DU-Bii-2020_NathPrince_Examen_FinalModules4-5

August 31, 2020

1 DU-Bii 2020 : Examen final Modules 4 et 5

1.1 Analyse simple, de données de reséquençage d'un génome bactérien.

1.1.1 Nathalie PRINCE

Notebook Jupyter: 2020-08-27 Consignes : Détailler les différentes étapes dans un rapport **HTML** ou PDF (généré via un Rmd, **jupyter lab** ou autre). - Téléchargement des données depuis les banques publiques - Contrôle qualité des données brutes (reads) - La qualité des bases vous paraît-elle satisfaisante ? Pourquoi ? - Quelle est la profondeur de séquençage (calculée par rapport à la taille du génome de référence)? - Nettoyage des reads - Quel pourcentage de reads sont filtrés et pourquoi ? - Alignement des reads contre le génome de reférence - Quel est le % de reads pairés alignés ? - Extraire dans un fichier BAM les reads chevauchant à au moins 50% le gène trmNF

Informations devant figurer dans le rapport - Présentation (par exemple à l'aide de la commande tree) de l'organisation du repretoire du projet - Justification des paramètres utilisés - Analyse succinte des résultats obtenus après chaque outil lancé (figures, tableaux ou texte)

Ne pas oubliez les informations nécessaires à la reproductibilité des analyses !!

Objectif de l'examen: Faire une première analyse de ces données, et rendre un rapport qui trace l'ensemble des étapes suivies. Ce rapport devra être mis à nôtre disposition dans un dépôt public GitHub. Les analyses devront pouvoir être rejouées sur le cluster de l'IFB.
Ces données sont issues de l'article suivant : "Complete Genome Sequences of 13 Bacillus subtilis Soil Isolates for Studying Secondary Metabolite Diversity" (doi:10.1128/MRA.01406-19).

1.2 Création d'un répertoire de travail et de ces sous-dossiers pour le projet dans mon "home directory":

PATH="/shared/projects/dubii2020/nprince/" Etape 1: Placement dans le Home Directory des projets:

[ ]:
```
cd /shared/projects/dubii2020/nprince/
```

Pour vérifier que l'on a bien changé de répertoire on utilise la commande (cmd) `pwd`:

[ ]:
```
pwd
```

Etape 2: Création du répertoire de travail pour cette analyse ainsi que des sous-dossiers de stockage des résultats générés, avec la cmd `mkdir`:

[ ]:
```
mkdir -p ExamenFin_M4_5 # "working directory" pour tous les dossiers et␣
↪fichiers de l'analyse.

mkdir -p ExamenFin_M4_5/raw_DIR # Stockage des données bruts (fastq) et␣
↪nettoyées (trimmed.fastq)
mkdir -p ExamenFin_M4_5/fastQC_DIR # Stockage des fichiers de "Contrôle Qualité"
mkdir -p ExamenFin_M4_5/GenomeRef_Index_DIR # Stockage des Index
mkdir -p ExamenFin_M4_5/Mapped_DIR # Stockage des sortie de 'Bowtie2, samtools␣
↪et bedtools'
```

Remarque:
L'option - '-p' permet de ne pas générer d'erreur si le dossier existe déjà et créer des répertoires parents si nécessaire

1.3 Présentation de l'organisation du repretoire du projet "Examen-Final_Modules4-5" à l'aide de la commande `tree` :

[ ]:
```
tree -fhpuAC /shared/projects/dubii2020/nprince/ExamenFin_M4_5
```

Remarque:

Les options: - '-f': Imprime le chemin complet pour chaque fichier; - '-p': Imprime les protections pour chaque fichier; - '-u' : Affiche le propriétaire du fichier; - '-h': Affiche la taille d'une manière plus lisible par l'homme; - 'A': Imprimer les lignes d'indentation graphique; - 'C': Active la colorisation en fonction du type de fichier.

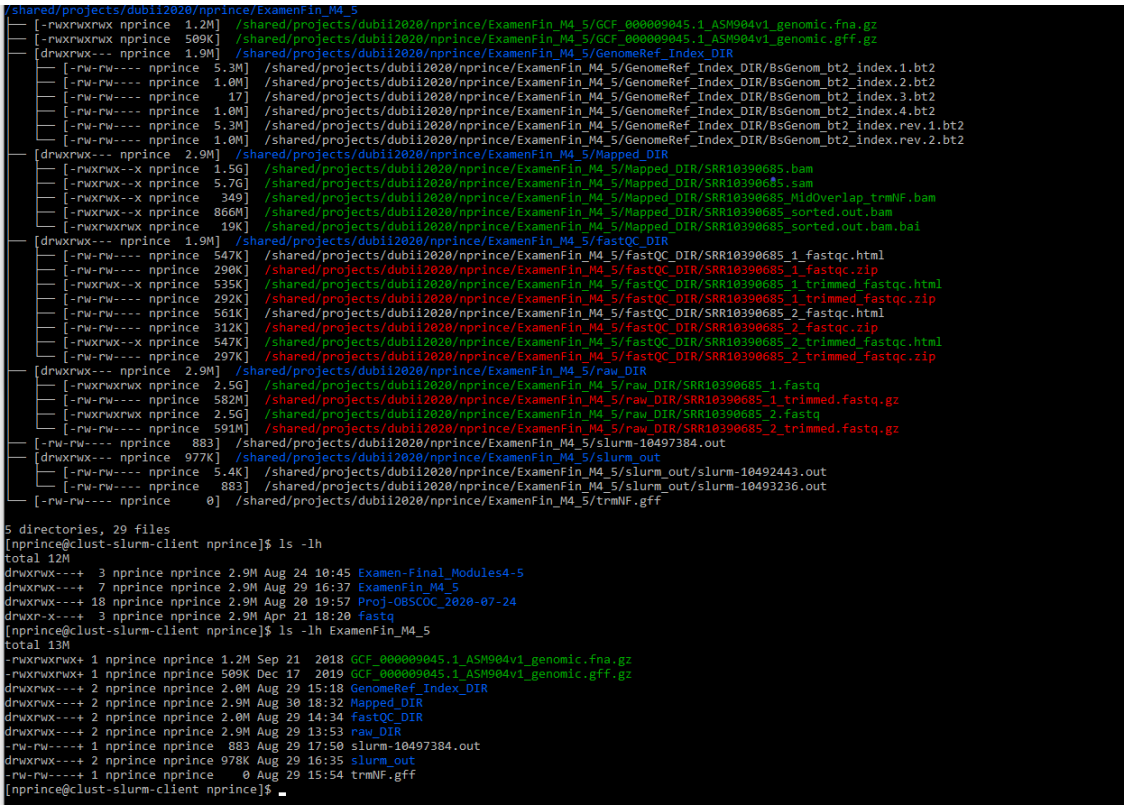

Figure 1: *Arborescence des dossiers du projet*

1.4 Chargement des programmes sra-tools, fastqc, cutadapt, bowtie2, samtools et bedtools nécessaire à la réalisation du projet :

```
[ ]: module load sra-tools
     module load fastqc/0.11.8
     module load cutadapt/2.10
     module load bowtie2/2.4.1
     module load samtools/1.9
     module load bedtools/2.29.2
```

1.5 Les commandes utilisées pour télécharger toutes les données nécessaire à la réalisation du projet :

1.5.1 Chargement des fichiers FASTQ (reads) issu de la publication (Identifiant du run : SRR10390685)

Etape 1: Aller dans le "working directory"

```
[ ]: cd ExamenFin_M4_5
```

Etape 2-a: Utilisation de l'Outil "fasterq-dump" de sra-tools pour récupérer les données "SRR…" :

[ ]:
```
srun --cpus-per-task 8 fasterq-dump -S -p SRR10390685 --outdir ./raw_DIR␣
↪--threads 8 &
```

Remarque:

Les options: - '–threads 8': 8 threads utilisés (dflt = 6); - '-S | –split-files': écrit des lectures dans différents fichiers; - '-p | –progres': Affiche la progression (impossible si stdout est utilisé); - '&': Permet de reprendre la main pour pouvoir visualiser la progression; - '-O |–outdir': chemin du fichier de sortie (remplace l'utilisation du répertoire courant)

Etape 2-b: Avec la commande (cmd) `ls -lh` on liste un répertoire (répertoire courant par defaut), pour vérifier si les fichiers ont bien été Téléchargés.
Les options: - '-l': Affiche un format de liste détaillé. - '-h': Affiche la taille des fichiers à un format plus facilement lisible par l'homme

[ ]:
```
ls -lh raw_DIR/
```

Affichage sur la sortie standard:

-rw-rw—-+ 1 nprince nprince 2.6G Aug 29 12:04 SRR10390685_1.fastq

-rw-rw—-+ 1 nprince nprince 2.6G Aug 29 12:04 SRR10390685_2.fastq

Remarque: Seuls 2 fichiers ont été téléchargés, il s'agit de 2 reads pairées mais sans droits d'exécution. Pour changer les permissions d'accès j'utilise la cmd `chmod ugo+rwx` qui permet d'ajouter les droits d'exécution à l'utilisateur (u), au groupe (g) et aux autres (o). "+" stipule l'ajout des arguments suivant et "x" correspond aux droits d'exécution, "w" aux droits d'écriture et "r" aux droits de lecture.

[ ]:
```
chmod ugo+rwx raw_DIR/*fastq
```

Affichage sur la sortie standard:

-rwxrwxrwx+ 1 nprince nprince 2.6G Aug 29 12:04 SRR10390685_1.fastq

-rwxrwxrwx+ 1 nprince nprince 2.6G Aug 29 12:04 SRR10390685_2.fastq

1.5.2 Chargement du Génome bactérien de référence au format FASTA :

Pour télécharger les données depuis les banques publiques j'utilise la commande `Wget` qui est un programme en ligne de commande qui permet le téléchargement de fichiers depuis le web (ici depuis le site de NCBI) :

[ ]:
```
wget https://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/009/045/GCF_000009045.
↪1_ASM904v1/GCF_000009045.1_ASM904v1_genomic.fna.gz
```

[ ]:
```
ls -lh  # Pour controler si mon fichier fasta à bien été chargé  dans le␣
↪répertoire de travail.
```

1.5.3 Chargement de l'annotation du génome bactérien au format GFF (puis dézipper les 2 fichiers .gz):

```
[ ]: wget https://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/009/045/GCF_000009045.
     ↪1_ASM904v1/GCF_000009045.1_ASM904v1_genomic.gff.gz
```

```
[ ]: ls -lh  # Pour controler si mon fichier gff à bien été chargé dans le␣
     ↪répertoire de travail.
```

Remarque-1: Ici encore les fichiers FASTA et GFF ont été téléchargé sans droits d'exécution. On utilise `chmod ugo+rwx`pour changer les permissions de ces fichiers :

Les 2 fichiers ayant la même racine ("GCF_000009045.1_ASM904v1_genomic."), je n'ajoute qu'un seul argument à la cmd `chmod`suivit d'une étoile `*` (à utiliser avec modération) pour préciser qu'il s'agit de tous les fichiers avec cette même racine:

```
[ ]: chmod ugo+rwx GCF_000009045.1_ASM904v1_genomic.*
```

Remarque-2: Le 'Script wrapper' de Bowtie2 permet d'avoir la capacité de gérer les entrées compressées dans Bowtie2. Dans ce cas il est inutile de décompresser les fichier FASTA et GFF pour gagner de l'espace disque. FastQC et Cutadapt supportent également les fichiers compressés.

1.6 Contrôle qualité des données brutes (reads)

Le module fastqc génére 2 fichiers de sortie (1 .html et 1 .zip) par fichier d'entrée (raw data: fastq ou fastq.gz) et permet de contrôler la qualité des reads.

Etape 1: Dans le répertoire de travail, lancer `fastqc`:

```
[ ]: # Créer les variables d'environnement pour les entrées et les sorties de fastQC:
     output_dir="fastQC_DIR"

     input_fastq=(/shared/projects/dubii2020/nprince/ExamenFin_M4_5/raw_DIR/*.fastq)


     # Lancer FastQC/
     srun fastqc -t 2 ${input_fastq[@]} -o ${output_dir} &
```

=> Une fois les 2 fichiers du contrôle qualité générés, le fichier .html permet de visualiser plusieurs paramètres représentatifs de la qualité des données bruts. Ainsi, il est nécessaire de rapatrier sur son PC le fichier HTML soit via 'Filezilla', soit avec la cmd `rsync -aPh <name>@core.cluster.france-bioinformatique.fr:/shared/projects/dubii2020/nprince/ExamenFin_M4_5/fastQC_DIR/*fastqc.html /mnt/c/Users/Nathalie/Documents/DUBii2020_NP/GitMethodesOutils_Module_4-5/Exam_M4-5/`

Les options dans les lignes de cmd ci-dessus (`mkdir`et `fastqc`):

mkdir: - '-p, –parents' : Fait en sorte qu'il n'y a pas d'erreur si le fichiier existe déjà et créer des répertoires parents si nécessaire.

fastqc: - '-o' : Sert à spécifier le fichier de sortie; - '-t 2' : Spécifie le nombre de threads, ici "2" - "[@]": Récupère chacuns des éléments contenus dans l'input_fastq - '&': permet de reprendre la main et de pouvoir suivre la progression sur la sortie standard.

QUESTION : La qualité des bases vous paraît-elle satisfaisante ? Pourquoi ?

REPONSE : Après visualisation du fichier HTML du fastQC, la qualité des bases me semble satisfaisante au regard : - du premier item du rapport FastQC, "Basic Statistics", qui stipule qu'il n'a aucune séquence marquée comme étant de mauvaise qualité (Sequences flagged as poor quality = 0); - du second item du rapport FastQC, "Per base sequence quality", dont le graphe montre que le score de qualité des bases se situe majoritairement au-dessus de 30 hormis pour l'extrémité 3' des séquences.

QUESTION : Quelle est la profondeur de séquençage (calculée par rapport à la taille du génome de référence) ?

REPONSE : La "Profondeur de séquençage" correspond au rapport entre la longueur de l'ensemble des séquences lues mises bout à bout et la longueur du génome cible. Par exemple, dans cette analyse de reséquençage d'un génome bactérien on dispose : - d'un total de 7066055 séquences d'une longueur comprise entre [35 et 151]pbs. Toutes ces séquences misent bout à bout donneront une longueur totale comprise entre [247 et 1067]Mb (ou [247311925 et 1066974305]pbs) - Et la taille du génome de Bacillus Subtilis est de: 4 Mb (4215606 pbs) (information dans le fichier .gff)

=> On a donc minimum 247 millions de bases (Mb) pour un génome de 4 Mb, soit: une profondeur supérieur à ~**60** équivalents du génome de Bacillus subtilis, ce que l'on note **60X**.

1.7 Nettoyage des reads :

Etape 1: Pour ne garder que les séquences d'intérêt et éliminer les adaptateurs de séquençage, j'ai utilisé le module "cutadapt". Utilisé comme suit, dans une **Boucle `for`**:

[ ]:
```
# Si ce n'est pas déjà fait (re-)charger cutadapt:
#module load cutadapt/2.10

input_fastqToCut=(/shared/projects/dubii2020/nprince/ExamenFin_M4_5/raw_DIR/*_1.
 ↪fastq)

For input_file in ${input_fastqToCut[@]}
    do
    sample_name=$(basename "${input_fastqToCut}" _1.fastq)


    srun cutadapt --nextseq-trim=10 -j 2 -a GATCGGAAGAGCACACGTCTGAACTCCAGTC -A␣
 ↪GATCGGAAGAGCACACGTCTGAACTCCAGTC -m 50:50\
        -o raw_DIR/${sample_name}_1_trimmed.fastq.gz -p raw_DIR/
 ↪${sample_name}_2_trimmed.fastq.gz raw_DIR/${sample_name}_1.fastq\
        raw_DIR/${sample_name}_2.fastq &
```

```
done
```

Remarque-1:

- Pour que la ligne de commande puisse fonctionner sur un plus grand nombre de fichiers d'entrée de manière reproductible, je dois faire en sorte de pouvoir répéter les mêmes instructions pour chaque fichier. Pour cela j'utilise une boucle `for`.
- Cutadapt peut prendre des fichiers décompressés (ou compressés) en entrée et génére des fichiers compressés en sortie; Il suffit de le stipuler dans le nom des fichiers de sortie dans la ligne de commande.
- L'analyse avec l'option `-j 1` a durée 18 min, pour diminuer le temps d'execution => On peut augmenter le nombre de coeurs à `-j 2`.

Remarque-2:

- D'après l'article dont sont issus les données, les reads par paires ont été générées sur un séquenceur Illumina NextSeq. Comme le NextSeq fait partie des instruments Illumina qui utilisent une chimie à deux couleurs pour coder les quatre bases et que l'algorithme de cutadapt par défault ne peut pas gérer cette situation, on doit utiliser l'option '–nextseq-trim'.
- La séquence de l'adaptateur "NEBNext Adaptor for Illumina" est une séquence universelle.

Les options de `cutadapt`: - '-j CORES': Nombre de cœurs de processeur à utiliser. Utilisez 0 pour la détection automatique. Par défaut: 1 - '-a': Séquence d'un adaptateur lié à l'extrémité 3' de la read 1. L'adaptateur et les bases suivantes sont coupés. - '-A': Séquence d'un adaptateur lié à l'extrémité 3' de la read 2. - '-m 50:50': Spécifie la taiile minimale des reads1 et reads2 respectivement. - '-o et -p': l'output de la read1 et l'output de la read2 respectivement.

La sortie standard du module cutadapt est la suivante: ***Affichage sur la sortie standard:***

=== Summary ===

Total read pairs processed: 7,066,055

- Read 1 with adapter: 262,113 (3.7%)
- Read 2 with adapter: 253,188 (3.6%)

Pairs written (passing filters): 7,048,148 (99.7%)

Total basepairs processed: 2,119,142,216 bp

- Read 1: 1,056,334,498 bp
- Read 2: 1,062,807,718 bp

Quality-trimmed: 6,857,492 bp (0.3%)

- Read 1: 2,510,757 bp
- Read 2: 4,346,735 bp

Total written (filtered): 2,108,657,437 bp (99.5%)

- Read 1: 1,051,734,134 bp

- Read 2: 1,056,923,303 bp

QUESTION : Quel pourcentage de reads sont filtrés et pourquoi ?

REPONSE :
- Le nombre de reads en entrée (in_reads) est de : 7066055 - Le nombre de reads pairées passant les filtres : 7048142 soit **99.7%**

_=> La qualité des bases étant très satisfaisantes avant l'étape de nettoyage, cela explique le faible pourcentage de reads finalement nettoyées (Read-1: 3.7% et Read-2: 3.6%) ainsi que le pourcentage élevé de 99,7% de read pairés passant les filtres.

Rmq : Un nombre résiduel de reads avec adaptateurs à nettoyer est un phénomène attendu même lorsque les étapes de workflow générant les données brut (en post-bioinformatique) se déroulent toutes convenablement._

1.8 Relance d'un contrôle qualité sur les données "Nettoyées" (trimmed reads)

[ ]:
```
# Si ce n'est pas déjà fait (re-)charger fastQC:
#module load fastqc/0.11.8

# Créer les variables d'environnement pour les entrées et les sorties de fastQC:
output_dir="fastQC_DIR"

inpuTrim_fastq=(/shared/projects/dubii2020/nprince/ExamenFin_M4_5/raw_DIR/
 ↪*_trimmed.fastq.gz)


# Lancer FastSC/
srun fastqc -t 2 ${inpuTrim_fastq[@]} -o ${output_dir} &
```

=> Dans ce 2nd rapport FastQC: - l'item "Overrepresented sequences" n'a pas généré de graphe cette fois-ci. Les reads ne contiennent donc plus aucune séquence surreprésentée. - Cependant l'item "Adapter Content" semble encore contenir quelques adaptateurs. Mais la longueur des séquences allant jusqu'à 150 pb cela ne devrait pas trop impacter sur les prochaines étapes d'alignement (ce qui n'est pas du tout le cas lorsqu'il s'agit de petites séquences comme les microARN de 20-30 pbs).

1.9 Alignement des reads contre le génome de reférence

Étape 1: Charger l'outil Bowtie qui permet d'aligner des reads contre un génome de référence (si ce n'est pas déjà fait)

[ ]:
```
module load bowtie2/2.4.1
```

Étape 2: Afficher l'aide de l'outil Bowtie2 qui permet d'aligner des reads contre un génome de référence

[ ]:
```
bowtie2 --help
```

L'aide fournit indique *l'usage* ainsi que les nombreuses *Options* à utiliser pour faire fonctionner Bowtie2. Les première lignes de l'aide de Bowtie2 sont:

Affichage sur la sortie standard:

Bowtie 2 version 2.4.1 by Ben Langmead (langmea@cs.jhu.edu, www.cs.jhu.edu/~langmea)

Usage:

bowtie2 [options]* -x 'bt2-idx' {-1 `m1` -2 `m2` | -U `r` | –interleaved | -b } -S `sam`

`bt2-idx` Index filename prefix (minus trailing .X.bt2)

NOTE: Bowtie 1 and Bowtie 2 indexes are not compatible

`m1` Files with #1 mates, paired with files in `m2`

Could be gzip'ed (extension: .gz) or bzip2'ed (extension: .bz2)

`m2` Files with #2 mates, paired with files in `m1` Could be gzip'ed (extension: .gz) or bzip2'ed (extension: .bz2)

`r` Files with unpaired reads

Could be gzip'ed (extension: .gz) or bzip2'ed (extension: .bz2)

`i` Files with interleaved paired-end FASTQ/FASTA reads Could be gzip'ed (extension: .gz) or bzip2'ed (extension: .bz2)

`bam` Files are unaligned BAM sorted by read name

`sam` File for SAM output (default: stdout)

`m1`, `m2`, `r` can be comma-separated lists (no whitespace) and can be

specified many times. E.g. '-U file1.fq,file2.fq -U file3.fq'

Remarque: '-x `bt2-idx` indique qu'il faut fournir des fichiers d'index à la ligne de commande. On obtient ces fichiers grace à la cmd "bowtie2-build" information issu du Manuel d'aide en ligne de Bowtie2:

Étape 3: Afficher l'aide de l'outil Bowtie2-build puis créer les indexs

```
[ ]: bowtie2-build --help
```

Affichage sur la sortie standard:

Bowtie 2 version 2.4.1 by Ben Langmead (langmea@cs.jhu.edu, www.cs.jhu.edu/~langmea)

Usage: bowtie2-build [options]* `reference_in` `bt2_index_base`

reference_in comma-separated list of files with ref sequences bt2_index_base write bt2 data to files with this dir/basename

Construction des Index de Bacillus subtilis à partir du fichier .fna du Génome de référence: GenomeRef_DIR/GCF_000009045.1_ASM904v1_genomic.fna

Etape1: "bowtie2-build" permet de générer les indexes du genome de référence de Bacillus subtilis (Bs) depuis le répertoire où se trouve la séquence de reference .fna (ici, le répertoire de travail):

```
[ ]: sbatch --wrap "bowtie2-build -f GCF_000009045.1_ASM904v1_genomic.fna.gz␣
     ↪GenomeRef_Index_DIR/BsGenom_bt2_index"
```

Remarque: Les indexes ne se génèrent qu'une seule fois pour l'ensemble de l'analyse. - L'option "–wrap": permet d'envelopper la ligne de commande qui la suit. Ainsi sbatch considèrera ce qui suit "–wrap" comme un script apparentière. - L'option "-f": indique à Bowtie le type d'extension .fa, .mfa, .fna ou similar

```
[ ]: # Vérifier que les indexes ont bien été générés en les listant sur la sortie␣
     ↪standard:
     ls -lh GenomeRef_Index_DIR/
```

Remarque: Si tout s'est bien déroulé, six fichiers d'indexes ont été générés dans le répertoire "GenomeRef_Index_DIR"

Alignement des reads préalablement nettoyées : Étape 4: Alignement des reads avec le module bowtie2

```
[ ]: # S'il n'existe pas créer le répertoire pour les fichiers de sortie de Bowtie2␣
     ↪(BAM ou SAM):
     # mkdir -p Mapped_DIR
```

```
[ ]: ## Definition d'une variable d'environnement pour les indexs du genome de␣
     ↪référence de Bs:
     BWT_INDEX_DIR="/shared/projects/dubii2020/nprince/ExamenFin_M4_5/
     ↪GenomeRef_Index_DIR/BsGenom_bt2_index"

     ## Definition d'une variable d'environnement pour les fichiers fastq nettoyés:
     FastqTrimmed="/shared/projects/dubii2020/nprince/ExamenFin_M4_5/raw_DIR/
     ↪*_1_trimmed.fastq.gz"


     # Faire une boucle 'for' pour que cette étape de l'analyse soit reproductible:
     for fastqCut_file in ${FastqTrimmed[@]}
         do
             BASE="$(basename $fastqCut_file _1_trimmed.fastq.gz)"


     # Alignement des séquences nettoyées sur le génome de Bs:

             sbatch -J ${BASE} --wrap "bowtie2 -D 15 -R 2 -N 0 -L 22 -i S,1,1.15 -x␣
     ↪${BWT_INDEX_DIR} -1 raw_DIR/${BASE}_1_trimmed.fastq.gz\
             -2 raw_DIR/${BASE}_2_trimmed.fastq.gz -S Mapped_DIR/${BASE}.sam"
```

```
done
```

Remarque:

- Ici encore, j'utilise une boucle `for` je définis plusieurs variables pour la reproductibiité des analyses.
- Les différentes options utilisées dans la ligne de commande Bowtie2 sont : `-x` se réfère aux indexes; `-1` fait reference à la read 1; `-2` se réfère à la read 2; et `-S` correspond à la sortie au formet SAM.
  `-D 15 -R 2 -N 0 -L 22 -i S,1,1.15`: Correspond à l'option "–sensitive" en mode end-to-end

En sortie du module bowtie2 on obtient un fichier SAM. En affichant les 11 premières lignes du fichier avec la cmd: 'head -20', on obtient sur la sortie std: ***Affichage sur la sortie standard:***

Bowtie 2 version 2.4.1 by Ben Langmead (langmea@cs.jhu.edu, www.cs.jhu.edu/~langmea)

@HD VN:1.0 SO:unsorted

@SQ SN:NC_000964.3 LN:4215606

@PG ID:bowtie2 PN:bowtie2 VN:2.4.1 CL:"/shared/software/miniconda/envs/bowtie2-2.4.1/bin/bowtie2-align-s –wrapper basic-0 -D 15 -R 2 -N 0 -L 20 -i S,1,0.75 -x /shared/projects/dubii2020/nprince/Examen-Final_Modules4-5/ExamenFin_M4_5/GenomeRef_Index_DIR/BsGenom_bt2_index -S Mapped_DIR/SRR10390685.sam -1 raw_DIR/SRR10390685_1_trimmed.fastq -2 raw_DIR/SRR10390685_2_trimmed.fastq"

SRR10390685.1 77 * 0 0 * * 0 0 GCTTTNTTGTTTGT-CATACGTTTGCCACTTATAAGTGCAAGGGATGTTATGTTTCT-TAAAAATTGAAACCAATAGGGGGAGTAAAAATGGAAAATTATACAACGTTTTTAGGTATTTACAAACCT
AAAAA#EEEEEEEEEEEEEEEEEEEAEEEEEEEEEEEEEEEEEEEEEEEEEEAEEEEEEEEEEEE//EE
YT:Z:UP

SRR10390685.1 141 * 0 0 * * 0 0 NNATTTGTATGATCACTATTGGTGAGTACAGAGTGAC-CAACTGGCGTATCGTCTAATAACTTGATGATGGTCCTTTTTTTACCCGCTCCAACAAGCATTGTATTCC
##AAAEAEEEEEE/EEEEEEEEEEEEEEEEAEEEEEEEE/E6E/EEAEEEEEEEEEEEEEEEEEAEEAEEA/
YT:Z:UP

SRR10390685.2 97 NC_000964.3 2046474 8 117M = 2046455 -136 CTGGCNC-CTTATGATGGGAGCCGTCATGTTATTGGTGTTTTCCTTTATATTTGAGGCG-GTCCCTTCAGCCGAGTGGACATATCAGGCGGTGTGGTCATTGCTGTTTAATGGCTTGCT
AAAAA#EEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE
AS:i:-1 XN:i:0 XM:i:1 XO:i:0 XG:i:0 NM:i:1 MD:Z:5A111 YS:i:-89 YT:Z:DP

Remarque: Pour la sortie SAM de lectures pairées, Bowtie2 imprime d'abord

- l'en-tête qui donne des informations sur le génome ou le sur le mapping. Les lignes d'en-tête commencent toutes par un @ , suivi de deux lettres.
- Puis le corps qui est un format tabulé. Signification des 12 colonnes:

1. (SRR10390685.2) le nom d'une lecture
2. (97) informations binaires sur la lecture. Un unique chiffre enregistre les informations suivantes : fragment mappé ou non, fragment d'un paired-end (ou d'un single-end), premier de la paire, etc. Pour savoir ce que signifie un nombre, il suffit d'aller sur le site dédié.
3. (NC_000964.3) la séquence sur lequel est mappée la lecture; on utilise * si la lecture n'est pas mappée
4. (2046474) position la plus 5' de la lecture
5. (8) qualité du mapping, soit $-10 \log 1\ 0$ (p), où p est la probabilité, estimée par l'outil de mapping, que la lecture soit mappée à la mauvaise place
6. (117M) format CIGAR de la lecture (cf infra)
7. (=) séquence sur lequel est mappé l'autre fragment, en cas de paired-end ; on utilise = si le chromosome est le même, et * si la lecture est single-end
8. (2046455) en cas de paired-end, osition la plus 5' de l'autre fragment ; on utilise 0 si l'on est en single-end, si l'autre fragment ne mappe pas, etc.
9. (-136) en cas de paired-end, taille de la lecture, soit la différence entre la position la plus 5' du fragment 5' et la position la plus 3' du fragment 3'
10. (CTGGCNCCTTATGATGGGAGCCG…) séquence du fragment
11. (AAAAA#EEEEEEEEEEEEEEEEE…) qualité du fragment (similaire au FASTQ)
12. (AS:i:-11 XN:i:0 XM:i:3) autres informations

A noter:

Lorsque Bowtie2 imprime un alignement SAM pour SAM des lectures pairées, il imprime deux enregistrements (c'est-à-dire deux lignes de sortie), une pour chaque élément d'une paire. La première ligne décrit l'alignement de l'élément 1 et la seconde ligne décrit l'alignement de l'élément 2.

$**QUESTION** : **$: Quel est le % de reads pairés alignés ?

$**REPONSE** : **$:

En sortie du module bowtie2 on obtient également un fichier ".out". dans lequel on trouve des informations sur l'alignement: 7048148 reads; of these:

- 7048148 (100.00%) were paired; of these:
  - 818768 (11.62%) aligned concordantly 0 times
  - 6129286 (86.96%) aligned concordantly exactly 1 time
  - 100094 (1.42%) aligned concordantly >1 times

———-

- 818768 pairs aligned concordantly 0 times; of these:
  - 193449 (23.63%) aligned discordantly 1 time

———-

625319 pairs aligned 0 times concordantly or discordantly; of these:

- 1250638 mates make up the pairs; of these:

- 1074934 (85.95%) aligned 0 times
- 164451 (13.15%) aligned exactly 1 time
- 11253 (0.90%) aligned >1 times

=> 92.37% overall alignment rate

Le résultat ci-dessus nous indique que: - 100% des lectures sont pairées, dont: - 11.62% sont concordante entre-elles mais ne se sont pas alignées. - 86.96% sont concordante entre-elles et se sont alignées 1 unique fois. - 1.42% sont concordante entre-elles et se sont alignées plus d'1 fois.

1.9.1 => Ainsi, on obtient 88.38% de reads pairés et alignés (1 fois ou plus).

Remarque: Le fichier SAM a été généré sans droits d'exécution. On utilise la cmd `chmod ugo+x` pour changer les permissions d'exécution de ce fichier.

1.10 Extraire dans un fichier BAM les reads chevauchant à au moins 50% le gène trmNF

Le fichier SAM contient, entre autres, les séquences de régions génomique de *Bacillus subtillis* qui ont été sequencées. Le fichiers GFF est un fichier d'annotation contenant divers informations et notamment les coordonnées de tous les gènes connus du génome bactérien.

L'étape suivante consiste donc à **croiser** les 2 fichiers pour déterminer quels sont les reads chevauchant au moins à 50% le gène trmNF après avoir extrait les coordonnées de ce gène.

**=> Pour manipulers les données alignées contenues dans le fichier SAM on doit au préalable convertir le fichier SAM en BAM qui est simplement un format biaire de SAM (plus facilement lisible pour une machine) et compressé. Les BAM sont donc plus facile à utiliser par les visualisateurs et les outils de traitement de lectures.

Ainsi pour manipuler et croiser ces données hétérogènes on utilise les modules "sametools" et "bedtools"**:

Étape 1: Charger les outils "samtools" et "bedtools" (si ce n'est pas déjà fait):

[ ]:
```
module load samtools/1.9
module load bedtools/2.29.2
```

Etape 2: Afin d'avoir une idée des informations contenues dans le fichier GFF j'utilise la cmd `head` sur le fichier d'annotation "GCF_000009045.1_ASM904v1_genomic.gff" (Cidessous j'affiche les 11 premières lignes):

[ ]:
```
head -n 11
```

##gff-version 3

#!gff-spec-version 1.21

#!processor NCBI annotwriter

#!genome-build ASM904v1

#!genome-build-accession NCBI_Assembly:GCF_000009045.1

##sequence-region NC_000964.3 1 4215606

##species https://www.ncbi.nlm.nih.gov/Taxonomy/Browser/wwwtax.cgi?id=224308

NC_000964.3 RefSeq region 1 4215606 . + . ID=NC_000964.3:1..4215606;Dbxref=taxon:224308;Is_circular=true;
DNA;strain=168;sub-species=subtilis;type-material=type strain of Bacillus subtilis

NC_000964.3 RefSeq gene 410 1750 . + . ID=gene-
BSU_00010;Name=dnaA;gbkey=Gene;gene=dnaA;gene_biotype=protein_coding;locus_tag=BSU_00010;old_l

NC_000964.3 RefSeq CDS 410 1750 . + 0 ID=cds-NP_387882.1;Parent=gene-
BSU_00010;Dbxref=EnsemblGenomes-Gn:BSU00010,EnsemblGenomes…

Remarque:

Les fichiers GFF semblent ne contenir que 2 lignes par gène et 9 colonnes d'informations après les lignes d'en-tête(#).
Pour chaque gène on a: - Une 1ère ligne qui concerne le gene; - Et une 2ème ligne qui concerne le CDS;

Etape 3: Pour extraire les informations concernant uniquement le gène *trmNF* du fichier d'annotation GFF, - j'utilise la cmd Unix `grep` sur le fichier GFF: pour ne récupérer que les ligne concernant "trmNF"; - Puis je pipe `|` la sortie vers une cmd `awk` pour ne conserver que la colonne 3 qui concerne le gène d'intérêt. - Enfin je redirige la sortie standard vers un fichier "trmNF.gff"

Rmq: Le pipe `|` permet d'envoyer la sortie d'une cmd directement vers une autre cmd.

[ ]:
```
srun grep "trmNF" GCF_000009045.1_ASM904v1_genomic.gff | awk '$3=="gene"' >␣
↪trmNF.gff
```

Puis la cmd `cat` permet de visualiser le contenu du fichier "trmNF.gff" généré sur la sotrt:

NC_000964.3 RefSeq gene 42917 43660 . + . ID=gene-BSU_00340;Name=trmNF;

gbkey=Gene;gene=trmNF;gene_biotype=protein_coding;locus_tag=BSU_00340;old_locus_tag=BSU00340

=> Le fichier d'annotation du gène "trmNF.gff" généré contient effectivement qu'une seule ligne.

Remarque:

1. (NC_000964.3) le chromosome
2. (RefSeq) la source de l'annotation, habituellement l'outil qui a généré l'annotation
3. (gene) le type d'annotation ;
4. (42917) le début de l'annotation
5. (43660) la fin de l'annotation
6. (.) pas important
7. (+) le brin
8. (.) pas important
9. (ID=gene-BSU_00340;Name=…) On peut y trouver le nom usuel du gène et nombre d'autres informations.

Étape 4: Pour générer un fichier "BAM" trié (SRR10390685_sort.bam) on peut utiliser la commande `sort` de "samtools" sur un fichier "BAM". Mais on dois au préalable convertir notre fichier SAM en BAM avec `samtools view`:

Étape 4-a: Lancer `samtools view` pour convertir le fichier SAM en BAM car ce dernier format est moins volumineux donc plus facilement maniable.

La boucle `for` permet la reproductibilité de l'analyse sur un plus grand nombre d'entrées:

[ ]:
```
sam_files=(/shared/projects/dubii2020/nprince/ExamenFin_M4_5/Mapped_DIR/*.sam)
 ↪ # Definition d'une variable

for In_sam in ${sam_files[@]}
    do
        BASE_SAM="$(basename $In_sam .sam)"


    srun --cpus-per-task 16 samtools view -@ 16  -bS Mapped_DIR/${BASE_SAM}.sam
 ↪> Mapped_DIR/${BASE_SAM}.bam & # Commande

    done
```

Remarque-1: Le fichier BAM (SRR10390685.bam) a été généré sans droits d'exécution. On utilise la cmd `chmod ugo+x` pour changer les permissions d'exécution de ce fichier.

[ ]:
```
chmod ugo+x Mapped_DIR/SRR10390685.bam
```

Remarque-2:
Les options: - '-@ 16': permet de Définir le nombre de threads de tri (ici '16') et de compression. Par défaut, l'opération est monothread ce qui peut être assez long lorsque l'on manipule de gros fichiers. - '>': redirige la sortie vers un fichier "${BASE_SAM}.bam" - '&': permet de reprendre la main et de pouvoir suivre la progression sur la sortie standard.

Étape 4-b: Lancer `samtools sort`qui trie les alignements (par les coordonnées de départ) d'un fichier BAM et produit un BAM trié.

La boucle `for` permet la reproductibilité de l'analyse sur un plus grand nombre d'entrées:

[ ]:
```
bam_files=(/shared/projects/dubii2020/nprince/ExamenFin_M4_5/Mapped_DIR/*.bam)
 ↪ # Definition d'une variable

for In_bam in ${bam_files[@]}
    do
        BASE_BAM="$(basename $In_bam .bam)"


    srun --cpus-per-task 16 samtools sort -@ 16 -o Mapped_DIR/
 ↪${BASE_BAM}_sorted.out.bam Mapped_DIR/${BASE_BAM}.bam & # Commande
```

```
    done
```

Remarque-1: Le fichier **BAM trié** a été généré sans droits d'exécution. On utilise la cmd `chmod ugo+x` pour changer les permissions d'exécution de ce fichier.

[ ]:
```
chmod ugo+x Mapped_DIR/SRR10390685_sorted.out.bam
```

Remarque-2:

Les options: - '-o': permet de récupérer le résultat de la sortie standard dans un un fichier de sortie ('SRR10390685_sort.out.bam') - '-@ 16': permet de Définir le nombre de threads de tri (ici '16') et de compression. Par défaut, l'opération est monothread ce qui peut être assez long lorsque l'on manipule de gros fichiers.

Étape 5: Enfin, pour extraire dans un fichier BAM les reads chevauchant à au moins 50% le gène trmNF on utilise la commande `bedtools intersect` de "bedtools" avec les options appropriées:

[ ]:
```
Sorted_bam_files=(/shared/projects/dubii2020/nprince/ExamenFin_M4_5/Mapped_DIR/
↪*sorted.out.bam)    # Definition d'une variable

for bam in ${Sorted_bam_files[@]}
    do
        BASE_BAMsort="$(basename $bam _sorted.out.bam)"
srun bedtools intersect -a Mapped_DIR/${BASE_BAMsort}_sorted.out.bam -b trmNF.
↪gff -f 0.50 -sorted > Mapped_DIR/${BASE_BAMsort}_MidOverlap_trmNF.bam &

done
```

Remarque: "bedtools intersect" permet de déterminer les chevauchements entre deux ensembles d'entités génomiques. bedtools intersect fonctionne avec les fichiers BED / GFF / VCF et BAM en entrée. Il génère des fichiers BAM compressés par défaut.

A noter: Lorsque l'on croise des fichiers très volumineux (ce qui peut vite être le cas avec des fichiers SAM/BAM et GFF et qui peut générer des problèmes d'utilisation excessive de la mémoire) il est préférable de **pré-trier** ses données par chromosome, puis par position de départ, puis d'utiliser l'option -sorted.

Les options de ma ligne de commande `bedtools intersect`: - '-a': Définition du fichier "A" (qui peut être un fichierBAM / BED / GFF / VCF). Chaque caractéristique de A est ainsi comparée à B à la recherche de chevauchements. - '-b': Définition du fichier "B" (qui peut être un fichier BAM / BED / GFF / VCF). - '-f 0.50': Chevauchement minimum requis en tant que fraction de A. En d'autres termes, '-f vaut 0,50', cela signifie que A chevauche au moins 50% de B. - '-sorted': Pour les fichiers volumineux, 'sorted' invoque un algorithme de «balayage» qui nécessite une entrée triée par position. Lorsqu'on utilise -sorted, l'utilisation de la mémoire reste faible même pour les fichiers très volumineux.

Remarque-1: Le fichier **BAM de chevauchement à 50%** a été généré sans droits d'exécution. On utilise la cmd `chmod ugo+x` pour changer les permissions d'exécution de ce fichier.

[ ]:
```
chmod ugo+x Mapped_DIR/SRR10390685_MidOverlap_trmNF.bam
```

=> Ainsi j'ai extrait dans un fichier BAM les reads chevauchant à au moins 50% le gène trmNF.
Ce fichier BAM est visualisable avec des Visualisateurs d'alignement tel que "tablet ou IGV".