CURS 5 - GENERAREA VARIABILELOR ALEATOARE

Scop: Generarea de variabile aleatoare

- discrete – iau un numar finit de valori

- continue – date de densitati de repartitie

METODA TRANSFORMARII INVERSE

- Trebuie sa cunoastem:
    - Variabila distribuita uniform U(0,1)
    - Functa de distributie a variabilei aleatoare pe care va urma sa o generam
- Metoda Transformarii Inverse
    - Definitie: $X=F^{-1}(U)$ (F^{-1} este inversa lui F)
    - Se face transformarea:
        - $F(F^{-1}(U)) = U = F(X)$
        - $F^{-1}(U)$ este acea valoare a lui X pentru care $F(X)=U$

Metoda transformarii inverse:

1. Se gaseste formula pentru F^{-1}.
2. Se genereaza un numar aleator U
3. Se detrmina numarul aleator $X=F^{-1}(U)$

Exemplu. Sa se genereze o variabila aleatoare X cu functia de repartitie $F(x)=x^n$, 0<x<1.

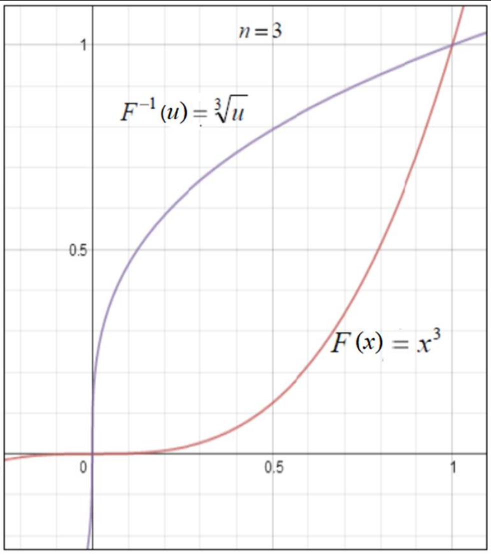
1. Se gaseste formula pentru F^{-1}:

$$F(x)= x^n = u \leftrightarrow x = \sqrt[n]{u} = F^{-1}(u).$$

2. Se formuleaza algoritmul de generare a variabilei aleatoare X:

- Se genereaza numarul aleator U
- Se calculeaza $X = \sqrt[n]{U}$.

Obs. Metoda se poate aplica numai pentru functii de repartitie inversabile. Deci, metoda nu se poate aplica pentru repartitia normala.



METODA ACCEPTARII – RESPINGERII

Exemplu.

- *Sa se dea algoritmul de respingere care genereaza variabila aleatoare X pentru care:*
  $f(x)=20x(1-x)^3$, $0<x<1$.
- Folositi $g(x)=1$, $0<x<1$.

Solutie.

1. Se genereaza procedura de respingere

Se determina cel mai mic *c* astfel incat *f(x)/g(x)*≤*c*, astfel:

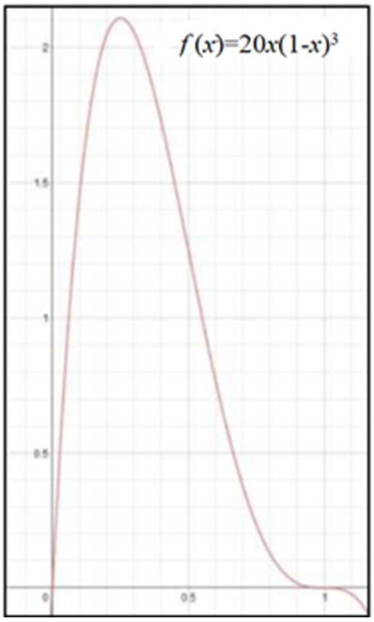
1.1. Se determina punctual de maxim pentrut *f(x)/g(x)*= *20x(1-x)*^{3}

1.1.1. Se determina derivata de ordin 1 a lui *f(x)/g(x)*:

$$\left(\frac{f(x)}{g(x)}\right)' = 20[(1-x)^3 - 3x(1-x)^2] = 20(1-x)^2(1-4x)$$

1.1.2. Se determina punctele critice prin rezolvarea ecuatiei: $\left(\frac{f(x)}{g(x)}\right)' = 0$. Rezulta *x*=1/4.

1.1.3. Se arata ca *x*=1/4 este punct de maxim.



1.2. Se determina cel mai mic c

1.2.1. $\frac{f(x)}{g(x)} \leq 20 \times \frac{1}{4} \times \left(1-\frac{1}{4}\right)^3 = \frac{135}{64} = c\ min$

1.2.2. $\frac{f(x)}{cg(x)} = \frac{256}{27}x(1-x)^3$,

1.3. Se formuleaza algoritmul de respingere

1.3.1. Se genereaza numeral U_1 *si* U_2

1.3.2. *Daca* $U_2 \leq \frac{256}{27}U_1(1-U_1)^3$ *stop si pune* $X = U_1$, altfel mergi la pasul 2.1.

- Numarul mediu de iteratii este chiar c=135/64=2.11.

GENERAREA UNOR VARIABILE ALEATOARE CU REPARTITII IMPORTANTE

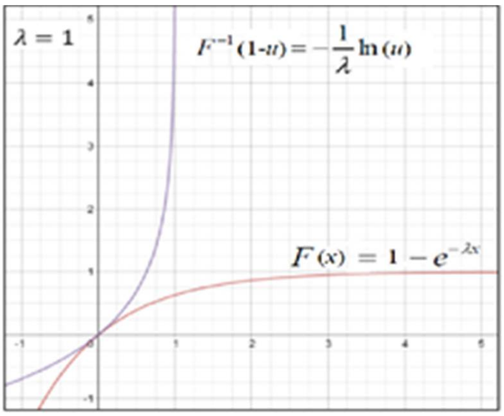
➢ *EXPONENTIALA:*

- *densitatea de repartitie exponentiala este* $f(x) = \lambda e^{-\lambda x}$, iar functia de repartitie este $F(x) = 1 - e^{-\lambda x}$.

- se aplica metoda transformarii inverse

$$\begin{aligned} F(x) = \quad & 1 - e^{-\lambda x} = u \\ \leftrightarrow \quad & 1 - u = e^{-\lambda x} \\ \leftrightarrow \quad & \ln(1-u) = -\lambda x \\ \leftrightarrow \quad & -\frac{1}{\lambda} * \ln(1-u) = x \ = F^{-1}(u) \\ \leftrightarrow \quad & -\frac{1}{\lambda} * \ln(u) = x \ = F^{-1}(1-u) \end{aligned}$$

Daca u este uniform distribuit pe (0,1), atunci si 1-u este uniform distribuit pe (0,1)



Se formuleaza algoritmul de generare a variabilei aleatoare *X:*

- *Se genereaza un numar aleator U*
- *Se calculeaza X*=(1/λ) *lnU*

➢ *NORMALA:*

Sa se dea algoritmul de respingere care genereaza un sir de variabile normal repartizate X_i

cu medie μ si varianta σ.

Abordare:

- Se genereaza variabile aleatoare semi-normale si se determina semnul fiecareia.

  Densitatea de repartitie este $f(x) = \frac{2}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$, 0<*x*<∞.
- Se optimizeaza algoritmul si se verifica eficienta sa.
- Se aplica metoda acceptarii-respingerii, folosind $g(x) = e^{-x}$.

I. Generarea procedurii de respingere

1. Se determina cel mai mic *c* astfel ca *f(x)/g(x)≤c*, astel:
   1.1. Se determina punctual de maxim pentru *f(x)/g(x)*:
      1.1.1. $\frac{f(x)}{g(x)} = \sqrt{\frac{2}{\pi}} e^{x-\frac{x^2}{2}}$ isi atimge maximul unde $x - \frac{x^2}{2}$ is atinge maximul.
      1.1.2. $\left(x - \frac{x^2}{2}\right)' = 1 - x$ →x=1 este valoare critica
      1.1.3. Se verifica faptul ca *x*=1 este punct de maxim.
   1.2. Se determina cel mai mic *c*
      1.2.1. max(*f(x)/g*(*x*))= $\sqrt{\frac{2}{\pi}} e^{1-\frac{1^2}{2}} = \sqrt{\frac{2e}{\pi}}$ = cel mai mic *c*
      1.2.2. $\frac{f(x)}{cg(x)} = \sqrt{\frac{2}{\pi}} e^{x-\frac{x^2}{2}} \sqrt{\frac{\pi}{2e}} = e^{-1+x-\frac{x^2}{2}} = e^{-\frac{x^2-2x+1}{2}} = e^{-\frac{(x-1)^2}{2}}$.

II. Formularea algoritmului de respingere

0. *i* =1.
1. Se genereaza o variabila aleatoare exponentiala cu rata 1, numita Y_1.
2. Se genereaza un numar aleator U_1,
3. Daca $U_1 \le e^{-\frac{(Y_1-1)^2}{2}}$, stop si fie $Y_2 = Y_1$,

   In caz contrar mergi la pasul 1.
4. Se genereaza un numar aleator U_2 si se defineste $X_i = \mu + \sigma \times \begin{cases} Y_2, & U_2 \le 0.5 \\ -Y_2, & U_2 > 0.5 \end{cases}$.
5. *i*=*i*+*1* si se merge la pasul 1.

III. Optimizare

➢ Transformare:

$$U \le e^{-\frac{(Y_1-1)^2}{2}} \leftrightarrow -\ln(U) \ge -\frac{(Y_1-1)^2}{2} \leftrightarrow Y_2 \ge \frac{(Y_1-1)^2}{2} \leftrightarrow Y_2 - \frac{(Y_1-1)^2}{2} \ge 0$$

(s-a folosit faptul ca –ln(*U*) este exponential cu rata 1)

➢ Algoritm optimizat:

0. *i* =1.
1. Se genereaza o variabila aleatoare exponentiala cu rata 1, numita Y_1.
2. Se genereaza o variabila aleatoare exponentiala cu rata 1, numita Y_2.
3. Daca $Y_2 - \frac{(Y_1-1)^2}{2} \ge 0$ stop si fie $Y_3 = Y_2 - \frac{(Y_1-1)^2}{2}$

In caz contrar de merge la pasul 1

4. Se genereaza un numar aleator *U* si se defineste $X_i = \mu + \sigma \times \begin{cases} Y_2, & U_2 \le 0.5 \\ -Y_2, & U_2 > 0.5 \end{cases}$,
5. *i*=*i*+*1*, se defineste $Y_1 = Y_3$ si se merge la pasul 2.

IV. Analiza eficientei:

- Numarul mediu de iteratii pentru generarea variabilelor aleatoare exponentiale = 1.64
    - Numarul mediu de iteratii (pasii 1 si 2) este 2*c*=2.64
    - Folosirea exponentialei *Y*: Numarul mediu de iteratii = 2.64 - 1=1.64
- Numarul medii de iteratii la pasul 3: *c* = 1.32.

➢ *PROCESE POISSON OMOGENE*

- *Evenimentele apar la intervale egale*
- *Rata λ*, care reprezinta numarul asteptat de evenimente, este constanta.

Sa notam prin

- *T*= primele *T* unitati
- *T* = timpul
- *I* = numarul de evenimente care apar in timpul *t*
- S(1), …. *S(I)*= timpul evenimentelor, in ordine crescatoare.

Algoritmul de generare este urmatorul (metoda transformarii inverse):

- Se gaseste functia F^{-1} *apoi:* $x = F^{-1}(u) = (-1/\lambda)\ln(u)$
- Se formuleaza algoritmul de generare a procesului Poisson:
    0. t=0, *I*=0
    1. Se genereaza *U*
    2. $T = t + (-1/\lambda)\ln(u) = t - (1/\lambda)\ln(u)$

       Daca *t>T, stop*
    3. *I*=*I*+1; S(t)=t
    4. Mergi la pasul 1.
- Solutia este sirul timpilro evenimetelor S(1), …. *S(I)*

➢ *PROCESE POISSON NEOMOGENE* – optional – ca tematica de proiect

- evenimentele apar random in timp
- rata de sosire variaza in timp
- rata λ, care reprezinta numarul de evenimente asteptate, nu este constanta
- intensitatea λ(t) reprezinta numarul de evenimente asteptate in vecinatatea timpului t

APPLICATII

1. <u>*Generarea unei variabile exponentiale*</u>

O companie de asigurari are 1000 asigurati, fiecare prezentand independent o cerere de despagubire in urmatoarea luna cu o probabilitate de 0.05. Totalul cererilor sunt variabile aleatoare *independente cu media $800. Folositi simularea pentru a estima probabilitatea ca suma acestor cereri de despagubire sa depaseasca $50000.*

Solutie.

- De ce folosim simularea?
  - Media sumei despagubirilor este normal distribuita cu media $40000=1000*0.05*$800
  - Totusi, nu stim media si abatera standard
- Algoritmul de generare a variabilei aleatoare *X*:
  - Se genereaza un numar aleator *U*
  - Se obtine X=-(1/λ)ln(U)

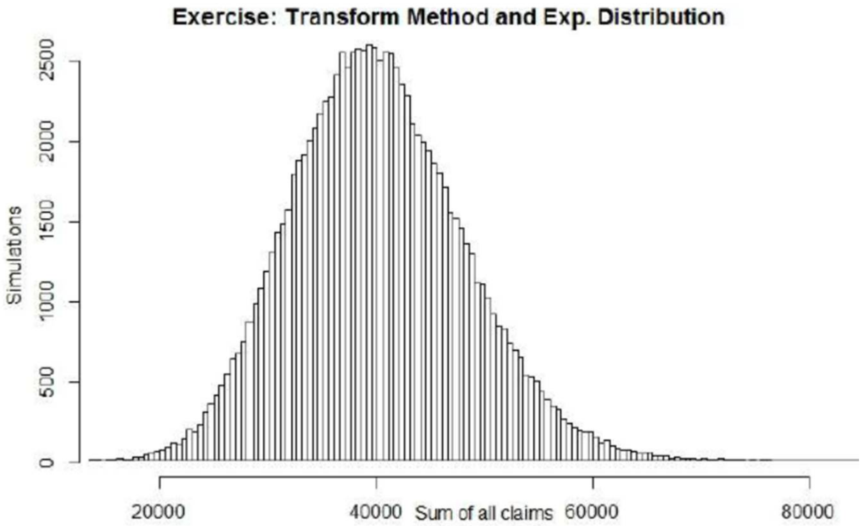
```
### Simularea unei distributii exponentiale prin metoda transformarii inverse - Author: Martin Kretzer
### 0. Functii:
### Simularea si returnarea listei valorilor rezultate
getSimulationsVector<-function( numberOfSimulations, numberOfCustomers, mean, probability )
{
   lambda<-1/mean
   simulationsVector<-c()
   for( i in 1:numberOfSimulations )
   {
      claimsPerSimulationVector<-c()
      #uniformVars<-runif( numberOfCustomers )
      for( j in 1:numberOfCustomers )
      {
         uniformVarProbability<-runif(1)
         if( uniformVarProbability<= probability )
        {
            uniformVarAlgorithm<-runif(1)
            claim<--(1/lambda)*log( uniformVarAlgorithm)
         claimsPerSimulationVector<-c( claimsPerSimulationVector, claim )
        }
      }
      simulationsVector<-c(simulationsVector, sum(claimsPerSimulationVector) )
   }
return(simulationsVector)
};
```

```
### Probabilitatea ca toate valorile sa fie mai mari ca limita
getProbability<-function( simulationsVector, limit )
{
  eventcount<-0
  for( I in 1:length( simulationsVector) )
 {
    if( simulationsVector[[i]] >= limit )
    {
       eventcount<-eventcount+ 1
     }
   }
   probability<-eventcount/ length( simulationsVector)
return(probability)
};

###############
###1. Variabile de intrare:
numberOfSimulations<-100000;
numberOfCustomers<-1000
mean<-800;
limit<-50000;
probability<-0.05;

### 2. Variabile de iesire:
simulationsVector<-getSimulationsVector( numberOfSimulations, numberOfCustomers, mean,
probability );
simulationsVector; #R maximum outputs 10000 values

### 3. Output:
hist(simulationsVector, 250, xlab="Sum of all claims",ylab="Simulations",main="Exercise:
Transform Method and Exp. Distribution");

### 4. Probabilitatea calculata:
probability<-getProbability( simulationsVector, limit );
probability;
```

Output:

- Numar de simulari:100,000

- Probabilitatea estimate ca suma cererilor sa depaseasca \$50,000: p =0.10917



2. *Aplicatie a metodei de respingere-acceptare pentru variabile normale*

Sa se scrie un program care genereaza efficient variabile aleatoare repartizate normal cu media 10 si abaterea standard 3, folosind metoda acceptarii-respingerii. Care esre abaterea standard estimate?

Solutie. Se foloseste algoritmul optimizat de generare a variabilelor aleatoare.

0. i =0
1. Se genereaza o variabila aleatoare exponentiala cu λ = 1, si o numim Y_1
2. Se genereaza o variabila aleatoare exponentiala cu λ = 1, si o numim Y_2
3. Daca $Y_2 - \frac{1}{2}(Y_1 - 1)^2 \geq 0$, stop si inlocuieste $Y_3 = Y_2 - \frac{1}{2}(Y_1 - 1)^2$

 Altfel mergi la pas 1

4. Se genereaza un n umar aleator U si se defineste: $X_i = \mu + \sigma \times \begin{cases} Y_{1,} & U \leq 0.5 \\ -Y_1, & U > 0.5 \end{cases}$
5. Fie i=i+1 si puneti $Y_1 = Y_2$ apoi mergeti la pasul 2.

\### 0. Functia:

\### **Se face simularea si rezulta vectorul tuturor valorilor simulate**

getSimulationsVector<-function(numberOfSimulations,mean,stdDev)

{

\# Se defines variabilele

simulations.vector<-rep(0,numberOfSimulations)

\# Se genereaza variabila exponentiala y1

y1<-runif(1)

```
y1<--log(y1)
for(i in 1:numberOfSimulations)
{

# Se genereaza variabila independentA exponentiala y2
y2<-runif(1)
y2<--log(y2)

# Procedura de acceptare-respingere
while(y2 < (y1-1)^2/2 )
{
# daca se respinge, se genereaza doua noi variabile independente y1 si y2 si se repeta procedura
y1<-runif(1)
y1<--log(y1)
y2<-runif(1)
y2<--log(y2)
}
y3<-y2-(y1-1)^2/2

# Se gebereaza un numar aleator U si se stocheaza variabila simulate y1 in simulations.vector
u<-runif(1)
if( u <= 0.5)
{
simulations.vector[i]<-mean + (y1*stdDev)
}
else
{
simulations.vector[i]<-mean + (-y1*stdDev)
}
y1<-y3
}
return(simulations.vector)
}

### Function End
simulations.vector<-getSimulationsVector(100000, 10, 3)
summary(simulations.vector)
sd(simulations.vector)
```

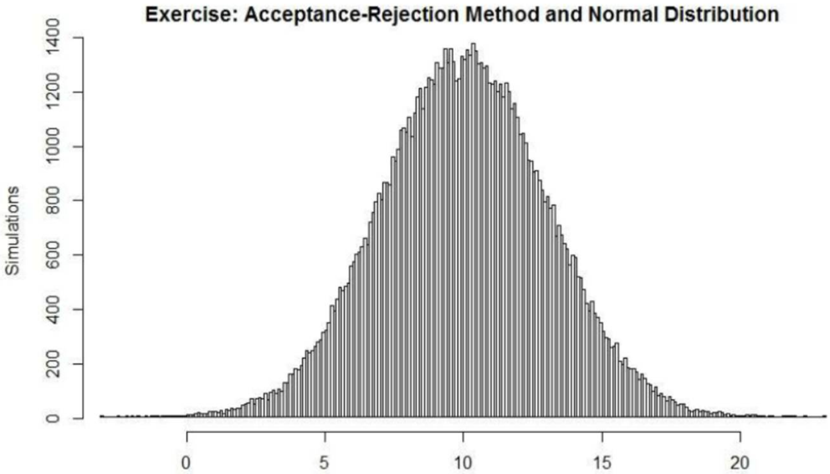
```
hist(simulations.vector, 250, xlab="",ylab="Simulations",main="Exercise: Acceptance-Rejection Method and Normal Distribution")
```

Output

Executed simulations: 100,000

Estimated mean: 10.010

Estimated standard deviation: 2.993299



Referinta:

Martin Kretzer - Generating Continuous Random Variables, University of Mannheim, Business School