

Un peu de bio-informatique

19 janvier 2009

Un peu d'histoire

La bio-informatique est la science de l'étude informatique de certains phénomènes biologiques. Cette discipline a eu son heure de gloire avec les études autour du séquençage du génome humain. L'idée est simple : le génome humain, code par des séquences d'ADN, est une mine incroyable d'information. Un simple brin d'ADN contient des milliers de données, on a donc besoin d'algorithmes efficaces pour traiter ces données. On va en avoir un aperçu au cours de ce TP !

L'ADN

Ce n'est pas à vous que je vais l'apprendre, mais l'ADN est constitué de bases azotées qui sont au nombre de 4 : A-T-C-G. Elles sont groupées par paires complémentaires : A-T et C-G. Elles codent pour des informations diverses, du nombre d'oreille à la couleur des yeux.

1 Chargement des données

J'ai fabriqué pour vous quelques jeux de données représentant des brins d'ADN de hobbit. Ces brins se présentent sous la forme de tableau de valeurs : chaque tableau contient quelques milliers de valeurs "A", "T", "C" ou "G". Comme vous n'avez pas envie de les recopier à la main, ils sont disponibles sur le site web <http://serge.liyun.free.fr/cours/scilab>. Une fois que vous avez trouvé le fichier, téléchargez-le puis chargez ce fichier dans **SCILAB**. Je vous laisse vérifier que les tableaux **brin1**, **brin2**, **brin3** sont bien définis. À l'aide de la fonction **size** vous pouvez jeter un coup d'œil à leur taille ...

2 Brin complémentaire

On vous propose (et vous acceptez) de créer le brin complémentaire du brin d'ADN **brin1**.

1. Quelle est la taille de ce brin ? On la stockera dans la variable **taille_brin1** ;
2. Créez un tableau **compl1**
3. Faites une boucle **for** qui regarde la valeur dans la n^{ieme} case de **brin1** et remplit la n^{ieme} case de **compl1** avec son complémentaire ;
Exemple 1: si **brin1** vaut { 'A' 'T' 'T' 'G' }, **compl1** vaudra { 'T' 'A' 'A' 'C' }

4. bonus : écrire une fonction nommée `calcul_complementaire` qui effectue cette transformation

3 vérification de complémentarité

Cette fois nous avons à notre disposition deux brins d'ADN : `brin2` et `brin3`. Nous voulons vérifier que ces deux brins sont bien complémentaires l'un de l'autre, et si ce n'est pas le cas, nous voulons savoir quelle colonne pose problème.

1. créez les variables `taille2` (resp. `taille3` contenant la taille de `brin2` (resp. `brin3`);
2. vérifiez que ces deux tailles sont égales. Si ce n'est pas le cas, autant s'arrêter tout de suite ! (un petit message d'erreur peut-être ?)
3. écrire une boucle `for` qui va comparer chaque élément de `brin2` avec l'élément à la même colonne de `brin3` et dire s'ils sont complémentaires
Aide: la fonction `break` permet de sortir d'une boucle avant la fin de l'exécution

4 recherche de séquence

Vous avez déjà écrit (plusieurs fois) le code pour chercher une lettre dans un tableau de lettre. Je vous épargne la tâche de le réécrire en vous en fournissant une version ici

```
function n=ChercherLettre(l,brin)
    n=-1;
    for i=1:size(brin,2)
        if brin(i) == l
            n=i;
        end
    end
end
endfunction
```

Dans cette fonction, on recherche une lettre `l`. Mais dans notre cas on a besoin de chercher des séquences d'ADN mal formée dans notre brin, pas seulement une base ! On va faire ça par étapes :

1. modifier la fonction `ChercherLettre` pour qu'elle recherche deux lettres de suite dans une séquence. Aller je suis sympa je vous donne le canevas :

```
function n=ChercherDeuxLettres(l1, l2,brin)
    n=-1;
    for i=1:size(brin,2)
        // il va falloir modifier un peu ça
        //if brin(i) == l
        //    n=i;
        //end
    end
end
endfunction
```

2. modifier cette fonction pour qu'elle recherche trois lettres de suite ;

3. vous avez compris le truc ? Alors faites en sorte que l'on puisse chercher une séquence arbitrairement longue. Je vous donne le canevas :

```
function n=ChercherMot(mot,brin)
    n=-1;
    for i=1:size(mot,2)
        // a vous les jeunes
        //if mot(i) == 1
        //    n=i;
        //end
    end
endfunction
```