Informatique - TD 9 | Simulation de v.a. à densité et approximations

Explications :

Le but de ce TD est de savoirs simuler des variables aléatoires suivant les lois à densité vues en cours cette année. Chacune de ces méthodes est à connaître pour le concours.

Avant toute chose, commencez, comme pour le TP sur les variable discrètes, par télécharger (sur le site habituel) et ouvrir dans pyzo le ficher python annexe à ce TD (disponible dans la case "documents".) Celui-ci vous servira à construire les différentes simulations demandées dans cette feuille d'exercices (en complétant les parties adéquates) et à tester la cohérence de vos fonctions.

Les consignes d'utilisation de cette feuille sont exactement les mêmes que pour la fiche sur les simulations de variables discrètes. (On ne touche pas aux noms de fonction, on ne modifie pas la fonction `test` *et on compile toutes le fichier avec CTRL+E avant de commencer.)*

Loi normale et python

Dans le cadre de nos simulations, nous allons avoir besoin d'apprendre à nous servir de Python pour obtenir quelques informations relatives à la fonction de répartition Φ d'une variable $X \hookrightarrow N(0,1)$. Pour ce faire, nous allons introduire une nouvelle bibliothèque :

```
import scipy.stats as sc
```

Cette bibliothèque va nous donner deux outils essentiels dans les calculs concernant le loi normale, qui sont les $\Phi(x)$ ainsi que les $\Phi^{-1}(u)$.

```
# cumulative distribution fonction (cdf) :
u=sc.norm.cdf(x)     # donne u=Phi(x)

#  percent point function (ppf) :
x=sc.norm.ppf(u)     # donne x=Phi^(-1)(u)

# aller dans l'aide des fonctions pour voir plus d'
    options, notamment pour déterminier directement
     le cas des lois quelconques N(mu,sigma2).
```

L'exercice ci-dessous va nous permettre de tester tout ceci. Rendez-vous dans le fichier python téléchargé afin de commencer.

Exercice 1:

1. Vérifier avec python que $\Phi(0) = \frac{1}{2}$.
2. Vérifier que :
   a) l'inverse de la fonction de répartition en 0.5 vaut 0
   b) l'inverse de la fonction de répartition en 0.95 vaut 1.64485...
3. On considère une variable Y qui suit une loi normale de moyenne 2 et de variance 3.
   a) Vérifier que $P(Y \leqslant 3) \simeq 0,72$
   b) Déterminer y tel que $P(Y \leqslant y) = 0.99$.
   (Après avoir fait votre calcul – et seulement après calcul ! – pour vérification, vous devez obtenir environ $6,03$.*)*
   c) Déterminer a tel que $P(|Y-2| \leqslant a) = 0.95$.
   (Il faudra certainement faire d'abord une petite manipulation pour passer à une loi $\mathcal{N}(0;1)$ *! Après calcul, pour vérification, vous devez obtenir environ* $3,395$.*)*

Simulations de lois à densité

Pour chaque loi à densité vue en cours cette année, il existe une façon d'en simuler une variable aléatoire à partir du seul générateur "pseudo"-aléatoire `random`.

Exercice 2: La base : Les lois à densité uniformes

1. Modifier et compléter la première variable `X=...`de manière à ce que sa valeur soit le résultat d'une simulation de variable suivant une loi uniforme sur $[0,1[$. *Il n'y a pas encore de test pour cette partie !*
2. En déduire la façon de construire la fonction `simul_uniforme(a,b)` de votre fichier de manière à ce que le résultat simule le résultat d'une variable aléatoire suivant une loi uniforme sur l'intervalle **réel** $[a,b[$.
3. Testez votre fonction en exécutant la cellule ou au moins la partie *–test–* située juste après votre fonction `simul_uniforme(a,b)` dans votre fichier téléchargé.

Exercice 3: Introdcution : un peu de maths avant de continuer ! Simulation de variables à densité "presque" quelconques

On se donne X une variable aléatoire réelle de fonction de répartition F.
On suppose que :
- $I = Supp(X)$ est un intervalle non nécessairement égal à $\mathbb{R}$ tout entier.
- $F : I \mapsto]0,1[$ est inversible (autrement dit, strictement croissante sur I).
- U est une variable telle que $U \hookrightarrow \mathcal{U}_{]0,1[}$.

On pose alors

$$Y = F^{-1}(U).$$

(appelée fonction quantile.)

1. Montrer alors que $F_Y(x) = F(x) \quad \forall x \in I$.

2. En déduire que $\mathcal{L}(X) = \mathcal{L}(Y)$, c'est-à-dire que X et Y ont la même loi.

> *Ce qui signifie que si F est la fonction de répartition de X,*
>
> *X peut être remplacée par $F^{-1}(U)$.*
>
> *et donc, si nécessaire, qu'on peut simuler X en simulant la variable $F^{-1}(U)$ à la place.*

Exercice 4: Simuler une loi exponentielle :

Supposons que U suive une loi uniforme sur $[0, 1[$.

1. Soit $\lambda > 0$. Montrer que $X = -\frac{\ln(1-U)}{\lambda}$ suit une loi exponentielle de paramètre λ. *(On pourra utiliser l'exercice précédent pour y parvenir ou le faire directement, la méthode étant à connaître pour le concours.)*

2. **a)** En déduire une façon de simuler une variable aléatoire suivant une loi exponentielle de paramètre λ. (On remplira pour ce faire la fonction `simul_expo(lam)` où `lam` est le paramètre λ de la loi exponentielle.)

b) Vérifier votre fonction grâce à la partie *–test–* située juste après votre fonction.

Exercice 5: Simuler une loi normale :

1. Simuler une loi normale centrée réduite $X \hookrightarrow \mathcal{N}(0, 1)$:
On note Φ la fonction de répartition de X.
D'après l'exercice 2, la fonction Φ étant inversible sur $\mathbb{R} = Supp(X)$, on sait que si $U \hookrightarrow \mathcal{U}_{[0,1[}$, on sait que

$$\mathcal{L}(X) = \mathcal{L}(\Phi^{-1}(U)) = \mathcal{N}(0, 1)$$

Si on a accès à Φ^{-1}, on va donc pouvoir simuler X. Pour ce faire, on rappelle que l'on dispose de

```
import scipy.stats as sc

# cumulative distribution fonction (cdf) :
sc.norm.cdf(x)  # donne u=Phi(x)

#  percent point function (ppf) :
sc.norm.ppf(u)  # donne x=Phi^(-1)(u)
```

2. En déduire une méthode pour simuler une variable X_0 qui suit une loi normale centrée réduite. (vérifier grâce à la partie *–test–*)

3. Simuler une loi normale quelconque $X \hookrightarrow \mathcal{N}(\mu, \sigma^2)$:
Connaissant la méthode de simulation ci-dessus pour $X_0 \hookrightarrow \mathcal{N}(0, 1)$, en déduire une méthode de simulation pour $X \hookrightarrow \mathcal{N}(\mu, \sigma^2)$.

Remarque : Python sait faire toutes les simulations précédentes à l'aide de sa bibliothèque ***numpy.random****. Voici un petit résumé de ses possibilités (que vous pourrez également utiliser si besoin au concours, mais il faut savoir faire tout ceci à la main si c'est demandé.)*

```
import numpy as np

np.random.rand()    # simule X qui suit une loi U_
    [0,1]
np.random.exponential(lam)  # simule Y qui suit une
     loi exponentielle de paramètre lam
np.random.normal()          # simule Z qui suit une
     loi N(0,1)
np.random.normal(mu,np.sqrt(sigma2))    # simule T
    qui suit une loi N(mu,sigma2)
```

La bilbiothèque `numpy.random` a également la possibilité de générer directement un tableau de nombres aléatoires. Par exemple, si on veut générer 10 simulations d'un seul coup, on peut faire :

```
np.random.rand(10)  # simule 10 fois la variable X
np.random.exponential(lam,10)   # simule 10 fois la
     variable Y
np.random.normal(mu,np.sqrt(sigma2),10) # simule 10
     fois la variable T
```

Attention, pour simuler la variable $Z \hookrightarrow N(0, 1)$, il faut passer par le rappel des paramètres $(0, 1)$:

```
np.random.normal(0,1,10)    # simule 10 fois la
    variable Z
```

En réalité, on peut également simuler les variables discrètes avec numpy.random !

Ci-dessous les méthodes correspondant à nos exemples fondamentaux de cours :

Loi :	Commande Python	
	Simuler une fois	*Simuler N fois*
$\mathcal{B}(n,p)$	`binomial(n,p)`	`binomial(n,p,N)`
$\mathcal{G}(p)$	`geometric(p)`	`geometric(p;N)`
$\mathcal{P}(\lambda)$	`poisson(lam)`	`poisson(lam,N)`
$\mathcal{E}(\lambda)$	`exponential(lam)`	`exponential(lam,N)`

⋆ *La loi géométrique* est bien la loi géométrique sur $\mathbb{N}^*$. Si on veut celle sur $\mathbb{N}$, il faut retrancher 1 !

⋆ *La taille N* peut être donnée en deux dimensions par exemple par `N=(2,3)`. On obtient alors un résultat sous la forme d'un tableau à 2 lignes et 3 colonnes.

Bonus : Convergence en loi *(pour ceux qui sont à l'aise)*

On veut ici obsesrver le fait qu'une suite de variables aléatoires (X_n) converge en loi vers une autre variable X pour les lois vues en cours.
On pourrait écrire "à la main" un programme permettant de construire les fonctions de répartition (plus ou moins simplement) de la plupart des variables de cours, mais la bibliothèque `scipy.stats` nous permet d'avoir accès à ces informations directement :

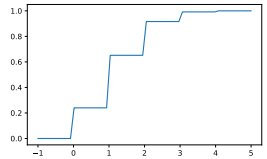
Loi :	Commande Python
	$F_X(x)$ *pour* $x \in Supp(X)$
$X \hookrightarrow \mathcal{B}(n,p)$	`sc.binom.cdf(x,n,p)`
$X \hookrightarrow \mathcal{G}(p)$	`sc.geom.cdf(x,p)`
$X \hookrightarrow \mathcal{P}(\lambda)$	`sc.poisson.cdf(x,lambda)`
$X \hookrightarrow \mathcal{E}(\lambda)$	`1-np.exp(-lambda*x)`
$X \hookrightarrow \mathcal{N}(\mu;\sigma^2)$	`sc.norm.cdf(x,mu,sigma)`

Remarque :
Vous noter qu'en effet on n'a pas mis de commande Python pour la loi exponentielle, car la formule de la foncton de répartition est tout aussi simple et qu'on doit la connaître quoi qu'il en soit...

Exercice 6: Les graphiques des fonctions de répartition

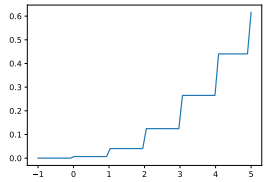
1. **a)** Écrire une fonction `Fbinom(a,b,n,p)` qui construit le graphique de la fonction de répartition d'une variable $X_n \hookrightarrow \mathcal{B}(n,p)$ sur l'intervalle $[a,b]$

b) Faire un essai avec $a=-1,\ b=5,\ n=4,\ p=0.3$ et observer que l'on obtient :



2. **a)** Écrire une fonction `FPoisson(a,b,lamb)` qui qui construit le graphique de la fonction de répartition d'une variable $X \hookrightarrow \mathcal{P}(\lambda)$ sur l'intervalle $[a,b]$.

b) Faire un essai avec $a=-1,\ b=5,\ \lambda=5$ et observer que l'on obtient :



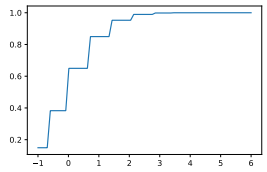
3. On note $T_n^* = \frac{\frac{T_n}{n}-p}{\frac{\sqrt{p(1-p)}}{\sqrt{n}}}$ où $T_n \hookrightarrow \mathcal{B}(n,p)$. On note F^* sa fonction de répartition et F la fonction de répartition de T_n.

a) Montrer que pour tout $x \in \mathbb{R}$, on a

$$F^*(x) = F\left(np + x\sqrt{np(1-p)}\right)$$

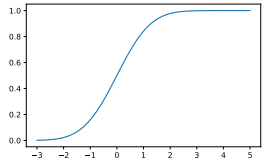
b) En déduire une fonction Python `FbinomCentreReduite(a,b,n,p)` qui qui construit le graphique de la fonction de répartition de T_n^* sur l'intervalle $[a,b]$

c) Faire un essai avec $a=-1,\ b=6,\ n=10,\ p=0.3$ et observer que l'on obtient :



4. **a)** Écrire une fonction `Fnormale(a,b)` qui qui construit le graphique de la fonction de répartition d'une variable $X \hookrightarrow \mathcal{N}(0;1)$ sur l'intervalle $[a,b]$.

b) Faire un essai avec $a=-3,\ b=5$ et observer que l'on obtient :



Exercice 7: Observer les convergences du cours

Soit `L=[5,10,30,50,100,200]` une liste de plusieurs "n".

1. On se donne une suite X_n de variables telles que $X_n \hookrightarrow \mathcal{B}(n, \frac{\lambda}{n})$.

a) Écrire une fonction `Convergence_BP(lamb,L)` représentant :

- l'ensemble des fonctions de répartions des X_n pour $n \in L$ sur l'intervalle $[0, 2\lambda]$, en n'affichant le résultat que s'il a un sens, c'est-à-dire $p = \frac{\lambda}{n} \leqslant 1$.
- la fonction de répartition de $X \hookrightarrow P(\lambda)$ sur l'intervalle $[0, 2\lambda]$
- une légende permettant d'identifier les courbes

(On n'hésitera pas à utiliser les fonctions précédemment définies, quitte à les modifier un peu.)

b) Testez avec plusieurs les valeurs de λ suivantes : $1, 5, 10, 30$.
Observez qu'il y a bien convergence en loi de (X_n) vers X et que, dans tous les cas, pour que l'approximation soit visuellement bonne, il faut au minimum que $n \geqslant 30$ et $p = \frac{\lambda}{n} \leqslant 0.1$

2. On se donne une suite X_n de variables telles que $X_n \hookrightarrow \mathcal{B}(n,p)$.

a) Écrire une fonction `Convergence_BN(p,L)` représentant :

- l'ensemble des fonctions de répartions des $\overline{X_n}^{\,*}$ pour $n \in L$ sur l'intervalle $[-4, 4]$
- la fonction de répartition de $X \hookrightarrow N(0;1)$ sur l'intervalle $[0, 4]$
- une légende permettant d'identifier les courbes

b) Testez avec plusieurs les valeurs de p suivantes : $0.1, 0.5, 0.9$.
Observez qu'il y a bien convergence en loi de $(\overline{X_n})$ vers X et que, dans tous les cas, pour que l'approximation soit visuellement bonne, il faut au minimum que $n \geqslant 30$, $np \geqslant 5$ et $n(1-p) \geqslant 5$.

3. Faire de même avec cette fois-ci les variables centrée réduites de $T_n \hookrightarrow P(n\lambda)$ et observez là aussi la convergence en loi et la nécessité des conditions énumérées en cours.