Tema probabilitati si statistica

05 februarie 2020

## 1. Problema 1

### 1) Generarea celor 1000 de realizari independente din fiecare repartitie si calculul mediei si variantei pt fiecare esantion.

#### Functiile mean() si var se folosesc pentru aflarea mediei, respectiv variantei.

# Normala

N = rnorm(1000,500)  
mean(N)

## [1] 499.989

var(N)

## [1] 0.9827341

# Poisson

P = rpois(1000,500)  
mean(P)

## [1] 500.517

var(P)

## [1] 468.288

# Exponentiala

E = rexp(1000,10)  
mean(E)

## [1] 0.09678547

var(E)

## [1] 0.009944151

# Bionomiala

B = rbinom(1000, 10, 0.1)  
mean(B)

## [1] 1.027

var(B)

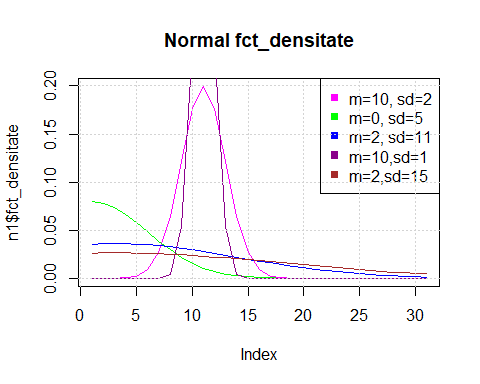
## [1] 0.9011722

### 2), 3) Ilustrarea grafica a functiilor de masa/densitate ulitizand si a functiilor de repartitie pentru cele 5 seturi de parametrii diferiti

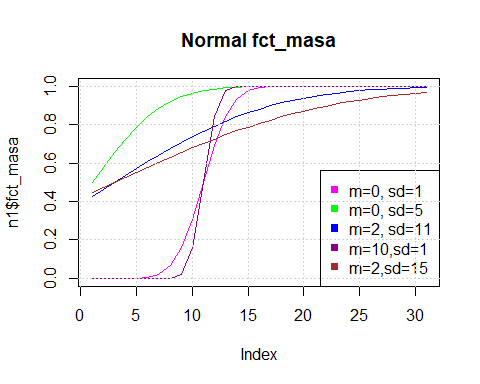
#### Functiile dbinom, dpois, dexp, dnorm genereaza densitatea/masa fiecarei repartitii.

#### Functiile pbinom, ppois, pexp, pnorm genereaza repartitiile corespunzatoare.

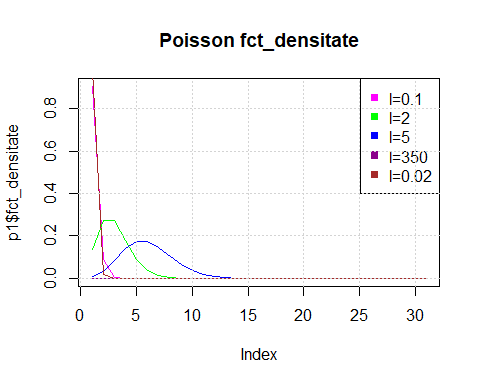
# graficele normal  
n1 = data.frame(fct\_densitate=dnorm(0:30, 10, 2), fct\_masa=pnorm(0:30, 10, 2))  
n2 = data.frame(fct\_densitate=dnorm(0:30, 0, 5), fct\_masa=pnorm(0:30, 0, 5))  
n3 = data.frame(fct\_densitate=dnorm(0:30, 2, 11), fct\_masa=pnorm(0:30, 2, 11))  
n4 = data.frame(fct\_densitate=dnorm(0:30, 10, 1), fct\_masa=pnorm(0:30, 10, 1))  
n5 = data.frame(fct\_densitate=dnorm(0:30, 2, 15), fct\_masa=pnorm(0:30, 2, 15))  
  
plot(n1$fct\_densitate, type="l",col="magenta", main="Normal fct\_densitate")  
lines(n2$fct\_densitate,type="l",col="green")  
lines(n3$fct\_densitate,type="l",col="blue")  
lines(n4$fct\_densitate,type="l",col="darkmagenta")  
lines(n5$fct\_densitate,type="l",col="brown")  
legend("topright",c("m=10, sd=2","m=0, sd=5","m=2, sd=11","m=10,sd=1", "m=2,sd=15"), col=c("magenta","green","blue","darkmagenta","brown"),pch=15)  
grid()



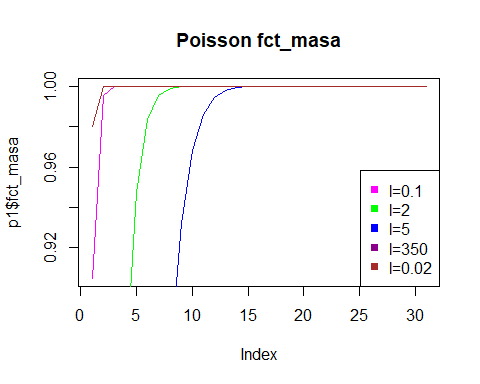
plot(n1$fct\_masa,type="l",col="magenta", main="Normal fct\_masa")   
lines(n2$fct\_masa,type="l",col="green")  
lines(n3$fct\_masa,type="l",col="blue")  
lines(n4$fct\_masa,type="l",col="darkmagenta")  
lines(n5$fct\_masa,type="l",col="brown")  
legend("bottomright",c("m=0, sd=1","m=0, sd=5","m=2, sd=11","m=10,sd=1", "m=2,sd=15"), col=c("magenta","green","blue","darkmagenta","brown"),pch=15)  
grid()



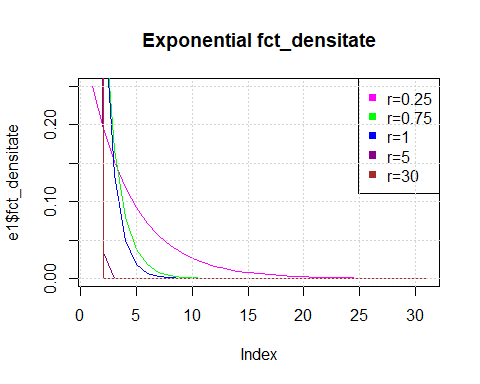
# graficele poisson  
p1 = data.frame(fct\_densitate=dpois(0:30, 0.1), fct\_masa=ppois(0:30, 0.1))  
p2 = data.frame(fct\_densitate=dpois(0:30, 2), fct\_masa=ppois(0:30, 2))  
p3 = data.frame(fct\_densitate=dpois(0:30, 5), fct\_masa=ppois(0:30, 5))  
p4 = data.frame(fct\_densitate=dpois(0:30, 350), fct\_masa=ppois(0:30, 350))  
p5 = data.frame(fct\_densitate=dpois(0:30, 0.02), fct\_masa=ppois(0:30, 0.02))  
  
plot(p1$fct\_densitate, type="l",col="magenta", main="Poisson fct\_densitate")   
lines(p2$fct\_densitate,type="l",col="green")  
lines(p3$fct\_densitate,type="l",col="blue")  
lines(p4$fct\_densitate,type="l",col="darkmagenta")  
lines(p5$fct\_densitate,type="l",col="brown")  
legend("topright",c("l=0.1","l=2","l=5","l=350", "l=0.02"), col=c("magenta","green","blue","darkmagenta","brown"),pch=15)  
grid()



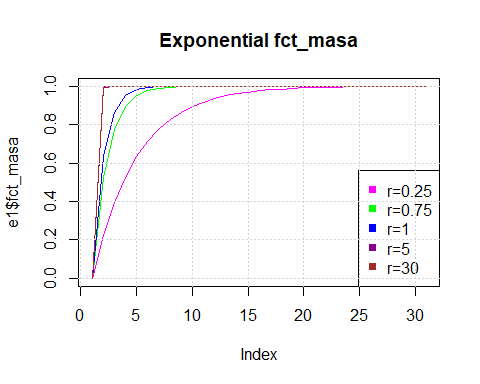
plot(p1$fct\_masa,type="l",col="magenta", main="Poisson fct\_masa")   
lines(p2$fct\_masa,type="l",col="green")  
lines(p3$fct\_masa,type="l",col="blue")  
lines(p4$fct\_masa,type="l",col="darkmagenta")  
lines(p5$fct\_masa,type="l",col="brown")  
legend("bottomright",c("l=0.1","l=2","l=5","l=350", "l=0.02"), col=c("magenta","green","blue","darkmagenta","brown"),pch=15)



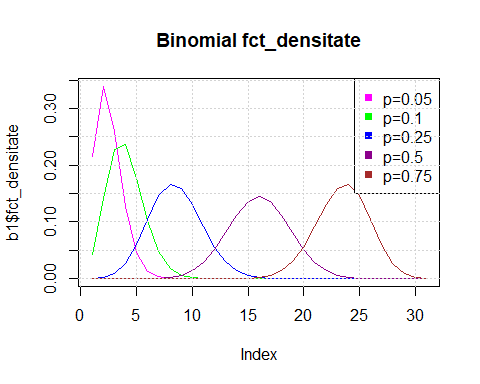
# graficele exponential  
e1 = data.frame(fct\_densitate=dexp(0:30, 0.25), fct\_masa=pexp(0:30, 0.25))  
e2 = data.frame(fct\_densitate=dexp(0:30, 0.75), fct\_masa=pexp(0:30, 0.75))  
e3 = data.frame(fct\_densitate=dexp(0:30, 1), fct\_masa=pexp(0:30, 1))  
e4 = data.frame(fct\_densitate=dexp(0:30, 5), fct\_masa=pexp(0:30, 5))  
e5 = data.frame(fct\_densitate=dexp(0:30, 30), fct\_masa=pexp(0:30, 30))  
  
plot(e1$fct\_densitate, type="l",col="magenta", main="Exponential fct\_densitate")  
lines(e2$fct\_densitate,type="l",col="green")  
lines(e3$fct\_densitate,type="l",col="blue")  
lines(e4$fct\_densitate,type="l",col="darkmagenta")  
lines(e5$fct\_densitate,type="l",col="brown")  
legend("topright",c("r=0.25","r=0.75","r=1","r=5", "r=30"), col=c("magenta","green","blue","darkmagenta","brown"),pch=15)  
grid()



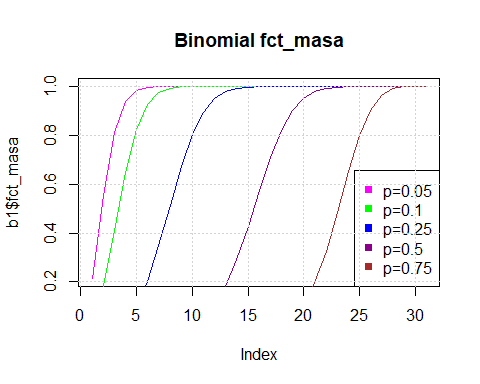
plot(e1$fct\_masa,type="l" ,col="magenta", main="Exponential fct\_masa")   
lines(e2$fct\_masa,type="l",col="green")  
lines(e3$fct\_masa,type="l",col="blue")  
lines(e4$fct\_masa,type="l",col="darkmagenta")  
lines(e5$fct\_masa,type="l",col="brown")  
legend("bottomright",c("r=0.25","r=0.75","r=1","r=5", "r=30"), col=c("magenta","green","blue","darkmagenta","brown"),pch=15)  
grid()



# graficele binomial  
b1 = data.frame(fct\_densitate=dbinom(0:30, 30, 0.05), fct\_masa=pbinom(0:30, 30, 0.05))  
b2 = data.frame(fct\_densitate=dbinom(0:30, 30, 0.1), fct\_masa=pbinom(0:30, 30, 0.1))  
b3 = data.frame(fct\_densitate=dbinom(0:30, 30, 0.25), fct\_masa=pbinom(0:30, 30, 0.25))  
b4 = data.frame(fct\_densitate=dbinom(0:30, 30, 0.5), fct\_masa=pbinom(0:30, 30, 0.5))  
b5 = data.frame(fct\_densitate=dbinom(0:30, 30, 0.75), fct\_masa=pbinom(0:30, 30, 0.75))  
  
plot(b1$fct\_densitate, type="l",col="magenta", main="Binomial fct\_densitate")  
lines(b2$fct\_densitate,type="l",col="green")  
lines(b3$fct\_densitate,type="l",col="blue")  
lines(b4$fct\_densitate,type="l",col="darkmagenta")  
lines(b5$fct\_densitate,type="l",col="brown")  
legend("topright",c("p=0.05","p=0.1","p=0.25","p=0.5", "p=0.75"), col=c("magenta","green","blue","darkmagenta","brown"),pch=15)  
grid()



plot(b1$fct\_masa,type="l",col="magenta", main="Binomial fct\_masa")  
lines(b2$fct\_masa,type="l",col="green")  
lines(b3$fct\_masa,type="l",col="blue")  
lines(b4$fct\_masa,type="l",col="darkmagenta")  
lines(b5$fct\_masa,type="l",col="brown")  
legend("bottomright",c("p=0.05","p=0.1","p=0.25","p=0.5", "p=0.75"), col=c("magenta","green","blue","darkmagenta","brown"),pch=15)  
grid()



### 4) Aproximarile functiilor de repartitie si de masa

AppBp <- function(n, p) {  
 x <- matrix(numeric(10 \* 6), ncol = 6,  
 dimnames = list(  
 1:10,  
 c(  
 "K",  
 "Binomiala",  
 "Poisson",  
 "Normala",  
 "Normala Corectie",  
 "Camp-Paulson"  
   
 )  
 ))  
 x[,1] <- 1:10  
 for(k in 1:10){  
 lambda <- n \* p  
   
 # binomiala  
 x[, 2] <- pbinom(1:10, n, p)  
   
 # poisson  
 x[, 3] <- ppois(1:10, lambda)  
 error <- max(abs(x[, 1] - x[, 2]))  
   
 # normala corectie  
 q = 1 - p  
 y <-rnorm(n, n \* p, sqrt(n \* p \* q)) # generare n v.a. normale de medie np  
 z.cc <- ((k + 0.5) - n\*p / sqrt(n\*p\*q)) # cu coeficient de corectie  
 x[k, 5] <- pnorm(z.cc)  
   
 # normala  
 z.ncc <- (k - n\*p / sqrt(n\*p\*q)) # fara coeficient de corectie  
 x[k, 4] <- pnorm(z.ncc)  
   
 # paulson  
 a <- (1 / (9 \* (n - k)))  
 b <- 1 / (9 \* (k + 1))  
 r <- ((k + 1) \* (1 - p) / (p \* (n - k)))  
 u <- 1 - a  
 C <- (1 - b) \* (r ^ (2 / 3))  
 C <- C - u  
 o <- a + b\*(r ^ (2 / 3))  
 C <- C / (sqrt(o))  
 x[k, 6] <- pnorm( C)  
 }  
  
 list(x = as.data.frame(x),  
 error = error,  
 param = c(n, p, lambda))  
 return (x)  
}

#### Apeluri pt n = {25, 50, 100} si p = {0.05, 0.1}

AppBp(25, 0.05)

## K Binomiala Poisson Normala Normala Corectie Camp-Paulson  
## 1 1 0.6423759 0.6446358 0.4415350 0.6379263 0.8452494  
## 2 2 0.8728935 0.8684677 0.8031485 0.9119596 0.9978167  
## 3 3 0.9659094 0.9617309 0.9680532 0.9906867 0.9999986  
## 4 4 0.9928351 0.9908757 0.9978340 0.9996002 1.0000000  
## 5 5 0.9987870 0.9981619 0.9999416 0.9999933 1.0000000  
## 6 6 0.9998312 0.9996799 0.9999994 1.0000000 1.0000000  
## 7 7 0.9999804 0.9999509 1.0000000 1.0000000 1.0000000  
## 8 8 0.9999981 0.9999933 1.0000000 1.0000000 1.0000000  
## 9 9 0.9999998 0.9999992 1.0000000 1.0000000 1.0000000  
## 10 10 1.0000000 0.9999999 1.0000000 1.0000000 1.0000000

AppBp(25,0.1)

## K Binomiala Poisson Normala Normala Corectie Camp-Paulson  
## 1 1 0.2712059 0.2872975 0.2524925 0.4338162 0.1684857  
## 2 2 0.5370941 0.5438131 0.6305587 0.7976716 0.6391512  
## 3 3 0.7635914 0.7575761 0.9087888 0.9666235 0.9554531  
## 4 4 0.9020064 0.8911780 0.9901847 0.9976967 0.9989229  
## 5 5 0.9666001 0.9579790 0.9995709 0.9999368 0.9999959  
## 6 6 0.9905236 0.9858127 0.9999927 0.9999993 1.0000000  
## 7 7 0.9977387 0.9957533 1.0000000 1.0000000 1.0000000  
## 8 8 0.9995425 0.9988597 1.0000000 1.0000000 1.0000000  
## 9 9 0.9999210 0.9997226 1.0000000 1.0000000 1.0000000  
## 10 10 0.9999883 0.9999384 1.0000000 1.0000000 1.0000000

AppBp(50,0.05)

## K Binomiala Poisson Normala Normala Corectie Camp-Paulson  
## 1 1 0.2794318 0.2872975 0.2669005 0.4513647 0.1828878  
## 2 2 0.5405331 0.5438131 0.6472051 0.8099700 0.6526215  
## 3 3 0.7604080 0.7575761 0.9158653 0.9697948 0.9550052  
## 4 4 0.8963832 0.8911780 0.9912915 0.9979976 0.9986459  
## 5 5 0.9622238 0.9579790 0.9996346 0.9999473 0.9999916  
## 6 6 0.9882136 0.9858127 0.9999940 0.9999995 1.0000000  
## 7 7 0.9968117 0.9957533 1.0000000 1.0000000 1.0000000  
## 8 8 0.9992440 0.9988597 1.0000000 1.0000000 1.0000000  
## 9 9 0.9998414 0.9997226 1.0000000 1.0000000 1.0000000  
## 10 10 0.9999704 0.9999384 1.0000000 1.0000000 1.0000000

AppBp(50,0.1)

## K Binomiala Poisson Normala Normala Corectie Camp-Paulson  
## 1 1 0.03378586 0.04042768 0.08738701 0.1957162 0.001713538  
## 2 2 0.11172876 0.12465202 0.36053744 0.5568460 0.019825210  
## 3 3 0.25029391 0.26502592 0.73988062 0.8734760 0.125490618  
## 4 4 0.43119841 0.44049329 0.94980619 0.9839425 0.413623698  
## 5 5 0.61612301 0.61596065 0.99589097 0.9991638 0.765657028  
## 6 6 0.77022684 0.76218346 0.99986525 0.9999829 0.953299166  
## 7 7 0.87785492 0.86662833 0.99999828 0.9999999 0.995855131  
## 8 8 0.94213279 0.93190637 0.99999999 1.0000000 0.999847728  
## 9 9 0.97546206 0.96817194 1.00000000 1.0000000 0.999997806  
## 10 10 0.99064540 0.98630473 1.00000000 1.0000000 0.999999988

AppBp(100,0.05)

## K Binomiala Poisson Normala Normala Corectie Camp-Paulson  
## 1 1 0.03708121 0.04042768 0.09780554 0.2135519 0.002085054  
## 2 2 0.11826298 0.12465202 0.38431884 0.5815431 0.022876992  
## 3 3 0.25783866 0.26502592 0.75985701 0.8860609 0.136345882  
## 4 4 0.43598130 0.44049329 0.95598131 0.9863025 0.427455608  
## 5 5 0.61599913 0.61596065 0.99659343 0.9993267 0.769405774  
## 6 6 0.76601398 0.76218346 0.99989466 0.9999870 0.951456458  
## 7 7 0.87203952 0.86662833 0.99999874 0.9999999 0.995095354  
## 8 8 0.93691041 0.93190637 0.99999999 1.0000000 0.999772831  
## 9 9 0.97181171 0.96817194 1.00000000 1.0000000 0.999995300  
## 10 10 0.98852759 0.98630473 1.00000000 1.0000000 0.999999957

AppBp(100,0.1)

## K Binomiala Poisson Normala Normala Corectie Camp-Paulson  
## 1 1 0.0003216881 0.0004993992 0.009815329 0.03337651 2.163691e-07  
## 2 2 0.0019448847 0.0027693957 0.091211220 0.20232838 2.574296e-06  
## 3 3 0.0078364871 0.0103360507 0.369441340 0.56618383 3.286724e-05  
## 4 4 0.0237110827 0.0292526881 0.747507462 0.87832750 3.529704e-04  
## 5 5 0.0575768865 0.0670859629 0.952209648 0.98486986 2.873930e-03  
## 6 6 0.1171556154 0.1301414209 0.996169619 0.99922902 1.689914e-02  
## 7 7 0.2060508618 0.2202206466 0.999877134 0.99998455 7.032530e-02  
## 8 8 0.3208738884 0.3328196788 0.999998469 0.99999988 2.069598e-01  
## 9 9 0.4512901654 0.4579297145 0.999999993 1.00000000 4.384381e-01  
## 10 10 0.5831555123 0.5830397502 1.000000000 1.00000000 6.956441e-01

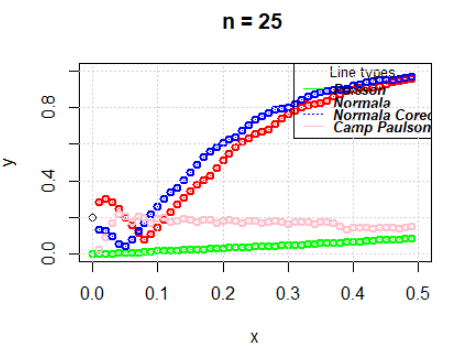
### 5) Pentru fiecare n = {25, 50, 100} se ilustreaza erorile maximale absolute

#### Se adauga in tabel cate o linie pentru eroarea maximala absoluta dintre repartitia binomiala si cele patru aproximari

AppBp2 <- function(n, p) {  
x<- matrix(numeric(10 \* 9), ncol = 9,  
 dimnames = list(  
 1:10,  
 c(  
   
 "Binomiala",  
 "Poisson",  
 "Normala",  
 "Normala Corectie",  
 "Camp-Paulson",  
 "Binom-Pois",  
 "Binom-Norm",  
 "Binom-NormC",  
 "Binom-Cam"  
 )  
 ))  
 x[,1] <- 1:10  
 for(k in 1:10){  
 lambda <- n \* p  
   
 # binomiala  
 x[, 2] <- pbinom(1:10, n, p)  
   
 # poisson  
 x[, 3] <- ppois(1:10, lambda)  
 error <- max(abs(x[, 1] - x[, 2]))  
   
 # normala corectie  
 q = 1 - p  
 y <-rnorm(n, n \* p, sqrt(n \* p \* q)) # generare n v.a. normale de medie np  
 z.cc <- ((k + 0.5) - n\*p / sqrt(n\*p\*q)) # cu coeficient de corectie  
 x[k, 5] <- pnorm(z.cc)  
   
 # normala  
 z.ncc <- (k - n\*p / sqrt(n\*p\*q)) # fara coeficient de corectie  
 x[k, 4] <- pnorm(z.ncc)  
   
 # paulson  
 a <- (1 / (9 \* (n - k)))  
 b <- 1 / (9 \* (k + 1))  
 r <- ((k + 1) \* (1 - p) / (p \* (n - k)))  
 u <- 1 - a  
 C <- (1 - b) \* (r ^ (2 / 3))  
 C <- C - u  
 o <- a + b\*(r ^ (2 / 3))  
 C <- C / (sqrt(o))  
 x[k, 6] <- pnorm( C)  
 }  
 x[,6] <- max(abs((x[,1]-x[,2])))  
 x[,7] <- max(abs((x[,1]-x[,3])))  
 x[,8] <- max(abs((x[,1]-x[,4])))  
 x[,9] <- max(abs((x[,1]-x[,5])))  
 list(x = as.data.frame(x),  
 error = error,  
 param = c(n, p, lambda))  
 return (x)  
}

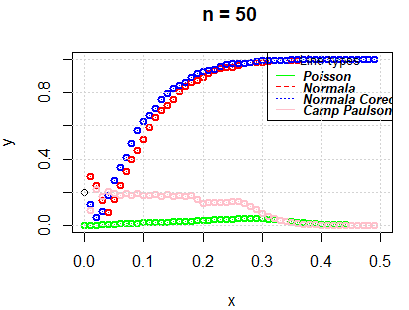
# n= 25

n<-25  
p<-0.01  
  
y <- matrix(nrow=10, ncol=4)   
for (i in 1:4) y[,i] <- matr[,i+6-1]  
plot(x=0,y=0.2,main="n = 25",xlab="x" ,lty=2,ylab="y",ylim=c(0,1),xlim=c(0,0.5))  
  
while(p<=0.5)  
{  
 x<-c(p,p,p,p,p,p,p,p,p,p)  
 matr<-AppBp2(n,p)  
 y <- matrix(nrow=10, ncol=4)   
 for (i in 1:4) y[,i] <- matr[,i+6-1]  
   
 lines(x=x, y=y[,1], col='green',lwd=2,type="o")  
 lines(x=x, y=y[,2], col='red',lwd=2,type="o")  
 lines(x=x,y=y[,3], col='blue',lwd=2,type="o")  
 lines(x=x, y=y[,4], col='pink',lwd=2,type="o")  
 p<-p+0.01  
  
}  
  
legend("topright", legend=c("Poisson", "Normala","Normala Corectie","Camp Paulson"),col=c("green","red", "blue","pink"), lty=1:3, cex=0.8 ,title="Line types", text.font=4, bg='transparent')



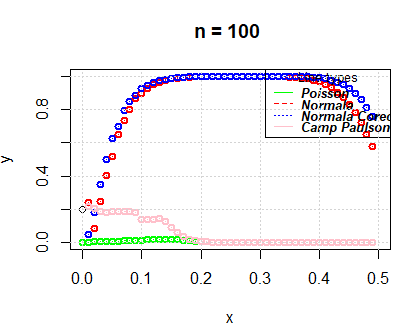
# n= 50

n<-50  
p<-0.01  
  
y <- matrix(nrow=10, ncol=4)   
for (i in 1:4) y[,i] <- matr[,i+6-1]  
plot(x=0,y=0, main="n = 50", xlab="x", ylab="y",ylim=c(0,1),xlim=c(0,0.5))  
while(p<=0.5)  
{  
 x<-c(p,p,p,p,p,p,p,p,p,p)  
 matr<-AppBp2(n,p)  
 y <- matrix(nrow=10, ncol=4)   
 for (i in 1:4) y[,i] <- matr[,i+6-1]  
 lines(x=x, y=y[,1], col='red',type="l",lwd=10)  
 lines(x=x, y=y[,2], col='green',type="l",lwd=10)  
   
 lines(x=x,y=y[,3], col='blue',type="l",lwd=10)  
   
 lines(x=x, y=y[,4], col='pink',type="l",lwd=10)  
   
 p<-p+0.01  
   
   
}  
  
legend("topright", legend=c("Poisson", "Normala","Normala Corectie","Camp Paulson"),col=c("green","red", "blue","pink"), lty=1:3, cex=0.8 ,title="Line types", text.font=4, bg='transparent')



# n= 100

n<-100  
p<-0  
  
y <- matrix(nrow=10, ncol=4)   
for (i in 1:4) y[,i] <- matr[,i+6-1]  
plot(x=0,y=0.2,main="n = 100",xlab="x" ,lty=2,ylab="y",ylim=c(0,1),xlim=c(0,0.5))  
  
while(p<=0.5)  
{  
 x<-c(p,p,p,p,p,p,p,p,p,p)  
 matr<-AppBp2(n,p)  
 y <- matrix(nrow=10, ncol=4)   
 for (i in 1:4) y[,i] <- matr[,i+6-1]  
 lines(x=x, y=y[,1], col='green',lwd=2,type="o")  
 lines(x=x, y=y[,2], col='red',lwd=2,type="o")  
 lines(x=x,y=y[,3], col='blue',lwd=2,type="o")  
 lines(x=x, y=y[,4], col='pink',lwd=2,type="o")  
 p<-p+0.01  
   
}  
  
legend("topright", legend=c("Poisson", "Normala","Normala Corectie","Camp Paulson"),col=c("green","red", "blue","pink"), lty=1:3, cex=0.8 ,title="Line types", text.font=4, bg='transparent')  
grid()



## 2. Problema 2

#### Se vor folosi urmatoarele librarii

library(pracma)  
library(data.table)

### a) Construirea functiei Gamma

##### Parametri

* **value** = valoarea pentru care se doreste calcularea functiei Gamma

##### Returneaza

* valoarea integralei Gamma, calculata dupa formula specificata in enunt

fgam <- function(value)  
{  
 if (value == 1)  
 return(1)  
   
 if (value == 1 / 2)  
 return(sqrt(pi))  
   
 if (value %% 1 == 0 && value > 0)  
 return(factorial(value - 1))  
   
 if (value > 1)  
 return((value - 1) \* fgam(value - 1))  
   
 else  
 if (value < 1) {  
 integrand <- function(x, A) {  
 x ^ (A - 1) \* exp(1) ^ (-x)  
 }  
   
 integrate\_val <-  
 integrate(integrand, lower = 0, upper = 100, A = value)  
 return(integrate\_val$value)  
 }  
}

##### **Exemplu:** Aplicarea functiei Gamma pe un numar intreg, respectiv pe unul rational

fgam(7)

## [1] 720

fgam(3.9)

## [1] 5.29933

### b) Construirea functiei Beta

##### Parametri

* **a, b** = doua valori numerice

##### Returneaza

* valoarea integralei Beta, calculata dupa formula specificata in enunt

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))  
}

##### **Exemplu:** Aplicarea functiei Beta pe doua numere pozitive a caror suma este egala cu 1, respectiv pe alte doua numere oarecare

fbet(0.35, 0.65)

## [1] 3.525892

fbet(13, 8)

## [1] 9.922998e-07

### c) Calculul probabilitatilor

#### Functia de densitate pentru repartitia Gamma

F\_Gamma <- function(x, a, b)  
{  
 if (x > 0 && a > 0 && b > 0) {  
 fct <- ((1 \* x ^ (a - 1) \* exp(1) ^ (-x / b)) / (b ^ a \* fgam(a)))  
 return(fct)  
 }  
 else {  
 print("Parametrii trebuie sa fie pozitivi!")  
 return(0)  
 }  
}

#### Functia de densitate pentru repartitia Beta

F\_Beta <- function(x, a, b)  
{  
 if (x > 0 && x < 1 && a > 0 && b > 0) {  
 fct1 <- ((1 \* x ^ (a - 1) \* (1 - x) ^ (b - 1)) / fbet(a, b))  
 return(fct1)  
 }  
 else  
 {  
 # print("Parametrul x nu este intre 0 si 1")  
 return(0)  
 }  
}

#### Rezolvarea celor 9 subpuncte prin functii separate

##### Parametri

* **val1, val2** = valorile corespunzatoare parametrilor a si b folositi pentru repartitiile Gamma si Beta

##### Returneaza

* rezultatul probabilitatii specificate in enunt, pentru o repartitie cu parametri setati anterior

fprob1 <- function(val1, val2) {  
   
 # X < 3  
   
 fprobgamma1 <- function(val1, val2)  
 {  
 return((integrate(  
 F\_Gamma,  
 lower = 0,  
 upper = 3,  
 a = val1,  
 b = val2  
 ))$value)  
 }  
   
 return(fprobgamma1(val1, val2))  
}

fprob1(2, 5)

## [1] 0.1219014

fprob1(5, 7)

## [1] 8.445787e-05

fprob1(1, 2)

## [1] 0.7768698

fprob2 <- function(val1, val2) {  
   
 # 2 < X < 5  
   
 fprobgamma2 <- function(val1, val2)  
 {  
 return((integrate(  
 F\_Gamma,  
 lower = 2,  
 upper = 5,  
 a = val1,  
 b = val2  
 ))$value)  
 }  
   
 return(fprobgamma2(val1, val2))  
}

fprob2(2, 5)

## [1] 0.2026892

fprob2(5, 7)

## [1] 0.0008464146

fprob2(1, 2)

## [1] 0.2857944

fprob3 <- function(val1, val2)  
{  
 fprobgamma3 <- function(val1, val2)  
 {  
 # 3 < X < 5  
   
 a <- (integrate(  
 F\_Gamma,  
 lower = 3,  
 upper = 4,  
 a = val1,  
 b = val2  
 )$value)  
   
 # X > 2  
   
 b <- (integrate(  
 F\_Gamma,  
 lower = 2,  
 upper = Inf,  
 a = val1,  
 b = val2  
 )$value)  
 return(a / b)  
 }  
   
 return(fprobgamma3(val1, val2))  
}

fprob3(2, 5)

## [1] 0.07385223

fprob3(5, 7)

## [1] 0.0002319724

fprob3(1, 2)

## [1] 0.2386512

fprob4 <- function(val1, val2) {  
   
 # Y > 2  
   
 fprobbeta4 <- function(val1, val2)  
 {  
 return((integrate(  
 Vectorize(F\_Beta),  
 lower = 2,  
 upper = Inf,  
 a = val1,  
 b = val2  
 ))$value)  
 }  
 return(fprobbeta4(val1, val2))  
}

fprob4(2, 5)

## [1] 0

fprob4(5, 7)

## [1] 0

fprob4(1, 2)

## [1] 0

fprob5 <- function(val1, val2) {  
   
 # 4 < X < 6  
   
 fprobgamma5 <- function(val1, val2)  
 {  
 return((integrate(  
 F\_Gamma,  
 lower = 4,  
 upper = 6,  
 a = val1,  
 b = val2  
 ))$value)  
 }  
 return(fprobgamma5(val1, val2))  
}

fprob5(2, 5)

## [1] 0.1461649

fprob5(5, 7)

## [1] 0.001585505

fprob5(1, 2)

## [1] 0.08554821

fprob6 <- function(val1, val2) {  
 fprobgamma6 <- function(val1, val2)  
 {  
   
 # 0 < X < 1  
   
 a <- (integrate(  
 F\_Gamma,  
 lower = 0,  
 upper = 1,  
 a = val1,  
 b = val2  
 )$value)  
   
 # X < 7  
   
 b <- (integrate(  
 F\_Gamma,  
 lower = 0,  
 upper = 7,  
 a = val1,  
 b = val2  
 )$value)  
 return(a / b)  
 }  
   
 return(fprobgamma6(val1, val2))  
}

fprob6(2, 5)

## [1] 0.04293116

fprob6(5, 7)

## [1] 0.0001202964

fprob6(1, 2)

## [1] 0.4057211

fprob7 <- function(val1, val2) {  
   
 # X + Y < 5  
   
 f\_suma\_gamma\_beta <- Vectorize(function(z) {  
 return(integral(function(x) {  
 (((z - x) ^ (val1 - 1) \* exp(-(z - x) / val2)) / ((val2 ^ val1) \* fgam(val1))) \*  
 (((x) ^ (val1 - 1) \* (1 - (x)) ^ (val2 - 1)) / fbet(val1, val2))  
 }, 0, 1))  
 })  
   
 return(integrate(f\_suma\_gamma\_beta, 0, 5)$value)  
   
}

fprob7(2, 5)

## [1] 0.2409693

fprob7(5, 7)

## [1] 0.0005875746

fprob7(1, 2)

## [1] 1.092108

fprob8 <- function(val1, val2) {  
   
 # X - Y > 0.5  
   
 f\_dif\_gamma\_beta <- Vectorize(function(z) {  
 return(integral(function(x) {  
 (((z + x) ^ (val1 - 1) \* exp(-(z + x) / val2)) / ((val2 ^ val1) \* fgam(val1))) \*  
 (((x) ^ (val1 - 1) \* (1 - (x)) ^ (val2 - 1)) / fbet(val1, val2))  
 }, 0, 1))  
 })  
   
 return(integrate(f\_dif\_gamma\_beta, 0.5, Inf)$value)  
   
}

fprob8(2, 5)

## [1] 0.9885107

fprob8(5, 7)

## [1] 0.9999996

fprob8(1, 2)

## [1] 0.6637293

fprob9 <- function(val1, val2)  
{  
   
 # x + Y > 3  
   
 f\_1 <- Vectorize(function(z) {  
 return(integral(function(x) {  
 (((z - x) ^ (val1 - 1) \* exp(-(z - x) / val2)) / ((val2 ^ val1) \* fgam(val1))) \*  
 (((x) ^ (val1 - 1) \* (1 - (x)) ^ (val2 - 1)) / fbet(val1, val2))  
 }, 0, 1))  
 })  
   
 # X - Y > 0.5  
   
 f\_2 <- Vectorize(function(z) {  
 return(integral(function(x) {  
 (((z + x) ^ (val1 - 1) \* exp(-(z + x) / val2)) / ((val2 ^ val1) \* fgam(val1))) \*  
 (((x) ^ (val1 - 1) \* (1 - (x)) ^ (val2 - 1)) / fbet(val1, val2))  
 }, 0, 1))  
   
 })  
   
 return((integral(f\_2, 0.5, Inf)) \* (integral(f\_1, 3, Inf)))  
   
}

fprob9(2, 5)

## For infinite domains Gauss integration is applied!  
## For infinite domains Gauss integration is applied!

## [1] 0.8860948

fprob9(5, 7)

## For infinite domains Gauss integration is applied!  
## For infinite domains Gauss integration is applied!

## [1] 0.9999566

fprob9(1, 2)

## For infinite domains Gauss integration is applied!  
## For infinite domains Gauss integration is applied!

## [1] 0.1762026

### d) Acelasi enunt ca la punctul anterior, dar folosind functiile de sistem pentru Gama si Beta

F\_Gamma\_sistem <- function(x, a, b)  
{  
 return(dgamma(x, shape = a, scale = b))  
}  
  
F\_Beta\_sistem <- function(x, a, b)  
{  
 return(dbeta(x, shape1 = a, shape2 = b))  
}

##### Parametri

* **val1, val2** = valorile corespunzatoare parametrilor a si b folositi pentru repartitiile Gamma si Beta (folosind functii din sistem)

##### Returneaza

* rezultatul probabilitatii specificate in enunt, pentru o repartitie cu parametri setati anterior

fprob1\_sistem <- function(val1, val2) {  
   
 # X < 3  
   
 fprobgamma1 <- function(val1, val2)  
 {  
 return((  
 integrate(  
 F\_Gamma\_sistem,  
 lower = 0,  
 upper = 3,  
 a = val1,  
 b = val2  
 )  
 )$value)  
 }  
   
 return(fprobgamma1(val1, val2))  
}

fprob1\_sistem(2, 5)

## [1] 0.1219014

fprob1\_sistem(5, 7)

## [1] 8.445787e-05

fprob1\_sistem(1, 2)

## [1] 0.7768698

fprob2\_sistem <- function(val1, val2) {  
   
 # 2 < X < 5  
 fprobgamma2 <- function(val1, val2)  
 {  
 return((  
 integrate(  
 F\_Gamma\_sistem,  
 lower = 2,  
 upper = 5,  
 a = val1,  
 b = val2  
 )  
 )$value)  
 }  
   
 return(fprobgamma2(val1, val2))  
}

fprob2\_sistem(2, 5)

## [1] 0.2026892

fprob2\_sistem(5, 7)

## [1] 0.0008464146

fprob2\_sistem(1, 2)

## [1] 0.2857944

fprob3\_sistem <- function(val1, val2)  
{  
 fprobgamma3 <- function(val1, val2)  
 {  
   
 # 3 < X < 5  
   
 a <- (integrate(  
 F\_Gamma\_sistem,  
 lower = 3,  
 upper = 4,  
 a = val1,  
 b = val2  
 )$value)  
   
 # X > 2  
   
 b <- (integrate(  
 F\_Gamma\_sistem,  
 lower = 2,  
 upper = Inf,  
 a = val1,  
 b = val2  
 )$value)  
 return(a / b)  
 }  
   
 return(fprobgamma3(val1, val2))  
}

fprob3\_sistem(2, 5)

## [1] 0.07385223

fprob3\_sistem(5, 7)

## [1] 0.0002319724

fprob3\_sistem(1, 2)

## [1] 0.2386512

fprob4\_sistem <- function(val1, val2) {  
 fprobbeta4 <- function(val1, val2)  
 {  
   
 # Y > 2  
   
 return(integrate(  
 F\_Beta\_sistem,  
 lower = 2,  
 upper = Inf,  
 a = val1,  
 b = val2  
 )$value)  
 }  
   
 return(fprobbeta4(val1, val2))  
}

fprob4\_sistem(2, 5)

## [1] 0

fprob4\_sistem(5, 7)

## [1] 0

fprob4\_sistem(1, 2)

## [1] 0

fprob5\_sistem <- function(val1, val2) {  
 fprobgamma5 <- function(val1, val2)  
 {  
   
 # 4 < X < 6  
   
 return((  
 integrate(  
 F\_Gamma\_sistem,  
 lower = 4,  
 upper = 6,  
 a = val1,  
 b = val2  
 )  
 )$value)  
 }  
   
 return(fprobgamma5(val1, val2))  
}

fprob5\_sistem(2, 5)

## [1] 0.1461649

fprob5\_sistem(5, 7)

## [1] 0.001585505

fprob5\_sistem(1, 2)

## [1] 0.08554821

fprob6\_sistem <- function(val1, val2) {  
 fprobgamma6 <- function(val1, val2)  
 {  
   
 # 0 < X < 1  
   
 a <- (integrate(  
 F\_Gamma\_sistem,  
 lower = 0,  
 upper = 1,  
 a = val1,  
 b = val2  
 )$value)  
   
 # X < 7  
   
 b <- (integrate(  
 F\_Gamma\_sistem,  
 lower = 0,  
 upper = 7,  
 a = val1,  
 b = val2  
 )$value)  
 return(a / b)  
 }  
   
 return(fprobgamma6(val1, val2))  
}

fprob6\_sistem(2, 5)

## [1] 0.04293116

fprob6\_sistem(5, 7)

## [1] 0.0001202964

fprob6\_sistem(1, 2)

## [1] 0.4057211

fprob7\_sistem <- function(val1, val2) {  
   
 # X + Y < 5  
   
 f\_suma\_gamma\_beta <- Vectorize(function(z) {  
 return(integral(function(x) {  
 F\_Gamma\_sistem((z - x), val1, val2) \* F\_Beta\_sistem(x, val1, val2)  
 }, 0, 1))  
 })  
   
 return(integrate(f\_suma\_gamma\_beta, 0, 5)$value)  
}

fprob7\_sistem(2, 5)

## [1] 0.243244

fprob7\_sistem(5, 7)

## [1] 0.0005875593

fprob7\_sistem(1, 2)

## [1] 0.9023377

fprob8\_sistem <- function(val1, val2) {  
   
 # X - Y > 0.5  
   
 f\_dif\_gamma\_beta <- Vectorize(function(z) {  
 return(integral(function(x) {  
 F\_Gamma\_sistem((z + x), val1, val2) \* F\_Beta\_sistem(x, val1, val2)  
 }, 0, 1))  
 })  
   
 return(integrate(f\_dif\_gamma\_beta, 0.5, Inf)$value)  
   
}

fprob8\_sistem(2, 5)

## [1] 0.9885107

fprob8\_sistem(5, 7)

## [1] 0.9999996

fprob8\_sistem(1, 2)

## [1] 0.6637293

fprob9\_sistem <- function(val1, val2) {  
   
 # x + Y > 3  
   
 f\_1 <- Vectorize(function(z) {  
 return(integral(function(x) {  
 F\_Gamma\_sistem((z - x), val1, val2) \* F\_Beta\_sistem(x, val1, val2)  
 }, 0, 1))  
 })  
   
 # x - Y > 3  
   
 f\_2 <- Vectorize(function(z) {  
 return(integral(function(x) {  
 F\_Gamma\_sistem((z + x), val1, val2) \* F\_Beta\_sistem(x, val1, val2)  
 }, 0, 1))  
 })  
   
 return(integral(f\_2, 0.5, Inf) \* integral(f\_1, 3, Inf))  
}

fprob9\_sistem(2, 5)

## For infinite domains Gauss integration is applied!  
## For infinite domains Gauss integration is applied!

## [1] 0.8860948

fprob9\_sistem(5, 7)

## For infinite domains Gauss integration is applied!  
## For infinite domains Gauss integration is applied!

## [1] 0.9999566

fprob9\_sistem(1, 2)

## For infinite domains Gauss integration is applied!  
## For infinite domains Gauss integration is applied!

## [1] 0.1762026

##### Pentru a pune datele in tabel, le salvam intai in vectori

sistem1 <- c(  
 fprob1\_sistem(2, 5),  
 fprob2\_sistem(2, 5),  
 fprob3\_sistem(2, 5),  
 fprob4\_sistem(2, 5),  
 fprob5\_sistem(2, 5),  
 fprob6\_sistem(2, 5),  
 fprob7\_sistem(2, 5),  
 fprob8\_sistem(2, 5),  
 fprob9\_sistem(2, 5)  
   
)

## For infinite domains Gauss integration is applied!  
## For infinite domains Gauss integration is applied!

implem1 <- c(  
 fprob1(2, 5),  
 fprob2(2, 5),  
 fprob3(2, 5),  
 fprob4(2, 5),  
 fprob5(2, 5),  
 fprob6(2, 5),  
 fprob7(2, 5),  
 fprob8(2, 5),  
 fprob9(2, 5)  
)

## For infinite domains Gauss integration is applied!  
## For infinite domains Gauss integration is applied!

sistem2 <- c(  
 fprob1\_sistem(5, 7),  
 fprob2\_sistem(5, 7),  
 fprob3\_sistem(5, 7),  
 fprob4\_sistem(5, 7),  
 fprob5\_sistem(5, 7),  
 fprob6\_sistem(5, 7),  
 fprob7\_sistem(5, 7),  
 fprob8\_sistem(5, 7),  
 fprob9\_sistem(5, 7)  
   
)

## For infinite domains Gauss integration is applied!  
## For infinite domains Gauss integration is applied!

implem2 <- c(  
 fprob1(5, 7),  
 fprob2(5, 7),  
 fprob3(5, 7),  
 fprob4(5, 7),  
 fprob5(5, 7),  
 fprob6(5, 7),  
 fprob7(5, 7),  
 fprob8(5, 7),  
 fprob9(5, 7)  
)

## For infinite domains Gauss integration is applied!  
## For infinite domains Gauss integration is applied!

sistem3 <- c(  
 fprob1\_sistem(1, 2),  
 fprob2\_sistem(1, 2),  
 fprob3\_sistem(1, 2),  
 fprob4\_sistem(1, 2),  
 fprob5\_sistem(1, 2),  
 fprob6\_sistem(1, 2),  
 fprob7\_sistem(1, 2),  
 fprob8\_sistem(1, 2),  
 fprob9\_sistem(1, 2)  
)

## For infinite domains Gauss integration is applied!  
## For infinite domains Gauss integration is applied!

implem3 <- c(  
 fprob1(1, 2),  
 fprob2(1, 2),  
 fprob3(1, 2),  
 fprob4(1, 2),  
 fprob5(1, 2),  
 fprob6(1, 2),  
 fprob7(1, 2),  
 fprob8(1, 2),  
 fprob9(1, 2)  
)

## For infinite domains Gauss integration is applied!  
## For infinite domains Gauss integration is applied!

#### Valorile pentru a si b pastrate in tabel

a\_vector1 <- c()  
a\_vector1[1:9] <- 2  
b\_vector1 <- c()  
b\_vector1[1:9] <- 9  
a\_vector2 <- c()  
a\_vector2[1:9] <- 5  
b\_vector2 <- c()  
b\_vector2[1:9] <- 7  
a\_vector3 <- c()  
a\_vector3[1:9] <- 1  
b\_vector3 <- c()  
b\_vector3[1:9] <- 2

#### Crearea efectiva a tabelului

DT = data.table(  
 a = c(a\_vector1, a\_vector2, a\_vector3),  
 b = c(b\_vector1, b\_vector2, b\_vector3),  
   
 My\_Values = c(implem1, implem2, implem3),  
 System\_Values = c(sistem1, sistem2, sistem3)  
)  
DT

## a b My\_Values System\_Values  
## 1: 2 9 1.219014e-01 1.219014e-01  
## 2: 2 9 2.026892e-01 2.026892e-01  
## 3: 2 9 7.385223e-02 7.385223e-02  
## 4: 2 9 0.000000e+00 0.000000e+00  
## 5: 2 9 1.461649e-01 1.461649e-01  
## 6: 2 9 4.293116e-02 4.293116e-02  
## 7: 2 9 2.409693e-01 2.432440e-01  
## 8: 2 9 9.885107e-01 9.885107e-01  
## 9: 2 9 8.860948e-01 8.860948e-01  
## 10: 5 7 8.445787e-05 8.445787e-05  
## 11: 5 7 8.464146e-04 8.464146e-04  
## 12: 5 7 2.319724e-04 2.319724e-04  
## 13: 5 7 0.000000e+00 0.000000e+00  
## 14: 5 7 1.585505e-03 1.585505e-03  
## 15: 5 7 1.202964e-04 1.202964e-04  
## 16: 5 7 5.875746e-04 5.875593e-04  
## 17: 5 7 9.999996e-01 9.999996e-01  
## 18: 5 7 9.999566e-01 9.999566e-01  
## 19: 1 2 7.768698e-01 7.768698e-01  
## 20: 1 2 2.857944e-01 2.857944e-01  
## 21: 1 2 2.386512e-01 2.386512e-01  
## 22: 1 2 0.000000e+00 0.000000e+00  
## 23: 1 2 8.554821e-02 8.554821e-02  
## 24: 1 2 4.057211e-01 4.057211e-01  
## 25: 1 2 1.092108e+00 9.023377e-01  
## 26: 1 2 6.637293e-01 6.637293e-01  
## 27: 1 2 1.762026e-01 1.762026e-01  
## a b My\_Values System\_Values

## 3. Problema 3

### a) Construirea functiei Gamma

##### Parametri

* **n** = numarul de … pentru variabila X
* **m** = numarul de … pentru variabila Y

##### Returneaza

* o matrice de dimensiuni (n + 2) x (m + 2), reprezentand repartitia comuna pentru X si Y (generate aleator), cu goluri pe diagonala principala

frepcomgen <- function(n, m)  
{  
 # se genereaza doua variabile aleatoare discrete x si y   
 # si repartitiile lor   
 x <- sample(-2:10, n)  
 x\_norm = x / sum(x)  
 y <- sample(-2:10, m)  
 y\_norm = y / sum(y)  
   
 A <- matrix(nrow = n + 2, ncol = m + 2)  
   
 #pe prima linie se gasesc valorile lui x  
 for (i in 2:(m + 1))  
 A[1, i] <- y[i - 1]  
 #pe prima coloana se gaasesc valorile lui y  
 for (j in 2:(n + 1))  
 A[j, 1] <- x[j - 1]  
   
 #generam tabelul cu repartitia comuna a v.a. x si y   
 #intr-o forma in care sa poata sa fie calculate  
   
 for (i in 2:(n + 1))  
 {  
 suma\_l <- 0  
 for (j in 2:(m + 1))  
 {  
 suma\_l <- suma\_l + x\_norm[i - 1] \* y\_norm[j - 1]  
 if (i != j)  
 {  
 A[i, j] <- x\_norm[i - 1] \* y\_norm[j - 1]  
 }  
 else  
 #lasam goluri pe positiile in care i este egal cu j  
 A[i, j] <- 0  
   
 }  
 # pe ultima coloana se gasesc probabilitatile valorilor din x  
 A[i, m + 2] <- suma\_l  
 }  
   
   
 for (j in 2:(m + 1))  
 {  
 suma\_c <- 0  
 for (i in 2:(n + 1))  
 {  
 suma\_c <- suma\_c + x\_norm[i - 1] \* y\_norm[j - 1]  
 }  
 # pe ultima linie se gasesc probabilitatile valorilor din y  
 A[n + 2, j] <- suma\_c  
 }  
   
 suma <- 0  
 for (i in 2:(n + 1))  
 suma <- suma + A[i, m + 2]  
 A[n + 2, m + 2] <- suma  
 return(A)  
}

A <- frepcomgen(5, 4)  
A

## [,1] [,2] [,3] [,4] [,5] [,6]  
## [1,] NA 0 1.00000000 -2.00000000 7.00000000 NA  
## [2,] 6 0 0.03846154 -0.07692308 0.26923077 0.23076923  
## [3,] 1 0 0.00000000 -0.01282051 0.04487179 0.03846154  
## [4,] 5 0 0.03205128 0.00000000 0.22435897 0.19230769  
## [5,] 4 0 0.02564103 -0.05128205 0.00000000 0.15384615  
## [6,] 10 0 0.06410256 -0.12820513 0.44871795 0.38461538  
## [7,] NA 0 0.16666667 -0.33333333 1.16666667 1.00000000

### b) Umplerea de “goluri” in matrice

##### Parametri

* **A** = o matrice reprezentand repartitia comuna, avand anumite valori lipsa

##### Returneaza

* matricea, avand golurile umplute cu valori corespunzatoare

fcomplrepcom <- function(A) {  
 n <- nrow(A)  
 m <- ncol(A)  
 min <- min(n,m)  
 # se calculeaza suma pe linii si se scade din probabilitatea  
 # totala a liniei  
 for (i in 2:(m - 1))  
 {  
 suma\_l <- 0  
 for (j in 2:(min - 1))  
 {  
 if (i != j) {  
 suma\_l <- suma\_l + A[i, j]  
 }  
 }  
   
 A[i, i] <- A[i, ncol(A)] - suma\_l  
 }  
   
 return(A)  
}

A\_complet <- fcomplrepcom(A)  
A\_complet

## [,1] [,2] [,3] [,4] [,5] [,6]  
## [1,] NA 0 1.000000000 -2.00000000 7.00000000 NA  
## [2,] 6 0 0.038461538 -0.07692308 0.26923077 0.23076923  
## [3,] 1 0 0.006410256 -0.01282051 0.04487179 0.03846154  
## [4,] 5 0 0.032051282 -0.06410256 0.22435897 0.19230769  
## [5,] 4 0 0.025641026 -0.05128205 0.17948718 0.15384615  
## [6,] 10 0 0.064102564 -0.12820513 0.44871795 0.38461538  
## [7,] NA 0 0.166666667 -0.33333333 1.16666667 1.00000000

### c) Efectuarea unor calcule

### Calculul covariantei: Cov(3X, 4Y)

##### Parametri

* **A** = o matrice reprezentand repartitia comuna

##### Returneaza

* covarianta pentru 3X si 4Y, rezultata din matrice

covariance <- function(A) {  
 sum <- 0  
 # calculam E[xy]  
 for (i in 2:(nrow(A) - 1)) {  
 for (j in 2:(ncol(A) - 1)) {  
 sum <- sum + A[i, j] \* A[1, j] \* A[i, 1]  
 }  
 }  
 # calculam E[y]  
 suma\_y <- 0  
 for (j in 2:(ncol(A) - 1)) {  
 suma\_y <- suma\_y + A[1, j] \* A[(nrow(A)), j]  
   
 }  
 # calculam E[x]  
 suma\_x <- 0  
 for (i in 2:(nrow(A) - 1)) {  
 suma\_x <- suma\_x + A[i, 1] \* A[i, (ncol(A))]  
   
 }  
 # calculam cov(3x,4y)  
 cov <- 12 \* (sum - (suma\_x \* suma\_y))  
   
 return(cov)  
}

c <- covariance(A\_complet)  
c

## [1] -8.526513e-14

### Calculul probabilitatii P(0 < X < 5|Y > 4)

##### Parametri

* **A** = o matrice reprezentand repartitia comuna

##### Returneaza

* rezultatul probabilitatii P(0 < X < 5|Y > 4), rezultand din matrice

prob1 <- function(A)  
{  
 # calculam P(0 < X < 5 | Y > 4)  
 suma1 <- 0  
 suma2 <- 0  
 for (i in 2:(nrow(A) - 1)) {  
 for (j in 2:(ncol(A) - 1)) {  
 if (A[i, 1] > 0 && A[i, 1] < 5) {  
 suma1 <- suma1 + A[i, j]  
 }  
 if (A[1, j] > 4) {  
 suma2 <- suma2 + A[i, j]  
 }  
 }  
 }  
 return(suma1 / suma2)  
}

pb1 <- prob1(A\_complet)  
pb1

## [1] 0.1648352

### Calculul probabilitatii P(X > 3, Y < 7)

##### Parametri

* **A** = o matrice reprezentand repartitia comuna

##### Returneaza

* rezultatul probabilitatii P(X > 3, Y < 7), rezultand din matrice

prob2 <- function(A)  
{  
 # calculam P(X > 3, Y < 7)  
 suma <- 0  
 for (i in 2:(nrow(A) - 1)) {  
 for (j in 2:(ncol(A) - 1)) {  
 if (A[i, 1] > 3 && A[1, j] < 7) {  
 suma <- suma + A[i, j]  
 }  
 }  
 }  
 return(suma)  
}

pb2 <- prob2(A\_complet)  
pb2

## [1] -0.1602564

### d) Verificarea independentei si a corelatiei

### Verificarea independentei

##### Parametri

* **A** = o matrice reprezentand repartitia comuna

##### Returneaza

* un mesaj specificand daca variabilele X si Y sunt independente sau nu

fverind <- function(A)  
{  
 ok <- 0  
   
 for (i in 2:(nrow(A) - 1)) {  
 for (j in 2:(ncol(A) - 1)) {  
 if (A[i, j] == A[i, ncol(A)] \* A[nrow(A), j])  
 ok <- 1  
 else  
 ok <- 0  
   
 }  
 }  
 if (ok == 1)  
 {  
 return("sunt independente")  
 }  
 else  
 {  
 return("nu sunt independente")  
 }  
}

fverind(A)

## [1] "nu sunt independente"

### Verificarea corelatiei

##### Parametri

* **A** = o matrice reprezentand repartitia comuna

##### Returneaza

* un mesaj specificand daca variabilele X si Y sunt corelate sau nu

fvernecor <- function(A)  
{  
   
 # calculam E[xy]  
   
 sum <- 0  
 for (i in 2:(nrow(A) - 1)) {  
 for (j in 2:(ncol(A) - 1)) {  
 sum <- sum + A[i, j] \* A[1, j] \* A[i, 1]  
 }  
 }  
   
 # E[x], E[y]  
   
 suma\_y <- 0  
 for (j in 2:(ncol(A) - 1)) {  
 suma\_y <- suma\_y + A[1, j] \* A[(nrow(A)), j]  
   
   
 }  
 suma\_x <- 0  
 for (i in 2:(nrow(A) - 1)) {  
 suma\_x <- suma\_x + A[i, 1] \* A[i, (ncol(A))]  
 }  
 # folosim formula E[xy]= E[x]E[y]  
 if (sum == suma\_x \* suma\_y)  
 {  
 return("nu sunt corelate")  
 }  
   
 else  
 return("sunt corelate")  
}

fvernecor(A)

## [1] "sunt corelate"