SIMULATIONS, ALGORITHMES EN PROBABILITÉ
APPLICATIONS AVEC **R**

Hubert RAYMONDAUD LEGTA de Carpentras-Serres (84) – Ministère de l'Agriculture

Table des matières

SIMULATIONS, ALGORITHMES EN PROBABILITÉ
APPLICATIONS AVEC **R**

Hubert RAYMONDAUD LEGTA de Carpentras-Serres (84) – Ministère de l'Agriculture

I - INTRODUCTION

Lorsque l'on m'a demandé de rejoindre le groupe d'experts pour proposer des exemples de mise en œuvre de l'algorithmique en probabilité et statistiques, j'ai fait un peu de bibliographie dans les ouvrages de lycée.

J'y ai constaté, sans ordre particulier :

- ♦ Des exemples peu variés, surtout des illustrations de cours, la modélisation n'est pas clairement abordée,
- ♦ Une grande hétérogénéité et la faible précision du “langage naturel” dont sont faits les algorithmes,
- ♦ Pas de notification de l'existence de stratégies alternatives pour tenir compte de la spécificité des langages dans lesquels sont traduits les algorithmes,
- ♦ Aucune prise en compte de l'aspect distribution (simulée) des variables étudiées, pas d'illustration graphique,
- ♦ Un outil simulation très pauvre et donc peu attractif pour les enseignants et les élèves.

Je vais donc tenter de montrer comment, avec **R**, on peut

- ● Faire de la simulation un enjeu didactique pour l'enseignement des probabilités et des statistiques, voire de la statistique (dans le supérieur court et long).
- ● Faire de la simulation un outil de “résolution*” de problèmes, “résolution*”" signifiant la recherche d'une solution asymptotiquement exacte,
- ● Mettre simplement en œuvre les algorithmes pour créer les programmes de simulation, avec un seul outil,
- ● Toucher du doigt les limites du "langage naturel" pour élaborer des stratégies de résolution, mettant ainsi en évidence la nécessité d'un réflexion élargie et approfondie sur ce sujet.

Le programme théorique de cette matinée :

- ► Voir quelques enjeux didactiques de la simulation,
- ► Pourquoi et comment faire de la simulation un véritable outil de résolution de problèmes,
- ► Comment exploiter et prolonger les résultats d'un expérience concrète ou d'un enquête,

Tout ceci en passant en revue les principales fonctions internes de **R** nécessaire à l'acquisition d'une certaine autonomie en simulation probabiliste et analyse exploratoire de données.

II - MISE EN ŒUVRE DE LA SIMULATION AVEC R

A. LES ENJEUX DIDACTIQUES DE LA LA SIMULATION COMME SUPPORT DE COURS

A1. L'EXPÉRIENCE HISTORIQUE (1620) D'UN GRAND DUC DE TOSCANE :

► **PARADOXE des chances d'obtenir 9 et d'obtenir 10 comme somme des valeurs des faces lors du jet de 3 dés équilibrés.**

- L'utilisation d'un générateur de nombres pseudo-aléatoires de distribution uniformes entre 0 et 1 est souvent un obstacle pour les élèves, surtout lorsque l'objectif est de les rendre capable de la réinvestir dans la mise en œuvre de simulations, en autonomie.
- La fonction **`sample()`**, spécifique du langage **R**, permettent de s'affranchir de cette difficulté et rend possible la "reproduction", le "mime" de l'expérience réelle que l'on cherche à simuler : exemples de la création d'un dé et de la simulation d'un jeu consistant à lancer 3 fois un dé à 6 faces équiprobables et à faire la somme **(x)** des valeurs des 3 faces obtenues.

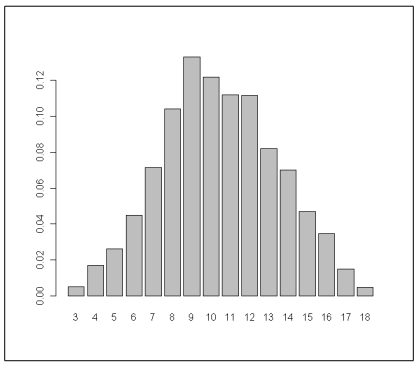
```
#*SIMULATION*PROBLÈME HISTORIQUE D'UN GRAND DUC DE TOSCANE : 3 DÉS
# ALGO A1_1 : LIGNES DE COMMANDES POUR SIMULER UN JEU
de <- seq(from = 1, to = 6, by = 1)
jeu <- sample(x = de, size = 3, replace = TRUE)
x <- sum(jeu)
```

```
> (de <- seq(from = 1, to = 6, by = 1))
[1] 1 2 3 4 5 6
> (jeu <- sample(x = de, size = 3, replace
   = TRUE))
[1] 3 4 3
> (x <- sum(jeu))
[1] 10
```

- On peut ainsi plus facilement repérer **le modèle** que l'on a inséré dans la simulation (une seule difficulté à la fois). Il est important de noter que **le modèle équiprobable** n'est pas introduit au niveau du dé, mais lors du "lancer" simulé par la fonction **`sample()`** qui effectue trois tirages équiprobables, avec remise, dans les six faces du dé.
- Les exemples d'algorithmes ou de programmes que j'ai rencontrés dans les manuels scolaires se limitent souvent à la simulation d'une seule valeur de la variable. Ils suggèrent ensuite de répéter manuellement cette simulation et d'utiliser un tableur pour saisir et illustrer graphiquement la série simulée. L'usage de plusieurs outils logiciels est un obstacle pratique et pédagogique.
- Comme je le montre ci-dessous, **R** possède une boite à outils complète et variée pour l'analyse exploratoire des séries simulées, allant de la création du tableau des effectifs à la production de graphiques variés, autant pour les variables qualitatives que quantitatives discrètes ou continues.

```
# ALGO A1_2 : LIGNES DE COMMANDES POUR SIMULER 2000 JEUX IDENTIQUES ET
# EN DÉCIRE LES RÉSULTATS
seriex <- NULL ; de <- seq(1, 6, 1)
for(i in 1:2000){
  jeu <- sample(x = de, size = 3, replace = TRUE)
  x <- sum(jeu)
  seriex <- c(seriex, x)
                }
tablfreqx <- table(seriex) / 2000
barplot(tablfreqx)

seriex
     3      4      5      6      7      8      9     10     11     12     13
0.0050 0.0115 0.0230 0.0470 0.0705 0.0970 0.1175 0.1255 0.1225 0.1230 0.0945
    14     15     16     17     18
0.0645 0.0450 0.0295 0.0165 0.0075
```

```
> barplot(tablfreqx)
```



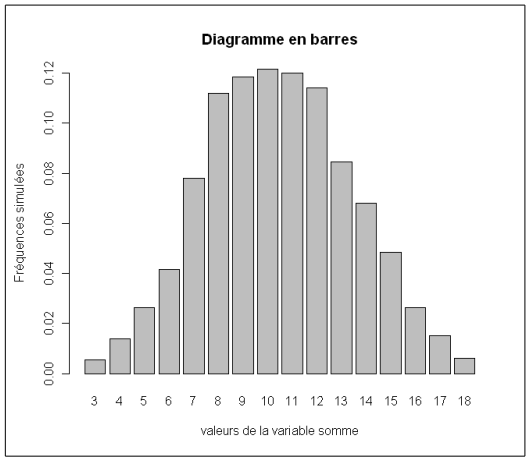
- Le symbole d'affectation préférentiels est **`<-`** mais on peut indifféremment utiliser **`=`**. **`c()`** est la fonction permettant de créer des listes indicées (**attention il n'y a pas d'indice 0**). Pour remplir une liste avec les valeurs successives d'une variable **x** on peut faire **`liste <- c(liste, x)`** ou bien **`liste[i] <- x`**.
- **R** est un langage "fonctionnel". Il n'y a donc pas d'instruction de "saisie", tout se fait par l'intermédiaire

des “paramètres” de la fonction créée, auxquels on peut attribuer des valeurs par défaut lors de la programmation et que l'on peut changer lors de l'exécution de la fonction.

- Pour les simulations il est plus pratique de pas avoir à saisir de valeur lors de l'exécution des fonctions.
- Les graphiques sont “paramétrables” à volonté.
- Les fonctions “internes” possèdent un nombre variable de “paramètres”. Elles sont toutes documentées, avec des références bibliographiques sur les outils statistiques mis en œuvre.
- Dans la fonction suivante les valeurs par défaut des deux paramètres sont **3** pour le nombre de jets **nbjets** et **2000** pour le nombre de répétition **nbsim** de l'expérience aléatoire. Si l'on veut simuler 10000 répétitions d'une expérience consistant à jeter 4 dés on exécutera les fonctions **`toscaneA1_3(4, 10000)`** ou bien **`toscaneA1_3(nbjets = 4, nbsim = 10000)`**.

```
# ALGO A1_3 : UNE FONCTION POUR FACILITER LE PARAMÉTRAGE
toscaneA1_3 <- function(nbjets = 3, nbsim = 2000){
  de <- seq(1, 6, 1) ; seriex <- NULL
  for(i in 1:nbsim){
    jeu <- sample(de, nbjets, replace = TRUE)
    x <- sum(jeu)
    seriex <- c(seriex, x)
                  }
  tablfreqx <- table(seriex) / nbsim
# Affichage des résultats et des graphiques
  cat("Tableau des fréquences simulés :\n")
  print(tablfreqx)
  barplot(tablfreqx, xlab = "valeurs de la variable somme",
    ylab = "Fréquences simulées", main = "Diagramme en barres")
}
```

```
> toscaneA1_3()
Tableau des fréquences simulés :
seriex
     3      4      5      6      7      8
0.0055 0.0140 0.0265 0.0415 0.0780 0.1120
     9     10     11     12     13     14
0.1185 0.1215 0.1200 0.1140 0.0845 0.0680
    15     16     17     18
0.0485 0.0265 0.0150 0.0060
```



- Deux stratégies alternatives pour simuler la même expérience.

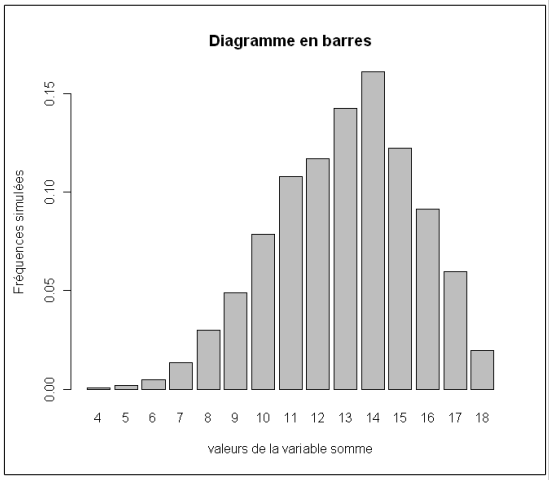
```
# ALGO A1_3bis : UNE AUTRE STRATÉGIE DE SIMULATION
# POUR ÉVITER LES BOUCLES ON EST FIXÉ À 3 JETS
toscaneA1_3bis <- function(nbsim = 2000){
  de <- seq(1, 6, 1)
  jets1 <- sample(de, nbsim, replace = TRUE)
  jets2 <- sample(de, nbsim, replace = TRUE)
  jets3 <- sample(de, nbsim, replace = TRUE)
  som3jets <- jets1 + jets2 + jets3
  tablfreqx <- table(som3jets) / nbsim
# Affichage des résultats et des graphiques
  print(tablfreqx)
  barplot(tablfreqx,
    xlab = "valeurs de la variable somme",
    ylab = "Fréquences simulées",
    main = "Diagramme en barres")
}
```

```
# ALGO A1_3ter : UNE AUTRE STRATÉGIE DE SIMULATION POUR
# ÉVITER LES BOUCLES ET FAIRE VARIER LE NOMBRE DE JETS
toscaneA1_3ter <- function(nbcoups = 3, nbsim = 2000){
  de <- seq(1, 6, 1)
  coups <- matrix(0, nrow = nbcoups + 1, ncol = nbsim)
  for(k in 1:nbcoups){
    coups[k, ] <- sample(de, nbsim, replace = TRUE)
                  }
    coups[nbcoups + 1, ] <- apply(coups, MARGIN = 2, sum)
  tablfreqx <- table(coups[nbcoups + 1, ]) / nbsim
# Affichage des résultats et des graphiques
  print(tablfreqx)
  barplot(tablfreqx, xlab = "valeurs de la variable somme",
    ylab = "Fréquences simulées", main = "Diagramme en
   barres")
}
```

- Selon la stratégie utilisée, on pourra ou non avoir le nombre de jets comme paramètre, pour pouvoir le faire varier sans changer de stratégie c'est à dire sans toucher au code de la fonction.
- À la différence de la stratégie initale dans laquelle les trois jets étaient effectués par la fonction **`sample()`**, elle effectue ici les 2000 répétition de chacun des trois jets. Celà permet de remplacer la boucle **`for()`** interprétée, par la boucle compilée de **`sample()`**.

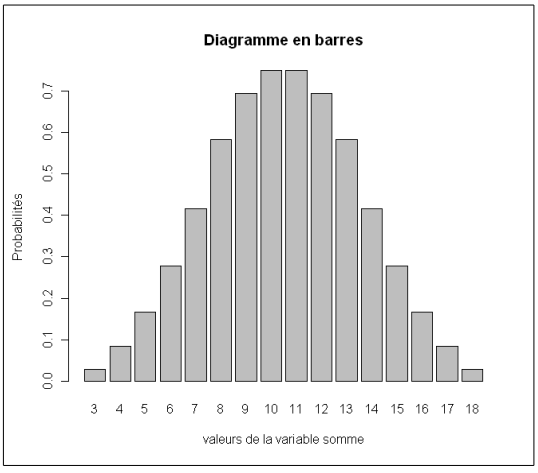
- Dans le cas d'un dé dont les faces ne sont plus équiprobables, c'est plus compliqué lorsque l'on n'a que le générateur pseudo-aléatoire de base. Cette simulation est rendue très simple avec les fonctions spécifiques de **R** : nous allons voir comment faire pour simuler un dé dont l'apparition d'un face est proportionnelle à la valeur de la face.
- Dans la fonction suivante, c'est le paramètre **`prob`** de la fonction **`sample()`** qui fixe la distribution de probabilité des valeurs des faces du dé.

```
# ALGO A1_4 FONCTION : ET SI LE DÉ N'EST PAS ÉQUILIBRÉ ??
# CAS D'ÉCOLE CLASSIQUE : PROBABILITÉ PROPORTIONNELLE AUX VALEURS
# DES FACES SOIT 1/21, 2/21, 3/21, 4/21, 5/21, 6/21
toscaneA1_4 <- function(nbjets = 3, nbsim = 2000,
                probafaces = c(1/21, 2/21, 3/21, 4/21, 5/21, 6/21)){
  de <- seq(1, 6, 1) ; seriex <- NULL
  for(i in 1:nbsim){
    jeu <- sample(de, nbjets, replace = TRUE, prob = probafaces)
    x <- sum(jeu)
    seriex <- c(seriex, x)
                    }
  tablfreqx <- table(seriex) / nbsim
# Affichage des résultats et des graphiques
  cat("Tableau des fréquences simulés :\n")
  print(tablfreqx)
  barplot(tablfreqx, xlab = "valeurs de la variable somme",
    ylab = "Fréquences simulées", main = "Diagramme en barres")
}
> toscaneA1_4()
Tableau des fréquences simulés :
seriex
     4      5      6      7      8      9
0.0005 0.0020 0.0050 0.0135 0.0300 0.0490
    10     11     12     13     14     15
0.0785 0.1080 0.1170 0.1425 0.1610 0.1225
    16     17     18
0.0915 0.0595 0.0195
```



- On pourrait tout aussi facilement créer un dé à **21 faces équiprobables**, avec les valeurs ad-hoc pour avoir la distribution recherchée : **`rep(c(1, 2, 3, 4, 5, 6), c(1, 2, 3, 4, 5, 6))`** (voir la fonction **`toscaneA1_4bis()`**).
- Mais comment peut-on déterminer la distribution de probabilité de la variable aléatoire **X** prenant pour valeurs **x** ? Voici une stratégie possible, qui utilise une table (**`array()`**) à 3 dimensions :

```
# ALGO A1_5 : UNE FONCTION POUR CONSTRUIRE LA DISTRIBUTION DE
# PROBABILITÉ DES SOMMES OBTENUES AVEC 3 DÉS
probaS3desA1_5 <- function(){
  x <- array(0, dim = c(6, 6, 6))
  for(i in 1:6){
      for(j in 1:6){
         for(k in 1:6) x[i, j, k] <- i + j + k
                    }
                }
print(table(x) / 216)
}
> probaS3desA1_5()
x
         3          4          5          6
0.02777778 0.08333333 0.16666667 0.27777778
         7          8          9         10
0.41666667 0.58333333 0.69444444 0.75000000
        11         12         13         14
0.75000000 0.69444444 0.58333333 0.41666667
        15         16         17         18
0.27777778 0.16666667 0.08333333 0.02777778
```

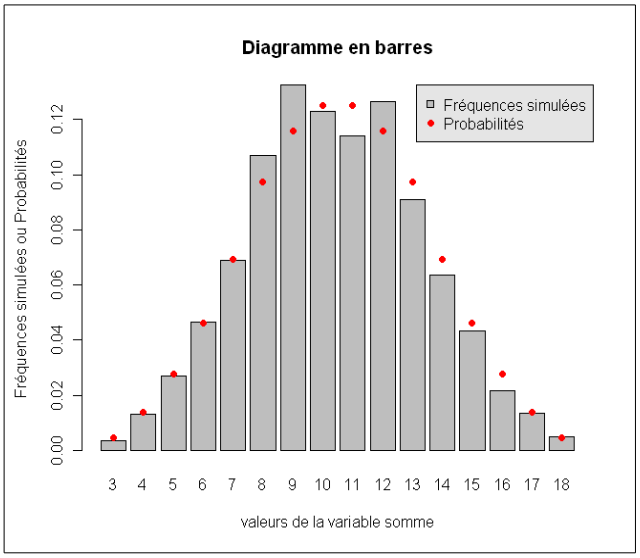


- Il est intéressant de noter que le paramétrage du nombre de jets (nbjet) est facilement réalisable dans la fonction de simulation, alors qu'il en est tout autrement dans la fonction **`probaS3desA1_5()`** déterminant la distribution de probabilité de **X**. Pour rendre ce paramétrage possible, il faut élaborer une autre stratégie (voir, **`probaS3desA1_5bis()`**, si elle existe).

► AJUSTER UNE DISTRIBUTION THÉORIQUE À UNE DISTRIBUTION SIMULÉE OBSERVÉE

- On veut superposer graphiquement la distribution simulée de **X** (fréquences simulées, barres grises) et sa distribution théorique (probabilités, points rouges). On dit que l'on a ajusté graphiquement une distribution théorique à une distribution simulée.
- Pour superposer les deux graphiques avec **R** on utilise d'abord **`barplot()`** que l'on fait suivre de **`points()`**. La présence de **`barplot(3:18, plot = FALSE)`** dans **`points()`** permet de disposer des abscisses des barres pour positionner correctement les points rouges.

```
# ALGO A1_6 : UNE FONCTION POUR SUPERPOSER LA REPRÉSENTATION
# GRAPHIQUE DE LA DISTRIBUTION SIMULÉE ET DE
#  LA DISTRIBUTION DE PROBABILITÉ
toscaneA1_6 <- function(nbsim = 2000){
  de <- seq(1, 6, 1) ; seriex <- NULL ; nbjets = 3
  for(i in 1:nbsim){
    jeu <- sample(de, nbjets, replace = TRUE)
    x <- sum(jeu)
    seriex <- c(seriex, x)
                  }
  tablfreqx <- table(seriex) / nbsim
# Distribution de probabilité de x
  probax <- array(0, dim = c(6, 6, 6))
  for(i in 1:6){
      for(j in 1:6){
         for(k in 1:6) probax[i, j, k] <- i + j + k
                  }
               }
  Distribprobax <- table(probax) / 216
# Affichage des résultats et des graphiques
  cat("Tableau des fréquences simulés :\n")
  print(tablfreqx)
  barplot(tablfreqx, xlab = "valeurs de la variable somme",
      ylab = "Fréquences simulées ou Probabilités",
      main = "Diagramme en barres")
    points(barplot(3:18, plot = FALSE), Distribprobax,
      pch = 21, col = "red", bg = "red")
    legend(x = "topright",
      legend = c("Fréquences simulées", "Probabilités"),
      bg = "grey90", pch = c(22, 21), pt.cex = c(1, 1),
      col = c("black", "red"), pt.bg = c("grey", "red"))
}
> toscaneA1_6()
Tableau des fréquences simulés :
seriex
     3      4      5      6      7      8      9     10
0.0035 0.0130 0.0270 0.0465 0.0690 0.1070 0.1325 0.1230
    11     12     13     14     15     16     17     18
0.1140 0.1265 0.0910 0.0635 0.0435 0.0215 0.0135 0.0050
```

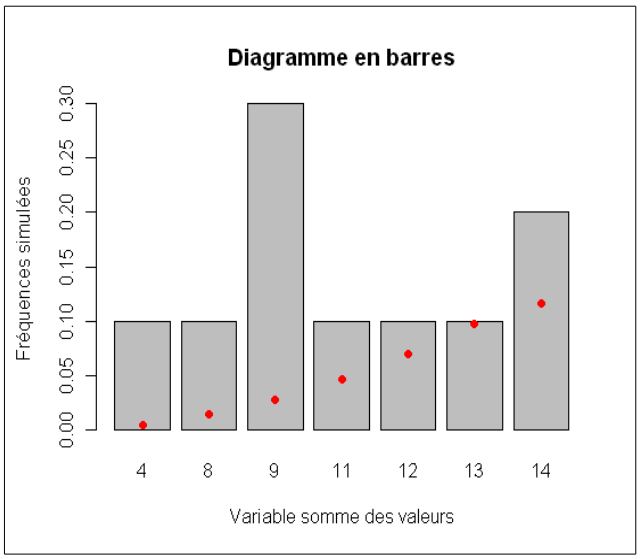


- Dans cette fonction la distribution de probabilité de **X** est contenue dans la table **Distribprobax** qui est calculée à l'intérieur de la fonction. Pour faire ces calculs, on peut utiliser une fonction externe comme le montre la fonction suivante.

- Pour éviter d'introduire les lignes de code du calcul de la distribution de probabilité, on peut en faire une fonction externe et l'appeler dans la fonction principale. C'est ce que fait la fonction contenue dans le fichier **"toscane3.R"**.
- Pour éviter de faire figurer la fonction de distribution de probabilité dans le fichier **"toscane3.R"**, on peut l'appeler dans le fichier "proba3de.R" en utilisant la commande "`source("proba3de.R")`". C'est ce qui est fait dans le fichier **"toscane3_1.R"**.

```
# ALGO A1_6 : UNE FONCTION POUR SUPERPOSER LA
# REPRÉSENTATION GRAPHIQUE DE LA DISTRIBUTION SIMULÉE
# ET DE LA DISTRIBUTION DE PROBABILITÉ

proba3de <- function(){
  x <- array(0, dim = c(6, 6, 6))
  for(i in 1:6){
    for(j in 1:6){
      for(k in 1:6){x[i, j, k] <- i + j + k}
                  }
                }
  tabloProb <- table(x) / 216
  return(tabloProb)
}
#
toscane3 <- function(nbsim = 2000){
  nbjets <- 3 ; seriex <- NULL
  de <- seq(from = 1, to = 6, by = 1)
  for(i in 1:nbsim){
    jeu <- sample(de, nbjets, replace = TRUE)
    x <- sum(jeu)
    seriex <- c(seriex, x)
                    }
  tabloProb <- proba3de()
  tabloFreq <- table(seriex) / nbsim
# Affichage des résultats et graphiques
  print(tabloFreq)
  print(tabloProb)
  barplot(tabloFreq,
          xlab = "Variable somme des valeurs",
          ylab = "Fréquences simulées",
          main ="Diagramme en barres")
    points(barplot(3:18, plot = FALSE), tabloProb,
           pch = 21, col = "red", bg = "red")
}
# Exemple où le graphique est faux :

> toscane3_1(10)
seriex
  4   8   9  11  12  13  14
0.1 0.1 0.3 0.1 0.1 0.1 0.2
x
         3          4          5          6          7
0.00462963 0.01388889 0.02777778 0.04629630 0.06944444
         8          9         10         11         12
0.09722222 0.11574074 0.12500000 0.12500000 0.11574074
        13         14         15         16         17
0.09722222 0.06944444 0.04629630 0.02777778 0.01388889
        18
0.00462963
```

```
toscane3_1 <- function(nbsim = 2000){
  source("proba3de.R")
  nbjets <- 3 ; seriex <- NULL
  de <- seq(from = 1, to = 6, by = 1)
  for(i in 1:nbsim){
    jeu <- sample(de, nbjets, replace = TRUE)
    x <- sum(jeu)
    seriex <- c(seriex, x)
                    }
  tabloProb <- proba3de()
  tabloFreq <- table(seriex) / nbsim
# Affichage des résultats et graphiques
  print(tabloFreq)
  print(tabloProb)
  barplot(tabloFreq,
          xlab = "Variable somme des valeurs",
          ylab = "Fréquences simulées",
          main ="Diagramme en barres")
    points(barplot(3:18, plot = FALSE), tabloProb,
           pch = 21, col = "red", bg = "red")
}
```



- En analysant la construction des deux graphiques on peut détecter un problème potentiel qui apparaitra en exécutant la fonction avec un petit nombre de répétitions comme avec `toscane3_1(10)`. Le graphique correspondant permet de mettre en évidence le problème.
- En effet lors de la simulation, toutes les valeurs possibles de **X** ne sont pas forcément atteintes, surtout lorsque le nombre de répétitions est faible. Alors que nous avons les probabilités de toutes les valeurs possibles de **X**.
- Il faut donc “installer” les effectifs simulés dans un tableau dans lequel figurent toutes les valeurs possibles de **X**. C'est ce qui est fait dans la fonction suivante.

► AJUSTER UNE DISTRIBUTION THÉORIQUE À UNE DISTRIBUTION SIMULÉE OBSERVÉE

- Pour résoudre le petit problème potentiellement présent dans la fonction précédente, il faut savoir ce que fait "**tabloFreqx[as.numeric(names(tablfreqx)) + 1] <- tablfreqx**", dans les lignes de commandes suivantes :

```
> n = 5 ; p = 1 / 2 ; nbsim = 20
> seriex <- rbinom(nbsim, n, p)
> (tablfreqx <- table(seriex) / nbsim)
seriex
   1    2    3    5
0.10 0.55 0.25 0.10

> (tabloFreqx <- rep(0, n + 1))
[1] 0 0 0 0 0 0
> 0:n
[1] 0 1 2 3 4 5
> names(tabloFreqx) <- 0:n
> tabloFreqx
0 1 2 3 4 5
0 0 0 0 0 0
tabloFreqx[as.numeric(names(tablfreqx) + 1)] <- tablfreqx
tabloFreqx
   0    1    2    3    4    5
0.00 0.10 0.55 0.25 0.00 0.10
```

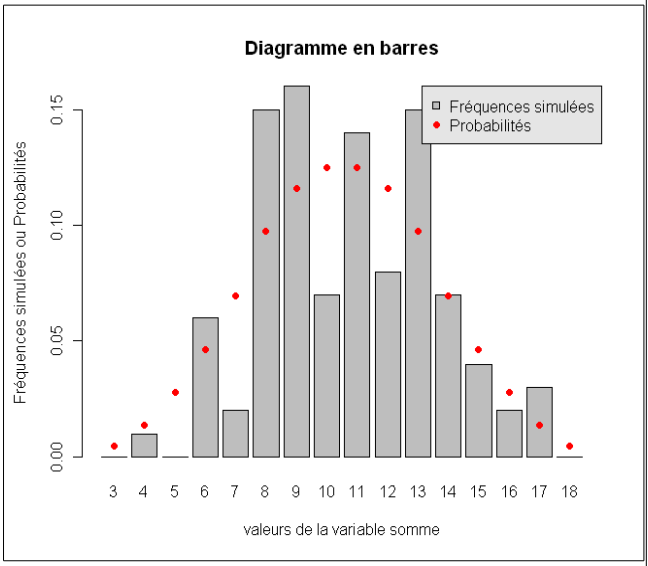
Que fait :
tabloFreqx[as.numeric(names(tablfreqx)) + 1] <- tablfreqx ?

names(tablfreqx) repère les intitulés des composantes de la liste tablfreqx, qui sont "1" "2" "3" "5" . Il manque "0" et "4".
as.numeric(names(tablfreqx) les transforme en nombres entiers, car les intitulés sont des caractères.
[as.numeric(names(tablfreqx)) + 1] les prend comme indices de la liste tabloFreqx avec + 1 car il y a un décalage de 1 entre les valeurs de la variable X qui commencent à 0 et les indices de la liste qui commencent à 1.
tabloFreqx[as.numeric(names(tablfreqx)) + 1] indique donc les indices des composantes de la liste tabloFreqx auxquelles vont être affectées les valeurs de la liste tablfreqx
On obtient donc au final :

```
>
tabloFreqx[as.numeric(names(tablfreqx)) + 1] <- tablfreqx
> tabloFreqx
   0    1    2    3    4    5
0.00 0.10 0.55 0.25 0.00 0.10
```

```
# ALGO A1_6bis : RÉSOUDRE LE PETIT PROBLÈME DE LA FONCTION
# A6 REPRÉSENTER DISTRIBUTION SIMULÉE ET DISTRIBUTION
# DE PROBABILITÉ
toscaneA1_6bis <- function(nbsim = 100){
  de <- seq(1, 6, 1) ; seriex <- NULL ; nbjets = 3
  for(i in 1:nbsim){
    jeu <- sample(de, nbjets, replace = TRUE)
    x <- sum(jeu)
    seriex <- c(seriex, x)
                    }
  tablfreqx <- table(seriex) / nbsim
# Pour avoir un tableau avec toutes les valeurs
# possibles de x
  tabloFreqx <- rep(0, 16) ; names(tabloFreqx) <- 3:18
  tabloFreqx[as.numeric(names(tablfreqx)) - 2] <- tablfreqx
# Distribution de probabilité de x
  probax <- array(0, dim = c(6, 6, 6))
  for(i in 1:6){
      for(j in 1:6){
         for(k in 1:6) probax[i, j, k] <- i + j + k
                    }
                }
  Distribprobax <- table(probax) / 216
# Affichage des résultats et des graphiques
  cat("Tableau des fréquences simulés :\n")
  print(tabloFreqx)
  barplot(tabloFreqx, xlab = "valeurs de la variable somme",
      ylab = "Fréquences simulées ou Probabilités",
      main = "Diagramme en barres")
    points(barplot(3:18, plot = FALSE), Distribprobax,
      pch = 21, col = "red", bg = "red")
    legend(x = "topright",
      legend = c("Fréquences simulées", "Probabilités"),
      bg = "grey90", pch = c(22, 21), pt.cex = c(1, 1),
      col = c("black", "red"), pt.bg = c("grey", "red"))
}
```

```
> toscaneA1_6bis(nbsim = 100)
Tableau des fréquences simulés :
   3    4    5    6    7    8    9   10
0.00 0.01 0.00 0.06 0.02 0.15 0.16 0.07
  11   12   13   14   15   16   17   18
0.14 0.08 0.15 0.07 0.04 0.02 0.03 0.00
```

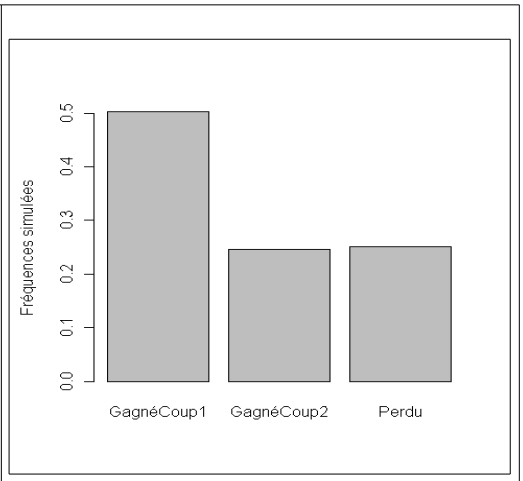


Les valeurs 3, 5, 18 ne sont pas atteintes dans la simulation.

A2. L'EXPÉRIENCE HISTORIQUE (1795) DU CROIX PILE DE D'ALEMBERT :

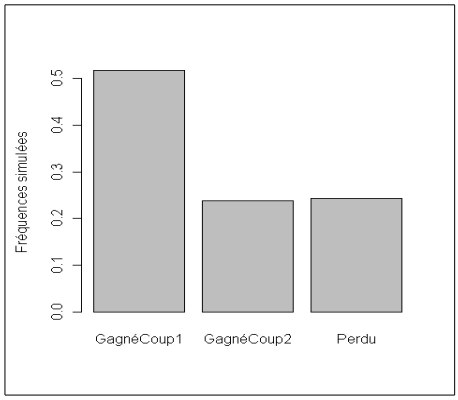
- Parier 3 contre 1 ou 2 contre 1 d'amener croix en 2 coups au jeu de croix-pile (face-pile).
- Cette expérience permet de mettre en évidence l'obstacle didactique du jeu qui s'arrête quand on gagne.
- Cet obstacle peut être levé grâce à la fonction **R** suivante, qui "mime" l'expérience décrite. Le jeu s'arrête quand on obtient croix, et on compte le nombre de fois où l'on gagne au premier jet, au second jet et quand on perd. On peut ainsi contourner la difficulté de la détermination de l'univers des résultats possibles des un ou deux jets de la pièce.

```
#*SIMULATION*DU PROBLÈME HISTORIQUE DU CROIX PILE DE D'ALAMBERT
# ALGO A2_1**FONCTION **ON S'ARRÊTE LORSQUE L'ON A AMENÉ CROIX
croixpileA2_1 <- function(nbsim = 2000){
  resultats <- rep(0, 3)
  names(resultats) <- c("GagnéCoup1", "GagnéCoup2", "Perdu")
  piece <- c("Pile", "Croix")
  for(i in 1:nbsim){
     coup1 <- sample(piece, 1)
     if(coup1 == "Croix") {resultats[1] <- resultats[1] + 1} else {
        coup2 <- sample(piece, 1)
        if(coup2 == "Croix") {resultats[2] <- resultats[2] + 1} else {
           resultats[3] <- resultats[3] + 1                          }
                                                                    }
                  }
# Affichage des résultats et des graphiques
  print(resultats / nbsim)
  barplot(resultats / nbsim, ylab = "Fréquences simulées")
}
> croixpileA2_1()
GagnéCoup1 GagnéCoup2      Perdu
    0.5025     0.2460     0.2515
```



- Là encore le **modèle équiprobable** n'est pas introduit au niveau de la pièce mais au niveau du jet simulé par la fonction **`sample()`** qui effectue un tirage équiprobable, dans les deux cotés de la pièce.
- Dans la simulation suivante (d'un jeu qui ne s'arrêterait lorsqu'on gagne) l'expérience consiste à jeter deux fois une pièce et à noter les cotés obtenus. Le jeu est gagné lors du premier croix obtenu, il peut donc être gagné au premier jet, au second jet ou perdu.

```
# *******SIMULATION**CROIX PILE DE D'ALEMBERT "SANS ARRÊT"********
# ALGO A2_2**FONCTION **SIMULATION DU JEU DE 2 JETS, PREMIER CROIX GAGNANT
premiercroixA2_2 <- function(nbsim = 2000){
  resultats <- rep(0, 3)
  names(resultats) <- c("GagnéCoup1", "GagnéCoup2", "Perdu")
  piece <- c("Pile", "Croix")
  for(i in 1:nbsim){
    jet1 <- sample(piece, 1)
    jet2 <- sample(piece, 1)
    if(jet1 == "Croix") resultats[1] <- resultats[1] + 1
    if(jet1 != "Croix" & jet2 == "Croix") resultats[2] <- resultats[2] + 1
    if(jet1 != "Croix" & jet2 != "Croix") resultats[3] <- resultats[3] + 1
                  }
# Affichage des résultats et des graphiques
  print(resultats / nbsim)
  barplot(resultats / nbsim, ylab = "Fréquences simulées")
}
> premiercroixA2_2()
GagnéCoup1 GagnéCoup2      Perdu
    0.5175     0.2385     0.2440
```



- Il est intéressant de noter la différence de stratégie de cette fonction avec la précédente. L'utilisation des connecteurs logiques la rend plus difficile à comprendre et à réaliser en autonomie par les élèves.
- Mais obtient-on vraiment les mêmes résultats qu'avec la simulation précédente ? Comment comparer les résultats des deux simulations ? On est confronté au caractère variable des résultats des simulations. De plus il faut comparer deux distributions simulées.
- Pensez-vous que les probabilités sont les mêmes ?
- Un autre simulation instructive serait de simuler le jet simultané de deux pièces indiscernables. C'est ce qui est fait par les fonctions du fichier **JetsTiragesSimultanes.r**.

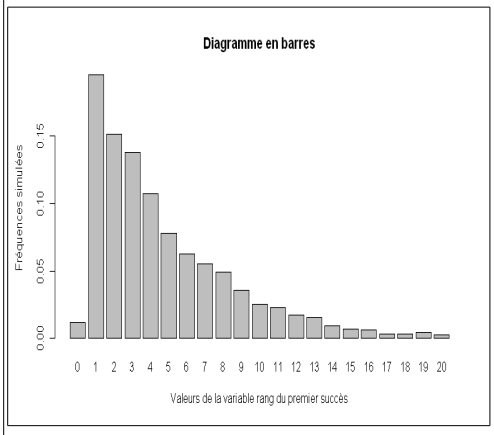
A3. GÉNÉRALISATION DU CROIX-PILE DE D'ALEMBERT

- Il s'agit d'un expérience à n épreuves identiques, chacune à deux issues, succès, échec. La variable aléatoire **G** est le rang du premier succès. Il n'est pas nécessaire de reconnaître la loi de **G** pour réaliser la simulation.
- Pour réaliser efficacement des recherches de rang dans une liste indicée, on a besoin de **`which()`**.
- On va en profiter pour comprendre comment fonctionne **`sum(valeurs logiques)`**.

```
# Modèle roulette 18 secteurs
> (roulette <- 1:18)
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18
> (x <- sample(roulette, 20, replace = TRUE))
[1] 11  6  5 12  9  2  9  8  2  9  6 18 18  4  3  4  8  4 16  7
> x == 9
 [1] FALSE FALSE FALSE FALSE  TRUE FALSE  TRUE FALSE FALSE  TRUE FALSE FALSE
[13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
> as.numeric(x == 9)
[1] 0 0 0 0 1 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0
> sum(x == 9)
[1] 3
> which(x == 9)
[1]  5  7 10
> (k <- min(which(x == 9)))
[1] 5
```

- La fonction **`which()`** remplace très avantageusement une boucle TantQue, tant que la recherche s'effectue sur une liste entièrement constituée, finie.
- Le **modèle d'échantillonnage n'est plus équiprobable**, puisque la fonction **`sample()`** tire parmi les deux alternatives “succès” “échec”, avec une distribution de probabilité : (p, 1 - p). L'intérêt pédagogique de cette fonction réside dans le fait que le tirage aléatoire est simplifié, et le modèle bien identifié.

```
# GÉNÉRALISATION DU CROIX-PILE DE D'ALEMBERT "SANS ARRÊT" :
# ALGO A3**FONCTION **SIMUL. RANG DU PREMIER SUCCÈS LORS DE N ÉPREUVES
# Modèle épreuves de Bernoulli répétées (indépendantes)   n, p
rangpremiersuccesA3 <- function(n = 20, p = 1 / 5, nbsim = 2000){
  deuxalternatives <- c("succes", "echec")
  seriesimx <- NULL
  for(i in 1:nbsim){
    x <- sample(deuxalternatives, n, prob = c(p, 1 - p), replace = T)
    if(sum(x == "succes") > 0){
      seriesimx[i] <- min(which(x == "succes"))
                          } else {
      seriesimx[i] <- 0               }
                  }
  tablfreqx <- table(seriesimx) / nbsim
# Affichage des résultats et des graphiques
  cat("\nTableau de distribution des fréquences de la\n",
      "variable Rang du premier succès\n")
  print(tablfreqx)
  barplot(tablfreqx,
    xlab = "Valeurs de la variable rang du premier succès",
    ylab = "Fréquences simulées", main = "Diagramme en barres")
}
> rangpremiersuccesA3()
Tableau de distribution des fréquences de la
 variable Rang du premier succès
seriesimx
     0      1      2      3      4      5      6      7      8
0.0120 0.1950 0.1510 0.1375 0.1070 0.0780 0.0625 0.0550 0.0490
     9     10     11     12     13     14     15     16     17
0.0355 0.0250 0.0230 0.0175 0.0155 0.0095 0.0070 0.0065 0.0035
    18     19     20
0.0030 0.0045 0.0025
```



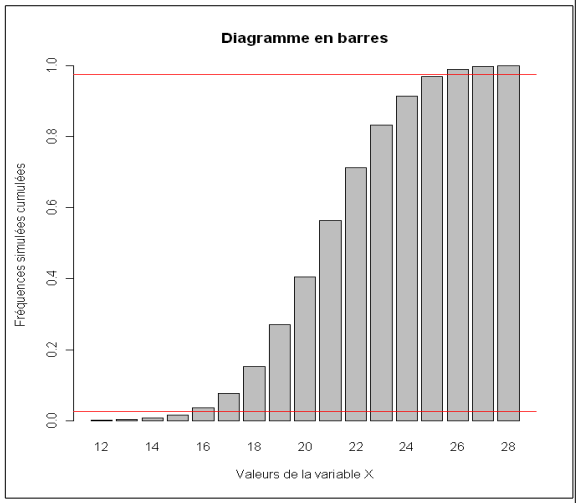
- L'enjeu didactique de cette simulation c'est de pouvoir obtenir une distribution simulée de **G** sans passer par la “case”modèle mathématique de cette loi. Un élève peut ainsi répondre à une question faisant intervenir cette loi sans être obligé de l'identifier. Il pourra proposer une solution asymptotiquement exacte.

A4. SIMULER UN INTERVALLE DE FLUCTUATION D'UNE VARIABLE FRÉQUENCE

- On simule **nbsim** = 2000 échantillons (tirages) avec remise, de taille **n** = 10 dans une urne de **nbtot** = 100 boules dont **nbrouges** = 70 rouges. **X** est la variable aléatoire prenant pour valeur le nombre de boules rouges obtenue. On obtient ainsi 2000 valeurs de **X** dans la liste **serieX**.
- L'intervalle de fluctuation bilatéral (au sens du programme de première S actuel) simulé, de probabilité 0,95, de la variable **X** est déterminé par les deux **Q**uantiles à 2,5 % et 97,5 % de la série **serieX**. On notera $IF^*_{0,95}$ de X = $[Q_{2,5\%} ; Q_{97,5\%}]$ et $IF^*_{0,95}$ de **X / n** = $[Q_{2,5\%} / n ; Q_{97,5\%} / n]$
- Un premier enjeu didactique est de montrer que l'on peut établir un intervalle de fluctuation simulé sans se servir du modèle binomial. On a simplement utilisé le **modèle équiprobable** généré par la fonction **sample()**.

```
# ALGO A4---Intervalle de fluctuation bilatéral SIMULÉ d'une
# variable X nombre de boules rouges dans un échantillon de n la
# et de variable fréquence X/n, de probabilité minimale 1 - e
# Le principe est de simuler nbsim répétitions du tirage d'un
# échantillon de taille n
# On obtient une série statistique de nbsim valeurs.
# Les valeurs simulées de a et b seront les quantiles d'ordre
# e/2 et - e/2 de cette série.
#----------------Version de base simulIF_1()
simulIF_A4 = function(n = 10, nbrouges = 70, nbtot = 100,
              e = .05, nbsim = 2000){
  urne <- rep(c("rouge", "autre"), c(nbrouges, nbtot - nbrouges))
  serieX <- NULL
  for(i in 1:nbsim){
    tirage <- sample(urne, n, replace = TRUE)
    X <- sum(tirage == "rouge")
    serieX <- c(serieX, X)
                    }
  tablFreqX <- table(serieX) / nbsim
  tablFreqCumX <- cumsum(tablFreqX)
  quantserieX <- quantile(serieX, probs = c(e / 2, 1 - e / 2))
  propIF_sim <- sum(serieX >= quantserieX[1] &
                    serieX <= quantserieX[2]) / nbsim
# Affichage des résultats
  barplot(tablFreqCumX,
    xlab = "Valeurs de la variable X",
    ylab = "Fréquences simulées cumulées",
    main = "Diagramme en barres")
    abline(h = c(e / 2, 1 - e / 2), col = "red")
  cat("Distribution simulée de la variable X\n")
  print(tablFreqX)
  cat("\nQuantiles série X\n")
  print(quantserieX)
  cat("\nQuantiles série X / n\n")
  print(quantserieX / n)
  cat("\nPourcentage de valeurs de la série comprises dans
   l'IF :",
      propIF_sim, "\n\n")
}
```

```
> simulIF_A4(n = 30)
Distribution simulée de la variable X
serieX
    12     13     14     15     16     17
0.0005 0.0020 0.0055 0.0085 0.0205 0.0390
    18     19     20     21     22     23
0.0770 0.1170 0.1345 0.1590 0.1485 0.1220
    24     25     26     27     28
0.0800 0.0565 0.0190 0.0085 0.0020

Quantiles série X
 2.5% 97.5%
   16    26

Quantiles série X / n
     2.5%     97.5%
0.5333333 0.8666667

Pourcentage de valeurs de la série comprises dans l'IF : 0.973
```



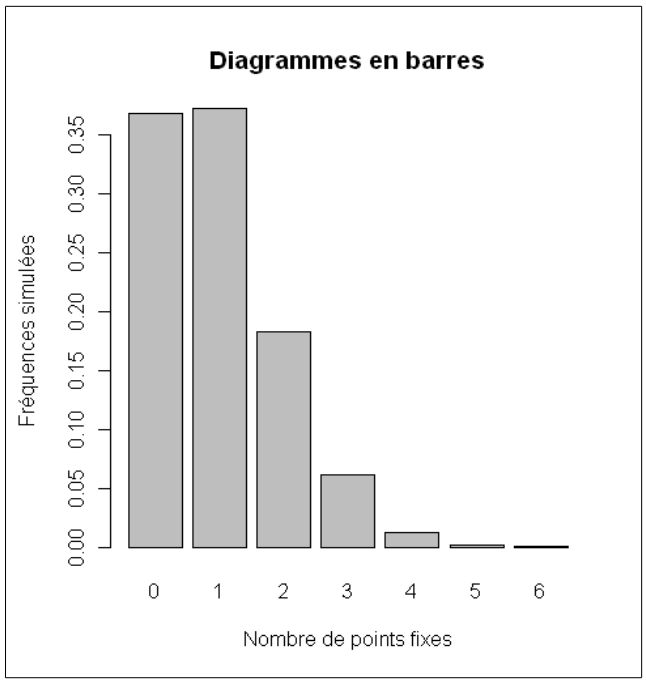
- Un deuxième enjeu didactique est de réinvestir les quantiles, outils de statistique descriptive, objet de nombreuses “controverses”quant à leurs calculs. **R** propose 9 types de calcul, parmi lesquels on retrouve ceux donnant les mêmes résultats que les tableurs classiques, que GeoGebra, que les calculettes ...

B. LA SIMULATION COMME OUTIL DE RÉSOLUTION DE PROBLÈMES : ÉLABORER DES STRATÉGIES

B1. LE PROBLÈME HISTORIQUE DES CHAPEAUX DE MONTMORT (1708) OU UNE HISTOIRE DE PERMUTATIONS SANS POINT FIXE

- 20 personnes assistant à une réunion accrochent leurs chapeaux à 20 crochets dans le couloir. Il y a une coupure d'électricité à la fin de la réunion et elles reprennent toutes leur chapeau aléatoirement. Quelle est la probabilité qu'aucune d'entre elles ne tombe sur son chapeau?
- C'est un problème dont la résolution mathématique n'est pas accessible, en autonomie, aux élèves du lycée.
- La stratégie pour repérer les points fixes consiste à faire la différence entre la liste des nombres de 1 à 20 et une permutation de cette liste. Les zéro de la liste des différences indiquent les points fixes.

```
#**LES CHAPEAUX DE MONTMORT (1708)**
#** PERMUTATIONS SANS POINT FIXE**
chapeauxdB2 = function(m = 20, nbsim = 2000){
  crochets <- seq(1, m) ; serienbfixe <- NULL
  sansfixe <- 0
  for(i in 1:nbsim){
     chapeaux <- sample(crochets, m)
     fixes <- crochets - chapeaux
     nbfixes <- sum(fixes == 0)
     serienbfixe <- c(serienbfixe, nbfixes)
                   }
  tabloFreqX <- table(serienbfixe) / nbsim
  FreqXZero <- tabloFreqX[which(names(tabloFreqX) == "0")]
# Affichage des résultats et des graphiques
  cat("Distribution des fréquences simulées du nombre de
points fixes\n")
  print(tabloFreqX)
  cat("\nEstimation de la probabilité d'aucun des", m,
      "chapeaux à leur place =\n", FreqXZero, "\n\n")
  barplot(tabloFreqX, xlab = "Nombre de points fixes",
    ylab = "Fréquences simulées",
    main = "Diagrammes en barres")
}
> chapeauxdB2()
Distribution des fréquences simulées du nombre de points
fixes
serienbfixe
     0      1      2      3      4      5      6
0.3675 0.3720 0.1825 0.0620 0.0125 0.0020 0.0015

Estimation de la probabilité d'aucun des 20 chapeaux à
leur place =
 0.3675
```



- C'est un bon exemple de problème dont la résolution* (* signifie asymptotiquement exacte) est rendue possible par une simulation élémentaire. Cette simulation ne nécessite que les outils algorithmiques et statistiques des programmes du lycée.
- L'exemple suivant présente aussi un cas de problème accessible, par la simulation, aux élèves du lycée, en utilisant une fonction particulière de **R** permettant de compter les “suites”d'éléments identiques qui se suivent dans une série statistique, appelées “chaînes ”ou “runs” en anglais.

B2. LE PROBLÈME DES CHAÎNES DE LONGUEUR 6 (OU PLUS)

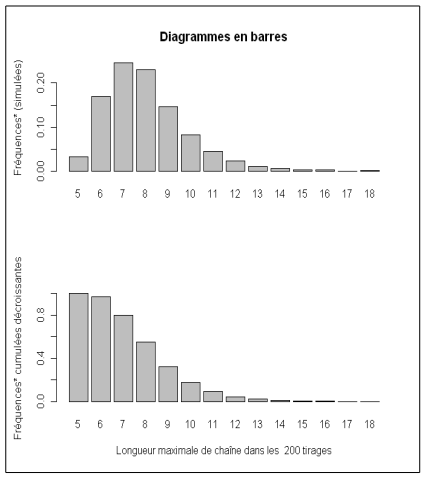
- On lance 200 fois une pièce équilibrée, quelle est la probabilité d'obtenir une suite d'au moins 6 piles ou 6 faces consécutifs (on dit une “chaines” de longueur 6).
- C'est un autre exemple de problème dont la résolution n'est pas accessible, en autonomie, aux élèves du lycée.
- Nous allons établir la distribution de la variable **LMax** longueur de la chaine de longueur maximale. Il faut utiliser la fonction interne **`rle()`** qui fournit un tableau des effectifs des longueurs de toutes les chaines rencontrées.
- Pour comprendre comment marche la fonction **`rle()`** de **R** qui détecte et comptabilise les “chaines” voici un exemple détaillé en lignes de commandes :

```
# QueFait rle() ?
> L = 6 ; piece = c("P", "F") ; tirages = 20
> serieLMax <- NULL ; i <- 1
> #
> (experience <- sample(piece, tirages, replace = T))
 [1] "F" "F" "F" "P" "F" "F" "P" "P" "F" "P" "P" "P"
[13] "F" "P" "P" "P" "F" "P" "F" "F"
> (chaines <- rle(experience))
Run Length Encoding
  lengths: int [1:11] 3 1 2 2 1 3 1 3 1 1 ...
  values : chr [1:11] "F" "P" "F" "P" "F" "P" ...
> chaines$length
 [1] 3 1 2 2 1 3 1 3 1 1 2
> serieLMax[i] <- max(chaines$length)
> i
[1] 1
>  serieLMax
[1] 3
> (FreqLongSupL <- sum(serieLMax >= L) / i)
[1] 0
> #
> i <- i + 1
```

```
> (experience <- sample(piece, tirages, replace = T))
 [1] "P" "P" "F" "P" "F" "P" "P" "P" "F" "P" "P" "F"
[13] "P" "P" "F" "F" "P" "F" "F" "P"
> (chaines <- rle(experience))
Run Length Encoding
  lengths: int [1:13] 2 1 1 1 3 1 2 1 2 2 ...
  values : chr [1:13] "P" "F" "P" "F" "P" "F" ...
> chaines$length
 [1] 2 1 1 1 3 1 2 1 2 2 1 2 1
> serieLMax[i] <- max(chaines$length)
> i
[1] 2
>  serieLMax
[1] 3 3
> (FreqLongSupL <- sum(serieLMax >= L) / i)
[1] 0
> #
> i <- i + 1
●●● Après 48 simulations de 20 jets, on obtient :
> i
[1] 48
>  serieLMax
 [1] 3 3 3 6 3 5 5 4 3 5 4 5 3 4 3 6 4 6 3 5 5 6 3 4
[25] 8 5 4 4 5 3 5 6 3 5 8 3 7 3 5 7 2 4 5 4 4 5 5 5
> (FreqLongSupL <- sum(serieLMax >= L) / i)
[1] 0.1875
```

- La fonction suivante permet d'obtenir une série de valeurs simulées de **LMax**, dans la liste **`serieLMax`** et d'en faire la description sous forme de tableau des fréquences et de diagrammes en barres.

```
# ALGO B1*SIMULATION*DU PROBLÈME DES "CHAÎNES" DE LONGUEUR 6
chainesdB1 = function(L = 6, tirages = 200, nbsim = 2000){
  serieLMax <- NULL ; piece = c("Pile", "Face")
  for(i in 1:nbsim){
    expe <- sample(piece, tirages, replace = TRUE)
    rexpe <- rle(expe)
    serieLMax <- c(serieLMax, max(rexpe$length))
                  }
  tablemax <- table(serieLMax)
  Ltable <- length(tablemax)
  StatL <- sum(tablemax[(as.numeric(names(tablemax)) >= L)]) / nbsim
# Affichage des résultats et des graphiques
  cat("Une estimation de la probabilité de chaînes de longueur",
       L, "ou plus")
  cat("vaut :", StatL, "\n\n")
  print("Distribution* de la chaîne (run) de longueur maximale ")
  print(tablemax / nbsim)
  par(mfrow = c(2, 1))
  barplot(tablemax / nbsim, main = "Diagrammes en barres",
    ylab = "Fréquences* (simulées)")
  barplot(cumsum(tablemax[Ltable:1])[Ltable:1] / nbsim,
    xlab = paste("Longueur maximale de chaîne dans les ",
    tirages, "tirages"),
  ylab = "Fréquences* cumulées décroissantes")
}
> chainesdB1()
Une estimation de la probabilité de chaînes de longueur 6 ou
plus vaut : 0.9675

[1] "Distribution* - simulée - de la chaîne de longueur maximale"
serieLMax
     5      6      7      8      9     10     11     12     13     14     15
0.0325 0.1700 0.2455 0.2305 0.1470 0.0820 0.0455 0.0240 0.0105 0.0055 0.0025
    16     17     18
0.0030 0.0005 0.0010
```



- On remarque que dans ces 2000 simulations de 200 lancers il y 9 simulations dans lesquelles la chaine de longueur maximale est de taille supérieure ou égale à 16. Un bon exemple contre-intuitif !
- Dans cet autre exemple, la simulation rend sa résolution accessible à des élèves de lycée. L'algorithme reste simple si l'on utilise la fonction **`rle()`**.

B3. 2012MarsNelleCaledonieOblig. Un dé et 2 urnes

- Tous les exercices suivants on été résolus par des simulations élémentaires ne faisant intervenir que les outils algorithmiques des programmes du lycée.
- Exercice 2 Question 1 b) L'énoncé figurent en annexe A.

```
# Bac S Nouvelle Calédonie Mars 2012 Exercice 2 Question 1 b
# Urne1 contient 3 boules rouges et 1 noire, Urne2 contient 3 rouges et 2 noires
# Dé équilibré à 6 faces numérotées 1 à 6.
# Une partie consiste à jeter le dé, puis
# si le résultat est 1 tirer une boule dans Urne1 sinon tirer une boule dans Urne2
#  Question1 b : Calculer la probabilité d'obtenir une boule noire (événement B)
# FONCTION NelleCaled2012_1b_1 version1 Cumul dans la variable boulenoiretir
NelleCaled2012_1b_1 <- function(nbsim = 2000){
  decub <-1:6
  urne1 <- rep(c("brouge", "bnoire"), c(3, 1))
  urne2 <- rep(c("brouge", "bnoire"), c(3, 2))
  boulenoire <- 0
  for(i in 1:nbsim){
    resultDe <- sample(decub, 1)
    if(resultDe == 1)              {
      resultUrne <- sample(urne1, 1)} else {
      resultUrne <- sample(urne2, 1)       }
    if(resultUrne == "bnoire") {boulenoire <- boulenoire + 1}
                  }
# AFFICHAGE DES RÉSULTATS
  cat("\nEstimation de la probabilité de tirer une boule noire :\n",
       boulenoire / nbsim, "\n")
}
> NelleCaled2012_1b_1()
Estimation de la probabilité de tirer une boule noire :
 0.3735
```

- Exercice 2 Question 1c)

```
# Bac S Nouvelle Calédonie Mars 2012 Exercice 2 Question 1 c
# Question1 c : sachant qu'on a tiré une boule noire calculer la
# probabilité d'avoir obtenu 1 au dé (événement A)
# FONCTION NelleCaled2012_1c_1 version1
# Cumul dans les variables boulenoireetde1 et de1
NelleCaled2012_1c_1 <- function(nbsim = 2000){
  decub <-1:6
  urne1 <- rep(c("brouge", "bnoire"), c(3, 1))
  urne2 <- rep(c("brouge", "bnoire"), c(3, 2))
  boulenoireetde1 <- 0 ; boulenoire <- 0
  for(i in 1:nbsim){
    resultDe <- sample(decub, 1)
    if(resultDe == 1)                          {
      resultUrne <- sample(urne1, 1)
      if(resultUrne == "bnoire")         {
         boulenoireetde1 <- boulenoireetde1 + 1
         boulenoire <- boulenoire + 1}
                                          } else {
      resultUrne <- sample(urne2, 1)
      if(resultUrne == "bnoire") {boulenoire <- boulenoire + 1}
                                                  }
                  }
  de1sachantboulenoire <- boulenoireetde1 / boulenoire
# AFFICHAGE DES RÉSULTATS
  cat("\nEstimation de la probabilité de tirer une boule noire :\n",
       boulenoire / nbsim, "\n")
  cat("\nEstimation de la probabilité d'obtenir 1 au dé sachant boule noire :\n",
```

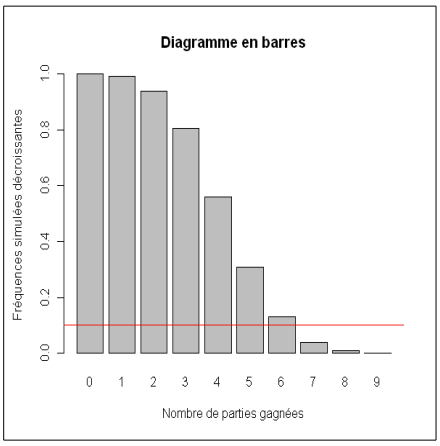
```
      de1sachantboulenoire, "\n")
}
> NelleCaled2012_1c_1()
Estimation de la probabilité de tirer une boule noire :
 0.3695
Estimation de la probabilité d'obtenir 1 au dé sachant boule noire :
 0.09878214
```

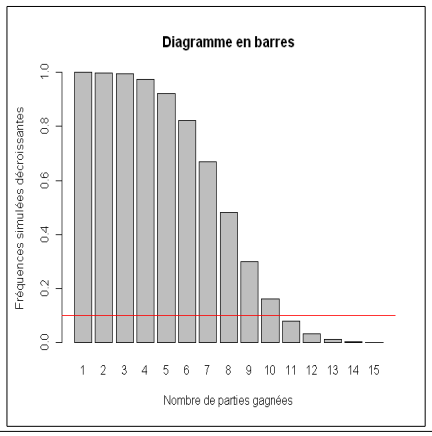
- Exercice 2 Question 2a) et 2b)

```
# Bac S Nouvelle Calédonie Mars 2012 Exercice 2 Questions 2a et 2b
# Une partie est gagnée si la boule tirée est noire. Une personne joue 10 parties.
# La variable aléatoire X prend pour valeur le nombre de parties gagnées sur les 10.
# La loi de X peut être modélisée par une loi bien connue.
# On peut aussi simuler l'expérience de la question 2
#  Question2 a : Calculer la probabilité de gagner exactement 3 parties sur 10
#  Question2 b : Calculer la probabilité de gagner au moins 1 partie sur 10
# FONCTION NelleCaled2012_2ab_1 version1 Cumul
NelleCaled2012_2ab_1 <- function(nbsim = 2000, nbpartie = 10,
                                 nbgagneexact = 3, nbgagneaumoins = 1){
  decub <-1:6
  urne1 <- rep(c("brouge", "bnoire"), c(3, 1))
  urne2 <- rep(c("brouge", "bnoire"), c(3, 2))
  gagneexact <- 0 ; gagneaumoins <- 0
  for(i in 1:nbsim){
    gagne <- 0
    for(k in 1:nbpartie){
    resultDe <- sample(decub, 1)
    if(resultDe == 1)              {
      resultUrne <- sample(urne1, 1)} else {
      resultUrne <- sample(urne2, 1)       }
    if(resultUrne == "bnoire") {gagne <- gagne + 1}
                        }
    if(gagne == nbgagneexact) {gagneexact <- gagneexact + 1}
    if(gagne >= nbgagneaumoins) {gagneaumoins <- gagneaumoins + 1}
                  }
# AFFICHAGE DES RÉSULTATS
  cat("\nEstimation de la probabilité de gagner", nbgagneexact, "fois exactement :\n",
      gagneexact / nbsim, "\n")
  cat("\nEstimation de la probabilité de gagner au moins", nbgagneaumoins, "fois :\n",
      gagneaumoins / nbsim, "\n")
}

> NelleCaled2012_2ab_1()
Estimation de la probabilité de gagner 3 fois exactement :
 0.2425
Estimation de la probabilité de gagner au moins 1 fois :
 0.988

> NelleCaled2012_2ab_1(nbpartie = 20, nbgagneexact = 6, nbgagneaumoins = 1)
Estimation de la probabilité de gagner 6 fois exactement :
 0.155
Estimation de la probabilité de gagner au moins 1 fois :
 0.9995
```

- Exercice 2 Question 2c)

```
# Bac S Nouvelle Calédonie Mars 2012 Exercice 2 Questions 2c (sans utiliser le tableau)
# Question2 c : Soit un entier N compris entre 1 et 10 et l'événement C : la personne
# gagne au moins N parties sur les 10. A partir de quelle valeur de N la probabilité
# de cet événement est-elle inférieure à 1/10 ?
# FONCTION NelleCaled2012_2c_1 version1 tableau des fréquences à k pas toujours complet
# On passe à la gestion de listes indicées et intitulées(vecteurs) : seriegagne, tabfreq
NelleCaled2012_2c_1 <- function(nbsim = 2000, nbpartie = 10, probaseuil = 1 / 10){
  decub <-1:6
  urne1 <- rep(c("brouge", "bnoire"), c(3, 1))
  urne2 <- rep(c("brouge", "bnoire"), c(3, 2))
  seriegagne <- NULL
  for(i in 1:nbsim){
    gagne <- 0
    for(k in 1:nbpartie){
    resultDe <- sample(decub, 1)
    if(lancede == 1)              {
      resultUrne <- sample(urne1, 1)} else {
      resultUrne <- sample(urne2, 1)       }
    if(resultUrne == "bnoire") {gagne <- gagne + 1}
                          }
    seriegagne[i] <- gagne
                   }
    tabfreq <- table(seriegagne) / nbsim
    tabfreqcum <- cumsum(tabfreq)
    tabfreqcumdec <- 1 - tabfreqcum + tabfreq
    N <- min(as.numeric(names(which(tabfreqcumdec < probaseuil))))
# AFFICHAGE DES RÉSULTATS
#  cat("\nDistribution estimée de la variable X nombre de partie(s) gagnée(s)\n")
#  print(tabfreq)
#  cat("\nRépartition estimée de la variable X nombre de partie(s) gagnée(s)\n")
#  print(tabfreqcum)
  cat("\nFréquences cumulées décroissantes estimées de la variable X\n")
  print(tabfreqcumdec)
  cat("\nPlus petite valeur N de X telle que P(X>=N)<",
probaseuil, "=", N, "\n")
}
> NelleCaled2012_2c_1()
Fréquences cumulées décroissantes estimées de la variable X
seriegagne
     0      1      2      3      4      5      6      7
1.0000 0.9915 0.9390 0.8045 0.5605 0.3070 0.1315 0.0405
     8      9
0.0080 0.0015

Plus petite valeur N de X telle que P(X>=N)< 0.1 = 7

> NelleCaled2012_2c_1(nbpartie = 20)
Fréquences cumulées décroissantes estimées de la variable X
seriegagne
     1      2      3      4      5      6      7      8
1.0000 0.9990 0.9945 0.9735 0.9225 0.8220 0.6685 0.4805
     9     10     11     12     13     14     15
0.2985 0.1615 0.0800 0.0340 0.0110 0.0035 0.0005

Plus petite valeur N de X telle que P(X>=N)< 0.1 = 11

> NelleCaled2012_2c_1(nbpartie = 20, probaseuil = .05)
Fréquences cumulées décroissantes estimées de la variable X
seriegagne
     1      2      3      4      5      6      7      8
1.0000 0.9975 0.9925 0.9675 0.9220 0.8175 0.6675 0.4845
     9     10     11     12     13     14     15     16
0.3080 0.1725 0.0775 0.0325 0.0100 0.0025 0.0010 0.0005
```





```
Plus petite valeur N de X telle que P(X>=N)< 0.05 = 12
```

- **Version Texas** de la simulation des réponses aux questions 1b et 1c :

Version indentée ; ne fonctionne pas comme programme :

```
Input"NOMBRE DE SIMULATIONS", M
0->A:0->B:0->C
For(I,1,M)
  6×NbrAléat->D
  If D<1
    Then
      A+1->A
      4×NbrAléat->U
      If U<1
        Then
          C+1->C
          B+1->B
      End
    Else
      5×NbrAléat->U
      If U<2
        Then
          B+1->B
      End
  End
End
Disp"PROBA DE1",A/M,"PROBANOIRES",B/M
Disp"PROBA DE1 SACHANT NOIRE",C/B
```

Version non indentée (écran machine) :

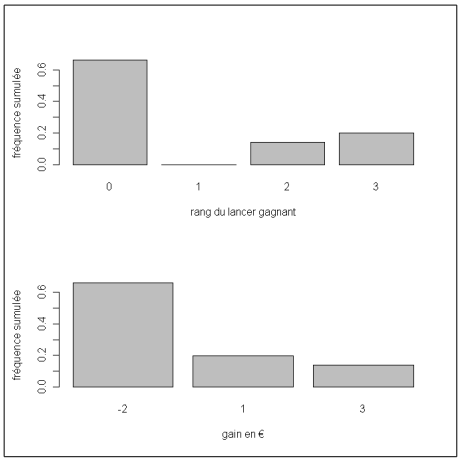
```
Input"NOMBRE DE SIMULATIONS", M
0->A:0->B:0->C
For(I,1,M)
6×NbrAléat->D
If D<1
Then
A+1->A
4×NbrAléat->U
If U<1
Then
C+1->C
B+1->B
End
Else
5×NbrAléat->U
If U<2
Then
B+1->B
End
End
End
Disp"PROBA DE1",A/M,"PROBANOIRES",B/M
Disp"PROBA DE1 SACHANT NOIRE",C/B
```

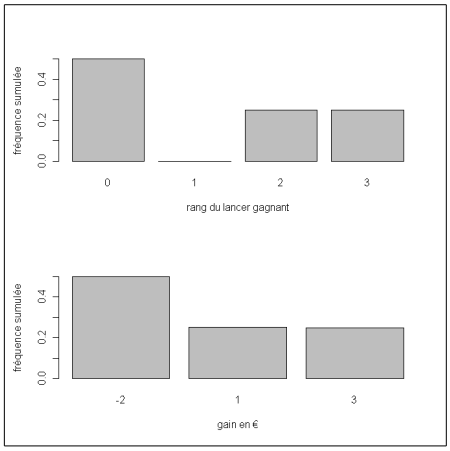
B4. 2011-S-Avril-Pondichéry : fléchettes, Épreuves successives, répétées, à trois issues

```
# On arrête les lancers dès que la partie est gagnée.
# Distribution simulée de variables
# et estimation de probabilités.
flechesD <- function(cible = c(0, 3, 5), proba = c(3/6, 2/6, 1/6),
                     n = 3, gagne = 8, nbsim = 2000){
 vectgagne <- vector(length = nbsim)
 tablogagne <- rep(0, n + 1)
 names(tablogagne) <- 0:n
 for(i in 1:nbsim){
   nblancers <- 0
   CumParties <- 0
   while(CumParties < gagne & nblancers < n){
      partie <- sample(cible, 1, proba, replace = TRUE)
      CumParties <- CumParties + partie
      nblancers <- nblancers + 1            }
   if(nblancers == n & CumParties < gagne) {vectgagne[i] <- 0} else {
      vectgagne[i] <- nblancers                                    }
                }
 tablogagne[as.numeric(names(table(vectgagne))) + 1 ] <- table(vectgagne)
 AuMoins1sur6 <- 1 - (tablogagne[1] / nbsim)^6
 distribX <- c(tablogagne[1], tablogagne[4], tablogagne[3]) / nbsim
 names(distribX) <- c(-2, 1, 3)
 esperanceX <- sum(distribX * as.numeric(names(distribX)))
#****************** Affichage des résultats **********************
 cat("\nDistribution simulée du rang du lancers gagnant sur 3 lancers \n")
 print(tablogagne / nbsim)
 cat("\nEstimation de la proba de perdre après 3 lancers =",
    tablogagne[1] / nbsim, "\n")
 cat("Estimation de la proba de gagner en 1 lancer =",
     tablogagne[2] / nbsim, "\n")
 cat("Estimation de la proba de gagner en 2 lancers =",
     tablogagne[3]/nbsim, "\n")
 cat("Estimation de la proba de gagner en 3 lancers =",
     tablogagne[4] / nbsim, "\n")
 cat("Estimation de la proba de gagner =",
     sum(tablogagne[2:4]) / nbsim, "\n")
 cat("Estimation de la proba degagner au moins une partie sur 6 =",
     AuMoins1sur6, "\n")
 cat("\n Distribution sumulée de la variable aléatoires X :\n")
 print(distribX)
 cat("Moyenne simulée de X =",esperanceX, "€ \n")
 par(mfrow = c(2, 1))
 barplot(tablogagne / nbsim, xlab = "rang du lancer gagnant",
         ylab = "fréquence sumulée")
 barplot(distribX, xlab = "gain en €", ylab = "fréquence simulée")
}
```

```
flechesD(gagne = 6)
 Distribution simulée du rang du lancers gagnant sur 3 lancers
     0      1      2      3
0.5005 0.0000 0.2490 0.2505

 Estimation de la proba de perdre après 3 lancers = 0.5005
 Estimation de la proba de gagner en 1 lancer = 0
 Estimation de la proba de gagner en 2 lancers = 0.249
 Estimation de la proba de gagner en 3 lancers = 0.2505
 Estimation de la proba de gagner = 0.4995
Estimation de la proba degagner au moins une partie sur 6 = 0.984281

 Distribution sumulée de la variable aléatoires X :
    -2      1      3
0.5005 0.2505 0.2490
 Moyenne simulée de X = -0.0035 €
```

```
flechesD()
 Distribution simulée du rang du
lancers gagnant sur 3 lancers
    0     1     2     3
0.662 0.000 0.139 0.199

 Estimation de la proba de perdre
après 3 lancers = 0.662
 Estimation de la proba de gagner
en 1 lancer = 0
 Estimation de la proba de gagner
en 2 lancers = 0.139
 Estimation de la proba de gagner
en 3 lancers = 0.199
 Estimation de la proba de gagner =
0.338
Estimation de la proba degagner au
moins une partie sur 6 = 0.9158318

 Distribution sumulée de la
variable aléatoires X :
   -2     1     3
0.662 0.199 0.139
 Moyenne simulée de X = -0.708 €
```





- À travers ces quelques exemples nous mettons en évidence la possibilité d'une résolution alternative, par simulation, des exercices de probabilité du baccalauréat. C'est une bonne occasion pour réinvestir l'algorithmique et la programmation, et ainsi motiver les élèves à leur apprentissage.

C. EXPLOITER LES RÉSULTATS D'UNE EXPÉRIENCE PRATIQUE, SIMULER UN PROLONGEMENT

C1. IMPORTER LES RÉSULTATS D'UNE EXPÉRIENCE PRATIQUE DEPUIS UNE FEUILLE DE TABLEUR

- C'est une expérience de contrôle de qualité d'un lot de semences, reproduit en utilisant des bouteilles, et plus récemment des "biberons". Le protocole complet figure dans les documents **ControlQualiteSem1Simul1Consignes.odt** et **R_TraitmentSemSimTD1.odt**. Je ne présente ici que quelques aspects du traitement des résultats.

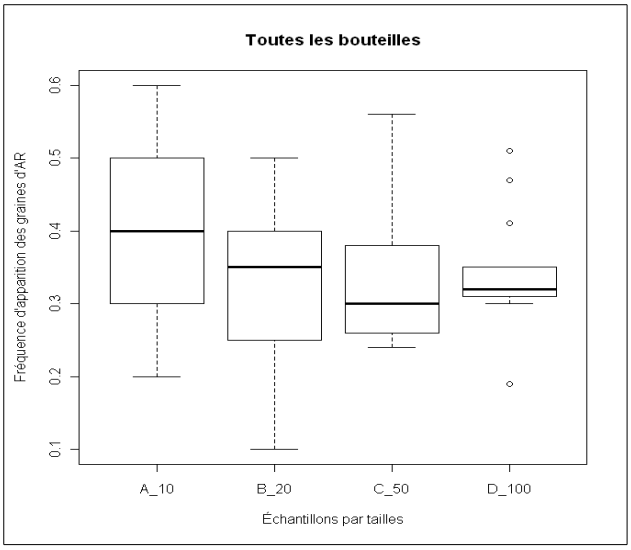
```
setwd("E:/HubW/CoursMath/CoursProba/Simulation/TPTD_Simul")
sem1 <- read.table("DataSem1Simul1BTS2011.csv", sep = ";", header = TRUE, dec = ",")
names(sem1)
[1] "noms"    "bouchon" "A_10"    "B_20"    "C_50"    "D_100"
sem1
        noms bouchon A_10 B_20 C_50 D_100
1  BOUNAUDET   ROUGE  0.3 0.40 0.38  0.32
2    DIACONO   ROUGE  0.4 0.40 0.24  0.41
3     GILLES   ROUGE  0.5 0.10 0.40  0.31
4   IGLESIAS    BLEU  0.4 0.35 0.52  0.51
5    LABREZE    BLEU  0.5 0.45 0.56  0.47
6    LATTIER   ROUGE  0.2 0.25 0.28  0.32
7  LHUILLERY   ROUGE  0.5 0.50 0.36  0.35
8    MARTROU   ROUGE  0.3 0.35 0.36  0.35
9    NOVELLA   ROUGE  0.2 0.25 0.30  0.32
10    VERGNE   ROUGE  0.4 0.50 0.24  0.31
11    BARDET   ROUGE  0.2 0.30 0.28  0.32
12      LUNA   ROUGE  0.3 0.25 0.30  0.30
13    POADAE   ROUGE  0.6 0.30 0.26  0.19
14   REBIERE   ROUGE  0.4 0.35 0.24  0.35

summary(sem1)
        noms     bouchon        A_10            B_20            C_50            D_100
 BARDET   :1   BLEU : 2   Min.   :0.2000   Min.   :0.1000   Min.   :0.2400   Min.   :0.1900
 BOUNAUDET:1   ROUGE:12   1st Qu.:0.3000   1st Qu.:0.2625   1st Qu.:0.2650   1st Qu.:0.3125
 DIACONO  :1              Median :0.4000   Median :0.3500   Median :0.3000   Median :0.3200
 GILLES   :1              Mean   :0.3714   Mean   :0.3393   Mean   :0.3371   Mean   :0.3450
 IGLESIAS :1              3rd Qu.:0.4750   3rd Qu.:0.4000   3rd Qu.:0.3750   3rd Qu.:0.3500
 LABREZE  :1              Max.   :0.6000   Max.   :0.5000   Max.   :0.5600   Max.   :0.5100
 (Other)  :8

Création d'un "data.frame" ne comprenant pas les données issues des bouchons bleus :
(sem1_1 <- sem1[sem1$bouchon != "BLEU", ])
        noms bouchon A_10 B_20 C_50 D_100
1  BOUNAUDET   ROUGE  0.3 0.40 0.38  0.32
2    DIACONO   ROUGE  0.4 0.40 0.24  0.41
3     GILLES   ROUGE  0.5 0.10 0.40  0.31
6    LATTIER   ROUGE  0.2 0.25 0.28  0.32
7  LHUILLERY   ROUGE  0.5 0.50 0.36  0.35
8    MARTROU   ROUGE  0.3 0.35 0.36  0.35
9    NOVELLA   ROUGE  0.2 0.25 0.30  0.32
10    VERGNE   ROUGE  0.4 0.50 0.24  0.31
11    BARDET   ROUGE  0.2 0.30 0.28  0.32
12      LUNA   ROUGE  0.3 0.25 0.30  0.30
13    POADAE   ROUGE  0.6 0.30 0.26  0.19
14   REBIERE   ROUGE  0.4 0.35 0.24  0.35

On vérifie que seul, les deux bouchons bleus ont bien été enlevés.
On construit les variables nécessaires au tracé des nuages de points.
couleurB <- rep(sem1$bouchon, 4)
taillechant1 <- rep(c(10, 20, 50, 100), each = 14)
  freq1AR <- c(sem1$A_10, sem1$B_20, sem1$C_50, sem1$D_100)
taillechant2 <- rep(c(10, 20, 50, 100), each = 12)
  freq2AR <- c(sem1_1$A_10, sem1_1$B_20, sem1_1$C_50, sem1_1$D_100)
```

```
On pourrait construire un "data.frame" à partir de ces variables, ou directement.
controlAR1 <- data.frame(taillechant1, freq1AR)
  controlAR2 <- data.frame(taillechant2, freq2AR)
dim(controlAR1)
[1] 56  2
dim(controlAR2)
[1] 48  2
```

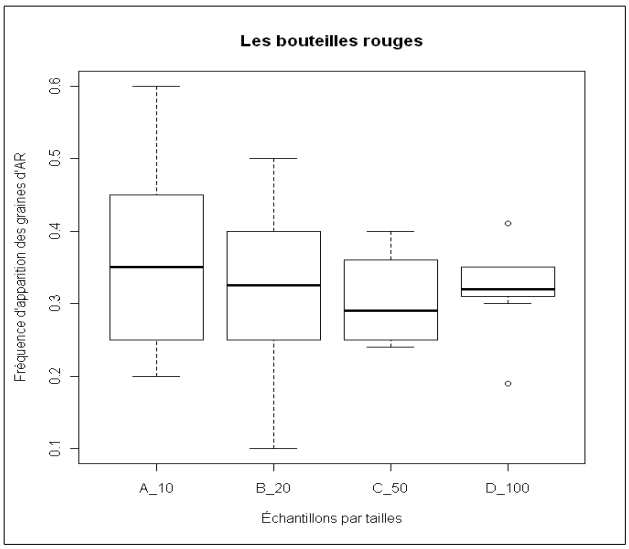
Décrire la structure des ces données.

BOÎTES DE DISPERSION OU BOÎTES À PATTES OU BOÎTES À MOUSTACHES

```
boxplot(sem1[, 3:6], xlab = "Échantillons par tailles", ylab = "Fréquence d'apparition des graines d'AR", main = "Toutes les bouteilles")
```



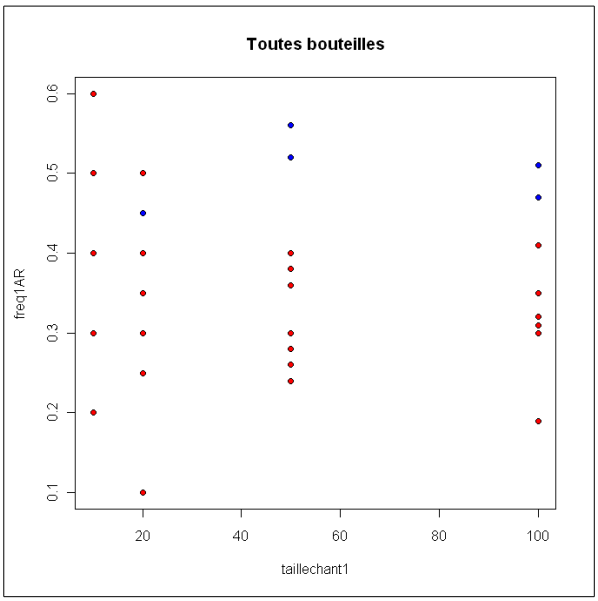

```
boxplot(sem1_1[, 3:6], xlab = "Échantillons par tailles", ylab = "Fréquence d'apparition des graines d'AR", main = "Les bouteilles rouges")
```



Quelles distributions se trouvent derrière les résumés par les boites de dispersion ?Comment faire une représentation moins résumée ?

On peut utiliser la fonction plot, avec une autre structure de données, mieux adaptée à ce type de traitement. Il faut restructurer les données en deux nouvelles variables, une variable taille d'échantillon et "en face" une variable fréquence d'apparition des graines d'AR. Avec **R**, cela se fait sans modifier les données initiales :

```
plot(taillechant1, freq1AR,
  main = "Toutes bouteilles",
  pch = 21, bg = c("blue", "red")[couleurB])
```

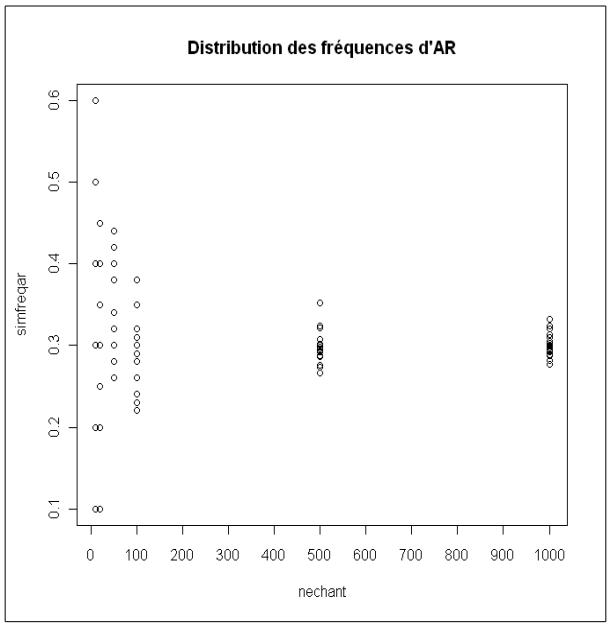


C2. PROLONGER L'EXPÉRIENCE PAR DES DONNÉES SIMULÉES

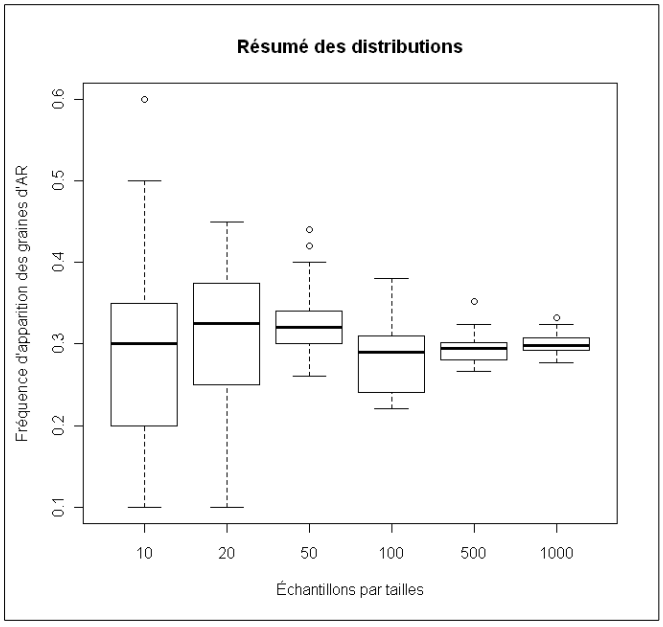
- Quand il devient difficile de réaliser les simulation à la main, on utilise la simulation à l'ordinateur.
- Nous allons simuler des séries de 20 (nombre de simulations) valeurs tirées d'une loi binomiale de paramètre p = 0,3 et n = 10, 20, 50, 100, 500 et 1000. Il faut donc créer deux nouvelles variables, **nechant** qui contiendra les tailles d'échantillons et **simfreqar** qui contiendra le résultat des séries de fréquences simulées. Dans un premier temps l'exploration des séries simulées se fait avec plot().

```
nechant <- rep(c(10, 20, 50, 100, 500, 1000), c(20, 20, 20, 20, 20, 20))
simfreqar <- c(rbinom(20, 10, .3)/10, rbinom(20, 20, .3)/20, rbinom(20, 50, .3)/50,
    rbinom(20, 100, .3)/100, rbinom(20, 500, .3)/500, rbinom(20, 1000, .3)/1000)
```

```
plot(nechant, simfreqar,
  xaxp = c(0, 1000, 10),
  main = "Distribution des fréquences d'AR")
```




```
plot(as.factor(nechant), simfreqar,
  xlab = "Échantillons par tailles",
  ylab = "Fréquence d'apparition des graines d'AR",
  main = "Résumé des distributions")
```



C3 ANALYSE EXPLORATOIRE DES RÉSULTATS D'UNE ENQUÊTE

- Cf. l'article sur la revue en ligne "Statistique et enseignement" à l'adresse :

http://publications-sfds.math.cnrs.fr/index.php/StatEns/article/view/95

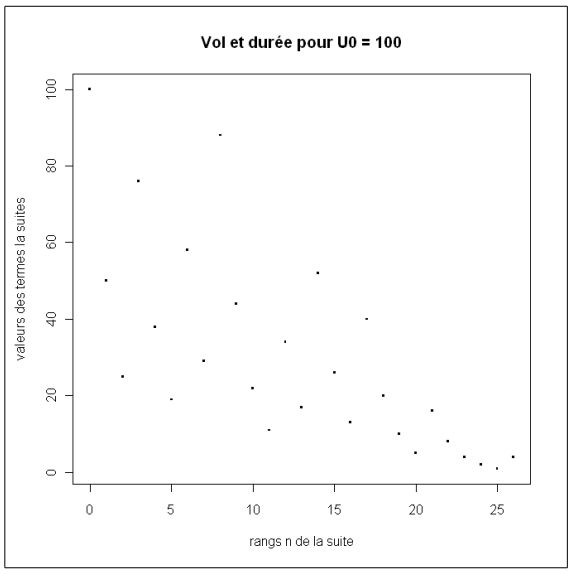
Ou le fichier : **R_ExploBudgetMenagesSimpl.pdf**

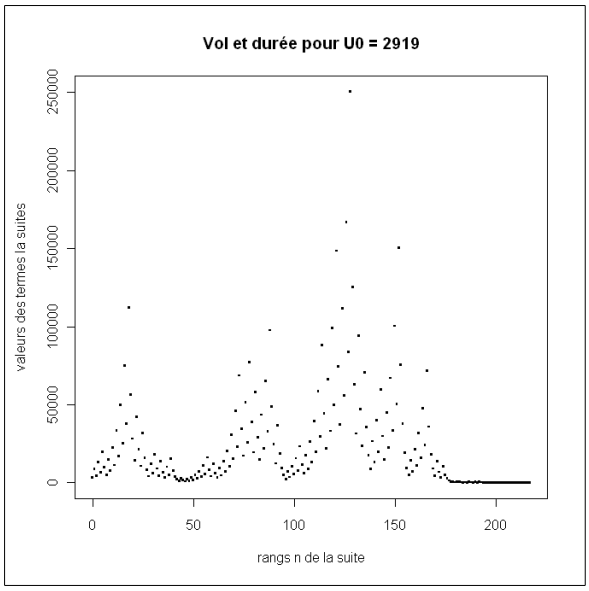
III - EXEMPLE(S) D'UTILISATION DE R EN ANALYSE

A. LA SUITE DE SYRACUSE SOUS UN ANGLE PARTICULIER

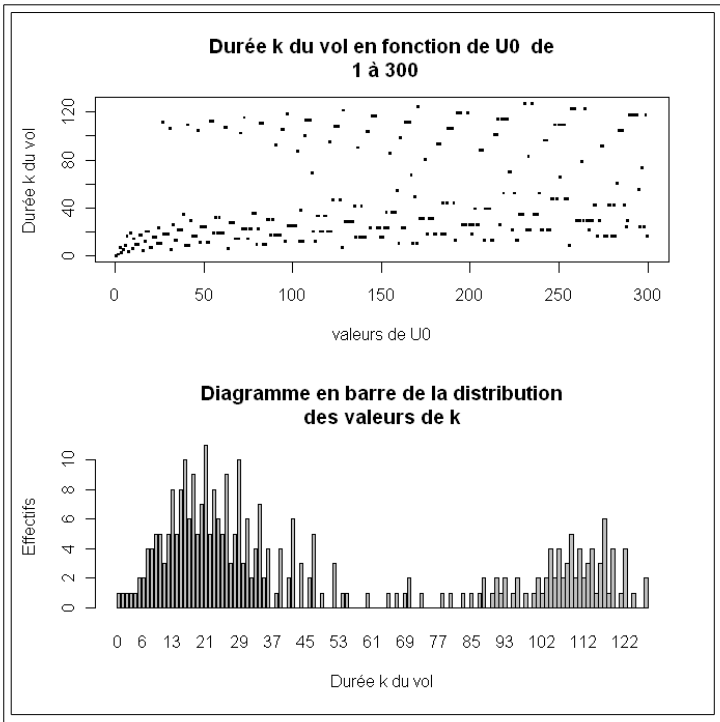
```
# Cette fonction calcule, en fonction de U0, le nombre
# d'itérations nécessaire pour atteindre la valeur 1
# La plus petite valeur d'indice tel que Un+1 < U0
#.ainsi que la valeur maximale de Un. Graphique de Un par n.
# Allez explorer le voisinage de U0 = 2919 **
syracn <- function(U0 = 100){
 if(U0 <= 0 | trunc(U0) != U0) {cat("U0 doit être entier naturel
différent de 0") ; stop()      }
 k <- 0 ; u <- U0
 while(u != 1){
   if(u %% 2 == 0) {u <- u / 2} else {u <- 3 * u + 1}
   k <- k + 1 }
 vectu <- vector(length = k + 2)
 names(vectu) <- 0:(k + 1)
 vectu[1] <- U0
 for(i in 1:(k + 1)){
   if(vectu[i] %% 2 == 0) {vectu[i + 1] <- vectu[i] / 2} else {
      vectu[i + 1] <- 3 * vectu[i] + 1                          }
                    }
tva <- min(which(vectu < U0)) - 2
 maxun <- max(vectu)
#******************** Affichage des résultats ******************
 cat("\nDurée k du vol =", k, "\n")
 cat("La plus petite valeur de n telle que Un+1 < U0, vaut :",
   tva, "\n")
 cat("La valeur maximale de Un vaut :", maxun, "\n")
 plot(as.numeric(names(vectu)), vectu,
   xlab = "rangs n de la suite",
   ylab = "Valeurs des termes de la suite",
      pch = ".", cex = 3, main = paste("Vol et durée pour U0 =",U0))
}
```

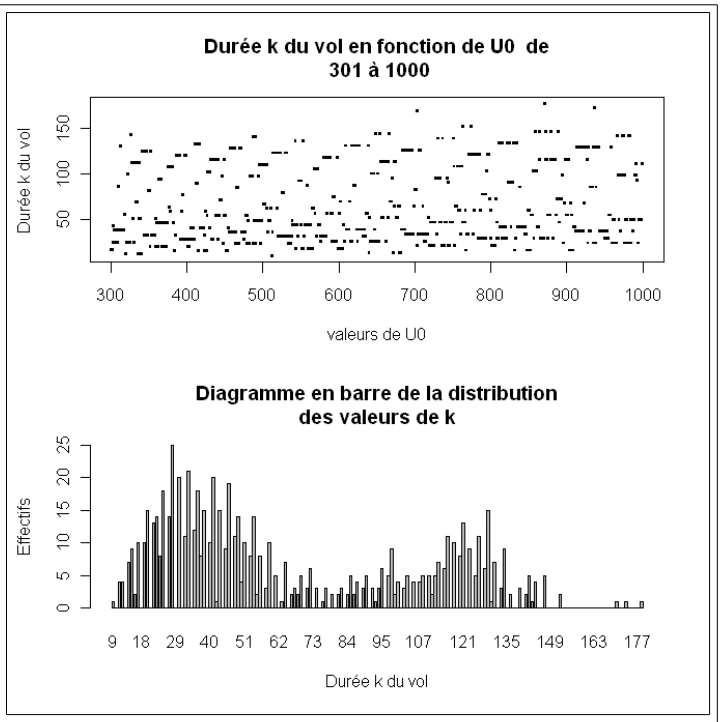
```
syracn()

Durée k du vol = 25
La plus petite valeur de
n telle que Un+1 < U0,
vaut : 0
La valeur maximale de Un
vaut : 100

syracn(2919)

Durée k du vol = 216
La plus petite valeur de
n telle que Un+1 < U0,
vaut : 41
La valeur maximale de Un
vaut : 250504
```





```
# Cette fonction calcule, pour un intervalle de valeurs de U0, les durées de vol, k.
# Graphique de k en fonction de U0. On constate que des valeurs de k se répètent.
# D'où l'idée de déterminer la distribution des valeurs de k, pour un intervalle
# de valeurs de U0 donné et d'en faire un diagramme en barre...
syr <- function(U0Inf = 1, U0Sup = 100){
 if(U0Inf <= 0 | U0Sup <= 0 | U0Sup <= U0Inf |
    trunc(U0Inf) != U0Inf | trunc(U0Sup) != U0Sup)                    {
    cat("Il faut des entiers naturels et 0 < U0Inf < U0Sup") ; stop()}
 taille <- (U0Sup - U0Inf + 1)
 vectn <- vector(length = taille)
 names(vectn) <- U0Inf:U0Sup
 j <- 1
 for(i in U0Inf:U0Sup){
   k <- 0 ; u <- i
   while(u != 1){
     if(u %% 2 == 0) {u <- u / 2} else {u <- 3 * u + 1}
     k <- k + 1 }
   vectn[j] <- k
   j <- j + 1          }
 minn <- min(vectn) ; maxn <- max(vectn)
 nbvaln <- maxn - minn + 1
 tableffec <- vector(length = nbvaln)
 names(tableffec) <- minn:maxn
 effec <- table(vectn)
 indicestablo <- as.numeric(names(effec)) - min(as.numeric(names(effec))) + 1
 tableffec[indicestablo] <- effec
# ************************ Affichage des résultats ************************
 par(mfrow = c(2, 1))
 plot(as.numeric(names(vectn)), vectn,
      xlab = "valeurs de U0", ylab = "Durée k du vol",
      pch = ".", cex = 3, main = paste("Durée k du vol en fonction de U0",
      " de\n", U0Inf, "à", U0Sup))
 barplot(tableffec, xlab = "Durée k du vol", ylab = "Effectifs",
    main = paste("Diagramme en barre de la distribution",
    "des valeurs de k", sep = "\n"))
}
```





IV - CONCLUSION

► Quels enjeux pour le lycée et le BTS ?

► Quels prolongements ?

► Quels moyens matériels (horaires, logiciels, formation des enseignants ...) pour une véritable mise en œuvre en classe ?

► Quelle place pour **R** ? (Je viens de découvrir qu'il faisait de la dérivation formelle !)

• • •

ANNEXES
A1 ÉNONCÉS DES EXERCICES DE PROBABILITÉ DU BAC S
2012MarsNelleCaledonieOblig. Un dé et deux urnes

EXERCICE 2 **4 points**

Commun à tous les candidats

On dispose de deux urnes et d'un dé cubique bien équilibré dont les faces sont numérotées de 1 à 6.
L'urne U_1 contient trois boules rouges et une boule noire.
L'urne U_2 contient trois boules rouges et deux boules noires.
Une partie se déroule de la façon suivante : le joueur lance le dé ; si le résultat est 1, il tire au hasard une boule dans l'urne U_1, sinon il tire au hasard une boule dans l'urne U_2.
On considère les évènements suivants :
A : « obtenir 1 en lançant le dé »
B : « obtenir une boule noire ».

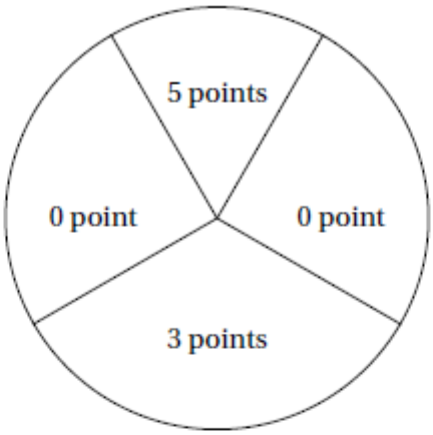
Baccalauréat S **A. P. M. E. P.**

1. a. Construire un arbre pondéré traduisant cettte expérience aléatoire.
   b. Montrer que la probabilité d'obtenir une boule noire est $\frac{3}{8}$.
   c. Sachant que l'on a tiré une boule noire, calculer la probabilité d'avoir obtenu 1 en lançant le dé.
2. On convient qu'une partie est gagnée lorsque la ·boule obtenue est noire. Une personne joue dix parties indépendantes en remettant, après chaque partie, la boule obtenue dans l'ume d'où elle provient. On note X la variable aléatoire égale au nombre de parties gagnées.
   a. Calculer la probabilité de gagner exactement trois parties. On donnera le résultat arrondi au millième.
   b. Calculer la probabilité de gagner au moins une partie. On donnera le résultat arrondi au millième.
   c. On donne le tableau suivant :

k	1	2	3	4	5	6	7	8	9	10
$P(X < k)$	0,009 1	0,063 7	0,211 0	0,446 7	0,694 3	0,872 5	0,961 6	0,992 2	0,999 0	0,999 9

Soit N un entier compris entre 1 et 10. On considère l'évènement : « la personne gagne au moins N parties ».
À partir de quelle valeur de N la probabilité de cet évènement est-elle inférieure à $\frac{1}{10}$?

2011-S-Avril-Pondichéry. Parties de jeu de fléchettes



On suppose que les lancers sont indépendants et que le joueur touche la cible à tous les coups.

1. Le joueur lance une fléchette.
   On note p_0 la probabilité d'obtenir 0 point.
   On note p_3 la probabilité d'obtenir 3 points.
   On note p_5 la probabilité d'obtenir 5 points.
   On a donc $p_0 + p_3 + p_5 = 1$. Sachant que $p_5 = \frac{1}{2}p_3$ et que $p_5 = \frac{1}{3}p_0$ déterminer les valeurs de p_0, p_3 et p_5.
2. Une partie de ce jeu consiste à lancer trois fléchettes au maximum. Le joueur gagne la partie s'il obtient un total (pour les 3 lancers) supérieur ou égal à 8 points. Si au bout de 2 lancers, il a un total supérieur ou égal à 8 points, il ne lance pas la troisième fléchette.
   On note G_2 l'évènement : « le joueur gagne la partie en 2 lancers ».
   On note G_3 l'évènement : « le joueur gagne la partie en 3 lancers ».
   On note P l'évènement : « le joueur perd la partie ».
   On note $p(A)$ la probabilité d'un évènement A.
   a. Montrer, en utilisant un arbre pondéré, que $p(G_2) = \frac{5}{36}$.
      On admettra dans la suite que $p(G_3) = \frac{7}{36}$
   b. En déduire $p(P)$.
3. Un joueur joue six parties avec les règles données à la question 2.
   Quelle est la probabilité qu'il gagne au moins une partie ?
4. Pour une partie, la mise est fixée à 2 €.
   Si le joueur gagne en deux lancers, il reçoit 5 €. S'il gagne en trois lancers, il reçoit 3 €. S'il perd, il ne reçoit rien.
   On note X la variable aléatoire correspondant au gain algébrique du joueur pour une partie. Les valeurs possibles pour X sont donc : -2, 1 et 3.
   a. Donner la loi de probabilité de X.
   b. Déterminer l'espérance mathématique de X. Le jeu est-il favorable au joueur ?

A2 UN INDEX DES FONCTIONS INTERNES LES PLUS UTILISÉES POUR LA SIMULATION ET LES STATISTIQUES

function() c() rep() vector() names() help() matrix() array() [] : ; ,	par(mfrow = c(2, 3)) par(new = T) dev.set() dev.new() cat() print() plot() barplot() boxplot() hist() abline() points()	table() which() rbinom() runif() sample() quantile() sum() rle() unique()