Formation Bioinformatique pour le traitement de données de séquençage

Assemblage

Annabelle Haudry

17 octobre 2023

Contents

Programme:

- Accèder au cloud
- Assembler un génome à partir de short reads
- Estimer la qualité d'un assemblage

Objectifs:

- se connecter par ssh sur une machine virtuelle
- transfert de données
- définir une pipeline d'analyse
- assemblage de novo d'un génome
- manipuler les outils et les lancer en ligne de commande

Connexion à la machine virtuelle

Connectez vous à une machine virtuelle équipée avec l'appliance Formation LBBE-NGS 2023 (1.0). A partir d'une fenetre de terminal de votre ordinateur, en spécifiant l'adresse IP de votre machine virtuelle:

```
ssh -X ubuntu@IP_adresse_de_la_VM
```

Nous allons utiliser les lectures de *S. aureus* HiSeq4 dont nous avons analysé la qualité dans le TD précédent.

```
cd data/mydatalocal/
mkdir denovo_assembly
```

Assembler un génome à partir de PE

1. Calculer la taille de k-mer optimale

kmergenie est un programme estimant la taille optimale de kmer pour réaliser un assemblage de novo à partir d'un jeu lectures (short reads). http://kmergenie.bx.psu.edu/.

Nous allons commencer par installer le programme **sur votre machine virtuelle:**

```
cd
wget http://kmergenie.bx.psu.edu/kmergenie-1.7051.tar.gz
tar -xvzf kmergenie-1.7051.tar.gz
cd kmergenie-1.7051
make
rm ../kmergenie-1.7051.tar.gz
export PATH=$PATH:/home/ubuntu/kmergenie-1.7051
```

Nous allons maintenant rechercher la taille de kmer optimale estimée pour notre jeu de données.

```
cd /home/ubuntu/data/mydatalocal/denovo_assembly/
```

Si besoin, dezipper les fichiers

```
gzip -d ../*.fastq.gz
```

Lancez kmergenie:

```
kmergenie ~/data/mydatalocal/SRR7748059_1trimmed.fastq
kmergenie ~/data/mydatalocal/SRR7748059_2trimmed.fastq
mkdir kmergenie
mv hist* kmergenie
```

La taille optimale de kmer estimée par le logiciel est-elle identique pour les reads R1 et R2 ?

2. ABySS avec une strategie optimale (kmer optimal)

ABySS est un assembleur utilisant une approche par graphes de DeBuijn qui donne de bons résultats (https://github.com/bcgsc/abyss/releases/download/2.3.1/abyss-2.3.1.tar.gz). Nous allons l'installer sur la machine virtuelle.

```
cd
sudo apt install abyss
#taper O pour finaliser l'installation
```

Nous utiliserons l'assembleur en lui spécifiant la taille de kmer optimale estimée avec kmergenie. Le signe "\\" signifie que la ligne de commande se poursuit à la ligne suivante (dans votre console, ne faites pas de retour à la ligne).

```
cd data/mydatalocal/denovo_assembly/
mkdir Abyss
cd Abyss
abyss-pe k=61 name=sa_abyss_k61 in='~/data/mydatalocal/SRR7748059_1trimmed.fastq \\
~/data/mydatalocal/SRR7748059_2trimmed.fastq'
```

Vous pouvez visualiser un histogramme représentant la fréquence de chaque kmer, en utilisant le programme R.

```
R
coverage=read.table(file="coverage.hist", header=F,  sep = "\t")
plot(log(coverage$V2),xlab="kmer", ylab="couverture")
```

Pour quitter R et retourner au terminal:

```
quit()
```

L'assemblage s'effectue en 3 étapes : assemblage des "unitigs" sans utilisation de l'information de pairage entre lectures (paired-end), alignement des paired-end reads sur l'assemblage initial ("contigs"), puis fusion des contigs joins grâce à l'information paired-end ("scaffolds"). Une fois l'assemblage terminé (seulement quelques minutes avec cet échantillon du jeu de données), vous pouvez obtenir quelques informations sur ses caractéristiques:

```
cat sa_abyss_k61-stats.tab
```

n est le nombre de contigs,

n:500 le nombre de contigs > 500bp,

L50 le plus petit nombre de contigs dont la somme des longueurs représente 50% de la longueur totale de l'assemblage,

min la taille du plus petit contig,

N75 la longueur du contig dont la taille cumulée avec les contigs plus grands représente 75% de l'assemblage,

N50 la longueur du contig dont la taille cumulée avec les contigs plus grands représente 50% de l'assemblage,

n:N50 nombre de contigs plus longs que la N50,

e-size taille attendue d'un contig en tirant aléatoirement dans l'assemblage,

max la taille du plus grand contig,

sum taille totale de l'assemblage (nombre total de ACGT dans des contigs >500pb).

Idéalement, on recherche à obtenir le moins de contigs (un par bras chromosomique) les plus longs possible. On voit l'évolution des statistiques de l'assemblage au cours des 3 étapes (unitigs, contigs et scaffolds).

Pour ne pas mélanger les résultats de plusieurs runs d'Abyss, classez vos outputs.

```
mkdir k61
mv sa* k61
mv coverage.hist k61
```

Tous vos fichiers de l'assemblage utilisant le kmer de taille 61 sont dans le dossier "k61" maintenant.

Pour information

Avec abyss, il est possible d'utiliser plusieurs librairies pour réaliser l'assemblage (*se* = files containing single-end reads,*pe*=list of paired-end libraries, *mp* =list of mate-pair libraries that will be used for scaffolding). Voir http://manpages.ubuntu.com/manpages/impish/en/man1/abyss-pe.1.html pour plus de détails.

3. IDBA

IDBA est un algorithme utilisant une approche par graphes de DeBuijn en utilisant plusieurs tailles de kmers afin de gérer l'hétèrogénéité en couverture de séquencage (problème en métagénomique, mais pas uniquement) : https://academic.oup.com/bioinformatics/article/28/11/1420/266973

Nous allons passer par l'installateur bioconda (déja installé sur la VM) pour installer le programme IDBA, et ses dépendances (en particulier le programme fq2fa qui converti des fichiers fastq au format fasta, préambule nécessaire a l'étape d'assemblage).

```
cd
conda init
source .bashrc
conda install -c bioconda idba
```

Placez vous dans le dossier qui contient les fichiers fastq (ou bien préciser le chemin) pour lancer la conversion de format.

```
cd /home/ubuntu/data/mydatalocal/denovo_assembly/
mkdir idba
```

La première étape consiste à merger les deux fichiers paired-end en un seul fichier au format fasta avec l'ensemble des données du run de séquencage.

```
fq2fa --merge --filter ../SRR7748059_1trimmed.fastq ../SRR7748059_2trimmed.fastq idba/SRR7748059.fa
```

On lance ensuite l'assemblage à proprement parlé, en précisant les options suivantes -r read.fa (fichier fasta contenant toutes les lectures) -o output_directory et le nombre de coeurs cpu à utiliser en parallele. Pour une liste complete des options, voir https://www.venea.net/man/idba_ud(1).

```
idba_ud -o idba -r idba/SRR7748059.fa --num_threads 6
ls -lh idba
mv idba/scaffold.fa idba/SRR7748059_trimmed_idba_assembly
```

L'assembleur va générer une quantité de fichiers, avec les détails pour chaque taille de kmer testée (contig-20.fa. . . .contig-100.fa). Le programme décide de facon interne quelle est la taille de k-mer optimale puis résume les résultats dans le fichier contigs.fa, procede au scaffolding des paired-end reads pour créer le fichier "final" scaffolds.fa. Notez que nous avons judicieusement renommé ce fichier afin qu'il soit plus explicit (la tracabilité des analyses est plus simple quand le nom des fichiers est informatif, et cela évite toute confusion).

Estimer la qualité d'un assemblage

Comme nous l'avons abordé lors du cours, estimer la qualité d'un assemblage est loin d'être trivial. Il existe plusieurs critères qui permettent d'estimer cette qualité, et il est rare d'obtenir un assemblage avec les meilleures valeurs pour tous les critères (ce qui est biensûr la situation idéale !). Il en revient à vous de choisir les critères qui vous semblent les plus pertinents à maximiser, selon l'utilisation des assemblages que vous souhaitez avoir. Afin de pouvoir obtenir le NG50, nous allons préciser que la taille estimée du génome (2.8 Mb).

Calculer les statistiques des assemblages

a.statistiques basiques Quast permet de calculer les statistiques basiques de caractérisation d'un assemblage (N50, longueurs des contigs, taille totale, etc.): http://quast.sourceforge.net/quast. Commencons par l'installer sur la machine virtuelle, et explorer les options proposées

```
cd
conda install -c bioconda quast
```

La commande la plus simple permettant de calculer les statistiques de base, sur les contigs de plus de 500nt (valeur par défaut qui peut être modifiée) est la suivante (précisez le chemin et le nom du fichier contenant l'assemblage):

```
quast --est-ref-size 2800000 assemblage.fa
```

Les sorties par défaut sont dans le répertoire "quast_results". Calculez les stats pour vos assemblages abyss et idba (remplacez les XX par le nom du dossier, soit la date, par défaut).

```
cat quast_results/resultsXXXXXX/report.txt
```

D'après ces statistiques, quel assemblage vous semble le meilleur (celui d'Abyss ou celui d'IDBA)?

Sortons de l'environnement conda:

```
conda deactivate
```

Notez que pour y retourner, il vous suffirait de donner la commande inverse:

```
conda activate
```

b.busco Le principe de ce pipeline est d'utiliser un set de gènes conservés dans un clade et de rechercher le nombre de ces gènes retrouvés dans l'assemblage pour en estimer la qualité. Ces sets de gènes sont présentés dans la base de données OrthoDB : https://www.orthodb.org/. (il en existe aussi pour les virus) Vous pouvez regarder le guide d'utilisation: https://busco.ezlab.org/busco_userguide.html.

Manni M, Berkeley MR, Seppey M, Simão FA, Zdobnov EM. 2021. BUSCO update: novel and streamlined workflows along with broader and deeper phylogenetic coverage for scoring of eukaryotic, prokaryotic, and viral genomes. Molecular Biology and Evolution. Available from: https://doi.org/10.1093/molbev/msab199

BUSCO est un pipeline d'analyse qui utilise plusieurs programmes, avec de nombreuses dépendances... son installation est très capricieuse !! Afin de contourner des soucis d'installation, nous allons utiliser un "container" docker, une sorte de "boîte" indépendante, avec tout le système requis pour faire tourner le logiciel d'intérêt. Les seuls fichiers auquels vous aurez accès de ce container sont les fichiers contenus dans le dossier /home/ubuntu/, donc copiez les assemblages à analyser dans ce dossier (en lieu et place de "assemblage.fa", indiquez le chemin et le nom du fichier d'assemblage à analyser).

```
cp assemblage.fa /home/ubuntu
```

Par exemple:

```
cd
cp data/mydatalocal/denovo_assembly/idba/SRR7748059_trimmed_idba_assembly /home/ubuntu
```

Téléchargez le jeu de données des orthologues orthoDB pour le clade des Bacillales d'après la liste des datasets. https://busco-data.ezlab.org/v4/data/lineages/. (toujours depuis dans /home/ubuntu)

```
wget https://busco-data.ezlab.org/v4/data/lineages/bacillales_odb10.2020-03-06.tar.gz
tar -xvzf bacillales_odb10.2020-03-06.tar.gz
```

Combien y a-t-il d'espèces dans cette référence ?

Combien de gènes sont considérés comme conservés (orthologues simple copie retrouvés chez la plupart des espèce du clade) ?

Regardez les fichiers info/species.info et links_to_ODB10.txt, et pensez a la commande linux "wc" (pour word count).

Le fichier species.info contient deux colonnes: le taxID (identifiant unique du taxon) et le nom de l'espece (voire de la souche). Le fichier links_to_ODB10.txt contient trois colonnes avec les annotations et liens a OrthoDB pour chaque gene: le code du groupe (1385 est le taxID des Bacillales), le nom du gene, le lien url avec la page correspondante dans la base de donnees OthoDB. *On trouve 409 especes et 450 genes dans ce set.*

```
cd
docker run -it -v /home/ubuntu:/busco_wd ezlabgva/busco:v5.2.2_cv1 bash
busco -i SRR7748059_trimmed_idba_assembly -l bacillales --datasets_version odb10 \\
-o outbusco_IDBA -m genome
```

Le programme appelle d'abord un prédicteur de gènes sur l'assemblage (Augustus pour les eucaryotes, Prodigal pour les procaryotes), et propose ainsi un set de séquences protéiques potentielles. Il fait ensuite une recherche de similarité de séquence (profils HMM) avec la base de données téléchargée contenant les protéines conservées chez les bacillales (HMMsearch).

Afin d'avoir une tableau résumant les résultats, regardez le fichier short_summary_xxxx qui se trouve dans le dossier de sortie (outbusco).

Afin de fermer le container docker taper simplement:

```
exit
```

Vos résultats busco se trouvent dans le dossier /home/ubuntu/outbusco. En vous aidant des différents fichiers de sortie obtenus et des commandes linux, pouvez-vous identifier:

- combien de gènes conservés sont identifiés dans vos différents assemblages ?
- combien de protéines ont été prédites par prodigal dans l'assemblage ?
- le fichier contenant les séquences nucléotidiques des gènes prédits dans l'assemblage ?
- la liste des fichiers de séquence nucléotidique des gènes "busco" trouvés en simple copie ?
- où se trouve le gène codant pour la DNA polymerase I dans l'assemblage ?

Un petit coup de pouce :

```
less outbusco_IDBA/short_summary*
wc -l /outbusco_IDBA/prodigal_output/predicted_genes/predicted.faa
less outbusco_IDBA/prodigal_output/predicted_genes/predicted.fna
ls outbusco_IDBA/run_bacillales_odb10/busco_sequences/single_copy_busco_sequences/*.fna
less outbusco_IDBA/run_bacillales_odb10/full_table.tsv
```

Pensez à rapatrier vos résultats de la VM sur votre machine à partir de votre machine locale :

```
cd /home/formation/Documents/data/
mkdir TP_denovo_assembly_J2
scp -r ubuntu@adresse_ip_machine_virtuelle:~/* /home/formation/Documents/data/TP_denovo_assembly_J2
```