TP 2 : Simulation de variables aléatoires, méthode de Monte-Carlo

1 Grands principes

1.1 Génération de nombres pseudo-aléatoires

Vouloir utiliser un ordinateur pour obtenir des nombres aléatoires apparaît paradoxal, sinon impossible : par définition, un nombre aléatoire n'est pas prévisible, tandis que l'ordinateur ne peut appliquer qu'une formule prédéfinie (un *algorithme*).

L'intérêt est pourtant grand, car bien des applications utilisent des nombres aléatoires : sécurité informatique (génération automatique d'identifiants, de clés secrètes), méthodes d'optimisation dans des espaces de grande dimension (algorithmes génétiques), simulations numériques de systèmes complexes (physique, ingénierie, finance, assurance, météo...), jeux vidéos (paysages aléatoires, intelligence artificielle,...),...

D'où l'importance d'algorithmes fournissant des suites de nombres ayant, de façon approximative, certaines propriétés d'une suite de variables aléatoires indépendantes et de même loi. On parle de nombres **pseudo-aléatoires**.

Ces suites sont définies en général par récurrence (ou sont déduites d'une suite définie par récurrence), tel l'exemple historique (et simpliste) des générateurs par congruence linéaire comme

$$X_{n+1} = 16807\ X_n \pmod{2^{31}-1}.$$

Le premier terme X_0 de la suite est appelé la **graine** (en anglais, *seed*) de l'algorithme. Afin de pouvoir reproduire la simulation, il suffit de mémoriser la graine. Inversement, si on veut obtenir une simulation « aléatoire », on peut utiliser, pour la graine, une quantité physique à laquelle l'ordinateur a accès, et peu prévisible : typiquement, le nombre de millisecondes de l'horloge du système au moment de la requête. Ce procédé introduit bien un aléa, rendant imprévisible le résultat de la simulation, mais ne peut être répété rapidement des milliers de fois tout en restant imprévisible, c'est pourquoi on définit seulement X_0 ainsi (le premier terme est aléatoire), mais les termes suivants se déduisent du premier (d'une façon qui « paraît » aléatoire). C'est souvent un bon compromis.

1) Voir l'aide de `rand` (début de la partie Description), et expliquer la commande suggérée : `rand('seed',getdate('s'))`

Notons qu'il n'est pas possible de dire si une suite (finie) donnée d'entiers est « aléatoire » : n'importe quelle suite finie de 0 et de 1 peut être le résultat d'une suite de tirages à pile ou face, et la suite constante (1,1,...,1) est aussi probable que n'importe quelle autre !

Pour une suite *infinie*, on pourrait en principe vérifier si elle satisfait la loi des grands nombres (en particulier l'équirépartition), et toutes les propriétés *presque sûres* des suites de variables aléatoires i.i.d.. Ce serait une façon de définir une suite aléatoire infinie.

Pour une suite finie pseudo-aléatoire, on s'attend notamment à ce qu'elle vérifie « approximativement » la loi des grands nombres, et soit « bien mélangeante ». Une définition plus précise est délicate, et dépendrait en fait de l'application voulue : parfois, on souhaite simplement une suite de valeurs « bien répartie » dans un ensemble, sans s'inquiéter de la dépendance entre termes successifs, même si d'autres contextes exigent des contraintes d'indépendance plus importantes. De plus, si on a besoin d'une suite plus longue, il faut un « meilleur » générateur (remarquons qu'une suite pseudo-aléatoire est périodique, si on l'itère assez longtemps). Évaluer la qualité d'un générateur est un problème difficile.

Dans la suite, on se basera sur la fonction `rand()` de Scilab (et `grand`), sans se demander comment elle est programmée : on supposera que les différents appels à `rand` renvoient des réalisations d'une suite de variables aléatoires indépendantes et de loi uniforme dans [0,1]. Notons que c'est donc une « fonction » d'un genre particulier : elle renvoie des résultats différents selon les fois.

Dès lors, **simuler (la loi d')une variable aléatoire** X signifie écrire une fonction qui, à l'aide de la fonction `rand`, fournit une valeur aléatoire distribuée comme X.

2) Par exemple, quelle loi est simulée par la commande `floor(2*rand())` ?

1.2 Utilisation pour la méthode de Monte-Carlo

On utilisera des nombres aléatoires pour divers types de tâches :
- se faire une idée qualitative du comportement d'un processus aléatoire (via une représentation graphique, en général) : observer une seule ou quelques simulations pour voir l'allure « typique » d'une réalisation
- simuler des données fictives afin ensuite de tester sur celles-ci l'efficacité de méthodes statistiques
- estimer une probabilité ou une espérance par la **méthode de Monte-Carlo**.

La méthode de Monte-Carlo consiste à obtenir une valeur approchée de l'espérance $E[X]$ d'une variable aléatoire (intégrable) X en simulant un grand nombre N de variables aléatoires $X_1,\dots,X_N$ indépendantes et de même loi que X, afin d'appliquer la loi forte des grands nombres :

$$E[X] \simeq \frac{X_1+\cdots+X_N}{N}.$$

En particulier, pour une fonction φ telle que $\varphi(X)$ est intégrable,

$$E[\varphi(X)] \simeq \frac{\varphi(X_1)+\cdots+\varphi(X_N)}{N}$$

et, pour $A \in \mathcal{B}(\mathbb{R}^d)$ si X est à valeurs dans $\mathbb{R}^d$,

$$P(X \in A) \simeq \frac{1}{N}\mathrm{Card}(\{1 \le n \le N | X_n \in A\}).$$

NB. Pour obtenir les variables aléatoires indépendantes $X_1,\dots,X_N$, il suffit de répéter N fois un programme qui simule X : la fonction `rand()` renverra à chaque fois des valeurs indépendantes des précédentes. Dès lors que l'on sait simuler X, la mise en œuvre est donc très simple.

La méthode de Monte-Carlo est une méthode de calcul approché d'intégrales : si X a une loi de densité f_X sur $\mathbb{R}^d$, alors on estime

$$E[\varphi(X)] = \int_{\mathbb{R}^d} \varphi(x) f_X(x) dx.$$

Par exemple, si X suit la loi uniforme sur $[0,1]^d$ (simulée par `X=rand(d,1)`) alors, pour $\varphi : [0,1]^d \to \mathbb{R}$ intégrable,

$$E[\varphi(X)] = \int_{[0,1]^d} \varphi(x) dx.$$

L'**erreur** d'estimation s'obtient par le théorème central limite :

$$\sqrt{N}\left(\frac{X_1+\cdots+X_N}{N} - E[X]\right) \xrightarrow[N]{\text{(loi)}} \mathcal{N}(0,\sigma_X^2)$$

où $\sigma_X^2 = \mathrm{Var}(X)$.

Ainsi, l'erreur est de l'ordre de $\frac{1}{\sqrt{N}}$ si on utilise N variables aléatoires. S'il s'agit de trouver la valeur approchée d'une intégrale sur $\mathbb{R}$, il est en général nettement plus efficace d'utiliser une méthode déterministe par subdivision de l'intervalle d'intégration en N morceaux (méthode des milieux : erreur d'ordre $\frac{1}{N^2}$, méthode de Simpson : erreur d'ordre $\frac{1}{N^4}$). Cependant,
- ces méthodes supposent la fonction régulière (de classe $\mathcal{C}^2$, de classe $\mathcal{C}^4$)
- ces méthodes s'étendent en dimension supérieure, mais au prix d'un coût de calcul important (subdivision en N^d cellules).

Par suite, pour un calcul d'intégrale, on n'utilisera la méthode de Monte-Carlo que si
- les fonctions sont très irrégulières
- la dimension est grande (à partir de 5 ?)
- il est beaucoup plus facile de programmer la méthode de Monte-Carlo (problème de nature probabiliste, ensemble d'intégration compliqué) et on n'a pas besoin d'une grande précision pour le résultat.

1.3 Intervalle de confiance sur la moyenne

On déduit du théorème central limite précédent un intervalle de confiance :

$$P\left(\left|\frac{X_1+\cdots+X_N}{N} - E[X]\right| > \frac{\sigma_X A}{\sqrt{n}}\right) \simeq P(|Z|>A) = 1 - 2\int_A^\infty e^{-t^2/2}\frac{dt}{\sqrt{2\pi}}$$

où $Z \sim \mathcal{N}(0,1)$. Pour $A = 1{,}96$, $P(|Z|>A) \simeq 0{,}95$ donc ceci donne : avec probabilité 95%, $E[X]$ (la vraie valeur) appartient à l'intervalle de centre $\overline{X}_N = \frac{X_1+\cdots+X_N}{N}$ (la moyenne empirique) et de demi-largeur $\frac{1{,}96\sigma_X}{\sqrt{N}}$.

3) Vérifier à l'aide de la fonction `cdfnor` la valeur $A = 1{,}96$ (voir l'aide pour comprendre `cdfnor("X",0,1,0.975,0.025)`)

Si la variance σ_X^2 n'est pas supposée connue, on peut l'approcher avec

$$S_n^2 = \frac{1}{N-1}\sum_{n=1}^{N}(X_n - \overline{X}_N)^2$$

(Remplacer σ^2 par S_n^2 dans le théorème central limite est justifié par le lemme de Slutzky, cf. exercices).
On pourra donc toujours assortir les approximations par méthode de Monte-Carlo d'un intervalle de confiance (asymptotique, pour $N \to \infty$) de niveau 95% : l'intervalle

$$\Big[\overline{X}_N - \frac{2\sqrt{S_N^2}}{\sqrt{N}}, \overline{X}_N + \frac{2\sqrt{S_N^2}}{\sqrt{N}}\Big].$$

NB. Dans Scilab, $\overline{X}_N$ s'obtient avec `mean` et S_n^2 avec `variance` (peuvent aussi s'appliquer ligne par ligne, ou colonne par colonne : voir l'aide)
Dans le cas où on estime une probabilité $p = P(X \in A)$ par

$$P(X \in A) \simeq \overline{p}_N = \frac{1}{N}\mathrm{Card}(\{1 \le N \le N | X_N \in A\}),$$

alors la variance est $p(1-p)$, que l'on peut aussi majorer par $\frac{1}{4}$ (ou bien estimer par $\overline{p}_N(1-\overline{p}_N)$) : avec probabilité $\ge 95\%$ (si N est assez grand), p appartient à l'intervalle

$$\Big[\overline{p}_N - \frac{1}{\sqrt{N}}, \overline{p}_N + \frac{1}{\sqrt{N}}\Big].$$

1.4 Dans Scilab

En plus de `rand`, Scilab fournit la fonction `grand` qui permet facilement de simuler les lois de probabilités les plus courantes.

4) En lisant l'aide, simuler un tableau de 10 variables aléatoires de loi Bin(15, 0.4), puis de loi Poiss(2), et enfin Geom(1/3)

Attention :
- `grand(m,n,'exp',Av)` simule des variables exponentielles de moyenne `Av`, donc de paramètre $\frac{1}{\texttt{Av}}$ (loi Exp(1/`Av`))
- `grand(m,n,'nor',Av,Sd)` simule des variables gaussiennes de moyenne `Av` et d'écart-type `Sd`, donc de variance `Sd`^{2} (loi souvent notée $\mathcal{N}$(`Av`,`Sd`^{2}))

5) Simuler $N = 10000$ variables aléatoires indépendantes de loi Exp(1/2), et calculer leur moyenne et leur variance empiriques. En déduire un intervalle de confiance pour leur espérance.

2 Méthodes de simulation

Bien que la fonction `grand` permette de simuler la plupart des lois usuelles, il est intéressant de savoir comment elle fonctionne : partant de la fonction `rand` (donc de variables de loi uniforme sur [0,1]), comment simuler les autres lois classiques ?
On évitera donc d'utiliser `grand` dans cette partie.

2.1 Loi uniforme dans $\{1,\dots,N\}$

6) Quelle loi suit le résultat de `floor(N*rand())` ?

7) Écrire une fonction `rand_uniforme(m,n,N)` qui renvoie une matrice de taille (`m`,`n`) de nombres indépendants de loi uniforme dans {1,2,...,`N`}

2.2 Méthode par inversion (de la fonction de répartition)

On rappelle un résultat de la fiche d'exercice : si F est la fonction de répartition d'une variable aléatoire réelle X, c'est-à-dire $F(t) = P(X \leq t)$ pour $t \in \mathbb{R}$, alors en définissant son inverse généralisée $F^{-1}(u) = \inf\{t | F(t) \geq u\}$, si U suit la loi uniforme sur [0,1], $F^{-1}(U)$ a même loi que X.
Si on peut calculer F^{-1}, on peut donc simuler X à partir de U.
Pour la loi Exp(λ), on a $F(x) = 1 - e^{-\lambda x}$ pour $x > 0$, d'où $F^{-1}(u) = \frac{1}{\lambda}\ln(1-u)$

8) Justifier que $\frac{1}{\lambda}\ln U \sim \text{Exp}(\lambda)$. En déduire une fonction `rand_exp(m,n,lambda)` qui renvoie une matrice de taille (`m`,`n`) de variables indépendantes de loi exponentielle de paramètre λ

Pour la loi de Cauchy (de densité $x \mapsto \frac{1}{\pi(1+x^2)}$), on a $F(x) = \frac{2}{\pi}\arctan(x) + \frac{1}{2}$, d'où $F^{-1}(u) = \tan\frac{\pi}{2}(2u-1)$.

9) En déduire une fonction `rand_cauchy(m,n)` qui renvoie une matrice de taille (`m`,`n`) de variables indépendantes de loi de Cauchy.

10) Représenter graphiquement la suite des moyennes $\frac{1}{N}(X_1 + \cdots + X_N)$ en fonction de N, où les X_i sont de loi Exp(1), puis de loi de Cauchy.

2.3 Lois discrètes

Si $U \sim \mathcal{U}([0,1])$, et $p = (p_1, p_2, \ldots)$ est un vecteur de probabilités ($p_i \geq 0$ et $\sum_i p_i = 1$), alors la variable X telle que $X = i$ si $p_1 + \cdots + p_{i-1} \leq U < p_1 + \cdots + p_i$ suit la loi $P(X = i) = p_i$.

11) Simuler une variable aléatoire X sur {1,2,3} de loi donnée par $P(X=1) = 0{,}3$, $P(X=2) = 0{,}1$ et $P(X=3) = 0{,}6$.

Remarque. On peut voir ceci comme un cas particulier de la méthode par inversion

2.4 Loi conditionnelle

Si A est un événement avec $P(A) > 0$, la loi conditionnelle sachant A peut se simuler en répétant la simulation de P de façon indépendante jusqu'à la première fois où A est réalisé.
NB. Le nombre de simulations suit une loi géométrique de paramètre $P(A)$, donc son espérance est $\frac{1}{P(A)}$. Si $P(A)$ est petit, il vaut mieux trouver une autre méthode...

12) Simuler deux variables aléatoires X,Y indépendantes de loi Exp(1) conditionnées à ce que $X + 2Y > 5$.
Estimer $E[X | X + 2Y > 5]$.

2.5 Méthode du rejet

Pour simuler des variables aléatoires à densité, la méthode du rejet se base sur une loi conditionnelle.
Exemple simple : Simulation de la loi uniforme sur $K \subset \mathbb{R}^2$ borné, avec Aire(K) > 0.
On choisit le plus petit pavé P tel que $K \subset P$. On sait simuler une variable aléatoire de loi uniforme sur P (les composantes sont indépendantes et uniformes). On montre facilement que la loi uniforme sur K est la loi de X sachant $\{X \in K\}$.

13) Simuler une variable de loi uniforme dans le disque $D(0,1)$. Combien d'appels à `rand()` faut-il en moyenne ?

Cas général. On souhaite simuler la loi de densité f, et on suppose que l'on sait simuler X de densité g où $f \leq Cg$ (avec C une constante qui faut connaître).
Simuler f revient à considérer l'abscisse d'un point de loi uniforme sous le graphe de f. De même pour g. Inversement, si X a pour densité g, le point $(X, g(X)U)$, où $U \sim \mathcal{U}([0,1])$ est indépendant de X, suit la loi uniforme sous le graphe de g, donc $(X,Y) = (X, Cg(X)U)$ suit la loi uniforme sous le graphe de Cg.

On en déduit que la loi de (X,Y) sachant que $Y \leq f(X)$ est uniforme sous le graphe de f, donc la loi de X sachant $Cg(X)U \leq f(X)$ a pour densité f.
D'où l'algorithme : On simule X de densité g et $U \sim \mathcal{U}([0,1])$ indépendant de X, et on recommence jusqu'à ce que $Cg(X)U \leq f(X)$. Alors X suit la loi de densité f.

14) Simuler la loi de densité $\frac{2}{\sqrt{2\pi}}e^{-x^2/2}$ (loi de $|Z|$ où $Z \sim \mathcal{N}(0,1)$) à l'aide de la loi Exp(1). (Vérifier $C = e^{-2}\sqrt{\frac{2}{\pi}} < 0{,}11$)
En déduire une fonction `rand_norm(Moy,Var)` qui simule une variable aléatoire de loi $\mathcal{N}$(`Moy`,`Var`)

2.6 Pour la loi gaussienne

La **méthode polaire** exploite l'invariance par rotation de la loi de (X,Y) où X,Y sont indépendants et de loi $\mathcal{N}(0,1)$.
Par changement de variables (cf. cours ou exercices) on vérifie que $(X,Y) = (R\cos\Theta, R\sin\Theta)$ où R,Θ sont indépendantes, Θ suit la loi uniforme sur $[0,2\pi]$ et R^2 suit la loi Exp(1/2).
Donc (X,Y) a même loi que $\sqrt{-2\ln U_1}(\cos(2\pi U_2), \sin(2\pi U_2))$. On simule ainsi simultanément deux variables aléatoires indépendantes de loi $\mathcal{N}(0,1)$.

15) En déduire une fonction `rand_norm2(n,Moy,Var)` qui renvoie `n` variables aléatoires indépendantes de loi $\mathcal{N}$(`Moy`,`Var`)

16) Estimer $P(|Z| > 1{,}96)$ où $Z \sim \mathcal{N}(0,1)$

17) Comment simuler une variable aléatoire de loi uniforme sur la sphère unité de $\mathbb{R}^3$?

3 Simulation de chaînes de Markov

La loi d'une chaîne de Markov étant définie par récurrence, on utilise en général une boucle `for`.
Dans les cas finis, on pourra aussi utiliser `grand(n,'markov',P,x0)` (mais ce n'est souvent pas l'idéal).

3.1 Ruine du joueur

Soit $0 < p < 1$ et $N \in \mathbb{N}^*$. On considère la chaîne de Markov de matrice P donnée par $P(0,0) = 1$, $P(N,N) = 1$ et, pour $0 < x < N$, $P(x,x+1) = p$ et $P(x,x-1) = 1-p$.

18) Écrire une fonction `ruine(N,p,x0)` qui renvoie la trajectoire de la chaîne de Markov jusqu'à ce qu'elle atteigne 0 ou N

19) Représenter sur un même graphique 10 trajectoires

On note T le temps d'atteinte de $\{0,N\}$.

20) Estimer $P(X_T = N)$ et $E[T]$, en faisant varier le point de départ $X_0 = k$.
Comparer avec $P(X_T = N) = \frac{k}{N}$ si $p = \frac{1}{2}$ et $P(X_T = N) = \frac{1-\rho^k}{1-\rho^N}$ si $p \neq \frac{1}{2}$, avec $\rho = \frac{1-p}{p}$.
Et $E[T] = k(N-k)$ si $p = \frac{1}{2}$, $E[T] = \frac{1}{2p-1}\left(n\frac{\rho^k-\rho^n}{1-\rho^n} - k\right)$ si $p \neq \frac{1}{2}$.

3.2 Urne d'Ehrenfest (prudente)

Soit $0 < p < 1$ et $N \in \mathbb{N}^*$. On considère la chaîne de Markov de matrice P donnée par $P(0,0) = 1$, $P(N,N) = 1$ et, pour $0 < x < N$, $P(x,x+1) = \frac{N-x}{2N}$, $P(x,x) = \frac{1}{2}$ et $P(x,x-1) = \frac{x}{2N}$.

21) Écrire une fonction `urne(N,p,x0,n)` qui renvoie la trajectoire de la chaîne de Markov jusqu'au temps n

22) Représenter graphiquement une trajectoire (sur plusieurs durées)

23) Estimer le temps d'atteinte de $\frac{N}{2}$ (si N est pair), partant de 0, pour plusieurs valeurs de N

24) Estimer la proportion de temps passé en chaque état au long d'une unique trajectoire

25) Estimer la loi de X_n où n est assez grand pour que (à en juger par le premier graphique) la chaîne de Markov se trouve approximativement à l'équilibre

26) Calculer (algébriquement) la loi invariante π de la chaîne de Markov ; comparer avec les résultats précédents

27) Calculer algébriquement la loi μ_n de X_n, et la distance L^1 entre μ_n et π, et la représenter graphiquement. Commenter.

3.3 Urne de Pólya

L'urne « de Pólya » contient initialement 1 boule rouge et 1 boule noire. On tire une boule puis on la replace dans l'urne avec une nouvelle boule de la même couleur, et on répète cette opération.

28) Représenter graphiquement l'évolution de la proportion de boules bleues. Étudier la répartition de cette proportion après un temps long (histogramme, pour commencer). Faire varier la composition initiale de l'urne.

4 Tester une méthode : comparaison statistique de lois

En supposant que les tirages $X_1,\dots,X_n$ sont bien indépendants et suivent la même loi qu'une variable X, comment estimer cette loi, ou vérifier que les tirages sont conformes à une loi prescrite ?

4.1 Lois discrètes : test du χ^2

On suppose que X est à valeurs dans $\{1,\dots,r\}$. Pour **estimer** sa loi, il suffit d'estimer $P(X=k)$ pour $k=1,\dots,r$, à l'aide des fréquences empiriques :

$$P(X=k) \simeq \hat{p}_k = \frac{N_k}{n} \quad \text{où } N_k = \mathrm{Card}(\{1 \le i \le n | X_i = k\}) \quad \text{pour } k=1,\dots,r.$$

Pour **comparer** la loi de X à un vecteur de probabilité donné $q=(q_1,\dots,q_r)$, on utilise le test du χ^2 d'adéquation : On définit la « distance du khi-deux »

$$D(\hat{p},q) = \sum_{k=1}^{r} \frac{(\hat{p}_k - q_k)^2}{q_k} = \frac{1}{n}\sum_{k=1}^{r} \frac{(N_k - nq_k)^2}{nq_k}.$$

On peut démontrer (voir cours de statistiques) que, si X suit la loi q,

$$nD(\hat{p},q) \xrightarrow[n]{\text{(loi)}} \chi^2_{r-1}$$

où χ^2_{r-1} est la loi du khi-deux à $r-1$ degrés de liberté. On remarque aussi que, si $q \neq p$, $D(\hat{p},q)$ admet une limite strictement positive p.s. donc $nD(\hat{p},q) \to_n +\infty$ p.s.

D'où un test consistant de niveau α : « On rejette l'hypothèse $p = q$ si $nD(\hat{p},q) > A$ » où A est tel que $\mathbb{P}(C > A) = \alpha$ avec $C \sim \chi^2_{r-1}$.
En pratique, on demande souvent d'avoir $N_k \geq 5$ pour tout k afin que le test soit significatif. Si ce n'est pas le cas, on peut par exemple regrouper des classes (quitte à avoir un test moins puissant).
Dans Scilab, on pourra obtenir la valeur de A (pour avoir $\alpha = 0{,}05$ par exemple) à l'aide de `cdfchi` (en l'occurrence, `cdfchi("X",r-1,0.95,0.05)`)

29) Tester la fonction que vous avez écrite à la question 11), sur 5000 tirages, au niveau 95% : la simulation est-elle conforme à la loi voulue ?

On peut aussi utiliser ce test pour des lois continues, en les discrétisant : on définit r classes $]-\infty,x_1[$, $[x_1,x_2[,\ldots,$ $[x_{r-1},+\infty[$, et on pose $Y = i$ si X est dans i-ième classe. On peut alors tester l'adéquation de tirages $Y_1,\ldots,Y_n$ avec la loi $P(Y = 1) = P(X < x_1)$, $P(Y = 2) = P(x_1 \leq X < x_2)$, etc.
NB. Il faut toujours que l'effectif de chaque classe soit au moins égal à 5 pour appliquer le test du χ^2

30) Tester l'adéquation de 5000 tirages de la fonction `rand_norm(0,1)` avec la loi $\mathcal{N}(0,1)$, en discrétisant $[-2,2]$ en 10 classes de même largeur (+ les extrêmes). Utiliser `cdfnor` pour calculer les probabilités théoriques des différentes classes.

4.2 Lois continues

4.2.1 Estimation de fonction de répartition

Pour estimer une loi (quelconque) sur $\mathbb{R}$ à partir d'observations $X_1,\ldots,X_n$ indépendantes et de même loi, on peut estimer sa fonction de répartition F à l'aide de la fonction de répartition empirique

$$F_n : t \mapsto F_n(t) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}_{\{X_i \leq t\}} = \frac{\text{Card}\{1 \leq i \leq n | X_i \leq t\}}{n}.$$

Par la loi forte des grands nombres, pour tout t, $F_n(t)$ converge vers $F(t)$ p.s. On peut justifier que p.s., pour tout t, $F_n(t)$ converge vers $F(t)$ (noter la différence), en utilisant le fait que F_n et F sont continues à droite et croissantes. Et on peut aussi, en utilisant un théorème de Dini (pas le plus courant) et le fait que F_n et F ont des limites en $-\infty$ et $+\infty$, montrer que la convergence est même uniforme :

$$\text{p.s.,} \quad \sup_{\mathbb{R}} \left|F_n - F\right| \underset{n}{\longrightarrow} 0.$$

Dans `scilab`, pour représenter la fonction de répartition empirique, on peut utiliser `plot2d2` (dessin de fonctions en escalier)

31) Expliquer la commande suivante, pour afficher la fonction de répartition empirique d'un échantillon X :
`N=50; X=rand(N,1); plot2d2(-gsort(-X),(1:N)/N);`

32) Écrire une fonction `fn_repartition(X,xmin,xmax)` qui dessine la fonction de répartition empirique sur [`xmin`,`xmax`] (en supposant les valeurs de `X` dans cet intervalle). (On ajoutera `xmin` et `xmax` à `X`, et `0` et `1` aux ordonnées)

33) Comparer graphiquement les fonctions de répartition empiriques de tirages des fonctions `rand_norm`, `rand_exp`, avec leur fonction de répartition théorique (utiliser `cdfnor`).

4.2.2 Estimation de densité

On est souvent plus intéressé à représenter la densité d'une loi inconnue que sa fonction de répartition, car elle est plus simple à lire. On utilise en général pour cela des histogrammes.
Il faut être prudent : si le nombre de classes de l'histogramme est mal choisi, il n'approche pas la densité. Avec trop peu de classes, on peut passer à côté de certaines fluctuations de la densité ; et avec trop de classes, on pourrait à l'inverse avoir seulement 0 ou 1 tirage dans chacune.

Une règle empirique consiste à prendre $n^{1/3}$ classes (optimalité dans le cas gaussien) ; par défaut Excel en prend $\sqrt{n}$. Pour une simple représentation graphique, on pourra ajuster le nombre de classes visuellement.
Dans `scilab`, on dispose de la commande `histplot(n,X)` (on peut aussi préciser les classes, voir l'aide)

34) Tester `histplot(20,grand(10000,1,"nor",0,1),2); x=-4:0.01:4;plot(x,exp(-x.^2/2)/sqrt(2*%pi),'r');`

35) Représenter des histogrammes des diverses lois simulées précédemment ; comparer aux densités théoriques

NB. Il existe des méthodes un peu plus élaborées (méthodes à noyaux) pour estimer des densités à l'aide de fonctions régulières, avec de meilleurs résultats que les histogrammes ; on verra peut-être un texte sur le sujet.

4.2.3 Test de Kolmogorov-Smirnov, et version non-asymptotique

On peut déduire de la convergence des fonctions de répartition empiriques un test, dit de **Kolmogorov-Smirnov**, d'adéquation à une loi continue.
On peut montrer que, si les X_i sont indépendantes et de loi de fonction de répartition F **continue**, alors

$$\sqrt{n}\sup_{t\in\mathbb{R}}\left|F_n(t)-F(t)\right| \xrightarrow[n]{\text{(loi)}} KS,$$

où KS est la loi de Kolmogorov-Smirnov : c'est une loi sur $\mathbb{R}_+$ qui ne dépend pas de μ.
On peut donc faire le test suivant, consistant et de niveau α : « On rejette l'hypothèse que la fonction de répartition est F si $\sqrt{n}\sup_{\mathbb{R}}|F_n-F|>A$ » où A est tel que $\mathbb{P}(Z>A)=\alpha$ si Z suit la loi de Kolmogorov-Smirnov.
Notons qu'en pratique le calcul de la distance $\sup_n|F_n-F|$ se résume au calcul du maximum des différences aux points $X_1,\dots,X_n$ (à gauche et à droite) du fait de la croissance des fonctions.
Dans `scilab`, la fonction de répartition (inverse) de la loi de Kolmogorov-Smirnov n'est pas présente par défaut. On peut utiliser `pks` de la boîte à outils Stixbox (disponible à l'agrégation).
On peut aussi calculer par méthode de Monte-Carlo la valeur dont on a besoin : comme la loi de $\sup_{\mathbb{R}}|F_n-F|$ ne dépend pas de la fonction de répartition continue F (ça pourrait être un exercice), on peut choisir une loi continue quelconque (donc la loi uniforme sur [0,1]) et estimer $\Psi_n(A)=P(\sqrt{n}\sup_{\mathbb{R}}|F_n-F|>A)$ pour un A donné. On obtient ainsi un **test non-asymptotique** de niveau $1-\alpha$: on calcule $A=\sqrt{n}\sup_{\mathbb{R}}|F_n-F|$ avec les données (et la fonction de répartition F testée), et si $\Psi_n(A)<\alpha$, on rejette le test : les données ne sont pas pas conforme à la loi F au niveau $1-\alpha$. (C'est-à-dire que seule une proportion α des tirages selon la loi F fournirait une valeur A aussi grande)

36) Écrire une fonction `pks(x,n)` qui renvoie $P(Z\leq x)$ où Z suit (approximativement) la loi de Kolmogorov-Smirnov : utiliser la limite ci-dessus, pour une famille $X_1,\dots,X_n$ de variables uniformes sur [0,1], et un grand nombre N de tirages d'une telle famille.

37) Tester l'adéquation de 1000 tirages de la fonction `rand_norm2` avec la loi $\mathcal{N}(0,1)$, au niveau 95%. On appellera `pks` sur la valeur $\sqrt{n}\sup|F_n-F|$ et avec paramètre n, et on comparera le résultat à 0.95. Utiliser `cdfnor` pour calculer F.