Statystyczna Analiza Danych
zajęcia laboratoryjne 2

Elementy środowiska i składania R

Bioinformatyka II rok

1 Indeksy/indeksowanie

Do elementów wektora, list, macierzy czy ramek, można się odwołać na kilka sposobów:

- **Notacja wektorowa** zmienna[zakres]. Przez zmienną rozumie się listę, wektor, macierz lub ramkę danych. Zakres może być wektorem liczb całkowitych lub jedną liczbą całkowitą. W przypadku indeksowania list, wektorów, macierzy czy ramek, możemy w nawiasach kwadratowych [] podać wektor nazw elementów. Ważne! Nie można w zakres mieszać indeksów dodatnich i ujemnych.

```
wektor <- 1:10
wektor[1:3] #trzy pierwsze elementy wektora
wektor[c(-1,-2,-5)] #wektor bez pierwszego, drugiego i piatego elementu

#indeksy nie musza byc liczbami, moga to byc nazwy elementow
wektor <- c(a = 1, b = 2, c =3)
wektor[c("a","b")]
wektor[length(wektor)] #ostatni element
```

- **Notacja macierzowa** zmienna[zakres1, zakres2]. Zmienna jest macierzą lub ramką danych. Wybierana jest pod-macierz o wskazanych indeksach.

```
macierz <- matrix(1:4,2,2)
macierz[1, ] #1 wiersz macierzy, zgubiony jest wymiar macierzy (wynik jest wektorem)
macierz[1, , drop = T] #wynik jest wektorem
macierz[1, , drop = F] #wynik jest macierza
macierz[-1, -1] #elementy poza pierwszym wierszem i pierwsza kolumna
```

- **Notacja listowa** zmienna$wartosc1. Zmienna to lista, wektor lub ramka danych. Wynikiem zastosowania tego operatora jest element listy lub wektora, lub kolumna z ramki danych o nazwie wartosc1.

```
osoba <- list(name = c("Anna","Joanna"), surname = "Kowalska", age = 25, married = T)
osoba$name
osoba[[2]]
```

- **Notacja nawiasowa** zmienna[[indeks1]]. Zmienna to wektor, lista, macierz lub ramka danych. Zwracany jest jeden element (jeden indeks). Indeks nie może być wektorem. Indeksowanie w ten sposób jest najczęstsze dla list. Ramka danych również jest listą, co oznacza, że ten sposób doprowadzi do wybrania wskazanej KOLUMNY, podczas gdy w przypadku macierzy doprowadzi do wybrania JEDNEJ wartości.

```
macierz <- matrix(1:4, 2, 2)
macierz[[2]] #macierz jest tak naprawde wektorem
macierz[[2, 2]]
osoba <- list(name = c("Anna","Joanna"), surname = "Kowalska", age = 25, married = T)
osoba[[1]] #1 element listy jest wektorem dwuelementowym
osoba[[c(1,2)]]
```

Przydatne funkcje:

- Funkcja which() - przydatna przy operacjach na indeksach. Wynikiem funkcji są indeksy elementów spełniające zadany warunek logiczny.

```
wektor = c(1,3,6,10,10.5,12)
which(wektor == 10.5)
which(wektor < 10)
```

- Funkcja which.min() i which.max() - zwracają indeks elementu minimalnego i maksymalnego. Zastanów się co jest wynikiem działania funkcji which dla następującego przykładu:

```
which.max(c(F,F,T,F,T))
```

- Funkcja match() - wynikiem są indeksy wystąpień wektora elementów w innym wektorze:

```
match(c(3,10,10.5), wektor)
```

Indeksowanie służy nie tylko do wyświetlania konkretnych danych, ale także do ich modyfikacji:

- modyfikacje wektora:

```
x <- 1:10
x[1] <- 4 #pierwszy element wektora x, ma teraz wartosc 4
x[] <- 0 #zero zastapi wszystkie elementy
x[13] <- 77 #dodanie nowej wartosci dla indeksu 13, indeksy 11 i 12 maja wartosc NA
```

- modyfikacje listy:

```
x <- list(1,2,3)
x[[1]] <- c(5,10) #pierwszy element podmieniono na dwuelementowy wektor
x[[5]] <- 22 #rozszerzenie listy
```

Wykorzystując powyższy przykład przetestuj i zaobserwuj różnicę pomiędzy instrukcjami:
x[1]<−NULL, a x[1]<−list(NULL).

2 Sekwencje liczb

Sekwencje to regularne wektory liczb całkowitych (ciągi arytmetyczne o dowolnym kroku). Do wygenerowania sekwencji można użyć funkcji seq() lub operatora ":":

- użycie operatora ":" oznacza, zakres od - do z krokiem 1:

```
−3:3
```

- użycie funkcji seq(), np. seq(5), oznacza wyświetl liczby od 1 do 5, czyli jest równoważne z użyciem operatora 1:5. Dodatkowo w funkcji seq() można podać zakres wektora od - do, oraz zdefiniować krok:

```
seq(5)
seq(5,15)
seq(10,20,by = 2)
seq(10,1,-2)
```

Operując na sekwencjach liczb, przydatna jest funkcja sample(). Funkcja ta losuje n-elementowy podzbiór z zadanego wektora. Argumenty funkcji sample: pierwszym argumentem jest wektor (np. *letters* to wektor małych liter z alfabetu romańskiego, drugim argumentem jest liczba losowo wygenerowanych liter z podanego wektora, a trzeci argument *replace* określa czy losujemy elementy ze zwracaniem (replace = T) czy bez (replace = F):

```
sample(letters,10,T)
sample(1:5,10,T)
```

3 Obiekty: wyświetlanie i formatowanie

- Funkcja cat() i print() czyli wyświetlenie sformatowane i niesformatowane:

```
napis = rep(c("Herbata stygnie,", "zapada zmrok,", "a pod piorem ciagle nic."), 2)
print(napis)
cat(napis)
```

Dodatkową różnicą tych funkcji jest fakt, że funkcję print() można przeciążać. Co oznacza, że można samemu określić jak mają zostać wyświetlone obiekty różnych klas. O przeciążaniu więcej będzie później.

- Funkcja format() stosowana przy konwersji danego obiektu na typ znakowy (zgodnie z ustalonym formatowaniem):

```
format(10/3)
format(10/3, digits = 2) # wyswietlenie z max. 2 cyframi znaczacymi
format(10/3, sci = TRUE) # notacja naukowa
```

- Funkcja paste() łączy wektory napisów:

```
paste("Ala", "ma", "kota")
```

- Inne funkcje do formatowania: sprintf(), toString(), encodeString(), formatFix(), iconv().

4 Instrukcje warunkowe i pętle

- Instrukcja warunkowa if ... else:

```
if(warunek)
    instrukcja1

if(warunek)
    instrukcja1 else instrukcja2

if(warunek){
    instrukcja1
}else{
    instrukcja2
}
```

Zwróć uwagę na różny sposób zapisu (z klamrami i bez):

```
if(4%%2==0) cat("parzysta") else cat("nieparzysta")
if(4%%2==0){cat("parzysta")}else{cat("nieparzysta")}
```

Jeśli piszemy kod, który ma być czytelny to stosujmy wcięcia jak w innych językach programowania, w takiej składni jest bardzo ważna zasada - słowo kluczowe else nie może rozpoczynać się w nowej linii. Pamiętaj, że R jest językiem interpretowanym! Interpreter nie spodziewa się kolejnych instrukcji. Wyjątkiem jest stosowanie takiej składni w ciele funkcji.

```
if(4%%2==0){
    cat("parzysta")
}else{
    cat("nieparzysta")
}
```

- Funkcje ifelse(): argumentem funkcji ifelse() jest wektor warunków logicznych, składnia:

```
ifelse(warunek, instrukcja1, instrukcja2)
```

W powyższej składni:
#warunek może być 1 wartością logiczną lub wektorem wartości logicznych
#wynik będzie miał wartości opisane przez instrukcja1 w pozycjach odpowiadających wartości TRUE
#wynik będzie miał wartości opisane przez instrukcja2 w pozycjach odpowiadających wartości FALSE

```
ifelse(1:10 < 5, "mniej", "wiecej")
```

- Funkcja switch(), składnia:

```
switch(klucz, wartosc1 = akcja1, wartosc2 = akcja2, ...)
```

Pierwszy argument powinien być typu znakowego lub wyliczeniowego. W zależności od wartości tego argumentu, wynikiem jest wartość zwrócona przez odpowiednią akcję.

```
x <- switch(3, "first", "second", "third", "fourth")
print(x)
switch("color", "color" = "red", "shape" = "square", "length" = 5)
switch(2, "color" = "red", "shape" = "square", "length" = 5)
```

Inna wersja instrukcji switch-case jest obsługiwana przez funkcję switchCase(), wymaga ona jednak zainstalowania odpowiedniego pakietu, zobacz w helpie jak użyć tej funkcji.

```
install.packages("switchcase")
library(switchcase)
help(package = "switchcase")
?switchCase()
```

- Pętla for: pętla wykonuje się tyle razy ile jest elementów w obiekcie kolekcja:

```
for(iterator in kolekcja){
    instrukcje
}
```

Przykład 1:

```
for(i in 1:10){
  cat(paste("wartosc:", i, "\n"))
}
```

Przykład 2 (określony krok):

```
for(i in seq(1,10, by=2)){
  cat(paste("wartosc:", i, "\n"))
}
```

Przykład 3:

```
wektor <- c("czerwony", "zielony","niebieski")
for (i in wektor){
  cat(paste(i,"\n"))
}
```

Wyznaczanie wektora indeksów pętli za pomocą funkcji seq_along(), argumentem tej funkcji jest wektor dowolnych wartości, a wynikiem wektor kolejnych liczb naturalnych (od 1)

```
for(i in seq_along(wektor)){
  cat(paste("indeks", i, "to element", wektor[i], "\n"))
}
```

- Pętla while:

```
while(warunek logiczny){
    instrukcje
}
```

Przykład 1:

```
i <- 0
while(i < 5){
  print(i)
  i <- i + 1
}
```

R umożliwia pisanie komend w jednej linii, jednak zwróć uwagę, że pojawia się wtedy potrzeba używania średników pomiędzy kolejnymi instrukcjami!

```
i <- 0; while(i < 5){print(i); i <- i + 1}
```

Pętla while() przerwana swoje działanie przez określony warunek.

- Pętla repeat:

  Pętla, która nie posiada warunku stop. Jej działanie zostaje przerwane w przypadku wystąpienia błędu lub instrukcji break.

```
repeat{
    instrukcje
}
```

Przykład 1:

```
x <- 1
repeat{
  print(x)
  x = x + 1
  if (x == 3){
    break
  }
}
```

break - przerywa działanie pętli

next - przerwanie wykonywania aktualnej iteracji i przejście do kolejnej

5 Zadania

Rozwiąż poniższe zadania, a utworzony skryptu .R zostaw do okazania pod koniec zajęć.

Zad. 1

Wykorzystaj funkcję seq() aby wyświetlić wektor 10 liczb w zakresie od 10 do 25. Funkcja seq() powinna sama ustawić krok.

Zad. 2

Przetestuj funkcję sample(). Jaki jest domyślny trzeci argument tej funkcji?

Zad. 3

Wylosuj wektor 20 liczb, liczby mają być losowane z zakresu od $1:3$, z określonymi prawdopodobieństwami wylosowania np. $(0.1, 0.3, 0.6)$.

Zad. 4

Wyświetlanie w sposób sformatowany i niesformatowany, do zmiennej napis przypisz wektor zawierający 3 zdania "ala ma kota", "ola nie ma kota", "ela ma psa". Zreplikuj wektor trzykrotnie, a następnie użyj funkcji print i cat do wyświetlenia zmiennej. Co zaobserwowałeś?

Zad. 5

Wykorzystaj instrukcje warunkowe, aby sprawdzić czy liczba 1515 jest parzysta. Jeśli liczba jest parzysta to wyświetl stosowny komunikat, oraz jeśli liczba jest nieparzysta to również wyświetl stosowny komunikat.

Zad. 6

Wykorzystaj funkcję switch(), aby sprawdzić jaka jest klasa danej zmiennej. Zadeklaruj zmienną liczba o wartości 1515. Jeśli zmienna jest klasy factor to wyświetl "typ czynnikowy", jeśli zmienna jest klasy logical to wyświetl "typ logiczny", jeśli zmienna jest klasy numeric to wyświetl "typ liczbowy". Jeśli wartość naszego klucza nie pasuje do etykiety, wykonaj akcję domyślną i wyświetl "inny typ".

Zad. 7

Wyświetl na ekranie wszystkie liczby całkowite podzielne przez 3 należące do zadanego przedziału: $[3, 333]$.

Zad. 8

Napisz program znajdujący wszystkie dzielniki liczby 33.

Zad. 9

Wyznacz 15 kolejnych liczb Fibonacciego. Podaj numer indeksu dla którego wartość jest równa 233.

Zad. 10

Utwórz ramkę danych, która w pierwszej kolumnie przechowuje id od 1 do 10, w drugiej kolumnie przechowuje 10 losowych wartości z zakresu $[1:10]$ bez zwracania, w trzeciej kolumnie przechowuje 10 losowych wartości z zakresu $[10:20]$ ze zwracaniem, następnie w pętli zwróć wartość średnią dla kolumny 2 i 3.