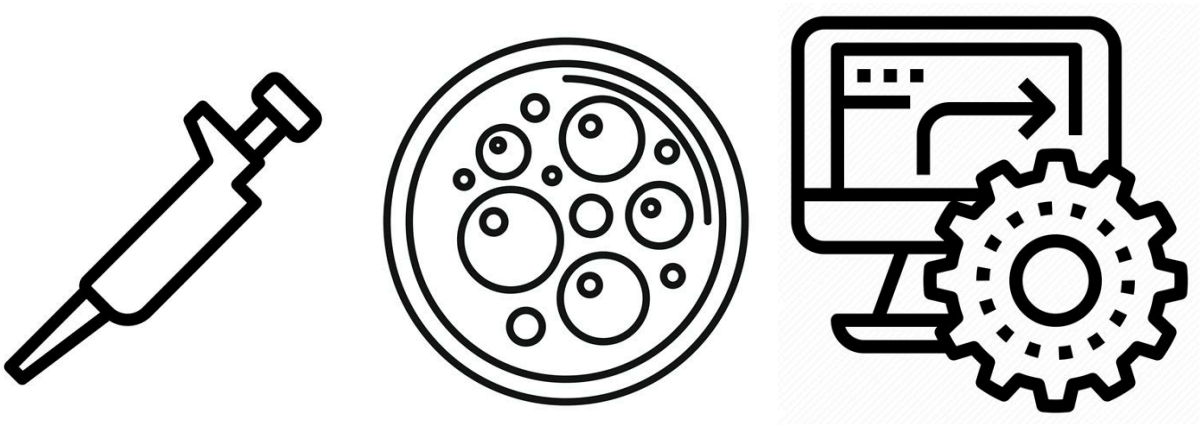
M2 - Analyse, Modélisation Et Ingénierie De L'information Biologique Et Médicale

Rapport UE ReproHackaton :

2024-25

Évaluation de la reproductibilité des résultats d'une étude transcriptomique du Staphylocoque doré intracellulaire.



https://github.com/Tikings/reprohackaton

BOUCHET Adrien
FOURMIGUE Vincent
ROTH Elise
SANCHEZ Timothée

1. Introduction

1.1. Reproductibilité en biologie et bioinformatique

Ce projet s'inscrit dans le cadre d'un travail autour de la thématique de la reproductibilité en biologie. En effet, ces dernières années, la recherche a vu son nombre de publications augmenter drastiquement, les publications étant devenues un moyen de s'assurer une bonne visibilité et donc des financements. On a assisté à la culture du "publish or perish". Cette course à la publication a engendré une "crise de la reproductibilité" qui englobe plusieurs phénomènes nuisant à la conduite d'une recherche qualitative : sans biais, reproductible et éprouvée. Cela s'exprime par le manque d'information dans les papiers autours des méthodes expérimentales et de l'analyse des données, de mauvaises pratiques, un manque d'études réplicatives qui empêchent d'éprouver correctement les résultats et nuisent à la fiabilité de la recherche publiée.

On distingue plusieurs types de reproductibilité en biologie. La reproductibilité empirique, liée aux variations expérimentales et à la qualité des descriptions de collecte de données ; la reproductibilité statistique, qui concerne le choix et la justification des tests et paramètres ; et la reproductibilité computationnelle, qui s'intéresse aux outils, aux codes et aux méthodes de manipulation des données. Cette dernière constitue un véritable enjeu en bioinformatique.
Dans le cadre de ce cours, nous allons uniquement explorer les solutions techniques pour évaluer et reproduire une partie d'une étude scientifique de microbiologie. Nous allons examiner la reproductibilité computationnelle ainsi que la reproductibilité statistique. Est-ce que l'on arrive à reproduire les résultats statistiques de l'étude, en utilisant les outils mis à disposition par le papier ? Quel niveau de reproductibilité le papier peut permettre ?

1.2. Présentation du papier étudié :

Le papier que nous allons étudier est : Peyrusson, F., Varet, H., Nguyen, T.K. et al. - I*ntracellular Staphylococcus aureus* persisters upon antibiotic exposure. *Nat Commun* **11**, 2200 (2020). https://doi.org/10.1038/s41467-020-15966-7. Il aborde la thématique des bactéries persistantes qui sont des sous-populations de bactéries qui une fois exposées à stress (comme un antibiotique) changent leurs profils d'expressions, les rendant ainsi résistantes à certains traitements.
On peut les caractériser par un état de non-croissance et une tolérance au stress. Cette persistance peut s'expliquer par un changement du profil d'expression de ces bactéries.
Cet article s'intéresse donc à comment l'expression des gènes du Staphylocoque doré intracellulaire, suspecté comme faisant partie des bactéries capables de persistances, diffèrent selon qu'il soit dans son état de persiteur ou non. La finalité de l'expérience est une analyse différentielle qualitative et quantitative du profil d'expression des bactéries via le séquençage de leur ARN.

On compare l'expression de deux échantillons. Un échantillon ne contient que des bactéries et un lysat cellulaire : c'est l'échantillon contrôle. L'autre échantillon contient des bactéries persistantes dans des cellules : il a été obtenu en exposant des bactéries à un antibiotique et en ne conservant que les bactéries qui ont survécu. Les ARN respectifs ont ensuite été séquencés par un protocole classique

1.3. But du projet

Le but est donc de reproduire certaines figures de l'article en partant de la même base de donnée qui est disponible en ligne et de fournir un module exécutable depuis n'importe quel ordinateur pour obtenir les mêmes résultats.

Les figures que l'on souhaite reproduire sont des "MA-plots" : ce sont des graphiques qui permettent de comparer l'expression de gènes selon deux conditions. L'axe des abscisses (A pour Average) représente la moyenne des intensités (logarithmiques) des lectures pour chaque gène entre les deux conditions. Cela reflète l'abondance (en termes de transcription) moyenne du gène. L'axe des ordonnées représente le LogFoldChange qui mesure le changement relatif d'expression d'un gène entre deux conditions. Ainsi, on s'intéresse aux gènes qui sont éloignés de l'axe des abscisses, indiquant un changement d'expression, et avec une abondance modérée à élevée, plutôt sur la droite du graphique.
Ici, l'article souligne que les gènes relatifs à la traduction sont exprimés de manière inhabituelle dans le cas des bactéries persistantes.

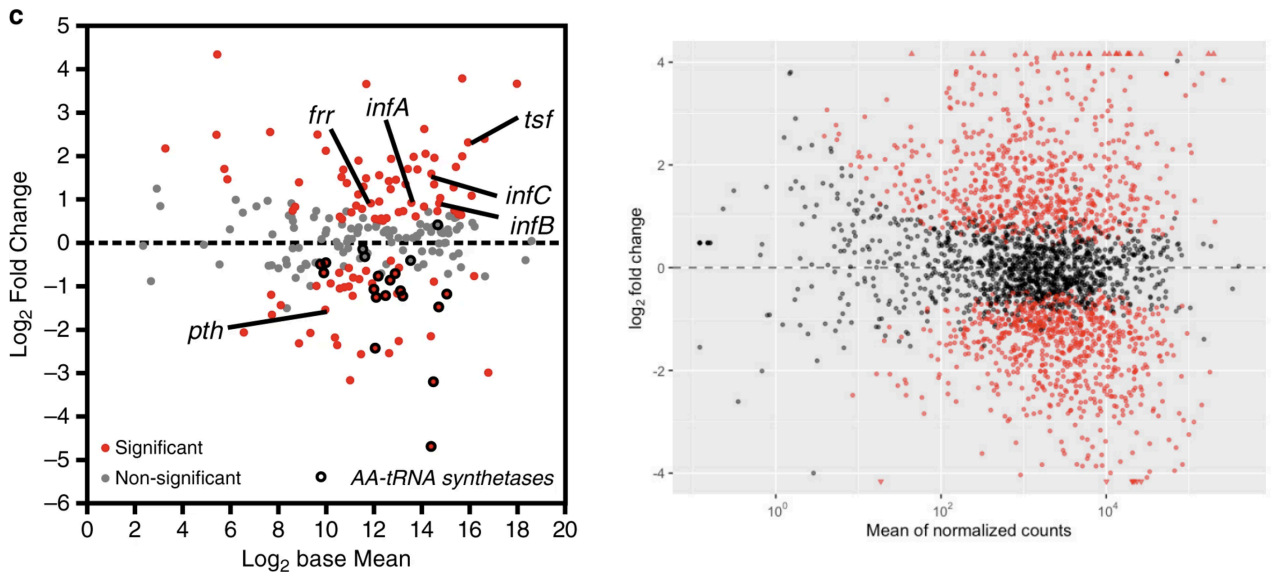


Figure 1 : *Figures à reproduire : (1) MA-plot of genes related to translation. The graph displays the log2 Fold Change expression as a function of log2 Base Mean (mean expression signal across all samples). Typical members of the function are pointed and aminoacyl-tRNA synthetases are shown in black. The dotted line indicates the basal expression level in control samples. Statistical significance is based on adjusted P-value. ; (2) MA-plot of complete RNA-seq dataset. The graph displays the log2 Fold Change expression as a function of mean counts across all samples. Red dots represent significantly differentially expressed features, based on adjusted P value.*

2. Matériel et méthodes

2.1. Outils généraux pour l'analyse de données et la reproductibilité

Pour procéder à l'analyse du séquençage de l'ARN du papier, nous allons avoir besoin de différents outils. L'enjeu est de reproduire les environnements utilisés qui comprennent les outils, ainsi que les bonnes librairies et leurs dépendances associées en reprenant les versions utilisées à l'époque de l'analyse afin de tenter de reproduire les résultats obtenus par les auteurs du papier.
Afin de gérer ces différents environnements, on utilise des conteneurs. Ce sont des environnements d'exécution légers qui permettent d'isoler les applications déployées pour éviter qu'elles interfèrent entre elles mais aussi pour pouvoir transmettre plus facilement ces outils afin

que nos résultats puissent être reproductibles. Ainsi nous allons pouvoir créer des conteneurs pour chacun des outils nécessaires à l'analyse, permettant de les isoler, de conserver la bonne version et d'importer les librairies nécessaires.

Pour ce projet, nous allons utiliser Docker, c'est une des plateformes les plus utilisées, disponible gratuitement et qui profite d'une grande communauté d'utilisateurs. Les conteneurs Dockers se créent à partir d'un fichier texte qui décrit les étapes nécessaires pour construire un environnement reproductible et autonome. Il définit les dépendances logicielles, la configuration et la commande principale de l'application. L'exécution de ce fichier (Dockerfile) permet d'obtenir une image qui en est la version prête à être exécutée et partagée grâce à Docker Hub, une plateforme d'échange des outils ainsi créés. Finalement le conteneur est une instance en cours d'exécution de l'image.

Un autre enjeu de l'analyse est de relier et faire communiquer les différents éléments de l'analyse, qui ne sont pas forcément dans des formats compatibles ou issus des mêmes languages. Pour pallier cela, on utilise des systèmes de workflow qui permettent de structurer des séquences d'analyse et de les automatiser tout en permettant un déploiement sur des clusters de calculs qui pourront paralléliser les opérations qui le nécessitent. Nous allons grâce à ce genre d'outils pouvoir lancer des séquences d'analyse, réutiliser leurs résultats dans un autre outil afin de pouvoir lancer une analyse multi-support de bout en bout. Nous allons pour ce projet utiliser Nextflow.

Un workflow Nextflow est constitué par l'assemblage de différents processus. Chaque processus peut être écrit dans n'importe quel langage de programmation qui peut être exécuté par la plateforme sur laquelle nous lançons l'analyse. L'analyse sera d'autant plus robuste et portable si à cela on couple les images Docker associées. Tout processus peut définir un ou plusieurs canaux indépendants comme 'input' et 'output'. L'interaction entre ces processus, et finalement le flux d'exécution du workflow lui-même, est implicitement définie par ces déclarations input et output.

L'utilisation des conteneurs et d'un système de workflow va pouvoir permettre de créer un ensemble exécutable, bien versionné de l'analyse ARN donnée dans l'article, tout en assurant la mise à disposition des outils utilisés grâce à Docker/ Docker Hub et des commandes utilisées grâce au process Nextflow.

2.2. Récupération des données génomiques : de référence et issues du séquençage NCBI & sra-toolkit

a. Récupération des données génomiques : de référence et issues du séquençage

La première étape à la base de tout le reste de l'analyse est le téléchargement des données de séquençage, et du génome de référence. Les auteurs mentionnent dans matériels et méthodes les *identifiants* du génome de référence (CP000253.1 sur NCBI), ainsi que ceux des 6 échantillons déposés par les auteurs (3 échantillons test, 3 contrôles).

Nous téléchargeons le génome de référence dans un fichier nommé reference.fasta, à l'aide de l'outil `curl` sur ce site.

Les arguments (type de fichier souhaité, identifiant de génome) sont transmis au serveur en paramètres de la requête. Nous exécutons cette commande dans un processus workflow ayant pour entrée un canal issu de la variable `linkRefGenome` (chaîne de caractères) et retournant le fichier `fasta` à travers le canal `ref_genome`. L'opération étant très basique et n'étant pas basée sur des outils particuliers, nous avons estimé qu'il n'était pas nécessaire de l'exécuter dans un conteneur.

Nous aurions pu, de la même manière, télécharger les fichiers .fastq issus du séquençage avec leurs liens associés et l'outil curl. Néanmoins, cette méthode n'est pas optimale (changement de lien, de nom de domaine…), et nous décidons plutôt d'utiliser l'outil `sra-toolkit` fourni par NCBI. Celui-ci permet de télécharger directement les fichiers `fastq` par la commande `fasterq-dump`. Afin d'obtenir une reproductibilité maximale, nous avons créé un conteneur Docker (adbouc/sra-toolkit_v2) basé sur Ubuntu:24.04 pour l'exécuter. On utilise la commande : `fasterq-dump --threads <nb-CPUS> --progress <SRAID>`**.** La variable SRAID correspond au numéro d'accession de l'échantillon sur NCBI : nous définissons une liste SRAIDs d'objets string contenant ces identifiants dans le fichier de configuration nextflow.config. La commande est exécutée au sein du processus `downloadFastq`, ayant pour input les valeurs SRAID issu du canal basé sur la liste SRAIDs (par la méthode channel.fromList()) et pour output les fichiers `fastq` téléchargés, transmis au canal `fastq_files`**.**

2.3. Traitement des séquences ARN des échantillons

2.3.1. Prétraitement / Découpage des séquences avec `Cutadapt`

Une fois les échantillons indexés et annotés, nous allons pouvoir procéder au prétraitement des séquences. Pour ce faire, nous utilisons le logiciel Cutadapt mentionné dans l'article. Il permet de retirer les les séquences non-nécessaires à l'analyse, comme les adapteurs (séquences artificielles ajoutées lors de la préparation des échantillons comme les amorces) ou les queues poly-A ; il supprime les séquences trop courtes ou de mauvaise qualité ainsi que les contaminants (séquences non spécifiques ou indésirables). Ainsi on essaye de conserver uniquement les séquences biologiques d'intérêt.
C'est un outil développé sur Python. L'article nous fournit la version : `cutadapt:1.11`, et un paramètre de pré-traitement : la longueur minimale des séquences : 25 nucléotides. D'autres paramètres ne sont cependant pas spécifiés.

Pour former l'image, nous avons utilisé une image Python de base, qui est suffisante et légère. On utilise `Python 2.7`, la version compatible avec la version 1.11 de `Cutadapt`.

Les autres paramètres requis pour permettre ce prétraitement des séquences sont les suivant :

- L'adapteur utilisé par le kit de séquençage, dans notre cas le papier à utiliser le TrueSeq stranded mRNA Sample preparation Kit.
- Qualité minimale requise pour qu'une séquence soit conservée (cela est visualisable avec l'outil fastqc).
- Le taux d'erreur maximal des séquences qui est par défaut à 0.1
- L'overlap minimum requis pour éviter que l'on supprime certaines séquences qui pourraient apporter de l'information à l'analyse. Il est par défaut à 3.

Ces 4 paramètres ne sont pas mentionnés dans le papier mais sont cependant requis pour faire le trimming.
Nous avons décidé d'utiliser les paramètres suivants qui sont les suivants :

- Séquence d'adapteur : "AGATCGGAAGAGC" qui est celle détectée par l'outil trim-galore que nous avons initialement utilisée pour se familiariser avec l'analyse.
- La qualité minimale requise égale à 20 qui semble être la valeur la plus communément utilisée et également celle suggérée par nos encadrants.
- Le taux d'erreur est resté celui par défaut par soucis d'indications dans le papier

- L'overlap minimum a été choisi à 1 pour garantir une qualité maximale des séquences conservées après cette étape.

2.3.2. Indexation, alignement

Après cette étape de prétraitement, on dispose des fichiers fastq compressés directement par la fonction `Cutadapt`. Ces fichiers contiennent les reads qui sont de qualité suffisante pour être analysés.

On a donc pu procéder à l'alignement des séquences sur le génome de référence qui a ensuite été trié. Pour effectuer cette étape, le papier utilise l'outil `bowtie version 0.12.7` pour l'alignement des séquences mais il ne fait cependant pas mention du tri des séquences qui s'ensuit. Nous avons tout de même décidé d'utiliser la commande `samtools sort` pour pouvoir produire un fichier au format `bam` pour chaque séquence qui est un format compressé du standard `sam` de `samtools` et qui peut directement être utilisé par les commandes qui nous permettra de faire le comptage.

L'image Docker que nous avons produite se base sur une version `Ubuntu:20.04` dans laquelle nous avons installé la version mentionnée de `bowtie` depuis une tarball disponible sur le site source forge. Le binaire ainsi obtenu après décompression a été placé via un lien symbolique dans le répertoire contenant les binaires pour une accès simplifié à la commande. La commande `samtools` quant à elle a été installée de manière plus conventionnelle via la commande `apt` disponible sur Linux qui installe la dernière version disponible.

Une fois le conteneur prêt, il nous a suffit de récupérer le génome de référence téléchargé préalablement pour produire un ensemble de fichiers d'index eu format `ebwt` nécessaires à l'alignement en utilisant la commande `bowtie-build`. Une fois cette étape terminée, l'alignement est effectué par bowtie sur chaque fichier `fastq` (décompressé au préalable) pour ensuite être "pipe" dans `samtools sort` qui nous produira les fichiers au format `bam` associés qui seront ensuite transmis au prochain processus.

2.4. Comptage des séquences et annotations des reads

Après l'obtention des séquences alignées par rapport au génome de référence, la fonction `featureCounts` issue de la bibliothèque de packages `subread` nous permet d'effectuer un comptage qui comme son nom l'indique est une évaluation du nombre de séquences qui sont alignés à une portion donnée du génome.

Pour produire l'image associée à ce processus, nous nous sommes basés sur `Ubuntu:20.04` pour la même raison que précédemment. Nous avons ensuite récupéré le tarball contenant tous les binaires précompilés que nous avons juste eu à extraire et à ajouter dans les binaires de l'image pour une utilisation plus simple de la commande.

Cette étape est relativement simple et rapide et ne requiert qu'un fichier d'annotation du génome pour faire la passerelle entre les positions du génome et les gènes associés. Ce fichier peut-être trouvé sur le site NCBI.

Les autres paramètres utilisés sont les suivants :

- g : spécification du type d'attribut utilisé pour grouper les features
- F : Le type d'annotation utilisé, ici nous téléchargeons un fichier d'annotation au format gff et on spécifie donc la valeur "GTF"
- s : paramètre qui indique quel type de comptage on doit effectuer. Dans le papier la valeur mentionnée est 1 pour un 'strande-specific read counting'.

La sortie de ce processus est un unique fichier au format txt qui contient un tableau du nombre d'alignements comptés pour chaque gène par individu.

2.5. Analyse statistique avec R

Après l'obtention du tableau de comptage que nous obtenons, l'analyse R peut être effectuée.
Dans les informations fournies dans le papier, nous disposons des informations suivantes : Il nous faut le logiciel R en version 3.4.1 ayant accès aux librairies (et toutes les dépendances associées) `DESeq2 version 1.16` et `EnrichmentBrowser 2.14.3`.
L'image Docker produite dans le cadre de nos travaux se base sur `Ubuntu:20.04` et compile R en version 3.4.1 à partir d'une tarball disponible sur le site officiel (cf. annexe). Une fois la compilation terminée, les packages sont ensuite installés depuis un miroir du CRAN datant de quelques mois après la sortie de cette version de R, ce qui est censé garantir la disponibilité des packages que l'on veut. Ce miroir est disponible sur ce site et la date utilisée est celle du 11-01-2028.
Les packages gérés par `Bioconductor` (version 3.5 ici), qui est un gestionnaire de packages de bio-informatique pour R, sont téléchargés depuis le site officiel.
Nous nous retrouvons avec les bonnes versions de `R (3.4.1)` et de `DESeq2 (1.16.1)` prêtent à l'emploi.
L'image ne colle cependant pas aux pré-requis du papier car la version d'`EnrichmentBrowser` utilisée ici est la 2.6.3 et non la 2.14.3 mentionnée dans les matériels et méthodes. Cela provient du fait qu'elle requiert des dépendances qui n'existaient pas en R 3.4.1 et est donc impossible à installer dans ce container. Le choix a été fait de ne travailler qu'avec cette version car elle est uniquement utilisée pour faire le pont entre KEGG et notre script pour récupérer les pathways des gènes que l'on a mis en évidence et la version 2.6.3 permet également de le faire sans problème.

À partir de là, l'analyse R a été faite via la librairie `DESeq2 version 1.16.1` (Reproduction de la version utilisée pour les figures du papier). Le fichier de comptage produit par l'aval du workflow est ainsi traduit en une patrice utilisable par `DESeq2` notamment grâce `DESeqDataSetFromMatrix` et aux informations issus de l'expérience à savoir quel *SRAID* est un contrôle ou un persisteur. Ensuite on a cherché à refaire la première figure en créant une nouvelle colonne pour la significativité de chaque point selon le seuil donné dans l'article de *0.05*. A noter que dans une approche plus récente d'analyse d'expression différentielle on aurait à cette étape réaliser un shrinkage des données pour obtenir un *logfoldchange estimate lfcSE* or les différentes méthodes utilisées lors de cette étape apparaissent dans la littérature aux mêmes dates de publications que notre papier et ne sont pas implémentées dans la version que nous utilisons de `DESeq2`. Ce shrinkage n'impacte pas la première image *log2FoldChange=f(baseMean = Means of normalized counts)* mais aura une influence sur la seconde image.

La deuxième image " MA-plot of genes related to translation " est quant-à elle sujet à plus de questionnement. Dans la légende [78] de l'article devant expliquer le lien à la traduction n'est qu'un lien vers KEGG sans aucune autre information alors que la recherche dans KEGG de "translation" est prolixe et fait ressortir plusieurs questions : Quels sont les *sao* des groupes de gènes utilisés ? Les sao utilisés (lien entre pathways et identifiants de gènes) avec notre approche sont se trouvés dans le *sao09122* (i.e. 03010, 00970 et 03008 seul ?) et les *facteurs de traduction sao03012*. Cependant lors de l'utilisation de tous ces gènes il y a moins de points dans la sélection kegg que dans l'image du papier. Ces pathways kegg sont télécharger directement depuis `DESeq2` via la fonction `download.kegg.pathways("sao")`. Ce moyen de récupération des sao ne permet pas de récupérer *sao03012* correspondant au facteurs de traduction surement à cause du non maintien du lien entre les anciennes versions de `DESeq2` et l'API de KEGG. Aussi le seul

gène de *sao03008* n'est pas récupérable non plus via l'API donc vu que en le spécifiant à la main dans le script R on ne voit aucune différence dans aucune des deux images il n'est pas spécifié dans la version finale du script R. D'autre part dans le papier les étiquettes de certains points sont présentes, elles correspondent au `pan_gene_symbol` téléchargés depuis la base de données de Aureowiki NCTC8325 `curl` directement depuis internet en input du processus.
A partir de la légende de l'image, on spécifie dans une nouvelle colonne le type des sao pour un traitement spécifique lors de l'affichage. Les "AA-tRNA Synthetase" sont marqués pour être entourés aussi les gènes à être étiquetés sont marqués "Typical members" (ils semblent correspondre d'ailleurs aux facteurs de traduction.
Le plot de la seconde image en plus de toutes ces conditions d'affichage est lui effectué à partir du *log2(BaseMean)* en abscisse, on garde pour *p-adj 0,05*.

2.6. Autres réglages généraux du workflow

2.6.1. Optimisation du traîtement des données : choix de la compression

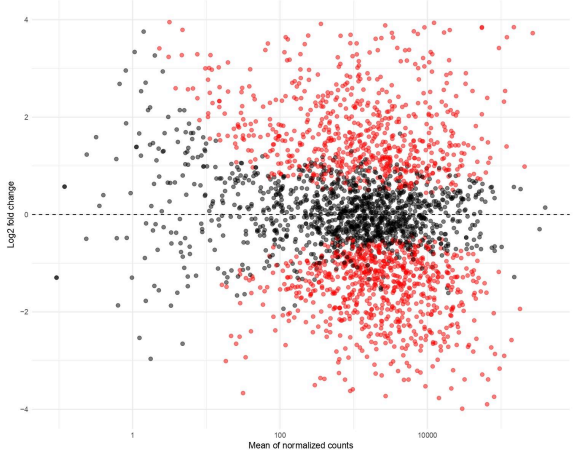
À plusieurs étapes, il a été possible d'optimiser le workflow pour que celui-ci soit moins demandant en ressources de calculs et en stockage. Nous avons fait le choix de réduire le stockage car cela rend le workflow plus portable.
En effet chaque fichier fastq téléchargé pèse environ 10Go et tous les fichiers issus du trimming également. On se retrouve donc très vite à saturer la mémoire en données.
Pour palier à ce problème, nous avons opté pour une compression à ces étapes en utilisant la commande `gzip`, qui nous a permis de compresser les fichiers en des fichiers environ 10 fois plus légers mais cela rend cependant les processus associés plus longs à exécuter.

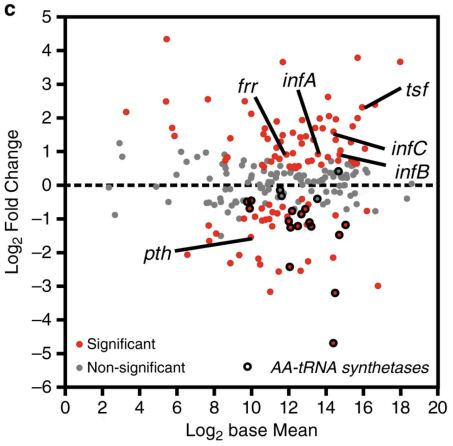
2.6.2. Assurer la compatibilité multi-architecture

Une problématique que nous avons rencontré lors de la production des images est que l'image docker est relative à l'architecture sur laquelle elles sont construites. En effet, depuis l'arrivée des nouvelles architectures ARM sur le marché (principalement sur les ordinateurs Apple) il existe des incompatibilités pour faire fonctionner les images sur certaines machines. En effet une image construite sur ARM (sans spécification d'architecture lors de la construction) ne pourra pas fonctionner sur un processeur de type AMD. L'inverse est cependant possible et nous avons donc décidé de construire toutes nos images sur une base `linux/amd64` pour la rendre la plus portable possible.

3. Resultats

3.1. Comparaison graphes 1 et 2





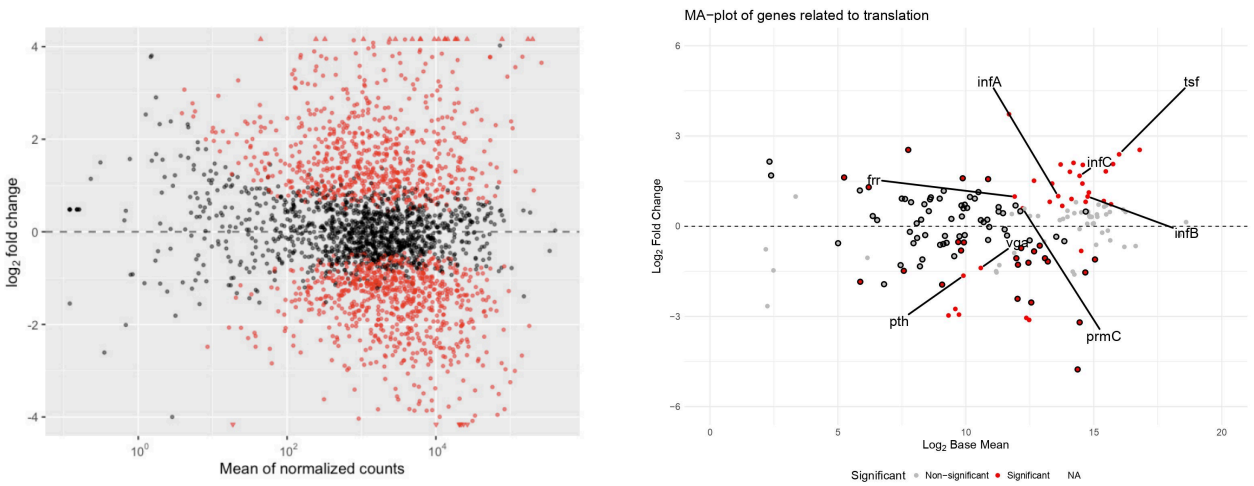


Figure 2 *: Comparaison des résultats du papier (au dessus) et des résultats obtenus à l'aide du Workflow avec les settings* `cutadapt` *donné dans la partie précédente*

Pour la première image, les deux sont très similaires, même si une étude attentive permet de discerner des points qui sont différents.
Pour la seconde image, les deux semblent différentes, d'abord parce que l'affichage carré est forcé dans le papier aussi on retrouve plus de points entourés dans notre version de l'image ce qui laisse supposé que les gènes qu'ils ont utilisés comme AA-tRNA synthetase ne sont pas tous ceux que KEGG fournit aujourd'hui. De plus, ces points ne sont pas dans notre figure aussi clairement exprimés négativement que dans le papier. Cependant, on retrouve des motifs de points dans le papier et dans notre reproduction : le motif vertical de quatre points significatifs et appartenant au groupe "AA-tRNA S.". Aussi les points étiquetés sont situés au même endroit dans les deux figures avec la même significativité.
On voit bien que, en terme de reproductibilité, spécifier des gènes dans un papier simplement en les nommant "typiques" ou "liés à la traduction" n'est pas suffisant pour les retrouver, que ce soit à cause de la variabilité de KEGG dans le temps ou à cause de problèmes liés au fonctions utilisées dans notre workflow .

3.2. Autres graphes et fine-tuning des paramètres

La différence entre les deux graphes provient probablement du prétraitement des données. C'est une étape pour laquelle on a peu d'information et qui est le plus susceptible d'être à l'origine de ces variabilités de par son nombre de paramètres.

Pour vérifier cette hypothèse nous avons essayé de faire tourner le process en modifiant un paramètre, ici nous avons dans un premier temps choisi de modifier la qualité minimale requise en passant d'une valeur initialement à 20 à une valeur de 30. On obtient ainsi les deux graphiques suivant (MA-plot complet) :

Figure 3 *: à gauche la figure initiale obtenue (q = 20) et à droite la figure avec une paramètre de qualité différent (q=30)*

On observe que les points avec un count faible ne sont plus présents mais nous nous éloignons de la figure initiale. Cela montre qu'il y a bien une grande influence du trimming sur les résultats finaux, mais ce paramètre ne semble pas être celui impliqué.

Nous avons également produit les images avec d'autres paramètres :

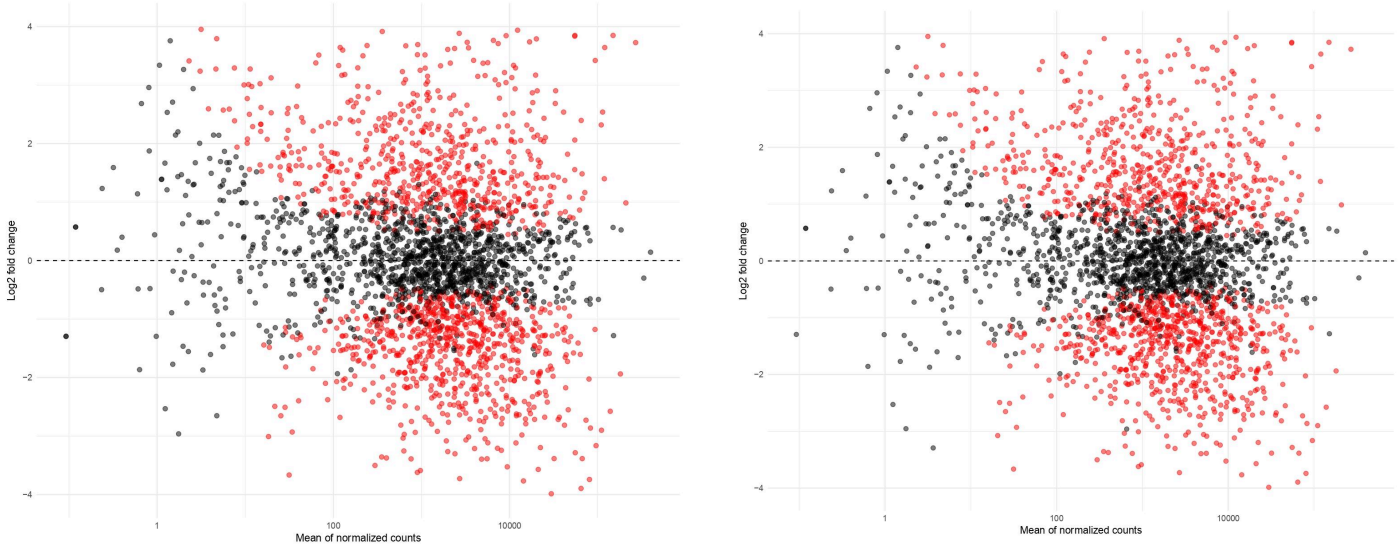


Figure 4 *: à droite image issue avec une modification de l'adapteur (issue du site Illumina : AGATCGGAAGAGCACACGTCTGAACTCCAGTCA au lieu de AGATCGGAAGAGC donné par* `trim_galore` *) et à gauche avec le paramètre d'overlap min (égal à 3 et non 1)*

Les images ont des modifications subtiles mais elles ne provoquent pas de changement drastique et ne permettent pas de se rapprocher des résultats du papier.

Au vu de ces différents essais, le trimming semble effectivement être le facteur le plus impactant pour notre problème de reproductibilité des résultats du papier. Il serait possible d'effectuer une approche similaire à une recherche en grille utilisée en machine learning pour essayer de retrouver les paramètres qui se rapprocherait le plus de ceux présentés dans le papier. Mais au vu du temps nécessaire au traitement d'une combinaison de paramètres, nous ne sommes pas parvenus à expérimenter cette approche. Bien que cela serait intéressant si il y avait une vraie nécessité de reproduire les figures à l'identique.

Un workflow `grid_search.nf` donnée dans le github permet de tester cette approche mais il est très demandant en ressources et le lancer aurait requis un temps considérable que nous n'avons malheureusement pas eu.

4. Conclusion

Comme nous l'avons vu, les éléments de méthode fournis par l'article permettent d'obtenir des résultats relativement similaires à ceux présentés par les auteurs. Toutefois, le manque de certaines informations techniques (paramètres utilisés pour certains outils) nous oblige à les choisir nous-même. Il est souvent difficile de savoir quel est l'impact du choix d'un paramètre particulier sur le traitement global des données, et donc de déterminer ceux permettant de reproduire les résultats avec exactitude. En ce sens, il est essentiel pour les auteurs d'expliciter les paramètres des outils qu'ils utilisent afin de garantir la reproductibilité et la fiabilité de leurs résultats.

Par ailleurs, la maîtrise de l'environnement de traitement des données (versions des outils utilisées, dépendances) est un autre enjeu essentiel pour la reproductibilité des résultats scientifiques. Notre travail montre qu'il est parfois difficile d'avoir accès à des versions spécifiques de certains outils/packages, ou même qu'il est impossible de faire cohabiter certaines d'entre-elles (versions de `R:3.4.1` et d`'EnrichmentBrowser:2.14.3` spécifiées dans l'article, mais incompatibles).

Des solutions permettant de garantir l'état d'un environnement d'exécution existent pourtant : nous pensons par exemple à la R Session, qui, si elle avait été communiquée, n'aurait pas permis d'ambiguïté quant aux versions de R et EnrichmentBrowser utilisées.

Au-delà même des informations communiquées en Matériel et Méthode, la reproduction d'une analyse scientifique est un processus chronophage et compliqué. Des outils comme la conteneurisation (à l'aide de Docker, par exemple) permettent à la fois de fixer l'état d'un environnement d'exécution, d'assurer la disponibilité de versions spécifiques de certains outils (en les figeant dans des images), et de faciliter le partage de ces environnements (avec Docker Hub, par exemple). L'utilisation des workflows (comme Nextflow) permet également d'assurer un environnement cadré et d'expliciter les paramètres des différents outils utilisés ainsi que leur enchaînement tout au long de l'analyse, en étant portable et facilement réutilisable. Combinés ensemble, la conteneurisation des outils utilisés et la publication d'un workflow permettent à quiconque disposant des données de reproduire les résultats obtenus par l'analyse bio-informatique rapidement et simplement.

Finalement, l'article nous permet donc d'obtenir un niveau de reproductibilité de type "replicate", selon les définition de *Cohen-Boulakia et al., FGCS, 2017*. En effet, nous avons effectué l'analyse de données dans un environnement légèrement différent (versions parfois différentes, certains paramètres ont été choisis ou mis par défaut), mais le protocole de l'analyse reste le même et nous obtenons des résultats assez semblables dont l'interprétation biologique reste la même.