Eksploracja danych - podstawy języka R

Ćwiczenia oparte są na:

1. Biecek, Przemysław. **Przewodnik po pakiecie R**. Wrocław: Oficyna Wydawnicza „GIS", 2011.
2. Walesiak, Marek, Gatnar, Eugeniusz, i Andrzej Bąk. **Statystyczna analiza danych z wykorzystaniem programu R**. Warszawa: Wydawnictwo Naukowe PWN, 2009.
3. Knell, Robert. **Introductory R: A Beginner's Guide to Data Visualisation and Analysis Using R** (pierwsze cztery rozdziały - za darmo można je pobrać ze strony autora http://www.introductoryr.co.uk/).
4. Dokumentacja ze strony projektu http://cran.r-project.org/, An Introduction to R
5. Zuur, Alain F, Elena N Ieno, i Erik H. W. G Meesters. **A Beginner's Guide to R**. Dordrecht; New York: Springer, 2009.

Ciekawe kursy języka R odnajdziemy na stronieOnline learning - RStudio. Innym dobrym sposobem na naukę języka R jest pakiet swirl.

Uruchamiamy program R i w konsoli wpisujemy polecenia.

Ćw. 1

Instalacja R i Rattle

- Najprościej zainstalować R korzystając ze skompilowanego pliku instalacyjnego, który możemy pobrać z http://cran.r-project.org/
- Po zainstalowaniu R klikamy ikonę R na pulpicie (32 bit lub 64 bit) i wpisujemy następujące polecenia w wierszu R. R poprosi o podanie mirrora (serwera lustrzanego) CRAN (Comprehensive R Archive Network - Kompleksowe Archiwum Sieciowe R). Wybieramy serwer znajdujący się w pobliżu.

  ```
  install.packages("rattle")
  ```

- W trakcie instalacji mogą zostać zainstalowane pakiety będące zależnościami Rattle.
- Wprowadzamy dwa następujące polecenia w wierszu R. To załaduje pakiet Rattle, a następnie startuje go:

  ```
  library(rattle)
  rattle()
  ```

- Jeśli pakiet RGtk2 nie został jeszcze zainstalowany, to pojawi się okienko błędu wskazujące, że brak pewnej biblioteki DLL. Klikamy na OK, a następnie pojawi się pytanie, czy chcemy zainstalować GTK +. Klikamy przycisk OK, aby to zrobić. Pakiet jest pobierany i instalują się odpowiednie biblioteki GTK + na naszym komputerze. Po zakończeniu warto wyjść z R i uruchomić go ponownie, tak aby uzyskać dostęp do zainstalowanych bibliotek.
- Podczas uruchamiania i pracy z Rattle spora liczba innych pakietów zostanie pobrana i zainstalowana (jeśli nie były wcześniej zainstalowane), a Rattle będzie pytał użytkownika przed zainstalowaniem o zgodę. Muszą one zostać zainstalowane tylko raz.

Ćw. 2

Przykładowe możliwości pakietu

```
>demo(graphics)
>demo(persp)
```

Ćw. 3

R jako kalkulator

```
>2+2
>2^10 - 1
>log(1024, 2)
>sin(pi/3)^2 + cos(pi/3)^2
```

Ćw. 4

System pomocy w R

R ma wbudowany system pomocy. Aby uzyskać informację na temat jakiejś funkcji, np. solve używamy komendy

```
>help(solve)
#Alternatywny zapis tego polecenia:
>?solve
```

Po użyciu tej komendy R najczęściej startuje serwer WWW i wyświetla pomoc w przeglądarce w formacie HTML. Jeśli szukana komenda zawiera znaki specjalne, to argument funkcji help musi byc zawarty w pojedyńczych lub podwójnych cudzysłowach, jest to wymagane również jeśli podajemy kilka słów zawierających np. słowa if, for lub function:

```
>help("[[")
```

Jeśli chcemy uruchomić cały interaktywny system pomocy w formie HTML, to wpisujemy:

```
help.start()
```

Polecenie to uruchomi serwer WWW i przeglądarkę ze stroną pomocy i odsyłaczami. System pomocy możemy przeszukiwać za pomocą jednego z dwóch poleceń:

```
>help.search('solve')
>??solve
```

Aby uzyskać pomoc o tym poleceniu i przykłady wpisujemy ?help.search. Możemy również zobaczyć przykłady użycia za pomocą polecenia:

```
example(solve)
```

Ćw. 5

Zmienne w R, przypisanie wartości i komentarze.

Znaki używane w nazwach zmiennych zależą od systemu operacyjnego i jego lokalizacji (locale). Normalnie wszystkie znaki alfanumeryczne są dopuszczalne (a dla niektórych krajów również pewne znaki z akcentami - ale raczej nie powinniśmy ich używać) oraz znaki '.' i '_'. Nazwa musi rozpoczynać się od kropki lub litery, a jeśli pierwsza jest kropka to kolejnym znakiem nie może być cyfra. Nazwy nie są ograniczone jeśli chodzi o liczbę znaków.

Uwaga, ciąg instrukcji:

```
zmienna_1 <- 5
if (zmienna1>4) print("Warunek spełniony")
```

powoduje wyświetlenie komunikatu o błędzie: Error: object "zmienna1" not found!

```
a <- 3
b <- 5
a + b
c <- a/b + 2*b + 1
wektor <- c(11, 13, 10.5, -3, 11)
```

Znak # w języku R rozpoczyna komentarz. Wszystkie inne polecenia występujące po nim do końca wiersza są traktowane jako komentarz:

```
#komentarz w języku R
```

Operatory przypisania można swobodnie łączyć. Na przykład wydanie polecenia:

```
zmienna1=zmienna2 <- 2+2 -> zmienna3
```

spowoduje, że zarówno zmienna1, jak i zmienna2 oraz zmienna3 przyjmą wartość 4.

Ćw. 6

Atrybuty i usuwanie obiektów

Każdy obiekt złożony w języku R ma zdefiniowaną listę atrybutów z nim związanych. Aby wyświetlić listę atrybutów związanych z obiektem, stosuje się polecenie attributes(). Przykładowe wywołanie tego polecenia:

```
library(rattle)
objects()
attributes(weather)
x<-1:6
x
rm(x)
x
```

Ćw. 7

Dostępne obiekty i ich usuwanie

Komenda R objects()

```
objects()
```

Zmienne (obiekty) usuwamy za pomocą rm()

```
rm(wektor,a,b)
wektor
#BŁĄD: nie znaleziono obiektu 'wektor'
```

Ćw. 8

Operator przypisania

1. Stwórz zmienną my_apples i przypisz do niej wartośc 5.
2. Wyświetl wartość zmiennej my_apples.
3. Stwórz zmienną my_oranges i przypisz do niej pewną wartość liczbową (całkowitą).
4. Niech zmienna my_fruit zawiera sumę dwóch poprzednich zmiennych.

Tworzenie zmiennych różnych typów:

```
my_numeric <- 42
my_character <- "forty-two"
my_logical <- FALSE
```

Ćw. 9

Funkcja print

Podczas pracy w trybie interaktywnym wyniki obliczenia pojawiają się natychmiast w oknie programu R. Inaczej jest przy uruchamianiu skryptów zewnętrznych. W takim przypadku w konsoli środowiska R pojawiają się tylko te wartości wyrażeń, które zostaną wprost wskazane przez programistę za pomocą funkcji print(). W podanym przykładzie funkcja print wyświetli wartość zmiennej wynik

```
wynik  <-  sin(pi/6)
print(wynik)
```

Za pomocą funkcji print można zdefiniować sposób wyświetlania liczb, określić liczbę znaczących cyfr (parametr digits), zdefiniować, czy tekst ma być wyświetlony w cudzysłowach (parametr quote), oraz określić sposób wyświetlania wartości niedostępnych (na.print) czy zer (zero.print):

```
print(sin(60*pi/180),digits=11)
print("Napis  w  cudzysłowie",  quote=TRUE)
print("Napis  bez  cudzysłowów",  quote=FALSE)
print(c(1,3,NA,5,3,NA,NA,14), na.print="Brak  wartości")
```

Ćw. 10

Pakiety

Zazwyczaj do instalacji typowego pakietu w systemie Windows wystarczy wybranie z menu polecenia Packages/Install package(s) (Pakiety / Zainstaluj pakiet(y)). Następnie należy wybrać serwer, z którego pakiet powinien zostać pobrany, najlepiej jak najbliżej miejsca pracy użytkownika. Serwer ten będzie używany do końca sesji, chyba że zostanie zmieniony poleceniem menu Packages/Set CRAN mirror. Zainstalujmy pakiet Hmisc.

Listę pakietów zainstalowanych w systemie wraz z krótkim ich opisem zwraca polecenie:

```
library()
```

Aby załadować pakiet do pamięci operacyjnej używamy tego samego polecenia z nazwą pakietu:

```
library(Hmisc)
library(rattle)
```

Żeby otrzymać opis pakietu i dostępne funkcje wykonujemy polecenie:

```
library(help = rattle)
help(package = rattle)
```

Żeby usunąć z pamięci pakiet używamy komendy:

```
detach(package:rattle, unload = TRUE)
```

Listę pakietów załadowanych do pamięci operacyjnej zwraca funkcja

```
search()
```

Ćw. 11

Liczby

Wyróżnioną wartością jest NaN (to skrót rozwijający się w ang. not a number, czyli „nie liczba”)

```
a<-log(-3)
a
```

Literały Inf i -Inf oznaczają plus i minus nieskończoność.

```
1/0
-2/0
```

Ćw. 12

Łańcuchy

funkcja cat() wyświetla napis w sposób niesformatowany, a funkcja paste() je łączy:

```
a<-"Dziś jest niedziela"
cat(a)
paste("Witaj ","świecie")
```

Ćw. 13

Wektory

Definiowanie wektorów

```
wektor1 <- c(3,4,2,4,5,7)
wektor2 <- 2:8
wektor3 <- 5:-1
wektor4 <- seq(3, 6)
wektor5  <-  seq(2,  23,  5)
wektor6 <- seq(11, 25, length.out=4)
print(wektor6)
wektor7 <- seq(10,20, along.with=c(14,23,17,2,6,10))
wektor8 <- rep(10, 20)
print(wektor8)
wektor9 <- rep(c(10,13,17),5)
print( wektor9)
wektor10 <- rep(c(1,4,-3), c(2,3,5))
wektor11 <- sample(1:10, 4)
print(wektor11)
wektor12 <- sample(1:20, 25, rep=TRUE)
# rep=TRUE oznacza, że elementy mogą się powtarzać
print(wektor12)
```

Długośc wektora

```
length(wektor11)
```

Wektory mogą zawierać dane innych typów niż liczby (ale typów nie można mieszać):

```
numeric_vector <- c(1, 10, 49)
character_vector <- c("a", "b", "c")
# stworz dowolny wektor boolowski
boolean_vector <-
```

Operacje na wektorach

Na wektorze możemy wykonywać operacje arytmetyczne.

```
wektor^2
1 / wektor
wektor - 2
```

Wektory można łączyć w jeszcze większe wektory.

```
c(wektor, 0, 3:5, wektor)
```

Długie sekwencje liczb łatwiej jest generować używając operatora :.

```
1:10
wektor2  <-  2:8
wektor3  <-  5:-1
```

Wektory możemy tworzyć funkcją seq():

```
wektor4  <-  seq(3,  6)
wektor5  <-  seq(2,  23,  5)
wektor6  <-  seq(11,  25,  length.out=4)
print(wektor6)
wektor7 <- seq(10,20,along.with=c(14,23,17,2,6,10))
```

Funkcja rep() zwraca wektor utworzony przez replikację elementów pierwszego argumentu.

```
wektor8 <- rep(10, 20)
print(wektor8)
wektor9 <- rep(c(10,13,17),5)
print( wektor9)
wektor10 <- rep(c(1,4,-3), c(2,3,5))
# lub alternatywnie
wektor10 <- c(rep(1,2),rep(4,3),rep(-3,5))
print(wektor10)
```

Wektor losowych liczb z określonego przedziału i o zadanej długości:

```
wektor11 <- sample(1:10, 4)
print(wektor11)
wektor12 <- sample(1:20, 25, rep=TRUE)
# rep=TRUE oznacza, że elementy mogą się powtarzać
print(wektor12)
```

Jaka jest długość wektora?

```
length(wektor)
```

Co jest w pierwszym elemencie wektora wektor? A co jest w jego drugim i trzecim elemencie?

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

Wypiszmy wartości dodatnie z wektora (wartości o indeksach odpowiadającym wartościom dodatnim).

```
wektor[wektor>0]
```

Ćw. 14

Wyjście z R

```
> q()
> q(save = "no")
```

Ćw. 15

Wpisywanie małych zbiorów danych.

Postaramy się teraz wprowadzić do środowiska R mały zbiór danych dotyczący pomiaru 8 ptaków.

Jest to zbiór danych z książki **A Beginner's Guide to R**. Są to dane morfometryczne (długość skrzydła - wing, nogi - tarsus, głowy - head i waga - weight).

Wyniki pomiaru 8 ptaków

Tworzymy odpowiednie wektory za pomocą c() (concatenate):

```
> Wingcrd <- c(59, 55, 53.5, 55, 52.5, 57.5, 53, 55)
> Wingcrd
> Tarsus <- c(22.3, 19.7, 20.8, 20.3, 20.8, 21.5, 20.6, 21.5)
> Tarsus
> Wt <- c(9.5, 13.8, 14.8, 15.2, 15.5, 15.6, 15.6, 15.7)
> Wt
```

Możemy teraz do tych wektorów używać welu funkcji statystycznych R (sum, mean, max, min, median, var, sd). Np. możemy policzyć sumę długości skrzydeł i przypisać ją do nowej zmiennej:

```
> S.win <- sum(Wingcrd)
> S.win
```

Co się stanie jeśli spróbujemy to samo zrobić dla Head?

```
> sum(Head)
```

Wynik nas trochę zaskoczył, większość funkcji R nie radzi sobie z wektorami, gdzie mamy brakujące wartości (NA). Możemy się z tym szybko uporać przekazując do funkcji parametr na.rm = TRUE

```
> sum(Head, na.rm = TRUE)
```

Utworzymy teraz z tych danych macierz:

```
> Z <- cbind(Wingcrd, Tarsus, Head, Wt)
> Z
```

Możemy uzyskać dostęp do jej pewnej części:

```
Z[, 1]
Z[1 : 8, 1]
Z[2, ]
Z[2, 1:4]
```

Następujące komendy są również poprawne:

```
> Z[1, 1]
> Z[, 2 : 3]
> X <- Z[4, 4]
> Y <- Z[, 4]
> W <- Z[, -3]
> D <- Z[, c(1, 3, 4)]
> E <- Z[, c(-1, -3)]
```

Rozmiar Z

```
> dim(Z)
```

Zamiast funkcji cbind() (column) możemy użyć rbind() (row):

```
>Z2 <- rbind(Wingcrd, Tarsus, Head, Wt)
>Z2
```

Ćw. 16

Factor - czynniki

- Typ factor - to typ wektorowy przeznaczony głównie do przechowywania danych jakościowych. Zawiera dodatkową informację o różnych klasach(przypisanych nazwach) danych oraz częstości ich występowania.
- Do przekształcenia wektora do typu factor służy funkcja **factor**

```
(wzrost<-rep(c("niski","średni","wysoki"),c(4,3,5)))
(factor_wzrost<-factor(wzrost))
```

Zasady przekształcania na typ factor

1. wyznaczanie zbioru wartości - levels - (podział na klasy);
2. przyporządkowanie kolejnych wartości naturalnych każdej klasie;
3. przekododowanie wektor, tak aby zawierał informacje o numerach poziomów a nie o nazwach.

Factors, to wektory liczb całkowitych z ustawionym atrybutem levels.

Często wykorzystywane funkcje:

1. levels - do nadawania/oczytu różnych klas z danych;
2. table, summary - do odczytu częstości.

```
levels(factor_wzrost)
table(factor_wzrost)
summary(factor_wzrost)
```

Funkcją "odwrotną" do factor() jest funkcja as.vector()

```
as.vector(factor_wzrost)
as.integer(factor_wzrost)
```

inny przykład

```
faktor <- factor(c(2,3,4), levels=1:5)
print(faktor)
levels(faktor)<-c("zły","może  być","przeciętny","dobry","idealny")
print(faktor)
as.numeric(faktor)
```

Zabronione są działania arytmetyczne na typie factor.

```
faktor2 <- factor(c(2,3,2,2,3,4), levels=1:5)
print(faktor2)
x<-faktor2+2
```

Próba podstawienia pod zmienną, ze zdefiniowanymi kategoriami, elementu niezdefiniowanego w atrybucie levels kończy się komunikatem o błędzie, a wartość tego elementu będzie nieokreślona NA.

```
faktor[3] <- "nijaki"
Komunikat ostrzegawczy:
W poleceniu ``[<-.factor`(`*tmp*`, 3, value = "nijaki")':
  invalid factor level, NA generated
```

Przy tworzeniu factor można ustalić porządek.

```
factor(some_vector,
       ordered = TRUE,
       levels = c("lev1", "lev2" ...))
```

Przykład

```
typ_v<-c("wolno","szybko","średnio","pyr pyr")
predkosc <- rep(typ_v,c(12,33,2,100))
v<- sample(predkosc, 50, replace = TRUE)
v
fv<-factor(v, ordered = TRUE, levels = c("wolno","szybko","średnio",
                                                    "pyr pyr"))
levels(fv)
```

W takim przypadku możemy również porównać wartości:

```
(da2 <-fv[2])
(da5 <-fv[15])
# Czy szybko>pyr pyr?
da2>da5
```

Poprawiając porządek, mamy

```
fv<-factor(v, ordered = TRUE, levels = c("pyr pyr","wolno",
                                         "średnio","szybko"))
levels(fv)
da2 <-fv[2]
da2
da5 <-fv[15]
da5
# Czy szybko>pyr pyr?
da2>da5
```

Używając typu factor mozna dokonać podziału wektora. Służy do tego funkcja cut, która tworzy factor z podanego wektora, dzieląc go na przedziały.

```
wiek <- sample(1:100, 16, replace = TRUE)
wiek
cut(wiek, c(0, 18, 26, 100))
kat_wiekowe <-cut(wiek, c(0, 18, 26, 100))
table(kat_wiekowe)
```

Dla funkcji cut można podać jako drugi argument liczbę podziałów.

Dla danego wektora oraz factora tej samej długości co wektor, można ustanowić podział na grupy

```
kat<-factor(rep(c("klI","klII","klIII","klIV"),4))
str(split(wiek,kat))
```

Funcja split

```
ma <- cbind(x = 1:10, y = (-4:5)^2)
split(ma, col(ma))

split(1:10, 1:2)
```

Ćw. 17

Lista

W języku \R dość często występuje sytuacja, gdy efektem działania procedury nie jest pojedyncza wartość, lecz obiekt złożony, zawierający wiele informacji - lista. Do utworzenia listy wykorzystywana jest funkcja list(). Na przykład, jeśli w jednym obiekcie listy ma być zawarta średnia, wartość maksymalna i minimalna wektora x, wówczas mamy

```
x <- c(2,3,2,5,4,3,6,7,8)
lista_x <- list(srednia=mean(x),minimum=min(x),maksimum=max(x))
```

Funkcja str wyświetla wszystkie elementy składowe listy. Ponadto typy składowych, długości tablic, aktualne wartości itp.

```
str(lista_x)

List of 3
 $ srednia : num 4.44
 $ minimum : num 2
 $ maksimum: num 8
```

W celu odwołania się do poszczególnych elementów składowych obiektu złożonego należy użyć selektora $, np.

```
print(lista_x$maksimum)
```

lub selektora [[numer]] oznaczającego wybór elementu o podanym numerze. Poprzednie polecenie może być zapisane również jako

```
lista_x[[3]]
```

Lista - różnice w indeksowaniu

```
a<-list(1:5, LETTERS[1:15], list("ą","ć","ę"))
```

- [[numer]] - zwraca element, który jest w liście,
- [numer] - zwróci listę - ograniczoną do elementu o wskazanym numerze.

```
 a[[2]]
[1] "A" "B" "C" "D" "E" "F" "G" "H" "I" "J" "K" "L" "M" "N" "O"

 a[2]
[[1]]
 [1] "A" "B" "C" "D" "E" "F" "G" "H" "I" "J" "K" "L" "M" "N" "O"
```

więc a[[2]][10] daje dostęp do 10 elementu w wektorze, a a[2][10] daje NULL.

Przy tak utworzonej liście a, nie można odwoływać się do składowych przy użyciu $.

Można wówczas użyć funkcji names.

```
nazwy<-c("cyferki","literki","ogonki")
names(a)<-nazwy
str(a)
a$cyferki
```

Ćw. 18

Ramka danych

Istotnym ograniczeniem tablic,macierzy jest jednorodność ich elementów. Ramki danych (data.frame) mogą zawierać dane liczbowe, dane w postaci łańcuchów tekstowych lub zmiennych logicznych TRUE/FALSE.

Ramka danych to kolumnowo niejednorodna macierz zaimplementowana przy pomocy listy.

Do utworzenia ramki danych wykorzystujemy funkcję data.frame()

```
 tabela.danych1 <- data.frame(LETTERS[1:10],1:10,rep(c(F,T),5))
 names(tabela.danych1) <- c("Inicjał","Kolejność","Czy Parzysty")
```

Do tworzenia tabel danych stosunkowo rzadko używa się funkcji data.frame, a o wiele częściej korzysta się z funkcji read.csv(), którą omówimy później.

Ramka danych służy do wczytywania zbiorów danych.

```
str(iris)
table(iris$Sepal.Length, factor(iris$Species))
```

Ćw. 19

Instrukcje w R

Wykonać polecenia

```
ifelse(1:7 < 4, "mniej", "wiecej")
ifelse(sin(1:5)>0, (1:5)^2, (1:5)^3)

liczba <- 1313

switch(class(liczba),
   logical = ,
   numeric = cat("typ liczbowy lub logiczny"),
```

```
   factor = cat("typ czynnikowy"),
   cat("trudno okreslic")
)

for (i in 1:5) {
     cat(paste("aktualna wartosc zmiennej i to ",
               i, "\n"))
}
```

Ćw. 20

Utwórz na trzy rożne sposoby wektor składający się z elementów 10, 9, 8, 7, 6 (wykorzystaj konstruktory c, :, seq)

Ćw. 21

Oblicz wyznacznik macierzy

```
|-1 4 |
|2  1 |
```

(wykorzystaj funkcje det(), matrix()).

Ćw. 22

Dokonaj mnożenia macierzy:

```
|1  2|   |1   3 5|
|-2 4|   |-4 -1 2|
|1  7|
```

wykorzystaj operator %*% i funkcję matrix().

Ćw. 23

Wyświetl za pomocą jednej instrukcji sześciany liczb od 1 do 20 (wykorzystaj operator : oraz ^)

Ćw. 24

Wyświetl za pomocą jednej instrukcji kwadraty liczb parzystych od 16 do 40 (wykorzystaj funkcję seq() i operator ^).

Ćw. 25

Funkcje w R

Prosty przykład funkcji w R:

```
funkcja1 <- function() {
      cat("Dzisiaj jest ")
      cat(format(Sys.time(), "%A %B %d"))
}

funkcja1()
```

Funkcja w R, która liczy silnię może wyglądać następująco:

```
fact<-function(n){
 if (n < 2) {
    return(n)
 } else {
    return (fact(n-1)*n)
 }

}
```

Definiowanie powyższej funkcji nie ma sensu w R, ponieważ mamy dostępną funkcję factorial(), ale na jej podstawie napisać funkcję liczącą wartość ciągu Fibonacciego dla podanego n.

Ćw. 26

R Commander

Instalacja

```
install.packages("Rcmdr",dependencies=TRUE)
library(Rcmdr)
```

Analizujemy i wykonujemy przykłady z pliku R commander an Introduction.

Ćw. 27

RStudio

Czytamy i zapoznajemy się z dokumentami o RStudio

1. Introduction to RStudio
2. Using RStudio

Ćw. 28

Praca domowa

1. Utwórz wektor 100 liczb naturalnych ze zbioru {1,2,3,4}, następnie przekształć go do typu factor nadając mu następujące etykiety c("czerwony", "zielony", "niebieski", "żłółty"). Podaj ile razy powtórzył się każdy z kolorów.
2. Podziel Sepal.Length, odczytany z bazy iris, na 5 grup, tak aby uzyskać poniższy rezultat

```
(4.3,5.02] (5.02,5.74] (5.74,6.46] (6.46,7.18]  (7.18,7.9]
        32          41          42          24          11
```

3. Podaj częstość występowania Sepal.Length krótszych niż 5 i pozostałych, w różnych gatunkach kwiatów, tak aby uzyskać poniższy rezultat

```
       setosa versicolor virginica
FALSE      30         49        49
TRUE       20          1         1
```

4. Dla wektora x<-c(1,3,4,7,11,18,29), utwórz listę x2 składającą się z x*2,x/2,sqrt(x) oraz ponazywaj składowe listy "x*2","x/2","sqrt(x)". Napisz polecenie, które wyświetli wektor 2.000000 2.645751 3.316625.
5. Utwórz data.frame postaci

```
       wiek wzrost waga płeć
Kasia    25    177   57    K
```

```
Ania     31    163   69    K
Tomek    23    190   83    M
Piotr    52    179   75    M
Maria    76    163   70    K
Karol    49    183   83    M
Sylwia   26    164   53    K
```

Następnie spróbuj szybko zamienić płeć każdej osobie.

6. Zapoznaj się ze zbiorem danych swiss. Następnie utwórz ramkę danych zawierającą tylko wiersze 1, 2, 3, 10, 11, 12 i 13, oraz zmienne Examination, Education i Infant.Mortality.