R Lab. - Exercise 4

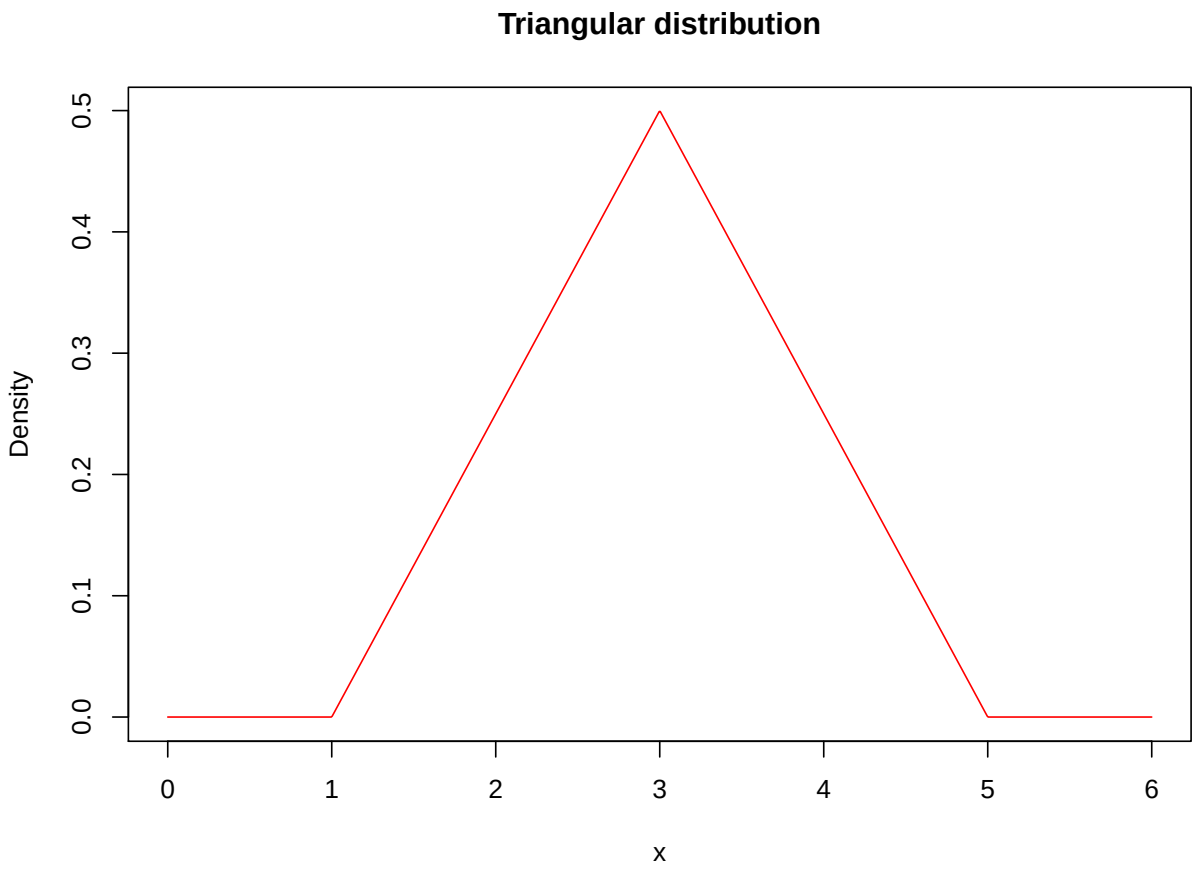
Michele Guadagnini - Mt. 1230663

May 9, 2020

Exercise 1 - Triangular distribution

A) Plot the triangular pdf

```
# definition of the distribution
dtriang <- function(x,a,b,c){
    y <- ifelse( (x>=a & x<c),
                 2*(x-a)/((b-a)*(c-a)),
                 ifelse( (x>=c & x<=b),
                         2*(b-x)/((b-a)*(b-c)),
                         0) )
    return (y)
}

A <- 1
B <- 5
C <- 3
x <- seq(A-1,B+1,length=1000)
plot(x, dtriang(x,A,B,C), type='l', col='red', main="Triangular distribution",
     ylab="Density")
```

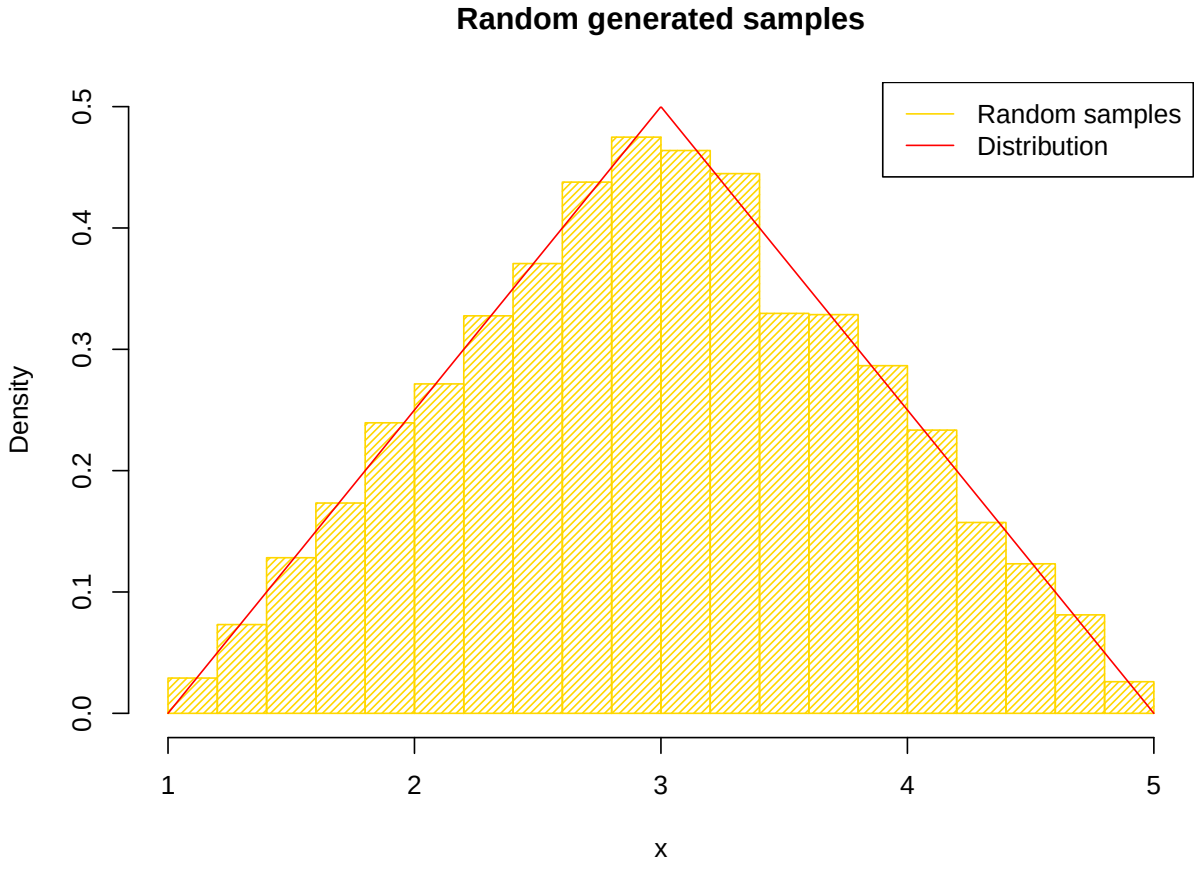


B) Write an algorithm to generate random numbers from the triangular distribution

```
rtriang <- function(n,a,b,c) {
    x1 <- runif(n, a, b)
    u2 <- runif(n, 0, 1)
    z <- ifelse(u2*dtriang(c,a,b,c) < dtriang(x1,a,b,c), x1, NA)
    return (z)
}
```

C) Generate 10^4 random numbers from the distribution, show them in an histogram and superimpose the analytical curve

```
nsamples <- 10^4
rands <- rtriang(nsamples,A,B,C)
eff <- sum(is.na(rands))/length(rands)

hist(rands, density=30, col='gold', main="Random generated samples",
     freq=FALSE, ylim=c(0,0.5), xlab="x", breaks=20)

x <- seq(A, B, length=1000)
lines(x, dtriang(x,A,B,C), col='red', type='l')
legend("topright", legend=c("Random samples","Distribution"),
       col=c("gold", "red"), lty=1:1)
```

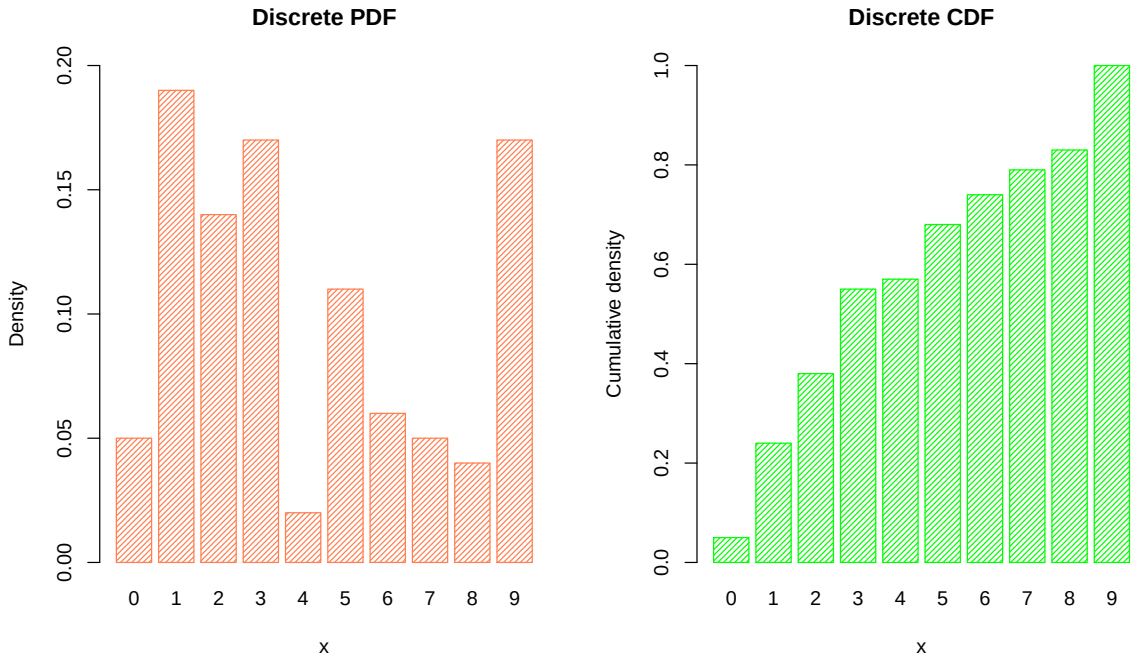


The efficiency of the random generation is 0.5009.

Exercise 2 - Discrete probability distribution

A) Plot the probability density function and the cumulative density function

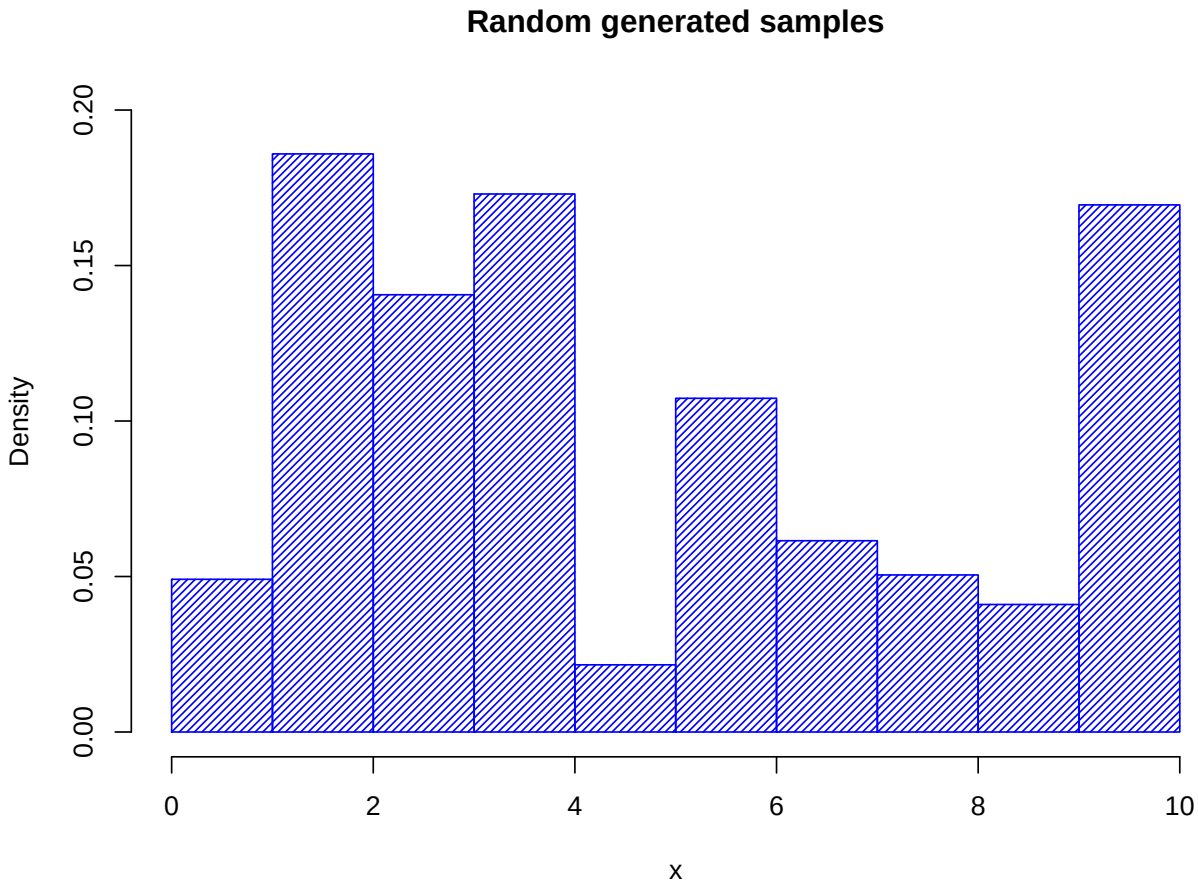
```
probs <- c(0.05, 0.19, 0.14, 0.17, 0.02, 0.11, 0.06, 0.05, 0.04, 0.17)

par(mfrow=c(1,2))
ddiscrete <- probs
x <- 1:length(ddiscrete)-1
barplot(ddiscrete, names=x, ylim=c(0,0.2), col='coral', density=30, border=TRUE,
        xlab='x', ylab="Density", main="Discrete PDF")

pdiscrete <- cumsum(probs)
barplot(pdiscrete, names=x, ylim=c(0,1), col='green', density=30, border=TRUE,
        xlab='x', ylab="Cumulative density", main="Discrete CDF")
```



B) Write an algorithm to generate random numbers from the discrete probability distribution

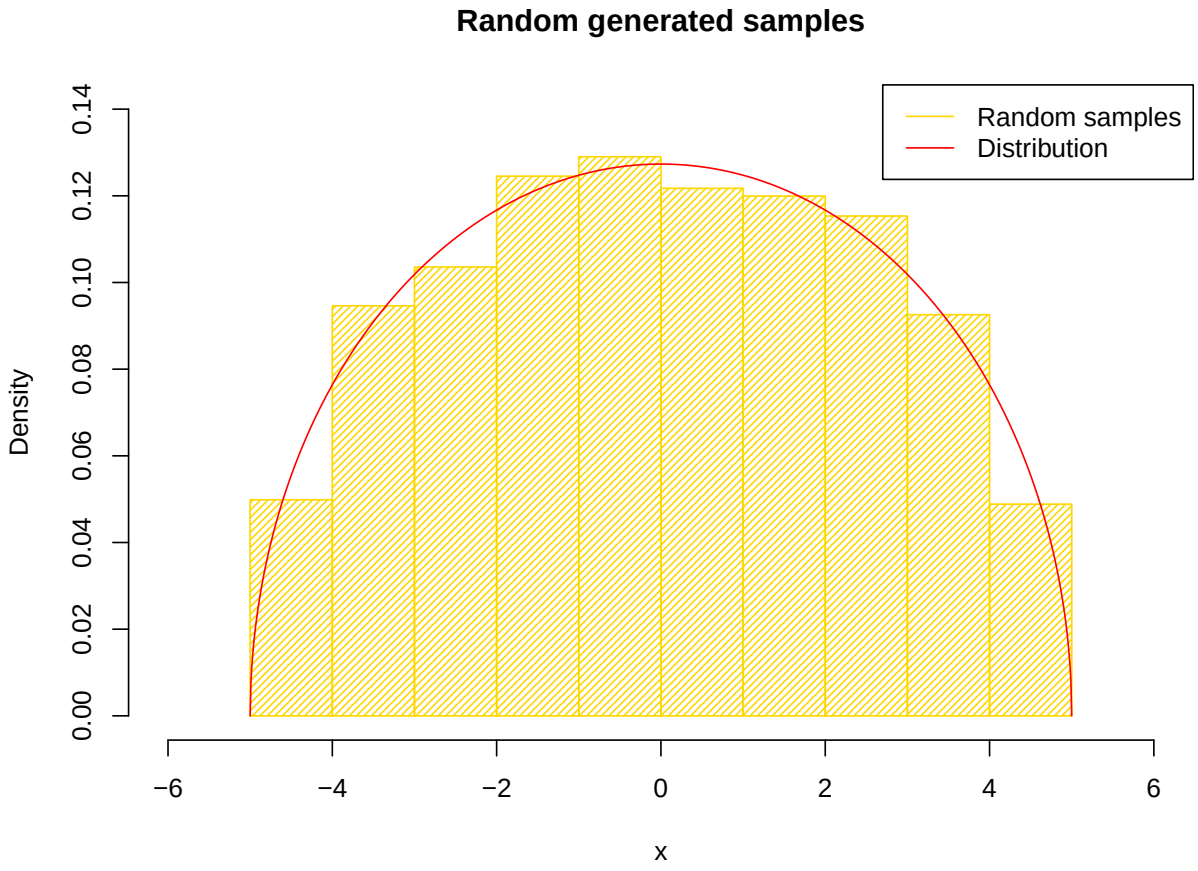
```
rdiscrete <- function(n, probs) {
    ps <- c(0,cumsum(probs))
    us <- runif(n, 0,1)
    z <- NULL
    for (i in 1:n) {
        for (j in 2:length(ps)) {
            if (us[i]>ps[j-1] & us[i]<=ps[j]) {
               z <- c(z,j-1)
               break
            }
        }
    }
    return (z)
}
```

```r
hist(rdiscrete(10000, probs), breaks=0:10, freq=FALSE, col='blue', density=30,
     xlab='x', main="Random generated samples", ylim=c(0,max(probs)*1.05))
```



Exercise 3 - Generate random variables from a distribution using the acceptance-rejection algorithm

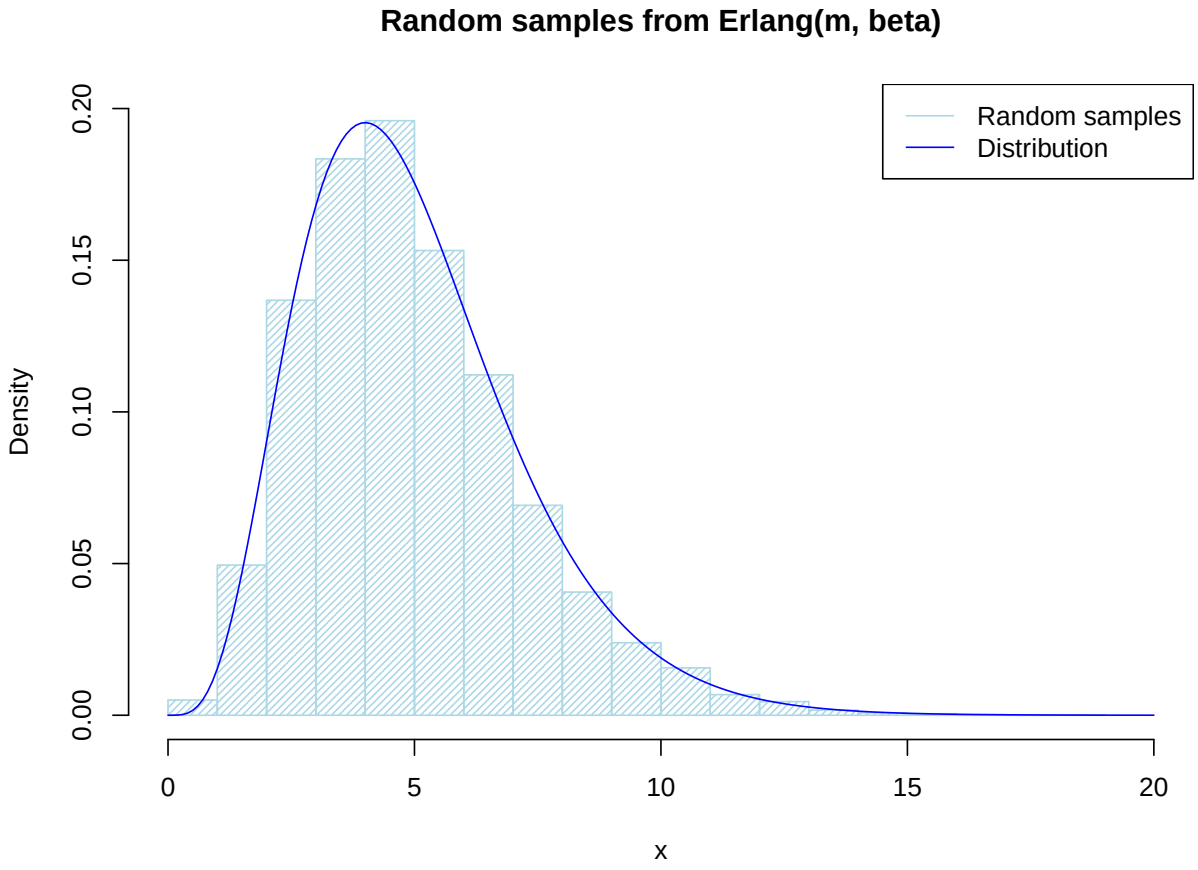
```
dfunc <- function(x, R) {
    z <- ifelse( (x<=R)&(x>=-R), (2/(pi*R^2))*sqrt(R^2-x^2), 0 )
    return(z)
}
rfunc <- function(n, R) {
    x1 <- runif(n, -R, R)
    u2 <- runif(n, 0, 1)
    M <- 2/(pi*R)
    z <- ifelse(u2*dfunc(M,R) < dfunc(x1,R), x1, NA)
    return (z)
}

n <- 10^4
R <- 5
x <- seq(-R,R,length=1000)
rands <- rfunc(n, R)
eff <- sum(is.na(rands))/length(rands)
hist(rands, xlim=c(-R-1,R+1), col='gold', density=30, ylim=c(0,0.14),
     main="Random generated samples", xlab="x", freq=FALSE)
lines(x, dfunc(x,R), type='l', col='red', main="PDF", ylab="Density")
legend("topright", legend=c("Random samples","Distribution"),
       col=c("gold", "red"), lty=1:1)
```



The efficiency of the algorithm is 0.2179.

Exercise 4 - Write an algorithm to sample variables from an Erlang distribution

```
rerlang <- function(n, m, beta) {
    y <- NULL
    for (i in 1:n) {
        y <- c( y,-beta*log(prod(runif(m,0,1))) )
    }
    return(y)
}

n <- 10^4
m=5
beta=1
hist(rerlang(n,m,beta), xlab="x", main="Random samples from Erlang(m, beta)",
     col="lightblue", density=30, xlim=c(0,20), ylim=c(0,0.2), freq=FALSE)
x <- seq(0,20,length=200)
lines(x, dgamma(x, shape=m, rate=beta), col="blue", type='l')
legend("topright", legend=c("Random samples","Distribution"),
       col=c("lightblue", "blue"), lty=1:1)
```



Exercise 5 - Middle square algorithm

```
init <- 3642467352
nsamples  <- 10000

rmidsquare <- function(n, init) {
    # retriveing number of digits of init
    ndigit <- length(unlist(strsplit(as.character(init),"")))
    max    <- 10^ndigit  #normalization constant

    k <- init
    z <- NULL
    for (i in 1:n) {
        sq <- k^2
        sq.string <- unlist(strsplit(as.character(sq),""))
        ll <- length(sq.string)
        sliced <- sq.string[floor((ll-ndigit)/2 +1):floor((ll+ndigit)/2)]

        k <- as.numeric(paste(sliced, collapse=""))
        z <- c(z, k/max)
    }
    return (z)
}

hist(rmidsquare(nsamples, init), col='green', density=30,
     main="Random generated samples", xlab="x", ylim=c(0,nsamples/20+100))
```