Projet *NextFlow*

Sébastien Cabanac

M2 Bioinformatique et Biologie des Systèmes

2022-2023

Introduction

Avec le développement des technologies de séquençage et l'augmentation du nombre de données disponibles, le nombre de publications et les résultats liés à la bio-informatique a grandement augmenté au cours de la dernière décennie. Malgré cela, la reproductibilité des résultats est souvent problématique et fastidieuse. Ainsi, la reproduction des résultats peut prendre un temps considérable, parfois plus d'un mois (Garijo *et al.*, 2013, *Quantifying Reproducibility in Computational Biology: The Case of the Tuberculosis Drugome*), et les résultats peuvent différer en fonction de nombreux facteurs, y compris le système d'exploitation utilisé (Di Tommaso *et al.*, 2017, *Nextflow enables reproducible computational workflows*). Pour résoudre ce problème Di Tommosa *et al.* ont mis au point *Nextflow*, un *workflow manager* reposant sur l'utilisation de *containers* permettant d'assurer des résultats consistants grâce à l'utilisation de *pipelines* bio-informatiques. Ces *pipelines* peuvent par ailleurs être importés dans *Nextflow* avec un minimum de recodage.

Dans le cadre de ce projet, le *pipeline rnaseq* (révision 3.4, utilisée avec la version 21.04.1 de *Nextflow*) a été utilisé afin de réaliser le comptage du nombre de *reads* associés à chaque gènes (ou plus globalement *loci*) d'un génome de référence. Les données utilisées ont été obtenues par la technologie *RNAseq* d'Illumina, et correspondent donc à un comptage d'ADNc obtenus par rétrotranscription d'ARN. Elles reflètent donc le niveau d'expression des gènes d'un individu au moment de l'extraction des ARN. Le pipeline *rnaseq* permet donc de réaliser le *trimming* (*i.e.* enlever les sous-séquences correspondant aux adaptateurs nécessaires au séquençage), le *mapping* (alignement des *reads* sur le génome de référence) et le comptage du nombre de *reads* pour chaque gènes/*loci*. Les données de séquençage sont divisées en quatre fichiers au format FASTQ (chaque *reads* est décrit par quatre lignes donnant respectivement son identifiant, sa séquence, un séparateur et la qualité du séquençage via un score Phred). Ces quatre fichiers sont nécessaires car deux conditions sont testées : mesures sur une lignée contrôle (ou « sauvage », ou *Wild Type* (*WT*)), ou mutée (*MT*), et les analyses sont en *pair-end* (il y a donc deux fichiers pour chaque condition : un pour les lectures des ADNc commençant en 5' et un pour les lectures commençant par l'extrémité 3' de ces ADNc). Le type de mutation n'est pas spécifiée mais on peut logiquement attendre des modifications des niveaux d'expression de certains gènes liée à cette mutation.

Ces analyses ont été réalisées sur des données provenant de la tomate (*Solanum lycopersicum*), et le *mapping* a été réalisé sur la version 2.3 du génome de référence de la tomate, assemblé par l'*International Tomato Annotation Group* (*ITAG*) en 2012. L'ensemble du projet s'est limitée au sixième chromosome de cette espèce.

Exercice 1

Le dossier `NEXTFLOW/` présent dans l'espace *work* du compte capucine (`/work/capucine/`) contient les sous-dossiers :

- *data* : contient les fichiers *fastq*, le génome au format FASTA et la *samplesheet* nécessaire au lancement de *rnaseq* (voir Exercice 2).
- *results* : contient les fichiers de résultats des différents outils utilisés par le *pipeline rnaseq*.
- *work* : contient les fichiers nécessaires aux fonctionnements de ces outils. Les sous-dossiers *work* et *results* sont générés automatiquement par le *pipeline rnaseq*.

Exercice 2

Afin d'utiliser le *pipeline rnaseq* sur les données fournies, un script *nextflow_rnaseq.sh* a été créé afin de soumettre un job (*i.e.* une suite de commandes) à l'ordonneur de tâches SLURM qui se chargera de trouver et d'allouer les ressources nécessaires au déroulé de la tâche parmi les ressources disponibles sur le *cluster* de calcul de Genotoul. Le script *nextflow_rnaseq.sh* contient les lignes de commandes spécifiées dans la figure 1.

```
#!/bin/bash
#SBATCH -J SebastienCabanac
#SBATCH -o terminal.out
#SBATCH -e error.out
#SBATCH --mem=6G
#SBATCH -p workq
#SBATCH --time=1-00:00:00

# Load binaries
module load system/singularity-3.7.3
module load bioinfo/Nextflow-v21.04.1

# Command lines
nextflow run nf-core/rnaseq -r 3.4 --input ./data/samplesheet.csv --
fasta ./data/SLycopersicum2.3_Chr6.fasta --gtf
./data/SLycopersicum2.3_Chr6.gtf -profile genotoul
```

Figure 1. Script *nextflow_rnaseq.sh*. Les lignes commençant par `#SBATCH` permettent de spécifier des options relatives au job auprès de SLURM. L'option `-J` permet de donner un nom au job, `-o` et `-e` permettant de donner un nom à l'*output* (sortie standard) et à l'*error* (erreur standard). `--mem` permet de spécifier la mémoire allouer au *job*, `-p` la « file d'attente » dans laquelle sera soumis le *job* (*workq* est prioritaire sur *unlimitq* mais les *jobs* ont une durée maximale réduite). `--time` permet de spécifier la durée maximale du *job*. Le module *singularity-3.7.3* a été chargé pour éviter une erreur où le *pipeline rnaseq* n'arrivait pas à *pull* l'image des *containers singularity*. Pour la ligne de commande `nextflow run nf-core/rnaseq`, l'option `-r` permet de spécifier la révision à utiliser, `--input` est le chemin vers la *samplesheet* au format CSV, `--gtf` le chemin vers le fichier d'annotation au format GTF correspondant au génome de référence utilisé et `--fasta` le chemin vers le fichier FASTA du génome de référence. Pour la *samplesheet*, celle-ci contient le nom des échantillons (colonne *sample*, appelé *control* pour les deux fichiers associés aux lignées *WT* et *mutant* pour les deux fichiers associés

aux mutants) et les deux fichiers d'un même essai (*WT* ou *mutant*) sont mis sur la même ligne (colonnes *fastq_1* et *fastq_2*) et spécifiés comme *unstranded* pour pouvoir faire des analyses *pair-end*.

Une fois le job lancé à l'aide de la commande `sbatch nextflow_rnaseq.sh`, il peut être suivi de plusieurs façons, notamment en suivant l'évolution du fichier *terminal.out* ou en regardant si le *job* a commencé et s'il ne s'est pas arrêté suite à une erreur à l'aide de `squeue -u capucine` (où `-u` permet de renseigner un utilisateur). La commande `seff` permet quant à elle de renseigner l'utilisateur sur les ressources monopolisées par son *job*. La figure 2 montre un exemple de l'utilisation de `seff` sur le job soumis grâce au script *nextflow_rnaseq.sh*. En cas d'erreur lors de l'exécution d'un *pipeline Nextflow*, ce dernier peut renvoyer un message d'erreur. Si cette erreur est corrigée par l'utilisateur, il peut reprendre l'exécution du *pipeline* là où il s'était arrêté à l'aide de la commande `-resume`.

```
Job ID: 37381356
Cluster: genobull
User/Group: capucine/formation
State: RUNNING
Cores: 1
CPU Utilized: 00:00:00
CPU Efficiency: 0.00% of 00:00:10 core-walltime
Job Wall-clock time: 00:00:10
Memory Utilized: 0.00 MB (estimated maximum)
Memory Efficiency: 0.00% of 6.00 GB (6.00 GB/node)
WARNING: Efficiency statistics may be misleading for RUNNING jobs.
```

Figure 2. Sortie standard de la commande `seff 37381356`, avec 37381356 l'identifiant du *job* soumis via le script *nextflow_rnaseq.sh* (voir figure 1). La ligne *state* renseigne sur l'état du job (*PENDING* s'il est en attente de lancement, *RUNNING* s'il est en cours, *FAILED* s'il s'est arrêté suite à une erreur ou qu'il a été interrompu, *COMPLETED* s'il s'est terminé normalement). *Cores* indique le nombre de cœurs utilisé par le *job*, alors que *CPU Utilized* et *CPU Efficiency* renseignent respectivement sur le temps pendant lequel le *CPU* a travaillé sur le job et ce même temps divisé par le nombre de cœurs utilisé. *Job Wall-clock* correspond à la durée « réel » du *job*, de son lancement jusqu'à son arrêt. *Memory Utilized* et *Memory Efficiency* renseignent quant à elles respectivement sur la mémoire utilisée et la mémoire utilisée par rapport à la mémoire demandée pour ce *job*.

```
-[nf-core/rnaseq] Pipeline completed successfully-
Completed at: 30-Sep-2022 11:27:19
Duration    : 16m 10s
CPU hours   : 2.9
Succeeded   : 79
```

Figure 2bis. Capture d'écran des cinq dernières lignes de *terminal.out*, fichier contenant la sortie standard du *job* lancé via le script *nextflow_rnaseq.sh*. Sur cette sortie figurent l'état de la complétion du job (*completed successfully*), la date et l'heure à laquelle le job s'est arrêté (30 septembre 2022 à 11 heures 27), la durée du job (16 minutes et 10 secondes ; cette durée élevée est liée à l'utilisation du *module load singularity-3.7.3* qui a entraîné le téléchargement en local dans le répertoire `/work/capucine/NEXTFLOW/work/singularity/` des images des *containers* nécessaires au *pipeline rnaseq*, permettant de contourner l'erreur liée au *pull* de *singularity*), le *CPU hours* et, bien que non-spécifié, *succeeded* correspond *probablement* aux nombre de tâches correctement effectuées au cours de l'exécution du *pipeline*.

Exercice 3

Deux dossiers sont générés au cours de l'exécution du *job* : *results* et *work*. Le dossier *work* contient un ensemble de sous-dossiers nommés par deux caractères alphanumériques contenant les différents fichiers nécessaires aux fonctionnements des différents outils appelés par le pipeline *rnaseq*, ainsi qu'un dossier *tmp* contenant des fichiers résumant le fonctionnement du pipeline (fichier au format YAML) et les versions de Python et de YAML utilisées. Dans le cas du projet, les différentes images des *containers singularity* ont également été enregistrés dans le dossier *singularity*. Le dossier *results* contient un ensemble de sous-dossiers relatifs aux résultats des différents outils utilisés par le *pipeline* ainsi qu'un dossier *pipeline_info* contenant des informations sur le déroulé du *pipeline*. Ce sous-dossier comprend notamment un fichier *software_versions.yml* listant les versions des différents outils utilisées et deux fichiers au format YML *execution_report* et *execution_timeline* renseignant respectivement sur les ressources utilisées (figure 3) par chaque outil et les temps/mémoire (figure 4) utilisée pour chaque outil.

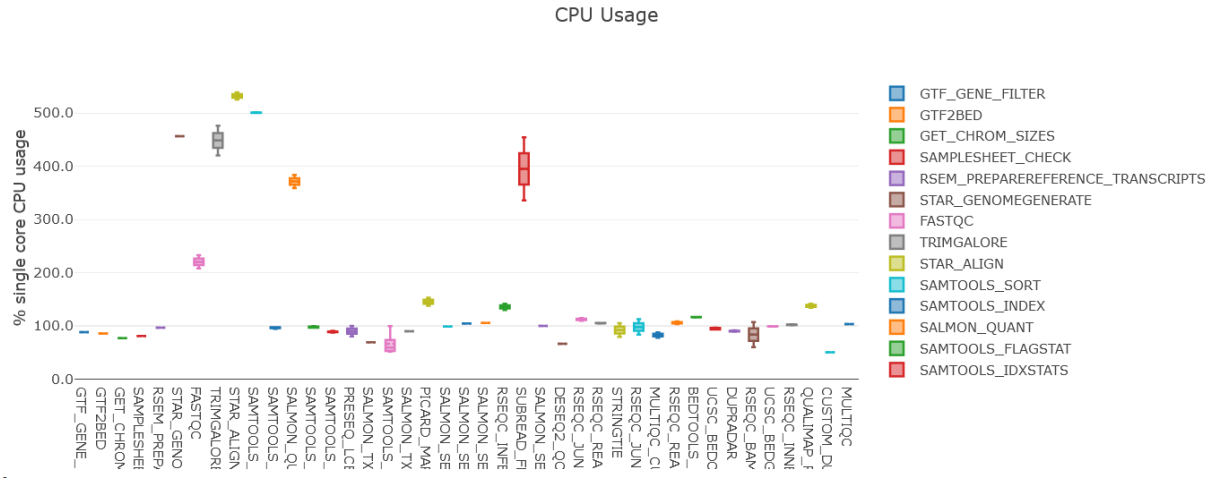

Figure 1. Exemple d'informations fournies par le fichier *execution_report* ; ici le *CPU usage* pour chacun des logiciels utilisés par le *pipeline rnaseq*.

Figure 2. Informations fournies par le fichier *execution_timeline* pour les dix premières opérations effectuées. Représente le temps de calcul consacrée à l'opération/usage de la *RAM*.

Les sous-dossiers contenant les résultats des outils utilisées sont :

- *genome* : contient le fichier d'annotation au format GTF, un fichier concernant la taille des différents *scaffolds* du génome (ici, seulement le chromosome six), le génome indexé (.FAI) et le fichier BED (figure 5).

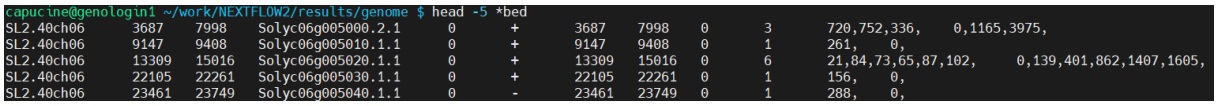

Figure 3. Fichier BED généré au cours de l'exécution du *pipeline*. Ce fichier contient différentes colonnes séparées par des tabulations, renseignant successivement sur le nom du chromosome, les positions de départ et de fin d'un *loci* considéré, le nom du *loci*, un score (si un score est attribué), l'orientation du brin, les positions à partir desquelles le *loci* sera représentée de manière plus visible lors de représentations graphiques (généralement les codons *start* et *stop* d'un gène, dans notre cas identique aux positions de départ et de fin du *loci*), le nombre de *blocks* (*i.e.* d'exons) sur le *loci* et leurs positions de départ de fin séparées par des virgules.

- *trimgalore* : contient un rapport concernant le *trimming* des séquences, autrement dit la suppression des sous-séquences des *reads* correspondant aux adaptateurs nécessaires aux séquençages. Par exemple, pour un des fichiers *fastq* contrôle, on peut trouver (en autres) comme renseignements :
    - Type de *trimming* : *paired-end*
    - Versions de *trimgalore* et *cutadapt* : 0.6.7 et 3.4
    - Le seuil du score Phred (*i.e.* qualité du séquençage) : 20
    - L'adaptateur détecté automatiquement : AGATCGGAAGAGC
    - L'erreur autorisée sur la séquence de l'adaptateur : 0.1
- *fastqc* : contient des rapports permettant de visualiser rapidement la qualité du séquençage (figure 7). On remarque que la qualité du séquençage (*Phred score*) baisse en fin de *read*, que la distribution des taux de A, T, C et G varient anormalement et que certaines séquences sont lues un nombre anormalement élevé de fois. Ces trois phénomènes sont liés à la présence des adaptateurs sur les *reads*, même si la chute de qualité en fin de *reads* est aussi liée à des erreurs systématiques techniques (affaiblissement des fluorophores, brins « bloqués », désynchronisation des brins au sein d'un *cluster*, …). Des rapports similaires sont fournis après le *trimming* des séquences dans le sous-dossier *trimgalore* (figure 8).

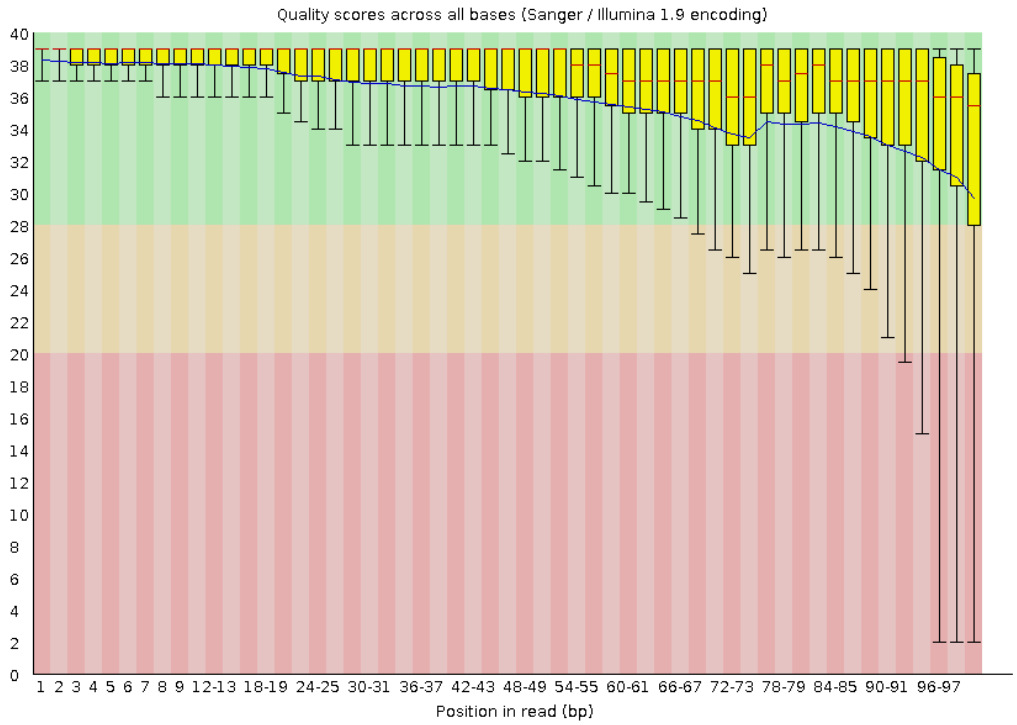

Figure 4. Exemple de graphiques fournis par le rapport *fastqc*. Ce *boxplot* représente l'évolution de la qualité du séquençage (*Phred score*) pour un nucléotide en fonction de la position de ce nucléotide sur le *reads*. On observe une baisse de la qualité en fin de lecture liée (entre autres) à la présence des adaptateurs.

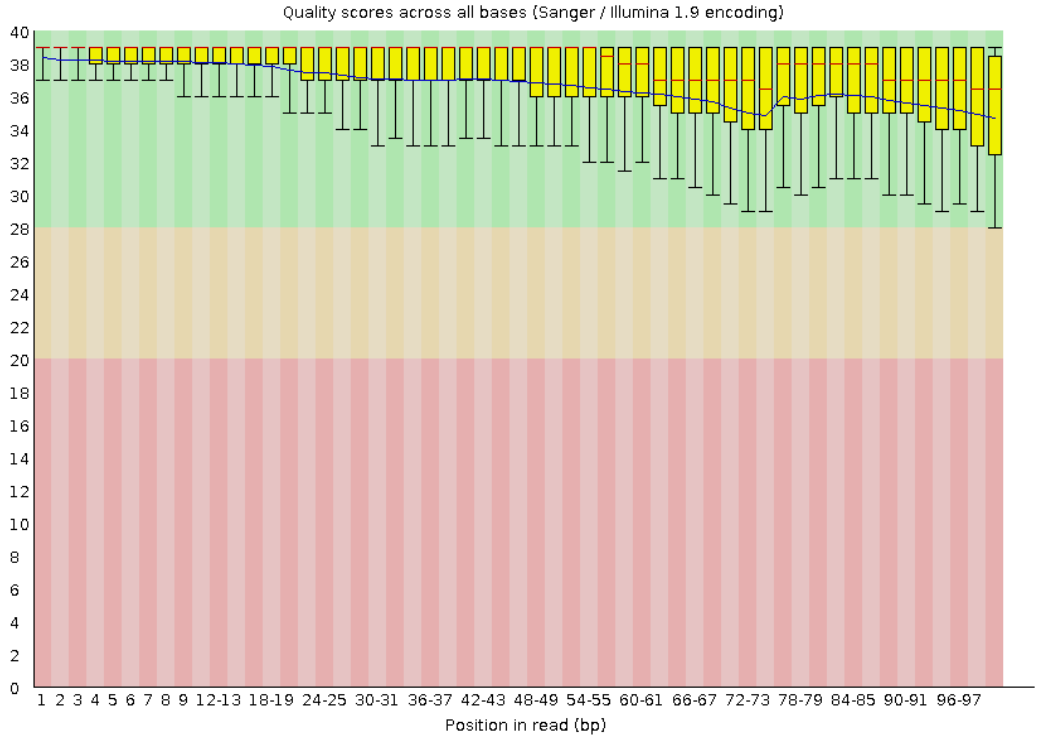

Figure 5. *Boxplot* obtenu sur les mêmes données et de la même manière que pour la figure 4, après un *trimming*. On observe une baisse plus modérée de la qualité en fin de *reads*.

- *star_salmon* : contient les fichiers et résultats des différents logiciels utilisés pour réaliser l'alignement et la quantification du nombre de *reads* par gènes. Parmi les différents fichiers présents, un rapport réalisé à l'aide du logiciel *qualimap* permet de visualiser rapidement la qualité du *mapping* (figure 6).

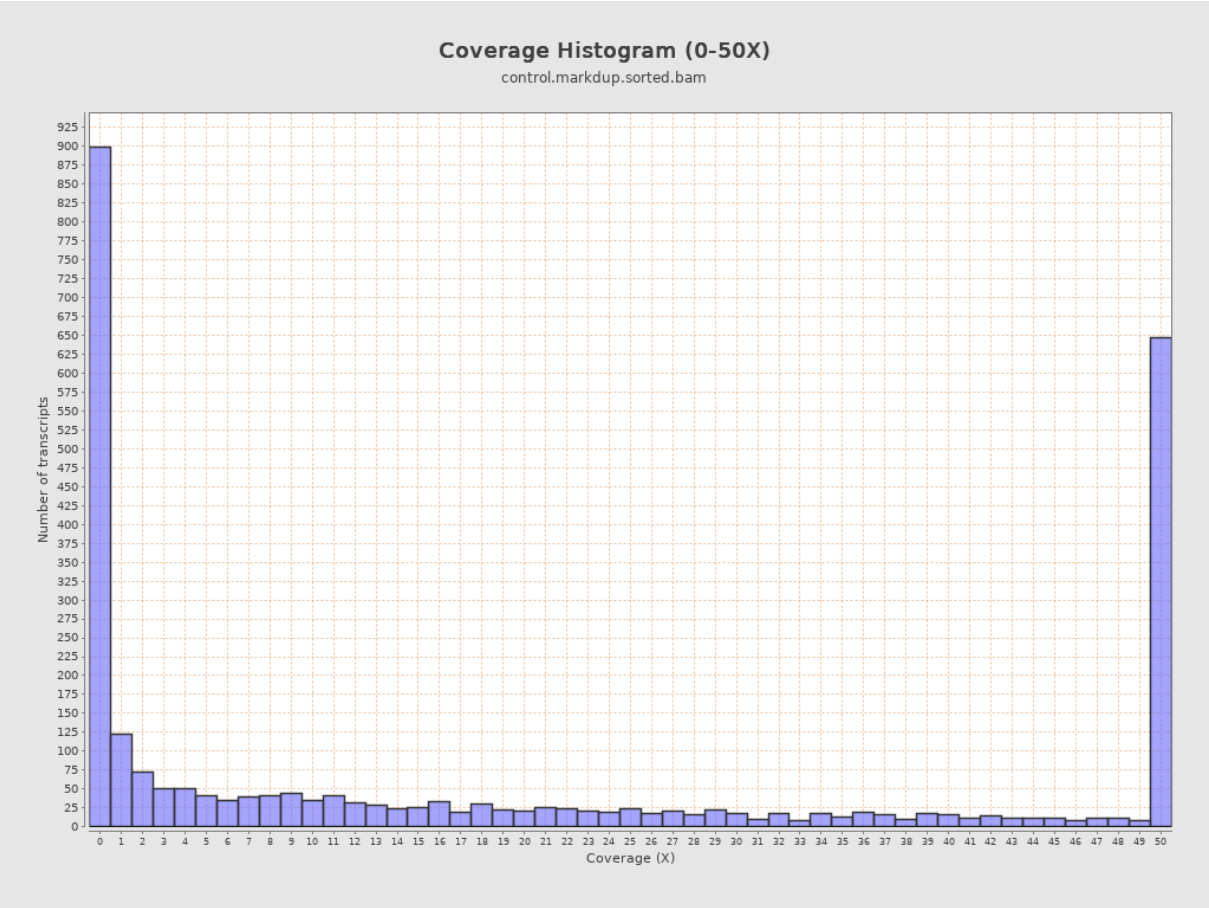

Figure 6. Exemple de graphiques fournis dans le rapport de *qualimap*. Cet histogramme représente le nombre de transcrits en fonction de la couverture (*i.e.* le nombre de fois où un transcrit a été séquencé). Cet exemple concerne uniquement les données sur la lignée *WT*.

- *multiqc* : contient un sous-dossier résumant les résultats des différents outils utilisées. Ces fichiers sont nommés suivants le format « *multiqc_nom_de_l_outil.txt* » et comprennent par exemple :
    - *cutadapt* : pour chaque fichier *fastq*, le nombre de *reads* traité, le nombre de *reads* où un adaptateur a été trouvé, le nombre de *reads* filtrés et écrits en sortie, le nombre de paire de bases traité, le nombre de paire de bases filtrées car leur qualité de séquençage était inférieure au seuil, et le pourcentage de séquences trimmées.
    - *fastqc* : pour chaque fichier *fastq*, donne différents renseignements sur les séquences (longueur, qualité par base, qualité par séquence, pourcentage de GC, …).
    - *fastqc_1* : *idem* que le fichier précédent, mais après *trimming*.
    - *picard_dups* : pour chaque condition (*WT* ou *mutant*) renseigne (entre autres) sur le nombre de *reads* non-appairés examinés, le nombre de *read* appairés examinés, le nombre de *reads* non-mappés et le pourcentage de duplication.

  L'ensemble de ces fichiers est synthétisé dans un résumé global et visuel au format HTML (*multiqc_report.html*). Après un tableau résumant les statistiques générales de l'analyse, le rapport comporte un ensemble de graphiques partiellement présentés dans les figures 7 à 19.

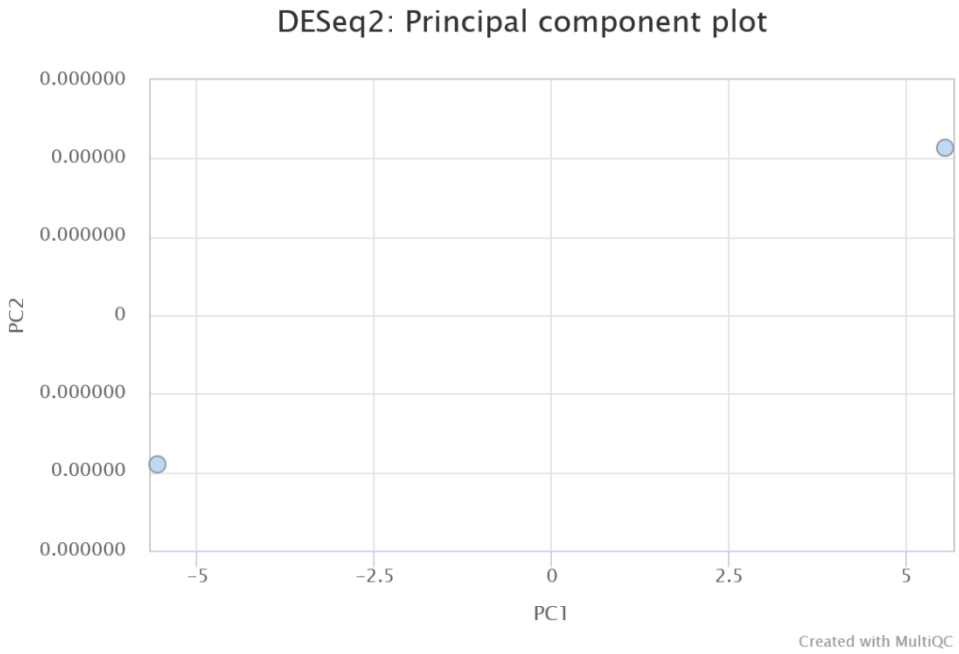

Figure 7. Analyse en composantes principales réalisée par *DESeq2* représentant les premières et secondes composantes principales (*PC1* et *PC2*) obtenues à l'aide du nombre normalisé de *reads* pour les 500 gènes les plus variables. Le point en bas à gauche correspond aux *controls* et le point en haut à droite représente les *mutants*. On observe une séparation nette des deux groupes suggérant que la mutation présente chez les lignées de tomates mutées a entraîné de fortes modifications dans l'expression de nombreux gènes. Les résultats précis de DESeq2, consultable dans le sous-dossier *star_salmon*, spécifient par ailleurs que la *PC1* représente plus de 99% de la variance. L'écart entre les deux groupes est donc très important.

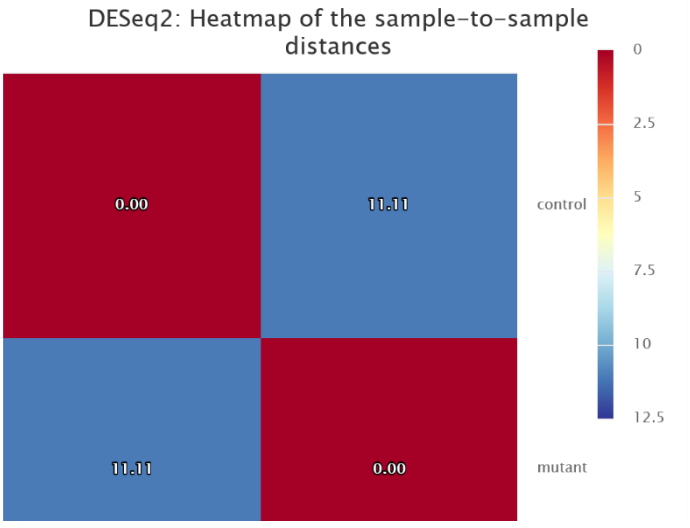

Figure 8. *Heatmap* représentant les distances euclidiennes moyennes d'échantillon-à-échantillon entre les lignées *mutants* et les *controls*. Une distance nulle est observée (et attendue) à la diagonale (comparaison d'un groupe envers lui-même) et une distance euclidienne moyenne de 11,11 est observée entre les deux groupes, suggérant à nouveau une différence importante entre les groupes.

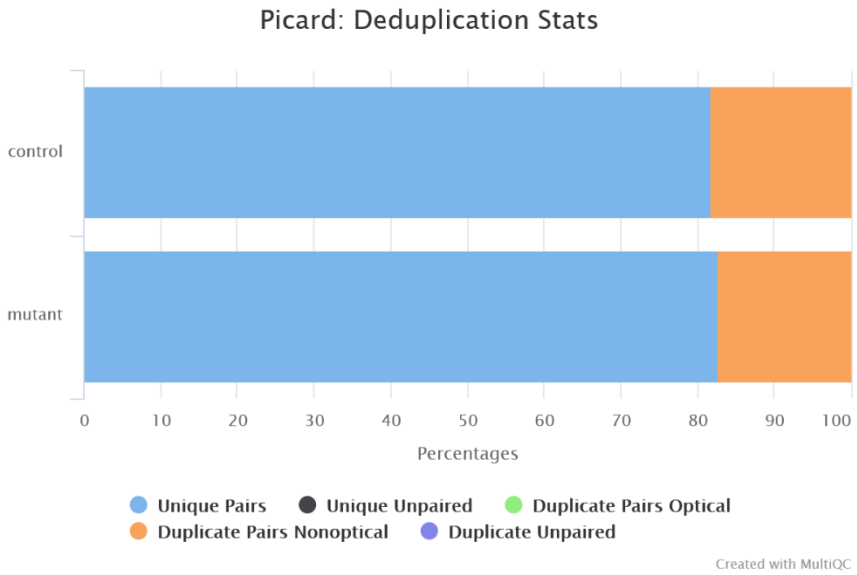

Figure 9. Barres empilées représentant le compte (/pourcentage) du nombre d'artefacts de duplications par l'outil *Picard*. Permet de visualiser le nombre de paires de *reads* uniques, de *reads* uniques *unpaired* (impossibilité de les reliés à une autre séquence en *paired-end*), et les différents artefacts de duplication : *duplicate pairs optical* montre que plusieurs clusters sur la puce de séquençage correspondent en fait à une même séquence, *duplicate pairs nonoptical* représente les erreurs de duplication d'ADN au cours des cycles de PCR et *duplicate unpaired* correspond à ces même artefacts mais non-appairés.

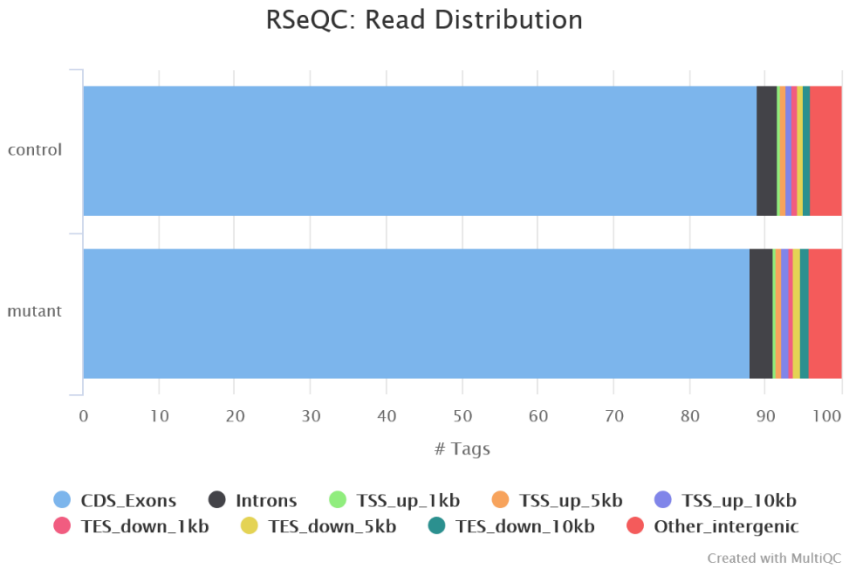

Figure 10. Barres empilées représentant le nombre de *reads* en fonction du type de séquences sur lequel ils ont été mappés, pour les *WT* et les *mutants*. *TSS_up_nkb* signifie *n* kb avant le *Transcription Start Site* d'un gène, et *TES_down_nkb* signifie *n* kb après le *Transcription Ending Site*.

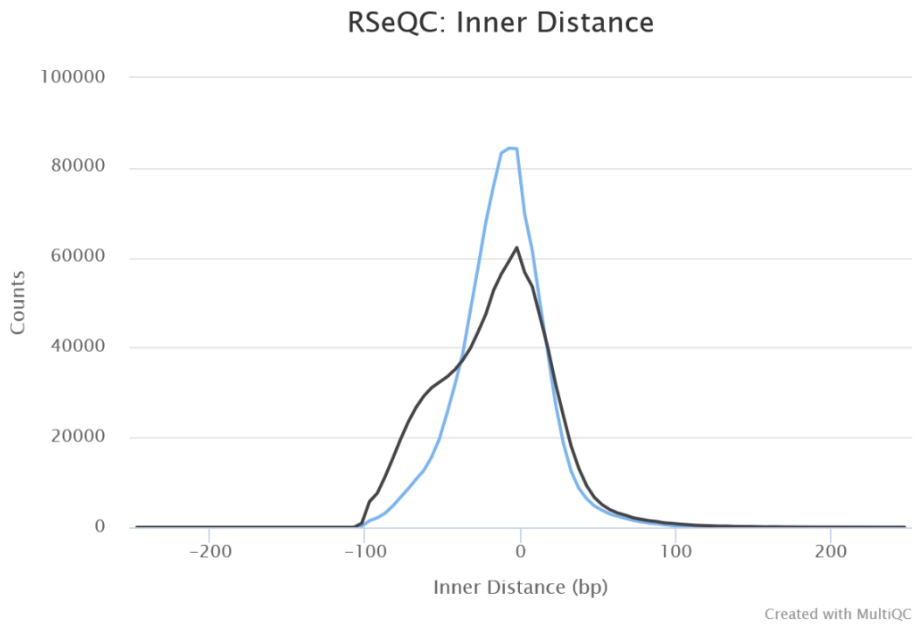

Figure 11. Les courbes ci-dessus représente le nombre de *reads* en fonction de la distance en paire de base entre les *reads* correspondant à une même paire (*pair-end*). La courbe bleue correspond aux contrôles et la courbe noire correspond aux mutants. On observe un nouveau pic chez les mutants, correspond à une *inner distance* de -62 paires de bases. Les *reads* associés à ce pic se chevauchent donc d'environ 62 paires de bases, et étant donné la longueur d'un *read* (environ 100 paires de bases), on peut supposer que ces *reads* correspondent à des petits ARN d'une quarantaine de base, pouvant éventuellement être impliqués dans la régulation épigénétique négative (système DICER) de l'expression de gènes.

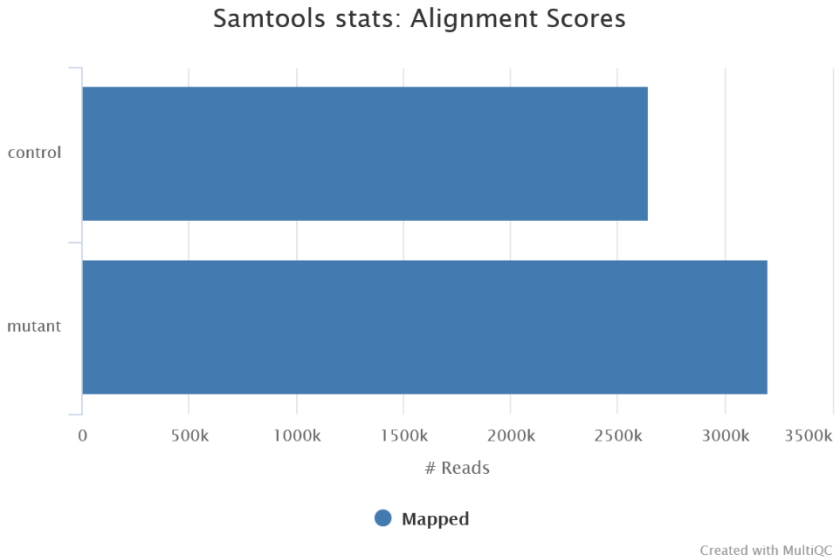

Figure 12. Barres empilées représentant le nombre de *reads* ayant pu être mappés sur le génome de référence pour chacun des groupes. Dans les deux cas, les nombres ci-dessus correspondent à 100% des *reads*.

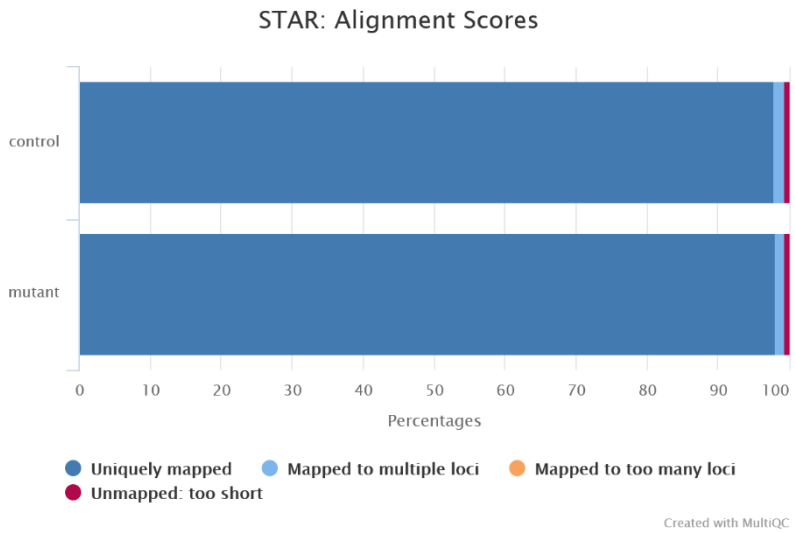

Figure 13. Barres empilées représentant le pourcentage de *reads* en fonction du nombre de fois où ils ont été mappés sur le génome de référence, pour chacun des groupes (*control* ou *mutant*). Une séquence d'ADNc correspond normalement à un *loci* précis et ne doit donc correspondre qu'à un seul *loci* (*uniquely mapped*) sur le génome de référence. Les *controls* et les *mutants* ont tous les deux un pourcentage de *reads* mappés une seule fois supérieur ou égal à 98%.

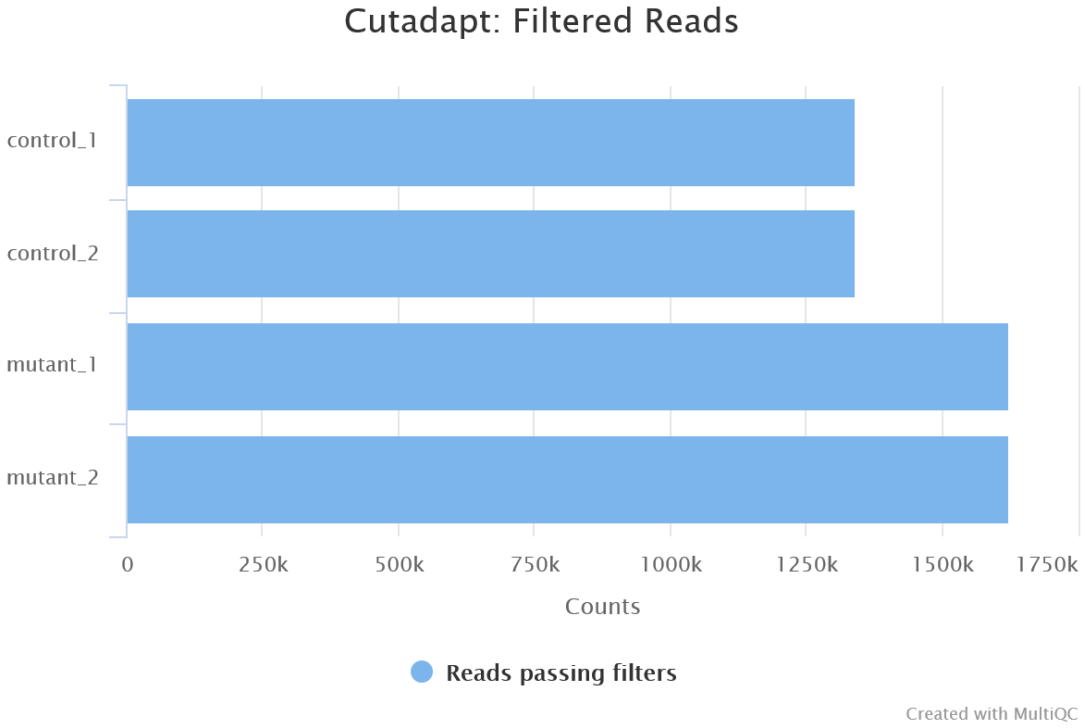

Figure 14. Nombre de *reads* ayant une qualité de séquençage supérieure au seuil de *cutdapat* (20) pour chacun des fichiers *fastq* initiaux. Dans chacun des cas, ce nombre représente quasiment 100% des *reads* (*cutadapt* arrondi à 100%, mais un faible nombre de séquences sont associées à une qualité inférieure à 20, voir figure 18).

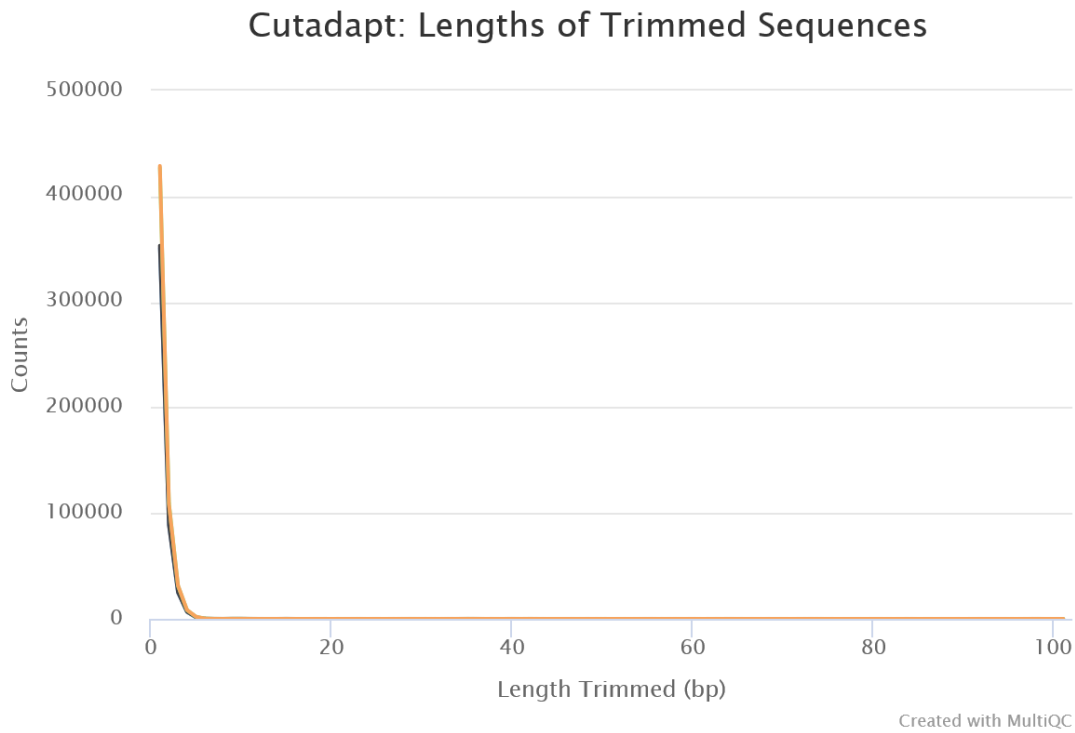

Figure 15. Courbe représentant le nombre de séquences trimmées en fonction de leurs longueurs. Comme mentionné précédemment, les séquences coupées des *reads* correspondent à celles d'un adapteur. Celui-ci ne mesurant que 13 paires de bases, il est normal de ne trouver aucune séquence avec une longueur supérieure à 13 paires de bases sur ce graphique.

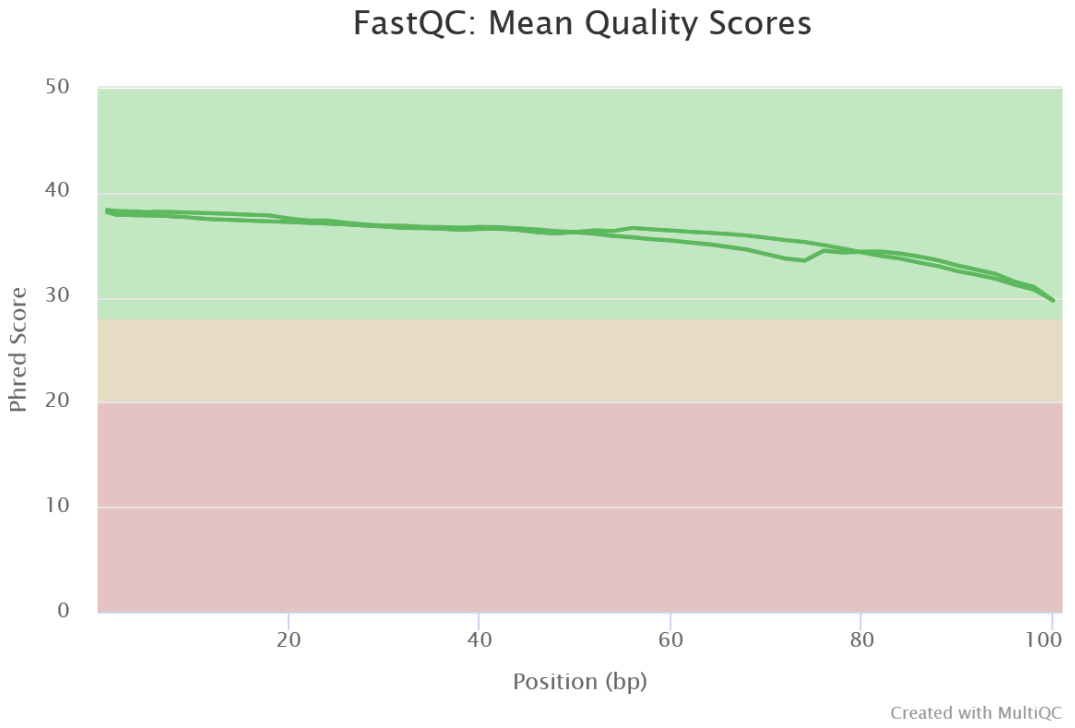

Figure 16. Courbes représentant la qualité moyenne du séquençage (*Phred score*) des *reads* des *controls* et des *mutants* avant le *trimming* en fonction de la position sur le *read*. On observe une baisse de la qualité en fin de lecture, liée aux phénomènes décrits précédemment et qui comprennent la présence des adaptateurs.

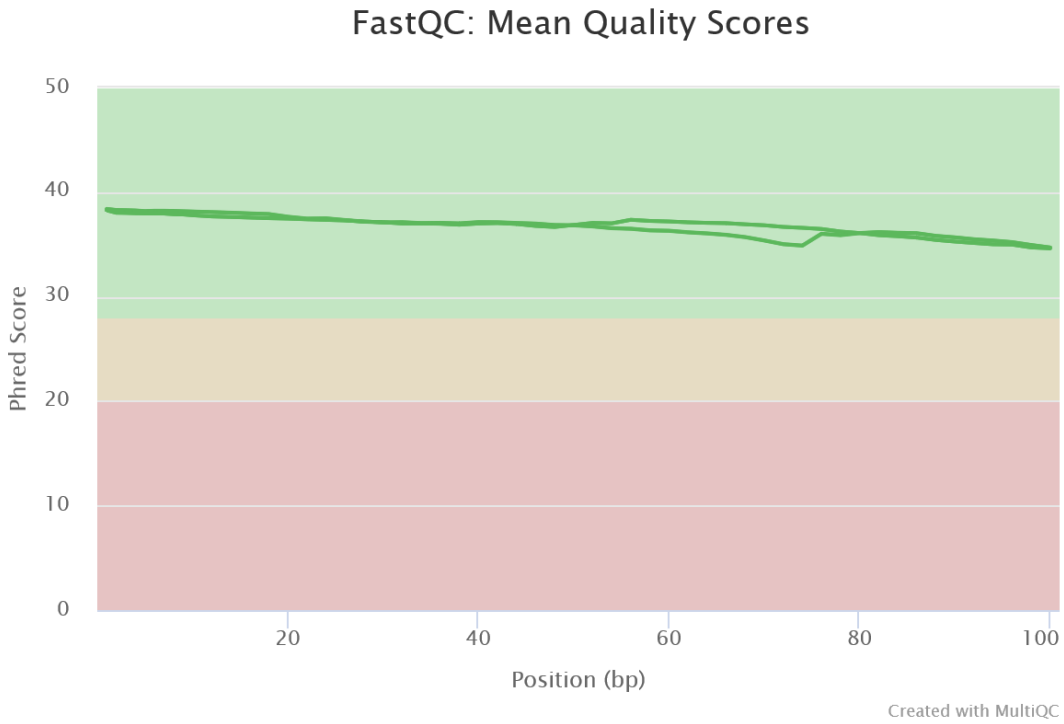

Figure 17. Courbes représentant la qualité moyenne du séquençage (*Phred score*) des *reads* des *controls* et des *mutants* après le *trimming*. Comparé à la figure 16, on observe une baisse plus modérée de la qualité en fin de lecture, qui reste toujours supérieure à 30.

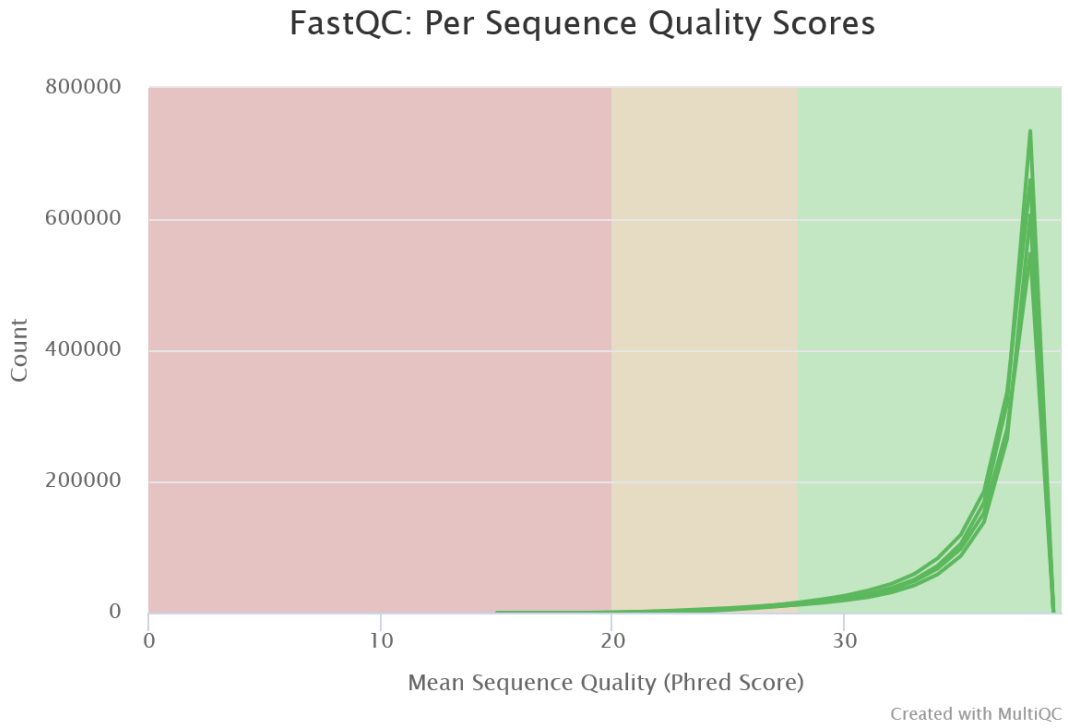

Figure 18. Compte du nombre de *reads* en fonction de leur qualité moyenne (moyenne des *Phred scores* pour chaque nucléotide qui constitue le *read*). On remarque que la qualité du séquençage est globalement élevée, et très peu de séquences ont une qualité inférieure à 20.

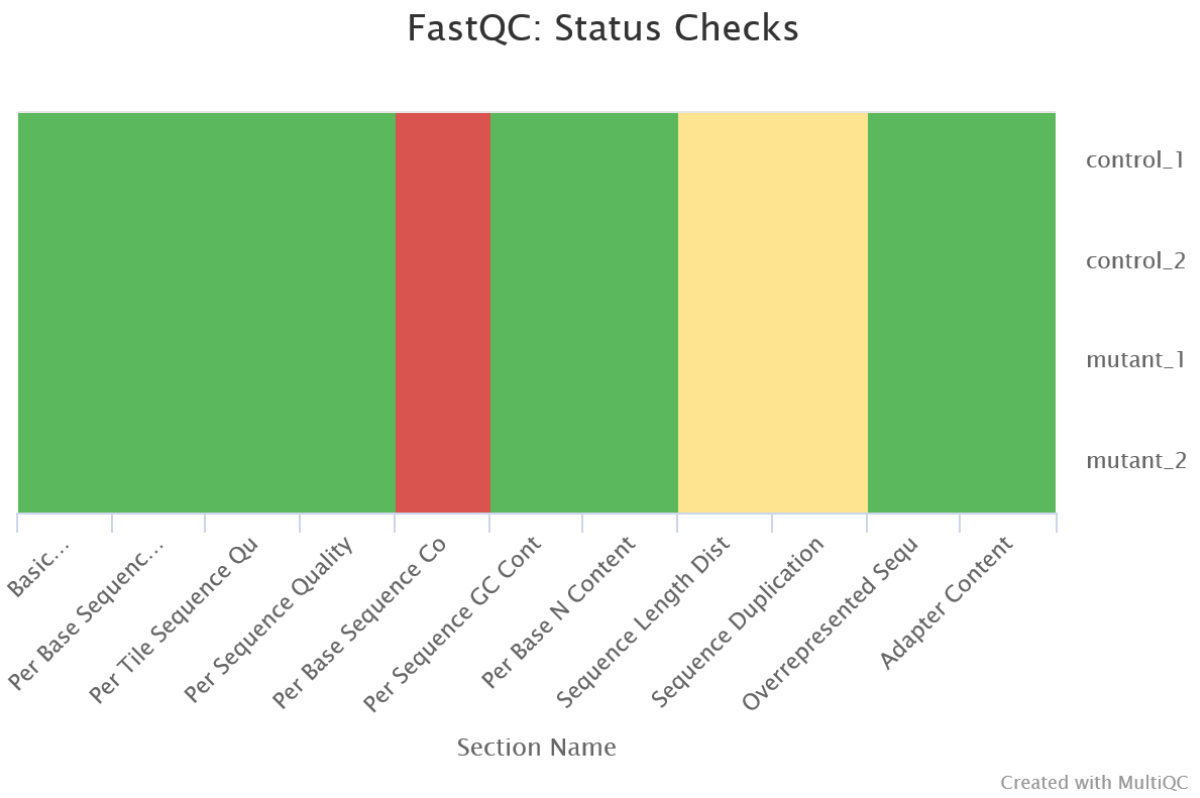

Figure 19. Compartimentage montrant le *status* pour chacune des sections de *fastqc*, indiquant si les résultats semblent normaux (verts), moyennement normaux (jaunes) ou anormaux (rouges). Le comptage du nombre de séquences par base, la distribution des longueurs des séquences et la duplication des séquences semblent moyennement normaux, voire anormaux. Ces résultats peuvent s'expliquer par plusieurs phénomènes biologiques. Il est par exemple possible que les ARNr (ou d'autres petits ARN), de petites tailles et présent en grande quantité, n'ait pas été filtrés, faussant « artificiellement » les sections *Per Base Sequence Content* (distribution des quatre types d'acides nucléique au sein d'une séquence, ici il y aurait un sur-représentation des ARNr), *Sequence Length Dist* (si les petits ARN n'ont pas été filtré, la distribution des longueurs de séquences devraient passer d'une distribution normale à polynomiale), et *Sequence Duplication*.

Conclusion

Grâce à l'outil *Nextflow*, le comptage du nombre de reads associé à chaque loci devrait permettre d'identifier les gènes dont l'expression a été modifié par la (ou les) mutation(s) chez *S. lycopersicum*. L'observation du rapport global obtenu via multiqc permet déjà de visualiser clairement une séparation des WT et des MT, mais cette séparation doit être confirmée par des méthodes statistiques reposant sur la comparaison du nombre de reads entre les deux conditions (disponible dans `/work/capucine/NEXTFLOW/results/star_salmon/salmon.merged.gene_counts.tsv`). On peut également observer une distribution binomiale de l'*inner distance* (figure 11) chez les mutants, signifiant la présence d'un grand nombre d'ARN (d'environ 40 pb) au moment de l'extraction. Il est donc possible, à première vue, que la mutation est entraînée une régulation négative d'autres gènes, car des ARN de cette taille pourrait correspondre à des pré-miARN impliqués dans la régulation négative épigénétique et post-traductionnelle via des complexes protéiques DICER/AGO. Il est également envisageable de filtrer les ARN de petites taille, pour ne s'intéresser qu'aux modifications d'expression pour les gènes codant pour des protéines (ceux-ci ayant très souvent une longueur en pb plus élevée que les petits ARN non-codants).

Concernant le projet, la principale difficulté a été de contourner le problème lié au *pull* de *singularity*. L'idée de charger manuellement le module est venue assez rapidement, mais des problèmes de connexion à *http://bioinfo.genotoul.fr* m'ont empêché de connaître le nom exact du module (*singularity-3.7.3*) et le charger, ce qui m'a fait perdre un peu de temps.