Année Universitaire 2020/2021

Section	1 ING IDSD
Matière	TP Théorie de l'information et de l'incertain
Enseignante	Trabelsi Nessrine

TP 3: Simulation de variables aléatoires & estimation non paramétrique

Objectifs du TP:

- Simulation de variables aléatoires suivant des lois discrètes et continues
- Affichage des histogrammes et fonctions de répartition (cdf) et densité (pdf)
- Estimation non paramétrique par histogramme et noyaux

Exercice 1

1. Ouvrir le fichier exercice1.py avec votre éditeur python.
2. En utilisant "numpy", générer un échantillon **X** (c'est à dire un ensemble de réels) de taille 1000 suivant la loi continue normale standard (moyenne=0, écart type=1).
3. Dresser l'histogramme d'un échantillon consiste à découper les réels en sous-intervalles, puis d'afficher des bâtons qui ont pour base ces sous-intervalles, et comme hauteur le nombre d'éléments (ou bien la probabilité de densité) de l'échantillon contenu dans chaque sous-intervalles.
    a. En utilisant "pyplot", afficher l'histogramme de **X** en utilisant les paramètres par défaut.
    b. Modifier l'histogramme précédent en ajoutant le paramètre edgecolor="black". Qu'apporte cette modification à l'affichage de l'histogramme?
    c. Dans une autre figure, diviser la fenêtre d'affichage en 8 partie (4 lignes et 2 colonnes) et afficher l'histogramme de X en utilisant ces paramètres:
        i. edgecolor="black" et un label adéquat pour chaque histogramme
        ii. Hist 1: Histogramme par défaut déjà donné, Hist 2: bins =5
        Hist 3: bins = 30, Hist 4: bins=[-4, -2, -1, -0.5, 0, 2, 3,4]
        Hist 5: rwidth=0.7, Hist 6: density=True
        Hist 7: histtype='step' , Hist 8: cumulative=True, density=True
    d. Comparer entre les 6 histogrammes affichés dans c.
4. En utilisant "numpy", générer un échantillon **Y** de taille 1000 suivant la loi discrète binomiale (n=10, p=0.5).

5. Pour les lois discrètes, avant d'afficher l'histogramme, il faut préciser un découpage adéquat pour le paramètre bins.
   A l'aide de "numpy.arange", créer la variable **intervals** de façon que chaque entier x (entre 0 et 10) soit représenté par une barre de largeur (x-0.5, x+0.5)
   → résultat à obtenir : **intervals**=[-0.5 0.5 1.5 2.5 3.5 4.5 5.5 6.5 7.5 8.5 9.5 10.5]
6. Afficher l'histogramme de **Y** en utilisant le découpage avec la variable **intervals,** color='red' et 'rwidth=red'.

La bibliothèque "scipy.stats" contient plus de lois que "numpy.random". Nous l'utiliserons dans la suite.

7. Générer des échantillons de taille égale à 10000 chacun suivant des lois de probabilité continues :
   i. X1 : loi normale
   ii. X2 : loi uniforme
   iii. X3 : loi triangulaire (c=0.5)
   iv. X4: loi exponentielle
8. Pour chaque loi, générer sa fonction de répartition (cdf) et de densité (pdf) sur les intervalles donnés sur le code (**abs1** pour loi normale, **abs2** pour loi uniforme, etc.)
9. Diviser une figure en 4 parties (2 lignes et 2 colonnes) et afficher dans chaque partie l'histogramme (paramètres density=True, edgecolor='black' et label adéquat), la fonction de répartition et densité d'un échantillon **Xi**. Mettre également un titre pour chaque partie.
10. Ajuster les valeurs de bins pour chaque histogramme, afin qu'il coïncide le mieux avec la fonction pdf.

Exercice 2

Annexes

1. Numpy.
   - **Numpy.random.normal** : génère des valeurs suivant la distribution normale (loi normale/gaussienne)
     - *Parameters*
       - **loc** : float or array_like of floats , Mean ("centre") of the distribution.
       - **scale** : float or array_like of floats, Standard deviation (spread or "width") of the distribution. Must be non-negative.
       - **size** :int or tuple of ints, optional
     - *Example:* numpy.random.normal(0,2,100) , numpy.random.normal(size=200)
   - **Numpy.random.binomial** : génère des valeurs suivant la distribution binomiale
     - *Parameters*
       - **n** : int or array_like of ints, Parameter of the distribution, >= 0

        - **p** : float or array_like of floats, Parameter of the distribution, >= 0 and <=1.
        - **size** : int or tuple of ints, optional, Output shape.
    - *Example:* numpy.randon.binomial(5,0.3,100)
- **Numpy.arange** : Return evenly spaced values within a given interval (retourne un tableau contenant des valeurs espacés d'un pas unique)
    - *Syntax*: numpy.arange([start,]stop, [step,]dtype=None, *, like=None)
    - *Parameters*
        - **start** : integer or real, optional, Start of interval. The interval includes this value. The default start value is 0.
        - **stop** : integer or real, End of interval. The interval does not include this value, except in some cases.
        - **step** : integer or real, optional, Spacing between values.The default step size is 1.
    - *Example*: numpy.arange(0,6,2) → retourne [0 2 4]

2. Matplotlib.pyplot
    - **Matplotlib.pyplot.hist** : Compute and draw the histogram of x.
        - *Syntax*: matplotlib.pyplot.hist(x, bins=None, range=None, density=False, weights=None, cumulative=False, bottom=None, histtype='bar', align='mid', orientation='vertical', rwidth=None, log=False, color=None, label=None, stacked=False, *, data=None, **kwargs)
        - *Parameters* (explication de certains paramètres ci-après)
            - **x** : (n,) array or sequence of (n,) arrays
            - **bins** : int or sequence or str, (default: 10)
              If bins is an integer, it defines the number of equal-width bins in the range.
              If bins is a sequence, it defines the bin edges, including the left edge of the first bin and the right edge of the last bin; in this case, bins may be unequally spaced. All but the last (righthand-most) bin is half-open.
            - **density** : bool, default: False. If True, draw and return a probability density: each bin will display the bin's raw count divided by the total number of counts and the bin width.
            - **cumulative** : bool or -1, default: False
              If True, then a histogram is computed where each bin gives the counts in that bin plus all bins for smaller values. The last bin gives the total number of datapoints.
            - **histtype** : {'bar', 'barstacked', 'step', 'stepfilled'}, default: 'bar'
              The type of histogram to draw.
            - **rwidth** : float or None, default: None
              The relative width of the bars as a fraction of the bin width. If None, automatically compute the width.
              Ignored if histtype is 'step' or 'stepfilled'.
            - **color** : color or array-like of colors or None, default: None

Color or sequence of colors, one per dataset. Default (None) uses the standard line color sequence.

- **label** : str or None, default: None
  String, or sequence of strings to match multiple datasets.

- **Matplotlib.pyplot.subplot** : Add an Axes to the current figure or retrieve an existing Axes.
  - *Syntax*: matplotlib.pyplot.subplot(nrows, ncols, index, **kwargs)
- **Matplotlib.pyplot.plot** : Pour afficher la courbe d'une fonction, par exemple une pdf : Matplotlib.pyplot.plot(x,pdf, label="pdf")

3. Scipy.stats
   - **scipy.stats.norm.rvs** : simulation d'un échantillon de variable aléatoire ayant la loi normale.
     - *Syntax*: scipy.stats.norm.rvs(loc=0, scale=1, size=1, random_state=None)
   - **scipy.stats.norm.pdf** :Probability density function. -> fonction densité de probabilité (elle prend des réels en argument)
     - *Syntax*: scipy.stats.norm.pdf(x, loc=0, scale=1)
   - **Scipy.stats.norm.cdf** :Cumulative distribution function. -> fonction de répartition
     - *Syntax*: scipy.stats.norm.cdf(x, loc=0, scale=1)
   - Quelques autres exemples de lois continues disponibles dans "scipy.stats":
     - uniform: A uniform continuous random variable.
       In the standard form, the distribution is uniform on [0, 1]. Using the parameters loc and scale, one obtains the uniform distribution on [loc, loc + scale].
       - rvs(loc=0, scale=1, size=1, random_state=None)
       - pdf(x, loc=0, scale=1)
       - cdf(x, loc=0, scale=1)
     - triang: A triangular continuous random variable. The triangular distribution can be represented with an up-sloping line from loc to (loc + c*scale) and then downsloping for (loc + c*scale) to (loc + scale).
       - rvs(c, loc=0, scale=1, size=1, random_state=None)
       - pdf(x, c, loc=0, scale=1)
       - cdf(x, c, loc=0, scale=1)
     - expon: An exponential continuous random variable.
       - rvs(loc=0, scale=1, size=1, random_state=None)
       - pdf(x, loc=0, scale=1)
       - cdf(x, loc=0, scale=1)
     - alpha: An alpha continuous random variable, takes "a" as a shape parameter.
       - rvs(a, loc=0, scale=1, size=1, random_state=None)
       - pdf(x, a, loc=0, scale=1)
       - cdf(x, a, loc=0, scale=1)

- beta: A beta continuous random variable, takes “a” and “b” as shape parameters.
    - rvs(a, b, loc=0, scale=1, size=1, random_state=None)
    - pdf(x, a, b, loc=0, scale=1)
    - cdf(x, a, b, loc=0, scale=1)