Laborator 4

Elemente de simulare în R

Obiectivul acestui laborator este de a prezenta câteva probleme de simulare folosind noțiunile și metodele învățate la curs.

1 Generarea variabilelor aleatoare discrete

Fie X o variabilă aleatoare cu valori în mulțimea $\{1, 2, 3, 4\}$. Repartiția ν a lui X este

$$\mathbb{P}(X = 1) = 0.2, \quad \mathbb{P}(X = 2) = 0.5, \quad \mathbb{P}(X = 3) = 0.1, \quad \mathbb{P}(X = 4) = 0.2.$$

1. Simulați un eșantion $\boldsymbol{u}$ de 10000 de variabile aleatoare i.i.d. repartizate $\mathcal{U}([0, 1])$.
2. Plecând de la acest eșantion, construiți un eșantion $\boldsymbol{x}$ de variabile aleatoare i.i.d. repartizate conform ν.
3. Comparați, cu ajutorul *diagramei cu bare verticale* (`barplot`), repartiția eșantionului $\boldsymbol{x}$ și cea teoretică ν.

1. Pentru a genera observații independente repartizate uniform vom folosi funcția `runif`:

```
n =  10000
u = runif(n)
```

2. Putem scrie

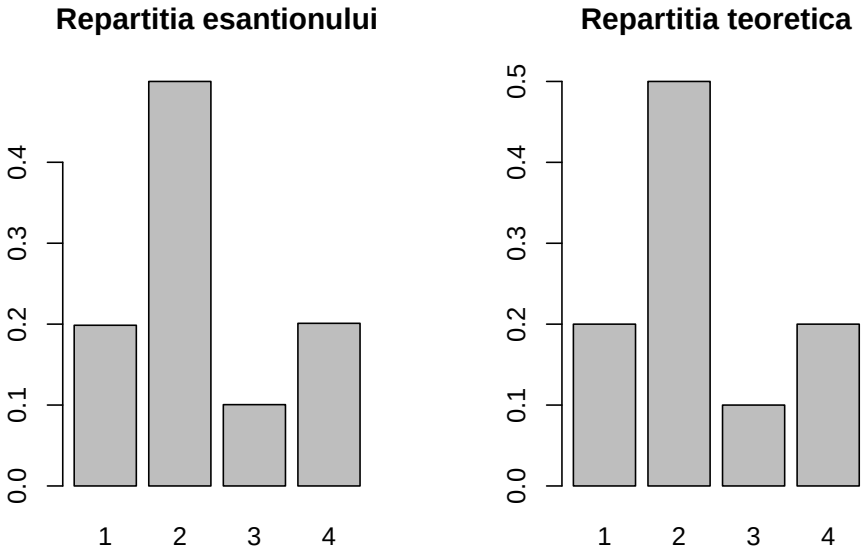
```
x = 1 + (u > 0.2) + (u > 0.7) + (u > 0.8)
```

Trebuie remarcat că funcția `sample` din R permite simularea repartițiilor discrete. Pentru exemplul nostru putem să extragem, cu întoarcere, n numere din mulțimea $\{1, 2, 3, 4\}$ urmând un vector de probabilități `prob`:

```
x = sample(1:4, n, replace = TRUE,
           prob = c(0.2, 0.5, 0.1, 0.2))
```

3. Pentru început trebuie să determinăm câte observații sunt din fiecare valoare unică a lui $\boldsymbol{x}$. Putem face acest lucru folosind funcția `table`:

```
x.freq = table(x)/n

par(mfrow = c(1,2))
# diagrama cu bare verticale pentru esantion
barplot(x.freq, main = "Repartitia esantionului")
# diagrama cu bare verticale teoretica
barplot(c(0.2, 0.5, 0.1, 0.2), names.arg = c(1,2,3,4),
        main = "Repartitia teoretica")
```



Observăm că eșationul este repartizat conform ν.

În acest exercițiu ne propunem să definim o funcție `rand_sample(n,x,p)` care permite generarea a n observații dintr-o mulțime x (vector numeric sau de caractere) cu probabilitatea p pe x (un vector de aceeași lungime ca x).

Funcția se poate construi sub forma următoare:

```
rand_sample = function(n,x,p){
  # n - numarul de observatii
  # x - multimea de valori
  # p - vectorul de probabilitati

  out = c()

  ind = 1:length(x)
  cs = cumsum(p)

  if (length(x)!=length(p)){
    return(print('Cei doi vectori ar trebui sa fie de aceeasi lungime !'))
  }

  for (i in 1:n){
    r = runif(1)

    m = min(ind[r<=cs])
    out = c(out,x[m])
  }

  return(out)
}
```

Pentru a testa această funcție să considerăm două exemple:

1. în acest caz: $n = 10$, $x = [1, 2, 3]$ și $p = [0.2, 0.3, 0.5]$

```
rand_sample(10,c(1,2,3),c(0.2,0.3,0.5))
 [1] 1 1 3 2 3 1 3 3 3 3
```

2. în acest caz: $n = 15$, $x = [a, b, c, d]$ și $p = [0.15, 0.35, 0.15, 0.45]$

```
rand_sample(15,c('a','b','c','d'),c(0.15,0.35,0.15,0.45))
 [1] "a" "d" "b" "b" "b" "c" "d" "c" "a" "b" "d" "c" "c" "d" "a"
```

O funcție un pic mai generală este:

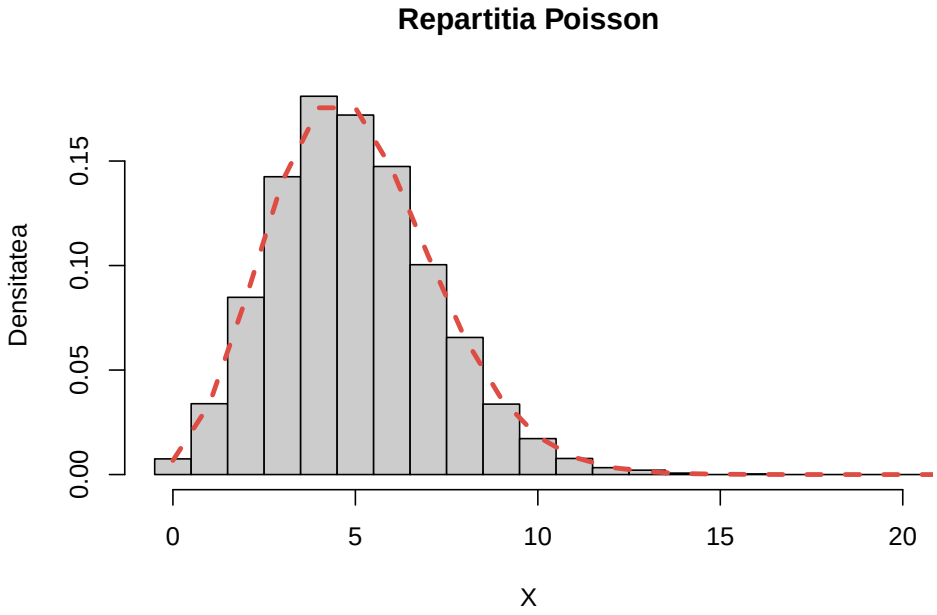
```
GenerateDiscrete = function(n = 1, x, p, err = 1e-15){
  # n numarul de observatii
  # x multimea de valori
  # p vectorul de probabilitati

  lp = length(p)
  lx = length(x)

  # verificarea conditiilor de aplicare
  if(abs(sum(p)-1)>err | sum(p>=0)!=lp){

    stop("Suma probabilitatilor nu este egala cu 1!")

  }else if(lx!=lp){

    stop("x si p trebuie sa aiba aceeasi marime!")

  }else{
    out = rep(0, n)

    indOrderProb = order(p, decreasing = TRUE) # index
    pOrdered = p[indOrderProb] # rearanjam valorile probabilitatilor
    xOrdered = x[indOrderProb] # rearanjam valorile lui x

    pOrderedCS = cumsum(pOrdered)

    for (i in 1:n){
      u = runif(1)

      k = min(which(u<=pOrderedCS))
      out[i] = xOrdered[k]
    }
  }

  return(out)
}
```

și pentru a o putea testa să considerăm cazul repartițiilor Poisson și Geometrică:

a) Poisson

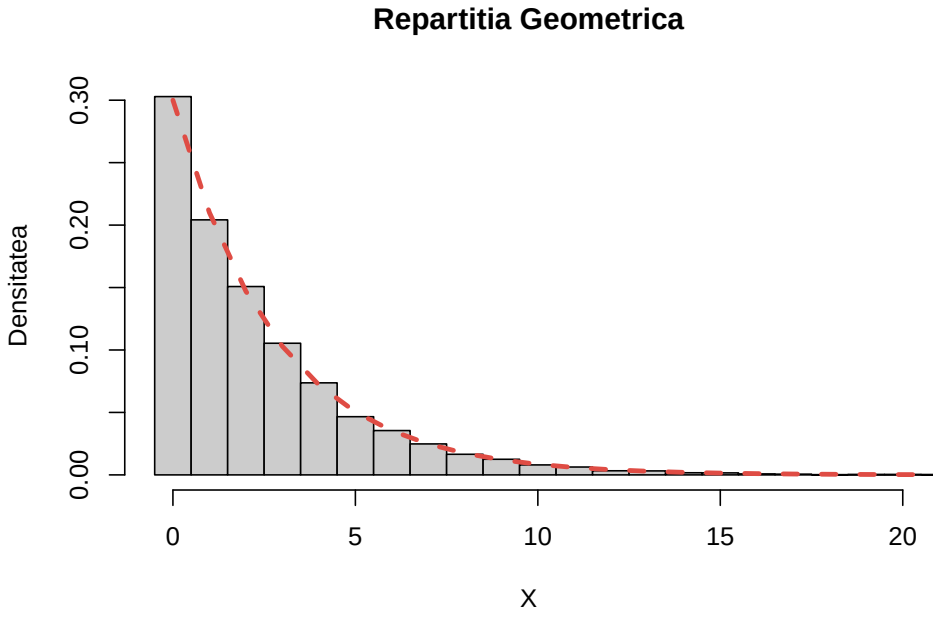
```
# Poisson
hist(GenerateDiscrete(10000, x = 0:50,
                      p = dpois(0:50, 5)),
```

```
     probability = TRUE,
     breaks = seq(-0.5,49.5, by = 1),
     xlim = c(-0.5, 20),
     col = "grey80",
     main = "Repartitia Poisson",
     xlab = "X",
     ylab = "Densitatea")

lines(0:50,
      dpois(0:50, 5),
      type = "l",
      col = myred, lty = 2, lwd = 3)
```



b) Geometrică

```
# Geometric
hist(GenerateDiscrete(10000, x = 0:100,
                      p = dgeom(0:100, 0.3)),
     probability = TRUE,
     breaks = seq(-0.5,99.5, by = 1),
     xlim = c(-0.5, 20),
     col = "grey80",
     main = "Repartitia Geometrica",
     xlab = "X",
     ylab = "Densitatea")

lines(0:100,
      dgeom(0:100, 0.3),
      type = "l",
      col = myred, lty = 2, lwd = 3)
```



2 Generarea unei variabile aleatoare folosind metoda inversă

Scrieți un program care să folosească metoda transformării inverse pentru a genera n observații din densitatea

$$f(x) = \begin{cases} \frac{1}{x^2}, & x \geq 1 \\ 0, & \text{altfel} \end{cases}$$

Testați programul trasând o histogramă a 10000 de observații aleatoare împreună cu densitatea teoretică f.

Primul pas este să determinăm funcția de repartiție F corespunzătoare acestei densități. Pentru $x < 1$ avem că $f(x) = 0$ deci $F(x) = 0$ iar pentru $x \geq 1$ avem

$$F(x) = \int_1^x \frac{1}{t^2}\, dt = 1 - \frac{1}{x}.$$

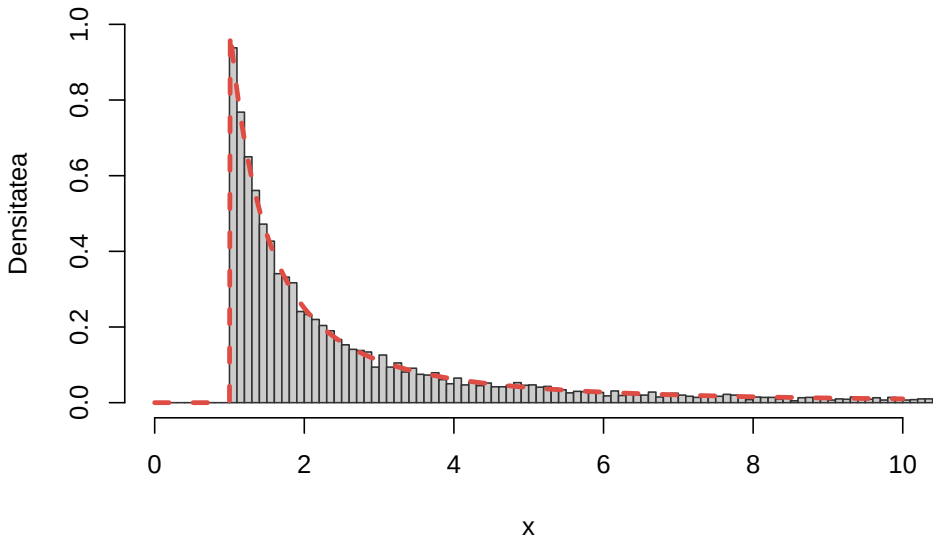
Cum F este continuă putem să determinăm F^{-1} rezolvând ecuația $F(x) = u$. Un calcul direct conduce la $F^{-1}(u) = \frac{1}{1-u}$ iar conform rezultatului văzut la curs concluzionăm că $X = \frac{1}{1-U}$ cu $U \sim \mathcal{U}([0,1])$.

Astfel putem simula un eșantion de talie n din populația f construind funcția

```
GenerateSampleX = function(n){
  u = runif(n)
  return(1/(1-u))
}
```

Pentru a testa comparăm valorile simulate cu densitatea teoretică

```
# simulate
x = GenerateSampleX(10000)
hist(x, freq=FALSE, breaks=seq(0, max(x)+1, 0.1),
     xlim=c(0,10), ylim=c(0,1),
     ylab = "Densitatea",
     main=NULL, col="gray80", border="gray20")

# densitatea teoretica
y = seq(0, 10, 0.01)
f = ifelse(y <= 1, 0, 1/y^2)
lines(y, f, col = myred, lty = 2, lwd = 3)
```



.enumienumi.

Folosind metoda inversă, simulați 10000 de observații independente din repartiția $\mathcal{E}(\lambda)$, cu $\lambda = 1$. Trasați histograma acestui eșantion și comparați cu densitatea repartiției.

2. Fie $X_1, \ldots, X_n$, n variabile aleatoare i.i.d. repartizate $\mathcal{E}(\lambda)$. Atunci variabila aleatoare $S_n = X_1 + X_2 + \cdots + X_n$ este repartizată $\Gamma(n, \lambda)$. Plecând de la acest rezultat, generați 10000 de observații din repartiția $\Gamma(n, \lambda)$ $(n = 10)$. Trasați histograma acestui eșantion și comparați cu densitatea legii.
3. Dacă definim $N = \sup\{n \geq 1 \,|\, S_n \leq 1\}$ (folosim convenția $N = 0$ dacă $S_1 > 1$), atunci N este repartizată Poisson $Pois(\lambda)$. Plecând de la acest rezultat, generați 10000 de observații independente din repartiția $Pois(\lambda)$, trasați histograma acestui eșantion și comparați cu repartiția teoretică.

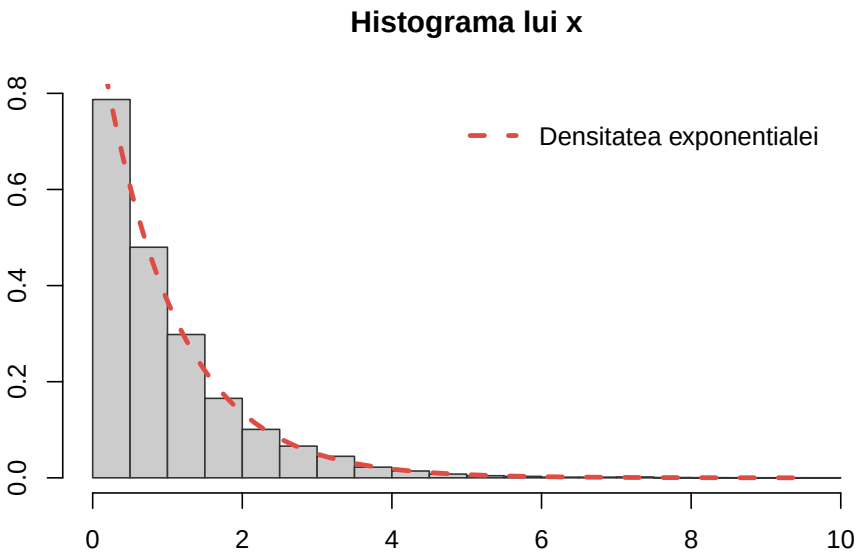
1. Metoda funcției inverse conduce la $-\frac{\log(U)}{\lambda} \sim \mathcal{E}(\lambda)$, cu $U \sim \mathcal{U}([0,1])$.

```
sim.exp = function(n, lambda = 1){
  return(-log(runif(n))/lambda)
}
```

```
n = 10000

x = sim.exp(n)
```

Pentru trasarea histogramei avem:

```
hist(x, freq = FALSE,
     col="gray80",
     border="gray20",
     main = "Histograma lui x",
     ylab = "", xlab = "")

t = seq(0, max(x), 0.01)
lines(t, dexp(t),
      col = myred,
      lty = 2, lwd = 3)
legend("topright",
       "Densitatea exponentialei",
       box.lty = 0,
       col = myred,
       lty = 2, lwd = 3, inset = 0.05)
```



Să observăm că în R putem folosi funcția `rexp(n, rate = ...)` pentru generarea de observații repartizate exponențial.

2. Pentru generarea a m variabile aleatoare i.i.d. repartizate $\Gamma(n, \lambda)$, simulăm $m \times n$ variabile repartizate exponențial pe care le stocăm într-o matrice cu m linii și n coloane. Suma elementelor de pe o linie reprezintă o realizate a repartiției $\Gamma(n, \lambda)$.

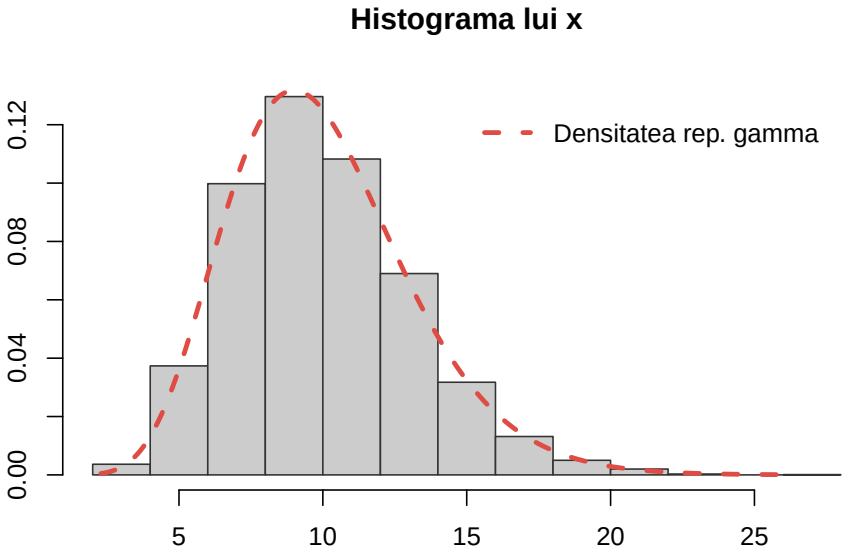
```
sim.gamma = function(m, n, lambda = 1){
  out = rowSums(matrix(sim.exp(m * n, lambda), m, n))
  return(out)
}
```

Din punct de vedere al costului de memorie, această metodă poate să nu fie optimă. Putem folosi o abordare mai simplă folosind bucla `for`:

```
sim.gamma2 = function(m, n, lambda = 1){
  out = numeric(m)

  for (i in 1:m){
    out[i] = sum(sim.exp(n, lambda))
  }

  return(out)
}
```

Pentru verificare avem

```
n = 10
m = 10000

x = sim.gamma(m, n)
t = seq(min(x), max(x), 0.01)

hist(x, freq = FALSE,
     col="gray80",
     border="gray20",
     main = "Histograma lui x", ylab = "",
     xlab = "")

lines(t, dgamma(t, n),
      col = myred, lwd = 3, lty = 2)
legend("topright",
       "Densitatea rep. gamma",
       box.lty = 0,
       col = myred,
       lty = 2, lwd = 3, inset = 0.05)
```



3. Următoarea funcție generează observații repartizate Poisson conform enunțului:

```
sim.pois1 = function(n, lambda = 1){
  out = numeric(n)

  for (i in 1:n){
    out[i] = 0
    s = -log(runif(1))/lambda
    while(s<=1){
      s = s - log(runif(1))/lambda
      out[i] = out[i] + 1
    }
  }

  return(out)
}
```

Această funcție ar putea fi costisitoare în ceea ce privește timpul de execuție (R nu este foarte eficient atunci când avem bucle imbricate). Avem următoare funcție:

```
sim.pois2 = function(n, lambda = 1){
  x = sim.exp(n, lambda)
  i = 0
  l = which(x<=1)
  # realizari cu valoarea 0
  out = rep(0, sum(x > 1))

  while(length(l) > 0){
    i = i + 1
    x = x[l] + sim.exp(length(l), lambda)
    l = which(x<=1)
    # realizari cu valoarea i
    out = c(out, rep(i, sum(x > 1)))
```

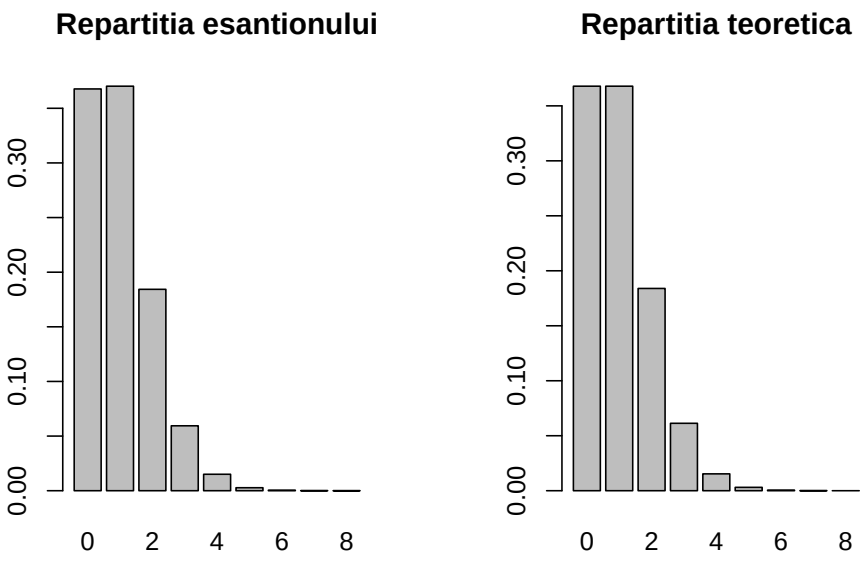
```
  }

  return(out)
}
```

Pentru a compara cele două funcții:

```
n = 100000

start = proc.time()
y = sim.pois1(n)
proc.time() - start
   user  system elapsed
   0.65    0.00    0.81

start = proc.time()
x = sim.pois2(n)
proc.time() - start
   user  system elapsed
   0.03    0.00    0.03
```

Eșantionul generat este repartizat conform repartiției Poisson de parametru λ:

```
x.freq = table(x)/n

par(mfrow = c(1,2))

barplot(x.freq,
        main = "Repartitia esantionului")

barplot(dpois(0:max(x), 1), names.arg = 0:max(x),
        main = "Repartitia teoretica")
```



3 Generarea unei variabile aleatoare folosind metoda respingerii

Fie f densitatea de repartiție definită prin

$$f(x) = \frac{2}{\pi}\sqrt{1-x^2}\mathbf{1}_{[-1,1]}(x)$$

1. Folosiți metoda respingerii pentru a genera 10000 de observații repartizate cu densitatea f.
2. Trasați histograma acestui eșantion și comparați cu densitatea f.

1. Observăm că

$$f(x) \leq \frac{2}{\pi}\mathbf{1}_{[-1,1]}(x) = \frac{4}{\pi}g(x)$$

unde $g(x) = \frac{1}{2}\mathbf{1}_{[-1,1]}(x)$ este densitatea repartiției uniforme pe $[-1,1]$. Astfel metoda respingerii sugerează următorul algoritm:

```
f = function(x){
  return(2/pi * sqrt(1-x^2)*(abs(x) <= 1))
}

sim.resp1 = function(n){
  x = rep(0, n)

  for (i in 1:n){
    # generam obs din g
    x[i] = runif(1, -1, 1)

    # generam uniforma
    u = runif(1)

    while(u > pi * f(x[i])/2){
      x[i] = runif(1, -1, 1)
      u = runif(1)
    }
  }

  return(x)
}
```

Putem îmbunătății codul de mai sus (vrem să evităm să avem și bucla `for` și bucla `while` imbricate) dacă ținem cont de faptul că probabilitatea de acceptare este $p = \frac{\pi}{4}$ iar pentru a genera un eșantion de m observații avem nevoie, în medie, de $\frac{m}{p}$ simulări. Avem

```
sim.resp2 = function(n){
  out = c() # esantionul final

  # cate obs mai avem de generat
  m = n

  # cat timp nu avem esantionul de talia dorita
  # continuam procedeul
  while (m>0){
```

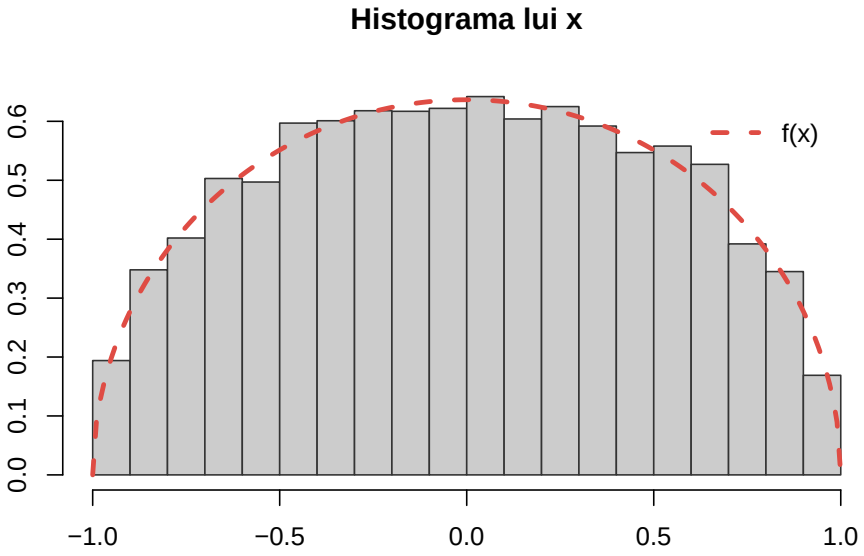
```
    # simulam m/p observatii
    x = runif((4*m)%/%pi + 1, -1, 1)
    u = runif((4*m)%/%pi + 1)

    # testam care obs sunt acceptate
    y = (u <= pi * f(x)/2)

    # pastram doar punctele acceptate
    out = c(out, x[which(y)])
    m = n - length(out)
  }

  return(out[1:n])
}
```

Putem compara cele două metode, în funcție de timpul de execuție:

```
n = 10000

# Metoda 1
star = proc.time()
x = sim.resp1(n)
proc.time() - start
   user  system elapsed
   0.35    0.03    0.58

# Metoda 2
star = proc.time()
x = sim.resp2(n)
proc.time() - start
   user  system elapsed
   0.38    0.03    0.62
```

2. Putem valida algoritmul propus prin metoda respingerii trasând histograma eșantionului:

```
hist(x, freq = FALSE,
     col="gray80",
     border="gray20",
     main = "Histograma lui x", ylab = "",
     xlab = "")

t = seq(-1, 1, 0.01)
lines(t, f(t),
      col = myred, lwd = 3, lty = 2)
legend("topright",
       "f(x)",
       box.lty = 0,
       col = myred,
       lty = 2, lwd = 3, inset = 0.05)
```



Plecând cu o propunere de tip $Exp(\lambda)$ vrem să generăm, cu ajutorul metodei acceptării-respingerii, un eșantion din următoarea densitate (jumătate de normală):

$$f(x) = \begin{cases} \frac{2}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}, & \text{dacă } x \geq 0 \\ 0, & \text{altfel} \end{cases}$$

Fie g densitatea repartiției exponențiale de parametru λ,

$$g(x) = \begin{cases} \lambda e^{-\lambda x}, & \text{dacă } x \geq 0 \\ 0, & \text{altfel} \end{cases}$$

Pentru a aplica algoritmul de acceptare-respingere trebuie să găsim valoarea lui $c > 0$ pentru care $f(x) \leq cg(x)$ pentru toate valorile $x \in \mathbb{R}$. Pentru $x \geq 0$ avem

$$\frac{f(x)}{g(x)} = \frac{2}{\lambda\sqrt{2\pi}} e^{-\frac{x^2}{2}+\lambda x}$$

și cum funcția $-\frac{x^2}{2} + \lambda x$ își atinge valoarea maximă în punctul $x = \lambda$ rezultă că

$$\frac{f(x)}{g(x)} \leq c^*, \ \forall x \geq 0$$

unde

$$c^* = \sqrt{\frac{2}{\pi\lambda^2}} e^{\lambda^2/2}.$$

Astfel algoritmul devine:

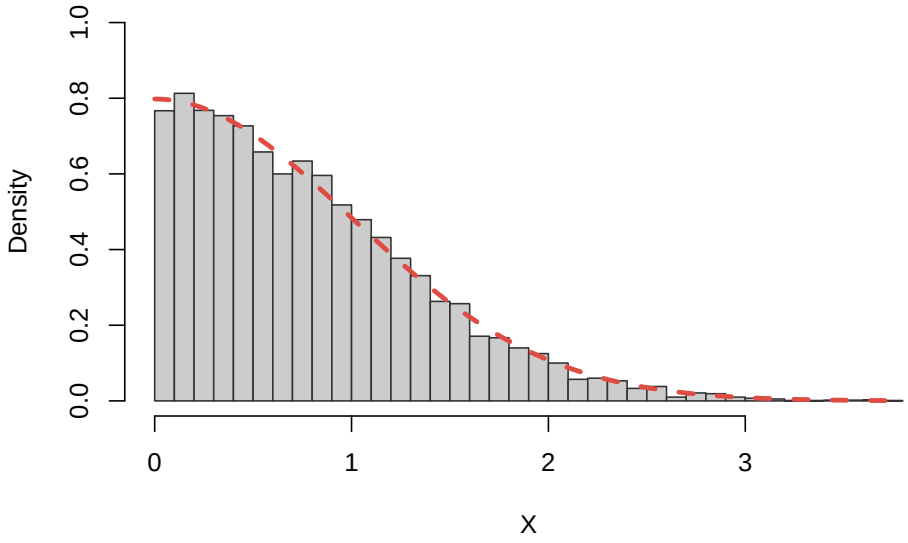
- pentru $n = 1, 2, \ldots$
- generează $X_n \sim Exp(\lambda)$
- generează $U_n \sim \mathcal{U}[0, 1]$
- dacă $U_n \leq \exp\left(-\frac{1}{2}(X_n - \lambda)^2\right)$ atunci
- intoarceți X_n

Avem funcția:

```
# generarea puntelor din densitatea f

f <- function(x) {
  return((x> 0) * 2 * dnorm(x,0,1))
}

g <- function(x) { return(dexp(x,1)) }

c <- sqrt(2 * exp(1) / pi)

rhalfnormal <- function(n) {
  res <- numeric(length=n)
  i <- 0
  while (i<n) {
    U <- runif(1, 0, 1)
    X <- rexp(1, 1)
    if (c * g(X) * U <= f(X)) {
      i <- i+1
      res[i] <- X;
    }
  }
  return(res)
}
```

Testăm

```
X <- rhalfnormal(10000)

hist(X,
     breaks=50,
     prob=TRUE,
     ylim=c(0,1),
     main=NULL,
     col="gray80",
     border="gray20")

curve(f, min(X), max(X), n=500, col = myred, lty = 2, lwd = 3, add=TRUE)
```



Modificați codul de la exercițiul precedent pentru a simula un eșantion dintr-o normală standard.

Cum f (din problema 1) este densitatea unei normale standard $X \sim \mathcal{N}(0,1)$ condiționată la $X > 0$ și cum densitatea normală este simetrică față de medie (0 în acest caz) algoritmul se modifică acceptând x_n și $-X_n$ cu probabilitatea de 0.5.

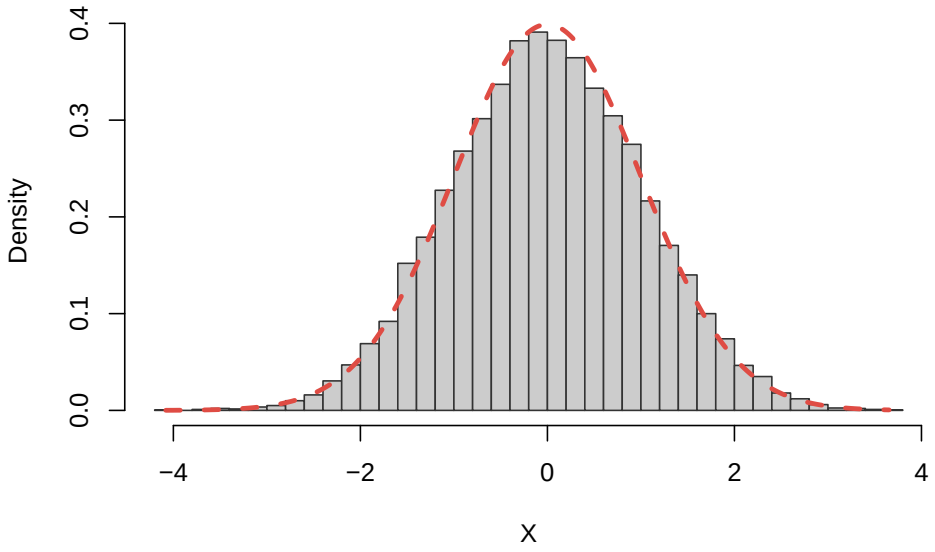
Astfel avem funcția:

```
f2 <- function(x) {
  return(dnorm(x,0,1))
}

normal1 <- function(n) {
  res <- numeric(length=n)
  i <- 0
  while (i<n) {
    U <- runif(1, 0, 1)
    X <- rexp(1, 1)
    if (c * g(X) * U <= f(X)) {
      i <- i+1

      res[i] <- ifelse(runif(1) <= 0.5, X, -X);
    }
  }
  return(res)
}
```

si testul

```
X <- normal1(10000)

hist(X, breaks=50,
```

```
     prob=TRUE,
     main=NULL,
     col="gray80", border="gray20")

curve(f2, min(X), max(X), n=500,col = myred, lty = 2, lwd = 3, add=TRUE)
```



4 Simularea unei uniforme pe disc

Considerăm pătratul $C = [0, L]^2$ și discul D de centru $(\frac{L}{2}, \frac{L}{2})$ și rază $\frac{L}{2}$. Considerăm șirul de v.a. $(Y_n)_{n\geq 1}$ pe $\mathbb{R}^2$ i.i.d. repartizate uniform pe pătratul C.

1. Aproximați valoarea lui π prin ajutorul numărului de puncte Y_n care cad în interiorul discului D (Metoda respingerii)
2. Simulați n puncte uniforme pe disc.

1. Definim v.a. $X_n = \mathbf{1}_{\{Y_n \in D\}}$, $n \geq 1$, care formează un șir de v.a. i.i.d. de lege $\mathcal{B}(\mathbb{P}(Y_n \in D))$, deoarece $(Y_n)_{n\geq 1}$ este un șir de v.a. i.i.d. repartizate uniform pe C, $\mathcal{U}(C)$. Din *Legea Numerelor Mari* avem că

$$\frac{1}{n}\sum_{i=1}^{n} X_i \xrightarrow{a.s.} \mathbb{E}[X_1] = \mathbb{P}(Y_1 \in D),$$

prin urmare trebuie să calculăm probabilitatea $\mathbb{P}(Y_1 \in D)$. Știm că densitatea v.a. Y_1 este dată de $f_{Y_1}(x, y) = \frac{1}{\mathcal{A}(C)}\mathbf{1}_C(x, y)$ de unde

$$\mathbb{P}(Y_1 \in D) = \iint_D f_{Y_1}(x,y)\,dxdy = \iint \mathbf{1}_D(x,y)\mathbf{1}_C(x,y)\,dxdy$$
$$= \frac{1}{\mathcal{A}(C)} \iint \mathbf{1}_D(x,y)\,dxdy = \frac{\mathcal{A}(D)}{\mathcal{A}(C)} = \frac{\pi\frac{L^2}{4}}{L^2} = \frac{\pi}{4}.$$

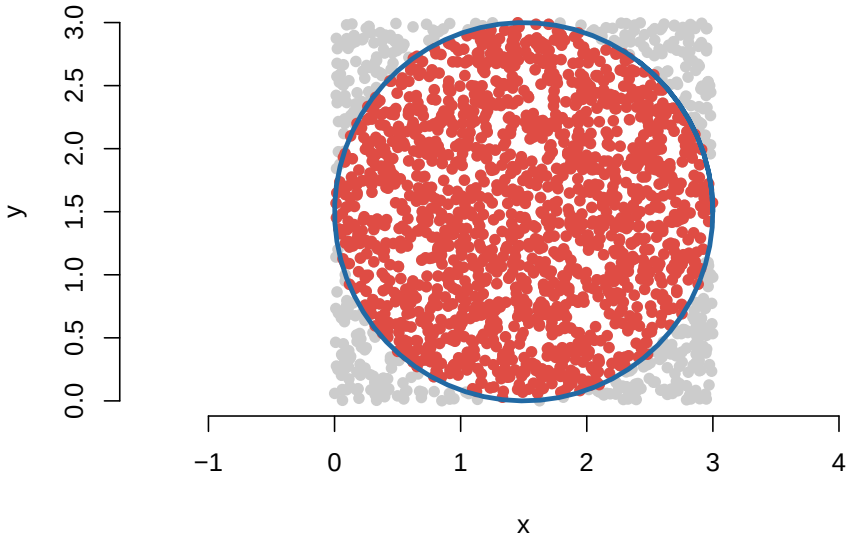
Astfel, putem estima valoarea lui π prin $\frac{4}{n}\sum_{i=1}^{n} X_i$ pentru valori mari ale lui n.

```
# Estimam valoarea lui pi

L = 3 # lungimea laturii patratului
R = L/2 # raza cercului inscris

n = 2000 # numarul de puncte din patratul C
# generam puncte uniforme in C
x = L*runif(n)
y = L*runif(n)

# metoda respingerii (rejectiei)
l = (x-R)^2+(y-R)^2 # distanta dintre centrul cercului si punct
ind = l<=(R)^2 # indicii pentru care distanta este mai mica sau egala cu R

xc = x[ind] # coordonatele punctelor din interiorul cercului
yc = y[ind]

estimate_pi = 4*sum(ind)/n # estimarea lui pi
err = abs(estimate_pi-pi) # eroarea absoluta
```

Aplicând acest procedeu obținem că valoarea estimată a lui π prin generarea a $n = 2000$ puncte este 3.156 iar eroarea absoluta este 0.01441.

2. Una dintre metodele prin care putem simula puncte uniform repartizate pe suprafața discului D este *Metoda respingerii*. Această metodă consistă în generarea de v.a. Y_n repartizate uniform pe suprafața pătratului C, urmând ca apoi să testăm dacă Y_n aparține discului D (deoarece $D \subset C$). Dacă da, atunci le păstrăm dacă nu atunci mai generăm. Următoarea figură ilustrează această metodă:

```
# figura
theta = seq(0, 2*pi+1, by = 0.1)
xd = R+R*cos(theta)
yd = R+R*sin(theta)

plot(x, y,
     col = "grey80", pch = 16,
     asp = 1,
     xlim = c(0,3), ylim = c(0,3),
     bty = "n")

points(xc, yc, col = myred, pch = 16)
lines(xd, yd, col = myblue, lwd = 3)
```



Vom da mai jos o altă metodă de simulare a punctelor distribuite uniform pe discul D de rază L. O primă idee ar fi să generăm cuplul de v.a. (X_1, Y_1) așa încât $X_1, Y_1 \sim \mathcal{U}([0, L])$ și ele să fie independente (ceea ce nu este adevărat în realitate). Vom vedea (printr-o ilustrație grafică) că această abordare este greșită (punctele sunt concentrate în centrul cercului).

O altă abordare este următoarea. Căutăm să simulăm un cuplu de v.a. (X, Y) care este uniform distribuit pe suprafața discului D, i.e. densitatea cuplului este dată de $f_{(X,Y)}(x, y) = \frac{1}{\pi L^2}\mathbf{1}_D(x, y)$. Considerăm schimbarea de variablile în coordonate polare: $x = r\cos(\theta)$ și $y = r\sin(\theta)$. Obiectivul este de a găsi densitatea variabilelor R și Θ.

Fie $g(x, y) = \left(\sqrt{x^2 + y^2}, \arctan(y/x)\right) = (r, \theta)$, transformarea pentru care avem $(R, \Theta) = g(X, Y)$. Știm că inversa acestei transformări este $g^{-1}(r, \theta) = (r\cos(\theta), r\sin(\theta))$, prin urmare

$$\begin{aligned} f_{(R,\Theta)}(r, \theta) &= f_{(X,Y)}\left(g^{-1}(r, \theta)\right) |\det(J_{g^{-1}}(r, \theta))| \\ &= \frac{1}{\pi L^2}\mathbf{1}_D(r\cos(\theta), r\sin(\theta)) \begin{vmatrix} \cos(\theta) & \sin(\theta) \\ r\sin(\theta) & -r\cos(\theta) \end{vmatrix} \\ &= \frac{1}{\pi L^2}\mathbf{1}_{[0,L]}(r)\mathbf{1}_{[0,2\pi]}(\theta) r. \end{aligned}$$

Observăm că densitatea (marginală) v.a. Θ este

$$\begin{aligned} f_\Theta(\theta) &= \int f_{(R,\Theta)}(r, \theta)\, dr = \mathbf{1}_{[0,2\pi]}(\theta) \int \frac{r}{\pi L^2}\mathbf{1}_{[0,L]}(r)\, d\theta \\ &= \frac{1}{\pi L^2}\mathbf{1}_{[0,2\pi]}(\theta)\frac{L^2}{2} = \frac{1}{2\pi}\mathbf{1}_{[0,2\pi]}(\theta), \end{aligned}$$

iar densitatea v.a. R este

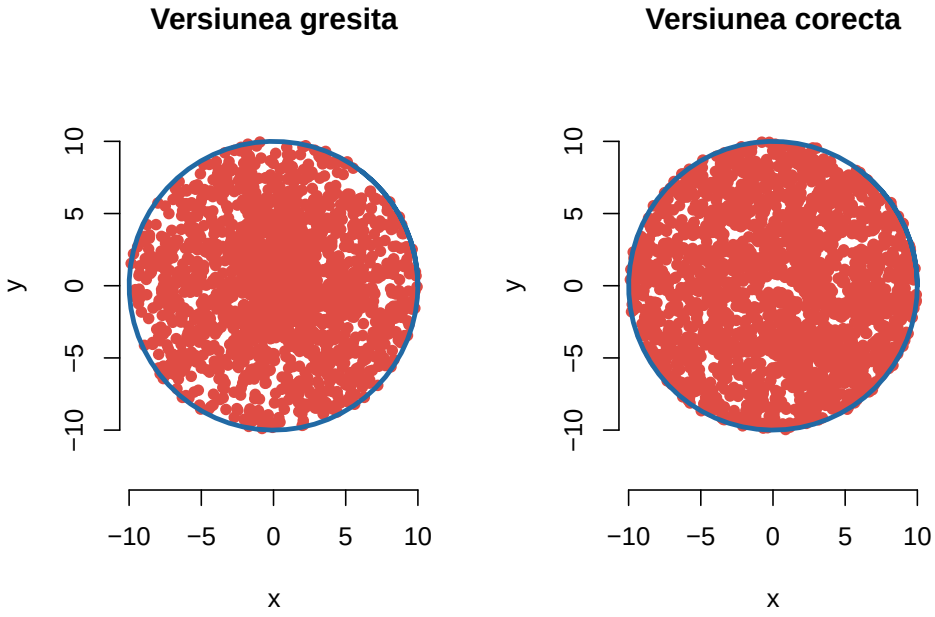
$$f_R(r) = \int f_{(R,\Theta)}(r,\theta)\,d\theta = \frac{r}{\pi L^2}\mathbf{1}_{[0,L]}(r)\int_0^{2\pi} d\theta$$
$$= \frac{r}{\pi L^2}\mathbf{1}_{[0,L]}(r)2\pi = \frac{2r}{L^2}\mathbf{1}_{[0,L]}(r).$$

Din expresiile de mai sus putem observa că Θ este o v.a. repartizată uniform pe $[0,2\pi]$ și putem verifica ușor că legea v.a. R este aceeași cu cea a v.a. $L\sqrt{U}$ unde $U \sim \mathcal{U}([0,1])$.

Astfel pentru simularea unui punct (X,Y) uniform pe D este suficient să simulăm o v.a. Θ uniform pe $[0,2\pi]$ și o v.a. U uniformă pe $[0,1]$ și să luăm $X = L\sqrt{U}\cos(\Theta)$ și $Y = L\sqrt{U}\sin(\Theta)$.

Următorul cod ne ilustrează cele două proceduri prezentate:

```
# rm(list=ls())

n = 2000;# numarul de puncte

R = 10;# raza cercului

theta = 2*pi*runif(n);# theta este uniforma pe [0,2*pi]

# versiunea gresita - r este uniforme pe [0,R]
r1 = R*runif(n);

x1 = r1*cos(theta);# coordonate polare
y1 = r1*sin(theta);

# versiunea corecta
r2 = R*sqrt(runif(n));

x2 = r2*cos(theta);# coordonate polare
y2 = r2*sin(theta);

# schimbarea de variabila in coordonate polare: cercul
theta2 = seq(0,2*pi+1,by=0.1)
xc = R*cos(theta2);
yc = R*sin(theta2);

# graficul

par(mfrow = c(1,2))

plot(x1, y1,
     ylim = c(-11, 11),
     col = myred, pch = 16,
     main = "Versiunea gresita", xlab = "x", ylab = "y", asp = 1, bty = "n")
lines(xc, yc, lwd = 3, col = myblue)

plot(x2, y2,
     ylim = c(-11, 11),
     col = myred, pch = 16,
     main = "Versiunea corecta", xlab = "x", ylab = "y", asp = 1, bty = "n")
lines(xc, yc, lwd = 3, col = myblue)
```


Versiunea gresita
Versiunea corecta
y
x
−10
−5
0
5
10