Documentatie proiect Probabilitati si Statistica

Membrii echipei
Cazacu Cristian – Gabriel
Tanislav Alexia
Velcea Mihnea-Andrei

Problema I

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

1) Construiți un algoritm în R care simulează 10^6 valori pentru v.a. *T* și în baza acestora aproximați *E(T)*. Reprezentați grafic într-o manieră adecvată valorile obținute pentru *T*. Ce puteți spune despre repartiția lui *T*?

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 *A* să finalizeze activitatea.

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

5) În baza simulărilor de la 1) determinați timpul minim și respectiv timpul maxim în care persoana *A* finalizează activitatea și reprezentați grafic timpii de finalizare a activității din fiecare simulare. Ce puteți spune despre repartiția acestor timpi de finalizare a activității?

6) În baza simulărilor de la 1) aproximați probabilitatea ca persoana *A* să se oprească din lucru înainte de etapa 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 *T*, apoi calculăm media lor pentru a aproxima *E(T)*. Reprezentăm grafic valorile lui *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 același 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 α_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()*.

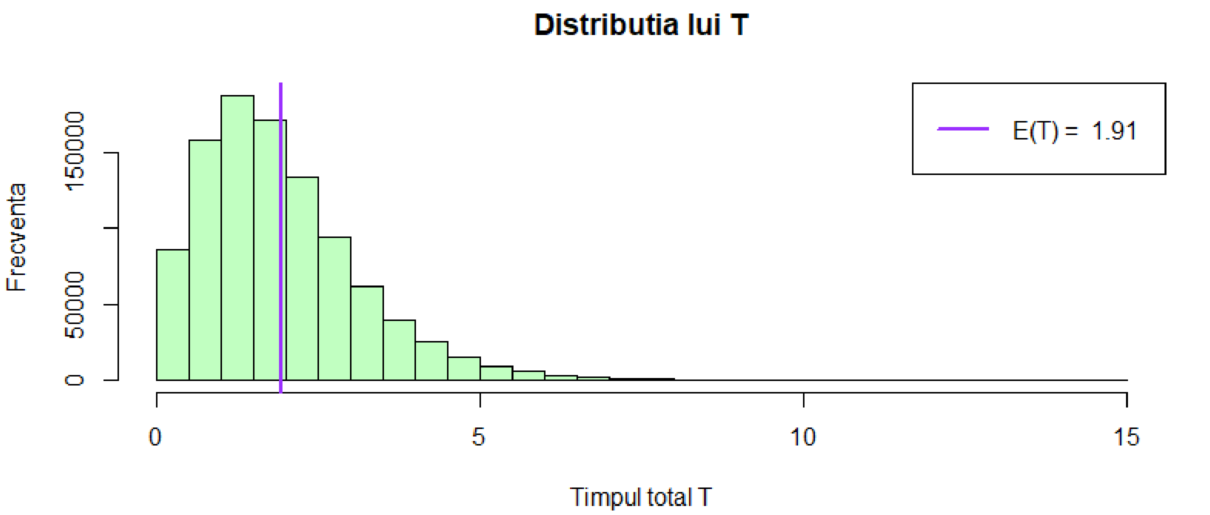
```r
set.seed(123)  # pt aceleasi val random cand dai run
n <- 10  # nr de etape
lambda <- c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)  # parametrii lambda pentru fiecare etapa
alpha <- c(0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1, 0.05)  # prob de trecere la etapa urm
# alfa de i este prob ca persoana A sa treaca 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 $T_i \sim Exp(\lambda_i)$, iar suma acestor timpi este condiționată de probabilitățile α_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 α_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, $T \sim Exp(\lambda)$.
  - → Dacă parcurge toate etapele, *T* este suma a *n* variabile exponențiale cu parametrii $\lambda_1, \lambda_2, \dots \lambda_n$

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:

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

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

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

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

- Valoarea așteptată 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.

```r
################## 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.

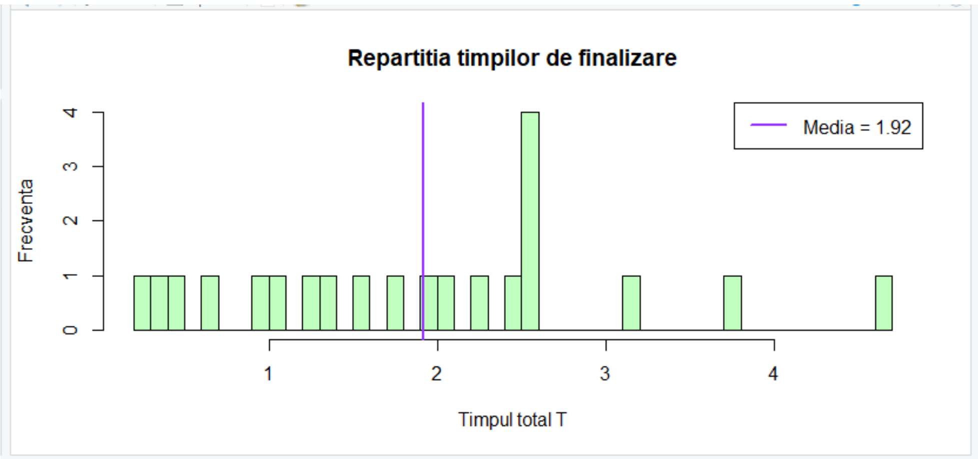
```
################## cerinta 5 ########################

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.

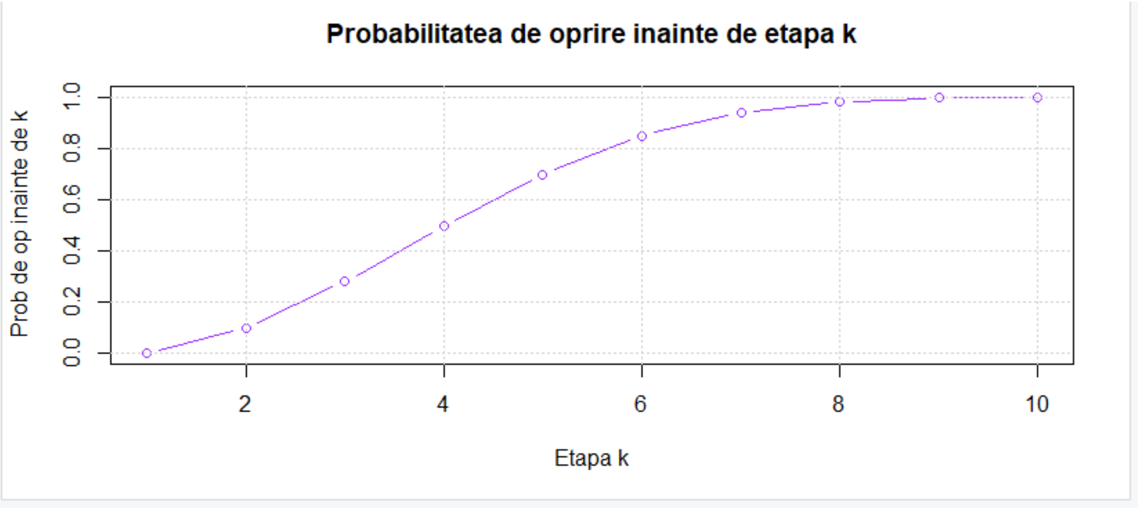
```r
################### cerinta 6 #########################

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.

Problema II

II. Construiti o aplicatie Shiny in care:

A. Sa reprezentati grafic functiile de repartitie pentru urmatoarele v.a:

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

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

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

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

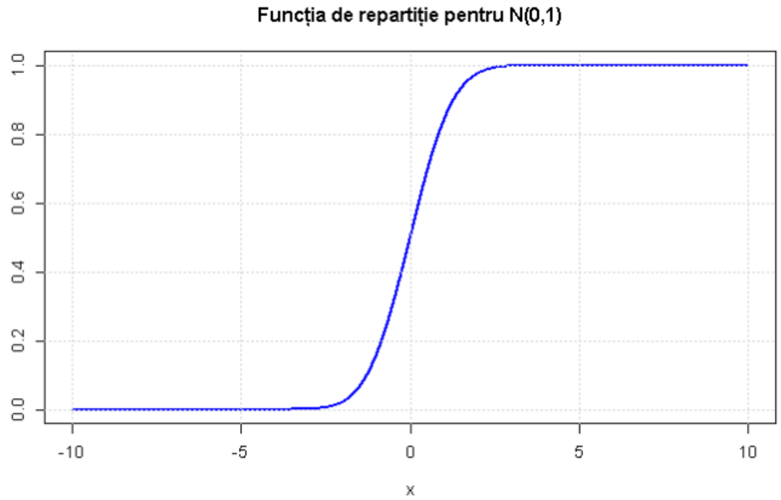
5) $X,\ 5X-4, X^3,\ \sum_{i=1}^{n} X_i$, unde $X,\ X_1, X_2 ... X_n\ i.i.d.\ \sim Binom(r, p), r \in \mathbb{N},\ p \in (0,1)$, $n \in \mathbb{N}\ fixat$

Functia de repartitie a unei v.a este F(x) = P(X <= x).

1) X fiind N(0,1)

Pasii de rezolvare:

i) Pentru densitatile definite pe toata multimea numerelor reale vom discretiza un interval relevant pentru graficul functiei de repartie.
Ex:



Intrucat F(x) pentru N(0,1) incepe sa creasca din aproximativ -3 si se opreste in 3, este suficient sa surprind un interval apropiat de acesta pentru grafic, nu intregul domeniu.

Functia R corespunzatoare: **x <- seq(-10, 10, 0.001)**

Intervalul trebuie discretizat deoarece [-10,10] contine o infinitate de numere, iar un computer poate lucra numai cu o multime finita. Am ales **0.001** gradul de granularitate pentru a obtine un grafic cat mai precis.

Pentru a calcula functia de repartitie folosim: **pnorm(x, mean = 0, sd = 1) din R,** iar plotarea graficului o facem folosind functia **plot** ce primeste mai multe argumente pentru a personaliza graficul.

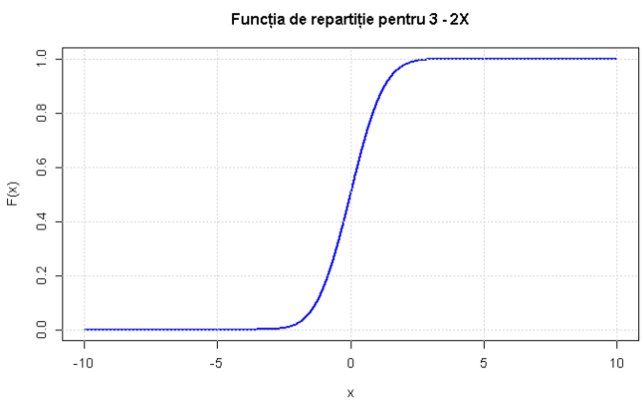
ii) 3 -2X, unde X este N(0,1)

Y = 3-2X, F(y) = P(Y<=y)=P(3-2X<=y)=P(X>=(3-x)/2)=1-Fx((3-x)/2)

Pentru calcularea lui Y=3-2X ne vom folosi de functia de reparitie a lui Fx.

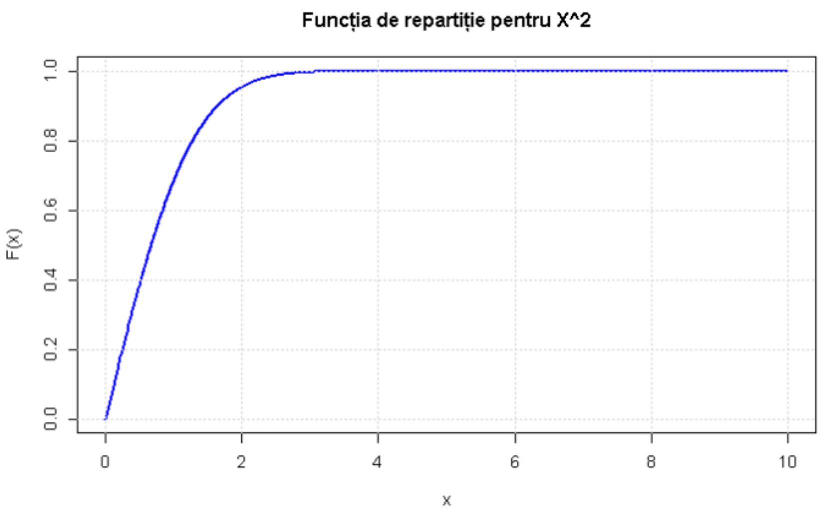
Daca domeniul pe care am reprezentat graficul lui X este [-10,10], pentru a putea compara cu graficul lui Y cu X vom recalcula capetele intervalului.

3-x / 2 = -10 ⇔ **x = 23**, 3-x /2 = 10 ⇔ **x = -17 => Noul domeniu este [-17,23]**



iii) X^2 , unde X este N(0,1)

Y= X^2, F(y) = P(Y<=y) = P(X^2<=y) = P(-sqrt(y) <= X <= sqrt(y)) = Fx(sqrt(y)) – Fx(-sqrt(y))



iv) Pentru X1, X2, X3, .. ,Xn i.i.d de repartitie N(0,1)

Xi de repartitie N(0,1) unde N(mu = 0, sd^2 = 1) => **E[Xi] = 0, Var[Xi] = 1**

Sn = X1 + X2 + ... + Xn

E[Sn] = E[X1] + E[X2] + ... + E[Xn] (din proprietati) = **n * 0 = E[Sn]**

Var[Sn] = Var[X1] + Var[X2] + ... + Var[Xn] (sunt independente) = **n * 1 = Var[Sn]**

Deci **Sn este N(0,n)** si functia de repartitie o calculam cu **pnorm(x, mean = 0, sd = sqrt(n))**

v) Pentru X1, X2, X3, .. ,Xn i.i.i de repartitie N(0,1)

Qn = X1^2 + X2^2 + ... + Xn^2

Var[Xi] = E[Xi^2] – E[Xi]^2 => E[Xi^2] = Var[Xi] + E[Xi]^2 = 1 + 0^2 = **1 = E[Xi^2]**

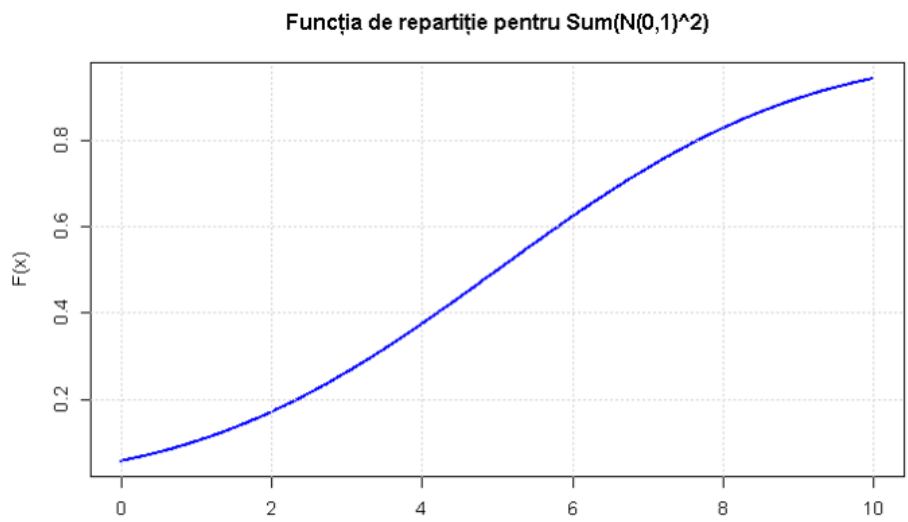
Var[Xi^2] = E[Xi^4] – E[Xi^2]^2 = 3 – 1 = **2 = Var[Xi^2]**

Obs: Pentru distributia normala standard momentele pare sunt cunoscute E[X^4] = 3

E[Qn] = E[X1^2] + E[X2^2] + ... + E[Xn^2] = 1 * n = **n = E[Qn]**

Var[Qn] = Var[X1^2 + Var[X2^2] + ... + Var[Xn^2] = **2*n = Var[Qn]**

Deci **Qn este de distributie N(2, 2n)** si folosim **pnorm(x, n, sqrt(2*n))**



Grafic generat pentru n = 5

Obs: Tehnicile prezentate anterior se vor regasi si la subpunctele urmatoare asa ca voi adauga doar acele puncte care indica un nivel mai mare de dificultate.

2) X, X1, X2, ..., Xn i.i.d de N(mu, sigma^2)

Sn = X1 + X2 + ... + Xn

E[Sn] = E[X1] + E[X2] + … + E[Xn] = n * mu

Var[Sn] = Var[X1] + Var[X2] + … + Var[Xn] = n * sigma^2

Deci **Sn are repartitia N(n*mu, n * sigma^2)**

Qn = X1^2 + X2^2 + … + Xn^2

E[Xi^2] = Var[Xi] + E[Xi]^2 = sigma^2 + mu^2

E[Qn] = E[X1^2] + E[X2^2] + … + E[Xn^2] = n * mu^2 + n * sigma^2

[Folosim formula E[X^4] =3 sigma^4 + 6mu^2sigma^2 + mu^4]

Var[Xi^2] = E[Xi^4] – E[Xi^2]^2 =

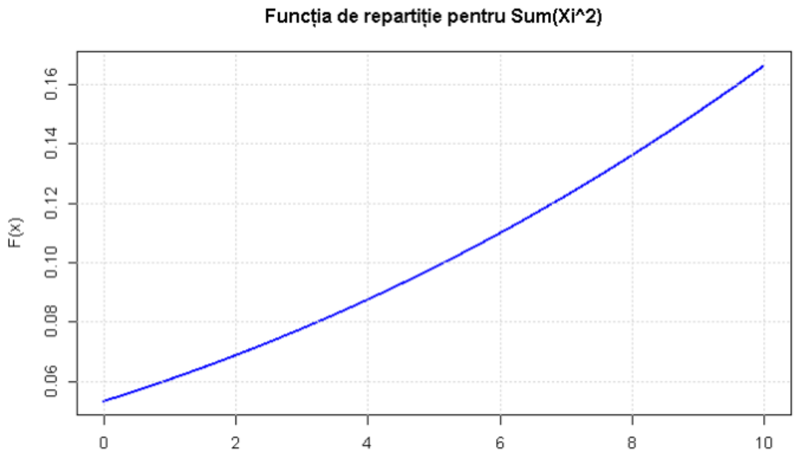
Obs: In continuare voi prescurta **sigma** cu **s**

= 3*s^4 + 6*mu^2*s^2 + mu^4 – s^4 – 2*s^2*mu^2 – mu^4 =

= 2*s^4 + 4*mu^2*s^2 = Var[Xi^2]

Deci Var[Qn] = n*(**2*s^4 + 4*mu^2*s^2**)

Concluzie: Qn e de repartitie **N(n*(mu^2 + s^2), n*(2*s^4 + 4*mu^2*s^2))**



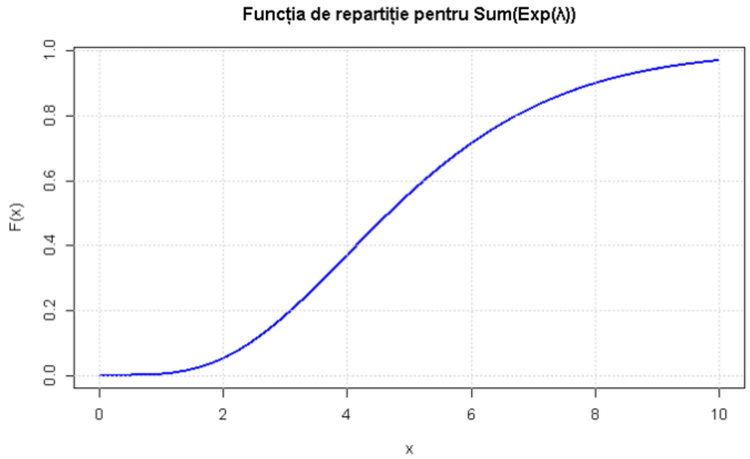
Grafic generat pentru n = 5, mu = 1, sd = 2

3) Pentru subpunctul cu densitatea functie **exponentiala** vom folosi pentru calcularea functiei de repartitie **functia din R: pexp(x, rate = lambda).**

In categoria aspectelor teoretice care depasesc nivelul_cursului_ si au ajutat la rezolvarea problemei voi incadra folosirea Transformatei LaPlace:

$$F(s) = \mathcal{L}\left\{f(t)\right\} = \int_{0^-}^{\infty} e^{-st} f(t)\, dt.$$

Valoarea transformatei cand f(t) este pdf exponentiala este: z =lambda/(s + lambda), iar pentru ca X1, X2, ..., Xn sunt independente transformata lui Sn = z ^ n este egala cu cea a functiei Gamma(n, lambda), motiv pentru care Sn are distributia Gamma(n, lamda) si vom folosi **pgamma(x, shape =n, rate = lambda)** pentru a calcula functia de repartitie.



Grafic generator pentru n = 5, lambda = 1

4) Pentru a gasi repartitia lui Sn, atunci cand Xi = Poisson(lambda) ne vom folosi de alt concept din afara cursului: **functia generatoare de probabilitati (MGF).**

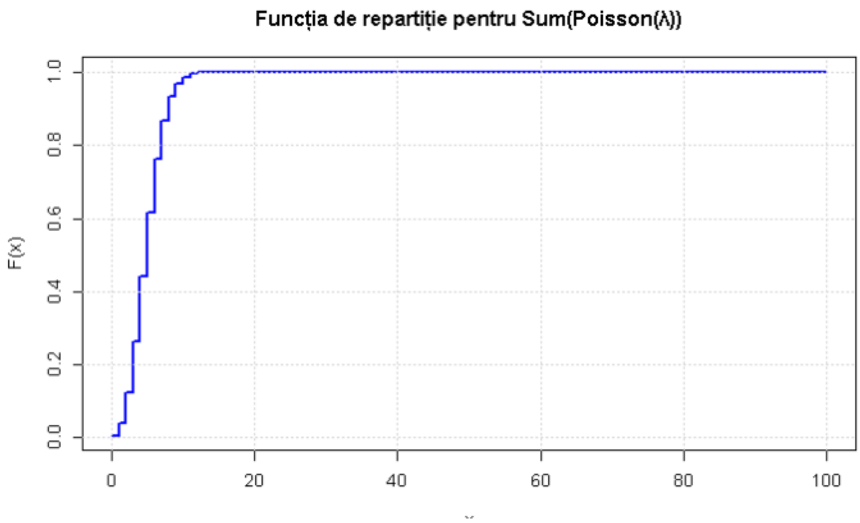
Pentru X ~ Poisson(lambda), Mx(t) = e^lambda(e^t-1).

Pentru n variabile independente Xi ~ Poisson(lambda) vom avea:

Msn(t) = Mx1(t)* Mx2(t) * ...* Mxn(t) = e^n*lambda(e^t-1),

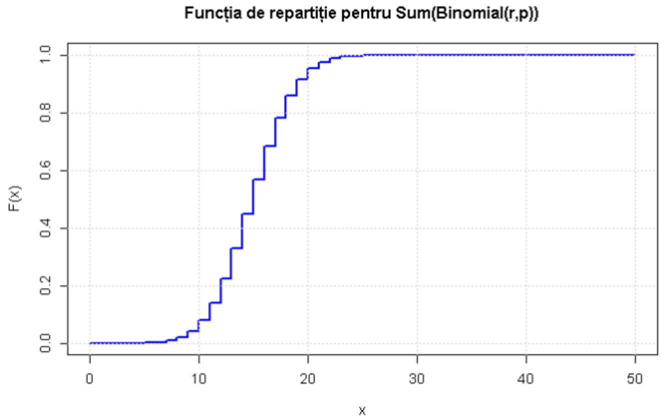
adica aceasta este functia generatoare de probabilitati a unei variabile X ~Poisson(n*lambda)

In concluzie **Sn ~ Poisson(n*lambda).**



Grafic generat pentru n = 5, lambda = 1

5) Pentru repartitia lui Sn, atunci cand Xi ~ Binomiala(r,p), vom alege un rationament mai intuitiv. Daca o variabila X~Binomial(r,p) reprezinta numarul de succese din r incercari, atunci daca avem n astfel de variabile, suma lor va reprezenta numarul de succese din n*r incercari, astfel Sn~Binomial(n*r,p).



Grafic generat pentru n = 5, r = 10, p = 0.3

In continuare vom discuta rezolvarile pentru partea a doua a problemei.

B. In aplicatie constuiti cate o functie in R care afiseaza functia pentru parametri particularizabili de catre utilizator si calculeaza, media si varianta pentru v.a X definita:

a) $f(x) = cx^4, \ x \in (0,2), \ c \in \mathbb{R}$

Pentru aceasta problema nu putem sa alegem noi constanta c pentru ca f este desitate de probabilitate cu proprietatile:

1. f(x) >= 0, oricare x din (0,2) => c >=0
2. Integrala (-Inf, Inf) f(x)dx = 1 => Integrala(0,2) cx^4 = 1 => **c = 5/32**

In cadrul aplicatie am lasat utilizatorul sa introduce ce valori considera pentru c, dar aplicatia va computa media si varianta, doar pentru densitatea valida.

Problema propune scrierea unei functii care sa afiseze functia si sa calculeze media, respectiv varianta. Pentru functiile de la aceasta problema am decis sa compartimentez functia in 3 functii: **una pentru afisare**, **una pentru calcularea mediei si variantei si una pentru afisarea rezultatelor.** Aceasta abordare ajuta la o citire si urmarire a programului mai usoara.

i) Afisarea functiei:

Scriem functia ce trebuie plotata folosind keyword-ul **function(x) .**

f <- function(x) input$numerator/input$denominator * x^4

Obs: Am ales sa impart c in numarator si impartitor pentru accesibilitate, pentru a fi mai usor introdus in interfata aplicatiei.

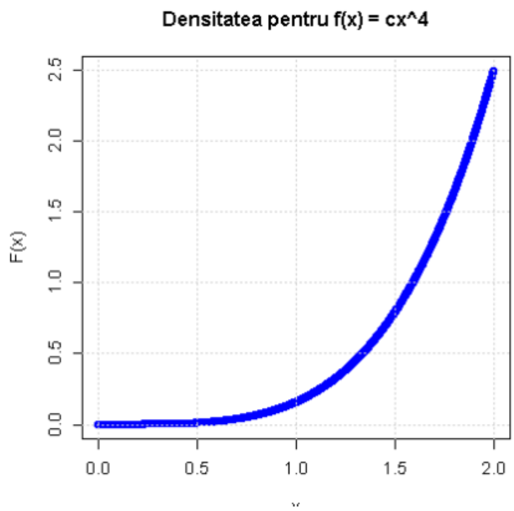
Tinand cont ca functia este definita pe intervalul (0,2), asa ca vom folosi din nou pasul de distretizare, utilizat si la problema cu functiile de repartitie.

interval <- seq(0.001, 1.999, 0.001)

Inainte de plotare vom aplica functia pe valorile intervalului folosind functia **sapply.**

f_x <- sapply(interval, f)

In cele din urma folosim functia **plot(interval, f_x ...)** pentru a afisa graficul.



Graficul functiei f cand c = 5/32

ii) Calcularea mediei si variantei

Media E[X] = Integrala(0, 2) x*f(x) dx

Varianta Var[X] = E[X^2] – E[X]^2

E[X^2] = Integrala(0,2) x^2 * f(x) dx

Functia pe ca o vom folosi pentru a calcula integralele in R este **integrate.**

Ex (din cod):

#Medie E(x)

average <- integrate(function(x) x * f_x_A()(x), 0, 2)$value unde:

- **f_x_A este densitatea variabilei** care este **inmultita cu x** pentru calcularea mediei
- **0, 2 este intervalul de integrare**

Media patratica se calculeaza analog, iar varianta se calculeaza prin scadere intre cele doua, respectand formula.

iii) Verificam daca densitatea este valida.

Calculam valoarea integralei pe tot domeniul pentru densitate:

#Integrala de validare

integral_f <- integrate(f_x_A(), 0, 2)$value, unde f_x_A este densitatea de prob.

Pentru o densitate corecta ea va fi aproximativ 1.

Inainte de afisare testam validitatea densitatii:

if (abs(stats$integral_f - 1) > 1e-6), facand abstractie de stats, verificam daca integrala este aproximativ 1 luand in considerare eroare de rotunjire (poate fi 0.999999).

In continuare plotam rezultate folosind functia **cat.**

```
Media E[X] = 1.666667
Varianța Var(X) = 0.07936508
```

*Media si varianta pt 5/32*x^4*

```
C nu face ca f(x) să fie o densitate validă!
Integrală = 1.2
```

Mesaj de eroare pentru densitatea invalida

Pentru subpunctele urmatoare conceptele prezentate anterior vor fi din nou folosite, cu mici modificari, motiv pentru care voi detalia doar problemele care adauga un nivel de noutate sau sunt mai dificile.

b) $f(x) = ax + bx^2, \; 0 < x < 1, \; a, b \in \mathbb{R}$

In cadrul acestui punct, in functie de a si b, exista mai multe densitati valide.

Ne vom folosi din nou de proprietatile densitatii de probabilitate:

- ax + bx^2 >= 0, oricare x din (0,1)
  Avem cazurile:

a>0, b>0, at pt orice a,b f(x) > 0

Cazuri favorabile: a =0, b >= 0; b =0, a >= 0

Cazuri nefavorabile: a=0, b<0; b=0, a<0; a<0, b<0

Raman cazurile a >0, b<0 si a < 0, b> 0

I. a>0, b<0, x din (0,1)

ax + bx^2 >=0 => x(a +bx) >= 0 => a + bx >= 0 ⇔ x <= -a/b ⇔ 1 <= -a/b ⇔ **a >= -b**

Vrem ca -a/b >= x pe tot intervalul (0,1), deci e suficient sa fie >= 1

II. **a < 0, b > 0, x din (0,1)**

ax + bx^2 >=0 => x(a +bx) >= 0 => a + bx >= 0 ⇔ x >= -a/b ⇔ 0 >=-a/b ⇔ **-a <0 imposibil a<0**

Vrem ca x >= -a/b pe tot domeniul (0,1), deci e suficient sa comparam cu minimul

- Integrala(0,1) (ax+bx^2) dx = 1, din calcule => ax^2/2 + bx^3/3|(0,1) = 1 ⇔ ⇔ a/2 + b/3 = 1 ⇔ **a = (6-2b)/3**

Folosind noile conditii de la f(x) >=0 le vom adauga la calculul integralei pe domeniu.

```
if (abs(stats$integral_f - 1) > 1e-6 & check) {
  cat("A, B nu fac ca f(x) să fie o densitate validă!\n")
  cat("Integrală =", stats$integral_f, "\n")
  check <- FALSE
}

if(check & a < 0){
  cat("A, B nu fac ca f(x) să fie o densitate validă!\n")
  check <- FALSE
}

if(check & b < 0){
  if(a == 0 | a <= 0){
    cat("A, B nu fac ca f(x) să fie o densitate validă!\n")
    check <- FALSE
  }
  else{
    if(a < -b)
    {
      cat("A, B nu fac ca f(x) să fie o densitate validă!\n")
      check <- FALSE
    }
  }
}
```

Cod care verifica f(x) >=0 , pt a,b alesi de utilizator

c) $f(x) = \dfrac{4}{x(x+1)(x+2)}, \quad x \in \mathbb{N}^*$

Pentru punctul c), avem prima v.a discreta. Vom calcula media si varianta astfel:

E[X] = x * p(x), unde f este pdf pentru X

E[X^2] = x^2 * p(x)

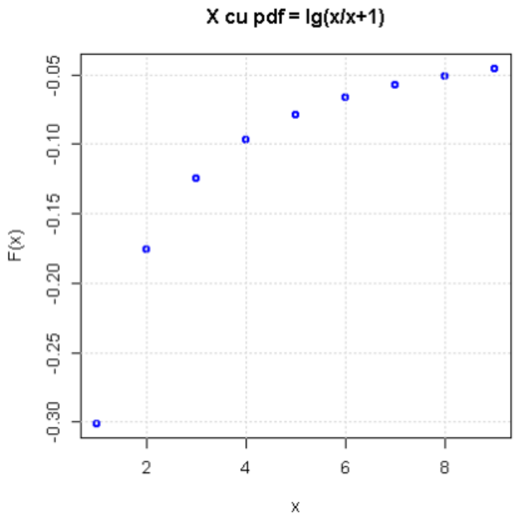
Var[X] = E[X^2] – E[X]^2

Pentru ca nu putem lucra pe domenii infinite, voi alege doar un numar foarte mare pe care voi reprezenta functia.

Pe acest domeniu, aplicam pdf si formulele prezentate anterior si afisam rezultatele fara a mai verifica conditii extra.

d) $f(x) = \lg(\frac{x}{x+1}), \ \ x \in \{1, 2, \ldots 9\}$

Punctul d) se rezolva identic cu c), pentru ca avem o v.a discreta si primim pdf.




```
Media E[X] = -3.440237
Varianța Var(X) = -29.72697
```

Grafic pdf pentru punctul d)

Obs: La aceste subpuncte nu am gasit alternative pentru a oferi utilizatorului optiunea de a selecta variabile.

e) $f(x) = \frac{\theta^2}{1+\theta}(1+x)e^{-\theta x}, \ \ x > 0, \ \ \theta > 0$

Punctul e) se rezolva la fel cu a), b) verificand proprietatile densitatii de probabilitate. Pentru e) vom demonstra ca pentru oricare teta > 0(notat „o” pentru o mai usoara scriere) ne ofera o densitate de probabilitate valida.

f densitate de probabilitate =>

i) f(x) >= 0, pt oricare x, ceea ce este adevarat pt ca primul termen este mereu pozitiv, paranteza deasemenea si exponentiala la fel.

ii) Integrala(0, Inf) f(x)dx = 1

$$\int_0^\infty \frac{\theta^2}{1+\theta}(1+x)e^{-\theta x}dx = 1 \Leftrightarrow \frac{\theta^2}{1+\theta}\int_0^\infty (1+x)e^{-\theta x}dx = 1 \Leftrightarrow$$

$$\Leftrightarrow \frac{\theta^2}{1+\theta}\left(\int_0^\infty e^{-\theta x}dx + \int_0^\infty x\cdot e^{-\theta x}dx\right) = 1 \quad (*)$$

$$\int_0^\infty e^{-\theta x}dx = \frac{1}{-\theta}\int_0^\infty -\theta e^{-\theta x}dx = -\frac{1}{\theta}\cdot e^{-\theta x}\Big|_0^\infty = -\frac{1}{\theta}(0-1) = \frac{1}{\theta}$$

$$\int_0^\infty x\cdot e^{-\theta x}dx = \int_0^\infty x\cdot\left(-\frac{1}{\theta}\cdot e^{-\theta x}\right)' dx = -\frac{1}{\theta}\cdot x\cdot e^{-\theta x}\Big|_0^\infty + \frac{1}{\theta}\int_0^\infty e^{-\theta x}dx =$$

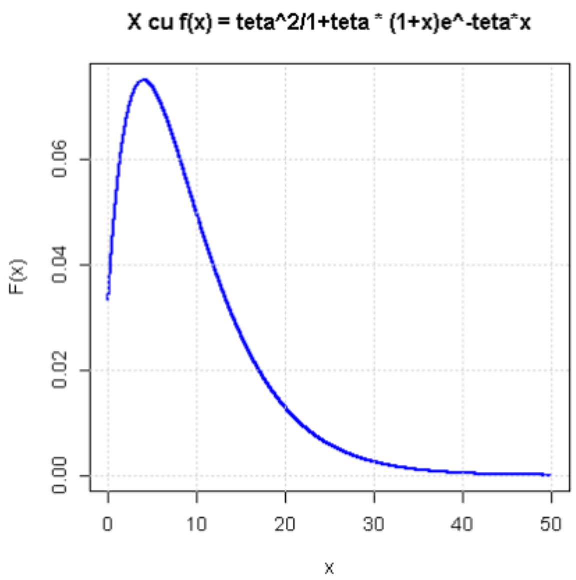
Putem folosi L'Hôpital

$$= 0 - 0 + \frac{1}{\theta}\cdot\frac{1}{\theta} = \frac{1}{\theta^2}$$

Din (*): $\frac{\theta^2}{1+\theta}\left(\int_0^\infty e^{-\theta x}dx + \int_0^\infty x\cdot e^{-\theta x}dx\right) = 1 \Leftrightarrow \frac{\theta^2}{1+\theta}\left(\frac{1}{\theta}+\frac{1}{\theta^2}\right) = 1 \Leftrightarrow$

$\Leftrightarrow \frac{\theta^2}{1+\theta}\cdot\frac{1+\theta}{\theta^2} = 1$ (A) pt $\forall \theta > 0, \forall x > 0$

Din demonstratie concluzionam ca pentru oricare teta ales de utilizator densitatea ramane valida. Am verificat asta prin refolosirea functiei care calculeaza integrala pe domeniu si o compara cu 1. Calcularea mediei si variantei se face la fel in in punctele anterioare prin integrare pe domeniu si folosirea formulelor.




```
Media E[X] = 9.166667
Varianța Var(X) = 49.30556
```

Graficul densitatii e) cu teta = 0.2

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

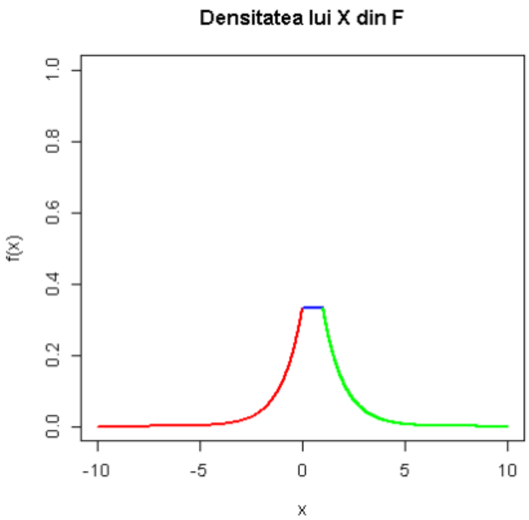
Pentru densitatea f), avem de fapt 3 densitati pe 3 intervale diferite. Pentru a calcula media vom calcula media pe fiecare dintre intervale folosind tehnicile prezentate anterior. Dupa ce le obtinem facem suma ponderata cu probabilitatile pe fiecare interval.

E[X] = P1 * E[X](-Inf,0) + P2* E[X][0,1) + P3*E[X][1,Inf],

unde Pi este probabilitatea ca x sa fie in intervalul i.

Vom calcula la fel E[X^2] si Varianta folosind formula E[X^2] – E[X]^2.

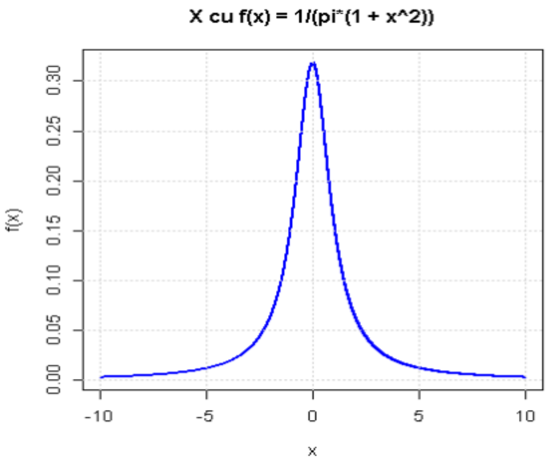
Pentru plotarea graficului pe ramuri am generat inital un grafic care se muleaza pe 0x pe tot domeniul si am adaugat ca si linii cele 3 densitati folosind functia **lines.**




```
Media E[X] = 0.1666667
Varianța Var(X) = 0.787037
```

g) $f(x)=\frac{1}{\pi(1+x^2)}$

Pentru distributia Cauchy media si varianta nu sunt definite in sensul clasic, ele fiind nelimitate. Densitatea este simetrica fata de 0, ceea ce face ca media teoretica sa fie 0, iar integrala pentru calculul variantei este infinita(sau divergenta) motiv pentru care am considerat luarea unui esantion semnificativ din distributie care sa captureze majoritatea valorilor importante. Am decis asupra intervalului [-30,30]. Ca si atunci cand este nevoie sa discretizam un interval infinit, R nu poate sa calculeze o integrala pe o infinitate de valori.




```
Media E[X] = 0
Varianța Var(X) = 62.66834
```

Graficul densitatii Cauchy

Cum functioneaza Shiny?

Singura biblioteca extra folosita in cadrul proiectului este „**shiny**” importata folosind **library(shiny).**

In crearea unei aplicatii shiny vom jongla constant intre cele 2 partii ale ei: componenta UI(user interface), adica ceea ce vede si foloseste utilizatorul pentru a interactiona si partea de Server care se ocupa de calculele din spatele rezultatelor afisate.

Cele mai folosite elemente de UI folosite in cadrul proiectului au fost **titlePanel, sidebarLayout, sidebarPanel, mainPanel, plotOutput, sliderInput, numericInput, verbatimTextOutput.**

Titlul si componentele de layout si panourile genereaza structura aplicatiei, respectiv modul in care este organizata.

SliderInput, numericInput permit utilizatorului sa modifice variabilele aplicatie si sunt capturate in componenta de Server folosind sintaxa „input$numeInput”.

PlotOutput si verbatimTextOutput permit afisarea graficelor si a mesajelor text.

Conventie: Pentru organizarea in componenta UI a componentelor pentru fiecare task am folosit comentarii cu rolul de a indruma utilizator.

Ex:

```
#UI Side Normal(0,1)
####################################################################
titlePanel("Functia de repartitie a unei variabile normale standard"),
sidebarLayout(
  sidebarPanel(),
  mainPanel(
    plotOutput("normalStdPlot"),
    plotOutput("transformedStdPlot"),
    plotOutput("squaredStdPlot")
  )
),
sidebarLayout(
  sidebarPanel(
    sliderInput("stdN", "Alege numarul de variabile", min = 1, max = 10, value = 1, step =1)
  ),
  mainPanel(
    plotOutput("sumStd"),
    plotOutput("sumSquaredStd")
  )
),

####################################################################
```

Blocul de cod a fost delimitat si deasupra vedem utilitatea acestuia, aici fiind prezente componentele UI pentru Ex1, plotarea graficelor pentru normala standard.

PlotOutput au parteneri in componenta Server care le indica ce sa afiseze folosind numele din paranteze.

Componenta Server

Ex:

```
#Server Side Normal(0,1)
####################################################################

output$normalStdPlot <- renderPlot({
  x <- seq(-10, 10, 0.001)
  y <- pnorm(x, mean = 0, sd = 1)
  plot(x, y, type = "l", col = "blue", lwd = 2,
       xlab = "x", ylab = "F(x)",
       main = paste("Funcţia de repartiţie pentru N(0,1)"))
  grid()
})

# ?Pentru Y = 3-2X, P(y)=P(3-2X<=y)=P(X>=(3-y)/2) = 1 - P(X<=(3-y)/2)
output$transformedStdPlot <- renderPlot({
  x <- seq(-17, 23, 0.001)
  transformedX <- (3-x)/2
  y <- 1- pnorm(transformedX, mean = 0, sd = 1)
  plot(-transformedX,y, type = "l", col = "blue", lwd = 2,
       xlab = "x", ylab = "F(x)",
       main = paste("Funcţia de repartiţie pentru 3 - 2X"))
  grid()
})
```

Putem observa ca conventia se pastreaza si pentru partenerii de pe partea de server.

Output$numeGrafic decide ce se va plota in componenta ui. Am grupat graficele in functie de repartia comuna. Toate graficele pentru normala standard sunt delimitate de marcatori folosind comentarii.

Dificultati in realizarea problemelor:

- Dificultati la rezolvarea seriilor de variabile aleatoare(Poisson, Exponentiala)
- Aspecte teoretice legate de distributia Cauchy

Concluzii:

- Aplicatiile Shiny pot fi folosite pentru a oferi o intelegere intuitiva a lucrului cu variabile aleatoare prin posibilitatea de a modifica parametrii ai acestora dupa bunul plac.
- Desi v.a au o multitudine de repartitii, prelucrarea acestora este foarte similara.

Problema III

3) Pentru v.a. X definită prin densitatea/funcția de masă de mai jos estimați parametrul teta prin metoda verosimilității maxime și prin metoda momentelor(pe foaie!), apoi construiți o funcție în R care preia eșantionul dat și întoarce estimațiile realizate în baza celor 2 estimatori. Comparați estimația dată de metoda verosimilității maxime cu valoarea lui teta dată de o metodă numerica.

DESCRIEREA PROBLEMEI

Problema constă în estimarea unui parametru necunoscut, notat cu θ, pentru o variabilă aleatoare X a cărei densitate (pentru variabile continue) sau funcție de masă (pentru variabile discrete) este cunoscută. Scopul este de a estima θ folosind două metode:

1. **Metoda Verosimilității Maxime (MLE - Maximum Likelihood Estimation)**:
   - Această metodă găsește valoarea lui θ care maximizează funcția de verosimilitate, adică probabilitatea de a observa eșantionul dat, condiționată de θ.
2. **Metoda Momentelor (MM - Method of Moments)**:
   - Această metodă estimează θ prin echivalarea momentelor teoretice ale distribuției cu momentele de eșantion (de exemplu, media eșantionului cu media teoretică).

După estimarea lui θ folosind cele două metode, se cere implementarea unei funcții în R care să calculeze și să returneze estimările. În plus, se compară estimația obținută prin MLE cu o estimație numerică a lui θ (obținută prin optimizare).

Un **scop palpabil** pentru această problemă ar fi obținerea unor estimări precise ale parametrului θ pentru distribuția variabilei aleatoare X, folosind metode statistice bine-cunoscute (MLE și MM), și validarea acestor estimări prin compararea lor cu o metodă numerică

ASPECTE TEORETICE

a) Funcția de verosimilitate (Likelihood Function)

- Funcția de verosimilitate $L(\theta)$ măsoară probabilitatea de a observa eșantionul dat, condiționată de parametrul θ.
- Pentru un eșantion $x1,x2,\ldots,xn$, funcția de verosimilitate este:

$$L(\theta) = \prod_{i=1}^{n} f(x_i; \theta),$$

unde $f(x_i; \theta)$ este densitatea (pentru variabile continue) sau funcția de masă (pentru variabile discrete).

b) Metoda Verosimilității Maxime (MLE)

- Scopul este de a găsi valoarea lui θ care maximizează $L(\theta)$.
- Se folosește adesea log-verosimilitatea $\ell(\theta) = lnL(\theta)$ pentru simplificarea calculelor.
- Estimatorul MLE este soluția ecuației:

$$\frac{d\ell(\theta)}{d\theta} = 0.$$

c) Metoda Momentelor (MM)

- Scopul este de a echivala momentele teoretice ale distribuției cu momentele de eșantion.
- De exemplu, pentru o distribuție cu un singur parametru θ:

$E[X]$=media teoretica=media de eșantion.

- Se rezolvă ecuația pentru θ.

d) Optimizare numerică

- Atunci când estimarea analitică este dificilă, se folosește optimizarea numerică (de exemplu, funcția optim în R) pentru a maximiza log-verosimilitatea.

e) Distributia Poisson

Definiție

- Distribuția Poisson este o distribuție discretă care modelează numărul de evenimente rare care au loc într-un interval de timp sau spațiu.
- Parametru: λ (rata de apariție a evenimentelor).

Funcția de masa

$$P(X = k) = \frac{\lambda^k e^{-\lambda}}{k!}, \quad k = 0, 1, 2, \ldots$$

Proprietăți

- Media: $E[X] = \lambda$.
- Varianța: $Var(X) = \lambda$.

Estimarea parametrului λ

- **MLE**: $\hat{\lambda}_{\text{MLE}} = \frac{1}{n} \sum_{i=1}^{n} x_i.$
- **MM**: $\hat{\lambda}_{\text{MM}} = \frac{1}{n} \sum_{i=1}^{n} x_i$

d) Distribuția Gamma

Definiție

- Distribuția Gamma este o distribuție continuă folosită pentru a modela timpi de așteptare sau sume de variabile aleatoare exponențiale.
- Parametri:
  - α (parametrul de formă, shape).
  - β (parametrul de scară, scale) sau $\theta = \frac{1}{\beta}$ (parametrul de rată, rate).

Functia de densitate

$$f(x;\alpha,\beta) = \frac{\beta^\alpha x^{\alpha-1} e^{-\beta x}}{\Gamma(\alpha)}, \quad x > 0,$$

Proprietăți

- Media: $E[X] = \frac{\alpha}{\beta}$
- Varianța: $Var(X) = \frac{\alpha}{\beta^2}$.

Estimarea parametrului λ

- **MLE**: $\hat{\theta}_{\mathrm{MLE}} = \frac{\alpha}{\bar{x}}$.
- **MM**: $\hat{\theta}_{\mathrm{MM}} = \frac{\alpha}{\bar{x}}$.

e) Distribuția Geometrică

Definiție

- Distribuția Geometrică este o distribuție discretă care modelează numărul de încercări până la primul succes într-o succesiune de încercări Bernoulli.
- Parametru: p (probabilitatea de succes).

Functia de masa

$$P(X = k) = (1-p)^{k-1} p, \quad k = 1, 2, 3, \ldots$$

Proprietăți

- Media: $E[X] = \frac{1}{p}$
- Varianța: $Var(X) = \frac{1-p}{p^2}$

Estimarea parametrului *p*

- **MLE**: $\hat{p}_{\text{MLE}} = \frac{1}{\bar{x}}$.
- **MM**: $\hat{p}_{\text{MM}} = \frac{1}{\bar{x}}$.

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

→Metoda Verosimilitatii Maxime (MLE)

Pentru f_θ dat avem functia de verosmiltate:

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

Pe aceasta, o folosim pentru a afla log-verosimilitatea:

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

Iar, derivand:

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

Si egaland cu 0 obtinem estimatorul MLE:

$$\hat{\theta}_{\ MLE} = \frac{\sum_{i=1}^{n} x_i}{2n}$$

→Metoda Momentelor

Pentru f_θ dat media distributiei Poisson este:

$$E[X] = 2\theta$$

Si deci avem estimatorul MM:

$$\hat{\theta}_{\ MM} = \frac{\overline{X}}{2}$$

Unde $\overline{X}$ reprezinta media esantionului

Avem urmatoare implementare in R:

```r
# Esantionul
sample_a <- 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)

# Estimator MLE
theta_mle_a <- sum(sample_a) / (2 * length(sample_a))

# Estimator MM
theta_mm_a <- mean(sample_a) / 2

# Definirea functiei de log-verosimilitate
log_likelihood_a <- function(theta) {
  -2 * theta * length(sample_a) + sum(sample_a) * log(2 * theta) - sum(lfactorial(sample_a))
}

# Maximizarea log-verosimilitatii
result_a <- optim(par = 1, fn = log_likelihood_a, method = "Brent", lower = 0, upper = 100,
control = list(fnscale = -1))

# Estimator numeric
theta_numeric_a <- result_a$par

# Afisare rezultate
cat("Estimator numeric pentru theta:", theta_numeric_a, "\n")
cat("Estimator MLE pentru theta:", theta_mle_a, "\n")
cat("Estimator MM pentru theta:", theta_mm_a, "\n")

```

In urma careia, obtinem:

```
 > cat("Estimator numeric pentru theta:", theta_numeric_a, "\n")
 Estimator numeric pentru theta: 4.97
> cat("Estimator MLE pentru theta:", theta_mle_a, "\n")
 Estimator MLE pentru theta: 4.97
 > cat("Estimator MM pentru theta:", theta_mm_a, "\n")
 Estimator MM pentru theta: 4.97
```

Deci, diferentele nu sunt de asa natura astfel incat sa altereze rezultatul semnificativ (dar in urma unei analize mai complexe asupra datelor stocate pot fi observate de la nivelul de acuratete $4.97 \cdot 10^{-11}$

theta_numeric_a	4.97000000079766
theta_mle_a	4.97
theta_mm_a	4.97

b) $f_\theta(x) = C_n^x \cdot \theta^x(1-\theta)^{(1-x)}, x \in \{1,2,3, \dots n\}, \theta \in (0,1)$, n fixat

→Metoda Verosimilitatii Maxime (MLE)

Pentru f_θ dat avem functia de verosmiltate:

$$L(\theta) = \prod_{i=1}^{n} \ln \binom{14}{x_i} \theta^{x_i}(1-\theta)^{14-x_i}$$

Pe aceasta, o folosim pentru a afla log-verosimilitatea:

$$ln\, L(\theta) = \sum_{i=1}^{n} \left(\ln \binom{14}{x_i} + x_i ln\theta + (14 - x_i) ln(1-\theta) \right)$$

Iar, derivand:

$$\frac{d}{d\theta} ln\, L(\theta) = \frac{\sum_{i=1}^{n} x_i}{\theta} - \frac{14n - \sum_{i=1}^{n} x_i}{1-\theta}$$

Si egaland cu 0 obtinem estimatorul MLE:

$$\hat{\theta}_{MLE} = \frac{\sum_{i=1}^{n} x_i}{14n}$$

→Metoda Momentelor

Pentru f_θ dat media distributiei Binomiale este:

$$E[X] = 14\theta$$

Si deci avem estimatorul MM:

$$\hat{\theta}_{MM} = \frac{\overline{X}}{14}$$

Unde $\overline{X}$ reprezinta media esantionului

Avem urmatoare implementare in R:

```r
# Esantionul
sample_b <- 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)

# Estimator MLE
theta_mle_b <- sum(sample_b) / (14 * length(sample_b))

# Estimator MM
theta_mm_b <- mean(sample_b) / 14

# Definirea functiei de log-verosimilitate
log_likelihood_b <- function(theta) {
  sum(dbinom(sample_b, size = 14, prob = theta, log = TRUE))
}

# Maximizarea log-verosimilitatii
result_b <- optim(par = 0.5, fn = log_likelihood_b, method = "Brent", lower = 0, upper = 1, control = list(fnscale = -1))

# Estimator numeric
theta_numeric_b <- result_b$par

# Afisare rezultate
cat("Estimator numeric pentru theta:", theta_numeric_b, "\n")
cat("Estimator MLE pentru theta:", theta_mle_b, "\n")
cat("Estimator MM pentru theta:", theta_mm_b, "\n")
```

In urma careia, obtinem:

```
> cat("Estimator numeric pentru theta:", theta_numeric_b, "\n")
Estimator numeric pentru theta: 0.2014286
 > cat("Estimator MLE pentru theta:", theta_mle_b, "\n")
 Estimator MLE pentru theta: 0.2014286
 > cat("Estimator MM pentru theta:", theta_mm_b, "\n")
 Estimator MM pentru theta: 0.2014286
```

Deci, diferentele nu sunt de asa natura astfel incat sa altereze rezultatul semnificativ (dar in urma unei analize mai complexe asupra datelor stocate pot fi observate de la nivelul de acuratete $0.2014286 \cdot 10^{-13}$

theta_numeric_b	0.201428571414209
theta_mm_b	0.201428571428571
theta_mle_b	0.201428571428571

c) $f_\theta(x) = e^{\frac{-x}{\theta}} \cdot \frac{x^{\alpha-1}}{\Gamma(\alpha)\cdot\theta^\alpha}$ $, x \in (0,\infty), \theta \in (0,\infty), \alpha \in (0,\infty)\ fixat$

→Metoda Verosimilitatii Maxime (MLE)

Pentru f_θ dat avem functia de verosmiltate:

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

Pe aceasta, o folosim pentru a afla log-verosimilitatea:

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

Iar, derivand:

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

Si egaland cu 0 obtinem estimatorul MLE:

$$\hat{\theta}_{MLE} = \frac{\sum_{i=1}^{n} x_i}{n\alpha}$$

→Metoda Momentelor

Pentru f_θ dat media distributiei Gamma este:

$$E[X] = \alpha\theta$$

Si deci avem estimatorul MM:

$$\hat{\theta}_{MM} = \frac{\overline{X}}{\alpha}$$

Unde $\overline{X}$ reprezinta media esantionului

Avem urmatoare implementare in R:

```r
# Esantionul
sample_c <- 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)
# Parametrul alpha
alpha_c <- 7

# Estimator MLE
theta_mle_c <- sum(sample_c) / (alpha_c * length(sample_c))

# Estimator MM
theta_mm_c <- mean(sample_c) / alpha_c

# Definirea functiei de log-verosimilitate
log_likelihood_c <- function(theta) {
  sum(dgamma(sample_c, shape = alpha_c, scale = theta, log = TRUE))
}

# Maximizarea log-verosimilității
result_c <- optim(par = 1, fn = log_likelihood_c, method = "Brent", lower = 0, upper = 100,
control = list(fnscale = -1))

# Estimator numeric
theta_numeric_c <- result_c$par

# Afisare rezultate
cat("Estimator MLE pentru theta:", theta_mle_c, "\n")
cat("Estimator MM pentru theta:", theta_mm_c, "\n")
cat("Estimator numeric pentru theta:", theta_numeric_c, "\n")
```

In urma careia, obtinem:

```
> cat("Estimator MLE pentru theta:", theta_mle_c, "\n")
Estimator MLE pentru theta: 1.993285
> cat("Estimator MM pentru theta:", theta_mm_c, "\n")
Estimator MM pentru theta: 1.993285
> cat("Estimator numeric pentru theta:", theta_numeric_c, "\n")
Estimator numeric pentru theta: 1.993285
```

Deci, diferentele nu sunt de asa natura astfel incat sa altereze rezultatul semnificativ (dar in urma unei analize mai complexe asupra datelor stocate pot fi observate de la nivelul de acuratete $n \cdot 10^{-10}$

theta_numeric_c	1.99328521693552
theta_mm_c	1.99328521571429
theta_mle_c	1.99328521571429

d) $f_{\theta}(x) = \frac{\theta^x}{(1+\theta)^{(1+x)}}, x \in \mathbb{N}, \theta \in (0, \infty)$

→Metoda Verosimilitatii Maxime (MLE)

Pentru f_θ dat avem functia de verosmiltate:

$$L(\theta) = \prod_{i=1}^{n} \frac{\theta^{x_i}}{(1+\theta)^{1+x_i}}$$

Pe aceasta, o folosim pentru a afla log-verosimilitatea:

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

Iar, derivand:

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

Si egaland cu 0 obtinem estimatorul MLE:

$$\hat{\theta}_{MLE} = \frac{\sum_{i=1}^{n} x_i}{n}$$

→Metoda Momentelor

Pentru f_θ dat media distributiei Geometrice este:

$$E[X] = \theta$$

Si deci avem estimatorul MM:

$$\hat{\theta}_{MM} = \overline{X}$$

Unde $\overline{X}$ reprezinta media esantionului

Avem urmatoare implementare in R:

```r
# Esantionul
sample_d <- 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)

# Estimator MLE
theta_mle_d <- mean(sample_d)

# Estimator MM
theta_mm_d <- mean(sample_d)

# Definirea functiei de log-verosimilitate
log_likelihood_d <- function(theta) {
  sum(dgeom(sample_d, prob = 1 / (1 + theta), log = TRUE))
}

# Maximizarea log-verosimilităţii
result_d <- optim(par = 1, fn = log_likelihood_d, method = "Brent", lower = 0, upper = 1000,
control = list(fnscale = -1))

# Estimator numeric
theta_numeric_d <- result_d$par

# Afisare rezultate
cat("Estimator MLE pentru theta:", theta_mle_d, "\n")
cat("Estimator MM pentru theta:", theta_mm_d, "\n")
cat("Estimator numeric pentru theta:", theta_numeric_d, "\n")In urma careia, obtinem:
1. > cat("Estimator MLE pentru theta:", theta_mle_d, "\n")
```

```
Estimator MLE pentru theta: 25.85
> cat("Estimator MM pentru theta:", theta_mm_d, "\n")
Estimator MM pentru theta: 25.85
> cat("Estimator numeric pentru theta:", theta_numeric_d, "\n")
Estimator numeric pentru theta: 25.85
```

Deci, diferentele nu sunt de asa natura astfel incat sa altereze rezultatul semnificativ (dar in urma unei analize mai complexe asupra datelor stocate pot fi observate de la nivelul de acuratete $n \cdot 10^{-8}$

theta_numeric_d	25.8500003901255
theta_mm_d	25.85
theta_mle_d	25.85

e) $f_\theta(x) = \frac{\alpha}{\theta} x^{\alpha-1} \cdot e^{\frac{-x}{\theta}}, x \in (0, \infty)\ , \theta \in (0, \infty), \alpha\ \in (0, \infty)\ fixat$

→Metoda Verosimilitatii Maxime (MLE)

Pentru f_θ dat avem functia de verosmiltate:

$$L(\theta)\ =\ \prod_{i=1}^{n} \frac{\alpha}{\theta} x_i^{\alpha-1} e^{\frac{-x}{\theta}}$$

Pe aceasta, o folosim pentru a afla log-verosimilitatea:

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

Iar, derivand:

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

Si egaland cu 0 obtinem estimatorul MLE:

$$\hat{\theta}\ _{MLE}\ =\ \frac{\sum_{i=1}^{n} x_i}{n\alpha}$$

→Metoda Momentelor

Pentru f_θ dat media distributiei Gamma este:

$$E[X] = \alpha\theta$$

Si deci avem estimatorul MM:

$$\hat{\theta}\ _{MM} = \frac{\overline{X}}{\alpha}$$

Unde $\overline{X}$ reprezinta media esantionului

Avem urmatoare implementare in R:

```
# Esantionul
sample_e <- 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)

# Parametrul alpha
alpha_e <- 3

# Estimator MLE
theta_mle_e <- sum(sample_e) / (alpha_e * length(sample_e))

# Estimator MM
theta_mm_e <- mean(sample_e) / alpha_e

# Definirea functiei de log-verosimilitate
log_likelihood_e <- function(theta) {
  sum(dgamma(sample_e, shape = alpha_e, scale = theta, log = TRUE))
}

# Maximizarea log-verosimilității
result_e <- optim(par = 1, fn = log_likelihood_e, method = "Brent", lower = 0, upper = 100, control = list(fnscale = -1))

# Estimator numeric
theta_numeric_e <- result_e$par

# Afișare rezultate
cat("Estimator MLE pentru theta:", theta_mle_e, "\n")
cat("Estimator MM pentru theta:", theta_mm_e, "\n")
cat("Estimator numeric pentru theta:", theta_numeric_e, "\n")
```

In urma careia, obtinem:

```
> cat("Estimator MLE pentru theta:", theta_mle_e, "\n")
Estimator MLE pentru theta: 1.124153
> cat("Estimator MM pentru theta:", theta_mm_e, "\n")
Estimator MM pentru theta: 1.124153
> cat("Estimator numeric pentru theta:", theta_numeric_e, "\n")
Estimator numeric pentru theta: 1.124153
```

Deci, diferentele nu sunt de asa natura astfel incat sa altereze rezultatul semnificativ (dar in urma unei analize mai complexe asupra datelor stocate pot fi observate de la nivelul de acuratete $n \cdot 10^{-10}$

theta_numeric_e	1.12415290480678
theta_mm_e	1.12415290633333
theta_mle_e	1.12415290633333

Concluzii

Putem observa in urma tuturor calculelor, ca folosind oricare din metode, obtinem rezultate extrem de similare, deci pentru orice aplicatie reala, o putem aplica pe oricare.

Sursele care au ajutat la rezolvarea problemelor:

P1.

- 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
- https://www.geeksforgeeks.org/exponential-distribution-in-r-programming-dexp-pexp-qexp-and-rexp-functions/

P2.

- Codul primit in cadrul laboratorului
- https://alexamarioarei.quarto.pub/curs-ps-fmi/Introducere_R/
- Materialele de curs ale domnului profesor Niculescu Cristian
- https://mathworld.wolfram.com/CauchyDistribution.html
- https://shiny.posit.co (documentatia shiny)
- Documentatia prezenta in R Studio
- Materialele de seminar pentru calculul cu v.a continue si discrete.

P3.

- https://stat.ethz.ch/R-manual/R-devel/library/stats/html/GammaDist.html
- https://stat.ethz.ch/R-manual/R-devel/library/stats/html/optim.html
- https://www.probabilitycourse.com/chapter8/8_2_3_max_likelihood_estimation.php
- Materialele de curs ale domnului profesor Niculescu Cristian
- Materialele de seminar pentru calculul cu v.a continue si discrete.

Anexa cod

Cod Problema 1

```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))



################### 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 #########################

simulate_T_final <- function(n, lambda, alpha) {
  total_time <- 0
  completed <- TRUE  # presupun ca finalizeaza activitatea
```

```r
  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))



################### 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))



################### 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)


################### 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()
```

Cod Problema 2

```
library(shiny)

ui <- fluidPage(

  #UI Side Normal(0,1)
  #################################################################
  titlePanel("Functia de repartitie a unei variabile normale standard"),
  sidebarLayout(
    sidebarPanel(),
    mainPanel(
      plotOutput("normalStdPlot"),
      plotOutput("transformedStdPlot"),
      plotOutput("squaredStdPlot")
    )
  ),
  sidebarLayout(
    sidebarPanel(
      sliderInput("stdN", "Alege numarul de variabile", min = 1, max = 10, value = 1, step =1)
    ),
    mainPanel(
      plotOutput("sumStd"),
      plotOutput("sumSquaredStd")
    )
  ),

  #################################################################

  #UI Side Normal(mu, sigma^2)
  #################################################################
  titlePanel("Funcția de repartiție a unei variabile normale"),
  sidebarLayout(
    sidebarPanel(
      sliderInput("mu", "Alege media (\u03BC):", min = -10, max = 10, value = 0, step = 0.1),
      sliderInput("sigma", "Alege deviația standard (\u03C3):", min = 0.1, max = 10, value = 1, step = 0.1)
    ),
    mainPanel(
      plotOutput("cdfPlot"),
      plotOutput("transformedPlot"),
      plotOutput("squaredPlot")
    )
  ),
  sidebarLayout(
    sidebarPanel(
      sliderInput("n", "Alege n:", min = 1, max = 10, value = 1, step = 1),
      sliderInput("muSum", "Alege media (\u03BC):", min = -10, max = 10, value = 0, step = 0.1),
      sliderInput("sigmaSum", "Alege deviația standard (\u03C3):", min = 0.1, max = 10, value = 1, step = 0.1)
    ),
    mainPanel(
      plotOutput("sumCdf"),
      plotOutput("sumSquaredCdf")
    )
  ),
  #################################################################

```

```
#UI Side Exponential(lambda)
#################################################################
titlePanel("Functia de repartitie a unei variable exponentiale"),
sidebarLayout(
  sidebarPanel(
    sliderInput("lambda", "Alege lambda (\u03BB)", min = 0.001, max = 10, value = 1, step = 0.1),
  ),
  mainPanel(
    plotOutput("cdfExpDefault"),
    plotOutput("transformedExp"),
    plotOutput("squaredExp")
  )
),
sidebarLayout(
  sidebarPanel(
    sliderInput("nExp", "Alege n", min = 1, max = 10, value = 1, step = 1),
    sliderInput("sumLambda", "Alege lambda (\u03BB)", min = 0.001, max = 10, value = 1, step = 0.1)
  ),
  mainPanel(
    plotOutput("sumExp")
  )
),
###############################################################

#UI Side Poisson(lambda)
###############################################################
titlePanel("Functia de repartitie a unei variabile poisson"),
sidebarLayout(
  sidebarPanel(
    sliderInput("lambdaPoisson", "Alege lambda (\u03BB)", min = 0.001, max = 30, value = 1, step = 0.1),
  ),
  mainPanel(
    plotOutput("cdfPoisson"),
    plotOutput("transformedPoisson"),
    plotOutput("squaredPoisson")
  )
),
sidebarLayout(
  sidebarPanel(
    sliderInput("poissonN", "Alege n", min = 1, max = 10, value = 1, step = 1),
    sliderInput("sumPoisLambda", "Alege lambda (\u03BB)", min = 0.001, max = 30, value = 1, step = 0.1),
  ),
  mainPanel(
    plotOutput("sumPoisson")
  )
),
###############################################################

#UI Side Binomial(r,p)
###############################################################
titlePanel("Functia de repartitie a unei variabile binomiale"),
mainPanel(
  sliderInput("r", "Alege numarul de experimente", min = 0, max = 100, value= 10, step = 1),
  sliderInput("p", "Alege probabilitatea de succes", min = 0.01, max = 1, value = 0.3, step = 0.01),
  plotOutput("cdfBinomial"),
  plotOutput("transformedBinomial"),
  plotOutput("squaredBinomial")
),
mainPanel(
  sliderInput("sumBinN", "Alege numarul de variabile", min = 1, max = 10, value = 1, step = 1),
  sliderInput("sumR", "Alege numarul de experimente", min = 0, max = 100, value= 10, step = 1),
```

```r
  sliderInput("sumP", "Alege probabilitatea de succes", min = 0.01, max = 1, value = 0.3, step = 0.01),
  plotOutput("sumBinomial")
),
################################################################

#Ex. 2
#Să construiți câte o funcție în R care afișează funcția pentru parametri particularizabili de
#către utilizator și calculează și media și varianța pentru v.a. X

################################################################

#a) f(x) =cx^4, x din (0,2), c din R
mainPanel(
  titlePanel("Densitatea, media si varianta pentru f(x) = cx^4"),
  sidebarLayout(
    sidebarPanel(
      numericInput("numerator", "Introdu un numarator pentru c:", value = 5, min = 1, max = 100, step = 1),
      numericInput("denominator", "Introdu un numitor pentru c:", value = 32, min = 1, max = 100, step = 1)
    ),
    mainPanel(
      plotOutput("densityA"),
      verbatimTextOutput("resultsA")
    )
  ),
),

################################################################

#b) f(x) = ax + bx^2, 0 < x < 1, a, b din R
mainPanel(
  titlePanel("Densitatea, media si varianta pentru f(x) = ax + bx^2, a = (6 - 2b) / 3"),
  sidebarLayout(
    sidebarPanel(
      numericInput("ex2A", "Introdu un A valid raportat la b: ", value = 0, min = -20, max = 20, step = 0.1),
      numericInput("ex2B", "Introdu un B valid raportat la a: ", value = 3, min = -20, max = 20, step = 0.1)

    ),
    mainPanel(
      plotOutput("densityB"),
      verbatimTextOutput("resultsB")
    )
  )
),

################################################################

#c) f(X) =  4/x(x+1)(x+2), x din N\{0}
mainPanel(
  titlePanel("Pdf, media si varianta pentru X cu pdf = 4/x(x+1)(x+2)"),
  sidebarLayout(
    sidebarPanel(
    ),
    mainPanel(
      plotOutput("densityC"),
      verbatimTextOutput("resultsC"),
    )
  )
),

################################################################
#d) f(x) = lg(x/x+1), x din {1,9}
mainPanel(
  titlePanel("Pdf, media si varianta pentru x cu pdf = lg(x/x+1)"),
  sidebarLayout(
```

```
      sidebarPanel(),
      mainPanel(
        plotOutput("densityD"),
        verbatimTextOutput("resultsD")
      )
    )
  ),

  ##############################################################
  #e) f(x) = teta^2/1+teta * (1+x)e^-teta*x, x>0, teta>0
  mainPanel(
    titlePanel("Densitatea, media si varianta pentru X din E"),
    sidebarLayout(
      sidebarPanel(
        sliderInput("teta", "Introdu variabila teta > 0: ", value = 1, min = 0.01, max = 5, step = 0.01)
      ),
      mainPanel(
        plotOutput("densityE"),
        verbatimTextOutput("resultsE"),
      )
    )
  ),

  ##############################################################
  #f) f(x) = 1/*3e^x, x<0; f(x) = 1/3, 0<=x<1; f(x) = 1/3*x^1-x, x>= 1
  mainPanel(
    titlePanel("Densitatea, media si varianta pentru X din F"),
    sidebarLayout(
      sidebarPanel(),
      mainPanel(
        plotOutput("densityF"),
        verbatimTextOutput("resultsF")
      )
    )
  ),

  ##############################################################
  #g) f(x) = 1/(pi*(1 + x^2))

  mainPanel(
    titlePanel("Densitatea, media si varianta pentru X din G"),
    sidebarLayout(
      sidebarPanel(),
      mainPanel(
        plotOutput("densityG"),
        verbatimTextOutput("resultsG")
      )
    )
  )

)


server <- function(input, output) {

  #Server Side Normal(0,1)
  ##############################################################

  output$normalStdPlot <- renderPlot({
    x <- seq(-10, 10, 0.001)
    y <- pnorm(x, mean = 0, sd = 1)
    plot(x, y, type = "l", col = "blue", lwd = 2,
         xlab = "x", ylab = "F(x)",
         main = paste("Funcția de repartiție pentru N(0,1)"))
    grid()
  })

  # ?Pentru Y = 3-2X, P(y)=P(3-2X<=y)=P(X>=(3-y)/2) = 1 - P(X<=(3-y)/2)
  output$transformedStdPlot <- renderPlot({
```

```r
    x <- seq(-17, 23, 0.001)
    transformedX <- (3-x)/2
    y <- 1- pnorm(transformedX, mean = 0, sd = 1)
    plot(-transformedX,y, type = "l", col = "blue", lwd = 2,
         xlab = "x", ylab = "F(x)",
         main = paste("Funcția de repartiție pentru 3 - 2X"))
    grid()
  })

  # Pentru Y = X^2, P(X^2<=y)=P(-sqrt(y)<=X<=sqrt(y))=Fx(sqrt(y)) - Fx(-sqrt(y))
  output$squaredStdPlot <- renderPlot({
    x <- seq(0, 100, 0.01)
    sqrtX <- sqrt(x)
    y <- pnorm(sqrtX, mean = 0, sd = 1) - pnorm(-sqrtX, mean = 0, sd = 1)
    plot(sqrtX, y, type = "l", col = "blue", lwd = 2,
         xlab = "x", ylab = "F(x)",
         main = paste("Funcția de repartiție pentru X^2"))
    grid()
  })

  #Pentru Y = Sn
  output$sumStd <- renderPlot({
    n <- input$stdN
    x <- seq(-10, 10, 0.001)
    y <- pnorm(x, mean = 0, sd = sqrt(n))
    plot(x, y, type = "l", col = "blue", lwd = 2,
         xlab = "x", ylab = "F(x)",
         main = paste("Funcția de repartiție pentru Sum(N(0,1))"))
    grid()
  })

  #Pentru Y = Qn
  output$sumSquaredStd <- renderPlot({
    n <- input$stdN
    x <- seq(0, 10, 0.01)
    y <- pnorm(x, mean = n, sd = sqrt(2 * n))
    plot(x, y, type = "l", col = "blue", lwd = 2,
         xlab = "x", ylab = "F(x)",
         main = paste("Funcția de repartiție pentru Sum(N(0,1)^2)"))
    grid()
  })

  ##############################################################

  #Server Side Normal(mu, sigma^2)
  ##############################################################

  #Pentru X = N(mu, sigma^2)
  output$cdfPlot <- renderPlot({
    x <- seq(-10, 10, 0.001)
    y <- pnorm(x, mean = input$mu, sd = input$sigma)

    plot(x, y, type = "l", col = "blue", lwd = 2,
         xlab = "x", ylab = "F(x)",
         main = paste("Funcția de repartiție pentru N(", input$mu, ",", input$sigma^2, ")"))
    grid()
  })

  #Pentru Y = 3 - 2X
  output$transformedPlot <- renderPlot({
    x <- seq(-17, 23, 0.001)
    transformed_x <- (3-x)/2
    y <- 1 - pnorm(transformed_x, mean = input$mu, sd = input$sigma)
    plot(-transformed_x, y, type = "l", col = "blue", lwd = 2,
         xlab = "x", ylab = "F(x)",
         main = paste("Funcția de repartiție pentru 3-2X"))
    grid()
  })

  #Pentru Y = X^2
```

```r
output$squaredPlot <- renderPlot({
  x <- seq(0, 100, 0.01)
  sqrtX <- sqrt(x)
  y <- pnorm(sqrtX, mean = input$mu, sd = input$sigma)
  plot(sqrtX, y, type = "l", col = "blue", lwd = 2,
       xlab = "x", ylab = "F(x)",
       main = paste("Funcția de repartiție pentru X^2"))
  grid()
})

#Pentru Sn = N(n*mu, n * sigma^2)
output$sumCdf <- renderPlot({
  n <- input$n
  x <- seq(-10, 10, 0.001)
  y <- pnorm(x, mean = n * input$muSum, sd = sqrt(n) * input$sigmaSum)
  plot(x, y, type = "l", col = "blue", lwd = 2,
       xlab = "x", ylab = "F(x)",
       main = paste("Funcția de repartiție pentru Sum(Xi)"))
  grid()
})

#Pentru Qn = N(n*(s^2 + mu^2), n *(2*s^4 + 4*mu^2*s^2))
output$sumSquaredCdf <- renderPlot({
  n <- input$n
  selMean <- input$muSum
  selSd <- input$sigmaSum
  x <- seq(0, 10, 0.001)

  mean <- n*(selMean^2 + selSd^2)
  sd <- sqrt(n*(2*selSd^4 + 4*selMean^2*selSd^2))

  y <- pnorm(x, mean = mean, sd = sd)
  plot(x, y, type = "l", col = "blue", lwd = 2,
       xlab = "x", ylab = "F(x)",
       main = paste("Funcția de repartiție pentru Sum(Xi^2)"))
  grid()
})

##############################################################

#Server Side Exponential(lambda)
##############################################################

# Pentru X = Exp(lambda)
output$cdfExpDefault <- renderPlot({
  x <- seq(0, 6, 0.001)
  y <- pexp(x, rate = input$lambda)
  plot(x, y, type = "l", col = "blue", lwd =2,
  xlab = "x", ylab = "F(x)",
  main = paste("Funcția de repartiție pentru Exp(\u03BB)"))
grid()
})

# Pentru Y = 2 + 5x, F(y)=P(Y<=y)=P(2+5X<=y)=P(X<=(y-2)/5)=Fx((y-2)/5)
output$transformedExp <- renderPlot({
  x <- seq(0.001, 6, 0.001)
  transformedX <- (x-2)/5
  y <- pexp(transformedX, rate = input$lambda)
  plot(transformedX, y, type = "l", col = "blue", lwd =2,
       xlab = "x", ylab = "F(x)",
       main = paste("Funcția de repartiție pentru 2 + 5X"))
  grid()
})

#Pentru Y = X^2
output$squaredExp <- renderPlot({
  x <- seq(0, 36, 0.01)
  sqrtX <- sqrt(x)
  y <- pexp(sqrtX, rate = input$lambda)  # y e mereu pozitiv, din definitia exp
  plot(sqrtX, y, type = "l", col = "blue", lwd =2,
```

```
        xlab = "x", ylab = "F(x)",
        main = paste("Funcția de repartiție pentru X^2"))
    grid()
  })

  #Pentru Sn = Gamma(n, lambda)
  output$sumExp <- renderPlot({
    n <- input$nExp
    x <- seq(0.001, 10, 0.001)
    y <- pgamma(x, shape = n, rate = input$sumLambda)
    plot(x, y, type = "l", col = "blue", lwd =2,
        xlab = "x", ylab = "F(x)",
        main = paste("Funcția de repartiție pentru Sum(Exp(\u03BB))"))
    grid()
  })

  ##############################################################

  #Server Side Poisson(lambda)
  ##############################################################

  # Pentru X = Poisson(lambda)
  output$cdfPoisson <- renderPlot({
    x <- seq(0, 20, 0.01)# am folosit aceasta discretizare pt a obtine un grafic mai estetic
    y <- ppois(x, input$lambdaPoisson)
    plot(x, y, type = "l", col = "blue", lwd =2,
        xlab = "x", ylab = "F(x)",
        main = paste("Funcția de repartiție pentru Poisson(\u03BB)"))
    grid()
  })

  #Pentru Y = 3X -2, F(y)=P(Y<=y)=P(3X-2<=y)=P(X<=(y+2)/3)
  output$transformedPoisson <- renderPlot({
    x <- seq(0, 20, 0.01)
    transformedX <- (x + 2)/3
    y <- ppois(transformedX, input$lambdaPoisson)
    plot(transformedX, y, type = "l", col = "blue", lwd =2,
        xlab = "x", ylab = "F(x)",
        main = paste("Funcția de repartiție pentru 3X-2"))
    grid()
  })

  #Pentru Y = X^2
  output$squaredPoisson <- renderPlot({
    x <- seq(0, 20, 0.01)
    sqrtX <- sqrt(x)
    y <- ppois(sqrtX, input$lambdaPoisson)
    plot(sqrtX, y, type = "l", col = "blue", lwd =2,
        xlab = "x", ylab = "F(x)",
        main = paste("Funcția de repartiție pentru X^2"))
    grid()
  })

  #Pentru Y = Poisson(n*lambda)
  output$sumPoisson <- renderPlot({
    n <- input$poissonN
    x <- seq(0.001, 20 * n, 0.001)
    y <- ppois(x, n * input$sumPoisLambda)
    plot(x, y, type = "l", col = "blue", lwd =2,
        xlab = "x", ylab = "F(x)",
        main = paste("Funcția de repartiție pentru Sum(Poisson(\u03BB))"))
    grid()
  })

  ##############################################################

  #Server Side Binomial(r,p)
  ##############################################################

  #Pentru X ~ Binomial(r,p)
```

```r
output$cdfBinomial <- renderPlot({
  r <- input$r
  p <- input$p
  x <- seq(0,r,0.01)
  y <- pbinom(x, size = r, prob = p)
  plot(x, y, col = "blue", lwd =2,type="l",
       xlab = "x", ylab = "F(x)",
       main = paste("Funcția de repartiție pentru Binomial(r,p)"))
  grid()
})

#Pentru Y = 5X-4=>P(5x-4<=y)=P(X<=(y+4)/5)
output$transformedBinomial <- renderPlot({
  r <- input$r
  p <- input$p
  x <- seq(-4, 46, 0.01)
  transformedX <- (x + 4)/5
  y <- pbinom(transformedX, size = r, prob = p)
  plot(x, y, col = "blue", lwd =2, type="l",
       xlab = "x", ylab = "F(x)",
       main = paste("Funcția de repartiție pentru 5X-4"))
  grid()
})

#Pentru Y=X^2
output$squaredBinomial <- renderPlot({
  r <- input$r
  p <- input$p
  x <- seq(0,r ^ 2, 0.01)
  sqrtX <- sqrt(x)
  y <- pbinom(sqrtX, size = r, prob = p)
  plot(sqrtX, y, col = "blue", lwd =2, type="l",
       xlab = "x", ylab = "F(x)",
       main = paste("Funcția de repartiție pentru X^2"))
  grid()
})

#Pentru Sn = Binomial(n*r, p)
output$sumBinomial <- renderPlot({
  n <- input$sumBinN
  p <- input$sumP
  r <- input$sumR
  x <- seq(0,(r * n),0.01)
  y <- pbinom(x, size = r * n, prob = p)
  plot(x, y, col = "blue", lwd =2, type="l",
       xlab = "x", ylab = "F(x)",
       main = paste("Funcția de repartiție pentru Sum(Binomial(r,p))"))
  grid()
})
################################################################

#Ex 2.

#a) f(x) = cx^4, x din (0,2), c din R
##############################################################


f_x_A <- reactive(function(x) (input$numerator / input$denominator) * x^4)

calculate_results_A <- reactive({
  num <- input$numerator
  den <- input$denominator
  c <- num / den

  #Integrala de validare
  integral_f <- integrate(f_x_A(), 0, 2)$value

  #Medie E(x)
  average <- integrate(function(x) x * f_x_A()(x), 0, 2)$value

```

```
    #Medie x^2 E(x^2)
    averageSqr <- integrate(function(x) x^2 * f_x_A()(x),  0, 2)$value

    #Varianta
    variance <- averageSqr - average^2

    list(integral_f = integral_f, average = average, variance = variance)

  })

  output$densityA <-renderPlot({
    interval <- seq(0.001, 1.999, 0.001)
    f <- function(x) input$numerator/input$denominator * x^4
    f_x <- sapply(interval, f)
    plot(interval, f_x, col = "blue", lwd =2,
         xlab = "x", ylab = "F(x)",
         main = paste("Densitatea pentru f(x) = cx^4"))
    grid()
  })

  output$resultsA <- renderPrint({
    stats <- calculate_results_A()

    #Verificam daca densitatea este valida
    if (abs(stats$integral_f - 1) > 1e-6) {  #Luam in considerare eroare de rotunjire
      cat("C nu face ca f(x) să fie o densitate validă!\n")
      cat("Integrală =", stats$integral_f, "\n")
    }
    else{
    cat("Media E[X] =", stats$average, "\n")
    cat("Varianța Var(X) =", stats$variance, "\n")
    }
  })

  ##############################################################

  #b) f(x) = ax + bx^2, x din (0,1), a,b din R
  ##############################################################

  f_x_B  <- reactive(function(x) input$ex2A * x + input$ex2B * x^2)

  calculate_results_B <- reactive({

    #integrala de validare

    integral_f <- integrate(f_x_B(), 0, 1)$value

    #Medie E(x)
    average <- integrate(function(x) x * f_x_B()(x), 0, 1)$value

    #Medie x^2 E(x^2)
    averageSqr <- integrate(function(x) x^2 * f_x_B()(x),  0, 1)$value

    #Varianta
    variance <- averageSqr - average^2

    list(integral_f = integral_f, average = average, variance = variance)
  })

  output$densityB <-renderPlot({
    interval <- seq(0.001, 0.999, 0.001)
    f <- function(x) input$ex2A * x + input$ex2B * x^2
    f_x <- sapply(interval, f)
    plot(interval, f_x, col = "blue", lwd =2,
         xlab = "x", ylab = "F(x)",
         main = paste("Densitatea pentru f(x) = ax + bx^2"))
    grid()
  })

  output$resultsB <- renderPrint({
```

```
    stats <- calculate_results_B()
    a <- input$ex2A
    b <- input$ex2B
    check <- TRUE

    if (abs(stats$integral_f - 1) > 1e-6 & check) {
      cat("A, B nu fac ca f(x) să fie o densitate validă!\n")
      cat("Integrală =", stats$integral_f, "\n")
      check <- FALSE
    }

    if(check & a < 0){
      cat("A, B nu fac ca f(x) să fie o densitate validă!\n")
      check <- FALSE
    }

    if(check & b < 0){
      if(a == 0 | a <= 0){
        cat("A, B nu fac ca f(x) să fie o densitate validă!\n")
        check <- FALSE
      }
      else{
        if(a < -b)
        {
          cat("A, B nu fac ca f(x) să fie o densitate validă!\n")
          check <- FALSE
        }
      }
    }

    if(check)
    {
      cat("Media E[X] =", stats$average, "\n")
      cat("Varianța Var(X) =", stats$variance, "\n")
    }
  })
  ##############################################################

  #c) f(x) = 4/x(x+1)(x+2), x din N\{0}
  ##############################################################

  f_x_C <- reactive(function(x) 4/(x*(x+1)*(x+2)))

  output$densityC <- renderPlot({
    dom <- 1 : 30
    pdf <- function(x) 4/(x*(x+1)*(x+2))
    p_x <- sapply(dom, pdf)
    plot(dom, p_x, col = "blue", lwd =2,
         xlab = "x", ylab = "F(x)",
         main = paste("X cu pdf = 4/(x(x+1)(x+2))"))
    grid()
  })

  calculate_results_C <- reactive({

    dom <- 1 : 30000

    pdf_avg <- function(x) 4/((x+1)*(x+2))
    pdf_avg_sqr <- function(x) 4*x/((x+1)*(x+2))

    pAvg_x <- sapply(dom, pdf_avg)
    pAvgSqr_x <- sapply(dom, pdf_avg_sqr)

    #Media E[x]
    average <- sum(pAvg_x)

    #Media E[X^2]
    averageSqr <- sum(pAvgSqr_x)

    #Varianta
```

```r
  variance <- averageSqr - average^2

  list(variance = variance,average = average)
})

output$resultsC <- renderPrint({
    stats <- calculate_results_C()

    cat("Media E[X] =", stats$average, "\n")
    cat("Varianța Var(X) =", stats$variance, "\n")
})

#d) f(x) = lg(x/x+1), x din {1,2..9}
##############################################################

f_x_D <- function(x) log10(x/(x+1))

output$densityD <- renderPlot({
  dom <- 1:9
  p_x <- sapply(dom, f_x_D)
  plot(dom, p_x, col = "blue", lwd =2,
       xlab = "x", ylab = "F(x)",
       main = paste("X cu pdf = lg(x/x+1)"))
  grid()
})

calculate_results_D <- reactive({

  dom <- 1 : 9

  pdf_avg <- function(x) x * log10(x/(x+1))
  pdf_avg_sqr <- function(x) x^2 * log10(x/(x+1))

  pAvg_x <- sapply(dom, pdf_avg)
  pAvgSqr_x <- sapply(dom, pdf_avg_sqr)

  #Media E[x]
  average <- sum(pAvg_x)

  #Media E[X^2]
  averageSqr <- sum(pAvgSqr_x)

  #Varianta
  variance <- averageSqr - average^2

  list(variance = variance,average = average)
})

output$resultsD <- renderPrint({
  stats <- calculate_results_D()

  cat("Media E[X] =", stats$average, "\n")
  cat("Varianța Var(X) =", stats$variance, "\n")
})

##############################################################
#e) f(x) = teta^2/1+teta * (1+x)e^-teta*x, x>0, teta>0

#t <- reactive(input$teta)
#f_x_E <- reactive(function(x) ((t^2)/(1+t))*(1+x)*(2.718)^(-t*x) )

output$densityE <- renderPlot({
  t <- input$teta
  interval <- seq(0.001, 50, 0.001)

  f <- function(x) ((t^2)/(1+t))*(1+x)*(exp(1))^(-t*x)
  f_x <- sapply(interval, f)

  plot(interval, f_x, col = "blue", lwd =2,
       xlab = "x", ylab = "F(x)", type = "l",
```

```
        main = paste("X cu f(x) = teta^2/1+teta * (1+x)e^-teta*x"))
    grid()
  })

  calculate_results_E <- reactive({

    #integrala de validare
    t <- input$teta
    f_x_E <- function(x) ((t*t)/(1+t))*(1+x)*(exp(1))^(-t*x)

    integral_f <- integrate(f_x_E, 0, upper = Inf)$value

    #Medie E(x)
    average <- integrate(function(x) x * f_x_E(x), 0, upper = Inf)$value

    #Medie x^2 E(x^2)
    averageSqr <- integrate(function(x) x^2 * f_x_E(x),  0, upper = Inf)$value

    #Varianta
    variance <- averageSqr - average^2

    list(integral_f = integral_f, average = average, variance = variance)
  })

  output$resultsE <- renderPrint({
    stats <- calculate_results_E()

    if (abs(stats$integral_f - 1) > 1e-6) {
      cat("Teta nu face ca f(x) să fie o densitate validă!\n")
      cat("Integrală =", stats$integral_f, "\n")
    }
    else{
      cat("Media E[X] =", stats$average, "\n")
      cat("Varianța Var(X) =", stats$variance, "\n")
    }
  })

  ##############################################################
  #f) f(x) = 1/*3e^x, x<0; f(x) = 1/3, 0<=x<1; f(x) = 1/3*x^1-x, x>= 1

  output$densityF <- renderPlot({
    interval1 <- seq(-10,-0.001, 0.001)
    interval2 <- seq(0, 0.999, 0.001)
    interval3 <- seq(1, 10, 0.001)
    fullInterval <- seq(-10,10,0.001)

    f1 <- function(x) 1/3 * (exp(1))^x
    f2 <- function(x) 1/3
    f3 <- function(x) 1/3* (exp(1))^(1-x)

    f_x_1 <- sapply(interval1, f1)
    f_x_2 <- sapply(interval2, f2)
    f_x_3 <- sapply(interval3, f3)


    plot(fullInterval, sapply(fullInterval, function(x) 0), type = "l", col = "white", lwd = 2, ylim = c(0, 1),
        xlab = "x", ylab = "f(x)", main = "Densitatea lui X din F")
    lines(interval1,f_x_1, col = "red", lwd = 2)
    lines(interval2,f_x_2, col = "blue", lwd = 2)
    lines(interval3,f_x_3, col = "green", lwd = 2)
  })

  calculate_results_F <- reactive({

    interval1 <- seq(-10,-0.001, 0.001)
    interval2 <- seq(0, 0.999, 0.001)
    interval3 <- seq(1, 10, 0.001)

    f1 <- function(x) 1/3 * (exp(1))^x
```

```r
    f2 <- function(x) 1/3
    f3 <- function(x) 1/3* (exp(1))^(1-x)

    #Calculam mediile pe ramuri
    avg1 <- integrate(function(x) x * f1(x), lower = Inf, 0)$value
    avg2 <- integrate(function(x) x * f2(x), 0, 1)$value
    avg3 <- integrate(function(x) x * f3(x), 1, upper = Inf)$value

    #Calculam ponderile
    P1 <- integrate(f1, lower = Inf, 0)$value
    P2 <- integrate(Vectorize(f2), lower = 0, 1)$value
    P3 <- integrate(f3, lower = 1, Inf)$value

    #Medie E(x)
    average <- P1 * avg1 + P2 * avg2 + P3 * avg3

    avg1Sqr <- integrate(function(x) x^2 * f1(x), lower = Inf, 0)$value
    avg2Sqr <- integrate(function(x) x^2 * f2(x), 0, 1)$value
    avg3Sqr <- integrate(function(x) x^2 * f3(x), 1, upper = Inf)$value

    #Medie x^2 E(x^2)

    averageSqr <- P1*avg1Sqr + P2*avg2Sqr + P3*avg3Sqr

    #Varianta
    variance <- averageSqr - average^2

    list(average = average, variance = variance)
  })

  output$resultsF <- renderPrint({
    stats <- calculate_results_F()

    cat("Media E[X] =", stats$average, "\n")
    cat("Varianța Var(X) =", stats$variance, "\n")
  })
  ##############################################################
  #g) f(x) = 1/(pi*(1 + x^2))

  output$densityG <- renderPlot({

    interval <- seq(-10, 10, 0.001)
    f <- function(x) 1/(pi * (1 + x^2))
    f_x <- sapply(interval, f)
    plot(interval, f_x, col = "blue", lwd =2,
         xlab = "x", ylab = "f(x)", type = "l",
         main = paste("X cu f(x) = 1/(pi*(1 + x^2))"))
    grid()
  })

  calculate_results_G <- reactive({

    f_x_E <- function(x) 1/(pi * (1 + x^2))

    #Medie E(x)
    average <- integrate(function(x) x * f_x_E(x), lower = -100, upper = 100)$value

    #Medie x^2 E(x^2)
    averageSqr <- integrate(function(x) x^2 * f_x_E(x),  lower = -100, upper = 100)$value

    #Varianta
    variance <- averageSqr - average^2

    list(average = average, variance = variance)
  })

  output$resultsG <- renderPrint({
    stats <- calculate_results_G()

      cat("Media E[X] =", stats$average, "\n")
```

```
        cat("Varianța Var(X) =", stats$variance, "\n")
    })

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

Cod Problema 3

```
# REZOLVARE a)
# REZOLVARE a)
# REZOLVARE a)
# REZOLVARE a)
# REZOLVARE a)
# REZOLVARE a)
# REZOLVARE a)
# REZOLVARE a)
# REZOLVARE a)
# REZOLVARE a)
# REZOLVARE a)
# REZOLVARE a)

# Esantionul
sample_a <- 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)

# Estimator MLE
theta_mle_a <- sum(sample_a) / (2 * length(sample_a))

# Estimator MM
theta_mm_a <- mean(sample_a) / 2

# Definirea functiei de log-verosimilitate
log_likelihood_a <- function(theta) {
  -2 * theta * length(sample_a) + sum(sample_a) * log(2 * theta) - sum(lfactorial(sample_a))
}

# Maximizarea log-verosimilitatii
result_a <- optim(par = 1, fn = log_likelihood_a, method = "Brent", lower = 0, upper = 100, control = list(fnscale = -1))

# Estimator numeric
theta_numeric_a <- result_a$par

# Afisare rezultate
cat("Estimator numeric pentru theta:", theta_numeric_a, "\n")
cat("Estimator MLE pentru theta:", theta_mle_a, "\n")
cat("Estimator MM pentru theta:", theta_mm_a, "\n")


# REZOLVARE b)
# REZOLVARE b)
# REZOLVARE b)
# REZOLVARE b)
# REZOLVARE b)
# REZOLVARE b)
# REZOLVARE b)
# REZOLVARE b)
# REZOLVARE b)
# REZOLVARE b)
# REZOLVARE b)
# REZOLVARE b)

# Esantionul
sample_b <- 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)

# Estimator MLE
theta_mle_b <- sum(sample_b) / (14 * length(sample_b))

# Estimator MM
theta_mm_b <- mean(sample_b) / 14

# Definirea functiei de log-verosimilitate
log_likelihood_b <- function(theta) {
  sum(dbinom(sample_b, size = 14, prob = theta, log = TRUE))
}

# Maximizarea log-verosimilitatii
result_b <- optim(par = 0.5, fn = log_likelihood_b, method = "Brent", lower = 0, upper = 1, control = list(fnscale = -1))

# Estimator numeric
theta_numeric_b <- result_b$par

# Afisare rezultate
cat("Estimator numeric pentru theta:", theta_numeric_b, "\n")
cat("Estimator MLE pentru theta:", theta_mle_b, "\n")
cat("Estimator MM pentru theta:", theta_mm_b, "\n")

# REZOLVARE c)
# REZOLVARE c)
# REZOLVARE c)
# REZOLVARE c)
# REZOLVARE c)
# REZOLVARE c)
# REZOLVARE c)
# REZOLVARE c)
# REZOLVARE c)
# REZOLVARE c)
# REZOLVARE c)
# REZOLVARE c)

# Esantionul
sample_c <- 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)

# Parametrul alpha
alpha_c <- 7

# Estimator MLE
theta_mle_c <- sum(sample_c) / (alpha_c * length(sample_c))

# Estimator MM
theta_mm_c <- mean(sample_c) / alpha_c

# Definirea functiei de log-verosimilitate
log_likelihood_c <- function(theta) {
  sum(dgamma(sample_c, shape = alpha_c, scale = theta, log = TRUE))
}

# Maximizarea log-verosimilității
result_c <- optim(par = 1, fn = log_likelihood_c, method = "Brent", lower = 0, upper = 100, control = list(fnscale = -1))
```

```r

# Estimator numeric
theta_numeric_c <- result_c$par

# Afisare rezultate
cat("Estimator MLE pentru theta:", theta_mle_c, "\n")
cat("Estimator MM pentru theta:", theta_mm_c, "\n")
cat("Estimator numeric pentru theta:", theta_numeric_c, "\n")

# REZOLVARE d)
# REZOLVARE d)
# REZOLVARE d)
# REZOLVARE d)
# REZOLVARE d)
# REZOLVARE d)
# REZOLVARE d)
# REZOLVARE d)
# REZOLVARE d)
# REZOLVARE d)
# REZOLVARE d)
# REZOLVARE d)

# Esantionul
sample_d <- 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)

# Estimator MLE
theta_mle_d <- mean(sample_d)

# Estimator MM
theta_mm_d <- mean(sample_d)

# Definirea functiei de log-verosimilitate
log_likelihood_d <- function(theta) {
  sum(dgeom(sample_d, prob = 1 / (1 + theta), log = TRUE))
}

# Maximizarea log-verosimilității
result_d <- optim(par = 1, fn = log_likelihood_d, method = "Brent", lower = 0, upper = 1000,
control = list(fnscale = -1))

# Estimator numeric
theta_numeric_d <- result_d$par

# Afisare rezultate
cat("Estimator MLE pentru theta:", theta_mle_d, "\n")
cat("Estimator MM pentru theta:", theta_mm_d, "\n")
cat("Estimator numeric pentru theta:", theta_numeric_d, "\n")


# REZOLVARE e)
# REZOLVARE e)
# REZOLVARE e)
# REZOLVARE e)
# REZOLVARE e)
# REZOLVARE e)
# REZOLVARE e)
# REZOLVARE e)
# REZOLVARE e)
# REZOLVARE e)
# REZOLVARE e)
# REZOLVARE e)

# Esantionul
sample_e <- 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)

# Parametrul alpha
alpha_e <- 3

# Estimator MLE
theta_mle_e <- sum(sample_e) / (alpha_e * length(sample_e))

# Estimator MM
theta_mm_e <- mean(sample_e) / alpha_e

# Definirea functiei de log-verosimilitate
log_likelihood_e <- function(theta) {
  sum(dgamma(sample_e, shape = alpha_e, scale = theta, log = TRUE))
}

# Maximizarea log-verosimilității
result_e <- optim(par = 1, fn = log_likelihood_e, method = "Brent", lower = 0, upper = 100, control = list(fnscale = -1))

# Estimator numeric
theta_numeric_e <- result_e$par

# Afișare rezultate
cat("Estimator MLE pentru theta:", theta_mle_e, "\n")
cat("Estimator MM pentru theta:", theta_mm_e, "\n")
cat("Estimator numeric pentru theta:", theta_numeric_e, "\n")
```