PROIECT PROBABILITATI SI STATISTICI

I. Se consideră o activitate care presupune parcurgerea secvențială a $\boldsymbol{n}$ etape. Timpul necesar finalizării etapei $\boldsymbol{i}$ de către o persoană $\boldsymbol{A}$ este o variabilă aleatoare $T_i \sim Exp(\lambda_i)$. După finalizarea etapei $\boldsymbol{i}$, A va trece în etapa $\boldsymbol{i+1}$ cu probabilitatea $\boldsymbol{\alpha_i}$ sau va opri lucrul cu probabilitatea $\mathbf{1} - \boldsymbol{\alpha_i}$. Fie $\boldsymbol{T}$ timpul total petrecut de persoana A în realizarea activității respective.

1) Construiți un algoritm în R care simulează 10^6 valori pentru v.a. $\boldsymbol{T}$ și în baza acestora aproximați ***E(T)***. Reprezentați grafic într-o manieră adecvată valorile obținute pentru $\boldsymbol{T}$. Ce puteți spune despre repartiția lui $\boldsymbol{T}$?
2) Calculați valoarea exactă a lui ***E(T)*** și comparați cu valoarea obținută prin simulare.
3) În baza simulărilor de la 1) aproximați probabilitatea ca persoana $\boldsymbol{A}$ să finalizeze activitatea.
4) În baza simulărilor de la 1) aproximați probabilitatea ca persoana $\boldsymbol{A}$ să finalizeze activitatea într-un timp mai mic sau egal cu $\boldsymbol{\sigma}$.
5) În baza simulărilor de la 1) determinați timpul minim și respectiv timpul maxim în care persoana $\boldsymbol{A}$ finalizează activitatea și reprezentați grafic timpii de finalizare a activității din fiecare simulare. Ce puteți spune despre repartiția acestor timpi de finalizare a activității?
6) În baza simulărilor de la 1) aproximați probabilitatea ca persoana $\boldsymbol{A}$ să se oprească din lucru înainte de etapa $\boldsymbol{k}$, unde $1 < k \leq n$. Reprezentați grafic probabilitățile obținute într-o manieră corespunzătoare. Ce puteți spune despre repartiția probabilităților obținute?

Cerința 1:

Explicarea cerinței: simulăm 1000000 valori pentru $\boldsymbol{T}$, apoi calculăm media lor pentru a aproxima ***E(T)***. Reprezentăm grafic valorile lui $\boldsymbol{T}$.

Pentru a simula cele 1000000 valori ale lui T am ales un nr de etape n = 10 și valori diferite pentru α (probabilitatea de a trece la următoare etapă) și λ (vor reprezenta parametrii pentru fiecare etapa a activității). De asemena, am setat un seed pentru a genera aceleași valori random de fiecare dată când rulăm programul.

Funcția ***simulate_T()*** va calcula timpul total ***T*** pentru activitate. Astfel, pentru o valoare a lui ***T,*** va trece prin toate cele ***n*** etape (dacă activitatea este finalizată), iar pentru fiecare etapă se va calcula timpul de finalizare a acelei etape cu ajutorul funcției ***rexp()***. Funcția ***rexp()*** va genera un număr random, dar care sa fie cât mai apropiat de $\frac{1}{\lambda_i}$ (lambda corespunde numărului etapei, acesta fiind rata evenimentelor pentru distrbuția exponențială). Mai departe se verifică dacă numărul generat random între 0 și 1 de ***runif()*** este mai mare decât probabilitate $\boldsymbol{\alpha_i}$ de a trece la următoarea etapă. În acest fel verificăm dacă activitatea se oprește la etapa ***i*** sau nu, la final returnând timpul total pentru îndeplinirea activității.

Pentru fiecare 1000000 valori a fost aplicată funcția ***simulate_T()*** și creat un vector cu acești timpi totali. Ca să aproximăm ***E(T)*** am calculat media aritmetică a tuturor timpilor din ***T_values.***

$$E(T) \approx \frac{1}{n}\sum_{i=1}^{n} T_i$$

n = numărul de simulări

T_i = valoarea lui ***T*** obținută la simularea ***i***

În continuare am reprezentat grafic valorile obținute printr-o histogramă cu ajutorul funcțiilor ***hist(), abline(), legend()***.

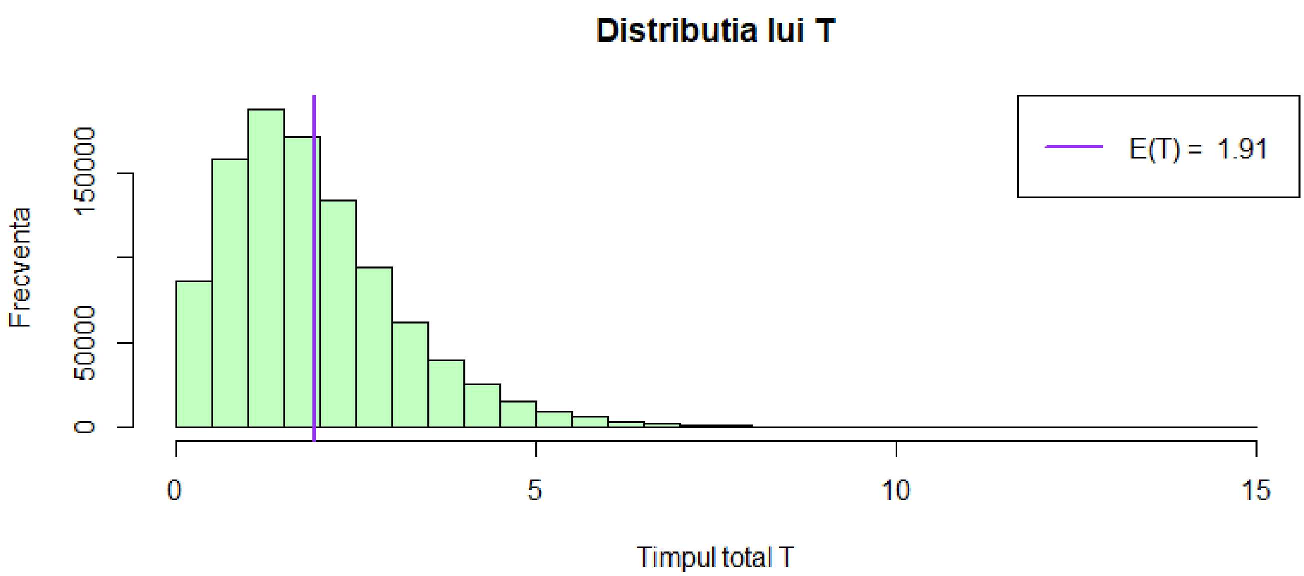
```r
set.seed(123)  # pt aceleasi val random cand dai run
n <- 10  # nr de etape
lambda <- c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)  # parametrii lambda pentru fiecare etapa
alpha <- c(0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1, 0.05)  # prob de trecere la etapa urm
# alfa de i este prob ca persoana A sa treaca de la etapa i la etapa i+1
# 1-alfa de i este prob ca persoana A sa se opreasca dupa etapa i
nrSim <- 1000000  # nr de simulari


################## cerinta 1 ########################

simulate_T <- function(n, lambda, alpha) {
  total_time <- 0
  for (i in 1:n) {
    time <- rexp(1, rate = lambda[i])  # timp pentru etapa i
    total_time <- total_time + time
    if (runif(1) > alpha[i]) {  # alege random un nr intre 0 si 1
      break  # stop dupa etapa i daca nr ales este mai mare decat prob alfa de i
    }
  }
  return(total_time)
}

T_values <- replicate(nrSim, simulate_T(n, lambda, alpha))
# aplica functia/operatia simulate_T de 10^6 ori pt T
mean_T <- mean(T_values)  # media ar a timpilor totali din simulari

# reprezentare grafica
hist(T_values, breaks = 50, main = "Distributia lui T", xlab = "Timpul total T",ylab="Frecventa", col = "darkseagreen1")
# daca freq=false arata probabilitatile
# breaks = nr bins/cosuri/drept alea verzi
abline(v = mean_T, col = "purple1", lwd = 2)
# linia mov pt media lui T
legend("topright", legend = paste("E(T) = ", round(mean_T, 2)), col = "purple1", lwd = 2)
# legenda pt linia mediei lui T
print(paste("Valoarea aproximata a lui E(T) =", mean_T))
```




```
> print(paste("Valoarea aproximata a lui E(T) =", mean_T))
[1] "Valoarea aproximata a lui E(T) = 1.91413497263285"
>
```

Ce putem spune despre repartiția lui ***T***?

- ***T*** este o sumă de variabile aleatoare exponențiale cu probabilități de trecere între etape. Fiecare etapă ***i*** are un timp $\boldsymbol{T_i \sim Exp(\lambda_i)}$, iar suma acestor timpi este condiționată de probabilitățile $\boldsymbol{\alpha_i}$.
- Repartiția lui ***T*** este asimetrică, deoarece timpul total poate fi foarte mare dacă persoana A parcurge multe etape, dar nu poate fi negativ.
- Are o coadă lungă la dreapta, deoarece există o probabilitate mică (dar nenulă) ca persoana A să parcurgă toate etapele.
- Media ***E*(*T*)** este finită și poate fi calculată exact (așa cum am făcut la punctul 2) și ar trebui să fie apropiată de valoarea teoretică (valoarea exactă).
- Dacă valorile $\boldsymbol{\alpha_i}$ sunt mici (probabilități mici de trecere la etapa următoare), persoana A se va opri rapid, iar ***T*** va avea valori mici.
- Repartiția lui *T* este condiționată de numărul de etape parcurse. De exemplu:
  - → Dacă persoana A se oprește după prima etapă, $\boldsymbol{T \sim Exp(\lambda)}$.
  - → Dacă parcurge toate etapele, ***T*** este suma a ***n*** variabile exponențiale cu parametrii $\lambda_1, \lambda_2, \ldots \lambda_n$

Cerința 2:

Explicarea cerinței: calculăm exact valoarea lui ***E(T)*** și o comparăm cu cea obținută prin simulare.

Ca să aflăm valoarea exactă a lui ***E(T)*** vom folosi formula:

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

$$\prod_{j=1}^{i-1} \alpha_j = produsul\ probablitășilor\ de\ trecere\ de\ la\ etapa\ i\ la\ i+1$$

$$\frac{1}{\lambda_i} = valoarea\ așteptată\ a\ timpului\ pentru\ etapa\ i$$

Această formulă este dedusă din mai multe (alte formule) şi anume:

- Valoarea aşteptată a sumei de variabile aleatoare: $E(T) = E(T_1) + E(T_2) + \cdots + E(T_k)$
- Probabilitatea de a ajunge la etapa i: $\prod_{j=1}^{i-1} \alpha_j$
- Contribuția fiecaruia la etapa i: $E(T_i) = \prod_{j=1}^{i-1} \alpha_j \cdot \frac{1}{\lambda_i}$
- Suma contribuțiilor: formula finală

Observăm ca cele doua rezultate obținute (valoarea aproximată si cea exactă) sunt foarte apropiate, diferența dintre ele fiind foarte mică.

```r

################### cerinta 2 ########################

exact_ET <- 0
for (i in 1:n) {
  prod_alpha <- if (i == 1) 1 else prod(alpha[1:(i-1)])  # produsul alpha1 * alpha2 * ... * alpha(i-1)
  exact_ET <- exact_ET + prod_alpha * (1 / lambda[i])
}

print(paste("Valoarea exacta a lui E(T) =", exact_ET))
print(paste("Dif intre valoarea exacta si cea simulata =", abs(exact_ET - mean_T)))



################### cerinta 3 ########################
```

```
>
> print(paste("Valoarea exacta a lui E(T) =", exact_ET))
[1] "Valoarea exacta a lui E(T) = 1.913027488"
> print(paste("Dif intre valoarea exacta si cea simulata =", abs(exact_ET - mean_T)))
[1] "Dif intre valoarea exacta si cea simulata = 0.00110748463285004"
>
```

Cerința 3:

Explicarea cerinței: aproximăm probabilitatea ca activitatea să fie finalizată (adică să ajungem la etapa n).

Am creat o funcție aproape identică cu cea de la cerința 1), dar în loc să returneze timpul total, aceasta va returna dacă activitatea a fost finalizată pentru toate cele 1000000 simulări ale lui ***T***. Astfel, am creat un vector logic ***comp*** care înregistrează toate valorile de TRUE/FALSE (finalizat/nefinalizat). Probabilitatea ca se calculează folosind funcția ***mean()*** (media aritemtică).

```r
################### cerinta 3 ########################

simulate_T_final <- function(n, lambda, alpha) {
  total_time <- 0
  completed <- TRUE  # presupun ca finalizeaza activitatea
  for (i in 1:n) {
    time <- rexp(1, rate = lambda[i])  # timp pentru etapa i
    total_time <- total_time + time
    if (runif(1) > alpha[i]) {
      completed <- FALSE  # nu a finalizat activitatea
      break
    }
  }
  return(completed)  # returneaza TRUE daca a finalizat, FALSE altfel
}

comp <- replicate(nrSim, simulate_T_final(n, lambda, alpha))
prob_final <- mean(comp)  # probabilitatea de finalizare
print(paste("Probabilitatea de finalizare a activitatii =", prob_final))
```

```
> comp <- replicate(nrSim, simulate_T_final(n, lambda, alpha))
> prob_final <- mean(comp)  # probabilitatea de finalizare
> print(paste("Probabilitatea de finalizare a activitatii =", prob_final))
[1] "Probabilitatea de finalizare a activitatii = 2.1e-05"
> 
```

Cerința 4:

Explicarea cerinței: calculăm probabilitatea ca T să fie ≤ σ (unde σ este un prag dat).

Alegem o valoare pentru σ (în cazul nostru 5 pentru a cuprinde cât mai multe cazuri). Vectorul ***T_values_sigma*** va conține doar valorile timpilor a căror activitate a fost finalizată. Ca să calculăm probabillitate cerută vom compara fiecare timp din ***T_values_sigma*** cu valoarea lui σ, returnând TRUE sau FALSE pentru fiecare comparație/valoare, iar cu ajutorul funcției ***mean()*** aceste valori de TRUE sunt numărate și mai apoi împărțite la numărul total de valori din vectorul ***T_values_sigma.***

Probabilitatea/rezultatul nostru a fost 1 deoarece am ales o valoare pentru σ destul de mare pentru a include toate cazurile.

```
################## cerinta 4 #########################

sigma <- 5  # valoarea lui sigma
# este ales mai mare sa acopere toate cazurile, deci prob o sa fie 1
T_values_sigma <- T_values[comp]
# filtreaza T_values folosind vectorul logic comp
# ia elementele de pe aceleasi pozitii iar daca in al doilea vector e true
# atunci pastreaza timpul lui T
prob_sigma <- mean(T_values_sigma <= sigma)
# compara fiecare T cu sigma is face med ar din val de true false <= sigma
print(paste("Probabilitatea ca T <= sigma:", prob_sigma))
```

```
> print(paste("Probabilitatea ca T <= sigma
[1] "Probabilitatea ca T <= sigma: 1"
>
```

Cerința 5:

Explicarea cerinței: determinăm timpul minim și maxim în care se finalizează activitatea și reprezentăm grafic toți timpii obținuți.

Minimul/maximul este luat din vectorul creat la cerința anterioară și pus in variabila ***min_T/max_T*** . Aceste valori sunt reprezentate cu ajutorul unei histograme.

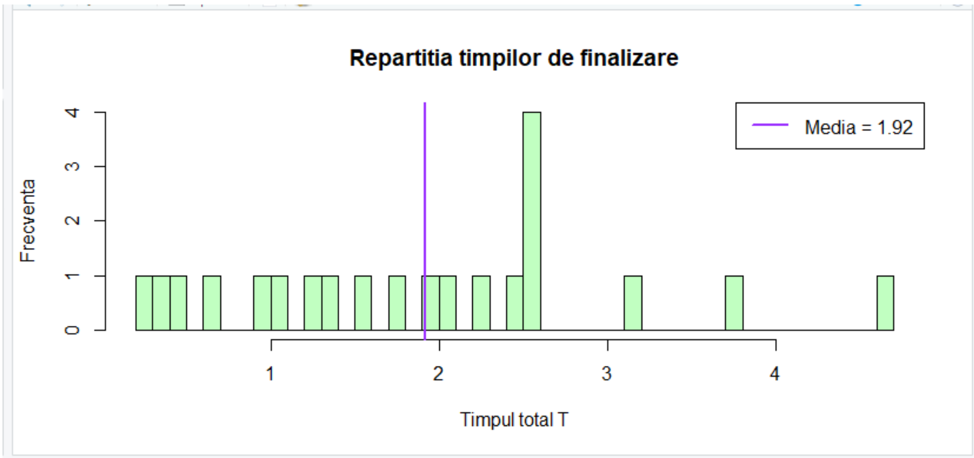
```
################## cerinta 5 #########################

min_T <- min(T_values_sigma)
max_T <- max(T_values_sigma)

print(paste("Timpul minim de finalizare=", min_T))
print(paste("Timpul maxim de finalizare=", max_T))

# reprezentare grafica a timpilor de finalizare
hist(T_values_sigma, breaks = 50, main = "Repartitia timpilor de finalizare", xlab = "Timpul total T", ylab = "Frecventa", col = "darkseagreen1")
abline(v = mean(T_values_sigma), col = "purple1", lwd = 2)  # linie pentru medie
legend("topright", legend = paste("Media =", round(mean(T_values_sigma), 2)), col = "purple1", lwd = 2)
```

```
>
> print(paste("Timpul minim de finalizare=", min_T))
[1] "Timpul minim de finalizare= 0.247252776406493"
> print(paste("Timpul maxim de finalizare=", max_T))
[1] "Timpul maxim de finalizare= 4.66314645086355"
>
> # reprezentare grafica a timpilor de finalizare
> hist(T_values_sigma, breaks = 50, main = "Repartitia timpilor
```



Ce putem spune despre repartiție?

- Distribuția este asimetrică, cu o coadă lungă la dreapta. Acest lucru înseamnă că există cazuri în care timpul de finalizare este mult mai mare decât media, dar acestea sunt mai puțin probabile.
- Vârful distribuției este în jurul valorii medii (1.92), ceea ce indică faptul că majoritatea timpilor de finalizare sunt concentrați în jurul acestei valori.
- Timpul minim de finalizare este aproape de 0, deoarece persoana A se poate opri foarte devreme (după prima etapă).
- Timpul maxim de finalizare este mai mare, reflectând cazurile în care persoana A parcurge toate etapele.

Cerința 6:

Explicarea cerinței: aproximăm probabilitatea ca activitatea să se oprească înainte de etapa k și reprezentăm aceste probabilități într-un grafic.

Am creat o funcție similară ca cea de la cerința 1), dar în loc să returneze timpul total, ea va returna etapa la care s-a oprit activitatea pentru fiecare ***T***. Aceste etape sunt reținute într-un vector stop_stages. Ca să calculăm probablitatea, am decis să iau pentru fiecare ***k***. Astfel, vectorul ***prob_stop_before_k*** conține toate probablitățile ca activitatea să se oprească înainte ***k*** pentru fiecare k de la ***1*** la ***n***. Reprezentarea grafică este dată de o linie, în loc de o histogramă.

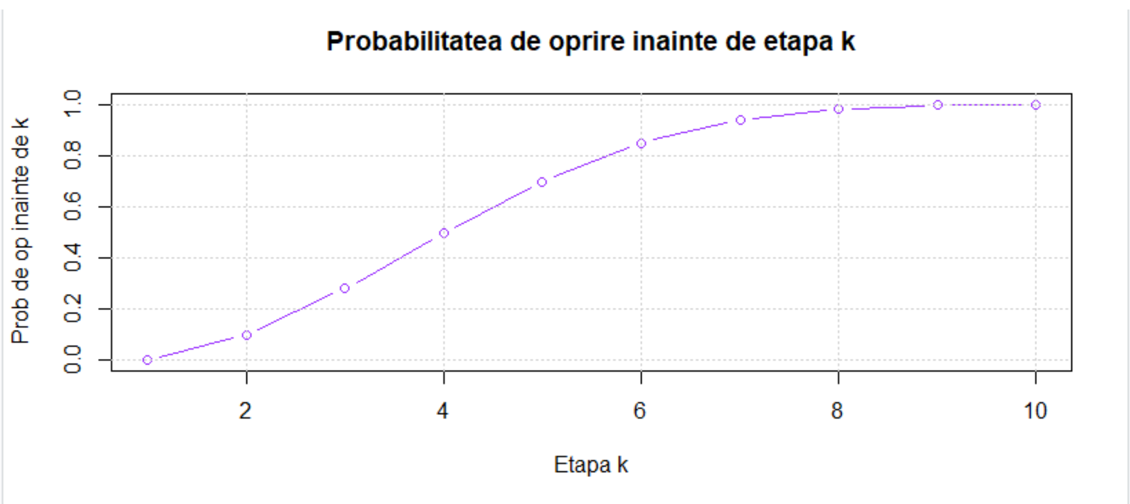
```r
################### cerinta 6 #########################

simulate_stop_stage <- function(n, lambda, alpha) {
  for (i in 1:n) {
    time <- rexp(1, rate = lambda[i])
    if (runif(1) > alpha[i]) {  # daca se opreste dupa etapa i
      return(i)  # returneaza etapa la care s-a oprit
    }
  }
  return(n)  # daca a trecut prin toate etapele
}
# functia care iti arata la ce etapa s-a oprit A

stop_stages <- replicate(nrSim, simulate_stop_stage(n, lambda, alpha))
# etapele la care s-a oprit A vector

prob_stop_before_k <- sapply(1:n, function(k) {
  mean(stop_stages < k)  # prob ca A sa se opreasca inainte de k
})
# se aplica pt toate k-urile de la 1 la n

# reprezentare grafica
plot(1:n, prob_stop_before_k, type = "b", col = "purple1",
     xlab = "Etapa k", ylab = "Prob de op inainte de k",
     main = "Probabilitatea de oprire inainte de etapa k")
grid()
```



Ce puteți spune despre repartiția probabilităților obținute?

- Aceste probabilități sunt crescătoare în raport cu ***k***, deoarece persoana A are mai multe șanse de a se opri pe măsură ce parcurge mai multe etape.
- Probabilitățile formează o curbă crescătoare.

- La $k = 1$, probabilitatea este 0, deoarece persoana A nu poate să se oprească înainte de prima etapă.
- La $k = n$, probabilitatea este maximă, deoarece persoana A poate să se oprească în orice etapă până la n.
- Graficul arată o curbă monoton crescătoare, care reflectă faptul că probabilitatea de oprire crește odată cu creșterea lui k.
- Panta curbei depinde de probabilitățile α_i. Dacă valorile alfa sunt mici, panta este mai abruptă.

Concluzie:

În concluzie, am analizat un proces format din mai multe etape, unde fiecare etapă are un timp de execuție aleator și o probabilitate de a continua spre următoarea. Proiectul arată cum putem folosi simulările pentru a înțelege mai bine procesele aleatorii și comportamentul lor în diverse situații.

Dificultățile în realizarea cerințelor:

Începutul a fost foarte greu neștiind de unde să încep, iar înțelegerea cerinței a durat mai mult timp decât mă așteptam. Găsirea soluțiilor cerințelor nu a fost una tocmai ușoară, fiind nevoie de documentație specială care nu a fost tocmai ușor de înțeles. În ciuda acestor dificultăți, am reușit finalizarea proiectului.

Sursele:

- Cursurile domnului profesor Niculescu Cristian
- https://alexamarioarei.quarto.pub/curs-ps-fmi/Introducere_R/Chapter_4/Elemente_grafica_R.html
- https://cran.r-project.org/doc/manuals/r-release/R-intro.html#Index-vectors

- https://www.math.uaic.ro/~maticiuc/didactic/Probability_Theory_Course_7_8_9_10.pdf
- https://en.wikipedia.org/wiki/Exponential_distribution
- https://en.wikipedia.org/wiki/Expected_value
- https://en.wikipedia.org/wiki/Markov_chain