

Documentație Proiect

Probabilitate si Statistica

Membri:

Gheorghe Bogdan-Alexandru

Andrei Cristian-David

Sincari Sebastian-George

Cosor Mihail

Ianuarie 2025

Contents

Cerinta I	2
Cerinta II	7
Cerinta III	8
Descrierea Problemei	8
Aspecte Teoretice	8
Reprezentări Grafice	9
Pachete Software Folosite	9
Codul Aplicației	10
app.R	10
ui.R	10
server.R	11
utils.R	12
normals.R	12
Dificultăți Întâmpinate	14
Probleme Deschise	14
Concluzii	14

Cerinta I

Descrierea Problemei

Se consideră o activitate care presupune parcurgerea secvențială a n etape. Timpul necesar finalizării etapei i de către o persoană A este o variabilă aleatoare $T_i \sim \text{Exp}(\lambda_i)$.

După finalizarea etapei i , A va trece în etapa $i + 1$ cu probabilitatea α_i sau va opri lucrul cu probabilitatea $1 - \alpha_i$.

Fie T timpul total petrecut de persoana A în realizarea activității respective.

```
1 set.seed(123)
2 n <- 100 # Nr etape
3 lambda <- runif(n, 0.5, 2) # Rata exp pt fiecare etapa
4 alpha <- runif(n-1, 0.8, 1) # Prob trecerii la urmatoarea etapa
5 alpha <- c(1, alpha) # Prob ca o persoana sa participe la prima etapa este 100%
6 nrSimulari <- 1000000 # Nr simulari
7
8 simulator <- function() {
9   Ti <- rexp(n, rate = lambda) # Timpul pentru fiecare etapa
10
11   stop_point <- which(runif(n - 1) > alpha[-1])[1] # Determinam momentul opririi
12
13   # Timpul total va fi cel pana la oprire/finalul vectorului daca nu se opreste
14   if (!is.na(stop_point)) {
15     return(sum(Ti[1:stop_point]))
16   } else {
17     return(sum(Ti))
18   }
19 }
20
21 # Simulam 10^6
22 valoriT <- replicate(nrSimulari, simulator())
```

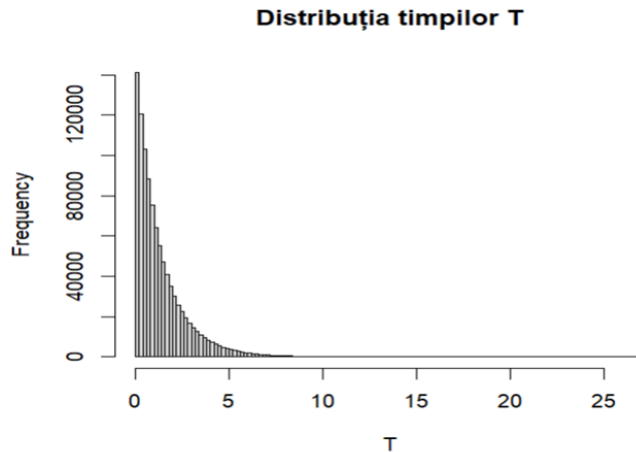
1. Construiți un algoritm în R care simulează 10^6 valori pentru v.a. T și în baza acestora aproximați $E(T)$. Reprezentați grafic într-o manieră adecvată valorile obținute pentru T . Ce puteți spune despre repartiția lui T ?

```
1 approx_E_T <- mean(valoriT)
2
3 hist(valoriT, breaks = 100, main = "Distributia timpilor T", xlab = "T")
```

Explicații:

Pentru calculul aproximativ al mediei am făcut pur și simplu media valorilor rezultate în urma simulării.

Afișând în histograma valorile obținute în `valoriT` rezultă distribuția lui T . Se poate observa că reprezentarea grafică a timpilor T urmează o distribuție exponențială.



Rezultat:

```
1 approx_E_T = 9.51113170988666
```

2. Calculați valoarea exactă a lui $E[T]$ și comparați cu valoarea obținută prin simulare.

```
1 exact_E_T <- sum(1 / \lambda * cumprod(alpha))
```

Explicații:

Valoarea exactă a lui $E[T]$ se calculează după următoarea formulă:

$$E[T] = E[T_1] + \alpha_1 E[T_2] + \alpha_1 \alpha_2 E[T_3] + \dots + \alpha_1 \alpha_2 \dots \alpha_{n-1} E[T_n]$$

Am folosit funcția `cumprod`, care returnează un vector cu produsul cumulativ. α are 99 de valori deoarece există $n - 1$ pași pentru trecerea de la o etapă la alta. Am atașat acestui vector la început valoarea 1, deoarece probabilitatea ca persoana să participe la prima etapă este 100. Apoi am înmulțit produsul cumulativ cu $1/\lambda$, deoarece $X \sim \text{Exp}(\lambda) \Rightarrow E[X] = 1/\lambda$. În final, am făcut suma tuturor mediilor pe fiecare etapă, rezultând $E[T]$.

În urma analizării rezultatului exact observăm că este foarte apropiat de rezultatul aproximativ obținut prin simulare.

Rezultat:

```
1 exact_E_T = 9.51299920302413
```

3. În baza simulărilor de la 1. aproximați probabilitatea ca persoana A să finalizeze activitatea.

```
1 prob_finalizare <- mean(valoriT >= sum(1 / lambda))
```

Explicații:

Am calculat probabilitatea ca o persoana sa finalizeze fiecare etapa luand ca exemplu in simularea noastra un numar de 100 de etape si o probabilitate de a trece de la o etapa la alta $\lambda=80$

In scrierea codului am calculat valoarea teoretica a timpului total pe care o persoana trebuie sa il petreaca in cele 100 de etape pentru a spune ca a finalizat. Astfel am calculat media unui vector de valori booleene care valoarea true corespunde rezultatului unei simulari $\lambda=$ valoarea timpului adica persoanele care au finalizat respectiv false pentru persoanele care nu au reusit sa finalizeze. Rezultatul este mic datorita numarului de etape, acesta ar creste o data cu micsorarea numarului de etape/ ar scadea o data cu cresterea numarului de etape. Totodata, rezultatul va scadea odata cu micsorarea marginii inferioare a valorilor probabilitatilor din alpha si invers.

Rezultat:

```
1      prob_finalizare = 0.00004
```

4. În baza simulărilor de la 1. aproximați probabilitatea ca persoana A să finalizeze activitatea într-un timp mai mic sau egal cu σ .

```
1      sigma <- 94
2      prob_sigma <- mean(valoriT <= sigma & valoriT >= sum(1 / lambda))
```

Explicatii:

Ne-am folosit de modalitatea de rezolvare a exercitiului anterior, astfel am folosit un vector de valori booleene care verifica 2 conditii $j = \text{sigma}$ si $i = \text{valoarea timpului total}$, apoi am facut media si ne-a rezultat valoarea probabilitatii ca o persoana sa finalizeze intr-un timp mai mic decat sigma. Este normal ca valoarea acestei probabilitati sa fie mai mica sau egala cu valoarea probabilitatii de a finaliza.

Rezultat:

```
1      prob_sigma = 0.000007
```

5. În baza simulărilor de la 1. determinați timpul minim și respectiv timpul maxim în care persoana A finalizează activitatea și reprezentați grafic timpii de finalizare a activității din fiecare simulare. Ce puteți spune despre repartiția acestor timpi de finalizare a activității?

```
1      timpMin <- min(valoriT[valoriT >= sum(1 / lambda)] )
2      timpMax <- max(valoriT[valoriT >= sum(1 / lambda)] )
3      cat("Timpul minim:", timpMin, "Timpul maxim:", timpMax, "\n")
4
5      hist(valoriT[valoriT >= sum(1 / lambda)], breaks=50,
6           main="Distributia timpilor de finalizare",
7           xlab="Timpul de finalizare (T)", ylab="Frecventa",
8           col="lightblue", border="black")
```

Explicatii:

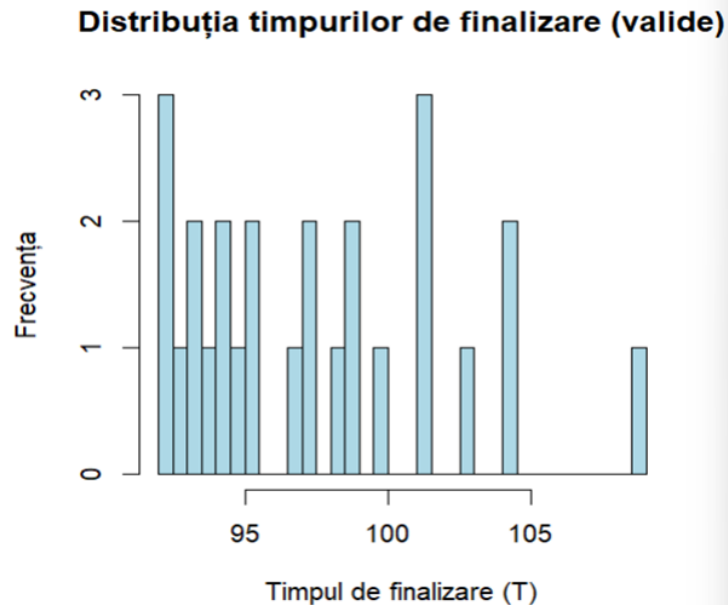
Am considerat variabilele `timpMin` și `timpMax` pentru valorile minime și maxime ale timpilor persoanelor care au finalizat evenimentul. Am folosit funcția `min` pentru minim și, ca parametru, am dat lista de timpi simulați, filtrată de valorile mai mari decât media timpului total. Am folosit din nou, pentru reprezentarea grafică, o histogramă, deoarece aceasta ajută la identificarea repartiției. Două observații sunt faptul că numărul de persoane care au reușit să finalizeze evenimentul este foarte mic în comparație cu numărul de simulări și că se observă că repartiția este uniformă, deoarece pentru alegerea valorilor random am folosit `runif`.

Rezultat:

```
1      timpMax = 108.54443085275
2      timpMin = 92.0171680316328
```

6. În baza simulărilor de la 1. aproximați probabilitatea ca persoana A să se oprească din lucru înainte de etapa k , unde $1 < k \leq n$. Reprezentați grafic probabilitățile obținute într-o manieră corespunzătoare. Ce puteți spune despre repartiția probabilităților obținute?

```
1      k <- 5
2      PStopK <- mean(sapply(valoriT, function(x) {
3      if(x <= sum(1/lambda[1:k-1]))
4      return (TRUE)
```



```

5     else
6         return(FALSE)
7     )))

```

Explicații:

Am creat o variabilă numită `PstopK` care primește probabilitatea ca persoana A să se oprească din lucru înainte de etapa k . Pentru această probabilitate am aplicat ca parametru unei funcții fiecare valoare simulată. Funcția returnează TRUE dacă timpul este mai mic decât media timpului total până la pasul $k-1$ (adică poate face maxim $k-1$ pași), și FALSE altfel. Variabila primește media valorilor din lista booleană.

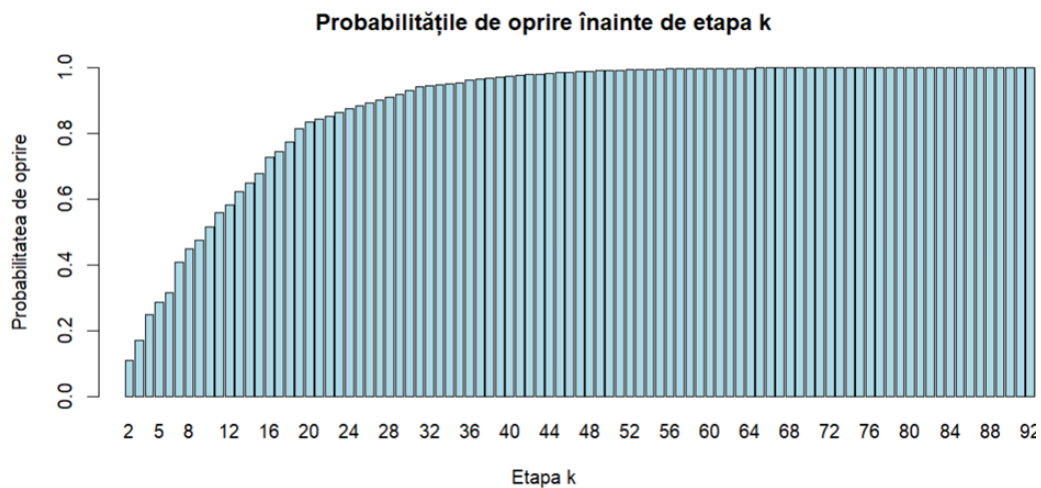
Rezultat:

```

1     PstopK = 0.284293  pentru k=5
2     PstopK = 0.51482  pentru k=10
3     PstopK = 0.991332 pentru k=50

1     calculate_PStopBeforeK <- function(valoriT, lambda, alpha, n) {
2     # Calculam timpii cumulativi pentru fiecare etapa
3     cumulative_times <- cumsum(1 / lambda)
4
5     # Calculam probabilitatile pentru fiecare k
6     PStopBeforeK <- sapply(2:n, function(k) {
7         mean(valoriT <= cumulative_times[k - 1])
8         # Probabilitatea de oprire inainte de etapa k
9     })
10
11     return(PStopBeforeK)
12 }
13 PStopBeforeK <- calculate_PStopBeforeK(valoriT, lambda, alpha, n)
14
15 barplot(PStopBeforeK, names.arg = 2:n, col = "lightblue",
16         main = "Probabilitatile de oprire inainte de etapa k",
17         xlab = "Etapa k", ylab = "Probabilitatea de oprire",
18         xlim = c(1, n), ylim = c(0, 1))

```



Observații:

Se poate observa că o persoană, cu cât ajunge mai departe în etape, cu atât are șanse mai mari să finalizeze evenimentul, iar în etapele inițiale șansele sunt foarte mici. Aceste observații sunt date datorită pantei abrupte în faza inițială, respectiv panta lină de la final.

Cerinta II

Mesaj pentru Sebi: (Pula n-are scoala ca sta sub banca)

Cerinta III

Descrierea Problemei

Scopul acestui proiect este de a construi o aplicație web interactivă folosind framework-ul **Shiny** din **R**, pentru a reprezenta grafic funcțiile de repartiție (CDF) ale unor variabile aleatoare definite conform cerinței din enunț. Aplicația permite vizualizarea CDF-urilor pentru distribuții normale, exponențiale, binomiale, Poisson.

Aspecte Teoretice

Pentru dezvoltarea aplicației, am utilizat următoarele aspecte teoretice importante:

- **Distribuția Normală:** Este definită ca o variabilă aleatoare continuă $X \sim N(\mu, \sigma^2)$, unde μ reprezintă media, iar σ^2 varianța.

Formula funcției de repartiție cumulativă:

$$F(x) = P(X \leq x) = \int_{-\infty}^x \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(t-\mu)^2}{2\sigma^2}} dt$$

- **Distribuția Exponențială:** O variabilă aleatoare $X \sim \text{Exp}(\lambda)$ este folosită pentru modelarea timpilor de așteptare între evenimente.

Formula funcției de repartiție cumulativă:

$$F(x) = P(X \leq x) = \begin{cases} 1 - e^{-\lambda x}, & \text{pentru } x \geq 0 \\ 0, & \text{pentru } x < 0 \end{cases}$$

- **Distribuția Poisson:** Reprezintă numărul de evenimente într-un interval fix de timp și este notată $X \sim \text{Poisson}(\lambda)$.

Formula funcției de repartiție cumulativă:

$$F(x) = P(X \leq x) = \sum_{k=0}^{\lfloor x \rfloor} \frac{\lambda^k e^{-\lambda}}{k!}$$

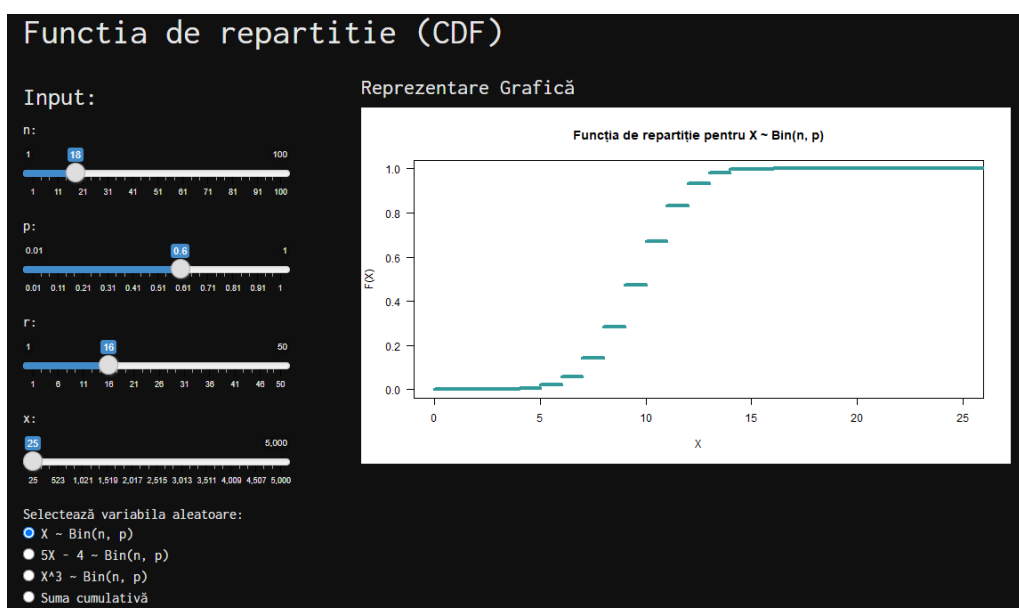
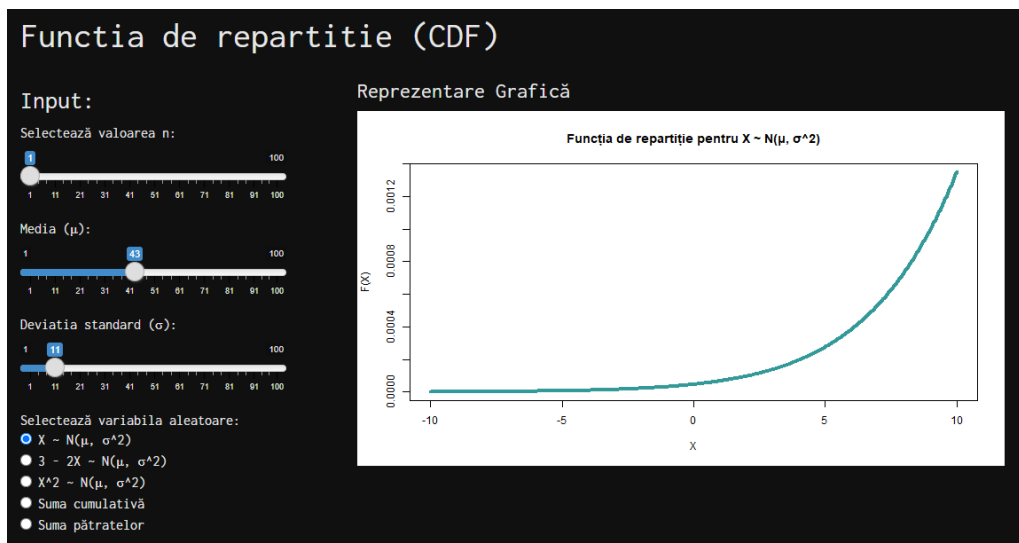
- **Distribuția Binomială:** Este definită de parametrii n și p și descrie numărul de succese în n experimente independente.

Formula funcției de repartiție cumulativă:

$$F(x) = P(X \leq x) = \sum_{k=0}^{\lfloor x \rfloor} \binom{n}{k} p^k (1-p)^{n-k}$$

Reprezentări Grafice

Aplicația oferă reprezentări grafice ale funcțiilor de repartiție pentru fiecare dintre variabilele aleatoare definite în cerințe. În cele ce urmează, vom ilustra câteva capturi de ecran din aplicație.



Pachete Software Folosite

Am folosit următoarele pachete software pentru implementarea proiectului:

- **shiny**: Crearea aplicației interactive.
- **bslib**: Personalizarea interfeței grafice a aplicației folosind teme Bootstrap.
- **graphics**: Generarea reprezentărilor grafice ale funcțiilor de repartiție.

Codul Aplicației

Codul aplicației a fost structurat în mai multe fișiere pentru a asigura o separare clară a logicii și o reutilizare ușoară a funcțiilor. Această abordare are ca scop evitarea redundanței și îmbunătățirea clarității codului, permițând adăugarea sau modificarea distribuțiilor fără a afecta structura generală a aplicației.

app.R

Fișierul principal **app.R** conține doar partea de inițializare a aplicației Shiny, inclusiv încărcarea fișierelor **ui.R**, **utils.R** și **server/server.R**. Aceasta facilitează gestionarea centralizată a componentelor aplicației. Fiecare distribuție are propriile fișiere în directorul **server**, în care sunt definite logicile de generare a graficelor corespunzătoare.

```
1 library(shiny)
2 library(bslib)
3
4 source("utils.R")
5 source("ui.R")
6 source("server/server.R")
7
8 shinyApp(ui = ui, server = server)
```

ui.R

Interfața utilizatorului este creată folosind funcția **fluidPage**, care organizează interfața pe tab-uri, câte unul pentru fiecare distribuție (normală, exponențială, binomială, Poisson). Fiecare tab este generat folosind funcția personalizată **create_tab**, care asociază un input specific distribuției respective și o zonă de afișare a graficului.

```
1 source("server/server.R")
2
3 create_tab <- function(tab_title, title, distribution) {
4   tabPanel(
5     tab_title,
6     div(
7       class = "container",
8       h1(title),
9       div(
10        class = "row",
11        div(
12          class = "col-4",
13          tags$h3("Input:"),
14          switch(
15            distribution,
16            BINOMIALA = create_binom_slider(),
17            NORMALA_STANDARD = create_std_normal_slider(),
18            NORMALA = create_normal_slider(),
19            EXPONENTIALA = create_exponential_slider(),
20            POISSON = create_pois_slider()
21          )
22        ),
23        div(
24          class = "col-8",
25          h4("Reprezentare Grafica"),
26          get_output_distribution(distribution)
27        )
28      )
29    )
30  }
31 }
32
```

```

33 ui <- fluidPage(
34   theme = bs_theme(
35     version = 4,
36     bg = "#101010",
37     fg = "#FFF",
38     primary = "#E69F00",
39     base_font = font_google("Inconsolata")
40   ),
41   navbarPage(
42     "CDF",
43     create_tab(
44       "Normala Standard",
45       "Functia de repartitie (CDF)",
46       NORMALA_STANDARD
47     ),
48     create_tab(
49       "Normala",
50       "Functia de repartitie (CDF)",
51       NORMALA
52     ),
53     create_tab(
54       "Exponentiala",
55       "Functia de repartitie (CDF)",
56       EXPONENTIALA
57     ),
58     create_tab(
59       "Binomiala",
60       "Functia de repartitie (CDF)",
61       BINOMIALA
62     ),
63     create_tab(
64       "Poisson",
65       "Functia de repartitie (CDF)",
66       POISSON
67     )
68   )
69 )

```

server.R

Acesta este punctul central al logicii serverului. Codul serverului include apeluri către fișierele individuale ale fiecărei distribuții, cum ar fi `std_normal.R`, `normal.R`, `binom.R` etc. Acest lucru permite fiecărei distribuții să fie tratată separat și modular, facilitând adăugarea sau modificarea distribuțiilor fără a afecta alte componente.

```

1  source("server/std_normal.R")
2  source("server/normal.R")
3  source("server/exponential.R")
4  source("server/binom.R")
5  source("server/poisson.R")
6
7  server <- function(input, output, session) {
8    std_normal_server(input, output, session)
9    normal_server(input, output, session)
10   exponential_server(input, output, session)
11   binom_server(input, output, session)
12   pois_server(input, output, session)
13 }

```

utils.R

Acest fișier conține definiții comune, cum ar fi identificatorii fiecărei distribuții și funcția `get_output_distribution` care returnează componenta grafică potrivită pentru distribuția selectată. Aceasta contribuie la eliminarea duplicării codului și la centralizarea logicii de redirectionare a graficelor.

```
1  NORMALA_STANDARD <- "NORMALA_STANDARD"
2  NORMALA <- "NORMALA"
3  EXPONENTIALA <- "EXPONENTIALA"
4  BINOMIALA <- "BINOMIALA"
5  POISSON <- "POISSON"
6
7  get_output_distribution <- function(distribution) {
8    switch(
9      distribution,
10     NORMALA_STANDARD = plotOutput("std_normal_plot"),
11     NORMALA = plotOutput("normal_plot"),
12     EXPONENTIALA = plotOutput("exponential_plot"),
13     BINOMIALA = plotOutput("binom_plot"),
14     POISSON = plotOutput("pois_plot")
15   )
16 }
```

normals.R

Generarea graficelor: Codul pentru reprezentarea grafică este similar pentru toate distribuțiile, însă exemplificarea este prezentată doar pentru distribuția normală în fișierul `normals.R`. Fiecare grafic este generat pe baza parametrilor selectați de utilizator (cum ar fi media μ și deviația standard σ) și utilizează funcția corespunzătoare de calcul al funcției de repartiție cumulativă. Iată principalele tipuri de grafice generate:

- ****Graficul funcției de repartiție cumulativă (CDF)**:** Este reprezentată funcția $F(x) = P(X \leq x)$, calculată cu funcția R corespunzătoare fiecărei distribuții (de exemplu, `pnorm` pentru distribuția normală).
- ****Transformări ale variabilelor**:** De exemplu, pentru $3 - 2X$ sau X^2 , transformările sunt aplicate asupra datelor înainte de a calcula și reprezenta $F(x)$.
- ****Sumă cumulativă**:** Pentru a reprezenta suma cumulativă $\sum X_i$ sau $\sum X_i^2$, se utilizează funcția `cumsum` din R, iar graficul este construit prin afișarea acestor sume pentru fiecare n .

```
1  create_normal_slider <- function() {
2    tagList(
3      sliderInput(
4        inputId = "normal_n",
5        label = "Selectează valoarea n:",
6        min = 1,
7        max = 100,
8        value = 10,
9        step = 1
10     ),
11     sliderInput(
12       inputId = "normal_mu",
13       label = "Media (\u03BC):",
14       min = 1,
15       max = 100,
16       value = 10,
17       step = 1
18     ),
19     sliderInput(
20       inputId = "normal_sigma",
21       label = "Deviația standard (\u03C3):",
22       min = 1,
23       max = 100,
```

```

24     value = 10,
25     step = 1
26   ),
27   radioButtons(
28     inputId = "normal_var",
29     label = "Selecteaza variabila aleatoare:",
30     choices = list(
31       "X ~ N(\u03BC, \u03C3^2)" = "var1",
32       "3 - 2X ~ N(\u03BC, \u03C3^2)" = "var2",
33       "X^2 ~ N(\u03BC, \u03C3^2)" = "var3",
34       "Suma cumulativa" = "var4",
35       "Suma patratelor" = "var5"
36     ),
37     selected = "var1"
38   )
39 )
40 }
41
42 normal_server <- function(input, output, session) {
43   output$normal_plot <- renderPlot({
44     var <- input$normal_var
45
46     if (var == "var1") {
47       x <- seq(-10, 10, length.out = 500)
48       cdf_values <- pnorm(x, mean = input$normal_mu, sd = input$normal_sigma)
49       plot(
50         x, cdf_values,
51         type = "l",
52         lwd = 4,
53         col = "#339999",
54         xlab = "X",
55         ylab = "F(X)",
56         main = "Functia de repartitie pentru X ~ N(\u03BC, \u03C3^2)"
57       )
58     } else if (var == "var2") {
59       x <- seq(-10, 10, length.out = 500)
60       transformed_x <- 3 - 2 * x
61       cdf_values <- pnorm(transformed_x, mean = input$normal_mu, sd = input$normal_sigma)
62       plot(
63         x, cdf_values,
64         type = "l",
65         lwd = 4,
66         col = "#FF6666",
67         xlab = "3 - 2X",
68         ylab = "F(3 - 2X)",
69         main = "Functia de repartitie pentru 3 - 2X ~ N(\u03BC, \u03C3^2)"
70       )
71     } else if (var == "var3") {
72       x <- seq(-10, 10, length.out = 500)
73       transformed_x <- x^2
74       cdf_values <- pnorm(transformed_x, mean = input$normal_mu, sd = input$normal_sigma)
75       plot(
76         x, cdf_values,
77         type = "l",
78         lwd = 4,
79         col = "#3399FF",
80         xlab = "X^2",
81         ylab = "F(X^2)",
82         main = "Functia de repartitie pentru X^2 ~ N(\u03BC, \u03C3^2)"
83       )
84     } else if (var == "var4") {
85       n <- input$normal_n
86       X <- rnorm(n, mean = input$normal_mu, sd = input$normal_sigma)
87       S_n <- cumsum(X)
88       plot(
89         1:n, S_n,

```

```

90     type = "l",
91     lwd = 2,
92     col = "#FF9900",
93     xlab = "n",
94     ylab = "\u2211 X_i",
95     main = "Suma cumulativa a variabilelor aleatoare X_i"
96   )
97   } else if (var == "var5") {
98     n <- input$normal_n
99     X <- rnorm(n, mean = input$normal_mu, sd = input$normal_sigma)
100    S_n2 <- cumsum(X^2)
101    plot(
102      1:n, S_n2,
103      type = "l",
104      lwd = 2,
105      col = "#33CC33",
106      xlab = "n",
107      ylab = "\u2211 X_i^2",
108      main = "Suma cumulativa a patratelor variabilelor X_i"
109    )
110  }
111 })
112 }

```

Dificultăți Întâmpinate

Dezvoltarea aplicației a implicat rezolvarea unor provocări tehnice importante legate de gestionarea parametrilor și a reprezentărilor grafice pentru fiecare distribuție. Principalele dificultăți întâmpinate sunt:

- **Ajustarea automată a parametrilor pentru fiecare distribuție:** Fiecare distribuție are un set unic de parametri care trebuie adaptați corect pentru a asigura o reprezentare grafică precisă a funcției de repartiție cumulativă (CDF). De exemplu, distribuția normală implică media μ și deviația standard σ , în timp ce distribuția binomială depinde de numărul de experimente n și probabilitatea de succes p . Provocarea a fost legată de crearea unui sistem flexibil în care utilizatorul să poată modifica rapid parametrii în interfață, iar graficele să se actualizeze automat în funcție de aceștia. Această problemă a fost rezolvată prin implementarea unor funcții modulare (de exemplu, `create_normal_slider()` și `create_binom_slider()`) care generează dinamic controale de input specifice fiecărei distribuții.

Probleme Deschise

Deși aplicația funcționează corespunzător și oferă o interfață interactivă pentru vizualizarea funcțiilor de repartiție cumulativă, există câteva direcții de dezvoltare viitoare și probleme deschise:

- **Posibilitatea extinderii aplicației pentru a suporta și alte distribuții:** Momentan, aplicația suportă distribuțiile normală, exponențială, binomială și Poisson. Cu toate acestea, există o varietate de alte distribuții utile în statistică, cum ar fi distribuțiile Bernoulli, binomiale sau geometrice, care ar putea fi integrate cu ușurință în structura modulară a aplicației. O extindere în această direcție ar implica adăugarea fișierelor corespunzătoare în directorul `server` și crearea funcțiilor de input specifice fiecărei noi distribuții.

Concluzii

Funcționalitățile existente permit utilizatorilor să vizualizeze funcțiile de repartiție ale variabilelor normale, exponențiale, binomiale și Poisson, dar structura modulară face posibilă integrarea rapidă a altor distribuții. De asemenea, prin intermediul graficelor dinamice, utilizatorii pot explora nu doar distribuțiile de bază, ci și diverse transformări ale variabilelor aleatoare, cum ar fi transformări liniare și sume cumulative.