

PROIECT STATISTICA 2024

ION ADINA 312

GAMAN RAZVAN-ANDREI 311

BOLOTĂ SEBASTIAN-IOAN 312

ANGHEL ANA MARIA 312 (lider echipa)

Exercitiul I)

$$\begin{aligned} 1) \quad & P(Z_n \leq x) = P\left(\sqrt{n}\left(\bar{x}_n - \mu\right) \leq \frac{x}{\sqrt{n}}\right) \\ & = P\left(\sqrt{n}\left(\bar{x}_n - \mu\right) \leq \frac{x}{\sqrt{n}}\right) = P\left(\sqrt{n}\left(\bar{x}_n - \mu\right) \leq T_x\right) = \\ & = P\left(\bar{x}_n - \mu \leq \frac{T_x}{\sqrt{n}}\right) = P\left(\bar{x}_n \leq \frac{T_x + \sqrt{n}\mu}{\sqrt{n}}\right) = \end{aligned}$$

$\boxed{\text{Stim } x_1, \dots, x_n \text{ esantion - } x_1, \dots, x_n \text{ v.a. iid.}}$

$$\bar{x}_n = \frac{1}{n} \sum_{i=1}^n x_i \quad ; n = \text{mărime esantion}$$

$$\begin{aligned} & = P\left(\frac{1}{n} \sum_{i=1}^n x_i \leq \frac{T_x + \sqrt{n}\mu}{\sqrt{n}}\right) = P\left(\sum_{i=1}^n x_i \leq \sqrt{n}\left(\frac{T_x + \sqrt{n}\mu}{\sqrt{n}}\right)\right) = \\ & = P\left(\sum_{i=1}^n x_i \leq \sqrt{n}(T_x + \sqrt{n}\mu)\right) = F_{\sum_{i=1}^n x_i}\left(\sqrt{n}(T_x + \sqrt{n}\mu)\right) \end{aligned}$$

$\text{Stim că } F_X(x) = \int_{-\infty}^x f(t) dt ; X \sim f(t) \text{ fct. densitate pt } X.$

$$\text{La măs } F_{\sum_{i=1}^n x_i}\left(\sqrt{n}(T_x + \sqrt{n}\mu)\right) = \int_{-\infty}^{\sqrt{n}(T_x + \sqrt{n}\mu)} f(t) dt,$$

, unde $\sum_{i=1}^n x_i \sim f(t)$ fct. densitate pentru $\sum_{i=1}^n x_i$.

- Sumă de liniari este liniară,
- Sumă de $x_i \sim \text{Bin}(n, p) \Rightarrow x_n \sim \text{Bin}(n, p) \Rightarrow \sum_{i=1}^n x_i \sim \text{Bin}\left(\sum_{i=1}^n n, p\right)$

Atunci la măs: $\sum_{i=1}^n x_i \sim f(x, \sum_{i=1}^n n, p)$ fct. densitate de masă

$$f(x, \sum_{i=1}^n n, p) = \binom{\sum_{i=1}^n n}{x} \cdot p^x \cdot (1-p)^{\sum_{i=1}^n n - x}$$

$$\text{Deci } \sum_{-\infty}^{\sqrt{n}(T_x + \sqrt{n}\mu)} \left(C_{\sum_{i=1}^n n}^x \cdot p^x \cdot (1-p)^{\sum_{i=1}^n n - x} \right)$$

- Sumă de v.a. Poisson este Poisson.

$$x_1 \sim \text{Poisson}(\lambda_1), \quad x_m \sim \text{Poisson}(\lambda_m)$$

$$x_1 \sim \text{Poisson}(\lambda_1), \dots, x_m \sim \text{Poisson}(\lambda_m) \sim f(x, \lambda)$$

$$\sum_{i=1}^m x_i \sim \text{Poisson}(\lambda_1 + \lambda_2 + \dots + \lambda_m) \sim f(x, \lambda)$$

$$f(x, \lambda = \lambda_1 + \dots + \lambda_m) = \frac{(\lambda_1 + \dots + \lambda_m)^x e^{-(\lambda_1 + \dots + \lambda_m)}}{x!}$$

$$\text{Deci } \sum_{-\infty}^{\infty} \left(\frac{(\lambda_1 + \dots + \lambda_m)^x e^{-(\lambda_1 + \dots + \lambda_m)}}{x!} \right)$$

• Uniforma $U[a, b]$

$$f(x) = \frac{1}{b-a}$$

• Normală $N(\mu, \sigma^2)$

$$f(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2} \left(\frac{x-\mu}{\sigma} \right)^2}$$

• Exponentială $\text{Exp}(\lambda)$

$$f(x) = \lambda e^{-\lambda x}$$

• Binomială $B(n, p)$

$$f(x) = \binom{n}{x} p^x (1-p)^{n-x}$$

• Poisson(λ)

$$f(x) = \frac{\lambda^x e^{-\lambda}}{x!}$$

• Geometrică Geom(p)

$$f(x) = p(1-p)^{x-1}$$

• Gamma (a, b)

$$f(x) = \frac{b^a}{\Gamma(a)} \cdot x^{a-1} \cdot e^{-bx}$$

• Beta (a, b)

$$f(x) = \frac{x^{a-1} (1-x)^{b-1}}{B(a, b)}$$

• Bernoulli ~~B(p)~~ $B(p)$

$$f(x) = \begin{cases} p & x=1 \\ 1-p & x=0 \end{cases}$$

#Penru Binomiala:

#Setează parametrii

$n < 5$

```

p<-0.5
x<-9
#Calculează media și deviația standard
media<-n * p
sigma<-sqrt(n*p*(1 - p))
#Calculează α
alfa<-sigma*x/sqrt(n)+media
#Calculează probabilitatea  $P(\prod \sum (k/n) * p^k * (1-p)^{n-k})$ 
probabilitate1<-sum((0:alfa)/n * p^(0:alfa) * (1 - p)^(n - (0:alfa))) #(se modifica
rata castigului, adica k, de la o v.a la alta)
#Afisează rezultatul
cat("Probabilitatea  $\sum ((k/n) * p^k * (1-p)^{n-k})$ :", probabilitate1, "\n")

```

```

#Geometrica
p_geom <- 0.5
x_geom <- 4
n_geom <- 5
# Calculează media și deviația standard pentru Geometrica
media_geom <- 1 / p_geom
sigma_geom <- sqrt((1 - p_geom) / (p_geom^2))
alfa_geom <- sigma_geom * x_geom / sqrt(n_geom) + media_geom
# Funcția pentru calculul probabilității individuale a distribuției geometrice
Geom2 <- function(p_geom, sigma, x, alfa, media) {
  return (sum((1 - p_geom)^(((sigma * x) / (0:alfa) + media) - 1) * p_geom))
}
cat(Geom2(p_geom, sigma, x, alfa, media))

```

```

#Pentru Poisson:
lambda<-3
x<-5
n<-7
#Calculează media și deviația standard pentru Poisson(lambda)
media_poisson<-lambda
sigma_poisson<-sqrt(lambda)
#Calculează α pentru Poisson(lambda)
alfa_poisson<-(sigma_poisson*x)/sqrt(n)+media_poisson
# Calculează probabilitatea  $P(\prod \sum (e^{-\lambda} * \lambda^k / k!))$ 
Poissonfff<-function(lambda,alfa_poisson){

```

```

return
(sum((exp(-lambda)*lambda^(0:alfa_poisson))/factorial(0:alfa_poisson)))
}
#Afișează rezultatul pentru Poisson(lambda)
cat("Probabilitatea  $\Sigma (e^{-\lambda} * \lambda^x / x!)$ :", Poissonfff(lambda,alfa_poisson), "\n")

```

#Uniforma caz discret:

```

x_vect <- 1:6 # valorile x din v.a.
probabilit <- 1/6
n <- 8      # câte v.a. avem
media_unif_dis <- 1/2 # media pentru o v.a.
varianta_unif_dis <- (n^2 - 1)/12 # varianța pentru o v.a.
#alfa <- (sqrt(varianta_unif_dis) * x_vect) / sqrt(n) + media_unif_dis
SumaDis<-function(x_vect,n, varianta_unif_dis, media_unif_dis){

  a<-sum(((sqrt(varianta_unif_dis) * x_vect[1:6]) / sqrt(n) +
media_unif_dis/n)- floor(((sqrt(varianta_unif_dis) * x_vect[1:6]) / sqrt(n) +
media_unif_dis -1))/n))/ 6
  return (a)

}
cat(SumaDis(x_vect,n, varianta_unif_dis, media_unif_dis))

```

#Pentru Uniforma pe caz continuu:

```

a <- 0
b <- 1
x <- 1/17
n <- 8
media <- (a + b) / 2
varianta <- ((b - a)^2) / 12
alfa <- (sqrt(varianta) * x) / sqrt(n) + media
# Funcția pentru densitatea de probabilitate a variabilei aleatoare uniforme
fCont <- function(x, a, b) {
  ifelse(x >= a & x <= b, 1 / (b - a), 0)
}
# Calculează integrala funcției f în intervalul [a, alfa]
i <- integrate(fCont, lower = a, upper = alfa, a = a, b = b)$value
cat("Integrala funcției f în intervalul [a, alfa]:", i, "\n")

```

```

#Pentru Exponentiala:
a <- 0
x <- 8
n <- 4
lambda <- 4
media_exp <- 1 / lambda
varianta_exp <- 1 / lambda^2
alfa <- (sqrt(varianta_exp) * x) / sqrt(n) + media_exp
# Modificare: Am ajustat funcția pentru a primi vectori și am adăugat 'lambda' ca argument
f2 <- function(x, lambda) {
  return(exp(-lambda * x) * lambda)
}
# Modificare: Am adăugat argumentul 'lambda' în apelul funcției integrate
ExpFunc<- function(f2,a ,alfa, lambda){
  i2<- integrate(f2, lower = a, upper = alfa, lambda = lambda)
  return (i2)
}
cat("Rezultatul integraliei:", (ExpFunc(f2,a ,alfa, lambda))$value, "\n")

```

```

#Pentru Gamma:
# Parametrii
alfa <- 3
beta <- 4
x <- 5
n <- 7
media_gamma <- alfa / beta
varianta_gamma <- alfa / beta^2
omega <- sqrt(varianta_gamma) * x / sqrt(n) + media_gamma
# Funcția de densitate de probabilitate pentru distribuția gamma
fgamma <- function(alfa, beta, x) {
  return ((beta^alfa) / gamma(alfa) * x^(alfa - 1) * exp(-beta * x))
}
# Calculul integralei
i3 <- integrate(fgamma, lower = 0, upper = omega, alfa = alfa, beta = beta)$value

```

```

cat("Rezultatul integrală:", i3, "\n")

#Pentru Beta:
alfa <- 3
beta <- 4
x <- 5
n <- 7
media_beta <- alfa / (beta + alfa)
varianta_beta <- (alfa * beta) / (((alfa + beta)^2) * (alfa + beta + 1))
omega <- sqrt(varianta_beta) * x / sqrt(n) + media_beta
# Funcția Beta
Beta <- function(alfa, beta) {
  return (factorial(alfa - 1) * factorial(beta - 1) / factorial(alfa + beta - 1))
}
# Funcția densitate de probabilitate pentru distribuția Beta
funcDensitate <- function(x, alfa, beta) {
  return ((1 / Beta(alfa, beta)) * (x^(alfa - 1)) * ((1 - x)^(beta - 1)))
}
# Calculul integralei
i4 <- integrate(funcDensitate, lower = 0, upper = omega, alfa = alfa, beta =
beta)$value
cat("Rezultatul integrală pentru distribuția Beta:", i4, "\n")

```

```

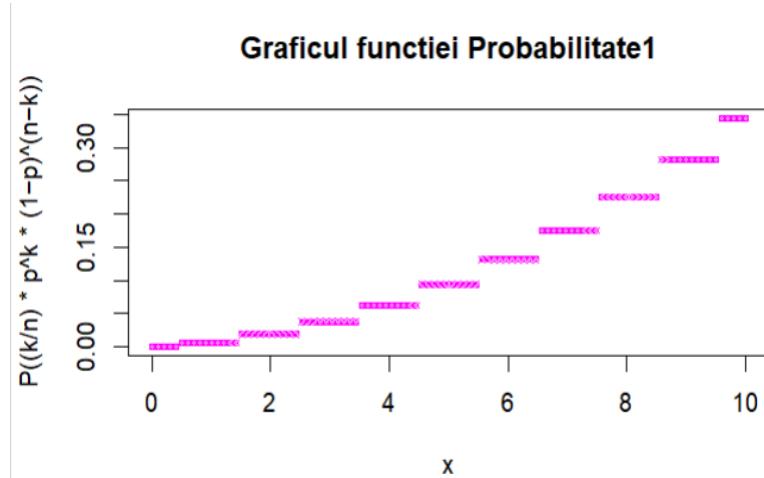
#Ex I 2)
#18.01.2024 incercare 2 (corect)
#Binomiala
# Setează valorile
n <- 5
p <- 0.5
x <- 9
# Calculează media și deviația standard
media <- n * p
sigma <- sqrt(n * p * (1 - p))
# Calculează α
alfa <- sigma * x / sqrt(n) + media
# Funcția pentru calculul probabilității
Probabilitate1 <- function(k, n, p) {

```

```

    return (sum((0:k) / n * p^(0:k) * (1 - p)^(n - (0:k))))
}
# Intervalul de valori pentru x
x_values <- seq(0, 10, length.out = 100)
# Calculul probabilității pentru fiecare valoare a lui x
y_values <- sapply(x_values, function(xi) Probabilitate1(round(xi), n, p))
# Desenarea graficului cu marcaje punctate
plot(x_values, y_values, col = "magenta", type = "p", pch = 13, cex = 0.5,
      xlab = "x", ylab = "P((k/n) * p^k * (1-p)^(n-k))",
      main = "Graficul functiei Probabilitate1")

```

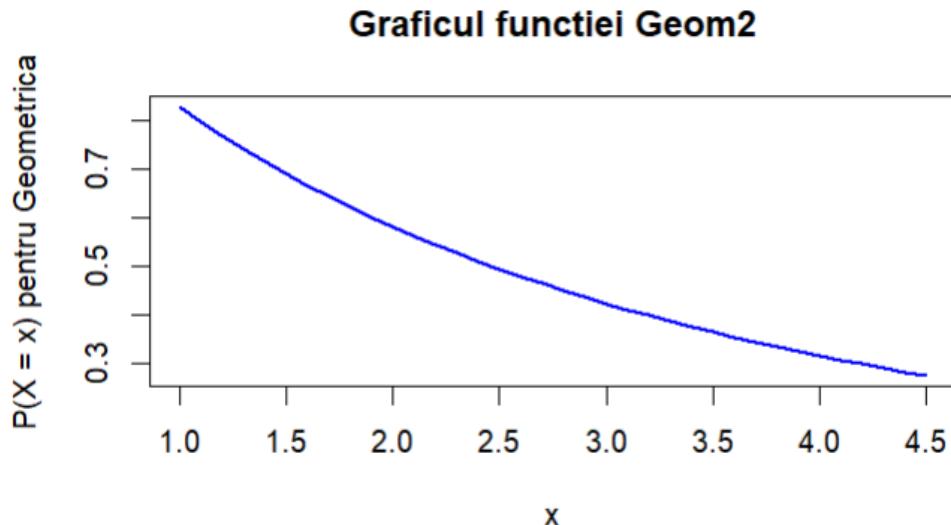


```

#Geometrica
p_geom <- 0.5
x_geom <- 4
n_geom <- 5
# Calculează media și deviația standard pentru Geometrica
media_geom <- 1 / p_geom
sigma_geom <- sqrt((1 - p_geom) / (p_geom^2))
alfa_geom <- sigma_geom * x_geom / sqrt(n_geom) + media_geom
# Funcția pentru calculul probabilității individuale a distribuției geometrice
Geom2 <- function(p, sigma, x, alfa, media) {
  return (sum((1 - p)^(((sigma * x) / (0:alfa) + media) - 1) * p))
}
cat(Geom2(p, sigma, x, alfa, media))
# Intervalul de valori pentru x în distribuția geometrică
x_values_geom <- seq(1, alfa_geom, by = 0.1)
# Calculul probabilității pentru fiecare valoare a lui x în distribuția geometrică
y_values_geom <- sapply(x_values_geom, function(xi) Geom2(p_geom,
  sigma_geom, xi, round(alfa_geom), media_geom))
# Desenarea graficului pentru distribuția geometrică

```

```
plot(x_values_geom, y_values_geom, col = "blue", type = "l", lwd = 2,
  xlab = "x", ylab = "P(X = x) pentru Geometrica",
  main = "Graficul functiei Geom2")
```

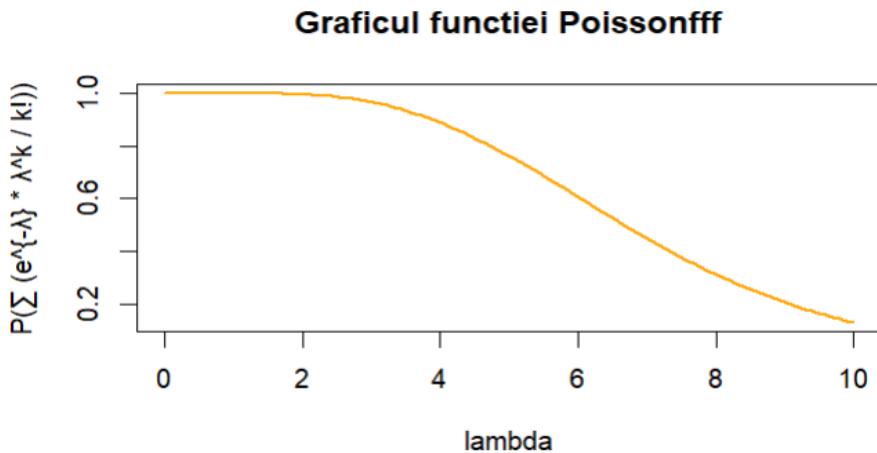


```
#Poisson:
lambda<-3
x<-5
n<-7
#Calculează media și deviația standard pentru Poisson(lambda)
media_poisson<-lambda
sigma_poisson<-sqrt(lambda)
#Calculează α pentru Poisson(lambda)
alfa_poisson<-(sigma_poisson*x)/sqrt(n)+media_poisson
# Calculează probabilitatea  $P(\prod \sum (e^{-\lambda} * \lambda^k / k!))$ 
Poissonfff<-function(lambda,alfa_poisson){
  return
  (sum((exp(-lambda)*lambda^(0:alfa_poisson))/factorial(0:alfa_poisson)))
}
#Afisează rezultatul pentru Poisson(lambda)
cat("Probabilitatea  $\sum (e^{-\lambda} * \lambda^x / x!):$ ", Poissonfff(lambda,alfa_poisson), "\n")
# Intervalul de valori pentru lambda
lambda_values <- seq(0, 10, length.out = 100)
# Calculul probabilității pentru fiecare valoare a lui lambda
y_values_poisson <- sapply(lambda_values, function(l) Poissonfff(l,
  alfa_poisson))
# Desenarea graficului
plot(lambda_values, y_values_poisson, col = "orange", type = "l", lwd = 2,
```

```

xlab = "lambda", ylab = "P( $\sum (e^{-\lambda} * \lambda^k / k!)$ )",
main = "Graficul functiei Poissonfff")

```



#Uniforma caz discret:

```

x <- 4
n <- 8
media_unif_dis <- 1/2
varianta_unif_dis <- (n^2 - 1)/12
#alfa <- ((sqrt(varianta_unif_dis) * x) / sqrt(n) + media_unif_dis)
SumaDis <- function(x, n, varianta_unif_dis, media_unif_dis) {
  a <- ((sqrt(varianta_unif_dis) * x) / sqrt(n) + media_unif_dis) *
    sum((floor(((sqrt(varianta_unif_dis) * x) / sqrt(n) + media_unif_dis) / n) -
        floor((((sqrt(varianta_unif_dis) * x) / sqrt(n) + media_unif_dis) / n) -
        * x) / sqrt(n) + media_unif_dis) - 1)) / n)) / 6)
  return(a)
}
cat(SumaDis(x, n, varianta_unif_dis, media_unif_dis))
x_values <- seq(1, ((sqrt(varianta_unif_dis) * x) / sqrt(n) + media_unif_dis),
length.out = 1000)
result <- sapply(x_values, function(x) SumaDis(x, n, varianta_unif_dis,
media_unif_dis))
plot(x_values, result, type = "p", col = "red", lwd = 0.5,
      main = "Graficul functiei SumaDis",
      xlab = "x", ylab = "Valoare",
      ylim = c(0, 1))

```

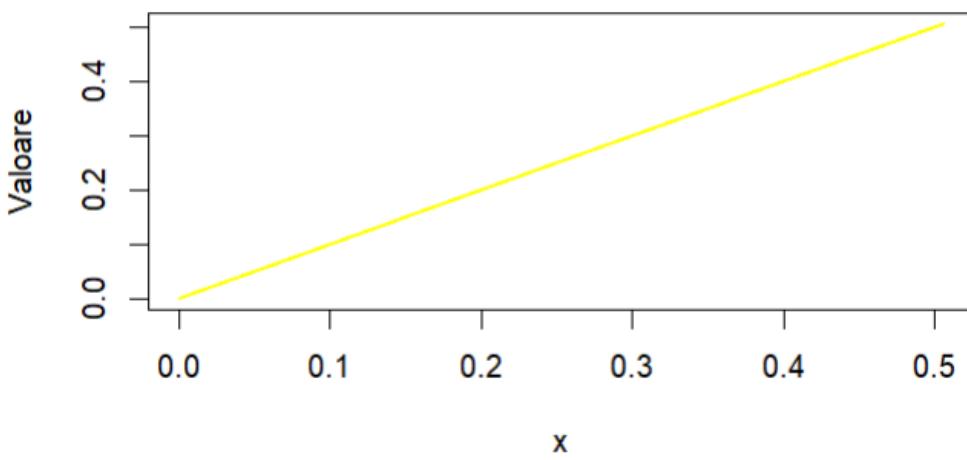
Uniforma pe caz continuu:

```

a <- 0
b <- 1
x <- 1/17
n <- 8
media <- (a + b) / 2
varianta <- ((b - a)^2) / 12
alfa <- (sqrt(varianta) * x) / sqrt(n) + media
# Funcția pentru densitatea de probabilitate a variabilei aleatoare uniforme
fCont <- function(x, a, b) {
  ifelse(x >= a & x <= b, 1 / (b - a), 0)
}
# Calculează integrala funcției f în intervalul [a, alfa]
ContFunc<- function(fCont,a,alfa,b){
  a<- integrate(fCont, lower = a, upper = alfa, a = a, b = b)$value
  return (a)
}
cat("Integrala funcției f în intervalul [a, alfa]:", ContFunc(fCont,a,alfa,b), "\n")
# Calculul valorii funcției într-un set de puncte între a și alfa
x_values <- seq(a, alfa, length.out = 1000)
y_values <- sapply(x_values, function(x) ContFunc(fCont, a, x, b))
# Reprezentarea graficului folosind plot
plot(x_values, y_values, type = "l", col = "yellow", lwd = 2,
      main = "Graficul funcției ContFunc",
      xlab = "x", ylab = "Valoare")

```

Graficul funcției ContFunc



#Exponentiala:

a <- 0

```

x <- 8
n <- 4
lambda <- 4
media_exp <- 1 / lambda
varianta_exp <- 1 / lambda^2
alfa <- (sqrt(varianta_exp) * x) / sqrt(n) + media_exp
# Modificare: Am ajustat funcția pentru a primi vectori și am adăugat 'lambda' ca argument
f2 <- function(x, lambda) {
  return(exp(-lambda * x) * lambda)
}
# Modificare: Am adăugat argumentul 'lambda' în apelul funcției integrate
ExpFunc<- function(a ,alfa, lambda){
  i2<- integrate(f2, lower = a, upper = alfa, lambda = lambda)
  return (i2)
}
cat("Rezultatul integraliei:", (ExpFunc(a ,alfa, lambda))$value, "\n")
# Calculul valorii funcției pentru diferite valori ale lui x
x_values <- seq(a, alfa, length.out = 1000)
result <- sapply(x_values, function(x) ExpFunc(a, x, lambda))
# Reprezentarea graficului folosind plot
plot(x_values, result$values, type = "l", col = "magenta", lwd = 2,
      main = "Graficul funcției ExpFunc",
      xlab = "x", ylab = "Valoare")

```

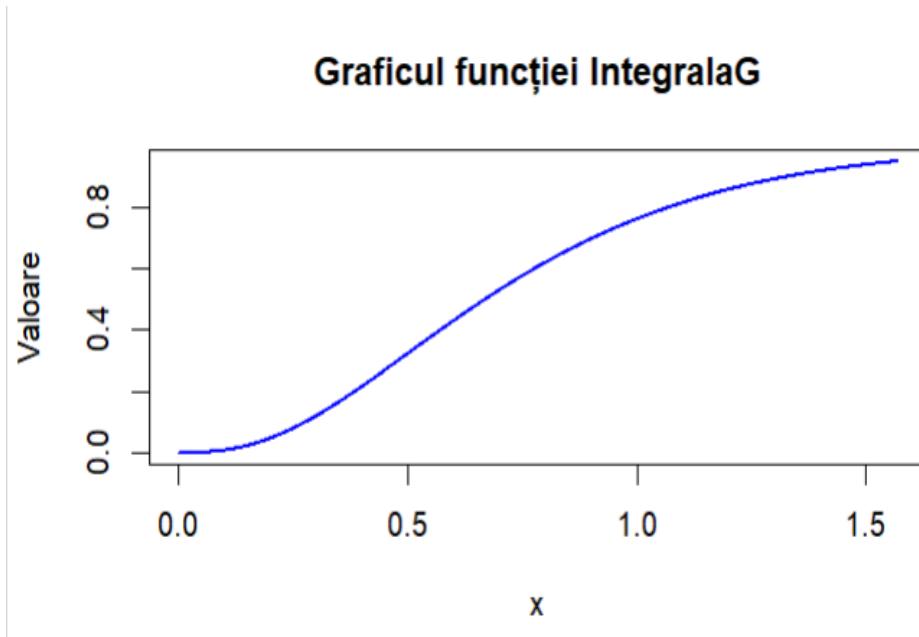
```

#Pentru Gamma:
# Parametrii
alfa <- 3
beta <- 4
x <- 5
n <- 7
media_gamma <- alfa / beta
varianta_gamma <- alfa / beta^2
omega <- sqrt(varianta_gamma) * x / sqrt(n) + media_gamma
# Funcția de densitate de probabilitate pentru distribuția gamma
fgamma <- function(alfa, beta, x) {
  return ((beta^alfa) / gamma(alfa) * x^(alfa - 1) * exp(-beta * x))
}
```

```

# Calculul integralei
IntegralaG<- function(fgamma,omega, alfa, beta){
  i3 <- integrate(fgamma, lower = 0, upper = omega, alfa = alfa, beta =
beta)$value
  return (i3)
}
cat("Rezultatul integrală:", IntegralaG(fgamma,omega, alfa, beta), "\n")
x_values <- seq(0, omega, length.out = 1000)
result <- sapply(x_values, function(x) IntegralaG(fgamma, x, alfa, beta))
# Reprezentarea graficului folosind plot
plot(x_values, result, type = "l", col = "blue", lwd = 2,
      main = "Graficul funcției IntegralaG",
      xlab = "x", ylab = "Valoare")

```



```

#Pentru Beta:
alfa <- 3
beta <- 4
x <- 5
n <- 7
media_beta <- alfa / (beta + alfa)
varianta_beta <- (alfa * beta) / (((alfa + beta)^2) * (alfa + beta + 1))
omega <- sqrt(varianta_beta) * x / sqrt(n) + media_beta
# Funcția Beta
Beta <- function(alfa, beta) {
  return (factorial(alfa - 1) * factorial(beta - 1) / factorial(alfa + beta - 1))
}

```

```

# Funcția densitate de probabilitate pentru distribuția Beta
funcDensitate <- function(x, alfa, beta) {
  return ((1 / Beta(alfa, beta)) * (x^(alfa - 1)) * ((1 - x)^(beta - 1)))
}

# Calculul integralei
IntegralaB<- function(funcDensitate, omega, alfa, beta){
  i4 <- integrate(funcDensitate, lower = 0, upper = omega, alfa = alfa, beta =
beta)$value
  return (i4)
}
cat("Rezultatul integrală pentru distribuția Beta:", IntegralaB(funcDensitate,
omega, alfa, beta), "\n")
x_values <- seq(0, omega, length.out = 1000)
result <- sapply(x_values, function(x) IntegralaB(funcDensitate, x, alfa, beta))
# Reprezentarea graficului folosind plot
plot(x_values, result, type = "l", col = "green", lwd = 2,
      main = "Graficul funcției IntegralaB",
      xlab = "x", ylab = "Valoare")

```

#Ex I 3)

```

#definefim fct lui Laplace (fct de repartitie a normalei standard)
Laplace <- function(t){
  a <- 1 / (sqrt(2 * pi))
  g <- function(x) {
    return(exp(-(x^2) / 2))
  }
  b <- integrate(g, lower = -Inf, upper = t)$value
  return(a * b)
}
absolut1 <- function(t){
  abs(0.175 - Laplace(t))
}
absolut2 <- function(t){
  abs(0.2181381 - Laplace(t))
}
absolut3 <- function(t){
  abs(0.9664915 - Laplace(t))
}
absolut4 <- function(t){
  abs(0 - Laplace(t))
}
```

```

}

absolut5 <- function(t){
  abs(0.5060037 - Laplace(t))
}

absolut6 <- function(t){
  abs(0.9932621 - Laplace(t))
}

absolut7 <- function(t){
  abs(0.9491704 - Laplace(t))
}

absolut8 <- function(t){
  abs(0.9670584 - Laplace(t))
}

# Utilizarea optimize pentru a găsi supremumul
rezultat_optimize <- optimize(absolut1, interval = c(-100, 100), maximum =
FALSE)
cat("Supremumul este:", rezultat_optimize$minimum, "\n")
rezultat_optimize <- optimize(absolut2, interval = c(-100, 100), maximum =
FALSE)
cat("Supremumul este:", rezultat_optimize$minimum, "\n")
rezultat_optimize <- optimize(absolut3, interval = c(-100, 100), maximum =
FALSE)
cat("Supremumul este:", rezultat_optimize$minimum, "\n")
rezultat_optimize <- optimize(absolut4, interval = c(-100, 100), maximum =
FALSE)
cat("Supremumul este:", rezultat_optimize$minimum, "\n")
rezultat_optimize <- optimize(absolut5, interval = c(-100, 100), maximum =
FALSE)
cat("Supremumul este:", rezultat_optimize$minimum, "\n")
rezultat_optimize <- optimize(absolut6, interval = c(-100, 100), maximum =
FALSE)
cat("Supremumul este:", rezultat_optimize$minimum, "\n")
rezultat_optimize <- optimize(absolut7, interval = c(-100, 100), maximum =
FALSE)
cat("Supremumul este:", rezultat_optimize$minimum, "\n")
rezultat_optimize <- optimize(absolut8, interval = c(-100, 100), maximum =
FALSE)
cat("Supremumul este:", rezultat_optimize$minimum, "\n")

```

#Ex I 4)

```
CalcMedia <- function(x) {
  if (grepl("Bin", x)) {
    # Definirea expresiei regulate pentru identificarea parametrilor
    pattern <- "^Bin\\((\\d+),(\\d+.?\\d*)\\)$"
    # Verificarea potrivirii cu expresia regulată
    match_result <- regmatches(x, regexec(pattern, x))
    # Extrage parametrii daca exista o potrivire
    if (!is.null(match_result[[1]])) {
      n <- as.numeric(match_result[[1]][2])
      p <- as.numeric(match_result[[1]][3])
      # Verificare pentru parametri valizi
      if (!is.na(n) && !is.na(p) && n > 0 && p >= 0 && p <= 1) {
        cat("n =", n, "\n")
        cat("p =", p, "\n")
        # Calculul și afișarea mediei
        media <- n * p
        cat("media =", media, "\n")
        varianta <- n*p*(1-p)
        cat("varianta", varianta)
        # Returnarea mediei
        return(media)
      } else {
        cat("Parametrii introdusi nu sunt valizi pentru distributia binomiala.\n")
        return(NULL)
      }
    } else {
      cat("Formatul introdus nu corespunde cu 'Bin(n,p)'.\n")
      return(NULL)
    }
  } else if (grepl("Geom", x)){
    # Definirea expresiei regulate pentru identificarea parametrilor
    pattern <- "^Geom\\((\\d+\\.?\\d*)\\)$" # Ajustare a expresiei regulate
    # Verificarea potrivirii cu expresia regulată
    match_result <- regmatches(x, regexec(pattern, x))
    # Extrage parametrii daca exista o potrivire
    if (!is.null(match_result[[1]])) {
      p <- as.numeric(match_result[[1]][2])
      # Verificare pentru parametri valizi
      if (!is.na(p) && p > 0 && p <= 1) {
```

```

cat("p =", p, "\n")
# Calculul și afișarea mediei
media <- 1/p
cat("media =", media, "\n")
varianta <- (1-p) / (p^2)
cat("varianta", varianta, "\n")
# Returnarea mediei
return(media)
} else {
  cat("Parametrii introdusi nu sunt valizi pentru distributia geometrica.\n")
  return(NULL)
}
} else {
  cat("Formatul introdus nu corespunde cu 'Geom(p)'.\n")
  return(NULL)
}
} else if (grepl("Pois", x)){
  # Distributia Poisson
  pattern <- "^Pois\\((\\d+\\.?\\d*)\\)$"
  match_result <- regmatches(x, regexec(pattern, x))
  if (!is.null(match_result[[1]])) {
    lambda <- as.numeric(match_result[[1]][2])
    if (!is.na(lambda) && lambda > 0) {
      cat("lambda =", lambda, "\n")
      media <- lambda
      cat("media =", media, "\n")
      varianta <- lambda
      cat("varianta", varianta, "\n")
      return(media)
    } else {
      cat("Parametrii introdusi nu sunt valizi pentru distributia Poisson.\n")
      return(NULL)
    }
  } else {
    cat("Formatul introdus nu corespunde cu 'Pois(lambda)'.\n")
    return(NULL)
  }
} else if (grepl("UnifD", x)){
  # Uniforma pe caz discret
  pattern <- "^UnifD\\((\\d+),(\\d+)\\)$"

```

```

match_result <- regmatches(x, regexec(pattern, x))
if (!is.null(match_result[[1]])) {
  a <- as.numeric(match_result[[1]][2])
  b <- as.numeric(match_result[[1]][3])
  if (!is.na(a) && !is.na(b) && a < b) {
    cat("a =", a, "\n")
    cat("b =", b, "\n")
    media <- (a + b) / 2
    cat("media =", media, "\n")
    varianta <- ((b - a + 1)^2 - 1) / 12
    cat("varianta", varianta, "\n")
    return(media)
  } else {
    cat("Parametrii introdusi nu sunt valizi pentru distributia uniforma pe caz
discret.\n")
    return(NULL)
  }
} else {
  cat("Formatul introdus nu corespunde cu 'UnifD(a,b)'.\n")
  return(NULL)
}
} else if (grepl("UnifC", x)){
  # Uniforma pe caz continuu
  pattern <- "^UnifC\\((\\d+\\.?\\d*),(\\d+\\.?\\d*)\\)$"
  match_result <- regmatches(x, regexec(pattern, x))
  if (!is.null(match_result[[1]])) {
    a <- as.numeric(match_result[[1]][2])
    b <- as.numeric(match_result[[1]][3])
    if (!is.na(a) && !is.na(b) && a < b) {
      cat("a =", a, "\n")
      cat("b =", b, "\n")
      media <- (a + b) / 2
      cat("media =", media, "\n")
      varianta <- (b - a)^2 / 12
      cat("varianta", varianta, "\n")
      return(media)
    } else {
      cat("Parametrii introdusi nu sunt valizi pentru distributia uniforma pe caz
continuu.\n")
      return(NULL)
    }
  }
}

```

```

    }
} else {
  cat("Formatul introdus nu corespunde cu 'UnifC(a,b)'.\n")
  return(NULL)
}
} else if (grepl("Exp", x)){
  # Exponentiala
  pattern <- "^Exp\\((\\d+\\.?\\d*)\\)$"
  match_result <- regmatches(x, regexexec(pattern, x))
  if (!is.null(match_result[[1]])) {
    lambda <- as.numeric(match_result[[1]][2])
    if (!is.na(lambda) && lambda > 0) {
      cat("lambda =", lambda, "\n")
      media <- 1 / lambda
      cat("media =", media, "\n")
      varianta <- 1 / (lambda^2)
      cat("varianta", varianta, "\n")
      return(media)
    } else {
      cat("Parametrii introdusi nu sunt valizi pentru distributia exponentiala.\n")
      return(NULL)
    }
  } else {
    cat("Formatul introdus nu corespunde cu 'Exp(lambda)'.\n")
    return(NULL)
  }
} else if (grepl("Gamma", x)){
  # Gamma
  pattern <- "^Gamma\\((\\d+\\.?\\d*),(\\d+\\.?\\d*)\\)$"
  match_result <- regmatches(x, regexexec(pattern, x))
  if (!is.null(match_result[[1]])) {
    alpha <- as.numeric(match_result[[1]][2])
    beta <- as.numeric(match_result[[1]][3])
    if (!is.na(alpha) && !is.na(beta) && alpha > 0 && beta > 0) {
      cat("alpha =", alpha, "\n")
      cat("beta =", beta, "\n")
      media <- alpha / beta
      cat("media =", media, "\n")
      varianta <- alpha / (beta^2)
      cat("varianta", varianta, "\n")
    }
  }
}

```

```

    return(media)
} else {
  cat("Parametrii introdusi nu sunt valizi pentru distributia gamma.\n")
  return(NULL)
}
} else {
  cat("Formatul introdus nu corespunde cu 'Gamma(alpha,beta).\n")
  return(NULL)
}
} else if (grepl("Beta", x)){
# Beta
pattern <- "^Beta\\((\\d+\\.?\\d*),(\\d+\\.?\\d*)\\)$"
match_result <- regmatches(x, regexec(pattern, x))
if (!is.null(match_result[[1]])) {
  alpha <- as.numeric(match_result[[1]][2])
  beta <- as.numeric(match_result[[1]][3])
  if (!is.na(alpha) && !is.na(beta) && alpha > 0 && beta > 0) {
    cat("alpha =", alpha, "\n")
    cat("beta =", beta, "\n")
    media <- alpha / (alpha + beta)
    cat("media =", media, "\n")
    varianta <- (alpha * beta) / ((alpha + beta)^2 * (alpha + beta + 1))
    cat("varianta", varianta, "\n")
    return(media)
  } else {
    cat("Parametrii introdusi nu sunt valizi pentru distributia beta.\n")
    return(NULL)
  }
} else {
  cat("Formatul introdus nu corespunde cu 'Beta(alpha,beta).\n")
  return(NULL)
}
} else {
  cat("Distributia introdusa nu este recunoscuta.\n")
  return(NULL)
}
}

# Citirea datelor de la tastatura
x <- readline(prompt = "Introduceti distributia: ")
# Apelarea functiei

```

rezultat <- CalcMedia(x)

I 4) Calcul medie

- Pentru fct. continuu $E = \int_0^\infty x \cdot f(x) dx$
(cu fct. densitate)

- Pentru fct. discrete $E = \sum_{i=1}^n x_i \cdot f(x_i)$
(cu fct. de masă)

• La măsă: $\text{Exp}(\frac{1}{3}) \Rightarrow f(x) = \frac{1}{3} e^{-\frac{x}{3}}$

Așa că $E = \int_0^\infty x \cdot \left(\frac{1}{3} e^{-\frac{x}{3}}\right) dx$ în funcție integrată

Calcul varianta (față de medie)

- Pentru fct. continuu $\text{Var} = \int_0^\infty x^2 \cdot f(x) dx - E^2$; $E = \text{medie}$
(cu fct. densitate)

- Pentru fct. discrete $\text{Var} = \sum_{i=1}^n x_i^2 \cdot f(x_i) - E^2$; $E = \text{medie}$.
(cu fct. de masă)

• La măsă: $\text{Exp}(\frac{1}{3}) \Rightarrow \text{Var} = \int_0^\infty x^2 \left(\frac{1}{3} e^{-\frac{x}{3}}\right) dx - E^2$; $E = \text{medie}$

5) momentul central 3 i.e. $E[(x-\mu)^3]$

5) momentul central $E[(x-\mu)^3] = M$

- Pentru fct. continuu $M = \int_0^\infty (x - E)^3 \cdot f(x) dx$; $E = \text{medie v. a.}$

- Pentru fct. discrete: $M = \sum_{i=1}^n |x_i - E|^3 \cdot f(x_i)$; $E = \text{medie v. a.}$

#Varianta 2:

Exponentiala<-function(x){#Parametru 1/3

y<-rep(0,length(x))

y[x>0]<-(-1/3)*exp(-(x[x>0])/3)#Construim suportul densității i.e. valorile fct de densitate pentru valorile x pozitive

return (y)

}

Uniforma<-function(x){#Parametrii 1 si 15

y<-rep(0,length(x))

y[x >= 1 & x <= 15] <- 1 / (15 - 1 + 1)

```

return(y)
}

Normala<-function(x){#Parametru 0 si 1
y<-rep(0,length(x))
y[x] <- (1 / sqrt(2*pi))*exp(-1/2*(x^2))
return(y)
}

Binomiala<-function(x){#Parametru n=16 si 1/8=p
y<-rep(0,length(x))
y[x>0]<-choose(16, x[x>0])*(1/8^(x[x>0]))*(1-1/8)^(16-x[x>0])
return(y)
}

Poisson<-function(x){#Parametru lambda=3
y<-rep(0,length(x))
y[x>0]<-3^(x[x>0])*exp(-3)/ factorial(x[x>0])
return(y)
}

Geometrica<-function(x){#Parametru 1/3=p
y<-rep(0,length(x))
y[x>0]<-1/3*((1-1/3)^(x[x>0]-1))
return(y)
}

Gamma<-function(x){#Parametru a=1, b=2
y<-rep(0,length(x))
y[x>0]<-(2^1/gamma(a))*(x[x>0])^(1-1)*exp(-2*(x[x>0]))
return(y)
}

Beta<-function(x){#Parametru a=1, b=2
y<-rep(0,length(x))
y[x>0]<-(x[x>0]^(1-1)*(1-(x[x>0]))^(2-1))/beta(1,2)
return(y)
}

```

```

#Functie ce calculeaza media, primind fct. de densitate ca parametrul f (facem
ori ca integrala, ori ca suma)
medie<-function(f,densitate=TRUE,n=10000){
  if(densitate){
    integrand<-function(x) return(x*f(x))
    E<-integrate(integrand,lower = -Inf,upper = Inf)$value
  }else{
    x=1:n
    E<-sum(x*f(x),na.rm = TRUE) #na.rm = TRUE este utilizat pentru a elimina
    orice valori lipsă din calcul, dacă există.
  }
  return(E)
}

```

#Functia varianta care primeste drept input functia medie definita mai sus.

Varianta o facem ori cu integrala, ori cu suma.

```

varianta<-function(f,densitate=TRUE,n=10000,E=medie){
  if(densitate){
    integrand2<-function(x) return((x^2)*f(x))
    Var<- integrate(integrand2,lower = -Inf,upper=Inf)$value - (E(f, densitate))^2
  }else{
    x<-1:n
    Var<-sum((x^2)*f(x),na.rm = TRUE) - (E(f,densitate))^2
  }
  return(Var)
}

```

```

media_exponentiala <- medie(Exponentiala)
varianta_exponentiala <- varianta(Exponentiala)
print(media_exponentiala)
print(varianta_exponentiala)

```

#eX I 5)

#folosim funcția pentru medie construită mai sus, la 4).

```

Momentcentral3<-function(f,densitate=TRUE,n=10000,E=medie){
  if(densitate){
    integrand3<-function(x) return(((x-E(f))^3)*f(x))
    MC3<-integrate(integrand3,lower=-Inf,upper=Inf)$value
  }else{

```

```

x<-1:n
MC3<-sum((abs(x-E(f,densitate))^3)*f(x),na.rm=TRUE)
}
return(MC3)
}

```

```

moment_centrat_3 <- Momentcentrat3(Exponentiala)
cat("Momentul centrat al treilea este:", moment_centrat_3, "\n")

```

6) Marginile luate din inegalitate : Skewness = $\frac{E[(x-\mu)^3]}{\sigma^3}$.

- Skewness binomială $\frac{1-\alpha p}{\sqrt{n}(1-p)}$
- Geometrică $\frac{3}{\sqrt{2}}$
- Poisson 2
- Uniformă 0 (este perfect simetrică)
- Exponentială 2
- Gamma $\frac{2}{\sqrt{k}} \quad (\text{Beta}(k, \theta))$
- Beta $\frac{4}{\sqrt{(a+b+1)(a+b)}} \quad (\text{Beta}(a, b))$

Nă vrem $\sup_{\infty} |P(Z_n \leq x) - \bar{P}(x)| \leq \frac{33}{4\sqrt{n}} \cdot \frac{E[(x-\mu)^3]}{\sigma^3}$

La noi $\cdot \exp\left(\frac{1}{3}\right) \Rightarrow \lambda = \frac{1}{3} \Rightarrow \sigma = \frac{1}{\sqrt{\lambda}} = \sqrt{3}$

#Ex 6)

```

#calculam skew pentru variabilele noastre cu parametrii aleși
skew<-rep(0,7)
skew[1]<-2/(5*sqrt(10))#Binomiala(n=20,p=1/10)
skew[2]<-3/sqrt(2)#Geometrica(p=1/2)
skew[3]<-1#Poisson(r=1)
skew[4]<-0#Uniforma(a=1,b=9)
skew[5]<-2#Exponentiala(r=1/3)

```

```

skew[6]<-2/sqrt(0.7)#Gamma(k=0.7,theta=13)
skew[7]<-4/sqrt(15)#Beta(a=1,b=3)

#Creăm un data.frame gol, în care vom pune valorile calculate
n<-c(30,100,1000)
MarginiBE<-data.frame("30"=rep(0,7),
                       "100"=rep(0,7),
                       "1000"=rep(0,7),
                       row.names =
c("Binomiala","Geometrica","Poisson","Uniforma","Exponentiala","Gamma","Beta"))

for (i in 1:7){
  for (j in 1:3){
    MarginiBE[i,j]<-(33/4)*(skew[i]/sqrt(n[j]))
  }
}
print(MarginiBE)

```

```

#Ex 7
discr<-41#numarul de puncte in care vom evalua functia
interval<-seq(-4,4,length.out=discr)

```

```

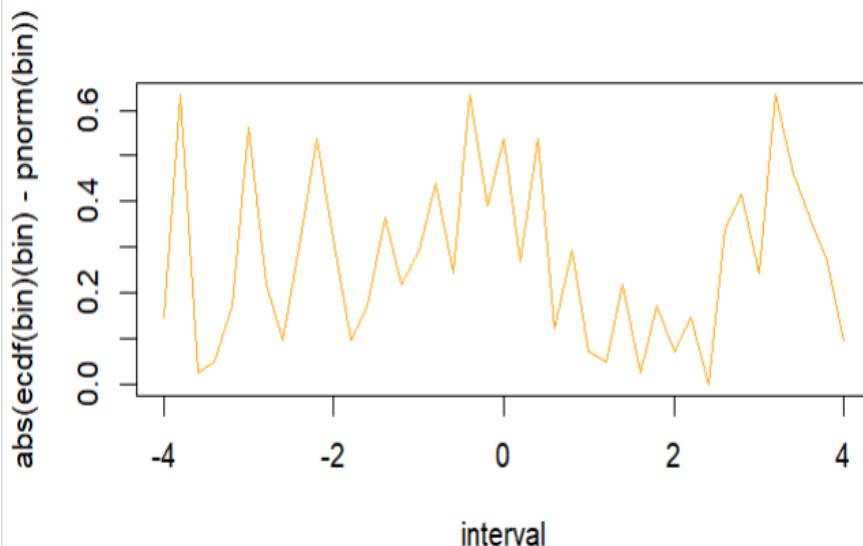
for (i in 1:3){
  bin<-numeric(0)
  geo<-numeric(0)
  pois<-numeric(0)
  unif<-numeric(0)
  exp<-numeric(0)
  gam<-numeric(0)
  bet<-numeric(0)
  for (k in 1:discr){ #generam de 41 de ori esantioane de dimensiune n[i]
    bin<-c(bin,sum((rbinom(n[i],20,1/10)-2)/(2/9)))
    #rbinom(n[i], 20, 1/10) generează un vector de dimensiune n[i] de
    esantioane aleatoare dintr-o distribuție binomială de dimensiune = 20 și prob
    = 1/10.
    #(rbinom(n[i], 20, 1/10) - 2) / (2/9) normalizarea datelor.
  }
}

```

`#sum((rbinom(n[i], 20, 1/10) - 2) / (2/9)).` Rezultatul transformării este apoi suma tuturor acestor eșantioane transformate. Această sumă este ulterior adăugată la vectorul bin.

`#bin <- c(bin, sum((rbinom(n[i], 20, 1/10) - 2) / (2/9))):` În cele din urmă, valoarea rezultată este adăugată la vectorul bin folosind funcția `c()`. Astfel, la fiecare iterație a buclei interioare, un nou punct este adăugat la vectorul bin, reprezentând suma eșantioanelor transformate pentru distribuția binomială la acea iterare specifică.

```
geo<-c(geo,sum((rgeom(n[i],1/2)-2)/2))
pois<-c(pois,sum((rpois(n[i],1)-1)))
unif<-c(unif,sum((sample(1:9,n[i],replace=TRUE,prob=rep(1/9,9))-5)/(20/3)))
exp<-c(exp,sum((rexp(n[i],1/3)-3)/9))
gam<-c(gam,sum((rgamma(n[i],0.7,1/13)-(13*0.7))/0.7*(13^2)))
bet<-c(bet,sum((rbeta(n[i],1,3)-(1/4))/(3/80)))
}
plot(interval,abs(ecdf(bin)(bin)-pnorm(bin)),type="l",lwd="1",col="orange")
```

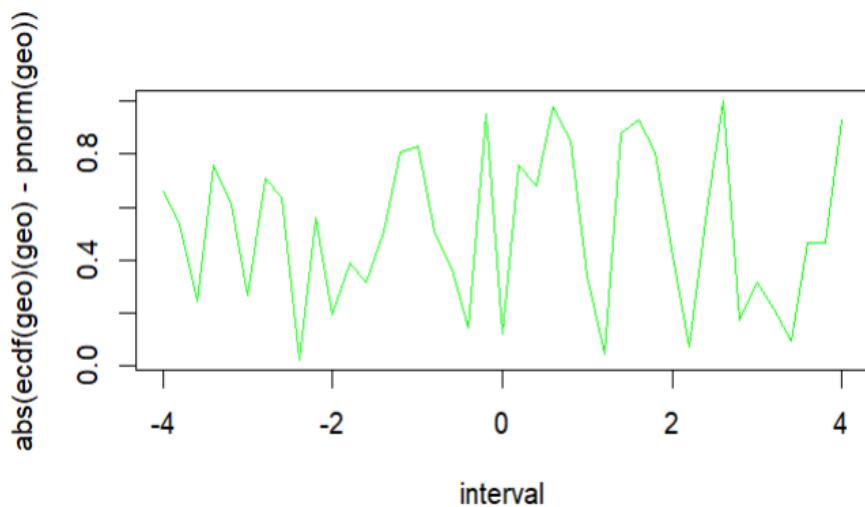


`#abs(ecdf(bin)(bin) - pnorm(bin)).` Această expresie calculează diferența absolută între ECDF și funcția de distribuție normală standard (`pnorm()`) pentru fiecare valoare din vectorul bin (care conține eșantioanele din distribuția binomială).

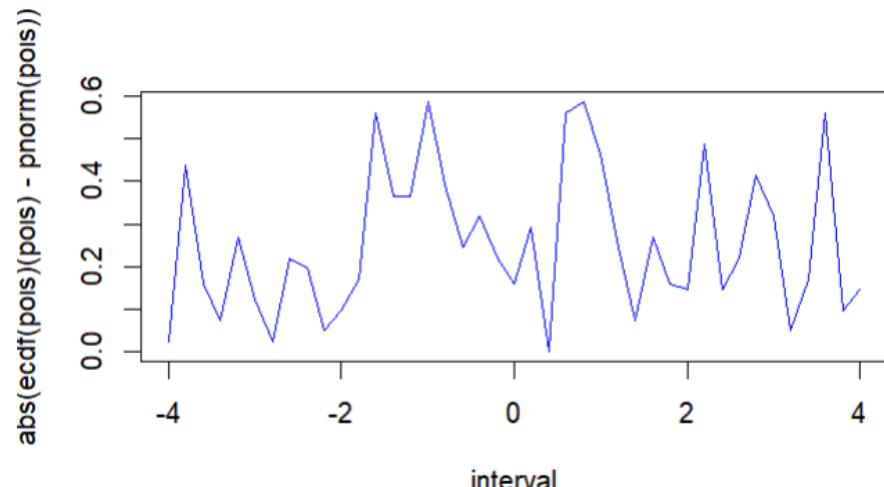
`#ecdf(bin)(bin).` Aici `ecdf(bin)` este o funcție care calculează ECDF pentru vectorul bin. `(bin)` este apoi aplicat pentru a evalua ECDF în punctele din bin.

`#pnorm(bin).` Aceasta calculează valorile pentru funcția de distribuție normală standard pentru vectorul bin.

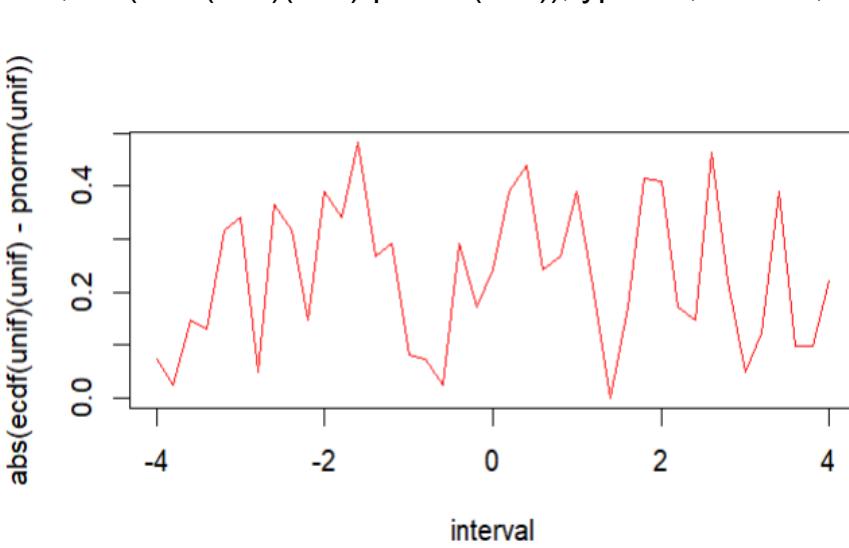
```
plot(interval,abs(ecdf(geo)(geo)-pnorm(geo)),type="l",lwd="1",col="green")
```



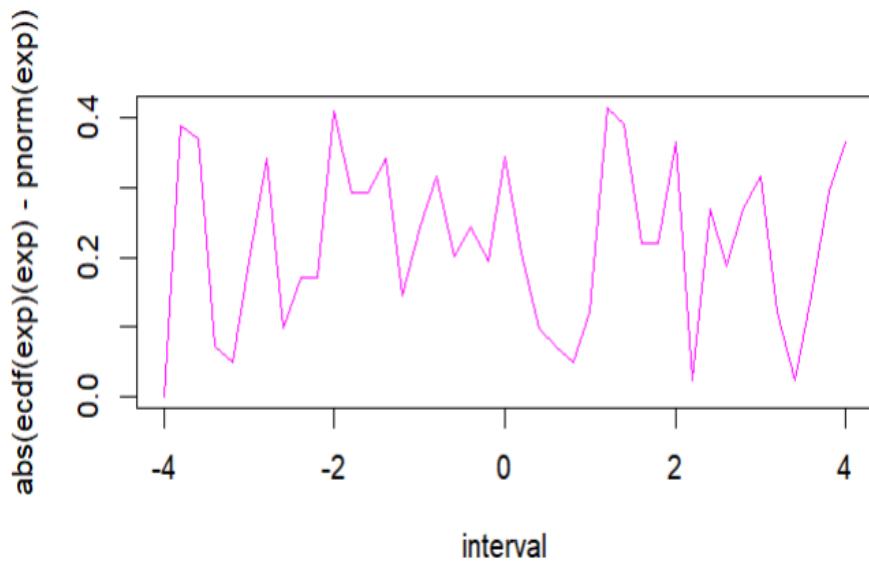
```
plot(interval,abs(ecdf(pois)(pois)-pnorm(pois)),type="l",lwd="1",col="blue")
```



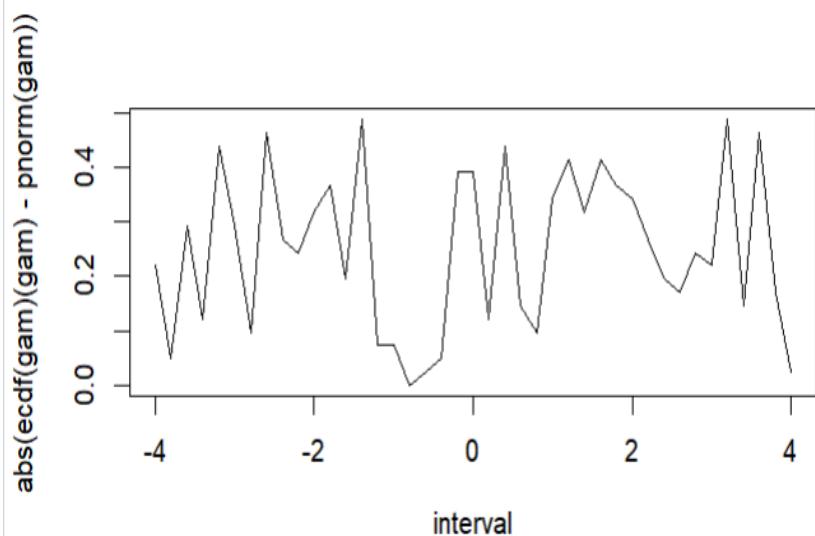
```
plot(interval,abs(ecdf(unif)(unif)-pnorm(unif)),type="l",lwd="1",col="red")
```



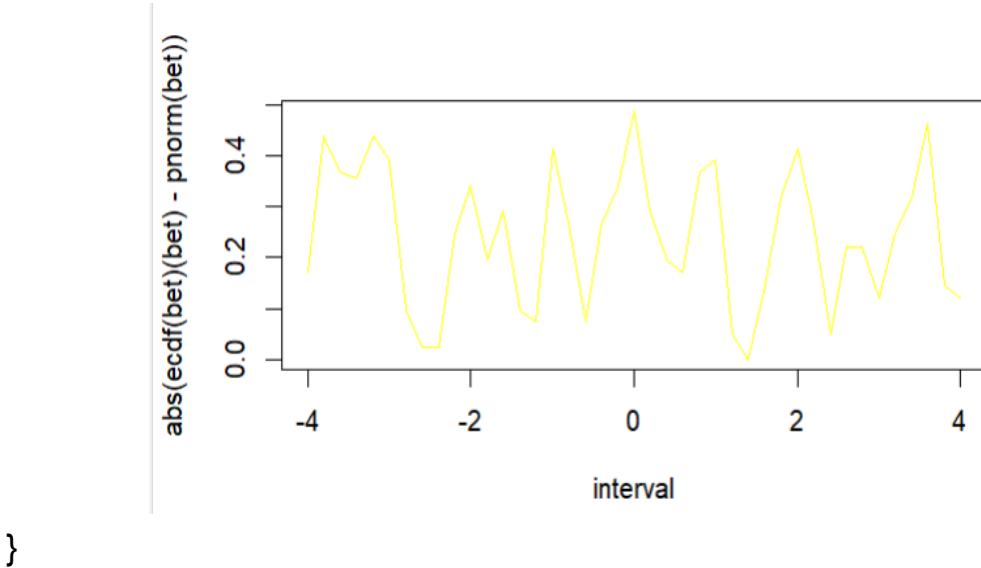
```
plot(interval,abs(ecdf(exp)(exp)-pnorm(exp)),type="l",lwd="1",col="magenta")
```



```
plot(interval,abs(ecdf(gam)(gam)-pnorm(gam)),type="l",lwd="1",col="black")
```



```
plot(interval,abs(ecdf(bet)(bet)-pnorm(bet)),type="l",lwd="1",col="yellow")
```



#Ex 8

#Folosim funcțiile construite anterior pentru medie, varianță și moment centrat de ordin trei.

```
MargineBE<-function(f,n,densitate=TRUE,E=medie,Var=varianta,MC3=Momentcentrat3){
  BE<-(33*MC3(f,densitate))/(4*(sqrt(n))^*((Var(f,densitate))^3))
  return(BE)
}
```

Exercitiul II)

Folosiți metoda respingerii pentru a genera observații din densitatea de probabilitate definită prin $f(x) = \exp(-x^2 / 2) \{ \sin(6x)^2 + 3 \cos(x)^2 \sin(4x)^2 + 1 \}$ parcurgând pașii următori:

(OBS: Notația "" înseamnă că $f(x)$ este proporțional cu expresia din dreapta)

a) Reprezentați grafic $f(x)$ și arătați că aceasta este mărginită de $Mg(x)$ unde $g(x)$ este densitatea de probabilitate a repartiției normale standard.
Determinați o valoare potrivită pentru constanta M , chiar dacă nu este optimă.

(Indiciu: Folosiți funcția optimise din R)

Vom folosi metoda respingerii pentru a genera observații pentru densitatea de probabilitate definită : (*) .

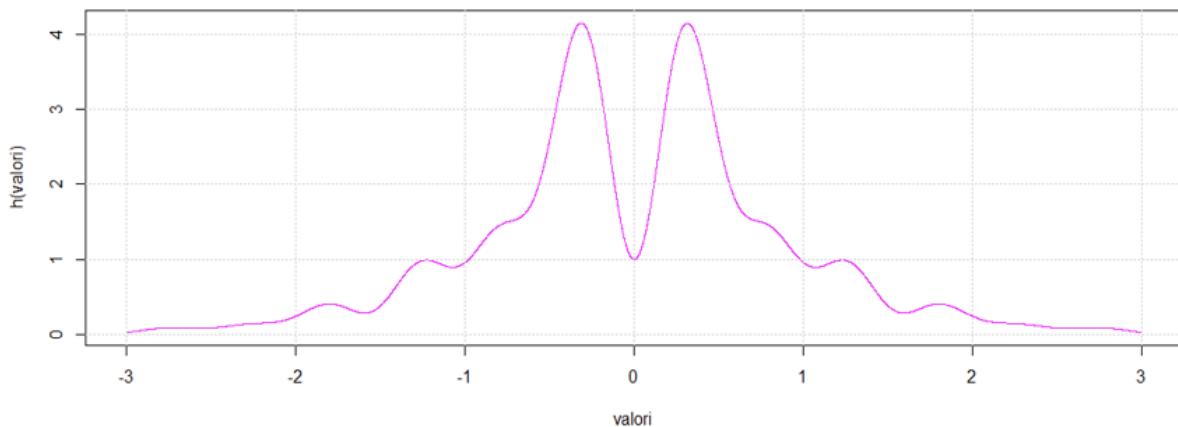
Notăm expresia (*) cu $h(x)$, unde $f(x)$ este proporțională cu $h(x) \rightarrow f(x) = C \cdot h(x)$.

Vom crea graficul funcției $h(x)$ folosind limbajul **R**:

```
h <- function(x){
  return (exp(-x^2/2)*(sin(6*x)^2 + 3*cos(x)^2*sin(4*x)^2 + 1))
}

valori <- seq(-3,3,0.0005)
plot(valori, h(valori), type = "l", col="magenta")
grid(nx=NULL,col="lightgray", lty="dotted", lwd=par("lwd"), equilogs=TRUE)
```

Am luat intervalul $[-3, 3]$ deoarece am observat că între $[-4, -2]$ graficul începe să urce. Prima dată am luat între $[-10, 10]$, dar graficul era mult mai compact iar între $[-10, -4]$ era o dreaptă.



Pentru a găsi constanta M , va trebui să studiem maximul funcției date de raportul:

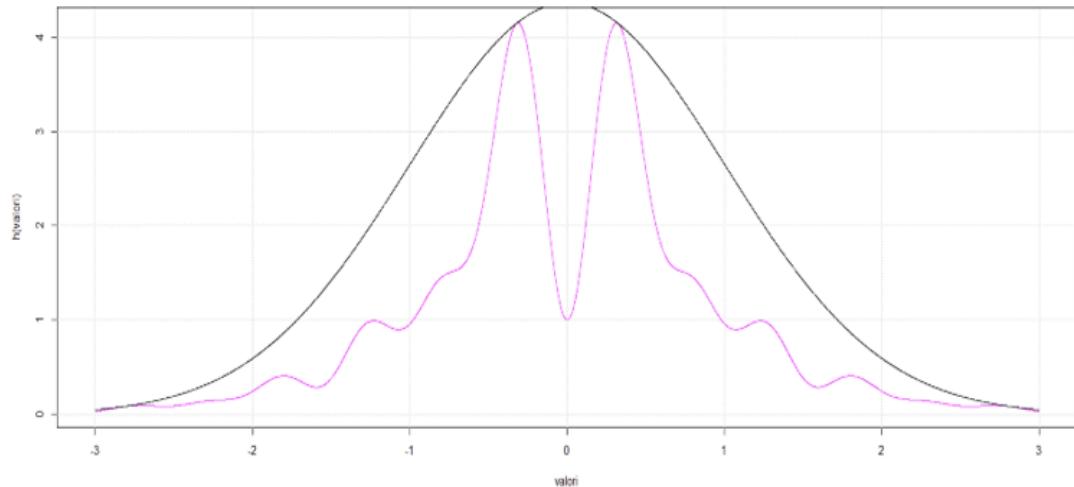
$$\frac{h(x)}{g(x)} = \sqrt{2\pi} * (\sin(6x)^2 + 3\sin(x)^2 * \sin(4x)^2 + 1)$$

Vom folosi funcția sugerată, **OPTIMISE**, pentru a găsi M:

```
raport <- function(x){  
  return (sqrt(2*pi) * ((sin(6*x)^2 + 3*cos(x)^2*sin(4*x)^2 + 1)))  
}  
  
M = optimise(raport, c(-2,2), maximum=TRUE)  
M[2]  
  
#$objective  
#[1] 10.94031
```

Folosim valoarea afaltă a lui M pentru a mărgini graficul funcției h(x):

```
raport <- function(x){  
  return (sqrt(2*pi) * ((sin(6*x)^2 + 3*cos(x)^2*sin(4*x)^2 + 1)))  
}  
M = optimise(raport, c(-2,2), maximum=TRUE)  
valori <- seq(-3,3,0.0005)  
plot(valori, h(valori), type ="l", col="magenta")  
grid(nx=NULL,col="lightgray", lty="dotted", lwd=par("lwd"), equilogs=TRUE)  
lines(valori, dnorm(valori)*M[[2]], col="black")
```



b) Generați 25000 de observații din densitatea de mai sus folosind metoda respingerii.

Vom folosi metoda respingerii pentru a genera 25000 de observații din densitatea de mai sus.

```

valorir <- c()
n <- 25000 #Nr. incercări
contor <- 0 #contorizam incercările bune
i <- 1

while( i<= n){

  u <- runif(1,0,1) #observatie din uniformă
  x <- rnorm(1,0,1) #observatie din normala standard
  if ( u <= h(x)/ (M[[2]] * dnorm(x))){
    valorir[contor] <- x
    contor <- contor + 1
  }
  i <- i + 1
}

```

c) Deducreți, pornind de la rata de acceptare a acestui algoritm, o aproximare a constantei de normalizare a lui $f(x)$, apoi comparați histograma valorilor generate cu reprezentarea grafică a lui $f(x)$ normalizată

Rata de acceptare:

```

ra <- contor/n
print("Rata de acceptare")

print(ra)

```

Prima rată de acceptare este:

```

#[1] "Rata de acceptare"

#[1] 0.58036

```

Vom reprezenta histograma după care facem comparația:

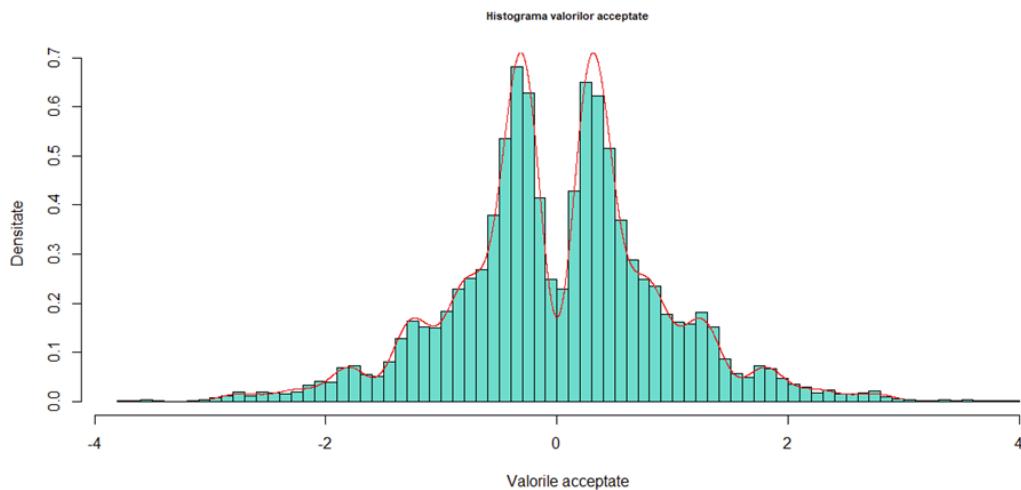
```

NormF <- 1/(M[[2]]*ra)
print("NormF")
print(NormF)

hist(valorir, breaks=100, freq = FALSE, col = "turquoise",
      xlab = "Valorile acceptate",
      ylab = "Densitate",
      main = "Histograma valorilor acceptate",
      cex.main = 0.7)

```

NormF reprezintă valoarea aproximativă a constantei ea fiind **0.1712862**



Exercitiul III)

#Ex 3) AnaAdina

#1)

```
library(ConvergenceConcepts)
```

#Plotul unei functii Binomiale de (1,1/2)

```
binom <- rbinom(1000,1,1/2)
```

```
hist(binom)
```

```
meanul<-mean(binom)
```

#Plotul unei functii Beta(1/n,1/n)

```
Beta<-function(n){
```

```
rbeta(n,1/n,1/n)
```

```
}
```

```
hist(Beta(1000))
```

```
check.convergence(nmax=10^3, M=500,genXn = Beta,mode="L", density =F,
```

```
densfunc=function(x){ dbinom(x,1,1/2)},
```

```
probfnc =function(x){ pbinom(x,1,1/2)},
```

```
tinf = 0, tsup = 1 )
```

#nmax: Specifică mărimea maximă a eşantionului pe care se va testa convergența.

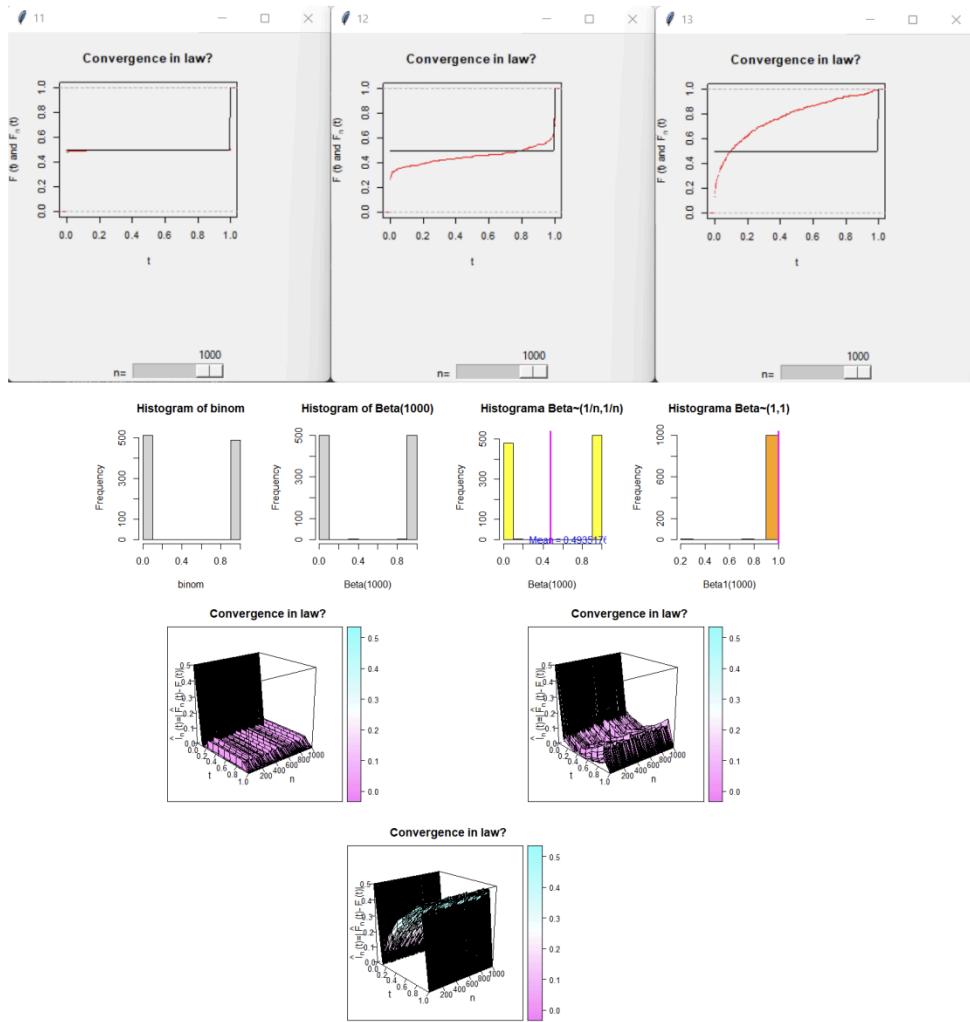
#M: Numărul de iterații pentru fiecare mărime a eşantionului (n).

#genXn: Funcția care generează eşantioanele distribuției de test (Beta în acest caz).

#densfunc: Funcția de densitate pentru distribuția de referință (în acest caz, distribuția binomială).

#probfnc: Funcția de distribuție cumulativă pentru distribuția de referință (în acest caz, distribuția binomială).

#Se poate observa ca Beta(1/n,1/n) converge (cand n tinde la infinit) la Bin(1,1/2)



#cercam plotarea functiei beta pentru $a>0$, $b>0$. Unde : Beta($a/n,b/n$).

#Am luat pentru a si b diverse valori

```
par(mfrow=c(1,2))
```

```
Beta<-function(n){
```

```
 rbeta(n,1/n,1/n)
```

```
}
```

```
hist(Beta(1000),col = 'yellow', main="Histograma Beta~(1/n,1/n)")
```

```
abline(v = mean(Beta(1000)),col = "magenta",lwd = 3)
```

```
text(x = mean(Beta(1000)) * 1.7,y = mean(Beta(1000)) * 1.7,
      paste("Mean =", mean(Beta(1000))),col = "blue",cex = 1)
```

```
Beta1<-function(n){
  rbeta(n,1,1/n)
}
hist(Beta1(1000), col = 'orange', main="Histograma Beta~(1,1/n)")
abline(v = mean(Beta1(1000)),col = "magenta",lwd = 3)
text(x = mean(Beta1(1000)) * 1.7,y = mean(Beta1(1000)) * 1.7,
      paste("Mean =", mean(Beta1(1000))),col = "blue",cex = 1)
```

```
# Se poate observa din grafic ca distributiile nu se aseamana
#Putem testa folosind functia pentru (a=b)<=n Check Convergence:
Beta1<-function(n){
  rbeta(n,100/n,100/n)
}
check.convergence(nmax=10^3, M=500,genXn = Beta1,mode="L", density =F,
                  densfunc=function(x){ dbinom(x,1,1/2)},
                  probfunc =function(x){ pbinom(x,1,1/2)},
                  tinf = 0, tsup = 1 )
#Testam functia (a!=b)<=n Check Convergence.
Beta3<-function(n){
  rbeta(n,3/10,1)
}
check.convergence(nmax=10^3, M=500,genXn = Beta3,mode="L", density =F,
                  densfunc=function(x){ dbinom(x,1,1/2)},
                  probfunc =function(x){ pbinom(x,1,1/2)},
                  tinf = 0, tsup = 1 )
#Concluzia este ca, marind a si b, distributia Beta nu converge catre
Binom(1,1/2).
#Observam ca pentru valorile 1/n si 1/n functia converge, iar pentru
#valori mai mari decat 1/n si diferite intre ele, functia nu mai converge.
```

```
#Subpunct 2)
#functieN reprezinta v.a i.i.d. uniforme pe intervalul {X1, X2, ...., Xn}
#functie1 reprezinta X~Unif(0,1)
#Am plotat histogrammele pentru a observa distributiile acestora.
par(mfrow=c(1,2))
```

```

functieN<-function(n)
{
  runif(n,min = 1/n, max= 1)
}
hist(functieN(10000))
functie1<-function(n)
{
  runif(n,0, max= 1)
}
hist(functie1(10000))

```

#Verificam convergenta distributie:

```

check.convergence(nmax=10^3, M=500,genXn = functieN,mode="L", density
=F,
  densfunc=function(x){ dunif(x,0,1)},
  probfunc =function(x){ punif(x,0,1)},
  tinf = 0, tsup = 1 )

```

#Verificam convergenta probabilitate:

```

check.convergence(nmax=10^3, M=500,genXn = functieN,mode="p", density
=F,
  densfunc=function(x){ dunif(x,0,1)},
  probfunc =function(x){ punif(x,0,1)},
  tinf = 0, tsup = 0 )

```

#Pentru subpunctul 2, functia converge.

#Subpunct 3)

#Pentru a verifica dacă $\min\{X_1, X_2, \dots, X_n\}$ converge aproape sigur la m și dacă $\max\{X_1, X_2, \dots, X_n\}$ converge aproape sigur la M :

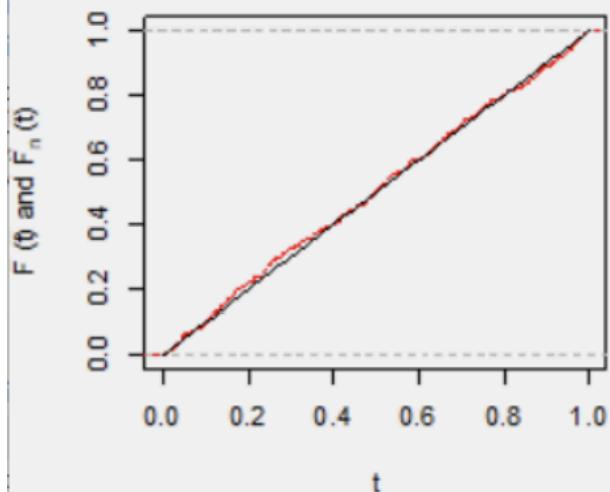
#Generăm un eşantion mare din distribuția dorită a variabilei aleatoare X .

#Calculăm minimele și maximele eşantionului pentru diferite dimensiuni ale eşantionului (de exemplu, pornind de la dimensiunea 1 până la dimensiunea totală a eşantionului).

#Verificăm dacă valorile minime și maxime se apropie de m și, respectiv, de M , pe măsură ce dimensiunea eşantionului crește.

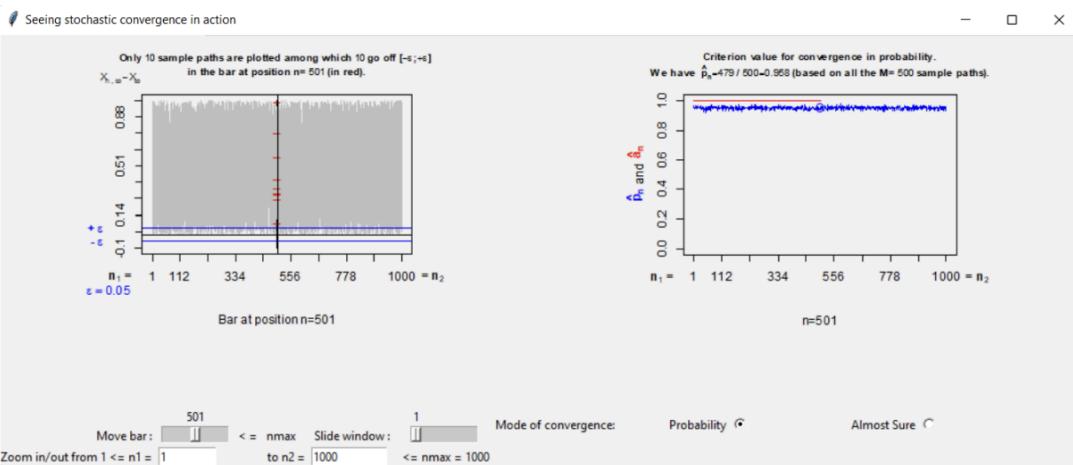
19

Convergence in law?



486

n =

#Fie $X \sim N(m, \sigma^2)$

```

m <- 10 #Medie
sigma <- 2 #deviatie
n <- 1000 #Dimensiunea esantion

X <- rnorm(n, mean = m, sd = sigma) #eșantion din distribuția normală
true_values <- list(min = m, max = m + sigma) #Valorile adevărate pentru m și
M

# Funcție pentru a extrage minimele și maximele
extract_min_max <- function(X_seq) {
  min_value <- sapply(X_seq, min)
  max_value <- sapply(X_seq, max)
  return(list(min = min_value, max = max_value))
}

F1<-function(n){
  c<-c()
  for (i in 1:n){
    val<-rnorm(n,mean = m, sd = sigma)
    v<-extract_min_max(val)
    c<-concat(v,c)
  }
}

# Funcție pentru a genera un eșantion de dimensiune n din distribuția normală
# cu parametrii m și sigma
F1 <- function(n) {
  X1 <- rnorm(n, mean = m, sd = sigma)
  return(X1)
}

# Verificăm convergența aproximativă a minimei și maximei la m și, respectiv,
M
check.convergence(min_max_values$min, true_values$min, genXn = F1,
mode = "almost surely")
check.convergence(min_max_values$max, true_values$max, genXn = F1,
mode = "almost surely")

```

Exercitiul IV)

Indicatii teoretice pentru 4.1.

a) Probabilitatea de a câștiga după o încercare cumpărând un bilet la categoria Simpler:

- 3 numere : $\frac{C_6^3 \cdot C_{43}^3}{C_{49}^6} \approx 0,0176$

- 4 numere : $\frac{C_6^4 \cdot C_{43}^2}{C_{49}^6} \approx 0,00095$

- 5 numere : $\frac{C_6^5 \cdot C_{43}^1}{C_{49}^6} \approx 1,9 \cdot 10^{-5}$

- 6 numere $\frac{C_6^6 \cdot C_{43}^0}{C_{49}^6} \approx 10^{-6}$

b) Probabilitatea de succes după k încercări este $(1-p_i)^{k-1} \cdot p_i$ unde k încercări sunt egale cu $k-1$ eșecuri.

c) Probabilitatea de r succese după k încercări este $(1-p_i)^{k-r} \cdot p_i^r$, $i = \overline{3,6}$

d) Probabilitatea dintre k eșecuri și r câștiguri este $(1-p_i)^k \cdot p_i^r$, $i = \overline{3,6}$

f) Într-un an putem juca s_2 de bilete aproximativ, iar probabilitatea de a nu câștiga niciodată pentru fiecare categorie de câștig este $(1-p)^{s_2}$.

4.1. Folosindu-vă de simulare (10^6 valori), estimați următoarele cantități, iar acolo unde este posibil, comparați cu rezultatul teoretic:

a) Probabilitatea de a câștiga din prima încercare, cu un bilet simplu, la fiecare dintre categoriile posibile de câștig.

```
# Funcție pentru calculul probabilității de a potrivi un anumit număr de numere
calc_prob <- function(k) {
  choose(6, k) * choose(43, 6-k) / choose(49, 6)
}

# Simularea pentru a estima probabilitatea de a câștiga din prima încercare
simu_prim_inc <- function(k, num_sim) {
  prob <- calc_prob(k)
  castiguri <- 0

  for (i in 1:num_sim) {
    if (any(runif(1) < prob)) {
      castiguri <- castiguri + 1
    }
  }

  return(castiguri / num_sim)
}

# Numărul de simulări
num_sim <- 10^6

# Calcularea probabilităților pentru fiecare categorie de câștig
prob_3n <- simu_prim_inc(3, num_sim)
prob_4n <- simu_prim_inc(4, num_sim)
prob_5n <- simu_prim_inc(5, num_sim)
prob_6n <- simu_prim_inc(6, num_sim)

# Afisarea probabilităților
prob_3n
prob_4n
prob_5n
prob_6n

> # Afisarea probabilitatilor
> prob_3n
[1] 0.017593
> prob_4n
[1] 0.000947
> prob_5n
[1] 1.9e-05
> prob_6n
[1] 1e-06
```

4.1.b) Probabilitatea de a câștiga după k încercări(k fiind un parametru), cu un bilet simplu, la fiecare dintre categoriile posibile de câștig.

```
# simularea pentru a estima probabilitatea de a câștiga cel puțin o dată după k încercări
sim_cpod <- function(k, prob, num_sim) {
  castiguri <- 0

  for (i in 1:num_sim) {
    if (any(runif(k) < prob)) {
      castiguri <- castiguri + 1
    }
  }

  return(castiguri / num_sim)
}

# Setarea numărului de încercări
k <- 10 # Aici poți schimba numărul de încercări
num_sim <- 10^6

# Calculul probabilităților pentru fiecare categorie de câștig
prob_3n <- simu_prim_inc(3, num_sim)
prob_4n <- simu_prim_inc(4, num_sim)
prob_5n <- simu_prim_inc(5, num_sim)
prob_6n <- simu_prim_inc(6, num_sim)

# Estimarea probabilității de a câștiga cel puțin o dată după k încercări prin simulare
prob_sim_k_3 <- sim_cpod(k, prob_3n, num_sim)
prob_sim_k_4 <- sim_cpod(k, prob_4n, num_sim)
prob_sim_k_5 <- sim_cpod(k, prob_5n, num_sim)
prob_sim_k_6 <- sim_cpod(k, prob_6n, num_sim)

# Afisarea rezultatelor simulate
prob_sim_k_3
prob_sim_k_4
prob_sim_k_5
prob_sim_k_6

> # Afisarea rezultatelor simulate
> prob_sim_k_3
[1] 0.163503
> prob_sim_k_4
[1] 0.010067
> prob_sim_k_5
[1] 0.000177
> prob_sim_k_6
[1] 0
```

4.1.c) Probabilitatea de a câștiga de r ori din k încercări(r și k fiind parametri), cu un bilet simplu, la fiecare dintre categoriile posibile de câștig.

```

# Simularea pentru a estima probabilitatea de a câștiga de r ori din k încercări
sim_rk <- function(r, k, prob, num_sim) {
  catiguri_r <- 0

  for (i in 1:num_sim) {
    succese <- sum(runif(k) < prob)
    if (succese == r) {
      catiguri_r <- catiguri_r + 1
    }
  }

  return(catiguri_r / num_sim)
}

# Setarea parametrilor
k <- 10 # Numărul total de încercări
r <- 2 # Numărul dorit de câștiguri
num_sim <- 10^6

# Calculul probabilităților pentru fiecare categorie de câștig
prob_3n <- calc_prob(3)
prob_4n <- calc_prob(4)
prob_5n <- calc_prob(5)
prob_6n <- calc_prob(6)

# Estimarea probabilității de a câștiga de r ori din k încercări prin simulare
prob_sim_r_k_3 <- sim_rk(r, k, prob_3n, num_sim)
prob_sim_r_k_4 <- sim_rk(r, k, prob_4n, num_sim)
prob_sim_r_k_5 <- sim_rk(r, k, prob_5n, num_sim)
prob_sim_r_k_6 <- sim_rk(r, k, prob_6n, num_sim)

# Afisarea rezultatelor simulate
prob_sim_r_k_3
prob_sim_r_k_4
prob_sim_r_k_5
prob_sim_r_k_6

> # Afisarea rezultatelor simulate
> prob_sim_r_k_3
[1] 0.012123
> prob_sim_r_k_4
[1] 3.9e-05
> prob_sim_r_k_5
[1] 0
> prob_sim_r_k_6
[1] 0

```

4.1.d) Probabilitatea de a câștiga de r ori după k eșecuri (r și k fiind parametri), cu un bilet simplu, la fiecare dintre categoriile posibile de câștig.

```

# Simularea pentru a estima probabilitatea de a câștiga de r ori după k eșecuri
sim_rke <- function(r, k, num_sim) {
  prob_3 <- calc_prob(3)
  prob_4 <- calc_prob(4)
  prob_5 <- calc_prob(5)
  prob_6 <- calc_prob(6)

  rez <- list('3' = 0, '4' = 0, '5' = 0, '6' = 0)
  |
  for (i in 1:num_sim) {
    count <- c('3' = 0, '4' = 0, '5' = 0, '6' = 0)

    for (j in 1:(k + r)) {
      if (runif(1) < prob_3) {
        count[['3']] <- count[['3']] + 1
      }
      if (runif(1) < prob_4) {
        count[['4']] <- count[['4']] + 1
      }
      if (runif(1) < prob_5) {
        count[['5']] <- count[['5']] + 1
      }
      if (runif(1) < prob_6) {
        count[['6']] <- count[['6']] + 1
      }
    }

    if (count[['3']] == r) {
      rez[['3']] <- rez[['3']] + 1
    }
    if (count[['4']] == r) {
      rez[['4']] <- rez[['4']] + 1
    }
    if (count[['5']] == r) {
      rez[['5']] <- rez[['5']] + 1
    }
    if (count[['6']] == r) {
      rez[['6']] <- rez[['6']] + 1
    }
  }

  return(list('3' = rez[['3']] / num_sim,
             '4' = rez[['4']] / num_sim,
             '5' = rez[['5']] / num_sim,
             '6' = rez[['6']] / num_sim))
}

# Restul codului poate rămâne neschimbat.

# Numărul de simulări
num_sim <- 10^6

# Exemplu de utilizare a funcției pentru r=2 câștiguri după k=5 eșecuri
r <- 2
k <- 5
sim_rke(r, k, num_sim)

$`3`
[1] 0.005975

$`4`
[1] 1.6e-05

$`5`
[1] 0

$`6`
[1] 0

```

4.1.e) Probabilitatea de a câștiga, în medie, cel puțin o dată pe an, dacă se joacă săptămânal câte un bilet simplu, timp de 30 ani.

```

# Simularea pentru a estima probabilitatea de a câștiga cel puțin o dată pe an
sim_cpdata <- function(num_sim) {
  prob_3 <- calc_prob(3)
  prob_4 <- calc_prob(4)
  prob_5 <- calc_prob(5)
  prob_6 <- calc_prob(6)

  ani_castigati <- 0

  for (i in 1:num_sim) {
    castig_anual <- FALSE
    # Fiecare an are 104 extrageri (2 pe săptămână)
    for (extragere in 1:104) {
      # Simulăm o extragere verificând contra tuturor probabilităților de câștig
      if (runif(1) < prob_3 || runif(1) < prob_4 || runif(1) < prob_5 || runif(1) < prob_6) {
        castig_anual <- TRUE
        break
      }
    }
    if (castig_anual) {
      ani_castigati <- ani_castigati + 1
    }
  }

  return(ani_castigati / num_sim)
}

# Numărul de simulări
num_sim <- 10^6

# Calcularea probabilității
prob_m <- sim_cpdata(num_sim)

# Afisarea probabilității
print(prob_m)

```

> # Afisarea probabilității
> print(prob_m)
[1] 0.858532

4.1.f) Probabilitatea de a juca săptămânal, timp de un an, câte un bilet simplu și a nu câștiga niciodată

```

sim_ncn <- function(num_sim) {
  prob_3 <- calc_prob(3)
  prob_4 <- calc_prob(4)
  prob_5 <- calc_prob(5)
  prob_6 <- calc_prob(6)

  # Probabilitatea totală de a câștiga la o singură extragere
  prob_castig <- prob_3 + prob_4 + prob_5 + prob_6

  # Probabilitatea de a nu câștiga la o singură extragere
  prob_nc <- 1 - prob_castig

  ani_fara_castig <- 0

  for (i in 1:num_sim) {
    # Simulăm 104 extrageri pe an
    if (all(runif(104) > prob_castig)) {
      ani_fara_castig <- ani_fara_castig + 1
    }
  }

  return(ani_fara_castig / num_sim)
}

# Numărul de simulări
num_sim <- 10000 # Ajustat pentru timpul de execuție

# Calcularea probabilității
prob_ncn <- sim_ncn(num_sim)

# Afisarea probabilității
prob_ncn

```

> prob_ncn
[1] 0.1428

4.1.g) Probabilitatea de a juca săptămânal, timp de un an, câte un bilet simplu și de a câștiga, cumulat, o sumă mai mare decât costul total al biletelor jucate.

```
# Parametrii jocului
cost_bilet <- 7
num_sap <- 52 * 2 # Dacă sunt două extrageri pe săptămână, ajustăm numărul de săptămâni
cost_an_bilet <- cost_bilet * num_sap

# Numărul de simulări
num_sim <- 10^3

# Probabilități de câștig pentru fiecare categorie
prob_c <- c(0, calc_prob(3), calc_prob(4), calc_prob(5), calc_prob(6))

# Sume câștig medii pentru fiecare categorie
sume_c <- c(0, 30, 363350 / 100, 390000 / 10, 1090000 / 2) # Exemple de sume medii

# Funcție pentru simularea câștigurilor pe o săptămână
sim_sap <- function() {
  rnd <- runif(1)
  sum_prob <- 0
  for (i in 1:length(prob_c)) {
    sum_prob <- sum_prob + prob_c[i]
    if (rnd < sum_prob) {
      return(sume_c[i])
    }
  }
  return(0)
}

# Simularea jocului pe un an
sim_an <- function() {
  castiguri_an <- sum(sapply(1:num_sap, function(x) sim_sap()))
  return(castiguri_an - cost_an_bilet) # Returnăm diferență netă, nu condiția
}

# Calculul rezultatelor
rezultate <- replicate(num_sim, sim_an())
# Calculăm procentul de cazuri cu profit net (diferență netă > 0)
prob_cpc <- mean(rezultate > 0)

prob_cpc
```

> prob_cpc
[1] 0.09

Acest rezultat este pentru 10^3 simulări. Pentru 10^6 simulări am stat foarte mult timp și nu-mi genera niciun rezultat

4.1.h) Probabilitatea de a câștiga din nou, la orice categorie, în săptămâna următoare, cu un bilet simplu, presupunând că în săptămâna curentă jucătorul a câștigat premiul cel mare.

```
# Funcție pentru calculul probabilității de a potrivi un anumit număr de numere
calc_prob <- function(k) {
  return(choose(6, k) * choose(43, 6-k) / choose(49, 6))
}

# Calcularea probabilității totale de a câștiga la orice categorie
prob_tc <- sum(sapply(3:6, calc_prob))

# simularea a 1.000.000 de încercări pentru a câștiga din nou în săptămâna următoare
num_sim <- 10^6
castiguri <- sum(runif(num_sim) < prob_tc)

# Calcularea probabilității
prob_cu <- castiguri / num_sim

# Afisarea probabilității
prob_cu

> # Afisarea probabilității
> prob_cu
[1] 0.018644
```

4.2. Folosindu-vă de simulare și/sau rezultate teoretice răspundeți justificat la următoarele întrebări:

a) Care este câștigul mediu anual estimat al Loteriei, știind că există 2 extrageri săptămânale?

```
# Funcție modificată pentru a include raportarea numărului de câștigători pentru 4, 5 și 6 numere
castig_mediul_anual <- function(analize=100, extrageri_pe_an=104, bilete_vandute_per_extragere=10^6, cost_bilet=7) {
  # Inițializarea vectorilor pentru stocarea rezultatelor
  castiguri_anuale <- numeric(analize)
  nr_castigatori_4 <- numeric(analize)
  nr_castigatori_5 <- numeric(analize)
  nr_castigatori_6 <- numeric(analize)

  for (i in 1:analize) {
    # Simularea numărului de câștigători pentru fiecare extragere și categorie
    castigatori_4 <- rbinom(extrageri_pe_an, bilete_vandute_per_extragere, P_4)
    castigatori_5 <- rbinom(extrageri_pe_an, bilete_vandute_per_extragere, P_5)
    castigatori_6 <- rbinom(extrageri_pe_an, bilete_vandute_per_extragere, P_6)

    # Calculul cheltuielilor și câștigurilor anuale
    cheltuieli_anuale <- sum(castigatori_4 * (fond_castig_4 / ifelse(castigatori_4 > 0, castigatori_4, 1))) +
      sum(castigatori_5 * (fond_castig_5 / ifelse(castigatori_5 > 0, castigatori_5, 1))) +
      sum(castigatori_6 * (fond_castig_6 / ifelse(castigatori_6 > 0, castigatori_6, 1)))

    venituri_anuale <- bilete_vandute_per_extragere * cost_bilet * extrageri_pe_an
    castiguri_anuale[i] <- venituri_anuale - cheltuieli_anuale

    # Stocarea numărului total de câștigători pentru fiecare categorie
    nr_castigatori_4[i] <- sum(castigatori_4)
    nr_castigatori_5[i] <- sum(castigatori_5)
    nr_castigatori_6[i] <- sum(castigatori_6)
  }

  # Raportarea rezultatelor
  list(
    castig_mediul_anual = mean(castiguri_anuale),
    nr_mediul_castigatori_4 = mean(nr_castigatori_4),
    nr_mediul_castigatori_5 = mean(nr_castigatori_5),
    nr_mediul_castigatori_6 = mean(nr_castigatori_6)
  )
}

# Rularea funcției modificată
rezultate <- castig_mediul_anual()
print(rezultate)
```

```

$castig_mediul_anual
[1] 641280400

$nr_mediul_castigatori_4
[1] 100724.3

$nr_mediul_castigatori_5
[1] 1915.7

$nr_mediul_castigatori_6
[1] 7.97

```

4.2.b)

Care este numărul minim de bilete simple care trebuie cumpărate într-un an pentru ca Loteria să nu fie în pierdere?

```

# Funcție pentru calculul combinațiilor
choose <- function(n, k) {
  factorial(n) / (factorial(k) * factorial(n - k))
}

# Numărul total de numere (n) și numărul de numere extrase (k)
n <- 49
k <- 6

# Calculul combinațiilor totale posibile
comb_total <- choose(n, k)

# Probabilități de câștig pentru fiecare categorie
# Pentru 3 numere ghicite
comb_3 <- choose(k, 3) * choose(n - k, k - 3)
prob_3 <- comb_3 / comb_total

# Pentru 4 numere ghicite
comb_4 <- choose(k, 4) * choose(n - k, k - 4)
prob_4 <- comb_4 / comb_total

# Pentru 5 numere ghicite
comb_5 <- choose(k, 5) * choose(n - k, k - 5)
prob_5 <- comb_5 / comb_total

# Pentru 6 numere ghicite
comb_6 <- choose(k, 6)
prob_6 <- comb_6 / comb_total

# Fondurile de câștig pentru fiecare categorie
castig_3 <- 30 # Suma fixă pentru 3 numere ghicite
castig_4 <- 363350
castig_5 <- 390000
castig_6 <- 1090000

# Așteptarea de câștig pentru un bilet simplu
asteptare_castig_per_bilet <- (prob_3 * castig_3) + (prob_4 * castig_4) + (prob_5 * castig_5) + (prob_6 * castig_6)

# Costul unui bilet
cost_bilet <- 7

# Afisarea așteptării de câștig și a costului biletului
print(asteptare_castig_per_bilet)
print(cost_bilet)

> # Afisarea așteptării de câștig și a costului biletului
> print(asteptare_castig_per_bilet)
[1] 359.7509
> print(cost_bilet)
[1] 7

```

4.2.c)

Un jucător are un buget lunar de 70 lei. Care ar fi strategia optimă de joc(bilete simple jucate în săptămâni diferite, un singur bilet cu o variantă complexă jucată o singură dată etc.) care să-i aducă cele mai mari şanse de câştig? Estimați/Calculați aceste şanse!

```
# Definirea funcției pentru calculul combinațiilor
combn <- function(n, k) {
  factorial(n) / (factorial(k) * factorial(n - k))
}

# Probabilități de câștig pentru un bilet simplu
prob_3_numere <- combin(6, 3) * combin(43, 3) / combin(49, 6)
prob_4_numere <- combin(6, 4) * combin(43, 2) / combin(49, 6)
prob_5_numere <- combin(6, 5) * combin(43, 1) / combin(49, 6)
prob_6_numere <- 1 / combin(49, 6)

# Calcularea şanselor de câștig pentru bilete simple (10 încercări)
# Presupunem că probabilitățile de câștig pentru fiecare bilet sunt independente
# și că jucătorul poate cumpăra 10 bilete simple cu bugetul de 70 lei.
total_prob_simplu <- 1 - (1 - prob_3_numere)^10
# Notă: Acest calcul simplificat presupune interes doar pentru câștigul la 3 numere

# Evaluarea opțiunii pentru o variantă combinată
# Exemplu: Calcul pentru 7 numere (cea mai accesibilă variantă combinată în buget)
numere_alese <- 7 # Alege 7 numere pentru varianta combinată
combinatii_posibile <- combin(numere_alese, 6)
cost_varianta_combinata <- combinatii_posibile * 7

# Verificăm dacă varianta combinată se încadrează în buget
if (cost_varianta_combinata <= 70) {
  print(paste("Costul pentru varianta combinată cu", numere_alese, "numere este:", cost_varianta_combinata, "lei"))
  # Calcul probabilitate de câștig pentru varianta combinată (doar pentru 3 numere, ca exemplu simplificat)
  total_prob_combinat <- 1 - (1 - prob_3_numere)^combinatii_posibile
} else {
  print(paste("O variantă combinată cu", numere_alese, "numere depășește bugetul de 70 lei."))
}

# Compararea probabilităților de câștig
print(paste("Probabilitatea cumulativă de câștig pentru bilete simple:", total_prob_simplu))
if (exists("total_prob_combinat")) {
  print(paste("Probabilitatea de câștig pentru varianta combinată:", total_prob_combinat))
}

+ s
[1] "Costul pentru varianta combinată cu 7 numere este: 49 lei"
>
> # Compararea probabilităților de câștig
> print(paste("Probabilitatea cumulativă de câștig pentru bilete simple:", total_prob_simplu))
[1] "Probabilitatea cumulativă de câștig pentru bilete simple: 0.163124778391904"
> if (exists("total_prob_combinat")) {
+   print(paste("Probabilitatea de câștig pentru varianta combinată:", total_prob_combinat))
+ }
[1] "Probabilitatea de câștig pentru varianta combinată: 0.11719965024618"
> |
```

4.2.d)

Care este numărul mediu de câștigători de la fiecare categorie de la o extragere? Puteți identifica o repartiție asociată?

```
# Numărul total de bilete jucate (presupunere)
total_bilete <- 1000000

# Calculul combinărilor posibile pentru 6 din 49
combinari_totale <- choose(49, 6)

# Calculul probabilităților pentru fiecare categorie de câștig (3, 4, 5, 6 numere corecte)
probabilitati <- list(
  `3_numere_corecte` = choose(6, 3) * choose(43, 3) / combinari_totale,
  `4_numere_corecte` = choose(6, 4) * choose(43, 2) / combinari_totale,
  `5_numere_corecte` = choose(6, 5) * choose(43, 1) / combinari_totale,
  `6_numere_corecte` = choose(6, 6) * choose(43, 0) / combinari_totale
)

# Calculul numărului mediu de câștigători pentru fiecare categorie
numar_mediul_castigator <- sapply(probabilitati, function(p) p * total_bilete)

# Afisarea numărului mediu de câștigători pentru fiecare categorie
print(numar_mediul_castigator)
```

```
> # Afisarea numărului mediu de câștigători pentru fiecare categorie
> print(numar_mediul_castigator)
3_numere_corecte 4_numere_corecte 5_numere_corecte 6_numere_corecte
1.765040e+04     9.686197e+02     1.844990e+01     7.151124e-02
```

4.4.

Loteria pune la dispoziție istoricul extragerilor anterioare din ultimii 20 de ani. În ce manieră puteți folosi aceste informații pentru a crește șansele de câștig cu un bilet simplu? Justificați răspunsul și ilustrați-l cu grafice relevante.

```
# Simulăm date pentru extragerile din ultimii 20 de ani (1040 extrageri/an, presupunând 2 extrageri/săptămână)
set.seed(123) # Pentru reproducibilitate
numar_extrageri <- 20 * 52 * 2
numere_extrase <- replicate(numar_extrageri, sample(1:49, 6, replace = FALSE))

# Convertim la un vector unidimensional pentru a calcula frecvența fiecărui număr
numere_extrase_vector <- as.vector(numere_extrase)

# Calculăm frecvența fiecărui număr
frecventa_numere <- table(numere_extrase_vector)

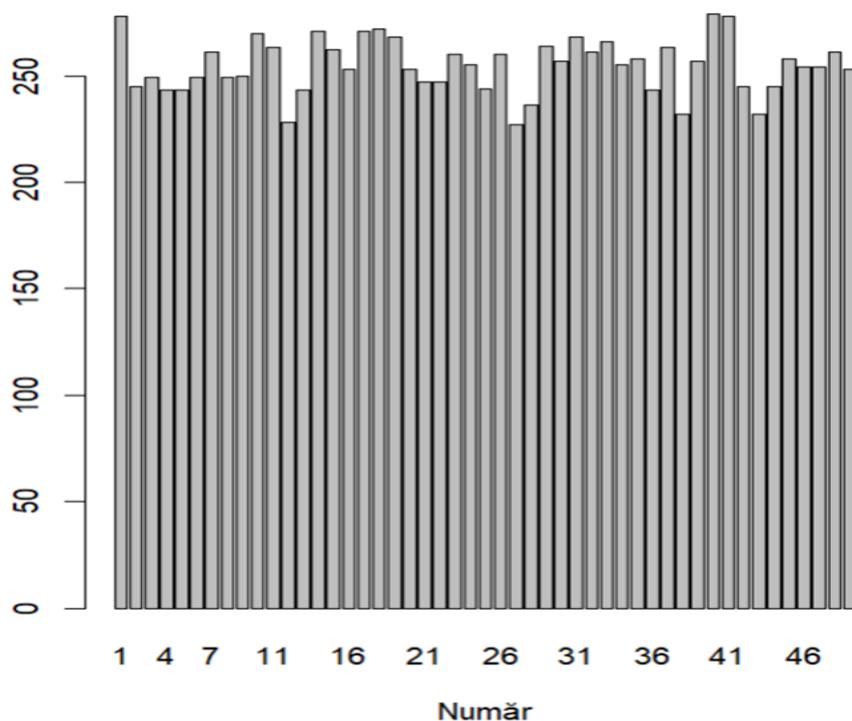
# Creăm un grafic cu frecvențele numerelor extrase
barplot(frecventa_numere, main = "Frecvența numerelor extrase în ultimii 20 de ani", xlab = "Număr", ylab = "Frecvență")

# Analiza sanse de câștig
pret_bilet <- 7
castiguri <- c(30, 363350, 390000, 1090000)
sanse_castig <- c(choose(49, 3) / choose(6, 3) * choose(43, 3), # Șanse pentru 3 numere
                  choose(49, 4) / choose(6, 4) * choose(43, 2), # Șanse pentru 4 numere
                  choose(49, 5) / choose(6, 5) * choose(43, 1), # Șanse pentru 5 numere
                  choose(49, 6)) # Șanse pentru 6 numere

# Calculăm sansele de câștig pentru fiecare caz
sansa_castig <- sum(castiguri * sanse_castig) - pret_bilet

sansa_castig
```

Frecvența numerelor extrase în ultimii 20 de ani



Loteria 6/49 este un exemplu clasic de proces aleator, unde fiecare extragere este un eveniment independent. Acest lucru înseamnă că rezultatul unei extrageri nu este influențat de rezultatele extragerilor anterioare. Prin urmare, fiecare număr are exact aceeași șansă de a fi ales în orice extragere dată, indiferent de frecvența sa istorică de apariție.

Din punct de vedere statistic, analiza datelor istorice nu va oferi o metodă de a "bate" sistemul sau de a crește șansele de câștig, deoarece șansele sunt fixate și nu sunt influențate de modelele anterioare. În jocuri cu adevărat aleatorii cum ar fi loteria, fiecare număr are o șansă de 1/49 să fie ales la fiecare extragere, indiferent de ceea ce s-a întâmplat în trecut.

Concluzia este că nu există o metodă statistică sau matematică pentru a folosi informațiile despre extragerile anterioare pentru a crește șansele de câștig la un bilet simplu la loteria 6/49.