Trabalho Estatística Computacional

Professor: Mário de Castro

Alunos: Gabriel Couto Tabak 10276995
Lucas Albano de Oliveira 10377688
Daniel Eiji Martins Chiyo 10276960

Data: 11/03/2020

Introduction

In this assignment, we solve three different questions from the exercise list. All of the questions are related to finding pseudo-random samples from a random variable with a distribution function given. Then, We do a Kolmogorov-Smirnov test, for each sample we take (if we are dealing with continuous function) or Chi-squared test (for discrete function). Therefore, it will be possible to see if the sample was generated from the distribution. For each exercise, we implement an algorithm in the R environment.

The first question shows us a method to generate n observations from a Normal distribution with mean zero and standard deviation one. So we justify why this method works and implement it. As well as do a Kolmogorov-Smirnov test.

The second question deals with the Poisson distribution, we are presented with a method to generate samples from a Poisson(lambda). In this case, we take different lambdas and get samples from all the lambdas. Then we are going to do the Chi-squared test. This way, it will be possible to understand if the method works well when changing the Poisson parameter.

The third question deals with the triangular density function, as a density function. We implement two different methods to generate random samples from this distribution. The first method is the inverse transform sampling. The second method is the acceptance-rejection method.

Methodology

Please note that if one wishes to replicate our work, you must use the R version 3.6.1 and the seed 3110. All the plots were done with the library ggplot2 from R (it's required to install it before running the program). All the times we took a random uniform sample, we used the runif function from R. The plots from the samples we took, the y-axis represents the density of the number (how many times it appears compared to the total number of samples) and the x-axis represents the number itself. In the discrete variable, we used the frequency.

In all the exercises we test the samples with a different number of observations, and check if increasing the number of observations per sample affects the proximity with the distribution used.

One method to generate random samples from a random variable, that we use is the inverse transform sampling.

In the method, we take the distribution of the random variable and calculate its cumulative function. The cumulative is measure by taking the integral of the distribution function, with the limits integration from minus infinite to x. (Devore)

$$F(x) = \int_{-\infty}^{x} f(y)\, dy$$

The next step is to get the inverse function of F(x), that we call $F^{-1}(x)$. Therefore, to get n samples we take n random samples from the uniform distribution and apply them in the inverse of the cumulative function. (Gentle)

Another method used is the acceptance-rejection method. Here we need an auxiliary random variable Y, with probability density, equals g(y). It is important that we can take random samples from Y, in order for the method to work. The method consists of 3 steps:

1 - generate a random sample from Y, called y.
2 - generate a random sample from the uniform distribution (0,1), called u.
3 - check u <= $\frac{f(y)}{Mg(y)}$, if it is true return y as the sample, otherwise, return to step one

M is a constant to assure that $\frac{f(y)}{Mg(y)}$ is majorized by 1, and the closer Mg(y) is to f(y), the faster the algorithm will be. (Gentle). With this in mind, we measure M by,

$M = sup_x\{\frac{g(x)}{f(x)}\}$. (Sigman)

This way, to generate n samples, we just repeat the method n times.

At last, we need some way to check if the random sample is truly taken from the desired random. With this in mind, we have two tests that can help to check the distribution of the samples generated.

One is the Chi-squared test, which is related to discrete distributions. Here the test compares the frequency of the sample (how many times each number appears in the random sample we took) to the expected frequency (the frequency we hope to acquire from a particular distribution). The expected frequency is computed by taking the number of samples and multiplying it by the probability of each number from the domain. Here we test the null hypothesis that the sample has the distribution we are working with (Maynooth University).

The other test we have is the Kolmogorov-Smirnov test, which is used for continuous functions. It compares the distance of two distributions functions. In this case, one of the functions will be the empirical distribution function for the random sample and the other the distribution function we seek. (Chakravart)

Results and discussions

Exercise 1

We are dealing with one method to generate random samples from a normal distribution (in our case, mean = 0 and standard deviation = 1). The method takes 12 random samples from the uniform distribution (0,1), add them, and deduct 6. After all this, we have generated a new sample.

$$U_{i,1},\ U_{i,2},\ \dots,\ U_{i,12} \sim Uniform(0,1)$$

$$X_i = \sum_{j=1}^{12} U_{i,j} - 6\ ,\ i = number\ of\ observations$$

The method works because of the central limit theorem (CLT). The CLT states that the sum of independent and identically distributed variables from any distribution (with finite variance) converges to a normal distribution as the number of variables in the sum grows. (Rouaud)

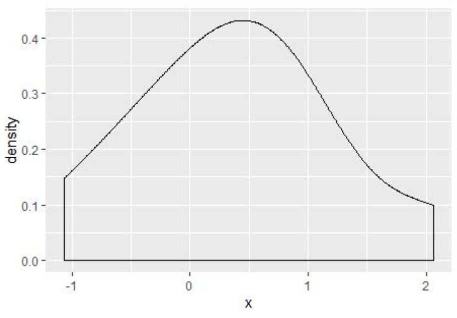
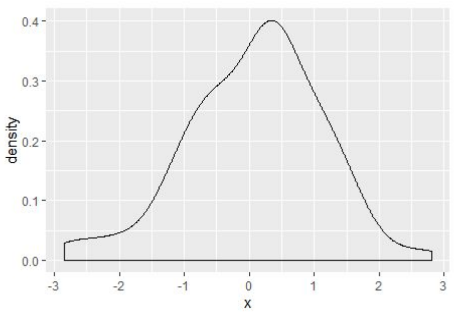
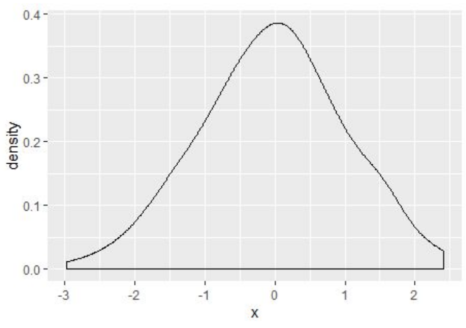
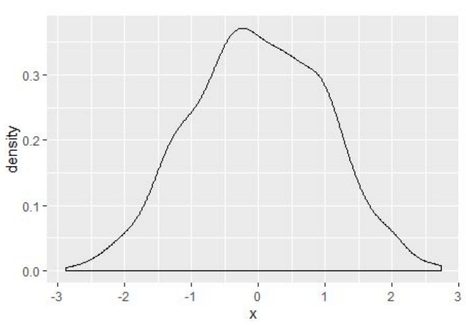
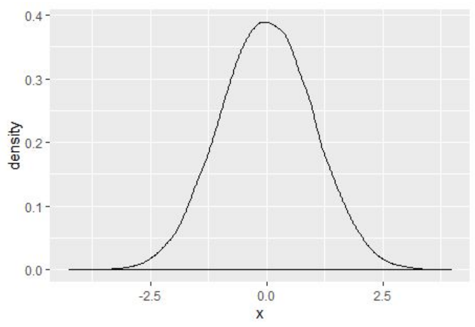
If the speed of convergence is fast, then, if we sum twelve uniform variables we should have a normal distribution. We can test that by generating these uniform variables adding them up and testing whether the transformed variable (sum) has a normal distribution.

The mean of the U(0,1) is 1/2 and the variance is 1/12. (Stephanie) Therefore

$$E(X) = E(\sum_{j=1}^{12} U_{i,j} - 6) = \tfrac{12}{2} - 6 = 0 \text{ and } VAR(X) = VAR(\sum_{j=1}^{12} U_{i,j}) = 12/12 = 1$$

Our analytical results suggest that the mean of our variable X is 0 and its variance is 1.

So, we implemented the method in R and generated some plots (the empirical distribution plot), with a different number of observations, n = 10, 100, 200, 1000 and 100000. The graphics are shown in order from 10 to 100000. It is possible to see that as the number of samples grows, the density graphic gets more similar to the normal distribution. And we also took the p-value for each sample, from the ks-test (as we are working with the normal distribution, a continuous function).

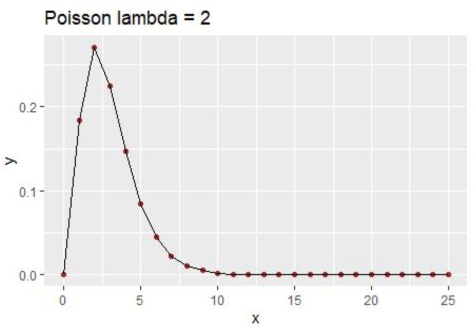






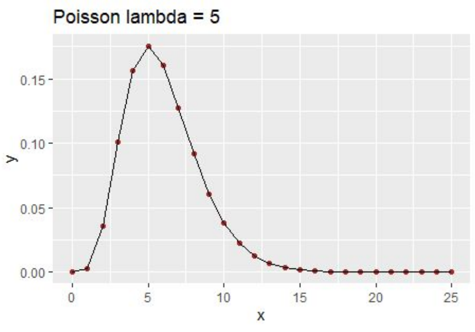
observations	10	100	200	1000	100000
p − value	0.365408	0.281899	0.306659	0.959311	0.232448

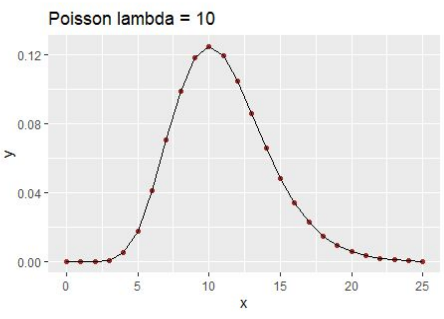
As we can see, all the p-values are greater than 0.05, which means we cannot reject the null hypothesis that the empirical distribution is Normal(0,1).

Exercise 2

This question presents a method to generate random samples from a Poisson distribution where the lambda variable is greater than zero and we can change it for different lambdas. The Poisson distribution is a discrete function, and for lambdas equals to 2, 5 and 10, we have the following graphs. As the Poisson distribution is discrete, the function has values only for the points in brown, the line serves only as a guide.







Now that we have an understanding of this distribution, we are going to take the random samples from it. The algorithm works in three steps, as explained below.

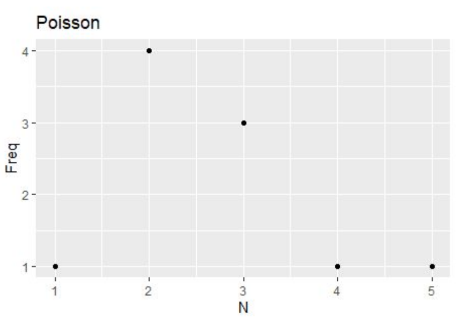
1- First take two variables P = 1 and N = 0

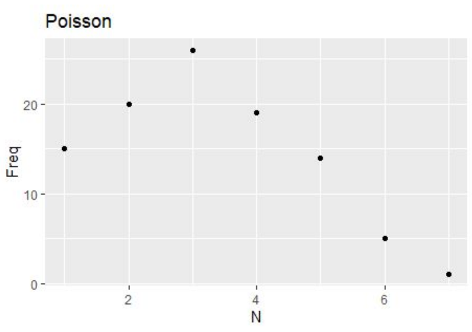
2- Get a random sample, U, from a uniform distribution, make P = P*U and N = N+1 until P < $e^{-\lambda}$. The λ here is chosen previously, as we said, in this case, we took λ =2, 5 and 10.

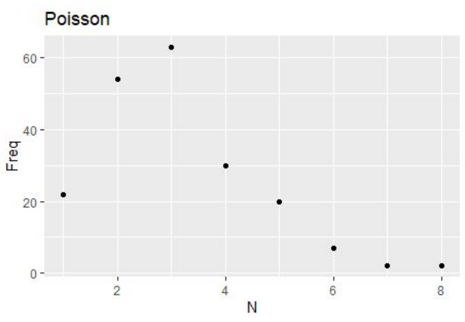
3- $X = N - 1 \sim Poisson(\lambda)$.

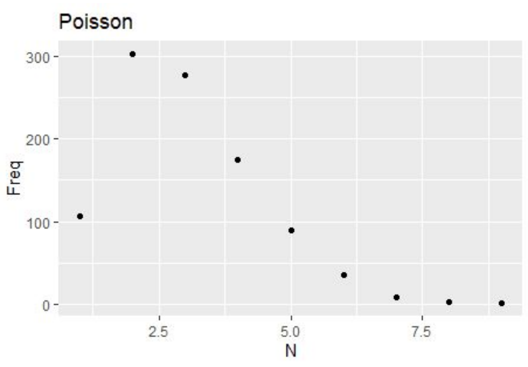
We implement this algorithm and vary the number of observations n= 10, 100, 200, 1000 and 100000. The five graphics for each λ, that we present below, show the progression of results when the sample is increasing. They all are in order, from 10 to 100000 (the last graphic for each λ).

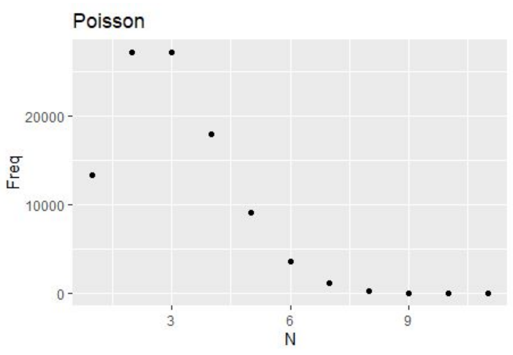
With $\lambda = 2$



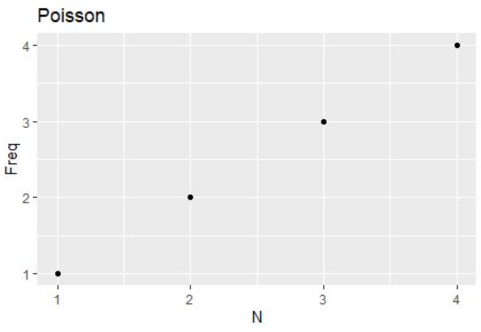


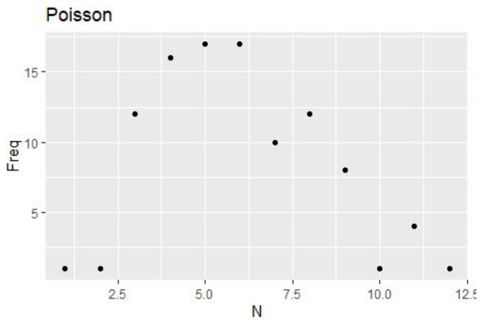


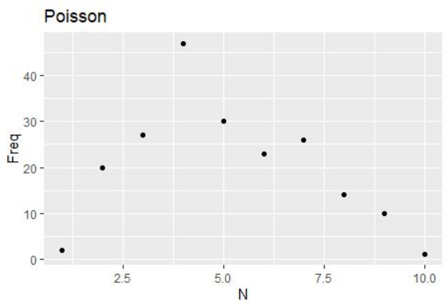


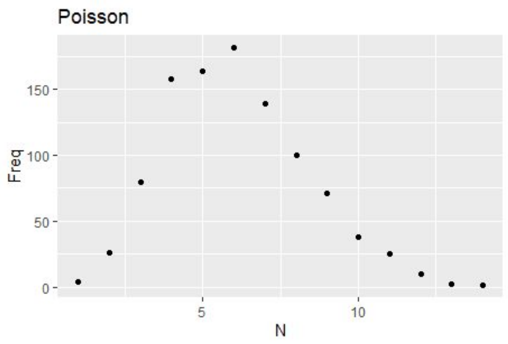


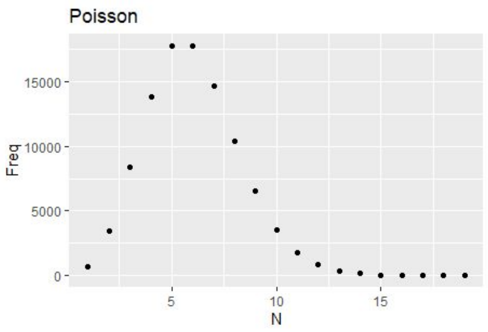
With λ = 5



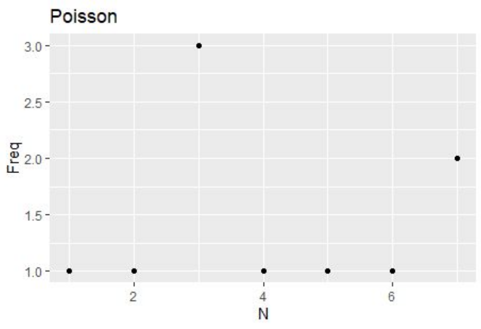


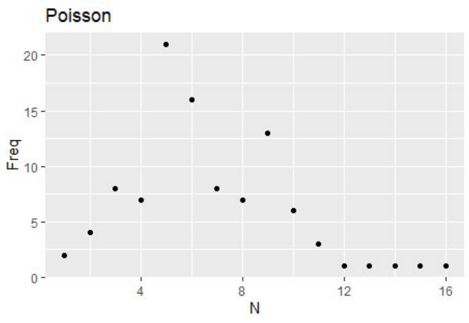


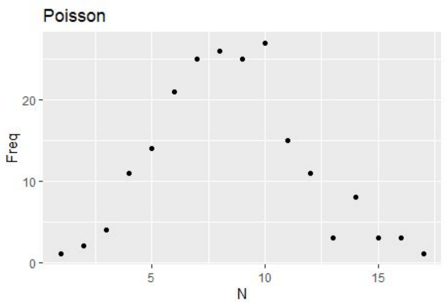


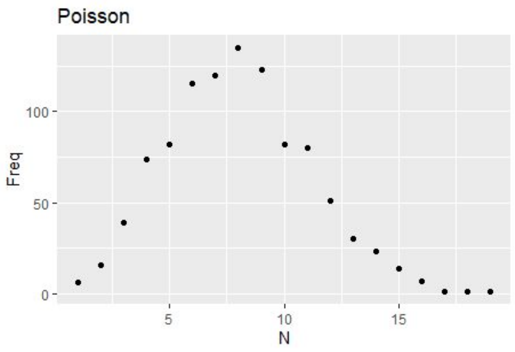


With λ = 10









Therefore we calculate the Chi-square test to test if the distribution is a Poison with parameter λ (null hypothesis). With a confidence level of 95%, if the p-value is 5% or lower, we will reject the null hypothesis. In our case, we don't want to reject the null hypothesis as we know our empirical distribution is Poison.

$p - value \setminus observations$	10	100	200	1000	100000
$\lambda = 2$	0.265025	0.226962	0.242638	0.230256	0.232204
$\lambda = 5$	0.213309	0.286091	0.231341	0.233982	0.235610
$\lambda = 10$	0.300708	0.282182	0.270972	0.253831	0.236898

As we can see, all the p-values are greater than 0.05, which means we cannot reject the null hypothesis that the empirical distribution is Poisson with parameter λ.

Exercise 3

Method 1

Here we have the function $f(x) = max(0, 1 - |x|)$, and we want to get a random sample from this function. The first method will be the inverse transform function. So, we need the cumulative function of f(x) and the inverse transform of it.

First, we need to understand the density function we are working on, we can divide f(x) into 4 intervals. If the absolute value of x is greater than one, 1 - |x|, will be a negative value, so f(x) = 0. This way, we have the following intervals.

$f(x) = 0, x <= -1$
$f(x) = 1 + x, -1 < x < 0$
$f(x) = 1 - x, 0 <= x < 1$
$f(x) = 0, 1 <= x$

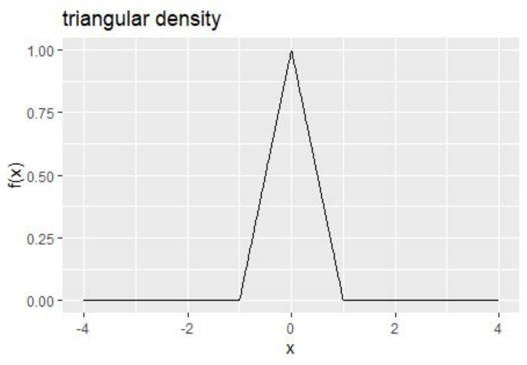
Now, we measure the cumulative function of f(x). Here it's possible to see that, as the integral to measure cumulative function starts from minus infinity and goes to x, we had to do different integrals for each interval of x.

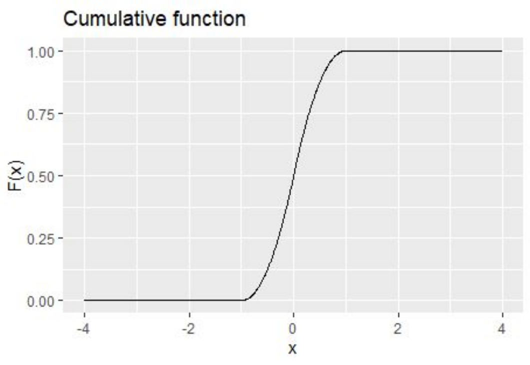
$F(x) = 0, x <= -1$
$F(x) = \frac{x^2}{2} + x + \frac{1}{2}, \ -1 < x < 0$
$F(x) = -\frac{x^2}{2} + x + \frac{1}{2}, \ 0 <= x < 1$
$F(x) = 1, \ 1 <= x$

The next step is taking the inverse transformation of the cumulative function. As we know, the cumulative function range is from 0 to 1, so, in order to take the inverse, we need some restrictions.

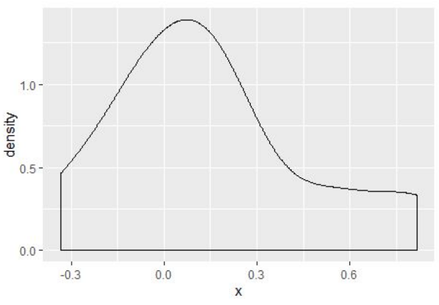
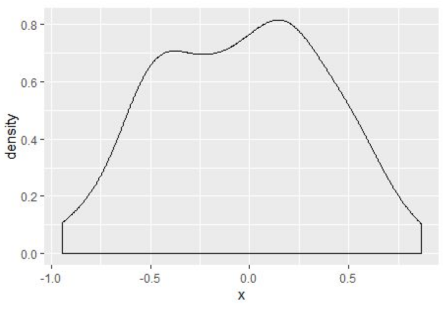
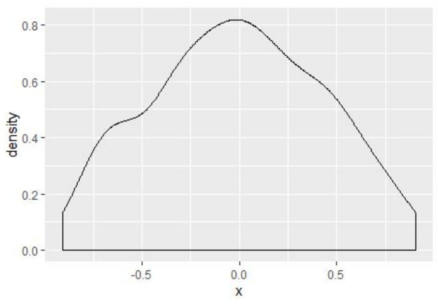
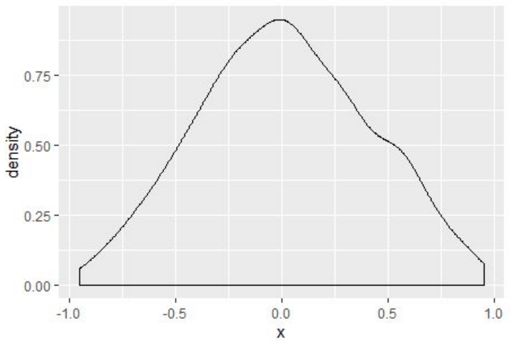
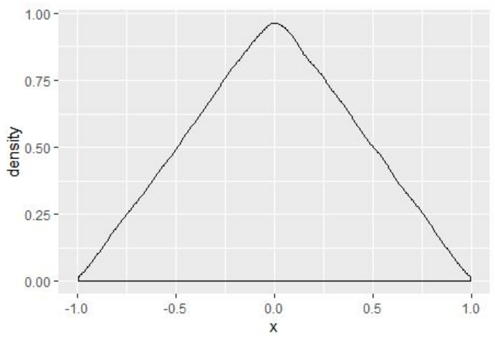
$F^{-1}(x) = -1 + \sqrt{2x}, \ 0 < x < \frac{1}{2}$
$F^{-1}(x) = 1 - \sqrt{2 - 2x}, \ \frac{1}{2} < x < 1$

We made some plots to help the understanding of the function.





Proceeding with the exercise, we took n samples from the uniform distribution and applied them to the inverse function ($F^{-1}(x)$). This way, we generated n samples. We choose n as 10, 100, 200, 1000 and 100000. This exercise produces the following graphics:

Then, we did the ks test for the samples and find the following results.

observations	10	100	200	1000	100000
$p - value$	0.384461	0.495058	0.306659	0.947899	0.20750

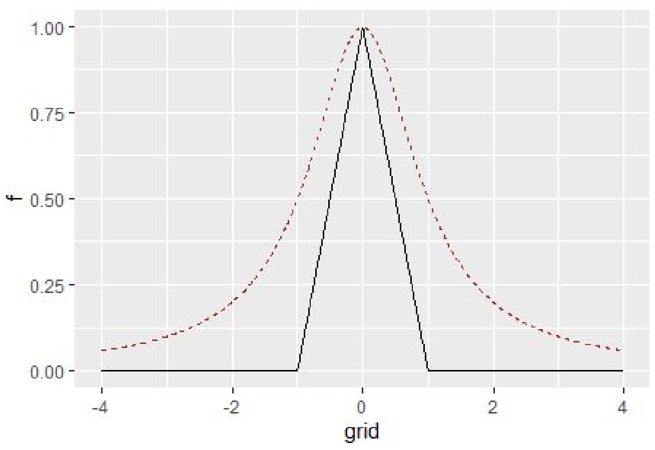
As we can see, all the p-values are greater than 0.05, which means we cannot reject the null hypothesis that the empirical distribution is the triangular density.

Method 2

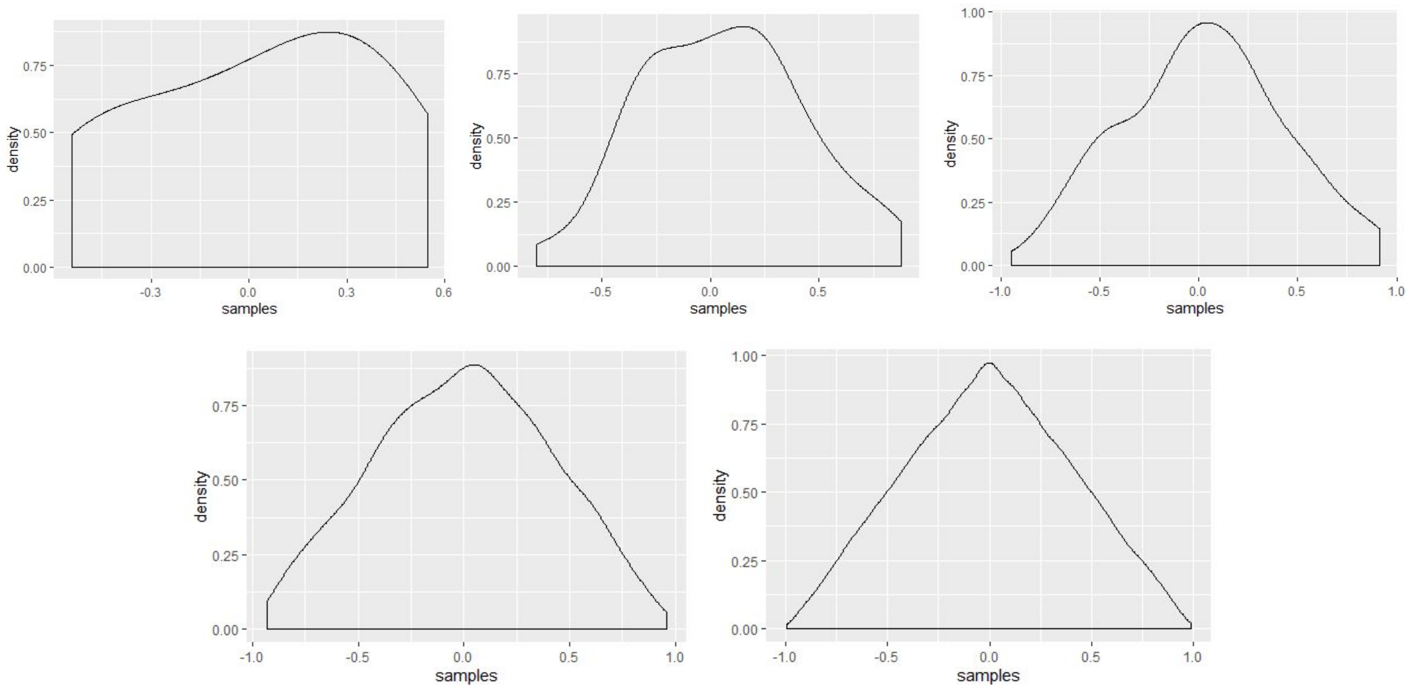
The second method we used was the acceptance-rejection method. As we explained before, this method needs an auxiliary function. We choose the Cauchy function, and to get a random sample from it, we use the inverse transform method.

$$g(x) = \frac{1}{\pi(1+x^2)} , \; G^{-1}(x) = tan(\pi + \tfrac{3}{2})$$

The next step is finding M, the constant, which is measured by the supreme of f(x)/g(x). R gave us that M = 3.129. With this, we plotted the function M*g(x), the dashed line, and the f(x).



This way, we took the samples from 10, 100, 200, 1000 and 100000 observations.



Then, we did the ks test for the samples and got the following results.

observations	10	100	200	1000	100000
$p - value$	0.945658	0.370683	0.564421	0.876914	0.261716

As we can see, all the p-values are greater than 0.05, which means we cannot reject the null hypothesis that the empirical distribution is the triangular density.

Conclusion

Overall, we could test the generation of random samples from random variables with different density functions, even discrete functions. And after all those tests, we saw that the null hypothesis shouldn't be discarded for most of the samples. And it was also possible to see graphically the proximity form the density function and the density of the samples.

References

1. Devore, J. L.(2010). Probability and Statistics for Engineering and the Sciences. 8th edition. Retrieved from https://www.colorado.edu/amath/sites/default/files/attached-files/ch4.pdf
2. Sigman, K. (2007). Acceptance-rejection method. Retrieved from https://www.colorado.edu/amath/sites/default/files/attached-files/ch4.pdf
3. Gentle, J. E. (2003). Random number generation and Monte Carlo methods. (2nd edition). New York, NY: Springer
4. Chakravart, Laha, & Roy (1967). Kolmogorov-Smirnov Goodness-of-fit Test Retrieved from https://www.itl.nist.gov/div898/handbook/eda/section3/eda35g.htm
5. Maynooth University. Chi-Squared Tests [Slides] Retrieved from http://www.thphys.nuim.ie/Notes/EE304/Notes/LEC14/ChiSlide.pdf
6. Rouaud, Mathieu (2013). *Probability, Statistics and Estimation* (PDF). Retrieved from http://www.incertitudes.fr/book.pdf
7. Stephanie (2013). Uniform distribution / Rectangular distribution: What is it? Retrieved from https://www.statisticshowto.datasciencecentral.com/uniform-distribution/

Codes in R.

```
#EXERCISE 1
library(ggplot2)

generator <- function(n){
  Sample <- rep(0,n)
  for(k in 1:n){
    U <- runif(12)
    Sample[k] <- sum(U) - 6
  }
  return(Sample)
}

plotSample <- function(x){
  df <- as.data.frame(x)
  names(df)[1] <- "x"
  print(ggplot(df,aes(x)) + geom_density())
  print(ks.test(x,"pnorm",0,1)$p.value)
}

n <- c(10,100,200,1000,100000)
set.seed(3110)
AllSamples <- sapply(n, generator)
sapply(AllSamples,plotSample)
```

```
#EXERCISE 2
library(ggplot2)

#generator function
generator <- function(n,lambda){
  S <- as.data.frame(rep(0,n))
  names(S)[1] <- "sample"

  for(i in 1:n){
    P <- 1; N <- 0
    while (1) {
      U <- runif(1)
      P <- P*U
      N <- N+1
      if(P<exp(-lambda)){
        S$sample[i] <- N-1
        break
      }
    }
  }
return(S)
}

#plots and chi sq test
Allplots <- function(Plots){
  comparison <- as.data.frame(table(Plots))
  comparison$Plots <- as.numeric(comparison$Plots)
  comparison$Expectedf <- dpois(comparison$Plots,lambda)*sum(comparison$Freq)

  ggobj <- ggplot(comparison) +
  geom_point(aes(Plots,Freq)) +
  ggtitle("Poisson")+
  labs(x = "N")
  print(ggobj)
  print(chisq.test(comparison$Freq,comparison$Expectedf)$p.value)
}

set.seed(3110)
#lambda poisson
lambda <- c(2,5,10)
#number of samples
n <- c(10,100,200,1000,100000)
```

```
AllSamples <- sapply(n,generator,lambda[1])
sapply(AllSamples,Allplots)

AllSamples <- sapply(n,generator,lambda[2])
sapply(AllSamples,Allplots)

AllSamples <- sapply(n,generator,lambda[3])
sapply(AllSamples,Allplots)


#plots Poisson with different lambda
grid <- 0:25
DfPoisson <- as.data.frame(grid)
DfPoisson$two <- dpois(lambda[1],grid)
DfPoisson$five <- dpois(lambda[2],grid)
DfPoisson$ten <- dpois(lambda[3],grid)

#Poisson lambda 2
ggplot(DfPoisson,aes(grid,two))  + geom_point(col = "brown")+ ggtitle("Poisson lambda = 2") +
  labs(x= "x",y= "y") + geom_line()
#Poisson lambda 5
ggplot(DfPoisson,aes(grid,five))  + geom_point(col = "brown")+ ggtitle("Poisson lambda = 5") +
  labs(x= "x",y= "y") + geom_line()
#Poisson lambda 10
ggplot(DfPoisson,aes(grid,ten))  + geom_point(col = "brown")+ ggtitle("Poisson lambda = 10") +
  labs(x= "x",y= "y") + geom_line()
```

```
#EXERCISE 3
##########              Method1       ####################
#function we are working on f(x) = max(0, 1 - |x|)
fx <- function(x){
  k <- 1 - abs(x)
  return(ifelse(k>0,k,0))
}

#cumulative function of f(x)
CummF <- function(x){
  condition1 <- x<0 & x> -1
  condition2 <- x>=0 & x<1
  condition3 <- x>= 1
  condition4 <- x<= -1
  x <- ifelse(condition1,x^2/2 + x + 1/2,x)
  x <- ifelse(condition2,-x^2/2 + x + 1/2,x)
  x <- ifelse(condition3,1,x)
  x <- ifelse(condition4,0,x)
  return(x)
}

#inverse of cumulative function
InverseF <- function(x){
  condition1 <- x<1/2 & x>= 0
  condition2 <- x>= 1/2 & x<=1
  x <- ifelse(condition1, -1 + sqrt(2*x),x)
  x <- ifelse(condition2,1 - sqrt(2*(1-x)),x)
  return(x)
}

#generate the samples
generator <- function(n){
  sample <- rep(0,n)
  for (i in 1:n) {
    U <- runif(1)
    sample[i] <- InverseF(U)
  }
  return(sample)
}
```

```
set.seed(3110)
n <- c(10,100,200,1000,100000)
AllSamples <- sapply(n,generator)

plotSamples <- function(x){
  x <- as.data.frame(x)
  names(x)[1] <- "x"
  print(ggplot() + geom_density(data = x,aes(x)))
  print(ks.test(x,CummF)$p.value)
}

sapply(AllSamples,plotSamples)

#Plots from cumulative function and f(x)
grid <- seq(-2,2,length = 100)
df <- as.data.frame(grid)
df$f <- fx(grid)
df$CumF <- CummF(grid)
ggplot(df,aes(grid,f)) + geom_line() + ggtitle("triangular density") + labs(y= "f(x)", x = "x")
ggplot(df,aes(grid,CumF)) + geom_line() + ggtitle("Cumulative function") + labs(y= "F(x)", x =
"x")

##########          Method2      ####################

#Auxiliary function Cauchy, density
dCauchy <- function(x){
  return(1/(pi*(1+x*x)))
}

#Inverse transform of cumulative function of Cauchy
IFCauchy <- function(x){
  return(tan(pi*(x+3/2)))
}

#Take the sample
sampleCauchy <- function(){
  k<- runif(1)
  return(IFCauchy(k))
}

#f(x) function
fx <- function(x){
  k <- 1 - abs(x)
  return(ifelse(k>0,k,0))
}

#This methods need to compute M, which is max(f(x)/g(x))
#Function f(x)/g(x)
fgx <- function(x){
  return(fx(x)/dCauchy(x))
}

grid <- seq(-4,4,length = 1000)
M <- max(fgx(grid))

#plot f(x) and Mg(x)
df <- as.data.frame(grid)
df$f <- fx(grid)
df$Mgx <- M*dCauchy(grid)
ggplot(df) + geom_line(aes(grid,f)) + geom_line(aes(grid,Mgx),col="brown",linetype = 2)

#take the sample
generator2 <- function(n){
  sample <- rep(0,n)
  for(i in 1:n){
    while(1){
      y <- sampleCauchy()
      u <- runif(1)
      if(u<=fgx(y)/M){
        sample[i] <- y
        break
      }
    }
  }
  return(sample)
}

plotSamples2 <- function(x){
  x <- as.data.frame(x)
  names(x)[1] <- "samples"
  print(ggplot(x)+
          geom_density(aes(samples)))
  print(ks.test(x,CummF)$p.value)
}
```

```
n <- c(10,100,200,1000,100000)
set.seed(3110)
AllSamples2 <- sapply(n,generator2)
sapply(AllSamples2,plotSamples2)

ggplot(df) + geom_density(aes(sample)) + geom_line(aes(grid,f),col = "brown",linetype=2)
```