Documentatie proiect 1 - r

Grupa 232

Negulescu Stefan (TL)

Hirnnea Gabriel

Cuprins

[Problema 1 2](#_Toc63255815)

[Problema 2 3](#_Toc63255816)

[Problema 4 4](#_Toc63255817)

[Problema 5 6](#_Toc63255818)

[Problema 6 8](#_Toc63255819)

[Problema 7 9](#_Toc63255820)

[Problema 10 11](#_Toc63255821)

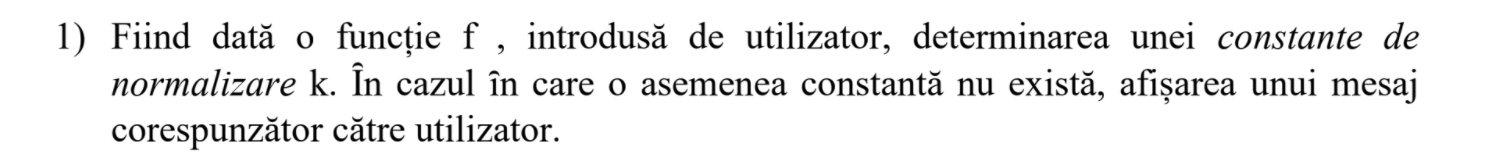
[Problema 11 14](#_Toc63255822)

[Problema 12 15](#_Toc63255823)

[Concluzie 16](#_Toc63255824)

[Surse 16](#_Toc63255825)

# Problema 1



Functia are 3 parametrii: functia f si parametrii a si b, ce reprezinta intervalul [a,b] pe care este definite functia respective. Se calculeaza integral functiei f pe intervalul [a,b]. Deoarece aceasta trebuie sa fie egala cu 1, constanta de normalizare va fi egala cu 1 / integral respective, densitatea de probabilitate fiind egala cu kf(x).

# 1

normConst <- function(f, a, b)

{

1 / integrate(Vectorize(f), a, b)$value

}

# Problema 2



Verificare\_densitate e o functie cu un paramentru. Prin singurul parametru se transmite functia care urmeaza a fi verificata.

Se va testa daca integrala de la -Inf la +Inf este 1 apoi pentru a verfica daca functia data este pozitiva se va lua un interval de valori(-10000 la 10000) cu pasul 0.001 si se vor verfica valorile functiei pe acest interval. Daca toate valorile vor fi > 0 se va afisa pe ecran "Functia este densitate de probabilitate" altfel "Functia nu este densitate de probabilitate".

# 2

verificare\_densitate <- function(functie)

{

ok <- 0

y <- integrate(Vectorize(functie), -Inf, Inf)

for(val in seq(-10000,10000,by=0.001))

{

if(functie(val)<0)

{

ok <- 1

break;

}

}

z <- round(y$value,digits=7)

if(ok==0&&z==1)

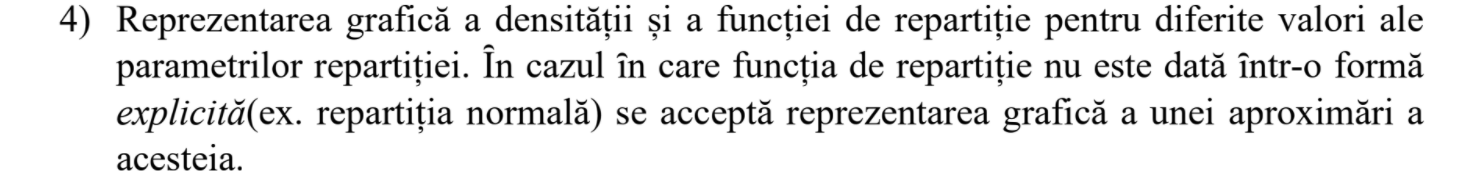
print("Functia este densitate de probabilitate")

else

print("Functia nu este densitate de probabilitate")

}

# Problema 4



Functia reprezentare\_grafica e o functie cu 3 parametri: primul paramtru densitatea v.a continue si celealte doua vor constitui intervalul pe care se va construi graficul. Se va construi F(x) functia de repartitie cu formula clasica din curs apoi se va contrui folosind functia plot un grafic cu cele doua functii.

# 4

reprezentare\_grafica <- function(functie,a,b)

{

x <- seq(a,b,0.01)

y <- c()

y1 <- c()

F1 <- function(x,functie)

{

y <- integrate(Vectorize(functie), -Inf, x)

return(y$value)

}

for(i in x)

{

y <- c(y,F1(i,functie))

}

for(i in x)

{

y1 <- c(y1,functie(i))

}

plot(x,y, col="red",main="Reprezentare grafica densitate si functie de repartitie",type = "l")

lines(x,y1, col="blue")

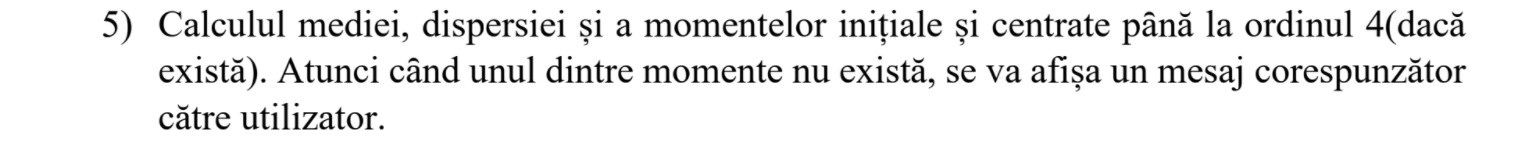
legend("topleft",

c("Densitate","Functia de repartitie"),

fill=c("blue","red"))

}

# Problema 5



Functia **media** are ca parametrii functia f, cat si a si b, ce reprezinta intervalul pe care f este definite [a,b]. Aceasta returneaza media functiei care este egala cu integral produsului x\*f(x)\*k pe intervalul [a,b].

Functia **media2** calculeaza E(X^2).

Functia **dispersia** are ca parametrii f,a,b cu acelasi rol ca in functia media. Aceasta calculeaza dispersia lui f prin formula E(X^2) – E(X)^2.

Functia **momentInitial** are ca parametrii f,a,b cu aveleasi roluri, iar r reprezinta rangul momentului. Calculeaza momentul initial de rang r folosind formula respective.

Functia **momentCentrat** are aceeasi structura ca momentInitial, diferind formula de calcul pentru momentul centrat de rang r.

# 5

# Media

media <- function(f,a = -Inf,b = Inf)

{

aux <- function(x)

{

x \* f(x) \* normConst(f,a,b)

}

integrate(aux,a,b)$value

}

media2 <- function(f,a = -Inf,b = Inf)

{

aux <- function(x)

{

x ^ 2 \* f(x) \* normConst(f,a,b)

}

integrate(aux,a,b)$value

}

# Dispersia

dispersia <- function(f, a = -Inf, b = Inf)

{

media2(f,a,b) - media(f,a,b) ^ 2

}

# Momente

momentInitial <- function(f,a,b,r)

{

aux <- function(x)

{

x ^ r \* f(x)

}

integrate(aux,a,b)$value

}

momentCentrat <- function(f,a,b,r)

{

aux <- function(x)

{

medie <- media(f,a,b)

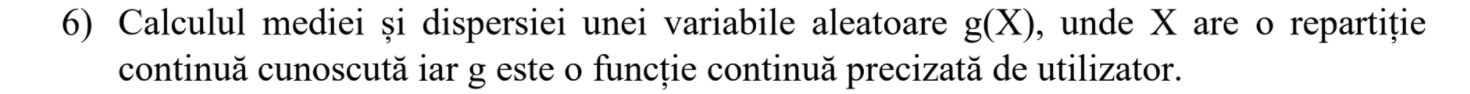
(x - medie) ^ r \* f(x)

}

integrate(aux,a,b)$value

}

# Problema 6



S-au contruit doua functii medie\_g si disperie\_g

Prima functie medie\_g va lua ca parametru functia contiuna g impreuna cu densitatea variablilei aleatoare continue X. Valoarea medie se va calcula folosind formula din curs.

Functia dispersie\_g are aceeasi parametri ca functia medie\_g. Pentru calcului dispersiei se va folosi formula Var(X)=E(X^2)-E(X).

medie\_g <- function(g,densitate)

{

force(g)

force(densitate)

y <- integrate(Vectorize(function(x)(g(x)\*densitate(x))), -Inf, Inf)

return(y$value)

}

dispersie\_g <- function(g,densitate)

{

force(g)

force(densitate)

h <- function(x){g(x)\*densitate(x)}

y <- integrate(Vectorize(h), -Inf, Inf)

a <- y$value

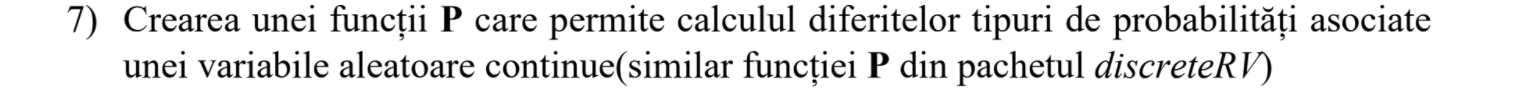
force(h)

k <- function(x){g(x)\*h(x)}

y <- integrate(Vectorize(k), -Inf, Inf)

b <- y$value return(b-a^2)}

# Problema 7



Functia **P** are ca parametrii functia f, a si b, ce reprezinta intervalul pe care este definite functia,

x si y, ce reprezinta parametrii unei probabilitati de tipul P(x <= X <= y), x1 si y1, ce reprezinta parametrii unei probabilitati de tipul P(x <= X <= y | x1 <= X <= y1), cu mentiunea ca in acest caz x1 sau y1 trebuie sa fie egale cu a sau b, iar cealalta sa fie introdusa de utilizator.

Aceasta functie calculeaza diferite probabilitati folosindu-se de integrale din pmf, pe un interval introdus de utilizator.

# 7

P <- function(f,a,b,x = a,y = b,x1 = a, y1 = b)

{

pmf <- function(x)

{

f(x) \* normConst(f,a,b)

}

if(x == y)

0

else

if(x < y & x1 == a & y1 == b)

{

if(x < a)

x <- a

if(y > b)

y <- b

integrate(Vectorize(pmf),x,y)$value

}

else

if(x < y & x1 < y1)

{

if(x1 < a)

x1 <- a

if(y1 > b)

y1 <- b

mini <- min(x,x1)

maxi <- max(y,y1)

if(mini < a)

mini <- a

if(maxi > b)

maxi <- b

integrate(Vectorize(pmf),mini,maxi)$value / integrate(Vectorize(pmf),x1,y1)$value

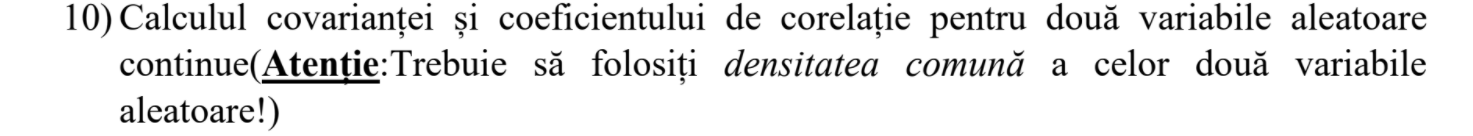
}

else

warning("Date incorecte!")

}

# Problema 10



Functiile **fX** si **fY** calculeaza pdf marginala pentru X, respective Y. Acesteaza au ca parametrii functie pmf, o valoare, si un interval, ce reprezinta intervalul lui Y in cazul lui fX si intervalul lui X in cazul lui fY. Acestea aplica formula pdf marginala.

Functia **cov** calculeaza covarianta a doua variabile aleatoare continue utlizand formula Cov(x,y) = E(XY) – E(X)\*E(Y). E(XY), E(X), E(Y) sunt calculate cu ajutorul fX si fY, care sunt calculate pronind de la pdf comuna. Aceasta are ca parametrii functia pmf comuna, a si b, reprezentand intervalul lui X, c si d, reprezentand intervalul lui Y.

Functia **cor** calculeaza coeficientul de corelatie dintre 2 variabile aleatoare continue, pornind de la pdf comuna, utlizand formula Cor(X,Y) = Cov(X,Y) / sqrt(Var(X)\*Var(Y)).

Parametrii acestei functii au acelasi rol ca la functia cov.

Var(X) este calculata cu formula Var(X)=E(X^2)-E(X), acestea fiind calculate cu ajutorul fX si fY.

Am utilizat functia din integral2 din pachetul “pracma” pentru a calcula integrale duble, folosite in calculul Cov.

# 10

library(pracma)

install.packages("pracma")

fX <- function(f,x,a,b)

{

aux <- function(y)

{

f(x,y)

}

integrate(Vectorize(aux),a,b)$value

}

fY <- function(f,y,a,b)

{

aux <- function(x)

{

f(x,y)

}

integrate(Vectorize(aux),a,b)$value

}

cov <- function(f,a,b,c,d)

{

aux <- function(x,y)

{

x \* y \* f(x,y)

}

fX1 <- function(x)

{

x \* fX(f,x,c,d)

}

fY1 <- function(y)

{

y \* fY(f,y,a,b)

}

integral2(aux,c,d,a,b)$Q - integrate(Vectorize(fX1),a,b)$value \* integrate(Vectorize(fY1),c,d)$value

}

cor <- function(f,a,b,c,d)

{

fX1 <- function(x)

{

x \* fX(f,x,c,d)

}

fY1 <- function(y)

{

y \* fY(f,y,a,b)

}

fX2 <- function(x)

{

fX1(x^2) \* fX1(x)^2

}

fY2 <- function(y)

{

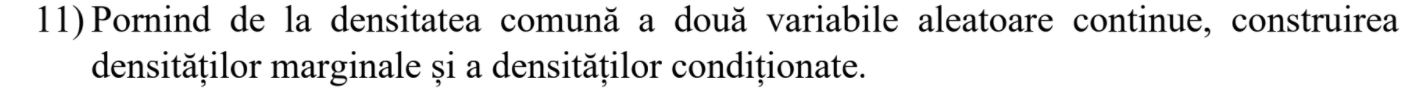
fY1(y^2) \* fY1(y)^2

}

cov(f,a,b,c,d) / sqrt(integrate(Vectorize(fX2),a,b)$value \* integrate(Vectorize(fY2),c,d)$value)

}

# Problema 11



Functiile **fX** si **fY** au fost prezentate la Problema 10.

Functiile **Px** si **Py** calculeaza cdf conditionate. Ambele au ca parametrii functia pdf comuna, cat si intervalele cu ajutorul carora se calculeza cdf. Parametrii a si b reprezinta intervalul probabilitatii P(a <= X <= y), iar c si d reprezinta intervalul pe care este definite Y in cazul Px si X in cazul Py. Functiile integreaza pe [a,b] x\*fX, respective y\*fY.

# 11

Px <- function(f,a,b,c,d)

{

fX1 <- function(x)

{

fX(f,x,c,d)

}

integrate(Vectorize(fX1),a,b)$value

}

Py <- function(f,a,b,c,d)

{

fY1 <- function(x)

{

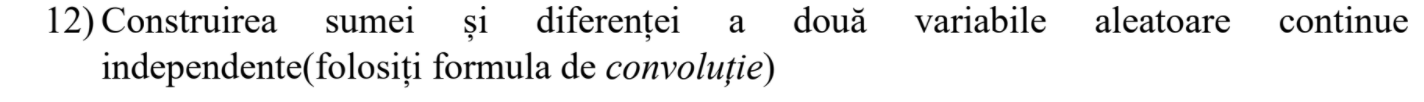
x \* fY(f,x,c,d)

}

integrate(Vectorize(fY1),a,b)$value

}

# Problema 12



Pentru rezolvarea acestei cerinte am construit una pentru suma: sum\_var si una pentru diferenta: dif\_var fiecare avand cate 3 parametri. Primul parametru densitatea primei v.a continuie al doileaparamentru densitatea celei de-a doua v. a continue, iar al treilea valoarea in care se doreste aflarea operatiei.

Pentru calcului sumei se va folosi formula de convolutie.

Analog pentru calcului diferentei.

# 12

sum\_var <- function(f,g,val)

{

force(g)

force(f)

sum <- function(z) (integrate (Vectorize(function(t) (f(t) \* g(z - t))), -Inf, +Inf))

return(sum(val))

}

dif\_var <- function(f,g,val)

{

force(g)

force(f)

dif <- function(z) (integrate (Vectorize(function(t) (f(t) \* g(t - z))), -Inf, +Inf))

return(dif(val))

}

# Concluzie

Proiectul are ca material de baza cursurile, cat si laboratoarele de Probabilitati.

A fost necesara utilizarea functiei **Vectorize** in calculul integralelor pentru a nu aparea eventuale erori.

Pentru calculul integralelor duble am utlizat functia **integral2** din pachetul **pracma**.

In concluzie, am realizat functiile necesare pentru 9 probleme, toate functiile fiind folositoare in diferite calculi pentru variabile aleatoare continue.

## Surse

Vectorize - <https://stat.ethz.ch/R-manual/R-devel/library/base/html/Vectorize.html>

Integral2 - <https://www.rdocumentation.org/packages/pracma/versions/1.9.9/topics/integral2>

Force - <https://www.rdocumentation.org/packages/base/versions/3.6.2/topics/force#:~:text=force%20forces%20the%20evaluation%20of,loop%20or%20an%20apply%20function>.