Programmieren mit R für Einsteiger

2. Grundlagen / 2.2 Logik

Berry Boessenkool 2022-02-25 11:40

Boolean = logical = Wahrheitswerte

```
7 > 4
## [1] TRUE
```

SHIFT + <

```
7 > 42
## [1] FALSE
```

```
T # = TRUE. T kann überschrieben werden, nicht nutzen
## [1] TRUE
```

```
! 7 > 4                # NICHT-Operator (Negierung, Gegenteil)
## [1] FALSE
```

```
TRUE & TRUE
## [1] TRUE
TRUE & FALSE
## [1] FALSE
```

SHIFT + 6 `# UND-operator`

```
TRUE | FALSE
## [1] TRUE
```

AltGr + <, Option + 7 `# ODER-operator`

Logische Operatoren I: Vektoren

```
x <- c(1, 2, 3, 4, 5)
y <- c(4, 5, 6, 7, 1)
```

Viele Operatoren sind vektorisiert, gehen also für einen ganzen Vektor:

```
x > 3
## [1] FALSE FALSE FALSE  TRUE  TRUE
y < 6
## [1]  TRUE  TRUE FALSE FALSE  TRUE
```

```
x>3 & y<6          # für Vektor mit mehreren Wahrheitswerten
## [1] FALSE FALSE FALSE FALSE  TRUE
x>3 | y<6
## [1]  TRUE  TRUE FALSE  TRUE  TRUE
```

`&&` und `||` evaluieren nur den ersten Wert:

```
x>3 && y<6                    # für einen einzigen Wahrheitswert
## [1] FALSE
```

Logische Operatoren II: Größenvergleich

```
werte <- c(32, 28, 29, 30, 31, 32)
```

```
werte < 30
## [1] FALSE  TRUE  TRUE FALSE FALSE FALSE
werte <= 30
## [1] FALSE  TRUE  TRUE  TRUE FALSE FALSE
```

```
werte > 30
## [1]  TRUE FALSE FALSE FALSE  TRUE  TRUE
werte >= 30
## [1]  TRUE FALSE FALSE  TRUE  TRUE  TRUE
```

```
werte == 30
## [1] FALSE FALSE FALSE  TRUE FALSE FALSE
werte != 30
## [1]  TRUE  TRUE  TRUE FALSE  TRUE  TRUE
```

Logische Operatoren III: `which`, `any`, `all`

```
werte <- c(32, 28, 29, 30, 31, 32)
werte < 30
## [1] FALSE  TRUE  TRUE FALSE FALSE FALSE
```

```
which(werte < 30) # -> Index: Stellen mit TRUE im Vektor
## [1] 2 3
```

```
which(werte == max(werte))
## [1] 1 6
```

```
which.max(werte) # nur der erste Index!
## [1] 1
```

```
any(werte < 30) # ist mindestens eins der T/F Werte wahr?
## [1] TRUE
```

```
all(werte < 30) # sind alle TRUE?
## [1] FALSE
```

Logische Operatoren IV: `sum`, `mean`

```
werte < 30
## [1] FALSE  TRUE  TRUE FALSE FALSE FALSE
```

```
as.numeric(werte < 30) # intern als Zahl: 0=FALSE, 1=TRUE
## [1] 0 1 1 0 0 0
```

```
sum(werte < 30) # Anzahl TRUEs im Vektor
## [1] 2
```

```
mean(werte < 30) # Anteil TRUE Werte
## [1] 0.3333333
```

Logische Selektion (= Subsetting, Indexing)

```
werte <- c( 32,  28,  29,  30,  31,  32)
namen <- c("a", "b", "c", "d", "e", "f")
```

```
namen[4]
## [1] "d"
```

```
werte < 30
## [1] FALSE  TRUE  TRUE FALSE FALSE FALSE
```

```
namen[werte < 30]
## [1] "b" "c"
```

Zum Auswählen mit logischen Werten ("Filtern") müssen beide Vektoren gleich lang sein.

Zusammenfassung

Logische Werte, Größenvergleich:

- `TRUE`, `FALSE` (nicht nutzen: `T`, `F`)
- `!`, `&`, `|`, `&&`, `||`
- `<`, `>`, `<=`, `>=`, `==`, `!=`
- `which`, `which.max`, `any`, `all`, `sum`, `mean`
- `vec[logical]`

Vorsicht bei Überprüfung von Gleichheit

Gleichheit nur für ganze oder gerundete Zahlen prüfen!

```
0.4 - 0.1 == 0.3          # nicht das erwartete Ergebnis!
## [1] FALSE
```

```
print(0.4-0.1 , digits=22)
## [1] 0.3000000000000004
```

```
round(0.4 - 0.1, digits=5) == round(0.3, digits=5)    # OK
## [1] TRUE
```

```
all.equal(0.4 - 0.1, 0.3) # Hat eine Fehlertoleranz
## [1] TRUE
```

Variante für Vektoren:

```
berryFunctions::almost.equal(c(6.34, 9.69, 3.77), 9.69)
## [1] FALSE  TRUE FALSE
```

Mehr solcher Tücken im R inferno.

Mehr Details zu Logik


```
T             # kann abgekürzt werden, allerdings:
## [1] TRUE
T <- 99       # kann T überschrieben werden
T <- FALSE    # Streich: heimlich beim Kollegen eintippen ;)
TRUE <- 77    # ist geschützt
## Fehler in TRUE <- 77:  invalid (do_set) left-hand side
to assignment
```

```
xor(TRUE, FALSE) # EXKLUSIVES ODER (genau 1 von 2 wahr?)
## [1] TRUE
```

```
isTRUE(T); isTRUE(F); isTRUE(NA) # T, F, F (für NAs)
```

Mehr Details zu `&`

`A & B` sowie `A && B` unterscheiden sich in einer weiteren Sache. Wenn `A` falsch ist, wird in der zweiten Variante `B` gar nicht ausgewertet, weil die Ausgabe kein Vektor sein kann, wo noch ein TRUE auftreten könnte.

```
rechnung <- function(out){cat("Rechnung läuft\n") ; out}

# Ausgabe beider 'rechnung'-Aufrufe:
rechnung(FALSE) & rechnung(TRUE)
## Rechnung läuft
## Rechnung läuft
## [1] FALSE

# Nur die linke Instanz wird ausgeführt:
rechnung(FALSE) && rechnung(TRUE)
## Rechnung läuft
## [1] FALSE
```

Wahrheitstabellen

`&` hat Vorrang vor `|` (operator precedence, wie `*` vor `+`):

```
berryFunctions::TFtest(a|b&c, a|(b&c), (a|b)&c, na=FALSE)
##       a     b     c __ a | b & c __ a | (b & c) __ (a | b) & c
## 1  TRUE  TRUE  TRUE          TRUE           TRUE           TRUE
## 2  TRUE  TRUE FALSE          TRUE           TRUE          FALSE
## 3  TRUE FALSE  TRUE          TRUE           TRUE           TRUE
## 4  TRUE FALSE FALSE          TRUE           TRUE          FALSE
## 5 FALSE  TRUE  TRUE          TRUE           TRUE           TRUE
## 6 FALSE  TRUE FALSE         FALSE          FALSE          FALSE
## 7 FALSE FALSE  TRUE         FALSE          FALSE          FALSE
## 8 FALSE FALSE FALSE         FALSE          FALSE          FALSE
```

Siehe auch `?Syntax`