Raport Proiect Statistica cu R-lang

2022-02-03

# Prerequisites

Setam environment pentru R markdown

knitr::opts\_chunk$set(fig.width = 10, fig.height = 10)

Initializam figurile/ploturile sa fie 1000x1000 ca dimensiune.

# Header

# Nota

acest proiect contine fisierul raport.Rmd plus raport.html si raport.docx care sunt fisierele generate cu RMarkdown compiler.

# Problema 1

### nota problema 1

Pentru problema 1 am creat o colectie de functii utile care pot fi reutilizate. Principul din spate se numeste functional programming.

Functie pentru extragerea de cei 2 minimi si maximul dintre valorile (a, b, c).

get\_min\_max\_2nd\_min <- function(a, b, c) {  
 # punem valorile intr-un array  
 values <- c(a, b, c)  
 # luam minim  
 minimum <- min(values)  
 # luam maxim  
 maximum <- max(values)  
 # alfam al doilea min  
 # '! values %in% c(minimum)' inseamna toate valorile care nu sunt in vectorul c(minimum)  
 # adica doar vectorul cu valoarea minima  
 second\_min <- min(values[! values %in% c(minimum)])  
 # returnam un vector to toate valorile in ordine crescatoare  
 # bune pentru formula de triunghi obtuz  
 return (c(minimum, second\_min, maximum))  
}

Functie pentru verificarea conditiei de triunghi obtuz din setul de 3 lungimi (a, b, c).

is\_obtuse\_triangle <- function(a, b, c) {  
 # formula este  
 # min^2 + second\_min^2 < max^2 -> if TRUE then obtuz else 'nu e obtuz'  
  
 # luam valorile si le ridicam la patrat pentru formula  
 values <- get\_min\_max\_2nd\_min(a, b, c) ^ 2  
 # membrul stang al egalitatii  
 min\_sum <- values[1] + values[2]  
  
 # daca suma minimelor < max^2  
 if (min\_sum < values[3]) {  
 # obtuz  
 return (TRUE)  
 }  
 # not obtuz  
 return (FALSE)  
}

Functie pentru verificarea conditiei de triunghi din setul de 3 lungimi (a, b, c).

is\_triangle <- function(a, b, c) {  
 # daca toate laturile sunt mai mici decat 1 pe 2 (0.5)  
 if (a <= 0.5 && b <= 0.5 && c <= 0.5) {  
 # triunghi  
 return (TRUE)  
 }  
 # non-triunghi  
 return (FALSE)  
}

## Cerinta

Pentru inceput setam variable esentiale pentru intreaga executie a problemei numarul 1.

total <- 5000 # nr total de simulari  
swap <- TRUE # pentru inversarea lui X si Y atunci cand X > Y

## Procedura 1

## Procedura 1 - Punctul 1

Functia pentru generare x si y aleator si uniform din [0, 1]

generate\_xy <- function(total = 5000) {  
 # 5000 individual unique random values between [0, 1]  
 x <- runif(total, 0, 1)  
 # 5000 individual unique random values between [0, 1]  
 y <- runif(total, 0, 1)  
 # why dataframe?  
 # its much easier to manage  
 return (data.frame(x = x, y = y))  
}

Functia pentru generare dataframe cu toate coloanele necesare in toata problema

exemplu:

## x y a b c color triangle obtuse  
## 1 0.50215305 0.3469016 0.34690161 0.1552514 0.49784695 #E5C07B TRUE TRUE  
## 2 0.40636783 0.7485510 0.40636783 0.3421832 0.25144896 #61AFEF TRUE FALSE  
## 3 0.59089269 0.4732470 0.47324695 0.1176457 0.40910731 #E5C07B TRUE TRUE  
## 4 0.73110424 0.3785572 0.37855720 0.3525470 0.26889576 #61AFEF TRUE FALSE  
## 5 0.02923464 0.2987789 0.02923464 0.2695443 0.70122107 #E06C75 FALSE FALSE  
## 6 0.07148460 0.9297189 0.07148460 0.8582343 0.07028107 #E06C75 FALSE FALSE

O sa avem un dataframe identic deoarece dorim sa salvam valorile x, y, a, b, c; ce culori sa aiba pentru fiecare pereche; daca este triunghi sau nu si daca sunt este obtuz sau nu.

generate\_dataframe <- function(  
 dataframe,  
 swap = FALSE  
) {  
 # avem nevoie de un dataframe cu toate valorile respective pe coloane  
 # pentru a le avea pe toate intr-o singura colectie sa fie mai usor de operat  
 pairs = list(list(dataframe$x), list(dataframe$y))  
 # culori, rosu sau albastru  
 colors = c()  
 # valori bool pentru triunghi  
 triangle = c()  
 # valori bool pentru triunghi obtuz  
 obtuse = c()  
 # array urile cu lungimi  
 aa <- c()  
 bb <- c()  
 cc <- c()  
 for (iter in seq\_len(length(dataframe$y))) {  
 # print(iter)  
 x <- pairs[[1]][[1]][iter]  
 y <- pairs[[2]][[1]][iter]  
  
 # le face swap pentru nu are sens  
 # dpdv matematic si alfabetic sa fie X > Y  
 # cand pe axa avem in ordine (0, X) (X, Y) (Y, 1)  
 if (swap) {  
 if (x > y) {  
 z <- x  
 x <- y  
 y <- z  
 }  
 }  
  
 # generam lungimile pentru un triunghi  
 a <- x  
 b <- abs(x - y)  
 c <- abs(1 - y)  
  
 # punem valorile in array uri  
 aa <- append(aa, a)  
 bb <- append(bb, b)  
 cc <- append(cc, c)  
  
 # check if the a, b, c lengths form a triangle  
 if (is\_triangle(a, b, c)) {  
 triangle <- append(triangle, TRUE)  
 colors <- append(colors, "#61AFEF")  
  
 # check if the a, b, c lengths form an obtuse triangle  
 if (is\_obtuse\_triangle(a, b, c)) {  
 obtuse <- append(obtuse, TRUE)  
 # am pus cele obtuze cu galben sa fie mai usor  
 # de vizualizat  
 } else {  
 obtuse <- append(obtuse, FALSE)  
 }  
 } else {  
 # have to initialize with default values  
 colors <- append(colors, "#E06C75")  
 triangle <- append(triangle, FALSE)  
 obtuse <- append(obtuse, FALSE)  
 }  
 }  
 # punem toate coloanele in dataframe  
 dataframe[,"a"] <- aa  
 dataframe[,"b"] <- bb  
 dataframe[,"c"] <- cc  
 dataframe[,"color"] <- colors  
 dataframe[,"triangle"] <- triangle  
 dataframe[,"obtuse"] <- obtuse  
 return (dataframe)  
}

### Punctul 1 solutie

### Justificare teoretica

ramane pentru vlad si bogdan

### Justificare practica

# generam valorile x si y  
xy <- generate\_xy()  
  
# cream dataframe cu tot ce am obtinut  
dataframe <- generate\_dataframe(  
 xy,  
 swap=swap)  
  
total\_triangles <- length(which(dataframe$triangle==TRUE))

Observam ca avem numar triunghiuri > 0, deci am aratat practic ca se pot forma triungiuri.

### Functii utile pentru problema 1

functiile pentru probabilitati

# functia pentru probabilitatea de triunghiuri  
get\_total\_triangles\_probability <- function(dataframe, total) {  
 # numaram in dataframe care valori sunt true in vectorul de boolene  
 total\_triangles = length(which(dataframe$triangle==TRUE))  
 # cazuri fav pe posibile  
 return (total\_triangles / total)  
}  
  
  
# probabilitatea pentru obtuze din toate puncturile  
get\_total\_obtuses\_probability\_from\_all\_points <- function(dataframe, total) {  
 # numaram in dataframe care valori sunt true in vectorul de boolene  
 total\_triangles = length(which(dataframe$obtuse==TRUE))  
 # cazuri fav pe posibile  
 # probabilitatea pentru triunghiuri obtuze din toate toate punctele  
 return (total\_triangles / total)  
}  
  
# functia pentru probabilitatea de triunghiuri obtuze  
get\_total\_obtuses\_probability\_from\_all\_triangles <- function(dataframe, total) {  
 total\_triangles = length(which(dataframe$triangle==TRUE))  
 total\_obtuses = length(which(dataframe$obtuse==TRUE))  
 # probabilitatea pentru triunghiuri obtuze din toate triunghiurile  
 return (total\_obtuses / total\_triangles)  
}

Functia pentru lungimiile medii

# functia care face media lungimilor a, b, c  
get\_mean\_lengths <- function(dataframe) {  
 # mediile pentru (a1, a2, ..., an)  
 as\_mean = paste("all a's mean: ", mean(dataframe$a), sep="")  
 print(as\_mean)  
 # mediile pentru (b1, b2, ..., bn)  
 bs\_mean = paste("all b's mean: ", mean(dataframe$b), sep="")  
 print(bs\_mean)  
 # mediile pentru (c1, c2, ..., cn)  
 cs\_mean = paste("all c's mean: ", mean(dataframe$c), sep="")  
 print(cs\_mean)  
}

## Procedura 1 - Punctul 2

### Solutie

#### Justificare teoretica

ramane pentru vlad si bogdan

#### Justificare practica

Folosim tot ce am generat punctele anterioare si apelam functia creata pentru acest punct.

# means for a, b, c  
get\_mean\_lengths(dataframe)

## [1] "all a's mean: 0.335489809137816"  
## [1] "all b's mean: 0.331321972978674"  
## [1] "all c's mean: 0.33318821788351"

Dupa cum observati, valorile lugimilor medii sunt intotdeauna 0.33 approximativ. Uneori mai apare si 0.32 sau 0.34.

## Procedura 1 - Punctul 3.a

### Solutie

# avem nevoie de nr total de obtuze si triunghiuri pentru a face un plot mai sugestiv  
total\_triangles <- length(which(dataframe$triangle==TRUE))  
triangle\_prob <- get\_total\_triangles\_probability(dataframe, total)  
  
# cream titlu pentru plot  
main\_text <- paste(  
 "(X, Y) @ [0, 1] -> range: ",  
 total,  
 "\n",  
 "total\_triangles: ",  
 total\_triangles,  
 " | ",  
 triangle\_prob, "%",  
 sep="")  
  
white <- "#ABB2BF"  
# setam culoare de background si foreground pentru plot  
par(bg = "#282C34", fg=white)  
  
# plotam rezultatele finale  
plot(  
 x = dataframe$x,  
 y = dataframe$y,  
 xlab = "X",  
 ylab = "Y",  
 xlim = c(0,1.1),  
 ylim = c(0,1.1),  
 main = main\_text, # titlu pentru plot  
 col.main = white,  
 col.sub = white, # Subtitle color  
 col.lab = white, # X and Y-axis labels color  
 col.axis = white, # Tick labels color  
 col = dataframe$color,  
 pch = 16,  
 cex = 1.5,  
 cex.main = 1.5, # Title size  
 cex.sub = 0.5, # Subtitle size  
 cex.lab = 2, # X-axis and Y-axis labels size  
 cex.axis = 1, # Axis labels size  
 # family = "monaco"  
)

## 

Plotare pentru 5000 de generari random de X si Y din Uniform[0, 1]. Graficul este simetric deci inseamna ca este corect. Daca nu facem swap la valorile X si Y atunci graficul este asimetric, deci incorect.

## Procedura 1 - Punctul 3.b

### Solutie

# calcuam probabilitatea pentru formare triunghiuri  
triangle\_prob <- get\_total\_triangles\_probability(dataframe, total)  
cat(triangle\_prob)

## 0.2466

Dupa cum observati, este mereu approx 0.25%, adica 1/4. Uneori este si 0.24 sau 0.26.

## Procedura 1 - Punctul 3c

### Justificare teoretica

ramane de facut pentru vlad si bogdan

## Procedura 1 - Punctul 4

### Solutie

### Justificare teoretica

ramane pentru vlad si bogdan

### Justificare practica

# calcuam probabilitatea pentru formare triunghiuri obtuze din toate punctele  
obtuses\_prob\_from\_all\_points <- get\_total\_obtuses\_probability\_from\_all\_points(dataframe, total)  
print(obtuses\_prob\_from\_all\_points)

## [1] 0.1704

# calcuam probabilitatea pentru formare triunghiuri obtuze doar din triunghiuri  
obtuses\_prob\_from\_triangles <- get\_total\_obtuses\_probability\_from\_all\_triangles(dataframe, total)  
print(obtuses\_prob\_from\_triangles)

## [1] 0.6909976

Dupa observati, probabilitatea este mereu approx cu 0.67%, evident, probabilitatea de formare a triunghiurile obtuze din toate triunghirile formate.

## Procedura 2

Singura diferenta fata de procedura 1 este modul in care alegem X si Y, in rest toate functiile aplicabile raman la fel. De aceea, au fost facut si asa, pentru reutilizare.

generate\_xy\_punctul\_4 <- function(total = 5000) {  
 # 5000 individual unique random values between [0, 1]  
 x <- runif(total, 0, 1)  
 y <- c()  
 for (xx in x) {  
 diff\_from\_0\_to\_x <- xx  
 diff\_from\_x\_to\_1 <- abs(1 - xx)  
 if (diff\_from\_0\_to\_x > diff\_from\_x\_to\_1) {  
 yy <- runif(1, min=0, max=xx)  
 } else {  
 yy <- runif(1, min=xx, max=1)  
 }  
 y <- append(y, yy)  
 }  
 # why dataframe?  
 # its much easier to manage  
 return (data.frame(x = x, y = y))  
}

Generam valorile x si y.

xy <- generate\_xy\_punctul\_4()  
  
# cream dataframe cu tot ce am obtinut  
dataframe <- generate\_dataframe(  
 xy,  
 swap=swap)  
  
head(dataframe)

## x y a b c color triangle obtuse  
## 1 0.07863899 0.2941534 0.07863899 0.2155144 0.70584659 #E06C75 FALSE FALSE  
## 2 0.62169643 0.4397931 0.43979312 0.1819033 0.37830357 #61AFEF TRUE TRUE  
## 3 0.92601030 0.5101469 0.51014685 0.4158635 0.07398970 #E06C75 FALSE FALSE  
## 4 0.97135837 0.3478275 0.34782748 0.6235309 0.02864163 #E06C75 FALSE FALSE  
## 5 0.46772459 0.8108162 0.46772459 0.3430916 0.18918378 #61AFEF TRUE TRUE  
## 6 0.92642816 0.3355617 0.33556169 0.5908665 0.07357184 #E06C75 FALSE FALSE

Am generat valorile si le-am pus in dataframe si am printat primele 6 linii ale dataframe-ului.

## Procedura 2 - Punctul 5

adica aflarea probabilitatii pentru nr total de triunghiuri si demonstratia teoretic

### Solutie

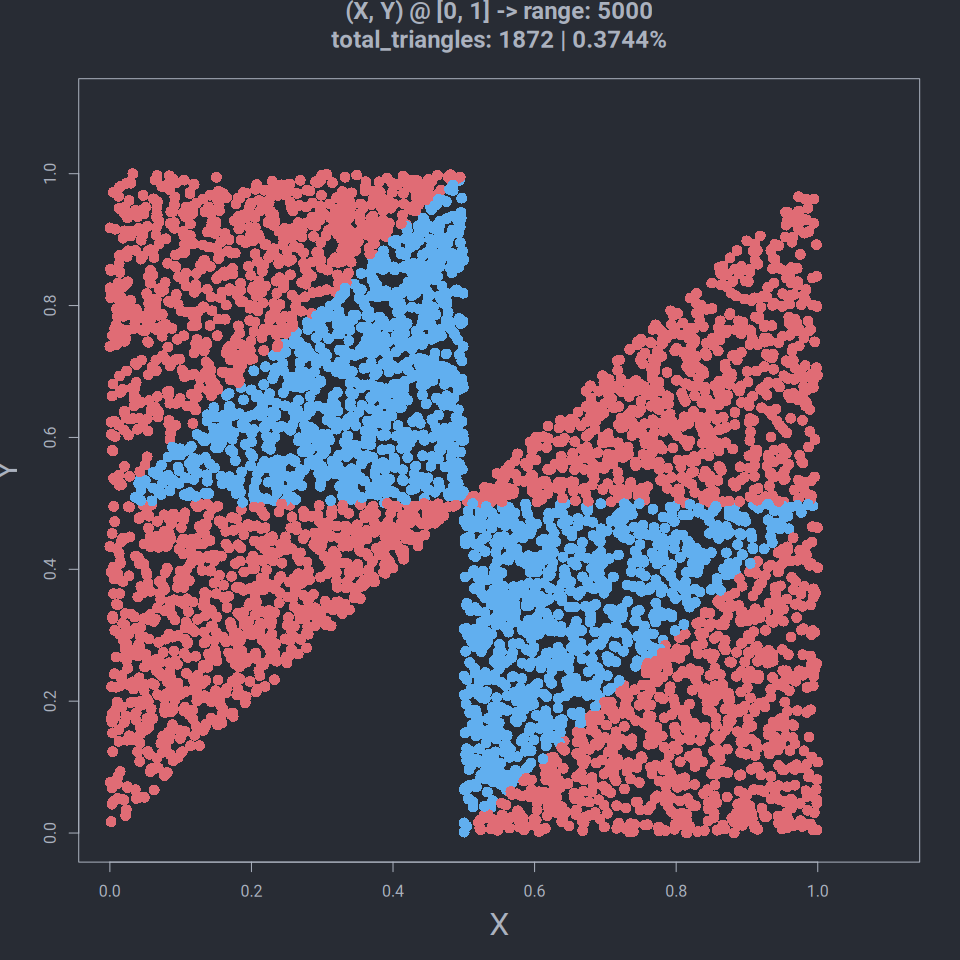
### Justificare teoretica

ramane pentru vlad si bogdan

### Justificare practica

Plotam sa vedem distributia de triunghiuri.

total\_triangles <- length(which(dataframe$triangle==TRUE))  
triangle\_prob <- get\_total\_triangles\_probability(dataframe, total)  
  
# adaugam titlu pentru plot  
main\_text <- paste(  
 "(X, Y) @ [0, 1] -> range: ",  
 total,  
 "\n",  
 "total\_triangles: ",  
 total\_triangles,  
 " | ",  
 triangle\_prob, "%",  
 "\n",  
 sep="")  
  
# punctul 3.a  
size <- 600  
white <- "#ABB2BF"  
# setam culoare de background si foreground pentru plot  
par(bg = "#282C34", fg=white)  
  
# plotam rezultatele finale  
plot(  
 x = dataframe$x,  
 y = dataframe$y,  
 xlab = "X",  
 ylab = "Y",  
 xlim = c(0,1.1),  
 ylim = c(0,1.1),  
 main = main\_text,  
 col.main = white,  
 col.sub = white, # Subtitle color  
 col.lab = white, # X and Y-axis labels color  
 col.axis = white, # Tick labels color  
 col = dataframe$color,  
 pch = 16,  
 cex = 1.5,  
 cex.main = 1.5, # Title size  
 cex.sub = 0.5, # Subtitle size  
 cex.lab = 2, # X-axis and Y-axis labels size  
 cex.axis = 1, # Axis labels size  
 # family = "monaco"  
)



calculam probabiliatea de formare pentru triunghiuri cu noua metoda de alegere

total\_triangles\_prob <- get\_total\_triangles\_probability(dataframe, total)  
print(total\_triangles\_prob)

## [1] 0.3744

Dupa cum se observa probabilitatea este mereu approx cu 0.38%

## Procedura 2 - Punctul 6

Nu are sens. Este exact ca la procedura 1. Daca citim cerinta o sa reiasa ca rezolvarea este exact la la procedura 1.

Functia de generate X si Z.

generate\_xz <- function(total = 5000) {  
 # 5000 individual unique random values between [0, 1]  
 x <- runif(total, 0, 1)  
 # 5000 individual unique random values between [0, 1]  
 z <- runif(total, 0, 1)  
 # why dataframe?  
 # its much easier to manage  
 return (data.frame(x = x, y = z))  
}

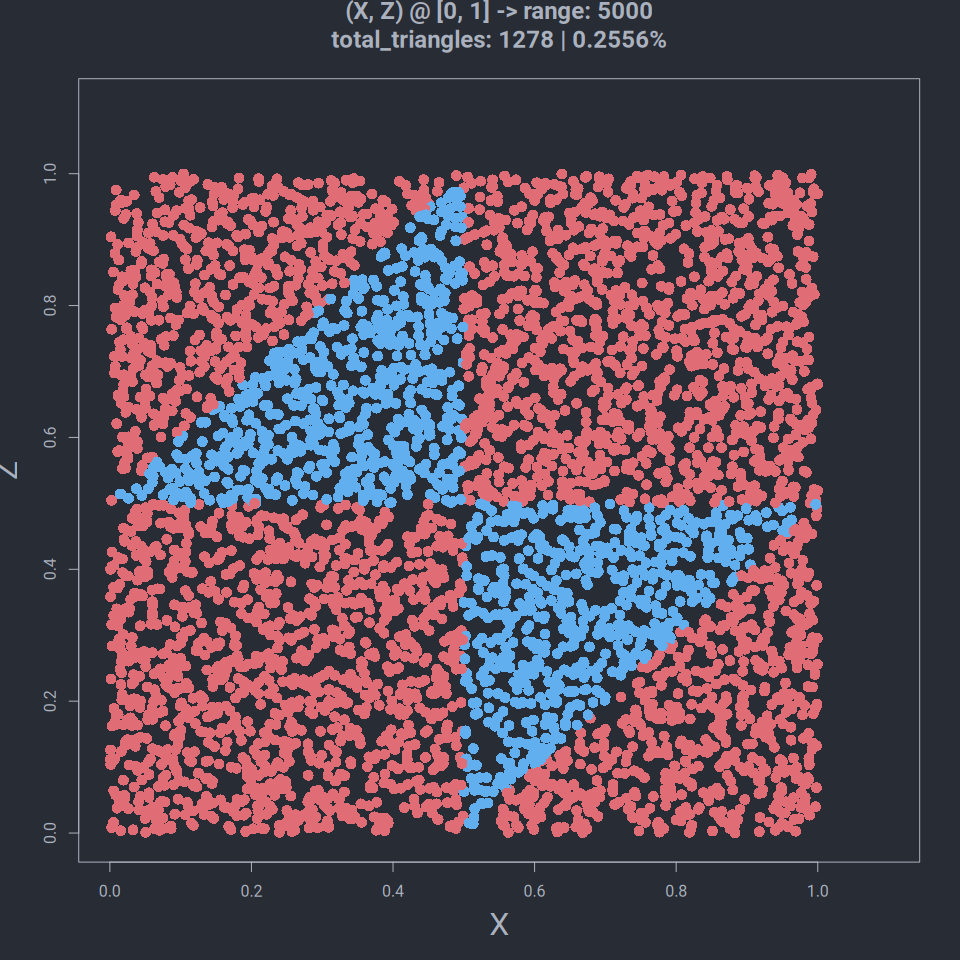
Generam valorile X si Z si le salvam in dataframe.

# generam valorile x si z  
xz <- generate\_xz()  
  
# cream dataframe cu tot ce am obtinut  
dataframe <- generate\_dataframe(  
 xz,  
 swap=swap)  
  
head(dataframe)

## x y a b c color triangle  
## 1 0.9348130 0.93284821 0.93284821 0.001964806 0.06518698 #E06C75 FALSE  
## 2 0.5326960 0.06393779 0.06393779 0.468758174 0.46730403 #61AFEF TRUE  
## 3 0.7121899 0.46096611 0.46096611 0.251223817 0.28781007 #61AFEF TRUE  
## 4 0.7450928 0.60151253 0.60151253 0.143580221 0.25490725 #E06C75 FALSE  
## 5 0.1623842 0.58297209 0.16238420 0.420587898 0.41702791 #61AFEF TRUE  
## 6 0.8525297 0.13110266 0.13110266 0.721427047 0.14747030 #E06C75 FALSE  
## obtuse  
## 1 FALSE  
## 2 FALSE  
## 3 TRUE  
## 4 FALSE  
## 5 FALSE  
## 6 FALSE

Plotam valorile.

total\_triangles <- length(which(dataframe$triangle==TRUE))  
triangle\_prob <- get\_total\_triangles\_probability(dataframe, total)  
# adaugam titlu pentru plot  
main\_text <- paste(  
 "(X, Z) @ [0, 1] -> range: ",  
 total,  
 "\n",  
 "total\_triangles: ",  
 total\_triangles,  
 " | ",  
 triangle\_prob, "%",  
 "\n",  
 sep="")  
  
# punctul 3.a  
size <- 600  
white <- "#ABB2BF"  
# setam culoare de background si foreground pentru plot  
par(bg = "#282C34", fg=white)  
  
# plotam rezultatele finale  
plot(  
 x = dataframe$x,  
 y = dataframe$y,  
 xlab = "X",  
 ylab = "Z",  
 xlim = c(0,1.1),  
 ylim = c(0,1.1),  
 main = main\_text,  
 col.main = white,  
 col.sub = white, # Subtitle color  
 col.lab = white, # X and Y-axis labels color  
 col.axis = white, # Tick labels color  
 col = dataframe$color,  
 pch = 16,  
 cex = 1.5,  
 cex.main = 1.5, # Title size  
 cex.sub = 0.5, # Subtitle size  
 cex.lab = 2, # X-axis and Y-axis labels size  
 cex.axis = 1, # Axis labels size  
)



## Procedura 3

Singura diferenta fata de procedura 1 si 2 este modul in care alegem X si Y, in rest toate functiile aplicabile raman la fel. De aceea, au fost facut si asa, pentru reutilizare.

Pentru aceasta procedura avem nevoie de o functie care sa selecteze random cap sau pajura

select\_cap\_sau\_pajura\_random <- function() {  
 data <- c("cap", "pajura")  
 result <- sample(x=data, size=1)  
 if (result == "cap") {  
 return ("stanga")  
 }  
 return ("dreapta")  
}

Functia de generare (X, Y)

generate\_xy\_procedura\_3 <- function(total = 5000) {  
 # 5000 individual unique random values between [0, 1]  
 x <- runif(total, 0, 1)  
 y <- c()  
 for (xx in x) {  
 zona <- select\_cap\_sau\_pajura\_random()  
 if (zona == "stanga") {  
 yy <- runif(1, min=0, max=xx)  
 } else {  
 yy <- runif(1, min=xx, max=1)  
 }  
 y <- append(y, yy)  
 }  
 # why dataframe?  
 # its much easier to manage  
 return (data.frame(x = x, y = y))  
}

generam si valorile necesare

# generam valorile x si y  
xy <- generate\_xy\_procedura\_3()  
  
# cream dataframe cu tot ce am obtinut  
dataframe <- generate\_dataframe(  
 xy,  
 swap=swap)  
  
head(dataframe)

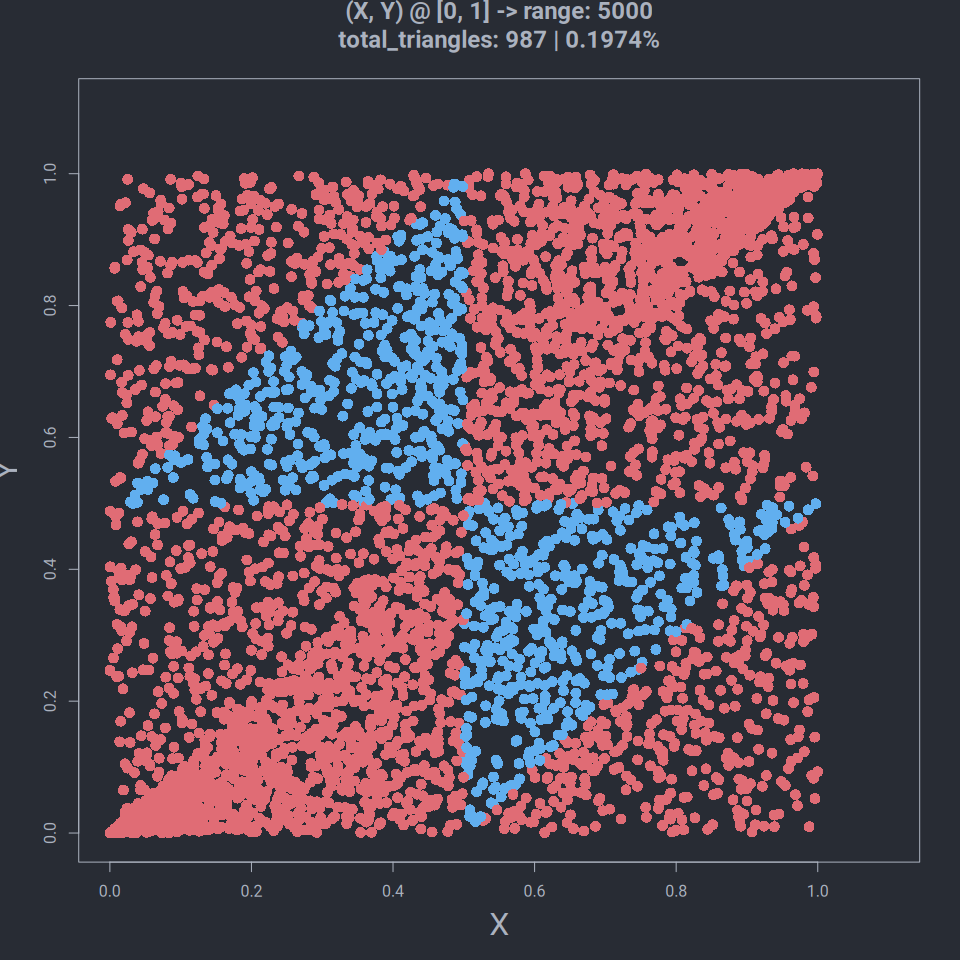
## x y a b c color triangle obtuse  
## 1 0.3636839 0.12322307 0.12322307 0.240460847 0.6363161 #E06C75 FALSE FALSE  
## 2 0.5925823 0.37425551 0.37425551 0.218326837 0.4074177 #61AFEF TRUE FALSE  
## 3 0.6830443 0.44569222 0.44569222 0.237352125 0.3169557 #61AFEF TRUE TRUE  
## 4 0.2058745 0.31771709 0.20587448 0.111842609 0.6822829 #E06C75 FALSE FALSE  
## 5 0.2700767 0.03828688 0.03828688 0.231789797 0.7299233 #E06C75 FALSE FALSE  
## 6 0.3853701 0.38149827 0.38149827 0.003871812 0.6146299 #E06C75 FALSE FALSE

## Procedura 3 - Punctul 7 - 5 (punctul 5 care se repeta)

### Solutie

Plotam sa vedem distributia de triunghiuri.

# variable pentru plot  
total\_triangles <- length(which(dataframe$triangle==TRUE))  
total\_obtuses <- length(which(dataframe$obtuse==TRUE))  
triangle\_prob <- get\_total\_triangles\_probability(dataframe, total)  
  
# adaugam titlu pentru plot  
main\_text <- paste(  
 "(X, Y) @ [0, 1] -> range: ",  
 total,  
 "\n",  
 "total\_triangles: ",  
 total\_triangles,  
 " | ",  
 triangle\_prob, "%",  
 "\n",  
 sep="")  
  
# punctul 3.a  
size <- 600  
white <- "#ABB2BF"  
# setam culoare de background si foreground pentru plot  
par(bg = "#282C34", fg=white)  
  
# plotam rezultatele finale  
plot(  
 x = dataframe$x,  
 y = dataframe$y,  
 xlab = "X",  
 ylab = "Y",  
 xlim = c(0,1.1),  
 ylim = c(0,1.1),  
 main = main\_text,  
 col.main = white,  
 col.sub = white, # Subtitle color  
 col.lab = white, # X and Y-axis labels color  
 col.axis = white, # Tick labels color  
 col = dataframe$color,  
 pch = 16,  
 cex = 1.5,  
 cex.main = 1.5, # Title size  
 cex.sub = 0.5, # Subtitle size  
 cex.lab = 2, # X-axis and Y-axis labels size  
 cex.axis = 1, # Axis labels size  
 # family = "monaco"  
)

 Dupa cum observati foarte multe puncte sunt localizare in colturile stanga jos si dreapta sus.

## Procedura 3 - Punctul 7 - 6 (punctul 6 care se repeta)

Nu are sens. Este exact ca la procedura 1. Daca citim cerinta o sa reiasa ca rezolvarea este exact la la procedura 1.

Functia de generate X si Z.

generate\_xz <- function(total = 5000) {  
 # 5000 individual unique random values between [0, 1]  
 x <- runif(total, 0, 1)  
 # 5000 individual unique random values between [0, 1]  
 z <- runif(total, 0, 1)  
 # why dataframe?  
 # its much easier to manage  
 return (data.frame(x = x, y = z))  
}

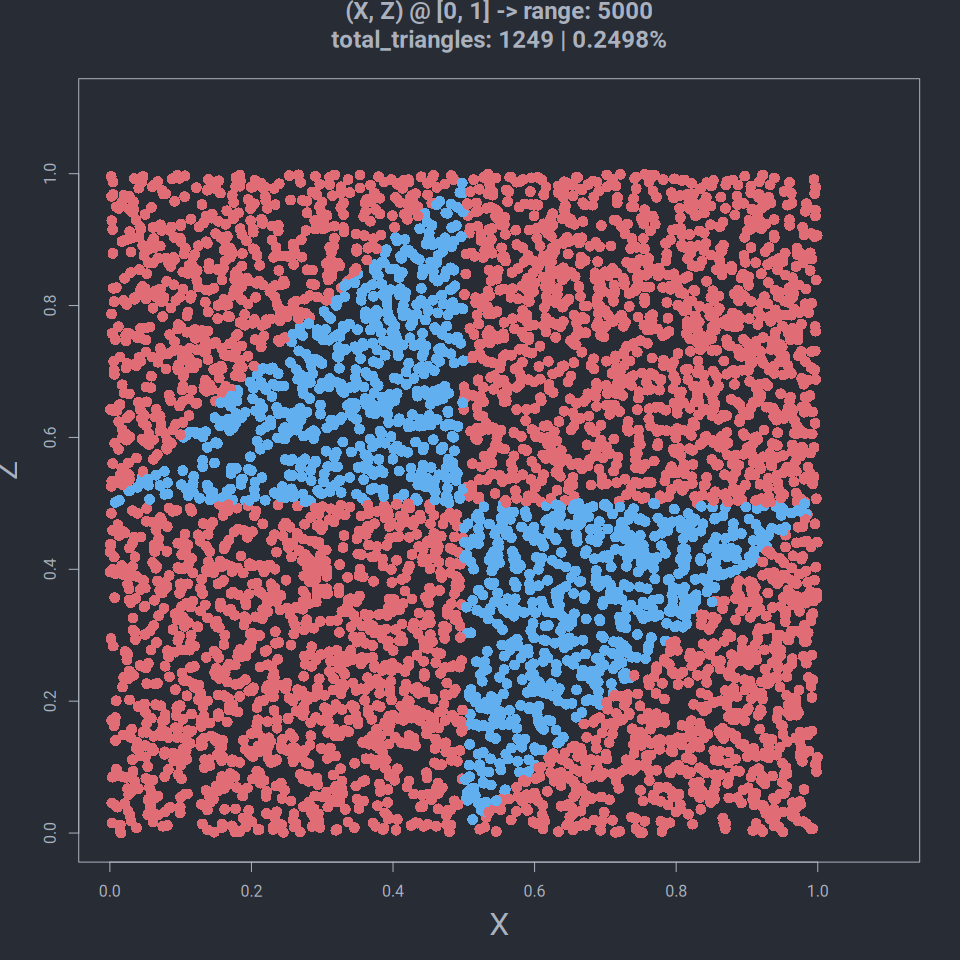
Generam valorile X si Z si le salvam in dataframe.

# generam valorile x si z  
xz <- generate\_xz()  
  
# cream dataframe cu tot ce am obtinut  
dataframe <- generate\_dataframe(  
 xz,  
 swap=swap)  
  
head(dataframe)

## x y a b c color triangle obtuse  
## 1 0.7641520 0.9930582 0.7641520 0.22890621 0.006941804 #E06C75 FALSE FALSE  
## 2 0.4441377 0.6708000 0.4441377 0.22666228 0.329200033 #61AFEF TRUE TRUE  
## 3 0.4639123 0.4345734 0.4345734 0.02933889 0.536087746 #E06C75 FALSE FALSE  
## 4 0.5272150 0.3610937 0.3610937 0.16612131 0.472785030 #61AFEF TRUE TRUE  
## 5 0.8084019 0.3376258 0.3376258 0.47077614 0.191598108 #61AFEF TRUE TRUE  
## 6 0.7323663 0.4755525 0.4755525 0.25681381 0.267633737 #61AFEF TRUE TRUE

Plotam valorile.

total\_triangles <- length(which(dataframe$triangle==TRUE))  
triangle\_prob <- get\_total\_triangles\_probability(dataframe, total)  
# adaugam titlu pentru plot  
main\_text <- paste(  
 "(X, Z) @ [0, 1] -> range: ",  
 total,  
 "\n",  
 "total\_triangles: ",  
 total\_triangles,  
 " | ",  
 triangle\_prob, "%",  
 "\n",  
 sep="")  
  
# punctul 3.a  
size <- 600  
white <- "#ABB2BF"  
# setam culoare de background si foreground pentru plot  
par(bg = "#282C34", fg=white)  
  
# plotam rezultatele finale  
plot(  
 x = dataframe$x,  
 y = dataframe$y,  
 xlab = "X",  
 ylab = "Z",  
 xlim = c(0,1.1),  
 ylim = c(0,1.1),  
 main = main\_text,  
 col.main = white,  
 col.sub = white, # Subtitle color  
 col.lab = white, # X and Y-axis labels color  
 col.axis = white, # Tick labels color  
 col = dataframe$color,  
 pch = 16,  
 cex = 1.5,  
 cex.main = 1.5, # Title size  
 cex.sub = 0.5, # Subtitle size  
 cex.lab = 2, # X-axis and Y-axis labels size  
 cex.axis = 1, # Axis labels size  
)



# Problema 2

Definim si functiile cu caciula in R.

f1\_caciula <- function(x1, x2) {  
 result <- exp((-1 / 2) \* (((x1 ^ 2) / 4) + (x2 ^ 2)))  
 return (result)  
}  
  
  
f2\_caciula <- function(y1, y2) {  
 # exp\_ <- exp((-1 / 2) \* (((y1 ^ 2) / 4) + (y2 ^ 2)))  
 exp\_ <- f1\_caciula(y1, y2)  
 cos\_sin <- (cos(y1) ^ 2) + (0.5 \* (sin(3 \* y2) ^ 2) \* (cos(y1) ^ 4))  
  
 return (exp\_ \* cos\_sin)  
}

Definim g(x) si functia raport r(x).

g <- function(x1, x2) {  
 result <- exp((-x1 / 2) - x2)  
 return (result)  
}  
  
raport\_f1 <- function(x1, x2) {  
 return (f1(x1, x2) / g(x1, x2))  
}  
  
raport\_f2 <- function(x1, x2) {  
 return (f2(x1, x2) / g(x1, x2))  
}

Si mai definim si functia densitate, i.e. f(x)

# f = f caciula \* c  
# fie c = 1  
f1 <- f1\_caciula  
f2 <- f2\_caciula

# Problema 2 - Punctul 1

### Solutie

# Problema 2 - Punctul 2

### Solutie

### Demonstratie matematica/teoretica

Matematic am demonstrat faptul ca pentru punctele x = (2, 1) raportul f(x)/g(x) este maxim. Iar M este e i.e. 2.71.

### Demonstratie practica

M1 <- raport\_f1(2, 1)  
  
  
raport\_M1 <- function(x1, x2) {  
 return (f1(x1, x2) / (M1 \* g(x1, x2)))  
}  
  
print(M1)

## [1] 2.718282

# Problema 2 - Punctul 3

Rejection sampling pentru functia f1

### Solutie

# esantion de lungime 100000  
x1\_accepted <- c()  
x2\_accepted <- c()  
n <- 100000  
total\_repeats <- 0  
for (i in 1:n) {  
 # acceptance\_prob  
 repeat {  
 u1 <- runif(1)  
 u2 <- runif(1)  
 # x1 <- -(1/2) \* log(rexp(1))  
 # x2 <- log(rexp(1))  
  
 x1 <- 1 + sqrt(-log(u1))  
 x2 <- 1 + sqrt(-5.4 \* log(u2))  
  
 total\_repeats <- total\_repeats + 1  
  
 if((u1 \* u2) < raport\_M1(x1, x2)) {  
 break  
 }  
 }  
 x1\_accepted <- append(x1\_accepted, x1)  
 x2\_accepted <- append(x2\_accepted, x2)  
}  
acceptance\_prob <- n / total\_repeats  
print(acceptance\_prob)

## [1] 0.3621286

Probabilitatea de acceptare este approx 0.36% intotdeauna, adica 1/M, care M este e.

# Problema 2 - Punctul 4

### Repartitia marginala X1 si X2

### Solutie

# Problema 2 - Punctul 5

### Indicatie

### Repartitia empirica X1 si X2

### Solutie

# Bibliografie

* <https://www.statology.org/remove-element-from-vector-r/>
* <https://stackoverflow.com/questions/13841599/calculate-frequency-of-occurrence-in-an-array-using-r>
* <https://r-coder.com/plot-r/#R_plot_color>
* <https://www.tutorialspoint.com/how-to-change-the-background-color-of-a-plot-created-by-using-plot-function-in-r>
* <https://r-charts.com/base-r/background-color/>
* <https://plotly.com/r/shapes/>
* <https://www.rdocumentation.org/packages/LaplacesDemon/versions/16.1.6/topics/RejectionSampling>
* <https://www.youtube.com/watch?v=6r-2N_bJdck>
* <https://stackoverflow.com/questions/8708243/import-stuff-from-a-r-file>
* <https://github.com/REditorSupport/languageserver#installation>
* <https://github.com/sublimelsp/LSP-R>
* <https://r-lang.com/r-append-to-list-how-to-append-element-in-r-list/>
* <https://www.w3schools.com/r/r_for_loop.asp>
* <https://stackoverflow.com/a/66996483/12172291>
* <https://www.tutorialspoint.com/how-to-create-a-random-sample-of-values-between-0-and-1-in-r>
* <https://www.w3schools.com/r/r_graph_plot.asp>
* <https://stackoverflow.com/questions/20454789/how-do-i-tell-r-to-fill-the-circle-dots-with-colour-on-a-scatter-plot>
* <https://www.statology.org/r-count-number-of-occurrences-in-column/>
* <https://www.statology.org/random-sample-in-r/>
* <https://rmarkdown.rstudio.com/lesson-9.html>
* <https://www.tutorialspoint.com/r/r_data_frames.htm>
* <https://www.youtube.com/watch?v=kMb4JlvuGlw>
* <https://stackoverflow.com/questions/12162278/appending-column-to-a-data-frame-r>
* <https://www.tutorialspoint.com/r/r_strings.htm>
* <https://stackoverflow.com/questions/23709060/change-r-markdown-plot-width>
* <https://yihui.org/knitr/options/#chunk_options>
* <https://www.youtube.com/watch?v=Dzqe1a_AIkc>
* <https://www.vedantu.com/formula/probability-formula>

Si multe alte linkuri, pentru ca nu ai cum sa salvezi ficare link pe care intri pentru documentatie, sunt sute accesate.