Rencontre Polytechnique - Enseignants du secondaire
Lucas Gerin (écrire) (mailto:lucas.gerin@polytechnique.edu)

Un exemple de modélisation et simulation probabiliste

Cette session se place dans le contexte suivant :

- Terminale Voie Générale (Mathématiques Expertes)
  - Graphes et matrices
    - Chaîne de Markov à deux ou trois états. Distribution initiale, représentée par une matrice ligne π_0 . Matrice de transition, graphe pondéré associé.

Nous allons traiter expérimentalement et théoriquement un modèle mathématique issu de la vie réelle : le problème des arrondis de monnaie au Canada. L'objectif est de présenter une démarche de recherche assez proche de ce qui se fait en Mathématiques Appliquées.

Deux supports pédagogiques peuvent compléter ce Notebook :

- Alexandre Marino. *Les Ehrenfest viennent en aide à Boltzmann*. CultureMath - Eduscol (http://culturemath.ens.fr/content/les-ehrenfest-viennent-en-aide-%C3%A0-boltzmann)
  (Introduction très claire aux chaînes de Markov)
- Louis-Marie Bonneval. *Chaînes de Markov au lycée*. Bulletin de l'APMEP n.503. (https://www.apmep.fr/IMG/pdf/07-Bonneval_C.pdf)
  (Etude de quelques exemples, moins formel que la 1ère référence)

Sommaire

```
# Cette cellule sert à faire un bel affichage du notebook
from IPython.core.display import HTML
def css_styling():
    styles = open("./style/custom2.css").read()
    return HTML(styles)
css_styling()
```

```
# On charge les librairies Python

import matplotlib.pyplot as plt # Pour tracer des graphiques
import numpy as np              # Pour faire des maths

# Cette commande demande que les sorties graphiques soient dans la fenêtre pri
%matplotlib inline
```

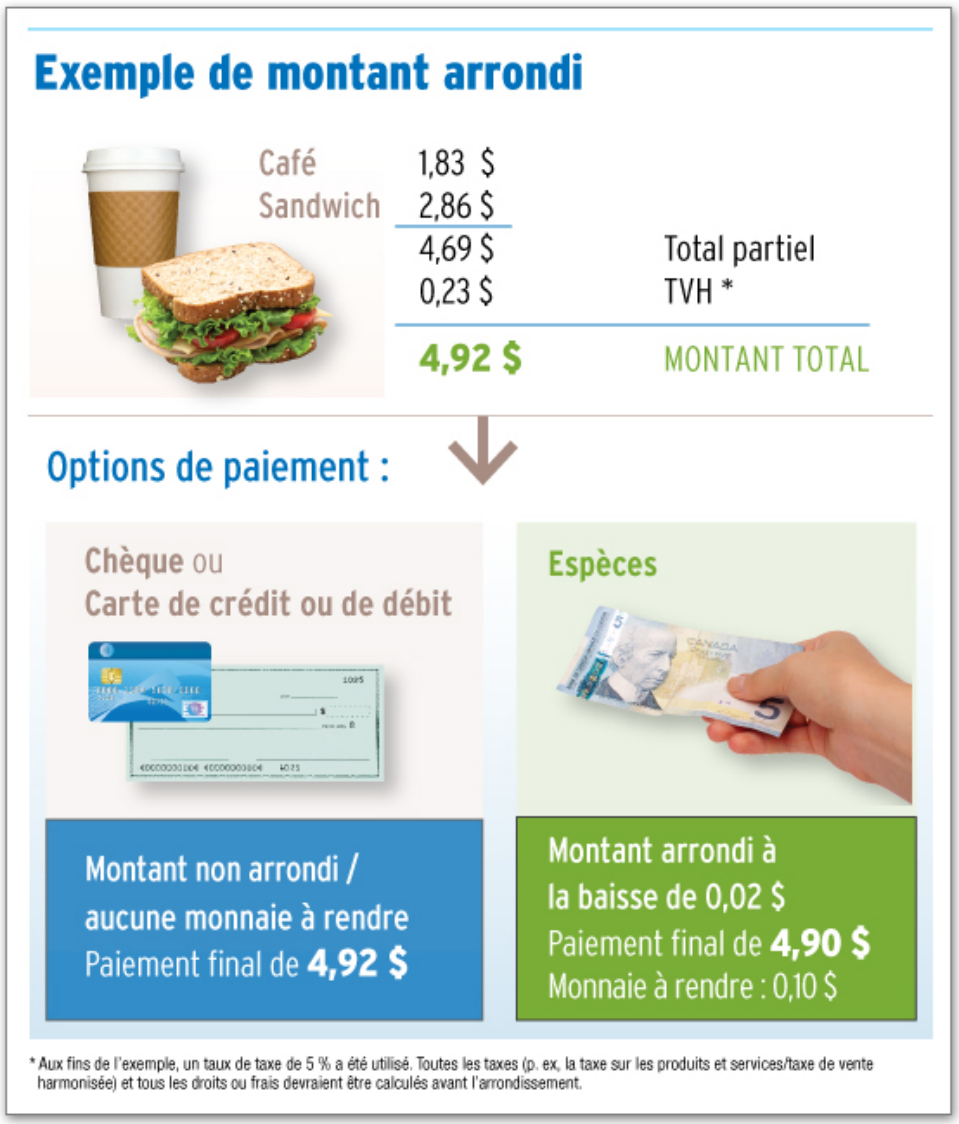
```
# Librairies spéciales pour pouvoir créer des boutons interactifs ("widgets")
from ipywidgets import widgets
from ipywidgets import interact
from IPython.display import display
```

Le problème : arrondis de monnaie au Canada

En 2012, le Canada a décidé d'arrêter la production et simulation des pièces de 1 cent. Toutefois il est possible de proposer des prix au cent près, la règle en vigueur pour un paiement en espèce est alors un arrondi au multiple de cinq cents le plus proche.

(Source : Ministère des Finances Canada)

Attention : La règle essentielle pour ce qui suit est que lorsqu'un paiement concerne plusieurs produits, l'arrondi se fait sur la somme finale :



(Source : Ministère des Finances Canada)

Les questions que l'on se pose sont justement liées au cas des paniers avec beaucoup de produits :

- Question 1 : Est-ce que de manière générale cette règle est plutôt favorable ou défavorable aux clients? (Est-ce que l'on gagne plus ou moins souvent 1 ou 2 cents que l'on en perd?)
- Question 2 : Est-il possible (pour un supermarché par exemple) de tricher en faisant en sorte que la plupart du temps l'arrondi profite au magasin (c'est-à-dire que la plupart des montants finissent par $3, 4, 8, 9$).

On voit que ces questions sont mathématiquement assez mal posées, il nous faut donc un modèle. Nous avons choisi un modèle probabiliste.

Le modèle probabiliste

Notations

On suppose qu'un client achète N produits, et pour $1 \leq n \leq N$ on note X_n ce qui nous intéresse: le second chiffre après la virgule dans le prix du n-ème produit.

Au final l'arrondi se produit sur le second chiffre après la virgule de $X_1 + \cdots + X_n$. On note R_N cet arrondi, on a

$$R_N = X_1 + \cdots + X_N \bmod 10.$$

(Si par exemple un client achète $N = 2$ *produits à* $3,74$ \$ *et* $11,68$ \$ *alors on a* $X_1 = 4$, $X_2 = 8$ et $R_2 = 4 + 8 \bmod 10 = 2$.)

- Si $R_N \in \{1, 2, 6, 7\}$, l'arrondi est favorable au client.
- Si $R_N \in \{3, 4, 8, 9\}$, l'arrondi est favorable au supermarché.

Hypothèses

On suppose que les X_i sont des variables aléatoires indépendantes et ont toutes la même loi. Pour tout $r = 0, 1, \ldots, 9$ on note

$$p_r = \mathbb{P}(X_n = r).$$

(Remarquons que ce nombre ne dépend pas de n car les X_n ont même loi.)

Une rapide étude de marché (Source : Catalogue de jouet de Walmart Canada (https://www.walmart.ca/fr)) nous a donné les estimations suivantes sur 127 jouets :

$$p_0 = 0.071, \quad p_1 = 0.087, \quad p_2 = 0.087, \quad p_3 = 0.024, \quad p_4 = 0.071,$$
$$p_8 = 0.047, \quad p_9 = 0.165,$$

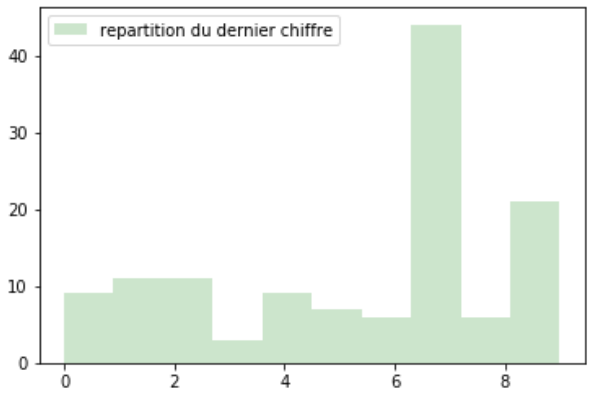
> Remarque. On voit en particulier que les derniers chiffres ne sont pas du tout uniformes : les p_r sont très différent. Je m'attendais personnellement à avoir p_9 beaucoup plus grand, mais c'est étrangement 7 qui est le chiffre le plus courant. En tout cas, si on achète un seul produit, la règle est favorable au client! En effet
>
> $$p_1 + p_2 + p_6 + p_7 = 0.567$$
>
> alors que
>
> $$p_3 + p_4 + p_8 + p_9 = 0.307$$

Voici les valeurs saisies dans une liste python :

```
PrixUnitaires=[0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1

DistributionWalmart=[0.071 , 0.087 , 0.087 , 0.024 , 0.071 , 0.055 , 0.047 , 0

plt.hist(PrixUnitaires, bins= 10,  facecolor='g', alpha=0.2,label='repartition
plt.legend(loc='best')
plt.show()
```



Reformulation mathématique du problème

- Question 1. Si N est grand, et que les valeurs des $(p_r)_{0\leq r\leq 9}$ sont comme ci-dessus, que peut-on dire de la loi de la variable aléatoire R_N ? En particulier, est-ce qu'elle prend plus souvent les valeurs $1, 2, 6, 7$ ou $3, 4, 8, 9$?
- Question 2 Existe-t-il un choix de valeurs pour $(p_r)_{0\leq r\leq 9}$ qui fasse en sorte que R_N prenne plus souvent les valeurs $3, 4, 8, 9$?

Pour répondre à ces questions nous allons :

1. Simuler un grand nombre de panier d'achats et représenter des histogrammes des valeurs R_N correspondantes.
2. Calculer de façon exacte la loi de R_N , à l'aide d'un graphe pondéré et d'une matrice de transition.

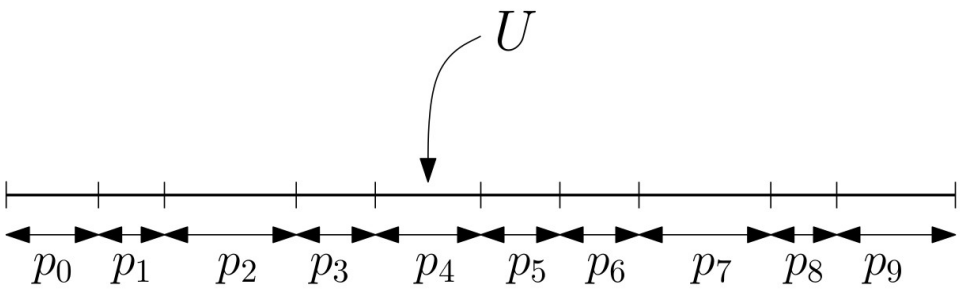
Simulations préliminaires : variables $X_1, \dots, X_N$

On cherche à définir une fonction `TiragePrix(Distribution)` qui prenne en entrée la liste des distributions des prix (par exemple on peut l'appliquer à `DistributionWalmart`) et qui renvoie à chaque tirage des variables X tirées selon la distribution `Distribution` .

Pour cela nous allons utiliser une variable aléatoire continue uniforme U dans l'intervalle $[0, 1]$. L'idée est alors de renvoyer $X = r$, où r est l'unique valeur dans $\{0, 1, \dots, 9\}$ telle que

$$p_0 + \cdots + p_{r-1} < U \leq p_0 + \cdots + p_r.$$

Cette méthode est illustrée dans la figure ci-dessous :



Pour simuler U nous allons utiliser la fonction `np.random.rand()` de python qui ne prend aucun argument en entrée, et renvoie une variable aléatoire continue uniforme dans l'intervalle $[0, 1]$.

```
def TiragePrix(Distribution):
    # entrée : vecteur "Distribution" =[p_0,p_1,p_2,...,p_9] de probabilités
    # sortie : tirage d'une variable X de loi "Distribution".
    VariableUniforme=np.random.rand()
    ProbasCumulees=np.cumsum(Distribution)  # fabrique le vecteur [p_0,p_0+p_1
    X=0   # X sera l'entier qu'on renvoie à la fin
    r=0   # r =0,1,...,9 désigne l'indice pour parcourir le vecteur "Distribut
    while VariableUniforme>ProbasCumulees[r]:
        # on parcourt ProbasCumulees jusqu'a tomber dans le bon intervalle
        X=X+1
        r=r+1
    return X

# test
Test=[]
for n in range(15):
    Test.append(TiragePrix(DistributionWalmart))

print(Test)
```

```
[1, 1, 9, 7, 5, 7, 7, 9, 4, 7, 6, 9, 7, 4, 7]
```

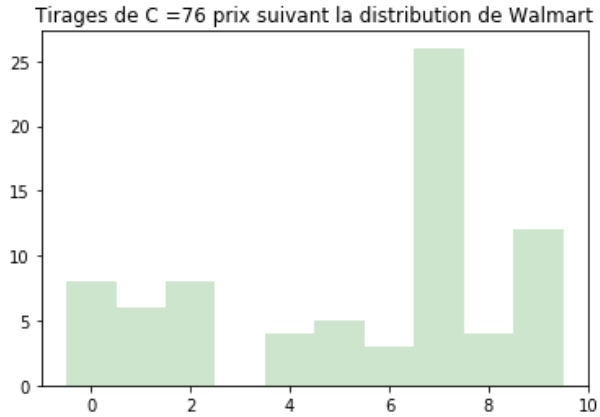
Visualiser des tirages de X

On commence par simuler les paniers de C clients qui achètent un seul produit selon la distribution `DistributionWalmart` , et on représente les résultats dans un histogramme.
Autrement dit, on tire au sort C fois la variable X .

```
def HistogrammeDesPrix(C):
    ListePrix=[TiragePrix(DistributionWalmart) for p in range(C)]
    plt.hist(ListePrix, bins=np.arange(-0.5,10.5),  facecolor='g', alpha=0.2)
    plt.title('Tirages de C ='+str(C)+' prix suivant la distribution de Walmar
    plt.show()

interact(HistogrammeDesPrix,C=(1,200,5))  # On lance la fonction avec un "widg
```

C 76




```
<function __main__.HistogrammeDesPrix(C)>
```

Expérience aléatoire : simulations de R_N

On peut maintenant faire des expériences aléatoires pour essayer de répondre aux Questions 1 et 2. Pour cela on va considérer C clients qui achètent chacun N produits. On va représenter les C valeurs de R_N dans un histogramme.

```
def TirageR(N,Distribution):
    # entrée : entier N , vecteur de probabilités "Distribution" de taille 10
    # sortie : Tirage d'une variable aléatoire R_N
    Tirages=[TiragePrix(Distribution) for n in range(N)]
    SommeDesPrix=np.sum(Tirages)
    return SommeDesPrix%10
```

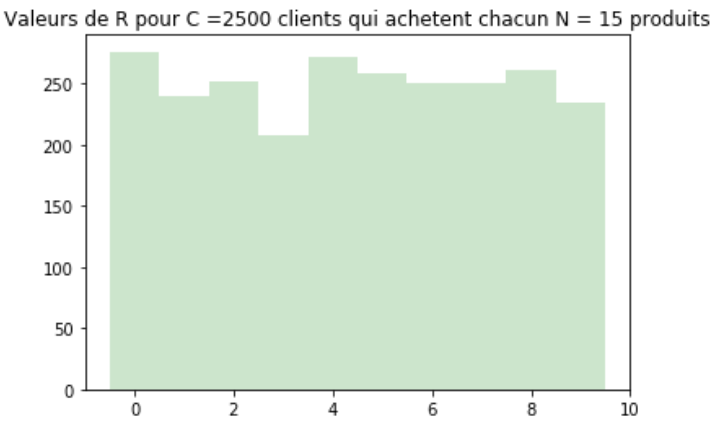
```
DistributionWalmart=[0.071 , 0.087 , 0.087 , 0.024 , 0.071 , 0.055 , 0.047 , 0
#Test de triche
DistributionTriche=[0 , 0 , 0 , 0.1 , 0.1 , 0 , 0 , 0 , 0 , 0.8]

def HistogrammeDesR(C,N):
    ListePrix=[TirageR(N,DistributionWalmart) for c in range(C)]
    plt.hist(ListePrix, bins=np.arange(-0.5,10.5),  facecolor='g', alpha=0.2)
    plt.title('Valeurs de R pour C ='+str(C)+' clients qui achetent chacun N =
    plt.show()

interact(HistogrammeDesR, C=(2000,3000,100), N=(1,30,1))  # On lance la fonction
```

C 2500

N 15




```
<function __main__.HistogrammeDesR(C, N)>
```

> Remarque. On observe que pour très peu de produits (dès $N = 4$ ou 5), la distribution de R_N semble uniforme.

On peut donc essayer de démontrer le résultat suivant :

> Theoreme. Pour toute distribution des prix $(p_r)_{0\leq r\leq 9}$, la variable aléatoire R_N converge vers la loi uniforme lorsque le nombre de produits tend vers l'infini :
>
> $$\forall r \in \{0, 1, \dots, 9\}, \qquad \mathbb{P}(R_N = r) \overset{N\to+\infty}{\to} \frac{1}{10}.$$

Calcul exact des probabilités : utilisation des matrices de transition

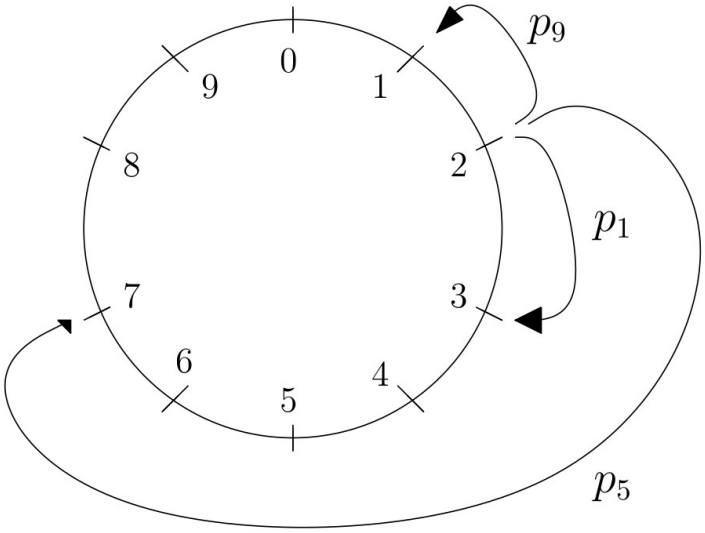
Nous allons adopter un point de vue un peu différent sur ce problème. Au lieu de faire des simulations nous allons calculer explicitement, pour une distribution donnée $(p_r)_{0\leq r\leq 9}$ et un entier N, les probabilités

$$\mathbb{P}(R_N = r).$$

L'objectif est de confirmer numériquement la convergence vers $1/10$.

Matrice de transition associée à la suite (R_n)

Ce calcul va se faire en utilisant des *matrices de transition*. Pour cela on remarque que la suite R_n peut être représentée comme une marche aléatoire sur le *graphe orienté pondéré* suivant (on ne représente que 3 arêtes) :



En effet, si $R_n = 2$ pour un certain n, alors $R_{n+1} = 7$ si X_{n+1} finit par un 5, ce qui arrive avec probabilité p_5. Plus formellement,

$$\mathbb{P}(R_{n+1} = 7 \mid R_n = 2) = p_5.$$

Plus généralement, pour tout $n \geq 0$ et tous $r, s \in \{0, 1, \ldots, 9\}$ on a

$$\mathbb{P}(R_{n+1} = s \mid R_n = r) = p_q,$$

où $q = s - r$ modulo 10.

La matrice de transition Q associée à la suite (R_n) est donc donnée par

$$Q = \begin{array}{c} 0 \\ 1 \\ 2 \\ \vdots \\ \\ \\ \\ \\ 9 \end{array} \begin{pmatrix} p_0 & p_1 & p_2 & \cdots & p_9 \\ p_9 & p_0 & p_1 & \cdots & p_8 \\ p_8 & p_9 & p_0 & \cdots & p_7 \\ & & & & \\ & & & \vdots & \\ & & & & \\ & & & & \\ p_2 & p_3 & p_4 & \cdots & p_1 \\ p_1 & p_2 & p_3 & \cdots & p_0 \end{pmatrix}.$$

Voici le code d'une fonction qui crée la matrice Q à partir des (p_r) .

```
def MatriceQ(Distribution):
    # entrée : vecteur des probabilités "Distribution"
    # sortie : matrice de transition associée
    Matrice=np.zeros([10,10])
    for r in range(10):
        for s in range(10):
            Matrice[r,s]= Distribution[(s-r)%10]
    return Matrice

# Test
print(MatriceQ(DistributionWalmart))
```

```
[[ 0.071  0.087  0.087  0.024  0.071  0.055  0.047  0.346  0.047
0.165]
 [ 0.165  0.071  0.087  0.087  0.024  0.071  0.055  0.047  0.346
0.047]
 [ 0.047  0.165  0.071  0.087  0.087  0.024  0.071  0.055  0.047
0.346]
 [ 0.346  0.047  0.165  0.071  0.087  0.087  0.024  0.071  0.055
0.047]
 [ 0.047  0.346  0.047  0.165  0.071  0.087  0.087  0.024  0.071
0.055]
 [ 0.055  0.047  0.346  0.047  0.165  0.071  0.087  0.087  0.024
0.071]
 [ 0.071  0.055  0.047  0.346  0.047  0.165  0.071  0.087  0.087
0.024]
 [ 0.024  0.071  0.055  0.047  0.346  0.047  0.165  0.071  0.087
0.087]
 [ 0.087  0.024  0.071  0.055  0.047  0.346  0.047  0.165  0.071
0.087]
 [ 0.087  0.087  0.024  0.071  0.055  0.047  0.346  0.047  0.165
0.071]]
```

On dispose alors du résultat suivant :

Theoreme. On note $(q_{r,s}^{(n)})_{0\leq r,s\leq 9}$ les coefficients de la matrice Q^n. Alors pour tout n, pour tous r, s on a

$$q_{r,s}^{(n)} = \mathbb{P}(R_n = s \mid R_0 = s).$$

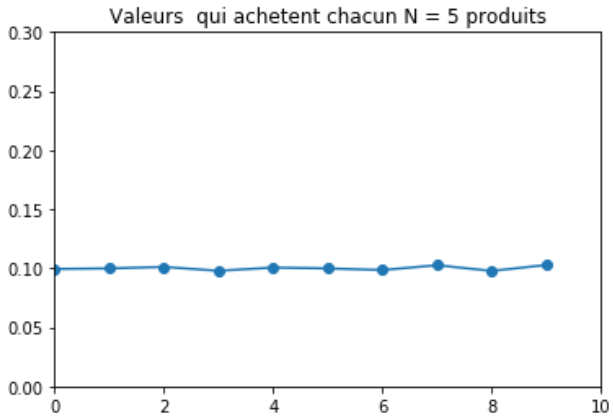
En extrayant la 1 ère ligne de Q^n, on a donc les valeurs des probabilités $\mathbb{P}(R_N = r)$.

```
DistributionWalmart=[0.071 , 0.087 , 0.087 , 0.024 , 0.071 , 0.055 , 0.047 , 0
#Test de triche
#DistributionTriche=[0 , 0 , 0 , 0.1 , 0.1 , 0 , 0 , 0 , 0 , 0.8]

def HistogrammeDesProbabilites(N):
    PuissanceMatrice=np.linalg.matrix_power(MatriceQ(DistributionWalmart),N)
    plt.axis([0,10,0,0.3])
    plt.plot([PuissanceMatrice[0,k] for k in range(10)], '-o')
    plt.title('Valeurs  qui achetent chacun N = '+str(N)+' produits')
    plt.show()

interact(HistogrammeDesProbabilites,N=(1,50,1))  # On lance la fonction avec u
```

N 5




```
<function __main__.HistogrammeDesProbabilites(N)>
```

Remarque.

1. Ce calcul confirme que pour la distribution `DistributionWalmart`, dès $N = 5$, la distribution de R_N est quasiment uniforme.
2. Pour la distribution `DistributionTriche`, la convergence est beaucoup plus lente. Cependant, pour $N = 50$ la loi de R_N est proche de l'uniforme.