Podstawowe typy i struktury danych w R

Bartłomiej Tartanus

Wektory

Typy atomowe (wektory)

- logiczne (`logical`)
- bajtów (`raw`)
- liczby całkowite (`integer`)
- liczby rzeczywiste (`double`)
- liczby zespolone (`complex`)
- napisy (`character`)
- typ pusty (`NULL`)

Typy rekurencyjne

- lista (`list`)
- funkcja (`function`)
- środowisko (`environment`)

Typy złożone

Reprezentowane za pomocą typów prostych, np:

- ramka danych (`data.frame`)
- czynnik (`factor`)
- macierz (`matrix`)

Tworzenie wektorów

- `c()` - combine
- `rep()` - replicate
- `seq()` - sequence
- : - sekwencja

Obiekty nazwane

Tworzymy przy użyciu operatora przypisania:

- `=`
- `<-`
- `->`
- `<<-`
- `->>`

Typy wektorów

- `typeof()` - typ
- `mode()` - tryb (bardziej ogólne)
- `class()` - klasa obiektu (złożonego)

Uzgadnianie typów

Wszystkie elementy jednego wektora są **tego samego typu**.
W przypadku niejednoznaczności następuje uzgadnianie typów.

```
> typeof(c(TRUE, 1, "A"))
[1] "character"
```

Rzutowanie

Uzgadnianie typów jest niejawnym rzutowaniem.
Na jawne rzutowanie pozwalają funkcje:

```
as.character()
as.double()
as.integer()
as.logical()
as.complex()
```

Wartości nietypowe

`NA` – not available
`NaN` – not a number
`NULL` – wartość pusta
`Inf` – +infinity
`-Inf` – -infinity

Operacje na wektorach

Wszystko (prawie) jest wektorem

W R nie ma pojedynczych wartości (skalarnych). Są one reprezentowane przez wektor jednoelementowy.

Wektoryzacja - unikamy pętli
Wektory różnej długości? Reguła zawijania

Operacje zwektoryzowane

-	minus	%in%	wyszukiwanie
+	plus	<	mniejsze
!	negacja	>	większe
*	mnożenie	==	równe
/	dzielenie	<=	mniejsze/równe
^	potęgowanie	>=	większe/równe
%%	dzielenie modulo	&	"i" logiczne
%/%	dzielenie całkowite	&&	
%*%	mnożenie macierzowe	|	"lub" logiczne
%o%	iloczyn diadyczny	||	

Funkcje zwektoryzowane

```
abs()
sign()
floor()
ceiling()
round()
sqrt()
exp()
log()
sin()  asin()  sinh()
cos()  acos()  cosh()
tan()  atan()  tanh()
```

```
Conj()
Re()
Im()
Mod()
Arg()
```

Agregacja wektorów

```
sum()
prod()
mean()
median()
min()
max()
var()
sd()
quantile()
any()
all()
```

Operacje na sąsiednich elementach

```
> cumsum(1:5)
[1]  1  3  6 10 15
> cumprod(1:5)
[1]   1   2   6  24 120
>
cummin(c(5,2,6,1,7,0))
[1] 5 2 2 1 1 0
>
cummax(c(5,2,6,1,7,0))
[1] 5 5 6 6 7 7
> diff(c(5,2,6,1,7,0))
[1] -3  4 -5  6 -7
```

Indeksowanie/filtrowanie

Operator `[` działa na cztery sposoby, w zależności od przekazanego wektora:

```
> x <- 0:10; x[y]
```

y - liczbowy, dodatni (**indeksujemy od 1**)

```
> x[c(2,5)]
[1] 1 4
```

Indeksowanie/filtrowanie

y - liczbowy, ujemny

```
> x[c(-2,-5)]
[1]  0  2  3  5  6  7  8  9 10
```

y - logiczny

```
> x[c(TRUE, FALSE)]
[1]  0  2  4  6  8 10
```

Indeksowanie/filtrowanie

y - napisy (wektor musi mieć nazwane elementy)

```
> x <- c(ala=5, janek=12)
> x
ala janek
  5    12
> x["ala"]
ala
  5
```

Modyfikacja elementów

```
> x <- 1:4
> x
[1] 1 2 3 4
> x[1:2] <- 50
> x
[1] 50 50  3  4
```

Modyfikacja nieistniejących elementów

```
> x <- 1:5
> x[7]
[1] NA
> x[7] <- 7
> x
[1]  1  2  3  4  5 NA  7
```

O wydajności słów kilka

Wystrzegać sie takich operacji:

```
> x <- 1:5
> x[length(x)+1] <- 50
```

Są one **WOLNE** - tworzony jest nowy wektor rozmiar większy, wszystkie "stare" elementy są przepisywane i na koniec dopisywana jest nowa wartość

Wyszukiwanie indeksów

Funkcja `which` zwraca indeksy wszystkich elementów `TRUE` w wektorze logicznym.

```
> 10:15 > 12
[1] FALSE FALSE FALSE  TRUE  TRUE  TRUE
> which(10:15 > 12)
[1] 4 5 6
```

Permutowanie elementów

```
> x <- c(5,8,12,4,1)

> sort(x)
[1]  1  4  5  8 12

> order(x)
[1] 5 4 1 2 3

> x[order(x)]
[1]  1  4  5  8 12
```

```
> rank(x)
[1] 3 4 5 2 1

> rev(x)
[1]  1  4 12  8  5

> sample(x)
[1]  5  1  8  4 12
```

Operacje na zbiorach

- `union()` - suma zbiorów
- `intersection()` - przecięcie
- `setdiff()` - różnica zbiorów
- `setequal()` - równość zbiorów
- `is.element()` - czy należy do zbioru
- `unique()` - usuwa duplikaty
- `duplicated()` - wskazuje duplikaty
- `anyDuplicated()` - wskazuje pierwszy duplikat (jeśli jest)

Listy

Tworzenie listy

```
> x <- list(TRUE, 1:5,
"abc", pi)
> x
[[1]]
[1] TRUE

[[2]]
[1] 1 2 3 4 5

[[3]]
[1] "abc"

[[4]]
[1] 3.141593
```

```
> str(x)
List of 4
 $ : logi TRUE
 $ : int [1:5] 1 2 3 4 5
 $ : chr "abc"
 $ : num 3.14
```

Indeksowanie listy

```
>x[1]
[[1]]
[1] TRUE

> str(x[1])
List of 1
 $ : logi TRUE
```

```
> x[[1]]
[1] TRUE
> str(x[[1]])
 logi TRUE
> x[[2]]
[1] 1 2 3 4 5
> str(x[[2]])
 int [1:5] 1 2 3 4 5
```

Funkcje na listach

```
> x <- list(1:5, 2:8, 4)
> sum(x)
Błąd wsum(x) : niepoprawny
'type' (list) argumentu
> lapply(x, sum)
[[1]]
[1] 15

[[2]]
[1] 35

[[3]]
[1] 4
```

```
> x[[1]]
[1] TRUE
> str(x[[1]])
 logi TRUE
> x[[2]]
[1] 1 2 3 4 5
> str(x[[2]])
 int [1:5] 1 2 3 4 5
```

Generowanie losowych wektorów

```
[dpqr](beta|binom|cauchy|chisq|exp|f|gamma|geom|hyper|lnorm|nbinom|norm|pois|t|unif|weibull)
```

Generowanie losowych wektorów

dxxx density (gęstość)
pxxx cumulative distribution (dystrybuanta)
qxxx quantile (odw. dystr.)
rxxx random (losowy wektor)

beta	beta
binom	dwumianowy
cauchy	Cauchy'ego
chisq	Chi-kwadrat
exp	wykładniczy
f	F-Snedecora

gamma	gamma
geom	geometryczny
hyper	hipergeometryczny
lnorm	log-normalny
multinom	wielomianowy
nbinom	ujemny dwumianowy
norm	normalny
pois	Poissona
t	t-Studenta
unif	jednostajny
weibull	Weibulla