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

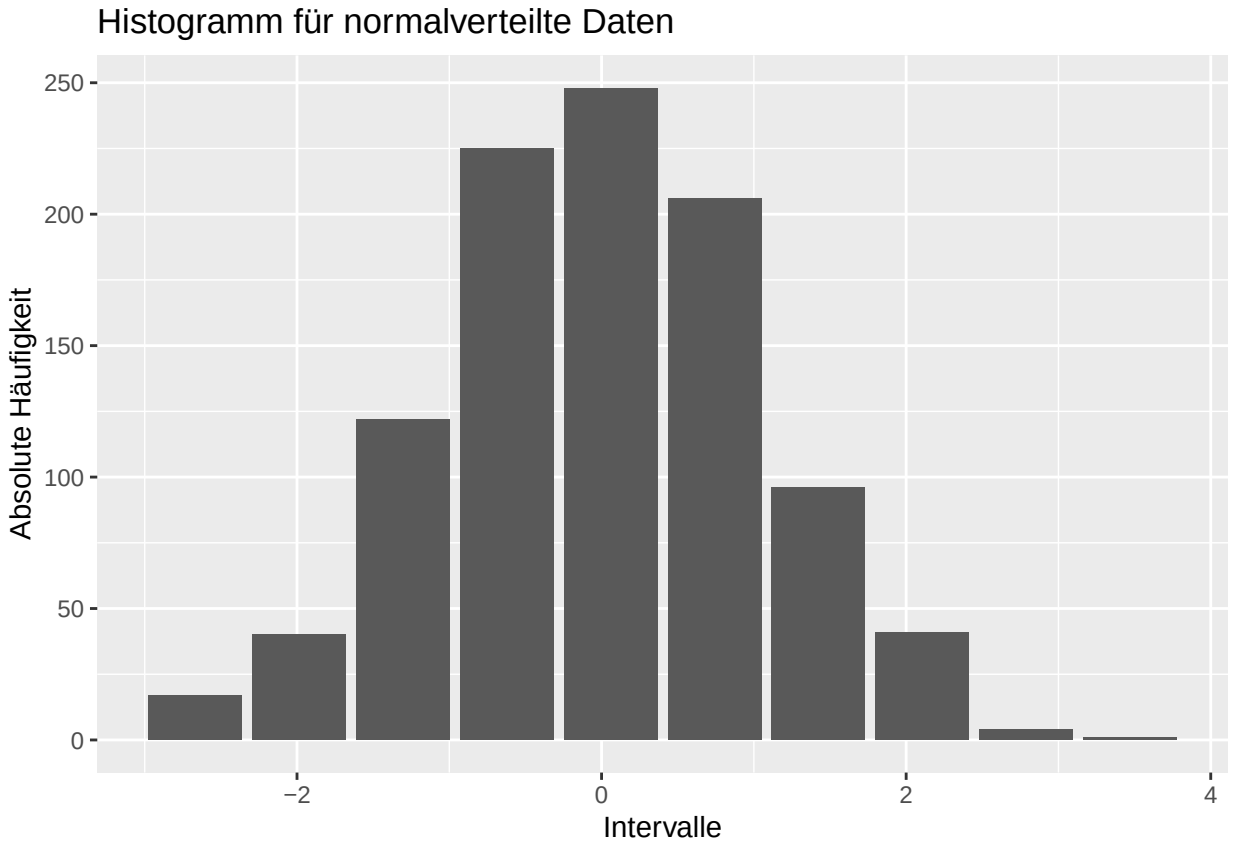
#print(h$borders[-1])
#print(h$borders[length(-h$borders)])

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
```