

Rapport de stage

Création d'un pipeline d'analyse de variants génomiques dans le cadre d'une expérience de mutagenèse aléatoire.

Résumé :

Ce stage s'est déroulé dans le cadre du master 2 de Bioinformatique au sein du Laboratoire de Recherche et Développement des plantes (RDP). Afin d'aider des généticiens, peu familiers avec les outils bio-informatiques, j'ai développé un pipeline Nextflow d'appel et d'analyse de variants. Ce pipeline permet d'obtenir une liste de mutations induites par mutagenèse sur un ou plusieurs échantillons mais également une classification de leurs impacts fonctionnels. Nous détaillerons les programmes utilisés et les essais d'optimisation du pipeline et nous présenterons les résultats obtenus pour un jeu de données issues d'une expérience de mutagenèse de mousse.

Romuald Marin
Année 2020/2021

Introduction	5
Présentation du pipeline	8
Matériels & Méthodes	10
Data	10
Séquences des plants mutants de mousse	10
Génome de référence et annotations fonctionnelles de mousse	11
Alignement des lectures	11
Mapping avec Bwa et samtools	11
Picard	12
Appel des variants	13
Petit variants	13
Variants structuraux	15
Prédiction de l'impact fonctionnel des variants et recherche de variants candidats	16
Snpeff et Snpsift	16
Python et R	17
Gestion de l'automatisation du pipeline	17
Serveur de calcul (PSMN, Pôle Scientifique de Modélisation Numérique)	17
Conteneurs de type “Docker” & Singularity	18
Nextflow	18
MultiQC	20
Résultats et Discussion	21
Validation du pipeline de recherche de variants, automatisé et simple d'utilisation	21
Optimisation de la recherche de petits variants	23
Optimisation de la recherche des variants structuraux	28
Exploration des résultats et premiers retour à la biologie	31
Conclusion	33
Références bibliographiques	35

Table des figures :

Figure 1 : Image de *Physcomitrium patens*

Figure 2 : Représentation de l'expérience de mutagenisation, à gauche le plant de départ avec un développement normal et à droite les plants mutés avec des caractères phénotypiques intéressants.

Figure 3 : Schéma du fonctionnement général du pipeline

Figure 4 : Schéma et photo de la mutagenèse

Figure 5 : Représentation des performances d'alignement des différents programmes

Figure 6 : Représentation des séquences dupliquées parmi les lectures alignées

Figure 7 : Représentation des différents variants structuraux possibles

Figure 8 : Exemple de fichier de configuration Nextflow

Figure 9 : Exemple de rapport fourni par nextflow

Figure 10 : Représentation de l'analyse avant la création du pipeline avec un total de 8 scripts différents dont les entrées/sorties ne sont pas enchaînées automatiquement

Figure 11 : Capture d'écran du terminal après le lancement du script

Figure 12 : Tableau de comptage du nombre de petits variants et de leur qualité moyenne

Figure 13 : Représentation de la densité des variants en fonction de la qualité (en bleu les variants spécifiques et en rouge les variants hérités)

Figure 14 : Tableau récapitulatif des différents essais d'amélioration de détection des petits variants

Figure 15 : Graphique de l'effet de la modification de l'option 'ploidy' dans le programme GATK

Figure 16 : Récapitulatif des petits variants trouvés avec GATK.

Figure 17 : Histogramme de la modification de la taille des variants structuraux du mutant 4

Figure 18 : Graphique récapitulatif des variants structuraux

Figure 19 : Représentation du nombre de variants structuraux trouvés par échantillons en fonction du type : Hérité ou Spécifique

Figure 20 : Graphique récapitulatif des impacts prédictifs et position des variants par le programme snpEff

Glossaire des abréviations

SNP : Single Nucleotide Polymorphisme

Indels : Insertions et délétions

Pipeline : Chaîne de traitement de donnée

Process : Processus de la chaîne de traitement

NGS : Next Generation Sequencing

UV : Rayonnement Ultraviolet

Csv : comma-separated values, fichier à valeurs séparées par des virgules

Breakpoints : Point de rupture entre le génome de référence et l'échantillon

BQSR : recalibration du score de qualité de base

VQSR : recalibration du score de qualité des variants

Liste des logiciels :

Samtools v1.7

Snpsift et.snpEff v5.01

Fastqc v0.11.9

Breakdancer v1.1

MultiQC v1.7

Pindel v0.2.0

Picardtools v2.18.11

MetaSV v0.5.4

Bowtie2 v2.4.3

Python v3.6

Bwa v0.7.17

R & Rstudio v4.1.0

GATK v4.1.9.0

CNVnator v0.4.1

Lumpy v0.2.13

Introduction

Le laboratoire qui m'accueille durant ce stage de 2eme années de Master Bioinformatique, est le Laboratoire RDP (Laboratoire Reproduction et Développement des Plantes) sous la tutelle de différents organismes (CNRS, ENS de Lyon, UCBL, INRA , INRIA). Le laboratoire, comme son nom l'indique, étudie le développement des plantes et l'évolution des structures reproductives de celle-ci au travers de plantes modèles (notamment *Arabidopsis thaliana* et *Physcomitrium patens*). Mon équipe s'intéresse plus particulièrement à la phyllotaxie et aux dynamiques de développement conditionnées par le génotype, en combinant des approches de génétiques et de microscopie. La phyllotaxie est l'étude de l'architecture aérienne de la plante et de la disposition des feuilles, tiges et autres. L'étude de ces arrangements est indissociable de l'étude des gènes qui en sont responsables via une signalisation hormonale et une machinerie cellulaire finement régulée.



Figure 1 : Image de *Physcomitrium patens*
Source : PRGdb.org

Une des expériences menée dans ce laboratoire est la mutagénèse aléatoire¹, qui consiste à induire des mutations de façon aléatoire dans l'ADN en exposant des cellules ou tissus végétaux à des rayonnements ou des produits chimiques. Après avoir obtenu des mutations aléatoires dans le génome, les mutants créés sont triés et seuls seront gardés ceux qui présentent les caractères phénotypiques intéressants que l'on souhaite observer. Cette technique a pour but d'identifier des gènes candidats pouvant avoir un lien avec le phénotype sélectionné, grâce aux variants génomiques présents. Dans ce projet, l'organisme modèle² choisi est la mousse *Physcomitrium patens*. Cette plante comporte l'avantage d'avoir la structure classique d'une plante (racine et partie aérienne) et d'être cultivable in vitro. De plus, cette mousse est une bryophyte qui a divergé il y a environ 443-490

millions d'années de la lignée menant aux autres plantes comme les plantes vascularisées ou plantes à fleurs. On peut donc chercher des caractéristiques communes présentes dans la plupart des plantes terrestres, ce qui en fait un modèle intéressant pour des comparaisons évolutives. De plus, le génome de la mousse a été séquencé, annoté et est disponible sur NCBI.

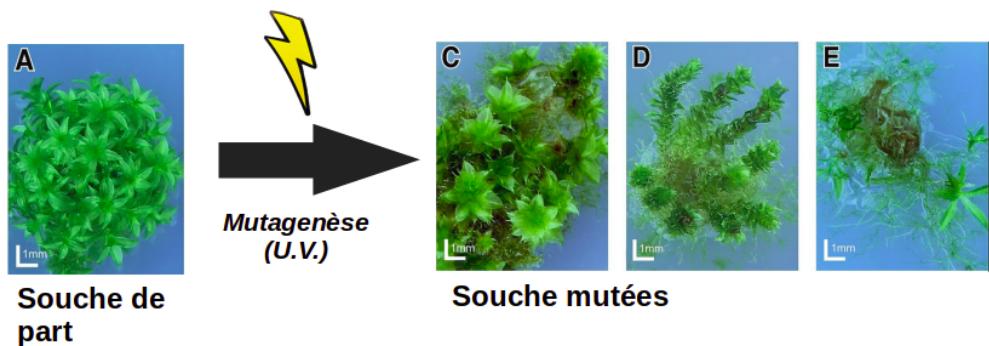


Figure 2 : Représentation de l'expérience du mutagenisation, à gauche le plant de départ avec un développement normal et à droite les plants mutés avec des caractères phénotypiques intéressants.

Toutefois, peu de personnes maîtrisent les outils d'analyses bioinformatiques dans le laboratoire d'accueil. Il est important que leurs utilisations se généralisent afin que ce type d'approche d'identification des gènes clefs par mutagenèse soit utilisée de manière plus routinière par les généticiens expérimentaux. L'enjeu de ce stage est donc d'améliorer les outils existants d'analyse de variants génomiques pour les rendre plus fiables, plus performants et plus accessibles à des scientifiques sans compétences de bio-informatiques. Dans cette optique, nous avons défini un cahier des charges en 3 parties avec mon maître de stage, Fabrice Besnard, chercheur au sein de l'Ecole normale supérieure de Lyon.

Une première partie de ce cahier des charges concerne l'amélioration et la mise à niveau des outils d'analyses de variants déjà utilisés dans le laboratoire et en particulier ceux liés à l'analyse des variants structuraux. Actuellement certains programmes d'appel de variants sont déjà utilisés dans le laboratoire. Mais leur version ne sont plus à jour et certains doivent être remplacés, pour cela une recherche des outils existants modernes a été réalisée.

Dans un second temps, la mise en place d'un système de gestion des différents programmes et fichiers utilisé sous la forme d'une chaîne de traitement afin d'en faciliter son utilisation. Comme il est très largement l'usage dans la communauté bioinformatique, nous utiliserons par la suite l'anglicisme de "pipeline" pour désigner cette chaîne de traitement automatisée. Le pipeline a été créé en utilisant le gestionnaire Nextflow, dans le souci notamment de permettre l'accès à ce genre d'analyse à n'importe quel type d'utilisateur. En effet, il est souvent difficile pour les personnes non formées à l'informatique d'utiliser les différents programmes en ligne de commandes et de jongler avec les différents formats des fichiers données en sortie des programmes. Il est donc important de fournir une solution "clefs en mains" à l'utilisateur novice mais aussi permettre à un utilisateur expérimenté de pouvoir modifier certains paramètres du pipeline. L'objectif final était de fournir un ou plusieurs fichier(s) de sortie facilement lisible, qui donneront des informations sur les différents types de variants présents, leur position dans le génomes ainsi que les gènes potentiellement impactés par ces variants. L'impact des variants sera analysé par un prédicteur d'effet de variant. La sortie doit être facilement compréhensible et lisible par un logiciel tableur (Excel, Calc ...) afin que l'utilisateur puisse trier les résultats selon ses propres critères.

Enfin, la troisième partie a consisté, à un test en condition réelle, à analyser des données concernant un expérience de mutagenèse en cours sur la mousse *Physcomitrium patens*. Le pipeline devrait pouvoir être utilisé par tous et facilement modifiable afin de le rendre le plus durable possible. Il permettra au généticien de mettre en corrélation les variations génomiques aux variations phénotypiques des organismes modèles grâce aux techniques modernes de séquençage (Séquençage massif haut débit dit de deuxième génération ou "Massive parallel sequencing") et aux techniques d'analyse de bioinformatique. Le pipeline doit détecter les variants génomiques, qui peuvent être de différentes natures : Polymorphismes nucléotidiques simples appelés "SNP", insertions/délétions aussi appelés "indels" et variations structurelles. La détection et la localisation génétique de ces variations permettent d'obtenir une meilleure compréhension des phénomènes génétiques et de leur impact possible sur le phénotype.

Présentation du pipeline

Le pipeline créé lors de ce stage nécessite pour fonctionner 3 entrées : le génome de référence de l'organisme étudié, un fichier d'annotation comportant les informations du génome de référence et enfin un ou plusieurs fichiers de lecture issus du séquençage des échantillons sur une plateforme de type Illumina. L'analyse se déroule en 3 étapes :

- La première étape consiste à retrouver dans le génome de référence la position des millions de lectures issues du séquençage massif parallélisé des banques d'ADN préparées à partir des échantillons. Cette première étape permet également de trier les lectures suivant différents critères (qualité, appariement des lectures...) qui aideront la suite de l'analyse.
- Dans un second temps, déterminer avec le plus de certitude possible les variants génétiques de chaque échantillon. Le traitement de ce type de données de séquençage génère de nombreux "faux positifs", dont il est souhaitable d'automatiser la filtration. Les variants de mauvaise qualité doivent être éliminés de même que les variants hérités. Ce sont les variants (par rapport au génome de référence considéré) qui sont présents dans tous les échantillons car ils étaient déjà présents dans la plante à l'origine de l'expérience de mutagenisation³. En effet, lors d'une expérience de mutagenèse aléatoire, la plante choisie par l'expérimentateur comme souche de départ porte souvent des variations avec le génome dit "de référence", et ces différences sont plus ou moins importantes selon la souche choisie.
- La dernière étape du pipeline consiste à l'annotation des variants selon leur possible impact fonctionnel et à générer un fichier de sortie

Le script produit plusieurs rapport en HTML permettant de voir un résumé des différentes options utilisées, ainsi que les durée et consommation de mémoire de chaque processus. Il fournit aussi en sortie un fichier par échantillon résumant les variants uniques présents dans chaque échantillon avec une description de son impact sur la transcription. Un documentation est fournie avec le pipeline

permettant une meilleure compréhension de chaque étape et des exemples de fichier d'entrée afin de limiter les erreurs d'inputs. Ce rapport détaillera les programmes utilisés et les décisions prises pendant l'élaboration pour la réalisation de chacune de ces trois étapes, ainsi que les différents résultats obtenus.

Fonctionnement General du pipeline

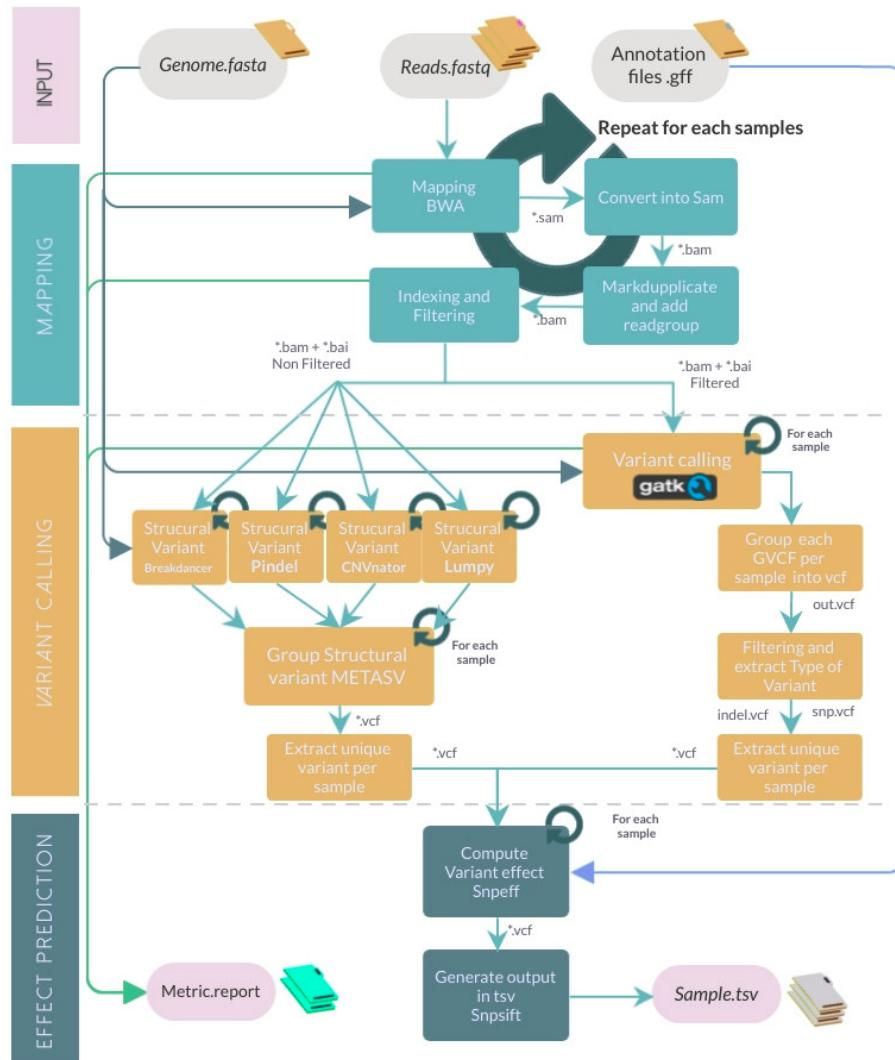


Figure 3 : Schéma du fonctionnement général du pipeline

Matériels & Méthodes

Data

Séquences des plants mutants de mousse

La souche de base de *Physcomitrium patens* provient de Villersexel-RFP, des protoplastes ont été mutés à 75-90 mJ/cm² par rayonnement UV. Les plants ont ensuite été cultivés à 23°C avec 16h de lumière par jour. Bien que le spectre des mutations des UV soit peu connu, une étude ⁴ indique que les substitutions de bases simples G:C à A:T sont plus fréquentes.

Les lectures sont en paired-end et font 100-bp. Elles ont été obtenues par séquençage illumina et encodé en illumina 1.9. Les données de séquençage ont été pré-traité par la société chargée du séquençage. La couverture moyenne par échantillons est de 12X.

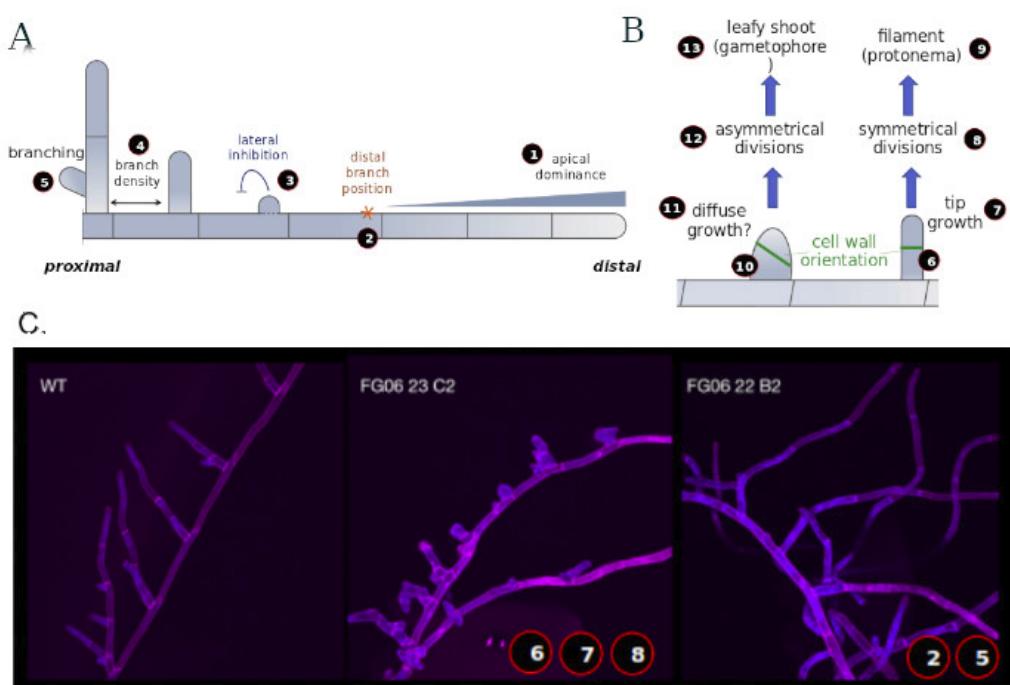


Figure 4 : Schema et photo de la mutagenèse

A. Caractéristiques clés du développement dans la morphogenèse des ramifications filamentueuses. B. Rupture de symétrie et de phyllotaxie

C. Exemples de mutants UV de plant de *Physcomitrium patens* (coloration Iodure de propidium)

Avec l'aimable autorisation de Yoan Coudert, données non publiées.

Génome de référence et annotations fonctionnelles de mousse

Le génome de référence de la mousse⁵, *Physcomitrium patens*, provient du NCBI. Il a été assemblé en 2018 par le Moss Consortium. Le génome est composé de 27 chromosomes et de 3,002 contigs pour une taille totale de 472.081 Mb.

Alignement des lectures

Mapping avec Bwa et samtools

Afin de retrouver l'emplacement des lectures des différents échantillons, chacune doit être alignée sur un génome de référence. C'est une étape clef de l'analyse car le programme utilisé lors du mapping assigne chaque lecture à sa position présumée, le long d'un génome de référence, dans notre cas le génome de la mousse. Donc un mauvais alignement pourrait entraîner des erreurs lors de l'appel des variants et créer des potentiels faux-positifs. Mais aussi dans le cas où le programme d'alignement ne trouve pas la position de la lecture, on ne verra pas le variants et cela entraînera des potentiels faux négatifs. Pour ce pipeline, différents programmes de mapping ont été testé : Bowtie⁶, ainsi que 2 programmes de BWA⁷ : BWA aln et BWA mem.

Dans une comparaison des performances de 3 programmes d'alignement : Bowtie2, BWA aln et BWA mem, on peut voir que BWA mem permet d'aligner une plus grande fraction de lectures de séquençage brutes. Toutefois, nous n'avons pas les moyens d'évaluer la justesse de ces alignements. Comme nous le verrons plus tard, la suite du pipeline contient de nombreuses d'étapes dont le but est d'éliminer des faux-positifs de différentes origines, donc des erreurs d'alignement notamment. En revanche, il est difficile de corriger *a posteriori* les locci manqués par l'alignement (faux

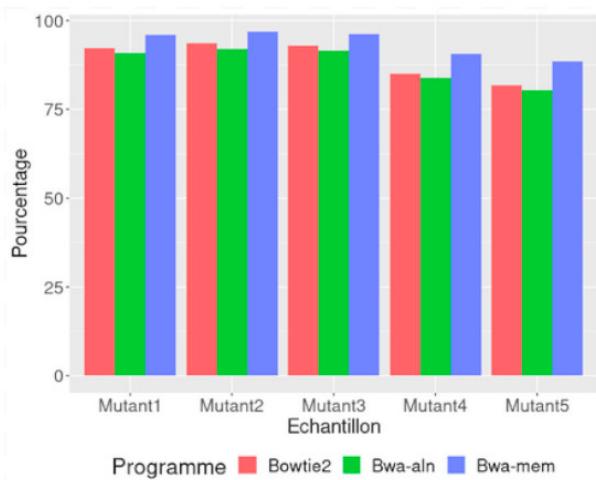


Figure 5 : Représentation des performances d'alignement des différents programmes

négatifs), qui disparaîtront de l'analyse. Dans cette première étape de pipeline, nous avons donc fait le choix de prioriser l'élimination des faux négatifs sur celle des faux positifs, et donc de maximiser le pourcentage de lecture alignées (choix du programme BWA-mem).

La sortie de l'alignement est un fichier qui contient les positions des lectures alignées sur le génome de référence, au format SAM (Sequence Alignment Map) ou sa version binaire, BAM (Binary Alignment Map) générée par le programme Samtools⁸.

Pour la suite du pipeline, nous avons utilisé la propriété des lectures appariées (paired-ends) de deux façons différentes. Concernant l'analyse de petits variants, nous privilégiions la qualité de l'alignement des lectures car l'information principale est la séquence lue (et les éventuelles variations par rapport à la référence). Les lectures alignées de façon non primaires (“not primary alignment”) ou les lectures alignées sans leur paire (“read mapped in proper pair”) ayant de forte chance d'être mal alignées, les différences de séquence avec la référence peuvent alors être purement artificielles. En revanche, la détection des variations de structures reposent moins sur la séquence lue que sur les propriétés de l'alignement (distance entre paires, brin d'adn aligné, chromosome, répétitions de l'alignement): pour ces analyses les fichiers bam ne doivent peut être filtrés.

Picard

La dernière étape avant l'appel de variant est la préparation des fichiers d'alignements pour les prochaines étapes grâce au programme Picard⁹. Cette étape comprend des opérations d'annotation des fichiers selon les échantillons et le nettoyage des données pour corriger les biais techniques: marquage des doublons et recalibrage des scores de qualité de base. Lors de la préparation des échantillons pour le séquençage, des

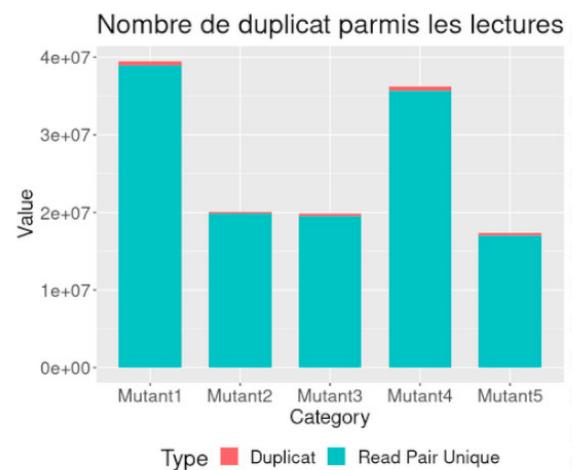


Figure 6 : Représentation des séquences dupliquées parmi les lectures alignées

doublons PCR apparaissent à l'étape d'amplification PCR des fragments. Puisqu'ils partagent la même séquence et la même position d'alignement, ils peuvent conduire à des problèmes de détection de variants et certaines allèles peuvent être surreprésentés. Pour surmonter ce problème, les doublons de PCR sont marqués avec une certaine étiquette à l'aide d'un algorithme (MarkDuplicates). De plus certains programmes dans la suite de l'analyse nécessitent des informations tel que les "read group" (@RG) dans les informations des en-têtes ("headers") des fichiers BAM afin de bien réaliser leur travail.

Appel des variants

Petit variants

Une fois l'alignement fait et les lectures filtrées, il faut maintenant commencer l'analyse des données pour trouver les variants génomiques, cette étape est appelée "appel de variants" (ou plus généralement "variant calling"). Pour ce faire, nous avons notamment utilisé un outil très populaire, GATK¹⁰ ("Genome Analysis Toolkit"). Initialement développée pour l'analyse des génomes humains, cette suite d'outils génomiques est particulièrement adaptée à la recherche de variants issues de séquençage par Illumina (ou de séquençage massif en parallèle) et a évolué pour s'adapter à tout type d'organisme. GATK permet de rechercher des variants de petite taille, généralement inférieure à 20 bases (snps et des indels).

Afin d'optimiser l'analyse en parallèle de plusieurs échantillons (agrégant potentiellement des données de séquençage différentes), nous utilisons l'outil "HaplotypeCaller"¹¹ en mode Genomic Variant Call Format (GVCF). Alors qu'un vcf classique ne contient que les positions "variantes" (par rapport à la référence), le GVCF incorpore des statistiques sur la confiance qu'une position non détecté initialement comme étant variante par rapport à la référence le soit vraiment (discriminant par exemple une position lue comme "référence" d'une position non couverte par les lectures). Après un appel de variant par échantillon, les fichiers GVCF générés sont alors combinés en un seul .vcf grâce à l'outil "GenotypeGVCF"s.

L'analyse simultanée d'une cohorte d'échantillons permet de détecter toutes les variations en commun entre les échantillons, lorsque ceux-ci sont apparentés: les variations de la souche d'origine (définies comme variations héritées) se retrouvent naturellement dans les souches dérivées tandis que les variations spécifiques d'un échantillon seraient plutôt issus de la mutagénèse.

Un fort niveau de confiance peut être émis pour ces "variations héritées", car elles sont détectées pour tous les échantillons. En ce sens, elles pourraient permettre de trouver des critères de sélection de bons variants, utiles pour éliminer les faux positifs des listes de variants d'intérêt (ceux spécifiques à chaque échantillon, par exemple les mutations induites par la mutagénèse).

Nous avons essayé plusieurs approches afin d'améliorer cette étape d'appel de petits variants :

- Une première étude de l'évolution globale du paramètre de qualité pour les variants hérités et non hérités. Afin de voir si on pouvait trouver une valeur seuil de qualité.
- Dans un second temps, une recalibration du score de qualité de base (BQSR). C'est une fonction de GATK qui permet de détecter les erreurs systématiques commises par le séquenceur lors de l'estimation de la qualité des bases. Grâce à un apprentissage automatique d'un modèle modélisant ces erreurs de manière empirique et ajustant les scores de qualité en conséquence. En prenant en compte le contexte de séquence et la position en lecture, etc. Ainsi, la même base peut voir son score de qualité modifié. C'est dans cette optique d'avoir un appel de variant le plus précis possible que nous avons appliqué cette recalibration.
- Un apprentissage des "bon" variants via la fonction "VQSR" (Variant Quality Score Recalibration) de GATK. Cette fonction permet de donner un jeu de données de "bons" variants afin d'entraîner un modèle permettant la classification des bons et des mauvais variants.

Variants structuraux

Les variants structuraux sont typiquement beaucoup plus grands que les indels détectés avec GATK (de quelques dizaines de bases à 1Mb) et incluent des variations complexes telles que des variations du nombre de copies (CNV), des réarrangements chromosomiques avec des translocations, des inversions ou des duplications. Afin d'avoir les résultats les plus fiables possibles, divers programmes ont été testés: Pindel, Breakdancer, CNVnator, Lumpy et MetaSV.

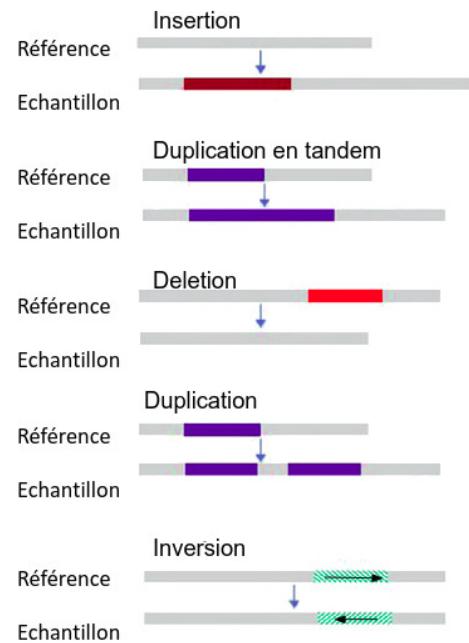


Figure 7 : Représentation des différents variants structuraux possible

Pindel¹² peut détecter les breakpoints des grandes délétions, des insertions de taille moyenne, des inversions, des duplications en tandem et d'autres variants structuraux à partir de données NGS. Il utilise une approche de “pattern growth”, une méthode de data mining permettant de définir si un motif (dans notre cas une séquence) est anormalement fréquent, afin d'identifier les identifier les breakpoints. Il fonctionne avec les “splits reads”, ce sont des lectures dont l'alignement n'est pas complet. Cela peut être causé par une insertion ou une délétion par rapport au génome de référence.

Breakdancer¹³ est un programme dont l'algorithme va détecter des distances anormales entre des alignements appariés compte tenu de la taille attendue des inserts de la banque d'ADN séquencée. Il fonctionne en 2 étapes : l'évaluation de distribution des tailles inserts, puis l'utilisation de cette statistique pour prédire les “breakpoints”, ce sont des cassures dans la séquence par rapport au génome de références. Cela permet de détecter la présence de différents types de variants structuraux : les indels, les inversions et les translocations, d'une taille théorique de 10 bases à 1 mégabase.

Le programme CNVnator¹⁴ permet de détecter les “Copy Number Variation”. Les CNV sont une forme fréquente de variation génétique qui se traduit par un nombre anormal de copies de grandes régions génomiques (événements de gain ou de perte). Pour cela, il utilise la profondeur de couverture afin de définir des zones avec un nombre de CNV anormal.

Lumpy¹⁵ est aussi un programme qui permet la découverte de variants structuraux. Ce programme a la particularité d'être découpé en plusieurs modules, chacun traitant d'un type de signal de variations : split-read, paire de lecture et autres. Cette approche rendrait la détection de variants structuraux plus sensible comparé aux méthodes classiques.

MetaSV¹⁶ permet de combiner les variants appelés par les différents programmes décrits précédemment. En théorie, il permet de pallier le manque de spécificité de chaque programme envers certains types de variants structuraux et de renforcer la confiance et la précision dans la caractérisation et la localisation des ces réarrangements génomiques complexes afin de gommer au maximum le manque de détection de chaque programme. Ce programme permet aussi de réaliser un assemblage des lectures afin d'améliorer la précision des variants structuraux via Spade¹⁷. Mais cette fonctionnalité étant très gourmande en ressource et en temps il a été choisi de la supprimer du pipeline, en effet cela entraînait une augmentation de la durée complète de l'analyse d'environ 2 jours par échantillons.

Prédiction de l'impact fonctionnel des variants et recherche de variants candidats

Snpeff et Snpsift

Pour prédire les effets de chaque variants trouvés sur la fonction du gène impacté nous avons utilisé SnpEff¹⁸.

SnpEff est un outil open source et fournit des prédictions à la fois très détaillées (de nombreux effets possibles sont envisagés, sur des séquences codantes et non

codantes) et classées suivant leur probabilité d'impact fonctionnel. SnpEff génère également des fichiers de rapport étendus et est facilement personnalisable. Enfin SnpEff fournit en même temps SnpSift, un programme qui regroupe plusieurs outils permettant de filtrer et de manipuler les fichiers vcf annotés.

Le but de cette annotation fonctionnelle est de cibler les mutations qui ont le plus de chance de causer le phénotype observé dans l'échantillon séquencé et donc de prioriser leur étude par les biologistes/généticiens.

Python et R

De nombreux scripts ont été créés pendant l'élaboration de ce pipeline. Python a été utilisé dans certains process Nextflow pour transformer les données et extraire les variants d'intérêts : les mutations spécifiques induites par la mutagénèse. Un script python a donc été créé pour la sélection des variants dit "spécifiques". Ces variants ne sont pas présents dans la souche de départ et sont présents dans un seul échantillon issus de la mutagenèse.

Les graphiques présentés dans ce rapport et ceux fournis en sortie du pipeline ont été générés avec R. Par la suite ces scripts seront inclus dans un docker.

Gestion de l'automatisation du pipeline

Serveur de calcul (PSMN, Pôle Scientifique de Modélisation Numérique)

Afin de lancer les analyses pouvant être gourmandes en ressources, j'ai bénéficié, après formation, de l'accès au PSMN (Pôle Scientifique de Modélisation Numérique) de l'ENS de Lyon, un serveur de calcul très puissant avec possibilité de parallélisation (environ 18 000 cœurs et 130 To de RAM sur un système d'exploitation Linux Debian 9). Le PSMN utilise Grid Engine, un gestionnaire de soumission de tâches (ou système de batch). Grid Engine permet de gérer de manière automatique la soumission de travaux et la répartition de charge sur les

différents clusters qui composent le PSMN, en fonction des ressources demandées par l'utilisateur et disponibles sur la grille de calcul.

Conteneurs de type “Docker” & Singularity

La technologie des containers a été utilisée dans ce projet afin d'en faciliter la maintenance. Un conteneur permet de regrouper dans un même endroit sous la forme d'une image, un ou plusieurs logiciels et toutes ses dépendances nécessaires à son exécution (librairies, programmes annexes, configuration...). Cette virtualisation de l'environnement d'exécution permet le fonctionnement stable d'un programme indépendamment du système d'exploitation, des programmes déjà installés, des problèmes de versions des dépendances ou autres. Cette technologie s'adapte à de nombreux cas d'utilisations car tous les programmes ou applications peuvent être transformés en conteneur. Il est possible de lancer docker depuis différentes plateformes (Ordinateur portable, Cloud, serveur distant, ...). De plus, de nombreux containers ont déjà été créés et sont disponibles sur la plateforme publique Dockerhub, où de nombreuses images sont publiées et mises à jour régulièrement.

Singularity un ensemble d'outils permettant l'utilisation de conteneur. Ce programme fonctionne sur Linux mais aussi sur windows et mac. Il nous permettra grâce au gestionnaires de pipeline Nextflow de lancer les programmes dans un conteneur afin de ne pas avoir de conflit de version ou de faire des demandes d'installation de programmes aux administrateurs du PSMN.

Nextflow

Pour réaliser ce projet, c'est le gestionnaire de pipeline Nextflow¹⁹ qui a été choisi. Ce gestionnaire de pipeline fonctionne avec un système de "channel" et de "process". Un "channel" peut être à la fois un fichier, une valeur, une liste de fichier ou autre, cela permet de faire les liaisons entre les différents "process", un peu comme des câbles électriques qui transmettent des informations à différents

appareils. Les “process” ou processus sont des actions à réaliser: ils peuvent être écrits dans n'importe quel langage de script qui peut être exécuté par la plate-forme Linux (Bash, Perl, Ruby, Python, etc.). Ils sont exécutés indépendamment et isolés les uns des autres afin de ne pas créer de conflit entre plusieurs processus. L'interaction entre ces processus est définie par les déclarations d'entrée et de sortie, qui correspondent au “channel”.

Nextflow possède de nombreux avantages dont celui d'avoir été spécialement créé pour des analyses bioinformatiques et de permettre le développement de pipeline portables et reproductibles sur différentes plateformes. Il permet l'utilisation automatique (via un fichier de configuration) de plusieurs plateformes telles que les serveurs Sun Grid Engine, AWS Batch, Google Cloud Life Sciences et autres. C'est un gros avantage lors de son utilisation sur le PSMN, l'utilisateur n'a donc pas à créer des fichiers de soumission pour le système de gestion SGE. Nextflow prend automatiquement en charge cela en exécutant en interne les demandes de soumission de job pour chaque “process” au serveur selon la configuration. Cela pouvant être compliqué pour un utilisateur débutant, ce gestionnaire de pipeline facilite grandement l'exploitation de ce pipeline par des utilisateurs novices. Il est possible de configurer le nombre de coeurs, la mémoire ram nécessaire, le cluster d'exécution ...

```
psmn{
    singularity.enabled = true
    singularity.cacheDir = "$baseDir/.singularity_psmn/"
    singularity.runOptions = "--bind /Xnfs,/scratch"
    process{
        withLabel: samtools {
            container = "lbmc/samtools:1.7"
            executor = "sge"
            clusterOptions = "-m e -cwd -V"
            memory = "20GB"
            time = "12h"
            queue = "monointeldeb192,monointeldeb96,monointeldeb128"
        }
        withLabel: fastqc {
            container = "biocontainers/fastqc:v0.11.9_cv8"
            executor = "sge"
            clusterOptions = "-cwd -V"
            cpus = 1
            memory = "5GB"
            time = "6h"
            queue = "monointeldeb96"
        }
    }
}
```

Figure 8 : Exemple de fichier de configuration Nextflow

Un autre de ses avantages est la gestion des différents programmes via la prise en charge intégrée de Conda, Docker, Singularity et Modules. Un des objectifs de ce projet est la durabilité et la maintenabilité du pipeline. La gestion de docker par Nextflow permettant d'encapsuler chaque programme dans son docker selon une version précise et la possibilité de mettre à jour le docker avec la dernière version du programme au choix est un énorme avantage.

NextFlow possède un système de cache très performant et pratique. Pour les développeurs, cela permet de récupérer les résultats des processus réussis et de recommencer à partir de l'endroit qui a échoué. Pour les utilisateurs finaux, cela leur permettra d'ajouter des fichiers de séquences d'un nouvel échantillon ou encore de modifier des paramètres (par exemple le seuil de qualité des variants) sans avoir besoin de recommencer l'analyse depuis le début.

MultiQC

Nous avons incorporé dans le pipeline l'utilisation du programme MultiQC²⁰ : celui-ci analyse les statistiques récapitulatives fournies par les autres outils bioinformatiques et génère des graphiques sous la forme d'un rapport html générée automatiquement par notre pipeline. Mais permet aussi d'incorporer ses propres graphiques au sein d'un rapport unique. Plusieurs étapes du pipeline fournissent des fichiers de métriques qui peuvent permettre de déceler d'éventuelles anomalies de séquençage ou d'analyse dans les échantillons. Grâce à ce programme, il est simple d'avoir une visualisation claire des résultats de l'alignement des lectures, de l'appel de variants avec GATK et d'autres étapes.

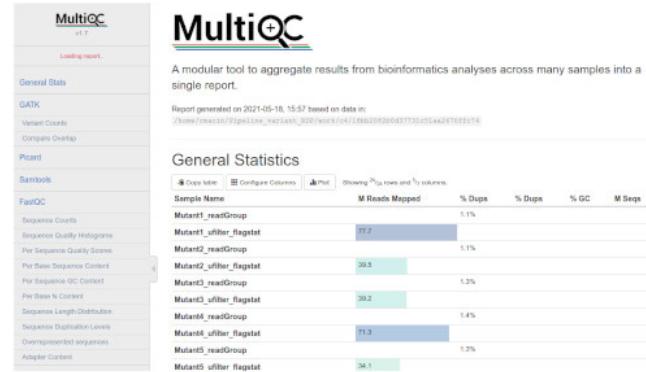


Figure 9 : Exemple de rapport fourni par nextflow

Résultats et Discussion

Validation du pipeline de recherche de variants, automatisé et simple d'utilisation

On démontre ici l'élaboration d'une nouvelle procédure d'appel et d'analyse de variants permettant d'identifier et catégoriser des variations génomiques propres à un échantillon dans le cadre d'une expérience de mutagenèse. Ce nouveau pipeline d'analyse est entièrement fonctionnel. De plus, il est très facilement exécutable par les généticiens. Le pipeline prend seulement 3 types de fichiers en entrée et permet d'être lancé en une seule ligne de commande. La gestion automatique du lancement des process sur le serveur permet d'optimiser les ressources nécessaires en évitant les erreurs de manipulation et permet aussi un gain de temps considérable.

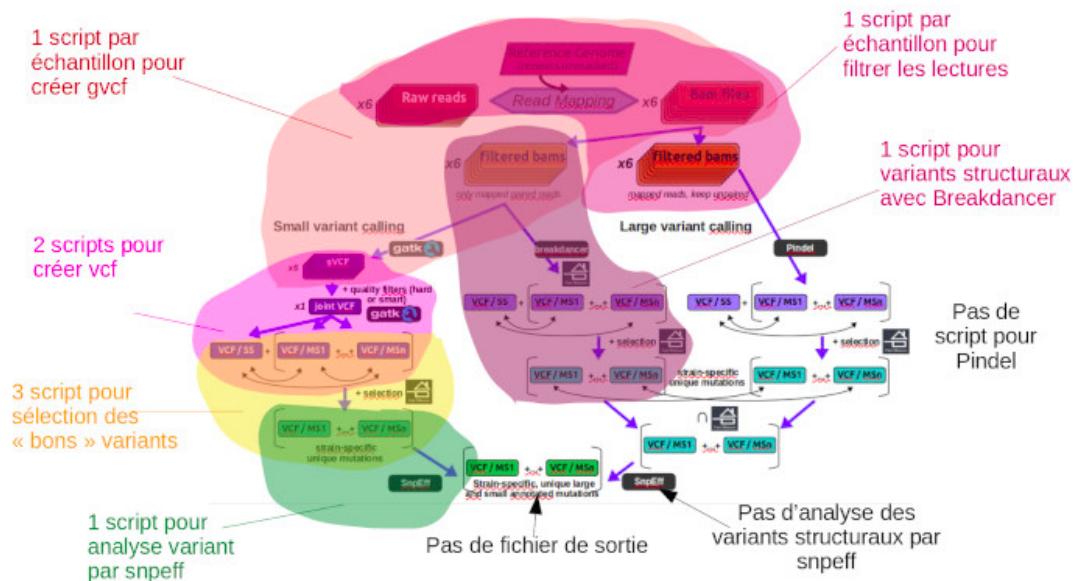


Figure 10 : Représentation de l'analyse avant la création du pipeline avec un total de 8 scripts différents dont les entrées/ sorties ne sont pas enchaînées automatiquement

On peut prendre comme point de comparaison la procédure déjà mise en place dans le laboratoire pour traiter ce genre de données (Figure 10). Voici les différents défauts :

- La précédente procédure d'analyse était composée de 8 scripts différents en python et en bash, dont chacun a des entrées différentes (fichier, dossiers ou valeur). De plus, tous ces scripts devaient être soumis au gestionnaires de calculs du serveur, sous forme de “job”, ajoutant une complexité supplémentaire pour les utilisateurs novices.
- Les scripts sont contraint aux programmes et à leurs versions qui doivent être installées sur le serveur
- La procédure ne permettait seulement de trouver que certains types de variants : les.snp et les petits indels (problème avec les SV).
- Le plus gros défaut est le manque d'adaptabilité de cette procédure, les scripts sont écrits uniquement pour un seul type de design expérimental : 5 échantillons mutés et 1 souche de départ.

Notre pipeline permet d'optimiser les performances de calcul mais aussi les performances de l'analyse en elle-même. Cette nouvelle procédure est plus complète sur différents points :

- La nouvelle procédure gère automatiquement le nombre d'échantillons et ne comporte qu'un seul script à exécuter sans devoir passer par le gestionnaire du serveur. Nextflow fournit une visualisation de l'avancement du pipeline avec un rappel des différents paramètres. Les performances et le temps d'exécution sont améliorés grâce à l'introduction de calcul parallèle, toutes les étapes sont enchaînées par soumission automatique au serveur.

```

scriptdir : /home/marin/Pipeline_variant_RDP/script
reads : /home/marin/ref/reads/*_{1,2}.fq.gz
genomewide_index : Not supplied
reference_fasta : ./ref/Physocontrella_patens.Phypa_V3.dna.toplevel.fa
Ploidy : 1
Config_Profile : psm
annotation_gff : Physocontrella_patens
annotation_gtf : Physocontrella_patens
sampletable : table
tmp_dir : /tmp
VQSR : skipped
removescaffold : Yes
varrate : 5.0
output_dir : GROSSEANALYSE_LUMPY_RM_IMPRECISE
executor : sge (2)
[1c/0d59e1] process > Index_fasta (Physocontrella_patens)
[65/4a22de] process > FastQC (V300042688_L4_AE49504879-612_2.fq.gz)
[cf/7a6341] process > Mapping (mutant1_sample_name) (Startingstrain)
[97/c39969] process > Add_Readgroup_and_HardclipCates_bam (Mutant4)
[e/4a425d] process > Filtering_and_Indexing_bam (Startingstrain)
[f/503922] process > Create_ref_dictionary (Physocontrella_patens.Phypa_V3.dna.toplevel.fa)
[5d/casefe] process > Create_ref_dictionary (Physocontrella_patens.Phypa_V3.dna.toplevel.fa)
[c9/ded7fb] process > Variant_calling (Mutant2)
[93/7a6341] process > Variant_calling (Mutant3)
[b2/2f31f] process > Extract_SNP_and_Filtering (final.vcf.gz)
[5d/9e5d92] process > Extract_INDEL_and_Filtering (final.vcf.gz)
[4f/74679] process > Structural_Variant_calling (Mutants)
[4f/74679] process > Structural_Variant_calling_breakdancer (Mutants)
[20/daf42d] process > Structural_Variant_calling_pindel (Mutants)
[4d/510b5b] process > Structural_Variant_lumpy (Mutants)
[3b/c3e422] process > Structural_Variant_CNVicator (Startingstrain)

```

Figure 11 : Capture d'écran du terminal après le lancement du script

- Le tri des variants issus de la mutagenèse et des faux positifs est plus précis. Le spectre des différents types de variations génomiques est entièrement couvert. L'ancienne procédure était beaucoup moins précise pour l'appel des variants structuraux et la prédiction de leurs effets n'était pas calculée.
- Le pipeline permet de synthétiser les résultats en les combinant par échantillons dans un seul fichier résumant les différentes types de variants et leurs effets présumés. Mais il apporte aussi des informations sur le déroulement de l'analyse via un rapport Nextflow mais aussi des informations sur la qualité des lectures en entrée et sur l'analyse en elle-même.
- Le pipeline apporte une meilleure standardisation de l'analyse et une plus grande reproductibilité des résultats. Grâce au docker qui permet d'utiliser des programmes qui sont figés dans une version et qui n'ont peut-être pas été installés sur le serveur.
- Le pipeline peut être facilement modifié pour y incorporer un nouveau programme ou encore changer pour la dernière version d'un programme. Il suffit simplement d'ajouter un process et/ou de changer le docker associé.
- Le temps d'exécution du pipeline pour l'ensemble des process varie entre 20 et 26 heures. Cette variation dépend de l'utilisation du serveur par d'autres utilisateurs.

Optimisation de la recherche de petits variants

Tout d'abord, il est important de noter que nous avons repéré un grand taux de snps qui confirme l'étude⁴ concernant l'analyse des variants issus des UV. Un fort taux de transition de G:C vers A:T a été repéré, G vers A et C vers T représentent respectivement 20% et 16% des snps totaux, or le taux "normal" de mutation d'une base vers une autre est d'environ 8% (1 / 12 nucléotides possibles).

Lors de la recherche des petits variants avec GATK, diverses tentatives d'optimisation de la recherche de variants hérités ont été testées. GATK

	Avant filtration		Après filtration	
	INDEL	SNP	INDEL	SNP
Qualité moyenne	1597.74	2138.24	1602.24	2518.96
Nombre total de variants	287890	3517493	286902	2788540

Figure 12 : Tableau de comptage du nombre de petits variants et de leur qualité moyenne

propose dans sa documentation d'appliquer un ensemble de filtres selon les indels et les snps sur différents paramètres (qualité de mapping, qualité de la position selon le score Phred, Qualité/profondeur, etc). Dans l'optique de gagner en qualité et supprimer les variants de faible confiance, nous avons appliqué les filtres recommandés. Cette filtration des variants nous a permis d'augmenter la moyenne de qualité comme on peut le voir sur le tableau, cependant beaucoup de snps ont été supprimés (-23% pour les indel et -2% pour les.snp). Une rapide vérification des snps filtrés nous a montré que la filtration était bénéfique car elle supprimait les snps de mauvaise qualité.

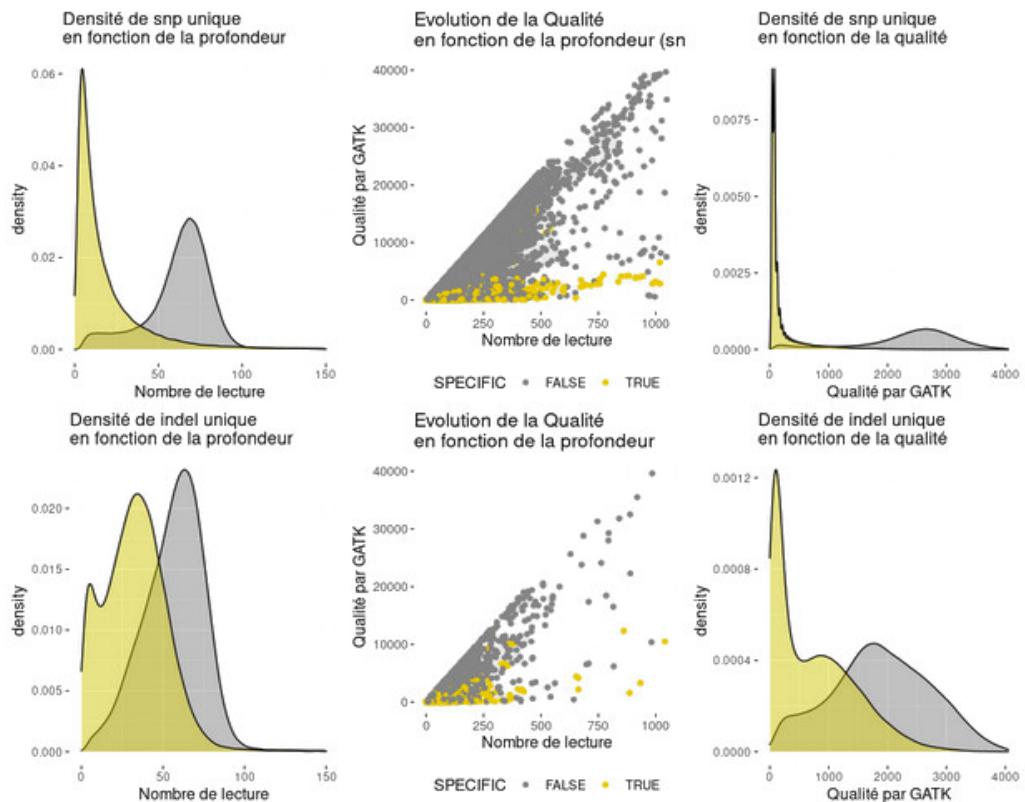


Figure 13 : Représentation de la densité des variants en fonction de la qualité (en jaune les variants spécifiques et en noir les variants héritées)

La première étude portait sur la répartition de la qualité des variants. Lors de cette exploration, on peut voir que le score de qualité de GATK est en lien avec le nombre de lectures associées à un variant. Cependant on peut observer dans la figure 13 que les variants hérités ont en général une meilleure qualité que les variants qui nous intéressent, ceux induits par la mutagenèse. Une des explications possibles à la meilleure qualité de ces variants hérités lus dans chaque échantillon, serait l'emplacement de ceux-ci dans des régions facile à séquencer ou facile à aligner. Ces variants ont donc une meilleure profondeur de lecture donnant des scores de qualité plus élevés.

Dans cette démarche de sélection des “bons” variants, il faut donc trouver un seuil de qualité adéquate afin de ne pas être trop strict et de ne pas éliminer des variants pouvant avoir un impact phénotypique. Nous avons donc fait le choix d’appliquer un filtre de qualité par défaut mais aussi d’offrir la possibilité de fixer un seuil minimum de profondeur de lecture par variants (assez bas). De plus, dans une démarche d’amélioration des résultats, le pipeline fournira en sortie les graphiques ci-dessus afin que l’utilisateur puisse définir lui-même un seuil de qualité adéquate selon son analyse.

Dans un second temps, nous avons évalué l’apport d’une recalibration du score de qualité de base (BQSR). Grâce à cette fonction développée par GATK, nous espérons éliminer efficacement des variants de faible qualité, vraisemblablement faux-positif. Cependant, cette étape supplémentaire n’apporte pas de réelles différences dans les résultats (Figure 14): le nombre de variants après BQSR n’est pas réduit, la BQSR se révélant moins efficace que les filtres précédemment appliqués. Dans un souci d’optimisation du temps de calcul, nous avons donc choisi de mettre cette étape de BQSR en option supplémentaire dans le pipeline mais elle n’est pas dans l’analyse de base.

La dernière étape des essais d’amélioration de la qualité des variants a été la recalibration des scores de variants (VQSR). Cette méthode permet de pallier au manque de spécificité des

	Brut	Avec filtre de base de GATK	BQSR	VQSR
SNPs totaux	3 507 616	2 794 156	2 797 874	2 805 140
INDELs totaux	284 786	283 972	283 541	269 618
Qualité moyenne	2097.92	2636.51	2191.65	2292.31

Figure 14 : Tableau récapitulatif des différents essais d’amélioration de détection des petits variants

méthodes classiques de filtration par critère de qualité, profondeur, nombre de variants dans le voisinage, etc. Cette méthode utilise un algorithme d'apprentissage automatique qui nécessite un jeu de données de "bons" et de "mauvais" variants. Malheureusement, contrairement à l'humain, il n'existe pas, pour la mousse, de données publiques massives de variants existants. Toutefois, comme nous avons observé que les variants hérités étaient de bonne qualité, nous avons essayé de les utiliser pour cet apprentissage. Malheureusement cette étape rallonge considérablement le temps d'exécution du pipeline (environ 25h de plus). De surcroît, le jeu d'apprentissage restant petit, cette recalibration n'apporte pas de réelle amélioration (Figure 14) : la qualité n'a pas augmenté et le nombre de variants reste très important.

Lors du développement du pipeline, nous avons remarqué que la configuration de base de GATK est faite pour des données humaines mais l'organisme modèle de cette expérience est haploïde. Nous avons donc étudié si la modification de la valeur de ploïdie dans GATK pouvait avoir un impact sur le nombre de variants. On peut voir sur le graphique de la figure 15 que le nombre de variants spécifiques a fortement augmenté. Pour la suite de cette analyse du jeu de données de mousse, nous garderons donc cette option qui améliore nos résultats.

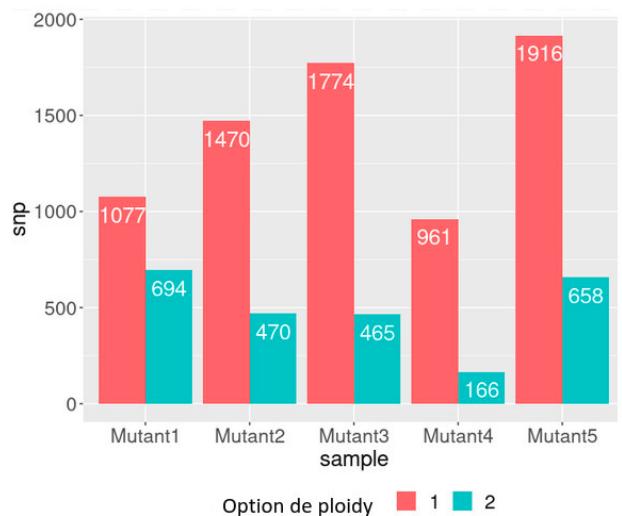


Figure 15 : Graphique de l'effet de la modification de l'option 'ploidy' dans le programme GATK

Grâce à toutes ces étapes, nous avons pu optimiser la recherche des petits variants. Même si les étapes de BQSR et VQSR n'ont pas été utiles du point de vue des appels de variants et ne sont pas proposées dans le pipeline de base, elles restent cependant disponibles en option. Le changement de la valeur de ploïdie a permis d'apporter plus de variants et donc potentiellement plus de variants candidats pour la suite de l'analyse. L'utilisateur peut lui-même modifier la valeur de seuil de qualité selon son analyse grâce aux graphiques fournis en sortie. De plus, grâce au système de cache de Nextflow seul les étapes nécessaires seront recommandées en

cas de nouvelle analyses avec ces paramètres modifiés, optimisant le temps d'utilisation du serveur. Malgré tout il reste potentiellement des variants Faux Positifs (faux spécifiques) qui peuvent survenir lors des étapes précédentes. Par exemple des artefacts issus des cycles PCR pendant la préparation des échantillons, des erreurs lors de l'assemblage, des erreurs issus de l'amplification en pont du séquençage (Illumina) ou bien des erreurs de lecture lors du "BaseCalling".

On peut observer dans la figure 16, le nombre de variants spécifiques obtenus grâce à cette partie du pipeline. Les différences de nombre de variants entre les différents échantillons s'expliquent par la couverture de séquençage qui varie de 7,7x à 17,4x selon les échantillons. On notera que les petits variants sont distribués de façon relativement uniforme dans les chromosomes (Figure 16).

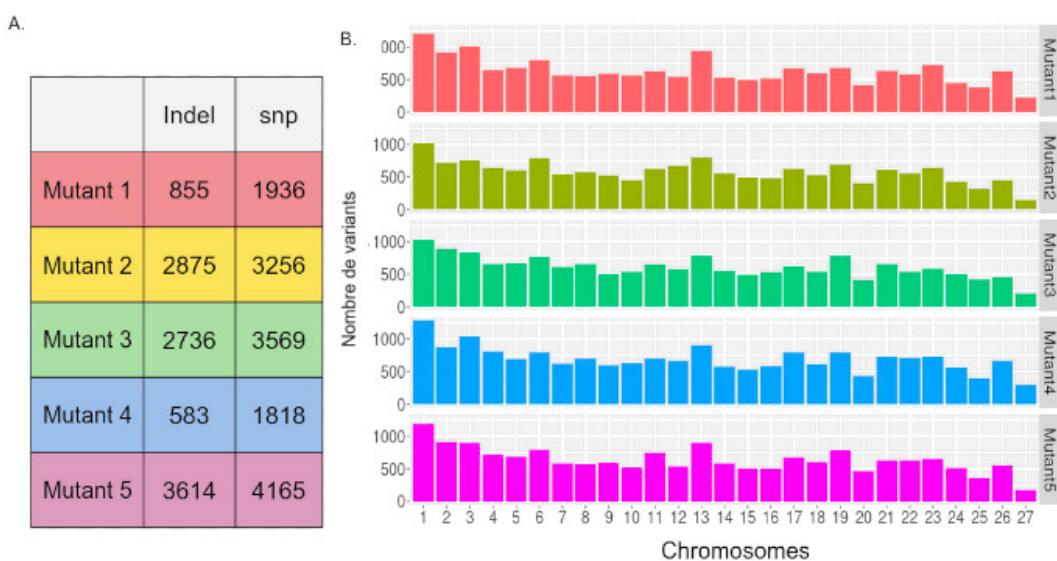


Figure 16 : Récapitulatif des petits variants trouvés avec GATK.
A. Tableau récapitulatif des petits variants spécifiques.
B. Fréquence de petits variants spécifiques en fonction des chromosomes dans les différents mutants

Optimisation de la recherche des variants structuraux

Nous avons entrepris une recherche des meilleurs programmes de détections des variants structuraux. Suite à cette recherche, nous avons sélectionné Pindel, Breakdancer, Lumpy, CNVnator et enfin MetaSV (Voir section matériel et méthode). Le premier enjeu était de découvrir si la filtration des lectures alignées pouvait avoir un impact positif sur la qualité de la détection. En filtrant les lectures sur breakdancer nous nous sommes aperçus que cela introduit des biais dans l'analyse car celui-ci fonctionne avec des "erreurs" lors de l'alignement des lectures appariées.

Suite à cette observation nous avons décidé de conserver tous les reads afin d'avoir les meilleurs résultats pour les programmes de détections de variants structuraux.

Comme cela est connu avec de tels outils, nous avons ensuite observé que les différents programmes breakdancer, pindel, CNVnator, lumpy émettaient souvent des positions et tailles des événements imprécis. Ainsi de nombreux variants structuraux appelés par des programmes différents semblent pourtant appartenir à

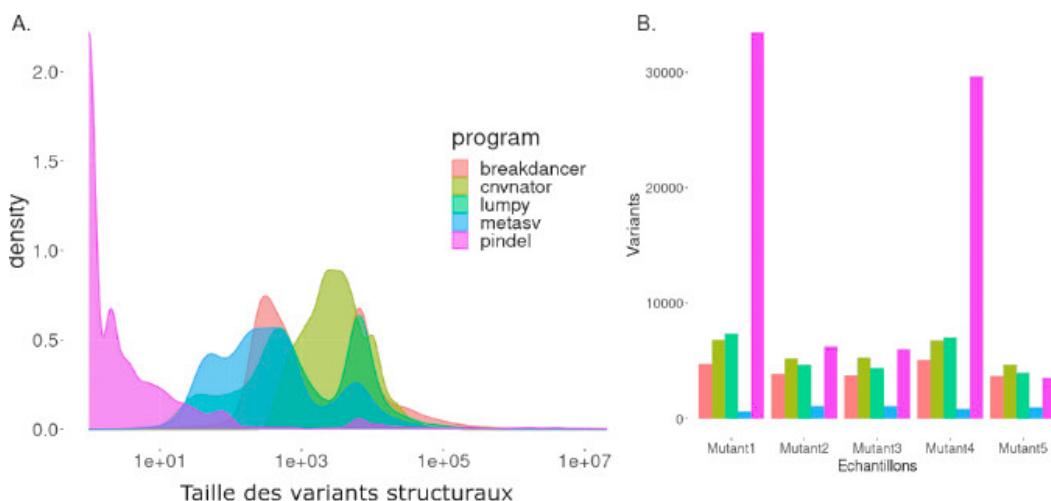
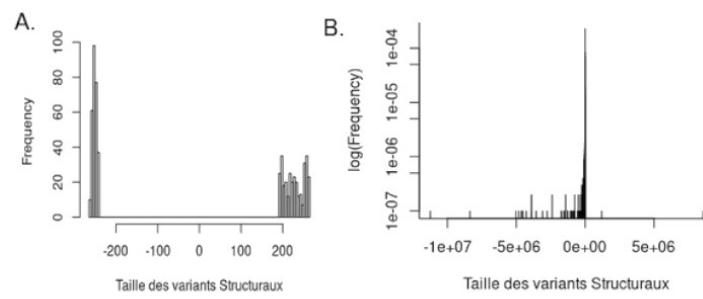


Figure 18 : Graphique récapitulatif des variants structuraux
A. Représentation de la densité du nombre de variants trouvé par les différents programmes en fonction de la taille des variants structuraux
B. Représentation du nombre de variants structuraux trouvé

un même événement de variation : taille plus ou moins identique et position quasiment semblable. Cela peut être dû au différences de méthode de détection des programmes pouvant engendrer des positions de variants légèrement différentes.

C'est pour cette raison que nous avons décidé d'utiliser metaSV, qui permet de prendre les différents variants fournis par les programmes et de les assembler afin de détecter de manière plus précise leurs positions ("breakpoints"). Nous pouvons voir sur la figure 18 que tous ces programmes n'ont pas le même spectre de détection. Ainsi on peut observer que le programme pindel détecte énormément de petits variants (inférieur à 50 bases) en comparaisons avec les autres programmes. De même, on notera que Pindel détecte beaucoup plus de variants que les autres programmes. Cela est probablement dû au fait que ce programme détecte en proportion beaucoup plus les petits variants structuraux. De plus, Pindel détecte beaucoup de positions pouvant être des positions avec des variants mais ne classe pas nécessairement les échantillons analysés comme étant variants sur ces positions. Enfin, nous avons vérifié que MetaSV effectue son travail de recouplement et de fusion des variants entre différents appels. Tout d'abord, il émet en sortie beaucoup moins de variants structuraux que les autres programmes (figure 18). On observant les résultats plus en détails, on remarque que cela est dû au fait que MetaSV regroupe plusieurs variants structuraux imprécis pour les assembler en un seul. Comme on peut le voir dans l'exemple suivant avec des variants détectés par metaSV qui regroupe des variants issus des autres programmes (exemple 1) voir même plusieurs variants issus d'un même programme (exemple 2).

Exemple de sortie des fichiers vcf de MetaSV :

```

2 11033989 . G <DEL> . PASS
CIEND=-9;6;END=11034921;SVLEN=-932;SVTYPE=DEL;CIPOS=-5,7;SVTOOL=MetaSV;SOURCES=2-11033989-2-11
034921-932-Pindel,2-11033993-2-11034921-928-Lumpy,2-11034001-2-11035000-999-CNvnator,2-11034037-2-
11034922-899-BreakDancer;NUM_SVMETHODS=3;NUM_SVTOOLS=4;VT=SV;SVMETHOD=RD,RP,SR;BD_CHR1=2
;BD_POS1=11034036;BD_ORI1=10+0-;BD_CHR2=2;BD_POS2=11034922;BD_ORI2=0+10-;BD_SCORE=99.0;BD_S
UPPORTING_READ_PAIRS=9 GT 1/1
3 25474944 . T <DEL> . PASS
END=25481413;SVLEN=-6057;SVTYPE=DEL;SVTOOL=MetaSV;SOURCES=3-25474907-3-25481390-6565-BreakDa
ncer,3-25475044-3-25481090-6046-Pindel,3-25475256-3-25481313-6057-Pindel;NUM_SVMETHODS=2;NUM_S
VTOOLS=2;VT=SV;SVMETHOD=RP,SR;BD_CHR1=3;BD_POS1=25474906;BD_ORI1=31+0-;BD_CHR2=3;BD_POS2=
25481390;BD_ORI2=1+28-;BD_SCORE=99.0;BD_SUPPORTING_READ_PAIRS=28 GT 1/1

```

Après avoir amélioré la détection de variants structuraux, nous avons, comme précédemment avec les petits variants, déterminé lesquels étaient hérités (issus de la souche de départ) et spécifiques (issus de la mutagenèse). Malgré le travail de fusion des variants structuraux effectué par MetaSV, nous avons observé des mutations très proches entre échantillons différents. Nous avons donc produit un script dédié permettant d'inférer des variants structuraux similaires (et donc potentiellement héritées) entre échantillons sur la base d'une variation des positions de début et de fin de X% par rapport à la taille d'un même variants structuraux. A la suite de plusieurs essais, nous avons déterminé qu'une valeur de 1% était suffisante pour déterminer les variants structuraux spécifiques car une trop forte valeur risquerait de détecter des variants comme étant hérités alors qu'ils sont spécifiques et induire des faux négatifs.

On peut observer sur ce graphique un faible nombre de variants structuraux hérités. Mais cela valide la méthode de détection des variants structuraux car cette méthode est capable de retrouver des variants hérités parmi les échantillons. Le fort taux de variants structuraux spécifiques pourrait s'expliquer par des mutations engendrées par la mutagenèse ou la présence d'éléments transposables lors de la croissances des plants mutés

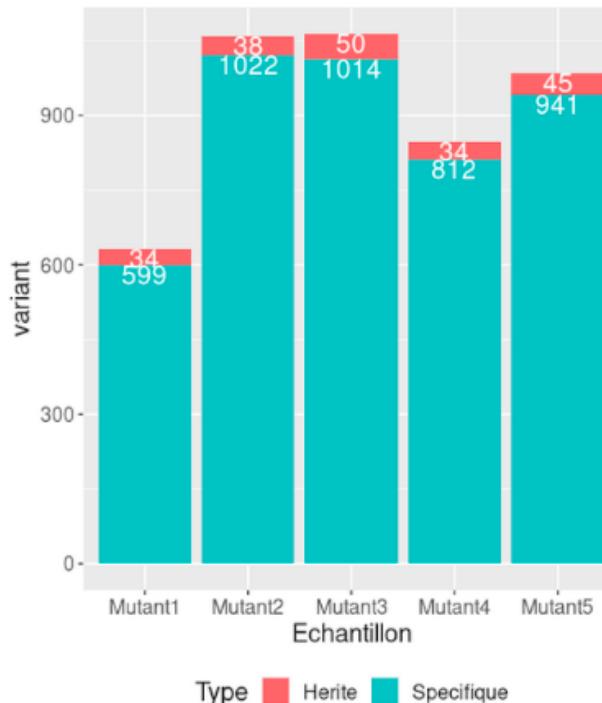


Figure 19 : Représentation du nombre de variants structuraux trouvés par échantillons en fonction du type : Hérité ou Spécifique

Exploration des résultats et premiers retour à la biologie

Lors de l'analyse des variants par snpEff, nous avions pour référence la dernière annotation de *Physcomitrium patens*. Grâce à cela, nous avons pu classer les variations par impact potentiel :

- Fort (HIGH) : Le variant est supposé avoir un impact élevé sur la protéine, provoquant probablement une troncature de la protéine, une perte de fonction ou une destruction induite par un non-sens.
- Modéré (MODERATE) : Le variant n'est pas perturbateur mais il pourrait modifier l'efficacité des protéines.
- Faible (LOW) : Le variant est peu susceptible de modifier le comportement des protéines.
- Nul (MODIFIER) : Le variant est non codant ou affecte des zones où les prévisions sont difficiles ou il n'y a aucune preuve d'impact.

On peut voir que la majorité des variants sont avec un impact nul et que cela est vrai pour tous les échantillons étudiés. En regardant la position des variants, on voit que la plupart de ceux-ci sont situés dans des régions non codantes du génome : les régions intergéniques dans les régions en amont et en aval des gènes.

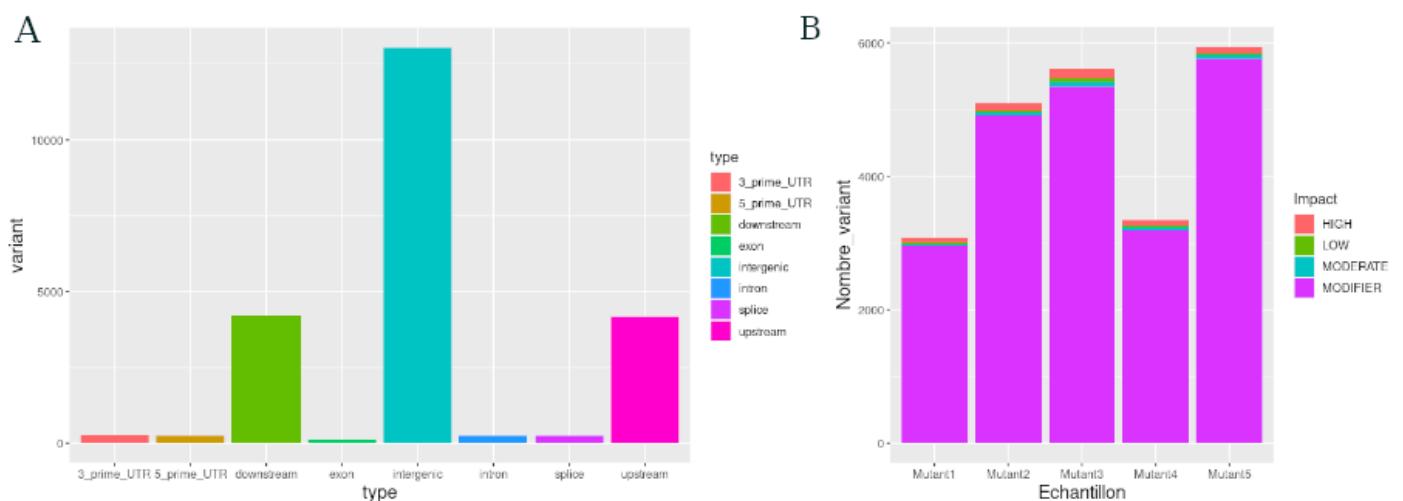


Figure 20 : Graphique récapitulatif des impacts prédict et position des variants par le programme snpEff
A. Position des variants au sein du génome
B. Impact prédict des variants par échantillons

Nous avons trouvé plusieurs variants intéressants dans les mutants en croisant les résultats avec la base de données de gene ontology. Dans les variants classés comme ayant un fort impact, nous avons pu trouver des variants ayant un impact dans :

- En grande majorité, les processus métabolique de phosphorylation de protéines
- Le réseau métabolique de l'auxine (hormone de croissance végétale permettant le contrôle du développement)
- La synthèse du siroheme, une molécule essentielle pour l'assimilation des nitrates et des sulfates et qui intervient aussi dans le stress oxydatif.
- La fabrication de la membrane cellulaire grâce à une enzyme, la 3-beta-glucuronosyltransferase.
- La fabrication de la pectine, un composant essentiel de la membrane des plantes.
- etc.

De nombreux autres gènes sont impactés de façon modérée, ceux-ci doivent être analysés par le chercheur en charge de l'expérience afin de déterminer s'il peut avoir un lien entre ces variants et les phénotypes observés, afin de pouvoir valider les variants candidats par une PCR.

Cependant, on peut remarquer, en fouillant les données, que de nombreux gènes fournis dans le fichier d'annotation sont juste hypothétiques. Beaucoup de variants ont un impact prédit comme "fort" mais n'ont pas de vrai gène lié ou d'orthologie évidente associée. Ceci est une limitation de la dernière annotation publiée du génome de *P. patens*. En effet, le format utilisé pour identifier les gènes de l'annotation (ex : Pp3c20_23130) ne correspond pas au format classique (comme NCBI ou genbank). C'est un format propre à l'annotation qui nécessite un fichier supplémentaire afin d'avoir la correspondance avec les bases de données d'ontologies de gènes.

Conclusion

L'objectif de ce stage était de développer un pipeline d'analyse d'appel et d'analyse de variants. Un des points crucial était sa facilité d'exécution par des personnes novices dans l'utilisation d'outils informatiques, afin de rendre son utilisation plus routinière au sein du laboratoire. Ce point a été correctement validé grâce au système de gestion de pipeline Nextflow et à un script développé de façon à simplifier au maximum l'analyse et à limiter les fichiers d'entrée et leurs complexités. Le pipeline nécessite uniquement des fichiers de lecture issus des différents échantillons ainsi qu'un fichier contenant le génome de référence, l'utilisateur peut aussi ajouter un fichier d'annotation de génome pour faire fonctionner le pipeline à son potentiel maximal.

En effet, le pipeline est capable de détecter une grande diversité de variants génomiques : en combinant et optimisant différents programmes de détection de variants dans une même analyse, les fichiers de résultats regroupent des snps, des indels mais aussi des variants structuraux tels que des grandes délétions, des grandes insertions, des duplications, des inversions. De plus, ce pipeline fournit également une classification de l'impact fonctionnel des variants détectés précédemment. Les variants sont classés selon leurs effets possibles, ce qui rend beaucoup plus facile la sélection de variants candidats pour le biologiste. De plus, le fichier fourni en sortie (format .csv), contenant toutes informations des variants de l'échantillon, est lisible par n'importe quel programme de tableur (excel, Libre/Open Office, etc.). Comme l'annotation des variants dépend d'un fichier d'annotation fourni en entrée par l'utilisateur, il est très facilement possible de bénéficier de la dernière mise à jour de l'annotation.

De nombreux essais de programme ont été réalisés pour améliorer la qualité des variants et rendre le spectre de détection des variants le plus large possible. Ces programmes sont tous à jour. Au besoin, de nouveau programme de détection ou mise à jour peuvent être ajoutés au pipeline grâce au système de process de nextflow.

Les différents points évoqués dans le cahier des charges ont été remplis. Il est possible d'utiliser le pipeline de façon très simple mais aussi d'accéder à des paramètres permettant de personnaliser l'analyse. Le pipeline génère une liste de variants spécifiques de chaque échantillon, issus de la mutagenèse. Mais aussi un renseignement sur l'impact fonctionnel de ceux-ci.

Cependant le stage n'est pas encore terminé et des améliorations sont toujours en cours de développement. Je travaille actuellement sur l'amélioration de la sélection des variants spécifiques contre les variants hérités afin de minimiser au maximum le nombre de faux négatifs pouvant expliquer des caractères phénotypiques observées chez la plante. Pour cela le script de sélection est en cours d'amélioration et des recherches sur des propriétés du variants sont en cours (qualité, profondeur intra échantillon en fonction de la couverture, comparaison moins stricte, etc.).

Concernant l'amélioration de l'appel de variants structuraux, d'autres programmes d'analyse de ceux-ci peuvent être ajoutés à l'analyse de MetaSV, des tests sont en cours pour vérifier la pertinence de ces programmes. Enfin, une simplification du script Nextflow permettant d'avoir une meilleure maintenabilité par des futurs bio-informaticiens est en cours ainsi qu'une optimisation des autres scripts pour ce pipeline.

D'autres jeux de données vont être soumis à ce pipeline permettant de confirmer ou d'infirmer la manière de sélectionner les variants. L'un de ces jeux de données contient des variants structuraux qui ont été validés par PCR et l'autre a un design expérimental un peu différent, la souche de départ n'a pas été séquencée et ne contient que 2 échantillons mutés.

Références bibliographiques

1. Ashton, N. W. & Cove, D. J. The isolation and preliminary characterisation of auxotrophic and analogue resistant mutants of the moss, *Physcomitrella patens*. *Mol. Gen. Genet. MGG* **154**, 87–95 (1977).
2. Rensing, S. A., Goffinet, B., Meyberg, R., Wu, S.-Z. & Bezanilla, M. The Moss *Physcomitrium* (*Physcomitrella*) *patens*: A Model Organism for Non-Seed Plants. *Plant Cell* **32**, 1361–1376 (2020).
3. Halligan, D. L. & Keightley, P. D. Spontaneous Mutation Accumulation Studies in Evolutionary Genetics. *Annu. Rev. Ecol. Evol. Syst.* **40**, 151–172 (2008).
4. Nakamura, M., Nunoshiba, T. & Hiratsu, K. Detection and analysis of UV-induced mutations in the chromosomal DNA of Arabidopsis. *Biochem. Biophys. Res. Commun.* **554**, 89–93 (2021).
5. Lang, D. et al. The *Physcomitrella patens* chromosome-scale assembly reveals moss genome structure and evolution. *Plant J. Cell Mol. Biol.* **93**, 515–533 (2018).
6. Ultrafast and memory-efficient alignment of short DNA sequences to the human genome | Genome Biology | Full Text. <https://genomebiology.biomedcentral.com/articles/10.1186/gb-2009-10-3-r25>.
7. Li, H. Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM. *ArXiv13033997 Q-Bio* (2013).
8. Li, H. et al. The Sequence Alignment/Map format and SAMtools. *Bioinforma. Oxf. Engl.* **25**, 2078–2079 (2009).
9. Broad Institute. Picard. <http://broadinstitute.github.io/picard>. Accessed 19 Sept 2018
10. McKenna, A. et al. The Genome Analysis Toolkit: A MapReduce framework for analyzing next-generation DNA sequencing data. *Genome Res.* **20**, 1297–1303 (2010).
11. Scaling accurate genetic variant discovery to tens of thousands of samples | bioRxiv. <https://www.biorxiv.org/content/10.1101/201178v3>.
12. Ye, K., Schulz, M. H., Long, Q., Apweiler, R. & Ning, Z. Pindel: a pattern growth approach to detect break points of large deletions and medium sized insertions from paired-end short reads. *Bioinformatics* **25**, 2865–2871 (2009).

13. Fan, X., Abbott, T. E., Larson, D. & Chen, K. BreakDancer – Identification of Genomic Structural Variation from Paired-End Read Mapping. *Curr. Protoc. Bioinforma. Ed. Board Andreas Baxevanis Al* **2014**, (2014).
14. Abyzov, A., Urban, A. E., Snyder, M. & Gerstein, M. CNVnator: an approach to discover, genotype, and characterize typical and atypical CNVs from family and population genome sequencing. *Genome Res.* **21**, 974–984 (2011).
15. Layer, R. M., Chiang, C., Quinlan, A. R. & Hall, I. M. LUMPY: a probabilistic framework for structural variant discovery. *Genome Biol.* **15**, R84 (2014).
16. Mohiyuddin, M. *et al.* MetaSV: an accurate and integrative structural-variant caller for next generation sequencing. *Bioinforma. Oxf. Engl.* **31**, 2741–2744 (2015).
17. Bankevich, A. *et al.* SPAdes: A New Genome Assembly Algorithm and Its Applications to Single-Cell Sequencing. *J. Comput. Biol.* **19**, 455–477 (2012).
18. Cingolani, P. *et al.* A program for annotating and predicting the effects of single nucleotide polymorphisms, SnpEff: SNPs in the genome of *Drosophila melanogaster* strain w1118; iso-2; iso-3. *Fly (Austin)* **6**, 80–92 (2012).
19. Di Tommaso, P. *et al.* Nextflow enables reproducible computational workflows. *Nat. Biotechnol.* **35**, 316–319 (2017).
20. MultiQC: summarize analysis results for multiple tools and samples in a single report | Bioinformatics | Oxford Academic.
<https://academic.oup.com/bioinformatics/article/32/19/3047/2196507>.