

# Projet R

De BRUGERE Arnaud ABDELLAH Safa AMROUCHE Henes

12/6/2020

Ce DM porte sur le projet numero 2, dans lequel il était question de créer une règle qui genererait des triangles à partir d'une generation aléatoire de 0 et de 1 au depart. Ensuite on etudierait cette regle et le nombre de 0 et de 1 au depart et trouverait une relation avec le nombre et la densité des triangles au final.

Notre premier but est donc ici de créer une matrice avec une generation de 0 et de 1 initiale (pour éviter de surcharger le Rmarkdown la matrice sera en 10x10)

(A préciser que toutes ces fonctions seront regroupées au sein d'une seule fonction nommé OneSimu() et que ce Rmarkdown sert à expliquer les composant de OneSimu())

```
largeur<-100
hauteur<-100
alea<-sample(c(0,1),size = largeur,replace=TRUE)
co<-matrix(0,hauteur,largeur)
co[1,]<-alea
```

Par la meme occasion on y insère la fonction rotate() permettant d'afficher correctement la matrice (en la tournant)

```
rotate <- function(x) t(apply(x, 2, rev))

image(rotate(co),
      main="Génération n°1",
      sub="Phase initiale",
      axes=FALSE,
      col.main="red", col.sub="black")
```

## Génération n°1



### Phase initiale

On cherche ensuite la loi que l'on utilisera pour former des triangles.

La loi utilisée sera:

|                 |       |       |       |       |       |       |       |       |
|-----------------|-------|-------|-------|-------|-------|-------|-------|-------|
| Cette lecture : | 1 1 1 | 1 1 0 | 0 1 1 | 1 0 1 | 0 0 1 | 1 0 0 | 0 1 0 | 0 0 0 |
| Génèrera :      | 0     | 0     | 0     | 0     | 1     | 1     | 0     | 0     |

On a donc créé une fonction appliquant la règle à la matrice

```
#génération de l'image#
generate <- function(hauteur,largeur,mat){
h<-1
l<-1
while(h<hauteur){
  while(l<largeur){
    if(paste(sapply(list(mat[h,l],mat[h,l+1]),function(x) x),collapse='')=="01"){ #bord gauche
      mat[h+1,l]<-1
    }else{
      mat[h+1,l]<-0
    }
  }
  l<-l+1
  while(l<largeur){
    if(paste(sapply(list(mat[h,l-1],mat[h,l],mat[h,l+1]),function(x) x),collapse='')=="001" || paste(
```

```

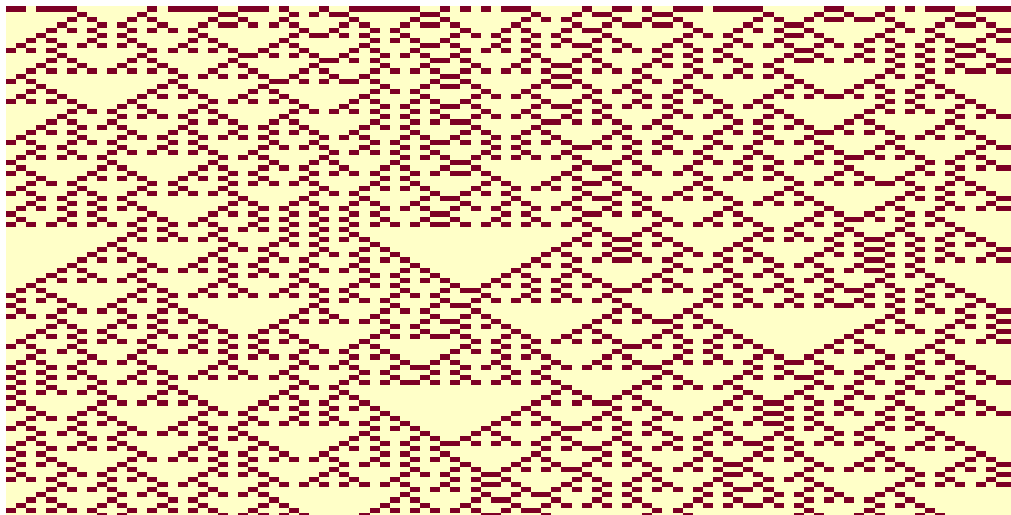
}
if(paste(sapply(list(mat[h,l-1],mat[h,l]),function(x) x),collapse='')=="10"){ #bord droit
  mat[h,l]<-1
}else{
  mat[h,l]<-0
}
l<-l+1
}
l<-1
h<-h+1
}
image(rotate(mat),
      main="Génération n°1",
      sub="Phase 1",
      axes=FALSE,
      col.main="red", col.sub="black")
return(mat)
}

```

On affiche donc le resultat (avec les dimensions de la matrice en 100x100)

```
co<- generate(hauteur,largeur,co)
```

## Génération n°1



## Phase 1

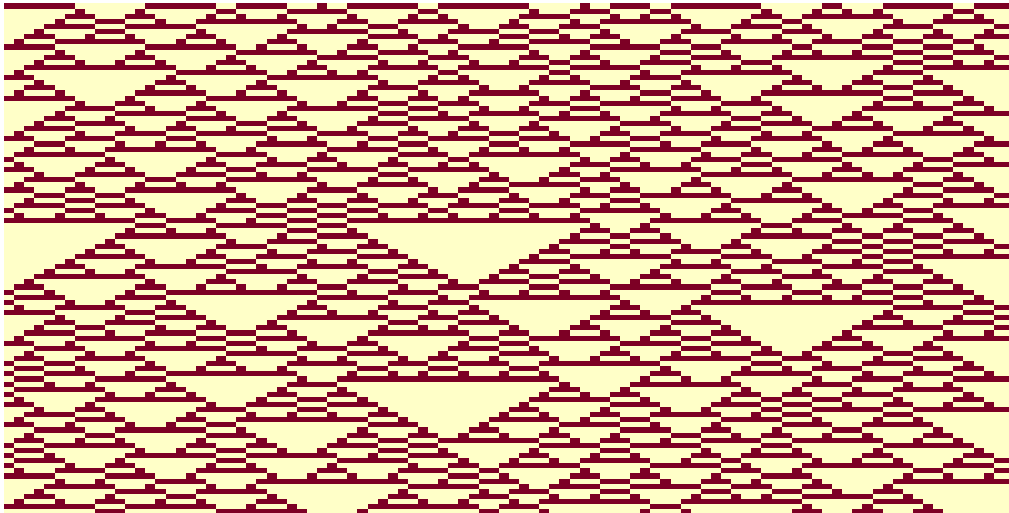
On cherche ensuite à calculer la densité des triangles trouvés et de les compter.

pour calculer la densité, on va changer les 0 qui n'appartiennent pas à des triangles en 1 via deux fonctions,

la première vise à remplacer les 101 en 111 (turn101to111())

```
turn101to111 <- function(mat){  
  h<-1  
  l<-1  
  while(h<=hauteur){  
    while(l+1<largeur){  
      if(paste(sapply(list(mat[h,l],mat[h,l+1],mat[h,l+2]),function(x) x),collapse=',')=="101"){  
        mat[h,l+1]<-1  
      }  
      l<-l+1  
    }  
    l<-1  
    h<-h+1  
  }  
  image(rotate(mat),  
        main="Génération n°1",  
        sub="Phase 2",  
        axes=FALSE,  
        col.main="red", col.sub="black")  
  return(mat)  
}  
  
co<-turn101to111(co)
```

## Génération n°1



## Phase 2

On cherche ensuite à supprimer ce qui ne ressemble pas à des triangles (les blocs) via la fonction SupprBloc()

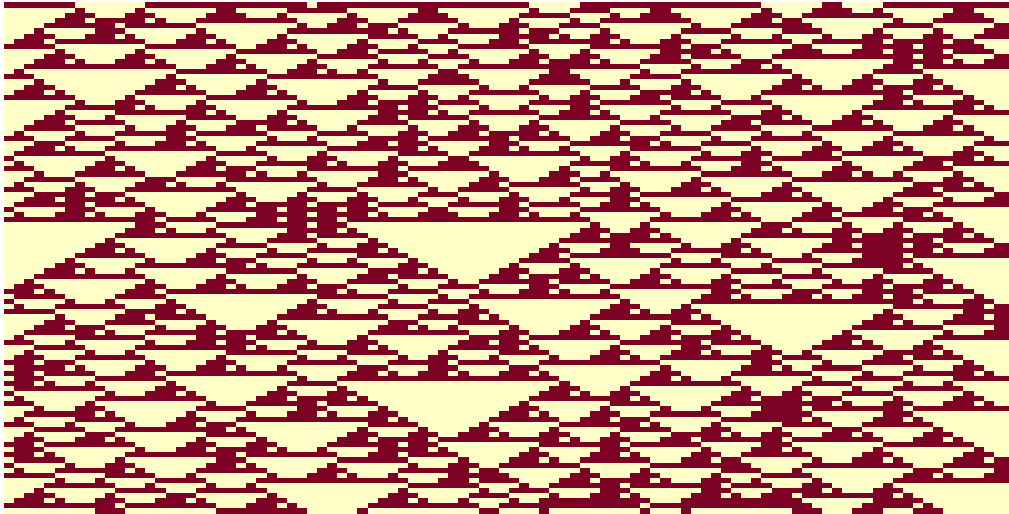
```

supprBloc <- function(mat){
#Premiere ligne
l<-1
h<-1
while(l+1<largeur){
  if(paste(sapply(list(mat[h,l],mat[h,l+1],mat[h,l+2],mat[h+1,l+1],mat[h+1,l+2]),function(x) x),collapse="")){
    mat[h,l+1]<-1
    mat[h,l+2]<-1
  }
  l<-l+1
}
#Milieu
l<-1
h<-h+1
while(h+1<hauteur){
  while(l+1<largeur){
    if(paste(sapply(list(mat[h-1,l+1],mat[h-1,l+2],mat[h,l],mat[h,l+1],mat[h,l+2],mat[h+1,l+1],mat[h+1,l+2]),function(x) x),collapse="")){
      mat[h,l+1]<-1
      mat[h,l+2]<-1
    }
    l<-l+1
  }
  l<-1
  h<-h+1
}
#Derniere ligne
while(l+1<largeur){
  if(paste(sapply(list(mat[h-1,l+1],mat[h-1,l+2],mat[h,l],mat[h,l+1],mat[h,l+2]),function(x) x),collapse="")){
    mat[h,l+1]<-1
    mat[h,l+2]<-1
  }
  l<-l+1
}
image(rotate(mat),
      main="Génération n°1",
      sub="Phase 3",
      axes=FALSE,
      col.main="red", col.sub="black")
return(mat)
}

co<-supprBloc(co)

```

## Génération n°1

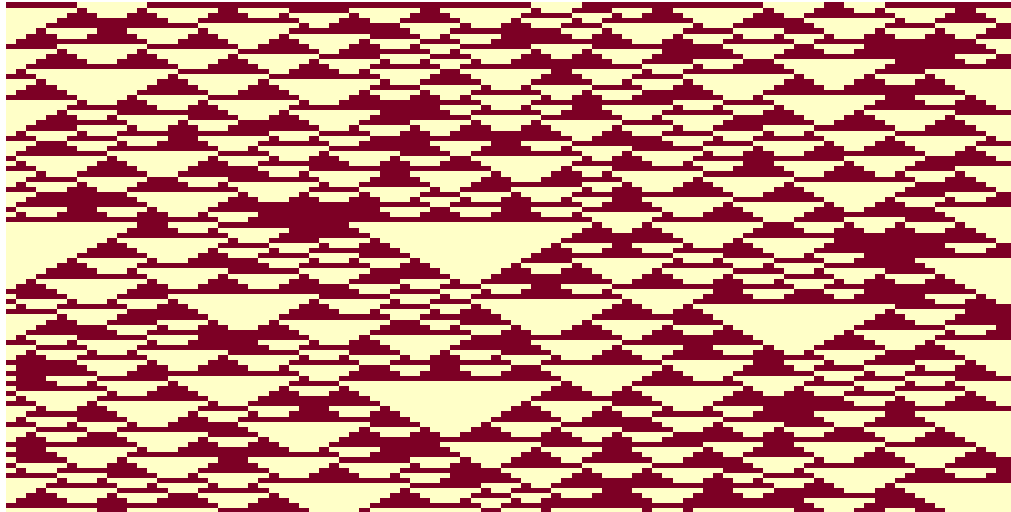


## Phase 3

On observe ensuite qu'il y a des nouvelles sections 101 qui ont été créées, alors on re-exécute `turn101to111()`

```
co<-turn101to111(co)
```

## Génération n°1



### Phase 2

On observe ensuite des 0 seuls sur les côtés donc on utilise une nouvelle fonction (detail()) pour les supprimer

```
detail <- function(mat){#
l<-1
boucle<-0
while(boucle<2){
  h<-1
  if(paste(sapply(list(mat[h,l],mat[h+1,l]),function(x) x),collapse='')=="01"){
    mat[h,l]<-1
  }
  while(h+2<hauteur){
    if(paste(sapply(list(mat[h,l],mat[h+1,l],mat[h+2,l]),function(x) x),collapse='')=="101"){
      mat[h+1,l]<-1
    }
    h<-h+1
  }
  if(paste(sapply(list(mat[h+1,l],mat[h+2,l]),function(x) x),collapse='')=="10"){
    mat[h+2,l]<-1
  }
  l<-largeur
  boucle<-boucle+1
}
image(rotate(mat),
      main="Génération n°1",
      sub="Phase 4",
      axes=FALSE,
```

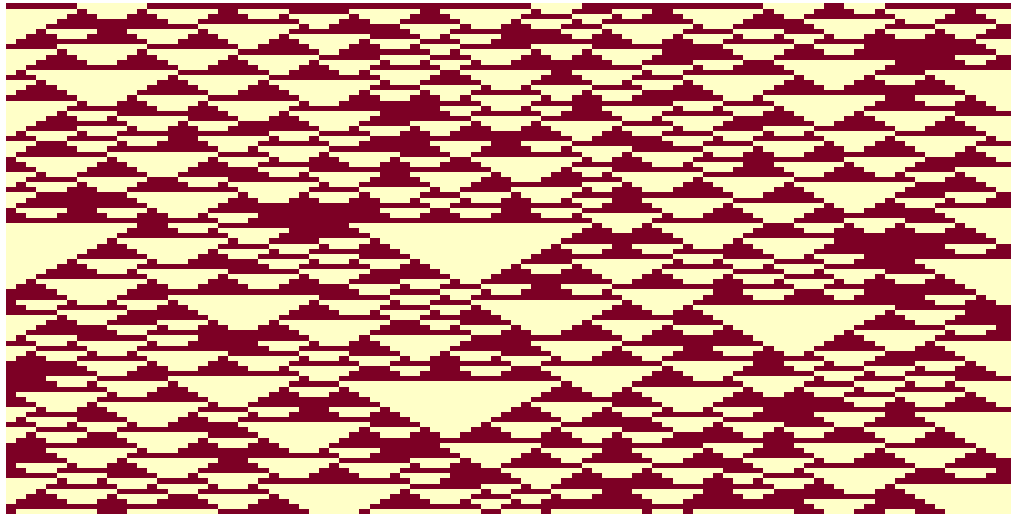
```

        col.main="red", col.sub="black")
return(mat)
}

co<- detail(co)

```

## Génération n°1



## Phase 4

On obtient donc une matrice de generation de triangles avec des 0 formant uniquement des triangles et des 1 formant le reste

Le défi maintenant va être de recevoir les données sous forme de tableau afin de pouvoir les traiter et essayer de trouver une relation entre la première ligne générée et le nombre et la densité des triangles au final

Pour cela, on va créer une table() sans valeur à l'intérieur

Les principales entêtes de ce tableau sont:

Gen0 : Nombre de 0 généré initialement Gen1 : Nombre de 1 généré initialement End0 : Nombre de 0 sur la figure au final End1 : Nombre de 1 sur la figure au final GenRatio : Gen0 / Gen1 EndRatio : End0 / End1 nbTriangle : nombre de triangles sur la figure

```
FinalTab<-head(data.frame(Gen0=NA,Gen1=NA,End0=NA,End1=NA,GenRatio=NA,EndRatio=NA,nbTriangle=NA),0)
```

```
FinalTab
```

```
## [1] Gen0      Gen1      End0      End1      GenRatio  EndRatio  nbTriangle
## <0 rows> (or 0-length row.names)
```



On cherche désormais à remplir ce tableau avec les données trouvées sur notre génération

Pour cela il faut des fonctions capables de récolter les données

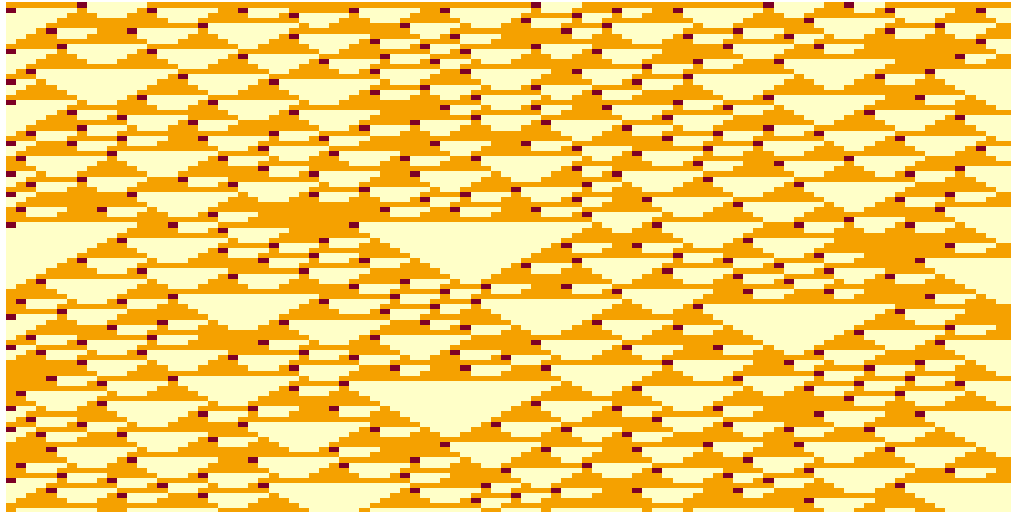
La première est : nbTriangle()

elle va compter le nombre de triangle d'une génération et le renvoyer

```
nbTriangle <- function(hauteur, largeur, mat){
  compt<-0
  h<-1
  l<-1
  #1ere ligne
  while(l<largeur-1){
    if(paste(sapply(list(mat[h,l],mat[h,l+1]),function(x) x),collapse='')=="10"){
      compt<-compt+1
      mat[h,l+1]<-2
    }
    l<-l+1
  }
  #milieu
  while(h<hauteur-1){
    l<-2
    while(l<largeur){
      if(paste(sapply(list(mat[h,l-1],mat[h-1,l],mat[h,l]),function(x) x),collapse='')=="110"){
        compt<-compt+1
        mat[h,l]<-2
      }
      l<-l+1
    }
    h<-h+1
  }
  #coté droit et gauche
  l<-1
  h<-1
  while(h+1<hauteur){
    if(paste(sapply(list(mat[h,l],mat[h+1,l]),function(x) x),collapse='')=="10"){
      compt<-compt+1
      mat[h+1,l]<-2
    }
    h<-h+1
  }
  image(rotate(mat),
        main="Génération n°1",
        sub="Marquage de triangles",
        col.main="red", col.sub="black",
        axes=FALSE
        )
  return(compt)
}

nbTriangle(hauteur, largeur, co)
```

## Génération n°1



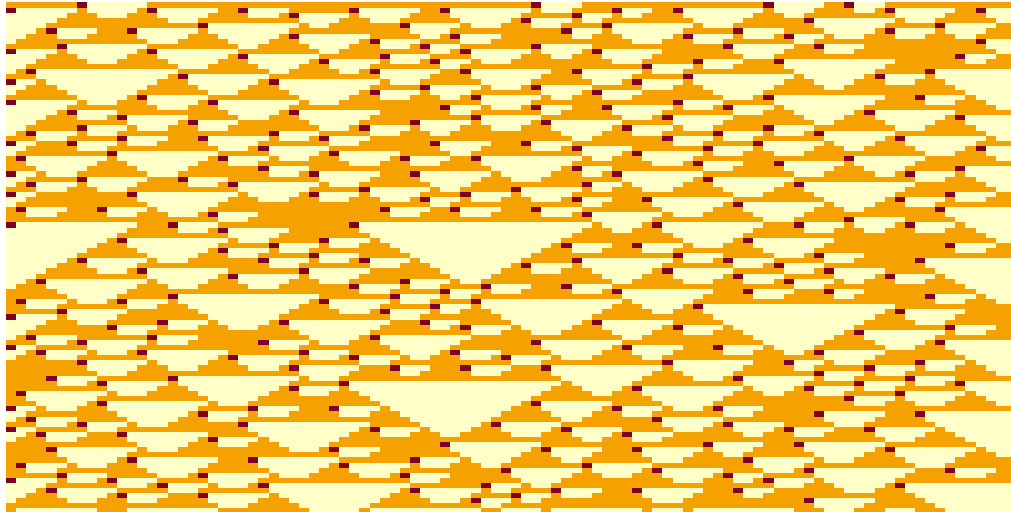
## Marquage de triangles

```
## [1] 325
```

La deuxième est : `counter()` elle permet de récupérer les données de `Gen0`, `Gen1`, `End0`, `End1`, `GenRatio` et `EndRatio` et de renvoyer l'ensemble (avec `nbTriangle` en plus) sous forme de `data.frame`

```
counter <- function(alea,mat,hauteur,largeur){  
  Alea0<-length(alea[alea==0])  
  Alea1<-length(alea[alea==1])  
  Mat0<-length(mat[mat==0])  
  Mat1<-length(mat[mat==1])  
  return(data.frame(Gen0=Alea0,Gen1=Alea1,End0=Mat0,End1=Mat1,GenRatio=Alea0/Alea1,EndRatio=Mat0/Mat1,nbTriangle=hauteur*largeur))  
}  
  
FinalTab<-rbind(FinalTab,counter(alea,co,hauteur,largeur))
```

## Génération n°1



## Marquage de triangles

FinalTab

```
##   Gen0 Gen1 End0 End1  GenRatio EndRatio nbTriangle
## 1    48   52 5340 4660 0.9230769 1.145923         325
```

Desormais on peut commencer à créer plusieurs generations et etudier leurs données, on va par conséquent faire evoluer le nombre de 0 dans la génération initiale (de 1 à 100) et observer l'évolution de la courbe en fonction du nombre de triangle pour cela on va avoir besoin d'une fonction qui va faire changer le nombre de 0 en fonction du nombre d'iteration soit : nb0()

```
nb0 <- function(nb){
  alea<-rep(1,100)
  where0<-sample(1:100,size=nb)
  i<-1
  while(i<=nb){
    alea[where0[i]]<-0
    i<-i+1
  }
  return(alea)
}
```

Ainsi, ici on aura une generation avec le nombre de 0 equivalent a la variable nb

Légère remarque à propos de OneSimu(), on a fait deux modes de simulation au sein de cette fonction: -un mode dans lequel on ne generait qu'une seule fois la figure, vu ci dessus(en initialisant evolve<-FALSE) -un

mode à 100 repetitions avec variations croissante du nombre de 0 dans la génération initiale, vu ci dessous (en initialisant `evolve<-TRUE`)

On lance une boucle à 100 itérations avec 1 à 100 zéros dans la valeur initiale, on obtient ce tableau :

(Note: on a chargé un tableau de données afin d'éviter de faire charger le fichier trop longtemps)

```
#Avec evolve<-TRUE  
  
load(file="triangle")  
  
head(FinalTab)
```

| ##   | Gen0 | Gen1 | End0 | End1 | GenRatio   | EndRatio | nbTriangle |
|------|------|------|------|------|------------|----------|------------|
| ## 1 | 0    | 100  | 9901 | 99   | 0.00000000 | 100.0101 | 1          |
| ## 2 | 1    | 99   | 9901 | 99   | 0.01010101 | 100.0101 | 1          |
| ## 3 | 2    | 98   | 9901 | 99   | 0.02040816 | 100.0101 | 1          |
| ## 4 | 3    | 97   | 9901 | 99   | 0.03092784 | 100.0101 | 1          |
| ## 5 | 4    | 96   | 9901 | 99   | 0.04166667 | 100.0101 | 1          |
| ## 6 | 5    | 95   | 9901 | 99   | 0.05263158 | 100.0101 | 1          |

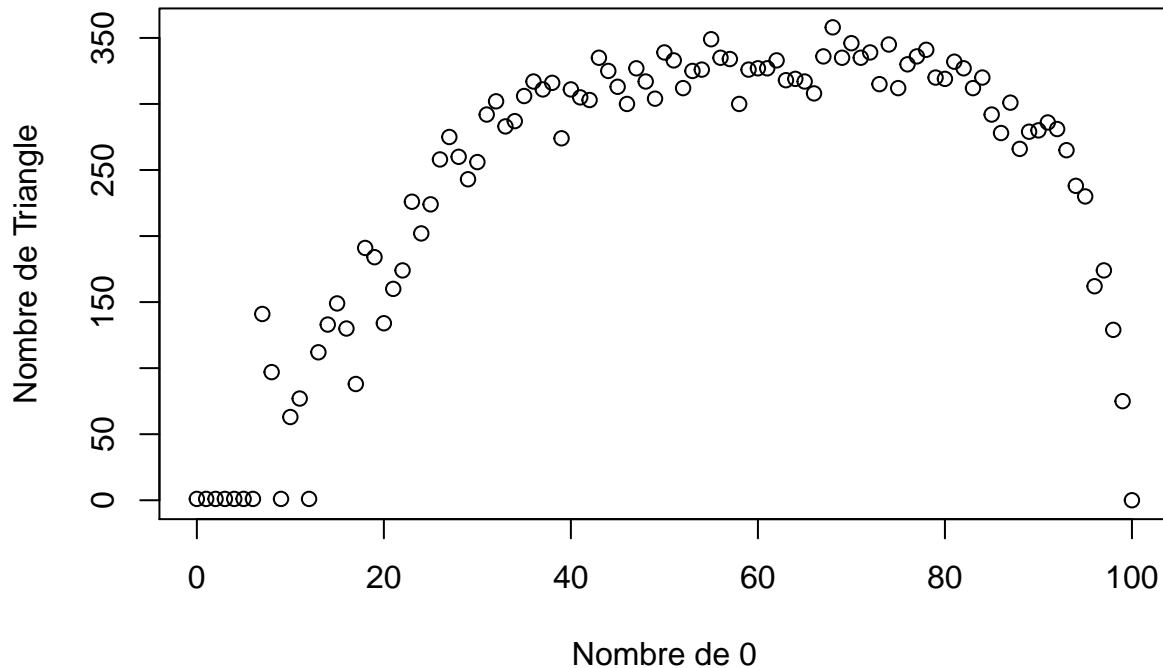
Ensuite pour terminer, on va créer un graphique avec le nombre de triangle en fonction du nombre de 0 initialement

(Note: que l'on relie le fait que plus il y a de triangles et plus leur densité est basse, moins il y en a et plus leur densité est haute)

on obtient

```
plot(x=FinalTab$Gen0,  
     y=FinalTab$nbTriangle,  
     main="Evolution du nombre de triangle en fonction du nombre de 0 initialement",  
     xlab="Nombre de 0",  
     ylab="Nombre de Triangle",  
     col.main="red")
```

## Evolution du nombre de triangle en fonction du nombre de 0 initialem



On cherche par conséquent une relation et on observe que le nombre de triangles est plus élevé approximativement à partir de 35-40 zéros dans la génération initiale.

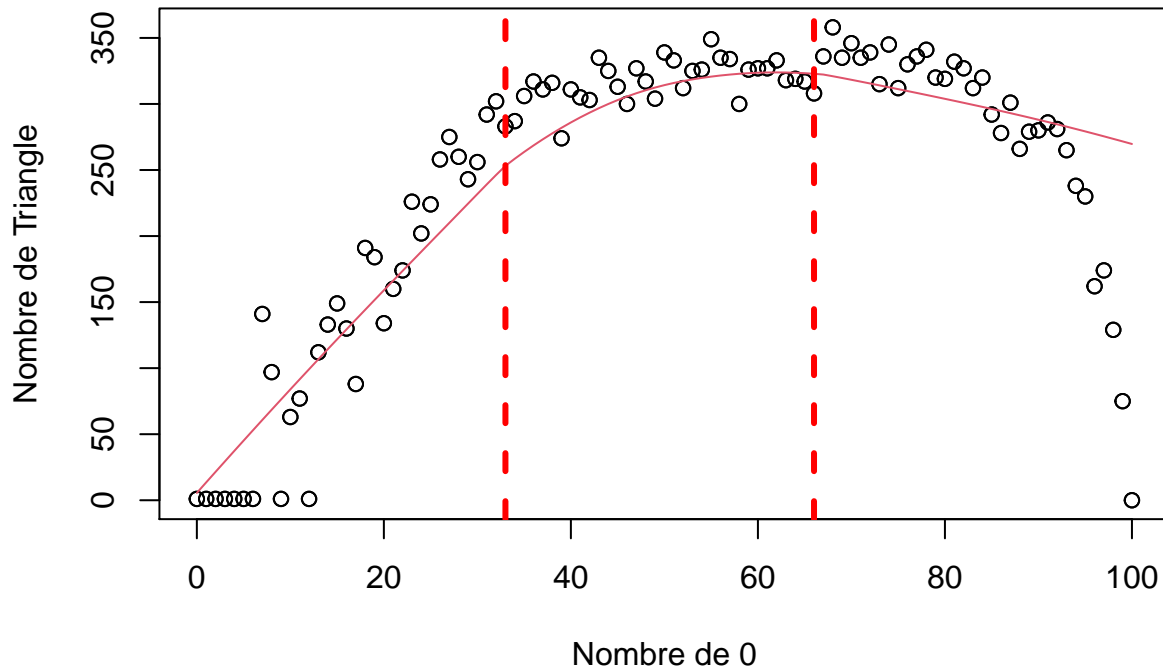
On a donc réfléchi et on a observé que ce qui code des 1 était 001 et 100, soit  $\frac{2}{3}$  de 0 et  $\frac{1}{3}$  de 1

|                     |       |       |       |       |       |       |       |       |
|---------------------|-------|-------|-------|-------|-------|-------|-------|-------|
| Cette lecture :     | 1 1 1 | 1 1 0 | 0 1 1 | 1 0 1 | 0 0 1 | 1 0 0 | 0 1 0 | 0 0 0 |
| (Rappel) Générera : | 0     | 0     | 0     | 0     | 1     | 1     | 0     | 0     |

On a essayé d'observer cela en traçant une courbe et en délimitant le nuage de points en 3 tier comme cela :

```
plot(x=FinalTab$Gen0,
     y=FinalTab$nbTriangle,
     main="Evolution du nombre de triangle en fonction du nombre de 0 initialement",
     xlab="Nombre de 0",
     ylab="Nombre de Triangle",
     col.main="red")
panel.smooth(FinalTab$Gen0, FinalTab$nbTriangle)
abline(v=33,col="red", lty=2, lwd=3)
abline(v=66,col="red", lty=2, lwd=3)
```

## Evolution du nombre de triangle en fonction du nombre de 0 initialem



On observe ainsi que l'instant avec le nombre le plus haut de triangles se fait lorsqu'on a une génération à approximativement  $2/3$  de zéros et  $1/3$  de uns.

En effet on peut dissocier ce graphique en trois principales parties: Une fonction affine décrivant une croissance du nombre de triangles générés en fonction du nombre de 0 (0-40) Un plateau/ Une hyperbole avec le maximum de triangles pouvant être générés qui correspond au  $2/3$  du nombre de 0 total (environ 66-67) Une fonction affine décrivant une décroissance du nombre de triangles générés en fonction du nombre de 0 (68-100) Ainsi nous pouvons donc nous apercevoir que ce plateau, ce seuil représenté par environ 67 zéros, indique le maximum de zéros qui génèrent la densité maximum d'un triangle En effet tous les triangles sont composés au maximum de 67 zéros, une fois que ce nombre est atteint alors on construit des triangles avec un nombre de zéros inférieurs. On peut donc de manière plus large, faire le parallèle pour expliquer de manière plus scientifique certains phénomènes concernant des modèles animaux, le coquillage. En effet de prime à bord nous pourrions penser qu'il s'agit là d'un revêtement de triangles causé par un nombre aléatoire de motifs sur ce coquillage. Mais grâce à cette loi on remarque que chaque triangle suit finalement une probabilité qui tient en compte du «voisins» et plus précisément de sa densité. C'est-à-dire que chaque triangle généré est finalement construit par rapport à ses deux voisins directs (celui de droite et celui de gauche). Ainsi encore une fois les mathématiques et diverses applications informatiques ont été au service de la science afin d'expliquer certains phénomènes de la vie...