Kmer signature in mitochondrial genomes for metagenome taxonomic assignation

Jérmy Fontaine and Samuel Blanquart

28 juillet 2014

1 Semaine 1 : Mardi 2 Juillet

1.1 Réduction de la BDD

La base de donnée complète totalise **30Go**, l'objectif était de réduire la taille de cette BDD et n'ayant les données qu'aux feuilles, pour les autres noeuds de l'arbre les données sont sous forme de lien symbolique. On obtient au final une BDD avec une taille de **1,1GO**.

Pour cela :

- Modification du fichier chemin-liste d'accession (voir figure 1)
- Script de remplissage des noeuds internes

tax1/tax2/.../taxonFeuilleX : accession_1,...,accession_N
tax1/tax2/.../taxonFeuilleY : accession_1,...,accession_M

FIGURE 1 – Exemple de fichier "chemin : liste d'accession"

L'idée pour générer les liens symboliques est de faire remonter chaque données feuilles vers la racine de la BDD (figure 2).

```
      //
Pour chaque feuille l de la BBD
{
    /* On se deplace au niveau de la feuille */
    cd l;
    Pour chaque fichier f du dossier courant;
    {
      /* on se deplace dans la parent de cette feuille */
      courant = pwd // commande unix
      tant que courant != racine
      {
        ln -s X
        /* on remonte d'un niveau */
        cd .. ;
        courant = pwd
      }
      /* on se replace a la feuille pour traiter le nouveau fichier */
      cd l;
    }
}
```

FIGURE 2 – Algo lien symbolique

1.2 Krona

krona est un utilitaire qui permet de visualiser des hiérarchies sous forme de camembert "zoomables". Pour cela cette hiérarchies doit être écrites sous formes de fichier xml. Krona construit ensuite un fichier html local, un navigateur web permet alors de visualiser le camembert. (figure 3)

```
<node name="Alveolata">
    <genomes><val>1009</val></genomes>

    <node name="Apicomplexa">
        <genomes><val>993</val></genomes>
        ...
    </node>
</node>
```

FIGURE 3 – Exemple simplifié d'un fichier pour krona

1.3 Scripts dmp et krona

En fin de semaine, deux scripts ont été développés pour :

- Récupérer/mettre à jour les fichiers dmp pour la construction de la bdd.
- Récupérer et installer krona

2 Semaine 2 : Lundi 7 Juillet

2.1 Générer le fichier weka à partir d'un dossier

L'idée première était de générer le fichier de comptage au format weka à partir d'un dossier. Pour se faire pour un dossier D il faut effectuer le comptage pour ses sous dossier $d_1, ..., d_n$ en prenant soins de récupérer chaque taxid pour chaque d_i afin de lancer le programme de comptage *count_ kmer* avec les bons arguments. Le programme a était réalisé en Perl.
(trunk/generate_learn/generate_learn.pl)

2.1.1 Parallélisme

Au lieu de lancer le programme réalisé auparavant il est possible à partir des options d'utiliser plusieurs processeurs afin d'effectuer plusieurs comptage. Cependant même si le programme offre cette fonctionnalité lors de l'appel system de Perl pour appeler le programme C de comptage, Perl rend immédiatement la main et lance l'appel système suivant, le parallélisme se fait alors par défaut.

2.2 Générer les fréquences de kmer aux feuilles

Le but de cette partie est de générer les fréquences uniquement aux feuilles. Ainsi pour construire le fichier weka à un noeud donnée d il suffit d'aller récupérer les fréquences aux feuilles du sous arbre ayant pour pour racine d. (trunk/generate_learn/generate_count.pl)

3 Semaine 3 : Mardi 15 Juillet

3.1 Gestion d'erreur en C

Une partie de cette semaine a été consacrée au débogage du comptage écrit en C. En effet le comptage à la racine avec une fenêtre de 50, provoquait des erreurs d'allocation mémoire. Ce problème fut régler grâce à la re-allocation de la mémoire.

3.2 Taille du jeu de fréquences à la racine

Avec une taille de fenêtre glissante constante, la taille du fichier de fréquence est proportionnel à la somme des tailles de séquences (nombre de nucléotides totale). On a pu évaluer ainsi la taille du fichier de fréquence au plus haut niveau avec une fenêtre de taille 50 et le pattern ####. On obtiendrai au niveau de la racine, une taille de **123Go** pour le fichier de fréquence et un temps de **120 min**. Ainsi grâce aux graphes ci dessous on a pu évaluer le temps (fig. 4) pour générer le fichier de fréquence et la taille de celui-ci (fig. 5) sur l'ensemble des séquences (251 371 097 nucléotides).

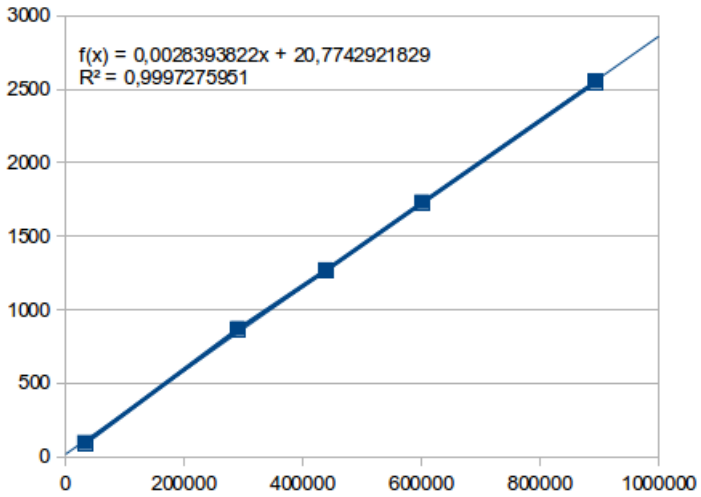

FIGURE 4 – Temps en seconde en fonction de la taille de la séquence avec une fenêtre de taille 50

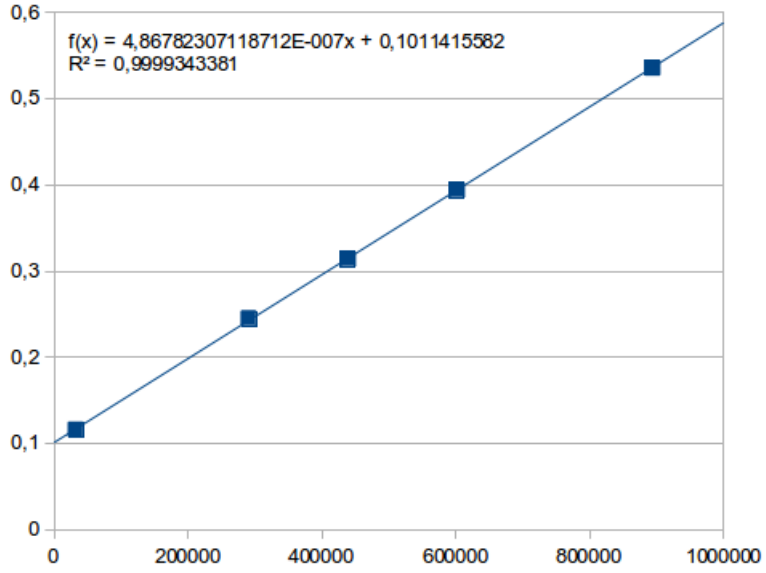

FIGURE 5 – Temps en seconde en fonction de la taille de la séquence avec une fenêtre de taille 50

3.3 Passage aux objets

Pour la suite du projet on décide de passer au c++ afin de tout gérer en tant qu'objet, on pourra ainsi plus aisément manipuler la table de fréquence en kmers. La génération de données pour weka, la validation croisée (avec les cartes...) ...

3.3.1 Les classes

Pour le moment on ne manipule que trois classes :

- classData : pour gérer les données (séquences,taille, nombre de jeux de données, responsable du comptage)
- classPattern : gestion d'un kmer
- FreqKmer : classe principale gérant principalement le tableau de fréquence.

L'objet à la possibilité de s'instancier à partir

. D'un fichier fasta

. D'un fichier contenant une liste de chemins vers des fichiers fasta.

De ce fait lorsqu'on souhaite obtenir la fréquence à un niveau, c'est au niveau des feuilles du sous arbre définit par ce niveau où l'on va effectuer le comptage.

En fin de semaine des tests de validation ont été rajoutés au programme. Le programme est fonctionnel et les tests le confirment.

4 Semaine 4 : Lundi 21 Juillet

4.1 Nouvelle manière de compter

Suite à la réunion du 18 Juillet, on a décide de ne plus compter avec une fenêtre glissante. En effet cette façon générer des redondances dans la matrice de fréquences en kmers. Pour cela l'utilisateur à le choix de fournir son propre décalage (en nucléotide) sinon le programme utilise un décalage égale à 20% de la taille de la fenêtre.

4.2 Optimisation

Dans un premier temps le programme à été revu pour prendre en compte ce décalage mais sans optimisation. L'optimisation consisterais à ne compter que les nouveau kmers lors du décalage de la fenêtre. Cette astuce était utilisée dans la version précédente. Mais étant que le décalage était égale à 1 nucléotides, une variable était suffisante pour mémoriser le kmer qui "sortait" de la nouvelle fenêtre comparé à la fenêtre précédente. On copiait ainsi la ligne de fréquence de la fenêtre précédente, on décrémenter la fréquence du kmer qui "sortait" et on calculait le nouveau kmer qui "entrait".

Avec un décalage égale à n, on a n kmer qui "sort" et n nouveaux kmer à compter (entrant). Pour se faire on se propose d'utiliser deux tampons :

. *courant* : correspondant à la ligne de comptage actuel, pour une fenêtre donnée.

. *precedent* : correspondant à la ligne de comptage pour la fenêtre précédente.

Ces tampons ont la même taille que le nombre de kmer présent dans la fenêtre, soit si $f-k+1$ si f est la taille de la fenêtre et k la taille du kmer. Ainsi lors qu'on compte une toute première fois cela permet d'initialiser le tampon courant et de mémoriser tous les kmers rencontrés. Lorsqu'on passe à la fenêtre suivante si on effectue un décalage de d nucléotides, il suffit donc :

. D'intervertir les deux tampons

. De copier les bonnes parties dans le nouveau tampon, courant

. Calculer les nouveaux kmers entrant et sauvegarder dans le tampon courant.

On obtient donc les kmers rencontrés, et on augmente dans le tableau de fréquence par rapport au tampon courant. Voir figure 6 pour illustration, avec un kmer=### et une fenêtre de taille 7.

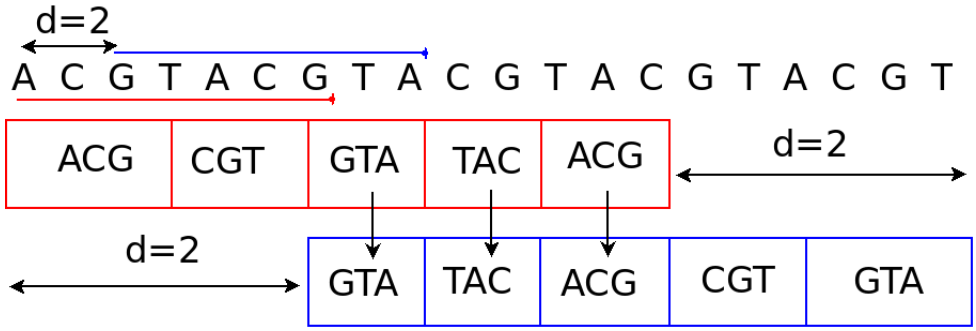

FIGURE 6 – Principe du décalage avec une fenêtre de taille 7 et un tri-mers

Même si les tests sont ok et les tableaux de fréquences corrects. Il y a des cas où valgrind détecte des erreurs au niveau des ces buffers. D'autre erreurs par ailleurs sont également détecté par valgrind. L'objectif principal avant de continuer et d'éliminer toutes ces erreurs afin de continuer sur un programme "propre".

5 Semaine 5 : Lundi 28 Juillet

6 Semaine 6 : Lundi 04 Août

7 Semaine 7 : Lundi 11 Août

8 Semaine 7 : Lundi 18 Août

9 Semaine 8 : Lundi 25 Août