Testat 1

Gruppe 1: Jan Hoffmann, Jannis Seefeld, Rabea Götz

1 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.

```
myhistogram = function(x, n=30, min=NA, max=NA) {
    sorted = sort(x)
    min=ifelse(is.na(min), min(x), min)
    max=ifelse(is.na(max), max(x) + 1, max)

    sortedNumbers = sorted[sorted >= min]
    sortedNumbers = sortedNumbers[sortedNumbers < max]

    outside = sorted[sorted < min]
    outside = sorted[sorted > max]

    if (length(outside) > 0) {
        warning(paste0('Zahlen außerhalb von Intervallgrenzen: ', paste(outside, collapse = ", ")))
    }

    borders = seq(min, max, length.out=n+1)
    counts = sapply(seq(from=1, to=n), function(e) {
        min = borders[e]
        max = borders[e+1]

        intervall = sortedNumbers[sortedNumbers >= min]
        intervall = intervall[intervall < max]
```

```
        length(intervall)
    })

    list(borders, counts)
}
```

2.1 Beispieldaten

1.2.1 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)
myhistogram(x, n = 10, min = -5, max = 6)
```

```
## [[1]]
##  [1] -5.0 -3.9 -2.8 -1.7 -0.6  0.5  1.6  2.7  3.8  4.9  6.0
##
## [[2]]
##  [1] 4 3 3 4 3 3 4 3 3 3
```

1.2.2 Beispiel 2

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

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

1.2.3 Beispiel 3

```
x = seq(-5, 5, by = 1 / 5)
myhistogram(x, n = 7, min = -10, max = 10)
```

```
## [[1]]
## [1] -10.000000  -7.142857  -4.285714  -1.428571   1.428571   4.285714   7.142857
## [8]  10.000000
##
## [[2]]
## [1]  0  4 14 15 14  4  0
```

1.2.4 Beispiel 4

```
x = seq(-100, 100, by = 1)
myhistogram(x, n = 15)
```

```
## [[1]]
##  [1] -100.0  -86.6  -73.2  -59.8  -46.4  -33.0  -19.6   -6.2    7.2   20.6
## [11]   34.0   47.4   60.8   74.2   87.6  101.0
##
```

```
## [[2]]
##  [1] 14 13 14 13 13 14 13 14 13 13 14 13 14 13 13
```

1.2.5 Beispiel 5

```
x = seq(-10, 11, by = 1 / 5)
myhistogram(x, min = -10, max = 10)
```

```
## Warning in myhistogram(x, min = -10, max = 10): Zahlen außerhalb von
## Intervallgrenzen: 10.2, 10.4, 10.6, 10.8, 11
```

```
## [[1]]
##  [1] -10.0000000  -9.3333333  -8.6666667  -8.0000000  -7.3333333  -6.6666667
##  [7]  -6.0000000  -5.3333333  -4.6666667  -4.0000000  -3.3333333  -2.6666667
## [13]  -2.0000000  -1.3333333  -0.6666667   0.0000000   0.6666667   1.3333333
## [19]   2.0000000   2.6666667   3.3333333   4.0000000   4.6666667   5.3333333
## [25]   6.0000000   6.6666667   7.3333333   8.0000000   8.6666667   9.3333333
## [31]  10.0000000
##
## [[2]]
##  [1] 4 3 3 4 3 3 4 3 3 4 3 3 4 3 3 4 3 3 4 3 3 4 3 3 4 3 3 4 3 3 4 3 3
```

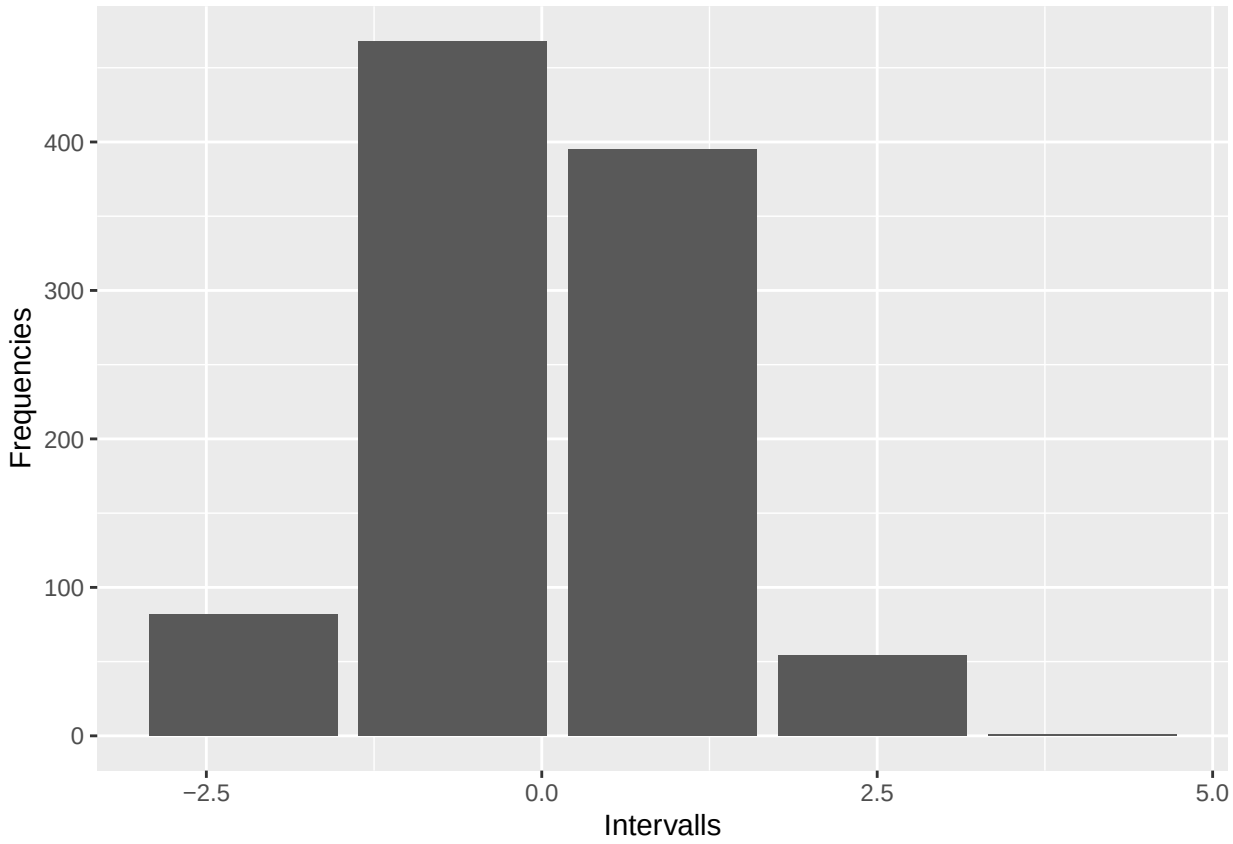
1.3 Barplot

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:

```
library(ggplot2)

set.seed(1)
x = rnorm(0, 1, n = 1000)
h = myhistogram(x, n = 5)
i = sapply(seq(from=1, to=length(h[[1]])-1), function(e) {
        lower = h[[1]][e]
        upper = h[[1]][e+1]

        (lower + upper) / 2
    })
df = data.frame(Intervalls=i, Frequencies=h[[2]])
colnames(df) = c('Intervalls', 'Frequencies')

ggplot(df) +
    geom_bar(aes(x=Intervalls, y=Frequencies), stat="identity", position="dodge")
```



2 Visualisierung von Datensätzen

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

Körpergewicht und Gehirngewicht bei Säugetieren

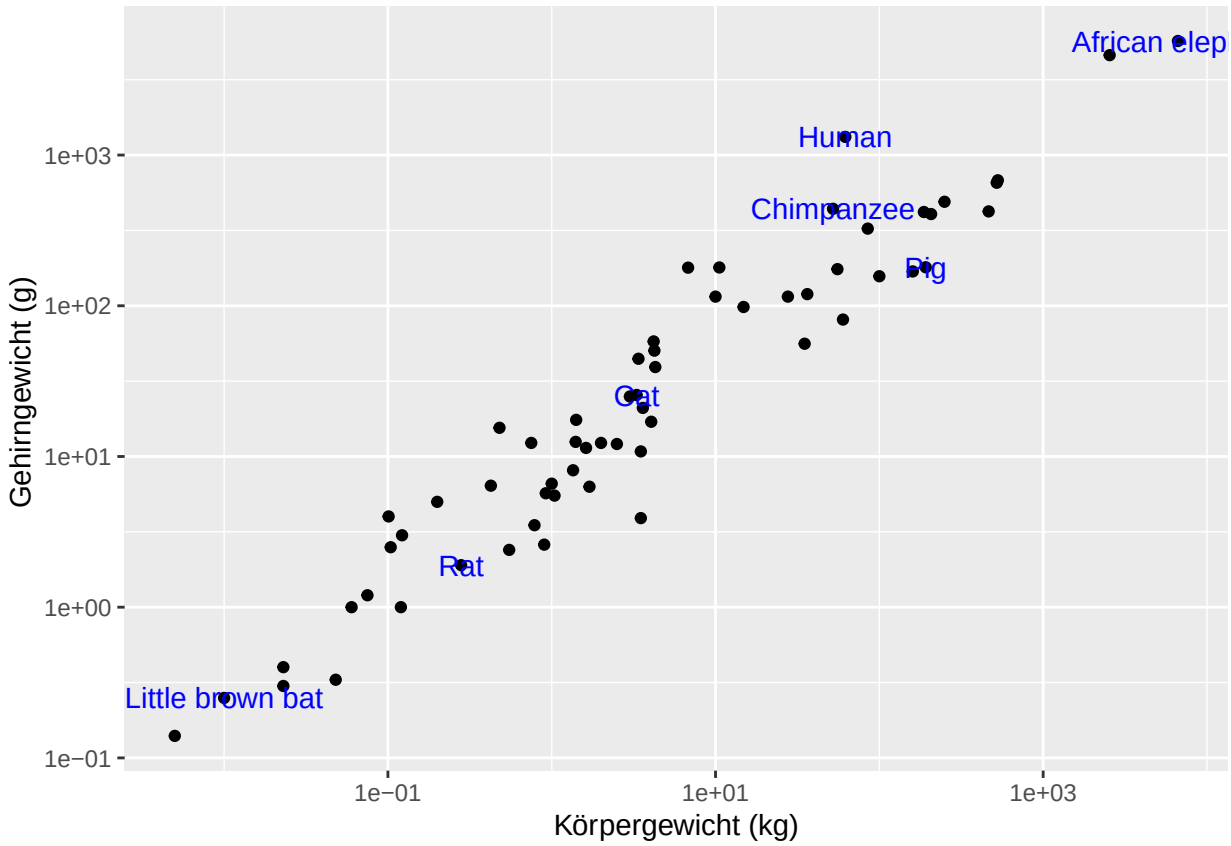
Nutzen Sie den Datensatz `MASS::mammals`. In der Hilfe finden Sie Hinweise, was dort gezeigt ist.

Körpergewicht vs. Gehirngewicht

Erzeugen Sie diese Graphik, indem Sie den nachfolgenden Chunk vervollständigen. Die gezeigten Tiernamen sind Pig, Rat, African elephant, Chimpanzee, Cat, Human, Little brown bat.

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

```
# Ihre Lösung:
mam = MASS::mammals
mam$name = row.names(mam)

lbl = c('Pig', 'Rat', 'African elephant', 'Chimpanzee', 'Cat', 'Human', 'Little brown bat')
lbl_data = mam[mam$name %in% lbl,]

ggplot(mam) +
 geom_point(aes(x=body, y=brain)) +
 scale_y_continuous('Gehirngewicht (g) ', trans='log10') +
 scale_x_continuous('Körpergewicht (kg)', trans='log10') +
 geom_text(aes(label=name, x=body, y=brain),color='blue', data=lbl_data)
```



Gehirn- zu Körpergewicht-Verhältnis

Geben Sie diejenigen 10 Tiere als Tabelle im Notebook aus, die das größte Gehirn- zu Körpergewicht-Verhältnis r haben. Die Liste soll nach r absteigend sortiert sein und den Tiernamen und r enthalten.

Vervollständigen Sie diesen Chunk:

```
# Ihre Lösung:
mam$r = mam$brain/mam$body
kable(head(mam[order(mam$r, decreasing=T),][c(4)],n=10))
```

	r
Ground squirrel	39.60396
Owl monkey	32.29167
Lesser short-tailed shrew	28.00000
Rhesus monkey	26.32353
Galago	25.00000
Little brown bat	25.00000
Mole rat	24.59016
Tree shrew	24.03846
Human	21.29032
Mouse	17.39130

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

Vervollständigen Sie diesen Chunk:

```
# Ihre Lösung:
kable(tail(mam[order(mam$r, decreasing=T),][4],n=10))
```

	r
Kangaroo	1.6000000
Jaguar	1.5700000
Giant armadillo	1.3500000
Giraffe	1.2854442
Horse	1.2571977
Water opossum	1.1142857
Brazilian tapir	1.0562500
Pig	0.9375000
Cow	0.9096774
African elephant	0.8584310

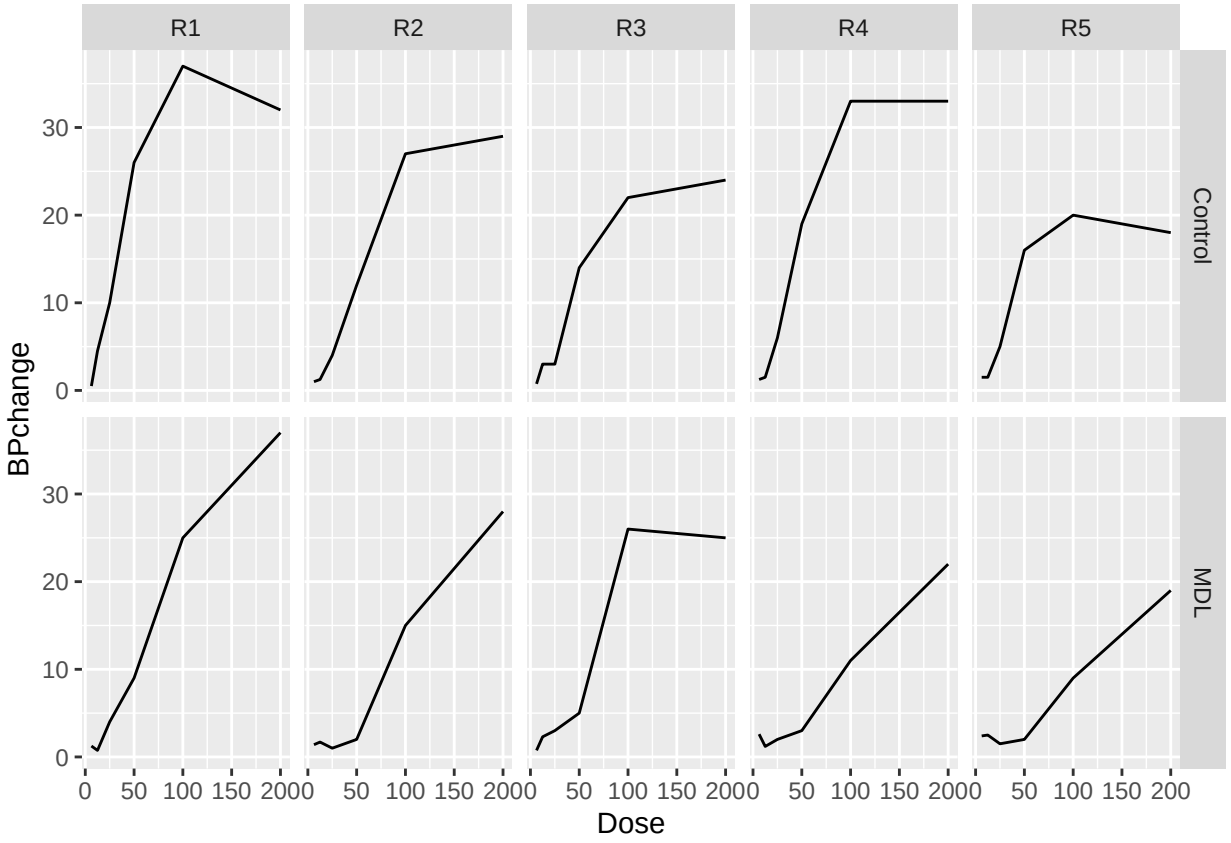
Blutdruckveränderung bei Medikamentengabe im Tierversuch

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

Überblick über Verlauf bei allen Kaninchen

Plotten Sie im folgenden Chunk den Verlauf der Blutdruckveränderung (y-Achse) bei gegebener Dosis Phenylbiguanide (x-Achse). Dies soll in einem Diagramm mit Unterdiagrammen erfolgen: ein Unterdiagramm zeigt den Verlauf für je ein Kaninchen und der Behandlung (Placebo oder MDL 72222).

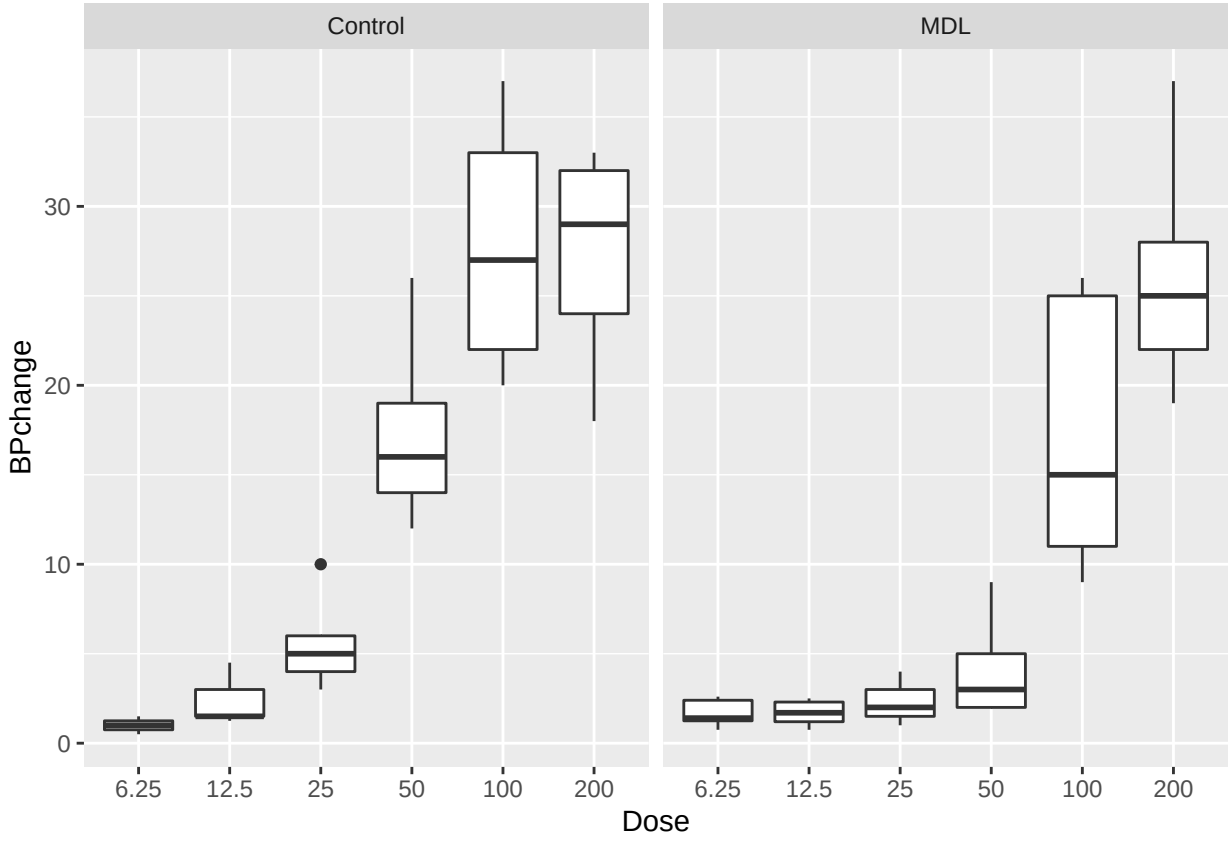
```
# Ihre Lösung:
ggplot(MASS::Rabbit) + geom_line(aes(y=BPchange, x=Dose)) + facet_grid( Treatment~Animal)
```



Boxplots der Blutdruckänderung je Dosis

Erzeugen Sie ein Diagramm, das in zwei Unterdiagrammen für die Placebo- und die MLD-Gruppe Boxplots erstellt. Die Boxplots geben die Verteilung der Blutdruckänderung je Dosis an. In Anlehnung an das obige Diagramm sollen die Boxplots vertikal ausgerichtet sein.

```
# Ihre Lösung:
ggplot(MASS::Rabbit) + geom_boxplot(aes(y=BPchange,x=factor(Dose), group=Dose)) +
  xlab("Dose")+
  facet_wrap(~Treatment)
```



3 Covid19: Impffortschritt in Deutschland

In dieser Aufgabe geht es um den Verlauf der Corona-Impfungen in Deutschland. Die folgenden URLs enthalten Daten ab 2020:

- https://impfdashboard.de/static/data/germany_vaccinations_timeseries_v2.tsv
- https://impfdashboard.de/static/data/germany_deliveries_timeseries_v2.tsv
- https://impfdashboard.de/static/data/germany_vaccinations_by_state.tsv

Sie sind der Webseite https://impfdashboard.de entnommen.

Einlesen der Daten

Lesen Sie die drei Dateien je in einen Data Frame ein mit den Variablennamen:

- `vacc`
- `deliv`
- `vaccState`

Wandeln Sie die Datums- und Zeitangaben von einem String in ein R-Datumsobjekt um. Geben Sie die ersten drei Zeilen und Spalten dieser Data Frames aus.

```
# Ihre Lösung:
# germany_vaccinations_timeseries_v2.tsv in Variable vacc
# germany_deliveries_timeseries_v2.tsv in Variable deliv
# germany_vaccinations_by_state.tsv in Variable vaccState


vacc = read.table('https://impfdashboard.de/static/data/germany_vaccinations_timeseries_v2.tsv',
                  sep='\t', header=T)

deliv = read.table('https://impfdashboard.de/static/data/germany_deliveries_timeseries_v2.tsv',
                   sep='\t', header=T)

vaccState = read.table('https://impfdashboard.de/static/data/germany_vaccinations_by_state.tsv',
                       sep='\t', header=T)

deliv$date = as.Date(deliv$date)
vacc$date = as.Date(vacc$date)


kable(head(vacc[c(1,2,3)], n=3))
```

date	dosen_kumulativ	dosen_biontech_kumulativ
2020-12-27	24421	24412
2020-12-28	42428	42417
2020-12-29	92483	92471

```
kable(head(deliv[c(1,2,3)], n=3))
```

date	impfstoff	impfstofftyp
2020-12-26	comirnaty	wildtyp
2020-12-26	comirnaty	wildtyp
2020-12-26	comirnaty	wildtyp

```
kable(head(vaccState[c(1,2,3)], n=3))
```

code	vaccinationsTotal	peopleFirstTotal
DE-BB	4999807	1722710
DE-BE	8488257	2899284
DE-BUND	543991	202088

Verimpfte Impfdosen pro Tag

Es soll untersucht werden, wie oft welcher Impfstoff an welchem Tag verimpft wurde.

Transformation

Der Data Frame `vacc` enthält leider keine Angaben, wie oft ein Impfstoff eines Herstellers täglich verabreicht wurde. Erzeugen Sie aus `vacc` einen neuen Data Frame `vacc2`, der die folgende Struktur hat:

Table 6: Neue Struktur: Data Frame `vacc2`.

Datum	Hersteller	Impfdosen pro Tag
09.04.21	biontech	123456
09.04.21	moderna	12345
...	...	...

Wie Sie die Impfstoffe (biontech, moderna, astra) nennen, bleibt Ihnen überlassen – solange die Bezeichnungen konsistent und schlüssig sind.

Geben Sie die letzten Zeilen von `vacc2` als `kable` aus. Tipp: `tail` gibt die letzten Zeilen eines Data Frames an (analog zu `head`).

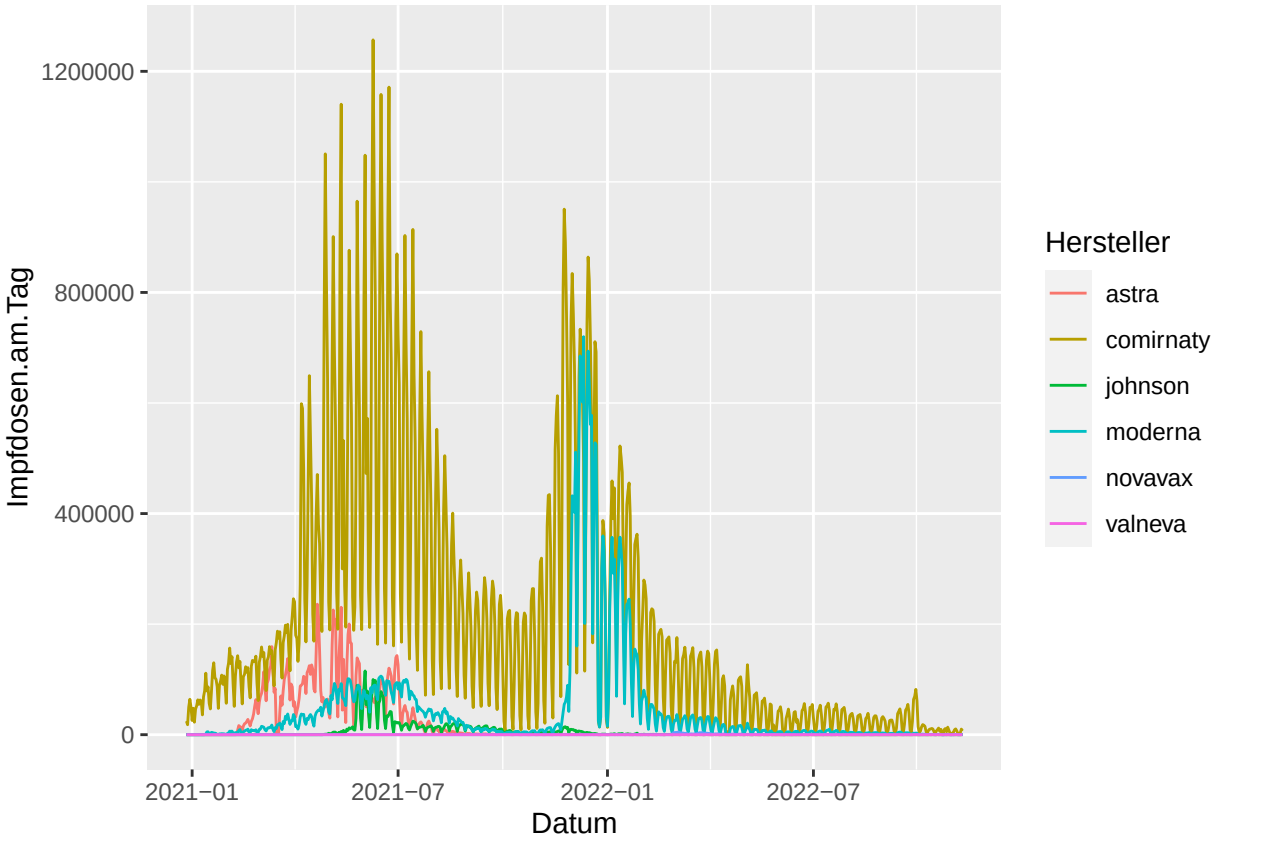
```
# Ihre Lösung:
data_set_size = length(vacc$date)

astra = (c(vacc$dosen_astra_kumulativ,0)-c(0,vacc$dosen_astra_kumulativ))[1:data_set_size]

comirnaty = (c(vacc$dosen_biontech_kumulativ,0)-c(0,vacc$dosen_biontech_kumulativ))[1:data_set_size]

johnson = (c(vacc$dosen_johnson_kumulativ,0)-c(0,vacc$dosen_johnson_kumulativ))[1:data_set_size]

moderna = (c(vacc$dosen_moderna_kumulativ,0)-c(0,vacc$dosen_moderna_kumulativ))[1:data_set_size]
novavax = (c(vacc$dosen_novavax_kumulativ,0)-c(0,vacc$dosen_novavax_kumulativ))[1:data_set_size]
valneva = (c(vacc$dosen_valneva_kumulativ,0)-c(0,vacc$dosen_valneva_kumulativ))[1:data_set_size]

vacc2 = data.frame(Datum=as.Date(vacc$date, "%d.%m.%Y"),
                   Hersteller=rep(c('astra', 'comirnaty', 'johnson', 'moderna','novavax','valneva'),
                                  each=data_set_size), "Impfdosen am Tag"=c(astra,comirnaty, johnson,
                                                                            moderna, novavax, valneva))


vacc2 = vacc2[order(vacc2$Datum),]
kable(tail(vacc2, n=10))
```

	Datum	Hersteller	Impfdosen.am.Tag
2048	2022-11-08	johnson	103
2731	2022-11-08	moderna	182
3414	2022-11-08	novavax	175
4097	2022-11-08	valneva	58
683	2022-11-09	astra	0
1366	2022-11-09	comirnaty	7053
2049	2022-11-09	johnson	85
2732	2022-11-09	moderna	129
3415	2022-11-09	novavax	201
4098	2022-11-09	valneva	65

Plot der täglichen Impfdosen nach Hersteller

Plotten Sie mit *ggplot* den Verlauf der täglichen Impfdosen für jeden Hersteller. Die x-Achse zeigt das Datum und die y-Achse die Anzahl der Impfdosen pro Tag. Überlegen Sie, welcher Diagrammtyp dafür am besten geeignet ist.

```
# Ihre Lösung:
ggplot(vacc2) + geom_line(aes(x=Datum, y=Impfdosen.am.Tag, group=Hersteller, color=Hersteller))
```



Zeitverzug Auslieferung bis Verimpfung

Es soll untersucht werden, wie schnell gelieferte Impfmengen der einzelnen Impfstoffe auch verimpft wurden.

Es bietet sich dafür an, die akkumulierten Impfdosen mit den akkumulierten Impflieferungen zeitlich plotten. Je größer die Lücke zwischen der Liefermenge und der Impfungen ist, desto mehr Impfstoff blieb liegen. Die Graphik soll Angaben für ganz Deutschland und nicht für die einzelnen Bundesländer zeigen.

Hinweis: Auch hier ist eine Vorverarbeitung der Daten nötig.

Plotten Sie dies mit *ggplot*:

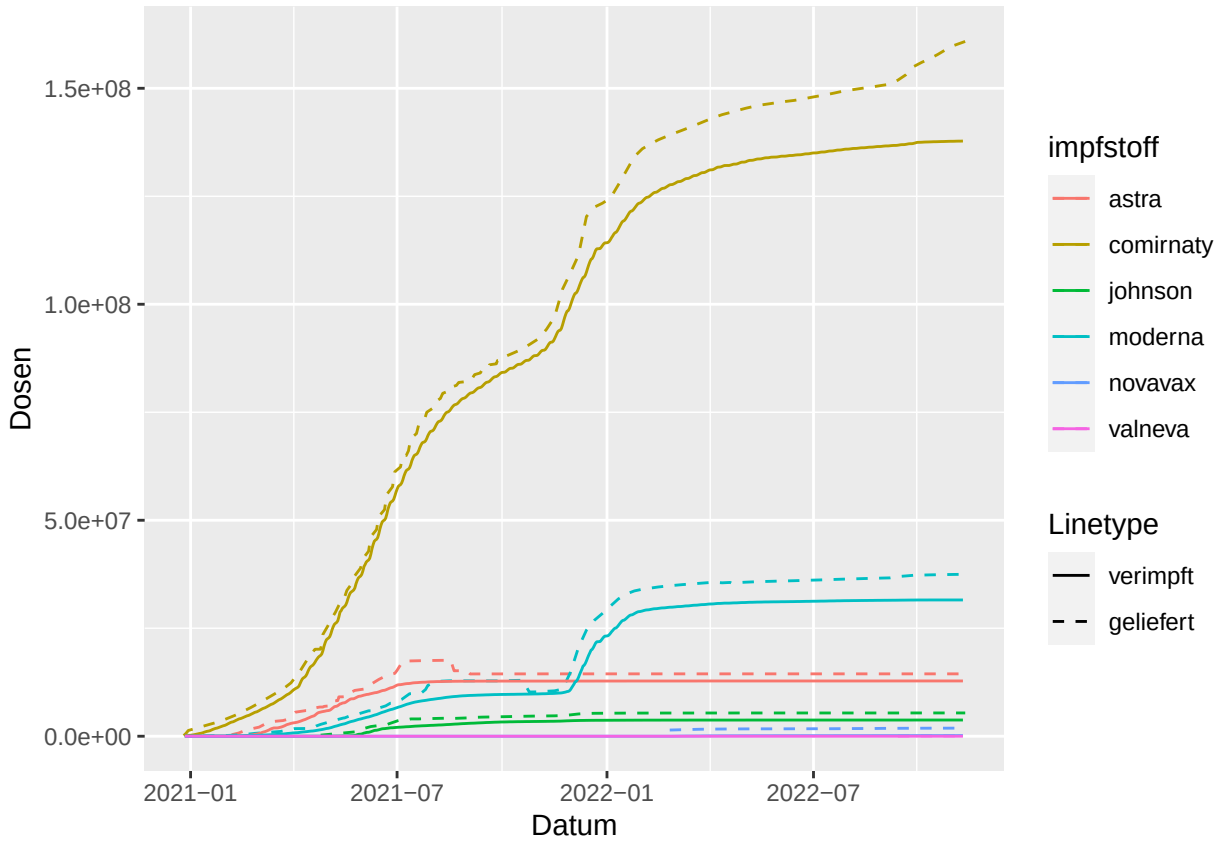
```
# Ihre Lösung:

deliv2 = data.frame(deliv |> group_by(impfstoff, date) |> summarise(dosen = sum(dosen)))
```

```
## `summarise()` has grouped output by 'impfstoff'. You can override using the
## `.groups` argument.
```

```
## gelieferte Dosen bis zum heutigen Tag vervollständigen
## - besonders für Astra relevant, da Lieferstopp
last = deliv2 |>
    group_by(impfstoff) |>
    slice_max('order_by'=date, n = 1)
last$date = Sys.Date()
deliv3 = rbind(deliv2, last)
```

```
## ende

ggplot() +
  geom_line(aes(x=as.Date(date), y=ave(dosen, impfstoff, FUN=cumsum) , group=impfstoff, color=impfstoff
            linetype='dashed', data=deliv3) +
  labs(x="Datum", y='Dosen') +
  geom_line(aes(x=as.Date(Datum, '%d.%m.%Y'), y=ave(Impfdosen.am.Tag, Hersteller, FUN=cumsum),
                group=Hersteller, color=Hersteller), data=vacc2)+

  geom_line(aes(x=Sys.Date(), y=0, linetype=factor(lt)), data=data.frame(lt=1:2)) +
scale_linetype_manual(values=c('solid', 'dashed'), name ='Linetype', labels=c('verimpft','geliefert'))
```

```
## geom_path: Each group consists of only one observation. Do you need to adjust
## the group aesthetic?
```




```
#+ scale_y_continuous(trans='sqrt')
##wurzelskala zieht die unteren Ergebnisse auseinander und macht es so ein bisschen anschaulicher
```