UNIVERSITATEA DIN BUCUREȘTI

FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ

DEPARTAMENTUL INFORMATICĂ

Proiect la probabilități și statistică

Studenți:

Pincu Iulia Maria Andreea

Bejenaru Andrada Elena

Radu Ioana Alexandra

Vlaicu Alexandra-Alina

## Exercițiul 1

Problema generală din exercițiul 1 se concentrează pe analiza și manipularea probabilităților asociate cu două variabile aleatoare discrete, X și Y. Scopul este de a dezvolta o serie de funcții în R care să faciliteze calculul repartițiilor comune și marginale, probabilităților condiționate, covarianței și altor probabilități legate de perechea (X, Y). Aceste funcții ajută la explorarea relațiilor dintre variabilele aleatoare în diferite contexte și la aplicarea conceptelor de bază din teoria probabilităților și statisticilor într-un cadru practic.

Definiții pentru conceptele folosite:

**Variabilă Aleatoare:** Este o funcție care asociază un număr real fiecărui rezultat al unui experiment aleator. Variabilele aleatoare pot fi de două tipuri: discrete sau continue.

**Variabilă Aleatoare Discretă**: Este o variabilă aleatoare care poate lua valori dintr-un set finit sau numărabil, de exemplu, rezultatul aruncării unui zar (numere întregi de la 1 la 6). Fiecare valoare posibilă a variabilei aleatoare poate fi asociată cu o probabilitate specifică.

**Repartiția Comună**: Reprezintă modul în care probabilitățile sunt distribuite peste combinațiile posibile ale valorilor a două sau mai multe variabile aleatoare. De exemplu, în cazul a două variabile X și Y, repartiția comună arată probabilitatea ca X să fie egal cu x și Y să fie egal cu y pentru toate valorile posibile ale lui x și y.

**Repartiția Marginală**: Oferă probabilitățile pentru o singură variabilă aleatoare dintr-un set de variabile, ignorând restul variabilelor. Este calculată prin sumarea probabilităților din repartiția comună peste valorile celorlalte variabile.

**Probabilitatea Condiționată**: Este probabilitatea ca un eveniment să se întâmple, având în vedere că un alt eveniment a avut loc deja. Matematic, probabilitatea lui A dat fiind B este notată ca P(A|B).

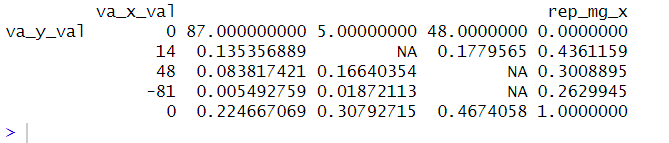
**Covarianța**: Este o măsură statistică a gradului în care două variabile aleatoare se schimbă împreună. Dacă variabilele tind să varieze în aceeași direcție, covarianța este pozitivă; dacă variază în direcții opuse, este negativă.

### a)

Funcția **frepcomgen** generează o matrice reprezentând repartiția comună a două variabile aleatoare discrete, X și Y. Procesul include:

* Generarea Matricei: Se creează inițial o matrice cu valori aleatoare folosind runif, care produce numere aleatoare uniform distribuite.
* Normalizarea: Matricea este normalizată astfel încât suma tuturor valorilor să fie egală cu 1, transformându-le în probabilități.
* Calculul Repartițiilor Marginale: Se calculează repartițiile marginale pentru X și Y prin sumarea valorilor pe rânduri și coloane.
* Adăugarea Repartițiilor Marginale și Valorilor X și Y: Repartițiile marginale și valorile generate pentru X și Y sunt adăugate ca rânduri și coloane suplimentare în matrice.
* Plasarea Valorilor NA: Se plasează valori NA (Not Available) în matrice pentru a indica locurile unde valorile lipsesc sau sunt necunoscute.

| frepcomgen <- function(n, m) {  # Generam o matrice cu repartiția comuna  matr <- matrix(runif(n \* m), n, m)  # Normalizam matricea pentru a avea sume de 1 pe randuri și coloane  matr <- matr / sum(matr)  # Calculam repartitiile marginale  rep\_mg\_x <- apply(matr, 1, sum)  rep\_mg\_y <- apply(matr, 2, sum)  # Adaugam repartițiile marginale la matrice  matr <- cbind(matr, rep\_mg\_x)  matr <- rbind(matr, c(rep\_mg\_y, sum(rep\_mg\_y)))  # Adaugam valorile pentru variabilele aleatoare X și Y  va\_x\_val <- c(sample(-100:100, n, FALSE), 0)  va\_y\_val <- c(0, sample(-100:100, m, FALSE), 0)  matr <- cbind(va\_x\_val, matr)  matr <- rbind(va\_y\_val, matr)  # Plasam o singură valoare NA pe fiecare linie  for (i in 2:(n + 1)) {  coloana\_NA <- sample(2:m+1, 1)  matr[i, coloana\_NA] <- NA  }  return(matr)  }  # Exemplu de utilizare  n <- 3  m <- 3  print(frepcomgen(n, m)) |
| --- |

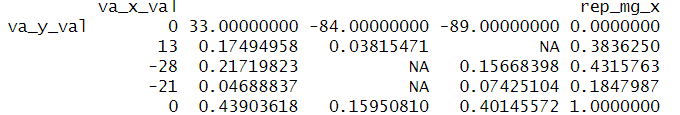


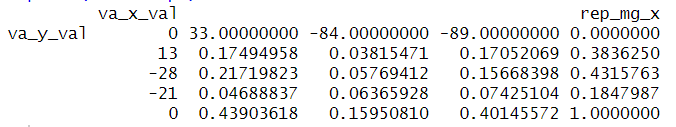
### b)

Funcția **fcomplrepcom** este folosită pentru a completa o matrice de repartiție comună incompletă a două variabile aleatoare discrete, X și Y. Procedează astfel:

* **Parcurgerea Matricei**: Funcția parcurge fiecare element al matricei, excluzând prima linie și prima coloană (care reprezintă valorile X și Y) și ultima linie și ultima coloană (repartițiile marginale).
* **Completarea Valorilor Lipsă (NA)**: Pentru fiecare celulă cu valoarea NA, se calculează suma valorilor pe linia respectivă (excluzând repartiția marginală) și se înlocuiește NA cu diferența dintre repartiția marginală a liniei și suma calculată.

| fcomplrepcom <- function(matr) {  nr <- nrow(matr)  nc <- ncol(matr)  for (i in 2:(nr - 1)) {  for (j in 2:(nc - 1)) {  if (is.na(matr[i, j])) {  # Suma valorilor pe linia respectiva, excluzand prima coloana și repartiția marginala  suma\_linie <- sum(matr[i, 2:(nc - 1)], na.rm = TRUE)  # Valoarea completata este repartiția marginala a liniei minus suma valorilor de pe linie  matr[i, j] <- matr[i, nc] - suma\_linie  }  }  }  return(matr)  }  # Exemplu de utilizare  matr\_ini <- frepcomgen(n, m)  print(matr\_ini)  matr\_compl <- fcomplrepcom(matr\_ini)  print(matr\_compl) |
| --- |



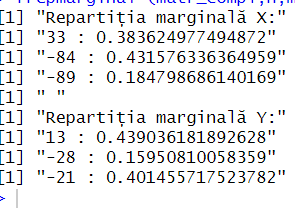


### c)

Funcția **frepmarginal** este destinată calculului repartițiilor marginale pentru două variabile aleatoare discrete X și Y, pe baza unei matrice de repartiție comună. Procedeul este următorul:

* **Calculul Repartițiilor Marginale**: Funcția parcurge matricea pentru a calcula repartițiile marginale pentru X și Y. Pentru X, sumează valorile pe coloane, iar pentru Y, pe rânduri.
* **Afișarea Repartițiilor Marginale:** Repartițiile marginale sunt afișate în consolă, oferind o vedere clară asupra distribuției probabilităților pentru fiecare valoare a variabilelor aleatoare X și Y.

| frepmarginal <- function(matr, n, m) {  marg\_x <- numeric(n)  marg\_y <- numeric(m)  # Calculăm repartițiile marginale pentru X  for (i in 2:(n + 1)) {  marg\_x[i - 1] <- matr[i, m + 2]  }  # Calculăm repartițiile marginale pentru Y  for (j in 2:(m + 1)) {  marg\_y[j - 1] <- matr[n + 2, j]  }  # Afișăm repartițiile marginale în format de perechi  print("Repartiția marginală X:")  for (i in 1:n) {  print(paste(matr[1, i + 1], ":", marg\_x[i]))  }  print(" ")  print("Repartiția marginală Y:")  for (j in 1:m) {  print(paste(matr[j + 1, 1], ":", marg\_y[j]))  }  }  #Exemplu de utilizare  frepmarginal (matr\_compl,n,m)  print(matr\_compl) |
| --- |



### d)

Funcția **fpropcov**  calculează covarianța a două variabile aleatoare, fiind ajutată și de f**medie** care calculează media unei variabile aleatoare, folosind valorile și probabilitățile asociate și de **fvar** care determină varianța unei variabile aleatoare.

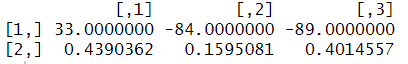
| # Calculează media unei variabile aleatoare  fmedie <- function(var){  medie <- 0  # Parcurge toate valorile și calculează suma valoare \* probabilitate  for(i in 1:dim(var)[2])  medie <- medie + (var[1,i] \* var[2,i])  return (medie)  }  # Calculează varianța unei variabile aleatoare  fvar <- function(X, b=1){  # Ridică la pătrat valorile variabilei aleatoare  Xpatrat <- matrix(c(X[1,]^2, X[2,]), nrow=2, byrow = TRUE)  # Din prop variantei, Var(aX) = a^2 \* Var(X)  return (b \* b \* (fmedie(Xpatrat) - fmedie(X) ^ 2))  }  # Calculează covarianța pentru combinații liniare ale variabilelor aleatoare  fpropcov <- function(matr, a=1, b=0, c=0, d=1){  # Inițializează vectorii pentru valori și probabilități  val\_var <- c()  prob\_var <- c()    # Extragerea valorilor și probabilităților pentru X  for(i in 2:dim(matr)[1]-1){  val\_var <- c(val\_var, matr[i,1])  prob\_var <- c(prob\_var, matr[i,dim(matr)[2]])  }  X <- matrix(c(val\_var, prob\_var), nrow=2, byrow = TRUE)  # Resetare vectori pentru Y  val\_var <- c()  prob\_var <- c()  # Extragerea valorilor și probabilităților pentru Y  for(i in 2:dim(matr)[2]-1){  val\_var <- c(val\_var, matr[1,i])  prob\_var <- c(prob\_var, matr[dim(matr)[1],i])  }  Y <- matrix(c(val\_var, prob\_var), nrow=2, byrow = TRUE)  # Verifică dacă X sau Y sunt constante  if(length(Y) == 1 || length(X) == 1) return (0)  # Calculul covarianței pentru combinații liniare  for(i in 1:dim(X)[2])  for(j in 1:dim(Y)[2]){  val\_var <- c(val\_var, X[1,i] \* Y[1,j])  prob\_var <- c(prob\_var, X[2,i] \* Y[2,j])  }  prob\_var <- tapply(prob\_var, val\_var, sum)  XY <- matrix(c(sort(unique(val\_var)), prob\_var), nrow=2, byrow = TRUE)  cova <- fmedie(XY)  covb <- fmedie(X) \* fmedie(Y)  covXY <- trunc((cova - covb) \* 10^4) / 10^4  cov <- a \* c \* fvar(X) + (a \* d + b \* c) \* covXY + b \* d \* fvar(Y)  return (cov)  }  # Exemplu de utilizare  fpropcov(matr\_compl, 1, 2, 1, 2) |
| --- |



### e)

Funcția **fPcond** calculează probabilitatea condiționată pentru variabilele aleatoare X și Y pe baza unei matrice de repartiție comună. Funcția gestionează trei cazuri: calculul probabilității condiționate a X dat fiind Y, a Y dat fiind X, și a unei perechi specifice (X, Y). Se utilizează principiul probabilității condiționate și formula P(A|B) = P(A ∩ B) / P(B), unde A și B sunt evenimente.

| fPcond <- function(matr, intX=NULL, compX=NULL, intY=NULL, compY=NULL){  # Calculul P(X|Y) dacă X nu este specificat  if(is.null(intX)) {  val\_var <- c() # Initializează vectorul pentru valorile lui X  prob\_var <- c() # Initializează vectorul pentru probabilitătile conditionate    # Parcurge valorile lui X și calculează P(X|Y)  for(i in 2:(dim(matr)[1]-1)){  val\_var <- c(val\_var, matr[i,1])  # Verifică dacă P(Y) este 0 pentru a evita impartirea la zero  if(fPcomun(matr,intY=intY,compY=compY)==0)  return ("Nu se poate")    # Calculeaza probabilitatea conditionată și o adauga la vector  prob\_var <- c(prob\_var, fPcomun(matr,matr[i,1],"==",intY,compY)  / fPcomun(matr,intY=intY,compY=compY))  }  # Creează o matrice cu valorile și probabilitatile condiționate  va\_noua <- matrix(c(val\_var,prob\_var), nrow = 2, byrow = TRUE)  return (va\_noua)  }  # Calculul P(Y|X) dacă Y nu este specificat  else if(is.null(intY)) {  val\_var <- c() # Inițializează vectorul pentru valorile lui Y  prob\_var <- c() # Inițializează vectorul pentru probabilitățile condiționate  # Parcurge valorile lui Y și calculează P(Y|X)  for(i in 2:(dim(matr)[2]-1)){  val\_var <- c(val\_var, matr[1,i])  # Verifica dacă P(X) este 0 pentru a evita imparțirea la zero  if(fPcomun(matr,intX,compX)==0)  return("Nu se poate")  # Calculeaza probabilitatea condiționata și o adauga la vector  prob\_var <- c(prob\_var, fPcomun(matr,intX,compX,matr[1,i],"==")  / fPcomun(matr,intX,compX))  }  # Creează o matrice cu valorile și probabilitatile condiționate  va\_noua <- matrix(c(val\_var,prob\_var), nrow = 2, byrow = TRUE)  return (va\_noua)  }  # Calculul P(X=x|Y=y) pentru valori specifice ale lui X și Y  else {  # Verifică dacă P(Y=y) este 0 pentru a evita împărțirea la zero  if(fPcomun(matr,intY=intY, compY=compY)==0) return(0)  # Returnează P(X=x, Y=y) / P(Y=y)  return (fPcomun(matr,intX, compX, intY, compY) / fPcomun(matr,intY=intY, compY=compY))  }  }  test <- fPcond(matr\_compl,intX=-30:20, compX="<")  print(test) |
| --- |



### f)

Funcția **fPcomun** în R este concepută pentru a calcula probabilitățile asociate cu variabilele aleatoare X și Y, bazate pe diferite condiții și intervale. Funcția gestionează mai multe scenarii:

* Când una dintre variabilele X sau Y nu este specificată, funcția calculează probabilitatea pentru cealaltă variabilă, luând în considerare un anumit interval sau o condiție dată (cum ar fi mai mic decât, mai mare decât, egal cu o anumită valoare).
* În situații în care ambele variabile sunt specificate, funcția parcurge matricea pentru a găsi suma probabilităților care corespund condițiilor impuse asupra ambelor variabile.

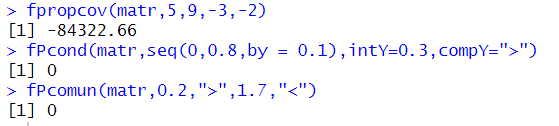
| fPcomun <- function(matr,intX=NULL, compX=NULL, intY=NULL, compY=NULL){  if(is.null(intX) && is.null(intY))  {  return (0)  }  if(is.null(intX))  {  #daca intX e NULL, atunci se calc doar prob lui Y  if(length(intY)==1) #daca dim e 1, inseamna ca avem interval nemarginit  {  if(is.null(compY))  return ("Eroare!")  if(compY=="<")  {  sum <- 0  for(i in 2:(dim(matr)[2]-1))  if(matr[1,i] < intY) sum <- sum + matr[dim(matr)[1],i]  return (sum)  }  else if(compY==">")  {  sum <- 0  for(i in 2:(dim(matr)[2]-1))  if(matr[1,i] > intY) sum <- sum + matr[dim(matr)[1],i]  return (sum)  }  else if(compY=="==")  {  sum <- 0  poz <- which(matr[1,2:(dim(matr)[2]-1)]==intY)  if(length(poz)>0) sum <- sum + matr[dim(matr)[1],poz+1]  return (sum)  }  else return ("Eroare!")  }  else #interval marginit  {  sum <- 0  for(i in 2:(dim(matr)[2]-1))  if(matr[1,i] > min(intY) && matr[1,i] < max(intY)) sum <- sum + matr[dim(matr)[1],i]  return (sum)  }  }  if(is.null(intY))  {  #daca intY e NULL, atunci se calc doar prob lui X  if(length(intX)==1) #daca dim e 1, inseamna ca avem interval nemarginit  {  if(is.null(compX))  return ("Eroare!")  if(compX=="<")  {  sum <- 0  for(i in 2:(dim(matr)[1]-1))  if(matr[i,1] < intX) sum <- sum + matr[i,dim(matr)[2]]  return (sum)  }  else if(compX==">")  {  sum <- 0  for(i in 2:(dim(matr)[1]-1))  if(matr[i,1] > intX) sum <- sum + matr[i,dim(matr)[2]]  return (sum)  }  else if(compX=="==")  {  sum <- 0  poz <- which(matr[2:(dim(matr)[1]-1),1]==intX)  #print(poz)  if(length(poz)>0) sum <- sum + matr[poz+1,dim(matr)[2]]  return (sum)  }  else return ("Eroare!")  }  else #interval marginit  {  sum <- 0  for(i in 2:(dim(matr)[1]-1))  if(matr[i,1] > min(intX) && matr[i,1] < max(intX)) sum <- sum + matr[i,dim(matr)[2]]  return (sum)  }  }  #cazul in care avem intervale si pt X, si pt Y  sum <- 0  for(i in 2:(dim(matr)[1]-1)) #parcurgem matr si cautam celule care indeplinesc ambele conditii  for(j in 2:(dim(matr)[2]-1))  {  if(length(intX)==1 && length(intY)==1)  {  # am format un string de conditie, de ex "matr[i,1] < 2"  condX <- paste("matr[i,1]",paste(compX, intX, sep = " "),sep = " ")  condY <- paste("matr[1,j]",paste(compY, intY, sep = " "),sep = " ")  # am parsat si evaluat stringul de conditie  if(eval(parse(text = condX)) && eval(parse(text = condY)))  {  sum <- sum + matr[i,j]  }  }  else if(length(intX)==1)  {  condX <- paste("matr[i,1]",paste(compX, intX, sep = " "),sep = " ")  # am folosit min si max in loc de %in% pentru cazurile in care  # extremitatile intervalului au zecimale  if(eval(parse(text = condX)) && matr[1,j] > min(intY) && matr[1,j] < max(intY))  sum <- sum + matr[i,j]  }  else if(length(intY)==1)  {  condY <- paste("matr[1,j]",paste(compY, intY, sep = " "),sep = " ")  if(eval(parse(text = condY)) && matr[i,1] > min(intX) && matr[i,1] < max(intX))  sum <- sum + matr[i,j]  }  else  {  if(matr[i,1] > min(intX) && matr[i,1] < max(intX)  && matr[1,j] > min(intY) && matr[1,j] < max(intY))  sum <- sum + matr[i,j]  }  }  return (sum)  }  print(matr\_compl)  fPcomun(matr\_compl,intY=-90:34,compY=">") |
| --- |



### g)

* **fpropcov(matr,5,9,-3,-2)**: Calculează covarianța pentru combinații liniare ale variabilelor aleatoare X și Y, unde 5, 9, -3 și -2 sunt coeficienții acestor combinații
* **fPcond(matr, seq(0,0.8, by = 0.1), intY=0.3, compY=">"):** Calculează probabilitatea condiționată a variabilei X dată fiind o condiție pe variabila Y (de exemplu, Y > 0.3)
* **fPcomun(matr, 0.2, ">", 1.7, "<"):** Calculează probabilitatea comună a variabilelor X și Y îndeplinind condițiile date (X > 0.2 și Y < 1.7)

| fpropcov(matr,5,9,-3,-2)  fPcond(matr,seq(0,0.8,by = 0.1),intY=0.3,compY=">")  fPcomun(matr,0.2,">",1.7,"<") |
| --- |



### h)

* **fverind:** Verifică dacă variabilele X și Y sunt independente. Folosește fpropcov pentru a calcula covarianța dintre X și Y. Dacă covarianța este 0, variabilele sunt considerate independente; altfel, sunt dependente. Independența variabilelor aleatoare este un concept central în probabilitate și statistici, indicând că prezența sau valoarea unei variabile nu o influențează pe cealaltă.
* **fvernecor**: Determină dacă variabilele X și Y sunt necorelate. Calculează coeficientul de corelație, care este covarianța normalizată prin divizarea cu produsul varianțelor lui X și Y. Dacă coeficientul este 0, variabilele sunt necorelate, ceea ce înseamnă că nu există o relație liniară între ele.

| fverind <- function(matr){  # initializam vectorii pentru valorile și probabilitățile variabilelor X și Y  val\_var <- c()  prob\_var <- c()  # extragem valorile și probabilitățile pentru X  for(i in 2:dim(matr)[1]-1){  val\_var <- c(val\_var, matr[i,1])  prob\_var <- c(prob\_var, matr[i, dim(matr)[2]])  }  X <- matrix(c(val\_var, prob\_var), nrow = 2, byrow = TRUE)  # resetare vectori pentru Y  val\_var <- c()  prob\_var <- c()  # extragem valorile si probabilitatile pentru Y  for(i in 2:dim(matr)[2]-1){  val\_var <- c(val\_var, matr[1, i])  prob\_var <- c(prob\_var, matr[dim(matr)[1], i])  }  Y <- matrix(c(val\_var, prob\_var), nrow = 2, byrow = TRUE)  # verificam daca covarianta este zero (ceea ce indica independenta)  if(fpropcov(matr) == 0)  return ("Independente")  else  return ("Dependente")  }  fverind(matr\_compl)  fvernecor <- function(matr){  #inițializam vectorii pentru valorile și probabilitatile variabilelor X și Y  val\_var <- c()  prob\_var <- c()  # extragem valorile și probabilitățile pentru X  for(i in 2:dim(matr)[1]-1){  val\_var <- c(val\_var, matr[i, 1])  prob\_var <- c(prob\_var, matr[i, dim(matr)[2]])  }  X <- matrix(c(val\_var, prob\_var), nrow = 2, byrow = TRUE)  # resetare vectori pentru Y  val\_var <- c()  prob\_var <- c()  # extragem valorile și probabilitatile pentru Y  for(i in 2:dim(matr)[2]-1){  val\_var <- c(val\_var, matr[1, i])  prob\_var <- c(prob\_var, matr[dim(matr)[1], i])  }  Y <- matrix(c(val\_var, prob\_var), nrow = 2, byrow = TRUE)  # calculeaza coeficientul de corelație și verifică dacă este zero  coef <- fpropcov(matr) / sqrt(fvar(X) \* fvar(Y))  if(coef == 0)  return ("Necorelate")  else  return (coef)  }  fvernecor(matr\_compl) |
| --- |





## Exercițiul 2:

Proiectul propus vizează dezvoltarea unor aplicații interactive în limbajul de programare R, utilizând pachete precum Shiny, Animate, Ggplot2, Pracma etc. care să permită manipularea și analiza variabilelor aleatoare continue bidimensionale.

Definiții pentru conceptele folosite:

**Teorema lui Fubini**: Este o teoremă importantă din calculul integral, care oferă o metodă de calcul a integralei duble. Dacă funcția f(x, y) este continuă pe un domeniu rectangular D definit de intervale [a, b] și [c, d], atunci integrala dublă poate fi calculată fie prin integrarea după x, apoi y, fie invers, rezultatul fiind același.

O **densitate de probabilitate** este o funcție non-negativă care măsoară probabilitatea ca valoarea unei variabile aleatoare să fie cuprinsă într-un anumit interval. Aceasta trebuie să satisfacă câteva proprietăți, cum ar fi să fie non-negativă și să aibă o sumă totală de 1 în domeniul său.

**Densitatea comună** a două variabile aleatoare, f(x, y), este o funcție care descrie probabilitatea ca cele două variabile să aibă anumite valori simultan.

**Densitatea marginală** a unei variabile aleatoare, X sau Y, reprezintă probabiliatea de a obține o anumită valoare pentru acea variabilă, ignorând valorile celeilate variabile.

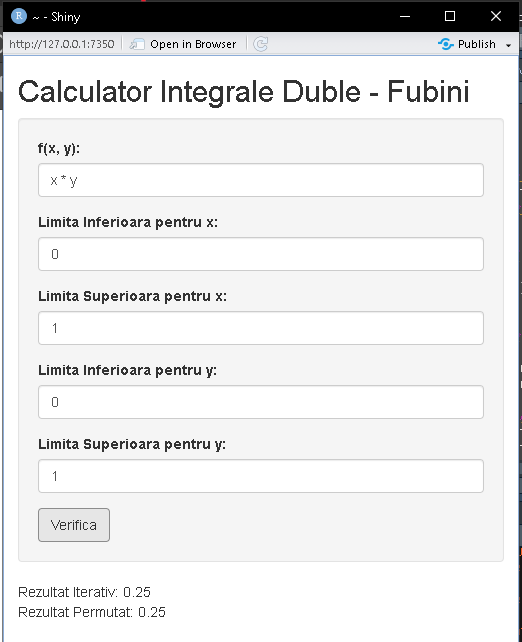
**Densitatea condițională** f(X | Y = y) a variabilei X dat fiind o valoare specifică a variabilei Y reprezintă distribuția probabilistică a variabilei X având în vedere o valoare specifică a variabilei Y.

1. Aplicația verifică posibilitatea aplicării Teoremei lui Fubini pentru calculul integralei duble și va afișa rezultatul în două moduri: iterativ și permutat (ambele rezultate trebuie să fie egale). În metoda iterativă, se calculează mai întâi integrala interioară, apoi cea exterioară. În metoda permutată, se inversează ordinea de integrare. Rezultatele ambelor metode sunt afișate utilizatorului.

Pașii sunt următorii:

* Definirea interfeței (UI-ului) care include elementele pentru introducerea funcției și limitelor de integrare, (în sidebar) și rezultatele (în panoul principal);
* Definirea server-ului logic unde sunt realizate calculul integralei duble prin cele două metode;
* Se verifică dacă utilizatorul a apăsat butonul ”Verifica”;
* Se definește funcția f(x, y) utilizând input-ul introdus de utilizator;
* Se generează un set de x și y variabile aleatoare bidimensionale, utilizând funcția ”**runif**” pentru a genera numere aleatoare uniform distribuite în intervalele specificate de utilizator;
* Pentru metoda iterativă, se definește o rețea regulată de puncte în domeniul de integrare și se evaluează funcția f(x, y) în aceste puncte. Apoi, se însumează valorile funcției în aceste puncte și se înmulțește cu suprafața fiecărui dreptunghi din rețea pentru a obține valoarea aproximativă a integralei;
* Pentru metoda permutată, se procedează ca la iterativă doar că se schimbă ordinea de integrare.

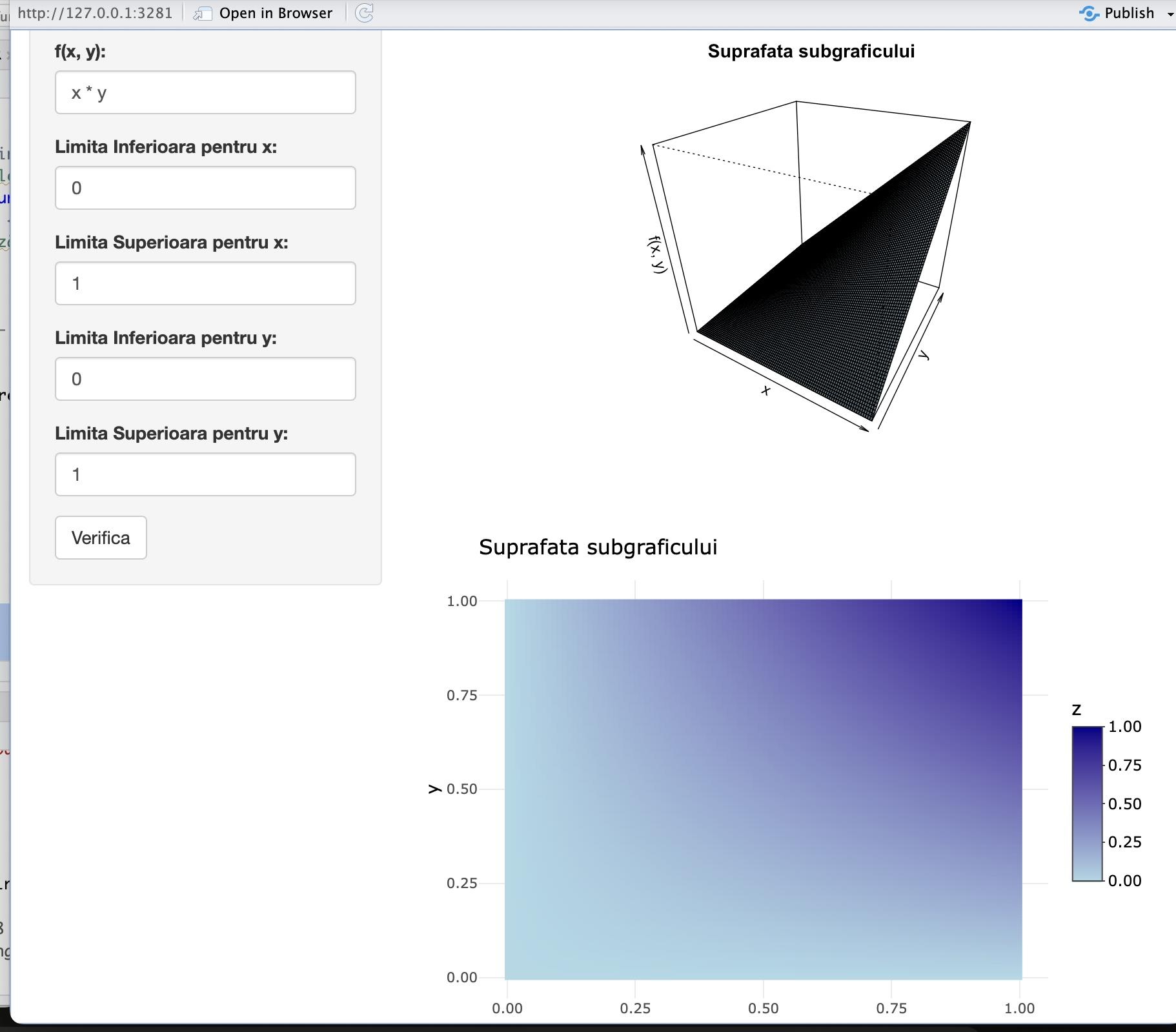
| library(shiny)  library(pracma)  # Definim UI pentru aplicatie  ui <- fluidPage(    # Afiseaza un panou de titlu pentru aplicatie  titlePanel("Calculator Integrale Duble - Fubini"),    # Layout cu panou lateral si principal  sidebarLayout(  sidebarPanel(  # Elementele de intrare pentru utilizator  textInput("functie", "f(x, y):", value = "x \* y"),  numericInput("x\_min", "Limita Inferioara pentru x:", value = 0),  numericInput("x\_max", "Limita Superioara pentru x:", value = 1),  numericInput("y\_min", "Limita Inferioara pentru y:", value = 0),  numericInput("y\_max", "Limita Superioara pentru y:", value = 1),  # Buton pentru declansarea calculului si afisarea rezultatului  actionButton("verifica", "Verifica")  ),  mainPanel(  # Ceea ce se afiseaza  textOutput("rezultat\_iterativ"),  textOutput("rezultat\_permutat")  )  )  )  # Definirea server-ului logic  server <- function(input, output) {    # Iterativa - integrala interioara mai intai  output$rezultat\_iterativ <- renderText({  # Conditia asigura faptul ca se efectueaza calculul doar atunci  # cand butonul "Verifica" e apasat  if (input$verifica > 0) {  # Definirea si evaluarea functiei  f <- function(x, y) {  eval(parse(text = input$functie), list(x = x, y = y))  }    # Generare de variabile aleatoare bidimensionale  nr\_observatii <- 1000  x <- runif(nr\_observatii, min = input$x\_min, max = input$x\_max)  y <- runif(nr\_observatii, min = input$y\_min, max = input$y\_max)    # Calculul integralelor folosind metoda iterativa  x\_values <- seq(input$x\_min, input$x\_max, length.out = 100)  y\_values <- seq(input$y\_min, input$y\_max, length.out = 100)  z\_values <- outer(x\_values, y\_values, Vectorize(function(x, y) f(x, y)))  integrala\_iterativa <- sum(z\_values) \* ((input$x\_max - input$x\_min) / length(x\_values)) \* ((input$y\_max - input$y\_min) / length(y\_values))    paste("Rezultat Iterativ: ", round(integrala\_iterativa, 4))  }  })    # Permutata - ordinea de integrare este inversata  output$rezultat\_permutat <- renderText({  if (input$verifica > 0) {  # Definirea si evaluarea functiei  f <- function(x, y) {  eval(parse(text = input$functie), list(x = x, y = y))  }    # Generare de variabile aleatoare bidimensionale  nr\_observatii <- 1000  x <- runif(nr\_observatii, min = input$x\_min, max = input$x\_max)  y <- runif(nr\_observatii, min = input$y\_min, max = input$y\_max)    # Calculul integralelor folosind metoda permutata  x\_values <- seq(input$x\_min, input$x\_max, length.out = 100)  y\_values <- seq(input$y\_min, input$y\_max, length.out = 100)  z\_permutat <- outer(y\_values, x\_values, Vectorize(function(y, x) f(x, y)))  integrala\_permutata <- sum(z\_permutat) \* ((input$y\_max - input$y\_min) / length(y\_values)) \* ((input$x\_max - input$x\_min) / length(x\_values))    paste("Rezultat Permutat: ", round(integrala\_permutata, 4))  }  })  }  # Rularea aplicatiei Shiny  shinyApp(ui = ui, server = server) |
| --- |



1. Această aplicație permite utilizatorului să introducă o funcție cu 2 parametrii, iar apoi să definească limitele inferioare și superioare pentru x si y. Aplicația va genera și va afișa graficul 3D al suprafeței și reprezentarea acestuia.

* Pentru a utiliza aplicația, utilizatorul trebuie să introducă o funcție f(x, y). Apoi, trebuie să definească limitele inferioare și superioare pentru x și y. După aceea, apăsând butonul "Verifică", aplicația va genera și afișa graficul 3D al suprafeței și reprezentarea bidimensională a acesteia.
* Funcția renderPlot este utilizată pentru a genera graficul 3D al suprafeței, folosind funcția persp, în timp ce funcția renderPlotly este utilizată pentru a genera reprezentarea suprafeței.

| library(shiny)  library(ggplot2)  library(plotly)    ui <- fluidPage(    titlePanel("Interpretarea Geometrică a Integralei Duble"),    sidebarLayout(  sidebarPanel(  #inputul de la utilizator  textInput("functie", "f(x, y):", value = "x \* y"),  numericInput("x\_min", "Limita Inferioara pentru x:", value = 0),  numericInput("x\_max", "Limita Superioara pentru x:", value = 1),  numericInput("y\_min", "Limita Inferioara pentru y:", value = 0),  numericInput("y\_max", "Limita Superioara pentru y:", value = 1),  actionButton("verifica", "Verifica")  ),  mainPanel(  plotOutput("grafic\_3D"),  plotlyOutput("grafic\_suprafata")  )  )  )    server <- function(input, output) {    output$grafic\_3D <- renderPlot({  #verificare ca inputul este bun  if (input$verifica > 0) {  #preluarea functiei de la utilizator si a limitelor  f <- function(x, y) {  eval(parse(text = input$functie), list(x = x, y = y))  }    x\_values <- seq(input$x\_min, input$x\_max, length.out = 100)  y\_values <- seq(input$y\_min, input$y\_max, length.out = 100)  z\_values <- outer(x\_values, y\_values, Vectorize(function(x, y) f(x, y)))  #generarea graficului 3D  persp(x = x\_values, y = y\_values, z = z\_values, theta = 30, phi = 30, col = "lightblue", shade = 0.5,  xlab = "x", ylab = "y", zlab = "f(x, y)", main = "Suprafata subgraficului")  }  })    output$grafic\_suprafata <- renderPlotly({  if (input$verifica > 0) {  f <- function(x, y) {  eval(parse(text = input$functie), list(x = x, y = y))  }    x\_values <- seq(input$x\_min, input$x\_max, length.out = 100)  y\_values <- seq(input$y\_min, input$y\_max, length.out = 100)  z\_values <- outer(x\_values, y\_values, Vectorize(function(x, y) f(x, y)))  #dataframe care contine valorile date x si y și z valoarea functiei in punctele date  df <- data.frame(x = rep(x\_values, each = length(y\_values)),  y = rep(y\_values, times = length(x\_values)),  z = as.vector(z\_values))  # se adauga un strat de tip "tile" peste grafic, care va fi colorat în functie de valorile din #coloana z, si se specifica culoarea conturului si dimensiunea acestuia.  ggplot(df, aes(x = x, y = y, z = z)) +  geom\_tile(aes(fill = z), color = "white", size = 0.5) +  scale\_fill\_gradient(low = "lightblue", high = "darkblue") +  labs(title = "Suprafata subgraficului", x = "x", y = "y", z = "f(x, y)") +  theme\_minimal() +  theme(axis.title = element\_text(size = 10),  axis.text = element\_text(size = 8),  plot.title = element\_text(size = 12))  }  })  }  shinyApp(ui = ui, server = server) |
| --- |

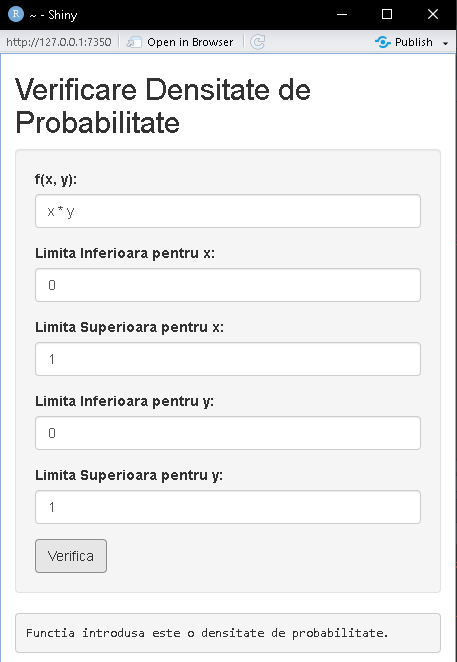
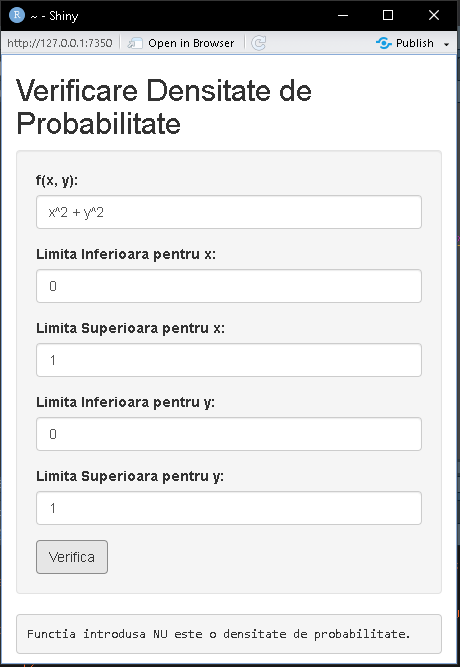


1. Aplicația va verifica dacă funcția introdusă de utilizator reprezintă o densitate de probabiliate. Pentru acesta, utilizăm o funcție ”**verificare\_densitate**” care primește ca argumente funcția f(x, y) și variabilele aleatoare bidimensionale x, y și returnează ”TRUE” dacă f(x, y) este o densitate de probabilitate. În cadrul acestei funcții se verifică dacă evaluarea lui f(x, y) s-a realizat cu succes și dacă valorile rezultate sunt pozitive, finite și în intervalul [0, 1].

Pașii sunt:

* Definirea interfeței (UI-ului) care include elementele pentru introducerea funcției și limitelor de integrare, (în sidebar) și rezultatul (în panoul principal);
* Definirea server-ului logic unde are loc apelul funcției ”**verificare\_densitate**”;
* Definirea și verificarea funcției introdusă de utilizator în câmpul text;
* Se generează un set de x și y variabile aleatoare bidimensionale, utilizând funcția ”**runif**” pentru a genera numere aleatoare uniform distribuite în intervalul specificat de utilizator;
* Se apelează funcția ”**verificare\_densitate**”, iar în funcție de rezultat, se efișează output-ul corespunzător.

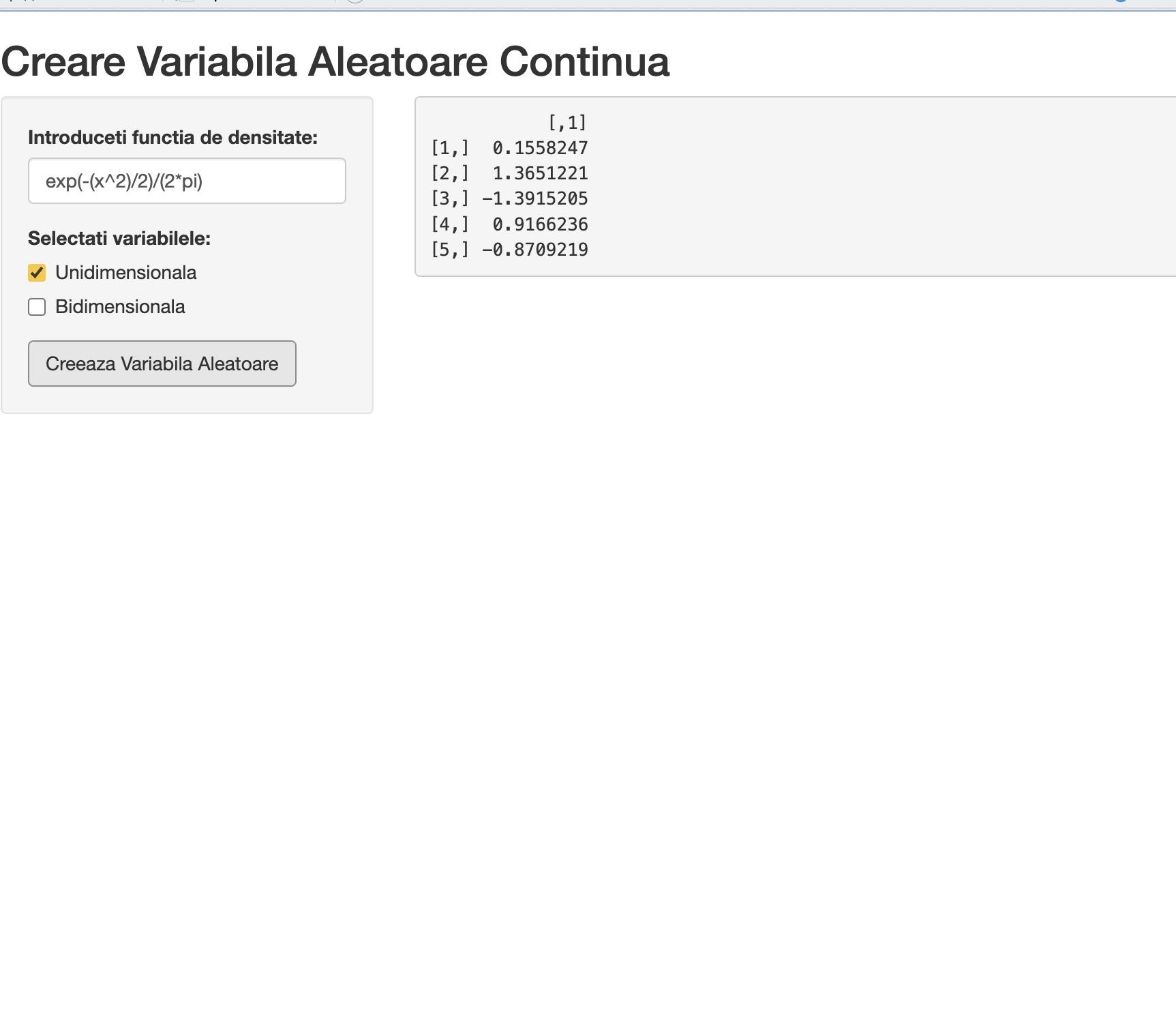
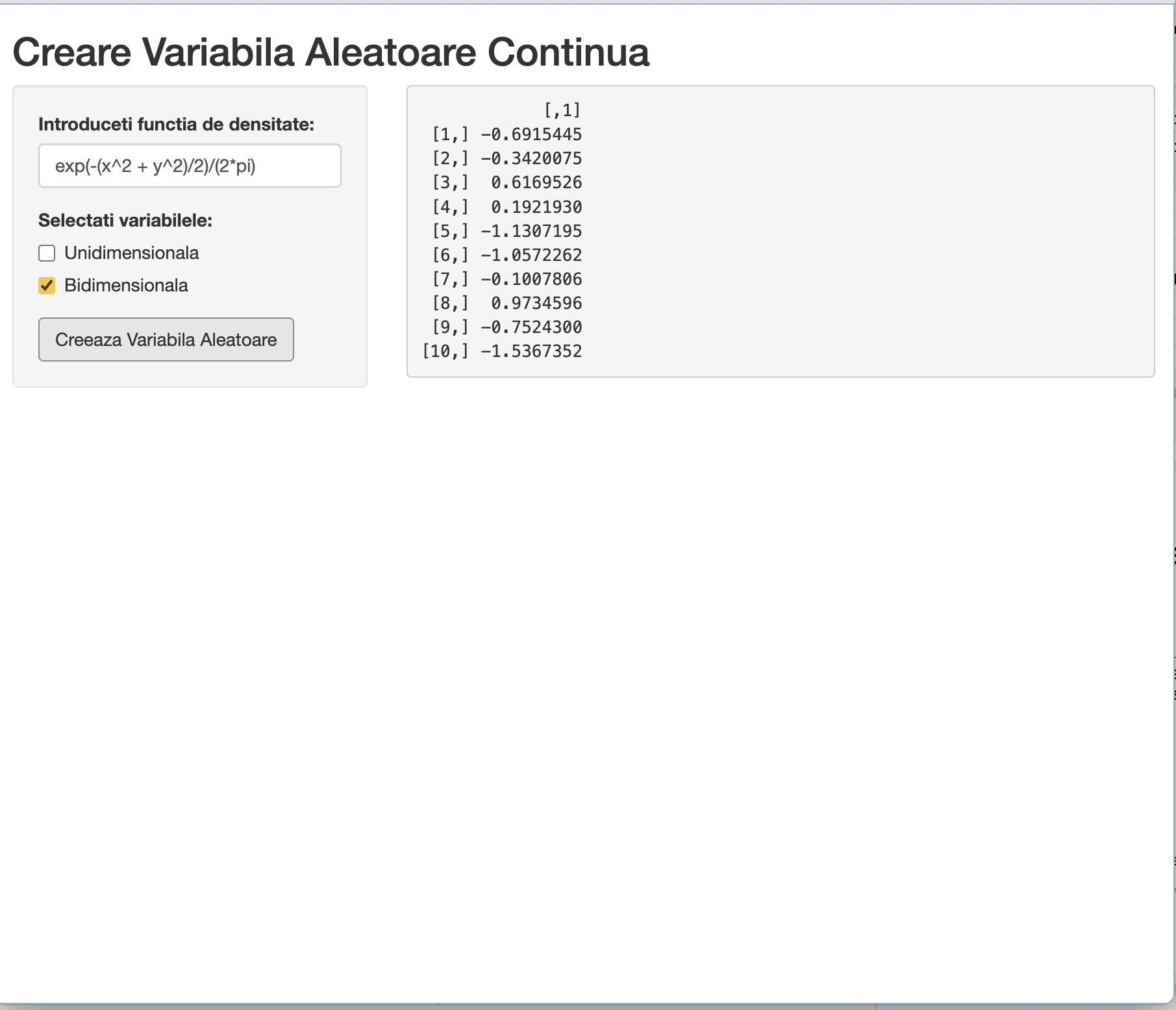
| library(shiny)  library(animate)  # Definirea funnctiei pentru verificarea densitatii  # de probabilitate  verificare\_densitate <- function(f, x, y) {  z <- try(f(x, y), silent = TRUE)    # Verificarea daca rezultatul este o densitate de probabilitate  if (!inherits(z, "try-error") && all(z >= 0) && all(z <= 1) && all(is.finite(z))) {  return(TRUE)  } else {  return(FALSE)  }  }  # Definire UI  ui <- fluidPage(    titlePanel("Verificare Densitate de Probabilitate"),  sidebarLayout(  sidebarPanel(  textInput("functie", "f(x, y):", value = "x \* y"),  numericInput("x\_min", "Limita Inferioara pentru x:", value = 0),  numericInput("x\_max", "Limita Superioara pentru x:", value = 1),  numericInput("y\_min", "Limita Inferioara pentru y:", value = 0),  numericInput("y\_max", "Limita Superioara pentru y:", value = 1),  actionButton("verifica", "Verifica")  ),  mainPanel(  verbatimTextOutput("rezultat")  )  )  )  # Definire server logic  server <- function(input, output) {  # Se activeaza verificarea la apasarea butonului  observeEvent(input$verifica, {  # Definirea si verificarea functiei  f <- function(x, y) {  eval(parse(text = input$functie), list(x = x, y = y))  }    # Generare de variabile aleatoare bidimensionale  nr\_observatii <- 1000  x <- runif(nr\_observatii, min = input$x\_min, max = input$x\_max)  y <- runif(nr\_observatii, min = input$y\_min, max = input$y\_max)    # verificare f densitate de probabilitate  if(verificare\_densitate(f, x, y)) {  output$rezultat <- renderText("Functia introdusa este o densitate de probabilitate.")  } else {  output$rezultat <- renderText("Functia introdusa NU este o densitate de probabilitate.")  }  })  }  # Rulare aplicatie  shinyApp(ui = ui, server = server) |
| --- |



1. Aplicația permite utilizatorului să introducă o funcție de densitate pentru o variabilă aleatoare continuă și să selecteze dacă dorește să creeze o variabilă aleatoare unidimensională sau bidimensională pe baza acelei funcții de densitate.

* Interfața grafică cuprinde un panou lateral în care utilizatorul poate introduce funcția de densitate într-un câmp de text și poate selecta tipul de variabilă aleatoare dorit (unidimensională sau bidimensională). Apoi, utilizatorul poate apăsa pe butonul "Creează Variabilă Aleatoare" pentru a genera variabila aleatoare bazată pe funcția de densitate și opțiunile selectate.
* Funcția de densitate introdusă de utilizator este evaluată. Dacă funcția este validă, se creează variabila aleatoare corespunzătoare pentru fiecare dimensiune selectată.
* Utilizatorul selectează dacă funcția este unidimensională sau bidimensională. Pentru variabilele unidimensionale, este utilizată funcția rnorm() pentru a genera variabile aleatoare normale. Pentru variabilele bidimensionale, este utilizată funcția mvrnorm() din pachetul MASS pentru a genera variabile aleatoare normale multivariate.
* La final rezultatele sunt afișate în panou.

| library(shiny)  library(stats)  library(MASS)  ui <- fluidPage(  titlePanel("Creare Variabila Aleatoare Continua"),    sidebarLayout(  sidebarPanel(  textInput("func\_input", "Introduceti functia de densitate:", "exp(-(x^2 + y^2)/2)/(2\*pi)"),  checkboxGroupInput("variabile", "Selectati variabilele:", c("Unidimensionala", "Bidimensionala")),  actionButton("creare\_variabila\_btn", "Creeaza Variabila Aleatoare")  ),  mainPanel(  verbatimTextOutput("rezultat\_text")  )  )  )  # Definire server  server <- function(input, output) {  observeEvent(input$creare\_variabila\_btn, {  tryCatch({  # Verifica daca functia de densitate introdusa de utilizator este valida  f <- function(x, y) eval(parse(text = input$func\_input))  if (is.function(f)) {  if ("Unidimensionala" %in% input$variabile) {  # Creaza variabila aleatoare unidimensionala  univar <- Vectorize(function(n) rnorm(n, mean = 0, sd = 1))  output$rezultat\_text <- renderPrint({  univar(5) # Afiseaza primele 5 numere  })  }  if ("Bidimensionala" %in% input$variabile) {  # Creaza variabila aleatoare bidimensionala  bivar <- Vectorize(function(n) mvrnorm(n, mu = c(0, 0), Sigma = matrix(c(1, 0.5, 0.5, 1), nrow = 2)))  output$rezultat\_text <- renderPrint({  bivar(5) # Afiseaza primele 5 numere  })  }  } else {  output$rezultat\_text <- renderText("Eroare: Functia de densitate introdusa nu este valida.")  }  }, error = function(e) {  output$rezultat\_text <- renderText("Eroare: Functia introdusa nu este valida.")  })  })  }  shinyApp(ui, server) |
| --- |

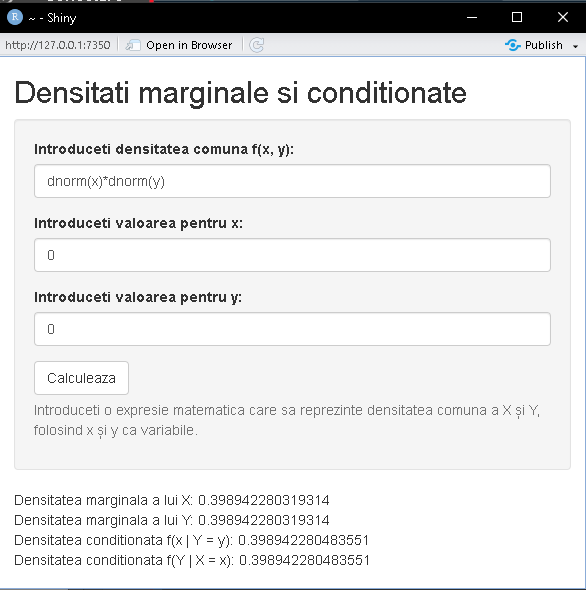


e) Aplicația permite utilizatorului să introducă o expresie matematică reprezentând densitatea comună a variabilelor X, Y și să calculeze densitățile marginale și cele condiționate pentru anumite valori ale acestor variabile. Funcțiile pentru calculul densității marginale a lui x (”**dens\_marg\_x**”), pentru y (”**dens\_marg\_y**”), calculul densitatii conditionate f(X | Y = y) (”**dens\_cond\_x**”) și f(Y | X = x) (”**dens\_cond\_y**”) sunt create la începutul programuli pentru a fi apelate mai târziu, în server.

Sunt efectuați următorii pași:

* Definirea interfeței (UI-ului) care include elementele pentru introducerea densității comune, valorile lui x, y (în sidebar) și rezultatele (în panoul principal);
* Definirea server-ului logic unde are loc apelul funcților precizate anterior;
* Se verifică și se preia expresia pentru densitatea comună;
* Se preiau valorile introduse pentru variabilele X și Y;
* Se calculează densitățile folosind funcțiile definite anterior;

| library(shiny)  # Functie pentru calculul densitatii marginale a lui X  dens\_marg\_x <- function(densitate, x) {  integrate(function(y) densitate(x, y), lower = -Inf, upper = Inf)$value  }  # Functie pentru calculul densitatii marginale a lui Y  dens\_marg\_y <- function(densitate, y) {  integrate(function(x) densitate(x, y), lower = -Inf, upper = Inf)$value  }  # Functie pentru calculul densitatii conditionate f(x|Y=y)  dens\_cond\_x <- function(densitate, x, y) {  densitate(x, y) / dens\_marg\_y(densitate, y)  }  # Functie pentru calculul densitatii conditionate f(Y|X=x)  dens\_cond\_y <- function(densitate, x, y) {  densitate(x, y) / dens\_marg\_x(densitate, x)  }  # UI  ui <- fluidPage(  titlePanel("Densitati marginale si conditionate"),  sidebarLayout(  sidebarPanel(  textInput("densityFunction", "Introduceti densitatea comuna f(x, y):", "dnorm(x)\*dnorm(y)"),  numericInput("x\_value", "Introduceti valoarea pentru x:", value = 0),  numericInput("y\_value", "Introduceti valoarea pentru y:", value = 0),  actionButton("goButton", "Calculeaza"),  br(),  helpText("Introduceti o expresie matematica care sa reprezinte densitatea comuna a X și Y, folosind x și y ca variabile.")  ),  mainPanel(  textOutput("marginalX"),  textOutput("marginalY"),  textOutput("conditionalX"),  textOutput("conditionalY")  )  )  )  # Server  server <- function(input, output) {    observeEvent(input$goButton, {  req(input$densityFunction)    # Definirea densitatii comune f(x, y)  densitate <- function(x, y) {  eval(parse(text = input$densityFunction))  }    x <- input$x\_value  y <- input$y\_value    # Calcul densitatea marginala a lui X  marginal\_x <- dens\_marg\_x(densitate, x)    # Calcul densitatea marginala a lui Y  marginal\_y <- dens\_marg\_y(densitate, y)    # Calcul densitatea conditionata f(x|Y=y)  conditional\_x <- dens\_cond\_x(densitate, x, y)    # Calcul densitatea conditionata f(Y|X=x)  conditional\_y <- dens\_cond\_y(densitate, x, y)    output$marginalX <- renderText({  paste("Densitatea marginala a lui X:", marginal\_x)  })    output$marginalY <- renderText({  paste("Densitatea marginala a lui Y:", marginal\_y)  })    output$conditionalX <- renderText({  paste("Densitatea conditionata f(x | Y = y):", conditional\_x)  })    output$conditionalY <- renderText({  paste("Densitatea conditionata f(Y | X = x):", conditional\_y)  })  })  }  shinyApp(ui, server) |
| --- |



h)Aceasta aplicație permite utilizatorului să introducă o funcție g(x) i să calculeze media și dispersia acesteia pentru o variabilă aleatoare X. Am considerat ca X are densitatea de probabilitate standard. Se creează un panou pentru introducerea datelor de către utilizator.

* Apoi utilizatorul apasă pe buton pentru a efectua calculul.
* Funcția calc\_medie\_si\_var primește funcția g(x) ca argument și calculează media și dispersia lui g(X) folosind integrale simple, deoarece X este o variabilă aleatoare unidimensională cu densitatea de probabilitate standard normală.
* Atunci când utilizatorul apasă butonul "Calculează", funcția observeEvent inițiază calculul și afișează rezultatele pe interfață.

| library(shiny)    #crearea panoului pentru introducerea datelor  ui <- fluidPage(  titlePanel("Calculator pentru media si dispersia lui g(X)"),  sidebarLayout(  sidebarPanel(  textInput("g\_function", "Introduceti functia g(x):", "x^2"),  actionButton("calculate", "Calculeaza"),  tags$hr(),  h4("Rezultate:"),  verbatimTextOutput("medie\_gX\_output"),  verbatimTextOutput("var\_gX\_output")  ),  mainPanel()  )  )    # Server pentru a calcula  server <- function(input, output) {  calc\_medie\_si\_var <- function(g\_function) {  medie\_gX <- integrate(function(x) eval(parse(text = g\_function)) \* dnorm(x), lower = -Inf, upper = Inf)$value  #calculam media si media patratului  medie\_gX\_sq <- integrate(function(x) (eval(parse(text = g\_function)))^2 \* dnorm(x), lower = -Inf, upper = Inf)$value    var\_gX <- medie\_gX\_sq - medie\_gX^2    return(list(medie\_gX = medie\_gX, var\_gX = var\_gX))  }  #initiaza calculul si afisarea rezultatului  observeEvent(input$calculate, {  results <- calc\_medie\_si\_var(input$g\_function)    output$medie\_gX\_output <- renderPrint(paste("Media lui g(X):", results$medie\_gX))  output$var\_gX\_output <- renderPrint(paste("Dispersia lui g(X):", results$var\_gX))  })  }  shinyApp(ui, server) |
| --- |



## Probleme înâmpinate

**Exercițiul 1**

**La punctul a)**, a fost dificil să ne dam seama care ar fi cea mai bună metodă de a lăsa tabelul incomplet pe anumite poziții, iar una dintre problemele pe care ni le-am pus a fost: care este numărul maxim de probabiliăți necompletate pe care îl putem lăsa? Am ajuns la n+m-1, dar în acest caz completarea de la **punctul b)** era mult mai dificilă deoarece ajungeam să rezolvăm ecuații care erau greu de rezolvat la implemetare. Varianta finală a rămas cu câte o probabilitate lipsă pe linie, la punctul b) fiind necesară doar suma liniei fără probabilitatea lipsă și valorile marginale.

**La punctul d)**, calculul covarianței returna o valoare greșită datorită numerelor floating point. Am rezolvat problema înmulțind rezultatul cu 10^4, trunchiindu-l și împărțindu-l la 10^4. Astfel am fixat numărul de zecimale la 4, iar un rezultat care era afișat sub formă științifică datorită zecimalelor a devenit un întreg.

Cod și rezultat inițial:

cova <- fmedie(XY)

covb <- fmedie(X) \* fmedie(Y)

covXY <- cova – covb

Cod după modificare:

cova <- fmedie(XY)

covb <- fmedie(X) \* fmedie(Y)

covXY <- trunc((cova - covb)\*10^4)/10^4

**La punctul f)** (calculul unor probabilități pe baza repartiției comune), am început prin a scrie doar cazul în care se dă o valoare doar pentru X sau doar pentru Y. În aceste cazuri am preluat direct semnul de comparație pentru variabila specificată.

Când am implementat cazul în care atât X, cât și Y au valori în care trebuie să se încadreze, am realizat că ar trebui să scriu 9 if-uri dacă aș fi folosit aceeași metodă de verificat condițiile. Am căutat cum se poate parsa o expresie dintr-un string și am obținut un singur if care evaluează un string construit în funcție de argumentele date.

Prima metodă de verificat semnul:

if(compY=="<")

{ ... }

else if(compY==">")

{ ... }

else if(compY=="==")

{ ... }

A doua metodă (cea cu parsat): (o celulă din matrice + argumentele compX și intX separate prin spațiu)

condX <- paste("matr[i,1]",paste(compX, intX, sep = " "),sep = " ")

condY <- paste("matr[1,j]",paste(compY, intY, sep = " "),sep = " ")

if(eval(parse(text = condX)) && eval(parse(text = condY)))

**Exercițiul 2**

**La punctul d**), ne-am confruntat cu provocarea de a genera variabile aleatoare corespunzător după crearea unui obiect de tip valoare aleatoare continuă. Inițial neclară, soluția a venit prin utilizarea funcțiilor rnorm și mvrnorm, care s-au dovedit a fi extrem de utile în acest context.

**Pentru punctul h)**, am întâmpinat dificultăți în stabilirea tipului exact de repartiție al lui X. Pentru a depăși această incertitudine, am ales să folosim repartiția standard, permițându-ne astfel să calculăm cu acuratețe media și dispersia. Această abordare, deși inițial neașteptată, ne-a facilitat înțelegerea și aplicarea conceptelor într-un mod practic.

## Concluzii

Pe parcursul acestui proiect, ne-am angajat într-o explorare profundă a aspectelor complexe ale teoriei probabilităților și statisticii, concentrându-ne pe analiza relațiilor dintre variabile aleatoare. Deși am întâmpinat provocări, în special în înțelegerea și aplicarea conceptelor precum covarianța, probabilitatea condiționată, independența și densitatea marginală, am perseverat le-am abordat cu o gândire analitică pentru a le depăși. În timpul proiectului, am dedicat resurse semnificative pentru a înțelege și a lucra cu pachetul Shiny în R. Acest pachet ne-a permis să dezvoltăm aplicații interactive și dinamice pentru vizualizarea și explorarea datelor. Deși inițial am întâmpinat dificultăți în înțelegerea funcționalităților complexe ale pachetului și în folosirea funcțiilor în R, am investit timp și efort pentru a ne familiariza cu acestea, ceea ce ne-a adus beneficii semnificative, contribuind la consolidarea cunoștințelor noastre în domeniu.

Prin acest proces, am dobândit o înțelegere mai profundă a materiei și o perspectivă valoroasă asupra puterii analizei statistice în interpretarea datelor.