Probabilités, Statistiques, Combinatoire : TM3

Simulations

Pour la génération aléatoire, on utilise le module `random` chargé avec la commande `import random` placée au début de votre fichier.

On utilisera deux fonctions du module `random` :

— `random.randint(a,b)`, un générateur aléatoire qui renvoie un nombre entier choisi au hasard (uniformément) dans $[[a, b]]$.

— `random.random()`, qui retourne un flottant compris entre 0 et 1, selon la "loi uniforme sur $[0, 1]$" : la probabilité de retourner une valeur située dans n'importe quel intervalle $[x, y]$ (avec $0 \leq x \leq y \leq 1$) est $y - x$, la longueur de l'intervalle.

Vous allez écrire beaucoup de fonctions **sans paramètres**, dont l'objet est de simuler une expérience aléatoire : à chaque appel de la fonction, celle-ci retournera un résultat possiblement différent (et les résultats d'appels successifs sont considérés comme indépendants ; c'est ce qui est plus ou moins garanti par les générateurs pseudo-aléatoires de la bibliothèque `random`). Pour une fonction `simulation()`, il existe deux écritures :

— `simulation()` représente le résultat d'un *appel* à la fonction : celle-ci retourne une valeur, possiblement nouvelle à chaque appel ;

— `simulation` (sans les parenthèses) représente la fonction elle-même ; on peut passer une fonction à une autre comme paramètre, comme dans beaucoup de langages (en `C` cela passe par un pointeur de fonction ; en `Ocaml` c'est très classique).

Conseils pour l'écriture de vos simulations. L'objectif de ce TM est de vous faire écrire des programmes qui simulent une expérience aléatoire simple, mais dont la probabilité de "succès" n'est pas forcément simple à deviner. Il a pour but de vous montrer que l'écriture de programmes de simulation permet (parfois) de remplacer des calculs de probabilités complexes. Pour qu'il remplisse son rôle, il est indispensable que les programmes de simulation soient les plus simples possibles : il doit être évident, pour quelqu'un qui n'a pas écrit le programme mais qui a lu la description du problème, que le programme simule bien la situation décrite. Plus ce sera le cas, plus vous pourrez être confiants dans les résultats obtenus.

Nous allons simuler des matches de différents sports de balle, dans un modèle très simple.

Dans un certain nombre de sports de balle (tennis, volley, tennis de table), on joue un certain nombre de coups ; à chaque coup, un des deux joueurs (ou équipes) *sert* (met la balle en jeu) et, à la fin du coup, l'un des joueurs gagne le point ; le vainqueur du match est le résultat d'une séquence de coups, selon des règle de "comptage des points" qui varient d'un jeu à l'autre.

— Au tennis de table, celui qui remporte un coup marque un point ; le premier qui atteint 11 points avec au moins 2 points d'avance remporte un set ; le premier qui remporte un nombre prédéterminé de sets remporte le match. Dans un set, chaque joueur sert pour 2 coups consécutifs, puis c'est à son adversaire de servir ; si l'on atteint 10-10, le service change tous les coups jusqu'à la fin du set.

— Au volley, le principe est le même mais le nombre de points à remporter pour un set est de 25 points ; dans un même set, le camp qui vient de marquer sert.

— Jusqu'en 1998, la marque au volley était différente : un point n'était marqué que si le camp qui servait remportait le coup ; pour le camp adverse, remporter le coup ne lui permettait que de récupérer le service (le nombre de points pour un set était plus bas, 15 seulement).

— Au tennis, il y a trois niveaux de marque : le premier qui remporte 4 points avec 2 points d'avance, remporte un jeu ; le premier qui remporte 6 jeux avec 2 jeux d'avance, remporte un set ; le premier qui remporte un nombre prédéterminé de sets, remporte le match. Le même joueur sert pour tous les coups d'un même jeu, puis le service alterne. On oublie la règle du jeu décisif !

Vous allez écrire des programmes qui simulent ces systèmes de marque, du point de vue d'un joueur donné A, sous l'hypothèse (extrêmement simplificatrice) que les différents coups d'un match sont indépendants. Dans un premier temps, on supposera que le joueur a toujours la même probabilité p de remporter chaque coup, qu'il ait le service ou non ; par la suite on affinera la simulation en introduisant deux paramètres p et q : p représentera la probabilité que A remporte un coup lorsqu'il sert, q la probabilité qu'il remporte un coup lorsque l'adversaire sert. (Normalement, $p > q$ sauf au volley, où servir est censé être un désavantage car c'est l'adversaire qui a la première occasion d'attaquer.)

Vos simulations seront écrites du point de vue du joueur A donné : celui qui entame le match. C'est-à-dire que le résultat 1 représentera la victoire (à un point, à un set, à un match) pour le joueur A.

1. Écrire une fonction `SimuleCoup(x)`, qui retournera le résultat d'un coup du point de vue du joueur A : 1 si le joueur remporte le coup, 0 sinon. Le paramètre x sera la probabilité de remporter le coup pour le joueur.
2. Écrire une fonction `SimuleSetTable(x)`, qui retourne le résultat d'un set de tennis de table pour un joueur dont la probabilité de remporter chaque coup est de x. La fonction `SimuleSetTableSimplifie(x)` pourra servir d'exemple ; elle simule un set, sans la règle des deux points d'avance.
3. En simulant un grand nombre (plusieurs milliers) de sets, estimer la probabilité de remporter un set si l'on a probabilité 0.7, 0.6, 0.55, 0.51 de remporter chaque coup. Que devrait être (exactement) cette probabilité pour $x = 0.5$? Estimer aussi l'effet de la règle des deux points d'écart sur la probabilité de remporter un set. On peut chercher à répondre à cette question de deux manières différentes : estimer l'écart entre la probabilité de gagner avec la règle des deux points d'écart et sans cette règle ; ou, pour une probabilité p donnée de gagner le point, chercher quelle probabilité p' donne les mêmes chances de gagner sans règle des deux points, qu'avec la probabilité p et la règle des deux points).
4. Écrire de même une fonction `SimuleMatchTable(x)`, qui simule un match en 3 sets gagnants pour un joueur qui a probabilité x de remporter chaque coup. Déterminer expérimentalement la probabilité de remporter un match en fonction de la probabilité de remporter chaque coup.
5. Faire de même pour le tennis (avec une fonction pour la simulation d'un jeu, une fonction pour la simulation d'un set, une fonction pour la simulation d'un match en 3 sets gagnants).

   Déterminer expérimentalement à partir de quelle probabilité de remporter chaque balle, un joueur a 9 chances sur 10 de remporter un match en 3 sets gagnants.
6. Reprendre vos fonctions, et en écrire des versions `Bis` avec deux paramètres p et q à la place de x : p représentera la probabilité que le joueur A gagne le point s'il sert, q la probabilité qu'il gagne le point si c'est son adversaire qui sert ; on considèrera que le joueur sert la première balle du match. Ajouter une fonction pour simuler l'ancienne règle du volley, où seule l'équipe qui sert peut marquer un point.