

Construirea unui pachet R pentru lucru cu variabile aleatoare continue

Proiect Probabilitati si Statistica

**Anghelea Gabriel-Bogdan
Diyar Mert Daniel
Nicoi Alexandru
Ulmeanu Cristian**

Cuprins

1	Introducere	2
2	Exercitii	2
2.1	Exercitiul 1	2
2.2	Exercitiul 2	2
2.3	Exercitiul 4	3
2.4	Exercitiul 5	9
2.5	Exercitiul 6	11
2.6	Exercitiul 8	12
2.7	Exercitiul 9	14
2.8	Exercitiul 10	15
2.9	Exercitiul 11	16
2.10	Exercitiul 12	17

1 Introducere

Folosind documentul suport și orice alte surse de documentare considerați potrivite construiți un pachet R care să permită lucru cu variabile aleatoare continue. Pentru a primi punctaj maxim, pachetul trebuie să implementeze cel puțin 8 din următoarele cerințe (oricare 8!):

2 Exerciții

2.1 Exercițiul 1

Fiind dată o funcție f , introdusă de utilizator, determinarea unei constante de normalizare k . În cazul în care o asemenea constantă nu există, afișarea unui mesaj corespunzător către utilizator.

Pentru a calcula constanta de normalizare k , vom determina inversul valorii integralei funcției f de la $-\infty$ la $+\infty$, adică $\frac{1}{\int_{-\infty}^{\infty} f(x)dx}$.

Pentru realizarea calcului, am implementat funcția "ConstantaDeNormalizareK", care primește ca parametru funcția. Dacă valoarea integralei este egală cu 0, atunci afișăm utilizatorului faptul că funcția nu admite constante de normalizare.

```
1 ConstantaDeNormalizareK <- function(funcție)
2 {
3   integrala <- integrate(vectorize(funcție), -Inf, Inf)$value;
4   if (integrala != 0){
5     return(1 / integrala);
6   }
7   else{
8     print("Aceasta funcție nu admite constante de normalizare");
9   }
10 }
11 #ConstantaDeNormalizareK(f1)
12 #the integral is probably divergent
13 #ConstantaDeNormalizareK(f2)
14 #"Aceasta funcție nu admite constante de normalizare"
```

2.2 Exercițiul 2

Verificarea dacă o funcție introdusă de utilizator este densitate de probabilitate.

Pentru realizarea acestui exercițiu am cautat documentația din cadrul cursului cât și a laboratorului astfel am găsit și am folosit 2 proprietăți ce caracterizează o funcție dreptă densitate de probabilitate. Cele 2 proprietăți folosite pentru verificare sunt:

1. $f(x) \geq 0$ (Funcția este pozitivă)
2. $\int_{-\infty}^{\infty} f(x)dx$

În cadrul verificării că funcția este pozitivă metoda folosită nu este în totalitate precisă deoarece folosind funcția "sequence" iau valorile din cadrul unui interval și calculez funcția astfel verificând pozitivitatea, dar din pricina limitărilor funcției aceasta permite un interval finit, și pe lângă acesta factor funcția permite ca punctele de minim și de maxim ale intervalului să fie maxim egale cu $\text{pas} \cdot 10E \pm 10$.

Apoi am calculat integrala și am verificat că rezultatul să fie egal cu 1, am decis să aproximez rezultatul deoarece am întâmpinat problema că rezultatul să fie foarte aproape de 1 dar să nu fie acceptat pentru verificare, acesta nefiind egal exact cu 1 (ex. 0.9999999999999292).

Funcția de verificare am creat-o să primească ca parametrii funcția de testare, intervalul pentru verificarea pozitivității și pasul, toate fiind inputuri de utilizator.

```

1  ## Verificarea daca o functie introdusa de utilizator este densitate de probabilitate.
2
3  densitate <- function(f,lower,upper,pas){
4    val <- seq(lower,upper,pas);
5    #Verificare functie pozitiva
6    if(all(f(val)>=0)=='TRUE'){
7      #Verificare integrabilitate
8      if(typeof(integrate(Vectorize(f),-Inf,Inf))=='logical' && integrate(Vectorize(f),-Inf,
9      Inf)==FALSE || integrate(Vectorize(f),-Inf,Inf) $ message!="OK"){
10         return(FALSE)
11       }
12     }
13     else{
14       i = integrate(Vectorize(f),-Inf,Inf) $ value
15       if (round(i)==1){
16         print("Funcția este densitate de probabilitate.")
17       }
18     }
19   }
20 }
21
22 else{
23   print("Funcția este negativă pe intervalul ales.")
24 }
25 }
26
27 # #Funcție ce nu indeplinește condițiile
28 # f1 <- function(x){
29 #   if (x > 0 && x < 4){
30 #     3/5 * (2*x-6*x^2)
31 #   }else{
32 #     0
33 #   }
34 # }
35
36 # #Funcție ce indeplinește condițiile
37 # f2 <- function(x){
38 #   if (x > 0 && x < 4){
39 #     3/20*(x^2-2*x+1)
40 #   }else{
41 #     0
42 #   }
43 # }
44
45 # test <- integrate(Vectorize(f),-Inf,Inf) $ value
46 # print(test)
47
48 # densitate(f1,-10000,10000,1)
49 # densitate(f2,-10000,10000,0.01)

```

2.3 Exercițiul 4

Reprezentarea grafică a densității și a funcției de repartiție pentru diferite valori ale parametrilor repartiției. În cazul în care funcția de repartiție nu este dată într-o formă explicită (ex. repartiția normală) se acceptă reprezentarea grafică a unei aproximări a acesteia.

Pentru realizarea calculelor, am implementat următoarele funcții:

- PDF_to_CDF(f, x) - integrează funcția densitate de probabilitate (PDF) pentru a obține valoarea funcției de repartiție cumulativă (CDF)
- verif_densitate(f, a, b) - verifică dacă funcția de densitate respectă condițiile necesare (f pozitivă și probabilitate totală egală cu 1)
- plot_densitate(f, a, b, name) - realizează graficul densității
- plot_repartiție(F, a, b, name) - realizează graficul repartiției

- `plot_repartitie_gen(f, a, b)` - functie care trateaza cazul in care este nevoie sa facem graficul pentru functii unde nu cunoastem functia de repartitie
- `parse_repartitii_cunoscute(name, CDF, ...)` - in functie de numele functiei de repartitie introdus in "name", verifica conditiile necesare ale acelei repartitii si realizeaza, dupa caz, ori graficul repartitiei, ori graficul densitatii.

Ultima functie listata are posibilitatea de a crea grafice pentru 4 tipuri de repartitii: uniform, exp, normal, cauchy.

Formulele necesare pentru repartitiile mentionate sunt urmatoarele:

$$\frac{1}{\pi} \arctan\left(\frac{x - x_0}{\gamma}\right) + \frac{1}{2}$$

Figure 1: Cauchy CDF

$$\frac{1}{\pi\gamma \left[1 + \left(\frac{x-x_0}{\gamma}\right)^2\right]}$$

Figure 2: Cauchy PDF

$$1 - e^{-\lambda x}$$

Figure 3: Exp CDF

$$\lambda e^{-\lambda x}$$

Figure 4: Exp PDF

$$\frac{1}{2} \left[1 + \operatorname{erf}\left(\frac{x - \mu}{\sigma\sqrt{2}}\right)\right]$$

Figure 5: Normal CDF

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

Figure 6: Normal PDF

$$\begin{cases} 0 & \text{for } x < a \\ \frac{x-a}{b-a} & \text{for } x \in [a, b] \\ 1 & \text{for } x > b \end{cases}$$

Figure 7: Uniform CDF

$$\begin{cases} \frac{1}{b-a} & \text{for } x \in [a, b] \\ 0 & \text{otherwise} \end{cases}$$

Figure 8: Uniform PDF

```

2 # parametrilor reparti??iei. ??n cazul ??n care func??ia de reparti??ie nu este data ??ntr
-o forma
3 # explicita(ex. reparti??ia normala) se accepta reprezentarea grafica a unei aproximari a
4 # acesteia.
5
6 library(pracma)
7
8 # Integram functia densitate de probabilitate(cdf) ca sa obtinem
9 # valoarea functiei de repartitie cumulativa
10
11 PDF_to_CDF <- function(f, x)
12 {
13   tryCatch(
14     {
15       integrate (f, 0, x)$value
16     },
17     error= function (e){
18       print(e)
19     }
20   )
21 }
22
23 # Verificam conditiile functiei de densitate
24 # 1. Este pozitiva
25
26 verific_densitate <- function(f, a, b) {
27   for (x in seq(a, b, 0.1)){
28     if( f(x) < 0 ){
29       print(paste("PDF nu poate sa aiba valori negative",))
30       return(FALSE)
31     }
32   }
33
34 # 2.Probabilitatea totala trebuie sa fie 1
35
36 total <- integral(Vectorize(f), max(-Inf, a), min(b, +Inf))
37
38 if(abs(total - 1) >0.1){
39   print ( paste ("Eroare, probabilitatea cumulata totala trebuie sa fie egala cu 1") )
40   return ( FALSE )
41 }
42
43 return(TRUE)
44
45 }
46
47 # Afisam graficul densitatii
48
49 plot_densitate <- function(f, a, b, name=""){
50   # Validam PDF
51   if(!verif_densitate(f, a, b)){
52     return()
53   }
54
55   #Afisam graficul PDF
56
57   ax <- seq(a, b, 0.1)
58   ay <- c()
59   for (x in ax){
60     ay= append(ay, f(x))
61   }
62
63   plot (ax , ay , type ="l", main = noquote ( paste ( name , " PDF ")), col =" magenta ",
64     xlab ="x", ylab="y")
65 }
66
67 #Afisam graficul repartitiei
68
69 plot_repartitie <- function(F, a, b, name){
70   ax <- seq(a, b, 0.01)
71   ay <- c()
72   for (x in ax){
73     ay = append (ay, F(x))
74   }
75
76   plot (ax, ay , col =" magenta ", type ="l", main = noquote ( paste ( name , " CDF ")),
77     xlab ="x", ylab="y")

```

```

75 }
76
77 #Cream o functie de afisare pentru functiile
78 # unde nu cunoastem functia de repartitie
79
80 plot_repartitie_gen <- function(f, a, b){
81
82   if( !verif_densitate(f, a, b)){
83     return()
84   }
85
86   ax <- seq(a, b, 0.01)
87   ay <- c()
88   for (x in ax) {
89     ay = append(ay, PDF_to_CDF(f, x))
90   }
91   plot (ax , ay , col =" magenta ", type ="l", main ="CDF ", xlab ="x", ylab ="y")
92
93 }
94
95 parse_repartitii_cunoscute <- function(name, CDF=FALSE, ...) {
96   params <- list(...)
97   if (name == "uniform") {
98     if (!is.null(params$a) && !is.null(params$b)) {
99       a <- params$a
100       b <- params$b
101       if (a >= b) {
102         return("parametri incorecti")
103       }
104
105       f <- function(x) 1 / (b - a)
106       F <- function(x)(x - a) / (b - a)
107       if (CDF) {
108         plot_repartitie(F, a, b, name)
109       }
110       else {
111         plot_densitate(f, a, b, name)
112       }
113     }
114     else {
115       return("parametrii necesari nu au fost pasati")
116     }
117   }
118   else if (name == "exp") {
119     if (!is.null(params$lambda)) {
120       lambda <- params$lambda
121       if (lambda <= 0) {
122         return("parametrii incorecti")
123       }
124
125       f <- function(x) (lambda * exp(1)^(-lambda * x))
126       F <- function(x) (1 - exp(1)^(-lambda * x))
127       if (CDF) {
128         plot_repartitie(F, 0, 70, name)
129       }
130       else {
131         plot_densitate(f, 0, 70, name)
132       }
133     }
134     else {
135       return("parametrii necesari nu au fost pasati")
136     }
137   }
138   else if (name == "normal") {
139     if (!is.null(params$mu) && !is.null(params$sigma)) {
140       mu <- params$mu
141       sigma <- params$sigma
142       if (sigma <= 0) {
143         return("parametrii incorecti")
144       }
145
146       f <- function(x) ((1 / (sigma * sqrt(pi * 2)))*(exp(1)^(-(x - mu)^2)/(2 * sigma
147 ~ 2))))
148       F <- function(x) (pnorm(x, mu, sigma))
149

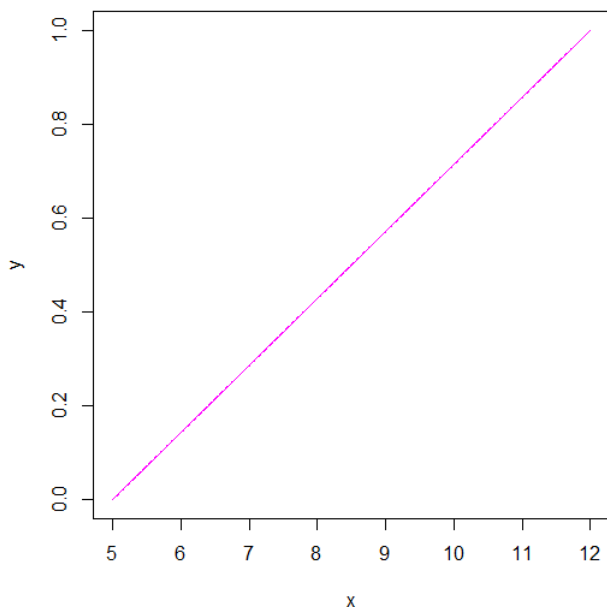
```

```

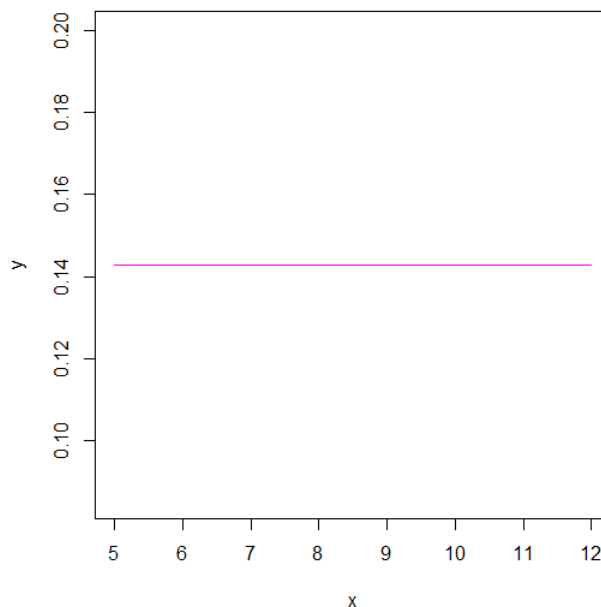
150         if (CDF) {
151             plot_repartitie(F, -30, 30, name)
152         }
153         else {
154             plot_densitate(f, -30, 30, name)
155         }
156     }
157     else {
158         return("parametrii necesari nu au fost pasati")
159     }
160 }
161 else if (name == "cauchy") {
162     if (!is.null(params$location) && !is.null(params$scale)) {
163         location <- params$location
164         scale <- params$scale
165         if (scale <= 0) {
166             return("parametrii incorecti")
167         }
168
169         f <- function(x) 1 / (pi * scale * (1 + ((x - location) / (scale))^2))
170         F <- function(x) (1 / pi) * atan((x - location) / scale) + 1 / 2
171
172         if (CDF) {
173             plot_repartitie(F, -30, 30, name)
174         }
175         else {
176             plot_densitate(f, -30, 30, name)
177         }
178     }
179 }
180 else {
181     print("repartitie necunoscuta")
182 }
183 }
184
185 #EXAMPLE:
186
187 # parse_repartitii_cunoscute("uniform", FALSE, a=5, b=12)
188 # parse_repartitii_cunoscute("uniform", TRUE, a=5, b=12)
189 # parse_repartitii_cunoscute("exp", FALSE, lambda=7)
190 # parse_repartitii_cunoscute("exp", TRUE, lambda=5)
191 # parse_repartitii_cunoscute("normal", FALSE, mu=2, sigma=3)
192 # parse_repartitii_cunoscute("normal", TRUE, mu=2, sigma=3)
193 # parse_repartitii_cunoscute("cauchy", FALSE, location=0, scale=3)
194 # parse_repartitii_cunoscute("cauchy", TRUE, location=0, scale=3)
195 # plot_repartitie_gen(function(x) x / 22, 0, 2)

```

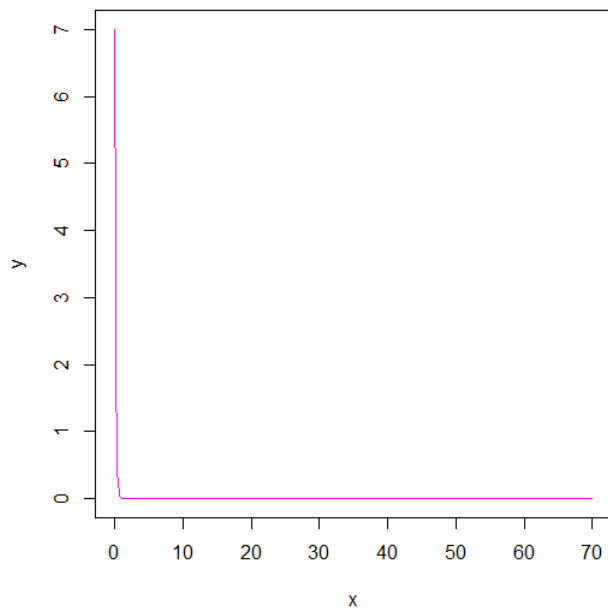
uniform CDF



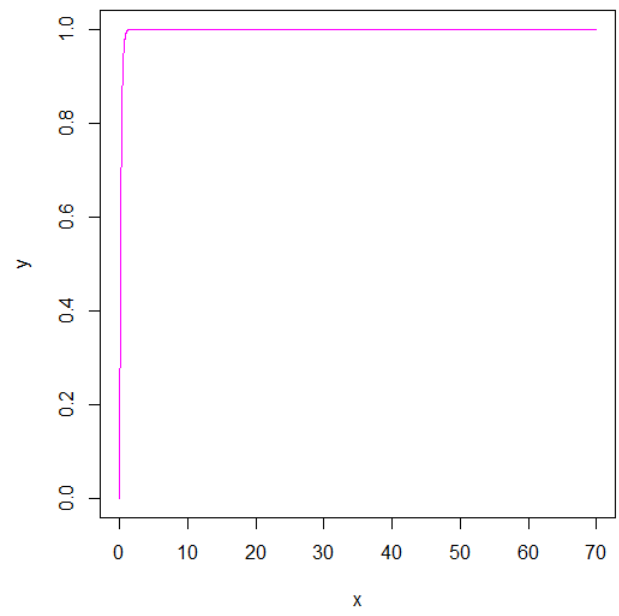
uniform PDF



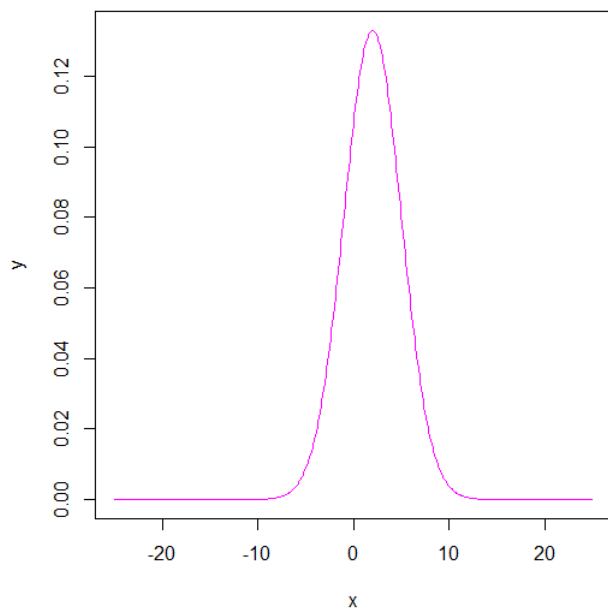
exp PDF



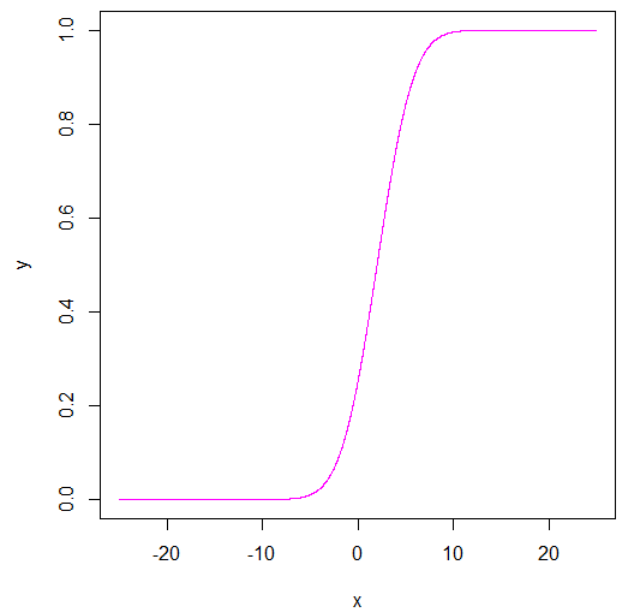
exp CDF

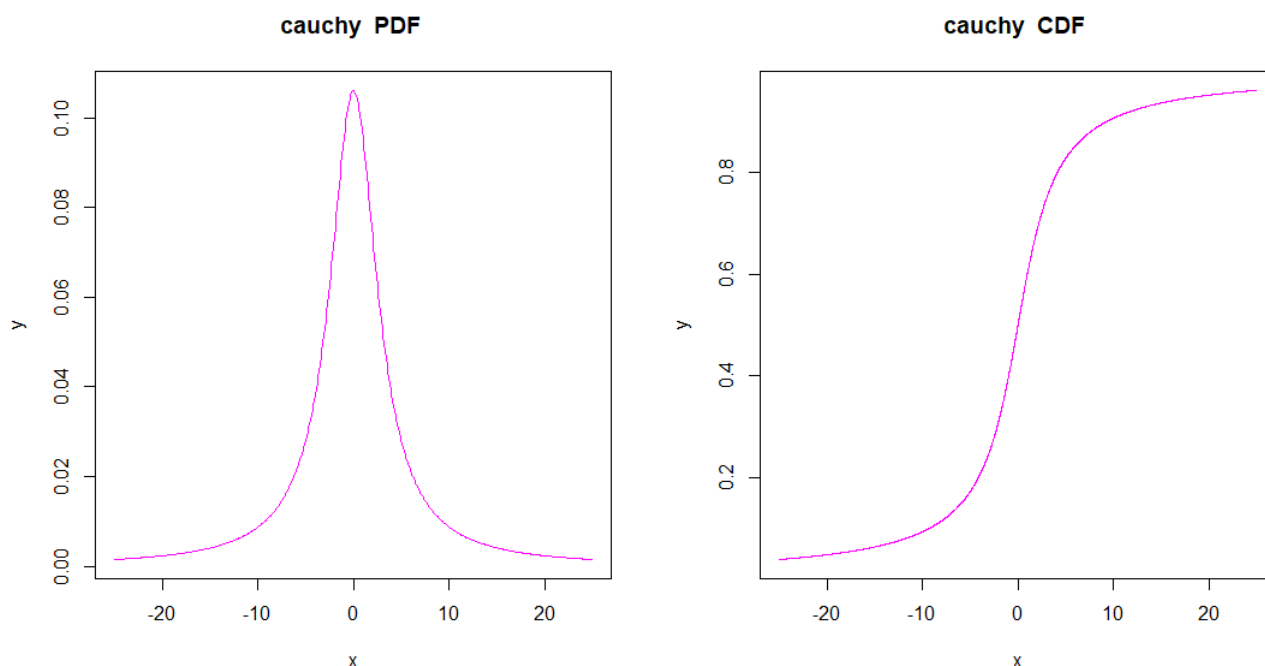


normal PDF



normal CDF





2.4 Exercițiul 5

Calculul mediei, dispersiei și a momentelor inițiale și centrate până la ordinul 4(dacă există). Atunci cand unul dintre momente nu există, se va afișa un mesaj corespunzător către utilizator.

Pentru realizarea cerinței, ne vom folosi de următoarele formule:

$$\begin{aligned}
 E(x) &= \int_{-\infty}^{\infty} x * f(x) dx && \text{Media unei v.a continue} \\
 Var(x) &= \int_{-\infty}^{\infty} (x - E(x))^2 * f(x) dx && \text{Dispersia unei v.a continue} \\
 \mu_r &= \int_{-\infty}^{\infty} (x - E(x))^r * f(x) dx && \text{Moment centrat de ordin r} \\
 m_r &= \int_{-\infty}^{\infty} x^r * f(x) dx && \text{Moment initial de ordin r}
 \end{aligned}$$

Pentru realizarea calculelor, am implementat următoarele funcții:

- `media(f)` - primește funcția ca parametru, returnează valoarea lui $E(x)$
- `dispersia(f)` - primește funcția ca parametru, returnează valoarea lui $Var(x)$
- `momentCentrat(f)` - primește funcția și ordinul ca parametri, returnează valoarea lui μ_r
- `momentInitial(f)` - primește funcția și ordinul ca parametri, returnează valoarea lui m_r

Pentru realizarea celor 4 momente inițiale și 4 momente centrate, am realizat funcția `patruMomente`, care realizează pentru ordinele de la 1 la 4 cele 8 momente.

```

1 media<-function(f){
2   result=tryCatch({
3     return(integrate(function(x){
4       produs<-x*f(x);

```

```

5     return(produs);
6   }, -Inf, Inf)$value);
7 }, eroare = function(e){
8   print("Nu se poate calcula media");
9   return(0);
10  });
11 }
12
13 dispersia<-function(f){
14   medie<-media(f)
15   if(medie==0){
16     print("Nu se poate calcula dispersia fara medie");
17     return(0);
18   }
19   result=tryCatch({
20     return(integrate(function(x){
21       produs<-((x-medic)^2)*f(x);
22       return(produs);
23     }, -Inf, Inf)$value);
24   }, eroare = function(e){
25     print("Nu se poate calcula dispersia");
26     return(0);
27   });
28 }
29
30 momentCentrat<-function(f,ordin){
31   medie<-media(f)
32   if(medie==0){
33     warning(c("Nu se poate calcula momentul centrat fara medie"));
34     return(0);
35   }
36   result=tryCatch({
37     return(integrate(function(x){
38       produs<-((x-medic)^ordin)*f(x);
39       return(produs);
40     }, -Inf, Inf)$value);
41   }, eroare = function(e){
42     print("Nu se poate calcula momentul centrat");
43     return(0);
44   });
45 }
46
47 momentInitial<-function(f,ordin){
48   result=tryCatch({
49     return(integrate(function(x){
50       produs<-((x^ordin)*f(x);
51       return(produs);
52     }, -Inf, Inf)$value);
53   }, eroare = function(e){
54     print("Nu se poate calcula momentul initial");
55     return(0);
56   });
57 }
58
59 patruMomente <- function(f)
60 {
61   print('Primele 4 momente initiale sunt:')
62   print(momentInitial(f, 1))
63   print(momentInitial(f, 2))
64   print(momentInitial(f, 3))
65   print(momentInitial(f, 4))
66   print('Primele 4 momente centrate sunt:')
67   print(momentCentrat(f, 1))
68   print(momentCentrat(f, 2))
69   print(momentCentrat(f, 3))
70   print(momentCentrat(f, 4))
71 }
72
73 #f1 <- function(x){
74 #   if(x > 0 && x < 20)
75 #     return(1/20)
76 #   else
77 #     return(0)
78 #}
79 #patruMomente(f1)
80 # [1] "Primele 4 momente initiale sunt:"

```

```

81 # [1] 5.625
82 # [1] 56.25
83 # [1] 632.8125
84 # [1] 7593.75
85 # [1] "Primele 4 momente centrate sunt:"
86 # [1] 1.40625
87 # [1] 16.69922
88 # [1] 84.04541
89 # [1] 780.5099

```

2.5 Exercițiul 6

Calculul mediei și dispersiei unei variabile aleatoare $g(X)$, unde X are o repartiție continuă cunoscută iar g este o funcție continuă precizată de utilizator.

Pentru realizarea acestui exercitiu am cautat documentatia din cadrul cursului cat si a laboratorului astfel am gasit si am folosit formulele specifice mediei cat si dispersiei unei variabile aleatorii continue.

$$E(x) = \int_{-\infty}^{\infty} x * f(x) dx \quad \text{Media unei v.a continue}$$

$$Var(x) = \int_{-\infty}^{\infty} (x - E(x))^2 * f(x) dx \quad \text{Dispersia unei v.a continue}$$

Pe care le-am adaptat incat sa fie ablicabile in cazul unei variabile aleatoare de tip $g(x)$, si unde X are o repartitie continua cunoscuta. Din moment ce X are o repartitie continua cunoscuta am presupus ca functia densitate de probabilitate a acestuia este cunoscuta, din acest motiv am ales ca functia pentru calcularea mediei cat si pentru calcularea dispersiei sa primeasca ca parametrii functia g , functia densitate de probabilitate pentru X si intervalul acesteia. Cum x -ul din formula mediei va fi inlocuit cu o functie $g(x)$ rezulta ca $f(x)$ va fi reprezentata de functia densitate de probabilitate a lui X .

$$E(x) = \int_{-\infty}^{\infty} g(x) * fX(x) dx, \text{ unde } fX(x) \text{ este functia densitate de probabilitate.}$$

Acelasi lucru se va intampla si pentru formula dispersiei.

$$Var(x) = \int_{-\infty}^{\infty} (g(x) - E(x))^2 * fX(x) dx, \text{ unde } fX(x) \text{ este functia densitate de probabilitate.}$$

```

1  ##Calculul mediei si dispersiei unei variabile aleatoare g(X), unde X are o repartitie
2  continua cunoscuta iar g este o functie continua precizata de utilizator.
3
4  #Verificare functie de repartitie
5  #x1<x2=>f(x1)<f(x2)
6  #f(x-0)=f(x)
7  #lim f(x)=0 cand x -> -Inf
8  #lim f(x)=1 cand x -> Inf
9
10 #Calculare medie
11 calcul_medie<- function(g,functie_dens_prob,lower,upper){
12   integrala_medie <- function(x){g(x)*functie_dens_prob(x)}
13   medie <- integrate(Vectorize(integrala_medie),lower,upper) $ value
14   return(medie)
15 }
16
17 #Calculare dispersie
18 calcul_dispersie <- function(g,functie_dens_prob,lower,upper){
19   integrala_dispersie <- function(x){(g(x)-calcul_medie(g,functie_dens_prob,lower,upper))
20   ^2*functie_dens_prob(x)}
21   dispersie <- integrate(Vectorize(integrala_dispersie),lower,upper) $ value
22   return(dispersie)

```

```

21 }
22
23 #Testare
24 #
25 # f1 <- function(x)(2*(x^3))
26 # f2 <- function(x)(exp(1)^(-2*x))
27 # calcul_medie(f1,f2,0,Inf)
28 # calcul_dispersie(f1,f2,0,Inf)

```

2.6 Exercițiul 8

Afisarea unei “fise de sinteza” care sa contina informatii de baza despre respectiva repartitie(cu precizarea sursei informatiei!). Relevant aici ar fi sa precizati pentru ce e folosita in mod uzual acea repartitie, semnificatia parametrilor, media, dispersia etc.

Pentru a putea afisa o fisa de sinteza, am utilizat libraria ”R6”, mai exact obiecte de tip ”R6Class” alaturi de obiectul ”Dictionary” din libraria ”ml3misc”(https://www.rdocumentation.org/packages/mlr3misc/versions/0.5.0/topics/Dictionary).

Astfel, pe baza de object oriented programming, am realizat o clasa cu structura urmatoare:

- Sursa
- Notatie
- Parametri
- Domeniu
- PDF
- CDF
- Media
- Mediana
- Utilizari

Structura aceasta a fost organizata similar felului in care repartitiile sunt prezentate pe Wikipedia. In lista de simteza se regasesc repartitii, atat din curs, cat si din studiu suplimentar(Cauchy).

```

1  library(mlr3misc)
2  library(R6)
3
4  # Cerinta: Afi??area unei "fi??e de sinteza" care sa con??ina informa??ii de baza despre
5  # reparti??ie(cu precizarea sursei informa??iei!). Relevant aici ar fi sa preciza??i
6  # folosita ??n mod uzual acea reparti??ie, semnifica??ia parametrilor, media, dispersia
7  # etc.
8
9  data = Dictionary$new()
10
11  item = R6Class("Item")
12
13      # UNIFORM #
14
15  data_ob = Dictionary$new()
16
17  data_ob$add("Sursa", R6Class(public = list(print = "Curs + Wikipedia")))
18
19  data_ob$add("Notatie", R6Class(public = list(print = "U(a, b)")))
20

```

```

21 data_ob$add("Parametri", R6Class(public = list(print = "-Inf < a < b < Inf")))
22
23 data_ob$add("Domeniu", R6Class(public = list(print = "x in [a, b]")))
24
25 data_ob$add("PDF", R6Class(public = list(print = "1 / (b - a)")))
26
27 data_ob$add("CDF", R6Class(public = list(print = "(x - a) / (b - a)")))
28
29 data_ob$add("Media", R6Class(public = list(print = "(1 / 2) * (a + b)")))
30
31 data_ob$add("Mediana", R6Class(public = list(print = "(1 / 2) * (a + b)")))
32
33 data_ob$add("Utilizari", R6Class(public = list(print = "1. In domeniul economiei pentru
34 inventariere, in special pentru analiza ciclului de viata a unui produs nou.
35 2. Solutie pentru erorile de
36 cuantificare, un exemplu fiind conversia analog-to-digital.")))
37 data$add("uniform", data_ob)
38
39 # EXPONENTIAL #
40
41 data_ob = Dictionary$new()
42 data_ob$add("Sursa", R6Class(public = list(print = "Curs + Wikipedia")))
43
44 data_ob$add("Parametri", R6Class(public = list(print = "labmda > 0")))
45
46 data_ob$add("Domeniu", R6Class(public = list(print = "x in [0, +Inf]")))
47
48 data_ob$add("PDF", R6Class(public = list(print = "lambda * exp(1)^(-lambda * x)")))
49
50 data_ob$add("CDF", R6Class(public = list(print = "1 - exp(1)^(-lambda * x)")))
51
52 data_ob$add("Media", R6Class(public = list(print = "1 / labmda")))
53
54 data_ob$add("Mediana", R6Class(public = list(print = "ln2 / lambda")))
55
56 data_ob$add("Utilizari", R6Class(public = list(print = "Pe baza unui set de date, aceasta
57 distributie poate estima precis evenimente care vor avea lic in viitor.
58 De exemplu: durata de
59 descompunere a unei particule radioactive;
60 durata dintre 'click
61 '-urile unui detector Geiger.")))
62 data$add("exponential", data_ob)
63
64 # NORMAL #
65
66 data_ob = Dictionary$new()
67 data_ob$add("Sursa", R6Class(public = list(print = "Curs + Wikipedia")))
68
69 data_ob$add("Notatie", R6Class(public = list(print = "N(mu, sigma^2)")))
70
71 data_ob$add("Parametri", R6Class(public = list(print = "mu in R, sigma ^ 2 >= 0")))
72
73 data_ob$add("Domeniu", R6Class(public = list(print = "x in [-Inf, +Inf]")))
74
75 data_ob$add("PDF", R6Class(public = list(print = "(1 / (sigma * sqrt(pi * 2)))*(exp(1)
76 ^((-x - mu)^2)/(2 * sigma ^ 2)"))))
77
78 data_ob$add("CDF", R6Class(public = list(print = "(1 / 2) * (1 + erf((x - mu) / (sigma *
79 sqrt(2))))"))))
80
81 data_ob$add("Media", R6Class(public = list(print = "mu")))
82
83 data_ob$add("Mediana", R6Class(public = list(print = "mu")))
84
85 data_ob$add("Utilizari", R6Class(public = list(print = "Numeroase aplicatii, variand de la
86 fizica pana la biologie, pana si la statisticarezultatelor unui examen. ")))
87
88 data$add("normal", data_ob)
89
90 # CAUCHY #

```

```

89 data_ob = Dictionary$new()
90 data_ob$add("Sursa", R6Class(public = list(print = "Curs + Wikipedia")))
91
92 data_ob$add("Parametrii", R6Class(public = list(print = "x0, y > 0")))
93
94 data_ob$add("Domeniu", R6Class(public = list(print = "x in [-Inf, Inf]")))
95
96 data_ob$add("CDF", R6Class(public = list(print = "(1 / pi) * arctan((x - x0) / y) + (1/2)"
97 )))
98
99 data_ob$add("PDF", R6Class(public = list(print = "1 / (pi * y * (1 + (x - x0) / y)^2)")))
100
101 data_ob$add("Mediana", R6Class(public = list(print = "x0")))
102
103 data_ob$add("Media", R6Class(public = list(print = "undefined")))
104
105 data_ob$add("Utilizari", R6Class(public = list(print = "Utilizata in hidrologie,
106 distributia Cauchy poate fi aplicata fenomenelor extreme, precum furtuni de o zi sau
107 inundatii.")))
108
109
110 # FUNCTIE DE AFISARE #
111
112 afisare <- function(distribution_name) {
113
114   out_data = data$get(distribution_name)
115
116   for (key in out_data$keys()) {
117
118     print(noquote(paste(key, ": ", out_data$get(key)$print, sep="")))
119
120   }
121 }
122
123 # EXEMPLE
124 #afisare("uniform")
125 #afisare("normal")
126 #afisare("exponential")
127 #afisare("cauchy")

```

2.7 Exercițiul 9

Generarea a n valori (unde n este precizat de utilizator!) dintr-o repartiție de variabile aleatoare continue (solicitați material suport pentru partea de simulare).

Pentru realizarea cerinței, am aplicat metoda inversa pentru simularea variabilelor aleatoare conform materialului suport. In cod, am implementat urmatoarele functii:

- `integrarePDF(f, x)` - primeste functia si capatul din dreapta a integralei ca parametri, returneaza functia densitate de probabilitate
- `generareInversa(f,y,stanga,dreapta)` - primeste ca parametri functia, y care va fi noul parametru al functiei inverse, capetele din stanga si dreapta ale intervalului de definire , returneaza functia inversa
- `generareNValori(f)` - primeste ca parametri functia, numarul de valori care urmeaza sa fie generate, capetele din stanga si dreapta ale intervalului de definire , afiseaza sirul generat de valori

```

1 integrarePDF<-function(f, x){
2   tryCatch({
3     return(integrate(Vectorize(f), 0, x)$value);
4   }, eroare = function(e){
5     print(e);
6     return(0);
7   });

```

```

8 }
9
10 genereareInversa<-function(f, y, stanga, dreapta){
11   uniroot((function(x) f(x) - y), lower=stanga, upper=dreapta)[1]
12 }
13
14 genereareNValori<-function(f, n, stanga, dreapta){
15   valori <- runif(n, stanga, dreapta)
16   export <- c();
17   for(y in valori){
18     export <- append(export, genereareInversa(function(x) (integrarePDF(f, x)), y, stanga,
19     dreapta)$root)
20   }
21   print(export);
22 }
23
24 #f1<-function(x){
25 #   if(x >= 0)
26 #     return(1-exp(-2*x))
27 #   else return(0)
28 #}
29 #genereareNValori(f1,50,0.1,100)
30 # [1] 25.446174 53.115397 74.557649 68.807575 30.584462 49.337176 71.337045 72.074684
31 18.379085 53.966918 76.665689 94.289155 95.916681
32 # [14] 18.057581 74.451717 42.136329 62.783452 13.919849 95.225450 89.564206 74.636752
33 65.059950 11.614840 47.619160 89.173798 46.462880
34 # [27] 66.338070 81.690513 30.974663 18.989209 32.714365 59.707631 86.227731 46.973578
35 93.553083 41.559197 28.137667 35.568942 44.815815
36 # [40] 7.876199 81.646375 91.597387 71.536997 52.775758 53.720603 75.384789 3.531075
37 18.790562 77.463315 38.960205

```

2.8 Exercițiul 10

Calculul covarianței și coeficientului de corelație pentru două variabile aleatoare continue (Atenție: Trebuie să folosiți densitatea comună a celor două variabile aleatoare!)

Formula covarianței pentru variabilele aleatoare continue este: $cov(X, Y) = E(XY) - E(X)E(Y)$, unde $E[X]$ reprezintă media variabilei aleatoare continue. Pentru calcularea $E(XY)$ se calculează media folosind integrala dublă: $E(XY) = \int_c^d \int_a^b x * y * f(x, y) dx dy$

Pentru a calcula corelația, folosim următoarea formulă: $cor(X, Y) = \frac{cov(X, Y)}{\sqrt{D(X)D(Y)}}$, unde $D(X)$ reprezintă dispersia variabilei X .

```

1 library(pracma)
2
3 media<-function(f, fst=-Inf, fdr=Inf){
4   result=tryCatch({
5     return(integrate(function(x){
6       produs<-x*f(x);
7       return(produs);
8     }, fst, fdr)$value);
9   }, error = function(e){
10    print("Nu se poate calcula media");
11    return(0);
12  });
13 }
14
15 dispersia<-function(f, fst=-Inf, fdr=Inf){
16   medie<-media(f, fst, fdr)
17   if(medie==0){
18     print("Nu se poate calcula dispersia fara medie");
19     return(0);
20   }
21   result=tryCatch({
22     return(integrate(function(x){
23       produs<-((x-medie)^2)*f(x);
24       return(produs);
25     }, fst, fdr)$value);
26   }, error = function(e){
27     print("Nu se poate calcula dispersia");

```



```

28     return(0);
29   });
30 }
31
32 densitate_marginalaX <- function(f,a,b,c = -Inf,d = Inf)
33 {
34   a <- max(a,c) #se compara maximul intre limitele inferioare dintre intervalul variabilei
35   opuse si cel conditional
36   b <- min(b,d) #se compara minimul intre limitele superioare dintre intervalul variabilei
37   opuse si cel conditional
38   tryCatch(
39     {
40       if(a>b) 0 #se compara capetele intervalului pentru a nu utiliza un interval
41       imposibil
42       else function(x) {integrate((function(y) {f(x, y)}), a, b)$value} #folosim formula
43       de calcul din teorema
44     }, error = function(e){
45       print("Functia de densitate comuna este incorecta!");
46     }
47   )
48 }
49
50 densitate_marginalaY <- function(f,a,b,c = -Inf,d = Inf)
51 {
52   a <- max(a,c) #se compara maximul intre limitele inferioare dintre intervalul variabilei
53   opuse si cel conditional
54   b <- min(b,d) #se compara minimul intre limitele superioare dintre intervalul variabilei
55   opuse si cel conditional
56   tryCatch(
57     {
58       if(a>b) 0 #se compara capetele intervalului pentru a nu utiliza un interval
59       imposibil
60       else function(y) {integrate((function(x) {f(x, y)}), a, b)$value} #folosim formula
61       de calcul din teorema
62     }, error = function(e){
63       print("Functia de densitate comuna este incorecta!");
64     }
65   )
66 }
67
68 covarianta_corelatia<-function(f, xst, xdr, yst, ydr){
69   fx = densitate_marginalaX(f, xst, xdr)
70   fy = densitate_marginalaY(f, yst, ydr)
71
72   medieX = media(fx, xst, xdr)
73   medieY = media(fy, yst, ydr)
74
75   functie<-function(x,y){return(x*y*f(x,y))}
76
77   cov = integral2(functie, xst, xdr, yst, ydr)$Q - (medieX*medieY)
78
79   dispersieX = dispersia(fx, xst, xdr)
80   dispersieY = dispersia(fy, yst, ydr)
81
82   cor = cov / (sqrt(dispersieX)*sqrt(dispersieY))
83
84   print(cov)
85   print(cor)
86 }
87
88 f<-function(x,y){
89   return(3/2*(x^2+y^2))
90 }
91
92 #covarianta_corelatia(f, 0, 1, 0, 1)
93 # [1] 0.125
94 # [1] 1.5

```

2.9 Exercițiul 11

Vom calcula densitatea marginala folosind teorema acesteia. In cazul functiei, vom avea 5 parametrii, unde primul parametru reprezinta functia de densitatea comuna a celor doua variabile, al 2-lea si al 3-lea parametru sunt capetele intervalului domeniului variabilei opuse iar parametrii 4 si 5 sunt optionali si vor fi folositi pentru calculul densitatii conditionate, unde acestia reprezinta capetele intervalului conditional. Am facut doua functii, una pentru fiecare

variabila aleatoare continua, pentru a calcula integrala in functie de pdf-ul variabilei alese.

Ne vom folosi de urmatoarea teorema: Fie X si Y doua variabile aleatoare continue, avand densitatea comuna continua $f(x,y)$ cu domeniul $[a,b] \times [c,d]$. Pdf-urile marginale sunt:

$$f_X(x) = \int_c^d f(x,y)dy \quad \text{si} \quad f_Y(y) = \int_a^b f(x,y)dx$$

```

1 densitate_marginalaX <- function(f,a,b,c = -Inf,d = Inf)
2 {
3   a <- max(a,c) #se compara maximul intre limitele inferioare dintre intervalul variabilei
4   opuse si cel conditional
5   b <- min(b,d) #se compara minimul intre limitele superioare dintre intervalul variabilei
6   opuse si cel conditional
7   tryCatch(
8     {
9       if(a>b) 0 #se compara capetele intervalului pentru a nu utiliza un interval imposibil
10      else function(x) (integrate(Vectorize(function(y) (f(x, y))), a, b)$value) #folosim
11      formula de calcul din teorema
12    }, eroare = function(e){
13      print("Functia de densitate comuna este incorecta!");
14    }
15  )
16 }
17 densitate_marginalaY <- function(f,a,b,c = -Inf,d = Inf)
18 {
19   a <- max(a,c) #se compara maximul intre limitele inferioare dintre intervalul variabilei
20   opuse si cel conditional
21   b <- min(b,d) #se compara minimul intre limitele superioare dintre intervalul variabilei
22   opuse si cel conditional
23   tryCatch(
24     {
25       if(a>b) 0 #se compara capetele intervalului pentru a nu utiliza un interval
26      imposibil
27      else function(y) (integrate(Vectorize(function(x) (f(x, y))), a, b)$value) #folosim
28      formula de calcul din teorema
29    }, eroare = function(e){
30      print("Functia de densitate comuna este incorecta!");
31    }
32  )
33 }

```

2.10 Exerciitiul 12

Construirea sumei și diferenței a două variabile aleatoare continue independente(folosiți formula de convoluție)

Pentru construirea sumei si diferentei folosind formula de convolutie m-am documentat de pe: <https://www.afahc.ro/ro/facultate/cursuri/ccg/MSE/C07%20-%20Convolutia%20semnalelor.pdf> pentru a obtine formula generala de convolutie analogica. M-am documentat despre suma si diferenta variabilelor aleatorii continue de pe: <https://math.stackexchange.com/questions/2628366/why-is-the-pdf-of-the-sum-of-two-continuous-random-variables-the-convolution-of>. Apoi am adaptat formula generala in functie de suma/diferenta variabilelor aleatorii continue.

Formula generala de convolutie analogica este: $a(n) = \int_{-\infty}^{\infty} f(\tau) * g(n - \tau) d\tau$. Asadar deoarece suma a 2 variabile aleatorii este $Z=X+Y$ atunci $Y=Z-X$ deci formula de convolutie pentru suma va fi egala cu: $a(z) = \int_{-\infty}^{\infty} f(z - x) * g(x) dx$, iar pentru diferenta $Z=X-Y$ atunci $Y=X-Z$ formula devine: $a(z) = \int_{-\infty}^{\infty} f(x - z) * g(x) dx$.

```

1 ##Construirea sumei si diferentei a doua variabile aleatoare continue independente(
2   folositi formula de convolutie)

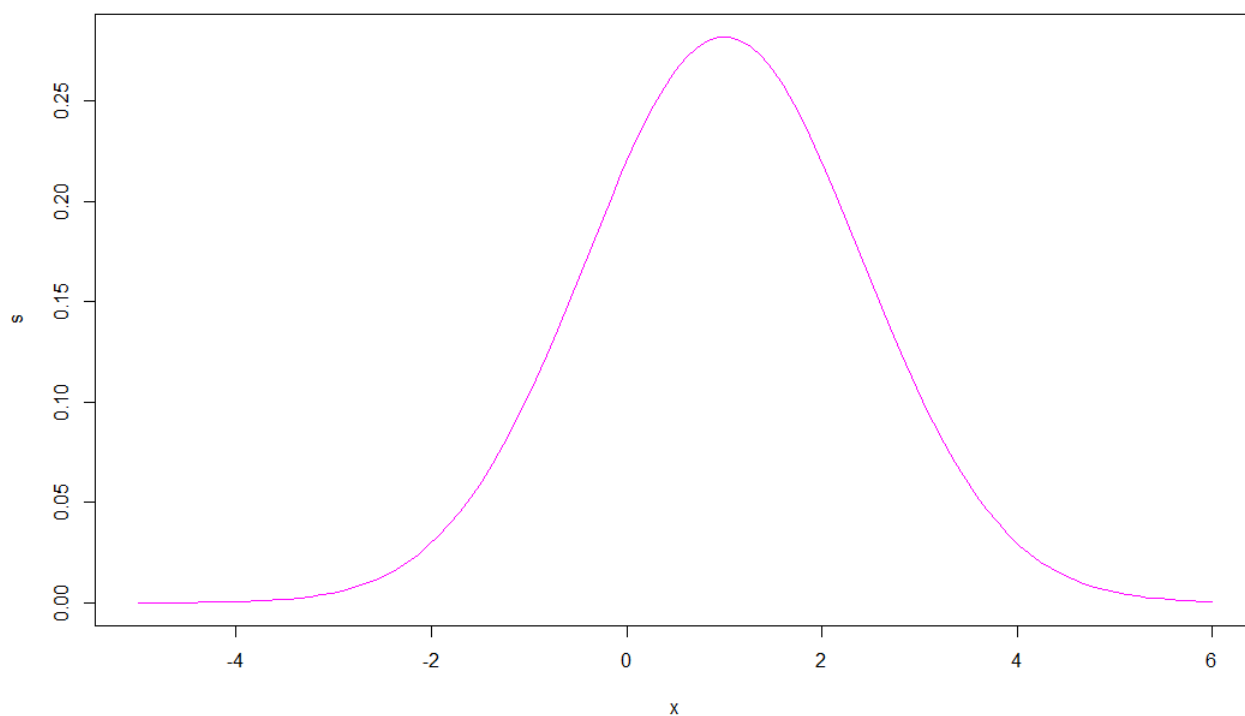
```

```

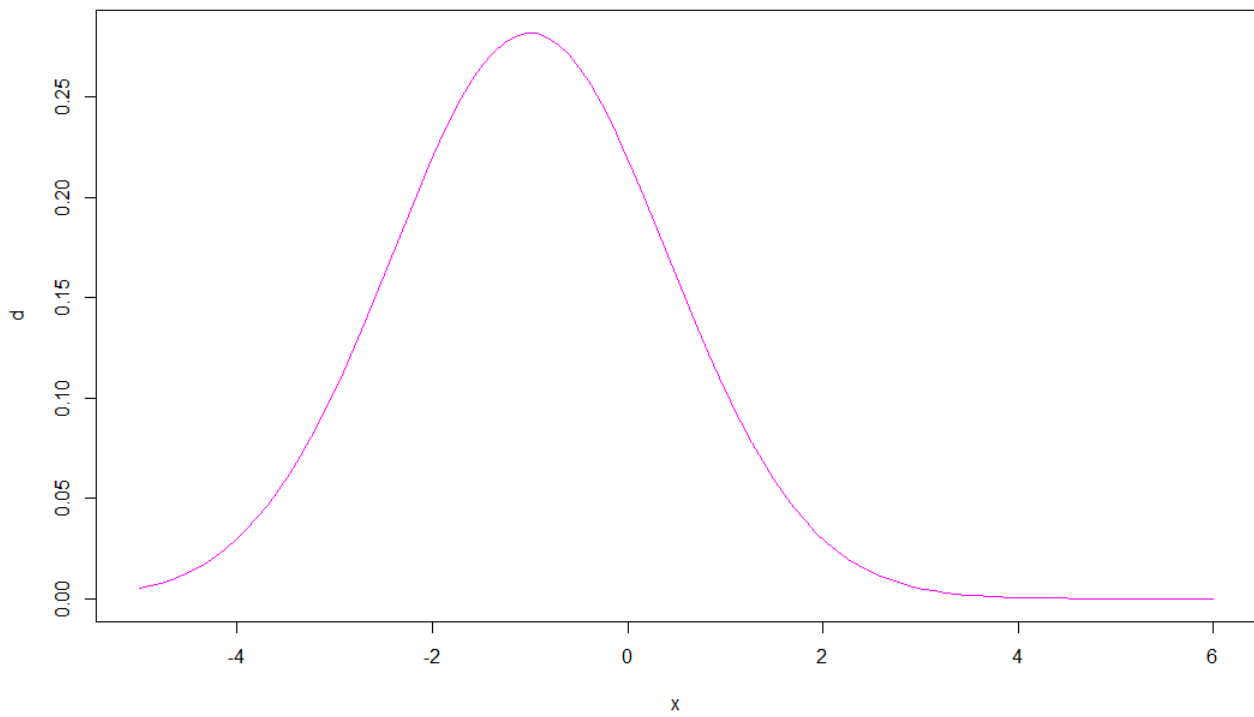
3 #Formula de convolutie analogica
4 #a(n)=Integrala(x(t)*y(n-t))
5
6
7 #Calcularea sumei folosind functia de convolutie Z=X+Y unde X si Y sunt variabile
8 aleatorii => Y=Z-X
9 suma_convolutie <- function (f,g){function(z){
10   integrala_con_sum <- function(x){g(z-x)*f(x)}
11   sum <- integrate(Vectorize(integrala_con_sum),-Inf,Inf) $ value
12   return(sum)}
13 }
14
15 #Calcularea diferentei folosind functia de convolutie Z=X-Y unde X si Y sunt variabile
16 aleatorii => Y=X-Z
17 diferenta_convolutie <- function (f,g){function(z){
18   integrala_con_dif <- function(x){g(x-z)*f(x)}
19   dif <- integrate(Vectorize(integrala_con_dif),-Inf,Inf) $ value
20   return(dif)}
21 }
22
23 ##Testare
24 # f <- function(x)(dnorm(x))
25 # g <- function(x) (dnorm(x,mean=1))
26 # s <- Vectorize(suma_convolutie(f,g))
27 # d <- Vectorize(diferenta_convolutie(f,g))
28 #
29 # plot(f,from=-5,to=6,type="l",col ="magenta")
30 # plot(g,from=-5,to=6,type="l",col ="magenta")
31 # plot(s,from=-5,to=6,type="l",col ="magenta")
32 # plot(d,from=-5,to=6,type="l",col ="magenta")

```

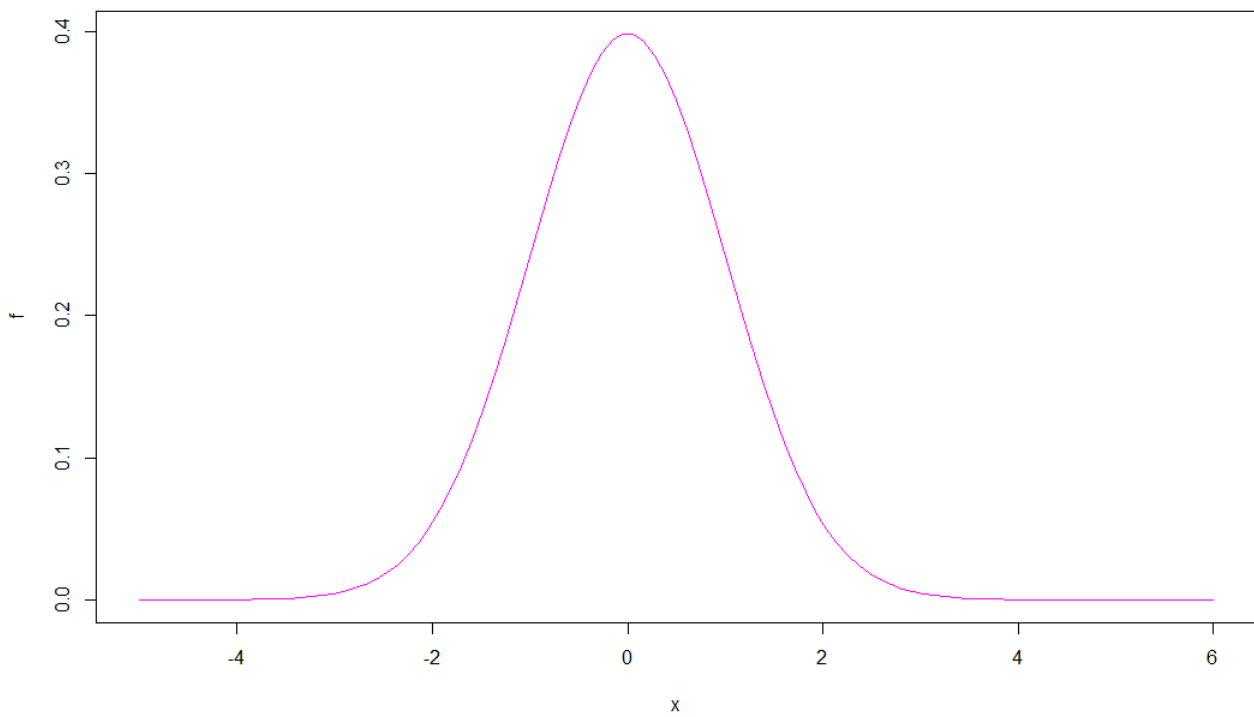
Graficul sumei.



Graficul diferentei.



Funcție densitate de probabilitate v.a.c 1.



Funcție densitate de probabilitate v.a.c 2.

