6
Analyse fréquentielle

1. L'analyse statistique d'un grand nombre de données consiste d'une part à déterminer l'**étendue** des valeurs rassemblées (*range*), c'est-à-dire l'intervalle compris entre la plus petite valeur et la plus grande valeur, et d'autre part à décrire la **répartition** des valeurs (*distribution*) pour distinguer les valeurs fréquentes des valeurs rares.

2. On peut ensuite chercher à comparer la répartition des données recueillies à une **loi de probabilité**. Un tel **modèle théorique**, s'il est validé, pourra être utilisé pour réaliser des prévisions.

I
Histogrammes

3. Un **histogramme** est une représentation graphique (et donc *qualitative*) de la répartition des données.

4. Construction d'un histogramme
On considère une famille $(x_i)_{i\in I}$ de N données réelles, où l'entier N est trop grand pour que la lecture des x_i permette de se faire une idée de leur répartition.

4.1 L'étendue de ces données est le segment $[m, M]$ où

$$m = \min_{i\in I} x_i \quad \text{et} \quad M = \max_{i\in I} x_i.$$

4.2 Pour obtenir un histogramme de ces données, il faut d'abord **grouper** ces données. Cela consiste à choisir une subdivision du segment $[m, M]$:

$$m = \alpha_0 < \alpha_1 < \cdots < \alpha_n = M.$$

Cette subdivision définit les n **classes** (*bins*)

$$I_0 = [\alpha_0, \alpha_1[\,, \ldots I_{n-2} = [\alpha_{n-2}, \alpha_{n-1}[\quad \text{et} \quad I_{n-1} = [\alpha_{n-1}, \alpha_n]$$

mais on peut aussi s'intéresser aux n classes

$$J_1 = [\alpha_0, \alpha_1], \quad J_2 =]\alpha_1, \alpha_2]\,, \ldots, J_n =]\alpha_{n-1}, \alpha_n]\,.$$

La convention choisie importe peu en pratique, l'essentiel étant de définir une *partition* du segment $[m, M]$.
Dans ce qui suit, nous adopterons la convention du module `numpy` de Python en utilisant les intervalles I_k (le langage Scilab utilise pour sa part les intervalles J_k).

4.3 On doit ensuite calculer l'**effectif** de chaque classe :

$$\forall\, 0 \leqslant k < n, \quad n_k = \#\big\{i \in I \,:\, x_i \in I_k\big\}.$$

La **fréquence** d'une classe est obtenue en rapportant l'effectif de cette classe au nombre total N de données :

$$\forall\, 0 \leqslant k < n, \quad f_k = \frac{n_k}{N}.$$

4.4 L'histogramme des données est alors obtenu en traçant une famille de rectangles basés sur la subdivision $(\alpha_k)_{0\leqslant k\leqslant n}$ et dont les *aires* sont proportionnelles aux effectifs des différentes classes.

5. Calcul de l'histogramme
Pour tout $0 \leqslant i < n$, l'aire du rectangle de base $(\alpha_{i+1} - \alpha_i)$ et de hauteur y_i est égale à $A_i = y_i(\alpha_{i+1} - \alpha_i)$ et par **[4.4]**, il existe un réel $\lambda > 0$ tel que

$$\forall\, 0 \leqslant i < n, \quad A_i = \lambda n_i.$$

5.1 La hauteur du i-ème rectangle est donnée par

$$y_i = \lambda \cdot \frac{n_i}{\alpha_{i+1} - \alpha_i}.$$

5.2 Lorsque les données sont regroupées en n classes de même taille,

$$\forall\, 0 \leqslant i < n, \qquad y_i = \lambda \cdot \frac{n}{M - m} \cdot n_i.$$

5.3 La somme des aires des différents rectangles est dans tous les cas égale à

$$\sum_{0\leqslant i<n} A_i = \lambda \sum_{0\leqslant i<n} n_i = \lambda N.$$

6. Histogramme ou diagramme en bâtons
Il importe de bien distinguer un histogramme d'un **diagramme en bâtons**.

6.1 Un diagramme en bâtons donne une représentation graphique de *données discrètes* (il n'y a qu'un nombre fini de valeurs possibles) et on ne regroupe pas les valeurs **[4.2]** (il y a autant de classes que de valeurs et l'effectif d'une classe est le nombre de fois que cette valeur est prise).

6.2 Dans un diagramme en bâtons, on trace un bâton pour chaque valeur et les **hauteurs** des bâtons sont proportionnelles aux effectifs des classes.

6.3 L'origine de la confusion est double :
– Pour des raisons de lisibilité, de nombreux logiciels tracent des rectangles au lieu de tracer des bâtons ;
– Lorsque les classes d'un histogramme sont des intervalles *de même longueur* **[5.2]**, les *hauteurs* y_i des rectangles (et pas seulement leurs aires A_i) sont proportionnelles aux effectifs n_i des différentes classes.

6.4 On retiendra que :
– dans un diagramme en bâtons, c'est la *longueur* des bâtons qui est significative ;
– dans un histogramme, c'est l'*aire* des rectangles qui est significative.

7. Modèle théorique
Comme l'aire $y \times \delta x$ d'un rectangle de base δx et de hauteur y mesure la probabilité, la hauteur y de ce rectangle doit être considérée non pas comme une probabilité, mais comme une *densité* de probabilité.

7.1 ▲ *On appelle* ***densité de probabilité*** *toute fonction f continue par morceaux et positive sur $\mathbb{R}$ telle que*

$$\int_{-\infty}^{+\infty} f(x)\,\mathrm{d}x = 1.$$

7.2 On dit que la loi d'une variable aléatoire réelle X est décrite par la densité de probabilité f lorsque

$$\mathbf{P}(a \leqslant X \leqslant b) = \int_a^b f(x)\,\mathrm{d}x,$$

quels que soient a et b dans $\mathbb{R}$ tels que $a \leqslant b$.

7.3 On peut interpréter la densité f en remarquant que

$$f(x) = \lim_{h\to 0} \frac{\mathbf{P}(x - h \leqslant X \leqslant x + h)}{2h}$$

pour tout $x \in \mathbb{R}$ en lequel la fonction f est continue.

7.4 Une variable X de densité f est d'espérance finie si, et seulement si, la fonction

$$[x \mapsto xf(x)]$$

est intégrable sur $\mathbb{R}$ et dans ce cas, son espérance est définie par

$$\mathbf{E}(X) = \int_{-\infty}^{+\infty} xf(x)\,\mathrm{d}x.$$

Histogrammes avec `numpy`

8. Soit A, un tableau `numpy` de taille quelconque, contenant des valeurs réelles (de type entier ou de type flottant).

```
import numpy.random as rd
A = rd.rand(1000).reshape(20, 50)
```

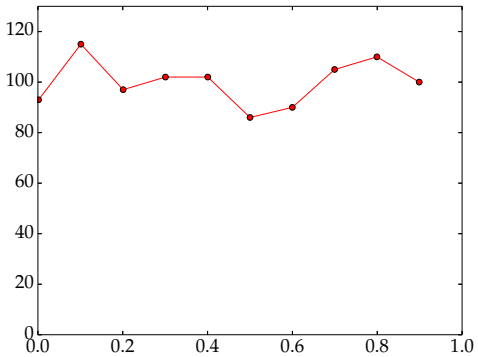
8.1 Fonctionnement par défaut

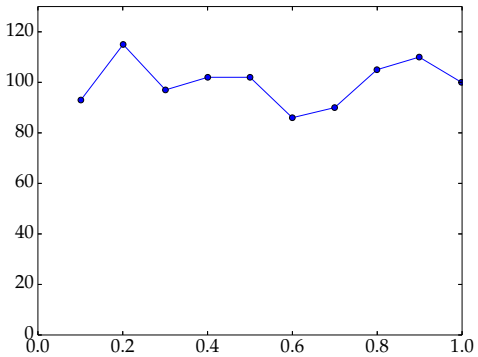
L'instruction élémentaire

```
y, x = np.histogram(A)
```

calcule l'étendue $[m, M]$ des valeurs contenues dans le tableau A, regroupe ces valeurs en dix classes et retourne d'une part la liste $y = (n_i)_{0\leqslant i<10}$ des effectifs de ces dix classes et d'autre par la subdivision $x = (\alpha_i)_{0\leqslant i\leqslant 10}$ de $[m, M]$ en dix sous-intervalles de même longueur.
On peut alors se faire une idée de la répartition des valeurs en traçant la ligne brisée définie par les points $(\alpha_i, n_i)_{0\leqslant i<10}$ ou par les points $(\alpha_{i+1}, n_i)_{0\leqslant i<10}$.

```
plt.ylim(0, 130)  # 1000 points répartis en 10 classes
plt.plot(x[:-1], y, 'ro-')  # Points (α_i, n_i)
plt.plot(x[1:], y, 'bo-')   # Points (α_{i+1}, n_i)
```





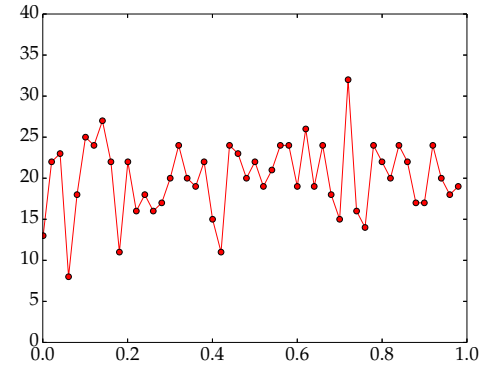
8.2 Nombre de classes

On peut modifier le nombre de classes en donnant une valeur entière à `bins`. L'instruction

```
y, x = np.histogram(A, bins=c)
```

procède comme en [8.1], mais avec c sous-intervalles de même longueur. Les listes x et y sont alors de tailles respectives $(c+1)$ et c.

```
y, x = np.histogram(A, bins=50)
plt.ylim(0, 40) # 1000 points répartis en 50 classes
plt.plot(x[:-1], y, 'ro-') # Points (α_i, n_i)
```



8.3 Normalisation

En précisant `density=True`, la liste y contient les *densités* de fréquence des sous-intervalles au lieu des *effectifs* n_i, de telle sorte que l'aire totale des rectangles [5.3] soit égale à 1.

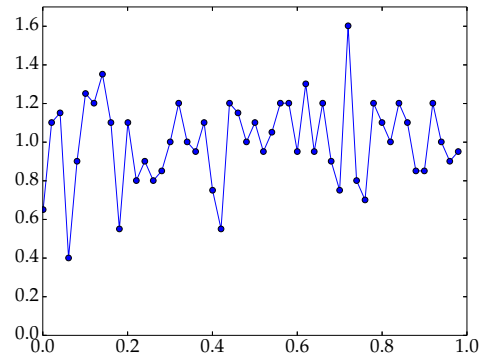
– Dans le cas où toutes les classes sont de même longueur :

$$\forall\, 0 \leqslant i < n, \quad y_i = \frac{1}{N} \cdot \frac{n}{M-m} \cdot n_i = \frac{n}{M-m} \cdot f_i.$$

– Dans le cas général :

$$\forall\, 0 \leqslant i < n, \quad y_i = \frac{1}{N} \cdot \frac{n_i}{\alpha_{i+1} - \alpha_i} = \frac{f_i}{\alpha_{i+1} - \alpha_i}.$$

```
y, x = np.histogram(A, bins=50, density=True)
plt.ylim(0,1.7)
plt.plot(x[:-1], y, 'bo-')
```

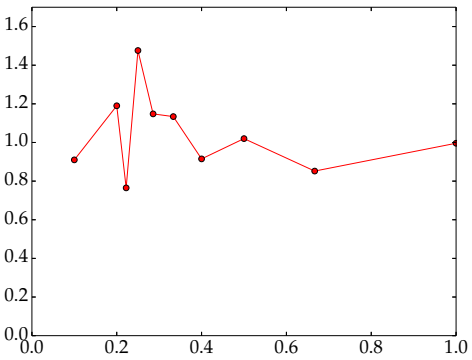


8.4 Classes de longueur variable

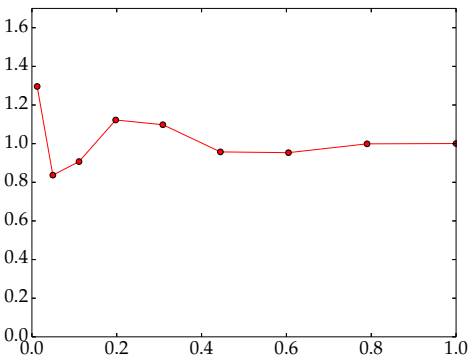
On peut aussi affecter à `bins` une liste ou un tableau contenant les bornes α_i des différentes classes rangées par ordre croissant. Dans ces conditions, les classes peuvent être des intervalles de longueurs différentes, ce qui permet d'adapter finement le choix des classes aux données à étudier.

Attention ! Si les classes ne sont pas toutes de même longueur, les valeurs doivent être normalisées avec l'option `density=True`, sans quoi le résultat *n'est pas un histogramme* ! →**[6]**

```
L = [0, 0.1]+[2/x for x in range(10, 1, -1)]   # liste
y, x = np.histogram(A, bins=L, density=True)
plt.plot(x[1:], y, 'ro-')
```




```
L = np.linspace(0, 1, 10)**2          # tableau numpy
y, x = np.histogram(A, bins=L, density=True)
plt.plot(x[1:], y, 'ro-')
```



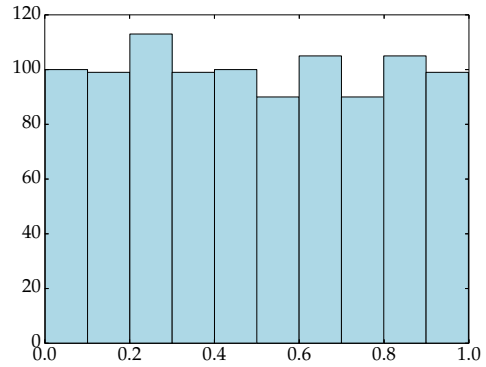
8.5 La seconde valeur retournée par `np.histogram` est alors la valeur de l'argument `bins` convertie en tableau `numpy`.

Histogrammes avec `matplotlib`

9. Le module `matplotlib` dispose d'une fonction `hist` dont le fonctionnement est très similaire à la fonction `histogram` de `numpy`.

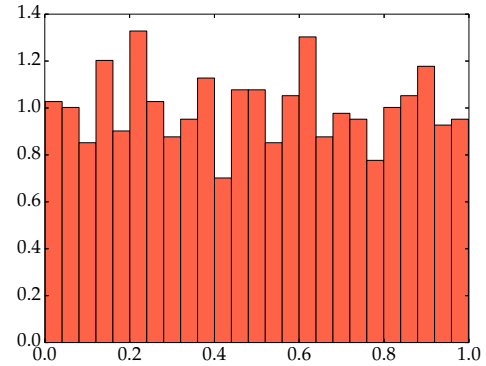
9.1 L'argument principal est une liste ou un tableau *unidimensionnel*. Si les données sont contenues dans un tableau multidimensionnel, il faut donc *aplatir* le tableau.

```
import matplotlib.pyplot as plt
# données sous forme d'un tableau bidimensionnel
A = rd.rand(1000).reshape(50,20)
# données aplaties (tableau unidimensionnel)
A = A.flatten()
plt.hist(A, color='lightblue')
```



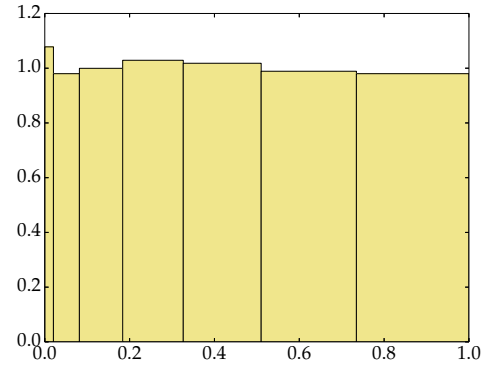
9.2 L'option `density` de `np.histogram` devient ici `normed`, avec le même fonctionnement.

```
plt.hist(A, bins=25, normed=True, color='tomato')
```



9.3 L'argument facultatif `bins` fonctionne comme au [**8.4**]. Si les classes ne sont pas des intervalles de même longueur, l'argument `normed=True` est nécessaire pour obtenir un histogramme correct.

```
L = np.linspace(0, 1, 8)**2
plt.hist(A, bins=L, normed=True, color='khaki')
```



9.4 La fonction `plt.hist` retourne un triplet :

- Les deux premiers éléments sont exactement les listes retournées par la fonction `np.histogram` ;
- Le dernier élément est un objet graphique : la liste des rectangles à dessiner (*a list of <n> Patch objects*).

9.5 La liste des couleurs disponibles se trouve à l'adresse suivante :

```
matplotlib.org/mpl_examples/color/named_colors.png
```

II

Loi des grands nombres

10. Soit $(X_n)_{n\in\mathbb{N}}$, une suite de variables aléatoires indépendantes et de même loi. Ces variables sont considérées comme des réalisations d'une même expérience aléatoire dans des conditions identiques.

10.1 La **moyenne empirique** est définie par

$$\forall\, n \in \mathbb{N}^*, \quad M_n = \frac{1}{n}\sum_{k=1}^{n} X_k.$$

10.2 D'après la **loi des grands nombres**, si la loi commune aux variables X_n est d'espérance finie, alors la moyenne empirique M_n converge en probabilité vers l'espérance $\mathbf{E}(X_0)$:

$$\forall\, \varepsilon > 0, \quad \lim_{n\to+\infty} \mathbf{P}\big(|M_n - \mathbf{E}(X_0)| \geqslant \varepsilon\big) = 0.$$

10.3 Si la loi commune aux variables X_n possède un moment d'ordre deux, l'**inégalité de Bienaymé-Tchebychev** précise la vitesse de cette convergence.

$$\forall\, \varepsilon > 0,\ \forall\, n \geqslant 1, \quad \mathbf{P}\big(|M_n - \mathbf{E}(X_0)| \geqslant \varepsilon\big) \leqslant \frac{\mathbf{V}(X_0)}{n\varepsilon^2}.$$

10.4 L'inégalité de Bienaymé-Tchebychev ne dépend de la loi des variables X_k *que* par les valeurs de l'espérance et de la variance. Il s'agit d'une estimation simple mais assez grossière.

Exemples

11. On suppose ici que X_0 suit la loi de Bernoulli

$$\mathbf{P}(X_0 = 1) = p \qquad \mathbf{P}(X_0 = 0) = 1 - p$$

de paramètre $p = 0,3$.

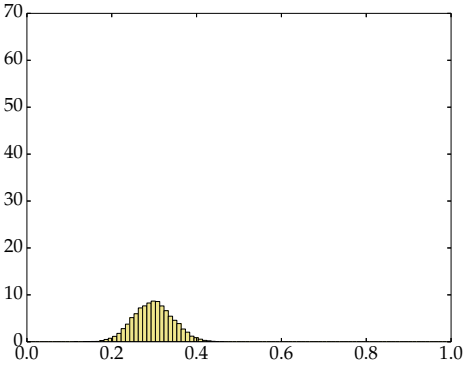
11.1 On peut appliquer l'inégalité de Bienaymé-Tchebychev avec $\mathbf{E}(X_0) = p$ et $\mathbf{V}(X_0) = p(1-p) = 0,21$.

11.2 Avec le module `numpy.random`, on peut simuler $N = 10^4$ réalisations de la moyenne empirique M_n et représenter graphiquement ces N réalisations au moyen d'un histogramme, en prenant

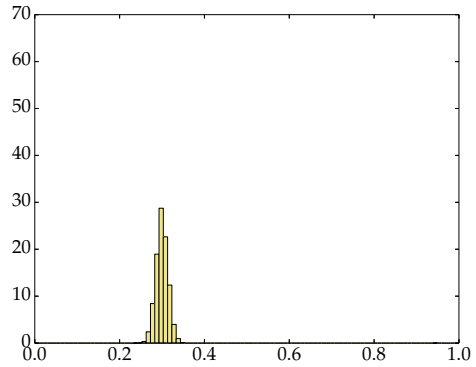
```
bins=np.linspace(0, 1, 100)
```

pour regrouper les données.

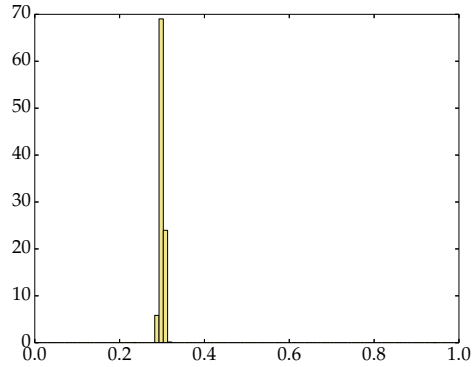
11.3 Avec $n = 100$



11.4 Avec $n = 1\,000$



11.5 Avec $n = 10\,000$



11.6 Ces trois figures confirment bien la Loi des grands nombres : plus l'échantillon est important (c'est-à-dire : plus l'entier n est grand), plus les valeurs de la moyenne empirique sont concentrées autour de la moyenne théorique qu'est l'espérance.

12. Le lancer d'un dé à 6 faces est ordinairement modélisé par la loi uniforme sur l'ensemble $\{1,2,3,4,5,6\}$. Ici encore, on peut appliquer l'inégalité de Bienaymé-Tchebychev avec

$$\mathbf{E}(X_0) = \frac{7}{2} \quad \text{et} \quad \mathbf{V}(X_0) = \frac{35}{12}.$$

12.1 On simule $N = 10^4$ réalisations de n lancers de dé et pour chacune des réalisations, on calcule la moyenne empirique.

```
lancers = rd.randint(1, 7, size=(n,N))
moy_emp = np.mean(lancers, axis=0)
```

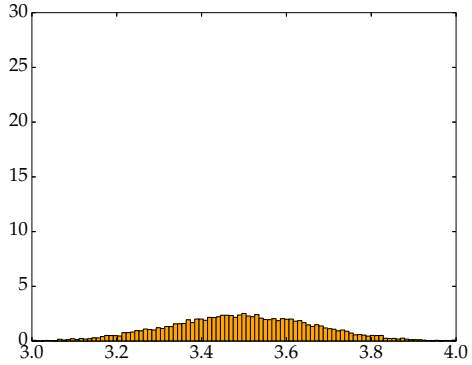
On représente graphiquement cet échantillon avec l'option

```
bins = np.linspace(3, 4, 100)
```

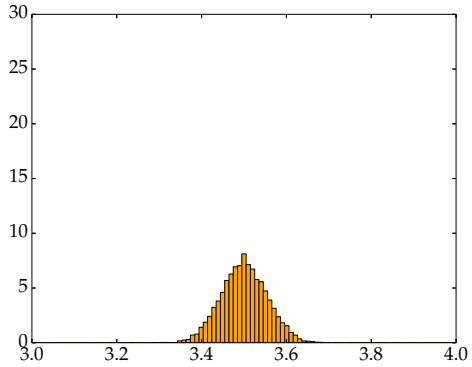
puisque les valeurs de la moyenne empirique doivent se rapprocher de $3,5$.

12.2 On peut remarquer ici l'influence de la variance [**10.4**] : dans ce second exemple, la variance est environ 15 fois plus grande que dans le premier exemple et la convergence de la moyenne empirique vers la moyenne théorique (c'est-à-dire l'espérance de loi) est sensiblement plus lente.

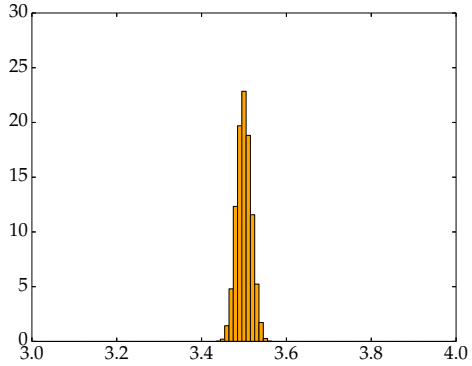
12.3 Avec $n = 100$



12.4 Avec $n = 1\,000$



12.5 Avec $n = 10\,000$



13. On suppose ici que X_0 suit la loi géométrique de paramètre p :

$$\mathbf{P}(X_0 = 0) = 0 \quad \text{et} \quad \forall\, k \geqslant 1, \quad \mathbf{P}(X_0 = k) = pq^{k-1}$$

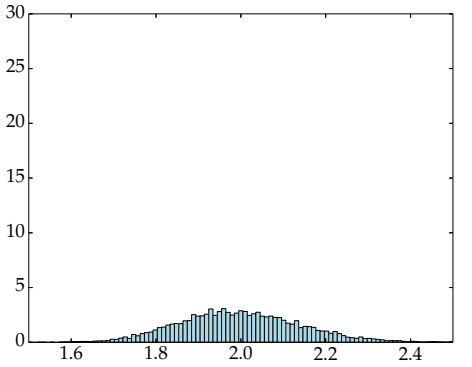
où $q = 1 - p$. On peut appliquer l'inégalité de Bienaymé-Tchebychev avec

$$\mathbf{E}(X_0) = \frac{1}{p} \quad \text{et} \quad \mathbf{V}(X_0) = \frac{q}{p^2}.$$

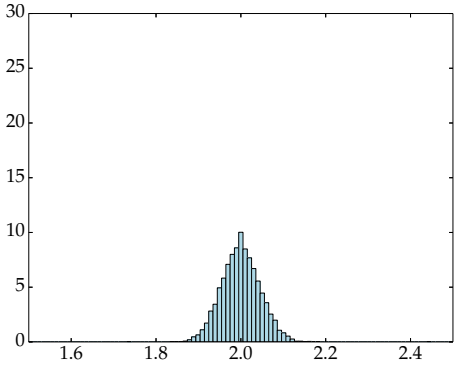
13.1 On procède d'une manière analogue à ce qu'on a fait plus haut avec $p = 0,5$. Les valeurs aléatoires de la moyenne empirique doivent donc se rapprocher de $\mathbf{E}(X_0) = 2$.

```
ech = rd.geometric(0.5, size=(n, N))
moy_emp = np.mean(ech, axis=0)
L = np.linspace(1.5, 2.5, 100)
plt.hist(moy_emp, bins=L, normed=True)
```

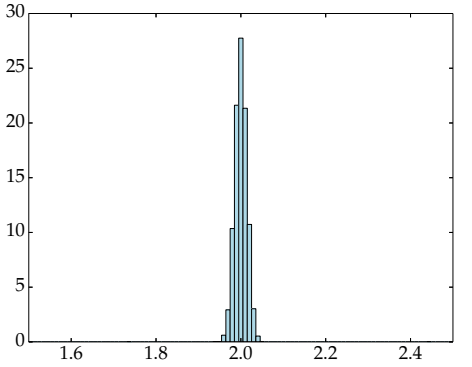
13.2 Avec $n = 100$



13.3 Avec $n = 1\,000$



13.4 Avec $n = 10\,000$



13.5 Avec $p = 0,5$, on a $\mathbf{V}(X_0) = 4$: la variance est un peu supérieure à la variance du second exemple mais on constate graphiquement que la convergence de la moyenne empirique vers la moyenne théorique est légèrement plus rapide. Cela confirme que l'inégalité de Bienaymé-Tchebychev est trop générale pour estimer précisément la vitesse de convergence. →**[10.4]**

III

Convergence en loi

14. La Loi des grands nombres montre que la moyenne empirique d'une suite de variables aléatoires tend vers une constante (c'est-à-dire une quantité *non* aléatoire).

14.1 En particulier, si $(X_n)_{n\in\mathbb{N}}$ est une suite de variables aléatoires indépendantes qui suivent toutes la loi de Bernoulli de paramètre $0 < p < 1$, la fréquence empirique

$$\frac{S_n}{n} = \frac{1}{n}\sum_{0\leqslant k<n} X_k$$

tend vers $\mathbf{E}(X_0) = p$.

14.2 Ce résultat justifie l'*interprétation fréquentiste* de la notion de probabilité : la probabilité p pour qu'un événement donné $[X = 1]$ se réalise est la limite de la fréquence S_n/n d'apparition de cet événement lorsqu'un grand nombre d'expériences aléatoires est réalisé dans des conditions identiques (les variables X_n indépendantes et de même loi).

15. Le **théorème de Moivre-Laplace** précise cette version de la Loi des grands nombres en décrivant la répartition des valeurs de la fréquence empirique (c'est-à-dire la loi de S_n/n.

15.1 Si $(X_n)_{n\in\mathbb{N}}$ est une suite de variables aléatoires indépendantes qui suivent toutes la loi de Bernoulli de paramètre $0 < p < 1$, la différence $S_n/n - p$ tend vers 0, donc $\sqrt{n}(S_n/n - p)$ est une forme indéterminée.

15.2 La variable aléatoire

$$S_n^* = \frac{\sqrt{n}(S_n/n - p)}{\sqrt{p(1-p)}} = \frac{S_n - np}{\sqrt{np(1-p)}}$$

est centrée (espérance nulle) et réduite (variance unité). Le théorème de Moivre-Laplace montre que la loi de cette variable aléatoire tend vers la loi gaussienne centrée réduite, au sens où

$$\mathbf{P}\Big(a \leqslant \frac{S_n - np}{\sqrt{np(1-p)}} \leqslant b\Big) \xrightarrow[n\to+\infty]{} \int_a^b \frac{1}{\sqrt{2\pi}} e^{-x^2/2}\,\mathrm{d}x$$

quels que soient les réels $a < b$.

15.3 En pratique, l'histogramme d'une série statistique de N réalisations de la variable S_n^* se rapproche de la *courbe en cloche* qui représente la fonction

$$\left[x \mapsto \frac{1}{\sqrt{2\pi}} \exp\frac{-x^2}{2}\right]$$

pourvu que les entiers n et N soient des nombres assez grands.

```
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(-3., 3.)
y = np.exp(-0.5*x**2)/np.sqrt(2*np.pi)
plt.plot(x, y, 'r')        # Courbe en cloche
```

16. On peut si bien généraliser le théorème de Moivre-Laplace que sa généralisation est présentée comme le **Théorème-limite central**.

16.1 Si $(X_n)_{n\in\mathbb{N}}$ est une suite de variables aléatoires indépendantes et de même loi, d'espérance m et de variance σ^2, alors la loi de la variable aléatoire centrée réduite

$$S_n^* = \frac{1}{\sqrt{n}\sigma} \cdot \Big(\sum_{0\leqslant k<n} X_k - n\,\mathbf{E}(X)\Big)$$

tend vers la loi gaussienne centrée réduite.

16.2 Ce théorème est très utile pour deux raisons.
- D'une part, les hypothèses sur les variables aléatoires X_n sont faibles : on suppose seulement qu'elles ont un moment d'ordre deux fini.
- D'autre part, la loi limite est la toujours même, quelle que soit la loi des X_n.

16.3 C'est pour cette dernière raison que la loi gaussienne est aussi appelée la **loi normale des erreurs** : une erreur (au sens d'incertitude) résulte de la superposition d'un grand nombre de facteurs inconnus, aucun de ces facteurs n'étant prédominant.

17. Cas de la loi de Bernoulli

La **loi de Bernoulli** $\mathscr{B}(p)$ est aussi la loi binomiale $\mathscr{B}(1, p)$.

17.1 On simule $N = 10^4$ réalisations de n variables indépendantes qui suivent toutes la loi de Bernoulli de paramètre $p = 0,3$.

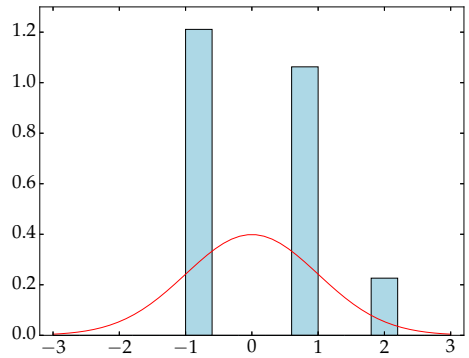
```
import numpy.random as rd

p = 0.3
q = 1-p
ech_iid = rd.binomial(1, p, size=(N,n))
```

On calcule ensuite les fréquences empiriques qu'on normalise pour obtenir N réalisations de la variable aléatoire centrée et réduite S_n^*.

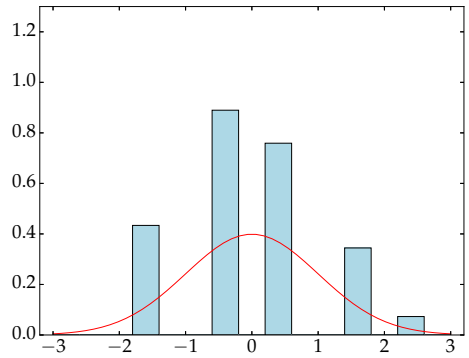
```
freq_emp = np.mean(ech_iid, axis=1)
ech_norm = (freq_emp-p)*np.sqrt(n/(p*q))
```

17.2 Comme une variable de Bernoulli ne peut prendre que les valeurs 0 et 1, la variable S_n^* ne peut prendre que $(n+1)$ valeurs distinctes. Il faut donc prendre l'entier n vraiment grand pour pouvoir confondre l'histogramme de S_n^* avec la courbe gaussienne.

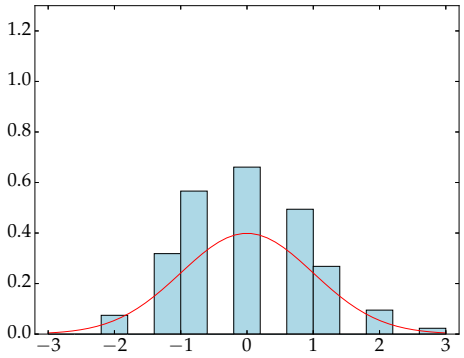
17.3 Avec $n = 2$



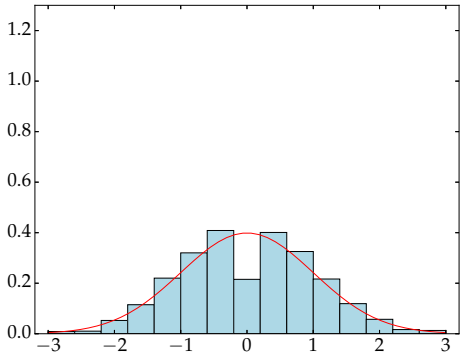
17.4 Avec $n = 5$



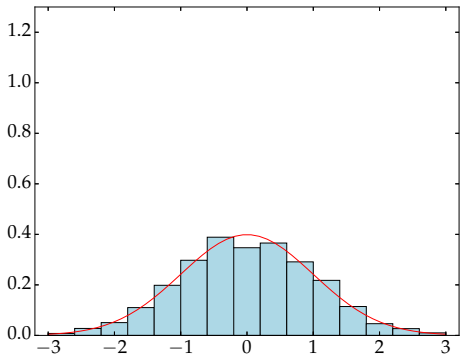
17.5 Avec $n = 10$



17.6 Avec $n = 100$



17.7 Avec $n = 1\,000$



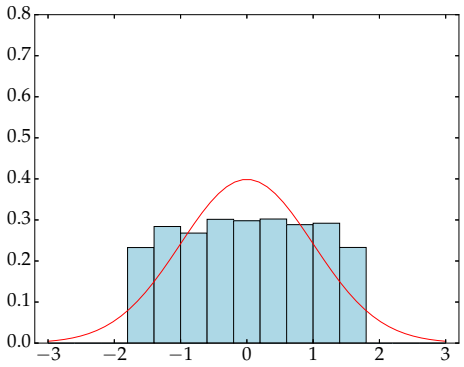
18. Cas de la loi uniforme sur $[0,1]$

Une variable aléatoire X suit la loi uniforme sur $[0,1]$ lorsque

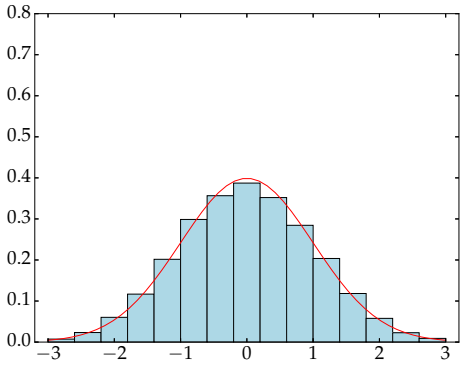
$$\forall\ 0 < a < b < 1, \quad \mathbf{P}(a \leqslant X \leqslant b) = \int_a^b \mathrm{d}x = (b-a).$$

L'espérance et la variance de X sont alors égales à $1/2$ et $1/12$.

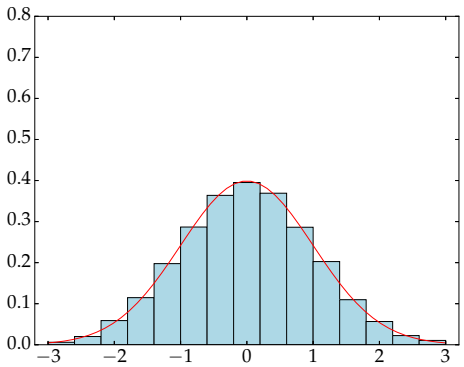
18.1 Avec $n = 1$: histogramme de la variable X



18.2 Avec $n = 5$



18.3 Avec $n = 10$



On constate cette fois qu'on peut confondre la loi de S_n^* et la loi normale gaussienne centrée réduite même pour des valeurs de n relativement petites.

19. Cas de la loi exponentielle
Une variable aléatoire X suit la loi exponentielle de paramètre $\lambda > 0$ lorsque

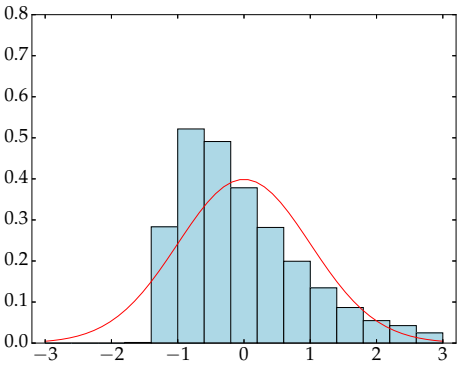
$$\forall\, 0 < a < b, \quad \mathbf{P}(a \leqslant X \leqslant b) = \int_a^b \lambda e^{-\lambda x}\, \mathrm{d}x.$$

L'espérance et la variance de X sont alors respectivement égales à $\beta = 1/\lambda$ et à β^2.

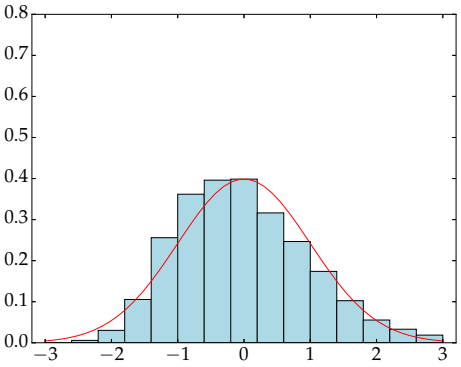
19.1 On simule un échantillon de variables aléatoires exponentielles en précisant non pas le paramètre mais l'espérance des variables.

```
beta = 2.
ech_Exp = rd.exponential(beta, size=(N, n))
```

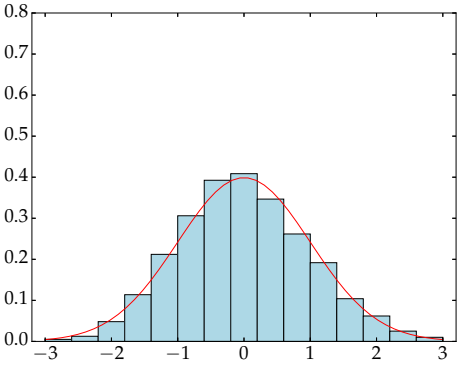
19.2 Avec $n = 2$



19.3 Avec $n = 10$



19.4 Avec $n = 50$



19.5 Avec $n = 1000$