*Proiect Probabilitati si Statistica*

*Facultatea de Matematica si Informatica*

*Milosi Elias - Craciun Florin*

*Seria 24*

*Grupa 241*

*Descrierea Problemei*

Realizarea unui proiect in limbajul de programare “R” ce foloseste pachete cum ar fi “shiny” sau “animate” pentru realizarea cerintelor ce calculeaza probabilitati,sau creaza diferite functii pentru a calcula medii,variante,covariante,probabilitati.

Cerintele implementeaza teoria de la laborator pentru variabile aleatoare uni si bidimensionale.

*Implementare*

Pachete software:

-shiny

-animate

-stats

-animation

-base

S-a folosit R studio pentru implementare.

Surse de inspiratie:

Geek for geeks

Stack overflow

Rdocumentation

Alexamarioarei.quarto.pub

Masini folosite:

MacOS 16GB Ram

Linux Fedora 16GB Ram

*Limitari*

Nu s-a reusit implementarea functiei care creaza un tabel cu repartitia comuna pentru 2 variabile x si y de dimensiuni OARECARE.

Probleme ce au ramas deschise:

Realizarea functiei care aplica proprietatile de covarianta

-functia care verifica daca variablilele sunt necorelate

-reprezentarea 3d pentru repartitia comuna a lui X , Y si Z

-unele subpuncte testeaza pentru o functie hardcodata

*Task-urile Proiectului*

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.

#\* am calculat numarul maxim de valori necunoscute cu formula (m + n - 1)

#\* am reusit doar pentru cazul (3, 2)

#\* generez mai intai repartitiile marginale pentru X si Y si valorile pi pentru repartitia comuna

#\* apoi generez pozitii random in matrice pe care sa le fac necunoscute

frepcomgen <- function(n, m) {

lim\_inf <- -(n+m)

lim\_sup <- (n+m)

missing\_values <- m + n - 1 # destule valori necunoscute

x\_values <- sample(lim\_inf:lim\_sup, n, replace=FALSE)

y\_values <- sample(lim\_inf:lim\_sup, m, replace=FALSE)

p\_values <- c(0.25, 0.4, 0.35)

q\_values <- c(0.35, 0.65)

pi\_values <- matrix(c(0.15, 0.05, 0.15, 0.10, 0.35, 0.20), nrow=n, ncol=m)

while (missing\_values > 0) {

i <- sample(1:(n+1), 1)

j <- sample(1:(m+1), 1)

if (i <= n && j <= m) {

if (is.na(pi\_values[i,j])) next

missing\_values <- missing\_values - 1

pi\_values[i,j] <- NA

next

}

if (i <= n) {

if (is.na(p\_values[i])) next

missing\_values <- missing\_values - 1

p\_values[i] <- NA

next

}

if (j <= m) {

if (is.na(q\_values[j])) next

missing\_values <- missing\_values - 1

q\_values[j] <- NA

}

}

# print(pi\_values)

pi\_values <- cbind(pi\_values, p\_values)

pi\_values <- rbind(pi\_values, c(q\_values, 1))

# print(pi\_values)

rez <- data.frame(pi\_values)

colnames(rez) <- NULL

rownames(rez) <- NULL

colnames(rez) <- c(y\_values, 'p')

rownames(rez) <- c(x\_values, 'q')

return(rez)

}

2. Construiţi o funcţie fcomplrepcom care completează repartiţia comună generată la punctul

anterior(pentru cazul particular sau pentru cazul general)

fcomplrepcom <- function() {

# Variabila ok are valoarea 1 atat timp cat mai exista valori necunoscute

ok <<- 1

while (ok != 0) {

# for pe linii

for (i in (n+1):1) {

ok <<- 0

# verific daca exista valoare necunoscuta pe linie

na\_values = sum(is.na(tabel[i,]))

# verific cate valori necunoscute exista pe linie pentru variabila ok

# daca sunt mai multe atunci nu le pot afla dintr-un calcul

# calculez doar daca exista o necunoscuta

if (na\_values >= 1) ok <<- 1

if (na\_values == 1) {

if (is.na(tabel[i,3])) tabel[i,3] <<- tabel[i,1] + tabel[i,2]

else {

if (is.na(tabel[i,1])) tabel[i,1] <<- tabel[i,3] - tabel[i,2]

else tabel[i,2] <<- tabel[i,3] - tabel[i,1]

}

}

}

ok <<- 0

# for pe coloane

for (j in (m+1):1) {

na\_values = sum(is.na(tabel[,j]))

if (na\_values >= 1) ok <<- 1

if (na\_values == 1) {

if (is.na(tabel[4,j])) tabel[4,j] <<- sum(tabel[1:3,j])

else {

if (is.na(tabel[1,j])) tabel[1,j] <<- tabel[4,j] - sum(tabel[2:3, j])

else if (is.na(tabel[2,j])) tabel[2,j] <<- tabel[4,j] - (tabel[1,j] + tabel[3,j])

else tabel[3,j] <<- tabel[4,j] - sum(tabel[1:2, j])

}

}

}

}

}

3. Construiți o funcție frepmarginal care construiește repartițiile marginale pentru X și Y

pornind de la repartiția lor comună.

# construieste o repartitie marginala cu valorile date

# se apeleaza frepmarginal(n, as.numeric(rownames(tabel)[1:3]), as.numeric(tabel[1:3,3]))

# functia construieste o matrice de 2 linii reprezentand reparitia marginala

frepmarginal <- function(k, X, p) {

temp <<- matrix(nrow=0, ncol=k)

temp <<- rbind(temp, X)

temp <<- rbind(temp, p)

return(temp)

}

4. Construiți o funcție fpropcov care aplică proprietățile covarianței pentru calculul acesteia

pentru v.a.Z aX bY= + și respectivT cX dY= + considerȃnd că toate informațiile necesare

despre X și Y sunt date de intrare.

5. Construiți o funcție fPcond care calculează probabilitatea condiționată pentru v.a. X și Y

pornind de la repartiția comună.

# functie care calculeaza P(numeVa1 | numeVa2 = val)

fPcond <- function(numeVa1, numeVa2, val) {

if (numeVa1 == "X") {

temp <- rep\_marg\_x

for (i in 1:n) {

temp[2,i] <- round(tabel[i,val] / tabel[(n+1),val], 2)

}

return(temp)

} else {

temp <- rep\_marg\_y

for (i in 1:m) {

temp[2, i] <- round(tabel[val,i] / tabel[val,(m+1)], 2)

}

return(temp)

}

}

6. Construiți o funcție fPcomun care calculează o probabilitate legată de perechea (X,Y)

pornind de la repartiția comună.

# calculeaza o probabilitate pentru (X, Y)

# exemplu: fPcomun("X", ">", 0.2, "Y", "<", 1.7) calculeaza P(X > 0.2, Y < 1.7)

fPcomun <- function(numeVa1, semn1, val1, numeVa2, semn2, val2) {

X <- rep\_marg\_x

Y <- rep\_marg\_y

rez <- 0

if (numeVa1 == "X") {

for (i in 1:n) {

for (j in 1:m) {

# pentru P(X > 0.2, Y < 1.7) verifica daca xi > 0.2 si yj < 1.7

if (eval(parse(text=paste(rownames(tabel)[i], semn1, val1))) && eval(parse(text=paste(colnames(tabel)[j], semn2, val2)))) {

# print(tabel[i,j])

rez <- rez + tabel[i,j]

}

}

}

} else {

for (i in 1:m) {

for (j in 1:n) {

if (eval(parse(text=paste(rownames(tabel)[i], semn1, val1))) && eval(parse(text=paste(colnames(tabel)[j], semn2, val2)))) {

rez <- rez + tabel[i,j]

}

}

}

}

return(rez)

}

7. 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<X<0.8|Y>0.3)

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

8. 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

# verifica daca repartitia comuna data este independenta prin relatia pi(i, j) = p(i) \* q(j)

fverind <- function() {

for (i in 1:n) {

for (j in 1:m) {

if (tabel[i, j] != (tabel[i, (m+1)] \* tabel[(n+1), i])) {

return(0)

}

}

}

return(1)

}

Incercare 1 a) pentru cazul general # incercarea sa p, q si pi random, dar nu am reusit sa generez pi a.i suma(linia i) = pi

# si suma(coloana i) = qi pentru x si y dependente

incercare <- function() {

pi\_values <- matrix(runif(m\*n, min=0.01, max=min(c(p\_values, q\_values))/2), nrow = n, ncol = m)

pi\_values <- round(pi\_values \* (1 / sum(min(q\_values, p\_values))), 2)

sp <- 0.99

sq <- 0.99

while(sp != 1) {

p\_values <- runif(n, min=0.01, max=1)

p\_values <- round(p\_values \* (1 / sum(p\_values)), 2) # sa aiba suma 1, cu 2 zecimale

sp <- sum(p\_values)

}

while(sq != 1) {

q\_values <- runif(m, min=0.01, max=1)

q\_values <- round(q\_values \* (1 / sum(q\_values)), 2)

sq <- sum(q\_values)

}

}

Partea a-2a

Folosind pachetele R shiny(https://shiny.rstudio.com/), animate(https://cran.r-

project.org/web/packages/animate/vignettes/introduction.html) 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:

a) 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.

library(shiny)

library(animate)

# Definirea funcției pentru calculul integralei duble

calculeaza\_integrala\_dubla <- function(f, a, b, c, d) {

integrare <- integrate(function(y) {

sapply(y, function(yy) integrate(function(x) f(x, yy), c, d)$value)

}, a, b, subdivisions = 100)$value

mesaj <- if (is.finite(integrare)) {

"Teorema lui Fubini poate fi aplicată. Rezultatul integraliei este:"

} else {

"Teorema lui Fubini nu poate fi aplicată pentru această funcție."

}

rezultat <- list(integrare = integrare, mesaj = mesaj)

return(rezultat)

}

# Interfața Shiny

ui <- fluidPage(

titlePanel("Verificarea Teoremei lui Fubini pentru integrale duble"),

sidebarLayout(

sidebarPanel(

textInput("functie", "Introdu funcția (ex: x^2 + y^2)"),

numericInput("a", "Limita inferioară pentru x:", 0),

numericInput("b", "Limita superioară pentru x:", 1),

numericInput("c", "Limita inferioară pentru y:", 0),

numericInput("d", "Limita superioară pentru y:", 1),

actionButton("calculeaza", "Calculează")

),

mainPanel(

textOutput("rezultat\_text")

)

)

)

# Server logic

server <- function(input, output) {

observeEvent(input$calculeaza, {

tryCatch({

# Convertirea string-ului introdus într-o funcție R

f <- function(x, y) eval(parse(text = input$functie))

# Calculul integraliei duble și afișarea rezultatului

rezultat <- calculeaza\_integrala\_dubla(f, input$a, input$b, input$c, input$d)

output$rezultat\_text <- renderText({

paste0(rezultat$mesaj, " ", rezultat$integrare)

})

}, error = function(e) {

output$rezultat\_text <- renderText({

paste("Eroare:", e$message)

})

})

})

}

shinyApp(ui, server)

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

de probabilitate.

library(shiny)

library(animate)

calculeaza\_integrala\_dubla <- function(f, a, b, c, d) {

integrare <- integrate(function(y) {

sapply(y, function(yy) integrate(function(x) f(x, yy), c, d)$value)

}, a, b, subdivisions = 100)$value

mesaj <- if (is.finite(integrare)) {

"Teorema lui Fubini poate fi aplicată. Rezultatul integraliei este:"

} else {

"Teorema lui Fubini nu poate fi aplicată pentru această funcție."

}

rezultat <- list(integrare = integrare, mesaj = mesaj)

return(rezultat)

}

# Interfața Shiny

ui <- fluidPage(

titlePanel("Verificarea densității de probabilitate"),

sidebarLayout(

sidebarPanel(

textInput("functie", "Introdu funcția (ex: exp(-x^2 - y^2))"),

numericInput("a", "Limita inferioară pentru x:", 0),

numericInput("b", "Limita superioară pentru x:", 1),

numericInput("c", "Limita inferioară pentru y:", 0),

numericInput("d", "Limita superioară pentru y:", 1),

actionButton("verifica", "Verifică")

),

mainPanel(

textOutput("rezultat\_text")

)

)

)

# Server logic

server <- function(input, output) {

observeEvent(input$verifica, {

tryCatch({

# Convertirea string-ului introdus într-o funcție R

f <- function(x, y) eval(parse(text = input$functie))

# Calculul integraliei duble și afișarea rezultatului

volum <- calculeaza\_integrala\_dubla(f, input$a, input$b, input$c, input$d)

output$rezultat\_text <- renderText({

paste0(volum$mesaj, " ", volum$integrare)

})

# Verificarea condițiilor pentru densitatea de probabilitate

conditie1 <- all(sapply(seq(0, 1, length.out = 100), function(y) all(f(seq(0, 1, length.out = 100), y) >= 0)))

conditie2 <- volum$integrare == 1

# Afișarea rezultatului

rezultat\_text <- if (conditie1 && conditie2) {

"Funcția este o densitate de probabilitate."

} else {

"Funcția NU este o densitate de probabilitate."

}

output$rezultat\_text <- renderText({

rezultat\_text

})

# Salvarea animației într-un document LaTeX

#output$grafic <- renderUI({

# saveLatex({

# ani.options(interval = 0.1)

# curve3d(f, xlim = c(input$a, input$b), ylim = c(input$c, input$d), col = "blue", theta = 30, phi = 30)

# }, interval = 0.1, movie.name = "densitate.tex", ani.width = 400, ani.height = 400)

# tags$iframe(src = "densitate.tex", height = 400, width = 400)

#})

}, error = function(e) {

output$rezultat\_text <- renderText({

paste("Eroare:", e$message)

})

})

})

}

# Rulează aplicația Shiny

shinyApp(ui, server)

e) Construirea densităților marginale și a celor condiționate pornind de la densitatea

comună f(x,y) a două v.a. unidimensionale X și Y.

#LIMITARE: creaza grafice pentrtu functia scrisa de noi la densitatea\_comuna

# Încărcare librării

library(shiny)

# Densitatea comună

densitate\_comuna <- function(x, y) {

exp(-x^2 - y^2)

}

# Densitatea marginală a lui X

densitate\_marginala\_X <- function(x) {

integrate(function(y) densitate\_comuna(x, y), lower = -Inf, upper = Inf)$value

}

# Densitatea marginală a lui Y

densitate\_marginala\_Y <- function(y) {

integrate(function(x) densitate\_comuna(x, y), lower = -Inf, upper = Inf)$value

}

# Densitatea condiționată a lui Y dat fiind X = x

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

densitate\_comuna(x, y) / densitate\_marginala\_X(x)

}

# Interfața Shiny

ui <- fluidPage(

titlePanel("Construirea densităților marginale și condiționate"),

sidebarLayout(

sidebarPanel(

),

mainPanel(

plotOutput("plot\_marginala\_X"),

plotOutput("plot\_marginala\_Y"),

plotOutput("plot\_conditionata\_Y\_X")

)

)

)

# Server logic

server <- function(input, output) {

output$plot\_marginala\_X <- renderPlot({

x\_vals <- seq(-3, 3, length.out = 100)

y\_vals <- sapply(x\_vals, function(x) densitate\_marginala\_X(x))

plot(x\_vals, y\_vals, type = "l", col = "blue", xlab = "X", ylab = "Densitate marginală a lui X")

})

output$plot\_marginala\_Y <- renderPlot({

y\_vals <- seq(-3, 3, length.out = 100)

x\_vals <- sapply(y\_vals, function(y) densitate\_marginala\_Y(y))

plot(y\_vals, x\_vals, type = "l", col = "red", xlab = "Y", ylab = "Densitate marginală a lui Y")

})

output$plot\_conditionata\_Y\_X <- renderPlot({

y\_vals <- seq(-3, 3, length.out = 100)

x\_val <- input$x\_val

densitati\_conditionate <- sapply(y\_vals, function(y) densitate\_conditionata\_Y\_X(y, x\_val))

plot(y\_vals, densitati\_conditionate, type = "l", col = "green", xlab = "Y", ylab = "Densitate condiționată a lui Y dat fiind X = x")

})

}

# Rulează aplicația Shiny

shinyApp(ui, server)

f) 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 obținepunctaj suplimentar dacă se realizează o animație care să pună ȋn valoare modificarea

funcției reprezentate la schimbarea parametrilor repartiției.

# Încărcare librării

library(shiny)

library(stats)

library(animate)

# Funcție de densitate de probabilitate - Exemplu: Distribuție normală unidimensională

densitate\_probabilitate\_unidimensionala <- function(x, mu, sigma) {

dnorm(x, mean = mu, sd = sigma)

}

# Funcție de densitate de probabilitate - Exemplu: Distribuție normală bidimensională

densitate\_probabilitate\_bidimensionala <- function(x, y, mu\_x, mu\_y, sigma\_x, sigma\_y) {

(1 / (2 \* pi \* sigma\_x \* sigma\_y)) \* exp(-0.5 \* ((x - mu\_x)^2 / sigma\_x^2 + (y - mu\_y)^2 / sigma\_y^2))

}

# Funcție de repartiție - Exemplu: Repartiție normală unidimensională

functie\_repartitie\_unidimensionala <- function(x, mu, sigma) {

pnorm(x, mean = mu, sd = sigma)

}

# Funcție de repartiție - Exemplu: Repartiție normală bidimensionă

functie\_repartitie\_bidimensionala <- function(x, y, mu\_x, mu\_y, sigma\_x, sigma\_y) {

pnorm(x, mean = mu\_x, sd = sigma\_x) \* pnorm(y, mean = mu\_y, sd = sigma\_y)

}

# Interfața Shiny

ui <- fluidPage(

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

sidebarLayout(

sidebarPanel(

selectInput("dimensiune", "Dimensiune variabilă aleatoare:", c("Unidimensională", "Bidimensională")),

numericInput("mu", "Medie (mu):", 0),

numericInput("sigma", "Deviație standard (sigma):", 1),

numericInput("mu\_x", "Medie X:", 0),

numericInput("mu\_y", "Medie Y:", 0),

numericInput("sigma\_x", "Deviație standard X:", 1),

numericInput("sigma\_y", "Deviație standard Y:", 1),

sliderInput("parametru", "Valoare parametru:", min = 0, max = 5, value = 2, step = 0.1),

actionButton("reprezinta", "Reprezintă")

),

mainPanel(

plotOutput("plot\_densitate"),

plotOutput("plot\_repartitie"),

uiOutput("animatie")

)

)

)

# Server logic

server <- function(input, output) {

output$plot\_densitate <- renderPlot({

parametru <- input$parametru

if (input$dimensiune == "Unidimensională") {

x\_vals <- seq(-5, 5, length.out = 100)

y\_vals <- densitate\_probabilitate\_unidimensionala(x\_vals, input$mu, input$sigma)

plot(x\_vals, y\_vals, type = "l", col = "blue", xlab = "X", ylab = "Densitate de probabilitate")

abline(v = parametru, col = "red", lty = 2)

} else {

x\_vals <- seq(-5, 5, length.out = 100)

y\_vals <- seq(-5, 5, length.out = 100)

z\_vals <- matrix(0, nrow = length(x\_vals), ncol = length(y\_vals))

for (i in 1:length(x\_vals)) {

for (j in 1:length(y\_vals)) {

z\_vals[i, j] <- densitate\_probabilitate\_bidimensionala(x\_vals[i], y\_vals[j], input$mu\_x, input$mu\_y, input$sigma\_x, input$sigma\_y)

}

}

persp(x\_vals, y\_vals, z\_vals, col = "lightblue", theta = 30, phi = 30, xlab = "X", ylab = "Y", zlab = "Densitate de probabilitate")

lines(c(parametru, parametru), c(-5, 5), c(0, 0), col = "red", lty = 2)

}

})

output$plot\_repartitie <- renderPlot({

parametru <- input$parametru

if (input$dimensiune == "Unidimensională") {

x\_vals <- seq(-5, 5, length.out = 100)

y\_vals <- functie\_repartitie\_unidimensionala(x\_vals, input$mu, input$sigma)

plot(x\_vals, y\_vals, type = "l", col = "green", xlab = "X", ylab = "Funcție de repartiție")

abline(v = parametru, col = "red", lty = 2)

} else {

x\_vals <- seq(-5, 5, length.out = 100)

y\_vals <- seq(-5, 5, length.out = 100)

z\_vals <- matrix(0, nrow = length(x\_vals), ncol = length(y\_vals))

for (i in 1:length(x\_vals)) {

for (j in 1:length(y\_vals)) {

z\_vals[i, j] <- functie\_repartitie\_bidimensionala(x\_vals[i], y\_vals[j], input$mu\_x, input$mu\_y, input$sigma\_x, input$sigma\_y)

}

}

persp(x\_vals, y\_vals, z\_vals, col = "lightgreen", theta = 30, phi = 30, xlab = "X", ylab = "Y", zlab = "Funcție de repartiție")

lines(c(parametru, parametru), c(-5, 5), c(0, 0), col = "red", lty = 2)

}

})

output$animatie <- renderUI({

if (input$dimensiune == "Unidimensională") {

sliderInput("parametru\_animatie", "Valoare parametru (animație):", min = 0, max = 5, value = 2, step = 0.1)

} else {

sliderInput("parametru\_animatie", "Valoare parametru (animație):", min = -5, max = 5, value = 0, step = 0.1)

}

})

observeEvent(input$reprezinta, {

parametre\_animatie <- seq(0, input$parametru\_animatie, length.out = 100)

saveLatex({

for (param in parametre\_animatie) {

if (input$dimensiune == "Unidimensională") {

x\_vals <- seq(-5, 5, length.out = 100)

y\_vals <- densitate\_probabilitate\_unidimensionala(x\_vals, input$mu, input$sigma)

plot(x\_vals, y\_vals, type = "l", col = "blue", xlab = "X", ylab = "Densitate de probabilitate")

abline(v = param, col = "red", lty = 2)

} else {

x\_vals <- seq(-5, 5, length.out = 100)

y\_vals <- seq(-5, 5, length.out = 100)

z\_vals <- matrix(0, nrow = length(x\_vals), ncol = length(y\_vals))

for (i in 1:length(x\_vals)) {

for (j in 1:length(y\_vals)) {

z\_vals[i, j] <- densitate\_probabilitate\_bidimensionala(x\_vals[i], y\_vals[j], input$mu\_x, input$mu\_y, input$sigma\_x, input$sigma\_y)

}

}

persp(x\_vals, y\_vals, z\_vals, col = "lightblue", theta = 30, phi = 30, xlab = "X", ylab = "Y", zlab = "Densitate de probabilitate")

lines(c(param, param), c(-5, 5), c(0, 0), col = "red", lty = 2)

}

Sys.sleep(0.1)

}

}, interval = 0.1, movie.name = "reprezentare\_animatie.tex", ani.width = 400, ani.height = 400)

})

}

# Rulează aplicația Shiny

shinyApp(ui, server)

g) 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.

library(shiny)

# Definirea funcției de densitate de probabilitate bidimensională f(x, y)

f <- function(x, y) {

x + y

}

# UI

ui <- fluidPage(

titlePanel("Calculul Momentelor pentru Variabilele Aleatoare Bidimensionale"),

sidebarLayout(

sidebarPanel(

numericInput("xmin", "Valoare minimă pentru X:", value = 0, min = -Inf, max = Inf),

numericInput("xmax", "Valoare maximă pentru X:", value = 1, min = -Inf, max = Inf),

numericInput("ymin", "Valoare minimă pentru Y:", value = 0, min = -Inf, max = Inf),

numericInput("ymax", "Valoare maximă pentru Y:", value = 1, min = -Inf, max = Inf),

actionButton("calculeaza", "Calculează")

),

mainPanel(

verbatimTextOutput("rezultate")

)

)

)

# Server logic

server <- function(input, output) {

observeEvent(input$calculeaza, {

# Calculul mediei (momentului de ordin 1) pentru X și Y

medie\_x <- integrate(function(x) x \* integrate(function(y) f(x, y), input$ymin, input$ymax)$value, input$xmin, input$xmax)$value

medie\_y <- integrate(function(y) y \* integrate(function(x) f(x, y), input$xmin, input$xmax)$value, input$ymin, input$ymax)$value

# Calculul dispersiei (momentului de ordin 2) pentru X și Y

var\_x <- integrate(function(x) (x - medie\_x)^2 \* integrate(function(y) f(x, y), input$ymin, input$ymax)$value, input$xmin, input$xmax)$value

var\_y <- integrate(function(y) (y - medie\_y)^2 \* integrate(function(x) f(x, y), input$xmin, input$xmax)$value, input$ymin, input$ymax)$value

# Calculul momentelor initiale și centrate până la ordinul 4 (dacă există)

moment\_ord\_3\_x <- integrate(function(x) (x - medie\_x)^3 \* integrate(function(y) f(x, y), input$ymin, input$ymax)$value, input$xmin, input$xmax)$value

moment\_ord\_3\_y <- integrate(function(y) (y - medie\_y)^3 \* integrate(function(x) f(x, y), input$xmin, input$xmax)$value, input$ymin, input$ymax)$value

moment\_ord\_4\_x <- integrate(function(x) (x - medie\_x)^4 \* integrate(function(y) f(x, y), input$ymin, input$ymax)$value, input$xmin, input$xmax)$value

moment\_ord\_4\_y <- integrate(function(y) (y - medie\_y)^4 \* integrate(function(x) f(x, y), input$xmin, input$xmax)$value, input$ymin, input$ymax)$value

# Afișarea rezultatelor sau mesajelor corespunzătoare dacă momentele nu există

if (is.na(medie\_x) || is.na(medie\_y) || is.na(var\_x) || is.na(var\_y)) {

output$rezultate <- renderPrint({

"Densitatea comună nu este bine definită."

})

} else {

output$rezultate <- renderPrint({

paste(

"Media X:", medie\_x, "\n",

"Media Y:", medie\_y, "\n",

"Dispersia X:", var\_x, "\n",

"Dispersia Y:", var\_y, "\n",

"Momentul de ordin 3 pentru X:", moment\_ord\_3\_x, "\n",

"Momentul de ordin 3 pentru Y:", moment\_ord\_3\_y, "\n",

"Momentul de ordin 4 pentru X:", moment\_ord\_4\_x, "\n",

"Momentul de ordin 4 pentru Y:", moment\_ord\_4\_y, "\n"

)

})

}

})

}

# Rularea aplicației Shiny

shinyApp(ui, server)

h) 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.

library(shiny)

# UI

ui <- fluidPage(

titlePanel("Calculul Așteptării și Dispersiei pentru o Funcție Unidimensională"),

sidebarLayout(

sidebarPanel(

textInput("g\_expresie", "Introduceți funcția g(x):", "x^2"),

actionButton("calculeaza", "Calculează")

),

mainPanel(

verbatimTextOutput("rezultate")

)

)

)

# Server logic

server <- function(input, output) {

observeEvent(input$calculeaza, {

# Convertirea expresiei într-o funcție

g <- function(x) eval(parse(text = input$g\_expresie))

# Definirea densității marginale pentru X (pentru exemplu, am folosit o densitate normală standard)

f\_X <- function(x) {

dnorm(x, mean = 0, sd = 1)

}

# Funcția pentru E[g(X)]

E\_gX <- function(x) {

g(x) \* f\_X(x)

}

# Funcția pentru E[g(X)^2]

E\_gX2 <- function(x) {

g(x)^2 \* f\_X(x)

}

# Calculul E[g(X)]

integrate\_E\_gX <- integrate(E\_gX, lower = -Inf, upper = Inf)

E\_gX\_value <- integrate\_E\_gX$value

# Calculul E[g(X)^2]

integrate\_E\_gX2 <- integrate(E\_gX2, lower = -Inf, upper = Inf)

E\_gX2\_value <- integrate\_E\_gX2$value

# Calculul dispersiei

Var\_gX <- E\_gX2\_value - E\_gX\_value^2

# Afișarea rezultatelor

output$rezultate <- renderPrint({

paste(

"E[g(X)]:", E\_gX\_value,

"Var(g(X)):", Var\_gX

)

})

})

}

# Rularea aplicației Shiny

shinyApp(ui, server)

*Concluzii*

Realizarea proiectului in echipa a adus un plus de cunostinte legat de limbajul R si de vizualizarile ce pot fi implementate cu ajutorul acestuia.

Am acumulat informatii noi ce ne pot ajuta chiar si la examenul de curs,si totodata ne-am imbunatatit lucrul in echipa ,acest limbaj fiind unul nou pentru noi,pana acum.

*Va Multumim!*