Facultatea de Matematică și Informatică
Universitatea din București
Probabilități și Statistică 2020-2021
Proiect Laborator - Anul II, Informatică
Grupa 243

Construirea unui pachet R pentru lucru cu variabile aleatoare continue

Băeșu Rareș Gabriel
Costrun Larisa-Bianca
Poinărița Andreea Diana
Titrat Cristina-Georgiana

04 Februarie 2020

Cuprins

1 Introducere

Lucrarea de față își propune să aducă în prim plan modul în care a fost implementat proiectul *Construirea unui pachet R pentru lucru cu variabile aleatoare continue* din cadrul laboratorului pentru cursul Probabilități și Statistică.

Am ales să construim un pachet R ca proiect final, deoarece am dorit să explorăm noul limbaj învățat și în același timp să contribuim în comunitatea open-source. De asemenea lucrul cu pachetul discreteRV ne-a făcut vrem să implementăm ceva asemănător.

Pachetul poate fi folosit în lucrul cu variabilele aleatoare continue, având implementate funcționalități precum: determinarea unei constante de normalizare, reprezentarea grafică a densității, calculul mediei, dispersiei și a momentelor inițiale și centrate etc. Dorim să aflați mai multe în paginile ce urmează. :D

2 Determinarea constantei de normalizare (1)

Cerință: Fiind dată o funcție f , introdusă de utilizator, determinarea unei *constante de normalizare* k. În cazul în care o asemenea constantă nu există, afișarea unui mesaj corespunzător către utilizator.

Noțiuni teoretice: În teoria probabilității, o constantă de normalizare este o constantă prin care o funcție non-negativă trebuie să fie înmulțită astfel încât aria de sub graficul său să fie 1.

Implementare: Am definit o funcție numită constantaNorm care primește ca parametru o funcție și calculează constanta k

```r
constantaNorm <- function(f) {
  #Primeste functia ca parametru
  x <- seq(-1000, 1000, 0.1)
  if(!all(f(x) >0)) {
    #o verificare pur orientativa a pozitivitatii
    #deoarece o verificare pe R nu este posibila
    #lasand adevarata verificare in seama utilizatorului

    return(NULL)
  }

  integrala <- integrate(f,-Inf, Inf)
  #presupunand ca e integrabila, o integrez pe R

  rez <- 1
  #rez reprezinta valoarea constantei de norm
  if(integrala$value != 1 && integrala$value != 0) {
    rez<-1/integrala$value
    #1/integrala pentru ca reprezinta acea valoare
    #cu care trb sa inmultim integrala
    #sa obtinem 1
  }
  else {
    rez <- NULL
    #null pentru ca nu exista
  }
  rez
}

test <- function(x){1/(x*x + 1)}
print(constantaNorm(test))
```

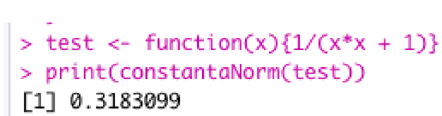


Figura 1: Exemplu rulare funcție constantaNorm

3 Densitate de probabilitate (2)

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

Noțiuni teoretice: Pentru ca funcția să fie densitate de probabilitate/repartiție, trebuie ca aceasta să îndeplinească următoarele proprietăți:

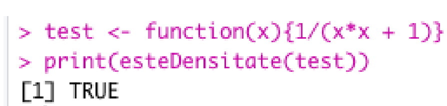
$$\int_{-\infty}^{\infty} f(x)\, dx = 1$$

$$f(x) \geq 0\, \forall x \in\ R$$

Implementare: Am definit o funcție numită sugestiv *esteDensitate* care primește ca parametru o funcție f. Inițial considerăm funcția primită ca fiind de densitate de probabilitate, avand variablia *este* cu valoarea $TRUE$. Ne folosim de funcția predefinită *all* pentru a verifica dacă valorile funcției sunt mai mari ca 0 si integram functia pentru a testa daca rezultatul obtinut este egal cu 1; în cazul in care cel putin una dintre proprietatile testate nu este indeplinita, atunci functia nu este densitate și returnăm $FALSE$

```r
esteDensitate <- function(f) {
  x <- seq(-1000,1000,0.1)
  #evident, o verificare pe intreg R-ul este imposibila
  #asa ca lasam la mana utilizatorului sa determine daca f(x) > 0 pentru orice x din R
  #aceasta verificare fiind pur orientativa
  este <- TRUE
  if(!all(f(x) > 0)) {
    return(FALSE)
  }
  integrala <- integrate(f,-Inf, Inf)
  if(integrala$value == 1) {
    este <- TRUE
  }
  este
}

test <- function(x){1/(x*x + 1)}
print(esteDensitate(test))
```



```
> test <- function(x){1/(x*x + 1)}
> print(esteDensitate(test))
[1] TRUE
```

Figura 2: Exemplu rulare funcție esteDensitate

4 Variabilă aleatoare continuă (3)

Cerință: Crearea unui obiect de tip variabilă aleatoare continuă pornind de la o densitate de probabilitate introdusă de utilizator. Funcția trebuie să aibă opțiunea pentru variabile aleatoare unidimensionale și respectiv bidimensionale.

Implementare: Pentru a crea obiectul am folosit o clasa S4, care are ca variabile doi vectori, unul în care se rețin funcțiile și respectiv un vector pentru intervalele pe care sunt definite acestea. Am considerat ca valoarea în afara acestor intervale este automat 0. Exista opțiunea ca datele sa fie cerute pe rand de la utilizator prin intermediul consolei (creareVariabilaTastatura) sau printr-o funcție care primeste ca pamaetru un vector cu functiile si unul cu intervalele corespunzatoare (creareVariabilaParam). Pentru mai mult detalii de implementare, se pot urmari comentariile care insotesc codul.

```
#Construiesc o clasa variabilaAleatoare cu func ia de densitate si intervale.
setClass("variabilaAleatoare",
    slots=list(intervaleDensitate="vector",#vector de intervale
    functiiDensitate="vector")) #vector de functii pe intervale
#valoarea functiei pe cazul "altfel" se considera 0

#Functie pentru a construi o noua variabila aleatoare
creareVariabilaTastatura <- function(tipVariabila){
  #Un nou obiect din clasa variabilaAleatoare
  var <- new("variabilaAleatoare")

  #Ii cer utilizatorului numarul de intervale pe care este definita densitatea
  nr_intervale_f_densitate <- as.integer(readline("Introduceti numarul de intervale/ramuri: "))-1

  #Dimensiunea vectorului este = nr_intervale_f_densitate-1 pentru ca tratez separat cazul "altfel" nu este nevoie sa retin o functie cu un interval aferent, se considera automat valoarea 0
  #Un vector in care retinem intervalele pe care este definita densitatea de repartitie (vector de liste)
  var@intervaleDensitate <- vector(mode = "list", nr_intervale_f_densitate)

  #Un vector in care retinem functia densitatii pe intervalul corespunzator
  var@functiiDensitate <- vector(mode = "list", nr_intervale_f_densitate)

  #Se citeste fiecare interval impreuna cu densitatea corespunzatoare
  for(i in 1:nr_intervale_f_densitate)
  {
    #Se retine functia (ca un string)
    var@functiiDensitate[[i]] <- readline("Introduceti functia: ")

    #Se retine si intervalul pe care aceasta este definita
```

```r
    startIntervalx <- as.integer(readline("Introduceti inceputul intervalului pentru x: "))
    finishIntervalx <- as.integer(readline("Introduceti sfarsitul intervalului pentru x: "))

    #Daca este variabila aleatoare bidimensionala, trebuie sa retinem si intervalul pentru y
    if(tipVariabila==2)
    {
      startIntervaly <- as.integer(readline("Introduceti inceputul intervalului pentru y: "))
      finishIntervaly <- as.integer(readline("Introduceti sfarsitul intervalului pentru y:"))

      var@intervaleDensitate[[i]] <- list(startIntervalx, finishIntervalx, startIntervaly, finishIntervaly)
    }
    else{
      var@intervaleDensitate[[i]] <- list(startIntervalx, finishIntervalx)
    }
  }

  #Returnez noul obiect construit (variabila aleatoare)
  return(var)
}

creareVariabilaParam <- function(vectorIntervale,vectorFunctii, valoareOutIntervale){
  #Un nou obiect din clasa variabilaAleatoare
  var <- new("variabilaAleatoare")
  var@intervaleDensitate <- vectorIntervale
  var@functiiDensitate <- vectorFunctii

  return(var)
}

#Daca se doreste o variabila aleatoare continua unidimensionala -> se apeleaza creareVariabila(1)
#Daca se doreste o variabila aleatoare continua bidimensionala -> se apeleaza creareVariabila(2)
varX <- creareVariabilaTastatura(1)

#numarul de "ramuri" ale functiei = length(varX@intervaleDensitate)

#Daca nu se doreste citirea de la tastatura, se va trimite ca parametru: vector de intervale si vector de functii
#Test
vectorIntervale <- vector(mode = "list", 1)
vectorIntervale[[1]] <- list(0,1)
vectorFunctii <- vector(mode = "list", 1)
vectorFunctii[[1]] <- "1/(2*sqrt(x)"

varX <- creareVariabilaParam(vectorIntervale,vectorFunctii)
```

```
> varX <- creareVariabilaParam(vectorIntervale,vectorFunctii)
> varX
An object of class "variabilaAleatoare"
Slot "intervaleDensitate":
[[1]]
[[1]][[1]]
[1] 0

[[1]][[2]]
[1] 1



Slot "functiiDensitate":
[[1]]
[1] "1/(2*sqrt(x)"
```

Figura 3: Exemplu rulare clasa pentru variabile aleatoare

5 Reprezentarea grafică a densității și a funcției de repartiție (4)

Cerință: Reprezentarea grafică a densității și a funcției de repartiție pentru diferite valori ale parametrilor repartiției. În cazul în care funcția de repartiție nu este dată într-o formă explicită(ex. repartiția normală) se acceptă reprezentarea grafică a unei aproximări a acesteia.

Noțiuni teoretice: Funcția de distribuție a unei variabile aleatoare continue X poate fi exprimată ca integrala probabilității densității funcției f_X după cum urmează:

$$F_X(x) = \int_{-\infty}^{x} f_X(t)\, dt$$

Implementare: Pentru funcția $afisareDensitate$ creez o secvență de valori care reprezintă intervalul $[a, b]$ având lungimea s. Utilizez funcția *density* pentru a retine un obiect de tip densitate în d pe care mai apoi o afișez folosind funcția *plot*. Pentru funcția $afisareDistributie$ definesc domeniul ca fiind intervalul $[a, b]$ cu s elemente. Folosindu-mă de formula distribuției rețin în $values$ valorile funcției de distribuție, iar mai apoi utilizând plot, afișez graficul corespunzător.

```
afisareDensitate <- function(f,a,b,s) {
  x <- seq(a,b,length=s)
  d <- density(f(x))
  plot(d)
}

afisareDistributie <- function(f,a,b,s){
  domain <- seq(a,b,length=s)
  values <- c()

  for(x in domain){
    aux <- integrate(f,-Inf, x)$value
    values<-append(values,aux)
  }
  plot(x = domain, y = values)

}

f <- function(x) {
  1/(1+x^2)
}
afisareDensitate(f,0,3,10000)
afisareDistributie(f,0,3,10000)
```

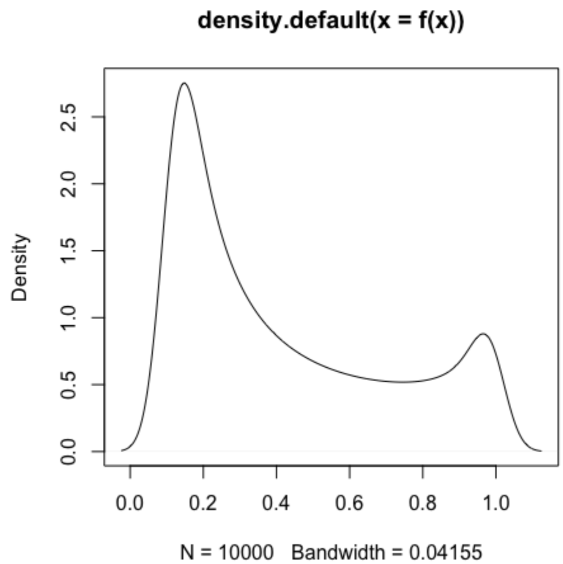


Figura 4: Exemplu afișare densitate pentru $f(x) = 1/(1 + x^2)$

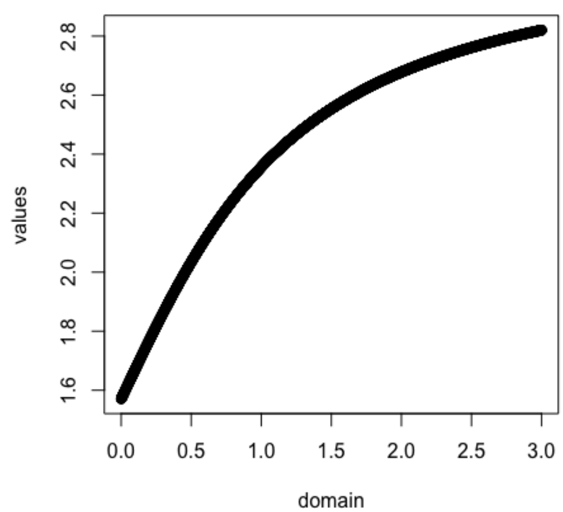


Figura 5: Exemplu afișare distribuție pentru $f(x) = 1/(1 + x^2)$

6 Calculul mediei, dispersiei și a momentelor inițiale centrate (5)

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

Implementare: Pentru a rezolva această cerință am implementat o funcție, $calcMoment$, care utilizează funcția $calcMedieCont$(utilizată în cadrul exercițiului 6). Pentru a determina fiecare moment inițial până la momentul 4 apelăm de fiecare dată $calcMedieCont$, dând ca parametru x ridicat la puterea indicată de ordin, conform formulei:

$$Moment\ inițial = E[x^{ordin}]$$

Pentru determinarea momentelor centrate problema este gestionata asemănător, urmând să le calculăm folosindu-ne de formula:

$$Moment\ centrat = E[(x - E[x])^{ordin}]$$

Funcția este creată într-un mod în care să permită determinarea momentelor și pe intervale mărginte superior și/sau inferior. În cazul în care utilizatorul pachetului dorește acest lucru, atunci pentru determinarea momentelor el va apela funcția: $calcMoment(funcție, limităInferioară, limităSuperioară)$. În cazul în care intervalul nu este mărginit, atunci momentele se vor calcula standard, i.e. media lui x la puterea ordinului va fi integrala de la minus inifinit la plus infinit. În cazul în care unul dintre aceste momente nu mai poate fi calculat (integrala este una divergenta sau nu se poate calcula) atunci va fi generată o eroare tratată printr-un bloc de tipul tryCatch. Dacă aceasta apare la determinarea unui moment atunci programul afișează un mesaj corespunzător și oprește execuția. Rezultatele sunt puse într-o matrice de 2 linii, prima linie reprezentând toate momentele inițiale calculate până în momentul respectiv, iar a doua linie reprezintă valorile momentelor centrate.

```
calcMoment<-function(variabila,limInf=-Inf,limSup=Inf){
  ordin=1
  #Cream un vector de momente initiale pe care sa il returnam
  momenteInitiale<-c()
  #Cream un vector de momente centrate pe care sa il returnam
  momenteCentrate<-c()
  ok1=0
  ok2=0

  #Calculez momementele pana la ordinul 4 sau pana unde acestea se
    mai pot calcula
  while(ok1==0 && ok2==0 && ordin<=4)
  {
    #Pentru formula avem nevoie de x^ordin * f(x)
    g<-function(x){x^ordin}
```

```
    check<-tryCatch({
      media<-calcMedieCont(variabila,g,limInf,limSup)
    },error=function(err){
      print(paste('Nu se mai pot calcula momente initiale!Incepand cu ordinul:',ordin))
      return(-1)

    })
    if(check==-1){
      rez<-rbind(momenteInitiale,momenteCentrate)
      return(rez)
      stop()
      break
      exit()
    } else{
      media<-calcMedieCont(variabila,g,limInf,limSup)
      #Adaugam momentul initial de ordin r in vector in cazul in care s-a putut calcula
      momenteInitiale<-c(momenteInitiale,media)
      #Daca am reusit sa calculam momentul initial de ordin r calculam mai departe
      h<-function(x){(x-media)^ordin}
      ok2=0
      check2<-tryCatch({
        #Adaugam momentul centrat de ordin r in vector in cazul in care s-a putut calcula
        momCentrat<-calcMedieCont(variabila,h,limInf,limSup)
      },error=function(err){
        print(paste('Nu se mai pot calcula momente centrate! Incepand cu ordinul:',ordin))
        return(-1)
      })
      #Daca nu s-a putut calcula, oprim functia
      if(check2==-1) {
        rez<-rbind(momenteInitiale,momenteCentrate)
        return(rez)
        stop()
        break
      } else{ #In caz contrar mergem mai departe
        #Adaugam momentul centrat de ordin r in vector in cazul in care s-a putut calcula
        momCentrat<-calcMedieCont(variabila,h,limInf,limSup)
        momenteCentrate<-c(momenteCentrate,momCentrat)
        ordin=ordin+1
      }
    }
  }
  rez<-rbind(momenteInitiale,momenteCentrate)
  return(rez)
}

#Teste
h<-function(x){x^2}
aux1<-calcMoment(h,1,2)
aux1
aux1[2]
```

```
> #Teste
> h<-function(x){x^2}
> aux1<-calcMoment(h,1,2)
> aux1
                  [,1]     [,2]      [,3]         [,4]
momenteInitiale  3.75  6.20000    10.500     18.14286
momenteCentrate -5.00 49.39333 -1645.612 174734.64704
> aux1[2]
[1] -5
```

Figura 6: Exemplu rulare funcție calcMoment()

7 Calculul mediei și dispersiei unei variabile aleatoare (6)

Cerință: 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.

Implementare: Pentru determinarea mediei și dispersiei unei variabile aleatoare $g(X)$, unde X are o repartiție comună cunoscută vom avea nevoie de formula mediei unei variabile aleatoare continue după cum urmează:

$$E[x] = \int_{-\infty}^{\infty} g(x) \cdot f(x)\, dx$$

Am definit două funcții: $calcMedieCont$ și $calcDispCont$ care primesc ca parametru funcția $g(x)$, repartiția și opțional limita superioară și/sau limita inferioară după care integrăm. Pentru determinarea dispersiei vom utiliza funcția $calcMedieCont$, urmând să respectăm formula

$$Var(x) = E[(x - E[x])^2]$$

După cum se poate observa, inițial programul va determina media utilizând funcția definită anterior, iar apoi, la nivel local va calcula dispersia integrând parametrii corespunzători fără a mai apela alte metode.

```
#Calculez media->ca valori primim functia si eventual capetele
    intervalului, in caz contrar va ramane integrala standard de la
     minus infinit la infinit
calcMedieCont<-function(func,G,limInf=-Inf,limSup=Inf){

  parametru<-function(x){G(x)*func(x)}
  #Aplicam integrala din functia primita ca parametru*repartitia
  rez<-integrate(parametru,lower=limInf,upper=limSup)$value
  return(rez)
}

#Teste
h<-function(x){x^(1/2)}
g<-function(x){3+x}
afis<-calcMedieCont(h,g,1,2)
afis

#Pentru calculul dispersiei vom folosi acelasi principiu
#Pentru a obtine media apelam functia definita anterior
#Aplicam formula pentru dispersie

calcDispCont<-function(func,G,limInf=-Inf,limSup=Inf)
{
  #Calculam dispersia=momentul centrat de ordin 2
  media<-calcMedieCont(func,G,limInf,limSup)
```

```
  parametru<-function(x){((x-media)^2)*func(x)}
  rez<-integrate(parametru,lower=limInf,upper=limSup)$value
  return(rez)
}

#Teste
afis2<-calcDispCont(h,g,0,3)
afis2
```

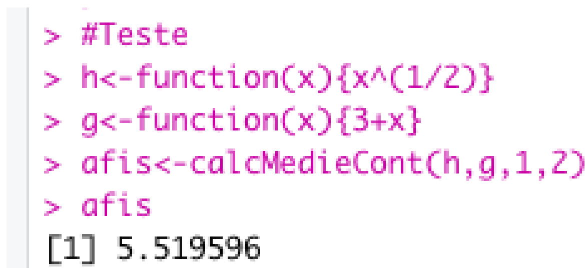


Figura 7: Testare funcție $calcMedieCont()$

Figura 8: Testare funcție $calcDispCont()$

8 Fișe de sinteză (8)

Cerință: Afișarea unei "fișe de sinteză" care să conțină informații de bază despre respectiva repartiție(cu precizarea sursei informației!). Relevant aici ar fi să precizați pentru ce e folosită în mod uzual acea repartiție, semnificația parametrilor, media, dispersia etc.

Implementare: Fiecare repartiție este reprezentată de un vector de string-uri cu informațiile aferente și în momentul apelării se dă un număr între 1 și 6, care afișează fișa corespunzătoare.

```
#Functia primeste ca parametru un numar care corespunde unei repartitii si se afiseaza informatiile aferente repartitiei cerute.
afisare_fisa_sinteza <- function(nr_fisa){
  if(nr_fisa==1){
    cat( c("REPARTITIA UNIFORMA\n
    O variabila aleatoare continua X este repartizata uniform pe intervalul (a,b) daca admite densitatea de repartitie:\n
    f(x) = 1/(b-a), x este in intervalul (a,b)\n
         = 0, in caz contrar\n
    Notatie:\n
    X ~ U((a,b)), unde (a,b) este intervalul pe care este repartizata variabila aleatoare X\n
    Media:\n
    E[X] = integrala de la a la b din x*f(x) dx = (a+b)/2\n
    Dispersia:\n
    Var(X) = E[X^2]-E[x]^2\n
    E[X^2] = integrala de la a la b din (x^2)*f(x) dx = (a^2 + a*b + b^2)/3\n
    Prin urmare, Var(X) = ((b-a)^2)/12\n
    Sursa: Probabilitati si Statistica Curs 9, Prof. Alexandru Amarioarei"))
  }

  if(nr_fisa==2){
    cat( c("REPARTITIA EXPONENTIALA\n
    O variabila aleatoare continua X este repartizata exponential de parametru lambda > 0 daca admite densitatea de repartitie:\n
    f(x) = lambda * e^(-lambda*x), x > 0\n
         = 0, in caz contrar\n
    Utilizare: Modeleaza timpul de asteptare pana la aparitia unui eveniment de interes.\n
    Notatie:\n
    X ~ Exp(lambda)\n
    Media:\n
    E[X] = integrala de la -Inf la +Inf din x*f(x) dx = 1/lambda \n
    Dispersia:\n
    Var(X) = E[X^2]-E[x]^2\n
    E[X^2] = integrala de la -Inf la +Inf din (x^2)*f(x) dx = 2/(lambda^2)\n
    Prin urmare, Var(X) = 1/(lambda^2)\n
    Sursa: Probabilitati si Statistica Curs 10, Prof. Alexandru Amarioarei"))
```

```
  }

  if(nr_fisa==3){
    cat( c("REPARTITIA NORMALA\n
    O variabila aleatoare continua X este repartizata normal (sau Gaussian) de parametru miu si sigma^2 daca admite densitatea de repartitie:\n
    f(x) = (1/(sqrt(2pi)*sigma))*e^(-(x-miu)^2/2*sigma^2), x in R \n
    Notatie:\n
    X ~  N(miu,sigma^2)\n
    In cazul in care miu = 0 si sigma = 1 spunem ca v.a. X este repartizata normal standard si notam X ~ N(0,1) si putem calcula:\n
    Media:\n
    E[X] = integrala de la -Inf la +Inf din x*f(x) dx = 0 \n
    Dispersia:\n
    Var(X) = E[X^2]-E[x]^2\n
    E[X^2] = integrala de la -Inf la +Inf din (x^2)*f(x) dx = 1\n
    Prin urmare, Var(X) = 1\n
    Sursa: Probabilitati si Statistica Curs 10, Prof. Alexandru Amarioarei"))
  }

  if(nr_fisa==4){
    cat(c("\nREPARTITIA GAMMA\n
    O variabila aleatoare X este distribuita gamma cu parametrii lambda si p daca are densitatea de probabilitate de forma:\n
    f(x) = (lambda^p * x^(p-1) * e^(-lambda*x))/gamma(p), x > 0 \n
    Notatie:\n
    X ~ Gamma [p, lambda]\n
    Media:\n
    E[X] = integrala de la -Inf la +Inf din x*f(x) dx = p/lambda \n
    Dispersia:\n
    Var(X) = E[X^2]-E[x]^2 = p/lambda^2\n
    Sursa: http://math.etc.tuiasi.ro/rstrugariu/cursuri/SPD2015/c7.pdf"))
  }

  if(nr_fisa==5){
    cat(c("\nREPARTITIA X-PATRAT\n
    O variabila aleatoare X este distribuita X-patrat cu n grade de libertate daca are densitatea de probabilitate de forma:\n
    f(x) = (1/(2^(n/2) * gamma(n/2))) * x^((n/2)-1) * e^(-x/2), x > 0 \n
    Notatie:\n
    X ~ X^2(n), n reprezentand numarul de grade de libertate\n
    Media:\n
    E[X] = integrala de la -Inf la +Inf din x*f(x) dx = n \n
    Dispersia:\n
    Var(X) = E[X^2]-E[x]^2 = 2n \n
    Sursa: https://cismasemanuel.files.wordpress.com/2020/04/seminar-variabile-aleatoare-continue-1.pdf"))
  }

  if(nr_fisa==6){
    cat(c("\nREPARTITIA STUDENT\n
```

```
    O variabila aleatoare X este distribuita Student cu n grade de
    libertate daca are densitatea de probabilitate de forma:\n
    f(x) = (gamma((n+1)/2))/(gamma(n/2) * sqrt (n*pi)) * (1+x^2/n)
    ^(-(n+1)/2), x apartine R \n
    Notatie:\n
    X ~ T(n), n reprezentand numarul de grade de libertate\n
    Media:\n
    E[X] = integrala de la -Inf la +Inf din x*f(x) dx = 0 \n
    Dispersia:\n
    Var(X) = E[X^2]-E[x]^2 = n/(n-2) \n
    Sursa: http://math.etc.tuiasi.ro/rstrugariu/cursuri/SPD2015/c7.
    pdf"))
  }
  afisare_fisa_sinteza(4)
}
```

```
> afisare_fisa_sinteza(4)

REPARTITIA GAMMA

        O variabila aleatoare X este distribuita gamma cu parametrii lambda si p daca are densitatea de probabilitate de forma:

        f(x) = (lambda^p * x^(p-1) * e^(-lambda*x))/gamma(p), x > 0

        Notatie:

        X ~ Gamma [p, lambda]

        Media:

        E[X] = integrala de la -Inf la +Inf din x*f(x) dx = p/lambda

        Dispersia:

        Var(X) = E[X^2]-E[x]^2 = p/lambda^2

        Sursa: http://math.etc.tuiasi.ro/rstrugariu/cursuri/SPD2015/c7.pdf>
```

Figura 9: Exemplu afișare fișă de sinteză

9 Covarianța și coeficientului de corelație (10)

Cerință: Calculul covarianței și coeficientului de corelație pentru două variabile aleatoare continue(Atenție:Trebuie să folosiți densitatea comună a celor două variabile aleatoare!)

Noțiuni teoretice: Pornind de la densitatea comună a două variabile aleatoare, putem afla densitatea marginală a acestora. Covarianța variabilelor X și Y poate fi definită astfel:

$$Cov[X,Y] = E[(X - E[X]) \cdot (Y - E[Y])],\ unde\ E[X] = media\ lui\ X$$

$$Cov[X,Y] = E[XY] - E[X] \cdot E[Y]$$

$$Cov[X,Y] = \iint (X - E[X])(Y - E[Y]) \cdot f(x,y)\,dx\,dy$$

$$\rho(x,y) = \frac{Cov(X,Y)}{\sqrt{Var(X)} \cdot \sqrt{Var(Y)}}$$

```r
#Calcul medie (ca valori primim functia si intervalul)
medie<-function(func,limInf=-Inf,limSup=Inf){
  parametru<-function(x){x*func(x)}
  rez<-integrate(parametru,lower=limInf,upper=limSup)$value
  return(rez)
}

#Calcul dispersie = momentul centrat de ordin 2
varianta<-function(func,limInf=-Inf,limSup=Inf)
{
  #Calculez media
  media<-medie(func,limInf,limSup)
  parametru<-function(x){((x-media)^2)*func(x)}
  rez<-integrate(parametru,lower=limInf,upper=limSup)$value
  return(rez)
}

# Calcul covarianta
covarianta <- function(fx,fy,f_dens_comuna, limInf=Inf, limSup=Inf)
     {
  #Pachet pentru "integral2"
  install.packages('pracma')
  library('pracma')

  #Fx si Fy ar trebui sa fie densitatile marginale ale lui X si Y
  #Integrarea ma constrange sa dau o valoarea, deci densitatea
    calculata la exercitiul 11 nu va fi o functie,
  #ci o valoare numerica, de aceea nu o pot folosi
  #De aceea, functiile trebuie date ca parametru separat

  mediaX <- as.numeric(medie(fx,limInf, limSup))
  mediaY <- as.numeric(medie(fy,limInf, limSup))

```

```
  #cov <- medie((x-mediaX)*(y-mediaY),limInf, limSup)
  #Covarianta = E[XY] - E[X]*E[Y] = E[(X-E[X])(Y-E[Y])]

  #Covarianta = integrala dubla din (x-E[x])(y-E[y])*f(x,y) dx dy
  functCov <- function(x,y){(x-mediaX)*(y-mediaY)*noquote(trimws(deparse(f_dens_comuna)[3]))}   #nu merge agregarea functiilor
  cov <- integral2(functCov,limInf, limSup, limInf, limSup)

  return (cov)
}


# Calculul coeficientului de corelatie
coeficient_corelatie <- function(fx, fy, limInf=Inf, limSup=Inf)
{
  #Coeficientul de corelatie = covarianta(X,Y) / (sqrt(varianta(X)) * sqrt(varianta(Y)))
  cov <- covarianta(fx, fy, limInf, limSup)
  return (cov / (sqrt(varianta(fx,limInf, limSup)) * sqrt(varianta(fy,limInf, limSup))))
}
```

10 Densitate marginală și condiționate (11)

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

Noțiuni teoretice: Densitatea marginala a unei variabile aleatoare continue X se poate determina integrând de la $-\infty$ la $+-\infty$ funcția densității comune în raport cu y:

$$f_X(x) = \int_{-\infty}^{\infty} f_{X,Y}(x,y)\,dy$$

Densitatea condițională a unei variabile aleatoare continue X la Y=y se poate determina prin raportul dintre funcția densității comune și densitatea marginala a lui Y:

$$f_Y(y) = \int_{-\infty}^{\infty} f_{X,Y}(x,y)\,dx$$

$$f_{X|Y}(x|y) = \frac{f_{x,y}(x,y)}{f_y(y)}$$

```
#Pornind de la densitatea comuna a doua variabile aleatoare
    continue, construirea densitatilor marginale si a densitatilor
    conditionate.

#Densitatea marginala
#fx(x) = integrala(fx,y(x,y)dy, -Inf, Inf)
#fy(y) = integrala(fx,y(x,y)dx, -Inf, Inf)

densitatea_marginala <- function(f_dens_comuna, startInterval = -
    Inf, finishInterval =Inf, punctValoareX=1, punctValoareY=1)
{
  #Sunt obligata sa dau o valoare pentru x/y
  densitateMarginalaX<-function(x){f_dens_comuna(x,y=punctValoareY)
    }
  densitateMarginalaY<-function(y){f_dens_comuna(y,x=punctValoareX)
    }

  #Construirea densitatilor marginale
  densitate_marginala_x <- integrate(densitateMarginalaY, lower =
    startInterval, upper = finishInterval)$value #trebuie sa
    derivez in functie de y
  densitate_marginala_y <- integrate(densitateMarginalaX, lower =
    startInterval, upper = finishInterval)$value  #trebuie sa
    derivez in functie de x

  return(c(densitate_marginala_x, densitate_marginala_y))
}


densitatea_conditionata <- function(f_dens_comuna,  punctValoareX
    =1, punctValoareY=1,startInterval=-Inf, finishInterval=Inf)
{
```

```r
  #Contruirea densitatii conditionate
  densitati_marginale <- densitatea_marginala(f_dens_comuna,
    startInterval, finishInterval, punctValoareX, punctValoareY)
  densitate_marginala_x <- as.numeric(densitati_marginale[1])
  densitate_marginala_y <- as.numeric(densitati_marginale[2])

  dens_com_XY <- f_dens_comuna(punctValoareX,punctValoareY)
  densitate_condit_x_la_y <- (dens_com_XY / densitate_marginala_y)
  densitate_condit_y_la_x <- (dens_com_XY / densitate_marginala_x)

  return(c(densitate_condit_x_la_y, densitate_condit_y_la_x))
}

Z <- function(x,y){x+y^4}
dens_marg <- densitatea_marginala(Z,0,4,1,1)
dens_margX <- as.numeric(dens_marg[1])
dens_margY <- as.numeric(dens_marg[2])
dens_margX
dens_margY
```

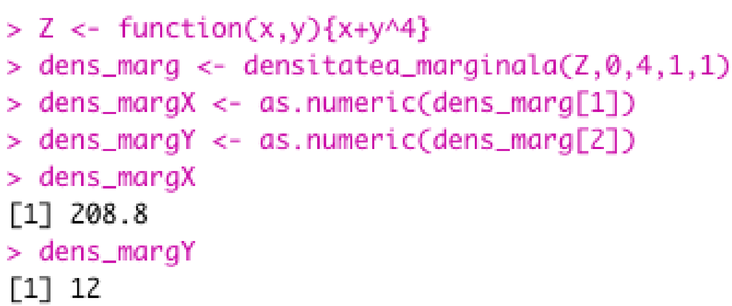


Figura 10: Exemplu afișare densități

11 Suma și diferența a două variabile aleatoare continue independente (12)

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

Noțiuni teoretice: Fie X și Y două variabile aleatoare continue independente cu funcțiile densitate de probabilitate $f(x)$ și $g(y)$. Considerăm $f(x)$ și $g(y)$ definite pe R. Atunci suma Z = X + Y este o variabile aleatoare continuă cu funcția de densitate $f_Z(z)$, unde $f_Z(z)$ este convoluția funcțiilor $f(x)$ și $g(y)$, dată de:

$$f_Z(z) = \int_{-\infty}^{\infty} f(z-y)g(y)\,dy$$

$$f_Z(z) = \int_{-\infty}^{\infty} g(z-x)f(x)\,dx$$

Pentru a calcula diferența între X și Y vom scrie:

$$f_Z(z) = \int_{-\infty}^{\infty} f(y-z)g(y)\,dy$$

Implementare:

```r
# Suma Z = X + Y
# fz(z) = integrala de la -Inf la Inf din (fx(z-y)fy(y)dy)
# fx = functia de densitate a lui x
# fy = functia de densitate a lui y

sumaVarIndep <- function(fx,fy, limInf= -Inf, limSup =Inf){
  function(z) (integrate(function(y,z) fy(y)*fx(z-y),-limInf,limSup
    ,z)$value)
}

# Test - nu merge
fx <- function(x) dnorm(x,1,0.5)
fy <- function(y) dlnorm(y,1.5, 0.75)
fz <- sumaVarIndep(fx,fy)
Vectorize(fz)


# Diferenta Z = X - Y
#fz(z) = integrala de la -Inf la Inf din (fx(y-z)fy(y)dy)
difVarIndep <- function(fx,fy, limInf= -Inf, limSup =Inf){
  function(z) (integrate(function(y,z) (fx(y-z) * fy(y)), limInf,
    limSup,z)$value)
}
```

12 Construirea pachetului R

Pentru a construi pachetul R am urmat informațiile din documentul transmis. Mai jos găsiți screenshot-uri din momentul construirii pachetului:

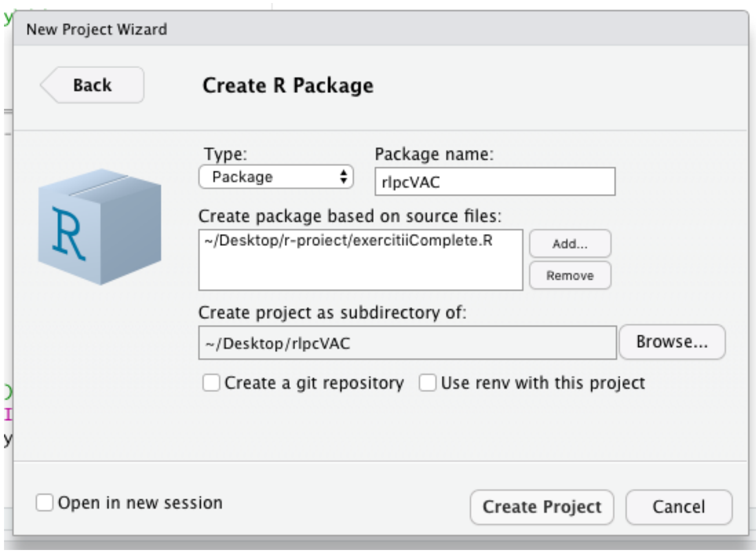


Figura 11: Creare pachet

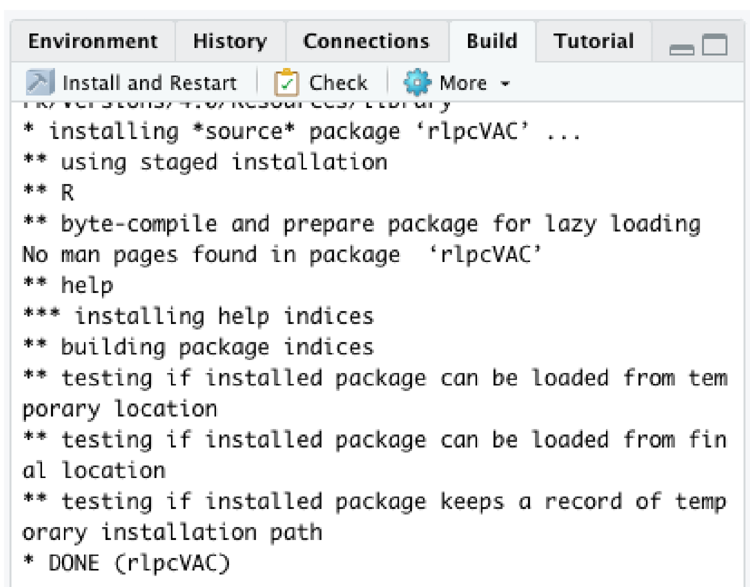


Figura 12: Compilare pachet

```
Console  Terminal ×  Jobs ×
~/Desktop/rlpcVAC/rlpcVAC/
Restarting R session...

> library(rlpcVAC)
> test <- function(x){1/(x*x + 1)}
> print(constantaNorm(test))
[1] 0.3183099
> print(esteDensitate(test))
[1] TRUE
>
> vectorIntervale <- vector(mode = "list", 1)
> vectorIntervale[[1]] <- list(0,1)
> vectorFunctii <- vector(mode = "list", 1)
> vectorFunctii[[1]] <- "1/(2*sqrt(x)"
>
> varX <- creareVariabilaParam(vectorIntervale,vectorFunctii)
> varX
An object of class "variabilaAleatoare"
Slot "intervaleDensitate":
[[1]]
[[1]][[1]]
[1] 0

[[1]][[2]]
[1] 1



Slot "functiiDensitate":
[[1]]
[1] "1/(2*sqrt(x)"


> h<-function(x){x^2}
> aux1<-calcMoment(h,1,2)
> aux1
                  [,1]     [,2]      [,3]         [,4]
momenteInitiale  3.75  6.20000    10.500     18.14286
momenteCentrate -5.00 49.39333 -1645.612 174734.64704
> aux1[2]
[1] -5
> h<-function(x){x^(1/2)}
> g<-function(x){3+x}
> afis<-calcMedieCont(h,g,1,2)
> afis
[1] 5.519596
> afis2<-calcDispCont(h,g,0,3)
> afis2
[1] 763.7565
```

Figura 13: Exemplu utilizare funcții din pachet

13 Concluzie

Realizarea acestui proiect a reprezentat o întreagă muncă de echipă și de aprofundare a cunoștințelor în programul RStudio, dobândite în cadrul cursului de Probabilități și Statistică.

Volumul mare de informație, formule, excepții și concepte a atras după sine o evoluție nu atât de progresivă a proiectului pe cat de mult ne-am fi dorit, însă pe măsură ce noțiunile erau din ce în ce mai fixate, clare și concise realizarea pachetului a devenit mult mai ușoară.

Ne bucurăm că am reușit să implementăm funcții cu o utilitate atât de mare, mai ales în domeniul de prelucrare a datelor și obținerea unor statistici referitoare la diferite evenimente.

Nu numai că am reușit să înțelegem importanța analizării datelor prin intermediul calculelor precise, însă am descoperit și o mică parte din ceea ce înseamnă Machine Learning, domeniu care asigură algoritmi cu o acuratețe foarte ridicată, testată prin analiza datelor.

În concluzie, a fost un proiect care a necesitat destul de multă muncă și implicare, însă care a atras după sine o îmbogățire a bagajului de cunoștințe în ceea ce privește programarea.

14 Bibliografie

Probabilități și Statistică Curs 2020-2021, Prof. Alexandru Amărioarei

https://www.datamentor.io/r-programming/object-class-introduction/

https://www.geeksforgeeks.org/classes-in-r-programming/

https://win-vector.com/2018/03/06/r-tip-use-vectormode-list-to-pre-allocate-lists/

http://math.etc.tuiasi.ro/rstrugariu/cursuri/SPD2015/c7.pdf

https://cismasemanuel.files.wordpress.com/2020/04/seminar-variabile-aleatoare-continue-1.pdf

http://math.etc.tuiasi.ro/rstrugariu/cursuri/SPD2015/c7.pdf

https://en.m.wikipedia.org/wiki/Normalizing_constant

https://en.m.wikipedia.org/wiki/Cumulative_distribution_function

https://stats.libretexts.org/Bookshelves/Probability_Theory/Book%3A_Introductory_Probability_(Grinstead_and_Snell)/07%3A_Sums_of_Random_Variables/7.02%3A_Sums_of_Continuous_Random_Variables