Matematické operace v R

Aritmetické operace

Aritmetické operace lze uskutečnit mezi dvěma (či více) hodnotami, ale i mezi dvěma (či více) vektory shodných délek.

Můžeme užít tzv. prefix notaci ve tvaru `'operátor'(…, …)` vhodnou pro dynamické iterování. Např. `2*3` zapíšeme jako `'*'(2, 3)`.

Sčítání

Sčítání provádíme znakem plus +.

```
15 + 25 + 35
#> [1] 75

c(1, 2) + c(10, 20)         # = c(11, 12)
#> [1] 11 12

'+'(2, 3)                   # 5
'+'(15, 25, 35)             # Error: operator needs one or two arguments
'+'('+'(15, 25), 35)        # 75
'+'(c(1, 2), c(10, 20))     # c(11, 22)
```

Odčítání

Odčítání provádíme znakem minus -.

```
35 - 25 - 15                # -5
c(12, 25) - c(3, 6)         # c(9, 19)
'-'(12, 3)                  # 9
'-'(35, 25, 15)             # Error: operator needs one or two arguments
'-'('-'(35, 25), 15)        # -5
'-'(c(12, 25), c(3, 6))     # c(9, 19)
```

Násobení

Násobení provádíme znakem hvězdičky *.

```
35 * 25 * 15                # 13125
c(12, 25) * c(3, 6)         # c(36, 150)
'*'(12, 3)                  # 36
'*'('*'(35, 25), 15)        # 13125
'*'(c(12, 25), c(3, 6))     # c(36, 150)
```

Dělení

Dělení provádíme znakem lomítka /.

```
12 / 3                      # 4
45 / 5 / 3                  # 3
c(12, 25) / c(3, 5)         # c(4, 5)
'/'(12, 3) # 4
'/'('/'(45, 5), 3)          # 3
'/'(c(12, 25), c(3, 5))     # c(4, 5)
```

Mocnění

Mocnění provádíme pomocí dvou hvězdiček ** nebo ^.

```
2 ^ 3                       # 8
2 ** 3                      # 8; Python-like notace
4 ^ 3 ** 2                  # 262144
4 ^ (3 ** 2)                # 262144
(4 ^ 3) ** 2                # 4096; pozor na uzávorkování !!!
c(25, 36) ^ 0.5             # c(5, 6); odmocňování
c(5, 3) ^ c(2, 3)           # c(25, 27)
c(5, 3) ** c(2, 3)          # c(25, 27)
'^'(2, 3)                   # 8
'**'(2, 3)                  # Error: could not find function "**"
'^'(4, 3, 2)                # Error: operator needs one or two arguments
'^'('^'(4, 3), 2)           # 4096
'^'(c(5, 3), c(2, 3))       # c(25, 27)
```

Modulo

Modulo vrací zbytek po celočíselném dělení. 20 modulo 6 je tedy 2 (20/6 je 3, zbytek 2). Modulo provádíme pomocí dvou znaků procent %%.

```
12 %% 3                    # 0
10 %% 3                    # 1
10 %% -3                   # -2
5 %% 0                     # NaN
17 %% 23                   # 17
```

Celočíselné dělení

Někdy je potřeba dělit s celočíselným výsledkem. Vlastně je tato funkce doplňkem modula. Celočíselné dělení provádíme pomocí sekvence %/%.

```
12 %/% 3                   # 4
10 %/% 3                   # 3
10 %/% -3                  # -2
5 %/% 0                    # ha, chyták, dělení nulou v R existuje! Vrací nekonečno (Inf).
17 %/% 23                  # 0
23 %/% 17                  # 1
17 %/% 5                   # 3
(17 - 17 %% 5) / 5         # 3
(17 %/% 5) * 5 + (17 %% 5)  # 17
```

Uživatelem definované operátory

Můžeme si definovat vlastní operátory s názvem mezi dvěma znaky procent.

```
# definuji vlastní operátor
'%očaruj_pomocí%' <- function(x, y){x^2 + y}

5%očaruj_pomocí% 4         # 5 ^ 2 + 4 = 29
6 %očaruj_pomocí% -3       # 6 ^ 2 - 3 = 33

# definuji operátor, který vrací TRUE, právě když je první číslo celočíselným dělitelem druhého
'%d%' <- function(d, n){n %% d == 0}

3 %d% 9                    # TRUE
4 %d% 9                    # FALSE
'%d%'(7, 21)               # TRUE; prefix notace
```

Logické operace

Logické operace lze užít obecně nad výroky, tj. nad objekty s datovým typem *logical*.

Short AND (operátor "&") / OR (operátor "|")

- Použitelná pro vektory
- Vyhodnocuje všechny výroky vektoru a vrací jejich hodnoty

```
c(FALSE, FALSE, TRUE, TRUE) & c(FALSE, TRUE, FALSE, TRUE) # c(FALSE, FALSE, FALSE, TRUE)
c(FALSE, FALSE, TRUE, TRUE) | c(FALSE, TRUE, FALSE, TRUE) # c(FALSE, TRUE, TRUE, TRUE)
```

Long AND (operátor "&&") / OR (operátor "||")

- Použitelná také pro vektory
- Vyhodnocuje pouze nutný počet výroků (možná optimalizace) a vrací jen jednu hodnotu

```
c(FALSE, FALSE, TRUE, TRUE) && c(FALSE, TRUE, FALSE, TRUE) # FALSE
c(FALSE, FALSE, TRUE, TRUE) || c(FALSE, TRUE, FALSE, TRUE) # FALSE, protože první dvojice vrací FALSE
```

XOR (funkce xor(), vylučovací OR)

```
xor(c(FALSE, FALSE, TRUE, TRUE), c(FALSE, TRUE, FALSE, TRUE)) # c(FALSE, TRUE, TRUE, FALSE)
```

NOT (operátor "!")

Vrací výroku opačnou logickou hodnotu, než které výrok nabývá

```
! TRUE # FALSE
! 2 > 3 # TRUE
```

Funkce all()

Vrací pro vektor výroků TRUE právě tehdy, jsou-li všechny výroky rovné TRUE

```
```

```r
all(c(3 > 2, 7 %% 3 == 1, 1 == 0)) # FALSE
all(c(3 > 2, 7 %% 3 == 1, 1 >= 0)) # TRUE
```

Funkce any()

Vrací pro vektor výroků TRUE právě tehdy, je-li alespoň jeden z výroků roven TRUE

```r
any(c(3 < 2, 7 %% 3 <= 0, FALSE)) # FALSE
any(c(3 > 2, 7 %% 3 >= 1, !FALSE)) # TRUE
```

Operace porovnávání (komparace)

- Pomocí operací porovnávání lze srovnat velikost či pořadí dvou objektů stejného datového typu
- Datový typ může přitom být v podstatě libovolný
- Výsledkem operace porovnání je výrok (logical)

```r
2 == 3 # FALSE
'=='("a", "a") # TRUE; prefix notace
all.equal(c(1, 2), c(1, 2 + 1e-13), tolerance = 1e-12) # TRUE; porovnává vektory volitelnou danou tolerancí
identical(c(1, 2), c(1, 2 + 1e-13)) # FALSE, porovnává objekty a vrací TRUE jen při úplné # shodě
2 < 3 # TRUE
"b" <= "a" # FALSE; porovnává pořadí v abecedě
'>'(12, 11) # TRUE; prefix notace
FALSE >= TRUE # FALSE; porovnává hodnotu v booelovské aritmetice (TRUE := 1, FALSE := 0)
2 != 3 # TRUE
TRUE != FALSE # TRUE
'!='(FALSE, FALSE) # FALSE; prefix notace
```

Porovnání typu je obsaženo ve (%in%)

```r
c(2, 6) %in% c(1:5) # c(TRUE, FALSE)
"k" %in% LETTERS # FALSE
"J" %in% letters # FALSE
"May" %in% month.name # TRUE
'%in%'("Jan", month.abb) # TRUE; prefix notace
"a" %in% "abeceda" # FALSE
```

Ekvivalentem (wrapperem) operace %in% je funkce is.element()

```r
is.element(c(2, 6), c(1:5)) # c(TRUE, FALSE)
is.element(c(1:5), c(2, 6)) # c(FALSE, TRUE, FALSE, FALSE, FALSE)
```

Funkce typu je pravdou (isTRUE())

```r
isTRUE(3^2 > 2^3) # TRUE
```

Množinové operace

Sjednocení množin

```r
union(c(1, 2, 3), c(5, 1)) # c(1, 2, 3, 5)
```

Průnik množin

```r
intersect(c(1, 2, 3), c(5, 1)) # c(1)
```

Asymetrický rozdíl množin

```r
setdiff(c(1, 2, 3), c(5, 1)) # c(2, 3)
```

Symetrický rozdíl množin

```r
union(setdiff(c(1, 2, 3), c(5, 1)), setdiff(c(5, 1), c(1, 2, 3)))  # c(2, 3, 5)
```

Jsou si množiny rovny

```r
setequal(c(1, 2, 3), c(5, 1)) # FALSE
setequal(c(1, 2, 3), c(3, 2, 1)) # TRUE
```

Zaokrouhlování, formátování čísel

Zaokrouhlení čísla x pomocí *round(x, digits)* na *digits* desetinných míst

```
round(1.4, digits = 0) # 1
round(-146.655, 2) # -146.66
```

Zaokrouhlení čísla x pomocí *signif(x, digits)* na *digits* platných cifer

```
signif(1.458, digits = 1) # 1
signif(1.458, digits = 2) # 1.5
signif(1.458, digits = 3) # 1.46
signif(1.458, digits = 4) # 1.458
```

Formátování čísla *x* pomocí *format(x, nsmall)* na *nsmall* fixních desetinných cifer

```
format(1.45, nsmall = 1) # "1.45"
format(1.45, nsmall = 2) # "1.45"
format(1.45, nsmall = 3) # "1.450"
```

Konstanty

Některé matematické a textové konstanty jsou součástí jádra, např.

```
pi # 3.141593; Ludolfovo číslo
exp(1) # 2.718282; Eulerovo číslo
Inf # nekonečno
-Inf # mínus nekonečno
letters # malá písmena anglické abecedy
LETTERS # velká písmena anglické abecedy
month.name # názvy měsíců
month.abb # zkratky názvů měsíců
weekdays(Sys.Date()) # vrátí aktuální den v týdnu
weekdays(seq(as.Date("2017-10-23"), as.Date("2017-10-29"), 1))  # vypíše názvy dnů v týdnu
```

Fyzikální konstanty jsou dostupné v balíčku *constants*

Odkazy

Související články

- Co je R
- R Markdown

Použitá literatura

Reference