

**Verificarea dacă o funcție introdusă de utilizator este densitate de probabilitate.**

Dificultate întâmpinată: funcția `integrate` returnează valori eronate pentru integranți cu valoarea 0 într-o mare parte a domeniului. Ca remediu, am ales să transmitem printr-un parametru suportul funcției de integrat (detalii în comentariul din cod).

De asemenea, în cazul funcțiilor pe ramuri, `Vectorize()` devine o necesitate: evaluarea condițiilor din if-uri (de exemplu,  $x > 0$ ) ia în considerare doar primul element al vectorului Boolean  $x > 0$ . Apare astfel conflict cu modul de lucru al procedurii „integrate”, care evaluează funcția-argument pe un vector de mostre, nu punct-cu-punct.

Definiția funcției este următoarea:

```

1      # Suportul functiei f este reuniunea intervalelor specificate prin
      # parametrul
2      # sup - o lista de vectori de cate doua elemente, reprezentand
      # extremitati de
3      # interval. f este densitate de probabilitate - deci dp(f, sup)==
      # TRUE - daca:
4      # a) f(x) >= 0 pentru orice x din suport,
5      # b) suma integralelor de f peste fiecare interval din sup este 1.
6
7      # Din documentatia pentru integrate: "f must accept a vector of
      # inputs and
8      # produce a vector of function evaluations at those points".
      # Considerand ca
9      # utilizatorul introduce functia dorita in regim scalar -> scalar,
      # aplicand
10     # Vectorize() se obtine argumentul dorit pentru integrate.
11     dp <- function(f, sup) {
12
13         sum <- 0
14         for (i in sup) {
15             tryCatch(
16                 sum <- sum + integrate(Vectorize(f), i[1], i[2], abs.tol = 0)$
                    value == 1,
17                 error = function(e) {
18                     print(sprintf("Integrala divergenta in intervalul [%.2f, %.2f]!", i[1], i[2]))
19                     return(FALSE)
20                 })
21
22             if (i[1] == -Inf && i[2] == Inf) {
23                 i[1] <- -1000
24                 i[2] <- 1000
25             }
26             else if (i[1] == -Inf)
27                 i[1] <- i[2] - 1000
28             else if (i[2] == Inf)
29                 i[2] <- i[1] + 1000
30
31             if (any(sapply(seq(i[1], i[2], length.out = 1000), f) < 0)) {

```

```
32         print(sprintf("Valoare negativa in intervalul [%f,%f]!",  
33             i[1], i[2]))  
33         return(FALSE)  
34     }  
35 }  
36 sum == 1  
37 }
```

Exemple:

```
> dp(function(x) 3*x^2, list(c(0, 1)))  
[1] TRUE  
  
> f <- function(x) if (x < 0) 1 + x else 1 - x  
> dp(f, list(c(-1, 1)))  
[1] TRUE  
  
> dp(function(x) x, list(c(1, 3), c(-1, 0)))  
[1] "Valoare negativa in intervalul [-1.00,0.00]!"  
[1] FALSE  
  
> dp(function(x) x, list(c(0, Inf)))  
[1] "Integrala divergenta in intervalul [0.00,Inf]!"  
[1] FALSE
```