

# 1 Cerința 7

7) Crearea unei funcții **P** care permite calculul diferitelor tipuri de probabilități asociate unei variabile aleatoare continue(similar funcției **P** din pachetul **discreteRV**)

Pentru a calcula probabilități pe variabile aleatoare continue, am definit funcția *P* ca o metodă a clasei **contRV**, astfel:

```
setMethod("P", "contRV",
function (object) {
  return (integrala(object)) # integreaza pe suport
})
```

După cum se poate observa, parametrul funcției este un obiect de tip **contRV**. Acest lucru poate părea ciudat la prima vedere; ar fi inutil să putem calcula doar probabilități de tipul  $P(X)$ , intrucât rezultatul ar fi întotdeauna egal cu 1. În contextul pachetului, însă, un obiect de tip **contRV** nu reprezintă întotdeauna o variabilă aleatoare propriu-zisă. Mai precis, orice obiect poate fi rezultatul unei expresii de tipul:  $X \leq x$ ,  $(X \leq x) \cap (Y \geq y)$ ,  $(X < a) \cup (X > b)$  etc. Astfel, prin evaluarea expresiilor, restrângem domeniul pe care integrăm densitățile în calculul probabilităților și să păstrăm notații cât mai apropiate de cele matematice(detalii în exemplele de la sfârșit).

Pentru a evalua expresiile, am suprîncărcat următorii operatori:

```
1 setMethod("<", c("contRV", "numeric"), function (e1, e2) {
2   comp(e1, e2, "<=") #  $P(X < x) = P(X \leq x)$ 
3 })
4 setMethod("<=", c("contRV", "numeric"), function (e1, e2) {
5   comp(e1, e2, "<=")
6 })
7 setMethod(">", c("contRV", "numeric"), function (e1, e2) {
8   comp(e1, e2, ">=") #  $P(X > x) = P(X \geq x)$ 
9 })
10 setMethod(">=", c("contRV", "numeric"), function (e1, e2) {
11   comp(e1, e2, ">=")
12 })
13 setMethod("==", c("contRV", "numeric"), function (e1, e2) {
14   comp(e1, e2, "==")
15 })
16 setMethod("%AND%", c("contRV", "contRV"), function (e1, e2) {
17   op(e1, e2, "&") # intersecție
18 })
19 setMethod("%OR%", c("contRV", "contRV"), function (e1, e2) {
20   op(e1, e2, "|") # reuniune
21 })
22 setMethod("|", c("contRV", "contRV"), function (e1, e2) {
23   cond(e1, e2) # condiționare
24 })
```

Pentru operatorii de inegalitate și egalitate, se observă ca aceștia au drept parametri un obiect **contRV** și un număr real. Se apelează funcția *comp*, ce va efectua restrângerea efectivă a suportului variabilei aleatoare pentru calculul probabilităților.

```

1      # Determina suportul pentru expresii de tipul  $X \leq x$ ,  $X \geq x$ 
2      comp <- function(X, x, c)
3      {
4
5          if (X@bidimen)
6              stop("Nu se poate compara o v.a. bidimensională cu un număr real!")
7
8          suportNou <- list()
9          nr <- 1
10
11         # Presupunem ca intervalele sunt in ordine crescatoare dupa
12         # si nu se intersecteaza!
13         if (c == "==")
14         {
15             for (i in X@suport[[1]])
16             {
17                 a <- i[1]
18                 b <- i[2]
19
20                 if (x < a)
21                     break # nu mai are rost sa cautam
22
23                 if (a <= x & b >= x) # daca x se afla in intervalul [a, b]
24                 {
25                     suportNou[[nr]] <- c(x, x) # suportul va fi intervalul [x,
26                     x]
27                     break
28                 }
29             }
30         else if (c == "<=")
31         {
32             # Exemplu: Daca suportul densitatii este format din [0, 2] U
33             # [4, 7] U [9, 11]
34             # Noul suport pentru  $X \leq 5$  va fi [0, 2] U [4, 5]
35
36             for (i in X@suport[[1]])
37             {
38                 a <- i[1]
39                 b <- i[2]
40
41                 if (x < a) # daca x este mai mic decat capatul inferior al
42                     intervalului
43                     break # am terminat de construit suportul
44
45                 if (a <= x & b >= x) # daca x se afla in intervalul [a, b]
46                 {
47                     suportNou[[nr]] <- c(a, x) # adaugam ultimul interval din

```

```

46         noul suport, adica [a, x]
47         break
48     }
49     # altfel, n-am ajuns la un interval care sa-l contina pe x,
50     # deci il adaugam in suportul nou
51     suportNou[[nr]] <- c(a, b)
52     nr <- nr + 1
53 }
54 else # ">="
55 {
56     # Exemplu: Daca suportul densitatii este format din [0, 2] U
57     #           [4, 7] U [9, 11]
58     # Noul suport pentru X >= 5 va fi [5, 7] U [9, 11]
59     # parcurgem intervalele in ordine descrescatoare dupa capatul
60     # inferior
61     for (i in rev(X@suport[[1]]))
62     {
63         a <- i[1]
64         b <- i[2]
65         if (x > b)
66             break
67         if (a <= x & b >= x) # daca x se afla in intervalul [a, b]
68         {
69             suportNou[[nr]] <- c(x, b) # adaugam ultimul interval din
70             # noul suport, adica [x, b]
71             break
72         }
73         # altfel, inca n-am ajuns la un interval care sa-l contina pe
74         # x, deci il adaugam in suportul nou
75         suportNou[[nr]] <- c(a, b)
76         nr <- nr + 1
77     }
78     # inversam ordinea din noul suport, intrucat am parcurs
79     # intervalele din suport in ordine inversa
80     suportNou <- rev(suportNou)
81 }
82
83 # atentie! rezultatul obtinut nu mai este o v.a! se foloseste
84 # doar pt a calcula probabilitati!
85 return (contrRV(densitate = X@densitate, val = X@val, bidimen =
86               X@bidimen, suport = suportNou,
87               ref_va_bidimen = X@ref_va_bidimen))
88 }

```

Pentru operatorii %AND% și %OR% ce implementează intersecția, respectiv reuniunea de variabile aleatoare continue, se apelează funcția *op*, care va avea un comportament diferit în funcție de relația dintre cei doi parametri *X* și *Y* de tip **contRV**. Mai exact, distingem următoarele cazuri:

- *X* și *Y* au aceeași densitate de probabilitate (apare de obicei în urma unor expresii de genul:  $(X < a) \cap (X > b)$ ), caz în care formăm un obiect de tip **contRV** cu densitatea cunoscută și drept suport intersecția dintre suporturile lui *X* și *Y*.
- *X* și *Y* au o referință către aceeași v.a bidimensională, așa că formăm un nou obiect **contRV** cu densitatea comună deja cunoscută și drept suport produsul cartezian între suportul lui *X* și cel al lui *Y*.
- *X* și *Y* au câte o referință către v.a bidimensionale diferite (sau nu au nicio referință), caz în care le considerăm independente și formăm un nou obiect **contRV** cu densitatea comună  $f(x, y) = f_X(x) \cdot f_Y(y)$  și drept suport produsul cartezian între suportul lui *X* și cel al lui *Y*.

Funcția *op* este definită astfel:

```

1      # Reuniune si intersecție de variabile aleatoare
2      op <- function(X, Y, o)
3      {
4          suportNou <- list()
5          nr <- 1
6
7          if (o == "&") # intersecție
8          {
9
10             if (!identical(X@densitate, Y@densitate))
11             {
12
13                 XY <- NULL
14
15                 if (!is.null(X@ref_va_bidimen) & identical(X@ref_va_bidimen,
16                     Y@ref_va_bidimen))
17                 {
18                     # aici se face o copie
19                     XY <- X@ref_va_bidimen
20                 }
21             else
22             {
23                 # consideram ca sunt independente
24                 XY <- contrV(densitate = function(x, y) {X@densitate(x) *
25                     Y@densitate(y)}, bidimen = TRUE)
26             }
27
28             XY@suport[[1]] <- X@suport[[1]]
29             XY@suport[[2]] <- Y@suport[[1]]
30
31             return (XY)
32         }
33     }

```

```

31
32     for (i in X@suport[[1]])
33     {
34         for (j in Y@suport[[1]])
35         {
36             # reuniuni de intersectii ale intervalelor din suport
37
38             A <- interval_intersect(i, j)
39             if (!is.null(A))
40             {
41                 suportNou[[nr]] <- A
42                 nr <- nr + 1
43             }
44         }
45     }
46
47     return (contrRV(densitate = X@densitate, val = X@val, bidimen =
48                     X@bidimen, suport = suportNou,
49                     ref_va_bidimen = X@ref_va_bidimen)) # intoarce contrRV pt a
50                                                         # integra suportul ramas
51 }
52 else # reuniune
53 {
54     return (P(X) + P(Y) - P(X %AND% Y)) # aici intoarce deja
55                                         # probabilitatea calculata
56                                         # problema este ca nu se mai pot aplica alte operatii pe v.a
57 }
58 }

```

O limitare a acestei funcții ar fi că în urma unei operații de reuniune, nu se întoarce un obiect **contrRV** pentru a fi folosit în alte expresii, ci direct rezultatul probabilității, adică  $P(X \in A) + P(Y \in B) - P(X \in A, Y \in B)$ .

Ultimul operator implementat este cel de condiționare:

```

setMethod("|", c("contrRV", "contrRV"), function (e1, e2) {
  cond(e1, e2)
})

```

pentru care se apelează funcția *cond*, care are la rândul ei un comportament diferit în funcție de relația dintre cele 2 obiecte  $X$  și  $Y$  de tipul **contrRV**. Dacă:

- $X$  și  $Y$  au densitățile marginale dintr-o v.a bidimensională  $(X, Y)$ , atunci calculează direct probabilitatea folosind formula:  $P(X \in A | Y = y) = \int_A f_{X|Y}(x|y) dx$ , unde densitatea condiționată o obținem din funcția *dens\_condit\_x\_de\_y*, descrisă în exercițiul 11.
- Altfel, avem ori  $X$  și  $Y$  independente, ori au aceeași densitate. În ambele cazuri, putem folosi formula:  $P(X \in A | Y \in B) = \frac{P(X \in A, Y \in B)}{P(Y \in B)}$ . Chiar și pentru  $X, Y$  independente, rezultatul ar fi cel așteptat, adică  $P(X \in A)$ .

Funcția cond arată în felul următor:

```

1  # Calculeaza probabilitatea conditionata
2  # X si Y pot fi expresii de tipul Z <= x, Z %AND% W etc.
3  cond <- function (X, Y)
4  {
5      if (!is.null(X@ref_va_bidimen) & identical(X@ref_va_bidimen,
6          Y@ref_va_bidimen))
7      {
8          # aici probabil ar trebui verificat daca X si Y referintiaza
9          # aceeasi v.a bidimensionala
10         XY <- X@ref_va_bidimen
11         fx_cond_y <- dens_condit_x_de_y(XY)
12
13         if (length(Y@suport[[1]]) == 0) # inseamna ca nu a fost gasit y
14             in suportul lui Y
15         {
16             return (0)
17         }
18
19         if (length(Y@suport[[1]]) != 1 || Y@suport[[1]][[1]][[1]] !=
20             Y@suport[[1]][[1]][[2]])
21         {
22             stop("Nu pot calcula asa ceva! Suportul lui Y trebuie sa fie
23                 un singur punct!!")
24         }
25
26         yfixat <- Y@suport[[1]][[1]][[1]]
27
28         sum <- 0
29         for (i in X@suport[[1]]) {
30             tryCatch(sum <- sum + integrate(fx_cond_y, i[1], i[2], y =
31                 yfixat, abs.tol = 1.0e-13)$value,
32                 error= function(err)
33                 {
34                     stop("Integrala a esuat.")
35                 })
36         }
37         return (sum)
38     }
39     else
40     {
41         return (integrala(X %AND% Y) / integrala(Y)) # P(X intersect Y)
42             / P(Y)
43     }
44 }

```

Comparație între notații:

contRV	Notăție matematică
$P(X \leq x)$	$P(X \leq x)$
$P((X \leq a) \%AND\% (Y \geq b))$	$P(X \leq a, Y \geq b)$
$P((X \leq a) \%OR\% (X \geq b))$	$P(X \leq a \cup X \geq b)$
$P((X \geq a) \%AND\% (X \leq b) \mid X < c)$	$P(a \leq X \leq b \mid X < c)$
$P(X > x \mid Y == y)$	$P(X > x \mid Y = y)$

Exemple de cod:

```
> XY <- contrRV(densitate = function (x, y) (6/7) * (x+y)^2,
                 bidimen = TRUE,
                 suport = list(list(c(0, 1)), list(c(0, 1))))
> X <- marginalaX(XY)
> Y <- marginalaY(XY)
> P((X <= 0.5) %AND% (X >= 0.2) | Y == 0.2)
[1] 0.1622093
> P((X <= 0.7) %AND% (Y >= 0.5))
[1] 0.3815

> func <- function(x)
+ {
+   if (x < -1)
+     0
+   else if (x < 0)
+     1 + x
+   else if (x < 1)
+     1 - x
+   else
+     0
+ }
> Z <- contrRV(densitate = Vectorize(func), bidimen = FALSE, suport
               = list(c(-1, 1)))
> P(Z <= 0)
[1] 0.5
> P(((Z <= 0.5) %AND% (Z >= -0.7)) %OR% (Z <= 1))
[1] 1
```