TP Statistique n°2
Simulation de lois

Rappel : taper `krdc vnc://nom_du_serveur` dans un terminal.
Après les TPs, les corrigés seront sur ma page
http://www.math.u-bordeaux1.fr/~chabanol/TP_ agreg.html La bibliothèque stixbox (`help stixbox`), gratuite (et disponible à l'agrégation, ce qui n'est pas le cas du "toolbox Statistics" de matlab), contient un certain nombre de fonctions statistiques : elle permet de générer un certain nombre de lois, fournit leur densité ("density function"), leur fonction de répartition ("distribution function"), leur fonction quantile, etc. Attention la loi géométrique est sur $\mathbb{N}$ au lieu de $\mathbb{N}^*$ et le deuxième paramètre de la loi normale est son écart-type, pas sa variance!
Les questions subsidiaires ne sont à faire que si vous avez fini les autres questions...

1. Représentations graphiques d'échantillons statistiques

1.1. **Rappel : Histogramme.** On utilise la commande `histo`. Si X est un vecteur, `histo` trace un histogramme non normalisé. Pour le normaliser, il faut donner 4 paramètres, par exemple `histo(X,10,0,1)`. Le deuxième paramètre est le nombre de classes (ou de colonnes, en anglais "bins" de l'histogramme), le troisième paramètre vaut 0 ou 1 (en général 0 pour une loi continue, 1 pour une discrète mais ce n'est pas crucial), et le quatrième vaut 1 pour indiquer que l'histogramme est normalisé.

1.2. **Fonction de répartition.** Il est souvent plus fiable de comparer des fonctions de répartition empirique et théorique, qu'un histogramme avec une densité (on a vu que le théorème de Glivenko Cantelli garantit une convergence uniforme de l'empirique vers la théorique).
Plusieurs moyens sont possibles pour tracer cette fonction de répartition empirique. Si X est un vecteur aléatoire de taille n, on peut par exemple effectuer les opérations suivantes :

`Y=sort(X);`	Trie le vecteur X .
`n=length(X);`	Détermine la taille de X .
`stairs(Y,[1:n]/n)`	Dessine (en escalier) la fonction de répartition empirique

1. Comprendre pourquoi les instructions précédentes font bien ce qu'il faut.
2. Trouver la fonction matlab de stixbox (utilisez `help stixbox`) qui donne la fonction de répartition d'une loi normale centrée réduite.
3. Générer un échantillon de 1000 réalisations d'une loi normale et tracer sa fonction de répartition empirique. Obtenir sur le même graphe la fonction de répartition théorique

2. Génération aléatoire

2.1. **Transformations plus ou moins élémentaires de la loi uniforme.**
4. Générer un échantillon X de 1000 réalisations d'une loi uniforme sur $[0,1]$. Obtenir à partir de X un échantillon de 1000 réalisations d'une loi uniforme sur $[0,2\pi]$ et un échantillon de 1000 réalisations d'une loi uniforme sur $[-1,1]$ (faire des histogrammes ou tracer les fonctions de répartition pour vérifier)
5. Obtenir à partir d'un échantillon d'une loi uniforme sur [0,1] un échantillon de loi de Bernoulli de paramètre $\frac{1}{4}$ (penser aux opérations booléennes !). Calculer sa moyenne et comparer avec son espérance.
6. Obtenir une réalisation d'une loi binomiale de paramètre $(10, 0.4)$ puis faites une boucle pour en obtenir un échantillon de taille 100.

Utiliser `type rbinom` *pour voir le code effectivement utilisé par stixbox pour générer une loi binomiale. Est-ce la même méthode ? Sinon, quels sont leurs intérêts respectifs ?*
Obtenir un vecteur avec les probabilités théoriques de votre loi binomiale $[P(0), P(1), \dots P(10)]$ *(là encore, stixbox peut vous aider).*
Comparer l'histogramme de votre échantillon avec la loi théorique. Ici on est dans le cas d'une loi discrète : Donc il vaut mieux dessiner les probabilités théoriques à l'aide de `stem`, qui fournit un "diagramme en bâton."
7. Obtenir à partir d'un échantillon d'une loi uniforme sur [0,1] un échantillon de loi uniforme discrète sur $\{1, 2, \dots 6\}$ *(Indication :* `floor(x)` *et* `ceil(x)` *fournissent respectivement l'entier inférieur et supérieur les plus proches d'un réel* x*)*
8. Obtenir un 100*-échantillon d'une loi sur* $\{1, 2, 3\}$ *avec probabilités respectives* $\frac{3}{5}$, $\frac{1}{5}$, $\frac{1}{5}$.
9. Question subsidiaire (à faire à la fin si vous avez fini le reste) : écrire une fonction `discrete` *qui prend en paramètre un entier* n *et un vecteur* p *et génère un* n*-échantillon d'une loi sur* $\{1, 2, \dots,$ `length`$(p)\}$ *avec probabilités respectives* $p(1)$, $p(2), \dots p($ `length`$(p))$.
10. Utiliser la définition de la loi géométrique comme instant de premier succès pour générer un échantillon de 1000 réalisations de loi géométrique de paramètre $\frac{1}{6}$ *(penser à utiliser une boucle* `while`*). Comparer la moyenne et l'espérance,*

2.2. **Inversion de la fonction de répartition.**
11. Ecrire une fonction `rexp` *qui prend en paramètre un entier* n *et un réel* a *et génère un* n*-échantillon d'une loi exponentielle de paramètre* a *(attention la fonction* ln *s'appelle* `log`*). L'utiliser pour générer un échantillon de loi exponentielle de paramètre 2, vérifier en traçant la fonction de répartition empirique et la fonction de répartition théorique sur un même graphe. Comparer la moyenne empirique et l'espérance.*
12. Ecrire une fonction `rpareto` *qui prend en paramètre un entier* n *et un réel* a *et génère un* n*-échantillon d'une loi de Pareto de densité* $\frac{a}{x^{a+1}} 1_{[1,+\infty[}$. *Vérifier en traçant la fonction de répartition.*

2.3. **Méthode par rejet.** La commande `find` va être très utile pour cette méthode : si X est un vecteur, `find(X)` retourne les indices pour lesquels X est non nul; et donc (par exemple) `J=find((X+Y)<1)` retourne l'ensemble des indices pour lesquels $X + Y$ est inférieur à 1. Ensuite, `X(J)` désigne le vecteur où on a gardé uniquement les composantes d'indice appartenant à J.
13. On cherche à générer des points uniformément à l'intérieur de l'ellipse E *d'équation* $\frac{x^2}{4} + y^2 = 1$. *Trouver un pavé* $[a, b] \times [c, d]$ *qui contienne l'ellipse* E.
Générer deux vecteurs X *et* Y *contenant les abscisses et les ordonnées de 10000 points distribués de manière uniforme sur* $[a, b] \times [c, d]$. *Utiliser* `find` *pour déterminer l'ensemble des indices* i *pour lesquels le point de coordonnées* $(X(i), Y(i))$ *appartient à l'intérieur de l'ellipse. Obtenir deux sous-vecteurs* Xd *et* Yd *ne contenant que les valeurs de* X *et de* Y *correspondant à ces indices. Tracer les points de coordonnées* Xd *et* Yd *(utiliser l'option* `'d'` *de* `plot` *pour ne pas avoir de segments de droite entre les points).*
Tracer l'ellipse sur la même figure (utiliser un paramètrage de l'ellipse)
Quelle est la loi de la taille de Xd *et* Yd *?*
A votre avis, Xd *et* Yd *suivent-elles une loi uniforme ? (réponse graphique)*
14. Générer maintenant deux échantillons indépendants r *et* t *de loi respectivement uniforme sur* $[0, 1]$ *et uniforme sur* $[0, 2\pi]$ *et tracer les points de coordonnées* $(2r \cos t, r \sin t)$. *On n'obtient ainsi que des points dans l'ellipse. La distribution vous semble-t-elle uniforme ? (réponse graphique)*
15. Déterminer un pavé qui contienne la région $A = \{(x, y) \in [0, 1], 0 \leq y \leq \frac{10}{3} x(1 - x)^3\}$. *Simuler*

un échantillon de loi uniforme sur A, puis obtenir un échantillon d'une variable aléatoire de loi β de densité $\frac{10}{3}x(1-x^3)1_{[0,1]}$.

3. Simulation de la loi de Poisson

Un algorithme possible pour simuler une loi de Poisson de paramètre λ repose sur la propriété suivante, que l'on verra dans le cadre des processus de Poisson :
Soit (T_n) une suite de v.a. de loi exponentielle indépendantes de paramètre λ. On définit pour tout n $S_n = \sum_{i=1}^{n} T_i$. Alors $N = \text{Card}\{n, S_n \in [0,1]\} = \max\{n, S_n \leq 1\}$ suit une loi de Poisson de paramètre λ.
16. Utiliser cette propriété pour simuler une loi de Poisson de paramètre 2, puis faites une boucle pour obtenir un échantillon.