TP6 : Simulation de lois via la bibliothèque `numpy.random`

▶ Dans votre dossier `Info_2a`, créez le dossier `TP_6`.

I. Utilisation des fonctions de `numpy.random`

Dans ce TP, nous délaissons la fonction `rd.random` pour utiliser les fonctions de la bibliothèque `numpy.random` qui présentent l'avantage de pouvoir simuler très simplement l'observation d'un échantillon de `N` v.a.r. indépendantes.
On commencera donc par importer cette bibliothèque.

```
1  import numpy.random as nr
```

I.1. Simulation d'une v.a.r. suivant une loi géométrique

I.1.a) Utilisation de la fonction `nr.geometric`

▶ Évaluer plusieurs fois la commande `nr.geometric(0.4)`. Qu'obtient-on ?

- On obtient la séquence de valeurs suivantes : 4, 1, 5, 4, 1, 7, 5, 1 ...
- La commande `nr.geometric(0.4)` permet de simuler une v.a.r. X suivant la loi géométrique de paramètre 0.4. On rappelle que si $X \hookrightarrow \mathcal{G}(p)$, on a :
  - $X(\Omega) = \mathbb{N}^*$
  - $\forall k \in \mathbb{N}^*,\ \mathbb{P}(X = k) = (1-p)^{k-1} \times p$

▶ Évaluer maintenant la commande `nr.geometric(0.4,10)`. Qu'obtient-on ?

- On obtient le tableau (`array`) de longueur 10 suivant : $[2, 1, 1, 2, 1, 1, 2, 10, 7, 12]$.
- La commande `nr.geometric(0.4,10)` permet de simuler un échantillon de 10 v.a.r. indépendantes X_i suivant toutes la loi géométrique de paramètre 0.4.

▶ Évaluer maintenant la commande `[nr.geometric(0.4,4) for k in range(2)]`. Qu'obtient-on ?

- On obtient la liste à 2 éléments suivante : $\Big[[7,\ 1,\ 7,\ 7],\ [7,\ 3,\ 8,\ 1]\Big]$.
- La commande `[nr.geometric(0.4,4) for k in range(2)]` permet de simuler 2 échantillons de 4 v.a.r. indépendantes X_i suivant toutes la loi géométrique de paramètre 0.4.

▶ Généraliser ce résultat.

La commande `[nr.geometric(p,N) for k in range(m)]` permet de simuler `m` échantillons de `N` v.a.r. indépendantes X_i suivant toutes la loi géométrique de paramètre `p`.

- Écrire un programmme qui :
  - × demande à l'utilisateur d'entrer la valeur d'un paramètre `p`,
  - × demande à l'utilisateur d'entrer la valeur d'un paramètre `N`,
  - × réalise la simulation d'un échantillon de `N` v.a.r. indépendantes de loi géométrique $\mathcal{G}(\texttt{p})$, *(on stocke le résultat obtenu dans une variable* `Obs`*)*
  - × trace le diagramme des effectifs de cette simulation.

  On pourra utiliser les fonctions `tabul` (pour calculer les effectifs) et la fonction `bar` (pour représenter le digramme en bâtons correspondant).

```
import matplotlib.pyplot as plt
p = float(input(‘‘Entrez la valeur du paramètre p : ‘‘))
N = int(input(‘‘Entrez la valeur de N : ‘‘))

Obs = nr.geometric(p,N)
plt.hist(Obs)
```

  (on enregistre ce programme sous le nom `distrib_geom.py`*)*

- Ajouter `rwidth=0.8, color = ‘‘red‘‘, edgecolor = ‘‘black‘‘` dans l'appel à la fonction `plt.hist`. À quoi servent ces arguments optionnels ?

  - Le paramètre `rwidth` permet de préciser la largeur des bâtons.
  - Le paramètre `color` permet de préciser la couleur (ici rouge) des bâtons.
  - Le paramètre `edgecolor` permet de préciser la couleur (ici noire) du bord des bâtons.

- Comment modifier le programme précédent afin d'obtenir le diagramme des fréquences ?

  Il suffit de régler le paramètre `normed` (ou `density`) de la fonction `plt.hist`.

  On remplace l'appel à `plt.hist` par : `plt.hist(Obs, normed = True, color = ‘‘red‘‘, edgecolor = ‘‘black‘‘)`.

I.1.b) Comparaison avec la distribution théorique

On propose d'ajouter les lignes suivantes au programme précédent.

```
lx = [k for k in range(1,11)]
ly = [(1-p)**(k-1) * p for k in range(1,11)]

plt.step(lx, ly)
```

- Que contient la variable `y` ? Que cela représente-t-il pour une v.a.r. X telle que $X \hookrightarrow \mathcal{G}(p)$?

  La variable `y` contient ici `[p, (1-p)*p, (1-p)**2 * p, (1-p)**3 * p, ..., (1-p)**10 * p]`.

  Autrement dit : $[\mathbb{P}([X=1]), \mathbb{P}([X=2]), \mathbb{P}([X=3]), \ldots, \mathbb{P}([X=10])]$.

- Le choix fait pour la variable `x` vous semble-t-il pertinent ?
  Comment la modifier de sorte à représenter autant de bâtons que dans le diagramme précédent ?

  - L'ensemble image d'une v.a.r. X telle que $X \hookrightarrow \mathcal{G}(p)$ est : $X(\Omega) = \mathbb{N}^*$.
    Comme on ne peut représenter toutes les valeurs de $\mathbb{P}([X = k])$ pour k dans $\mathbb{N}^*$, on fait ici le choix de s'arrêter à $k = 10$. L'idée est que `(1-p)**k` devient rapidement faible.
    Évidemment, ceci dépend du choix de `p`.
  - Pour dessiner le même nombre de bâtons que dans l'appel à `plt.hist` il faut trouver l'abscisse la plus élevée des bâtons précédents. On peut procéder comme suit :

```
m = max(Obs) # abscisse maximale
lx = [k for k in range(1,m+1)]
```

- Comment s'affiche les deux diagrammes obtenus ? Comment améliorer cet affichage ?

  - Les bâtons du diagramme ne sont pas centrés sur les valeurs entières et la première marche de la fonction en escalier n'est pas visible.
  - Pour améliorer la lisibilité, on peut forcer l'affichage des deux points précédents de la façon suivante :

```
m = max(Obs)
x = [k for k in range(m+1)]
plt.hist(Obs, normed = True, bins = x, color = ‘‘red‘‘,
         edgecolor = ‘‘black‘‘)

lx = [k for k in range(m+2)]
ly = [p] + [(1-p)**(k-1) * p for k in range(1, m+1)]
plt.step(lx, ly)
```

- On pourra enfin ajouter un titre et une légende à ce diagramme à l'aide des commandes suivantes.

```
plt.hist(Obs, normed = True, bins = x, color = ‘‘red‘‘,
         edgecolor = ‘‘black‘‘, hatch = ‘‘/‘‘,
         label = ‘‘Diagramme distribution approchée‘‘)
plt.step(lx, ly, label = ‘‘Diagramme distribution théorique‘‘)
plt.legend()
plt.title(‘‘Comparaison de la distribution théorique et de la distribution‘‘
         + ‘‘approchée pour une var suivant une loi géométrique‘‘)
```

I.2. Simulation d'une v.a.r. suivant une loi de Poisson

I.2.a) Utilisation de la fonction `grand`

▶ Si `N` et `lam` sont donnés, à quoi sert l'appel `nr.poisson(lam, N)` ?

> La commande `nr.poisson(lam, N)` permet de simuler un échantillon de `N` v.a.r. indépendantes X_i suivant toutes la loi de Poisson de paramètre `lam`.

▶ Quel commande permet d'obtenir la simulation de `m` échantillons de `N` v.a.r. indépendantes X_i suivant toutes la loi de Poisson de paramètre `lam` ?

> Il s'agit de la commande : `[nr.poisson(lam, N) for k in range(m)]`.

I.2.b) Comparaison avec la distribution théorique

▶ Soit X une v.a.r. . Que signifie que $X \hookrightarrow \mathcal{P}(\lambda)$? Détailler.

> - $X(\Omega) = \mathbb{N}$
> - $\forall k \in \mathbb{N},\ \mathbb{P}(X = k) = \mathrm{e}^{-\lambda} \times \dfrac{\lambda^k}{k!}$

▶ Dans un nouvel onglet **Python**, copier le programme `distrib_geom.py` et l'adapter au cas de la loi de Poisson. Pour la distribution théorique, on pourra se servir de la fonction `np.math.factorial`.

```
import numpy as np
# Valeur des paramètres
lam = float(input(‘‘Entrez la valeur du paramètre lambda : ‘‘))
N = int(input(# Entrez la valeur de N : ))
# Valeurs observées
Obs = nr.poisson(lam, N)
# Distribution empirique
m = max(Obs)
x = [k for k in range(m+1)]
plt.hist(Obs, normed = True, bins = x, color = ‘‘red‘‘,
        edgecolor = ‘‘black‘‘, hatch = ‘‘/‘‘,
        label = ‘‘Diagramme distribution approchée‘‘)
# Distribution théorique
lx = [k for k in range(m+2)]
ly = [np.exp(-lam)] + [np.exp(-lam) * lam**k * np.math.factorial(k)
            for k in range(m+1)]
plt.step(lx, ly, label = ‘‘Diagramme distribution théorique‘‘)
# Paramètres d'affichage
plt.legend()
plt.title(‘‘Comparaison de la distribution théorique et de la distribution‘‘
        + ‘‘approchée pour une var suivant une loi de Poisson‘‘)
```

(on enregistre ce programme sous le nom `distrib_poisson.py`*)*

II. Bilan sur la bibliothèque `numpy.random`

▶ Chercher de l'aide en ligne sur la bibliothèque `numpy.random` sur le site numpy.org et prendre connaissance des informations contenues dans la section `Random sampling`. Compléter alors le texte ci-dessous.

`nr.random (N)` génère un échantillon de `N` variables aléatoires suivant la loi $\mathcal{U}([0,1[)$.

`rd.randint(a,b,N)` génère un échantillon de `N` variables aléatoires suivant la loi $\mathcal{U}([\![a,b]\!])$.

`nr.binomiale (n,p,N)` génère un échantillon de `N` v.a.r. suivant la loi $\mathcal{B}(\texttt{n},\texttt{p})$.

`nr.geometric(p,N)` génère un échantillon de `N` v.a.r. suivant la loi $\mathcal{G}(p)$.

`nr.poisson (lam,N)` génère un échantillon de `N` v.a.r. suivant la loi $\mathcal{P}(\texttt{lam})$.

`nr.exponential(alpha,N)` génère un échantillon de `N` v.a.r. suivant la loi $\mathcal{E}(\texttt{alpha})$.

II.1. Simulation d'une v.a.r. suivant une loi normale

II.1.a) Utilisation de la fonction `grand`

▶ Si `N`, `esp` et `sigma` sont donnés, à quoi sert l'appel `nr.normal(esp,sigma,N)` ?

La commande `nr.normal(esp,sigma,N)` permet de simuler un échantillons de `N` v.a.r. indépendantes X_i suivant toutes la loi normale de paramètres $(\texttt{esp},\texttt{sigma}^2)$.

II.1.b) Comparaison avec la distribution théorique

▶ Soit X une v.a.r. . Que signifie que $X \hookrightarrow \mathcal{N}(0,1)$? Détailler.

- $X(\Omega) =]-\infty,+\infty[$
- X admet pour densité la fonction φ définie par :

$$\varphi \ : \left| \begin{array}{ccc} \mathbb{R} & \to & \mathbb{R} \\ x & \mapsto & \dfrac{1}{\sqrt{2\pi}}\,\mathrm{e}^{-\frac{x^2}{2}} \end{array} \right.$$

▶ Écrire la fonction `densiteNormaleCR`, densité de la loi normale centrée réduite.

```
def densiteNormaleCR(x) :
    y = 1 / sqrt(2 * np.pi) * np.exp(-x**2 / 2)
    return y
```

- On considère un vecteur `Obs` contenant `N` valeurs comprises entre -5 et 5.
  On souhaite, grâce à `plt.hist`, générer le diagramme des fréquences de `Obs`. Comment le générer de sorte qu'il contienne 100 bâtons dont le premier commence en -5 et le dernier finit en 5 ?

  > On souhaite 100 écarts (pour définir 100 classes). Il faut donc que la liste définissant les classes contiennent 101 éléments. Pour cela, on règle le paramètre `bins` pour qu'il contienne : `x = np.linspace(-5,5,101)`.

- Dans un nouvel onglet **Python**, adapter les programmes précédents au cas de la loi normale centrée réduite.

```
# Valeur des paramètres
N = int(input(“Entrez la valeur de N : “))

# Valeurs observées
Obs = nr.normal(0, 1, N)

# Tracé des diagrammes
x = linspace(-5, 5, 101)
plt.hist(Obs, normed = True, bins = x, edgecolor = “black“,
         label = “Distribution approchée“)

lx = np.linspace(-5, 5)
ly = [densiteNormaleCR(x) for x in lx]
plt.plot(lx, ly, color = “red“, label = “Densité“)

plt.title(“Loi normale centrée réduite : comparaison de la“
       + “distribution approchée et de la fonction densité“)
plt.legend()
```