

Laborator 1 - R și RStudio

R este un mediu dedicat analizei statistice introdus în 1996; este întreținut și dezvoltat în lumea academică și are avantajul de a fi open-source spre deosebire de alte pachete statistice cunoscute (Minitab, SPSS etc).

Cu R se poate lucra din linia de comandă sau cu ajutorul unei interfețe grafice; în cele ce urmează vom folosi o astfel de interfață grafică: RStudio care este open-source și poate fi folosită pe Linux, Windows sau Mac.

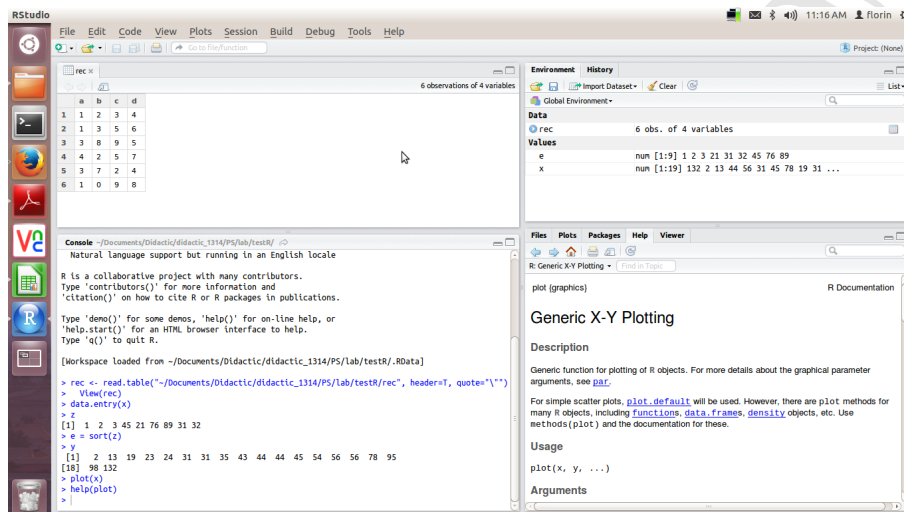


Figure 1: RStudio screenshot

RStudio conține (vezi figura de mai sus) uzual patru panouri (începând din stânga sus, în sens geometric): unul în care se pot edita și executa scripturile R (care conțin funcții și comenzi) sau fișierele de date, unul dedicat vizualizării variabilelor, istoricului comenzilor etc., în al treilea apar graficele, se poate parcurge help-ul atașat comenzilor și conținutul fișierelor din directorul curent, iar ultimul panel este cel care conține prompterul liniei de comandă (>) - aici se pot executa comenzi R.

0.1 Sesiunea RStudio

O sesiune trebuie să înceapă prin setarea directorului de lucru: **Session** → **Set Working Directory** → **Choose Directory** și se va termina prin salvarea spațiului de lucru (în fereastra de dialog, la întrebarea "Save workspace image to `/.RData?`" alegeți "Save" din **Session** → **Save Workspace As**).

0.2 Variabile și tipuri

În R variabilele sunt uzual vectori sau matrici. Orice variabilă poate fi vizualizată prin simpla apelare a numelui. Tipurile folosite sunt numeric, șiruri de caractere (de exemplu " a43fdt") și boolean (TRUE sau T, FALSE sau F)

Asignarea Există două simboluri pentru asignare în R: = și <- (fără spații, se recomandă folosirea lui pentru compatibilitate cu versiuni mai vechi de R).

Crearea unui vector Avem mai jos trei metode diferite de a crea un vector:

- prin concatenare folosind funcția `c()`,

- ca o secvență de numere întregi consecutive - pentru acest exemplu am afișat conținutul vectorului care este indexat începând cu 1 (la afișare fiecare linie va fi introdusă prin indicele primului element de pe acea linie - dacă vectorul încapă pe o singură linie va apare doar [1])
- sau construit ca o secvență a cărei îi sunt indicate numărul inițial, cel final și numărul de termeni cu ajutorul funcției *seq()*

```
> x = c(1, 3, 2, 15, 6, 21, 34, 54, 7)
> x = c(T, T, F, T, F)
> x
[1] TRUE TRUE FALSE TRUE FALSE
> x = -5:13
> x
[1] -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7 8 9
[16] 10 11 12 13
> x = seq(-3, 3, length=100)
```

Elementele unui vector pot fi accesate în felul următor

```
> x = c(23, 21, 32, 25, 34, 19, 32, 45, 67)
> x[4] # al 4-lea element
[1] 25
x[2:6] # elementele de la 2 până la 6 inclusiv
[1] 21 32 25 34 19
x[-3] # toate elementele mai puțin al 3-lea
[1] 23 21 25 34 19 32 45 67
```

Un vector poate fi editat folosind funcția *data.entry(vector)*.

0.3 Operații aritmetice și funcții predefinite

Se pot face direct din linia de comandă folosind valori numerice sau variabile

```
> sin(1)
[1] 0.841471
> log(2)
[1] 0.6931472
> x = 3
> x^2
[1] 9
> exp(x)
[1] 20.08554
```

Operațiile efectuate cu vectori sunt de obicei la nivelul fiecărei componente

```
> x = c(1, 3, 2, 15, 6, 21, 34, 54, 7)
> y = c(22, 11, 32, 25, 54, 13, 27, 36, 2)
> x^2
[1] 1 9 4 225 36 441 1156 2916 49
> x + y
[1] [1] 23 14 34 40 60 34 61 90 9
```

R conține în general funcții matematice și statistice care pot manipula vectori, matrici sau variabile simple

```
> x <- c(1, 3, 2, 15, 6, 21, 34, 54, 7)
> length(x)
> [1] 9
> sort(x)
> [1] 1 2 3 6 7 15 21 34 54
> sqrt(x)
[1] 1.000000 1.732051 1.414214 3.872983 2.449490
[6] 4.582576 5.830952 7.348469 2.645751
> exp(x)
[1] 2.718282e+00 2.008554e+01 7.389056e+00
[4] 3.269017e+06 4.034288e+02 1.318816e+09
[7] 5.834617e+14 2.830753e+23 1.096633e+03
```

Informații despre o funcție pot fi obținute folosind `help(ume_funcție)` în linia de comandă.

0.4 Grafice

Un grafic simplu în două coordonate poate fi făcut astfel: spre exemplu pentru reprezentarea funcției logaritm în baza 2, vom face o reprezentare între 0.001 și 10, alegând un număr suficient de valori intermediare, 200:

```
> x = seq(0.001, 10, length = 200)
> y = log2(x)
> plot(x, y, type = 'l', main='grafic', sub = 'subtitlul', xlab = 'axa x', ylab = 'axa y')
# tipul graficului: linie
```

Un alt tip de grafic este cel care reprezintă coloane (bars) de înălțime egală cu valorile date în argument. Spre exemplu putem reprezenta în acest fel funcția de masă de probabilitate a repartiției binomiale $B(20, 0.4)$

```
> n = 20
> x = seq(0, n, 1) # x conține valorile dela 0 la 20
> y = (dbinom(x, n, 0.4)
> barplot(y, space = 0, main='barplot', sub = 'subtitlul', xlab = 'axa x', ylab = 'axa y')
# între coloane spațiul este zero
```

0.5 Funcții definite de către utilizator

O funcție poate fi definită din linia de comandă astfel: să presupunem că dorim să calculăm dispersia unei distribuții

```
> dispersie = function (x, p) {
+ media = sum(p*x);
+ dispersie = sum(p*(x - media)^2);
+ return (dispersie)
+ }
> y = c(23, 32, 31, 27, 27, 33, 25, 21)
> q = c(1/8, 1/16, 1/8, 1/16, 1/8, 1/16, 1/8, 5/16)
> dispersie(y, q)
```

Dar, o astfel de funcție poate fi scrisă într-un script astfel **File** → **New File** → **R Script** și în fereastra de editare se scrie codul

```
dispersie = function(x, p) {  
  media = sum(p*x);  
  dispersie = sum(p*(x - media)^2);  
  return (dispersie)  
}
```

RStudio. După editare, scriptul este salvat (**Ctrl+S**) cu un nume de tipul "my_script.R" și este încărcat cu **Code** → **Source File** (**Ctrl+Shift+O**) sau din linia de comandă cu **source(script_file)**.

În același script funcția astfel scrisă poate fi și executată pentru anumite argumente, de exemplu putem adăuga la script

```
> y = c(23, 32, 31, 27, 27, 33, 25, 21)  
> q = c(1/8, 1/16, 1/8, 1/16, 1/8, 1/16, 1/8, 5/16)  
> dispersie(y, q)
```

RStudio. O dată încărcat scriptul, o funcție care face parte din acest script se poate executa din linia de comandă: **dispersie(y, q)** sau din fereastra de editare astfel: se selectează liniile dorite a fi executate și **Ctrl+Enter**, iar scriptul în întregime se execută cu **Ctrl+Alt+R**.

O funcție poate fi modificată în fereastra de editare a scriptului sau din linia de comandă cu *fix(function_name)*

```
> fix(dispersie)
```

0.6 Manipularea fișierelor cu date

Dacă un fișier "my_file" (aflat în directorul de lucru, altfel trebuie adăugată și calea relativă) conține un singur vector de date (fără antet), el poate fi citit și transformat într-un vector astfel

```
> x = scan("my_file")
```

Dacă fișierul conține un antet (să spunem că două dintre coloane sunt numite "col1" și "col2"), atunci procedăm astfel

```
> y = read.table("my_file", header = T) # acest obiect conține și antetul  
> x1 = y[,"col1"] # acest vector conține doar datele numerice din coloana "col1"  
> x2 = y[,"col2"] # acest vector conține doar datele numerice din coloana "col2"
```

Pot fi citite și fișiere de tip .csv (**comma separated values**):

```
> x = read.csv(file="date.csv", header = T)
```

0.7 Structuri iterative și de control

R conține structuri standard pentru iterații și de control utile în scrierea funcțiilor:

```
> if (condition){  
+   statement  
> }else  
+   {  
+   alternative  
+   }
```

```
> for (var in sequence){  
+   statement  
> }
```

```
> while (condition){  
+   statement  
> }
```

Următoarea funcție folosește astfel de structuri:

```
vector_sqrt = function(x) {  
  for(i in 1:length(x)) {  
    if(x[i] > 0)  
      x[i] = sqrt(x[i])  
    else  
      x[i] = sqrt(-x[i])  
  }  
}
```

Exerciții propuse

1. Introduceți următoarele date (care reprezintă valoarea facturii de telefon de-a lungul unui an, în \$) într-un vector numit b :

46 33 39 37 46 30 48 32 49 35 30 48

Datele pot fi introduse de la tastatura astfel (exemplu)

```
> b = scan(n = 7)
```

Scieți o funcție care aplicată unui vector b să determine factura cu valoare maximă, cea cu valoare minimă, media facturilor, suma totală plătită de-a lungul anului, numărul de luni cu o factură cu valoare sub 35\$ și procentul tuturor facturilor cu valoare peste 40\$.

2. Pentru un vector dat x , scrieți câte o funcție care să determine și să returneze un nou vector care să conțină

(a) $x_i / (\sum_{i=1}^n x_i)$, pentru orice i .

(b) $\frac{x_i - \min x}{\max x}$, pentru orice i ;

(Un vector nou se crează cu $vector()$. n este lungimea lui x .)

3. Creați în directorul de lucru un fișier numit "vector.txt" care să conțină un vector (coloană) și modificați funcțiile de mai sus în așa fel încât să citească vectorul din fișier. (Funcțiile vor avea ca parametru numele fișierului.)
4. Scrieți o funcție care să reprezinte grafic densitatea repartiției binomiale $B(n, p)$, folosind $barplot()$ și apoi aplicați această funcție pentru $(18, 0.25)$, $(40, 0.5)$ și $(30, 0.8)$. (Funcția va avea parametrii n și p .)
5. Scrieți o funcție care să determine probabilitatea maximă a repartiției binomiale $B(n, p)$. (Funcția va avea parametrii n și p .)

6. Scrieți o funcție care să creeze un vector care să conțină primele $(n + 1)$ probabilități ale repartiției $Geometric(p)$ și să le reprezinte grafic. (Funcția va avea parametrii p și n .)
7. Scrieți o funcție care să citească cele două coloane (numite "AA" și "BB") ale fișierului "test.txt" în doi vectori x și y și apoi reprezentați grafic perechile de puncte (x_i, y_i) , folosind $plot()$. încercați pe rând diverșii parametri ai acestei funcții. (Folosiți *help()*.)