Documentaţie Proiect

Probabilitate si Statistica

Membri:
Gheorghe Bogdan-Alexandru
Andrei Cristian-David
Sincari Sebastian-George
Cosor Mihail

Ianuarie 2025

Contents

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.

```
set.seed(123)
n <- 100 # Nr etape
lambda <- runif(n, 0.5, 2) # Rata exp pt fiecare etapa
alpha <- runif(n-1, 0.8, 1) # Prob trecerii la urmatoarea etapa
alpha <- c(1, alpha) # Prob ca o persoana sa participe la prima etapa este 100%
nrSimulari <- 1000000 # Nr simulari


simulator <- function() {
  Ti <- rexp(n, rate = lambda)  # Timpul pentru fiecare etapa

  stop_point <- which(runif(n - 1) > alpha[-1])[1]  # Determinam momentul opririi

  # Timpul total va fi cel pana la oprire/finalul vectorului daca nu se opreste
  if (!is.na(stop_point)) {
    return(sum(Ti[1:stop_point]))
  } else {
    return(sum(Ti))
  }
}

# Simulam 10^6
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?

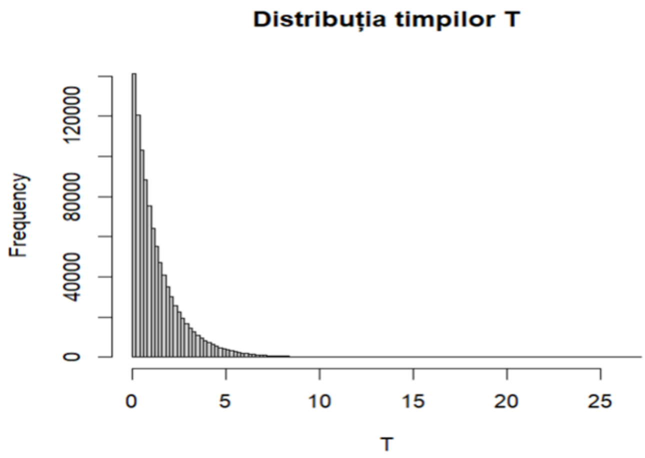
```
approx_E_T <- mean(valoriT)

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 timpiilor T urmează o distribuție exponențială.



Rezultat:

```
approx_E_T = 9.51113170988666
```

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

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

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

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

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:

```
    exact_E_T = 9.51299920302413
```

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

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

Explicatii:

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 ¿=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¿= 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:

```
prob_finalizare = 0.00004
```

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

```
sigma <-94
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 ¡= sigma si ¿= 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:

```
prob_sigma = 0.000007
```

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

```
timpMin <- min(valoriT[valoriT >= sum(1 / lambda)] )
timpMax <- max(valoriT[valoriT >= sum(1 / lambda)] )
cat("Timpul minim:", timpMin, "Timpul maxim:", timpMax, "\n")

hist(valoriT[valoriT >= sum(1 / lambda)], breaks=50,
            main="Distributia timpilor de finalizare",
            xlab="Timpul de finalizare (T)", ylab="Frecventa",
            col="lightblue", border="black")
```

Explicatii:

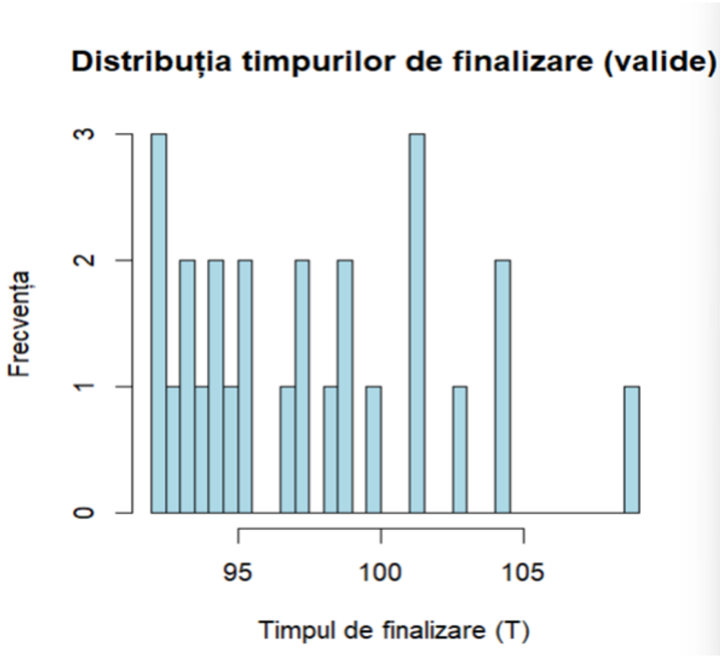
Am considerat variabilele `timpMin` și `timpMax` pentru valorile minime și maxime ale timpilor persoanelor care au finalizat evenimentul. Am folosit funcția `min` pentru minim și, ca parametru, am dat lista de timpi simulați, filtrată de valorile mai mari decât media timpului total. Am folosit din nou, 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:

```
timpMax = 108.54443085275
timpMin = 92.0171680316328
```

6. În baza simulărilor de la 1. aproximaţi probabilitatea ca persoana A să se oprească din lucru înainte de etapa k , unde 1<k≤n . Reprezentaţi grafic probabilităţile obţinute într-o manieră corespunzătoare. Ce puteţi spune despre repartiţia probabilităţilor obţinute?

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




```
    else
        return(FALSE)
    }))
```

Explicatii:

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:

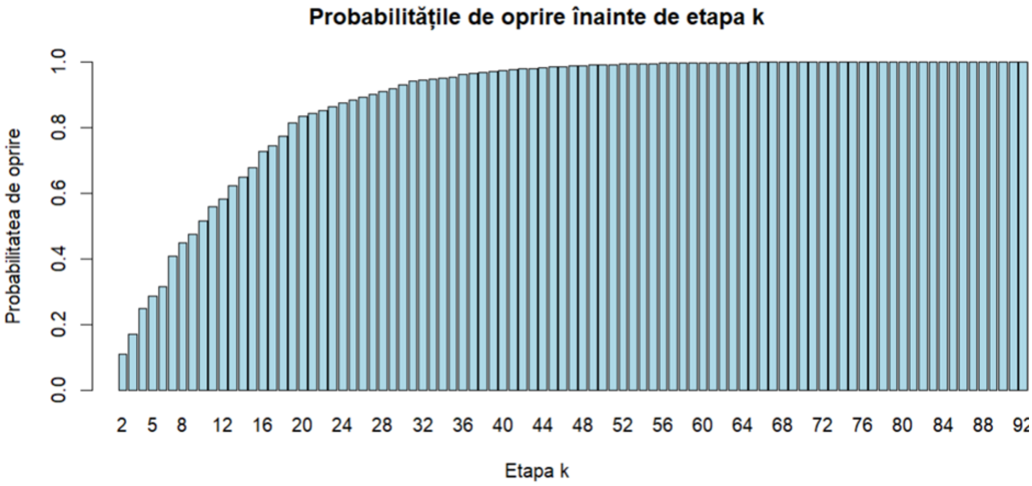
```
PstopK = 0.284293  pentru k=5
PstopK = 0.51482 pentru k=10
PstopK = 0.991332 pentru k=50
```

```
calculate_PStopBeforeK <- function(valoriT, lambda, alpha, n) {
# Calculam timpii cumulativi pentru fiecare etapa
cumulative_times <- cumsum(1 / lambda)

# Calculam probabilitatile pentru fiecare k
PStopBeforeK <- sapply(2:n, function(k) {
    mean(valoriT <= cumulative_times[k - 1])
            # Probabilitatea de oprire inainte de etapa k
})

return(PStopBeforeK)
}
PStopBeforeK <- calculate_PStopBeforeK(valoriT, lambda, alpha, n)

barplot(PStopBeforeK, names.arg = 2:n, col = "lightblue",
        main = "Probabilitatile de oprire inainte de etapa k",
        xlab = "Etapa k", ylab = "Probabilitatea de oprire",
        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

Descrierea Problemei

Scopul acestui proiect este de a construi o aplicaţie web interactivă folosind framework-ul **Shiny** din **R**, pentru a reprezenta cele 5 formulari ale repartitiei Negativ Binomiale, graficele animate şi situaţii concrete in care se pot aplica aceste formulări. Aplicaţia permite vizualizarea repartitiei Negativ Binomiale pentru diferite valori ale parametrilor r şi p.

Aspecte Teoretice

Pentru dezvoltarea aplicaţiei, am utilizat următoarele aspecte teoretice importante:

Distribuția Negativ Binomială: Este definită ca o variabilă aleatoare discretă $X \sim \mathrm{NB}(r, p)$, unde r reprezintă numărul de succese dorite, iar p este probabilitatea de succes pentru o singura incercare.

Repartiția Negativ Binomială poate fi definită în mai multe moduri, în funcție de context. Iată cele 5 formulări principale:

1. **k eșecuri, dat fiind r succese**
   Probabilitatea de a avea k eșecuri înainte de a obține r succese:
   $$f(k; r, p) \equiv \Pr(X = k) = \binom{k+r-1}{k} p^r (1-p)^k$$
   Alternativ:
   $$f(k; r, p) = \binom{k+r-1}{r-1} p^r (1-p)^k$$
   Suport: $k = 0, 1, 2, \ldots$
2. **n încercări, dat fiind r succese**
   Probabilitatea de a avea n încercări pentru a obține r succese:
   $$f(n; r, p) \equiv \Pr(X = n) = \binom{n-1}{r-1} p^r (1-p)^{n-r}$$
   Alternativ:
   $$f(n; r, p) = \binom{n-1}{n-r} p^r (1-p)^{n-r}$$
   Suport: $n = r, r+1, r+2, \ldots$
3. **n încercări, dat fiind r eșecuri**
   Probabilitatea de a avea n încercări pentru a obține r eșecuri:
   $$f(n; r, p) \equiv \Pr(X = n) = \binom{n-1}{r-1} p^{n-r} (1-p)^r$$
   Alternativ:
   $$f(n; r, p) = \binom{n-1}{n-r} p^{n-r} (1-p)^r$$
   Suport: $n = r, r+1, r+2, \ldots$
4. **k succese, dat fiind r eșecuri**
   Probabilitatea de a avea k succese înainte de a obține r eșecuri:
   $$f(k; r, p) \equiv \Pr(X = k) = \binom{k+r-1}{k} p^k (1-p)^r$$
   Alternativ:
   $$f(k; r, p) = \binom{k+r-1}{r-1} p^k (1-p)^r$$
   Suport: $k = 0, 1, 2, \ldots$

5. **k succese, dat fiind n încercări**
   Aceasta este de fapt repartiția binomială, nu cea negativ binomială:

$$f(k; n, p) \equiv \Pr(X = k) = \binom{n}{k} p^k (1-p)^{n-k}$$

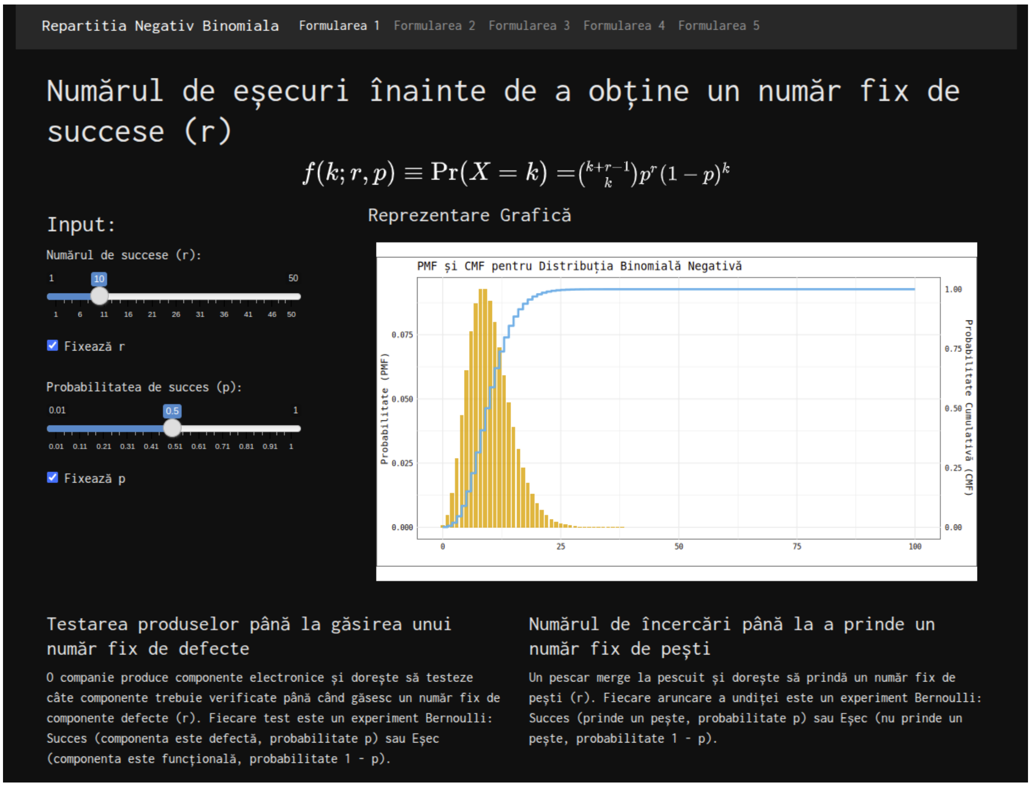
   Alternativ:

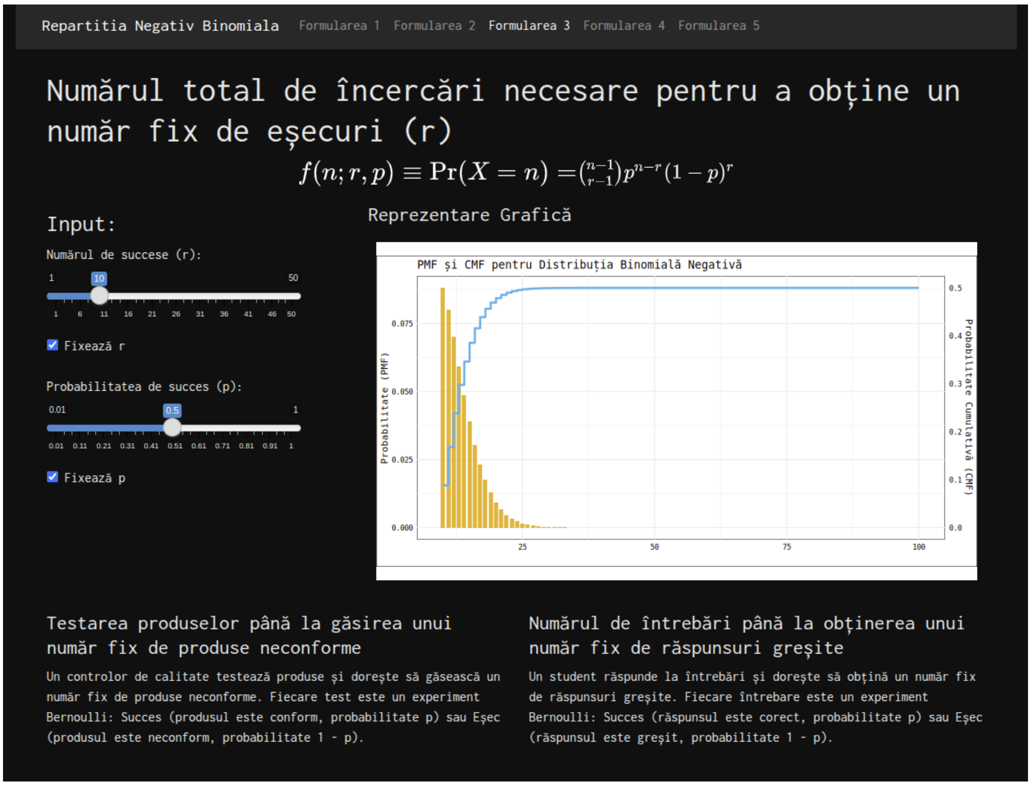
$$f(k; n, p) = \binom{n}{n-k} p^k (1-p)^{n-k}$$

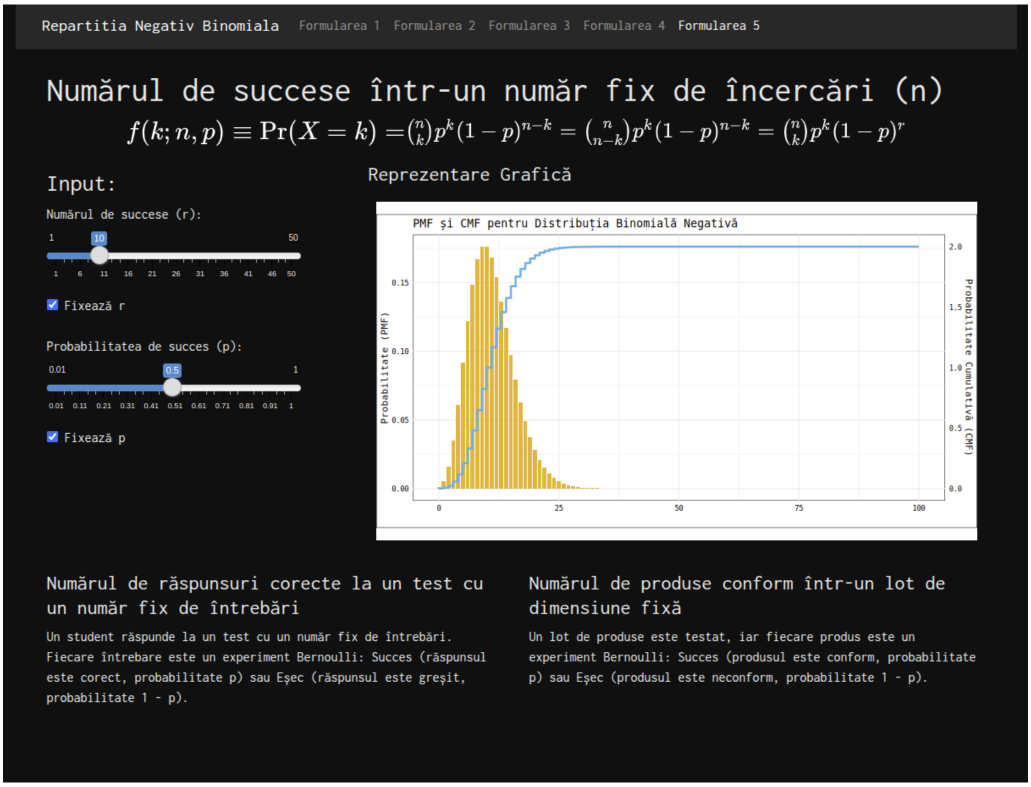
   Suport: $k = 0, 1, 2, \ldots, n$

Reprezentări Grafice

Aplicaţia oferă reprezentări grafice ale funcţiei de masă şi a funcţiilor de repartiţie. În cele ce urmează, vom ilustra cateva capturi de ecran din aplicaţie.







Cerinta III

Descrierea Problemei

Scopul acestui proiect este de a construi o aplicaţie web interactivă folosind framework-ul **Shiny** din **R**, pentru a reprezenta grafic funcţiile de repartiţie (CDF) ale unor variabile aleatoare definite conform cerinţei din enunţ. Aplicaţia permite vizualizarea CDF-urilor pentru distribuţii normale, exponenţiale, binomiale, Poisson.

Aspecte Teoretice

Pentru dezvoltarea aplicaţiei, am utilizat următoarele aspecte teoretice importante:

Aspecte Teoretice

Pentru dezvoltarea aplicaţiei, am utilizat următoarele aspecte teoretice importante:

- **Distribuţia Normală**: Este definită ca o variabilă aleatoare continuă $X \sim N(\mu, \sigma^2)$, unde μ reprezintă media, iar σ^2 varianţa.
  Formula funcției de repartiție cumulativă:

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

  Suma variabilelor normale: Dacă $X_1, X_2, ..., X_n \sim N(\mu, \sigma^2)$ sunt variabile aleatoare independente și identic distribuite, atunci suma lor:

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

  - Media sumei: $E(X) = n\mu$
  - Varianța sumei: $\text{Var}(X) = n\sigma^2$

  Suma pătratelor variabilelor normale: Considerând $X_i \sim N(\mu, \sigma^2)$, suma pătratelor acestora urmează o distribuție χ^2 non-centrală:

  $$X = \sum_{i=1}^{n} X_i^2 \sim \chi^2(n, \lambda)$$

  unde:

  - $\lambda = n \cdot \left(\frac{\mu}{\sigma}\right)^2$ este parametrul de non-centralitate.
  - Dacă $\mu = 0$, atunci $X \sim \chi^2(n)$, adică distribuția χ^2 standard cu n grade de libertate.

- **Distribuţia Exponenţială**: O variabilă aleatoare $X \sim \text{Exp}(\lambda)$ este folosită pentru modelarea timpilor de aşteptare între evenimente.
  Formula funcției de repartiție cumulativă:

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

  Suma variabilelor exponențiale: Dacă $X_1, X_2, ..., X_n \sim \text{Exp}(\lambda)$ sunt variabile aleatoare independente, atunci suma lor:

  $$X = \sum_{i=1}^{n} X_i \sim \text{Gamma}(n, \lambda)$$

  - Media sumei: $E(X) = \frac{n}{\lambda}$
  - Varianța sumei: $\text{Var}(X) = \frac{n}{\lambda^2}$

- **Distribuţia Poisson**: Reprezintă numărul de evenimente într-un interval fix de timp şi este notată $X \sim \text{Poisson}(\lambda)$.
  Formula funcției de repartiție cumulativă:

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

Suma variabilelor Poisson: Dacă $X_1, X_2, ..., X_n \sim \text{Poisson}(\lambda)$ sunt variabile aleatoare independente, atunci suma lor:

$$X = \sum_{i=1}^{n} X_i \sim \text{Poisson}(n\lambda)$$

- Media sumei: $E(X) = n\lambda$
- Varianța sumei: $\text{Var}(X) = n\lambda$

- **Distribuţia Binomială**: Este definită de parametrii n şi p şi descrie numărul de succese în n experimente independente.
  Formula funcției de repartiție cumulativă:

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

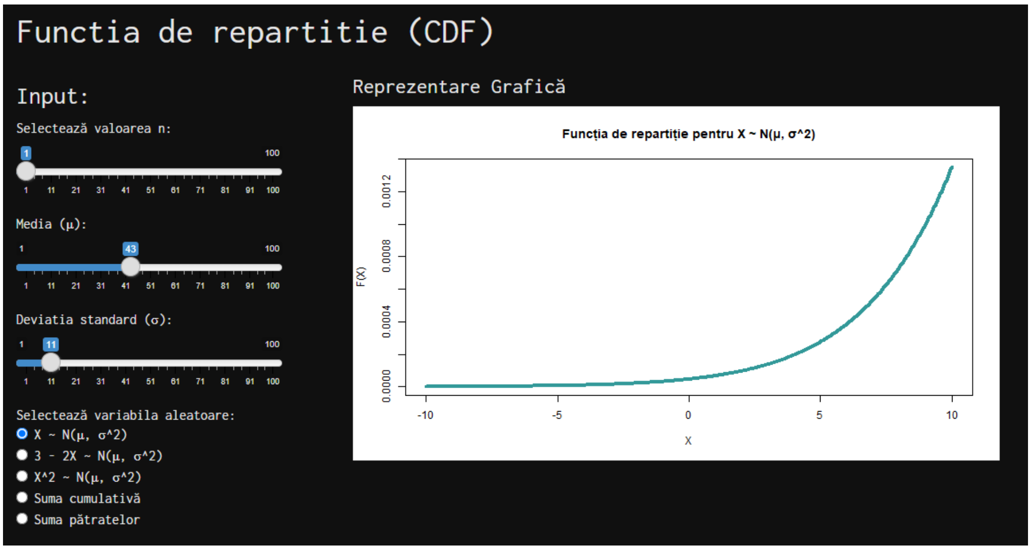
  Suma variabilelor binomiale: Dacă $X_1, X_2, ..., X_m \sim \text{Binomial}(n, p)$ sunt variabile aleatoare independente, atunci suma lor:

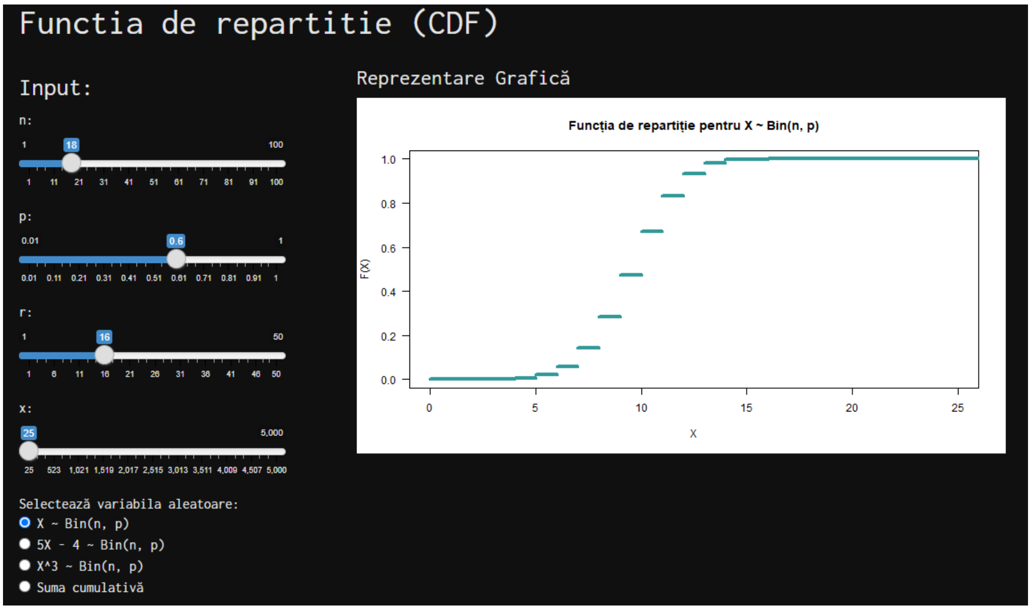
  $$X = \sum_{i=1}^{m} X_i \sim \text{Binomial}(m \cdot n, p)$$

  - Media sumei: $E(X) = m \cdot n \cdot p$
  - Varianța sumei: $\text{Var}(X) = m \cdot n \cdot p(1-p)$

Reprezentări Grafice

Aplicaţia oferă reprezentări grafice ale funcţiilor de repartiţie pentru fiecare dintre variabilele aleatoare definite în cerinţe. În cele ce urmează, vom ilustra câteva capturi de ecran din aplicaţie.





Pachete Software Folosite

Am folosit următoarele pachete software pentru implementarea proiectului:

- **shiny**: Crearea aplicaţiei interactive.
- **bslib**: Personalizarea interfeţei grafice a aplicaţiei folosind teme Bootstrap.
- **graphics**: Generarea reprezentărilor grafice ale funcţiilor de repartiţie.

Codul Aplicaţiei

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.

```r
library(shiny)
library(bslib)

source("utils.R")
source("ui.R")
source("server/server.R")

shinyApp(ui = ui, server = server)
```

ui.R

Interfața utilizatorului este creată folosind funcția `fluidPage`, care organizează interfața pe taburi, 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.

```r
source("server/server.R")

create_tab <- function(tab_title, title, distribution) {
  tabPanel(
    tab_title,
    div(
      class = "container",
      h1(title),
      div(
        class = "row",
        div(
          class = "col-4",
          tags$h3("Input:"),
          switch(
            distribution,
            BINOMIALA = create_binom_slider(),
            NORMALA_STANDARD = create_std_normal_slider(),
            NORMALA = create_normal_slider(),
            EXPONENTIALA = create_exponential_slider(),
            POISSON = create_pois_slider()
          )
        ),
        div(
          class = "col-8",
          h4("Reprezentare Grafica"),
          get_output_distribution(distribution)
        )
      )
    )
  )
}

```

```r
ui <- fluidPage(
  theme = bs_theme(
    version = 4,
    bg = "#101010",
    fg = "#FFF",
    primary = "#E69F00",
    base_font = font_google("Inconsolata")
  ),
  navbarPage(
    "CDF",
    create_tab(
      "Normala Standard",
      "Functia de repartitie (CDF)",
      NORMALA_STANDARD
    ),
    create_tab(
      "Normala",
      "Functia de repartitie (CDF)",
      NORMALA
    ),
    create_tab(
      "Exponentiala",
      "Functia de repartitie (CDF)",
      EXPONENTIALA
    ),
    create_tab(
      "Binomiala",
      "Functia de repartitie (CDF)",
      BINOMIALA
    ),
    create_tab(
      "Poisson",
      "Functia de repartitie (CDF)",
      POISSON
    )
  )
)
```

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.

```r
source("server/std_normal.R")
source("server/normal.R")
source("server/exponential.R")
source("server/binom.R")
source("server/poisson.R")

server <- function(input, output, session) {
  std_normal_server(input, output, session)
  normal_server(input, output, session)
  exponential_server(input, output, session)
  binom_server(input, output, session)
  pois_server(input, output, session)
}
```

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 redirecționare a graficelor.

```
NORMALA_STANDARD <- "NORMALA_STANDARD"
NORMALA <- "NORMALA"
EXPONENTIALA <- "EXPONENTIALA"
BINOMIALA <- "BINOMIALA"
POISSON <- "POISSON"

get_output_distribution <- function(distribution) {
  switch(
    distribution,
    NORMALA_STANDARD = plotOutput("std_normal_plot"),
    NORMALA = plotOutput("normal_plot"),
    EXPONENTIALA = plotOutput("exponential_plot"),
    BINOMIALA = plotOutput("binom_plot"),
    POISSON = plotOutput("pois_plot")
  )
}
```

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 repartitie cumulativă. Iată principalele tipuri de grafice generate:

- **Graficul funcției de repartitie 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.

```
create_normal_slider <- function() {
  tagList(
    sliderInput(
      inputId = "normal_n",
      label = "Selecteaza valoarea n:",
      min = 1,
      max = 100,
      value = 10,
      step = 1
    ),
    sliderInput(
      inputId = "normal_mu",
      label = "Media (\u03BC):",
      min = 1,
      max = 100,
      value = 10,
      step = 1
    ),
    sliderInput(
      inputId = "normal_sigma",
      label = "Deviatia standard (\u03C3):",
      min = 1,
      max = 100,
```

```
      value = 10,
      step = 1
    ),
    radioButtons(
      inputId = "normal_var",
      label = "Selecteaza variabila aleatoare:",
      choices = list(
        "X ~ N(\u03BC, \u03C3^2)" = "var1",
        "3 - 2X ~ N(\u03BC, \u03C3^2)" = "var2",
        "X^2 ~ N(\u03BC, \u03C3^2)" = "var3",
        "Suma cumulativa" = "var4",
        "Suma patratelor" = "var5"
      ),
      selected = "var1"
    )
  )
}

normal_server <- function(input, output, session) {
  output$normal_plot <- renderPlot({
    var <- input$normal_var

    if (var == "var1") {
      x <- seq(-10, 10, length.out = 500)
      cdf_values <- pnorm(x, mean = input$normal_mu, sd = input$normal_sigma)
      plot(
        x, cdf_values,
        type = "l",
        lwd = 4,
        col = "#339999",
        xlab = "X",
        ylab = "F(X)",
        main = "Functia de repartitie pentru X ~ N(\u03BC, \u03C3^2)"
      )
    } else if (var == "var2") {
      x <- seq(-10, 10, length.out = 500)
      transformed_x <- 3 - 2 * x
      cdf_values <- pnorm(transformed_x, mean = input$normal_mu, sd = input$normal_sigma)
      plot(
        x, cdf_values,
        type = "l",
        lwd = 4,
        col = "#FF6666",
        xlab = "3 - 2X",
        ylab = "F(3 - 2X)",
        main = "Functia de repartitie pentru 3 - 2X ~ N(\u03BC, \u03C3^2)"
      )
    } else if (var == "var3") {
      x <- seq(-10, 10, length.out = 500)
      transformed_x <- x^2
      cdf_values <- pnorm(transformed_x, mean = input$normal_mu, sd = input$normal_sigma)
      plot(
        x, cdf_values,
        type = "l",
        lwd = 4,
        col = "#3399FF",
        xlab = "X^2",
        ylab = "F(X^2)",
        main = "Functia de repartitie pentru X^2 ~ N(\u03BC, \u03C3^2)"
      )
    } else if (var == "var4") {
      n <- input$normal_n
      X <- rnorm(n, mean = input$normal_mu, sd = input$normal_sigma)
      S_n <- cumsum(X)
      plot(
        1:n, S_n,
```

```
          type = "l",
          lwd = 2,
          col = "#FF9900",
          xlab = "n",
          ylab = "\u2211 X_i",
          main = "Suma cumulativa a variabilelor aleatoare X_i"
        )
      } else if (var == "var5") {
        n <- input$normal_n
        X <- rnorm(n, mean = input$normal_mu, sd = input$normal_sigma)
        S_n2 <- cumsum(X^2)
        plot(
          1:n, S_n2,
          type = "l",
          lwd = 2,
          col = "#33CC33",
          xlab = "n",
          ylab = "\u2211 X_i^2",
          main = "Suma cumulativa a patratelor variabilelor X_i"
        )
      }
    })
  }
```

Dificultăţi 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 repartitie 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 repartitie 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 repartitie 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.