Probabilités et marches aléatoires

Vincent Delecroix Bruno Grenet Damien Jamet

EJC IM 2018

Dans cette feuille de travail nous proposons quelques expériences autour du chapitre 1 "Combinatoire et couplage probabiliste" du livre "Informatique Mathématique. Une photographie en 2018". Un des objectifs est de construire des générateurs aléatoires d'objets combinatoires à partir de primitives très simples.

1 Aléatoire en SageMath

Nous utiliserons seulement les fonctions de génération quasi-aléatoires élémentaires suivantes :

— `random()` : un flottant uniforme dans $[0, 1]$,

— `randint(i,j)` : un entier uniforme dans $\{i, i + 1, \ldots, j\}$.

Pour faire le lien avec les variables aléatoires du chapitre 1 vous devez considérer que des appels successifs à la fonction `random()` est une *réalisation* d'une suite de variables indépendantes uniformes.

Une construction pratique de Python est la *liste en compréhension*. C'est ce qui correspond en mathématiques à la notation ensembliste

$$S = \{x : x \in \mathbb{N}, x^2 > 3\}.$$

Voici l'exemple des nombres premiers < 20 et d'un échantillon de loi uniforme

```
sage: l = [n for n in range(1, 20) if is_prime(n)]
sage: l = [random() for _ in range(10)]
```

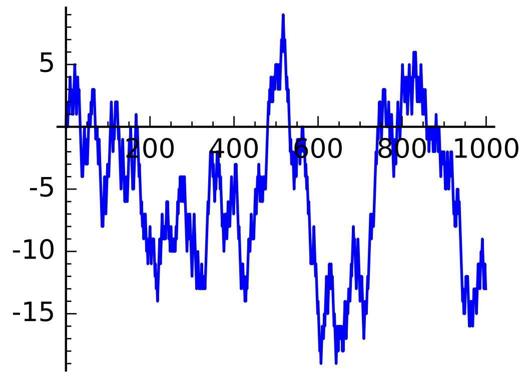
Une autre manière de construire des listes est via la méthode `append` (qui ajoute un élément à une liste). La petite boucle suivante simule une marche aléatoire de 1000 pas sur $\mathbb{Z}$ dont chaque pas consiste à choisir entre $+1$ et -1 avec probabilité $1/2$

```
sage: x = 0   # position courante
sage: W = []  # marche
sage: for i in range(1000):
....:     W.append(x)
....:     x = x + 1 - 2*randint(0,1)
```

Notez la boucle avec `range` qui permet de répéter 1000 fois une opération.

On peut visualiser la suite `W` en utilisant la commande

```
sage: list_plot(W, plotjoined=True)
```



2 Combinatoire

On peut définir les nombres binomiaux $B(n,m) = \frac{n!}{m!(n-m)!}$ par l'équation

$$(1+x)^n = \sum_{m=0}^{n} B(n,m)x^m. \tag{1}$$

1. En utilisant la relation $(1+x)^n = (1+x)(1+x)^{n-1}$, montrer que les nombres binomiaux vérifient la récurrence

$$B(n,m) = B(n-1,m) + B(n-1,m-1). \tag{2}$$

2. Quel est le rapport entre les nombres binomiaux et la marche aléatoire sur $\mathbb{Z}$?

Pour fabriquer les listes des nombres binomiaux $B(n,m)$ à n fixé, la récurrence (2) se traduit en des boucles imbriquées

```
sage: B = [1]    # liste pour n=0
sage: for n in range(1, 5):
....:     B.append(1)
....:     for i in range(n-1, 0, -1):
....:         B[i] = B[i] + B[i-1]
....:     print(B)
[1, 1]
[1, 2, 1]
[1, 3, 3, 1]
[1, 4, 6, 4, 1]
```

Notez la fonction `range(n - 1, 0, -1)` qui permet de faire une boucle de `n - 1` (inclus) à `0` (exclus) avec un pas `-1`.

3. Pour les valeurs de $n = 3, 5, 10, 30, 100$ faire un graphique de la suite $m \mapsto B(n,m)$ (utiliser `list_plot` comme on l'a fait précdemment et possiblement `graphics_array` pour inclure plusieurs graphiques dans une image).

Les nombres de Stirling (de première espèce non signés) $S(n,m)$ sont définis par l'équation

$$x^{(n)} := x(x+1)(x+2)\dots(x+n-1) = \sum_{m=0}^{n} S(n,m)x^m. \tag{3}$$

4. En utilisant l'équation $x^{(n)} = (x+n)x^{(n-1)}$, montrer que les nombres de Stirling satisfont la récurrence

$$S(n,m) = (n-1)S(n-1,m) + S(n-1,m-1). \tag{4}$$

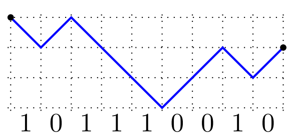
5. Modifier le programme pour les binomiaux pour qu'il génère les nombres de Stirling.
6. Quel est le lien entre les nombres de Stirling et les permutations ?
7. Pour les valeurs de $n = 3, 5, 10, 30, 100$ faire un graphique de la suite $m \mapsto S(n,m)$.
8. Montrer que les nombres eulériens $A(n,m)$ (voir chapitre 1) vérifient

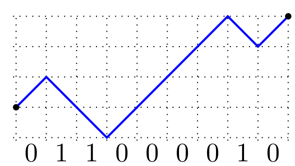
$$A(n,m) = (m+1)A(n-1,m) + (n-m)A(n-1,m-1). \tag{5}$$

9. Pouvez-vous donner une "formule" pour les nombres eulériens de la même forme que (1) ou (3) ?
10. Utiliser la récurrence (5) pour engendrer les nombres eulériens.
11. Pour les valeurs de $n = 3, 5, 10, 30, 100$ faire un graphique de la suite $m \mapsto A(n,m)$.
12. Que constatez-vous ? Quels sont les liens entre les graphiques de $B(n,m)$, $S(n,m)$ et $A(n,m)$?

3 Excursion brownienne discrète

On se propose maintenant d'implanter la génération aléatoire d'analogues discrets de l'excursion brownienne. On considère des marches aléatoires sur $\mathbb{Z}$ représentées par des mots finis de $\{0,1\}^*$ (0 indique un pas montant et 1 un pas descendant). Plus formellement, la marche associée à $s = s_1 s_2 \ldots s_n \in \{0,1\}^n$ est la suite d'entiers $(w_0, w_1, \ldots, w_n) \in \mathbb{Z}^{n+1}$ définie par $w_0 = 0$ et pour tout $i \in \{1, \ldots, n\}$ $w_i = w_{i-1} + 1$ si $s_i = 0$ et $w_i = w_{i-1} - 1$ si $s_i = 1$. On pourra voir deux exemples de mots s ainsi que les marches associées ci-dessous.





3.1 Génération aléatoire à évaluation fixée

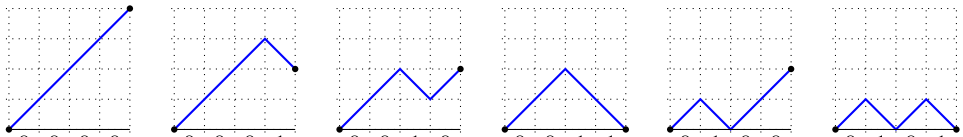
Étant donné un mot s dans $\{0,1\}^n$, le *vecteur d'évaluation* de s est le vecteur $(|s|_0, |s|_1)$ des nombres d'occurrences de 0 et 1 dans w. En reprenant les exemples ci-dessus, les vecteurs d'évaluation de 101110010 et 011000010 sont respectivement $(4,5)$ et $(6,3)$. On remarquera que le point final de la marche w_n est égal à $|s|_0 - |s|_1$.

Nous proposons d'implanter une chaîne de Markov non-homogène pour générer uniformément un mot de $\{0,1\}^*$ dont le vecteur d'évaluation (n_0, n_1) est donné. Pour cela, on choisit la première lettre avec une loi de Bernoulli de paramètre $p = n_1/(n_0+n_1)$. Si la première lettre est un 0 alors on recommence avec le nouveau vecteur d'évaluation $(n_0 - 1, n_1)$ sinon, on recommence à partir de $(n_0, n_1 - 1)$.

13. Justifier cette méthode.
14. La programmer dans SageMath.
15. Faire des graphiques pour les évaluations $(30,30)$, $(30,10)$ et $(1000,1000)$.
16. Décrire la renormalisation appropriée pour qu'on ait une convergence en loi vers l'excursion brownienne.
17. Refaire le graphique pour l'évaluation $(1000,1000)$ à l'échelle de l'excursion brownienne.

3.2 Marches positives et équilibrées

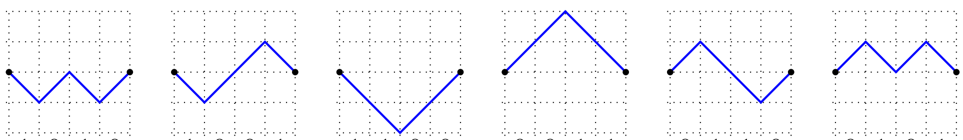
Une marche est dite *positive* si le mot associé s de $\{0,1\}^n$ est tel que pour tout préfixe u de s on ait $|u|_0 \geq |u|_1$. Autrement dit, toutes les étapes de la marche w sont ≥ 0. On note $\mathcal{P}_n \subset \{0,1\}^n$ l'ensemble des marches positives de longueur n et $\mathcal{P}_n^{(k)}$ celles parmi $\mathcal{P}_n$ qui finissent en k. Ci-dessous sont dessinées les 6 marches positives de longueur 4.



Sur ces exemple, les coordonnées d'arrivée sont respectivement 4, 2, 2, 0, 2 et 0.

On dit qu'une marche est *équilibrée* si le mot s vérifie $|s|_0 = |s|_1$. Autrement dit, il s'agit d'un élément dont l'évaluation est (m, m) pour un certain entier $m \geq 0$. Ou encore, ce sont les marches qui terminent en $w_{2m} = 0$.

On note $\mathcal{B}_{2m}^{(k)} \subset \{0,1\}^{2m}$ les mots équilibrés de $\{0,1\}^{2m}$ pour lesquels la marche correspondante possède exactement k pas descendants de 0 vers -1. Ci-dessous sont dessinées les 6 marches équilibrées de longueur 4.



Sur ces exemples, les nombre de descentes de 0 vers -1 sont respectivement 2, 1, 1, 0, 1, 0.

Un *mot de Dyck* est un élément de $\mathcal{P}_{2m} \cap \mathcal{B}_{2m}$. Finallement, étant donné un mot $u \in \{0,1\}^*$ on note $\overline{u}$ le mot consistant à échanger les lettres 0 et 1.

18. Montrer que tout mot w de $\mathcal{P}_{2m}^{(2k)}$ se décompose de manière unique en $w = d_0 1 d_1 1 d_2 1 \ldots 1 d_2 k$ où les facteurs d_i sont des mots de Dyck.

19. À un mot w de $\mathcal{P}_{2m}^{(2k)}$ décomposé en $w = d_0 1 d_1 1 \ldots 1 d_{2k}$ comme dans la question précedente, on associe $\phi(w) = d_0 0 \overline{d_1} 1 d_2 0 \ldots 0 \overline{d_{2k-1}} 1 d_{2k}$.

20. Montrer que ϕ induit une bijection de $\mathcal{P}_{2m}^{(2k)}$ vers $\mathcal{B}_{2m}^{(k)}$ et implanter cette fonction dans Sage.

21. Décrire la bijection inverse et l'implanter dans Sage.

3.3 Génération des marches positives par rejet

On s'intéresse à une autre méthode de génération aléatoire uniforme de marches positives. On va générer des marches de taille n et faire du rejet anticipé lorsqu'elle n'est pas positive. Plus précisément, on génère des bits aléatoires les uns après les autres. Si la marche associée arrive en -1 on efface tout et on recommence. On s'arrête lorsque la longueur de la suite est n.

22. Implanter cette méthode de génération aléatoire pour $\mathcal{P}_n$.

23. Donner une génération aléatoire alternative pour $\mathcal{B}_{2m}$ en utilisant la génération aléatoire de $\mathcal{P}_{2m}$ et la bijection de l'exercice précédent.

24. Déterminer la moyenne du nombre de bits aléatoires nécessaires pour chacune des deux méthodes. Quelle méthode est la plus économe en aléa ?