Tema1.R - Probabilitati si Statistica

Student: Albastroiu Dragos, grupa 251

Incarc pachetele discreteRV si prob, functia suppressMessages este folosita pentru a nu primi output atunci cand pachetul a fost incarcat

```
suppressMessages(library(discreteRV, quietly = T))
suppressMessages(library(prob, quietly = T))
suppressMessages(library(MASS, quietly = T))
```

Problema 1

Problema 1 (scris)

Instantiam cele doua variabile aleatoare prin functia RV(outcomes, probs)

```
(X <-  RV(c(2,3),c(1/5,4/5)))
```

```
## Random variable with 2 outcomes
##
## Outcomes   2   3
## Probs    1/5 4/5
```

```
(Y <-  RV(c(-3,-2),c(4/5,1/5)))
```

```
## Random variable with 2 outcomes
##
## Outcomes  -3  -2
## Probs    4/5 1/5
```

Putem sa aplicam operatiile matematice direct in R, fara sa apelam alte functii

```
(X3 <- 3*X)
```

```
## Random variable with 2 outcomes
##
## Outcomes   6   9
## Probs    1/5 4/5
```

```
(Xem1 <- fractions(X^-1))
```

```
## Random variable with 2 outcomes
##
## Outcomes 1/2 1/3
## Probs    1/5 4/5
```

```
(Xcos <- fractions(cos((pi/2)*X)))
```

```
## Random variable with 2 outcomes
##
## Outcomes  -1   0
## Probs    1/5 4/5
```

```
(Ye2 <- Y^2)
```

```
## Random variable with 2 outcomes
##
## Outcomes   4   9
## Probs    1/5 4/5
```

```
(Yp3 <- Y+3)
```

```
## Random variable with 2 outcomes
##
## Outcomes   0   1
## Probs    4/5 1/5
```

Rezultatele au dat la fel ca pe foaie, cu precizarea ca am folosit functia fractions pentru a transforma numerele decimale in fractii si pentru a scapa de eroarea de calcul a cosinusului, unde `-1.83690953073357e-16` este de fapt 0

Problema 2 (scris)

```
(X2pY3 <- 2*X+3*Y)
```

```
## Random variable with 4 outcomes
##
## Outcomes    -5    -3    -2     0
## Probs     4/25 16/25  1/25  4/25
```

```
(X3mY <- 3*X-Y)
```

```
## Random variable with 4 outcomes
##
## Outcomes     8     9    11    12
## Probs     1/25  4/25  4/25 16/25
```

```
(Xe2Ye3 <- X^2 * Y^3)
```

```
## Random variable with 4 outcomes
##
## Outcomes 4,-27 9,-27  4,-8  9,-8
## Probs     4/25 16/25  1/25  4/25
```

Rezultatele sunt la fel, o limitare a pachetului discreteRV este ca atunci cand inmultim 2 v.a. la afisare o sa avem valorile despartite cu , nu inmultite

```
(Xe2Ye3 <- RV(c(4*-27,9*-27,4*-8,9*-8), c(4/25,16/25,1/25,4/25)))
```

```
## Random variable with 4 outcomes
##
## Outcomes  -243  -108   -72   -32
## Probs    16/25  4/25  4/25  1/25
```

Problema 3 (scris)

R nu are suport nativ pentru expresii simbolice, pentru a nu folosi un alt pachet am ales sa calculez valoarea lui p folosind polyroot, conform conditiilor scrise pe foaie

In cazul nostru polyroot o sa ne intoarca 2 radacini (ecuatie de gradul 2) deci trebuie sa o alegem pe cea pozitiva

```
(radacini <- polyroot(c(0.1*0.2+0.02-1*0.2,0,1)))
```

```
## [1]  0.4+0i -0.4-0i
```

```
(radaciniRe <- Re(radacini))
```

```
## [1]  0.4 -0.4
```

```
(radaciniIm <- Im(radacini))
```

```
## [1]  1.29247e-26 -1.29247e-26
```

Vedem ca radaciniIm primeste valorile foarte apropiate de 0, chiar daca rezultatul initial a dat 0 Avem nevoie de o functie care sa ne zica daca o valoare este foarte apropiata de 0 Am vazut online ca exista functii gen is.zero sau isZero, doar ca eu nu le am in pachetul de baza

```
almostEqual <- function(x, y, tolerance=1e-8) {
  diff <- abs(x - y)
  mag <- pmax(abs(x), abs(y))
  ifelse(mag > tolerance, diff/mag <= tolerance, diff <= tolerance)
}
```

Folosim functia stopifnot pentru a impune conditiile necesare pentru v.a.

```
stopifnot(all(almostEqual(radaciniIm, 0)))
(p <- radaciniRe[radaciniRe>0])
```

```
## [1] 0.4
```

```
stopifnot(length(p) == 1)
(q <- 1-p)
```

```
## [1] 0.6
```

```
(Prob3X <- RV(c(1,2),c(p,q)))
```

```
## Random variable with 2 outcomes
##
## Outcomes   1   2
## Probs    2/5 3/5
```

```
(Prob3Y <- RV(c(3,9),c(0.1, (p^2+0.02)/0.2)))
```

```
## Random variable with 2 outcomes
##
## Outcomes    3    9
## Probs    1/10 9/10
```

Problema 4 (scris)

O sa definesc functia myCondP care calculeaza probabilitatea conditionata pentru 2 v.a. independente

```
myCondP <- function(X, Y)
{
  py <- P(Y)
  if (py==0) return(NaN)
  return((P(X)*P(Y))/P(Y))
}

(P(2*X+3*Y > 1))
```

```
## [1] 0
```

```
(myP <- myCondP(2*X+3*Y > 1, X>0))
```

```
## [1] 0
```

```
(pachetP <- P(2*X+3*Y > 1 | X>0))
```

```
## [1] 0
```

Cele doua valori sunt egale, primul eveniment are p=0 sansa de aparitie deci indiferent de ce ar fi conditionat acesta nu s-ar putea intampla

```
(myP <- myCondP(2*X+3*Y < 3, Y< -2))
```

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

```
(pachetP <- P(2*X+3*Y < 3 | Y < -2))
```

```
## [1] 0.25
```

Observam ca raspunsul a dat diferit in acest caz. Pachetul discreteRV are probleme atunci cand in dreapta semnului este un numar negativ. Putem sa rescriem Y < -2 ca -Y > 2, dar trebuie sa folosim repartitia comuna a celor doua v.a. pentru a obtine valoarea buna

```
negY <- Y*(-1)

produsCartezian <- expand.grid(probs(X2pY3),probs(negY))
jointProbs <- produsCartezian$Var1 * produsCartezian$Var2
(jointNegYsiX2pY3 <- jointRV(list(outcomes(negY), outcomes(X2pY3)), probs=jointProbs))
```

```
## Random variable with 8 outcomes
##
## Outcomes   2,-5   2,-3   2,-2   3,-5    2,0   3,-3   3,-2    3,0
## Probs    16/125 64/125  4/125  4/125 16/125 16/125  1/125  4/125
```

```
(margNegY <- discreteRV::marginal(jointNegYsiX2pY3, 1))
```

```
## Random variable with 2 outcomes
##
## Outcomes   2   3
## Probs    4/5 1/5
```

```
(margX2pY3 <- discreteRV::marginal(jointNegYsiX2pY3, 2))
```

```
## Random variable with 4 outcomes
##
## Outcomes    -5    -3    -2     0
## Probs     4/25 16/25  1/25  4/25
```

```
(myP <- myCondP(2*X+3*Y < 3, Y< -2))
```

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

```
(pachetP <- P(margX2pY3 < 3 | margNegY > 2))
```

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

Rezultatul este acum corect

```
(P(X^2*Y^3 > 3))
```

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

Observam ca P(X^2*Y^3 > 3) a dat gresit deoarece discreteRV nu a inmultit valorile posibile, o sa folosim variabila Xe2Ye3 care contine v.a. in forma buna

```
(P(Xe2Ye3 > 3))
```

```
## [1] 0
```

```
(P(Xe2Ye3 <= 3))
```

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

Pentru ultimul subpunct trebuie sa folosim repartitiile marginale din repartitia comuna a 2X+3Y si 3X-Y Deoarece repartitia comuna are 4 linii si 4 coloane, am decis sa o construiesc prin cod

```
produsCartezian <- expand.grid(probs(X3mY),probs(X2pY3))
jointProbs <- produsCartezian$Var1 * produsCartezian$Var2
(jointX3mYsiX2pY3 <- jointRV(list(outcomes(X2pY3), outcomes(X3mY)), probs=jointProbs))
```

```
## Random variable with 16 outcomes
##
## Outcomes    -5,8    -5,9   -5,11   -5,12    -3,8    -3,9   -3,11   -3,12    -2,8    -2,9   -2,11   -
## Probs      4/625  16/625  16/625  64/625  16/625  64/625  64/625 256/625   1/625   4/625   4/625  16/
##
## Displaying first 12 outcomes
```

```
(margX2pY3 <- discreteRV::marginal(jointX3mYsiX2pY3, 1))
```

```
## Random variable with 4 outcomes
##
## Outcomes    -5    -3    -2     0
## Probs     4/25 16/25  1/25  4/25
```

```
(margX3mY <- discreteRV::marginal(jointX3mYsiX2pY3, 2))
```

```
## Random variable with 4 outcomes
##
## Outcomes     8     9    11    12
## Probs     1/25  4/25  4/25 16/25
```

```
(P(margX2pY3 < margX3mY))
```

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

Cred ca pentru acest subpunct merge si daca le scadem

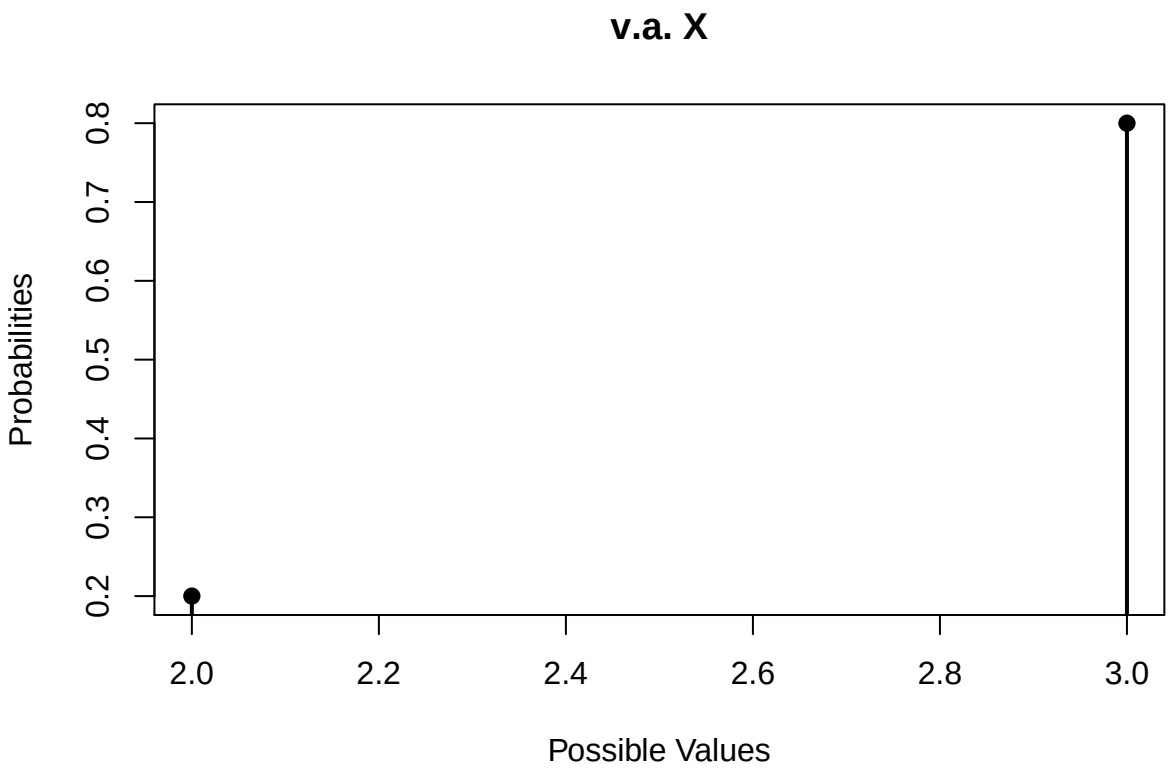
```
(P(X2pY3-X3mY < 0))
```

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

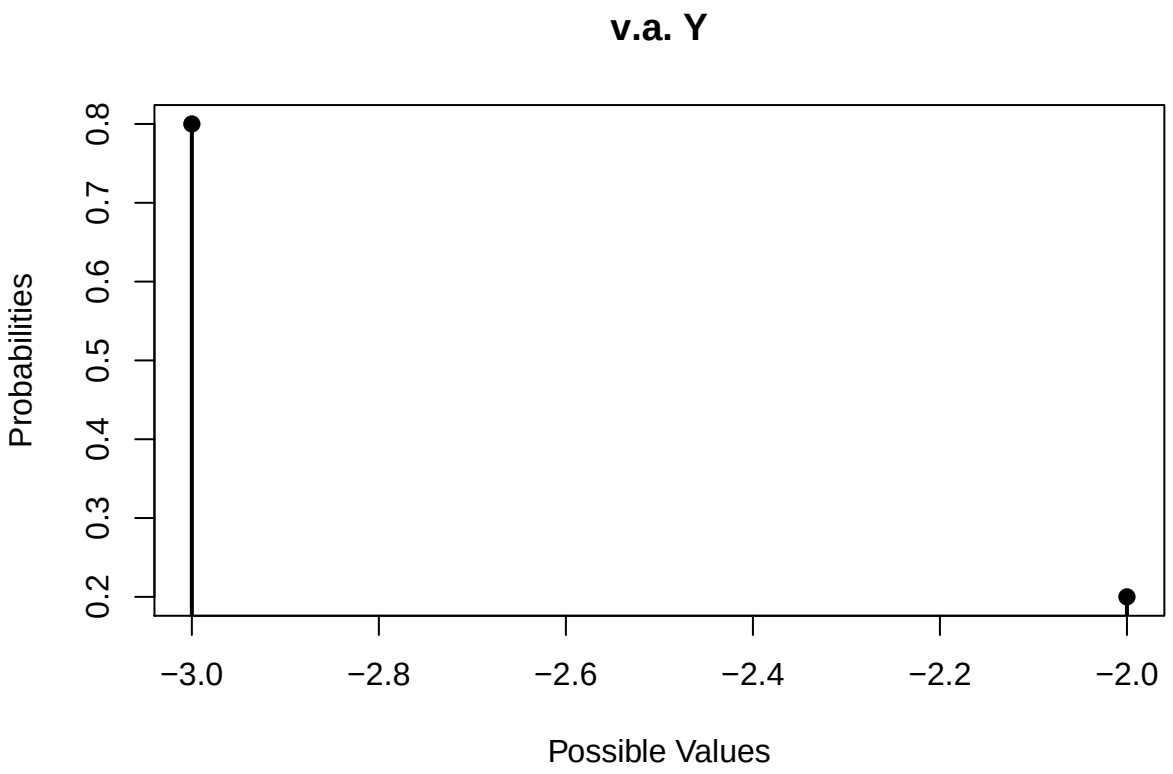
Avem acelasi rezultat

Problema 2

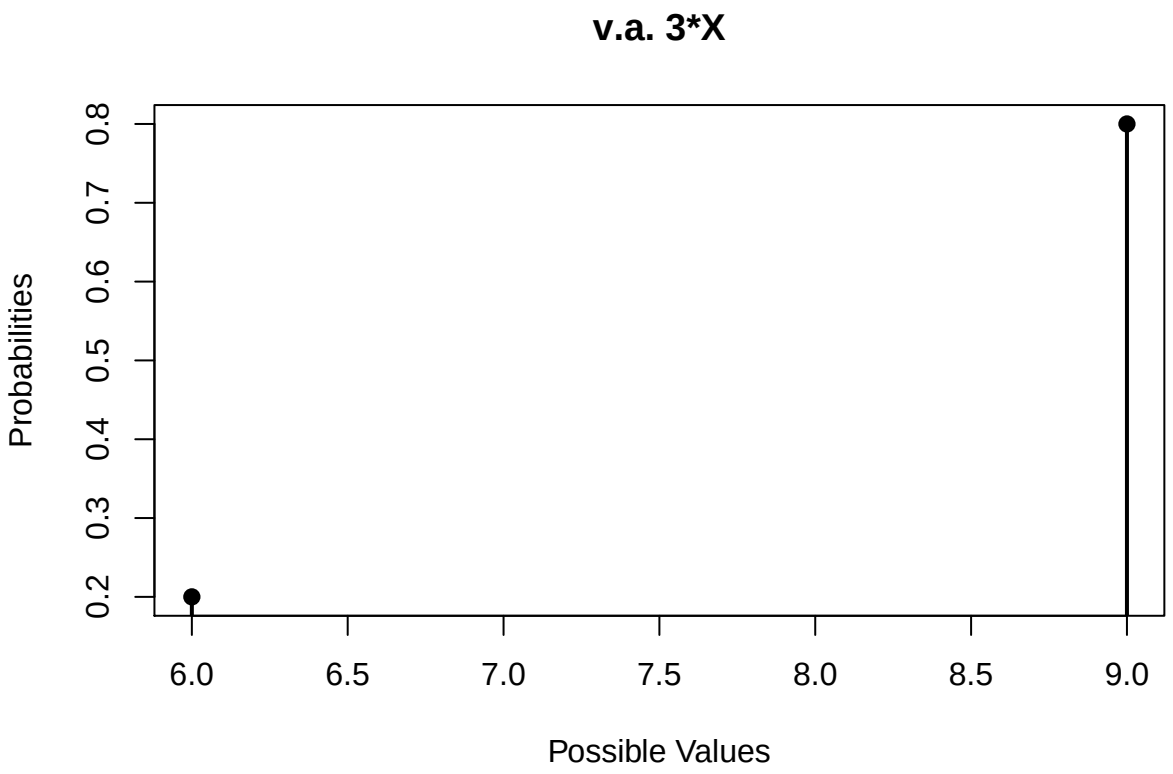
```
plot(X, main='v.a. X')
```



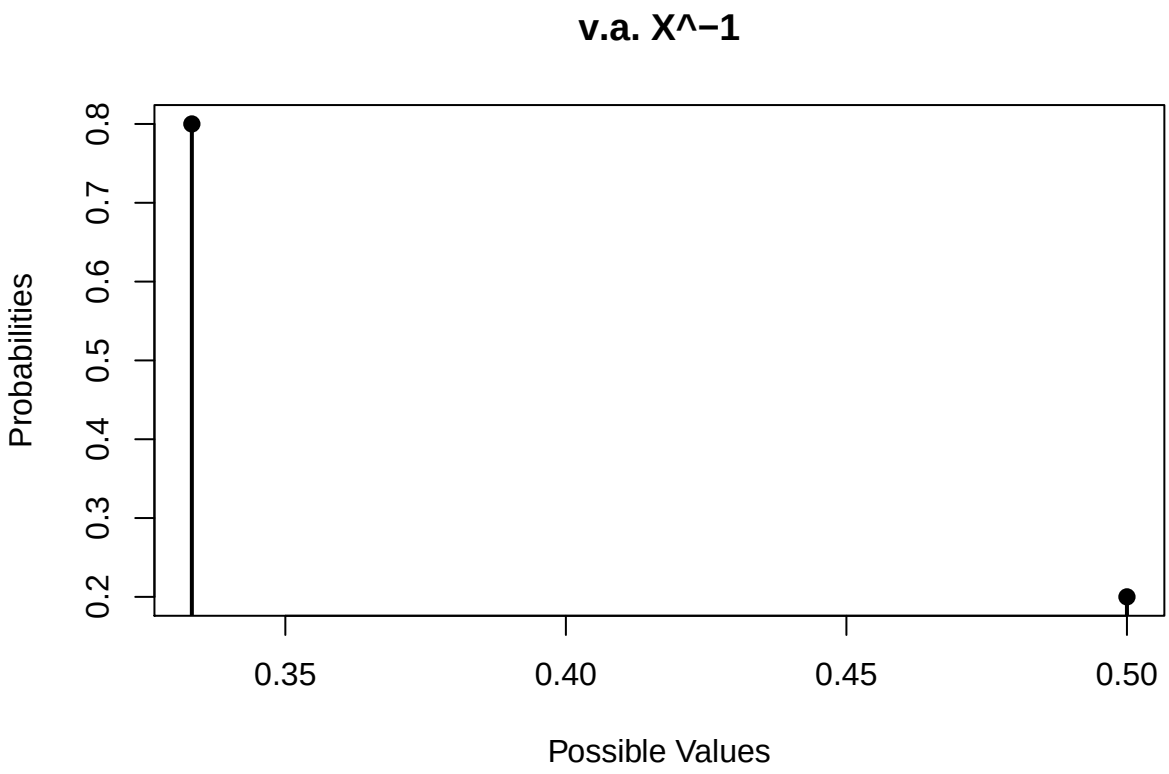
```r
plot(Y, main='v.a. Y')
```



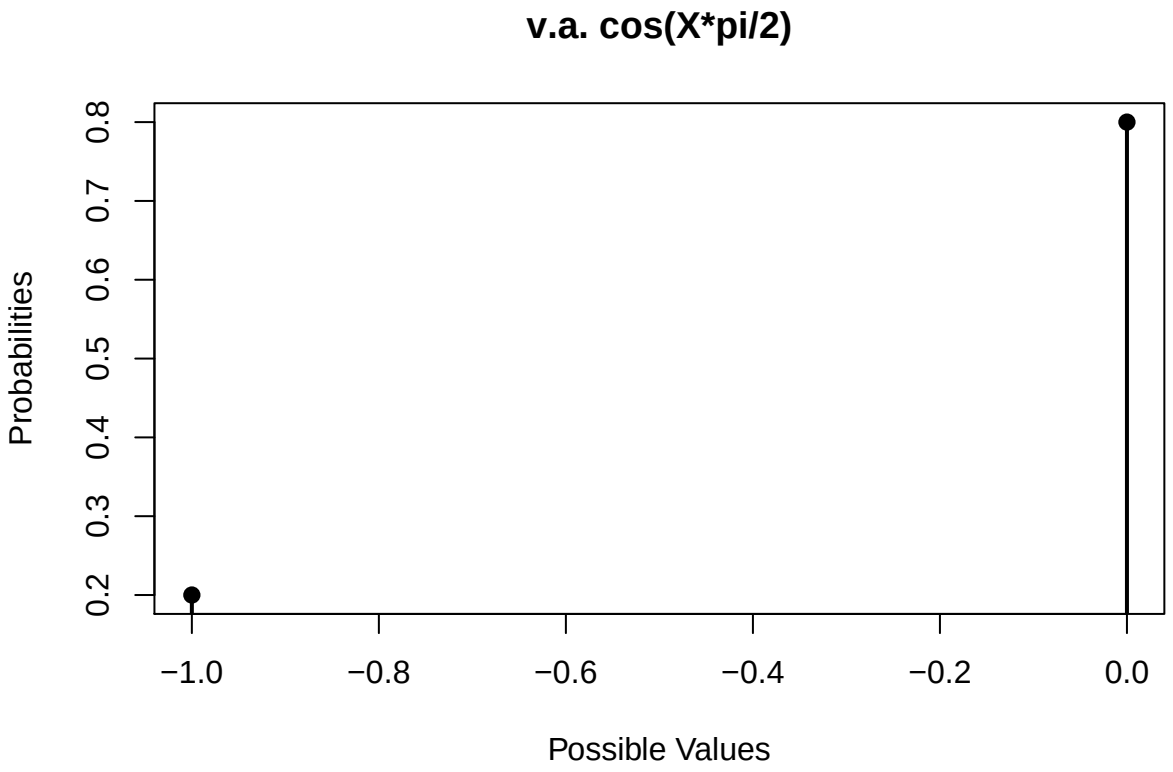
```r
plot(X3, main='v.a. 3*X')
```



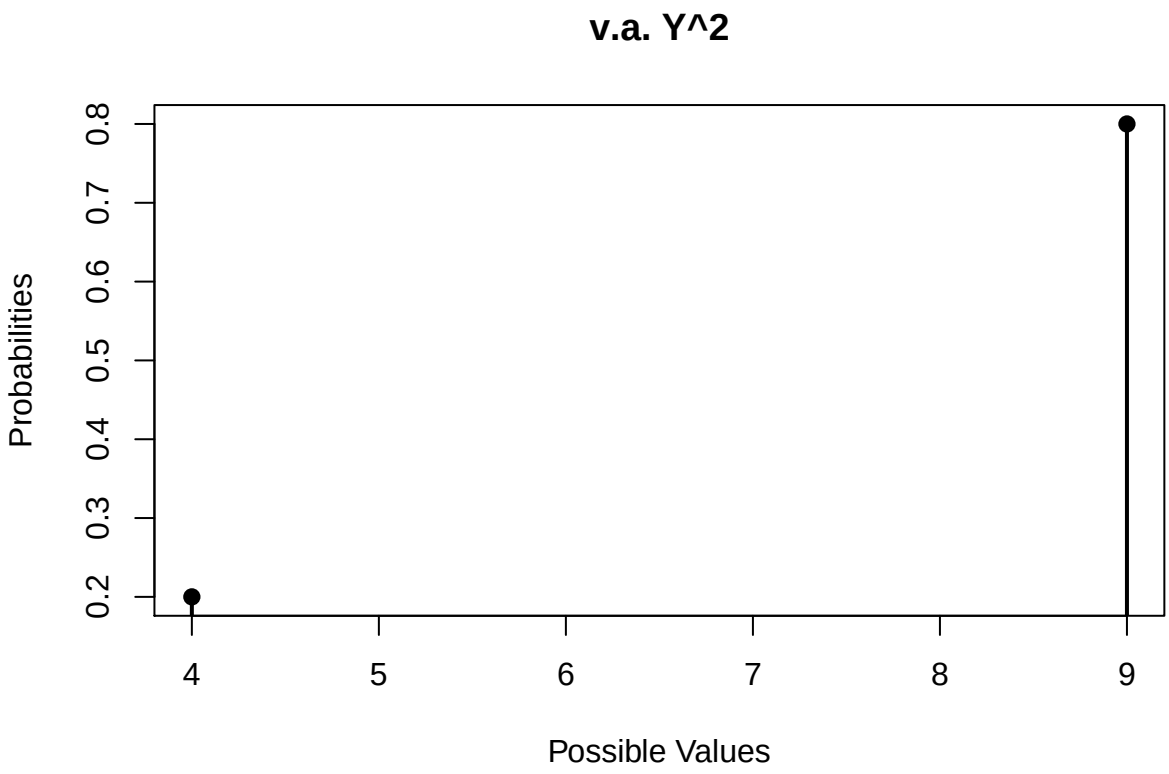
```r
plot(Xem1, main='v.a. X^-1')
```



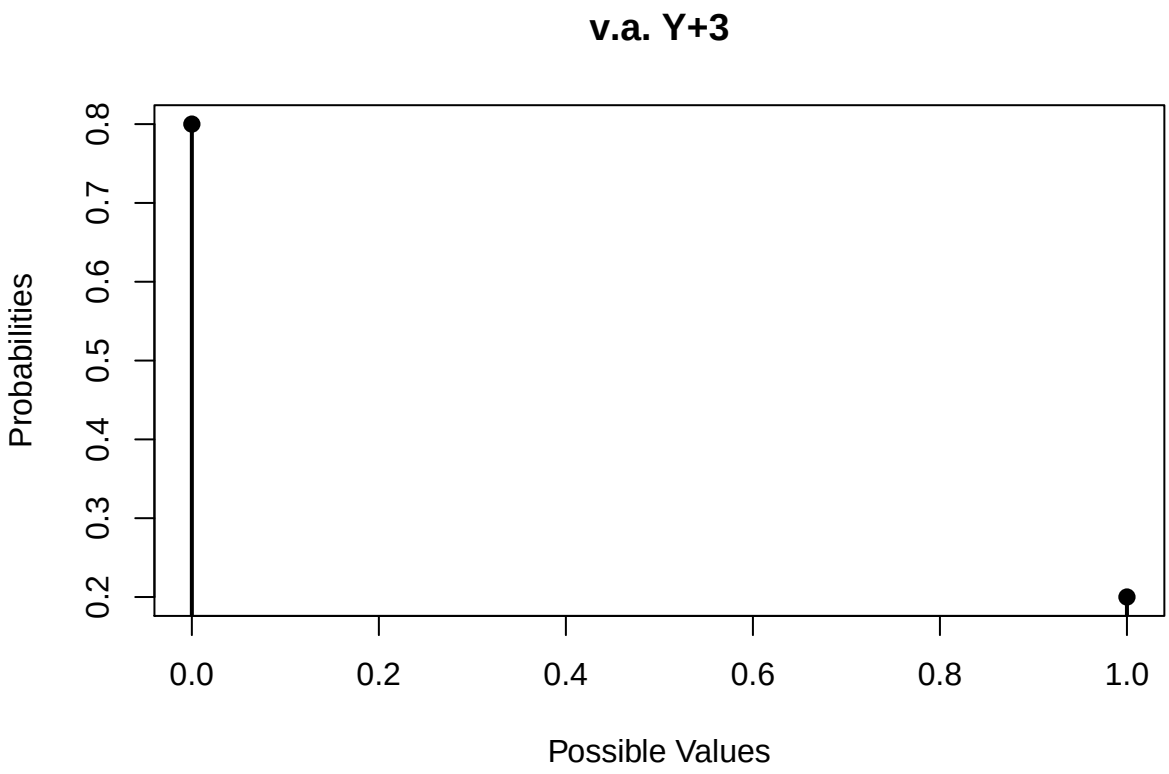
```r
plot(Xcos, main='v.a. cos(X*pi/2)')
```



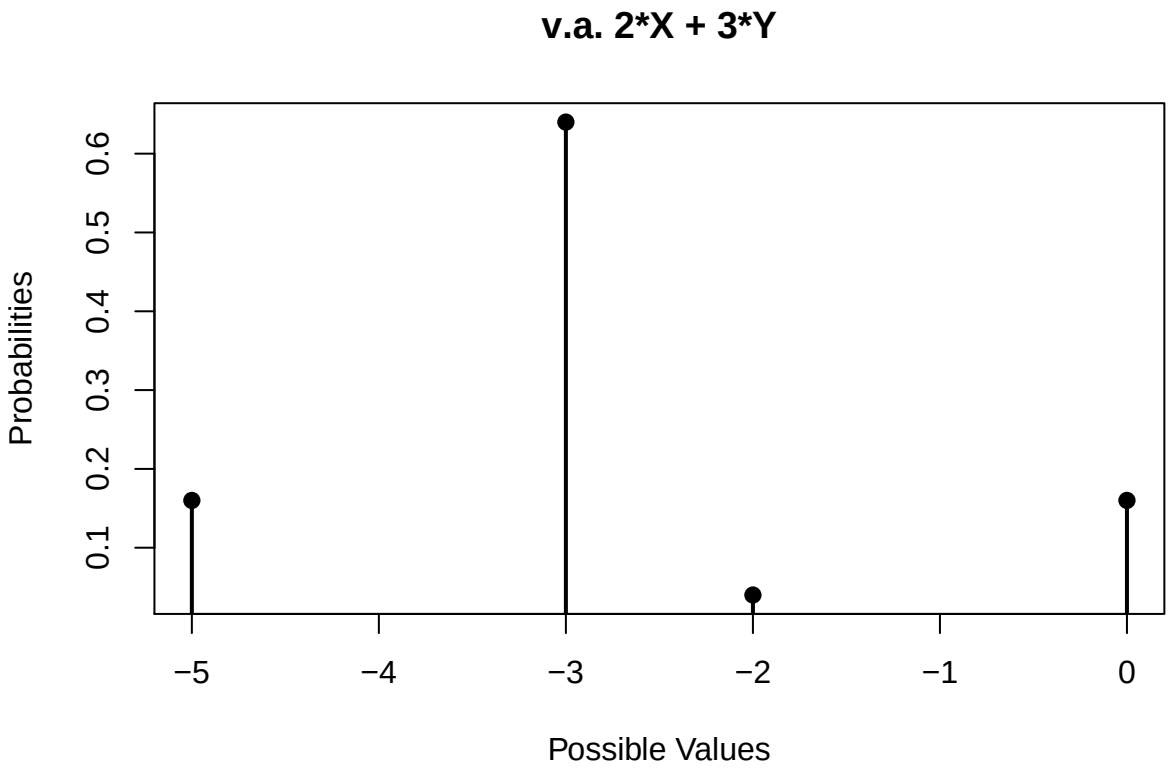
```r
plot(Ye2, main='v.a. Y^2')
```



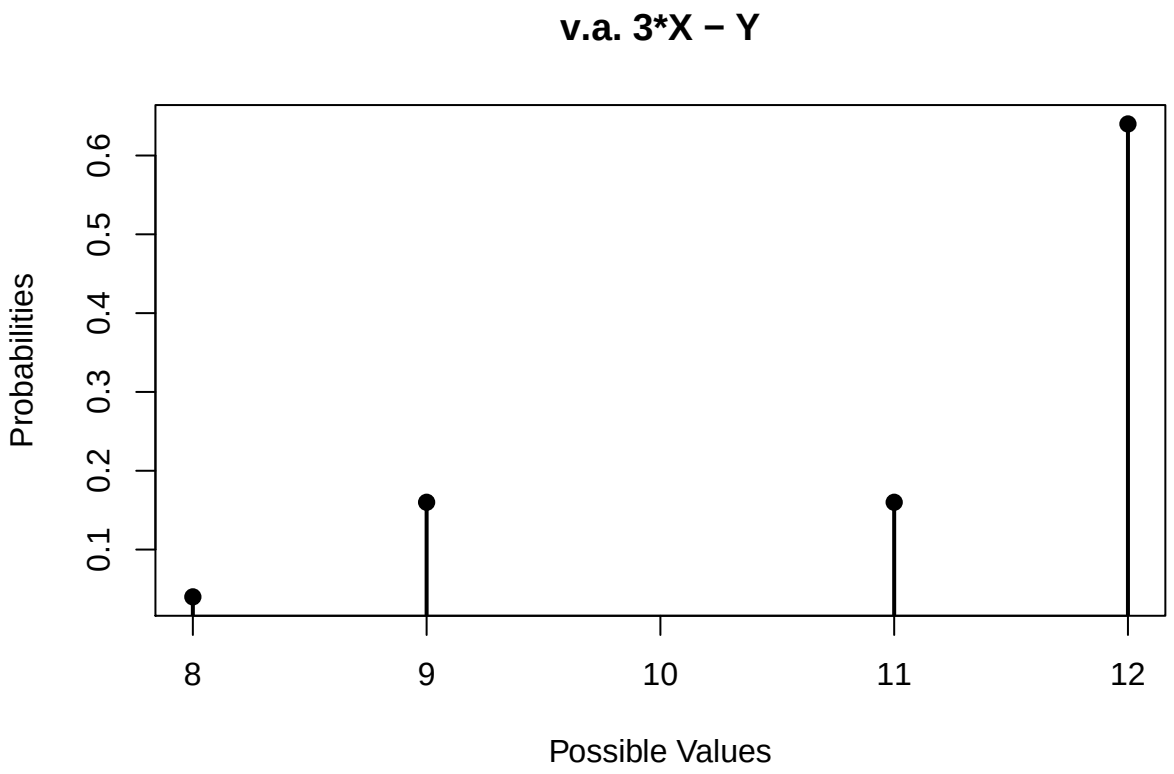
```r
plot(Yp3, main='v.a. Y+3')
```



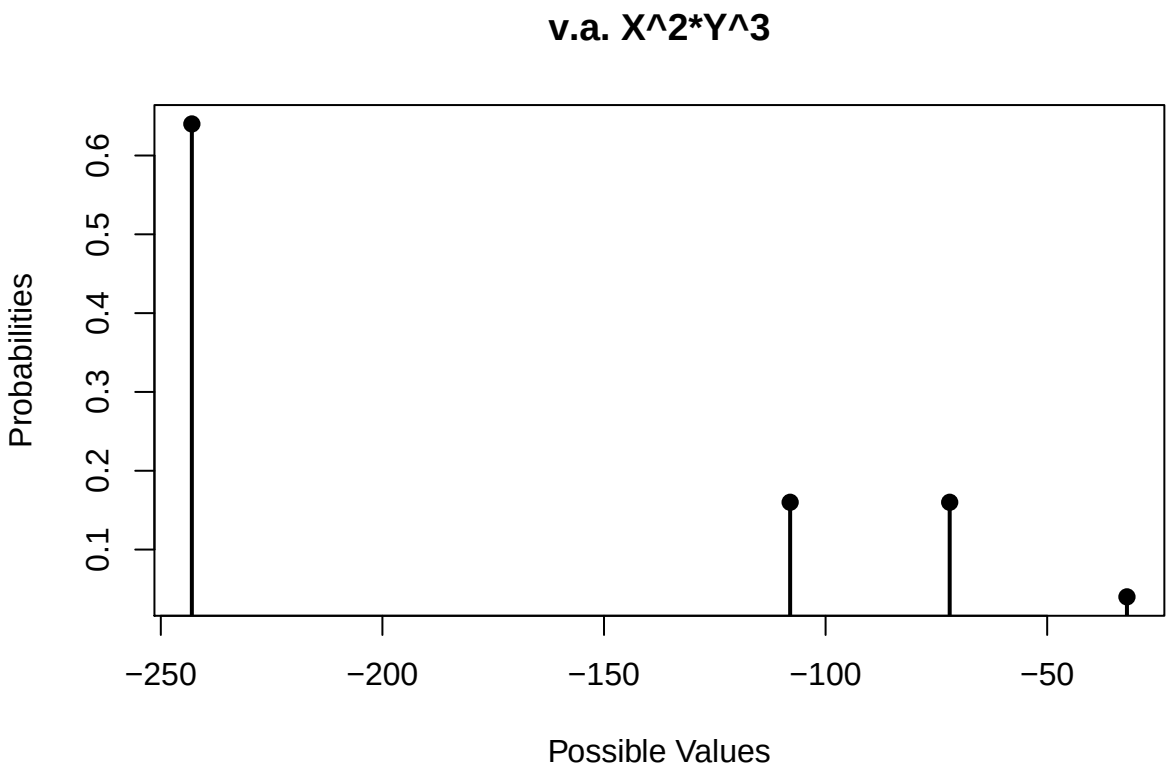
```r
plot(X2pY3, main='v.a. 2*X + 3*Y')
```



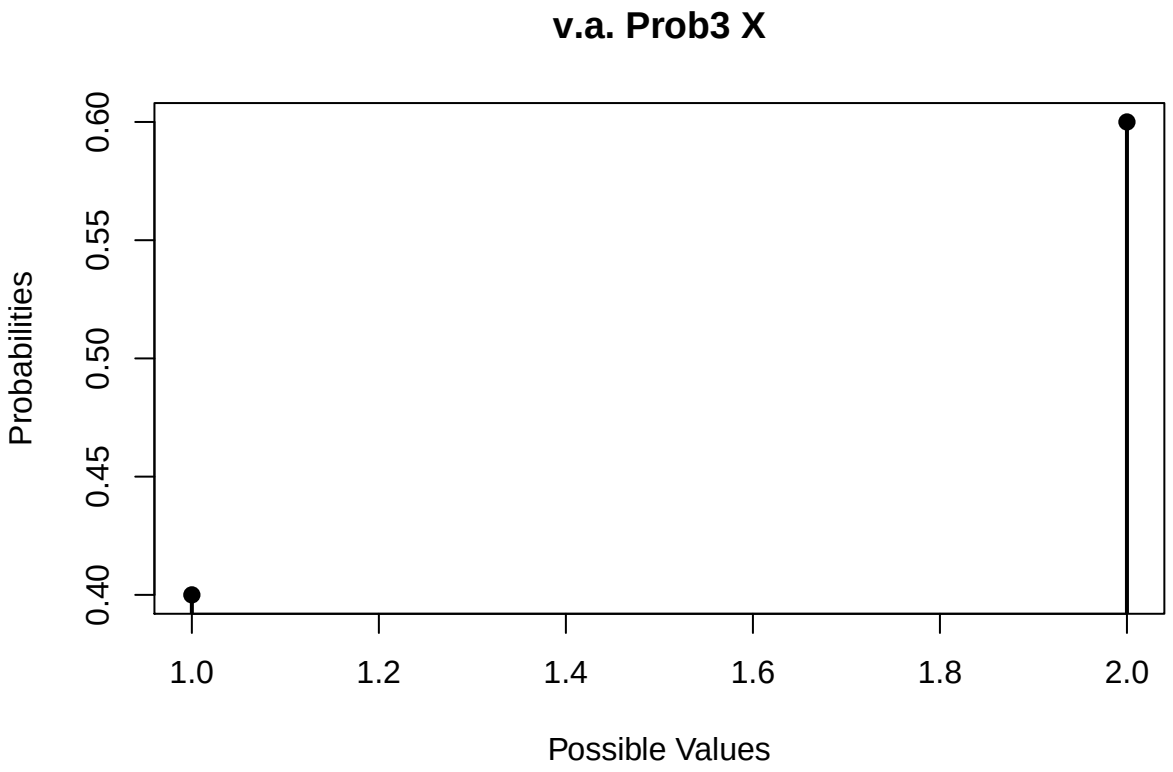
```r
plot(X3mY, main='v.a. 3*X - Y')
```



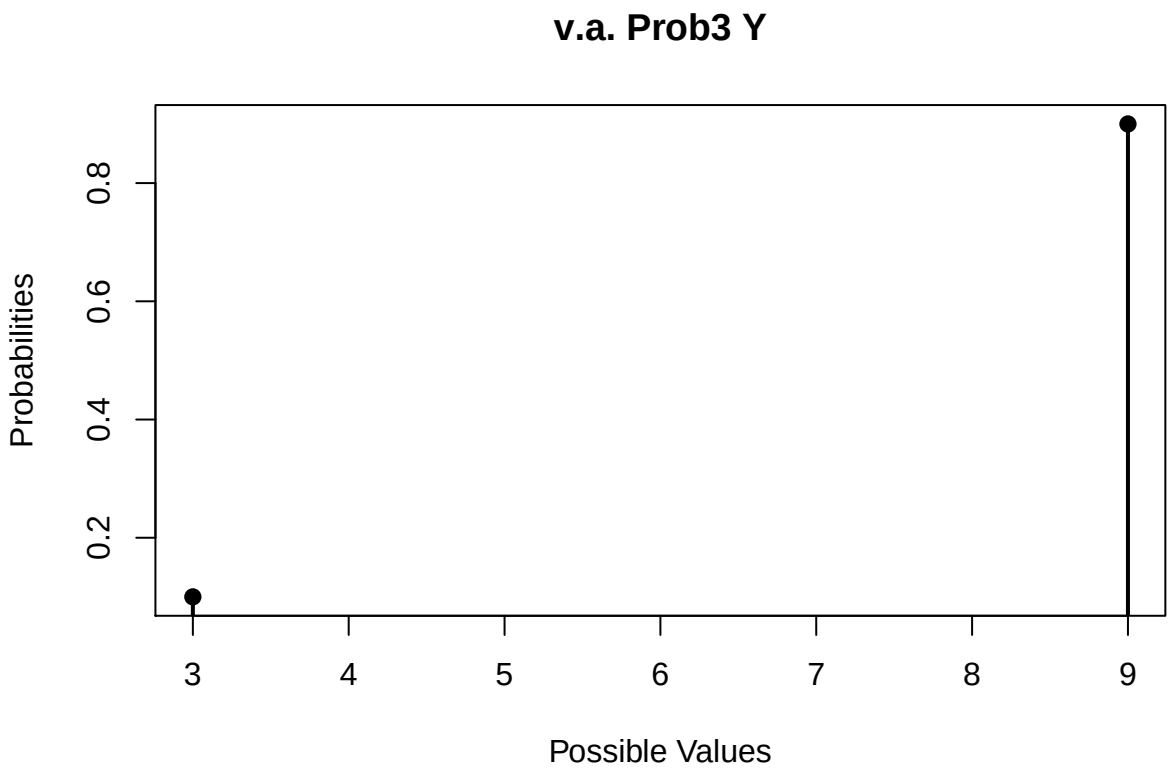
```r
plot(Xe2Ye3, main='v.a. X^2*Y^3')
```



```r
plot(Prob3X, main='v.a. Prob3 X')
```



```r
plot(Prob3Y, main='v.a. Prob3 Y')
```

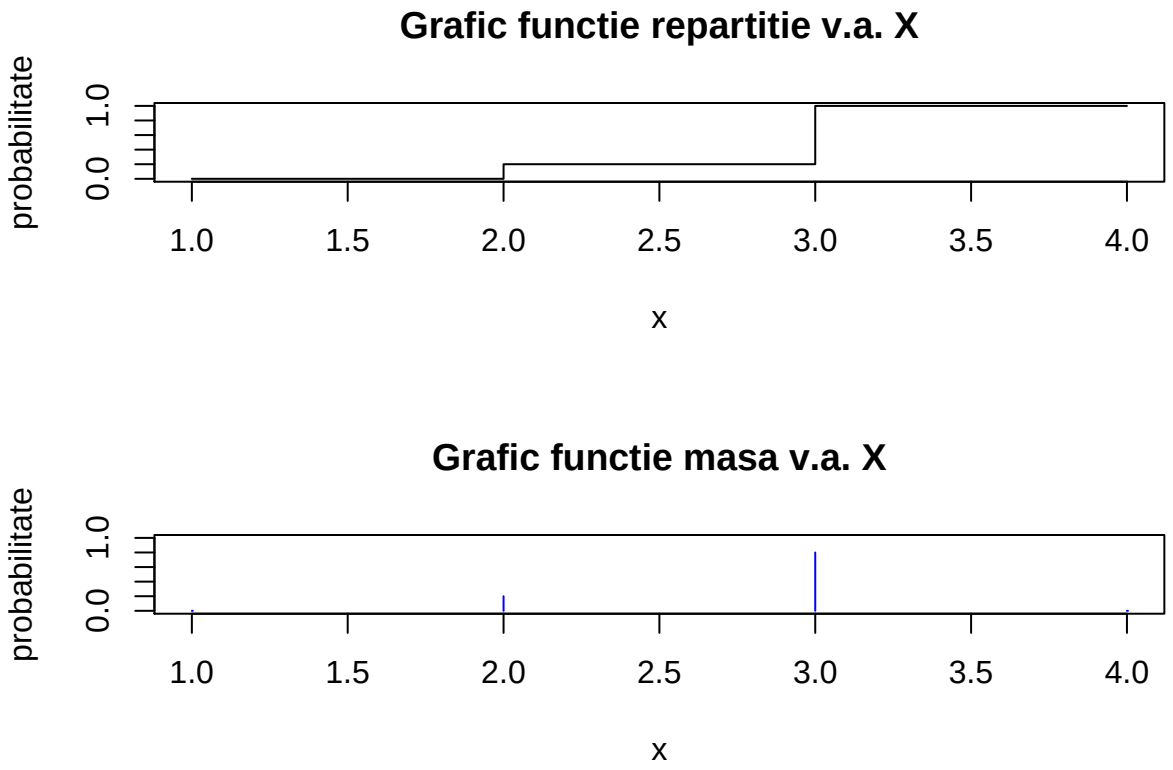


Problema 3

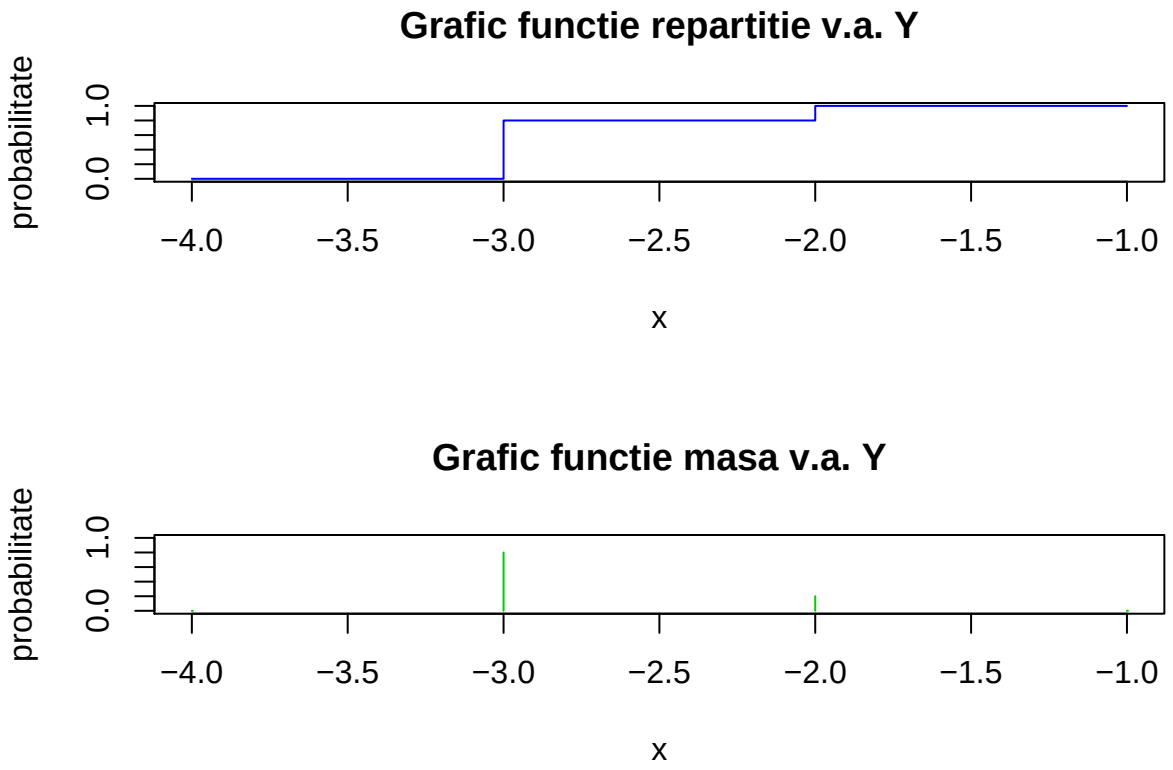
```r
fMasa <- function(X, x)
{
  xi <- outcomes(X)
  idx <- match(x, xi)

  el <- probs(X)[idx]
  el[is.na(el)] <- 0
  return(unname(el))
}

fRepartitie <- function(X, x)
{
  ps = c()
  for (val in x)
  {
    p <- sum(probs(X)[outcomes(X) <= val])

    ps <- c(ps,p)
  }
```

```
  return(ps)
}

fAfisare <- function(X, nume='')
{
  culori <- sample(c(1:4,6), 2)

  LB <- min(outcomes(X))
  LB <- LB - floor(abs(LB/10)) - 1
  UB <- max(outcomes(X))
  UB <- UB + floor(abs(UB/10)) + 1
  x_axis <- LB:UB
  x_axis <- sort(union(x_axis, outcomes(X)))

  par(mfrow=c(2,1))
  plot(x_axis, fRepartitie(X, x_axis), main=paste('Grafic functie repartitie',nume)
       , col=culori[1], 's', xlab='x', ylab='probabilitate', ylim=c(0,1))
  plot(x_axis, fMasa(X, x_axis), main=paste('Grafic functie masa',nume)
       , col=culori[2],type="h", xlab='x', ylab='probabilitate', ylim=c(0,1))
}
```

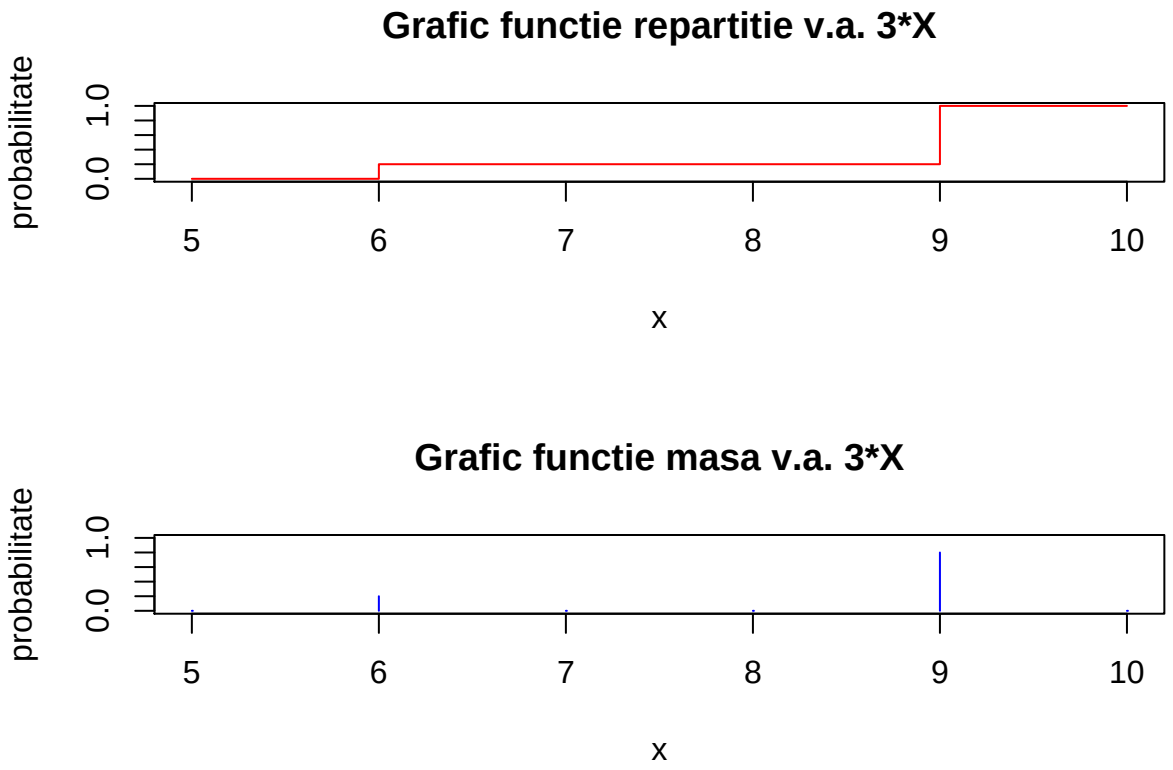
```
fAfisare(X, 'v.a. X')
```



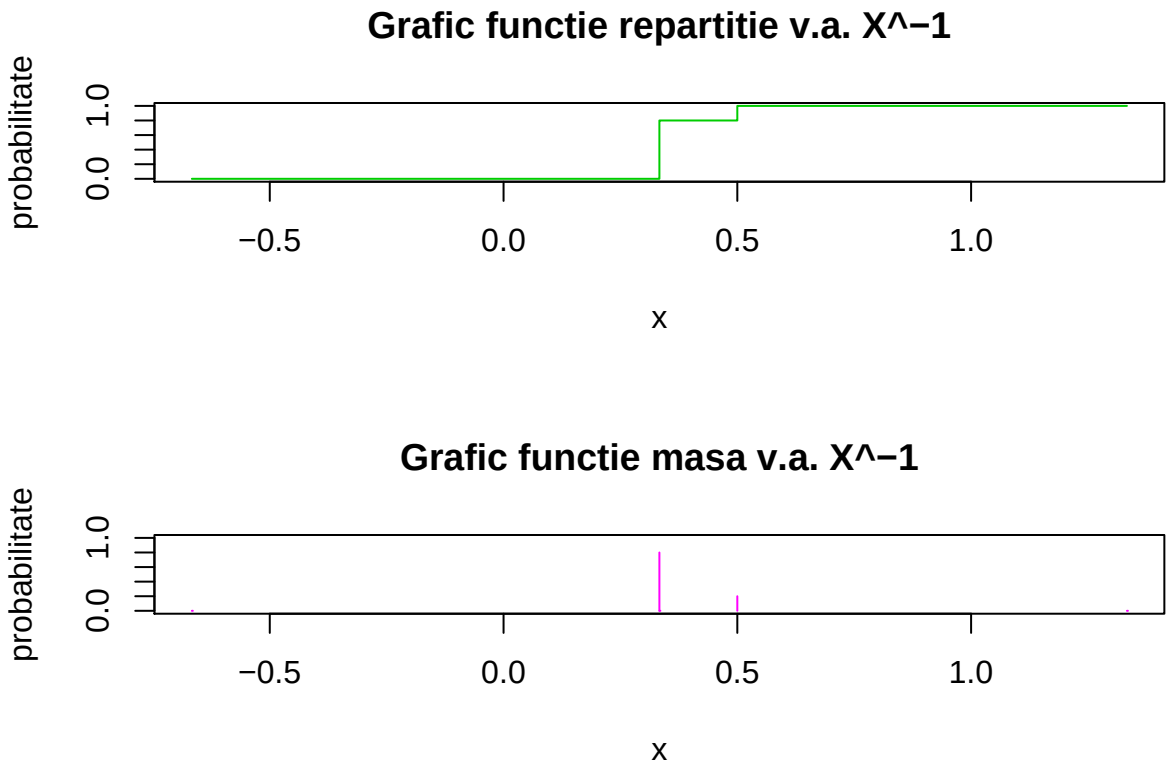
```
fAfisare(Y, 'v.a. Y')
```



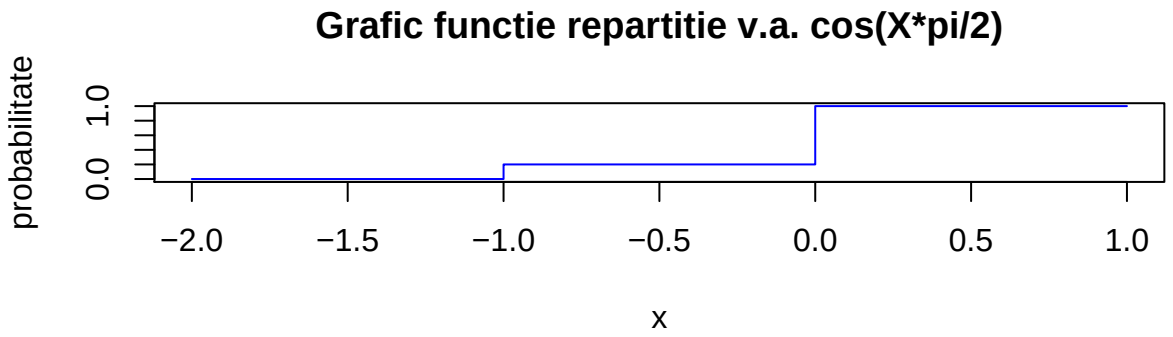
```
fAfisare(X3, 'v.a. 3*X')
```

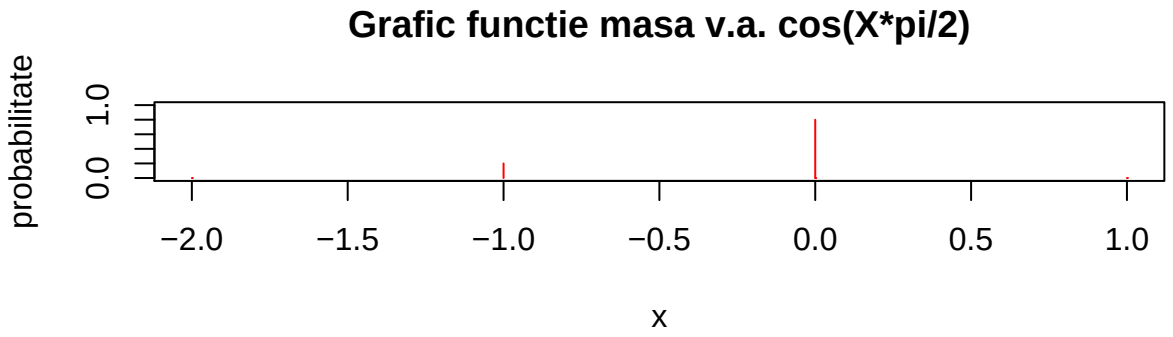


```
fAfisare(Xem1, 'v.a. X^-1')
```

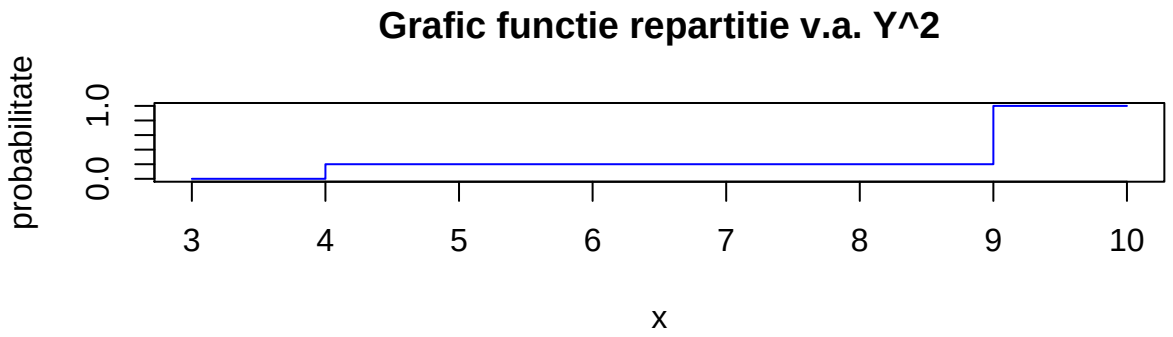


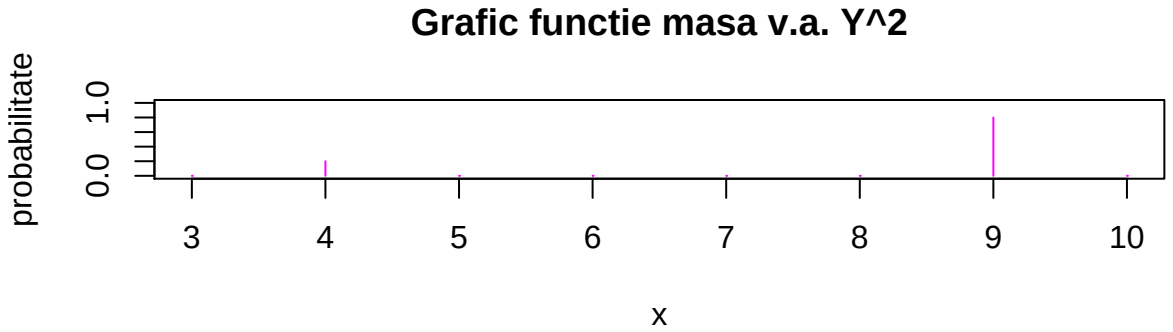
```
fAfisare(Xcos, 'v.a. cos(X*pi/2)')
```



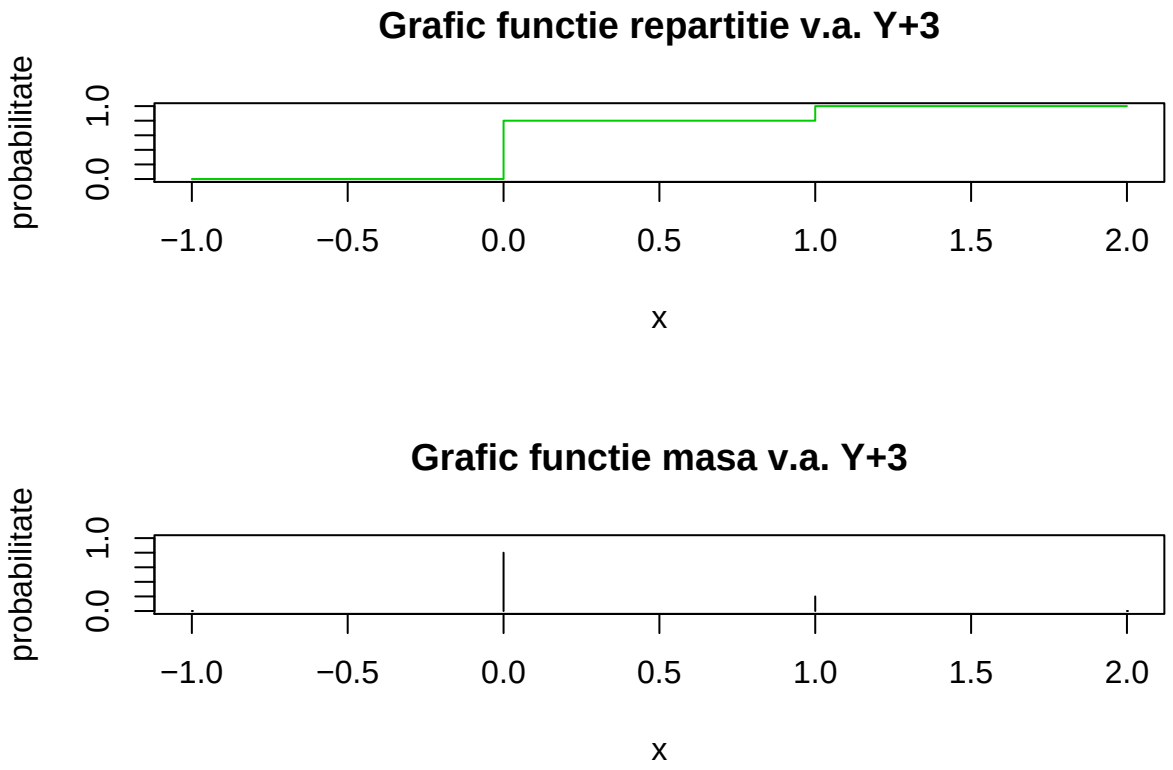


```
fAfisare(Ye2, 'v.a. Y^2')
```

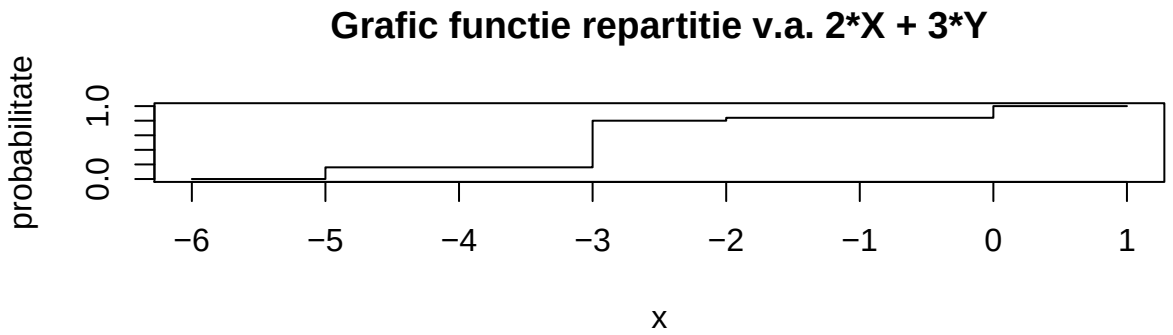


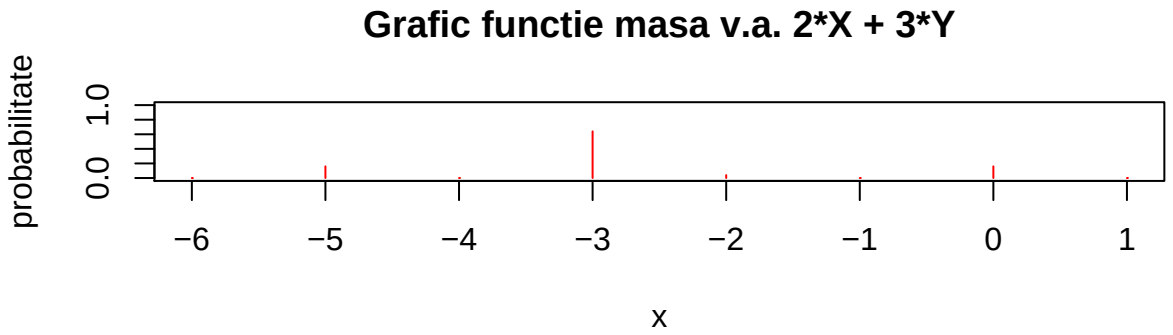


```
fAfisare(Yp3, 'v.a. Y+3')
```

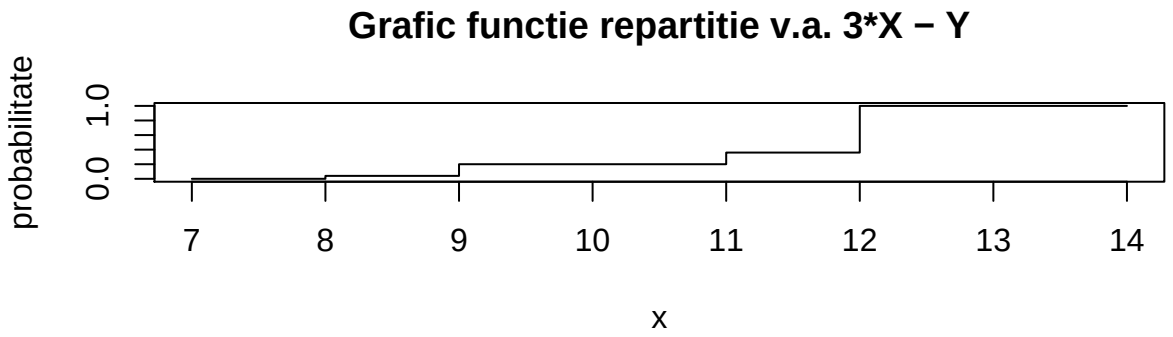


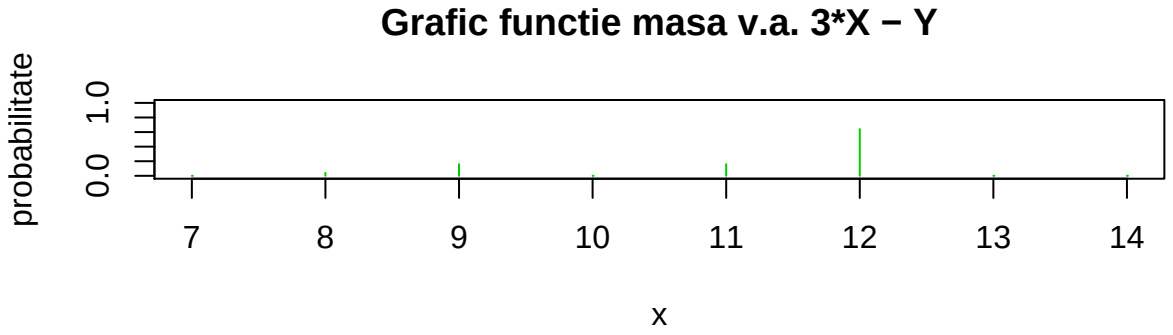
```
fAfisare(X2pY3, 'v.a. 2*X + 3*Y')
```



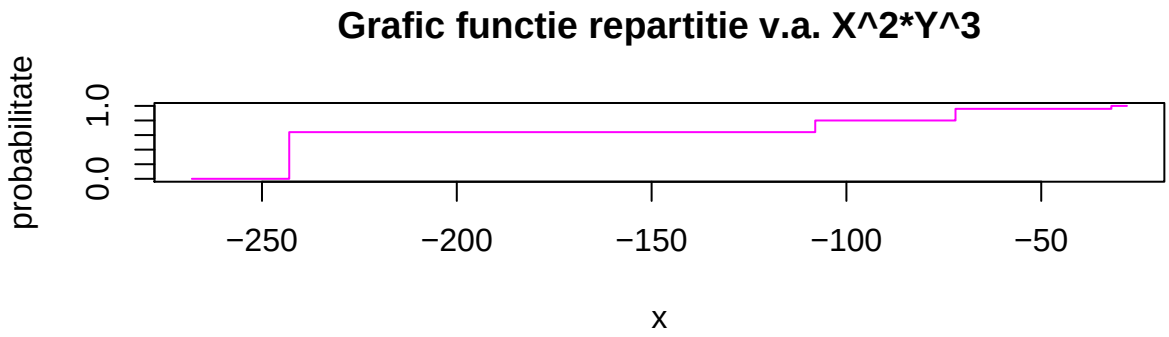


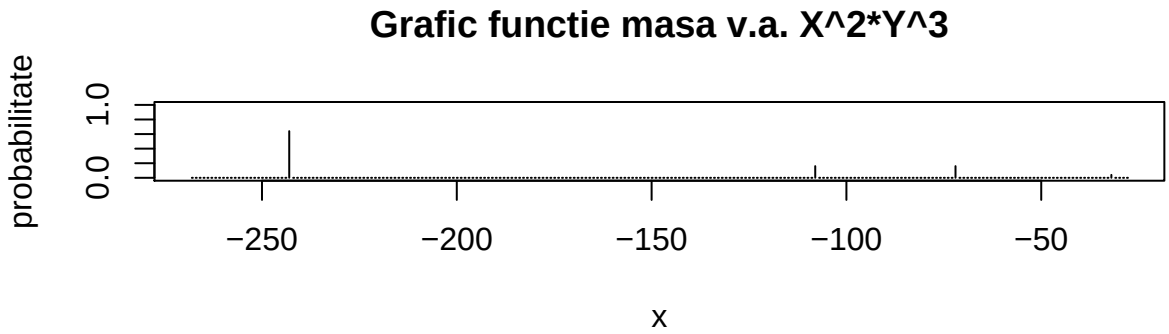
```
fAfisare(X3mY, 'v.a. 3*X - Y')
```



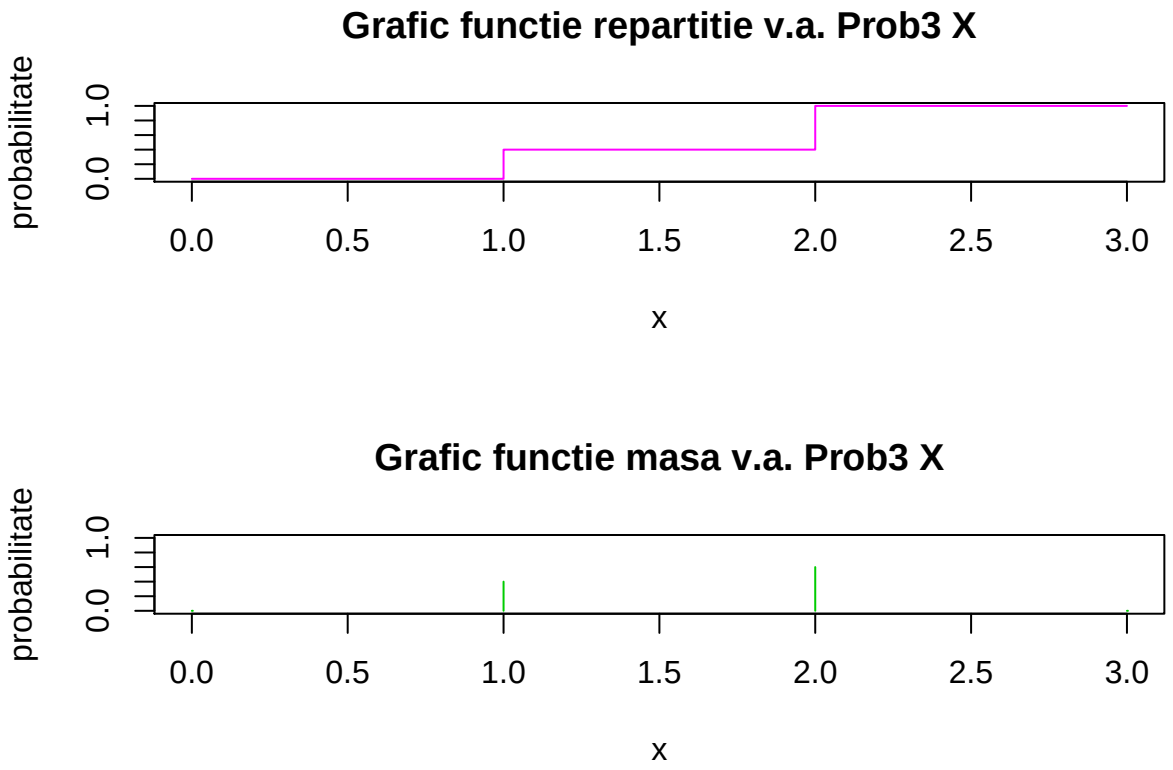


```
fAfisare(Xe2Ye3, 'v.a. X^2*Y^3')
```

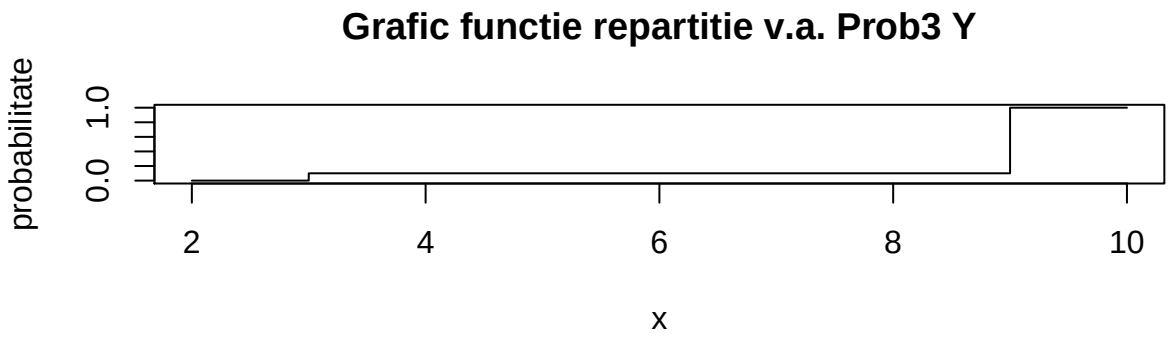


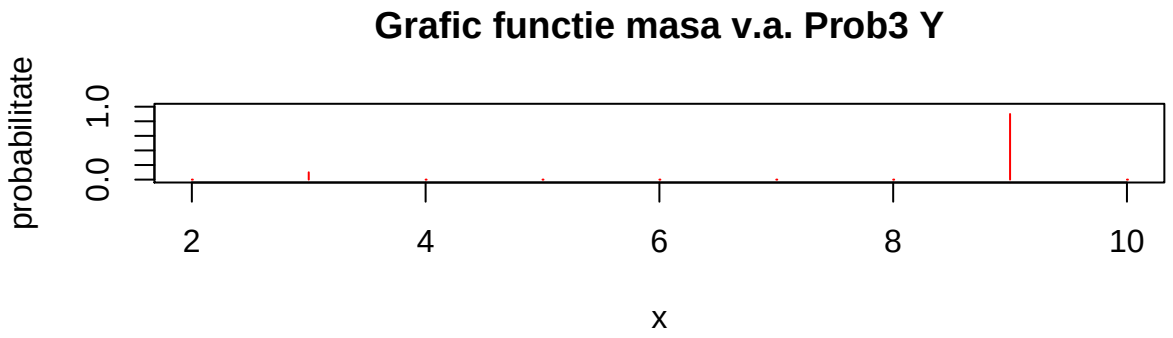


```
fAfisare(Prob3X, 'v.a. Prob3 X')
```



```
fAfisare(Prob3Y, 'v.a. Prob3 Y')
```





Problema 4

```
Z <- function(X, Y)
{
  moneda <- sample(c('H','T'),1)

  if (moneda == 'H') return(X)
  else return(Y)
}
```

generateVA este functia care genereaza n v.a. din Z

```
generateVA <- function(n, Z, X, Y)
{
  vas <- list()

  for (i in 1:n)
  {
    vas[[i]] <- Z(X,Y)
  }
```

```
  return(vas)
}


n <- 1000
Zs <- generateVA(n, Z, X, Y)
nr_X <- 0
```

Calculam de cate ori apar X si Y in generateVA

```
LB <- Inf
UB <- -Inf
for (i in 1:n)
{
  if (all(outcomes(Zs[[i]]) == outcomes(X)))
  {
    nr_X <- nr_X + 1
  }
  LB <- min(c(outcomes(Zs[[i]]), LB))
  UB <- max(c(outcomes(Zs[[i]]), UB))
}
nr_Y <- n-nr_X

(nr_X)
```

```
## [1] 491
```

```
(nr_Y)
```

```
## [1] 509
```

```
(probX <- nr_X/n)
```

```
## [1] 0.491
```

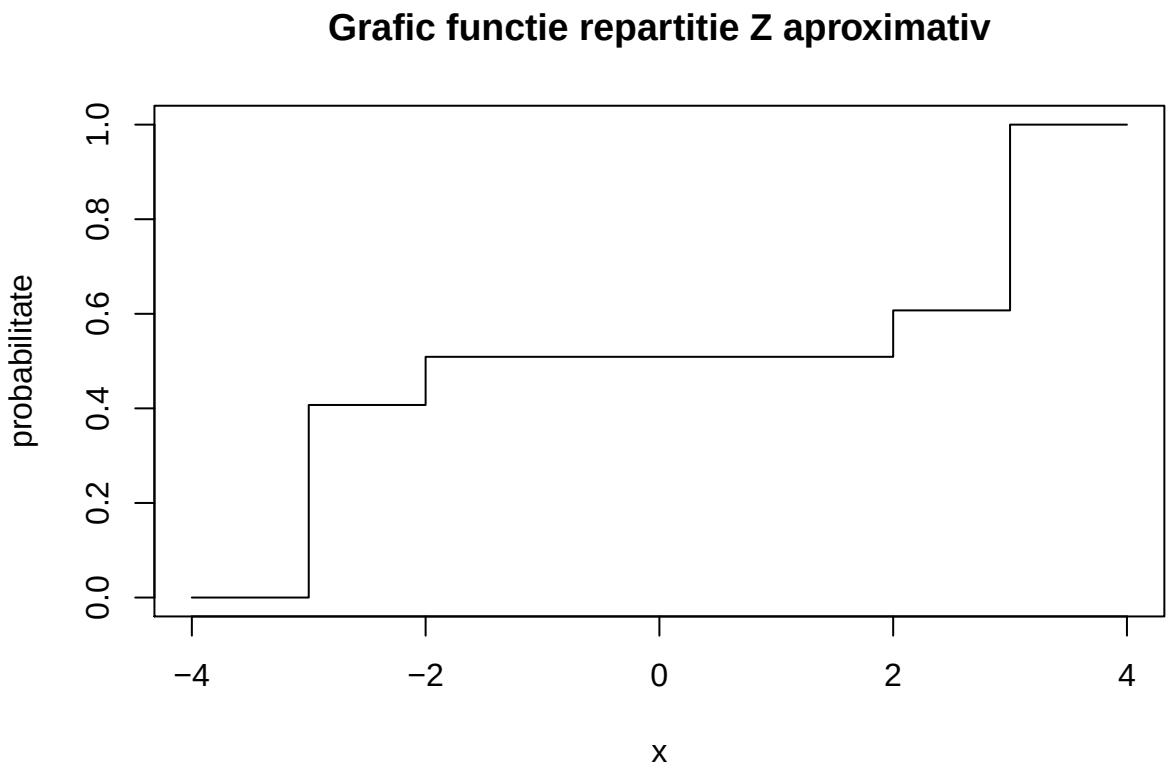
```
(probY <- nr_Y/n)
```

```
## [1] 0.509
```

Calculam o repartitie “ponderata”

```
LB <- LB - floor(abs(LB/10)) - 1
UB <- UB + floor(abs(UB/10)) + 1
x_axis <- LB:UB
x_axis <- sort(union(x_axis, outcomes(X), outcomes(Y)))

y_axis <- c()

for (i in x_axis)
{
```

```
  y_axis <- c(y_axis, probX * fRepartitie(X, i) + probY * fRepartitie(Y, i))
}

plot(x_axis, y_axis, main=paste('Grafic functie repartitie','Z aproximativ'),
     col=1, 's', xlab='x', ylab='probabilitate', ylim=c(0,1))
```



Din grafic putem deduce punctele in care repartitia se schimba (creste), acelea vor fi valorile posibile din v.a. Z

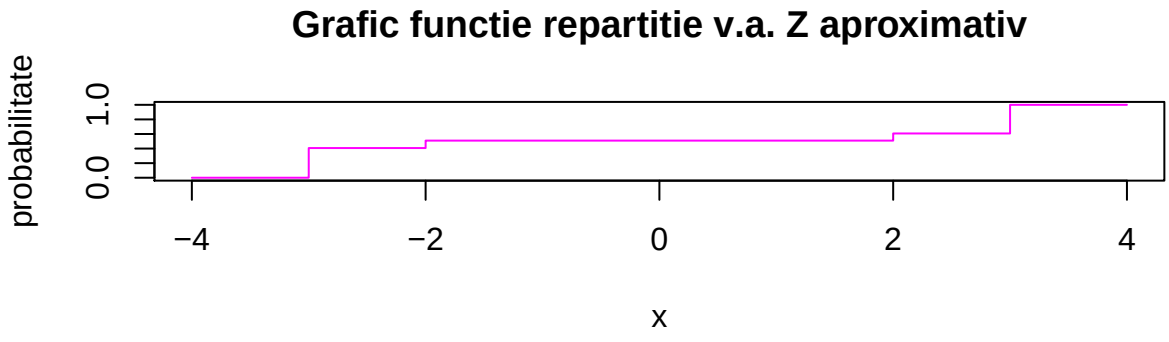
```
val_va <- c()
prob_va <- c()
lastVal <- numeric(0)
for (i in 1:length(x_axis))
{
  if (isTRUE(y_axis[i] != lastVal))
  {
    val_va <- c(val_va, x_axis[i])
    prob_va <- c(prob_va, y_axis[i]-lastVal)
  }
  lastVal <- y_axis[i]
}
```

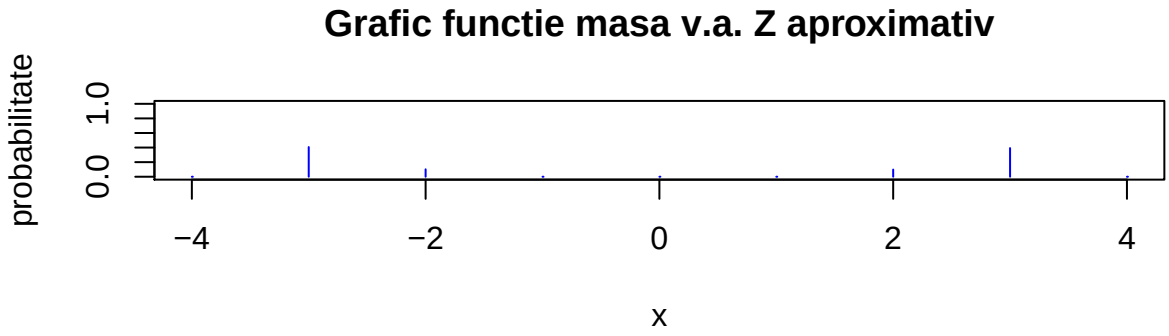
Construim o v.a. aproximativa

```
Zaprox <- RV(val_va,probs = prob_va)
(Zaprox)
```

```
## Random variable with 4 outcomes
##
## Outcomes       -3       -2        2        3
## Probs    509/1250 509/5000 491/5000 491/1250
```

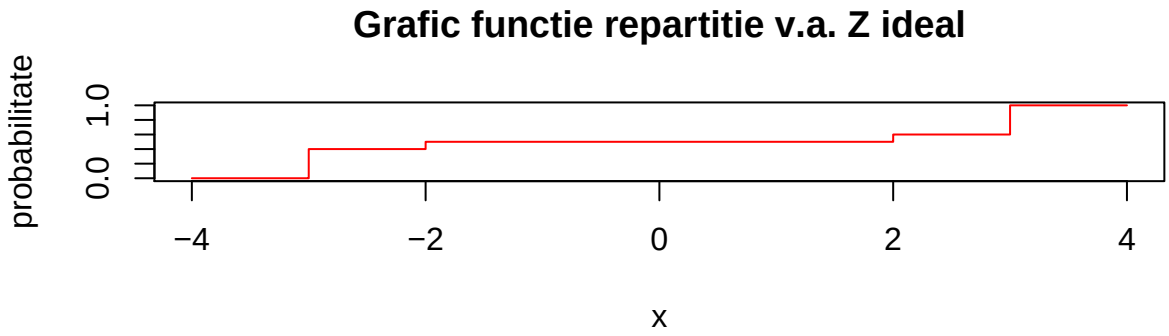
```
fAfisare(Zaprox, 'v.a. Z aproximativ')
```

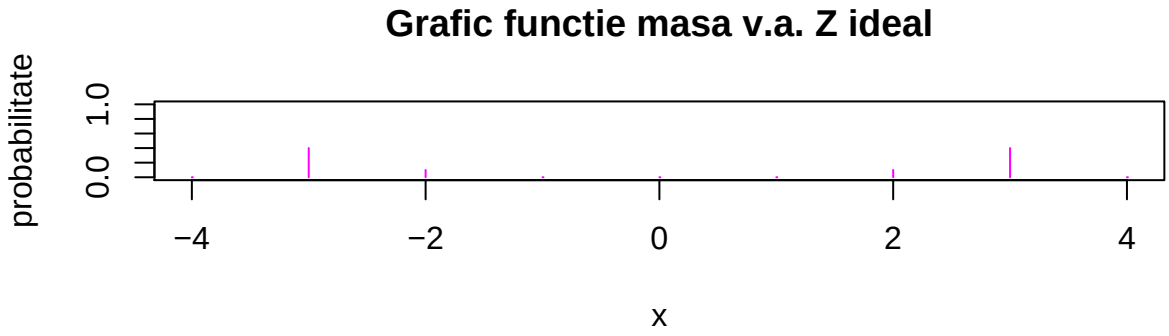




Din valori ne putem da seama cum ar arata Z ideal, atunci cand n -> Inf

```
Zideal <- RV(c(-3,-2,2,3),c(2/5,1/10,1/10,2/5))

fAfisare(Zideal, 'v.a. Z ideal')
```





In concluzie, este posibil sa aflam repartitia si functia de masa pentru Z, indiferent de probabilitatea de aparitie a lui X sau Y sau a numarului de v.a. din Z