Universitatea din București
Facultatea de Matematică și Informatică

Proiect Probabilități și statistică

Studenți: Iosif Gabriel, Jilavu Alexandru, Matei Gabriel

Grupa: 243

Profesor Coordonator: Cojocea Simona

Problema 1

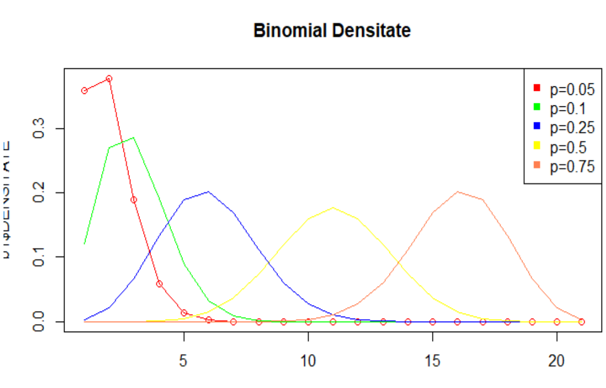
Exercitiul 1

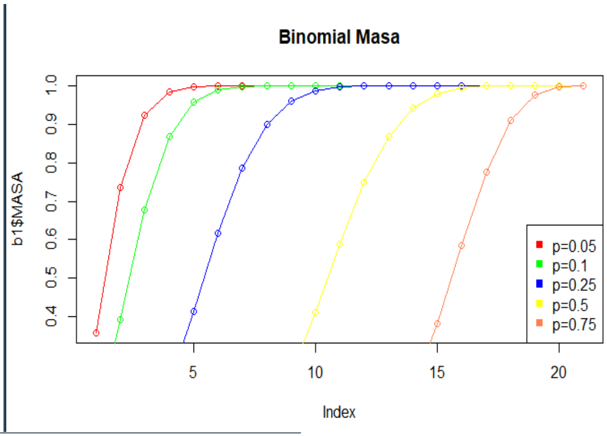
Media si varianța a unui eșantion de 1000 de realizări pentru distribuțiile: Binomiala, Poisson, Exponențială și Normală.

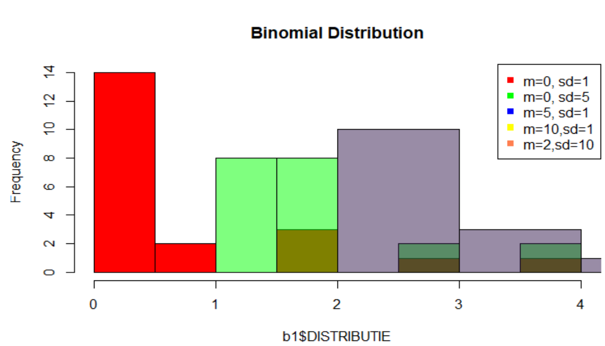
```
> p = rpois(1000,5.2)
> b = rbinom(1000, 10, 0.1)
> e = rexp(1000,10)
> n = rnorm(1000,5)
> print(var(p))
[1] 5.251151
> print(mean(p))
[1] 5.21
> print(var(b))
[1] 0.8437077
> print(mean(b))
[1] 0.944
> print(var(e))
[1] 0.01019411
> print(mean(e))
[1] 0.104552
> print(var(n))
[1] 0.9825983
> print(mean(n))
[1] 4.921296
```

Exercițiul 2 și 3

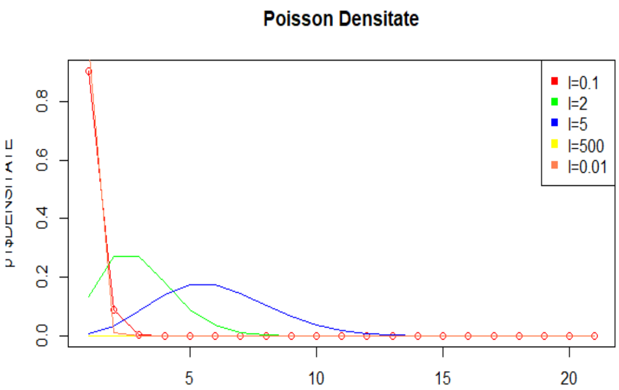
Grafice cu un set de 5 eșantioane pentru funcția de masa, densitate și repartiția distribuțiilor de mai sus.

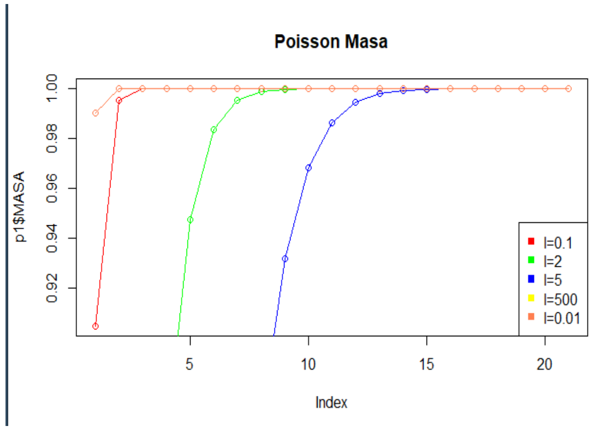


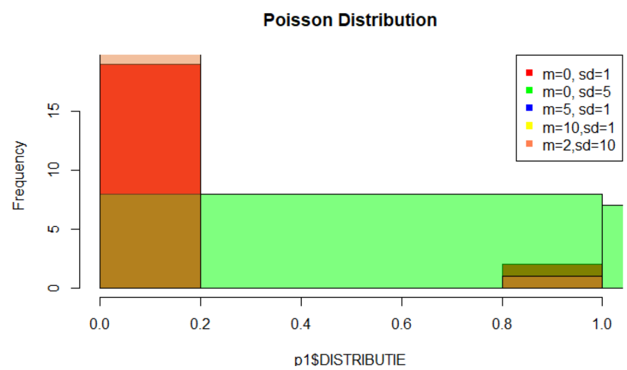




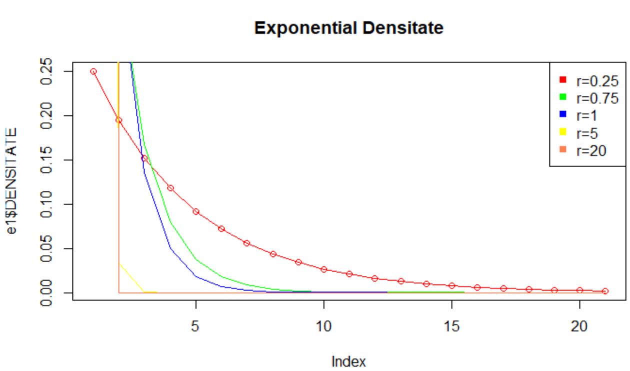
Poison:

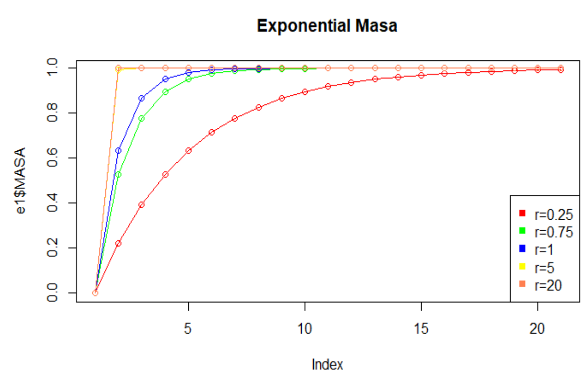


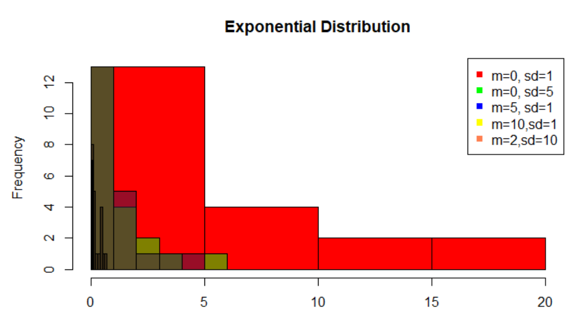




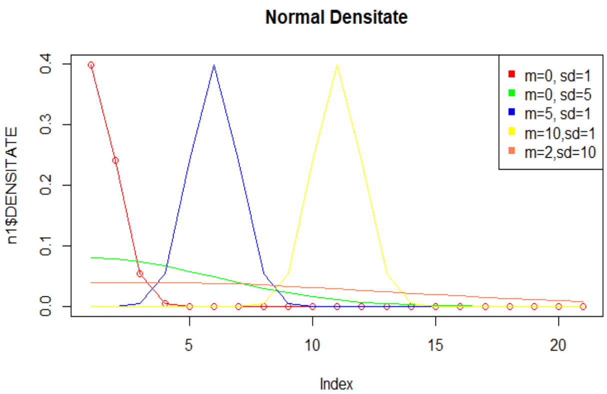
Exponentială:

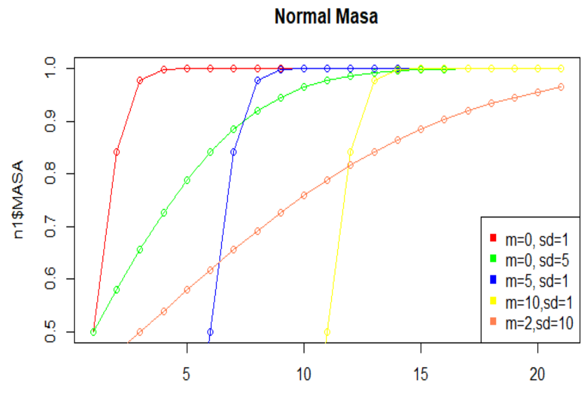


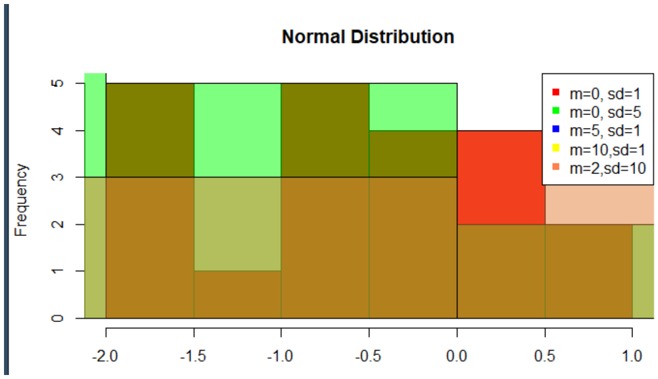




Normală:


Normal Densitate
n1$DENSITATE
Index
m=0, sd=1
m=0, sd=5
m=5, sd=1
m=10,sd=1
m=2,sd=10


Normal Masa
n1$MASA
m=0, sd=1
m=0, sd=5
m=5, sd=1
m=10,sd=1
m=2,sd=10


Normal Distribution
Frequency
m=0, sd=1
m=0, sd=5
m=5, sd=1
m=10,sd=1
m=2,sd=10

Exercițiul 4

N= 25,50,100

P=0.05, 0.1

Un tabel cu k, și aproximările Binomialei la Poison, Normala, Normală Corecție și Camp Paulson.

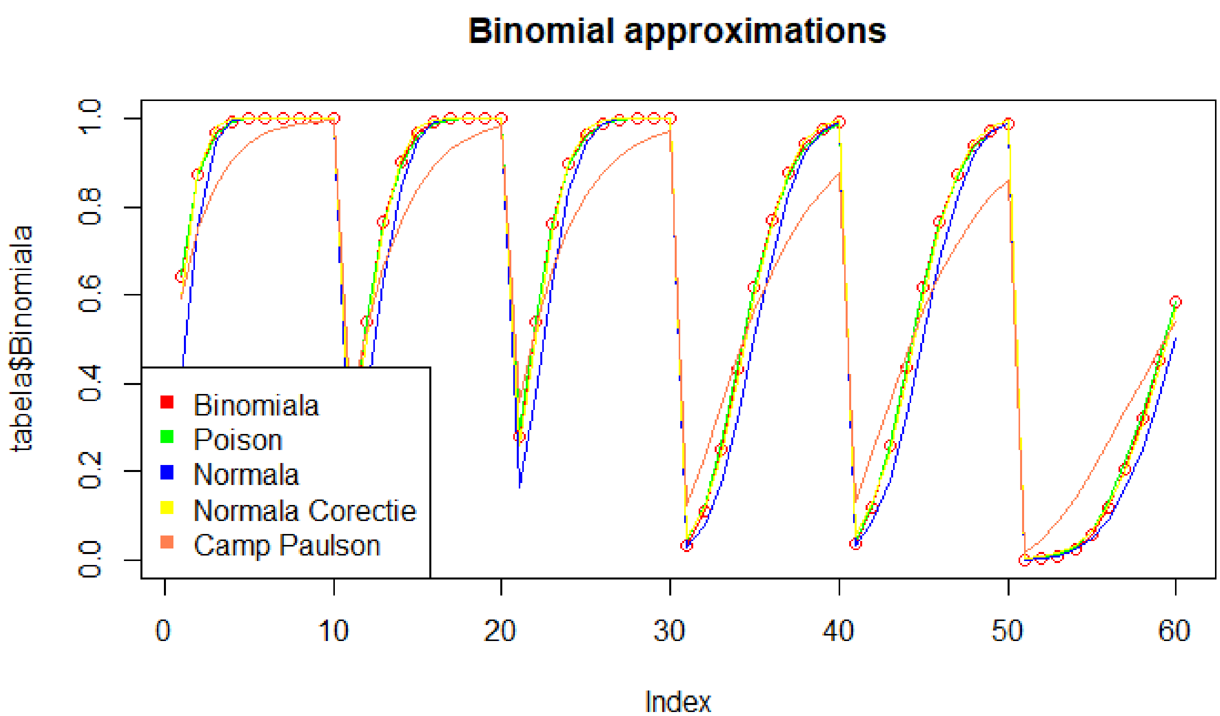
	k	Binomiala	Poison	Normala	NormalaCorectie	CampPaulson
1	1	0.6423758535	0.6446357929	0.409272904	0.590727096	0.59262058
2	2	0.8728935043	0.8684676655	0.754351438	0.874325446	0.75008969
3	3	0.9659093985	0.9617309457	0.945853172	0.980526272	0.84724971
4	4	0.9928350521	0.9908757208	0.994191554	0.998570031	0.90699872
5	5	0.9987870387	0.9981619145	0.999710468	0.999951917	0.94362640
6	6	0.9998312468	0.9996798716	0.999993464	0.999999274	0.96600199
7	7	0.9999804194	0.9999509353	0.999999934	0.999999995	0.97961594
8	8	0.9999980846	0.9999932891	1.000000000	1.000000000	0.98786008
9	9	0.9999998408	0.9999991715	1.000000000	1.000000000	0.99282518
10	10	0.9999999887	0.9999999068	1.000000000	1.000000000	0.99579656
11	1	0.2712059065	0.2872974952	0.158655254	0.252492538	0.34879090
12	2	0.5370940501	0.5438131159	0.369441340	0.500000000	0.52298433
13	3	0.7635913576	0.7575761331	0.630558660	0.747507462	0.66063151
14	4	0.9020063788	0.8911780189	0.841344746	0.908788780	0.76398668
15	5	0.9666000554	0.9579789618	0.952209648	0.977249868	0.83900181
16	6	0.9905236393	0.9858126880	0.990184671	0.996169619	0.89206722
17	7	0.9977386884	0.9957533045	0.998650102	0.999570940	0.92881841
18	8	0.9995424507	0.9988597472	0.999877134	0.999968329	0.95380085
19	9	0.9999210181	0.9997226479	0.999992657	0.999998469	0.97049307
20	10	0.9999883190	0.9999383731	0.999999713	0.999999952	0.98146280
21	1	0.2794317523	0.2872974952	0.165195024	0.258206134	0.35661204
22	2	0.5405331227	0.5438131159	0.372801394	0.500000000	0.52467687
23	3	0.7604079610	0.7575761331	0.627198606	0.741793866	0.65540920
24	4	0.8963831899	0.8911780189	0.834804976	0.902817045	0.75335968
25	5	0.9622238270	0.9579789618	0.947621255	0.974212068	0.82513051
26	6	0.9882135522	0.9858126880	0.988429534	0.995277917	0.87693828
27	7	0.9968116568	0.9957533045	0.998249762	0.999411567	0.91393108
28	8	0.9992440153	0.9988597472	0.999820739	0.999950558	0.94012409
29	9	0.9998414367	0.9997226479	0.999987663	0.999997216	0.95854385
30	10	0.9999703540	0.9999383731	0.999999432	0.999999895	0.97142244
31	1	0.0337858597	0.0404276820	0.029673219	0.049480077	0.12334881
32	2	0.1117287563	0.1246520195	0.078649604	0.119296415	0.23394200
33	3	0.2502939060	0.2650259153	0.172889293	0.239750061	0.35040859
34	4	0.4311984068	0.4404932851	0.318675944	0.406831858	0.46212605
35	5	0.6161230077	0.6159606548	0.500000000	0.593168142	0.56324017
36	6	0.7702268418	0.7621834630	0.681324056	0.760249939	0.65113980
37	7	0.8778549164	0.8666283259	0.827110707	0.880703585	0.72528802
38	8	0.9421327943	0.9319063653	0.921350396	0.950519923	0.78637678
39	9	0.9754620643	0.9681719427	0.970326781	0.983052573	0.83574536
40	10	0.9906453984	0.9863047314	0.990788937	0.995239054	0.87499962
41	1	0.0370812093	0.0404276820	0.033228710	0.054146828	0.13161208
42	2	0.1182629812	0.1246520195	0.084334309	0.125674554	0.24324379
43	3	0.2578386591	0.2650259153	0.179397679	0.245648562	0.35766722
44	4	0.4359813007	0.4404932851	0.323177598	0.409272904	0.46545072
45	5	0.6159991280	0.6159606548	0.500000000	0.590727096	0.56193827
46	6	0.7660139840	0.7621834630	0.676822402	0.754351438	0.64541063
47	7	0.8720395214	0.8666283259	0.820602321	0.874325446	0.71586989
48	8	0.9369104094	0.9319063653	0.915665691	0.945853172	0.77425144
49	9	0.9718117058	0.9681719427	0.966771290	0.980526272	0.82192820
50	10	0.9885275899	0.9863047314	0.989109269	0.994191554	0.86041030

50	10	0.9885275899	0.9863047314	0.989109269	0.994191554	0.86041030
51	1	0.0003216881	0.0004993992	0.001349898	0.002303266	0.01698554
52	2	0.0019448847	0.0027693957	0.003830381	0.006209665	0.04494246
53	3	0.0078364871	0.0103360507	0.009815329	0.015130140	0.08646480
54	4	0.0237110827	0.0292526881	0.022750132	0.033376508	0.13948212
55	5	0.0575768865	0.0670859629	0.047790352	0.066807201	0.20111062
56	6	0.1171556154	0.1301414209	0.091211220	0.121672505	0.26826865
57	7	0.2060508618	0.2202206466	0.158655254	0.202328381	0.33806030
58	8	0.3208738884	0.3328196788	0.252492538	0.308537539	0.40798684
59	9	0.4512901654	0.4579297145	0.369441340	0.433816167	0.47603787
60	10	0.5831555123	0.5830397502	0.500000000	0.566183833	0.54070328

Pentru fiecare K de la 1 la 10, se genereaza distributia binomială și aproximarea sa.

Exercițiul 5

Reprezentarea grafică a tabelului de mai sus pentru evidențierea distanței Kolomogorov.



Valorile distanței se gasesc aici:

```
+ }
[1] 0.004425839
[1] 0.2331029
[1] 0.05164876
[1] 0.1228038
[1] 0.01609159
[1] 0.1676527
[1] 0.03709405
[1] 0.1380197
[1] 0.007865743
[1] 0.1677317
[1] 0.04053312
[1] 0.1430235
[1] 0.01473201
[1] 0.116123
[1] 0.02436655
[1] 0.155756
[1] 0.007187256
[1] 0.1159991
[1] 0.0267084
[1] 0.162659
[1] 0.01416978
[1] 0.08315551
[1] 0.017474
[1] 0.151113
>
```

Distanța Kolomogorov dintre Poison și Binomială

Distanța Kolomogorov dintre Normală și Binomială

Distanța Kolomogorov dintre Normală Corecție și Binomială

Distanța Kolomogorov dintre CampPaulson și Binomială

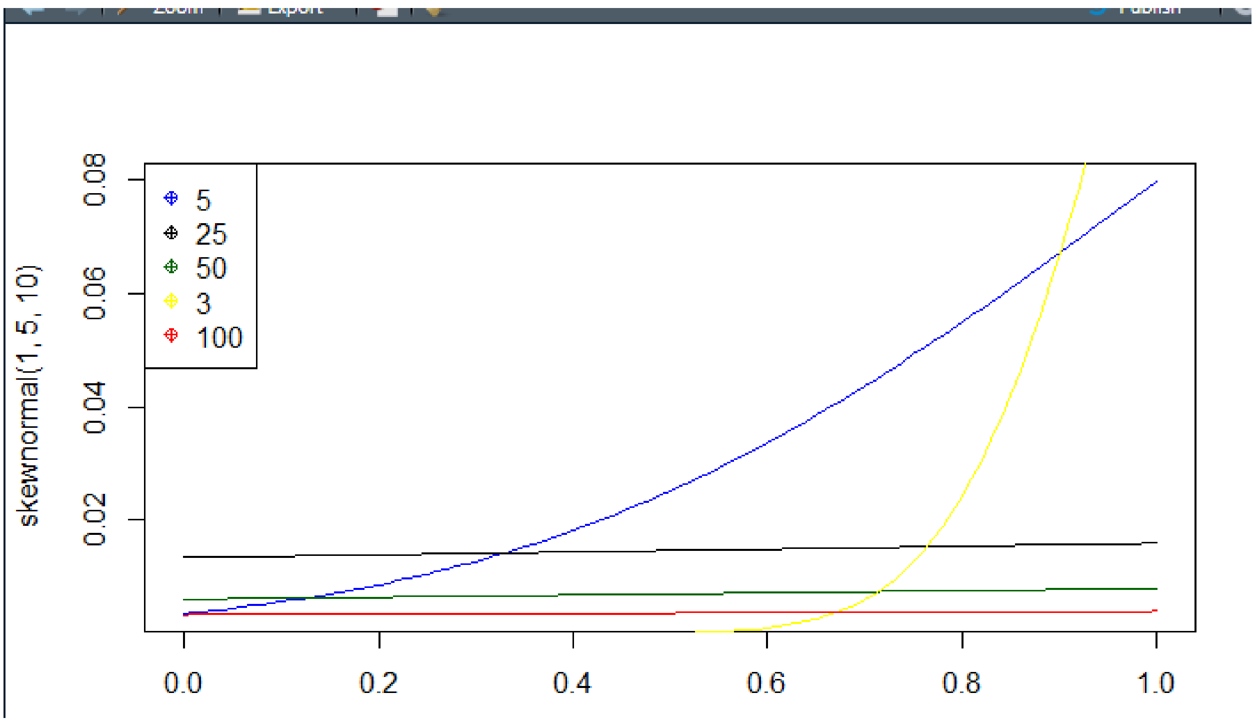
Aceeași ordine se repetă în continuare, deoarece pentru fiecare N și P facem un set de aproximări.

```r
##ex 5
for(set in 0:5){
  BP <- 0
  BN <- 0
  BNC <- 0
  BCP <- 0
  for(i in 1:10){
    if(abs(tabela[set*10+i,2] - tabela[set*10+i,3]) >= BP )
      BP = abs(tabela[set*10+i,2] - tabela[set*10+i,3])

    if(abs(tabela[set*10+i,2] - tabela[set*10+i,4]) >= BN )
      BN = abs(tabela[set*10+i,2] - tabela[set*10+i,4])

    if(abs(tabela[set*10+i,2] - tabela[set*10+i,5]) >= BNC )
      BNC = abs(tabela[set*10+i,2] - tabela[set*10+i,5])

    if(abs(tabela[set*10+i,2] - tabela[set*10+i,6]) >= BCP)
      BCP = abs(tabela[set*10+i,2] - tabela[set*10+i,6])

  }

  print(BP)
  print(BN)
  print(BNC)
  print(BCP)
}
```

Exercițiul 6

5 seturi de date suprapuse intr-un grafic pentru funcția de densitate a repartiției normale-asimetrice.

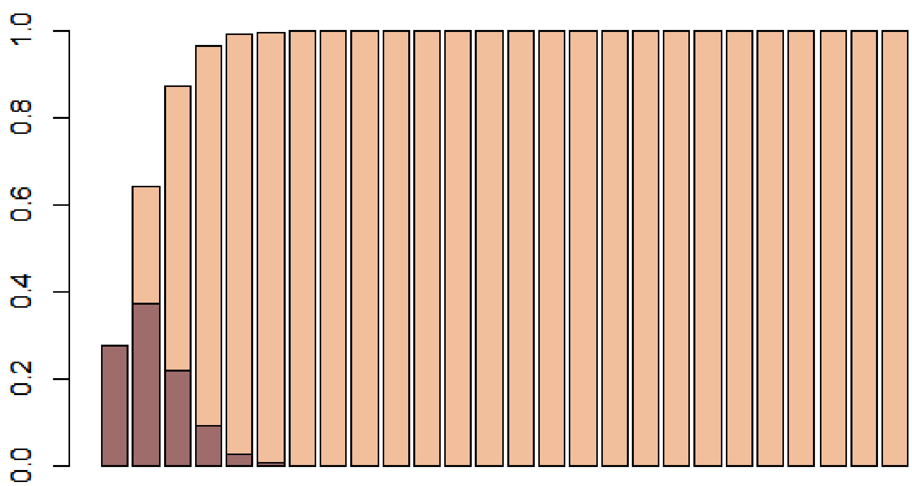
```
skewnormal <- function(miu, sigma, lambda) {
  return(Vectorize(function(x) {
    return(2 / sigma *
             dnorm((x - miu) / sigma, mean = 0, sd = 1) *
             pnorm((lambda * ((x - miu) / sigma)), mean = 0, sd = 1))
  }))
}

plot(skewnormal(1, 5, 10),col="blue")
plot(skewnormal(1, 25, 5), add=TRUE, col = "black")
plot(skewnormal(1, 50, 15), add=TRUE, col = "dark green")
plot(skewnormal(1, 3, 20), add=TRUE, col = "yellow")
plot(skewnormal(1, 100, 20), add=TRUE, col = "red")
legend("topleft", c("5","25","50","3","100"),col=c("blue","black","dark green","yellow", "red"), pch = 10)
```

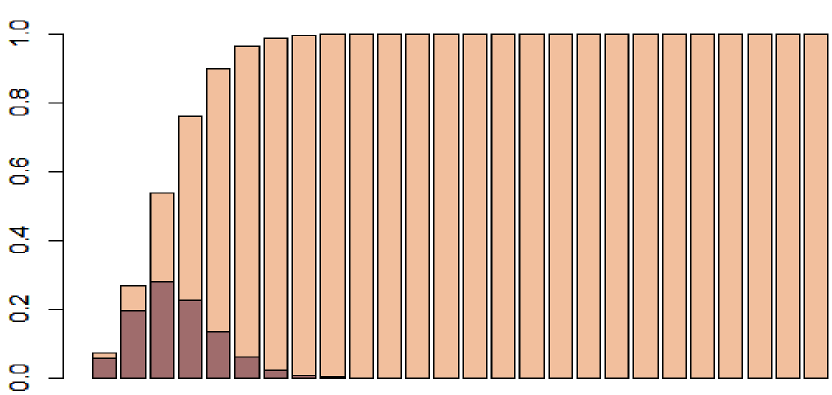


Exercițiul 7

Repartiția binomială suprapusă cu densitatea funcției skew-normal.

```r
n <- 25
myfunction <- function(n,p){
  dreapta <- (n*p*(1-p))/((1-2*p)^2)
  lambdasquared <- uniroot(function(x)  ((1-(2/pi)*((x)/(1+x)))^3)/((2/pi)*((4/pi-1)^2)*((x/(1+x))^3))-dreapta, c(0,500), extendInt = "yes")$root

  lambda <- sign(1-2*p)*sqrt(lambdasquared)
  sigma <- sqrt((n*p*(1-p))/(1-(2/pi)*((lambda^2)/(1+lambda^2))))
  miu <- n*p - sigma*sqrt((2/pi)*((lambda^2)/(1+lambda^2)))

  library(sn)
  barplot(dsn(x= 0:n, dp=c(miu,sigma,lambda)),col=rgb(0.3,0.1,0.23,0.5), add=T)

}

# pt p=0.05

barplot(pbinom(q=0:n,size=n, prob=0.05), col=rgb(0.9,0.5,0.23,0.5))
myfunction(n,0.05)

# pt p=0.1

barplot(pbinom(q=0:n, size=n, prob=0.1), col=rgb(0.9,0.5,0.23,0.5))
myfunction(n,0.1)
```





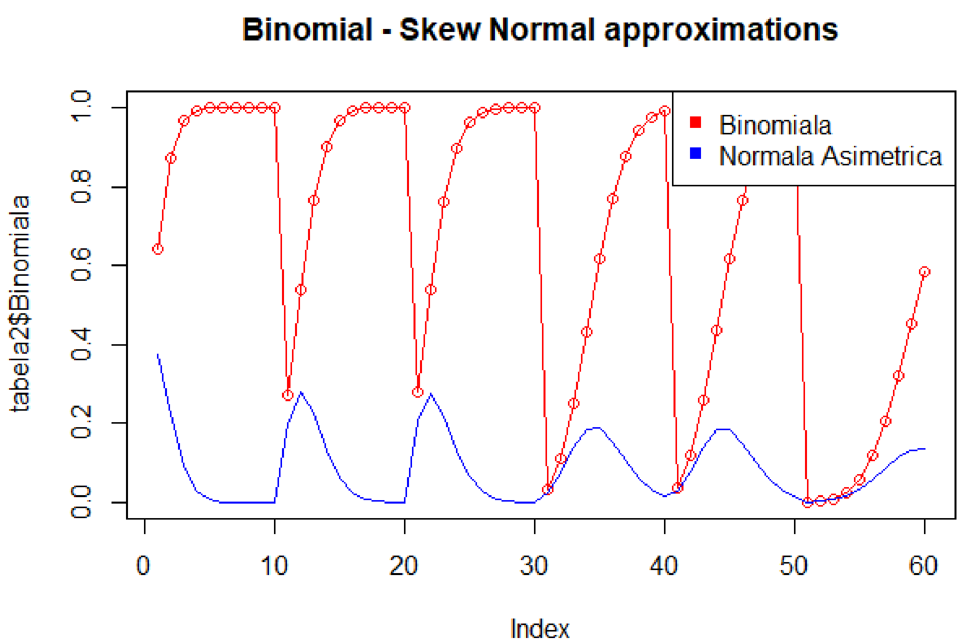
Exercițiul 8:

Tabel pentru funcția de masă a binomialei și skew-normal.

> print(tabel2)

	k	Binomiala	NormalaAsimetrica
1	1	0.6423758535	3.741620e-01
2	2	0.8728935043	2.204771e-01
3	3	0.9659093985	9.316742e-02
4	4	0.9928350521	2.828649e-02
5	5	0.9987870387	6.170329e-03
6	6	0.9998312468	9.670570e-04
7	7	0.9999804194	1.088956e-04
8	8	0.9999980846	8.810138e-06
9	9	0.9999998408	5.121175e-07
10	10	0.9999999887	2.138804e-08
11	1	0.2712059065	1.979144e-01
12	2	0.5370940501	2.800966e-01
13	3	0.7635913576	2.256307e-01
14	4	0.9020063788	1.333761e-01
15	5	0.9666000554	6.336622e-02
16	6	0.9905236393	2.448343e-02
17	7	0.9977386884	7.697486e-03
18	8	0.9995424507	1.969207e-03
19	9	0.9999210181	4.099203e-04
20	10	0.9999883190	6.943417e-05
21	1	0.2794317523	2.040528e-01
22	2	0.5405331227	2.746358e-01
23	3	0.7604079610	2.172257e-01
24	4	0.8963831899	1.311623e-01
25	5	0.9622238270	6.509916e-02
26	6	0.9882135522	2.674109e-02
27	7	0.9968116568	9.092948e-03
28	8	0.9992440153	2.559486e-03
29	9	0.9998414367	5.963804e-04
30	10	0.9999703540	1.150314e-04
31	1	0.0337858597	2.646028e-02
32	2	0.1117287563	7.286527e-02
33	3	0.2502939060	1.378476e-01
34	4	0.4311984068	1.862691e-01
35	5	0.6161230077	1.893583e-01
36	6	0.7702268418	1.536684e-01
37	7	0.8778549164	1.049970e-01
38	8	0.9421327943	6.253825e-02
39	9	0.9754620643	3.298047e-02
40	10	0.9906453984	1.547224e-02
41	1	0.0370812093	2.847368e-02
42	2	0.1182629812	7.607500e-02
43	3	0.2578386591	1.396349e-01
44	4	0.4359813007	1.838924e-01
45	5	0.6159991280	1.840216e-01
46	6	0.7660139840	1.491349e-01
47	7	0.8720395214	1.033354e-01
48	8	0.9369104094	6.324825e-02
49	9	0.9718117058	3.465022e-02
50	10	0.9885275899	1.705111e-02

50	10	0.9885275899	1.705111e-02
51	1	0.0003216881	5.957236e-04
52	2	0.0019448847	2.095295e-03
53	3	0.0078364871	6.185493e-03
54	4	0.0237110827	1.537451e-02
55	5	0.0575768865	3.230482e-02
56	6	0.1171556154	5.767519e-02
57	7	0.2060508618	8.805872e-02
58	8	0.3208738884	1.159104e-01
59	9	0.4512901654	1.328342e-01
60	10	0.5831555123	1.340715e-01



Problema 2

Punctul a)

```
fgamaux <-function(x,a) {
  return (x^(a-1)*exp(-x));
}

fgam <- function(x) {
  if(x==1)
    return (1)
  if(x==1/2)
    return (sqrt(pi))
  if(x%%1==0 && x>0)
    return (factorial(x-1))
  if(x>1)
    return ((x-1)*fgam(x-1))
  return (integrate(fgamaux, 0, Inf,a=x)$value)
}
```

Punctul b)

```r
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))
}
```

Punctul c)

Am implementat functiile *fdistgamma* si *fdistbeta* ce reprezinta densitatile repartitiilor Gamma si respective Beta

```r
fdistgamma <- function(x,a,b) {
  if(x>0 && a>0 && b>0)
    return ((x^(a-1)*exp(-x/b))/(b^a*fgam(a)))
  return (0)
}

fdistbeta <-function(x,a,b) {
  if(0<x && x<1 && a>0 && b>0)
    return ((x^(a-1)*(1-x)^(b-1))/(fbet(a,b)))
  return (0)
}
```

<table>
<tr><th>#</th><th>Cod</th><th>Exemplu</th></tr>
<tr><td>1)</td><td><pre>fprobgamma1 <- function(A,B) {
  return (integrate(fdistgamma,0,3,a=A,b=B)$value)
}</pre></td><td><pre>> p1 <- fprobgamma1(1,2)
> print(p1)
[1] 0.7768698</pre></td></tr>
<tr><td>2)</td><td><pre>fprobgamma2 <- function(A,B) {
  F1 <- integrate(fdistgamma,0,5,a=A,b=B)
  F2 <- integrate(fdistgamma,0,2,a=A,b=B)
  return (F1$value-F2$value)
}</pre></td><td><pre>> p2 <- fprobgamma2(2,3)
> print(p2)
[1] 0.3520269</pre></td></tr>
<tr><td>3)</td><td><pre>fprobgamma3 <- function(A,B) {
  F1 <- integrate(fdistgamma, 0, 4,a=A,b=B)$value
  F2 <- integrate(fdistgamma, 0, 3,a=A,b=B)$value
  F3 <- integrate(fdistgamma, 0, 2,a=A,b=B)$value
  return (F1-F2)/(1-F3)
}</pre></td><td><pre>> p3 <- fprobgamma3(3,4)
> print(p3)
[1] 0.03979596</pre></td></tr>
<tr><td>4)</td><td><pre>fprobbeta4 <-function(A,B) {
  F1 <- integrate(Vectorize(fdistbeta),0,2,a=A,b=B)$value
  return (1 - F1)
}</pre></td><td><pre>> p4 <- fprobbeta4(5,10)
> print(p4)
[1] 1.110223e-16</pre></td></tr>
<tr><td>5)</td><td><pre>fprobgamma5 <- function(A,B) {
  F1 <- integrate(fdistgamma, 0, 6,a=A,b=B)$value
  F2 <- integrate(fdistgamma, 0, 4,a=A,b=B)$value
  return (F1-F2)
}</pre></td><td><pre>> p5 <- fprobgamma5(1,3)
> print(p5)
[1] 0.1282619</pre></td></tr>
<tr><td>6)</td><td><pre>fprobgamma6 <-function(A,B) {
  F1 <- integrate(Vectorize(fdistgamma), 0, 1,a=A,b=B)$value
  F2 <- integrate(Vectorize(fdistgamma), 0, 0,a=A,b=B)$value
  F3 <- integrate(Vectorize(fdistgamma), 0, 7,a=A,b=B)$value
  return ((F1-F2)/F3)
}</pre></td><td><pre>> p6 <- fprobgamma6(2,5)
> print(p6)
[1] 0.04293116</pre></td></tr>
</table>

Punctul d)

```r
sdistgamma <- function(x,a,b) {
  if(x>0 && a>0 && b>0)
    return ((x^(a-1)*exp(-x/b))/(b^a*gamma(a)))
  return (0)
}

sdistbeta <-function(x,a,b) {
  if(0<x && x<1 && a>0 && b>0)
    return ((x^(a-1)*(1-x)^(b-1))/(beta(a,b)))
  return (0)
}
```

#	Cod	Exemplu

1)	`sprobgamma1 <- function(A,B) {`
	`  return (integrate(sdistgamma,0,3,a=A,b=B)$value)`
	`}`
		`> s1 <- sprobgamma1(1,2)`
		`> print(s1)`
		`[1] 0.7768698`

2)	`sprobgamma2 <- function(A,B) {`
	`  F1 <- integrate(sdistgamma,0,5,a=A,b=B)`
	`  F2 <- integrate(sdistgamma,0,2,a=A,b=B)`
	`  return (F1$value-F2$value)`
	`}`
		`> s2 <- sprobgamma2(2,3)`
		`> print(s2)`
		`[1] 0.3520269`

3)	`sprobgamma3 <- function(A,B) {`
	`  F1 <- integrate(sdistgamma, 0, 4,a=A,b=B)$value`
	`  F2 <- integrate(sdistgamma, 0, 3,a=A,b=B)$value`
	`  F3 <- integrate(sdistgamma, 0, 2,a=A,b=B)$value`
	`  return (F1-F2)/(1-F3)`
	`}`
		`> s3 <- sprobgamma3(3,3)`
		`> print(s3)`
		`[1] 0.07033005`

4)	`sprobbeta4 <-function(A,B) {`
	`  F1 <- integrate(Vectorize(sdistbeta),0,2,a=A,b=B)$value`
	`  return (1 - F1)`
	`}`
		`> s4 <- sprobbeta4(5,10)`
		`> print(s4)`
		`[1] 1.110223e-16`

5)	`sprobgamma5 <- function(A,B) {`
	`  F1 <- integrate(Vectorize(sdistgamma), 0, 6,a=A,b=B)$value`
	`  F2 <- integrate(Vectorize(sdistgamma), 0, 4,a=A,b=B)$value`
	`  return (F1-F2)`
	`}`
		`> s5 <- sprobgamma5(1,3)`
		`> print(s5)`
		`[1] 0.1282619`

6)	`sprobgamma6 <-function(A,B) {`
	`  F1 <- integrate(Vectorize(sdistgamma), 0, 1,a=A,b=B)$value`
	`  F2 <- integrate(Vectorize(sdistgamma), 0, 0,a=A,b=B)$value`
	`  F3 <- integrate(Vectorize(sdistgamma), 0, 7,a=A,b=B)$value`
	`  return ((F1 - F2) / F3)`
	`}`
		`> s6 <- sprobgamma6(2,5)`
		`> print(s6)`
		`[1] 0.04293116`

Tabelul in care sunt centralizate rezltatele obtinute:

```
        Punctul_C    Punctul_D
[1,] 7.768698e-01 7.768698e-01
[2,] 3.520269e-01 3.520269e-01
[3,] 3.979596e-02 7.033005e-02
[4,] 1.110223e-16 1.110223e-16
[5,] 1.282619e-01 1.282619e-01
[6,] 4.293116e-02 4.293116e-02
```

Problema 3

La aceasta problema, am tratat cazul particular in care m = 2 si n = 3. Pentru acest caz, am construit o repartitia comuna a celor doua v.a. discrete, pornind de la valori generate aleator pentru x1, x2 si y1, y2, y3 si de la probabilitati generate aleator (dar care respecta regulile de formare ale tabelului) pe pozitii prestabilite. Apoi, pornind de la aceasta distributie aleatoare de la subpunctul a), am completat repartitia cu valorile potrivite, gasite la b).

Punctul a)

```
frepcomgen = function(m, n) {
  m=2
  n=3
  
  # Generam m valori pentru x in intervalul [0, m * 10] si
  # n valori pentru y in [0, n * 10]
  xValues <- sample.int(m * 10, m);
  yValues <- sample.int(n * 10, n);

  # Adaugam in acesti vectori Qi si Pi pentru capetele de tabel
  xValues <- c(xValues, 'Qi')
  yValues <- c(yValues, 'Pi')

  # Construim tabelul de repartitie, reprezentat ca o matrice, unde valoarea -1
  # semnifica faptul ca probabilitatea pentru (X = xi si Y=yi) nu este cunoscuta
  repart <- matrix(-1, nrow=m+1, ncol=n+1)
  rownames(repart) <- xValues
  colnames(repart) <- yValues
  repart[m+1, n+1] = 1


  # Completam repartitia cu valori aleatoare, dar care vor respecta totusi conditia ca
  # suma pe linii/coloane sa fie mai mica ca 1 (sau egala)
  repart[1, 4] <- runif(1, 0, 1)
  repart[1, 1] <- runif(1, 0, repart[1, 4])

  repart[3, 1] <- runif(1, repart[1, 1], 1)

  repart[2, 2] <- runif(1, 0, 1 - repart[3, 1])
  while(repart[3, 2] - repart[2, 2] + repart[1, 1] > repart[1, 4] ||
          repart[3, 1] - repart[1, 1] + repart[2, 2] > 1 - repart[1, 4])
    repart[2, 2] <- runif(1, 0, 1 - repart[3, 1])

  repart[3, 2] <- runif(1, min(repart[2, 2], repart[3, 1]), 1 - max(repart[2, 2], repart[3, 1]))
  while(repart[3, 2] + repart[3, 1] > 1 || repart[2, 2] > repart[3, 2])
    repart[3, 2] <- runif(1, min(repart[2, 2], repart[3, 1]), 1 - max(repart[2, 2], repart[3, 1]))

  return(repart)
}
```

Un exemplu de repartitie comuna generata ar fi:

```
          28          5  2         Pi
11  0.1440337 -1.0000000 -1  0.3551308
13 -1.0000000  0.4984637 -1 -1.0000000
Qi  0.1503825  0.5008721 -1  1.0000000
```

Unde –1, fie el pe pozitia (I, j), semnifica ca P(X = xi, Y=yj) este necunoscuta, iar valorile mai mari ca 0 sunt probabilitatile cunoscute.

Punctul b)

La punctul b) primim ca input repartitia incompleta generata la a) si completam tabelul.

```
fcomplrepcom <- function(repart) {
  repart[2, 4] = 1 - repart[1, 4]
  repart[1, 2] = repart[3, 2] - repart[2, 2]
  repart[1, 3] = repart[1, 4] - repart[1, 1] - repart[1, 2]
  repart[2, 1] = repart[3, 1] - repart[1, 1]
  repart[2, 3] = repart[2, 4] - repart[2, 2] - repart[2, 1]
  repart[3, 3] = 1 - repart[3, 1] - repart[3, 2]
```

Pe exemplul considerat mai sus, tabelul arata dupa cum urmeaza:

```
           X28       X5        X2        Pi
11 0.144033707 0.0024084 0.2086887 0.3551308
13 0.006348784 0.4984637 0.1400567 0.6448692
Qi 0.150382492 0.5008721 0.3487454 1.0000000
```

Anexa -- link github -> https://github.com/IosifGabriel/Probability-Project

Cod problema 1

#Problema 1

```
#Ex 1 media si varianta 1000 v.a. independente

p = rpois(1000,5.2)
b = rbinom(1000, 10, 0.1)
e = rexp(1000,10)
n = rnorm(1000,5)
print(var(p))
print(mean(p))
print(var(b))
print(mean(b))
print(var(e))
print(mean(e))
print(var(n))
print(mean(n))

#Ex 2 si 3 graficele functiilor de masa, densitate si functiile de repartitie (distributie)

##graficele poisson
p1 = data.frame(DENSITATE=dpois(0:20, 0.1), MASA=ppois(0:20, 0.1), DISTRIBUTIE= rpois(0:20,
0.1))
p2 = data.frame(DENSITATE=dpois(0:20, 2), MASA=ppois(0:20, 2), DISTRIBUTIE= rpois(0:20, 2))
p3 = data.frame(DENSITATE=dpois(0:20, 5), MASA=ppois(0:20, 5), DISTRIBUTIE= rpois(0:20, 5))
p4 = data.frame(DENSITATE=dpois(0:20, 500), MASA=ppois(0:20, 500), DISTRIBUTIE= rpois(0:20,
500))
p5 = data.frame(DENSITATE=dpois(0:20, 0.01), MASA=ppois(0:20, 0.01), DISTRIBUTIE= rpois(0:20,
0.01))

plot(p1$DENSITATE, type="o",col="red", main="Poisson Densitate")
lines(p2$DENSITATE,Type="o",col="green")
lines(p3$DENSITATE,Type="o",col="blue")
lines(p4$DENSITATE,Type="o",col="yellow")
lines(p5$DENSITATE,Type="o",col="coral")
legend("topright",c("l=0.1","l=2","l=5","l=500", "l=0.01"),
col=c("red","green","blue","yellow","coral"),pch=15)

plot(p1$MASA,type="o",col="red", main="Poisson Masa")
lines(p2$MASA,type="o",col="green")
lines(p3$MASA,type="o",col="blue")
lines(p4$MASA,type="o",col="yellow")
lines(p5$MASA,type="o",col="coral")
legend("bottomright",c("l=0.1","l=2","l=5","l=500", "l=0.01"),
col=c("red","green","blue","yellow","coral"),pch=15)

hist(p1$DISTRIBUTIE, col="red", main="Poisson Distribution")
hist(p2$DISTRIBUTIE, col=rgb(0,1,0,0.5), add=T)
```

```
hist(p3$DISTRIBUTIE, col=rgb(0.2,0.1,0.3,0.5), add=T)
hist(p4$DISTRIBUTIE, col=rgb(0.1,0.05,0.04,0.5), add=T)
hist(p5$DISTRIBUTIE, col=rgb(0.9,0.5,0.23,0.5), add=T)
legend("topright",c("m=0, sd=1","m=0, sd=5","m=5, sd=1","m=10,sd=1", "m=2,sd=10"),
col=c("red","green","blue","yellow","coral"),pch=15)


##graficele binomial
b1 = data.frame(DENSITATE=dbinom(0:20, 20, 0.05), MASA=pbinom(0:20, 20, 0.05),
DISTRIBUTIE=rbinom(0:20, 20, 0.05))
b2 = data.frame(DENSITATE=dbinom(0:20, 20, 0.1), MASA=pbinom(0:20, 20, 0.1),
DISTRIBUTIE=rbinom(0:20, 20, 0.1))
b3 = data.frame(DENSITATE=dbinom(0:20, 20, 0.25), MASA=pbinom(0:20, 20, 0.25),
DISTRIBUTIE=rbinom(0:20, 20, 0.25))
b4 = data.frame(DENSITATE=dbinom(0:20, 20, 0.5), MASA=pbinom(0:20, 20, 0.5),
DISTRIBUTIE=rbinom(0:20, 20, 0.5))
b5 = data.frame(DENSITATE=dbinom(0:20, 20, 0.75), MASA=pbinom(0:20, 20, 0.75),
DISTRIBUTIE=rbinom(0:20, 20, 0.75))

plot(b1$DENSITATE, type="o",col="red", main="Binomial Densitate")
lines(b2$DENSITATE,Type="o",col="green")
lines(b3$DENSITATE,Type="o",col="blue")
lines(b4$DENSITATE,Type="o",col="yellow")
lines(b5$DENSITATE,Type="o",col="coral")
legend("topright",c("p=0.05","p=0.1","p=0.25","p=0.5", "p=0.75"),
col=c("red","green","blue","yellow","coral"),pch=15)

plot(b1$MASA,type="o",col="red", main="Binomial Masa")
lines(b2$MASA,type="o",col="green")
lines(b3$MASA,type="o",col="blue")
lines(b4$MASA,type="o",col="yellow")
lines(b5$MASA,type="o",col="coral")
legend("bottomright",c("p=0.05","p=0.1","p=0.25","p=0.5", "p=0.75"),
col=c("red","green","blue","yellow","coral"),pch=15)

hist(b1$DISTRIBUTIE, col="red", main="Binomial Distribution")
hist(b2$DISTRIBUTIE, col=rgb(0,1,0,0.5), add=T)
hist(b3$DISTRIBUTIE, col=rgb(0.2,0.1,0.3,0.5), add=T)
hist(b4$DISTRIBUTIE, col=rgb(0.1,0.05,0.04,0.5), add=T)
hist(b5$DISTRIBUTIE, col=rgb(0.9,0.5,0.23,0.5), add=T)
legend("topright",c("m=0, sd=1","m=0, sd=5","m=5, sd=1","m=10,sd=1", "m=2,sd=10"),
col=c("red","green","blue","yellow","coral"),pch=15)


##graficele exponential
e1 = data.frame(DENSITATE=dexp(0:20, 0.25), MASA=pexp(0:20, 0.25), DISTRIBUTIE=rexp(0:20,
0.25))
e2 = data.frame(DENSITATE=dexp(0:20, 0.75), MASA=pexp(0:20, 0.75), DISTRIBUTIE=rexp(0:20,
0.75))
e3 = data.frame(DENSITATE=dexp(0:20, 1), MASA=pexp(0:20, 1), DISTRIBUTIE=rexp(0:20, 1))
e4 = data.frame(DENSITATE=dexp(0:20, 5), MASA=pexp(0:20, 5), DISTRIBUTIE=rexp(0:20, 5))
e5 = data.frame(DENSITATE=dexp(0:20, 20), MASA=pexp(0:20, 20), DISTRIBUTIE=rexp(0:20, 20))

plot(e1$DENSITATE, type="o",col="red", main="Exponential Densitate")
lines(e2$DENSITATE,Type="o",col="green")
lines(e3$DENSITATE,Type="o",col="blue")
lines(e4$DENSITATE,Type="o",col="yellow")
```

```
lines(e5$DENSITATE,Type="o",col="coral")
legend("topright",c("r=0.25","r=0.75","r=1","r=5", "r=20"),
col=c("red","green","blue","yellow","coral"),pch=15)

plot(e1$MASA,type="o",col="red", main="Exponential Masa")
lines(e2$MASA,type="o",col="green")
lines(e3$MASA,type="o",col="blue")
lines(e4$MASA,type="o",col="yellow")
lines(e5$MASA,type="o",col="coral")
legend("bottomright",c("r=0.25","r=0.75","r=1","r=5", "r=20"),
col=c("red","green","blue","yellow","coral"),pch=15)

hist(e1$DISTRIBUTIE, col="red", main="Exponential Distribution")
hist(e2$DISTRIBUTIE, col=rgb(0,1,0,0.5), add=T)
hist(e3$DISTRIBUTIE, col=rgb(0.2,0.1,0.3,0.5), add=T)
hist(e4$DISTRIBUTIE, col=rgb(0.1,0.05,0.04,0.5), add=T)
hist(e5$DISTRIBUTIE, col=rgb(0.9,0.5,0.23,0.5), add=T)
legend("topright",c("m=0, sd=1","m=0, sd=5","m=5, sd=1","m=10,sd=1", "m=2,sd=10"),
col=c("red","green","blue","yellow","coral"),pch=15)


##graficele normal
n1 = data.frame(DENSITATE=dnorm(0:20, 0, 1), MASA=pnorm(0:20, 0, 1), DISTRIBUTIE=rnorm(0:20,
0, 1))
n2 = data.frame(DENSITATE=dnorm(0:20, 0, 5), MASA=pnorm(0:20, 0, 5), DISTRIBUTIE=rnorm(0:20,
0, 5))
n3 = data.frame(DENSITATE=dnorm(0:20, 5, 1), MASA=pnorm(0:20, 5, 1), DISTRIBUTIE=rnorm(0:20,
5, 1))
n4 = data.frame(DENSITATE=dnorm(0:20, 10, 1), MASA=pnorm(0:20, 10, 1), DISTRIBUTIE=rnorm(0:20,
10, 1))
n5 = data.frame(DENSITATE=dnorm(0:20, 2, 10), MASA=pnorm(0:20, 2, 10), DISTRIBUTIE=rnorm(0:20,
2, 10))

plot(n1$DENSITATE, type="o",col="red", main="Normal Densitate")
lines(n2$DENSITATE,Type="o",col="green")
lines(n3$DENSITATE,Type="o",col="blue")
lines(n4$DENSITATE,Type="o",col="yellow")
lines(n5$DENSITATE,Type="o",col="coral")
legend("topright",c("m=0, sd=1","m=0, sd=5","m=5, sd=1","m=10,sd=1", "m=2,sd=10"),
col=c("red","green","blue","yellow","coral"),pch=15)

plot(n1$MASA,type="o",col="red", main="Normal Masa")
lines(n2$MASA,type="o",col="green")
lines(n3$MASA,type="o",col="blue")
lines(n4$MASA,type="o",col="yellow")
lines(n5$MASA,type="o",col="coral")
legend("bottomright",c("m=0, sd=1","m=0, sd=5","m=5, sd=1","m=10,sd=1", "m=2,sd=10"),
col=c("red","green","blue","yellow","coral"),pch=15)

hist(n1$DISTRIBUTIE, col="red", main="Normal Distribution")
hist(n2$DISTRIBUTIE, col=rgb(0,1,0,0.5), add=T)
hist(n3$DISTRIBUTIE, col=rgb(0.2,0.1,0.3,0.5), add=T)
hist(n4$DISTRIBUTIE, col=rgb(0.1,0.05,0.04,0.5), add=T)
hist(n5$DISTRIBUTIE, col=rgb(0.9,0.5,0.23,0.5), add=T)
legend("topright",c("m=0, sd=1","m=0, sd=5","m=5, sd=1","m=10,sd=1", "m=2,sd=10"),
col=c("red","green","blue","yellow","coral"),pch=15)
```

```
## ex 4


## 3*2*10 elemente
i <- 1
myk <- numeric(60)
Mass <- numeric(60)
Poison <- numeric(60)
Norm <- numeric(60)
NormCorect <- numeric(60)
CampP <- numeric(60)

for(n in c(25,50,100)){
  for(p in c(0.05, 0.1)){
    for(k in 1:10){
      a <- 1/(9*(n-k))
      b <- 1/(9*(k+1))
      r <- ((k+1)*(1-p))/(p*(n-k))
      c <- (1-b)*r^(1/3)
      miu <- 1-a
      sigmapatrat <- a + (b*r)^(2/3)

      myk[i] <- k
      Mass[i] <- pbinom(k, n, p)  ## asta e rezultatul pt care caut aproximari
      DISTRIBUTIE <- rbinom(k,n,p)
      mylambda <- n*p
      Poison[i] <- ppois(k, lambda = mylambda)
      Norm[i] <-  pnorm((k-n*p)/sqrt(n*p*(1-p)))
      NormCorect[i] <-  pnorm((k + 0.5 - n*p)/sqrt(n*p*(1-p)))
      CampP[i] <- pnorm((c-miu)/sqrt(sigmapatrat))
      i <- i+1

    }
  }
}

tabela <- data.frame(k = myk, Binomiala = Mass, Poison = Poison, Normala= Norm,
NormalaCorectie= NormCorect, CampPaulson = CampP)
print(tabela)


##ex 5
for(set in 0:5){
  BP <- 0
  BN <- 0
  BNC <- 0
  BCP <- 0
  for(i in 1:10){
    if(abs(tabela[set*10+i,2] - tabela[set*10+i,3]) >= BP )
      BP = abs(tabela[set*10+i,2] - tabela[set*10+i,3])

    if(abs(tabela[set*10+i,2] - tabela[set*10+i,4]) >= BN )
      BN = abs(tabela[set*10+i,2] - tabela[set*10+i,4])

    if(abs(tabela[set*10+i,2] - tabela[set*10+i,5]) >= BNC )
      BNC = abs(tabela[set*10+i,2] - tabela[set*10+i,5])
```

```
    if(abs(tabela[set*10+i,2] - tabela[set*10+i,6]) >= BCP)
      BCP = abs(tabela[set*10+i,2] - tabela[set*10+i,6])

  }

  print(BP)
  print(BN)
  print(BNC)
  print(BCP)
}

plot(tabela$Binomiala, type="o",col="red", main="Binomial approximations")
lines(tabela$Poison,Type="o",col="green")
lines(tabela$Normala,Type="o",col="blue")
lines(tabela$NormalaCorectie,Type="o",col="yellow")
lines(tabela$CampPaulson,Type="o",col="coral")
legend("bottomleft",c("Binomiala","Poison","Normala","Normala Corectie", "Camp Paulson"),
col=c("red","green","blue","yellow","coral"),pch=15)



## ex 6

  skewnormal <- function(miu, sigma, lambda) {
    return(Vectorize(function(x) {
      return(2 / sigma *
               dnorm((x - miu) / sigma, mean = 0, sd = 1) *
               pnorm((lambda * ((x - miu) / sigma)), mean = 0, sd = 1))
    }))
  }

  plot(skewnormal(1, 5, 10),col="blue")
  plot(skewnormal(1, 25, 5), add=TRUE, col = "black")
  plot(skewnormal(1, 50, 15), add=TRUE, col = "dark green")
  plot(skewnormal(1, 3, 20), add=TRUE, col = "yellow")
  plot(skewnormal(1, 100, 20), add=TRUE, col = "red")
  legend("topleft", c("5","25","50","3","100"),col=c("blue","black","dark green","yellow",
"red"), pch = 10)



## ex 7
n <- 25
myfunction <- function(n,p){
  dreapta <- (n*p*(1-p))/((1-2*p)^2)
  lambdasquared <- uniroot(function(x)  ((1-(2/pi)*((x)/(1+x)))^3)/((2/pi)*((4/pi-
1)^2)*((x/(1+x))^3))-dreapta, c(0,500), extendInt = "yes")$root

  lambda <- sign(1-2*p)*sqrt(lambdasquared)
  sigma <- sqrt((n*p*(1-p))/(1-(2/pi)*((lambda^2)/(1+lambda^2))))
  miu <- n*p - sigma*sqrt((2/pi)*((lambda^2)/(1+lambda^2)))

  library(sn)
  barplot(dsn(x= 0:n, dp=c(miu,sigma,lambda)),col=rgb(0.3,0.1,0.23,0.5), add=T)

}
```

```
# pt p=0.05

barplot(pbinom(q=0:n,size=n, prob=0.05), col=rgb(0.9,0.5,0.23,0.5))
myfunction(n,0.05)

# pt p=0.1

barplot(pbinom(q=0:n, size=n, prob=0.1), col=rgb(0.9,0.5,0.23,0.5))
myfunction(n,0.1)


## ex 8

    i <- 1
    myk <- numeric(60)
    Binomiala <- numeric(60)
    NormalaAsimetrica <- numeric(60)

    for(n in c(25,50,100)){
      for(p in c(0.05, 0.1)){
        for(k in 1:10){
          myk[i] <- k
          Binomiala[i] <- pbinom(k, n, p)
          dreapta <- (n*p*(1-p))/((1-2*p)^2)
          lambdasquared <- uniroot(function(x)  ((1-(2/pi)*((x)/(1+x)))^3)/((2/pi)*((4/pi-
1)^2)*((x/(1+x))^3))-dreapta, c(0,500), extendInt = "yes")$root

          lambda <- sign(1-2*p)*sqrt(lambdasquared)
          sigma <- sqrt((n*p*(1-p))/(1-(2/pi)*((lambda^2)/(1+lambda^2))))
          miu <- n*p - sigma*sqrt((2/pi)*((lambda^2)/(1+lambda^2)))
          NormalaAsimetrica[i]<- dsn(x=k, dp=c(miu,sigma,lambda))

          i <- i+1

        }
      }
    }

    tabela2 <- data.frame(k = myk, Binomiala =Binomiala, NormalaAsimetrica= NormalaAsimetrica)
    print(tabela2)

   ## afisarea distante Kolomogorov
    for(set in 0:5){
      maxim <- 0
      for(k in 1:10){
        if(abs(tabela2[set*10+k,2] - tabela2[set*10+k,3]) > maxim )
          Kolomogorov = abs(tabela2[set*10+k,2] - tabela2[set*10+k,3])
      }
      print(Kolomogorov)
    }



    plot(tabela2$Binomiala, type="o",col="red", main="Binomial - Skew Normal approximations")
    lines(tabela2$NormalaAsimetrica,Type="o",col="blue")
    legend("topright",c("Binomiala","Normala Asimetrica"), col=c("red","blue"),pch=15)
```

Cod problema 2

```
# Punctul a

fgamaux <-function(x,a) {
  return (x^(a-1)*exp(-x));
}

fgam <- function(x) {
  if(x==1)
    return (1)
  if(x==1/2)
    return (sqrt(pi))
  if(x%%1==0 && x>0)
    return (factorial(x-1))
  if(x>1)
    return ((x-1)*fgam(x-1))
  return (integrate(fgamaux, 0, Inf,a=x)$value)
}

fgam(0.2)
gamma(0.2)

# Punctul b

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))
}

beta(2,4)
fbet(2,4)

# Punctul c

fdistgamma <- function(x,a,b) {
  if(x>0 && a>0 && b>0)
    return ((x^(a-1)*exp(-x/b))/(b^a*fgam(a)))
  return (0)
}

fdistbeta <-function(x,a,b) {
  if(0<x && x<1 && a>0 && b>0)
    return ((x^(a-1)*(1-x)^(b-1))/(fbet(a,b)))
  return (0)
}

# 1) P(X < 3)
fprobgamma1 <- function(A,B) {
  return (integrate(fdistgamma,0,3,a=A,b=B)$value)
}
```

```
# 2) P(2 < X < 5)
fprobgamma2 <- function(A,B) {
  F1 <- integrate(fdistgamma,0,5,a=A,b=B)
  F2 <- integrate(fdistgamma,0,2,a=A,b=B)
  return (F1$value-F2$value)
}

# 3) P(3 < X < 4 | X > 2)
fprobgamma3 <- function(A,B) {
  F1 <- integrate(fdistgamma, 0, 4,a=A,b=B)$value
  F2 <- integrate(fdistgamma, 0, 3,a=A,b=B)$value
  F3 <- integrate(fdistgamma, 0, 2,a=A,b=B)$value
  return (F1-F2)/(1-F3)
}

# 4) P(Y > 2)
fprobbeta4 <-function(A,B) {
  F1 <- integrate(Vectorize(fdistbeta),0,2,a=A,b=B)$value
  return (1 - F1)
}

# 5) P(4 < X < 6)
fprobgamma5 <- function(A,B) {
  F1 <- integrate(fdistgamma, 0, 6,a=A,b=B)$value
  F2 <- integrate(fdistgamma, 0, 4,a=A,b=B)$value
  return (F1-F2)
}

# 6) P(0 < X < 1 | X < 7)
fprobgamma6 <-function(A,B) {
  F1 <- integrate(Vectorize(fdistgamma), 0, 1,a=A,b=B)$value
  F2 <- integrate(Vectorize(fdistgamma), 0, 0,a=A,b=B)$value
  F3 <- integrate(Vectorize(fdistgamma), 0, 7,a=A,b=B)$value
  return ((F1-F2)/F3)
}

p1 <- fprobgamma1(1,2)
print(p1)
p2 <- fprobgamma2(2,3)
print(p2)
p3 <- fprobgamma3(3,4)
print(p3)
p4 <- fprobbeta4(5,10)
print(p4)
p5 <- fprobgamma5(1,3)
print(p5)
p6 <- fprobgamma6(2,5)
print(p6)

# Punctul d

sdistgamma <- function(x,a,b) {
  if(x>0 && a>0 && b>0)
    return ((x^(a-1)*exp(-x/b))/(b^a*gamma(a)))
  return (0)
}
```

```
sdistbeta <-function(x,a,b) {
  if(0<x && x<1 && a>0 && b>0)
    return ((x^(a-1)*(1-x)^(b-1))/(beta(a,b)))
  return (0)
}

# 1) P(X < 3)
sprobgamma1 <- function(A,B) {
  return (integrate(sdistgamma,0,3,a=A,b=B)$value)
}

# 2) P(2 < X < 5)
sprobgamma2 <- function(A,B) {
  F1 <- integrate(sdistgamma,0,5,a=A,b=B)
  F2 <- integrate(sdistgamma,0,2,a=A,b=B)
  return (F1$value-F2$value)
}

# 3) P(3 < X < 4 | X > 2)
sprobgamma3 <- function(A,B) {
  F1 <- integrate(sdistgamma, 0, 4,a=A,b=B)$value
  F2 <- integrate(sdistgamma, 0, 3,a=A,b=B)$value
  F3 <- integrate(sdistgamma, 0, 2,a=A,b=B)$value
  return (F1-F2)/(1-F3)
}

# 4) P(Y > 2)
sprobbeta4 <-function(A,B) {
  F1 <- integrate(Vectorize(sdistbeta),0,2,a=A,b=B)$value
  return (1 - F1)
}

# 5) P(4 < X < 6)
sprobgamma5 <- function(A,B) {
  F1 <- integrate(Vectorize(sdistgamma), 0, 6,a=A,b=B)$value
  F2 <- integrate(Vectorize(sdistgamma), 0, 4,a=A,b=B)$value
  return (F1-F2)
}

# 6) P(0 < X < 1 | X < 7)
sprobgamma6 <-function(A,B) {
  F1 <- integrate(Vectorize(sdistgamma), 0, 1,a=A,b=B)$value
  F2 <- integrate(Vectorize(sdistgamma), 0, 0,a=A,b=B)$value
  F3 <- integrate(Vectorize(sdistgamma), 0, 7,a=A,b=B)$value
  return ((F1 - F2) / F3)
}

s1 <- sprobgamma1(1,2)
print(s1)
s2 <- sprobgamma2(2,3)
print(s2)
s3 <- sprobgamma3(3,3)
print(s3)
s4 <- sprobbeta4(5,10)
print(s4)
s5 <- sprobgamma5(1,3)
```

```
print(s5)
s6 <- sprobgamma6(2,5)
print(s6)

vf <- c(p1,p2,p3,p4,p5,p6)
vs <- c(s1,s2,s3,s4,s5,s6)

mat <- cbind(Punctul_C=vf,Punctul_D=vs)
print(mat)
```

Cod problema 3

```
# Functia care genereaza repartitia comuna incompleta

frepcomgen = function(m, n) {
  m=2
  n=3

  # Generam m valori pentru x in intervalul [0, m * 10] si
  # n valori pentru y in [0, n * 10]
  xValues <- sample.int(m * 10, m);
  yValues <- sample.int(n * 10, n);

  # Adaugam in acesti vectori Qi si Pi pentru capetele de tabel
  xValues <- c(xValues, 'Qi')
  yValues <- c(yValues, 'Pi')

  # Construim tabelul de repartitie, reprezentat ca o matrice, unde valoarea -1
  # semnifica faptul ca probabilitatea pentru (X = xi si Y=yi) nu este cunoscuta
  repart <- matrix(-1, nrow=m+1, ncol=n+1)
  rownames(repart) <- xValues
  colnames(repart) <- yValues
  repart[m+1, n+1] = 1



  # Completam repartitia cu valori aleatoare, dar care vor respecta totusi conditia ca
  # suma pe linii/coloane sa fie mai mica ca 1 (sau egala)
```

```
  repart[1, 4] <- runif(1, 0, 1)
  repart[1, 1] <- runif(1, 0, repart[1, 4])

  repart[3, 1] <- runif(1, repart[1, 1], 1)

  repart[2, 2] <- runif(1, 0, 1 - repart[3, 1])
  while(repart[3, 2] - repart[2, 2] + repart[1, 1] > repart[1, 4] ||
          repart[3, 1] - repart[1, 1] + repart[2, 2] > 1 - repart[1, 4])
    repart[2, 2] <- runif(1, 0, 1 - repart[3, 1])

  repart[3, 2] <- runif(1, min(repart[2, 2], repart[3, 1]), 1 - max(repart[2, 2], repart[3,
1]))
  while(repart[3, 2] + repart[3, 1] > 1 || repart[2, 2] > repart[3, 2])
    repart[3, 2] <- runif(1, min(repart[2, 2], repart[3, 1]), 1 - max(repart[2, 2], repart[3,
1]))

  return(repart)
}

# Functia fcomplrepcom efectueaza operatii pe linii si coloane pentru a determina valorile
# necunoscute din repartitia comuna
fcomplrepcom <- function(repart) {
  repart[2, 4] = 1 - repart[1, 4]
  repart[1, 2] = repart[3, 2] - repart[2, 2]
  repart[1, 3] = repart[1, 4] - repart[1, 1] - repart[1, 2]
  repart[2, 1] = repart[3, 1] - repart[1, 1]
  repart[2, 3] = repart[2, 4] - repart[2, 2] - repart[2, 1]
  repart[3, 3] = 1 - repart[3, 1] - repart[3, 2]

  data.frame(repart)
}

repart <- frepcomgen(2, 3)
#repart
```

```
fcomplrepcom(repart)
```