Laborator 1 - R şi RStudio

R este un mediu dedicat analizei statistice introdus în 1996; este întreţinut şi dezvoltat în lumea academică şi 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 interfeţe grafice; în cele ce urmează vom folosi o astfel de interfaţă grafică: RStudio care este open-source şi poate fi folosită pe Linux, Windows sau Mac.

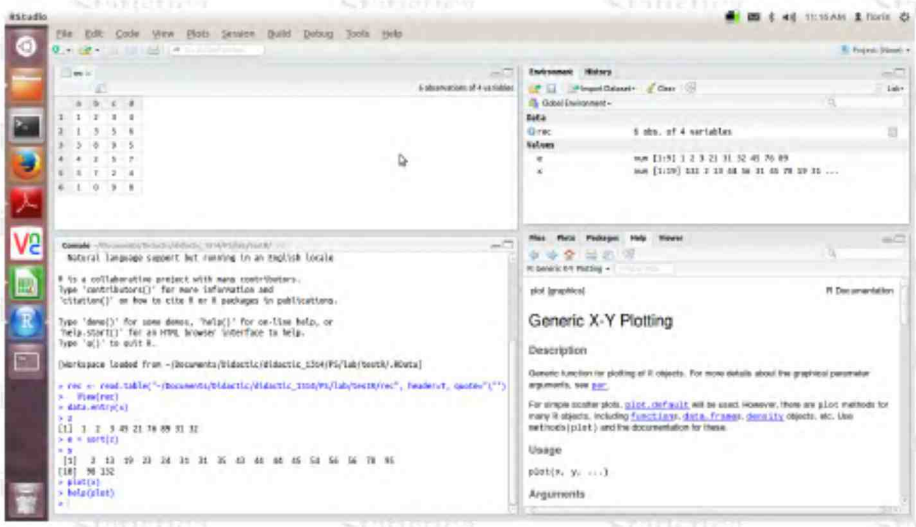

Figure 1: RStudio screenshot

RStudio conţine (vezi figura de mai sus) uzual patru paneluri (începând din stânga sus, în sens geometric): unul în care se pot edita şi executa scripturile R (care conţin funcţii şi comenzi) sau fişierele de date, unul dedicat vizualizării variabilelor, istoricului comenzilor etc., în al treilea apar graficele, se poate parcurge help-ul ataşat comenzilor şi conţinutul fişierelor din directorul curent, iar ultimul (sic!) este cel care conţine 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** şi se va termina prin salvarea spaţiului de lucru (în fereastra de dialog, la întrebarea "Save workspace image to /.RData?" alegeţi "Save").

0.2 Variabile şi tipuri

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

Asignarea Există două simboluri pentru asignare în R: = şi <− (fără spaţii, 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 funcţia $c()$,

- ca o secvenţă de numere întregi consecutive - pentru acest exemplu am afişat conţinutul vectorului care este indexat începând cu 1 (la afişare fiecare linie va fi introdusă prin indicele primului element de pe acea linie - dacă vectorul încape pe o singură linie va apare doar [1])
- sau construit ca o secvenţă cărei îi sunt indicate numărul iniţial, cel final şi numărul de termeni cu ajutorul funcţiei *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 puţin al 3-lea
[1] 23 21 25 34 19 32 45 67
```

Un vector poate fi editata folosind funcţia *data.entry*(*vector*).

0.3 Operaţii aritmetice şi funcţii 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
```

Operaţiile 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 conţine în general funcţii matematice şi 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
```

Informaţii despre o funcţie pot fi obţinute folosind *help*(*nume_funcţie*) în linia de comandă.

0.4 Grafice

Un grafic simplu în două coordonate poate fi făcut astfel: spre exemplu pentru reprezentarea funcţiei logaritm în baza 2, vom face o reprezentare între 0.001 şi 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 = 'subtilul', xlab ='axa x', ylab = 'axa y')
# tipul graficului: linie
```

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

```
> n = 20
> x = seq(0,n,1) # x conţine 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 spaţiul este zero
```

0.5 Funcţii definite de către utilizator

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

```
> 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 funcţie poate fi scrisă într-un script astfel **File** → **New File** → **R Script** şi în ferestra 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" şi este încărcat cu **Code → Source File** (**Ctrl+Shift+O**) sau din linia de comandă cu **source(script_file)**.

În acelaşi script funcţia astfel scrisă poate fi şi 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 funcţie 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 şi **Ctrl+Enter**, iar scriptul în întregime se execută cu **Ctrl+Alt+R**.

O funcţie poate fi modificată în fereastra de editare a scriptului sau din linia de comandă cu $fix(function\_name)$

```
> fix(dispersie)
```

0.6 Manipularea fişierelor cu date

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

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

Dacă fişierul conţine un antet (să spunem că două dintre coloane sunt numite "col1" şi "col2"), atunci procedăm astfel

```
> y = read.table("my_file", header = T) # acest obiect conţine şi antetul
> x1 = y[['col1']] # acest vector conţine doar datele numerice din coloana "col1"
> x2 = y[['col2']] # acest vector conţine doar datele numerice din coloana "col2"
```

Pot fi citite şi fişiere de tip .csv (comma separated values):

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

0.7 Structuri iterative şi de control

R conţine structuri standard pentru iteraţii şi de control utile în scrierea funcţiilor:

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

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

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

Următoarea funcţie foloseşte 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])
  }
}
```

Exerciţii propuse

1. Introduceţi 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

   ```
   > b =scan()
   ```

   Determinaţi apoi factura cu valoare maximă, cea cu valoare minimă, suma totală plătită de-a lungul anului, numărul de luni cu o factură cu valoare peste 40\$ şi procentul tuturor facturilor cu valoare sub 35\$.

2. Pentru un vector dat x, scrieţi câte o funcţie care să determine şi să returneze un nou vector care să conţină

   (a) $\ln(x_i)$, pentru orice i.

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

   (Un vector nou se crează cu *vector*().)

3. Creaţi în directorul de lucru un fişier numit "vector.txt" care să conţină un vector (coloană) şi modificaţi funcţiile de mai sus în aşa fel încât să citească vectorul din fişier. (Funcţiile vor avea eventual ca parametru numele fişierului.)

4. Scrieţi o funcţie care să reprezinte grafic densitatea repartiţiei binomiale $B(n, p)$, folosind *barplot*() şi apoi aplicaţi această funcţie pentru $(25, 0.3)$, $(50, 0.7)$ şi $(35, 0.5)$. (Funcţia va avea parametrii n şi p.)

5. Scrieţi o funcţie care să determine probabilitatea maximă a repartiţiei binomiale $B(n, p)$. (Funcţia va avea parametrii n şi p.)

6. Scrieţi o funcţie care să creeze un vector care să conţină primele $(n+1)$ probabilităţi ale repartiţiei $Poisson(\lambda)$. (Funcţia va avea parametrii λ şi n.)

7. Scrieţi o funcţie care să citească cele două coloane (numite "AA" şi "BB") ale fişierului "test.txt" în doi vectori x şi y şi apoi reprezentaţi grafic perechile de puncte (x_i, y_i), folosind *plot*(). încercaţi pe rând diverşii parametri ai acestei funcţii.