http://irem.univ-reunion.fr/spip.php?article493

Traitement algorithmique des fondements des probabilités

- Lycée et post-bac
- Probabilités et statistiques
-

Date de mise en ligne : mardi 26 avril 2011

Lorsque Blaise Pascal répondait à des lettres d'Antoine Gombaud, chevalier de Méré, il inventait le **calcul** des probabilités. La **théorie** des probabilités n'est apparue qu'au début du siècle précédent avec Kolmogorov, en utilisant la théorie de la mesure de Lebesgue, et donc la théorie des ensembles. Or les langages de programmation objet possèdent des objets de type Â« set Â», ou Â« ensemble Â», qui permettent de traiter algorithmiquement le calcul sur les ensembles (appelés évènements dans cet article) même en début d'année de Seconde, donc pour introduire les notions d'intersection et de réunion d'évènements.

Parce qu'il permet d'agir sur un objet de type Â« script Â» sans délai, ce qui permet une interactivité immédiate entre la figure et le script, et que le langage qu'il utilise (Smalltalk) est assez proche de l'Anglais, le logiciel choisi ici est DrGeoII, logiciel de géométrie dynamique, mais tout-à-fait adapté à ce genre d'activité expérimentale sur les évènements et leur probabilité. De surcroît, DrGeoII est assez rapide à prendre en main (c'est un logiciel pour enfants !) et libre et multiplateforme comme il se doit !

Pour éviter aux élèves de faire tout à la main, on va considérer un dé icosaédrique [1], pipé ou non, et des évènements comme

- Le résultat du lancer est pair ;
- Le résultat du lancer est impair ;
- Le résultat du lancer est inférieur à 13 (Â« petit Â») ;
- Le résultat du lancer est premier.

Création d'évènements par description

Chacun de ces évènements peut être défini par extension (en écrivant la liste des éventualités qu'il contient) ou, c'est la magie de Smalltalk, par description : On dit à Smalltalk en quoi consiste l'ensemble, et Smalltalk se débrouille pour calculer la liste en question.

La première chose à faire est donc de télécharger DrGeoII (mais pas besoin de l'installer, il est muni d'une machine virtuelle sur laquelle il tourne sans problème), puis de démarrer le logiciel ce qui fait apparaître une figure initialement vide. Pour afficher les évènements qu'on va calculer (puis plus tard, leur probabilité), on crée un script Smalltalk, en cliquant sur la première des icônes représentant un script (la deuxième sera utilisée tout-à-l'heure) :

[http://irem.univ-reunion.fr/local/cache-vignettes/L400xH292/evts1-92e2e.png]

On voit alors apparaître une nouvelle fenêtre dans laquelle on entre (ou plutôt modifie) un script, dont le titre (en noir ci-dessous) est le nom de l'objet tel qu'il apparaîtra dans la liste des scripts (au moment où on le placera dans la figure), et en deuxième ligne, entre traits verticaux, la liste des variables dont on aura besoin [2]. Par exemple, *omega* pour l'univers (un ensemble) et *pair* pour l'évènement « le résultat est pair » (donc encore un ensemble : celui des nombres pairs entre 1 et 20). On définit ces ensembles par description :

- *omega* est constitué des nombres entre 1 et 20, considéré comme un ensemble (donc converti)

- *pair* est constitué des éléments de *omega* qui sont pairs (even en Anglais : On choisit parmi les éléments de *omega* ceux qui sont pairs) :

[http://irem.univ-reunion.fr/local/cache-vignettes/L400xH158/evts2-ca2a3.png]

On voit qu'il a fallu ajouter une ligne pour renvoyer *omega* (le caractère Â« chapeau Â» représente l'idée de pousser *omega* vers l'affichage dans la figure). On constate aussi que chaque ligne doit se terminer par un point, et que DrGeoII est muni d'une belle coloration syntaxique (les erreurs apparaissant en rouge pour attirer l'attention sur elles).

Une fois qu'on a tapé le script, un Â« Control-S Â» permet non seulement de l'enregistrer mais aussi de laisser DrGeoII vérifier qu'il est correct. Mais on ne voit encore rien, pour cela il faut placer le script quelque part dans la figure, ce qui se fait en choisissant la deuxième des icônes représentant un script :

[http://irem.univ-reunion.fr/local/cache-vignettes/L400xH163/evts3-fcac4.png]

Ceci affiche la liste des scripts disponibles, notamment un qui s'appelle *evenements*, que l'on sélectionne, puis en cliquant quelque part en haut à gauche de la figure, on a ceci :

[http://irem.univ-reunion.fr/local/cache-vignettes/L400xH166/evts4-898e0.png]

(l'évènement « le résultat est pair » est décrit comme un ensemble (Â« a Set Â» en Anglais) contenant 2 4 6 etc.)

Et ce qui est très rapide, c'est de modifier le code précédent : Dès qu'on enregistre la nouvelle version du script, sous réserve que celle-ci ne comprenne pas d'erreur de syntaxe [3], l'affichage est automatiquement et immédiatement mis à jour, sans avoir à cliquer sur le Â« lancer le programme Â» habituel...

Donc si on remplace Â« x even Â» de l'évènement « le résultat est pair » par Â« x isPrime Â» qui veut dire Â« x est premier Â», on a directement l'évènement « le résultat est premier » représenté par un ensemble :

[http://irem.univ-reunion.fr/local/cache-vignettes/L400xH157/evts5-73896.png]

Affichage très pertinent pour rappeler que la liste des éléments d'un ensemble n'a pas à être ordonnée...

On peut quand même ranger

Pour cela il suffit de remplacer l'affichage de *omega* par celui de

```
omega asOrderedCollection
```

Opérations sur les évènements

Outre les deux créations d'évènements par extension ou par description, on peut également comme le faisait George Boole, construire de nouveaux évènements à partir des précédents.

Contraire d'un évènement

Un nombre impair, c'est un nombre qui n'est pas pair. C'est une notion qui, ici, dépend de l'univers (on enlève à l'univers, ou on *reject*, les nombres pairs) et qui est donc une opération binaire...

Problème de notation : « contenir comme élément » se note en Smalltalk Â« includes Â» ce qui n'est pas vraiment très heureux :

[http://irem.univ-reunion.fr/local/cache-vignettes/L400xH162/evts6-7d4c0.png]

Réunion d'ensembles

L'évènement « le résultat est premier ou inférieur à 13 » se note Â« union Â» en Smalltalk :

[http://irem.univ-reunion.fr/local/cache-vignettes/L400xH157/evts7-b4ba8.png]

Intersection d'ensembles

L'évènement « le résultat est à la fois premier et pair » se note Â« intersection Â» en Smalltalk :

[http://irem.univ-reunion.fr/local/cache-vignettes/L400xH160/evts8-15cbe.png]

Il n'y a pas beaucoup de nombres premiers pairs !

Des exemples d'algorithmes en algèbre de Boole

On peut par exemple aller vers une découverte expérimentale des lois de deMorgan, ou de l'associativité des intersection et réunion, voire de leur distributivité l'une par rapport à l'autre. On peut aussi tout simplement vérifier expérimentalement le statut du contraire d'un contraire...

Probabilités

Tout ceci n'empêche pas de faire de la simulation. Par exemple si on veut choisir un entier (inférieur à 20) premier au hasard (Â« at random Â» en Anglais), on demande juste à DrGeoII d'en choisir un au hasard :

[http://irem.univ-reunion.fr/local/cache-vignettes/L400xH164/evts9-4af2b.png]

Bien entendu, on peut faire cela dans une boucle, comme avec les autres outils d'algorithmique. Mais ce n'est pas le propos de cet article, qui cherche à appliquer des algorithmes, non à des nombres, mais directement aux évènements (voir onglet précédent).

Mais en plus des algorithmes de l'onglet précédent qui, à partir d'évènements, construisent d'autres évènements, on peut aussi à partir d'un évènement, calculer un nombre : sa probabilité. Ce qu'on va faire ici (et on peut même utiliser l'algorithme ci-dessous pour **définir** la notion de probabilité).

Cas équiprobable

En Smalltalk, le nombre d'éléments d'un ensemble s'appelle sa taille (Â« size Â» en Anglais). On peut donc tout simplement décider d'appeler probabilité d'un ensemble, le quotient de sa taille par celle de l'univers (beaucoup de propriétés sont aisées à démontrer à partir de cette définition) :

[http://irem.univ-reunion.fr/local/cache-vignettes/L400xH160/evts10-50e2c.png]

Il a été nécessaire de convertir le résultat en chaîne de caractères (Â« string Â» en Anglais) parce que sinon, DrGeoII l'aurait affiché comme un nombre décimal, et c'est tout de même mieux d'avoir la valeur en fraction, qui ne pose pas de problèmes d'arrondis : Au lieu de dire

« la probabilité d'avoir un nombre premier est de 0,4 »

on dit

« on a 2 chances sur 5 d'avoir un nombre premier »

Cas non équiprobable

Supposons maintenant que le dé est pipé, de telle manière que la probabilité de chaque évènement élémentaire soit proportionnelle au numéro affiché par le dé. Alors comme[\sum_{k=1}^{20}k=210] , la probabilité d'avoir le chiffre[k] est[\frac{k}{210}] . On va donc construire une liste (pas un ensemble cette fois-ci, ce sera une Â« orderedCollection Â» pour Smalltalk) contenant ces nombres, mais sans boucle, juste avec Â« collect Â» qui est l'équivalent du Â« map Â» des logiciels de calcul formel. Et pour calculer la probabilité d'un évènement, on doit initialiser une variable *p* à laquelle on va successivement additionner les valeurs des probabilités des évènements élémentaires qui le constituent, ce qui se fait directement avec Â« inject Â» :

[http://irem.univ-reunion.fr/local/cache-vignettes/L400xH161/evts11-20e00.png]

Les boucles de Smalltalk sont très concises avec ce Â« collect Â» ! On peut aussi calculer l'espérance de la loi en modifiant à peine le code ci-dessus.

En Ruby

Lorsque Yukihiro Matsumoto a inventé le langage de programmation Ruby, il s'est tellement inspiré de Smalltalk qu'on a parfois du mal à voir la différence entre les deux langages ! Aussi la version Ruby des activités précédentes a été presque immédiate à élaborer (mais sans l'engagement direct permis par DrGeoII, il faut à chaque fois lancer le programme pour le tester).

Cas équiprobable

Lorsque le dé n'est pas pipé, on obtient ceci, avec les Â« select Â» de rigueur :

```
require 'mathn'
$omega=(1..20).to_a
petit=$omega.select { |x| x<13 }
pair=$omega.select { |x| x%2==0 }
premier=$omega.select { |x| Prime.prime?(x) }
impair=$omega-pair
puts(premier|petit)
puts(premier&pair)
def proba(e)
 return Rational(e.size,$omega.size)
end
puts(proba(premier))<div class='code_download'   style='text-align: right;'> <a
href='http://irem.univ-reunion.fr/local/cache-code/dae646f28e687f35f900f3d2b74b3a12.txt'   style='font-family:
verdana, arial, sans; font-weight: bold; font-style: normal;'>Télécharger
```

On a besoin du module Â« mathn Â» pour les fractions. L'univers *omega* est converti en tableau (Â« to array Â») et non en ensemble, Ruby ne faisant pas la différence entre les deux. Les blocs sont écrits avec des accolades et non des crochets, ce qui leur fait ressembler encore plus à la notation ensembliste. L'opération de complémentaire se note simplement comme une soustraction, et les opérations booléennes par l'esperluette (Â« et Â») et le trait vertical (Â« ou Â»). Enfin le nom de la variable *omega* doit être précédé du symbole Â« dollar Â» pour rendre la variable globale [4].

Cas non équiprobable

Si le dé est pipé, avec la même loi de probabilité que dans l'onglet précédent, on a ceci :

```
require 'mathn'
$omega=(1..20).to_a
$loi=((1..20).collect {|x| x/210 }).to_a
petit=$omega.select { |x| x<13 }
pair=$omega.select { |x| x%2==0 }
premier=$omega.select { |x| Prime.prime?(x) }
impair=$omega-pair
def proba(e)
 return Rational(e.inject(0) {|p,x| p+$loi[x]})
end
puts(proba(premier))<div class='code_download'   style='text-align: right;'> <a
href='http://irem.univ-reunion.fr/local/cache-code/65773999b2aeb72e4546da143f2ef7b7.txt'   style='font-family:
verdana, arial, sans; font-weight: bold; font-style: normal;'>Télécharger
```

La syntaxe Â« inject Â» est très visiblement héritée de celle de Smalltalk !

En Python

Python (langage) est lui aussi un peu inspiré par Smalltalk (parce qu'il est objet) et en plus, possède depuis la version 3.1 un objet Â« Set Â» (ensemble) qui permet presque de refaire tout ce qui précède en Python. Presque, parce que la version 3.2 de Python ne possède pas de test de primalité. Mais ce n'est pas grave dans le cas présent, puisqu'on peut toujours définir l'ensemble par extension. Et Python permet quelque chose de mieux que les langages précédents : La notation par accolades, plus cohérente avec le cours...

Comme dans Smalltalk, le premier objet défini (Â« les entiers de 1 à 20 Â») n'est pas un ensemble, mais un itérateur (quelque chose d'assez abstrait semble-t-il...). Mais il suffit d'écrire Â« set(range(1,21)) Â» pour en faire un ensemble (il est alors converti en ensemble).

Les opérations sur les ensembles se notent comme en Ruby :

[http://irem.univ-reunion.fr/local/cache-vignettes/L400xH385/evts13-d55e9.png]

Pour gérer les fractions il suffit d'importer leurs méthodes, et la taille d'un évènement (voir l'onglet précédent) est remplacée par sa longueur :

[http://irem.univ-reunion.fr/local/cache-vignettes/L400xH111/evts14-ae3cf.png]

Avec ça il suffit d'afficher la probabilité de l'évènement Â« premier Â» pour vérifier qu'elle est toujours de 2 chances sur 5 :

[http://irem.univ-reunion.fr/local/cache-vignettes/L194xH67/evts15-8cc1b.png]

Pour en (sa)voir plus :

- La description de DrGeoII dans MathemaTICE peut être considérée comme un hors d'oeuvre pour le présent article ;
- Les probabilités avec Ruby et Python sont également explorées (mais en poussant un peu plus loin) dans ce livre en ligne ;
- La version GeoGebra est dans le cours de Seconde (en bas de page) ; dans le même domaine (celui du calcul formel), Maxima et Xcas permettent aussi de faire ce genre de calculs...
- Sur la programmation avec DrGeoII, on peut regarder ces exemples (dont certains sur les probabilités)
- Enfin une autre manière d'explorer les intersection et réunion (en 3D) avait déjà été évoquée à propos des diagrammes de Venn

---

[1] objet de l'étude des probabilités des dés en jeu de rôle, sujet qui risque d'accrocher certains des élèves...

[2] obligatoire bien que Smalltalk soit dynamiquement typé, d'où l'intérêt de donner des noms explicites aux variables

[3] dans le cas contraire, la fenêtre de débogage apparaît automatiquement, encore un outil qui rend DrGeoII très puissant pour la programmation en Smalltalk !

[4] Ça par contre, c'est l'héritage de Perl (langage)...