Victor LEMAIRE
Alexis RADICE
Thi Thuy Trang NGUYEN

Projet intégré Bio-informatique
Compte rendu

INTRODUCTION :

Le but du projet était de concevoir un pipeline informatique qui collecte et opère un nettoyage, une identification et une analyse de la masse de données disponibles des génomes de levure servant à faire du pain ou de la bière. Ceci pour obtenir une table de génotype qui servirait aux chercheurs en biologie pourraient estimer les histoires évolutives de la levure naturelle et domestiquées.

Les molécules d'ADN sont les porteurs de l'information génétiques. Elles sont composées de chromosomes. Eux-mêmes, fait de nucléotides A, C, G, T. Toutes les molécules d'ADN d'un être vivant forment son génome. L'expression de ce dernier donne une partie du phénotype (ensemble des caractères observables d'un individu : morphologiques, physiologiques ou biochimiques). A la création d'un nouvel individu, le génome peut muter. C'est-à-dire qu'il a une ou plusieurs parties de son ADN qui a été modifiée, supprimée ou rajoutée. Si cette mutation est bénéfique dans l'environnement de l'être vivant alors elle proliférera. En comparant les génomes de ces différentes souches, on peut retracer l'histoire évolutive de l'espèce. Dans le cas des levures étudiées, leur environnement était contrôlé par l'homme forçant le sens de leur évolution pour répondre à ces besoins. C'est la domestication. On souhaite retracer leur évolution, pour ce faire on va pouvoir étudier leur génome qui a été séquencé.

Ici, le séquençage est effectué par la technologie Illumina qui consiste à découper l'ADN en brin de petites tailles, de les rendre monobrin avec des oligonucléotide aux extrémités. Ces derniers permettent de s'accrocher à une plaque de verre appelé cellule de flux. Sur cette cellule on vient recréer le brin complémentaire puis retirer le brin d'origine. Le complémentaire s'hybride à un autre point d'attache de la cellule et un nouveau brin complémentaire est créé puis les deux brins se détachent. Cette étape est répétée plusieurs fois jusqu'à remplir la cellule. Une fois fait, on supprime tous les brins complémentaires pour ne garder que les copies de l'original. Cette partie finie, on commence le vrai séquençage en incorporant un type de nucléotide à la fois qui émet de la lumière de couleur précise ce qui permet de les déterminer.

D'un point de vue informatique, l'objectif était de créer un pipeline, c'est-à-dire une série de tâches qui s'exécutent les unes après les autres, sans intervention extérieure et qui peut inclure des vérifications, de la création de fichiers intermédiaires, des choix de l'utilisateur et de la parallélisation pour minimiser sa durée. Dans ce pipeline, écrit en Python, on a dû intégrer des script R pour obtenir des histogrammes et des diagrammes, ou encore des algorithmes comme Gatk, bcfTools, BedTools, Bwa, samtools, Vcftools.

MATERIEL ET METHODE :

Présentation du pipeline :

Notre pipeline est écrit en python 3.9.2 sous linux (Manjaro). Il fait appel à des scripts en R et à une multitude d'outils (scripts, algorithmes etc.) externes spécialisé pour la Bio-Informatique. A partir d'un fichier TSV, le pipeline produit de nouveaux types de fichiers, temporaire ou non, rangés dans des dossiers ainsi que des tableaux, des graphiques, des documents texte et histogrammes. Le TSV contient les informations de chaque échantillon ligne par ligne, dont le simple-alias, le lien de rechargement et le md5. Si l'échantillon est un single-end, il n'y aurait entre autres qu'un seul lien et un seul md5, sinon il y en a 2 et c'est un paired-end. Il est composé de 4 grandes parties : la collecte, le nettoyage, l'identification, l'analyse.

Grandes Etapes :

Collecte : Dans cette partie, à partir d'un TSV, nous stockons pour chaque échantillon le ou les liens de téléchargement de leurs fichiers fastq.gz ainsi que leurs md5 et leur sample_alias. A partir du tableau des liens nous téléchargeons tous les fichiers .fastq.gz et nous vérifions que les md5 correspondent aux md5 récupérés précédemment. Les sample_alias servirons ultérieurement. A la fin nous affichons dans le terminal s'il y a eu des erreurs de md5 et le nom des fichiers concernés.

Nettoyage : Ici, on commence par le mapping (c'est-à-dire aligner les séquences des échantillons avec le génome de référence). Pour ce faire, on commence par indexer le génome de référence ce qui génère des fichiers nécessaires pour l'utilisation de bwa mem. Ensuite, on applique cet algorithme sur les fichiers .fastq.gz en indiquant leur read groupe respectif (indispensable pour plus tard). On obtient donc des fichiers .sam.gz. On les décompresse instantanément pour avoir des .sam car ils doivent l'être pour la suite.

Avant de faire le marquage, on doit convertir les .sam en .bam pour qu'ils puissent être lu par l'algorithme MarkDuplicatesSpark de l'outil Gatk. Pour ce faire, on utilise d'abord l'algorithme samtools view et on précise avec l'option -b que la sortie est sous format .bam. Puis l'on replace les séquences en fonction de leurs coordonnées d'alignement sur chaque chromosome avec la commande samtools sort.

Une fois ces transformations effectuées, on peut appliquer le marquage qui consiste à indiquer les duplicats pour les outrepasser pendant l'identification de variants. Il se fait en utilisant la commande MarkDuplicateSpark sur les .bam et ce qui redonne un .bam en sortie.

Identification : Pour l'identification, on commence par l'appel des variants en relevant les sites contenant un variant avec l'algorithme HaplotypeCaller de Gatk qui compare le génome de référence avec nos échantillons. Avec l'option -ERC GVCF, on obtient un GCVF par échantillon, obligatoire pour la suite. Ensuite, on fait un appel joint sur nos 26 échantillons en fusionnant les GVCF avec Gatk GenomicsDBImport. L'option L permet de donner les intervalles. On donne aussi la sample.map (créée en aval) qui contient les simple_alias et le chemin d'accès de leur GVCF respectif. On obtient la base de données décrivant les relations entités/attributs pour chaque échantillon et chaque site. On crée ensuite la table résumée avec Gatk GenotypeGVCFs à partir de cette base de données et du génome de référence.

Nous sélectionnons exclusivement les SNPs dans la base de données avec Gatk SelectVariants et l'option –Select-type-to-include SNP. On donne les noms des différents filtres dans un nouveau fichier projetIBI_annotations. Ce dernier est rempli avec les valeurs extraites de table de résumée par la commande bcftools query. Ensuite, on annote les sites du VCF avec les filtres que l'on a déterminés dans une optique de sensibilité appliqué par Gatk VariantFiltration. Puis on retire tous les sites qui ne sont pas filtrés du VCF grâce à vcftools avec l'option --remove-filtered-all.

Analyse : Nous avons 3 Scripts .R pour générer des diagrammes. Filtration.R prend les valeurs dans projetIBI_annotation et dessine une courbe pour chaque type de filtre représentant les valeurs calculées du type en fonction de la densité des sites correspondant à ces valeurs. Il y est donné, pour chaque courbe, les valeurs du filtre et le pourcentage de SNPs qui passe le filtre. Filtration.R dessine aussi un diagramme de Venn pour les sites passant les filtres QD, SOR, MQ, MQRS et RPRS. ACP.R prend la database qui a été filtrée. Il en calcule un ACP et dessine sa représentation graphique. Le dernier script .R est Histogramme.R (écrit par nos soins) qui prend flagstat.csv et ressort l'histogramme de ces valeurs.

On produit aussi des fichiers .txt et .csv. Il y a flagstat.txt et flagstat.csv qui comprennent à peu près la même chose si ce n'est que le .txt possède le minimum et le maximum. Ces fichiers contiennent le pourcentage de données qui mappe, obtenu par la commande samtools flagstat. Le dernier fichier est CouvertureMoyenneGenomeEchantillons.txt qui contient la couverture moyenne (c'est-à-dire le nombre de fois qu'un site est lu lors du séquençage), le minimum et le maximum pour chaque échantillon, ainsi que l'échantillon avec la couverture minimale et la couverture maximale.

RÉSULTATS :

Résultat d'analyse de séquençage :

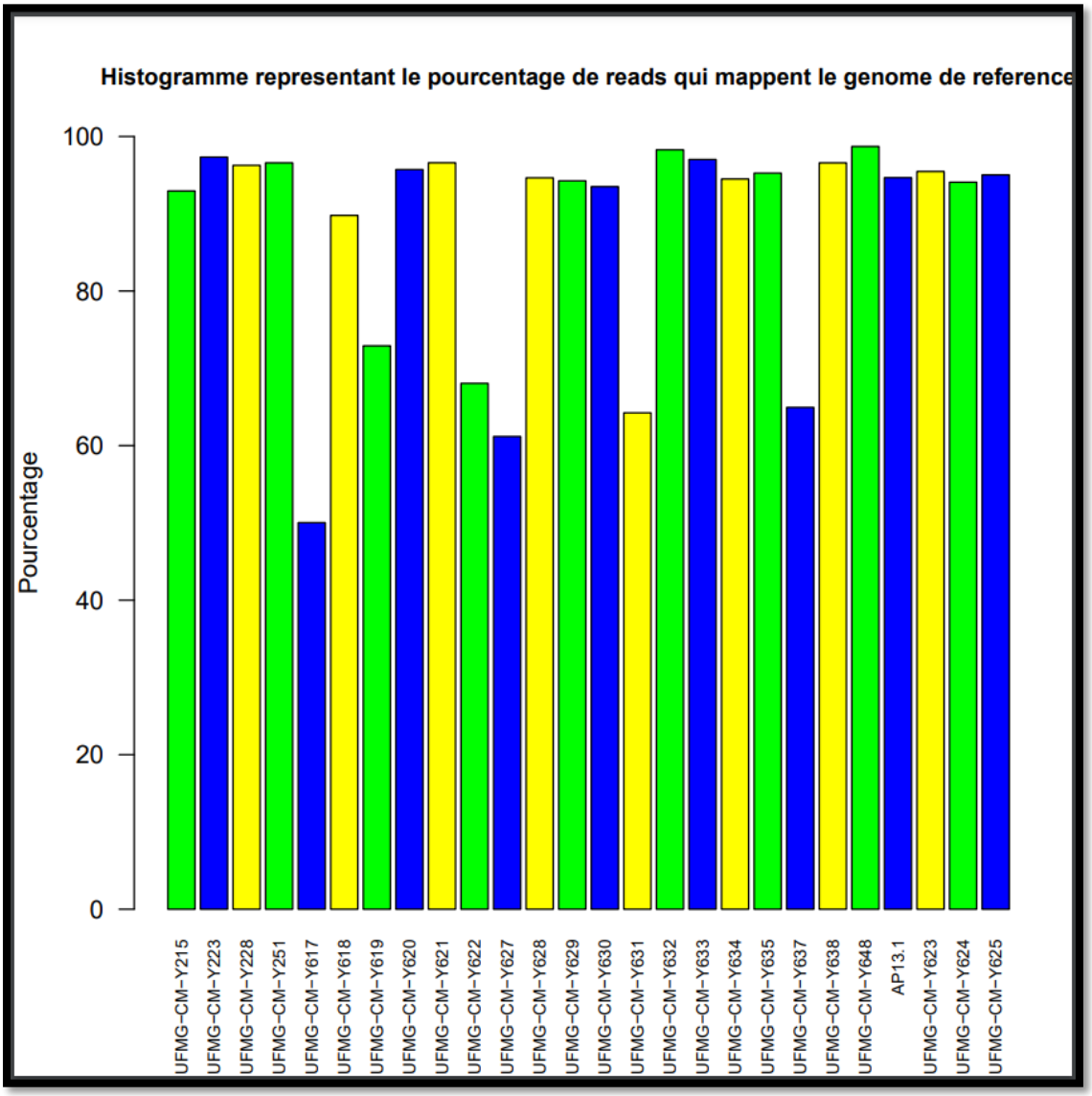


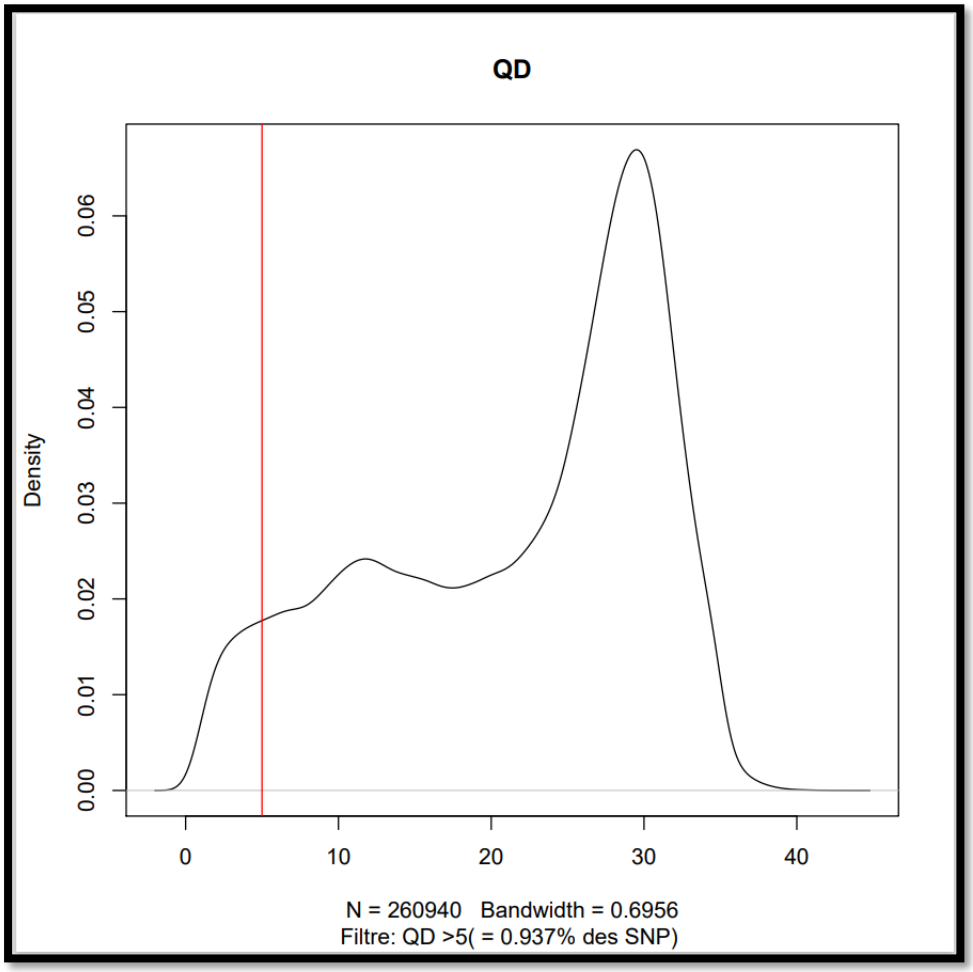
Fig1 : Histogramme représentant le pourcentage de reads qui mappent le génome de référence (obtenu par un script Histogramme.R)

```
Minimum : 50.03
Maximum : 98.71
```

<u>Fig2 :</u> Pourcentage minimum et maximum de reads mappant le génome de référence (obtenu de flagstat.txt)

Sur la Figure 1, dans cet histogramme représentant le pourcentage de reads mappant le génome pour chaque échantillon, on observe que UFMG-CM-Y617, UFMG-CM-Y627, UFMG-CM-Y631 et UFMG-CM-Y637 ont des valeurs inférieures à 65 %. Or, dans la Figure 2 qui contient le minimum et maximum de ces pourcentages, on voit que le minimum est de 50.03% et le maximum de 98.71%. Ces échantillons sont beaucoup plus proches du minimum que du maximum. On peut donc supposer qu'il y a eu une contamination des échantillons pendant le séquençage et qu'ils ne sont alors pas très pertinents à étudier par rapport aux autres.

Résultats de filtration :



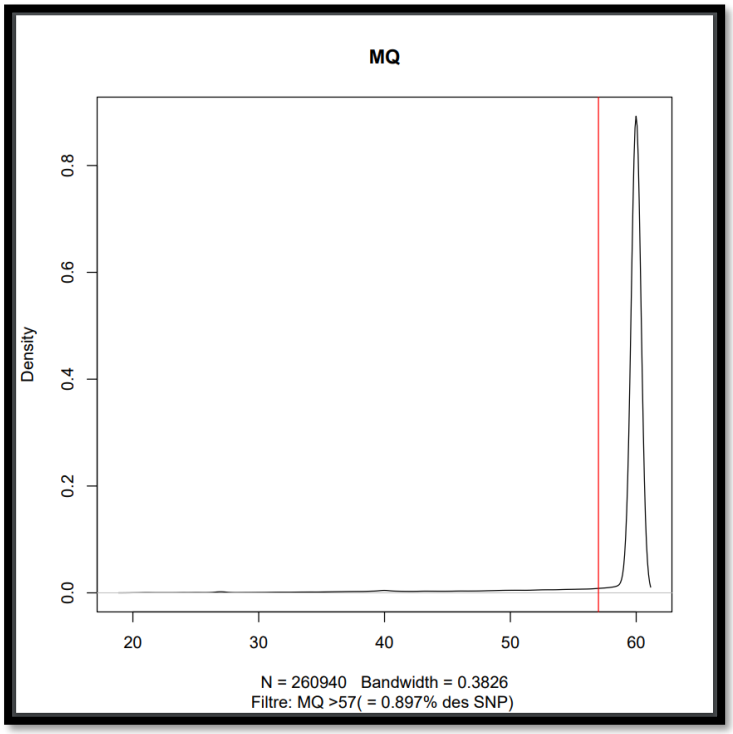
<u>Fig3 :</u> Courbe représentant les valeurs de QD du VCF et la valeur du filtre par la droite rouge (obtenu par le script Filtration.R)

```
UFMG-CM-Y215: 64.43800157193327     minimum: 18.509087305750825 (ref|NC_001224|)    maximum: 101.14474896051391 (ref|NC_001144|)
UFMG-CM-Y223: 116.22344322605228    minimum: 33.69649914314692 (ref|NC_001224|)     maximum: 149.25210146385982 (ref|NC_001144|)
UFMG-CM-Y228: 56.50680826673822     minimum: 49.74448566141657 (ref|NC_001133|)     maximum: 103.63432998477987 (ref|NC_001144|)
UFMG-CM-Y251: 28.68515994477547     minimum: 7.5367980508049754 (ref|NC_001224|)    maximum: 48.915245826983885 (ref|NC_001144|)
UFMG-CM-Y617: 16.558781604515964    minimum: 3.214178295386983 (ref|NC_001224|)     maximum: 25.9586375891899 (ref|NC_001144|)
UFMG-CM-Y618: 45.73010702027801     minimum: 13.746126674360857 (ref|NC_001224|)    maximum: 123.26456045714201 (ref|NC_001144|)
UFMG-CM-Y619: 32.20559836164726     minimum: 12.742990708681612 (ref|NC_001224|)    maximum: 62.55504986658035 (ref|NC_001144|)
UFMG-CM-Y620: 17.562185512469746    minimum: 6.028771610767204 (ref|NC_001224|)     maximum: 24.543933880986145 (ref|NC_001144|)
UFMG-CM-Y621: 20.43341309716812     minimum: 5.424148101516688 (ref|NC_001224|)     maximum: 28.412629837215967 (ref|NC_001144|)
UFMG-CM-Y622: 29.970028841022856    minimum: 9.468716119329907 (ref|NC_001224|)     maximum: 44.04584312223318 (ref|NC_001144|)
UFMG-CM-Y627: 35.36466734752922     minimum: 29.455768011189395 (ref|NC_001133|)    maximum: 50.61273738327563 (ref|NC_001224|)
UFMG-CM-Y628: 32.00345927660548     minimum: 7.447592067988668 (ref|NC_001224|)     maximum: 50.10970554927437 (ref|NC_001144|)
UFMG-CM-Y629: 58.20545382033273     minimum: 0.012322363282388463 (ref|NC_001224|)  maximum: 108.99986273125842 (ref|NC_001144|)
UFMG-CM-Y630: 29.97853752697409     minimum: 8.058662376572354 (ref|NC_001224|)     maximum: 40.142701986779535 (ref|NC_001144|)
UFMG-CM-Y631: 36.625957297406714    minimum: 28.800185042003665 (ref|NC_001133|)    maximum: 56.943386846501085 (ref|NC_001144|)
UFMG-CM-Y632: 53.309759620832665    minimum: 42.3980531496234 (ref|NC_001133|)      maximum: 109.85238811364087 (ref|NC_001224|)
UFMG-CM-Y633: 35.66254035076567     minimum: 8.174798027489246 (ref|NC_001224|)     maximum: 69.00360318990289 (ref|NC_001141|)
UFMG-CM-Y634: 48.18112775458774     minimum: 3.813882185616526 (ref|NC_001224|)     maximum: 106.16509441399695 (ref|NC_001144|)
UFMG-CM-Y635: 59.167159427928915    minimum: 17.83678989029949 (ref|NC_001224|)     maximum: 84.90619649109942 (ref|NC_001139|)
UFMG-CM-Y637: 35.09913023419848     minimum: 29.805162102612528 (ref|NC_001224|)    maximum: 84.20334601832538 (ref|NC_001144|)
UFMG-CM-Y638: 69.32163352286805     minimum: 51.77366255144033 (ref|NC_001224|)     maximum: 120.52273884529164 (ref|NC_001144|)
UFMG-CM-Y648: 31.03853167581627     minimum: 11.316231245409716 (ref|NC_001224|)    maximum: 54.09141819942366 (ref|NC_001144|)
AP13.1: 38.96582886751925           minimum: 31.93861470432373 (ref|NC_001133|)     maximum: 56.87480618799473 (ref|NC_001224|)
UFMG-CM-Y623: 8.233730631081347     minimum: 6.013591465480545 (ref|NC_001133|)     maximum: 39.66499959197473 (ref|NC_001224|)
UFMG-CM-Y624: 8.372738912048774     minimum: 5.0304667749698115 (ref|NC_001133|)    maximum: 49.492859557700605 (ref|NC_001224|)
UFMG-CM-Y625: 6.0436931390222846    minimum: 3.0281574758703225 (ref|NC_001147|)    maximum: 33.91066578066893 (ref|NC_001224|)
Maximum: 116.22344322605228(UFMG-CM-Y223)
Minimum: 6.0436931390222846(UFMG-CM-Y625)
```

<u>Fig4 :</u> Tableau comprenant la couverture moyenne de chaque Échantillon ainsi que son minimum et maximum et l'échantillon avec la couverture minimale et maximale (obtenu de CouvertureMoyenneGenomeEchantillons.txt)

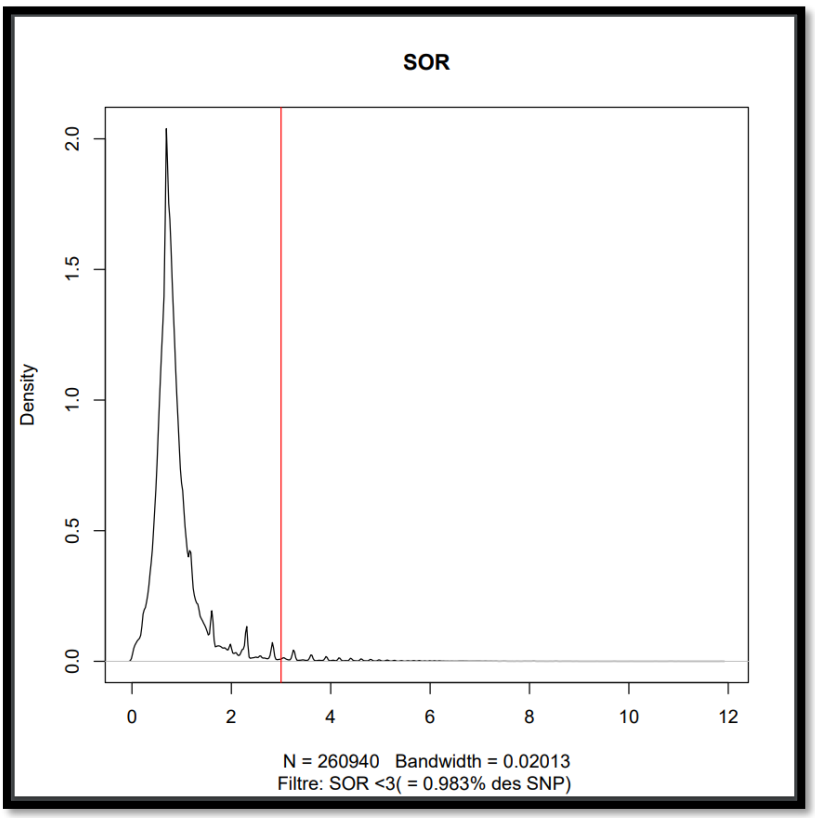
Sur la Figure 4, qui représente la couverture moyenne c'est-à-dire le nombre de fois qu'un site est lu, on peut voir que UFMG-CM-Y623, UFMG-CM-Y624 et UFMG-CM-Y625 possèdent la couverture la plus basse. Or si la couverture est basse, on a une confiance moindre dans leur qualité. Donc ils seront éliminés pour QD.

Sur la Figure 3, qui représente la courbe des valeurs de QD du VCF, on peut observer 2 pics, l'un à environ 10 et l'autre à environ 30. Or si on prend pour valeur supérieur à 10 de QD, on perd les homozygotes. Donc on décide de donner une valeur de 5 au filtre de QD pour éliminer le maximum de site contestable, tout en conservant le mix hétérozygote et homozygotes.



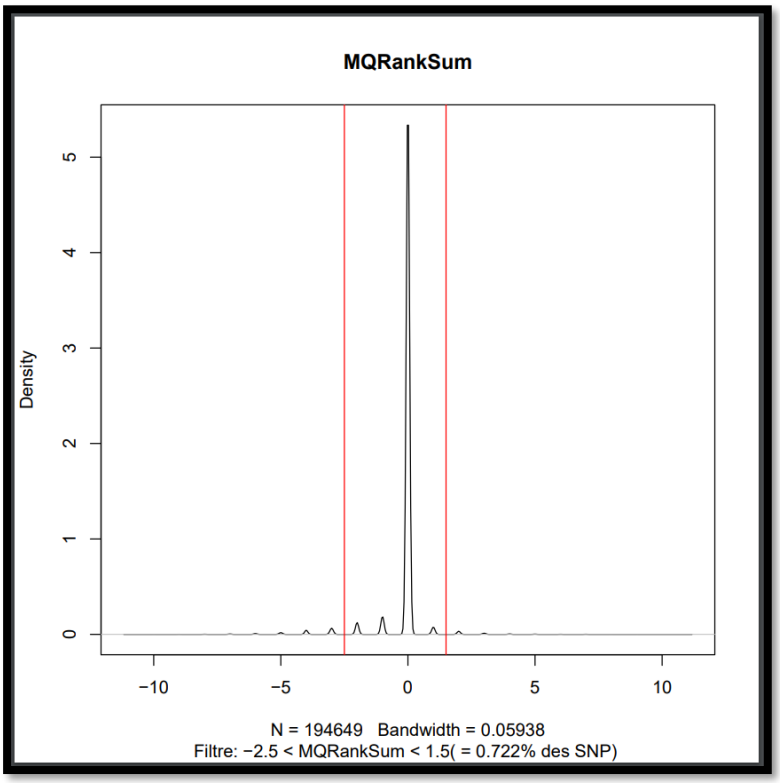
<u>Fig5 : Courbe</u> représentant les valeur de MQ du VCF et la valeur du filtre par la droite rouge (obtenu par le script Filtration.R)

Sur la Figure 5, qui représente la courbe des valeurs de MQ du VCF, on peut voir un pic de densité à environ 60. Or MQ calcule la qualité des reads mappés. On cherche donc la valeur la plus haute, soit ce pic avec un maximum de SNP, d'où la valeur du filtre MQ > 5



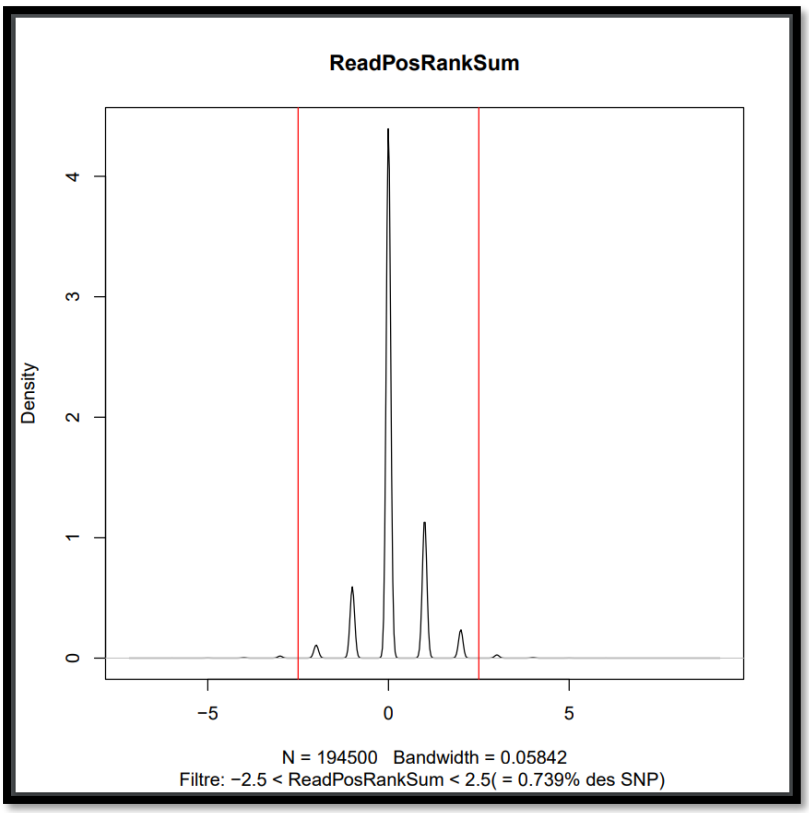
<u>Fig6 : Courbe</u> représentant les valeurs de SOR du VCF et la valeur du filtre par la droite rouge (obtenu par le script Filtration.R)

Sur la Figure 6, qui représente la courbe des valeurs de SOR du VCF, on peut voir un pic de densité à environ 1. Or SOR estime le biais de brins pour un allèle spécifique, on cherche donc la valeur la plus proche de 0 tous en gardant un maximum de SNP, soit ce pic avec un maximum de SNP d'où la valeur du filtre SOR < 3.



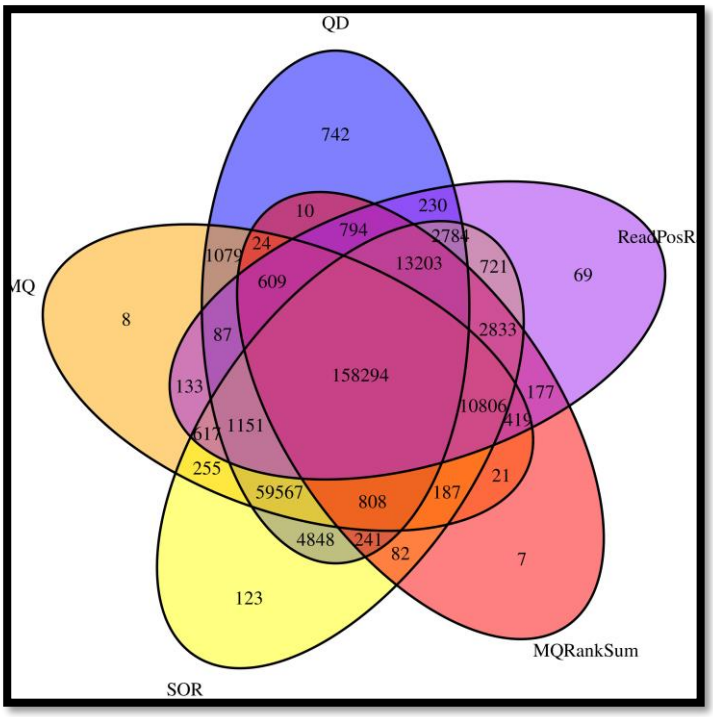
<u>Fig7 : Courbe</u> représentant les valeurs de MQRankSum du VCF et la valeur des filtres par les droite rouges (obtenu par le script Filtration.R)

Sur la Figure 7, qui représente la courbe des valeurs de MQRankSum du VCF, on peut voir un pic de densité à environ 0. Or MQRankSum Compare la qualité du mapping entre les reads avec un allèle alternatif et ceux de référence. On cherche donc la valeur la plus proche de 0 (car il y a peu de différence) tout en gardant un maximum de SNP, soit ce pic avec un maximum de SNP d'où la valeur du filtre -2.5<MQRankSum < 1.5.



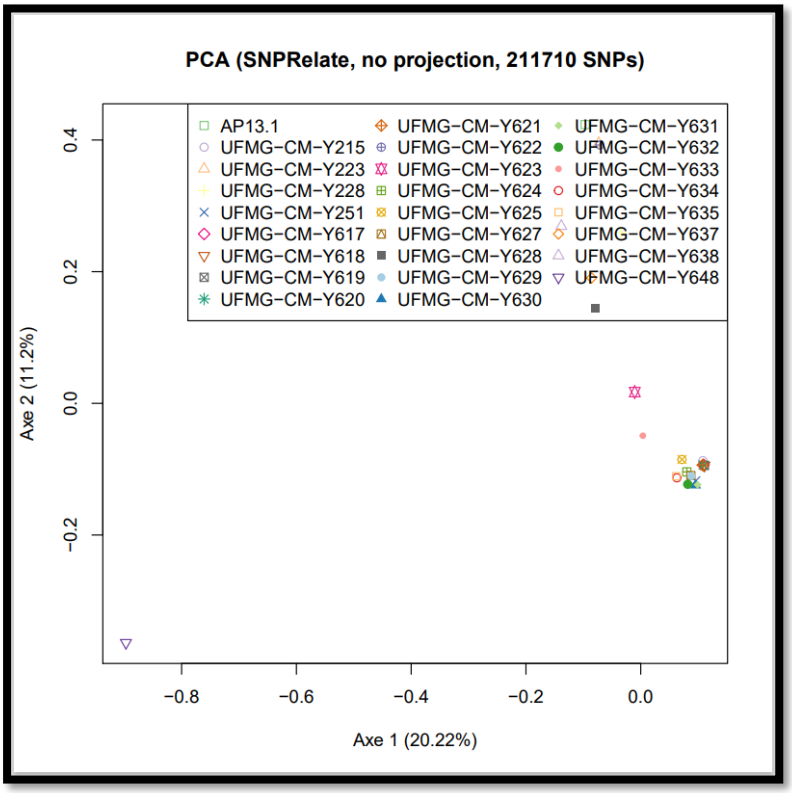
<u>Fig8 : Courbe</u> représentant les valeurs de ReadPosRankSum du VCF et la valeur des filtres par les droites rouges (obtenu par le script Filtration.R)

Sur la Figure 8, qui représente la courbe des valeurs de ReadPosRankSum du VCF, on peut voir un pic de densité à environ 0. Or MQRankSum mesure si l'allèle alternatif a une position médiane dans les reads différente de la référence. On cherche donc la valeur supérieure à 0 (pour que l'allèle alternatif soit vu avec une probabilté equivalente quelque soit le lieu), tout en gardant un maximum de SNP, soit ce pic avec un maximum de SNP d'où la valeur du filtre -2.5<ReadPosRankSum < 2.5.

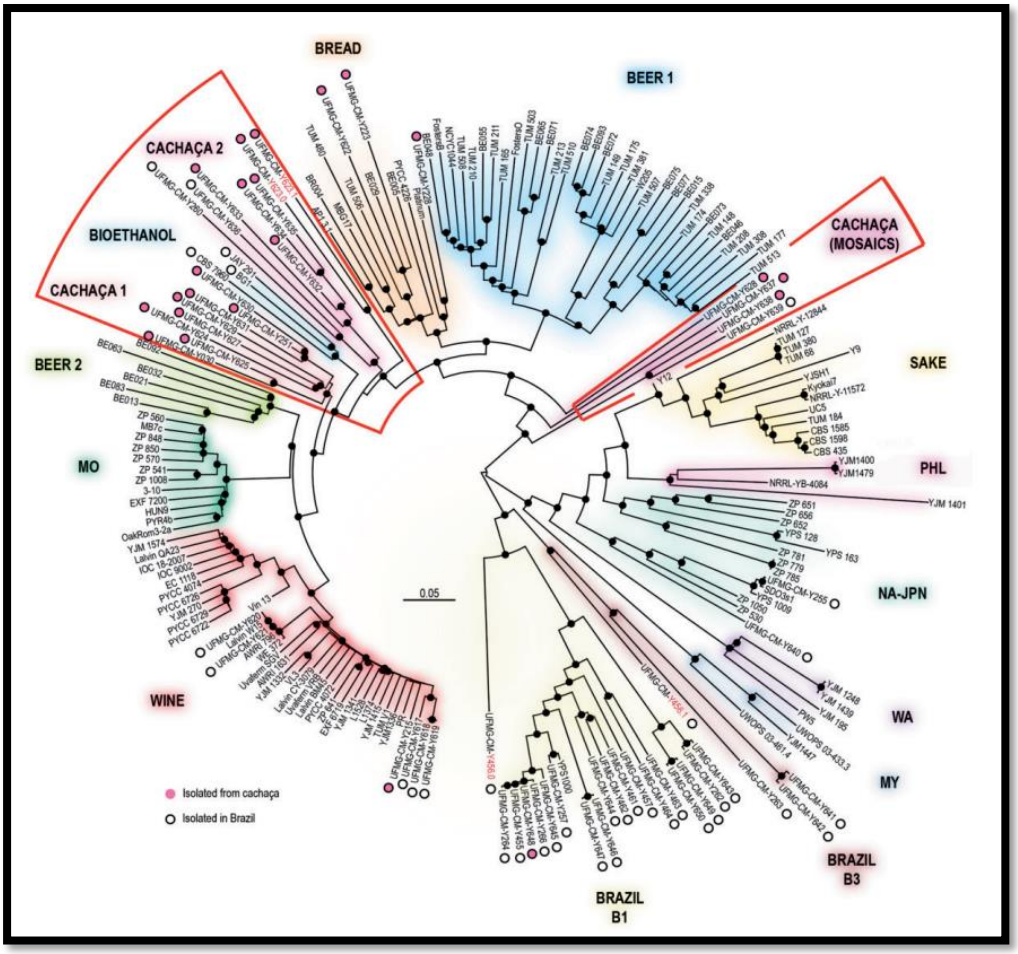


<u>Fig9 :</u> Diagramme de Venn composé des filtres MQ, QD, ReadPosRankSum, MQRankSum et SOR sur le VCF (obtenu par le script Filtration.R)

Sur la Figure 9, qui représente un diagramme de Venn des 5 filtres sur le VCF, on observe qu'il y a beaucoup de SNP qui passent les 5 filtres et peu qui n'en passent qu'un ou deux. On peut donc en déduire que nos filtres nous donnent une approche sensible.



<u>Fig10 :</u> graphique d'un ACP des échantillons (obtenu par le script ACP.R)



<u>Fig11:</u> Arbre phylogénétique du cachaça(obtenu sur Multiple Rounds of Artificial Selection Promote Microbe Secondary Domestication—The Case of Cachac¸a Yeasts)

Sur la Figure 10, qui représente un ACP des échantillons, on observe que UFMG-CM-Y648 est isolé, et que UFMG-CM-Y223, UFMG-CM-Y622 et AP131 sont regroupé. Or sur la Figure 11, qui représente l'arbre phylogénétique, on peut voir que UFMG-CM-Y648 est sur une branche éloignée des autres et que UFMG-CM-Y223, UFMG-CM-Y622 et AP131 sont assez proches sur l'arbre. On en conclut que nos résultats concordent avec ceux de Multiple Rounds of Artificial Selection Promote Microbe Secondary Domestication—The Case of Cachac¸a Yeasts.

<u>Résumé de l'article</u> *<u>Multiple Rounds of Artificial Selection Promote Microbe Secondary Domestication—The Case of Cachac¸a Yeasts :</u>*

Les chercheurs se demandent si les pressions sélectives posées par cette fermentation ont donné naissance à une lignée domestiquée distincte de celles déjà connues, comme les levures de vin, de bière, de pain et de saké. Leurs résultats montrent que les levures de cachaça proviennent de levures de vin qui ont subi un cycle supplémentaire de domestication, qu'ils définissent comme domestication secondaire. En conséquence, les souches de cachaça combinent les caractéristiques des levures de vin, comme la présence de gènes pertinents pour la fermentation du vin et des inactivations géniques avantageuses, avec des caractéristiques des levures de bière comme la résistance aux effets des composés inhibiteurs présents dans la mélasse. Ils proposent un modèle de domestication microbienne multicouche englobant non seulement les transitions des populations sauvages aux populations principalement domestiquées, comme dans le cas des levures de vin, mais aussi les domestications secondaires comme celles des levures de cachaça

CONCLUSION ET PERSPECTIVES

Ce projet nous a permis de développer nos capacités en python et en écriture de script. Mais aussi de nous essayer à la création d'un pipeline, qui malgré ses défauts (nous y reviendrons) nous le trouvons assez honnête pour un premier. Sans compter l'impression de faire une chose qui fonctionne grâce au retour graphique comme les diagrammes ce qui change des programmes habituels beaucoup moins parlant. Pour la repartissions du travail, nous avons toujours travaillé ensemble en distanciel sur l'ordinateur d'Alexis. Pendant que l'un écrivait, les autres l'aidaient ou cherchaient. Dans certains cas rares, on travaillait chacun de notre côté mais c'était dans les cas où le nombre ralentissait plus qu'autre chose.

Comme dit plus tôt, pour le projet, on a dû utiliser des outils spécifiques à la bio-informatique, et ce fut l'une des plus grosses difficultés que l'on ait rencontrées de par le fait d'une documentation soit trop technique, soit trop pauvres, soit trop brouillonne. Sans oublier l'absence presque total de forum et autre aide sur internet pourtant habituellement présentes pour de l'informatique classique. Le problème fut moindre avec R car il reste un langage informatique assez utilisé. L'utilisation de ces outils fut aussi fastidieuse car on ne comprenait pas toujours ce qu'ils faisaient ou, ce que nous, nous devions faire.

Après cette expérience accumulée, on s'est rendu compte que notre script aurait pu être amplement amélioré. Par exemple, pour l'automatisation, on aurait pu mettre le téléchargement et l'installation (dont le renommage des dossiers les comportant) des outils dans le script. Ou encore essayer de déterminer la valeur des filtres automatiquement. On aurait pu aussi essayer de le rendre le plus flexible, possible en vérifiant qu'il était utilisable sur la plupart des OS, donner des options à l'utilisateur pour choisir les actions à faire (la sensibilité des filtres, garder les fichiers déjà existant etc..) et les sorties voulues, ainsi que changer le code pour au cas où les Err dans le TSV ne soit pas donné dans l'ordre alphabétique, ce qui dans notre cas empêcherait l'équivalence avec les bons sample_alias. Le script aurait aussi pu être optimisé à plusieurs endroit comme le fait que markduplicate spark fait aussi le travail de samtools sort, en supprimant les fichiers temporaires ou en ne retéléchargeant pas les fichiers déjà présents.