

Aplicație Shiny pentru distribuții de variabile aleatoare și funcții personalizate

„Probabilitatea este arta de a lua decizii în condiții de incertitudine; statistica ne arată cum să ne bazăm acele decizii pe observații și date. Iar Shiny ne ajută să le vedem pe toate în timp real.”

1. Echipa de exercițiu

- Andruță Andra Mihaela
 - Tunaru Ioana Alexandra
-

2. Introducere și Context

Într-un peisaj al statisticii și probabilităților mereu în expansiune, aplicațiile interactive reprezintă o cale excelentă de a învăța, de a vizualiza și de a înțelege mai profund comportamentul variabilelor aleatoare. Proiectul de față a fost creat pentru a demonstra cum se pot genera eșantioane din distribuții bine-cunoscute (Normală, Exponențială, Poisson, Binomială), cum se pot realiza diverse transformări asupra acestora (de pildă $3 - 2X$, ΣX , ΣX^2 etc.) și cum pot fi reprezentate, în timp real, funcțiile de repartiție empirică (ECDF) corespunzătoare.

De asemenea, aplicația oferă posibilitatea de a explora un set de funcții personalizate, definite analitic, și de a le reprezenta grafic într-o manieră interactivă, cu ajustări de parametri și intervale de definire. Astfel, utilizatorii pot experimenta atât concepte de bază (precum generarea unui eșantion random), cât și elemente ceva mai avansate (ex. funcția densitate de tip Cauchy, exponențială modificată etc.).

Rezultatul este o aplicație Shiny (un pachet R special conceput pentru interfețe web interactive) care integrează elegant teoria distribuțiilor de probabilitate cu o componentă practică și vizuală, esențială în înțelegerea dinamică a datelor și a funcțiilor.

3. Cerințe și Obiective Principale

Proiectul răspunde următoarelor cerințe:

1. Generarea și reprezentarea grafică a funcțiilor de repartiție (ECDF) pentru:
 - Variabila X provenită din diverse distribuții: Normală ($N(0,1)$, respectiv $N(\mu, \sigma^2)$), Exponențială ($Exp(\lambda)$), Poisson ($Pois(\lambda)$) și Binomială ($Binom(r, p)$).
 - Transformări ale lui X de tipul $3 - 2X$, $2 + 5X$, $3X - 2$, $5X - 4$, X^2 , X^3 , ΣX și ΣX^2 .
2. Implementarea unor funcții personalizate (noteate A, B, C, D, E, F, G), pe care să le putem afișa grafic în R și pentru care să putem calcula (cel puțin teoretic) valorile de medie și varianță când sunt interpretate drept densități de probabilitate (acolo unde acest lucru se aplică).
3. Elemente de interactivitate:
 - Selectarea distribuției și a parametrilor (ex. μ , σ pentru Normală, λ pentru Exponențială etc.).
 - Posibilitatea de a alege transformarea dorită (ex. $3 - 2X$, X^3 , ΣX , etc.).
 - Afisarea funcțiilor personalizate într-o varietate de moduri: densități pe intervale continue, valori discrete (tip bară).

Scopul final a fost să creăm un instrument educațional și interactiv care să le permită utilizatorilor să vadă, să simuleze și să înțeleagă pe viu concepte fundamentale de statistică și probabilități, totul într-un singur loc.

4. Aspecte Teoretice și Metodologice

4.1 Generarea de mostre din distribuții

Pentru a obține date dintr-o distribuție cunoscută, am apelat la funcțiile de bază ale limbajului R:

- `rnorm(n, mean, sd)` pentru Normală
- `rexp(n, rate)` pentru Exponențială
- `rpois(n, lambda)` pentru Poisson
- `rbinom(n, size, prob)` pentru Binomială

Parametrizarea directă a acestor funcții simplifică foarte mult procesul de simulare. Un aspect interesant este reprezentat de transformarea variabilei inițiale (obținută din eșantion) prin diverse funcții (ex. X^2 sau $3 - 2X$). Această etapă demonstrează faptul că putem obține noi variabile aleatoare prin simpla aplicare a unor transformări matematice asupra valorilor generate.

4.2 Funcția de repartiție empirică (ECDF)

După ce generăm setul de date, putem obține repartiția empirică prin `stat_ecdf` (o resursă din `ggplot2`). Aceasta oferă o reprezentare de tip scară a frecvențelor cumulative, un mod intuitiv de a observa

distribuția (sau transformarea) obținută. Astfel, utilizatorii pot vizualiza variațiile și se pot familiariza cu conceptul de ECDF – un concept-cheie în statistica inferențială.

4.3 Construirea interfeței cu Shiny

- UI (User Interface): Afisează elementele de intrare (selectori, controale numerice, slider-e) și zona de afișare a graficelor și a ieșirilor textuale.
- Server: Reacționează la modificările de input, regenerază datele (dacă este cazul) și trimite către UI informațiile de afișat (ploturi, text etc.).

Prin această arhitectură reactivă, schimbarea unui parametru (de exemplu, a valorii σ în Normală) determină imediat regenerarea eșantionului și replotarea ECDF-ului.

4.4 Funcții personalizate

Acestea reprezintă un exercițiu suplimentar – de la simple polinoame ($c x^4$, $ax + bx^2$) până la funcții de densitate cunoscute (ex. Cauchy $\frac{1}{[\pi(1+x^2)]}$).

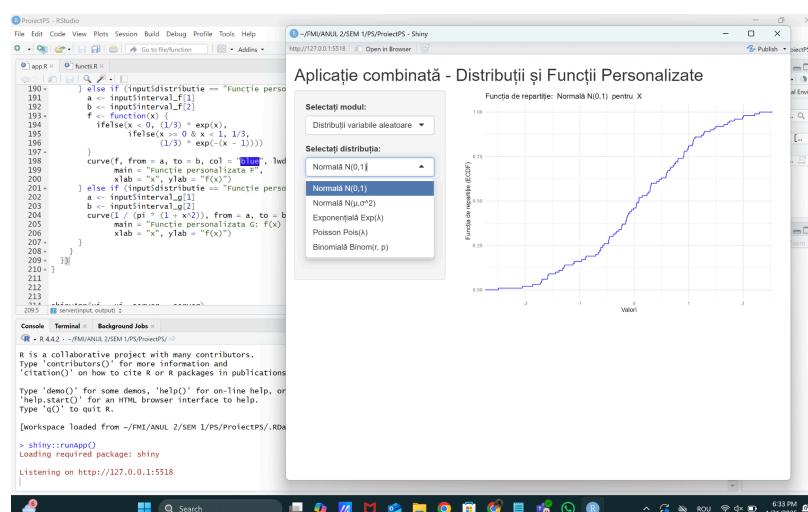
Scopul lor este de a oferi un cadru general prin care utilizatorul să poată vedea, într-un singur tab, mai multe feluri de funcții și să observe forma grafică, eventual discuții teoretice despre convergență, normalizare (integrarea pe suport), medie și varianță (dacă există).

5. Reprezentări Grafice și Elemente Interactive

5.1 Modul „Distribuții variabile aleatoare”

În acest modul, utilizatorul:

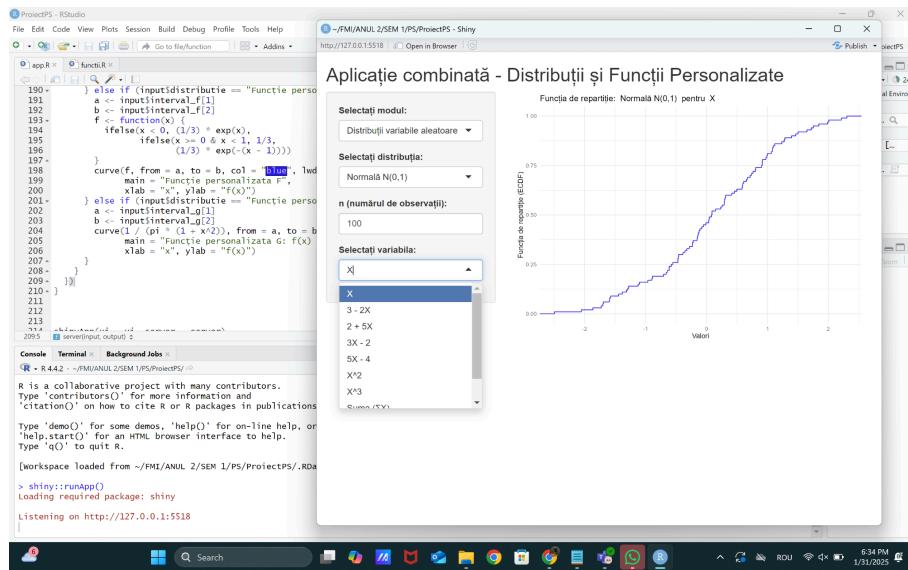
1. Selectează tipul distribuției: Normală (0,1), Normală (μ, σ), Exponențială (λ), Poisson (λ), Binomială (r, p).



2. De asemenea, poate alege numărul de observații n care să fie generate.

Odată aleasă distribuția și parametrul/parametrii specifici (acolo unde este cazul), aplicația generează un eșantion random de mărime n.

3. Alege transformarea dorită a variabilei X (prin selectInput cu opțiunile „X”, „3 - 2X”, „X²”, „Suma(Σ X)” și aşa mai departe).



La fiecare modificare, se generează un eșantion nou, se aplică transformarea și se afișează ECDF.

Concret:

- **Generarea datelor (exemplu):** `data <- rnorm(n, mean = 0, sd = 1)`

În acest fragment folosim `rnorm()` pentru a genera n valori dintr-o distribuție Normală cu media 0 și deviația standard 1. Similar, dacă distribuția aleasă de utilizator ar fi Exponențială, am fi folosit `rexp(n, rate = lambda)`, iar pentru Binomială, `rbinom(n, size = r, prob = p)` etc.

Motivația pentru folosirea acestor funcții predefinite din R (`rnorm`, `rexp`, `rpois`, `rbinom`) este tocmai ușurința cu care ne putem baza pe algoritmi verificați și eficienți de generare de valori pseudo-aleatoare din distribuții cunoscute.

- **Transformarea (exemplu):** `variable_data <- 3 - 2 * data`

Aici, pentru fiecare valoare generată în `data`, aplicăm transformarea matematică $3 - 2X$ (unde X este elementul din eșantion). Putem crea, astfel, o varietate de variabile noi, fie prin relații liniare ($2 + 5X$, $3X - 2$), fie prin puteri (X^2 , X^3) sau chiar prin sume cumulate (`cumsum(data)`).

Motivul pentru care am inclus aceste transformări este de a exemplifica felul în care o variabilă inițială (distribuită după un anumit model) poate fi manipulată pentru a studia comportamente noi, eventual pentru a observa cum se modifică anumite caracteristici statistice (media, varianța etc.).

- **Plotarea:**

```
ggplot(data.frame(x = variable_data), aes(x)) + stat_ecdf(geom =
"step", color = "blue") + ...
```

În această secvență, creăm mai întâi un data frame simplu, în care plasăm valorile transformate în coloana `x`. Apoi, prin `ggplot()`, folosim coordonatele `aes(x)` pentru a specifica variabila de interes.

Funcția `stat_ecdf()` din `ggplot2` este cea care calculează și desenează **ECDF** (Empirical Cumulative Distribution Function), folosind un stil de „trepte” (prin `geom = "step"`) pentru a arăta cum crește probabilitatea cumulată pe măsură ce ne deplasăm de la valorile mici ale variabilei la cele mari.

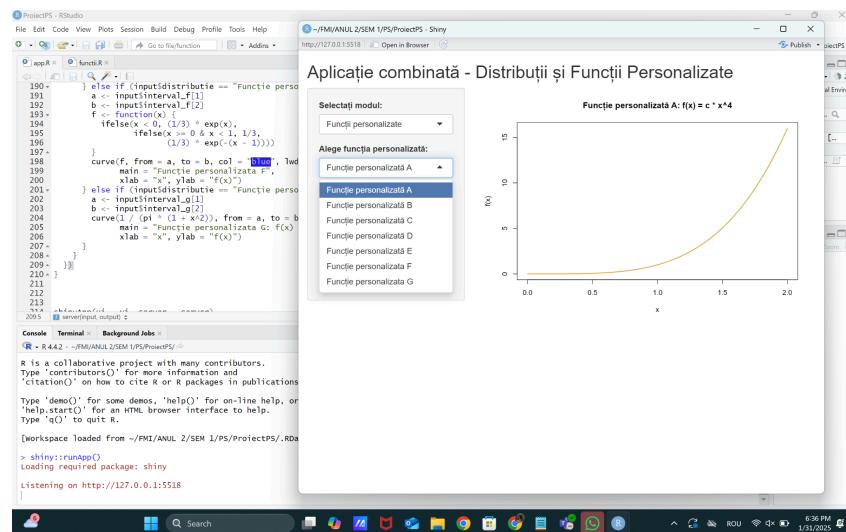
Alegerea funcției de repartiție empirică pentru ilustrare este foarte utilă pentru a vedea cum sunt distribuite valorile efectiv generate. De exemplu, la o distribuție Normală $N(0,1)$, ECDF-ul poate fi comparat cu forma teoretică a repartiției cumulative a Normalelor, iar dacă transformăm variabila în $3-2X$, putem vedea cum se deplasează treptele corespunzătoare acestei noi distribuții.

De asemenea, folosirea `ggplot2` facilitează personalizarea graficului (culori, titluri, etichete, etc.) și oferă un rezultat estetic și consistent.

Prin această structură — generarea unui eșantion random, aplicarea unei transformări și afișarea rezultatelor sub formă de funcție de repartiție empirică — reușim să acoperim rapid o serie largă de scenarii didactice și să ne jucăm cu distribuțiile și transformările lor. Aceasta le permite utilizatorilor să obțină o intuiție imediată asupra modului în care datele se distribuie și cum se modifică atunci când intervenim cu diverse operații (cum ar fi înmulțirea cu -2 sau ridicarea la pătrat).

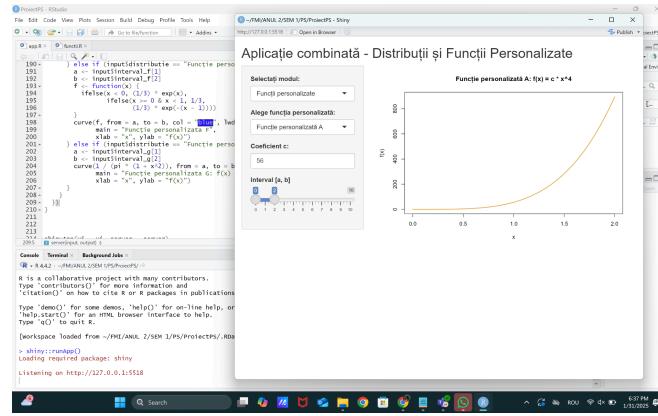
5.2 Modul „Functii personalizate”

Aici, aplicația folosește un meniu separat (`selectInput("distributie", ...)`) și, în funcție de selecție, apare un set de parametri/controale diferite. De exemplu:



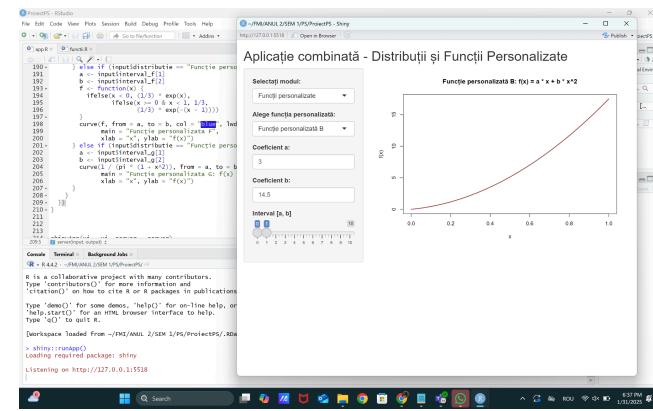
- Funcția A: $f(x) = c \cdot x^4$, pentru $x \in [0,2]$. Aplicația afișează un slider pentru a alege intervalul de plot (implicit $[0,2]$) și un input numeric pentru c.

În cod: `curve(c * x^4, from = a, to = b, col = "orange", lwd = 2)`



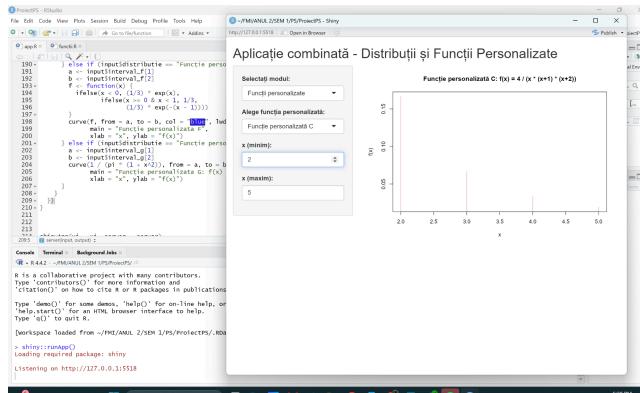
- Funcția B: $f(x) = ax + bx^2$, cu $x \in (0,1)$. Parametrii a și b sunt introdusi în UI drept numericInput, iar intervalul e $[0,1]$.

Plotarea se face tot prin curve().



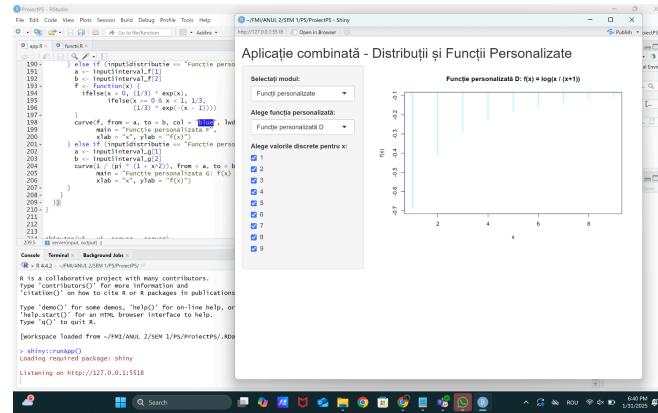
- Funcția C: $\frac{4}{[x(x+1)(x+2)]}$ pentru valori întregi de x.

Aici am preferat să o reprezentăm discret (ex. type="h"), pentru a semăna cu o funcție de masă (PMF).



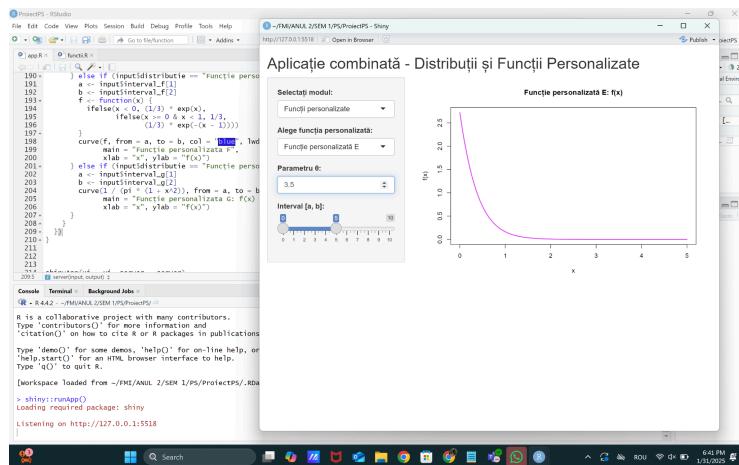
- Funcția D: $\log\left(\frac{x}{x+1}\right)$. Este tot o funcție discretă în codul nostru, căci am invitat utilizatorul să selecteze valori din {1,2,...,9}.

Plotul cu type="h" redă stâlpi verticali pentru fiecare x.



- Funcția E: $\frac{\theta^2}{(1+\theta)}(1+x)e^{-\theta x}$, $x > 0$, $\theta > 0$.

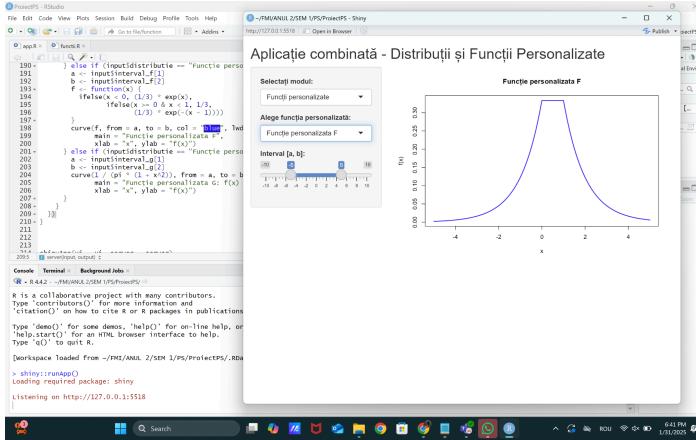
Aceasta este plotată cu curve(), parametrul θ fiind personalizabil.



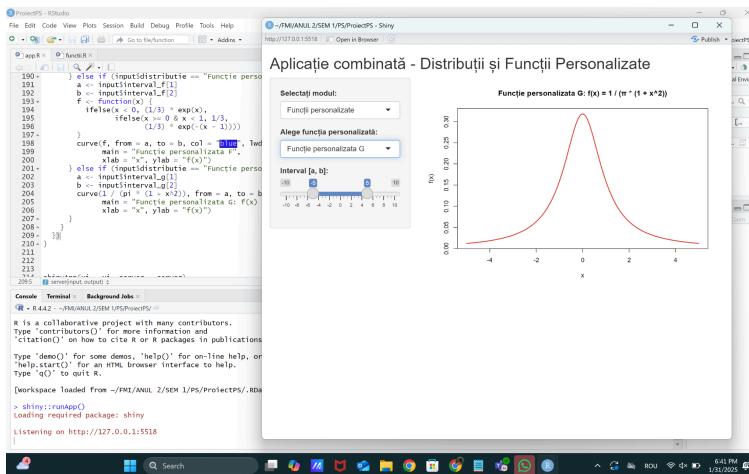
- Funcția F: O compusă definită pe trei regiuni:

$$f(x) = \begin{cases} \frac{1}{3}e^x, & x < 0 \\ \frac{1}{3}, & 0 \leq x < 1 \\ \frac{1}{3}e^{-(x-1)}, & x \geq 1 \end{cases}$$

În cod, s-a folosit un ifelse imbricat. Utilizatorul poate extinde domeniul la $[-10, 10]$, de exemplu, prin slider.



- Funcția G: $\frac{1}{\pi(1+x^2)}$ – aceasta este cunoscută drept funcția de densitate Cauchy, plotată tot cu curve().



Desi nu am inclus direct în interfață un calcul explicit al mediei și varianței, mare parte dintre aceste funcții pot fi integrate (dacă sunt densități valide) pentru a obține parametrii ceruți – se pot face scripturi suplimentare pentru a vedea că, de pildă, Cauchy nu are medie și varianță finite în sens clasic, iar funcția E (sau F) ar avea alt tip de discuție.

6. Implementarea Practică – Explicații Detaliate ale Codului

În continuare, vom prezenta fișierul principal `app.R` (sau scriptul unic ce conține UI și server), alături de explicații pas cu pas. Îl puteți rula direct în RStudio sau folosind `runApp()`.

6.1 Biblioteci și Interfață Utilizatorului (UI)

```
# Biblioteca necesara
```

```
library(shiny)
```

```
library(ggplot2)
```

- **library(shiny)**: încarcăm pachetul Shiny pentru a construi interfața și logica aplicației.
- **library(ggplot2)**: îl folosim pentru reprezentări grafice, în special `stat_ecdf` și pentru personalizarea temelor.

```
# Interfața utilizatorului (UI)
ui <- fluidPage(
  titlePanel("Aplicație combinată - Distribuții și Funcții Personalizate"),
  sidebarLayout(
    sidebarPanel(
      selectInput("mod", "Selectați modul:",
                 choices = c("Distribuții variabile aleatoare",
                             "Funcții personalizate")),
      # Panou pentru distribuții
      conditionalPanel(
        condition = "input.mod == 'Distribuții variabile aleatoare'",
        selectInput(
          inputId = "distribution",
          label = "Selectați distribuția:",
          choices = c("Normală N(0,1)", "Normală N( $\mu, \sigma^2$ )",
                     "Exponențială Exp( $\lambda$ )", "Poisson Pois( $\lambda$ )",
                     "Binomială Binom(r, p)"),
          selected = "Normală N(0,1")
        ),
        numericInput("n", "n (numărul de observații):", 100, min =
1),
        selectInput(
          inputId = "variable",
          label = "Selectați variabila:",
          choices = c("X", "3 - 2X", "2 + 5X", "3X - 2", "5X - 4",
                     "X2", "X3", "Suma ( $\Sigma X$ )", "Suma pătratelor
( $\Sigma X^2$ )"),
          selected = "X"
        ),
        conditionalPanel(
          condition = "input.distribution == 'Normală N( $\mu, \sigma^2$ )'",
          numericInput("mu", " $\mu$  (media):", 0),
          conditionalPanel(
            condition = "input.variable == 'X'",
```

```

        numericInput("sigma", "σ (deviația standard):", 1, min =
0.1),
),
conditionalPanel(
    condition = "input.distribution == 'Exponențială
Exp(λ)'",
    numericInput("lambda", "λ (rata):", 1, min = 0.1)
),
conditionalPanel(
    condition = "input.distribution == 'Poisson Pois(λ)'",
    numericInput("lambda_pois", "λ (media):", 1, min = 0)
),
conditionalPanel(
    condition = "input.distribution == 'Binomială Binom(r,
p)'",
    numericInput("r", "r (număr de experimente):", 10, min =
1),
    numericInput("p", "p (probabilitatea de succes):", 0.5,
min = 0, max = 1)
)
),

# Panou pentru funcții personalizate
conditionalPanel(
    condition = "input.mod == 'Funcții personalizate'",

selectInput("distributie", "Alege funcția personalizată:",
choices = c("Funcție personalizată A",
            "Funcție personalizată B",
            "Funcție personalizată C",
            "Funcție personalizată D",
            "Funcție personalizată E",
            "Funcție personalizată F",
            "Funcție personalizată G")),

conditionalPanel(
    condition = "input.distributie == 'Funcție personalizată
A'",

numericInput("c", "Coeficient c:", value = 1, min = 0),
sliderInput("interval_a", "Interval [a, b]", min = 0,
max = 10, value = c(0, 2))
),
conditionalPanel(
    condition = "input.distributie == 'Funcție personalizată
B'",

numericInput("a", "Coeficient a:", value = 1),
numericInput("b", "Coeficient b:", value = 1),
sliderInput("interval_b", "Interval [a, b]", min = 0,
max = 10, value = c(0, 1)))

```

```

) ,
conditionalPanel(
  condition = "input.distributie == 'Funcție personalizată C'",
  numericInput("x_c_min", "x (minim):", value = 1, min =
1),
  numericInput("x_c_max", "x (maxim):", value = 5, min =
1)
),
conditionalPanel(
  condition = "input.distributie == 'Funcție personalizată D'",
  checkboxGroupInput("x_d_values", "Alege valorile discrete pentru x:",
choices = 1:9, selected = 1:9)
),
conditionalPanel(
  condition = "input.distributie == 'Funcție personalizată E'",
  numericInput("theta", "Parametru θ:", value = 1, min =
0.1),
  sliderInput("interval_e", "Interval [a, b]:", min = 0,
max = 10, value = c(0, 5))
),
conditionalPanel(
  condition = "input.distributie == 'Funcție personalizata F'",
  sliderInput("interval_f", "Interval [a, b]:", min = -10,
max = 10, value = c(-5, 5))
),
conditionalPanel(
  condition = "input.distributie == 'Funcție personalizata G'",
  sliderInput("interval_g", "Interval [a, b]:", min = -10,
max = 10, value = c(-5, 5))
)
)

mainPanel(
  plotOutput("distPlot"),
  verbatimTextOutput("summary")
)
)
)

```

Explicații-cheie:

- `fluidPage(...)`: construiește un layout flexibil, adaptabil la dimensiunea ferestrei de browser.
- `titlePanel("...")`: afișează un titlu în partea de sus a aplicației.
- `sidebarLayout(...)`: împarte ecranul în două regiuni – `sidebarPanel` pentru controale și `mainPanel` pentru rezultate.
- `selectInput("mod", ...)`: utilizatorul poate selecta unul din cele două moduri ("Distribuții variabile aleatoare" vs. "Funcții personalizate").
- `conditionalPanel(condition = "...")`: afisează dinamic componentele din interior doar dacă expresia JavaScript este adevărată. Aici, e folosită pentru a separa controalele aferente distribuțiilor de cele aferente funcțiilor personalizate.
- **Componente pentru distribuții** (e.g. `numericInput("mu", ...)`, `numericInput("lambda", ...)`) apar doar dacă distribuția selectată le solicită.
- **Componente pentru funcții personalizate**: diferite `conditionalPanel`-uri pentru a ajusta parametrii și intervalul de plot.

6.2 Logica Serverului (Server)

```
# Logica serverului (Server)
server <- function(input, output) {
  output$distPlot <- renderPlot({
    if (input$mod == "Distribuții variabile aleatoare") {
      n <- input$n
```

- `server <- function(input, output)`: definim o funcție care primește lista de `input` (valorile setate de utilizator) și pregătește obiecte `output` (ploturi, texte etc.).
- `renderPlot({...})`: instruim Shiny să genereze și să afișeze un plot. Orice schimbare în `input` reactivează acest cod.

6.2.1 Generarea și transformarea variabilei aleatoare

```
# Generăm distribuția aleatoare

  if (input$distribution == "Normală N(0,1)") {

    data <- rnorm(n, mean = 0, sd = 1)

  } else if (input$distribution == "Normală N( $\mu$ ,  $\sigma^2$ )") {

    data <- rnorm(n, mean = input$mu, sd = input$sigma)

  } else if (input$distribution == "Exponențială Exp( $\lambda$ )") {
```

```

    data <- rexp(n, rate = input$lambda)

} else if (input$distribution == "Poisson Pois(λ)") {

    data <- rpois(n, lambda = input$lambda_pois)

} else if (input$distribution == "Binomială Binom(r, p)") {

    data <- rbinom(n, size = input$r, prob = input$p)

}

```

- **if/else** pe baza selecției utilizatorului (**input\$distribution**).
- Apelurile la **rnorm**, **rexp**, **rpois**, **rbinom** creează un vector **data** cu **n** observații.
- **Parametrii** (**mean**, **sd**, **rate**, etc.) sunt citiți din **input\$...**

```

# Calculăm variabila selectată

variable_data <- switch(input$variable,
                         "X" = data,
                         "3 - 2X" = 3 - 2 * data,
                         "2 + 5X" = 2 + 5 * data,
                         "3X - 2" = 3 * data - 2,
                         "5X - 4" = 5 * data - 4,
                         "X^2" = data^2,
                         "X^3" = data^3,
                         "Suma (ΣX)" = cumsum(data), # Suma
cumulativă

                         "Suma pătratelor (ΣX^2)" =
cumsum(data^2) # Suma pătratelor

)

```

- **switch(...)**: alege transformarea dorită. Dacă utilizatorul alege "3 - 2X", atunci **variable_data** devine **3 - 2 * data**.
- Notăm că "**Suma (ΣX)**" și "**Suma pătratelor (ΣX^2)**" sunt implementate prin **cumsum**, ceea ce returnează un vector al sumelor cumulative, nu o singură valoare. Astfel, vom avea un vector de aceeași lungime **n**, ale cărui elemente sunt sumele parțiale.

```
# Creăm un data frame pentru grafic
```

```

df <- data.frame(x = variable_data)

# Reprezentăm ECDF (funcția de repartiție empirică)

ggplot(df, aes(x)) +
  stat_ecdf(geom = "step", color = "blue") +
  labs(
    title = paste("Funcția de repartitie: ",
input$distribution, " pentru ", input$variable),
    x = "Valori",
    y = "Funcția de repartitie (ECDF)"
  ) +
  theme_minimal()

```

- Construim un **data.frame** simplu, **df**, care conține valori în coloana **x**.
- **ggplot(...)**: inițializăm plotul cu **df** ca sursă de date și estetică **aes(x = x)**.
- **stat_ecdf(geom = "step")**: trasează ECDF-ul ca o funcție în trepte.
- **labs(...)**: adaugă titlu și etichete pentru axe.
- **theme_minimal()**: un stil de graficare mai simplu, minimalist.

6.2.2 Plotarea funcțiilor personalizate

```

} else {

  if (input$distributie == "Functie personalizata A") {

    a <- input$interval_a[1]

    b <- input$interval_a[2]

    c <- input$c

    curve(c * x^4, from = a, to = b, col = "orange", lwd = 2,
          main = "Functie personalizata A: f(x) = c * x^4",
          xlab = "x", ylab = "f(x)")
  }
}
```

- **curve(...)** e o funcție de bază R pentru a trasa o funcție continuă pe un interval **[a, b]**.
- Aici, **c * x^4** reprezintă formula funcției dorite, iar **a**, **b** și **c** provin din input-urile setate de utilizator.

```
} else if (input$distributie == "Functie personalizata B") {
```

```

    a <- input$interval_b[1]

    b <- input$interval_b[2]

    coef_a <- input$a

    coef_b <- input$b

    curve(coef_a * x + coef_b * x^2, from = a, to = b, col =
"brown", lwd = 2,
          main = "Functie personalizata B: f(x) = a * x + b *
x^2",
          xlab = "x", ylab = "f(x)")

```

- Similar, doar că aici folosim parametrii **a** și **b** pentru a genera polinomul $a \cdot x + b \cdot x^2$.

```

} else if (input$distributie == "Functie personalizata C") {

  x_vals <- seq(input$x_c_min, input$x_c_max, by = 1)

  f_vals <- 4 / (x_vals * (x_vals + 1) * (x_vals + 2))

  plot(x_vals, f_vals, type = "h", col = "pink", lwd = 2,
        main = "Functie personalizata C: f(x) = 4 / (x *
(x+1) * (x+2))",
        xlab = "x", ylab = "f(x)")

```

- **Discret:** generăm **x_vals** cu **seq(...)**, incrementând cu 1.
- Calculăm valorile funcției $f(x)=4/[x(x+1)(x+2)]$
- **plot(..., type = "h"):** trasează bare verticale, util pentru funcții discrete (asemănător unui PMF).

```

} else if (input$distributie == "Functie personalizata D") {

  x_vals <- as.numeric(input$x_d_values)

  f_vals <- log(x_vals / (x_vals + 1))

  plot(x_vals, f_vals, type = "h", col = "cyan", lwd = 2,
        main = "Functie personalizata D: f(x) = log(x /
(x+1))",
        xlab = "x", ylab = "f(x)")

```

- Utilizatorul selectează un set de valori discrete via **checkboxGroupInput**.

- Facem transformarea $f(x) = \log(x+1)$
- Reprezentăm tot cu **type = "h"** pentru a sublinia caracterul discret.

```

} else if (input$distributie == "Functie personalizata E") {

  a <- input$interval_e[1]

  b <- input$interval_e[2]

  theta <- input$theta

  curve(((theta^2) / (1 + theta)) * (1 + x) * exp(-theta *
x), from = a, to = b,

        col = "magenta", lwd = 2,

        main = "Functie personalizata E: f(x)",

        xlab = "x", ylab = "f(x)")

```

- $f(x) = \frac{\theta^2}{(1+\theta)}(1+x)e^{-\theta x}$
- Parametrul **theta** și intervalul **[a, b]** sunt introduse de utilizator.

```

} else if (input$distributie == "Functie personalizata F") {

  a <- input$interval_f[1]

  b <- input$interval_f[2]

  f <- function(x) {

    ifelse(x < 0, (1/3) * exp(x),

           ifelse(x >= 0 & x < 1, 1/3,

                  (1/3) * exp(-(x - 1))))}

  curve(f, from = a, to = b, col = "blue", lwd = 2,

        main = "Functie personalizata F",

        xlab = "x", ylab = "f(x)")

```

Aici definim o **functie compusă** pe intervale:

$$f(x) = \begin{cases} \frac{1}{3}e^x, & x < 0 \\ \frac{1}{3}, & 0 \leq x < 1 \\ \frac{1}{3}e^{-(x-1)}, & x \geq 1 \end{cases}$$

- **if else** imbricate ne ajută să evaluăm fiecare punct pe porțiuni de domeniu diferite.

```

} else if (input$distributie == "Funcție personalizata G") {

    a <- input$interval_g[1]

    b <- input$interval_g[2]

    curve(1 / (pi * (1 + x^2)), from = a, to = b, col = "red",
lwd = 2,
            main = "Funcție personalizata G: f(x) = 1 / (π * (1
+ x^2))",

            xlab = "x", ylab = "f(x)")

}

}

}

shinyApp(ui = ui, server = server)

```

- **Funcția personalizată G:** densitatea Cauchy standard: $\frac{1}{[\pi(1+x^2)]}$
- Plotăm în roșu, cu lățimea liniei 2.

7. Dificultăți Întâmpinate și Observații

1. Parametri invalizi:

În cazul Normală (μ, σ), trebuie avut grija ca $\sigma > 0$.

Pentru Binomială, p trebuie să fie în $(0,1)$. Am adăugat restricții minime/maxime la intrări.

2. Discrepanțe la transformările cumulate: **cumsum(data)** înseamnă sumă cumulativă, ceea ce e logic, dar puțin diferit de ideea de „sumă totală” a n observații (care s-ar fi putut face cu **sum(data)**).

Am optat pentru **cumsum()** deoarece oferă un vector pe care îl putem reprezenta tot cu **stat_ecdf**.

3. **Discreție vs. Continuitate**: Pentru Poisson și Binomială, repartitia e discretă; totuși, **stat_ecdf** o va afișa sub formă de trepte. E important să înțelegem că e tot o **ecdf**, însă spațiul valorilor posibile ale lui X este unul discret.
 4. Funcțiile personalizate: Am combinat densități (de ex. Cauchy) cu funcții care încă pot fi interpretate ca densități pe un anume suport (dacă parametrii sunt potrivitori). Pentru a fi complet riguroși, ar trebui verificate integralele să fie 1.
-

8. Posibilități de Extindere

- Compararea ECDF cu CDF-ul teoretic: Ar fi util un overlay grafic (ex. `geom_line`) pentru a vedea cât de aproape este eșantionul de distribuția ideală.
 - Calcul efectiv al momentelor (media, varianța) direct în Shiny, cu formate text. De pildă, după generarea eșantionului, să afișăm `mean(variable_data)` și `var(variable_data)` în `verbatimTextOutput`. Similar, pentru funcțiile personalizate (dacă integrale).
 - Tab-uri multiple: În loc de un singur ecran, se pot organiza mai multe tab-uri (unul pentru distribuții, altul pentru funcții personalizate).
 - Export de rezultate: Se poate implementa un buton de „Download” care să permită salvarea graficelor sau a datelor generate în format CSV.
-

9. Concluzii Finale

Aplicația realizată îmbină într-o manieră armonioasă partea de programare reactivă în R Shiny cu elementele fundamentale de statistică și probabilități. Prin intermediul acesteia, am reușit:

- Să generăm eșantioane din distribuții clasice (Normală, Exponențială, Poisson, Binomială).
- Să transformăm variabilele generate conform cerințelor (ex. $3 - 2X$, X^2 , ΣX).
- Să vizualizăm funcția de repartie empirică (ECDF) a acestor transformări, făcând astfel tangibile noțiunile abstractive.
- Să explorăm un set de funcții personalizate (mai simple sau mai speciale), totul într-o manieră interactivă.

Dincolo de utilitatea didactică, proiectul prezintă și un punct de pornire pentru analize mai sofisticate, experimentări cu distribuții mai puțin uzuale, studierea convergențelor de tip CLT sau generarea rapidă de scenarii test.

Sperăm ca aplicația să constituie un sprijin pentru oricine își dorește să își sedimenteze cunoștințele de statistică, combinând teoria cu partea interactivă și exploratorie, definitorie pentru un proces de învățare aprofundat.

Referințe:

1. **Shiny** (RStudio)

<https://shiny.rstudio.com/>

<https://cran.r-project.org/web/packages/shiny/shiny.pdf>

2. **ggplot2**

<https://ggplot2.tidyverse.org/>

<https://cran.r-project.org/web/packages/ggplot2/ggplot2.pdf>

3. **R** (limbajul în sine)

<https://cran.r-project.org/manuals.html>