Wprowadzeni do R

A. M. Machno

28 września 2015

Kalkulator

Środowisko R można wykorzystac do właściwie dowolnych obliczeń arytmetycznych na liczbach. To co jest wyświetlane w szarej ramce wpisane należy wpisać do konsoli i po wciśnieciu *enter* otrzymamy to co jest w białej ramce (po ##).

```
1+15
```

```
## [1] 16
```

```
4/2
```

```
## [1] 2
```

```
6^3
```

```
## [1] 216
```

```
625^(.5)
```

```
## [1] 25
```

```
(1+3)/(2+3^2)
```

```
## [1] 0.3636364
```

Funkcje

Możemy korzystać z ogromnej ilości funkcji wbudowanych w R oraz tworzyć własne. Funkcję wywołujemy wpisując jej nazwę oraz argumenty w nawiasie okrągłym.

-pierwiastek:

```
sqrt(64)
```

```
## [1] 8
```

-funkcja eksponencjalna

```
exp(2)
```

```
## [1] 7.389056
```

Aby dowiedzieć się więcej na temat danej funkcji wpisujemy `?nazwa_funkcji`. Niektóre funkcje mają więcej argumentów, np. funkcja licząca logarytm (`log`) ma dwa argumenty (liczbę, której logarytm chcemy policzy.c oraz podstawę logarytmu). W takiej sytuacji musimy musimy podac dwa argumenty.

liczymy $log_2(16)$:

```
log(16, 2)
```

```
## [1] 4
```

Spoglodając do opisu funkcji `log` możemy dowiedzieć się, że liczba logarytmowana to argument `x`, a podstawa to argument `base`. Wywołując funkcję możemy odwołać się do tych nazw nie pamietając o kolejności argumentów.

```
log(x=27, base=3)
```

```
## [1] 3
```

```
log(base=3, x=27)
```

```
## [1] 3
```

Niektóre agumenty mają wartości domyślne (właściwie to jest bardzo częsty przypadek). W przypadku logarytmu, argument `base` ma wartość domyślną równą *e*. Zatem jeżeli chcemy policzyć logarytm naturalny nie musimy wpisywać podstawy.

```
log(2)
```

```
## [1] 0.6931472
```

```
log(x=12)
```

```
## [1] 2.484907
```

Jednak jeżeli funkcja nie ma wartości domyślnej do argumentu, nie wpisując wartości otrzymamy błąd. Np. `log(base=5)`.

Wiele funkcji nie ma argumentów, lub wszystkie jej argumenty mają wartości domyślne, i tak możemy dowiedzieć się w jakim katalogu pracujemy wpisując `getwd()` lub wyświetlając wszystkie obiekty, które przechowujemy w naszym środowisku `ls()`.

```
getwd()
```

```
## [1] "C:/Users/Artur/Dropbox/AGH/dydaktyka/statystyka ZiP/2015-2016/R"
```

```
ls()
```

```
## character(0)
```

Jeszcze nie stworzyliśmy ani nie wczytaliśmy nic do naszego środowiska, dlatego wyświetla pustą listę.

Wektory

Podstawowymi wartościami w R są wektory. Podstawową funkcją, która tworzy wektor jest `c()`.

```
c(1,3,11,-1,.3)
```

```
## [1]  1.0  3.0 11.0 -1.0  0.3
```

Pracując na wektorach, przeważnie chcemy przypisać jakiejś zmiennej wartości. Stórzmy zatem dwa 5-elementowe wektory `x` i `y`. Do przypisywania wartości służy operator `<-` (lub `->`).

```
x<-c(2, -1, 3, 1.5, -.5)
y<-c(1, 2, 3, -1, -5)
```

Tym razem po wpisaniu komend do consoli nie dostaliśmy żadnej odpowiedzi, ale możemy odwołać sie do zmiennych.

```
x; y
```

```
## [1]  2.0 -1.0  3.0  1.5 -0.5
```

```
## [1]  1  2  3 -1 -5
```

Wszystkie podstawowe funkcje liczbowe możemy stosować do wektorów. Działania są wykonywane dla pierwszych elementów wektorów i wartość jest pierwszym elementem wektora wyniku itd. Innymi słowy jest działanie wyraz po wyrazie.

```
exp(x)
```

```
## [1]  7.3890561  0.3678794 20.0855369  4.4816891  0.6065307
```

```
x+y
```

```
## [1]  3.0  1.0  6.0  0.5 -5.5
```

```
x/y
```

```
## [1]  2.0 -0.5  1.0 -1.5  0.1
```

```
x^y
```

```
## [1]   2.0000000   1.0000000  27.0000000   0.6666667 -32.0000000
```

Wiele funkcji jako argument (lub kilka argumentów) ma wektor. Podstawowe to np. suma `sum()`, średnia arytmetyczna `mean()`.

```r
sum(x); sum(y)
```

```
## [1] 5
```

```
## [1] 0
```

```r
mean(x); mean(y)
```

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

```
## [1] 0
```

Dzieki temu bardzo szybko możemy policzyć np. wariancję `x`:

```r
mean((x-mean(x))^2); mean((y-mean(y))^2)
```

```
## [1] 2.3
```

```
## [1] 8
```

```r
sum((x-mean(x))^2)/length(x); sum((y-mean(y))^2)/length(y)
```

```
## [1] 2.3
```

```
## [1] 8
```

```r
sum((x-mean(x))^2)/(length(x)-1); sum((y-mean(y))^2)/(length(y)-1)
```

```
## [1] 2.875
```

```
## [1] 10
```

Oczywiście jest też funckja `var()`, która oblicza wariancję dla wartości z wektora.

```r
var(x); var(y)
```

```
## [1] 2.875
```

```
## [1] 10
```

Obiekty podstawowe

W R rozróżniamy 6 podstawowych typów obiektów (które przeważnie są elementami wektora):

1. charakter (napis). Wpisujemy w cudzysłowiu, np. `"to jest napis"`.
2. complex (liczba zespolona). Wpisujemy cześć rzeczywistą plus część urojoną i, bez znaku mnożenia, np. `1+2i`.
3. numeric (liczba rzeczywista). Po prosu wpisujemy liczbę, może byc poprzedzona minusem aby była ujemna, część dziesiętna po **kropce**.
4. integer (liczba naturalna). Wpisujemy liczbę i L, np. `2L`.
5. logical (wartość logiczna TRUE/FALSE). Wpsujemy `TRUE` lub `FALSE`, przekształcane są w odpowiednio 1 i 0.

Rozróżnienie jest ważne, dlatego, że wektory muszą być jednorodne ze względu na obiekty (wszystkie elementy wektora muszą mieć ten sam typ). Typy wypisałem w nieprzypadkowej kolejności, zamiana wektora, który ma wyższy typ bezproblemowo zamienimy na tym niższy, odwrotna operacja przeważnie nie jest możliwa. Jezeli spróbujemy stworzyć wektor niejednorodny, R wybierze obiekt najniższego rzędu w tym wektorze i pozostałe lementy przekształci w ten tym.

```
mix_vect_1<-c(1L, TRUE, 2+3i)
typeof(mix_vect_1); mix_vect_1
```

```
## [1] "complex"
```

```
## [1] 1+0i 1+0i 2+3i
```

```
mix_vect_2<-c(1,2, 'napis')
typeof(mix_vect_2); mix_vect_2
```

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

```
## [1] "1"     "2"     "napis"
```

Z punktu widzenia tego kursu najważniejsze są wektory numeryczne i logiczne (okazjonalnie będziemy korzystać z napisów).

Wartości specjalne

W wyniku niektórych operacji możemy otrzymac nieksończoność `Inf`. Nieskończoność możemy też używać do obliczeń.

```
1/0
```

```
## [1] Inf
```

```
.99^Inf
```

```
## [1] 0
```

Jednak niektórych działań nie da się wykonać i otrzymamy liczbe nioznaczoną.

```
0/0
```

```
## [1] NaN
```

```
(-1)^.1
```

```
## [1] NaN
```

Jeżeli chcielibysmy przekształcić napis w liczbę, a napis nie będzie liczba otrzymamy również wartość nieoznaczoną:

```
as.numeric(mix_vect_2)
```

```
## Warning: pojawiły się wartości NA na skutek przekształcenia
```

```
## [1]  1  2 NA
```

Widzimy, że dwa pierwsze lementy przekształcono w liczny, a trzeci został zastąpiony przez `NA`. Różnicą między `NaN` i `NA` nie będziemy się zajmować i obie będziemy traktować jak wartości brakujące lub błędne w zalezności od kontekstu.

Atrybuty

Wiele obiektów w R posiada atrybuty, w szczególności wektory. Przykładowe atrybuty to:

- names, dimnames (nazwa, nazwa wymiarów dla macierzy i innych wielowymiarowych obiektów).
- dim (wymiar).
- class (klasa).
- length (długość).

Wszystkie atrybuty danego obiektu możemy uzyskać dzięki funkcji `attributes()`.

Przypiszemy nazwy wartościom w wektorze `mix_vect_1`, odpowiadjące jego klasie.

```
names(mix_vect_1)
```

```
## NULL
```

```
names(mix_vect_1)<-c('integer', 'logical', 'complex')
names(mix_vect_1)
```

```
## [1] "integer" "logical" "complex"
```

```
mix_vect_1
```

```
## integer logical complex
## 1+0i 1+0i 2+3i
```