Abgabe Testat 2

Gruppe Nr. 1

Gruppenmitglieder

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

Veranstaltung:	DSR
Dozent:	Prof. Gumbel
Abgabedatum:	22.11.2026

Zentrale Grenzwertsatz

Zentrale Grenzwertsatz

Für jedes $n = 1, 2, \ldots$ seien die Zufallsvariablen $X_1, X_2, \ldots, X_n$ unabhängig und besitzen die gleiche Verteilung mit dem Erwartungswert $\mu = E(X_i)$ und der Varianz $\sigma^2 = Var(X_i)$. Dann gilt für die Verteilungsfunktion der standardisierten Summen $G_n = \frac{\sum_{i=1}^{n} X_i - n\mu}{\sqrt{n}\sigma}$:

$$\lim_{n\to\infty} P\left(G_n \leq x\right) = \Phi(x)$$

Dabei ist Φ die Verteilungsfunktion der Standard-Normalverteilung $N(0, 1)$.

Überprüfen Sie mit R den zentralen Grenzwertsatz.

Verteilung

Erzeugen Sie einen Data Frame, der die gezogenen Zufallszahlen G_n enthält. Die Stichprobe soll einen Umfang von 50000 haben (Anzahl Zeilen der Tabelle). Der Data Frame hat sechs Spalten, die die Werte für die Zahlen $n = 1, 2, 3, 10, 100, 1000$ enthalten. Geben Sie die ersten 10 Zeilen des Data Frames aus.

Welche Verteilung Sie für X_i nehmen, bleibt Ihnen überlassen. Es soll nur keine Normalverteilung sein.

```
# Erwartungswert der Chi-Quadrat Verteilung ist n (Anzahl der Freiheitsgrade).
# Standardabweichung ist \sqrt(2*n)
library(stats)
set.seed(1)
standardized_sum = function(vec, mu, sd){
  n = length(vec)
  (sum(vec)- n*mu) / (sqrt(n)*sd)
}
iter <- 1
list_n_1 <- list()
list_n_2 <- list()
list_n_3 <- list()
list_n_10 <- list()
list_n_100 <- list()
list_n_1000 <- list()
while (iter <= 50000){
  list_n_1[[iter]] <- rchisq(1,2)
  list_n_2[[iter]] <- rchisq(2,2)
  list_n_3[[iter]] <- rchisq(3,2)
  list_n_10[[iter]] <- rchisq(10,2)
  list_n_100[[iter]] <- rchisq(100,2)
  list_n_1000[[iter]] <- rchisq(1000,2)
  iter = iter + 1

}
chi_n_1  = sapply(list_n_1, standardized_sum, mu=2, sd=sqrt(2*2))
chi_n_2  = sapply(list_n_2, standardized_sum, mu=2, sd=sqrt(2*2))
```

```r
chi_n_3  = sapply(list_n_3, standardized_sum, mu=2, sd=sqrt(2*2))
chi_n_10  = sapply(list_n_10, standardized_sum, mu=2, sd=sqrt(2*2))
chi_n_100 = sapply(list_n_100, standardized_sum, mu=2, sd=sqrt(2*2))
chi_n_1000  = sapply(list_n_1000, standardized_sum, mu=2, sd=sqrt(2*2))


df = data.frame(chi_n_1,chi_n_2,chi_n_3,chi_n_10,chi_n_100,chi_n_1000)
head(df,n=10)
```

```
##       chi_n_1     chi_n_2     chi_n_3    chi_n_10   chi_n_100 chi_n_1000
## 1  -0.8448586  1.19282839  0.12532920 -1.26006741 -0.19350511  0.4415122
## 2  -0.6293512 -0.72408987  0.01530347 -0.96961016  0.02297818  1.5065334
## 3  -0.9793138  0.10522989  2.02770501  1.72259012 -0.91369892 -0.7437185
## 4  -0.7149816 -0.52377355 -0.76685819  0.66180366 -0.25252525  0.6859781
## 5   2.4376516 -1.17980219 -1.04677051 -0.63974014 -0.51230427  1.6671277
## 6   0.1190443  0.21464318  0.29801712  0.44085283 -0.73575882 -0.8684604
## 7   2.1183481 -0.01419154  1.67340914 -0.51935835 -0.19399917  0.6932579
## 8  -0.3350435 -0.85341453 -0.33333033 -1.00187868 -0.08948624 -0.3697799
## 9  -0.6611039 -1.35604313  0.71771453  0.02504821  0.22588269  1.5910629
## 10 -0.4521271  1.37780897 -0.69495649  0.94941848  0.14203376  0.5201319
```

Plot

Plotten Sie sechs Histogramme, die je für $n = 1, 2, 3, 10, 100, 1000$ die Verteilung im Vergleich zu einer $(0, 1)$-Normalverteilung zeigen. Die Intervallbreite soll 0,25 sein.

Tipp: Platzieren Sie die sechs Plots auf einem 3×2-Gitter.

```r
# Ihre Lösung:
library(tidyr)
library(dplyr)
```

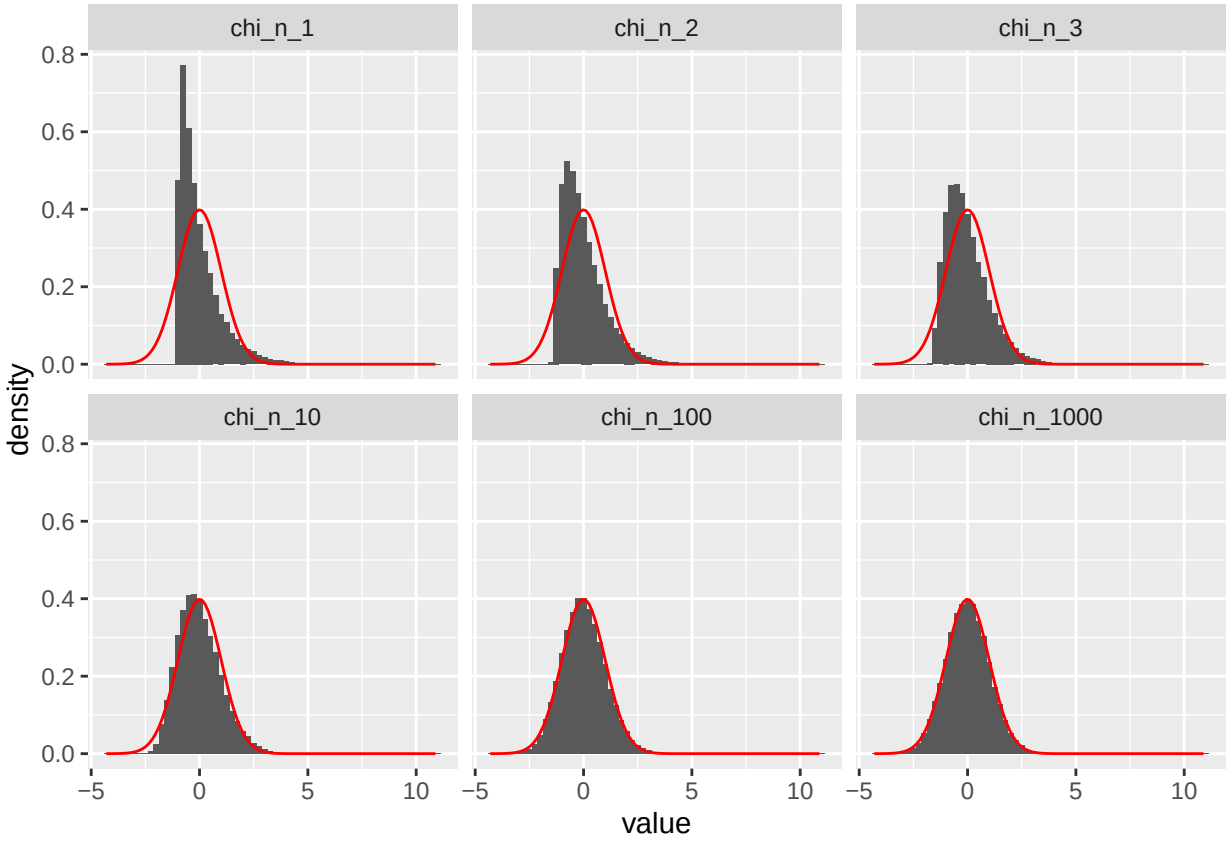
```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(ggplot2)

df_long = pivot_longer(df, cols=everything(), names_to="n", values_to="value")
df_long$n <- factor(df_long$n, levels= c("chi_n_1", "chi_n_2", "chi_n_3",
             "chi_n_10", "chi_n_100", "chi_n_1000"))
normal = dnorm(1)

#ggplot(normal, aes(x=value))+
 # geom_histogram(bins=50)

ggplot(df_long, aes(x=value))+
  geom_histogram(aes(y=after_stat(density)),binwidth=0.25)+
  # Histogramme müssen auf density gestellt werden,
  # damit Normalverteilung eingezeichnet werden kann
```

```
  facet_wrap(~n, ncol=3)+
  stat_function(fun=dnorm, color="red")
```



Abweichung

Nun soll die Abweichung der Standardisierten G von der Normalverteilung für alle n quantifiziert werden. Hierzu soll für alle Balken (bins) der Histogramme aus der vorigen Aufgabe die Differenz von G zur Normalverteilung gebildet und quadriert werden. Diese Werte werden aufaddiert und durch die Anzahl der Intervalle geteilt. Daraus wird die Wurzel gezogen.

Sie können sich auch ein anderes Maß zur Bestimmung der Abweichung überlegen.

Geben Sie die Abweichungen aus. Stimmt es, dass die Abweichungen mit größerem n kleiner werden?

Tipp: `hist(plot = FALSE)` erzeugt ein Histogramm, ohne es zu plotten. Gerne können Sie auch Ihr eigenes Histogramm nutzen.

```
sims <- list(
  "1" = chi_n_1,
  "2" = chi_n_2,
  "3" = chi_n_3,
  "10" = chi_n_10,
  "100" = chi_n_100,
  "1000" = chi_n_1000
)
diff_norm = function(G){
  # Intervallbreiten der vorigen Aufgabe generieren (0.25er Schritte)
  breaks <- seq(floor(min(G)), ceiling(max(G)) + 0.25, by = 0.25)
```

```r
  h <- hist(G, breaks = breaks, plot=FALSE)  # errechne Histogramm ohne zu plotten
  sqrt(mean((h$density- dnorm(h$mids))^2)) # berechnet die Abweichung
}

sapply(sims, diff_norm)
```

```
##           1           2           3          10         100        1000
## 0.110211417 0.064618054 0.051631678 0.024998170 0.007589740 0.003755681
```

Ja die Abweichungen werden mit größerem n kleiner.

Untersuchungen zur koronaren Herzkrankheit (t-tests)

Untersuchungen zur koronaren Herzkrankheit

In diesem Abschnitt sollen Daten von Probanden bzw. Patienten auf das Risiko für koronare Herzkrankheit untersucht werden. Dies ist eine Erkrankung der Herzkranzgefäße (Koronararterien), die sich durch Ablagerungen in den Gefäßwänden verengen. Der Original-Herz-Datensatz ist unter

- https://archive.ics.uci.edu/ml/datasets/Heart+Disease

beschrieben. Wir nutzen eine konsolidierte CSV-Datei, die bereits Header enthält. Download unter:

- https://oc.informatik.hs-mannheim.de/s/wyzFq34K9HiNjXR/download

Die Datei enthält 13 Merkmale, die einen Einfluss auf eine koronare Herzkrankheit haben können. Das 14. Merkmal `goal` (im Original auch `num`) ist die Diagnose (Klassifizierung). Der Wert ist 0, falls keine krankhafte Verengung der Gefäße vorliegt, oder 1, 2, 3 oder 4, falls – je nach Stärke – eine krankhafte Verengung der Gefäße vorliegt. Wir unterscheiden im Folgenden nur die Zustände "gesund" (0) und "krank" (1, 2, 3, 4). Unter

- https://archive.ics.uci.edu/ml/machine-learning-databases/heart-disease/heart-disease.names

finden Sie eine Beschreibung aller Attribute. Hier ist eine Zusammenfassung. Wir benötigen insbesondere die Merkmale `sex`, `trestbps`, `chol` und `goal`.

Feld	Bedeutung
age	age in years
sex	sex (1 = male; 0 = female)
cp	chest pain type (1 = typical angina; 2 = atypical angina; 3 = non-anginal pain; 4 = asymptomatic)
trestbps	resting systolic blood pressure (in mmHg on admission to the hospital)
chol	serum cholestoral in mg/dl
fbs	fasting blood sugar > 120 mg/dl (1 = true; 0 = false)
restecg	resting electrocardiographic results (0 = normal; 1 = having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV); 2 = showing probable or definite left ventricular hypertrophy by Estes' criteria)
thalach	maximum heart rate achieved
exang	exercise induced angina (1 = yes; 0 = no)
oldpeak	ST depression induced by exercise relative to rest.[1]
slope	slope of the peak exercise ST segment (1 = upsloping; 2 = flat; 3 = downsloping)
ca	number of major vessels (0-3) colored by flourosopy
thal	3 = normal; 6 = fixed defect; 7 = reversable defect
goal	diagnosis of heart disease (0: < 50% diameter narrowing ; 1, 2, 3, 4: > 50% diameter narrowing)

[1] ST depression refers to a finding on an electrocardiogram, wherein the trace in the ST segment is abnormally low below the baseline.

Einlesen der Herz-Daten

Lesen Sie die Datei aus der URL als Data Frame zur weiteren Bearbeitung ein. Überlegen Sie, ob sie Faktoren sinnvoll einsetzen können. Geben Sie die ersten drei Zeilen und fünf Spalten aus[2]:

```
heart_df = read.csv("https://oc.informatik.hs-mannheim.de/s/wyzFq34K9HiNjXR/download")
# Kategoriale Spalten in Faktoren
# Geschlecht in Faktoren mit sprechendem Namen umwandeln
heart_df$sex <- factor(heart_df$sex, levels=c(1,0), labels=c("male", "female"))
heart_df$cp = as.factor(heart_df$cp)
heart_df$restecg = as.factor(heart_df$restecg)
heart_df$exang = as.factor(heart_df$exang)
heart_df$slope = as.factor(heart_df$slope)
heart_df$thal = as.factor(heart_df$thal)
heart_df[1:3,1:5] # erste 3 Zeilen und 5 Spalten
```

```
##   age  sex cp trestbps chol
## 1  63 male  1      145  233
## 2  67 male  4      160  286
## 3  67 male  4      120  229
```

Cholesterin im Vergleich Männer/Frauen

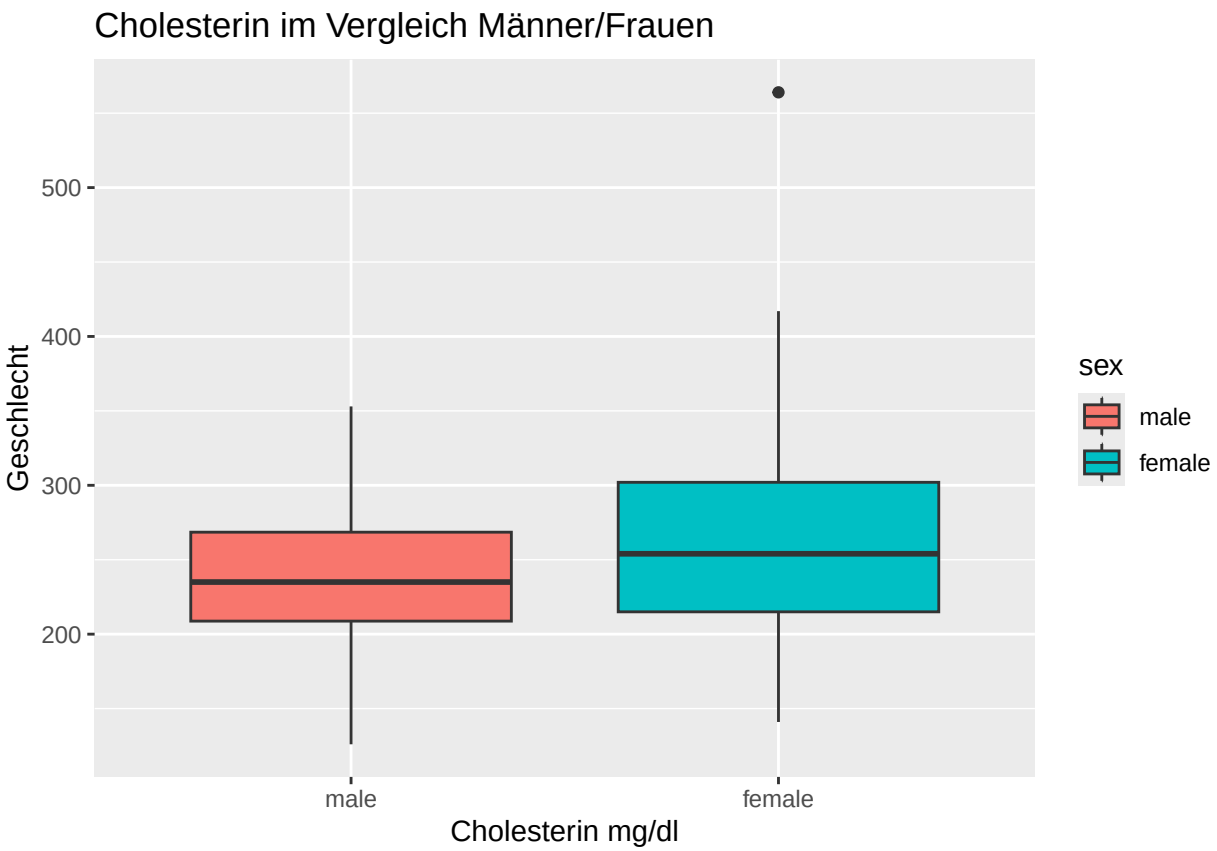
Nun sollen die Cholesterin-Werte untersucht werden – zunächst im Vergleich Männer zu Frauen.

Überblick über Cholesterin-Daten

Verschaffen Sie sich einen Überblick, indem Sie ein Boxplot für das Cholesterin gruppiert nach dem Geschlecht plotten.

```
ggplot(heart_df)+
  geom_boxplot(aes(x=sex, y=chol, fill=sex))+
  labs(title="Cholesterin im Vergleich Männer/Frauen", x="Cholesterin mg/dl", y="Geschlecht")
```

[2] Möglicherweise kommt es zu einem Fehler beim Einlesen des ersten Attributs (`age`). Manuelles Umbenennen hilft.



Konfidenz-Intervall

Berechnen Sie das Konfidenz-Intervall (Niveau 95%) für den Cholesterin-Level jeweils für Männer und Frauen.

Tabelle Geben Sie das Ergebnis als `kable`-Tabelle aus:

```
library(kableExtra)
# Ausführen der beiden zweiseitigen t-tests getrennt nach Geschlecht
test_male = t.test(heart_df$chol[heart_df$sex=="male"])
test_female = t.test(heart_df$chol[heart_df$sex=="female"])
# Zusammenbauen der Tabelle:
result <- data.frame(Geschlecht=c("Mann", "Frau"),
                     Mittelwert=c(test_male$estimate, test_female$estimate),
                     Untere_Grenze=c(test_male$conf.int[1],test_female$conf.int[1]),
                     Obere_Grenze=c(test_male$conf.int[2],test_female$conf.int[2]))

kable(result, digits=2, caption="Konfidenzintervalle für Cholesterin nach Geschlecht") |>
  kable_styling(full_width = FALSE) # Für schönere Darstellung der Tabelle
```

Table 1: Konfidenzintervalle für Cholesterin nach Geschlecht

Geschlecht	Mittelwert	Untere_Grenze	Obere_Grenze
Mann	239.60	233.74	245.46
Frau	261.75	248.67	274.83

Überlappung? Überlappen sich die Bereiche? -> Nein

Cholesterin im Vergleich zur Erkrankung

Nun sollen die Cholesterin-Werte in Abhängigkeit der Diagnose untersucht werden.

Überblick über Cholesterin-Daten

Verschaffen Sie sich einen Überblick, indem Sie einen Boxplot für das Cholesterin gruppiert nach der Diagnose plotten.

```
library(dplyr)
```

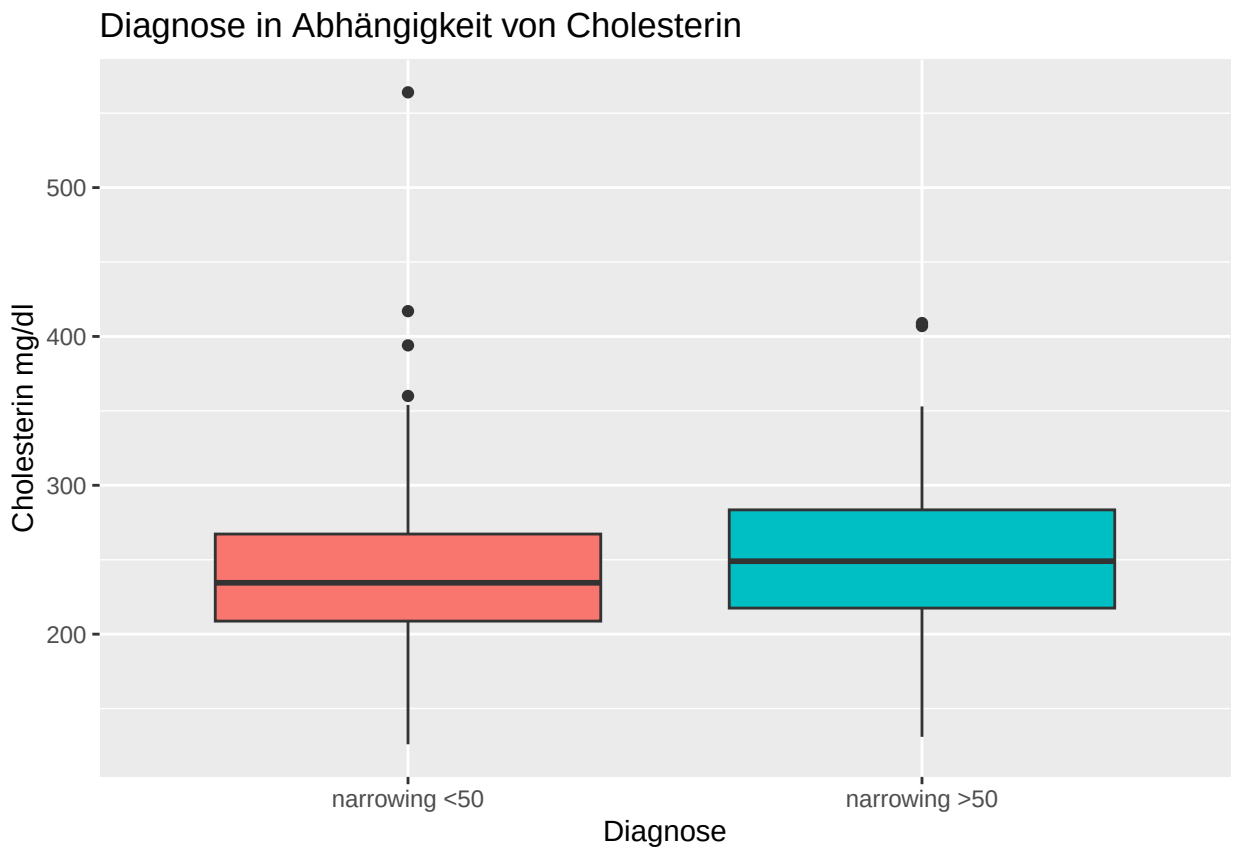
```
##
## Attaching package: 'dplyr'
```

```
## The following object is masked from 'package:kableExtra':
##
##     group_rows
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
# Narrowing größer 50 steht für die kranken Patienten, wohingegen eine Verengung
# kleiner 50 die Gesunden repräsentiert.
# Im Datensatz ist alles größer 0 krank
heart_df <- mutate(heart_df,diagnosis=ifelse(goal>0,"narrowing >50","narrowing <50"))

ggplot(heart_df)+
  geom_boxplot(aes(x=diagnosis, fill=diagnosis,y=chol))+
  labs(title="Diagnose in Abhängigkeit von Cholesterin",
       x="Diagnose",
       y="Cholesterin mg/dl") +
  theme(legend.position="none")
```



Konfidenz-Intervall

Berechnen Sie die Konfidenzintervalle für beide Gruppen und geben Sie das Ergebnis als `kable`-Tabelle aus:

```
narrow_greater_50 = heart_df |> filter(diagnosis=="narrowing >50")
narrow_less_50 = heart_df |> filter(diagnosis=="narrowing <50")
test_greater_50 = t.test(narrow_greater_50$chol,conf.level=0.95)
test_less_50 = t.test(narrow_less_50$chol,conf.level=0.95)
result <- data.frame(Gruppe=c("Verengung >50","Verengung <50"),
                     "Mittelwert_der_Stichprobe"=c(test_greater_50$estimate,
                                                   test_less_50$estimate),
                     Untere_Grenze=c(test_greater_50$conf.int[1],test_less_50$conf.int[1]),
                     Obere_Grenze=c(test_greater_50$conf.int[2],test_less_50$conf.int[2]))

kable(result) |> kable_styling()
```

Gruppe	Mittelwert_der_Stichprobe	Untere_Grenze	Obere_Grenze
Verengung >50	251.4748	243.1752	259.7744
Verengung <50	242.6402	234.3977	250.8828

```
# result <- heart_df |>
# group_by(Gruppe = diagnosis) |>
# summarise(test = list(t.test(chol)),
# Mittelwert = mean(chol),
# Untere_Grenze=test[[1]]$conf.int[1],
```

```
#   Obere_Grenze=test[[1]]$conf.int[2]) |>
# select(-test)
#kable(result, digits=2) |> kable_styling()
```

Test

Es sieht so aus, als ob der Cholesterin-Wert bei den erkrankten Patienten höher ist als bei den nicht erkrankten. Überprüfen Sie das mit einem Hypothesen-Test.

Wie lauten die Hypothesen? Formulieren Sie die Hypothesen (H_0 und H_1).

H_0: Die Cholesterin-Werte der erkrankten Patienten sind niedriger oder gleich der Gesunden.

H_1: Die Cholesterin-Werte der erkrankten Patienten sind höher als die der nicht erkrankten

Testanwendung Wenden Sie den Test mit R an. Was ist das Ergebnis?

```
t.test(narrow_greater_50$chol, narrow_less_50$chol, alternative = "greater")
```

```
##
##  Welch Two Sample t-test
##
## data:  narrow_greater_50$chol and narrow_less_50$chol
## t = 1.4924, df = 298.64, p-value = 0.06832
## alternative hypothesis: true difference in means is greater than 0
## 95 percent confidence interval:
##  -0.9327489        Inf
## sample estimates:
## mean of x mean of y
##  251.4748  242.6402
```

```
# Dieser Test prüft ob der Mittelwert der ersten Gruppe (krank) höher als der der Zweiten ist
```

Da der P-Wert größer 0.05 bzw 5% ist, kann die Nullhypothese nicht verworfen und die Alternativhypothese nicht angenommen werden.

Grund dafür könnten die Ausreißer in der gesunden Gruppe sein. Durch sie wird der Mittelwertsunterschied kleiner.

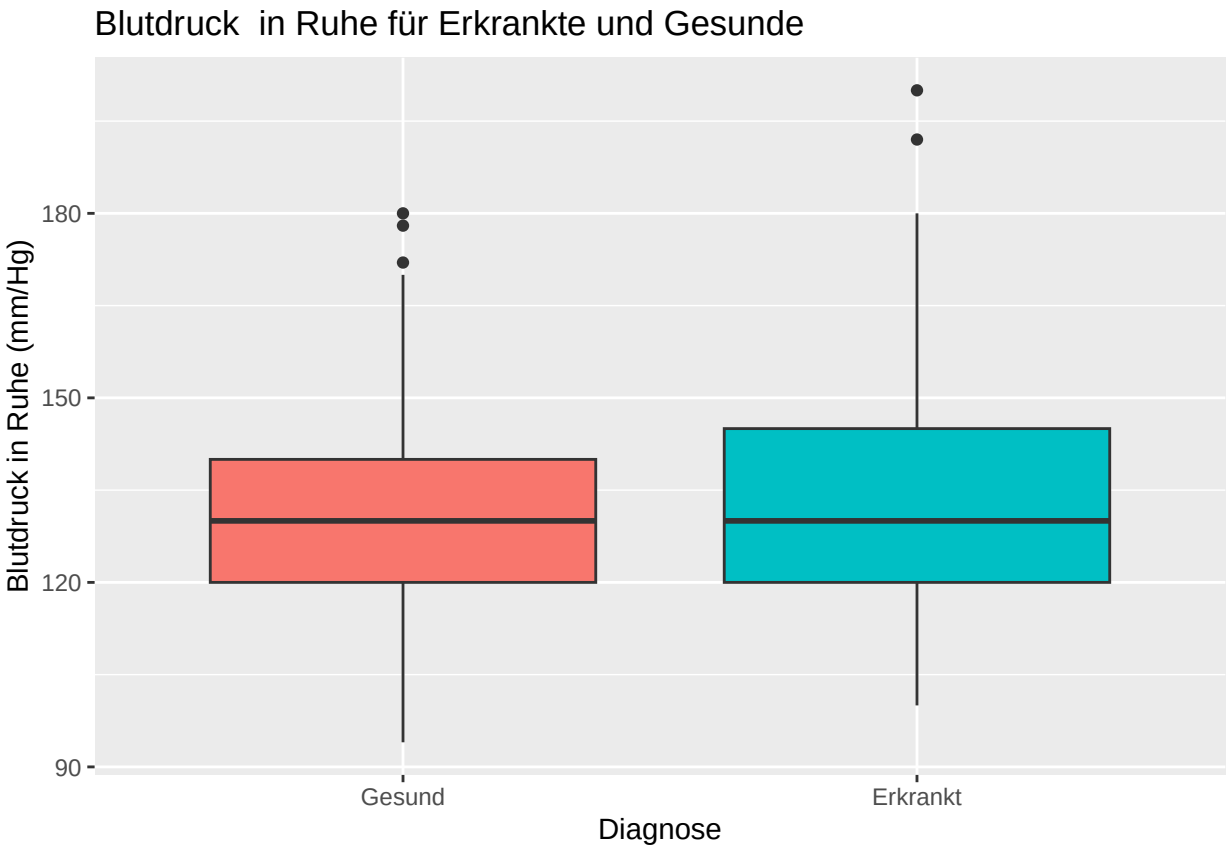
Systolischer Ruheblutdruck

Der systolische Blutdruck liegt beim gesunden Menschen bei ca. 120 mmHg.

Überblick über Blutdruck

Plot Verschaffen Sie sich einen Überblick, indem Sie einen Boxplot für den Blutdruck in Ruhe gruppiert nach der Diagnose plotten.

```
ggplot(heart_df) +
  geom_boxplot(aes(y=trestbps, x=diagnosis ,fill = diagnosis)) +
  labs(title="Blutdruck  in Ruhe für Erkrankte und Gesunde",
       x="Diagnose",
       y="Blutdruck in Ruhe (mm/Hg)")+
  scale_x_discrete(labels = c("narrowing <50" = "Gesund",
                              "narrowing >50" = "Erkrankt"))+
  theme(legend.position = "none") # entfernt die Legende
```



Normalverteilt? Kann überhaupt davon ausgegangen werden, dass die Daten normalverteilt sind? Da der Blutdruck von vielen Faktoren abhängt, könnte angenommen werden er sei normalverteilt. Denn viele voneinander unabhängige Einflüsse *addieren* sich (zentraler Grenzwertsatz). Aber Blutdruck-Werte sind auch nach unten begrenzt und nach oben ist viel Platz. Das kann für eine rechtsschiefe Verteilung sorgen.

Konfidenzintervalle nach Erkrankung

Berechnen Sie die Konfidenzintervalle für den Ruheblutdruck aufgeschlüsselt nach der Diagnose (erkrankt/nicht erkrankt) und geben Sie das Ergebnis als `kable`-Tabelle aus:

```
result_bp = heart_df |>
  group_by(Gruppe = diagnosis) |>
  summarise(test = list(t.test(trestbps)),
            Mittelwert = mean(trestbps),
            Untere_Grenze = test[[1]]$conf.int[1],
            Obere_Grenze = test[[1]]$conf.int[2]) |>
  select(-test) # Das sorgt dafür, dass die zuvor erstellte Spalte Test,
# von der die anderen Spalten abgeleitet wurden, wieder entfernt wird

kable(result_bp, digits=2) |> kable_styling()
```

Gruppe	Mittelwert	Untere_Grenze	Obere_Grenze
narrowing <50	129.25	126.75	131.75
narrowing >50	134.57	131.42	137.72

Test, ob Kranke höheren Ruhe-Blutdruck haben

Überprüfen Sie mit einem Hypothesen-Test, ob Erkrankte einen höheren Ruhe-Blutdruck haben als gesunde Probanden.

Wie lauten die Hypothesen? Formulieren Sie die Hypothesen (H_0 und H_1).

H_0: Gesunde haben einen gleich großen oder höheren Ruhe-Blutdruck als Erkrankte. $\mu_{krank} \leq \mu_{gesund}$

H_1: Erkrankte haben einen höheren Ruhe-Blutdruck als Gesunde. $\mu_{krank} > \mu_{gesund}$

Testanwendung Wenden Sie den Test mit R an. Was ist das Ergebnis?

```r
# Testet die auf die Alternativhypothese, dass das erste Argument größer als das Zweite ist
restbps_diff = t.test(narrow_greater_50$trestbps,
                      narrow_less_50$trestbps,
                      alternative = "greater")
print(restbps_diff)
```

```
##
##  Welch Two Sample t-test
##
## data:  narrow_greater_50$trestbps and narrow_less_50$trestbps
## t = 2.6152, df = 274.64, p-value = 0.004705
## alternative hypothesis: true difference in means is greater than 0
## 95 percent confidence interval:
##  1.962045      Inf
## sample estimates:
## mean of x mean of y
##  134.5683  129.2500
```

Der P-Wert ist mit 0.0047 niedriger als 0.05. Somit kann die Alternativhypothese angenommen werden und die Nullhypothese abgelehnt werden. Erkrankte haben also einen höheren Ruhe-Blutdruck als Gesunde.

Unüberwachtes Lernen mit dem Herz-Datensatz

Clustering und PCA auf die Herzdaten

Einlesen der Herz-Daten

Es werden wieder die Herzdaten aus der letzten Aufgabe genutzt. Lesen Sie diese als Data Frame ein.

```
# Ihre Lösung:
df = read.csv("https://oc.informatik.hs-mannheim.de/s/wyzFq34K9HiNjXR/download", header= TRUE, na.strin
kable(df[1:10 , ])
```

age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	goal
63	1	1	145	233	1	2	150	0	2.3	3	0	6	0
67	1	4	160	286	0	2	108	1	1.5	2	3	3	2
67	1	4	120	229	0	2	129	1	2.6	2	2	7	1
37	1	3	130	250	0	0	187	0	3.5	3	0	3	0
41	0	2	130	204	0	2	172	0	1.4	1	0	3	0
56	1	2	120	236	0	0	178	0	0.8	1	0	3	0
62	0	4	140	268	0	2	160	0	3.6	3	2	3	3
57	0	4	120	354	0	0	163	1	0.6	1	0	3	0
63	1	4	130	254	0	2	147	0	1.4	2	1	7	2
53	1	4	140	203	1	2	155	1	3.1	3	0	7	1

Bedeutet "ähnliche Merkmale" auch "gleiche Diagnose"?

Für jeden Datensatz ist bekannt, zu welcher Klasse er gehört: 0 (gesund) und 1 (erkrankt). Wir wollen untersuchen, wie gut *ähnliche* Datensätze zur gleichen Klasse gehören. Dafür soll mit dem k-means-Clusterverfahren der Datensatz in zwei Cluster eingeteilt werden.

Nur reelle Merkmale

Zunächst sollen **nur die nummerischen Merkmale** benutzt werden und nicht jene, die Faktoren sind.

Clustering Clustern Sie diese Daten. Überlegen Sie, ob Sie die Daten standardisieren wollen.

```
# Ihre Lösung:

df_cleaned = na.omit(df) # Zeilen mit NA Werten entfernen

str(df)
```

```
## 'data.frame':    303 obs. of  14 variables:
##  $ age     : int  63 67 67 37 41 56 62 57 63 53 ...
##  $ sex     : int  1 1 1 1 0 1 0 0 1 1 ...
##  $ cp      : int  1 4 4 3 2 2 4 4 4 4 ...
##  $ trestbps: int  145 160 120 130 130 120 140 120 130 140 ...
##  $ chol    : int  233 286 229 250 204 236 268 354 254 203 ...
##  $ fbs     : int  1 0 0 0 0 0 0 0 0 1 ...
```

```
##  $ restecg : int  2 2 2 0 2 0 2 0 2 2 ...
##  $ thalach : int  150 108 129 187 172 178 160 163 147 155 ...
##  $ exang   : int  0 1 1 0 0 0 0 1 0 1 ...
##  $ oldpeak : num  2.3 1.5 2.6 3.5 1.4 0.8 3.6 0.6 1.4 3.1 ...
##  $ slope   : int  3 2 2 3 1 1 3 1 2 3 ...
##  $ ca      : int  0 3 2 0 0 0 2 0 1 0 ...
##  $ thal    : int  6 3 7 3 3 3 3 3 7 7 ...
##  $ goal    : int  0 2 1 0 0 0 3 0 2 1 ...
```

```r
df_cleaned$restecg = factor(df_cleaned$restecg)
df_cleaned$fbs = factor(df_cleaned$fbs)
df_cleaned$exang = factor(df_cleaned$exang)
df_cleaned$cp = factor(df_cleaned$cp)
df_cleaned$thal = factor(df_cleaned$thal)
df_cleaned$sex = factor(df_cleaned$sex)
df_cleaned$slope = factor(df_cleaned$slope) # the slop of the peak exersice ST segment
df_cleaned$ca = factor(df_cleaned$ca)
df_cleaned$goal = factor((df_cleaned$goal))

str(df_cleaned)
```

```
## 'data.frame':    297 obs. of  14 variables:
##  $ age     : int  63 67 67 37 41 56 62 57 63 53 ...
##  $ sex     : Factor w/ 2 levels "0","1": 2 2 2 2 1 2 1 1 2 2 ...
##  $ cp      : Factor w/ 4 levels "1","2","3","4": 1 4 4 3 2 2 4 4 4 4 ...
##  $ trestbps: int  145 160 120 130 130 120 140 120 130 140 ...
##  $ chol    : int  233 286 229 250 204 236 268 354 254 203 ...
##  $ fbs     : Factor w/ 2 levels "0","1": 2 1 1 1 1 1 1 1 1 2 ...
##  $ restecg : Factor w/ 3 levels "0","1","2": 3 3 3 1 3 1 3 1 3 3 ...
##  $ thalach : int  150 108 129 187 172 178 160 163 147 155 ...
##  $ exang   : Factor w/ 2 levels "0","1": 1 2 2 1 1 1 1 2 1 2 ...
##  $ oldpeak : num  2.3 1.5 2.6 3.5 1.4 0.8 3.6 0.6 1.4 3.1 ...
##  $ slope   : Factor w/ 3 levels "1","2","3": 3 2 2 3 1 1 3 1 2 3 ...
##  $ ca      : Factor w/ 4 levels "0","1","2","3": 1 4 3 1 1 1 3 1 2 1 ...
##  $ thal    : Factor w/ 3 levels "3","6","7": 2 1 3 1 1 1 1 1 3 3 ...
##  $ goal    : Factor w/ 5 levels "0","1","2","3",..: 1 3 2 1 1 1 4 1 3 2 ...
##  - attr(*, "na.action")= 'omit' Named int [1:6] 88 167 193 267 288 303
##   ..- attr(*, "names")= chr [1:6] "88" "167" "193" "267" ...
```

```r
# Ihre Lösung:


# neue disease Spalte erzeugen
# wenn goal == 0, dann wird neue Spalte auch 0 (gesund)
# wenn nicht, dann 1 (krank)
df_cleaned$disease = ifelse(df_cleaned$goal == 0, 0, 1)
df_cleaned$disease = factor(df_cleaned$disease)

# Spalte als Ground Truth speichern
gt = df_cleaned$disease


# für jede Spalte den Wert TRUE oder FALSE zurückgeben
# Namen der Spalten mit TRUE zurückgeben
numeric_cols = names(df_cleaned)[vapply(df_cleaned, is.numeric, logical(1))]
```

```r
# Daten standardisieren (Mittelwert 0, Standardabweichung 1)
# Spalten haben verschiedene Wertebereiche
# größere Zahlenwerte sollen aber nicht größeren Einfluss haben
df_numeric = data.frame(scale(df_cleaned[, numeric_cols, drop= FALSE]))

kable(df_numeric[1:10, ])
```

age	trestbps	chol	thalach	oldpeak
0.9346033	0.7491157	-0.2759776	0.0174650	1.0671642
1.3766051	1.5935769	0.7433005	-1.8132735	0.3811301
1.3766051	-0.6583195	-0.3529043	-0.8979043	1.3244270
-1.9384088	-0.0953454	0.0509607	1.6302584	2.0962153
-1.4964070	-0.0953454	-0.8336958	0.9764232	0.2953758
0.1611000	-0.6583195	-0.2182826	1.2379573	-0.2191498
0.8241028	0.4676287	0.3971306	0.4533551	2.1819696
0.2716005	-0.6583195	2.0510537	0.5841221	-0.3906583
0.9346033	-0.0953454	0.1278873	-0.1133021	0.2953758
-0.1704014	0.4676287	-0.8529275	0.2354100	1.7531983

```r
# Clustern auf zwei Cluster mit 20 Startcluster
km = kmeans(df_numeric, centers=2, nstart=20)
cluster_num = km$cluster
kable(cluster_num[1:30])
```

x
1
1
1
2
2
2
1
2
1
1
2
1
2
2
2
2
2
2
2
2
2
1
1
1
1
1

x
2
2
1
2
2

Richtig? Berechnen Sie, wie viel Prozent der Datensätze richtig einem Cluster eingeordnet wurden und geben Sie die Zahl auf zwei Nachkommastellen gerundet aus.

Hinweis: Berücksichtigen Sie, dass die Vergabe der Clusternummern zufällig ist. D.h. sowohl die Cluster (1, 2) wie auch (2, 1) sind möglich.

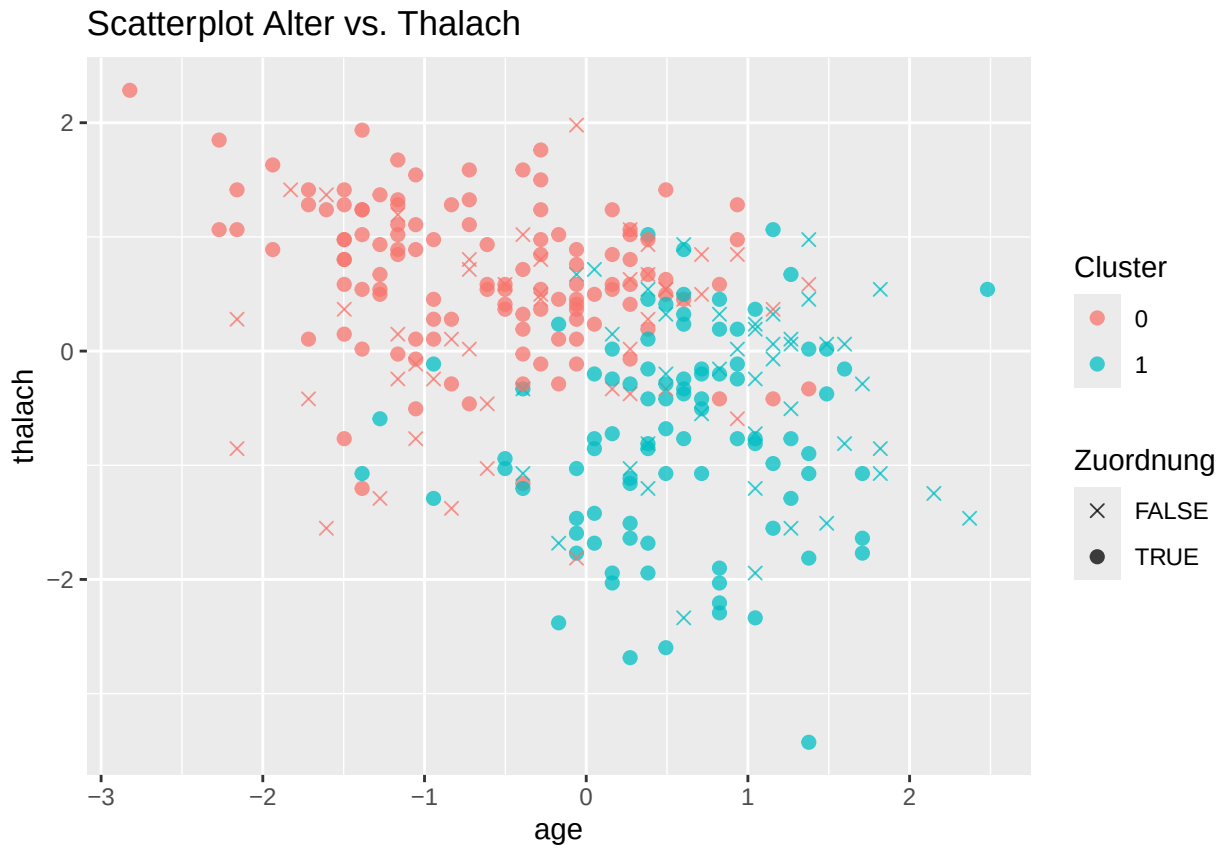
```
# Ihre Lösung:

# Clusternummern können vertauscht sein, deswegen beide Zuordnungen prüfen
match_clusters = function(cluster, gt) {
  gt = as.character(gt)

  # Beide Varianten prüfen
  # (cluster == 1) == (gt == "0") gibt boolsche Werte als Vektor zurück
  # mean berechnet den Anteil der TRUE Werte (Mittelwert von TRUE=1 und FALSE=0)

  acc_1_to_0 = mean((cluster == 1) == (gt == "0"))
  acc_1_to_1 = mean((cluster == 1) == (gt == "1"))


  # Klassenlabel (0, 1) richtig zuordnen
  if (acc_1_to_0 >= acc_1_to_1) {
    predicted = ifelse(cluster == 1, "0", "1")
  } else {
    predicted = ifelse(cluster == 1, "1", "0")
  }

  list(
    accuracy = max(acc_1_to_0, acc_1_to_1), # höhere Accuracy wählen
    predicted = predicted
  )
}

result_num = match_clusters(cluster_num, df_cleaned$disease)
correct_num = result_num$predicted == gt # TRUE FALSE Werte

sprintf("Richtig zugeordnet: %.2f%%", 100 * result_num$accuracy)
```

```
## [1] "Richtig zugeordnet: 70.03%"
```

Scatterplot age vs. thalach Plotten Sie die Merkmale `age` und `thalach` als Scatterplot. Färben Sie die Punkte gemäß ihrer Clusterzuordnung ein. Die Form (`shape`) eines Punkts soll zeigen, ob die Klassifikation (d.h. der Cluster) richtig oder falsch ist.

```
# Ihre Lösung:

# Dataframe mit allen Informationen für Plot erstellen
```

```r
plot_num = data.frame(
  age = df_numeric$age,
  thalach = df_numeric$thalach, # maximum heart rate
  cluster = factor(result_num$predicted), # Vorhersagen
  correct = factor(correct_num, levels = c(FALSE, TRUE)) # gt
)

ggplot(plot_num, aes(x = age, y = thalach,
                     color = cluster, shape = correct)) +
  geom_point(alpha = 0.75, size = 2) +
  scale_shape_manual(
    values = c("FALSE" = 4, "TRUE" = 19), # Formen festlegen
  ) +
  labs(title = "Scatterplot Alter vs. Thalach", color = "Cluster", shape = "Zuordnung", x = "age", y =
```



Mit Dummy-Variablen

Nun sollen **alle Merkmale** benutzt werden.

Clustering Clustern Sie diese Daten. Überlegen Sie, wie die Faktoren zu Zahlen werden.

```r
# Ihre Lösung:

# Vektor mit allen Merkmalen
target_cols = c("age","sex","cp","trestbps","chol","fbs","restecg","thalach",
                "exang", "oldpeak", "slope", "ca", "thal")
```

```r
# Spalten aus Dataframe auswählen
df_all = df_cleaned[, target_cols]

# nimmt alle Spalten des df
# erzeugt für alle Faktorenstufen eine eigene Spalte
# zwei Stufen = eine Spalte
mm = model.matrix(~ . -1, data=df_all)
kable(mm[1:10,])
```

age	sex0	sex1	cp2	cp3	cp4	trestbps	chol	fbs1	restecg1	restecg2	thalach	exang1	oldpeak	slope2	slope3	ca1	ca2	ca3	thal6	thal7
63	0	1	0	0	0	145	233	1	0	1	150	0	2.3	0	1	0	0	0	1	0
67	0	1	0	0	1	160	286	0	0	1	108	1	1.5	1	0	0	0	1	0	0
67	0	1	0	0	1	120	229	0	0	1	129	1	2.6	1	0	0	1	0	0	1
37	0	1	0	1	0	130	250	0	0	0	187	0	3.5	0	1	0	0	0	0	0
41	1	0	1	0	0	130	204	0	0	1	172	0	1.4	0	0	0	0	0	0	0
56	0	1	1	0	0	120	236	0	0	0	178	0	0.8	0	0	0	0	0	0	0
62	1	0	0	0	1	140	268	0	0	1	160	0	3.6	0	1	0	1	0	0	0
57	1	0	0	0	1	120	354	0	0	0	163	1	0.6	0	0	0	0	0	0	0
63	0	1	0	0	1	130	254	0	0	1	147	0	1.4	1	0	1	0	0	0	1
53	0	1	0	0	1	140	203	1	0	1	155	1	3.1	0	1	0	0	0	0	1

```r
mm_scaled = scale(mm)

# Clustering mit allen Merkmalen
km_all = kmeans(mm_scaled, centers = 2, nstart = 50)
```

Richtig? Berechnen Sie für diesen Fall, wie viel Prozent der Datensätze richtig einem Cluster eingeordnet wurden und geben Sie die Zahl auf zwei Nachkommastellen gerundet aus. Wie hat sich der Wert verändert? Warum ist dies so?

```r
# Ihre Lösung:

cluster_all = km_all$cluster

result_all = match_clusters(cluster_all, gt)
correct_all = result_all$predicted == gt

sprintf("Richtig zugeordnet: %.2f%%", 100 * mean(correct_all))
```

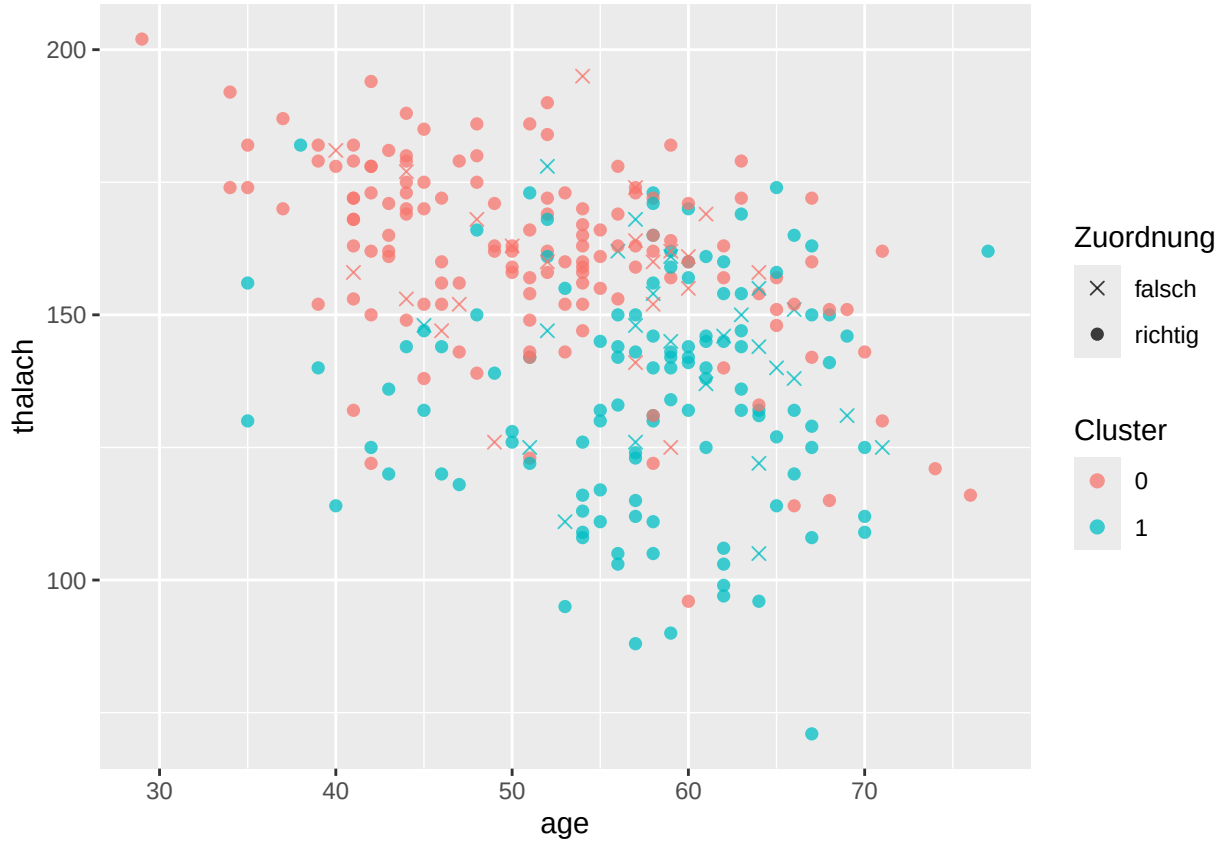
```
## [1] "Richtig zugeordnet: 84.51%"
```

Wie hat sich der Wert verändert und Warum? ANTWORT Der Wert hat sich deutlich verbessert, weil mehr Merkmale in die Berechnung eingeflossen sind und somit auch mehr Informationen. Faktoren wie das Geschlecht können wertvolle Informationen und Abhändigkeiten enthalten, die das Cluster Ergebniss verbessern.

Scatterplot age vs. thalach Plotten Sie erneut und schauen Sie, wie die richtigen nun Punkte verteilt sind.

```r
# Ihre Lösung:

plot_all = data.frame(
```

```
  age = df_all$age,
  thalach = df_all$thalach,
  cluster = factor(result_all$predicted), # Vorhersagen
  correct = factor(correct_all, levels = c(FALSE, TRUE)) # gt
)

ggplot(plot_all, aes(x = age, y = thalach,
                     color = cluster, shape = correct)) +
  geom_point(alpha = 0.75, size = 2) +
  scale_shape_manual(
    values = c("FALSE" = 4, "TRUE" = 16),
    labels = c("FALSE" = "falsch", "TRUE" = "richtig")
  ) +
  labs(color = "Cluster", shape = "Zuordnung", x = "age", y = "thalach")
```



PCA

Wenden Sie eine PCA auf diesen Datensatz an. Es sollen alle Merkmale berücksichtigt werden.

Wichtige Merkmale

Welche Merkmale der ersten Hauptkomponente tragen am meisten zur Varianz bei? Geben Sie die TOP-10-Merkmale an.

```
# Ihre Lösung:
```

```r
# Daten sind bereits zentriert und standardisiert
pca = prcomp(mm_scaled, center = FALSE, scale. = FALSE)

kable(pca$rotation[, 1])
```

	x
age	-0.2214695
sex0	0.1897896
sex1	-0.1897896
cp2	0.2496325
cp3	0.2146476
cp4	-0.3789151
trestbps	-0.1226259
chol	-0.0439295
fbs1	-0.0324478
restecg1	-0.0648217
restecg2	-0.1197593
thalach	0.3601178
exang1	-0.3387436
oldpeak	-0.3422748
slope2	-0.2640842
slope3	-0.0849604
ca1	-0.0898730
ca2	-0.1493219
ca3	-0.1394602
thal6	-0.1100339
thal7	-0.2998447

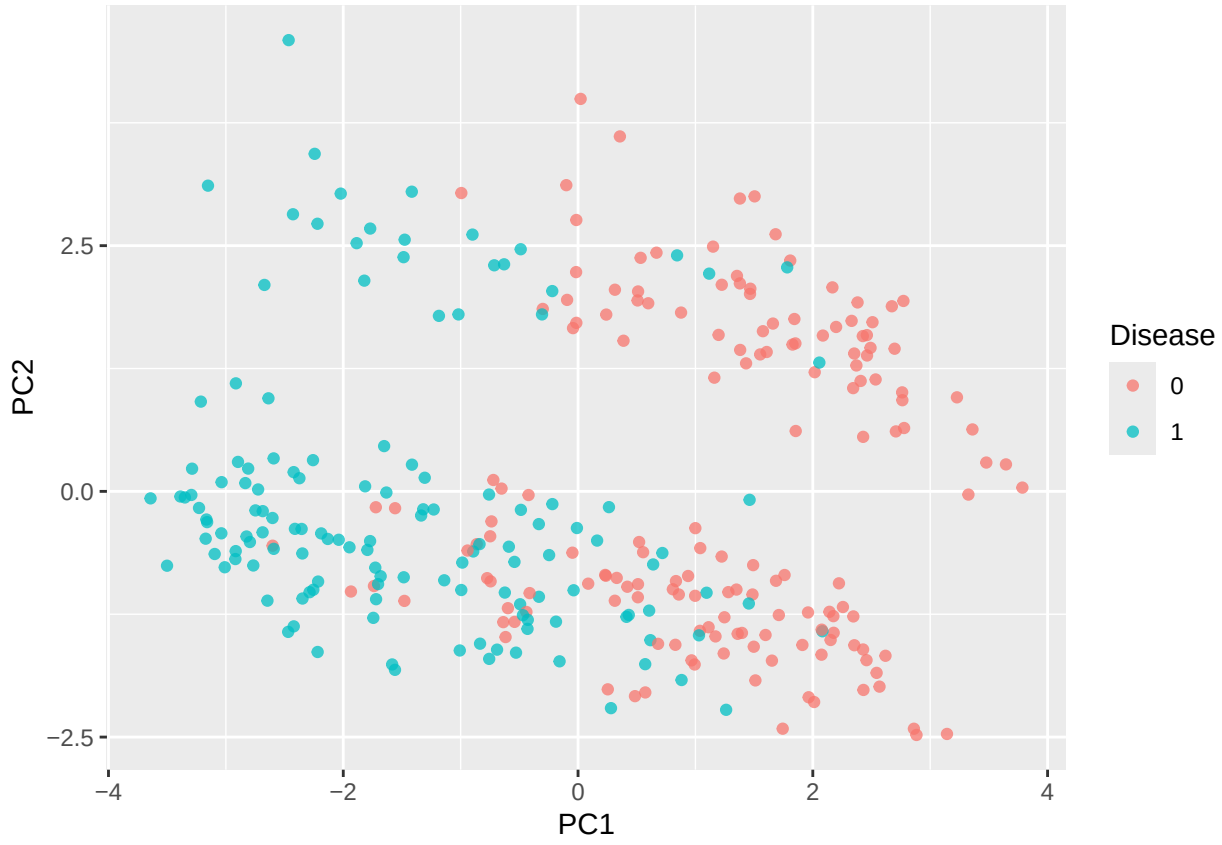
```r
# Beitrag der einzelnen Merkmale zur Varianz
# Aus Rotationsmatrix die Loadings (Eigenvektoren) der ersten Hauptkomponente quadrieren zurückgeben un
contribution_pc1 = pca$rotation[, 1]^2
top10 = sort(contribution_pc1, decreasing = TRUE)[1:10]

kable(top10)
```

	x
cp4	0.1435767
thalach	0.1296849
oldpeak	0.1171520
exang1	0.1147472
thal7	0.0899068
slope2	0.0697405
cp2	0.0623164
age	0.0490488
cp3	0.0460736
sex1	0.0360201

Erste und zweite Hauptkomponente

Plotten Sie die erste und zweite Hauptkomponente als Scatterplot. Färben Sie die Punkte gemäß ihrer Klasse (Disease) ein.

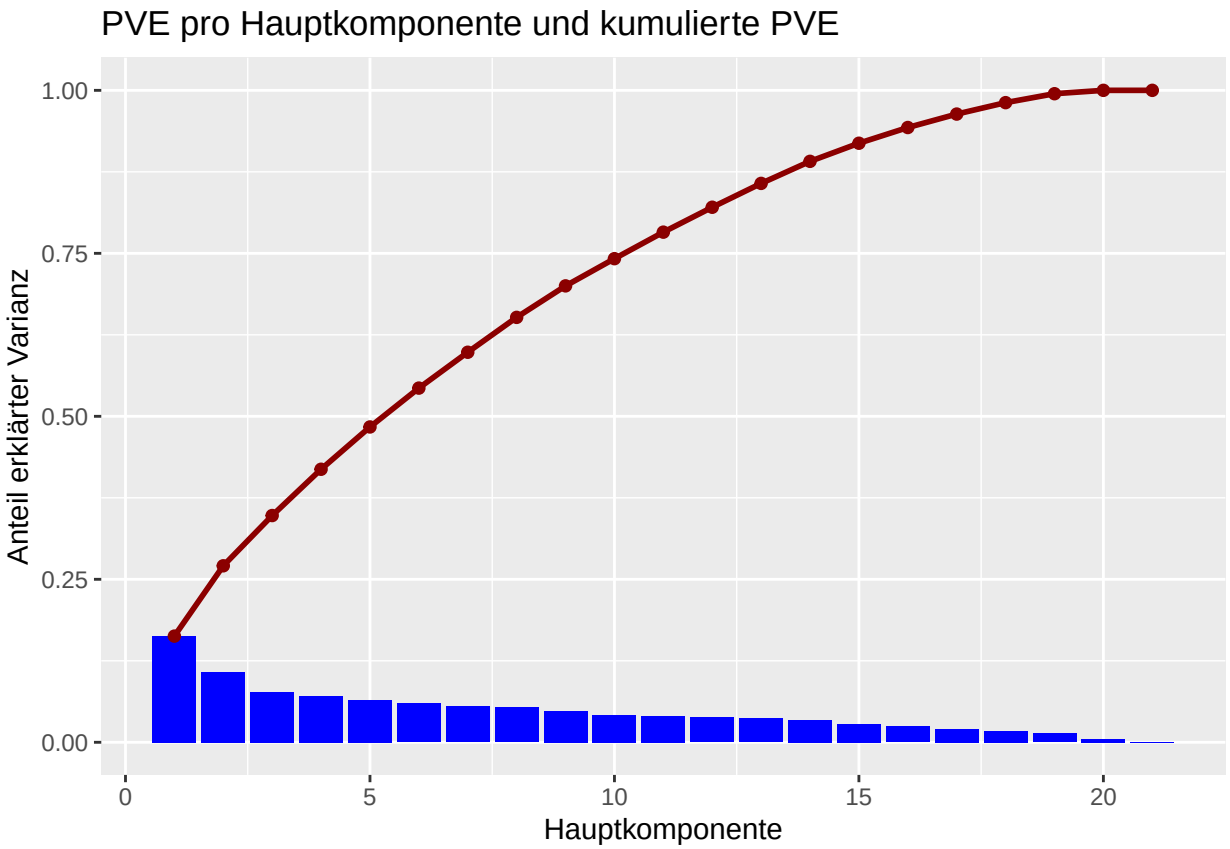
```r
# Ihre Lösung:

# wieder Dataframe für Plot erstellen
pca_plot = data.frame(
  PC1 = pca$x[, 1],
  PC2 = pca$x[, 2],
  Disease = factor(gt) # um Punkte anhand der Diagnose einzufärben
)

ggplot(pca_plot, aes(x = PC1, y = PC2, color = Disease)) +
  geom_point(alpha = 0.75)
```



Plot: die Punkte bilden erkennbare Gruppen, die sich aber überlagern Die ersten beiden Hautpkomponenten können die Unterschiede zwischen gesunden und erkrankten Patienten relativ gut sichtbar machen.

PVE

Plot Plotten Sie die Proportion of Variance explained (PVE) für jede Hauptkomponente sowie die akkumulierte PVE.



Wichtige Hauptkomponenten Wie viele Hauptkomponenten erklären mehr als 50% der Varianz?

Antwort: 6 Hauptkomponenten

Möglicherweise tragen bei Ihrem Ergebnis die letzten Hauptkomponenten keine Varianz mehr bei. Überlegen Sie, woran das liegen könnte.

Antowrt: Durch die Dummy-Codierung aller Faktorstufen entstehen lineare Abhängigkeiten zwischen den Variablen. Der df enthält also weniger unabhängige Informationen.