr` checken

```
sort(unique(krebs_tib$Jahr))
```

```
##  [1] 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014 2015 2016 2017 2018
## [16] 2019 2020 2021 2022 2023 2024
```

2. Werte der Spalte `Diagnose` checken

```r
unique(krebs_tib$Diagnose)
```

```
##  [1] "Krebs gesamt (C00-C97 ohne C44)"
##  [2] "Mundhöhle und Rachen (C00-C14)"
##  [3] "Speiseröhre (C15)"
##  [4] "Magen (C16)"
##  [5] "Dünndarm (C17)"
##  [6] "Darm (C18-C20)"
##  [7] "Anus u. Analkanal (C21)"
##  [8] "Leber (C22)"
##  [9] "Gallenblase und Gallenwege (C23-C24)"
## [10] "Bauchspeicheldrüse (C25)"
## [11] "Kehlkopf (C32)"
## [12] "Lunge (C33-C34)"
## [13] "Malignes Melanom der Haut (C43)"
## [14] "Mesotheliom (C45)"
## [15] "Weichteilgewebe ohne Mesotheliom (C46-C49)"
## [16] "Brustdrüse (C50)"
## [17] "Vulva (C51)"
## [18] "Gebärmutterhals (C53)"
## [19] "Gebärmutterkörper (C54-C55)"
## [20] "Eierstöcke (C56)"
## [21] "Prostata (C61)"
## [22] "Hoden (C62)"
## [23] "Niere (C64)"
## [24] "Harnblase (C67)"
## [25] "Zentrales Nervensystem (C70-C72)"
## [26] "Schilddrüse (C73)"
## [27] "Morbus Hodgkin (C81)"
## [28] "Non-Hodgkin-Lymphome (C82-C88)"
## [29] "Multiples Myelom (C90)"
## [30] "Leukämien (C91-C95)"
```

3. Werte der Spalte `Geschlecht` checken

```r
unique(krebs_tib$Geschlecht)
```

```
## [1] "weiblich" "männlich"
```

4. Werte der Spalte `Altersgruppe` checken

```r
unique(krebs_tib$Altersgruppe)
```

```
## [1] "0 - 44"       "45 - 54"      "55 - 64"      "65 - 74"      "75 und älter"
```

5. Werte der Spalte `Prävalenz` checken

We should manually check if there are any non-numeric values. We can simply convert the entire column to numeric and anything that is not able to be converted will automatically be converted to `NA`. Then we can just count those `NA`s to see how many they are. *(It's important to know how many they are so that later we can either fill them or delete the entire row, etc...)*

```r
# Original values
krebs_prävalenz = krebs_tib$Prävalenz

# Converting to numeric
converted = as.numeric(gsub("\\.", "", krebs_prävalenz))
```

```
## Warning: NAs durch Umwandlung erzeugt
```

```
# Find all values that could not be converted
invalid_values = krebs_prävalenz[is.na(converted)]

# Show all the non-numeric values that were not converted
unique(invalid_values)
```

```
## [1] "x"
```

```
length(invalid_values)
```

```
## [1] 630
```

Deleting those rows will lead to a loss of information, so we will keep them.

```
colSums(is.na(krebs_tib))
```

Fehlende Werte pro Spalte prüfen (vor der Data Type Conversion)

```
##         Jahr     Diagnose   Geschlecht Altersgruppe    Prävalenz
##            0            0            0            0            0
```

Convert Data Types

```
krebs_tib = krebs_tib |>
  mutate(
    Geschlecht = as.factor(Geschlecht),
    Altersgruppe = as.factor(Altersgruppe),
    Prävalenz = as.numeric(gsub("\\.", "", Prävalenz))
    # Removes the '.' from '600.000' => '600000'
  )
```

```
## Warning: There was 1 warning in `mutate()`.
## i In argument: `Prävalenz = as.numeric(gsub("\\.", "", Prävalenz))`.
## Caused by warning:
## ! NAs durch Umwandlung erzeugt
```

```
sapply(krebs_tib, class)
```

Datentypen nach der Conversion checken

```
##         Jahr     Diagnose   Geschlecht Altersgruppe    Prävalenz
##    "integer"  "character"     "factor"     "factor"    "numeric"
```

Noch einmal auf fehlende Werte checken *We see that these 630 missing values in* ***Prävalenz*** *are the values we analysed above.*

```
colSums(is.na(krebs_tib))
```

```
##         Jahr     Diagnose   Geschlecht Altersgruppe    Prävalenz
##            0            0            0            0          630
```

Warum enthält der Datensatz x-Werte?

Bei der Datenaufbereitung wurden **630 Einträge (Cells) mit dem Wert `x`** in der Spalte `Prävalenz` gefunden.

Laut der Datenbank kennzeichnet `x` eine **nicht sinnvolle Merkmalskombination**, also zum Beispiel bei geschlechtsspezifischen Krebsdiagnosen.

Das sind alle Kombinationen:

Diagnose	Geschlecht	Anzahl `x`
Eierstöcke (C56)	männlich	105
Gebärmutterhals (C53)	männlich	105
Gebärmutterkörper (C54–C55)	männlich	105
Vulva (C51)	männlich	105
Hoden (C62)	weiblich	105
Prostata (C61)	weiblich	105

Beispiel:

Prostatakrebs ist in dieser Statistik eine männliche Krebsdiagnose. Deshalb wird für die Kombination `Prostata (C61)` und `weiblich` keine numerische Prävalenz angegeben, sondern `x`.

Die Werte sind also **absichtlich nicht angegeben**. Es handelt sich nicht um zufällig fehlende Messwerte. Also macht es auch keinen Sinn die Werte zu **imputieren**.

Lösung: Geben Sie die ersten 10 Zeilen und 7 Spalten als Tabelle aus

```
head(krebs_tib, 10)
```

```
## # A tibble: 10 x 5
##     Jahr Diagnose                          Geschlecht Altersgruppe Prävalenz
##    <int> <chr>                             <fct>      <fct>            <dbl>
##  1  2004 Krebs gesamt (C00-C97 ohne C44) weiblich   0 - 44           69417
##  2  2004 Krebs gesamt (C00-C97 ohne C44) weiblich   45 - 54          92221
##  3  2004 Krebs gesamt (C00-C97 ohne C44) weiblich   55 - 64         149642
##  4  2004 Krebs gesamt (C00-C97 ohne C44) weiblich   65 - 74         174556
##  5  2004 Krebs gesamt (C00-C97 ohne C44) weiblich   75 und älter    188884
##  6  2004 Krebs gesamt (C00-C97 ohne C44) männlich   0 - 44           45244
##  7  2004 Krebs gesamt (C00-C97 ohne C44) männlich   45 - 54          54175
##  8  2004 Krebs gesamt (C00-C97 ohne C44) männlich   55 - 64         156830
##  9  2004 Krebs gesamt (C00-C97 ohne C44) männlich   65 - 74         244275
## 10  2004 Krebs gesamt (C00-C97 ohne C44) männlich   75 und älter    167108
```

Note

Im originalen Datensatz haben wir nur 5 Spalten und nicht 7.

3.2 Übersicht aller Krebsarten

Daten für die Visualization vorbereiten

```
präv_krebs = krebs_tib |>
  filter(Jahr == 2024 & Altersgruppe == "55 - 64" &
           Diagnose != "Krebs gesamt (C00-C97 ohne C44)") |>
```

```
  drop_na(Prävalenz)

dim(präv_krebs)
```

```
## [1] 52  5
```

```
head(präv_krebs, 5)
```

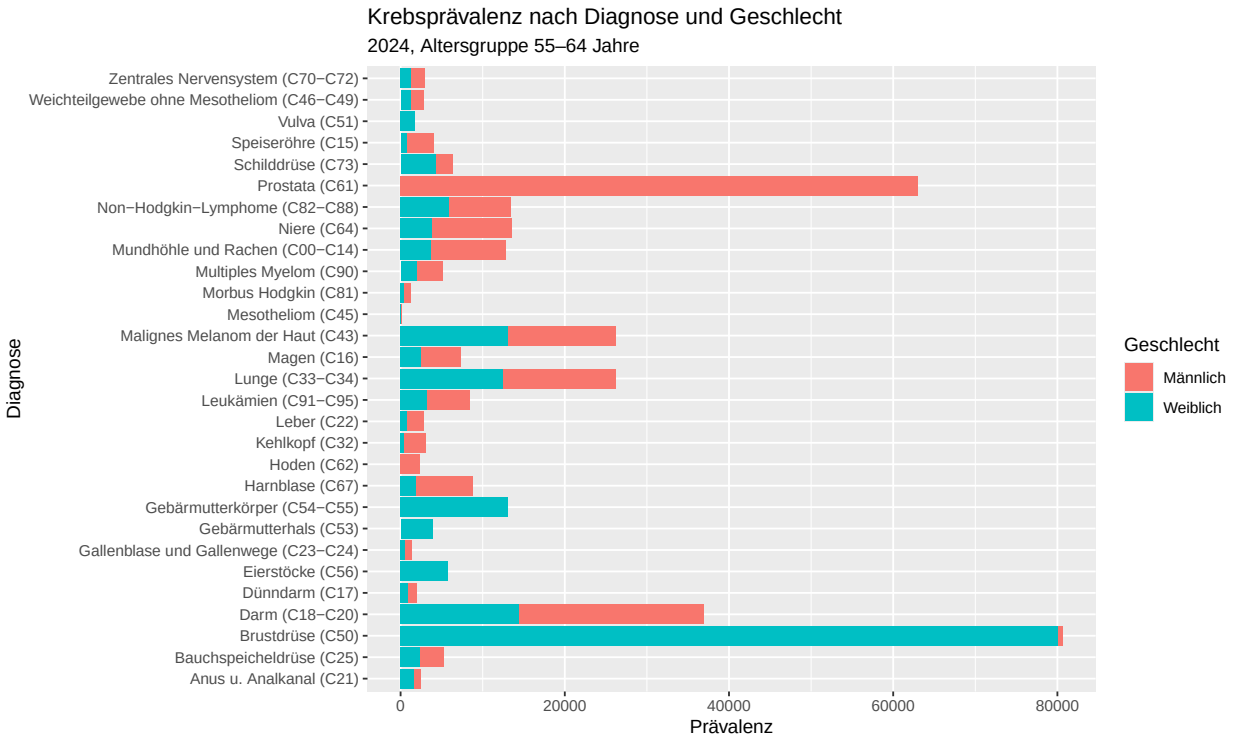
```
## # A tibble: 5 x 5
##    Jahr Diagnose                          Geschlecht Altersgruppe Prävalenz
##   <int> <chr>                             <fct>      <fct>            <dbl>
## 1  2024 Mundhöhle und Rachen (C00-C14) weiblich   55 - 64           3777
## 2  2024 Mundhöhle und Rachen (C00-C14) männlich   55 - 64           9055
## 3  2024 Speiseröhre (C15)              weiblich   55 - 64            727
## 4  2024 Speiseröhre (C15)              männlich   55 - 64           3273
## 5  2024 Magen (C16)                    weiblich   55 - 64           2498
```

Note

We filter out "Krebs gesamt" because it isn't an actual Diagnosis.

Lösung: Balkendiagramm erstellen

```
ggplot(präv_krebs) +
  geom_bar(aes(y = Diagnose, x = Prävalenz, fill = Geschlecht), stat = "identity") +
  labs(
    title = "Krebsprävalenz nach Diagnose und Geschlecht",
    subtitle = "2024, Altersgruppe 55-64 Jahre"
  ) +
  scale_fill_discrete(
    labels = c("Männlich", "Weiblich")
  )
```



Begründung unserer Motivation für den Plot Design

Für die Position der Bars, haben wir den default value `stacked` benutzt, da es die unterschiedlichen Geschlechte schon klar darstellt. Eine andere Option wäre `dodge`, aber da der Plot schon ziemlich groß ist, würde es ein bisschen chaotisch sein. Zusatätzlich, haben wir die Legend Labels überschrieben und titel/ subtitel hinzugefügt.

3.3 Prävalenz nach Altergruppen und Geschlecht

Dataset kurz für einen Überblick zeigen

```
head(krebs_tib, 100)
```

```
## # A tibble: 100 x 5
##     Jahr Diagnose                          Geschlecht Altersgruppe Prävalenz
##    <int> <chr>                             <fct>      <fct>            <dbl>
##  1  2004 Krebs gesamt (C00-C97 ohne C44) weiblich   0 - 44           69417
##  2  2004 Krebs gesamt (C00-C97 ohne C44) weiblich   45 - 54          92221
##  3  2004 Krebs gesamt (C00-C97 ohne C44) weiblich   55 - 64         149642
##  4  2004 Krebs gesamt (C00-C97 ohne C44) weiblich   65 - 74         174556
##  5  2004 Krebs gesamt (C00-C97 ohne C44) weiblich   75 und älter    188884
##  6  2004 Krebs gesamt (C00-C97 ohne C44) männlich   0 - 44           45244
##  7  2004 Krebs gesamt (C00-C97 ohne C44) männlich   45 - 54          54175
##  8  2004 Krebs gesamt (C00-C97 ohne C44) männlich   55 - 64         156830
##  9  2004 Krebs gesamt (C00-C97 ohne C44) männlich   65 - 74         244275
## 10  2004 Krebs gesamt (C00-C97 ohne C44) männlich   75 und älter    167108
## # i 90 more rows
```

Daten für die Visualization vorbereiten

```
präv_krebs_2 = krebs_tib |>
  filter(Diagnose == "Krebs gesamt (C00-C97 ohne C44)") |>
  # We only keep the 'Krebs gesamt' Diagnosen
  mutate(
    Altersstufe = ifelse( # Add a new column Altersstufe consisting of
      Altersgruppe == "0 - 44" | Altersgruppe == "45 - 54",
      # values either `bis 54` or `ab 55` for each
      "bis 54", # existing Altersgruppe
      "ab 55"
    )
  ) |>
  group_by(Jahr, Geschlecht, Altersstufe) |>
  summarize("Prävalenz Altersstufe" = sum(Prävalenz))
```

```
## `summarise()` has regrouped the output.
## i Summaries were computed grouped by Jahr, Geschlecht, and Altersstufe.
## i Output is grouped by Jahr and Geschlecht.
## i Use `summarise(.groups = "drop_last")` to silence this message.
## i Use `summarise(.by = c(Jahr, Geschlecht, Altersstufe))` for per-operation
##   grouping (`?dplyr::dplyr_by`) instead.
```

```
  # Add a new column Prävalenz Altersstufe
  # containing the summed Prävalenz per Altersstufe

dim(präv_krebs_2)
```

```
## [1] 84  4
```

```
head(präv_krebs_2, 100)
```

```
## # A tibble: 84 x 4
## # Groups:   Jahr, Geschlecht [42]
##     Jahr Geschlecht Altersstufe `Prävalenz Altersstufe`
##    <int> <fct>      <chr>                         <dbl>
##  1  2004 männlich   ab 55                        568213
##  2  2004 männlich   bis 54                        99419
##  3  2004 weiblich   ab 55                        513082
##  4  2004 weiblich   bis 54                       161638
##  5  2005 männlich   ab 55                        595420
##  6  2005 männlich   bis 54                       101820
##  7  2005 weiblich   ab 55                        522429
##  8  2005 weiblich   bis 54                       163736
##  9  2006 männlich   ab 55                        627969
## 10  2006 männlich   bis 54                       105856
## # i 74 more rows
```

Here we don't have to drop `NA`*s because we are only using the* `Diagnose` *"Krebs gesamt" which contains all the summed* `Prävalenzen` *of each* `Diagnose`*, so we wouldn't get any 0 values/ Null values.*

Lösung: 2 Liniendiagramme erstellen

```
ggplot(präv_krebs_2, aes(Jahr, `Prävalenz Altersstufe`, color = Altersstufe)) +
  geom_line(linewidth = 1.5) +
  facet_wrap(~ Geschlecht) +
```

```r
  labs(
    title = "Krebsprävalenz nach Altersstufe und Geschlecht",
    subtitle = "2004 bis 2024, Alterstufen: bis 54 & ab 55"
  ) +
  scale_y_continuous(
    labels = scales::label_number(),
    limits = c(0, NA)
  )
```

