ECE 1 - Année 2016-2017
Lycée français de Vienne
Mathématiques - F. Gaunard
http://frederic.gaunard.com

Informatique - T.P n°9

Simulation de lois (usuelles) finies

On a déjà vu que, sans davantage de précision, tout appel de la fonction `rand()` renvoyait un nombre réel au hasard entre 0 et 1 (on dit que la fonction simule la loi uniforme sur $[0;1]$, qui sera introduite avec plus de détails dans le chapitre sur les v.a.r à densité).

Exercice 1. (Loi uniforme sur $[\![a;b]\!]$)
Écrire, à l'aide de la fonction `rand()` une fonction `unifZ()` prenant en arguments deux entiers relatifs a et b ($a<b$) et simulant une loi uniforme sur $[\![a;b]\!]$.

Exercice 2. Que font les suites d'instructions suivantes?

```
-->x=1:20; y=1+floor(6*rand(x)); bar(x,y)
-->clf; x=1:1000; y=1+floor(6*rand(x)); histplot(0:6,y)
```

Exercice 3. (Tirages dans une urne)
On considère une urne U contenant 3 boules bleues, 4 boules blanches et 5 boules rouges. On effectue n tirages dans cette urne et on note X_i la variable aléatoire égale à 1 si la boule tirée au $i-$ème coup est bleue, 2 si elle est blanche et 3 si elle est rouge.

(1) Créer un programme qui demande à l'utilisateur un nombre de tirages n et simule l'expérience en affichant les valeurs des X_i. (☞ On pourra utiliser une loi uniforme sur $[\![1;12]\!]$.)
(2) Déterminer la loi et calculer mathématiquement $E(X_i)$.
(3) Modifier le programme pour qu'il affiche la moyenne des X_i. Qu'observe-t-on pour n grand?

Exercice 4. On désigne par n un entier naturel supérieur ou égal à 2. On dispose de n urnes, numérotées de 1 à n, contenant chacune n boules. On répète n épreuves, chacune consistant à choisir une urne au hasard et à en extraire une boule au hasard (qu'on ne remet pas). On suppose que les choix des urnes sont indépendants les uns des autres.
On note X la variable aléatoire prenant la valeur 1 si l'urne numérotée 1 contient toujours n boules au bout de ces n épreuves, et qui prend la valeur 0 sinon.

(1) Écrire une fonction `y=hasard(n)` permettant de simuler la variable X avec le paramètre n.
(2) On note N la variable aléatoire correspondant au nombre de boules manquantes à l'issue des n épreuves. Modifier la fonction précédente pour qu'elle renvoie également la valeur de N.

Exercice 5. (Sauts de puce)
Une puce se déplace sur le dos d'un animal (assimilué à un axe gradué de 0 à N) de façon aléatoire. Excepté à partir des deux extrémités où le prochain mouvement est déterminé, elle avant de une unité vers la gauche ou vers la droite, de manière équiprobable à chaque instant. On note X_n la variable aléatoire correspondant à la position de la puce après n sauts. Écrire un programme permettant de simuler X_n.

Exercice 6. (Loi de Bernoulli et loi Binomiale)

(1) Écrire une fonction `Bern()` qui prend en argument un nombre réel $p \in]0;1[$ et retourne ensuite la valeur prise, lors d'une réalisation, par une variable aléatoire $X \hookrightarrow \mathcal{B}(1,p)$.

(2) Créer à la suite une fonction `Bino()` qui prend en argument un entier naturel $n \geq 1$, un réel $p \in]0;1[$ et renvoie la valeur prise par une v.a. $S \hookrightarrow \mathcal{B}(n,p)$.

(3) Ajouter à la suite du programme des instructions pour demander à l'utilisateur n et p et effectuer ensuite 10000 réalisations de S puis afficher ensuite l'histogramme de ces réalisations.

Exercice 7. (Poolage sanguin, d'après **CCIP 1994**) On étudie une méthode de détection des porteurs d'un parasite au sein d'un ensemble donné de N individus tirés au sort. La probabilité d'être porteur du parasite dans la population est $p \in]0;1[$. Les personnes sont atteintes indépendemment les unes des autres.

On dispose d'un test permettant d'établir de façon certaine qu'un échantillon de sang contient ou non le parasite, le résultat de ce test étant dit positif dans le premier cas et négatif dans le second.

Pour chacun des N individus, on possède un prélèvement sanguin. On envisage alors deux méthodes de détection:

- Première méthode: on teste un à un les N prélèvements, effectuant ainsi N tests.

- Seconde méthode (*poolage*):
  - On fixe un entier naturel non nul ℓ. On suppose que N est un multiple de ℓ et on pose $N = n \times \ell$. On répartit alors les N prélèvements en n groupes $G_1, G_2, \ldots, G_n$, chaque groupe G_i contenant le même nombre ℓ de prélèvements. Pour chacun des groupes G_i, on extrait une quantité de sang de chacun des l prélèvements qu'il contient, puis on mélange ces extraits, obtenant ainsi un échantillon de sang H_i, caractéristique du groupe G_i.

  - On teste alors H_i:
    - si le test de H_i est négatif, aucun des individus au sein du groupe G_i n'est porteur du parasite. Le travail sur le groupe G_i est alors terminé;
    - si le test de H_i est positif, on teste un à un les prélèvements de G_i pour détecter les porteurs du parasite au sein du groupe G_i.

Soient X la variable aléatoire égale au nombre de groupes G_i pour lesquels le test de H_i a été positif et T la variable aléatoire égale au nombre total de tests effectués dans la réalisation de la méthode du poolage.

(1) Écrire une fonction `y=pool(l,p)` simulant le test sur un groupe de taille l (avec probabilité p de contamination de chaque individu). (La fonction renverra 0 ou 1 selon que le test est négatif ou non).

(2) Écrire alors une fonction `y=poolage_sanguin(N,p)` renvoyant le résultat de la simulation de X et N.

Dans tous les derniers exercices, on a pu voir comment la fonction `rand()` permettait de simuler la réalisation de certaines expériences ou variables aléatoires. S'il est capital de savoir refaire tous ces exercices et de n'utiliser que la fonction `rand()`, on peut directement appeler les lois usuelles à l'aide de la fonction `grand()`.

Plus précisément, `Y=grand(m,n,'bin',N,p)` affecte à `Y` une matrice de taille $m \times n$ dont chaque coefficient suit une loi binomiale de paramètres N, p. (Les $n \times m$ variables `Y(i,j)` modélisent des épreuves indépendantes.

☞ Reprendre les exercices avec cette remarque.