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 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     suportNou <- list()
5     nr <- 1
6
7     if (o == "&") # intersectie
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,
              Y@ref_va_bidimen))
16        {
17          # aici se face o copie
18          XY <- X@ref_va_bidimen
19        }
20        else
21        {
22          # consideram ca sunt independente
23          XY <- contRV(densitate = function(x, y) {X@densitate(x) *
                Y@densitate(y)}, bidimen = TRUE)
24        }
25
26        XY@suport[[1]] <- X@suport[[1]]
27        XY@suport[[2]] <- Y@suport[[1]]
28
29        return (XY)
30      }
```

```
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 (contRV(densitate = X@densitate, val = X@val, bidimen =
                X@bidimen, suport = suportNou,
48          ref_va_bidimen = X@ref_va_bidimen)) # intoarce contRV pt a
                integra suportul ramas
49        }
50        else # reuniune
51        {
52          return (P(X) + P(Y) - P(X %AND% Y)) # aici intoarce deja
                probabilitatea calculata
53          # problema este ca nu se mai pot aplica alte operatii pe v.a
54        }
55      }
```

O limitare a acestei funcții ar fi că în urma unei operații de reuniune, nu se întoarce un obiect **contRV** 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("contRV", "contRV"), 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 **contRV**. Dacă:

- X și Y au densitățile marginale dintr-o v.a bidimensională (X, Y), atunci calculează direct probabilitatea folosind formula: $P(X \in A \mid Y = y) = \int_A f_{X|Y}(x|y)\,dx$, 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: $P(X \in A \mid Y \in B) = \frac{P(X \in A, Y \in B)}{P(Y \in B)}$. Chiar și pentru X, Y independente, rezultatul ar fi cel așteptat, adică $P(X \in A)$.

Funcția cond arată în felul următor:

```
    # Calculeaza probabilitatea conditionata
    # X si Y pot fi expresii de tipul Z <= x, Z %AND% W etc.
    cond <- function (X, Y)
    {
      if (!is.null(X@ref_va_bidimen) & identical(X@ref_va_bidimen,
          Y@ref_va_bidimen))
      {
        # aici probabil ar trebui verificat daca X si Y referintiaza
            aceeasi v.a bidimensionala
        XY <- X@ref_va_bidimen
        fx_cond_y <- dens_condit_x_de_y(XY)

        if (length(Y@suport[[1]]) == 0) # inseamna ca nu a fost gasit y
             in suportul lui Y
        {
          return (0)
        }

        if (length(Y@suport[[1]]) != 1  || Y@suport[[1]][[1]][[1]] !=
            Y@suport[[1]][[1]][[2]])
        {
          stop("Nu pot calcula asa ceva! Suportul lui Y trebuie sa fie
              un singur punct!!")
        }


        yfixat <- Y@suport[[1]][[1]][[1]]

        sum <- 0
        for (i in X@suport[[1]]) {
          tryCatch(sum <- sum + integrate(fx_cond_y, i[1], i[2], y =
              yfixat, abs.tol = 1.0e-13)$value,
          error= function(err)
          {
            stop("Integrala a esuat.")
          })
        }
        return (sum)

      }
      else
      {
        return (integrala(X %AND% Y) / integrala(Y)) # P(X intersect Y)
            / P(Y)
      }
    }
```

Comparație între notații:

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

Exemple de cod:

```
> XY <- contRV(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 <- contRV(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
```