**Ministerul Educației și Cercetării al Republicii Moldova**

**Universitatea Tehnică a Moldovei**

**Facultatea Calculatoare, Informatică şi Microelectronică**

**Departamentul Informatică şi Ingineria Sistemelor**

**RAPORT**

Lucrare de laborator nr.3

la cursul „Programarea Declarativă”

**Tema 3:** „ Funcții și grafuri”

A efectuat : **st. gr. TI-214 Buza Cătălin**

A verificat:  **M.Rusu**

**Chișinău 2023**

**Exercițiul 1:**

Creați o funcție care va calcula media unui vector cu o precizie de patru zecimale.

1. `media\_cu\_precizie <- function(vector) {`

- `media\_cu\_precizie <- function(vector)` este modul în care definim o funcție în R. În acest caz, se definește o funcție numită `media\_cu\_precizie` care va primi un argument numit `vector`.

2. `rezultat <- round(mean(vector), 4)`

- `mean(vector)` calculează media aritmetică a elementelor din `vector`.

- `round(..., 4)` ia rezultatul mediei și îl rotunjește la 4 zecimale folosind funcția `round()`. Aceasta asigură precizia specificată de patru zecimale.

3. `return(rezultat)`

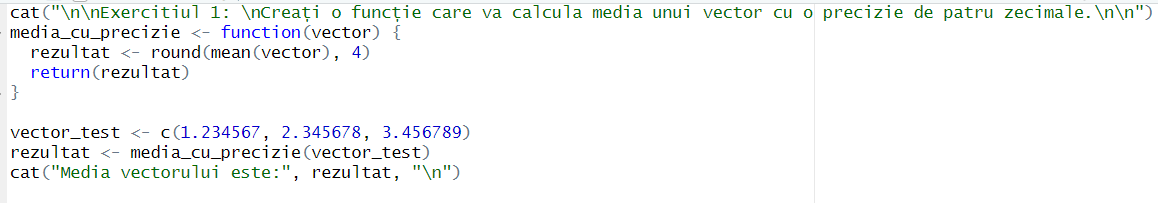
- `return(rezultat)` indică că funcția va returna valoarea calculată a mediei.

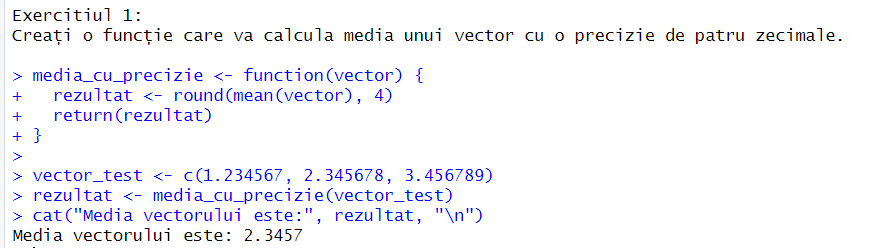
Practic, această funcție face următoarele lucruri:

- Primește un vector ca argument.

- Calculează media acestui vector cu o precizie de patru zecimale.

- Returnează rezultatul.





**Exercițiul 2:**

Calculați factorial n, n!=1×2×.. .×(n−1)×n utilizând : o buclă for(), funcția prod().

**Utilizând o buclă for():**

1. `factorial\_cu\_bucla <- function(n) {`

- Începem prin definirea unei funcții numită `factorial\_cu\_bucla`, care primește un argument `n`.

2. `rezultat <- 1`

- Inițializăm o variabilă `rezultat` cu valoarea 1. Aceasta va fi folosită pentru a calcula factorialul.

3. `for (i in 1:n) {`

- Aici începe bucla `for`. Va itera de la 1 până la `n`.

4. `rezultat <- rezultat \* i`

- În fiecare iterație, valoarea lui `i` (care crește de la 1 la `n`) este înmulțită cu `rezultat`. Astfel, `rezultat` acumulează produsul.

5. `return(rezultat)`

- Funcția returnează `rezultatul` care reprezintă factorialul lui `n`.

**Utilizând funcția prod():**

1. `factorial\_cu\_prod <- function(n) {`

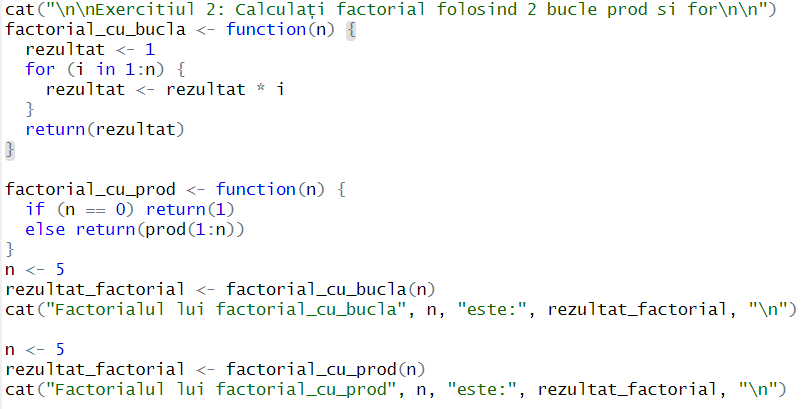
- Definim o funcție numită `factorial\_cu\_prod`, care primește un argument `n`.

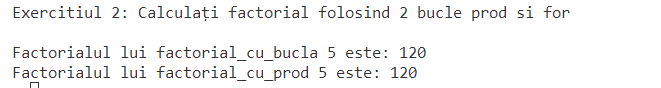
2. `if (n == 0) return(1)`

- Aici se verifică dacă `n` este 0. Dacă este, funcția returnează 1, deoarece factorialul lui 0 este 1.

3. `else return(prod(1:n))`

- Dacă `n` nu este 0, funcția `prod(1:n)` calculează produsul elementelor de la 1 la `n`, care reprezintă factorialul lui `n`.





**Exercițiul 3:**

Creați o funcție care va calcula media, mediana și abaterea standard a unui vector și va afișa

una dintre opțiunile alese „medie”, „mediană” sau „SD”(standard deviation).

**1. `calcul\_statistici <- function(vector, optiune) {`**

- Se definește funcția `calcul\_statistici` care primește doi parametri: `vector` (reprezentând vectorul de date) și `optiune` (reprezentând alegerea utilizatorului pentru ce statistică să se calculeze).

**2. Blocul `if-else`:**

- `if (optiune == "medie") { ... } else if (optiune == "mediană") { ... } else if (optiune == "SD") { ... } else { ... }`

- Aici începe o serie de condiții `if-else` pentru a verifica opțiunea aleasă de utilizator.

**3. Calculul statisticilor:**

- În fiecare ramură `if`, se calculează statisticile corespunzătoare:

- Dacă `optiune` este "medie", se calculează media cu `rezultat <- mean(vector)`.

- Dacă `optiune` este "mediană", se calculează mediana cu `rezultat <- median(vector)`.

- Dacă `optiune` este "SD", se calculează abaterea standard cu `rezultat <- sd(vector)`.

**4. Tratarea cazului invalid:**

- Dacă opțiunea aleasă nu este niciuna dintre cele trei valide, se afișează un mesaj de eroare și `rezultat` este setat la `NULL`.

**5. Verificarea și afișarea rezultatului:**

- `if (!is.null(rezultat)) { ... }`

- Se verifică dacă `rezultat` nu este nul. Dacă nu este nul, înseamnă că am obținut o statistică validă.

**6. Afișarea rezultatului:**

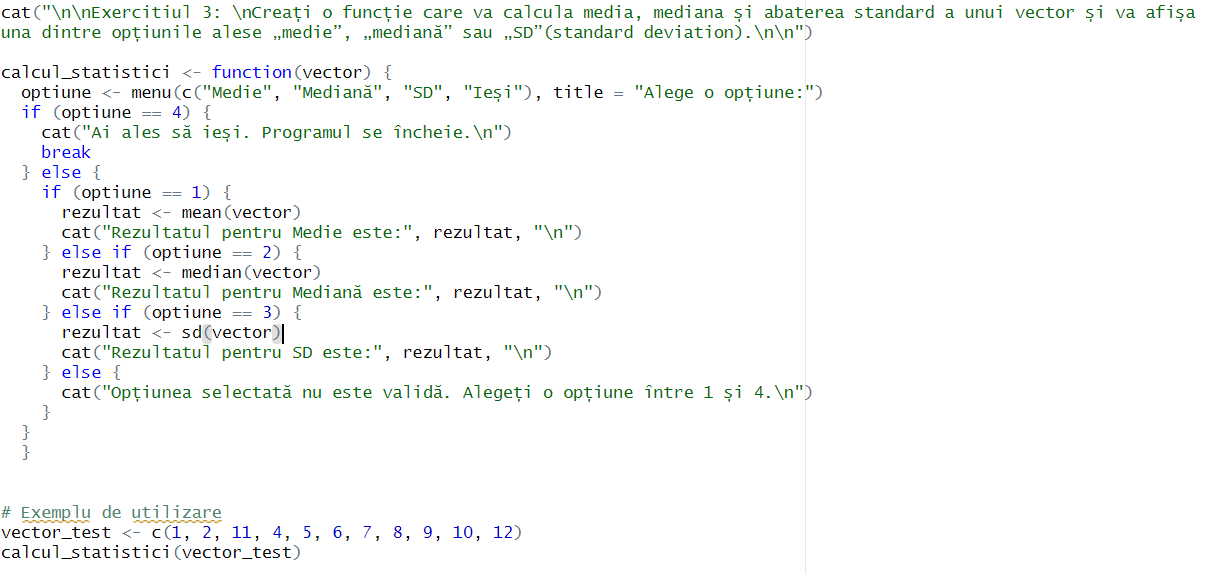
- `cat("Rezultatul pentru", optiune, "este:", rezultat, "\n")`

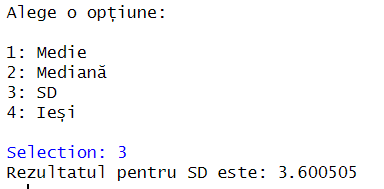
- Se folosește `cat()` pentru a afișa rezultatul, împreună cu opțiunea aleasă.

**7. Apelul funcției în exemplu:**

- `vector\_test <- c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)` definește un vector de test.

- Apelurile funcției `calcul\_statistici` sunt utilizate pentru a calcula și afișa media, mediana și abaterea standard pentru `vector\_test`, folosind diferite opțiuni.





**Exercițiul 4:**

Scrieți o funcție care înlocuiește valorile negative ale unui vector cu valorile lor absolute și

apoi afișează vectorul modificat.

**Definirea funcției:**

1. `inlocuire\_negativ\_cu\_absolute <- function(vector) {`

- Aici se definește o funcție numită `inlocuire\_negativ\_cu\_absolute`, care primește un argument `vector`.

**Înlocuirea valorilor negative cu valorile absolute:**

2. `vector\_modificat <- abs(vector)`

- Această linie de cod creează o variabilă numită `vector\_modificat`. Aceasta conține valorile absolute ale fiecărui element din `vector`, obținute cu ajutorul funcției `abs()` care calculează valoarea absolută.

**Returnarea vectorului modificat:**

3. `return(vector\_modificat)`

- Funcția returnează `vector\_modificat`, care conține valorile absolute ale elementelor din `vector`.

**Utilizarea funcției:**

4. Apelarea funcției cu un exemplu de test:

vector\_test <- c(1, -2, 3, -4, 5, -6, 7, -8, 9, -10)

vector\_modificat <- inlocuire\_negativ\_cu\_absolute(vector\_test)

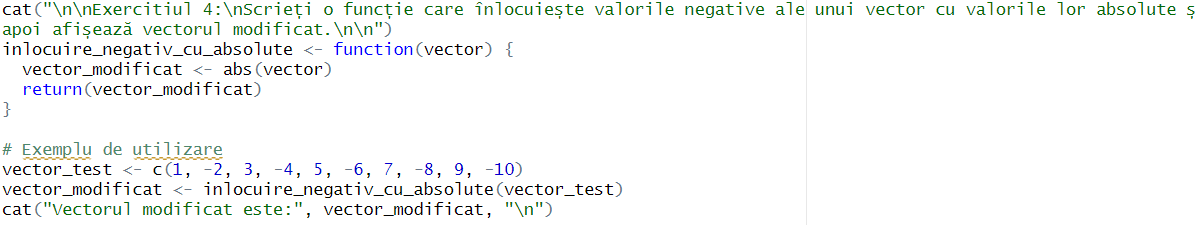
- În acest exemplu, am creat un vector numit `vector\_test` care conține atât valori pozitive, cât și negative.

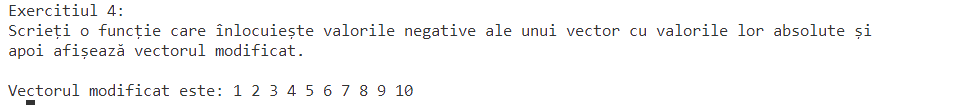
- Apoi, am apelat funcția `inlocuire\_negativ\_cu\_absolute` cu `vector\_test` ca argument. Rezultatul este salvat în variabila `vector\_modificat`.

**Afișarea rezultatului:**

5. `cat("Vectorul modificat este:", vector\_modificat, "\n")`

- Folosind `cat()`, am afișat mesajul "Vectorul modificat este:" urmat de `vector\_modificat`, care conține valorile absolute ale elementelor din vector. `\n` este un simbol pentru un nou rând, astfel încât mesajul să fie afișat pe linia următoare.





**Exercițiul 5:**

**a) Desenarea graficului funcției sinus între 0,5 și 3π (utilizați pi):**

1. `if (!requireNamespace("ggplot2", quietly = TRUE)) { install.packages("ggplot2") }`

- Verificăm dacă pachetul `ggplot2` este instalat. Dacă nu este, îl instalăm folosind `install.packages("ggplot2")`.

2. `library(ggplot2)`

- Încărcăm pachetul `ggplot2` pentru a-l putea utiliza în R.

3. `x <- seq(0.5, 3\*pi, length.out = 100)`

- Cream un vector `x` care conține 100 de valori între 0.5 și 3π folosind `seq()`. Acesta va reprezenta valorile pe axa x.

4. `y <- sin(x)`

- Calculăm valorile funcției sinus pentru fiecare valoare din vectorul `x` și le salvăm într-un vector `y`.

5. `ggplot(data = data.frame(x = x, y = y), aes(x = x, y = y))`

- Inițializăm obiectul `grafic\_sinus` utilizând datele din `x` și `y`. `aes(x = x, y = y)` specifică că `x` și `y` vor fi folosite ca atribute pentru axele x și y.

6. `geom\_line(color = "red", size = 1.5)`

- Adăugăm o linie la grafic cu funcția `geom\_line()`. Atributul `color = "red"` setează culoarea liniei la roșu, iar `size = 1.5` specifică grosimea liniei.

**b) Adăugarea titlului, culorii și grosimii liniei:**

1. `ggtitle("Graficul funcției sinus")`

- Adăugăm un titlu graficului cu `ggtitle()`.

2. `xlim(0.5, 3\*pi)`

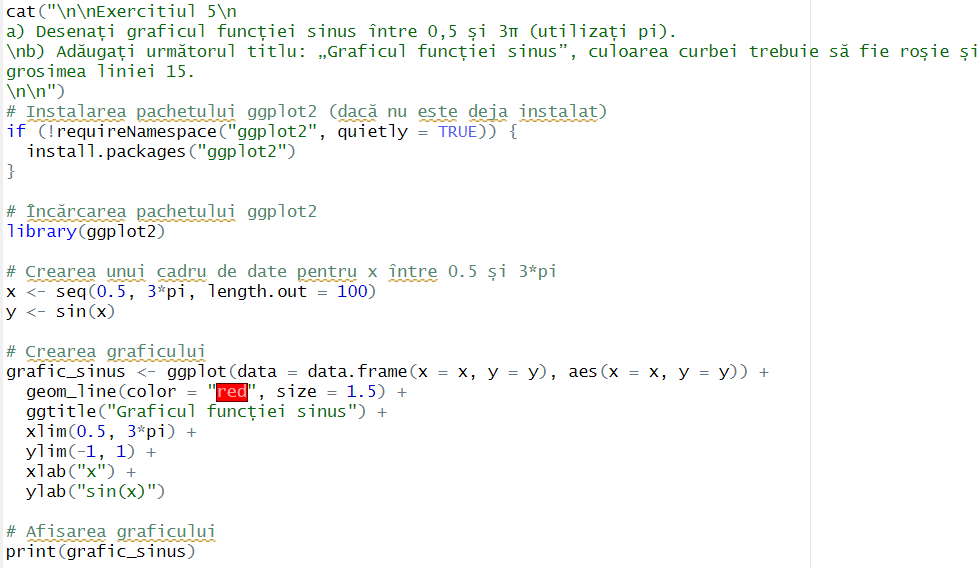
- Setăm limitele axei x între 0.5 și 3π cu `xlim()`.

3. `ylim(-1, 1)`

- Setăm limitele axei y între -1 și 1 cu `ylim()`.

4. `xlab("x")` și `ylab("sin(x)")`

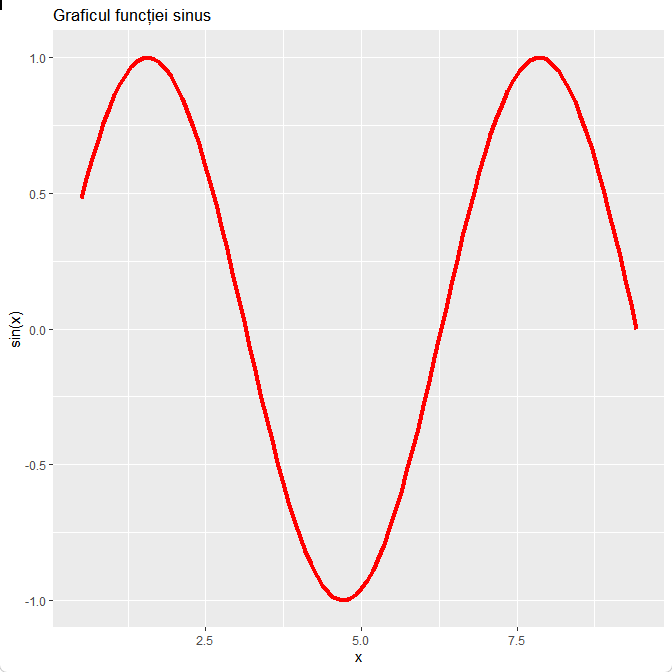
- Adăugăm etichete pentru axe x și y cu `xlab()` și `ylab()`.



**Afisarea graficului:**

1. `print(grafic\_sinus)`

- Afișăm graficul folosind `print()`, iar `grafic\_sinus` este obiectul nostru ggplot.

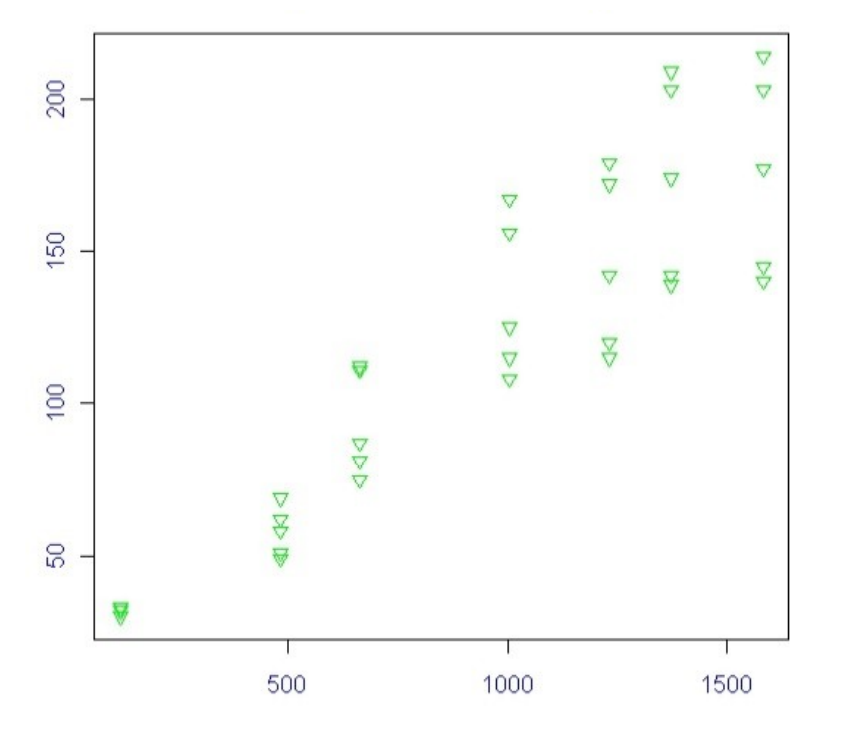


**Exercițiul 6:**

Încărcați setul de date Orange. Desenați un grafic de dispersie a vârstei variabile în funcție de

circumferința variabilă. Modificați parametrii graficului funcției (pch, col.main, sub, ylab ....)

pentru a obține următoarea reprezentare:



1. `data(Orange)`

- Această linie încarcă setul de date "Orange" din pachetul Orange. Setul de date conține informații despre portocali, inclusiv circumferința și vârsta acestora.

2. `plot(Orange$circumference, Orange$age, pch = 24, col = "blue", main = "Grafic de dispersie", xlab = "Circumferință", ylab = "Vârstă")`

- Funcția `plot()` este folosită pentru a crea un grafic de dispersie.

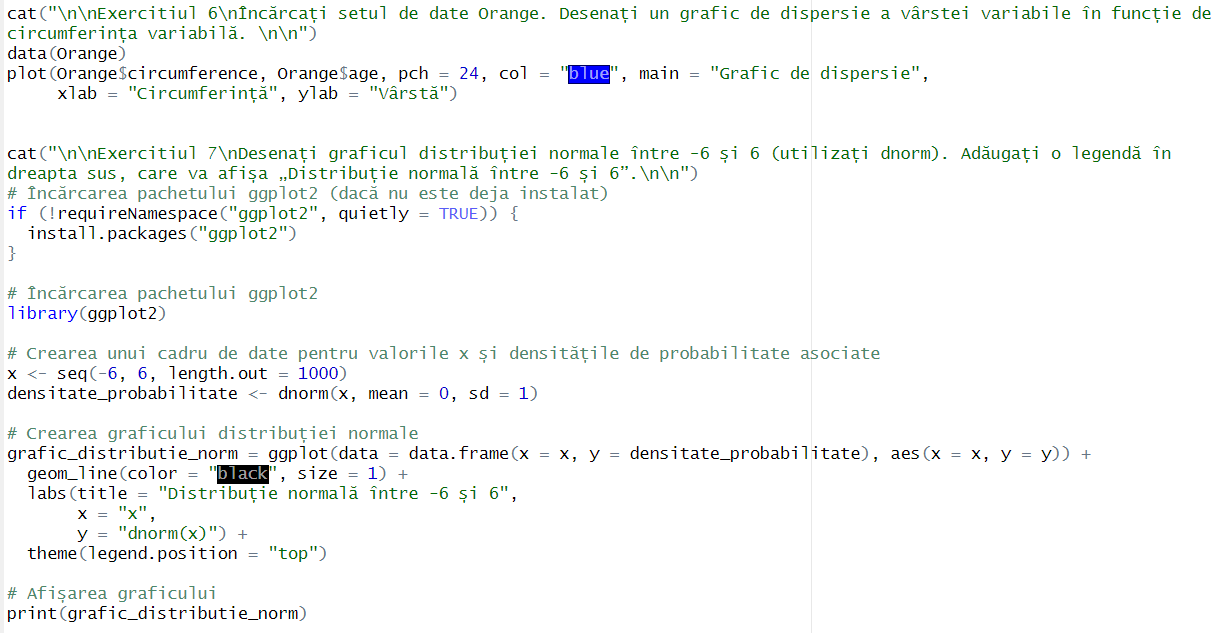
- `Orange$circumference` reprezintă circumferința portocalilor, iar `Orange$age` reprezintă vârsta.

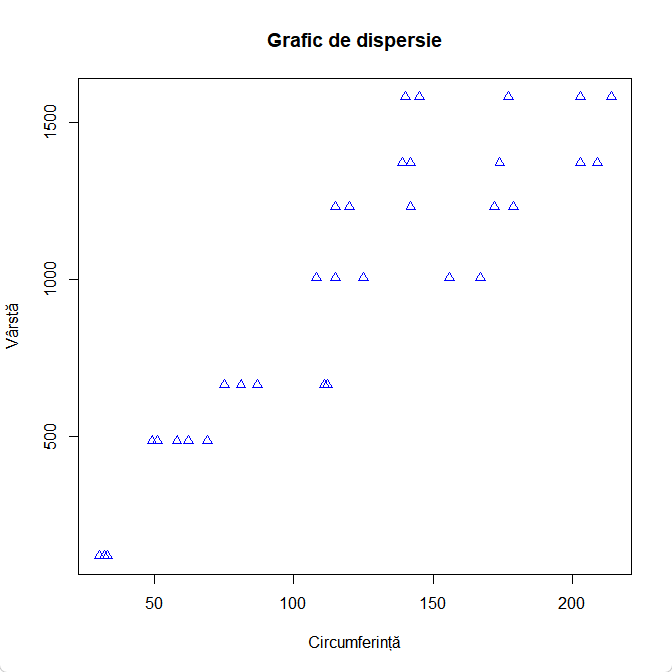
- `pch = 24` specifică că forma simbolului va fi un triunghi orientat în jos.

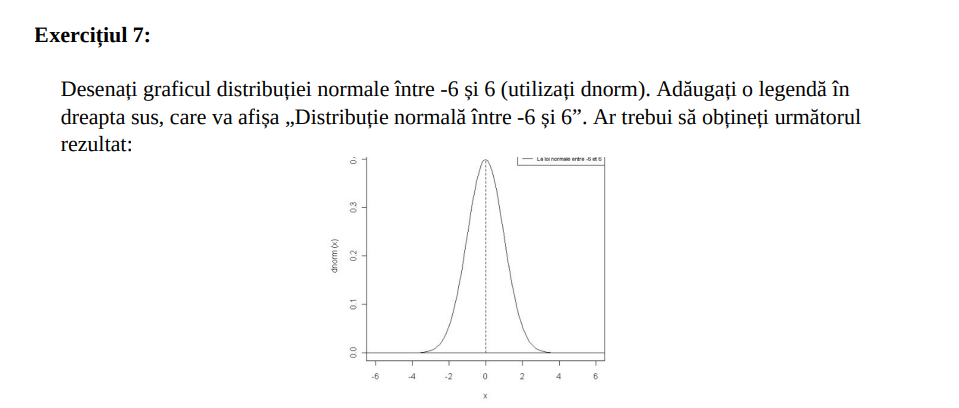
- `col = "blue"` setează culoarea punctelor la albastru.

- `main = "Grafic de dispersie"` specifică titlul principal al graficului.

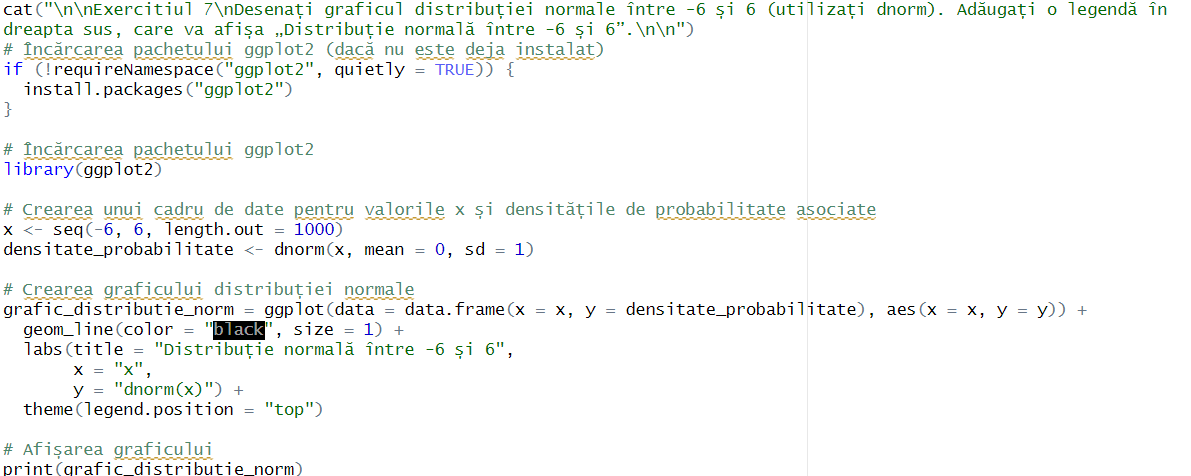
- `xlab = "Circumferință"` și `ylab = "Vârstă"` adaugă etichete pentru axe.

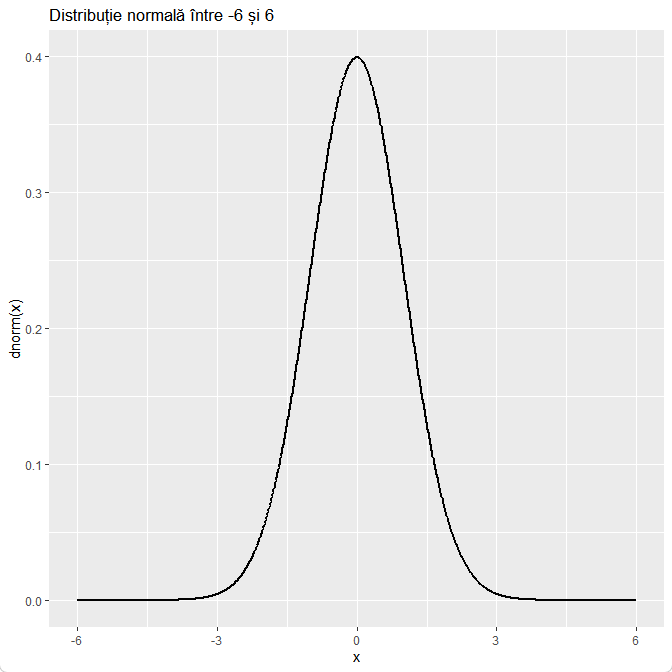


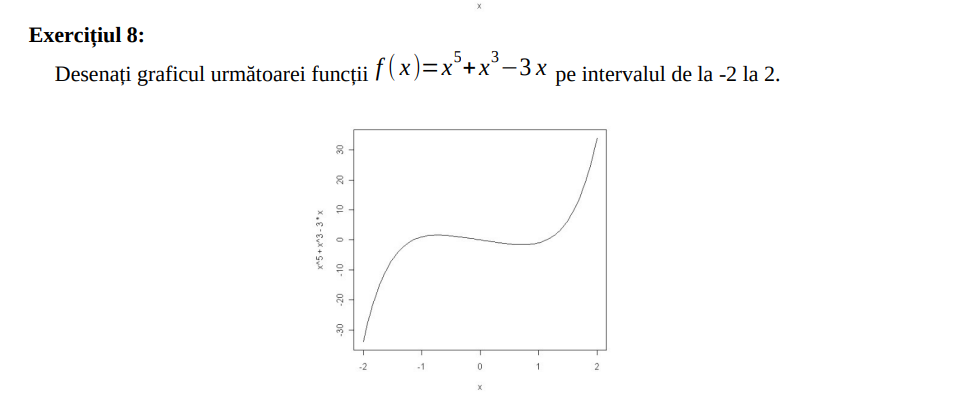




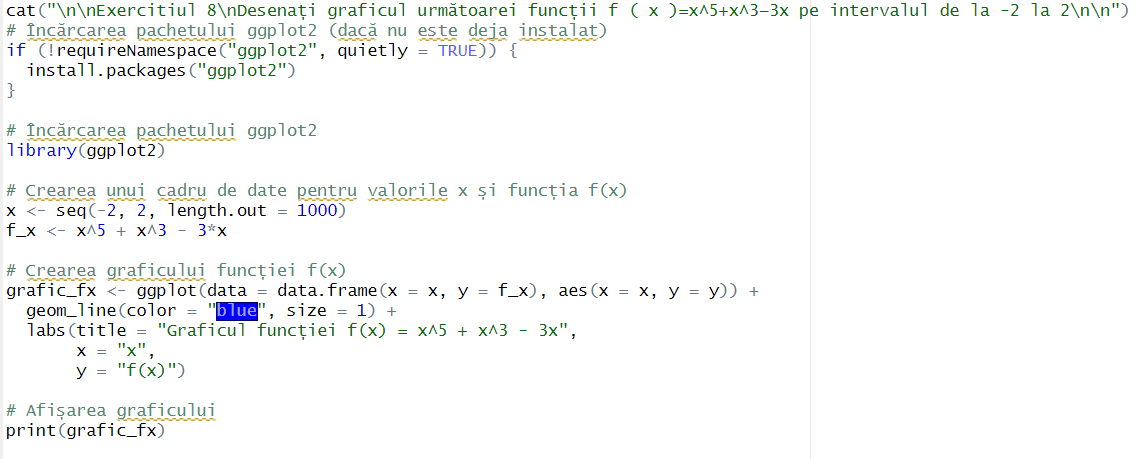
* Verificăm dacă pachetul ggplot2 este instalat. Dacă nu este, îl instalăm folosind install.packages("ggplot2").
* Încărcăm pachetul ggplot2 pentru a-l putea utiliza.
* Cream un vector x care variază de la -6 la 6 cu 1000 de puncte intermediare.
* Calculăm densitatea de probabilitate a distribuției normale standard pentru fiecare valoare din x utilizând funcția dnorm.
* Creăm un obiect grafic\_distributie\_norm folosind datele x și densitate\_probabilitate. Setăm x și y ca atribute pentru axele x și y.
* Adăugăm o linie reprezentând densitatea de probabilitate a distribuției normale.
* Adăugăm titlul și etichetele pentru axe.
* Setăm poziția legendei în partea de sus a graficului folosind theme(legend.position = "top").
* Adăugăm un text ca legendă folosind annotate(). Acest text indică "Distribuție normală între -6 și 6" și este plasat în partea de sus a graficului.
* Afișăm graficul cu print(grafic\_distributie\_norm).

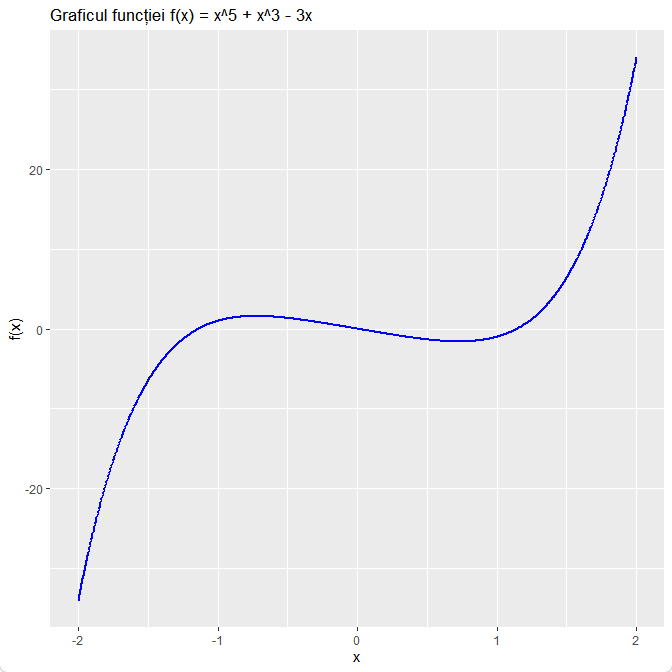






* Verificăm dacă pachetul ggplot2 este instalat. Dacă nu este, îl instalăm folosind install.packages("ggplot2").
* Încărcăm pachetul ggplot2 pentru a-l putea utiliza.
* Cream un vector x care variază de la -2 la 2 cu 1000 de puncte intermediare.
* Calculăm valorile funcției f ( x )=x^5+x^3−3x pentru fiecare valoare din x.
* Creăm un obiect grafic\_fx folosind datele x și f\_x. Setăm x și y ca atribute pentru axele x și y.
* Adăugăm o linie reprezentând funcția
* Adăugăm titlul și etichetele pentru axe.
* Afișăm graficul cu print(grafic\_fx).





**Concluzie**

În cadrul acestui laborator, am explorat utilizarea limbajului R pentru a efectua diverse operațiuni statistice și de vizualizare a datelor. Am acoperit următoarele aspecte:

1. Calcularea Statisticilor Descriptive: Am creat funcții pentru a calcula media, mediana și abaterea standard a unui set de date. Aceste statistici ne ajută să înțelegem distribuția și caracteristicile centrale ale datelor.

2. Manipularea Datelor cu R: Am învățat să lucrăm cu vectori și să efectuăm operații precum adunare, scădere, înmulțire și împărțire element cu element. Am învățat, de asemenea, să selectăm și să filtrăm datele dintr-un vector.

3. Funcții și Controlul Fluxului: Am definit funcții personalizate pentru a efectua operații specifice și am folosit bucle și declarații condiționale pentru a controla fluxul programului.

4. Vizualizarea Datelor cu ggplot2: Am creat grafice de dispersie și grafice de linii folosind pachetul ggplot2. Am personalizat aspectul graficelor prin adăugarea de titluri, etichete pentru axe și modificarea culorilor și simbolurilor.

5. Lucrul cu Seturi de Date: Am învățat cum să încărcăm și să lucrăm cu seturi de date preîncărcate, cum ar fi setul de date Orange.

6. Desenarea Graficului unei Funcții Matematice: Am desenat graficul funcției matematice \(f(x) = x^5 + x^3 - 3x\) pe un interval specificat.

Prin intermediul acestor exerciții, am acumulat cunoștințe esențiale pentru manipularea și analiza datelor utilizând R, precum și pentru crearea de vizualizări eficiente. Aceste abilități vor fi utile în viitor pentru a aborda probleme complexe de analiză a datelor și pentru a comunica rezultatele într-un mod clar și eficient.