Espace canonique

Dans tout cours de probabilités élémentaires qui se respecte, on passe un temps fou à écrire explicitement ce qu'est l'espace des événements dans les cas simples : $\Omega = \{1, \cdots, 6\}$ si on lance un dé à 6 faces, $\Omega = \{\text{pile}, \text{face}\}$, etc. Puis quand les concepts se compliquent un peu, généralement au moment de l'introduction des variables aléatoires continues, cette rigueur disparaît et Ω devient un objet indéfini, malléable à merci, sur lequel l'on suppose qu'il existe une bonne mesure de probabilités et une tribu.

C'est compréhensible dans le sens où seules sont intéressantes les valeurs des variables aléatoires, qui généralement se trouvent être des variables à valeurs réelles. On est donc souvent dans la situation où l'on a

$$X : \Omega \longmapsto E,$$

et tout ce que qui est intéressant (la loi de X, les moments de X, etc.) se calculent dans E qui est un sous-ensemble des entiers ou des réels. Mais alors pourquoi ne pas décréter directement que Ω est E et que X est l'application *identité* ? C'est ce que l'on appelle l'espace canonique sur lequel est défini la variable X.

Ainsi, si l'on considère le résultat du lancer de 3 dés à 6 faces, on prendre $\Omega = \{1, \cdots, 6\}^3$ et les variables aléatoires X_1, X_2, X_3 sont les applications coordonnées :

$$\begin{aligned} X_i : \Omega = \{1, \cdots, 6\}^3 &\longrightarrow \{1, \cdots, 6\} \subset \mathbf{N} \\ \omega = (\omega_1, \omega_2, \omega_3) &\longmapsto X_i(\omega) = \omega_i. \end{aligned}$$

Imaginons maintenant que l'on joue 10 fois à pile ou face, l'espace d'états peut-être pris comme $\{0, 1\}^{10}$ ou $\{\text{pile}, \text{face}\}^{10}$, ou plus généralement comme le produit cartésien 10 copies de n'importe quel espace à deux éléments. Si l'on gagne 1 à chaque occurrence de *pile* et que l'on perd -1 à chaque occurrence de *face*, on introduira les variables aléatoires :

$$\begin{aligned} X_i : \Omega = \{\alpha, \beta\}^{10} &\longrightarrow \{-1, 1\} \subset \mathbf{N} \\ \omega = (\omega_1, \cdots, \omega_{10}) &\longmapsto \begin{cases} +1 & \text{si } \omega_i = \alpha \\ -1 & \text{si } \omega_i = \beta. \end{cases} \end{aligned}$$

Dans un jeu infini de *pile/face*, l'espace naturel est donc $\{-1, 1\}^{\mathbf{N}}$ ou $\{0, 1\}^{\mathbf{N}}$, ce qui pose une question sur la façon dont on définit une mesure de probabilités sur un espace de cardinal infini. Mais tout d'abord, il y a infini et infini. Commençons par étudier le seul infini qui soit au programme des CPGE, c'est-à-dire l'infini dénombrable, $\aleph_0$ si on veut faire érudit.

Dénombrabilité

Définition 1. Un ensemble est fini s'il est en bijection avec un certain $\{1, \cdots, n\}$.

Définition 2. Un ensemble est dit dénombrable s'il est en bijection avec $\mathbf{N}$, l'ensemble des entiers naturels.

1. $2\mathbf{N}$, l'ensemble des entiers pairs, qui intuitivement contient deux fois moins d'éléments que $\mathbf{N}$ est bien en bijection avec $\mathbf{N}$ et est donc dénombrable. La bijection est donnée par $n \mapsto 2n$.
2. $\mathcal{P}(\mathbf{N})$, l'ensemble des parties de $\mathbf{N}$, ne l'est pas. Il est en bijection avec l'ensemble des réels $\mathbf{R}$. En effet, pour une partie A de $\mathbf{N}$, on peut définir « la suite indicatrice » $\chi_A(n)$ par

$$\begin{cases} \chi_A(n) & = 1 \text{ si } n \in A, \\ & = 0 \text{ sinon.} \end{cases}$$

Puis à cette suite, on peut associer le réel x_A défini par :

$$x_A = \sum_{n=1}^{\infty} 2^{-n} \chi_A(n).$$

Il est évident que $\mathcal{P}(\mathbf{N})$ est alors en bijection avec $\{0,1\}^{\mathbf{N}}$. Etant donné qu'à tout réel entre 0 et 1, on peut associer son développement diadique, il existe une injection $[0,1]$ dans $\{0,\ 1\}^{\mathbf{N}}$ donc $\mathcal{P}(\mathbf{N})$ n'est pas dénombrable. [1]

La dénombrabilité est stable par réunion et produit cartésien.

Théorème 3. Si X et Y sont deux ensembles dénombrables alors $X \times Y$ et $X \cup Y$ sont dénombrable.

Tout est basé sur le résultat essentiel qui stipule de $\mathbf{N} \times \mathbf{N}$ est en bijection avec $\mathbf{N}$. La bijection est construite comme indiquée sur la figure 0.1. Par exemple, l'élément $(2,0)$ est envoyé sur 3 et l'élément $(0,2)$ est envoyé sur 5.

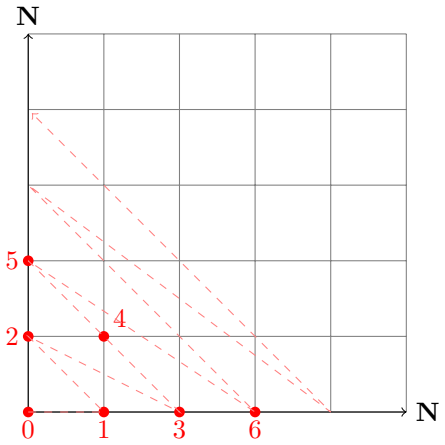

Figure 0.1– Bijection de N × N avec N.

Corollaire 4. L'ensemble des entiers relatifs, $\mathbf{Z}$, est dénombrable.

Le théorème suivant est loin d'être trivial (cf. [1]).

Théorème 5. S'il existe une injection de l'ensemble X dans l'ensemble Y et une injection de Y dans X alors il existe une bijection de X dans Y.

Corollaire 6. L'ensemble des rationnels, $\mathbf{Q}$ est dénombrable.

Démonstration. Il existe évidemment une injection de $\mathbf{N}$ dans $\mathbf{Q}$ et par construction de $\mathbf{Q}$, il existe une injection de $\mathbf{Q}$ dans $\mathbf{Z} \times \mathbf{Z}$, qui d'après le théorème précedent est en bijection avec $\mathbf{N}$. Par conséquent, $\mathbf{Q}$ est en bijection avec $\mathbf{N}$. □

1. Par conséquent, notre espace canonique pour la suite infinie de *pile/face* n'est pas dénombrable. Cet espace sort donc du cadre strict du programme de CPGE mais il est nécessaire de le considérer car sinon on ne peut pas définir ne serait-ce que la loi géométrique. En effet, celle-ci se définit comme le nombre de tentatives nécessaires avant un succès dans une suite d'épreuves indépendantes et identiques. Comme il n'y a pas de majorant fixe du nombre de tentatives, on est bien obligé de travailler sur $\{0,\ 1\}^{\mathbf{N}}$, d'où un léger souci d'ordre éthique. On se gardera donc de s'interroger sur les prémices théoriques de cet ensemble dans le cadre des CPGE puisqu'en fait « tout marche ».

Le matheux est donc content, il a trouvé un cadre relativement naturel et général pour faire des probabilités dites « discrètes ». L'informaticien l'est beaucoup moins : si un ensemble dénombrable est en bijection avec $\mathbf{N}$, cette bijection n'est pas unique et n'est pas toujours facile à représenter informatiquement parlant.

Représentation informatique

Intéressons-nous de plus près à la façon de représenter un espace probabilisé dans un langage qui tend à devenir celui de référence pour les CPGE : Python. Le bon outil pour coder un espace probabilisé, nécessairement fini, semble les *dictionnaires*.

Lancer de 3 dés

On construit l'espace de probabilités pour le lancer de 1 dé, sous forme d'un dictionnaire :

Création de l'espace probabilisé pour un dé à 6 faces

```
1 from numpy import *
2 import numpy as np
3 import itertools
4 de=np.arange(6)+1
5 proba=np.ones(6)/6
6 espace_de=dict(zip(de,proba))
7 espace_de
```

Les trois premières lignes importent les outils nécessaires. En ligne 4, on crée une liste de 1 à 6. La commande permet de constituer une liste de paires en prenant les éléments de même rang de chacune de deux listes, ici *de* et le tableau de six cases, toutes égales à 1/6. Ce qui donne le résultat suivant.

```
1 {1: 0.16666666666666666,
2  2: 0.16666666666666666,
3  3: 0.16666666666666666,
4  4: 0.16666666666666666,
5  5: 0.16666666666666666,
6  6: 0.16666666666666666}
```

Ensuite, on peut tensoriser cette construction pour obtenir l'espace correspondant au lancer de trois dés. La fonction Python itertools.product est idéalement faite pour cela puisqu'elle permet de construire le produit cartésien de plusieurs ensembles. Dans le cas de variables aléatoires indépendantes, donc de mesure produit, on obtient la probabilité sur l'espace produit (cartésien) en faisant le produit tensoriel des vecteurs de probabilités sur chacun des espaces [2].

2. Il faut faire attention à convertir le *array* résultant en *list* pour pouvoir appliquer la fonction *zip*. D'autres solutions plus faciles existent sans doute

Création de l'espace probabilisé pour trois dés à 6 faces

```
proba_trois_de=list(np.tensordot(np.tensordot(proba,proba,axes=0),proba,axes=0).
    reshape(1,6**3)[0])
espace_trois_de=dict(zip(list(itertools.product(de,repeat=3)),proba_trois_de))
```

Le résultat qui contient 216 lignes est trop long pour être reproduit mais commence comme suit.

```
{(1, 1, 1): 0.0046296296296294,
 (1, 1, 2): 0.0046296296296294,
 (1, 1, 3): 0.0046296296296294,
 (1, 1, 4): 0.0046296296296294,
 (1, 1, 5): 0.0046296296296294,
 (1, 1, 6): 0.0046296296296294,
 (1, 2, 1): 0.0046296296296294,
 (1, 2, 2): 0.0046296296296294,
```

Une question amusante est de calculer dans ce cas, la probabilité que la somme des faces soit paire. On peut le faire astucieusement comme nous le verrons plus tard, on peut aussi le faire de façon très très brutale en énumérant tous les états qui correspondent à cette condition puis en sommant les probabilités de chacun de ces singletons.

```
somme_paire=[k for k in espace_trois_de if ((k[0]+k[1]+k[2]) % 2) ==0]
```

Dans ce code, la variable `somme_paire` contient la liste des triplets dont la somme des composantes est paire. Pour avoir la probabilité de cet ensemble, il n'y a plus qu'à faire la somme des deuxièmes composantes du dictionnaire pour les entrées correspondant aux éléments de `somme_paire`.

```
np.sum(espace_trois_de[k] for k in somme_paire)
```

On obtient 0.50000000000000078 (alors que le résultat exact est 0.5) en raison des erreurs d'approximation dans les calculs. Une variante que permet d'étudier ce code et que ne permet pas de faire les calculs explicites, consiste à utiliser un truc biaisé et à se poser la même question.

« Biaiser les dés » se traduit juste par le changement du vecteur *proba*, le reste du code étant non modifié. Truquons les dés de sorte que la probabilité d'apparition d'un 1 soit 7/24 et celle de 6 (la face opposée) soit de 1/24. La seule ligne à changer est

```
proba=np.array([7./24,1./6,1./6,1./6,1./6,1./24])
```

On trouve alors que la probabilité que la somme des faces soit paire tombe à 0.4921875.

Permutations

L'un des exercices classiques de probabilités est le problème des mariages. On considère N couples dont les membres de mélangent lors d'une soirée passablement éméchée. Au moment de partir, les couples se reforment de façon aléatoire. On s'intéresse à la probabilité qu'aucun couple de départ ne se soit correctement reformé. Cet exercice comporte deux difficultés : la première est de trouver le bon modèle, c'est-à-dire le bon espace de probabilités. Affectons un numéro à chaque couple, numéro dont hérite chacun des membres du couple. Au départ, les 1 vont ensemble, les 2 aussi, etc. A la fin, on obtient une situation qui se résume par un tableau de la forme

$$\begin{pmatrix} 1 & 2 & \dots & N \\ 4 & 3 & \dots & \end{pmatrix}$$

où la première (resp. deuxième) colonne signifie que l'un des membres du couple 1 (resp. 2) est parti avec l'un des membres du couple 4 (resp. 3). Il est clair que chacun des nombres de 1 à N apparaît une et une seule fois dans chacun des lignes. Ce tableau peut donc s'interpréter comme une bijection de $\{1, \cdots, N\}$ dans lui-même, c'est-à-dire une permutation à N éléments. Cet ensemble usuellement noté $\mathfrak{S}_n$ contient $n!$ éléments. La question posée revient à déterminer la probabilité qu'une permutation « choisie au hasard » n'ait pas de point fixe.

Enumérer informatiquement les permutations n'est pas si simple que cela. La supériorité de SAGE http://www.sagemath.org sur les autres implémentations de Python telles Jupyter, iPython, etc. est de posséder en plus des capacités de calcul formel et de combinatoire, sans perdre les avantages de l'interface graphique. Ainsi le code suivant permet de créer un objet qui contient les 120 permutations de $\{1, \cdots, 5\}$. La propriété *.list()* permet d'avoir la liste intégrale des éléments

```
p=Permutations(5)
p.list()
```

On obtient le début de résultat suivant

```
WARNING: Output truncated!
full_output.txt


[[1, 2, 3, 4, 5],
 [1, 2, 3, 5, 4],
 [1, 2, 4, 3, 5],
 [1, 2, 4, 5, 3],
 [1, 2, 5, 3, 4],
 [1, 2, 5, 4, 3],
 [1, 3, 2, 4, 5],
 [1, 3, 2, 5, 4],
 [1, 3, 4, 2, 5],
 [1, 3, 4, 5, 2],
 [1, 3, 5, 2, 4],
 [1, 3, 5, 4, 2],
 [1, 4, 2, 3, 5],
 [1, 4, 2, 5, 3],
 [1, 4, 3, 2, 5],
....
```

Avec cet outil, on peut essayer de répondre à la question du début et en fait à une question plus générale : si l'on choisit une permutation σ au hasard (c'est-à-dire selon la loi uniforme sur un ensemble à $N!$ éléments), on compte son nombre de points fixes, noté $X_N(\sigma)$. On peut prouver que la loi de X_N tend

vers une loi de Poisson de paramètre 1 lorsque N tend vers l'infini. Pour quantifier cette convergence, il nous faut définir une distance entre mesures de probabilités. Il en existe plusieurs possibles, celle qui va nous intéresser ici est celle appelée distance en variation totale. Pour deux mesures de probabilités μ et ν sur $\mathbf{N}$,

$$\mathrm{dist}_{\mathrm{VT}}(\mu,\,\nu) = \frac{1}{2}\sum_{k\in\mathbf{N}} |\mu(\{k\}) - \nu(\{k\})|.$$

Le code suivant met en place cette analyse. La fonction `pt_fixe` retourne le nombre de points fixes d'une permutation rentrée sous forme d'un tableau de N cases. La fonction `np.arange(1,1N)+` renvoie un tableau de N cases contenant les entiers de 1 à N. La différence entre les deux tableaux se fait case par case, donc une case de la différence est à 0 à chaque fois que `permut` a un point fixe. Ensuite la comparaison à 0 se fait aussi case par case, ce qui fait que le résultat est un tableau de N cases valant chacune `True` ou `False` selon que la case correspondante est à 0 ou pas. La fonction `np.sum` convertit les `True` et +1 et les `False` en 0 et fait la somme. Cela renvoie donc bien le nombre total de points fixes de `permut` sans faire de boucle explicite (avec une fonction `for`).

```
def pt_fixe(permut,N):
    return np.sum((np.asarray(permut)-np.arange(1,1+N))==0)

N=8
p=Permutations(N)
q=[pt_fixe(k,N) for k in p]
hist_pt_fixe=np.histogram(q,bins=N,range=(0,N),normed=True)
hist_poisson=np.asarray([poisson.pmf(k,1.) for k in np.arange(0,N)])
distance=(np.sum(abs(hist_pt_fixe[0]-hist_poisson))+1-poisson.cdf(N,1.))/2
distance
```

Après avoir calculé le nombre de points fixes pour chacune des permutations, on utilise la fonction `histogram` qui permet de calculer le nombre de permutations pour lesquelles ce nombre vaut 0, puis 1, etc. La variable `distance` contient la distance en variation totale entre la loi de X_N et la loi de Poisson de paramètre 1.

N	Distance	Durée
5	0.027	3 ms
6	0.008	40 ms
7	0.002	260 ms
8	0.000 5	1.95 s
9	0.000 1	17.5 s

On constate que les temps de calcul deviennent rapidement prohibitifs, comme on pouvait s'y attendre en présence d'un processus qui comprend un nombre exponentiellement croissant d'étapes. C'est la limitation fondamentale à l'approche exhaustive du calcul des probabilités même sur des ensembles finis en utilisant un ordinateur.

Sommabilité

Il ressort des exemples précédents qu'il n'y a pas forcément d'ordre sur l'espace de probabilités. Cela pose un problème pour définir la convergence de *séries*. D'où le chapitre sur les familles sommables dont on trouvera un excellent résumé dans la page Wikipedia https://fr.wikipedia.org/wiki/Famille_sommable.

En résumé, si les quantités que l'on somme sont positives (ce qui est souvent le cas), pour qu'une somme soit bien définie, il suffit que les sommes finies soient majorées : il suffit qu'il existe $M < \infty$ tel

que

$$\sup_{J \text{ fini} \subset \Omega} \sum_{\omega \in J} u(\omega) \leq M,$$

pour que la somme $\sum_{\omega \in \Omega} u(\omega)$ soit bien définie au sens des familles sommables. Ensuite, si u n'est pas à valeurs positives mais seulement à valeurs dans $\mathbf{R}$, $\mathbf{C}$ ou des produits de ses espaces, il suffit que $\|u\|$ satisfasse le critère précédent, i.e. que la famille soit absolument sommable.

En bref, rien de magique, seule l'absolue sommabilité a un intérêt, comme c'est le cas dans toute la théorie de la mesure.

BIBLIOGRAPHIE

[1] K. Kuttler, Modern analysis, Studies in Advanced Mathematics, CRC Press, 1998 (English).