Podstawowe typy danych

Statystyka I z R

Bartosz Maćkiewicz

Podstawowe typy danych

W tej części kursu omówimy podstawowe typy danych z R. Będą to typy: `logical`, `integer`, `numeric`, `character` oraz `factor`.

Na kursie nie będziemy omawiać typu pozwalającego pracować z liczbami urojonymi (typ `complex`). Zainteresowani mogą doczytać o tym typie w dokumentacji R (`help(complex)`)

`logical`

W R elemety wektoru typu `logical` przyjąć mogą dwie wartości - `TRUE` lub `FALSE`. Tak też nazywają się odpowiednie stałe logiczne w R. Zamiast pełnych nazw możemy w kodzie używać skrótów: `T` oraz `F`.

Typ ten jest bardzo ważny i używa się go niemal cały czas kiedy filtrujemy/indeksujemy zbiory danych. Na razie jednak należy zapamiętać tylko to, że w R nazwy wartości logicznych piszemy wielkimi literami.

Dygresja: funkcja `c()`

```
# przypisujemy wartość zmiennej x, funkcja c służy do tworzenia wektorów
x <- c(FALSE, FALSE, FALSE, FALSE, TRUE)
```

W tym fragmencie kodu widzimy dwie bardzo istotne w kontekście pisania kodu w R rzeczy. Po pierwsze, w przeciwieństwie do wielu języków programowania, w R operatorem przypisania wartości zmiennej nie jest znak równośći (`=`) ale strzałka (`<-`). Znak równości również jest interpretowany przez R jako operator przypisania, konwencje stylistyczne mówią jednak, że powinnismy używać strzałki.

Drugą istotną rzeczą w tym fragmencie kodu jest funkcja `c()`. Jej nazwa jest skrótem od angielskiego słowa *combine*. Pozwala ona tworzyć wektory. W R wszystkie podstawowe typy danych są wektorowe. Na nasze potrzeby oznacza to po prostu, że mogą mieć więcej niż jeden element. Funkcja ta pozwala połączyć n wartości w jeden n-elementowy wektor. Funkcja `c()` co do zasady tworzy wektor takiego typu, jakiego były jej argumenty (sprawdź, co się stanie, gdy przekażesz jej kilka argumentów różnych typów!).

```
print(x) # wyświetlamy na ekranie wartość x
```

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

```
class(x) # sprawdzamy jakiego typu jest to zmienna
```

```
## [1] "logical"
```

```
# możemy także stworzyć "pusty" wektor typu logicznego.
# domyslną wartością jego elementów będzie FALSE
logical(5)
```

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

integer

Typ `integer` to po prostu liczby całkowite. R domyślnie traktuje wszystkie liczby nie jako `integer`, lecz jako `numeric`, nawet jeżeli są to liczby całkowite. Możemy to łatwo zobaczyć sprawdzając, jaki ma typ wpisana wybrana przez nas liczba:

```
class(5) # sprawdźmy typ liczby 5
```

```
## [1] "numeric"
```

W niektórych kontekstach R automatycznie przeprowadza konwersje z `numeric` na `int`, czasami jednak tego nie robi. Jeżeli zależy nam na tym, aby nasze pracować z typem `integer`, to możemy użyć suffiksu `L`:

```
class(c(4L, 6L, 8L))
```

```
## [1] "integer"
```

Możemy również *explicite* przekonwertować liczby z jednego typu do drugiego. Służy do tego funkcja `as.integer`.

```
x <- as.integer(c(1,11,2,2,3,3,4,4))
print(x)
```

```
## [1]  1 11  2  2  3  3  4  4
```

```
class(x)
```

```
## [1] "integer"
```

W przypadku użycia funkcji `as.integer` na liczbach, które nie są liczbami całkowitymi, R zaokrągla je w dół:

```
# Uwaga! Gdy konwertujemy numeric do integer R obcina wszystko po kropce (floor)
x <- as.integer(c(1.2, 2.3, 2.8))

# nie musimy używać do wyświetlania instrukcji print(), kiedy wywołujemy samą zmienną R sam to zakłada
x
```

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

```
# można sprawdzić typ danej zmiennej za pomocą funkcji is.{typ}
# funkcje te zwracają wartość TRUE lub FALSE
is.integer(x)
```

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

numeric

Z matematycznego punktu widzenia typowi `numeric` najbliżej do liczb rzeczywistych. Z informatycznego punktu widzenia są to po prostu liczby zmiennoprzecinkowe. Jest to podstawowy typ wartości liczbowych w R i najczęściej będziemy korzystać właśnie z niego.

```
x <- c(1, 2, 3.2, 3.4, 5.2)
x
```

```
## [1] 1.0 2.0 3.2 3.4 5.2
```

character

Standardowe w programowaniu ciągi znaków. W R możemy wybrać, czy chcemu używać pojedynczego (') czy podwójnego (") cudzysłowu.

```
x <- 'Raz' # kiedy nie używamy funkcji c() to również tworzymy wektor
length(x) # o długości 1, co można sprawdzić funkcją length()
```

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

```
y <- c(x, 'Dwa', 'Trzy') # możemy dodawać do siebie wektory za pomocą funkcji c()
y
```

```
## [1] "Raz"  "Dwa"  "Trzy"
```

```
# stwórzmy więc wektor z powtórzonego dwukrotnie wektora x oraz dwukrotnie wektora y
z <- c(x,x,y,y)
z
```

```
## [1] "Raz"  "Raz"  "Raz"  "Dwa"  "Trzy" "Raz"  "Dwa"  "Trzy"
```

```
length(z) # ma on długość 1+1+3+3 czyli 8
```

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

```
class(z) # sprawdźmy, czy rzeczywiście nasza zmienna jest typu character
```

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

```
class(class(z)) # i sprawdźmy jakiego typu wartość zwraca funkcja class
```

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

factor

`factor` jest typem bardzo podobnym do `character`, z tą różnicą, że przechowuje dodatkowe informacje. Istotą tego typu jest to, że przechowuje on nie tylko wartości, ale również informacje o możliwych wartościach danej zmiennej. Z tego względu swietnie nadaje się do przechowywania danych nominalnych takich jak płeć, wykształcenie, status socjoekonomiczny, itp.

Stwórzmy wektor typu `character`, w którym przechowywać będziemy informacje o wykształceniu:

```
# wektor typu character z wykształceniem
edu <- c('Wyższe', 'Średnie', 'Średnie', 'Wyższe', 'Wyższe', "Podstawowe", 'Średnie')
```

Za pomocą funkcji `table` może policzyć, ile wystąpień poszczególnych wartości znajduje się w naszym wektorze:

```
table(edu) # podstawowa tabela z liczbą wystapień
```

```
## edu
## Podstawowe    Średnie     Wyższe
##          1          3          3
```

Dygresja: funkcje `str` i `summary`

W R dostępne są dwie uniwersalne funkcje, za pomocą których możemy uzyskać podstawowe informacje na temat tego, co przechowuje nasza zmienna. Warto o nich pamiętać!

```
# funkcja str zwraca podstawowe informacje o obiekcie,
# który jej przekazujemy w argumencie - tutaj typ, długość i 5 wartości
str(edu)
```

```
##  chr [1:7] "Wyższe" "Średnie" "Średnie" "Wyższe" "Wyższe" "Podstawowe" ...
```

```
summary(edu)
```

```
##    Length     Class      Mode
##         7 character character
```

Z wektora typu `character` możemy stworzyć `factor` za pomocą funkcji `factor`. Funkcja ta przyjmuje trzy argumenty:

```
factor(dane, levels = c("level 1", "level2", ...), ordered = T/F}
```

- `dane` - wektor typu `character` z danymi
- `levels` - wektor typu `character` zawierający poziomy (możliwe wartości) naszej zmiennej
- `ordered` - wektor typu `logical` (`TRUE` lub `FALSE`) mówiący, czy poziomy naszej zmiennej są uporządkowane

```
edu_factor <- factor(edu,
                     levels = c('Podstawowe', 'Gimnazjalne',
                                'Średnie', 'Wyższe'),
                     ordered = TRUE)
```

Kiedy na naszej nowej zmiennej wywołamy funkcję `table` zobaczymy, że w tabeli ze zliczeniami ujęte będzie również wykształcenie gimnazjalne, którego nie ma w naszym wektorze z danymi.

```
table(edu_factor)
```

```
## edu_factor
##  Podstawowe Gimnazjalne     Średnie      Wyższe
##           1           0           3           3
```

Warto również prześledzić, co w przypadku zmiennej typu `factor` zwracają znane nam funkcje `str` i `summary`.

```
# funkcja str mówi nam, że mamy do czynienia
# z uporzadkowanym (ordered) factorem z czterema poziomami
str(edu_factor)
```

```
##  Ord.factor w/ 4 levels "Podstawowe"<"Gimnazjalne"<..: 4 3 3 4 4 1 3
```

```
# a funkcja summary zwraca tabelkę podobną do funkcji `table`
summary(edu_factor)
```

```
##  Podstawowe Gimnazjalne     Średnie      Wyższe
##           1           0           3           3
```