Proiect

Olaru Bogdan-Ioan & Nastase Stefan

6 februarie 2018

Problema 1

1.1

Consideram un esantion de N = 1000 pentru care calculam media si varianta repartitiilor Binomiala, Poisson, Exponentiala si Normala.

```
N = 1000
# definim functii pentru medie si vaianta.
medie = function(v) {
  return(sum(v)/N)
}
varianta = function(v, x) {
  v = v - x
  v = v * v
  return(sum(v) / (N - 1))
}
# generam un esantion de talie N pentru Bin, ii calculam media si varianta
obs  = rbinom(N, 100, 0.5)
obs[0:50]
```

```
##  [1] 51 54 51 61 41 50 45 41 41 57 44 50 48 56 50 49 53 48 54 54 47 42 47
## [24] 50 54 50 44 55 46 51 49 52 48 49 53 49 48 52 49 60 48 44 46 49 57 51
## [47] 51 54 55 45
```

```
medie(obs)
```

```
## [1] 49.576
```

```
varianta(obs, medie(obs))
```

```
## [1] 22.94917
```

```
# generam un esantion de talie N pentru Pois, ii calculam media si varianta
obs = rpois(N, 3)
obs[0:50]
```

```
##  [1] 7 3 1 1 3 2 1 5 1 0 4 2 1 1 4 3 3 5 3 4 2 4 5 2 5 2 2 3 3 2 2 4 1 3 2
## [36] 7 1 3 1 6 2 2 4 2 2 2 2 5 1 1
```

```
medie(obs)
```

```
## [1] 2.925
```

```
varianta(obs, medie(obs))
```

```
## [1] 2.948323
```

```
# generam un esantion de talie N pentru Exp, ii calculam media si varianta
obs = rexp(N, 2)
obs[0:50]
```

```
##  [1] 0.223072389 1.304739675 0.772792547 0.255261880 1.051559982
##  [6] 0.651814243 0.700947740 0.901760350 1.877632829 0.110557648
## [11] 2.693352722 0.109740760 0.477217744 0.181612593 0.090483553
## [16] 0.009096910 1.214356820 0.777995152 0.023300619 0.945710362
## [21] 0.196522174 0.444038692 0.299681807 1.340703393 0.238807102
## [26] 1.878327141 0.288870593 0.086196278 0.776029184 0.032647765
## [31] 0.730604195 1.845258008 0.593086950 0.342813145 0.011127647
## [36] 0.271269627 0.641590411 0.009611667 0.110049932 0.868261232
## [41] 0.362205058 0.365531821 0.766694986 0.291525779 0.602757192
## [46] 0.725769087 0.578464979 0.290933422 0.290556261 0.095917561
```

```r
medie(obs)
```

```
## [1] 0.5095308
```

```r
varianta(obs, medie(obs))
```

```
## [1] 0.2493575
```

```r
# generam un esantion de talie N pentru Norm, ii calculam media si varianta
obs = rnorm(N);
obs[0:50]
```

```
##  [1] -0.95296438 -0.14520532 -0.60244947 -1.10210663 -0.49847191
##  [6] -0.10710148 -0.11114191 -0.10239629  0.28142394 -0.28812523
## [11] -0.02256463  0.05336802  0.10900262  2.23462365 -1.86508722
## [16] -0.45551916  0.62439635 -1.03847301  0.61873230  0.91291373
## [21] -0.34193975 -2.29270585 -0.42609043  0.13048171  1.46722200
## [26]  0.42242175 -2.66720252  0.99546394 -0.62114546  0.75044139
## [31] -0.34432181 -1.89421893  0.46421538 -0.06014991 -1.00535001
## [36] -0.84715246 -0.88807347  0.75614269 -1.87194056 -1.83838714
## [41] -0.35383544  1.17174614 -0.41757484  0.43119402 -1.33983483
## [46]  0.54056444  0.44161787  0.90009372  0.31522290 -0.94427570
```

```r
medie(obs)
```

```
## [1] -0.0003410489
```

```r
varianta(obs, medie(obs))
```

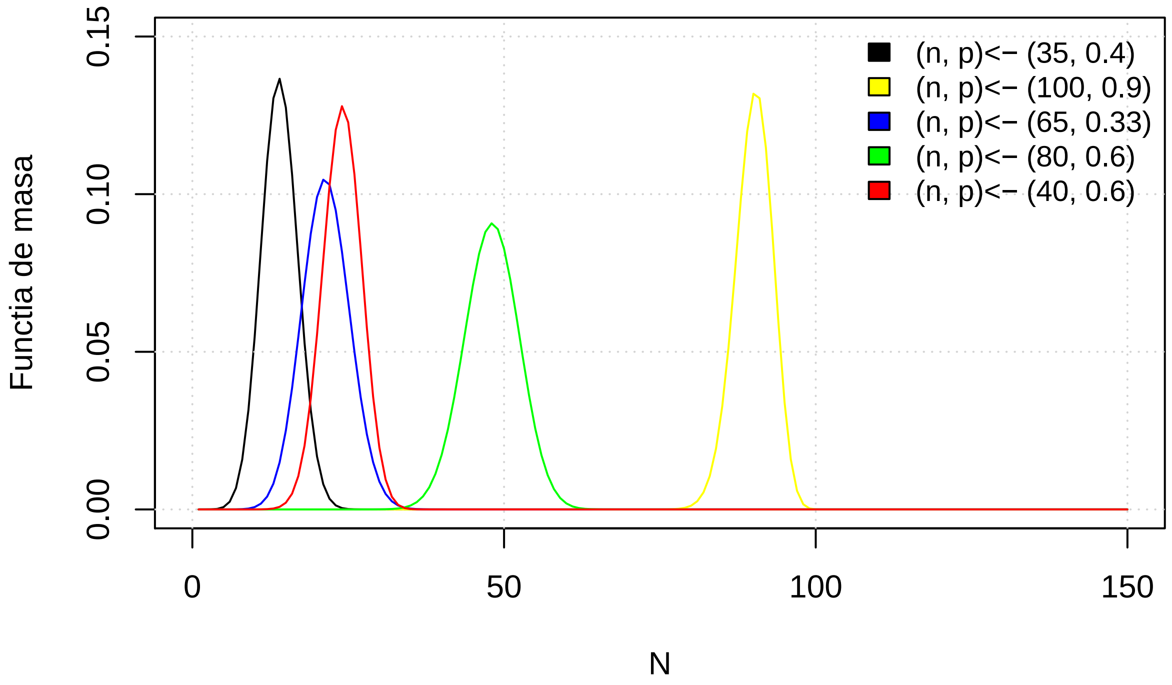
```
## [1] 0.9928095
```

1.2

Reprezentari grafice ale functiilor de masa, respectiv densitate.

Bin(n, p)

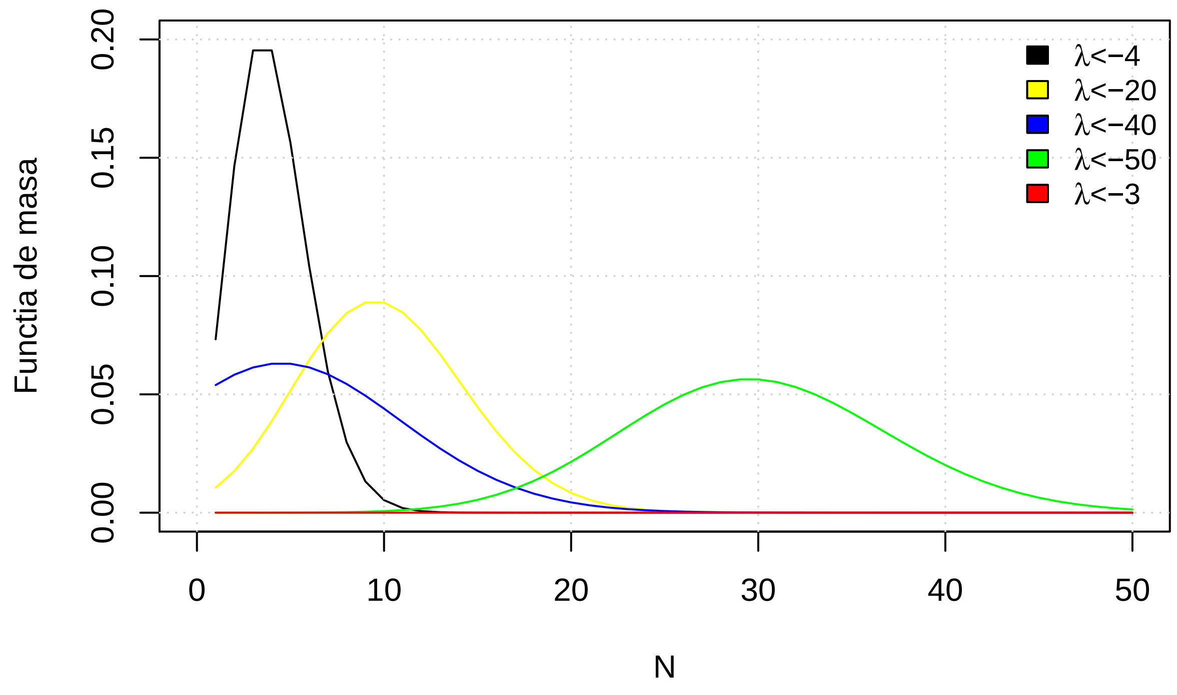
```r
# consideram setul de parametri
N = 150
n = c(35, 100, 65 , 80, 40)
p = c(0.4, 0.9, 0.33, 0.6, 0.6)
culori = c("black", "yellow", "blue", "green", "red")
# trasam graficul
plot(1:N, dbinom(1:N, n[1], p[1]), type = "l",
     ylab = "Functia de masa", xlab = "N", col = culori[1],
     xlim = c(0, N), ylim = c(0, 0.15))
grid()
for(i in 2:5) {
  lines(1:N, dbinom(1:N, n[i], p[i]), col = culori[i])
}
legend("topright", legend = paste("(n, p)<-",
      c("(35, 0.4)", "(100, 0.9)", "(65, 0.33)", "(80, 0.6)", "(40, 0.6)")),
        fill = culori, cex = 0.9, bty = "n")
```



Pois(λ)

```r
# consideram setul de parametri
N = 50
x = matrix(nrow = 5, ncol = N)
x[1,] = 1:50
x[2,] = 11:60
x[3,] = 36:85
x[4,] = 21:70
x[5,] = 41:90
l = c(4, 20, 40, 50, 3)
culori = c("black", "yellow", "blue", "green", "red")
# trasam graficul
plot(1:N, dpois(x[1,], l[1]), type = "l", ylab = "Functia de masa",
     xlab = "N", col = culori[1], xlim = c(0, N), ylim = c(0, 0.2))
grid()
```

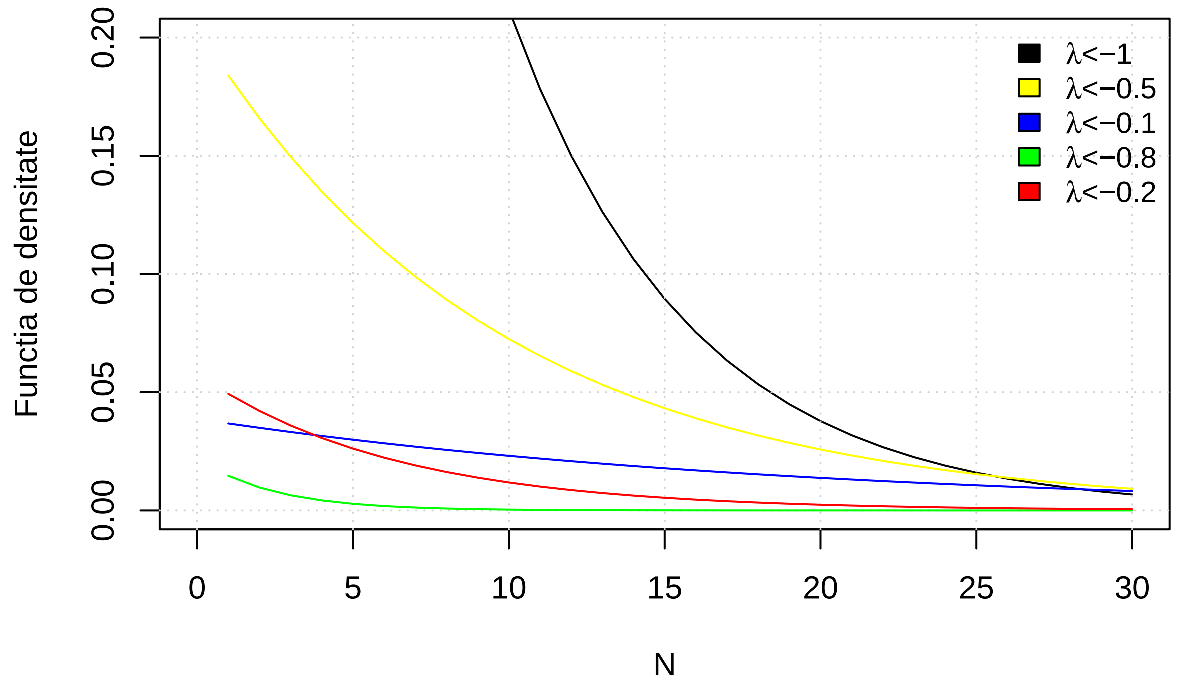
```
for(i in 2:5) {
  lines(1:N, dpois(x[i,], l[i]), col = culori[i])
}
legend("topright", legend = c(expression(paste(lambda, "<-4")),
                              expression(paste(lambda, "<-20")),
                              expression(paste(lambda, "<-40")),
                              expression(paste(lambda, "<-50")),
                              expression(paste(lambda, "<-3"))),
        fill = culori, cex = 0.9, bty = "n")
```



Exp(λ)

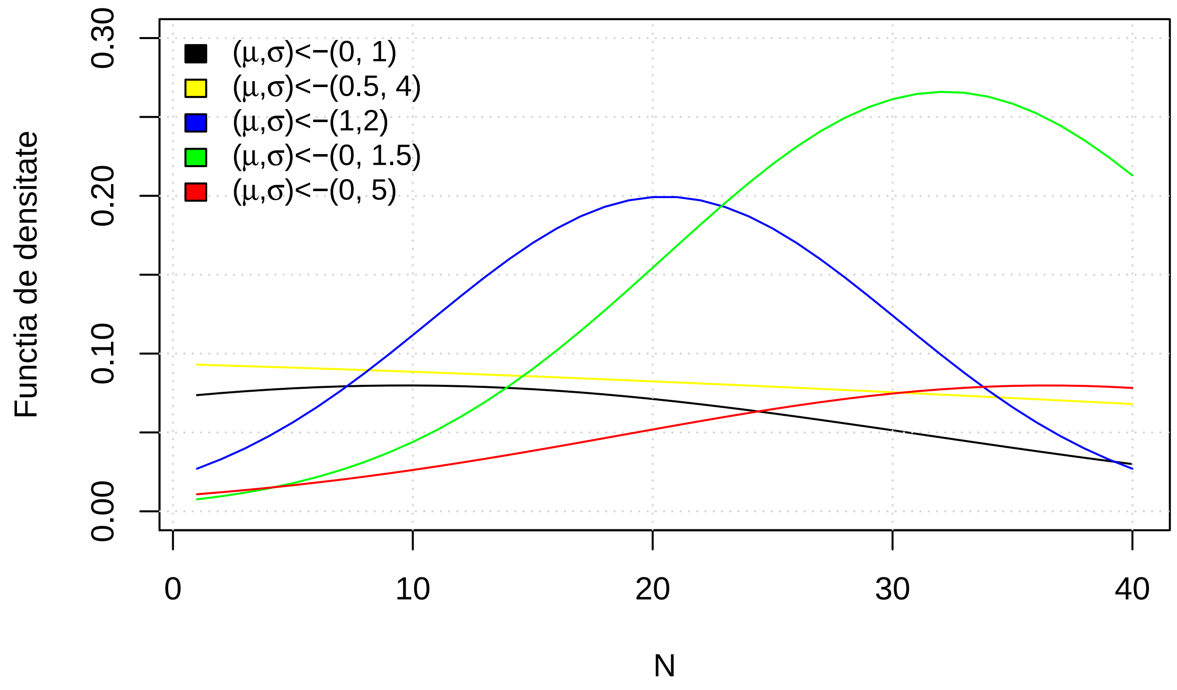
```
N = 30
x = matrix(nrow = 5, ncol = N)
x[1,] = seq(0, 5, length.out = 30)
x[2,] = seq(2, 8, length.out = 30)
x[3,] = seq(10, 25, length.out = 30)
x[4,] = seq(5, 20, length.out = 30)
x[5,] = seq(7, 30, length.out = 30)
r = c(1, 0.5, 0.1, 0.8, 0.2)
culori = c("black", "yellow", "blue", "green", "red")
# trasam graficul
plot(1:N, dexp(x[1,], rate = r[1]), type = "l",
     ylab = "Functia de densitate", xlab = "N", col = culori[1],
     xlim = c(0, N), ylim = c(0, 0.2))
```

```r
grid()
for(i in 2:5) {
  lines(1:N, dexp(x[i,], rate = r[i]), col = culori[i])
}
legend("topright", legend = c(expression(paste(lambda, "<-1")),
                              expression(paste(lambda, "<-0.5")),
                              expression(paste(lambda, "<-0.1")),
                              expression(paste(lambda, "<-0.8")),
                              expression(paste(lambda, "<-0.2"))),
        fill = culori, cex = 0.9, bty = "n")
```



N(μ, σ^2)

```r
N = 40
x = matrix(nrow = 5, ncol = N)
x[1,] = seq(-2, 7, length.out = N)
x[2,] = seq(2, 4, length.out = N)
x[3,] = seq(-3, 5, length.out = N)
x[4,] = seq(-4, 1, length.out = N)
x[5,] = seq(-10, 1, length.out = N)
mn = c(0, 0.5, 1, 0, 0)
s = c(1, 4, 2, 1.5, 5)
culori = c("black", "yellow", "blue", "green", "red")
# trasam graficul
plot(1:N, dnorm(x[1,], mean = mn[1], sd = s[i]), type = "l",
     ylab = "Functia de densitate", xlab = "N", col = culori[1],
     ylim = c(0, 0.3))
grid()
for(i in 2:5) {
  lines(1:N, dnorm(x[i,], mean = mn[i], sd = s[i]), col = culori[i])
}
legend("topleft", legend = c(expression(paste("(", mu, ",", sigma, ")<-(0, 1)")),
                             expression(paste("(", mu, ",", sigma, ")<-(0.5, 4)")),
                             expression(paste("(", mu, ",", sigma, ")<-(1,2)")),
                             expression(paste("(", mu, ",", sigma, ")<-(0, 1.5)")),
                             expression(paste("(", mu, ",", sigma, ")<-(0, 5)"))),
        fill = culori, cex = 0.9, bty = "n")
```



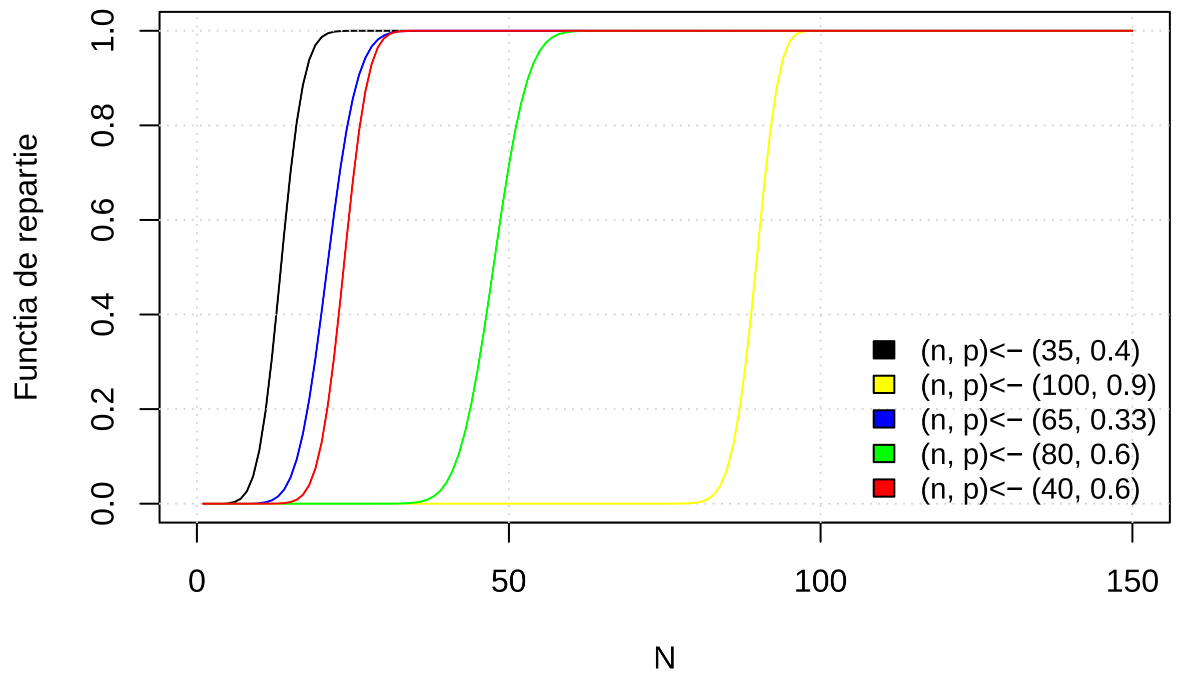
1.3

Reprezentari grafice ale functiilor de repartitie.

Bin(n, p)

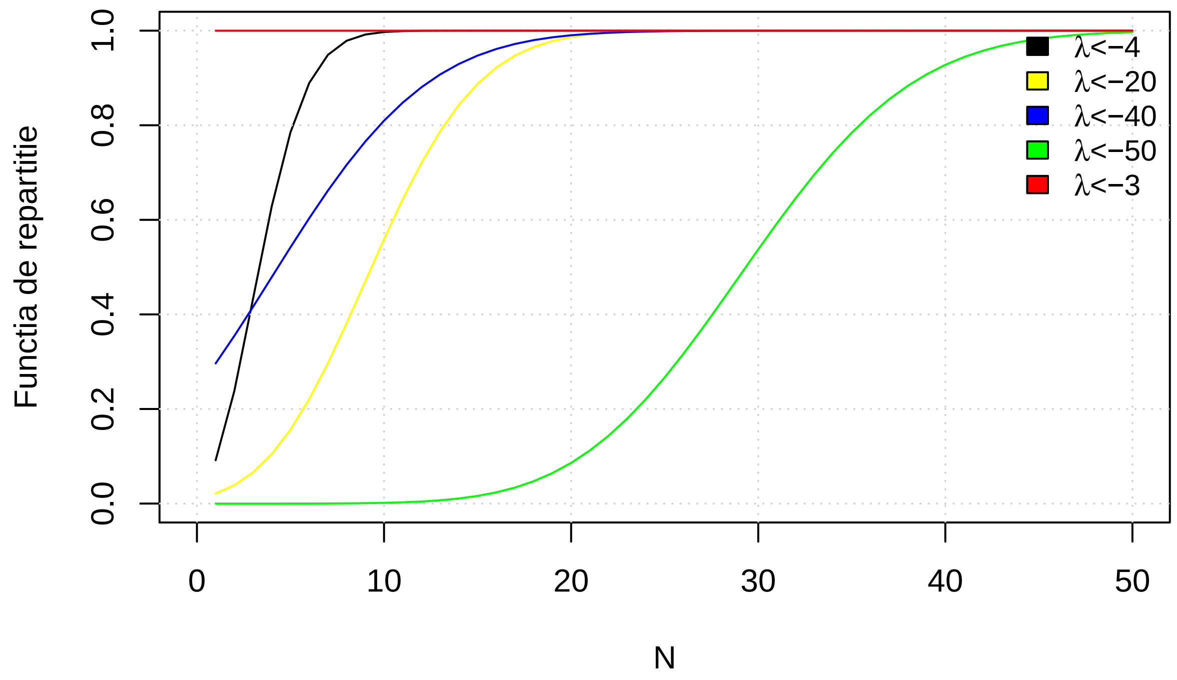
```r
# consideram setul de parametri
N = 150
n = c(35, 100, 65 , 80, 40)
p = c(0.4, 0.9, 0.33, 0.6, 0.6)
culori = c("black", "yellow", "blue", "green", "red")
# trasam graficul
```

```r
plot(1:N, pbinom(1:N, n[1], p[1]), type = "l",
     ylab = "Functia de repartie", xlab = "N", col = culori[1],
     xlim = c(0, N), ylim = c(0, 1))
grid()
for(i in 2:5) {
  lines(1:N, pbinom(1:N, n[i], p[i]), col = culori[i])
}
legend("bottomright", legend = paste("(n, p)<-",
      c("(35, 0.4)", "(100, 0.9)", "(65, 0.33)", "(80, 0.6)", "(40, 0.6)")),
        fill = culori, cex = 0.9, bty = "n")
```



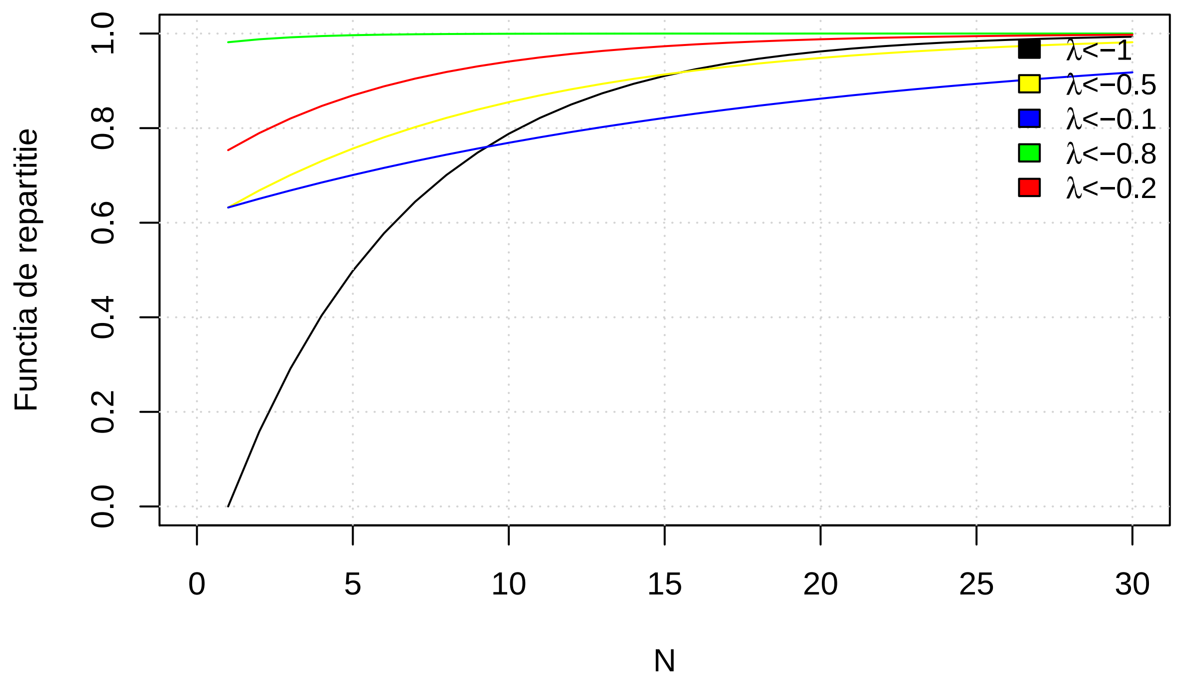
Pois(λ)

```r
# consideram setul de parametri
N = 50
x = matrix(nrow = 5, ncol = N)
x[1,] = 1:50
x[2,] = 11:60
x[3,] = 36:85
x[4,] = 21:70
x[5,] = 41:90
l = c(4, 20, 40, 50, 3)
culori = c("black", "yellow", "blue", "green", "red")
# trasam graficul
plot(1:N, ppois(x[1,], l[1]), type = "l", ylab = "Functia de repartitie",
     xlab = "N", col = culori[1], xlim = c(0, N), ylim = c(0, 1))
grid()
for(i in 2:5) {
  lines(1:N, ppois(x[i,], l[i]), col = culori[i])
}
legend("topright", legend = c(expression(paste(lambda, "<-4")),
                              expression(paste(lambda, "<-20")),
                              expression(paste(lambda, "<-40")),
                              expression(paste(lambda, "<-50")),
                              expression(paste(lambda, "<-3"))),
        fill = culori, cex = 0.9, bty = "n")
```



Exp(λ)

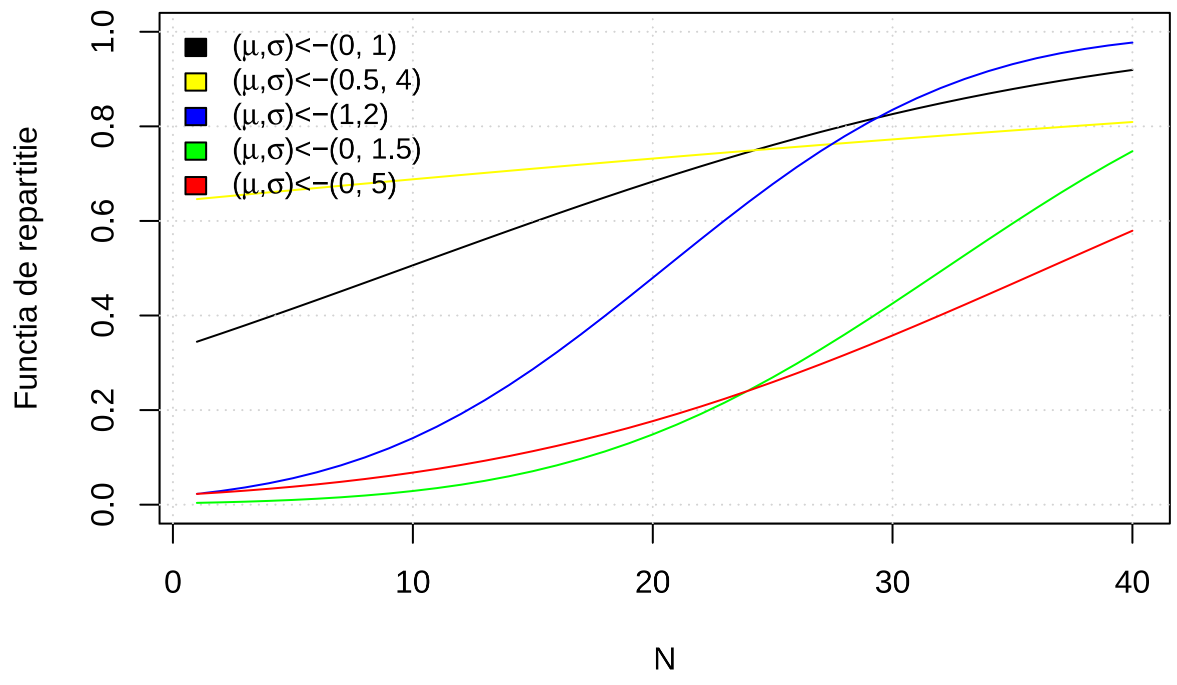
```r
N = 30
x = matrix(nrow = 5, ncol = N)
x[1,] = seq(0, 5, length.out = 30)
x[2,] = seq(2, 8, length.out = 30)
x[3,] = seq(10, 25, length.out = 30)
x[4,] = seq(5, 20, length.out = 30)
x[5,] = seq(7, 30, length.out = 30)
r = c(1, 0.5, 0.1, 0.8, 0.2)
culori = c("black", "yellow", "blue", "green", "red")
# trasam graficul
plot(1:N, pexp(x[1,], rate = r[1]), type = "l",
     ylab = "Functia de repartitie", xlab = "N", col = culori[1],
     xlim = c(0, N), ylim = c(0, 1))
grid()
for(i in 2:5) {
  lines(1:N, pexp(x[i,], rate = r[i]), col = culori[i])
}
legend("topright", legend = c(expression(paste(lambda, "<-1")),
                              expression(paste(lambda, "<-0.5")),
                              expression(paste(lambda, "<-0.1")),
                              expression(paste(lambda, "<-0.8")),
                              expression(paste(lambda, "<-0.2"))),
        fill = culori, cex = 0.9, bty = "n")
```



$\mathbf{N}(\mu, \sigma^2)$

```
N = 40
x = matrix(nrow = 5, ncol = N)
x[1,] = seq(-2, 7, length.out = N)
x[2,] = seq(2, 4, length.out = N)
x[3,] = seq(-3, 5, length.out = N)
x[4,] = seq(-4, 1, length.out = N)
x[5,] = seq(-10, 1, length.out = N)
mn = c(0, 0.5, 1, 0, 0)
s = c(1, 4, 2, 1.5, 5)
culori = c("black", "yellow", "blue", "green", "red")
```

```r
# trasam graficul
plot(1:N, pnorm(x[1,], mean = mn[1], sd = s[i]), type = "l",
     ylab = "Functia de repartitie", xlab = "N", col = culori[1],
     ylim = c(0, 1))
grid()
for(i in 2:5) {
  lines(1:N, pnorm(x[i,], mean = mn[i], sd = s[i]), col = culori[i])
}
legend("topleft", legend = c(expression(paste("(", mu, ",", sigma, ")<-(0, 1)")),
                             expression(paste("(", mu, ",", sigma, ")<-(0.5, 4)")),
                             expression(paste("(", mu, ",", sigma, ")<-(1,2)")),
                             expression(paste("(", mu, ",", sigma, ")<-(0, 1.5)")),
                             expression(paste("(", mu, ",", sigma, ")<-(0, 5)"))),
        fill = culori, cex = 0.9, bty = "n")
```



1.4

Tabele cu aproximari

```r
N = c(25, 50, 100)
P = c(0.05, 0.1)
k = 1:10
# definim functiile de aproximare
aproximarePoisson = function(s, n, p) {
  sum = 0
  l = n * p
  for(j in 0:s){
      sum = sum + exp(-l) * l ^ j / factorial(j)
  }
  return(sum)
}
aproximareNormala = function(n, p) {
  return(pnorm(k, n * p, sqrt(n * p * (1 - p))))
}
aproximareNormalaCorectie = function(n, p) {
  return(pnorm(k + 0.5, n * p, sqrt(n * p * (1 - p))))
}
aproximareCampPaulson = function(n, p) {
  a = 1 / (9 * (n - k))
  b = 1 / (9 * (k + 1))
  r = floor((k + 1) * (1 - p)) / floor(p * (n - k))
  sigma = sqrt(a + b * r ^ (2 / 3))
  mu = 1 - a
  c = (1 - b) * r ^ (1 / 3)
  return(pnorm(c, mu, sigma))
}
# functia pentru crearea tabelului
table = function(n, p) {
  Binomiala = pbinom(k, n, p)
  Poisson = aproximarePoisson(1, n, p)
  for(i in 2:10) {
    Poisson = c(Poisson, aproximarePoisson(i, n, p))
  }
  Normala = aproximareNormala(n, p)
  NormalaCorectie = aproximareNormalaCorectie(n, p)
  CampPaulson = aproximareCampPaulson(n, p)
  rez = data.frame(k, Binomiala, Normala, Poisson,
                   "Normala Corectie" = NormalaCorectie,
                   "Camp-Paulson" = CampPaulson)
  return(rez)
}
# n = 25, p = 0.05
table(N[1], P[1])
```

```
##     k Binomiala   Normala   Poisson Normala.Corectie Camp.Paulson
## 1   1 0.6423759 0.4092729 0.6446358        0.5907271    0.4177769
## 2   2 0.8728935 0.7543514 0.8684677        0.8743254    0.8063752
## 3   3 0.9659094 0.9458532 0.9617309        0.9805263    0.9478824
## 4   4 0.9928351 0.9941916 0.9908757        0.9985700    0.9878266
## 5   5 0.9987870 0.9997105 0.9981619        0.9999519    0.9974352
## 6   6 0.9998312 0.9999935 0.9996799        0.9999993    1.0000000
## 7   7 0.9999804 0.9999999 0.9999509        1.0000000    1.0000000
## 8   8 0.9999981 1.0000000 0.9999933        1.0000000    1.0000000
## 9   9 0.9999998 1.0000000 0.9999992        1.0000000    1.0000000
## 10 10 1.0000000 1.0000000 0.9999999        1.0000000    1.0000000
```

```
# n = 25, p = 0.1
table(n[1], P[2])
```

```
##      k Binomiala    Normala   Poisson Normala.Corectie Camp.Paulson
## 1    1 0.1223765 0.07947816 0.1358882        0.1298982    0.0241518
## 2    2 0.3062503 0.19901236 0.3208472        0.2865690    0.1911140
## 3    3 0.5309849 0.38907984 0.5366327        0.5000000    0.4453206
## 4    4 0.7307490 0.61092016 0.7254450        0.7134310    0.6760624
## 5    5 0.8683642 0.80098764 0.8576136        0.8701018    0.8341383
## 6    6 0.9448172 0.92052184 0.9347119        0.9545155    0.9862577
## 7    7 0.9800098 0.97569671 0.9732611        0.9878939    0.9958787
## 8    8 0.9936958 0.99438506 0.9901263        0.9975776    0.9988454
## 9    9 0.9982578 0.99902879 0.9966851        0.9996384    0.9996937
## 10  10 0.9995757 0.99987504 0.9989806        0.9999599    0.9998228
```

```
# n = 50, p = 0.05
table(N[2], P[1])
```

```
##      k Binomiala   Normala   Poisson Normala.Corectie Camp.Paulson
## 1    1 0.2794318 0.1651950 0.2872975        0.2582061   0.09933565
## 2    2 0.5405331 0.3728014 0.5438131        0.5000000   0.43052606
## 3    3 0.7604080 0.6271986 0.7575761        0.7417939   0.72090308
## 4    4 0.8963832 0.8348050 0.8911780        0.9028170   0.88629297
## 5    5 0.9622238 0.9476213 0.9579790        0.9742121   0.95986805
## 6    6 0.9882136 0.9884295 0.9858127        0.9952779   0.98738227
## 7    7 0.9968117 0.9982498 0.9957533        0.9994116   0.99639362
## 8    8 0.9992440 0.9998207 0.9988597        0.9999506   0.99904834
## 9    9 0.9998414 0.9999877 0.9997226        0.9999972   0.99976529
## 10  10 0.9999704 0.9999994 0.9999384        0.9999999   0.99994535
```

```
# n = 50, p = 0.1
table(n[2], P[2])
```

```
##      k    Binomiala     Normala      Poisson Normala.Corectie Camp.Paulson
## 1    1 0.0003216881 0.001349898 0.0004993992      0.002303266 2.004564e-06
## 2    2 0.0019448847 0.003830381 0.0027693957      0.006209665 3.071794e-04
## 3    3 0.0078364871 0.009815329 0.0103360507      0.015130140 3.498753e-03
## 4    4 0.0237110827 0.022750132 0.0292526881      0.033376508 1.667608e-02
## 5    5 0.0575768865 0.047790352 0.0670859629      0.066807201 5.038235e-02
## 6    6 0.1171556154 0.091211220 0.1301414209      0.121672505 1.138333e-01
## 7    7 0.2060508618 0.158655254 0.2202206466      0.202328381 2.094965e-01
## 8    8 0.3208738884 0.252492538 0.3328196788      0.308537539 3.308337e-01
## 9    9 0.4512901654 0.369441340 0.4579297145      0.433816167 4.645171e-01
## 10  10 0.5831555123 0.500000000 0.5830397502      0.566183833 4.668155e-01
```

```
# n = 100, p = 0.05
table(N[3], P[1])
```

```
##      k  Binomiala    Normala    Poisson Normala.Corectie Camp.Paulson
## 1    1 0.03708121 0.03322871 0.04042768       0.05414683  0.003982235
## 2    2 0.11826298 0.08433431 0.12465202       0.12567455  0.066857333
## 3    3 0.25783866 0.17939768 0.26502592       0.24564856  0.228261230
## 4    4 0.43598130 0.32317760 0.44049329       0.40927290  0.445213081
## 5    5 0.61599913 0.50000000 0.61596065       0.59072710  0.651075373
## 6    6 0.76601398 0.67682240 0.76218346       0.75435144  0.805516130
## 7    7 0.87203952 0.82060232 0.86662833       0.87432545  0.902629168
## 8    8 0.93691041 0.91566569 0.93190637       0.94585317  0.955672429
## 9    9 0.97181171 0.96677129 0.96817194       0.98052627  0.981455276
## 10  10 0.98852759 0.98910927 0.98630473       0.99419155  0.992805443
```

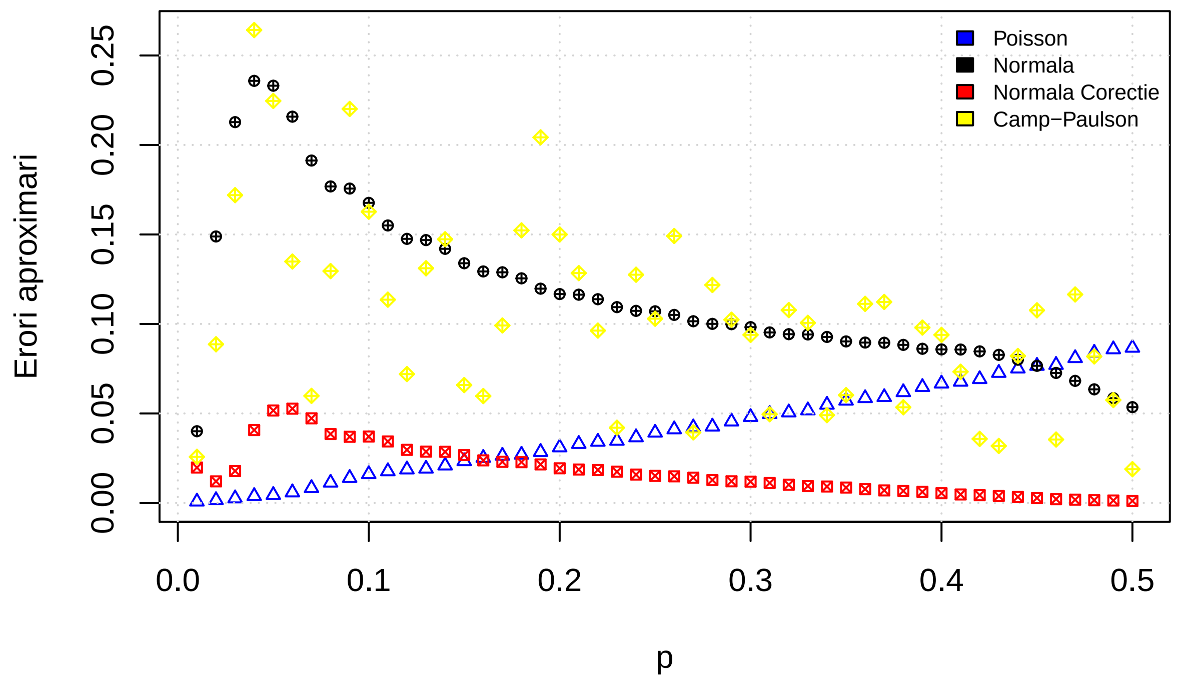
```
# n = 100, p = 0.1
table(n[3], P[2])
```

```
##     k   Binomiala    Normala    Poisson Normala.Corectie Camp.Paulson
## 1   1 0.008724737 0.01148389 0.01127579       0.01935551 0.0002221183
## 2   2 0.035973163 0.03140592 0.04303595       0.04908444 0.0090647127
## 3   3 0.099552825 0.07393839 0.11184961       0.10742347 0.0514287810
## 4   4 0.209051132 0.15065627 0.22367182       0.20414737 0.1466613068
## 5   5 0.357482170 0.26757173 0.36904068       0.33963880 0.2905609749
## 6   6 0.522405546 0.41811249 0.52652362       0.50000000 0.6325970896
## 7   7 0.676857596 0.58188751 0.67275778       0.66036120 0.7752792715
## 8   8 0.801277303 0.73242827 0.79157303       0.79585263 0.8738408958
## 9   9 0.888831912 0.84934373 0.87738405       0.89257653 0.9344219192
## 10 10 0.943310335 0.92606161 0.93316121       0.95091556 0.9432684020
```

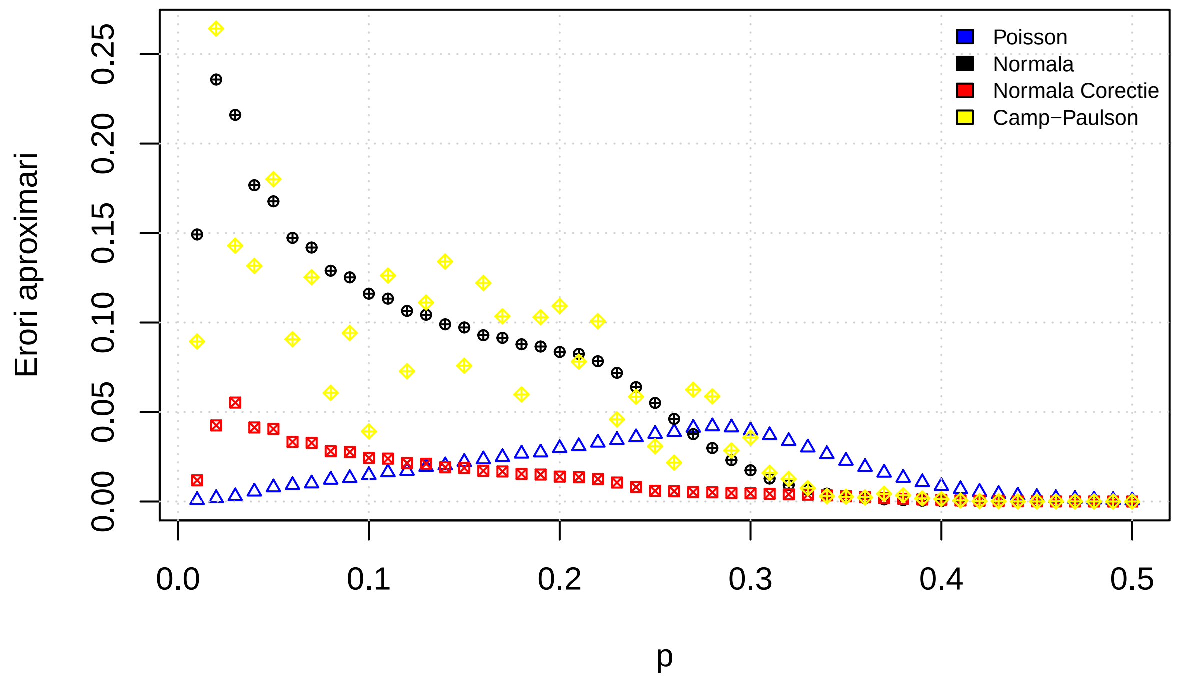
1.5

Cuantificarea acuratetii aproximarilor folosind distanta Kolmogorov

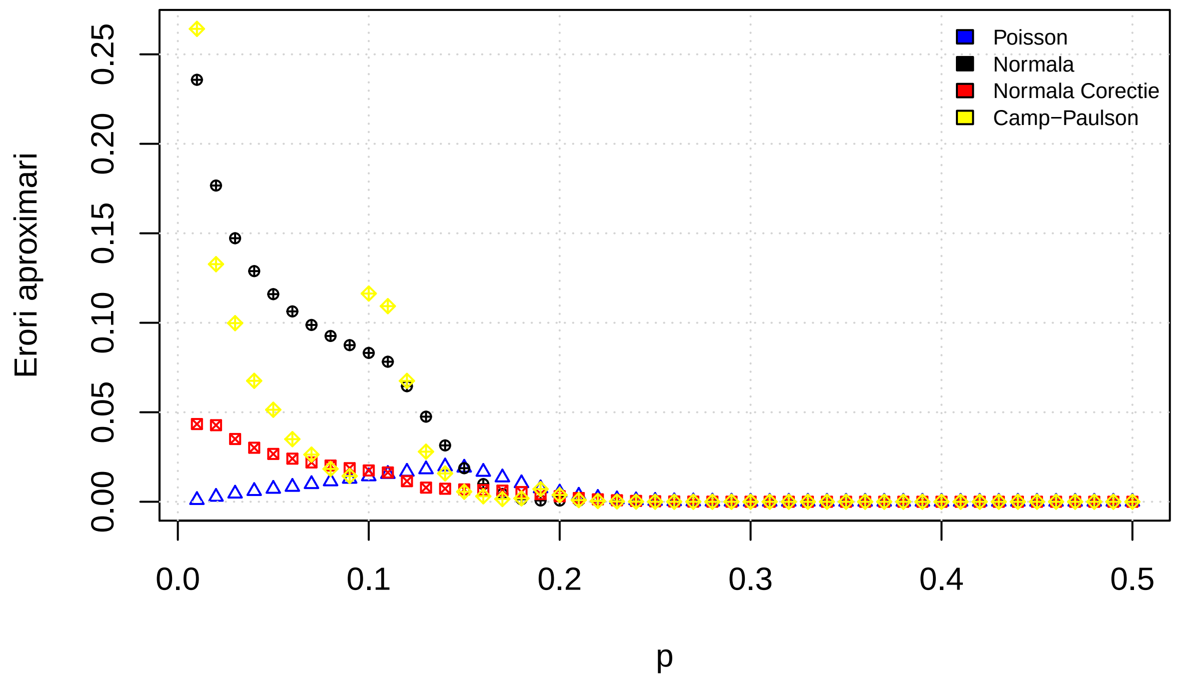
```
P = seq(0.01, 0.5, 0.01)
error = function(n){
  diferentaPoisson  = c()
  diferentaNormala = c()
  diferentaNormalaCorectie = c()
  diferentaCampPaulson = c()
  for(p in P) {
    Bin = pbinom(k, n, p)
    Pois = aproximarePoisson(1, n, p)
    for(i in 2:10)
      Pois = c(Pois, aproximarePoisson(i, n, p))
    Norm = aproximareNormala(n, p)
    NormCorectie = aproximareNormalaCorectie(n, p)
    CampPaulson = aproximareCampPaulson(n, p)
    diferentaPoisson = c(diferentaPoisson, max( abs(Bin - Pois) ))
    diferentaNormala = c(diferentaNormala, max( abs(Bin - Norm) ))
    diferentaNormalaCorectie = c(diferentaNormalaCorectie,
                                 max( abs(Bin - NormCorectie) ))
    diferentaCampPaulson = c(diferentaCampPaulson,
                             max( abs(Bin - CampPaulson) ))
  }
  max_y = max(diferentaPoisson, diferentaNormala, diferentaNormalaCorectie,
              diferentaCampPaulson)
  plot(P, diferentaPoisson, xlim = c(0.01, 0.5), ylim = c(0, max_y),
       xlab = "p", ylab = "Erori aproximari", col = "blue", type = "p",
       pch = 2, cex = 0.7)
  grid()
  points(P, diferentaNormala, col = "black", type = "p", pch = 10, cex = 0.7)
  points(P, diferentaNormalaCorectie, col = "red", type = "p",
         pch = 7, cex = 0.7)
  points(P, diferentaCampPaulson, col = "yellow", type = "p", pch = 9, cex = 0.7)
  legend("topright", legend = c("Poisson", "Normala", "Normala Corectie",
         "Camp-Paulson"), fill = c("blue", "black", "red", "yellow"),
           bty = "n", cex = 0.7)
}
error(25)
```



```
error(50)
```


Poisson
Normala
Normala Corectie
Camp−Paulson
Erori aproximari
p
0.00
0.05
0.10
0.15
0.20
0.25
0.0
0.1
0.2
0.3
0.4
0.5

```
error(100)
```



Dupa cum se putea observa si din tabele anterioare, aproximarea Normala cu Corectie este cea mai precisa, alaturi de aproximarea Camp-Paulson. Aproximarea Poisson este buna pentru $p < 0.05$ si $n > 20$. Iar aproximarea Normala se este buna pe masura ce n si p cresc. Toate tind sa aproximeze mai bine pe masura ce numarul de observatii creste.

Problema 2

2.1

Justificati teoretic ca putem simula un vector (cuplu) aleator repartizat uniform pe patratul [-1, 1] plecand de la doua v.a. independente repartizate uniform pe segmentul [-1, 1].

$$X \sim U([-1,1])$$
$$Y \sim U([-1,1])$$

Cerinta:

$$(X,Y) \sim U([-1,1]^2)$$

$(X,Y) \sim U([-1,1]^2) \Longleftrightarrow f_{(X,Y)} = \frac{1}{A} * 1_{[-1,1]^2}(x,y)$
$A = (1-(-1)) * (1-(-1) = 4$
$\mathbb{P}((X,Y) \in [-1,1]^2) = \mathbb{P}(X \in [-1,1], Y \in [-1,1]) = \mathbb{P}(X \in [-1,1]) * \mathbb{P}(Y \in [-1,1])$
Cum $X \perp Y$
$\mathbb{P}((X,Y) \in [-1,1]^2) = \mathbb{P}(X \in [-1,1]) * \mathbb{P}(Y \in [-1,1]) = \int_{-1}^{1} f_X(x)dx * \int_{-1}^{1} f_Y(y)dy = \int_{-1}^{1}\int_{-1}^{1} f_X(x) * f_Y(y)dxdy$

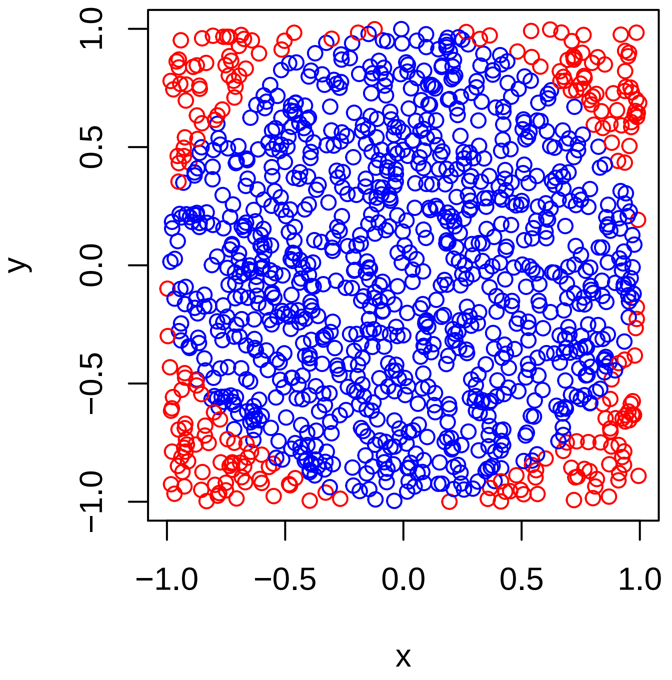
Cum
$\mathbb{P}((X,Y) \in [-1,1]^2) = \int_{-1}^{1}\int_{-1}^{1} f_{(X,Y)}(x,y)dxdy$
$\Rightarrow \int_{-1}^{1}\int_{-1}^{1} f_{(X,Y)}(x,y)dxdy = \int_{-1}^{1}\int_{-1}^{1} f_X(x) * f_Y(y)dxdy \Rightarrow f_X(x) * f_Y(y) = f_{(X,Y)}(x,y)$ Avem
$f_X(x) * f_Y(y) = \frac{1}{2} * 1_{[-1,1]}(x) * \frac{1}{2} * 1_{[-1,1]}(y) = \frac{1}{4} * 1_{[-1,1]^2}(x,y) = \frac{1}{A} * 1_{[-1,1]^2}(x,y)$

2.2

Prin metoda acceptarii si respingerii simulati N = 1000 de puncte independente repartizate uniform pe discul unitate D(1). Reprezentati grafic punctele (Xi, Yi) din interiorul discului unitate cu albastru, si pe celelalte cu rosu

Incepem prin a crea un plot gol. Generam repetitiv observatii pentru o abscisa din intervalul [-1,1] si pentru o ordonata din [-1,1] (ambele repartizate uniform pe intervalele corespunzatoare). Calculam distata de la punctul generat la origine, iar daca aceasta este valida, coloram punctul albastru, daca nu, rosu. Numarul punctelor albastre stim ca va fi in final N.

```r
N=1000
par(pty="s")
plot(0,pch='',ylab='y',xlab='x',xlim = c(-1,1),ylim=c(-1,1))
n=0;
x=0;y=0;
while(n<N)
{
  x=runif(1,min=-1,max = 1)
  y=runif(1,min=-1,max = 1)
  dist=sqrt(x^2+y^2)
  if(dist<=1)
    {points(x,y,col="blue")
    n=n+1}
  else
    points(x,y,col="red")
}
```



2.3

Calculati media aritmetica a distantei care separa cele N puncte de origine. Comparati rezultatul cu media teoritica a variabilei corespunzatoare.

Calculam media aritmetica a distantelor ce separa cele N puncte de origine. Consideram 2 vectori de dimensiune N initializati cu valoarea 0. Generam apoi repetitiv observatii(pana ajung la numarul N cele valide) cu ordonata repartizata uniform in intervalul [-1,1] si abscisa repartizata uniform tot in intervalul [-1,1]. Calculam distanta de la punctul generat la originea (0,0), iar daca punctul este valid(se afla in interiorul Cercului de raza 1 si centru (0,0)),ii adaugam coordonatele in vectorii xi, respectiv yi. Apoi, reiteram prin cei doi vectori(deoarece avem doar puncte valide) si calculam media distantei de la punctele corespunzatoare la origine.

```
N=1000
n=0;
x=0;y=0;
xi=rep(N,0);
yi=rep(N,0);
med=0;
while(n<N)
{
  x=runif(1,min=-1,max = 1)
  y=runif(1,min=-1,max = 1)
  dist=sqrt(x^2+y^2)
  if(dist<=1)
  {n=n+1
```

```
    xi[n]=x;
    yi[n]=y;}
}
for(i in 1:N)
{
  dist=sqrt(xi[i]^2+yi[i]^2)
  med=med+dist;
}
med=med/N;
med
```

```
## [1] 0.6760978
```

Calculam in continuare media teoretica:
Stim ca $t : \mathbb{R}^2 \to \mathbb{R}, t - continua$ si $Z = t(X, Y)$ atunci $\mathbb{E}[Z] = \iint_{\mathbb{R}^2} t(x, y) * f_{(X,Y)}(x, y)dxdy$
La noi $t(x, y) = \sqrt{x^2 + y^2}$
$\mathbb{E}[Z] = \iint_{\mathbb{R}^2} t(x, y) * f_{(X,Y)}(x, y)dxdy = \iint_{D(1)} \sqrt{x^2 + y^2} * \frac{1}{\pi} * 1_{D(1)}(x, y)dydx$
Deoarece
$0 \le x^2 + y^2 \le 1$ si $0 \le x \le 1$ si $0 \le y \le 1 \Rightarrow \iint_{D(1)} \sqrt{x^2 + y^2} * \frac{1}{\pi} * 1_{D(1)}(x, y)dydx = \frac{1}{\pi} \int_{-1}^{1} \int_{-\sqrt{1-x^2}}^{\sqrt{1-x^2}} \sqrt{x^2 + y^2}dydx$
Vom folosi schimbarea de variabila $x = rcos(\theta)$ si $y = rsin(\theta), r > 0$
Cum avem $1_{D(1)}(x, y) = 1_{\{x^2+y^2 \le 1\}}(x, y) = 1_{\{r^2cos(\theta)^2+r^2sin(\theta)^2 \le 1\}}(r, \theta) = 1_{\{r^2 \le 1\}}(r, \theta) = 1_{[0,1]}(r) * 1_{[0,2\pi]}(\theta)$
Fie $G \subseteq \mathbb{R}^2$ multime deschisa si $g : G \to \mathbb{R}^2, g \in \mathcal{C}^1$ g-injectiva si Jacobianul lui g nu se anuleaza. Atunci pentru $f * 1_{g(G)}$ este pozitiva si integrabila $\iint_{g(G)} f(x, y)dxdy = \iint_G f(g(r, \theta))|detJ_g|drd\theta$
Cum $G = [0, 1] \times [0, 2\pi]$ si $g(r, \theta) = (x, y) = (r\cos(\theta), r\sin(\theta))$ $J_g = \left(\frac{\partial g_i}{\partial x_j}\right)_{i,j} = \begin{pmatrix} \frac{\partial g_1}{\partial r} & \frac{\partial g_1}{\partial \theta} \\ \frac{\partial g_2}{\partial r} & \frac{\partial g_2}{\partial \theta} \end{pmatrix}$
$detJ_g = det \begin{pmatrix} \cos(\theta) & -r\sin(\theta) \\ \sin(\theta) & r\cos(\theta) \end{pmatrix} = r$
$\iint_{D(1)} \sqrt{x^2 + y^2} * \frac{1}{\pi} * 1_{D(1)}(x, y)dydx = \frac{1}{\pi} \int_{-1}^{1} \int_{-\sqrt{1-x^2}}^{\sqrt{1-x^2}} \sqrt{x^2 + y^2}dydx = \frac{1}{\pi} \int_0^{2\pi} \int_0^1 \sqrt{r^2}rdrd\theta$
$\int_0^1 \sqrt{r^2}rdr = \frac{r^3}{3}|_0^1 = \frac{1}{3}$
$\frac{1}{\pi} \int_0^{2\pi} \frac{1}{3}d\theta = \frac{1}{3\pi} \int_0^{2\pi} d\theta = \frac{2\pi}{3\pi} = \frac{2}{3} = 0.6(7)$

2.4

4. Plecand de la densitatea cuplului (X, Y), gasiti densitatea v.a. R si θ.

$$\iint_{D(1)} f_{(X,Y)}(x, y)dxdy$$

Vom folosi schimbarea de variabila $x = rcos(\theta)$ si $y = rsin(\theta), r > 0$
Cum avem $1_{D(1)}(x, y) = 1_{\{x^2+y^2 \le 1\}}(x, y) = 1_{\{r^2cos(\theta)^2+r^2sin(\theta)^2 \le 1\}}(r, \theta) = 1_{\{r^2 \le 1\}}(r, \theta) = 1_{[0,1]}(r) * 1_{[0,2\pi]}(\theta)$
Fie $G \subseteq \mathbb{R}^2$ multime deschisa si $g : G \to \mathbb{R}^2, g \in \mathcal{C}^1$ g-injectiva si Jacobianul lui g nu se anuleaza. Atunci pentru $f * 1_{g(G)}$ este pozitiva si integrabila $\iint_{g(G)} f(x, y)dxdy = \iint_G f(g(r, \theta))|detJ_g|drd\theta$
Cum $G = [0, 1] \times [0, 2\pi]$ si $g(r, \theta) = (x, y) = (r\cos(\theta), r\sin(\theta))$ $J_g = \left(\frac{\partial g_i}{\partial x_j}\right)_{i,j} = \begin{pmatrix} \frac{\partial g_1}{\partial r} & \frac{\partial g_1}{\partial \theta} \\ \frac{\partial g_2}{\partial r} & \frac{\partial g_2}{\partial \theta} \end{pmatrix}$
$detJ_g = det \begin{pmatrix} \cos(\theta) & -r\sin(\theta) \\ \sin(\theta) & r\cos(\theta) \end{pmatrix} = r$

$$\iint_{D(1)} f_{(X,Y)}(x, y)dxdy = \iint_G f_{(X,Y)}(g(r, \theta))rdrd\theta$$

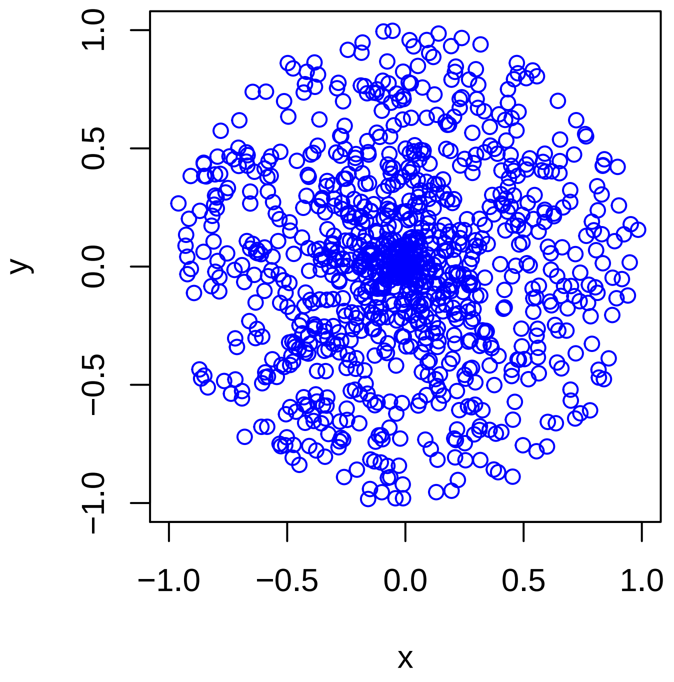
$f_R(r) = \int r f_{(X,Y)}(g(r,\theta))1_G(r,\theta)d\theta = \int_0^{2\pi} \frac{1}{\pi} r 1_{[0,1]}(r)d\theta = r\frac{1}{\pi}1_{[0,1]}(r)\int_0^{2\pi} d\theta = 2r1_{[0,1]}(r)$
$f_\theta(\theta) = \int r f_{(X,Y)}(g(r,\theta))1_G(r,\theta)dr = \int_0^1 \frac{1}{\pi} r 1_{[0,2\pi]}(\theta)dr = \frac{1}{\pi}1_{[0,2\pi]}(\theta)\int_0^1 rdr = \frac{1}{\pi}1_{[0,2\pi]}(\theta)\frac{r^2}{2}|_0^1 = \frac{1}{2\pi}1_{[0,2\pi]}(\theta)$

2.5

Simulati N = 1000 de puncte prin aceasta metoda si ilustrati grafic aceste puncte (incluzand conturul cercului).

Avem la inceput un plot gol. Consideram N observatii, coordonatele polare fiind distribuite fiecare uniform, raza pe intervalul [0,1],iar unghiul polar pe intervalul [0,2pi]. Adaugam punctele generate graficului initial si observam ca toate observatiile se afla in interiorul cercului de centru (0,0) si raza 1.

```
N=1000
par(pty="s")
plot(0,pch='',ylab='y',xlab='x',xlim = c(-1,1),ylim=c(-1,1))
for(i in 1:N)
{
  r=runif(1,min=0,max=1)
  te=runif(1,min=0,max=2*pi)
  points(r*cos(te),r*sin(te),col="blue")
}
```



1.0
0.5
0.0
−0.5
−1.0
y
−1.0
−0.5
0.0
0.5
1.0
x