Documentație PS

Ilicea Anca / Sava Daniel / Gherasim Mihai

2/6/2020

### Problema 1

## Exercitiul 1

Calcularea mediei si a variantei cu ajutorul functiilor din R mean(), respectiv var() Am generat repartitiile cu ajutorul urmatoarelor functii:

* rpois()
* rbinom()
* rexp()
* rnorm()

## function ()   
## {  
## pois = rpois(1000, 50)  
## print(var(pois))  
## print(mean(pois))  
## binom = rbinom(1000, 15, 0.2)  
## print(var(binom))  
## print(mean(binom))  
## exp = rexp(1000, 5)  
## print(var(exp))  
## print(mean(exp))  
## norm = rnorm(1000, 20)  
## print(var(norm))  
## print(mean(norm))  
## }

Rezultate:

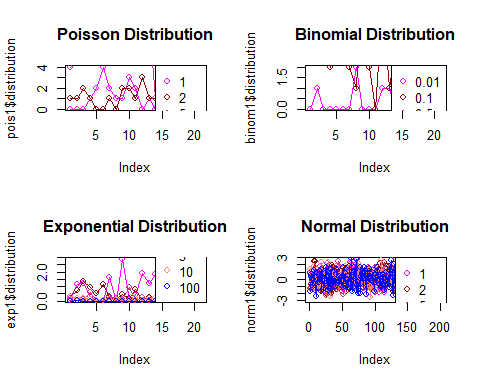
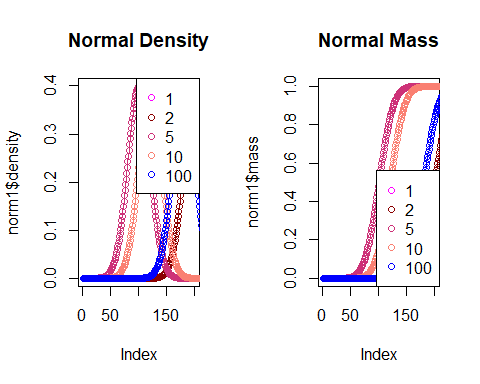
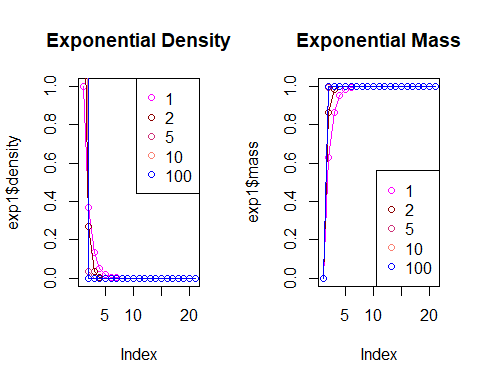
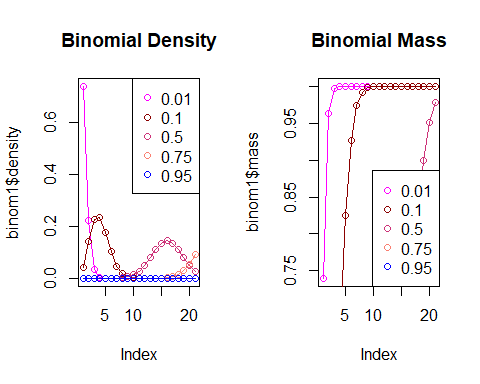
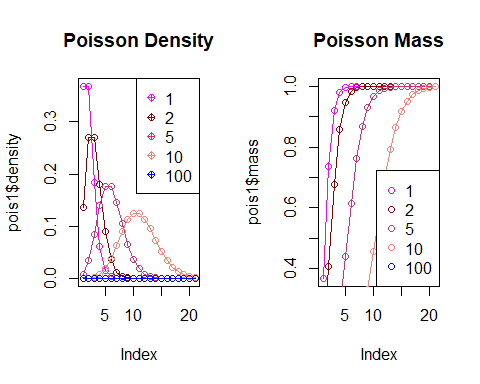
## [1] 48.21619  
## [1] 49.905  
## [1] 2.292644  
## [1] 3.093  
## [1] 0.04382053  
## [1] 0.2056995  
## [1] 0.9976532  
## [1] 19.91671

## Exercitiul 2

Graficele pentru functiile de densitate si functiile de masa

## Exercitiul 3

Graficele pentru functiile de repartitie pentru fiecare repartitie



### Exercitiul 4

## Implementare aproximari

## function (n, p)   
## {  
## aproximarePoisson <- function(k, lambda) {  
## suma <- 0  
## for (x in 0:k) suma <- suma + exp(-lambda) \* (lambda^x)/factorial(x)  
## return(suma)  
## }  
## aproximareNormalaTLC <- function(k, n, p) {  
## return(pnorm((k - n \* p)/sqrt(n \* p \* (1 - p)), mean = 0,   
## sd = 1))  
## }  
## aproximareNormalaFactorCorectie <- function(k, n, p) {  
## return(pnorm((k - 0.5 - n \* p)/sqrt(n \* p \* (1 - p)),   
## mean = 0, sd = 1))  
## }  
## aproximareCampPaulson <- function(c, miu, sigma) {  
## return(pnorm((c - miu)/sigma, mean = 0, sd = 1))  
## }  
## raspunsuri <- matrix(ncol = 6, nrow = 10)  
## for (k in 1:10) {  
## a <- 1/(9 \* (n - k))  
## b <- 1/(9 \* (k + 1))  
## r <- ((k + 1) \* (1 - p))/(p \* (n - k))  
## sigmaPatrat <- a + b \* r^(2/3)  
## c <- (1 - b) \* r^(1/3)  
## miu <- 1 - a  
## lambda <- n \* p  
## aprox\_a <- aproximarePoisson(k, lambda)  
## aprox\_b <- aproximareNormalaTLC(k, n, p)  
## aprox\_c <- aproximareNormalaFactorCorectie(k, n, p)  
## aprox\_d <- aproximareCampPaulson(c, miu, sqrt(sigmaPatrat))  
## raspunsuri[k, 1] <- k  
## raspunsuri[k, 2] <- dbinom(k, n, p)  
## raspunsuri[k, 3] <- aprox\_a  
## raspunsuri[k, 4] <- aprox\_b  
## raspunsuri[k, 5] <- aprox\_c  
## raspunsuri[k, 6] <- aprox\_d  
## }  
## return(raspunsuri)  
## }  
## <bytecode: 0x0000000018331e30>

### Exercitiul 5

Tabele:

n = 25, p = 0.05

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| k | Binomiala | Poisson | Normala | Normala Corectie | Camp-Paulson |
| 1 | 0.3649863 | 0.6446358 | 0.4092729 | 0.2456486 | 0.6452200 |
| 2 | 0.2305177 | 0.8684677 | 0.7543514 | 0.5907271 | 0.8733858 |
| 3 | 0.0930159 | 0.9617309 | 0.9458532 | 0.8743254 | 0.9651268 |
| 4 | 0.0269257 | 0.9908757 | 0.9941916 | 0.9805263 | 0.9922684 |
| 5 | 0.0059520 | 0.9981619 | 0.9997105 | 0.9985700 | 0.9985791 |
| 6 | 0.0010442 | 0.9996799 | 0.9999935 | 0.9999519 | 0.9997790 |
| 7 | 0.0001492 | 0.9999509 | 0.9999999 | 0.9999993 | 0.9999705 |
| 8 | 0.0000177 | 0.9999933 | 1.0000000 | 1.0000000 | 0.9999966 |
| 9 | 0.0000018 | 0.9999992 | 1.0000000 | 1.0000000 | 0.9999997 |
| 10 | 0.0000001 | 0.9999999 | 1.0000000 | 1.0000000 | 1.0000000 |

n = 25, p = 0.1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| k | Binomiala | Poisson | Normala | Normala Corectie | Camp-Paulson |
| 1 | 0.1994161 | 0.2872975 | 0.1586553 | 0.0912112 | 0.2706080 |
| 2 | 0.2658881 | 0.5438131 | 0.3694413 | 0.2524925 | 0.5383546 |
| 3 | 0.2264973 | 0.7575761 | 0.6305587 | 0.5000000 | 0.7647348 |
| 4 | 0.1384150 | 0.8911780 | 0.8413447 | 0.7475075 | 0.9021393 |
| 5 | 0.0645937 | 0.9579790 | 0.9522096 | 0.9087888 | 0.9662328 |
| 6 | 0.0239236 | 0.9858127 | 0.9901847 | 0.9772499 | 0.9901932 |
| 7 | 0.0072150 | 0.9957533 | 0.9986501 | 0.9961696 | 0.9975741 |
| 8 | 0.0018038 | 0.9988597 | 0.9998771 | 0.9995709 | 0.9994841 |
| 9 | 0.0003786 | 0.9997226 | 0.9999927 | 0.9999683 | 0.9999050 |
| 10 | 0.0000673 | 0.9999384 | 0.9999997 | 0.9999985 | 0.9999848 |

n = 50, p = 0.05

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| k | Binomiala | Poisson | Normala | Normala Corectie | Camp-Paulson |
| 1 | 0.2024868 | 0.2872975 | 0.1651950 | 0.0971830 | 0.2787915 |
| 2 | 0.2611014 | 0.5438131 | 0.3728014 | 0.2582061 | 0.5419320 |
| 3 | 0.2198748 | 0.7575761 | 0.6271986 | 0.5000000 | 0.7617348 |
| 4 | 0.1359752 | 0.8911780 | 0.8348050 | 0.7417939 | 0.8966284 |
| 5 | 0.0658406 | 0.9579790 | 0.9476213 | 0.9028170 | 0.9618574 |
| 6 | 0.0259897 | 0.9858127 | 0.9884295 | 0.9742121 | 0.9878222 |
| 7 | 0.0085981 | 0.9957533 | 0.9982498 | 0.9952779 | 0.9965871 |
| 8 | 0.0024324 | 0.9988597 | 0.9998207 | 0.9994116 | 0.9991503 |
| 9 | 0.0005974 | 0.9997226 | 0.9999877 | 0.9999506 | 0.9998103 |
| 10 | 0.0001289 | 0.9999384 | 0.9999994 | 0.9999972 | 0.9999617 |

n = 50, p = 0.1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| k | Binomiala | Poisson | Normala | Normala Corectie | Camp-Paulson |
| 1 | 0.0286321 | 0.0404277 | 0.0296732 | 0.0169474 | 0.0334704 |
| 2 | 0.0779429 | 0.1246520 | 0.0786496 | 0.0494801 | 0.1109253 |
| 3 | 0.1385651 | 0.2650259 | 0.1728893 | 0.1192964 | 0.2497303 |
| 4 | 0.1809045 | 0.4404933 | 0.3186759 | 0.2397501 | 0.4314063 |
| 5 | 0.1849246 | 0.6159607 | 0.5000000 | 0.4068319 | 0.6168541 |
| 6 | 0.1541038 | 0.7621835 | 0.6813241 | 0.5931681 | 0.7708904 |
| 7 | 0.1076281 | 0.8666283 | 0.8271107 | 0.7602499 | 0.8781270 |
| 8 | 0.0642779 | 0.9319064 | 0.9213504 | 0.8807036 | 0.9420684 |
| 9 | 0.0333293 | 0.9681719 | 0.9703268 | 0.9505199 | 0.9752641 |
| 10 | 0.0151833 | 0.9863047 | 0.9907889 | 0.9830526 | 0.9904630 |

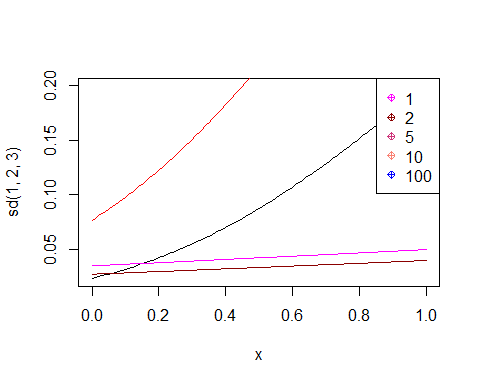
n = 100, p = 0.05

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| k | Binomiala | Poisson | Normala | Normala Corectie | Camp-Paulson |
| 1 | 0.0311607 | 0.0404277 | 0.0332287 | 0.0194737 | 0.0366911 |
| 2 | 0.0811818 | 0.1246520 | 0.0843343 | 0.0541468 | 0.1173545 |
| 3 | 0.1395757 | 0.2650259 | 0.1793977 | 0.1256746 | 0.2572236 |
| 4 | 0.1781426 | 0.4404933 | 0.3231776 | 0.2456486 | 0.4362128 |
| 5 | 0.1800178 | 0.6159607 | 0.5000000 | 0.4092729 | 0.6168067 |
| 6 | 0.1500149 | 0.7621835 | 0.6768224 | 0.5907271 | 0.7667729 |
| 7 | 0.1060255 | 0.8666283 | 0.8206023 | 0.7543514 | 0.8723891 |
| 8 | 0.0648709 | 0.9319064 | 0.9156657 | 0.8743254 | 0.9368812 |
| 9 | 0.0349013 | 0.9681719 | 0.9667713 | 0.9458532 | 0.9716068 |
| 10 | 0.0167159 | 0.9863047 | 0.9891093 | 0.9805263 | 0.9883152 |

n = 100, p = 0.1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| k | Binomiala | Poisson | Normala | Normala Corectie | Camp-Paulson |
| 1 | 0.0002951 | 0.0004994 | 0.0013499 | 0.0007710 | 0.0003769 |
| 2 | 0.0016232 | 0.0027694 | 0.0038304 | 0.0023033 | 0.0020473 |
| 3 | 0.0058916 | 0.0103361 | 0.0098153 | 0.0062097 | 0.0079372 |
| 4 | 0.0158746 | 0.0292527 | 0.0227501 | 0.0151301 | 0.0237017 |
| 5 | 0.0338658 | 0.0670860 | 0.0477904 | 0.0333765 | 0.0573670 |
| 6 | 0.0595787 | 0.1301414 | 0.0912112 | 0.0668072 | 0.1167730 |
| 7 | 0.0888952 | 0.2202206 | 0.1586553 | 0.1216725 | 0.2056576 |
| 8 | 0.1148230 | 0.3328197 | 0.2524925 | 0.2023284 | 0.3206667 |
| 9 | 0.1304163 | 0.4579297 | 0.3694413 | 0.3085375 | 0.4513689 |
| 10 | 0.1318653 | 0.5830398 | 0.5000000 | 0.4338162 | 0.5834688 |

### Exercitiul 6

Repartitiile normalei asimetrice 

### Exercitiul 7

Calculul ecuatiei:

p1ex7

## function (n, p)   
## {  
## ecuatie <- function(n, p) {  
## return(Vectorize(function(lambda) {  
## argUp1 <- (1 - ((2/pi) \* ((lambda^2)/(1 + lambda^2))))^3  
## argDown1 <- (2/pi) \* (4/pi - 1)^2 \* ((lambda^2)/(1 +   
## lambda^2))^3  
## argUp2 <- n \* p \* (1 - p)  
## argDown2 <- (1 - 2 \* p)^2  
## return(argUp1/argDown1 - argUp2/argDown2)  
## }))  
## }  
## functie <- ecuatie(n, p)  
## solution <- uniroot(f = functie, interval = c(0, 1000))  
## solutie <- solution$root  
## print(solutie)  
## lambda <- sign(1 - (2 \* p)) \* sqrt(solutie)  
## print(lambda)  
## sigmaPatrat <- n \* p \* (1 - p)/(1 - ((2/pi) \* (solutie/(1 +   
## solutie))))  
## print(sigmaPatrat)  
## sigma <- sqrt(sigmaPatrat)  
## miu <- n \* p - sigma \* sqrt((2/pi) \* (solutie/(1 + solutie)))  
## print(miu)  
## result <- c(lambda, sigma, miu)  
## library(sn)  
## barplot(pbinom(q = 0:n, size = n, prob = 0.05), col = "salmon")  
## barplot(pbinom(q = 0:n, size = n, prob = 0.1), col = "green",   
## add = TRUE)  
## barplot(dsn(x = 0:n, dp = c(miu, sigma, lambda)), col = "dark red",   
## add = TRUE)  
## return(result)  
## }

Graficul:

p1ex7(25, 0.05)

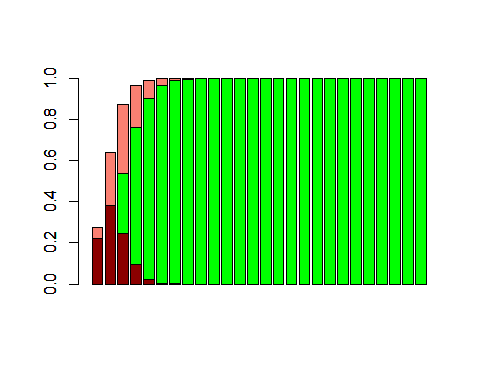
## [1] 4.558586  
## [1] 2.135085  
## [1] 2.484781  
## [1] 0.1110175

## Warning: package 'sn' was built under R version 3.6.2

## Loading required package: stats4

##   
## Attaching package: 'sn'

## The following object is masked from 'package:stats':  
##   
## sd



## [1] 2.1350846 1.5763188 0.1110175

### Exercitiul 8

Calculul integralei si construirea tabelului

p1ex8

## function (n, p)   
## {  
## vector <- p1ex7(25, 0.05)  
## print(vector)  
## lambda <- vector[1]  
## sigma <- vector[2]  
## miu <- vector[3]  
## raspunsuri <- matrix(ncol = 3, nrow = 10)  
## aproximareNormalaSimetrica <- function(k, n, p) {  
## return(Vectorize(function(x) {  
## D <- function(t) {  
## 2 \* dnorm(t) \* pnorm(t \* lambda)  
## }  
## return(integral(D, -Inf, x))  
## }))  
## for (k in 1:10) {  
## aproxNS <- aproximareNormalaSimetrica(k, n, p)  
## raspunsuri[k, 1] <- k  
## raspunsuri[k, 2] <- dbinom(k, n, p)  
## raspunsuri[k, 3] <- aproxNS(k, n, p)(x)  
## }  
## return(raspunsuri)  
## }  
## }

## Problema 2

# a) Implementarea integralei Gamma

fgam

## function (a)   
## {  
## if (a == 1) {  
## 1  
## }  
## else if (a%%1 == 0) {  
## factorial(a - 1)  
## }  
## else if (a == 1/2) {  
## sqrt(pi)  
## }  
## else if (a > 1) {  
## (a - 1) \* fgam(a - 1)  
## }  
## else {  
## integrate(function(x, a) {  
## x^(a - 1) \* exp(-x)  
## }, 0, Inf, a)$value  
## }  
## }

# b) Implementarea integralei Beta

fbet

## function (a, b)   
## {  
## if (a > 0 && b > 0 && a + b == 1) {  
## pi/sin(a \* pi)  
## }  
## fgam(a) \* fgam(b)/fgam(a + b)  
## }

# c) Calcul de probabilități

Pentru a calcula probabilitățile, trebuie să calculăm funcțiile de densitate:

* pentru distribuția Gamma

gamma\_function

## function (a, b)   
## {  
## function(x) {  
## if (x > 0 && a > 0 && b > 0) {  
## 1/(b^a \* fgam(a)) \* x^(a - 1) \* exp(-x/b)  
## }  
## else {  
## 0  
## }  
## }  
## }

* pentru distribuția Beta

beta\_function

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

**Rezultate:**

1. P(X < 3)

## [1] 0.1911532

## [1] 0.2032366

## [1] 0.9999607

1. P(2 < X < 5)

## [1] 0.3758855

## [1] 0.2968306

## [1] 0.002769391

1. P(3 < X < 4 | X > 2)

## [1] 0.1437106

## [1] 0.1140614

## [1] 0.01402939

1. P(Y > 2)

## [1] 0

## [1] 0

## [1] 0

1. P(4 < Y < 6)

## [1] 0

## [1] 0

## [1] 0

1. P(0 < Y < 1 | Y < 7)

## [1] 0.9999999

## [1] 0.9999999

## [1] 1

1. P(X + Y < 5)

## [1] 0.3774311

## [1] 0.3673825

## [1] 0.9971515

1. P(X - Y > 0.5)

## [1] 0.9803959

## [1] 0.9749825

## [1] 0.02947578

1. P(X + Y > 3 | X - Y > 0.5)

## [1] 0.8963464

## [1] 0.8534223

## [1] 0.07824592

# d) Comparare cu funcții de sistem

a = 3, b = 2

|  |  |  |
| --- | --- | --- |
| Nr. | User | System |
| 1 | 0.191153169461942 | 0.191153169461942 |
| 2 | 0.375885487045276 | 0.375885487045276 |
| 3 | 0.143710574237327 | 0.143710574240434 |
| 4 | 0 | 0 |
| 5 | 0 | 0 |
| 6 | 0.999999855797116 | 1 |
| 7 | 0.377431088268533 | 0.37716111940715 |
| 8 | 0.980395878624417 | 0.997865820908179 |
| 9 | 0.896346351347886 | 0.88065374148371 |

Se observă unele diferențe între probabilitățile calculate manual și cele în care au fost folosite funcții din R. Acest lucru este cauzat de erori de aproximare atunci când se fac calcule cu mai multe zecimale. Totuși, aceste diferențe sunt neglijabile.

## Problema 3

1. Acest subpunct a presupus în mare parte algoritmică. Pentru a avea termen de comparație între funcții am creat un set de funcții care crează o repartiție între doua variabile aleatoare X și Y care sunt independente și un set de funcții pentru X și Y luate total random.

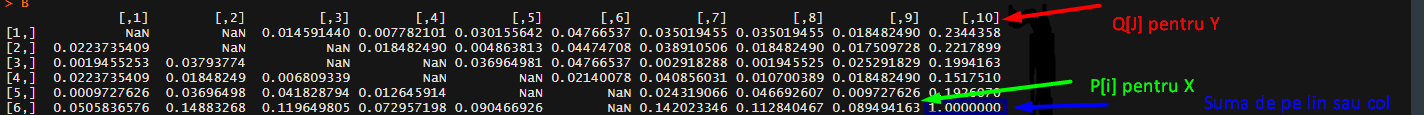
Vom considera ca repartitia este de forma unei matrice A cu n linii si m coloane indexate de la 1.

1)Pentru X, Y independente:

I)Pentru setul X, Y independente am generat P[i] (pentru X) si Q[j] (pentru Y) pe marginile din jos (linia n + 1), respectiv dreapta (coloana m + 1).

II)In functie de aceste doua generari am creat fiecare element din matricea interioara A dupa formula A[i][j] = A[n + 1][j] \* A[i][m + 1].

III)Am sters elementele in forma aceasta: (NaN = sters)



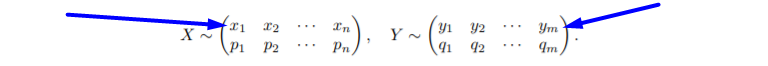
2)Pentru X, Y random:

I) Am generat pentru fiecare element din matricea interioara un numar intre [0, 50].

II) Am facut suma lor pe linii si pe coloane si le-am notat in locul lui P[i] pentru coloana i si Q[j] pentru linia j.

III) Am calculat suma totala din matricea interioara si am impartit fiecare numar din interior (A[][]) si exterior (P si Q) la suma totala, in afara de elementele care vor ramane NaN sub forma de la punctul anterior.

3)Pentru X, Y random SI pentru X, Y independente trebuie sa generam valorile X[i] si Y[i] pentru punctele viitoare (c si d, ne trebuie acolo). Aceste valori generate vor fi pe linia n + 2 pentru X si pe coloana m + 2 pentru Y.



b)Ambele tipuri de repartitii, independente sau random folosesc acelasi algoritm.

1)Incepem din stanga sus si putem calcula aceasta valoare deoarece stim toate valorile de pe COLOANA EI.

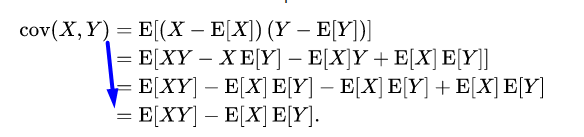
2)Mergem in dreapta o casuta si o putem calcula deoarece ACUM stim toate valorile de pe LINIA ei. Daca patratica nu mai e NaN oprim loop-ul.

3)Coboram o patratica si daca e egal cu NaN sari la b1), altfel e gata algoritmul.

c)Aici am avut doar de aplicat teoremele din clasa:

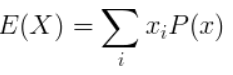
1)

I)Folosim formula:



II)Am creat functii separate pentru E[X], E[Y], E[X,Y] care vor lua aceste X si Y DIRECT din matrice.

Folosim formula de mai jos pentru a calcula E[X]:



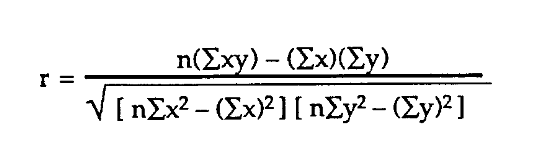
2) Trebuie doar sa parcurgem zona din matrice pentru care 0 < X[i] < 5 si 4 < Y[j].

3) Trebuie doar sa parcurgem zona din matrice pentru care 3 < X[i] si Y[j] < 7.

d)Din moment ce am creat o repartitie independenta si una cel mai probabil dependenta, putem testa cu succes aceste functii.

1) Trebuie doar sa verificam conditia de independenta apelata la punctul a) 1) II).

2)Aplicam formula:

I)

II) Numaratorul este calculat de la punctul c) 1) Pentru coeficientii 1 amandoi. Nu 3 si 4.

III) Vom crea o functie separata pentru numitor, una care calculeaza E[x^2].

IV) Unim toate calculele care trebuie facute prin apelarea functiilor de E[x] si cov(X,Y) de la punctul c) 1) si prin calcule elementare.

DISCLAIMER COD PROBLEMA 3:

Am incercat sa fac o documentatie in Doxygen, dar nu a mers cum eram obisnuit in C++ (de asta vor fi multe @ in cod). Am lasat comentariile pentru Doxygen in cod pentru a putea fi inteles mai usor.