Compte Rendu Nextflow

Jean Hétier

11 octobre 2023

Table des matières

1 Introduction **2**

2 Exercice n°1 : Connexion à Genologin et préparation de l'espace de travail **2**
2.1 Connexion à Genologin 2
2.2 Préparation de l'espace de travail 2

3 Exercice n°2 : Préparation du fichier bash de lancement Nextflow **2**
3.1 Préparation du fichier de lancement 2
3.2 Suivi du job avec seff 5
3.3 Intérêt du resume 6

4 Exercice n°3 : Interprétation des résultats **6**
4.1 Interprétation des principaux résultats 6
4.2 Interprétation du rapport MultiQC 6

5 Exercice n°4 : Lancement du pipeline sur des données NCBI **7**
5.1 Choix des échantillons et références fasta 7
5.2 Résultats obtenus 8

6 Annexes **10**

1 Introduction

Ce compte rendu vise à présenter une vue d'ensemble des approches vues lors de la formation, associée à un approfondissement de l'interprétation des commandes utilisées comme des résultats obtenus, en expliquant au maximum toute la démarche et la compréhension personnelle d'outils nombreux et parfois nouveaux qui s'est voulue la plus complète possible.

2 Exercice n°1 : Connexion à Genologin et préparation de l'espace de travail

2.1 Connexion à Genologin

Elle se fait par la commande suivante depuis le bash :

— ssh -XY glaieul@genologin.toulouse.inra.fr

"glaieul" étant le nom du compte de formation attribué à titre personnel et genologin.toulouse.inra.fr étant le nom d'hôte du cluster de calcul.ssh correspond à Secure Shell qui est le processus de connexion à distance, et les options X et Y permettent de faciliter l'affichage d'interfaces graphiques à distance (exemple : ouverture d'une fenetre firefox directement sur le cluster, à distance).Il suffit ensuite de tapper le mot de passe qui nous a été communiqué pour l'occasion et on se trouve sur le /home de notre nom d'utilisateur.

2.2 Préparation de l'espace de travail

Par défaut on retrouve deux dossiers : 'save' et 'work' l'un pour sauvegarder des données de manière pérennes et l'autre comme espace de travailler où peuvent se côtoyer divers fichiers temporaires. C'est dans ce deuxième répertoire que l'on va se placer pour la suite du projet. Création du répertoire : Tout d'abord on crée un répertoire spécifique pour l'exercice à faire qu'on appelle en l'occurence NEXTFLOW. Cela est réalisé à l'aide de la commande 'mkdir'. On se place dans ce répertoire ('cd NEXTFLOW/') puis on organise l'espace de travail en plusieurs dossiers permettant d'ordonner de manière pertinente les fichiers qui seront utilisés. Organisation des sous-répertoires : Ainsi nous crééons plusieurs sous-dossiers dont : -genome qui contiendra le fichier du génome de référence au format fasta -annotation qui contiendra le fichier d'annotation de ce génome au format gtf ou gff -fastq qui contiendra les différents fichiers fastq correspondant aux échantillons de l'expérience considérée -log dans lequel seront stockés les sorties des batchs pour éviter une potentielle accumulation dans le dossier principal -results qui est créé à la suite d'une analyse nextflow et contenant notamment l'analyse multiqc du projet Téléchargement des fichiers :

Pour la première partie du travail concernant les données du 'TP Tomates' l'intégralité des fichiers a pu être récupérée simplement à l'aide de la commande wget suivi du lien

— ("http ://genoweb.toulouse.inra.fr/s̃igenae/sarah/UPS/DATA/")

, après s'être placé dans le dossier correspondant. Pour la deuxième partie et l'utilisation de données NCBI, la tâche s'est avérée plus complexe (fastq-dump ne fonctionnant pas avec la version conseillée de sra-toolkit) et après différentes tentatives de méthodes voici celle qui a été retenue : Soit directement sur le terminal soit dans un batch, module load bioinfo/sra-toolkit.3.0.0 Cela est suivi d'un message indiquant que cette version n'a pas été configurée sur le cluster. On peut cependant le faire simplement à l'aide de la commande vdb-config –interactive. Cela ouvre une fenêtre graphique de paramétrage (voir figure 1) mais il suffit d'appuyer sur f pour régler les paramètres par défaut puis x pour quitter l'interface graphique. Suite à cela, il ne reste plus qu'à réaliser une méthode classique de sra-toolkit : prefetch SRR... puis fasterq-dump -p SRR... (-p permettant de suivre le progrès du téléchargement). Les fastq sont automatiquement split, il n'y à plus qu'à les zipper pour le pipeline.

3 Exercice n°2 : Préparation du fichier bash de lancement Nextflow

3.1 Préparation du fichier de lancement

Voici le fichier bash qui a été utilisé :2. Reprenons ligne par ligne son fonctionnement et le choix des paramètres.

1. #!/bin/bash : Permet d'indiquer que le script doit être exécuté avec Bash.
2. #SBATCH -J HetierJean : Définit le nom du job comme au format "NomPrénom" comme demandé. Sans cela, le nom du job serait par défaut slurm suivi du numéro unique du job.
3. #SBATCH -p workq : Pour soumettre le job spécifiquement à la queue de traitement nommée "workq", bien que ce soit celle utilisée par défaut si cela n'est pas précisé. Elle a la particularité de ne permettre d'utiliser qu'un coeur CPU à la fois et 8GO de RAM au maximum.
4. #SBATCH –mem=6G : Ici on précise la quantité de mémoire désirée pour le job, soit 6 Go comme demandé.

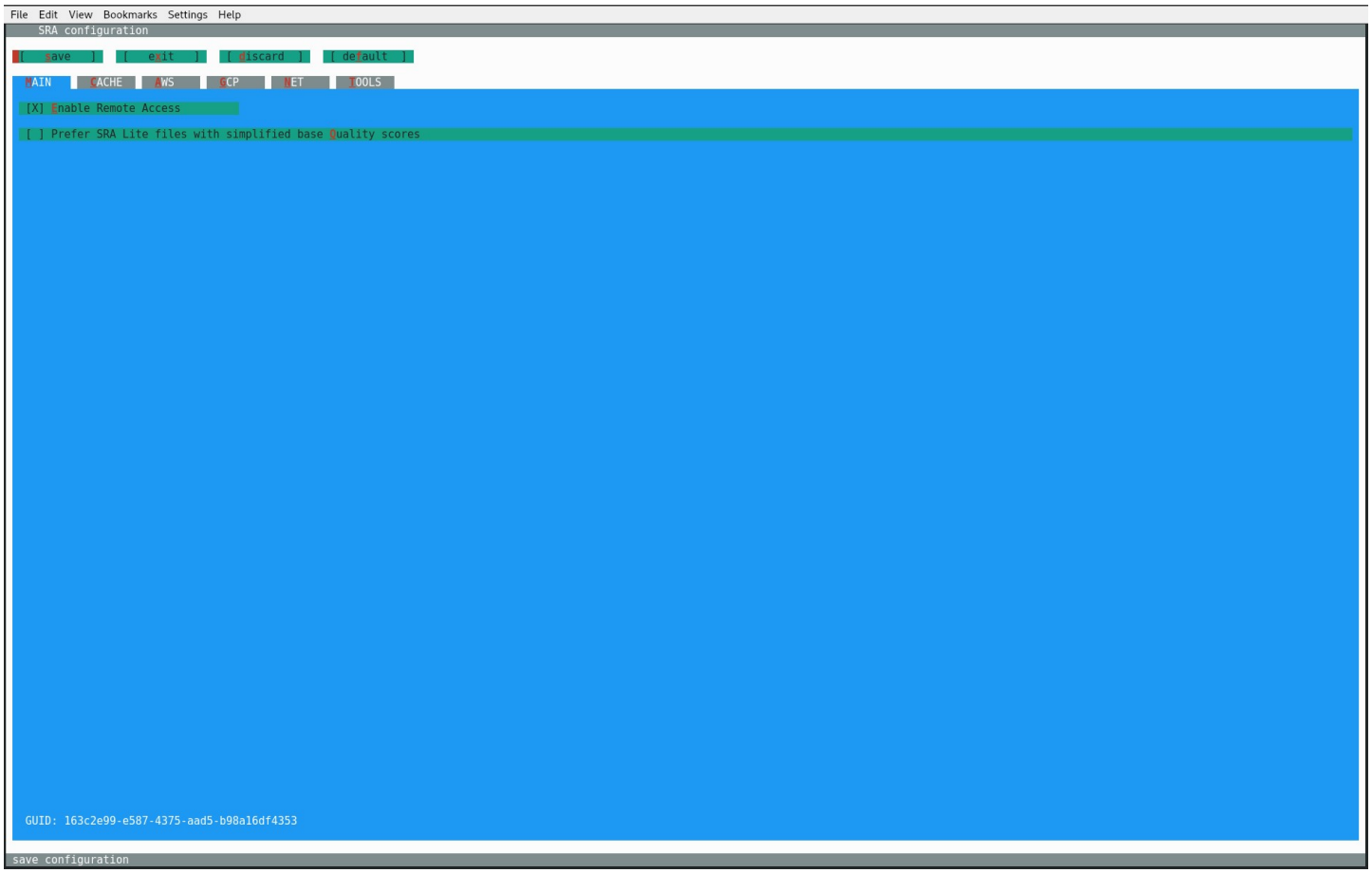

FIGURE 1 – Interface graphique de paramétrage de sratoolkit.

```
glaieul@genologin2 ~/work/NEXTFLOW $ more run_pipeline.sh
#!/bin/bash
#SBATCH -J HetierJean
#SBATCH -p workq
#SBATCH --mem=6G
#SBATCH -t 1-00:00:00
#SBATCH -o /home/glaieul/work/NEXTFLOW/log/%x_%j.out
#SBATCH -e /home/glaieul/work/NEXTFLOW/log/%x_%j.err


module purge
module load bioinfo/nfcore-Nextflow-v21.04.1

input=/home/glaieul/work/NEXTFLOW/inputs.csv
gtf=/home/glaieul/work/NEXTFLOW/annotation/ITAG2.3_genomic_Ch6.gtf
fasta=/home/glaieul/work/NEXTFLOW/genome/ITAG2.3_genomic_Ch6.fasta
config=/home/glaieul/work/NEXTFLOW/sm_config.cfg

nextflow run nf-core/rnaseq -r 3.0 -profile genotoul --input $input --fasta $fasta --gtf $gtf --aligner star_rsem -c $config
```

FIGURE 2 – Fichier bash Nextflow Tomates.

```
glaieul@genologin2 ~/work/NEXTFLOW $ more inputs.csv
group,replicate,fastq_1,fastq_2,strandedness
mutant,1,/home/glaieul/work/NEXTFLOW/FASTQ/MT_rep1_1_Ch6.fastq.gz,/home/glaieul/work/NEXTFLOW/FASTQ/MT_rep1_2_Ch6.fastq.gz,unstranded
wild,1,/home/glaieul/work/NEXTFLOW/FASTQ/WT_rep1_1_Ch6.fastq.gz,/home/glaieul/work/NEXTFLOW/FASTQ/WT_rep1_2_Ch6.fastq.gz,unstranded
```

FIGURE 3 – inputs Tomates.

5. #SBATCH -t 1-00 :00 :00 : Cette option permet de fixer la durée maximale du job à 1 jour. C'est une forme de filet de sécurité, d'une part car on sait d'avance que les traitements à effectuer ne sont pas censés dépasser cette durée et aussi pour éviter qu'un job 'buggé' et oublié continue de consommer des ressources inutilement.
6. #SBATCH -o /home/glaieul/work/NEXTFLOW/log/%x_%j.out : J'ai choisi d'utiliser cette option pour rediriger spécifiquement la sortie standard du job vers un fichier explicitement nommé avec le nom du job et son ID dans le répertoire log présenté en première partie. L'idée était d'avoir une approche plus propre dans l'éventualité d'une multitude de soumissions de jobs.
7. #SBATCH -e /home/glaieul/work/NEXTFLOW/log/%x_v%j.err : Même chose, mais pour rediriger l'éventuelle sortie d'erreur du job.
8. - module purge : Cette commande permet de supprimer tous les modules chargés éventuellement chargés auparavant. Cela permet d'être sur de ne pas avoir d'interférences imprévues.
9. - module load bioinfo/nfcore-Nextflow-v21.04.1 : Ici on charge notre module d'intérêt, c'est à dire le module Nextflow avec la version spécifiée.

Ensuite on déclare nos variables c'est à dire les chemins d'accès pour les différents fichiers nécessaires à l'exécution. Ces variables seront ensuite appelées dans la ligne nextflow pour récupérer les fichiers correspondants.

- nextflow run nf-core/rnaseq -r 3.0 -profile genotoul –input $input –fasta $fasta –gtf $gtf –aligner star_rsem -c $config : Plusieurs choses ici :

1. nextflow run nf-core/rnaseq : Exécution du pipeline RNAseq de nf-core.
2. -r 3.0 : Spécifie la révision du pipeline à utiliser (version 3.0).
3. -profile genotoul : Utilise le profil "genotoul", comme demandé ce qui va probablement permettre de définir des paramètres spécifiques à cet environnement pour l'exécution du job.
4. –input $input : Appelle la variable input contenant le chemin vers le fichier d'entrée, au format csv et dont on expliquera le contenu par la suite.
5. –fasta $fasta : Appelle la variable fasta contenant le chemin vers le fichier FASTA du génome de référence.
6. –gtf $gtf : Appelle la variable gtf contenant le chemin vers le fichier GTF contenant les annotations de ce génome.
7. –aligner star_rsem : Choix de l'aligneur. Ici, l'aligneur STAR couplé à RSEM est utilisé pour l'alignement et la quantification. D'après mes recherches, cette association est la plus souvent utilisée car réputée pour être efficace et précise.
8. -c $config : Appelle la variable input contenant le chemin vers un fichier de configuration personnalisé dont on expliquera le contenu à la suite.

Concernant le fichier inputs :3, il s'agit d'un tableau au format csv, c'est à dire qu'on sépare les colonnes par une virgule ',' et les lignes par un saut de ligne. Le tableau s'organise de la manière suivante : Une première ligne contenant le nom des colonnes soit : - group, ici soit wild soit mutant - replicate pour le numéro du réplicat - fastq1 et fastq2 pour les deux parties du fichier fastq correspondant à l'échantillon - strandedness : pour l'orientation de la séquence, 'unstranded' signifie qu'on ne sait pas.

Quant à lui, le fichier config :4 va permettre de suivre et d'enregistrer l'exécution de chaque tâche de la pipeline, ce qui peut être utile pour comprendre les bugs éventuels. Rentrons dans le détail de ce qui est fait :

1. enabled = true : C'est ce qui active la fonction de "tracing". Si elle était réglée sur false, aucune information de "tracing" ne serait enregistrée.
2. file = 'pipeline_trace.txt' : Ici on spécifie le nom du fichier dans lequel les informations de "tracing" seront enregistrées.
3. fields = 'task_id,name,status,exit,realtime,%cpu,rss,script' : On sélectionne les informations qui seront enregistrées pour chaque tâche. Pour rentrer un peu plus dans le détail :
   (a) task_id : Un identifiant unique pour chaque tâche.
   (b) name : Le nom de la tâche.

```
glaieul@genologin2 ~/work/NEXTFLOW $ more sm_config.cfg
trace {
        enabled = true
        file = 'pipeline_trace.txt'
        fields = 'task_id,name,status,exit,realtime,%cpu,rss,script'
}
```

FIGURE 4 – Fichier de configuration pipeline.

```
glaieul@genologin2 ~/work/NEXTFLOW $ seff 50757554
Job ID: 50757554
Cluster: genobull
User/Group: glaieul/formation
State: COMPLETED (exit code 0)
Cores: 1
CPU Utilized: 00:03:10
CPU Efficiency: 1.08% of 04:53:48 core-walltime
Job Wall-clock time: 04:53:48
Memory Utilized: 1.82 GB
Memory Efficiency: 30.39% of 6.00 GB
```

FIGURE 5 – Interface graphique de paramétrage de sratoolkit.

(c) status : L'état de la tâche (par exemple, si elle a été complétée avec succès, si elle a échoué, etc...)
(d) exit : Le code de sortie de la tâche (0=success...).
(e) realtime : Comme son nom l'indique, le temps que la tâche a pris à s'exécuter.
(f) %cpu : La proportion du temps CPU utilisé par la tâche.
(g) rss : La "Resident Set Size", qui est la portion de la mémoire occupée par une tâche qui est gardée dans la mémoire principale (RAM).
(h) script : Le script qui est exécuté pour cette tâche.

3.2 Suivi du job avec seff

seff est un outil de résumé des jobs slurm sur cluster. Il permet un suivi à la fois de l'état d'avancement du workflow lancé (donc de savoir s'il a réussi ou échoué) mais permet aussi de donner une vue rapide de la façon dont le workflow performe par rapport aux ressources demandées. Rentrons maintenant dans le détail des informations données ligne par ligne. On prendra ici l'exemple du batch/job utilisé sur les données NCBI.

1. Job ID : Identifiant unique du job.
2. Cluster : Nom du cluster.
3. User/Group : Nom d'utilisateur et groupe de l'utilisateur qui a soumis le job.
4. State : État final du job. Dans cet exemple (5), il a été complété avec succès (exit code 0). D'après mon expérience, il y a 4 états possibles : COMPLETED comme c'est le cas ici, FAILED lorsque le job a rencontré une erreur lors de son exécution, dans ce cas on peut trouver une explication dans le fichier .out du job, il y a l'état CANCELLED lorsqu'on décide d'interrompre manuellement un job à l'aide de la commande 'scancel' et enfin l'état RUNNING quand le job est en cours d'éxecution.
5. Cores : Nombre de coeurs CPU demandés pour le job, dans notre cas, un seul (ce qui peut expliquer des temps d'executions parfois longs).
6. CPU Utilized : C'est le temps total pendant lequel le job a réellement utilisé le CPU. Dans ce cas, il a utilisé le CPU pendant seulement 3 minutes et 10 secondes.

7. CPU Efficiency : C'est le pourcentage du temps pendant lequel le CPU a été utilisé par rapport au temps total disponible (en considérant le nombre de coeurs). Ce job donc a utilisé seulement 1,08 pourcent du temps CPU disponible, ce qui parait étonnant et extremement faible.
8. Job Wall-clock time : C'est la durée totale d'exécution du job,ici 4 heures, 53 minutes et 48 secondes.
9. Memory Utilized : La quantité de mémoire que le job a utilisé, en l'occurrence 1,82 Go de RAM.
10. Memory Efficiency : Pourcentage de la mémoire que le job a utilisé par rapport à la mémoire que vous demandée. 6 Go avaient été demandés et donc le job n'a utilisé que 30,39 pourcents de cette valeur.

Ce job est intéressant car il permet d'observer un décalage important entre le temps d'utilisation du CPU et le temps total d'exécution. Une piste d'explication possible pourrait être en lien avec la taille des fichiers utilisés, relativement importante (quelques dizaines de Go).On appelle cela les attentes d'Entrées/Sorties (E/S) : Si le job fait beaucoup d'opérations d'E/S, par exemple en lisant ou écrivant de gros fichiers sur le disque, il peut passer beaucoup de temps à attendre que ces opérations se terminent, pendant lesquelles le CPU n'est pas activement utilisé.

3.3 Intérêt du resume

C'est une fonctionnalité très utile de Nextflow car elle permet la reprise d'un workflow à partir du point où il s'est arrêté lors d'un précédent 'fail', plutôt que de recommencer depuis le début. Cela permet donc dans certains cas une économie évidente de temps sur de gros workflows ainsi que de ressources car les calculs déjà faits ne sont pas à refaire. Cela peut aussi être très utile en cas de pannes imprévues et indépendantes du workflow (réseau, matériel ou autres) car on pourra reprendre au point d'arrêt.

4 Exercice n°3 : Interprétation des résultats

4.1 Interprétation des principaux résultats

Tout d'abord on constate que le dossier résultats créé se décompose en 6 sous-dossiers différents. Expliquons rapidement à quoi chacun correspond et les principaux fichiers contenus.

1. fastqc : Ce dossier contient les résultats de l'outil FastQC, servant lui même à l'analyse de la qualité des données brutes de séquençage.Pour chaque échantillon, il génère un rapport qui fournit des informations sur la qualité des séquences, la distribution de la longueur des séquences ou la répartition des séquences sur les bases entre autres choses.On aura classiquement tendance à regarder ce rapport en première instance pour vérifier la qualité initiale des données avant de poursuivre les analyses suivantes.
2. genome : Il contient les fichiers et les indices associés au génome de référence qui a été utilisé pour l'alignement des séquences.Cela inclut donc le génome lui-même, les indices pour l'aligneur, et d'autres fichiers annexes nécessaires pour certaines étapes du pipeline.
3. multiqc : MultiQC compile les résultats de nombreux autres outils de contrôle de qualité et on va pouvoir notamment y trouver un rapport unique en faisant la synthèse au format html.C'est un rapport très exhaustif contenant quasiment toutes les étapes de la pipeline et les différents outils/mesures de qualité.
4. pipeline_info : Ce dossier contient des informations générales sur la pipeline elle-même.On peut y trouver des informations sur la version de la pipeline, et d'autres détails de métadonnées sur l'exécution elle-même.
5. star_rsem : Ce dossier contient les résultats de l'alignement et de la quantification.STAR est l'outil utilisé pour aligner les séquences sur le génome de référence.RSEM est ensuite utilisé pour quantifier les transcrits ou les gènes basés sur les alignements.Comme nous l'évoquions précédemment, cette association d'outils est couramment faite. On y trouve donc des fichiers BAM représentant les alignements, ainsi que des matrices de comptage et d'autres sorties de quantification.
6. trimgalore : trimgalore est un outil utilisé pour nettoyer les données brutes de séquençage.Il élimine les adaptateurs et les bases de faible qualité des séquences. Il permet d'automatiser des choses que l'on pourrait faire avec l'outil cutadapt par exemple, qu'il m'était déjà arrivé d'utiliser à d'autres occasions pour l'anecdote.Dans le dossier on trouve donc notamment les données nettoyées après l'élagage, et des rapports sur ce qui a été élagué et pourquoi.

4.2 Interprétation du rapport MultiQC

Le plus gros des résultats se trouve au niveau du rapport MultiQC, particulièrement dense et dont nous allons essayer de faire la synthèse la plus pertinente possible. Par soucis de concision et de non redondance, il s'agira dans

cette partie d'expliquer de manière générale les différents outils présents et ce que j'ai compris de leur rôle et nous ferons une interprétation plus détaillée des résultats présents pour la partie sur les données NCBI. Le rapport se compose de plusieurs grandes parties qui sont les suivantes :

1. General Statistics : Tableau très complet reprenant une bonne partie des métriques mesurées dans le rapport, permettant une comparaison rapide entre échantillons et permettre par exemple de voir rapidement si un échantillon mal mappé est par exemple corrélé avec un certain pourcentage de GC ou de duplication.
2. Biotype Counts : Cette section reflète la répartition des lectures (reads) en fonction des différents types biologiques (biotypes) des caractéristiques génomiques. Les biotypes peuvent comprendre une variété de catégories, comme codant pour une protéine, miRNA, pseudogenes et bien d'autres .
3. DupRadar : Cet outil évalue la duplication dans les données de séquençage.
4. Picard - Mark Duplicates : Outil générant un histogramme qui montre la distribution des lectures en fonction de leur distance optique ou leur position de départ. Cela aide à distinguer les duplicatas vrais (techniques) des duplicatas qui sont attendus en raison de la biologie (par exemple, des gènes hautement exprimés).
5. Preseq - Complexity curve : C'est un outil pour estimer la complexité des bibliothèques de séquençage, autrement dit le nombre de séquences uniques qui pourraient être obtenues si l'on continuait à séquencer davantage. Il s'agit de comparer la courbe expérimentale avec la courbe idéale où un read= une séquence unique. Plus la courbe expérimentale s'en rapproche, plus on est content
6. QualiMap : Permet d'avoir un aperçu de l'origine génomique des reads. On attend une majorité d'exons c'est à dire les régions codantes. En effet les introns sont éliminés lors de l'épissage et les régions intergéniques sont entre les gènes comme leur nom l'indique. Si on mappe sur ces régions a un taux élevé c'est probablement qu'il y a un problème de design pour l'alignement.
7. Rsem : Pour estimer l'abondance d'isoformes avec notamment un affichage de la distribution de multimapping, c'est à dire qu'on peut le nombre de reads s'étant alignés un certain nombre de fois. On a généralement un fort pic au début puis une courbe plate correspondant aux outliers.
8. RSeQC : Cet outil permet notamment d'avoir des informations encore plus détaillées que qualimap sur la distribution des lectures sur les régions génomiques. On pourra par exemple savoir si le read s'aligne sur la partie 3' ou 5' d'un exon. En cas de problème d'alignement, ça doit pouvoir permettre d'identifier encore plus finement une zone mal calibrée.
9. Samtools : outil fondamental pour manipuler les formats de fichiers comme BAM et SAM et les 'parser'. Il permet entre autres la conversion de formats, le tri des lectures, l'indexation, et la détection de variants, permettant la fluidité et l'efficacité du pipeline.
10. FastQC : c'est un incontournable pour analyser la qualité des données de séquençage. Il offre une large palette d'outils d'évaluation de qualité et permet une compréhension précise du comportement des données. Il y a les parties raw et trimmed qui permettent de bien voir la différence avant/après élagage des adaptateurs.
11. Cutadapt : On retrouve en fait cutadapt ici, utilisé par trimgalore pour couper les adaptateurs des séquences. Si on ne le faisait pas, on aurait un problème car on incluerait une partie redondante pour toutes les séquences.
12. nf-core/rnaseq : Paragraphe qui reprend tous les paramètres utilisés pour la pipeline, comme par exemple version spécifique du pipeline nf-core pour l'analyse.

5 Exercice n°4 : Lancement du pipeline sur des données NCBI

5.1 Choix des échantillons et références fasta

Concernant le choix des échantillons sur NCBI, il s'est fait sur la base du mail commun à ce sujet, car les tentatives individuelles de chacun ne furent pas toujours fructueuses. J'ai donc choisi de travailler directement sur ces SRR en l'occurence des échantillons de Gadus morhua : SRR2045415 , SRR2045416, SRR2045417 correspondant respectivement à des échantillons de cerveau, de branchies et d'ovaires pour cette espèce. La démarche de récupération des données a été expliquée plus tôt dans le rapport.

Concernant le script d'execution, les paramètres sont les mêmes que pour le précédent exercice, il a suffit de changer les variables dans le fichier sh pour donner les chemins vers les nouveaux fichiers (voir Annexes). On change aussi le nom du fichier lui même, pour ne pas écraser le travail fait précédemment. Dans la foulée, il fallait aussi reprendre le fichier d'inputs pour avoir les fichiers appropriés (voir Annexes)

Sample Name	M Reads	% rRNA	dupInt	% Dups	5'-3' bias	M Aligned	% Alignable	% Proper	Error
brain_R1	59.4	0.53%	0.01%	9.9%	1.21	27.0	66.1%	64.3%	0.61%
gills_R1	68.6	2.80%	0.07%	33.2%	1.35	29.5	86.4%	50.1%	0.59%
ovary_R1	47.7	0.23%	0.09%	22.0%	1.24	20.0	92.2%	58.6%	1.01%

TABLE 1 – Tableau 1 : Mesures des échantillons (Partie 1)

Sample Name	M Non-Primary	M Reads	% Mapped	% Proper	M Total	% Dups	% GC	M Seqs
brain_R1	5.3	54.1	80.7%	80.5%	67.1	39.8%	50%	33.5
gills_R1	9.5	59.1	92.3%	92.1%	64.1	55.5%	49%	32.0
ovary_R1	7.5	40.2	94.6%	94.4%	42.4	47.0%	53%	21.2

TABLE 2 – Tableau 2 : Mesures des échantillons (Partie 2)

5.2 Résultats obtenus

Commençons par le tableau General Statistics que j'ai retranscrit ici en LaTeX :
Ce que je comprends à la lecture de ce tableau :

1. Qualité des échantillons : Les échantillons "brain_R1" et "gills_R1" ont des taux d'alignement (% Alignable) relativement élevés, ce qui est généralement un signe d'une bonne qualité d'échantillonnage et de séquençage. L'échantillon "ovary_R1", quant à lui, présente également un taux d'alignement élevé, bien que légèrement inférieur aux deux autres.
2. Contamination par de l'ARN ribosomique (rRNA) : Les échantillons "brain_R1" et "ovary_R1" ont des taux très faibles de rRNA, ce qui est bon car une faible contamination par du rRNA signifie que la plupart des lectures sont représentatives de l'ARN messager. "gills_R1" a un taux de rRNA légèrement plus élevé, mais il reste modéré.
3. Duplication : L'échantillon "gills_R1" a le taux de duplications (% Dups) le plus élevé parmi les trois, ce qui pourrait indiquer soit un biais technique pendant la préparation de la bibliothèque, soit une abondance réelle de certaines séquences. Une meilleure connaissance du design expérimental permettrait d'approfondir cette remarque.
4. Taux d'erreur : Les trois échantillons ont des taux d'erreur assez faibles, ce qui est également un bon signe quant à la qualité du séquençage.
5. Efficacité du séquençage : En général, un taux élevé de paires appropriées (% Proper Pairs) est un signe positif. Cela suggère que la plupart des lectures peuvent être correctement appariées à leur contrepartie, indiquant une bonne qualité du séquençage.

On a globalement un séquençage de bonne qualité, on va pouvoir comparer les différences entre types d'organes et en théorie pouvoir en faire des interprétations biologiques. Analysons maintenant les principales métriques calculées dans ce pipeline :

1. Biotype counts :6 : On observe une écrasante majorité de biotype 'protein coding' ce qui est attendu et rassurant sur la qualité de l'analyse.
2. DupRadar :7 : On peut observer ici une progression relativement linéaire du taux de duplication en fonction du nombre de readsmais cela parait cohérent et normal car les gènes qui sont fortement exprimés auront un grand nombre de reads, et comme ces reads proviennent du même gène, ils seront considérés comme des duplicatas.
3. Picard :8 : On remarque une différence marquée entre brain r1 présente 72% de unique pairs et 8% de duplicate pairs non optical ainsi que 18% de unmaped tandis que gills r1 présente 62% de unique pairs mais 30% de duplicate pairs non optical. Bien que cela corresponde globalement à des pourcentages de lecture uniques plutôt bons, on constate un taux significativement plus élevé de duplicatas dans les échantillons de branchies. Deux pistes possibles à mon sens : soit problème lors de la préparation de la bibliothèque, peut-être une amplification excessive lors des étapes de PCR, soit simplement dû à la nature de l'échantillon, si certaines régions ou transcrits sont très abondants dans les branchies en particulier.
4. Preseq :9 : Résultats qui semblent cohérents avec les précédents : pour le cerveau, une courbe de relative bonne qualité mais avec une dégradation progressive correspondant au nombre croissant de duplicatas. On observe par contre une différence marquée avec les deux autres régions, qui montrent un plafonnement

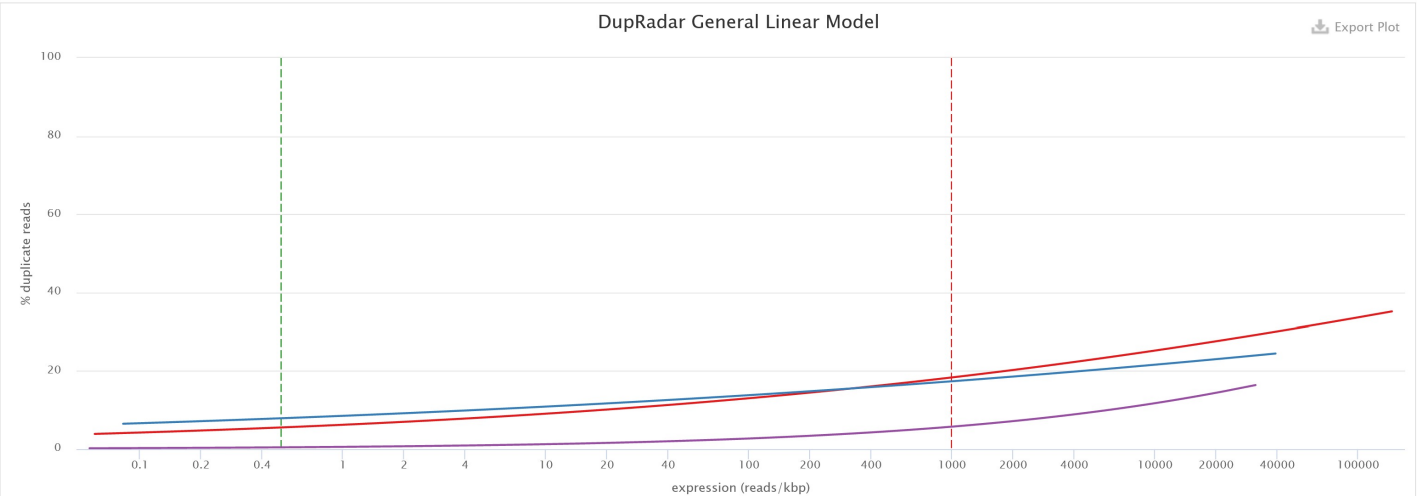

FIGURE 6 – Outil Biotype counts.

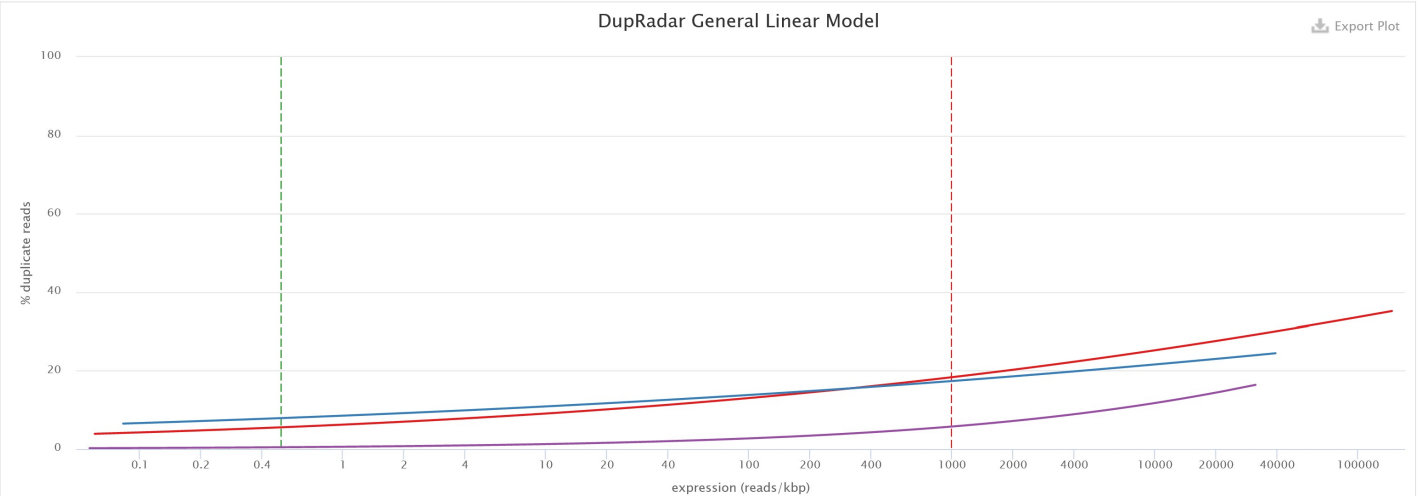

FIGURE 7 – Outil DupRadar.

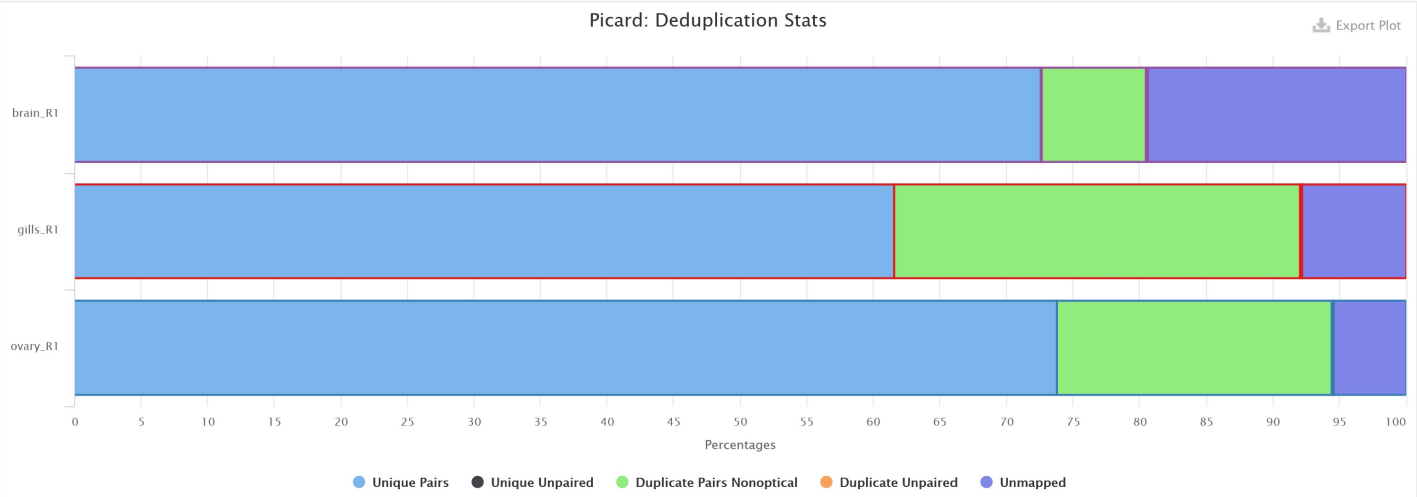

FIGURE 8 – Outil Picard.

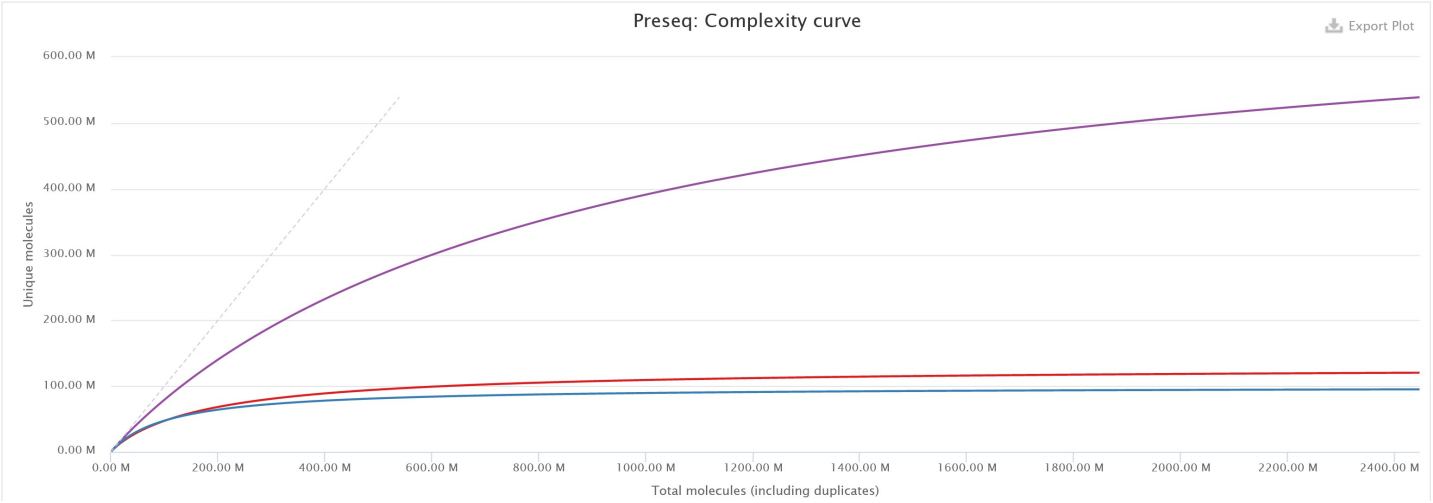

FIGURE 9 – Outil Preseq.

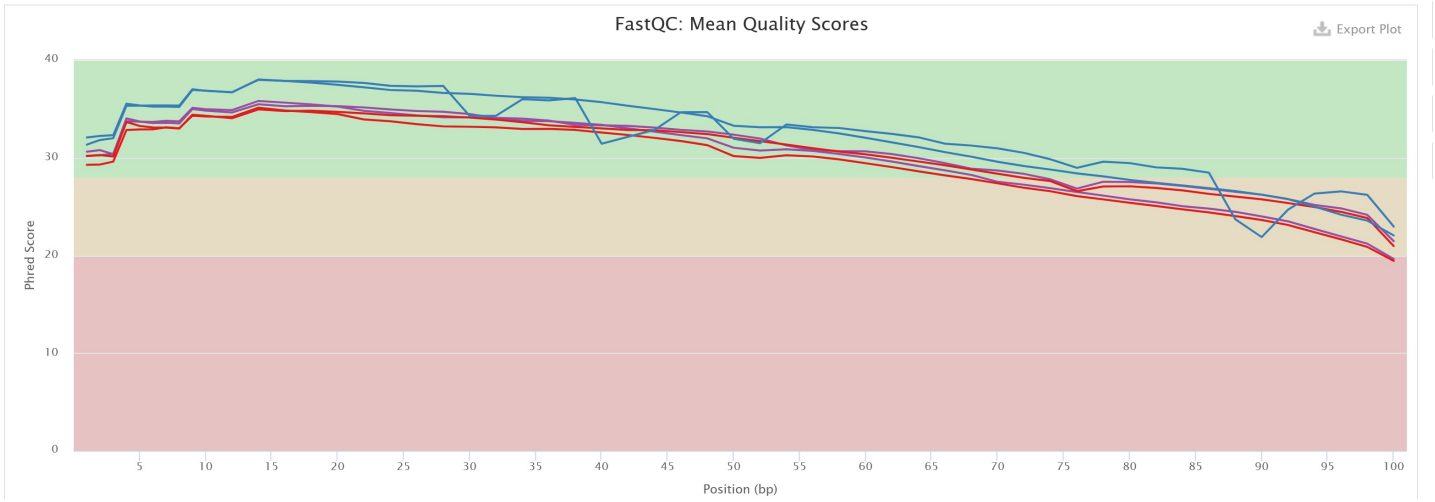

FIGURE 10 – Qualité Avant.

rapide et donc une faible complexité de la bibliothèque. A nouveau, soit cela peut réveler des problèmes techniques tels que l'amplification excessive au cours des étapes de PCR ou encore des contaminations ou bien tout simplement une quantité d'ADN de départ trop faible, soit cela correspond au type d'échantillon, mais cela paraitrait étonnant car ce genre de profil est plutôt attendu sur des études de cellules uniques.

5. FastQC : On peut observer entre une nette différence avant/après l'élagage des adapteurs en termes de qualité des reads. Avant :10, on observe une nette décroissance de qualité en fonction du nombre de bases alors qu'après :11 on a une courbe beaucoup plus homogène et exploitable.

En conclusion, on a ici des données de bonne qualité et qui paraissent exploitables mais un doute subsiste sur le taux de duplication qui pourrait évoquer des erreurs techniques.

Sur un bilan plus global et à titre personnel, ce travail m'a permis d'apprécier la grande variété d'outils à la disposition du bioinformaticien et la rigueur qui est de mise pour faire fonctionner une pipeline même relativement simple de bout en bout.

6 Annexes

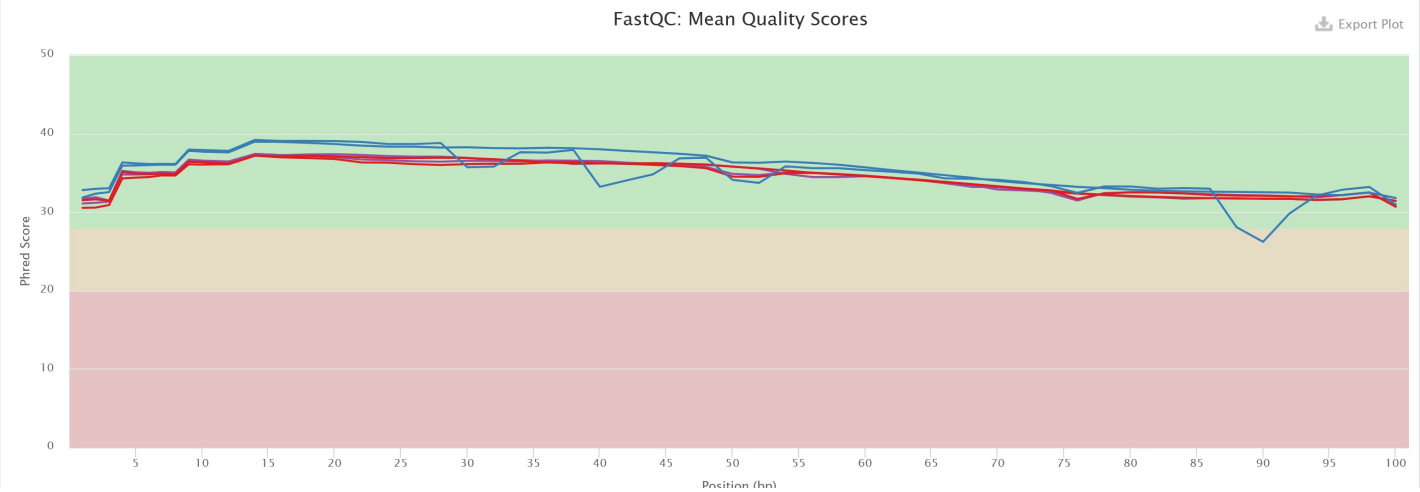

FIGURE 11 – Qualité Après.

```
glaieul@genologin2 ~/work/NEXTFLOW $ more run_pipeline2.sh
#!/bin/bash
#SBATCH -J HetierJean
#SBATCH -p workq
#SBATCH --mem=6G
#SBATCH -t 1-00:00:00
#SBATCH -o /home/glaieul/work/NEXTFLOW/log/%x_%j.out
#SBATCH -e /home/glaieul/work/NEXTFLOW/log/%x_%j.err


module purge
module load bioinfo/nfcore-Nextflow-v21.04.1

input=/home/glaieul/work/NEXTFLOW/inputs2.csv
gtf=/home/glaieul/work/NEXTFLOW/Gadus_morhua.gadMor3.0.110.gtf
fasta=/home/glaieul/work/NEXTFLOW/Gadus_morhua.gadMor3.0.dna.toplevel.fa
config=/home/glaieul/work/NEXTFLOW/sm_config.cfg

nextflow run nf-core/rnaseq -r 3.0 -profile genotoul --input $input --fasta $fasta --gtf $gtf --aligner star_rsem -c $config
```

FIGURE 12 – Fichier sh Nextflow NCBI.

```
glaieul@genologin2 ~/work/NEXTFLOW $ more inputs2.csv
group,replicate,fastq_1,fastq_2,strandedness
ovary,1,/home/glaieul/work/fastq/SRR2045415_1.fastq.gz,/home/glaieul/work/fastq/SRR2045415_2.fastq.gz,unstranded
brain,1,/home/glaieul/work/fastq/SRR2045416_1.fastq.gz,/home/glaieul/work/fastq/SRR2045416_2.fastq.gz,unstranded
gills,1,/home/glaieul/work/fastq/SRR2045417_1.fastq.gz,/home/glaieul/work/fastq/SRR2045417_2.fastq.gz,unstranded
```

FIGURE 13 – inputs NCBI.