Universitatea din București
Facultatea de Matematică și Informatică

Anul universitar 2023-2024 - Probabilități și statistică

PROIECT PROBABILITĂȚI ȘI STATISTICĂ

Dobromirescu Mihaela - 242 (Lider)

Popa Silviu Andrei - 242

Negraru Celestin - 242

Firulescu Lawrence - 242

I.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. Construiți un algoritm în R care simulează 10^6 valori pentru variabila aleatoare(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)?

Descrierea Algoritmului:

1. In algoritmul nostru am definit varibilele importante și anume: nr_sim - numarul de simulari, n - numarul de etape, lambda - ratele distributiilor exponentiale pentru fiecare etapa și alpha - probabilitatile ca procesul sa continue la etapa urmatoare
2. Definim functia simulate_T() care returneaza timpul total petrecut in etapele prin care a trecut și ultima etapa inainte ca activitatea sa fie atinsa.
3. Initializam T_total și last_stage cu 0.
4. Iteram prin toate etapele i.
5. Generam un timp aleatoriu folosind rexp(1, lambda[i]) care ne da timpul petrecut in etapa i.
6. Adaugam acest timp la T_total.
7. Daca runif(1) > alpha[i] atunci activitatea se opreste, altfel continuam.
8. Returnam timpul total și ultima etapa la care ne-am oprit.
9. Acest proces este apoi repetat de 10^6 ori.
10. Dupa simulare, estimam valoarea medie a timpului total adica E(T).

Rezultat:

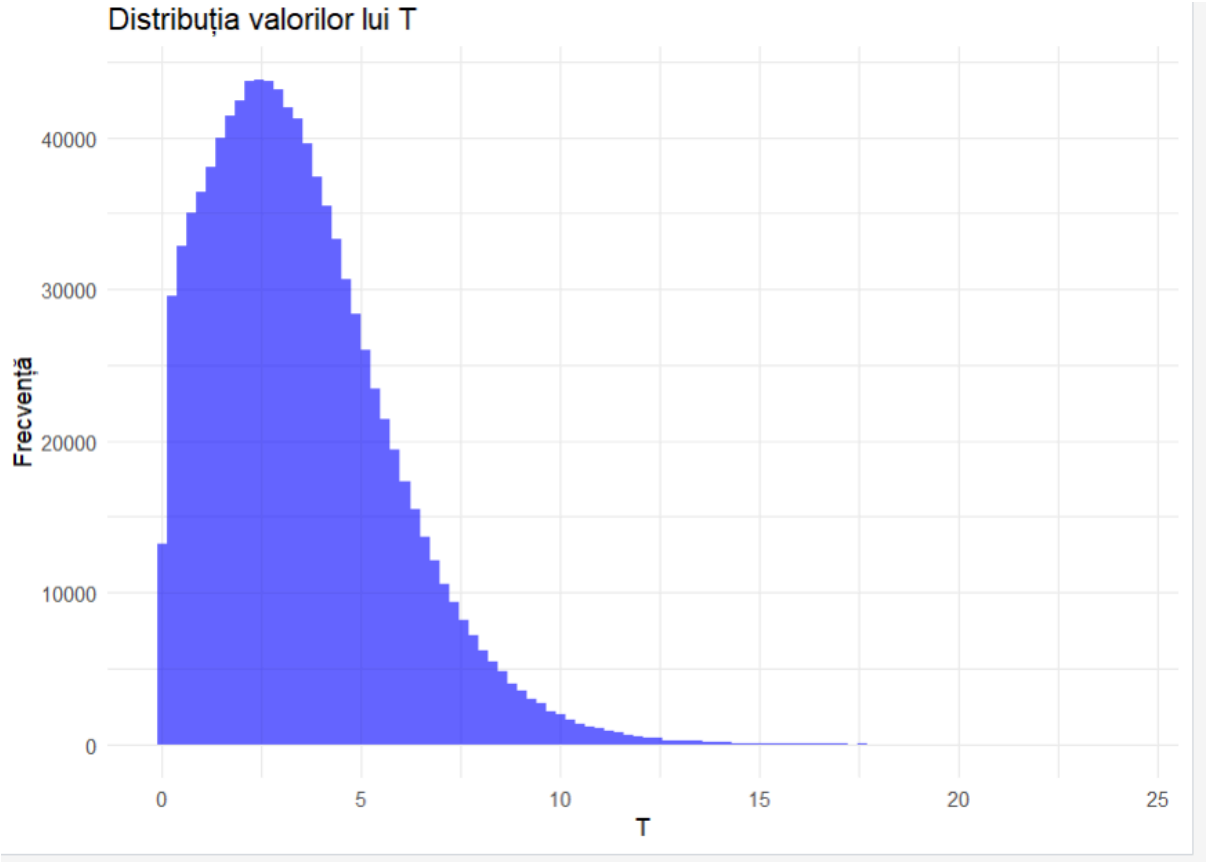
```
Valoarea estimată prin simulare: 3.484113
```

Cod:

```
library(ggplot2)
library(dplyr)

num_sim <- 10^6 # nr de simulari


n <- 5  # numarul de etape
lambda <- c(1, 1.2, 0.8, 1.5, 1)  # Rratele distributiilor exponentiale
pentru fiecare etapa
alpha <- c(0.9, 0.8, 0.85, 0.7, 0.6)  # probabilitatile de a continua
la pasul urmator

#functia pentru a obtine timpul total si ultima activitate inainte ca
activitatea sa se opreasca
simulate_T <- function() {
  T_total <- 0  # se initializeaza cu 0, adica activitatea nu a inceput
inca
  last_stage <- 0
  for (i in 1:n) {  #iteram prin toate etapele
    T_total <- T_total + rexp(1, lambda[i])  # se genereaza un timp
exponential care se adauga la T_total
    last_stage <- i # ultima activitate devine i
    if (runif(1) > alpha[i]) break  # verificam daca activitatea se
opreste
  }
  return(c(T_total, last_stage))
}

# simulam acest proces de 10^6 ori
sim_results <- replicate(num_sim, simulate_T())
T_values <- sim_results[1, ]
last_stages <- sim_results[2, ]

# 1) estimam valoarea medie a timpului total si il afisam
E_T_simulated <- mean(T_values)
cat("Estimarea lui E(T):", E_T_simulated, "\n")

# reprezentarea grafica a distributiei lui T
ggplot(data.frame(T_values), aes(x = T_values)) +
  geom_histogram(bins = 100, fill = "blue", alpha = 0.6) +
```

```
  labs(title = "Distributia valorilor lui T",
       x = "T",
       y = "Frecventa") +
  theme_minimal()
```

Reprezentarea grafica pentru distributia valorilor lui T:



Din reprezentarea grafica putem observa urmatoarele lucruri despre repartitia lui T:

1. Este asimetrica spre dreapta, lucru de asteptat pentru o suma de variabile exponentiale.
2. Exista o coada lunga la dreapta ceea ce inseamna ca, desi majoritatea activitatilor dureaza putin, exista cazuri rare in care timpul de finalizare este foarte mare.

2. Calculați valoarea exactă a lui E(T) și comparați cu valoarea obținută prin simulare.

Pentru a calcula valoarea exactă a lui E(T), folosim o formulă bazată pe distribuțiile exponențiale și probabilitățile de tranziție. Formula generala este:

$$E(T) = \sum_{i=1}^{n} \frac{1}{\lambda_i} \prod_{j=1}^{i-1} \alpha_j$$

unde:

- λ_i este rata distribuției exponențiale pentru etapa i,
- α_j este probabilitatea de a continua la etapa j,
- $\prod_{j=1}^{i-1} \alpha_j$ reprezintă probabilitatea ca procesul să fi ajuns până la etapa i fără să se fi oprit anterior.

Cod:

```
E_T_exact <- sum(sapply(1:n, function(i) {

  (1 / lambda[i]) * ifelse(i == 1, 1, prod(alpha[1:(i-1)]))

}))
```

Rezultat:

```
Valoarea exactă a E(T): 3.4864
```

Interpretarea rezultatelor:

Valoarea estimata prin simulare(3,484113) este foarte apropiata de valoarea exacta calculata teoretic(3.4864). Diferenta absoluta este de doar 0.002347 ceea ce indica faptul ca cu un numar suficient de mare de simulari(10^6), estimarea lui E(T) prin simulare este foarte precisa.

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

La primul subpunct am rulat procesul de 10^6 ori și am înregistrat ultima etapa atinsa pentru fiecare executie. Probabilitatea ca persoana A sa finalizeze activitatea este raportul dintre dintre numarul de simulari care au ajuns la etapa n și numarul de simulari.

Cod:

```
p_finalizare_simulat <- sum(last_stages == n) / num_sim

cat("Probabilitatea estimată prin simulare:", p_finalizare_simulat,
"\n")
```

Rezultat:

```
Probabilitatea estimată prin simulare: 0.427969
```

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

Folosim simularile efectuate la primul subpunct, unde am generat un numar mare de valori pentru T și exprimam probabilitatea ca raportul dintre numarul de simulari care au avut un timp de finalizare mai mic sau egal cu σ și numarul total de simulari

Cod:

```
# 4) Probabilitatea ca T <= σ

sigma <- 5

p_T_sigma <- sum(T_values <= sigma) / num_sim #T_values este vectorul
care contine timpii simulati ai activitalitor finalizate

cat("Probabilitatea ca T ≤", sigma, ":", p_T_sigma, "\n")
```

Rezultat:

```
Probabilitatea ca T ≤ 5 : 0.769776
```

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?

Aceasta cerinta presupune sa aflam timpul minim și timpul maxim in care persoana A finalizeaza activitatea, adica cel mai rapid timp in care activitatea poate fi finalizata, respectiv cel mai lent.

Cod:

```
T_min <- min(T_values)

T_max <- max(T_values)

cat("Timpul minim de finalizare:", T_min, "\n")

cat("Timpul maxim de finalizare:", T_max, "\n")

ggplot(data.frame(T_values), aes(x = T_values)) +

  geom_histogram(bins = 100, fill = "blue", alpha = 0.6) +

  labs(title = "Distribuția timpilor de finalizare",

       x = "Timp de finalizare",

       y = "Frecvență") +

  theme_minimal()
```

Rezultat:

```
Timpul minim de finalizare: 8.401105e-06
```

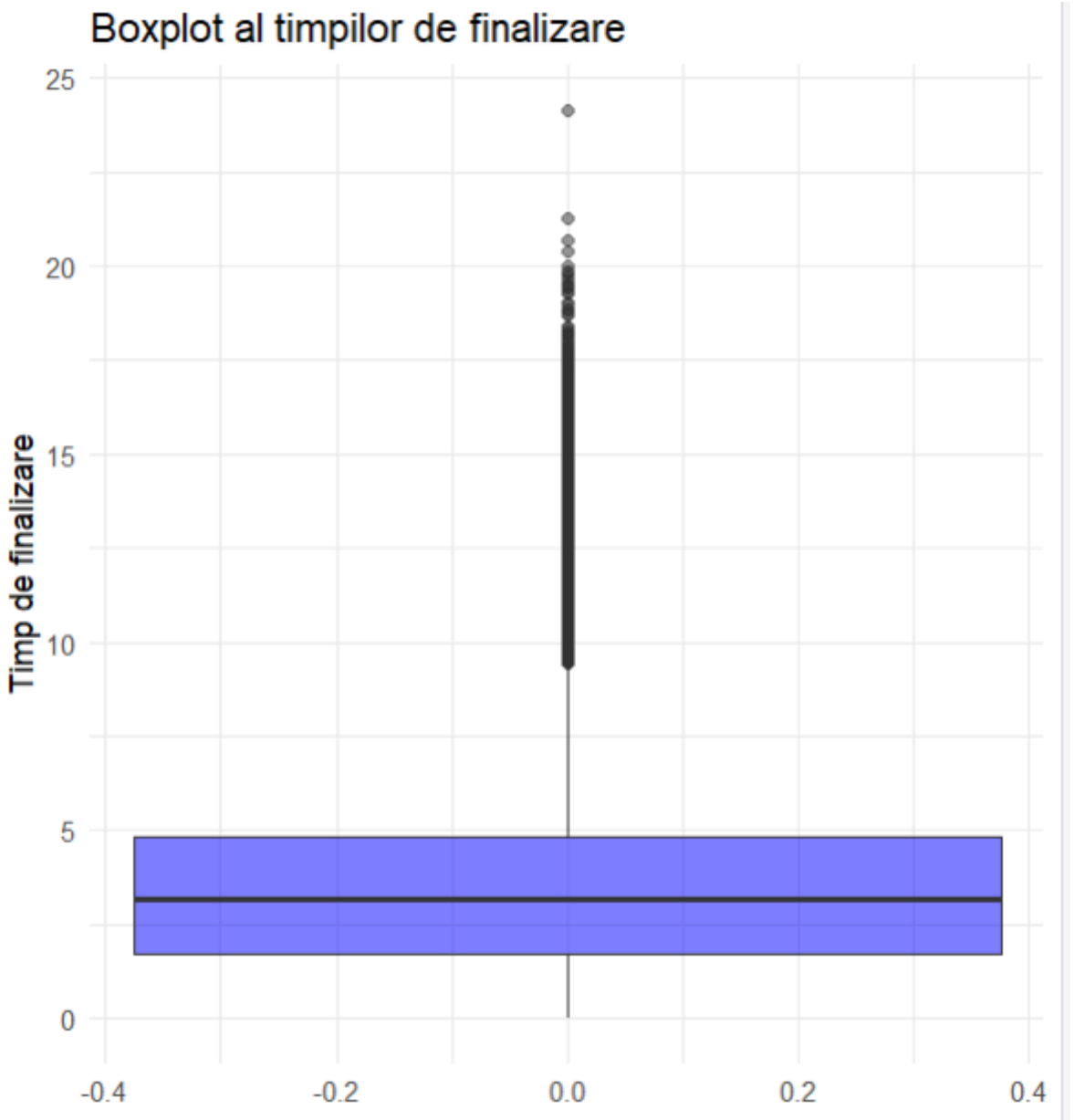
```
Timpul maxim de finalizare: 24.15052
```

Interpretare rezultate:

Timpul minim de finalizare este foarte aproape de 0, ceea ce indica cazuri in care activitatea se finalizeaza foarte rapid. Acest lucru se intampla cand persoana A se opreste devreme, fara a parcurge multe etape.

Timpul maxim de finalizare este mai mare decat media, ceea ce confirma exista unor cazuri extreme in care activitatea dureaza semnificativ mai mult.

Histograma timpilor de finalizare:



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

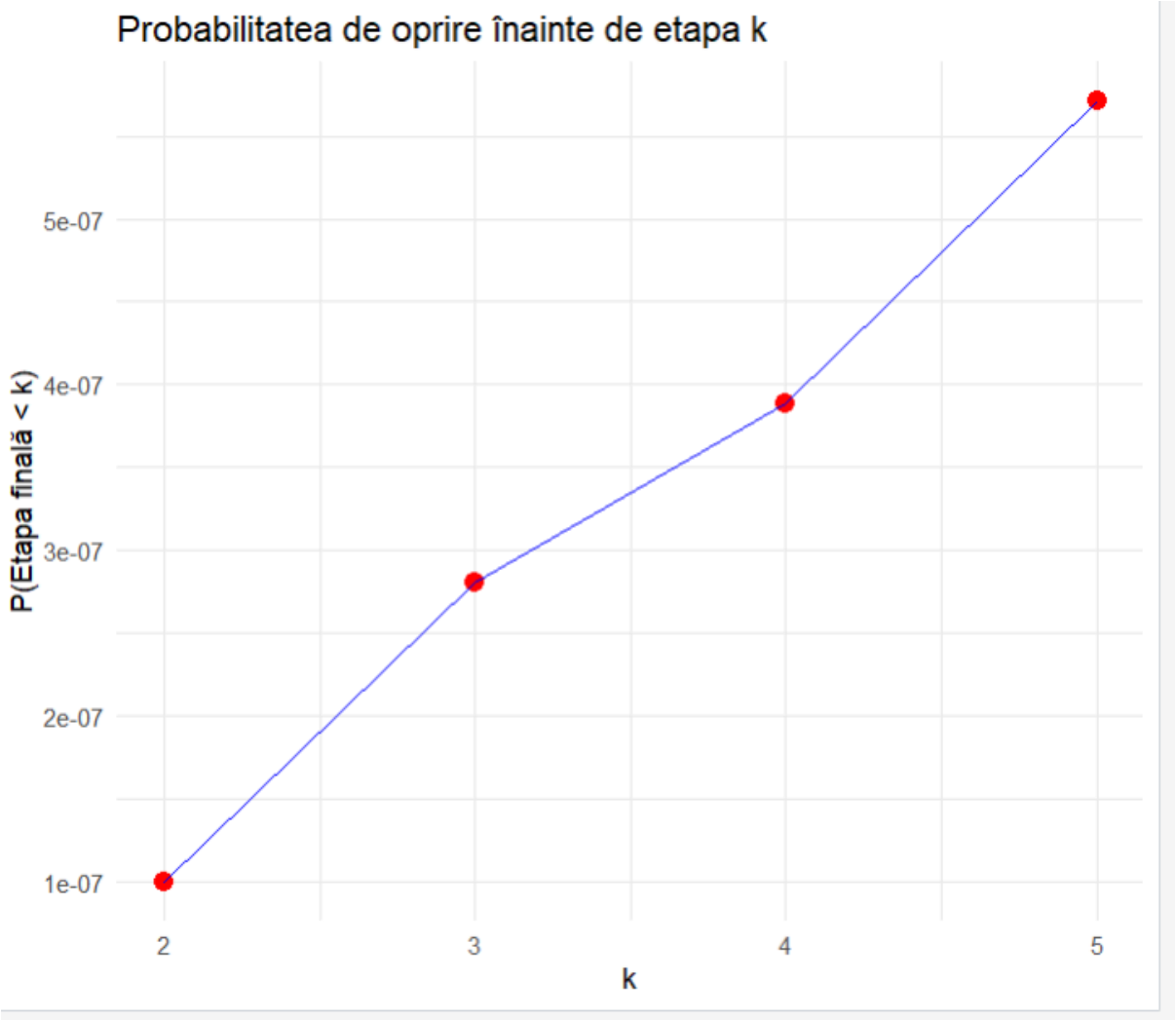
Aproximarea probabilitatii ca persoana a sa se opreasca inainte de etapa k, adica sa nu ajunga niciodata la aceasta etapa este P(last_stage < k)

Cod:

```
# 6) Probabilitatea de oprire înainte de etapa k

k_values <- 1:n

prob_k <- sapply(k_values, function(k) {

  sum(last_stages < k) / num_sim

})



df_prob <- data.frame(k = k_values, Probabilitate = prob_stop_before_k)



ggplot(df_prob, aes(x = k, y = Probabilitate)) +

  geom_point(color = "red", size = 3) +

  geom_line(color = "blue") +

  labs(title = "Probabilitatea de oprire inainte de etapa k",

       x = "k",

       y = "P(Etapa finală < k)") +

  theme_minimal()
```

Rezultat:

```
  k Probabilitate
1 2   1.00067e-07
2 3   2.80690e-07
3 4   3.88280e-07
4 5   5.71571e-07
```



Pachete Software Utilizate

- **R**: limbajul principal de programare
- **ggplot2**: pentru vizualizarea grafică
- **dplyr**: pentru manipularea datelor

II. Construiți o aplicație Shiny(Shiny - Welcome to Shiny) în care să prezentați cele 5 formulări alternative pentru repartiția Negativ Binomială(Negative binomial distribution - Wikipedia), cu toate parametrizările cunoscute și cu repartițiile înrudite, împreună cu exemple de utilizare a repartiției. În realizarea temei se urmăresc în mod prioritar următoarele aspecte:
1) Reprezentarea grafică a funcției de masă și respectiv a funcției de repartiție cu diferiți parametri
2) Construirea unei animații care să pună în evidență schimbarea formei funcțiilor de la 1) pe măsură ce parametrii se modifică(vezi Wikipedia)
3) Ilustrarea unor exemple de aplicații în care repartiția Negativ Binomială este relevantă

- O scurtă introducere a conceptelor teoretice

Repartiția Negativ Binomială reprezintă o distribuție de probabilitate discretă utilizată frecvent pentru a modela numărul de eșecuri într-o serie de experimente Bernoulli înainte de a obține un anumit număr dorit de succese, fiecare având aceeași probabilitate. Aceasta are numeroase aplicații în domenii precum biostatistica, econometria, marketing-ul și ingineria fiabilității.

Distribuția negativ binomială are mai multe formulări alternative, fiecare adaptată la diferite interpretări și aplicații. Iată câteva dintre cele mai utilizate:

1. **Numărul de eșecuri înainte de r succese**

Variabila aleatoare X reprezintă numărul de eșecuri care apar înainte de a obține **exact** r succese într-un proces Bernoulli.

Formula pentru PMF este

$$\Pr(X = k) = \binom{k + r - 1}{k} (1 - p)^k p^r$$

Iar formula pentru CDF este

$$F_X(x) = \mathrm{P}(X \leq x) \textbf{(Eq.1)}$$

, altfel spus, suma tuturor probabilităților, unde variabila aleatoare X sa fie mai mică sau egală cu k, numărul de eșecuri.

2. **Numărul total de încercări pentru a obține r succese**

Variabila aleatoare X′ reprezintă **numărul total de încercări** necesare pentru a obține **exact** r succese (include atât succesele, cât și eșecurile, spre deosebire de prima abordare).

Relația dintre X′ și X este: X′=X+r

De data aceasta, se considera n încercări totale pentru a obține tot r succese

2	*n* trials, given *r* successes	$f(n;r,p) \equiv \Pr(X=n) =$	$\binom{n-1}{r-1}p^r(1-p)^{n-r}$ [5][11][12][13][14]

Această bucată din documentație prezintă o aplicație web Shiny care ilustrează grafic atât funcția de masă (Probability Mass Function - PMF), cât și funcția de repartiție cumulativă (Cumulative Distribution Function - CDF), în toate cele 5 formulări alternative pentru repartiția Negativ Binomială, conform Wikipedia.

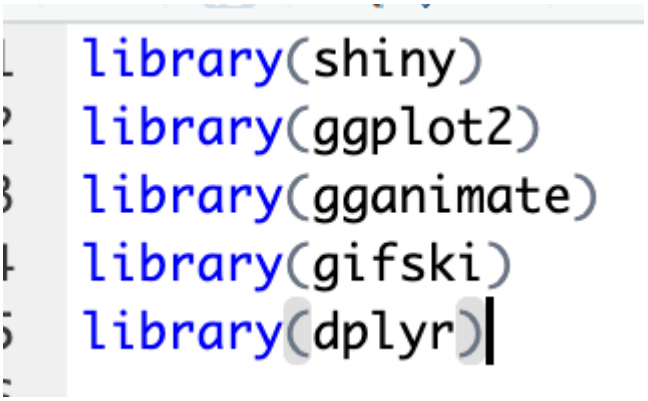
- Pachete și programe software folosite în realizarea aplicației

Ca și limbaj de programare am utilizat R în mediul de dezvoltare R Studio, împreuna cu pachetul Shiny pentru a realiza interfața grafică a aplicației, dar și pachete precum Ggplot2, Gganimate și Gifski pentru plotarea și animarea graficelor.

- Prezentarea codului și a interfeței utilizatorului

În cele ce urmează, vom prezenta părțile componente ale aplicației , dar și graficele rezultate în urma rulării programului app.R !

Mai jos avem directivele de includere a pachetelor folosite pentru realizarea aplicației



```
library(shiny)
library(ggplot2)
library(gganimate)
library(gifski)
library(dplyr)
```

Apoi avem componenta UI a aplicației ce reprezintă meniul interactiv de unde utilizatorul își poate modifica în timp real parametrii, dar și opțiuni de selecție a tipului de reprezentare și a graficului funcției dorite (PMF sau CDF)

```
ui <- fluidPage(
  titlePanel("Distribuția Negativ Binomială"),
  sidebarLayout(
    sidebarPanel(
      sliderInput("size", "Parametrul size (numărul de succese):", min = 1, max = 20, value = 5),
      sliderInput("prob", "Probabilitatea succesului per încercare:", min = 0.1, max = 1, value = 0.5, step = 0.05),
      sliderInput("x_max", "Limita superioară pentru X:", min = 10, max = 100, value = 50),
      selectInput("distType", "Formulare a distribuției:",
                  choices = c("Număr eșecuri înainte de r succese" = "failures_before_r",
                              "Număr total încercări pentru r succese" = "total_trials",
                              "Sumă de variabile geometrice" = "sum_geom",
                              "Model Poisson-Gamma" = "poisson_gamma",
                              "Recompense Bernoulli" = "bernoulli_rewards")),
      selectInput("plotType", "Tip de reprezentare:",
                  choices = c("Funcția de Masă a Probabilității (PMF)" = "pmf",
                              "Funcția de Repartiție Cumulativă (CDF)" = "cdf")),
      actionButton("animate", "Generează animație")
    ),
    mainPanel(
      plotOutput("distPlot"),
      imageOutput("animation")
    )
  )
)
```

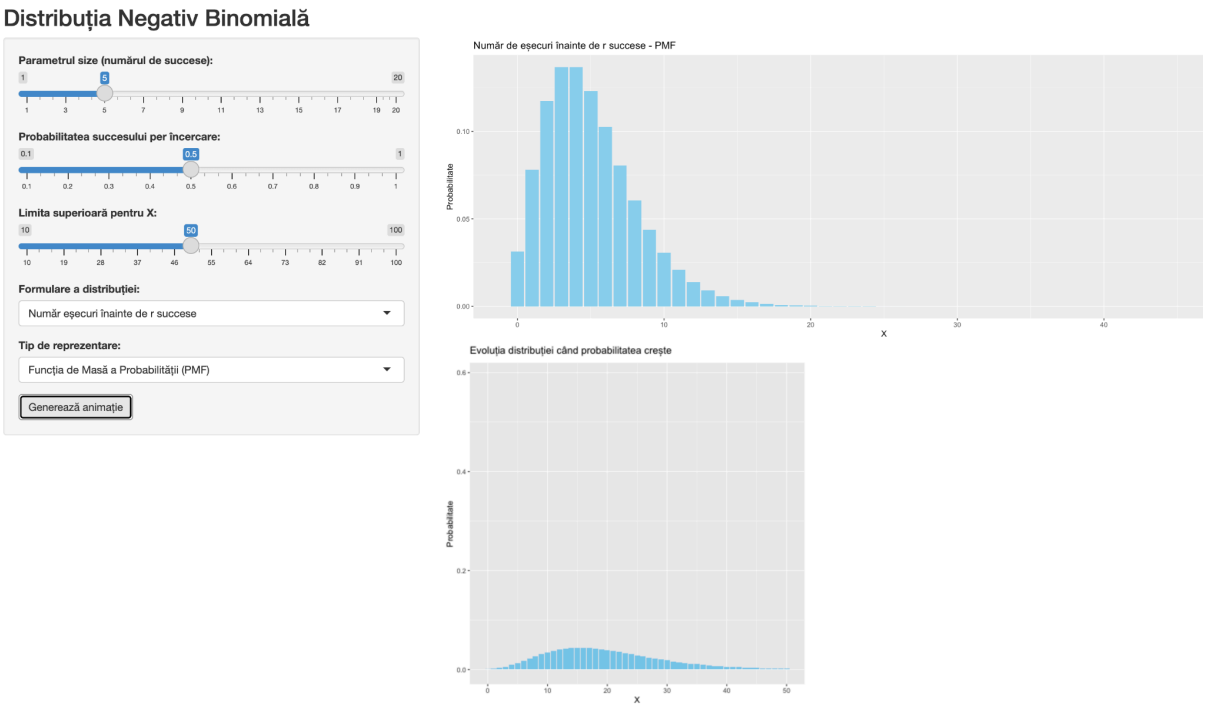
Urmează componenta server, responsabilă de gestionarea logicii de business dintre utilizator și interfața grafică

```r
server <- function(input, output) {

  output$distPlot <- renderPlot({
    x_vals <- 0:input$x_max
    df <- data.frame(x = x_vals)

    if (input$distType == "failures_before_r") {
      df$pmf <- dnbinom(x_vals, size = input$size, prob = input$prob)
      df$cdf <- pnbinom(x_vals, size = input$size, prob = input$prob)
      title <- "Număr de eșecuri înainte de r succese"

    } else if (input$distType == "total_trials") {
      df$pmf <- dnbinom(x_vals - input$size, size = input$size, prob = input$prob)
      df$cdf <- pnbinom(x_vals - input$size, size = input$size, prob = input$prob)
      title <- "Număr total de încercări pentru r succese"

    } else if (input$distType == "sum_geom") {
      simulated <- replicate(10000, sum(rgeom(input$size, input$prob)))
      df <- data.frame(x = simulated)
      title <- "Sumă de variabile geometrice"

      # La distribuțiile care nu au pmf sau cdf, putem să nu le includem
      df$pmf <- NULL
      df$cdf <- NULL

    } else if (input$distType == "poisson_gamma") {
      lambda_vals <- rgamma(10000, shape = input$size, scale = (1 - input$prob) / input$prob)
      poisson_samples <- rpois(10000, lambda = lambda_vals)
      df <- data.frame(x = poisson_samples)
      title <- "Model Poisson-Gamma"

      # La distribuțiile care nu au pmf sau cdf, putem să nu le includem
      df$pmf <- NULL
      df$cdf <- NULL

    } else if (input$distType == "bernoulli_rewards") {
      rewards <- rbinom(10000, size = input$size, prob = input$prob)
      df <- data.frame(x = rewards)
      title <- "Recompense în proces Bernoulli"

      # La distribuțiile care nu au pmf sau cdf, putem să nu le includem
      df$pmf <- NULL
      df$cdf <- NULL
    }

    # Verificăm dacă pmf sau cdf există înainte de a le plota
    if (input$plotType == "pmf" && !is.null(df$pmf)) {
      ggplot(df, aes(x, pmf)) +
        geom_col(fill = "skyblue") +
        labs(title = paste(title, "- PMF"), x = "X", y = "Probabilitate")

    } else if (input$plotType == "cdf" && !is.null(df$cdf)) {
      ggplot(df, aes(x, cdf)) +
        geom_line(color = "blue") +
        geom_point(color = "red") +
```

În cele din urmă, avem animația ce evidențiază cum se schimbă forma funcțiilor de distribuție pe măsură ce parametrul de probabilitate variază

```
output$animation <- renderImage({
  req(input$animate)

  # Definirea valorilor probabilității
  probs <- seq(0.1, 0.9, by = 0.1)

  # Crearea datelor pentru animație
  animation_data <- expand.grid(x = 0:input$x_max, prob = probs)

  # Calcularea pmf pentru fiecare combinație de x și prob
  animation_data$pmf <- mapply(function(x, prob) dnbinom(x, size = input$size, prob = prob),
                               animation_data$x, animation_data$prob)

  # Verificarea că pmf nu conține valori NA sau NULL
  animation_data <- animation_data[!is.na(animation_data$pmf), ]

  # Crearea animației folosind ggplot
  frames <- ggplot(animation_data, aes(x = x, y = pmf, frame = prob)) +
    geom_col(fill = "skyblue") +
    transition_states(prob, transition_length = 2, state_length = 1) +
    labs(title = "Evoluția distribuției când probabilitatea crește",
         x = "X", y = "Probabilitate")

  # Salvarea animației ca GIF
  gif_path <- tempfile(fileext = ".gif")
  anim_save(gif_path, frames, renderer = gifski_renderer())

  # Returnarea GIF-ului generat
  list(src = gif_path, contentType = 'image/gif')
}, deleteFile = TRUE)
```

Aceasta este interfața utilizatorului când acesta pornește aplicația



În cele din urmă, mai jos se află un demo al aplicației!

[Demo]

III. Construiți o aplicație Shiny în care să reprezentați grafic funcțiile de repartiție pentru următoarele variabile aleatoare:

$X,\quad 3-2X,\quad X^2,\quad \sum_{i=1}^{n} X_i,\quad \sum_{i=1}^{n} X_i^2,$ unde $X, X_1, X_2, \ldots, X_n \; i.i.d. \sim \mathcal{N}(0,1),\quad n \in \mathbb{N}$ fixat

2) $X,\quad 3-2X,\quad X^2,\quad \sum_{i=1}^{n} X_i,\quad \sum_{i=1}^{n} X_i^2,$ unde $X, X_1, X_2, \ldots, X_n \; i.i.d. \sim \mathcal{N}(\mu, \sigma^2),\quad \mu \in \mathbb{R}, \sigma > 0$

$X,\quad 2+5X,\quad X^2,\quad \sum_{i=1}^{n} X_i,$ unde $X, X_1, X_2, \ldots, X_n \; i.i.d. \sim \text{Exp}(\lambda),\quad \lambda > 0,\quad n \in \mathbb{N}$ fixat

4) $X,\quad 3X-2,\quad X^2,\quad \sum_{i=1}^{n} X_i,$ unde $X, X_1, X_2, \ldots, X_n \; i.i.d. \sim \text{Pois}(\lambda),\quad \lambda > 0,\quad n \in \mathbb{N}$ fixat

5) $X,\quad 5X-4,\quad X^3,\quad \sum_{i=1}^{n} X_i,$ unde $X, X_1, X_2, \ldots, X_n \; i.i.d. \sim \text{Binom}(r,p),\quad r \in \mathbb{N},\quad p \in (0,1), n \in \mathbb{N}$ fixat

Această aplicație Shiny a fost realizată pentru a permite vizualizarea funcțiilor de repartitie pentru variabile aleatoare extrase din diferite distribuții statistice.

Utilizatorul poate selecta distribuția dorită, seta parametrii corespunzători și vizualiza funcțiile de repartiție aferente.

Pachete Software Utilizate

- **R**: limbajul principal de programare
- **Shiny**: pentru interfața interactivă
- **Base R**: pentru generarea distribuțiilor (rnorm(), rexp(), rpois(), rbinom())

Aspecte Teoretice

Pentru realizarea acestei aplicații, s-au utilizat urmatoarele concepte:

- Funcțiile de repartiție cumulative $F_X(x) = P(X \leq x)$, care descriu probabilitatea ca o variabilă aleatoare să fie mai mică sau egală cu o anumită valoare .
- Distribuțiile de probabilitate utilizate:
  - Normală
  - Exponențială
  - Poisson
  - Binomială

ui.R

```
library(shiny)



ui <- fluidPage(

  titlePanel("Vizualizare Functii de Repartitie"),

  sidebarLayout(

    sidebarPanel(

      selectInput("distribution", "Alege distributia:",

                  choices = c("Normala (0,1)", "Normala (mu, sigma^2)",
"Exponentiala", "Poisson", "Binomiala")),
```

```
      numericInput("param1", "Parametru 1:", value = 1, min = 0.1),
      numericInput("param2", "Parametru 2:", value = 1, min = 0.1),
      numericInput("n", "Numar de esantioane:", value = 1000, min =
100)
    ),
    mainPanel(
      plotOutput("distPlot")
    )
  )
)
```

server.R

```
server <- function(input, output) {
  output$distPlot <- renderPlot({
    set.seed(123)
    n <- input$n



    base_data <- switch(input$distribution,
                        "Normala (0,1)" = rnorm(n, mean = 0, sd = 1),
                        "Normala (mu, sigma^2)" = rnorm(n, mean =
input$param1, sd = sqrt(input$param2)),
                        "Exponentiala" = rexp(n, rate = input$param1),
                        "Poisson" = rpois(n, lambda = input$param1),
                        "Binomiala" = rbinom(n, size =
as.integer(input$param1), prob = input$param2)
```

```
    )




    transform_data <- switch(input$distribution,

                             "Normala (0,1)" = list("X" = base_data,
"3-2X" = 3 - 2 * base_data, "X^2" = base_data^2, "Suma X" =
cumsum(base_data), "Suma X^2" = cumsum(base_data^2)),

                             "Normala (mu, sigma^2)" = list("X" =
base_data, "3-2X" = 3 - 2 * base_data, "X^2" = base_data^2, "Suma X" =
cumsum(base_data), "Suma X^2" = cumsum(base_data^2)),

                             "Exponentiala" = list("X" = base_data,
"2+5X" = 2 + 5 * base_data, "X^2" = base_data^2, "Suma X" =
cumsum(base_data)),

                             "Poisson" = list("X" = base_data, "3X-2" =
3 * base_data - 2, "X^2" = base_data^2, "Suma X" = cumsum(base_data)),

                             "Binomiala" = list("X" = base_data, "5X-4"
= 5 * base_data - 4, "Suma X" = cumsum(base_data))

    )




    plot(ecdf(transform_data$X), main = paste("Functia de repartitie
pentru", input$distribution),

         xlab = "Valori", ylab = "Functia de repartitie (CDF)", col =
"blue", lwd = 2)




    if (!is.null(transform_data$`3-2X`))
lines(ecdf(transform_data$`3-2X`), col = "red", lwd = 2)

    if (!is.null(transform_data$`X^2`))
lines(ecdf(transform_data$`X^2`), col = "green", lwd = 2)
```

```
    if (!is.null(transform_data$`Suma X`))
lines(ecdf(transform_data$`Suma X`), col = "purple", lwd = 2)

    if (!is.null(transform_data$`Suma X^2`))
lines(ecdf(transform_data$`Suma X^2`), col = "orange", lwd = 2)



    legend("bottomright", legend = names(transform_data), col =
c("blue", "red", "green", "purple", "orange"), lwd = 2)

  })

}
```

app.R

```
library(shiny)

source("ui.R")

source("server.R")

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

Vizualizare Functii de Repartitie

Alege distributia:

Normala (0,1)

Parametru 1:

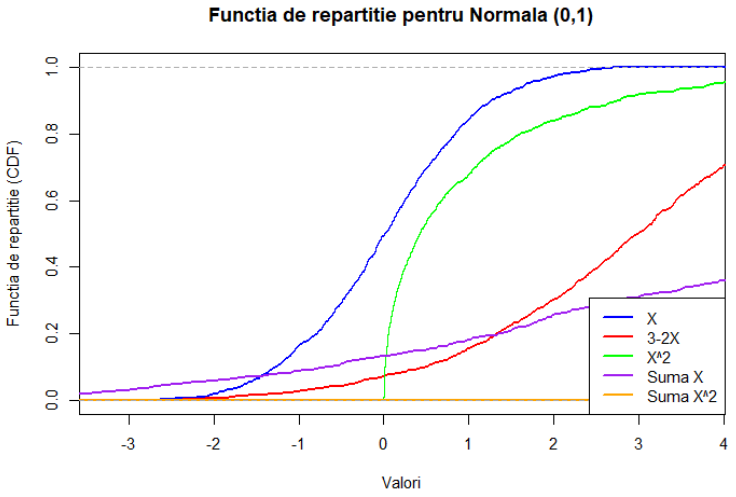
1

Parametru 2:

1

Numar de esantioane:

1000



Vizualizare Functii de Repartitie

Alege distributia:

Normala (mu, sigma^2)

Parametru 1:

1

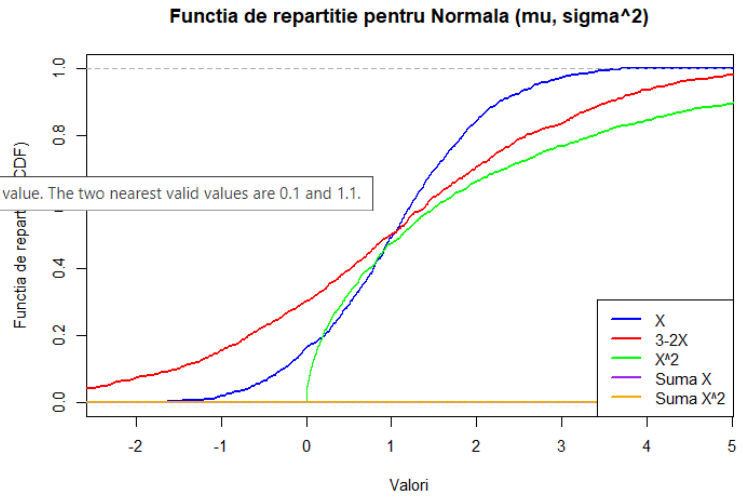
Please enter a valid value. The two nearest valid values are 0.1 and 1.1.

Parametru 2:

1

Numar de esantioane:

1000



Vizualizare Functii de Repartitie

Alege distributia:

Poisson

Parametru 1:

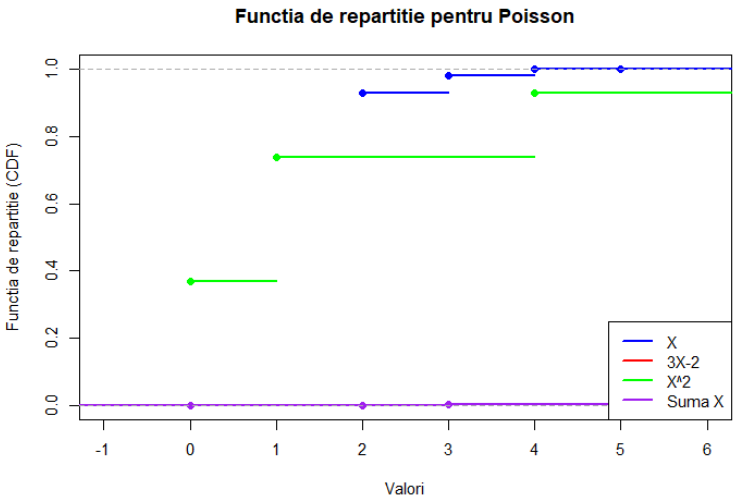
1

Parametru 2:

1

Numar de esantioane:

1000



Vizualizare Functii de Repartitie

Alege distributia:

Binomiala

Parametru 1:

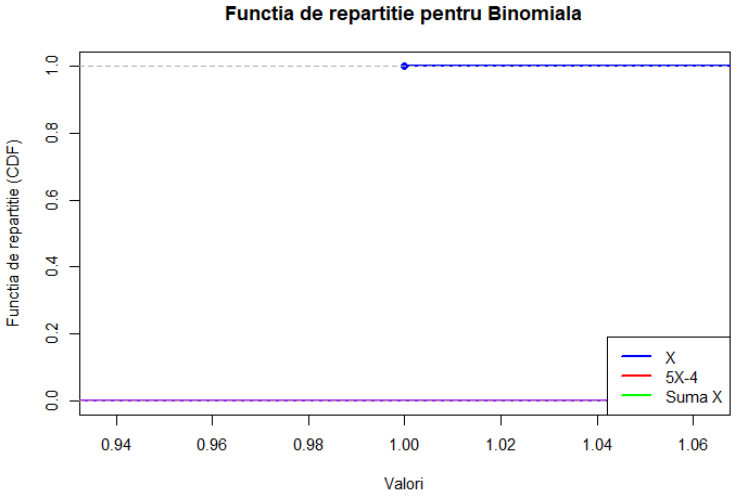
1

Parametru 2:

1

Numar de esantioane:

1000



Bibliografie:

1. **GitHub - RStudio/Shiny Examples. Disponibil la: https://github.com/rstudio/shiny-examples**
2. **Wikipedia contributors (2024). *Negative Binomial Distribution.* Wikipedia, The Free Encyclopedia. Disponibil la: https://en.wikipedia.org/wiki/Negative_binomial_distribution**
3. **R Core Team (2024). *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing. Disponibil la: https://www.r-project.org**
4. **Chang, W. (2021). *Mastering Shiny: Build Interactive Apps, Reports, and Dashboards with R.* O'Reilly Media. Disponibil la: https://mastering-shiny.org/**
5. **Wickham, H. (2024). *An Introduction to ggplot2.* Disponibil la: https://ggplot2-book.org/**