Programmieren mit R für Einsteiger

1. Grundlagen / 1.4 Statistik

Berry Boessenkool 2022-02-25 11:40

Statistische Maßzahlen I: Mittelwert, Streuung, Wertebereich

```
# Vektor mit Körpergrößen:
groesse <- c(149.3, 173.6, 172.2, 172.9, 161.6, 179.2,
             164.8, 162.8, 180.5, 165.1, 181.7, 171.4,
             172.1, 148.1, 161.1, 171.9, 186.9) # cm
```

```
mean(groesse)    # Mittelwert
## [1] 169.1294
```

```
var(groesse)     # Varianz: cm^2
## [1] 112.591
sd(groesse)      # Standardabweichung: cm
## [1] 10.61089
```

```
min(groesse)     # Minimum
## [1] 148.1
max(groesse)     # Maximum
## [1] 186.9
```

```
range(groesse)   # Wertebereich
## [1] 148.1 186.9
```

Statistische Maßzahlen II: auch ohne Normalverteilung

```
median(groesse) # Ausreißer-unabhängig (anders als mean)
## [1] 171.9
```

```
mad(groesse)    # Median absolute deviation
## [1] 10.82298
```

```
quantile(groesse) # Anteil < bestimmte Werte
##    0%   25%   50%   75%  100%
## 148.1 162.8 171.9 173.6 186.9
```

```
quantile(groesse, probs=0.80) # 80% ist hier drunter
##    80%
## 178.08
```

```
summary(groesse)
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   148.1   162.8   171.9   169.1   173.6   186.9
```

Zeigt auch Anzahl NAs an (falls vorhanden), kann auch für Tabellen verwendet werden, siehe entsprechenden Abschnitt 3.1

Sortierungen

```
groesse <- round(groesse[1:8])    ;    groesse
## [1] 149 174 172 173 162 179 165 163
```

```
sort(groesse) # Aufsteigend sortieren
## [1] 149 162 163 165 172 173 174 179
```

```
sort(groesse, decreasing=TRUE)
## [1] 179 174 173 172 165 163 162 149
```

```
order(groesse)
## [1] 1 5 8 7 3 4 2 6
```

Das kleinste ist an Stelle 1, das zweitkleinste in `groesse[5]`, etc.

```
gewicht <- c(49, 77, 66, 91, 69, 72, 73, 74)
gewicht[order(groesse)] # Sortieren nach Reihenfolge Größe
## [1] 49 69 74 73 66 91 77 72
```

Zufallszahlen

```
sample(0:9, size=7)          # Zufällig Werte aus Vektor ziehen
## [1] 2 7 1 4 9 8 6
```

```
sample(0:9, size=7, replace=TRUE)      # Ziehen mit Zurücklegen
## [1] 5 6 9 0 7 6 5
```

Kontinuierliche Verteilungen:

```
rnorm(n=5, mean=20, sd=3.5)              # aus Normalverteilung
## [1] 21.9 19.0 20.4 19.9 11.2
rexp(n=5, rate=1/20)                     # Exponentialverteilung
## [1]  7.27 23.47 22.79  8.56 29.17
runif(n=5, min=15, max=25)          # Gleichverteilung (uniform)
## [1] 22.8 19.3 24.3 22.7 17.6
rbeta(n=5, shape1=3, shape2=9)                  # Beta-verteilung
## [1] 0.1879 0.0687 0.2998 0.4172 0.1498
```

Diskrete Verteilungen:

```
rpois(n=5, lambda=20)                    # Poisson-verteilung
## [1] 19 13 23 29 29
rbinom(n=5, size=100, prob=1/5)          # Binomial-verteilung
## [1] 27 16 21 27 22
```

Zusammenfassung

Statistische Maßzahlen, Sortierungen und Zufallszahlen:

- `mean`, `var`, `sd`
- `min`, `max`, `range`, `median`, `quantile`, `summary`
- `round`, `sort`, `order` (decreasing)
- `sample`, `rnorm` etc

unique und duplicated

```
val <- c(1, 7, 3, 3, 8, 5, 6, 6, 6, 7)
```

```
unique(val) # ursprüngliche Reihenfolge beibehalten
## [1] 1 7 3 8 5 6
```

```
duplicated(val)
##  [1] FALSE FALSE FALSE  TRUE FALSE FALSE FALSE  TRUE
##  [9]  TRUE  TRUE
duplicated(val, fromLast=TRUE)
##  [1] FALSE  TRUE  TRUE FALSE FALSE FALSE  TRUE  TRUE
##  [9] FALSE FALSE
```

Runden

```
werte <- c(149.3, 173.6, 172.2, 172.9, 161.6, 179.2, 164.8, 142.8)
```

```
round(werte, digits=-1) # Auf 10er runden
## [1] 150 170 170 170 160 180 160 140
round(werte, -2) # Auf 100er runden
## [1] 100 200 200 200 200 200 200 100
```

```
round(werte/5)*5 # Auf 5er runden
## [1] 150 175 170 175 160 180 165 145
```

Optionen zur Ausgabe

```
pi
## [1] 3.141593
```

Das Verhalten von R kann in vielen Optionen angepasst werden, z.B. für den Umgang mit Warnmeldungen oder Ausgaben (printed output).

`digits` regelt, wieviele relevante Nachkommastellen angezeigt werden (bezogen auf die Größenordnung der Zahl):

```
oo <- options(digits=3) # ca 2 Nachkommastellen anzeigen
oo # bisheriger Wert jetzt in oo (old options)
## $digits
## [1] 7
```

```
pi
## [1] 3.14
```

```
options(oo)  ;  rm(oo) # Einstellungen zurücksetzen
```

Reproduzierbarkeit

```
sample(1:50, 3)
## [1] 18 45  5
sample(1:50, 3)
## [1] 37 29 22
```

Für die Folien immer wieder die gleichen "Zufallszahlen" erzeugen
-> Startpunkt für den RNG (Random Number Generator) setzen:

```
set.seed(12345)
```

```
sample(1:50, 3)
## [1] 14 16 26
```

```
set.seed(12345)
sample(1:50, 3)
## [1] 14 16 26
```