Statystyczna Analiza Danych
zajęcia laboratoryjne 1

Wprowadzenie do R

Bioinformatyka II rok

1 Krótko o R

R jest wolnym (otwartym i darmowym), zaawansowanym środowiskiem oraz językiem programowania. R jest często nazywany pakietem statystycznym, z uwagi na liczbę dostępnych funkcji statystycznych (możliwości R są jednak znacznie większe).

2 Cechy języka

- składnia przypomina trochę C/C++,
- jest językiem ogólnego zastosowania (można w nim zaimplementować praktycznie każdy algorytm),
- zwięzła składnia (krótki kod, złożony rezultat),
- jest językiem interpretowanym, a nie kompilowanym
- można generować grafiki, wykresy, diagramy i inne,
- tryb interaktywny i tryb wsadowy,
- posiada repozytorium CRAN (Comprehensive R Archive Network), gdzie znajdują się tysiące pakietów rozszerzających R.

3 Organizacja pracy – R i RStudio

- Instalacja R: https://cran.r-project.org/
- Instalacja RStudio: https://www.rstudio.com/products/rstudio/download/
- RStudio ma pewną przewagę nad R:
  - ułatwia pracę dzięki licznym rozszerzeniom możliwości konsoli,
  - wygodne zarządzanie plikami źródłowymi, w tym projektami,
  - zintegrowany system pomocy, narzędzia wspomagające,
  - obsługa systemów kontroli wersji,
  - w RStudio pracuje się w obszarze roboczym, można tworzyć projekty,
  - system podpowiedzi (Tab).

4 Praca w R

4.1 Podstawowe znaki

Wyróżniamy kilka podstawowych znaków:

- ">" znak zachęty (command prompt) oznacza, że platforma R jest gotowa do realizacji polecenia,
- "+" znak kontynuacji, znak oczekujący na dokończenie polecenia,
- "<−" lub "=" symbol przypisania. W większości zastosowań operatory = i <- można stosować zamiennie. Jednak nie zawsze mają one to samo działanie. Operator <- ma wyższy priorytet.
- "\n" znak sterujący, znak nowej linii
- "#" znak komentarza

Problem z operatorami przypisania <- i = przedstawiono na dwóch poniższych przykładach. W pierwszym przypadku operator "=" wskazuje, który argument funkcji jest określany, nie przypisuje wartości:

```
1 plot(liczby = 1:10)
```

W drugim przypadku operator "<-" przypisuje wartość do zmiennej

```
1 plot(liczby<-1:10)
```

Pierwsze uruchomienie R i wyświetlenie informację o wersji środowiska:

```
1 R.version
2 R.version.string
3 getRversion()
```

4.2 Instalowanie i ładowanie pakietów

Zainstalowanie podstawowego zbioru bibliotek daje nam już duże możliwości. A jeszcze większe możliwości zapewniają dodatkowe pakiety w których znajdują się różne funkcje (funkcje w R pogrupowane są w pakietach/bibliotekach). Instalowanie pakietu:

```
1 install.packages()
```

Po zainstalowaniu nowego pakietu, pliki z danymi, funkcjami i plikami pomocy znajdą się na dysku twardym komputera. Jednak wszystkie pakiety są wgrywane jako podkatalogi do katalogu library. Chcąc skorzystać z wybranych funkcji należy przed pierwszym użyciem załadować odpowiedni pakiet:

```
1 library()
```

4.3 Pomoc w R

Czyli zapoznanie się z funkcją help() i help.search(). Funkcja help() wyświetla stronę z pomocą dla funkcji o podanej nazwie, w argumencie funkcji help podajemy nazwę. Zamiast nazwy funkcji help(), możemy podać znak zapytania:

```
1 help("sd")
2 ?sd
```

Funkcja help.search() jest wykorzystywana do szukania słów kluczowych. Podobnie jak wyżej w argumencie funkcji help.search() podajemy szukaną frazę. Zamiast nazwy funkcji help.search(), możemy podać dwa znaki zapytania:

```
1 help.search("standard deviation")
2 ??"standard deviation"
```

Funkcji help()/? używasz, gdy wiesz dokładnie, czego szukasz (np. nazwy funkcji, pakietów), a drugiej funkcji help.search() używasz, gdy wiesz w przybliżeniu, czego szukasz i chcesz znaleźć słowa kluczowe.

Wykorzystaj powyższe informacje aby zainstalować i załadować pakiet MASS. Sprawdź przy użyciu help i help.search funkcję data(). Sprawdź jak ją wykorzystać, aby załadować zbiór danych "cats". Narysuj wykres dla wybranego zbioru danych używając funkcji plot(cats).

5 Podstawowe elementy R

5.1 Podstawowe elementy składni R

Do podstawowych elementów składni zaliczamy typy i obiekty. Wszystko czym można operować w języku R jest obiektem. Obiekty można podzielić na kilka typów:

- typ liczbowy (numeryczny): liczby całkowite, rzeczywiste, wartość NaN, literały Inf i -Inf.
- typ czynnikowy (wyliczeniowy/kategoryczny): typ do przechowywania wektorów wartości. Występujących w kilku kategoriach (kilku poziomach).
- typ znakowy: wartościami obiektów są napisy/łańcuchy znaków.
- typ logiczny: FALSE i TRUE.
- wektor elementów: zbiór obiektów tego samego typu, funkcja c(). Przykład:

```
1 c(1:10)
```

- lista: zbiór obiektów różnego typu, funkcja list(). Przykład:

```
1 list(imie = "Jan", nazwisko = "Kowalski", wiek = 25)
```

- macierz i tablica: funkcja matrix() i array().
- ramka danych/tabela danych: elementy w każdej kolumnie są tego samego typu, ale typy mogą być inne pomiędzy kolumnami, funkcja data.frame(). Przykład:

```
1 ramka <- data.frame(id = c(1:3), wiek = c(21:23), czyKobieta=c(T,T,F))
2
3 #Odwolanie do poszczegolnych danych w ramce
4
5 ramka$wiek
6 ramka[ ,"wiek"]
7 ramka[ ,2]
8 ramka[2]
9 ramka[2, ]
```

- typ funkcyjny: wykorzystuje się słowo kluczowe function.

5.2 Bardziej szczegółowy podział typów w języku R

- typy podstawowe
  a) typy atomowe
     - logical - wektor wartości logicznych
     - raw - wektor bajtów
     - integer - wektor wartości całkowitych
     - double - wektor wartości rzeczywistych

  - complex - wektor wartości zespolonych
  - character - wektor napisów
  - NULL - typ pusty

b) o strukturze rekurencyjnej
  - list - lista, wektor uogólniony
  - closure/function - funkcja
  - environment - środowisko

c) typy reprezentujące nieobliczone wyrażenia języka R
  - name - nazwa
  - call - wywołanie
  - expression - ciąg wyrażeń (zaliczany również do typów rekurencyjnych)

- typy złożone (reprezentowane przy użyciu obiektów typów podstawowych)
  - klasa matrix, array - macierz i tablica
  - klasa ts - szereg czasowy
  - klasa factor - czynnik
  - klasa data.frame - ramka danych
  - klasa formula – formuła

Typ każdego obiektu można poznać wykorzystując funkcję typeof():

```
1 c(typeof(TRUE), typeof(5), typeof("tekst"))
```

Przetestuj powyższą instrukcję ponownie, ale użyj funkcji mode() zamiast typeof().

Hierarchia i uzgadnianie typów, poniżej przedstawiono wektor zawierający elementy różnych typów:

```
1 c(TRUE, 1L, 5.5, 3.0i, 3, "siedem")
```

Wynikiem jest wektor napisów. R uzgadnia typ, tak aby informacje o najbardziej ogólnym z obiektów dało się przechować bez znaczącej straty. Np. konwersja liczb rzeczywistych na napisy może prowadzić do zaokrąglenia wartości. Uzgodnienie typów w R zwane jest koercją.

Hierarchia typów (od najbardziej do najmniej szczegółowych):

- typ logical
- typ integer
- typ double
- typ complex
- typ character

5.3 Konwersja / Rzutowanie typów

Typ zmiennej nie jest przypisany do zmiennej (czy do jej wartości) na stałe. Możemy zmieniać typy bez podawania nowej wartości dla tej zmiennej. Jednak w przypadku rzutowania, zgadzamy się na ewentualną utratę informacji (w przypadku rzutowania z typu mniej do bardziej szczegółowego). Najczęstsze konwersje to zamiana na typ znakowy lub na typ liczbowy:

```
1 as.character()
2 as.character(c(TRUE,FALSE))
3 as.complex(c(TRUE,FALSE))
4 as.numeric()
```

Konwertować można pojedyncze wartości jak również złożone struktury, jak np. lista. Można także sprawdzić jakiego typu jest dana zmienna, np.:

```
is.integer()
```

5.4 Operatory arytmetyczne i logiczne

Operatory arytmetyczne i logiczne są podobne jak w języku C, proszę jednak zwrócić uwagę na dzielenie modulo:

```
-x                  #zmiana znaku x
x + y               #suma
x - y               #roznica
x * y               #iloczyn
x / y               #iloraz
x ^ y               #liczba x do potegi y
x %% y              #dzielenie modulo (reszta z dzielenia x przez y)
x %/% y             #czesc calkowita z dzielenia x przez y
<,==,>,<=,>=, !=    #standardowe operatory porownania wartosci liczbowych
!                   #operator negacji
&,&& ,| ,||         #logiczny iloczyn i logiczna suma
any(), all()        #"ktorykolwiek" i "wszystkie"
```

Operatory & i | służą do wykonywania operacji na listach i wektorach (porównują wszystkie elementy zadanych wektorów i ich wynikiem może być wektor o długości większej niż 1), natomiast operatory && i || na pojedynczych wartościach (porównują tylko pierwszy element każdego wektora, zawsze zwracają tylko jedną wartość). Natomiast operatory any(), all() sprawdzają czy wszystkie lub którykolwiek z elementów obiektu przyjmuje wartość TRUE.

5.5 Wektory

Do utworzenia wektora wykorzystuje się funkcję c() (ang. combine). Przetestuj poniższe instrukcje:

```
#utworzenie wektora
wektor <- c(1, 2, 3.5, -4, 0)

#wykonywanie podstawowych operacji:
wektor^2
wektor-2

#sprawdzenie dlugosci wektora (ilosci elementow)
length(wektor)
```

Dodatkowo wektory można łączyć (poniżej wektor wartości logicznych):

```
c(c(TRUE,FALSE), c(TRUE,FALSE))
```

Oprócz funkcji c() istnieje jeszcze rep() i operator ":". Można replikować wektory, funkcją rep() (ang. replicate). Natomiast dwukropek generuje ciągi arytmetyczne z krokiem 1 lub -1:

```
rep(TRUE, 3)
rep(1:2, times=5)
rep(1:2, each=5)
c(-3:3, 7:1)
```

Do generowania ciągów arytmetycznych o dowolnym kroku służy funkcja seq(), która pojawi się w kolejnym materiale.

Odwoływanie do poszczególnych elementów wektora i manipulacje na wybranym fragmencie wektora:

```
wektor[3]
wektor[1:2]
```

```
3 wektor[c(1,3)]
4 wektor[3]-2
5 wektor[wektor>0]
```

5.6 Macierze

Macierze, czyli funkcja matrix() i podstawowe działania:

Utworzenie macierzy zawierającej same 0, macierz 2x3

```
1 macierz <- matrix(0,2,3)
```

Podobnie jak na wektorach można wykonywać operacje arytmetyczne

```
1 macierz + 1
```

Wypełnienie macierzy elementami od 1 do 6

```
1 macierz <- matrix(1:6,2,3)
```

Wyświetlenie kolumny pierwszej i wiersza pierwszego:

```
1 macierz[,2]
2 macierz[2,]
```

Mnożenie macierzy poprzez operator %*%

5.7 Wartości specjalnego znaczenia

- Brak danych, czyli NA (ang. not available). NA nie należy mylić z NULL (zobacz w R: ?NA)
- Nie-liczba: wartość nieokreślona NaN (ang. not a number)
- Wartości nieskończona Inf (ang. infinity)

6 Zadania

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

Zad. 1

Przypisz dwóm zmiennym wartości i przetestuj operator modulo oraz zwróć całkowitą wartość dzielenia.

Zad. 2

Sprawdź jak to działa, jakie wartości mają zmienne a i b? Dlaczego jest błąd?

```
1 a = b = 3
2 a <- b <- 3
3 a = b <- 3
4 a <- b = 3
```

Zad. 3

Do zmiennej x przypisz wartość 7, sprawdź czy jest typu integer. Przetestuj funkcje mode() i typeof().

Zad. 4

Wykorzystaj R jako precyzyjny kalkulator. Przetestuj funkcje sqrt(), log(), sin(). Wykorzystaj funkcję log(), aby wyliczyć logarytm naturalny i logarytm dziesiętny z przykładowych liczb.

Zad. 5

Zadeklaruj wektor o wartościach 1, 10, NaN. Wykorzystaj funkcję, która sprawdzi czy w wektorze występuje wartość NaN. Jaka to funkcja?

Zad. 6 (zadanie dodatkowe)

Zapoznaj się z przydatnymi funkcjami: help.search(), q(), demo(), args(), apropos(), example().