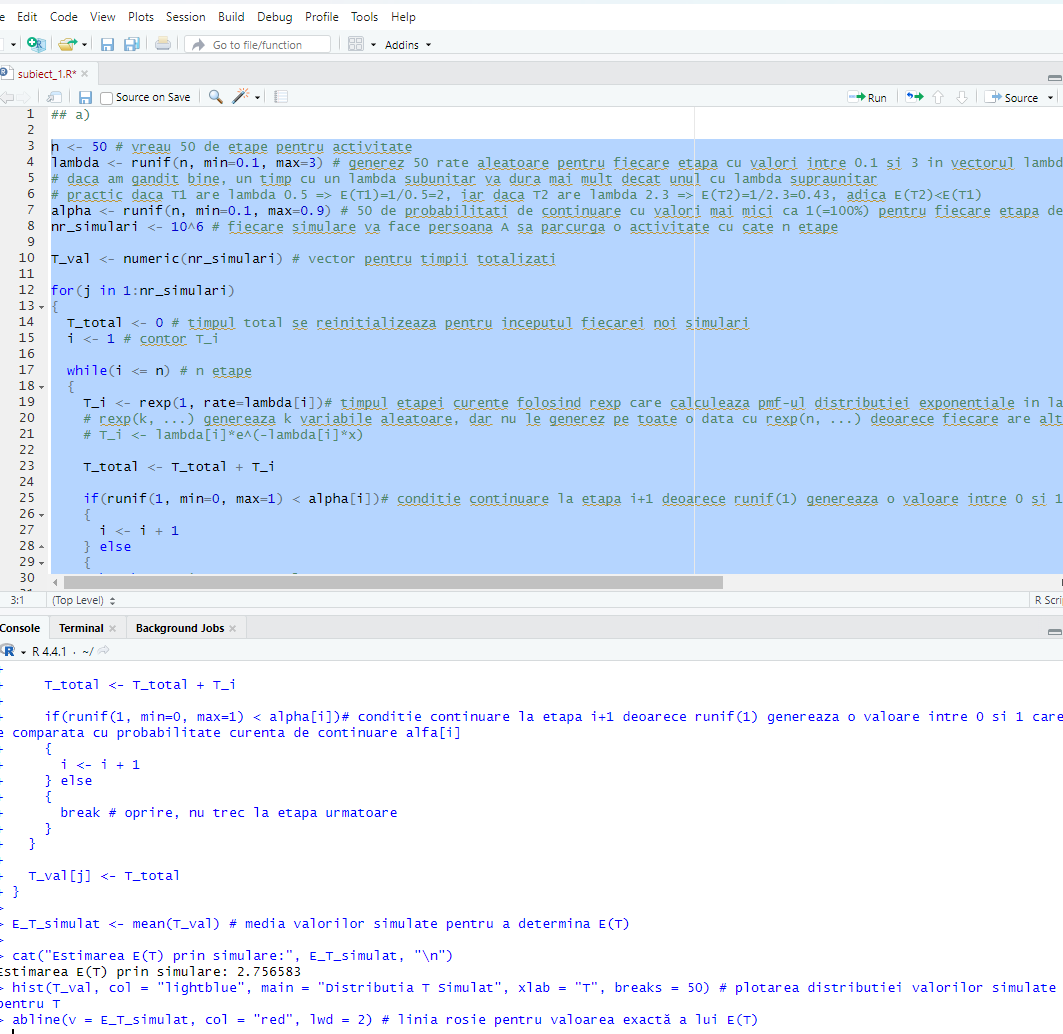
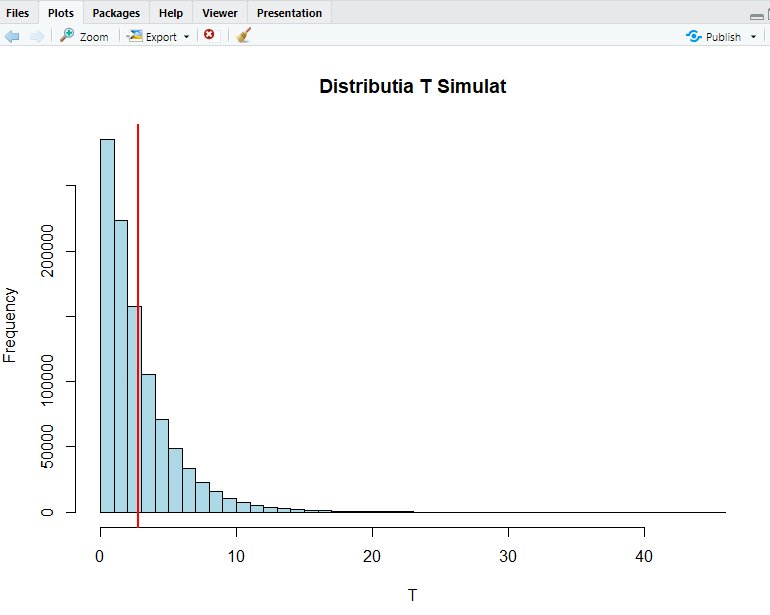
1) Construiți un algoritm in R care simuleaza valori pentru v.a. T si in baza acestora aproximațti . Reprezentati grafic intr-o manieră adecvata valorile obtinute pentru T. Ce puteți spune despre repartitia lui T?

Voi simula n etape cu rata corespunzatoare fiecarei etape si generez variabile aleatoare ce reprezinta distributia exponentiala (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 . In final calculez media valorilor simulate pentru a aproxima .





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)

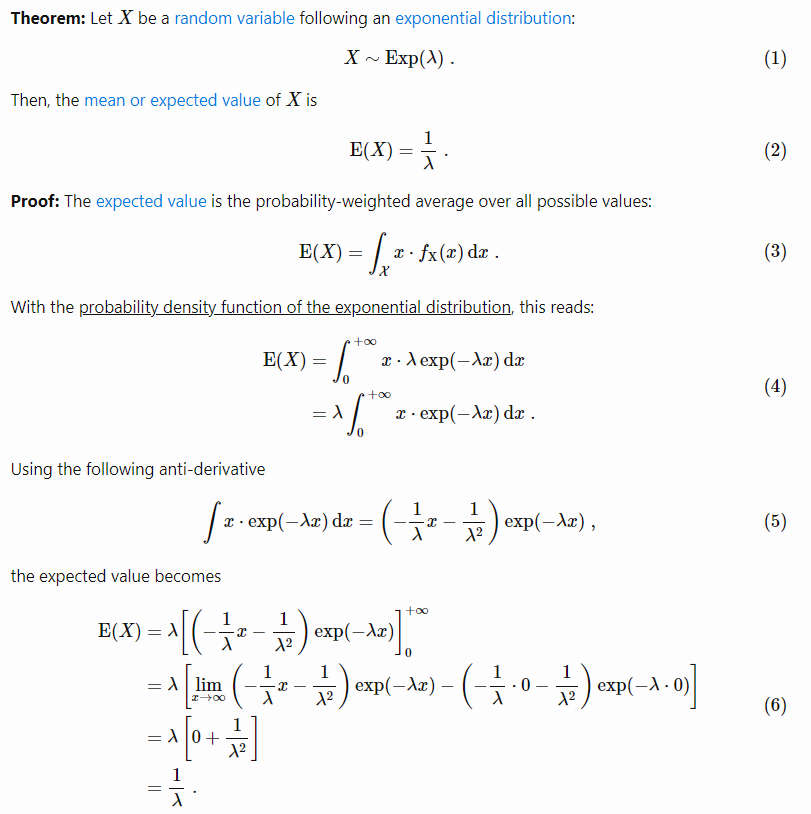
cat("Estimarea E(T) prin simulare:", E\_T\_simulat, "\n")

hist(T\_val, col = "lightblue", main = "Distributia T Simulat", xlab = "T", breaks = 50) # plotarea distributiei valorilor simulate pentru T

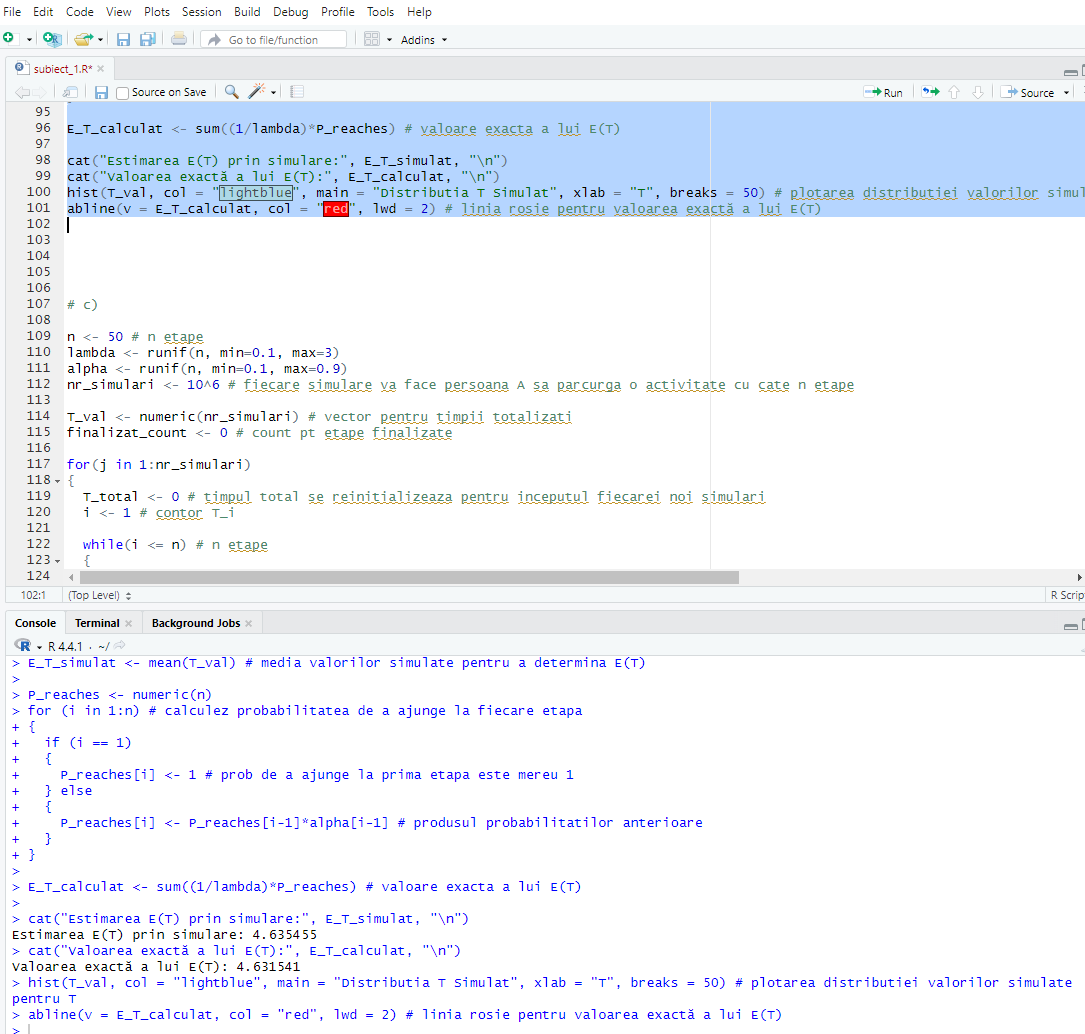
abline(v = E\_T\_simulat, col = "red", lwd = 2) # linia rosie pentru valoarea exactă a lui E(T)

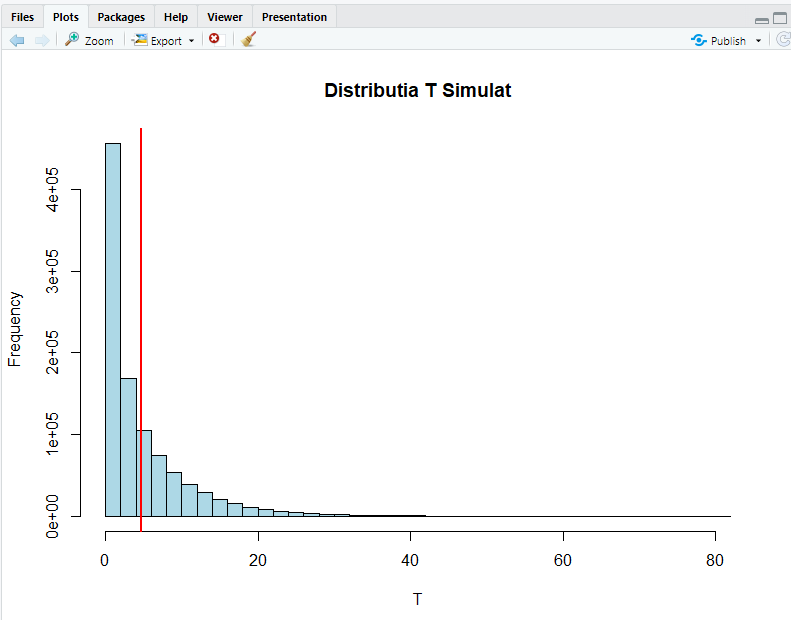
2) Calculati valoarea exacta a lui .

Media timpului total T pentru toate etapele poate fi calculata ca . Stiu ca are media din



Stiu ca , adica e produsul probabilitatilor anterioare.





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")

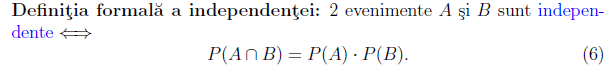
cat("Valoarea exactă a lui E(T):", E\_T\_calculat, "\n")

hist(T\_val, col = "lightblue", main = "Distributia T Simulat", xlab = "T", breaks = 50) # plotarea distributiei valorilor simulate pentru T

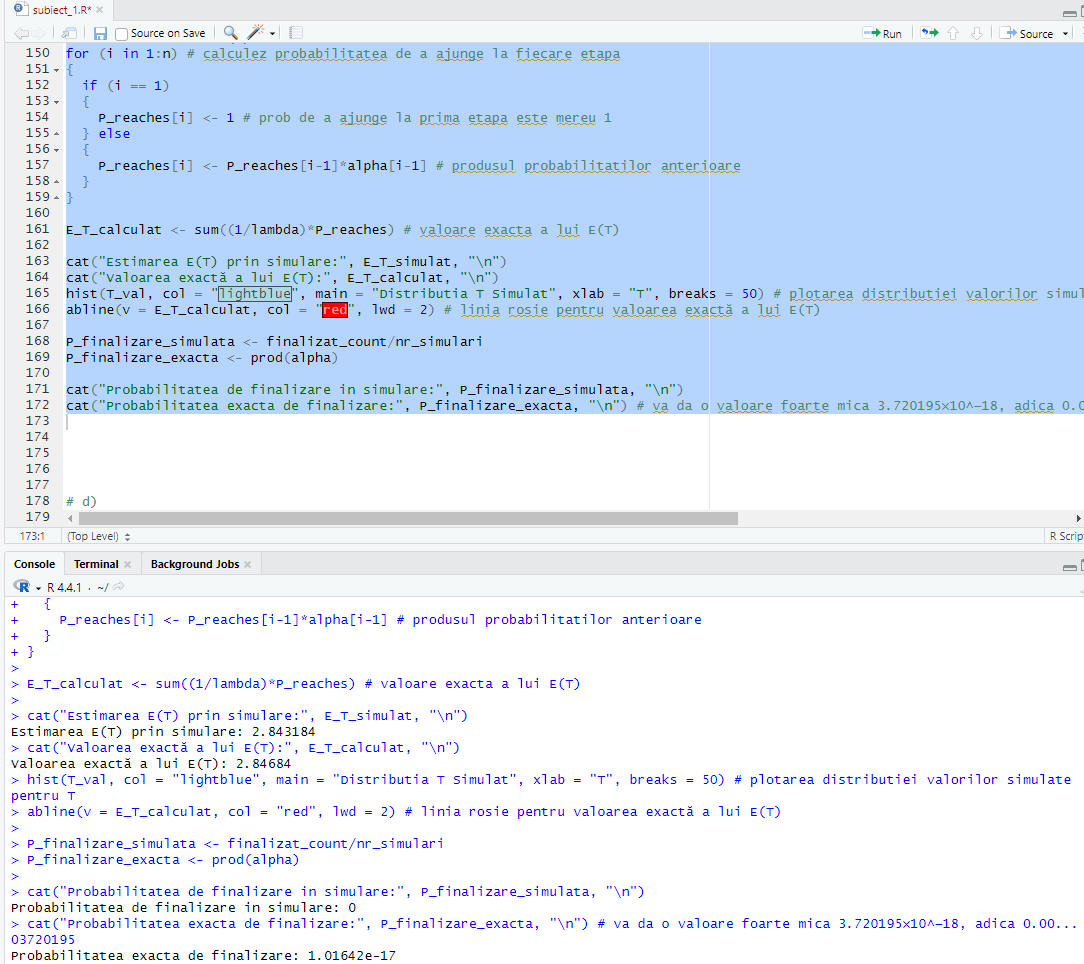
abline(v = E\_T\_calculat, col = "red", lwd = 2) # linia rosie pentru valoarea exactă a lui E(T)

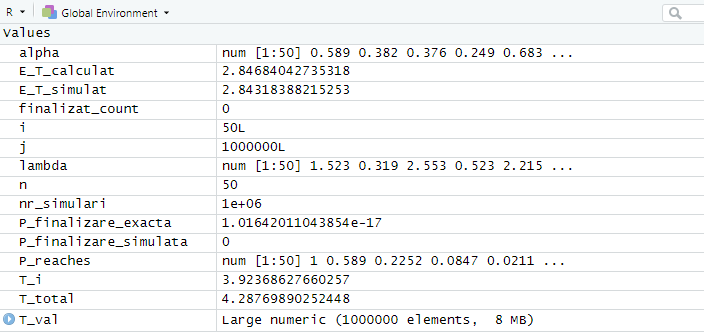
3) Aproximati probabilitatea ca persoana A sa finalizeze activitatea.

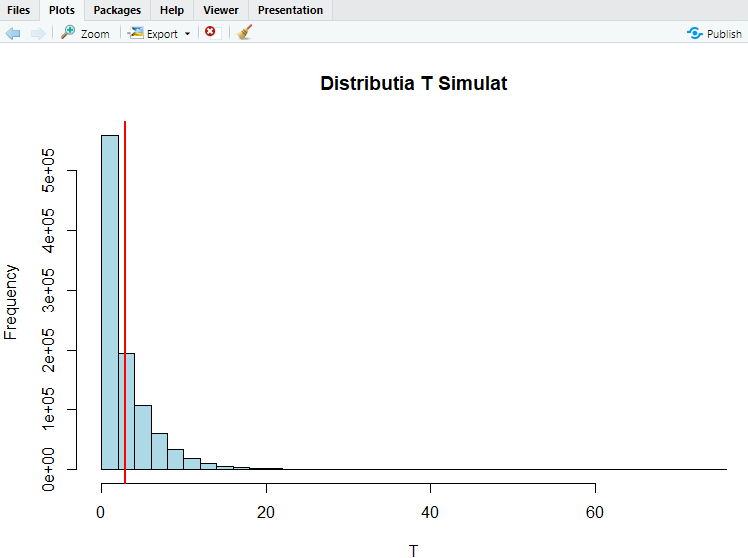
Probabilitatea ca persoana A sa finalizeze activitatea, adica sa fie parcurse toate n etapele, este de fapt produsul probabilitatilor de continuare pentru fiecare etapa. E datorita ca fiecare etapa este un eveniment independent (timpul petrecut la o etapa i pentru terminarea etapei nu afecteaza timpul altei 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).



Deci







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")

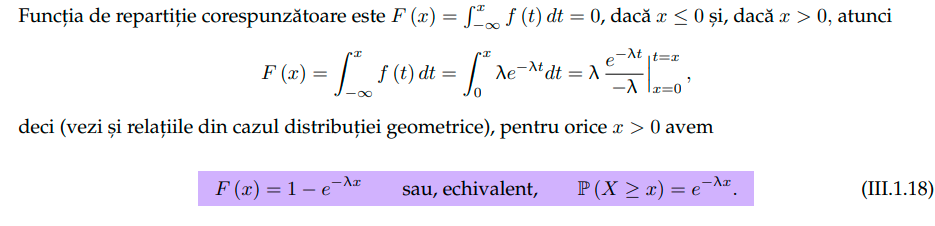
cat("Probabilitatea exacta de finalizare:", P\_finalizare\_exacta, "\n") # va da o valoare foarte mica 3.720195×10^−18, adica 0.00...03720195

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

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

In realitate, probabilitatea ca timpul total de finalizare este mai mic sau egal cu  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 distribuita exponențial cu parametru , adica voi folosi pentru timpul unei singure etape.

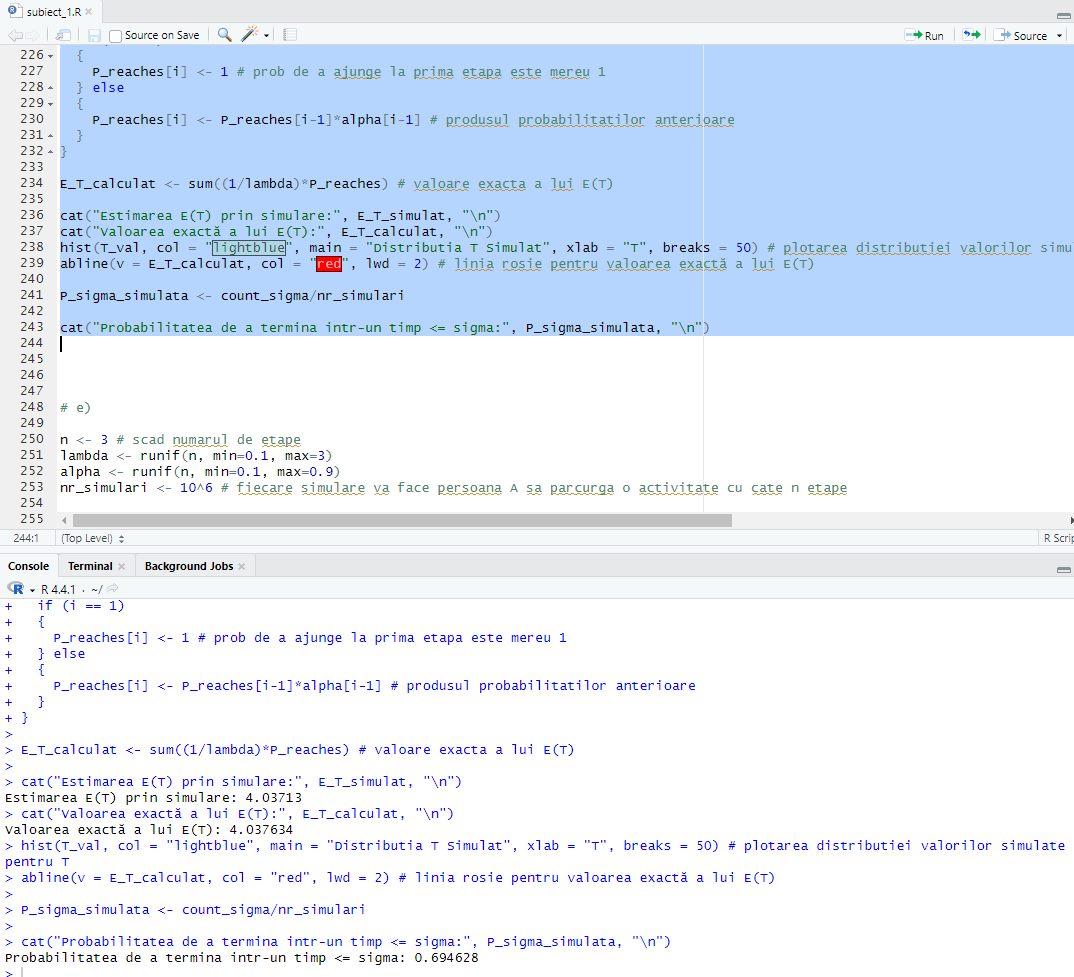
Daca timpul total de finalizare T este suma timpilor pentru fiecare etapa, iar fiecare este independent, atunci CDF-ul pentru suma acestor variabile aleatoare nu este direct exponential, dar poate fi calculat prin metode numerice sau simulari. Deci simulez si retin numarul de variabile cu timpul si ma folosesc de logica asta: .

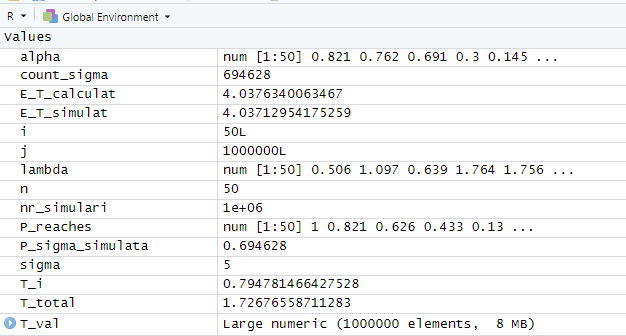


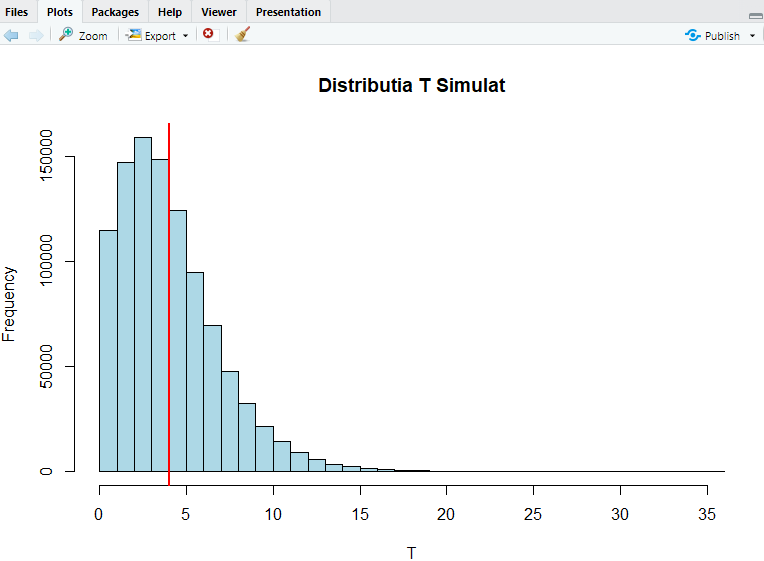
Practic cele 2 etape sunt:

1) Pentru fiecare simulare, verificati daca timpul total este mai mic sau egal cu

2) Proportia de simulari in care timpul total este mai mic sau egal cu







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)

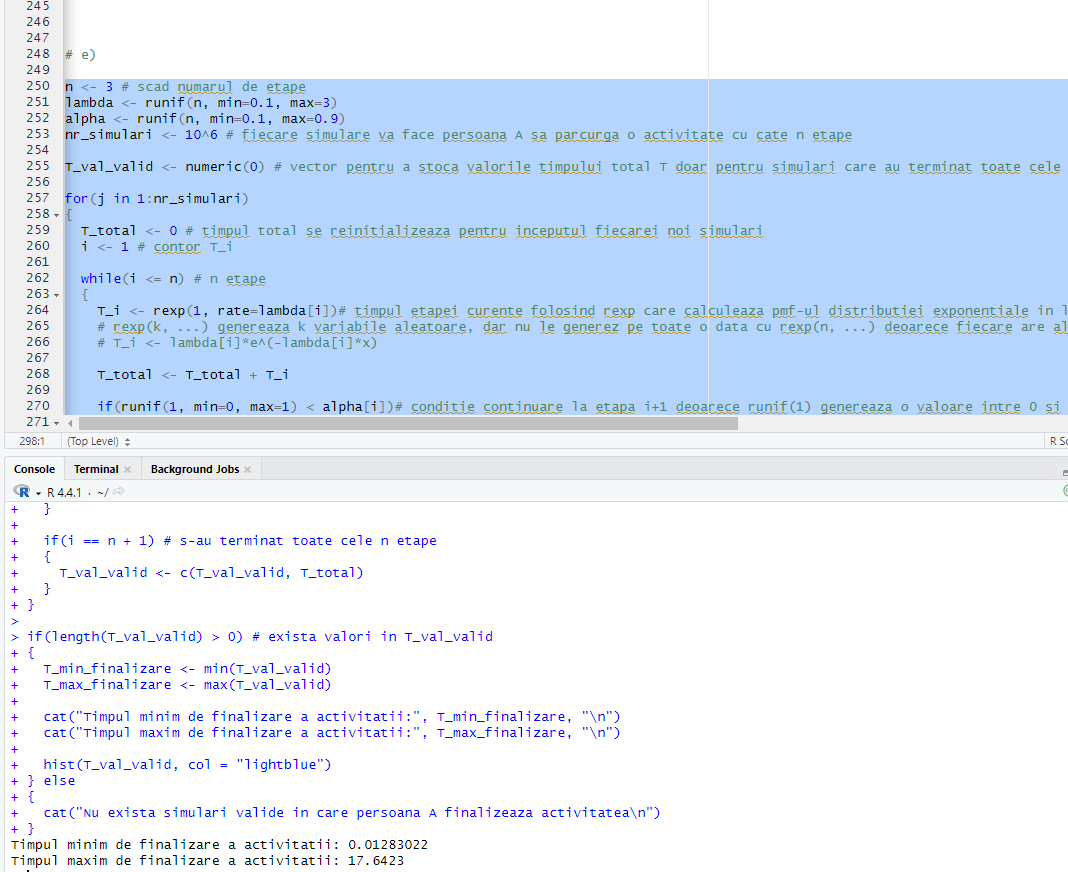
P\_sigma\_simulata <- count\_sigma/nr\_simulari

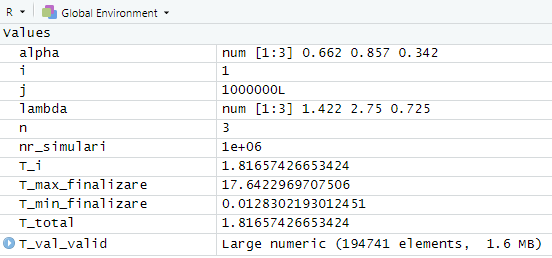
cat("Probabilitatea de a termina intr-un timp <= sigma:", P\_sigma\_simulata, "\n")

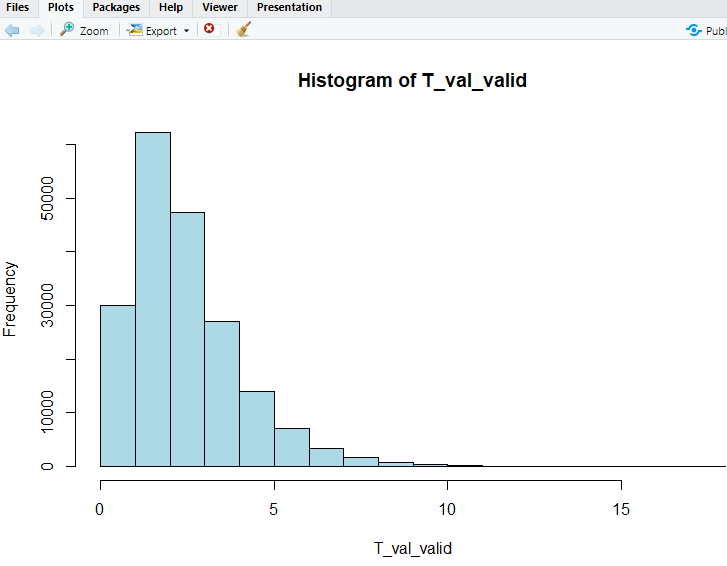
5) Determinați timpul minim si respectiv timpul maxim in care persoana A finalizeaza activitatea si reprezentati grafic timpii de finalizare a activitatii din fiecare simulare. Ce puteti spune despre repartitia acestor timpi de finalizare a activitatii?

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() și max() pe vectorul T\_val\_valid.

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







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

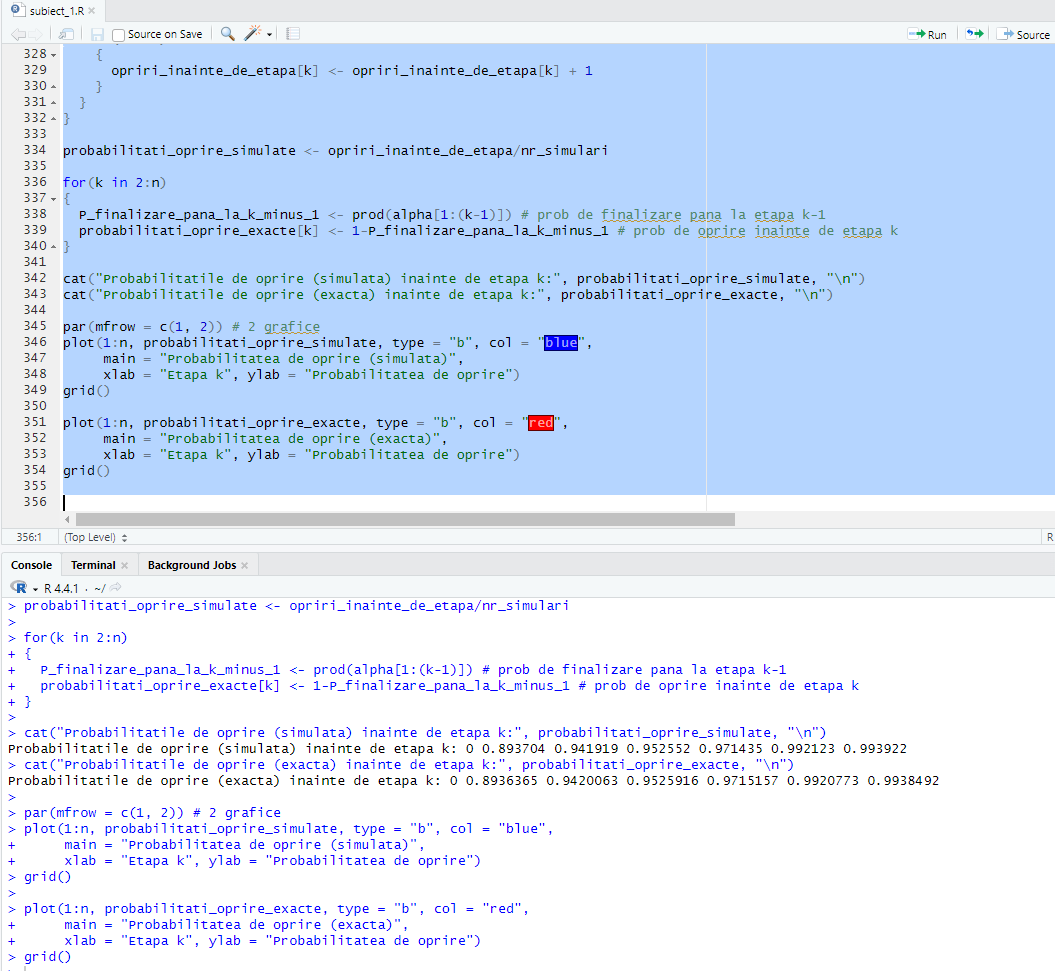
{

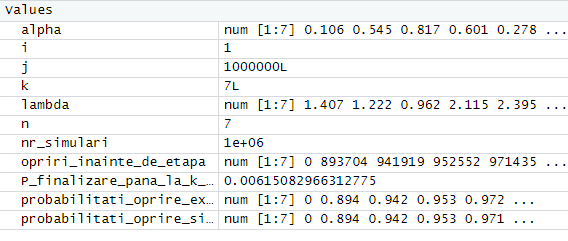
cat("Nu exista simulari valide in care persoana A finalizeaza activitatea\n")

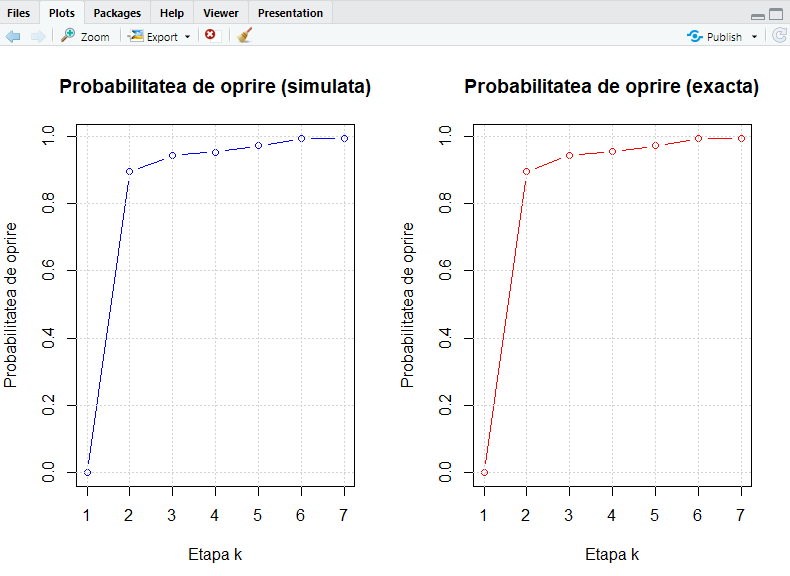
}

6) Aproximati probabilitatea ca persoana A sa se opreasca din lucru inainte de etapa , unde . 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 este data de complementul probabilitatii ca activitatea sa fie finalizata pana la etapa , iar de la subpunctul 3) se deduce , deci







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()

Referinte:

<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>