**PROBABILITĂȚI ȘI STATISTICĂ**

**PROIECT**

**CIURESCU IRINA – ALEXANDRA (lider)**

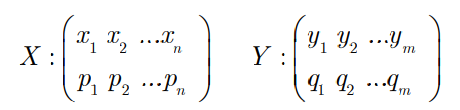
**STOINEA MIRUNA-MARIA**

**GRUPA 242**

Punctul 1

Pachete software folosite **discreteRV**

1. Fie două variabile aleatoare discrete X şi Y cu repartițiile:



1. Construiţi o funcţie ***frepcomgen*** care primeşte ca parametri m şi n şi care generează un tabel cu repartiţia comună a v.a. X şi Y incompletă, dar ȋntr-o formă ȋn care poate fi completată ulterior.

**Rezolvare:**

Funcția ***frepcomgen*** va returna o lista cu 5 elemente:

* n – de tip numeric
* m – de tip numeric
* X – de tip RV
* Y – de tip RV
* M – matricea în care este memorată repartiția comună a v.a. X și Y

**Matricea M** trebuie să îndeplinească următoarele condiții:

* C1) are n coloane și m linii
* C2) suma elementelor trebuie să fie egală cu 1
* C3) suma elementelor de pe linia i trebuie sa fie egală cu pi
* C4) suma elementelor de pe coloana j trebuie sa fie egală cu qj

Pentru condiția **C2)** am generat inițial o matrice cu n linii și m coloane valori n\*m valori aleatoare din intervalul [1, n\*m]. Am calculat suma elementelor matricei și am împărțit fiecare element la suma obținută. Astfel suma elementelor matricei va fi 1.

**Pentru variabila aleatoare discretă X:**

* am generat aleator valorile posibile **x1, x2, ..... xn**
* valorile probabilităților asociate le-am determinat astfel:

**pj=suma elementelor de pe linia i din matricea generată**

pentru orice i=1, 2, ..... n

* am apelat **funcția RV()** din pachetul **discreteRV** pentru generarea variabilei X

**Pentru variabila aleatoare discretă Y:**

* am generat aleator valorile posibile **y1, y2, ..... ym**
* valorile probabilităților asociate le-am determinat astfel:

**qi=suma elementelor de pe coloana j din matricea generată**

pentru orice j=1, 2, ..... m

* am apelat **funcția RV()** din pachetul **discreteRV** pentru generarea variabilei Y

Pentru că funcția trebuia să returneze o repartiție incompletă (care putea fi completată ulterior) am înlocuit câte o valoare de pe fiecare linie cu -1 (coloana pe care se află această valoare am generat-o aleator).

#Functia primeste ca parametri 2 numere n si m si returneaza o lista cu 5 elemente

# n

# m

# X de tin RV

# Y de tip RV

# M - matrice ce memoreaza repartitia comuna

# Nume campuri lista returnata: n,m,x,y,M

frepcomgen <- function(n, m) {

# Generare matrice M cu n linii și m coloane,

#unde suma elementelor este 1 (repartitia comuna)

valori <- sample(seq(1,100), n\*m)

matrice <- matrix(valori, nrow = n, ncol = m)

suma\_elemente <- sum(matrice)

matrice <- matrice / suma\_elemente

# Generare variabila aleatoare discreta x

# 1) generare valori

vx <- sample(1:1000, n, replace = FALSE)

# 2) calculare probabilitati - suma valorilor de pe

# fiecare linie din repartitia comuna M

probx <- rowSums(matrice)

# 3) creare variabila aleatoare x

x <- RV (vx, probx)

# Generare variabila aleatoare discreta y

# 1) generare valori

vy <- sample(1:1000, m, replace = FALSE)

# 2) calculare probabilitati - suma valorilor de pe fiecare

#coloana din repartitia comuna M

proby <- colSums(matrice)

# 3) creare variabila aleatoare y

y <- RV (vy, proby)

print("Matricea dupa generarea initiala:")

print(matrice)

#Stergem cate o valoare de pe fiecare linie

#ca sa obtinem o matrice incompletaGeneram aleator numarul

#coloanei de pe care vom elimina elementul de pe fiecare linie

#Elementele necompletate vor avea valoarea -1

for(i in 1:n)

{ j <- sample(seq(1,m),1)

matrice[i,j] <- -1

}

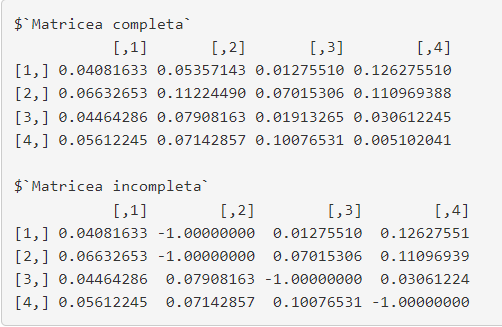
print("Matricea incompleta (valorile lipsa sunt egale cu -1:")

print(matrice)

lista\_rezultat <- list(n=n, m=m, x = x, y = y, M = matrice)

return(lista\_rezultat)

}



1. Construiţi o funcţie **fcomplrepcom** care completează repartiţia comună generată la punctul anterior (pentru cazul particular sau pentru cazul general).

**Rezolvare:**

Funcția **fcomplrepcom**  va fi primi ca parametru o listă cu 5 elemente (returnată de funcția ***frepcomgen)*** și va returna o variabilă de tip RV reprezentând repartiția comună a celor 2 variabile aleatoare discrete X și Y.

Pentru completarea repartiției incomplete transmisă ca parametru:

* determinăm numărul valorile egale cu -1 din matricea M și pozițiile acestora
* încercăm să completăm valorile (trebuie să existe cel mult o valoare necompletată pe linia sau coloana corespunzătoare)
* reluăm procedeul până când au fost completate toate valorile

Pe baza variabilelor X și Y de tip RV și pe baza matriciei M generăm repartiția comună cu ajutorul funcției **jointRV()** din pachetul **discreteRV.**

# Functia fcomplrepcom primeste ca parametru o lista L (cu cele 5 elemente precizate mai sus)

# si returneaza o variabila de tip RV reprezentand repartitia comuna (obtinuta cu functia jointRV)

# Pentru a completa matricea:

# cautam un element necompletat, fie acesta M[i,j]

# verificam daca ii putem determina val. calculand cate valori lipsesc pe linia i / coloana j

# - daca lipseste doar o singura valoare pe linia/coloana lui atunci ii completam valoarea

# - in completare ne bazam pe repartitiile marginale X si Y

fcomplrepcom <- function(L)

{ n <- L$n; m <- L$m; X <- L$x; Y <- L$y; M <- L$M

#preluam din variabilele aleatoare X si Y probabilitatile si valorile

probX = slot(X, "probs")

probY = slot(Y, "probs")

valuesX = slot(X, "outcomes")

valuesY = slot(Y, "outcomes")

#cautam valori egale cu -1 (necompletate)

contor = any(M ==-1 )

#cat timp mai sunt valori necompletate

while (contor > 0)

{ # preluam pozitiile elementelor egale cu -1 intr-o matrice unde fiecare rand

# reprezinta o pereche (linie,coloana) unde se gaseste x

pozitii = which(M == -1, arr.ind = TRUE)

# incercam sa completam fiecare pozitie necompletata

for (i in 1:nrow(pozitii))

{ #preluam pozitia unei valori lipsa

linie = pozitii[i,1]

coloana = pozitii[i,2]

#verificam daca se poate calcula valoarea elementului de la pozitia (linie,coloana)

#trebuie sa avem un singur -1 pe linia sau coloana elementului

numar\_elemente\_minus\_unu = sum(M[linie, ] == -1)

if(numar\_elemente\_minus\_unu == 1)

{ suma\_elemente\_pe\_linie = sum(M[linie, ]) + 1

M[linie,coloana] = probX[linie] - suma\_elemente\_pe\_linie

}

else

{ numar\_elemente\_minus\_unu = sum(M[,coloana] == -1)

if(numar\_elemente\_minus\_unu == 1)

{ suma\_elemente\_pe\_coloana = sum(M[,coloana]) + 1

M[linie,coloana] = probY[coloana] - suma\_elemente\_pe\_coloana

}

}

}

# verificam daca au mai ramas valori egale cu -1 (necompletate)

contor = any(M == -1)

}

print("Matricea dupa completare:")

print(M)

#creare variabila discreta bidimensionala cu ajutorul functiilor din pachetul discreteRV

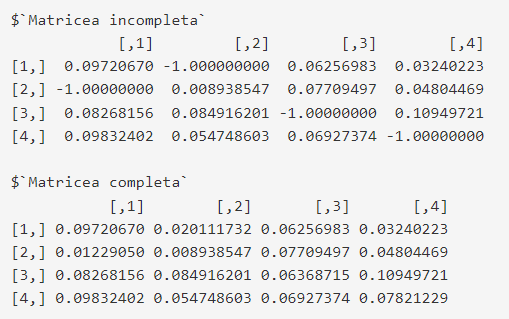
XandY = jointRV(outcomes = list(valuesX, valuesY), probs = c(M))

print("Repartitia comuna a variabilelor aleatoare X si Y")

print(XandY)

return(XandY)

}

****

1. Construiți o funcție **frepmarginal** care construiește repartițiile marginale pentru X și Y pornind de la repartiția lor comună.

**Rezolvare:**

Funcția **frepmarginal** primește ca parametru repartiția comună a două variabile aleatoare discrete X și Y prin intermediul variabilei *rep\_comuna* și returnează o listă care conține repartițiile marginale pentru fiecare dintre acestea (X și Y).

Calculul repartițiilor marginale se realizează cu ajutorul funcției **marginal** inclusă în pachetul **discreteRV**, care, primind ca parametru o repartiție comună a două variabile aleatoare discrete și o valoare, 1 respectiv 2, returnează repartiția marginală pentru prima, respectiv a doua variabilă aleatoare discretă. În plus, funcția **suppressWarnings** este folosită pentru a suprima avertizările care ar putea apărea în timpul calculului repartițiilor marginale, astfel încât afișarea repartițiilor marginale rezultate să fie mai vizibilă.

În cele din urmă, repartiția marginală pentru variabila aleatoare X, respectiv Y, va putea fi afișată prin intermediul apelului *frepmarginal(rep\_comuna)$X*, respectiv *$Y*.

frepmarginal <- function(rep\_comuna) {

X <- suppressWarnings(marginal(rep\_comuna, 1))

Y <- suppressWarnings(marginal(rep\_comuna, 2))

# Afisam rezultatele

print("Repartiția marginală pentru X:")

print(X)

print("Repartiția marginală pentru Y:")

print(Y)

return (list(X = X, Y = Y))

}

#Exemplu de utilizare:

rep\_com\_necompletata<-frepcomgen(3,4)

rep\_com\_necompletata$x

rep\_com\_necompletata$y

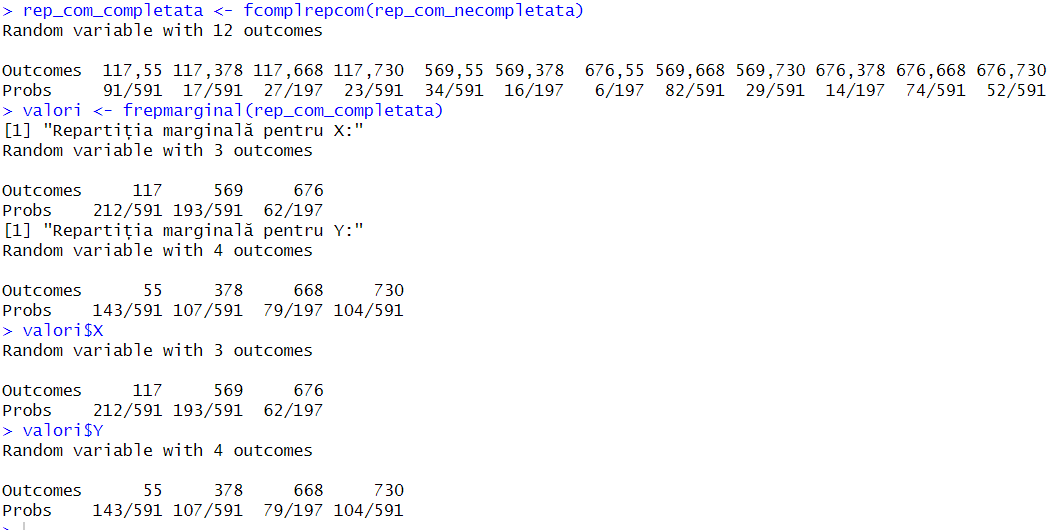
rep\_com\_completata <- fcomplrepcom(rep\_com\_necompletata)

valori <- frepmarginal(rep\_com\_completata)

valori$X

valori$Y

Afișare pentru codul dat:



1. Construiți o funcție **fpropcov** care aplică proprietățile covarianței pentru calculul acesteia pentru v.a. Z=aX+bY și respectiv T=cX+dY considerȃnd că toate informațiile necesare despre X și Y sunt date de intrare.

**Rezolvare:**

Funcția **fpropcov** primește două variabile aleatoare X și Y, împreună cu patru parametri numerici a, b, c și d, care sunt opționali. Scopul funcției este să calculeze covarianța dintre valorile aleatoare Z=aX+bY și T=cX+dY pentru valorile a, b, c, d dacă acestea sunt introduse ca parametri, respectiv covarianța dintre X și Y, în cazul lipsei acestora.

Pașii aplicați în crearea funcției:

1. Se calculează o nouă variabilă Z ca o combinație liniară a variabilelor X și Y, folosind coeficienții a și b.
2. Se calculează o altă variabilă T ca o altă combinație liniară a variabilelor X și Y, folosind coeficienții c și d.
3. Se calculează covarianța dintre Z și T utilizând formula specificată mai sus. Pentru aceasta, se folosește funcția **E()**, prezentă în pachetul **discreteRV**, care returnează media unei valori aleatoare discrete date ca parametru.

Formula folosită pentru calculul covarianței:

fpropcov <- function(X,Y,a = 1,b = 0,c = 0, d = 1)

{

Z<-X\*a+Y\*b

T<-X\*c+Y\*d

cov<-E(Z\*T)-E(Z)\*E(T)

return (cov)

}

fpropcov(valori$X, valori$Y, 2, 1, 1, 1)

1. Construiți o funcție **fPcond** care calculează probabilitatea condiționată pentru v.a. X și Y pornind de la repartiția comună.

**Rezolvare:**

Funcția **fPcond** primește ca parametri o reprezentare comună a două variabile aleatoare (*rep\_comuna*), precum și valoarea dorită pentru Y (*valoare\_Y*) și opțional valoarea dorită pentru X (*valoare\_X*) și are ca scop să calculeze probabilitatea condiționată pentru variabila X dată valoarea Y.

Pașii abordați în crearea funcției:

1. Se apelează funcția **frepmarginal** pentru a obține repartițiile marginale pentru variabilele X și Y din reprezentarea comună;
2. Se extrag repartițiile marginale pentru variabilele X și Y din rezultatul funcției **frepmarginal**
3. Se verifică dacă *valoare\_X* este NULL, adică dacă utilizatorul a furnizat sau nu o valoare pentru variabila X;
4. Dacă *valoare\_X* este NULL, atunci funcția returnează variabila aleatoare determinată de variabila X condiționat la variabila Y=*valoare\_Y* (*X | Y = valoare\_Y*). Aceasta utilizează operatorul | pentru a realiza condiționarea prin intermediul pachetului *discreteRV*;
5. Dacă *valoare\_X* nu este NULL, atunci funcția calculează probabilitatea condiționată a variabilei X dată valoarea X și valoarea Y. Aceasta utilizează, de asemenea, operatorul | pentru a condiționarea specificată;
6. Funcția returnează rezultatul *prob\_cond*, adică probabilitatea condiționată calculată.

În plus, codul scris în comentarii oferă, de asemenea, o soluție validă, în cazul căreia nu este necesară utilizarea pachetului **discreteRV** în calculul probabilității condiționate.

fPcond <- function(rep\_comuna, valoare\_Y, valoare\_X=NULL) {

# Extragem probabilitățile marginale pentru X și Y

rep\_marginale <- frepmarginal(rep\_comuna)

X <- rep\_marginale$X

Y <- rep\_marginale$Y

#Cream un pattern pentru o valoare y data pentru (x,y) din repartitia comuna

#regex\_pattern <- paste0(",", valoare\_Y, "$")

#Cautam pozitiile corespunzatoare valorii Y = valoare\_Y

#pozitii <- grep(regex\_pattern, outcomes(rep\_com\_completata))

#selectam probabilitatile

#probabilitati <- (probs(rep\_com\_completata))[pozitii]

#calculam prob\_cond

#prob\_cond <- RV(outcomes(X), probabilitati)

if(is.null(valoare\_X))

prob\_cond <- X | (Y == valoare\_Y)

else

prob\_cond <- (X == valoare\_X) | (Y == valoare\_Y)

return(prob\_cond)

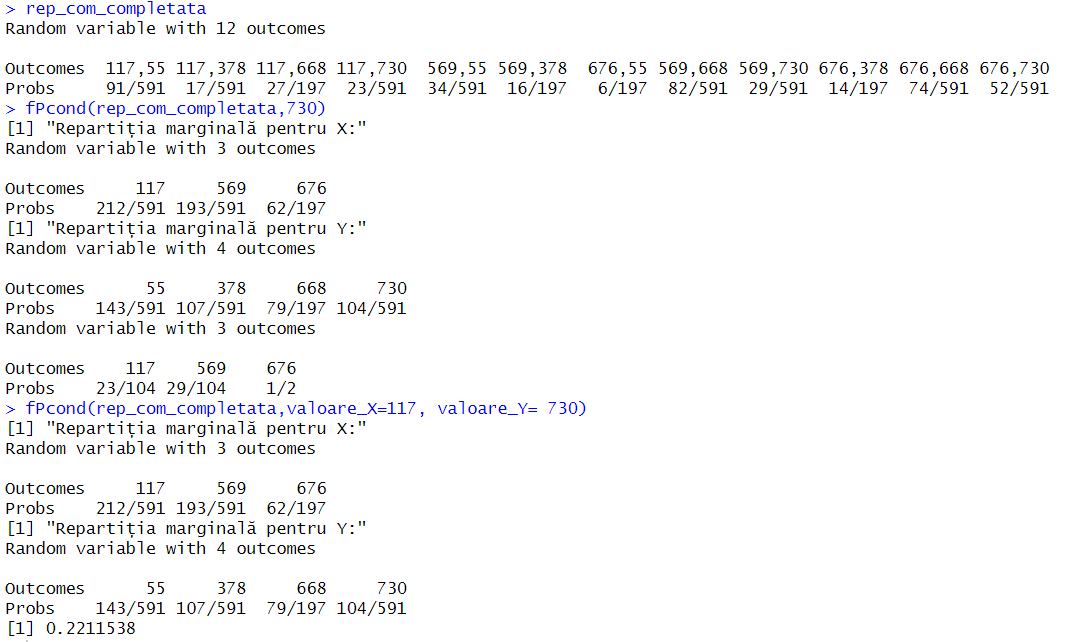
}

#Exemplu utilizare

fPcond(rep\_com\_completata,316)

fPcond(rep\_com\_completata,valoare\_X=367, valoare\_Y= 316)

Exemplu de afișare:



1. Construiți o funcție **fPcomun** care calculează o probabilitate legată de perechea (X,Y) pornind de la repartiția comună.

**Rezolvare:**

Funcția **fPcomun** returnează calculul unei probabilități asociate perechii de variabile aleatoare (X, Y) pe baza repartiției comune a acestora, în funcție de parametrii de intrare dați:

* *X*: Variabila aleatoare discretă X.
* *Y*: Variabila aleatoare discretă Y.
* *ax, bx, ay, by*: Limitele intervalului pentru variabilele X și Y, specificate pentru calculul probabilității asociate perechii (X, Y).
* *eg1, eg2, eg3, eg4*: Parametri booleani care indică dacă se dorește adăugarea unui epsilon mic la limita corespunzătoare a intervalului pentru tratarea inclusivității.
* *just\_eg1, just\_eg2*: Parametri booleani care indică dacă se dorește calculul doar pentru o valoare exactă a lui X sau Y, ignorând intervalele specificate.

Astfel, probabilitatea calculată va fi:

Pașii parcurși în realizarea algoritmului funcției:

1. Se verifică dacă intervalul specificat pentru cel puțin una dintre variabilele aleatoare X sau Y este infinit; dacă da, funcția returnează -1 ca semnalizare a unei erori;
2. Dacă se specifică, se adaugă un epsilon mic la limitele intervalului pentru a trata inclusivitatea;
3. Probabilitatea va fi calculată cu ajutorul funcției **P()** din cadrul pachetului **discreteRV** și cu ajutorul condițiilor specifice pentru a decide modul în care aceasta va fi calculată:
   1. Dacă se dorește doar o valoare exactă pentru atât X, cât și Y, se calculează probabilitatea pentru această pereche exactă;
   2. Dacă se dorește o valoare exactă pentru X, dar un interval pentru Y, se calculează probabilitatea în funcție de această condiție;
   3. Dacă se dorește o valoare exactă pentru Y, dar un interval pentru X, se calculează probabilitatea în funcție de această condiție;
   4. În celelalte cazuri, se calculează probabilitatea pentru perechea (X, Y) în conformitate cu intervalele specificate pentru ambele variabile.
4. Funcția returnează probabilitatea calculată.

fPcomun <- function(X, Y, ax = -Inf, bx = Inf, ay = -Inf, by = Inf, eg1 = FALSE, eg2 = FALSE, eg3 = FALSE, eg4 = FALSE, just\_eg1 = FALSE, just\_eg2 = FALSE) {

# Trebuie să avem măcar o valoare indicată pentru fiecare dintre valorile aleatoare discrete X si Y

if ((ax == -Inf && bx == Inf) || (ay == -Inf && by == Inf))

return (-1)

# Adaugam epsilon pentru a trata inclusivitatea

if (eg1) {

ax <- ax + 1e-10

}

if (eg2) {

bx <- bx + 1e-10

}

if (eg3) {

ay <- ay + 1e-10

}

if (eg4) {

by <- by + 1e-10

}

if (just\_eg1 && just\_eg2){

prob <- P( (X == ax) %AND% (Y == ay) )

} else if (just\_eg1){

if (ax == -Inf){

return (-1)

}

prob <- P( (X == ax) %AND% (Y > ay) %AND% (Y < by) )

} else if (just\_eg2){

if (bx == -Inf){

return (-1)

}

prob <- P( (X > ax) %AND% (X < bx) %AND% (Y == ay) )

} else {

prob <- P( (X > ax) %AND% (X < bx) %AND% (Y > ay) %AND% (Y < by) )

}

return(prob)

}

1. Având la dispoziţie repartiţia comună a v.a. X şi Y de la punctul b) calculaţi:

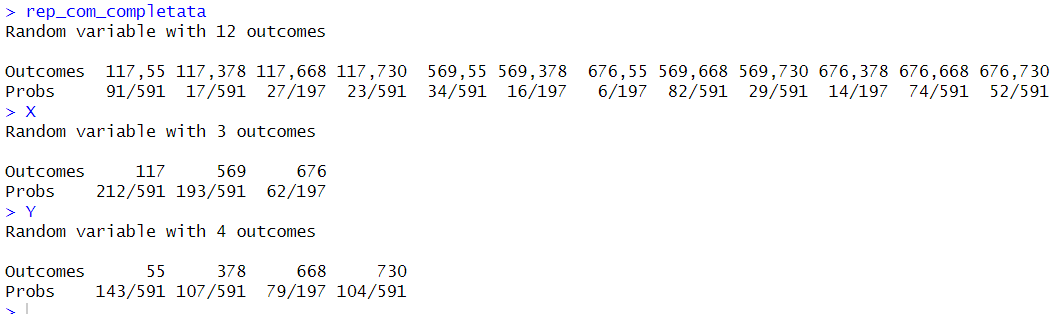
**1) Cov(5X+9,-3Y-2)**

**2) P(0<X0.3)**

**3) P(X>0.2,Y)**

**Rezolvare:**

Pentru calcularea acestor valori, vom considera repartiția comună deja completată în cadrul subpunctelor anterioare cu ajutorul funcției de la punctul b), dar și repartițiile marginale ale variabilelor aleatoare X și Y cu ajutorul funcției de la punctul c).



Astfel, pentru fiecare calcul:

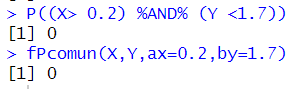
1. : Se utilizează funcția fpropcov de la punctul d) pentru a calcula covarianța dintre variabilele aleatoare Z=5X+9 și T=-3Y-2, fără alți parametri, pentru a specifica faptul că acest calcul trebuie făcut în mod direct între aceste două variabile.



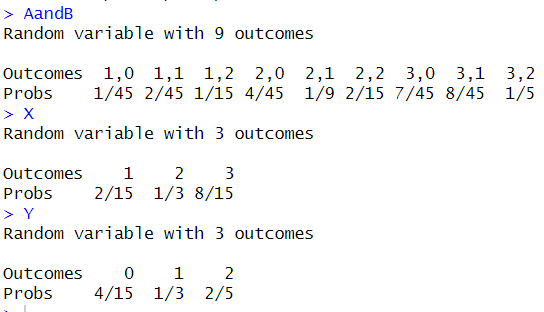
1. : Se utilizează funcția **P()**, din cadrul pachetului **discreteRV**, pentru calcularea probabilității dintre variabila aleatoare X cu valori din intervalul (0, 0.8) și variabila aleatoare Y cu valori din intervalul (0.3, ∞).



1. : Se utilizează funcția **P()**, din cadrul pachetului **discreteRV**, pentru calcularea probabilitatii dintre variabila aleatoare X cu valori din intervalul (0.2, ∞) și variabila aleatoare Y cu valori din intervalul (-∞, 1.7). O altă modalitate, însă, o reprezintă folosirea funcției de la punctul f) după cum urmează:



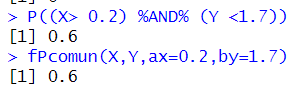
Un exemplu mai dinamic, este, însă, pentru o repartiție comună completă construită de la bun început cu ajutorul funcției **jointRV** din cadrul pachetului **discreteRV**, care are ca valori pentru X 1,2,3, iar pentru Y 0,1,2, cu probabilități distribuite aleatoriu.



În acest caz, rezultatele asociate celor trei calcule cerute vor fi:

****

****

****

# AandB <- jointRV(outcomes = list(1:3, 0:2), probs = 1:9 / sum(1:9))

# X <- marginal(AandB, 1)

# Y <- marginal(AandB, 2)

rep\_marginale <- frepmarginal(rep\_com\_completata)

X <- rep\_marginale$X

Y <- rep\_marginale$Y

#1) Cov(5X+9,-3Y-2)

fpropcov(5\*X+9,-3\*Y-2)

#2) P(0<X<0.8|Y>0.3)

P(((X> 0) %AND% (X<0.8) ) | (Y > 0.3))

#3) P(X>0.2,Y<1.7)

P((X> 0.2) %AND% (Y <1.7))

fPcomun(X,Y,ax=0.2,by=1.7)

1. Pentru exemplul obţinut la punctul b) construiţi două funcţii **fverind** şi respectiv **fvernecor** cu ajutorul cărora să verificaţi dacă variabilele X şi Y sunt: 1) independente 2) necorelate

**Rezolvare:**

Funcția **fverind()** primește ca parametru o variabilă de tip RV, determină repartițiile marginale X și Y și testează dacă acestea sunt independente cu ajutorul funcției **independent()** din pachetul **discreteRV.**

Funcția **fvernecor()** primește ca parametru o variabilă de tip RV, determină repartițiile marginale X și Y și testează dacă acestea sunt independente cu ajutorul funcției **cor()** din pachetul **discreteRV.**

# Test variabile X si Y independente

fverind <- function(XandY)

{

X=marginal(XandY, 1)

Y=marginal(XandY, 2)

T=independent(X,Y)

if (T==0) print("NU sunt independente")

else print("Sunt independente")

}

# Test variabile X si Y necorelate

fvernecor <-function(XandY)

{

X=marginal(XandY, 1)

Y=marginal(XandY, 2)

if(length(X)==length(Y))

{ correlation=cor(X,Y)

print("Coeficientul de corelatie este: ")

print(correlation)

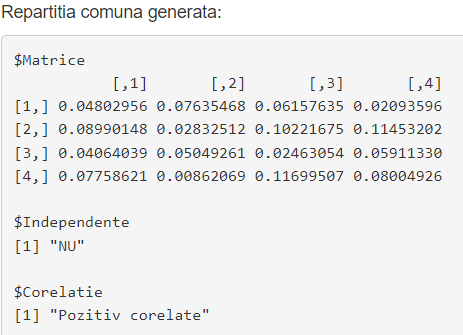
if(correlation==0) print("Necorelate")

else print("Corelate")

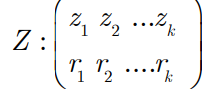
}

else print("X si Y au dimensiuni diferite")

}

****

1. Adăugȃnd ȋncă o v.a.

****

propuneți o manieră vizuală de reprezentare a repartiției comune pentru v.a. X, Y și Z. Care ar fi interpretarea repartițiilor marginale ȋn cazul acestei v.a. tridimensionale și cum ar putea fi obținute?

**Rezolvare**

Putem folosi o reprezentare de forma:

|  |  |  |  |
| --- | --- | --- | --- |
| **X** | **Y** | **Z** | **P(x,y,z)** |
| **x1** | **y1** | **z1** | **P111 = P(x=x1, y=y1, z=z1)** |
| **x1** | **y1** | **z2** | **P112 = P(x=x1, y=y1, z=z2)** |
| **x1** | **y1** | **z3** | **P113 = P(x=x1, y=y1, z=z3)** |
| **...** | **....** | **....** | **......** |
| **xn** | **ym** | **zk** | **Pnmk = P(x=xn, y=ym, z=zk)** |

Repartitiile marginale pot fi calculate astfel:

Repartiția marginală a unei variabile aleatoare discrete tridimensionale reprezintă distribuția de probabilitate a acelei variabile, luată individual, ignorând informația despre celelalte două.

[**https://isl.stanford.edu/~abbas/ee178/lect03-2.pdf**](https://isl.stanford.edu/~abbas/ee178/lect03-2.pdf)

Punctul 2

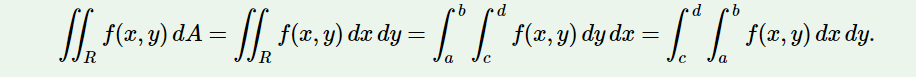
2) Folosind pachetele **R shiny, animate** și orice alte surse de documentare considerați potrivite construiți un proiect R care să permită lucru cu variabile aleatoare continue bidimensionale. Opțiunile din proiect trebuie să implementeze următoarele funcționalități:

1. Verificarea posibilitații de aplicare a teoremei lui Fubini pentru calculul integralei duble dintr-o funcție f , introdusă de utilizator și afișarea unui mesaj corespunzător către utilizator. Calculul propriu-zis al integralei ȋn această manieră, atunci cȃnd este posibil.

**Rezolvare:**

Rezolvarea problemei constă în primirea de la utilizator a unei funcții f(x,y) și capetele intervalelor lui x (*x1, x2*), respectiv y (*y1, y2*); și în verifică dacă, pentru o astfel de funcție, se poate aplica teorema lui Fubini și totodată se vor afișa valorile celor două integrale duble.

Soluția propusă are la bază **Teorema lui Fubini** reprezentată prin egalitatea:

****

library(shiny)

# UI

ui\_Fubini <- fluidPage(

titlePanel("Verificarea Teoremei lui Fubini"),

sidebarLayout(

sidebarPanel(

textInput("function", "Introduceți funcția f(x, y):", ""),

numericInput("x1", "x1:", value = 0),

numericInput("x2", "x2:", value = 1),

numericInput("y1", "y1:", value = 0),

numericInput("y2", "y2:", value = 1),

actionButton("check\_button", "Verifică aplicabilitatea teoremei lui Fubini"),

textOutput("result\_text")

),

mainPanel(

textOutput("result\_x"),

textOutput("result\_y")

)

)

)

# Server

server\_Fubini <- function(input, output) {

observeEvent(input$check\_button, {

# Definirea funcției

f <- function(x, y) {

eval(parse(text = input$'function'), list(x = x, y = y))

}

x1 <- input$x1

x2 <- input$x2

y1 <- input$y1

y2 <- input$y2

# Verificăm aplicabilitatea teoremei lui Fubini

integral\_xy <- integrate(Vectorize(function(y) integrate(Vectorize(function(x) f(x, y)), x1, x2)$value), y1, y2)$value

integral\_yx <- integrate(Vectorize(function(x) integrate(Vectorize(function(y) f(x, y)), y1, y2)$value), x1, x2)$value

if (abs(integral\_xy - integral\_yx) < 1e-10) {

result <- "Teorema lui Fubini este aplicabilă. Integralele duble sunt egale."

} else {

result <- "Teorema lui Fubini nu este aplicabilă. Integralele duble nu sunt egale."

}

output$result\_text <- renderText({

result

})

output$result\_x <- renderText({

paste("Rezultatul integralei duble în x:", format(integral\_yx, digits = 4))

})

output$result\_y <- renderText({

paste("Rezultatul integralei duble în y:", format(integral\_xy, digits = 4))

})

})

}

# Rulează aplicația

shinyApp(ui = ui\_Fubini, server = server\_Fubini)

1. Interpretarea geometrică a integralei duble.

**Rezolvare:**

Codul elaborat pentru rezolvarea cerinței este o aplicație Shiny care permite utilizatorului să introducă o funcție bidimensională în funcție de x și y ( *f(x, y)* ) și valori care să reprezinte capetele intervalelor asociate lui x și a lui y, valori opționale care, în cazul în care nu sunt completate, se vor considera Inf/-Inf (valoare aproximată).

Această aplicație permite utilizatorului să vizualizeze suprafața ocupată de integrala dublă a acestei funcții folosind biblioteca **plotly** într-un mediu interactiv, bazându-ne totodată pe formula:

Acest lucru se realizează prin intermediul funcției **plot\_ly**, care este utilizată pentru un obiect care va fi utilizat pentru a reprezenta graficul 3D al integralei duble a funcției introduse de utilizator, primind ca parametri:

1. x, y și z ale graficului. În cazul nostru:

* x = *x\_vals*: Reprezintă valorile pe axa x, generate anterior în funcția seq pentru intervalul specificat.
* y = *y\_vals*: Reprezintă valorile pe axa y, generate similar cu valorile pe axa x.
* z = *z\_vals*: Reprezintă valorile pe axa z, care sunt calculate anterior în bucla for pentru fiecare pereche de valori (x,y).

1. %>% (pipe operator): Acest operator este utilizat pentru a transmite rezultatul unei expresii către următoarea expresie ca prim argument. În acest caz, rezultatul obținut din funcția plot\_ly este transmis mai departe pentru a fi configurat prin funcția layout.
2. *layout*: Această funcție este utilizată pentru a configura aspectul și comportamentul graficului plotly. În cadrul acestei funcții, specificăm diverse proprietăți ale graficului. În cazul nostru:
3. *scene* = list(aspectmode = "cube"): Acest lucru specifică modul de aspect al scenei. În acest caz, am specificat "cube", ceea ce înseamnă că un cub se va folosi pentru a reprezenta suprafața 3D, astfel încât toate cele trei dimensiuni să aibă aceeași scală, ceea ce ajută la o mai bună percepție a formei și a proporțiilor graficului.

library(shiny)

library(plotly)

ui\_Geometric <- fluidPage(

titlePanel("Interpretarea Geometrica a Integralei Duble"),

sidebarLayout(

sidebarPanel(

textInput("functie", "Introduceti functia f(x, y):", ""),

numericInput("ax", "Valoare pentru ax (opțional):", value = NA),

numericInput("bx", "Valoare pentru bx (opțional):", value = NA),

numericInput("ay", "Valoare pentru ay (opțional):", value = NA),

numericInput("by", "Valoare pentru by (opțional):", value = NA),

actionButton("deseneaza", "Deseneaza")

),

mainPanel(

plotlyOutput("plot")

)

)

)

server\_Geometric <- function(input, output) {

observeEvent(input$deseneaza, {

tryCatch({

expr <- parse(text = paste("f <- function(x, y) {", input$functie, "}"))

eval(expr)

# Verifica validitatea functiei

if (!is.function(f)) {

stop("Introduceti o functie valida.")

}

# Definește limitele pentru ax, bx, ay, by sau folosește valori implicite

ax <- ifelse(is.na(input$ax), -1e6, input$ax)

bx <- ifelse(is.na(input$bx), 1e6, input$bx)

ay <- ifelse(is.na(input$ay), -1e6, input$ay)

by <- ifelse(is.na(input$by), 1e6, input$by)

# Genereaza date pentru grafic

x\_vals <- seq(ax, bx, length.out = 100)

y\_vals <- seq(ay, by, length.out = 100)

z\_vals <- outer(x\_vals, y\_vals, Vectorize(f))

# Deseneaza graficul interactv

p <- plot\_ly(x = x\_vals, y = y\_vals, z = z\_vals, type = "surface") %>%

layout(scene = list(aspectmode = "cube"))

output$plot <- renderPlotly({

p

})

}, error = function(e) {

print(e)

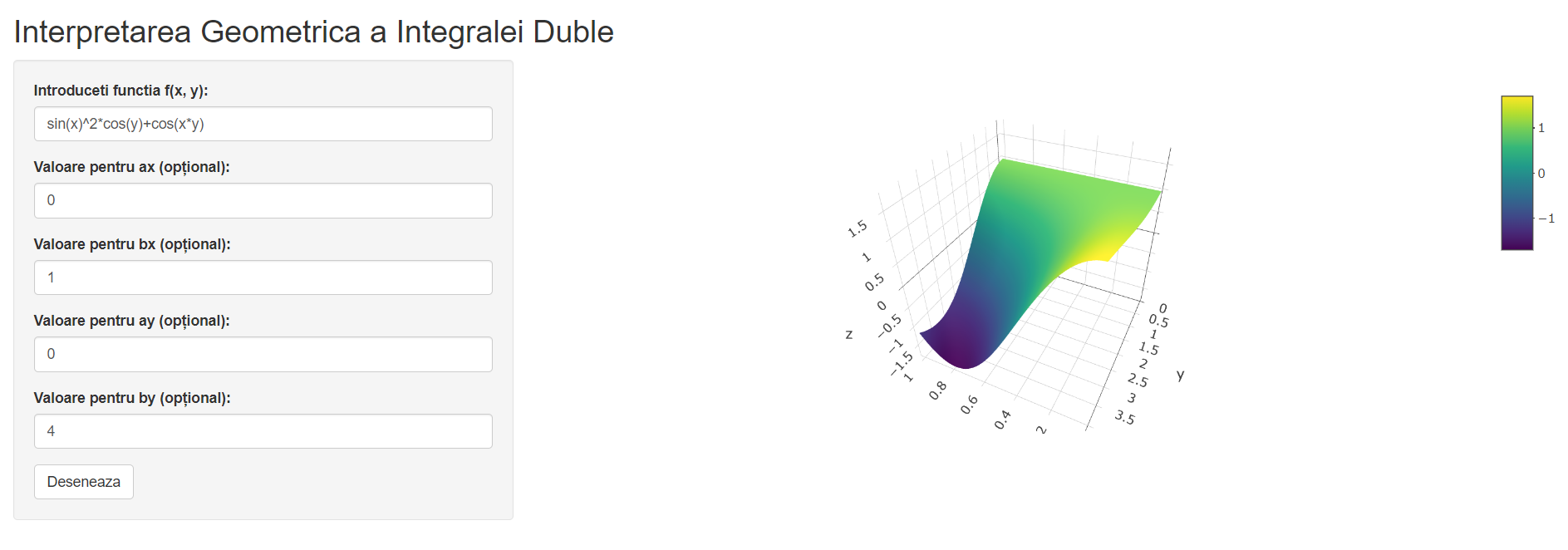
})

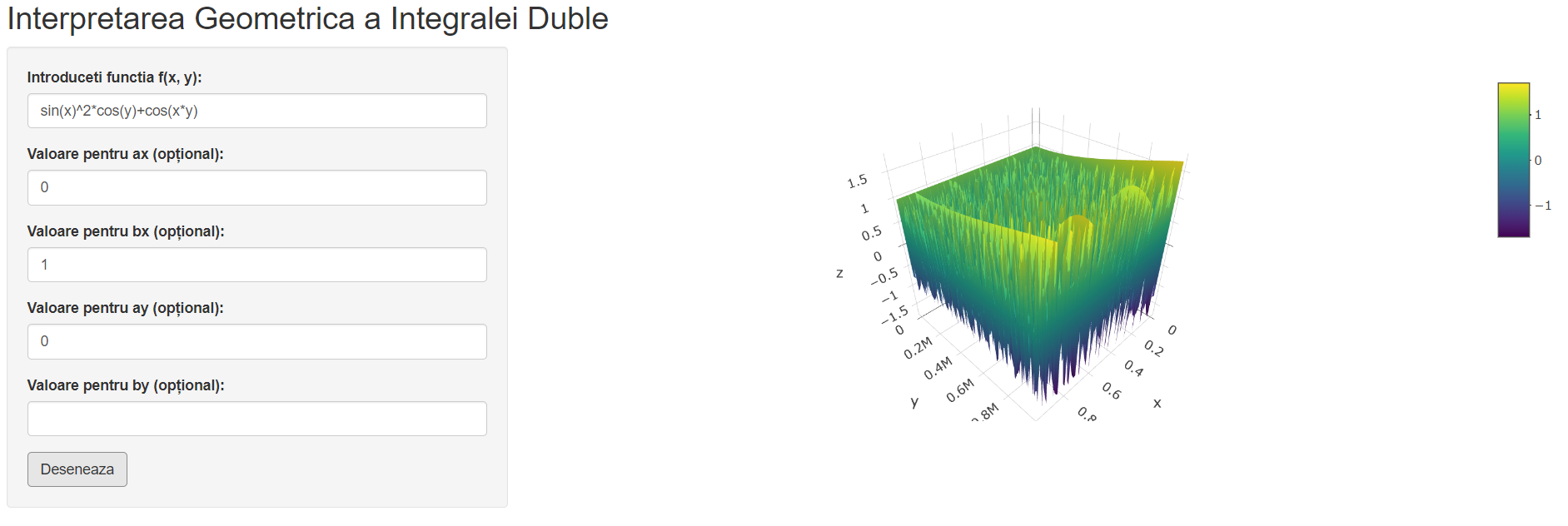
})

}

shinyApp(ui = ui\_Geometric, server = server\_Geometric)

Exemple de reprezentare:





1. Verificarea dacă o funcție cu două variabile f(x,y), introdusă de utilizator este densitate de probabilitate.

**Rezolvare:**

Am verificat dacă:

Pemtru a verifica dacă funcția este pozitivă am determinat valorile acesteia pe intervalul [x1,x2] x [y1,y2] iar pentru calculul integralei duble am utilizat funcția **integrate2** din pachetul **pracma**.

# test densitate de probabilitate pentru o functie cu 2 parametri (x,y)

# - trebuie sa verificam sa fie pozitiva si probabilitatea totala sa fie 1

# Exemple pentru teste

# f(x,y)=4\*x\*y, 0<X<1, 0<y<1

# 0 in rest

# f(x,y)= 1/5\*(x+y+1), 0<=X<=1, 0<=y<=2

# 0 in rest

library(shiny)

library(pracma)

ui <- fluidPage(

titlePanel("Test densitate de probabilitate"),

sidebarLayout(

sidebarPanel(

textInput("functie", "Introduceti functia f(x, y):", "1/5\*(x+y+1)"),

numericInput("x1\_val", "Valoarea pentru x1:", 0),

numericInput("x2\_val", "Valoarea pentru x2:", 1),

numericInput("y1\_val", "Valoarea pentru y1:", 0),

numericInput("y2\_val", "Valoarea pentru y2:", 2),

actionButton("calculeaza", "Calculeaza")

),

mainPanel(

h3("Rezultat:"),

verbatimTextOutput("rezultat")

)

)

)

server <- function(input, output) {

observeEvent(input$calculeaza, {

tryCatch({

expr <- parse(text = paste("f <- function(x, y) {", input$functie, "}"))

eval(expr)

x1 <- input$x1\_val

x2 <- input$x2\_val

y1 <- input$y1\_val

y2 <- input$y2\_val

# test valori pozitive pe domeniul [x1,x2] x [y1,y2]

ok <- 1

for(i in seq(x1,x2, 0.01))

for(j in seq (y1,y2, 0.01))

if( f(i,j) < 0) ok<-0

#test probabilitatea totala egala cu 1

integral <- integral2(Vectorize(f), x1, x2, y1, y2)$Q

if (abs(integral-1) > 0.1) ok <- 0

if (ok==1) rezultat <- "Functia este densitate de probabilitate"

else rezultat <- "Functia nu este densitate de probabilitate"

output$rezultat <- renderPrint({

rezultat

})

}, error = function(e) {

output$rezultat <- renderPrint({

as.character(e)

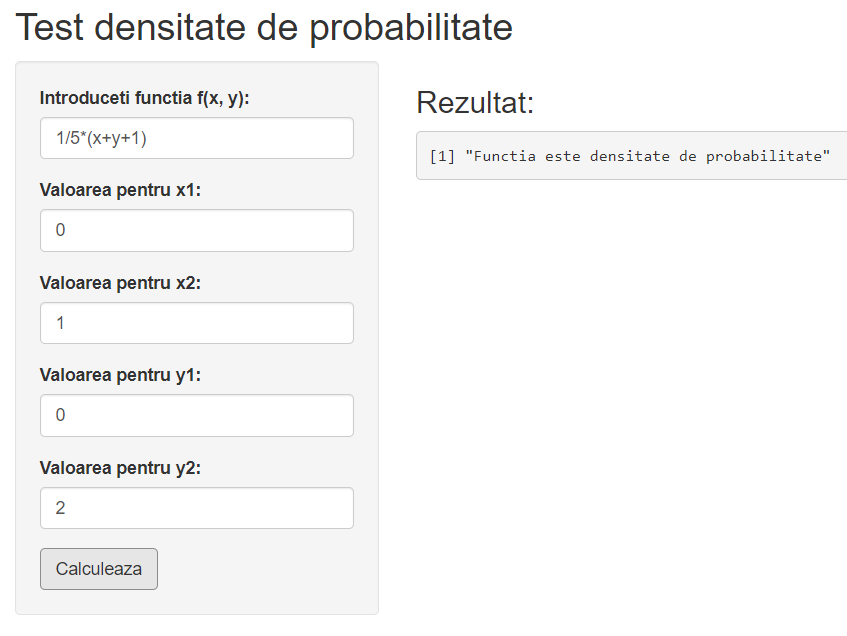
})

})

})

}

shinyApp(ui = ui, server = server)

****

1. 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.

**Rezolvare:**

Am definit clasa **ContinuousRV** pentru lucrul cu obiecte de tip variabile aleatoare continue unidimensionale. Constructorul va primi ca parametri densitatea de probabilitate, intervalul [x1,x2] iar la crearea obiectului se vor calcula, pornind de la acestea, media și dispersia.

# -------------------------------Punctul d pentru unidimensionale -------------------------------------------------

# crearea unei v.a. continue unidimensionale pentru care se introduce densitatea de repartitie

# utilizatorul va introduce densitatea de repartitie si intervalul [x1,x2]

# Exemplu din materialul de la Variabile aleatoare continue

# f(x)=3/8\*(4\*x-2\*x\*x) 0<x<2

# 0 altfel

# E(X)=1, Var(X)=1/5

library(shiny)

library(pracma)

ui <- fluidPage(

titlePanel("Creare variabila aleatoare continua unidimensionala"),

sidebarLayout(

sidebarPanel(

textInput("functie", "Introduceti densitatea de repartitie f(x):", "3/8\*(4\*x-2\*x\*x)"),

numericInput("x1\_val", "Valoarea pentru x1:", 0),

numericInput("x2\_val", "Valoarea pentru x2:", 2),

actionButton("calculeaza", "Genereaza obiect")

),

mainPanel(

h5("A fost creat un obiect de tip v.a. continua unidimensionala cu datele membru:"),

verbatimTextOutput("rezultat"),

)

)

)

server <- function(input, output) {

observeEvent(input$calculeaza, {

tryCatch({

expr <- parse(text = paste("f <- function(x) {", input$functie, "}"))

eval(expr)

x1 <- input$x1\_val

x2 <- input$x2\_val

setClass("ContinuousRV",

slots = c(

pdf = "function", # densitatea de probabilitate

x1 = "numeric", # capatul inferior al domeniului de definiție

x2 = "numeric", # capatul superior al domeniului de definiție

mean = "numeric", # media

variance = "numeric" # dispersia

))

# Constructor pentru clasa

ContinuousRV <- function(pdf, x1, x2) {

mean\_value <- integrate(function(x) x \* pdf(x), lower = x1, upper = x2)$value

variance\_value <- integrate(function(x) (x - mean\_value)^2 \* pdf(x), lower = x1, upper = x2)$value

new("ContinuousRV", pdf = pdf, x1 = x1, x2 = x2, mean = mean\_value, variance = variance\_value)

}

X <-ContinuousRV(f,x1,x2)

output$rezultat <- renderPrint({

rezultat\_list <- list("PDF"=X@pdf, "X1"=X@x1, "X2"=X@x2, "Media" = X@mean, "Dispersia" = X@variance)

names(rezultat\_list) <- c("PDF", "X1", "X2", "Media", "Dispersia")

rezultat\_list

})

}, error = function(e) {

output$rezultat <- renderPrint({

as.character(e)

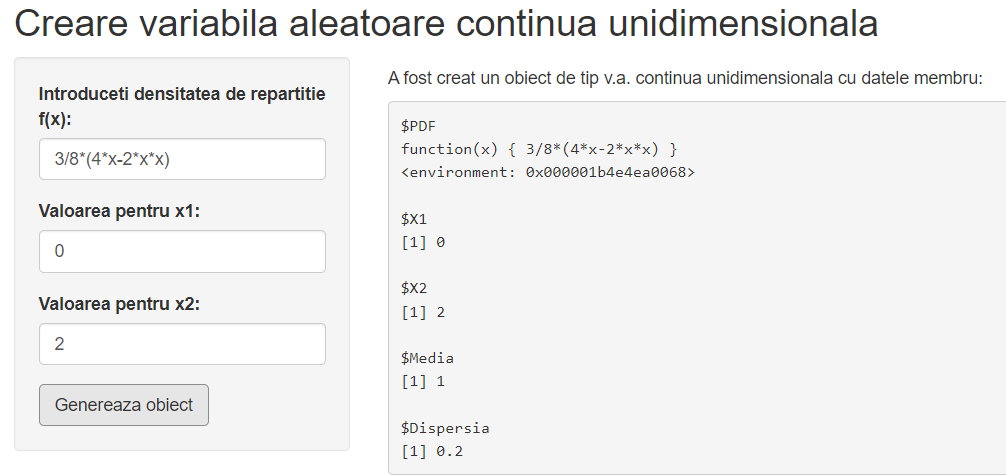
})

})

})

}

shinyApp(ui = ui, server = server)



Pentru variabialele aleatoare continue bidimensionaleam definit clasa **ContinuousRV2**, pe aceeași idee, dar am adăugat date membru noi: media pentru XY, X și Y, dispersia pentru X, Y, covarianta și corelația pentru XY.

# -------------------------------Punctul d pentru bidimensionale -------------------------------------------------

# crearea unei v.a. continue bidimensionale pentru care se introduce densitatea de repartitie

# utilizatorul va introduce densitatea de repartitie si intervalele [x1,x2], [y1,y2]

# Exemplu pentru teste

# f(x,y)=1/5\*(x+y+1) x din [0,1], y din [0,2]

# 0 in rest

# E(x)=8/15=0.53

# E(y)=17/15=1.13

# Var(x)= 37/450=0.08

# Var(y)=71/225=0.31

# C(x,y)=-1/225=-0.04

# Cor(x,y)=-0.02

library(shiny)

library(pracma)

ui <- fluidPage(

titlePanel("Creare variabila aleatoare continua unidimensionala"),

sidebarLayout(

sidebarPanel(

textInput("functie", "Introduceti densitatea de repartitie f(x):", "1/5\*(x+y+1)"),

numericInput("x1\_val", "Valoarea pentru x1:", 0),

numericInput("x2\_val", "Valoarea pentru x2:", 1),

numericInput("y1\_val", "Valoarea pentru y1:", 0),

numericInput("y2\_val", "Valoarea pentru y2:", 2),

actionButton("calculeaza", "Genereaza obiect")

),

mainPanel(

h5("A fost creat un obiect de tip v.a. continua bidimensionala cu datele membru:"),

verbatimTextOutput("rezultat")

)

)

)

server <- function(input, output) {

observeEvent(input$calculeaza, {

tryCatch({

expr <- parse(text = paste("f <- function(x,y) {", input$functie, "}"))

eval(expr)

x1 <- input$x1\_val

x2 <- input$x2\_val

y1 <- input$y1\_val

y2 <- input$y2\_val

setClass("ContinuousRV2",

slots = c(

pdf = "function", # densitatea de probabilitate

x1 = "numeric", # capatul inferior pentru X

x2 = "numeric", # capatul superior pentru X

y1 = "numeric", # capatul inferior pentru Y

y2 = "numeric", # capatul superior pentru Y

mean = "numeric", #media pentru XY

mean\_x = "numeric", # media pentru X

mean\_y = "numeric", # media pentru Y

variance\_x = "numeric", # dispersia X

variance\_y = "numeric", # dispersia Y

covariance = "numeric", # covarianta

correlation = "numeric" # coeficientul de corelatie

))

# Constructor pentru clasa

ContinuousRV2 <- function(pdf, x1, x2, y1, y2) {

# Repartiția marginală a lui X

f\_X <- function(x) { integrate(function(y) f(x, y), y1, y2)$value}

# Repartiția marginală a lui Y

f\_Y <- function(y) { integrate(function(x) f(x, y), x1, x2)$value}

# Media

mX <- integrate (Vectorize(function (x) {x \* f\_X(x)}), x1, x2)$value

mY <- integrate (Vectorize(function (y) {y \* f\_Y(y)}), y1, y2)$value

mXY <- integral2(Vectorize(function (x ,y ) {(x)\*(y)\*f(x ,y)}),x1,x2,y1,y2)$Q

# Dispersia

varX <- integrate (Vectorize(function (x) {(x-mX)\*(x-mX)\*f\_X(x)}),x1,x2)$value

varY <- integrate (Vectorize(function (y) {(y-mY)\*(y-mY)\*f\_Y(y)}),y1,y2)$value

# Covarianta

cov\_XY <- integral2(Vectorize(function (x ,y ) {(x-mX)\*(y-mY)\*f(x ,y)}),x1,x2,y1,y2)$Q

#Coeficientul de corelatie

cor\_XY <- cov\_XY/ (sqrt(varX)\*sqrt(varY))

new("ContinuousRV2", pdf = pdf, x1 = x1, x2 = x2, y1 = y1, y2 = y2,

mean = mXY, mean\_x = mX, mean\_y = mY,

variance\_x = varX, variance\_y = varY,

covariance = cov\_XY, correlation = cor\_XY)

}

X <-ContinuousRV2(f,x1,x2,y1,y2)

output$rezultat <- renderPrint({

rezultat\_list <- list("PDF"=X@pdf, "X1"=X@x1, "X2"=X@x2, "Y1"=X@y1, "Y2"=X@y2,

"E(X,Y)" = X@mean, "E(X)" = X@mean\_x, "E(Y)" = X@mean\_y,

"Var(X)" = X@variance\_x, "Var(Y)" = X@variance\_y,

"Covarianta" = X@covariance, "Corelatia" = X@correlation)

names(rezultat\_list) <- c("PDF", "X1", "X2", "Y1", "Y2", "E(X,Y)", "E(X)", "E(Y)",

"Var(X)", "Var(Y)", "Covarianta", "Corelatia")

rezultat\_list

})

}, error = function(e) {

output$rezultat <- renderPrint({

as.character(e)

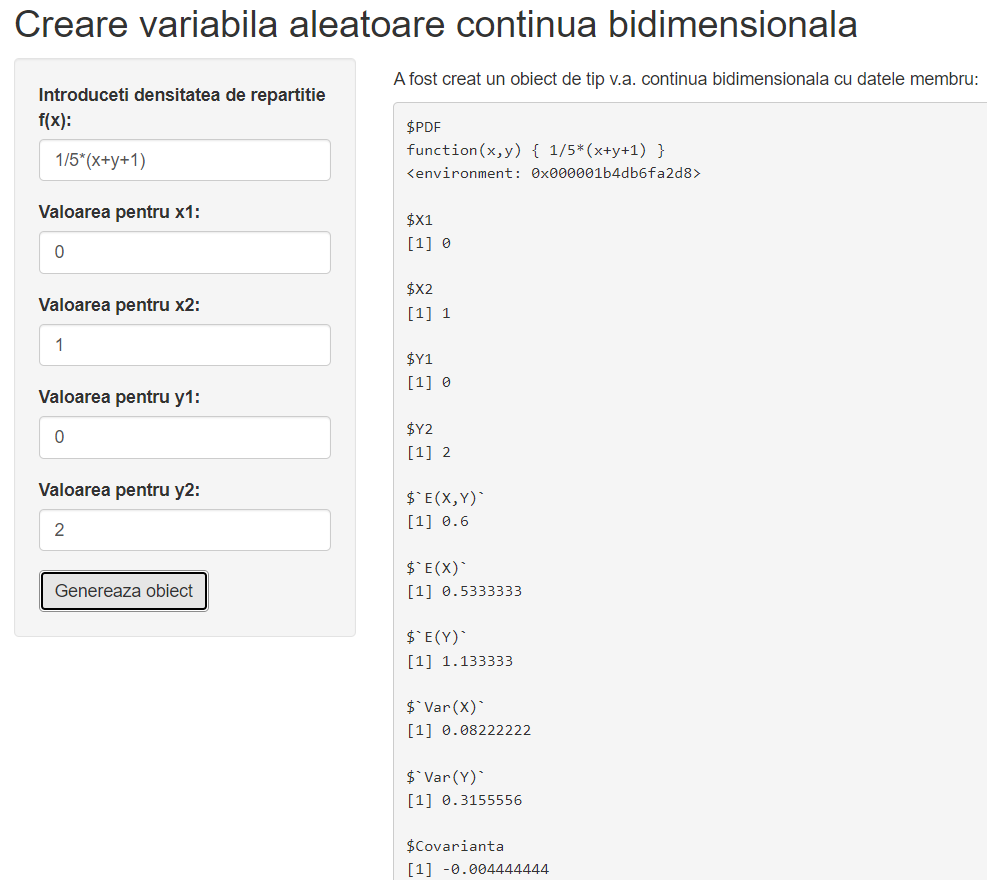
})

})

})

}

shinyApp(ui = ui, server = server)



1. Construirea densităților marginale și a celor condiționale pornind de la densitatea comună f(x,y) a două v.a. unidimensionale X și Y.

**Rezolvare:**

Am construit densitățile marginale și condiționale conform formulelor:

Am calculat valorile acestor funcții pentru o serie de valori pentru x și y din intervalele [x1,x2], [y1,y2].

library(shiny)

library(pracma)

ui <- fluidPage(

titlePanel("Construirea densitatilor marginale si a celor conditionale"),

sidebarLayout(

sidebarPanel(

textInput("functie", "Introduceti functia f(x, y):", "(1/(2\*3.14159265359))\*exp(-(x\*x+y\*y)/2)"),

numericInput("x1\_val", "Valoarea pentru x1:", -1),

numericInput("x2\_val", "Valoarea pentru x2:", 1),

numericInput("y1\_val", "Valoarea pentru y1:", -2),

numericInput("y2\_val", "Valoarea pentru y2:", 3),

numericInput("x\_val", "Densitatea conditionala pentru x:", 1),

numericInput("y\_val", "Densitatea conditionala pentru y:", 1),

actionButton("calculeaza", "Calculeaza")

),

mainPanel(

h3("Densitatile marginale:"),

verbatimTextOutput("rezultat\_marginale"),

h3("Densitatile conditionate:"),

verbatimTextOutput("rezultat\_conditionale")

)

)

)

server <- function(input, output) {

observeEvent(input$calculeaza, {

tryCatch({

expr <- parse(text = paste("f <- function(x, y) {", input$functie, "}"))

eval(expr)

x1 <- input$x1\_val

x2 <- input$x2\_val

y1 <- input$y1\_val

y2 <- input$y2\_val

xc <- input$x\_val

yc <- input$y\_val

# Repartiția marginală a lui X

f\_X <- function(x) {

integrate(function(y) f(x, y), y1, y2)$value

}

# Repartiția marginală a lui Y

f\_Y <- function(y) {

integrate(function(x) f(x, y), x1, x2)$value

}

# Repartiția condiționată a lui X dat fiind y

f\_X\_Y <- function(x, y) {

f(x, y) / f\_Y(y)

}

# Repartiția condiționată a lui Y dat fiind x

f\_Y\_X <- function(y, x) {

f(x, y) / f\_X(x)

}

# Testare functii

x\_values <- seq(x1, x2, by = 0.5) # serie de valori pentru x

y\_values <- seq(y1, y2, by = 0.5) # serie de valori pentru y

# Repartiția marginală a lui X pentru fiecare valoare de x

marginal\_X <- sapply(x\_values, f\_X)

# Repartiția marginală a lui Y pentru fiecare valoare de y

marginal\_Y <- sapply(y\_values, f\_Y)

# Repartia conditionata a lui X pentru y=1

cond\_x\_by\_y <- f\_X\_Y(x\_values, yc)

# Repartia conditionata a lui Y pentru x=1

cond\_y\_by\_x <- f\_Y\_X(y\_values, xc)

output$rezultat\_marginale <- renderPrint({

rezultat\_list <- list("Marginal\_X" = marginal\_X, "Marginal\_Y" = marginal\_Y)

names(rezultat\_list) <- c("Marginal\_X", "Marginal\_Y")

rezultat\_list

})

output$rezultat\_conditionale<- renderPrint({

rezultat\_list <- list("Conditional\_X" = cond\_x\_by\_y, "Conditional\_Y" = cond\_y\_by\_x)

names(rezultat\_list) <- c("Conditional\_X", "Conditional\_Y")

rezultat\_list

})

}, error = function(e) {

output$rezultat <- renderPrint({

as.character(e)

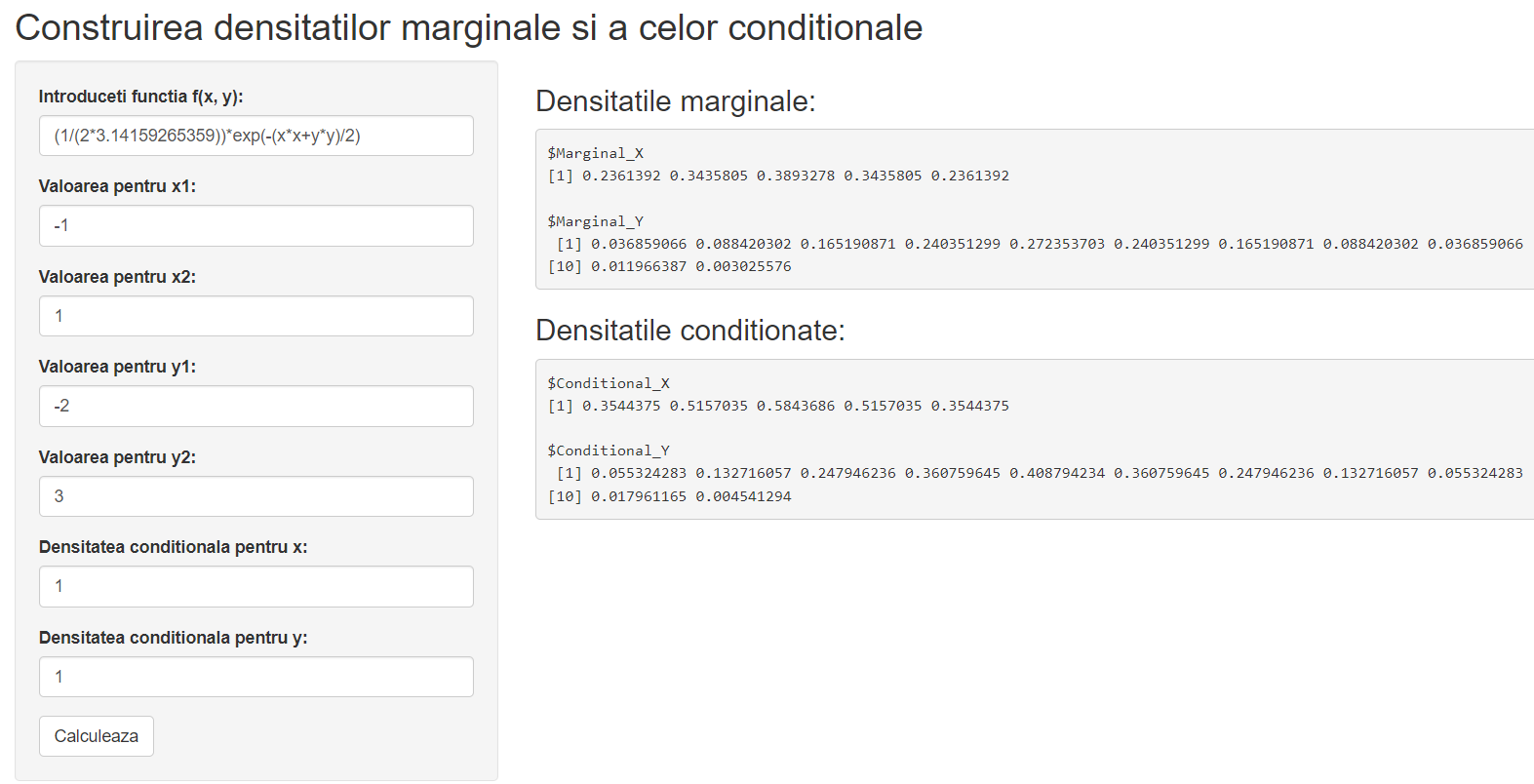
})

})

})

}

shinyApp(ui = ui, server = server)



1. Reprezentarea grafică a densității și a funcției de repartiție a unei v.a. unidimensionale /bidimensionale pentru diferite valori ale parametrilor repartiției. Ȋn cazul ȋn care funcția de repartiție nu este dată ȋntr-o formă explicită(ex. repartiția normală) se acceptă reprezentarea grafică a unei aproximări a acesteia. Se obține punctaj suplimentar dacă se realizează o animație care să pună ȋn valoare modificarea funcției reprezentate la schimbarea parametrilor repartiției.

**Rezolvare:**

Aplicația construită primește de la utilizator informații precum: tipul de variabilă aleatoare continuă pe care vrea să o vizualizeze, densitatea acesteia reprezentată prin funcția f(x) pentru cazul v.a unidimensionale, respectiv prin funcția f(x,y) pentru cazul v.a. bidimensionale, dar și capetele intervalelor lui x, respectiv y, pentru care are loc densitatea dată.

Funcția va afișa reprezentarea grafică a funcției densitate și a funcției de repartiție cu ajutorul funcțiilor **renderPlot** și **ggplot** din pachetul **ggplot2**.

Pentru calculul funcției de repartiție în cazul v.a .unidimensionale:

1. Aproximarea integralei pentru funcția de repartiție = F(x):

Folosim o metodă numerică de aproximare a integralei pentru a calcula F(x). În cazul codului dat, folosim o variantă discretă a metodei trapezoidale.

Iterăm prin fiecare pereche de valori consecutive (x i-1, x i) și aproximăm integrala între aceste două puncte utilizând formula trapezoidală , unde h este diferența dintre două puncte consecutive:



1. Normalizarea funcției de repartiție:

După calculul F(x), normalizăm valorile astfel încât să fie între 0 și 1. Acest lucru se face prin împărțirea tuturor valorilor F(x) la valoarea maximă a F(x).

Pentru calculul funcției de repartiție în cazul v.a .bidimensionale:

1. Aproximarea integralei pentru funcția de repartiție F(x,y):

Iterăm prin fiecare pereche de valori consecutive (x i-1, x i) și (y i-1, y i) și aproximăm integrala bidimensională între aceste două puncte utilizând o formă discretizată a integralei duble.

1. Normalizarea funcției de repartiție:

După calculul F(x,y), normalizăm valorile astfel încât să fie între 0 și 1.

library(shiny)

library(ggplot2)

# UI

ui\_Grafice <- fluidPage(

titlePanel("Reprezentarea grafică a densității și a funcției de repartiție"),

sidebarLayout(

sidebarPanel(

radioButtons("var\_type", "Tip variabilă:", c("Unidimensională", "Bidimensională")),

textInput("density\_function", "Funcția densității (f(x)):", ""),

numericInput("x1", "Limita inferioară a intervalului x:", value = 0),

numericInput("x2", "Limita superioară a intervalului x:", value = 1),

conditionalPanel(

condition = "input.var\_type == 'Bidimensională'",

numericInput("y1", "Limita inferioară a intervalului y:", value = 0),

numericInput("y2", "Limita superioară a intervalului y:", value = 1)

),

actionButton("plot\_button", "Generează graficul")

),

mainPanel(

plotOutput("density\_plot"),

plotOutput("cdf\_plot")

)

)

)

# Server

server\_Grafice <- function(input, output) {

observeEvent(input$plot\_button, {

if (input$var\_type == "Unidimensională") {

x <- seq(input$x1, input$x2, length.out = 100)

density <- eval(parse(text = input$density\_function))

# Calculăm funcția de repartiție unidimensională

cdf <- rep(0, length(x))

for (i in 2:length(x)) {

cdf[i] <- cdf[i - 1] + (x[i] - x[i - 1]) \* (density[i] + density[i - 1]) / 2

}

cdf <- cdf / max(cdf) # Normalizăm pentru a obține o funcție de repartiție între 0 și 1

data <- data.frame(x, density, cdf)

output$density\_plot <- renderPlot({

ggplot(data, aes(x, density)) +

geom\_line() +

labs(title = "Densitatea")

})

output$cdf\_plot <- renderPlot({

ggplot(data, aes(x, cdf)) +

geom\_line() +

labs(title = "Funcția de repartiție")

})

} else {

x <- seq(input$x1, input$x2, length.out = 100)

y <- seq(input$y1, input$y2, length.out = 100)

z <- outer(x, y, function(x, y) eval(parse(text = input$density\_function)))

df <- data.frame(expand.grid(x = x, y = y), z = as.vector(z))

# Calculăm funcția de repartiție bidimensională

cdf <- matrix(0, nrow = length(x), ncol = length(y))

for (i in 2:length(x)) {

for (j in 2:length(y)) {

cdf[i, j] <- cdf[i - 1, j] + cdf[i, j - 1] - cdf[i - 1, j - 1] +

(x[i] - x[i - 1]) \* (y[j] - y[j - 1]) \* (z[i, j] + z[i - 1, j] + z[i, j - 1] + z[i - 1, j - 1]) / 4

}

}

cdf <- cdf / max(cdf) # Normalizăm pentru a obține o funcție de repartiție între 0 și 1

output$density\_plot <- renderPlot({

ggplot(df, aes(x, y, z = z)) +

geom\_contour() +

labs(title = "Densitatea")

})

output$cdf\_plot <- renderPlot({

contour(x, y, cdf,

xlab = "x",

ylab = "y",

main = "Funcția de repartiție")

})

}

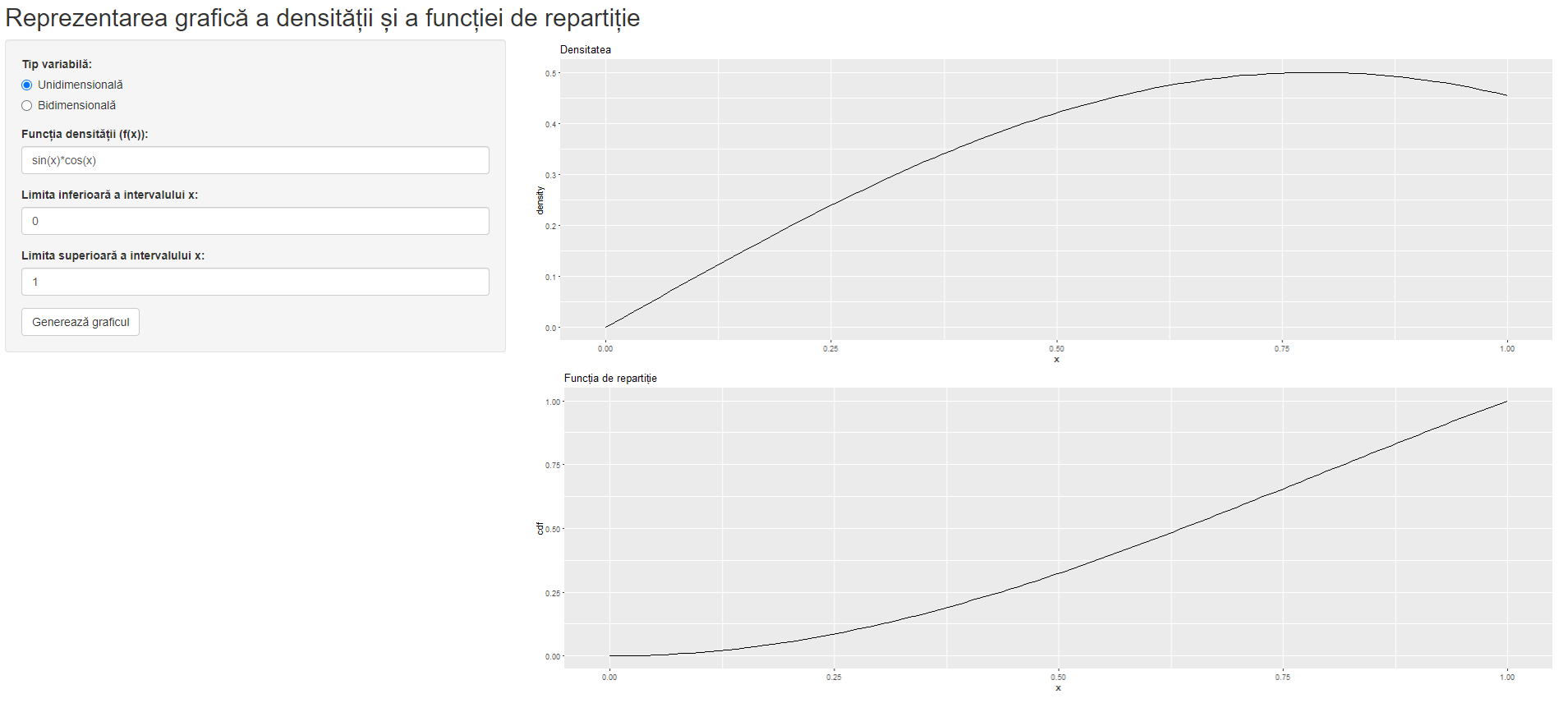
})

}

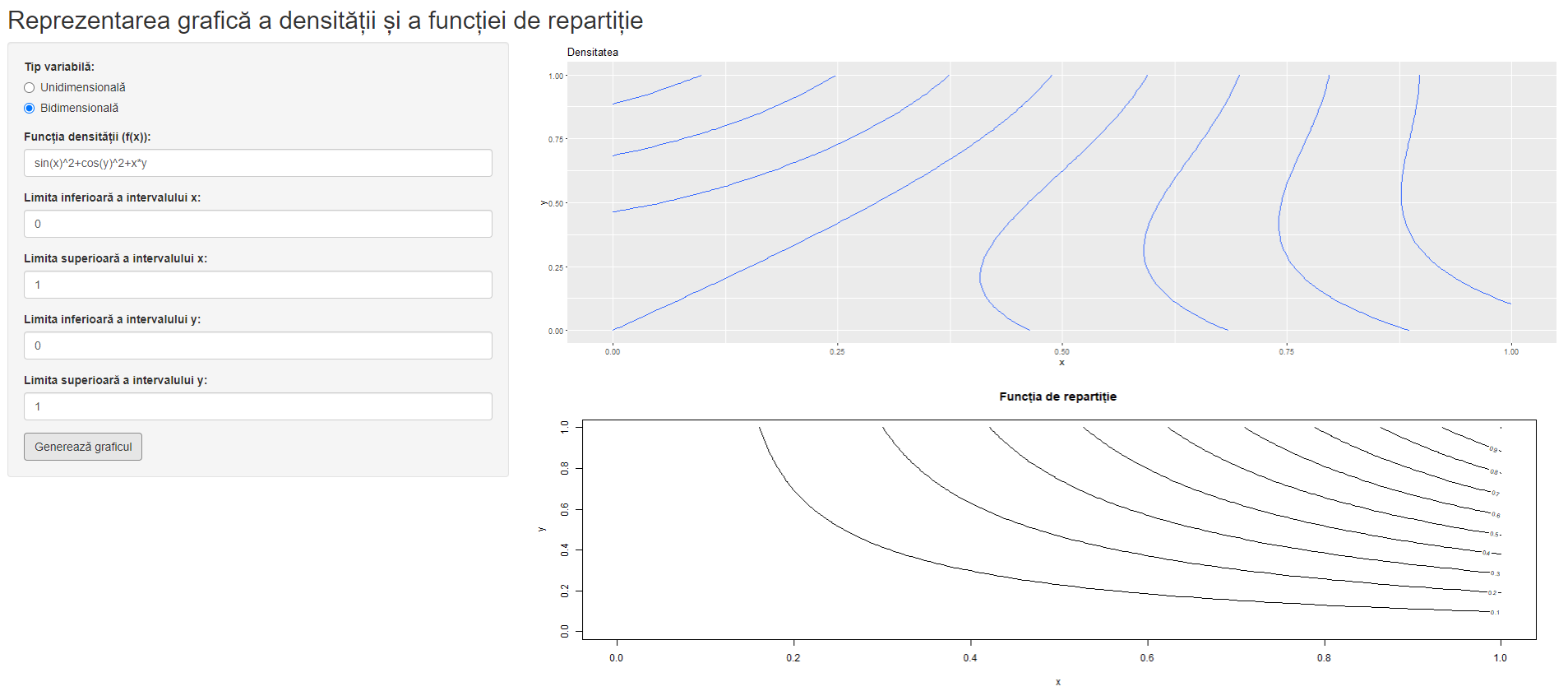
# Rulăm aplicația

shinyApp(ui = ui\_Grafice, server = server\_Grafice)

Exemplu pentru cazul v.a. unidimensională:



Exemplu pentru cazul v.a. bidimensională:



1. Calculul mediei, dispersiei și a momentelor inițiale și centrate pȃnă la ordinul 4 (dacă există) atȃt pentru v.a. bidimensională cȃt și pentru v.a. unidimensionale ce o compun. Atunci cȃnd unul dintre momente nu există, se va afișa un mesaj corespunzător către utilizator.

**Rezolvare:**

Pentru momentele v.a. X și Y se va preciza ordinul acestora de catre utilizator. Pentru v.a. bidimensională XY am calculat momentele Am utilizat următoarele formule:

**Variabile unidimensionale**

**Variabile bidimensionale**

 Am verificat corectitudinea valorilor calculate folosind relațiile:

# Exemplu pentru teste

# f(x,y)=1/5\*(x+y+1) x din [0,1], y din [0,2]

# 0 in rest

# E(x)=8/15=0.53

# E(y)=17/15=1.13

# Var(x)= 37/450=0.08

# Var(y)=71/225=0.31

# Cov(x,y)=-1/225=-0.04

# Cor(x,y)=-0.02

library(shiny)

library(pracma)

ui <- fluidPage(

titlePanel("Media, dispersia, momentele initiale si centrate"),

sidebarLayout(

sidebarPanel(

textInput("functie", "Introduceti functia f(x, y):", "1/5\*(x+y+1)"),

numericInput("x1\_val", "Valoarea pentru x1:", 0),

numericInput("x2\_val", "Valoarea pentru x2:", 1),

numericInput("y1\_val", "Valoarea pentru y1:", 0),

numericInput("y2\_val", "Valoarea pentru y2:", 2),

numericInput("momX\_val", "Valoarea pentru momentul lui X:", 2),

numericInput("momY\_val", "Valoarea pentru momentul lui Y:", 2),

actionButton("calculeaza", "Calculeaza")

),

mainPanel(

h5("Media:"),

verbatimTextOutput("rezultat\_medie"),

h5("Dispersia:"),

verbatimTextOutput("rezultat\_dispersie"),

h5("Momentele initiale si centrate pentru v.a. bidimensionala (01) (10) (20) (02):"),

verbatimTextOutput("rezultat\_momente\_XY"),

h5("Momentul initial si centrat pentru v.a unidimensionala X"),

verbatimTextOutput("rezultat\_momente\_X"),

h5("Momentul initial si centrat pentru v.a unidimensionala Y:"),

verbatimTextOutput("rezultat\_momente\_Y")

)

)

)

server <- function(input, output) {

observeEvent(input$calculeaza, {

tryCatch({

expr <- parse(text = paste("f <- function(x, y) {", input$functie, "}"))

eval(expr)

x1 <- input$x1\_val

x2 <- input$x2\_val

y1 <- input$y1\_val

y2 <- input$y2\_val

momX <- input$momX\_val

momY <- input$momY\_val

# Repartiția marginală a lui X

f\_X <- function(x) {

#integrate(function(y) f(x, y),-Inf, Inf)$value

integrate(function(y) f(x, y), y1, y2)$value

}

# Repartiția marginală a lui Y

f\_Y <- function(y) {

#integrate(function(x) f(x, y),-Inf, Inf)$value

integrate(function(x) f(x, y), x1, x2)$value

}

# Media

mX <- integrate (Vectorize(function (x) {x \* f\_X(x)}), x1, x2)$value

mY <- integrate (Vectorize(function (y) {y \* f\_Y(y)}), y1, y2)$value

mXY <- integral2(Vectorize(function (x ,y ) {(x)\*(y)\*f(x ,y)}),x1,x2,y1,y2)$Q

# Dispersia

varX <- integrate (Vectorize(function (x) {(x-mX)\*(x-mX)\*f\_X(x)}),x1,x2)$value

varY <- integrate (Vectorize(function (y) {(y-mY)\*(y-mY)\*f\_Y(y)}),y1,y2)$value

#Momentele initiale

mmX<- integrate (Vectorize(function (x) {x^momX \* f\_X(x)}), x1, x2)$value

if (!is.finite(mmX)) mmX <- "Nu se poate calcula"

mmY <- integrate (Vectorize(function (y) {y^momY \* f\_Y(y)}), y1, y2)$value

if (!is.finite(mmY)) mmY <- "Nu se poate calcula"

mXY\_01 <-integral2(Vectorize(function (x ,y ) {(x^0)\*(y^1)\*f(x ,y)}),x1,x2,y1,y2)$Q

if (!is.finite(mXY\_01)) mXY\_01 <- "Nu se poate calcula"

mXY\_10 <-integral2(Vectorize(function (x ,y ) {(x^1)\*(y^0)\*f(x ,y)}),x1,x2,y1,y2)$Q

if (!is.finite(mXY\_10)) mXY\_10 <- "Nu se poate calcula"

mXY\_02 <-integral2(Vectorize(function (x ,y ) {(x^0)\*(y^2)\*f(x ,y)}),x1,x2,y1,y2)$Q

if (!is.finite(mXY\_02)) mXY\_02 <- "Nu se poate calcula"

mXY\_20 <-integral2(Vectorize(function (x ,y ) {(x^2)\*(y^0)\*f(x ,y)}),x1,x2,y1,y2)$Q

if (!is.finite(mXY\_20)) mXY\_20 <- "Nu se poate calcula"

#Momentele centrate

mcX <- integrate (Vectorize(function (x) {(x-mX)^momX\*f\_X(x)}),x1,x2)$value

if (!is.finite(mcX)) mcX <- "Nu se poate calcula"

mcY <- integrate (Vectorize(function (y) {(y-mY)^momY\*f\_Y(y)}),y1,y2)$value

if (!is.finite(mcY)) mcY <- "Nu se poate calcula"

mcXY\_01 <-integral2(Vectorize(function (x ,y ) {(x-mX)^0\*(y-mY)^1\*f(x ,y)}),x1,x2,y1,y2)$Q

if (!is.finite(mcXY\_01)) mcXY\_01 <- "Nu se poate calcula"

mcXY\_10 <-integral2(Vectorize(function (x ,y ) {(x-mX)^1\*(y-mY)^0\*f(x ,y)}),x1,x2,y1,y2)$Q

if (!is.finite(mcXY\_10)) mcXY\_10 <- "Nu se poate calcula"

mcXY\_02 <-integral2(Vectorize(function (x ,y ) {(x-mX)^0\*(y-mY)^2\*f(x ,y)}),x1,x2,y1,y2)$Q

if (!is.finite(mcXY\_02)) mcXY\_02 <- "Nu se poate calcula"

mcXY\_20 <-integral2(Vectorize(function (x ,y ) {(x-mX)^2\*(y-mY)^0\*f(x ,y)}),x1,x2,y1,y2)$Q

if (!is.finite(mcXY\_20)) mcXY\_20 <- "Nu se poate calcula"

output$rezultat\_medie <- renderPrint({

rezultat\_list <- list("Media\_X"=mX, "Media\_Y"=mY,"Media\_XY"=mXY)

names(rezultat\_list) <- c("Media\_X", "Media\_Y","Media\_XY")

rezultat\_list

})

output$rezultat\_dispersie <- renderPrint({

rezultat\_list <- list( "Dispersia\_X"=varX, "Dispersia\_Y"=varY)

names(rezultat\_list) <- c("Dispersia\_X", "Dispersia\_Y")

rezultat\_list

})

output$rezultat\_momente\_XY <- renderPrint({

rezultat\_list <- list( "M\_initial\_01\_XY"=mXY\_01, "M\_initial\_10\_XY"=mXY\_10,

"M\_initial\_02\_XY"=mXY\_02, "M\_initial\_20\_XY"=mXY\_20,

"M\_centrat\_01\_XY"=mcXY\_01, "M\_centrat\_10\_XY"=mcXY\_10,

"M\_centrat\_02\_XY"=mcXY\_02, "M\_centrat\_20\_XY"=mcXY\_20)

names(rezultat\_list) <- c("M\_initial\_01\_XY", "M\_initial\_10\_XY", "M\_initial\_02\_XY",

"M\_initial\_20\_XY", "M\_centrat\_01\_XY", "M\_centrat\_10\_XY",

"M\_centrat\_02\_XY", "M\_centrat\_20\_XY")

rezultat\_list

})

output$rezultat\_momente\_X <- renderPrint({

rezultat\_list <- list( "M\_initial\_X"=mmX, "M\_centrat\_X"=mcX)

names(rezultat\_list) <- c("M\_initial\_X", "M\_centrat\_X")

rezultat\_list

})

output$rezultat\_momente\_Y <- renderPrint({

rezultat\_list <- list("M\_initial\_Y"=mmY, "M\_centrat\_Y"=mcY)

names(rezultat\_list) <- c("M\_initial\_Y", "M\_centrat\_Y")

rezultat\_list

})

}, error = function(e) {

output$rezultat <- renderPrint({

as.character(e)

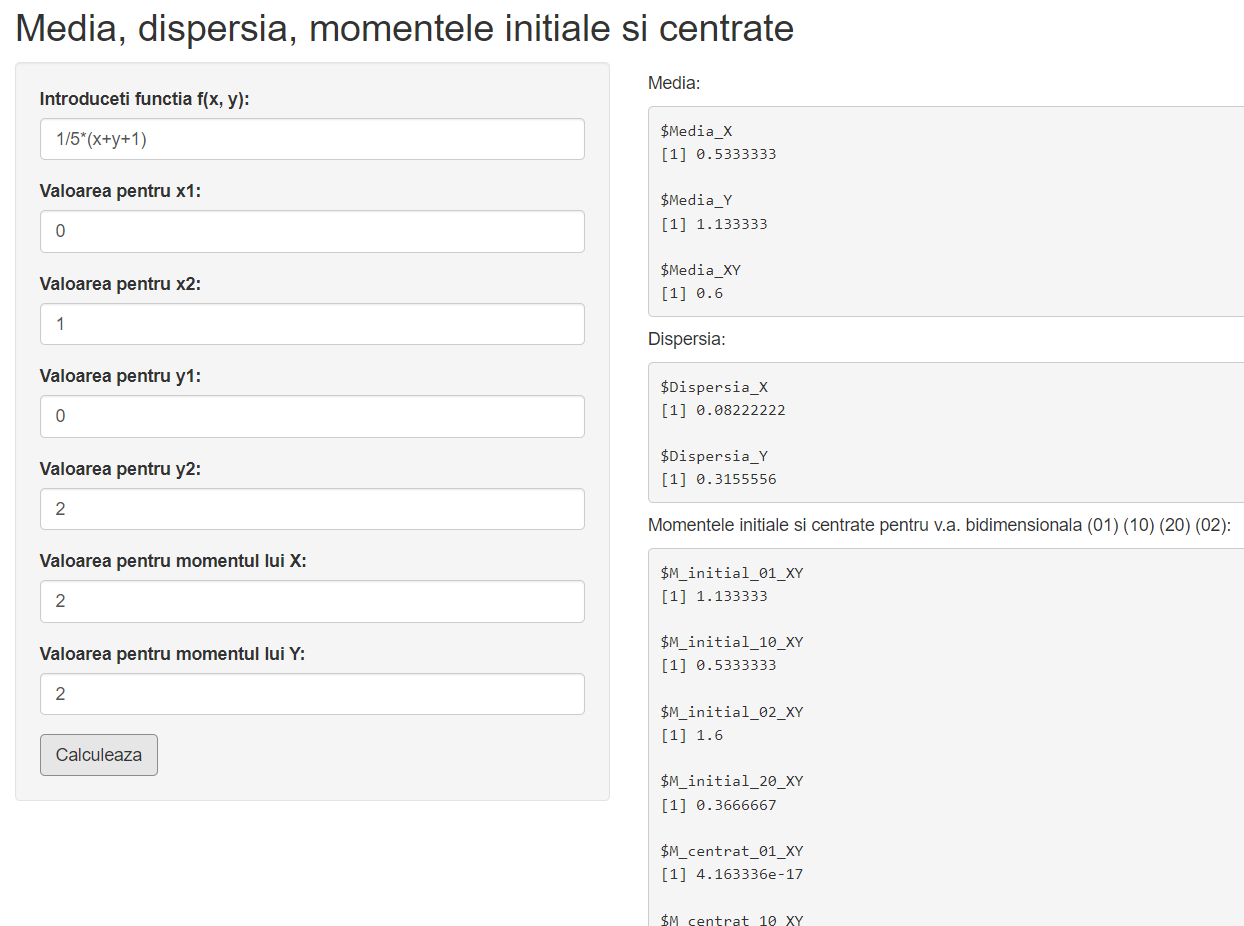
})

})

})

}

shinyApp(ui = ui, server = server)



1. Calculul mediei și dispersiei unei variabile aleatoare g(X), unde X are o repartiție continuă unidimensională cunoscută iar g este o funcție continuă precizată de utilizator. Se obține punctaj bonus pentru realizarea aceleiași cerințe și pentru cazul bidimensional.

**Rezolvare:**

Codul elaborat pentru rezolvarea cerinței este o aplicație Shiny care permite utilizatorului să introducă două funcții unidimensionale, f(x) și g(x), împreună cu două parametri numerici, a și b. Apoi, calculează media și dispersia pentru variabila aleatoare g(X), unde X este o variabilă aleatoare cu densitatea de probabilitate f(x).

Pentru calculul integralelor a fost necesar să utilizăm funcția **integrate** din pachetul **stats**.

library(shiny)

library(stats)

ui\_MediaDispersie <- fluidPage(

titlePanel("Calculul Mediei și Dispersiei pentru g(X)"),

sidebarLayout(

sidebarPanel(

textInput("functie\_f", "Introduceti functia f(x):", ""),

textInput("functie\_g", "Introduceti functia g(x):", ""),

numericInput("parametru\_a", "Parametrul a:", 0),

numericInput("parametru\_b", "Parametrul b:", 1),

actionButton("calculeaza", "Calculeaza")

),

mainPanel(

h3("Rezultate:"),

verbatimTextOutput("rezultat\_medie"),

verbatimTextOutput("rezultat\_dispersie")

)

)

)

server\_MediaDispersie <- function(input, output) {

observeEvent(input$calculeaza, {

tryCatch({

# Extrage functiile f(x) si g(x) introduse de utilizator

expr\_f <- parse(text = paste("f <- function(x) {", input$functie\_f, "}"))

expr\_g <- parse(text = paste("g <- function(x) {", input$functie\_g, "}"))

eval(expr\_f)

eval(expr\_g)

# Verificam validitatea functiilor f si g

if (!is.function(f) || !is.function(g)) {

stop("Introduceti functii valide pentru f(x) si g(x).")

}

# Extragem parametrii a si b introdusi de utilizator

a <- input$parametru\_a

b <- input$parametru\_b

# Calculam media pentru g(X) unde X ~ f(x)

media\_g <- integrate(function(x) g(x) \* f(x), lower = a, upper = b)$value

# Calculam dispersia pentru g(X) unde X ~ f(x)

dispersie\_g <- integrate(function(x) g(x)^2 \* f(x), lower = a, upper = b)$value - media\_g^2

output$rezultat\_medie <- renderPrint({

paste("Media g(X):", media\_g)

})

output$rezultat\_dispersie <- renderPrint({

paste("Dispersia g(X):", dispersie\_g)

})

}, error = function(e) {

output$rezultat\_medie <- renderPrint({

as.character(e)

})

output$rezultat\_dispersie <- renderPrint({

as.character(e)

})

})

})

}

shinyApp(ui = ui\_MediaDispersie, server = server\_MediaDispersie)

1. Crearea unei funcții P care permite calculul diferitelor tipuri de probabilități asociate unei variabile aleatoare continue unidimensionale/bidimensionale.

**Rezolvare:**

Codul elaborat permite utilizatorului să selecteze tipul de variabilă aleatoare continuă (unidimensională/ bidimensională) pentru care vrea să calculeze o probabilitate de formă standard, adică:

respectiv

unde semnele pot fi atât de mai mare/mic strict cât și de mai mare/mic sau egal, acestea fiind la fel pentru fiecare caz.

Pentru calculul efectiv al probabilităților, vom stabili capetele intervalului pentru care se va efectua integrarea, iar apoi vom integra, folosind funcția integrate din pachetul pracma pentru cazul unei variabile aleatoare continue unidimensionale, respectiv funcția integral2 pentru calculul integralei duble în cazul unei variabile continue bidimensionale.

library(shiny)

library(pracma)

ui\_Probabilitati <- fluidPage(

titlePanel("Calculul Probabilitatilor"),

sidebarLayout(

sidebarPanel(

radioButtons("tip\_variabila", "Tip variabilă:",

choices = c("Unidimensională", "Bidimensională"),

selected = "Unidimensională"),

textInput("functie\_densitate", "Introduceti functia densitatii f(x):", ""),

textInput("valoare\_x1", "Valoare x1:", ""),

textInput("valoare\_x2", "Valoare x2:", ""),

conditionalPanel(

condition = "input.tip\_variabila == 'Bidimensională'",

textInput("valoare\_y1", "Valoare y1:", ""),

textInput("valoare\_y2", "Valoare y2:", "")

),

textInput("valoare\_a1", "Valoare a1:", ""),

textInput("valoare\_b1", "Valoare b1:", ""),

conditionalPanel(

condition = "input.tip\_variabila == 'Bidimensională'",

textInput("valoare\_a2", "Valoare a2:", ""),

textInput("valoare\_b2", "Valoare b2:", "")

),

actionButton("calculeaza\_probabilitati", "Calculeaza Probabilitati")

),

mainPanel(

h3("Rezultate:"),

textOutput("rezultat\_probabilitate\_1"),

textOutput("rezultat\_probabilitate\_2"),

textOutput("rezultat\_probabilitate\_3"),

textOutput("rezultat\_probabilitate\_4")

)

)

)

server\_Probabilitati <- function(input, output) {

observeEvent(input$calculeaza\_probabilitati, {

tryCatch({

# Extrage functia densitatii f(x) introdusa de utilizator

expr\_densitate <- parse(text = paste("f <- function(x, y) {", input$functie\_densitate, "}"))

eval(expr\_densitate)

# Verificam validitatea functiei densitatii

if (!is.function(f)) {

stop("Introduceti o functie valida pentru densitatea f(x).")

}

# Extragem valorile a1, b1, a2, b2, x1,x2, y1, y2 introduse de utilizator

x1 <- as.numeric(input$valoare\_x1)

x2 <- as.numeric(input$valoare\_x2)

y1 <- as.numeric(input$valoare\_y1)

y2 <- as.numeric(input$valoare\_y2)

a1 <- as.numeric(input$valoare\_a1)

b1 <- as.numeric(input$valoare\_b1)

a2 <- as.numeric(input$valoare\_a2)

b2 <- as.numeric(input$valoare\_b2)

# Initializam a1, a2, b1 și b2 cu valori adecvate

a1 <- max(a1, x1)

b1 <- min(b1, x2)

if (input$tip\_variabila == "Bidimensională")

{

a2 <- max(a2, y1)

b2 <- min(b2, y2)

}

# Tratam valorile infinite explicit

a1 <- ifelse(a1 == -Inf, -1e10, a1)

b1 <- ifelse(b1 == Inf, 1e10, b1)

a2 <- ifelse(a2 == -Inf, -1e10, a2)

b2 <- ifelse(b2 == Inf, 1e10, b2)

# Calculam probabilitatile asociate variabilei aleatoare X ~ f(x) sau X,Y ~ f(x, y)

if (input$tip\_variabila == "Unidimensională") {

prob\_a1\_X\_b1 <- integrate(Vectorize(f), a1, b1)$value

prob\_a1\_le\_X\_b1 <- integrate(Vectorize(f), a1, b1)$value

prob\_a1\_X\_le\_b1 <- integrate(Vectorize(f), a1, b1)$value

prob\_a1\_le\_X\_le\_b1 <- integrate(Vectorize(f), a1, b1)$value

} else {

prob\_a1\_X\_b1 <- integral2(Vectorize(f), a1, b1, a2, b2)$Q

prob\_a1\_le\_X\_b1 <- integral2(Vectorize(f), a1, b1, a2, b2)$Q

prob\_a1\_X\_le\_b1 <- integral2(Vectorize(f), a1, b1, a2, b2)$Q

prob\_a1\_le\_X\_le\_b1 <- integral2(Vectorize(f), a1, b1, a2, b2)$Q

}

output$rezultat\_probabilitate\_1 <- renderText({

paste("P(", a1, "< X <", b1, "): ", prob\_a1\_X\_b1)

})

output$rezultat\_probabilitate\_2 <- renderText({

paste("P(", a1, "<= X <", b1, "): ", prob\_a1\_le\_X\_b1)

})

output$rezultat\_probabilitate\_3 <- renderText({

paste("P(", a1, "< X <=", b1, "): ", prob\_a1\_X\_le\_b1)

})

output$rezultat\_probabilitate\_4 <- renderText({

paste("P(", a1, "<= X <=", b1, "): ", prob\_a1\_le\_X\_le\_b1)

})

}, error = function(e) {

output$rezultat\_probabilitate\_1 <- renderText({

as.character(e)

})

output$rezultat\_probabilitate\_2 <- renderText({

""

})

output$rezultat\_probabilitate\_3 <- renderText({

""

})

output$rezultat\_probabilitate\_4 <- renderText({

""

})

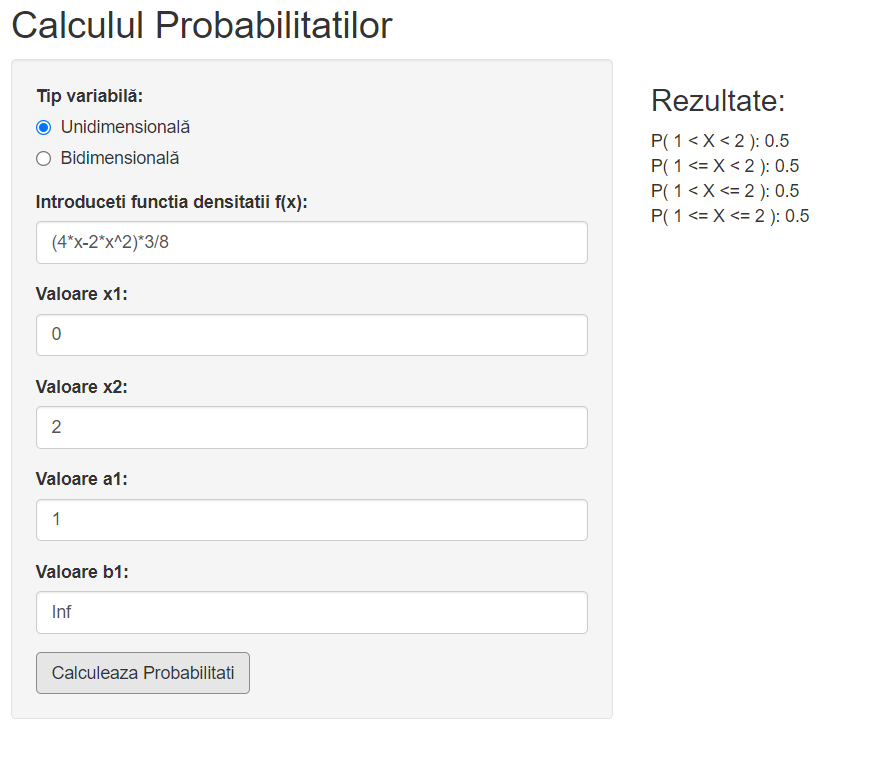
})

})

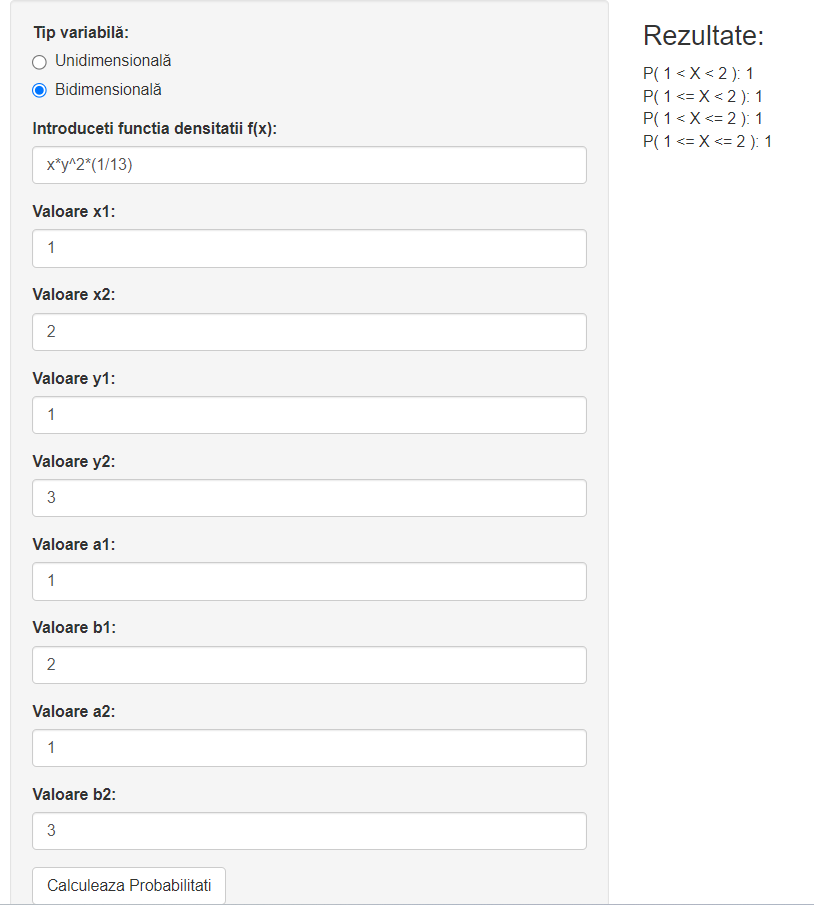
}

shinyApp(ui = ui\_Probabilitati, server = server\_Probabilitati)

Exemplul 1



Exemplu 2



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

**Rezolvare:**

Am folosit formulele:

=

# Exemplu 1 pentru teste

# f(x,y) = 2\*x\*y\*exp(-(x^2+y^2)), x>0, y>0

# 0, in rest

# E(x)=E(y)=sqrt(pi)/4=0.11

# E(x^2)=1/2

# Var(X)=Var(Y)=1/2-pi/16

# Exemplu 2 pentru teste

# f(x,y)=1/5\*(x+y+1) x din [0,1], y din [0,2]

# 0 in rest

# E(x)=8/15=0.53

# E(y)=17/15=1.13

# Var(x)= 37/450=0.08

# Var(y)=71/225=0.31

# C(x,y)=-1/225=-0.04

# Cor(x,y)=-0.02

library(shiny)

library(pracma)

ui <- fluidPage(

titlePanel("Covarianta si coeficientul de corelatie pentru doua variabile aleatoare continue"),

sidebarLayout(

sidebarPanel(

textInput("functie", "Introduceti functia f(x, y):", "1/5\*(x+y+1)"),

numericInput("x1\_val", "Valoarea pentru x1:", 0),

numericInput("x2\_val", "Valoarea pentru x2:", 1),

numericInput("y1\_val", "Valoarea pentru y1:", 0),

numericInput("y2\_val", "Valoarea pentru y2:", 2),

actionButton("calculeaza", "Calculeaza")

),

mainPanel(

h3("Rezultat:"),

verbatimTextOutput("rezultat")

)

)

)

server <- function(input, output) {

observeEvent(input$calculeaza, {

tryCatch({

expr <- parse(text = paste("f <- function(x, y) {", input$functie, "}"))

eval(expr)

x1 <- input$x1\_val

x2 <- input$x2\_val

y1 <- input$y1\_val

y2 <- input$y2\_val

# Repartiția marginală a lui X

f\_X <- function(x) {

integrate(function(y) f(x, y), y1, y2)$value

}

# Repartiția marginală a lui Y

f\_Y <- function(y) {

integrate(function(x) f(x, y), x1, x2)$value

}

# Mediile celor 2 variabile aleatoare

mX <- integrate (Vectorize(function (x) {x \* f\_X(x)}), x1, x2)$value

mY <- integrate (Vectorize(function (y) {y \* f\_Y(y)}), y1, y2)$value

# Dispersiile celor 2 variabile aleatoare

varX <- integrate (Vectorize(function (x) {(x-mX)\*(x-mX)\*f\_X(x)}),x1,x2)$value

varY <- integrate (Vectorize(function (y) {(y-mY)\*(y-mY)\*f\_Y(y)}),y1,y2)$value

# Covarianta

cov\_XY <- integral2(Vectorize(function (x ,y ) {(x-mX)\*(y-mY)\*f(x ,y)}),x1,x2,y1,y2)$Q

#Coeficientul de corelatie

cor\_XY <- cov\_XY/ (sqrt(varX)\*sqrt(varY))

output$rezultat <- renderPrint({

rezultat\_list <- list("Ex"=mX, "Ey"=mY, "Varx"=varX, "Vary"=varY,

"Covarianta" = cov\_XY,"Corelatia"=cor\_XY)

names(rezultat\_list) <- c("Ex","Ey","Varx", "Vary", "Covarianta", "Corelatia")

rezultat\_list

})

}, error = function(e) {

output$rezultat <- renderPrint({

as.character(e)

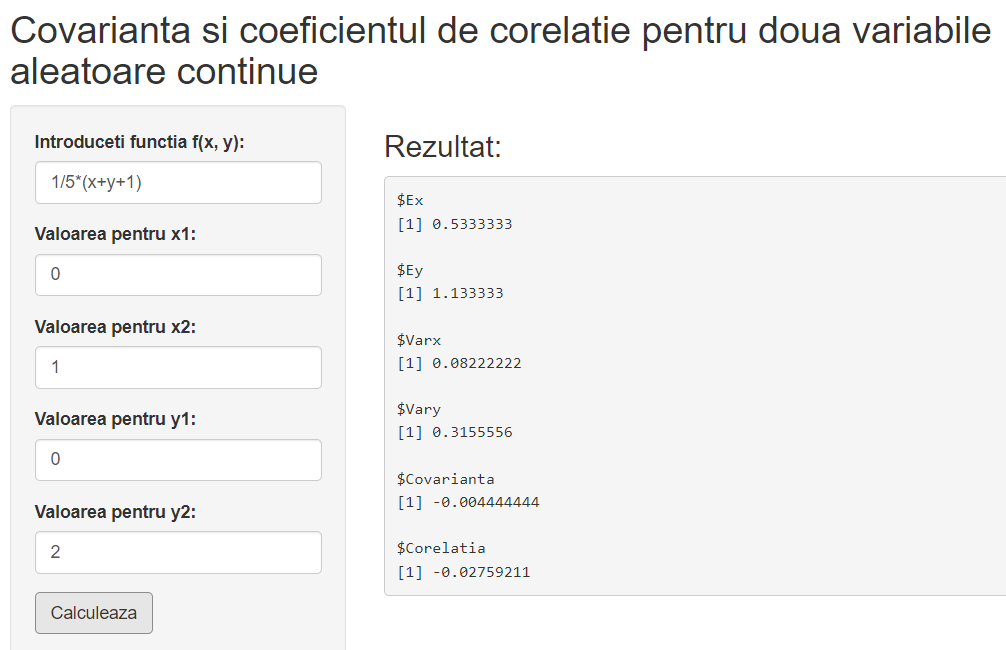
})

})

})

}

shinyApp(ui = ui, server = server)



**Concluzii**

Am învățat să lucrăm cu pachetele R ***discretRV, pracma, shiny***. Am aplicat în rezolvarea problemelor noțiunile teoretice studiate la orele de curs și de laborator / seminar.