DU DISCRET AU CONTINU : AUTOUR DU THÉORÈME DE MOIVRE-LAPLACE − MISE EN ŒUVRE AVEC R

Hubert Raymondaud, LEGTA Louis Giraud 84200 CARPENTRAS
Anne Perrut, université Claude Bernard LYON-1

Le logiciel **R** est un logiciel libre de statistique et un langage de programmation. Il est très utilisé dans les universités scientifiques, aussi bien par les mathématiciens que par les biologistes, aussi bien pour l'enseignement que pour la recherche, car il est facile d'accès (gratuit) et car il permet de développer rapidement de nouveaux outils. Pour les mêmes raisons, ce logiciel tient une place de plus en plus importante dans les entreprises.

A − PRISE EN MAIN DE R ET PRÉSENTATION DES ACTIVITÉS

Pour commencer avec **R**, il faut l'installer (http://cran.R-project.org) et installer une interface agréable (http://www.rstudio.com/). Ensuite, il faut apprendre le langage de **R**. Il n'est pas forcément facile pour les élèves car l'aide est en anglais (mais quel bon exercice !) et elle s'adresse à des statisticiens. Bref, il faut déjà connaître les outils pour les utiliser. Par ailleurs, le langage est assez simple à apprendre, proche de celui de scilab ou matlab. Il ne nécessite pas de définition de variables et permet de programmer vite grâce à la programmation vectorielle : on peut appliquer directement une fonction aux vecteurs.

La fenêtre la plus importante de l'interface est la console. Vous pouvez l'utiliser comme une calculatrice pour commencer. Puis rapidement, pour sauvegarder votre travail, il faudra écrire les lignes de programme dans un fichier texte, un script **R**, que vous pourrez enregistrer. Il faut donc ouvrir un nouveau script et l'enregistrer avec un nom raisonnable, se terminant par l'extension **.R**. **Attention :** les noms de fichiers ne doivent pas comporter d'espace ni d'accent et doivent commencer par une lettre. Taper une opération simple, comme 2+3 dans votre script. Pour effectuer cette opération dans la console, il suffit de placer le curseur sur la ligne du script à effectuer et de taper Ctrl+Entrée ou de cliquer sur l'icône Run.

Avant tout, vous aurez besoin d'aide. Tapez dans le console : `help(rnorm)` ou `?rnorm` (aide de la fonction `rnorm`) . On peut aussi taper directement `normal distribution` par exemple, dans le cadre de recherche de la fenêtre d'aide.

Les objets de base sont les vecteurs et les matrices. Les différents types sont les entiers, les réels, les complexes, les booléens (TRUE, FALSE), les caractères. L'affectation se fait avec les signes `=` ou `<-`.

créer un vecteur	:	`x <- c(1, 5, 12, 3, 52, 43, 25)`
la suite (1,2,3,4,5,6,7)	:	`1:7`
concaténation de deux suites	:	`y <- c(x, 1:7)`
suites arithmétiques	:	`seq(from = 1, to = 10, by = 2)`
répétitions	:	`rep(1:5, 2)`

Testez maintenant les différentes commandes qui permettent de faire une étude statistique unidimensionnelle après avoir créé un échantillon artificiel de la loi binomiale B(100 ; 0,5)

```
ech <- rbinom(500, 100, 0.5)
summary(ech)
hist(ech)
barplot(table(ech))
boxplot(ech, range = 0)
sd(ech)
```

La fonction de base pour tracer des graphiques est `plot`. Soient x et y deux vecteurs de même longueur.

`plot(x)`	: trace le nuage des points $(i ; x_i)$
`plot(x, y)`	: trace le nuage des points $(x_i ; y_i)$
`plot(x, y, type = 'l')`	: trace une ligne brisée qui joint les points $(x_i ; y_i)$

La fonction plot possède de nombreuses options comme `main = "Mon titre"` (titre du graphique). On peut aussi jouer avec les options `pch, lty, col` pour changer les types de points, de lignes ou les couleurs. Tous les paramètres concernant les fenêtres graphiques sont décrits là : `?par`.

Pour tracer des fonctions, on utilise tout simplement curve. Par exemple,

```
curve(dnorm(x, 2, 3), from = -6, to = 10)
```

On peut évidemment mener tous types de calculs probabilistes. En ce qui concerne la loi normale,

`dnorm(x, mean = 0, sd = 1)`	: densité de la loi normale N(0,1)
`pnorm(q, mean = 0, sd = 1)`	: fonction de répartition de la loi normale
`qnorm(p, mean = 0, sd = 1)`	: fonction fractile (quantile, réciproque de la fonction de répartition)
`rnorm(n, mean = 0, sd = 1)`	: simulation de n nombres aléatoires suivant la loi normale

On peut bien évidemment changer la moyenne et l'écart-type, les valeurs 0 et 1 étant les valeurs par défaut. On peut retrouver toutes les fonctions pour les autres lois de probabilités, en changeant le `norm` par :

`binom`	: loi binomiale
`pois`	: loi de Poisson
`geom`	: loi géométrique
`unif`	: loi uniforme sur un intervalle
`exp`	: loi exponentielle

La fonction sample permet de simuler n'importe quelle loi discrète, à support fini.

On peut définir une **fonction** ou plus généralement une **procédure**. Voici l'exemple de la création d'une fonction qui permet de calculer le résumé numérique d'un échantillon, avec les quartiles du lycée.

```
resume <- function(echantillon, dig = 3){
   m <- mean(echantillon)
   s <- sd(echantillon)
   med <- median(echantillon)
   q <- quantile(echantillon, type = 1)
   resume <- c(m, s, q[2], med, q[4], q[1], q[5])
   names(resume) <- c("moyenne", "écarttype",
                      "Q1", "médiane", "Q3", "min", "max")
   print(resume, digit = dig)
}
```

La syntaxe pour les boucles est la suivante

```
for (i in 1:50) { ... }
while (cond) { ... }
```

et pour les tests logiques :

```
if (cond) {...} else {...}
```

Après avoir testé ces quelques commandes, nous pouvons écrire (copier) des programmes plus complexes. Nous vous proposons trois activités :

1° Exemple d'une séance de travaux pratiques d'algorithmique illustrant le théorème de Moivre-Laplace, en classe de terminale.

R est utilisé pour illustrer graphiquement la distribution d'une variable binomiale X, que l'on habille avec des rectangles (qu'on évite d'appeler histogramme, terme réservé à la représentation graphique d'une série "observée") puis pour illustrer la distribution de la variable centrée réduite correspondante et enfin pour observer ce qui se passe quand n augmente. Le passage à la variable centrée réduite entraine le passage à la densité pour la graduation des ordonnées, ce qui permet la superposition de la courbe de densité de la loi de Gauss centrée réduite.

On passe alors des lignes de commande **R** à une fonction **R**. **R** est utilisé sous **RStudio** qui est un environnement facilitant son utilisation (coloration syntaxique, complétion automatique, visualisation des objets créés en mémoire, rémanence des 40 derniers graphiques créés ...).

2° Illustration de la fluctuation d'échantillonnage et de la loi des grands nombres par la recherche d'un algorithme et sa mise en œuvre d'une simulation avec R.

On simule une série de n variables de Bernoulli dont on fait les sommes cumulées croissantes, puis les fréquences cumulées croissantes en divisant par les effectifs cumulés croissants. On obtient ainsi n simulations non indépendantes d'échantillons d'effectifs croissants de 1 à n. Les fréquences obtenues sont illustrées par un graphique en lignes brisées ou, mieux, en nuages de points, pour mettre l'accent sur l'aspect discret de la variable effectif. Cette activité permet d'illustrer l'efficacité de l'utilisation des couleurs avec **R**. Un prolongement possible consiste à simuler des échantillons indépendants et à illustrer la suite des distributions ainsi obtenues.

3° Exploration des intervalles de fluctuation de la seconde à la terminale.

L'activité consiste à analyser et à mettre en œuvre un algorithme permettant de calculer et d'illustrer graphiquement la probabilité binomiale "exacte" de séries d'intervalles de fluctuation asymptotiques de seconde et de terminale, en fonctions de la valeur de n. Les représentations graphiques permettent de visualiser la complexité de la relation liant la probabilité binomiale à la valeur de n. C'est la conséquence de la "discrétisation" autrement dit le fait de passer d'une fréquence à une valeur entière de la variable binomiale correspondante. L'activité se termine par la réalisation en **R**, des fameuses abaques, représentant les trois "types" d'IF (seconde, première et terminale) en fonction de p. Ces abaques électroniques permettent d'illustrer les problèmes d'approximation lorsque p se rapproche de 0 ou de 1.

Les pages suivantes présentent la feuille de route de ces trois activités, comprenant le code des commandes utilisées ainsi que quelques exemples de résultats obtenus. Le fichier archive **AtelierA4_CodesR.zip** contient le code de toutes les fonctions **R** présentées dans ce document.

Pour certaines activités nous avons proposé quelques réflexions et prolongements, montrant comment l'utilisation des TICE permet de présenter et d'illustrer des notions mathématiques délicates comme la loi des grands nombres et de surmonter des obstacles didactiques.

Références

Un manuel en français est disponible sur le site de **R**~: ***R*** *pour les débutants* de Emmanuel Paradis, à l'adresse http://cran.r-project.org/doc/contrib/Paradis-rdebuts_fr.pdf

Des milliers d'autres tutoriels, forums sont disponibles sur internet : notes de cours d'Anne Philippe

ou cours et exercices.

Un ouvrage pour bien commencer : *Initiation à la statistique avec* ***R*** de Frédéric Bertrand et Myriam Maumy-Bertrand, chez Dunod.

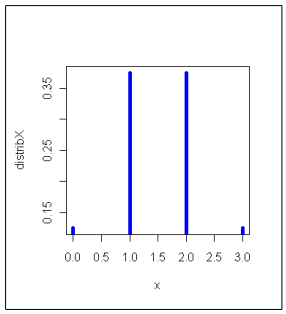
B – ACTIVITÉ 1 : DU DIAGRAMME EN BÂTONS À L'HISTOGRAMME POUR ILLUSTRER LE THÉORÈME DE MOIVRE-LAPLACE

I – Représenter graphiquement la distribution d'une loi binomiale

Un premier exemple d'**une variable aléatoire X** de loi binomiale de paramètres n = 3 et p = 0,5 :

On crée une liste contenant la distribution des (n + 1) probabilités ponctuelles de X :

```
(x <- 0:10) ; n <- 3 ; p <- 0.5
(distribX <- dbinom(0:n, n, p))
[1] 0.125 0.375 0.375 0.125
```



On réalise l'illustration graphique habituelle en diagrammes en barres :

```
plot(x, distribX, type = "h", col = "blue", lwd = 5)
```

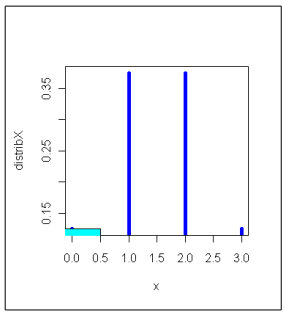
On va "habiller" avec des rectangles :

On prépare les bornes des intervalles encadrant les valeurs entières de X :

```
(bornesX <- seq(-.5, n + .5, 1))
[1] -0.5  0.5  1.5  2.5  3.5
```

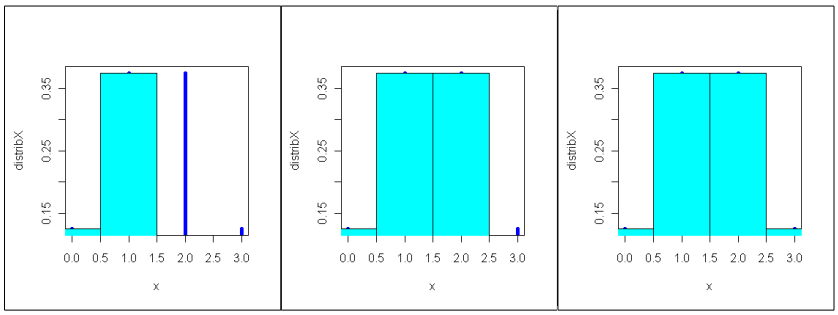
On superpose le premier rectangle :

```
polygon(c(-0.5, -0.5, 0.5, 0.5),
c(0, distribX[1], distribX[1], 0), col = "cyan")
```



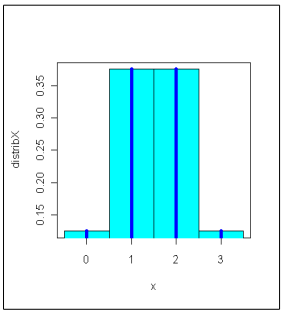
On superpose le deuxième rectangle puis le troisième et le quatrième :

```
polygon(c(0.5, 0.5, 1.5, 1.5), c(0, distribX[2], distribX[2], 0),
        col = "cyan")
polygon(c(1.5, 1.5, 2.5, 2.5), c(0, distribX[3], distribX[3], 0),
        col = "cyan")
polygon(c(2.5, 2.5, 3.5, 3.5), c(0, distribX[4], distribX[4], 0),
        col = "cyan")
```



Il y a un petit aménagement à faire sur l'axe des abscisses, qui doit commencer à -0,5 et finir à n + 0,5. Il faut reprendre la construction du graphique depuis la commande plot, mais avec le type "n". Placer ensuite les rectangles. Enfin on peut superposer les barres ...

Que vaut la surface totale des rectangles ?



II – Représentation graphique de la distribution de la variable centrée réduite (X – E(X)) / racine(V(X))

On crée une liste contenant les valeurs de la variable centrée réduite :

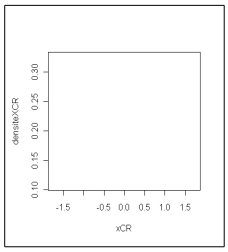
```
EspX <- n * p ; etypX <- sqrt(n * p * (1 - p))
(xCR <- (x - EspX) / etypX)
[1] -1.7320508 -0.5773503  0.5773503  1.7320508
```

On calcule les bornes centrées réduites des intervalles utilisés pour le graphique précédent.

```
(BornesXCR <- (bornesX - EspX) / etypX)
[1] -2.309401 -1.154701  0.000000  1.154701  2.309401
```

On calcule les densités (probabilité divisée par l'étendue des intervalles qui vaut 1 / etypX).

```
(densiteXCR <- distribX * etypX)
[1] 0.1082532 0.3247595 0.3247595 0.1082532
```

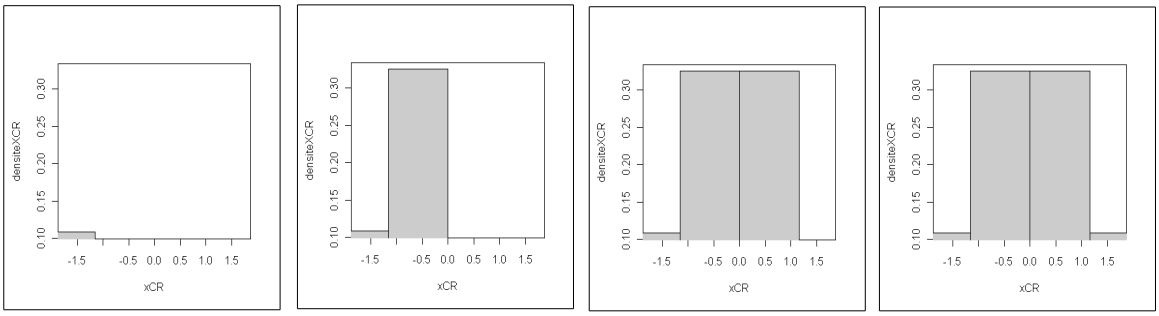


On ouvre une fenêtre graphique de tracé, vide

```
plot(xCR, densiteXCR, type = "n")
```

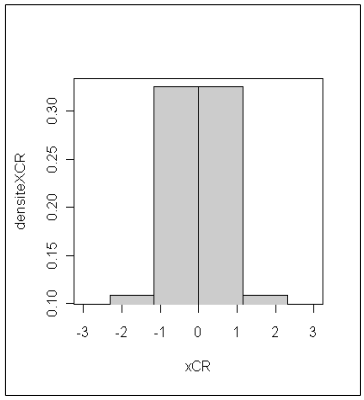
On trace les rectangles un par un :

```
polygon(c(BornesXCR[1], BornesXCR[1], BornesXCR[2], BornesXCR[2]),
        c(0, densiteXCR[1], densiteXCR[1], 0), col = "grey80")
polygon(c(BornesXCR[2], BornesXCR[2], BornesXCR[3], BornesXCR[3]),
        c(0, densiteXCR[2], densiteXCR[2], 0), col = "grey80")
polygon(c(BornesXCR[3], BornesXCR[3], BornesXCR[4], BornesXCR[4]),
        c(0, densiteXCR[3], densiteXCR[3], 0), col = "grey80")
polygon(c(BornesXCR[4], BornesXCR[4], BornesXCR[5], BornesXCR[5]),
        c(0, densiteXCR[4], densiteXCR[4], 0), col = "grey80")
```



Il y a un petit problème sur l'axe des abscisses qu'il faut s'empresser de résoudre en choisissant correctement les extrémités de l'axe.

```
minX <- floor(min(BornesXCR))
maxX <- ceiling(max(BornesXCR))
plot(xCR, densiteXCR, type = "n", xlim = c(minX, maxX))
# IL SUFFIT DE REFAIRE LES POLYGONES (sélectionner, exécu-
ter)
```



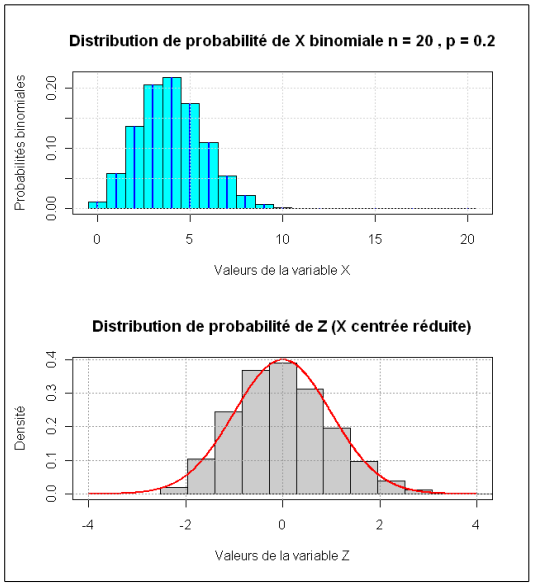
Un autre algorithme pour tracer les rectangles avec une boucle "pour".

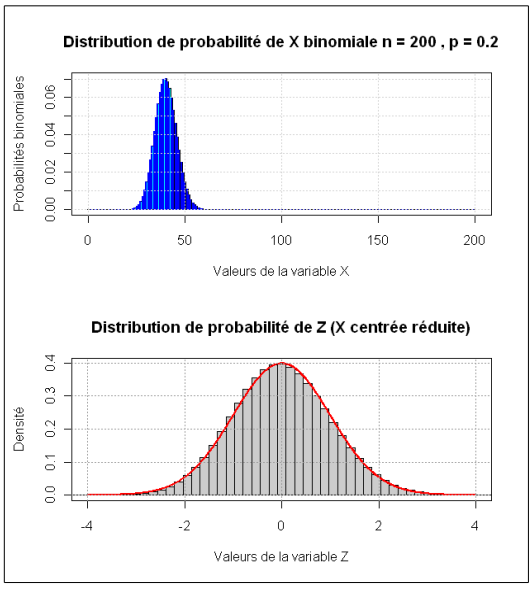
```
plot(xCR, densiteXCR, type = "n", xlim = c(minX, maxX))
for (k in 1:(n + 1)) {
    polygon(c(BornesXCR[k], BornesXCR[k], BornesXCR[k + 1], BornesXCR[k + 1]),
            c(0, densiteXCR[k], densiteXCR[k], 0), col= "grey80")
}
```

III – Pour pouvoir facilement changer les valeurs de n et p, on fabrique une fonction R

On rassemble les lignes de commande précédentes pour en faire une fonction **R**. Il faut juste leur donner la bonne structure, à l'aide de la fonction **`function(...){...}`**. On ajoute quelques légendes ...

```
MoivreLap0 <- function(n = 10, p = .5){
   x <- 0:n
   distribX <- dbinom(x, n, p)
   EspX <- n * p ; etypX <- sqrt(n * p * (1 - p))
   densiteXCR <- distribX * etypX
   subdivX <- seq(-.5, n + .5, 1)
   xCR <- (x - EspX) / etypX
   subdivXCR <- (subdivX - EspX) / etypX
   # "HISTOGRAMME" DE LA DISTRIBUTION BINOMIALE DE LA VARIABLE X
   par(mfrow = c(2, 1)) # Partage la fenêtre graphique en 2 lignes et 1 colonne.
   plot(x, distribX, type = "n", xlim = c(-.5, n + .5),
         xlab = "Valeurs de la variable X",
         ylab = "Probabilités binomiales",
         main = paste("Distribution de probabilité de X binomiale n =", n, ", p =", p))
   for (k in 1:(n + 1)) {
      polygon(c(subdivX[k], subdivX[k], subdivX[k + 1], subdivX[k + 1]),
              c(0, distribX[k], distribX[k], 0), col = "cyan")
   }
   points(x, distribX, type = "h", col = "blue", lwd = 2)
   grid()
   # "HISTOGRAMME" DE LA VARIABLE CENTRÉE RÉDUITE Z = (X - E(X)) / écartype(X)
   plot(xCR, densiteXCR, type = "n", xlim = c(-4, 4),
         xlab = "Valeurs de la variable Z (X centrée réduite)",
         ylab = "Densité",
         main = "Distribution de probabilité de Z",
   for (k in 1:(n + 1)) {
      polygon(c(subdivXCR[k], subdivXCR[k], subdivXCR[k + 1], subdivXCR[k + 1]),
              c(0, densiteXCR[k], densiteXCR[k], 0), col = "grey80")
   }
   xGauss <- seq(-4, 4, length = 1000) ; yGauss <- dnorm(xGauss)
   lines(xGauss, yGauss, col = "red", lwd = 2)
   grid()
}
```





C – ACTIVITÉ 2 : DE LA FLUCTUATION D'ÉCHANTILLONNAGE À LA LOI DES GRANDS NOMBRES

I – Simuler et représenter graphiquement les fluctuations d'échantillonnage d'une fréquence

- **Simulation de nbn=18 variables de Bernoulli de paramètre p,** les valeurs sont stockées dans le “vecteur” px.

```
p <- .5 ; nbn <- 18
px <- rbinom(nbn, 1, p)
[1] 1 1 1 0 0 1 1 0 1 0 0 0 0 1 1 0 0 0
```

- **Calcul des 18 (nbn) fréquences cumulées** correspondantes dont les valeurs sont stockées dans le vecteur repartfreqsim. On fait commencer le calcul des fréquences à partir de l'échantillon de taille 5.

```
(repartfreqsim <- cumsum(px)[5:nbn] / (5:nbn))
[1] 0.6000000 0.6666667 0.7142857 0.6250000 0.6666667 0.6000000 0.5454545
[8] 0.5000000 0.4615385 0.5000000 0.5333333 0.5000000 0.4705882 0.4444444
```

- **Illustration graphique en ligne brisée** (type = "l") de la première série de 18 simulations.

```
plot((5:nbn), repartfreqsim, type = "l", ylim = c(.3, .7),
      main = paste("Simulations d'une variable fréquence",
      "\nIFasympt. de seconce (en vert), IFasympt. de term (en noir)"),
      ylab = "Fréquences* simulées",
      xlab = "Nombre de tirages (Effectif de l'échantillon)")
```

- **Simulation de la deuxième série** de 18 simulations.

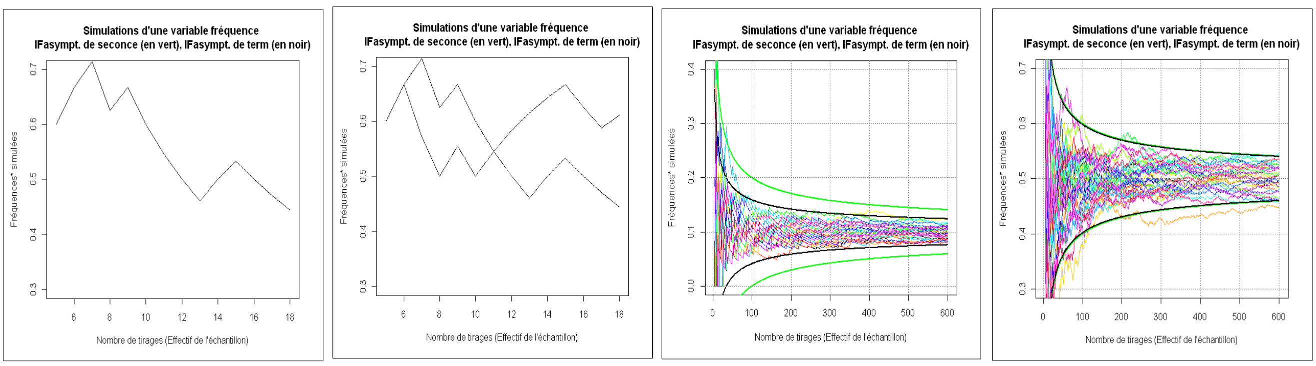
```
(rfs <- cumsum(rbinom(nbn, 1, p))[5:nbn] / (5:nbn))
 [1] 0.6000000 0.6666667 0.5714286 0.5000000 0.5555556 0.5000000 0.5454545
 [8] 0.5833333 0.6153846 0.6428571 0.6666667 0.6250000 0.5882353 0.6111111
```

- Illustration graphique, une deuxième ligne brisée se superpose à la première :

```
   lines(5:nbn, rfs)
```

- **On continue à simuler des séries d'échantillons** de tailles croissantes de 1 à 18, à l'aide d'une boucle for. Pour différencier les 18 séries, on fait intervenir la gestion de la couleur avec **R** et sa fonction rainbow(...)

```
   for (i in 1:nbsim) {
      (rfs <- cumsum(rbinom(nbn, 1, p))[5:nbn] / (5:nbn))
      lines(5:nbn, rfs), col = rainbow(nbsim)[i])
   }
```



- Les principaux inconvénients de cette simulation sont que les échantillons de différentes tailles ne sont pas indépendants et qu'elle ne permet pas d'illustrer la suite des différentes distributions associées aux tailles d'échantillons.

II − Une simulation alternative illustrant la suite des distributions

Cette fois ci les échantillons de différentes tailles sont indépendants. Nous avons fait deux types d'illustrations graphiques mettant en évidence quelques aspects des distributions des échantillons de différentes tailles.

En fait cette simulation numérique est le prolongement d'une activité de simulation de contrôle de qualité par attribut, réalisée avec des bouteilles contenant des boules de couleurs simulant des lots de marchandise, à l'aide d'un protocole et des grilles de saisie représentant une situation concrète. Cette activité est détaillée sur le site de l'IREM de Lyon http://math.univ-lyon1.fr/irem/spip.php?article506.

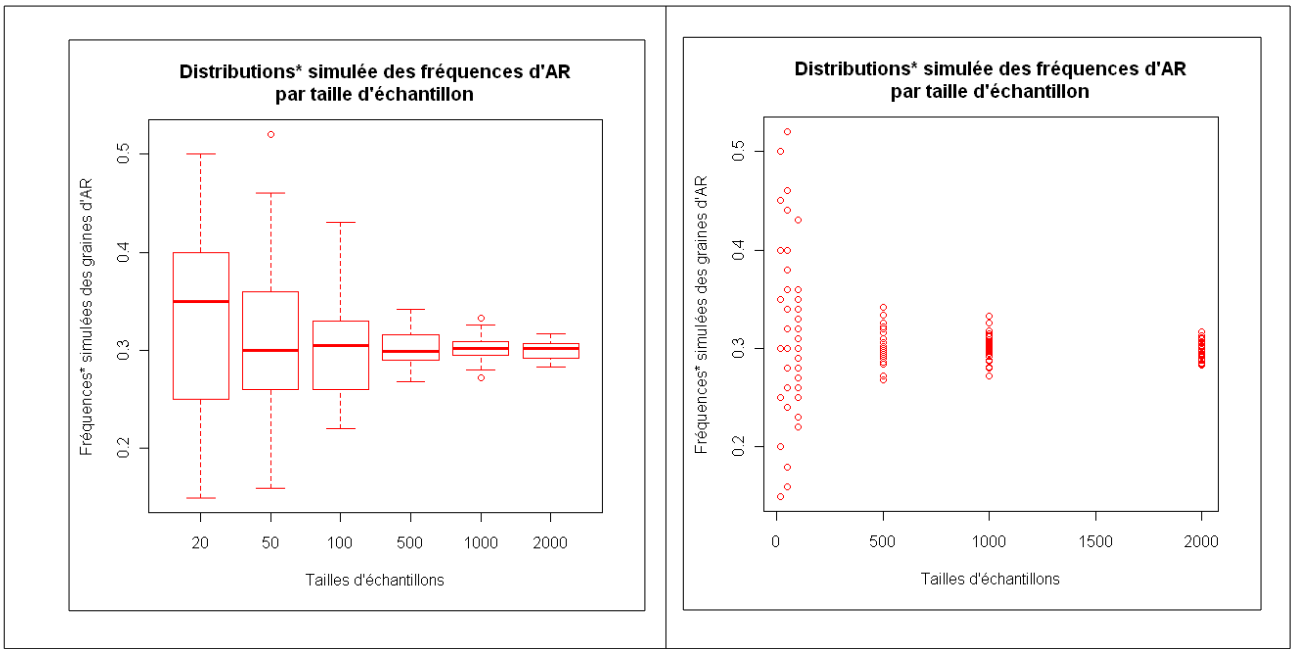
Lorsque la taille de l'échantillon augmente (n = 500, n = 1000, n = 2000) on ne peut plus faire d'expérience réelle, on la remplace par une simulation numérique, comme cela se fait dans beaucoup d'autres domaines scientifiques, techniques, biologiques.

On va créer des données simulées que l'on stockera dans 2 variables, taille d'échantillons `(taillEch)`, et fréquence simulée de **graines d'AR** `(FreqBino)`, **représentées par des boules rouges, représentant le succès.** Les tailles d'échantillons simulés seront de 20, 50, 100, 500, 1000, 2000. On fera 30 répétitions de chaque simulation (comme si c'était les 30 élèves d'une classe qui réalisaient les simulations).

```
# Les paramètres des lots (Niveau de Qualité Acceptable)
# correspondant aux bouteilles à bouchons rouges, contenant 30 % de boules rouges)
p <- .3 ; nbsim = 30

# Création des tailles d'échantillons
taillEch <- rep(c(20, 50, 100, 500, 1000, 2000), each = nbsim)

# Création d'une fonction pour la Simulation de séries "binomiales".
EchBino <- function(taille){rbinom(nbsim, taille, p)}

# Simulation numérique de 6 séries de 30 (nbsim) échantillons de tailles
# n = 20, 50, 100, 500, 1000, 2000. La variable X est le nombre de
# graine d'AR (boules rouges) dans l'échantillon
EffBino <- c(EchBino(20), EchBino(50), EchBino(100), EchBino(500), EchBino(1000),
EchBino(2000))

# On fait le graphique (en boite)
plot(as.factor(taillEch), EffBino)

# On passe donc à la variable X/n fréquence de graines d'AR
# dans l'échantillon de taille n
FreqBino <- c(EchBino(20) / 20, EchBino(50) / 50, EchBino(100) / 100,
   EchBino(500) / 500, EchBino(1000) / 1000, EchBino(2000) / 2000)

# Cette fois-ci le graphique (en boite) est pertinent
plot(as.factor(taillEch), FreqBino,
   main = "Distributions* simulée des fréquences d'AR\npar taille d'échantillon",
   xlab = "Tailles d'échantillons",
   ylab = "Fréquences* simulées des graines d'AR",
   border = "red")
```

```
# On poursuit par le nuage de points
plot(taillEch, FreqBino,
   col = "red",
   main = "Distributions* simulée des fréquences d'AR\npar taille d'échantillon",
   xlab = "Tailles d'échantillons",
   ylab = "Fréquences* simulées des graines d'AR")
```

Ce qui donne les graphiques suivants :



Les diagrammes en boite fournissent un résumé graphique permettant de comparer facilement les distributions des échantillons de différentes tailles. Par contre l'échelle horizontale n'est pas numérique, les tailles d'échantillons étant prises comme modalités d'un facteur (variable catégorielle).

Dans le graphique en nuage de points les deux axes sont numériques. À la précision du graphique près, tous les points sont représentés, donnant une idée plus précise de la distribution des fréquences en fonction de la taille des échantillons. On peut ainsi donner du sens à la définition mathématique de la loi des grands nombres.

Cette présentation a d'ailleurs été reprise par Annette Corpart et Nelly Lassalle de la CII statistique et probabilité, de l'IREM de Clermont Ferrand (http://www.irem.univ-bpclermont.fr/spip.php?article366).

D – ACTIVITÉ 3 : LES INTERVALES DE FLUCTUATION DE LA SECONDE À LA TERMINALE ; DES PROBABILITÉS BINOMIALES AUX ABAQUES

I – Probabilité d'un intervalle de fluctuation IF1 d'une variable binomiale (programme de première)

Le langage **R** permet de mettre directement en œuvre la méthode de calcul, sans passer par les boucles tant que qui constituent un obstacle supplémentaire :

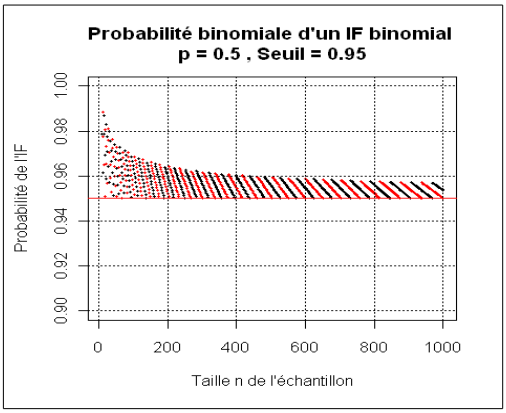
L'intervalle de fluctuation bilatéral au seuil minimal de probabilité s d'une variable binomiale X de paramètres n et p, défini dans le programme de première, est l'intervalle [a ; b] où

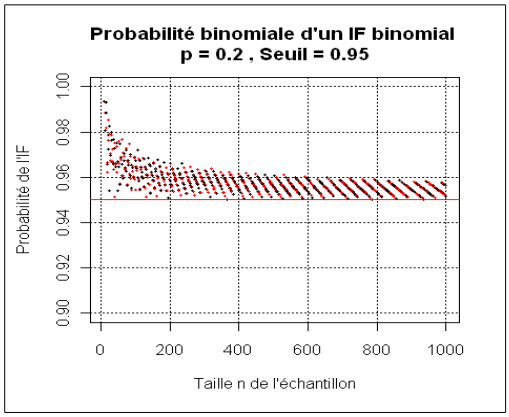
a est la plus petite valeur de X telle que P(X <= a) > (1 – s) / 2)
b est la plus petite valeur de X telle que P(X <= b) >= (1 – (1 – s) / 2)

[a/n ; b/n] est l'intervalle de fluctuation de la variable fréquence (de succès) X/n.

```
# Calcul de l'IF1 et de sa probabilité "exacte", pif.
PIF_Bino <- function(n, p, s){
   repartX <- pbinom(0:n, n, p)
   rang_a <- min(which(repartX > (1 - s) / 2))
   a <- rang_a - 1
   rang_b <- min(which(repartX >= (1 + s) / 2))
   b <- rang_b - 1
   pif <- sum(dbinom(a:b, n, p))
   return(pif)
}
```

```
# Représentation graphique de pif en fonction de la taille n de l'échantillon.
GrafPifBino <- function(N = 5:1000, p = .5, s = .95){
   SeriePif <- NULL
   for (i in N) {SeriePif <- c(SeriePif, PIF_Bino(i, p, s))}
   plot(N, SeriePif, pch = 19, col = c("black", "red"), cex = .4,
           ylim = c(.9, 1),
           main = paste("Probabilité binomiale d'un IF binomial",
                        "\np =", p, ", Seuil =", s),
           xlab = "Taille n de l'échantillon",
           ylab = "Probabilité de l'IF")
      abline(h = p, col = "red")
      grid(col = "black")
}
```

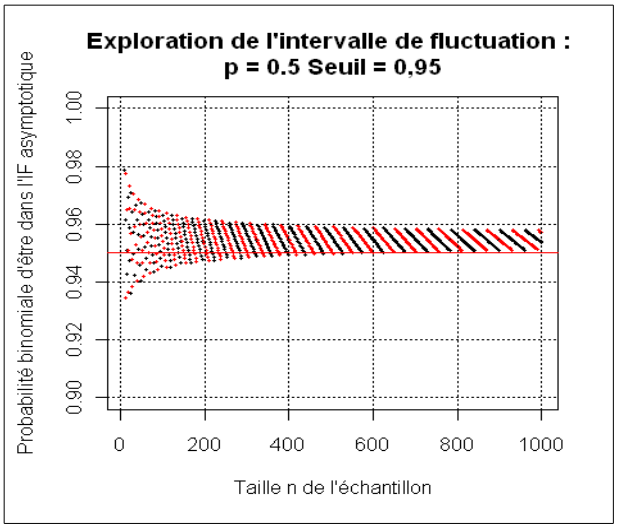


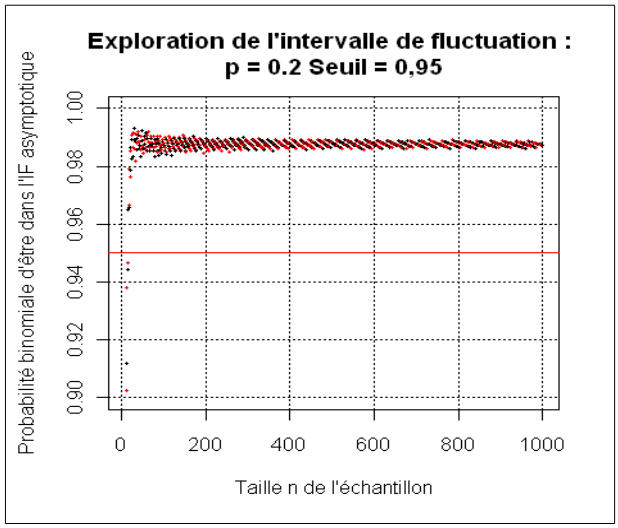


II – Probabilité d'un intervalle de fluctuation asymptotique IFasymp2 d'une fréquence (programme de seconde)

Il s'agit de représenter graphiquement la probabilité binomiale de l'intervalle de fluctuation asymptotique du programme de seconde, d'une variable fréquence. Le principe consiste à passer des valeurs de la variable fréquence aux valeurs correspondantes de la variable binomiale X, et à calculer la probabilité binomiale de l'intervalle ainsi obtenu. C'est cette “discrétisation” qui explique la forme particulière du nuage de points obtenu.

```
# Probabilité binomiale de l'IF asymptotique Gaussien de seconde : p +- 1/racine(n)
# en fonction de n
pIFasy1_1 <- function(N = 5:1000, p = .5){
   P <- function(n, p) {
      binf <- max(floor(n * p - sqrt(n)), 0)
      bsup <- min(floor(n * p + sqrt(n)), n)
      sum(dbinom((binf + 1):bsup, n, p))
   }
   y <- NULL
   for (i in N) {y <- c(y, P(i, p))}
   #-----Affichage des graphiques-----
   plot(N, y, pch = 19, cex = .4, col = c("black", "red"),
         ylim = c(.9, 1),
         xlab = "Taille n de l'échantillon",
         ylab = "Probabilité binomiale d'être dans l'IF asymptotique",
         main = paste("Exploration de l'intervalle de fluctuation :",
                      "\np =", p, ", Seuil = 0,95"))
      abline(h = .95, col = "red")
      grid(col = "black")
}
```





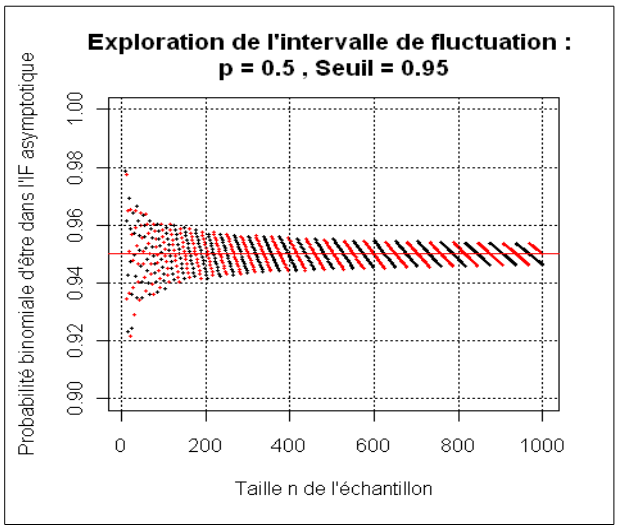
Pour p = 0,5 on observera qu'il existe une valeur n_0 de n telle que la probabilité de l'IFasymp2 soit toujours supérieure à 0,95. La détermination de cette valeur, qui se situe vers 600, peut être l'occasion de la recherche d'un algorithme ad-hoc. Il est aussi intéressant d'observer que cette valeur seuil est différente pour p = 0,2 et que dans ce cas la probabilité binomiale de l' IFasymp2 se stabilise autour de 0,99, pour un seuil nominal de 0,95.

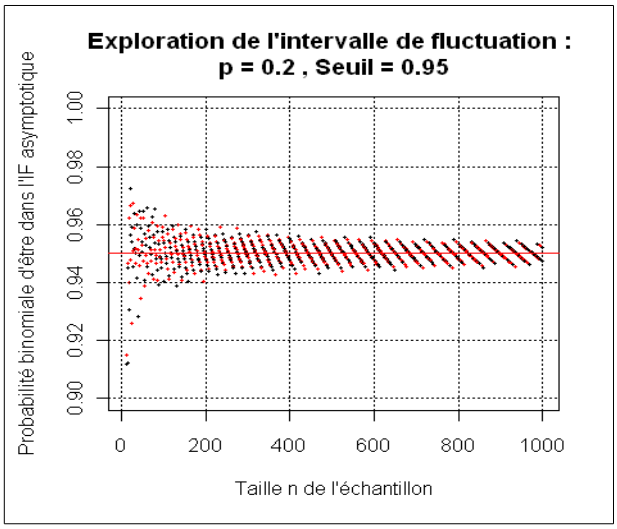
L'alternance des couleurs noire et rouge correspond à la parité de n.

III – Probabilité d'un intervalle de fluctuation asymptotique IFasympT d'une fréquence (programme de terminale)

Il s'agit de représenter graphiquement la probabilité binomiale de l'intervalle de fluctuation asymptotique du programme de terminale, d'une variable fréquence. Le principe est le même que pour l'IF1.

```
# Probabilité binomiale de l'IF asymptotique Gaussien de terminale
# en fonction de n, (p +- 1,96*racine(p(1-p)/n))
pIFasy2_1 <- function(N = 5:1000, p = .5, s = .95){
   P <- function(n, p, s) {
      g <- qnorm(1 - (1 - s) / 2)
      binf <- max(floor(n * p - g * sqrt(p * (1 - p) * n)), 0)
      bsup <- min(floor(n * p + g * sqrt(p * (1 - p) * n)), n)
      sum(dbinom((binf + 1):bsup, n, p))
   }
   y <- NULL
   for (i in N) {y <- c(y, P(i, p, s))}
   #-----Affichage des graphiques-----
   plot(N, y, pch = 19, cex = .4, col = c("black", "red"),
         ylim = c(.9, 1),
         xlab = "Taille n de l'échantillon",
         ylab = "Probabilité binomiale d'être dans l'IF asymptotique",
         main = paste("Exploration de l'intervalle de fluctuation :",
                      "\np =", p, ", Seuil =", s),
      abline(h = s, col = "red")
      grid(col = "black")
}
```





L'importante particularité que permet d'observer ces graphiques c'est que les valeurs de la probabilité semblent osciller autour de 0,95, même pour des valeurs de n allant jusqu'à 10^6 que nous avons essayées.

Il semble donc qu'il ne sera pas possible de trouver une valeur minimale n_0 de n telle que la probabilité binomiale de l'IFasympT soit toujours supérieur à 0,95.

IV – Construction d'abaques d'intervalles de fluctuation binomiaux d'une variable fréquence

On cherche à déterminer les intervalles de confiance d'une proportion. Cet algorithme tracera les abaques des intervalles de fluctuation suivants :

- asymptotique du programme de seconde, noté **IFasymp2 (tiretés noirs sur le graphique).** Cet intervalle n'apparaîtra que pour le seuil de 95 %.
- "exact" du programme de première, noté **IF1 (traits pleins verts sur le graphique).**
- asymptotique du programme de terminale, noté **IfasympT (tiretés points rouges sur le graphique).**

On a les paramètres suivants en entrée :

- n : la taille de l'échantillon (le nombre d'épreuves indépendantes)
- seuil : le seuil de probabilité de l'I.F.
- nbvalp : la subdivision de valeurs de p, proportion dans la population ou probabilité de succès
- NumIF : série des rang des valeurs de p pour lesquelles l'I.F. Est représenté graphiquement.

Détermination d'un intervalle de confiance d'une proportion à l'aide des abaques

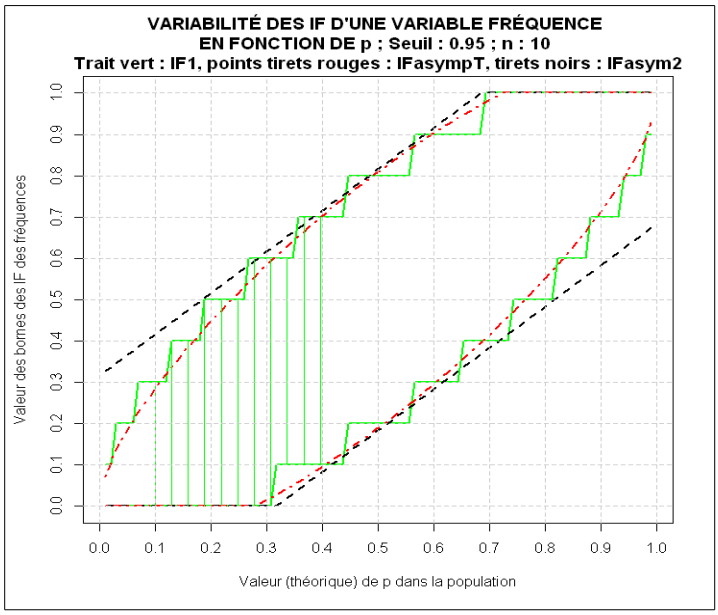
Il suffit de lire l'abaque en sens "inverse" : pour une valeur de la fréquence f lue sur l'axe des ordonnées, on détermine les deux valeurs de p sur l'axe des abscisses...

L'utilisation de l'abaque de l'**IF1** nécessite des précautions particulières .... On expliquera pourquoi...

Il existe un algorithme spécifique qui permet de déterminer l'intervalle de confiance "exact" d'une proportion, c'est la méthode de CLOPPER-PEARSON qui peut donner lieu à bel exercice d'algorithmique, utilisant simplement des répartitions binomiales.

```
# Fonction utilitaire IF_Bino calcule l'intervalle de fluctuation d'une variable fré-
quence
IF_Bino <- function(n, p, s){
   repartX <- pbinom(0:n, n, p)
   if (max(repartX) <= (1 - s) / 2) {
      a <- 0
   } else {
      rang_a <- min(which(repartX > (1 - s) / 2))
      a <- rang_a - 1
   }
   if (min(repartX) > (1 + s) / 2) {
       b <- n
   } else {
      rang_b <- min(which(repartX >= (1 + s) / 2))
      b <- rang_b - 1
   }
   IF <- c(a, b)
   return(IF)
}

#----------------------------
# Cette fonction utilise IF_Bino
# elle trace une représentation graphique de quelques intervalles de fluctuation
abak_IF <- function(n = 10, seuil = .95, nbvalp = 100, NumIF = seq(10, 40, 3)){
   subdivp <- seq(.01, .99, length = nbvalp)
   bInfIF <- NULL ; bSupIF <- NULL
```

```
   bInfIFasyT <- NULL ; bSupIFasyT <- NULL
   if (seuil == .95) {bInfIFasy2 <- NULL ; bSupIFasy2 <- NULL}
   g <- qnorm(1 - (1 - seuil) / 2)
   for (p in subdivp) {
      bInfIF <- c(bInfIF, IF_Bino(n, p, seuil)[1] / n)
      bSupIF <- c(bSupIF, IF_Bino(n, p, seuil)[2] / n)
      if (seuil == .95){
         bInfIFasy2 <- c(bInfIFasy2, max(p - 1 / sqrt(n), 0))
         bSupIFasy2 <- c(bSupIFasy2, min(p + 1 / sqrt(n), 1))
      }
      bInfIFasyT <- c(bInfIFasyT, max(p - g * sqrt(p * (1 - p) / n), 0))
      bSupIFasyT <- c(bSupIFasyT, min(p + g * sqrt(p * (1 - p) / n), 1))
   }
   # GRAPHIQUES
      if (seuil == .95) {
         minY <- min(c(bInfIF, bInfIFasy2, bInfIFasyT))
         maxY <- max(c(bSupIF, bSupIFasy2, bSupIFasyT))
      } else {
         minY <- min(c(bInfIF, bInfIFasyT))
         maxY <- max(c(bSupIF, bSupIFasyT))
      }
   plot(subdivp, bInfIF, type = "l", col = "green", ylim = c(minY, maxY), lwd = 2,
         main = paste("VARIABILITÉ DES IF D'UNE VARIABLE FRÉQUENCE",
         "\nEN FONCTION DE p ; Seuil :", seuil, "; n :", n,
         "\nTrait vert : IF1, points tirets rouges : IFasympT, tirets noirs : IFasym2"),
         xlab = "Valeur (théorique) de p dans la population",
         ylab = "Valeur des bornes des IF des fréquences",
         xaxp = c(0, 1, 10), yaxp = c(0, 1, 10))
      lines(subdivp, bSupIF, col = "green", lwd = 2)
      if (seuil == .95) {
         lines(subdivp, bInfIFasy2, lty = 2, lwd = 2)
         lines(subdivp, bSupIFasy2, lty = 2, lwd = 2)
      }
      lines(subdivp, bInfIFasyT, lty = 4, col = "red", lwd = 2)
      lines(subdivp, bSupIFasyT, lty = 4, col = "red", lwd = 2)
      for (i in NumIF) {
         lines(c(subdivp[i], subdivp[i]), c(bInfIF[i], bSupIF[i]), col = "green")
      }
      abline(h = seq(0, 1, .1), v = seq(0, 1, .1), col = "grey80", lty = 2)
}
```



Les traits verticaux verts représentent l'étendue de quelques intervalles de fluctuation “exacts” IF1.

Il est intéressant d'observer et de comparer la position relative des différents intervalles de fluctuation en fonction de p. Lorsque p s'approche de 0 ou de 1, les approximations asymptotiques perdent de la précision, elles s'éloignent des valeurs de l'IF1.

V – Prolongement possible : L'intervalle de confiance “exact” d'une proportion p, par la méthode de CLOPPER-PEARSON

Les données consistent en t succès à la fin de n épreuves de Bernoulli indépendantes, de paramètre p. X est la variable aléatoire prenant pour valeur le nombre de succès à la fin des n épreuves. On supposera qu'elle suit une loi binomiale de paramètres n et p.

L'intervalle de confiance “exact” de p au niveau de confiance s (par exemple 0,95),
est l'intervalle $[p_{inf} ; p_{sup}]$ où :

p_{inf} est tel que $p_{inf} = 0$ si $t = 0$ et tel que $P_{p = p_{inf}}(X \geq t) = (1 - s) / 2)$ si $0 < t \leq n$.

p_{sup} est tel que $p_{sup} = 1$ si $t = n$ et tel que $P_{p = p_{sup}}(X \leq t) = (1 - s) / 2)$ si $0 \leq t < n$.

Un excellent exemple à mettre en œuvre avec un algorithme et un programme. À paraître bientôt, programmé en **R**, sur le site de mathémaTICE (http://revue.sesamath.net/).

E – CONCLUSION

Ces trois exemples nous montrent comment les activités algorithmiques et les activités mathématiques s'enrichissent mutuellement. L'algorithmique permet de mettre en œuvre, d'illustrer, de donner du sens à des notions délicates comme le théorème de MOIVRE-LAPLACE et le passage du discret au continu pour les variables aléatoires, les suites de distributions dans la loi des grands nombres, la probabilité binomiale de certains intervalles de fluctuation asymptotiques d'un variable fréquence, la construction des abaques pour déterminer des intervalles de fluctuation d'une variable fréquence et des intervalles de confiances d'une proportion dans une population. Cet enrichissement passe aussi par un outil logiciel capable de faire sans technicité excessive, des représentations graphiques de qualité. **R** nous permet d'obtenir cette qualité.