

Laboratorium 1 – Wprowadzenie do pakietu R

1. Źródła:

- | | |
|---|---|
| https://www.r-project.org | - Oficjalna strona projektu R |
| https://www.cran.r-project.org | - Dokumentacja i pakiety dla projektu R |
| https://cloud.r-project.org | - Odnosiniki do pakietów instalacyjnych |
| https://cloud.r-project.org/bin/windows/base/ | - wersja instalacyjna dla Windows |
| WWW.rstudio.com/products/rstudio/ | - adres do instalacji RStudio |

2. Ważne elementy pakietu R:

- kompilator i interpreter
- GUI, podstawowe środowisko graficzne, konsola
- RStudio, rozbudowane środowisko graficzne zapewnia: obsługę plików źródłowych, konsolę, edycję plików źródłowych i kontrolę syntaktyczną, śledzenie stanu przetwarzania, prezentacja grafiki (wykresy, diagramy, etc.) tworzonych przez skrypty
- pakiety zawierające funkcje i makra możliwe do ładowania ze środowiska RStudio
- zbiory danych do ćwiczeń i przykładów

3. Podręczniki:

Podręcznik interaktywny w projekcie R

<https://cran.r-project.org/doc/manuals/r-release/R-intro.html>

Polskie publikacje niedostępne w sieci:

[Podstawy statystyki z przykładami w R](#). Autor: Tomasz Górecki, Wydawnictwo BTC 2011, ISBN 978-83-60233-69-6, do kupienia BCT

<https://wydawnictwo.btc.pl/matematyka/178794-podstawy-statystyki-z-przykladami-w-r.html>, 89,10 PLN, e-book 64 PLN

[Programowanie w języku R. Analiza danych, obliczenia, symulacje](#). Autor: Marek Gągolewski, Wydawnictwo Naukowe PWN 2014 (wydanie pierwsze), 2016 (wydanie drugie poszerzone), ISBN 978-83-01-18939-6

[Przewodnik po pakiecie R](#). Autor: Przemysław Biecek, Oficyna Wydawnicza GiS 2008 (wydanie pierwsze), 2011 (wydanie drugie), 2014 (wydanie trzecie), 2017 (wydanie czwarte), ISBN 978-83-89020-79-6, <http://www.eksiegarnia.pl/Przewodnik-po-pakiecie-R;s,karta,id,392249>, 52,80 PLN, Początkowe rozdziały dostępne w sieci.

Publikacje dostępne w sieci:

Łukasz Komsta; Wprowadzenie do środowiska R

John Verzani; Using R for introductory statistics

John Verzani; simple R - Using R for introductory statistics

John Maindonald and W. John Braun; Data Analysis and Graphics using R

John Maindonald; Using R for data analysis and graphics

Petra Kuhnert and Bill Venables; An Introduction to R: Software for Statistical Modelling & Computing. Course materials and exercises.

W. N. Venables, D. M. Smith; An introduction to R.

4. Podstawowe komendy R i środowiska RStudio

`setwd(<path>)` ustawia kartotekę roboczą

`getwd()` zwraca kartotekę roboczą w której zapisywane są dane aktualnego przetwarzania

`quit()` lub `q()` zatrzymuje pracę środowiska

`source („<nazwa pliku skryptu>”)` uruchamia skrypt R. Nazwa może być względem kartoteki aktualnej jaką pamięta interpreter.

Konsola w RStudio wygląda podobnie jak w oryginalnym GUI. Znak promptu `>` nie jest czerwony tylko niebieski. W takim samym kolorze pisane są komendy. Na konsoli drukowane są wyniki. Linie z wynikami rozpoczynają się od liczby w nawiasach kwadratowych [1], [7]. Numer ten oznacza indeks pierwszej wartości w danej linii wydruku. Jeżeli po wpisaniu i próbie wykonania jakiejś komendy z konsoli (również RStudio) pojawi się znak `+`, oznacza to, że komenda ma być w tej linii uzupełniona (dokończona).

5. Pomoc w R

`man <nazwa>` podaje informacje o obiektach R w dodatkowym okienku
`help()`; `help („<nazwa>”)`, `?<nazwa>` wyświetla stronę tytułową systemu pomocy; wyświetla informacje o obiekcie `nazwa`.
w szczególności `help(package=<nazwa>)` wyświetla opis pakietu o podanej nazwie.
`apropos („<string>”)`, `find („<string>”)` wyświetlają listę funkcji w nazwie pojawia się `string`.
`F1` uruchamia pomoc w RStudio, należy wybrać repozytorium i znać listę nazw pakietów.

6. Nieco o pakietach

Ładowane pakiety możemy przejrzeć w prawym dolnym oknie RStudio, obcja Packages. Pakiety możemy instalować w RStudio opcja Tools -> Install packages. Pakiety możemy instalować z repozytorium CRAN, lub zmieniając obcję na Packages archive, z pliku archiwum.
Pakiety włączamy na początku skryptu komendą `library („<packagename>”)`
Napis `<packagename>::<funcname>` gwarantuje użycie wersji funkcji publicznych z podanego pakietu. Wyrażenie `<packagename>::<funcname>` daje możliwość odwołania się również do funkcji prywatnych z podanego pakietu.
Część większych pakietów jest w repozytorium GitHub. Można je łąadować instrukcją `devtools::install_github („<packagename>”)`. Należy mieć doinstalowany wcześniej pakiet `devtools`
Możemy poszukać funkcji w zainstalowanych pakietach komendą `help.search („<packagename>”)`
Ścieżkę do katalogu z pakietami można uzyskać komendą `.libPaths()`
Lista poręcznych pakietów: `devtools`, `tidyr`, `dplyr`, `ggplot2`,
Przewodnik (dane i pomoce do książki Biecka), `PogromcyDanych`, `car` (funkcja `scatterplot` i dane, `readxl` (wersja 0.1.0) zawiera funkcje do wczytywania danych, `e1071`, `psych`, `dprep` (pakiet poza CRAN, jest w archiwum: <https://cran.r-project.org/src/contrib/Archive/dprep/>) zawierają funkcje miar położenia i zmienności.
`install.packages („<packagename_vector>”, lib= „<where to instal>”, repos= „<URL of repository>”)`
instaluje pakiety.
Uwaga, w przypadku instalacji z pliku należy doładować wszystkie pakiety powiązane.

5. Podstawowe typy danych

Atrybuty danych: typ i długość

Typy proste:

Numeryczny (liczba),

Znakowy - łańcuch znaków ujętych w " " lub ' ', może zawierać znaki sterujące (np. \n - nowa linia, \t - tabulator),
Zespolony - liczba zespolona,
Logiczny - TRUE/T lub FALSE/F,

Typy dodatkowe: funkcja, wyrażenie.

Obiekty o elementach stałego typu:

- wektor - wszystkie typy proste,
- czynniki - numeryczny i znakowy,
- tabela - wszystkie typy proste.

Obiekty o elementach różnego typu:

- ramka - wszystkie typy proste,
- lista - wszystkie typy proste i dodatkowe.

Polecenia: `mode`, `typeof` i `length` podają odpowiednio typ i długość danej.

5.1. Wektory

Wektor, sekwencja wartości tego samego typu. Nawet jedna liczba też jest wektorem.

Wektory możemy tworzyć przy pomocy:

polecenia `vector('<type>', <length>)` lub funkcji `numeric(length)` oraz `integer`, `character`, `logical`, `complex` o identycznej semantyce (?).

funkcją `c(<lista wartości>).v <- c()` tworzy wektor pusty

Wektory możemy tworzyć podstawiając do ich nazwy sekwencje, np. `v<-3:3` (tutaj mamy stały krok sekwencji równy 1) lub przy użyciu funkcji

`v <- seq(from=<początek>, to=<koniec>, by=<krok>)`, lub skrótowo
`v <- seq(<początek>, <koniec>, <krok>)`.

Funkcja `rep(<wektor>, <iliterazy>)` tworzy wektor będący powtórzeniem zmiennej wektor.

Indeksowanie i inne operacje na wektorach:

`v[3]` jest odniesieniem do 3-go elementu wektora `v`.

`v[-3]` jest odniesieniem do wszystkich elementów poza trzecim,

`v[3,5]` jest odniesieniem do elementów 2,3,4,5.

`v[c(2,3,5)]` jest odniesieniem do elementów 2,3,5.

`v[v>1]` jest odniesieniem do wszystkich elementów większych od 1.

Na wektorach można wykonywać prawie wszystkie operacje matematyczne. Są one wykonywane na współrzędnych.

Współrzędne wektora mogą mieć wartość NA (not available). Jest to sposób na oznaczenie brakujących danych. Funkcja `is.na(v)` zwraca wektor logiczny tej samej długości co `v`, który ma współrzędne TRUE gdy współrzędna `v` przyjmuje wartość NA natomiast FALSE w przeciwnym przypadku.

Współrzędne wektora mogą przyjmować wartości nieokreślone na skutek obliczeń numerycznych NaN (Not a Numeric). Funkcja `is.na(v)` wykrywa je podobnie jak wartości NA.

Współrzędne wektora mogą przyjmować wartości nieskończone `+Inf`, `-Inf`. Wartości te wykrywa funkcja `is.inf(v)` podobnie jak `is.na(v)` wartości nieokreślone.

5.2. Czynniki (factor)

Jest strukturą, która oprócz danych przechowuje liczbę wystąpień (coś jak multizbiór). Tworzymy go funkcją `factor(<vector>)`. Częstość wystąpień możemy uzyskać funkcją `table()` lub `ftable()`. Możemy tworzyć faktory wielowymiarowe, łącząc dane o różnych cechach tych samych osobników.

Funkcja `prop.table(<tablica>)` podaje rozkład częstości dwuwymiarowej dla wyrazów tablicy. Funkcja `margin.table(<tablica>, <index wymiaru>)` wylistowuje tabele częstości brzegowych

Na przykład piąty_skrypt_R.

5.3. Tablice

Są to tablice 2 i więcej wymiarowe.

Tablice 2-wymiarowe tworzymy komendą `G=matrix(<dane, wektor>, nrow=<l-wierszy>, ncol=<l_kolumn>, byrows=TRUE/FALSE, dimnames=<NULL/nazwa_listy>)`. Parametr `byrows=TRUE` powoduje wypełnienie tablicy danymi wiersz po wierszu, `FALSE` kolumna po kolumnie.

Dla macierzy 2 wymiarowych dostępne są liczne funkcje: `t(A)` – transpozycja, `A%*%B` – iloczyn A i B , `det(A)` – wyznacznik, `solve(A, b)` – rozwiązuje układ liniowy $Ax=b$, `eigen(A)` – oblicza wartości i wektory własne A . Argumenty każdej z tych funkcji muszą spełniać odpowiednie założenia.

Tablice tworzone są kolumnowo. Tablice 3-wymiarową można tworzyć z wektora lub innych danych x komendą

`dim(x) <- c(<l_kolumn>, <l_wierszy>, <l_wartsw>)`. Polecenie to usuwa wiersze i kolumny komentarzy (etykiet) z x .

Funkcja `dim(x)` zwraca wektor ilości kolumn, wierszy, warstw etc.

Funkcje `rbind(A, <dane_w>)` i `cbind(A, <dane_k>)` dołączają nowy wiersz i kolumnę do istniejącej macierzy A .

Polecenie `y=array(<vector>, c(<l_kolumn>, <l_wierszy>, <l_wartsw>))` tworzy tablicę trójwymiarową z wektora.

Indeksowanie macierzy lub tablicy `y[1, 2, 2, 2]` element tablicy 4 wymiarowej.

`z[, 1]` pierwsza kolumna macierzy. Liczne funkcje na macierzach 2 wymiarowych.

Wierszom, kolumnom, ... tablicy A możemy nadawać etykiety funkcją `dimnames(A) = list(c("<et_w1>", "<et_w2>", ...), c("<et_k1>", "<et_k2>", ...), ...)`

5.4. Listy

Listy mogą mieć elementy różnych typów. Listy tworzymy komendą `list(<lista elementów>)`. Indeksowanie list wykorzystuje podwójny nawias kwadratowy, tj.

`<listname>[[<numer elementu>]]`. Elementy listy mogą mieć nazwy stringowe `L=list(<name1>=<element_1>, ..., <name_end>=<element_end>)`.

Wówczas indeksujemy i -ty element `L$<name_i>`.

5.5. Ramki (frames)

Specyficzna struktura R. Macierz, której poszczególne kolumny mogą zawierać elementy różnego typu. Najczęściej służy do obsługi obserwacji reprezentowanych przez wiersze, a poszczególne cechy (wyniki) to kolumny.

Tworzymy ją funkcją `data.frame`. Do kolumny możemy odwoływać się poprzez indeksowanie `<frame>$<collon name>`, tak jak w przypadku elementu listy, lub tak jak w przypadku tablicy `<frame>[, <numer kolumny>]`.

Ważną funkcją dla ramek i innych struktur jest `sapply(<frame>, <funkcja>)`. W jej wyniku funkcja wykonywana jest dla wszystkich elementów struktury, którą może być,

ramka, wektor/tablica i lista, lub ich fragment. Funkcja ta ma dużo wrapperów dla różnych struktur.

5.6. Wyrażenie (expression)

Wyrażenie to ciąg operacji poprawnych semantycznie w R. Wyrażenie tworzymy instrukcją `expression(<napis>)`. Jeżeli wyrażenie ma określone wartości zmiennych możemy obliczyć komendą `eval(<wyrażenie>)`.

5.7. Konwersja typów danych

Funkcja `class(<zmienna>)` zwraca string z nazwą typu zmiennej

Komenda `is.<typ>(<obiekt>)`, np. `is.numeric(<obiekt>)` sprawdza, czy obiekt jest liczbą.

Funkcja `as.<nowy typ>(<obiekt>)` konwertuje obiekt do nowego typu, np. `as.array(<obiekt>)` traktuje obiekt jako tablicę.

6. Zbiory danych, obiekty

Są danymi zawartymi w pakietach w postaciach zgodnych z semantyką R. Są dostępne przez nazwy.

Dane w dostępnych pakietach są obiektami typu ramka (najczęściej). Można ich używać przez nazwę indeksować i wykonywać legalne funkcje.

Komenda `data(package = .packages(all.available = TRUE))` podaje wszystkie dostępne zbiory danych. Komenda `data(package = '<packagename>')` podaje dane dostępne w konkretnym pakiecie, lub

`data(package = c('<packagename_1>', ..., '<packagename_n>'))`.

`data(<dataname>)` ładuje zbiór danych, niezbędne w starszych wersjach.

`head, tail(<dataname>)` wyświetla początkowe lub końcowe 6 linii zbioru danych

`View('<dane>')` wyświetla zawartość obiektu w podstawowym okienku.

6.1. Operatory przypisania i zmienne nazwane w R

`<nazwa zmiennej> = <wartość>` przyporządkowuje (podstawia) wartość nazwie.

Przyporządkowanie jest lokalne, tj. w trakcie przetwarzania jednego skryptu.

`<nazwa zmiennej> <- <wartość>` przyporządkowuje wartość nazwie.

Przyporządkowanie to zostaje zapisane w środowisku i pojawia się w prawym górnym oknie. Przyporządkowanie jest trwałe, jeżeli środowisko jest zapisane, funkcjonuje przy ponownym wejściu do RStudio.

6.2 Warunki i pętle

```
if (<wyr_logiczne>) <instrukcja>
```

```
if (<wyr_logiczne>) {  
  <instrukcja>  
  <instrukcja>  
  . . .  
  <instrukcja>  
}
```

```
if (<wyr_logiczne>) <instrukcja>  
else
```

```

<instrukcja>

if (<wyr_logiczne>) {
<instrukcja>
<instrukcja>
. . .
<instrukcja>
} else {
<instrukcja>
<instrukcja>
. . .
<instrukcja>
}

```

Warunek to jednoelementowy wektor (zmienna) logiczny. Konstrukcje `if` mogą być zagnieżdżone. Wyrażenia logiczne możemy konstruować za pomocą koniunkcji `&` i alternatywy `|` wektorowej (po każdej współrzędnej) lub podobnych operatorów skalarnych `&&` i `||` oraz relacja równości `==`.

```

while(<wyr_logiczne>) {
<instrukcja>
<instrukcja>
. . .
<instrukcja>
}

```

Przebieg pętli możemy przerwać instrukcją `break`.

```

repeat {
<instrukcja>
<instrukcja>
. . .
<instrukcja>
}

```

Jest to pętla nieskończona, której działanie musimy przerwać instrukcją `break`.

```

for(<zm_sterująca> in <wektor>) <instrukcja>

```

lub

```

for(<zm_sterująca> in <wektor>) {
<instrukcja>
<instrukcja>
. . .
<instrukcja>
}

```

Pętle mogą być zagnieżdżone.

Uwaga! Położenie fragmentów zaznaczonych na czerwono jest krytyczne!

6.3 Funkcje

Funkcje w R są także obiektami. Mogą być bez nazwy, lub nazwane. Syntaktyka funkcji jest w uproszczeniu następująca:

```
<f_name> <- function(<argument_list>) {  
  <instrukcja>  
  <instrukcja>  
  . . .  
  return(<wyrażenie>)}  
}
```

Instrukcje powinny zależeć od argumentów (być wykonywane na argumentach). Pod nazwę funkcji podstawiana jest wartość wyrażenia, argumentu `return()`, lub wartość wyrażenia w ostatniej instrukcji (`return` jest opcjonalne).

7. Wczytywanie plików tekstowych i arkuszy Excel

W RStudio możemy to zrobić opcją File -> Import Dataset. Po zaimportowaniu zbioru jest on wyświetlany w okienku podstawowym w ty gdzie są skrypty) komendą `View(<fname!noextension!>)`. Uwaga, nazwy plików nie powinny zawierać znaków diakrytycznych, powoduje to błąd syntaktyczny, ale plik się otwiera. Opcja menu File -> Import Dataset->From Excell nie działa, bez pakietu `readxl` 0.1.0 Pliki excela czyta funkcja `data= read_excel('<fname>')`. dane mają typ ramki.

Pliki Excela otwierają się bez problemów. Interpretacja danych będących elementami tabel, jest skomplikowana, np. pojawiają się "ukryte zera".

Pojedynczy wektor wczytujemy komendą

```
vector=scan('<fname>')
```

Dane z pliku tekstowego można wczytać

```
dane = read.table('<fname>', header=T)
```

Parametr `header` określa, czy w pliku znajdują się nagłówki kolumn. Dyfułth to `header=FALSE`.

Funkcja

```
write.table(<obiekt>,"<file_n>", col.names=T/F, row.names=T/N, ...  
) zapisuje obiekt do pliku, col.names (row.names)=T powoduje zapis etykiet kolumn lub/i wierszy
```

Wczytane dane mają typ ramki.

Pliki excela czyta funkcja `data= read_excel('<fname>')`. dane mają typ ramki. Trudno ustalić typy składników, np. zapisane w excelu liczby nie czytają się jako liczby.

Interpretacja plików tekstowych przygotowywanych poza R jest również skomplikowana.

Podsumowując, najlepiej przygotować dane przy pomocy skryptu R w postaci frame i zapisać w pliku tekstowym funkcją `write.table`. Można odczytać go funkcją `read.table`.

Zadania:

1. Działania na wektorach i macierzach:
 - a. Utwórz dwa wektory liczbowe x, y o 10 współrzędnych. Utwórz macierz A o dwóch kolumnach x, y . Oblicz transpozycję A^T . Oblicz iloczyn A^T przez y .
 - b. Utwórz macierz kwadratową A symetryczną o wymiarze 3×3 o wyrazach dodatnich, diagonalnie zdominowaną, oraz wektor liczbowy b o 3 współrzędnych. Oblicz wyznacznik A oraz rozwiąż układ równań liniowych $Ax = b$.
 - c. Utwórz wektor c o 3 współrzędnych. Utwórz macierz B przez dołączenie c do A jako ostatniej kolumny. Utwórz wektor d o 4 współrzędnych. Utwórz macierz G przez dołączenie do B ostatniego wiersza d .
 - d. Dla macierzy G z poprzedniego zadania stwórz etykiety nazw kwiatów dla kolumn i imion żeńskich dla wierszy. Zbadaj wymiar macierzy G po dołączeniu etykiet.
 - e. Utwórz macierz liczbową 3×3 wraz z etykietami poleceniem `matrix`.
 - f. Utwórz z wektora x tablicę Z o wymiarze $3 \times 3 \times 2$. Wykonaj to samo poleceniem `dim`.
2. Działania na listach i ramkach
 - a. Utwórz listę `list1` z nazwami pozycji zawierającą wektor stringów, macierz liczbową i macierz wartości logicznych. Zbadaj typ tej zmiennej. Oblicz macierz pierwiastków współrzędnych macierzy liczbowej zawartej w liście odwołując się do niej przez nazwę.
 - b. Utwórz 3 wektory o 10 współrzędnych: `palenie`, `plec` i `wiek`. `Palenie` to wektor logiczny, `plec` to wektor stringowy zawierający wartości „K” lub „M” a `wiek` to wektor liczbowy o współrzędnych z przedziału 1 – 100. Utwórz ramkę `badanie` o kolumnach: `czy_pali`, `plec` i `wiek` z tych wektorów. Sprawdź, która kolumna ramki zawiera wartości liczbowe. Zlicz ilości kobiet i mężczyzn w danych badanie.
3. Pliki i dane (obiekty) z pakietów
 - a. Zapisz plik w lokalnej kartotece zawierający ramkę `badanie` z zadania 2.b. Wczytaj z tego pliku dane do ramki `Nowe_badanie`.
 - b. Zapisz dane `beaver1` z pakietu `boot` jako plik w lokalnej kartotece.
4. Funkcje, pętle i warunki
 - a. Przy pomocy pętli oblicz iloczyn skalarny pozycji `temp` i `activ` obiektu `beaver1` z pakietu `boot`.
 - b. Napisz funkcję obliczającą ilość współrzędnych zerowych w wektorze liczbowym.
 - c. Napisz funkcję, która znajduje indeks pierwszego i ostatniego elementu wektora logicznego o wartości `TRUE`. Wynik zapisz w postaci wektora 2-elementowego. Jeżeli żaden z elementów nie jest `TRUE` funkcja zwraca `c(NA,NA)`.
 - d. Napisz funkcję `moda(x)`, zwracającą najczęściej występującą wartość współrzędnej x .