Documentație Proiect 1

Probabilități și Statistică

January, 2021

Contents

1 Echipă **2**

2 Introducere **2**

3 Cerințe **2**
3.1 Cerința 1 . . . 2
3.2 Cerința 2 . . . 2
3.3 Cerința 3 . . . 4
3.4 Cerința 4 . . . 4
3.5 Cerința 5 . . . 4
3.6 Cerința 6 . . . 5
3.7 Cerința 7 . . . 5
3.8 Cerința 8 . . . 8
3.9 Cerința 10 . . . 8
3.10 Cerința 11 . . . 9
3.11 Cerința 12 . . . 11

4 Concluzie **13**

1 Echipă

Membrii: Ștefan Radu, Comorașu Ana-Maria, Buhai Darius și Ciaușu Nicoleta.
Grupă: 234

2 Introducere

Construirea unui pachet R pentru lucru cu variabile aleatoare continue.

Folosind documentul suport și orice alte surse de documentare considerați potrivite construiți un pachet R care să permită lucru cu variabile aleatoare continue. Pentru a primi punctaj maxim, pachetul trebuie să implementeze cel puțin 8 din următoarele cerințe (oricare 8):

3 Cerințe

3.1 Cerința 1

Enunț

Fiind dată o funcție f, introdusă de utilizator, să se determine constanta de normalizare. Pentru cazul în care o asemenea constantă nu există, afișarea unui mesaj corespunzător către utilizator.

Rezolvare

Pentru calcularea constantei de normalizare, vom folosi regula: $k \int_{-\infty}^{\infty} f(x)dx = 1$. Astfel, funcția *compute_normalizing_constant* primește 3 parametri, printre care ultimii doi pot avea valori implicite. Se va încerca să se calculeze integrala funcției pe domeniul său - sau pe intervalul $(-\infty, \infty)$. Dacă aceasta a fost calculată cu succes, se va calcula constanta de normalizare drept i^{-1}, pentru i integrala rezultată. Dacă încercarea calculării constantei de normalizare eșuează, se va prinde eroarea și se va afișa mesajul corespunzător, iar valoarea returnată va fi 0.

Cod

```
awaiting code here
```

Teste

```
f <- function(x){
  return (exp((-x^2)/2))
}
compute_normalizing_constant(f, 0, 1)
# [1] 1.168737
compute_normalizing_constant(f)
# [1] 0.3989423

g <- function(x){
  x^2
}
# Caught an error!
# <simpleError in integrate(f, lower = lowerBound, upper = upperBound): the integral is
   probably divergent>
# [1] 0
```

3.2 Cerința 2

Enunț

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

Rezolvare

Pentru a verifica dacă funcția f este densitate de probabiliate, am testat următoarele 2 proprietăți:

1. $f(x) \geq 0$ (f este nenegativă)
2. $\int_{-\infty}^{\infty} f(x)\,dx = 1$

În primă parte, pentru a verifica dacă funcția dată este pozitivă, i-am calculat toate valorile din intervalul [-10000000, 10000000] și am verificat dacă sunt pozitive.

Pentru a evita erorile generate în urma integrării funcțiilor divergente, am creat o funcție safe_integrate ce returnează FALSE în cazul în care funcția dată nu poate fi integrată.

Cod

```

  # Integrate a function and return False if any errors are generated
  safe_integrate <- function(f, d = c(-Inf, Inf)){
    tryCatch(integrate(Vectorize(f), d[1], d[2]),
             error = function(e){
               print(e)
               FALSE
             })
  }

  # Check if a function is positive in a given interval
  is_positive <- function(f, d = c(-10000000, 10000000), step=10000000){
    # Set maximum lower and upper bounds
    if(d[2] == Inf)
      d[2] = 10000000;
    if(d[1] == -Inf)
      d[1] = -10000000;
    # Calculate values in that intervals
    vals <- seq(d[1], d[2], len=step)
    # Check if all values are positive
    all(f(vals)>=0)
  }

  check_pdf <- function(f, d = c(-Inf, Inf)){
    # Check if the function is positive
    if(!is_positive(f, d)){
      print("Function is negative")
      return(FALSE)
    }
    # Safe integrate, handle errors
    i = safe_integrate(f, d)
    # Cannot integrate
    if(typeof(i)=='logical' && i==FALSE)
      return(FALSE)
    v = i $ value
    round(v) == 1
  }
```

Teste

```
f1 <- function(x){
  if (x > 0 && x < 2){
    3/8 * (4*x-2*x^2)
  }else{
    0
  }
}

check_pdf(f1)
# TRUE

```

```r
f2 <- function(x){
  x * (4*x-2*x^2)
}

check_pdf(f2)
# FALSE
```

Probleme întâlnite

Verificarea funcțiilor pozitive nu este 100% precisă atunci când domeniul nu este definit, deoarece pe un interval infinit, valorile funcției sunt limitate la valori între -10000000 și 10000000.

3.3 Cerința 3

Enunț

Rezolvare

3.4 Cerința 4

Enunț

Rezolvare

3.5 Cerința 5

Enunț

Calculul mediei, dispersiei și a momentelor inițiale și centrate până la ordinul 4 (dacă există). Atunci când unul dintre momente nu există, se va afișa un mesaj corespunzător către utilizator.

Rezolvare

În rezolvarea acestui exercițiu am folosit următoarele formule, preluate de pe Wikipedia:

Momentul Central: $\mu_n = \int_{-\infty}^{\infty}(x-\mu)^n * f(x)\,dx$
Momentul Inițial: $\mu_n = \int_{-\infty}^{\infty} x^n * f(x)\,dx$

Iar pentru a determina dacă un moment dat nu există, am folosit funcția safe_integrate ce returnează FALSE în cazul în care pdf-ul dat nu poate fi integrat, afișând pe ecran un mesaj corespunzător.

Cod

```r

  # Calculating central moment of order o
  central_moment <- function(f, o){
    m = medium(f)
    f_new <- function(x){
      ((x - m) ^ o) * f(x)
    }
    res = safe_integrate(f)
    if(typeof(res)=="logical" && res==FALSE){
      print("Momentul nu exist !")
      return(0)
    }
    res $ value
  }

  # Calculating initial moment of order o
  initial_moment <- function(f, o){
    f_new <- function(x){
      x ^ o * f(x)
    }
    res = safe_integrate(f)
```

```
    if(typeof(res)=="logical" && res==FALSE){
      print("Momentul nu exist !")
      return(0)
    }
    res $ value
  }
```

Teste

```
f = function (x) {
  if(x<0 || x>1)
    return(0)
  x ^ 3
}

initial_moment(f, 2)
central_moment(f, 2)
```

3.6 Cerința 6

Enunț

Calculul mediei și dispersiei unei variabile aleatoare g(X), unde X are o repartiție continuă cunoscută iar g este o funcție continuă precizată de utilizator.

Cod

```
  # am nevoie de functia g, pdf-ul lui X si domeniul de definitie
  median_and_dispersion <- function(g, fx, domeniu_valori) {
    # Y = g(X) poate fi vazut ca o v.a.c. noua cu pdf-ul g(f(x)), ambele functii fiind
  continue le compun

    # folosesc formula pt media functiilor de x
    # folosesc integral(din pachetul pracma) deoarece integer imi da eroare cand incerc
   sa integrez la +-inf
    e_y <- integral(Vectorize(function(x){g(x) *fx(x)}),domeniu_valori[1],domeniu_
  valori[2])

    # pt dispersie, mai folosesc o data formula mediei functiilor de x
    # pentru a afla X^2
    e_y2 <- integral(Vectorize(function(x){x^2 * g(x) *fx(x)}),domeniu_valori[1],
  domeniu_valori[2])
    dispersie <- e_y2 - e_y^2

    print(paste("Media: ",e_y))
    print(paste("Dispersia:", dispersie))
  }
```

Teste

```
  # exemplu
  f1 <- function(x)(x^2)
  f2 <- function(x) (1 * exp(1)^(-1 * x))

  median_and_dispersion(f1,f2, c(0,Inf)) # 2, 20
```

3.7 Cerința 7

Enunț

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

Rezolvare

```
####
# Cerinta: Crearea unei func ii P care permite calculul diferitelor tipuri de
# probabilit  i asociate unei variabile aleatoare continue(similar func iei P din pachetul discreteRV)
#
# Header functie: myP(f, p)
#    - unde f este o functie densitate de probabilitate (pdf)
#    - iar p este un string ce reprezinta probabilitatea (conditionata sau independenta).
#
# Obligatoriu, var se va afla in stanga operatorului
####

myP <- function(f, p) {
  operatii_posibile=c("<=",">=","=","<",">")

 ########### Functii ajutatoare ###########
  # transforma string-ul dat in ceva ce pot utiliza
  parseaza_expresie <- function(expresie) {

    # scot whitespace
    expresie <- gsub(" ", "", expresie)

    # iau fiecare operatie posibila pe rand si incerc sa dau split dupa ea, daca pot, mi-am gasit operatorul si returnez parametri
    for(op in operatii_posibile) {

      # am dat split corect => in stanga am variabila, in dreapta am bound-ul
      split <- unlist(strsplit(expresie, op, fixed = TRUE))
      splitSize <- length(split)

      if (splitSize == 2) {
        # returnez (v.a.c, operatie, bound)
        return (c(split[1],op,split[2]))
      }
    }
    #daca am ajuns aici => nu am operator, deci eroare
    return(c(-1))


  }

# calculez cdf, adica integrala -inf, bound, adica P(X <= bound)
cdf <- function(bound) { return (integrate(f, -Inf, bound) $ value)}

# transform din string in double
compute_bound <-function(bound) {
  rez <- switch(bound,
                "-Inf" = -Inf,
                "+Inf" = +Inf,
                as.double(bound))
  return (rez)
}

## Calculeaza probabilitatea ##
evalueaza <- function(operator, bound) {

  # parsez bound-ul, transform in double sau in +-inf
  bound = compute_bound(bound)
  integrala <- cdf(bound)

  #pentru >,>= din toata aria(1) scad cdf-ul si obtin restul
  ans <- switch(
    operator,
    "=" = 0,
    "<=" = integrala,
    "<" = integrala,
    ">=" = 1 - integrala,
```

```
      ">" = 1 - integrala)

    return(ans)

    }

# daca am o singura expresie e prob independenta
prob_independenta <- function(expresie) {

  # scot parametri din expresie
  parametri <- parseaza_expresie(expresie)

  if(length(parametri) != 3)
    return("Eroare la parsarea probabilitatii")

  # aici presupun ca expresiile mele sunt mereu de forma x operator bound
  # ar trebui o verificare, poate, a ordinii

  operator <- parametri[2]
  bound <- parametri[3]

  print(evalueaza(operator, bound))

}

#daca am 2 expresii e prob conditionata
prob_conditionata <- function(expresie1, expresie2) {

  # scot parametri in variabile ca sa fie readable ce fac mai jos
  parametri1 <- parseaza_expresie(expresie1)
  parametri2 <- parseaza_expresie(expresie2)
  op1 <- parametri1[2]
  op2 <- parametri2[2]
  bound1 <- parametri1[3]
  bound2 <- parametri2[3]

  # am formula P(a depinde de b) = P(a intersectat b)/P(b)
  # calculez cdf pentru fiecare functie si iau pe cazuri
  ans1 <- evalueaza(op1, bound1)
  ans2 <- evalueaza(op2, bound2)

  # daca vreuna e 0, intersectia va da 0
  if(ans1 == 0)
      return(0);
  if(ans2 == 0)
      return ("Cannot divide by zero")

  ## cazuri

  ## caz in care conditionarea face probabilitatea sa fie imposibila
  # p(x < 3 | x > 5) = 0
  if (op1 %in% c("<=","<") && op2 %in% c(">=", ">") && bound1 >=bound2)
    return (0);

  # p(x > 5 | x < 3) = 0
  if (op1 %in% c(">=",">") && op2 %in% c("<=", "<") && bound1 >=bound2)
    return (0);

 ## caz in care am acelasi fel de operator, facand intersectia defapt doar aleg
intervalul cel mai restrans
  # p(x> 3 | x>7) => iau (-inf,3)
  if(op1 %in% c(">=",">") && op2 %in% c(">=",">"))
    if(bound1 > bound2)
      return (ans1/ans2)
    else return (1);
 # p( x<3 | x<7) => iau (-inf,3)
  if(op1 %in% c("<=","<") && op2 %in% c("<=","<"))
    if(bound1 < bound2)
    return (ans1/ans2)
    else return (1)
```

```
135
136       ## daca nu e niciunul de mai sus, e intersectie de forma x > 5 | x < 7 si fac
      diferenta
137       ## din cdf mai mare scad cdf mai mic si mi da bucata de dintre => intersectia
138       return ((cdf(compute_bound(bound2))-cdf(compute_bound(bound1)))/ans2)
139
140     }
141
142     ####### Body functie principala ##########
143
144       # parsez parametri
145       parti = unlist(strsplit(p, "|", fixed = TRUE))
146       len = paste(length(parti))
147       switch(len,
148              "0" = return("Eroare"),
149              "1" = return(prob_independenta(p)),
150              "2" = return(prob_conditionata(parti[1],parti[2])),
151              )
152       return ("eroare");
153
154     }
```

Teste

```
1      g <- function (x) {
2           fun <- 0.1*(3*(x^2) + 1)
3           fun[x<0] = 0
4           fun[x>2]=0
5           return ( fun )
6      }
7
8     myP(g,"x>1|x<1.5") #
```

3.8 Cerința 8

Enunț

Rezolvare

3.9 Cerința 10

Enunț

Calculul covarianței și coeficientului de corelație pentru două variabile aleatoare continue.

Rezolvare

În rezolvarea problemei am folosit următoarele formule de calcul ale covarianței și a coeficientului de corelație preluate din Cursurile 8 și 6.
De asemenea, pentru a determina funcțiile pdf ale variabilelor x și y din densitatea comună, am integrat pdf_comun în funcție de x și y pe domeniile aferente.

$\mu x = E(x) = \int_a^b x * f(x)\,dx$
$cov(x,y) = (\int_c^d \int_a^b x * y * f(x,y)\,dx\,dy) - \mu x * \mu y$
$cor(x,y) = \frac{cov(x,y)}{\sigma x * \sigma y}$

Cod

```
1
2     # Calculates medium
3     medium <- function(f, d = c(-Inf, Inf)){
4       integrate(function(x){ x * f(x)}, d[1], d[2]) $ value
5     }
```

```r

  # Extracts x out of common density of x and y
  extract_x_marginal <- function(f, dx){
    Vectorize(function(y){
      integrate(function(x){f(x, y)}, dx[1], dx[2]) $ value
    })
  }

  # Extracts y out of common density of x and y
  extract_y_marginal <- function(f, dy){
    Vectorize(function(x){
      integrate(function(y){f(x, y)}, dy[1], dy[2]) $ value
    })
  }

  # Calculates Covariance and Correlation coefficient
  covariance_and_correlation <- function(pdf, dx, dy){

    fx = extract_x_marginal(pdf, dx)
    fy = extract_y_marginal(pdf, dy)

    mx = medium(fx, dx)
    my = medium(fy, dy)

    cov = double_integrate(function(x,y){x*y*pdf(x,y)}, dx, dy) - (mx*my)

    cor = cov / (mx*my)

    list(cov = cov, cor = cor)
  }
```

Teste

```r
f <- function (x, y) {
   return (3/2 * (x^2+y^2))
}
covariance_and_correlation(f, c(0,1), c(0,3))

# $cov
# [1] -78.04688
# $cor
# [1] -0.8222222
```

Probleme întâlnite

În rezolvarea problemei am găsit dificil determinarea pdf_x și pdf_y din pdf_comun, lucru pe care l-am rezolvat prin funcțiile extract_x_marginal și extract_y_marginal.

3.10 Cerința 11

Enunț

Pornind de la densitatea comună a două variabile aleatoare continue, construirea densităților marginale și a densităților condiționate.

Rezolvare

Cod

```r
  # conditia 1 pentru o functie de densitate corecta
  # f trebuie sa aiba valori pozitive pentru
  # toate valorile din domeniu
  # cum in domeniu sunt o infinitate de puncte
  # am ales sa verific 'pe sarite' o parte dintre ele
  check_positive_2d <- function(f, ab, cd, step) {
```

```
    for (x in seq(ab[1], ab[2], step)) {
      for (y in seq(cd[1], cd[2], step)) {
        if (f(x, y) < 0) return(FALSE)
      }
    }

    return(TRUE)
  }

  # conditia 2 pentru o functie de densitate corecta
  # f terbuie sa aiba probabilitatea totala egala cu 1
  # pentru a calcula integrala dubla am folosit functia
  # 'integral2' din pachetul 'pracma'
  check_integral_is_1 <- function(f, ab, cd) {
    integral <- integral2(f, ab[1], ab[2], cd[1], cd[2])$Q
    return(abs(integral - 1) < 0.0000000001)
  }

  # intoarce o noua functie g(y) definita
  # ca f(x, y), unde x este o valoare fixata
  f_set_x <- function(x, f) function(y) f(x, y)

  # intoarce o noua functie g(x) definita
  # ca f(x, y), unde y este o valoare fixata
  f_set_y <- function(y, f) function(x) f(x, y)

  # calculeaza densitatea marginala in X
  # il setam pe x si integram peste y
  x_marginal_pdf <- function(f, x, c, d) {
    return (integrate(f_set_x(x, f), c, d)$value)
  }

  # calculeaza densitatea marginala in Y
  # il setam pe y si integram peste x
  y_marginal_pdf <- function(f, y, a, b) {
    return (integrate(f_set_y(y, f), a, b)$value)
  }

  # calculeaza densitatea in X conditionata de Y = y
  h_x_conditioned_by_y <- function(y, x, f, c, d) {
    f(x, y) / x_marginal_pdf(f, x, c, d)
  }

  # calculeaza densitatea in Y conditionata de X = x
  h_y_conditioned_by_x <- function(x, y, f, a, b) {
    f(x, y) / y_marginal_pdf(f, y, a, b)
  }

  marginal_conditional_densities <- function(f, x, y, ab, cd) {

    tryCatch({
      if (!check_positive_2d(f, ab, cd, 0.01)) {
        print("functia nu este pozitiva pentru valorile din domeniu")
        return ()
      }

      if (!check_integral_is_1(f, ab, cd)) {
        print("probabilitatea totala este diferita de 1")
        return ()
      }

      print(paste("testez cu x=", x, " y=", y, " pe [", ab[1], ",",
                  ab[2], "]x[", cd[1], ",", cd[2], "]", sep=""))

      # densitatea marginala pentru X cu x fixat
      print(paste("pdf marginal cu x fixat:", x_marginal_pdf(f, x, cd[1], cd[2])))

      # densitatea marginala pentru Y cu y fixat
      print(paste("pdf marginal cu y fixat:", y_marginal_pdf(f, y, ab[1], ab[2])))

```

```
        # densitatea conditionata in x cand Y = y
        print(paste("pdf conditionat de y=", y, ": ", h_x_conditioned_by_y(y, x, f, cd
    [1], cd[2]), sep=""))

        # densitatea conditionata in y cand X = x
        print(paste("pdf conditionat de x=", x, ": ", h_y_conditioned_by_x(x, y, f, ab
    [1], ab[2]), sep=""))
      }, error = function(e) {
        print(paste("A aparut o eroare", e, sep = " "))
      })
    }
```

Teste

```
    test_1 <- function() {
      f <- function(x, y) (8 / 3) * x ^ 3 * y
      marginal_conditional_densities(f, 0.5, 1.5, c(0, 1), c(1, 2))
    }

    test_2 <- function() {
      f <- function(x, y) (3 / 2) * (x ^ 2 + y ^ 2)
      marginal_conditional_densities(f, 0, 0.5, c(0, 1), c(0, 1))
    }

    test_1()
    test_2()
```

3.11 Cerința 12

Enunț

Construirea sumei și diferenței a două variabile aleatoare continue independente (folosiți formula de convoluție)

Documentare

Întâi a trebuit să mă documentez despre formula de convoluție, așa că m-am orientat spre urmatoarele surse:

1. Pagina Convolutions de pe StatLect, unde am găsit formula pentru convoluția a două pdf-uri ale unor variabile aleatoare continue independente.
2. O parte dintr-un curs de pe MIT OpenCourseWare despre suma variabilelor aleatoare continue independente unde a fost prezentată și o demonstrație pe care voi încerca să o explic mai jos.
3. Acest video de pe YouTube care vorbește despre formula pentru diferență.

Pentru rezolvarea cerinței, am folosit formulele următoare care nu sunt prezente in curs:

$f_Z(z) = \int_{-\infty}^{\infty} f_X(z-y) f_Y(y)\, dy$ pentru $Z = X + Y$
$f_Z(z) = \int_{-\infty}^{\infty} f_X(y-z) f_Y(y)\, dy$ pentru $Z = X - Y$.

Cod

```
    # suma a doua variabile aleatoare continue independente folosind formula de
    convolutie

    convolution_sum <- function(fx,fy) {
      res <-function(z) {
        integrate(function(y) {
          fx(z-y) * fy(y)
          },-Inf,Inf)
      }
    }
```

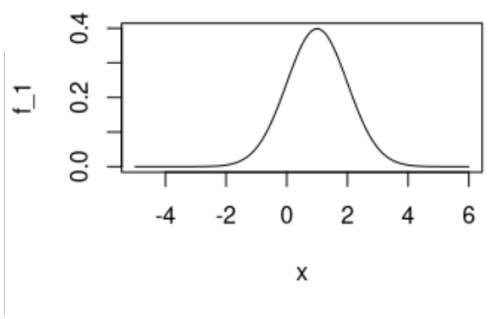
```r

  # diferenta a doua variabile aleatoare continue independente folosind formula de convolutie
  convolution_diff <- function(fx,fy) {
    res <-function(z) {
      integrate(function(y) {
        g(y-z)*f(y)
      },-Inf,Inf) $ value

    }
  }
```

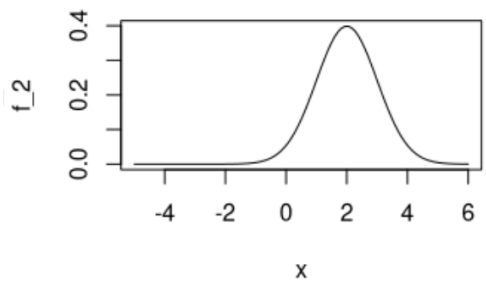
Teste

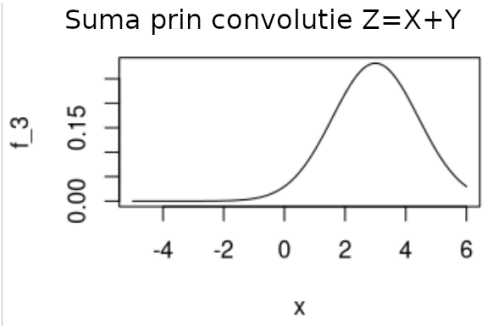
```r
  # doua distributii normale, suma si diferenta facuta cu formula de convolutie
  f_1 <- function(x)(dnorm(x,mean=1))
  f_2 <- function(x) (dnorm(x,mean=2))
  f_3 <- Vectorize(convolution_sum(f_1, f_2))
  f_4 <- Vectorize(convolution_diff(f_1,f_2))

  plot(f_1,from=-10,to=10,type="l")
  plot(f_2,from=-10,to=10,type="l")
  plot(f_3(t),from=-10,to=10,type="l")
  plot(f_4(t),from=-10,to=10,type="l")
```

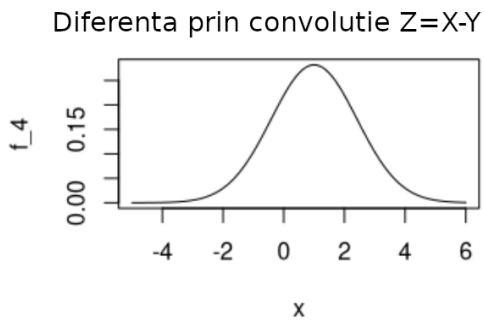
Output

Pentru două v.a.c cu distributii normale:









Probleme întălnite

1. Nu am reusit sa integrez spre $+-\infty$ cu functia standard *integrate* fara sa obtin erori, asa ca am folosit functia *integral* din pachetul **pracma**.

2. A fost nevoie sa vectorizez pdf-urile rezultate prin convolutie ca sa pot sa le plotez.

4 Concluzie

Prin urmare putem spune că prin rezolvarea acestui proiect, am învățat să aplicăm diferite formule de variabile aleatoare continue în limbajul R, cât și să construim un pachet complet R, cu documentație și manual.