Grupa: 244 - Donici Alexandra-Maria
Muscalu Elena-Claudia

Proiect-Probabilităţi şi Statistică

PROBLEMA 1

Considerăm următoarele distribuţii: **B(n, p), Pois(λ), Exp(λ), N (μ, σ2)**

1. Generăm 1000 de realizări independente din fiecare repartiţie de mai sus, apoi le vom calcula media şi varianţa eşantionului folosind formulele din curs.

```
#generarea a 1000 de realizari independente
b <- rbinom(1000,size=10,p=0.7)
p <- rpois(1000, lambda=3)
e <- rexp(1000,3)
n <- rnorm(1000, mean=0, sd=1)

# varianta esantionului
variance <- function(vec)
{
  range(vec)
  counts <- hist(vec + .01, breaks = 7)$counts
  fx <- counts / (sum(counts))
  x <- c(min(vec): max(vec))
  exp <- sum(x * fx)  #media
  exp.square <- sum(x^2 * fx)
  var <- exp.square - (exp)^2 ; var #varianta
}

#media esantionului
mean <- function(vec)
{
  range(vec)
  counts <- hist(vec + .01, breaks = 7)$counts
  fx <- counts / (sum(counts))
  x <- c(min(vec): max(vec))
  exp <- sum(x * fx) ; exp
}
```

#media si varianta repartitiei binomiale
mean(b)
variance(b)

#media si varianta repartitiei Poisson
mean(p)
variance(p)

#media si varianta repartitiei exponentiale
mean(e)
variance(e)

#media si varianta repartitiei uniforme
mean(n)
variance(n)

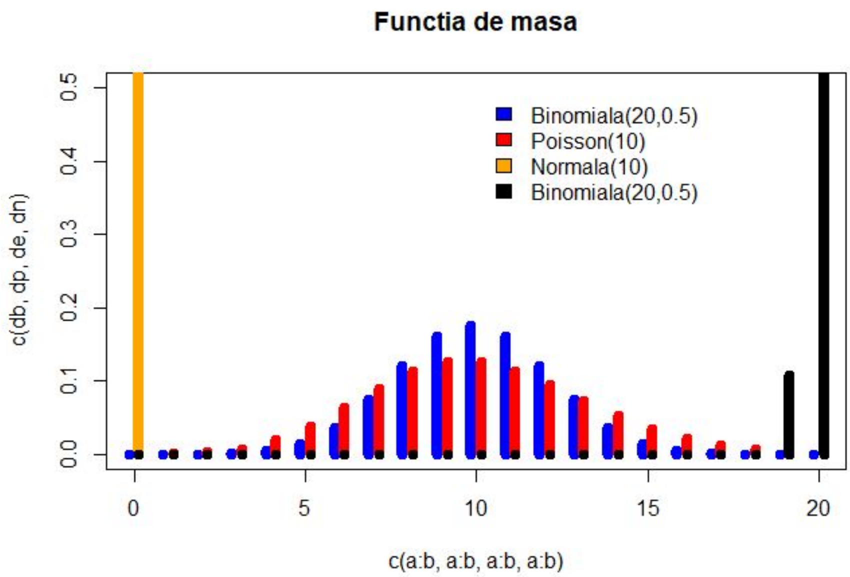
##

2. Functiile de MASA si de DENSITATE

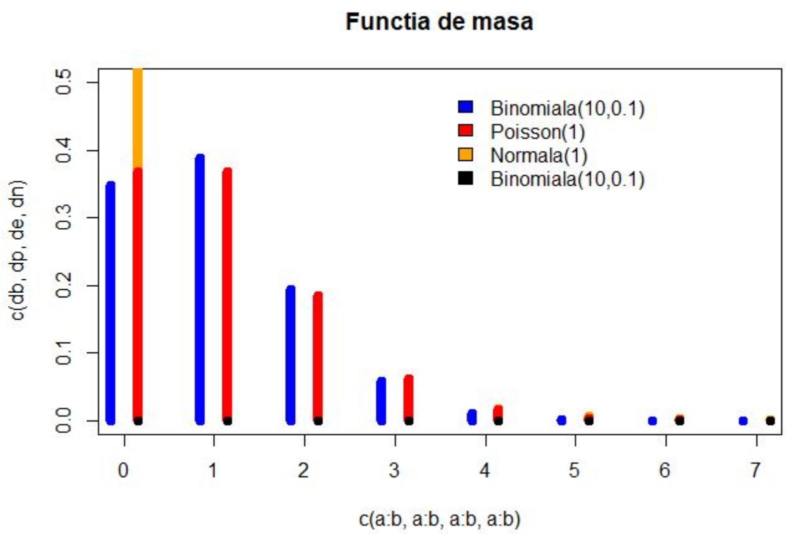
```
fct2 <- function(a,b,n,p)
{
  lambda = n*p
db <- dbinom(a:b, size = n, prob =p)
dp <- dpois(a:b, lambda)
de <- dexp(a:b,lambda)
dn <- dnorm(a:b, mean=n, sd=p)
plot(c(a:b,a:b,a:b,a:b), c(db,dp,de,dn),type="n", ylim=c(0, 0.5), main="Functia de masa")

points((a:b)+.15, de, type = "h",
       col ="orange", lwd = 8)
points((a:b)-.15, db, type = "h",
       col ="blue", lwd = 8)
points((a:b)+.15, dp, type = "h",
       col ="red", lwd = 8)

points((a:b)+.15, dn, type = "h",
       col ="black", lwd = 8)
  legend(b-b/2, 0.5, legend =
c(paste0("Binomiala(",n,",",p,")"),paste0("Poisson(",lambda,")"),
```

paste0("Normala(",lambda,")"),paste0("Binomiala(",n,",",p,")")), fill = c("blue", "red", "orange", "black"), bg="white", bty = "n")
}

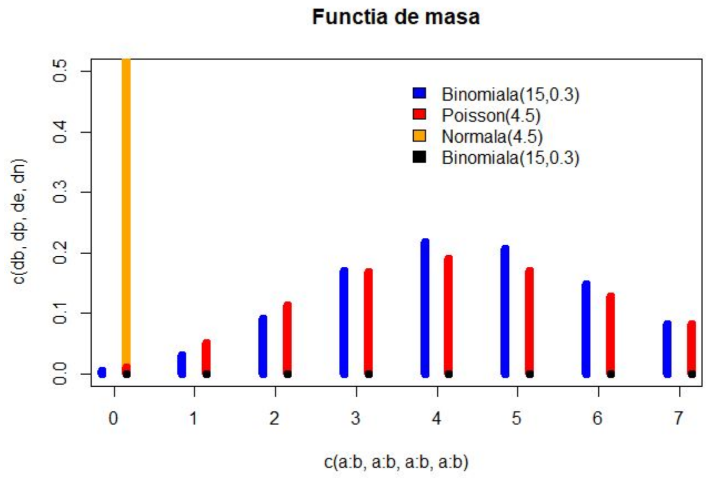
#cele 5 seturi de parametri
fct2(0,20,20,0.5)



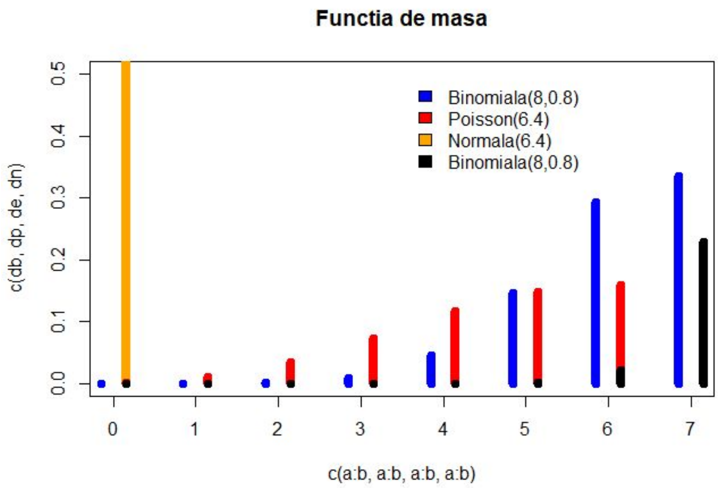
fct2(0,7,10,0.1)



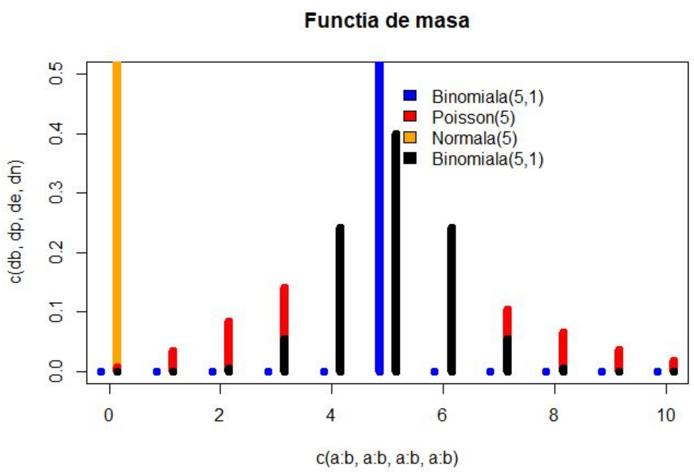
fct2(0,7,15,0.3)



fct2(0,7,8,0.8)



fct2(0,10,5,1)



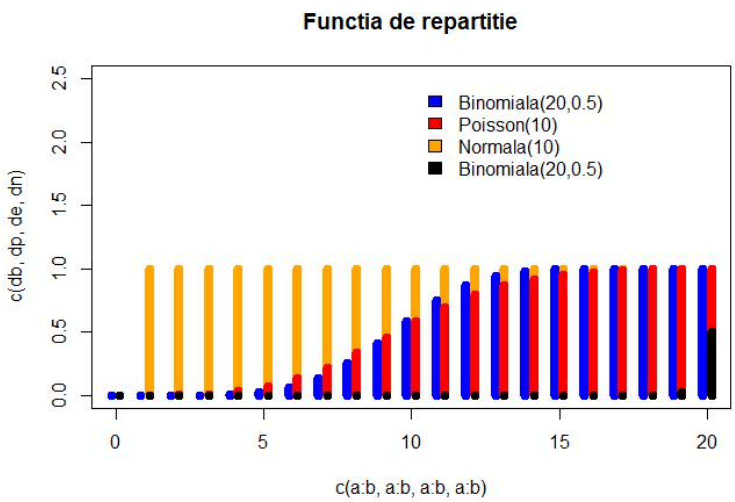
3. Functiile de repartitie

#Functiile de repartitie

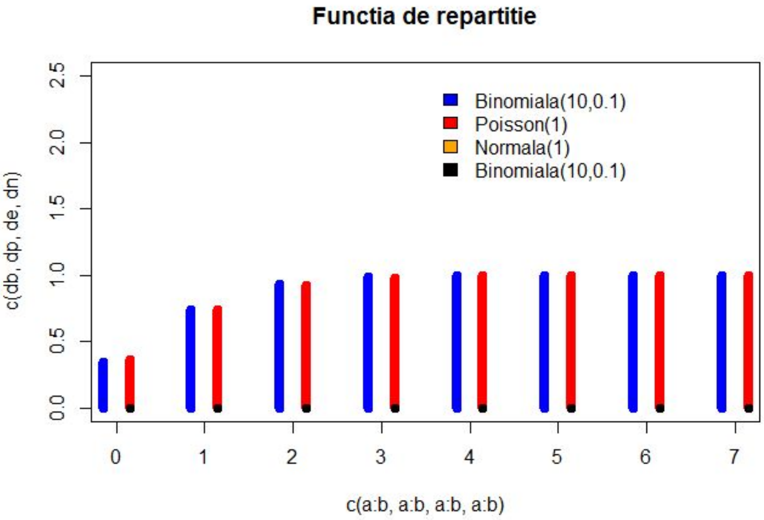
```
fct3 <- function(a,b,n,p)
{
  lambda = n*p
  db <- pbinom(a:b, size = n, prob =p)
  dp <- ppois(a:b, lambda)
  de <- pexp(a:b,lambda)
  dn <- pnorm(a:b, mean=n, sd=p)
  plot(c(a:b,a:b,a:b,a:b), c(db,dp,de,dn),type="n", ylim=c(0, 2.5), main="Functia de
repartitie")

  points((a:b)+.15, de, type = "h",
         col ="orange", lwd = 8)
  points((a:b)-.15, db, type = "h",
         col ="blue", lwd = 8)
  points((a:b)+.15, dp, type = "h",
         col ="red", lwd = 8)

  points((a:b)+.15, dn, type = "h",
         col ="black", lwd = 8)
  legend(b-b/2, 2.5, legend =
c(paste0("Binomiala(",n,",",p,")"),paste0("Poisson(",lambda,")"),
paste0("Normala(",lambda,")"),paste0("Binomiala(",n,",",p,")")), fill = c("blue", "red",
"orange", "black"), bg="white", bty = "n")
}
```

 # apel pentru cele 5 seturi ale functiei fct3

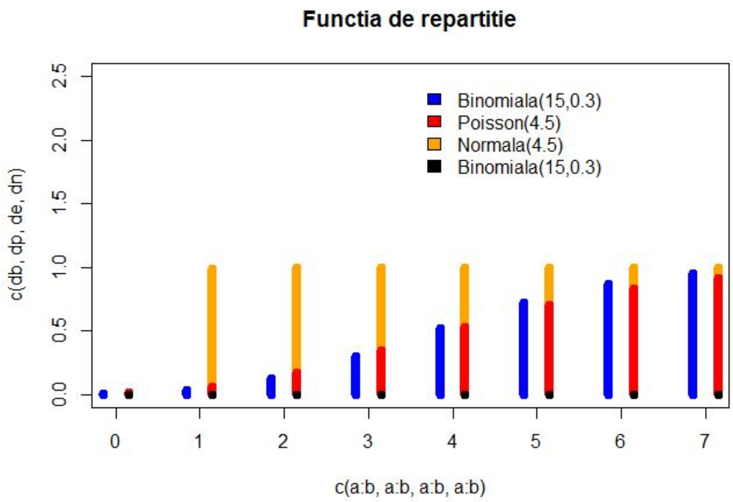
```
fct3(0,20,20,0.5)
```



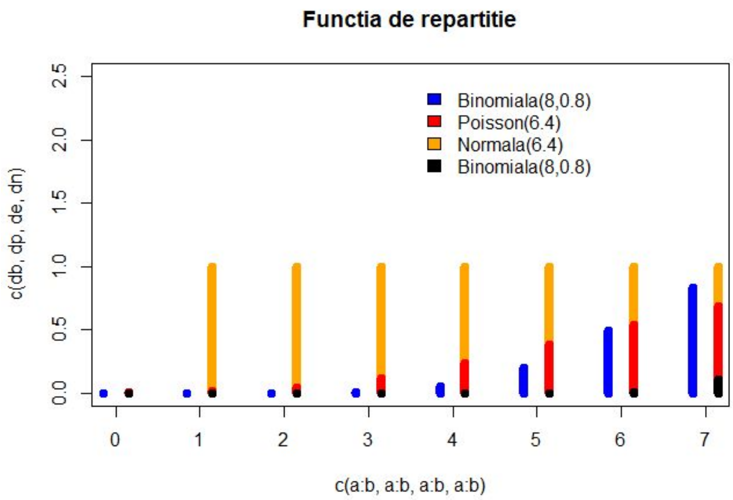
fct3(0,7,10,0.1)



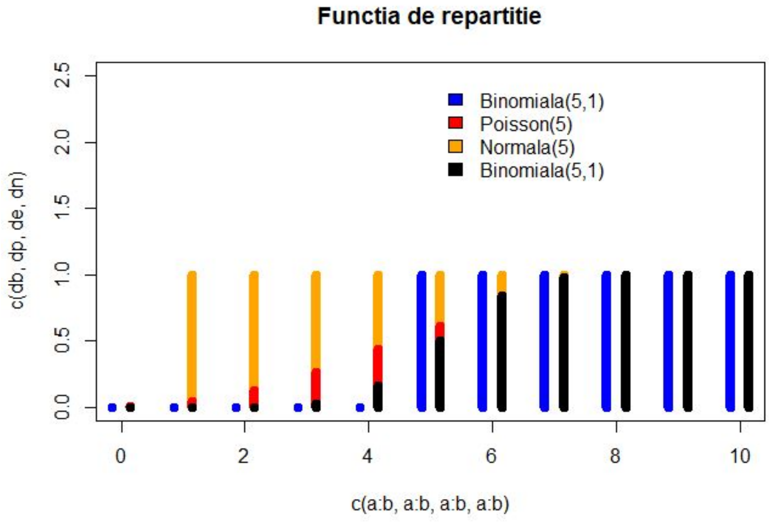
fct3(0,7,15,0.3)



fct3(0,7,8,0.8)



fct3(0,10,5,1)



###

4. Tabel - aproximari functie binomiala

```
tabel4 = function(n,p,a,b)
{
  lambda = n*p
  x = matrix(numeric((b-a+1)*6),ncol=6,dimnames = list(a:b,
c("k","Binomiala","Poisson","Normala","Normala Corectie", "Camp-Paulson"))

  for(k in a:b)
    x[k][1] = k

  x[,2] = pbinom(a:b,n,p)
  x[,3] = ppois(a:b,lambda)

  for( k in a:b)
  {
    ab <- (k-n*p)/sqrt(n*p*(1-p))
    ac <- (k+0.5-n*p)/sqrt(n*p*(1-p))
    x[k,4] = pnorm(ab,0,1)
    x[k,5] = pnorm(ac,0,1)
  }
```

```
  for(k in a:b)
  {
    a1 = 1/(9*(n-k))
    b1 = 1/(9*(k+1))
    r = ((k+1)*(1-p))/(p*(n-k))
    c = (1-b1)*r^(1/3)
    miu = 1-a1
    sigma = sqrt(a1+ (b1*r^(2/3)))
    ad <- (c-miu)/sigma
    x[k,6] = pnorm(ad)

  }

  error = max(abs(x[,6]))
  return(list(x = as.data.frame(x), error = error, param = c(n, p, lambda)))
}

#apeluri pentru n apartine {25,50, 100} si p apartine {0.05, 0.1} si k apartine [a,b]
tabel4(25,0.05,1,10)
tabel4(25,0.1,1,10)
tabel4(50,0.05,1,10)
tabel4(50,0.1,1,10)
tabel4(100,0.05,1,10)
tabel4(100,0.1,1,10)
```

Pentru apelul **tabel4(25, 0.05, 1, 10)**:

	k	**Binomiala**	**Poisson**	**Normala**	**Normala Corectie**	**Camp-Paulson**
1	1	0.2712059	0.2872975	0.1586553	0.2524925	0.2706080
2	2	0.5370941	0.5438131	0.3694413	0.5000000	0.5383546
3	3	0.7635914	0.7575761	0.6305587	0.7475075	0.7647348
4	4	0.9020064	0.8911780	0.8413447	0.9087888	0.9021393
5	5	0.9666001	0.9579790	0.9522096	0.9772499	0.9662328
6	6	0.9905236	0.9858127	0.9901847	0.9961696	0.9901932
7	7	0.9977387	0.9957533	0.9986501	0.9995709	0.9975741
8	8	0.9995425	0.9988597	0.9998771	0.9999683	0.9994841
9	9	0.9999210	0.9997226	0.9999927	0.9999985	0.9999050
10	10	0.9999883	0.9999384	0.9999997	1.0000000	0.9999848

PROBLEMA 2

a) Construcţia funcţiei **fgam** care implementează funcţiei Gamma după cum urmează:

```
fgam <- function(a)
{
 f2 <- function(x)
 {
  x^(a-1)*exp(-x)
 }
 if(a<=0)
  return(-1)

 if(a==1)
  return(1)
 if(a==1/2)
  return(sqrt(pi))

 if(is.integer(a)==TRUE && a>0)
  return(factorial(a-1))

 if(a>1)
  return((a-1)*fgam(a-1))

 return (integrate(f2,0,Inf))

}
```

b) Construcţia funcţiei **fbet** care implementează funcţiei Beta după cum urmează:

```
fbet <- function(a,b)
{
 if(a+b==1 && a>0 && b>0)
  return(pi/sin(a*pi))
 return((fgam(a)*fgam(b))/fgam(a+b))

}
```

c) X~Gamma(a,b)

Y~Beta(a,b)

Pentru a calcula probabilitățile pentru cele două variabile aleatoare vom calcula funcția de repartiție:

Funcţia de repartiţie a variabilei aleatoare X având repartiţia gamma este
(5.4.2)

$$F_X(x) = \begin{cases} \int_0^x f_X(u)\,du = \frac{\lambda^\eta}{\Gamma(\eta)} \int_0^x u^{\eta-1} e^{-\lambda u} du = \frac{\Gamma(\eta,\lambda x)}{\Gamma(\eta)}, \text{ pentru } x > 0; \\ 0, \text{ altfel.} \end{cases}$$

5.2. Calcul de probabilităţi. Funcţia de repartiţie asociată repartiţiei beta este
(5.5.3)

$$F_X(x) = \begin{cases} 0, \text{ pentru } x \le 0, \\ \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} \int_0^x u^{\alpha-1}(1-u)^{\beta-1}\,du, \text{ pentru } 0 < x < 1, \\ 1, \text{ pentru } x \ge 1. \end{cases}$$

Implementările celor două funcții de repartiție sunt implementate mai jos:
FGamma1 si **FBeta1**

```
FGamma1 <- function(x,miu,lambda)
{
 f <- function(u)
 {
  u^(miu-1)*exp((-1)*lambda*u)
 }
 z <- integrate(f,0,x)
 y <- ((lambda^miu) / fgam(miu))*z$value
 if(x>0)
  return (y)
 if(x<=0)
  return (0)
}

FBeta1 <- function(x, alfa, beta)
{
 if(x<=0)
  return (0)

 if(x>=1)
  return (1)

 f <- function(u)
 {
  u^(alfa-1)*(1-u)^(beta-1)
 }
```

```
  z <- integrate(f,0,x)
  y <- fgam(alfa+beta)/(fgam(alfa)*fgam(beta))*z$value
  return (y)
}
```

Am efectuat următoarele calcule pentru a calcula cerinţele de la 1-9:

PROBLEMA 2 c)

Fie F_g - fct. de repartiție pt. fct. de masă Gamma

F_b - fct. de repartiție pt. fct. de masă Beta

1) $\mathbb{P}(X<3) = F_g(3)$

2) $\mathbb{P}(2<X<5) = F_g(5) - F_g(2)$

3) $\mathbb{P}(3<X<4 \mid X>2) = \dfrac{\mathbb{P}(3<X<4 \cap X>2)}{\mathbb{P}(X>2)}$

$= \dfrac{\mathbb{P}(3<X<4)}{\mathbb{P}(X>2)} = \dfrac{F_g(4) - F_g(3)}{1 - F_g(2)}$

4) $\mathbb{P}(Y>2) = 1 - F_b(2)$

5) $\mathbb{P}(4<X<6) = F_g(6) - F_g(4)$

6) $\mathbb{P}(0<X<1 \mid X<7) = \dfrac{\mathbb{P}(0<X<1 \cap X<7)}{\mathbb{P}(X<7)}$

$= \dfrac{\mathbb{P}(0<X<1)}{\mathbb{P}(X<7)} = \dfrac{F_g(1) - F_g(0)}{F_g(7)}$

7) $\mathbb{P}(X+Y) < 5)$

Cum $X \perp\!\!\!\perp Y \Rightarrow$ repartiția comună $f_{XY}(x,y)$ este egală cu: $f_{XY}(x,y) = f_{GAMMA}(x) \cdot f_{BETA}(y)$

Pentru rezolvarea problemei vom calcula o integrală dublă $\int_0^5 \int_0^{\min(1,5-x)} f_{XY}(x,y)\,dx\,dy$

8) $\mathbb{P}(X - Y > 0,5)$

Calculăm integrala dublă:

$$\int_{0,5}^{\infty} \int_{\min(1,\, x-0,5)}^{1} f_{XY}(x,y)\, dx\, dy.$$

9) $\mathbb{P}(X + Y > 3 \mid X - Y > 0,5)$

$$= \frac{\mathbb{P}(X+Y > 3 \cap X - Y > 0,5)}{\mathbb{P}(X - Y) > 0,5)}$$ — o calculăm de la 8)

Calculăm integrala dublă

$$\int_{2}^{\infty} \int_{\max(3-x,\, 0)}^{\min(1,\, x-0,5)}$$

Codul pentru aceste cerințe

```
#### 1) P(X < 3)

fprobgamma1 <- function(a,b)
{
  return (FGamma1(3,a,b))
}

### apel pentru 3 seturi de valori
fprobgamma1(2,1)
fprobgamma1(3,2)
fprobgamma1(4,1)

####



#### 2) P(2 < X < 5)
fprobgamma2 <- function(a,b)
{
  return (FGamma1(5,a,b)-FGamma1(3,a,b))
}

fprobgamma2(2,1)
fprobgamma2(3,2)
fprobgamma2(3,1)
```

```r
### 3) P(3 < X < 4| X>2)
fprobgamma3 <- function(a,b)
{
  m=FGamma1(4,a,b)-FGamma1(3,a,b)
  n= 1-FGamma1(2,a,b)
  return (m/n)
}

fprobgamma3(2,1)
fprobgamma3(3,2)
fprobgamma3(3,1)


#### 4) P(Y > 2)
fprobbeta4 <- function(a,b)
{
  return (1-FBeta1(2,a,b))
}

fprobbeta4(2,1)
fprobbeta4(3,2)
fprobbeta4(3,1)


#### 5) P(4 < X < 6)
fprobgamma5 <- function(a,b)
{
  return (FGamma1(6,a,b)-FGamma1(4,a,b))
}

fprobgamma5(2,1)
fprobgamma5(3,2)
fprobgamma5(3,1)

#### 6) P(0< X < 1 | X < 7)
fprobgamma6 <- function(a,b)
{
  x <- FGamma1(1,a,b)-FGamma1(0,a,b)
  y <- FGamma1(7,a,b)
  return (x/y)
}
```

```
fprobgamma6(2,1)
fprobgamma6(3,2)
fprobgamma6(3,1)
```

7) P(X + Y < 5)

```
fprob7 <- function(a, b)
{
  f <- function(x, y)
  {
  #densitatea pentr X~ Gamma
  fG <- function(x)
  {
      fG=1/(b^a*fgam(a))*x^(a-1)*(exp(1))^(-x/b)
      return (fG)
  }

   #densitatea pentru Y ~ Beta
  fB <-  function(x)
  {

    fB= 1/(fbet(a,b))*x^(a-1)*(1-x)^(b-1)
 return (fB)

  }

  #densitatea comuna f(x,y)
  return (fG(x)* fB(y))
  }


ymax <-  function(x)
{
  # stim ca y apartine(0,1) functia Beta
  min <-  min(1,5-x)
}

I <- integral2(f,0,5,0, ymax)
return (I)

}
```

```
fprob7(2,1)
fprob7(3,2)
fprob7(3,1)
```

####8) P(X-Y>0.5)

```
fprob8 <- function(a,b)
{

  InnerFunc <- function(x, y)
  {
      fG=1/(b^a*fgam(a))*x^(a-1)*(exp(1))^(-x/b)
      fB= 1/(fbet(a,b))*y^(a-1)*(1-y)^(b-1)
       #densitatea comuna f(x,y)
       (fG(x)* fB(y))
  }
  InnerIntegral= Vectorize( function(x) { integral(InnerFunc,x-0.5,1)$value})
  return(integral(InnerIntegral,0.5,Inf))

}

fprob8(2,1)
fprob8(3,2)
fprob8(3,1)
```

####9) P(X+Y>3|X-Y>0.5)

```
fprob9 <- function(a,b)
{

  InnerFunc <- function(x, y)
  {

      fG=1/(b^a*fgam(a))*x^(a-1)*(exp(1))^(-x/b)


      fB= 1/(fbet(a,b))*y^(a-1)*(1-y)^(b-1)
```

```
    #densitatea comuna f(x,y)
    (fG(x)* fB(y))
  }

  ymin <- function(x)
  {
    min <- (1,(x-0.5))
  }

  ymax <- function(x)
  {
    max <- max(3-x,0)
  }


  InnerIntegral= Vectorize( function(x) { integral(InnerFunc,ymin,ymax)$value})
  return(integral(InnerIntegral,2,Inf))

}

fprob9(2,1)
fprob9(3,2)
fprob9(3,1)
```

d) #Tabel diferente functii implementate si functii predefinite

```
tabel<- function()
{
  x <-  matrix(nrow=9,ncol=2, dimnames = list(1:9, c("punctul c)","punctul d)")))

    x[1,1]=fprobgamma1(2,1)
    x[2,1]=fprobgamma2(2,1)
    x[3,1]=fprobgamma3(2,1)
    x[4,1]=fprobbeta4(2,1)
    x[5,1]=fprobgamma5(2,1)
    x[6,1]=fprobgamma6(2,1)
    x[7,1]=fprob7(2,1)
    x[8,1]=fprob8(2,1)
    x[9,1]=fprob9(2,1)

    x[1,2]=pgamma(3,2,1)
    x[2,2]=pgamma(5,2,1)-pgamma(3,2,1)
```

```
  x[3,2]=(pgamma(4,2,1)-pgamma(3,2,1))/(1-pgamma(3,2,1))
  x[4,2]=1-pbeta(2,2,1)
  x[5,2]=pgamma(6,2,1)-pgamma(4,2,1)
  x[6,2]=(pgamma(1,2,1)-pgamma(0,2,1))/(pgamma(7,2,1))
  x[7,2]=0
  x[8,2]=0
  x[9,2]=0

error <-  0.5
 return(list(x = as.data.frame(x), error = error))
}
tabel()
```

Observatii

Construind tabelul cu doua coloane ce păstrează în prima coloană valorile calculate cu funcţiile de repartiţie Gamma şi Beta (***FGamma1*** si **FBeta1)**, iar cea de-a doua coloană păstrează valorile calculate cu funcţiile de sistem din R, am observat că de la punctul 1)- 6) nu există diferenţe, acest fapt rezultă din implementarea funcţiilor de repartiţie identice cu cele din R: **pgamma** şi **pbeta.**

PROBLEMA 3

a)

In functia frepcomgen am generat random probabilitatile p1, p2, …… , pn pentru variabila aleatoare X, respectiv probabilitatile q1, q2, …… , qm pentru variabila aleatoare Y si le-am pozitionat pe ultima coloana, respectiv linie in matricea repartitiei comune. Am avut grija ca suma probabilitatilor pentru fiecare variabila aleatoare sa fie 1. De asemenea, am generat random, valorile tabelului repartitiei comune si am tinut cont de proprietatea ca suma valorilor de pe o linie i (1<=i<=n) sa fie p[i], respectiv suma valorilor de pe o coloana j (1<=j<=m) sa fie q[j].

Am considerat valorile necunoscute pe care urmeaza sa le determinam la punctul b, le-am pozitionat pe diagonala si le-am codificat cu -1.

```
frepcomgen<- function(n,m)
{

 x <-  matrix(nrow=n+1,ncol=m+1)

 a <- c("x1")

 for(k in 2:n )
   a <- c(a, paste("x",k,sep=""))
 a <- c(a,"pi")

 b <- c("y1")
 for(k in 2:m)
   b <- c(b, paste("y",k, sep=""))
 b <- c(b,"qi")
 colnames(x, do.NULL = FALSE)
 colnames(x) <- b
 rownames(x, do.NULL = FALSE)
 rownames(x) <- a

 x[,]=0

q <- 0
p <- 0

for(i in 1:(n-1))
{
  r <- runif(1,0.00001, 1-q)
  x[i,m+1] <- r
  q <- q+r
}

x[n,m+1] <-1-q

for(j in 1:(m-1))
{
  r <- runif(1, 0.00001, 1-p)
  x[n+1,j] <- r
  p <- p+r
}
x[n+1,m]=1-p
```

```
s <- rep(0,n)
t <- rep(0,m)
for(i in 1:(n-1))
  {
  for(j in 1:(m-1))
  {

      if(x[i,m+1]-s[i] < x[n+1,j]-t[j])
      r <- runif(1, 0.00001, x[i,m+1]-s[i])
      else
       r <- runif(1,0.00001,x[n+1,j]-t[j])
      s[i] <- s[i]+r
      t[j] <- t[j]+r
    x[i,j] <- r
  }

}
for(i in 1:n)
{
  x[i,m] <- x[i,m+1]-s[i]

}

for(j in 1:m)
  x[n,j] <- x[n+1,j]-t[j]

min1 <- min(n,m)
for(k in 1:min1)
  x[k,k] <- -1

 error <-  1
 list(x = as.data.frame(x), error = error)
 return (x)

}
frepcomgen(3,4)
```

b)

In functia fcomplrepcom determinam valorile codificate cu -1 astfel: calculam suma valorilor diferite de -1 pe fiecare coloana, salvam rezultatele intr-un vector t, iar valorile cautate vor fi egale cu diferenta dintre q[j] si t[j].

```
fcomplrepcom <- function(n,m)
{
 x <- frepcomgen(n,m)



 t <- rep(0,m)
  min1 <- min(m,n)
 for(i in 1:n)
 {
   for(j in 1:min1)
   {
    if(x[i,j]!=-1)
    t[j] <- t[j]+x[i,j]
   }

 }



 for(j in 1:min1)
   x[j,j] <- x[n+1,j]-t[j]
 list(x = as.data.frame(x), error = error)
 return (x)

}
fcomplrepcom(3,4)
```

!Nota: Am construit si functiile frepcomgen_verif si fcomplrepcom_verif unde am adaugat pentru fiecare valoare din tabel 1/n*m, iar la pi si qj am calculat sumele pe linie, respectiv coloana pentru a avea niste valori mai “accesibile” pentru verificarea calculelor de la punctele urmatoare.

```
frepcomgen_verif<- function(n,m)
{

  x <-  matrix(nrow=n+1,ncol=m+1)

  a <- c("x1")
```

```
  for(k in 2:n )
    a <- c(a, paste("x",k,sep=""))
  a <- c(a,"pi")

  b <- c("y1")
  for(k in 2:m)
    b <- c(b, paste("y",k, sep=""))
  b <- c(b,"qi")
  colnames(x, do.NULL = FALSE)
  colnames(x) <- b
  rownames(x, do.NULL = FALSE)
  rownames(x) <- a

  x[,]=1/(m*n)

  x[n+1,]=1/m
  x[,m+1]=1/n

  min1 <- min(n,m)
  for(k in 1:min1)
    x[k,k] <- -1

  error <-  1
  list(x = as.data.frame(x), error = error)
  return (x)

}
frepcomgen_verif(4,6)

fcomplrepcom_verif <- function(n,m)
{
  x <- frepcomgen_verif(n,m)



  t <- rep(0,m)
  min1 <- min(m,n)
  for(i in 1:n)
  {
    for(j in 1:min1)
    {
      if(x[i,j]!=-1)
```

```
      t[j] <- t[j]+x[i,j]
    }

  }


  for(j in 1:min1)
    x[j,j] <- x[n+1,j]-t[j]
  list(x = as.data.frame(x), error = error)
  return (x)

}
fcomplrepcom_verif(4,6)
```

!Nota: Pentru punctele aleatoate am considerat valorile pentru variabila aleatoare X -2, -1, 0, 1, …. n-3, iar pentru variabila aleatoare Y: 1,2,3, …. , m

c)

- Covarianta variabilelor aleatoare X si Y

$$\begin{aligned}\operatorname{cov}(X,Y) &= \mathrm{E}[(X-\mathrm{E}[X])(Y-\mathrm{E}[Y])] \\ &= \mathrm{E}[XY - X\,\mathrm{E}[Y] - \mathrm{E}[X]Y + \mathrm{E}[X]\,\mathrm{E}[Y]] \\ &= \mathrm{E}[XY] - \mathrm{E}[X]\,\mathrm{E}[Y] - \mathrm{E}[X]\,\mathrm{E}[Y] + \mathrm{E}[X]\,\mathrm{E}[Y] \\ &= \mathrm{E}[XY] - \mathrm{E}[X]\,\mathrm{E}[Y].\end{aligned}$$

Am calculat media variabilei X (media_x), media variabilei Y (media_y) si media repartitiei comune (media_xy), iar pentru calculul covariantei, am inlocuit cele trei medii in formula de mai sus.

```
cov <- function(n,m)
{
  xy <- fcomplrepcom_verif(n,m)
  x <- c(-2)
  for(k in 2:n)
    x <- c(x, (-3+k))

  y <- c(1)
  for(k in 2:m)
    y <- c(y, k)
```

```
  media_x <- 0
  for(i in 1:n)
  media_x <-media_x + x[i]*xy[i,m+1]

  media_y <- 0
  for(j in 1:m)
    media_y <-media_y + y[j]*xy[n+1,j]

  media_xy <- 0
  for(i in 1:n)
    for(j in 1:m)
      media_xy <- media_xy + x[i]*y[j]*xy[i,j]

  covarianta <- 12*media_xy - 12*media_x*media_y
  return (covarianta)
}

cov(2,3)
```

- Probabilitatea P(0<X<5 | Y>4)

```
prob1 <- function(n,m)
{
  xy <- fcomplrepcom_verif(n,m)
  x <- c(-2)
  for(k in 2:n)
    x <- c(x, (-3+k))

  y <- c(1)
  for(k in 2:m)
    y <- c(y, k)

  prob_numarator <- 0
  prob_numitor <- 0
  for(i in 1:n)
    for(j in 1:m)
    {
      if(x[i]>0 && x[i]<5 && y[j]>4)
        prob_numarator <- prob_numarator+xy[i,j]
      if(y[j]>4)
        prob_numitor <- prob_numitor+xy[i,j]
```

```
  }
 prob <- prob_numarator/prob_numitor
 return (prob)
}

prob1(7,5)
```

- Probabilitatea P(X>3, Y<7)

```
prob2 <- function(n,m)
{
 xy <- fcomplrepcom_verif(n,m)
 x <- c(-2)
 for(k in 2:n)
  x <- c(x, (-3+k))

 y <- c(1)
 for(k in 2:m)
  y <- c(y, k)

 prob <- 0
 for(i in 1:n)
  for(j in 1:m)
  {
   if(x[i]>3 && y[j]<7)
    prob <- prob+xy[i,j]
  }
 return (prob)
}

prob2(7,5)
```

d)

- Variabile aleatoare independente

Pentru a verifica acest fapt, ne-am folosit de urmatoarea observatie:
X si Y sunt independente daca si numai daca xy[i,j] = p[i]*q[j], oricare ar fi i,j, unde xy este tabelul repartitiei comune.

```
fverind <- function(n,m)
{
 xy <- fcomplrepcom_verif(n,m)

 verif <- 1

 for(i in 1:n)
  for(j in 1:m)
  {
   if(xy[i,j]!=xy[i,m+1]*xy[n+1,j])
    verif <- 0
  }

 if(verif==1)
  mesaj <- "Variabilele aleatoare X si Y sunt independente"
 else
  mesaj <- "Variabilele aleatoare X si Y nu sunt independente"
 return (mesaj)
}

fverind(2,3)
```

- Variabile aleatoare necorelate

2 variabile aleatoare X si Y sunt necorelate daca se indeplineste urmatoarea conditie: coeficientul de corelatie este egal cu 0. Aceasta conditie este echivalenta cu cov(X,Y)=0, calculata la punctul anterior.

```
fvernecor <- function(n,m)
{
 x <- cov(n,m)
 if(x==0)
  mesaj <- "Variabilele aleatoare X si Y sunt necorelate"
 else
  mesaj <- "Variabilele aleatoare X si Y nu sunt necorelate"
 return (mesaj)
}
fvernecor(2,3)
```