HMSN118 – Analyse de données en bioinformatique : de l'individu à la population

TP #1: Les lignes de commande et scripts

Anna-Sophie Fiston-Lavier

Le premier objectif du TP est d'apprendre à utiliser les ligne de commande et d'en apprécier l'efficacité en comparaison avec les interfaces graphiques. Le deuxième objectif est de vous accompagner vers l'implémentation de scripts (shell) pour l'analyse automatisée de données biologiques.

Révisions et Quizz « Ligne de commande »

Exercice 1 : Ligne de commande *versus* interface graphique

Préparez votre environnement de travail en créant dans votre HOME un répertoire HMSN118, un sous-répertoire TP1 et un sous-répertoire de TP1 nommé TEST (Voir HMIN113M). Déplacez-vous ensuite dans TEST.

1. Aller sur le site de l'University of California Santa Cruz (UCSC) : http://genome.ucsc.edu/.
2. Sélectionner « Downloads » puis « Genome data » .
3. Choisir la release apiMel3 du génome de l'Abeille (*Apis mellifera*).
4. Télécharger le fichier Group1.fa.gz.
5. Décompresser ce fichier.

Réflexion sur le temps estimé si l'on souhaite télécharger tous les fichiers de cette manière !

6. Reprendre à l'étape 3
7. Lire les instructions de téléchargement pour télécharger automatiquement toutes les séquences des chromosomes en quelques commandes *via* le terminal.

Pensez à vérifier que les deux fichiers soit bien identiques : soit en analysant les différences de taille (commandes du -sh ou wc -l) de contenu entre les deux fichiers (commande diff), soit en les comparant (commandes md5sum ou sha1sum).

Exercice 2 : Analyse de séquences

Les séquences nucléotidiques ou d'acides aminés sont généralement stockées au format Fasta (https://en.wikipedia.org/wiki/FASTA_format). On peut stocker une séquence ou plusieurs séquences dans un même fichier. Il existe une multitude d'autre formats de séquence (http://iubio.bio.indiana.edu/cgi-bin/readseq.cgi?morehelp). Ci-dessous des séquences nucléotidiques au format FASTA.

HMSN118 – Analyse de données en bioinformatique : de l'individu à la population

```
>R1 UbiA repeat unit 1
CATGCAAATCTTCGTCAAAACGTTGACTGGAAAAACTATCACCCTGGAGGTGGAGGCTTCCGATACCATCGAAAATGTCA
AAGCCAAGATCCAAGACAAGGAAGGAATTCCACCAGATCAGCAGAGACTTATTTTTGCTGGTACGTTGGCAAAATATCTA
ATATTTGACCTAAAATTTATTATATATTTTCAGGAAAGCAACTCGAGGATGGCCGTACCCTTTCGGATTACAATATCCAG
AAGGAATCAACCCTCCATTTGGTCCTCCGCCTAAGAGGAGG

>R2 UbiA repeat unit 2
AATGCAGATCTTCGTCAAGACTTTGACCGGAAAGACTATTACACTTGAGGTTGAAGCTTCTGACACTATCGAGAATGTGA
AGGCCAAGATCCAAGACAAGGAAGGTATCCCTCCGGATCAACAGCGTTTGATCTTTGCCGGAAAGCAACTCGAGGATGGC
CGTACTCTCTCCGATTACAACATCCAAAAGGAGTCTACTCTTCATCTGGTTCTGCGTCTCCGAGGAGG

>R3 UbiA repeat unit 3
AATGCAAATCTTCGTCAAGACTCTTACTGGAAAGACCATCACCCTCGAAGTCGAAGCCTCCGATACCATCGAGAACGTGA
AGGCCAAGATTCAGGACAAGGAAGGAATTCCACCAGATCAGCAGCGTCTCATCTTCGCCGGAAAGCAGCTCGAGGACGGC
CGCACCCTTTCTGACTACAACATCCAGAAGGAATCTACTCTTCACTTGGTTCTTCGTTTGAGAGGAGG

>R4 UbiA repeat unit 4
AATGCAGATCTTTGTCAAGACTTTGACTGGAAAGACCATCACACTTGAAGTTGAAGCTTCCGACACGATCGAGAACGTCA
AGGCCAAGATTCAAGACAAGGAGGGAATCCCGCCAGATCAGCAGCGTCTTATCTTTGCTGGTATGTTACATATAACAAAT
TTTGTTCATGAGAGACTAATTTTTTCAGGAAAGCAATTGGAAGATGGACGCACACTCTCTGATTACAATATTCAGAAAGA
GTCTACTCTCCACTTGGTGCTCCGTCTCAGAGGAGG
```

1 - Sauver ces séquences dans un fichier que vous nommerez ***ubia.fasta.***

Plusieurs manipulations sont possibles *via* les commandes pré-existantes wc, sed, tr ou grep. Lire les manuels de ces commandes et tester les exemples.

2 – Peut-on compter le nombre de séquences en ligne de commande ? Si oui, quelle est la ligne de commande ?

3 – Avec la commande *sed*, pouvez-vous remplacer le premier motif *ATT* par *NNN* puis tous les motifs en jouant avec les options ?

4 - Pouvez-vous extraire et sauver la première séquence dans un autre fichier ***ubiaR1.fasta*** en utilisant les commandes *sed* ou *tr* et *grep ?*

Exercice 2 : Mon premier script shell

Jusqu'à présent vous avez exécuté des commandes Unix, c'est-à-dire des fonctions préexistantes. Dans cet exercice, vous allez exécuter votre propre fonction, c'est-à-dire un script que vous allez écrire nommé `monScript.sh`. Votre script correspondra à une suite de fonctions telles que des commandes Unix. Pour exécuter votre script, il vous suffira de taper la commande suivante dans un terminal :

```
monScript.sh
```

HMSN118 – Analyse de données en bioinformatique : de l'individu à la population

Nous allons ensemble créer un premier script qui va afficher dans le terminal "Hello world !"
Pour cela, veuillez suivre les instructions ci-dessous :
1 – créez un fichier `monScript1.sh` et ouvrez le avec l'éditeur emacs
2 – écrivez à la première 1ere ligne du script : `#!/bin/sh`
3 – écrivez à la ligne suivante la commande : `echo "Hello world !"`
4 – enregistrez le fichier (en tapant ctrl-x ctrl-s par exemple)
5 – changez les droits de votre script pour le rendre exécutable : `chmod +x monScript1.sh`
6 – exécutez votre script en tapant dans le terminale : `./monScript1.sh`

Vous devriez voir s'afficher sur l'écran de votre terminal la phrase "Hello world !". Si oui, alors passez à l'exercice suivant. Si non, recommencez à l'étape 1.

Il est possible de fournir à un script des paramètres (options et arguments) nécessaires à l'exécution du script. Ce sont des informations/données que le script va utiliser pour s'exécuter :

```
monScript.sh <paramètres>
```

Nous allons maintenant créer un deuxième script qui affiche le texte choisi et indiqué par l'utilisateur. Ce texte va être passé en paramètre au script. A la lecture de la ligne de commande, chaque paramètre est stocké dans une variable réservée numérotée à partir de 1. Le premier paramètre sera donc stocké dans la variable 1 qui est accessible en utilisant l'expression `$1`.

Veuillez suivre les instructions ci-dessous :
1 – copiez `monScript1.sh` et nommez le nouveau script `monScript2.sh`
2 – remplacer la commande `echo "Hello world !"` par la commande : `echo $1`
3 – enregistrez le fichier
4 – exécuter ce script en tapant dans le terminale : `./monScript2.sh "coucou"`

Vous constatez que le script affiche maintenant sur le terminal "coucou" car dans cet exemple `"coucou"` est affecté à la variable réservée 1 et est appelé dans le script avec `$1` (`echo $1`).

Il est possible de donner plusieurs paramètres qui seront affectés aux variables réservées 1, 2 ,3 ,... Ces variables sont appelées dans le script à l'aide des expressions `$1, $2, $3, …`

Il existe des paramètres spéciaux qui donne accès des informations sur les paramètres eux-même :
`$0` : contient le nom du script tel qu'il a été invoqué
`$*` : l'ensemble des paramètres sous la forme d'un seul argument
`$@` : l'ensemble des arguments dans une seule structure de données (type tableau)
`$#` : le nombre de paramètres passés au script
`$?` : Le code retour de la dernière commande
`$$` : l'identifiant du processeur (PID) du shell qui exécute le script
`$!` : le PID du dernier processus lancé en arrière-plan

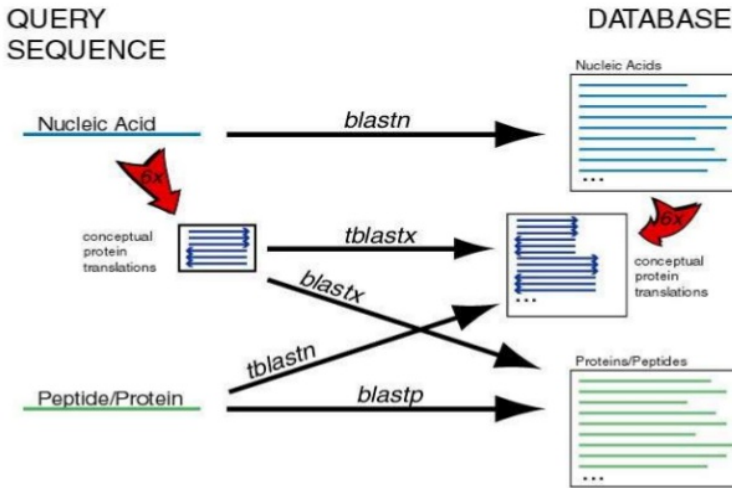
HMSN118 – Analyse de données en bioinformatique : de l'individu à la population

Exercice 3 : Scripts d'analyse de séquences avec Blast

Vous allez maintenant créer un script (`monScript3.sh`) permettant de récupérer une séquence sur un site web (Query) puis de comparer cette séquence avec une banque de données (Subject) à l'aide de l'outil de comparaison de séquences `BLAST` avec le programme `blastn` (figure 1). La première étape lors de l'utilisation de `BLAST` consiste à construire cette banque de données qui est en fait un ensemble de séquences. `BLAST` travaille sur des banques de données pré-indexées, c'est-à-dire traitées de manière à respecter une structure de données précise et donc à accélérer l'accès aux données. Le programme `formatdb`, fourni dans le package `blast2` (ou `makeblastdb` en fonction de la version du package blast utilisé), permet de créer une telle banque de données, à partir de séquences stockées au format FASTA. Sur le site du NCBI, il est néanmoins possible de télécharger un certain nombre de banque de données pré-indexées.

Pour cet exercice, vous commencerez par utiliser comme séquence, celle du CDS du gène de polyubiquitin (UbiA) de *Caenorhabditis elegans* :https://www.ncbi.nlm.nih.gov/nuccore/156481?report=fasta et comme base de données la base `nr` du NCBI.

Pour comparer une séquence requête à une de séquences, il existe plusieurs programmes :



- **blastp** blast protéique
  Pour comparer une séquence requête protéique à une banque de séquences protéiques
- **blastx** blast nucléique ***vs*** protéique
  Pour comparer une séquence requête nucléique à une banque de séquences protéiques
- **tblastn** blast protéique ***vs*** nucléique
  Pour comparer une séquence requête protéique à une banque de séquences nucléiques
- **tblastx** blast nucléique ***vs*** nucléique en passant par un alignement protéique
  Pour comparer une séquence requête nucléique à une banque de séquences nucléiques en alignant les séquences protéiques induites par les séquences nucléiques

Pour cela :
1 – copiez le script `monScript2.sh` et nommez le nouveau fichier `monScript3.sh`
2 – Ajouter la commande `wget` afin de récupérer la séquence disponible à l'adresse https://www.ncbi.nlm.nih.gov/nuccore/156481?report=fasta.
3 – Ajouter la commande Blast pour comparer la séquence récupérée avec une banque du NCBI (ajouter l'option `-remote` à la fin de la commande Blast pour utiliser une banque du NCBI)

```
blastall -p blastn -d nr -i sequence_ubi.fasta
```

On souhaite afficher aux environs de 5000 résultats (-v), avec un seuil égal à 0.15 (E-value < 0.15) mais sans afficher les alignements (-b 0). Pour cela, utilisez la commande:

```
blastall -p blastn -d nr -i sequence_ubi.fasta -e 0.15 -v 5000 -b 0
```

Vous allez créer une deuxième version où, l'utilisateur va donner les paramètres en entrée.

Copiez le script `monScript3.sh` et nommez le nouveau fichier `monScript4.sh`
- choix de la séquence/génome téléchargée (num_acc ou génome)
- choix du type de données : protéique ou nucléique
- choix de la banque de données sur laquelle s'exécute le Blast

Vous allez maintenant créer une troisième version où, l'utilisateur va donner deux fichiers Fasta avec le deuxième qui sera utilisé pour créer une banque de données avec la fonction `formatdb`.

```
formatdb -i database_sequences_ubi.fasta -p F -n datadase_ubi

blastall -p blastn -d database_ubi -i test.txt -o test.out
```