Package DANA (Diploid Amplicon NGS Analysis) (Pas mal le nom DANA… ?)

*Traitement données MISEQ Paired-End, locus diploïdes avec potentiels paralogues*

Ce que sont les données : Des données de sortie de miseq (séquences paired-ends) …(Illumina, librairies nextseq).

Le type de run miseq pour lequel on a besoin de réaliser ce package informatique contient, en général, entre 96 et 285 librairies d’amplicons. GENOTOUL (la plate-forme qui l’a fait) nous fournit des fichiers pour chacune des librairies (car ils ont réalisé une étape de démultiplexage, utilisant les séquences des indexs en bordure de séquence, insérés avant le run avant de pooler les librairies, à cette fin). Tout fournisseur donnera les mêmes données brutes.

Info peut-être utile pour non-familiers des NGS : Un run MISEQ typique donne de l’ordre de 10-20 millions de reads au total (parfois moins : nano-run= 1 million). Il faut avoir de nombreuses copies d’une « séquence unique » identiques pour la valider dans un individu car la qualité n’est jamais excellente…elle doit être attestée par la quantité de reads identiques… On utilise souvent des Paired-Ends (un read part de chaque côté de chaque fragment). La longueur de read est choisie au départ –souvent 150, ou 200 ou bien 300bp, mais soucis techniques apparus donc abandon de cette longueur là)

Pour chacune de ces librairies (correspondant en fait à un individu) on avait réalisé l’amplification par PCR à partir de plusieurs paires d’amorces différentes, de plusieurs régions d’ADN, amplification qui donnait des fragments selon le marqueur faisant entre 350 et 800 bp dans le cas présent. Les organismes sont diploïdes et on veut reconstituer les « deux » séquences de chaque individu à chaque locus lorsque c’est possible. Parfois, une paire d’amorces amplifie en fait plusieurs locus (paralogues). On ne le sait pas à l’avance car on travaille sur des espèces non modèles et ces paires d’amorces sont définies à partir de données fragmentaires. Apparemment il y a souvent de tels locus multiples (cf Alexandra Weber, qui avait aussi défini des locus à partir de données de séquences de transcriptomes d’espèces non modèles…).

On part donc pour chaque individu d’un fichier contenant tous les reads (séquences) côté Forward (R1) et un fichier contenant tous les reads côté reverse (R2), pour toutes les paires d’amorces si plusieurs ont été utilisées. (Attention R1 et R2 ne se rejoignent pas forcément, la longueur séquencée est fixe mais la qualité baisse vers la fin donc « l’utilisable » est un peu variable)

Données brutes (Raw data) : contient deux fichiers fastq par puits (un forward R1 et un reverse R2).

Les id des séquences du fichier forward (R1) ressemblent à ça : @MISEQ3:7:xxxx-AFMxxxx1:N:0:GACGAT

Les id des séquences du fichier reverse (R2) ressemble à ça : @MISEQ3:7:xxxx-AFMxxxx2:N:0:GACGAT

**I \_ Etapes préparatoires (filtrer/qualité, recoller, aligner, filtrer/couverture)**

1. Démultiplexage préliminaire à faire (sépare toutes les paires d’amorces)!

- Une première étape consiste à démultiplexer entre « paires d’amorces » : MOTHUR permet de le faire car on lui fournit la liste des amorces. Cela permet donc de séparer par « paire d’amorce » :

Dans MOTHUR cela peut se faire ainsi :

-Demultiplexage : séparation des différents marqueurs. Fonction **split.groups** (apparemment,avant, il faut avoir fait**: fastq.info,** cela crée un fichier fasta et un fichier de qualité à partir du fichier fastq)

On analyse les reads correspondant à une seule paire d’amorces, par la suite.

1. Mapping reads F et R

- choix du Seuil qualité /**phread** (on doit pouvoir choisir, ex: 30 ou mieux, 35 …, calcule sur fenêtre glissante de 10..) **fonction trim.seqs**

-Avant de recoller il faut reverse transcrire le reverse (R2)

- choix du seuil de chevauchement minimal (ex **: m** nucleotides)

Si chevauchement inférieur (c < m), couper d’un côté les c dernières bases (là où la qualité est moins bonne, ou bien, toujours du même côté plutôt : F, ou R à choisir), (ajouter une base notée X après cette troncation serait pratique et pas gênant ?)

-Recoller Forward (R1) et Reverse (R2) (en faisant compl. inv. d’un côté) et en mettant, si ne chevauche pas, au moins un N à chaque bout pour qu'on sache qu'ils ne se chevauchent pas.

-On a donc un fichier fasta par locus par puits (=par individu) contenant toutes les séquences (readF-N-reversecomp(readR)) ayant passé le contrôle qualité.

-On regroupe toutes les séquences pour un même marqueur et une même espèce dans un fichier fasta. Il faudra peut-être modifier les noms des séquences en ajoutant le nom du puits pour ne pas perdre l'identification des individus.

3) Alignement automatique de tous les reads (idée d’utiliser information extérieure disponibles préalablement (par séquences Sanger ou contigs de transcriptomes, donc prévoir si un fichier de séquences alignées référence peut-être fourni ou pas): résultat : un fichier de séquences alignées.

4) Avant d’identifier les « haplotypes différents », il faut se rapporter aux mêmes limites de fragment exactement. Pour bien choisir ces limites, il faut connaitre la distribution des longueurs de reads avant l’éventuelle partie non chevauchante et après, etc… : calculer donc ces statistiques pour pouvoir en déduire une longueur exacte: pas simple diverses situations peuvent survenir ;

Pour les forward (R1) on trace la distribution de la position dans l'alignement de la dernière base avant le N (ce qui serait encore mieux : faire la distribution cumulée en partant de petit vers grand… donnera plus l’idée de tous les reads exploitables potentiels. La figure actuelle n’est pas une distribution cumulée).

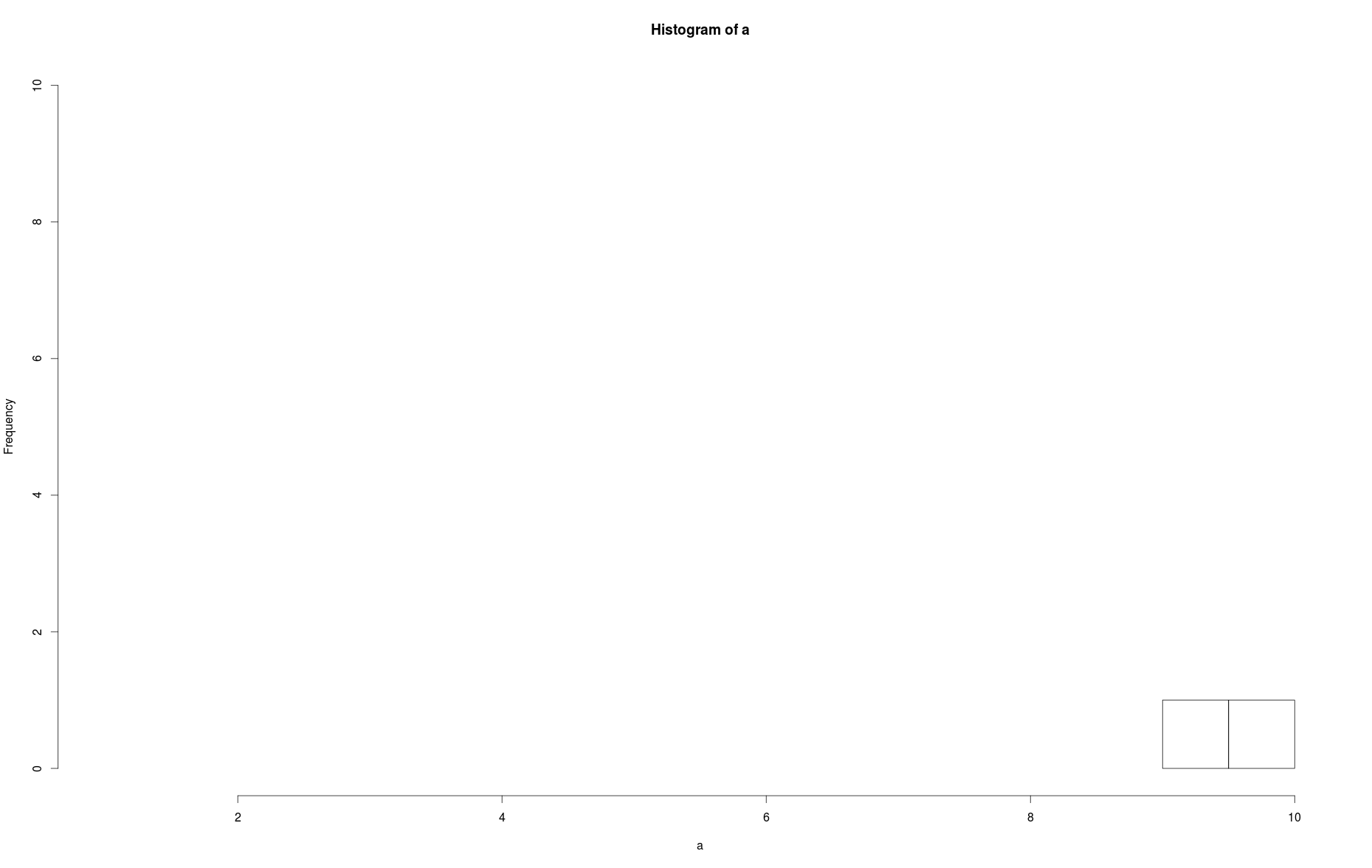
Exemple :

12345678910

ATCGTATCGN

ATCGTATCN

Dans cet alignement à deux séquences on aurait le graph suivant :



Pour les reverse (R2) on fait pareil mais on prend la position juste après le dernier N.

Pour les séquences n’ayant aucun N (celles avec chevauchement du R1 et R2, les mettre dans la classe la plus grande, soit la longueur de l’alignement)

A l’aide de ces distributions, choisir les paramètres de position dans l’alignement à considérer : noter LR1, LR2 les deux limites si on accepte les reads non chevauchants, et prévoir que l’on puisse indiquer si l’on souhaite ne conserver que les reads sans partie manquante (chevauchants).

Ensuite, ne considérer que les séquences à ces limites exactes, éventuellement disjointes. (bien-sûr : ne pas « jeter » les plus longues, mais les tronquer ou ne traiter que la partie exactement identique pour tous les reads recollés)

Il faut avoir une idée de l’effet de ces limites sur le nb de reads qui restent exploitables pour chaque individu... Je propose qu’on puisse choisir un paramètre « **nb read mini/indiv** » par individu, et que le programme donne en retour le nb d’individus n’atteignant pas ce seuil étant donné les limites (**LR1,LR2**). Si ce nbre d’individus nous semble trop important, on peut changer en proposant d’autres valeurs pour certains paramètres (en remontant éventt à la qualité **phred**, ou au chevauchement **m**, **LR1, LR2,** et au **NbRMin**). Pouvoir changer paramètres, tâtonner.

**II- Identification des séquences uniques et construction des majoritary count-tables**

5) Identifier chaque haplotype différent (appelé dans MOTHUR : **Unique.seqs**) pour ces limites précises choisies sur l’alignement (ne pas tenir compte de ce qui est en dehors). Calculer le tableau (count table) du nbre de copies de chacun de ces haplotypes pour chaque individu, à partir des fichiers de reads recollés et alignés… tronqués LR1, LR2.

L’utilisateur est ensuite censé regarder ces count-tables, au moins pour quelques individus pour orienter le choix de paramètre de l’étape suivante. On a souvent une multitude de variants rares probablement des erreurs de séquençage ou PCR… Pour faire le tri du bon et du mauvais, il est utile de privilégier « l’info » qui se retrouve au niveau de tous les individus … Donc :

6) Pour chaque individu, il faut ensuite sélectionner seulement les haplotypes (unique-seqs) majoritaires (car..) : cela permet de regarder tous les haplotypes ensemble au niveau de tous les individus et d’identifier éventuellement des groupes de paralogie.

J’ai pensé à deux ou trois façons d’extraire ces haplotypes (unique-seqs) non rares des différents individus pour les examiner de façon globale… (au choix méthode a, b ou c)

-a) sélectionner les R « unique.seqs » majoritaires (les R les plus représentés de chaque individu). (l’utilisateur doit pouvoir choisir **R**).

-b) sélectionner les unique.seqs les plus fréquents pour chaque individu jusqu’à atteindre au moins la proportion P de tous les reads de l’individu (l’utilisateur doit pouvoir choisir **P**).

-c) sélectionner les unique.seqs présents à plus de X exemplaires dans chaque individu (choisir **X**).

7) Identifier **tous** ces unique.seqs **majoritaires**, et remplir le tableau de leur nbre pour chacun des individus (attention : on doit bien mettre aussi les nbres de reads pour les unique.seqs qui ne sont pas sortis comme majoritaires chez un individu, mais qui le sont chez l’un ou l’autre, puisque dans ce tableau…). Ce fichier pourrait s’appeler la « majoritary-count-table » (il diffère donc de la count-table donnée par MOTHUR qui n’analyse qu’au niveau d’un individu).

NB : ainsi, même avec une limite à R=2 (à l’étape 6), il est donc tout à fait possible d’identifier quand même la présence de deux locus paralogues car cette count table peut ensuite montrer qu’un individu contient 3 ou 4 (par ex !) haplotypes en proportions correctes.

8) Calculer les totaux marginaux (voir autre fichier word pour détails) nb cases non vides par individu, nb individus présentant tel haplotype, etc…

**III\_ Analyse des réseaux d’haplotypes (unique-seqs) et recherche de paralogues/groupes d’allèles**

9) réseaux d’unique.seqs majoritaires, identification de clusters, recherche de clusters pouvant représenter un locus mendélien (les regrouper ou splitter pour tenter…), en jouant sur le tableau (avoir 1 à 2 unique.seqs différents par individu, si plus, ne pas jeter trop vite mais envisager différentes types d’erreurs …cf autre fichier word)

Envoyer les fichiers de données à OC ? et pourquoi pas les scripts déjà écrits. On en parle lundi avec Aurélien. OC a vu que TCS et HaploNet étaient en code source dispo… Il regarde un peu MOTHUR…