université
PARIS-SACLAY

Master 2- AMI2B

UE Hackathon

Projet Hackathon Reproductible

RIAHI Ilhem
WENG Lorraine
CHERIFI Sana
AMMAR Fatma

Enseignants :
Mr.Lemoine Frédéric
Mr.Cokelaer Thomas

2021 — 2022

Table des matières

1 Introduction :

La reproductibilité :

Le monde de la recherche prend conscience depuis une dizaine d'années d'une crise de reproductibilité des démarches bioinformatiques. Le terme de reproductibilité fait référence à la capacité d'un chercheur ou d'une équipe à reproduire les résultats d'une étude réalisée antérieurement en se basant sur les mêmes matériaux que ceux utilisés par le chercheur ou l'équipe initiale. Loin d'être une pratique simple, la reproduction nécessite des indications précises sur le déroulement détaillé de l'expérience, le matériel requis ainsi que sur les méthodes d'analyses et de collecte des données. De même, les moyens pour mobiliser ces ressources doivent être précisément décrits.
De façon de plus en plus importante, les travaux et publications en sciences sont discutés pour leur manque de rigueur, et un nombre croissant de chercheurs appellent à un renouvellement radical des pratiques scientifiques. Ainsi, une enquête publiée par Nature en 2016 [1] démontre que sur 1576 chercheurs, plus de 70% d'entre eux peinent à reproduire des expériences préexistantes.
Ce sérieux manque de reproductibilité fragilise le monde des sciences et remet en cause la fiabilité des résultats de nombreuses publications. La plupart des problèmes de reproductibilité s'expliquent en grande partie par des pratiques discutables pouvant aller jusqu'à une fabrication complète des résultats.
On distingue trois principaux types de reproductibilité : la reproductibilité empirique dans un premier temps qui tient compte des expériences et observations scientifiques. Le caractère variable et aléatoire des phénomènes biologiques constitue un des principaux obstacles à cette reproductibilité. De même les techniques expérimentales en elles-mêmes apportent une variabilité selon les conditions et opérateurs. Ensuite, la reproductibilité statistique qui dépend du choix pertinent des tests statistiques, des paramètres et des valeurs seuils. Enfin, la reproductibilité computationnelle concerne le code en lui-même mais également les détails sur les logiciels et versions utilisés.

Les objectifs de ce projet :

L'ensemble du projet repose sur l'étude du mélanome de l'uvée, à savoir le cancer de l'œil le plus fréquent chez l'adulte, touchant chaque année près de 600 individus en France. Cette forme de cancer est en particulier associée à la mutation du gène SF3B1. Ce gène code pour un facteur d'épissage (splicing factor 3b) appartenant au complexe nucléaire ribonucléoprotéique U2 du spliceosome, impliqué dans l'épissage des ARN pré-messagers.
Nous nous sommes basés sur les deux publications suivantes :

- Harbour JW, Roberson ED, Anbunathan H, Onken MD, Worley LA, Bowcock AM. Recurrent mutations at codon 625 of the splicing factor SF3B1 in uveal melanoma. Nat Genet. 2013 ;45(2) :133-135. doi :10.1038/ng.2523
- Furney SJ, Pedersen M, Gentien D, et al. SF3B1 mutations are associated with alternative splicing in uveal melanoma. Cancer Discov. 2013 ;3(10) :1122-1129. doi :10.1158/2159- 8290.CD-13-0330

Les expériences menées par le premier groupe de chercheurs ont été reprises par les seconds. Ainsi, ils ont séquencé un génome entier, ainsi que des échantillons d'ARN provenant de 12 patients atteints de mélanomes uvéaux de différents types histologiques. Le but étant de mettre en évidence la présence ou l'absence de gènes différentiellement exprimés selon deux groupes : les sujets atteints du mélanome uvéal accompagné d'une mutation de SF3B1 et les sujets atteints sans mutation (WT). Il est défini dans l'article que les mélanomes peuvent être classifiés selon deux groupes : les mélanomes de classe I et ceux de classe II. Les premiers mélanomes ont tendance à être moins invasifs et ne se métastasent (multiplication cellulaire engendrant une tumeur) que rarement, tandis que le

second type se métastase plus fréquemment. Afin d'explorer les effets du mutant SF3B1 sur l'expression de l'ARN, l'équipe de Furney et al. ont analysé 3 patients mutants SF3B1 et 9 SF3B1- wild type (WT). Harbour et al ont quant à eux mené leurs travaux sur 3 échantillons mutants et 5 wild-types, tous présentant une tumeur de classe 1.
À l'issue de l'analyse des données RNA-seq, Harbour et al. n'ont pas montré de résultats probants concernant le lien entre la mutation SF3B1 et les erreurs d'épissage en dépit de son implication dans le processus d'épissage. Tandis que de leur côté Furney et al. observent une expression différentielle de certains gènes entre les individus dépourvus de mutation et les individus avec.
En particulier, dans leur papier Harbour et al. ont mis en évidence l'implication de BAP1 dans le mélanome et cherchent ainsi à identifier d'autres mutations pouvant éventuellement détenir un rôle dans le mélanome.
Harbour et al ont montré que les types de mélanomes divergent selon la mutation impliquée. Selon eux, les mutations apparaissant uniquement sur l'arginine-625 de la protéine SF3B1 et engendrent des mélanomes avec un meilleur pronostic vital pour le patient (mélanome dits de "bas grade"). Par ailleurs, il est montré qu'il n'y a pas de différence d'épissage alternatif des ARNm entre les deux groupes de patients. Ils ont également mis en évidence que l'ensemble des tumeurs arborent des mutations des protéines GNAQ (guanine nucleotide-binding protein G(q) subunit alpha) ou GNA11 (guanine nucleotide-binding protein subunit alpha-11). Ces deux paralogues forment ensemble la sous-unité alpha de la protéine Gq. En outre, certaines tumeurs sont aussi porteuses de mutations sur SF3B1 ou sur BAP1. Contrairement aux mutations Gq, ces deux dernières mutations sont secondaires et n'apparaissent jamais chez le même patient simultanément. Il est également mentionné qu'une mutation SF3B1 ou BAP1 peut apparaître après que la tumeur ait subi des mutations GNAQ ou GNA11.
Furney et al quant à eux ont mis en évidence le fait que les séquences codantes, tout comme les non codantes subissent des épissages alternatifs. Ainsi, les gènes codants pour les protéines BCC5, UQCC et CRNDE subiraient différents épissages alternatifs causant ainsi les mutations de SF3B1. Par ailleurs, leurs résultats montrent que dans 85% des cas environ les gènes GNAQ ou GNA11 (oncogènes moteurs du mélanome uvéal) subissent des mutations exclusives. De même, pour 85% des tumeurs métastatiques, on observe une mutation du suppresseur de tumeur BAP1 fortement liée à la propagation du cancer.
Dans l'étude bioinformatique qui suit, nous nous baserons ainsi sur le même jeu de données dans le but de comprendre les divergences se dessinant entre ces deux publications. Il sera question de mettre en évidence l'expression différentielle des gènes pour chacun des échantillons selon leur appartenance à un sujet SFB3-mutant ou à un wild-type.
Pour cela nous avons construit une pipeline d'analyses RNAseq sur les mêmes données, à savoir 8 échantillons d'ARNm de patients atteints de mélanome : les 5 individus wild-types et les 3 patients présentant une mutation du facteur d'épissage alternatif SF3B1.
Pour optimiser la reproductibilité de notre travail nous avons opté pour les outils Nextflow, Docker et Github. Nous avons ainsi choisi de travailler avec l'outil Nextflow pour le management de notre workflow et Docker pour l'écriture des différents containers utiles pour nos process Nextflow. L'outil Github s'est avéré très utile pour le partage et le versionnage de nos scripts.

2 Organisation de l'équipe

Dans cette partie, nous expliquons la manière par laquelle nous avons organisé notre travail. Dans un premier temps, nous décrivons la répartition des tâches et les outils de partage utilisés, puis dans un second temps, la répartition de la rédaction du compte-rendu du projet et enfin, nous listons les problèmes que nous avons rencontrés au cours du projet.

2.1 Répartition des tâches

Tout d'abord, chaque membre de notre groupe a commencé par lire séparément les deux publications sur lequel se base ce projet. Cela nous a permis de contextualiser les problématiques, de comprendre les études, ainsi que les résultats mis en évidence dans ces deux articles. Nous en avons ensuite discuté ensemble et nous nous sommes mises d'accord sur les points essentiels de compréhension de cette lecture des articles. Nous avons ainsi pu commencer à mettre en évidence les enjeux et les problématiques du projet.
Étant donné qu'il était préférable de travailler avec une seule machine virtuelle par groupe, nous nous sommes organisées pour travailler en présentiel deux fois par semaine, avec tous les membres de l'équipe. Dans un premier temps, ce travail en présentiel avait pour but le téléchargement des fichiers SRR. Mais au vu du temps important consacré au téléchargement de ces fichiers bruts, nous avons décidé de changer de stratégie pour la répartition des tâches.
Nous avons donc décidé de travailler en binôme pour optimiser notre efficacité : Ilhem et Fatma se sont ainsi concentrées sur le code Nextflow, tandis que Sanâ et Lorraine ont travaillé sur la partie statistique. Pour cela, nous avons organisé deux réunions hebdomadaires en présentiel ou visio-conférence pour pouvoir informer tous les membres du groupe de l'avancement. De plus, ces réunions nous permettaient de discuter pour soulever des questions ou suggérer des modifications et clarifications sur certains points. A la fin de chaque rencontre, nous faisions un résumé de l'état actuel d'avancement du projet et nous organisions les étapes et réunions suivantes. Cette manière de travailler nous a permis de gagner en efficacité. En outre, chaque membre a été amené à comprendre l'intégralité du projet, à savoir : le code Nextflow, la partie statistique et la démarche d'analyse des données biologiques.
Pour préparer nos réunions hebdomadaires avec nos encadrants, chaque semaine l'une des membres du groupe se chargeait du diaporama. Ce dernier comportait les scripts de nos codes et les éventuelles erreurs renvoyées. Chacune de nous commentait ensuite les diapositives qui regroupent toutes nos questions. Lors des rendez-vous, deux personnes prenaient la main sur la présentation et deux autres membres notaient les remarques et les conseils de l'enseignant. Celles-ci rédigeaient ensuite un compte-rendu partagé, pour garder une trace de tout ce qui avait été dit et surtout les réponses précises à nos questions. Ces rendez-vous se sont révélés d'une grande aide pour la compréhension du contexte et des objectifs du projet et nous ont également permis de débloquer certains points non fonctionnels des scripts.
La nouvelle stratégie d'organisation du travail, nous a ainsi permis d'optimiser notre efficacité dans l'avancement du projet. Tous les membres du groupe ont ainsi pu à la fois travailler sur l'écriture du code Nextflow, mais également sur l'analyse statistique, du fait de ce gain de temps.
Finalement, nous avons testé notre code sur une machine virtuelle vierge, lors de notre dernière réunion de groupe, afin d'apprécier la reproductibilité de notre code sur n'importe quelle machine virtuelle.

2.2 Outil de partage du travail et des scripts

Notre groupe s'est organisé de manière à ce que chaque membre du groupe participe dans toutes les étapes du projet. Effectivement, nous avons souvent travaillé par binôme dans les étapes de recherches d'informations et de compréhension des objectifs attendus, afin d'optimiser notre efficacité dans l'avancement du projet. Mais étant donné que nous travaillions fréquemment en présentiel sur une seule machine, nous avons pu communiquer et collaborer activement lors des étapes d'écriture et d'exécution de notre workflow. Nous nous sommes davantage servis de github pour pusher la partie statistique. En effet, étant donné les difficultés que nous avons rencontré pour l'écriture du workflow, nous n'avons pu le déposer sur le github, qu'une fois le code fonctionnel. Ainsi pour la partie statistique nous avons utilisé le versionnage sur git et chacune a pu déposer ou mettre à jour le code. Cela nous a permis au final de suivre l'avancement au fur et à mesure.
Nous avons également utilisé Kanbanchi pour organiser notre gestion des tâches pour ce projet. Cet outil est doté de tableaux Kanban et des listes de tâches, qui permettent de visualiser en temps réel le flux de travail des étapes de notre projet. De plus, nous avons utilisé LaTex pour rédiger notre rapport écrit, afin d'obtenir une présentation plus professionnelle. Ce document était partagé de manière collaborative, pour observer la progression en temps réel des différentes parties du rapport.

2.3 Répartition du compte-rendu

La rédaction du compte rendu s'est faite dans l'optique que chaque membre du groupe maîtrise l'ensemble des étapes du projet. Ainsi, nous avons réparti les rôles de manière à ce que chacune rédige la ou les parties sur lesquelles elle n'a pas directement participé (lors des processus de recherche d'informations et compréhension des méthodes). Ainsi les réunions hebdomadaires nous ont permis d'être constamment à jour sur chacun des points du projet et donc de maîtriser et comprendre tous les process du workflow. A terme nous étions ainsi en mesure de comprendre et d'expliciter chacune des étapes. Ainsi, Sanâ et Lorraine se sont penchées sur la rédaction relative au code Nextflow et Ilhem et Fatma se sont concentrées sur la rédaction de la partie statistique. Plus particulièrement, Sanâ a rédigé la partie introductive ainsi que les explications du process mapping, indexation du mapping et de la matrice de comptage des reads. Lorraine s'est chargée de la partie relative à la configuration du workflow au docking, au téléchargement des données et à l'indexation du génome. Ilhem s'est ensuite occupée de la partie relative aux analyses statistiques sur R et Fatma s'est penchée sur la partie concernant les résultats des analyses statistiques. Par ailleurs, Lorraine et Sanâ se sont chargées des résultats relatifs au code workflow.
Nous avons ensuite réfléchi à la discussion et à la conclusion de notre rapport ensemble en mettant en commun, nos résultats et idées afin qu'elle soit la plus complète possible.
Enfin, pour la partie organisation du projet, chacune de nous s'est attelée à un paragraphe : Lorraine pour la partie répartition du projet, Ilhem pour les outils de partage, Sanâ pour la répartition du compte rendu et Fatma pour les problèmes rencontrés.

2.4 Problèmes rencontrés

Nous avons rencontré plusieurs difficultés pendant l'élaboration de notre projet, parmi elles, le téléchargement du fichier SRA062359 des données brutes RNAseq. En effet, nous avons été bloquées pendant une période de 3 semaines sur celui-ci. Nous avons d'abord rencontré des problèmes de capacité de stockage car ces fichiers sont volumineux. Nous avons alors changé plusieurs fois la configuration de notre machine virtuelle (Virtual Machine ou VM), tout en veillant à terminer le déploiement des VM précédentes. Nous avons également essayé de télécharger les fichiers à partir du site de l'ENA (European Nucleotide Archive), en téléchargeant à la main chaque paire de fichiers

fastq, ce qui prenait un temps considérable. Puis, fastq-dump étant la méthode la plus optimisée en temps de calcul, qui permet d'obtenir les fichiers fastq zippés (fastq.gz), nous avons essayé d'utiliser uniquement cet outil pour le téléchargement des données brutes, sans succès. Finalement, nous avons décidé d'utiliser la méthode "wget" d'abord pour récupérer les fichiers SRR sur le site du NCBI (National Center for Biotechnology Information), puis nous avons compressé les fichiers à l'aide de fastq-dump.
Nous avons également rencontré des problèmes de mémoire, ainsi que lors du choix du nombre CPUs (ou Central Processing Unit, le nombre d'unités de calculs) alloués ; en effet, les process d'indexation et de mapping sont relativement longs, et coûteux en mémoire. Nous avons ainsi fait face à des interruptions d'exécution de notre workflow, en raison d'une configuration insuffisante de capacité en CPUs ou de mémoire pour notre machine virtuelle et incompatible avec les paramètres requis pour notre code nextflow.

3 Matériels et méthodes

Pour pouvoir exécuter notre workflow, nous avons besoin de quelques prérequis tels qu'un compte IFB Cloud, un compte GitHub et un IDE (environnement de développement intégré) pour éditer les scripts comme mousepad.

"La biosphère est utilisée pour la production scientifique dans les sciences du vivant développements, et pour soutenir des événements tels que des sessions de formation cloud et scientifiques, des hackathons ou des ateliers". Nous nous sommes connectées en utilisant nos comptes de l'université Paris Saclay et nous avons récupéré et ajouté la public key à nos comptes IFB Cloud, pour avoir accès au SSH (Secure Shell, pour la connexion à distance). Ensuite, nous avons lancé la VM nommée "BioPipes" à partir de l'onglet RainBio, en choisissant les paramètres par défaut suivants : le groupe : ReproHack2021, le cloud : ifb-core-cloud et le gabarit d'image cloud : ifb.m4.large. Cette étape nous a permis d'obtenir l'adresse IP de la machine " : xxx.xxx.xxx.xxx". La connexion à cette machine virtuelle se fait en local, dans un terminal en copiant son adresse IP avec la commande "ssh -Y ubuntu@xxx.xxx.xxx.xxx".

3.1 Configuration du workflow

Nous avons tout d'abord choisi par défaut une machine virtuelle "ifb.m4.large" avec une configuration comme suit : 2 vCPU, 8Go GB RAM, 120 Go GB local disk. Cependant, nous nous sommes vite rendu compte que ces paramètres n'étaient pas suffisants pour exécuter notre workflow car nos process s'arrêtaient à cause du manque de ressources. Nous avons ensuite estimé la capacité de stockage nécessaire pour le téléchargement de nos fichiers RNAseq (SRA062359) et notamment les process d'indexation du génome ou de mapping, après avoir lancé notre workflow avec des VM de configurations différentes. Les fichiers SRA comprennent huit échantillons paired-end d'environ 8 Go chacun, c'est-à-dire environ 64 Go au total pour ces données brutes. Cependant, afin de réduire la capacité de stockage, ces données brutes sont conservées dans un dossier zippé (il en va de même pour les données du génome humain).
Les étapes suivantes concernent les process d'indexation du génome et de mapping. Ces process requièrent un nombre important de ressources et de temps de calcul. Par conséquent, nous avons choisi de paramétrer notre process d'indexation avec 8 cpus et celui de mapping à 4 cpus. Nous avons aussi alloué 30 GB de mémoire, afin d'augmenter le temps de calcul en exécutant les différents process en parallèle. En tenant compte de l'ensemble des conditions, nous avons finalement choisi de travailler dans le répertoire "/mnt/mylocaldata" en utilisant une VM "ifb.m4.2xlarge" de

configuration 8 vCPU, 32 Go GB RAM, 200 Go GB local disk. Cela nous permettrait ainsi d'exécuter notre workflow sans interruption pour manque de capacité de stockage.
Ces informations sur les paramètres choisis pour nos process sont présents sur github dans le fichier nommé "nextflow.config" et contient toutes les informations relatives aux configurations de notre workflow. Ce fichier indique également que nous avons utilisé des containers Docker, développés par "evolbioinfo" pour nos différents process (cf 3.2) Les différents process du workflow). Nous avons utilisé les versions suivantes de ces différents outils : sratoolkit :v2.10.8, star :v2.7.6a, samtools :v1.11, subread :v2.0.1 et deseq2 :1.28.1. La dernière information présente dans ce fichier est la clé API (Application Programming Interface) du NCBI (National Center for Biotechnology Information) qui permet d'accéder au site du NCBI pour télécharger les données RNAseq.

3.2 Les différents process du workflow

1-Les Dockers :

Le Docker est un outil qui va nous permettre de créer, de déployer et de lancer nos applications en utilisant des containers (ou conteneurs). La mise en place de ces containers, necessite de créer des images Dockers qui vont permettre de configurer tout l'environnement dans lequel le container va s'exécuter. L'utilisateur pourra ainsi exécuter le pipeline avec les versions exactes des packages utilisés sans necessiter de packages supplémentaires.
Dans notre workflow, nous avons utilisé différents conteneurs, développés par le groupe "evolbioinfo" [5]. Le premier "evolbioinfo/sratoolkit :v2.10.8" nous a notamment permis de compresser nos fichiers. noter que nous avons utilisé cette version de sratoolkit, plus récente que la version v2.5.7 suggérée dans l'énoncé du projet. En effet, cette ancienne version n'est aujourd'hui plus fonctionelle. Ensuite, "evolbioinfo/star :v2.7.6a" était nécessaire pour faire les étapes d'indexation du génome et de mapping. Egalement, le conteneur "evolbioinfo/samtools :v1.11" a été utilisé pour la génération des fichiers Bam et enfin "evolbioinfo/subread :v2.0.1" pour la matrice des comptage.
Par ailleurs, pour la partie analyses statistiques, nous avons créé une image d'un docker "fatmaammar/test14" avec la commande "docker build -t fatmaammar/test14" à partir d'un DockerFile. En effet cette dernière nous a permis de faire une mise à jour sur R-base avec la version 4.1.2 et d'obtenir tous les packages necessaire au bon fonctionnement de notre script (dplyr, ggrepel, BiocManager, pheatmap, RColorBrewer, tidyverse, EnhancedVolcano, factoextra, DEGreport, apeglm, FactoMineR).
Cependant, nous avons rencontré un probléme au niveau de la containerisation du package tidyverse, nous avons alors opté pour la commande "install.packages "tidyverse" dans notre code R .
Notre workflow est composé de dix process qui s'exécutent en parallèle, en communiquant à l'aide de plusieurs channels. (voir Annexes, figure 8 : Schéma du workflow complet)

2-Téléchargement des données brutes et du génome humain de référence :

Le premier process de notre workflow se nomme "DownloadSRR" et correspond à l'étape de téléchargement des données brutes RNASeq. Ces données ont pour numéro d'accession SRA062359 (SRP017413), au sein de la base du NCBI Sequence Read Archive (SRA). Cette archive comprend huit fichiers SRR (pour Sequence Read Runs, de SRR628582 à SRR628589) qui correspondent aux échantillons qui constituent les reads paired-end. Ces échantillons sont le résultat du séquençage de fragments RNAseq. Nous avons ainsi récupéré les identifiants des huit fichiers SRR en input sur le site de téléchargement du NCBI SRA, à l'aide de la méthode "wget" et de la clé API NCBI "fdf41c51b459ab4f389d6ddcab2199c87508" définie dans le fichier nextflow.config et génére selon les étapes décrite dans ce lien [4]. En outre, nous avons indiqué les valeurs des identifiants SRR correspondant aux données à analyser via la communication par un channel nommé file_SRP (Sequence Read Project). Les fichiers "SRRxxxxxx.sra" ainsi obtenus en output sont stockés dans un dossier

SRR, appartenant au répertoire “results” qui contient tous les résultats de notre pipeline. Comme il a été dit précédemment, les fichiers SRA sont volumineux, ce qui explique la nécessité de compresser ces fichiers pour les analyses et les manipulations suivantes. Nous avons choisi de séparer les étapes de téléchargement et de compression des fichiers en deux process pour permettre à ces deux process de s’exécuter en parallèle.

Le deuxième process est ainsi nommé “fastqZipping”. Celui-ci récupère en entrée les valeurs des identifiants SRA depuis les fichiers SRR et stocke ces valeurs dans la variable “val(sraid)” de type tuple. Nous obtenons ainsi en output des fichiers fastq compressés : “SRRxxxxxx_1.fastq.gz” et “SRRxxxxxx_2.fastq.gz”, à l’aide de la commande “fastq-dump –gzip –split-files”. De plus, chaque ligne des fichiers SRR commence par un symbole “@” avec le nom des reads, leur séquence, la longueur de la séquence et les informations relatives à la qualité du fichier (différents scores, score PHRED, estimation de la probabilité d’erreur Pe, etc.). Par ailleurs, l’outil fastq-dump fait appel au container sratoolkit qui est précisé dans le fichier nextflow.config. En effet, chaque échantillon SRR fonctionne en paire de deux reads (paired-end), que nous avons respectivement défini par “1” et “2”. Chaque paire de reads correspond au début et à la fin de la séquence du même fragment ARN.
Ensuite, le troisième et quatrième process correspondent au téléchargement des chromosomes. Ils représentent les données du génome humain de référence, qui permettent d’indexer le génome, pour le process de mapping des reads. Dans un premier temps, nous avons le process “chromosomedownloading” qui récupère les données brutes provenant du séquençage d’un génome humain de référence consensus “Homo sapiens GRCh38” sur le site ensembl (base de données répertoriant les génomes de vertébrés) à l’aide de la méthode “wget”. De plus, un channel appelé “chromosome” permet de communiquer avec le site ensembl pour récupérer les noms des 22 chromosomes, l’ADN mitochondrial (MT) et les chromosomes sexuels X et Y, afin d’obtenir tous les chromosomes qui correspondent au génome humain, en input. Cette communication se fait par un protocole ftp ou File Transfert Protocole, un protocole de communication de partage de fichiers sur un réseau. Nous obtenons ainsi en output les fichiers des chromosomes compressés “x.fa.gz”.
Dans un deuxième temps, nous avons le process “CompleteGenome” qui collecte en entrée les fichiers des chromosomes, compressés par la commande gunzip. Ces chromosomes sont ensuite concaténés dans un fichier fasta de sortie nommé “HumanGenome.fa”

3-L’indexation du génome :

Le cinquième process correspond ensuite à “genomeIndexation”. Il permet ainsi l’indexation du génome complet necessaires aux étapes suivantes, à savoir le mapping des reads sur le génome complet. Ce process prend en entrée le génome humain généré par le process précédent et crée un fichier avec le génome indexé en sortie. Nous utilisons pour cela le container STAR avec comme paramètres : runThreadN égal à 8 CPUs, l’option runmode utilise la commande genomeGenerate pour générer les index du génome à partir d’un génome de référence, l’option genomeDir définit le répertoire “ref” comme répertoire de sortie et l’option genomeFastaFiles spécifie que le fichier fasta à prendre en input est le génome humain.
Le process suivant est appelé “HumanGenomeAnnotation”. Il s’agit du process qui permet de récupérer le fichier tabulé GTF (General Transfert Format) / GFF (General Feature Format) comportant notamment des informations et annotations sur les séquences, les gènes et les exons. Plus précisément, nous pouvons retrouver dans ce fichier, des informations ordonnées selon l’ordre suivant [6] : le nom de la séquence des gènes (seqname), la provenance des données (source), le type de l’élément (feature), la position de départ de la séquence (start), la position terminale (end), le score de confiance de l’annotation (score), le brin codant ou non (strand), la phase de la séquence (frame) ainsi que d’autres informations complémentaires (attribute). Ce fichier gtf.gz compressé et annoté du génome humain de référence est récupéré par une méthode “wget” depuis le site ensembl et à

nouveau à l'aide d'un protocole ftp. L'output de ce process est donc un fichier "HumanGenome.gtf" décompressé avec une commande gunzip.

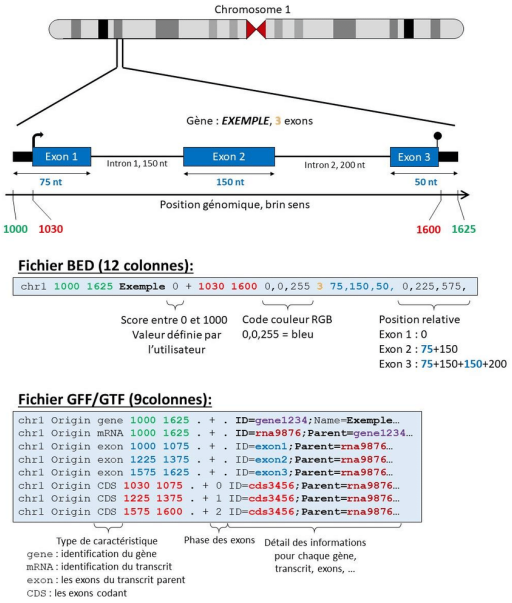


FIGURE 1 – Représentation d'un génome humain [7]

4-Mapping :

L'étape de mapping consiste en l'alignement des séquences d'ARN sur le génome humain précédemment indexé. Autrement dit, ce sont les reads RNAseq paired-end au format fastq.gz que l'on cherche ici à mapper. Ce processus se base principalement sur l'utilisation de l'outil STAR (Spliced Transcripts Alignment to a Reference).
STAR est une méthode rapide et performante pour aligner les reads. L'outil a été conçu dans l'optique de permettre la découpe des read en différents fragments, cela permet la découverte de nouvelles jonctions d'épissage et donc de nouveaux transcrits. Pour chaque read qu'il aligne, STAR va chercher à trouver la plus longue séquence matchant parfaitement avec un ou plusieurs loci sur le génome de référence. Les séquences les plus longues, appelées MMP (Maximal Mappable Prefixes) sont définies par STAR en prenant en compte la séquence du read, la position dans le génome et la séquence du génome. Les MMP sont ensuite regroupées et un score d'alignement est calculé (en tenant compte à la fois du nombre de MMPs, de la distance entre eux et du nombre de disparités entre chaque MMP et le génome de référence). Une seconde stratégie d'alignement de STAR consiste à renseigner plusieurs échantillons en entrée. Ainsi, les jonctions d'épissage détectées sont partagées et supportées à travers les alignements, ce qui permet d'augmenter le nombre de nouvelles jonctions d'épissage détectées.
Dans ce septième process**"mappingfq"**, on impose ainsi en entrée les 16 fichiers paired end et on renseigne le chemin où se trouve le fichier du génome indexé. Suite à l'alignement, les séquences et coordonnées des reads sont enregistrées et triées dans un fichier BAM de sortie de type "SRRxxxxxx.bam", ce dernier étant l'élément clé pour la suite des analyses des données RNA-seq. Le format BAM correspond à une version compressée et binaire du format SAM.
Ainsi les fichiers SAM sont des fichiers texte contenant les mêmes informations que les BAM. Les colonnes sont séparées par des tabulations. L'en-tête des fichiers est identifiée par le symbole « @ ». Pour chaque read, 11 champs doivent être renseignés. On y retrouve en particulier l'identifiant du read, la première position de la séquence, la taille de celle-ci, le contenu de la séquence, mais également la position exacte sur le chromosome où s'est fait l'alignement et les scores de qualité

PHRED (provenant du programme Phred d'automatisation du séquençage d'ADN).
Le fichier contient également le flag (c'est-à-dire un ensemble de valeurs comprises entre 1 et 2048 correspondant aux annotations concernant l'alignement pour chaque chromosome). ainsi que des précisions concernant la qualité du mapping, l'information CIGAR (indiquant l'état de l'alignement de la séquence en termes de présence de match ou mismatch, d'insertion/délétion, de gap, mais également si on retrouve des phénomènes de soft ou hard clip ou encore du padding), les informations paired-end, la séquence nucléotidique, la qualité des bases et sur d'autres tags optionnels.
Ainsi par le biais des informations précédemment citées il est facile d'observer les évènements d'insertions de délétion ainsi que les différents match entre nos reads et le génome de référence. Par ailleurs, l'alignement nous permet de repérer si le gène est issu d'épissage (alternatif). En effet, si l'on observe que les reads sont fragmentés et que leur alignement est interrompu, cela laisse penser à la présence de potentiels introns.
L'ensemble de ces informations est primordial pour notre analyse dans la mesure où elles permettent de compléter les informations obtenues dans nos fichiers gff.

5-L'indexation du mapping :

Par la suite, nous avons le process **"generating_bam_files"** qui prend en input les fichiers BAM (bam1). Ces fichiers sont de nature binaire et sont triés par coordonnées suite au traitement par l'outil STAR. Dans cette étape, nous utilisons le container "samtools" et la commande samtools index pour trier chaque read par leur position sur le chromosome associé. Nous obtenons ainsi en output des fichiers
"SRRxxxxxx.bai", dans le répertoire bam_files, qui peuvent être interrogés facilement lors d'étape de recherches, de sélection de reads à une certaine position donnée ou encore pour la visualisation des séquences avec l'outil IGV (Integrative Genomics Viewer).
À noter que les variables bam1 et bam2 prennent toutes les deux en entrée les fichiers BAM, utilisés respectivement pour le process d'indexation et le process de création d'une matrice de comptage car on ne peut pas utiliser le même input dans 2 channels.

6-Matrice de comptage des reads :

Le dernier et neuvième process est le "counting_Reads_Matrix" qui prend en entrée les fichiers bam indexés (variable bam2) par position ainsi que le fichier gtf du génome humain annoté, afin d'établir une grande matrice de tableau de comptage. En effet, nous avons fait correspondre ces deux types de fichiers grâce à leur index. Nous utilisons pour cela, le container Subread qui permet de réaliser l'alignement de reads RNAseq et l'outil featureCounts qui permet de compter le nombre de reads mappés dans chacun de nos huit échantillons et donc le niveau d'expression des gènes. Le fichier en sortie est un fichier texte nommé "featureCounts_Matrix.txt" appartenant à un dossier "featureCounts". Le fichier ainsi obtenu a pour objectif final d'analyser la quantification de l'expression de chacun des gènes mutés ou contrôle, en nombre de reads, à l'aide d'outils d'analyses statistiques.

7-Analyses statistiques :

Pour faire ces analyses et tests statistiques, nous utilisons une image docker que nous avons créé sur dockerhub appelée "fatmaammar/test14" qui contient notamment DESeq2 et des packages pour tracer différents plots sur le logiciel R. Le process est nommé **"stat_analysis"** et prend en entrée le fichier metaData.txt à l'aide d'un channel, les comptages des reads (count_Data) et donne en sortie des fichiers "DE_gene.txt" et différents plots en utilisant le script stat.analysis.R qui doit être placer dans un répertoire "bin" au sein du répertoire mydatalocal.

4 Méthodes des analyses statistiques (DESeq2)

Cette étape consiste à identifier les gènes présentant des différences de moyenne d'expression entre les WT et mutés (SF3b1-mutation) parmi les 60612 gènes obtenus à l'issu du process "counting_Reads_Matrix". Pour déterminer les niveaux d'expression des gènes, nous avons généré un script R qui utilise principalement les packages suivants DESeq2, FactoMineR, factoextra, EnhancedVolcano.

Nous avons suivi les étapes détaillées dans l'image ci-dessous [8] :

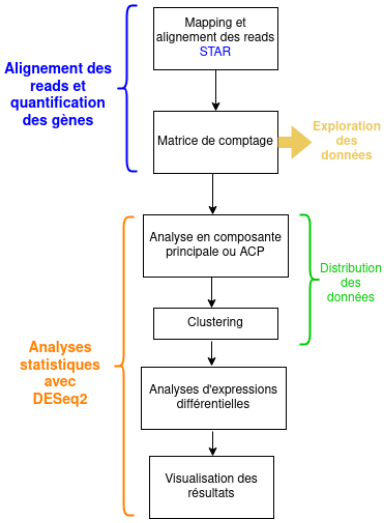


FIGURE 2 – Les étapes de la partie statistique

4.1 Exploration des données

La distribution des données :

Dans le but de comprendre la façon dont les comptages d'ARN-seq sont distribués, nous avons tout d'abord tracé les comptages pour l'échantillon SRR628582 à l'aide de la fonction ggplot.
En effet les nombres de reads sont des entiers positifs, les lois de probabilités discrètes (loi binomiale et loi de poisson) apparaissent ainsi comme des modélisations de choix pour représenter ces données.
Avec les données RNA-Seq, un très grand nombre d'ARN sont représentés et la probabilité d'extraire un transcrit particulier est très faible. Ainsi, il serait approprié d'utiliser la distribution de Poisson. Cependant, une propriété unique de cette distribution est que la moyenne doit être égale à la variance. Ainsi, étant donné que les échantillons séquencés sont issus de patients différents il existe donc une certaine variation du nombre de reads, et la distribution de Poisson ne parvient plus à la modéliser, en particulier pour les gènes fortement exprimés. La distribution binomiale négative s'avère alors plus adaptée pour modéliser ce type de données.
Pour cela nous avons ensuite tracé un ggplot qui modélise les moyennes et les variances de nombre de reads avec un ajustement par la distribution binomiale négative.
Les résultats de cette étape (les 2 ggplot) apparaissent en annexes (cf Annexes, figure 10 et 11).

4.2 Evaluation de la qualité

a-ACP :

Cette étape consiste à évaluer la similitude globale entre les échantillons, autrement dit déterminer parmi les 8 échantillons dont on dispose, quels sont ceux similaires les uns aux autres, et quels sont ceux qui diffèrent. Pour explorer les similitudes entre nos échantillons nous avons effectué un contrôle qualité au niveau de l'échantillon à l'aide de l'analyse en composantes principales (ACP) et des méthodes de regroupement hiérarchique. En effet l'ACP est une méthode d'analyse des données qui se base sur la statistique multivariée. Elle consiste à transformer des variables corrélées en nouvelles variables décorrélées. Ces nouvelles variables sont nommées composantes principales. Cette méthode nous permet de visualiser la distribution et la similitude des 8 échantillons dans l'espace des gènes. Pour ce faire, nous utilisons la fonction PCA des packages FactorMineR et factoextra. Cette fonction génère un graphe de représentation des individus qui permet de voir à quel point la variation des gènes est similaire entre les échantillons.

b-Classification Hiérarchique :

Afin de bien visualiser la corrélation entre les 8 échantillons et déterminer la présence ou l'absence d'un ou plusieurs échantillons aberrants, nous utilisons la méthode de clustering hiérarchique.
Cette méthode indique quels échantillons sont les plus similaires en calculant la corrélation de l'expression des gènes normalisés pour toutes les combinaisons de paires d'échantillons dans l'ensemble des données. Comme la majorité des gènes ne sont pas exprimés de manière différentielle, les échantillons ont généralement des corrélations élevées entre eux (valeurs supérieures à 0,80). Les échantillons inférieurs à 0,80 peuvent indiquer une valeur aberrante dans les données et/ou une contamination de l'échantillon. Pour cela nous avons utilisé la fonction "cor" qui calcule la corrélation de pearson par paire d'échantillon et nous avons représenté les résultats sous forme de Heatmap en utilisant le package "heatmap".

4.3 Analyse d'expression différentielle

Dans cette étape, nous voulons essentiellement déterminer si les niveaux d'expression moyens des gènes des patients WT sont significativement différents par rapport aux patients SFB31 mutés et extraire les 10 TOP gènes qui sont différentiellement exprimés. Pour ce faire, nous utilisons le package "DESeq2".

Comme indiqué dans la figure ci-dessous, l'analyse différentielle avec DESeq2 implique plusieurs étapes.

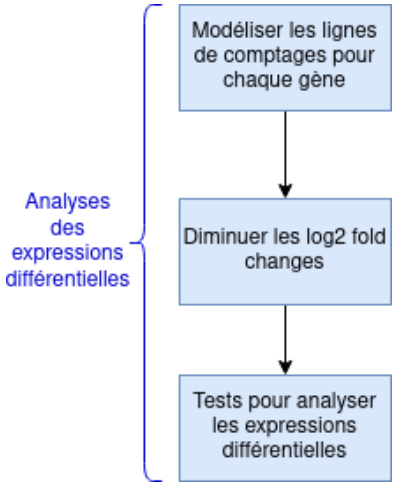


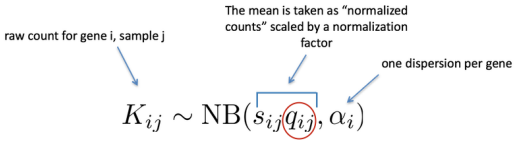
FIGURE 3 – Analyse des expressions différentielles

DESeq modélise les comptages bruts en utilisant des facteurs de normalisation (facteur de taille) pour tenir compte des différences de la profondeur de séquençage (nombre de reads) par échantillon. Ensuite, il estime les dispersions génétiques et réduira ces estimations pour générer des estimations plus précises de la dispersion et modéliser les comptages. Enfin, DESeq2 s'adapte au modèle binomial négatif et effectue des tests d'hypothèses à l'aide du test de Wald .
Pour obtenir nos résultats d'expression différentielle à partir de nos données de comptages brutes, nous avons tout d'abord créé l'objet DESeqDataSet à l'aide de la fonction DESeqDataSetFromMatrix qui prend comme Input :

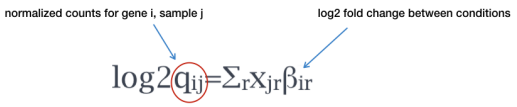
- **La matrice de comptage :** cette matrice doit contenir pour chaque échantillon SRR le comptage du nombre de reads par gène. Pour avoir ce format, nous avons adapté le fichier featureCounts_Matrix.txt issu du process "counting_Reads_Matrix" en enlevant les colonnes inutiles et en gardant uniquement en colonnes les noms des 8 échantillons et en lignes les gènes.
- **Un tableau décrivant le plan d'expérience :**Ce tableau contient les informations sur les 8 échantillons, il associe à chaque échantillon le statut muté ou non pour le gène SF3B1. Nous avons généré à partir de ce tableau un objet dataframe contenant en colonne les identifiants des échantillons et leurs statuts (WT ou muté). Le statut "WT" ou "M" (respectivement pour Wild-type et muté) représente le critère selon lequel l'analyse différentielle est réalisée.
- **Un schéma pour l'analyse différentielle :** On spécifie ici la colonne mutation qui contient la variable à 2 niveaux (WT ou muté) selon lequel DESeq devra effectuer l'analyse différentielle. Il s'agit du paramètre design dans la fonction **DESeqDataSetFromMatrix**.

Par la suite, nous avons analysé l'objet DESeqDataSet par la fonction DESeq. En effet, cette fonction normalise les données de comptage selon la méthode de la médiane des ratios.
Cette méthode se base sur le calcul d'un facteur de normalisation (facteur taille) pour chaque échantillon. Ce facteur correspond à la valeur médiane de tous les ratios pour un échantillon donné. Chaque valeur de la matrice du comptage est divisée par ce facteur afin de générer des valeurs de comptage normalisées. Cette normalisation a pour but de contrer les déséquilibres de la sous/surexpression génique ainsi que de faire face autco grand nombre de gènes exprimés de manière différentielle en tenant compte de la profondeur de séquençage et de la composition de l'ARN de l'échantillon.

Cette étape de normalisation est automatiquement effectuée par la fonction DESeq et les résultats sont accessibles par l'intermédiaire de la fonction results.
La dernière étape de flux de travail DESeq2 consiste à ajuster le modèle binomial négatif pour chaque gène et à effectuer des tests d'expression différentielle.
Comme expliqué dans l'étape de la distribution, les données RNA-seq présentent une surdispersion (Variance>moyenne). Pour contrer cette surdispersion, DESeq2 ajuste les données de comptage normalisées au modèle binomiale négative selon l'équation suivante :



Une fois le modèle ajusté, les coefficients ainsi que leur erreur-type sont estimés pour chaque groupe d'échantillons à l'aide de la formule :



Le log2 fold change est le ratio obtenu par le niveau d'expression d'un gène chez les sujets mutés divisé par le niveau d'expression de ce même gène chez les patients wild-type. En revanche, les estimations du log2 fold change (LFC) ne tiennent pas compte de la surdispersion discutée précédemment. Pour éviter ce biais, le log2 fold change calculé par le modèle doit être ajusté.
D'autre part, l'identification des gènes différentiellement exprimés entre les 2 groupes de patients (WT et M) se base sur le test de Wald utilisé par le package DESeq2 avec les hypothèses suivantes :

H0 : Absence de différences significatives d'expression différentielle entre les patients WT et les patients ayant une SF3B1 mutation. (LFC==0)
H1 : Présence de différences significatives d'expression différentielle entre les patients WT et les patients ayant une SF3B1 mutation.

Ainsi la p-value du test est calculée pour chaque gène dans chaque échantillon. Néanmoins cette p value est sensible aux faux positifs. Plus le nombre de gènes à tester est grand et plus le nombre de faux positifs le sera également. Pour cela, DESeq corrige les p-value par p_adj (p-value ajustée) qui est moins sensible aux faux positifs et permet par conséquent de déterminer les gènes significatifs.
Pour extraire les gènes significativement exprimés, nous avons tout d'abord fixé un seuil pour la p_adjust=0.05. Mais nous nous sommes rendues compte que ce seuil, à lui seul ne semble pas réduire le nombre de gènes importants. Par ailleurs, du fait du grand nombre de gènes significatifs dont on dispose, il peut être difficile d'extraire une pertinence biologique significative. Pour pouvoir les extraire, nous avons décidé de fixer un seuil égal à 0.5 pour le LFC. Ce seuil est basé sur le fait que l'on travaille en log2 donc le foldchange est en réalité égal à 2. Un fold change de 2 signifie que le niveau d'expression du gène est 2 fois plus important chez les sujets mutés que chez les sujets wild-types.

4.4 Visualisation des résultats

Pour visualiser nos résultats, nous avons choisi d'utiliser le package R "EnhancedVolcano" permettant de construire un volcano plot. Les seuils de la p-value ajustée et de la LFC ont été également fixés sur le volcano plot. Ainsi, nous avons pu représenté les 10 TOP gènes différentiellement exprimés sur ce graphe.

5 Résultats

5.1 Exécution du workflow

Comme il a été expliqué précédemment dans la section "matériels et méthodes", nous avons exécuté notre pipeline sur une machine virtuelle appartenant à Biopipes du serveur de l'IFB cloud, dans un terminal Linux et sur MobaXterm sous Windows, via une commande ssh. Sur une VM qui contient les outils Nextflow et Docker, il faut tout d'abord cloner notre répertoire github avec la commande "git clone git@github.com :Ilhem36/Hackathon-reproductible-AMI2B.git". Il faut ensuite se placer dans le répertoire hack_git, pour récupérer les fichiers main.nf, nextflow.config, metaData.txt, ainsi que le fichier stat_dernier.R, qui se trouve dans un sous-répertoire stat. Nous y trouvons également des fichiers complémentaires comme le Dockerfile, le readme.txt et un sous-répertoire resultats, qui contient le fichier featureCounts_Matrix.txt (fichier avec les comptages des gènes différentiellement exprimés) et les différents résultats sous forme de représentations graphiques.

Ensuite, il est nécessaire de se placer dans un répertoire de travail avec une capacité de travail suffisamment grande pour exécuter notre workflow, avec la commande "cd path/to/your/repertory') et de créer un répertoire stat pour y placer le code stat_dernier.R. Lorsque toutes ces étapes sont mises en place, il est alors possible de lancer notre pipeline en activant nextflow par la commande "conda activate nextflow" puis en utilisant la commande "nextflow run main.nf".
Les dix process de ce workflow s'exécutent en parallèle avec un temps de calcul total d'environ 3h (cf Annexes, figure 9). Le temps de calcul est relativement court (quelques secondes, voire minutes) pour les étapes de récupération des données brutes : "DownloadSRR", "chromosomedownloading", "CompleteGenome", 'HumanGenomeAnnotation". L'étape de compression des fichiers SRR "fastq-zipping" a un temps de calcul d'environ quarante minutes. Puis, les process "GenomeIndexation", "mappingfq" ont un temps de calcul de deux heures. Ce sont des process longs, qui manipulent des fichiers volumineux, c'est pourquoi ces deux process ont besoin respectivement de 8 cpus et 4 cpus ainsi qu'une capacité de mémoire au minimum de 30 Go, afin d'optimiser la vitesse de calcul. Ensuite, le process suivant generating_bam_files est rapide mais celui-ci prend en entrée les fichiers de sortie du process de mapping. Le process counting_Reads_Matrix a un temps de calcul d'environ quinze minutes. Tandis que le process stat_analysis a un temps de calcul d'environ onze minutes.
Par conséquent, le temps de calcul de notre pipeline reste relativement long. Cela s'explique en partie par la nature des fichiers RNAseq et les process coûteux en temps de calcul que nous manipulons dans ce projet. L'utilisation d'un workflow permet donc d'optimiser ce temps des différents process en permettant leur exécution en parallèle. Cependant d'autres améliorations peuvent également être envisagées. Par exemple, si nous avions pris une VM avec une capacité plus importante, nous aurions pu allouer un nombre de CPUs et une mémoire plus grande pour optimiser les temps de calcul des process d'indexation et de mapping. Cependant, cette option n'était pas envisageable étant donné la limitation du quota disponible sur le serveur de l'IFB cloud. Il existe donc un compromis à prendre en compte, entre le choix de la configuration de la VM et la capacité du serveur qui héberge notre VM.

5.2 Résultats biologiques

PCA :

La figure ci-dessous nous montre que les échantillons sont bien séparés selon le sample type (WT ou M). Nous voyons également que les échantillons WT sont regroupés ensemble contrairement aux échantillons mutés qui sont dispersés. Cette figure nous permet de suggérer que le sample type est une source de variation d'expression des gènes entre les 2 groupes de patients.

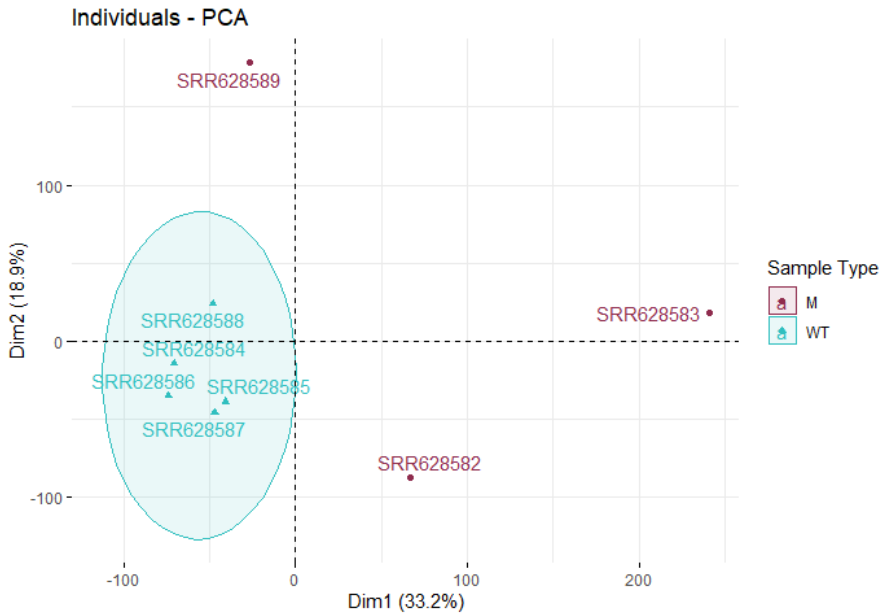


FIGURE 4 – Résultats de l'ACP

Clustering :

Nous observons d'après la figure ci-dessous que les corrélations sont assez élevées (>0.975), ce qui suggère l'absence d'échantillons aberrants. En plus, nous voyons parfaitement que les échantillons WT se regroupent ensemble comme pour le graphique PCA.
Les 2 graphes (Heatmap et PCA) nous confirment que nos données sont de bonne qualité et nous pourrons procéder à une analyse d'expression différentielle.

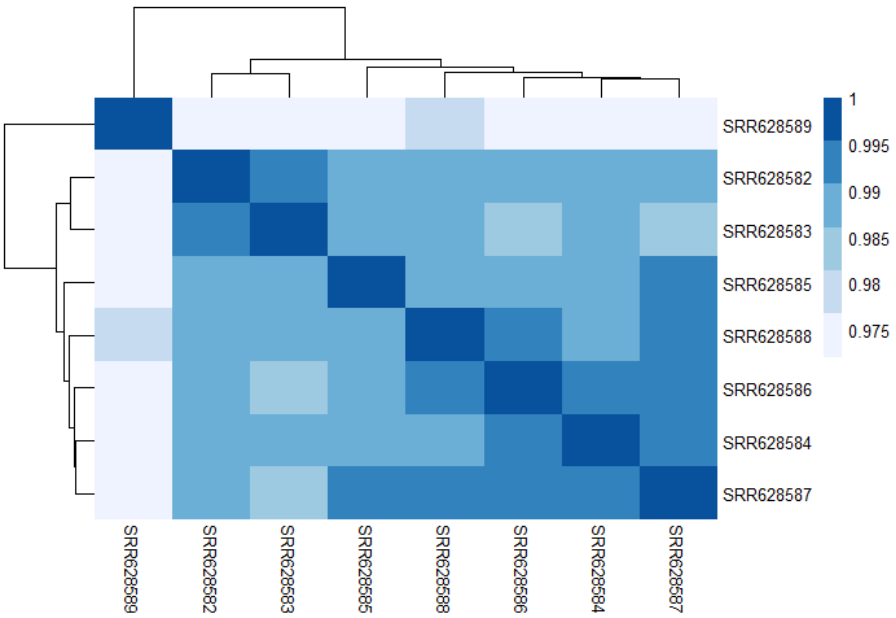


FIGURE 5 – Heatmap : Résultats du clustering des échantillons

DE_Hack_last_version.R | DF_top_10_gene

Filter

	gene	baseMean	log2FoldChange	lfcSE	stat	pvalue	padj
1	164659	286.80988	-5.004364	0.5168972	-8.714235	2.927287e-18	6.429201e-14
2	234261	79.00771	5.948810	0.8204254	6.641444	3.106242e-11	3.411119e-07
3	153283	52.28029	5.656751	0.8526379	6.047996	1.466590e-09	8.052679e-06
4	286122	27.03640	7.321919	1.1279147	6.048258	1.464201e-09	8.052679e-06
5	186973	28.74261	6.421597	0.9871011	5.998977	1.985646e-09	8.722148e-06
6	7038	32.53690	8.761554	1.3926981	5.932049	2.991767e-09	1.095136e-05
7	214212	63.30590	4.103196	0.6141666	5.866806	4.442699e-09	1.393929e-05
8	142621	27.57786	5.365624	0.8549272	5.691273	1.260954e-08	3.461791e-05
9	103449	74.00318	-4.298865	0.6737634	-5.638277	1.717598e-08	4.191512e-05
10	244649	22.55096	7.060190	1.1842652	5.539460	3.034057e-08	6.057908e-05

FIGURE 6 – Les TOP 10 gènes différentiellement exprimés

Après avoir fixé les seuils de la p-value ajustée et le log2 fold change, nous avons arrangé, selon la p-value ajustée, les gènes différentiellement exprimés et nous avons extrait les 10 TOP gènes ayant les 10 p-values ajustées les plus faibles. Comme indiqué dans le tableau ci-dessus, les 10 TOP gènes différentiellement exprimés chez les patients mutés ont des valeurs de log2FoldChange allant de -5 à 8. Si on prend l'exemple du gène 6 qui a un log2FoldChange égale à 8, cela signifie que ce gène est 8 fois plus exprimé chez les mutés que chez les Wild-Type.

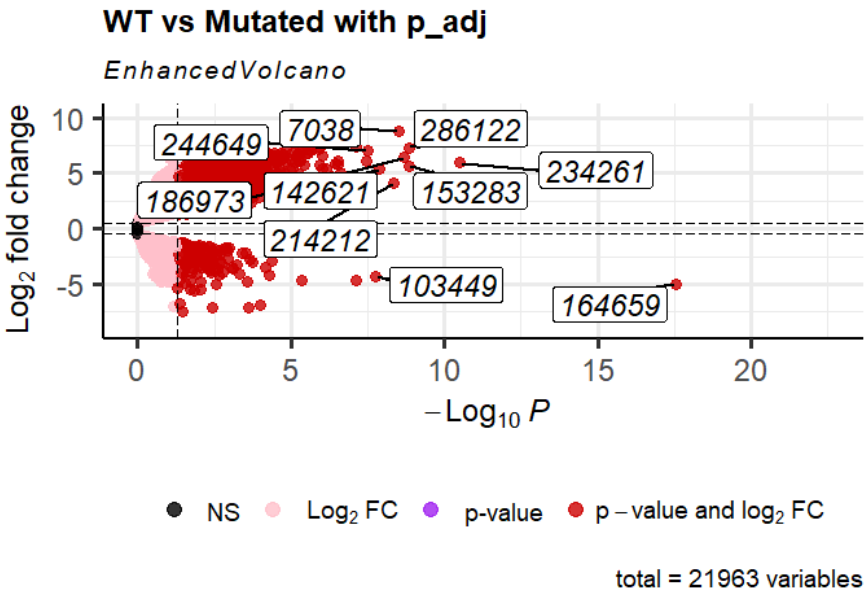


Figure 7 – Résultats sous forme de volcano plot

Cette figure a été construite à l'aide de la fonction Enhanced Volcano, après avoir éliminé les données avec une valeur NA (les gènes qui présentent 0 read dans tous les échantillons). Les points rouges correspondent aux gènes ayant une différence d'expression significative (p adjust<0.05) entre les 2 groupes de patients (WT et mutés). Ainsi, nous observons que les gènes ayant une p-value ajustée <0.05 ont également un log2 fold change inférieur à 1 (L'expression moyenne de ces gènes est au moins deux fois moins importantes chez les patients mutés) ou supérieur à 1 (L'expression moyennes de ces gènes est au moins deux fois plus importantes chez les patients mutés). Dans le graphe ci-dessus, nous pouvons voir que parmi les 10 meilleurs gènes qui sont différentiellement exprimés, 8 gènes sont surexprimés chez les mutés et seulement 2 gènes sont sous-exprimés chez les patients mutés au gène SF3B1.
Pour nos interprétations, nous avons fixé le seuil du ratio ajusté du log2 fold changes (LFC) à 0.5 (cf 4. Méthodes des analyses statistiques) et nous n'avons pas pris en compte les valeurs nulles (Non Available, NA). Nous observons ainsi 266 gènes sur-exprimés correspondant à un LFC supérieur à 0.5 (up) avec une probabilité de 1.2% et 16 gènes sous-exprimés, avec une probabilité de 0.073% et un LFC < 0.5 (down)(cf Annexes, figure 12).

Nous allons à présent regarder si nos résultats diffèrent des conclusions mis en évidence dans les deux publications sur lesquelles se base ce projet. En effet, dans l'article de Harbour et al., l'étude a été menée sur trois échantillons présentant une mutation SF3B1 et cinq échantillons SF3B1 sauvages et concernait exclusivement la mutation arginine 625 de la sous-unité 1 du facteur d'épissage 3B. Cette étude a ainsi mis en évidence une expression différentielle pour uniquement 10 gènes et sans modification fonctionnelle significative de la mutation SF3B1. Tandis que dans le second article de Furney et al., 325 gènes ont été prédits comme étant différentiellement exprimés chez les mutants SF3B1, avec 46 gènes sur-exprimés et 279 gènes sous-exprimés. Cette étude a été faite en complément de celle réalisée dans le premier article de Harbour et al. Elle a été réalisée sur douze patients, dont trois patients possèdent des mutations SF3B1 et neuf autres patients possèdent un gène SF3B1 non muté. Ils ont ainsi identifié, grâce à différentes analyses, trois évènements d'épissages alternatifs en présence des gènes UQCC, CRNDE et ABCC5. Ils sont également parvenus à des conclusions précises, en confirmant que les gènes GNAQ et GNA11 sont des gènes moteurs de mutations onco-

géniques (conclusion similaire dans l'article de Harbour et al.). De plus, il a été mis en évidence que le gène BAP1 est un suppresseur de tumeur mutée et que le gène SF3B1 est muté dans 15% des cas, avec une mutation au niveau de l'arginine 625.

Nous avons donc ici mis en évidence un nombre plus important de gènes différentiellement exprimés que dans l'article d'Harbour et al. (282 gènes contre 10 gènes différentiellement exprimés), avec le même jeu de données. Cependant, nos résultats diffèrent également de ceux obtenus dans la publication de Furney et al. En effet, ils ont observé 46 gènes surexprimés et 279 gènes sous-exprimés (325 au total), tandis que nos analyses nous ont donné une part plus importante de gènes surexprimés que sous-exprimés (respectivement 266 et 16).

6 Conclusion et Discussion

Nous avons utilisé les mêmes données que les publications d'Harbour et al. et Furney et al., pourtant nous avons trouvé des résultats différents. Comment cela s'explique-il ? Tout d'abord, il faut prendre en compte le fait que les études faites dans ces deux articles ont été réalisées en 2013 et que nous les avons reproduites en 2021, soit huit ans après. En effet, notre projet se présente dans un contexte différent des études entreprises dans ces deux articles. D'autant plus, que notre workflow utilise des versions récentes et des packages différents. Les différences notables des matériels et méthodes utilisées ont été regroupées dans le tableau suivant :

Matériels et méthodes	Article 1 (Harbour et al.)	Article 2 (Furney et al.)	Notre workflow
Génome de référence	GRCh37	GRCh37	GRCh38
Mapping	TopHat (avec un RefSeq GTF)	TopHat + trimmed à 99 bp (contrôle qualité)	STAR v2.7.6a
Version de samtools	samtools v0.1.18	N/A	samtools :v1.11
Version de DESeq	DESeq v1.8.3	DEXseq	DESeq2
Tests utilisés	Test de Fisher, Test de Student binomial	test du χ^2 (khi-2) et Fisher	Test de Wald
Seuils utilisés	FDR (False discovery rate) ≤ 0.05	FDR $\leq$ 0.1 (= significatif). P-value $\leq$ 0.05 (two-sided)	P-value ajustée ≤ 0.05. LFC ≥ 0.5 (= gène surexprimé)

En effet, nous pouvons remarquer que dans le cas de notre pipeline, nous avons choisi d'utiliser la méthode STAR pour l'alignement de nos reads RNAseq sur le génome de référence GRCh38, contrairement aux études faites par Harbour et al. ainsi que Furney et al. qui ont utilisé la méthode TopHat et le génome antérieur GRCh37. De plus, pour l'analyse statistique nous avons décidé le test de Wald pour déterminer si un gène est significativement différentiellement exprimé ou non, alors que les auteurs des deux articles ont choisi d'utiliser des tests de Fisher ainsi que Student binomial et le test du 2.

En conclusion, cela met en avant les limites de la reproductibilité d'une étude bioinformatique de données RNAseq. Comme nous l'avons décrit dans l'introduction de ce rapport, nous avons cherché à respecter les trois types de reproductibilité (empirique, computationnel et statistiques) au cours de ce projet. Compte tenu du fait que nous avons utilisé les mêmes données RNAseq provenant du site du NCBI SRA, nous avons essayé de respecter la reproductibilité empirique des expériences biologiques réalisées dans les deux publications de Harbour et al. et Furney et al. De plus, nous avons utilisé les mêmes process que ces derniers. Cependant, comme nous l'avons vu précédemment, nous nous sommes servis d'outils et matériels plus récents que ceux utilisés dans les deux articles. En conséquence, nous pouvons expliquer ce défaut de reproductibilité computationnel et statistiques par cette faiblesse dans le design de l'expérience et des informations concernant les matériels et méthodes utilisés.

En effet, ce manque de puissance expérimentale et de documentation concernant la pipeline utilisée, les hypothèses statistiques ou encore le facteur de normalisation des données utilisés, peuvent engendrer une capture importante de bruit de fond et du bruit biologique dans nos résultats. Dans notre cas, nous avons cherché à rendre notre projet le plus reproductible possible en utilisant un workflow automatisé avec un script nextflow. Nous avons également utilisé des containers Docker open source et nous avons documenté nos scripts commentés, en fournissant un read.me et un rapport très détaillée explicitant toutes les étapes de ce projet.
Par ailleurs, des suggestions d'améliorations peuvent être proposées pour optimiser notre workflow, notamment en temps de calcul. En effet, nous avons utilisé pour le mapping, l'outil featureCounts (provenant du docker subread), une technique couramment utilisée dans les analyses RNAseq. Cependant, avec l'amélioration des techniques RNAseq et la révolution des outils bioinformatiques des dernières années, il existe aujourd'hui des algorithmes plus performants de normalisation des reads RNAseq, notamment pour aligner les reads dit ambigus (par exemple les reads qui s'aligne en partie sur une jonction entre un exon et un intron).
Effectivement, les méthodes utilisant l'algorithme Expectation Maximisation (EM) comme Sailfish, Salmon ou Kallisto sont dix fois plus rapides et plus précises que les méthodes de type featureCounts. FeatureCounts utilise une normalisation RPKM (Reads Per Kilobases Per Million), tandis que les algorithmes EM normalisent directement les reads avec les transcrits de référence (Transcrit Per Million) et donne une meilleure quantification des expressions différentielles des gènes. Il serait donc intéressant de refaire notre pipeline avec un algorithme EM pour améliorer cette analyse bioinformatique de données RNAseq.

7 Bibliographie

[1] Baker M. 1,500 scientists lift the lid on reproducibility. Nature 2016 ;533 :452–454. Available at : https ://www.nature.com/articles/533452a. Accessed November 29, 2021.

[2] ENA Browser. Available at : URL :
https ://www.ebi.ac.uk/ena/browser/view/SRA062359 ?show=reads. Accessed November 29, 2021.

[3] NCBI SRA Run Selector. Available at :
https ://www.ncbi.nlm.nih.gov/Traces/study/ ?acc=SRP017413o=acc_s. Accessed November 29, 2021.

[4] NCBI Insights : New API Keys for the E-utilities. NCBI Insights 2017. Available at :
https ://ncbiinsights.ncbi.nlm.nih.gov/2017/11/02/new-api-keys-for-the-e-utilities/. Accessed November 29, 2021.

[5] Docker Hub. Available at :
URL : https ://hub.docker.com/u/evolbioinfo. Accessed November 29, 2021.

[6] GFF/GTF File Format. Available at :
URL : https ://m.ensembl.org/info/website/upload/gff.html. Accessed November 29, 2021.

[7] Raphaël, Leman. (2019). Développement d'outils biostatistiques et bioinformatiques de prédiction et d'analyse des défauts de l'épissage : application aux gènes de prédisposition aux cancers du sein et de l'ovaire. Available at :
URL : https ://www.researchgate.net/publication/338829076_Developpement_d'outils_biostatisques_et_bioinformatiques_de_prediction_et_d'analyse_des_defauts_de_l'epissage_application_aux_genes_de_predisposition_aux_cancers_du_sein_et_de_l'ovaire. Accessed November 29, 2021.

[8] Introduction to DGE-ARCHIVED. Available at :
URL : https ://hbctraining.github.io/DGE_workshop/lessons/02_DGE_count_normalization.html ?fbclid=IwAR18bZaeXah0qHSg0q5VR08iHnrd3DGiPcsA4vS0AKGrGPSbuZT5C2flcd8. Accessed November 29, 2021.

8 Annexes

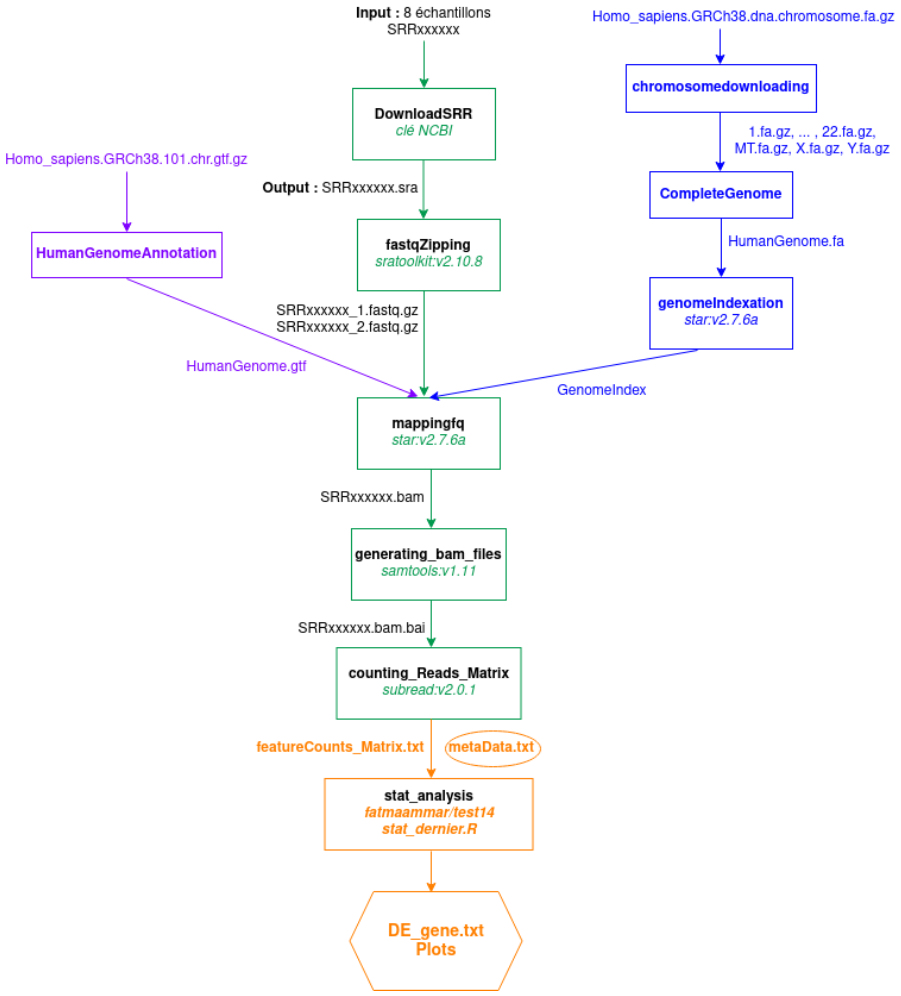


FIGURE 8 – Schéma du workflow complet (nextflow)

```
executor >  local (18)
[4a/4d6459] process > DownloadSRR (6)            [100%] 8 of 8, cached: 8 ✔
[d2/43114c] process > fastqZipping (4)           [100%] 8 of 8, cached: 8 ✔
[96/1087d6] process > chromosomedownloading (25) [100%] 25 of 25, cached: 25 ✔
[17/349f66] process > CompleteGenome             [100%] 1 of 1, cached: 1 ✔
[52/4f0682] process > genomeIndexation           [100%] 1 of 1, cached: 1 ✔
[8a/a71a27] process > HumanGenomeAnnotation      [100%] 1 of 1, cached: 1 ✔
[da/79e566] process > mappingfq (1)              [100%] 8 of 8 ✔
[ca/34a949] process > generating_bam_files (8)   [100%] 8 of 8 ✔
[46/2be1a4] process > counting_Reads_Matrix      [100%] 1 of 1 ✔
[84/63d8f0] process > stat_analysis (1)          [100%] 1 of 1 ✔
Sucess
Completed at: 29-Nov-2021 23:37:04
Duration    : 2h 8s
CPU hours   : 18.8 (62.8% cached)
Succeeded   : 18
Cached      : 44
```

FIGURE 9 – Capture d'écran montrant les process terminés du pipeline. Le temps de calcul estimé est de 3h.

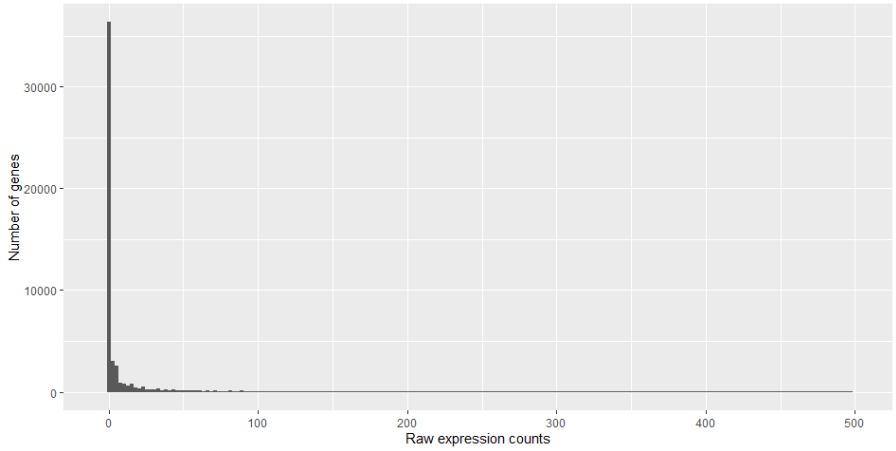


FIGURE 10 – Histogramme avec les comptages bruts d'expression des gènes

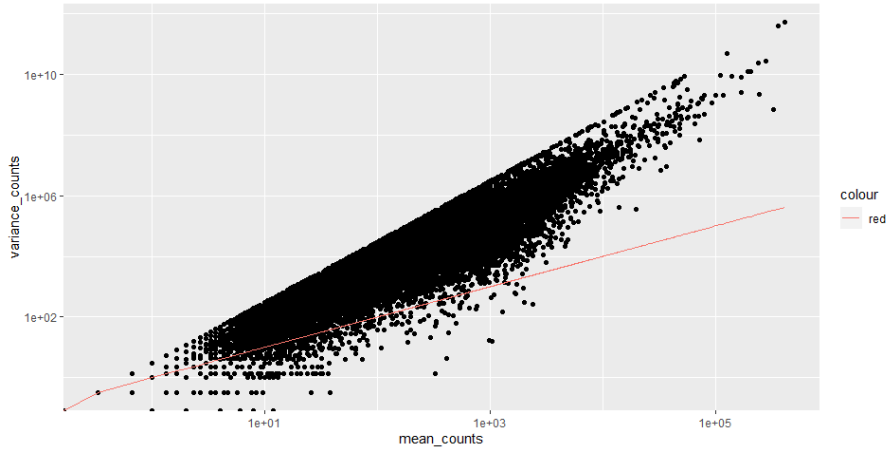


FIGURE 11 – Graphe de dispersion des comptages

Avant de fixer le seuil :

```
out of 30029 with nonzero total read count
adjusted p-value < 0.1
LFC > 0 (up)       : 1098, 3.7%
LFC < 0 (down)     : 169, 0.56%
outliers [1]       : 903, 3%
low counts [2]     : 8265, 28%
(mean count < 2)
[1] see 'cooksCutoff' argument of ?results
[2] see 'independentFiltering' argument of ?results
```

Avant d'éliminer les NA :

```
out of 30029 with nonzero total read count
adjusted p-value < 0.05
LFC > 0.50 (up)    : 266, 0.89%
LFC < -0.50 (down) : 16, 0.053%
outliers [1]       : 903, 3%
low counts [2]     : 7163, 24%
(mean count < 1)
[1] see 'cooksCutoff' argument of ?results
[2] see 'independentFiltering' argument of ?results
```

Après avoir fixé le seuil et éliminer les NA :

```
out of 21963 with nonzero total read count
adjusted p-value < 0.05
LFC > 0.50 (up)    : 266, 1.2%
LFC < -0.50 (down) : 16, 0.073%
outliers [1]       : 0, 0%
low counts [2]     : 0, 0%
(mean count < 1)
[1] see 'cooksCutoff' argument of ?results
[2] see 'independentFiltering' argument of ?results
```

FIGURE 12 – Captures d'écrans de la sortie de la fonction results dans les trois cas