Introduction à la détection de variants avec un génome de référence

Carina Mugal, Anamaria Necsulea, Laure Ségurel

19 octobre 2023

1 Introduction

L'objectif de cette séance de TP est de détecter des polymorphismes nucléotidiques simples (SNP) chez le gobemouche nain (*Ficedula parva*). Dans le groupe d'espèces du genre *Ficedula*, les mieux étudiées sont le gobemouche à collier (*Ficedula albicollis*) et le gobemouche noir (*Ficedula hypoleuca*), qui sont des modèles pour la recherche en écologie et dans le domaine de la spéciation. Des ressources génétiques et génomiques sont à ce jour disponibles pour ces deux espèces. Pour le gobemouche nain, jusqu'à récemment les analyses génétiques s'étaient restreintes à quelques gènes et aucune étude à l'échelle du génome entier n'avait été effectuée. Pour mieux comprendre les caractéristiques génétiques du gobemouche nain, en 2021 un groupe de chercheur.e.s (Chase et al 2021) a séquencé les génomes de 15 individus de cette espèce. Le génome du gobemouche à collier a été utilisé comme référence pour l'analyse des données.

Au cours de cette séance, nous allons analyser les données brutes de séquençage provenant d'un de ces individus. Nous allons les aligner sur un fragment de chromosome et nous allons faire la détection de variants pour cet individu. Ensuite, nous allons combiner les données obtenues avec celles fournies pour les 14 autres individus. Dans un deuxième temps, après des étapes de filtrage qui vont nous permettre d'améliorer la confiance dans les variants identifiés, nous allons utiliser ces variants pour rechercher des régions qui pourraient être sujettes à de la sélection balancée.

2 Détection de variants

2.1 Téléchargement et inspection des données fournies

1. Connectez-vous à la machine distante sur le cloud IFB, en activant l'utilisation des fenêtres graphiques avec l'option `-X` (`-Y` depuis MacOS) :

   ```
   ssh -X ubuntu@<host_name>
   ```

2. Vérifiez que vous êtes bien à la racine de votre répertoire en utilisant la commande `pwd`.

   ```
   pwd
   ```

3. Téléchargez les données avec la commande `wget`

   ```
   wget http://pbil.univ-lyon1.fr/members/necsulea/FormationNGS_2023/\
   octobre/Jour4/Variant_Calling.tar.gz
   ```

   Dans la commande précédente, vous avez remarqué l'utilisation du signe "\". Ce caractère permet de découper une commande longue sur plusieurs lignes. Le terminal comprend que la commande n'est pas finie et attend la suite. Vous pouvez donc copier toute la ligne de commande et la coller dans le terminal.

4. Désarchivez et décompressez ce fichier :

   ```
   tar -xzvf Variant_Calling.tar.gz
   ```

5. Déplacez-vous dans le nouveau répertoire qui a été créé, qui sera le point de départ pour toutes vos analyses.

```
cd Variant_Calling
```

6. Inspectez le contenu du répertoire avec la commande `tree`. Vous devriez y trouver les fichiers et les dossiers suivants :
   — Un fichier fasta contenant l'assemblage du génome, et l'index (.fai) correspondant :
   ```
   /home/ubuntu/Variant_Calling/assembly/fAlb15_MTmask_mtfZan.fasta
   /home/ubuntu/Variant_Calling/assembly/fAlb15_MTmask_mtfZan.fasta.fai
   ```
   — Des données de reséquençage pour un individu (Sample_31), au format fastq compressé. Ces données sont appariées, nous avons deux fichiers séparés pour les deux lectures.
   ```
   /home/ubuntu/Variant_Calling/fastq/Sample_31_R1.fastq.gz
   /home/ubuntu/Variant_Calling/fastq/Sample_31_R2.fastq.gz
   ```
   — Un fichier BAM contenant les alignements obtenus pour un individu (échantillon 31), et l'index (.bai) correspondant.
   ```
   /home/ubuntu/Variant_Calling/bam/N00208_Sample_31.merged.recal.bam
   /home/ubuntu/Variant_Calling/bam/N00208_Sample_31.merged.recal.bai
   ```
   — Des fichiers GVCF contenant les informations pour les variants détectés pour 14 autres individus, pour la séquence N00208.
   ```
   /home/ubuntu/Variant_Calling/gvcf/
   ```
   — Un fichier contenant la correspondance entre les identifiants des 15 individus et les fichiers GVCF leur correspondant, pour la séquence N00208.
   ```
   /home/ubuntu/Variant_Calling/map_file_15ind_N00208.txt
   ```
   — Un fichier au format BED contenant les positions des éléments répétés.
   ```
   /home/ubuntu/Variant_Calling/fAlb15.fa.rm3.0.out.bed
   ```
   — Un fichier VCF filtré contenant les informations des variants détectés dans les 15 individus pour une autre séquence (identifiant N00135).
   ```
   /home/ubuntu/Variant_Calling/vcf/N00135_15ind_allsites_finalFiltered.vcf.gz
   ```

2.2 Analyse de qualité et alignement des lectures sur le génome de référence

1. Commencez par faire l'analyse de la qualité des données de séquençage pour l'échantillon `Sample_31` en utilisant `fastqc`.
2. Pour aligner les lectures sur le génome de référence, nous allons utiliser le logiciel BWA, dont l'aide se trouve ici :

   `https://bio-bwa.sourceforge.net/bwa.shtml`

   Cet outil n'est pas encore installé sur les machines virtuelles. Installez-le avec la commande `apt install`.

   ```
   sudo apt install bwa
   ```

3. Pour aligner les lectures, nous allons utiliser la commande `bwa mem`. Selon le manuel trouvé sur internet ou dans le terminal, quels sont les paramètres de cette commande ?
4. Réalisez l'alignement avec la commande suivante (attention, elle doit être sur une même ligne de code !)

   ```
   bwa mem assembly/fAlb15_MTmask_mtfZan.fasta fastq/Sample_31_R1.fastq.gz fastq/Sample_31_R2.fastq.gz
         > bam/Sample_31.sam
   ```

5. Triez par position et convertissez l'alignement obtenu au format BAM.

   ```
   samtools sort -m 2G -@ 2 -o bam/Sample_31.bam -O bam bam/Sample_31.sam

   samtools index bam/Sample_31.bam
   ```

6. Regardez les premiers alignements présents dans le fichier `bam/Sample_31.sam` avec la commande `less`. Sur quelle séquence se trouvent-ils ?
7. Visualisez les alignements obtenus avec IGV, sur la séquence d'intérêt que vous avez identifiée précédemment. Est-ce que vous pouvez déjà identifier des variants possibles ?

   Pour la suite, nous allons utiliser un autre alignement pour ce même échantillon, qui a été traité et recalibré avec la suite d'outils PicardTools (https ://broadinstitute.github.io/picard/).

2.3 Détection de variants dans une cohorte d'individus avec GATK

Pour la détection de variants proprement dite, nous allons utiliser la suite d'outils GATK, dont la documentation se trouve ici :

`https://gatk.broadinstitute.org/hc/en-us/categories/360002302312`

Cette suite d'outils n'est pas installée sur les machines virtuelles. Nous allons l'utiliser à travers un conteneur Docker. Cette option nous offre l'avantage de la reproductibilité et de la simplicité : au lieu d'installer la suite GATK et toutes ses dépendances, il suffit d'installer l'outil Docker (c'est déjà fait sur les machines virtuelles que nous utilisons pendant le TP) et de connaître l'adresse de l'image Docker contenant l'outil GATK. Toutes les commandes commenceront donc par un préfixe contenant la commande `docker` et l'adresse du conteneur.

1. Pour demander de l'aide sur GATK, utilisez la commande suivante :

   ```
   docker run broadinstitute/gatk:latest gatk --help
   ```

2. Pour demander de l'aide sur une commande de la suite GATK, par exemple HaplotypeCaller, utilisez la commande suivante :

   ```
   docker run broadinstitute/gatk:latest gatk HaplotypeCaller --help
   ```

3. Lancez la commande HaplotypeCaller pour l'individu numéro 15, sur un seul scaffold (N00208) :

   ```
   docker run -v /home/ubuntu:/home/ubuntu broadinstitute/gatk:latest gatk \
   --java-options "-Xmx16g" HaplotypeCaller \
   -R /home/ubuntu/Variant_Calling/assembly/fAlb15_MTmask_mtfZan.fasta \
   -I /home/ubuntu/Variant_Calling/bam/N00208_Sample_31.merged.recal.bam \
   -O /home/ubuntu/Variant_Calling/gvcf/RB_Sample_31.N00208.g.vcf.gz \
   -ERC GVCF -L N00208
   ```

   Avec l'option `-ERC GVCF`, nous avons demandé de créer des fichiers GCVF. Un fichier GVCF contient de l'information pour tous les sites du génome, y compris pour ceux qui sont homozygotes pour l'allèle de référence. Cela vous semble-t-il important ? A quoi cette information pourrait-elle nous servir plus tard ?

4. La commande précédente vous a permis d'obtenir le fichier `GVCF` pour l'un des 15 individus. Les données des 14 autres vous sont fournies pré-calculées. Nous pouvons maintenant faire la "consolidation", c'est à dire combiner l'information provenant des 15 individus pour obtenir l'ensemble des variants pour la cohorte. Cette commande crée une base de données combinant les variants détectés pour tous les individus.

   ```
   mkdir temp_dir

   docker run -v /home/ubuntu:/home/ubuntu broadinstitute/gatk:latest gatk \
   --java-options "-Xmx16g -Xms12g" GenomicsDBImport \
   --genomicsdb-workspace-path /home/ubuntu/Variant_Calling/N00208_15ind.db_path \
   -L N00208 --tmp-dir /home/ubuntu/Variant_Calling/temp_dir --sample-name-map \
   /home/ubuntu/Variant_Calling/map_file_15ind_N00208.txt
   ```

   Pour demander de l'aide sur l'outil GenomicDBImport, utilisez la commande suivante :

   ```
   docker run broadinstitute/gatk:latest gatk GenomicsDBImport --help
   ```

5. Nous allons maintenant faire le génotypage des 15 individus, en utilisant la commande GenotypeGVCF.

   ```
   docker run -v /home/ubuntu:/home/ubuntu broadinstitute/gatk:latest gatk --java-options "-Xmx12g" \
   GenotypeGVCFs -R /home/ubuntu/Variant_Calling/assembly/fAlb15_MTmask_mtfZan.fasta \
   -V gendb:///home/ubuntu/Variant_Calling/N00208_15ind.db_path -all-sites \
   -L N00208 -O /home/ubuntu/Variant_Calling/vcf/1-N00208_15ind_allsites.vcf
   ```

Pour demander de l'aide sur l'outil GenotypeGVCFs, utilisez la commande suivante :

```
docker run broadinstitute/gatk:latest gatk GenotypeGVCFs --help
```

Analysez l'en-tête du fichier VCF obtenu et comparez-le à l'en-tête du fichier GVCF que vous avez obtenu précédemment. Quelle(s) différences observez-vous ? Regardez également les dernières lignes du fichier VCF et comparez-les à celles du fichier GVCF. Quelle(s) différences observez-vous ?

2.4 Filtrage des fichiers VCF

1. Nous allons commencer par enlèver les insertions/délétions (indels) et par garder uniquement les sites mono-allèliques et bi-allèliques. Pour cela, nous utilserons l'outils `vcftools`, dont la documentation se trouve ici :

   `https://vcftools.github.io/index.html`

```
vcftools --vcf vcf/1-N00208_15ind_allsites.vcf --remove-indels --recode \
--max-alleles 2 --recode-INFO-all --stdout > vcf/2-N00208_15ind_allsites_removeIndels_max2.vcf

bgzip vcf/2-N00208_15ind_allsites_removeIndels_max2.vcf

tabix -p vcf vcf/2-N00208_15ind_allsites_removeIndels_max2.vcf.gz
```

   Nous avons réalisé la procédure au-dessus en plusieurs étapes. D'abord, nous avons utilisé `vcftools` pour créer un nouveau fichier VCF restreint, qui exclut les indels et les sites avec plus de 2 allèles. Nous avons ensuite compressé ce fichier de sortie avec la commande `bgzip` et nous avons créé un index avec la commande `tabix`, pour accélerer le traitement du fichier résultant.

2. Nous allons ensuite appliquer des contrôles de qualité pour enlèver les sites pour lesquels les alignements ne sont pas de qualité suffisante pour détecter des variants. Les critères sur lesquels il est possible de filtrer des variants sont expliqués ici :

   `https://gatk.broadinstitute.org/hc/en-us/articles/360036434492-VariantFiltration`

```
docker run -v /home/ubuntu:/home/ubuntu broadinstitute/gatk:latest gatk --java-options "-Xmx10G" \
VariantFiltration -R /home/ubuntu/Variant_Calling/assembly/fAlb15_MTmask_mtfZan.fasta \
-V /home/ubuntu/Variant_Calling/vcf/2-N00208_15ind_allsites_removeIndels_max2.vcf.gz \
-O /home/ubuntu/Variant_Calling/vcf/2-N00208_15ind_allsites_removeIndels_max2_hardFilter.vcf.gz \
--filter-expression "QD < 2.0 || FS > 60.0 || MQ < 40.0 || MQRankSum < -12.5 \
|| StrandOddsRatio > 3 || ReadPosRankSum < -8.0" --filter-name "hard_filter" -L N00208
```

   Cette commande va créer un nouveau fichier VCF, qui contient une colonne correspondant au filtrage avec des valeurs PASS/FAIL. Nous allons maintenant enlèver du fichier les sites qui n'ont pas passé ces contrôles de qualité, en utilisant la commande `vcftools`.

```
vcftools --gzvcf vcf/2-N00208_15ind_allsites_removeIndels_max2_hardFilter.vcf.gz \
--remove-filtered-all --recode --recode-INFO-all \
--stdout > vcf/3-N00208_15ind_allsites_removeIndels_max2_hardFiltered.vcf

bgzip vcf/3-N00208_15ind_allsites_removeIndels_max2_hardFiltered.vcf
```

3. Une autre étape importante est d'enlèver les variants qui se trouvent dans les éléments répétés, car la présence de ceux-ci pose des problèmes pour les alignements des lectures courtes. Pour ce faire, nous allons utiliser `vcftools` et le fichier BED contenant les coordonnées des régions répétées.

```
vcftools --gzvcf vcf/3-N00208_15ind_allsites_removeIndels_max2_hardFiltered.vcf.gz \
--exclude-bed fAlb15.fa.rm3.0.out.bed --recode --recode-INFO-all --stdout \
 > vcf/4-N00208_15ind_allsites_removeIndels_max2_hardFiltered_repeatFiltered.vcf

bgzip vcf/4-N00208_15ind_allsites_removeIndels_max2_hardFiltered_repeatFiltered.vcf
```

4. Enfin, nous allons faire un filtrage sur la qualité des variants et nous allons enlèver les positions qui ont des couvertures (nombre de lectures qui s'y alignent) extrêmes dans les deux sens : couverture trop faible et trop élevée. A votre avis, pourquoi ce filtre est-il utile ?

```
vcftools --gzvcf vcf/4-N00208_15ind_allsites_removeIndels_max2_hardFiltered_repeatFiltered.vcf.gz \
--minGQ 30 --minDP 5 --maxDP 200 --max-missing 0.9 \
--recode --recode-INFO-all --stdout > vcf/5-N00208_15ind_allsites_finalFiltered.vcf

bgzip vcf/5-N00208_15ind_allsites_finalFiltered.vcf
```

2.5 Visualisation de la couverture dans R

1. Tout d'abord, nous allons calculer la couverture (le nombre de lectures qui s'alignent à chaque position), pour la séquence N00135.

```
vcftools --gzvcf /home/ubuntu/Variant_Calling/vcf/5-N00135_15ind_allsites_finalFiltered.vcf.gz \
--site-mean-depth --out 5-N00135_15ind_allsites_finalFiltered
```

2. Lancez R depuis le dossier `/home/ubuntu/Variant_Calling`. Dans R, pour vérifier que vous êtes dans le bon répertoire, vous pouvez utiliser la commande `getwd`.

```
getwd()
```

3. Chargez les données dans R avec la fonction `read.table` et inspectez les résultats.

```
vcf <- read.table("5-N00135_15ind_allsites_finalFiltered.ldepth.mean", h=T)

head(vcf)
```

4. Tracez le graphique représentant la couverture en fonction de la position sur la séquence et enregistrez le résultat dans un fichier `png`. Pour ce faire, nous allons d'abord faire appel à la fonction `png`, qui initialise le fichier dans lequel l'image sera sauvegardée. Après toutes les commandes qui créent ou modifient le graphique, nous allons utiliser la fonction `dev.off` pour fermer le dispositif graphique.

```
png("coverage_plot_N00135.png")
plot(vcf$POS, vcf$MEAN_DEPTH, cex=0.1, pch= 1, xlab=paste("Position sur N00135"),
       ylab="Couverture", col="black")
dev.off()
```

5. Rapatriez le fichier `png` obtenu sur vos machines avec la fonction `scp` et visualisez-le.

```
scp ubuntu@<host_name>:/home/ubuntu/Variant_Calling/coverage_plot_N00135.png .
```

6. Refaites toute cette procédure (points 1 à 5) pour la séquence N00208. Vous devriez voir l'intérêt de l'organisation de toutes vos commandes dans un script qui permet de revenir facilement en arrière.
7. Inspectez les graphiques obtenus. Que remarquez-vous pour la variation de la couverture le long des deux séquences ? Comment pourrait-on expliquer ces variations ?