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, întrucât ar fi inutil să putem calcula doar probabilități de tipul $P(X)$, intrucât rezultatul ar fi întodeauna 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 supraîncărcat următorii operatori:

```
1   setMethod("<", c("contRV", "numeric"), function (e1, e2) {
2     comp(e1, e2, "<=") # P(X < x) = P(X <= 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 >= 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, "&") # intersectie
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) # conditionare
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 <= x, X >= x
2   comp <- function(X, x, c)
3   {
4
5     if (X@bidimen)
6       stop("Nu se poate compara o v.a bidimensionala cu un numar real
          !")
7
8     suportNou <- list()
9     nr <- 1
10
11    # Presupunem ca intervalele sunt in ordine crescatoare dupa
        capatul inferior
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,
              x]
26          break
27        }
28      }
29    }
30    else if (c == "<=")
31    {
32      # Exemplu: Daca suportul densitatii este format din [0, 2] U
          [4, 7] U [9, 11]
33      # Noul suport pentru X <= 5 va fi [0, 2] U [4, 5]
34
35      for (i in X@suport[[1]])
36      {
37        a <- i[1]
38        b <- i[2]
39
40        if (x < a)  # daca x este mai mic decat capatul inferior al
            intervalului
41        break # am terminat de construit suportul
42
43        if (a <= x & b >= x) # daca x se afla in intervalul [a, b]
44        {
45          suportNou[[nr]] <- c(a, x) # adaugam ultimul interval din
```

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

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 nou obiect **contRV** cu densitatea comună deja cunoscută și drept suport produsul cartezian intre 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 $f(x, y) = f_X(x) * f_Y(y)$ și drept suport produsul cartezian intre suportul lui X și cel al lui Y

Funcția *op* este definită astfel:

```
1    # Reuniune si intersectie de variabile aleatoare
2    op <- function(X, Y, o)
3    {
4
5      suportNou <- list()
6      nr <- 1
7
8      if (o == "&") # intersectie
9      {
10
11       if (!identical(X@densitate, Y@densitate))
12       {
13
14         XY <- NULL
15
16         if (!is.null(X@ref_va_bidimen) & identical(X@ref_va_bidimen,
               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) *
                 Y@densitate(y)}, bidimen = TRUE)
25         }
26
27         XY@suport[[1]] <- X@suport[[1]]
28         XY@suport[[2]] <- Y@suport[[1]]
29
30         return (XY)
31       }
32
```

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