Pakiety Matematyczne - R Zestaw 1.

Zadania z kasynem pochodzą ze strony datacamp.com

Instalacja pakietu R

• Strona główna projektu: http://www.r-project.org/

• Instalacja: http://r.meteo.uni.wroc.pl/

• (jedno z kilku) GUI RStudio: http://www.rstudio.com/

Szukanie pomocy

- `help("nazwa"), ?nazwa` – wyświetla (jeżeli jest dostępna) pomoc do nazwa
- `example(nazwa)` – wyświetla i wykonuje (jeżeli są dostępne)przykłady do nazwa
- `apropos("nazwa")` – wyświetla funkcje zawierające nazwa

W programie RStudio pomoc jest wyświetlaną w jednym z okien programu.

Skróty klawiszowe dla RStudio

Operatory i funkcje elementarne

- `#` – komentarz
- `x<-y, y->x, x=y` – przypisanie wartości y do zmiennej x (równoważne sposoby)
- `+, -, *, /, ^` – operatory arytmetyczne
- `==, !=, <=, >=, &&, , TRUE, FALSE` – operatory relacji i logiczne
- `x %% y` – reszta z dzielenia x mod y
- `x %/% y` – część całkowita z dzielenia x przez y
- `sin(x), cos(x), tan(x), . . .` , asin(x),... – funkcje trygonometryczne
- sqrt(x), abs(x), log(x) – pierwiastek kwadratowy, wartość bezwzględna, logarytm naturalny

Zadanie 0.

- Zapoznaj się z pomocą do funkcji `sin`
- Wykonać polecenie `?complex` i zapoznać się z odpowiednim tekstem pomocy
- Odkryć przeznaczenie funkcji: `factorial(x), choose(a,b)`

Zadanie 1. Zdefiniuj zmienną **kasyno** i przypisz do niej wartość **“wchodzę”.**

Wektory

W **R** podstawową strukturą danych jest wektor. Możemy je tworzyć na kilka sposobów np.:

1. `a:b` – liczby całkowite a, a+1, a+2, ..., b

```
n <- 10
1:n-1
1:(n-1)
#zwróć uwagę na różnicę
```

2. `c(a,b,c,...)` – tworzy wektor zawierający ciąg a, b, c,...
3. `seq(from, to)` – identyczne z `from:to`
4. `seq(from, to,by=step)` – tworzy ciąg liczb od from do to z krokiem step
5. `rep(x,n)` - tworzy n-elementowy wektor o wartościach x

Wektory można do siebie dodawać, ale należy czynić to uważnie. Sprawdź jaki będzie wynik działania:

```
rep(2,3)+rep(4,7)
```

Wektory mogą mieć wartości numeryczne, znakowe bądź logiczne

```
integer.vector <- c(1, 2, 3)
character.vector <- c("a", "b", "c")
logical.vector<-c(TRUE, FALSE, FALSE )
```

Co się stanie jeśli spróbujemy stworzyć wektor z elementów różnego typu?

```
c("hjh",2,TRUE)
c(TRUE,7)
```

Zadanie 2. Od tygodnia jesteś w kasynie. Twoje zwycięstwa i porażki przedstawiono poniżej.
Stwórz zmienne **poker.wektor** oraz **bandyta.wektor** i przypisz do nich wartości wygranych (jako dodatnie) i przegranych (jako ujemne). Wyświetl oba wektory.

Poker	Jednoręki Bandyta
• W poniedziałek wygrałeś $100 • We wtorek przegrałeś $80 • W środę wygrałeś $20 • W czwartek przegrałeś $120 • W piątek wygrałeś $180 • W sobotę wygrałeś $30 • W niedzielę przegrałeś $90	• W poniedziałek przegrałeś $110 • We wtorek wygrałeś $50 • W środę wygrałeś $40 • W czwartek przegrałeś $120 • W piątek przegrałeś $100 • W sobotę wygrałeś $230 • W niedzielę przegrałeś $70

Korzystając z funkcji `names()` możemy przypisywać nazwy elementom wektora.

```
przykladowy_wektor <- c("Anna", "Nowak")
names(przykladowy_wektor ) <- c("imie", "nazwisko")
```

Zadanie 3. Stwórz wektor **dni_tygodnia** zawierający nazwy kolejnych dni tygodnia. Następnie korzystając z niego nazwij elementy wektorów **poker.wektor** i **bandyta.wektor.** Wyświetl oba wektory.

Zadanie 4.

a) Ile wygrałeś/przegrałeś każdego dnia łącznie w obu grach? Przypisz wynik do wektora **zysk_dzienny**
b) Ile wyniosła całkowita wygrana/przegrana w poprzednim tygodniu? Wyniki zapisz do zmiennych **suma_poker, suma_bandyta, suma_tygodniowa.** Skorzystaj z funkcji **sum().**
c) Sprawdź, czy suma Twoich wygranych w pokera była większa niż suma wygranych w jednorękiego bandytę.

`wektor[i]` - wybieranie i -tego elementu z wektora (zaczynając od 1)
`wektor[c(i,j,k)]` **-** wybieranie i -tego, j-go i k-go elementu z wektora
`wektor[i:j]` **-** wybieranie elementów od i -tego do j-go z wektora
`wektor["nazwa"]` **-** wybieranie elementu wskazanego przez nazwę z wektora
`wektor[c("nazwa1","nazwa2")]` **-** wybieranie elementów wskazanych przez nazwy z wektora
`mean(wektor)` - średnia elementów wektora

Zadanie 5. Z wektora **poker.wektor** wybierz obserwacje

a) ze środy czwartku i piątku i zapisz je do wektora **srodkowe_poker**
b) z poniedziałku wtorku i środy i zapisz je do wektora **poczatkowe_poker**
c) z piątku soboty i niedzieli i zapisz je do wektora **koncowe_poker**

Zadanie 6. Oblicz sumy i średnie wektorów z zadania 5 i przypisz je do zmiennych **suma_poker_poczatek**, **sumapoker_srodek, suma_poker_koniec** **srednia_poker_poczatek**, **srednia_poker_srodek, srednia_poker_koniec** .

Operatory logiczne mogą być używane również na wektorach

```
wektor<-c(1,-8,3,-6)
czy_ujemne<-wektor<0 #sprawdzamy, czy elementy wektora są ujemne
czy_ujemne #wynikiem jest wektor o wartościach logicznych
ujemne=wektor[czy_ujemne]
ujemne
#możemy też skorzystać z funkcji which()
ktore<-which(wektor != 3)
ktore
wektor[ktore]
```

ZADANIE DOMOWE

Zadanie 7. Sprawdź, które z elementów wektorów **poker.wektor, bandyta.wektor** są dodatnie. Wyniki tego sprawdzenia zapisz do wektorów o wartościach logicznych **czy_ wygrana_poker**, **czy_ wygrana_bandyta** i wyświetl te wektory. Stwórz wektory **wygrane_poker**, **wygrane_bandyta** które zawierają tylko wartości dodatnie.

Zadanie 8. Korzystając z funkcji `rep` utwórz wektory:

- **x** - ciąg liczb: 1, 2, 3 powtórzony 26 razy
- **y** - ciąg liczb: 1, 1, 2, 2, 3, 3 powtórzony 13 razy ← ZADANIE DOMOWE
- **z** - ciąg liczb: 1, 1, 2, 2, 2, 3, 3, 3, 3, 3, 4, 4 o długości 100

```
w = c(3,5,1,7,3,9,2,6)
order(w)
w[order(w)]
sort(w)
cumsum(w)
diff(w)
```

Macierze

Macierze możemy tworzyć za pomocą polecenia

`matrix(data,nrow,ncol).`

Na przykład

```
matrix(0,3,3)
matrix(0,nrow=3,ncol=3) matrix(1,3,3)
```

```
matrix(c(1,2,3,4,5,6,7,8,9),3,3) matrix(1:9,3,3,byrow=TRUE)
```

Parametr data oznacza elementy macierzy (domyślnie kolumnami). Inne funkcje tworzące macierze: `diag(x), lower.tri(x), upper.tri(x)`

Aby dostać się do wybranych elementów macierzy używamy operatora `[]`

```
M = matrix(2:16,5,3)
M[1]
M[4]
M[13]
M[1,1]
M[2,1]
M[3,3]
M[3,]
M[,3]
M[c(1,4,13)]
M
```

Przypisanie realizujemy w podobnie:

```
M <- matrix(1:15,5,3)
M[1] <- 0
M[3,3] <- 0
M[5,] <- c(-1 -1 -1)
M[2:3,1:2] <- 99
M
```

Niektóre funkcje, które można zastosować na wektorach/macierzach:

1. operatory arytmetyczne i logiczne
2. length(M) – liczba elementów wektora/macierzy
3. dim(M) – wymiar macierzy
4. `min(M), max(M), mean(M), sum(M)` – element minimalny, element maksymalny, średnia, suma
5. `sort(M), sort(M,decreasing=TRUE)` – sortowanie rosnąco sortowanie malejąco

Inne polecenia warte uwagi: rev, cumsum, prod, cumpod, which.max, summary, rbind, cbind

```
M = matrix(2:16,5,3)
```

```
apply(M,1,sum) # suma elementów w wierszach
apply(M,2,min) # najmniejszy element w kolumnach
colSums(M) # suma w kolumnach, podobnie wiersze
rowMeans(M) # średnia w wierszach, podobnie w kolumnach
rbind(M) # łączenie macierzy wierszami
cbind(M) # łączenie macierzy wierszami kolumnami
```

Zadanie 9. Utwórz macierze:

- M1 - jednostkową 4x4,
- M2 - diagonalną o wyrazach: 5,44,3,22,1 na diagonali
- M3 - której pierwszy wiersz stanowi wektor **poczatkowe_poker**, drugi **srodkowe_poker**, a trzeci **koncowe_poker**.
- M4 - której pierwszą kolumnę stanowi wektor **poker.wektor**, a drugą **bandyta.wektor**

Znajdź: długość, element najmniejszy, element największy, wartość średnią i sumę elementów dla tych macierzy. Wykonać `summary(M4)`. Jak działa funkcja `summary`?

* liczbę elementów