MASTER SCIENCES
ET NUMERIQUE POUR LA SANTÉ

Rapport de stage

Master 1 - BCD

23 avril 2018 - 23 aout 2018

Intégration de données de reséquençage de diversité : constitution et caractérisation d'un jeu de données commun à partir de plusieurs études de reséquençage massif sur le riz

Encadrant :

Manuel Ruiz et Gaëtan Droc

Tuteurs Pédagogique :

Alban Mancheron

Bellot Clément

4 juillet 2018

Table des matières

Abstract

Dans le but d'acquérir les annotations d'une nouvelle version d'un génome de riz issus d'une étude de reséquençage, nous avons cherché des méthodes permettant de transférer les positions des annotations d'une version antérieure. Deux méthodes ont été étudiés : une méthode se reposant sur un alignement complet des deux versions du génome pour ensuite transposer les positions en fonctions du fichier d'alignement et une méthode réalisant un mapping des séquences annotés avec les régions flanquantes. La deuxième méthode a été utilisé dans le script python TransPo-RG qui permet de transférer de manière sûre les SNP. Des améliorations sont nécessaires pour les autres types d'annotation et un benchmark devra être réaliser afin de conclure sur l'efficacité de ce script et de cette méthode.

1 Introduction

De plus en plus d'étude de diversité génétique sont réalisé et génèrent des quantités de données conséquentes. Ces données, afin de réaliser des études plus poussés de comparaison de génome ou pour obtenir un maximum d'information de meilleure qualité, doivent être intégré dans un jeu de données commun. Cela a été réalisé pour 3 000 accessions de riz cultivé[1] (*Oryza sativa*) par exemple.

Grâce aux avancées technologiques des techniques de séquençage, les génomes issus de reséquençage sont de plus en plus précis et permettent des analyses plus précises. Ainsi, une mise à jour des génomes de référence doit être réalisé afin d'être remplacé ou amélioré par une version plus récente.

La problématique qui se pose est le temps nécessaire pour réaliser cette mise à jour. En effet, les analyses sont longues et amène souvent à délaisser les nouvelles données pour continuer à travailler sur le génome de référence qui lui est bien connu.

La structure globale d'un génome ne variant pas ou très peu d'une version à une autre, refaire des analyses comme du SNP calling et l'annotation de gène par exemple semble peu efficace en terme de gain par rapport au temps que cela demanderait. De plus, cela pourrait nous permettre de comparer les annotations transférés avec les potentielles nouvelles annotations si on décide de les générer. En effet, il est possible que certaines annotations ne soit pas détecter par les nouvelles analyses pour diverses raison notamment si les outils utilisés ne sont pas les mêmes que ceux utilisés pour annoter la version antérieur du génome.

L'idée serait de pouvoir transposer les annotations déjà généré pour le génome de référence sur le nouveau génome issu de reséquençage mais pas seulement, comme préciser ci-dessus, la comparaison de fichier d'annotation de la même version ou sur une version antérieur pourrait être intéressant pour obtenir le plus possible d'information précise sur un génome d'intérêt.

2 Méthodes

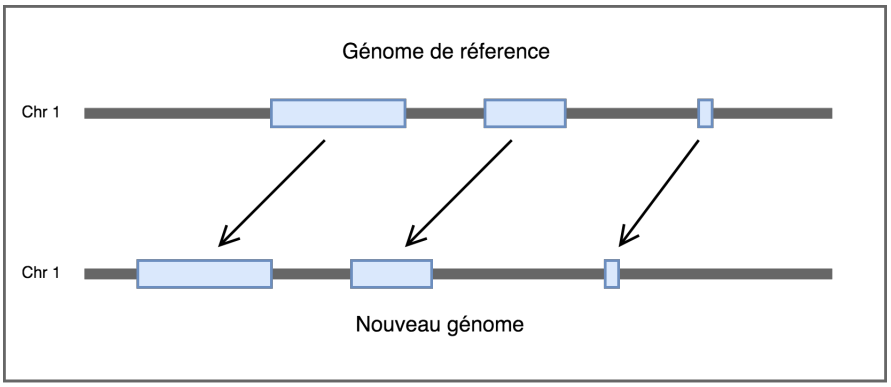

FIGURE 1: Schéma illustrant les différences de position des annotations entre deux génomes.

On dispose du génome de référence au format FASTA et d'un ou plusieurs fichiers tabulés associé au format VCF, BED ou GFF. On souhaite alors transposer les données des fichiers tabulés sur un autre génome plus récent (ou antérieur pour comparer) c'est-à-dire issu d'un reséquençage On souhaite obtenir un fichier tabulé pour le génome reséquencé avec les nouvelles positions des annotations convertie pour le nouveau génome. Cela nous permettrait de pouvoir comparer les différents fichiers d'analyses entre deux versions du génome (confirmer la présence d'un SNP par exemple) et de pouvoir utiliser le nouveau génome avec ses annotations.

On peut distinguer deux méthodes pour réaliser ce transfert de positions : la première est de réaliser un alignement local d'un génome sur l'autre (nouvelle assemblage sur la référence) et d'utiliser le fichier d'alignement comme fichier de correspondance par comparaison. Cela nous permet de n'avoir besoin que du fichier d'alignement et du fichier qui contient les positions des annotations pour réaliser la conversion. En effet, il suffit ensuite de récupérer les positions de match sur le nouveau génome assemblé et de modifier le fichier tabulé avec les nouvelles positions.

L'université de Californie Santa Cruz (UCSC) a développé un outil utilisant la première méthode qui permet de convertir les positions d'un fichier d'annotation d'un génome à un autre : LiftOver[2]. Cet outil utilise un fichier d'alignement au format CHAIN[3] qui permet de décrire un alignement par pair qui autorise les gaps dans les deux séquences simultanément. Le format CHAIN enregistre les blocs sans gaps et note le nombre de gap présent jusqu'au prochain bloc sans gap. On perd alors des informations sur la nature des blocs avec gaps, notamment la position de chaque gap. Cet outils nécessite de réaliser un alignement entre les deux génomes, ils utilisent leur logiciel BLAT. De plus, plusieurs scripts doivent être utilisés afin de passer d'un fichier d'alignement classique à un fichier chain et cela nécessite la création d'un workflow spécifique. L'intégralité des deux génomes est aussi requis ce qui n'est pas optimale lorsque l'on souhaite transférer les positions de SNP par exemple.

CrossMap[4] (Fig. 2) est un outil qui se base lui aussi sur un fichier d'alignement de type chain généré par les scripts de l'UCSC. À partir du fichier chain, il va organiser la

structure dans un arbre d'intervalle qui permet de plus facilement réaliser des requêtes à partir du fichier contenant les annotations. CrossMap réalise un mapping des positions des annotations sur l'alignement entre les deux assemblages en requêtant l'arbre d'intervalle.

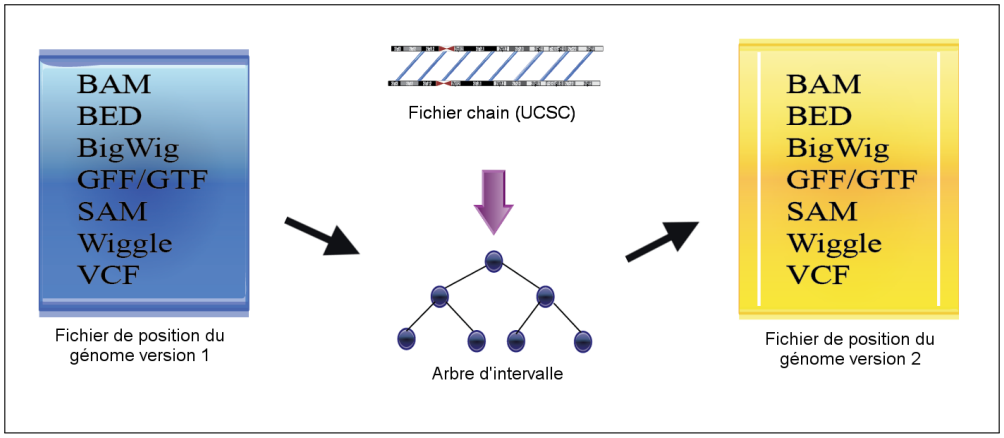

FIGURE 2: Schéma du workflow de CrossMap. Adapté depuis
http://crossmap.sourceforge.net/

On retrouve alors les mêmes contraintes qu'avec l'outil LiftOver notamment sur le fait de devoir généré un fichier chain et donc de devoir réalisé un alignement puis d'utiliser les différents scripts de l'UCSC.

On peut également cité segment_liftover[5] qui est un outil récent (mars 2018) en python qui se base lui aussi sur LiftOver de l'UCSC.

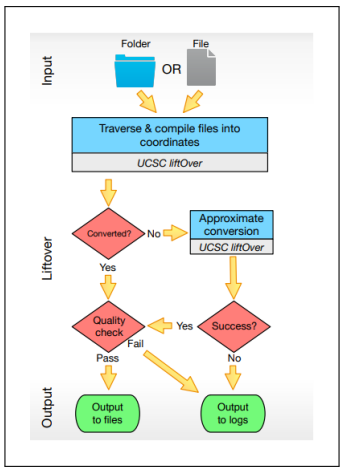

FIGURE 3: Schéma du workflow de segment_liftover. Adapté depuis
https://www.biorxiv.org/content/early/2018/03/01/274084

QOD[6] (Fig. 4)est un autre outil utilisant cette méthode c'est-à-dire qui se base sur un fichier compilant tout les blocs qui rentrent dans le seuil défini par l'utilisateur. Toute position d'annotation qui se trouve dans ces blocs seront converti et enregistré dans un nouveau fichier d'annotation pour le nouveau génome. Donc un alignement entre les deux génomes doit avoir été créé au préalable et cette étape est à prendre en compte dans le temps nécessaire pour réaliser ce transfert.

La seconde méthode est de réaliser un mapping sur le nouveau génome avec seulement les séquences de nos annotations. Le but ici étant de récupérer les séquences flanquantes des annotations fournis dans les fichiers tabulés et de les mapper sur le génome cible. Par exemple, si l'on a un SNP à la position 90 dans la première référence, on va récupérer la séquence de 39 à 140 (50 bp de part en part par défaut) que l'on va ensuite mapper sur la deuxième référence. Dans le cas d'un CDS, on soustrait 51 à la position de départ (50 + 1 pour inclure le SNP) et on ajoute 50 à la position de fin. On obtient ainsi un fichier d'alignement que l'on traite de la même façon que celui de la première méthode, c'est-à-dire que l'on a juste à extraire les nouvelles positions pour recréer un fichier tabulé (du même format que celui passé en entrée).

Cette méthode à l'avantage d'être théoriquement plus rapide notamment grâce au fait que l'on n'a pas besoin d'aligner tout le génome sur le nouveau mais seulement les séquences d'intérêt avec les régions flanquantes associés.

Cette méthode ainsi décrite est utilisé dans mon script : TransPo-RG (TPRG) pour *Transfer of Position to Resequenced Genome*.

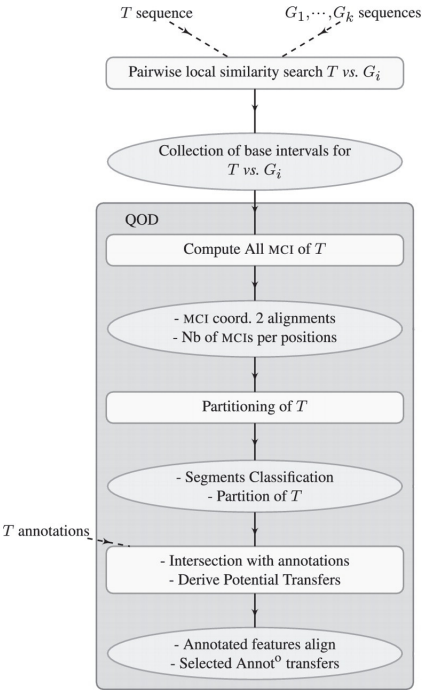

FIGURE 4: Workflow du fonctionnement de l'outil QOD. Le génome T est aligné au préalable (avec BLAST, BLAT ou YASS) sur les séquences des génomes de référence Gi. Adapté depuis https://academic.oup.com/nar/article/39/15/e101/1016355.

3 Description du script TransPo-RG

3.1 Librairie python

Pour faciliter la gestion de nos différentes entrées, on utilise des librairies python : argparse, biopython et pybedtools :

- biopython permet d'encapsuler les fichiers fasta. On l'utilise ainsi pour récupérer efficacement la taille de chaque séquence dans la première référence fasta, utile pour la méthode getflank() dans le cas d'un fichier vcf (qui modifie les positions dans les fichiers tabulés).
- pybedtools permet d'encapsuler tout les fichiers tabulés et d'utiliser plusieurs outils de Bedtools comme getfasta (.sequence() ici) et slop (qui permet de récupérer les séquences et les séquences flanquantes).
- argparse permet de gérer les arguments à l'aide d'option. On peut ainsi exécuter le script avec l'option -h (ou --help) pour afficher les options disponibles. Par exemple, l'option -t (ou --type) permet de récupérer seulement les annotations d'un type souhaité à partir d'un fichier .gff (-t cds,gene permet de récupérer tout les gènes et

tout les CDS mais pas les exons,mRNA,etc). Trois arguments sont obligatoires : les deux références fasta et un fichier tabulé.

On utilise également différentes librairies pour gérer plus facilement les sorties : tempfile, warnings et subprocess :

- tempfile permet la création de fichier temporaire afin de limiter le nombre fichier crée en sortie.
- warnings permet de gérer quelque warning si l'utilisateur rentre des options non compatibles avec les formats de fichier rentré par exemple.
- subprocess permet de gérer les quelques appels systèmes dont on ne peut se passer comme pour réaliser l'alignement avec bwa mem.

On a aussi crée un autre module appelé version qui permet de gérer les numéros de version et d'afficher la plus récente et la date de création.

3.2 Fonctionnement

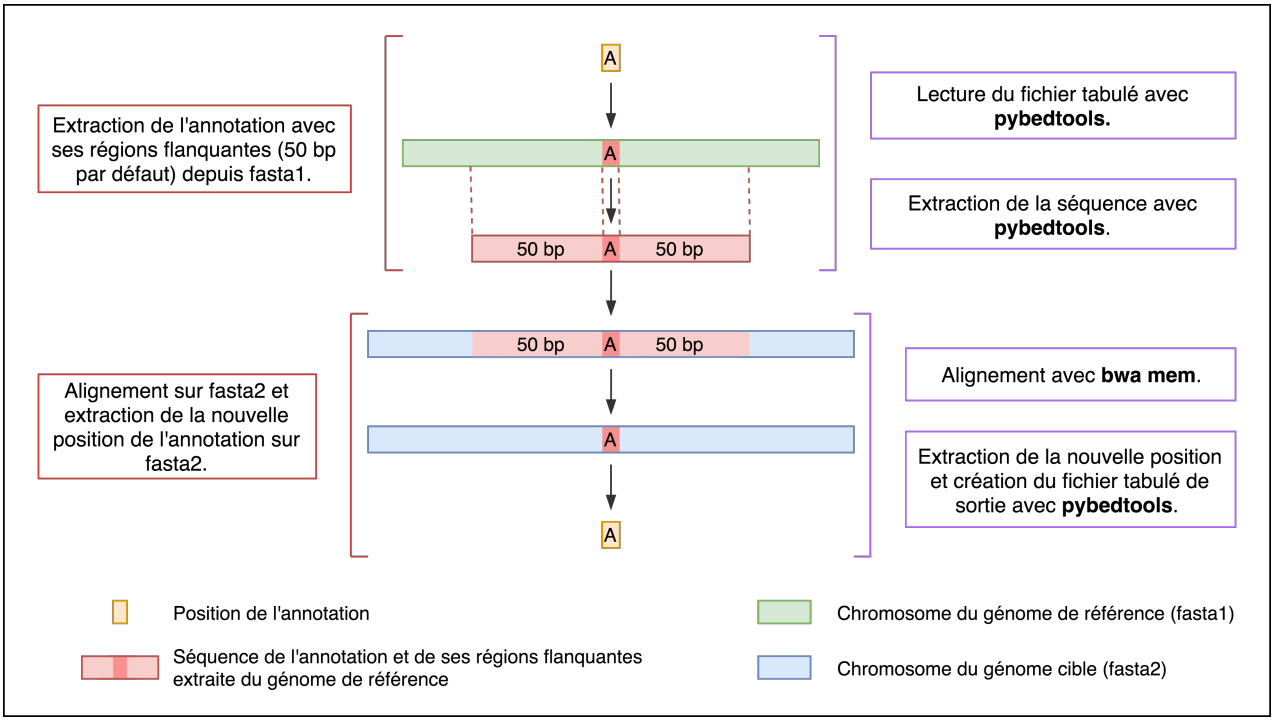

FIGURE 5: Schéma synthétique du fonctionnement du programme pour un SNP.

Fichiers en entrée :

- Génome de référence au format FASTA (argument --fasta1).
- Fichier tabulé au format VCF, BED ou GFF contenant les annotations du génome fasta1 (argument --tabinput).
- Génome sur lequel on souhaite transposer les annotations (argument --fasta2).

Fichiers en sortie :

- Fichier tabulé au format VCF, BED ou GFF en fonction du fichier tabulé passé en entrée. Il contient les annotations avec les positions convertit pour le génome fasta2.

```
chromosome_1    37307   .       T       A       1179.77 .       AC=2;AF=1.00;AN=2;DP=28;FS=0.
000;MLEAC=2;MLEAF=1.00;MQ=60.00;MQ0=0;QD=28.18     GT:AD:DP:GQ:PL  1/1:0,28:28:84:1208,84,0
chromosome_1    39457   .       G       C       477.77  .       AC=2;AF=1.00;AN=2;DP=15;FS=0.
000;MLEAC=2;MLEAF=1.00;MQ=29.00;MQ0=0;QD=31.85     GT:AD:DP:GQ:PL  1/1:0,15:15:45:506,45,0
```

FIGURE 6: Capture d'écran de deux SNP de la référence fasta1 dans un fichier VCF.

Tout d'abord on va vérifier que les trois fichiers requis sont bien spécifié (les trois fichiers décrit ci-dessus) ainsi que les différentes versions des outils utilisés sont bien installés (*bwa* et *bedtools*). On vérifie également le format du fichier tabulé pour pouvoir appeler les méthodes adéquates en fonction. Ensuite on récupère les identifiants et les tailles des chromosomes présent dans le fasta1 ; cela nous servira lorsque l'on modifiera les positions dans le fichier tabulé pour inclure les régions flanquantes.

Si le fichier tabulé est au format GFF et que l'on souhaite filtrer les types d'annotations avec l'option --type, on utilise la méthode cutGff() qui générera un fichier temporaire de type GFF qui sera alors utilisé à la place du fichier passé en entrée.

La méthode getFlank() génère un fichier temporaire tabulé contenant les nouvelles positions avec les régions flanquantes incluses. La taille des régions à sélectionner est par défaut de 50 nucléotides de par et d'autres de l'annotation. Elle peut être spécifié avec l'option -b (--flank). Ainsi, dans le cas d'un SNP, on sélectionnera 101 nucléotides par défaut. Pour les fichiers GFF et BED, on utilise l'outil slop de bedtools qui étend les positions dans chaque direction. Pour les fichiers VCF on doit rajouter une colonne pour la position stop, on génère alors un fichier tabulé intermédiaire contenant l'identifiant du chromosome, la position de début et la position de fin (Fig. 3).

```
chromosome_1    37256   37357
chromosome_1    39406   39507
```

FIGURE 7: Capture d'écran du fichier tabulé intermédiaire contenant les postions des deux régions flanquantes des SNP.

L'outil getfasta de bedtools (méthode sequence() de pybedtools) permet de sélectionner les séquences spécifiés dans le fichier temporaire tabulé généré au préalable. Cet outil nécessite que l'identifiant du chromosome soit similaire dans le fichier FASTA et dans le fichier tabulé. On vérifie donc que c'est bien le cas sinon on regénére un fichier temporaire tabulé en modifiant le préfixe du fichier tabulé pour que les deux concordent.

Ensuite, on va réaliser l'alignement de ces séquences sélectionnés sur le fichier fasta2. Pour cela, on doit indexer le fichier fasta2. Cette opération est coûteuse en ressources

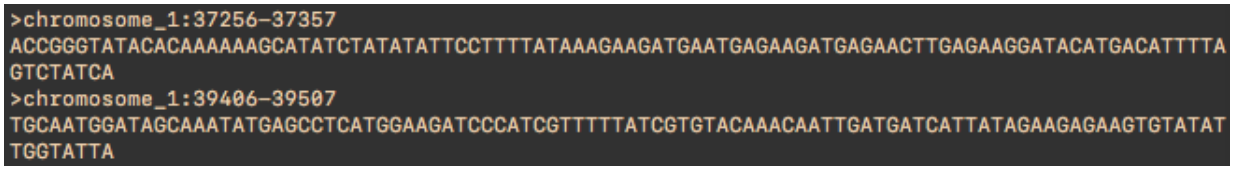

FIGURE 8: Capture d'écran du fichier contenant les séquences des régions flanquantes des SNP. Il est au format FASTA et c'est ce fichier qui sera aligné sur le nouveau génome.

matérielle, c'est pourquoi on la réalise uniquement si l'utilisateur le souhaite (option -i) et si l'index n'existe pas déjà (l'option -ii permet de forcer la création de l'index). Après alignement avec l'algorithme *bwa mem*, on obtient un fichier d'alignement .sam dont on extrait les positions moins la longueur des régions flanquantes et on génère un fichier tabulé contenant ces nouvelles positions avec toute les annotations présentes dès le début et que l'on ne modifie pas.

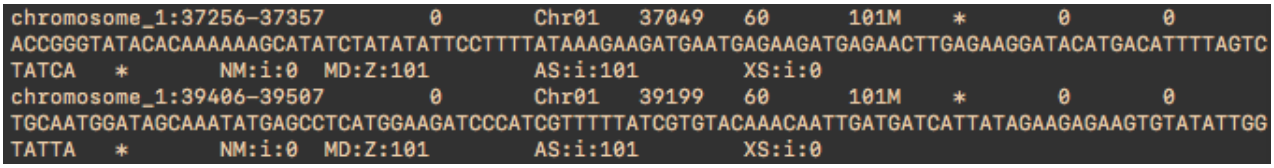

FIGURE 9: Capture d'écran du fichier d'alignement.

Dans le cas d'un fichier GFF contenant les annotations de type CDS et (ou gene), la méthode getPosCds([file]) qui est appelé si l'option -c (--cds) est passé permet de récupérer les postions relatives des CDS dans les ARNm (ou dans les gènes). Cela nous permet par la suite de comparer ces positions relatives entre les deux versions du fasta pour vérifier l'intégrité des ARNm (ou gène) dans le fichier fasta2 et de regénérer un fichier tabulé contenant uniquement les ARNm (et/ou gènes) qui correspondent aux critères de sélections.

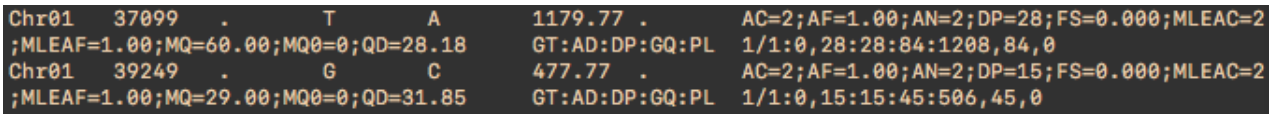

FIGURE 10: Capture d'écran du fichier VCF généré en sortie.

Tout les fichiers créés sont stockés dans le dossier 'result/' par défaut ou dans le dossier précisé avec l'option -d (--directory). En sortie, lors d'une utilisation basique, un seul fichier est crée : le fichier tabulé contenant les annotations, au même format que le fichier tabulé passé en entré, transposer sur le fichier fasta2. Par défaut, ce fichier est nommé [nom de fasta2]_out.[format tabinput] mais peut être défini par l'utilisateur avec l'option -o (--output). Si l'option -c (--cds) est passé et que les conditions sont vérifiés, un autre fichier tabulé est crée contenant les annotations filtrés, son nom est le même nom que le fichier de sortie mais avec filtered_ en prefix.

Si l'option -n (--notempfile) est précisé, trois autres fichiers sont alors créés :

- Le fichier tabulé intermédiaire contenant les positions modifiés pour récupérer les régions flanquantes. (nommé [nom du fichier tabulé en entré]_out.[format du fichier tabulé en entré])

- Le fichier contenant les séquences sélectionnées au format FASTA (nommé [nom du fichier fasta1]_selected_[format du fichier tabulé en entré].fasta)
- Le fichier d'alignement (nommé aln_out_[format du fichier tabulé en entré].sam)

3.2.1 Bonne pratique

Afin de versionner le code du script, un répertoire git a été mis en place ainsi que des tags associé à certain commit dans le but d'obtenir et de mettre à jour un numéro de version. Le module python version a été créé dans cette optique la, avec l'option –update (-u) du script TransPo-RG, il permet la création d'un fichier .version qui enregistre la date et le numéro de version actuel en faisant appel aux logs du git (git log –tags). En exécutant le script avec l'option –version (-ver), ce module permet l'affichage de la date de dernière mise à jour de la version et du numéro de version en consultant le fichier .version.

Les commentaires ont été écrit à l'aide de docstring, c'est une syntaxe particulière qui permet à des outils de les reconnaître et de les extraire. Cela permet de pouvoir générer automatiquement une documentation dynamique du script avec des outils de documentation comme Sphynx ou Doxygen.

3.3 Problèmes rencontrés

Un des principales soucis rencontré a été la prise en charge des différents format d'entrée (VCF, BED ou GFF3). En effet, chaque fichier à sa spécificité et donc doit subir un traitement différent. Bien que la librairie pybedtools permet d'encapsuler ces différents formats, leur manipulation reste compliqué. Pour commencer, les VCF peuvent contenir des lignes de meta-information et un entête. Ils faut donc pouvoir les transférer dans le nouveau fichier en sortie. De plus, un fichier VCF ne possède pas de position stop, il faut se référer aux champs REF et ALT pour calculer la longueur de l'annotation.

Un autre problème est la différence de position en fonction de si les positions dans le fichier sont "1-based" ou "0-based" (Fig. 11). En effet, le VCF, le GFF et le SAM sont "1-based" tandis que le BED est "0-based". Un traitement particulier est donc nécessaire lors de l'ajout et du retrait (à l'écriture) des régions flanquantes.

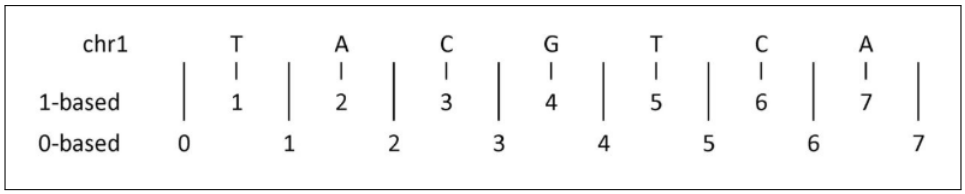

FIGURE 11: Schéma montrant les différences de position entre un fichier 1-based et un fichier 0-based. Adapté depuis https://www.biostars.org/p/84686/

Pour les SNP, on ne récupère que les matchs parfait. C'est-à-dire que si on prend 50 nucléotides de part et d'autre, il faut que l'alignement soit parfait pour les 101 nucléotides sélectionnés. Pour contrôler ceci, on va récupérer le CIGAR qui témoigne de l'alignement. Dans notre exemple, il faut que le CIGAR soit 101M (pour 101 Matchs). Mais cela n'ait

pas suffisant car bwa mem accepte des mismatchs sans le faire apparaître dans le CIGAR. Il est donc nécessaire de vérifier dans les tags si on a bien 101 matchs parfaits.

Un autre problème de cette méthode des régions flanquantes est qu'il faut obligatoirement que l'annotation soit entouré par suffisamment de nucléotides pour pouvoir être sélectionnés. Dans l'exemple de ci-dessus, si on a un SNP à la position 30, la séquence extraite sera de 0 à 80. Elle fera seulement 80 nucléotides de long et ne rentrera pas dans le cas des 101 matchs parfait. De même en fin de chromosome.

4 Discussion

4.1 Limite

La limite de cette technique est que l'on ne pourra jamais obtenir d'annotations plus qualitatives et en plus grand nombre alors que l'intérêt d'un reséquençage est justement d'acquérir de nouvelle information sur le génome. En effet, au mieux, on pourra confirmer la présence ou non d'une annotation et avoir le même nombre d'annotation. C'est une solution temporaire pour pouvoir se faire une idées de la qualité du reséquençage ou appuyer les précédentes analyses sur une version antérieure ou encore pour pouvoir réaliser des analyses mineurs à partir des fichiers nouvellement générés.

4.2 Perspective

Afin de quantifier les différentes méthodes, il serait intéressant de réaliser un benchmark des outils cités ci-dessus. Cependant, cela requiert du temps notamment pour pouvoir réaliser des tests similaires puisque les outils ne prennent pas tous les mêmes format de fichier en entrée.

Dans une optique d'avoir un génome annoté le plus précisément possible, on pourra par la suite fusionner des fichiers d'annotations au format VCF. En gardant l'intersection des différents fichiers des différentes version d'un génome on pourrait confirmer avec exactitude les SNP présents ou pas.

Une interface graphique serait la bienvenue pour le script TransPo-RG pour faciliter l'utilisation à une personne non-initier au ligne de commande. L'implémentation sur Galaxy, plateforme web hébergeant des outils, permettrait de facilement fournir cette interface ainsi que de faciliter la diffusion.

Ces perspectives seront explorés lors de la seconde moitié du stage.

5 Conclusion

Dans une optique d'intégrer de plus en plus de données à partir de plusieurs études de reséquençage sur le riz, de nombreuses analyses sont nécessaires et sont souvent coûteuse en temps et en ressources. Une technique de transfert de postions entre deux génomes est proposé afin de générer un fichier d'annotation pour une version d'un génome à partir d'un fichier d'annotation sur une autre version de ce génome. Deux méthodes pour réaliser ce transfert ont été décrite et nous avons décidé d'utiliser la seconde, à savoir d'extraire les régions flanquantes des annotations et de réaliser un mapping sur la version cible.

Le script python TransPo-RG permet de transférer efficacement les SNP d'une version d'un génome à une autre. Un benchmark est a réalisé par la suite pour déterminer qualitativement si cette méthode est optimale pour ce type d'opération. Plusieurs difficultés sont apparus, notamment la gestion des mismatchs. Pour le cas d'un SNP, seul les matchs parfaits sont conservés donc ce problème n'existe pas mais dès que l'on veut transférer les positions d'un gène par exemple cela ce complique.

Des améliorations sont à réaliser sur ce script et une réfléxion plus poussés sur la méthode utilisés est à faire, notamment pour le cas des annotations autre que les SNP.

Références

[1] 3,000 Rice Genomes Project. GigaScience, 2014, 3 : 7

[2] Robert M. Kuhn, David Haussler and W. James Kent. The UCSC genome browser and associated tools. Briefings in Bioinformatics, 2012, 14 : 2, 144-161

[3] W. James Kent, Robert Baertsch, Angie Hinrichs, Webb Miller, and David Haussler. Evolution's cauldron : Duplication, deletion, and rearrangement in the mouse and human genomes. PNAS, 2003, 100 : 20, 11484-11489

[4] Hao Zhao, Zhifu Sun, Jing Wang, Haojie Huang, Jean-Pierre Kocher and Liguo Wang. CrossMap : a versatile tool for coordinate conversion between genome assemblies. Bioinformatics, 2014, 30 : 7, 1006–1007

[5] Bo Gao, Qingyao Huang, and Michael Baudis. segment liftover : a Python tool to convert segments between genome assemblies. bioRxiv, 2018

[6] Alban Mancheron, Raluca Uricaru and Eric Rivals. An alternative approach to multiple genome comparison. Nucleic Acids Research, 2011, 39 : 15, 101