Biologie évolutive : modélisation et mise en pratique sous Python

Feuilles d'exercices

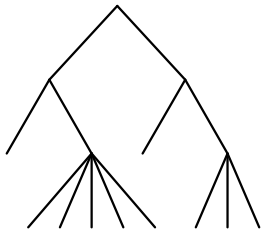
Cours recherche, Jean-Jil Duchamps
M2 Modélisation statistique, 2022–2023

1 Introduction : notion d'arbre en mathématiques

Exercice 1 : Énumérer les ensembles d'arbres :

a) non enracinés à quatre sommets.
b) enracinés à quatre sommets.
c) enracinés planaires à quatre sommets.
d) enracinés binaires à quatre feuilles.
e) non enracinés binaires à quatre feuilles étiquetées.
f) non enracinés à trois sommets étiquetés.
g) enracinés binaires à trois feuilles étiquetées.
h) binaires ordonnés, comprenant quatre points de branchement.
i) binaires ordonnés planaires, comprenant trois points de branchement.

Exercice 2 : Compléter l'arbre ci-contre en notant pour chaque nœud son étiquette selon la notation de Neveu.



Exercice 3 : Pour un arbre T, on appelle f_T son nombre de feuilles, i_T son nombre de points de branchement (ou sommets internes), et a_T son nombre d'arêtes. On note aussi $n_T = i_T + f_T$ son nombre total de sommets.

a) Justifier que $a_T = n_T - 1$.
b) Montrer par récurrence sur f_T qu'un arbre *binaire enraciné* T satisfait toujours $i_T = f_T - 1$.
c) Que devient cette formule dans le cas des arbres binaires non enracinés ?
d) Peut-on avoir une telle relation en retirant l'hypothèse de binarité ?

Dans la suite, tous les arbres considérés sont enracinés.

Exercice 4 : On note b_n le nombre d'arbres *binaires planaires* à n feuilles. Par convention, on pose $b_1 = 1$ (on accepte que l'arbre constitué d'un seul sommet est un arbre binaire à une feuille).

a) Calculer b_2, b_3 et b_4 en énumérant tous les arbres en question.
b) Montrer que l'on a, pour tout $n \geqslant 2$, $b_n = \sum_{k=1}^{n-1} b_k b_{n-k}$. Calculer b_5 et b_6.

Il existe plusieurs manières de voir que l'on peut exprimer les b_n en fonctions des nombres de Catalan C_n qui vérifient une récurrence similaire. On admettra ici que l'on a $b_n = C_{n-1} = \frac{1}{n}\binom{2n-2}{n-1}$.

c) Montrer que le nombre d'arbres planaires à *n sommets* est égal à b_n.
Il s'agit de trouver une bijection qui transforme un point de branchement à k enfants en arbre binaire à k + 1 feuilles.

Exercice 5 : On pose E_n l'ensemble des arbres *binaires à n feuilles étiquetées* de 1 à n, et on note e_n son cardinal.

a) Calculer e_1, e_2 et e_3.

b) Soit $n \geqslant 2$ et $T \in E_n$. On retire à T la feuille étiquetée n pour obtenir un arbre $T' \in E_{n-1}$. On cherche à retrouver T en observant seulement T'. Combien de possibilités a-t-on pour T ?

c) En déduire que $e_n = (2n-3)!! = (2n-3) \cdot (2n-5) \cdots 3 \cdot 1$.

d) Si e_n^p désigne le nombre d'arbres binaires planaires à n feuilles étiquetées de 1 à n, montrer que $e_n^p = 2^{n-1} e_n$.

* **Exercice 6 :** En raisonnant comme pour l'exercice précédent, montrer que le nombre d'arbres *binaires ordonnés à n feuilles étiquetées* de 1 à n est égal à $\frac{n!(n-1)!}{2^{n-1}}$.

1.1 Programmation

Exercice 7 :

a) Proposer une structure Python qui représente un arbre planaire, dont tous les sommets peuvent contenir une information (un objet Python quelconque).
Dans la suite, on l'adaptera si nécessaire afin de répondre aux questions.

b) Écrire une fonction `tree_size(tree)` qui renvoie le nombre de sommets de l'arbres.

c) Écrire une fonction `get_value(tree, node)` qui récupère la variable associée au nœud spécifié par l'argument `node`.
La variable `node` est la numérotation de Neveu–Ulam–Harris du nœud en question, sous forme de tuple : par exemple `()` pour la racine, `(2,)` pour la troisième fille de la racine, `(0,0,1)` pour la deuxième fille de la première petite-fille de la racine, etc.

d) Écrire une fonction `get_subtree(tree, node)` qui récupère le sous-arbre complet descendant du nœud donnée.

e) Écrire une fonction `graft(tree, node, trees)` qui greffe (c'est-à-dire ajoute en tant qu'enfant) les arbres de la liste `trees` sur le nœud `node` de `tree`.

f) Écrire une fonction `join(trees)` qui greffe la liste d'arbres `trees` sur une racine.

* g) Écrire une fonction de visualisation basique, qui permette de représenter la structure de l'arbre et l'information des nœuds, par exemple :

```
> print_tree(tree)
racine
+-- sommet 1
|   +-- sommet 4
|   +-- sommet 5
+-- sommet 2
+-- sommet 3
```