TP de Probabilités

NASSREDDINE, HUA, CROWLEY

12/05/2020

XXX Short description of what this is XXX

#Partie 1.

##Question 1

explanation explanation

explanation explanation *en italique* ou **en gras**.

- Item 1
- Item 2
  - sub-item 1
  - sub-item 2 Et des tableaux :

Permière colonne	Deuxième colonne
a	x
b	y

Ou mettre des formules : $Aire = \pi\, r^2$.

```
STANDARD_MINI <- function(k,graine)
{
  suite = matrix(nrow=k, ncol=1)
  suite[1]=graine
  for (i in 2:k)
  {
    suite[i]=(16807*suite[i-1])%%((2^31)-1)
  }
  return(suite)
}
```

##Question 2.1

explanation explanation

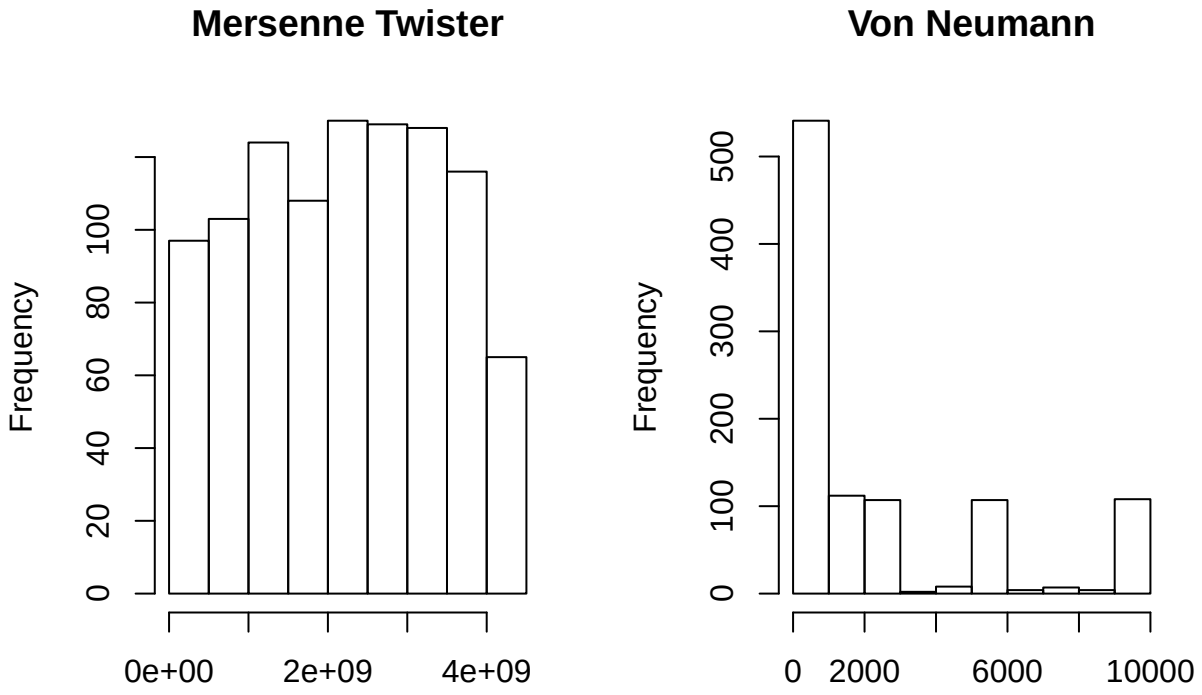
```
library(randtoolbox)
```

```
## Loading required package: rngWELL
```

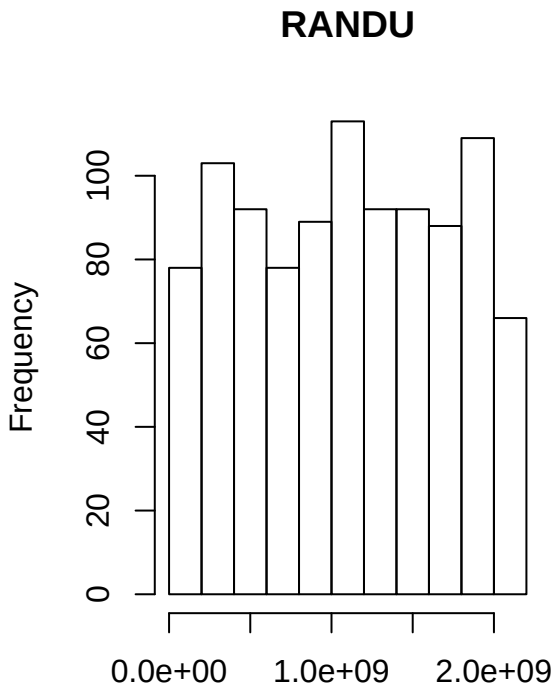
```
## This is randtoolbox. For an overview, type 'help("randtoolbox")'.
```

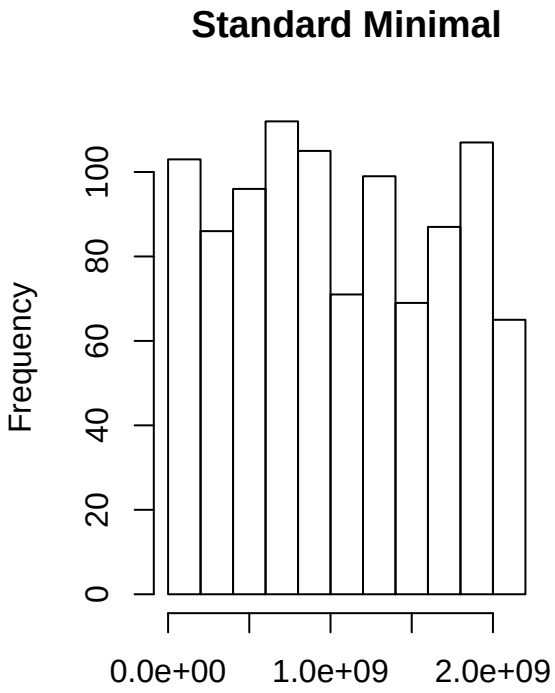
```
source('generateurs.R')

sVN <- 9721
```

```
sMT <- 2504
Nsimu <- 1000
Nrepet <- 1
grain <- 999

vn <- VonNeumann(Nsimu,Nrepet,sVN)
mt <- MersenneTwister(Nsimu,Nrepet,sMT)
ru <- RANDU(Nsimu, grain)
sm <- STANDARD_MINI(Nsimu, grain)

par(mfrow=c(1,2))
hist(mt[,1],xlab='',main='Mersenne Twister')
hist(vn[,1],xlab='',main='Von Neumann')
```




```
hist(ru[,1],xlab='',main='RANDU')
hist(sm[,1],xlab='',main='Standard Minimal')
```

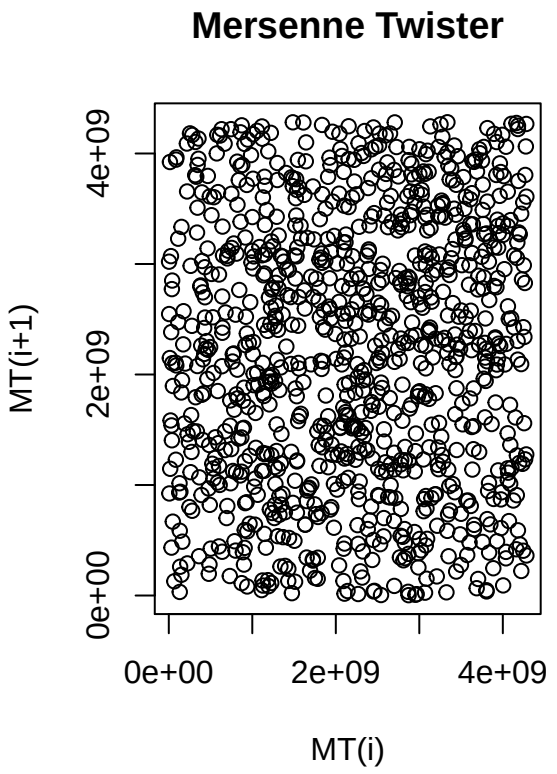


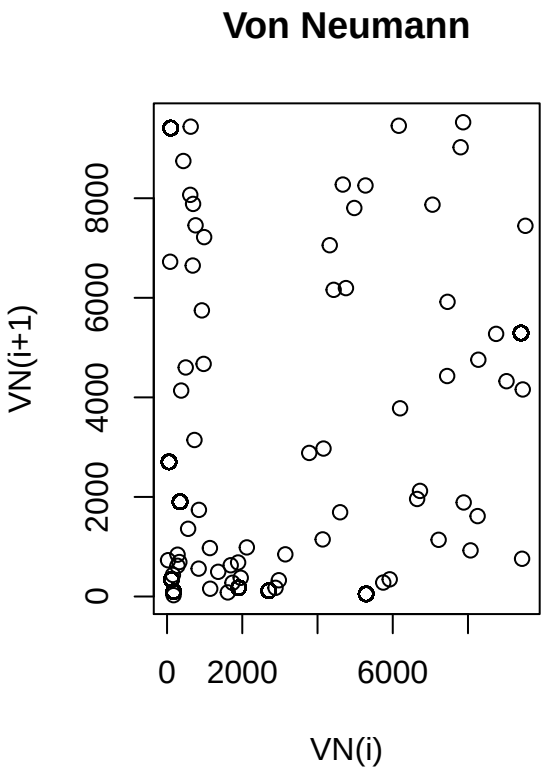


##Question 2.2

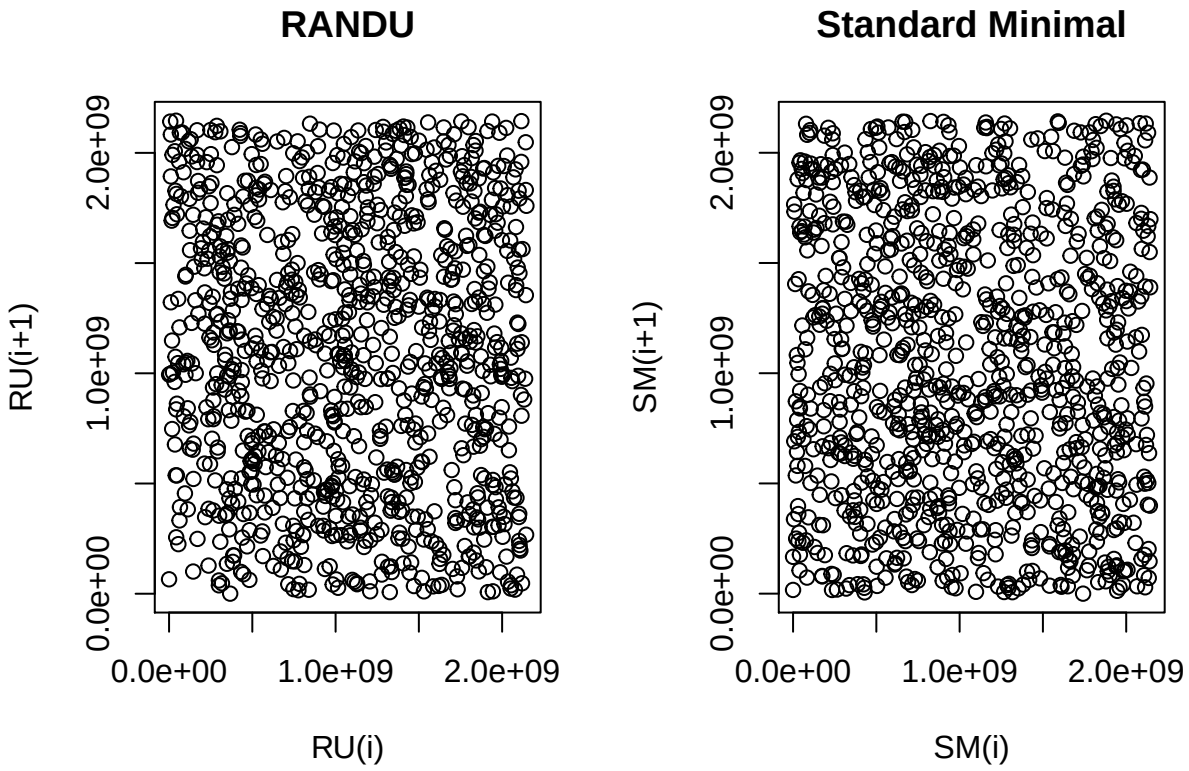
explanation explanation

```
par(mfrow=c(1,2))
plot(mt[1:(Nsimu-1),1],mt[2:Nsimu,1],xlab='MT(i)', ylab='MT(i+1)', main='Mersenne Twister')
plot(vn[1:(Nsimu-1),1],vn[2:Nsimu,1],xlab='VN(i)', ylab='VN(i+1)', main='Von Neumann')
```





```
plot(ru[1:(Nsimu-1),1],ru[2:Nsimu,1],xlab='RU(i)', ylab='RU(i+1)', main='RANDU')
plot(sm[1:(Nsimu-1),1],sm[2:Nsimu,1],xlab='SM(i)', ylab='SM(i+1)', main='Standard Minimal')
```



##Question 3

explanation explanation

```
Frequency <- function(x,nb)
{
  s<-0
  for(i in 1:length(x))
  {
    bin=binary(x[i])
    for(j in 0:(nb-1))
    {
      s <- s + (2*bin[32-j]-1)
    }
  }
  Sobs <- abs(s)/sqrt(nb*length(x))
  Pvaleur <- 2*(1-pnorm(Sobs))
  return(Pvaleur)
}
```

##Question 4

explanation explanation

```
Runs <- function(x,nb)
{
  #Obtention de la séquence concaténée
  V = binary(x[1])
  V = V[(32-nb+1):32]
```

```
  for(i in 2:length(x)) {
    bin = binary(x[i])
    V = c(V,bin[(32-nb+1):32])
  }
  n <- length(V)
  #pre-test
  pi <- sum(V)/n
  tau <- 2/sqrt(n)
  Pvaleur <- 0
  if(abs(pi-0.5)<tau)
  {
    Vnobs <- 1
    for(j in 1:(n-1))
    {
      if(V[j]!=V[j+1]){
        Vnobs <- Vnobs + 1
      }
    }
    Pvaleur <- 2*(1-pnorm(abs(Vnobs-2*n*pi*(1-pi))/(2*sqrt(n)*pi*(1-pi))))
  }
  return(Pvaleur)
}
```

##Question 5

explanation explanation

```
samples = sample.int(100000,100)

PVMT_O = matrix(nrow=length(samples), ncol=1)
PVRandU_O = matrix(nrow=length(samples), ncol=1)
PVSM_O = matrix(nrow=length(samples), ncol=1)
PVVnM_O = matrix(nrow=length(samples), ncol=1)

for(i in 1:length(samples))
{
  vn <- VonNeumann(Nsimu,Nrepet,samples[i])
  mt <- MersenneTwister(Nsimu,Nrepet,samples[i])
  randu = RANDU(Nsimu, samples[i])
  sm = STANDARD_MINI(Nsimu, samples[i])

  PVMT_O [i] = randtoolbox::order.test(mt[,1], d=4, echo=FALSE)$p.value
  PVRandU_O [i] = randtoolbox::order.test(randu[,1], d=4, echo=FALSE)$p.value
  PVSM_O [i] = randtoolbox::order.test(sm[,1], d=4, echo=FALSE)$p.value
  PVVnM_O [i] = randtoolbox::order.test(vn[,1], d=4, echo=FALSE)$p.value
}
```

#Partie 2.

##Question 1

explanation explanation

```
LoiBinomiale <- function (n,p)
{
  U = runif(n,min = 0,max = 1)
```

```
  X = sum(U < p)
  return (X)
}
```

##Question 2

explanation explanation

#Partie 3.

##Question 6

explanation explanation

```
FileMM1 <- function (lambda, mu, D)
{
  arrive <- c()
  depart <- c()

  time <- 0
  k <- 0
  while(time < D)
  {
    time <- time + rexp(1, rate=lambda)
    if (time < D)
    {
      arrive <- c(arrive, time)
      k <- k+1;
    }
  }

  print(arrive)

  if(length(arrive)>0)
  {
    depart [1] = arrive [1] + rexp(1, rate=mu)
    for (i in 2:length(arrive))
    {
      if (arrive [i] > depart[i-1])
      {
        time <- arrive [i] + rexp(1, rate=mu)
      }
      else
      {
        time <- depart [i-1] + rexp(1, rate=mu)
      }
      if (time < D)
      {
        depart <- c(depart, time)
      }
      else
      {
        break
      }
    }
  }
```

```
  queue <- list(arrive, depart)
  return(queue)
}
```

##Question 7

explanation explanation

```
MM1Evolution <- function(queue)
{
  arrive <- queue[[1]]
  depart <- queue[[2]]
  nbarrive <- length(arrive)
  nbdepart <- length(depart)
  time <- matrix(nrow=nbarrive + nbdepart +1, ncol=1)
  attendees <- matrix(nrow=nbarrive + nbdepart +1, ncol=1)
  time[1] = 0
  attendees[1]= 0

  i <- 1
  j <- 1
  while ( i <= nbarrive && j <= nbdepart)
  {
    if (arrive[i] <= depart[j])
    {
      attendees[i+j] = attendees [i+j-1] + 1
      time [i+j] = arrive [i]
      i <- i+1
    }
    else
    {
      attendees[i+j] = attendees [i+j-1] - 1
      time [i+j] = depart [j]
      j <- j+1
    }
  }
  if (i <= nbarrive)
  {
    for (k in i:nbarrive)
    {
      attendees[k+j] = attendees[k+j-1] + 1;
      time [k+j] = arrive [k]
    }
  }
  else if (j <= nbdepart)
  {
    for (k in j:nbdepart)
    {
      attendees[k+i] = attendees[k+i-1] - 1;
      time [k+i] = depart [k]
    }
  }
  results <- list(time, attendees)
  return(results)
}
```

##Question 8

explanation explanation