Proiect Final PS

AB Mihaela, DLarisa ~ Grupa 241

2020-02-07

# Problema 1  
  
## Exercitiul 1  
  
### Repartitia Binomiala Bin(n, p)  
  
bin <- rbinom(10, 1000, 0.1)  
mediaBin = mean(bin)  
variantaBin = var(bin)  
print ("Media: ")

## [1] "Media: "

print(mediaBin)

## [1] 100.8

print("Varianta: ")

## [1] "Varianta: "

print(variantaBin)

## [1] 106.1778

### Repartitia Poisson Pois(lambda)  
  
pois = rpois(1000, sqrt(2))  
mediaPois = mean(pois)  
variantaPois = var(pois)  
print ("Media: ")

## [1] "Media: "

print(mediaPois)

## [1] 1.381

print("Varianta: ")

## [1] "Varianta: "

print(variantaPois)

## [1] 1.39123

### Repartitia Exponentiala Exp(lambda)  
  
exp = rexp(1000, rate = 2) # lambda = 2  
mediaExp = mean(exp)  
variantaExp = var(exp)  
print ("Media: ")

## [1] "Media: "

print(mediaExp)

## [1] 0.5104392

print("Varianta: ")

## [1] "Varianta: "

print(variantaExp)

## [1] 0.2577806

### Repartitia Normala   
  
norm = rnorm(1000)  
mediaNorm = mean(norm)  
varinataNorm = var(norm)  
print ("Media: ")

## [1] "Media: "

print(mediaExp)

## [1] 0.5104392

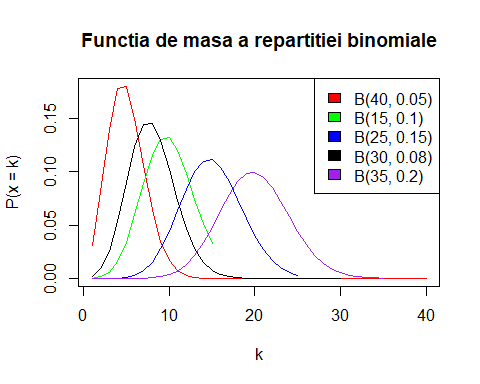
print("Varianta: ")

## [1] "Varianta: "

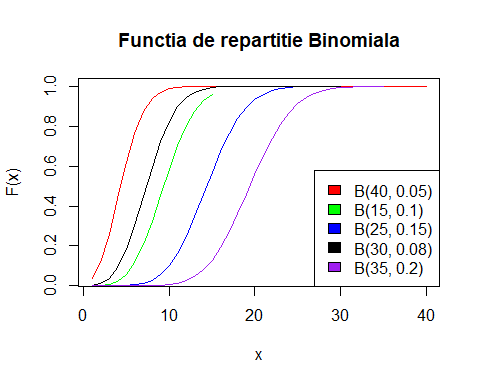
print(variantaExp)

## [1] 0.2577806

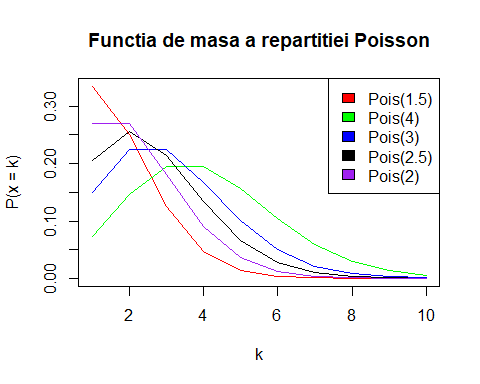
######################  
  
## Exercitiile 2 si 3  
  
  
### Repartitia Binomiala  
  
  
plot(1:40,dbinom(1:40, 100, 0.05),type="l", col="red",  
 xlab = "k",  
 ylab = "P(x = k)",   
 main = "Functia de masa a repartitiei binomiale",  
)  
lines(1:15, dbinom(1:15, 100, 0.1), col="green")  
lines(1:25, dbinom(1:25, 100, 0.15), col = "blue")  
lines(1:30, dbinom(1:30, 100, 0.08), col = "black")  
lines(1:35, dbinom(1:35, 100, 0.2), col = "purple")  
legend("topright", c("B(40, 0.05)", "B(15, 0.1)", "B(25, 0.15)", "B(30, 0.08)", "B(35, 0.2)"),  
 fill = c("red", "green", "blue", "black", "purple")  
)



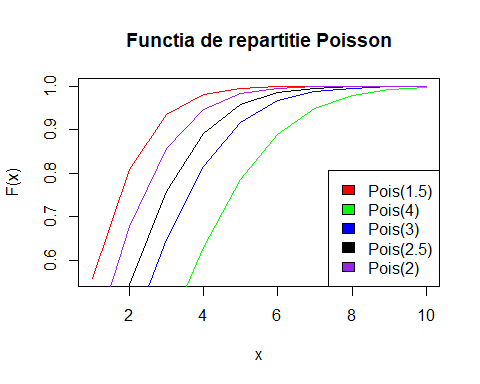
plot(1:40,pbinom(1:40, 100, 0.05),type="l", col="red",  
 xlab = "x",  
 ylab = "F(x)",   
 main = "Functia de repartitie Binomiala",  
)  
lines(1:15, pbinom(1:15, 100, 0.1), col="green")  
lines(1:25, pbinom(1:25, 100, 0.15), col = "blue")  
lines(1:30, pbinom(1:30, 100, 0.08), col = "black")  
lines(1:35, pbinom(1:35, 100, 0.2), col = "purple")  
legend("bottomright", c("B(40, 0.05)", "B(15, 0.1)", "B(25, 0.15)", "B(30, 0.08)", "B(35, 0.2)"),  
 fill = c("red", "green", "blue", "black", "purple")  
)



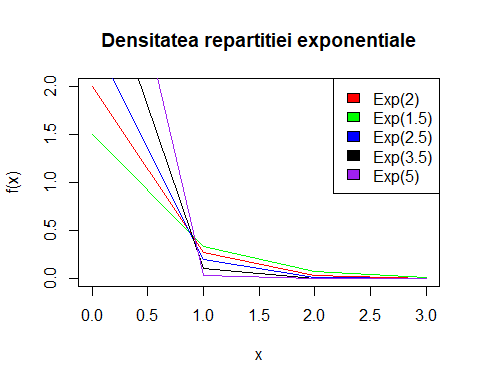
### Repartitia Poisson - Functia de masa si Functia de repartitie  
  
  
x = 1:10  
  
plot(x,dpois(x, 1.5),type="l", col="red",  
 xlab = "k",  
 ylab = "P(x = k)",   
 main = "Functia de masa a repartitiei Poisson",  
)  
lines(x, dpois(x, 4), col="green")  
lines(x, dpois(x, 3), col = "blue")  
lines(x, dpois(x, 2.5), col = "black")  
lines(x, dpois(x, 2), col = "purple")  
legend("topright", c("Pois(1.5)", "Pois(4)", "Pois(3)", "Pois(2.5)", "Pois(2)"),  
 fill = c("red", "green", "blue", "black", "purple")  
)



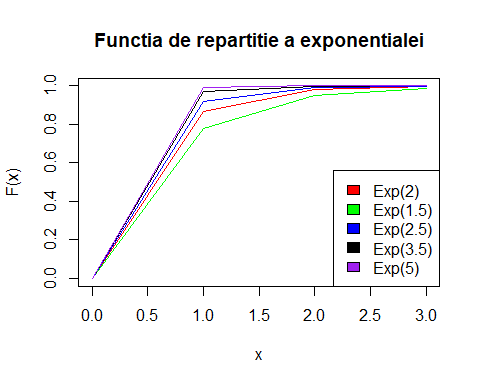
y = 1:10  
  
plot(y,ppois(y, 1.5),type="l", col="red",  
 xlab = "x",  
 ylab = "F(x)",   
 main = "Functia de repartitie Poisson",  
)  
lines(y, ppois(y, 4), col="green")  
lines(y, ppois(y, 3), col = "blue")  
lines(y, ppois(y, 2.5), col = "black")  
lines(y, ppois(y, 2), col = "purple")  
legend("bottomright", c("Pois(1.5)", "Pois(4)", "Pois(3)", "Pois(2.5)", "Pois(2)"),  
 fill = c("red", "green", "blue", "black", "purple")  
)



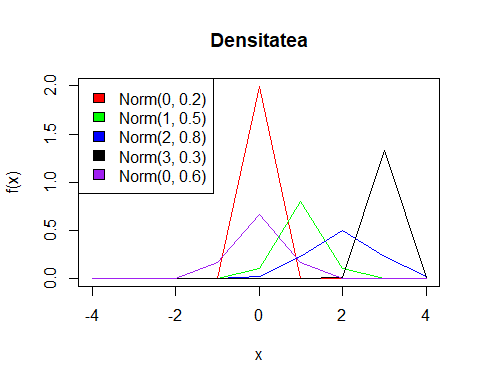
### Repartitia Exponentiala - Densitatea si Functia de repartitie  
  
x = 0:3  
  
plot(x,dexp(x, rate = 2),  
 type="l", col="red",  
 xlab = "x",  
 ylab = "f(x)",   
 main = "Densitatea repartitiei exponentiale",  
)  
lines(x, dexp(x, rate = 1.5), col="green")  
lines(x, dexp(x, rate = 2.5), col = "blue")  
lines(x, dexp(x, rate = 3.5), col = "black")  
lines(x, dexp(x, rate = 5), col = "purple")  
legend("topright", c("Exp(2)", "Exp(1.5)", "Exp(2.5)", "Exp(3.5)", "Exp(5)"),  
 fill = c("red", "green", "blue", "black", "purple")  
)



plot(x,pexp(x, rate = 2),  
 type="l", col="red",  
 xlab = "x",  
 ylab = "F(x)",   
 main = "Functia de repartitie a exponentialei",  
)  
lines(x, pexp(x, rate = 1.5), col="green")  
lines(x, pexp(x, rate = 2.5), col = "blue")  
lines(x, pexp(x, rate = 3.5), col = "black")  
lines(x, pexp(x, rate = 5), col = "purple")  
legend("bottomright", c("Exp(2)", "Exp(1.5)", "Exp(2.5)", "Exp(3.5)", "Exp(5)"),  
 fill = c("red", "green", "blue", "black", "purple")  
)



### Repartitia Normala - Densitatea si Functia de repartitie  
  
x = -4:4  
  
plot(x,dnorm(x, mean = 0, sd = 0.2),  
 type="l", col="red",  
 xlab = "x",  
 ylab = "f(x)",   
 main = "Densitatea",  
)  
lines(x, dnorm(x, mean = 1, sd = 0.5), col="green")  
lines(x, dnorm(x, mean = 2, sd = 0.8), col = "blue")  
lines(x, dnorm(x, mean = 3, sd = 0.3), col = "black")  
lines(x, dnorm(x, mean = 0, sd = 0.6), col = "purple")  
legend("topleft", c("Norm(0, 0.2)", "Norm(1, 0.5)", "Norm(2, 0.8)", "Norm(3, 0.3)", "Norm(0, 0.6)"),  
 fill = c("red", "green", "blue", "black", "purple")  
)



######################  
  
## Exercitiul 4  
  
### Functie care calculeaza aproximarile - Functia de masa  
  
aproxFunctiaDeMasa <- function(n, p, a, b) {  
 lambda = n\*p;  
 x = matrix(numeric((b - a + 1) \* 5),   
 ncol = 5,  
 dimnames = list(a:b, c("Binomiala", "Poisson", "Normala", "Normala corectie", "Camp-Paulson")))  
 y = rnorm(n, n \* p, sqrt(n \* p \* (1 - p)))  
 x[, 1] = dbinom(a:b, n, p) # binomiala  
 x[, 2] = dpois(a:b, lambda) # poisson  
 x[, 3] = dnorm(a:b - mean(y))/sd(y) # fara coeficient de corelatie  
 x[, 4] = dnorm((a:b + 0.5) - mean(y))/sd(y) # cu coeficient de corelatie  
   
 a1 = 1 / (n - a:b)  
 b1 = 1 / (a:b + 1)  
 r = ((a:b + 1)\*(1 - p)) / (p \* (n - a:b))  
 c = (1 - b1) \* (r^(1 / 3))  
 sigma = sqrt(a1 + b1 \* r ^ (2 / 3))  
 miu = 1 - a1  
   
 x[, 5] = dnorm((c - miu) / sigma) # Camp-Paulson  
   
 cat("Aproximari pentru n = ", n, " p = ", p, " - Functia de masa\n")  
 print(x)  
}  
aproxFunctiaDeMasa(25, 0.05, 1, 10)

## Aproximari pentru n = 25 p = 0.05 - Functia de masa  
## Binomiala Poisson Normala Normala corectie Camp-Paulson  
## 1 3.649863e-01 3.581310e-01 3.568232e-01 3.299658e-01 0.3617621  
## 2 2.305177e-01 2.238319e-01 2.376354e-01 1.332845e-01 0.3980424  
## 3 9.301589e-02 9.326328e-02 5.822034e-02 1.980597e-02 0.3886877  
## 4 2.692565e-02 2.914478e-02 5.247396e-03 1.082725e-03 0.3554081  
## 5 5.951987e-03 7.286194e-03 1.739877e-04 2.177436e-05 0.3109167  
## 6 1.044208e-03 1.517957e-03 2.122260e-06 1.610936e-07 0.2630516  
## 7 1.491726e-04 2.710638e-04 9.523231e-09 4.384466e-10 0.2165693  
## 8 1.766518e-05 4.235371e-05 1.572083e-11 4.389961e-13 0.1741649  
## 9 1.756187e-06 5.882460e-06 9.547116e-15 1.617000e-16 0.1371462  
## 10 1.478894e-07 7.353075e-07 2.132919e-18 2.191115e-20 0.1059094

aproxFunctiaDeMasa(50, 0.05, 1, 10)

## Aproximari pentru n = 50 p = 0.05 - Functia de masa  
## Binomiala Poisson Normala Normala corectie Camp-Paulson  
## 1 0.2024867770 0.2052124966 8.892396e-02 1.715179e-01 0.2938275  
## 2 0.2611013704 0.2565156207 2.576480e-01 3.014187e-01 0.3629452  
## 3 0.2198748382 0.2137630172 2.746250e-01 1.948661e-01 0.3936937  
## 4 0.1359752289 0.1336018858 1.076859e-01 4.634555e-02 0.3979973  
## 5 0.0658406372 0.0668009429 1.553401e-02 4.054947e-03 0.3841768  
## 6 0.0259897252 0.0278337262 8.243533e-04 1.305173e-04 0.3585171  
## 7 0.0085981046 0.0099406165 1.609344e-05 1.545454e-06 0.3258329  
## 8 0.0024323585 0.0031064427 1.155819e-07 6.732090e-09 0.2897592  
## 9 0.0005974214 0.0008629007 3.053770e-10 1.078820e-11 0.2529577  
## 10 0.0001289172 0.0002157252 2.968167e-13 6.359954e-15 0.2172928

aproxFunctiaDeMasa(100, 0.05, 1, 10)

## Aproximari pentru n = 100 p = 0.05 - Functia de masa  
## Binomiala Poisson Normala Normala corectie Camp-Paulson  
## 1 0.03116068 0.03368973 8.668275e-05 5.322616e-04 0.1949973  
## 2 0.08118177 0.08422434 2.545329e-03 9.479581e-03 0.2708711  
## 3 0.13957568 0.14037390 2.749544e-02 6.210959e-02 0.3267341  
## 4 0.17814264 0.17546737 1.092655e-01 1.497041e-01 0.3651452  
## 5 0.18001783 0.17546737 1.597389e-01 1.327438e-01 0.3882478  
## 6 0.15001486 0.14622281 8.591007e-02 4.330124e-02 0.3981749  
## 7 0.10602554 0.10444486 1.699742e-02 5.196272e-03 0.3970866  
## 8 0.06487089 0.06527804 1.237164e-03 2.293977e-04 0.3871091  
## 9 0.03490130 0.03626558 3.312662e-05 3.725561e-06 0.3702606  
## 10 0.01671588 0.01813279 3.263116e-07 2.225870e-08 0.3483903

aproxFunctiaDeMasa(25, 0.1, 1, 10)

## Aproximari pentru n = 25 p = 0.1 - Functia de masa  
## Binomiala Poisson Normala Normala corectie Camp-Paulson  
## 1 1.994161e-01 0.2052124966 8.765440e-02 1.568830e-01 0.3016464  
## 2 2.658881e-01 0.2565156207 2.186778e-01 2.373885e-01 0.3692352  
## 3 2.264973e-01 0.2137630172 2.006971e-01 1.321445e-01 0.3964076  
## 4 1.384150e-01 0.1336018858 6.776152e-02 2.706099e-02 0.3956796  
## 5 6.459368e-02 0.0668009429 8.416483e-03 2.038656e-03 0.3759679  
## 6 2.392358e-02 0.0278337262 3.845773e-04 5.650017e-05 0.3441479  
## 7 7.215049e-03 0.0099406165 6.464608e-06 5.760512e-07 0.3055657  
## 8 1.803762e-03 0.0031064427 3.997664e-08 2.160617e-09 0.2642861  
## 9 3.785674e-04 0.0008629007 9.094438e-11 2.981263e-12 0.2232754  
## 10 6.730087e-05 0.0002157252 7.611162e-14 1.513311e-15 0.1845730

aproxFunctiaDeMasa(50, 0.1, 1, 10)

## Aproximari pentru n = 50 p = 0.1 - Functia de masa  
## Binomiala Poisson Normala Normala corectie Camp-Paulson  
## 1 0.02863208 0.03368973 1.362361e-04 8.007165e-04 0.1985944  
## 2 0.07794290 0.08422434 3.665151e-03 1.306566e-02 0.2753365  
## 3 0.13856515 0.14037390 3.627414e-02 7.843127e-02 0.3313829  
## 4 0.18090450 0.17546737 1.320711e-01 1.732020e-01 0.3692136  
## 5 0.18492460 0.17546737 1.768985e-01 1.407090e-01 0.3909600  
## 6 0.15410383 0.14622281 8.716576e-02 4.205294e-02 0.3988201  
## 7 0.10762807 0.10444486 1.580059e-02 4.623560e-03 0.3950820  
## 8 0.06427788 0.06527804 1.053674e-03 1.870088e-04 0.3820432  
## 9 0.03332927 0.03626558 2.584904e-05 2.782615e-06 0.3619136  
## 10 0.01518333 0.01813279 2.332858e-07 1.523175e-08 0.3367345

aproxFunctiaDeMasa(100, 0.1, 1, 10)

## Aproximari pentru n = 100 p = 0.1 - Functia de masa  
## Binomiala Poisson Normala Normala corectie Camp-Paulson  
## 1 0.0002951267 0.0004539993 4.098298e-20 3.658911e-18 0.09230384  
## 2 0.0016231966 0.0022699965 2.544055e-16 1.377614e-14 0.14366666  
## 3 0.0058916025 0.0075666550 5.809716e-13 1.908133e-11 0.19253746  
## 4 0.0158745955 0.0189166374 4.880775e-10 9.722890e-09 0.23792430  
## 5 0.0338658038 0.0378332748 1.508441e-07 1.822584e-06 0.27880250  
## 6 0.0595787289 0.0630554580 1.715036e-05 1.256855e-04 0.31431555  
## 7 0.0888952464 0.0900792257 7.173378e-04 3.188516e-03 0.34384950  
## 8 0.1148230266 0.1125990321 1.103773e-02 2.975756e-02 0.36705103  
## 9 0.1304162771 0.1251100357 6.248004e-02 1.021672e-01 0.38381692  
## 10 0.1318653468 0.1251100357 1.301093e-01 1.290421e-01 0.39426786

### Functie care calculeaza aproximarile - Functia de repartitie  
  
aproxFunctiaDeRepartitie <- function(n, p, a, b) {  
 lambda = n\*p;  
 x = matrix(numeric((b - a + 1) \* 5),   
 ncol = 5,  
 dimnames = list(a:b, c("Binomiala", "Poisson", "Normala", "Normala corectie", "Camp-Paulson")))  
 y = rnorm(n, n \* p, sqrt(n \* p \* (1 - p)))  
 x[, 1] = pbinom(a:b, n, p) # binomiala  
 x[, 2] = ppois(a:b, lambda) # poisson  
 x[, 3] = pnorm((a:b - mean(y))/sd(y)) # fara coeficient de corelatie  
 x[, 4] = pnorm(((a:b + 0.5) - mean(y))/sd(y)) # cu coeficient de corelatie  
   
 a1 = 1 / (n - a:b)  
 b1 = 1 / (a:b + 1)  
 r = ((a:b + 1)\*(1 - p)) / (p \* (n - a:b))  
 c = (1 - b1) \* (r^(1 / 3))  
 sigma = sqrt(a1 + b1 \* r ^ (2 / 3))  
 miu = 1 - a1  
 #y = rnorm(miu, sigma)  
 x[, 5] = pnorm((c - miu) / sigma) # Camp-Paulson  
   
 cat("Aproximari pentru n = ", n, " p = ", p, " - Functia de repartitie\n")  
 print(x)  
}  
aproxFunctiaDeRepartitie(25, 0.05, 1, 10)

## Aproximari pentru n = 25 p = 0.05 - Functia de repartitie  
## Binomiala Poisson Normala Normala corectie Camp-Paulson  
## 1 0.6423759 0.6446358 0.3929767 0.5535775 0.3291236  
## 2 0.8728935 0.8684677 0.7057391 0.8282453 0.4732089  
## 3 0.9659094 0.9617309 0.9120565 0.9607793 0.5902595  
## 4 0.9928351 0.9908757 0.9848473 0.9949495 0.6846454  
## 5 0.9987870 0.9981619 0.9985523 0.9996440 0.7599384  
## 6 0.9998312 0.9996799 0.9999250 0.9999865 0.8192872  
## 7 0.9999804 0.9999509 0.9999979 0.9999997 0.8654974  
## 8 0.9999981 0.9999933 1.0000000 1.0000000 0.9010381  
## 9 0.9999998 0.9999992 1.0000000 1.0000000 0.9280401  
## 10 1.0000000 0.9999999 1.0000000 1.0000000 0.9483053

aproxFunctiaDeRepartitie(50, 0.05, 1, 10)

## Aproximari pentru n = 50 p = 0.05 - Functia de repartitie  
## Binomiala Poisson Normala Normala corectie Camp-Paulson  
## 1 0.2794318 0.2872975 0.1956542 0.2914090 0.2170840  
## 2 0.5405331 0.5438131 0.4046616 0.5265796 0.3318210  
## 3 0.7604080 0.7575761 0.6460390 0.7525772 0.4353581  
## 4 0.8963832 0.8911780 0.8390586 0.9029543 0.5274530  
## 5 0.9622238 0.9579790 0.9459228 0.9722230 0.6082039  
## 6 0.9882136 0.9858127 0.9868750 0.9943044 0.6780511  
## 7 0.9968117 0.9957533 0.9977333 0.9991736 0.7377060  
## 8 0.9992440 0.9988597 0.9997242 0.9999158 0.7880601  
## 9 0.9998414 0.9997226 0.9999765 0.9999940 0.8301004  
## 10 0.9999704 0.9999384 0.9999986 0.9999997 0.8648420

aproxFunctiaDeRepartitie(100, 0.05, 1, 10)

## Aproximari pentru n = 100 p = 0.05 - Functia de repartitie  
## Binomiala Poisson Normala Normala corectie Camp-Paulson  
## 1 0.03708121 0.04042768 0.02174022 0.03679307 0.1157467  
## 2 0.11826298 0.12465202 0.05946377 0.09185741 0.1894377  
## 3 0.25783866 0.26502592 0.13577141 0.19225149 0.2637150  
## 4 0.43598130 0.44049329 0.26117011 0.34095615 0.3369633  
## 5 0.61599913 0.61596065 0.42858890 0.51990665 0.4078332  
## 6 0.76601398 0.76218346 0.61018701 0.69486695 0.4752601  
## 7 0.87203952 0.86662833 0.77022281 0.83384413 0.5384638  
## 8 0.93691041 0.93190637 0.88480506 0.92353270 0.5969261  
## 9 0.97181171 0.96817194 0.95145515 0.97055520 0.6503586  
## 10 0.98852759 0.98630473 0.98295070 0.99058276 0.6986647

aproxFunctiaDeRepartitie(25, 0.1, 1, 10)

## Aproximari pentru n = 25 p = 0.1 - Functia de repartitie  
## Binomiala Poisson Normala Normala corectie Camp-Paulson  
## 1 0.2712059 0.2872975 0.3059783 0.4079547 0.2273070  
## 2 0.5370941 0.5438131 0.5166166 0.6240506 0.3470111  
## 3 0.7635914 0.7575761 0.7226090 0.8065033 0.4550533  
## 4 0.9020064 0.8911780 0.8727643 0.9213233 0.5509869  
## 5 0.9666001 0.9579790 0.9543425 0.9751754 0.6347351  
## 6 0.9905236 0.9858127 0.9873715 0.9939962 0.7066384  
## 7 0.9977387 0.9957533 0.9973351 0.9988965 0.7673895  
## 8 0.9995425 0.9988597 0.9995740 0.9998468 0.8179306  
## 9 0.9999210 0.9997226 0.9999487 0.9999840 0.8593523  
## 10 0.9999883 0.9999384 0.9999954 0.9999987 0.8928059

aproxFunctiaDeRepartitie(50, 0.1, 1, 10)

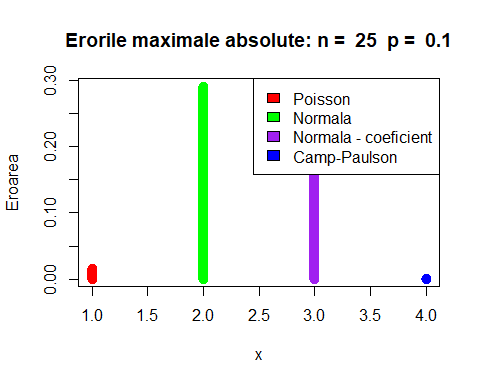
## Aproximari pentru n = 50 p = 0.1 - Functia de repartitie  
## Binomiala Poisson Normala Normala corectie Camp-Paulson  
## 1 0.03378586 0.04042768 0.07979197 0.1152457 0.1187725  
## 2 0.11172876 0.12465202 0.16066810 0.2164211 0.1945671  
## 3 0.25029391 0.26502592 0.28198350 0.3558474 0.2712068  
## 4 0.43119841 0.44049329 0.43557312 0.5180163 0.3469561  
## 5 0.61612301 0.61596065 0.59969349 0.6772175 0.4203279  
## 6 0.77022684 0.76218346 0.74771304 0.8091283 0.4901252  
## 7 0.87785492 0.86662833 0.86038859 0.9013783 0.5554533  
## 8 0.94213279 0.93190637 0.93278039 0.9558283 0.6157056  
## 9 0.97546206 0.96817194 0.97203495 0.9829530 0.6705336  
## 10 0.99064540 0.98630473 0.98999970 0.9943570 0.7198092

aproxFunctiaDeRepartitie(100, 0.1, 1, 10)

## Aproximari pentru n = 100 p = 0.1 - Functia de repartitie  
## Binomiala Poisson Normala Normala corectie Camp-Paulson  
## 1 0.0003216881 0.0004993992 0.002016121 0.003362222 0.04354215  
## 2 0.0019448847 0.0027693957 0.005467736 0.008672377 0.07647190  
## 3 0.0078364871 0.0103360507 0.013418521 0.020258316 0.11369988  
## 4 0.0237110827 0.0292526881 0.029849798 0.042937667 0.15464229  
## 5 0.0575768865 0.0670859629 0.060315382 0.082767486 0.19862641  
## 6 0.1171556154 0.1301414209 0.110994309 0.145525186 0.24492889  
## 7 0.2060508618 0.2202206466 0.186630109 0.234242582 0.29281356  
## 8 0.3208738884 0.3328196788 0.287907232 0.346764018 0.34156141  
## 9 0.4512901654 0.4579297145 0.409576429 0.474804581 0.39049365  
## 10 0.5831555123 0.5830397502 0.540716715 0.605525871 0.43898892

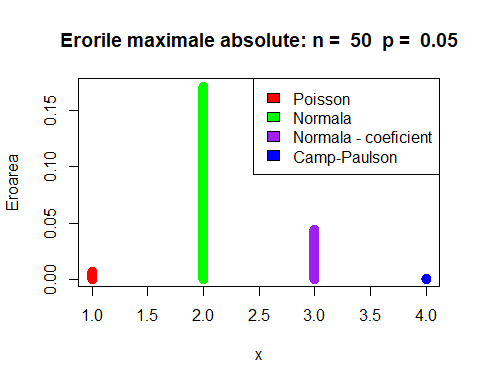
######################  
  
## Exercitiul 5  
  
### Functie care ilustreaza grafic erorile maximale absolute  
eroareMaxAbs <- function(n, p, a, b) {  
 errPois = errNorm = errNormCoef = errCamp = 0  
 bin = pbinom(a:b, n, p) # binomiala  
 lambda = n\*p;  
 for(i in 1:1000) {  
 y = rnorm(n, n \* p, sqrt(n \* p \* (1 - p)))  
 nrm = ((a:b + 0.5) - mean(y))/sd(y) # cu coeficient de corelatie  
 normCoef = pnorm(nrm)  
 nrm = (a:b - mean(y))/sd(y) # fara coeficient de corelatie  
 norm = pnorm(nrm)  
 pois = ppois(a:b, lambda) # poisson  
   
 a1 = 1 / (9\*(n - a:b))  
 b1 = 1 / (9\*(a:b + 1))  
 r = ((a:b + 1)\*(1 - p)) / (p \* (n - a:b))  
 c = (1 - b1) \* (r^(1 / 3))  
 sigma = sqrt(a1 + b1 \* (r ^ (2 / 3)))  
 miu = 1 - a1  
   
 camp = pnorm((c - miu) / sigma) # Camp-Paulson  
   
 }  
   
 errPois = max(abs(bin - pois))  
 errNorm = max(abs(bin - norm))  
 errNormCoef = max(abs(bin - normCoef))  
 errCamp = max(abs(bin - camp))  
 print(errPois)  
 print(errNorm)  
 print(errNormCoef)  
 print(errCamp)  
   
 plot(1:4, c(errPois, errNorm, errNormCoef, errCamp), type = "h",   
 main = paste("Erorile maximale absolute: n = ", n, " p = ", p), xlab = "x", ylab = "Eroarea",  
 lwd = 10, col = c("red", "green", "purple", "blue"))  
 legend("topright", c("Poisson", "Normala", "Normala - coeficient", "Camp-Paulson"),  
 fill = c("red", "green", "purple", "blue")  
 )  
   
}  
  
  
eroareMaxAbs(25, 0.1, 1, 10)

## [1] 0.01609159  
## [1] 0.2903693  
## [1] 0.1761699  
## [1] 0.001260515



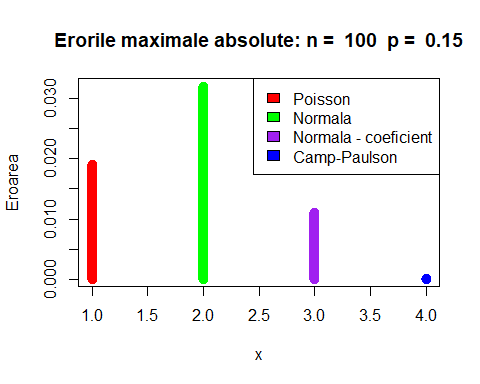
eroareMaxAbs(50, 0.05, 1, 10)

## [1] 0.007865743  
## [1] 0.1709743  
## [1] 0.04466368  
## [1] 0.001398888

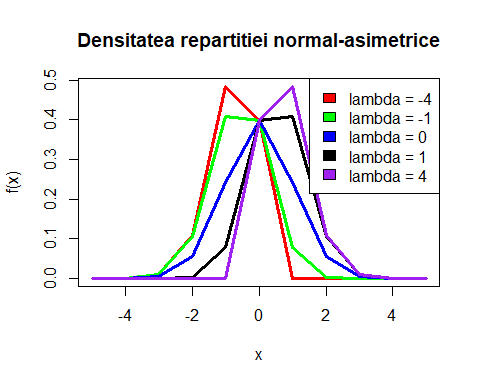


eroareMaxAbs(100, 0.15, 1, 10)

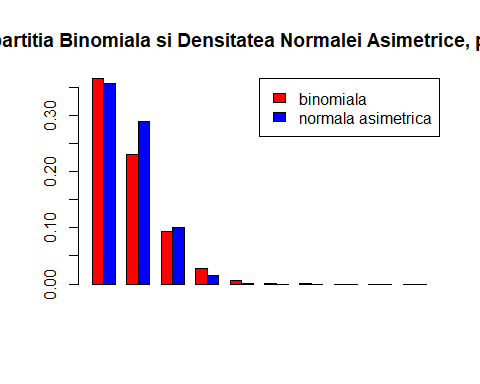
## [1] 0.01901701  
## [1] 0.03192767  
## [1] 0.01113482  
## [1] 0.0001973995



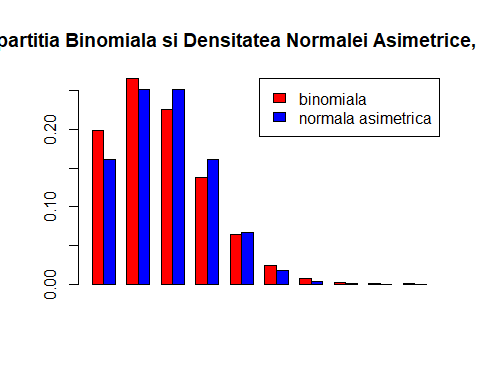
#### Observatii:   
#### Aproximarile normala si normala cu coeficient sunt cu atat mai bune cu cat n este mai mare, iar p se apropie de 0.5  
#### Se poate ca aproximarea normala fara coeficient sa aiba o eroare mai mica decat cea cu coeficient  
#### Aproximarea Camp-Paulson are eroarea cea mai apropiata de 0  
  
  
######################  
  
## Exercitiul 6  
  
### Functie care determina densitatea repartitiei normal-asimetrice  
  
dsnorm <- function(x, miu = 0, sigma = 1, lambda) {  
 dens = dnorm((x - miu)/sigma)  
 rep = pnorm(lambda\*((x - miu)/sigma))  
 return ((2 / sigma)\* dens \* rep)  
}  
  
  
plot(-5:5, dsnorm(-5:5, 0, 1, -4),type="l", col="red",  
 xlab = "x",  
 ylab = "f(x)",   
 main = "Densitatea repartitiei normal-asimetrice",  
 lwd = 3  
)  
lines(-5:5, dsnorm(-5:5, 0, 1, -1), col="green", lwd = 3)  
lines(-5:5, dsnorm(-5:5, 0, 1, 0), col = "blue", lwd = 3)  
lines(-5:5, dsnorm(-5:5, 0, 1, 1), col = "black", lwd = 3)  
lines(-5:5, dsnorm(-5:5, 0, 1, 4), col = "purple", lwd = 3)  
legend("topright", c("lambda = -4", "lambda = -1", "lambda = 0", "lambda = 1", "lambda = 4"),  
 fill = c("red", "green", "blue", "black", "purple")  
)



######################  
  
## Exercitiul 7  
  
### Functie care determina cei lambda, sigma si miu  
  
valoriParam <- function(n, p) {  
   
 f <- function(x) ((1 - (2 / pi) \* (x / (1 + x)))^3) / ((2 / pi) \* ((4 / pi - 1)^2) \*   
 (x / (1 + x))^3) - (n \* p \* (1 - p)) / ((1 - 2 \* p)^2)   
   
 val = uniroot(f, lower = 0, upper = 1, extendInt = "yes")  
   
 x = as.numeric(val[1])  
   
 lambda = sign(1 - 2 \* p) \* sqrt(x)  
   
 sigma = sqrt((n \* p \* (1 - p)) / (1 - (2 / pi) \* (x / (1 + x))))  
   
 miu = n \* p - sigma \* sqrt((2 / pi) \* (x / (1 + x)))  
   
 return(list(lambda, sigma, miu))  
   
   
}  
  
# p = 0.05  
par = as.numeric(valoriParam(25, 0.05))  
  
var = rbind(dbinom(1:10, 25, 0.05), dsnorm(1:10, par[3], par[2], par[1]))  
barplot(var, beside = T,  
 col = c("red", "blue"),   
 main = paste("Repartitia Binomiala si Densitatea Normalei Asimetrice, p = ", 0.05))  
legend("topright", c("binomiala", "normala asimetrica"),  
 fill = c("red", "blue"))



# p = 0.1  
par = as.numeric(valoriParam(25, 0.1))  
  
var = rbind(dbinom(1:10, 25, 0.1), dsnorm(1:10, par[3], par[2], par[1]))  
barplot(var, beside = T,  
 col = c("red", "blue"),   
 main = paste("Repartitia Binomiala si Densitatea Normalei Asimetrice, p = ", 0.1))  
legend("topright", c("binomiala", "normala asimetrica"),  
 fill = c("red", "blue"))



######################  
  
## Exercitiul 8  
  
### Functie care calculeaza eroarea de aproximare prin normala asimetrica - Functia de masa  
aproxNormStandDens <- function(n, p, a, b) {  
 lambda = n\*p;  
 par = as.numeric(valoriParam(n, p))  
 x = matrix(numeric((b - a + 1) \* 2),   
 ncol = 2,  
 dimnames = list(a:b, c("Binomiala", "Normala Asimentrica Standard")))  
   
 x[, 1] = dbinom(a:b, n, p) # binomiala  
 x[, 2] = dsnorm(a:b, par[3], par[2], par[1])  
   
 cat("Aproximari pentru n = ", n, " p = ", p, " - Functia de masa\n\n")  
 print(x)  
 error = max(abs(x[, 1] - x[, 2]))  
 print(error)  
 return(error)  
}  
  
err1 = aproxNormStandDens(25, 0.05, 1, 10)

## Aproximari pentru n = 25 p = 0.05 - Functia de masa  
##   
## Binomiala Normala Asimentrica Standard  
## 1 3.649863e-01 3.565861e-01  
## 2 2.305177e-01 2.888910e-01  
## 3 9.301589e-02 1.008282e-01  
## 4 2.692565e-02 1.516032e-02  
## 5 5.951987e-03 9.820034e-04  
## 6 1.044208e-03 2.740288e-05  
## 7 1.491726e-04 3.294260e-07  
## 8 1.766518e-05 1.706075e-09  
## 9 1.756187e-06 3.806422e-12  
## 10 1.478894e-07 3.658595e-15  
## [1] 0.05837336

err2 = aproxNormStandDens(25, 0.1, 1, 10)

## Aproximari pentru n = 25 p = 0.1 - Functia de masa  
##   
## Binomiala Normala Asimentrica Standard  
## 1 1.994161e-01 1.613138e-01  
## 2 2.658881e-01 2.515888e-01  
## 3 2.264973e-01 2.515888e-01  
## 4 1.384150e-01 1.613138e-01  
## 5 6.459368e-02 6.631809e-02  
## 6 2.392358e-02 1.748126e-02  
## 7 7.215049e-03 2.954567e-03  
## 8 1.803762e-03 3.201806e-04  
## 9 3.785674e-04 2.224727e-05  
## 10 6.730087e-05 9.911482e-07  
## [1] 0.03810229

err3 = aproxNormStandDens(50, 0.05, 1, 10)

## Aproximari pentru n = 50 p = 0.05 - Functia de masa  
##   
## Binomiala Normala Asimentrica Standard  
## 1 0.2024867770 1.611981e-01  
## 2 0.2611013704 2.455956e-01  
## 3 0.2198748382 2.455956e-01  
## 4 0.1359752289 1.611981e-01  
## 5 0.0658406372 6.944455e-02  
## 6 0.0259897252 1.963612e-02  
## 7 0.0085981046 3.644286e-03  
## 8 0.0024323585 4.439238e-04  
## 9 0.0005974214 3.549307e-05  
## 10 0.0001289172 1.862593e-06  
## [1] 0.04128871

err4 = aproxNormStandDens(50, 0.1, 1, 10)

## Aproximari pentru n = 50 p = 0.1 - Functia de masa  
##   
## Binomiala Normala Asimentrica Standard  
## 1 0.02863208 0.03178518  
## 2 0.07794290 0.06918458  
## 3 0.13856515 0.12058244  
## 4 0.18090450 0.16828634  
## 5 0.18492460 0.18806319  
## 6 0.15410383 0.16828634  
## 7 0.10762807 0.12058243  
## 8 0.06427788 0.06918458  
## 9 0.03332927 0.03178518  
## 10 0.01518333 0.01169312  
## [1] 0.01798271

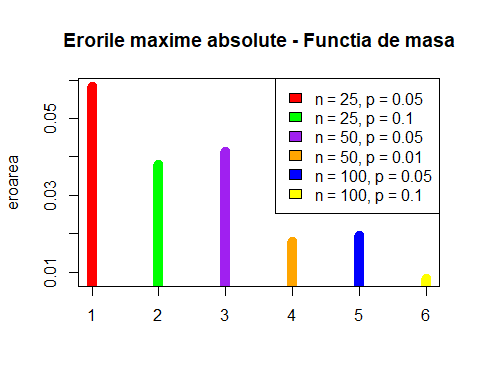
err5 = aproxNormStandDens(100, 0.05, 1, 10)

## Aproximari pentru n = 100 p = 0.05 - Functia de masa  
##   
## Binomiala Normala Asimentrica Standard  
## 1 0.03116068 0.03397191  
## 2 0.08118177 0.07097843  
## 3 0.13957568 0.12014410  
## 4 0.17814264 0.16475859  
## 5 0.18001783 0.18304727  
## 6 0.15001486 0.16475858  
## 7 0.10602554 0.12014409  
## 8 0.06487089 0.07097843  
## 9 0.03490130 0.03397191  
## 10 0.01671588 0.01317295  
## [1] 0.01943158

err6 = aproxNormStandDens(100, 0.1, 1, 10)

## Aproximari pentru n = 100 p = 0.1 - Functia de masa  
##   
## Binomiala Normala Asimentrica Standard  
## 1 0.0002951267 0.001477282  
## 2 0.0016231966 0.003798661  
## 3 0.0058916025 0.008740629  
## 4 0.0158745955 0.017996988  
## 5 0.0338658038 0.033159046  
## 6 0.0595787289 0.054670026  
## 7 0.0888952464 0.080656911  
## 8 0.1148230266 0.106482672  
## 9 0.1304162771 0.125794411  
## 10 0.1318653468 0.132980760  
## [1] 0.008340355

plot(1:6, c(err1, err2, err3, err4, err5, err6), type = "h", lwd = 10,  
 col = c("red", "green", "purple", "orange", "blue", "yellow"),   
 main = "Erorile maxime absolute - Functia de masa",   
 xlab = "", ylab = "eroarea")  
legend("topright", c("n = 25, p = 0.05", "n = 25, p = 0.1", "n = 50, p = 0.05", "n = 50, p = 0.01",  
 "n = 100, p = 0.05", "n = 100, p = 0.1"),   
 fill = c("red", "green", "purple", "orange", "blue", "yellow"))



## Functie care calculeaza eroarea de aproximare prin normala asimetrica - Functia de repartitie  
aproxNormStandFuncRep <- function(n, p, a, b) {  
   
 par = as.numeric(valoriParam(n, p))  
 lambda = as.numeric(par[1])  
 miu = as.numeric(par[2])  
 sigma = as.numeric(par[3])  
   
 x = matrix(numeric((b - a + 1) \* 2),   
 ncol = 2,  
 dimnames = list(a:b, c("Binomiala", "Normala Asimentrica Standard")))  
   
 x[, 1] = pbinom(a:b, n, p) # binomiala  
   
 t = (x - miu) / sigma  
   
 psi <-function(t) 2 \* dnorm(t) \* pnorm(lambda \* t)  
   
 for (i in a:b) {  
 y = integrate(psi, lower = -Inf, upper = (i + 0.5 - miu) /sigma)  
   
 x[i, 2] = as.numeric(y[1])  
 }  
   
   
 cat("Aproximari pentru n = ", n, " p = ", p, " - Functia de masa\n\n")  
 print(x)  
 error = max(abs(x[, 1] - x[, 2]))  
 print(error)  
 return(error)  
}  
  
err1 = aproxNormStandFuncRep(25, 0.05, 1, 10)

## Aproximari pentru n = 25 p = 0.05 - Functia de masa  
##   
## Binomiala Normala Asimentrica Standard  
## 1 0.6423759 0.6269492  
## 2 0.8728935 0.8703730  
## 3 0.9659094 0.9733529  
## 4 0.9928351 0.9968990  
## 5 0.9987870 0.9998009  
## 6 0.9998312 0.9999931  
## 7 0.9999804 0.9999999  
## 8 0.9999981 1.0000000  
## 9 0.9999998 1.0000000  
## 10 1.0000000 1.0000000  
## [1] 0.01542667

err2 = aproxNormStandFuncRep(25, 0.1, 1, 10)

## Aproximari pentru n = 25 p = 0.1 - Functia de masa  
##   
## Binomiala Normala Asimentrica Standard  
## 1 0.2712059 0.4975166  
## 2 0.5370941 0.6536817  
## 3 0.7635914 0.7872118  
## 4 0.9020064 0.8845977  
## 5 0.9666001 0.9451778  
## 6 0.9905236 0.9773198  
## 7 0.9977387 0.9918650  
## 8 0.9995425 0.9974788  
## 9 0.9999210 0.9993266  
## 10 0.9999883 0.9998453  
## [1] 0.2263107

err3 = aproxNormStandFuncRep(50, 0.05, 1, 10)

## Aproximari pentru n = 50 p = 0.05 - Functia de masa  
##   
## Binomiala Normala Asimentrica Standard  
## 1 0.2794318 0.4909328  
## 2 0.5405331 0.6475876  
## 3 0.7604080 0.7824109  
## 4 0.8963832 0.8813787  
## 5 0.9622238 0.9433408  
## 6 0.9882136 0.9764277  
## 7 0.9968117 0.9914962  
## 8 0.9992440 0.9973490  
## 9 0.9998414 0.9992877  
## 10 0.9999704 0.9998354  
## [1] 0.211501

err4 = aproxNormStandFuncRep(50, 0.1, 1, 10)

## Aproximari pentru n = 50 p = 0.1 - Functia de masa  
##   
## Binomiala Normala Asimentrica Standard  
## 1 0.03378586 0.4479603  
## 2 0.11172876 0.5277902  
## 3 0.25029391 0.6065155  
## 4 0.43119841 0.6811014  
## 5 0.61612301 0.7489896  
## 6 0.77022684 0.8083538  
## 7 0.87785492 0.8582250  
## 8 0.94213279 0.8984752  
## 9 0.97546206 0.9296842  
## 10 0.99064540 0.9529323  
## [1] 0.4160614

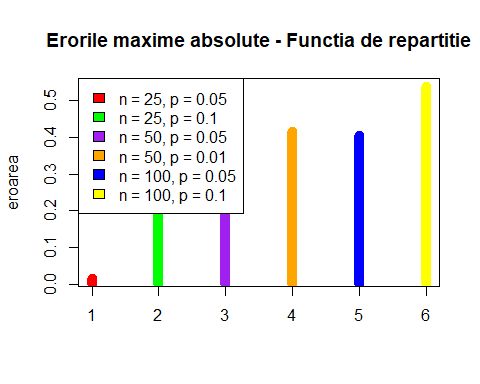
err5 = aproxNormStandFuncRep(100, 0.05, 1, 10)

## Aproximari pentru n = 100 p = 0.05 - Functia de masa  
##   
## Binomiala Normala Asimentrica Standard  
## 1 0.03708121 0.4433491  
## 2 0.11826298 0.5231511  
## 3 0.25783866 0.6020322  
## 4 0.43598130 0.6769396  
## 5 0.61599913 0.7452783  
## 6 0.76601398 0.8051749  
## 7 0.87203952 0.8556093  
## 8 0.93691041 0.8964078  
## 9 0.97181171 0.9281147  
## 10 0.98852759 0.9517877  
## [1] 0.4062679

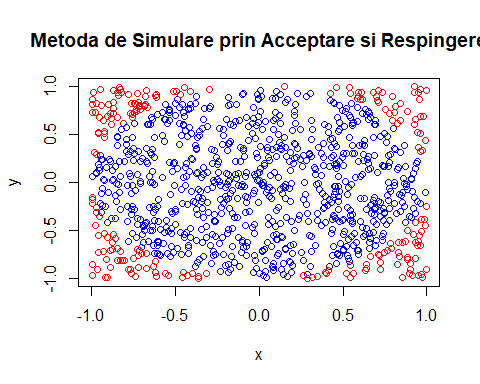
err6 = aproxNormStandFuncRep(100, 0.1, 1, 10)

## Aproximari pentru n = 100 p = 0.1 - Functia de masa  
##   
## Binomiala Normala Asimentrica Standard  
## 1 0.0003216881 0.4378186  
## 2 0.0019448847 0.4775461  
## 3 0.0078364871 0.5174981  
## 4 0.0237110827 0.5572750  
## 5 0.0575768865 0.5964824  
## 6 0.1171556154 0.6347428  
## 7 0.2060508618 0.6717064  
## 8 0.3208738884 0.7070609  
## 9 0.4512901654 0.7405388  
## 10 0.5831555123 0.7719234  
## [1] 0.5389055

plot(1:6, c(err1, err2, err3, err4, err5, err6), type = "h", lwd = 10,  
 col = c("red", "green", "purple", "orange", "blue", "yellow"),   
 main = "Erorile maxime absolute - Functia de repartitie",   
 xlab = "", ylab = "eroarea")  
legend("topleft", c("n = 25, p = 0.05", "n = 25, p = 0.1", "n = 50, p = 0.05", "n = 50, p = 0.01",  
 "n = 100, p = 0.05", "n = 100, p = 0.1"),   
 fill = c("red", "green", "purple", "orange", "blue", "yellow"))



##################################################################  
  
# Problema 2  
  
## Metoda de Simulare prin Acceptare si Respingere  
  
for(i in 1:1000)  
{  
 x <- c(runif(1, -1, 1))  
 y <- c(runif(1, -1, 1))   
}  
plot (x, y, main="Metoda de Simulare prin Acceptare si Respingere",   
 col = ifelse(x\*x+y\*y<=1, "blue", "red"))



s = 0  
nr = 0  
for(i in 1:1000)  
 if(x[i]\*x[i]+y[i]\*y[i]<=1)  
 {  
 distanta = sqrt(x[i]\*x[i]+y[i]\*y[i])   
 s = s+distanta  
 nr = nr + 1  
 }   
print(s/nr)

## [1] 0.6598466

##################################################################  
  
# Problema 3  
  
## Subpunct A  
f <- function(t, z)  
{return(t^(z-1)\*exp(-t))}  
  
  
fgam <- function(x)  
{  
 if(x==1) return(1)  
 else if(x%%1==0 && x>0) #daca e nr natural mai mare de 0  
 {  
 count <- 1  
 for(i in 1:(x-1)) count = count \* i  
 return(count)  
 }  
 else if(x==1/2)  
 {  
 return(sqrt(pi))  
 }  
 else if(x>1) return((x-1) \* fgam(x-1))  
 else if(x>0 && x<1)  
 {  
 return(integrate(f, lower=0, upper=Inf, z=x)$value)  
 }  
}  
  
  
######################  
  
## Subpunct B  
fbet <- function(a, b)  
{  
 if(a>0 && b>0 && a+b==1)  
 {return(pi/sin(a\*pi))}  
 return(fgam(a)\*fgam(b)/fgam(a+b))  
}  
  
  
  
######################  
  
## Subpunct C  
Gamma <- function(a, b, x)  
{return(1/(b^a\*fgam(a))\*x^(a-1)\*exp(-x/b))}  
Beta <- function(a, b, x)  
{return(1/fbet(a, b)\*x^(a-1)\*(1-x)^(b-1))}  
  
fprobgamma1 <- function(a, b)  
{return(integrate(Gamma, 0, 3, a=a, b=b)$value)}  
  
fprobgamma2 <- function(a, b)  
{return(integrate(Gamma, 2, 5, a=a, b=b)$value)}  
  
fprobgamma3 <- function(a, b)  
{  
 x=integrate(Gamma, 3, 4, a=a, b=b)$value  
 y=integrate(Gamma, 2, Inf, a=a, b=b)$value  
 return(x/y)  
}  
  
fprobbeta4 <- function(a, b)  
{return(ceiling(1-integrate(Beta, 0, 1, a=a, b=b)$value))}  
  
fprobgamma5 <- function(a, b)  
{return(integrate(Gamma, 4, 6, a=a, b=b)$value)}  
  
fprobgamma6 <- function(a, b)  
{  
 x=integrate(Gamma, 0, 1, a=a, b=b)$value  
 y=integrate(Gamma, 0, 7, a=a, b=b)$value  
 return(x/y)  
}  
  
fcomuna <- function(x, y, a, b) #pt ca sunt independente  
{return(1/(b^a\*fgam(a))\*x^(a-1)\*exp(-x/b)\*1/fbet(a, b)\*y^(a-1)\*(1-y)^(b-1))}  
  
fprob7 <- function(a, b)   
{  
 integrate(function(y, a, b)   
 {  
 sapply(y, function(y)   
 {  
 integrate(function(x, a, b) fcomuna(x, y, a, b), 0, 5-y, a=a, b=b)$value  
 })  
 }, 0, 1, a=a, b=b)$value  
}  
  
fprob8 <- function(a, b)  
{return(1-integrate(function(y, a, b)   
{  
 sapply(y, function(y)   
 {  
 integrate(function(x, a, b) fcomuna(x, y, a, b), 0, 0.5+y, a=a, b=b)$value  
 })  
}, 0, 1, a=a, b=b)$value)  
}  
  
  
### Exemple:  
  
fprobgamma1(1, 2)

## [1] 0.7768698

fprobgamma1(2, 2)

## [1] 0.4421746

fprobgamma1(3, 5)

## [1] 0.02311529

fprobgamma2(1, 2)

## [1] 0.2857944

fprobgamma2(2, 2)

## [1] 0.4484614

fprobgamma2(3, 5)

## [1] 0.07237507

fprobgamma3(1, 2)

## [1] 0.2386512

fprobgamma3(2, 2)

## [1] 0.2063442

fprobgamma3(3, 5)

## [1] 0.02450152

fprobbeta4(1, 2)

## [1] 0

fprobbeta4(2, 2)

## [1] 0

fprobbeta4(3, 5)

## [1] 1

fprobgamma5(1, 2)

## [1] 0.08554821

fprobgamma5(2, 2)

## [1] 0.2068576

fprobgamma5(3, 5)

## [1] 0.07309031

fprobgamma6(1, 2)

## [1] 0.4057211

fprobgamma6(2, 2)

## [1] 0.1043893

fprobgamma6(3, 5)

## [1] 0.006897692

fprob7(1, 2)

## [1] 0.9023377

fprob7(2, 2)

## [1] 0.6566302

fprob7(3, 5)

## [1] 0.06713886

fprob8(1, 2)

## [1] 0.6637293

fprob8(2, 2)

## [1] 0.9078794

fprob8(3, 5)

## [1] 0.9991456

######################  
  
## Subpunct D  
  
m = c( 0, fprobgamma1(1, 2), 0, fprobgamma2(1, 2),   
 0, fprobgamma3(1, 2), 0, fprobbeta4(1,2),   
 0, fprobgamma5(1, 2), 0, fprobgamma6(1, 2),  
 0, fprob7(1, 2), 0, fprob8(1, 2))  
  
m[1] = pgamma(q = 3, shape = 1, scale = 2)  
m[3] = pgamma(q = 5, shape = 1, scale = 2) - pgamma(q = 2, shape = 1, scale = 2)  
m[5] = (pgamma(q = 4, shape = 1, scale = 2) - pgamma(q = 3, shape = 1, scale = 2)) / (1 - pgamma(q = 2, shape = 1, scale = 2))  
m[7] = 1 - pbeta(q = 2, 1, 2)  
m[9] = pgamma(q = 6, shape = 1, scale = 2) - pgamma(q = 4, shape = 1, scale = 2)  
m[11] = (pgamma(q = 1, shape = 1, scale = 2) - pgamma(q = 0, shape = 1, scale = 2)) / pgamma(q = 7, shape = 1, scale = 2)  
m[13] = pgamma(q = 5, shape = 1, scale = 2) \* pbeta(q = 1, 1, 2)  
m[15] = 1 - pgamma(q = 0.5, shape = 1, scale = 2) \* pbeta(q = 1, 1, 2)  
  
MATRICE <- matrix(m, ncol=2, byrow=TRUE)  
  
colnames(MATRICE) = c("R","Punct C")  
rownames(MATRICE) = c("P(X < 3)","P(2 < X < 5)","P(3 < X < 4 | X > 2)","P(Y > 2)","P(4 < X < 6)","P(0 < X < 1 | X < 7)", "P(X + Y < 5)", "P(X - Y > 0.5)")  
MATRICE <- as.table(MATRICE)  
  
MATRICE

## R Punct C  
## P(X < 3) 0.77686984 0.77686984  
## P(2 < X < 5) 0.28579444 0.28579444  
## P(3 < X < 4 | X > 2) 0.23865122 0.23865122  
## P(Y > 2) 0.00000000 0.00000000  
## P(4 < X < 6) 0.08554821 0.08554821  
## P(0 < X < 1 | X < 7) 0.40572105 0.40572105  
## P(X + Y < 5) 0.91791500 0.90233772  
## P(X - Y > 0.5) 0.77880078 0.66372929