SIMULATIONS D'EXPÉRIENCES ALÉATOIRES CONTINUES

Les parties marquées d'un ⛰, ne doivent être abordées que si vous avez de l'avance.

Objectifs, motivations et premier pas

Le but de ce TP est de représenter les fonctions de densité et de répartitions pour les lois classiques. Dans un second temps nous utilisons les résultats de mathématiques pour simuler des loi classiques

Pour chacune des trois lois classiques, uniforme, exponentielle, normale, nous allons effectuer les taches suivantes.

1. Utiliser les résultats mathématiques pour simuler cette loi en utilisant uniquement `rrandom()` qui simule une loi uniforme sur [0; 1[
2. simuler un grand nombre d'expériences aléatoires en utilisant
3. Tracer du graphe des fréquences sous forme d'histogramme en utilisant `hist` du module matplotlib.pyplot
4. Tracer la fonction de densité théorique en utilisant `plot`.
5. Tracer l'histogramme lié à la fonction de répartition.
6. Tracer la fonction de théorique de répartition en utilisant `plot`.

Remarque : Les fonctions du module `numpy.random` permettent de simuler un grand noobre de variables aléatoires. Les noms des fonctions utilisées sont les noms anglais des loi usuelles. Les paramètres peuvent être différents de ceux utilisés en mathématiques. Il ne faut pas hésiter à chercher de l'aide sur ces fonctions.

À programmer 1 (Modules)**.**
Importer les modules

- `numpy.random` avec l'alias `rd`
- `numpy` avec l'alias `np`
- `matlotlib.pyplot` avec l'alias `plt`

I Loi Uniforme

I.1 Un exemple la loi : $\mathscr{U}([0;1[)$

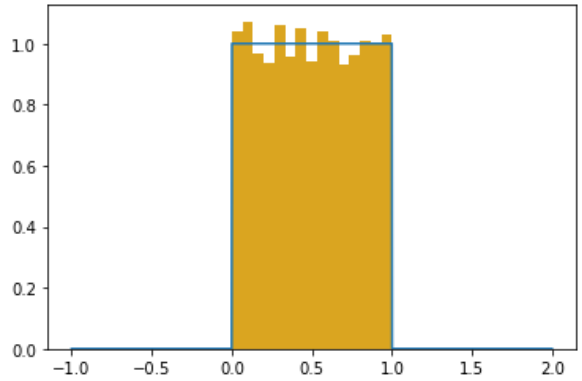
À programmer 2 (Recopier et tester)**.**

```
def uniforme_densite(x):
        if x<0:
                return 0
        elif 0<x<1:
                return 1
        else:
                return 0

X=np.arange(-1,2,0.001)
Y=[uniforme_densite(x) for x in X]
plt.plot(X,Y)

D=rd.random(10000)
plt.hist(D,color='goldenrod',density=True,cumulative=False,bins=15)
plt.show()
```

Vous devez obtenir



On commence par définir une fonction de densité de la loi uniforme sur [0; 1[, puis par afficher son graphe.

Dans un deuxième temps on simule un grand nombre de fois une loi uniforme sur [0; 1[, et on affiche un histogramme, les options sont

`D` est une série de valeurs

`bins` nombre de classes et donc de barres de l'histogramme. Avec cette syntaxe, l'étendu des données est séparées en `bins` classes de même largeur.

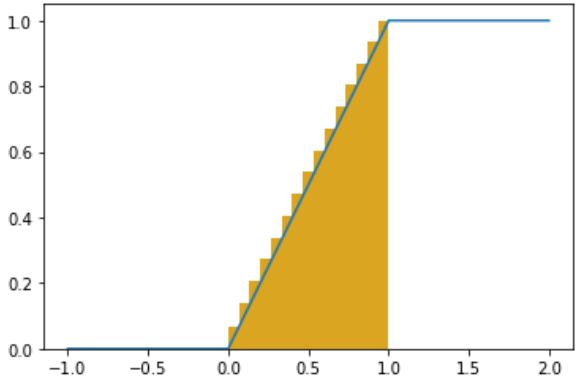
`density` Permet d'obtenir les fréquences , par opposition aux effectifs. Si `density=True` est choisie la hauteur de la barre i est donnée par nombre de valeur dans la classe i/effectif total.

`Cumulative` permet de choisir entre un graphe de distribution (Probability Density Function) ou de fonction de répartition (*cumulative distribution function*). Si les options `cumulative` **et** `density` sont activées la hauteur de la $i$ème barre est donnée par nombre de valeur dans les classes 1,2, …, i/effectif total.

À programmer 3.
En utilisant les explications précédentes copier le script précédent et le modifier pour

- Réaliser un grand nombre de simulation d'une loi $\mathscr{U}([0;1[)$.
- Créer une fonction `uniforme_repartition(x)` qui pour tout réel x renvoie la valeur de la fonction de répartition de la loi $[0;1[$.
- Afficher l'histogramme des fréquences cumulatives et l'allure de la fonction de répartition.

Vous devez obtenir



I.2 Simuler une loi $\mathscr{U}([a;b[)$

Sur papier 1 (Résultat mathématique)**.**
Soit α un réel strictement positif et β un réel. Soit $X \hookrightarrow \mathscr{U}([0;1[)$ on pose $Y = (b-a)X + a$ Montrer que $Y \hookrightarrow \mathscr{U}([a;b[)$.

Nous allons donc, pour simuler une variable aléatoire suivant loi $\mathscr{U}([a;b[)$

- Choisir un nombre au hasard uniformément dans $[0;1[$.
- Lui appliquer la fonction $x \mapsto (b-a)x + a$.

À programmer 4 (Simulation)**.**
Écrire une fonction `uniforme(a,b,n)` qui renvoie une liste de n valeurs simulées d'une variable aléatoire suivant la loi $\mathscr{U}([a;b[)$.

À programmer 5 (Représentation d'une densité)**.**

1. Écrire une fonction `uniforme_densite(x,a=0,b=1)` qui renvoie la valeur de la densité choisie.
2. En utilisant la fonction `uniforme(a,b,n)` simuler un grand nombre de fois une variable suivant la loi $\mathscr{U}([a;b[)$
3. Sur un même graphique faire afficher l'allure de la densité et l'histogramme des fréquences

À programmer 6 (Représentation de la fonction de répartition)**.**

1. Écrire une fonction `repartition_densite(x,a=0,b=1)` qui renvoie la valeur de la fonction de répartition.
2. En utilisant la fonction `uniforme(a,b,n)` simuler un grand nombre de fois une variable suivant la loi $\mathscr{U}([a;b[)$
3. Sur un même graphique faire afficher l'allure de la fonction de répartition et l'histogramme des fréquences cumulatives.

I.3 Loi uniforme sur $[\![1,n]\!]$.

À programmer 7 (Approche mathématique)**.**
Soit p un entier naturel non nul et $X \hookrightarrow \mathscr{U}([1;p+1[)$, on pose $Y = \lfloor X \rfloor$.
Montrer que $Y \hookrightarrow \mathscr{U}([\![1,p]\!])$.

À programmer 8 (Implémentation)**.**
En utilisant la fonction `uniforme(n,a,b)`, écrire une fonction `unifentier(n,p)` qui renvoie une liste de longueur n dont chaque terme est choisi selon loi $\mathscr{U}([\![1,p]\!])$.

II Loi exponentielle

II.1 Loi exponentielle par transfert

Sur papier 2 (Calculs)**.**
Soit $X \hookrightarrow \mathscr{U}([0;1[)$ et soit $\lambda > 0$. On pose $Y = -\frac{1}{\lambda}\ln(1-X)$. Montrer que $Y \hookrightarrow \mathscr{E}(\lambda)$.

Nous allons donc, pour simuler une variable aléatoire suivant loi $\mathscr{E}(\lambda)$

- Choisir un nombre au hasard uniformément dans $[0;1[$.
- Lui appliquer la fonction $x \mapsto -\frac{1}{\lambda}\ln(1-x)$.

À programmer 9 (Simulation)**.**
En utilisant la fonction `rd.random()` écrire une fonction `exponentielle(lamb,n)` qui renvoie une liste de longueur n dont chaque terme est choisi selon loi $\mathscr{E}(lam)$.

À programmer 10 (Représentation d'une densité)**.**

1. Écrire une fonction `exponentielle_densite(x,lamb)` qui renvoie la valeur de la densité choisie.
2. En utilisant la fonction `exponentielle_densite(x,lamb)` simuler un grand nombre de fois une variable suivant la loi $\mathscr{E}(1)$

3. Sur un même graphique faire afficher l'allure de la densité et l'histogramme des fréquences.
4. Recommencer en changeant la valeur du paramètre λ et- décrire la différence

À programmer 11 (Représentation de la fonction de répartition)**.**

1. Écrire une fonction `exponentielle_repartition(x,lam)` qui renvoie la valeur de la densité choisie.
2. En utilisant la fonction `exponentielle_repartition(x,lamb)` simuler un grand nombre de fois une variable suivant la loi $\mathscr{E}(1)$
3. Sur un même graphique faire afficher l'allure de la fonction de répartition et l'histogramme des fréquences cumulatives.
4. Recommencer en changeant la valeur du paramètre λ et- décrire la différence

II.2 Loi géométrique

Sur papier 3 (Approche mathématique)**.**
Pour tout nombre réel x, on note $\lfloor x \rfloor$ la partie entière de x.
Soit X la variable aléatoire suivant la loi exponentielle de paramètre λ ($\lambda > 0$).
On pose $Y = \lfloor X \rfloor$, Y est donc la partie entière de X.

1. Montrer que Y prend ses valeurs dans $\mathbb{N}$.
2. Pour tout k de $\mathbb{N}^*$, calculer $P(Y = k-1)$.
3. En déduire que la variable aléatoire $Y+1$ suit une loi géométrique dont on donnera le paramètre.

À programmer 12 (Implémentation)**.**
En utilisant les questions précédentes et la fonction `expo` précédente, écrire une fonction `geometrique(n,p)` qui renvoie une liste à n éléments dont chaque terme est choisi selon loi $\mathscr{G}(p)$.

III Cas général et loi normale

III.1 Tracer des gaussiennes pour différents paramètres

À programmer 13 (Implémentation)**.**
écrire une fontion `normale_densite(x,m,sigma)` qui renvoie la valeur d'une densité de la loi normale

À programmer 14 (Variations des paramètres)**.**
1. Sur un même graphique tracer les densités en fixant $\sigma = 1$ et en faisant varier le paramètre m. Décrire ce que constatez.
2. Sur un autre graphique tracer les densités en fixant $m = 0$ et en faisant varier le paramètre σ. Décrire ce que constatez.

III.2 Résultats mathématiques

Sur papier 4.
Soit $U \hookrightarrow \mathscr{U}(0,1)$, on note Φ la fonction de répartition de U.

1. Pourquoi Φ est elle **strictement** croissante et continue sur $\mathbb{R}$?
2. Montrer que Φ induit une bijection de $]-\infty;+\infty[$ dans un intervalle que l'on déterminera. Quelle est la monotonie de Φ^{-1}?
3. On note alors $Y = \Phi^{-1}(U)$, pour x réel montrer que
$$P(Y \leqslant x) = \Phi(x)$$
4. En déduire que Y suit la loi $\mathscr{N}(0,1)$.

Remarque : C'est de faite la méthode que nos avons appliquer dans les cas précédent, en utilisant la bijection réciproque de la fonction de répartition en se limitant à un intervalle ou la fonction de répartition est strictement croissante.

III.3 Simulation de la loi Normale centrée réduite

Nous allons donc, pour simuler une loi $\mathscr{N}(0,1)$

- Choisir un nombre au hasard uniformément dans $[0;1[$.
- Lui appliquer la fonction Φ^{-1}.

À programmer 15 (À recopier)**.**
On vous donne le code pour calculer la bijection réciproque de Φ.

```
from scipy.stats import norm
def Finv(x):
        return norm.ppf(x)
```

À programmer 16 (Simulation)**.**
En utilisant la fonction `rd.random()` écrire une fonction `normale_cr(,n)` qui renvoie une liste de longueur n dont chaque terme est choisi selon loi $\mathscr{N}(0,1)$.

À programmer 17 (Représentation d'une densité)**.**
Représenter sur un même graphique un histogramme de densité et la valeur théorique de la densité pour la loi $\mathscr{N}(0,1)$

III.4 Simulation dans le cas général

Sur papier 5 (⛰).
Montrer que si X suit la loi $\mathcal{N}(0,1)$ alors $Y = \sigma X + \mu$ suit la loi $(N(, mu, \sigma^2)$.

À programmer 18.
content

À programmer 19 (Représentation d'une densité)**.**
Représenter sur un même graphique un histogramme de densité et la valeur théorique de la densité pour la loi $\mathcal{N}(1,1)$ puis$\mathcal{N}(0,2)$.

IV ⛰ D'autres Lois

Les lois suivantes, hors programmes, ont été étudiées de nombreuses fois dans des sujets de concours

À programmer 20 (Lois log-normales)**.**
1. Se renseigner sur Wikipedia sur les lois log-normales. Bien noter les paramètres, et l'utilisation de ces lois.
2. Trouver la fonction du module `numpy.random` qui permet de simuler cette loi.
3. Reprendre ce qui a été fait sur les autres lois pour celle ci.

À programmer 21 (Loi de Gumbel)**.**
1. Se renseigner sur Wikipedia sur la loi de Gumbel
2. Trouver la fonction du module `numpy.random` qui permet de simuler cette loi.
3. Reprendre ce qui a été fait sur les autres lois pour celle ci.

À programmer 22 (Loi de Logistique)**.**

1. Se renseigner sur Wikipedia sur la loi logistique.
2. Trouver la fonction du module `numpy.random` qui permet de simuler cette loi.
3. Reprendre ce qui a été fait sur les autres lois pour celle ci.