PROIECT STATISTICĂ

Studenți: Olaru Alexandru-Vergiliu (311)
Matei Elena-Elisabeta (312)
Trache Andrei-Daniel (311)
Negru Liviu-Bogdan (312)

Facultatea de Matematică și Informatică - Universitatea din București
Specializarea Matematică - Informatică

Februarie 2022

PROBLEMA I

1/ Pentru a determina numărul de elemente aproximativ din intervalul (a, b) a căror sumă depășește $c \geq b$, s-a creat o funcție average în care se generează uniform numere din (a, b). Acestea au fost contorizate cu ajutorul variabilei counter și însumate în variabila localSum până când se îndeplinește criteriul de ieșire din structura repetitivă.

Funcția average a fost apelată de 10^9 ori cu ajutorul funcției replicate, ce a returnat un vector cu care a fost făcută media aproximărilor. În plus, pentru cazul

iterations $\leftarrow 10^9$;
$a \leftarrow 0$;
$b \leftarrow 1$;
$c \leftarrow 1$;,

s-a observat că aproximarea aproxK = e, unde e este numărul lui Euler.

```
## 1.

a <- 1
b <- 6
c <- 20
iterations <- 10^7

average <- function() { #calculam numarul de elemente aleatorii necesare din (a,b) pentru a depasi c

  if(a+b >= 0) {
    localSum <- 0
    counter <- 0

    repeat {
      x <- runif(1, a, b)
      localSum <- localSum + x
      counter <- counter + 1
      if(localSum > c) break

      }
    return(counter)

  } else
    return(Inf)
}

aproxK <- sum(replicate(iterations, average()))/iterations #calculam media
```

2/ $X \sim Unif(a, b)$, $a \le b \le c$, $b \ge 0$.

Am văzut la 1) că K = numărul de variabile aleatoare care însumate dau suma c.

Considerăm distribuția $f_X : (a,b) \to \mathbb{R}$, $f(x) = \frac{1}{b-a}$.

Media $E[X] = \int_a^b x f_X(x) dx = \int_a^b \frac{x}{b-a} dx = \int_a^b \frac{x}{b-a} dx = \frac{x^2}{2(b-a)} \Big|_a^b = \frac{b^2 - a^2}{2(b-a)}$.

Obținem $E[X] = \frac{a+b}{2}$.

Avem următoarele 3 cazuri:

$\boxed{\text{Cazul 1}}$ $a+b>0 \Rightarrow E[X]>0 \Rightarrow K = \left[\frac{c}{E[X]} + 1\right] = \left[\frac{c}{\frac{b+a}{2}}\right] + 1 = \left[\frac{2c}{b+a}\right] + 1$.

Deci, pentru $a+b>0$, avem $K = \left[\frac{2c}{a+b}\right] + 1$.

$\boxed{\text{Cazul 2}}$ $\left.\begin{array}{l} a+b<0 \Rightarrow E[X]<0 \\ c \ge 0 \end{array}\right\} \Rightarrow$ în acest caz, ne așteptăm ca, în medie, suma variabilelor să nu atingă c niciodată.

$\boxed{\text{Cazul 3}}$ $\left.\begin{array}{l} a+b=0 \Rightarrow E[X]=0 \\ \text{Cum } X \in (a,b) \text{ și } c \ge b \end{array}\right\} \Rightarrow$ aceeași concluzie ca în cazul 2.

După ce a fost calculată valoarea exactă, având deja valoarea aproximativă am creat o variabilă errorK, unde am calculat în procente eroarea dintre cele 2 valori.

```
## 2.

exactK <- ((2*c)/(a+b)) + 1
errorK <- (abs(exactK - aproxK)/exactK)*100 #calculam eroarea in procente
```

Values	
a	1
aproxK	6.2995505
b	6
c	20
errorK	6.17690744680852
exactK	6.71428571428571
iterations	1e+07
Functions	
average	function ()

PROBLEMA II

1/ Pentru a reduce semnificativ din costul computațional, au fost generate o singură dată mulțimile alfa, reprezentând vectorul probabilităților, și Ti, reprezentând vectorul duratelor de timp ale fiecărei etape, fiind necesare în algoritmul curent. În plus, iterations reprezintă numărul persoanelor, iar n reprezintă numărul de etape dintr-o secvență.

Pentru a determina timpul în care persoana termină secvența, a fost creată funcția runStages în care se ia o variabilă totalTime pentru a calcula timpul total și o uniformă nextStage pentru a decide dacă se trece la etapa următoare. Aceasta din urmă este comparată cu probabilitatea de trecere alfa[i]. În final, funcția returnează timpul total pe care persoana îl petrece în secvența de etape până la ieșire. Această funcție se execută pentru cele 10^6 persoane, de unde rezultă un vector cu care se calculează timpul mediu petrecut pentru rezolvarea celor n etape.

```r
## 1.

iterations <- 10^6 #numarul de persoane
n <- 20 #numarul de etape

alfa <- runif(n-1, 0, 1) #probabilitatea ca o persoana sa termine fiecare activitate, ca vector
Ti <- rexp(n, runif(1, 0, 1)) #timpul necesar pentru fiecare activitate, ca vector

runStages <- function() { #persoana isi parcurge etapele pana la esec
  totalTime <- Ti[1]
  nextStage <- runif(1, 0, 1)

  if(nextStage < alfa[1]) {
    totalTime <- totalTime + Ti[2]

      for (i in 2:(n-1)) {
        nextStage <- runif(1)

        if(nextStage < alfa[i]) {
          totalTime <- totalTime + Ti[i+1]
        }
      }
  }

  return(totalTime)
}

totalTimes <- replicate(iterations,runStages()) #timpul T pentru fiecare persoana, ca vector
averageTime <- sum(totalTimes)/iterations # media timpului necesar fiecarei persoane
```

2/ $i=\overline{1,m}$, $T_i \sim Exp(\lambda_i)$, $\alpha_i \sim Unif(0,1)$, $\alpha_1, \dots, \alpha_{m-1}$ independente

T = timpul total pentru finalizarea activității

$\mathbb{E}[T] = ?$

Avem pentru $\forall i$, $\mathbb{E}[T_i] = \frac{1}{\lambda_i}$, $\mathbb{E}[\alpha_i] = \frac{1}{2}$.

De asemenea, avem că T, timpul total petrecut de persoana A, poate lua valorile T_1, T_1+T_2, $T_1+T_2+T_3$, ..., $T_1+T_2+\dots+T_m$.

Așadar, $P(T=T_1) = 1-\alpha_1$ (A se oprește după etapa 1);

$P(T=T_1+T_2) = \alpha_1(1-\alpha_2)$ (A termină etapa 1, trece mai departe la etapa 2 și se oprește după aceasta).

Analog, $P(T=T_1+T_2+\dots+T_k) = \alpha_1 \cdot \alpha_2 \cdot \dots \cdot \alpha_{k-1}(1-\alpha_k)$, $k=\overline{1,m-1}$.

$$P\left(T=\sum_{i=1}^{m} T_i\right) = \alpha_1 \cdot \alpha_2 \cdot \dots \cdot \alpha_{m-1}$$

$$\mathbb{E}[P(T=T_1)] = \mathbb{E}[1-\alpha_1] = 1-\frac{1}{2} = \frac{1}{2}$$

$$\mathbb{E}[P(T=T_1+T_2)] = \mathbb{E}[\alpha_1(1-\alpha_2)] = \frac{1}{4}$$

Analog, $\mathbb{E}[P(T=T_1+T_2+\dots+T_k)] = \frac{1}{2^k}$, $k=\overline{1,m-1}$

$$\mathbb{E}[P(T=T_1+T_2+\dots+T_m)] = \frac{1}{2^{m-1}}$$

Notăm $T^{(k)} = T_1+T_2+\dots+T_k$ și $p^{(k)} = \mathbb{E}[P(T=T^{(k)})]$.

Avem deci
$$\mathbb{E}[T] = \sum_{k=1}^{m} \mathbb{E}[T^{(k)}] \cdot p^{(k)}$$
$$= \frac{1}{2}\cdot\frac{1}{\lambda_1} + \frac{1}{2^2}\left(\frac{1}{\lambda_1}+\frac{1}{\lambda_2}\right) + \dots + \frac{1}{2^{m-1}}\left(\frac{1}{\lambda_1}+\dots+\frac{1}{\lambda_{m-1}}\right) + \frac{1}{2^{m-1}}\left(\frac{1}{\lambda_1}+\dots+\frac{1}{\lambda_m}\right)$$

3/ Pentru a afla probabilitatea în care o persoană termină cele n etape ale unei secvențe, s-a făcut produsul alfa[i]-urilor.

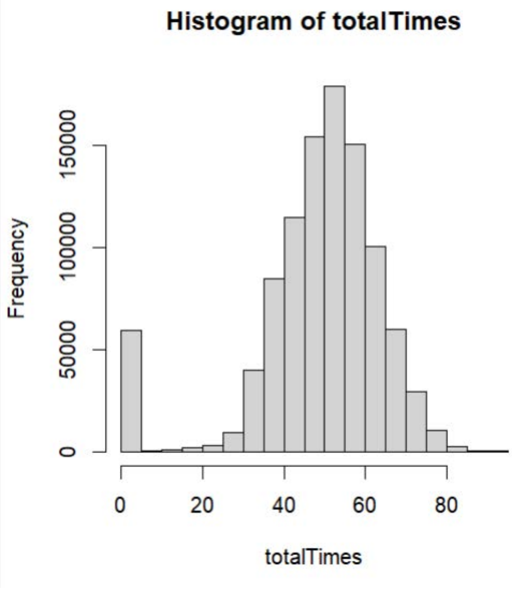
```
## 3.

finishProb <- prod(alfa) #se inmultesc probabilitatile din alfa
```

4/ Pentru a calcula probabilitatea ca o persoană ca o persoană să iasă din secvență înainte de timpul sigma, care este luat aleator folosind Ti, se adună timpii până se atinge sigma, contorizându-se poziția de ieșire în variabila counter, apoi se înmulțesc probabilitățile din alfa până la counter-1. La final, acest rezultat se înmulțește cu 1-alfa[counter], aceasta din urmă fiind probabilitatea de ieșire.

```
## 4.

sigma <- runif(1, min(cumsum(Ti)), max(cumsum(Ti))) #generam un timp aleator
counter <- 0
timeSum <- 0

if(sigma >= Ti[1]) { #daca sigma e prea mic atunci nu se termian prima etapa

  for(i in 1:n) {

    if((timeSum + Ti[i]) <= sigma) {
      timeSum <- timeSum + Ti[i]
      counter <-  counter + 1

    } else {
      break
    }
  }
  sigmaProb <- prod(alfa[1:(counter-1)])*(1-alfa[counter])

} else {
  sigmaProb <- 0

}
```

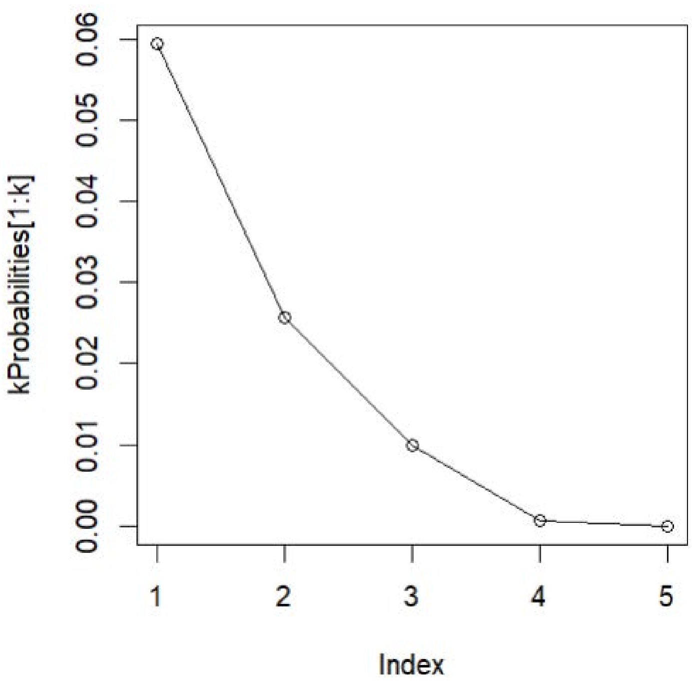
5/ Pentru a determina timpul minim și timpul maxim în care o persoană termină activitatea, s-au apelat min(times), respectiv max(times), unde times reprezintă vectorul celor 10^6 persoane.

```
## 5.

minTime <- min(totalTimes) #timpul minim
maxTime <- max(totalTimes) #timpul maxim
hist(totalTimes)
```



6/ Pentru a determina probabilitatea de ieșire după o anumită etapă, se alege k etapa și se înmulțesc probabilitățile alfa[i] până la k-1, apoi se înmulțește cu probabilitatea de ieșire 1-alfa[i]. Valorile anterioare lui k sunt, de asemenea, adăugate într-un vector probabilities cu care se crează un plot. În acest plot se poate observa faptul că este o probabilitate mai mare ca o persoană să iasă înainte să se termine cele k etape.

```
## 6.

k <- 5
probabilities <- c(1 - alfa[1])

for (i in 2:(n-1)) {
  tempValue <-  prod(alfa[1:(i-1)])*(1-alfa[i])
  probabilities <- c(probabilities, tempValue)
}

kProbabilities <- cumprod(probabilities) #vectorul de probabilitati de a ajunge la k
kProb <- kProbabilities[k] #prbabilitatea de ajunge la etapa k

plot(kProbabilities[1:k])
lines(kProbabilities[1:k])
```



Values	
alfa	num [1:19] 0.941 0.54 0.242 0.431…
averageTime	48.7916529679921
counter	14
finishProb	2.51505655620341e-10
i	19L
iterations	1e+06
k	5
kProb	3.23477046646175e-05
kProbabiliti…	num [1:19] 5.94e-02 2.57e-02 9.89…
maxTime	93.9794317960589
minTime	3.17448657228229
n	20
probabilities	num [1:19] 0.0594 0.4326 0.385 0.…
sigma	70.8369098514613
sigmaProb	1.56235500269097e-05
tempValue	4.71328684641105e-10
Ti	num [1:20] 3.174 2.636 3.343 0.48…
timeSum	68.8537747138407
totalTimes	Large numeric (1000000 elements, …
Functions	
runStages	function ()

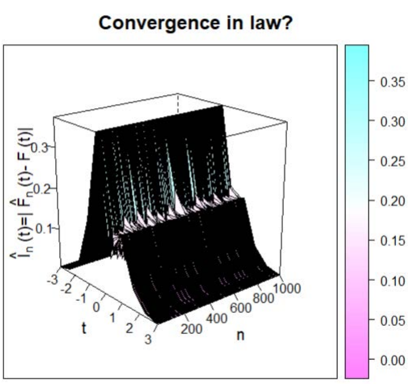
PROBLEMA III

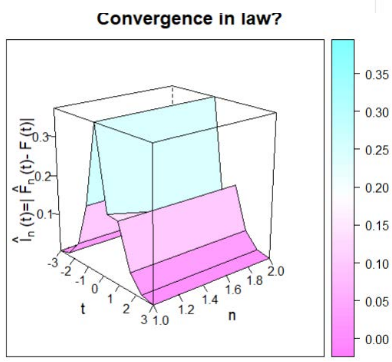
1/ Pentru a verifica dacă șirul de variabile aleatoare $X_1, X_2, \dots, X_n$ converge în lege (distribuție) la X, aplicăm funcția check.convergence pentru ambele distribuții și comparăm graficele rezultate. Ținem cont că toate X_i-urile aparțin unei distribuții Beta$(\frac{1}{n}, \frac{1}{n})$ iar X este distribuit pe Binomial$(1, \frac{1}{2})$. Pentru a genera probabilități aleatoare cu distribuția Beta, ne folosim de funcția valuesGenBeta care ne întoarce o listă cu n probabilități. Facem același lucru și pentru variabila repartizată binomial, adică funcția valuesGenBin care ne întoarce o probabilitate la care șirul X_i ar trebui să conveargă.

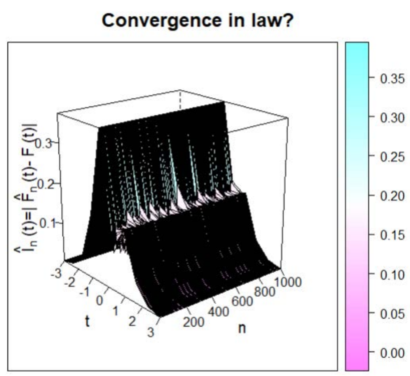
Apelăm funcția check.convergence pentru fiecare distribuție, cu parametrii respectivi: nmax = numărul de puncte n, M = numărul de seturi de n valori care să fie generate, genXn = funcția care ne generează șirul de probabilități, mode = modul de convergență (în cazul nostru, avem "L"). Astfel, comparând ~~ale~~ 2 forme rezultate, observăm că șirul X_i converge în lege la X, când $n \to \infty$.

Pentru cazul în care X_i sunt în distribuția Beta$(\frac{a}{n}, \frac{b}{n})$, cu $a, b > 0$, observăm că nu se aplică convergența în lege deoarece formele rezultate sunt diferite. Convergența se aplică în schimb, doar în cazul în care $a = b$.

```r
# III.) ########################################################################

require("ConvergenceConcepts")


## 1.

n <- 1000

valuesGenBeta <- function(n) {rbeta(n, 1/n, 1/n)}
valuesGenBin  <- function(n) {rbinom(n, 1, 1/2)}

dataBetaL <- check.convergence(nmax = n, M = 5000, genXn = valuesGenBeta, mode = "L")
dataBinL  <- check.convergence(nmax = 2, M = 5000, genXn = valuesGenBin,  mode = "L")

a <- runif(1, 0, 100)
b <- runif(1, 0, 100)

valuesGenBetaC <- function(n) {rbeta(n, a/n, b/n)}

dataBetaLC <- check.convergence(nmax = n, M = 5000, genXn = valuesGenBetaC, mode = "L")
dataBinL   <- check.convergence(nmax = 2, M = 5000, genXn = valuesGenBin,   mode = "L")
```

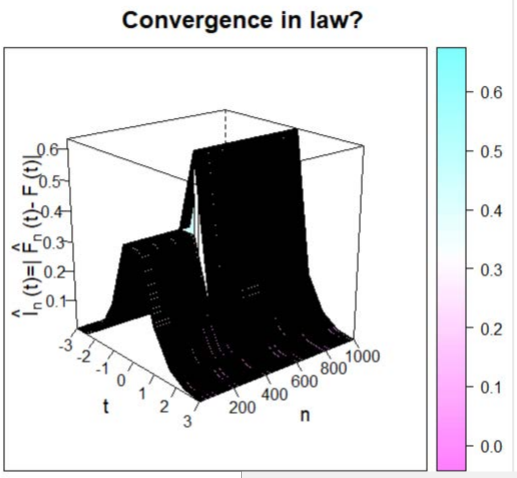


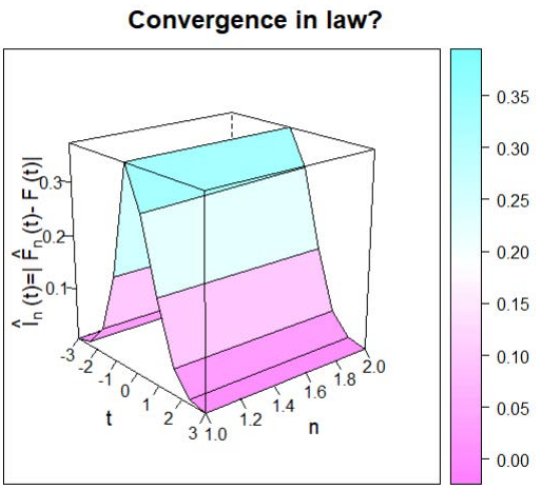


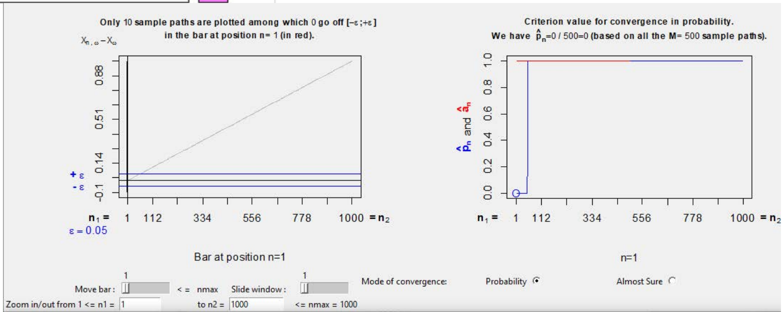


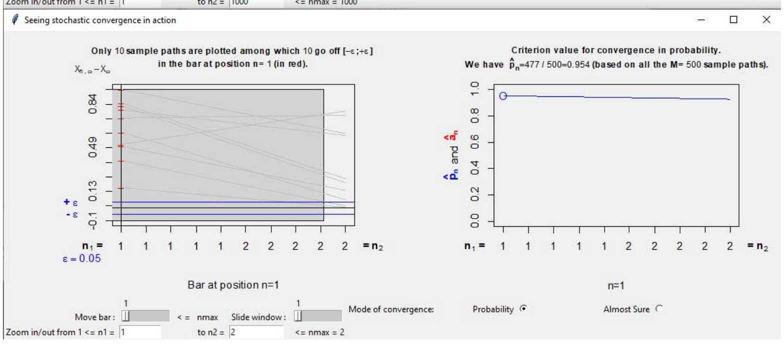
2/ Urmăm un proces asemănător ca la punctul anterior, numai că X_i sunt distribuite uniform pe mulțimea $\{1/n, 2/n, \ldots, 1\}$ iar X este uniform distribuit în $(0,1)$. Generăm lista de probabilități cu funcția valuesGenUnifC, o variabilă uniformă cu funcția valuesGenUnif și comparăm rezultatele. Pentru convergența în distribuție observăm că cele 2 forme rezultate sunt diferite, deci $X_i \not\to X$, iar pentru convergența în probabilitate, observăm că probabilitatea la care șirul X_i converge pe 0, care este diferită de probabilitatea repartizată uniform X.

```r
## 2.
valuesGenUnif <- function(n) {runif(n, 0, 1)}
valuesGenUnifC <- function(n) {
  X <- c()

  for(i in 1:n) {
    X  <- c(X,(runif(1, i/n, i/n)))
  }
  return(X)
}

dataUnifLC <- check.convergence(nmax=n, M = 5000, genXn=valuesGenUnifC, mode="L")
dataUnifL  <- check.convergence(nmax=2, M = 5000, genXn=valuesGenUnif,  mode="L")

dataUnifPC <- check.convergence(nmax=n, M = 5000, genXn=valuesGenUnifC, mode="p")
dataUniPL  <- check.convergence(nmax=2, M = 5000, genXn=valuesGenUnif,  mode="p")
```









PROBLEMA IV

1/ a) $\text{Binom}(3,p) \Rightarrow p_K = P(x=k) = C_3^K p^K (1-p)^{3-K}$, $K \in \{0,1,2,3\}$

$$= C_3^K \exp\left(\log\left(\frac{p}{1-p}\right) \cdot K + 3\log(1-p)\right)$$

$$= h(K)\exp(\eta(p)\cdot T(K) - A(p))$$

Avem: $h(K) = C_3^K$

$\eta(p) = \log\left(\frac{p}{1-p}\right)$ $\Rightarrow$ repartiția binomială face parte din familia exponențială.

$T(K) = K$

$A(p) = -3\log(1-p)$

(considerăm $\theta = p$)

b) $\text{Binom}(m,p) \Rightarrow p_K = P(x=k) = C_m^K p^K (1-p)^{m-K}$

$$= C_m^K \exp\left(\log\left(\frac{p}{1-p}\right)\cdot K + m\log(1-p)\right)$$

$$= h(K)\exp(\eta(p)\cdot T(K) - A(p))$$

Avem: $h(K) = C_m^K$

$\eta(p) = \log\left(\frac{p}{1-p}\right)$ $\Rightarrow$ analog a)

$T(K) = K$

$A(p) = -m\log(1-p)$

(considerăm $\theta = p$)

c) $\text{Geom}(p)$

Fie $f: \mathbb{N}^* \to \mathbb{R}$, $f(x) = (1-p)^{x-1}\cdot p$ și consider funcția $I(x) = \begin{cases} 1 & x \in \mathbb{N}^* \\ 0 & \text{altfel} \end{cases}$

Avem că $f(x) = I(x)\cdot p\cdot(1-p)^{x-1}$.

$$= I(x)\cdot\left(\frac{p}{1-p}\right)\cdot(1-p)^x$$

$$= I(x)\cdot\frac{p}{1-p}\exp(x\log(1-p))$$

Avem: $h(x) = I(x)\cdot\frac{p}{1-p}$

$\theta = p$

$T(x) = x$

$\eta(\theta) = \log(1-\theta)$

Deci, $\text{Geom}(p) \in$ familiei exponențiale

d) $Pois(\lambda) \Rightarrow P_\theta = \mathbb{P}(X=n) = e^{-\lambda} \cdot \frac{\lambda^n}{n!}$

$$= \frac{1}{n!} \exp(\log(\lambda) \cdot n - \lambda)$$
$$= h(n) \exp(\eta(\lambda) \cdot T(n) - A(\lambda))$$

Avem : $h(n) = \frac{1}{n!}$

(considerăm $x = n$) $\Rightarrow$ repartiția Poisson face parte din familia exponențială

$\eta(\lambda) = \log(\lambda)$

(considerăm $\lambda = \theta$)

$T(n) = n$

$A(\lambda) = \lambda$

e) Gamma(α, β)

$$p(x \mid \alpha, \beta) = \frac{\alpha^\beta x^{\beta-1} e^{-\alpha x}}{\Gamma(\beta)}, \quad 0 < x < \infty$$

iar $\Gamma(\beta) = \int_0^\infty \alpha^\beta x^{\beta-1} e^{-\alpha x} dx$

$$p(x \mid \alpha, \beta) = \left[\frac{x^{\beta-1}}{\Gamma(\beta)}\right] \exp(-\alpha x + \beta \log(\alpha))$$
$$= h(x) \exp(\eta T(x) - A(\eta))$$

Avem : $h(x) = \left[\frac{x^{\beta-1}}{\Gamma(\beta)}\right]$

$\eta = -\alpha$ $\Rightarrow$ repartiția Gamma face parte din familia exponențială

$T(x) = x$

$A(\eta) = -\beta \log(\alpha) = -\beta \log(-\alpha)$

f) $Beta(\alpha, \beta) \Rightarrow$ fie $f(x) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)+\Gamma(\beta)} x^{\alpha-1}(1-x)^{\beta-1}$

$$f(x) = \exp\left(\log\left(\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)+\Gamma(\beta)}\right)\right) \cdot \frac{1}{x(1-x)} \exp(\alpha \log(x) + \beta \log(1-x))$$
$$= \frac{1}{x(1-x)} \exp\left(\log(\Gamma(\alpha+\beta)) - \log(\Gamma(\alpha)) - \log(\Gamma(\beta)) + \alpha \log(x) + \beta \log(1-x)\right)$$

Avem : $h(x) = \frac{1}{x(1-x)}$; $\theta = (\alpha, \beta)$ $\Rightarrow$ repartiția Beta face parte din familia exponențială.

$$\alpha \log(x) + \beta \log(1-x) = \underbrace{(\alpha \ \beta)}_{\eta(\theta)} \underbrace{\begin{bmatrix} \log(x) \\ \log(1-x) \end{bmatrix}}_{T(x)}$$

$$A(\theta) = \left(\log(\Gamma(\alpha+\beta)) - \log(\Gamma(\alpha)) - \log(\Gamma(\beta))\right)$$

g) $\chi^2(\nu)$ îi corespunde lui Gamma$\left(\frac{\nu}{2}, \frac{1}{2}\right) \Rightarrow$ face parte din familia exponențială