UE BIF : Algorithmique des Séquences
Projet mapper MMM

Catherine Belleannée, Lolita Lecompte, Claire Lemaitre, Pierre Peterlongo

2017-2018

Ce projet est à réaliser en **binôme**, durant les séances de TP et pendant votre temps libre.

Il devra être rendu au plus tard le **Lundi 18 Décembre 2017**, par mail aux adresses :

`claire.lemaitre@inria.fr` et `pierre.peterlongo@inria.fr` avec comme sujet `[UE BIF] Projet` suivi de vos noms.

Le sujet, les fonctions fournies, ainsi que les données de test sont disponibles sur la page suivante `http://tinyurl.com/M1BIF2017/`.

N'hésitez pas à consulter régulièrement cette page qui sera mise à jour régulièrement.

1 Contexte

Un séquenceur nouvelle génération vient de vous fournir jusqu'à quelques millions de courtes séquences nucléiques de taille 100, qu'on appellera *reads*. Vous avez à votre disposition un génome de référence sous la forme d'une unique séquence.

Pour chaque read, on veut obtenir toutes les positions possibles sur le génome de référence où on peut aligner le read entièrement avec au plus un certain nombre de substitutions (erreurs de séquençage et/ou différences entre le génome de l'individu séquencé et le génome de référence).Vous implémenterez donc un mapper, qu'on appellera MMM (My Mini Mapper).

2 Caractéristiques algorithmiques

Comme la majorité des heuristiques d'alignement, MMM sera basé sur le principe "seed-and-extend" avec 2 étapes pour chaque read :

1. identification de matchs exacts de taille k entre le read et le génome de référence (graines) permettant "d'ancrer" un read sur un alignement potentiel .
2. alignement simple (substitutions uniquement) entre le read en entier et la partie du génome de référence correspondant.

Pour la première partie, vous implémenterez l'indexation du génome de référence par un FM-index (transformée de Burrows Wheeler) ainsi que les fonctions de requête sur cette structure de données [1].

Concernant la seconde partie, il faudra faire en sorte d'éviter de tester plusieurs fois le même alignement (même read à la même position sur le génome).

1. Une implémentation pour l'obtention du tableau des suffixes vous sera fournie.

3 Entrées - sorties

Entrées : deux fichiers `.fasta` contenant respectivement les reads et le génome de référence. Le format fasta est un format texte pour stocker une ou plusieurs séquences dans un même fichier. Chaque séquence est représentée sur 2 lignes successives, la première débute par le caractère "$>$" suivi par l'identifiant ou le nom de la séquence, la deuxième ligne contient la séquence elle-même.

Paramètres :

- k : la taille de mot utilisé comme graine,
- d_{max} : le nombre maximum de différences (substitutions) autorisées dans un alignement.

Sortie : MMM écrira **1 fichier** texte qui contiendra pour chaque read l'ensemble de ses alignements "retenus" dans le format suivant :

```
...
>id_read i
  >>alignment 0
  #pos=0
  #strand=+1
  #d=2
  ATCGGGATTG
  ||||||:||:
  ACCGGGTTTT
  >>alignment 1
  #pos=18620
  #strand=+1
  #d=3
  ATCGGGATTG
  |:||:||||:
  ACCGAGATTA
  >>alignment 2
  #pos=1245
  #strand=-1
  #d=0
  GCAATCCCGA
  ||||||||||
  GCAATCCCGA
>id_read i+1
  >>alignment 0
...
```

- L'identifiant des reads est celui présent dans le fichier fasta d'entrée,
- l'identifiant des alignements pour un read est simplement un entier que l'on incrémente dans l'ordre des alignements ajoutés (il est remis à 0 pour un nouveau read),
- pour chaque alignement, on indique :
  - pos : la position du début de l'alignement dans le génome de référence (la première position étant 0).
  - strand : +1 si le read s'aligne sur le brin direct, -1 si c'est le read en reverse-complement qui s'aligne à cette position (cf. section bonus). Dans la version basique, ce champ sera toujours égal à +1.
  - d : le nombre de différences dans l'alignement.

— l'alignement est représenté sur 3 lignes : la première est la séquence du read (ou de son reverse-complement - si implémenté en bonus), la deuxième indique les matchs (|) et les mismatchs (:), la troisième est la séquence issue du génome de référence.

— Pour un read, l'ordre d'affichage des positions où il mappe sur le génome n'est pas imposé.

4 Aspects pratiques

— langage de programmation : python
— Pour simplifier le codage, l'implémentation du tableau des suffixes vous sera fournie par le fichier *tools_karkkainen_sanders.py* : "*simple_kark_sort(sequence s)*" qui prend une séquence de caractères sur $\Sigma = \{ACGT\}$ et qui renvoie le tableau des suffixes pour cette séquence.
— En cas d'erreur lors de l'appel de MMM ou d'option -h, le programme proposera un message détaillé des options possibles. [anglais]
— Vous coderez entièrement toutes les autres fonctions - utilisation par exemple de dictionnaires python ou de bibliothèques extérieures type *BioPython* interdite.
— Votre code sera largement commenté. [anglais]
— Vos noms de variable et de fonction auront un sens. [anglais]
— Vous proposerez un court manuel développeur (max 3 pages) présentant les concepts et fonctions clefs et l'organisation globale de votre code. [francais ou anglais]
— Vous proposerez un manuel utilisateur (max 6 pages) pour présenter les fonctionnalités de MMM et vos conclusions concernant le choix des paramètres. [francais ou anglais]

Nous vous conseillons de **débuter** par la rédaction du manuel développeur avant de vous lancer dans le codage. N'hésitez pas à nous solliciter afin de valider votre approche avant de coder.

5 Tests et influence des paramètres

Il vous sera fourni 2 jeux de données pour tester MMM :
— un petit jeu de séquences artificielles pour faire des tests rapides de l'exactitude des résultats obtenus : répertoire `test1`
— un vrai jeu de données, avec plusieurs centaines de milliers de reads permettant d'évaluer également le temps d'exécution et l'empreinte mémoire : répertoire `test2`

Pour chacun de ces jeux de données, vous disposerez d'un ensemble de reads (fichier `reads.fasta`) et d'un génome de référence (`reference.fasta`), ainsi que les fichiers de résultats attendus pour certaines valeurs de k et d_{max}.

5.1 Qualité des résultats en fonction de k et d_{max}

L'utilisation de graines conduit à un algorithme heuristique : en fonction de k et d_{max}, des bons alignements peuvent être ratés par MMM. Votre manuel utilisateur contiendra une ou deux pages présentant plusieurs tableaux :

k	recall (%)	time (s)

qui indiqueront sur le jeu de données `test2` (`reads_1K.fasta`) l'influence de k (le *recall* indique le pourcentage d'alignements retrouvés). Chaque tableau correspondra à une valeur de d_{max} pour, d_{max} dans {0,2,4, et 10}.

Vous vous baserez sur ces tableaux pour conseiller l'utilisateur quant au choix des paramètres.

6 Bonus

— Les reads peuvent provenir des **deux brins de l'ADN** alors que la séquence du génome de référence ne représente qu'un seul brin : il faudra donc chercher les alignements pour le read, **et** pour son reverse-complément [2], on notera alors dans le fichier de sortie des alignements, le "sens" de l'alignement (champ strand).
— Pour limiter l'espace mémoire utilisé, vous pourrez implémenter les versions sous-échantillonnées du tableau des suffixes et/ou de la structure donnant le rang des caractères dans la BWT. Dans ce cas, le pas de sous-échantillonnage du tableau des suffixes (P_{SA}) et du rang (P_R) seront des paramètres d'entrée de votre programme et seront également discutés dans votre rapport utilisateur.

7 Notation

La notation de vos projets (rendus à temps) prendra en compte les aspects suivants :
— Choix algorithmiques et succès de l'implémentation
— Organisation, lisibilité et commentaires du code
— Qualité du manuel utilisateur
— Qualité du manuel développeur
— Temps d'exécution
— Bonus
Ne sous-estimez pas l'importance des rapports et la forme du code. Un projet "qui marche" n'est pas nécessairement synonyme de bonne note.

2. pour obtenir le reverse-complement d'une séquence nucléique il faut transformer chaque lettre par sa base complémentaire ($A \leftrightarrow T$, $G \leftrightarrow C$) et la lire de droite à gauche. Ex. : $rev\_comp(ATTGA) = TCAAT$