Identificarea unor eventuale dificultăți în realizarea cerințelor:

La primul exercițiu am întâmpinat dificultăți la subpunctele a) și b), mai precis nu am putut decide câte celule din matrice ar trebui completate inițial pentru a putea fi ulterior calculată toată repartiția. Am ales să completez inițial aproximativ 50% din valori și să mă folosesc de 0-uri. Un alt impediment a fost faptul că am ales să continui rezolvarea exercițiilor chiar dacă primele două nu au fost corecte de la început. De aceea, spre final, după ce am rezolvat primele două subpuncte, am descoperit că modificările făcute la output-ul funcției nu se potrivesc cu datele de input pentru funcțiile de la următoarele subpuncte.

Încă nu am reușit sa raționalizez un algoritm perfect pentru completarea matricei, în principal cred că e din cauză că la inițializarea cu valori nu mi-am dat seama exact din cerința cum ar trebui să fie făcute. Am ales un 50%, dar dacă exista vreun raționament pentru reinițializare, atunci poate aș fi reușit sa aduc funcția de completare la un nivel mai satisfăcător.

Pentru cea de-a doua parte a proiectului, dificultățile au apărut la calculul integralelor și folosirea funcțiilor din pachetele noi, dar au fost rezolvate cu ajutorul resurselor. Descoperirea unor pachete noi și a implementării acestora nu a fost ușoară, însă a simplificat procesul de rezolvare al cerințelor.

Probleme care au rămas deschise în urma implementării:

Calculele nu au precizie de 100% din cauza limitărilor floating point-ului și am fost încurcat la început când lucram cu prea multe zecimale. Din această cauză valorile deviază puțin, lucru care are impact asupra rezultatului final. Câteodată am numere care apar cu notație științifică cum ar fi cu un „e” la mijloc etc., în cazuri rare.

Nu îmi pot da seama de precizia exactă a diagramelor sau a graficelor pentru toate valorile, însă sunt exacte pentru valorile mici sau de mărime medie.

Nu sunt sigur de exactitatea calculelor din partea a 2-a pentru toate valorile posibile introduse de user.

Concluzii:

În urma realizării acestui proiect m-am familiarizat cu limbajul R, învățând să scriu programe complexe cu ajutorul documentației. De asemenea, mi-am dezvoltat aptitudinile de înțelegere și interpretare a problemelor de matematică, reușind să transpun chestiuni teoretice în programe care să ajute și să simplifice mult calculul efectiv făcut de mână.

Exercițiul 1

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

$$X:\begin{pmatrix} x_1\ x_2\ \ldots x_n \\ p_1\ p_2\ \ldots p_n \end{pmatrix} \qquad Y:\begin{pmatrix} y_1\ y_2\ \ldots y_m \\ q_1\ q_2\ \ldots q_m \end{pmatrix}$$

a) 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 într-o formă în care poate fi completată ulterior.

```
frepcomgen <- function(n, m) {
```

```
  # verificam daca numarul de elemente cerute este mai mic decat limita
  if (n > 201 || m > 201) {
      stop("Prea multe elemente cerute, am hardcodat sa fie 100 limita la o
va, se poate schimba tho")
  }

  # generam probabilitati aleatorii pentru X și le normalizam pentru a avea
suma 1
  pX <- runif(n) pX <-
  round(pX / sum(pX), 6)

  # similar pentru Y pY <-
  runif(m) pY <- round(pY /
  sum(pY), 6)

  # cream dataframe-uri pentru X și Y cu valori si probabilitati
  X <- data.frame(val = sample(-100:100, n, replace = FALSE), p = pX)
  Y <- data.frame(val = sample(-100:100, m, replace = FALSE), p = pY)

  # initializam matricea de repartitie comuna cu NA not allocated gen
  P <- matrix(NA, nrow = m + 1, ncol = n + 1)
  # setam probabilitatile marginale in matrice
  P[m+1, 1:n] <- pX
  P[1:m, n+1] <- pY

  # completam aleatoriu unele valori in matricea de repartitie
  comuna for (j in 1:m) { for (i in 1:n) { if (runif(1) < 0.5) {
       P[j, i] <- round(runif(1, min = 0, max = min(P[m+1, i], P[j, n+1])),
3)
      }
      }
  }
  # setam totalul probabilitatilor la 1
  P[m+1, n+1] <- 1

  # returnam o lista cu variabilele aleatorii X, Y și matricea P
  list(X = X, Y = Y, P = P)
}
```

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

```
# functia pentru completarea matricei P cu valorile lipsa
fcomplrepcom <- function(P, X, Y) {
  m <- nrow(P) - 1
  n <- ncol(P) - 1

  # calculam valorile lipsa in matrice
  for (j in 1:m) { for (i
      in 1:n) { if
      (is.na(P[j, i])) {
      # calculam deficitul pentru fiecare coloana și rand
      col_deficit <- P[j, n+1] - sum(P[j, 1:n], na.rm = TRUE)
      row_deficit <- P[m+1, i] - sum(P[1:m, i], na.rm = TRUE)
      val_missing <- min(col_deficit, row_deficit)
      P[j, i] <- max(val_missing, 0)
      }
      }
  }

  # ajustam sumele pentru fiecare coloana si rand
  P <- adjust_sums(P, m, n)
    # redenumim coloanele pentru variabilele aleatorii       și  returnarea
rezultatelor names(X) <-
  c("x", "p") names(Y)
  <- c("y", "q")
  list(X = X, Y = Y, Pij = P)
}

# functia pentru ajustarea sumelor în matrice
adjust_sums <- function(P, m, n) {
  for (i in 1:n) { sum_col <- sum(P[1:m, i],
      na.rm = TRUE) if (sum_col != P[m+1,
      i]) {
      distribute_adjustment(P[1:m, i], P[m+1, i] - sum_col)
      }
  }
  for (j in 1:m) { sum_row <- sum(P[j, 1:n],
      na.rm = TRUE) if (sum_row != P[j,
      n+1]) {
      distribute_adjustment(P[j, 1:n], P[j, n+1] - sum_row)
      }
  }
  return(P)
}
# functia pentru distribuirea ajustarii în celulele necompletate ale
matricei
distribute_adjustment <- function(cells, adjustment) {
```

```
  na_cells <- which(is.na(cells)) if (length(na_cells) >
  0 && adjustment != 0) { adjustment_per_cell <-
  adjustment / length(na_cells) cells[na_cells] <-
  adjustment_per_cell }
}
```

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

```
frepmarginal <- function(tabel_va) {
  repCom <- tabel_va$Pij
  m <- nrow(repCom)-1 #(initializare pt cazul in care deja avem pe n+1 si
m+1 repartitiile marginale)
  n <- ncol(repCom)-1

  #initializam pentru vectorii repartitiilor marginale
  rep_margin_X <- numeric(n)
  rep_margin_Y <- numeric(m)

  # calculam repartitia marginala pentru Y
  for(i in 1:m) {
    rep_margin_Y[i] <- sum(repCom[i,1:n ]) # suma de elem de pe coloanele
corespunzatoare cele de a i-a linie
  }

  # repartitia marginala pt X
  for(j in 1:n) {
    rep_margin_X[j] <- sum(repCom[1:m, j]) # suma de pe linii pentru colana
j
  }

  list(X = rep_margin_X, Y = rep_margin_Y)
}
repartitii_marginale <- frepmarginal(tabel_completat)
# Afișarea repartițiilor marginale
print(repartitii_marginale$X)
print(repartitii_marginale$Y)
```

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

```
#calculam media unei variabile aleatoare la patrat
fMedPatrat<- function(valori, probabilitati) {
  # Combinam valorile patrate si probabilitatile intr-un cadru de date
  data_combined <- data.frame(x_patrat = valori^2, prob = probabilitati)
```

```
  # Vector pentru rezultate
  rezultat <- numeric(length(unique(valori^2)))

  # iteram valorile patrate unive
  for (i in seq_along(rezultat)) {
    valoare_patrata <- unique(data_combined$x_patrat)[i]

    # adunam probabilitatile daca se repeta vreo valoare
    rezultat[i] <- sum(data_combined$prob[data_combined$x_patrat
    ==
valoare_patrata])
  }

  # calculam suma valorilor patrate inmultite cu probabilitatile
  rezultat_final <- sum(unique(data_combined$x_patrat) * rezultat)

  return(rezultat_final)
}

fpropcov<-function(a,b,c,d,tabel_va){
  m <- nrow(tabel_va$Pij)-1
  n <- ncol(tabel_va$Pij)-1
  #pasul 1. calculam mediile celor doua variabile
  E_X<-sum(tabel_va$X$x*tabel_va$X$p)
  E_Y<-sum(tabel_va$Y$y*tabel_va$Y$q)

  #pasul 2. calculam mediile variabilelor la patrat
  E_X_patrat<-fMedPatrat(tabel_va$X$x,tabel_va$X$p)
  E_Y_patrat<-fMedPatrat(tabel_va$Y$y,tabel_va$Y$q)

  #pasul 3. calculam variantele
  Var_X<-E_X_patrat-(E_X*E_X)
  Var_Y<-E_Y_patrat-(E_Y*E_Y)

  #pasul 4. calculam media
  XY XY<-numeric(n*m)
  probXY<-numeric(n*m) for(i
  in 1:n){
    for(j in 1:m)
      XY<-c(tabel_va$X$x[i]*tabel_va$Y$y[j]) probXY<-
    c(tabel_va$Y$q[j]*tabel_va$X$p[i]) }
  covXY<-(sum(XY*probXY)-(E_X*E_Y))

  #pasul 5. Aplicam formula de la laborator
  return((a*c*Var_X)+(b*d*Var_Y)+(a*d+b*c)*covXY)
```

```
}

(fpropcov(5,9,-3,-2,tabel_completat))
```

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

```
# P(X|Y=yi) si P(Y|X=xi) daca probCond=1 conditionam pe X la Y, daca
probCond=0 conditionam pe Y la X
fPcond <- function(tabel_va, probCond, xi = -Inf, yi = -Inf) {
  repCom <-
  tabel_va$Pij m <-
  nrow(repCom) n <-
  ncol(repCom)

  if (probCond == 1 && yi %in% tabel_va$Y$y) { #verificam sa existe yi in
repartitie si cum vrem sa conditionam variabilele intre ele poz_yi <-
match(yi, tabel_va$Y$y) #aflam pozitia lui yi in repartitia comuna
rep_X_conditionat_Y <- numeric(n-1)
    for (i in 1:(n-1)) {
      rep_X_conditionat_Y[i] <- repCom[poz_yi, i] / repCom[poz_yi, n]
#pentru fiecare variabila de la linia corespunzatoare pozitiei lui yi facem
impartirea cu probabilitatea lui x de la linia yi
    }
    return(rep_X_conditionat_Y)
  } else if (probCond == 0 && xi %in% tabel_va$X$x) { #facem acelasi lucru
si pentru xi, numai ca aici pozitia coloanei ramane neschimbata si se
schimba pozitia pozitiile liniilor poz_xi <- match(xi, tabel_va$X$x)
    rep_Y_conditionat_X <- numeric(m-1)
    for (i in 1:(m-1)) {
      rep_Y_conditionat_X[i] <- repCom[i, poz_xi] / repCom[m, poz_xi]
    }
    return(rep_Y_conditionat_X)
  }
}

(X_cond_Y<-fPcond(tabel_completat,1,yi=2)) (Y_condX<-
fPcond(tabel_completat,0,xi=2))
```

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

```
# cazul in care vrem P(X=xi,Y=yi) fComun<-
function(tabel_va,xi,yi){
  repCom <- tabel_va$Pij
```

```
  # verificam existenta valorilor in reprezentarea variabilelor
  if(yi %in% tabel_va$Y$y && xi %in% tabel_va$X$x){
    return(repCom[match(yi,tabel_va$Y$y),match(xi,tabel_va$X$x)]) #daca
exista, afisam valoarea de la pozitia [yi,xi]
  }
  else{
    return("Valorile     nu     exista     in     repartitiile
variabilelor") } }

(rezultat<-fComun(tabel_completat,xi=-1,yi=2)) (rezultat<-
fComun(tabel_completat,xi=-2,yi=7))
```

h) 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

```
#tabelul contine si repartitiile marginale ale lui X si Y
fverind<-function(repCom){
  m <- nrow(repCom)
  n <- ncol(repCom)
  for(i in 1:(m-
  1)){
    for(j in 1:(n-1)){
      if(repCom[i,j]!=(repCom[i,n]*repCom[m,j]))
         {return("X si Y sunt
    dependente!")} }
  }
  return("X si Y sunt independente!")
}
(rezultat<-fverind(tabel_completat$Pij))

fvernecor<-function(tabel_va){
  #intai trebuie sa calculam coeficientul de corelatie al celor doua
variabile
  #pentru asta calculam media, varianta si covarianta a celor 2 variabile
  repCom<-tabel_va$Pij

  m <- nrow(repCom)-1
  n <- ncol(repCom)-1
  #pasul 1. calculam
  mediile celor doua
  variabile
  E_X<-sum(tabel_va$X$x*tabel_va$X$p)
  E_Y<-sum(tabel_va$Y$y*tabel_va$Y$q)
```

```
  #pasul 2. calculam mediile variabilelor la patrat
  E_X_patrat<-fMedPatrat(tabel_va$X$x,tabel_va$X$p)
  E_Y_patrat<-fMedPatrat(tabel_va$Y$y,tabel_va$Y$q)

  #pasul 3. calculam variantele
  Var_X<-E_X_patrat-(E_X*E_X)
  Var_Y<-E_Y_patrat-(E_Y*E_Y)

  #pasul 4. calculam media
  XY XY<-numeric(n*m)
  probXY<-numeric(n*m) for(i
  in 1:n){
    for(j in 1:m)
      XY<-c(tabel_va$X$x[i]*tabel_va$Y$y[j]) probXY<-
      c(tabel_va$Y$q[j]*tabel_va$X$p[i])
  }
  EXY<-sum(XY*probXY)

  #pasul 5. Calculam formula coeficientului de corelatie
  coef_cor<-(EXY-(E_X*E_Y))/sqrt(Var_X*Var_Y)
  if(coef_cor==0)
    return("Variabile aleatoare necorelate")
  else return("Variabile aleatoare corelate")
}

(fvernecor(tabel_completat))
```

Exercitiul 2

2. Folosind pachetele R shiny(https://shiny.rstudio.com/), animate(https://cran.rproject.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 în această manieră, atunci când este posibil.

b) Interpretarea geometrică a integralei duble.

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

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

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

Pentru implementarea teoremei Fubini dar si a calculului integralei duble m-am folosit de librariile shiny(interfata utilizatorului si server back-end) , ggplot2 (reprezentari grafice), reshape2 (restructurarea datelor - daca datele sunt mult prea lungi) , plotly (notiuni interactive pe ggplot 2), MASS(pentru mrvnorm)

```
library(shiny)
library(ggplot2)
library(reshape2)
library(plotly)
library(MASS)

#Pentru interfata utilizatorului am impartit in 2 sectiuni : sectiunea de
input si cea de output

  ui <- fluidPage(
    titlePanel("Integrarea Functiilor Bidimensionale folosind Teorema lui
Fubini"),
    sidebarLayout(
      sidebarPanel(
        textInput("func", "Introduceti functia f(x, y):", "x^2 + y^2"),
        numericInput("xmin", "Limita inferioara x:", -1),
        numericInput("xmax", "Limita superioara x:", 1),
        numericInput("ymin", "Limita inferioara y:", -1),
        numericInput("ymax", "Limita superioara y:", 1),
        actionButton("calc", "Calculeaza"),
        actionButton("check_density", "Verifica Densitatea de
Probabilitate"), selectInput("var_type", "Selectati tipul variabilei
        aleatoare:",
                    choices = c("Unidimensionala", "Bidimensionala")),
        textInput("density", "Introduceti densitatea de probabilitate:",
"1/(sqrt(2*pi)) * exp(-x^2/2)"), actionButton("create_var", "Creeaza
        Variabila Aleatoare"), actionButton("calc_marginal", "Calculeaza
```

```
        Densitatile Marginale"), actionButton("calc_conditional",
        "Calculeaza Densitatile
Conditionate"),

        numericInput("x_val_conditional", "Valoarea X pentru densitatea
conditionata:", 0), numericInput("y_val_conditional", "Valoarea Y
        pentru densitatea
conditionata:", 0),

        numericInput("param1", "Parametrul 1 (medie pentru
unidimensional):", 0), numericInput("param2", "Parametrul 2 (deviatie
        standard pentru
unidimensional):", 1), selectInput("graphic_type", "Alegeti tipul
        graficului:", choices = c("Densitate", "Functie de
        Repartitie")),
        actionButton("plot_graphic", "Reprezinta Graficul")
      ),
      mainPanel(
        plotlyOutput("plot"),
        plotlyOutput("plot3D"),
        textOutput("result"),
        plotOutput("graphic_plot")
      )
    )
  )

server <- function(input, output, session) {
  f <- function(x, y) {
    sapply(X = x, FUN = function(x) eval(parse(text = paste("(",
input$func, ")")), list(x = x, y = y)))
 # calculez valorile functiei introduse de user pentru setul de valori "x"
avand in vedere valorile "y" }

#punctul a si b
  observeEvent(input$calc, {
    tryCatch({ # tryCatch pentru erori
      test_val <- f(0, 0) # fac o valoare de test
      if (length(test_val) != 1) {
        stop("Functia trebuie sa returneze o singura
      valoare.") }
    }, error = function(e) {
      output$result <- renderText(paste("Eroare: Functia introdusa nu este
valida. Eroare:", e$message))
      return()
    })
```

```
# calculez integrala
    result <- tryCatch({
      integral <- integrate(Vectorize(function(x) { # integrarea functiei
in raport cu x integrate(Vectorize(function(y) f(x, y)),
        input$ymin,
input$ymax)$value # integrarea #functiei in raport cu y }),
      input$xmin, input$xmax)$value # limitele de integrare
      paste("Valoarea integralei este: ", integral)
    }, error = function(e) { paste("Eroare la calculul
      integralei: ", e$message)
    }) output$result <-

    renderText(result)

#generarea valorilor pentru x si y xvals <- seq(input$xmin,
    input$xmax, length.out = 30) yvals <- seq(input$ymin,
    input$ymax, length.out = 30) grid <- expand.grid(x =
    xvals, y = yvals) #si creez grila grid$z <- mapply(f,
    grid$x, grid$y)

#afisarea grafului interactiv
    output$plot <- renderPlotly({
      p <- ggplot(grid, aes(x = x, y = y, z = z)) +
        geom_tile(aes(fill = z)) +
        stat_contour() +
        theme_minimal()

      ggplotly(p)
    })

    # Grafic 3D
    output$plot3D <- renderPlotly({
#fac o matrice z cu rezultatele functiei (x,y) pentru fiecare punct
  zvals <- outer(xvals, yvals, Vectorize(f)) plot_ly(x = ~xvals, y
  = ~yvals, z = ~zvals, type = "surface") }) })

#punctul c
  observeEvent(input$check_density, {
#verific daca valorile functiei sunt pozitive pe grid
    is_positive <- tryCatch({ test_grid <- expand.grid(x = seq(-10,
      10, length.out = 100), y =
seq(-10, 10, length.out = 100)) all(mapply(f,
      test_grid$x, test_grid$y) >= 0)
    }, error = function(e)
      { FALSE
    })
```

```
#calculez integrala totala a functiei
    integral_total <- tryCatch({
      integrate(Vectorize(function(x) {
        integrate(Vectorize(function(y) f(x, y)), -Inf, Inf)$value
      }), -Inf, Inf)$value
    }, error = function(e)
      { NA
    })

#verific daca integrala totala este aproape de 1 si valorile sunt pozitive
    is_density <- is_positive && !is.na(integral_total) &&
abs(integral_total - 1) < 0.01

#afisez rezultatul density_result <- if
    (is_density) {
      "Functia introdusa este o densitate de probabilitate."
    } else {
      "Functia introdusa NU este o densitate de probabilitate."
    }

    output$result <-
  renderText(density_result) })

#punctul d

  observeEvent(input$create_var, {
    if (input$var_type == "Unidimensionala")
      { tryCatch({
#evalueaz functia de densitate de probabilitate dens_func <-
        eval(parse(text = paste("function(x) {",
input$density, "}")))

 # verific daca integrala densitatii de probabilitate este 1

        if (abs(integrate(dens_func, -100, 100)$value - 1) > 0.01) {
          stop("Densitatea de probabilitate nu este
        valida.") }
#creez o variabila aleatoare unidimensionala
        var_aleatoare <- stats::rnorm(1000)
#mesajul de confirmare output$result <- renderText("Variabila aleatoare
        unidimensionala
creata.")
      }, error = function(e) { output$result <-
      renderText(paste("Eroare:", e$message)) })
```

```
  } else if (input$var_type == "Bidimensionala")
      { tryCatch({
#creez o functie de densitate de probabilitate bidimensionala
        dens_func_bi <- eval(parse(text = paste("function(x, y) {",
input$density, "}")))
#verific daca integrala totala a densitatii de probabilitate este aprox 1
        integral_valid <- abs(integrate(function(x) {
          integrate(function(y) dens_func_bi(x, y), -100, 100)$value
        }, -100, 100)$value - 1) < 0.01
        if (!integral_valid) {
          stop("Densitatea de probabilitate bidimensionala nu este
valida.")
        }
#presupun o distributie normala bidimensionala
        mu <- c(input$param1, input$param1)
        sigma <- matrix(c(input$param2^2, 0, 0, input$param2^2), nrow = 2)
#generez o variabila aleatoare bidimensionala var_aleatoare_bi <-
        mvrnorm(1000, mu = mu, Sigma = sigma) output$result <-
        renderText("Variabila aleatoare bidimensională
creata.")
      }, error = function(e) { output$result <-
      renderText(paste("Eroare:", e$message)) })
    }
  })

#punctul e

  observeEvent(input$calc_marginal,
    { tryCatch({
#limita pentru intervalul de integrare
      interval_limit <- 10

#calculez densitatea marginala X prin integrarea functiei f(0,y) pe
intervalul [-10,10] marginal_X <- integrate(function(y) f(0, y), -
interval_limit,
interval_limit)$value
#calculez densitatea marginala Y prin integrarea functiei f(x,0) pe
intervalul [-10,10] marginal_Y <- integrate(function(x) f(x, 0), -
interval_limit,
interval_limit)$value
#rezultatul output$result <- renderText(paste("Densitatea marginala
X: ", marginal_X,
                                          "\nDensitatea marginala Y: ",
marginal_Y))
#daca apar erori
```

```
    }, error = function(e) { output$result <- renderText(paste("Eroare
la calculul densitatilor marginale: ", e$message,
                                          "\nVerificati functia si intervalul
de integrare."))
    })
  })

  observeEvent(input$calc_conditional, {
    tryCatch({
#intervalul limita pentru integrare
      interval_limit <- 10

#calculul densitatii marginale X
      marginal_X_func <- function(x) {
        integrate(function(y) f(x, y), -interval_limit,
interval_limit)$value
      }
#calculul densitatii marginale Y
      marginal_Y_func <- function(y) {
        integrate(function(x) f(x, y), -interval_limit,
interval_limit)$value
      }

#calculul densitatii conditionate X de Y si invers
      conditional_X_given_Y <- function(x, y) {
        f(x, y) / marginal_Y_func(y)
      }
      conditional_Y_given_X <- function(x, y) {
        f(x, y) / marginal_X_func(x)
      }

#preiau valorile introduse de utilizator
      x_val <- input$x_val_conditional
      y_val <- input$y_val_conditional
      #densitatile conditionate
      conditional_X_at_given_Y <-
      conditional_X_given_Y(x_val, y_val)
      conditional_Y_at_given_X <-
      conditional_Y_given_X(x_val, y_val)

      output$result <- renderText(paste("f(X|Y=", y_val, ") la X=", x_val,
": ", conditional_X_at_given_Y,
                                          "\nf(Y|X=", x_val, ") la Y=",
y_val, ": ", conditional_Y_at_given_X))
    }, error = function(e) { output$result <- renderText(paste("Eroare
      la calculul densităţilor
```

```
condiționate: ", e$message))
    })
  })

# punctul f
  observeEvent(input$plot_graphic, {
#pentru graficul densitate si variabila unidimensionala
    if (input$graphic_type == "Densitate") {
      if (input$var_type == "Unidimensionala") {

#fac un set de date pentru a reprezenta grafic densitatea unidimensionala
        data <- data.frame(x = seq(input$xmin, input$xmax, length.out =
100))

#calculez densitatea pentru fiecare punct x ( folosesc ditributia normala )
        data$y <- dnorm(data$x, mean = input$param1, sd = input$param2)
#creez si afisez graficul cu pplot p <- ggplot(data, aes(x, y)) +
        geom_line() + ggtitle("Densitate
Unidimensionala") output$graphic_plot <-
        renderPlot(p)
      } else { if (input$var_type == "Bidimensionala" &&
        input$graphic_type ==
"Densitate") {

#fac un grid pentru a reprezenta grafic densitatea bidimensionala
          xvals <- seq(input$xmin, input$xmax, length.out = 30)
          yvals <- seq(input$ymin, input$ymax, length.out = 30)
          grid <- expand.grid(x = xvals, y = yvals)

#setez parametrii pentru ditributia normala bidimensionala
          mu <- c(input$param1, input$param1)
          sigma <- matrix(c(input$param2^2, 0, 0, input$param2^2), nrow =
2)
#calculez densitatea pentru fiecare pereche (x,y) din grid grid$z <-
apply(grid, 1, function(v) dmvnorm(v[1:2], mean = mu, sigma = sigma))
#creez si afisez graficul
          output$graphic_plot <- renderPlot({
            ggplot(grid, aes(x, y, z = z)) +
              geom_tile(aes(fill = z)) +
              stat_contour() +
              ggtitle("Densitate Bidimensionala")
          })
        }
      }
```

```
    } else { if (input$var_type ==
      "Unidimensionala") {

#fac un set de date pentru reprezentarea grafica a functiei de repartitie
unidimensionala data <- data.frame(x = seq(input$xmin, input$xmax,
length.out =
100))

#calculez functia de repartitie pentru fiecare punct x folosind distributia
normala data$y <- pnorm(data$x, mean = input$param1, sd = input$param2)
#creez si afisez graficul p <- ggplot(data, aes(x, y)) + geom_line() +
        ggtitle("Functie de
Repartitie Unidimensionala")
        output$graphic_plot <- renderPlot(p)
      } else {

        if (input$var_type == "Bidimensionala" && input$graphic_type ==
"Functie de Repartitie") {

#fac un grid pentru reprezentarea grafica a functiei de repartitie
bidimensionala xvals <- seq(input$xmin, input$xmax, length.out =
30) yvals <- seq(input$ymin, input$ymax, length.out = 30) grid <-
expand.grid(x = xvals, y = yvals)
#parametrii pentru distributia normala bidimensionala
          mu <- c(input$param1, input$param1)
          sigma <- matrix(c(input$param2^2, 0, 0, input$param2^2), nrow =
2)
#calculez probabilitatea cumulativa pentru fiecare pereche (x,y) din grid
          grid$z <- apply(grid, 1, function(v) pmvnorm(lower = -Inf, upper
= v[1:2], mean = mu, sigma = sigma))
#fac graficul si il afisez
          output$graphic_plot <- renderPlot({
            ggplot(grid, aes(x, y, z = z)) +
              geom_tile(aes(fill = z)) +
              stat_contour() +
              ggtitle("Functie de Repartitie
          Bidimensionala") })
        }
      }
    }
  })


} shinyApp(ui = ui, server =

server)
```

~ - Shiny

http://127.0.0.1:6333 | Open in Browser | Publish

Integrarea Functiilor Bidimensionale folosind Teorema lui Fubini

Introduceti functia f(x, y):

x^2 + y^2

Limita inferioara x:

-1

Limita superioara x:

1

Limita inferioara y:

-1

Limita superioara y:

1

Calculeaza

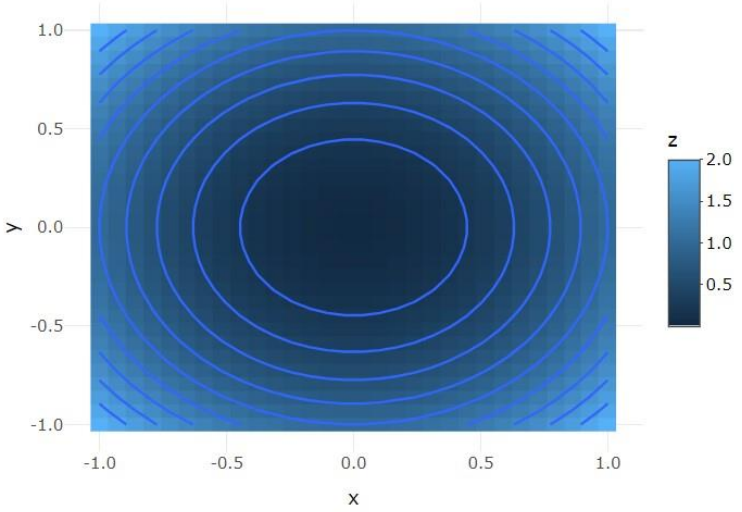
Verifica Densitatea de Probabilitate

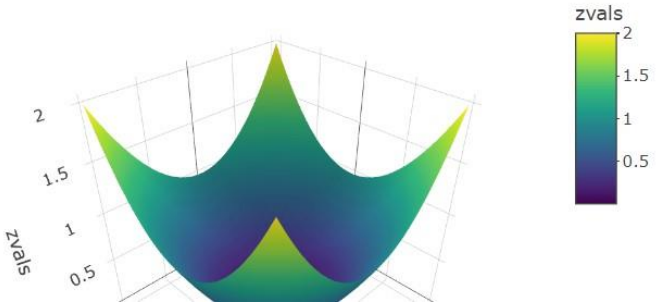
Selectati tipul variabilei aleatoare:

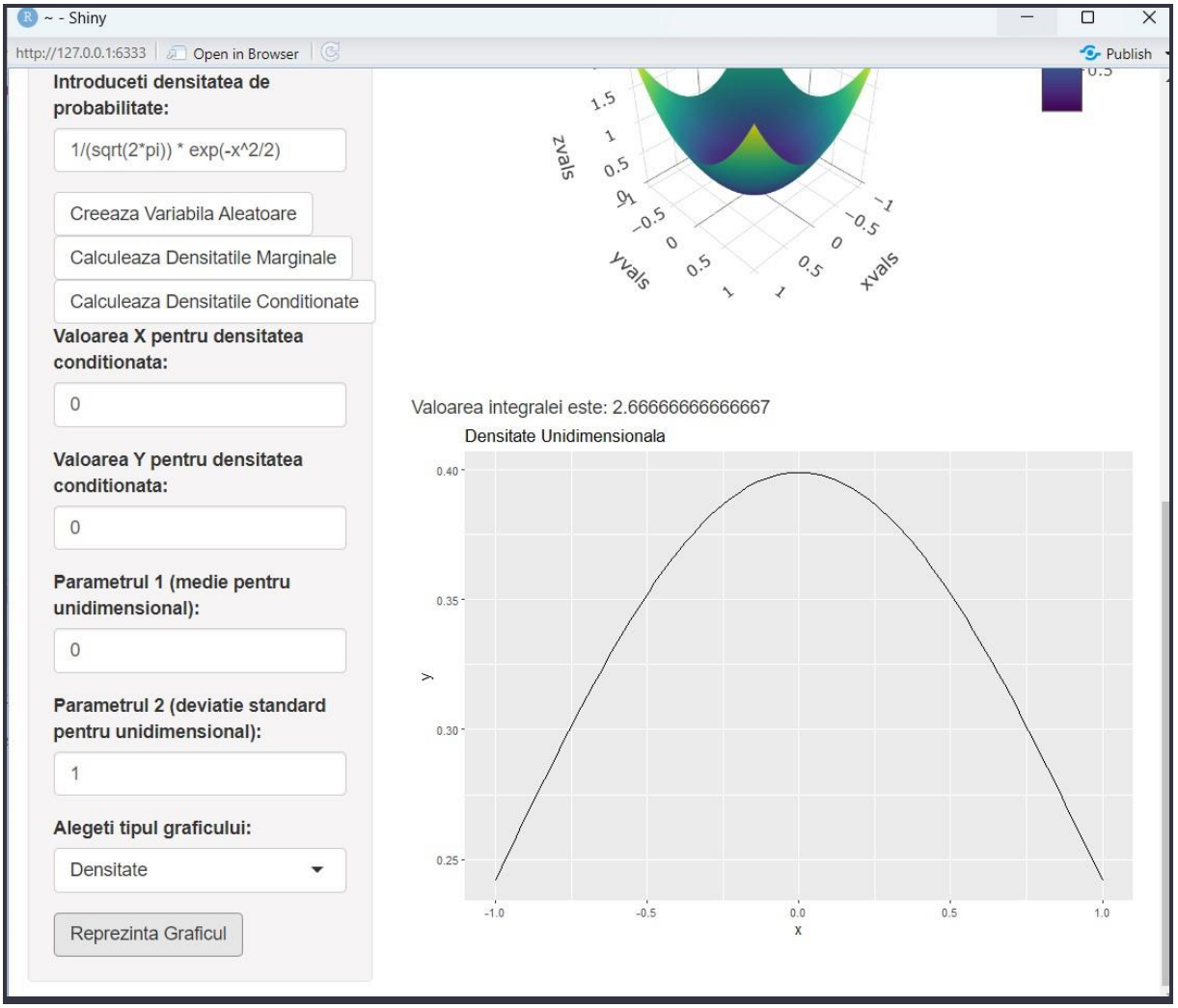
Unidimensionala

Introduceti densitatea de probabilitate:

1/(sqrt(2*pi)) * exp(-x^2/2)







Resurse suplimentare:

https://ggplot2.tidyverse.org
https://www.rdocumentation.org/packages/reshape2/versions/1.4.4
https://plotly.com/r/ https://r-graph-gallery.com/interactive-charts.html
https://www.r-bloggers.com/2019/07/integration-in-r/
https://www.geeksforgeeks.org/a-guide-to-dnorm-pnorm-rnorm-and-qnorm-in-r/
https://www.rdocumentation.org/packages/MASS/versions/7.3-60.0.1
https://stackoverflow.com/questions/tagged/r