Analiza datelor în R

Curs 3

Elemente de grafică în R

Fie x, y doi vectori numerici, $x = (x[1], x[2], \ldots, x[n])$, $y = (y[1], y[2], \ldots, y[n])$.

Comanda

```
plot(x,y)
```

reprezintă grafic punctele de coordonate $(x[1], y[1])$, $(x[2], y[2])$, $\ldots$, $(x[n], y[n])$.

Exemplu:

```
x=seq(0,10*pi,by=0.1)
y=sin(x)
plot(x,y)
```

Argumente pentru plot

- type="p" (point - implicit) sau "l" (line)
- main="titlu grafic"
- sub="subtitlu"
- xlab="nume axa Ox", ylab="nume axa Oy"
- col="nume culoare" sau valoare numerică
- xlim=c(valStart,valFinal), ylim=c(valStart,valFinal) - domeniu de reprezentare orizontal/vertical

Pentru grafice de tip *point*:

- pch=valoare numerică (plotting character) - tip simbol
- cex=valoare numerică (character expansion) - dimensiune simbol

Pentru grafice de tip *line*:

- lty="solid/dashed/dotted/dotdash/..." (line type)
- lwd=valoare numerică (line width)

Elemente adăugate la grafice existente

- Puncte:
  `points(vectorAbscise,vectorOrdonate)`
- Segmente:
  `lines(vectorAbscise,vectorOrdonate)`
- Etichete text:
  `text(vectorAbscise,vectorOrdonate,`
  `labels=vectorEtichete)`
- Drepte:
  `abline(a=a0,b=b0)` - dreapta $y = a0 + b0 \cdot x$
  `abline(h=h0)` - dreapta orizontală $y = h0$
  `abline(v=v0)` - dreapta verticală $x = v0$

Parametri grafici

- Marginile exterioare ale graficului sunt numerotate de la 1 la 4, în sensul acelor de ceasornic, începând cu cea de jos.
- Dimensiunea marginilor, în linii de text, poate fi modificată cu `par(mar=c(n1,n2,n3,n4))`
- Suprapunere grafice:
  `par(new=TRUE)`
- Reprezentarea mai multor grafice alăturate:
  `par(mfrow=c(nrLinii, nrColoane))`
  - graficele vor fi dispuse pe *nrLinii* linii şi *nrColoane* coloane.
- Pentru detalii suplimentare: `?par`

Exportarea graficelor

Graficele pot fi salvate în diverse formate (JPEG, PNG, TIFF, PDF etc).

Exportarea se face cu

`jpeg("numeFisier.jpeg")` (png, tiff, pdf, ...)

```
...instructiuni care creeaza imaginea...
graphics.off( )
```

Elemente de programare

1. **for**

Sintaxa:

```
for(contor in vector) {instructiuni}
```

- se execută setul de instrucţiuni pentru valorile contorului egalate succesiv cu elementele vectorului

2. **while**

Sintaxa:

```
while(conditie) {instructiuni}
```

Exerciţii:

1. Să se genereze primii 20 de termeni din şirul lui Fibonacci.
2. Să se genereze toţi termenii şirului lui Fibonacci mai mici sau egali cu 1000.

Elemente de programare

3. **if**

Sintaxa:

```
if(conditie){
instructiuniTRUE
} else {
instructiuniFALSE
}
```

Se pot folosi valori numerice în locul condiţiei. Acestea vor fi interpretate ca:

- 0 = FALSE
- $\neq$0 = TRUE
- NA - eroare

Elemente de programare

4. **break**
Produce ieşirea dintr-o instrucţiune repetitivă.

5. **next**
Întrerupe iteraţia curentă şi revine la începutul buclei.

6. **repeat**

Sintaxa:
`repeat {instructiuni}`

Se repetă nelimitat grupul de instrucţiuni specificat; întreruperea se poate face dacă în acest grup de instrucţiuni există
`if (conditie) break`

Definirea funcţiilor

Sintaxa:

```
numeFunctie=function(argumente){
instructiuni;
}
```

- În R, orice funcţie produce un singur output, care se returnează cu `return(...)`. Dacă această instrucţiune lipseşte din funcţie, se returnează ultima valoare calculată.
- Variabilele definite în consolă sunt globale. Variabilele definite într-o funcţie sunt locale.
- Se pot defini funcţii în interiorul altor funcţii.
- Pentru a simplifica utilizarea funcţiei, unele argumente pot avea valori setate implicit.
- "Repararea" definiţiei unei funcţii se poate face cu `fix(numeFunctie)`

Optimizarea programelor R

- R este optimizat pentru a lucra cu vectori. Operaţiile vectorizate sunt în general mult mai rapide decât dacă ar fi executate pe câte un element odată.
- Funcţia `system.time({secventa})` măsoară timpul de execuţie al secvenţei argument.

Exemplu:
Fie vectorii `x=1:100000`, `y=1:100000`. Vom calcula `x+y`.

a)
```
z=c()
for (i in 1:100000) z=c(z,x[i]+y[i])
```

b)
```
z=numeric(100000)
for (i in 1:100000) z[i]=x[i]+y[i]
```

c)
```
z=x+y
```

Funcţia *sample*

- Selectează aleator elemente dintr-un vector dat.
- Sintaxa:
  `sample(vector, nrElemente, replace=T/F, prob=p)`
  unde:
  - `replace=T` sau `F` stabileşte dacă selecţia se face cu sau fără revenire;
  - `p` este un vector ce conţine probabilităţile de selecţie pentru fiecare componentă din `vector`. Dacă argumentul lipseşte, orice componentă din `vector` poate fi selectată cu aceeaşi probabilitate.

Exemple:

a) Simularea aruncării cu zarul de 10 ori.

b) Numărarea aşilor dintr-o mână de poker.