Projet Hackathon
sur la Reproductibilité

IODAA - AMI2B

Chaimae EL HOUJJAJI - Kexin LI - Anthony MOREAU - Pauline TURK

29 novembre 2021

Table des matières

1 Introduction

1.1 Crise de la reproductibilité

Le terme crise de la reproductibilité est apparu au début des années 2010 pour désigner la prise de conscience collective des failles méthodologiques affectant les articles publiés et revus par les pairs menant à l'impossibilité de reproduire une grande partie des résultats publiés dans les revues scientifiques de différents domaines. Cette crise a pris une ampleur d'autant plus grande du fait que dans certaines disciplines des découvertes considérées comme majeures se sont avérées difficile à reproduire.

Une des raisons qui explique ce phénomène majeur est que les revues scientifiques et les chercheurs incitent la publication de résultats significatifs et positifs. Ce phénomène est accentué par la culture du *publish or perish* qui voit les chercheurs jugés sur la quantité et l'impact de leurs publications scientifiques les forçant à publier pour avoir une carrière.

Les statistiques permettent de mieux comprendre pourquoi cela a conduit à la crise actuelle. Premièrement, seule une fraction des hypothèses scientifiques testées s'avérera significative. Si on teste de nombreuses hypothèses majoritairement fausses au seuil habituellement admis de 5% en ne publiant que les résultats positifs, on obtiendrait une partie non négligeable voire une majorité des publications correspondant en réalité à des faux positifs [1]. Deuxièmement, la reproduction de publications déjà existantes est fortement découragée car une reproduction d'étude est considérée comme moins novatrice donc serait moins publiée et aurait moins d'impact. Cela signifie donc qu'en cas de faux positif, le résultat ne serait probablement que peu vérifié et contredit.

Cependant, en plus des biais statistiques courants dans les publications scientifiques, on peut noter qu'un des problèmes provient de la difficulté à répliquer leurs protocoles expérimentaux. Certains projets visant à reproduire des publications déjà existantes ont soulevé que ces protocoles étaient souvent insuffisamment documentés et qu'une documentation détaillée améliorerait les chances de pouvoir reproduire le résultat. [2].

1.2 Présentation des articles

Dans ce projet centré sur la reproductibilité, nous nous sommes intéressés à deux publications :
— Article 1 : SF3B1 mutations are associated with alternative splicing in uveal melanoma
— Article 2 : Recurrent mutations at codon 625 of the splicing factor SF3B1 in uveal melanoma

Ces publications scientifiques de biologie portent sur le rôle des mutations du gène SF3B1 dans le cancer uvéal. Ces deux publications ont mis en évidence la présence de mutations dans ce gène pour une certaine proportion de tumeurs uvéales. Ce gène étant un facteur impliqué dans l'épissage, les chercheurs se demandent si ces mutations ne pourraient pas modifier la manière dont ces gènes s'expriment. Pour ce faire ces chercheurs ont récolté des données d'expression de l'ADN et la deuxième publication les a ré-analysées. Cependant, ils ne trouvent pas les mêmes résultats.

1.3 Objectifs du projet

Notre projet possède donc deux objectifs principaux. Premièrement nous allons reproduire la série d'analyses d'expression de l'ARN qui a été effectuée dans les deux publications décrites précédemment. Deuxièmement nous allons nous attacher à faire en sorte que cette analyse soit le plus facile à répliquer possible.

1. Why Most Published Research Findings Are False
2. Raise standards for preclinical cancer research

En effet, dans la première publication la section matériel et méthodes est absente, il faut donc consulter les suppléments de l'article pour avoir des informations sur les modes d'analyses utilisés. Dans la deuxième publication, la section matériel et méthodes est présente et donne des renseignements sur les analyses. Cet article possède aussi des suppléments mais ne donne pas tous les détails comme la version des logiciels utilisés.

Même si toutes les informations étaient présentes, il ne serait pas pour autant facile de répliquer le plus fidèlement possible leurs analyses car trouver et installer les bonnes versions des logiciels peut être compliqué sans compter le fait que nous n'avons pas accès aux codes exacts utilisés pour le faire. Pour répondre à ces problèmes, nous avons utilisé Snakemake. Ce programme permet d'exécuter une série d'analyses à partir d'un seul fichier. De plus Snakemake permet d'utiliser des conteneurs. Les conteneurs encapsulent la version utilisée du logiciel avec tout ce dont elle a besoin pour fonctionner. Cela permet une bonne robustesse du pipeline dans le temps et une facilité de déploiement pour pouvoir aisément répliquer les analyses.

Tout cela est fait dans le but de faciliter la tâche de futurs chercheurs qui voudraient reproduire nos résultats avec leurs propres données. L'objectif final est qu'on puisse rapidement récupérer et refaire tourner nos analyses.

1.4 Répartition des tâches

Le projet Hackathon reproductible a débuté en octobre par une présentation des différents outils et des attendus du projet. Nous avons commencé par la lecture et l'analyse des documents la première semaine de Novembre. Ensuite, Anthony a démarré les premières règles du Snakefile, que nous avons améliorées et déboguées à quatre. Cette étape a duré 2 à 3 semaines. Une fois le Snakefile fonctionnel, c'est à dire lorsque nous avons obtenu les fichiers de comptage, Chaimae a commencé le script DESeq2.R pour l'analyse de l'expression différentielle. Celui-ci a par la suite été débogué par nous tous. Pauline a rajouté dans le deuxième script R la comparaison de notre étude avec les deux articles. En parallèle, nous avons tous contribué à l'écriture du rapport. Kexin a initié l'écriture du README et structuré le répertoire Github.

2 Matériel et Méthodes

2.1 Liens vers la documentation du projet

Nous avons créé un répertoire github pour que nous puissions travailler ensemble à distance. Les fichiers utilisés et les résultats de l'analyse se trouvent dans le dépot github du groupe :
https://github.com/anthony-moreau/hackaton-ACKP

Il contient les documents suivants :

1. **README.md** : Explications pour faire tourner le pipeline.
2. **Snakefile** : Le pipeline d'analyses Snakemake a exécuter, plus d'explication la section **2.3**.
3. **DESeq2.R** : Le fichier R utilisé pour l'analyse différentielle dans le Snakefile.
4. **Rapport_hackathon_Groupe4.pdf** : Rapport du projet.
5. Le dossier **Groupe_4_R_results** :
   (a) **Analysis.Rdata** : Les données générées par le Snakefile. Ce fichier peut être importé dans R studio et ensuite utilisé pour d'autres analyses.
   (b) **Analysis_Rplots.pdf** : Les graphes générés par DESeq2.R, y compris un graphe des 9 gènes avec l'expression différentielle la plus grande entre les sauvages et les mutants, un Volcano Plot et une ACP.
   (c) **Analysis.html** : Le fichier html de la version R Markdown de DESeq2.R.

6. Le dossier **Groupe_4_counts_result** : Les fichiers .count correspondants aux expressions des gènes pour les différents fichiers SRR.
7. Le dossier **Groupe_4_Analyse_Sup_R** :
   (a) **ACP_EnhancedVolcano_Article2Plots.Rmd** : Script R d'analyse statistique supplémentaire.
   (b) **ACP_EnhancedVolcano_Article2Plots.html** : Sortie du script susmentionné en html.
   (c) **liste_fusion.xlsx** : Liste des gènes de notre étude et leurs caractéristiques (padj, LFC, ...) ainsi que les 325 gènes différentiellement exprimés de l'article 2 que nous avons récupéré dans la table supplémentaire 8.

2.2 Choix des outils

2.2.1 Snakemake

La répétabilité est essentielle pour l'analyse des données scientifiques. Snakemake est un outil de gestion de workflow permettant de créer des analyses reproductibles et extensibles. Il simplifie l'analyse des données en un processus textuel très lisible.

Snakemake décompose le workflow en règles. Ces règles définissent les fichiers d'entrée (input), les fichiers de sortie (output), et comment aller du fichier d'entrée au fichier de sortie. Snakemake connecte les sorties aux entrées correspondantes pour former un workflow. Lorsqu'il détecte qu'un output est produit et qu'une autre règle l'utilise en entrée, Snakemake lance la règle suivante.
La syntaxe de base d'une règle est la suivante :

rule exemple

```
rule nom_de_la_règle :
    input :
        a = "chemin/des/inputs/{nom_des_fichiers}.ext",
        b = "chemin/des/inputs/inputfile"
    output :
        "chemin/des/outputfiles"
    shell :
        """
        une_commande {input.a} > {output}
        une_autre_commande {input.b} > {output}
        """
```

Il est possible de donner plusieurs entrées et ou générer plusieurs sorties par règle, simplement en donnant le nom de chaque fichier via nom_des_fichiers = ["nom du fichier1", "nom du fichier2",...] et en utilisant la fonction expand et le système de wildcard.

Les règles ne sont pas limitées aux commandes shell. La directive "shell" peut être remplacée par : "run" pour écrire du code Python, "script" pour faire référence à un fichier Python ou R externe et "wrapper" pour faire appel à un script dans le *Snakemake Wrappers Repository*. Cela simplifie le travail et rend Snakemake plus pratique.

Snakemake peut utiliser la directive "threads" pour spécifier le nombre de cœurs de CPU dont une règle a besoin lors de son exécution. Si le nombre de threads est inférieur au nombre de `--cores`, Snakemake cherchera d'autres règles pour utiliser le CPU restant. Le worflow peut donc être accéléré en ajustant la nombre de threads.

2.2.2 Singularity

Le processus d'analyse bio-informatique consomme souvent des dizaines de milliers de ressources de calcul. En outre, il fait appel à un grand nombre de programmes et de scripts, qui nécessite la configuration et la gestion de l'environnement d'exécution. Cela peut engendrer des problèmes de compatibilité et de dépendances. L'utilisation de conteneurs permet de palier ce problème.

Singularity est un programme permettant d'exécuter des conteneurs. Un conteneur est une image permettant d'intégrer une application et tout ce dont elle a besoin pour fonctionner. Les avantages des conteneurs pour l'analyse automatisée des données scientifiques sont donc multiples. Par exemple, les conteneurs permettent une plus grande stabilité dans le temps des différentes versions d'une application mais également l'élimination des conflits de dépendances entre les applications. En effet, chaque application tourne dans un environnement virtuel séparé. Ces différents avantages permettent une plus grande reproductibilité dans le temps des analyses en évitant qu'un pipeline ne puisse plus être exécuté du fait des évolutions systèmes et de l'obsolescence des programmes utilisés. Snakemake permet d'utiliser des conteneurs via la directive "singularity".

Contrairement à Docker, Singularity ne demande pas que les fichiers créés appartiennent au groupe de l'administrateur système et donc les pipelines peuvent s'exécuter sur des clusters de calcul où les utilisateurs ne possèdent habituellement ces droits. C'est donc une solution plus populaire pour l'analyse en bioinformatique.

2.2.3 Machine Virtuelle (VM)

Nous travaillons en distanciel sur les VM ubuntu de cloud de l'Institut Français de Bioinformatique (IFB). Nous avons utilisé les VM de 2 Cœurs, 8 Go de RAM et 120 Go de stockage pour nos tests et une VM de 16 Cœurs, 64 Go de RAM et 920 Go de stockage pour faire tourner le pipeline.

2.3 Description du workflow - les étapes du Snakefile

2.3.1 DAG (Directed Acyclic Graph)

La figure 1 est le DAG de notre workflow illustrant les étapes de notre analyse reproductible. Celui-ci a été généré avec la commande snakemake `-- rulegraph`.

Chacune de ces étapes peut être répétée une ou plusieurs fois, selon les données analysées. Par exemple, l'étape de téléchargement des chromosomes `download_chromosomes_sequence` se fera autant de fois qu'il y a de chromosomes à télécharger, alors que l'étape de création du génome de référence `create_reference_file` ne se fait qu'une seule fois comme elle concatène les chromosomes téléchargés. Ainsi, un travail de parallélisation a été effectué dans ce workflow par l'attribution de valeurs de threads appropriées à chaque étape.

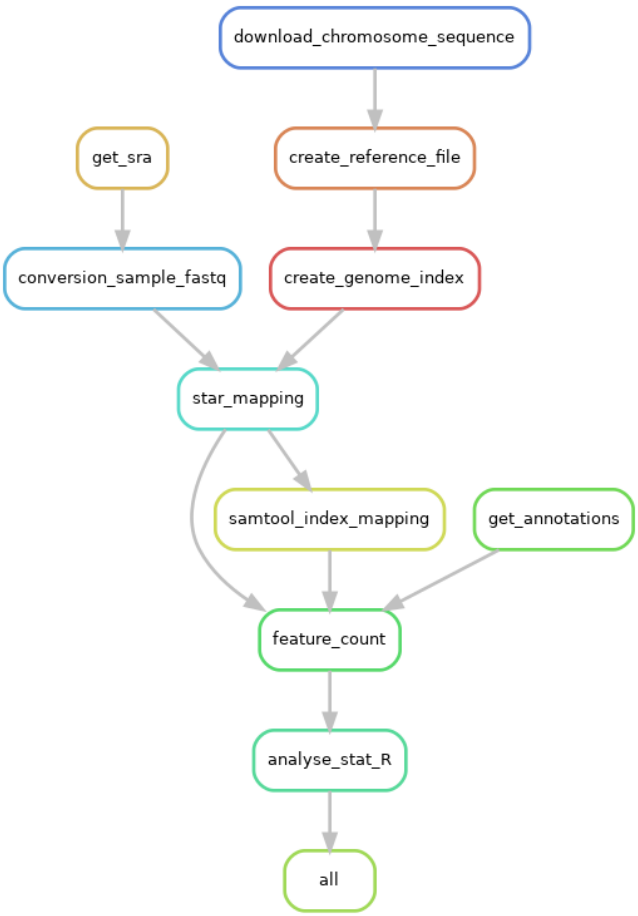


FIGURE 1: DAG décrivant les différentes étapes du workflow.

2.3.2 Téléchargement des échantillons et conversion en .fastq

Huit échantillons transcriptomiques mentionnés dans l'article 1 ont été choisis pour constituer l'ensemble des échantillons tests. Ils sont représentés par leur numéro d'accession dans la liste `accession_numbers` et sont téléchargés depuis le portail du NCBI sous format .sra et placés dans le dossier `SRA_files`.

rule get_sra

accessions_numbers = [”SRR628582”, ”SRR628583”, ”SRR628584”, ”SRR628585”, ”SRR628586”, ”SRR628587”,”SRR628588”, ”SRR628589”]

rule get_sra :
    output :
        ”SRA_files/{sample}.sra”
    threads : 8
    shell :
        ”wget -O {output} `https://sra-downloadb.be-md.ncbi.nlm.nih.gov/sos1/sra-pub-run-5/{wildcards.sample}/{wildcards.sample}`”

Chaque échantillon est ensuite converti en deux fichiers .fastq grâce à la fonction fastq-dump du container `evolbioinfo/sratoolkit:v2.5.7` importé par Singularity. L'option `--split-files` permet d'appliquer la méthode "Paired-end" pour une double lecture *forward* et *backward* des échantillons. Ces fichiers sont placés dans le dossier `fastq_files`.

rule conversion_sample_fastq

```
rule conversion_sample_fastq :
    input :
        "SRA_files/{sample}.sra"
    output :
        "fastq_files/{sample}_1.fastq",
        "fastq_files/{sample}_2.fastq"
    singularity :
        "docker ://evolbioinfo/sratoolkit :v2.5.7"
    threads :4
    shell :
        " fastq-dump --split-files ./{input} -O fastq_files"
```

2.3.3 Travail préliminaire sur le génome humain de référence

Afin de mapper les fichiers `.fastq` générés précédemment sur le génome humain de référence et pour l'étape de comptage pour estimer l'expression des gènes, un travail préliminaire de reconstruction, indexation et annotation du génome humain est effectué.

Téléchargement des chromosomes du génome de référence

La première étape consiste en le téléchargement des chromosomes du génome humain de référence choisi : GRCh38-release-101. Les chromosomes de la liste `chromosomes` sont téléchargés depuis Ensembl au format `.fa.gz` et enregistrés dans le fichier `chr_files`.

rule download_chromosome_sequence

```
chromosomes = ["1", "2","3","4","5","6","7","8","9","10","11","12","13","14","15","16","17","18",
"19","20","21","22","MT"]

rule download_chromosome_sequence :
    output :
        "chr_files/{chr}.fa.gz"
    threads : 8
    shell :
        """
        wget -O {output} ftp://ftp.ensembl.org/pub/release-101/fasta/homo_sapiens/dna/
Homo_sapiens.GRCh38.dna.chromosome.{wildcards.chr}.fa.gz
        """
```

Reconstruction du génome de référence

Afin de reconstituer le génome humain de référence complet pour le mapping des fichiers fastq, les chromosomes téléchargés sont dézippés avec la commande `gunzip` puis concaténés avec la fonction `expand` en un fichier unique `ref.fa`.

rule create_reference_file

```
rule create_reference_file :
    input :
        expand("chr_files/{chr}.fa.gz", chr= chromosomes)
    output :
        "ref.fa"
    shell :
        "gunzip -c {input} > {output}"
```

Indexation du génome de référence

Le génome de référence reconstruit `ref.fa`, est ensuite indexé grâce à la fonction `STAR` du conteneur `evolbioinfo/star:v2.7.6a`. Plusieurs paramètres de cette fonction sont utilisés :

- `--runThreadN {threads}` : parallélise des alignements.
- `--runMode genomeGenerate` : indexe un génome.
- `--genomeDir ref_index/` : le chemin d'accès du génome indexé généré.
- `--genomeSAindexNbases 6` : ce paramètre a été utilisé pour le travail de débogage lorsque nous testions notre workflow sur l'ADN mitochondrial. Il n'est pas obligatoire lorsqu'on travaille sur génome entier [3].
- `--genomeFastaFiles {input}` : le chemin d'accès du génome à indexer.

Nous précisons les fichiers `ref_index/SA` et `ref_index/SAindex` pour éviter que la prochaine étape de mapping ne débute avant que l'étape d'indexation ne soit terminée. Le dossier `ref_index` obtenu contient le génome de référence indexé qui va permettre d'accélérer l'étape de mapping.

rule create_genome_index

```
rule create_genome_index :
    input :
        "ref.fa"
    output :
        "ref_index/SA",
        "ref_index/SAindex"
    threads : 4
    singularity :
        "docker ://evolbioinfo/star :v2.7.6a"
    shell :
        """
        STAR --runThreadN {threads} --runMode genomeGenerate --genomeDir ref_index/
--genomeSAindexNbases 6 --genomeFastaFiles {input}
        """
```

3. Pour plus d'informations, consulter la section "2.2.5 Very small genome" du manuel de STAR

Téléchargement du génome humain annoté

Les annotations du génome humain pris comme référence (GRCh38.101) au format `.gtf.gz` sont téléchargées depuis Ensembl et enregistrées ensuite dans le fichier `annotations` ainsi que leur version dézippée. Ces annotations permettront d'identifier les gènes lors de l'étape de comptage.

rule get_annotations

```
rule get_annotations :
    output :
        genome_annot_zip = "annotations/human_genome_annotation.chr.gtf.gz",
        genome_annot = "annotations/human_genome_annotation.chr.gtf"
    shell :
        """
        wget -O {output.genome_annot_zip} ftp://ftp.ensembl.org/pub/release-101/gtf/
homo_sapiens/Homo_sapiens.GRCh38.101.chr.gtf.gz
        gunzip -c {output.genome_annot_zip} > {output.genome_annot}
        """
```

2.3.4 Mapping des fichiers fastq sur le génome humain de référence

Nous introduisons par la suite une règle qui prend en entrée les fichiers de séquençage du dossier `fastq_files`. Elle prend également en entrée les fichiers `refindex/SA` et `refindex/SAindex` pour indiquer au Snakefile de ne pas lancer le mapping avant que l'indexation ne soit terminée. Cette règle permet de remettre dans le bon ordre les séquences par une technique d'alignement. En effet, on aligne les reads sur le génome humain de référence obtenu dans la règle `create_genome_index`. On obtient en sortie des fichiers `.bam` pour **B**inary **A**lignment **M**ap. Il s'agit d'un format de cartographie d'alignement binaire qui est constitué de données brutes complètes sur le séquençage génomique.
Nous utilisons les options suivantes dans le container STAR :

`--outSAMstrandField intronMotif` : On filtre les reads avec des introns incohérents et/ou non canonique.
`--outFilterMismatchNmax 4` : On garde les alignements qui ont moins de 4 mauvais appariements.
`--outFilterMultimapNmax 10` : Le nombre maximum de loci auxquels les reads sont autorisés à correspondre équivaut à 10.
`--genomeDir ref_index/` : On indique que le génome de référence se situe dans le dossier `ref_index`
`--readFilesIn {input.fastq1} {input.fastq2}` : Il s'agit des fichiers `fastq1` et `fastq2` à cartographier.
`--runThreadN {threads}` : Nombre de threads pour exécuter STAR.
`--outSAMunmapped None` : On ne demande pas en sortie les fichiers pour les lectures non mappées.
`--outSAMtype BAM SortedByCoordinate` : Commande pour trier les fichiers BAM en sortie.
`--genomeLoad NoSharedMemory` : Paramètre qui contrôle la façon dont le génome est chargé en mémoire.
`--limitBAMsortRAM 50000000000` : RAM maximale disponible pour le tri des BAM.
`> {output}` : Pour enregistrer dans le fichier de sortie.

Plus de détails sur le container `STAR` sont disponibles dans le manuel utilisateur STAR.

rule star_mapping

```
rule star_mapping :
    input :
        fastq1 = "fastq_files/{sample}_1.fastq",
        fastq2 = "fastq_files/{sample}_2.fastq",
        genome_SA = "ref_index/SA",
        genome_SAindex = "ref_index/SAindex"
    output :
        "mapping/{sample}.bam"
    threads : 16
    singularity :
        "docker ://evolbioinfo/star :v2.7.6a"
    shell :
        """
        STAR --outSAMstrandField intronMotif --outFilterMismatchNmax 4
--outFilterMultimapNmax 10 --genomeDir ref_index/ --readFilesIn {input.fastq1} {input.fastq2}
--runThreadN {threads} --outSAMunmapped None --outSAMtype BAM SortedByCoordinate
--outStd BAM_SortedByCoordinate --genomeLoad NoSharedMemory --limitBAMsortRAM
50000000000 > {output}
        """
```

Par la suite, les fichiers `.bam` sont convertis en fichiers `.bam.bai` à l'aider du conteneur `evolbioinfo/samtools:v1.11`. En effet, les fichiers BAM sont indexés au travers d'un fichier BAI qui les accompagne. Celui-ci est équivalent à une table des matières qui accompagne le fichier BAM et permet d'accéder directement à des parties spécifiques du fichier concerné. Cela permet d'accélérer fortement l'accès aux données. La fonction `samtools index {input}` permet d'indexer les fichiers BAM.

Plus de détails sur le conteneur `Samtools` sont disponibles dans le manuel utilisateur Samtools.

rule samtool_index_mapping :

```
rule samtool_index_mapping :
    input :
        "mapping/{sample}.bam"
    output :
        "mapping/{sample}.bam.bai"
    singularity :
        "docker ://evolbioinfo/samtools :v1.11"
    shell :
        "samtools index {input}"
```

2.3.5 Comptage des "reads"

La règle `feature_count` permet de générer en sortie pour chaque SRR deux fichiers : un fichier de comptage `.counts` et un fichier de résumé : `.counts.summary` qui sont localisés dans le dossier `result`. Cette règle prend en entrée les fichiers de mapping, le génome annoté ainsi que les fichiers de mapping indexés. Elle utilise le conteneur `evolbioinfo/subread :v2.0.1`.

On utilise les paramètres suivants dans le conteneur `subread` :

`-T {threads}` : permet d'indiquer le nombre de threads qu'on souhaite.

`-t gene` : permet de spécifier le type de la séquence, il est défini à "exon" par défaut.
`-g gene_id` : permet de spécifier le type d'attribut utilisé pour regrouper les caractéristiques en méta-caractéristiques lorsque l'annotation GTF est fournie.
`-s 0` : 3 valeurs sont possibles pour cet argument : 0 (*unstranded*), 1 (*stranded*) ou 2 (*reversely stranded*).
`-a {input.genome_annotation}` : fournit le nom du fichier d'annotation.
`-o {output}` : donne le nom du fichier de sortie. Celui-ci contient le nombre de lectures attribuées à chaque méta-caractéristique.

Plus de détails sur le container subread sont disponibles dans le manuel utilisateur Subread.

rule feature_count :

```
rule feature_count :
    input :
        sample_mapping = "mapping/{sample}.bam",
        genome_annotation = "annotations/human_genome_annotation.chr.gtf",
        index = "mapping/{sample}.bam.bai"
    output :
        "result/{sample}.counts"
    threads : 4
    singularity :
        "docker ://evolbioinfo/subread :v2.0.1"
    shell :
        "featureCounts -T {threads} -t gene -g gene_id -s 0 -a {input.genome_annotation} -o {output}
{input.sample_mapping}"
```

2.4 Analyse statistique

2.4.1 Analyse clef

La dernière règle `rule_analyse_stat_R` permet de tester la significativité de l'expression différentielle des gènes entre les cas sauvages (5 échantillons) et mutés en SF3B1 (3 échantillons). Elle prend en entrée les fichiers `.counts` obtenus précédemment pour chaque échantillon. L'analyse se fait grâce à la bibliothèque Desq2 du conteneur `evolbioinfo/deseq2:v1.28.1` et l'appel au script R `DESeq2.R`.

On obtient en sortie le dossier `analysis` contenant le fichier `Analysis.Rdata` ainsi que le fichier `Analysis_Rplots.pdf` contenant les résultats et graphes d'intérêt pour cette analyse différentielle.

Le dossier des résultats de l'analyse est obtenu par la règle `rull_all`.

rule_analyse_stat_R :

```
rule_analyse_stat_R :
    input :
        expand("result/{sample}.counts", sample = accessions_numbers)
    output :
        "analysis_R/Analysis.Rdata"
    threads : 4
    singularity :"docker ://evolbioinfo/deseq2 :v1.28.1"
    script :
        "DESeq2.R"
```

rule all :

```
rule all :
    input :
        "analysis_R/Analysis.Rdata"
```

2.4.2 Analyse supplémentaire

Pour compléter l'analyse statistique, un deuxième script R, `ACP_EnhancedVolcano_Article2Plots.Rmd` extérieur au Snakefile et qui reprend l'analyse faite dans `DESeq2.R` est disponible sur le répertoire github du groupe. Il réalise une ACP sur les 8 échantillons de l'étude grâce aux bibliothèques `FactoMineR` et `factoextra` afin de visualiser leur répartition sur les dimensions les plus explicatives.

De plus, les bibliothèques `BiocManager` et `EnhancedVolcano` sont aussi utilisées pour obtenir des *Enhanced Volcano Plot* permettant de mieux visualiser les gènes à plus forte expression différentielle.

Ce script supplémentaire permet aussi de bien visualiser les gènes à expression différentielle de notre étude et de les comparer avec quelques résultats de l'article 2.

Nous sommes conscients que cette analyse supplémentaire n'est pas optimale au niveau de la reproductibilité puisqu'elle nécessite l'installation de bibliothèques supplémentaires. De plus, cela dépend également de l'environnement ainsi que de la version de R installée. Toutefois, nous tenions à rajouter cette analyse afin de pouvoir comprendre biologiquement nos résultats.

3 Résultats

3.1 Analyse en Composantes Principales

Nous effectuons dans un premier temps, une visualisation des 8 échantillons par une ACP sur RStudio.

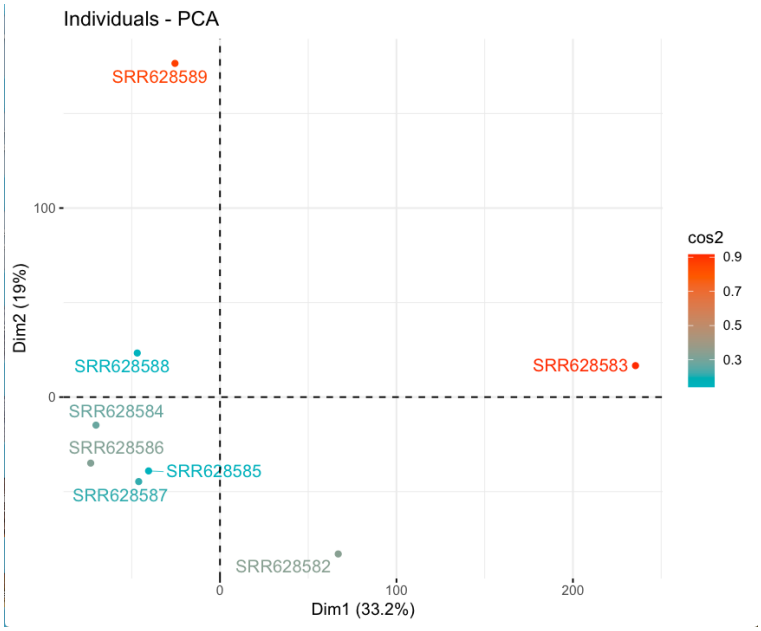


FIGURE 2: Analyse en Composantes Principales des 8 échantillons.

Sur la figure obtenue (fig.2), les deux points extrêmes (avec cos^2 proche de 1) correspondent aux échantillons SRR628589 et SRR628583 qui se distinguent fortement du reste du groupe. Il s'agit d'échantillons qui possèdent le gène SF3B1 muté. Nous pouvons aussi relever l'échantillon SRR628582 qui se différencie un peu moins du groupe que les deux échantillons précédents mais qui correspond également à un échantillon possèdant le gène SF3B1 muté. L'ACP permet de visualiser les individus "Wild Type" avec le gène non muté qui sont quant à eux bien regroupés.

Ainsi l'ACP est une première approche pour distinguer le groupe des mutants des non mutants. Cela laisse supposer une expression différentielles des gènes entre ces deux groupes que nous allons confirmer par la suite. Toutefois, même si les dimensions de projections 1 et 2 cumulent un bon pourcentage explicatif des variances (52,2%), il aurait été intéressant de réitérer l'ACP en retirant les gènes de poids trop faible dans la définition des axes de projection afin d'espérer mieux distinguer les cas mutés des cas sauvages.

3.2 Expression différentielle des gènes

DESeq 2 est l'outil choisi pour cette analyse différentielle comme étant adapté aux données biologique de RNAseq (Differential expression analysis using DESeq2). Il prend en entrée le tableau des comptages pour chaque gène (en ligne) et ce pour chaque individu (en colonne) ainsi qu'une table des méta-données qui associe chaque individu au groupe expérimental auquel il appartient (wild-type ou muté en SF3B1). Ainsi, on peut obtenir le *Volcano Plot* (fig.3) qui permet de mettre en évidence les gènes à plus forte expression différentielle.

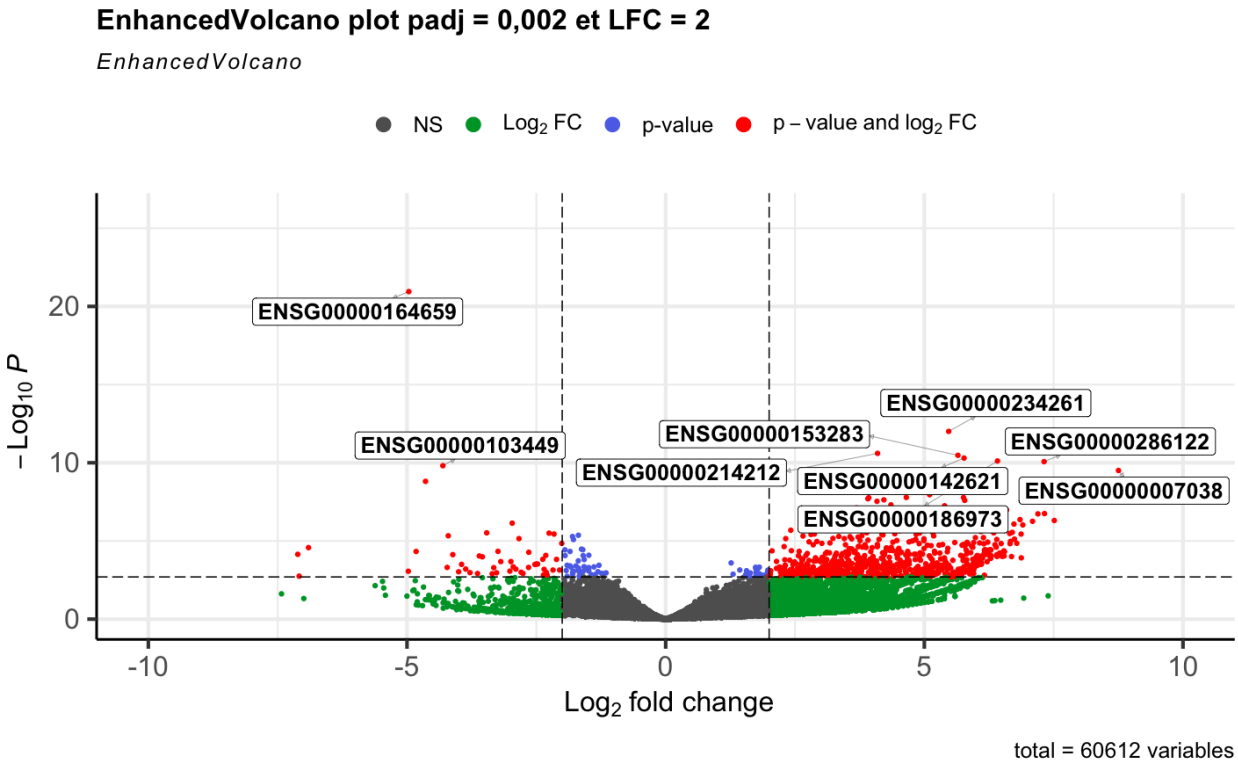


FIGURE 3: *Volcano plot* sur 60612 transcripts. Chaque point est associé à un gène dont le comptage à l'issue du workflow est non nul. Le Fold Change est calculé en prenant le cas Wilde-type comme référence. La zone en rouge est celle d'intérêt principal puisqu'elle est associée aux gènes dont l'expression différentielle dans le cas muté par rapport au cas sauvage est la plus grande et avec une forte significativité.

Le **fold change** est défini comme le rapport des counts entre les deux groupes expérimentaux. Le cas Wild-type a été pris comme référence, ainsi les cas de sur-expression de gène se situent dans les valeurs positives en abscisse, alors que les cas de sous-expression sont dans le domaine des valeurs négatives. De plus, la transformation en Log_2 permet une interprétation intuitive des abscisses. En effet, un Log_2 fold change = 1 signifie une expression multipliée par 2^1 dans le cas muté par rapport au cas wild-type et un Log_2 fold change = -1 signifie une expression divisée par 2^1 dans le cas muté par rapport au cas wild-type.

En ordonnée, $-Log_{10}\mathbf{P}$ permet de simplifier la lecture de l'axe des ordonnées avec une p-value d'autant plus faible et donc une significativité de l'expression différentielle d'autant plus forte que la valeur de $-Log_{10}$P est élevée tout en permettant d'afficher de grandes plages de valeurs de p-value. La p-value adjusted est obtenue à partir de la p-value par correction de Benjamini-Hochberg, qui est la méthode par défaut de DESeq2.

Les valeurs de p-value adjusted à 0,002 et Log_2 fold change à 2 ont été choisies de telle sorte à sélectionner la zone des gènes à expression différentielle élevée et où la densité des points qui les représentent sur le *Volcano Plot* est suffisamment faible pour pouvoir voir des points individuels.

Nous observons une asymétrie du *Volcano Plot* avec plus de gènes sur-exprimés dans les cas mutés (zone rouge de droite) que de gènes sous-exprimés (zone rouge de gauche). Ceci est en désaccord avec les résultats obtenus dans les 2 articles qui ont observé plus de gènes sous-exprimés que sur-exprimés.

Les étiquettes sur le *Volcano Plot* (fig.3) sont les numéros d'accession dans la base Ensembl des 9 gènes à expression différentielle dont les p-value ajustées sont les plus faibles.

La figure 4 représente le comptage de ces 9 gènes pour les 8 échantillons de notre étude. La normalisation du comptage (en ordonnée) se fait par la division du comptage pour chaque gène dans chaque échantillon par la moyenne géométrique de ce gène sur tous les échantillons.

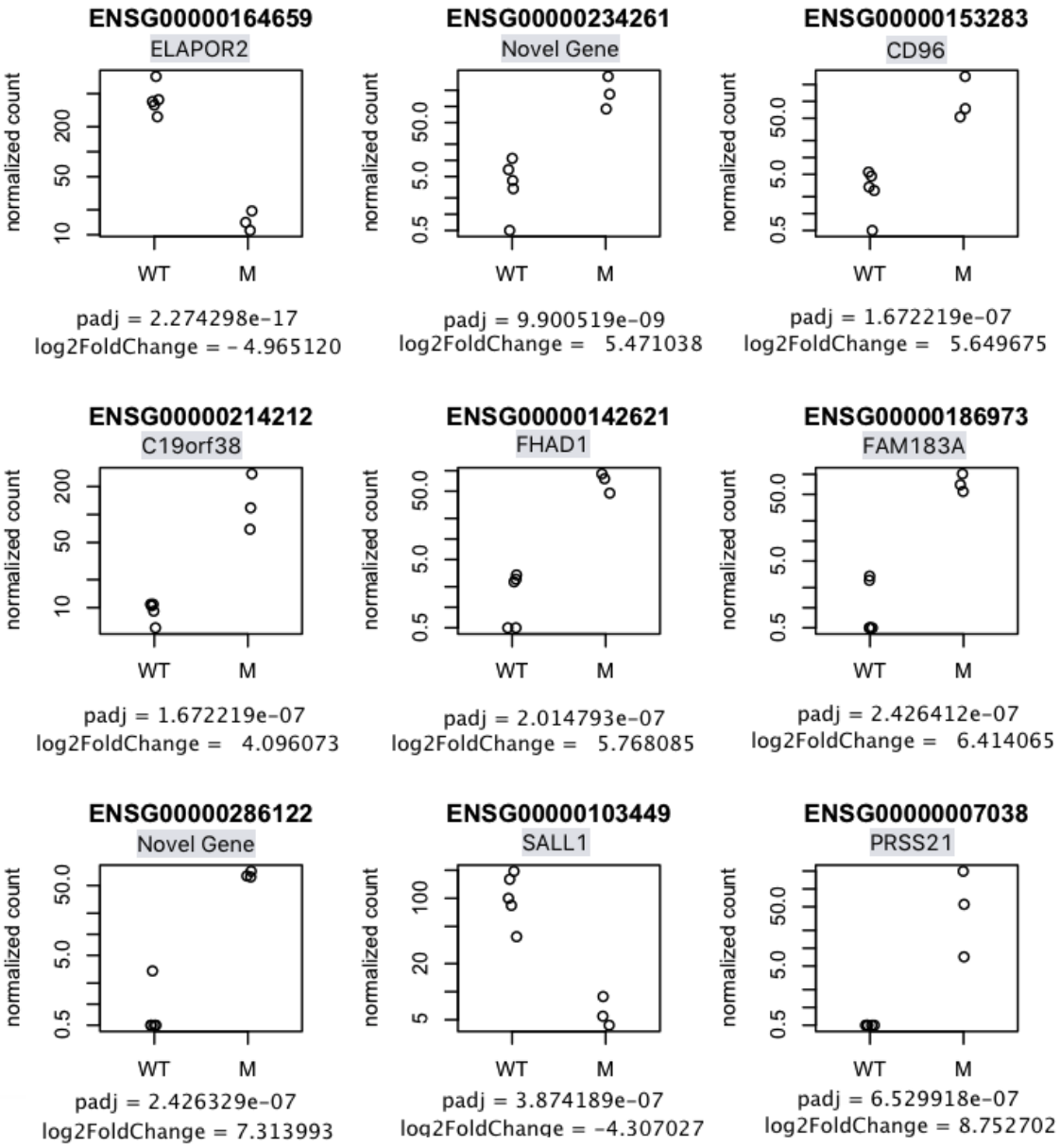


FIGURE 4: Compte normalisé des 9 gènes à p-value ajustées les plus basses obtenues par DESeq2 sur les 8 échantillons de l'étude. M : SF3B1 Muté, WT : SF3B1 Wild-Type

3.3 Comparaison des résultats d'expression différentielle avec ceux des articles

3.3.1 Article 1 : SF3B1 mutations are associated with alternative splicing in uveal melanoma

Le tableau ci-dessous regroupe les 10 gènes à plus forte expression différentielle dans l'article 1 par ordre de significativité décroissante (p-value ajustée croissante). Nous pouvons définir `Rang_padj_1` comme ce rang de significativité croissante dans l'article 1. Un rang équivalent défini pour notre étude, ordonne les 60612 transcripts par p-value ajustée croissante, il est noté `Rang_padj_Hack`.

Rang_padj_1	Gène	N° d'accession (Ensembl)	Rang_padj_Hack
1	SERINC3	ENSG00000132824	7560
2	PIH1D1	ENSG00000104872	10030
3	ANKHD1	ENSG00000131503	31571
4	NDUFB8	ENSG00000166136	45055
5	ELP2	ENSG00000134759	2311
6	CCL28	ENSG00000151882	16250
7	SEPT2	ENSG00000286588	26255
8	MMUT	ENSG00000146085	2994
9	NRD1	ENSG00000178607	5999
10	DOCK7	ENSG00000116641	4438

Nous constatons que les 10 gènes à plus forte expression différentielle dans l'article 1 ne correspondent pas à ceux obtenus dans notre étude.

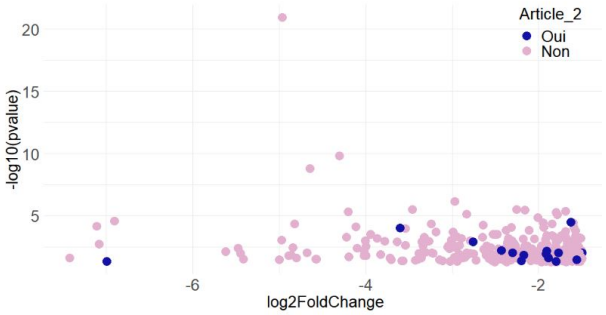
3.3.2 Article 2 : Recurrent mutations at codon 625 of the splicing factor SF3B1 in uveal melanoma

Dans l'article 2, les gènes à expression différentielle significative sont ceux vérifiant :

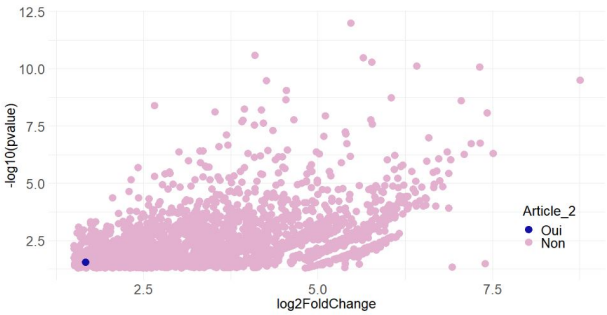
* $|Fold - change| \geq 1{,}5$
* p-value $\leq 0{,}05$.

Dans cet article, 325 gènes vérifient ces conditions dont 46 sont associés à une sur-expression et 279 à une sous-expression dans les cas SF3B1 muté par rapport aux cas SF3B1 wild-type.

La comparaison de nos résultats de comptage à ceux de l'article 2 [4] est résumée par les graphes de la figure 5. En effet, les points représentent les transcrits vérifiant les conditions sur Fold-change et p-value définies dans l'article 2 pour considérer une expression différentielle. Les figures 5a et 5b correspondent respectivement aux zones en rouge de gauche et de droite de la figure 3. Les points en bleu sont les transcrits de ces zones qui ont été observés comme étant à expression différentielle dans l'article 2.



(a) Graphe des transcrits sous-exprimés d'après notre étude suivant les critères de l'article 2, avec en bleu les transcripts communs à l'étude et à l'article 2



(b) Graphe des transcrits sur-exprimés d'après notre étude suivant les critères de l'article 2, avec en bleu les transcripts communs à l'étude et à l'article 2

FIGURE 5: Comparaison des résultats d'expression différentielle des transcripts entre notre étude et l'article 2

4. Données de comptage disponibles dans le "Table 8" des données supplémentaires de l'article 2.

Ainsi, on retrouve quelques transcripts en commun pour les cas de sous-expression (fig.5a), mais pas forcément ceux pour lesquels nous avons trouvé un plus grand différentiel d'expression et avec une plus forte significativité.
Pour les cas de sur-expression, nous ne trouvons qu'un seul transcript en commun, malgré le fait que nous trouvons plus de transcripts sur-exprimés que sous-exprimés. De plus celui-ci fait parti des sur-expressions faibles dans notre étude.

D'autre part, SF3B1 n'est pas dans la liste des 325 transcripts à expression différentielle significative de l'article 2. Ceci est en conformité avec nos résultats, l'expression de ce gène semble être indépendant de l'état sauvage ou muté (fig.6).

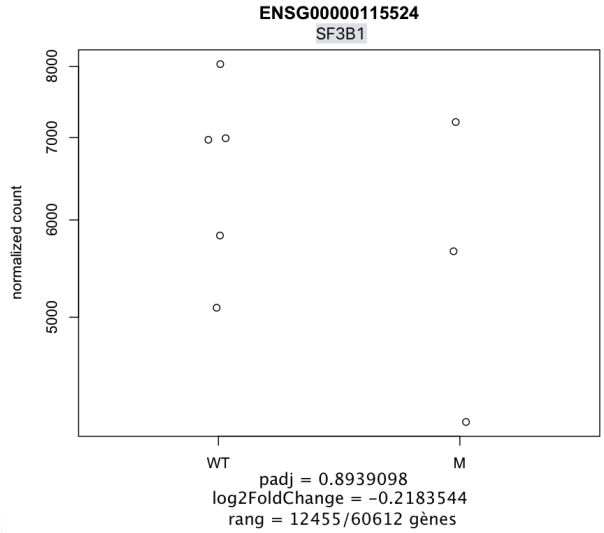


FIGURE 6: Comptage normalisé du gène SF3B1 obtenu par DESeq2 sur les 8 échantillons de l'étude. M : SF3B1 Muté, WT : SF3B1 Wild-Type

4 Difficultés rencontrées

4.1 Le workflow Snakemake

1. **Compréhension du workflow**

   L'une des principales difficultés de ce projet concerne l'implémentation dans Snakemake d'un pipeline complexe. Celle-ci a nécessité l'apprentissage de l'utilisation de Snakemake ainsi que la compréhension des différentes règles du pipeline, ce qu'elles prennent en entrée, leurs fonctions, les conteneurs utilisés ainsi que les sorties produites. La plupart des programmes utilisés sont documentés mais la documentation n'est pas toujours très explicite sur certains points ce qui a ralentit le débogage du workflow.

2. **Faire fonctionner le workflow**

   Parvenir à solutionner les bugs s'est avéré être la partie qui a demandé le plus de temps. Trouver la source d'un problème dans le workflow Snakemake est souvent un processus long et compliqué demandant beaucoup d'essais et d'erreurs. Une des raison est liée aux messages d'erreurs renvoyés qui ne sont pas toujours évidentes à comprendre, Snakemake pouvant parfaitement afficher qu'il y a une erreur de syntaxe à la ligne 151 pour un fichier de seulement 70 lignes de long. Une autre erreur

récurrente qu'on obtenait et qui n'était pas simple à déboguer était " `one of the commands exited with non-zero exit code; note that snakemake uses bash strict mode !` ".

Il s'agissait pour les quatre membres du groupe de la première fois que nous utilisions cet outil dédié au pipeline, ce qui explique un temps assez conséquent passé à déboguer et réaliser les tests. En effet, dans nos tests nous avons utilisé un seul chromosome pour simplifier les calculs. Cependant STAR est calibré pour un génome de taille humaine et demande un ajustement si la taille est différente. Se rendre compte de l'erreur demande de lancer le workflow une première fois, de voir l'avertissement de STAR, puis faire une correction avant de relancer le workflow.

4.2 Quota d'utilisation du cloud biosphère

Le groupe Hackaton reproductible disposait d'un quota commun d'utilisation du cloud biosphère. En cours de projet, ce quota s'est retrouvé épuisé. Cela nous a ralenti car nous avions besoin d'une machine virtuelle plus puissante pour relancer le Snakefile avec tous les échantillons et le génome humain complet. Il aurait cependant été possible de consommer moins de ressources en éteignant les VM lorsqu'elles n'étaient pas utilisées.

5 Conclusion

Dans ce projet nous nous sommes attachés à reproduire les analyses de deux publications scientifiques tout en nous assurant que nos résultats soient facilement reproductibles. Les systèmes de workflow et les conteneurs sont des technologies très utiles pour y parvenir.

Les deux publications scientifiques arrivent à des conclusions différentes. Nos analyses apportent encore des résultats différents des deux études précédentes. Cela incite donc à une forte prudence vis-à-vis des résultats et des thèses avancées dans les deux articles car, contrairement à ce qu'on pourrait penser en lisant simplement un des articles, les conclusions concernant SF3B1 sont peu robustes face aux méthodes utilisées et aux modèles choisis dans les différents programmes impliqués dans les analyses.

À l'issu de ce projet, on peut donc dire que la faible reproductibilité des résultats des articles amène à penser qu'il n'y a pas de conclusion claire sur l'influence des mutations du gène SF3B1 dans le cancer uvéal. Une des pistes pour trancher de manière plus convaincante serait de refaire des analyses sur un nombre d'échantillons beaucoup plus important. Cela permettrait de diminuer l'impact de la variabilité entre individus sur les résultats et ainsi espérer obtenir des résultats plus robustes et reproductibles.