

TP : SIMULATION DE VARIABLES ALÉATOIRES, EXEMPLES ET APPLICATIONS

1 Principes de la simulation aléatoire

1.1 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

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

(Ici, $2^{-31}X_n$ suit approximativement la loi uniforme sur $[0,1]$)

Le premier terme X_0 de la suite est appelé la **graine** (en anglais, *seed*) de l'algorithme. Dans certains cas, il est utile de choisir une graine particulière : notamment, pour pouvoir reproduire la simulation. Dans d'autres cas, on souhaite au contraire introduire une part d'« aléatoire » dans le choix de la graine, à savoir lorsque l'on veut obtenir des résultats différents (presque) à chaque fois. Par exemple, la graine est fixée en fonction du nombre de millisecondes de l'horloge de l'ordinateur au moment où l'on commence la simulation. Attention, on n'utilise cette méthode que pour la graine, tandis que les tirages suivants s'en déduisent : c'est une forme de compromis.

- 1) La commande `rand('seed',n)` initialise la fonction `rand()` à l'aide d'une valeur entière `n`. Taper dans la console `rand('seed',10)` puis regarder quelques valeurs renvoyées par `rand()`. Recommencer : que constate-t-on ? 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 $(1,1,\dots,1)$ est aussi probable que n'importe quelle autre !

Qu'entend-on donc par « pseudo-aléatoire » ? On s'attend à ce qu'une suite pseudo aléatoire 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.

1.2 Simuler une variable aléatoire

Dans la suite, on se basera sur la fonction `rand()` de Scilab (et sur `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 quand on l'appelle plusieurs fois.

Dès lors, **simuler (la loi d')une variable aléatoire X** signifie écrire un programme 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 `X=(rand())<p` ? (où `p=0.3` par exemple)

1.3 Utilisations dans le cadre de l'option Modélisation

On utilisera des nombres aléatoires pour divers types de tâches : principalement,

- 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 (par exemple, pour suggérer une convergence presque sûre) ;
- estimer une probabilité ou une espérance (voire une loi) par la **méthode de Monte-Carlo** ;
- simuler des **données fictives** pour ensuite éprouver sur celles-ci l'efficacité de méthodes statistiques, ou pour **étalonner des tests**.

2 Quelques méthodes de simulation

On s'intéresse ici à la façon de simuler une loi donnée.

2.1 Dans Scilab : grand

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

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

Attention :

- `grand(m,n, 'exp', Av)` simule des variables exponentielles de moyenne Av , donc de paramètre $\frac{1}{Av}$ (loi $\text{Exp}(1/Av)$)
- `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 $\mathcal{N}(Av, Sd^2)$)

- 4) Simuler $N = 10000$ variables aléatoires indépendantes de loi $\text{Exp}(1/2)$, et calculer leur moyenne et leur variance empiriques. Donner un intervalle de confiance pour leur espérance.

Bien que la fonction `grand` permette de simuler la plupart des lois usuelles, il est intéressant de savoir comment elle fonctionne : en partant de la fonction `rand` (dont on admet qu'elle fournit des réalisations de variables aléatoires indépendantes et de loi uniforme sur $[0,1]$), comment simuler les autres lois classiques ?

NB. On évitera donc d'utiliser `grand` dans la suite de cette partie.

2.2 Loi uniforme dans $\{1, \dots, r\}$

- 5) Quelle loi suit le résultat de `floor(r*rand())` ?

- 6) Écrire une fonction `rand_uniforme(m,n,r)` qui renvoie une matrice de taille (m,n) de nombres indépendants de loi uniforme dans $\{1, 2, \dots, r\}$

2.3 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 $\text{Exp}(\lambda)$, on a $F(x) = 1 - e^{-\lambda x}$ pour $x > 0$, d'où $F^{-1}(u) = -\frac{1}{\lambda} \ln(1 - u)$

- 7) 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 $F(x) = \frac{1}{\pi} \arctan(x) + \frac{1}{2}$, d'où $F^{-1}(u) = \tan\left(\frac{\pi}{2}(2u - 1)\right)$.

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

2.4 Lois discrètes

Si $U \sim \mathcal{U}([0,1])$, et $p = (p_1, p_2, \dots)$ est un vecteur de probabilités ($p_i \geq 0$ et $\sum_i p_i = 1$), alors la variable X telle que

$$X = \begin{cases} 1 & \text{si } U < p_1 \\ 2 & \text{si } p_1 \leq U < p_1 + p_2 \\ 3 & \text{si } p_1 + p_2 \leq U < p_1 + p_2 + p_3 \\ \dots & \dots \end{cases}$$

suit la loi p , c'est-à-dire que, pour $i = 1, 2, \dots$, $P(X = i) = p_i$.

- 9) Écrire une fonction `rand_discrete(n)` qui renvoie un vecteur de taille n de variables indépendantes, à valeurs dans $\{1, 2, 3\}$, de loi donnée par $P(X = 1) = 0,3$, $P(X = 2) = 0,1$ et $P(X = 3) = 0,6$.

- 10) Simuler une variable aléatoire N de loi de Poisson de paramètre 5.

2.5 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...

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

2.6 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 $\text{Aire}(K) > 0$.

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 $\{X \in K\}$.

- 12) 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 .

- 13) Simuler la loi de densité $\frac{2}{\sqrt{2\pi}} e^{-x^2/2} \mathbf{1}_{\mathbb{R}_+}(x)$ (loi de $|Z|$ où $Z \sim \mathcal{N}(0,1)$) à l'aide de la loi $\text{Exp}(1)$. (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 $\mathcal{N}(\text{Moy}, \text{Var})$

2.7 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 $\text{Exp}(1/2)$. Donc (X,Y) a même loi que $(\sqrt{-2 \ln U_1} \cos(2\pi U_2), \sqrt{-2 \ln U_1} \sin(2\pi U_2))$. On simule ainsi simultanément deux variables aléatoires indépendantes de loi $\mathcal{N}(0,1)$.

- 14) En déduire une fonction `rand_norm2(n, Moy, Var)` qui renvoie n variables aléatoires indépendantes de loi $\mathcal{N}(\text{Moy}, \text{Var})$
- 15) Estimer $P(|Z| > 1,96)$ où $Z \sim \mathcal{N}(0,1)$

2.8 Quelques processus aléatoires

On appelle **processus aléatoire** une suite de variables aléatoires.

- 16) Soit $(X_n)_{n \geq 1}$ une suite de v.a. de loi $\text{Exp}(1)$. On note $Y_n = \frac{1}{\ln n} \max_{1 \leq k \leq n} X_k$. Représenter graphiquement une réalisation de Y_1, \dots, Y_{1000} . (Par l'exercice 17, Y_n converge en probabilité vers 1, mais pas presque sûrement)
- 17) Pour une suite X_1, \dots, X_N de variables indépendantes de loi $\text{Exp}(1)$, représenter graphiquement la suite des moyennes successives $\frac{1}{n}(X_1 + \dots + X_n)$ pour $n = 1, \dots, N$. De même pour la loi de Cauchy. Commenter.

Un exemple typique de processus est celui d'une chaîne de Markov. La loi d'une chaîne de Markov étant définie par récurrence, on utilise en général une boucle `for` pour simuler les termes successivement. Dans les cas finis, on pourra aussi utiliser `grand(n, 'markov', P, x0)` (mais ce n'est souvent pas l'idéal). Voyons quelques exemples.

2.8.1 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) Représenter une trajectoire de ce processus jusqu'au temps $n=100$, pour $N = 30$ et partant de $X_0 = 15$. On pourra utiliser `grand`.
- 19) Représenter une trajectoire de ce processus jusqu'à ce qu'elle atteigne 0 ou N (tester avec plusieurs valeurs de N , et plusieurs points de départ).

2.8.2 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.

- 20) Représenter graphiquement l'évolution de la proportion de boules bleues, pour plusieurs suites de tirages.

3 Applications en probabilités

3.1 Estimation d'une espérance, d'une probabilité : 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 :

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

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

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

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

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

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.

3.1.1 Remarque : calcul d'intégrale

La méthode de Monte-Carlo fournit en particulier une méthode de calcul approché d'intégrales : si X suit 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 $\varphi : [0,1]^d \rightarrow \mathbb{R}$ intégrable,

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

3.1.2 Erreur, et intervalle de confiance

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

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

où $\sigma_X^2 = \text{Var}(X)$.

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.

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

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

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 $\bar{X}_N = \frac{X_1 + \dots + X_N}{N}$ (la moyenne empirique) et de demi-largeur $\frac{1,96\sigma_X}{\sqrt{N}}$.

21) 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

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

(Remplacer σ^2 par S_n^2 dans le théorème central limite est justifié par le lemme de Slutsky, cf. exercices).

On pourra donc toujours assortir les approximations par méthode de Monte-Carlo d'un intervalle de confiance (asymptotique, pour $N \rightarrow \infty$) de niveau 95% : l'intervalle

$$\left[\bar{X}_N - \frac{2\sqrt{S_N^2}}{\sqrt{N}}, \bar{X}_N + \frac{2\sqrt{S_N^2}}{\sqrt{N}} \right].$$

NB. Dans Scilab, \bar{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

$$P(X \in A) \simeq \bar{p}_N = \frac{1}{N} \text{Card}(\{1 \leq n \leq N | X_n \in A\}),$$

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

$$\left[\bar{p}_N - \frac{1}{\sqrt{N}}, \bar{p}_N + \frac{1}{\sqrt{N}} \right].$$

22) Pour la ruine du joueur : On note T le temps d'atteinte de $\{0, N\}$. Estimer $P(X_T = N)$ et $E[T]$, en faisant varier le point de départ $X_0 = k$. On précisera des intervalles de confiance. 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 $p = \frac{1}{2}$, $E[T] = \frac{1}{2p-1} (n \frac{\rho^k - \rho^n}{1-\rho^n} - k)$ si $p \neq \frac{1}{2}$.

3.2 Estimer une loi

En supposant que les tirages X_1, \dots, X_N sont indépendants et suivent la même loi qu'une variable X , comment estimer cette loi ?

3.2.1 Loïs discrètes

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 :

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

23) Tester ceci sur la fonction que vous avez écrite à la question 9), à partir de 5000 tirages.

3.2.2 Loïs réelles quelconques : estimation de fonction de répartition

Pour estimer une loi (quelconque) sur \mathbb{R} à partir d'observations X_1, \dots, 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 et le fait que F_N et F ont des limites en $-\infty$ et $+\infty$, montrer que la convergence est même uniforme :

$$\text{p.s., } \sup_{\mathbb{R}} |F_N - F| \xrightarrow{N} 0.$$

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

- 24) 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);`
- 25) 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)
- 26) Comparer graphiquement les fonctions de répartition empiriques de tirages des fonctions `rand_norm`, `rand_exp`, avec leur fonction de répartition théorique (utiliser `cdfnor`).
- 27) Pour l'urne de Pólya :
 Estimer la loi de la proportion de boules bleues après un temps long.

3.2.3 Lois continues : 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)

- 28) 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');`
- 29) 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.