

## Projet Nextflow

### Exercice 1 :

Connexion à génologin :

```

10/10/2022 14:42.29 /home/mobaxterm ssh -XY jonquille@genologin.
toulouse.inrae.fr
jonquille@genologin.toulouse.inrae.fr's password:
jonquille@genologin.toulouse.inrae.fr's password:
Last failed login: Mon Oct 10 14:42:43 CEST 2022 from lfbn-tou-1-114-140.w86-201
.abo.wanadoo.fr on ssh:notty
There was 1 failed login attempt since the last successful login.
Last login: Mon Oct 10 14:39:29 2022 from lfbn-tou-1-114-140.w86-201.abo.wanadoo
.fr

```

Ensuite dans le répertoire nextflow puis tomatoes j'ai téléchargé les données ainsi que créer le fichier .sh pour lancer la commande. J'exécuterai par la suite le .sh dans ce répertoire également.

```

jonquille@genologin2 ~/work/nextflow/tomatoes $ ls
ITAG2.3_genomic_Ch6.fasta  MT_rep1_1_Ch6.fastq.gz  results  scriptrna.sh  work  WT_rep1_2_Ch6.fastq.gz
ITAG2.3_genomic_Ch6.gtf   MT_rep1_2_Ch6.fastq.gz  samplesheet.csv  slurm-37374184.out  WT_rep1_1_Ch6.fastq.gz

```

### Exercice 2 :

Voici le fichier sh qui est nommé *scriptrna.sh* :

```

GNU nano 2.3.1 File: scriptrna.sh
#!/bin/bash
#SBATCH -p workq
#SBATCH --time=1-0
#SBATCH --cpus-per-task=8
#SBATCH --mem=6G
#SBATCH -J IlanFIJALKOW

module purge

module load bioinfo/Nextflow-v21.10.6
module load system/singularity-3.5.3

nextflow run nf-core/rnaseq --reads ~/work/nextflow/tomatoes/MT_rep1_1_Ch6.fastq.gz,~/work/nextflow/tomatoes/MT_rep1_2_Ch6.fastq.gz,~/work/nextflow/tomatoes/

```

La commande en entier sur la dernière ligne est la suivante :

```
nextflow run nf-core/rnaseq --reads
~/work/nextflow/tomatoes/MT_rep1_1_Ch6.fastq.gz,~/work/nextflow/tomat
oes/MT_rep1_2_Ch6.fastq.gz,~/work/nextflow/tomatoes/WT_rep1_1_Ch6.f
astq.gz,~/work/nextflow/tomatoes/WT_rep1_2_Ch6.fastq.gz --outdir results
--fasta ~/work/nextflow/tomatoes/ITAG2.3_genomic_Ch6..fasta --gtf
~/work/nextflow/tomatoes/ITAG2.3_genomic_Ch6.gtf -profile genotoul --
input samplesheet.csv
```

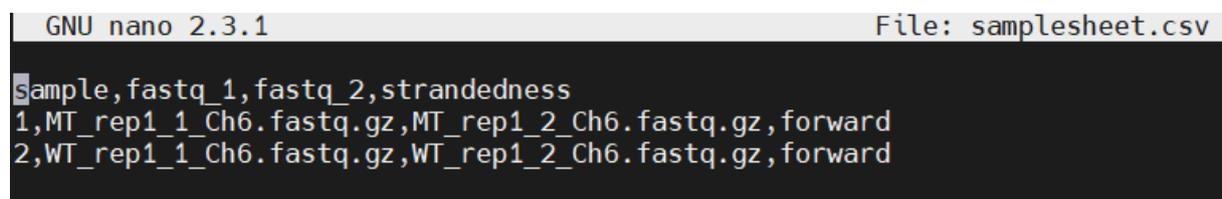
Les paramètres du fichier sh sont ceux demandé dans l'énoncé soit :

- De lancer le job dans la workq
- Le temps d'exécution est d'un jour maximum
- J'ai alloué 8 CPU pour exécuter la commande afin qu'elle puisse se réaliser sans problème.
- La mémoire du job est fixée au maximum à 6G
- Le nom du job est prénomNOM

J'ai ensuite chargé les module nextflow avec la version 21.10.6 (celle qui marchait le mieux pour ma commande) et le module singularity avec la version 3.5.3 sinon la commande ne s'exécutait pas. Bien sur plusieurs tests ont été réalisé avant de trouver les versions et les modules adéquats. Afin de comprendre les erreurs issues de chaque commande je regardais le retour de celle-ci qui se situait dans le fichier *slurm.out*. Au fur à mesure j'ajustait donc les paramètres.

Dans la commande j'ai utilisé les datas téléchargées précédemment. Le fichier de sortie est nommé results. Je me suis positionné sur le profile genotoul et en input j'ai également donné un fichier *samplesheet.csv* afin que la commande s'exécute correctement.

Voici le fichier renseigné en input :



```
GNU nano 2.3.1 File: samplesheet.csv
Sample,fastq_1,fastq_2,strandedness
1,MT_rep1_1_Ch6.fastq.gz,MT_rep1_2_Ch6.fastq.gz,forward
2,WT_rep1_1_Ch6.fastq.gz,WT_rep1_2_Ch6.fastq.gz,forward
```

Pour suivre l'exécution du job on utilisait la commande **seff numérodujob**.

Le dernier seff obtenu pour ma commande ayant marché est le suivant :

```
jonquille@genologin2 ~/work/nextflow/tomatoes $ seff 37374184
Job ID: 37374184
Cluster: genobull
User/Group: jonquille/formation
State: COMPLETED (exit code 0)
Nodes: 1
Cores per node: 8
CPU Utilized: 00:02:21
CPU Efficiency: 2.75% of 01:25:28 core-walltime
Job Wall-clock time: 00:10:41
Memory Utilized: 1.19 GB
Memory Efficiency: 19.80% of 6.00 GB
```

On peut voir en première ligne l'identifiant du job ou numéro du job.

Sur la deuxième ligne, le cluster sur lequel a été exécuté le job. On peut voir ensuite l'utilisateur ayant lancé le job ainsi que le groupe associé, ici l'utilisateur était moi (Jonquille) et mon groupe est le groupe formation. On peut ensuite voir l'état du job, ici il est *Completed* ce qui signifie qu'il est terminé. Lorsque l'exécution rencontrait un problème, il y avait marqué *Failed* à cet endroit-là (C'est à ce moment que l'on consultait le slurm afin de voir où la commande avait planté). On peut ensuite voir le nombre de nœud sur lequel le job est exécuté. A la ligne suivante, le nombre de CPU par nœud alloué est spécifié. Les CPU ont été utilisés 2min21 et ont été efficaces à 2.75% dans le sens ou seulement 2.75% des capacité des CPU ont été utilisés. Le temps d'exécution du job est de 10min41, la mémoire utilisée est de 1.19GB ce qui est environ 19.80% de la mémoire allouée à la base.

En bref cette commande permet d'afficher les ressources utilisées par un job ainsi que le calcul de son efficacité. On peut voir ici que mon job n'a pas utilisé un haut pourcentage de CPU pour réaliser le job et également un faible pourcentage de mémoire. Il n'a donc pas été efficace en termes d'optimisation et d'utilisation des ressources qui lui étaient données.

### **Exercice 3 :**

Commençons par expliquer les principaux répertoires et fichiers de sorties de la commande se trouvant dans le répertoire *results* obtenu à la suite de l'exécution.

```

jonquille@genologin2 ~/work/nextflow/tomatoes/results $ ls
fastqc genome multiqc pipeline_info star_salmon trimgalore
jonquille@genologin2 ~/work/nextflow/tomatoes/results $ cd genome/
jonquille@genologin2 ~/work/nextflow/tomatoes/results/genome $ ls
genome.transcripts.fa index ITAG2.3_genomic_Ch6.bed ITAG2.3_genomic_Ch6.fasta.fai ITAG2.3_genomic_Ch6.fasta.sizes ITAG2.3_genomic_Ch6_genes.gtf rsem
jonquille@genologin2 ~/work/nextflow/tomatoes/results/genome $ cd ..
jonquille@genologin2 ~/work/nextflow/tomatoes/results $ cd pipeline_info/
jonquille@genologin2 ~/work/nextflow/tomatoes/results/pipeline_info $ ls
execution_report_2022-09-30_10-34-39.html execution_trace_2022-09-30_10-24-21.txt pipeline_dag_2022-09-30_10-34-39.html
execution_report_2022-09-30_10-44-28.html execution_trace_2022-09-30_10-26-05.txt pipeline_dag_2022-09-30_10-44-28.html
execution_report_2022-09-30_10-46-18.html execution_trace_2022-09-30_10-27-08.txt pipeline_dag_2022-09-30_10-46-18.html
execution_timeline_2022-09-30_10-34-39.html execution_trace_2022-09-30_10-32-14.txt samplesheet.valid.csv
execution_timeline_2022-09-30_10-44-28.html execution_trace_2022-09-30_10-34-39.txt software_versions.yml
execution_timeline_2022-09-30_10-46-18.html execution_trace_2022-09-30_10-44-28.txt
execution_trace_2022-09-30_10-17-52.txt execution_trace_2022-09-30_10-46-18.txt
jonquille@genologin2 ~/work/nextflow/tomatoes/results/pipeline_info $ cd ..
jonquille@genologin2 ~/work/nextflow/tomatoes/results $ cd star_salmon/
jonquille@genologin2 ~/work/nextflow/tomatoes/results/star_salmon $ ls
1 bigwig preseq salmon.merged.gene_counts_scaled.rds salmon.merged.transcript_tpm.tsv
1.markdup.sorted.bam deseq2 qc qualmap salmon.merged.gene_counts_scaled.tsv salmon_tx2gene.tsv
1.markdup.sorted.bam.bai dupradar rseqc salmon.merged.gene_counts.tsv samtools_stats
2 featurecounts salmon.merged.gene_counts_length_scaled.rds salmon.merged.gene_tpm.tsv stringtie
2.markdup.sorted.bam log salmon.merged.gene_counts_length_scaled.tsv salmon.merged.transcript_counts.rds
2.markdup.sorted.bam.bai picard_metrics salmon.merged.gene_counts.rds salmon.merged.transcript_counts.tsv
jonquille@genologin2 ~/work/nextflow/tomatoes/results/star_salmon $ cd ..
jonquille@genologin2 ~/work/nextflow/tomatoes/results $ cd trimgalore
jonquille@genologin2 ~/work/nextflow/tomatoes/results/trimgalore $ ls
1_1.fastq.gz_trimming_report.txt 1_2.fastq.gz_trimming_report.txt 2_1.fastq.gz_trimming_report.txt 2_2.fastq.gz_trimming_report.txt fastqc
jonquille@genologin2 ~/work/nextflow/tomatoes/results/trimgalore $ cd ..
jonquille@genologin2 ~/work/nextflow/tomatoes/results $ cd fastqc/
jonquille@genologin2 ~/work/nextflow/tomatoes/results/fastqc $ ls
1_1_fastqc.html 1_1_fastqc.zip 1_2_fastqc.html 1_2_fastqc.zip 2_1_fastqc.html 2_1_fastqc.zip 2_2_fastqc.html 2_2_fastqc.zip

```

Comme on peut le voir à l'aide des commandes de navigations shell ci-dessus, le répertoire *results* contient 6 autres répertoires :

*fastqc*, *genome*, *multiqc*, *pipeline\_info*, *star\_salmon* et *trimgalore*.

Premièrement, le répertoire *genome* va contenir plusieurs fichiers d'annotations du génome étudié. Le répertoire *pipeline\_info* lui va contenir toutes les informations liées à l'exécution de la commande.

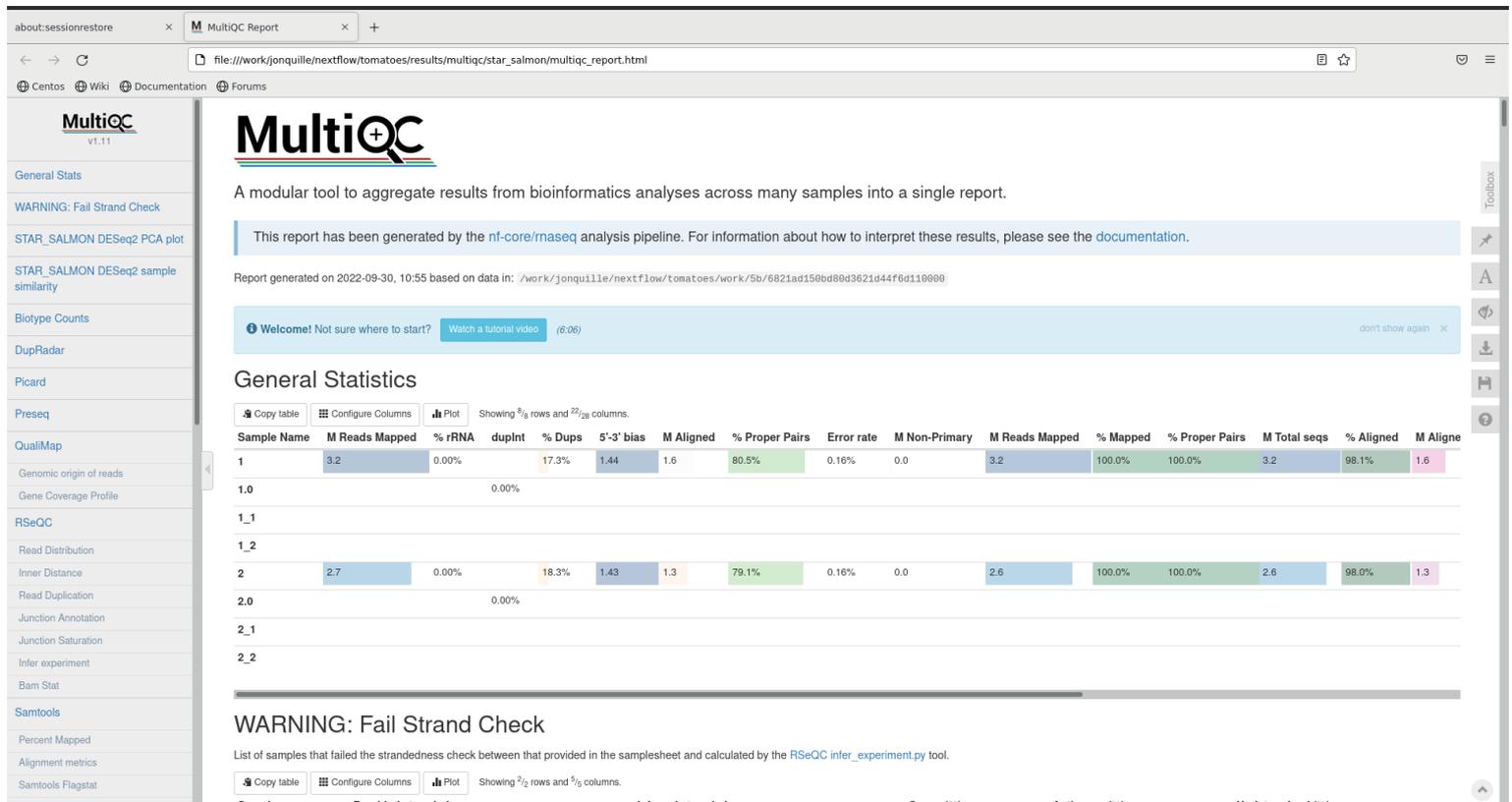
Pour le dossier *trimgalore* en effectuant des recherches sur internet j'ai trouvé qu'il s'agissait d'un ajustement de la qualité et des adaptateurs et de l'élimination de la contamination de ceux-ci tout en faisant un contrôle qualité des séquences et en les élaguant si des régions sont de faibles qualités. J'imagine donc que ce dossier contient les rapports des séquences traités avec l'outil *trimgalore*.

Pour le répertoire *star\_salmon* il s'agit des différents modules utilisés pour représenter les résultats de la commandes (comme DESeq2 par exemple).

Enfin le répertoire *fastqc* et *multiqc* contiennent respectivement les différents rapports fastqc pour chaque échantillon et le rapport multiqc utilisant ces fastqc.

Intéressons-nous maintenant plus en profondeur sur le rapport multiQC :

Voici le fichier *multiqc\_report.html* donnant tout les graphiques et analyses réalisées. Ce fichier multiQC permet de regrouper toutes les informations provenant des multiples rapports de fastQC.



Un multiQC est souvent utilisé lors de la présence de plusieurs échantillons ce qui est notre cas ici.

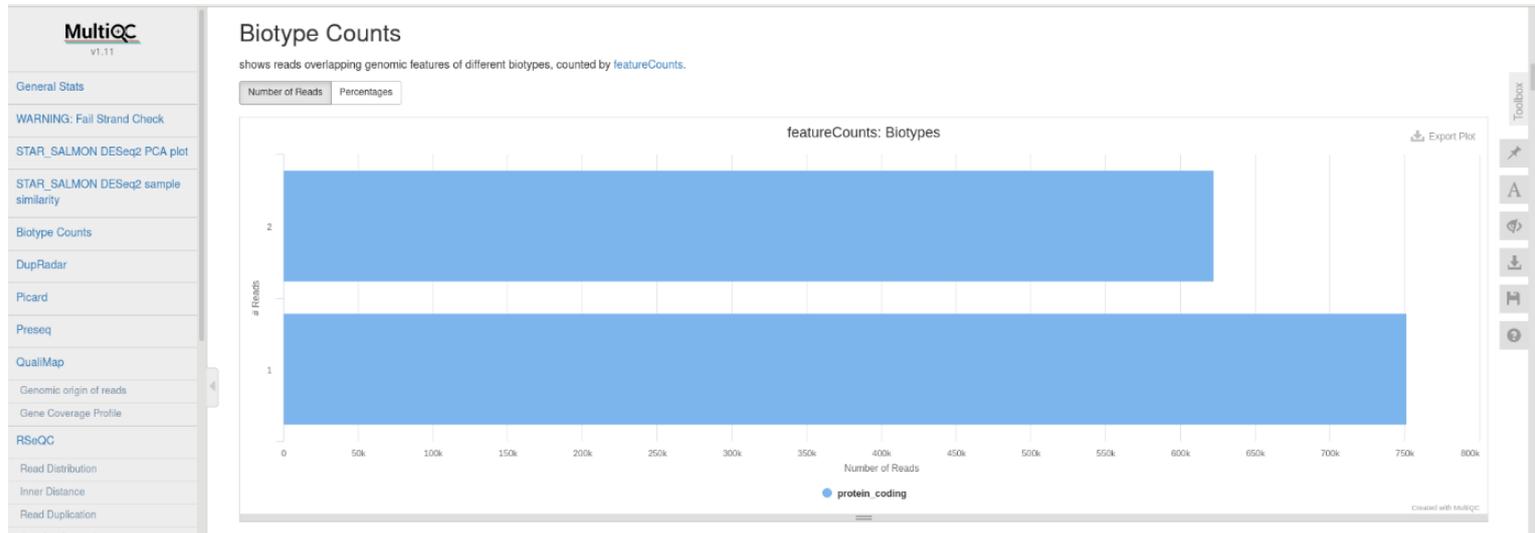
## General Statistics :

Premièrement, en haut de la page on voit les statistiques générales du multiQC sur tous les échantillons, comme le pourcentage de GC, le pourcentage d'alignement, celui des reads mappés, etc

Ensuite, vont suivre plusieurs analyses des échantillons, réalisées avec différents modules, ainsi que les graphiques associés.

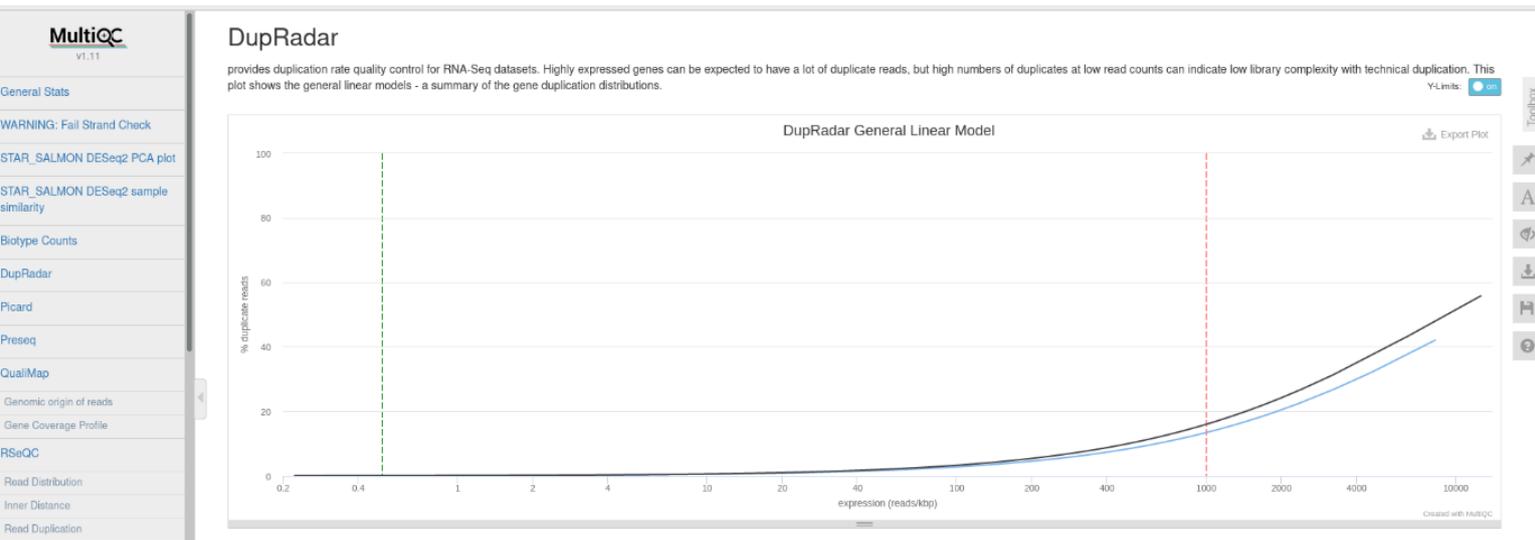
Penchant nous sur plusieurs graphiques, qui me semble intéressant en termes d'analyse car tous n'ont pas retourné de résultats intéressants (probablement lié à ma commande mal exécutée ou alors les échantillons qui en les comparant de certaines manières ne donne pas de résultats). Par exemple, une ACP a été réalisée mais elle n'a donné aucun résultats significatifs ou analysables.

## Biotype count :



Ce graphique montre combien de reads par échantillon sont impliqués ici dans la conception de protéine. On peut voir que pour l'échantillon 1 on a environ 625k reads qui codent dans des protéines et pour l'échantillon 2 environ 750k. Ce graphique a été obtenu à l'aide du module featurecounts. Ici l'échantillon 1 sont les deux fastq qu'on retrouve sur la première ligne du samplesheet.csv et l'échantillon 2 ceux sur la deuxième ligne. Cela correspond à 100% des reads mappés.

**DupRadar :**



DupRadar est issue de la bibliothèque bioconductor de R. DupRadar permet de représenter graphiquement le taux de duplication par rapport à l'expression de chaque reads. Ainsi, un bon échantillon sera un échantillon avec peu de duplication ou un nombre élevé de duplication pour les gènes hautement exprimés. Ici on peut voir que nos deux échantillons sont bons puisque seulement les reads hautement exprimés sont sujets aux duplications.

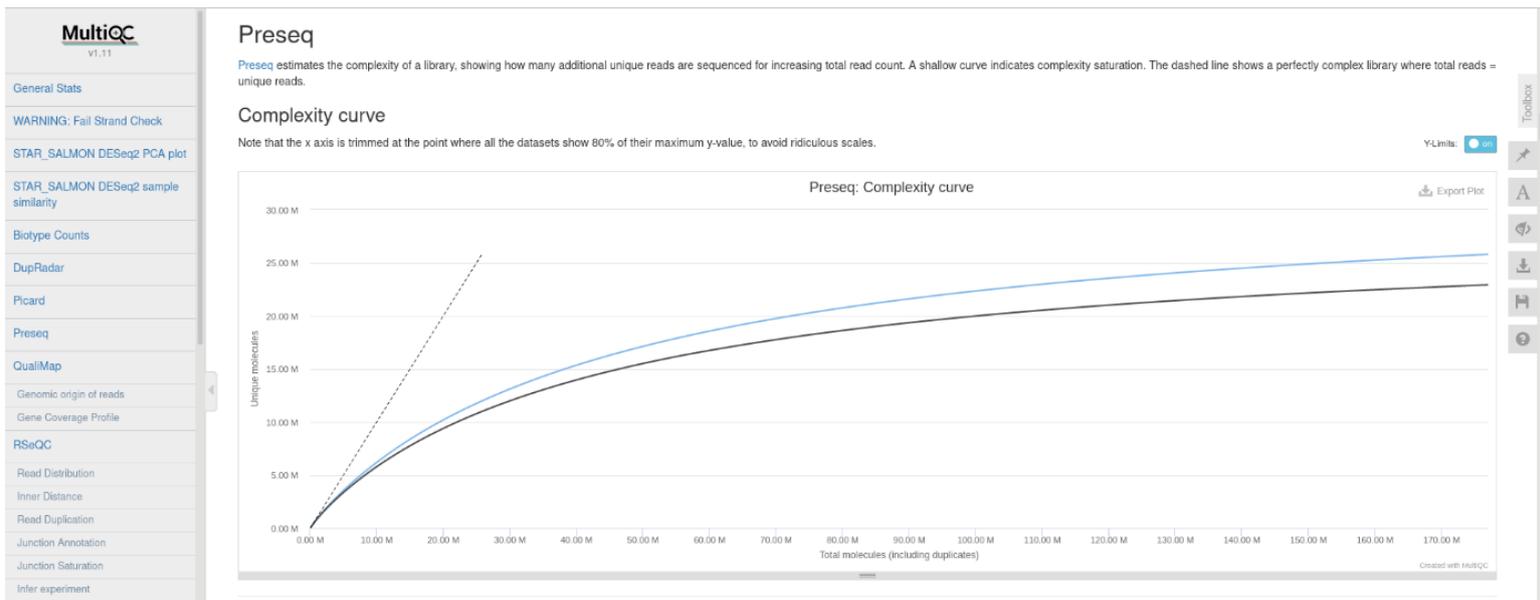
## Picard :

Passons à présent au graphiques Mark Duplicates de Picard qui est un ensemble de commande java pour la manipulation de données de séquençage à haut débit :



Sur le graphique on peut voir le nombre ou pourcentage de reads catégorisé par état de duplication. On peut voir que pour les deux échantillons environ 82 à 83% des reads sont des paires unique et le reste sont des doubles paires non optiques.

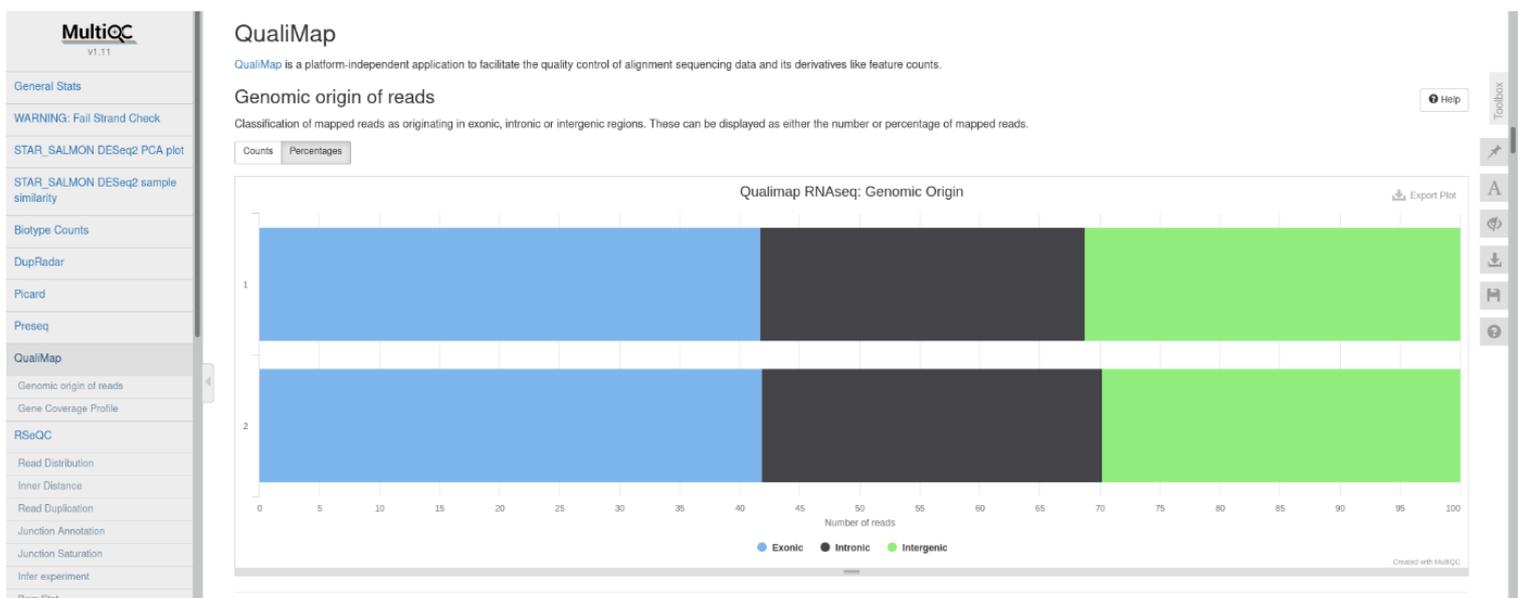
## Graphique PreSeq complexity curve :



Ce graphique montre la complexité d’une bibliothèque estimée par PreSeq et permet de montrer combien de reads supplémentaires sont séquencées pour augmenter le nombre total de reads. La ligne en pointillé représente une bibliothèque parfaitement complexe ou le nombre total de reads est égal au nombre de reads unique.

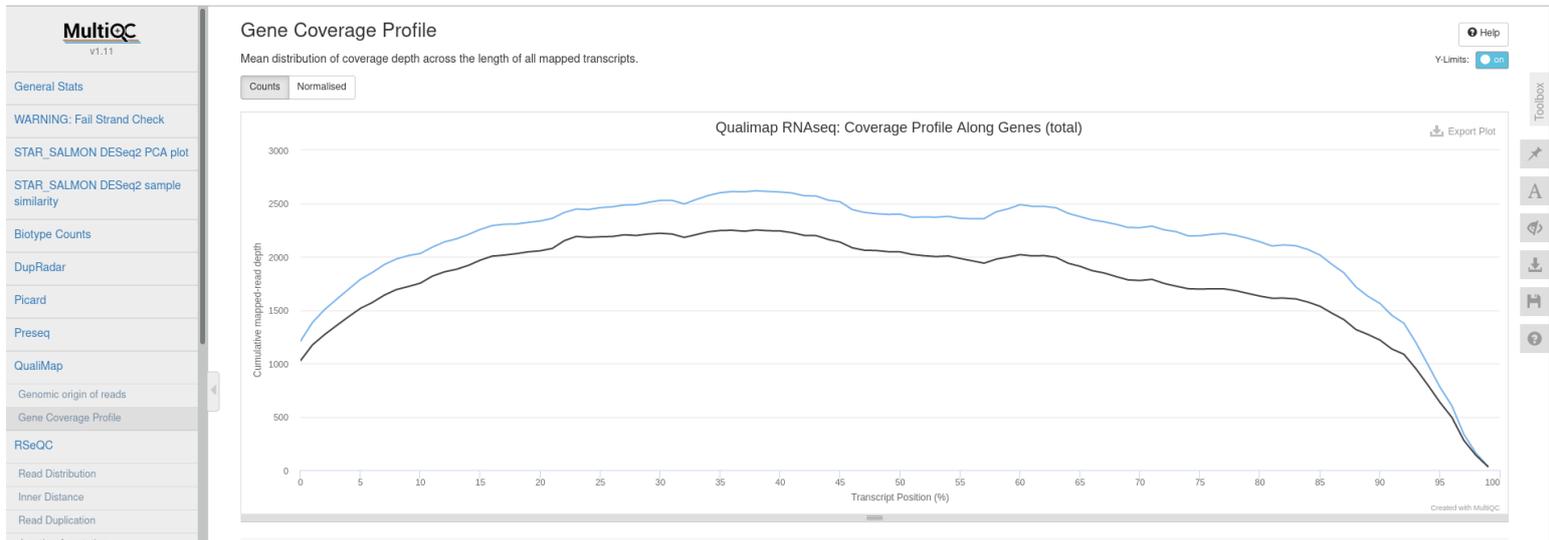
## QualiMap :

A présent intéressons-nous aux graphiques donnés par QualiMap étant une plateforme indépendante permettant de faciliter le contrôle qualité d’un alignement.



Sur ce graphique on peut voir l'origine des reads et s'ils proviennent d'exons, d'introns ou de régions intergéniques. Pour les deux échantillons on obtient globalement les mêmes résultats.

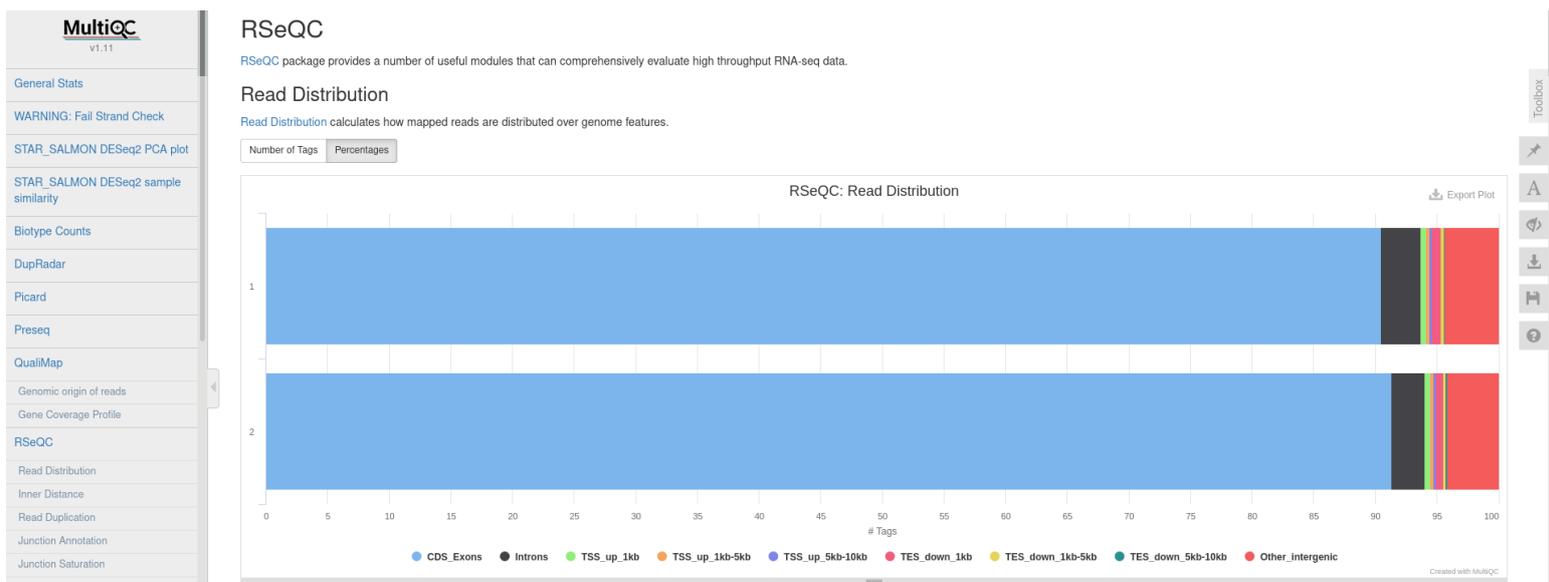
**Gene Coverage Profile :**



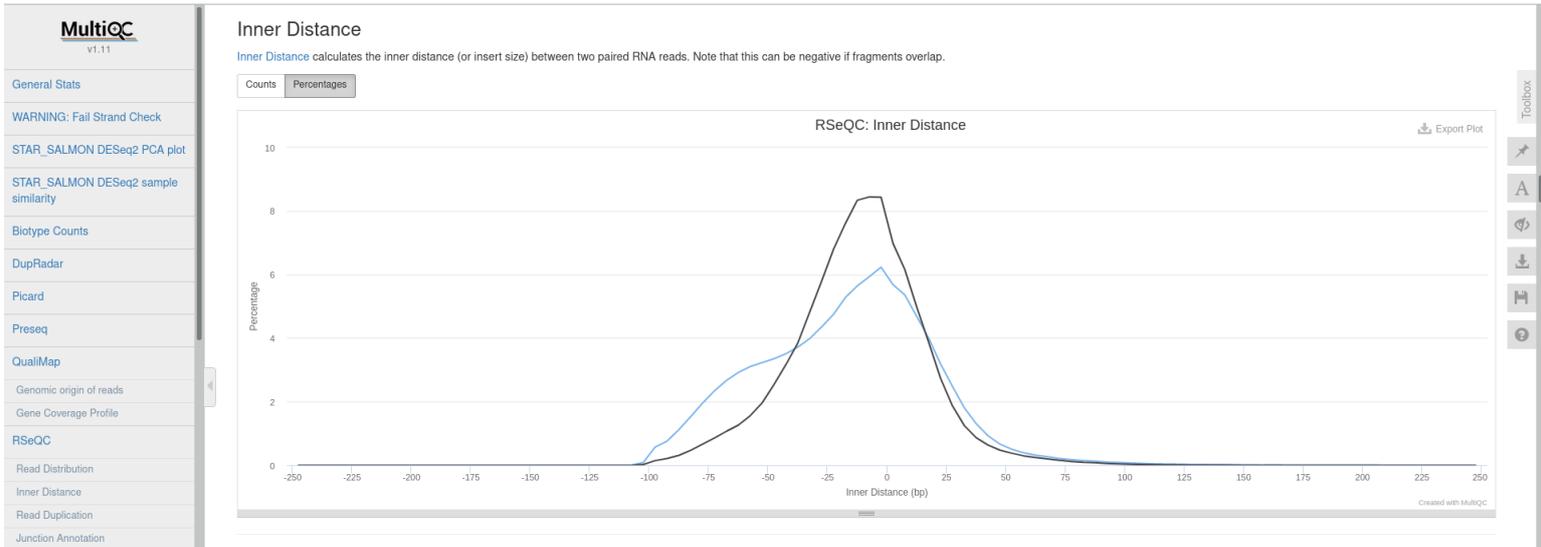
Sur celui-ci on peut voir la distribution moyenne de la profondeur de couverture sur la longueur des reads mappés. Pour les deux échantillons on obtient les mêmes résultats mais pas tout à fait à la même profondeur.

**RSeQC :**

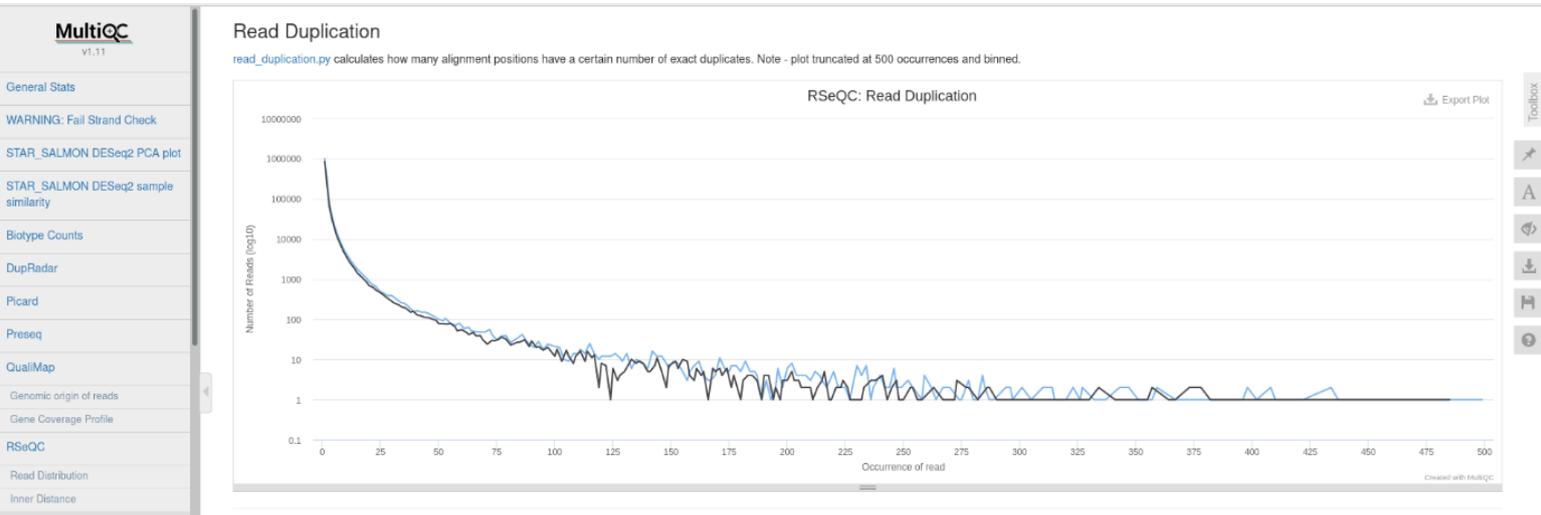
Passons maintenant aux graphiques de RSeQC qui est un package de scripts conçus pour évaluer la qualité des données RNAseq.



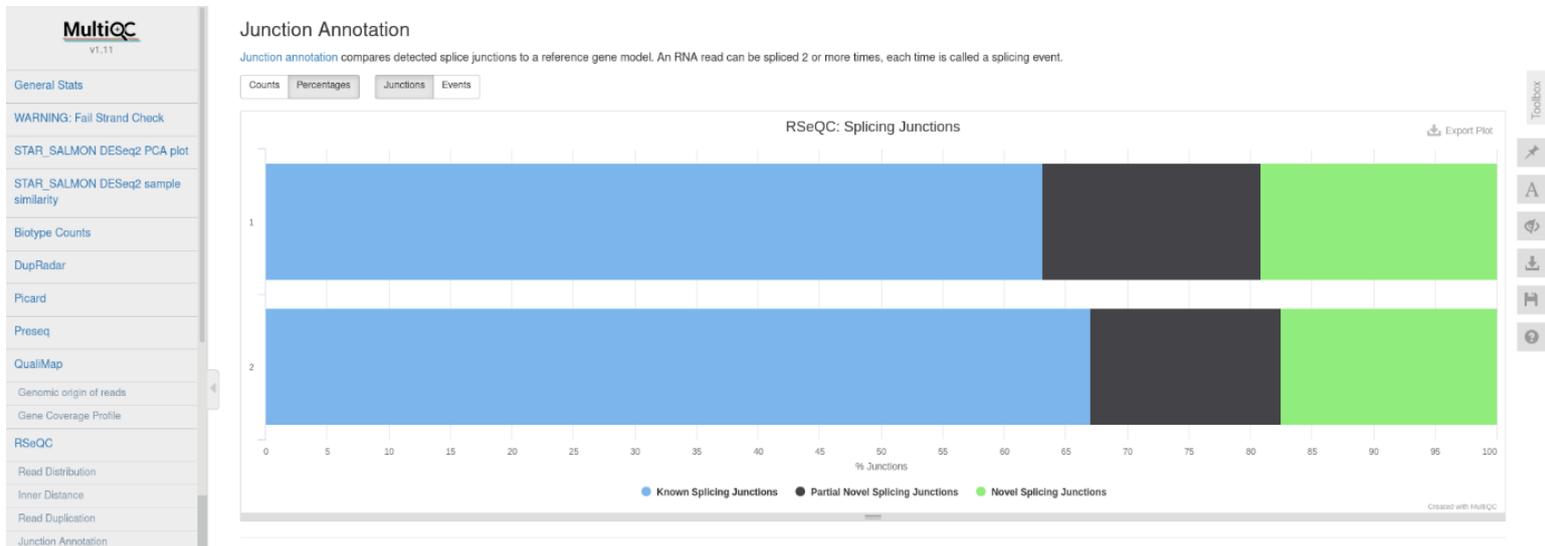
On peut voir sur ce graphique la distribution des reads autrement dit, comment les reads sont mappés sur les différentes parties du génome.



Celui-ci-dessus calcule la distance entre deux paires de reads d'ARN. On peut voir que beaucoup de reads ont très peu de distance entre eux sur les deux échantillons.



Celui-ci calcule le nombre de position d'alignement ont un certain nombre de duplicats. Ce taux de duplication de reads a été calculé avec deux stratégies : une basée sur la séquence (reads avec une séquence identique) et une sur le mapping (reads mappés au même endroits).



Ce graphique montre le pourcentage de jonction d'épissage pour chaque échantillon en comparant les jonctions d'épissage détectées au modèle du gène de référence. Le plus haut pourcentage pour les deux échantillons sont les jonctions d'épissages déjà connus. On peut voir également le pourcentage des nouvelles jonctions d'épissage partielles et des nouvelles jonctions d'épissages.

Il y a ensuite un graphique comptant le nombre de jonction connus observés dans chaque échantillon. Les résultats sont similaires pour les deux échantillons. Sur le graphique les deux courbes forment ce qui s'apparentent à un plateau, cela signifie que la profondeur est suffisante.

Un graphique montre ensuite le pourcentage de reads qui match avec le brin sens ou antisens. A peu près 50% des reads matchent avec chaque brin.

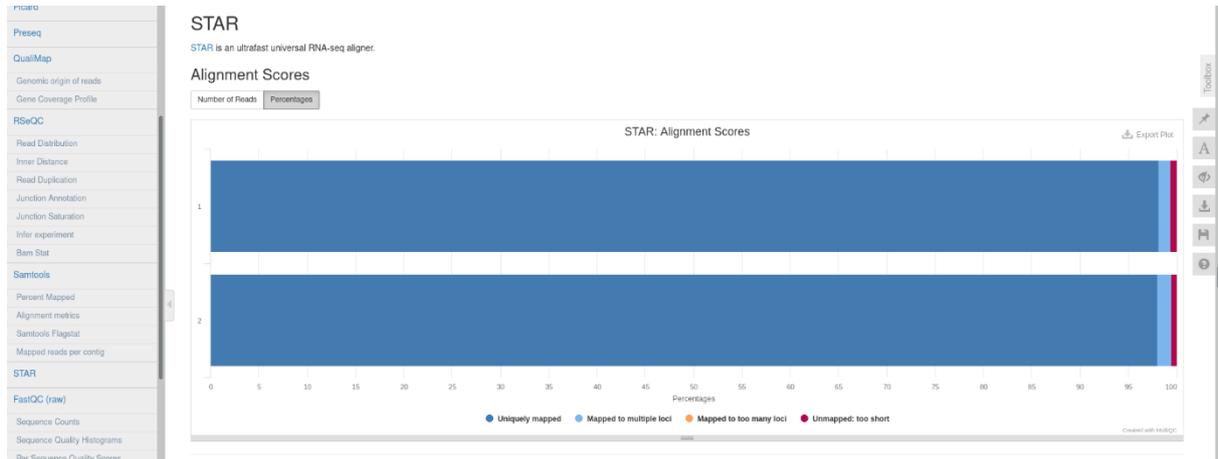
### Samtools :

Le premier graphique de l'outil samtools montre le pourcentage de reads mappés. Pour les deux échantillons, 100% des reads ont été mappés.

Le deuxième graphique montrant les read mappé par contig ne montre aucun résultat et donc n'est pas analysable.

## STAR :

STAR réalise des alignement ultrarapide et universel d'RNAseq.

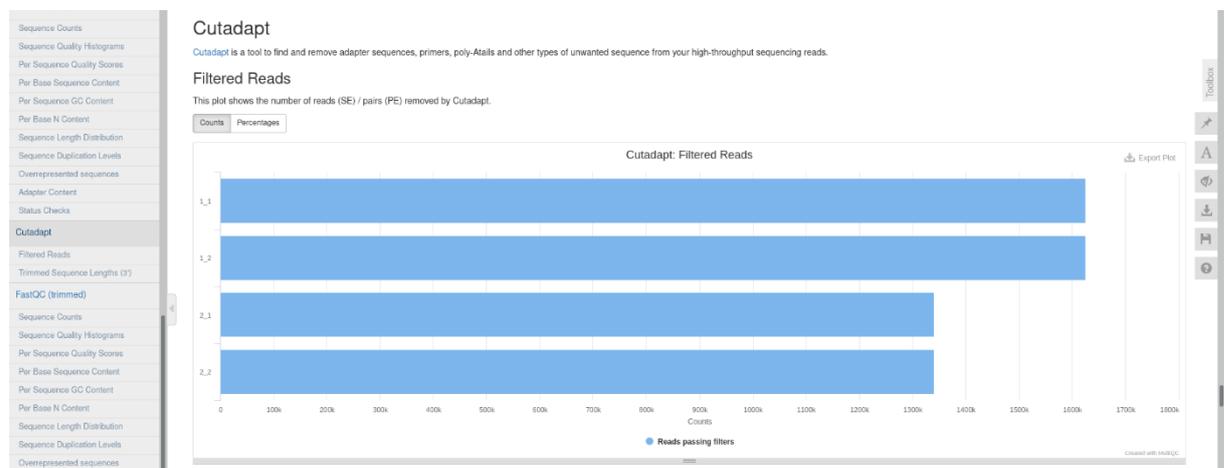


Sur le graphique on peut voir que pour les deux échantillons environ 97% des reads sont uniquement mappés sur un seul loci et que 2% sont marqués sur plusieurs. Environ 1% n'est pas mappé car les reads sont trop petits.

## CUTADAPT :

Cutadapt est un outil pour trouver et supprimer tout ce qui n'est pas génomique comme les queues poly A, les adaptateurs, etc

- Filtered Reads :



Ce graphique montre que tout ce qui n'est pas génomique a été supprimé sur chaque fastq.

- Trimmed Sequence Lengths :

Ce graphique montre le nombre de reads avec la longueur des adaptateurs triés. Pour tout les fastq , la longueur est maximul 5pb et très peu de reads ont des adaptateurs plus grands.

### **FASTQC (RAW/Trimmed) :**

Deux fastQC sont réalisé un avant et un après cutadapt nous allons donc différencier les deux.

- Sequence count :

Sur ce Graphique on peut voir le pourcentage de reads unique et dupliqué pour chaque fastq de la commande. Pour chacun des fastq environ 50% ou un peu plus des reads sont uniques. (Aucune différence entre Raw et Trimmed)

- Sequence Quality Histograms :

Ce graphique montre pour chaque échantillons la moyenne de la valeur de qualité sur une échelle de 100 paires de bases. Pour le FastQC trimmed, on peut voir que la qualité ne baisse pas au niveau des 100bp et qu'elle reste constante. Cependant, cela varie très peu puisque pour les deux FastQC la qualité est très bonne.

- Per Sequence quality scores :

Ce graphique va montrer le nombre de reads en fonction du score de qualité. Pour chaque fastq la qualité est bonne et le nombre de reads avec une bonne qualité est très important alors qu'avec une moins basse qualité il est presque nul. (Aucune différence entre les deux FastQC)

- Per Base sequence content :

Ce graphique montre la proportion de chaque base (ATCG) pour chaque position. Pour les deux FastQC on ne remarque aucune différence la proportion est la même.

- Per sequence GC Content :

Ce graphique montre le nombre de reads avec leur pourcentage GC.

Les quatres fastq ont tous un plus grand nombre de reads avec 40 à 45% de GC. Entre les deux fastQC cela ne varie pas beaucoup sauf la distribution qui est un peu plus étalé pour le trimmed.

- Per Base N Content :

Ce graphique montre le pourcentage de base N à certaines positions. On remarque un très faible pourcentage sur tout les reads avec de légers petits pics à 0.2% en position 2,22et 100 (Aucune différence entre les deux fastQC).

- Sequence length Distribution :

Pour le FastQC raw, la distribution de la longueur est toujours la même puisque les reads font toujours 100pb. En revanche vu que certaines parties des reads ont été supprimé pour le FastQC trimmed, la distribution n'est pas la même. En effet il y a certes un petit nombre mais quelques reads qui font 95pb par exemple.

- Sequence Duplication Levels :

Le niveau de duplication est globalement supérieur à 10 pour tout les fastq (Aucune différence entre les deux fastQC).

- Overrepresented sequences :

Pour les deux FastQC les 4 fastq ont moins de 1% de reads issues de séquences surreprésentées.

- Adapter Content :

Pour chaque FastQC aucune contamination d'adaptateur supérieure à 0.1% n'a été détectés.

- Status Checks :

Pour les deux FastQC l'analyse Per Base sequence content est considéré comme vraiment inhabituelle et la duplication des séquences est considéré comme un peu anormale. En revanche pour le FastQC Trimmed, La longueur des sequences est un peu anormale aussi du fait de la suppression de certaines parties de reads.

**Software versions :**

Cela répertorie toutes les versions des modules et outils utilisés.

**Workflow Summary :**

Cela montre les configurations, les datas utilisées et les répertoires utilisés pour le job.