TP BCPST
Probabilités, dénombrement, etc.

Julien Reichert

Une histoire de probabilités...

Partons d'une citation entendue lors d'un tournoi en mai 2023 [traduite en français] : « Soit les trois carreaux sont dans la même main, soit ils sont répartis deux et un. C'est une chance sur deux. »

Le but de ce sujet est de rétablir la vérité mathématique sur la base d'une modélisation d'une part et d'un calcul rigoureux d'autre part.

Les cartes seront modélisées par des booléens : `True` pour les trois carreaux, `False` pour les autres.

Pour engendrer deux mains d'un nombre de cartes en argument, on écrira ceci (avec `import random` en début de fichier) :

```
def mains(n):
    liste = [False] * (2*n-3) + [True] * 3
    random.shuffle(liste)
    return liste[:n], liste[n:]
```

Les `True` seront noyés parmi les `False` dans la fonction ainsi écrite. Il faudra garder cela à l'esprit.

Exercice 1 : Tester dix mille fois une répartition pour des mains de **dix** cartes et compter combien de fois les trois carreaux étaient ensemble. On n'hésitera pas à découper le travail en plusieurs fonctions. En déduire une estimation de la probabilité.

Exercice facultatif de mathématiques : Prouver que la formule exacte donnant la probabilité est $\frac{2\binom{17}{7}}{\binom{20}{10}}$.

Exercice 2 : Écrire une fonction prenant en argument deux entiers naturels `n` et `k` (dont on suppose le deuxième plus petit que le premier) et renvoyant le coefficient binomial associé. On utilisera de préférence la formule dite du capitaine, mais l'utilisation de factorielles (par une fonction annexe dans ce cas) reste acceptable.

Exercice 3 : Calculer la probabilité qui est à prouver dans l'exercice facultatif de mathématiques en Python. On devrait trouver environ vingt-et-un pour cent, et en pratique il s'agit de quatre dix-neuvièmes.

Exercice 4 : Recommencer toute l'expérience avec des mains de treize cartes (en devinant comment la formule avec les coefficients binomiaux s'adapte). La valeur exacte est facile à trouver sur internet, puisque le nombre de cartes par main correspond au bridge, qui a un gros historique de théorie mathématique.

Exercice 5 : Recommencer à présent avec le cas extrême, où les mains contiennent trois cartes. Confirmer par ailleurs que la même formule avec strictement moins de trois cartes donnerait un coefficient binomial nul.

Exercice 6 : Construire une liste avec les valeurs pour des mains de `n` cartes pour `n` valant des puissances consécutives de dix entre cent et un million (ou imprimer les valeurs). Observer la tendance que prend la probabilité.

Exercice facultatif de mathématiques : Faire le calcul associé à l'exercice précédent au niveau des coefficients binomiaux. Calculer la limite et justifier pourquoi on pouvait s'y attendre.

Le jeu de Yam's

Cette section propose de faire réaliser par l'ordinateur un calcul de probabilités pointu.

Le contexte est le jeu « Yam's ». Aucune connaissance du jeu ou de la théorie des jeux n'est nécessaire ici.

Le but du jeu est de réaliser diverses combinaisons à l'aide de **cinq dés**, avec la possibilité de relancer un ou plusieurs dés jusqu'à deux fois.

La meilleure combinaison, rapportant le plus de points, est le Yam's : cinq dés identiques. C'est à celle-ci uniquement que nous nous intéresserons.

Dans toute la section, on assimilera un lancer de k dés à un tirage indépendant de k entiers de 1 à 6 suivant tous la loi uniforme sur l'univers associé. On le représentera par la liste de ces entiers.

Exercice facultatif de mathématiques : Quelle est la probabilité de réaliser un Yam's au premier lancer ?

Exercice 7 : Écrire en Python une fonction qui détermine si un lancer (en argument) correspond à un Yam's.

Exercice 8 : Écrire en Python une fonction qui simule n lancers (n en argument) de 5 dés et qui calcule la fréquence des Yam's. Tester la fonction pour de grandes valeurs de n et comparer avec la probabilité calculée à l'exercice facultatif de mathématiques.

Après le premier lancer, le joueur a tout intérêt de conserver un sous-ensemble de dés qui maximise sa probabilité de réaliser finalement un Yam's (on se place dans l'hypothèse où il ne souhaite pas tenter d'autre combinaison). Inutile de justifier ce fait, qui est par ailleurs intuitif.

Nous supposons donc que le joueur conserve effectivement un sous-ensemble de dés identiques de taille maximale.

Exercice facultatif de mathématiques : Déterminer la probabilité que le plus grand sous-ensemble de dés identiques au premier lancer soit de taille quatre exactement, puis de même pour trois exactement. Calculer aussi la probabilité que les cinq dés aient une valeur différente au premier lancer et en déduire la probabilité que le plus grand sous-ensemble de dés identiques au premier lancer soit de taille deux exactement.

Exercice 9 : Écrire en Python une fonction qui détermine le nombre maximum de dés identiques d'un lancer (en argument).

C'est maintenant que les choses se compliquent : pour chacun des cinq scénarios envisagés, le nombre de dés relancés sera différent, et les cinq scénarios possibles après le deuxième lancer auront une probabilité différente suivant le scénario après le premier lancer. Ainsi, le calcul global sera obtenu par un programme.

Exercice facultatif de mathématiques : Déterminer la probabilité que trois dés exactement soient identiques au deuxième lancer, alors que deux dés identiques avaient été conservés après le premier lancer.

On remarquera que le nombre de dés identiques ne peut pas diminuer d'un lancer à l'autre. Cela fait cependant neuf (car si tous les dés sont différents, autant tous les relancer, et si tous sont identiques, on arrête) probabilités conditionnelles (tout de même) à calculer pour passer du premier lancer au deuxième lancer.

L'avantage est que les mêmes probabilités conditionnelles peuvent être utilisées pour le troisième lancer, et qu'elles ne dépendent en particulier que du nombre de dés conservés au deuxième lancer, sans tenir compte du résultat du premier lancer.

Exercice facultatif de mathématiques : Calculer les autres probabilités conditionnelles.

Exercice 10 : Écrire en Python un programme qui détermine à partir de toutes les données déterminées dans cet exercice la probabilité de réaliser un Yam's.

Remarque : L'exercice précédent ne peut se réaliser que si toutes les probabilités ont effectivement été calculées.

Exercice 11 : Écrire en Python un programme qui simule dix mille essais (chacun étant la séquence de jusqu'à trois lancers, donc) de Yam's et qui compte le nombre de réussites avec la stratégie mise en œuvre dans cette section. Comparer avec la probabilité si celle-ci a été calculée.

Exercice 12 : Une partie de Yam's se déroule en treize tours. Quelle est la probabilité de réussir un Yam's au moins deux fois au cours d'une partie, en considérant qu'on tente de le réaliser à chaque tour ? (Donner par exemple la formule à partir de la probabilité p de réaliser un Yam's au cours d'un tour et écrire un programme qui applique cette formule.)

Un peu de dénombrement...

Cette partie propose de faire réaliser par l'ordinateur un calcul de dénombrement qui s'appuie sur une suite récurrente telle que chaque terme dépend de tous les précédents.

Le contexte est le nombre de surjections. On utilisera les définitions et notations usuelles du dénombrement.

Dans tout le sujet, on considèrera un ensemble E à n éléments (n sera fixé) et un ensemble F à k éléments (k sera variable). Par principe, on assimilera E à l'intervalle d'entiers entre 0 et $n-1$ et F à l'intervalle d'entiers entre 0 et $k-1$.

Remarque : Puisqu'il ne peut pas y avoir de surjection d'un ensemble fini dans un ensemble strictement plus grand, on aura forcément $n \geq k$.

Exercice facultatif de mathématiques : Combien y a-t-il de surjections de E dans F si $k = 0$? si $k = 1$? si $k = n$?

Pour représenter une fonction de E dans F, on écrira une liste d'autant d'éléments qu'il y en a dans E, et ces éléments seront compris entre zéro et la taille de F moins un.

Exercice 13 : Écrire une fonction prenant en argument une liste représentant ainsi une fonction ainsi que la valeur k et déterminant si la liste correspond à une surjection.

Exercice 14 : Écrire une fonction permettant d'engendrer toutes les listes représentant les fonctions de E dans F (avec la taille de E et celle de F comme seuls arguments), ou éventuellement les surjections seulement. En déduire le nombre exact de surjections. Inutile de faire coexister toutes les listes, il y en a trop quand k devient grand.

Il s'avère que le nombre de surjections de E dans F est $2^n - 2$ si $k = 2$ et $3^n - 3 \times 2^n + 3$ si $k = 3$. Plus généralement, on a la relation de récurrence suivante, en notant $u_k^{(n)}$ le nombre de surjections d'un ensemble à n éléments dans un ensemble à k éléments :

$$u_k^{(n)} = k^n - \sum_{i=0}^{k-1} \binom{k}{i} u_i^{(n)}.$$

Grâce à ceci, le calcul va être grandement accéléré.

Exercice facultatif de mathématiques : Vérifier la formule précédente pour quelques petites valeurs de n et k valant 2 ou 3 à l'aide du résultat de l'exercice 14.

Exercice 15 : Écrire une fonction en Python qui calcule $u_k^{(n)}$ avec n et k en entrée.

Exercice facultatif de mathématiques (difficile) : Déterminer le nombre de surjections de E dans F dans le cas où $k = n - 2$. On pourra conjecturer la formule en se servant des fonctions écrites précédemment pour diverses valeurs de n.