

PROBABILITATI SI STATISTICA

*3 Studii de Caz în
Probabilități și Statistică*

Echipa de Proiect:

Nume Student	Grupa	Rol
Alexandrescu Andra	232	Lider de echipă
Andruta Andra Mihaela	232	-
Perli Davide	232	-
Tunaru Ioana Alexandra	232	-

Universitatea din București

Facultatea de Matematică și Informatică

2025

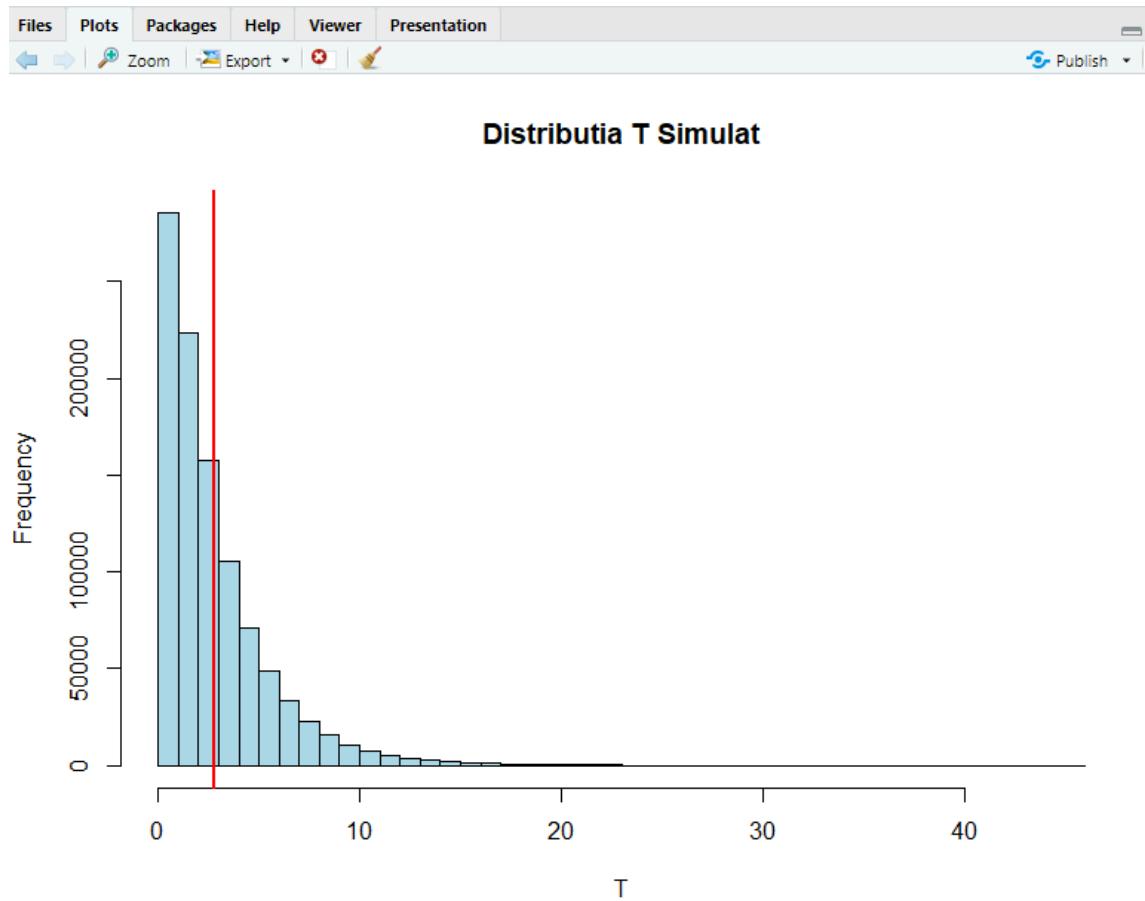
EXERCITIU 1

Perli Davide & Alexandrescu Andra

1) Construiți un algoritm in R care simuleaza 10^6 valori pentru v.a. T si in baza acestora aproximați $E(T)$. Reprezentati grafic intr-o manieră adecvata valorile obtinute pentru T. Ce puteți spune despre repartitia lui T?

Voi simula n etape cu rata λ_i corespunzatoare fiecarei etape si generez variabile aleatoare ce reprezinta distributia exponentiala $T_i \sim Exp(\lambda_i) \sim \lambda_i e^{-\lambda_i x}$ (e chiar functia de masa PMF – probability mass function), ele se vor aduna pe rand la un timp total T_total pentru fiecare simulare. Pentru o etapa curenta, generez o valoare pentru probabilitatea de continuare si verific sa fie mai mica ca α_i . In final calculez media valorilor simulate pentru a aproxima $E(T)$.

```
## a)
n <- 50 # vream 50 de etape pentru activitate
lambda <- runif(n, min=0.1, max=3) # generez 50 rate aleatoare pentru fiecare etapa cu valori intre 0.1 si 3 in vectorul lambda
# daca am gandit bine, un timp cu un Lambda subunitar va dura mai mult decat unul cu lambda supravantier
# practic daca T1 are lambda 0.5 => E(T1)=1/0.5=2, iar daca T2 are lambda 2.3 => E(T2)=1/2.3=0.43, adica E(T2)<E(T1)
alpha <- runif(n, min=0.1, max=0.9) # 50 de probabilitati de continuare cu valori mai mici ca 1(<100%) pentru fiecare etapa
nr_simulari <- 10^6 # fiecare simulare va face persoana A sa parcurga o activitate cu cate n etape
T_val <- numeric(nr_simulari) # vector pentru timpuri totalizati
for(j in 1:nr_simulari)
{
  T_total <- 0 # timpul total se reinicializeaza pentru inceputul fiecarei noi simulari
  i <- 1 # contor T_i
  while(i <= n) # n etape
  {
    T_i <- rexp(1, rate=lambda[i])# timpul etapei curente folosind rexp care calculeaza pmf-ul distributiei exponentiale in l
    # rexp(k, ...) genereaza k variabile aleatoare, dar nu le generez pe toate o data cu rexp(n, ...) deoarece fiecare are alt
    # T_i <- lambda[i]*exp(-lambda[i]*x)
    T_total <- T_total + T_i
    if(runif(1, min=0, max=1) < alpha[i])# conditie continuare la etapa i+1 deoarece runif(1) genereaza o valoare intre 0 si 1 care
    {
      i <- i + 1
    } else
    {
      break # oprire, nu trec la etapa urmatoare
    }
  }
  T_val[j] <- T_total
}
E_T_simulat <- mean(T_val) # media valorilor simulate pentru a determina E(T)
cat("Estimarea E(T) prin simulare:", E_T_simulat, "\n")
hist(T_val, col = "lightblue", main = "Distributia T Simulat", xlab = "T", breaks = 50) # plotarea distributiei valorilor simulate
# plotarea liniei rosii pentru valoarea exactă a lui E(T)
abline(v = E_T_simulat, col = "red", lwd = 2) # linia rosie pentru valoarea exactă a lui E(T)
```



```

n <- 50 # vreau 50 de etape pentru activitate
lambda <- runif(n, min=0.1, max=3) # generez 50 rate aleatoare pentru fiecare etapa cu valori intre
# 0.1 si 3 in vectorul lambda
# daca am gandit bine, un timp cu un lambda subunitar va dura mai mult decat unul cu lambda
# supraunitar
# practic daca T1 are lambda 0.5 => E(T1)=1/0.5=2, iar daca T2 are lambda 2.3 =>
# E(T2)=1/2.3=0.43, adica E(T2)<E(T1)
alpha <- runif(n, min=0.1, max=0.9) # 50 de probabilitati de continuare cu valori mai mici ca
# 1(=100%) pentru fiecare etapa de a trece la urmatoarea
nr_simulari <- 10^6 # fiecare simulare va face persoana A sa parcurga o activitate cu cate n etape

T_val <- numeric(nr_simulari) # vector pentru timpii totalizati

for(j in 1:nr_simulari)
{
  T_total <- 0 # timpul total se reinitializeaza pentru inceputul fiecarei noi simulari
  i <- 1 # contor T_i

  while(i <= n) # n etape
  {
    T_i <- rexp(1, rate=lambda[i])# timpul etapei curente folosind rexp care calculeaza pmf-ul
    distributiei exponentiale in lambda[i]
  }
}

```

```

# rexp(k, ...) genereaza k variabile aleatoare, dar nu le generez pe toate o data cu rexp(n, ...)
deoarece fiecare are alta rata si ar trebui sa fie incrementat i-ul
# T_i <- lambda[i]*e^(-lambda[i]*x)

T_total <- T_total + T_i

if(runif(1, min=0, max=1) < alpha[i])# conditie continuare la etapa i+1 deoarece runif(1) genereaza
o valoare intre 0 si 1 care e comparata cu probabilitate curenta de continuare alfa[i]
{
  i <- i + 1
} else
{
  break # oprire, nu trec la etapa urmatoare
}
}

T_val[j] <- T_total
}

E_T_simulat <- mean(T_val) # media valorilor simulate pentru a determina E(T)

cat("Estimarea E(T) prin simulare:", E_T_simulat, "\n")
hist(T_val, col = "lightblue", main = "Distributia T Simulat", xlab = "T", breaks = 50) # plotarea
distributiei valorilor simulate pentru T
abline(v = E_T_simulat, col = "red", lwd = 2) # linia rosie pentru valoarea exactă a lui E(T)

```

2) Calculati valoarea exacta a lui $E(T)$.

Media timpului total T pentru toate etapele poate fi calculata ca

$$E(T) = P(\text{reaches next stage}) * \sum_{i=1}^n E(T_i) = P(\text{reaches next stage}) * \sum_{i=1}^n \frac{1}{\lambda_i}. \text{ Stiu ca } T_i \text{ are}$$

media $E(T_i) = \frac{1}{\lambda_i}$ din

Theorem: Let X be a random variable following an exponential distribution:

$$X \sim \text{Exp}(\lambda). \quad (1)$$

Then, the mean or expected value of X is

$$\mathbb{E}(X) = \frac{1}{\lambda}. \quad (2)$$

Proof: The expected value is the probability-weighted average over all possible values:

$$\mathbb{E}(X) = \int_{\mathcal{X}} x \cdot f_X(x) dx. \quad (3)$$

With the probability density function of the exponential distribution, this reads:

$$\begin{aligned} \mathbb{E}(X) &= \int_0^{+\infty} x \cdot \lambda \exp(-\lambda x) dx \\ &= \lambda \int_0^{+\infty} x \cdot \exp(-\lambda x) dx. \end{aligned} \quad (4)$$

Using the following anti-derivative

$$\int x \cdot \exp(-\lambda x) dx = \left(-\frac{1}{\lambda}x - \frac{1}{\lambda^2} \right) \exp(-\lambda x), \quad (5)$$

the expected value becomes

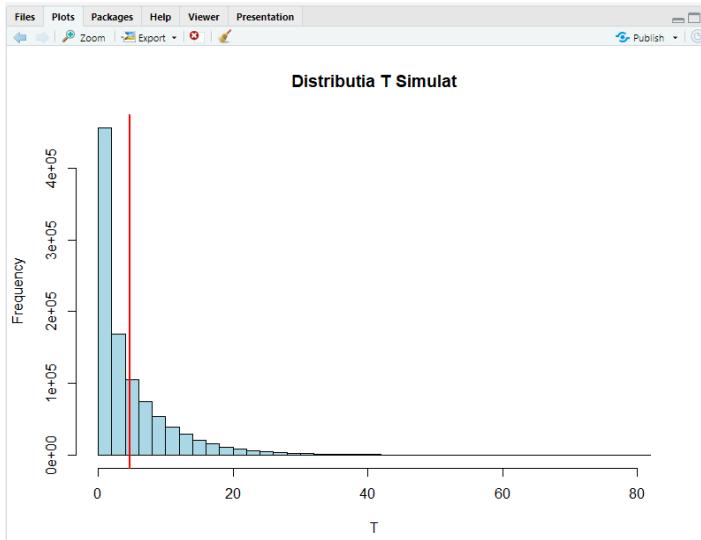
$$\begin{aligned} \mathbb{E}(X) &= \lambda \left[\left(-\frac{1}{\lambda}x - \frac{1}{\lambda^2} \right) \exp(-\lambda x) \right]_0^{+\infty} \\ &= \lambda \left[\lim_{x \rightarrow \infty} \left(-\frac{1}{\lambda}x - \frac{1}{\lambda^2} \right) \exp(-\lambda x) - \left(-\frac{1}{\lambda} \cdot 0 - \frac{1}{\lambda^2} \right) \exp(-\lambda \cdot 0) \right] \\ &= \lambda \left[0 + \frac{1}{\lambda^2} \right] \\ &= \frac{1}{\lambda}. \end{aligned} \quad (6)$$

Stiu ca $P(\text{reaches next stage } i + 1) = \prod_{j=1}^i \alpha_j$, adica e produsul probabilitatilor anterioare.

```

File Edit Code View Plots Session Build Debug Profile Tools Help
File Edit Code View Plots Session Build Debug Profile Tools Help
subject_1.R x
Source on Save | Go to file/function | Addins | Run | Source |
95
96 E_T_calculat <- sum((1/lambda)*P_reaches) # valoare exacta a lui E(T)
97
98 cat("Estimarea E(T) prin simulare:", E_T_simulat, "\n")
99 cat("Valoarea exactă a lui E(T):", E_T_calculat, "\n")
100 hist(T_val, col = "lightblue", main = "distributia T Simulat", xlab = "T", breaks = 50) # plotarea distributiei valorilor simulati
101 abline(v = E_T_calculat, col = "red", lwd = 2) # linia rosie pentru valoarea exactă a lui E(T)
102
103
104
105
106
107 # c)
108 n < 50 # n etape
109 lambda <- runif(n, min=0.1, max=3)
110 alpha <- runif(n, min=0.1, max=0.9)
111 nr_simulari <- 10^6 # fiecare simulare va face persoana A sa parcurga o activitate cu cate n etape
112
113 T_val <- numeric(nr_simulari) # vector pentru timpul totalizat
114 finalizat_count <- 0 # count pt etape finalizate
115
116 for(j in 1:nr_simulari)
117 {
118   for(i in 1:n) # fiecare etapa
119   {
120     T_total <- 0 # timpul total se reinitializeaza pentru inceputul fiecarei noi simulari
121     i <- 1 # contor T_i
122     while(i <= n) # n etape
123     {
124       i <=
125       if(i == 1)
126       {
127         P_reaches[i] <- 1 # prob de a ajunge la prima etapa este mereu 1
128       }
129       else
130       {
131         P_reaches[i] <- P_reaches[i-1]*alpha[i-1] # produsul probabilitatilor anterioare
132       }
133     }
134   }
135   E_T_calculat <- sum((1/lambda)*P_reaches) # valoare exacta a lui E(T)
136
137 cat("Estimarea E(T) prin simulare:", E_T_simulat, "\n")
138 cat("Valoarea exactă a lui E(T):", E_T_calculat, "\n")
139
140 hist(T_val, col = "lightblue", main = "distributia T Simulat", xlab = "T", breaks = 50) # plotarea distributiei valorilor simulati
141 abline(v = E_T_calculat, col = "red", lwd = 2) # linia rosie pentru valoarea exactă a lui E(T)
142

```



```

n <- 50 # n etape
lambda <- runif(n, min=0.1, max=3)
alpha <- runif(n, min=0.1, max=0.9)
nr_simulari <- 10^6 # fiecare simulare va face persoana A sa parcurga o activitate cu cate n etape

T_val <- numeric(nr_simulari) # vector pentru timpii totalizati

for(j in 1:nr_simulari)
{
  T_total <- 0 # timpul total se reinitializeaza pentru inceputul fiecarei noi simulari
  i <- 1 # contor T_i

  while(i <= n) # n etape
  {
    T_i <- rexp(1, rate=lambda[i])# timpul etapei curente folosind rexp care calculeaza pmf-ul
    distributiei exponentiale in lambda[i]
    # rexp(k, ...) genereaza k variabile aleatoare, dar nu le generez pe toate o data cu rexp(n, ...)
    deoarece fiecare are alta rata si ar trebui sa fie incrementat i-ul
    # T_i <- lambda[i]*e^(-lambda[i]*x)

    T_total <- T_total + T_i

    if(runif(1, min=0, max=1) < alpha[i])# conditie continuare la etapa i+1 deoarece runif(1) genereaza
    o valoare intre 0 si 1 care e comparata cu probabilitate curenta de continuare alfa[i]
    {
      i <- i + 1
    } else
    {
      break # oprire, nu trec la etapa urmatoare
    }
  }
}

```

```

T_val[j] <- T_total
}

E_T_simulat <- mean(T_val) # media valorilor simulate pentru a determina E(T)

P_reaches <- numeric(n)
for (i in 1:n) # calculez probabilitatea de a ajunge la fiecare etapa
{
  if (i == 1)
  {
    P_reaches[i] <- 1 # prob de a ajunge la prima etapa este mereu 1
  } else
  {
    P_reaches[i] <- P_reaches[i-1]*alpha[i-1] # produsul probabilitatilor anterioare
  }
}

E_T_calculat <- sum((1/lambda)*P_reaches) # valoare exacta a lui E(T)

cat("Estimarea E(T) prin simulare:", E_T_simulat, "\n")
cat("Valoarea exactă a lui E(T):", E_T_calculat, "\n")
hist(T_val, col = "lightblue", main = "Distributia T Simulat", xlab = "T", breaks = 50) # plotarea
distributiei valorilor simulate pentru T
abline(v = E_T_calculat, col = "red", lwd = 2) # linia rosie pentru valoarea exactă a lui E(T)

```

3) Aproximati probabilitatea ca persoana A sa finalizeze activitatea.

Probabilitatea ca persoana A sa finalizeze activitatea, adica sa fie parcurse toate n etapele, este de fapt produsul probabilitatilor de continuare α_i pentru fiecare etapa. E datorita ca fiecare etapa este un eveniment independent (timpul petrecut la o etapa i pentru terminarea etapei nu afecteaza timpul altel etape i+1), iar finalizarea activitatii depinde de succesul in fiecare dintre aceste etape (starea viitoare (decizia de a continua) depinde doar de starea curenta (finalizarea etapei) si nu de cum s-a ajuns acolo d.p.d.v. al timpilor care nu depind unul de celalalt).

Definiția formală a independenței: 2 evenimente A și B sunt **independente** \iff

$$P(A \cap B) = P(A) \cdot P(B). \quad (6)$$

Deci $P(finalizare_n_etape) = \prod_{i=1}^n \alpha_i$

Screenshot of RStudio showing an R script named "subject_1.R" and its corresponding console output.

```

150 for (i in 1:n) # calculez probabilitatea de a ajunge la fiecare etapa
151 [
152   if (i == 1)
153   {
154     P_reaches[i] <- 1 # prob de a ajunge la prima etapa este mereu 1
155   } else
156   {
157     P_reaches[i] <- P_reaches[i-1]*alpha[i-1] # produsul probabilitatilor anterioare
158   }
159 ]
160
161 E_T_calculat <- sum((1/lambda)*P_reaches) # valoare exactă a lui E(T)
162
163 cat("Estimarea E(T) prin simulare:", E_T_simulat, "\n")
164 cat("Valoarea exactă a lui E(T):", E_T_calculat, "\n")
165 hist(T_val, col = "lightblue", main = "distributia T Simulat", xlab = "T", breaks = 50) # plotarea distributiei valorilor simulati
166 abline(v = E_T_calculat, col = "red", lwd = 2) # linia rosie pentru valoarea exactă a lui E(T)
167
168 P_finalizare_simulata <- finalizat_count/nr_simulari
169 P_finalizare_exacta <- prod(alpha)
170
171 cat("Probabilitatea de finalizare in simulare:", P_finalizare_simulata, "\n")
172 cat("Probabilitatea exactă de finalizare:", P_finalizare_exacta, "\n") # va da o valoare foarte mică 3.720195x10^-18, adică 0.0...
173
174
175
176
177
178 # d)
179 ]

```

Console Output:

```

R 4.4.1 . ~/ ◊
+ {
+   P_reaches[i] <- P_reaches[i-1]*alpha[i-1] # produsul probabilitatilor anterioare
+ }
>
> E_T_calculat <- sum((1/lambda)*P_reaches) # valoare exactă a lui E(T)
>
> cat("Estimarea E(T) prin simulare:", E_T_simulat, "\n")
Estimarea E(T) prin simulare: 2.843184
> cat("Valoarea exactă a lui E(T):", E_T_calculat, "\n")
Valoarea exactă a lui E(T): 2.84684
> hist(T_val, col = "lightblue", main = "distributia T simulat", xlab = "T", breaks = 50) # plotarea distributiei valorilor simulate pentru T
> abline(v = E_T_calculat, col = "red", lwd = 2) # linia rosie pentru valoarea exactă a lui E(T)
>
> P_finalizare_simulata <- finalizat_count/nr_simulari
> P_finalizare_exacta <- prod(alpha)
>
> cat("Probabilitatea de finalizare in simulare:", P_finalizare_simulata, "\n")
Probabilitatea de finalizare in simulare: 0
> cat("Probabilitatea exactă de finalizare:", P_finalizare_exacta, "\n") # va da o valoare foarte mică 3.720195x10^-18, adică 0.0...
03720195
Probabilitatea exactă de finalizare: 1.01642e-17

```

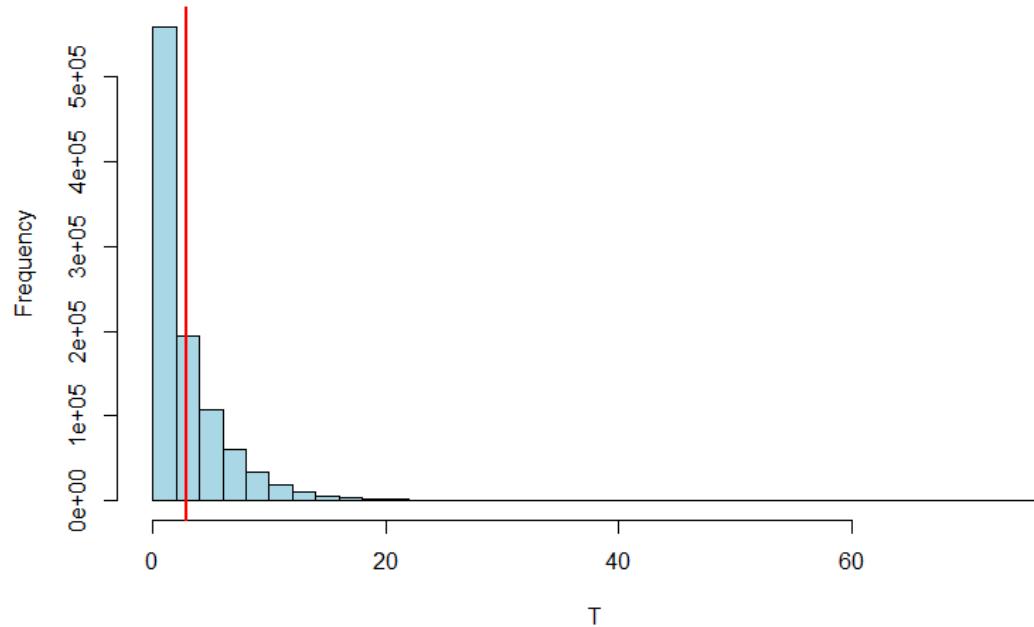
R ▾ Global Environment ▾

values	
alpha	num [1:50] 0.589 0.382 0.376 0.249 0.683 ...
E_T_calculat	2.84684042735318
E_T_simulat	2.84318388215253
finalizat_count	0
i	50L
j	1000000L
lambda	num [1:50] 1.523 0.319 2.553 0.523 2.215 ...
n	50
nr_simulari	1e+06
P_finalizare_exacta	1.01642011043854e-17
P_finalizare_simulata	0
P_reaches	num [1:50] 1 0.589 0.2252 0.0847 0.0211 ...
T_i	3.92368627660257
T_total	4.28769890252448
T_val	Large numeric (1000000 elements, 8 MB)

Files Plots Packages Help Viewer Presentation

Zoom Export Publish

Distributia T Simulat



```

n <- 50 # n etape
lambda <- runif(n, min=0.1, max=3)
alpha <- runif(n, min=0.1, max=0.9)
nr_simulari <- 10^6 # fiecare simulare va face persoana A sa parcurga o activitate cu cate n etape
  
```

```

T_val <- numeric(nr_simulari) # vector pentru timpii totalizati
finalizat_count <- 0 # count pt etape finalizate
  
```

```

for(j in 1:nr_simulari)
{
  T_total <- 0 # timpul total se reinitializeaza pentru inceputul fiecarei noi simulari
  i <- 1 # contor T_i

  while(i <= n) # n etape
  {
    T_i <- rexp(1, rate=lambda[i])# timpul etapei curente folosind rexp care calculeaza pmf-ul
    distributiei exponentiale in lambda[i]
    # rexp(k, ...) genereaza k variabile aleatoare, dar nu le generez pe toate o data cu rexp(n, ...)
    deoarece fiecare are alta rata si ar trebui sa fie incrementat i-ul
    # T_i <- lambda[i]*e^(-lambda[i]*x)

    T_total <- T_total + T_i

    if(runif(1, min=0, max=1) < alpha[i])# conditie continuare la etapa i+1 deoarece runif(1) genereaza
    o valoare intre 0 si 1 care e comparata cu probabilitate curenta de continuare alfa[i]
    {
      i <- i + 1
    } else
    {
      break # oprire, nu trec la etapa urmatoare
    }
  }

  T_val[j] <- T_total

  if (i > n)
  {
    finalizat_count <- finalizat_count + 1
  }
}

E_T_simulat <- mean(T_val) # media valorilor simulate pentru a determina E(T)

P_reaches <- numeric(n)
for (i in 1:n) # calculez probabilitatea de a ajunge la fiecare etapa
{
  if (i == 1)
  {
    P_reaches[i] <- 1 # prob de a ajunge la prima etapa este mereu 1
  } else
  {
    P_reaches[i] <- P_reaches[i-1]*alpha[i-1] # produsul probabilitatilor anterioare
  }
}

E_T_calculat <- sum((1/lambda)*P_reaches) # valoare exacta a lui E(T)

```

```

cat("Estimarea E(T) prin simulare:", E_T_simulat, "\n")
cat("Valoarea exactă a lui E(T):", E_T_calculat, "\n")
hist(T_val, col = "lightblue", main = "Distributia T Simulat", xlab = "T", breaks = 50) # plotarea
distributiei valorilor simulate pentru T
abline(v = E_T_calculat, col = "red", lwd = 2) # linia rosie pentru valoarea exactă a lui E(T)

P_finalizare_simulata <- finalizat_count/nr_simulari
P_finalizare_exacta <- prod(alpha)

cat("Probabilitatea de finalizare in simulare:", P_finalizare_simulata, "\n")
cat("Probabilitatea exacta de finalizare:", P_finalizare_exacta, "\n") # va da o valoare foarte mica
3.720195×10^-18, adica 0.00...03720195

```

4) Aproximați probabilitatea ca persoana A sa finalizeze activitatea într-un timp mai mic sau egal cu σ .

Dupa finalizarea etapei i , persoana A va trece la etapa $i+1$ cu o probabilitate α_i sau se va opri cu complementul probabilitatii, adica $1 - \alpha_i$.

In realitate, probabilitatea ca timpul total de finalizare este mai mic sau egal cu σ se exprima prin functia de distributie cumulativă (CDF – cumulative distribution function) a distributiei exponentiale, iar timpul necesar finalizarii unei singure etape i este o variabila aleatoare T_i distribuita exponential cu parametru λ_i , adica voi folosi $F(t) = P(x \leq t) = 1 - e^{-\lambda_i t}$ pentru timpul unei singure etape.

Daca timpul total de finalizare T este suma timpilor pentru fiecare etapa, iar fiecare T_i este independent, atunci CDF-ul pentru suma acestor variabile aleatoare nu este direct exponential, dar poate fi calculat prin metode numerice sau simulari. Deci simulez si retin numarul de variabile T_i cu timpul $\leq \sigma$ si ma folosesc de logica asta: $P(T \leq \sigma) = \frac{\text{nr simulari timp total } T \leq \sigma}{\text{nr total simulari}}$.

Funcția de repartiție corespunzătoare este $F(x) = \int_{-\infty}^x f(t) dt = 0$, dacă $x \leq 0$ și, dacă $x > 0$, atunci

$$F(x) = \int_{-\infty}^x f(t) dt = \int_0^x \lambda e^{-\lambda t} dt = \lambda \frac{e^{-\lambda t}}{-\lambda} \Big|_{x=0}^{t=x},$$

deci (vezi și relațiile din cazul distribuției geometrice), pentru orice $x > 0$ avem

$$F(x) = 1 - e^{-\lambda x} \quad \text{sau, echivalent,} \quad \mathbb{P}(X \geq x) = e^{-\lambda x}. \quad (\text{III.1.18})$$

Practic cele 2 etape sunt:

- 1) Pentru fiecare simulare, verificati daca timpul total este mai mic sau egal cu σ
- 2) Proportia de simulari in care timpul total este mai mic sau egal cu σ

subject_1.R

```

226  {
227     P_reaches[i] <- 1 # prob de a ajunge la prima etapa este mereu 1
228  } else
229  {
230     P_reaches[i] <- P_reaches[i-1]*alpha[i-1] # produsul probabilitatilor anterioare
231  }
232 }
233
234 E_T_calculat <- sum((1/lambda)*P_reaches) # valoare exactă a lui E(T)
235
236 cat("Estimarea E(T) prin simulare:", E_T_simulat, "\n")
237 cat("Valoarea exactă a lui E(T):", E_T_calculat, "\n")
238 hist(T_val, col = "lightblue", main = "Distributia T simulat", xlab = "T", breaks = 50) # plotarea distributiei valorilor simulate
239 abline(v = E_T_calculat, col = "red", lwd = 2) # linia rosie pentru valoarea exactă a lui E(T)
240
241 P_sigma_simulata <- count_sigma/nr_simulari
242
243 cat("Probabilitatea de a termina intr-un timp <= sigma:", P_sigma_simulata, "\n")
244
245
246
247
248 # e)
249
250 n <- 3 # scad numarul de etape
251 lambda <- runif(n, min=0.1, max=3)
252 alpha <- runif(n, min=0.1, max=0.9)
253 nr_simulari <- 10^6 # fiecare simulare va face persoana A sa parcurga o activitate cu cate n etape
254
255

```

244:1 (Top Level) ↴ R Script

Console Terminal × Background Jobs ×

```

R 4.4.1 . ~/~
+ if (i == 1)
+ {
+   P_reaches[i] <- 1 # prob de a ajunge la prima etapa este mereu 1
+ } else
+ {
+   P_reaches[i] <- P_reaches[i-1]*alpha[i-1] # produsul probabilitatilor anterioare
+ }
>
> E_T_calculat <- sum((1/lambda)*P_reaches) # valoare exactă a lui E(T)
>
> cat("Estimarea E(T) prin simulare:", E_T_simulat, "\n")
Estimarea E(T) prin simulare: 4.03713
> cat("Valoarea exactă a lui E(T):", E_T_calculat, "\n")
Valoarea exactă a lui E(T): 4.037634
> hist(T_val, col = "lightblue", main = "Distributia T simulat", xlab = "T", breaks = 50) # plotarea distributiei valorilor simulate pentru T
> abline(v = E_T_calculat, col = "red", lwd = 2) # linia rosie pentru valoarea exactă a lui E(T)
>
> P_sigma_simulata <- count_sigma/nr_simulari
>
> cat("Probabilitatea de a termina intr-un timp <= sigma:", P_sigma_simulata, "\n")
Probabilitatea de a termina intr-un timp <= sigma: 0.694628
> |

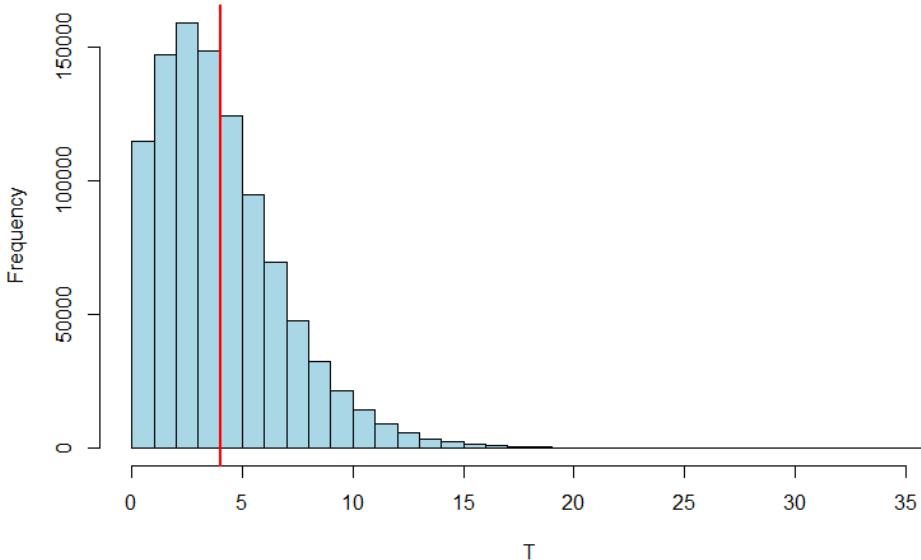
```

R Global Environment

values	
alpha	num [1:50] 0.821 0.762 0.691 0.3 0.145 ...
count_sigma	694628
E_T_calculat	4.0376340063467
E_T_simulat	4.03712954175259
i	50L
j	1000000L
lambda	num [1:50] 0.506 1.097 0.639 1.764 1.756 ...
n	50
nr_simulari	1e+06
P_reaches	num [1:50] 1 0.821 0.626 0.433 0.13 ...
P_sigma_simulata	0.694628
sigma	5
T_i	0.794781466427528
T_total	1.72676558711283
T_val	Large numeric (1000000 elements, 8 MB)



Distributia T Simulat



```
n <- 50 # n etape
lambda <- runif(n, min=0.1, max=3)
alpha <- runif(n, min=0.1, max=0.9)
nr_simulari <- 10^6 # fiecare simulare va face persoana A sa parcurga o activitate cu cate n etape
sigma <- 5
```

```
T_val <- numeric(nr_simulari) # vector pentru timpii totalizati
count_sigma <- 0 # contor pentru simularile cu T_total <= sigma
```

```
for(j in 1:nr_simulari)
{
  T_total <- 0 # timpul total se reinitializeaza pentru inceputul fiecarei noi simulari
  i <- 1 # contor T_i

  while(i <= n) # n etape
  {
    T_i <- rexp(1, rate=lambda[i])# timpul etapei curente folosind rexp care calculeaza pmf-ul
    distributiei exponentiale in lambda[i]
    # rexp(k, ...) genereaza k variabile aleatoare, dar nu le generez pe toate o data cu rexp(n, ...)
    deoarece fiecare are alta rata si ar trebui sa fie incrementat i-ul
    # T_i <- lambda[i]*e^(-lambda[i]*x)

    T_total <- T_total + T_i

    if(runif(1, min=0, max=1) < alpha[i])# conditie continuare la etapa i+1 deoarece runif(1) genereaza
    o valoare intre 0 si 1 care e comparata cu probabilitate curenta de continuare alfa[i]
```

```

{
  i <- i + 1
} else
{
  break # oprire, nu trec la etapa urmatoare
}
}

T_val[j] <- T_total

if (T_total <= sigma)
{
  count_sigma <- count_sigma + 1
}
}

E_T_simulat <- mean(T_val) # media valorilor simulate pentru a determina E(T)

P_reaches <- numeric(n)
for (i in 1:n) # calculez probabilitatea de a ajunge la fiecare etapa
{
  if (i == 1)
  {
    P_reaches[i] <- 1 # prob de a ajunge la prima etapa este mereu 1
  } else
  {
    P_reaches[i] <- P_reaches[i-1]*alpha[i-1] # produsul probabilitatilor anterioare
  }
}

E_T_calculat <- sum((1/lambda)*P_reaches) # valoare exacta a lui E(T)

cat("Estimarea E(T) prin simulare:", E_T_simulat, "\n")
cat("Valoarea exactă a lui E(T):", E_T_calculat, "\n")
hist(T_val, col = "lightblue", main = "Distributia T Simulat", xlab = "T", breaks = 50) # plotarea
distributiei valorilor simulate pentru T
abline(v = E_T_calculat, col = "red", lwd = 2) # linia rosie pentru valoarea exactă a lui E(T)

P_sigma_simulata <- count_sigma/nr_simulari

cat("Probabilitatea de a termina intr-un timp <= sigma:", P_sigma_simulata, "\n")

```

5) 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 repartitia acestor timpi de finalizare a activitatii?

Practic in T_val_valid avem rezultatul timpului total din fiecare simulare care a reusit sa parcurga toate cele n etape (adica sa se termine activitatea), deci minimul timpul minim si maxim de finalizarii a tuturor celor n etape inseamna sa folosesc min() si max() pe vectorul T_val_valid.

Dureaza foarte mult sa fie rulat dintr-un motiv sau altul.

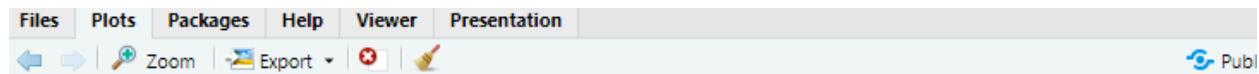
```

245
246
247
248 # e)
249
250 n <- 3 # scad numarul de etape
251 lambda <- runif(n, min=0.1, max=3)
252 alpha <- runif(n, min=0.1, max=0.9)
253 nr_simulari <- 10^6 # fiecare simulare va face persoana A sa parcurga o activitate cu cate n etape
254
255 T_val_valid <- numeric(0) # vector pentru a stoca valorile timpului total T doar pentru simulari care au terminat toate cele
256
257 for(j in 1:nr_simulari)
258 {
259   T_total <- 0 # timpul total se reinitializeaza pentru inceputul fiecarei noi simulari
260   i <- 1 # contor T_i
261
262   while(i <= n) # n etape
263   {
264     T_i <- rexp(1, rate=lambda[i])# timpul curenta folosind rexp care calculeaza pmf-ul distributiei exponentiale in l
265     # rexp(k, ...) genereaza k variabile aleatoare, dar nu le generez pe toate o data cu rexp(n, ...) deoarece fiecare are al
266     # T_i <- lambda[i]*e^(-lambda[i]*x)
267
268     T_total <- T_total + T_i
269
270     if(runif(1, min=0, max=1) < alpha[i])# conditie continua la etapa i+1 daca randul generat este intre 0 si
271   }
298:1 [Top Level] R Sc
Console Terminal ✘ Background Jobs ✘
R 4.4.1 . ~/ ↴ R Sc
+ }
+
+ if(i == n + 1) # s-au terminat toate cele n etape
+ {
+   T_val_valid <- c(T_val_valid, T_total)
+ }
+ }
>
> if(length(T_val_valid) > 0) # exista valori in T_val_valid
+ {
+   T_min_finalizare <- min(T_val_valid)
+   T_max_finalizare <- max(T_val_valid)
+
+   cat("Timpul minim de finalizare a activitatii:", T_min_finalizare, "\n")
+   cat("Timpul maxim de finalizare a activitatii:", T_max_finalizare, "\n")
+
+   hist(T_val_valid, col = "lightblue")
+ } else
+ {
+   cat("Nu exista simulari valide in care persoana A finalizeaza activitatea\n")
+ }
Timpul minim de finalizare a activitatii: 0.01283022
Timpul maxim de finalizare a activitatii: 17.6423

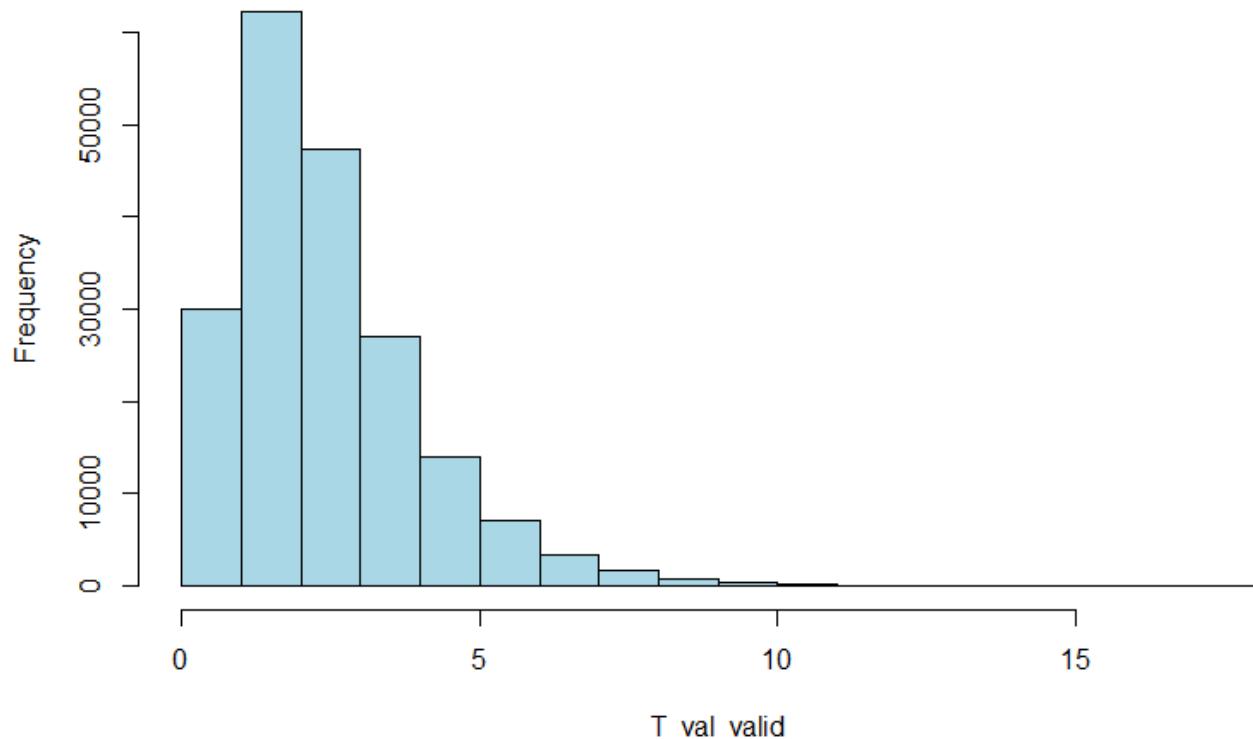
```

R | Global Environment ▾

values	
alpha	num [1:3] 0.662 0.857 0.342
i	1
j	1000000L
lambda	num [1:3] 1.422 2.75 0.725
n	3
nr_simulari	1e+06
T_i	1.81657426653424
T_max_finalizare	17.6422969707506
T_min_finalizare	0.0128302193012451
T_total	1.81657426653424
T_val_valid	Large numeric (194741 elements, 1.6 MB)



Histogram of T_val_valid



```

n <- 3 # scad numarul de etape
lambda <- runif(n, min=0.1, max=3)
alpha <- runif(n, min=0.1, max=0.9)
nr_simulari <- 10^6 # fiecare simulare va face persoana A sa parcurga o activitate cu cate n etape

```

```

T_val_valid <- numeric(0) # vector pentru a stoca valorile timpului total T doar pentru simulari care
au terminat toate cele n etape

for(j in 1:nr_simulari)
{
  T_total <- 0 # timpul total se reinitializeaza pentru inceputul fiecarei noi simulari
  i <- 1 # contor T_i

  while(i <= n) # n etape
  {
    T_i <- rexp(1, rate=lambda[i])# timpul etapei curente folosind rexp care calculeaza pmf-ul
    distributiei exponentiale in lambda[i]
    # rexp(k, ...) genereaza k variabile aleatoare, dar nu le generez pe toate o data cu rexp(n, ...)
    deoarece fiecare are alta rata si ar trebui sa fie incrementat i-ul
    # T_i <- lambda[i]*e^(-lambda[i]*x)

    T_total <- T_total + T_i

    if(runif(1, min=0, max=1) < alpha[i])#conditie continuare la etapa i+1 deoarece runif(1) genereaza
    o valoare intre 0 si 1 care e comparata cu probabilitate curenta de continuare alfa[i]
    {
      i <- i + 1
    } else
    {
      break # oprire, nu trec la etapa urmatoare
    }
  }

  if(i == n + 1) # s-au terminat toate cele n etape
  {
    T_val_valid <- c(T_val_valid, T_total)
  }
}

if(length(T_val_valid) > 0) # exista valori in T_val_valid
{
  T_min_finalizare <- min(T_val_valid)
  T_max_finalizare <- max(T_val_valid)

  cat("Timpul minim de finalizare a activitatii:", T_min_finalizare, "\n")
  cat("Timpul maxim de finalizare a activitatii:", T_max_finalizare, "\n")

  hist(T_val_valid, col = "lightblue")
} else
{
  cat("Nu exista simulari valide in care persoana A finalizeaza activitatea\n")
}

```

6) Aproximati probabilitatea ca persoana A sa se opreasca din lucru inainte de etapa k , unde $1 < k \leq n$. Reprezentati grafic probabilitatile obtinute intr-o manieră corespunzatoare. Ce puteti spune despre repartitia probabilitatilor obtinute?

Formula pentru probabilitatea ca persoana A sa se opreasca inainte de etapa k este data de complementul probabilitatii ca activitatea sa fie finalizata pana la etapa $k - 1$, iar de la subpunctul 3)

se deduce $P(\text{finalizare}_{k-1} \text{ etape}) = \prod_{i=1}^{k-1} \alpha_i$, deci

$$P(\text{oprire}_{\text{etapa } k}) = 1 - P(\text{finalizare}_{k-1} \text{ etape}) = 1 - \prod_{i=1}^n \alpha_i$$

The screenshot shows the RStudio interface with two main panes: the code editor and the R console.

Code Editor:

```

328     {
329         opriri_inainte_de_etapa[k] <- opriri_inainte_de_etapa[k] + 1
330     }
331 }
332 }
333 probabilitati_oprire_simulate <- opriri_inainte_de_etapa/nr_simulari
335
336 for(k in 2:n)
337 {
338     p_finalizare_pana_la_k_minus_1 <- prod(alpha[1:(k-1)]) # prob de finalizare pana la etapa k-1
339     probabilitati_oprire_exacte[k] <- 1-p_finalizare_pana_la_k_minus_1 # prob de oprire inainte de etapa k
340 }
341
342 cat("Probabilitatile de oprire (simulata) inainte de etapa k:", probabilitati_oprire_simulate, "\n")
343 cat("Probabilitatile de oprire (exacta) inainte de etapa k:", probabilitati_oprire_exacte, "\n")
344
345 par(mfrow = c(1, 2)) # 2 grafice
346 plot(1:n, probabilitati_oprire_simulate, type = "b", col = "blue",
347       main = "Probabilitatea de oprire (simulata)",
348       xlab = "Etapa k", ylab = "Probabilitatea de oprire")
349 grid()
350
351 plot(1:n, probabilitati_oprire_exacte, type = "b", col = "red",
352       main = "Probabilitatea de oprire (exacta)",
353       xlab = "Etapa k", ylab = "Probabilitatea de oprire")
354 grid()
355
356

```

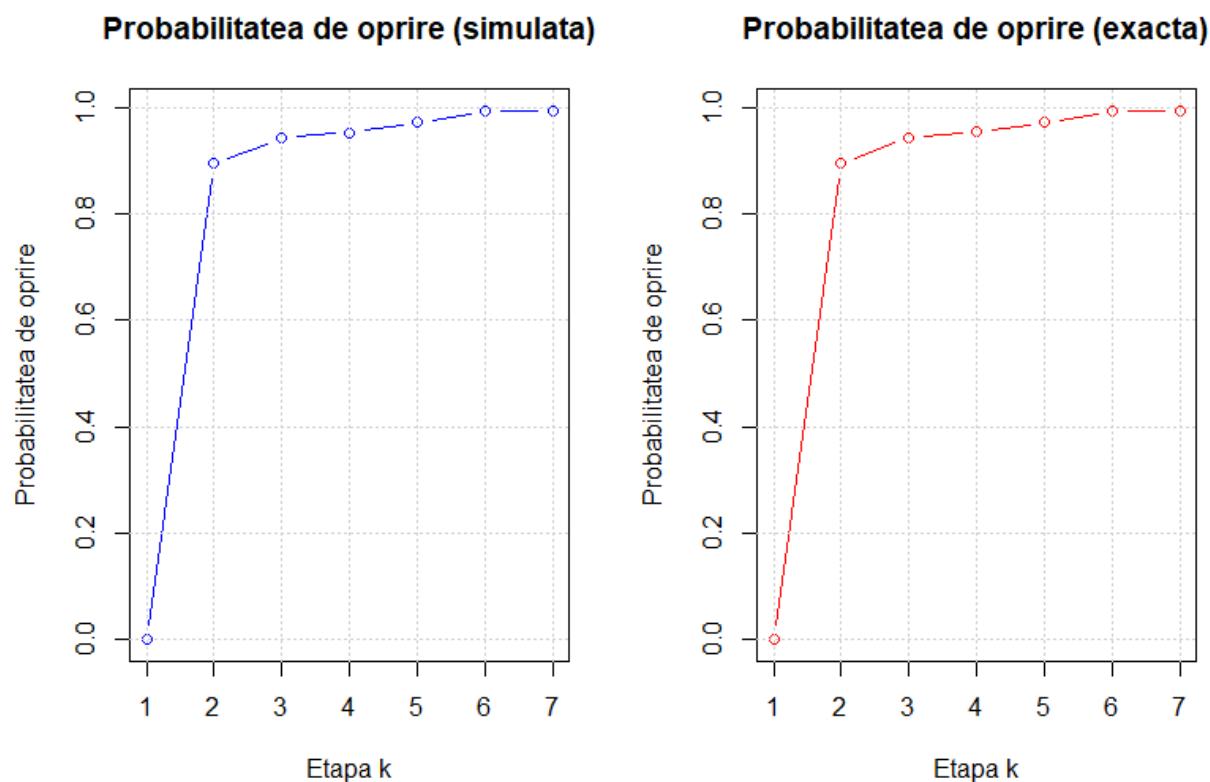
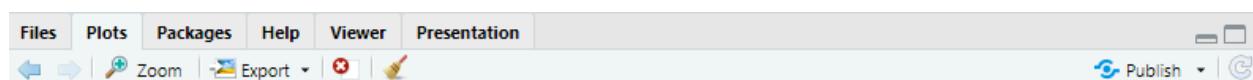
R Console:

```

> probabilitati_oprire_simulate <- opriri_inainte_de_etapa/nr_simulari
>
> for(k in 2:n)
+ {
+     p_finalizare_pana_la_k_minus_1 <- prod(alpha[1:(k-1)]) # prob de finalizare pana la etapa k-1
+     probabilitati_oprire_exacte[k] <- 1-p_finalizare_pana_la_k_minus_1 # prob de oprire inainte de etapa k
+ }
>
> cat("Probabilitatile de oprire (simulata) inainte de etapa k:", probabilitati_oprire_simulate, "\n")
Probabilitatile de oprire (simulata) inainte de etapa k: 0 0.893704 0.941919 0.952552 0.971435 0.992123 0.993922
> cat("Probabilitatile de oprire (exacta) inainte de etapa k:", probabilitati_oprire_exacte, "\n")
Probabilitatile de oprire (exacta) inainte de etapa k: 0 0.8936365 0.9420063 0.9525916 0.9715157 0.9920773 0.9938492
>
> par(mfrow = c(1, 2)) # 2 grafice
> plot(1:n, probabilitati_oprire_simulate, type = "b", col = "blue",
+       main = "Probabilitatea de oprire (simulata)",
+       xlab = "Etapa k", ylab = "Probabilitatea de oprire")
> grid()
>
> plot(1:n, probabilitati_oprire_exacte, type = "b", col = "red",
+       main = "Probabilitatea de oprire (exacta)",
+       xlab = "Etapa k", ylab = "Probabilitatea de oprire")
> grid()

```

values	
alpha	num [1:7] 0.106 0.545 0.817 0.601 0.278 ...
i	1
j	1000000L
k	7L
lambda	num [1:7] 1.407 1.222 0.962 2.115 2.395 ...
n	7
nr_simulari	1e+06
opriri_inainte_de_etapa	num [1:7] 0 893704 941919 952552 971435 ...
p_finalizare_pana_la_k...	0.00615082966312775
probabilitati_oprire_ex...	num [1:7] 0 0.894 0.942 0.953 0.972 ...
probabilitati_oprire_si...	num [1:7] 0 0.894 0.942 0.953 0.971 ...



```

n <- 7 # n etape
lambda <- runif(n, min=0.1, max=3)
alpha <- runif(n, min=0.1, max=0.9)
nr_simulari <- 10^6 # fiecare simulare va face persoana A sa parcurga o activitate cu cate n etape
  
```

```

opriri_inainte_de_etapa <- numeric(n) # vector pentru a numara opririle inainte de fiecare etapa
probabilitati_oprire_exacte <- numeric(n)
  
```

```

for(j in 1:nr_simulari)
{
  i <- 1
  while(i <= n)
  {
    if(runif(1, min=0, max=1) < alpha[i]) # conditie continuare la etapa i+1 deoarece runif(1) genereaza
    o valoare intre 0 si 1 care e comparata cu probabilitate curenta de continuare alfa[i]
    {
      i <- i + 1
    } else
    {
      break # oprire, nu trec la etapa urmatoare
    }
  }

  for(k in 1:n) # creste numarul de opriri pentru fiecare etapa la care s-a oprit
  {
    if(i < k)
    {
      opriri_inainte_de_etapa[k] <- opriri_inainte_de_etapa[k] + 1
    }
  }
}

probabilitati_oprire_simulate <- opriri_inainte_de_etapa/nr_simulari

for(k in 2:n)
{
  P_finalizare_pana_la_k_minus_1 <- prod(alpha[1:(k-1)]) # prob de finalizare pana la etapa k-1
  probabilitati_oprire_exacte[k] <- 1-P_finalizare_pana_la_k_minus_1 # prob de oprire inainte de
  etapa k
}

cat("Probabilitatile de oprire (simulata) inainte de etapa k:", probabilitati_oprire_simulate, "\n")
cat("Probabilitatile de oprire (exacta) inainte de etapa k:", probabilitati_oprire_exacte, "\n")

par(mfrow = c(1, 2)) # 2 grafice
plot(1:n, probabilitati_oprire_simulate, type = "b", col = "blue",
      main = "Probabilitatea de oprire (simulata)",
      xlab = "Etapa k", ylab = "Probabilitatea de oprire")
grid()

plot(1:n, probabilitati_oprire_exacte, type = "b", col = "red",
      main = "Probabilitatea de oprire (exacta)",
      xlab = "Etapa k", ylab = "Probabilitatea de oprire")
grid()

```

EXERCITIU 2

Aplicație Shiny pentru distribuții de variabile aleatoare și funcții personalizate

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

1. Echipa de exercițiu

- Andruță Andra Mihaela
 - Tunaru Ioana Alexandra
-

2. Introducere și Context

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

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

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

3. Cerințe și Obiective Principale

Proiectul răspunde următoarelor cerințe:

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

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

4. Aspecte Teoretice și Metodologice

4.1 Generarea de mostre din distribuții

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

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

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

4.2 Funcția de repartiție empirică (ECDF)

După ce generăm setul de date, putem obține repartiția empirică prin stat_ecdf (o resursă din ggplot2). Aceasta oferă o reprezentare de tip scară a frecvențelor cumulative, un mod intuitiv de a observa distribuția (sau transformarea) obținută. Astfel, utilizatorii pot vizualiza variațiile și se pot familiariza cu conceptul de ECDF – un concept-cheie în statistica inferențială.

4.3 Construirea interfeței cu Shiny

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

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

4.4 Funcții personalizate

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

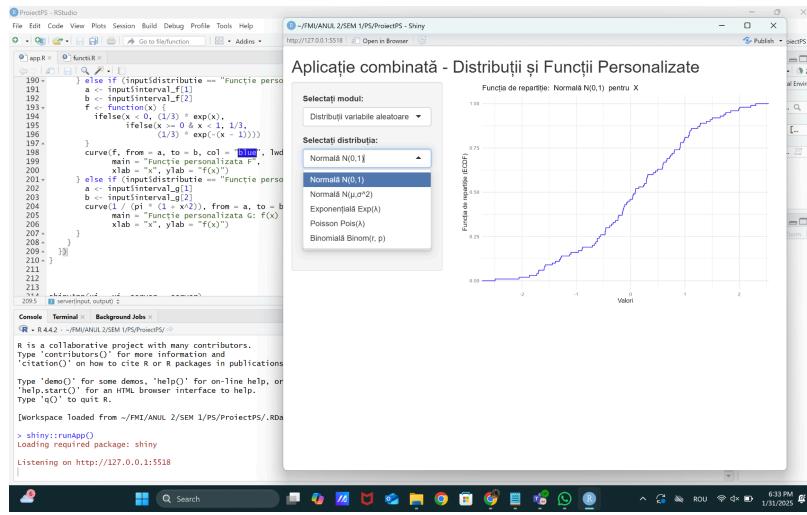
Scopul lor este de a oferi un cadru general prin care utilizatorul să poată vedea, într-un singur tab, mai multe feluri de funcții și să observe forma grafică, eventual discuții teoretice despre convergență, normalizare (integrarea pe suport), medie și varianță (dacă există).

5. Reprezentări Grafice și Elemente Interactive

5.1 Modul „Distribuții variabile aleatoare”

În acest modul, utilizatorul:

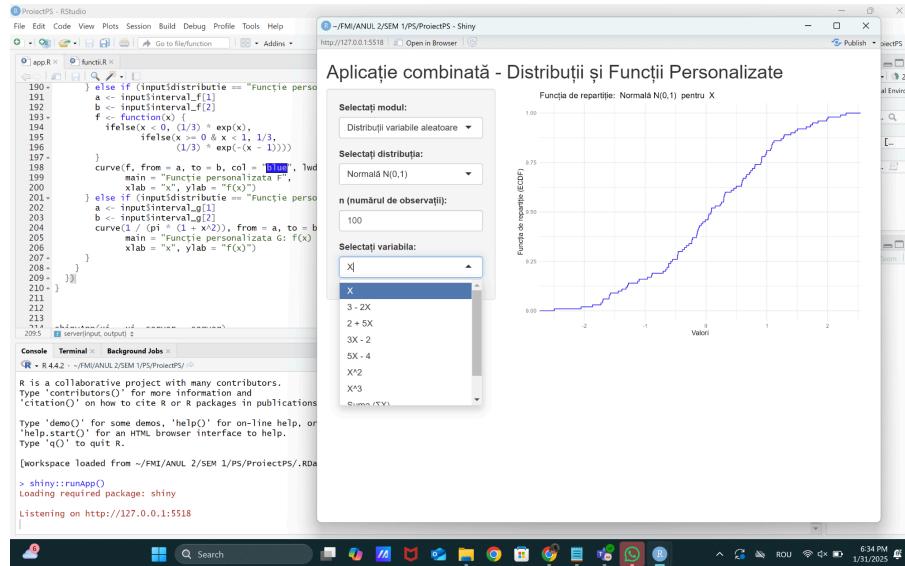
1. Selectează tipul distribuției: Normală (0,1), Normală (μ, σ), Exponențială (λ), Poisson (λ), Binomială (r, p).



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

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

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



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

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

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

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

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

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

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

- **Plotarea:**

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

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

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

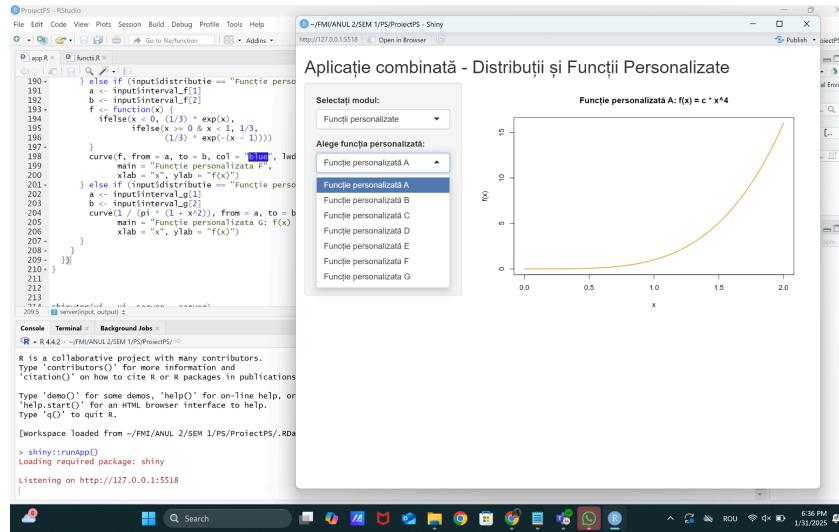
Alegerea funcției de repartiție empirică pentru ilustrare este foarte utilă pentru a vedea cum sunt distribuite valorile efectiv generate. De exemplu, la o distribuție Normală $N(0,1)$, ECDF-ul poate fi comparat cu forma teoretică a repartiției cumulative a Normalelor, iar dacă transformăm variabila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

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

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

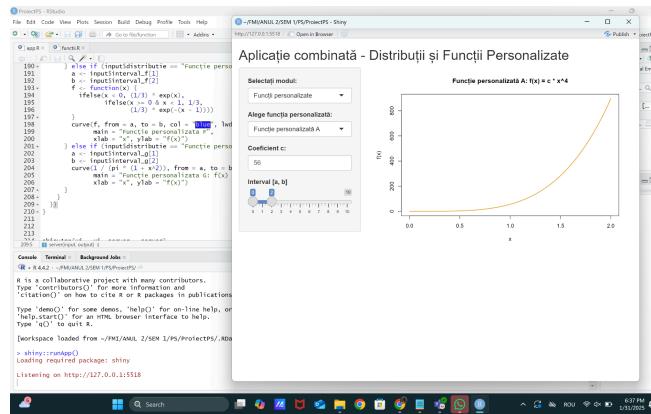
5.2 Modul „Funcții personalizate”

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



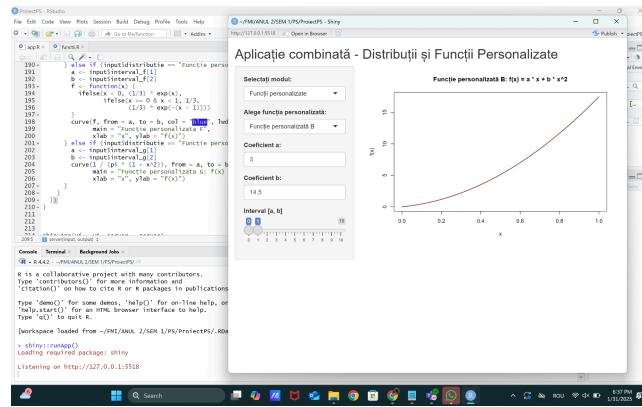
- Funcția A: $f(x) = c \cdot x^4$, pentru $x \in [0,2]$. Aplicația afișează un slider pentru a alege intervalul de plot (implicit $[0,2]$) și un input numeric pentru c.

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



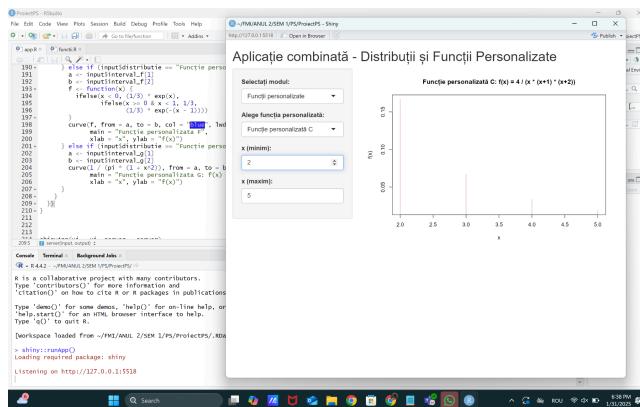
- Funcția B: $f(x) = ax + bx^2$, cu $x \in (0,1)$. Parametrii a și b sunt introdusi în UI drept numericInput, iar intervalul e $[0,1]$.

Plotarea se face tot prin curve().



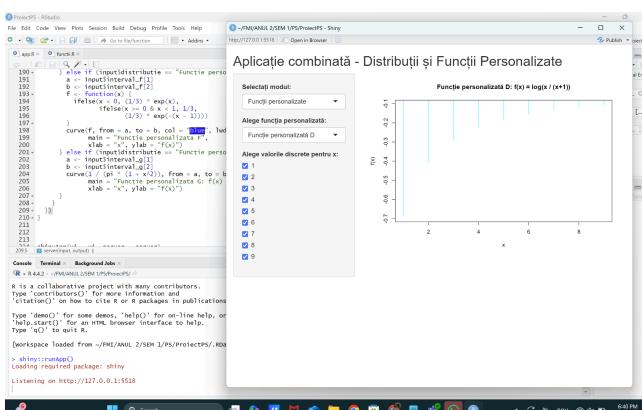
- Funcția C: $\frac{4}{[x(x+1)(x+2)]}$ pentru valori întregi de x.

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



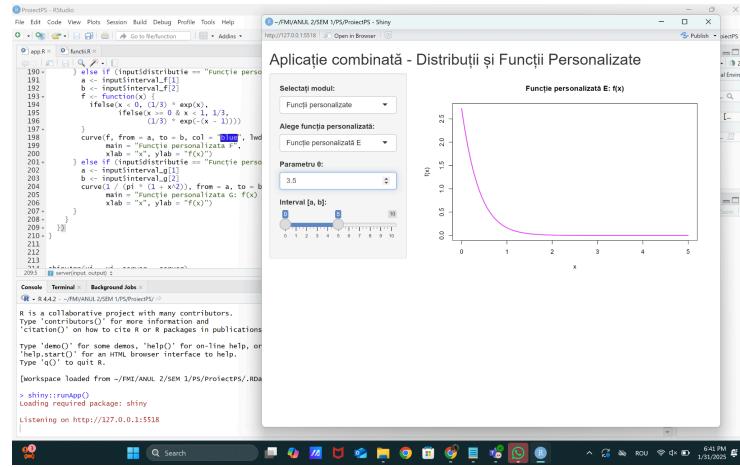
- Funcția D: $\log(\frac{x}{x(x+1)})$. Este tot o funcție discretă în codul nostru, căci am invitat utilizatorul să selecteze valori din {1,2,...,9}.

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



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

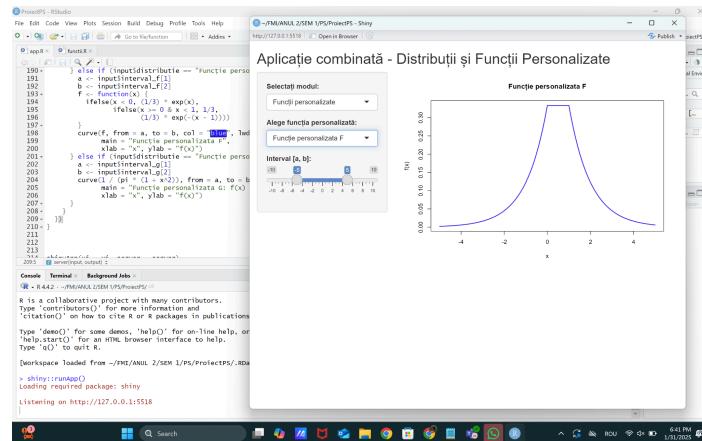
Aceasta este plotată cu `curve()`, parametrul θ fiind personalizabil.



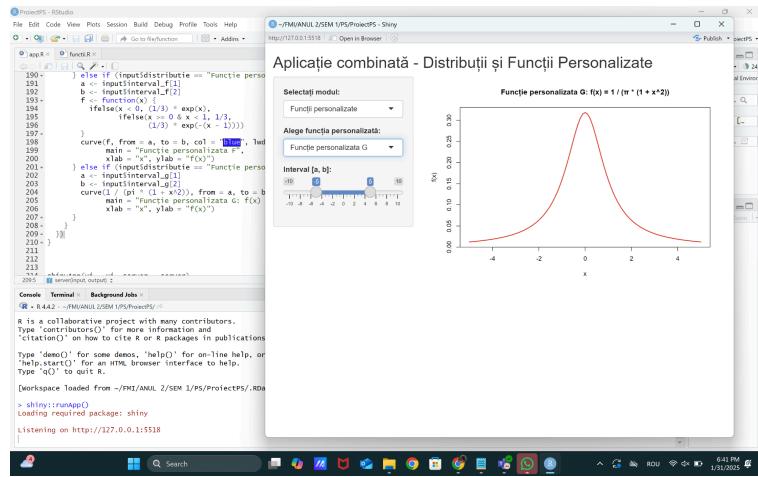
- Funcția F: O compusă definită pe trei regiuni:

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

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



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



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

6. Implementarea Practică – Explicații Detaliate ale Codului

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

6.1 Biblioteci și Interfață Utilizatorului (UI)

Biblioteca necesara

`library(shiny)`

`library(ggplot2)`

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

Interfața utilizatorului (UI)

`ui <- fluidPage(`

`titlePanel("Aplicație combinată - Distribuții și Funcții Personalizate"),`

```

sidebarLayout(
  sidebarPanel(
    selectInput("mod", "Selectați modul:",
      choices = c("Distribuții variabile aleatoare",
                  "Funcții personalizate")),

    # Panou pentru distribuții
    conditionalPanel(
      condition = "input.mod == 'Distribuții variabile aleatoare'",

      selectInput(
        inputId = "distribution",
        label = "Selectați distribuția:",
        choices = c("Normală N(0,1)", "Normală N( $\mu, \sigma^2$ )",
                   "Exponențială Exp( $\lambda$ )", "Poisson Pois( $\lambda$ )",
                   "Binomială Binom(r, p)"),
        selected = "Normală N(0,1)"

      ),

      numericInput("n", "n (numărul de observații):", 100, min = 1),

      selectInput(
        inputId = "variable",
        label = "Selectați variabila:",
        choices = c("X", "3 - 2X", "2 + 5X", "3X - 2", "5X - 4",
                   "X2", "X3", "Suma ( $\Sigma X$ )", "Suma pătratelor ( $\Sigma X^2$ )"),
        selected = "X"

      ),

      conditionalPanel(
        condition = "input.distribution == 'Normală N( $\mu, \sigma^2$ )'",

        numericInput("mu", "μ (media):", 0),
        numericInput("sigma", "σ (deviația standard):", 1, min = 0.1)
      )
    )
  )
)

```

```

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

```

```

# Panou pentru funcții personalizate
conditionalPanel(
    condition = "input.mod == 'Funcții personalizate'" ,
    selectInput("distributie", "Alege funcția personalizată:" ,
    choices = c("Funcție personalizată A" ,
    "Funcție personalizată B" ,
    "Funcție personalizată C" ,
    "Funcție personalizată D" ,
    "Funcție personalizată E" ,
    "Funcție personalizata F" ,
    "Funcție personalizata G")),
conditionalPanel(

```

```

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

numericInput("c", "Coeficient c:", value = 1, min = 0),

sliderInput("interval_a", "Interval [a, b]", min = 0, max = 10, value = c(0, 2))

),

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

  numericInput("a", "Coeficient a:", value = 1),

  numericInput("b", "Coeficient b:", value = 1),

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

),

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

  numericInput("x_c_min", "x (minim):", value = 1, min = 1),

  numericInput("x_c_max", "x (maxim):", value = 5, min = 1)

),

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

  checkboxGroupInput("x_d_values", "Alege valorile discrete pentru x:",

    choices = 1:9, selected = 1:9

  ),

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

    numericInput("theta", "Parametru θ:", value = 1, min = 0.1),

    sliderInput("interval_e", "Interval [a, b]:", min = 0, max = 10, value = c(0, 5))

  ),

  conditionalPanel(
    condition = "input.distributie == 'Funcție personalizata F'",

    sliderInput("interval_f", "Interval [a, b]:", min = -10, max = 10, value = c(-5, 5))

  )
)

```

```

conditionalPanel(
  condition = "input.distributie == 'Funcție personalizata G'",

  sliderInput("interval_g", "Interval [a, b]:", min = -10, max = 10, value = c(-5, 5))

),
),

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

```

Explicații-cheie:

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

6.2 Logica Serverului (Server)

Logica serverului (Server)

```

server <- function(input, output) {

  output$distPlot <- renderPlot({

```

```
if (input$mod == "Distribuții variabile aleatoare") {
```

```
  n <- input$n
```

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

6.2.1 Generarea și transformarea variabilei aleatoare

```
# Generăm distribuția aleatoare
```

```
if (input$distribution == "Normală N(0,1)") {  
  
  data <- rnorm(n, mean = 0, sd = 1)  
  
} else if (input$distribution == "Normală N( $\mu, \sigma^2$ )") {  
  
  data <- rnorm(n, mean = input$mu, sd = input$sigma)  
  
} else if (input$distribution == "Exponențială Exp( $\lambda$ )") {  
  
  data <- rexp(n, rate = input$lambda)  
  
} else if (input$distribution == "Poisson Pois( $\lambda$ )") {  
  
  data <- rpois(n, lambda = input$lambda_pois)  
  
} else if (input$distribution == "Binomială Binom(r, p)") {  
  
  data <- rbinom(n, size = input$r, prob = input$p)  
  
}
```

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

```
# Calculăm variabila selectată
```

```
variable_data <- switch(input$variable,  
  
  "X" = data,  
  
  "3 - 2X" = 3 - 2 * data,  
  
  "2 + 5X" = 2 + 5 * data,
```

```

    "3X - 2" = 3 * data - 2,
    "5X - 4" = 5 * data - 4,
    "X^2" = data^2,
    "X^3" = data^3,
    "Suma ( $\Sigma X$ )" = cumsum(data), # Suma cumulative
    "Suma pătratelor ( $\Sigma X^2$ )" = cumsum(data^2) # Suma pătratelor
)

```

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

```

# Creăm un data frame pentru grafic
df <- data.frame(x = variable_data)

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

```

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

6.2.2 Plotarea funcțiilor personalizate

```
} else {  
  
  if (input$distributie == "Funcție personalizată A") {  
  
    a <- input$interval_a[1]  
  
    b <- input$interval_a[2]  
  
    c <- input$c  
  
    curve(c * x^4, from = a, to = b, col = "orange", lwd = 2,  
  
          main = "Funcție personalizată A: f(x) = c * x^4",  
  
          xlab = "x", ylab = "f(x)")  
  }  
}
```

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

```
} else if (input$distributie == "Funcție personalizată B") {  
  
  a <- input$interval_b[1]  
  
  b <- input$interval_b[2]  
  
  coef_a <- input$a  
  
  coef_b <- input$b  
  
  curve(coef_a * x + coef_b * x^2, from = a, to = b, col = "brown", lwd = 2,  
  
        main = "Funcție personalizată B: f(x) = a * x + b * x^2",  
  
        xlab = "x", ylab = "f(x)")  
}  
}
```

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

```
} else if (input$distributie == "Funcție personalizată C") {  
  
  x_vals <- seq(input$x_c_min, input$x_c_max, by = 1)  
  
  f_vals <- 4 / (x_vals * (x_vals + 1) * (x_vals + 2))  
  
  plot(x_vals, f_vals, type = "h", col = "pink", lwd = 2,  
       xlab = "x", ylab = "f(x)")  
}  
}
```

```
main = "Funcție personalizată C: f(x) = 4 / (x * (x+1) * (x+2))",
```

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

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

```
} else if (input$distributie == "Funcție personalizată D") {
```

```
x_vals <- as.numeric(input$x_d_values)
```

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

```
plot(x_vals, f_vals, type = "h", col = "cyan", lwd = 2,
```

```
main = "Funcție personalizată D: f(x) = log(x / (x+1))",
```

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

- Utilizatorul selectează un set de valori discrete via `checkboxGroupInput`.
- Facem transformarea $f(x)=\log(x+1)$
- Reprezentăm tot cu `type = "h"` pentru a sublinia caracterul discret.

```
} else if (input$distributie == "Funcție personalizată E") {
```

```
a <- input$interval_e[1]
```

```
b <- input$interval_e[2]
```

```
theta <- input$theta
```

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

```
col = "magenta", lwd = 2,
```

```
main = "Funcție personalizată E: f(x)",
```

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

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

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

```
a <- input$interval_f[1]
```

```

b <- input$interval_f[2]

f <- function(x) {

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

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

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

}

curve(f, from = a, to = b, col = "blue", lwd = 2,
      main = "Funcție personalizata F",
      xlab = "x", ylab = "f(x)")

```

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

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

- `ifelse` imbricate ne ajută să evaluăm fiecare punct pe porțiuni de domeniu diferite.

```

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

  a <- input$interval_g[1]

  b <- input$interval_g[2]

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

}

```

```

    }
})

}

shinyApp(ui = ui, server = server)

```

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

7. Dificultăți Întâmpinate și Observații

1. Parametri invalizi:

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

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

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

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

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

8. Posibilități de Extindere

- Compararea ECDF cu CDF-ul teoretic: Ar fi util un overlay grafic (ex. `geom_line`) pentru a vedea cât de aproape este eșantionul de distribuția ideală.

- Calcul efectiv al momentelor (media, varianța) direct în Shiny, cu formate text. De pildă, după generarea eșantionului, să afișăm `mean(variable_data)` și `var(variable_data)` în `verbatimTextOutput`. Similar, pentru funcțiile personalizate (dacă integrale).
 - Tab-uri multiple: În loc de un singur ecran, se pot organiza mai multe tab-uri (unul pentru distribuții, altul pentru funcții personalizate).
 - Export de rezultate: Se poate implementa un buton de „Download” care să permită salvarea graficelor sau a datelor generate în format CSV.
-

9. Concluzii Finale

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

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

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

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

EXERCITIU 3

Alexandrescu Andra

3.

a)

$$a) f(x) = e^{-2\theta} \cdot \frac{(2\theta)^x}{x!}, x \in \mathbb{N}, \theta \in \mathbb{R}$$

I. Metoda momentelor (MM)

$$E(X) = \bar{x} \quad \text{cum } X \text{ variabilă aleatoare discrete} \Rightarrow E(X) = \bar{x} = \sum_{x=0}^{\infty} x \cdot P(X=x)$$

Racirea furnizată de metoda a probabilității:

$$P(X=x) = f(x) = e^{-2\theta} \cdot \frac{2\theta^x}{x!}, x \in \mathbb{N}$$

(probabilitatea a la evenimente contruie un interval)

Se poate să observăm că este o distribuție Poisson

In ceea ce urmă, e vorba despre o evenimente (substituim în x) și unde $\lambda = 2\theta$: $\frac{2\theta^x \cdot e^{-2\theta}}{x!}$

Distribuția Poisson cu $\lambda = 2\theta$

are medie variabilă aleatoare Poisson $E(X) = \lambda = 2\theta$

(cum că general)

$$E(X) = \sum_{x=0}^{\infty} x \cdot \frac{2\theta^x}{x!} \cdot e^{-2\theta} = \lambda \cdot e^{-\lambda} \sum_{x=1}^{\infty} \frac{2\theta^{x-1}}{(x-1)!} = \lambda \cdot e^{-\lambda} \cdot \lambda = \lambda$$

Racirea:

$$E(X) = \sum_{x=0}^{\infty} x \cdot e^{-2\theta} \cdot \frac{2\theta^x}{x!} = e^{-2\theta} \sum_{x=1}^{\infty} x \cdot \frac{2\theta^x}{x!} \quad \overbrace{\lambda = 2\theta}^{\text{substituim}} \cdot e^{-2\theta} = \sum_{y=0}^{\infty} y \cdot \frac{2\theta^y}{y!}$$

Substituim x cu y unde $y = x - 1$

$x = 1 \Rightarrow y = 0$

$x \rightarrow \infty \Rightarrow y \rightarrow \infty$

$$= 2\theta \cdot e^{-2\theta} \cdot e^{2\theta} = 2\theta \cdot e^{2\theta - 2\theta} = 2\theta \cdot 1 = 2\theta$$

Obs. Că general formula exponentială: $e^x = \sum_{y=0}^{\infty} \frac{x^y}{y!}$

$$\Rightarrow e^{2\theta} = \sum_{y=0}^{\infty} \frac{2\theta^y}{y!}$$

Așadar $E(X) = 2\theta$, iar estimarea lui λ presupune să fie egală media teoretică $E(X)$ cu ea empirică \bar{x}

$$\text{unde } \bar{x} = \frac{1}{n} \sum_{i=1}^n x_i; \quad j. 2\theta = \bar{x} \Leftrightarrow \theta = \frac{\bar{x}}{2} = \frac{1}{2n} \sum_{i=1}^n x_i$$

acest set de date x_1, \dots, x_n nu da la media empirică să fie o val. ciprătoare de 2θ

II. Metoda Verosimilității Maxime

Fie x_1, x_2, \dots, x_n un eșantion cu valoriile de selecție x_1, x_2, \dots, x_n

Că general:

Functia de verosimilitate pt distribuția Poisson:

$$L(\lambda) = L(\lambda | x_1, x_2, \dots, x_n) = \prod_{i=1}^n P(X=x_i) = \prod_{i=1}^n e^{-\lambda} \cdot \frac{\lambda^{x_i}}{x_i!}$$

$$\text{Așadar, } L(2\theta | x_1, x_2, \dots, x_n) = \prod_{i=1}^n e^{-2\theta} \cdot \frac{2\theta^{x_i}}{x_i!} = e^{-2n\theta} \cdot \prod_{i=1}^n \frac{2\theta^{x_i}}{x_i!}$$

$$= e^{-2n\theta} \cdot (2\theta)^{\sum_{i=1}^n x_i} \cdot \prod_{i=1}^n \frac{1}{x_i!}$$

Logaritmarea verosimilității:

$$\ln L(\lambda) \stackrel{\text{devenire sumă}}{=} \ln(e^{-2n\theta}) + \ln((2\theta)^{\sum_{i=1}^n x_i}) + \ln\left(\prod_{i=1}^n \frac{1}{x_i!}\right) = -2n\theta + \sum_{i=1}^n x_i \cdot \ln(2\theta) + \sum_{i=1}^n \ln\left(\frac{1}{x_i!}\right)$$

$$= -2n\theta + \sum_{i=1}^n x_i \cdot \ln(2\theta) - \sum_{i=1}^n \ln(x_i!) = -2n\theta + \sum_{i=1}^n x_i \cdot \ln 2 + \sum_{i=1}^n x_i \cdot \ln \theta - \sum_{i=1}^n \ln(x_i!)$$

Derivarea log-verosimilității în raport cu λ :

$$\frac{\partial \ln L}{\partial \theta} = -2n + \sum_{i=1}^n x_i \cdot \frac{1}{\theta}$$

deci $\frac{\partial}{\partial \theta} (-2n\theta) = -2n$

deci $\frac{\partial}{\partial \theta} \left(\sum_{i=1}^n x_i \cdot \ln 2\theta \right) = \sum_{i=1}^n x_i \cdot \frac{1}{\theta}$

Egalarea derivatelor cu 0 pt a afla maximul:

$$-2n + \sum_{i=1}^n x_i \cdot \frac{1}{\theta} = 0 \Leftrightarrow \frac{\sum_{i=1}^n x_i}{\theta} = 2n \Leftrightarrow \hat{\theta} = \frac{\sum_{i=1}^n x_i}{2n} = \frac{1}{2} \cdot \frac{1}{n} \sum_{i=1}^n x_i = \frac{\bar{x}}{2}$$

Așadar, estimarea lui θ este $\hat{\theta} = \frac{\bar{x}}{2}$

```

## a) f(teta)(x) = e^(-2*teta)*((2*teta)^x/x!), x nr natural, teta nr real

# esantionul care e de fapt un vector numeric (c combina valorile intr-un vector)
date_esantion <- c(8, 12, 6, 14, 9, 12, 15, 7, 15, 7, 10, 10, 14, 9, 12, 15, 11, 6, 8, 6, 8, 8, 9, 12, 13, 10, 11,
11, 13, 15, 10, 8, 7, 8, 13, 9, 9, 13, 12, 9, 10, 6, 10, 8, 10, 11, 12, 11, 9, 10, 7, 8, 8, 16, 7, 15, 10, 10, 8, 14,
13, 4, 11, 13, 6, 9, 13, 10, 10, 12, 11, 5, 6, 4, 9, 6, 9, 7, 13, 9, 11, 5, 5, 9, 15, 10, 11, 10, 14, 7, 11, 9, 14,
10, 5, 10, 8, 12, 13, 11)

## MM
x_bar <- mean(date_esantion) # media 1/n*(suma dupa i de la 1 la n din)xi
teta_estimat <- x_bar/2 # formula determinata pentru functie folosind MM

## MVM
sum_x <- sum(date_esantion)
sum_log_factorial <- sum(lfactorial(date_esantion))

logVerosimilitate <- function(teta)
{
  log_verosim <- (-2)*length(date_esantion)*teta+log(2)*sum_x+log(teta)*sum_x-sum_log_factorial
  return(log_verosim)
}

t <- seq(0.001, 0.5, by = 0.0001)
plot(t, sapply(t, logVerosimilitate), type = "l", col = "blue")

rez <- optimize(logVerosimilitate, interval = c(0.001,100), maximum = TRUE)
teta_mvm <- rez$maximum

cat("Estimarea lui teta folosind metoda momentelor:", teta_estimat, "\n")
cat("Estimarea lui teta folosind metoda verosimilitatii maxime:", teta_mvm, "\n")

```

The figure shows the RStudio interface with a script file open. The script contains R code for estimating the parameter θ from a sample of size $n=10$. It uses the log-likelihood function and optimization to find the maximum likelihood estimate (MLE). The MLE is approximately 4.97. A plot of the log-likelihood function is shown on the right.

```

## a) f(teta)(x) = x^(2*teta)*(2^teta)Ax/x!, teta_nr natural, teta_nr real
# esantionul care de fapt un vector numeric ( combină valoarea într-un vector)
date_esantion <- c(8, 12, 6, 14, 9, 12, 15, 7, 5, 10, 14, 9, 12, 15, 11, 6, 8, 6, 8, 6)
## MM
x_bar <- mean(date_esantion) # media 1/n*(suma după i de la 1 la n din x)
teta_estimat <- x_bar/2 # formula determinată pentru funcție folosind MM
## MM
sum_x <- sum(date_esantion)
sum_log_factorial <- sum(factorial(date_esantion))
sum_x_log_teta <- sum_x*log(teta)
log_verosimilitate <- function(teta)
{
  log_verosim <- (-2)*length(date_esantion)*teta+log(2)*sum_x_log_teta+sum_x-sum_log_factorial
  return(log_verosim)
}
t <- seq(0.001, 0.5, by = 0.0001)
plot(t, sapply(t, log_verosimilitate), type = "l", col = "blue")
rez <- optimize(log_verosimilitate, interval = c(0.001,100), maximum = TRUE)
teta_mvm <- rez$maximum
teta_mvm
cat("Estimarea lui teta folosind metoda momentelor:", teta_estimat, "\n")
Estimarea lui teta folosind metoda momentelor: 4.97
cat("Estimarea lui teta folosind metoda verosimilității maxime:", teta_mvm, "\n")
Estimarea lui teta folosind metoda verosimilității maxime: 4.969991

```

b)

$$b) f_{\theta}(x) = C_m^x \cdot \theta^x \cdot (1-\theta)^{(1-x)}, x \in \{1, 2, 3, \dots, m\}, \theta \in (0, 1), m \text{ float}$$

X. variegata aletochore discrete

X variabila deosebită discrete
 $f_0(x)$ se poate interpreta ca o distribuție de probabilitate, numărăță cel
 deosebită lăuntrică (probabilitatea puterii a terminului $(1-\theta)$ din $(m-x)$ în $(1-\theta)$), ceea
 ce poate fi interpretat ca o lăuntrică invocată cu doar rezultate posibile (succes și
 eșec), în loc să se refere la un număr de succese din n invocări. Astăzi, în casul
 acesta, f_0 este probabilitatea de succes, iar $(1-\theta)^{(1-x)}$ probabilitatea de eșec, unde dacă $x=0$, va fi
 un eșec garantat.

b) $f_p(x) = C_m^x p^x (1-p)^{m-x}$ \Rightarrow seap x numeric
prob de success
 $m \in \mathbb{N}^*, x \in \{0, 1, \dots, m\}, p \in (0, 1)$
 x real. aleatorie discrete \downarrow nr. total evenimente
nr. total sucese

I. Metoda momentelor

Cat general media distributiei binomiale:

$$\begin{aligned} E(X) &= \bar{x} = \sum_{x=0}^m x \cdot C_m^x \cdot p^x \cdot (1-p)^{m-x} \xrightarrow{\text{Pd. } x=0} \sum_{x=1}^m x \cdot C_m^x \cdot p^x \cdot (1-p)^{m-x} \\ &\xlongequal{\substack{\uparrow \\ x \cdot C_m^x = x \cdot \frac{m!}{(m-x)!} = \frac{m!}{(x-1)!} \cdot C_{m-1}^{x-1}}} \sum_{x=1}^m C_{m-1}^{x-1} \cdot p^x \cdot (1-p)^{(m-1)-(x-1)} = \\ &= mp \cdot \sum_{x=1}^m C_{m-1}^{x-1} \cdot p^{x-1} \cdot (1-p)^{(m-1)-(x-1)} \xlongequal{\substack{\text{substitu} \ x \text{ cu } y = x-1 \\ x=1 \Rightarrow y=0 \\ x=m \Rightarrow y=m-1}} mp \cdot \sum_{y=0}^{m-1} C_{m-1}^y \cdot p^y \cdot (1-p)^{(m-1)-y} = \\ &= mp \cdot 1 = mp \end{aligned}$$

$$\sum_{j=0}^m C_{m-1}^j \cdot p^j \cdot (1-p)^{(m-1)-j} = (p + (1-p))^{m-1} = 1$$

$$E(X) = np = \bar{x} \Rightarrow \hat{p} = \frac{\bar{x}}{n} = \frac{1}{m^2} \cdot \sum_{x=1}^m x \cdot C_m^x$$

II. Metoda verosimilitatii maxime

Fie x_1, x_2, \dots, x_n esantion de valoriile de selecție x_1, x_2, \dots, x_n

Funcția de verosimilitate:

$$L(p) = L(p | x_1, x_2, \dots, x_n) = \prod_{i=1}^n P(x_i | X=x_i) = \prod_{i=1}^n C_m^{x_i} \cdot p^{x_i} \cdot (1-p)^{m-x_i}$$

Log-verosimilitate:

$$\ln L(p) = \sum_{i=1}^n \ln C_m^{x_i} + \sum_{i=1}^n x_i \ln(p) + \sum_{i=1}^n (m-x_i) \ln(1-p) =$$

Derivarea log-verosimilitatii in raport cu p :

$$\begin{aligned} \frac{\partial \ln L(p)}{\partial p} &= \frac{\sum_{i=1}^n x_i}{p} - \frac{\sum_{i=1}^n (m-x_i)}{1-p} \\ \frac{\partial}{\partial p} (\ln(p) \cdot \sum_{i=1}^n x_i) &= \frac{\sum_{i=1}^n x_i}{p} \\ \frac{\partial}{\partial p} (\ln(1-p) \cdot \sum_{i=1}^n (m-x_i)) &= \frac{-1}{1-p} \cdot \sum_{i=1}^n (m-x_i) \end{aligned}$$

Egalarea derivatei cu 0 pt sa afle maximul:

$$\begin{aligned} \frac{\sum_{i=1}^n x_i}{p} - \frac{\sum_{i=1}^n (m-x_i)}{1-p} &\Leftrightarrow (1-p) \sum_{i=1}^n x_i = p \sum_{i=1}^n (m-x_i) \Leftrightarrow \sum_{i=1}^n x_i - p \sum_{i=1}^n x_i = p \sum_{i=1}^n (m-x_i) \\ \Leftrightarrow p &= \frac{\sum_{i=1}^n x_i}{\sum_{i=1}^n (m-x_i) + \sum_{i=1}^n x_i} = \frac{\sum_{i=1}^n x_i}{n \cdot n} = \frac{\bar{x}}{n} = \frac{1}{m^2} \cdot \sum_{i=1}^m x_i \end{aligned}$$

b) $f(\text{teta})(x) = (\text{combinari de } n \text{ luate cate } x) * \text{teta}^x * (1-\text{teta})^{n-x}$, x apartine $1, 2, 3, \dots, n$, teta apartine $(0, 1)$, n fixat

esantionul care e de fapt un vector numeric (c combina valorile intr-un vector)

date_esantion <- c(3, 2, 1, 4, 2, 3, 4, 1, 3, 2, 2, 4, 2, 1, 7, 5, 4, 5, 5, 2, 3, 4, 3, 1, 2, 4, 1, 1, 2, 3, 1, 3, 1, 4, 1, 3, 1, 6, 1, 3, 3, 4, 3, 1, 3, 2, 2, 3, 2, 4,

1, 1, 2, 6, 3, 1, 3, 6, 1, 2, 3, 6, 3, 2, 2, 2, 4, 2, 1, 3, 3, 4, 2, 3, 4, 1, 4, 4, 6, 3, 3, 5, 2, 2, 2, 3, 1, 3, 1, 3, 3, 5, 3, 4, 3, 2, 4, 2, 3, 3)

```

n <- 14

## MM
x_bar <- mean(date_esantion) # media 1/n*(suma dupa i de la 1 la n din)xi
teta_estimat <- x_bar/n # formula determinata pentru functie folosind MM

## MVM
sum_x <- sum(date_esantion)
sum_n_minus_x <- sum(n-date_esantion)
sum_log_combinari <- sum(log(choose(n,date_esantion)))

logVerosimilitate <- function(teta)
{
  log_verosim <- sum_log_combinari+log(teta)*sum_x+log(1-teta)*sum_n_minus_x
  return(log_verosim)
}

t <- seq(0.001, 0.999, by = 0.001)
plot(t, sapply(t, logVerosimilitate), type = "l", col = "blue")

rez <- optimize(logVerosimilitate, interval = c(0.001,0.999), maximum = TRUE)
teta_mvm <- rez$maximum

cat("Estimarea lui teta folosind metoda momentelor:", teta_estimat, "\n")
cat("Estimarea lui teta folosind metoda verosimilitatii maxime:", teta_mvm, "\n")


```

The screenshot shows the RStudio interface with the following components:

- Top Bar:** RStudio, File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Project Pane:** Project Subject 3.8, Go to file/function, Addins.
- Environment Pane:** Shows variables and their values:

Variable	Type	Value
date_esantion	num [1:100]	3 2 1 4 2 3 4 1 3 2 ...
n	num	14
sum_log_combinari	num	530.172851340239
sum_n_minus_x	num	1118
sum_x	num	282
t	num [1:999]	0.001 0.002 0.003 0.004 0.005 0.006 0.007 0.0...
teta_estimat	num	0.201428571428571
teta_mvm	num	0.201443553358429
- Plots Pane:** A line plot titled "sapply(t, logVerosimilitate)" showing the log-likelihood function. The x-axis is labeled "t" and ranges from 0.0 to 1.0. The y-axis ranges from -7000 to -1000. The curve starts at approximately (-0.05, -1000), rises to a peak around (0.1, -1000), and then gradually decreases towards -7000 as t approaches 1.0.
- Console Pane:** Shows the R session history with the same code as the script, including the output of the cat statements.

c)

$$c) f_{\theta}(x) = \frac{x^{\alpha-1} \cdot e^{-\frac{x}{\theta}}}{\Gamma(\alpha) \cdot \theta^\alpha} \cdot \mathbf{1}_{(0, \infty)}, \theta > 0, \alpha > 0 \text{ cunoscut, Distribuția Gamma}$$

x variație abatoare continuă

I. Metoda momentelor

$$E(x) = \bar{x} = \int_{-\infty}^{\infty} x \cdot f_{\theta}(x) dx = \int_0^{\infty} x \cdot \frac{x^{\alpha-1} \cdot e^{-\frac{x}{\theta}}}{\Gamma(\alpha) \cdot \theta^\alpha} dx = \frac{1}{\Gamma(\alpha) \cdot \theta^\alpha} \cdot \int_0^{\infty} x^\alpha \cdot e^{-\frac{x}{\theta}} dx$$

$$= \frac{1}{\Gamma(\alpha) \cdot \theta^\alpha} \cdot \theta^{\alpha+1} \cdot \int_0^{\infty} t^\alpha \cdot e^{-t} dt = \frac{\theta}{\Gamma(\alpha)} \cdot \Gamma(\alpha+1) = \frac{\theta \cdot \alpha \cdot \Gamma(\alpha)}{\Gamma(\alpha+1)} = \alpha \theta \Rightarrow \hat{\theta} = \frac{\bar{x}}{\alpha}$$

Prop. (recursivitate) $\Gamma(\alpha+1) = \alpha \Gamma(\alpha)$

II. Metoda verosimilității maxime

Fie x_1, x_2, \dots, x_n un esantion cu valoare de selecție x_1, x_2, \dots, x_n

Funcția de verosimilitate:

$$L(\theta) = L(\theta | x_1, x_2, \dots, x_n) = \prod_{i=1}^n f_{\theta}(x_i) = \prod_{i=1}^n \frac{x_i^{\alpha-1} \cdot e^{-\frac{x_i}{\theta}}}{\Gamma(\alpha) \cdot \theta^\alpha} = \left(\frac{1}{\Gamma(\alpha) \cdot \theta^\alpha} \right)^n \cdot \left(\prod_{i=1}^n x_i^{\alpha-1} \right) \cdot e^{-\frac{1}{\theta} \sum_{i=1}^n x_i}$$

Log-verosimilitate:

$$\ln L(\theta) = -n \ln(\Gamma(\alpha)) - n \ln(\theta) - \frac{\sum_{i=1}^n x_i}{\theta}$$

Derivarea log-verosimilității în raport cu θ :

$$\frac{\partial \ln L(\theta)}{\partial \theta} = -\frac{n}{\theta} + \frac{\sum_{i=1}^n x_i}{\theta^2}$$

Egalarea derivatiei cu 0 pt a afila maximul:

$$\frac{\sum_{i=1}^n x_i}{\theta^2} = \frac{n}{\theta} \quad \left| \begin{array}{l} \theta \Leftrightarrow \frac{n}{\theta} = n \alpha \left| \frac{1}{m} \Leftrightarrow \frac{\bar{x}}{\theta} = \alpha \Leftrightarrow \hat{\theta} = \frac{\bar{x}}{\alpha} \end{array} \right. \right.$$

c) $f(\text{teta})(x) = (x^{\alpha-1} e^{-x/\text{teta}}) / (\text{gamma}(\alpha) \cdot \text{teta}^\alpha)$, x nr din $(0, \infty)$, $\text{teta} > 0$, $\alpha > 0$ cunoscut

esantionul care e de fapt un vector numeric (c combina valorile intr-un vector)

date_esantion <- c(6.269128, 25.204245, 13.994878, 13.391437, 11.458827, 10.565065, 11.706398,

10.625808,

7.485952, 16.353358, 9.277565, 8.566438, 14.788638, 6.830955, 9.542004, 20.272463,
 36.562137, 12.244005, 16.084879, 11.454008, 15.592298, 6.332908, 13.106441, 6.198981,
 15.726780, 7.883712, 35.124934, 11.856011, 13.766200, 16.534869, 16.803648, 11.196542,
 19.785629, 26.300717, 21.270154, 7.192149, 5.882948, 15.812796, 10.963237, 24.963600,
 13.802383, 15.281262, 10.310398, 20.940469, 23.992540, 15.869985, 12.041726, 12.521264,
 10.869006, 15.386514, 14.636832, 18.104562, 17.029779, 4.506616, 20.941222, 12.050877,
 9.757833, 20.070802, 12.472900, 6.474476, 15.059776, 13.157344, 9.124414, 13.768482,
 24.354934, 12.363936, 11.110749, 9.092514, 17.856801, 14.757801, 13.898665, 9.119410,
 11.430184, 11.958829, 13.516191, 10.701083, 14.713596, 10.121266, 16.945351, 13.524070,
 14.742403, 19.165805, 10.338392, 12.327837, 19.619227, 7.328246, 14.894399, 19.631003,
 7.622796, 12.343832, 13.138183, 10.061520, 17.674638, 9.675168, 12.115561, 15.182861,
 13.292479, 17.888244, 16.695139, 2.952334)

```

alfa <- 7

## MM
x_bar <- mean(date_esantion) # media 1/n*(suma dupa i de la 1 la n din)xi
teta_estimat <- x_bar/alfa # formula determinata pentru functie folosind MM

## MVM
sum_x <- sum(date_esantion)
log_gamma_alfa <- log(lfactorial(alfa-1))

logVerosimilitate <- function(teta)
{
  log_verosim <-
    (-length(date_esantion))*log_gamma_alfa-length(date_esantion)*alfa*log(teta)-sum_x/teta
  return(log_verosim)
}

t <- seq(0.001, 0.5, by = 0.0001)
plot(t, sapply(t, logVerosimilitate), type = "l", col = "blue")

```

```

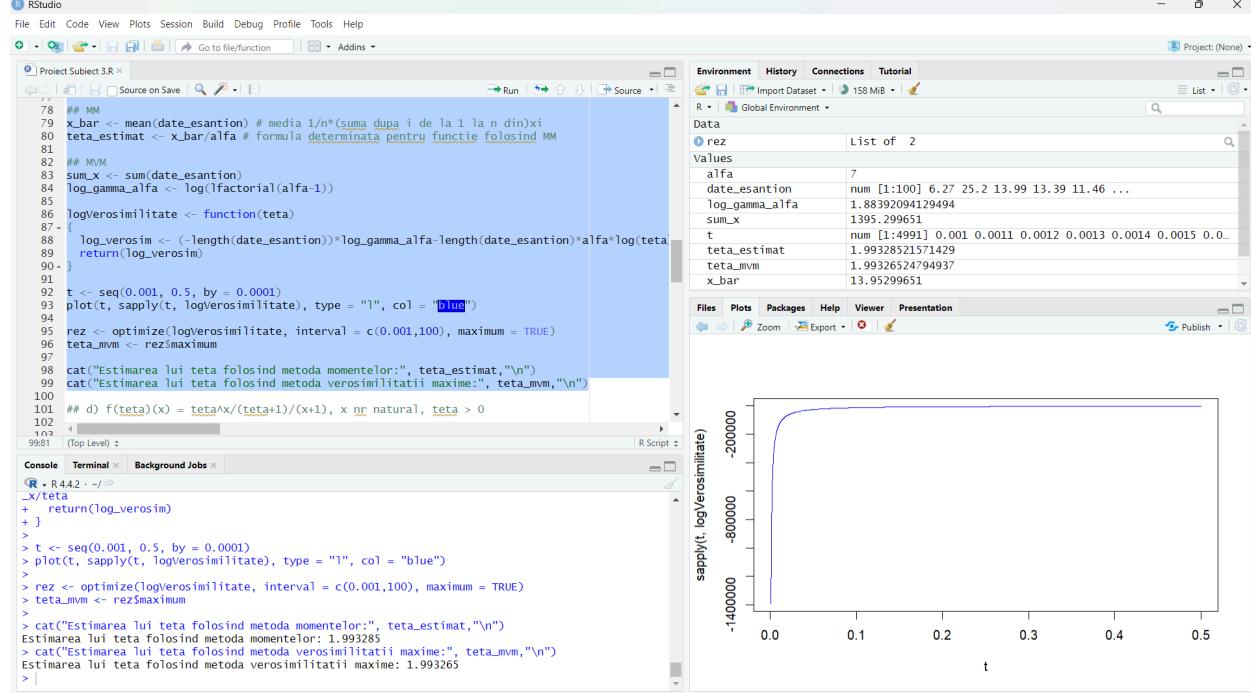
rez <- optimize(logVerosimilitate, interval = c(0.001,100), maximum = TRUE)
teta_mvm <- rez$maximum

```

```

cat("Estimarea lui teta folosind metoda momentelor:", teta_estimat, "\n")
cat("Estimarea lui teta folosind metoda verosimilitatii maxime:", teta_mvm, "\n")

```



d)

d) $f_\theta(x) = \frac{\theta^x}{(\theta+1)^{x+1}}, x \in \mathbb{N}, \theta > 0$
 X rezolvarele sunt efectuate de la 0
 Metoda momentelor

$$E(X) = \bar{x} = \sum_{x=0}^{\infty} x \cdot f_\theta(x) = \sum_{x=0}^{\infty} x \cdot \frac{\theta^x}{(\theta+1)^{x+1}} = \frac{1}{\theta+1} \cdot \sum_{x=0}^{\infty} x \cdot \left(\frac{\theta}{\theta+1}\right)^x$$

Stim ca se găsește: $\frac{1}{\theta+1} \cdot E(\theta X) = \theta \Rightarrow \hat{\theta} = \bar{x}$

$$\sum_{x=0}^{\infty} x \cdot p^{x-1} = \frac{1}{1-p} \quad (p \text{ prob. de succes})$$

$$\text{stfel } \sum_{x=0}^{\infty} x \cdot p^x = \frac{1}{1-p} \cdot \frac{d}{dp} \text{ derivatia fct. dep. } \Leftrightarrow \frac{d}{dp} \left(\sum_{x=0}^{\infty} x \cdot p^x \right) = \frac{1}{(1-p)^2}$$

$$\Leftrightarrow \sum_{x=0}^{\infty} x \cdot p^{x-1} = \frac{1}{(1-p)^2}; \text{ Adunam } \sum_{x=0}^{\infty} x \cdot p^x = p \cdot \sum_{x=0}^{\infty} x \cdot p^{x-1} = \frac{n}{(1-p)}$$

Deci: $\sum_{x=0}^{\infty} x \cdot \left(\frac{\theta}{\theta+1}\right)^x = \frac{\theta}{\theta+1} \cdot \frac{1}{\left(1 - \frac{\theta}{\theta+1}\right)^2} = \frac{\theta}{(\theta+1) \cdot \left(\frac{\theta+1-\theta}{\theta+1}\right)^2} = \frac{\theta}{(\theta+1) \cdot \frac{1}{(\theta+1)^2}} = \theta(\theta+1) = \theta^2 + \theta$

II. Metoda verosimilității maxime

Fie x_1, x_2, \dots, x_n un esantion cu valorile de selecție x_1, x_2, \dots, x_n

Functie de verosimilitate:

$$L(\theta) = L(\theta | x_1, x_2, \dots, x_n) = \prod_{i=1}^n f_\theta(x_i) = \prod_{i=1}^n \frac{\theta^{x_i}}{(\theta+1)^{x_i+1}} = \frac{\theta^{\sum_{i=1}^n x_i}}{(1+\theta)^{\sum_{i=1}^n (x_i+1)}}$$

Log-verosimilitate:

$$\ln L(\theta) = \left(\sum_{i=1}^n x_i \right) \cdot \ln \theta - \left(\sum_{i=1}^n x_i + n \right) \cdot \ln (\theta+1)$$

Derivarea log-verosimilității:

$$\frac{\partial \ln L(\theta)}{\partial \theta} = \frac{n\bar{x}}{\theta} - \frac{n\bar{x} + n}{\theta+1}$$

Egalarea derivatei cu 0 pt a afla maximul:

$$\frac{n\bar{x}}{\theta} = \frac{n(\bar{x}+1)}{\theta+1} \quad | \cdot \frac{1}{n} \Leftrightarrow \frac{\bar{x}}{\theta} = \frac{\bar{x}+1}{\theta+1} \Leftrightarrow \cancel{\bar{x}} \theta + \cancel{\bar{x}} = \theta\bar{x} + \theta \Leftrightarrow \hat{\theta} = \bar{x}$$

d) $f(\text{teta})(x) = \text{teta}^x / (\text{teta}+1) / (x+1)$, x nr natural, teta > 0

esantionul care e de fapt un vector numeric (c combina valorile intr-un vector)
 date_esantion <- c(6, 3, 24, 24, 4, 56, 10, 13, 2, 28, 24, 2, 22, 11, 2, 8, 118, 2, 14, 19, 7, 9, 8, 189, 2, 9, 21, 6,

6, 2, 3, 2, 3, 18, 3, 2, 21, 1, 5, 9, 11, 13, 19, 76, 1, 5, 9, 4, 57, 1, 2, 16, 5, 2, 20, 8, 1,
 40, 6, 4, 19, 6, 3, 2, 4, 9, 1, 5, 10, 12, 6, 525, 19, 6, 17, 2, 5, 159, 5, 62, 6, 3, 45, 21, 23,
 3, 17, 2, 1, 1, 474, 15, 3, 3, 7, 7, 13, 4, 38, 4)

MM

x_bar <- mean(date_esantion) # media 1/n*(suma dupa i de la 1 la n din)xi
 teta_estimat <- x_bar # formula determinata pentru functie folosind MM

MVM

sum_x <- sum(date_esantion)

```
logVerosimilitate <- function(teta)
{
  log_verosim <- log(teta)*sum_x-log(teta+1)*(length(date_esantion)+sum_x)
```

```

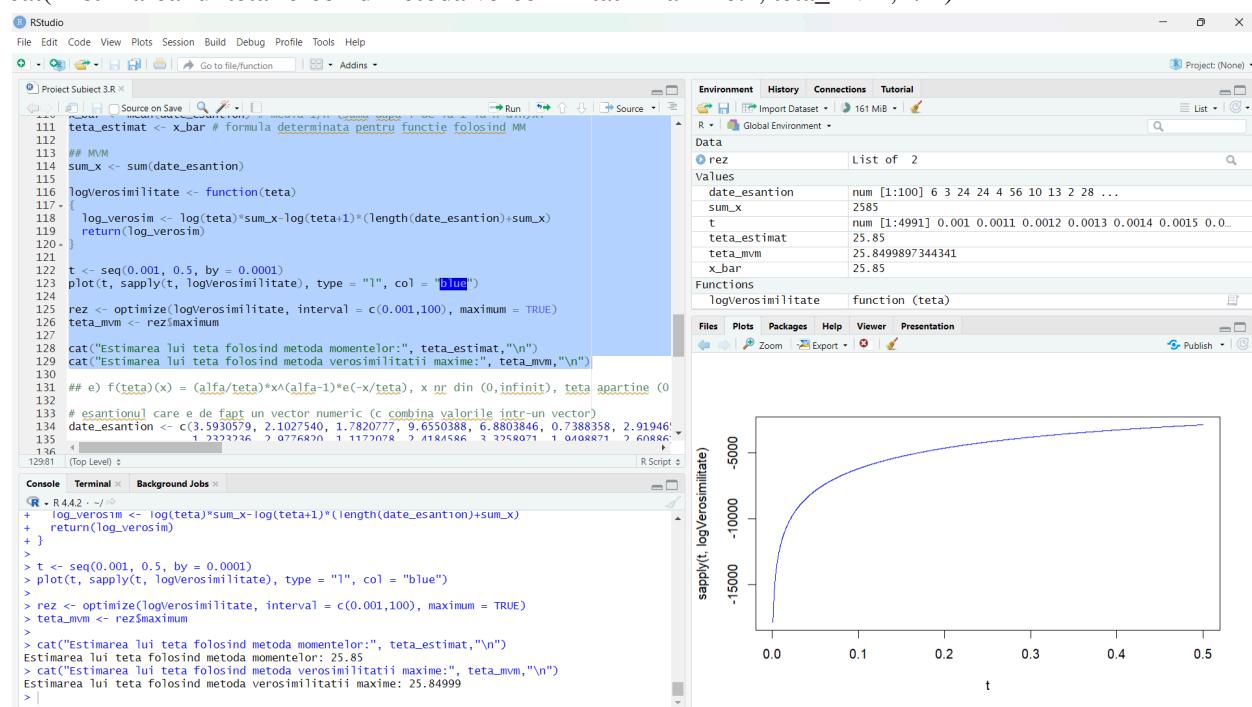
return(log_verosim)
}

t <- seq(0.001, 0.5, by = 0.0001)
plot(t, sapply(t, logVerosimilitate), type = "l", col = "blue")

rez <- optimize(logVerosimilitate, interval = c(0.001, 100), maximum = TRUE)
teta_mvm <- rez$maximum

cat("Estimarea lui teta folosind metoda momentelor:", teta_estimat, "\n")
cat("Estimarea lui teta folosind metoda verosimilitatii maxime:", teta_mvm, "\n")

```



e)

$$\begin{aligned}
& \text{2) } f(x) = \frac{\lambda}{\theta} \cdot x^{\lambda-1} \cdot e^{-\frac{x}{\theta}}, \quad \theta > 0, \quad x > 0 \\
& \text{X este variabila aleatoare continuu} \\
& \text{I. Metoda momentelor} \\
& E(X) = \int_0^\infty x \cdot \frac{\lambda}{\theta} \cdot x^{\lambda-1} \cdot e^{-\frac{x}{\theta}} dx = \frac{\lambda}{\theta} \int_0^\infty x^\lambda \cdot e^{-\frac{x}{\theta}} dx = \frac{\lambda}{\theta} \cdot \int_0^\infty \theta^\lambda \cdot t^\lambda \cdot e^{-\frac{\theta t}{\theta}} dt = \lambda \cdot \theta^\lambda \cdot \int_0^\infty t^\lambda \cdot e^{-\theta t} dt \\
& \underset{\text{Simplificare}}{\cancel{\theta^\lambda}} \underset{\text{Simplificare}}{\cancel{dt}} \underset{\text{Simplificare}}{\cancel{e^{-\theta t}}} \underset{\text{Simplificare}}{\cancel{\theta^\lambda}} \underset{\text{Simplificare}}{\cancel{dt}} \underset{\text{Simplificare}}{\cancel{t^\lambda}} \underset{\text{Simplificare}}{\cancel{e^{-\theta t}}} \\
& = \lambda \cdot \theta^\lambda \cdot \lambda! \Rightarrow \theta = \frac{\bar{x}}{\bar{x}(\lambda+1)} \\
& \hat{\theta} = \left(\frac{\bar{x}}{\bar{x}(\lambda+1)} \right)^{\frac{1}{\lambda}}
\end{aligned}$$

II. Metoda verosimilității maxime

Fie x_1, x_2, \dots, x_n un esantion cu valorile de selecție x_1, x_2, \dots, x_n

Funcția de verosimilitate:

$$L(\theta) = L(\theta | x_1, x_2, \dots, x_n) = \prod_{i=1}^n f_\theta(x_i) = \prod_{i=1}^n \left(\frac{\lambda}{\theta} x_i^{\lambda-1} \cdot e^{-\frac{x_i}{\theta}} \right) = \left(\frac{\lambda}{\theta} \right)^n \left(\prod_{i=1}^n x_i^{\lambda-1} \right) \cdot e^{-\frac{1}{\theta} \left(\sum_{i=1}^n x_i \right)}$$

Log-verosimilitate:

$$\ln L(\theta) = \underbrace{m \ln \lambda - m \ln \theta}_{\text{constante}} + \underbrace{(n-1) \left(\sum_{i=1}^n \ln x_i \right)}_{\text{constante}} - \frac{1}{\theta} \cdot \sum_{i=1}^n x_i$$

Derivarea log-verosimilității:

$$\frac{\partial \ln L(\theta)}{\partial \theta} = -\frac{n}{\theta} + \frac{1}{\theta^2} \left(\sum_{i=1}^n x_i \right) \approx n\bar{x}$$

Egalarea derivatăi cu opt să afle maximul:

$$\begin{aligned} \frac{1}{\theta^2} \cdot n\bar{x} &= \frac{n}{\theta} \mid \cdot \frac{1}{n} \Leftrightarrow \frac{\bar{x}}{\theta^2} = \frac{1}{\theta} \Leftrightarrow \theta \bar{x} - \theta^2 = 0 \\ &\quad \theta(\bar{x} - \theta) = 0 \\ \Leftrightarrow \theta &= 0 \text{ sau } \hat{\theta} = \bar{x} \\ \text{dar} \\ \theta &> 0 \end{aligned}$$

(coerență)

e) $f(\text{teta})(x) = (\text{alfa}/\text{teta})^x \cdot (\text{alfa}-1)^{-(x-\text{teta})}$, x nr din $(0, \infty)$, teta apartine $(0, \infty)$, alfa apartine $(0, \infty)$ fixat

esantionul care e de fapt un vector numeric (c combina valorile intr-un vector)

date_esantion <- c(3.5930579, 2.1027540, 1.7820777, 9.6550388, 6.8803846, 0.7388358, 2.9194654, 3.1178660,

1.2323236, 2.9776820, 1.1172078, 2.4184586, 3.3258971, 1.9498871, 2.6088612, 3.9535062, 3.0389107, 4.4226628, 3.9366318, 2.4551569, 5.2814487, 5.6778622, 4.7683935, 1.1581498, 3.1270783, 4.1473311, 7.4830426, 1.1342893, 1.7773392, 7.7510826, 1.3919927, 2.3613291, 2.6234826, 1.6562602, 1.4992235, 2.3455062, 3.8458809, 5.8333841, 3.3834034, 1.5202546, 3.1248186, 5.3029567, 3.6225571, 4.8309931, 3.1579595, 3.2640258, 3.9538891, 4.0796841, 4.0991772, 3.2779944, 2.5002127, 3.0654695, 1.6996010, 3.2175175, 1.9033087, 4.4052061, 2.3158379, 2.4778345, 5.4382190, 4.9141207, 6.0978745, 1.1428936, 3.5639106, 7.4541937, 7.7778289, 3.2859563, 0.7432908, 1.4442696, 3.6619932, 2.8361371, 4.3180773, 1.6763585, 4.4464154, 2.5049617, 0.4448735, 5.0518839, 3.4151834, 1.6823650, 5.4517583, 2.8212788, 2.1566837, 2.9893287, 1.6925123, 6.5197938, 4.2165408, 1.6728425, 2.7650830, 2.6742755, 2.9622047, 0.7809781, 1.3913415, 5.3430751, 2.4859925, 3.7329465, 6.3129236, 0.6635228, 3.7640343, 2.1850174, 4.3773328, 5.0931544)

alfa <- 3

MM

x_bar <- mean(date_esantion) # media $1/n \cdot (\text{suma după } i \text{ de la 1 la } n \text{ din } x_i)$

asta e singurul subpunkt în care formula de la MM nu corespunde cu cea de la MVM
 teta_estimat_mm <- (x_bar / (alfa^(2 * lfactorial(alfa-1))))^(1/alfa)

MVM

sum_x <- sum(date_esantion)

sum_log_x <- sum(log(date_esantion))

```

logVerosimilitate <- function(teta)
{
  log_verosim <-
  length(date_esantion)*log(alfa)-length(date_esantion)*log(teta)+(alfa-1)*sum_log_x-(1/teta)*sum(date_e
  santion)
  return(log_verosim)
}

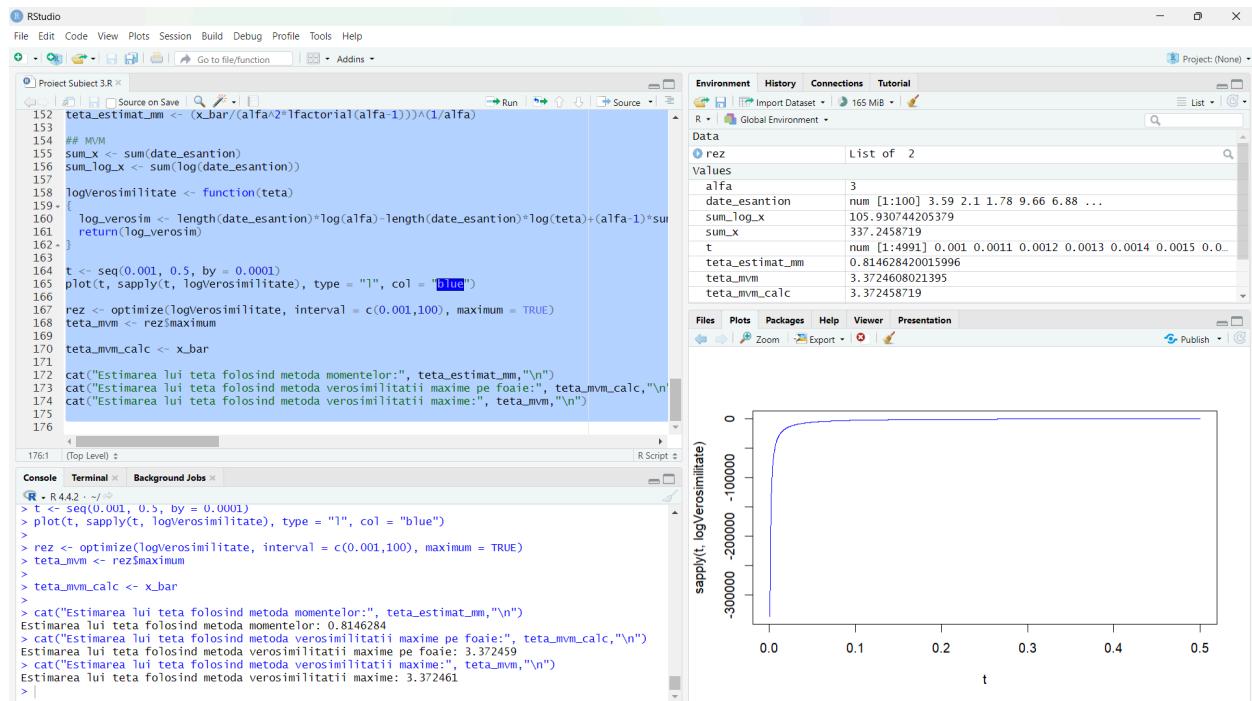
t <- seq(0.001, 0.5, by = 0.0001)
plot(t, sapply(t, logVerosimilitate), type = "l", col = "blue")

rez <- optimize(logVerosimilitate, interval = c(0.001,100), maximum = TRUE)
teta_mvm <- rez$maximum

teta_mvm_calc <- x_bar

cat("Estimarea lui teta folosind metoda momentelor:", teta_estimat_mm, "\n")
cat("Estimarea lui teta folosind metoda verosimilitatii maxime pe foaie:", teta_mvm_calc, "\n")
cat("Estimarea lui teta folosind metoda verosimilitatii maxime:", teta_mvm, "\n")

```



Dupa cum se poate observa pentru aceasta functie estimatorul teta da diferit in ambele metode, atat prin calcul direct, cat si la nivel de cod.

Referinte:

- Exercitiul 1:

<http://math/etc.tuiasi.ro/rstrugariu/cursuri/SPD2017/c7.pdf>

<https://statproofbook.github.io/P/exp-mean.html>

[https://www.math.uaic.ro/~maticiuc/didactic/Probability Theory Course 7 8 9 10.pdf](https://www.math.uaic.ro/~maticiuc/didactic/Probability_Theory_Course_7_8_9_10.pdf)

- Exercitiul 2:

→ Shiny (RStudio)

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

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

→ ggplot2

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

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

→ R (limbajul în sine)

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