Proiect

Probabilitati si statistica.

Membrii: Stan David Florin, Stinga Alexandru Ionut, Veisa Radu George

Documentatie exercitiu 1

Se consideră o activitate care presupune parcurgerea secvențială a nn etape. Timpul necesar finalizării etapei ii de către o persoană A este o variabilă aleatoare Ti∼Exp(λi). După finalizarea etapei ii, A va trece în etapa i+1 cu probabilitatea αi sau va opri lucrul cu probabilitatea 1−αi. Fie T timpul total petrecut de persoana A în realizarea activității respective.

Rezolvarea cerintelor de la I:

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?

Rezolvare:

Prin intermediul unei functii vom calcula valoarea lui T, apoi o vom simula si vom face o medie a acestor rezultate. Se observa din rezultate urmatoarele:

- Asimetrie pozitivă / spre dreapta:
  - Histogramul arată o distribuție puternic asimetrică spre dreapta.
  - Majoritatea valorilor lui T sunt mici, dar există o coadă lungă spre valori mari.
  - Această asimetrie este caracteristică sumei variabilelor exponențiale trunchiate.
- Dominarea valorilor mici:
  - Într-un număr semnificativ de cazuri, persoana A nu finalizează toate etapele.
  - Timpul total T este mai mic în aceste cazuri, ceea ce produce un vârf semnificativ la valori mici ale lui T.

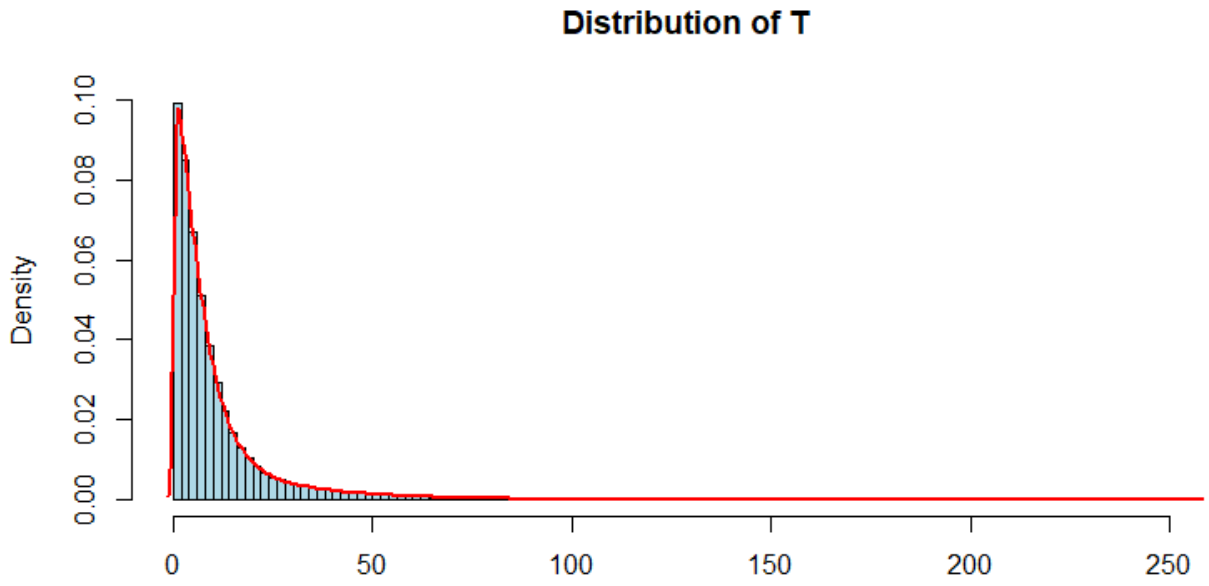
Cod:

```
n <- 10
lambda <- c(1, 0.8, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1, 0.1, 0.05)  # valorile pentru distributiile exponentiale
alpha <- c(0.9, 0.85, 0.8, 0.75, 0.7, 0.65, 0.6, 0.55, 0.5, 0.45)  # probabilitatile de a continua la pasul urmator
num_simulations <- 1e6

# Functie care simuleaza variabila aleatoare T
simulate_T <- function() {
  T <- 0
  for (i in 1:n) {
    T <- T + rexp(1, rate = lambda[i])
    # Genereaza un numar aleatoriu uniform in intervalul U(0,1). Daca acest numar este mai mare decat alpha[i], se opreste.
    if (runif(1) > alpha[i]) break
  }
  return(T)  # Returneaza suma totala
}

set.seed(123)
T_values <- replicate(num_simulations, simulate_T())  # Executa simularile
```

```
E_T <- mean(T_values)  # Calculeaza media valorilor simulate
cat("E(T) aproximat:", E_T, "\n")

# Reprezinta histograma lui T
hist(T_values, breaks = 100, probability = TRUE, main = "Distributia lui T",
     xlab = "T", ylab = "Densitate", col = "lightblue")

# Suprapune functia de densitate estimata
lines(density(T_values), col = "red", lwd = 2)

# Afiseaza statistici sumare despre T
cat("Rezumatul lui T:\n")
summary(T_values)
```

```
> summary(T_values)
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
  0.00001   2.56139   5.95830  10.53164  12.40778 256.75446
>
```

Histograma distributiei lui T.



2. Calculați valoarea exactă a lui E(T) și comparați cu valoarea obținută prin simulare.
   Rezolvare:
   Timpul total *T* reprezintă suma timpilor petrecuți în fiecare etapă prin care procesul trece, ponderată cu probabilitatea ca procesul să ajungă la acea etapă:

$$T = \sum_{i=1}^{n} T_i \cdot P(A \text{ ajunge la etapa } i)$$

   unde:

   - $T_i \sim Exp(\lambda_i)$ este timpul necesar pentru finalizarea etapei i, cu media: $1/\lambda_i$

- P(A ajunge la etapa i) este probabilitatea ca procesul să parcurgă toate etapele anterioare și să ajungă efectiv la etapa i.

Astfel E(T) devine:

$$E(T) = \sum_{i=1}^{n} E(T_i) \cdot P(A \text{ ajunge la etapa } i)$$

Probabilitatea ca *A ajunge la etapa i*:

$$P(A \text{ ajunge la etapa } i) = \prod_{j=1}^{i-1} \alpha_j$$

unde unde α_j este probabilitatea de a continua de la etapa j la j+1.

Stim ca $E(T_i) = 1/\lambda_i$, astfel incat:

Calculul pentru E(T):

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

Cod:

```
calculate_exact_ET <- function(lambda, alpha) {
  n <- length(lambda)
  ET <- 0
  prob_continue <- 1  # initial este 1 cand incepem pentru etapa 1

  for (i in 1:n) {
    ET <- ET + (1 / lambda[i]) * prob_continue
    prob_continue <- prob_continue * alpha[i]
  }
  return(ET)
}

# calculam exact E(T)
E_T_exact <- calculate_exact_ET(lambda, alpha)

# Rezultatele:
cat("Exact E(T):", E_T_exact, "\n")
cat("Approximated E(T) from simulation:", E_T, "\n")
cat("Absolute difference:", abs(E_T - E_T_exact), "\n")
```

```
Exact E(T): 10.51817
> cat("Approximated E(T) from simulation:", E_T, "\n")
Approximated E(T) from simulation: 10.53164
> cat("Absolute difference:", abs(E_T - E_T_exact), "\n")
Absolute difference: 0.01346798
```

Se observa ca diferenta dintre cele doua medii este foarte mica.

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

Rezolvare:

Vom modifica functia de la 1 ca sa putem returna un 2-tuplu (sau lista) (T,completed) care au urmatoarele semnificatii:

- T -> timpul de finalizare
- completed -> de tip bool care ne spune daca activitatea A a fost completata

Vom creea o lista cu valorile lui completed extrase din acel 2-tuplu si calculam media din ea.

Cod:

```
n <- 10
lambda <- c(1, 0.8, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1, 0.1, 0.05)
alpha <- c(0.9, 0.85, 0.8, 0.75, 0.7, 0.65, 0.6, 0.55, 0.5, 0.45)
num_simulations <- 1e6

simulate_T <- function() { #vom modifica vechea functie a.i sa returneze o lista compusa
din valoarea T si valoarea booleana completed.
  T <- 0
  for (i in 1:n) {
    T <- T + rexp(1, rate = lambda[i])
    if (runif(1) > alpha[i]) {
      return(list(T = T, completed = FALSE))  # Se opreste inainte de finalizare
    }
  }
  return(list(T = T, completed = TRUE))  # Daca toate etapele sunt parcurse, activitatea e
completa
}
set.seed(123)

results <- replicate(num_simulations, simulate_T(), simplify = FALSE)

# Extragem timpul total T din fiecare simulare
T_values <- sapply(results, function(x) x$T)

# Extragem indicatorul de finalizare a activitatii (TRUE / FALSE)
completion_flags <- sapply(results, function(x) x$completed)

# Calculam probabilitatea estimata ca activitatea sa fie finalizata complet
prob_completion <- mean(completion_flags)
cat("Probabilitatea ca persoana A sa finalizeze activitatea:", prob_completion, "\n")
```

```
> cat("Probabilitatea ca persoana A să finalizeze activitatea:", prob_completion, "\n")
Probabilitatea ca persoana A să finalizeze activitatea: 0.015683
>
> E_T <- mean(T_values)
> cat("Valoarea așteptată a timpului T pe baza simulărilor:", E_T, "\n")
Valoarea așteptată a timpului T pe baza simulărilor: 10.53164
>
> #3
> prob_completion <- mean(completion_flags)
> cat("Probabilitatea ca persoana A să finalizeze activitatea:", prob_completion, "\n")
Probabilitatea ca persoana A să finalizeze activitatea: 0.015683
```

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

   Din T_values, preluam numai acei T_values_i are completion_flags_i TRUE (1) si facem media din aceste valori.
   Cod:

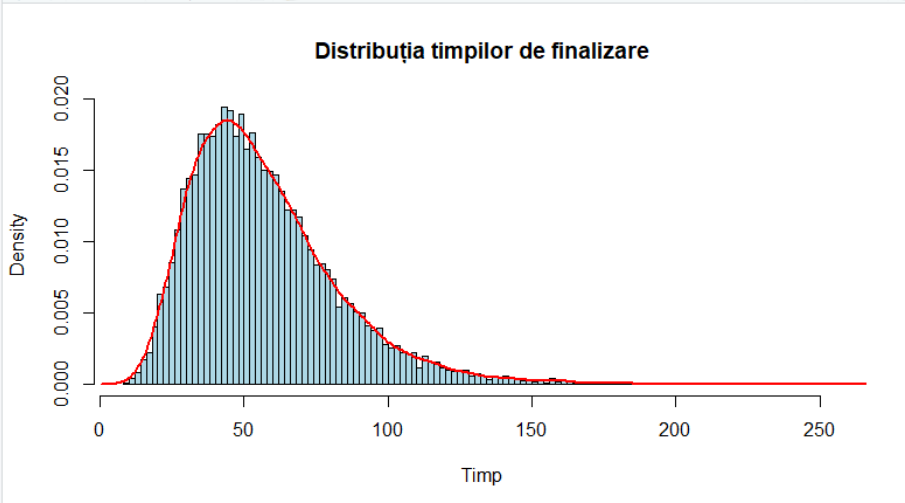
```
sigma <- 50
prob_T_le_sigma <- mean(T_values[completion_flags] <= sigma)
cat("Probabilitatea ca persoana A să finalizeze activitatea într-un timp ≤", sigma, ":",
prob_T_le_sigma, "\n")
```

```
> sigma <- 50
> prob_T_le_sigma <- mean(T_values[completion_flags] <= sigma)
> cat("Probabilitatea ca persoana A să finalizeze activitatea într-un timp ≤", sigma, ":", prob_T_le_sigma, "\n")
Probabilitatea ca persoana A să finalizeze activitatea într-un timp ≤ 50 : 0.4611363
```

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?
   Rezolvare:
   Preluam numai acei T_values unde activitatea A a fost terminat si aplicam min si max.
   Cod:

```
T_completed <- T_values[completion_flags]  # Timpii în care activitatea a fost completată
min_T <- min(T_completed)
max_T <- max(T_completed)
```

```
> #5
> T_completed <- T_values[completion_flags]  # Timpii în care activitatea a fost completată
> min_T <- min(T_completed)
> max_T <- max(T_completed)
> T_completed <- T_values[completion_flags]  # Timpii în care activitatea a fost completată
> min_T <- min(T_completed)
> max_T <- max(T_completed)
>
> cat("Timpul minim de finalizare:", min_T, "\n")
Timpul minim de finalizare: 9.65046
> cat("Timpul maxim de finalizare:", max_T, "\n")
Timpul maxim de finalizare: 256.7545
```



Se observa urmatoarele:

- Distributia este asimetrica spre dreapta, astfel majoritatea timpilor de finalizare sunt in mare parte mici.
- Timpul minim de finalizare este determinat de T1 ~ Exp(λ_1) (adica se opreste la prima etapa)
- Timpul maxim este determinat daca sunt parcurse toate cele n etape, timpul devenind:

$$T = \sum_{i=1}^{n} T_i$$

- Frecvența finalizărilor scade odată cu creșterea timpului total
- Timpul de finalizare este influențat de ratele λiλi și probabilitățile de continuare αiαi, valori mai mari ale λi reduc timpul de finalizare, iar valori mai mari ale αi îl cresc.

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?

Rezolvare:

Vom modifica din functia de la 1 ca sa putem returna si numarul de etape finalizate pe langa timpul T si completed.

Dupa executarea simulariilor, pentru fiecare etapa k, verifica pentru fiecare daca activitatea s-a oprit inate de etapa k, si face apoi media pentru fiecare.

Cod:

```
n <- 10  # Numarul maxim de etape
lambda <- c(1, 0.8, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1, 0.1, 0.05)
alpha <- c(0.9, 0.85, 0.8, 0.75, 0.7, 0.65, 0.6, 0.55, 0.5, 0.45)
num_simulations <- 1e6

simulate_T <- function() {
        T <- 0
        stages_completed <- 0  # Contor pentru numarul de etape finalizate

        for (i in 1:n) {
          T <- T + rexp(1, rate = lambda[i])
          stages_completed <- stages_completed + 1
```

```
    if (runif(1) > alpha[i]) return(list(T = T, completed = FALSE, stages_completed =
stages_completed))
  }
  return(list(T = T, completed = TRUE, stages_completed = n))
}
set.seed(123)

# Rulam simularile
results <- replicate(num_simulations, simulate_T(), simplify = FALSE)

probabilities <- sapply(2:n, function(k) {
  stop_before_k <- sapply(results, function(x) {
    return(!x$completed && x$stages_completed < k)  # Verificam daca activitatea s-a
oprit inainte de k
  })
  mean(stop_before_k)  # Media valorilor TRUE/FALSE da probabilitatea de oprire
inainte de k
})

# Generam graficul probabilitatilor de oprire inainte de etapa k
plot(2:n, probabilities, type = "b", pch = 19, col = "blue",
     main = "Probabilitatea de oprire inainte de etapa k",
     xlab = "Etapa k", ylab = "Probabilitate")

# Afisam probabilitatile sub forma de bara
barplot(probabilities, names.arg = 2:n, col = "blue",
        main = "Probabilitatea de oprire inainte de etapa k",
xlab = "Etapa k", ylab = "Probabilitate")
```

Diagrama cu barplot:

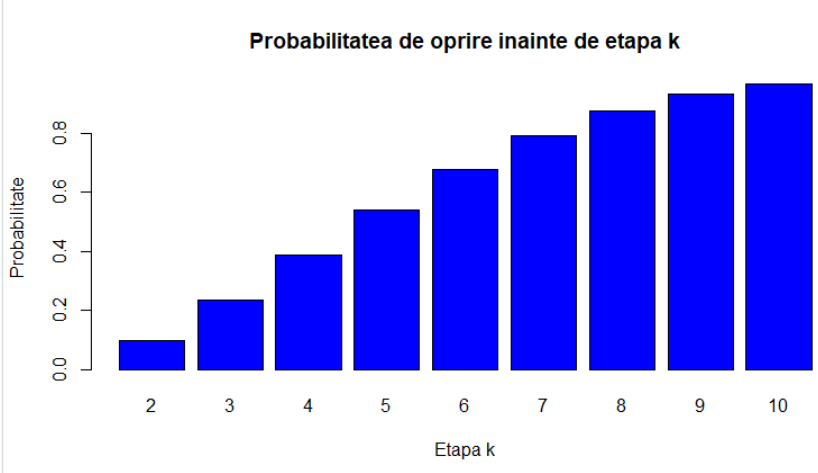
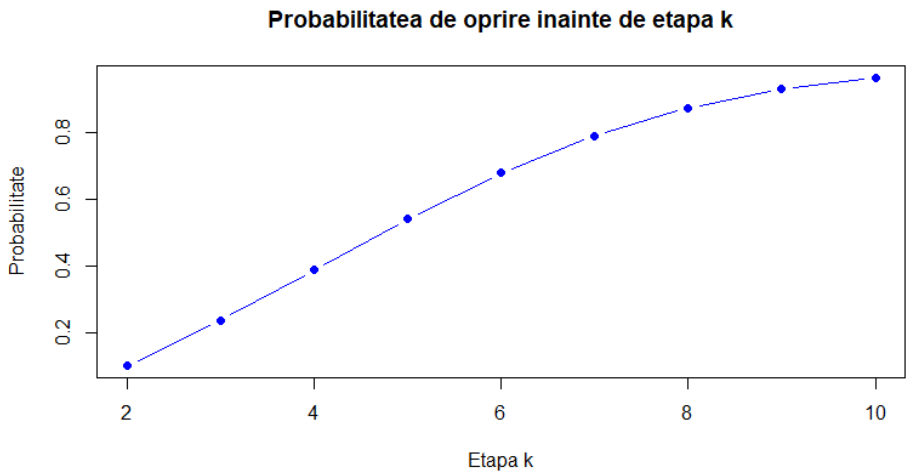


Diagrama cu plot:



Avem urmatoarele observatii:

- Distribuția probabilităților P(K<k) este monoton crescătoare
- La început (valori mici ale lui k), probabilitatea de oprire este mică, deoarece majoritatea simulărilor continuă cel puțin câteva etape. Pe măsură ce k crește, probabilitatea de oprire devine tot mai mare, deoarece sunt mai multe oportunități pentru ca activitatea să fie întreruptă.
- Pentru k=n, probabilitatea de oprire este aproape 1, ceea ce înseamnă că foarte puține simulări finalizează toate cele n etape.
- Forma distribuției reflectă direct valorile parametrilor αi – valori mai mici ale lui αi determină o creștere mai rapidă a probabilității de oprire.

Identificarea unor eventuale dificultăți în realizarea cerințelor:

- Invatarea functiilor de baza din R, cum se realizeaza ele, cum functioneaza functiile in R in general si ce manevre se pot realiza cu ele
- Interpretarea rezultatelor pe graphic, intrucat nu se stia despre ce trebuie discutat si ce referinta sa se ghideze
- Analiza influentelor parametrilor αi si λ_i

Notiuni noi: nu au existat

Nu au fost utilizate alte **pachete**, iar ca **surse de inspiratie** au fost:
https://www.rdocumentation.org/
https://www.r-project.org/other-docs.html
https://stackoverflow.com/

Documentație Cerința 2

1. Introducere

Această aplicație Shiny este destinată vizualizării și explorării repartiției Negative Binomiale sub cele cinci formulări alternative. Utilizatorii pot interacționa cu aplicația prin selectarea parametrilor și observarea efectelor acestora asupra funcției de masă a probabilității (PMF) și a funcției de repartiție (CDF). De asemenea, aplicația include o histograma realizata pe baza unui exemplu de problema care faciliteaza utilizarea unei repartitii negativ binomiala.

2. Funcționalități principale

- Selectarea uneia dintre cele cinci formulări ale repartiției negativ binomiale.
- Reglarea parametrilor distribuției prin controale interactive.
- Reprezentarea grafică a funcției de masă a probabilității (PMF) și a funcției de repartiție (CDF).
- Simularea unui scenariu real de utilizare a repartitiei.

3. Exemple de utilizare practică

Un exemplu practic in care repartitia negativ binomiala este relevanta este:

> Un agent de vânzări încearcă să încheie contracte prin apeluri telefonice. Șansele ca un apel să se finalizeze cu succes (un contract semnat) sunt de 20% (p = 0.2). De cate ori va esua agentul sa incheie contracte până când reușește să încheie 5 contracte?

Acest exemplu este ilustrat folosind o simulare a repartiției Negative Binomiale, permițând utilizatorului să observe distribuția numărului de apeluri necesare pentru a obține cele 5 succese dorite.

4. Descrierea implementării

4.1 Definirea interfeței utilizatorului (UI)

Interfața utilizatorului este proiectată astfel încât să fie intuitivă și interactivă. Utilizatorul poate selecta una dintre cele cinci formulări disponibile pentru distribuția Negative Binomială prin intermediul unui sidebarPanel().

Pentru fiecare formulare, utilizatorul are posibilitatea de a regla parametrii corespunzători:

- Numărul de succese (r)
- Numărul de eșecuri (k)
- Numărul total de încercări (n)
- Probabilitatea de succes a fiecărei încercări (p)

(semnificatia parametrilor poate suferi schimbari i.e. k apare si ca successuri si ca esecuri in formulari)

Acești parametri sunt setați cu ajutorul sliderInput(), oferind utilizatorului flexibilitate în explorarea distribuției.

`Pentru accesarea ploturilor, a histogramei si a exemplului se utilizeaza un mainPanel() si un tabelsetPanel(), in cadrul caruia se afla fiecare obiectiv sub forma de tabPanel().

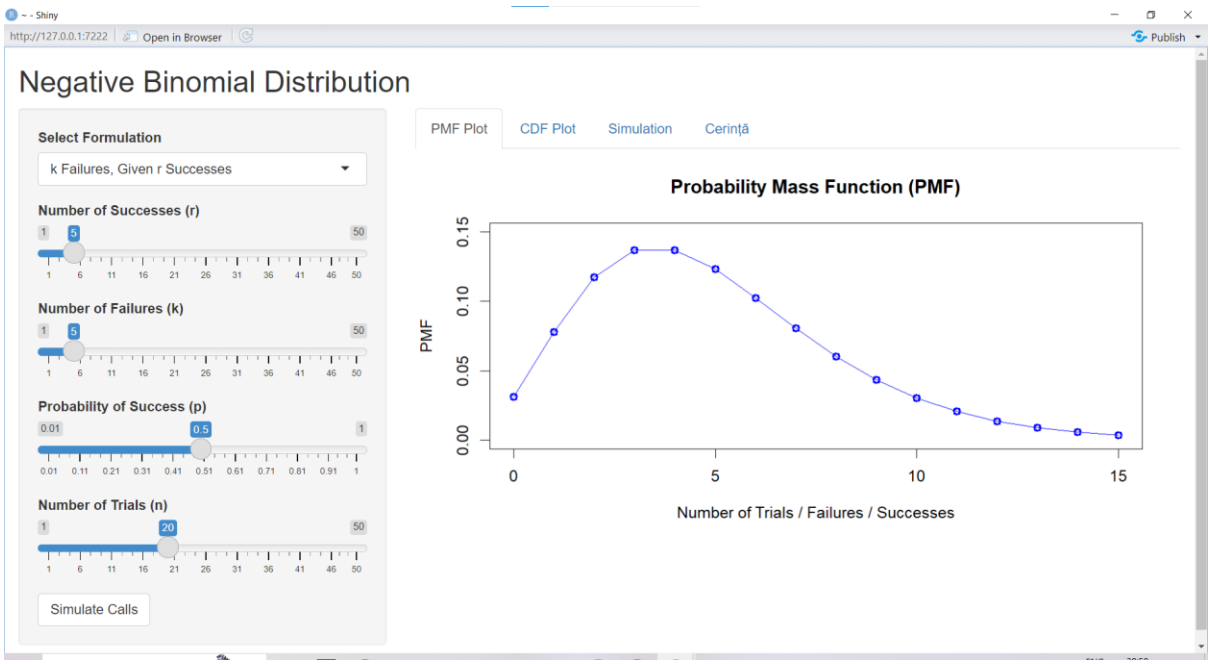
4.2 Logica serverului

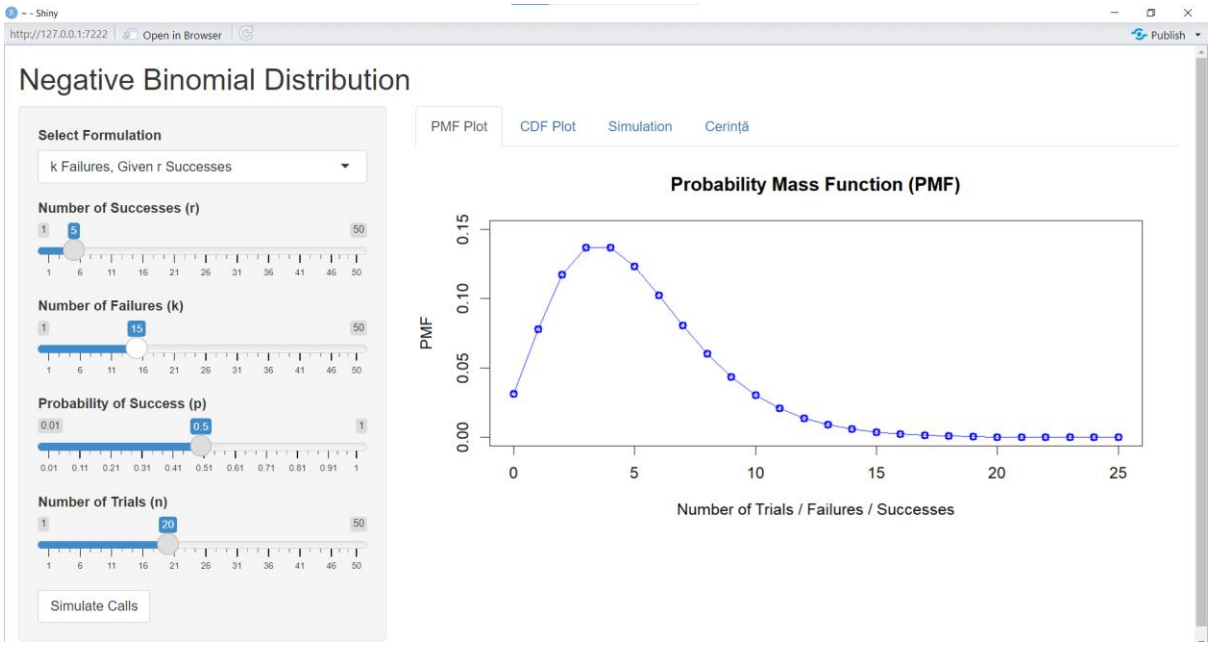
La nivelul serverului, pentru fiecare formulare selectată, se calculează valorile necesare trasării funcției de masă a probabilității (PMF). Aceste calcule se realizează folosind formula matematică specifică fiecărei formulări, având în vedere relațiile combinatoriale și probabilistice corespunzătoare.

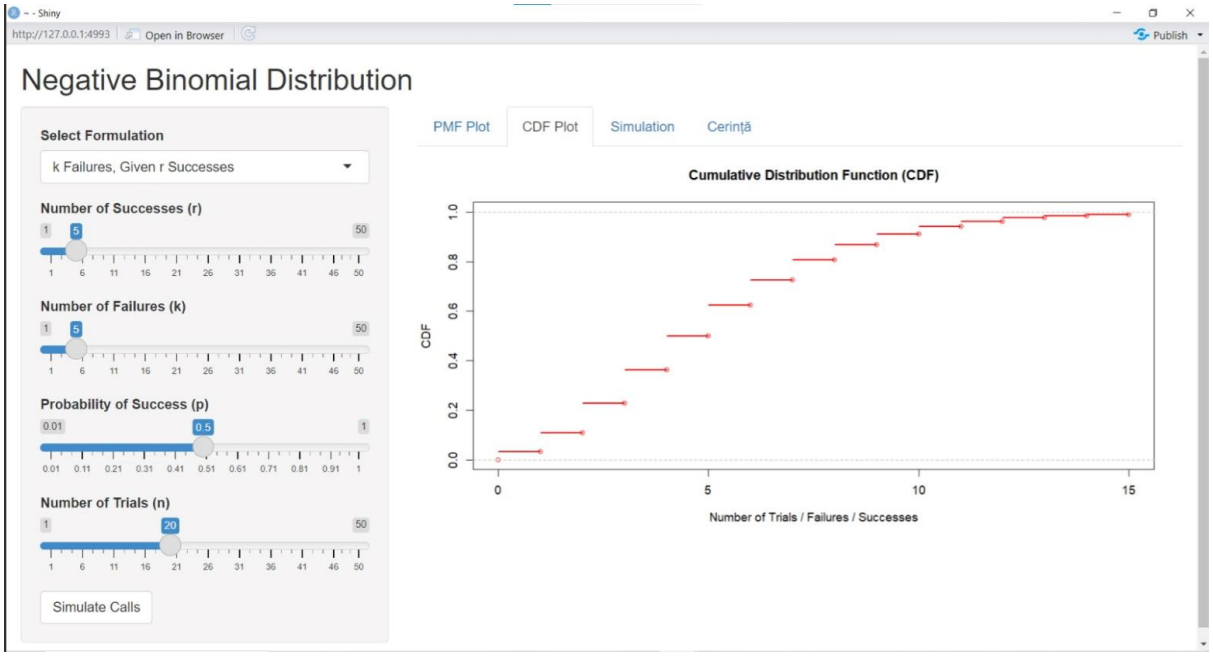
Funcția reactive() este utilizată pentru a calcula valorile PMF și CDF în funcție de parametrii introduși. Pentru trasarea graficelor, se folosește renderPlot(), iar cumsum() este utilizată pentru calculul funcției de repartiție cumulată (CDF).

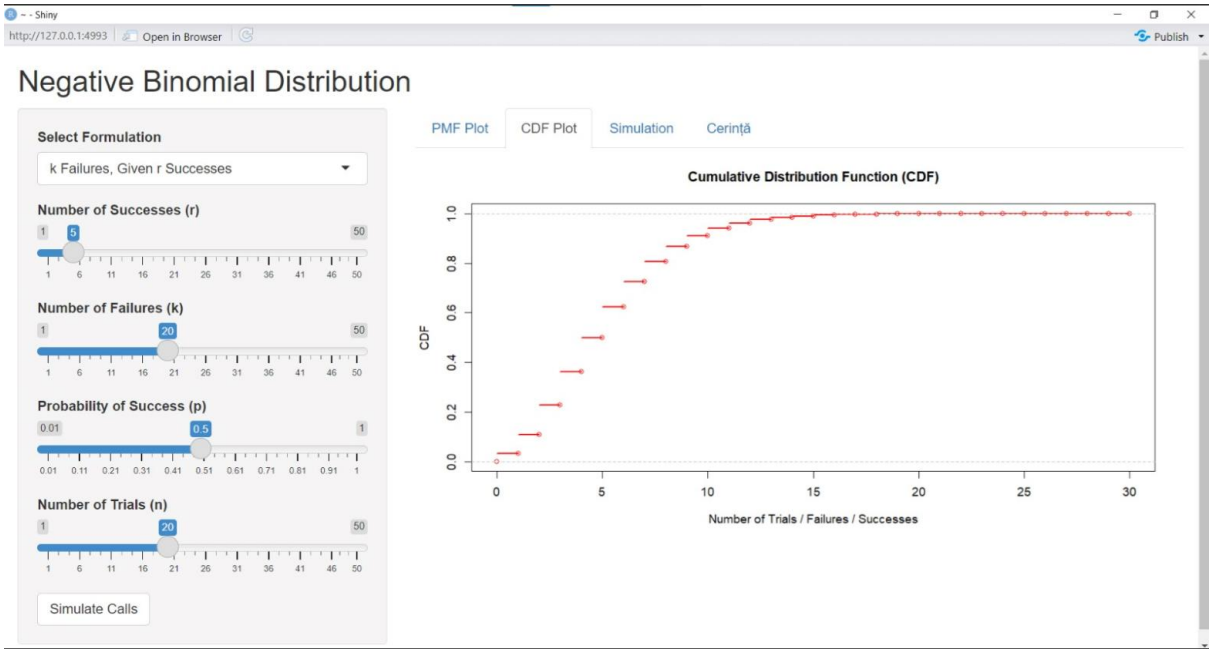
Pentru simularea unui scenariu practic, observeEvent() declanșează generarea unui set de date aleatorii pe baza distribuției Negative Binomiale, utilizând rnbinom(). Rezultatele sunt apoi afișate sub forma unei histograme, ilustrând distribuția nr de apeluri esuate pentru a obține un anumit număr de succese.

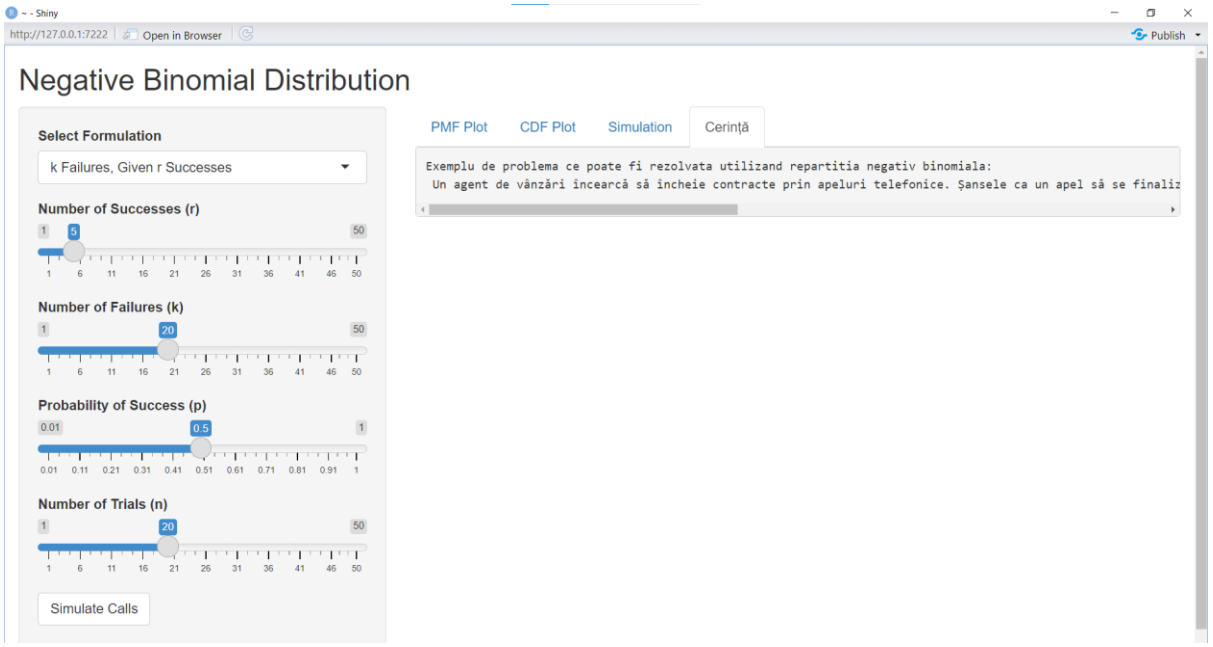
În plus, aplicația afișează un scenariu explicativ care ilustrează utilizarea distribuției într-un caz real. Acest scenariu este generat folosind renderText() și apare într-un tab separat din interfață.




~ - Shiny
http://127.0.0.1:7222
Open in Browser
Publish
Negative Binomial Distribution
Select Formulation
k Failures, Given r Successes
Number of Successes (r)
Number of Failures (k)
Probability of Success (p)
Number of Trials (n)
Simulate Calls
PMF Plot
CDF Plot
Simulation
Cerință
Probability Mass Function (PMF)
PMF
Number of Trials / Failures / Successes


~ - Shiny
http://127.0.0.1:4993
Open in Browser
Publish
Negative Binomial Distribution
Select Formulation
k Failures, Given r Successes
Number of Successes (r)
Number of Failures (k)
Probability of Success (p)
Number of Trials (n)
Simulate Calls
PMF Plot
CDF Plot
Simulation
Cerință
Cumulative Distribution Function (CDF)
CDF
Number of Trials / Failures / Successes


~ - Shiny
http://127.0.0.1:4993
Open in Browser
Publish
Negative Binomial Distribution
Select Formulation
k Failures, Given r Successes
Number of Successes (r)
Number of Failures (k)
Probability of Success (p)
Number of Trials (n)
Simulate Calls
PMF Plot
CDF Plot
Simulation
Cerință
Cumulative Distribution Function (CDF)
CDF
Number of Trials / Failures / Successes


~ - Shiny
http://127.0.0.1:7222
Open in Browser
Publish
Negative Binomial Distribution
Select Formulation
k Failures, Given r Successes
Number of Successes (r)
Number of Failures (k)
Probability of Success (p)
Number of Trials (n)
Simulate Calls
PMF Plot
CDF Plot
Simulation
Cerință
Exemplu de problema ce poate fi rezolvata utilizand repartitia negativ binomiala:
Un agent de vânzări încearcă să încheie contracte prin apeluri telefonice. Șansele ca un apel să se finaliz

5.Cod

```
library(shiny)


## realizarea interefetei
ui <- fluidPage(
  titlePanel("Negative Binomial Distribution"),


  ## selector pentru formulare
  sidebarLayout(
    sidebarPanel(
      selectInput("formulation", "Select Formulation",
                  choices = c(
                    "k Failures, Given r Successes",
                    "n Trials, Given r Successes",
                    "n Trials, Given r Failures",
```

```
                "k Successes, Given r Failures",
                "k Successes, Given n Trials"
              )),
    ## imputul parametrilor
    sliderInput("r", "Number of Successes (r)", min = 1, max = 50, value = 5),
    sliderInput("k", "Number of Failures (k)", min = 1, max = 50, value = 5),
    sliderInput("p", "Probability of Success (p)", min = 0.01, max = 1, value = 0.5, step = 0.01),
    sliderInput("n", "Number of Trials (n)", min = 1, max = 50, value = 20),
    actionButton("simulate", "Simulate Calls")
  ),

  mainPanel(
    tabsetPanel(
      tabPanel("PMF Plot", plotOutput("plot_pmf")),
      tabPanel("CDF Plot", plotOutput("plot_cdf")),
      tabPanel("Simulation",
               plotOutput("histogram"),
               verbatimTextOutput("summary")),
      tabPanel("Cerință",
               verbatimTextOutput("problem_statement"))
    )
  )
 )
)

server <- function(input, output, session) {

  data <- reactive({
```

```
    req(input$formulation, input$r, input$k, input$p, input$n)

    ## Ne asigurăm că toate valorile de intrare necesare (formularea, r, k, p, n) sunt disponibile

    x_vals <- NULL

    pmf_vals <- NULL

    cdf_vals <- NULL

    ## verificarea formularii si calculul valorilor pe baza formulelor matematice

    if (input$formulation == "k Failures, Given r Successes") {

      x_vals <- 0:(input$k+10)

      ##Definirea valorilor (am adaugat 10 pt o vedere mai buna pe grafic)

      pmf_vals <- choose(x_vals + input$r - 1, x_vals) * (input$p^input$r) * ((1 - input$p)^x_vals)

    } else if (input$formulation == "n Trials, Given r Successes") {

      x_vals <- input$r:input$n

      pmf_vals <- choose(x_vals - 1, input$r - 1) * (input$p^input$r) * ((1 - input$p)^(x_vals -
input$r))

    } else if (input$formulation == "n Trials, Given r Failures") {

      x_vals <- input$r:input$n

      pmf_vals <- choose(x_vals - 1, input$r - 1) * (input$p^(x_vals - input$r)) * ((1 -
input$p)^input$r)

    } else if (input$formulation == "k Successes, Given r Failures") {

      x_vals <- 0:(input$k + 10)

      pmf_vals <- choose(x_vals + input$r - 1, x_vals) * (input$p^x_vals) * ((1 - input$p)^input$r)

    } else if (input$formulation == "k Successes, Given n Trials") {

      x_vals <- 0:input$n

      pmf_vals <- dbinom(x_vals, size = input$k, prob = input$p)

      ## utilizarea functiei specifice repartitiei binomiale

    }



    cdf_vals <- cumsum(pmf_vals)

    ## vectorul de sume cumulate
```

```
  list(x = x_vals, pmf = pmf_vals, cdf = cdf_vals)
})

## Trasarea ploturilor si a histogramei
output$plot_pmf <- renderPlot({
  data_vals <- data()
  plot(data_vals$x, data_vals$pmf, col = "blue", lwd = 2,
       main = "Probability Mass Function (PMF)",
       xlab = "Number of Trials / Failures / Successes", ylab = "PMF",
       ylim = c(0, max(data_vals$pmf, na.rm = TRUE) * 1.1))

  lines(data_vals$x, data_vals$pmf, col = "blue")
}, res = 96)

## plot de tip staircase pt cdf
output$plot_cdf <- renderPlot({
  data_vals <- data()

  # Add a 0 at the start of the cumsum vals
  cdf_vals_cdf <- c(0, data_vals$cdf)
  x_vals_cdf <- c(0, data_vals$x)

  # Create an empty plot
  plot(x = NA, y = NA, pch = NA,
       xlim = c(0, max(x_vals_cdf)),
       ylim = c(0, 1),
       xlab = "Number of Trials / Failures / Successes",
       ylab = "CDF",
```

```
      main = "Cumulative Distribution Function (CDF)")



  # Add open points (pch = 1)
  points(x = x_vals_cdf[-1], y = cdf_vals_cdf[-length(cdf_vals_cdf)], pch = 1, col = "red")


  # Draw horizontal lines between consecutive points
  for (i in 1:(length(x_vals_cdf)-1)) {
    segments(x_vals_cdf[i], cdf_vals_cdf[i], x_vals_cdf[i+1], cdf_vals_cdf[i], col = "red", lwd = 2)
  }


  # Add horizontal lines at y = 0 and y = 1
  abline(h = c(0, 1), col = "grey", lty = 2)
})



## eventul este apasarea butonului de simulare
observeEvent(input$simulate, {
  req(input$r, input$p)
  ## Ne asigurăm că valorile r și p sunt disponibile pentru simulare
  trials <- rnbinom(1000, size = input$r, prob = input$p)
  ## Generăm 1000 de valori simulatoare folosind distribuția binomială negativă
  ## cu r succesuri și probabilitatea de succes p
  output$histogram <- renderPlot({
    hist(trials, breaks = 30, col = "skyblue", main = "Simulation of Sales Calls",
         xlab = "Number of Calls Needed", probability = FALSE)
    ## Construim histograma pentru valorile generate
    ## "breaks" reprezintă numărul de intervale pe axa X
```

```
  })

 })



 ## afisarea exemplului

 output$problem_statement <- renderText({

   "Exemplu de problema ce poate fi rezolvata utilizand repartitia negativ binomiala:\n Un agent de vânzări încearcă să încheie contracte prin apeluri telefonice. Șansele ca un apel să se finalizeze cu succes (un contract semnat) sunt de 20% (p = 0.2).  De cate ori va esua agentul sa incheie contracte până când reușește să încheie 5 contracte?"

 })

}



shinyApp(ui, server)







library(shiny)

runApp("C:/prob_R/prapp2")
```

6. Pachete software utilizate si surse de inspiratie

Cele doua pacheta pe care le-am utilizat in realizarea cerintei au fost shiny(utilizat in realizarea aplicatiei si a interfetei grafice) si stats (utilizat indirect prin functiile repartitiilor (ex: dbinom,rnbinom))

In ceea ce priveste sursele de inspiratie, am utilizitat informatiile de pe site-ul https://shiny.posit.co/ pentru partea de R si https://en.wikipedia.org/wiki/Negative_binomial_distribution/ pentru a ma documenta despre repartitia negativ binomiala.

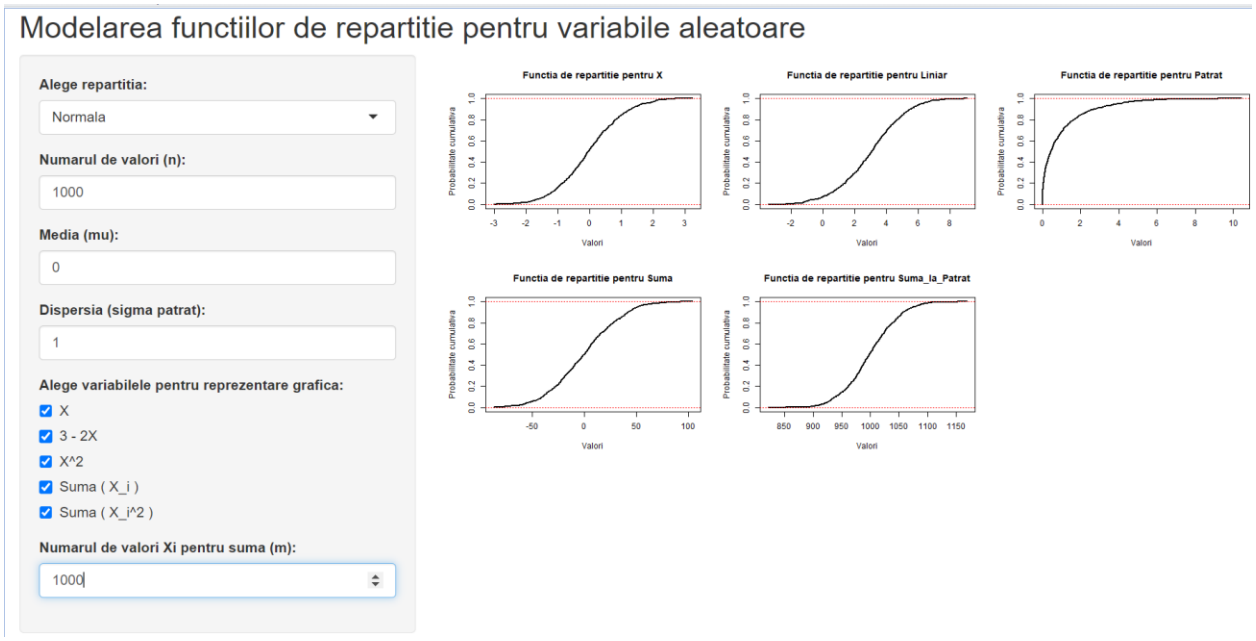
7. Notiuni noi

In ceea ce priveste notiunile de calcul probabilistic nu am utilizat nimic suplimentar fata de curs. Noutatea vine din partea de R, si mai concret partea de shiny (structurarea in ui si server, functii precum fluidPage(), titlePanel(), observeEvent(), renderText() etc
).

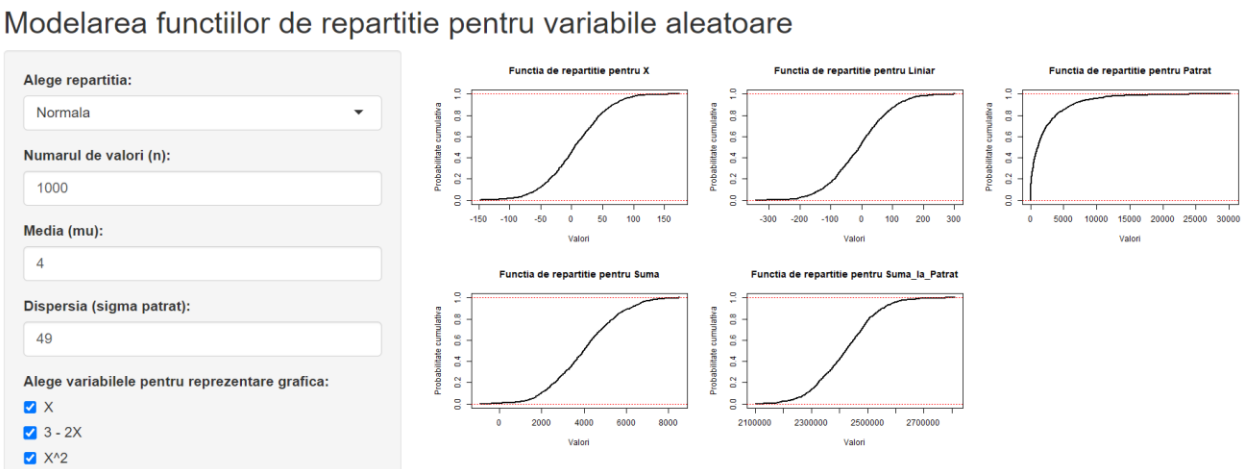
Documentatie exercitiu 3

Cerinta: Construiti o aplicatie Shiny in care sa reprezentati grafic functiile de repartitie pentru urmatoarele variabile aleatoare :

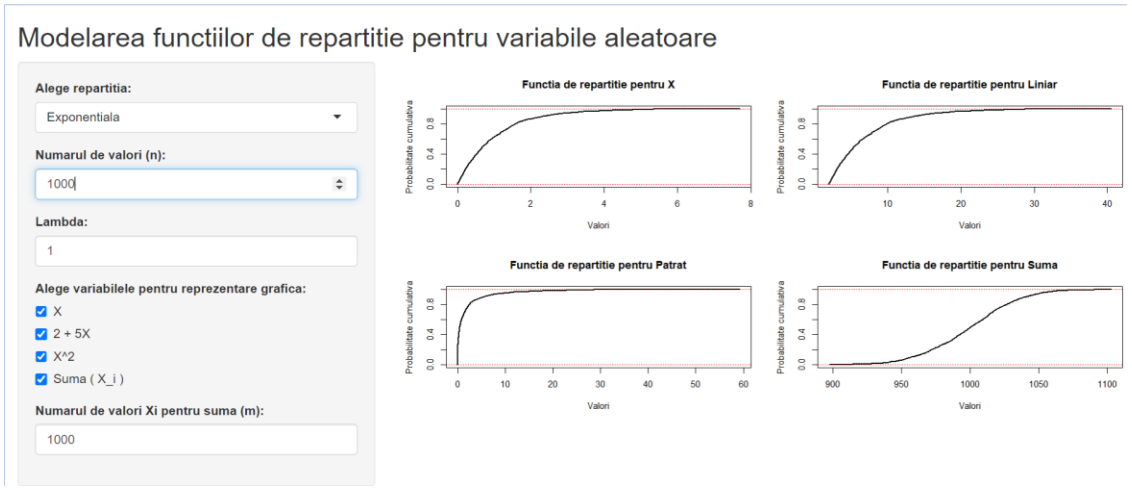
1) X, 3-2X, X^2, $\sum_{i=1}^{n} X\,i$, $\sum_{i=1}^{n} X\,i^2$, unde X, X_1, X_2, …, X_n i.i.d ~ N(0, 1), n ∈ ℕ fixat



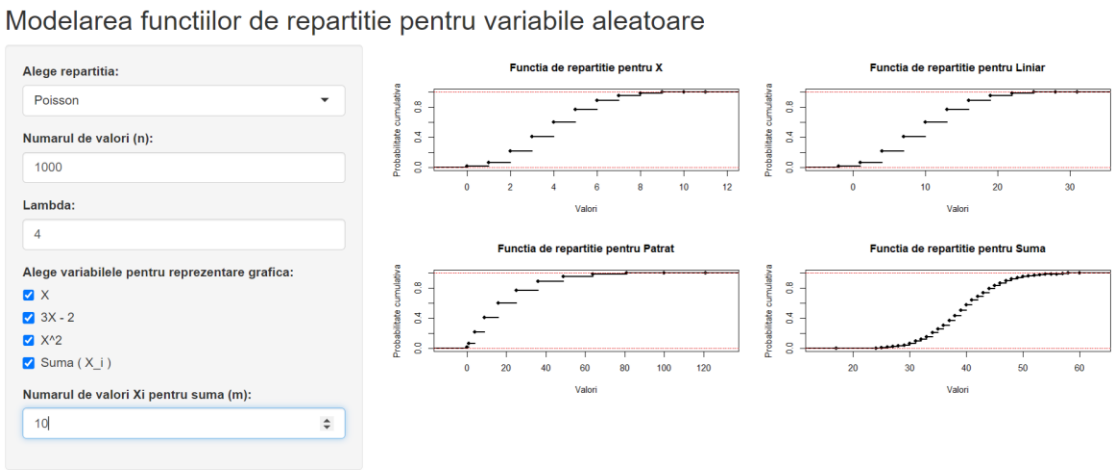
2) X, 3-2X, X^2, $\sum_{i=1}^{n} X\,i$, $\sum_{i=1}^{n} X\,i^2$, unde X, X_1, X_2, …, X_n i.i.d ~ N(μ, σ^2), μ ∈ ℝ, σ>0, n ∈ ℕ fixat



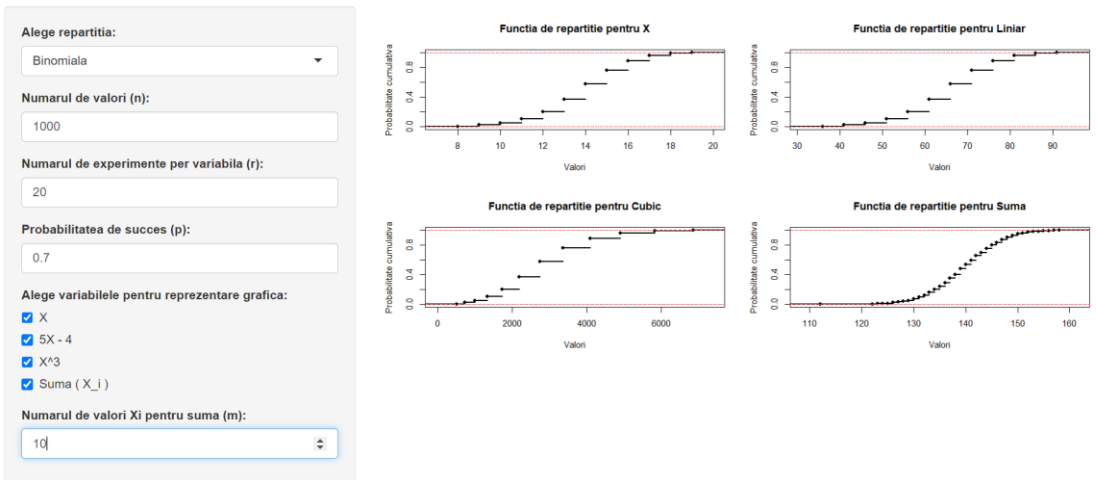
3) X, 2+5X, X^2, $\sum_{i=1}^{n} X\,i, unde\ X, X1, X2, \dots, Xn\ i.i.d \sim Exp(\lambda), \lambda > 0, n \in \mathbb{N}\ fixat$



4) X, 3X-2, X^2, $\sum_{i=1}^{n} X\,i$, unde X, X_1, X_2, …, X_n i.i.d ~ Pois(λ), λ>0, n ∈ ℕ fixat



5) X, 5X-4, X^3, $\sum_{i=1}^{n} X\,i$, unde X, X_1, X_2, …, X_n i.i.d ~ Binom(r, p), r ∈ ℕ, p ∈ (0,1), n ∈ ℕ fixat



Cod:

```
library(shiny)
library(grid)

set.seed(999)

# Definirea (UI)
ui <- fluidPage(
  titlePanel("Modelarea functiilor de repartitie pentru variabile aleatoare"),  # Titlul aplicatiei
  sidebarLayout(  # Dispunerea aplicatiei in 2 coloane
    sidebarPanel(  # Panoul lateral pentru selectii si intrari
      selectInput("rep", "Alege repartitia:",  # Selecteaza tipul de repartitie
                  choices = list("Normala" = "norm",
                                 "Exponentiala" = "exp",
                                 "Poisson" = "pois",
                                 "Binomiala" = "binom")),
      numericInput("n", "Numarul de valori (n):", value = 100, min = 1),  # Intrare pentru numarul de
valori generate
      conditionalPanel(  # Panou conditionat pentru repartitia normala
        condition = "input.rep == 'norm'",
        numericInput("mu", "Media (mu):", value = 0),
        numericInput("sigma", "Dispersia (sigma patrat):", value = 1, min = 0.01)
      ),
      conditionalPanel(  # Panou conditionat pentru repartitia exponentiala
        condition = "input.rep == 'exp'",
        numericInput("lambda", "Lambda:", value = 1, min = 0.01)
      ),
      conditionalPanel(  # Panou conditionat pentru repartitia Poisson
        condition = "input.rep == 'pois'",
        numericInput("lambda_pois", "Lambda:", value = 0.5, min = 0.01)
      ),
      conditionalPanel(  # Panou conditionat pentru repartitia binomiala
        condition = "input.rep == 'binom'",
        numericInput("size", "Numarul de experimente per variabila (r):", value = 10, min = 1),
        numericInput("prob", "Probabilitatea de succes (p):", value = 0.5, min = 0, max = 1)
      ),
      checkboxGroupInput("vars", "Alege variabilele pentru reprezentare grafica:", choices = list()),
      conditionalPanel(  # Panou conditionat pentru intrarea m, care este necesara doar la suma
        condition = "input.vars.includes('Suma') || input.vars.includes('Suma_la_Patrat')",
        numericInput("m", "Numarul de valori Xi pentru suma (m):", value = 10, min = 1)
      )
    ),
    mainPanel(  # Panoul principal pentru afisarea graficelor
      plotOutput("cdfPlot")  # Zona in care se va afisa graficul CDF
    )
  )
)

# Definirea serverului
server <- function(input, output, session) {
```

```
  observeEvent(input$rep, {  # Reactia la schimbarea repartitiei selectate
    choices <- switch(input$rep,
                      "norm" = list("X" = "X", "3 - 2X" = "Liniar", "X^2" = "Patrat", "Suma ( X_i )" =
"Suma", "Suma ( X_i^2 )" = "Suma_la_Patrat"),
                      "exp" = list("X" = "X", "2 + 5X" = "Liniar", "X^2" = "Patrat", "Suma ( X_i )" = "Suma"),
                      "pois" = list("X" = "X", "3X - 2" = "Liniar", "X^2" = "Patrat", "Suma ( X_i )" = "Suma"),
                      "binom" = list("X" = "X", "5X - 4" = "Liniar", "X^3" = "Cubic", "Suma ( X_i )" =
"Suma"))
    updateCheckboxGroupInput(session, "vars", choices = choices, selected = "X")  # Actualizeaza
optiunile grafice disponibile
  })
  X_data <- reactive({
    req(input$n, input$rep)  # Asigura-te ca n si rep sunt furnizate
    switch(input$rep,
           "norm" = rnorm(input$n, mean = input$mu, sd = input$sigma),
           "exp" = rexp(input$n, rate = input$lambda),
           "pois" = rpois(input$n, lambda = input$lambda_pois),
           "binom" = rbinom(input$n, size = input$size, prob = input$prob))
  })

  # Calculez Suma si Suma_la_Patrat doar cand se modifica m
  Suma_data <- reactive({
    req(input$m)
    replicate(input$n, sum(switch(input$rep,
                                  "norm" = rnorm(input$m, mean = input$mu, sd = input$sigma),
                                  "exp" = rexp(input$m, rate = input$lambda),
                                  "pois" = rpois(input$m, lambda = input$lambda_pois),
                                  "binom" = rbinom(input$m, size = input$size, prob = input$prob))))
  })

  Suma_la_Patrat_data <- reactive({
    req(input$m)
    replicate(input$n, sum(switch(input$rep,
                                  "norm" = rnorm(input$m, mean = input$mu, sd = input$sigma)^2,
                                  "exp" = rexp(input$m, rate = input$lambda)^2,
                                  "pois" = rpois(input$m, lambda = input$lambda_pois)^2,
                                  "binom" = rbinom(input$m, size = input$size, prob = input$prob)^2)))
  })

  # Renderizarea graficului pentru functia de repartitie
  output$cdfPlot <- renderPlot({
    if (length(input$vars) == 0) {
      grid::grid.newpage()
      grid::grid.rect(gp = grid::gpar(col = "white"))  # Pagina goala
      return()
    }
    data <- data.frame(X = X_data())

    # Adaugarea variabilelor aleatoare
```

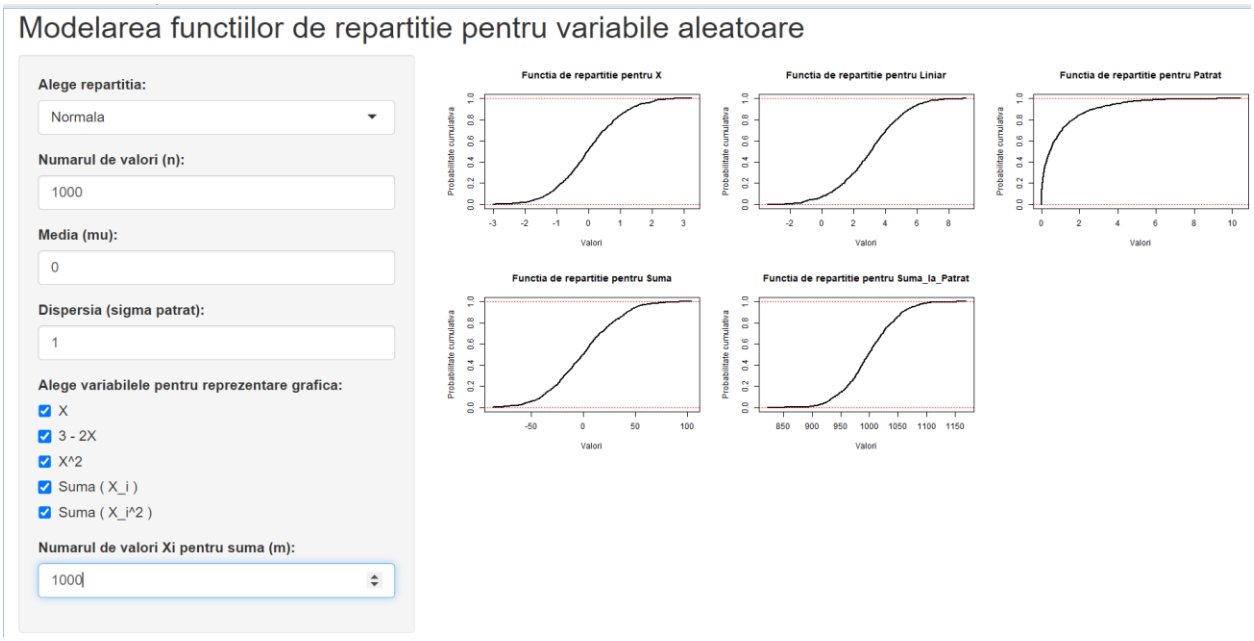
```
    if ("Liniar" %in% input$vars) data$Liniar <- switch(input$rep, "norm" = 3 - 2 * data$X, "exp" = 2 +
5 * data$X, "pois" = 3 * data$X - 2, "binom" = 5 * data$X - 4)
    if ("Patrat" %in% input$vars) data$Patrat <- data$X^2
    if ("Cubic" %in% input$vars) data$Cubic <- data$X^3
    if ("Suma" %in% input$vars) data$Suma <- Suma_data()
    if ("Suma_la_Patrat" %in% input$vars) data$Suma_la_Patrat <- Suma_la_Patrat_data()

    # Seteaza layout-ul in functie de numarul de variabile selectate
    num_plots <- length(input$vars)
    num_cols <- ceiling(sqrt(num_plots))  # Determina numarul de coloane
    num_rows <- ceiling(num_plots / num_cols)  # Determina numarul de randuri

    par(mfrow = c(num_rows, num_cols))  # Imparte fereastra in mai multe subgrafice

    for (var in input$vars) {
      if (input$rep %in% c("pois", "binom")) {
        cdf <- ecdf(data[[var]])
        plot(cdf, main = paste("Functia de repartitie pentru", var),
             xlab = "Valori", ylab = "Probabilitate cumulativa",
             col = "black", lwd = 2, pch = 16, ylim = c(0, 1))
        abline(h = 1, col = "red", lty = 3)
        abline(h = 0, col = "red", lty = 3)

      } else {
        cdf <- ecdf(data[[var]])
        df <- data.frame(x = sort(data[[var]]), y = cdf(sort(data[[var]])))

        plot(df$x, df$y, type = "l", col = "black", lwd = 2,
             xlab = "Valori", ylab = "Probabilitate cumulativa",
             main = paste("Functia de repartitie pentru", var),
             xlim=range(df$x))
        abline(h = 1, col = "red", lty = 3)
        abline(h = 0, col = "red", lty = 3)
      }
    }

    par(mfrow = c(1,1))  # Reseteaza layout-ul
  })
}

# Lansarea aplicatiei Shiny
shinyApp(ui = ui, server = server)
```

Mod de lucru: Am utilizat pachetele shiny si grid atat pentru partea de reprezentare a meniurilor de introducere a parametrilor si desenarea graficelor respectivelor variabile aleatoare, cat si pentru generarea si calcularea lor si a ecdf-ului. Am setat un seed fix, pentru a putea obtine aceleasi rezultate de fiecare data cand este rulata aplicatia.

Generare: A fost realizata prin intermediul functiilor random specific repartitiilor (rnorm, rexp, rpois, rbinom). Parametrii utilizati au fost cei introdusi in panel-urile de input. Deoarece fiecare repartitie are parametrii sai separati, am utilizat conditionalPanels, specifice partii de UI din aplicatiile Shiny.

```
X_data <- reactive({
  req(input$n, input$rep)  # Asigura-te ca n si rep sunt furnizate
  switch(input$rep,
         "norm" = rnorm(input$n, mean = input$mu, sd = input$sigma),
         "exp" = rexp(input$n, rate = input$lambda),
         "pois" = rpois(input$n, lambda = input$lambda_pois),
         "binom" = rbinom(input$n, size = input$size, prob = input$prob))
})
```

Reprezentare si trasarea graficelor: Am ales sa se poata arata graficelor mai multor variabile aleatoare ce sunt cerute pentru fiecare repartitie de o data. Acesta a fost motivul pentru care am utilizat pachetul grid, dar si pentru care am utilizat in server variabile reactive, adica X, Suma si Suma_la_Patrat.



Dupa ce se genera X, prin repsectivele fucntii, se verifica ce variabile aleatoare au fost selectate de catre utilizator, urmand ca pentru fiecare in parte sa apara insasi valorile. Un caz mai special sunt $\sum_{i=1}^{n} X\,i$, $\sum_{i=1}^{n} X\,i^2$, deoarece acestea nu depindeau de X, deci trebuiau selectate separat. Din acest motiv, outputul fiind de tip renderPlot, daca se modifica m-ul pentru Sume, X-ul isi dadea iar render, motiv pentru care graficele se schimbau pentru un nr mic de valori. Solutia a fost sa utilizez variabile reactive.

```
Suma_data <- reactive({
  req(input$m)
  replicate(input$n, sum(switch(input$rep,
                                "norm" = rnorm(input$m, mean = input$mu, sd = input$sigma),
                                "exp" = rexp(input$m, rate = input$lambda),
                                "pois" = rpois(input$m, lambda = input$lambda_pois),
                                "binom" = rbinom(input$m, size = input$size, prob = input$prob))))
})

Suma_la_Patrat_data <- reactive({
  req(input$m)
  replicate(input$n, sum(switch(input$rep,
                                "norm" = rnorm(input$m, mean = input$mu, sd = input$sigma)^2,
                                "exp" = rexp(input$m, rate = input$lambda)^2,
                                "pois" = rpois(input$m, lambda = input$lambda_pois)^2,
                                "binom" = rbinom(input$m, size = input$size, prob = input$prob)^2)))
})
```

Apoi ce a mai ramas a fost utilizarea functiei ecdf, adica functia de repartitie empirica, si plasarea in plot a valorilor. Cu toate acestea, a trebuit sa apara o distinctie intre modul in care sunt reprezentate cdf-ul unei

variabile discrete si cdf-ul unei variabile continue. De asta apare separarea pe cazuri si tratarea diferita a respectivelor grafice.

```
if (input$rep %in% c("pois", "binom")) {
  cdf <- ecdf(data[[var]])
  plot(cdf, main = paste("Functia de repartitie pentru", var),
       xlab = "Valori", ylab = "Probabilitate cumulativa",
       col = "black", lwd = 2, pch = 16, ylim = c(0, 1))
  abline(h = 1, col = "red", lty = 3)
  abline(h = 0, col = "red", lty = 3)

} else {
  cdf <- ecdf(data[[var]])
  df <- data.frame(x = sort(data[[var]]), y = cdf(sort(data[[var]])))

  plot(df$x, df$y, type = "l", col = "black", lwd = 2,
       xlab = "Valori", ylab = "Probabilitate cumulativa",
       main = paste("Functia de repartitie pentru", var),
       xlim=range(df$x))
  abline(h = 1, col = "red", lty = 3)
  abline(h = 0, col = "red", lty = 3)
}
```

Asezare in pagina : Initial se genereaza o pagina de tip blank, urmand ca imediat sa fie selectata varianta default, adica repartitia normala standard cu variabila aleatoare X. Daca sunt selectate mai multe variabile aleatoare, atunci se utilizeaza o matrice de grafice cu un nr fix de linii si coloane in functie de cat de multe variabile aleatore sunt selectate. Dupa procesarea si trasarea lor in server, tot acolo sunt puse pe pozitiile lor. Pagina poate sa para ca se misca putin lent, dar asta este cauzata doar de specificatiile calculatorului, dar si de modul in care aplicatia a fost structurata. Daca as putea afisa un singur grafic intr-o instanta, atunci rander-urile ar fi mult mai rapide.

```
if (length(input$vars) == 0) {
  grid::grid.newpage()
  grid::grid.rect(gp = grid::gpar(col = "white"))  # Pagina goala
  return()
}

num_plots <- length(input$vars)
num_cols <- ceiling(sqrt(num_plots))  # Determina numarul de coloane
num_rows <- ceiling(num_plots / num_cols)  # Determina numarul de randuri

par(mfrow = c(num_rows, num_cols))  # Imparte fereastra in mai multe subgrafice
```

Dificultati majore in abordarea cerintei :

1) Necesitatea invatarii si utilizarii unui pachet de-a dreptul nou, cum este shiny.
2) Modul vag in care cerinta a fost structurata, deoarece se lasa loc de interpretare, existand 2 metode prin care aceasta ar fi putut rezolvata : aceasta prin generarea random si cea prin care se alege o secventa de numere si se utilizeaza functiile pnorm, pexp, ppois si pbinom, pentru generarea graficului, fiind necesara doar punerea in plot a acestora.
3) Distinctia dintre infatisarea cdf-urilor variabilelor discrete si continue. Motiv pentru care, desi am utilizat cdf, a trebuit sa abordez/ utilizez diferit plot-ul, la continue fiind necesar sa utilizez un data.frame si sa utilizez stilul line in plot.
4) Rerandarea tuturor graficelor la cea mai mica modificare de parametru, desi nu erau afectate de respectivul parametru.
5) Sintaxa ce functioneaza in Server, dar in UI nu, motiv pentru care am pierdut mult timp sa incerc sa rezolv (Exemplu general : „String” %in% input$vector).

Notiuni extra in afara de curs: Referitor la partea de scris/ de calculat, nu am utilizat nimic in plus, dar cand vine vorba de partea de R, am utilizat toate notiunile de pachet shiny, data.frame-uri, griduri si fucntia de repartitie empirica.

Surse de inspiratie:

https://shiny.posit.co/r/getstarted/shiny-basics/lesson1/

https://shiny.posit.co/r/getstarted/shiny-basics/lesson2/

https://shiny.posit.co/r/getstarted/shiny-basics/lesson3/

https://shiny.posit.co/r/getstarted/shiny-basics/lesson4/

https://en.wikipedia.org/wiki/Binomial_distribution

https://en.wikipedia.org/wiki/Normal_distribution

https://en.wikipedia.org/wiki/Poisson_distribution

https://en.wikipedia.org/wiki/Exponential_distribution