Université de Rennes
Modélisation - Proba/stat

Préparation à l'agrégation
année 2019/2020

Feuille de travaux pratiques - Python #2

Emeline LUIRARD

1 Illustrations graphiques

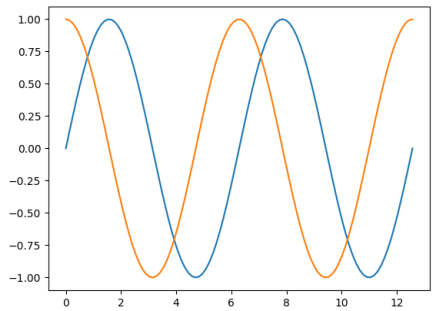
1.1 Tracés en dimension deux

Pour tracer des graphes, nous avons besoin du sous-module `pyplot` de `matplotlib`. Ensuite, on définit la figure, qu'on appelle en général `fig`. Puis dans la figure on définit les sous-figures, selon les axes. On va utiliser la fonction `plot` qui prend en arguments des `array`.

```
fig, axs=plt.subplots(n,m) # on aura des sous-figures sur n lignes et m colonnes
#sans argument quand on n'en veut qu'un.
axs[i,j].plot(x,y,"r", label="nom fonction") # trace dans le graphique (i,j),
#en couleur rouge, et associe une legende
fig.suptitle("titre principal")
axs[0,0].set_title("titre graphe 1")
axs[0,1].set_title(r"$math latex$") # le 'r' dit à Python de ne pas interpreter
#les caracteres speciaux
#Ne pas oublier:
ax.legend() # affiche les légendes
plt.show() # pour ouvrir la fenetre graphique
plt.clf() #efface la fenêtre graphique
```

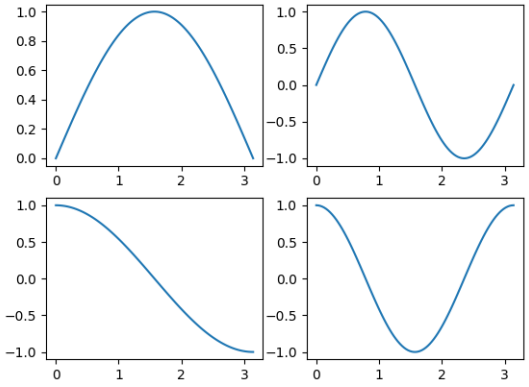
Si on veut représenter plusieurs courbes sur le même graphique, on fait des appels à la fonction `plot` les uns à la suite des autres, ou on lui donne les différentes courbes en arguments :

```
x=np.linspace(0,4*pi,100)
y=np.sin(x)
z=np.cos(x)
fig, ax=plt.subplots()
ax.plot(x,y)
ax.plot(x,z)
#ou
ax.plot(x,y, x,z)
plt.show()
```



Voici un exemple de l'utilisation de `subplots` :

```
x=np.linspace(0,pi,100)
fig, axs=plt.subplots(2,2)
axs[0,0].plot(x,np.sin(x))
axs[0,1].plot(x,np.sin(2*x))
axs[1,0].plot(x, np.cos(x))
axs[1,1].plot(x,np.cos(2*x))
plt.show()
```



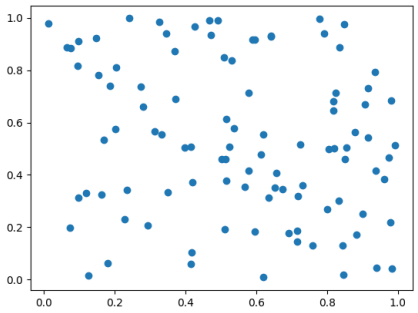
N.B. *Pour tracer une fonction en escalier, on utilisera l'argument de* `plot` *:* `drawstyle='steps-post'` *ou* `drawstyle='steps-pre'`.

Options graphiques

Lorsque vous présentez une illustration graphique au jury, vous **devez** préciser ce qu'elle représente i.e. qu'est ce qui est tracé, en fonction de quoi, quels sont les paramètres etc. Pour faciliter la compréhension et la lisibilité de vos illustrations, vous **devez** ainsi vous habituer à les "décorer" en ajoutant un titre général, un également pour chaque axe. Lorsqu'il y a plusieurs courbes, une légende permet d'afficher la correspondance entre les couleurs/motifs et les courbes représentées.

Le troisième argument de `plot` est le style du trait. On mettra "o" pour que les points d'une courbe ne soient pas reliés ou pour avoir un nuage de points. Voici un exemple :

```
x=stats.uniform.rvs(size=(2,100))
fig, ax=plt.subplots()
ax.plot(x[0,:],x[1,:], 'o')
plt.show()
```



On change les couleurs avec : "r" pour red, "b" pour blue, "k" pour black ; le style avec "." pour un petit point, "o-" pour des gros points et une ligne, " :" pour des pointillés. Et on peut mixer le tout ! Par exemple : "ko".
D'autres commandes :

`axs[..].set_ylabel("titre ordonnees")`	Donne un titre aux ordonnées
`axs[..].set_ylim([a,b])`	limite le graphe aux ordonnées $[a, b]$.
`ax[..].set_xticks(rray)`	graduation axe des abscisses
`ax[..].set_sticklabels(["0", "1",...])`	Noms des graduations de l'axe des abscisses

Voici un exemple :

```
fig, ax=plt.subplots()
#echantillon de variables exponentielles de parametre 2:
X=stats.expon.rvs(scale=0.5, size=10000)
#histogramme associe:
ax.hist(X,bins=30, density=True, label="histogramme")
fig.suptitle("Exponentielle de paramètre 2")
#densité de la loi associée:
x=np.arange(0,max(X), 0.01)
ax.plot(x,stats.expon.pdf(x, scale=0.5), label="densité")
ax.legend()
plt.show()
```

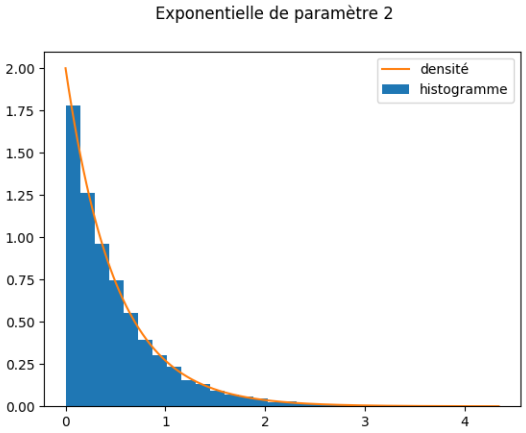
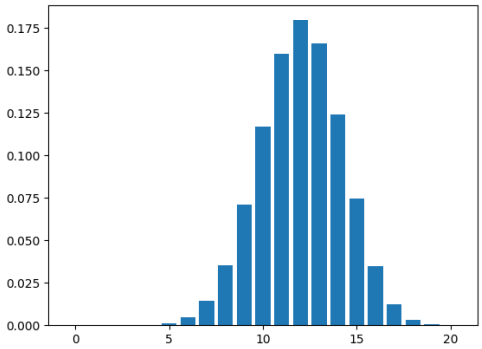


Diagramme en bâtons : histogrammes de lois discrêtes

`ax.bar(x,y,width=)` à la place de `plot`. Voici un exemple :

```
# echantillon
x=stats.binom.pmf(np.arange(0,21),20,0.6)
fig, ax=plt.subplots()
ax.bar(np.arange(0,21),x)
plt.show()
```



Histogramme

L'histogramme d'un échantillon est un diagramme constitué de barres verticales juxtaposées, chacune de ces barres représentant le nombre de termes de l'échantillon appartenant à une classe donnée. Pour représenter un histogramme d'un échantillon d'une loi réelle, on commence donc par

répartir cet échantillon en classes, chacune de ces classes correspondant aux valeurs appartenant à un certain intervalle de $\mathbb{R}$, et on représente cette classe par un rectangle vertical dont l'aire est proportionnelle à l'effectif de la classe. On normalise souvent ces aires de façon à ce que l'aire totale soit égale à 1.
`plt.hist(X,bins=n)` répartit les données de X dans n sous intervalles de $[min(X), max(X)]$. `hist` prend la place de `plot`, on garde la même syntaxe que précédemment pour faire différents graphes par exemple. Mais pour mettre plusieurs histogrammes sur le même on fait `plt.hist([X1,X2,X3], bins=n, label=...)`.

N.B. *Avec `a=plt.hist(X,bins=n)`, on récupère en premier la hauteur des bâtons et en deuxième le découpage des sous-intervalles fait par Python.*

N.B. *On peut donner ses propres sous-intervalles dans un `array` dans `bins`.*

Pour superposer avec une densité, il faut demander à Python de normaliser les histogrammes, ça se fait avec l'option de `hist`, `density=True`. Puis on plote la densité.

Exercice 1. Télécharger le fichier `donnees2.csv` dans votre espace de travail. Importer le tableau dans Python, et tracer la première colonne en fonction de la seconde.

Exercice 2. Télécharger le fichier `donnees3.csv` dans votre espace de travail. Importer le document dans Python. Tracer le nuage de points dont les abscisses et ordonnées sont stockées dans le tableau. Les points seront matérialisés par des triangles.

Exercice 3. Dans une même fenêtre graphique, tracer les unes en dessous des autres les trois fonctions trigonométriques usuelles (sin, cos et tan) sur une période et en regard, sur la droite, leur fonctions réciproques (là où elles sont définies !).

Exercice 4. Générer une fonction qui prend en entrée un entier n et qui en sortie trace la fonction étagée binomiale correspondante i.e. la fonction qui vaut C_n^k sur l'intervalle $[k, k+1]$, pour k variant de 0 à $n-1$.

Exemple d'utilisation pour les lois empiriques :

Pour représenter les lois empiriques associées à une expérience aléatoire, on peut tracer l'histogramme empirique ou la fonction de répartition empirique.

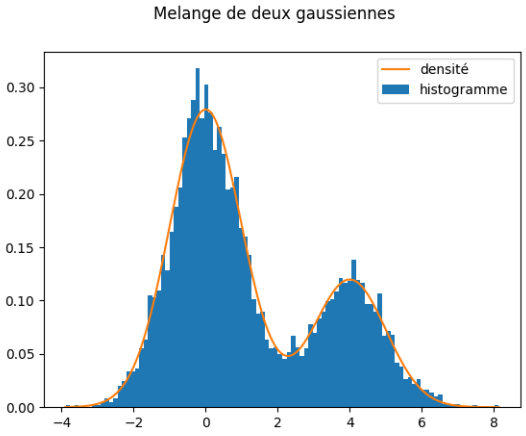
```
N=5000 p=0.7
#echantillons:
X=stats.binom.rvs(1,p, size=N)
Y1=stats.norm.rvs(loc=0, scale=1, size=N)
Y2=stats.norm.rvs(loc=4, scale=1, size=N)
Z=X*Y1 +(1-X)*Y2

def f(x):
        y=(p/sqrt(2*pi))*np.exp(-x**2/2)+((1-p)/sqrt(2*pi))*np.exp(-(x-4)**2/2)
        return y

u=np.linspace(min(Z),max(Z),200)
fig, ax=plt.subplots()
```

```
#histogramme:
ax.hist(Z,bins=100, density=True, label="histogramme")
ax.plot(u,f(u), label="densité")
fig.suptitle("Melange de deux gaussiennes")
ax.legend()
plt.show()
```

Le résultat :

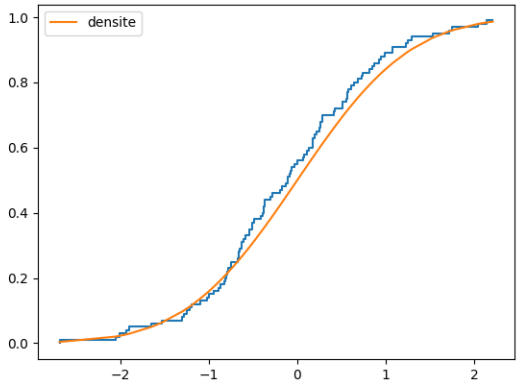


La fonction de répartion empirique d'un n-échantillon $(X_1, \ldots, X_n)$ est la fonction croissante $F_n : \mathbb{R} \to [0, 1]$ définie par

$$F_n(y) = \frac{1}{n} \sum_{k=1}^{n} \mathbb{1}_{X_k \leq y}.$$

Si on dispose d'un échantillon de taille n, on peut tracer le graphe de la fonction de répartition empirique de cet échantillon en ordonnant préalablement cet échantillon. L'exemple suivant tire un échantillon de taille 100 d'une loi normale, trace le graphe de sa fonction de répartition empirique et la compare à la fonction de répartition de la loi normale.

```
n=100
#un échantillon
X=stats.norm.rvs(loc=0,scale=1, size=n)
fig,ax=plt.subplots()
#on le classe:
X_sort=np.sort(X)
ax.plot(X_sort,np.arange(n)/n,drawstyle='steps-pre')
ax.plot(X_sort,stats.norm.cdf(X_sort), label='densite')
ax.legend()
plt.show()
```

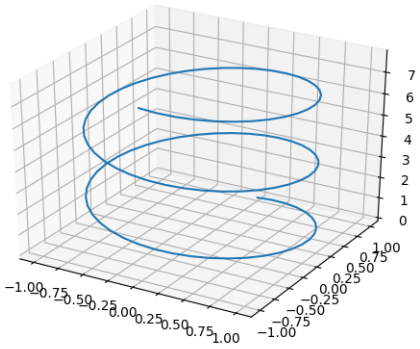


1.2 Tracés en dimension trois

Pour tracer une courbe paramétrée en dimension 3, il nous faut procéder un peu différemment pour créer la figure. On utilisera toujours la fonction `plot` mais avec trois arguments : les projections sur chacun des axes :

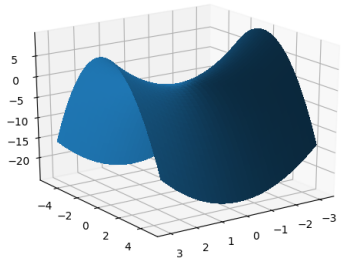
```
#import est nécessaire même si Axes3D
#n'apparait pas ensuite.
from mpl_toolkits.mplot3d import Axes3D

fig = plt.figure()
ax = fig.add_subplot(1,1,1, projection='3d')
t = np.linspace(0, 5 * np.pi, 100)
r = np.sin(t)
x = np.cos(t)
y = t/2
ax.plot(x, y, z)
ax.legend()
```



La commande de base pour représenter une surface est `plot_surface`. La surface d'équation $z = f(x, y)$ est obtenue en entrant les matrices X et Y de coordonnées et une matrice Z de même taille telle que $Z[i, j] = f(X[i, j], Y[i, j])$. X et Y forment une grille. Voici un exemple :

```
#toujours aussi important de l'importer !
from mpl_toolkits.mplot3d import Axes3D
fig = plt.figure()
ax = fig.add_subplot(1,1,1, projection='3d')
X = np.arange(-pi, pi, 0.2)
Y = np.arange(-5, 5, 0.2)
#matrices avec l'ensemble des coordonnées
#de la grille
X, Y = np.meshgrid(X, Y)
Z = X**2 - Y**2
surf = ax.plot_surface(X, Y, Z, antialiased=False)
plt.show()
```



Pour toutes les options de style et de couleur, on se reportera à l'aide.

Exercice 5. Télécharger le fichier donnees4.csv dans votre espace de travail. Importer le document dans Python : la première ligne et la prmière colonne sont les coordonnées. Tracer la surface correspondant aux entrées du tableau.

2 Simulation des variables aléatoires

Pour tirer un réel aléatoire dans $[0, 1]$, on peut utiliser la fonction `np.random.random(size=)`

2.1 Lois usuelles

Rappels :

- `rvs` : simulation
- `pdf(x, parametre)` : probability density function i.e. densité en x
- `pmf(k, parametre)` : probability mass function i.e. densité discrête en k
- `cdf(x,parametre)` : cumulative density function i.e. fonction de répartition en x
- `ppf(q, parametre)` : percent point function i.e. q-ième quantile.

On les utilisera comme ceci : `stats.norm.rvs(parametres)`. Les paramètres sont (paramètres de forme, loc=μ, scale=σ, size=taille échantillon). Les paramètres μ et σ sont tels que, si l'appel de `stats.xxx.rvs()` renvoie une v.a X, alors `stats.xxx.rvs(loc=mu,scale=sigma)` renvoie une v.a ayant la même loi que $\sigma X + \mu$. En tapant `stats.expon?` ou `help(stats.expon)`, vous aurez la normalisation choisie pour la loi exponentielle par exemple. Seuls les paramètres de forme sont obligatoires.

loi(paramètres de forme)	Lois
`norm()`	Loi normale $\mathcal{N}(0,1)$.
`gamma(a=)`	Loi gamma $\gamma(a,1)$.
`beta(a=, b=)`	Loi béta $\beta(a,b)$.
`t(df=n)`	Loi Student de n degrés de liberté.
`cauchy()`	Loi de Cauchy.
`expon(scale=1/lambda)`	Loi exponentielle $\mathcal{E}(\lambda)$.
`uniform(loc=, scale=)`	Loi uniforme $\mathcal{U}(]loc, loc+scale[)$.
`randint(low, high)`	Loi uniforme $\mathcal{U}(\{low, low+1, \cdots, high-1\})$.
`binom(n=, p=)`	Loi binomiale $\mathcal{B}(n,p)$.
`geom(p=)`	Loi géométrique $\mathcal{G}(p)$.
`poisson(mu=)`	Loi de Poisson $\mathcal{P}(\mu)$.

N.B. *Toutes ces lois, ainsi que les relations entre elles sont rappelées dans [Vig18][p. 70].*

N.B. *On peut aussi demander à Python de calculer les moments d'une variable aléatoire. Pour le moment d'ordre n :* `stats.xxx.moment(n,parametres, loc=, scale=)`.

Exercice 6. Simuler une matrice $(M_{i,j})_{i=1\ldots100, j=1\ldots200}$ contenant 100 échantillons de taille 200 de variables aléatoires de loi exponentielle de paramètre 1. Pour illustrer le théorème central limite, tracer l'histogramme associé et sur le même graphique la densité gaussienne correspondante.

2.2 Chaînes de Markov

On s'intéresse aux chaînes de Markov discrètes, homogènes en temps, à espace d'états dénombrable.
En Python, il n'y a pas (à ma connaissance) de fonction toute faite pour générer une chaîne de Markov à partir de sa matrice de transition. Mais ce n'est pas bien difficile et vous pouvez retrouver la méthode dans [Vig18][p. 98] :
On commence par créer la matrice de transition P dans un `array` (cf TP 1). On rappelle que

$$\mathbb{P}(X_{n+1} = y | X_n = x) = P(x,y).$$

Ensuite on va utiliser la fonction `np.random.choice(a,p)` qui renvoie une variable aléatoire à valeurs dans a. p contient les probabilités associées.

```
def markov_avec_P(n,P,x0):
        "simule les n premiers pas d'une chaine de Markov "
        "à partir de sa matrice de transition"
        X=np.zeros(n,dtype=np.int) #on indique que X contiendra seulement des entiers.
        X[0]=x0
        for k in range(n-1):
                X[k+1]=np.random.choice(a=range(len(P)), p=P[X[k],:])
                # Les états sont numérotés de 0 à len(P)-1
        return X
```

Exercice 7. Simuler les 100 premiers pas d'une chaîne de Markov dont la matrice de transition

est donnée par

$$P = \begin{pmatrix} 1/3 & 1/3 & 1/6 & 1/6 \\ 0 & 1/2 & 0 & 1/2 \\ 1/4 & 1/4 & 1/4 & 1/4 \\ 1/3 & 1/3 & 1/3 & 0 \end{pmatrix}.$$

Estimer la probabilité invariante à partir des temps d'occupation empiriques.

2.3 Autres techniques de simulation

2.3.1 Inversion d'une fonction de répartition

Lorsqu'on veut simuler une variable aléatoire dont la loi n'est pas classique, on ne peut malheureusement pas utiliser le sous-module `stats`. Mais, si on connaît sa fonction de répartition F , c'est gagné !

Proposition 1. *Soit X une variable aléatoire réelle de fonction de répartition F. Pour $u \in [0,1]$, on désigne par $F^{-1}(u) := \inf\{x \in \mathbb{R},\ F(x) \geq u\}$ l'inverse généralisée de la fonction de répartition F. Si $U \sim \mathcal{U}([0,1])$ alors $F^{-1}(U)$ a pour fonction de répartition F.*

Exercice 8. Simuler une variable aléatoire à valeurs dans $\{0,1,2,3\}$ dont les probabilités sont respectivement $0.1, 0.3, 0.2, 0.4$. En utilisant `np.random.choice` puis sans.

Exercice 9. Simuler les variables aléatoires décrites ci-dessous par inversion :

1. À partir de variables uniformes sur $[0,1]$, générer un $n-$échantillon de variables de Bernoulli de paramètre $1/2$ à valeurs dans $E = \{0,1\}$. Même question lorsque $E = \{-1,1\}$.
2. Simuler une variable aléatoire Y de loi uniforme à valeurs dans $\{0,1,2,3\}$.
3. À partir de variables uniformes, simuler un $n-$échantillon $(X_1, \ldots, X_n)$ de variables aléatoires de loi exponentielle de paramètre 2. Mettre en évidence la loi des grands nombres en traçant la fonction $k \mapsto (X_1 + \ldots + X_k)/k$ pour k allant de 1 à n.

2.3.2 Méthode de rejet

Proposition 2. *Soient A, $D \in \mathcal{B}(\mathbb{R}^d)$ tels que $A \subset D$. Soit $(X_i)_{i\in\mathbb{N}}$ une suite i.i.d. de variables aléatoires définies sur un espace de probabilité $(\Omega, \mathcal{F}, \mathbb{P})$ uniformes sur D. Introduisons le temps $\tau := \inf\{i \in \mathbb{N}^*,\ X_i \in A\}$, alors X_τ est uniformément distribuée dans A, i.e. pour $B \in \mathcal{B}(\mathbb{R}^d)$, $B \subset A$, $\mathbb{P}(X_\tau \in B) = |B|/|A|$. Le nombre de tirages avant l'obtention d'une réalisation de la loi uniforme sur A par ce procédé suit une loi géométrique $\mathcal{G}(p)$, avec $p = |A|/|D|$. En particulier le nombre moyen de tirages nécessaires est $|D|/|A|$.*

De la proposition, on déduit la méthode de simulation suivante :

Soit X une variable aléatoire réelle de loi μ_X possèdant une densité continue f à support compact inclus dans $[a,b] \subset \mathbb{R}$ et telle que son graphe soit inclus dans $[a,b] \times [0,M]$, pour un certain $M \in \mathbb{R}^+$. On considère une suite $(Z_i)_{i\in\mathbb{N}} = (X_i, Y_i)_{i\in\mathbb{N}}$ de variables aléatoires uniformes dans $[a,b] \times [0,M]$ et l'on définit $\tau = \inf\{i \in \mathbb{N},\ f(X_i) > Y_i\}$. Alors X_τ a pour loi μ_X.

N.B. *Plus généralement, si f est une fonction positive, à support compact inclus dans $[a,b] \subset \mathbb{R}$ et telle que son graphe soit inclus dans $[a,b] \times [0,M]$, pour un certain $M \in \mathbb{R}^+$. Alors X_τ a pour densité $\dfrac{f}{\int f}$.*

Exercice 10. Utiliser la méthode de simulation ci-dessus pour générer une variable aléatoire de densité $x \mapsto \frac{\pi}{2}\sin(\pi x)$ sur l'intervalle $[0,1]$.

On peut généraliser la méthode de rejet de la façon suivante. Soit X une variable aléatoire réelle de loi μ_X qui possède une densité continue f_X (peut ne pas être à support compact) majorée uniformément par une densité g, i.e. il existe une constante $C \geq 1$ telle que

$$\forall x \in \mathbb{R},\ f_X(x) \leq Cg(x), \quad \text{avec} \quad \int g(y)dy = 1.$$

On suppose que l'on sait facilement simuler des variables aléatoires de loi de densité g. Soient donc $(X_i)_{i\in\mathbb{N}}$ une suite i.i.d. de variables aléatoires de loi de densité g et $(U_i)_{i\in\mathbb{N}}$ une suite i.i.d. de variables aléatoires uniformes sur l'intervalle $[0,1]$, indépendantes de $(X_i)_{i\in\mathbb{n}}$. Si on considère le temps $\tau := \inf\{i \geq 1,\ f(X_i) > Cg(X_i)U_i\}$, alors X_τ a pour loi μ_X.

Exercice 11. On souhaite simuler une variable aléatoire X de loi $\mathcal{N}(0,1)$. On montre sans trop de difficulté la majoration suivante :

$$\forall x \in \mathbb{R},\ \underbrace{\left(\frac{e^{-x^2/2}}{\sqrt{2\pi}}\right)}_{:=f_X(x)} \leq \underbrace{\sqrt{\frac{2e}{\pi}}}_{:=C} \times \underbrace{\left(\frac{1}{2}e^{-|x|}\right)}_{:=g(x)}.$$

1. Montrer que si ε suit une loi de Bernoulli de paramètre $1/2$ dans $\{-1,1\}$ et si Z est une variable exponentielle de paramètre 1 indépendante de ε, alors εZ a pour densité la fonction g sur $\mathbb{R}$.
2. Implémenter un algorithme qui simule une variable gaussienne via la méthode de rejet.
3. Simuler 1000 variables selon cet algorithme et vérifier que l'histogramme correspondant approche bien la densité gaussienne.

2.3.3 Simulation de loi conditionnelle

Si A est un événement de probabilité strictement positive associé à une certaine expérience aléatoire et si on veut simuler sous la probabilité conditionnelle $\mathbb{P}(\cdot|A)$ une variable aléatoire dépendant de cette expérience, on effectue une suite de tirages successifs sous la probabilité initiale $\mathbb{P}$ et on conserve uniquement les tirages pour lesquels l'événement A est réalisé.

Exercice 12. Soit X une variable aléatoire réelle de loi de Pareto, admettant la densité

$$f_X(x) = 2/x^3, \quad \text{sur} \quad [1,+\infty[.$$

La fonction de répartition $F_X(x) = 1 - x^{-2}$ s'inverse facilement de sorte que X a même loi que $(1-U)^{-1/2}$ ou encore $U^{-1/2}$ où U suit la loi uniforme sur $[0,1]$. Une propriété remarquable d'une telle variable est que, pour tout réel $c \geq 1$, on a $\mathbb{E}[X|X \geq c] = 2c$. Vérifier empiriquement cette propriété en simulant un échantillon de grande taille de cette loi et en ne conservant que les valeurs supérieures à c.

3 Exercices

Exercice 13. La loi de Fréchet de paramètres $a > 0$ et $\alpha > 0$ est la loi sur $[0,+\infty[$ de fonction de répartition $F_X(x) = \exp(-(a/x)^\alpha)$. Simuler par inversion un échantillon de grande taille de cette

loi. Représenter sur un même graphique la fonction de répartition empirique de cet échantillon et la fonction de répartition théorique de la loi.

Exercice 14. On rappelle que la distance en variation totale entre deux lois de probabilité μ et ν supportées par un ensemble au plus dénombrable E est définie par

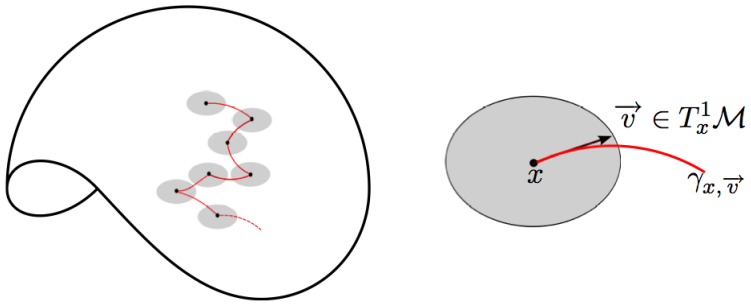
$$\mathrm{d}_{TV}(\mu,\nu) = \sum_{x\in E} |\mu(\{x\}) - \nu(\{x\})|.$$

Evaluez numériquement la distance en variation totale entre la loi de Poisson de paramètre λ et la loi binomiale de paramètres n et $p = \lambda/n$ pour différentes valeurs de n. Représentez graphiquement ces deux lois sur une même figure, puis sur deux figures juxtaposées.

3.1 Pour aller plus loin...

Exercice 15. Estimer par simulation la probabilité $p(n)$ que le déterminant d'une matrice aléatoire A de taille n dont les coefficients sont des variables aléatoires i.i.d. de loi uniforme sur $\{-1, 1\}$, soit nul. On estimera $p(n)$ pour n variant de 2 à 20 et on tracera la courbe représentative de $p(n)$ en fonction de n. Estimer par simulation l'espérance de $\det(A)^2$ pour $n = 2 \dots 5$. Énoncer, puis démontrer une conjecture sur l'espérance de ce déterminant.

Exercice 16. On se propose de simuler le mouvement brownien sur la sphère $\mathbb{S}^2 \subset \mathbb{R}^3$, que l'on munira de la métrique euclidienne induite. Comme le mouvement brownien euclidien est la limite d'échelle "universelle" de marches aléatoires symétriques, on peut montrer que le mouvement brownien sphérique peut être obtenu comme la limite d'échelle d'une marche aléatoire géodésique par morceaux, marche illustrée ci-dessous et que l'on peut décrire comme suit :



On fixe ainsi un paramètre $\varepsilon > 0$ (le pas de notre marche) et on note X_n la position de la marche au temps n.

- Si $X_n = x \in \mathbb{S}^2$,
- on choisit une direction v_x uniformément dans $T^1_x\mathbb{S}^2 = \{v \in T_x\mathbb{S}^2, ||v|| = 1\}$,
- sur une distance $\varepsilon > 0$, on suit la géodésique issue de x et dirigée selon v_x, autrement dit, on suit le grand cercle passant par x tangent au vecteur v_x,
- on arrive ainsi à nouveau point $y \in \mathbb{S}^2$, on pose $X_{n+1} = y$ et on recommence...

1. Implémenter un algorithme qui, étant donné X_n, fournit une réalisation de X_{n+1}.
2. Représenter la marche obtenue sur la sphère de dimension 2.
3. Ensuivant la même démarche, simuler le mouvement brownien sur le tore plat $\mathbb{T}^2 \subset \mathbb{R}^3$.

Références

[Vig18] Vincent Vigon. *python proba stat*. Independently published, October 2018.