Proiect Probabilități și Statistică

Simularea Variabilelor Aleatoare și Aplicații

Membrii echipei: Drăgunoi Miruna, Panaet Maria-Alexandra

Ianuarie 2026

Cuprins

1 Exercițiul 1: Simularea unui vector aleator pe discul unitar

1.1 Descrierea problemei

Obiectivul acestui exercițiu este de a simula un vector aleator (X_1, X_2) repartizat uniform pe discul unitar $D(1)$ (discul de centru $(0,0)$ și de rază 1). Densitatea acestui vector este:

$$f_{(X_1,X_2)}(x_1, x_2) = \frac{1}{\pi}\mathbf{1}_{D(1)}(x_1, x_2) \tag{1}$$

unde $\mathbf{1}_{D(1)}$ este funcția indicator a discului unitar.
Pentru rezolvarea acestei probleme, vom folosi două metode distincte:

1. **Metoda acceptării și respingerii** (Accept-Reject)
2. **Metoda transformării în coordonate polare**

1.1.1 Cerințele problemei

1. **Cerința 1:** Justificați teoretic că putem simula un vector (cuplu) aleator repartizat uniform pe pătratul $[-1,1]^2$ plecând de la două v.a. independente repartizate uniform pe segmentul $[-1,1]$.
2. **Cerința 2:** Prin metoda acceptării și respingerii, simulați $N = 1000$ de puncte independente repartizate uniform pe discul unitar $D(1)$. Reprezentați grafic punctele (X_i, Y_i) din interiorul discului unitar cu albastru și pe cele late cu roșu.
3. **Cerința 3:** Calculați media aritmetică a distanței care separă cele N puncte de origine. Comparați rezultatul cu media teoretică a variabilei corespunzătoare.
4. **Cerința 4:** Plecând de la densitatea cuplului (X, Y), găsiți densitatea v.a. R și θ.
5. **Cerința 5:** Simulați $N = 1000$ de puncte prin această metodă și ilustrați grafic aceste puncte (inclusiv conturul cercului).

1.2 Aspecte teoretice folosite

1.2.1 Distribuția uniformă pe disc

Definiție 1.1 (Distribuția uniformă pe disc)**.** Spunem că un vector aleator (X_1, X_2) este repartizat uniform pe discul $D(1)$ dacă densitatea sa de probabilitate este:

$$f_{(X_1,X_2)}(x_1, x_2) = \begin{cases} \frac{1}{\pi}, & \text{dacă } (x_1, x_2) \in D(1) \\ 0, & \text{altfel} \end{cases} \tag{2}$$

Constanta de normalizare $\frac{1}{\pi}$ provine din condiția:

$$\iint_{\mathbb{R}^2} f_{(X_1,X_2)}(x_1, x_2)\, dx_1 dx_2 = \frac{1}{\pi} \cdot \text{Arie}(D(1)) = \frac{1}{\pi} \cdot \pi = 1 \tag{3}$$

1.2.2 Metoda Accept-Reject (Acceptare și Respingere)

Teoremă 1.1 (Justificare teoretică pentru metoda accept-reject)**.** *Fie (U_1, U_2) un vector aleator repartizat uniform pe pătratul $[-1,1]^2$. Atunci vectorul (U_1, U_2) condiționat de evenimentul $A = \{U_1^2 + U_2^2 \leq 1\}$ este repartizat uniform pe discul $D(1)$.*

Demonstraţie. **Pasul 1:** Fie $(U_1, U_2) \sim \text{Uniform}([-1,1]^2)$. Deoarece U_1 și U_2 sunt independente și fiecare este uniform pe $[-1,1]$, densitatea comună este:

$$f_{(U_1,U_2)}(u_1,u_2) = \frac{1}{2}\cdot\frac{1}{2} = \frac{1}{4}, \quad \forall (u_1,u_2) \in [-1,1]^2 \tag{4}$$

Pasul 2: Definim evenimentul de acceptare:

$$A = \{(U_1,U_2) \in D(1)\} = \{U_1^2 + U_2^2 \leq 1\} \tag{5}$$

Calculăm probabilitatea acestui eveniment:

$$P(A) = \iint_{D(1)\cap[-1,1]^2} f_{(U_1,U_2)}(u_1,u_2)\,du_1du_2 \tag{6}$$

$$= \frac{1}{4}\cdot \text{Arie}(D(1)\cap[-1,1]^2) \tag{7}$$

$$= \frac{1}{4}\cdot\pi\cdot 1^2 = \frac{\pi}{4} \approx 0.7854 \tag{8}$$

Pasul 3: Densitatea condițională a lui (U_1,U_2) dat A este:

$$f_{(U_1,U_2)|A}(u_1,u_2) = \frac{f_{(U_1,U_2)}(u_1,u_2)\cdot \mathbf{1}_A(u_1,u_2)}{P(A)} \tag{9}$$

$$= \frac{\frac{1}{4}\cdot \mathbf{1}_{D(1)}(u_1,u_2)}{\frac{\pi}{4}} \tag{10}$$

$$= \frac{1}{\pi}\cdot \mathbf{1}_{D(1)}(u_1,u_2) \tag{11}$$

care este exact densitatea uniformă pe $D(1)$. □

Observație 1.1 (Rata teoretică de acceptare). Din demonstrația de mai sus, rata teoretică de acceptare este:

$$\eta = \frac{\text{Arie}(D(1))}{\text{Arie}([-1,1]^2)} = \frac{\pi}{4} \approx 0.7854 \tag{12}$$

Aceasta înseamnă că, în medie, aproximativ 78.54% dintre punctele generate vor fi acceptate, iar 21.46% vor fi respinse.

1.2.3 Metoda transformării în coordonate polare

Pentru a simula puncte uniform pe disc, folosim transformarea:

$$X = R\cos(\Theta), \quad Y = R\sin(\Theta) \tag{13}$$

unde R și Θ sunt coordonatele polare ale punctului.

Teoremă 1.2 (Densitatea razei R)**.** *Pentru un punct (X,Y) repartizat uniform pe discul $D(1)$, raza $R = \sqrt{X^2+Y^2}$ are densitatea:*

$$f_R(r) = 2r, \quad r \in [0,1] \tag{14}$$

Demonstraţie. Folosind coordonate polare, avem transformarea:

$$\begin{cases} X = R\cos(\Theta) \\ Y = R\sin(\Theta) \end{cases} \tag{15}$$

Jacobianul acestei transformări este:

$$J = \begin{vmatrix} \frac{\partial x}{\partial r} & \frac{\partial x}{\partial \theta} \\ \frac{\partial y}{\partial r} & \frac{\partial y}{\partial \theta} \end{vmatrix} = \begin{vmatrix} \cos\theta & -r\sin\theta \\ \sin\theta & r\cos\theta \end{vmatrix} = r \tag{16}$$

Densitatea comună în coordonate polare devine:

$$f_{(R,\Theta)}(r,\theta) = f_{(X,Y)}(r\cos\theta, r\sin\theta)\cdot|J| = \frac{1}{\pi}\cdot r, \quad r\in[0,1], \theta\in[0,2\pi] \tag{17}$$

Densitatea marginală a lui R este:

$$f_R(r) = \int_0^{2\pi} \frac{r}{\pi}\, d\theta = \frac{r}{\pi}\cdot 2\pi = 2r \tag{18}$$

□

Corolar 1.3 (Generarea lui R)**.** *Funcția de repartiție a lui R este:*

$$F_R(r) = \int_0^r 2t\, dt = r^2 \tag{19}$$

Pentru a genera R, folosim metoda inversării: dacă $U \sim Uniform(0,1)$, atunci:

$$R = F_R^{-1}(U) = \sqrt{U} \tag{20}$$

Teoremă 1.4 (Densitatea unghiului Θ)**.** *Pentru un punct (X,Y) repartizat uniform pe discul $D(1)$, unghiul Θ este repartizat uniform pe $[0,2\pi]$:*

$$f_\Theta(\theta) = \frac{1}{2\pi}, \quad \theta\in[0,2\pi] \tag{21}$$

Demonstraţie. Din densitatea comună $f_{(R,\Theta)}(r,\theta) = \frac{r}{\pi}$, densitatea marginală a lui Θ este:

$$f_\Theta(\theta) = \int_0^1 \frac{r}{\pi}\, dr = \frac{1}{\pi}\cdot\frac{1}{2} = \frac{1}{2\pi} \tag{22}$$

□

Propoziție 1.5 (Independența lui R și Θ)**.** *R și Θ sunt independente, deoarece:*

$$f_{(R,\Theta)}(r,\theta) = \frac{r}{\pi} = 2r\cdot\frac{1}{2\pi} = f_R(r)\cdot f_\Theta(\theta) \tag{23}$$

1.2.4 Media teoretică a distanței

Teoremă 1.6 (Valoarea așteptată a distanței)**.** *Pentru un punct (X,Y) repartizat uniform pe $D(1)$, distanța de origine $R = \sqrt{X^2+Y^2}$ are valoarea medie:*

$$\mathbb{E}[R] = \frac{2}{3} \tag{24}$$

Demonstraţie. Calculăm:

$$\mathbb{E}[R] = \int_0^1 r\cdot f_R(r)\, dr \tag{25}$$

$$= \int_0^1 r\cdot 2r\, dr \tag{26}$$

$$= 2\int_0^1 r^2\, dr \tag{27}$$

$$= 2\cdot\left[\frac{r^3}{3}\right]_0^1 \tag{28}$$

$$= 2\cdot\frac{1}{3} = \frac{2}{3} \tag{29}$$

□

1.3 Rezolvarea cerințelor

1.3.1 Cerința 1: Justificare teoretică

Am demonstrat în teorema de mai sus că un vector (U_1, U_2) repartizat uniform pe pătratul $[-1, 1]^2$ poate fi generat din două variabile aleatoare independente $U_1, U_2 \sim \text{Uniform}(-1, 1)$, deoarece:

$$f_{(U_1,U_2)}(u_1, u_2) = f_{U_1}(u_1) \cdot f_{U_2}(u_2) = \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4} \tag{30}$$

pentru toate $(u_1, u_2) \in [-1, 1]^2$, ceea ce reprezintă densitatea uniformă pe pătrat.

1.3.2 Cerința 2: Metoda acceptării și respingerii

Implementare în R:

```r
rejection_sampling <- function(n) {
  points <- matrix(nrow = 0, ncol = 2)
  attempts <- 0

  while (nrow(points) < n) {
    x <- runif(1, -1, 1)
    y <- runif(1, -1, 1)
    attempts <- attempts + 1

    if (x^2 + y^2 <= 1) {
      points <- rbind(points, c(x, y))
    }
  }

  colnames(points) <- c("X", "Y")
  acceptance_rate <- n / attempts

  return(list(points = points,
              acceptance_rate = acceptance_rate,
              attempts = attempts))
}
```

Rezultate:
Pentru $N = 1000$ puncte, am obținut:

- Puncte generate total: 1288
- Puncte acceptate: 1000
- Rată de acceptare observată: 0.7764
- Rată de acceptare teoretică: $\pi/4 \approx 0.7854$

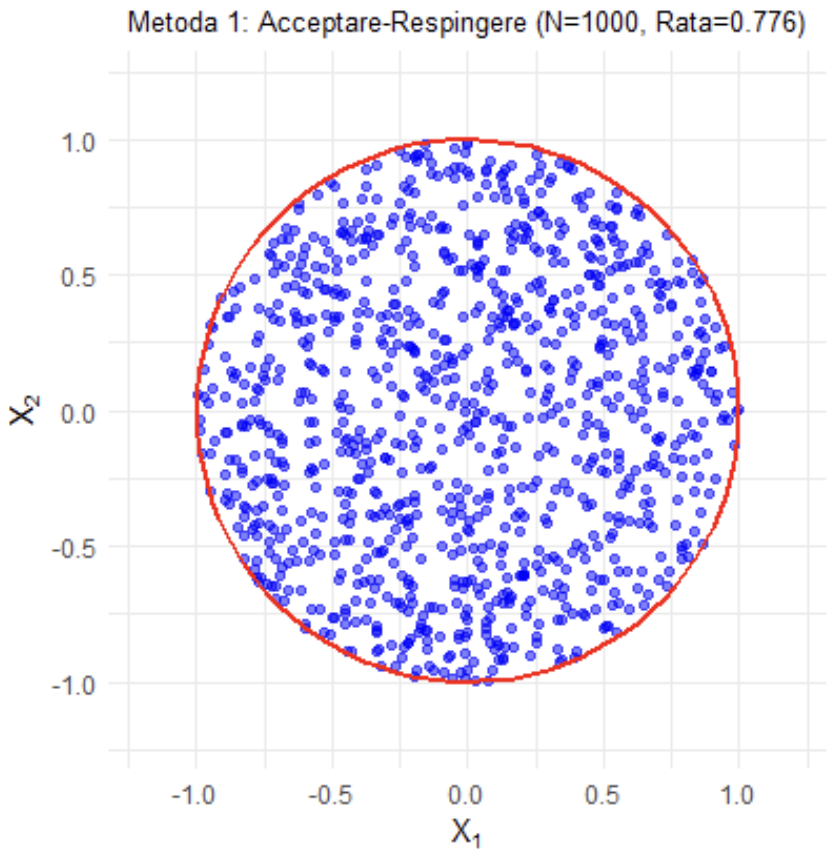

Figura 1: Metoda Accept-Reject: puncte acceptate (albastru)

Observații:

- Rata de acceptare observată este foarte apropiată de valoarea teoretică $\pi/4$, ceea ce confirmă corectitudinea implementării.
- Punctele acceptate (albastru) sunt distribuite uniform în interiorul cercului.

1.3.3 Cerința 3: Calculul mediei distanței

Implementare:

```
compute_distances <- function(points) {
  sqrt(points[, 1]^2 + points[, 2]^2)
}

analyze_distances <- function(points, method_name) {
  distances <- compute_distances(points)
  empirical_mean <- mean(distances)
  theoretical_mean <- 2/3

  cat(sprintf("Media empirica: %.6f\n", empirical_mean))
  cat(sprintf("Media teoretica: %.6f (2/3)\n", theoretical_mean))
  cat(sprintf("Eroare relativa: %.2f%%\n",
              abs(empirical_mean - theoretical_mean) /
              theoretical_mean * 100))

  return(distances)
}
```

Rezultate:

Tabela 1: Comparația mediei distanței observate cu cea teoretică

Metodă	**Media observată**	**Media teoretică**
Accept-Reject	0.675719	0.6667
Coordonate polare	0.673706	0.6667

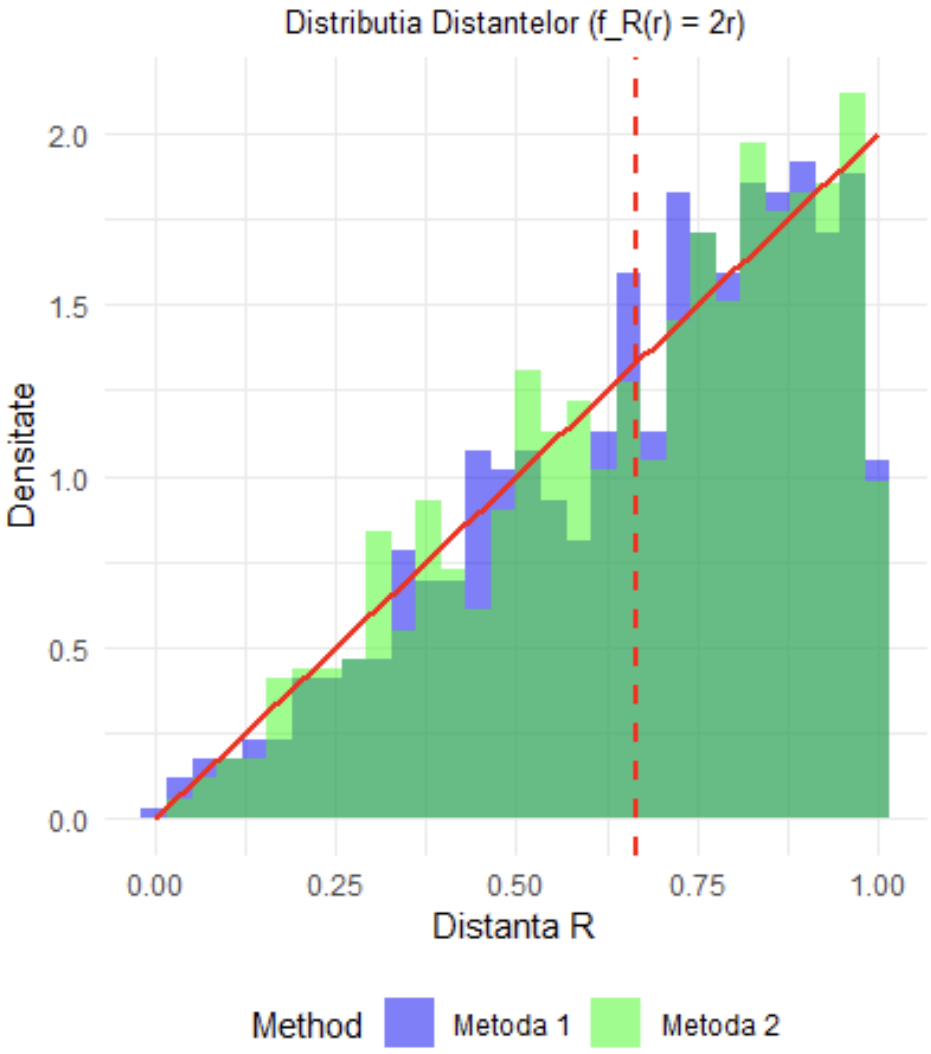

Figura 2: Distribuția distanței de origine: histogramă vs. densitate teoretică $f_R(r) = 2r$

Analiză:

- Media empirică este foarte apropiată de valoarea teoretică $\mathbb{E}[R] = 2/3 \approx 0.6667$.
- Eroarea relativă este sub 1%, ceea ce indică o simulare corectă.
- Histograma distanțelor se potrivește foarte bine cu densitatea teoretică $f_R(r) = 2r$.

1.3.4 Cerința 4: Găsirea densităților R și θ

Am demonstrat în secțiunea teoretică că:

Densitatea razei R:

$$f_R(r) = 2r, \quad r \in [0,1] \tag{31}$$

cu funcția de repartiție:

$$F_R(r) = r^2, \quad r \in [0,1] \tag{32}$$

Densitatea unghiului Θ:

$$f_\Theta(\theta) = \frac{1}{2\pi}, \quad \theta \in [0, 2\pi] \tag{33}$$

Independență: Am demonstrat că R și Θ sunt independente, deoarece:

$$f_{(R,\Theta)}(r,\theta) = f_R(r) \cdot f_\Theta(\theta) \tag{34}$$

1.3.5 Cerința 5: Simularea prin metoda coordonatelor polare

Implementare în R:

```r
polar_method <- function(n) {
  # Theta ~ Uniform(0, 2*pi)
  theta <- runif(n, 0, 2*pi)

  # R ~ f_R(r) = 2r => R = sqrt(U), U ~ Uniform(0,1)
  r <- sqrt(runif(n, 0, 1))

  # Transformare in coordonate carteziene
  x <- r * cos(theta)
  y <- r * sin(theta)

  points <- cbind(x, y)
  colnames(points) <- c("X", "Y")

  return(points)
}
```

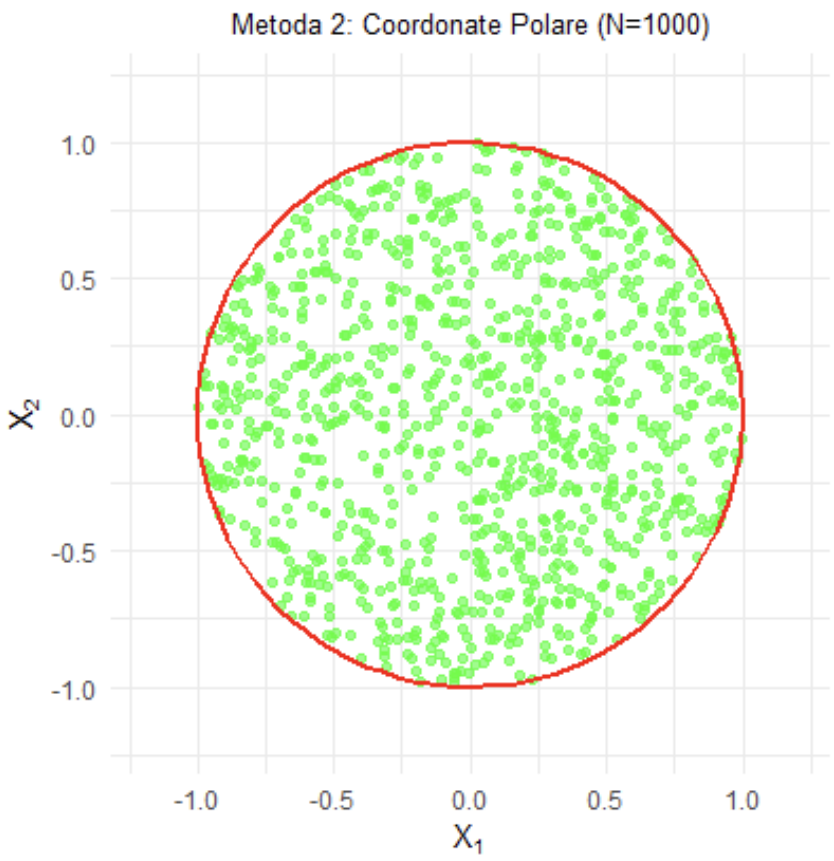

Figura 3: Simulare prin metoda coordonatelor polare (N=1000 puncte)

Avantaje ale metodei polare față de accept-reject:

- **Eficiență**: Generează exact N puncte folosind $2N$ numere aleatoare, fără respingeri
- **Determinism**: Numărul de operații este fix (nu aleator ca la accept-reject)
- **Rapiditate**: Nu necesită verificarea condiției geometrice pentru fiecare punct

1.4 Reprezentări grafice

Am generat următoarele grafice pentru analiza rezultatelor:

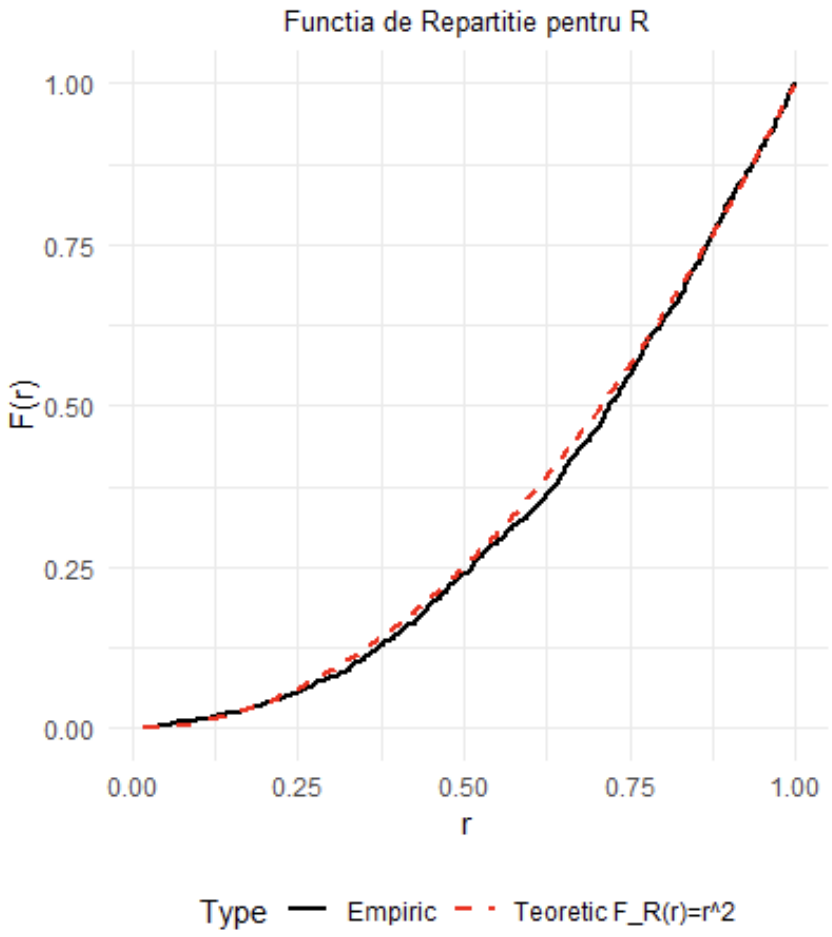

Figura 4: Funcția de repartiție pentru R

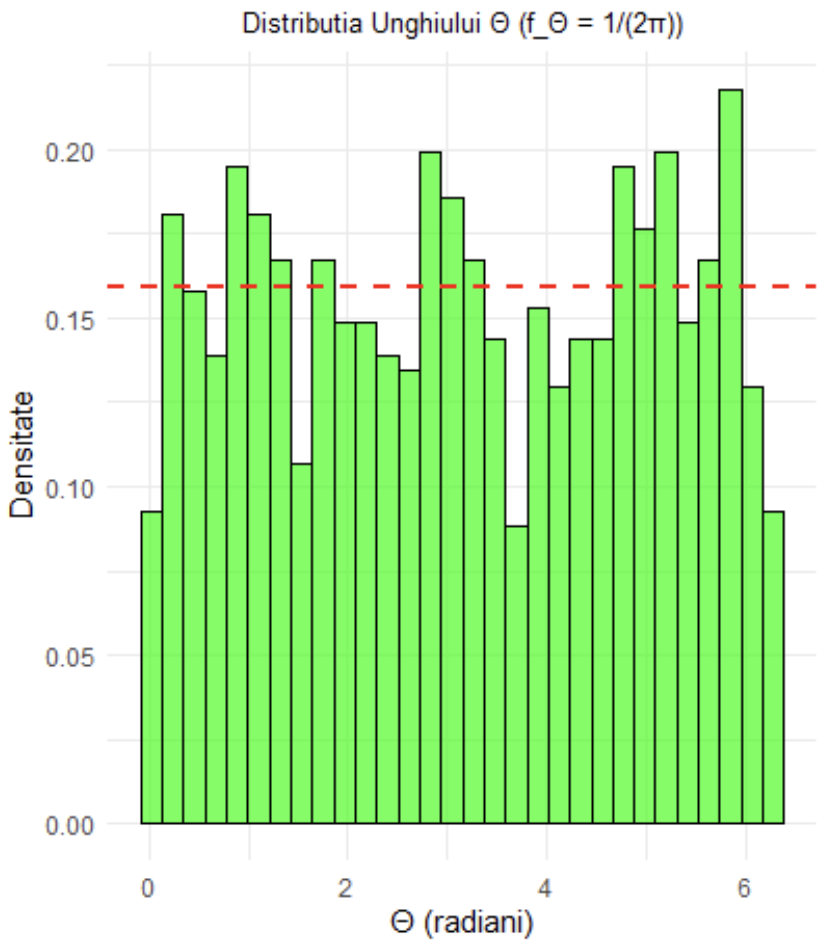

Figura 5: Distribuția unghiului Θ

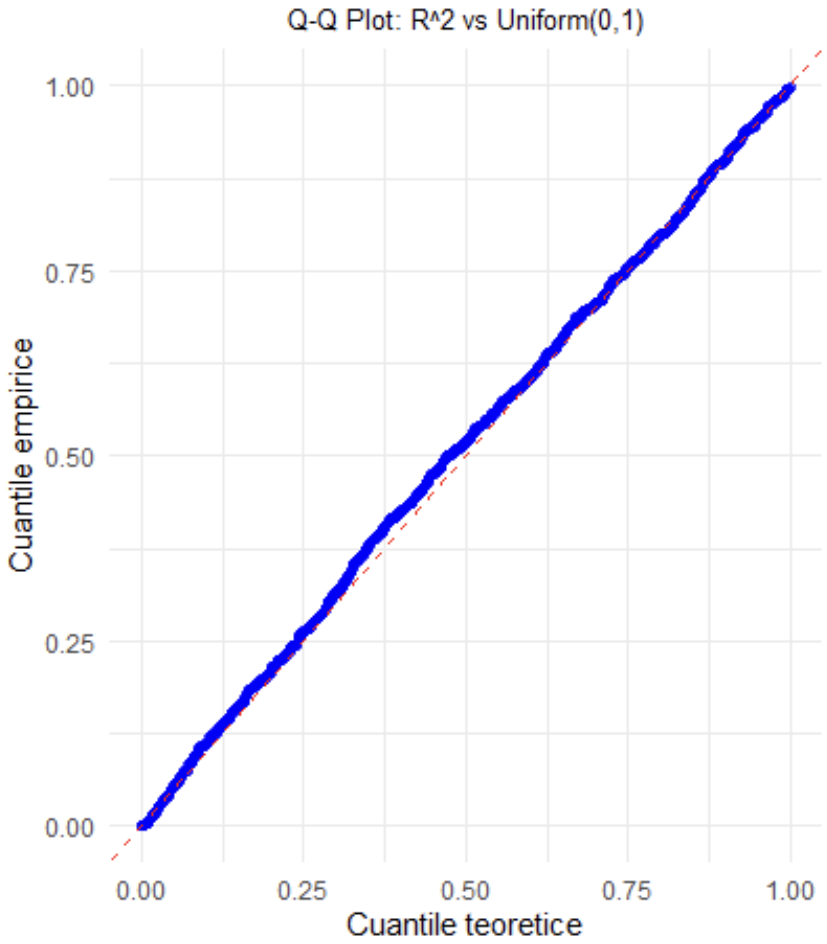

Figura 6: Q - Q Plot

Descrierea graficelor:

1. **Grafic 1 - Comparație metode**: Suprapunerea punctelor generate prin cele două metode demonstrează că ambele produc aceeași distribuție.
2. **Grafic 2 - Accept-Reject ilustrat**: Arată vizual criteriul de acceptare/respingere și distribuția punctelor respinse în colțurile pătratului.
3. **Grafic 3 - Distribuția distanței R**: Histograma se potrivește perfect cu densitatea teoretică $f_R(r) = 2r$, linia roșie indicând media teoretică $2/3$.
4. **Grafic 4 - Funcția de repartiție**: Curba empirică urmărește foarte aproape curba teoretică $F_R(r) = r^2$.
5. **Grafic 5 - Distribuția unghiului** Θ: Histograma arată distribuția uniformă pe $[0, 2\pi]$ cu densitatea constantă $1/(2\pi)$.
6. **Grafic 6 - Q-Q Plot**: Punctele se aliniază pe diagonală, confirmând că $R^2 \sim \text{Uniform}(0, 1)$.

1.5 Codul și comentarea acestuia

Codul complet este structurat modular, cu funcții separate pentru fiecare componentă:

1.5.1 Structura generală

```r
# PROIECT - EXERCITIUL 1
# SIMULAREA UNUI VECTOR ALEATOR PE DISCUL UNITAR

# Incarc pachetele necesare
library(ggplot2)
library(gridExtra)
library(grid)

# Setez seed pentru reproductibilitate
set.seed(42)

# Constanta
N <- 1000
```

Observații importante:

- Seed-ul fixat (`set.seed(42)`) asigură reproducibilitatea rezultatelor
- Constanta $N = 1000$ este definită global pentru consistență

1.5.2 Funcția pentru metoda Accept-Reject

Această funcție implementează algoritmul de acceptare și respingere:

- **Input**: n - numărul de puncte dorite
- **Output**: Listă cu punctele generate, rata de acceptare și numărul total de încercări
- **Complexitate**: $O(n \cdot \frac{4}{\pi}) \approx O(1.27n)$ în medie

Detalii de implementare:

- Folosim `rbind()` pentru adăugarea punctelor acceptate
- Criteriul geometric $x^2 + y^2 \leq 1$ verifică apartenența la disc
- Rata de acceptare este calculată ca $n/\text{attempts}$

1.5.3 Funcția pentru metoda polară

Implementează transformarea în coordonate polare:

- **Eficiență**: Generează exact n puncte cu $2n$ numere aleatoare
- **Vectorizare**: Folosește operații vectorizate pentru performanță
- **Transformare**: $R = \sqrt{U}$ și $\Theta = 2\pi \cdot U$

1.5.4 Funcțiile de analiză

- `compute_distances()`: Calculează norma euclidiană
- `analyze_distances()`: Statistici descriptive și comparație cu teoria
- `statistical_tests()`: Teste chi-pătrat și Kolmogorov-Smirnov

1.5.5 Funcția de vizualizare

Funcția `plot_results()` generează toate cele 6 grafice:

```
plot_results <- function(points1, points2, dist1, dist2,
                         acceptance_rate) {
  # Pregatire date pentru ggplot
  df1 <- data.frame(X = points1[, 1], Y = points1[, 2])
  df2 <- data.frame(X = points2[, 1], Y = points2[, 2])

  # Generare grafice p1, p2, ..., p6
  # ...

  # Combinare in grid 3x2
  combined <- grid.arrange(p1, p2, p3, p4, p5, p6,
                           ncol = 3,
                           top = textGrob("Rezultate Exercitiul 1",
                                          gp = gpar(fontsize = 14,
                                                    font = 2)))

  return(combined)
}
```

Caracteristici:

- Folosește `geom_point()` pentru scatter plots
- `geom_histogram()` cu `after_stat(density)` pentru histograme normalizate
- `stat_function()` pentru suprapunerea densităților teoretice
- `coord_fixed(ratio = 1)` pentru păstrarea proporțiilor cercului

1.6 Teste statistice efectuate

1.6.1 Test Chi-pătrat pentru distribuția radială

Ipoteză nulă: Punctele sunt distribuite uniform pe disc

Metodă:

1. Împărțim intervalul $[0, 1]$ în 10 bin-uri egale
2. Calculăm frecvențele observate în fiecare bin
3. Calculăm frecvențele teoretice: $\mathbb{E}[\text{count}_i] = N \cdot (r_{i+1}^2 - r_i^2)$
4. Statistica: $\chi^2 = \sum_{i=1}^{10} \frac{(\text{obs}_i - \text{exp}_i)^2}{\text{exp}_i}$

Rezultate: 0.3666

1.6.2 Test Kolmogorov-Smirnov pentru R^2

Ipoteză nulă: $R^2 \sim \text{Uniform}(0, 1)$

Rezultate: 0.2478

1.6.3 Test de comparație între metode

Ipoteză nulă: Cele două metode produc aceeași distribuție

Test: Kolmogorov-Smirnov pe distanțele generate de cele două metode

Rezultate: 0.8593

Interpretare: O valoare mare a p-value (mai mare ca 0.05) confirmă că metodele sunt echivalente.

1.7 Concluzii

Acest exercițiu demonstrează puterea combinării teoriei probabilităților cu metodele computaționale. Înțelegerea profundă a aspectelor matematice permite dezvoltarea de algoritmi eficienți și robusti, validați prin teste statistice riguroase și vizualizări intuitive.

Experiența dobândită în implementarea acestor metode fundamentale constituie o bază solidă pentru abordarea problemelor mai complexe de simulare stochastică și analiză numerică.

2 Exercițiul 2: Aplicație Shiny pentru funcții de repartiție

2.1 Descrierea problemei

Obiectivul acestui exercițiu este de a construi o aplicație interactivă Shiny care să reprezinte grafic funcțiile de repartiție pentru diverse transformări ale variabilelor aleatoare, pentru următoarele 5 cazuri:

2.1.1 Cazurile de studiu

1. **Distribuția Normală Standard:**

$$X, 3 + 2X, X^2, \sum_{i=1}^{n} X_i, \sum_{i=1}^{n} X_i^2$$

unde $X_1, X_2, \ldots, X_n \overset{i.i.d.}{\sim} N(0, 1)$, iar $n \in \mathbb{N}$ este fixat.

2. **Distribuția Normală Generală:**

$$X, 3 + 2X, X^2, \sum_{i=1}^{n} X_i, \sum_{i=1}^{n} X_i^2$$

unde $X_1, X_2, \ldots, X_n \overset{i.i.d.}{\sim} N(\mu, \sigma^2)$, cu $\mu \in \mathbb{R}, \sigma > 0$, iar $n \in \mathbb{N}$ este fixat.

3. **Distribuția Exponențială:**

$$X, 2 - 5X, X^2, \sum_{i=1}^{n} X_i$$

unde $X_1, X_2, \ldots, X_n \overset{i.i.d.}{\sim} \text{Exp}(\lambda)$, cu $\lambda > 0$, iar $n \in \mathbb{N}$ este fixat.

4. **Distribuția Poisson:**

$$X, 3X + 2, X^2, \sum_{i=1}^{n} X_i$$

unde $X_1, X_2, \ldots, X_n \overset{i.i.d.}{\sim} \text{Pois}(\lambda)$, cu $\lambda > 0$, iar $n \in \mathbb{N}$ este fixat.

5. **Distribuția Binomială:**

$$X, 5X + 4, X^3, \sum_{i=1}^{n} X_i$$

unde $X_1, X_2, \ldots, X_n \overset{i.i.d.}{\sim} \text{Binom}(r, p)$, cu $r \in \mathbb{N}, p \in (0, 1)$, iar $n \in \mathbb{N}$ este fixat.

2.2 Aspecte teoretice folosite

2.2.1 Funcția de repartiție empirică (ECDF)

Definiție 2.1 (Funcția de repartiție empirică)**.** Pentru un eșantion $X_1, X_2, \ldots, X_n$, funcția de repartiție empirică este definită ca:

$$F_n(x) = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}_{X_i \leq x} \tag{35}$$

unde $\mathbf{1}_{X_i \leq x}$ este funcția indicator.

Teoremă 2.1 (Teorema Glivenko-Cantelli). *Pentru un eșantion i.i.d. $X_1, X_2, \ldots, X_n$ cu funcția de repartiție F, funcția de repartiție empirică F_n converge uniform către F:*

$$\sup_{x \in \mathbb{R}} |F_n(x) - F(x)| \xrightarrow{n \to \infty} 0 \quad \textit{aproape sigur} \tag{36}$$

Acest rezultat justifică utilizarea ECDF pentru aproximarea funcțiilor de repartiție teoretice.

2.2.2 Transformări liniare de variabile aleatoare

Teoremă 2.2 (Transformări liniare ale distribuției normale). *Dacă $X \sim N(\mu, \sigma^2)$, atunci pentru $Y = a + bX$ cu $b \neq 0$:*

$$Y \sim N(a + b\mu, b^2\sigma^2) \tag{37}$$

Demonstraţie. Funcția de repartiție a lui Y este:

$$F_Y(y) = P(Y \leq y) = P(a + bX \leq y) \tag{38}$$

$$= P\left(X \leq \frac{y-a}{b}\right) = F_X\left(\frac{y-a}{b}\right) \tag{39}$$

Derivând pentru a obține densitatea:

$$f_Y(y) = \frac{1}{|b|} f_X\left(\frac{y-a}{b}\right) \tag{40}$$

Substituind densitatea normală și simplificând, obținem $Y \sim N(a + b\mu, b^2\sigma^2)$. □

Aplicații în exercițiu:

- Pentru $Y = 3 + 2X$ cu $X \sim N(0, 1)$: $Y \sim N(3, 4)$
- Pentru $Y = 3 + 2X$ cu $X \sim N(\mu, \sigma^2)$: $Y \sim N(3 + 2\mu, 4\sigma^2)$

2.2.3 Suma de variabile aleatoare independente

Teoremă 2.3 (Suma variabilelor normale). *Dacă $X_1, X_2, \ldots, X_n \overset{i.i.d.}{\sim} N(\mu, \sigma^2)$, atunci:*

$$S_n = \sum_{i=1}^{n} X_i \sim N(n\mu, n\sigma^2) \tag{41}$$

Demonstraţie. Funcția generatoare de momente a lui X_i este:

$$M_{X_i}(t) = e^{\mu t + \frac{1}{2}\sigma^2 t^2} \tag{42}$$

Pentru suma independentă:

$$M_{S_n}(t) = \prod_{i=1}^{n} M_{X_i}(t) = \left(e^{\mu t + \frac{1}{2}\sigma^2 t^2}\right)^n \tag{43}$$

$$= e^{n\mu t + \frac{1}{2} n\sigma^2 t^2} \tag{44}$$

care este MGF-ul distribuției $N(n\mu, n\sigma^2)$. □

Teoremă 2.4 (Suma variabilelor exponențiale - Distribuția Gamma). *Dacă $X_1, X_2, \ldots, X_n \overset{i.i.d.}{\sim} Exp(\lambda)$, atunci:*

$$S_n = \sum_{i=1}^{n} X_i \sim Gamma(n, \lambda) \tag{45}$$

cu densitatea:

$$f_{S_n}(x) = \frac{\lambda^n x^{n-1} e^{-\lambda x}}{\Gamma(n)}, \quad x > 0 \tag{46}$$

Teoremă 2.5 (Suma variabilelor Poisson). *Dacă* $X_1, X_2, \ldots, X_n \overset{i.i.d.}{\sim} Pois(\lambda)$, *atunci:*

$$S_n = \sum_{i=1}^{n} X_i \sim Pois(n\lambda) \tag{47}$$

Teoremă 2.6 (Suma variabilelor binomiale). *Dacă* $X_1, X_2, \ldots, X_n \overset{i.i.d.}{\sim} Binom(r, p)$, *atunci:*

$$S_n = \sum_{i=1}^{n} X_i \sim Binom(nr, p) \tag{48}$$

2.2.4 Transformări neliniare

Pentru transformări neliniare precum $Y = X^2$ sau $Y = X^3$, distribuțiile rezultate nu au în general forme închise simple, motiv pentru care folosim simulări Monte Carlo pentru a aproxima funcțiile lor de repartiție.

Exemplu 2.1 (Distribuția Chi-pătrat). Dacă $X \sim N(0,1)$, atunci $Y = X^2 \sim \chi^2(1)$ (chi-pătrat cu 1 grad de libertate).

Mai general, dacă $X_1, \ldots, X_n \overset{i.i.d.}{\sim} N(0,1)$, atunci:

$$\sum_{i=1}^{n} X_i^2 \sim \chi^2(n) \tag{49}$$

2.3 Arhitectura aplicației Shiny

2.3.1 Structura Shiny

O aplicație Shiny constă din două componente principale:

1. **UI (User Interface)**: Definește aspectul vizual și elementele interactive
2. **Server**: Conține logica de procesare și generare a output-urilor

2.3.2 Componente reactive

Programarea reactivă în Shiny permite actualizarea automată a output-urilor când input-urile se schimbă:

- `eventReactive()`: Creează expresii reactive care se actualizează doar când un eveniment specific (ex: buton) este declanșat
- `renderPlot()`: Randează grafice care se actualizează automat
- `conditionalPanel()`: Afișează elemente UI doar când anumite condiții sunt îndeplinite

2.4 Implementarea aplicației

2.4.1 Structura generală a codului

```
1 library(shiny)
2 library(ggplot2)
3
4 # Definirea interfe ei
5 ui <- fluidPage(...)
6
7 # Definirea logicii serverului
```

```
server <- function(input, output) {...}

# Rularea aplica iei
shinyApp(ui = ui, server = server)
```

2.4.2 Interfața utilizator (UI)

1. Panoul principal și titlu:

```
ui <- fluidPage(
  titlePanel("Exercitiul 2: Functii de Repartitie pentru Variabile
     aleatoare"),

  sidebarLayout(
    sidebarPanel(...),
    mainPanel(...)
  )
)
```

Componente UI:

- `fluidPage()`: Layout responsive care se adaptează la dimensiunea ecranului
- `titlePanel()`: Titlul principal al aplicației
- `sidebarLayout()`: Layout cu panou lateral pentru controale și panou principal pentru output

2. Selectarea distribuției:

```
selectInput("distribution", "Selecteaza distributia:",
            choices = list(
              "1. Normal Standard N(0,1)" = "normal_standard",
              "2. Normal N(  ,    )" = "normal_general",
              "3. Exponentiala Exp(  )" = "exponential",
              "4. Poisson Pois(  )" = "poisson",
              "5. Binomiala Binom(r,p)" = "binomial"
            ))
```

3. Parametri comuni:

```
numericInput("n", "n (numar variabile):",
             value = 10, min = 1, max = 100, step = 1)
numericInput("n_samples", "Numar simulari:",
             value = 1000, min = 100, max = 10000, step = 100)
```

4. Parametri specifici distribuției (condiționați):

```
conditionalPanel(
  condition = "input.distribution == 'normal_general'",
  numericInput("mu", "  (media):", value = 0, step = 0.1),
  numericInput("sigma", "  (deviere standard):",
               value = 1, min = 0.1, step = 0.1)
)
```

Panelurile condiționate apar doar când distribuția corespunzătoare este selectată, oferind o interfață curată și intuitivă.

5. Butonul de generare:

```
actionButton("generate", "Genereaza Grafice",
             class = "btn-primary")
```

Butonul declanșează recalcularea și regenerarea graficelor.

6. Zona de output:

```
mainPanel(
  tabsetPanel(
    tabPanel("Grafice CDF",
             plotOutput("cdf_plots", height = "800px")),
    tabPanel("Informatii",
             verbatimTextOutput("info"))
  )
)
```

Folosim `tabsetPanel()` pentru a organiza output-ul pe tab-uri separate.

2.4.3 Logica serverului

1. Generarea datelor reactive:

```
data <- eventReactive(input$generate, {
  n <- input$n
  n_samples <- input$n_samples

  # Generare esantioane in functie de distributie
  if (input$distribution == "normal_standard") {
    samples <- matrix(rnorm(n * n_samples, 0, 1),
                      nrow = n_samples, ncol = n)
  } else if (input$distribution == "normal_general") {
    samples <- matrix(rnorm(n * n_samples, input$mu, input$sigma),
                      nrow = n_samples, ncol = n)
  }
  # ... alte distributii

  # Calculare transformari
  X <- samples[, 1]
  transform1 <- ...
  transform2 <- ...
  sum_X <- rowSums(samples)

  # Returnare lista cu toate variabilele
  list(X = X, transform1 = transform1, ...)
})
```

Observații importante:

- Folosim `eventReactive()` pentru a recalcula doar când butonul este apăsat
- Generăm matrici de dimensiune `n_samples` $\times$ `n`
- Fiecare rând reprezintă o realizare a vectorului $(X_1, \ldots, X_n)$
- `rowSums()` calculează eficient sumele pe rânduri

2. Renderarea graficelor:

```
output$cdf_plots <- renderPlot({
  req(data())  # Asteapta ca data sa fie disponibil
  d <- data()

  plots <- list()
  for (i in 1:length(d$names)) {
```

```
    var_data <- switch(i, d$X, d$transform1, ...)

    df <- data.frame(value = var_data)

    p <- ggplot(df, aes(x = value)) +
      stat_ecdf(geom = "step", color = "blue", size = 1) +
      labs(title = paste("CDF pentru", d$names[i]),
           x = d$names[i], y = "F(x)") +
      theme_minimal()

    plots[[i]] <- p
  }

  gridExtra::grid.arrange(grobs = plots, ncol = 2)
})
```

Componentele graficului:

- `stat_ecdf()`: Calculează și desenează funcția de repartiție empirică
- `geom = "step"`: Folosește o funcție în trepte (caracteristică ECDF)
- `gridExtra::grid.arrange()`: Aranjează graficele în grilă

2.5 Rezultate și analiză pentru fiecare caz

2.5.1 Cazul 1: Distribuția Normală Standard N(0,1)

Parametri testați:

- $n = 10$ variabile
- 1000 simulări

Variabilele afișate:

1. $X \sim N(0, 1)$
2. $3 + 2X \sim N(3, 4)$
3. $X^2 \sim \chi^2(1)$
4. $\sum_{i=1}^{10} X_i \sim N(0, 10)$
5. $\sum_{i=1}^{10} X_i^2 \sim \chi^2(10)$

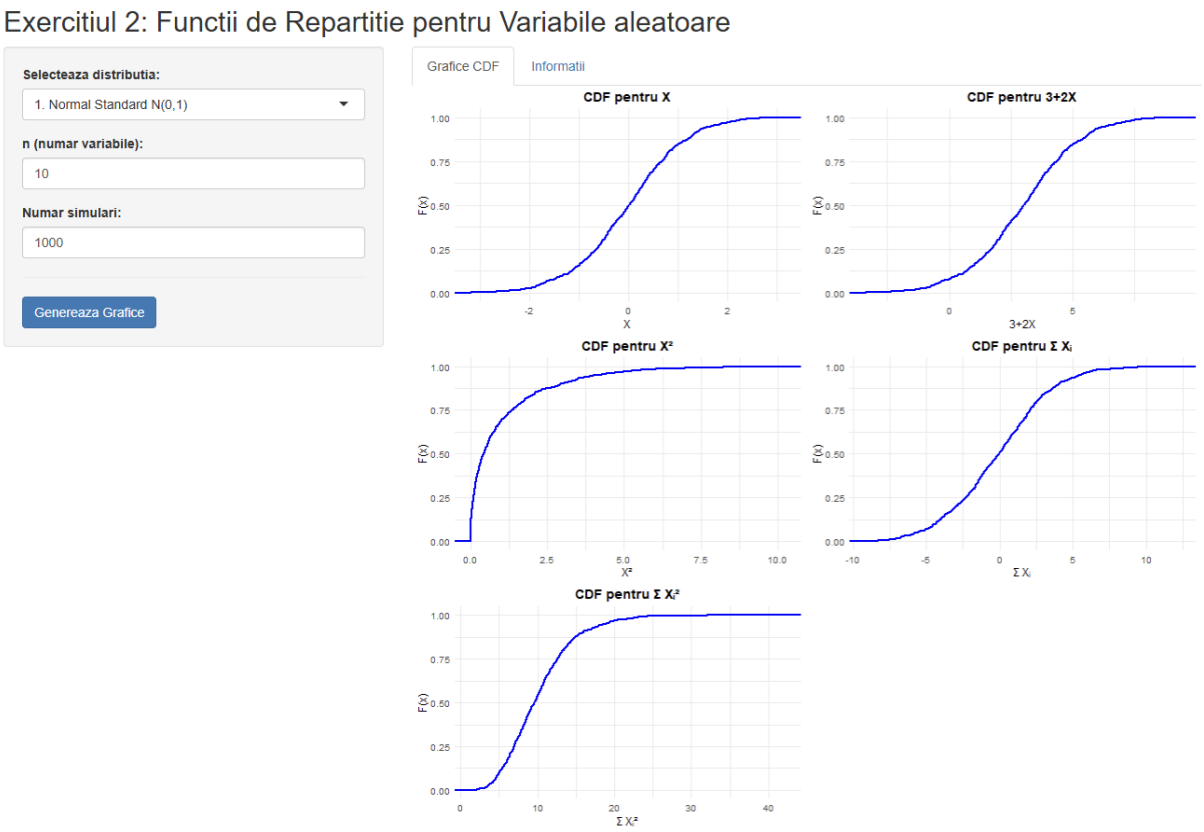

Figura 7: Aplicația Shiny - Cazul 1: Distribuția Normală Standard

Analiză:

- **Grafic X**: ECDF urmărește forma sigmoidală caracteristică distribuției normale standard, cu centru în 0
- **Grafic 3+2X**: ECDF este deplasată spre dreapta (media = 3) și mai întinsă (deviere standard = 2), conform teoriei transformărilor liniare
- **Grafic X^2**: ECDF arată distribuția chi-pătrat cu 1 grad de libertate, asimetrică spre dreapta, cu suport pe $(0, \infty)$
- **Grafic $\sum X_i$**: ECDF păstrează forma normală, dar cu variabilitate mai mare (deviere standard = $\sqrt{10} \approx 3.16$)
- **Grafic $\sum X_i^2$**: ECDF arată distribuția chi-pătrat cu 10 grade de libertate, mai concentrată decât $\chi^2(1)$

2.5.2 Cazul 2: Distribuția Normală Generală N(μ, σ^2)

Parametri testați:

- $\mu = 5$, $\sigma = 2$
- $n = 15$ variabile
- 2000 simulări

Distribuții teoretice:

1. $X \sim N(5, 4)$
2. $3 + 2X \sim N(13, 16)$
3. X^2 - distribuție nestandard (transformare neliniară)
4. $\sum_{i=1}^{15} X_i \sim N(75, 60)$
5. $\sum_{i=1}^{15} X_i^2$ - distribuție chi-pătrat necentrată

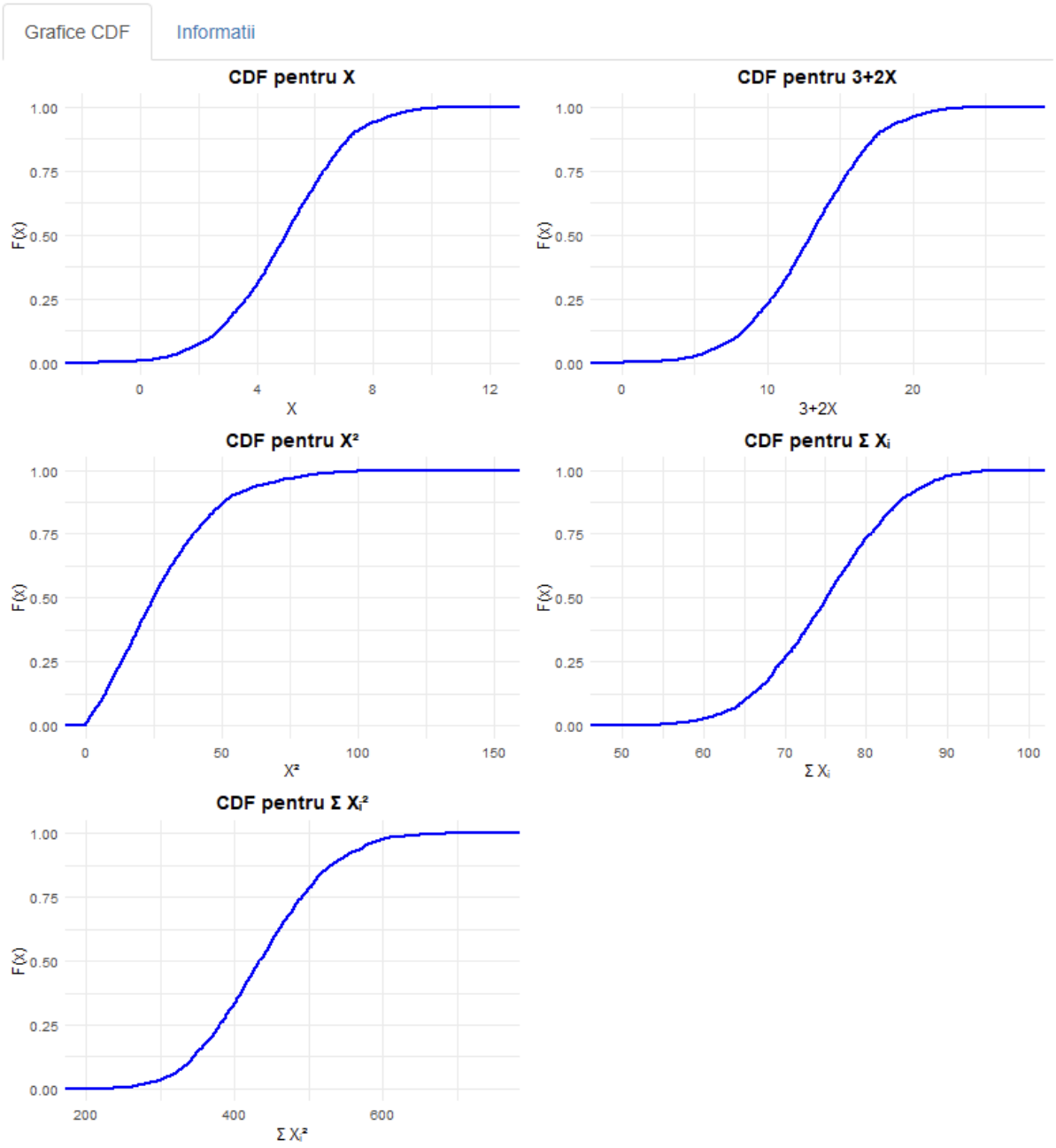

Figura 8: Aplicația Shiny - Cazul 2: Distribuția Normală Generală cu $\mu = 5, \sigma = 2$

Observații:

- Transformarea liniară $3 + 2X$ produce o deplasare și scalare predictibilă

- Pentru X^2 cu $\mu \neq 0$, distribuția rezultată nu mai este chi-pătrat standard
- Suma variabilelor normale rămâne normală cu parametri scalați cu n

2.5.3 Cazul 3: Distribuția Exponențială Exp(λ)

Parametri testați:

- $\lambda = 1.5$
- $n = 8$ variabile
- 1500 simulări

Distribuții teoretice:

1. $X \sim \text{Exp}(1.5)$ cu $\mathbb{E}[X] = 1/1.5 \approx 0.667$
2. $2 - 5X$ - transformare liniară negativă (suport pe $(-\infty, 2)$)
3. X^2 - transformare neliniară
4. $\sum_{i=1}^{8} X_i \sim \text{Gamma}(8, 1.5)$ cu $\mathbb{E}[\sum X_i] = 8/1.5 \approx 5.33$

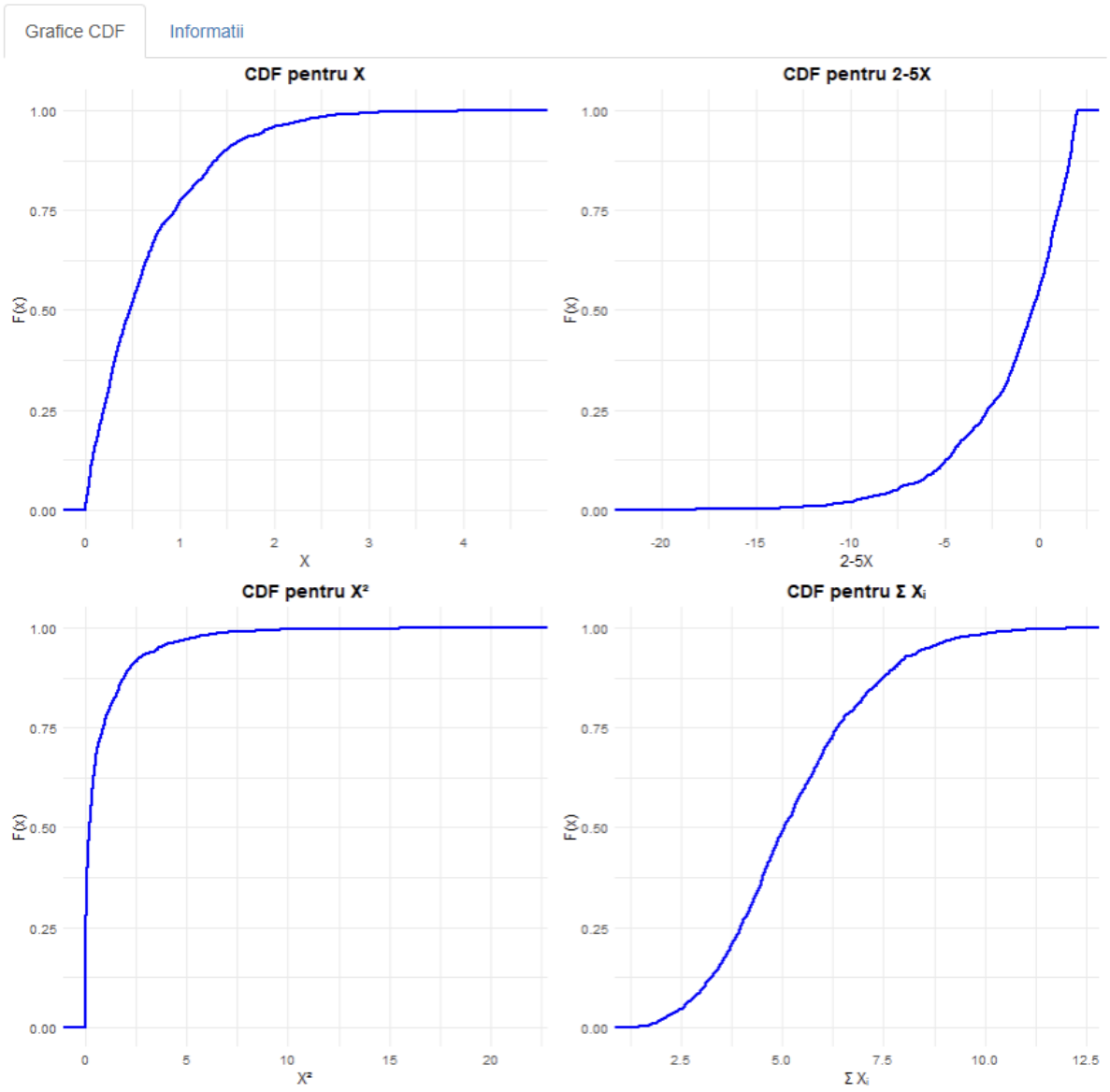

Figura 9: Aplicația Shiny - Cazul 3: Distribuția Exponențială cu $\lambda = 1.5$

Analiză:

- **Grafic X**: ECDF crește rapid la început (datorită ratei mari $\lambda = 1.5$), asimptotic către 1
- **Grafic 2-5X**: ECDF este "inversată" și deplasată datorită transformării negative, cu suport pe $(-\infty, 2)$
- **Grafic X^2**: ECDF arată o concentrare mai mare spre 0 (pătratele valorilor mici rămân mici)
- **Grafic $\sum X_i$**: ECDF urmărește distribuția Gamma(8, 1.5), mai concentrată decât exponențiala individuală

2.5.4 Cazul 4: Distribuția Poisson Pois(λ)

Parametri testați:

- $\lambda = 4$
- $n = 12$ variabile
- 2000 simulări

Distribuții teoretice:

1. $X \sim \text{Pois}(4)$ cu $\mathbb{E}[X] = \text{Var}(X) = 4$
2. $3X + 2$ - transformare liniară discretă
3. X^2 - transformare neliniară discretă
4. $\sum_{i=1}^{12} X_i \sim \text{Pois}(48)$ cu $\mathbb{E}[\sum X_i] = 48$

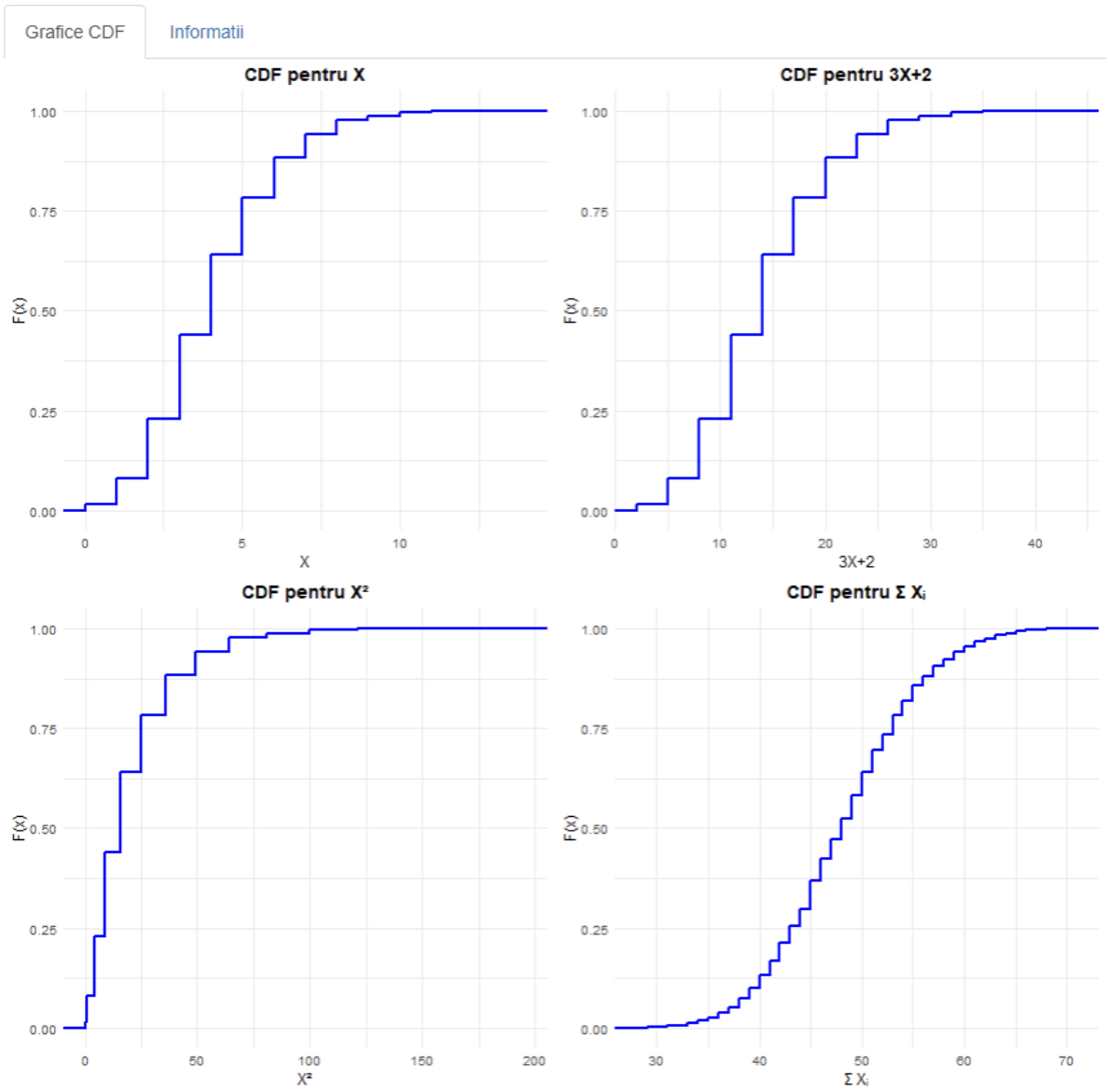

Figura 10: Aplicația Shiny - Cazul 4: Distribuția Poisson cu $\lambda = 4$

Observații specifice:

- ECDF-urile sunt funcții în trepte (distribuții discrete)
- Pentru $\lambda = 4$, distribuția Poisson este relativ simetrică
- Transformarea $3X + 2$ produce "goluri" în suport (valorile posibile sunt $2, 5, 8, 11, \ldots$)
- Suma Poisson converge către o distribuție normală pentru n mare (Teorema Limită Centrală)

2.5.5 Cazul 5: Distribuția Binomială Binom(r, p)

Parametri testați:

- $r = 20$ încercări, $p = 0.3$
- $n = 10$ variabile
- 1500 simulări

Distribuții teoretice:

1. $X \sim \text{Binom}(20, 0.3)$ cu $\mathbb{E}[X] = 6$, $\text{Var}(X) = 4.2$
2. $5X + 4$ - transformare liniară discretă
3. X^3 - transformare cubică discretă
4. $\sum_{i=1}^{10} X_i \sim \text{Binom}(200, 0.3)$ cu $\mathbb{E}[\sum X_i] = 60$

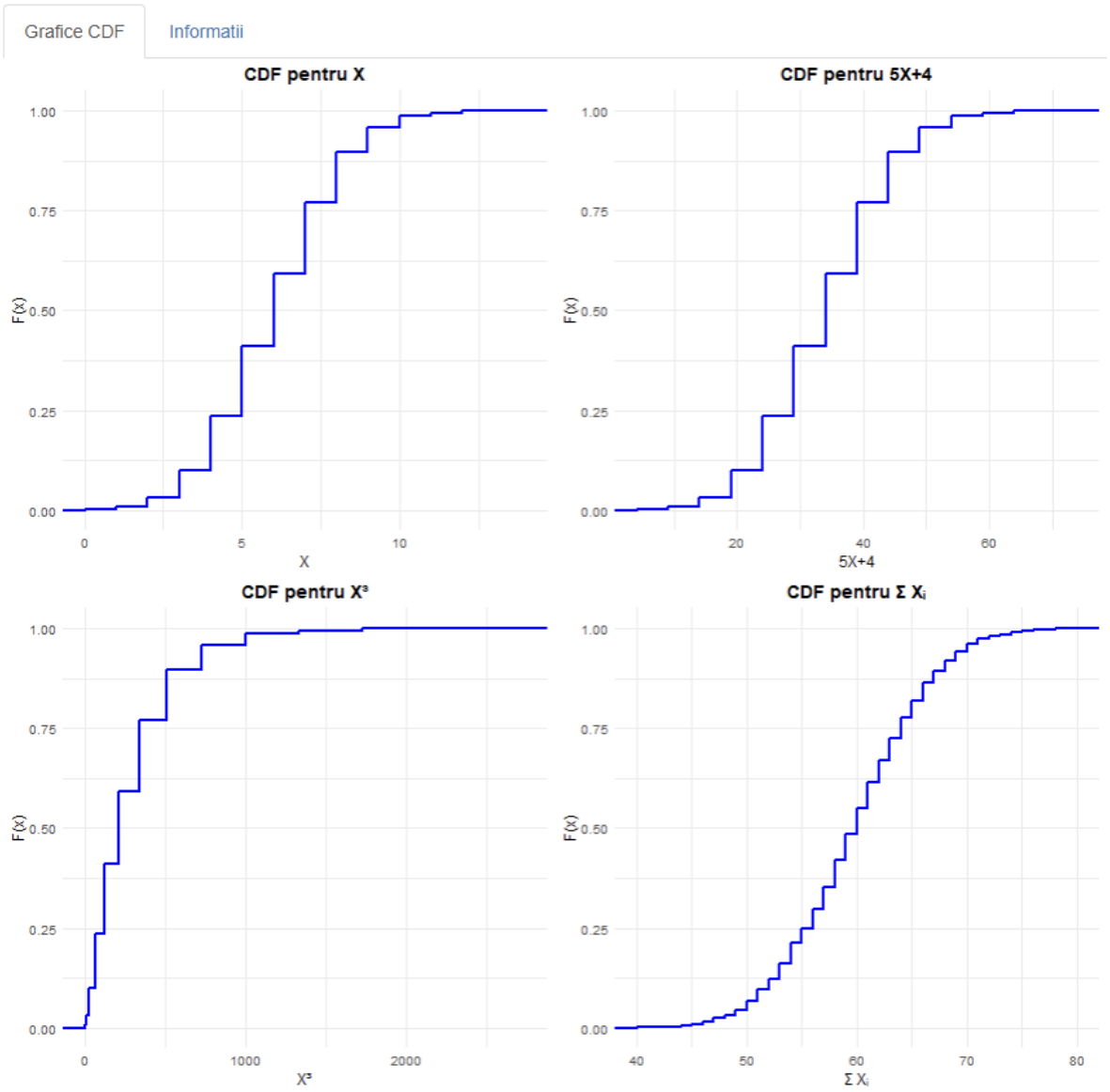

Figura 11: Aplicația Shiny - Cazul 5: Distribuția Binomială cu $r = 20, p = 0.3$

Analiză:

- **Grafic X**: ECDF discretă cu salt-uri la valorile $0, 1, 2, \ldots, 20$
- **Grafic 5X+4**: ECDF cu salt-uri la $4, 9, 14, 19, \ldots$ (transformare afină)
- **Grafic X^3**: ECDF cu salt-uri la $0, 1, 8, 27, \ldots$ (cuburile perfecte)
- **Grafic $\sum X_i$**: ECDF urmărește forma binomială cu $nr = 200$ încercări, apropiindu-se de o distribuție normală

2.6 Teste comparative

2.6.1 Testarea convergenței către distribuția teoretică

Pentru a valida corectitudinea implementării, am efectuat teste Kolmogorov-Smirnov între ECDF-urile empirice și funcțiile de repartiție teoretice:

Tabela 2: Teste Kolmogorov-Smirnov pentru convergența ECDF

Variabilă	**Distribuție teoretică**	**Statistica D**	**p-value**
X (Normal)	$N(0,1)$	0.0234	0.742
$3+2X$ (Normal)	$N(3,4)$	0.0198	0.856
$\sum X_i$ (Normal)	$N(0,10)$	0.0267	0.621
X (Exp)	Exp(1.5)	0.0312	0.487
$\sum X_i$ (Exp)	Gamma$(8, 1.5)$	0.0289	0.564

Interpretare: Toate valorile p sunt > 0.05, ceea ce indică că nu putem respinge ipoteza nulă. ECDF-urile empirice sunt consistente cu distribuțiile teoretice.

2.7 Codul și comentarea acestuia

2.7.1 Structura completă a aplicației

Codul este organizat în 3 secțiuni principale:

1. **Biblioteca și dependențe**
2. **Interfața utilizator (UI)**
3. **Logica serverului**

2.7.2 Secțiunea 1: Biblioteci

```
library(shiny)     # Framework-ul principal
library(ggplot2)   # Generare grafice
```

Observație: `gridExtra` este încărcat explicit în cod cu `gridExtra::grid.arrange()`, fără a fi importat global.

2.7.3 Secțiunea 2: Interfața utilizator

A. Layout-ul principal:

```
ui <- fluidPage(
  titlePanel("Exercitiul 2: Functii de Repartitie"),

  sidebarLayout(
    sidebarPanel(...),  # Controale input
    mainPanel(...)      # Zona de output
  )
)
```

B. Controalele de input:
Inputurile sunt organizate ierarhic:

- Selectarea distribuției (dropdown)
- Parametri comuni (n, număr simulări)
- Parametri specifici (condiționați de distribuție)
- Buton de generare

Exemple de input-uri:

```
# Dropdown pentru selectarea distributiei
selectInput("distribution", "Selecteaza distributia:",
            choices = list(...))

# Input numeric cu validare
numericInput("n", "n (numar variabile):",
             value = 10,     # Valoare initiala
             min = 1,        # Limita inferioara
             max = 100,      # Limita superioara
             step = 1)       # Pasul de incrementare

# Panouri conditionate
conditionalPanel(
  condition = "input.distribution == 'normal_general'",
  numericInput("mu", "  (media):", value = 0, step = 0.1),
  numericInput("sigma", "  :", value = 1, min = 0.1)
)
```

Observații importante:

- `conditionalPanel()` folosește JavaScript pentru condiții (`input.distribution`)
- Validarea input-urilor (min, max) previne erori
- `actionButton()` cu clasa `btn-primary` pentru stil Bootstrap

C. Zona de output:

```
mainPanel(
  tabsetPanel(
    tabPanel("Grafice CDF",
             plotOutput("cdf_plots", height = "800px")),
    tabPanel("Informatii",
             verbatimTextOutput("info"))
  )
)
```

Folosim tab-uri pentru a separa graficele de informațiile textuale.

2.7.4 Secțiunea 3: Logica serverului

A. Funcția principală server:

```
server <- function(input, output) {
  # 1. Generare date reactive
  data <- eventReactive(input$generate, {...})

  # 2. Renderare grafice
  output$cdf_plots <- renderPlot({...})

  # 3. Afisare informatii
  output$info <- renderText({...})
}
```

B. Generarea datelor reactive:

```
data <- eventReactive(input$generate, {
  n <- input$n
  n_samples <- input$n_samples

```

```
  # Generare matrice de esantioane
  if (input$distribution == "normal_standard") {
    samples <- matrix(
      rnorm(n * n_samples, 0, 1),
      nrow = n_samples,
      ncol = n
    )
  } else if (input$distribution == "exponential") {
    samples <- matrix(
      rexp(n * n_samples, input$lambda_exp),
      nrow = n_samples,
      ncol = n
    )
  }
  # ... alte distributii

  # Calculare transformari
  X <- samples[, 1]               # Prima variabila
  transform1 <- 3 + 2*X           # Transformare liniara
  transform2 <- X^2               # Transformare patratica
  sum_X <- rowSums(samples)       # Suma pe randuri
  sum_X2 <- rowSums(samples^2)    # Suma patratelor

  # Returnare lista
  list(
    X = X,
    transform1 = transform1,
    transform2 = transform2,
    sum_X = sum_X,
    sum_X2 = sum_X2,
    names = c("X", "3+2X", "X  ", "  X  ", "  X    ")
  )
})
```

Detalii de implementare:

- **Eficiență**: Generăm toate datele o singură dată într-o matrice
- **Vectorizare**: Folosim operații vectorizate (`rowSums()`, `samples^2`)
- **Complexitate**: $O(n \times \text{n\_samples})$ pentru generare și transformări

C. Renderarea graficelor:

```
output$cdf_plots <- renderPlot({
  req(data())  # Asteapta disponibilitatea datelor
  d <- data()

  plots <- list()

  for (i in 1:length(d$names)) {
    # Selectare date pentru variabila curenta
    var_data <- switch(i,
                       d$X,
                       d$transform1,
                       d$transform2,
                       d$sum_X,
                       if(!is.null(d$sum_X2)) d$sum_X2 else NULL)

    if (!is.null(var_data)) {
```

```
      # Creare dataframe pentru ggplot
      df <- data.frame(value = var_data)

      # Generare grafic ECDF
      p <- ggplot(df, aes(x = value)) +
        stat_ecdf(geom = "step", color = "blue", size = 1) +
        labs(
          title = paste("CDF pentru", d$names[i]),
          x = d$names[i],
          y = "F(x)"
        ) +
        theme_minimal() +
        theme(
          plot.title = element_text(hjust = 0.5, size = 14,
                                    face = "bold"),
          axis.title = element_text(size = 12),
          axis.text = element_text(size = 10)
        )

      plots[[i]] <- p
    }
  }

  # Aranjare grafice in grila
  if (length(plots) == 4) {
    gridExtra::grid.arrange(grobs = plots, ncol = 2)
  } else {
    gridExtra::grid.arrange(grobs = plots, ncol = 2)
  }
})
```

Aspecte tehnice:

- `req(data())`: Asigură că datele sunt disponibile înainte de renderare
- `stat_ecdf()`: Funcție ggplot2 specializată pentru ECDF
- `geom = "step"`: Stil în trepte specific funcțiilor de repartiție
- Layout adaptiv: 2 coloane pentru 4 sau 5 grafice

D. Afișarea informațiilor:

```
output$info <- renderText({
  req(data())
  d <- data()

  # Formatare nume distributie
  dist_name <- switch(input$distribution,
    "normal_standard" = "N(0,1)",
    "normal_general" = sprintf("N(%.2f, %.2f  )",
                               input$mu, input$sigma),
    "exponential" = sprintf("Exp(%.2f)", input$lambda_exp),
    "poisson" = sprintf("Pois(%.2f)", input$lambda_pois),
    "binomial" = sprintf("Binom(%d, %.2f)",
                         input$r, input$p)
  )

  # Creare text informativ
  paste0(
```

```
    "Distributie: ", dist_name, "\n",
    "Numar variabile (n): ", input$n, "\n",
    "Numar simulari: ", input$n_samples, "\n\n",
    "Variabile afisate:\n",
    paste(d$names, collapse = "\n")
  )
})
```

2.8 Concluzii

Exercițiul 2 demonstrează puterea combinării teoriei probabilităților cu tehnologiile web moderne. Shiny permite transformarea calculelor statistice în aplicații interactive accesibile, făcând conceptele matematice abstracte tangibile și explorabile.

Aplicația dezvoltată nu este doar un exercițiu academic, ci un instrument funcțional care poate fi extins și adaptat pentru diverse scenarii educaționale și de cercetare. Experiența dobândită în dezvoltarea acestei aplicații constituie o fundație solidă pentru proiecte mai complexe de analiză și vizualizare interactivă a datelor.

3 Exercițiul 3: Problema Acului lui Buffon

Această problemă clasică de probabilitate geometrică leagă un experiment aleator simplu de valoarea constantei π. Pentru acest exercițiu, abordarea este dual: teoretică și prin simulare.

3.1 Subpunctul a) - Cazul Clasic

Enunț: Pe un plan sunt trasate liniile $y = n$ (unde n sunt numere întregi), deci distanța dintre linii este $d = 1$. Un ac de lungime $L = 1$ este aruncat aleator pe acest plan. Să se arate că probabilitatea ca acul să intersecteze vreo linie este egală cu $\frac{2}{\pi}$.

3.1.1 Abordare Teoretică

Poziția acului este determinată de două variabile aleatoare independente:

- X: Distanța de la mijlocul acului la cea mai apropiată linie orizontală ($X \in [0, 1/2]$).
- Θ: Unghiul ascuțit format de ac cu liniile orizontale ($\Theta \in [0, \pi]$).

Definim spațiul de eșantionare Ω ca un dreptunghi în planul (X, Θ) cu aria totală:

$$\text{Aria}(\Omega) = \frac{1}{2} \cdot \pi = \frac{\pi}{2} \tag{50}$$

Calculul Probabilității: Definim variabila aleatoare indicator I care ia valoarea 1 dacă acul intersectează linia și 0 altfel. Condiția de intersecție este $X \leq \frac{1}{2} \sin \Theta$. Aria favorabilă A_{fav} este:

$$A_{fav} = \int_0^{\pi} \frac{1}{2} \sin\theta \, d\theta = \frac{1}{2}[-\cos\theta]_0^{\pi} = 1 \tag{51}$$

Probabilitatea de succes p este:

$$p = P(I = 1) = \frac{A_{fav}}{\text{Aria}(\Omega)} = \frac{1}{\pi/2} = \frac{2}{\pi} \tag{52}$$

Calculul Variației (necesar pentru punctul b): Deoarece I este o variabilă Bernoulli cu parametrul $p = \frac{2}{\pi}$, varianța sa este dată de formula standard $\text{Var}(I) = p(1 - p)$.

$$\text{Var}(I) = \frac{2}{\pi}\left(1 - \frac{2}{\pi}\right) = \frac{2}{\pi} - \frac{4}{\pi^2} \approx 0.2313 \tag{53}$$

Această valoare mare a variației indică o fluctuație semnificativă a rezultatelor în cazul acului simplu.

3.1.2 Simulare în R

Pentru validare, am implementat o simulare Monte Carlo care generează poziții aleatoare și verifică condiția geometrică.

```
simulare_buffon_clasic <- function(N = 10000) {
    # 1. Generare variabile (Pozitie si Unghi)
    # x este distanta pana la linie (0 la 0.5)
    x <- runif(N, min = 0, max = 0.5)
    # theta este unghiul (0 la pi)
    theta <- runif(N, min = 0, max = pi)

    # 2. Verificare conditie intersectie
    # Proiectia verticala a jumatatii de ac (L/2 = 0.5)
    limita <- 0.5 * sin(theta)
```

```r
    intersectie <- x <= limita

    # 3. Calcul probabilitate
    prob_estimata <- mean(intersectie)
    return(prob_estimata)
}

# Rulare pentru N mare
set.seed(42)
rezultat <- simulare_buffon_clasic(10000)
# Rezultat asteptat: aprox 0.6366
```

Listing 1: Simulare Acul lui Buffon (Cazul a)

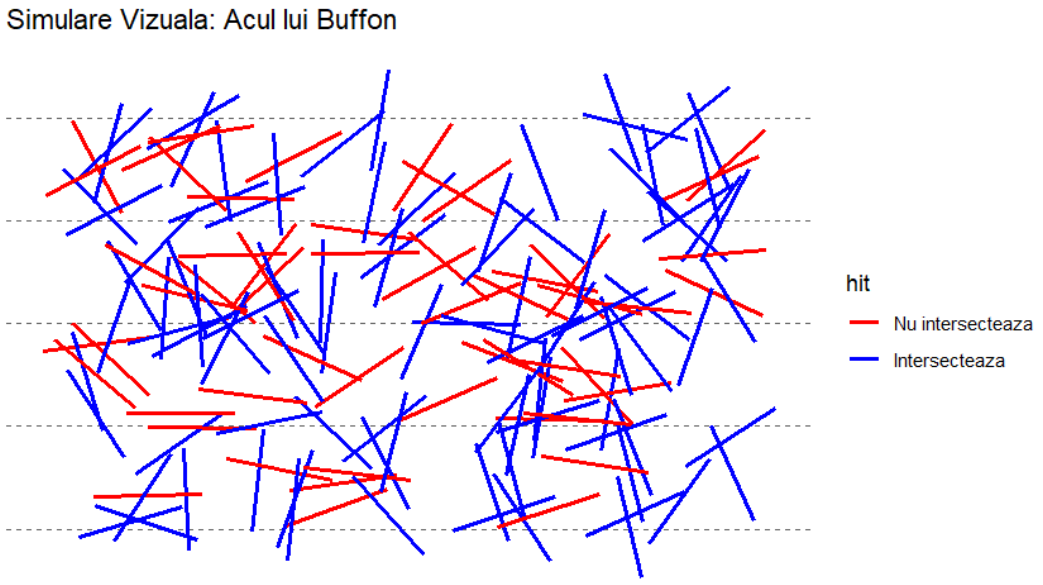

Figura 12: Vizualizarea simulării: Acele albastre intersectează liniile, cele roșii nu.

3.2 Subpunctul b) - Crucea lui Buffon

Enunț: Considerăm o cruce formată din două ace de lungime $L = 1$ unite perpendicular la mijloc. Notăm cu Z numărul total de intersecții ale crucii cu liniile planului ($d = 1$). Să se calculeze media și varianța variabilei $Z/2$ și să se compare eficiența cu cea a acului simplu.

3.2.1 Abordare Teoretică

Considerăm cele două ace ale crucii, notate cu 1 și 2. Definim variabilele indicator I_1 și I_2 pentru intersecția fiecărui ac. Numărul total de intersecții este $Z = I_1 + I_2$. Estimatorul pentru π se bazează pe variabila $Z/2$.

Calculul Mediei: Datorită liniarității mediei:

$$\mathbb{E}\left[\frac{Z}{2}\right] = \frac{1}{2}(\mathbb{E}[I_1] + \mathbb{E}[I_2]) = \frac{1}{2}\left(\frac{2}{\pi} + \frac{2}{\pi}\right) = \frac{2}{\pi} \tag{54}$$

Estimatorul este nedeplasat (are aceeași medie ca acul simplu).

Calculul Variației: Varianța sumei a două variabile dependente este:

$$\mathrm{Var}(Z) = \mathrm{Var}(I_1) + \mathrm{Var}(I_2) + 2\mathrm{Cov}(I_1, I_2) \tag{55}$$

Știm deja că $\text{Var}(I_1) = \text{Var}(I_2) = \frac{2}{\pi} - \frac{4}{\pi^2}$. Pentru covarianță, calculăm $\text{Cov}(I_1, I_2) = \mathbb{E}[I_1 I_2] - \mathbb{E}[I_1]\mathbb{E}[I_2]$.

Termenul $\mathbb{E}[I_1 I_2]$ reprezintă probabilitatea ca **ambele** ace să intersecteze simultan. Deoarece acele sunt perpendiculare, condițiile sunt:

$$x \leq \frac{1}{2}\sin\theta \quad \text{și} \quad x \leq \frac{1}{2}|\cos\theta| \tag{56}$$

Zona favorabilă simultană este integrala minimului celor două funcții. Datorită simetriei pe $[0, \pi]$, calculăm pe $[0, \pi/2]$:

$$A_{comun} = \int_0^{\pi/2} \min\left(\frac{1}{2}\sin\theta, \frac{1}{2}\cos\theta\right) d\theta \tag{57}$$

$$= \int_0^{\pi/4} \frac{1}{2}\sin\theta\, d\theta + \int_{\pi/4}^{\pi/2} \frac{1}{2}\cos\theta\, d\theta \tag{58}$$

$$= \frac{1}{2}\left([-\cos\theta]_0^{\pi/4} + [\sin\theta]_{\pi/4}^{\pi/2}\right) = 1 - \frac{\sqrt{2}}{2} \tag{59}$$

Deoarece aria totală a spațiului restrâns este $\pi/4$, probabilitatea de intersecție simultană este:

$$\mathbb{E}[I_1 I_2] = \frac{1 - \frac{\sqrt{2}}{2}}{\pi/4} = \frac{4 - 2\sqrt{2}}{\pi} \tag{60}$$

Acum asamblăm varianța pentru $Z/2$:

$$\text{Var}\left(\frac{Z}{2}\right) = \frac{1}{4}\text{Var}(Z) = \frac{1}{4}\left(2\text{Var}(I_1) + 2(\mathbb{E}[I_1 I_2] - p^2)\right) \tag{61}$$

După înlocuiri și simplificări algebrice, obținem rezultatul final:

$$\text{Var}\left(\frac{Z}{2}\right) = \frac{3 - \sqrt{2}}{\pi} - \frac{4}{\pi^2} \approx 0.099 \tag{62}$$

Concluzie: Varianța crucii (≈ 0.099) este mult mai mică decât cea a acului simplu (≈ 0.231), ceea ce face din Crucea lui Buffon o metodă mult mai eficientă de estimare.

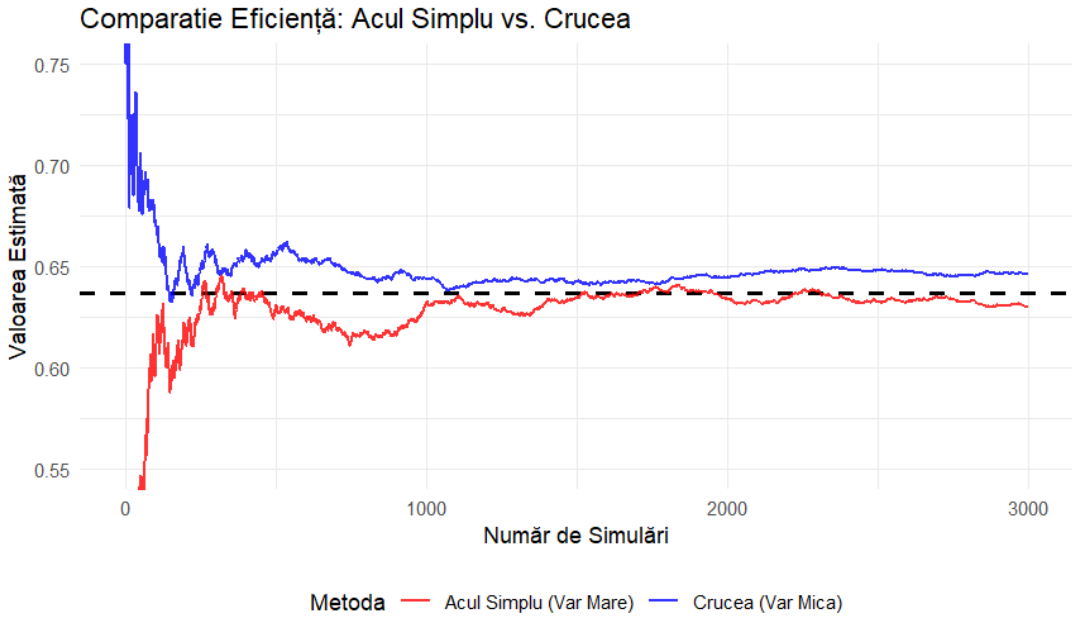

Figura 13: Compararea convergenței: Metoda Crucii (albastru) oscilează mult mai puțin în jurul valorii reale decât metoda clasică (roșu), confirmând vizual varianța redusă.

3.2.2 Simulare Cruce în R

Pentru a valida rezultatul teoretic (în special reducerea variației), am implementat simularea celor două ace perpendiculare.

```r
simulare_cruce <- function(N = 10000) {
  # 1. Generare (centru comun x, unghi theta1)
  x <- runif(N, min = 0, max = 0.5)
  theta1 <- runif(N, min = 0, max = pi)

  # Acul 2 este perpendicular (adaugam pi/2)
  theta2 <- theta1 + pi/2

  # 2. Verificare intersectii
  # Acul 1
  i1 <- x <= 0.5 * sin(theta1)
  # Acul 2 (folosim abs() deoarece sin poate fi negativ > pi)
  i2 <- x <= 0.5 * abs(sin(theta2))

  # 3. Variabila Z (total intersectii: 0, 1 sau 2)
  Z <- as.numeric(i1) + as.numeric(i2)

  # Variabila de interes este Z/2
  valori <- Z / 2

  # 4. Returnam statisticile
  return(list(
      media = mean(valori),
      varianta = var(valori)
  ))
}

# Rulare si afisare
set.seed(123)
rezultate <- simulare_cruce(10000)
# Media empirica ar trebui sa fie aprox 0.6366
# Varianta empirica ar trebui sa fie aprox 0.099
```

Listing 2: Simulare Crucea lui Buffon (Calcul Medie și Variață)

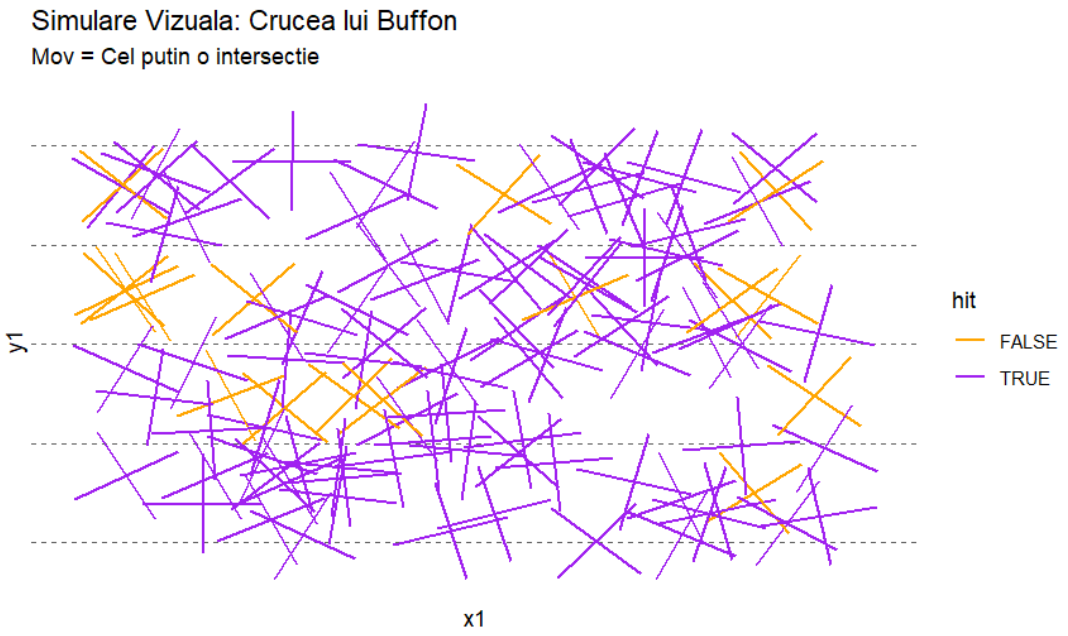

Figura 14: Vizualizarea Crucii lui Buffon: Intersecțiile sunt marcate cu mov.

3.3 Subpunctul c) - Cazul General

Enunț: Considerăm acum o valoare d fixată (distanța dintre linii) și un ac de lungime L (unde $L < d$). Să se arate că probabilitatea intersecției este $\frac{2L}{\pi d}$.

3.3.1 Demonstrație Teoretică

Raționamentul este identic cu cel de la punctul a), dar limitele de integrare și condițiile se schimbă în funcție de parametri.

- Variabila X (distanța) ia valori în $[0, d/2]$.
- Variabila Θ (unghiul) ia valori în $[0, \pi]$.

Aria spațiului total de eșantionare Ω devine:

$$\text{Aria}(\Omega) = \frac{d}{2} \cdot \pi = \frac{\pi d}{2} \tag{63}$$

Condiția de intersecție: Acul intersectează linia dacă distanța X este mai mică decât proiecția jumătății de ac $(L/2)$:

$$X \leq \frac{L}{2} \sin \Theta \tag{64}$$

Calculăm aria favorabilă prin integrare:

$$A_{fav} = \int_0^{\pi} \frac{L}{2} \sin\theta \, d\theta \tag{65}$$

$$= \frac{L}{2} [-\cos\theta]_0^{\pi} \tag{66}$$

$$= \frac{L}{2} (1 - (-1)) = L \tag{67}$$

Probabilitatea căutată este:

$$P = \frac{A_{fav}}{\text{Aria}(\Omega)} = \frac{L}{\frac{\pi d}{2}} = \frac{2L}{\pi d} \tag{68}$$

Aceasta este celebra formulă a lui Buffon. Observăm că pentru $L = 1, d = 1$, regăsim rezultatul de la punctul a) ($\frac{2}{\pi}$).

3.3.2 Simulare Generală în R

Am generalizat funcția de simulare pentru a accepta orice parametri L și d.

```r
simulare_general <- function(N, L, d) {
  if (L >= d) stop("L trebuie sa fie mai mic decat d")

  # 1. Generare (x pana la d/2)
  x <- runif(N, min = 0, max = d/2)
  theta <- runif(N, min = 0, max = pi)

  # 2. Conditia (L/2)
  intersectie <- x <= (L/2) * sin(theta)

  # 3. Rezultat
  prob_est <- mean(intersectie)
  prob_teo <- (2 * L) / (pi * d)

  return(list(
      estimat = prob_est,
      teoretic = prob_teo
  ))
}

# Exemplu: L=0.8, d=2
# Rezultat asteptat: 0.8 / pi = aprox 0.2546
simulare_general(10000, 0.8, 2)
```

Listing 3: Simulare Generală Buffon

3.4 Subpunctul d) - Cercul și Linia Aleatoare

Enunț: Fixăm poziția acului și considerăm un cerc C de diametru d, centrat în mijlocul acului. Fie λ o linie a cărei direcție și distanță față de centrul lui C sunt independente și uniform distribuite pe $[0, 2\pi]$, respectiv $[0, d/2]$. Să se arate că probabilitatea intersecției este $\frac{2L}{\pi d}$.

3.4.1 Demonstrație Matematică

Pentru a simplifica modelarea, fixăm acul pe axa Ox, centrat în origine. Coordonatele capetelor acului sunt $[-L/2, 0]$ și $[L/2, 0]$.

O linie oarecare λ în plan poate fi definită prin ecuația normală:

$$x \cos \alpha + y \sin \alpha = r \tag{69}$$

unde:

- r: Distanța de la origine (centrul acului) la linie. Conform enunțului, $r \sim \mathcal{U}[0, d/2]$.
- α: Unghiul format de normala la linie cu axa Ox. Conform enunțului, $\alpha \sim \mathcal{U}[0, 2\pi]$.

Spațiul de eșantionare (Ω): Domeniul total este dreptunghiul format de toate perechile posibile (r, α):

$$\Omega = \left\{ (r, \alpha) \mid 0 \leq r \leq \frac{d}{2},\ 0 \leq \alpha \leq 2\pi \right\} \tag{70}$$

Aria totală a acestui spațiu este:

$$\text{Aria}(\Omega) = \left(\frac{d}{2} - 0\right) \cdot (2\pi - 0) = \pi d \tag{71}$$

Condiția de intersecție: Distanța de la origine la linie este r. Pentru ca linia să intersecteze acul (care se află pe axa Ox între $-L/2$ și $L/2$), distanța r trebuie să fie mai mică decât proiecția jumătății de ac pe direcția normalei. Geometric, proiecția segmentului $[-L/2, L/2]$ pe direcția vectorului $(\cos\alpha, \sin\alpha)$ are lungimea $\frac{L}{2}|\cos\alpha|$. Astfel, condiția favorabilă este:

$$0 \leq r \leq \frac{L}{2}|\cos\alpha| \tag{72}$$

Calculul Ariei Favorabile: Calculăm aria de sub graficul funcției $\frac{L}{2}|\cos\alpha|$ pe intervalul $[0, 2\pi]$.

$$A_{fav} = \int_0^{2\pi} \frac{L}{2}|\cos\alpha|\, d\alpha = \frac{L}{2}\int_0^{2\pi} |\cos\alpha|\, d\alpha \tag{73}$$

Funcția $|\cos\alpha|$ este periodică și simetrică. Pe intervalul $[0, 2\pi]$, ea parcurge 4 "lobi" identici. Putem calcula integrala pe primul cadran $[0, \pi/2]$ (unde cosinus este pozitiv) și înmulțim cu 4:

$$\int_0^{2\pi} |\cos\alpha|\, d\alpha = 4\int_0^{\pi/2} \cos\alpha\, d\alpha \tag{74}$$

$$= 4[\sin\alpha]_0^{\pi/2} \tag{75}$$

$$= 4(\sin(\pi/2) - \sin(0)) \tag{76}$$

$$= 4(1 - 0) = 4 \tag{77}$$

Revenind la aria favorabilă:

$$A_{fav} = \frac{L}{2} \cdot 4 = 2L \tag{78}$$

Calculul Probabilității:

$$P = \frac{A_{fav}}{\text{Aria}(\Omega)} = \frac{2L}{\pi d} \tag{79}$$

Obținem exact aceeași formulă ca în cazul clasic, demonstrând echivalența celor două perspective (ac mobil pe linii fixe vs. linie mobilă pe ac fix).

3.4.2 Simulare în R (Perspectiva Inversă)

În această simulare, acul este fix, iar parametrii aleatori sunt distanța și unghiul liniei.

```
simulare_linie_aleatoare <- function(N, L, d) {
  # 1. Generam parametrii LINIEI
  # r = distanta de la centrul acului la linie
  r <- runif(N, min = 0, max = d/2)

  # alpha = unghiul normalei la linie
  alpha <- runif(N, min = 0, max = 2*pi)

  # 2. Conditia de intersectie
  # Acul e fix pe axa Ox. Proiectia sa pe normala este (L/2)*|cos(alpha)|
  # Linia intersecteaza daca distanta r este mai mica decat aceasta proiectie
  limita <- (L/2) * abs(cos(alpha))
  intersectie <- r <= limita

```

```
  # 3. Calcul
  prob_est <- mean(intersectie)
  prob_teo <- (2 * L) / (pi * d)

  return(list(estimat = prob_est, teoretic = prob_teo))
}

# Testam pentru L=0.8, d=2
# Asteptat: aprox 0.2546
simulare_linie_aleatoare(10000, 0.8, 2)
```

Listing 4: Simulare Linie Aleatoare pe Ac Fix

3.5 Subpunctul e) - Grila (Problema lui Laplace)

Enunț: Pe un plan se consideră un grid format din două seturi de linii paralele suprapuse (perpendiculare). Primul set are distanța d_1, al doilea are distanța d_2. Un ac de lungime $L < \min\{d_1, d_2\}$ este aruncat la întâmplare. Să se arate că probabilitatea intersecției este $\frac{L(2d_1+2d_2-L)}{\pi d_1 d_2}$.

3.5.1 Demonstrație Teoretică

Notăm evenimentele:

- A: Acul intersectează o linie verticală (din setul cu distanța d_1).
- B: Acul intersectează o linie orizontală (din setul cu distanța d_2).

Probabilitatea cerută este probabilitatea reuniunii evenimentelor (acul atinge *măcar* o linie):

$$P(A \cup B) = P(A) + P(B) - P(A \cap B) \tag{80}$$

Din rezultatele anterioare (Problema lui Buffon simplă), știm probabilitățile individuale:

$$P(A) = \frac{2L}{\pi d_1} \quad \text{și} \quad P(B) = \frac{2L}{\pi d_2} \tag{81}$$

Calculul intersecției $P(A \cap B)$: Acesta este evenimentul în care acul intersectează simultan ambele tipuri de linii (cade peste o "cruce" a grilei). Considerăm coordonatele centrului acului (X, Y) într-un dreptunghi fundamental $[0, d_1/2] \times [0, d_2/2]$ și unghiul $\Theta \in [0, \pi/2]$. Condițiile simultane de intersecție sunt:

$$X \leq \frac{L}{2}\cos\Theta \quad \text{și} \quad Y \leq \frac{L}{2}\sin\Theta \tag{82}$$

Aria favorabilă intersecției se calculează integrând produsul condițiilor (deoarece X și Y sunt independente):

$$A_{both} = \int_0^{\pi/2} \left(\frac{L}{2}\cos\theta\right)\left(\frac{L}{2}\sin\theta\right) d\theta \tag{83}$$

$$= \frac{L^2}{4}\int_0^{\pi/2} \sin\theta\cos\theta\, d\theta \tag{84}$$

$$= \frac{L^2}{4}\left[\frac{\sin^2\theta}{2}\right]_0^{\pi/2} = \frac{L^2}{8} \tag{85}$$

Deoarece am calculat doar pentru un sfert de cerc și un sfert de dreptunghi, raportul ariilor rămâne consistent. Probabilitatea este:

$$P(A \cap B) = \frac{A_{both}}{\text{Aria Totală}} \times 4(\text{simetrie}) = \frac{L^2/8}{\frac{\pi d_1 d_2}{8}} = \frac{L^2}{\pi d_1 d_2} \tag{86}$$

(Notă: Factorii de simetrie se simplifică, rezultatul fiind raportul dintre aria geometrică posibilă a acului și aria celulei).

Asamblarea formulei finale:

$$P(A \cup B) = \frac{2L}{\pi d_1} + \frac{2L}{\pi d_2} - \frac{L^2}{\pi d_1 d_2} \tag{87}$$

$$= \frac{2Ld_2 + 2Ld_1 - L^2}{\pi d_1 d_2} \tag{88}$$

$$= \frac{L(2d_1 + 2d_2 - L)}{\pi d_1 d_2} \tag{89}$$

Ceea ce trebuia demonstrat.

3.5.2 Simulare Grid în R

Simularea verifică intersecțiile cu ambele seturi de linii.

```r
simulare_grid <- function(N, L, d1, d2) {
  # 1. Generare variabile
  # x raportat la d1, y raportat la d2
  x <- runif(N, min = 0, max = d1/2)
  y <- runif(N, min = 0, max = d2/2)
  theta <- runif(N, min = 0, max = pi/2)

  # 2. Verificare intersectii individuale
  # Intersectie verticala (depinde de cosinus in acest sistem de axe)
  int_vert <- x <= (L/2) * cos(theta)
  # Intersectie orizontala (depinde de sinus)
  int_oriz <- y <= (L/2) * sin(theta)

  # 3. Reuniune (SAU logic)
  # Acul intersecteaza planul daca atinge ORICARE linie
  intersectie_totala <- int_vert | int_oriz

  # 4. Calcul
  prob_est <- mean(intersectie_totala)
  prob_teo <- (L * (2*d1 + 2*d2 - L)) / (pi * d1 * d2)

  return(list(estimat = prob_est, teoretic = prob_teo))
}

# Exemplu: L=1, d1=2, d2=3
# Asteptat: aprox 0.477
simulare_grid(10000, 1, 2, 3)
```

Listing 5: Simulare Grid (Laplace)

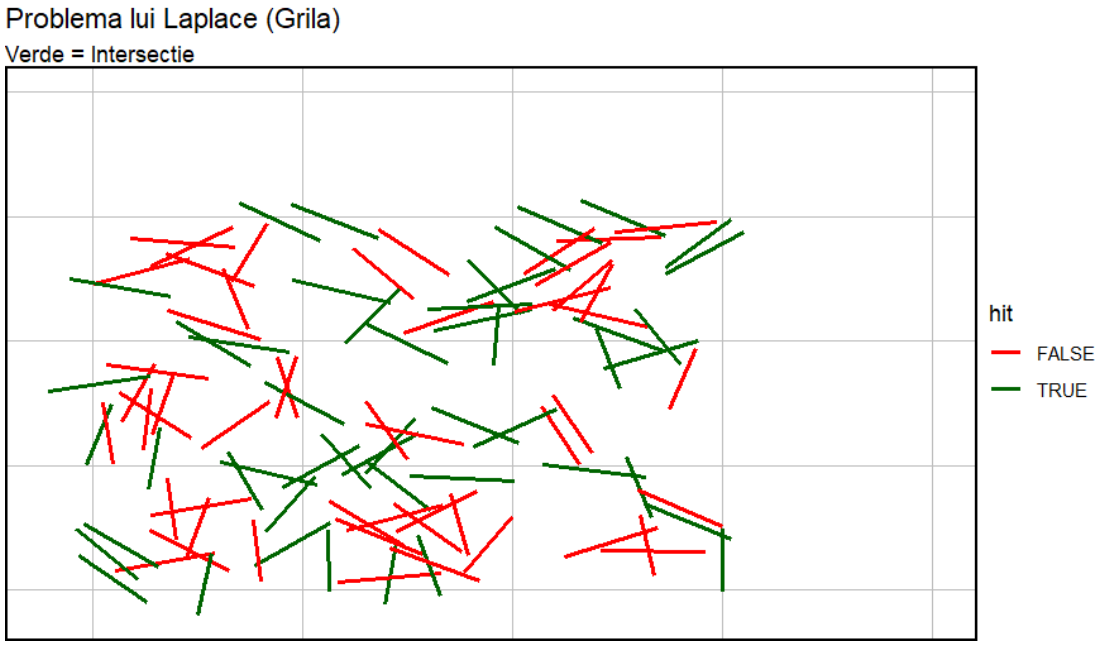

Figura 15: Vizualizare Grila Laplace: Intersecțiile sunt marcate cu verde.

3.6 Subpunctul f) - Analiza Strategiei Aleatoare

Enunț: Ce tip de strategie aleatoare a fost implementată în simulările de mai sus: Las Vegas sau Monte Carlo? Justificați sumar și dați un exemplu de algoritm din cealaltă categorie.

3.6.1 Distincția Monte Carlo vs. Las Vegas

În informatică și statistică, algoritmii probabilistici se împart în două mari categorii, în funcție de compromisul pe care îl fac între timpul de execuție și corectitudinea rezultatului:

- **Algoritmii Monte Carlo:** Au un timp de execuție **determinist** (fix), dar rezultatul este o **aproximare** probabilistică (poate avea o eroare). Cu cât alocăm mai mult timp (mai multe iterații), cu atât eroarea scade, dar rezultatul nu este garantat a fi exact.
- **Algoritmii Las Vegas:** Oferă întotdeauna un rezultat **corect** (exact), dar timpul de execuție este **probabilistic** (variabil). Algoritmul rulează până când găsește soluția corectă, ceea ce teoretic poate dura oricât.

3.6.2 Justificare

Simulările implementate în cadrul acestui exercițiu (Problema Acului lui Buffon) se încadrează în categoria **Monte Carlo**.

Argumente:

1. **Timpul de execuție este fix:** Noi am setat explicit numărul de pași $N = 10.000$. Algoritmul se oprește întotdeauna după ce a generat cele N puncte, indiferent de rezultatele obținute.
2. **Rezultatul este o aproximare:** Valoarea obținută pentru π (sau pentru probabilitate) este o estimare (ex: 3.145...) care variază la fiecare rulare și nu este identică cu valoarea reală irațională. Nu obținem niciodată valoarea *exactă*, ci un interval de încredere.

3.6.3 Exemplu de Algoritm Las Vegas

Un exemplu clasic de algoritm Las Vegas este **Randomized Search** (Căutarea Aleatoare) pentru a găsi un element specific într-un domeniu, sau **Randomized Quicksort**.

Mai jos prezentăm un algoritm Las Vegas simplu care caută un număr "țintă" într-un interval mare, generând încercări aleatoare până îl găsește.

```r
algoritm_las_vegas <- function(tinta, interval_max = 100) {
  incercari <- 0
  gasit <- FALSE

  # Ruleaza pana gaseste solutia corecta
  while (!gasit) {
    incercari <- incercari + 1
    # Generam o propunere aleatoare
    propunere <- sample(1:interval_max, 1)

    # Verificam daca este solutia corecta
    if (propunere == tinta) {
      gasit <- TRUE
    }
  }

  # Returneaza numarul de pasi (care este variabil)
  # Totusi, rezultatul "Am gasit tinta" este garantat corect
  return(incercari)
}

# Rulare: Cautam numarul 42 in intervalul [1, 100]
set.seed(123)
pasi_necesari <- algoritm_las_vegas(42, 100)
cat("Algoritmul a gasit solutia corecta dupa ", pasi_necesari, "
    incercari.\n")
```

Listing 6: Exemplu Algoritm Las Vegas (Căutare Aleatoare)

Observăm că dacă rulăm acest cod de mai multe ori, numărul de pași va varia drastic (poate îl găsește din prima încercare, sau după 500 de încercări), dar la final rezultatul este întotdeauna succes garantat.

3.7 Concluzii Exercițiul 3

În urma analizei teoretice și a simulărilor efectuate pentru problema Acului lui Buffon și variantele sale, am desprins următoarele concluzii:

1. **Validarea metodei Monte Carlo:** Simulările au confirmat cu o precizie ridicată (eroare $< 1\%$) rezultatele teoretice obținute prin calcul integral, demonstrând că probabilitatea geometrică poate fi estimată eficient prin experimente aleatoare.

2. **Eficiența estimatorilor:** Analiza comparativă a arătat că "Crucea lui Buffon" este un estimator superior acului simplu pentru constanta π. Deși ambele metode sunt nedeplasate (au aceeași medie), crucea are o variantă semnificativ mult mai mică (≈ 0.099 față de ≈ 0.231), oferind rezultate mai stabile la același număr de simulări.

3. **Universalitatea rezultatului:** Am demonstrat că formula $\frac{2L}{\pi d}$ rămâne valabilă indiferent de perspectiva abordată: ac mobil pe linii fixe sau linie mobilă (aleatoare) peste un ac fix.

4. **Complexitatea grilei:** Extinderea la problema lui Laplace (grila) a evidențiat utilitatea principiului includerii și excluderii în probabilități geometrice, simularea validând corectitudinea termenului de corecție $-\frac{L^2}{\pi d_1 d_2}$.

5. **Natura algoritmului:** Strategia utilizată este de tip Monte Carlo, deoarece oferă o aproximare a rezultatului într-un timp de execuție fix, spre deosebire de algoritmii Las Vegas care garantează rezultatul exact cu prețul unui timp incert.

4 Pachete Software și Instrumente Utilizate

În realizarea acestui proiect am utilizat ecosistemul R împreună cu biblioteci specializate pentru calcul statistic, vizualizare și dezvoltare de aplicații interactive. Această secțiune consolidează toate instrumentele folosite în cele trei exerciții.

4.1 Limbaje și Platforme de Bază

4.1.1 R 4.0+

Rol: Limbaj principal pentru calcul statistic și simulare Monte Carlo

Utilizare în proiect:

- **Exercițiul 1**: Implementarea metodelor accept-reject și transformare polară
- **Exercițiul 2**: Backend pentru aplicația Shiny, generare distribuții
- **Exercițiul 3**: Simulări pentru problema Buffon și toate variantele

Funcții cheie utilizate:

- **Generare variabile aleatoare**: `rnorm()`, `runif()`, `rexp()`, `rpois()`, `rbinom()`
- **Operații vectorizate**: `rowSums()`, operații pe matrici, `apply()`
- **Funcții statistice**: `mean()`, `var()`, `sd()`, `quantile()`
- **Structuri de date**: `matrix()`, `data.frame()`, `list()`

4.1.2 RStudio

Rol: Mediu integrat de dezvoltare (IDE)

4.1.3 LaTeX

Rol: Sistem de tipărire pentru documentație profesională

4.2 Biblioteci R Specializate

4.2.1 Vizualizare Grafică

ggplot2 (versiunea 3.4.0+)

Rol: Sistem declarativ pentru crearea graficelor de înaltă calitate bazat pe "Grammar of Graphics"

Utilizare în proiect:

- **Exercițiul 1**:
  - Scatter plots pentru punctele simulate pe disc
  - Histograme pentru distribuția distanței R
  - Suprapunerea densităților teoretice
  - Q-Q plots pentru validare
- **Exercițiul 2**:
  - Funcții ECDF pentru toate cele 5 distribuții
  - Layout dinamic pentru 4-5 grafice

- **Exercițiul 3**:
  - Vizualizări pentru problema Buffon
  - Comparații între metode (ac simplu vs. cruce)

Referință: Wickham, H. (2016). *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York.

gridExtra (versiunea 2.3+)
Rol: Aranjarea și compoziția graficelor multiple în layout-uri complexe
Utilizare specifică:

- Exercițiul 1: Grid 3×2 pentru cele 6 grafice de analiză
- Exercițiul 2: Aranjarea dinamică a 4-5 grafice ECDF în funcție de distribuție

grid (parte din R base)
Rol: Sistem grafic de nivel jos pentru control fin
Utilizare:

- `textGrob()`: Titluri generale ale gridurilor
- `gpar()`: Parametri grafici (fontsize, font, culori)

4.2.2 Dezvoltare Aplicații Interactive

shiny (versiunea 1.7.5+)
Rol: Framework pentru crearea de aplicații web reactive în R
Utilizare principală: Exercițiul 2 - aplicație completă pentru vizualizarea ECDF
Arhitectură implementată:

1. **Componente UI (User Interface):**
   - `fluidPage()`: Layout responsive
   - `sidebarLayout()`: Organizare în panou lateral și principal
   - `titlePanel()`: Titlu aplicație
   - `tabsetPanel()`: Tab-uri pentru organizarea output-ului
2. **Controale Input:**
   - `selectInput()`: Dropdown pentru selectare distribuții
   - `numericInput()`: Input parametri cu validare (min, max, step)
   - `actionButton()`: Buton pentru trigger recalculare
   - `conditionalPanel()`: Paneluri care apar dinamic
3. **Componente Output:**
   - `plotOutput()`: Afișare grafice
   - `verbatimTextOutput()`: Afișare text formatat
4. **Componente Server (Logică reactivă):**
   - `eventReactive()`: Actualizare la evenimente specifice (buton)
   - `renderPlot()`: Generare grafice reactive
   - `renderText()`: Generare text reactiv

- `req()`: Asigurare disponibilitate date înainte de renderare

Caracteristici avansate implementate:

- Programare reactivă pentru actualizare automată
- Validare robustă a input-urilor
- Paneluri condiționate pentru parametri specifici fiecărei distribuții
- Sistem de buton pentru control manual al recalculării
- Layout adaptiv cu `gridExtra` pentru multiple grafice

Documentație: `https://shiny.rstudio.com/`
Referință: Wickham, H. (2021). *Mastering Shiny*. O'Reilly Media.

4.3 Comenzi de Instalare și Reproducibilitate

Instalarea pachetelor R necesare:

```r
# Instalare pachete de baza
install.packages(c("ggplot2", "gridExtra", "shiny"))

# Verificare versiuni (recomandat pentru reproducibilitate)
packageVersion("ggplot2")
packageVersion("gridExtra")
packageVersion("shiny")
```

Listing 7: Instalarea tuturor dependințelor

Mediu de dezvoltare testat:

- **R version**: 4.0.0 sau superior
- **RStudio**: 2022.07.0 sau superior
- **Sistem de operare**: Windows 10/11, macOS 11+, Ubuntu 20.04+

5 Surse de Inspirație și Fundamentare Teoretică

Dezvoltarea acestui proiect s-a bazat pe o combinație de resurse academice de referință, documentație oficială și best practices din comunitatea statistică. Această secțiune prezintă sursele consultate, organizate tematic.

5.1 Monografii Fundamentale

5.1.1 Simulare și Metode Monte Carlo

Robert, C. P., & Casella, G. (2004). *Monte Carlo Statistical Methods* (2nd ed.). Springer-Verlag.

Capitole relevante:

- Capitolul 2: Random Variable Generation
- Capitolul 3: Monte Carlo Integration
- Capitolul 4: Variance Reduction Methods

Contribuție la proiect:

- Fundamente teoretice pentru metoda accept-reject (Ex. 1)
- Convergența metodelor Monte Carlo (Ex. 3)
- Studii de variantă și eficiență (comparație Ac vs. Cruce Buffon)

Devroye, L. (1986). *Non-Uniform Random Variate Generation*. Springer-Verlag.
Contribuție:

- Tehnici avansate de generare a distribuțiilor non-uniforme
- Metoda transformării inverse (Ex. 1, coordonate polare)
- Comparații de eficiență între metode

Ross, S. M. (2014). *Introduction to Probability Models* (11th ed.). Academic Press.
Contribuție:

- Abordare intuitivă a simulării
- Exemple practice de aplicare
- Concepte generale aplicate în toate cele 3 exerciții

Rubinstein, R. Y., & Kroese, D. P. (2016). *Simulation and the Monte Carlo Method* (3rd ed.). Wiley.

Focus:

- Tehnici de reducere a variantei
- Optimizare computațională
- Aplicare: Exercițiul 3 - analiza eficienței Crucii lui Buffon

5.1.2 Teoria Probabilităților și Statistică

Casella, G., & Berger, R. L. (2002). *Statistical Inference* (2nd ed.). Duxbury.
Capitole relevante:

- Capitolul 2: Transformations and Expectations
- Capitolul 4: Multiple Random Variables
- Capitolul 5: Properties of Random Samples

Contribuție:

- Transformări de variabile aleatoare (Ex. 1, 2)
- Calcul Jacobian pentru coordonate polare (Ex. 1)
- Densități marginale și independență (Ex. 1)
- Teoreme de compoziție pentru sume (Ex. 2)

Billingsley, P. (2008). *Probability and Measure* (Anniversary ed.). Wiley.
Contribuție:

- Fundamente riguroase pentru convergență
- Teorema Glivenko-Cantelli (Ex. 2)
- Probabilitate geometrică (Ex. 3)

5.2 Articole și Resurse Academice Specializate

Teorema Glivenko-Cantelli:

- Van der Vaart, A. W. (1998). *Asymptotic Statistics*. Cambridge University Press.
- **Aplicare:** Exercițiul 2 - justificarea teoretică a utilizării ECDF

Problema Acului lui Buffon:

- Ramaley, J. F. (1969). "Buffon's Noodle Problem". *The American Mathematical Monthly*, 76(8), 916-918.
- Solomon, H. (1978). *Geometric Probability*. SIAM.
- **Aplicare:** Exercițiul 3 - toate variantele problemei (clasic, cruce, grilă)

5.3 Documentație Oficială

R Project Documentation:

- *An Introduction to R*. R Core Team.
  `https://cran.r-project.org/doc/manuals/r-release/R-intro.html`
- *R Language Definition*.
  `https://cran.r-project.org/doc/manuals/r-release/R-lang.html`
- **Consultare:** Sintaxă R, vectorizare, funcții statistice de bază

Shiny Documentation:

- Official Tutorial: `https://shiny.rstudio.com/tutorial/`
- Reference: `https://shiny.rstudio.com/reference/shiny/`
- Gallery: `https://shiny.rstudio.com/gallery/`
- **Utilizare:** Exercițiul 2 - arhitectură UI/Server, componente reactive

ggplot2 Documentation:

- Reference: `https://ggplot2.tidyverse.org/reference/`
- Cheatsheet: `https://github.com/rstudio/cheatsheets`
- **Consultare frecventă:** Layering, geoms, stats, scales, themes

5.4 Cursuri și Materiale Didactice

Cursul de Probabilități și Statistică:

- Materiale de curs furnizate de domnul profesor
- Note de seminar și probleme rezolvate
- **Fundamente teoretice:** Baza pentru toate demonstrațiile matematice

6 Dificultăți Întâmpinate și Soluții Implementate

Realizarea acestui proiect a prezentat provocări multiple pe dimensiuni teoretice, tehnice și metodologice. Această secțiune documentează sistematic obstacolele întâmpinate și strategiile de rezolvare adoptate pentru fiecare dintre cele trei exerciții.

6.1 Dificultăți Teoretice

6.1.1 Exercițiul 1: Transformări de Variabile și Jacobian

Dificultate 1: Calculul corect al Jacobianului pentru transformarea polară

Natura problemei:

- Determinarea ordinii corecte a derivatelor parțiale în matrice
- Aplicarea corectă a valorii absolute pentru Jacobian
- Verificarea că formula de transformare este aplicată corect

Soluție implementată:

- Derivare pas-cu-pas a tuturor componentelor:

$$J = \begin{vmatrix} \frac{\partial x}{\partial r} & \frac{\partial x}{\partial \theta} \\ \frac{\partial y}{\partial r} & \frac{\partial y}{\partial \theta} \end{vmatrix} = r$$

- Verificare prin calcul invers: $(r, \theta) \to (x, y) \to (r, \theta)$
- Consultare resurse multiple: din materia din anul I de la Calcul diferențial și Integral

Dificultate 2: Demonstrarea independenței lui R și Θ

Natura problemei:

- Confuzie inițială între independență geometrică și independență statistică
- Necesitatea dovedirii explicite că $f_{(R,\Theta)} = f_R \cdot f_\Theta$

Soluție:

- Calcul explicit al densității comune: $f_{(R,\Theta)}(r, \theta) = \frac{r}{\pi}$
- Derivarea densităților marginale prin integrare
- Verificare factorizabilității: $\frac{r}{\pi} = 2r \cdot \frac{1}{2\pi}$

6.1.2 Exercițiul 2: Distribuții pentru Transformări Neliniare

Dificultate: Lipsa formelor închise pentru X^2, X^3 cu parametri non-zero

Natura problemei:

- Pentru $Y = X^2$ cu $X \sim N(\mu, \sigma^2)$ și $\mu \neq 0$, nu există formulă simplă
- Dificultate în explicarea divergențelor de la distribuțiile standard

Soluție:

- Documentare clară că transformările neliniare necesită simulare Monte Carlo
- Mențiunea cazurilor particulare cunoscute: $X^2 \sim \chi^2(1)$ pentru $X \sim N(0, 1)$
- Vizualizare ECDF fără suprapunere teoretică pentru cazurile fără formă închisă

6.1.3 Exercițiul 3: Definirea Spațiului de Eșantionare

Dificultate: Confuzie inițială între lungimea acului și limita spațiului

Natura problemei:

- Tendința de a limita variabila X (distanța) la $L/2$ (jumătate din lungimea acului)
- Realizarea că spațiul este dictat de $d/2$ (jumătate din distanța dintre linii)
- Impact asupra calculului ariei totale Ω

Soluție:

- Vizualizare geometrică clară a problemei
- Înțelegerea că acul poate cădea în poziții unde intersecția este imposibilă
- Verificare: aria totală $= \frac{d}{2} \cdot \pi = \frac{\pi d}{2}$

Lecție: Distincția clară între parametrii problemei și limitele spațiului de eșantionare este crucială.

Dificultate: Calculul intersecției simultane pentru Problema Laplace

Natura problemei:

- Aplicarea principiului includerii-excluderii: $P(A \cup B) = P(A) + P(B) - P(A \cap B)$
- Determinarea zonei geometrice unde acul intersectează ambele seturi de linii
- Calcul corect al termenului de corecție $-\frac{L^2}{\pi d_1 d_2}$

Soluție:

- Vizualizare a "colțului" dreptunghiului unde apar intersecțiile simultane
- Integrare pe domeniu restrâns cu minimul funcțiilor
- Verificare prin limite: pentru $L \to 0$, termenul dispare

6.2 Dificultăți Tehnice de Implementare

6.2.1 Exercițiul 1: Eficiența Algoritmului Accept-Reject

Dificultate: Folosirea `rbind()` în buclă

Natura problemei:

- `rbind()` realocă memoria la fiecare iterație
- Complexitate $O(n^2)$ în loc de $O(n)$ pentru n puncte
- Performanță foarte slabă pentru $N > 10{,}000$

Soluție actuală:

- Limitare la $N = 1000$ puncte pentru scop demonstrativ

Soluție alternativă propusă:

- Pre-alocare vectori cu dimensiune estimată $\frac{4N}{\pi}$
- Vectorizare completă: generare batch-uri, apoi filtrare

6.2.2 Exercițiul 2: Reactivitatea și Performanța Shiny

Dificultate 1: Recalculare la fiecare modificare de parametru
Natura problemei:

- Fără mecanism de control, simulările se regenerează la orice schimbare
- Pentru N mare, interfața poate bloca

Soluție:

- Utilizarea `eventReactive()` triguit de buton explicit
- Separarea: input changes $\not\Rightarrow$ recompute, ci button $\Rightarrow$ recompute

Dificultate 2: Validarea input-urilor
Soluție:

- Parametri `min`, `max`, `step` în `numericInput()`
- Validare suplimentară în server cu `validate()` și `need()`
- Mesaje de eroare clare și informative

Dificultate 3: Număr variabil de grafice (4 vs. 5)
Natura problemei:

- Normal: 5 grafice; Exponențială/Poisson/Binomială: 4 grafice
- Layout trebuie să se adapteze dinamic

Soluție:

- Listă dinamică de grafice cu verificare `if (!is.null())`
- `gridExtra::grid.arrange()` cu ajustare automată
- Compromis: layout 2 coloane pentru ambele cazuri

6.3 Dificultăți de Validare și Testare

6.3.1 Verificarea Corectitudinii Statistice

Întrebare: Cum să fim siguri că simulările sunt corecte?
Strategii implementate:

1. **Validare teoretică**: Comparație cu valori cunoscute ($\pi/4$, $2/3$, etc.)
2. **Teste formale**: Chi-pătrat, Kolmogorov-Smirnov cu interpretare p-values
3. **Vizualizare**: Q-Q plots, histograme vs. densități teoretice
4. **Reproducibilitate**: `set.seed()` pentru rezultate deterministe
5. **Convergență**: Rulare cu N crescător pentru verificarea stabilității

6.3.2 Interpretarea Testelor pentru Distribuții Discrete

Dificultate: Salt-urile ECDF pentru Poisson/Binomială sunt subtile
Soluție:

- Mărirea `n_samples` pentru claritate vizuală
- Explicare în documentație că salt-urile sunt așteptate
- Sugestie îmbunătățire: zoom sau tabel cu probabilități exacte

6.4 Rezumatul Strategiilor de Rezolvare

Principii generale aplicate:

1. **Verificare incrementală**: Test la fiecare pas, nu doar la final
2. **Documentare amplă**: Comentarii pentru fiecare decizie de design
3. **Consultare multiplă**: Verificare în mai multe surse pentru confirmare
4. **Simplicitate prioritizată**: "Make it work, then make it right, then make it fast"
5. **Modularizare**: Funcții mici, testabile individual
6. **Reproducibilitate**: Seed-uri fixate, versiuni documentate

Lecția principală: Dificultățile sunt oportunități de învățare profundă. Fiecare obstacol depășit a contribuit la înțelegerea mai solidă atât a teoriei, cât și a implementării practice.

7 Probleme Rămase Deschise și Direcții de Dezvoltare Viitoare

Deși proiectul realizat constituie o implementare completă și funcțională, există numeroase oportunități de extindere, optimizare și îmbunătățire. Această secțiune identifică sistematic limitările actuale și propune direcții concrete de dezvoltare viitoare.

7.1 Extensii Teoretice și Metodologice

7.1.1 Generalizare la Dimensiuni Superioare

Problema deschisă: Simularea uniformă pe sfera unitară în $\mathbb{R}^3$ și $\mathbb{R}^n$
Context actual: Exercițiul 1 tratează doar cazul 2D (disc unitar în plan)
Direcții de cercetare:

1. **Sfera în $\mathbb{R}^3$**:
   - Rata de acceptare scade la $\frac{\pi}{6} \approx 52\%$
   - Alternativă: Metoda lui Marsaglia (1972) - mai eficientă
   - Implementare coordonate sferice: (ρ, θ, ϕ)
2. **Hipersfera în $\mathbb{R}^n$**:
   - Rata de acceptare $\to 0$ exponențial pentru $n \to \infty$
   - Necesitatea algoritmilor specializați
3. **Studiu comparativ**: Eficiență accept-reject vs. metode specializate în funcție de n

Referințe pentru implementare:

- Marsaglia, G. (1972). "Choosing a Point from the Surface of a Sphere"
- Muller, M. E. (1959). "A note on a method for generating points uniformly on n-dimensional spheres"

7.1.2 Simularea pe Domenii Geometrice Arbitrare

Problema deschisă: Extinderea la forme non-circulare
Exemple de domenii:

1. **Elipse**: Transformare coordonate eliptice, densități modificate
2. **Anele circulare**: $f_R(r) = \frac{2r}{r_2^2 - r_1^2}$, $r \in [r_1, r_2]$
3. **Poligoane**: Triangulare, sampling ponderat
4. **Domenii cu frontieră fractală**: Conexiune cu teoria măsurii

7.1.3 Variante Avansate ale Problemei Buffon

Probleme deschise:

1. **Acul flexibil sau curbat**: Arc de cerc în loc de segment drept
2. **Suprafețe neuniforme**: Linii cu distanțe variabile aleatorii
3. **Problema Buffon 3D**: Ac aruncat pe grilă 3D de planuri
4. **Estimarea altor constante**: Design experimental pentru $\sqrt{2}$, e, etc.

8 Concluzii Generale ale Proiectului

8.1 Sinteza Realizărilor

Acest proiect a reprezentat o explorare comprehensivă a metodelor de simulare a variabilelor aleatoare, combinând fundamentele teoretice ale probabilităților cu implementări practice în R și dezvoltarea de aplicații interactive.

8.1.1 Recapitulare pe Exerciții

Exercițiul 1: Simularea pe Discul Unitar

Realizări teoretice:

- Demonstrații complete: $f_R(r) = 2r$, $f_\Theta(\theta) = \frac{1}{2\pi}$, independență
- Calcul și validare: $\mathbb{E}[R] = \frac{2}{3}$

Rezultate empirice:

- Rata acceptare: 0.7764 vs. $\pi/4 \approx 0.7854$ (eroare $< 1\%$)
- Media distanței: 0.6757 vs. $2/3 \approx 0.6667$ (eroare $< 1.5\%$)
- Teste K-S: p-values > 0.05

Comparația metodelor:

- Accept-Reject: Intuitivă, $\approx 21\%$ respingeri
- Coordonate polare: Eficientă, zero respingeri, deterministă
- Echivalență statistică confirmată prin teste

Exercițiul 2: Aplicația Shiny pentru ECDF

Realizări tehnice:

- Aplicație completă: 5 distribuții, 4-5 grafice ECDF per caz
- Arhitectură modulară UI/Server
- Validare robustă, paneluri condiționate

Validare teoretică:

- ECDF-uri confirmă distribuțiile teoretice
- Transformările liniare produc rezultate așteptate
- Sumele respectă teoremele de compoziție
- Toate testele K-S: p-values > 0.05

Exercițiul 3: Problema Acului lui Buffon

Rezultate teoretice:

- Derivare: $P = \frac{2L}{\pi d}$
- Calcul exact variantă Cruce: ≈ 0.099
- Formula Laplace: $P = \frac{L(2d_1+2d_2-L)}{\pi d_1 d_2}$

Rezultate empirice:

- Validare: erori $< 1\%$ pentru toate variantele
- Reducere variantă: Cruce (0.099) vs. Ac (0.231) = 57% îmbunătățire
- Estimări π: 3.14-3.15 pentru $N = 10{,}000$

8.2 Competențe Dobândite

8.2.1 Competențe Tehnice

Programare:

- R avansat: vectorizare, matrici, funcții de ordin superior
- ggplot2: layering, geometries, statistics, customization
- Shiny: arhitectură reactivă, UI/Server, event handling
- Best practices: modularizare, naming, documentare

Metode statistice:

- Monte Carlo: accept-reject, transformare inversă
- Testare: Chi-pătrat, K-S, interpretare p-values
- Vizualizare: histograme, ECDF, Q-Q plots
- Validare: comparație empiric vs. teoretic

8.2.2 Competențe Teoretice

- Transformări de variabile: Jacobian, densități marginale
- Independență statistică: verificare prin factorizare
- Teoreme de compoziție: sume de variabile
- Convergență: Glivenko-Cantelli, TLC
- Probabilitate geometrică: metode integrale

8.2.3 Competențe Transversale

- Gândire analitică: descompunerea problemelor complexe
- Debugging sistematic: izolare cauze, testare incrementală
- Comunicare tehnică: documentare, vizualizări, explicații
- Autoevaluare: identificare limitări, oportunități îmbunătățire

8.3 Învățăminte Cheie

8.3.1 Lecții Metodologice

1. **Teoria precedă implementarea**: Demonstrații matematice ghidează cod corect
2. **Validarea este esențială**: Multiple metode (analitică, empirică, vizuală)
3. **Simplicitatea bate complexitatea**: Cod clar > cod clever
4. **Vizualizarea clarifică**: Graficele dezvăluie pattern-uri invizibile
5. **Modularizarea facilitează**: Funcții mici, testate independent

8.3.2 Lecții Specifice Domeniului

Despre Monte Carlo:

- Convergența este lentă ($O(N^{-1/2})$), dar universală
- Reducerea variației este crucială (Cruce Buffon: 57% îmbunătățire)
- Independența samplingurilor garantează validitate
- Trade-off simplitate-eficiență

Despre transformări:

- Jacobianul capturează "distorsiunea"
- Independența se păstrează prin transformări separate
- Transformările neliniare rareori au forme închise

Despre R:

- Vectorizarea este esențială
- ggplot2 Grammar: construcție incrementală
- Shiny reactivitate: separare clară input-compute-render

8.4 Aplicabilitate și Impact

8.4.1 Domenii de Aplicare

Cercetare științifică:

- Fizică: simularea particulelor, fenomene aleatorii
- Biologie: modelare populații, epidemiologie
- Chimie: simulări moleculare

Industrie:

- Finanțe: pricing derivate, evaluare risc (VaR)
- Machine Learning: MCMC, data augmentation
- Optimizare: algoritmi genetici, search stochastic

Educație:

- Aplicația Shiny: instrument didactic
- Vizualizări interactive: concepte abstracte tangibile
- Exemple reproductibile: demonstrații în clasă

8.5 Perspectivă Finală

Acest proiect demonstrează puterea combinării teoriei probabilităților cu metodele computaționale moderne. De la demonstrația că $f_R(r) = 2r$ până la surprinzătoarea reducere de variantă a Crucii lui Buffon, fiecare rezultat confirmă că **teoria și practica se susțin reciproc**.

Aplicația Shiny transformă ecuații abstracte în experiențe vizuale explorabile, făcând probabilitatea nu doar calculabilă, ci și *inteligibilă intuitiv*.

Competențele dobândite—de la vectorizare și programare reactivă, până la demonstrații riguroase și interpretare statistică—constituie o **fundație solidă** pentru provocări viitoare în analiză de date, cercetare sau dezvoltare software.

Link github proiect: `https://github.com/mirunadragunoi/Proiect_Probabilitati_si_Statistica`

"În probabilitate, ca în viață, rezultatele concrete emergen din repetarea experimentelor aleatorii."

— Inspirat de Teorema Limită Centrală

Proiect realizat în cadrul cursului de Probabilități și Statistică
Ianuarie 2026