

Sommersemester 2022

Jens Klenke

## R-Propädeutikum

### Lösung Übungsaufgaben 1

#### 1 Übungsaufgaben zu Vektoren

1.1 Erzeuge einen Vektor `numbers` mit den Elementen  $(4 \ 6 \ -3 \ 2.5 \ 18 \ \pi \ 85)$ .

```
numbers <- c(4 , 6, -3, 2.5, 18, pi, 85)
```

1.2 Berechne das arithmetische und das harmonische Mittel von `numbers`.

```
# arithmetisches Mittel  
mean(numbers)
```

```
## [1] 16.52023
```

```
# harmonisches Mittel  
length(numbers) / (sum(1/numbers)) # 1. Möglichkeit
```

```
## [1] 8.055574
```

```
1/mean(1/numbers) # 2. Möglichkeit
```

```
## [1] 8.055574
```

- 1.3 Wir kommen zu dem Schluss, dass die höchste und die niedrigste Zahl die Schätzung verzerren und entscheiden darum, diese Werte zu ignorieren. Ersetze beide Werte durch NA und berechne die Mittelwerte aus Aufgabe 1.2 erneut.

```
numbers[which.min(numbers)] <- NA      # kleinsten Wert ersetzen
numbers[which.max(numbers)] <- NA      # größten Wert ersetzen
# oder einfach
numbers[c(3, 7)] <- NA                 # 3. und 7. Wert ersetzen

# Mittelwerte berechnen und 'NA's ignorieren
mean(numbers, na.rm = TRUE)
```

```
## [1] 6.728319
```

```
1/mean(1/numbers, na.rm = TRUE)
```

```
## [1] 4.199803
```

- 1.4 Nutze die Funktion `seq()` um die Folge  $(0, 0.5, 1, 1.5, \dots, 99, 99.5, 100)$  zu erzeugen. Wie viele Elemente besitzt dieser Vektor? Überprüfe Deine Vermutung mit `length()`.

```
x <- seq(from = 0, to = 100, by = 0.5)
x
```

```
## [1] 0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5
## [11] 5.0 5.5 6.0 6.5 7.0 7.5 8.0 8.5 9.0 9.5
## [21] 10.0 10.5 11.0 11.5 12.0 12.5 13.0 13.5 14.0 14.5
## [31] 15.0 15.5 16.0 16.5 17.0 17.5 18.0 18.5 19.0 19.5
## [41] 20.0 20.5 21.0 21.5 22.0 22.5 23.0 23.5 24.0 24.5
## [51] 25.0 25.5 26.0 26.5 27.0 27.5 28.0 28.5 29.0 29.5
## [61] 30.0 30.5 31.0 31.5 32.0 32.5 33.0 33.5 34.0 34.5
## [71] 35.0 35.5 36.0 36.5 37.0 37.5 38.0 38.5 39.0 39.5
## [81] 40.0 40.5 41.0 41.5 42.0 42.5 43.0 43.5 44.0 44.5
## [91] 45.0 45.5 46.0 46.5 47.0 47.5 48.0 48.5 49.0 49.5
## [101] 50.0 50.5 51.0 51.5 52.0 52.5 53.0 53.5 54.0 54.5
## [111] 55.0 55.5 56.0 56.5 57.0 57.5 58.0 58.5 59.0 59.5
```

```
## [121] 60.0 60.5 61.0 61.5 62.0 62.5 63.0 63.5 64.0 64.5
## [131] 65.0 65.5 66.0 66.5 67.0 67.5 68.0 68.5 69.0 69.5
## [141] 70.0 70.5 71.0 71.5 72.0 72.5 73.0 73.5 74.0 74.5
## [151] 75.0 75.5 76.0 76.5 77.0 77.5 78.0 78.5 79.0 79.5
## [161] 80.0 80.5 81.0 81.5 82.0 82.5 83.0 83.5 84.0 84.5
## [171] 85.0 85.5 86.0 86.5 87.0 87.5 88.0 88.5 89.0 89.5
## [181] 90.0 90.5 91.0 91.5 92.0 92.5 93.0 93.5 94.0 94.5
## [191] 95.0 95.5 96.0 96.5 97.0 97.5 98.0 98.5 99.0 99.5
## [201] 100.0
```

```
length(x)
```

```
## [1] 201
```

**1.5 Erzeuge einen neuen Vektor characters mit den Elementen  $(a \ a \ a \ b \ b \ b \ b \ c \ c)$ . Finde dazu heraus wie die Funktion `rep()` funktioniert und nutze diese.**

```
characters <- rep(c("a", "b", "c"), times = c(3, 4, 2))
characters
```

```
## [1] "a" "a" "a" "b" "b" "b" "b" "c" "c"
```

**1.6 Überschreibe jetzt den Vektor characters mit  $(x \ y \ z \ x \ y \ z \ x \ y \ z)$ . Nutze wieder die Funktion `rep()`.**

```
characters <- rep(c("x", "y", "z"), times = 3)
characters
```

```
## [1] "x" "y" "z" "x" "y" "z" "x" "y" "z"
```

**1.7 Ersetze nun alle Elemente mit dem Inhalt "z" durch "v".**

```
characters[which(characters == "z")] <- "v"  
characters
```

```
## [1] "x" "y" "v" "x" "y" "v" "x" "y" "v"
```

Kürzer:

```
characters[characters == "z"] <- "v"  
characters
```

```
## [1] "x" "y" "v" "x" "y" "v" "x" "y" "v"
```

## 1.8 Kopiere folgenden Code in Dein R-Skript:

```
a <- c(2,5,7,5,12,6)  
b <- c(1,2,3,4,5,6)  
x <- c(1:2)  
y <- 3  
z <- c(1,2,3,4)
```

## 1.9 Berechne nun $a + b$ , $a + x$ , $a + y$ , $a + z$ . Finde heraus, wie R jeweils vorgeht und schreibe einen kurzen Kommentar.

```
a + b
```

```
## [1] 3 7 10 9 17 12
```

Normale Vektoraddition.

```
a + x
```

```
## [1] 3 7 8 7 13 8
```

$x$  wird so häufig wiederholt, bis der Vektor die gleiche Länge hat wie der Vektor  $a$  und wird dann addiert.

```
a + y
```

```
## [1]  5  8 10  8 15  9
```

Der Skalar  $y$  wird einfach auf jedes Element von  $a$  addiert.

```
a + z
```

```
## Warning in a + z: longer object length is not a multiple of shorter  
## object length
```

```
## [1]  3  7 10  9 13  8
```

Achtung:  $a$  und  $z$  haben nicht die gleiche Länge und  $a$  ist kein vielfaches von  $z$ ! R wiederholt den Vektor und füllt die fehlenden Werte von vorne auf. Allerdings wird eine Warnmeldung ausgegeben.

**1.10 Erzeuge einen Vektor mit den Elementen  $(1 \ 2 \ 3 \ a \ b)$  (Also eine Mischung aus numeric und character). Was passiert? Schreibe einen Kommentar.**

```
v <- c(1, 2, 3, "a", "b")  
class(v)
```

```
## [1] "character"
```

R nimmt immer die bestmögliche Klasse (die Klasse, welche die meisten “Berechnungen” zulässt, vgl. Skalenniveau), in der die Elemente vereint werden können. In diesem Fall ist es die Klasse `character`.

## 2 Übungsaufgaben zu Matrizen

**2.1 Erzeuge mit dem Inputvektor 1:12 und `matrix()` folgende Matrix  $X$ .**

$$X = \begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \\ 7 & 8 \\ 9 & 10 \\ 11 & 12 \end{pmatrix}$$

```
X <- matrix(1:12, ncol = 2, byrow = TRUE)
X
```

```
##      [,1] [,2]
## [1,]    1    2
## [2,]    3    4
## [3,]    5    6
## [4,]    7    8
## [5,]    9   10
## [6,]   11   12
```

**2.2** Nimm die Matrix aus 2.1 und vertausche die Spalten. Das Ergebnis soll an die Variable  $Y$  übergeben werden.

```
Y <- X[ , 2:1] # oder
Y <- X[ , c(2,1)]
Y
```

```
##      [,1] [,2]
## [1,]    2    1
## [2,]    4    3
## [3,]    6    5
## [4,]    8    7
## [5,]   10    9
## [6,]   12   11
```

**2.3** Berechne  $XY^T$ .

```
X %*% t(Y)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    4   10   16   22   28   34
## [2,]   10   24   38   52   66   80
## [3,]   16   38   60   82  104  126
## [4,]   22   52   82  112  142  172
## [5,]   28   66  104  142  180  218
## [6,]   34   80  126  172  218  264
```

## 2.4 Erzeuge eine $2 \times 2$ Matrix aus der 2. und 5. Zeile der Matrix $X$ .

```
X[c(2, 5), ]
```

```
##      [,1] [,2]  
## [1,]    3    4  
## [2,]    9   10
```

## 2.5 Erzeuge die Matrix $X$ mit $X \leftarrow \text{matrix}(8:-7, \text{nrow} = 4)$ .

```
X <- matrix(8:-7, nrow = 4)  
X
```

```
##      [,1] [,2] [,3] [,4]  
## [1,]    8    4    0   -4  
## [2,]    7    3   -1   -5  
## [3,]    6    2   -2   -6  
## [4,]    5    1   -3   -7
```

Es wird jetzt `nrow` und nicht `ncol` benutzt. Außerdem wurde eine abfallende Folge von natürlichen Zahlen mithilfe von `:` erzeugt.

1. Ersetze die Elemente auf der Hauptdiagonalen durch 'NA's.

```
diag(X) <- NA
```

2. Ersetze jetzt alle 'NA's in der Matrix mit dem Wert 1. Nutze dazu die Funktion `'is.na()'`.

```
X[is.na(X)] <- 1
```

`is.na(X)` gibt eine Matrix mit logischen Einträgen (`TRUE`/ `FALSE`) zurück, wobei für Elemente mit NA der Wert `TRUE` gesetzt wird.