Introduction à l'analyse de données de RNA-seq avec génome de référence

Formation CNRS - LBBE
Adil El Filali et Anamaria Necsulea

17 octobre 2023

Table des matières

Préambule

Dans le cadre de cette séance de TP, nous allons effectuer une analyse transcriptomique "classique", en utilisant un génome de référence. Nous allons découvrir plusieurs outils couramment utilisés dans le domaine (HISAT2, kallisto, DESeq2), qui nous permettront d'effectuer toutes les étapes nécessaires pour l'analyse, en partant de l'alignement des données brutes jusqu'à l'identification des gènes différentiellement exprimés entre deux conditions.

La plupart des outils seront appelés en ligne de commande. Les codes qui doivent être exécutés dans le terminal Linux sont donnés dans des encadrés de couleur vert clair dans le document PDF, comme ci-dessous :

```
cd ~
ls
```

Les résultats des commandes exécutées dans le terminal seront donnés (si nécessaire) dans des encadrés vert clair, avec bordure, comme ci-dessous :

```
Bibliothèque calibre examples.desktop Musique Téléchargements
Bureau              igv             Public  Tools
Documents           Images          R       Vidéos
emacs_backups       Modèles         Rlibs   Zotero
```

Lorsque l'on parle des commandes à utiliser dans le terminal directement dans le texte, celles-ci sont formatées avec une police True Type (par exemple : `cd`, `less`).

Nous allons également utiliser le logiciel R pour identifier les gènes différentiellement exprimés. Il est possible de lancer R directement dans le terminal Linux, ou bien utiliser l'interface graphique R Studio. Les commandes R sont exactement les mêmes dans les deux cas. Elles sont données dans des encadrés roses, comme ci-dessous :

```
x=rnorm(1)
x
```

Les résultats des commandes R seront donnés dans des encadrés roses, avec bordure, comme ci-dessous :

```
[1] -0.5533498
```

A la fin de chaque section, vous allez trouver des encadrés gris, contenant des questions supplémentaires pour ceux qui souhaitent aller plus loin dans l'analyse :

Pour aller plus loin : . . .

Il est possible de copier les commandes présentées dans le document PDF et les coller dans le terminal et dans R. Les liens vers des documents externes (pages de documentation des logiciels, téléchargement de données) sont cliquables. **Pour aller plus vite, vous pouvez faire copier-coller pour les commandes qui vous sont données dans l'énoncé.** Il est recommandé de créer un fichier texte (un script), pour noter toutes les lignes de commande utilisées, avec des commentaires.

Enfin, dans le cadre de ce TP nous vous proposons d'utiliser le cloud de l'Institut Français de Bioinformatique (`https://www.france-bioinformatique.fr/fr/cloud`), qui propose des machines configurées spécialement pour l'analyse des données NGS. Vous utiliserez une instance de l'*appliance* Formation LBBE-NGS 2023. Vous pouvez vous connecter à votre machine virtuelle (VM) par `ssh` si vous travaillez sous Linux ou MacOS.

Pour utiliser le mode graphique si vous vous connectez en `ssh` sur la machine virtuelle, utilisez l'option `-X` (`-Y` si vous vous connectez depuis MacOS).

1 Introduction

Au cours de cette séance, nous allons analyser des données tirées d'une publication qui s'est intéressée aux fonctions d'un long ARN non-codant chez la souris [1]. Il s'agit de l'ARN non-codant appelé **Hotair** [2]. Il a été proposé que ce long ARN non-codant serait impliqué dans la régulation des gènes *HOX* chez les vértebrés. Les gènes *HOX*, qui sont organisés chez les vertébrés sous la forme de 4 clusters génomiques (les clusters *HOXA*, *HOXB*, *HOXC* et *HOXD*, contenant chacun une dizaine de gènes protéiques), codent pour des protéines importantes pour le développement embryonnaire. Le gène *Hotair* est localisé au sein du cluster *HOXC*, en antisens par rapport aux gènes qui codent pour des protéines.

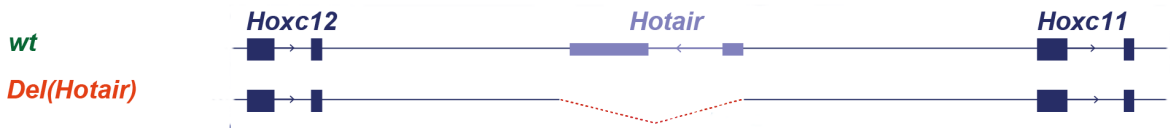

FIGURE 1 – Localisation du gène *Hotair* dans le génome de la souris, sur le chromosome 15. Il est transcrit sur le brin négatif, entre les gènes protéiques *Hoxc11* et *Hoxc12*, qui eux sont codés sur le brin positif. La première ligne du graphique correspond à la configuration présente dans le génome sauvage. La deuxième ligne du graphique représente la région qui a été enlevée lors de la manipulation génétique. Image tirée de [1].

Pour étudier la fonction de Hotair chez la souris, un groupe de chercheurs a généré une souris "knock-out" qui comporte une délétion de ce locus [3] (figure 1). En comparant les transcriptomes des souris sauvages et des souris mutantes, on a observé une augmentation du niveau d'expression des gènes *HOXD*. Cette analyse transcriptomique avait été faite *in vitro*, sur des cultures cellulaires produites à partir de la partie postérieure du tronc de la souris embryonnaire. Pour vérifier si ces conclusions sont valables aussi *in vivo*, un autre groupe de chercheurs a analysé le transcriptome des souris sauvages et des souris mutantes dans des tissus embryonnaires [1]. Plusieurs tissus ont été analysés : les membres supérieurs (FL) et inférieurs (HL), le tubercule génital (GT), ainsi que trois régions du tronc (T1, T2 et T3). Pour chaque tissu, on a généré des données de RNA-seq pour deux individus sauvages (wt) et deux mutants (ko). Les données ont été générées avec le protocole Illumina TruSeq, single-end, avec sélection polyA, brin-spécifique.

Identifiant	Tissu	Génotype	Accession SRA
FL_wt_1	membres supérieurs	sauvage	SRR3213056
FL_wt_2	membres supérieurs	sauvage	SRR3213062
FL_ko_1	membres supérieurs	mutant	SRR3213050
FL_ko_2	membres supérieurs	mutant	SRR3213068
T3_wt_1	tronc postérieur	sauvage	SRR3213061
T3_wt_2	tronc postérieur	sauvage	SRR3213067
T3_ko_1	tronc postérieur	mutant	SRR3213055
T3_ko_2	tronc postérieur	mutant	SRR3213073

TABLE 1 – Données de RNA-seq utilisées au cours de ce TP.

Ces données sont accessibles dans la base de données NCBI SRA avec le numéro d'accession SRP071333 `https://www.ncbi.nlm.nih.gov/sra?term=SRP071333`. Au cours de cette séance, nous allons analyser les échantillons générés pour les membres supérieurs et pour le segment postérieur du tronc (Tableau 1). Nous souhaitons ainsi prédire et analyser les gènes cibles (directes et indirectes) de l'ARN non-codant Hotair, en comparant les transcriptomes des souris sauvages et mutantes.

2 Données fournies

Les données que vous utiliserez au cours de ce TP sont fournies dans l'archive TP_RNASeq_genome.tar.gz. Téléchargez ce fichier dans votre répertoire avec la commande suivante :

```
wget http://pbil.univ-lyon1.fr/members/necsulea/FormationNGS_2023/TP_RNASeq_genome.tar.gz
```

Désarchivez-le et placez-vous dans le répertoire correspondant :

```
tar -xzvf TP_RNASeq_genome.tar.gz

cd TP_RNASeq_genome
```

Les données sont réparties dans plusieurs sous-dossiers. Le dossier RNASeq contient 8 fichiers au format fastq, comme décrit dans le Tableau 1. Ces fichiers ne contiennent qu'un petit sous-échantillon des lectures disponibles, ce qui nous permettra d'effectuer rapidement toute l'analyse au cours de cette séance. Les données sont présentées dans des fichiers au format FASTQ. Ces fichiers de RNA-seq ont été téléchargés depuis la base de données SRA, d'où les identifiants des lectures (par exemple SRR3213056.97467). Pour les visualiser, vous pouvez utiliser la commande head dans le terminal :

```
head -n 4 RNASeq/FL_wt_1.fastq
```

```
@SRR3213056.97467
GGCCATTTCTCCTTCCCCAACACCTCTTTACCAAGCAGCACCTACTATTGCTGCGAATCCTTCTGGCATCTAGACAGAGGACTCAACCTTTCTTCTATGT
+
@@CFFFFFHHHHFIJJJJJJJJJIIIJJGGGHI@FHIIJJIJIGHGGHICEHGGHBGGHIJJJJHGHHGFFFFEF>CCCBBDDA5><A?AAACDDDDEED
```

Pour savoir combien de lectures on a dans chaque fichier de RNA-seq, on peut utiliser la commande wc, qui compte les caractères, mots et lignes :

```
wc -l RNASeq/*.fastq
```

```
   6980 RNASeq/FL_ko_1.fastq
  13648 RNASeq/FL_ko_2.fastq
   9432 RNASeq/FL_wt_1.fastq
   7324 RNASeq/FL_wt_2.fastq
  81680 RNASeq/T3_ko_1.fastq
  96268 RNASeq/T3_ko_2.fastq
 113120 RNASeq/T3_wt_1.fastq
 102856 RNASeq/T3_wt_2.fastq
 431308 total
```

Le dossier genome contient un seul fichier (genome.fa), qui contient une partie du génome de la souris au format fasta. Pour réduire le temps d'exécution des commandes, ici nous allons analyser une petite région du génome de la souris, comprenant environ 2Mb sur deux chromosomes différents (2 et 15).

```
head genome/genome.fa
```

```
>15
GTGGAAAATACTCAAGAAATAGCTAGGAATATATGGGTGTTGCTCTTGATCTCACATGGG
CATCAGGCATGCAGATGTGCACTCTTTCTAGGCTTTACTGACAACAACTTGATGGACCAT
GGAGTCTGGGGAAGAAAATGAGCTGAGGTAGTATCCAGAGACAACAGACATGTACCTGCT
ATCTATGTACAAGACACACTCCAGCATGCATTTGTCACAAAAGAGCTCAGCTAAGAACTT
GTTTCCAAGCCAGTCTTTACTGAATTTGACAATGTCCCTTTATCCAGAAGCACCGTGTGA
GTCTGTGTCAGGGCCTGTAAGCCAGAGCCATATCTCCCCATATTACCTGTGAGAGCTGGA
GTCCAAGATTCCACTATGGACAACAGGGAGTCCCAGGAAACATCCACTTCTGTCTGCCTT
CCCTGTGGACCATCTGGGCTGATATGACTACTGTGTGTCTCTCTCTGCTTTACAGGCCCA
TTTATGAACATTTTTTATGAGCAGGAATATTGATGAGTTTAAATAAGACTCCTCTTTTCA
```

Le dossier annot contient un fichier au format GTF (annot.gtf), extrait de la base de données Ensembl 93 (http://www.ensembl.org). Le format GTF (General Transfer Format) permet de présenter les annotations des exons, transcrits, genes, CDS etc. Chaque ligne correspond à une région génomique, et il y a plusieurs colonnes séparées par des tabulations :

- chromosome
- origine de l'annotation (par exemple la base de données Ensembl)
- type de région annotée (par exemple le gène, exon, CDS etc.)
- coordonnée de début
- coordonnée de fin

— score (une valeur numérique, qui peut représenter la qualité de l'annotation ; les valeurs manquantes sont représentées par un point .)
— brin (+ ou -)
— cadre de lecture (0, 1, 2 pour le CDS, . pour les autres régions génomiques)
— informations supplémentaires (plusieurs informations peuvent être données, séparées par des point-virgule).

```
head -n 2 annot/annot.gtf
```

```
15     ensembl_havana gene   73359  81070  .      +      .      gene_id "ENSMUSG00000001655";
    gene_version "6"; gene_name "Hoxc13"; gene_source "ensembl_havana"; gene_biotype "protein_coding";
15     ensembl_havana transcript    73359  81070  .      +      .      gene_id "ENSMUSG00000001655";
    gene_version "6"; transcript_id "ENSMUST00000001700"; transcript_version "6"; gene_name "Hoxc13";
    gene_source "ensembl_havana"; gene_biotype "protein_coding"; transcript_name "Hoxc13-201";
    transcript_source "ensembl_havana"; transcript_biotype "protein_coding"; tag "CCDS"; ccds_id "
    CCDS27890"; tag "basic"; transcript_support_level "1";
```

Le dossier sequences contient les séquences des cDNAs et des ncRNAs codés par ces fragments de chromosome, au format fasta :

```
head sequences/cDNAs_ncRNAs.fa
```

```
>ENSMUSG00000001655:ENSMUST00000001700
GGGGAGGGAAAAAGAGAGCGAGGTGTCTCCCTAGCTCGCTGCCTCTGGCAAGTGGAGTTT
TTAAAAAGCTGGAGCAGATCATGTCATGACGACTTCGCTGCTCCTGCACCCGCGCTGGCC
GGAGAGCCTTATGTACGTCTATGAGGACAGCGCGGCGGAGAGCGGCAGCGGCGGCGGCGG
GGGAGGCGGCGGCGCGGGCGGCGCGGGGGGTGGCTGCAGCGGAGCGAGCCCCGGCAAAGC
CCCGAGCATGGACGGGCTGGGCGGTAGCTGCCCGGCCAGCCACTGCCGCGACCTGCTCCC
GCACCCTGTATTGGCCCGCCCGCCGGCTCCTCTGGGTGCCCCTCAGGGCGCCGTCTACAC
GGACATCCCAGCCCCGGAGGCGGCGCGCCAATGCGCCCCACCGCCGGCGCCCCCCACCTC
GTCCAGCGCCACCCTGGGCTATGGTTACCCATTTGGGGGCAGCTACTACGGCTGCCGCCT
GTCGCACAACGTGAACCTGCAGCAGAAGCCTTGCGCCTACCATCCGGGGGACAAGTACCC
```

Nous aurons besoin de ces séquences pour l'estimation des niveaux d'expression des gènes.

Si vous souhaitez effectuer l'analyse sur le jeu de données complet (en dehors de la séance de TP), vous pouvez télécharger la séquence entière du génome de la souris sur le site FTP d'Ensembl : `ftp://ftp.ensembl.org/pub/release-93/fasta/mus_musculus/dna/Mus_musculus.GRCm38.dna.primary_assembly.fa.gz`. De même, les annotations complètes sont disponibles à l'adresse suivante : `ftp://ftp.ensembl.org/pub/release-93/gtf/mus_musculus/Mus_musculus.GRCm38.93.gtf.gz`, les séquences des cDNAs à l'adresse `ftp://ftp.ensembl.org/pub/release-93/fasta/mus_musculus/cdna/Mus_musculus.GRCm38.cdna.all.fa.gz`, les séquences des ncRNAs à l'adresse `ftp://ftp.ensembl.org/pub/release-93/fasta/mus_musculus/ncrna/Mus_musculus.GRCm38.ncrna.fa.gz`..

3 Alignement des données de RNA-seq sur le génome

Pour aligner les lectures de RNA-seq sur le génome, nous allons utiliser le logiciel HISAT2 [4], version 2.1.0. Ce logiciel permet l'alignement rapide sur le génome et est très sensible pour la détection des lectures épissées. Le manuel de HISAT2 se trouve à l'adresse suivante : `https://daehwankimlab.github.io/hisat2/manual/`.

3.1 Construction de l'index génomique

Comme la plupart des logiciels utilisés pour l'alignement de lectures de RNA-seq, HISAT2 construit un index pour le génome avant toute étape d'alignement. Cet index est une représentation du génome optimisée pour la recherche d'alignements des lectures de RNA-seq. La principale source d'information pour la construction de l'index est donc la séquence du génome. Pour permettre une recherche efficace pour les alignements épissés, HISAT2 utilise les annotations génomiques pour inclure les introns connus dans l'index. La construction de l'index se fait donc en deux étapes. Premièrement, il faut extraire les coordonnées des exons et des introns (ou sites d'épissage) à partir des annotations au format GTF, dans un format lisible par HISAT2. Cela se fait avec les scripts `hisat2_extract_exons.py` et `hisat2_extract_splice_sites.py`, qui sont fournis avec le programme HISAT2.

```
cd annot

hisat2_extract_exons.py annot.gtf > exons.txt

hisat2_extract_splice_sites.py annot.gtf > introns.txt

cd ..

ls annot/
```

Ouvrez les fichiers ainsi créés avec la commande `head`. Vous remarquerez que ces fichiers ont un format très simple : il s'agit de plusieurs colonnes séparées par des tabulations, contenant les coordonnées des exons et des introns, respectivement.

Ensuite, on peut construire l'index génomique à partir de ces deux fichiers et de la séquence du génome, avec la fonction `hisat2-build`.

```
cd genome

hisat2-build -p 2 --ss ../annot/introns.txt --exon ../annot/exons.txt genome.fa genome

ls

cd ..
```

Nous avons utilisé le programme hisat2-build en précisant plusieurs paramètres :

- `-p` nombre de processeurs (ou threads) utilisés pour le calcul ;
- `--ss` fichier contenant les coordonnées des introns, obtenu précédemment ;
- `--exon` fichier contenant les coordonnées des exons, obtenu précédemment ;
- `genome.fa` fichier contenant la séquence du génome au format fasta ;
- genome préfixe qui sera utilisé pour les fichiers de sortie.

Cette commande a créé 8 fichiers de sortie, qui s'appellent `genome.1.ht2`, `genome.2.ht2`, ...`genome.8.ht2`. Dans ce cas-ci, l'étape d'indexation prend seulement quelques minutes. Attention, lorsque vous créez un index pour un génome de taille importante, comme celui de l'humain ou de la souris, cette étape peut prendre plusieurs heures, selon la puissance de la machine utilisée (nombre de processeurs et mémoire vive disponible).

Vérifiez l'index construit en utilisant la commande `hisat2-inspect` :

```
cd genome

hisat2-inspect -s genome

cd ..
```

Combien y a-t-il de chromosomes dans l'index obtenu ? Combien d'exons et d'introns ? Que remarquez-vous pour les tailles des chromosomes ?

3.2 Alignement des données de RNASeq

Nous allons ensuite aligner les échantillons de RNA-seq fournis, avec le programme principal de HISAT2. Pour cela, nous allons créer le répertoire `res_hisat`, et ensuite un sous-répertoire pour chaque échantillon. Dans un premier temps, nous allons faire cela pour l'échantillon `T3_wt_1`, en utilisant les commandes suivantes :

```
mkdir res_hisat

cd res_hisat

mkdir T3_wt_1

hisat2 -p 2 -x ../genome/genome -U ../RNASeq/T3_wt_1.fastq -S T3_wt_1/aln.sam --rna-strandness R --no-unal

cd ..
```

Nous avons utilisé la commande `hisat2` en précisant plusieurs paramètres :

- `-p` correspond au nombre de processeurs utilisés pour le calcul (dans ce cas-ci, on utilise 2 processeurs) ;
- `-x` chemin d'accès pour l'indéx génomique créé précédemment ;
- `-U` chemin d'accès pour le fichier contenant les données de RNA-seq. Ici on utilise l'option `-U` car les données sont single-end. Pour des données paired-end, on précise les chemins des fichiers contenant les reads 1 et 2 avec les options `-1` et `-2`.

— -S chemin du fichier de sortie, au format SAM (qui sera expliqué plus loin).
— --rna-strandness pour préciser que les données de RNA-seq sont orientées (strand-spécifique). Dans ce cas-ci, les données sont de type "R", ce qui signifie que l'on séquence le brin complémentaire à l'ARNm.
— --no-unal sert à préciser que nous ne souhaitons pas inclure les lectures non-alignées dans le fichier de sortie.

Nous pouvons maintenant faire l'alignement pour les autres échantillons. Pour cela, nous avons deux options : soit nous remplaceons "manuellement" T3_wt_1 par les autres noms d'échantillons dans les commandes au-dessus, soit nous utilisons une boucle for dans le terminal, ce qui permet d'automatiser la procédure. Cette deuxième option est plus pratique :

```
cd res_hisat

for s in FL_wt_1 FL_wt_2 FL_ko_1 FL_ko_2 T3_wt_2 T3_ko_1 T3_ko_2 ## T3_wt_1 deja fait

do

mkdir ${s}

hisat2 -p 2 -x ../genome/genome -U ../RNASeq/${s}.fastq -S ${s}/aln.sam --rna-strandness R --no-unal

done

cd ..
```

Cette étape est extrêmement rapide, d'une part grâce à la performance du logiciel utilisé, mais aussi car nos échantillons sont très réduits - seulement quelques milliers de lectures. Pour des échantillons de RNA-seq complets, contenant plusieurs millions de lectures, on a des temps de calcul de l'ordre de l'heure.

3.3 Visualisation des alignements

Les fichiers au format SAM peuvent être ouverts directements en ligne de commande, par exemple avec la commande head.

```
head res_hisat/FL_wt_1/aln.sam
```

Nous pouvons ainsi déjà voir que les résultats de HISAT2 ne sont pas triés par rapport au chromosome et à la position du début (@HD VN:1.0 SO:unsorted). Pour que les programmes d'analyse et de visualisation des alignements fonctionnent de manière efficace, les alignements doivent être triés. Nous allons faire cela avec le programme samtools. En plus de trier les fichiers, nous allons aussi créer des indexes, ce qui permettra de mieux les analyser et visualiser ensuite.

```
cd res_hisat

for s in FL_wt_1 FL_wt_2 FL_ko_1 FL_ko_2 T3_wt_1 T3_wt_2 T3_ko_1 T3_ko_2

do

samtools sort -m 2G -@ 2 -o ${s}/sorted_aln.bam -O bam ${s}/aln.sam

samtools index ${s}/sorted_aln.bam

done

cd ..
```

Les fichiers de sortie de la procédure de tri sont au format BAM, qui est la variante binaire de SAM. Pour les visualiser, il faut cette fois-ci utiliser le programme samtools.

```
samtools view res_hisat/T3_wt_1/sorted_aln.bam | head
```

La lecture du fichier d'alignement au format SAM ou BAM peut donner beaucoup d'informations, mais ce n'est pas la façon la plus pratique de visualiser des alignements. Nous allons utiliser le logiciel IGV, qui est un navigateur de génome. Il permet de parcourir chaque chromosome, en affichant plusieurs types de données, y compris les annotations génomiques et les sorties de HISAT2.

Pour lancer le navigateur, tapez simplement la commande suivante :

```
igv
```

Nous allons ensuite charger la séquence génomique de référence. Allez dans le menu `Genomes`, `Load genome from file`, et sélectionnez le fichier `TP_RNASeq_genome/genome/genome.fa`.

Ensuite, pour charger les annotations, utilisez le menu `File`, `Load from file`, et sélectionnez le fichier `TP_RNASeq_genome/annot/annot.gtf`. Cela permet de visualiser les gènes présents sur ce chromosome. Il est possible de faire des recherches par mot-clef. Par exemple, il est possible de chercher le gène *Hotair*. Pour charger les alignements obtenus avec HISAT2, utilisez toujours le menu `File`, `Load from file` et chargez les fichiers `sorted_aln.bam` pour les échantillons `T3_wt_1` et `T3_ko_1`. Vous pouvez renommer les données en faisant clic droit sur le nom initialement affiché dans la colonne de gauche, `Rename track`. Que remarquez-vous dans la région correspondant au gène *Hotair* ?

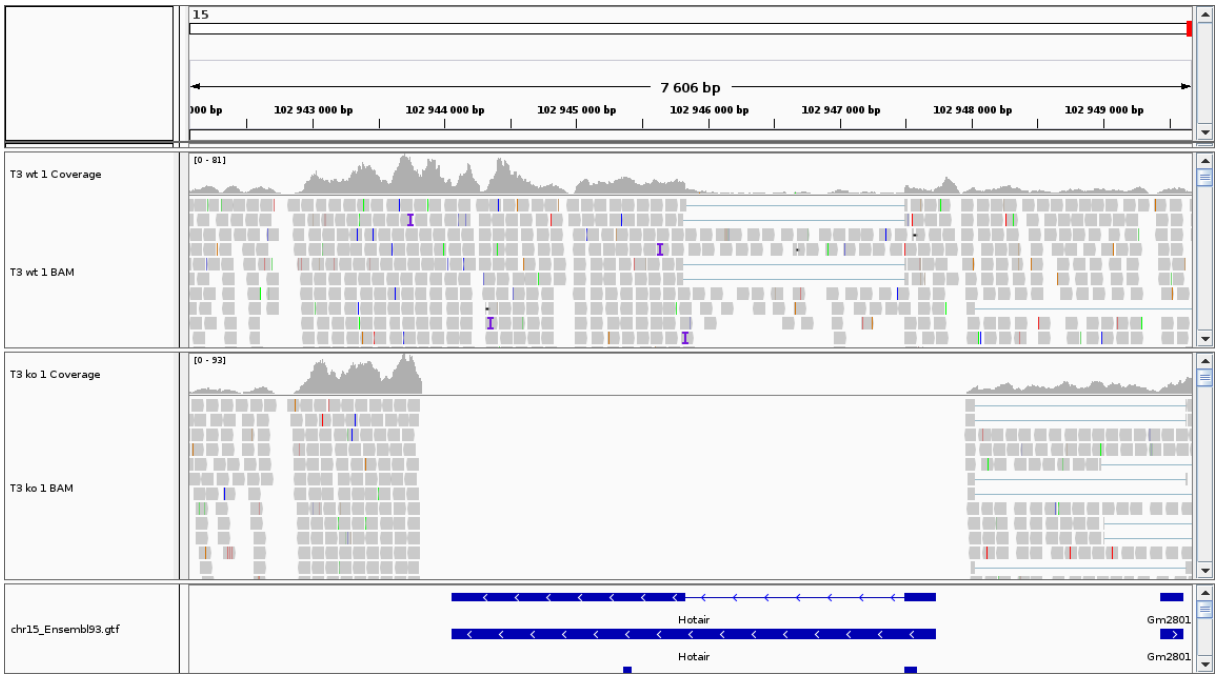

FIGURE 2 – Visualisation avec IGV des résultats de HISAT2 pour `T3_wt_1` et `T3_ko_1`, autour du gène *Hotair*.

4 Quantification des niveaux d'expression des gènes

Les alignements obtenus avec HISAT2 nous permettent également d'estimer les niveaux d'expression des gènes de manière quantitative. Il existe plusieurs façons d'estimer les niveaux d'expression, comme par exemple les mesures RPKM et TPM que nous avons vues en cours. Au cours de cette séance, nous allons comptabiliser tout simplement le nombre de lectures qui s'alignent de manière non-ambiguë sur chaque gène. Ces comptages nous seront utiles pour les analyses d'expression différentielle.

4.1 Comptages des nombres de lectures par gène

Pour calculer le nombre de lectures qui s'alignent sur chaque gène, nous allons utiliser la bibliothèque `Rsubread` dans R. Premièrement, nous devons installer cette bibliothèque, en utilisant les commandes suivantes (répondez "yes" aux deux questions posées) :

```
if (!requireNamespace("BiocManager", quietly = TRUE))
    install.packages("BiocManager")

BiocManager::install("Rsubread")
```

Nous allons utiliser la fonction `featureCounts` de cette bibliothèque. Regardez l'aide de cette fonction avec la commande `?featureCounts`.

```
library(Rsubread)

?featureCounts
```

Nous avons besoin de préciser le fichier bam ou sam contenant les alignements, ainsi que les annotations génomiques. Nous allons calculer les comptages par gène, en excluant les lectures qui chevauchent plusieurs gènes, d'abord pour un seul échantillon :

```
comptages.t3 <- featureCounts("res_hisat/T3_wt_1/sorted_aln.bam", isPairedEnd=F,
  annot.ext="annot/annot.gtf", isGTFAnnotationFile=T, allowMultiOverlap=F, strandSpecific=2)

head(comptages.t3)

head(comptages.t3$counts)
```

Nous avons utilisé `featureCounts` en précisant les paramètres suivants :

- `alnfile` pour indiquer le chemin de l'alignement construit auparavant ;
- `isPairedEnd=F` pour préciser que les données sont single-end ;
- `annot.ext` pour préciser le fichier contenant les annotations du génome ;
- `isGTFAnnotationFile=T` pour indiquer que les annotations sont au format GTF ;
- `allowMultiOverlap=F` pour préciser que nous voulons exclure les lectures qui chevauchent avec plusieurs gènes ;
- `strandSpecific=2` pour préciser le type de librairie de RNA-seq utilisé (TruSeq Illumina).

Ensuite, nous pouvons faire ce calcul pour tous les échantillons, avec une boucle `for` :

```
all.counts <- list()

for(sample in c("FL_wt_1", "FL_wt_2","FL_ko_1","FL_ko_2","T3_wt_1","T3_wt_2","T3_ko_1","T3_ko_2")){

  ## la commande paste permet de coller des chaînes de caractères

  alnfile <- paste("res_hisat/",sample,"/sorted_aln.bam",sep="")

  results <- featureCounts(alnfile, isPairedEnd=F, annot.ext="annot/annot.gtf", isGTFAnnotationFile=T,
  allowMultiOverlap=F, strandSpecific=2)

  colnames(results$counts) <- sample

  all.counts[[sample]] <- results$counts

}
```

```
all.counts <- as.data.frame(all.counts)

head(all.counts)
```

```
                 FL_wt_1 FL_wt_2 FL_ko_1 FL_ko_2 T3_wt_1 T3_wt_2 T3_ko_1
ENSMUSG00000001655   216     299     304     398    4625    3177    2398
ENSMUSG00000050328    98     150     121     135    2646    2541    1601
ENSMUSG00000086903     2       4       0       0     357     289       0
ENSMUSG00000099099     0       1       0       0      10      12       0
ENSMUSG00000098572     0       0       0       0       1       1       0
ENSMUSG00000099328     0       0       2       0       7       4      50
                 T3_ko_2
ENSMUSG00000001655  3700
ENSMUSG00000050328  1502
ENSMUSG00000086903     0
ENSMUSG00000099099     0
ENSMUSG00000098572     0
ENSMUSG00000099328    31
```

Nous pouvons afficher les comptages obtenus pour le gène *Hotair*, en utilisant son identifiant Ensembl :

```
all.counts['ENSMUSG00000086903',]
```

```
                   FL_wt_1 FL_wt_2 FL_ko_1 FL_ko_2 T3_wt_1 T3_wt_2 T3_ko_1
ENSMUSG00000086903     2       4       0       0     357     289       0
                 T3_ko_2
ENSMUSG00000086903     0
```

Nous pouvons déjà nous rendre compte que *Hotair* n'est pas détecté dans les échantillons mutants, ce qui est attendu et rassurant. Nous remarquons également que le niveau d'expression de ce gène paraît plus grand dans la partie postérieure du tronc (T3) que dans les membres (FL). Par ailleurs, il ne semble pas évident que le niveau d'expression de *Hotair* soit significativement plus grand dans les souris sauvages que dans les souris mutantes pour les membres. Pour quantifier ces impressions, nous devons comparer des niveaux d'expression normalisés, ainsi que faire une analyse d'expression différentielle.

4.2 Calcul des niveaux d'expression des gènes

Nous allons maintenant calculer les niveaux d'expression des gènes, exprimés sous la forme TPM (transcripts per million). Pour ce faire, nous allons utiliser `kallisto`, un programme qui permet de quantifier l'abondance des transcrits avec une approche de pseudo-alignement (`https://pachterlab.github.io/kallisto/`).

Nous allons d'abord télécharger la dernière version de `kallisto`, dans l'espace de travail courant sur la machine virtuelle :

```
wget https://github.com/pachterlab/kallisto/releases/download/v0.46.1/kallisto_linux-v0.46.1.tar.gz

tar -xzvf kallisto_linux-v0.46.1.tar.gz

mv kallisto kallisto_0.46.1

mv kallisto_0.46.1/kallisto . ## on déplace l'exécutable dans l'espace de travail courant
```

Ce programme a besoin des séquences des ARNs codants et non-codants. Tout comme HISAT2, `kallisto` construit d'abord un index (ou dictionnaire) des k-mers présents dans les séquences, pour accélérer l'attribution des lectures aux transcrits.

```
./kallisto index -i sequences/cDNAs_ncRNAs sequences/cDNAs_ncRNAs.fa
```

L'index créé se trouve dans le fichier `sequences/cDNAs_ncRNAs`. Pour quantifier l'expression des gènes, nous allons utiliser la commande `kallisto quant`.

D'abord pour l'échantillon `T3_wt_1` :

```
mkdir res_kallisto

mkdir res_kallisto/T3_wt_1

./kallisto quant -i sequences/cDNAs_ncRNAs --single --rf-stranded -l 200 -s 20 \

-o res_kallisto/T3_wt_1 RNASeq/T3_wt_1.fastq
```

Ensuite pour tous les autres échantillons :

```
for s in FL_wt_1 FL_wt_2 FL_ko_1 FL_ko_2 T3_wt_2 T3_ko_1 T3_ko_2 ## T3_wt_1 déjà fait

do

mkdir res_kallisto/${s}

./kallisto quant -i sequences/cDNAs_ncRNAs --single --rf-stranded -l 200 -s 20 \

-o res_kallisto/${s} RNASeq/${s}.fastq

done
```

Nous avons utilisé `kallisto quant` en précisant les paramètres suivants :

- `-i` pour indiquer le chemin de l'index construit auparavant ;
- `--single` pour que les données sont single-end ;
- `--rf-stranded` pour préciser le type de librairie de RNA-seq utilisé (rf-stranded pour des librairies TruSeq Illumina).
- `-l` pour la taille moyenne des fragments d'ARN sélectionnés.
- `-s` pour l'écart type de la taille des fragments sélectionné
- `-o` pour le répertoire de sortie.

Comme vous pouvez le remarquer, `kallisto` a besoin d'une estimation de la taille des fragments de mRNAs sélectionnés, mais ce uniquement lorsque les données sont single-end. Les auteurs de ce logiciel recommandent d'utiliser dans tous les cas du séquençage paired-end, car la taille des fragments peut alors être estimée sans ambiguité à partir des données.

Les fichiers de sortie qui nous intéressent sont ceux nommés `abundance.tsv`. Regardez le format de ce fichiers avec la commande `head`. Nous allons également chercher les valeurs d'expression de *Hotair* dans ces fichiers, avec la commande `grep` :

```
grep ENSMUSG00000086903 res_kallisto/*/abundance.tsv
```

Vous devriez voir que les comptages estimés (colonnes `est_counts`) sont différents de ceux estimés avec `featureCounts`. Les valeurs de TPM (dernière colonne des fichiers) sont aussi surprenamment élevées. Il y a une explication très simple : ici on utilise kallisto sur un jeu de données "jouet", qui contient uniquement quelques milliers de lectures et très peu de gènes. Les estimations de `kallisto` sont fiables uniquement lorsque l'on utilise les jeux de données complets. La valeur donnée aux paramètres `l` et `s` a aussi un impact non-négligeable, il faut l'estimer au mieux lorsque l'on travaille avec des données single-end.

5 Analyse de l'expression différentielle

Pour cette partie-là des analyses, nous allons utiliser le tableau de comptages de lectures par gène extrait de la publication originale, comme précédemment. Pour faire une analyse d'expression différentielle, nous ne pouvons pas utiliser les comptages que nous avons obtenus nous-mêmes, car ceux-ci concernent très peu de gènes. Nous allons utiliser R et la librairie DESeq2 [5] pour l'analyse de l'expression différentielle.

5.1 Préparation des données

Pour cette analyse, en plus du tableau contenant les nombres de lectures, nous avons besoin de créer des variables factorielles qui associent à chaque échantillon le tissu et le génotype correspondant. Ces informations sont disponibles dans le fichier `comptages_publication/columnData.txt`. Les données doivent être présentées sous la forme d'un tableau dont les colonnes représentent les variables factorielles (tissu et génotype dans ce cas), et dont les noms des lignes correspondent aux noms des échantillons. Attention, les lignes doivent apparaître dans le même ordre que les colonnes du tableau contenant les comptages !

Chargez le tableau contenant les comptages des lectures pour tous les gènes avec la commande suivante :

```
data <- read.table('comptages_publication/UniqueReadCounts.txt', h=T, stringsAsFactors=F)

head(data)
```

```
                                GeneID GeneName ExonicLength FL_wt_1
ENSMUSG00000000001_Gnai3 ENSMUSG00000000001 Gnai3      3262   8715
ENSMUSG00000000003_Pbsn ENSMUSG00000000003 Pbsn         902      0
ENSMUSG00000000028_Cdc45 ENSMUSG00000000028 Cdc45      2252   1862
ENSMUSG00000000031_H19 ENSMUSG00000000031   H19        2372 210565
ENSMUSG00000000037_Scml2 ENSMUSG00000000037 Scml2      6039    224
ENSMUSG00000000049_Apoh ENSMUSG00000000049 Apoh        1594      0
                     FL_wt_2 FL_ko_1 FL_ko_2 T3_wt_1 T3_wt_2 T3_ko_1
ENSMUSG00000000001_Gnai3 9466    7800   10837    9605    8742   10144
ENSMUSG00000000003_Pbsn     0       0       0       0       0       0
ENSMUSG00000000028_Cdc45 1826    1929    2453    1717    1481    1738
ENSMUSG00000000031_H19 178310 152329 223100 129665 125728  85720
ENSMUSG00000000037_Scml2  263     195     265     237     196     164
ENSMUSG00000000049_Apoh     1       0       0       0       0       0
                     T3_ko_2
ENSMUSG00000000001_Gnai3 11636
ENSMUSG00000000003_Pbsn     0
ENSMUSG00000000028_Cdc45 2120
ENSMUSG00000000031_H19  99315
ENSMUSG00000000037_Scml2  160
ENSMUSG00000000049_Apoh     0
```

```
reads <- data[,4:11] ## on garde uniquement les comptages

coldata <- read.table('comptages_publication/columnData.txt', h=T, stringsAsFactors=T)

coldata
```

```
      tissue genotype
FL_wt_1    FL       wt
FL_wt_2    FL       wt
FL_ko_1    FL       ko
FL_ko_2    FL       ko
T3_wt_1    T3       wt
T3_wt_2    T3       wt
T3_ko_1    T3       ko
T3_ko_2    T3       ko
```

```
all(colnames(reads)==rownames(coldata)) # on vérifie que l'ordre est bien le même
```

```
[1] TRUE
```

Nous devons ensuite préparer le jeu de données pour faire l'analyse différentielle. Au cours de cette séance, nous allons faire deux analyses séparées, la première pour les membres (FL), et la deuxième pour le tronc (T3). Pour cela, nous allons séparer les comptages par tissu, en utilisant la commande `which` dans R.

```
reads.fl <- reads[ ,c('FL_wt_1', 'FL_wt_2', 'FL_ko_1', 'FL_ko_2')] # on extrait uniquement les colonnes FL

reads.t3 <- reads[ ,c('T3_wt_1', 'T3_wt_2', 'T3_ko_1', 'T3_ko_2')] # on extrait uniquement les colonnes T3

coldata.fl <- coldata[c('FL_wt_1', 'FL_wt_2', 'FL_ko_1', 'FL_ko_2'), ]

coldata.t3 <- coldata[c('T3_wt_1', 'T3_wt_2', 'T3_ko_1', 'T3_ko_2'), ]
```

5.2 Analyse à un facteur

Nous pouvons ensuite charger la librairie DESeq2, et créer des objets DESeq2 pour chaque tissu :

```
library(DESeq2)

dds.fl <- DESeqDataSetFromMatrix(countData = reads.fl, colData=coldata.fl, design=~ genotype)

dds.t3 <- DESeqDataSetFromMatrix(countData = reads.t3, colData=coldata.t3, design=~ genotype)
```

Nous allons ensuite réaliser l'analyse différentielle proprement-dite. Ici, nous avons choisi d'utiliser le test de Wald. Dans DESeq2, il est aussi possible d'utiliser le test LRT (Likelihood Ratio Test), qui est notamment utile pour analyser des modèles complexes avec plusieurs variables explicatives.

```
test.fl <- DESeq(dds.fl, test='Wald')

results.fl <- results(test.fl)

test.t3 <- DESeq(dds.t3, test='Wald')

results.t3 <- results(test.t3)
```

Les résultats sont donnés dans le même ordre que le tableau de départ. Nous pouvons les trier plutôt par la significativité du test, c'est à dire par le taux de fausses découvertes (ou p-valeur ajustée) :

```
results.fl <- results.fl[order(results.fl$padj),]

results.t3 <- results.t3[order(results.t3$padj),]
```

Nous pouvons donc afficher les gènes différentiellement exprimés entre souris sauvages et mutantes, par ordre de significativité :

```
head(results.fl)
```

```
log2 fold change (MLE): genotype wt vs ko
Wald test p-value: genotype wt vs ko
DataFrame with 6 rows and 6 columns
                        baseMean log2FoldChange   lfcSE      stat
                       <numeric>     <numeric> <numeric> <numeric>
ENSMUSG00000027597_Ahcy  510.1309    -2.4342189 0.18724241 -13.000361
ENSMUSG00000059040_Eno1b 1499.7414     1.2695116 0.13868639 9.153829
ENSMUSG00000047686_Zcchc5 994.2958     1.1865070 0.14417717 8.229507
ENSMUSG00000027570_Col9a3 3777.8756    0.8849682 0.11551834 7.660846
ENSMUSG00000020908_Myh3  4885.4317     0.6196101 0.08295618 7.469126
ENSMUSG00000027966_Col11a1 5412.5184   0.7119670 0.09507667 7.488347
                            pvalue        padj
                           <numeric>   <numeric>
ENSMUSG00000027597_Ahcy 1.217677e-38 1.650074e-34
ENSMUSG00000059040_Eno1b 5.495036e-20 3.723162e-16
ENSMUSG00000047686_Zcchc5 1.879860e-16 8.491327e-13
ENSMUSG00000027570_Col9a3 1.847120e-14 6.257581e-11
ENSMUSG00000020908_Myh3  8.072915e-14 1.823268e-10
ENSMUSG00000027966_Col11a1 6.974666e-14 1.823268e-10
```

```
head(results.t3)
```

```
log2 fold change (MLE): genotype wt vs ko
Wald test p-value: genotype wt vs ko
DataFrame with 6 rows and 6 columns
                             baseMean   log2FoldChange           lfcSE
                            <numeric>       <numeric>       <numeric>
ENSMUSG00000059751_Rps3a3 7423.5619916396 -3.20105394233614 0.147858989127859
ENSMUSG00000034674_Tdg  304.609881662406 -3.90335392759817 0.266964131953343
ENSMUSG00000038872_Zfhx3 2550.78433818141 1.53193033416745 0.123727380148036
ENSMUSG00000030761_Myo7a 334.591430880191 -2.61815484768379 0.231019841743774
ENSMUSG00000082420_Gm6517 445.777835386689 -2.47903070419432 0.219217111003779
ENSMUSG00000021268_Meg3 661.732005664433 1.91538574566527 0.170454398170653
                                  stat              pvalue
                                 <numeric>         <numeric>
ENSMUSG00000059751_Rps3a3 -21.649369857169 6.16169587548295e-104
ENSMUSG00000034674_Tdg  -14.6212672805062 2.05538533396497e-48
ENSMUSG00000038872_Zfhx3 12.3814981965555 3.29140603577255e-35
ENSMUSG00000030761_Myo7a -11.3330302190563 9.00349758163333e-30
ENSMUSG00000082420_Gm6517 -11.3085638837362 1.1902164129188e-29
ENSMUSG00000021268_Meg3  11.2369394173546 2.68531350263941e-29
                                      padj
                                    <numeric>
ENSMUSG00000059751_Rps3a3 9.26472591837616e-100
ENSMUSG00000034674_Tdg  1.54523869407486e-44
ENSMUSG00000038872_Zfhx3 1.6496527051292e-31
ENSMUSG00000030761_Myo7a 3.38441474093597e-26
ENSMUSG00000082420_Gm6517 3.57921879692942e-26
ENSMUSG00000021268_Meg3  6.72939563761437e-26
```

Les tableau de résultats contient plusieurs colonnes :

- `baseMean` : le nombre moyen de lectures par gènes, normalisé par DESeq2 ;
- `log2FoldChange` : l'intensité du changement du niveau d'expression du gène, fold-change en anglais. Attention, ce paramètre est toujours présenté avec une transformation logarithmique : un `log2FoldChange` négatif représente une diminution du niveau d'expression entre les deux conditions.
- `lfcSE` : l'erreur standard du `log2FoldChange` ;
- `stat` : la statistique du test. Dans ce cas, il s'agit de la statistique du test de Wald.
- `pvalue` : la p-valeur du test.
- `padj` : le taux de fausses découvertes (FDR), aussi appelé p-valeur corrigée ou ajustée. Dans DESe2, des valeurs `NA` pour `padj` indiquent la présence d'un potentiel point abérrant, qui montre beaucoup de variabilité entre réplicats.

Les premières lignes des tableaux de résultats correspondent aux gènes qui sont très fortement différentiellement exprimés entre conditions ; nous pouvons remarquer des p-valeurs et des FDR très basses, inférieures à 10^{-10}. En plus de la p-valeur du test, nous devons également évaluer l'intensité du changement d'expression (fold-change). Pour visualiser ces deux variables, nous pouvons tracer un graphique de type "volcan" (volcano-plot), qui affiche la FDR (avec une transformation logarithmique) en fonction du fold-change :

```
par(mfrow=c(1,2))

plot(results.fl$log2FoldChange, -log10(results.fl$padj), xlab='log2FoldChange', ylab='-log10 FDR',
  main='FL', pch=20, xlim=c(-5, 5), ylim=c(0, 50))

plot(results.t3$log2FoldChange, -log10(results.t3$padj), xlab='log2FoldChange', ylab='-log10 FDR',
  main='T3', pch=20, xlim=c(-5, 5), ylim=c(0, 50))
```

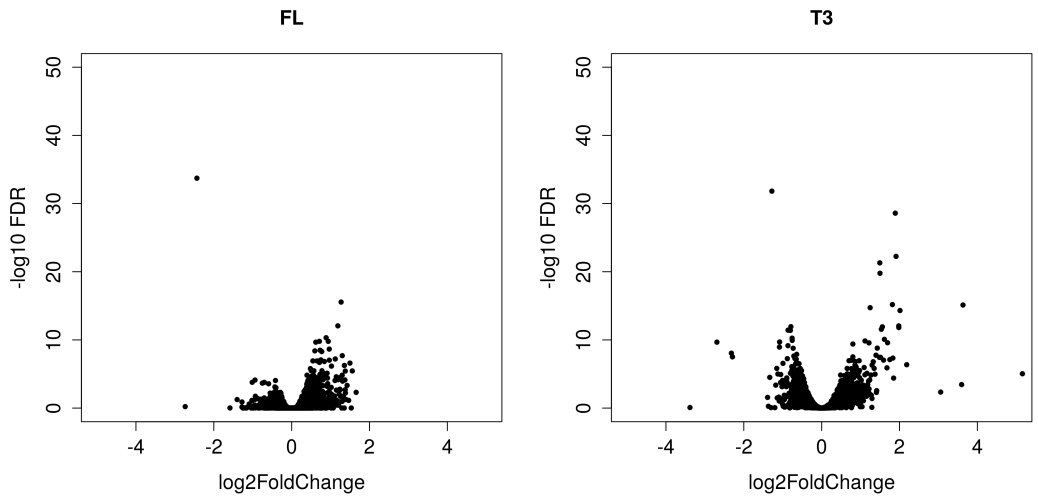

FIGURE 3 – Graphique du taux de fausses découvertes (FDR) en fonction de l'intensité du changement d'expression (fold-change), pour les deux tissus. Chaque point représente un gène. Tous les gènes sont affichés, y compris ceux pour lesquels le test n'est pas significatif.

Nous pouvons également demander combien de gènes sont significativement différentiellement exprimés pour chaque test, avec un seuil de FDR de 0.05 :

```
length(which(results.fl$padj<0.05))
```

```
[1] 264
```

```
length(which(results.t3$padj<0.05))
```

```
[1] 2574
```

Pour continuer à explorer les données, nous pouvons tracer l'intensité du changement d'expression ("fold-change") en fonction du nombre moyen de lectures par gène. Ce graphique peut se faire facilement dans DESeq2 avec la fonction `plotMA`.

```
plotMA(results.fl, main='FL')

plotMA(results.t3, main='T3')
```

Ce graphique (figure 4) illustre la dépendance entre le niveau d'expression moyen du gène (*i.e.*, le nombre moyen de lectures) et l'intensité du changement (apparent) d'expression entre conditions. Pour les gènes faiblement exprimés, on peut observer de forts changements de niveau d'expression entre conditions, car la variance est plus forte lorsque les comptages sont faibles. Ces changements ne sont donc pas considérés comme significatifs par les méthodes implémentées dans DESeq2, qui tiennent compte de cette sur-dispersion du nombre de lectures.

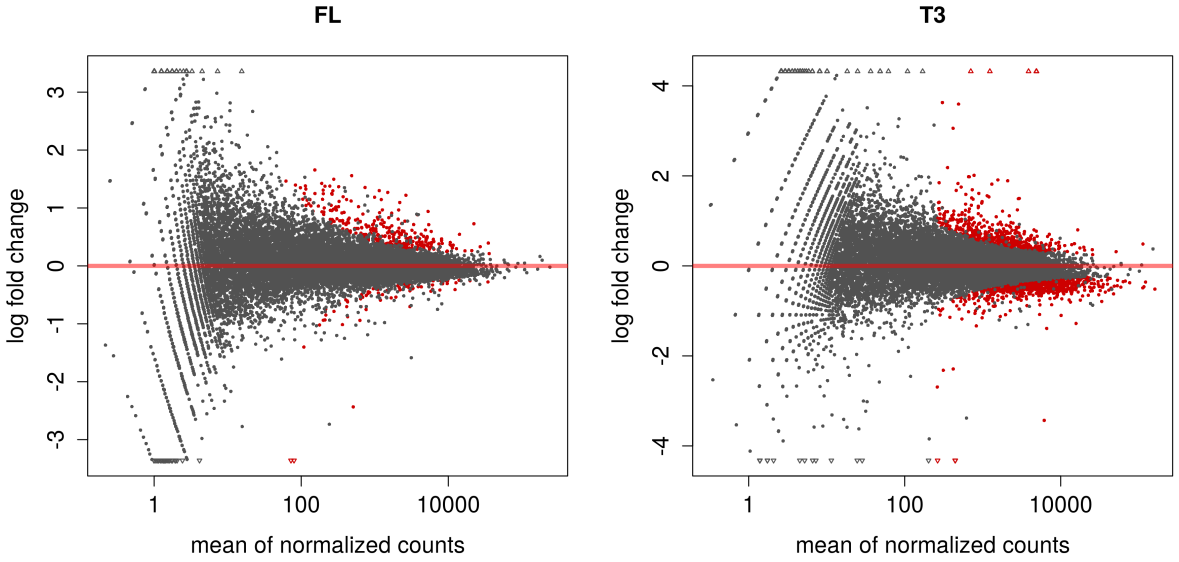

FIGURE 4 – Analyse de l'expression différentielle avec DESeq2. Les graphiques montrent l'intensité du changement d'expression ("fold-change") en fonction du nombre moyen de lectures par gène, pour FL (à gauche) et pour T3 (à droite). Les gènes pour lesquels le test est significatif, avec un taux de fausses découvertes <0.1, sont affichés en rouge.

A-t-on des changements d'expression significatifs pour les gènes du cluster *HOXD* en l'absence de *Hotair* ?

```
results.t3[grep("Hoxd",rownames(results.t3)),]
```

```
log2 fold change (MLE): genotype wt vs ko
Wald test p-value: genotype wt vs ko
DataFrame with 10 rows and 6 columns
                               baseMean    log2FoldChange
                              <numeric>         <numeric>
ENSMUSG00000079277_Hoxd3 386.126797261641 0.578518354715588
ENSMUSG00000101174_Hoxd4 269.770987218125 0.577932275266609
ENSMUSG00000042448_Hoxd1 307.903336058892 -0.498978534418955
ENSMUSG00000050368_Hoxd10 1431.86506418939 0.241508581195282
ENSMUSG00000001819_Hoxd13 1041.27761442183 -0.266538039384888
ENSMUSG00000027102_Hoxd8 557.371833183616 0.257233273487138
ENSMUSG00000001823_Hoxd12 1833.01593022728 -0.188696570986973
ENSMUSG00000042499_Hoxd11 758.475112405366 -0.181082320068269
ENSMUSG00000043342_Hoxd9 654.287214188934 -0.104457146110652
ENSMUSG00000052371_Hoxd3os1 92.1644483926131 0.0311990846635391
                                 lfcSE             stat
                              <numeric>         <numeric>
ENSMUSG00000079277_Hoxd3 0.216437851146615 2.67290749585061
ENSMUSG00000101174_Hoxd4 0.226539056438739 2.55113746985564
ENSMUSG00000042448_Hoxd1 0.226870299986759 -2.19939998513722
ENSMUSG00000050368_Hoxd10 0.14012415778313 1.72353279417432
ENSMUSG00000001819_Hoxd13 0.179969416354463 -1.48101852405812
ENSMUSG00000027102_Hoxd8 0.178525840227562 1.44087417910623
```

```
ENSMUSG00000001823_Hoxd12 0.153232677532729 -1.23143818945975
ENSMUSG00000042499_Hoxd11 0.199518410716537 -0.907597045395169
ENSMUSG00000043342_Hoxd9 0.222489538977613 -0.469492393173429
ENSMUSG00000052371_Hoxd3os1 0.366803332991399 0.0850567098425755
                                  pvalue             padj
                                <numeric>        <numeric>
ENSMUSG00000079277_Hoxd3  0.007519698304809 0.0452264734844433
ENSMUSG00000101174_Hoxd4 0.0107371965049559 0.0587926025668304
ENSMUSG00000042448_Hoxd1 0.0278494936999945 0.114923995568578
ENSMUSG00000050368_Hoxd10 0.0847922200293569 0.242532411750641
ENSMUSG00000001819_Hoxd13 0.138601637344587 0.333229008492678
ENSMUSG00000027102_Hoxd8  0.149620232010008 0.349113874690018
ENSMUSG00000001823_Hoxd12 0.218159021095297 0.440655432722848
ENSMUSG00000042499_Hoxd11 0.364091155914993 0.596738022709597
ENSMUSG00000043342_Hoxd9  0.638717720841238 0.810033708718696
ENSMUSG00000052371_Hoxd3os1 0.932216306074767 0.970491198375697
```

Pour aller plus loin :

- ⋆ Combien trouve-t-on de gènes différentiellement exprimés en commun entre les deux tissus ?
- ⋆ Refaites le test d'expression différentielle en utilisant le test LRT au lieu du test de Wald, pour chaque tissu. Combien a-t-on de gènes différentiellement exprimés avec ce test ? Que remarquez-vous ?

5.3 Analyse d'enrichissement en catégories fonctionnelles

Notre analyse de l'expression différentielle nous a permis d'identifier plusieurs centaines de gènes qui changent significativement leurs niveaux d'expression entre souris sauvages et mutantes, aussi bien pour les membres que pour le tronc. Pour donner une première interprétation biologique à ces listes des gènes, nous pouvons effectuer une analyse d'enrichissement en catégories fonctionnelles, en utilisant les annotations Gene Ontology. Nous allons utiliser le serveur web GORILLA [6], qui est disponible à l'adresse suivante : `http://cbl-gorilla.cs.technion.ac.il/`

Lorsque nous faisons une analyse d'enrichissement en catégories fonctionnelles GO, nous souhaitons tester si certaines catégories GO sont sur-représentés dans notre liste de gènes d'intérêt, par rapport à l'ensemble des gènes analysés. Pour chaque catégorie GO, le serveur GORILLA va mesurer les valeurs suivantes :

— n = le nombre de gènes d'intérêt ;
— N = le nombre total de gènes ;
— b = le nombre de gènes associés avec une certaine catégorie GO dans le jeu de gènes d'intérêt ;
— B = le nombre de gènes associés avec une certaine catégorie GO dans l'ensemble des gènes analysés.

Pour tester s'il y a un enrichissement, il faut comparer les proportions $\frac{b}{n}$ et $\frac{B}{N}$, ce qui se fait typiquement avec un test hypergéométrique. Pour faire cette analyse, nous avons besoin d'extraire la liste des gènes différentiellement exprimés, ainsi que la liste complète des gènes qui sont exprimés dans ce tissu, qui vont nous servir de référence pour l'analyse d'enrichissement. Nous allons sélectionner tous les gènes différentiellement exprimés entre souris mutantes et sauvages, avec un seuil de FDR de 0.05.

```
de.t3 <- rownames(results.t3)[which(results.t3$padj<0.05)] ## on extrait les identifiants des gènes

head(de.t3)
```

```
[1] "ENSMUSG00000059751_Rps3a3" "ENSMUSG00000034674_Tdg"
[3] "ENSMUSG00000038872_Zfhx3" "ENSMUSG00000030761_Myo7a"
[5] "ENSMUSG00000082420_Gm6517" "ENSMUSG00000021268_Meg3"
```

Le serveur GORILLA accepte comme entrée soit les identifiants Ensembl (par exemple ENSMUSG00000059751), soit les noms usuels des gènes (par exemple Rps3a3). Nous allons récupérer les identifiants Ensembl à partir du tableau `data`, que nous avons chargé au tout début de notre session R, et les écrire dans un fichier de sortie avec la fonction `writeLines`.

```
id.t3 <- data[de.t3, 'GeneID'] ## on récupère les identifiants Ensembl

writeLines(id.t3, con='DiffExpT3.txt') ## on les écrit dans un fichier txt
```

Nous allons faire de même pour tous les gènes exprimés (c'est à dire, qui ont un comptage moyen de lectures supérieur à 0) dans le tissu T3 :

```
allexp.t3 <- rownames(results.t3)[which(results.t3$baseMean>0)] ## tous les gènes exprimés

allid.t3 <- data[allexp.t3, 'GeneID'] ## on récupère les identifiants Ensembl pour ces gènes

writeLines(allid.t3, con='AllExpT3.txt') ## on les écrit dans un fichier txt
```

Il faut maintenant rapatrier ces données sur vos machines locales, pour pouvoir les télécharger sur le serveur GORILLA.

```
## a executer en local !

scp login@adresse_IP:TP_RNASeq_genome/DiffExpT3.txt .

scp login@adresse_IP:TP_RNASeq_genome/AllExpT3.txt .
```

Nous pouvons maintenant télécharger ces données sur GORILLA et lancer l'analyse. Choisissez bien l'espèce *Mus musculus* et le mode `Two unranked lists of genes`, et téléchargez le fichier `DiffExpT3.txt` dans le champ `Target set` et le fichier `AllExpT3.txt` dans le champ `Background set`. Choisissez d'analyser toutes les ontologies (processus biologiques, fonction moléculaire et composant cellulaire), et demandez une sortie au format Excel.

Dans ce cas précis, nous observons que de très nombreuses catégories fonctionnelles sont enrichies parmi les gènes différentiellement exprimés entre souris sauvages et mutantes. La plupart des catégories GO significatives (taux de fausses découvertes ou q-values < 0.05) sont liées au métabolisme cellulaire. Attention, les catégories GO ne sont pas indépendantes ! Elles sont imbriquées les unes dans les autres, par exemple 'response to stress' est une sous-catégorie de 'response to stimulus', etc. Le serveur GORILLA permet de visualiser la hiérarchie des catégories GO, avec un code couleur lié au résultat du test d'enrichissement (figure 5).

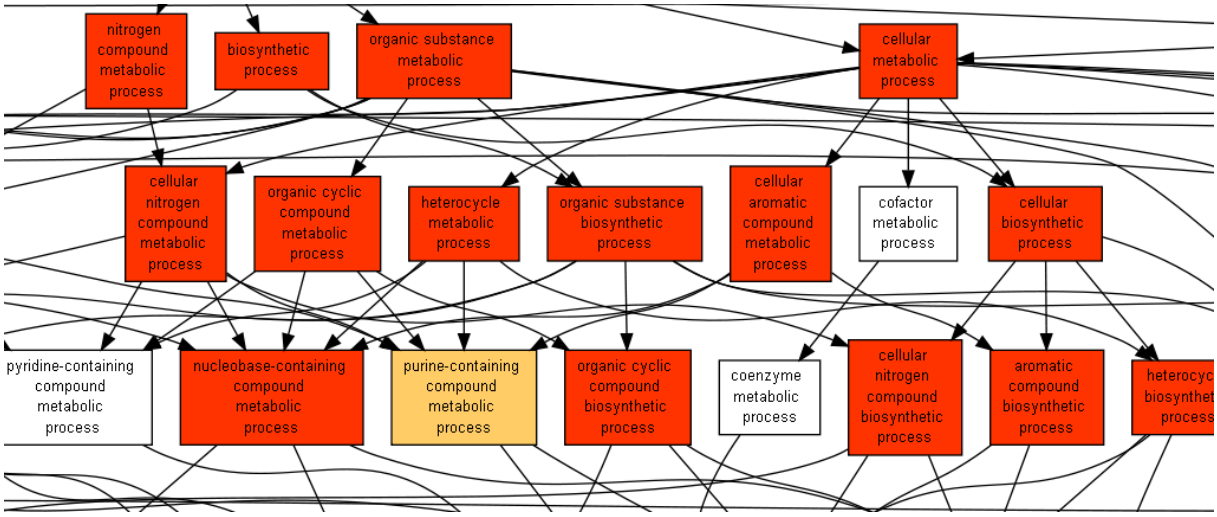

FIGURE 5 – Capture d'écran du serveur web GORILLA, pour l'enrichissement en catégories fonctionnelles GO pour les gènes différentiellement exprimés dans le tronc T3.

Pour aller plus loin :

- ⋆ Que se passe-t-il si l'on fait l'analyse d'enrichissement GO en séparant les gènes sur-exprimés et les gènes sous-exprimés dans les souris mutantes par rappport aux souris sauvages ?
- ⋆ Et si l'on utilise l'ensemble des gènes annotés dans le génome (et non pas seulement les gènes exprimés dans ce tissu) comme jeu de données de référence ?
- ⋆ Refaites cette analyse sur les gènes différentiellement exprimés dans les membres (FL).

5.4 Analyse à deux facteurs

Pour l'instant, nous avons fait l'analyse de l'expression différentielle séparément pour chaque tissu, pour simplifier. Il n'est pas nécessaire de faire cette simplification, car DESeq2 peut analyser des modèles complexes, à plusieurs facteurs. Les principes sont similaires à ceux que l'on applique pour une régression linéaire ou pour une ANOVA.

Dans R, nous allons d'abord créer un objet DESeq qui cette fois-ci prend en compte les deux facteurs (tissu et génotype)

```
dds.2F <- DESeqDataSetFromMatrix(countData = reads, colData=coldata, design=~ tissue+genotype)
```

Nous avons défini ici un modèle additif, dans lequel on cherche un effet commun du génotype pour les deux tissus, sans intéraction. On peut donc faire le test d'expression différentielle :

```
test.2F <- DESeq(dds.2F, test='Wald')
```

Nous pouvons ensuite récupérer les résultats pour l'effet du génotype :

```
res.genotype <- results(test.2F, contrast=c("genotype", "wt", "ko"))
```

Comme avant, nous pouvons visualiser les résultats après les avoir triés par FDR :

```
res.genotype <- res.genotype[order(res.genotype$padj),]
head(res.genotype)
```

```
log2 fold change (MLE): genotype wt vs ko
Wald test p-value: genotype wt vs ko
DataFrame with 6 rows and 6 columns
                             baseMean   log2FoldChange           lfcSE
                            <numeric>       <numeric>        <numeric>
ENSMUSG00000027570_Col9a3 2301.93837746578 0.995981567916788 0.09632214299974
ENSMUSG00000024529_Lox  994.796549847142 0.890540836267011 0.0905187427657554
ENSMUSG00000024899_Papss2 1166.41640909052 1.11285329328406 0.119850782858872
ENSMUSG00000016995_Matn4 1585.28529722844 1.2373619928662 0.135031624156191
ENSMUSG00000032735_Ablim3 1227.51380574994 0.790403429304518 0.0882565122570393
ENSMUSG00000074971_Fibin 2111.54856872063 0.544414105391183 0.066943800105493
                                stat             pvalue
                            <numeric>          <numeric>
ENSMUSG00000027570_Col9a3 10.340110143931 4.64003918216634e-25
ENSMUSG00000024529_Lox  9.83819272182729 7.70830466375081e-23
ENSMUSG00000024899_Papss2 9.28532352262131 1.61216649059855e-20
ENSMUSG00000016995_Matn4 9.16349781466702 5.02418233274843e-20
ENSMUSG00000032735_Ablim3 8.95575192233451 3.37427954264586e-19
ENSMUSG00000074971_Fibin 8.13240515974999 4.20855437234877e-16
                                    padj
                                <numeric>
ENSMUSG00000027570_Col9a3 7.37070224087124e-21
ENSMUSG00000024529_Lox  6.12232097918408e-19
ENSMUSG00000024899_Papss2 8.53642156771935e-17
ENSMUSG00000016995_Matn4 1.99522840889272e-16
ENSMUSG00000032735_Ablim3 1.07200861069859e-15
ENSMUSG00000074971_Fibin 1.11421477007934e-12
```

Nous voyons que les gènes qui ont les FDR les plus fortes sont différents de ceux que l'on avait observés pour les analyses séparées pour FL et T3. C'est normal, car cette DESeq2 a cette fois-ci estimé la variance en utilisant tous les échantillons pris ensemble, et non pas les tissus séparés. Comme la significativité des résultats dépend de l'estimation de la variance, on ne s'attend pas à avoir le même nombre de résultats.

6 Amélioration des annotations génomiques

Nous avons vu lors de la première partie du TP que les annotations génomiques étaient imparfaites, même pour une espèce modèle comme la souris. Dans cette situation, il est possible d'améliorer les annotations en utilisant les données de RNA-seq. Pour ce faire, nous allons utiliser StringTie, un outil dont l'aide se trouve ici :

`https://ccb.jhu.edu/software/stringtie/`

Cet outil n'est pas encore installé sur les VMs, mais peut l'être très facilement avec la commande apt.

```
sudo apt install stringtie

stringtie --help
```

Quelle est la valeur par défaut utilisée par StringTie pour la longueur minimale des transcrits identifiés ? Et pour la fréquence minimale des isoformes ?

StringTie prend en entrée un alignement de données de RNA-seq sur le génome de référence. Si une annotation génomique est disponible, elle peut être fournie comme entrée à StringTie.

Pour améliorer la sensibilité de l'annotation, nous allons d'abord combiner les alignements des différents échantillons, en utilisant la commande `samtools`.

```
cd res_hisat

mkdir merged

samtools merge --threads 2 -o merged/sorted_aln.bam *_*/sorted_aln.bam

samtools index merged/sorted_aln.bam

cd ..
```

Nous allons maintenant faire tourner StringTie sur cet alignement combiné, avec ou sans annotation de référence.

```
mkdir res_stringtie

stringtie res_hisat/merged/sorted_aln.bam -o res_stringtie/assembled_transcripts_withref.gtf \
-G annot/annot.gtf --rf

stringtie res_hisat/merged/sorted_aln.bam -o res_stringtie/assembled_transcripts_noref.gtf \
--rf
```

Nous avons laissé quasiment toutes les options par défaut pour StringTie, à part le type de librairie de séquençage, indiqué avec l'option `-rf` pour les librairies RNA-seq.

Nous pouvons maintenant visualiser ces annotations avec IGV. Ouvrez ce navigateur, comme précédemment. Chargez les annotations de référence, celles obtenues avec StringTie, ainsi que l'alignement obtenu en combinant les données de tous les échantillons. Que remarquez-vous autour des gènes *Hotair*, *Hoxc12*, *Hoxc13* ? Est-ce que vous voyez des différences entre les deux annotations obtenues avec StringTie (avec ou sans annotation de référence) ?

7 Conclusions

Au cours de cette séance de TP, nous avons effectué plusieurs étapes importantes pour l'analyse de données de RNA-seq avec génome de référence :

1. l'alignement des lectures de RNA-seq sur le génome avec HISAT2 ;
2. la visualisation des alignements obtenus avec IGV ;
3. la comptabilisation du nombre de lectures attribuées à chaque gène avec Rsubread ;
4. le calcul des niveaux d'expression normalisés (TPM), en utilisant kallisto ;
5. l'analyse de l'expression différentielle entre deux conditions, avec DESeq2 dans R ;
6. l'analyse d'enrichissement en catégories fonctionnelles Gene Ontology avec GORILLA ;
7. l'amélioration des annotations génomiques avec StringTie.

Ces analyses fournissent déjà de bons éléments de réponse pour la question biologique qui nous intéressait, qui était d'identifier et d'analyser les gènes cibles de l'ARN non-codant Hotair dans ces tissus. Nos analyses d'expression différentielle nous ont permis de mettre en évidence de nombreux gènes qui sont différentiellement exprimés dans les souris mutantes par rapport aux souris sauvages, aussi bien dans les membres que dans le tronc. Cependant, les gènes *Hoxd*, initialement proposés comme cibles de l'ARN non-codant Hotair, n'apparaissent pas comme fortement différentiellement exprimés, dans aucun des deux tissus analysés, ce qui contredit ainsi la fonction initialement proposée pour Hotair.

Références

[1] A. R. Amândio, A. Necsulea, E. Joye, B. Mascrez, and D. Duboule. Hotair Is Dispensible for Mouse Development. *PLoS Genet.*, 12(12) :e1006232, December 2016.

[2] J. L. Rinn, J. K. Wang, S. L. Squazzo, X. Xu, S. A. Brugmann, L. H. Goodnough, J. A. Helms, P.J. Farnham, E. Segal, and H. Y. Chang. Functional demarcation of active and silent chromatin domains in human HOX loci by noncoding RNAs. *Cell*, 129(7) :1311–1323, June 2007.

[3] L. Li, B Liu, O. L. Wapinski, M Tsai, K. Qu, J. Zhang, M. Lin, F. Fang, R. A. Gupta, J. A. Helms, and H. Y. Chang. Targeted disruption of Hotair leads to homeotic transformation and gene derepression. *Cell Rep*, 5(1) :3–12, October 2013.

[4] D. Kim, B. Langmead, and S. Salzberg. HISAT : A fast spliced aligner with low memory requirements. *Nat. Methods*, 12(4) :357–360, April 2015.

[5] M. I. Love, W. Huber, and S. Anders. Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. *Genome Biol.*, 15(12) :550, 2014.

[6] E. Eden, R. Navon, I. Steinfeld, D. Lipson, and Z. Yakhini. GOrilla : A tool for discovery and visualization of enriched GO terms in ranked gene lists. *BMC Bioinformatics*, 10 :48, February 2009.