Proiect Probabilități și Statistică

Proiectul 1: Implementare Pachet R

Iordache Denisa-Elena

Iuga Paula

Neagoe Tavi

Nedelcu Radu-Andrei

Februarie, 2020

Cuprins

Proiect Probabilități și Statistică.......... 1
Proiectul 1: Implementare Pachet R 1
0. Introducere 1
1. Constanta de normalizare k 2
2. Densitate de probabilitate.......... 3
3. Un obiect R de tip v.a. continuă.......... 4
4. Reprezentarea grafică a densității și a funcției de repartiție 6
8. Afișarea unei fișe de sinteză.......... 6
I.Repartiția exponențială.......... 6
II.Repartiția uniformă.......... 8
III.Repartiția normală (Gaussiană) 11
IV.Repartiția Pareto.......... 14
5. Calculul mediei, dispresiei,.......... 17
momentelor inițiale și centrale până la ordinul 4 17
6. Media și dispersia unei v.a. g(X) 19
9. Calcularea a n valori dintr-o repartiție de v.a. continue.......... 21
10. Calculul covalenței și al coeficientului de covariație pentru 2 v.a. continue 23
11. Densități marginale și condiționate pentru 2 v.a. continue 26
Bibliografie 28

0. Introducere

Variabilele aleatoare continue apar în practică atunci când într-un anumit experiment măsurăm o anumită cantitate, de exemplu lungimea unui șurub, voltajul într-un circuit electric sau timpul dintre doua aterizări. Variabilele aleatoare continue se definesc peste mulțimea variabilelor aleatoare astfel:

Variabila aleatoare X este o variabilă aleatoare continuă dacă funcția de distribuție corespunzătoare **F : R** → **R** este o funcție continuă pe **R**.

Lucrarea de față își propune să realizeze un pachet R care să ofere funcții ce facilitează lucrul cu aceste variabile.

1. Constanta de normalizare k

Pentru o funcție f, introdusă de utilizator se determină o **constantă de normalizare k**.

În teoria probabilităților, o constantă de normalizare este o constantă prin care trebuie înmulțită o funcție non-negativă pe toate ramurile, astfel încât aria de sub graficul său să fie 1. De exemplu, constanta de normalizare se aplică pentru a face din funcția dată o funcție de densitate de probabilitate sau o funcție de masă.

Funcția *constanta* din codul de mai jos calculează o constantă de normalizare pentru o funcție non-negativă f, astfel:

- Se calculează $\int_{-\infty}^{+\infty} f(t)dt$ din funcția dată
- Dacă valoarea acesteia este 0 atunci nu există o astfel de constantă, se afișează mesajul *constanta nu exista* și se iese din funcția de bază cu ajutorul funcției *stop*.
- Altfel, se returnează 1/(valoarea intergralei) determinată la primul pas.

Am folosit **cubintegrate din pachetul cubature** în loc de **integrate** pentru a evita erorile de cod datorate erorilor de calcul. Funcția cubintegrate primește ca parametru o funcție de integrat, limita inferioară și superioară a integralei și methoda. Pentru aceasta am folosit *pcubature* care aplică integrale multivariate peste hipercurbe.

```r
#1
constanta <- function(f)
{
  temp <- cubintegrate(f, lower = -Inf, upper = Inf, method = "pcubature")
  if(temp$integral==0)
    stop("Constanta nu exista")
  return (1/temp$integral)
}

```

Pachetul cubature va fi dat ca referință în pachetul de import asociat pachetului creat.

2. Densitate de probabilitate

Se verifică dacă o funcție f introdusă de utilizator este densitate de probabilitate.

O funcție ***f** : **R*** $\rightarrow$ ***R*** este *densitate de probabilitate* dacă și numai dacă aceasta îndeplinește următoarele condiții simultan:

- $\int_{-\infty}^{+\infty} \boldsymbol{f(x)dx = 1}$
- $f(x) \geq 0$, **pentru orice x ∈ R.**

```r
#2
e_densitate <- function(f)
{
  temp <- cubintegrate(f, lower = -Inf, upper = Inf, method = "pcubature")
  if(!(temp$integral+temp$error>=1 && temp$integral-temp$error<=1)) #Daca integrala nu e aproximativ 1
    return (F)

  for (i in seq(-10000,10000,0.003))
    if(f(i)<0)
      return(F)

  return (T)
}
```

Funcția de mai sus, *e_densitate* este o funcție de tip bool. Aceasta calculează integrala descrisă la primul pas cu ajutorul funcției *cubintegrate* apoi verifică dacă valoarea e aproximativ 1 (spunem aproximativ din cauza micilor erori de calcul ce pot apărea în cadrul determinării integralei). Dacă rezultatul nu verifică această condiție, se returnează false. Altfel, se continuă execuția programului și se verifică a doua condiție, dacă funcția ia doar valori pozitive. Am folosit funcția seq pt. a genera un interval destul de mare pe post de R (de la -10000 la +10000 din 0.003 in 0.003).

3. Un obiect R de tip v.a. continuă

Se creează un obiect de tip v.a. continuă pornind de la o densitate de probabilitate introdusă de utilizator.

Conform conceptelor specifice POO, un obiect este o instanță a unei clase.

Clasa este planul care ajută la crearea unui obiect și conține variabilele membre ale acestuia împreună cu atributele.

În R există 3 tipuri de clase: S3, S4 și RC (Reference classes).

Am implementat o clasa de tipul S4 numită VC (de la variabila continuă).

```
setClass("VC",representation(f="function"),validity = check_VC)
```

La instanțierea fiecărui obiect, utilizatorul dă o funcție de densitate

```
x <-new("VC",f=function(x){x})
```

și apoi, se verifică prin funcția check_vc dacă denisitatea dată este densitate pt o v.a. continuă:

```
check_VC <- function(object)
{ errors <- character()
  g <- object@f
  temp <-  cubintegrate(g, lower = -Inf, upper = Inf, method = "pcubature")
  if(!(temp$integral+temp$error>=1 && temp$integral-temp$error<=1)) #Daca integrala nu e aproximativ 1
  {
    msg <- paste("Valoarea integralei este aproximativ", temp$integral, ".  Ar trebui sa fie 1.", sep = "")
    errors <- c(errors, msg)
    return (errors)
  }

  for (i in seq(-1000,1000,0.003))
    if(g(i)<0)
    {
      msg <- paste("Densitatea nu poate lua valori negative")
      errors <- c(errors, msg)
      return (errors)
    }

  return (TRUE)
}
```

setGeneric anunță că funcția dată ca parametru este o funcție generică. *setMethod* atribuie acea funcție clasei.

```r
setGeneric("constant", valueClass = "numeric", function(object){
  standardGeneric("constant")
})
setMethod("constant",signature(object="VC"),function(object)
{
  constanta(object@f)
}
  )

setGeneric("medie", valueClass = "numeric", function(object){
  standardGeneric("medie")
})
setMethod("medie",signature(object="VC"),function(object)
{
  medie(object@f)
}
)
```

(*un exemplu de 2 funcții atribuite clasei VC, impelentarea acestora se găsește in cerințele anterioare*)

Notă!

Un obiect de tip VC va conține funcțiile impelmentate în proiect la cerințele 1, 2, 5, 6 -> a se vedea codul sursă.

4. Reprezentarea grafică a densității și a funcției de repartiție

Subpunctul 4 a fost tratat împreună cu subpunctul 8.

8. Afișarea unei fișe de sinteză

Introducem câteva dintre repartițiile continue fundamentale:

I.Repartiția exponențială

Definiție:

O variabilă aleatoare X are o repartiție exponențială (negativă) de parametru **μ** dacă funcția sa de probabilitate este de forma: $f(x)= \mu \bullet e^{-\mu x}$, $x \geq 0$, $\mu > 0$, sau funcția de repartiție $F(x) = 1-\mu \bullet e^{-\mu x}$, pentru $x \geq 0$.

Media și dispersia:

$M(X)= 1/\mu$

$D(X) = 1/\mu^2$

Aplicații:

Repartiția exponențială este folosită pentru a calcula timpul de așteptare pentru procesele care se petrec într-un mod continuu și independent.

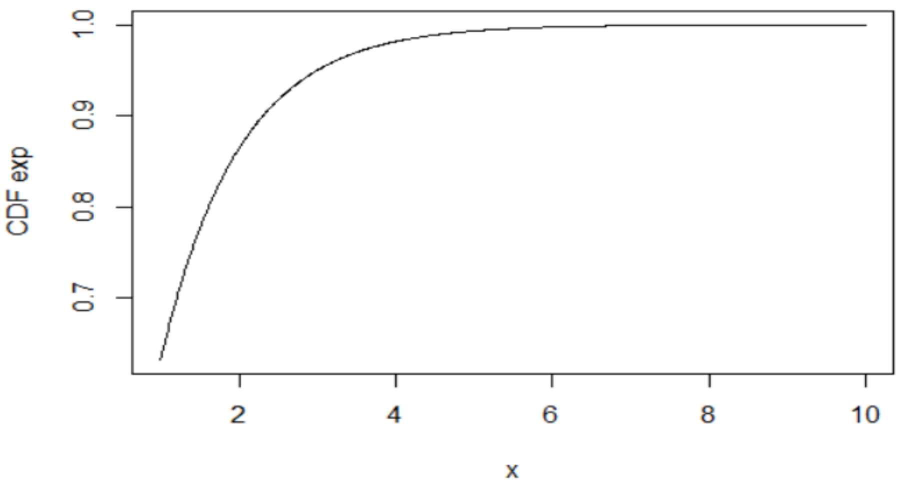
Exemple: timpul pana la sosirea primului autobuz in statie, timpul pana la primul cutremur, timpul pana la sosirea primului client intr-un magazin etc.

Implementarea funcțiilor în R:

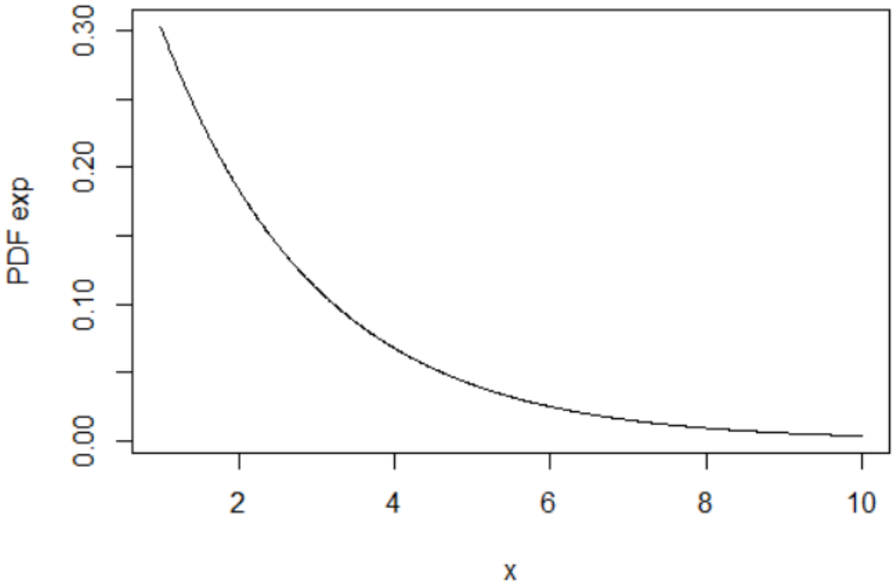
```
d_exp <- function (x,lambda)
{

    lambda * exp (-lambda*x)
}

r_exp <- function (x,lambda)
{
  1-((exp)(-lambda*x))
}

media_exp <- function(x,lambda)
{
  1/lambda
}

mediana_exp <- function (x,lambda)
{
  log(2)/lambda
}

var_exp <- function (x,lambda)
{
  1/(lambda^2)
}
```

Reprezentarea grafică pentru CDF și PDF:

Atât reprezentarea grafică a funcțiilor, cât și afișarea fișei de sinteză au fost făcute în cadrul funcției ”info_exp()”.




```
info_exp <- function()
{
  x<-seq(1,10,by=0.01)
  plot(x,d_exp(x,0.5),type="l",ylab="PDF exp")
  plot(x,r_exp(x,1.0),type="l",ylab="CDF exp")

  print("Repartitia exponentiala, notata exponential(lambda) sau exp(lambda), este distributia de probabilitate a timpului dintre evenimente intr-un proces de t

}

info_exp()
```



II.Repartiția uniformă

Definiție:

Spunem că variabila aleatoare X este repartizată uniform pe intervalul [a,b], a<b, dacă are funcția densitate de probabilitate de forma f(x) = 1/(b-a).

Funcția de repartiție:

F(x)= (x-a)/(b-a), pentru $a \leq x \leq b$

Media și dispersia:

M(X)= (a+b)/2

$D(X) = (b-a)^2 / 12$

Aplicații:

Repartiția uniformă descrie un experiment unde există un rezultat arbitrar care se află între anumite limite. Ea se ocupă cu evenimentele care au aceeași probabilitate de a se petrece. Prin urmare, există numeroase aplicații în care acest tip de repartiție poate fi folosită: situații de testare a ipotezelor, cazuri de eșantionare aleatoare, finanțe etc.

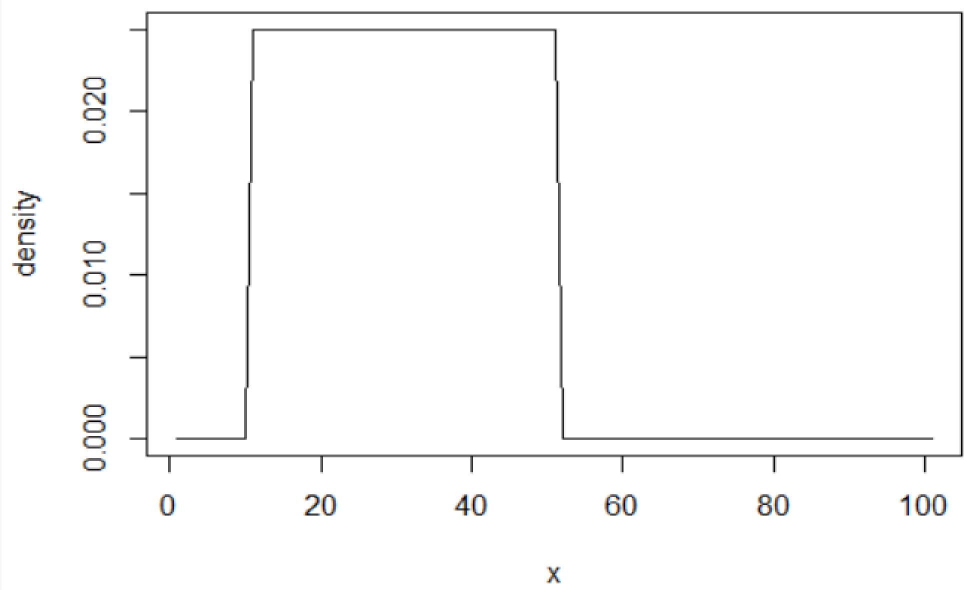
Implementarea în R:

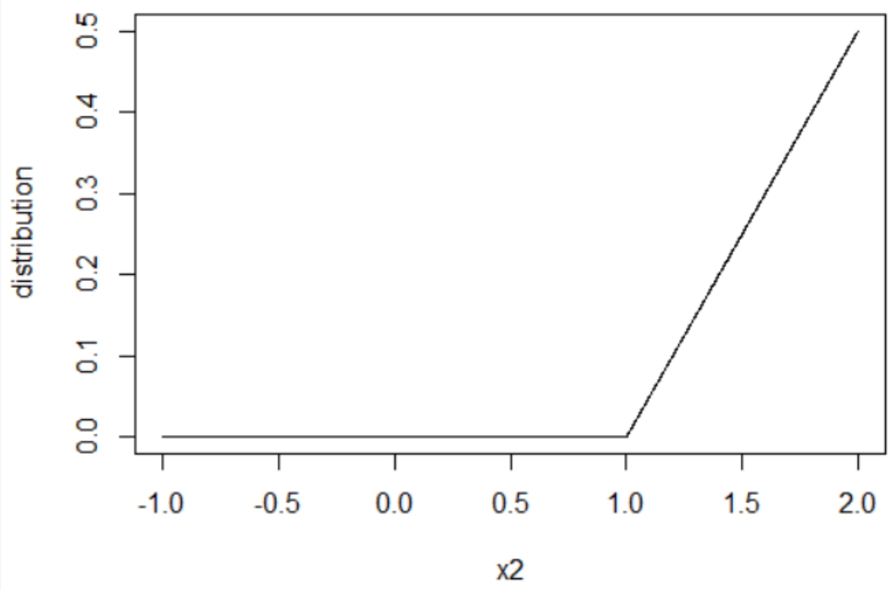
```
media_unif <- function(x,a,b)
{
  (a+b)/2
}

mediana_unif <- function(x,a,b)
{
  (a+b)/2
}

var_unif <- function (x,a,b)
{
  ((b-a)^2)/12
}

```

Reprezentarea grafică

Pentru a trasa graficele funcțiilor densitate de probabilitate, respectiv de repartiție, am folosit funcțiile predefinite din R specifice acestei repartiții, anume dunif() și punif().

```
info_uniform <- function()
{
  x1 <- seq(0,100,by=1)
  y1 <- dunif(x1,10,50)
  plot(y1,type="l",xlab="x",ylab="density")


  x2<- seq(-1,2,by=0.001)
  y2 <- punif(x2,1,3)
  plot(x2,y2, type="l",ylab="distribution")
  print("Repartitia uniforma, notata uniform(a,b) sau U(a,b), reprezinta o familie de distributii de probabil


}

info_uniform()
```


density
0.000
0.010
0.020
0
20
40
60
80
100
x


distribution
0.0
0.1
0.2
0.3
0.4
0.5
-1.0
-0.5
0.0
0.5
1.0
1.5
2.0
x2

III.Repartiția normală (Gaussiană)

Definiție:

O variabilă aleatoare X are o repartiție normală dacă funcția sa de densitate este de forma:

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

Funcția de repartiție nu are o formulă, așa că folosim tabele sau comenzi specifice R pentru a calcula F(x).

Media și dispersia:

M(X)= μ

D(X)= σ^2

Aplicații:

Repartiția normală este folosită în științele naturale și sociale pentru a reprezenta variabile aleatoare cu valori reale ale căror distribuții nu sunt cunoscute.

Importanța acestei repartiții este oferită, în mare de teorema limită centrală, ce afirmă că, în anumite condiții, media mai multor eșantioane ale unei variabile aleatoare cu medie finită și dispersie este ea însăși o variabilă aleatoare a cărei distribuție converge la o distribuție normală pe măsură ce numărul eșantioanelor crește.

Câteva exemple de aplicații: măsurarea erorii, inteligenței, abilității, propagarea incertitudinii, ajustarea parametrilor celor mai mici pătrate etc.

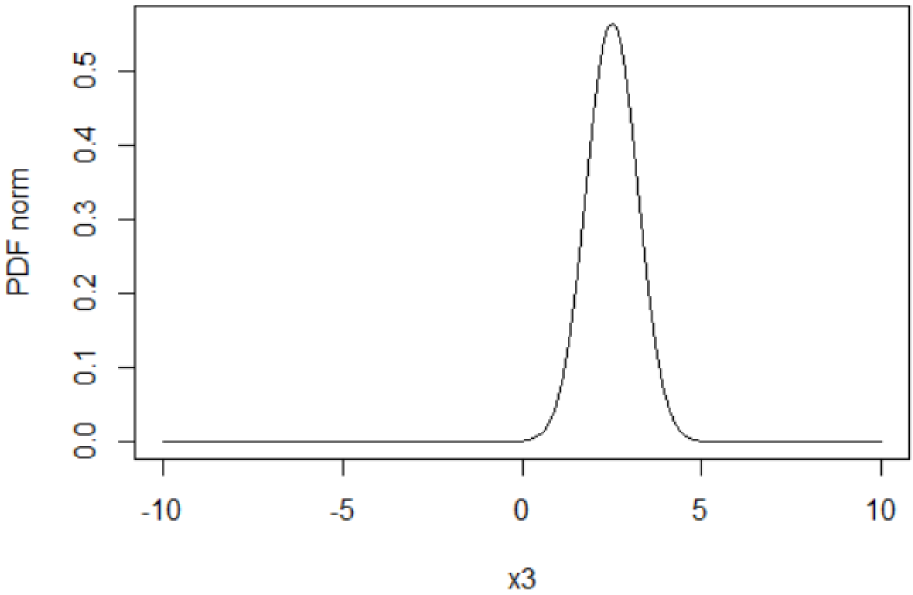
Implementarea în R:

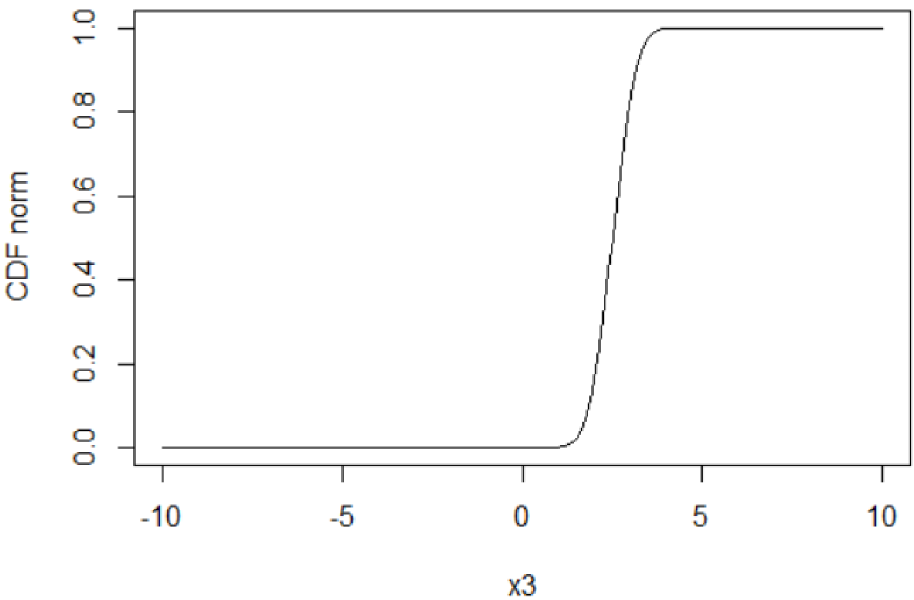
```r
d_norm <- function(x,miu, sigma_p)
{
  sigma <- sqrt(sigma_p)
  (1/(sigma*(sqrt(2*pi)))) * exp((-(x-miu)^2)/(2*sigma_p))
}

media_norm <- function (x,miu,sigma_p)
{
  miu
}

mediana_norm <- function (x,miu,sigma_p)
{
  miu
}

var_norm <- function (x,miu,sigma_p)
{
  sigma_p
}
```

Reprezentarea grafică:

```r
info_norm <- function()
{
  x3 <- seq (-10,10,by=0.1)
  val <- d_norm(x3,2.5,0.5)
  plot(x3,val,type="l",ylab="PDF norm")
  y4 <- pnorm(x3,mean=2.5,sd=0.5)
  plot(x3,y4,type="l",ylab="CDF norm")
  print("Repartitia normala sau repartitia Gaussiana, notata normal(miu,sigma^2) sau N(miu,sigma^2), este un
      Parametrul miu, numar real, reprezinta media repartitiei (de asemenea si mediana), iar sigma^2 (sigma -
      Repartitia normala este folosita in stiintele naturale si sociale pentru a reprezenta variabile aleatoa
      ce afirma ca, în anumite conditii, media mai multor esantioane ale unei variabile aleatoare cu medie fi
      Exemple de probleme: masurarea erorii, inteligentei/abilitatii, inaltimii, mediilor loturilor de date,
}

info_norm()
```

Graficul acestei repartiții este celebrul clopot al lui Gauss.


PDF norm
0.0
0.1
0.2
0.3
0.4
0.5
-10
-5
0
5
10
x3


CDF norm
0.0
0.2
0.4
0.6
0.8
1.0
-10
-5
0
5
10
x3

IV.Repartiția Pareto

Definiție:

O variabilă aleatoare X are o repartiție Pareto dacă funcția sa densitate de probabilitate este de forma: $f(x)= \alpha m^{\alpha} / x^{\alpha+1}$, $m>0$, $\alpha>0$, $x>m$.

Funcția de repartiție:

$F(x)= 1- m^{\alpha}/x^{\alpha}$

Media și dispersia:

$M(X)= \alpha m/\alpha-1$, $\alpha>1$

$D(X)= (m^2\alpha)/((\alpha-1)^2(\alpha-2))$, $\alpha>2$

Aplicații:

Repartiția Pareto modelează o lege de putere, unde probabilitatea ca un eveniment să aibă loc variază ca o putere a unui atribut al evenimentului.

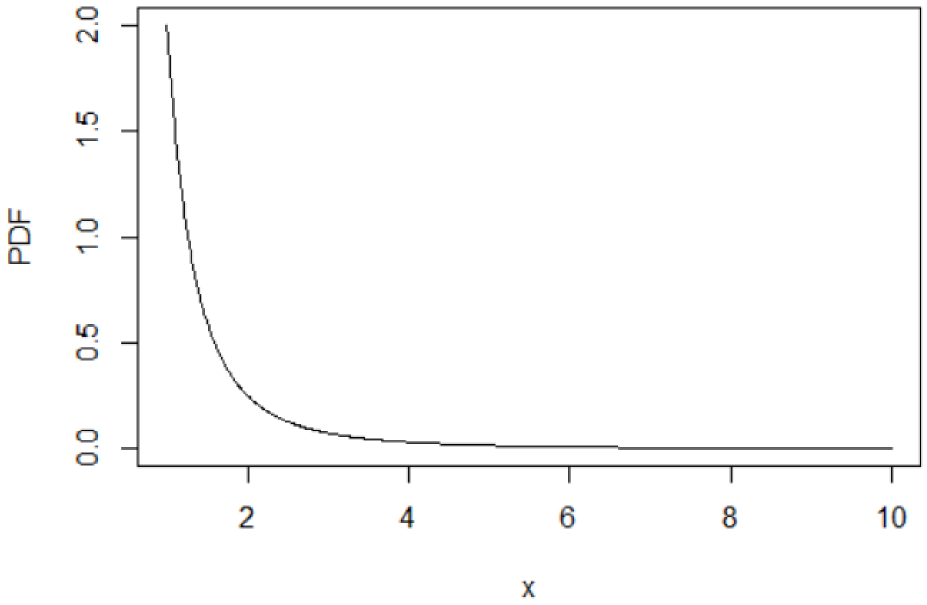
Câteva aplicații practice în care este folosită repartiția Pareto: evaluarea angajaților, business management, mărimea meteoriților, nivelurile populației în orașe etc.

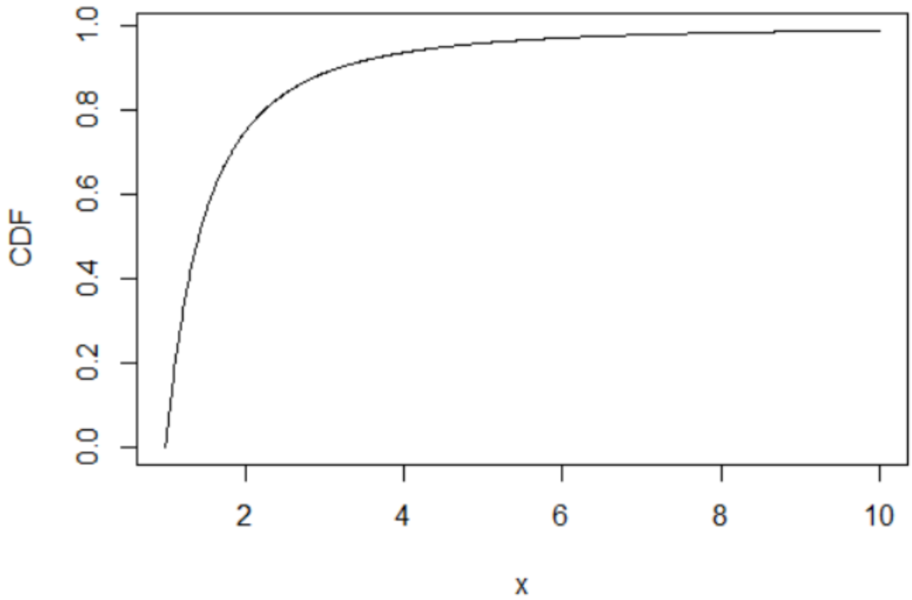
Implementarea în R:

```
dpareto <- function(x,m,alpha)
{
  (alpha*m^alpha)/(x^(alpha+1))

}

rpareto <- function (x,m,alpha)
{
  1-(m^alpha/x^alpha)
}


media_pareto <- function (x,m,alpha)
{
  if (alpha<=1)
     Inf
  else
    return (alpha*m)/(alpha-1)
}

mediana_pareto <- function (x,m,alpha)
{
  m*((2)^(1/alpha))
}

var_pareto <- function (x,m,alpha)
{
  if(alpha<=2)
    Inf
  else
    ((m^2)*alpha)/((alpha-1)^2*(alpha-2))
}


```

Reprezentarea grafică:

```
info_Pareto <- function()
{
  x <- seq(1, 10, 0.01)
  plot(x, dpareto(x, 1,2), xlab="x", ylab="PDF", type="l")
  plot(x,rpareto(x,1,2),xlab="x",ylab="CDF",type="l")
  print("Reparitia Pareto, numita astfel dupa inginerul,economistul si sociologul italian Vilfredo Pareto est
      Initial, a fost folosita pentru a descrie distributia averilor intr-o societate, sustinand ideea ca o m
      Acesti tip de repartitie modeleaza o lege de putere, unde probabilitatea ca un eveniment sa aiba loc va
}


info_Pareto()
```





CDF
0.0
0.2
0.4
0.6
0.8
1.0
2
4
6
8
10
x

5. Calculul mediei, dispresiei, momentelor inițiale și centrale până la ordinul 4

Acest capitol se referă la *caracteristiciile numerice* ale unei v.a. continue:

- **Media** variabilei aleatoare continue X este:

  $$m = M[X] = \int_{-\infty}^{+\infty} xf(x)\,dx$$

  Este implementată de funcția medie care primește ca paramentru funcția f, calculează funcția g drept x*f(x) și apoi integrează funcția g.

- **Momentul inițial de ordin r** al variabilei X este:

  $$m_r = M[X^r] = \int_{-\infty}^{+\infty} x^r f(x)dx$$

  Este implementat de funcția *moment_inițial* care primește ca paramentru funcția f și ordinul r și calculează funcția g drept $x^r f(x)$ pe care o integrează și îi intoarce valoarea.

- **Momentul centrat de ordin r** al variabilei X este:

  $$\mu_r = M[(X - M[X])^r] = \int_{-\infty}^{+\infty} (x-m)^r f(x)dx$$

  Este implementat de funcția *moment_centrat* care primește ca paramentru funcția f și ordinul r, calculează întai m prin funcția de medie definită la primul punct, iar apoi funcția g care este $(x-m)^r f(x)$ pe care o integrează și returnează rezultatul.

  Funcția *moment_centrat4* afișează daca există momente centrate de ordin r unde r este în intervalul [1,4]. Dacă densitatea lui f este pozitivă, atunci sa poate calcula un punct centrat. Pentru acest lucru se apelează funcția *moment_centrat*. Altfel, nu există niciun punct centrat și se afișează nu există momentul centrat de ordin r.

- **Dispersia** variabilei X este momentul centrat de ordinul 2:

  $$\delta^2 = D^2[X] = \mu_2 = \int_{-\infty}^{+\infty} (x-m)^2 f(x)dx$$

  Este implementată de funcția dispresie care primește ca paramentru funcția f și întoarce rezultatul funcției pentru un moment centrat cu valoarea 2 pt r.

```r
#5
medie <- function(f)
{
  g<-function(x)
  {
    x*f(x)
  }
  return (cubintegrate(f, lower = -Inf, upper = Inf, method = "pcubature")$integral)
}

medie(f)

moment_centrat<- function(f,k) #Functie, rang
{
  med <- medie(f)
  g<-function(x)
  {
    (x-med)^k*f(x)
  }
  return (cubintegrate(f, lower = -Inf, upper = Inf, method = "pcubature")$integral)
}

moment_centrat(f)

moment_initial  <- function(f)
{

  g<-function(x)
  {
    x^k*f(x)
  }
  return (cubintegrate(f, lower = -Inf, upper = Inf, method = "pcubature")$integral)
}

moment_initial(f)


moment_centrat4 <- function(f)
{
  adev <- e_densitate(f)
  for (r in seq(1,4))
  {
    if(adev == T)
    {
      cat("Momentul centrat de ordin ", r, " este ", moment_centrat(f,r))
      #print("Momentul centrat de ordin " ++ r ++ " este "  ++ moment_centrat(f,r))
    }
    else
      cat("Nu exista moment centrat de ordin", r, "\n")
      #print("Nu exista moment centrat de ordin " ++ r)
  }
}
```

```r
dispersie <- function(f)
{ moment_centrat(f,2)}
```

6. Media și dispersia unei v.a. g(X)

Se numește funcție de repartiție a variabilei aleatoare continue X (la fel ca funcția de repartiție a variabilelor aleatoare), funcția reală

F : R →R,

F(x) = P {X ≤ x} = P({e ∈ E | X(e) ≤ x}) $= \int_{-\infty}^{x} f(t)dt$

pentru orice x ∈ R.

Funcția de repartiție F asociază oricărui număr real x probabilitatea ca valorile lui X să fie mai mici sau cel mult egale cu x.

În cerința, X este o variabilă aleatoare continuă cu f o repartiție continua. Funcția g este continuă(graficul acesteia nu are întreruperi sau "rupturi"). Astfel, compunerea dintre cele 2 este o v.a. continuă.

Pentru a trece X prin funcția g, la v.a. continue este suficient să înmulțim funcțiile. Cum în R nu putem folosi operatorul * pentru înmulțirea a doua funcții am folosit un wrapper numit *multiply* care face asta:

```
multiply = function(a,b)
{
  force(a)
  force(b)
  function(x){a(x)*b(x)}
}
```

Funcția *medie2* implementată mai jos îmulțește funcțiile date ca parametru, pe care, apoi le integrează ca în formula de medie și întoarce valoarea integralei.

Am folosit **cubintegrate din pachetul cubature** în loc de **integrate** pentru a evita erorile de cod daturate erorilor de calcul. Funcția cubintegrate primește ca parametru o funcție de integrat, limita inferioară și superioară a integralei și methoda. Pentru aceasta am folosit *pcubature* care aplică integrale multivariate peste hipercurbe.

Mai jos se găsește și un exemplu.

```r
medie2 <- function(f,g)
{
  h = multiply(g,f)
  cubintegrate(h, lower = -Inf, upper = Inf, method = "pcubature")
}


f<-function(x){
  return(x+2)
}

g<-function(x){
  return(x^2)
}

dispersie2(f,g)

medie2(f,g)
```

```
178:1  (Top Level)

Console  Terminal  Jobs
D:/documente/An2_sem1/Prob&Statistici/Lab/
+  # return (integrate(h,-Inf,Inf)$value)
+ }
> medie2(f,g)
$integral
[1] 6.427752e+60

$error
[1] 3.213876e+60

$neval
[1] 131073

$returnCode
[1] 1

>
```

Dispersia este calculată prin funcția *dispersie2* care înmulțește funcțiile date ca parametru, și trasmite funcția h astfel formată ca parametru pentru *dispersie* scrisă la subpunctul anterior.

```r
#dispersie
dispersie2 <- function(f,g)
{
  h = multiply(g,f)
 dispersie(h)

}
```

Rezolvările se bazează pe urmatoarea proprietate:

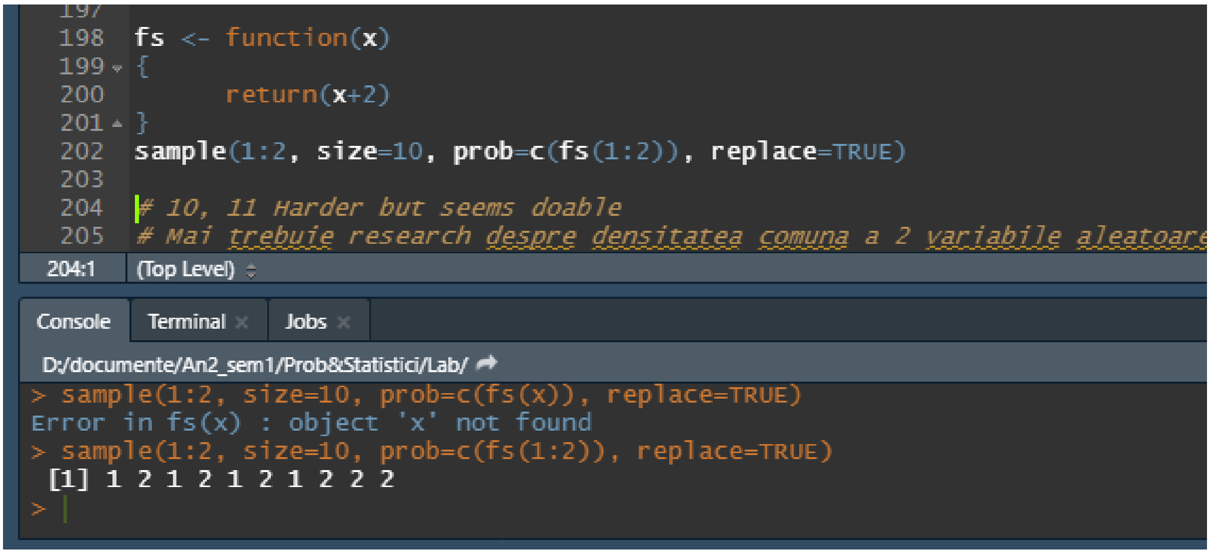
Dacă X este o variabilă aleatoare și Y = g(X) este o variabilă obținută printr-o transformare cu ajutorul funcției g : R → R, continuă și bijectivă, atunci **media** transformării Y = g(X) este :

$$M[Y] = +\int_{-\infty}^{+\infty} g(x) \cdot f(x)dx$$

9. Calcularea a n valori dintr-o repartiție de v.a. continue

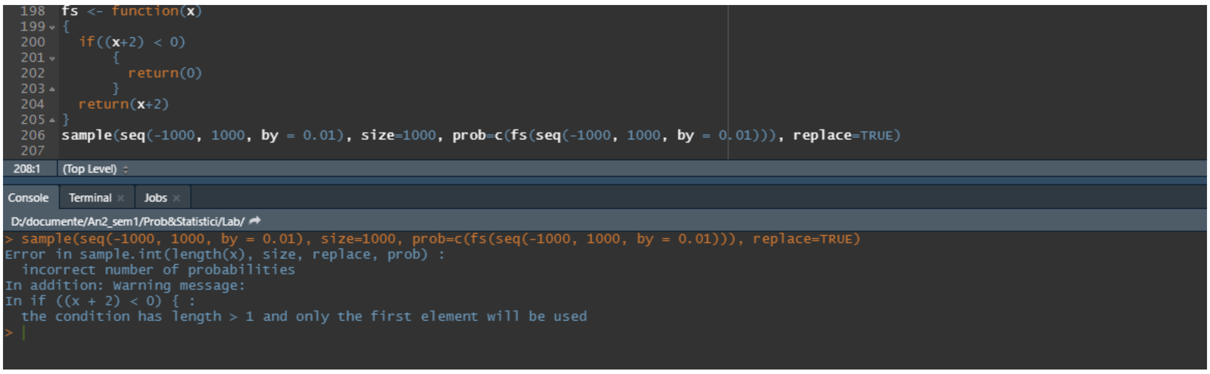
Se vor genera n valori dintr-o repartiție de variabile aleatoare continuă.

Din intervalul R voi extrage un numar n de valori cu probabilitatea dată de densitatea v.a. continue (funcția f). Pentru a afla valoarea acestora voi calcula f(valoare extrasa).



În exemplul de mai sus prin funcția *sample* extragem 10 numere 1 2 astfel încat acestea apar repetate cu probabilitatea data de functia fs.

Întrucât pentru următorul exemplu am primit o eroare, am utilizat funcția *lapply*, care aplică funcția pe un intreval de valori(dat cu *seq* cu pasul de 0.01):



```
sample(seq(-1000, 1000, by = 0.01), size=6, prob=lapply(seq(-1000, 1000, by = 0.01), fs), replace=TRUE)
```

```
D:/documente/An2_sem1/Prob&Statistici/Lab/
> sample(seq(-1000, 1000, by = 0.01), size=6, prob=lapply(seq(-1000, 1000, by = 0.01), fs), replace=TRUE)
[1] 474.36 497.14 463.07 252.69 732.02 529.66
>
```

Pornind de la aceste exemple, vom implementa o funcție *generează* care generează n numere din repartiția unei v.a continue pe R:

```
#9
genereaza <- function(n,f)
{
  sample(seq(-1000, 1000, by = 0.01), size=n, prob=lapply(seq(-1000, 1000, by = 0.01), f), replace=TRUE)
}
```

10. Calculul covalenței și al coeficientului de covariație pentru 2 v.a. continue

Pentru 2 v.a. continue vom **calcula colalența** și **coeficientul de covariație,** ca să putem realiza aceste calcule, avem nevoie de **densitatea comuna** a celor două v.a. continue.

Funcția denisității și probabilității comune reprezintă probabilitatea ca perechile ordonate de v.a să ia valori în intervalele (închise sau deschise) [a, b] și [c, d]:

$$P(a \leq X \leq b, c \leq Y \leq d) = \int_{a}^{b} \int_{c}^{d} f_{XY}(x, y) dy dx$$

În cazul discret, dacă XX și YY sunt două variabile aleatorii, atunci fiecărei perechi de rezultate posibile X = x și Y = y i se poate atribui numărul $p_{xy}(x, y)$ probabilitatea acelei perechi de rezultate.

Suma tuturor perechilor posibile de rezultate este apoi egală cu una în cazul discret:

$$\sum_{xy} p_{xy}(x, y) = 1$$

Pentru cazul continuu, înlocuim suma cu integrale și p_{xy} cu f_{xy} obținând funcția denisității și probabilității comune pentru 2 v.a. comune.

$$\iint f_{xy}(x, y) dx dy = 1$$

Covarianța variabilelor X ¸si Y este definită prin:

$$Cov[X, Y] = M[(X - M[X])(Y - M[Y])]$$

sau

$$Cov(X, Y) = E(XY) - E(X)(Y)$$

unde E(X) este media v.a continue calculată prin:

$$E(X) = \iint x f_{XY}(x, y) dy dx$$

Și implementarea ei în R:

```r
#10 - covarianta

cov <- function(joint_pdf, interval_X_lower, interval_X_higher, interval_Y_lower, interval_Y_higher)
{
  #verificare intervale
  if(interval_X_lower > interval_X_higher | interval_Y_lower > interval_Y_higher){
    stop("Unul dintre intervalele specificate este invalid.")
  }

  #determinam mediile necesare formulei de covariatie

  temp_func_x <- function(x, y) x*joint_pdf(x, y)

  Ex <- integrate(function(x) {
    sapply(x, function(x) {
      integrate(function(y) temp_func_x(x, y), interval_Y_lower, interval_Y_higher)$value
    })
  }, interval_X_lower, interval_X_higher)$value

  temp_func_y <- function(x, y) y*joint_pdf(x, y)

  Ey <- integrate(function(x) {
    sapply(x, function(x) {
      integrate(function(y) temp_func_y(x, y), interval_Y_lower, interval_Y_higher)$value
    })
  }, interval_X_lower, interval_X_higher)$value

  temp_func_xy <- function(x, y) x*y*joint_pdf(x, y)

  Exy <- integrate(function(x) {
    sapply(x, function(x) {
      integrate(function(y) temp_func_xy(x, y), interval_Y_lower, interval_Y_higher)$value
    })
  }, interval_X_lower, interval_X_higher)$value

  return(Exy - Ex*Ey)

}
```

Se numește **coeficient de corelație** dintre variabilele aleatoare X și Y și se notează cu $\rho(X,Y)$ raportul dacă există:

$$\rho(X,Y) = \frac{cov(X,Y)}{D(X)D(Y)}$$

Unde D este **abaterea medie pătratică:**

$$\sigma = D[X] = \sqrt{D^2[x]}$$

Implementarea coeficientului de corelație în R:

```r
#10 - coeficientul de corelatie

coef <- function(joint_pdf, interval_X_lower, interval_X_higher, interval_Y_lower, interval_Y_higher)
{
  #verificare intervale
  if(interval_X_lower > interval_X_higher | interval_Y_lower > interval_Y_higher){
    stop("Unul dintre intervalele specificate este invalid.")
  }

  #determinam variatia fiecarei variabile apoi dispersia si in final raportul care ne da coeficientul de corelatie

  temp_func_x <- function(x, y) x*joint_pdf(x, y)

  Ex <- integrate(function(x) {
    sapply(x, function(x) {
      integrate(function(y) temp_func_x(x, y), interval_Y_lower, interval_Y_higher)$value
    })
  }, interval_X_lower, interval_X_higher)$value

  temp_func_x2 <- function(x, y) x*x*joint_pdf(x, y)

  Ex2 <- integrate(function(x) {
    sapply(x, function(x) {
      integrate(function(y) temp_func_x2(x, y), interval_Y_lower, interval_Y_higher)$value
    })
  }, interval_X_lower, interval_X_higher)$value

  Var_x = Ex2 - Ex*Ex

  #acum pt Y

  temp_func_y <- function(x, y) y*joint_pdf(x, y)

  Ey <- integrate(function(x) {
    sapply(x, function(x) {
      integrate(function(y) temp_func_y(x, y), interval_Y_lower, interval_Y_higher)$value
    })
  }, interval_X_lower, interval_X_higher)$value

  temp_func_y2 <- function(x, y) y*y*joint_pdf(x, y)

  Ey2 <- integrate(function(x) {
    sapply(x, function(x) {
      integrate(function(y) temp_func_y2(x, y), interval_Y_lower, interval_Y_higher)$value
    })
  }, interval_X_lower, interval_X_higher)$value

  Var_y = Ey2 - Ey*Ey

  #dispersiile

  if(Var_x * Var_y < 0){
    stop("Produsul variatiilor negativ, nu putem aplica radical.")
  }

  numitor <- sqrt(Var_x * Var_y)

  #raportul

  #conditia de existenta (numitorul diferit de 0)


  if(numitor <= 0){
    stop("Numitor egal cu 0, coeficientul nu exista.")
  }

  return(cov(joint_pdf, interval_X_lower, interval_X_higher, interval_Y_lower, interval_Y_higher) / numitor)

}
```

11. Densități marginale și condiționate pentru 2 v.a. continue

La fel ca la subiectul anterior, pentru a calcula densitățile marginale și condiționale a doua v.a. continue avem nevoie de **densitatea comuna** a celor două v.a. continue:

$$\iint f_{xy}(x,y)dxdy = 1$$

Funcţia densitate de probabilitate marginală:

$$f_x(a) = \int_{-\infty}^{+\infty} f_{xy}(a,b)db$$

$$f_x(b) = \int_{-\infty}^{+\infty} f_{xy}(a,b)da$$

Și implementarea ei în R:

```r
#11 - densitatile marginale

marginale <- function(joint_pdf, interval_X_lower, interval_X_higher, interval_Y_lower, interval_Y_higher)
{
  #verificare intervale
  if(interval_X_lower > interval_X_higher | interval_Y_lower > interval_Y_higher){
    stop("Unul dintre intervalele specificate este invalid.")
  }

  #pentru a obtine marginala pentru x, integram dupa y pdf comuna si viceversa.

  pdf_x <- function(x) integrate(function(y) joint_pdf(x, y), interval_Y_lower, interval_Y_higher)

  pdf_y <- function(y) integrate(function(x) joint_pdf(x, y), interval_Y_lower, interval_Y_higher)

  pdf_list <- list("Fx" = pdf_x, "Fy" = pdf_y)

  return(pdf_list)
}
```

Știm că pentru variabile discrete, pentru a calcula probabilitatea lui Y condiționat de x $P(Y = y|\, X = \ x)$, putem utiliza următoarea formulă, conform **Teoremei lui Bayes:**

$$P(Y = y|\, X = \ x) = \frac{P(X = x|\, Y = \ y)}{P(X = x)}$$

Similar, pentru v.a. continue, putem calcula **funcţia densitate de probabilitate condiționată** utilizând densitățile marginale:

$$f_Y(y\,|X = x) = \frac{f_X(x\,|\,Y = \ y)}{f_X(x)} f_Y(y)$$

Implementarea în R:

```r
#11 - densitatile conditionate

conditionate <- function(joint_pdf, interval_X_lower, interval_X_higher, interval_Y_lower, interval_Y_higher)
{
  #verificare intervale
  if(interval_X_lower > interval_X_higher | interval_Y_lower > interval_Y_higher){
    stop("Unul dintre intervalele specificate este invalid.")
  }

  #obtinem marginalele pentru calculul conditionatelor

  marginale_list <- marginale(joint_pdf, interval_X_lower, interval_X_higher, interval_Y_lower, interval_Y_higher)

  marginala_x <- marginale_list$Fx

  marginala_y <- marginale_list$Fy

  #definim conditionatele si cu un if pentru criteriul de existenta

  conditionata_x <- function(x, y) {

    if(marginala_x(x)$value <= 0)
      stop("Densitatea marginala este mai mica sau egala cu 0.")

    return(joint_pdf(x, y)/marginala_x(x)$value)

  }

  conditionata_y <- function(x, y) {

    if(marginala_y(y)$value <= 0)
      stop("Densitatea marginala este mai mica sau egala cu 0.")

    return(joint_pdf(x, y)/marginala_y(y)$value)

  }

  conditionate_list <- list("Fy" = conditionata_x, "Fx" = conditionata_y)

  return(conditionate_list)
}
```

Bibliografie

- Seminar an2 sem 1 Prop && Statistica – V.a continue.pdf – Simona Cojocea
- CURS MS II VARIABILE ALEATOARE CONTINUE 1 – Daniela Rosu
- Teoria probabilitatilor si statistica matematica Barbacioru Iuliana Carmen – CURSUL 10
- Wikipedia
- https://www.datacamp.com/community/tutorials/r-objects-and-classes
- https://brilliant.org/wiki/continuous-random-variables-joint-probability/
- https://brilliant.org/wiki/conditional-probability-distribution/
- https://cran.r-project.org/web/packages/cubature/cubature.pdf