# Méthodes informatiques pour l'analyse de données de séquençage de petits ARN

Matthias Zytnicki

Université de Toulouse

Habilitation à diriger des recherches
en informatique

présentée et soutenue publiquement le 13 avril 2022 par

Matthias Zytnicki

# Méthodes informatiques pour l'analyse de données de séquençage de petits ARN

INRAE — Unité Mathématiques et Informatiques Appliquées de Toulouse
École Doctorale « Mathématiques, Informatique, Télécommunications de Toulouse »

**Jury**

| | | | |
|---|---|---|---|
| M. | Dominique Lavenier | Rapporteur | Directeur de recherche |
| Mme | Marie-France Sagot | Rapportrice | Directrice de recherche |
| Mme | Hélène Touzet | Rapportrice | Directrice de recherche |
| M. | Alain Denise | Examinateur | Professeur |
| M. | Yann Ponty | Examinateur | Directeur de recherche |
| M. | Éric Rivals | Examinateur | Directeur de recherche |
| Mme | Christine Gaspin | Examinateur | Directrice de recherche |
| Mme | Gwennaële Fichant | Examinateur | Professeure |

# TABLE DES MATIÈRES

# AVANT-PROPOS

## PRÉSENTATION DU DOCUMENT

Le manuscrit déposé en vue d'obtenir l'habilitation à diriger des recherches poursuit deux objectifs contradictoires. Le premier est qu'il doit présenter, de façon plus ou moins exhaustive, les travaux du candidat. Le second est qu'il doit développer une structure cohérente.

Ne pouvant satisfaire aux deux exigences à la fois, j'ai donc choisi de ne parler que d'une partie de mes travaux, centrée autour des petits ARN. Le fil de discussion n'est pas chronologique, mais thématique, et, j'espère, logique (du point de vue du lecteur). En l'occurrence, j'ai souhaité montrer comment les outils que j'ai proposés peuvent être utilisés successivement afin d'analyser des données de séquençage de petits ARN. J'espère montrer que mes travaux forment un tout cohérent.

Ceci implique donc de ne pas présenter, dans le corps du manuscrit, une large de partie de mes travaux. Pour être complet, je vais toutefois brièvement mentionner mes travaux non présentés en détail dans la section suivante.

J'ai également pris le parti de ne pas parler des applications en biologie des outils que j'avais développés. Ces outils ont en effet été créés pour répondre à des questions précises posées de concert avec des collègues biologistes. Toutefois, les applications sur lesquelles j'ai été amené à contribuer, même si l'on se restreint à l'épigénétique, sont relativement diverses, et n'ont pas l'unité que je cherchais à produire dans ce document. Je présenterai donc ici uniquement les travaux méthodologiques liés au développement de méthodes informatiques pour l'analyse des petits ARN.

J'espère avoir concilié au mieux les deux objectifs cités précédemment, et vous souhaite bonne lecture.

## RÉSUMÉ DES TRAVAUX NON PRÉSENTÉS EN DÉTAIL

Mes travaux de thèse ont eu pour objet d'étude principal un formalisme d'intelligence artificielle nommé *réseaux de fonctions de coûts*. Celui-ci permet d'exprimer des problèmes d'optimisation combinatoire, en dissociant de façon efficace les méthodes d'inférence (telles que « si $x$ vaut $a$, alors $y$ ne peut valoir que $b$ ») et les mécanismes de recherche (« que se passe-t-il si je suppose que $x$ vaut $a$ »). J'ai notamment proposé plusieurs nouveaux mécanismes d'inférence (Cooper et al., 2008 ; Heras et al., 2005 ; Zytnicki, Gaspin, De Givry et al., 2009), qui donnent de bons résultats, et sont toujours utilisés. Le formalisme des réseaux de fonctions de coûts est suffisamment général pour qu'il puisse s'appliquer à de très nombreux cas, et je l'adapté à la recherche d'ARN non-codants (Zytnicki, Gaspin et Schiex, 2008). Ici, on suppose connue une famille d'ARN non-codants, que l'on peut décrire par des éléments de structure telles que des mots conservés, ou des repliements en tige-boucle. On modélise les éléments par des fonctions de coûts, et on recherche, dans un génome, les régions qui peuvent adopter la conformation décrite.

Lors de mon séjour post-doctoral, je me suis focalisé sur l'analyse des ARN non-codants (Bussotti et al., 2011 ; Ørom et al., 2011), en utilisant notamment pour la première fois des données de séquençage (Campo et al., 2013).

Lors de mon premier poste, à l'URGI (INRA de Versailles), je me suis concentré sur les éléments transposables, qui sont les objets d'étude favori du laboratoire, ainsi qu'à leur régulation par les petits ARN (Zanni et al., 2013). J'ai par exemple étudié ces éléments dans le cadre du consortium d'assemblage et d'annotation du blé, une espèce très riche en éléments transposables (International Wheat Genome Sequencing Consortium,

2014 ; Marcussen et al., 2014 ; Pfeifer et al., 2014), ainsi que d'*Arabis alpina* une plante pérenne étudiée en comparaison avec *Arabidopsis thaliana* (Willing et al., 2015). Pour étudier plus avant ces éléments transposables, j'ai créé un assembleur dédié, qui se propose de constituer des éléments pleine longueur à partir de données de séquençage type Illumina (Dennenmoser et al., 2019 ; Zytnicki, Akhunov et Quesneville, 2014). Pendant cette période, j'ai également travaillé en collaboration avec l'IJPB sur l'étude des petits ARN (miARN et siARN) chez *A. thaliana* (Charbonnel, A. Niazi et al., 2017 ; Parent, V. V. Jauvion et al., 2015 ; Shamandi, Zytnicki, Charbonnel, Elvira-Matelot, Bochnakian, Comella, A. Mallory et al., 2015), ainsi que le remodelage de la chromatine chez la même espèce (Molitor, Latrasse, Zytnicki, P. P. Andrey et al., 2016).

De retour à Toulouse, en plus des travaux exposés ci-après, j'ai principalement collaboré avec ma doctorante Leila Khajavi, autour de l'épigénétique du cerveau (Martínez de Paz et al., 2019) et la narcolepsie (Bernard-Valnet et al., accepté pour publication).

# PROJET DE RECHERCHE : LES PETITS ARN

# 1

# INTRODUCTION

## 1.1 LES PETITS ARN

Les petits ARN eucaryotes sont définis comme des ARN de taille inférieure à 200 paires de bases, habituellement non-traduits. Ils ont été découverts vers les années 2000 et ont été, depuis lors, identifiés comme un objet d'étude particulièrement important. En témoignent le prix Nobel en physiologie/médecine, adressé à Andrew Z. Fire et Craig C. Mellow pour leur travaux sur « la découverte du mécanisme d'ARN interférant, et la répression de gènes par ARN double brin », ainsi que les nombreux articles consacrés à leur étude (Grosshans et Filipowicz, 2008 ; Storz, 2002), notamment sur leur implication dans différents aspects de la vie cellulaire (Borges et Martienssen, 2015 ; Stefani et Slack, 2008).

Les petits ARN sont en général classés selon leur taille, biogenèse, et voie fonctionnelle. Parmi eux, les microARN (miARN), de taille généralement comprise entre 20 et 24 nucléotides, sont certainement les plus connus. Les miARN sont issus de longs transcrits primaires, qui sont rapidement clivés pour obtenir un transcrit précurseur, d'une centaine de nucléotides environ. Ce transcrit précurseur se replie sur lui-même dans une structure caractéristique, reconnue par une machinerie moléculaire, et encore une fois clivée pour obtenir deux petits transcrits, appelés en général miARN mature et miARN *star*. Il est entendu que le miARN *star* est moins exprimé que le mature, même s'il s'agit plutôt d'une convention dans la mesure où l'expression relative du mature et du *star* peut varier considérablement. Le miARN mature, ainsi que le *star* peuvent subir des éditions, où certains nucléotides, souvent aux extrémités 5' et 3' sont modifiées (Blow et al., 2006). La structure caractéristique de l'ARN précurseur, ainsi que l'accumulation de transcrits aux positions bien définies du miARN mature et du *star* font que les miARN sont généralement relativement bien prédits.

Le mode d'action des miARN est complexe, et peut varier entre les animaux et les plantes. Il inclut, dans tous les cas, une hybridisation du miARN mature sur sa cible. Cette hybridisation peut faire intervenir une partie (à partir de 6 à 8 nucléotides, appelée graine) ou l'intégralité du miARN, avec ou sans mésappariement. Un certain nombre de miARN, notamment les plus exprimés, sont regroupés en familles. Deux membres d'une même famille de miARN, situés à des positions génomiques différentes, ont des miARN matures presque identiques, surtout au niveau de la graine. Les membres d'une même famille ont donc, *a priori*, des fonctions similaires.

D'autres classes de petits ARN peuvent également être identifiées grâce à leur structure caractéristique : les petits ARN nucléolaires (snoARN), les petits ARN nucléaires (snARNs ou U-ARN), et les petits ARN dérivés de fragments d'ARN de transfert (tsARN) ont tous des repliements et des mots conservés bien identifiés, qui permettent une reconnaissance relativement aisée.

À l'opposé, on peut citer les ARN associé à PIWI, nommés piARN, que l'on trouve chez les animaux. Ces petits ARN, d'une taille comprise entre 26 et 31 nucléotides, régulent l'expression des éléments répétés. Ils sont issus de deux longs transcrits anti-sens, dont l'un est éventuellement issu un élément transposable. Les transcrits vont être clivés à intervalle relativement régulier, donnant lieu à des piARN. À la différence des miARN, les piARN sont relativement difficiles à prédire, car ils n'ont pas de structure secondaire (pas de repliement caractéristique), et sont éparpillés le long des transcrits anti-sens.

D'autres classes de petits ARN sont définies uniquement par leur position relative à un gène codant, ou à un long ARN non-codant (Affymetrix/Cold Spring Harbor

Laboratory ENCODE Transcriptome Project, 2009). Certains petits ARN sont produits près des sites de début de transcription, près des jonctions exon/intron, dans les promoteurs de gènes, ou bien dans des régions aval. Ils n'ont pas de repliement caractéristique, et leur fonction est mal connue.

## 1.2 LE SÉQUENÇAGE

Le séquençage massif est un outil extrêmement intéressant pour étudier, entre autres, le génome, la régulation (épi)-génétique, la conformation tridimensionnelle de l'ADN, etc. La liste est loin d'être exhaustive et elle est en continuelle expansion.

Même du point de vue du néophyte, le principe général est simple à énoncer : le séquençage permet de lire des séquences d'ADN (voire d'ARN, dans certains cas). Trois types de séquenceurs existent actuellement. Certains peuvent lire des centaines de millions, voire des milliards de séquences courtes d'environ 100 nucléotides avec peu d'erreurs (type Illumina). D'autres séquenceurs produisent des millions de lectures longues d'environ 10 000 à 100 000 nucléotides avec beaucoup d'erreurs (type ONT). Les derniers peuvent lire une séquence très longue sans erreur (type Sanger). Entre ces trois extrêmes, il existe quelques autres séquenceurs, comme le PacBio HiFi, qui peuvent lire des séquences relativement longues sans erreurs. Les coûts, en décroissance constante et soutenue, rendent cette technologie particulièrement abordable.

Le miracle du séquençage réside également dans la sophistication des protocoles biotechnologiques, qui permettent de transformer tout question biologique en séquence d'ADN. Le plus simple à expliquer, le DNA-Seq, consiste en une extraction de l'ADN, suivi d'un clivage (*sonication*) en taille acceptable pour le séquenceur. Il est en général utilisé pour assembler un génome, ou bien trouver les variations génomiques propres à un individu, ou une population. Le RNA-Seq consiste principalement à effectuer une rétro-transcription de l'ARN en ADN complémentaire, puis à effectuer le séquençage. Ainsi, il permet de mesurer l'expression des transcrits et des gènes. Le ChIP-Seq commence par une étape *cross-linking* entre l'ADN et une protéine d'intérêt, une découpe de l'ADN, une immunoprécipitation de la protéine, et une séparation de la protéine avant séquençage. Le Hi-C est un protocole particulièrement complexe permettant séquencer les brins d'ADN relativement proches dans le noyau. Sans entrer dans les détails, plus l'on séquence simultanément la séquence $A$ et la séquence $B$, plus la probabilités que $A$ et $B$ soient proches est grande. Il est extrêmement intéressant de noter la diversité des applications du séquençage.

## 1.3 LES SPÉCIFICITÉS DU SÉQUENÇAGE DE PETITS ARN

Malgré leur grande diversité, tous les petits ARN peuvent être étudiés en une seule manipulation, même si plusieurs protocoles existent (Dard-Dascot et al., 2018). La préparation s'effectue comme suit : purification de l'ARN, sélection par la taille, éventuellement enrichissement en molécules portant une terminaison 5' phosphate, puis séquençage. En fonction de la technologie et du multiplexage, plusieurs dizaines de millions de lectures peuvent être produites. En fonction du protocole, on peut avoir un enrichissement en miARN, ou bien une représentation quantitative de tous les petits ARN.

Par rapport au séquençage génomique ou transcriptomique classique (longs ARN), le séquençage des petits ARN (sRNA-Seq) a plusieurs spécificités :

- Il produit majoritairement des lectures de faible longueur. Quel que soit le protocole, les tsRNA (qui font en général jusqu'à 34 nucléotides) et les miARN sont le plus souvent les ARN majoritaires.
- Une part importante des séquences est lue plusieurs fois. Si, par exemple, un miARN est très exprimé, beaucoup de transcrits identiques seront séquencés.
- Une même séquence peut être produite par plusieurs régions génomiques, sans qu'il soit possible, *a priori* d'en déterminer le *locus* d'origine. Dans le cas des miARN, un

élément peut correspondre à plusieurs membres de la même famille. Dans le cas des piARN, comme ils sont souvent produits par des transcrits d'éléments répétés, ils peuvent naturellement correspondre à beaucoup de *loci* également.

— Il est parfois difficile d'attribuer une classe à une séquence. Les classes de petits ARN sont parfois mal connues, et mal caractérisées, surtout dans les espèces non modèles. En conséquence, l'annotation des petits ARN est souvent très lacunaire. Si il est général simple d'identifier *de novo* les lectures provenant de miARN, c'est beaucoup plus difficile pour les piARN, et il existe sans doute des classes encore inconnues.

## 1.4 ANALYSE DES DONNÉES DE SÉQUENÇAGE DE PETITS ARN

Le pipe-line d'analyse des petits ARN dépend beaucoup des questions biologiques que l'on se pose. Toutefois, on distingue en général deux types d'analyses principales :

*A* Constitution du répertoire des petits ARN. Il s'agit ici simplement de compter les petits ARN présents dans chaque classe.

*B* Recherche des petits ARN différentiellement exprimés. On suppose ici un contexte différentiel, où l'on compare, par exemple, un tissu sain et un tissu malade, et on recherche tous les petits ARN qui sont exprimés différemment entre les deux tissus.

Pour ce faire, il existe un certain nombres d'étapes :

1. Nettoyage des lectures. Il faut en général enlever les adaptateurs, et supprimer les lectures de basse complexité (contenant des répétitions de `AAAAA`, par exemple).
2. Alignement des lectures sur le génome.
3. Si l'on aligne les lectures sur le génome, et que l'annotation disponible est incomplète, il peut être utile de rechercher de nouveaux petits ARN, en se basant sur les lectures.
4. (*pour l'analyse A*) Constituer le répertoire des petits ARN.
5. (*pour l'analyse B*) Quantifier l'expression des gènes, et procéder aux tests d'expression différentielle.
6. (*ou bien*) Si l'annotation disponible est incomplète, rechercher directement les petits ARN différentiellement exprimés.

Je présenterai, dans ce rapport, mes contributions sur les points 2 à 6 précédemment présentés. Elles constituent ma volonté de proposer une ensemble d'outils cohérents et complémentaires pour l'analyse de séquençages de petits ARN (voir figure 1.1).

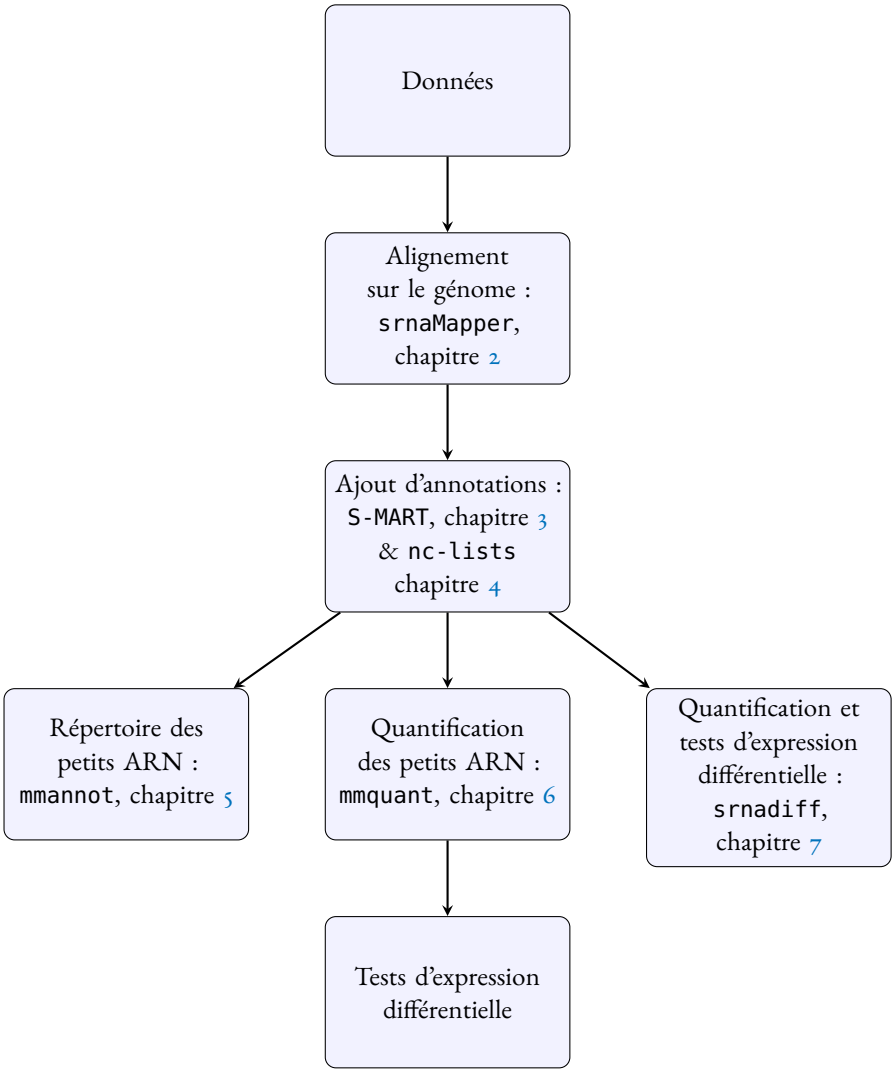



FIGURE 1.1 – *Workflows* possibles de l'analyse de petits ARN. Les contributions présentées ici sont présentées par les noms des logiciels.

# 2

# SRNAMAPPER

## 2.1 INTRODUCTION

Après l'étape de séquençage vient en général l'alignement, qui peut se définir comme la prédiction du locus ayant produit l'ARN. Or, nous l'avons mentionné précédemment, les *loci* produisant des petits ARN sont souvent dupliqués. De plus, certains de ces petits ARNs subissent une étape d'édition, qui supprime, mais parfois ajoute ou modifie les nucléotides situés aux extrémités des lectures.

Un bon outil d'alignement, spécifique pour les petits ARN, doit donc être capable de retrouver tous les *loci*, même lorsque les transcrits ont été édités.

Beaucoup d'outils d'alignement ont été conçus, mais aucun n'est à ce jour spécialement adapté aux petits ARN. On utilise donc des outils créés pour l'alignement classique, comme bowtie (Langmead, Trapnell et al., 2009), bowtie2 (Langmead et Salzberg, 2012), ou bwa (H. Li et Durbin, 2009), avec des paramètres particuliers. Ces outils sont tous incomplets, dans le sens où ils utilisent des heuristiques qui accélèrent la recherche. bowtie fait une recherche locale avec backtrack, qui n'est pas exhaustive (120 backtracks maximum par défaut). bowtie2 fait une recherche exacte de facteurs (par défaut de taille 16), qui rend impossible la recherche exhaustive. bwa impose un nombre maximum d'erreurs dans les premiers nucléotides (32 par défaut). En conséquence, aucun des outils actuels ne peut retrouver toutes lectures, surtout si elles contiennent quelques erreurs, et qu'elles se trouvent dans des régions répétées.

Nous présentons ici un nouvel outil, srnaMapper, spécialement conçu pour les petits ARN, dont une première ébauche a été conçue avec l'aide de mon premier doctorant, Walid Ben Saoud Benjerri. Il implémente un algorithme original, basé sur la comparaison d'arbres approchée, afin d'effectuer une recherche exhaustive. Il permet de retrouver tous les *loci* distants de moins de $k$ mutations, où $k$ est donné.

## 2.2 MÉTHODE

### 2.2.1 *Description générale*

Les lectures sont tout d'abord stockées dans un arbre, où chaque branche stocke un ou plusieurs nucléotides. Un nœud terminal indique la fin d'une lecture. L'ensemble des lectures du fichier d'entrée est donc retrouvable en parcourant l'arbre de la racine aux nœuds terminaux. À chaque nœud terminal, on ajoute plusieurs informations : le nombre de fois que l'on trouve la lecture en question, la qualité, et le nom des lectures.

Le génome est stocké dans un tableau des suffixes (nous avons utilisé l'implémentation de bwa). Étant donné qu'un tableau des suffixes s'utilise en pratique de la même manière qu'un arbre des suffixes (si ce n'est qu'il est lu de droite à gauche, c'est pourquoi nous retournons les lectures), nous considérerons, pour la simplicité de l'exposition, que nous utiliserons un arbre des suffixes (et que les lectures ne sont pas inversées).

Étant donnés un seuil $k$, un arbre des lectures, et un arbre du génome, notre but sera ici de trouver, pour chaque nœud terminal des lectures, l'ensemble des *loci* du génome, dont la distance d'édition est minimale, et d'au plus $k$. Lorsque $k = 0$, le problème se réduit à trouver le plus grand sous-arbre des lectures compris dans l'arbre du génome. Si $k > 1$, le problème peut être décrit comme une recherche de sous-arbre approchée. C'est, à notre connaissance, la première fois que cette question a été formulée.

```
@read1
A
+
H
@read2
CG
+
HI
@read3
CG
+
IH
@read4
CGA
+
HHI
@read5
CGC
+
IIH
@read6
CT
+
II
```

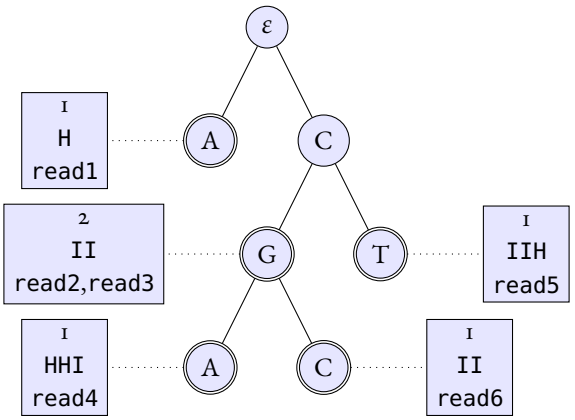



FIGURE 2.1 – Un exemple d'un fichier de lectures au format fastq, avec 6 lectures (à gauche), et sa représentation en arbre (à droite). Par exemple, la lecture `CGA`, nommée `read4`, est vue une fois, avec une qualité de `HHI`. L'information attachée au nœud terminal `CGA` est donc `1, HHI, read4`. La lecture `GC` est vue deux fois (dans `read2` et `read3`), avec pour qualité `HI` et `IH`. Le maximum, base par base, de la qualité est stocké (`II`), car `I`, qui est une qualité de 40 est supérieur à `H`, qui est une qualité de 39.

Dans notre implémentation, nous parcourons récursivement l'arbre des lectures, en trouvant, pour chaque nœud des lectures, l'ensemble des nœuds correspondants (c'est-à-dire dont la distance est minimale, et à distance de $k$ ou moins) dans l'arbre du génome. Plusieurs optimisations ont été ajoutées, afin d'accélérer la recherche.

### 2.2.2 *L'arbre des lectures*

Les lectures sont stockées dans un arbre compressé (cf. Fig. 2.1). Dans chaque nœud terminal, on stocke le nombre d'occurrences de chaque séquence, la qualité, et le nom des lectures. Afin d'économiser de la place pour la qualité des lectures, nous ne stockons que la qualité maximale (base par base).

Lorsque l'on lit plusieurs fichiers, les lectures sont stockées dans le même arbre, et le nombre d'occurrences dans chaque fichier est stocké.

Dans notre implémentation, les lectures sont tout d'abord stockées dans un arbre non-compressé, où chaque nœud exactement quatre fils, et chaque branche contient au plus un nucléotide. Une fois cet arbre construit, la structure est transférée dans un arbre optimisé, où les nœuds frères sont stockés dans des adresses mémoires contiguës, ce qui permet d'accélérer l'accès aux données.

### 2.2.3 *L'alignement*

L'alignement consiste ici à trouver, pour chaque nœud des lectures, et pour chaque $i \in [0,k]$, l'ensemble des nœuds correspondants dans l'arbre des génomes.

Plus formellement, on appelle :

— un arbre $A$ un ensemble de nœuds, dont un nœud spécial, sa racine, noté $\varepsilon_A$ ;
— $L$ l'arbre des lectures ;
— $G$ l'arbre du génome ;
— $f(n), n \in A$, les fils d'un nœud $n$, qui est un ensemble de nœuds de $A$, étiqueté par des listes de nucléotides ;
— $e(n')$, l'étiquette du fils $n' \in f(n)$ (informellement, l'étiquette est la liste de nucléotides qui permet de passer du nœud père au nœud fils) ;
— $c(n), n \in A$, le chemin d'un nœud, c'est-à-dire la concaténation des étiquettes allant de $\varepsilon_A$ à $n$ (on suppose que, comme dans tout arbre, il existe un chemin allant de la racine à chacun des nœuds) ;
— $p(n)$, la profondeur d'un nœud, soit la taille $|c(n)|$ ;
— la distance entre $n_L \in L$ et $n_G \in G$, notée $d(n_L, n_G)$, la distance d'édition entre $c(n_L)$ et $c(n_G)$ ;
— la table de correspondance, pour $n_L \in L$, et $i \in [0,k]$, $t(n_L, i) = \{n_G \in G, d(n_L, n_G) = i\}$.

Notre problème revient donc à remplir cette table de correspondance. Le cas de base est le suivant : les nœuds correspondants à la racine de l'arbre des lectures, pour $i$ erreurs données, est l'ensemble des nœuds de profondeur $i$, qui ont donc $i$ suppressions. Plus formellement : $t(\varepsilon_L, i) = \{n_G : |n_G| = i\}$.

On peut donc maintenant supposer que l'on connaît $t(n_L, i), \forall i \in [0,k]$. On va donc calculer $t(n'_L, i), \forall i \in [0,k]$, où $n'_L \in f(n_L)$. On va supposer également que la taille de l'étiquette $e(n'_L)$ est de 1. Le cas général peut s'étendre facilement à partir de ce cas particulier.

Comme d'habitude, on discrimine plusieurs cas : l'appariement, le mésappariement, l'insertion, et la suppression. Pour l'appariement, nous avons : $t(n'_L, i) = \bigcup_{n_G \in t(n_L, i)} \{n'_G \in f(n_G) | e(n'_G) = e(n'_L)$. Pour le mésappariement, nous avons : $t(n'_L, i+1) = \bigcup_{n_G \in t(n_L, i)} \{n'_G \in f(n_G) | e(n'_G) \neq e(n'_L)$.

Pour l'insertion (c'est-à-dire que le nucléotide qui étiquette le fils du nœud de l'arbre des lectures n'est pas aligné), nous avons : $t(n'_L, i+1) = t(n'_L, i+1) \cup t(n_L, i)$. Le cas de la suppression ajoute : $t(n'_L, i+1) = t(n'_L, i+1) \bigcup_{n_G \in t(n'_L, i)} f(n_G)$.

Il est à noter que le cas de la suppression complète $t(n'_L, i+1)$ en utilisant l'information collectée pour $t(n'_L, i)$. Il est donc essentiel de compléter $t(n'_L, .)$ par ordre croissant de $i$.

### 2.2.4 *Optimisations*

*Approche optimiste*

Dans notre implémentation, nous ne calculons pas *a priori* $t(., i)$ pour tous les $i \leq k$. On essaie tout d'abord d'aligner sans erreur (c'est-à-dire avec $i = 0$). Si, au nœud $n'_L$, on ne peut pas trouve de nœud dans le génome correspondant, ($t(n_L, 0) = \varnothing$), alors on calcule $t(n_L, 1)$. Ceci exige éventuellement de calculer $t(n'_L, 1)$, où $n'_L$ est le père de $n_L$, si cela n'a pas été fait. On peut donc remonter récursivement jusqu'à la racine, où $t(\varepsilon_L, i)$ a été calculé, pour tout $i$.

Il est à noter que lorsque $t(n_L, i)$ a été calculé, il peut servir à tous les nœuds descendants $n'_L$, si l'on veut calculer $t(n'_L, i)$.

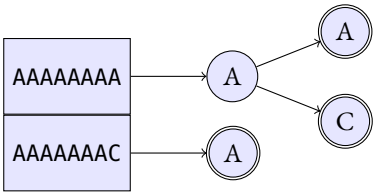



FIGURE 2.2 – Un exemple de liste d'arbres stockant les lectures AAAAAAAAAA, AAAAAAAAAC et AAAAAAAACA.

*Stockage des lectures dans une liste d'arbres*

Nous avons observé que, jusqu'à 8 nucléotides, l'arbre était presque complet, c'est-à-dire que chaque nœud a exactement quatre fils. Dans ce cas, un arbre n'est pas le plus adapté, et il est possible de stocker de façon plus concise l'information. Nous utilisons donc un vecteur de $4^8$, où le premier arbre stocke toutes les séquences commençant par AAAAAAAA, le deuxième arbre stocke toutes les séquences commençant par AAAAAAAC, etc. (cf Fig. 2.2).

*Suppression des lectures de basse complexité*

Nous avons observé qu'il existait, dans la majorité des séquençages, beaucoup de séquences de basse complexité, de type AAAAAA, ou ATATATA. Ces séquences, difficiles à interpréter biologiquement, prennent du temps à être alignées car elles peuvent, en général, être assignées à beaucoup de *loci* sur le génome. En conséquence, l'utilisateur supprime en général ces lectures après ou avant l'alignement. Nous avons ajouté une fonctionnalité permettant de supprimer ces lectures à la volée. Avant l'alignement, l'arbre des lectures est parcouru, et nous comptons, pour chaque nœud $n_L$, le nombre de fois que l'on lit chaque triplet dans $c(n_L)$. Si un triplet est vu plus de $n$ fois, $n$ étant donné par l'utilisateur, on estime que toutes les lectures partant de ce nœud sont de basse complexité, et peuvent être retirées. Ceci revient à supprimer la branche.

## 2.3 RÉSULTATS

Nous avons comparé notre méthode avec plusieurs outils largement utilisés pour aligner les séquences de sRNA-Seq : bowtie (LANGMEAD, TRAPNELL et al., 2009), bowtie2 (LANGMEAD et SALZBERG, 2012), et bwa (H. LI et DURBIN, 2009), en utilisant les paramètres préconisé dans un article comparant les outils d'alignement de lectures pour le sRNA-Seq (ZIEMANN, KASPI et EL-OSTA, 2016). Nous avons utilisé les outils sur deux organismes différents : l'humain, et *A. thaliana*.

Étant donné que les outils précédents utilisent des heuristiques, et donc oublient des *hits*, nous nous sommes également comparés à des outils de recherche exhaustifs, comme Yara (SIRAGUSA, WEESE et REINERT, 2013). Nous avons également essayé d'autres outils (FEM (H. ZHANG et al., 2018), Hobbes (AHMADI et al., 2011), et BitMapper2 (CHENG, Y. ZHANG et XU, 2019)), qui sont basés sur du filtrage par $q$-gram. Ceux-ci ne fonctionnent pas car les lectures de 21 nucléotides avec 2 erreurs doivent être coupés en $q$-grams trop petits pour être utilisables.

Les résultats montrent que srnaMapper aligne presque toutes lectures que les autres outils alignent (Fig. 2.3). Les rares exceptions sont les lectures de basse complexité. En revanche, srnaMapper aligne beaucoup de lectures que les autres n'arrivent pas à aligner. Parmi les lectures qui sont également alignées par les autres outils, srnaMapper arrive à en aligner avec moins d'erreurs. Enfin parmi les lectures qui sont alignées avec autant d'erreurs, srnaMapper arrive à en aligner à plus de *loci*.

Concernant le temps, srnaMapper est plus lent que les autres outils qui alignent moins de lectures, mais prend un temps comparable à ceux qui alignent presque autant (Fig. 2.4).

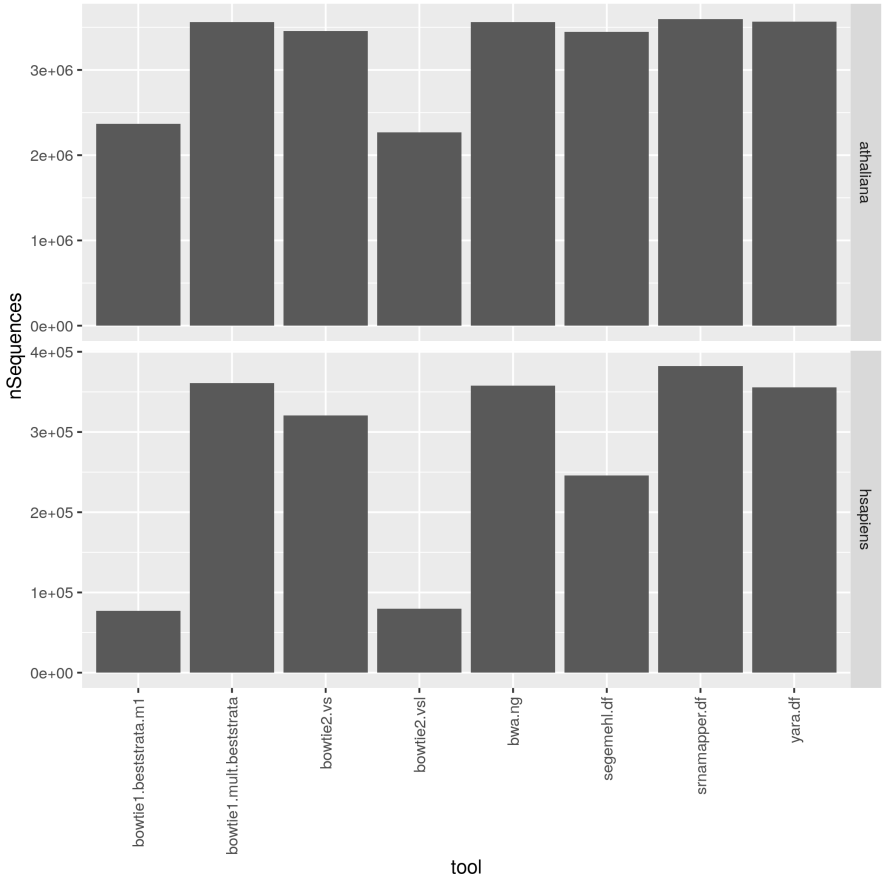



FIGURE 2.3 – Nombre de lectures alignées par divers outils, en utilisant plusieurs paramétrages possibles.

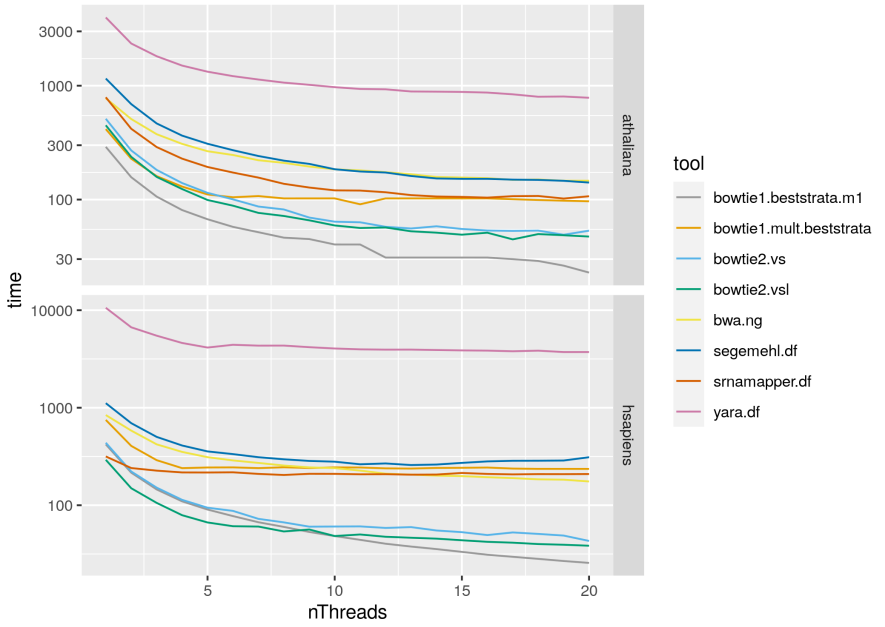



FIGURE 2.4 – Temps (en secondes) pour aligner les lectures. Les ordonnées sont en échelle log.

# 3 S-MART

Cet partie vient de l'article (Zytnicki et Quesneville, 2011).

## 3.1 INTRODUCTION

Les données de sRNA-Seq peuvent souvent être utilisées afin d'enrichir l'annotation des petits ARN. En pratique, les données ne co-localisant pas avec les annotations connues sont regroupées en unités transcriptionnelles, et classées en fonction de différentes caractéristiques, telles que la taille des lectures, la position des lectures entre elles, etc. Les pipe-lines sont souvent *ad hoc*, et nécessitent des ajustements en fonction de l'espèce, ou de la classe de petits ARN. Pour fixer les idées, nous présentons ici un pipe-line basé sur une étude présentée par Brennecke et al., 2007, qui se propose de trouver les piARN à partir de données de CLIP-Seq. Le CLIP-Seq est le séquençage des ARN interagissant avec une protéine d'intérêt.

On suppose ici que l'on a des données CLIP-Seq de Piwi, où l'ARN ciblé par Piwi a été immuno-precipité, l'ADN complémentaire recréé puis séquencé. Les lectures ont été alignées sur le génome.

Les étapes possibles d'analyse possibles sont :

— Sélectionner les lectures de 28 à 30 nucléotides, qui sont la taille attendue des piARN.
— Fusionner les intervalles proches en unités transcriptionnelles.
— Filtrer ces unités, où seules celles qui ont au moins 10 lectures sont gardées.

Étant donné que les lectures proviennent de régions répétées, il est possible que toutes les lectures d'une unité transcriptionnelle puissent s'aligner à plusieurs endroits. Ceci signifie que l'on ne peut pas être certain de l'existence de cette unité à la position prédite. Il faudrait donc qu'il y ait au moins une lecture s'alignant uniquement sur le génome par unité transcriptionnelle pour la conserver. Voici les étapes pour ce faire :

— Sélectionner les lectures alignées exactement une fois.
— Sélectionner les unités transcriptionnelles qui chevauchent au moins une des lectures.

Afin de partager ces unités transcriptionnelles validées, on peut :

— Produire la courbe de densité de ces unités sur le génome.
— Convertir ces unités transcriptionnelles en format CSV, compatible avec Excel.

Les piARN ayant majoritairement un U en première position, et un A en dixième position, il est également possible de fournir des figures vérifiant ces faits sur les données.

— Sélection des lectures qui chevauchent les unités transcriptionnelles validées.
— Produire la courbe de proportion des nucléotides, le long des lectures précédemment trouvées.

Chacune de ces étapes demande l'utilisation d'un outil particulier, ou de scripts dédiés, qui passent parfois difficilement à l'échelle. Nous avons fait en sorte qu'elles soient toutes réalisables dans S-MART. Cette boîte à outils inclut un ensemble de briques de bases qui permettent l'analyse modulaire des petits ARN, et des données de séquençage en général. S-MART permet :

— de convertir des formats de fichier : BED, GFF, GTF, SAM, BAM, WIG et CSV ;
— de modifier les données d'un fichier ;
— de comparer une paire de fichiers ;
— de visualiser des données.

Un effort important a été fait pour rendre l'outil simple d'utilisation, utilisable quel que soit le système d'exploitation. Nous avons également adapté plusieurs outils à l'interface Galaxy.

## 3.2 LES OUTILS

### 3.2.1 *Données analysées par les outils*

S-MART travaille sur plusieurs types de données différents :

— les intervalles génomiques : les transcrits ou les gènes sont stockés comme des intervalles, qui contiennent eux-mêmes des intervalles ;
— les positions génomiques : en général donné au format WIG, il s'agit d'un nombre donné pour chaque position de chaque chromosome, qui peut être une couverture en nombre de lectures, ou un score de conservation ;
— les séquences (comme des lectures, ou des génomes).

Les intervalles génomiques sont stockés en mémoire en utilisant une représentation structurée uniforme, contenant éventuellement des méta-données. À ce titre, le format préféré de S-MART pour représenter les intervalles est le GFF ou le GTF, qui peut écrire ces méta-données dans le dernier champ.

Chaque type de données peut être indexé. Pour les positions génomiques, nous stockons à part un fichier contenant l'adresse mémoire des informations relatives à la position 1, 10kb, 20kb, etc. Pour les séquences, nous utilisons le fait que les fichiers sont structurés avec exactement 60 nucléotides par ligne (le nombre exact de nucléotides par ligne est en fait déterminé à la volée). Seules les adresses mémoires correspondant à chaque début de chromosomes sont stockés, et la position exacte du nucléotide demandé est déterminé dynamiquement. La comparaison d'intervalles est en fait la question la plus complexe informatiquement. Nous avons donc utilisé et adapté une structure de donnée originale, les NC-list, qui seront décrites dans le chapitre 4.

### 3.2.2 *Présentation des outils*

La plupart des outils s'applique directement au sRNA-Seq, mais également à d'autres types de données.

*Sélection et modification des données*

— Sélection (sur différent critères, comme le *locus*, la taille, le nombre d'erreurs, le nombre de *hits*, etc.) et statistiques sur des lectures alignées,
— sélection et modification de données de séquences (notamment la suppression d'adaptateurs),
— sélection et modification d'intervalles génomiques,
— comparaison de deux jeux d'intervalles génomiques, suivi d'opérations telles que la sélection des intervalles chevauchants, mais également l'intersection, l'union, ou la différence,
— analyse d'intervalles génomiques par fenêtres glissantes.

*Visualisation*

— Redondance des séquences (nombre de fois qu'un transcrit est séquencé),
— distribution de taille, d'utilisation des nucléotides, des positions sur le génome,
— distribution d'une information génomique (comme la conservation, la présence d'une histone), moyennée sur l'ensemble des gènes d'intérêt.

Beaucoup des outils proposés ici ont été développés indépendamment (et après) dans d'autres connues, telles les bedtools (Quinlan, 2014) ou les deeptools (Ramírez et al., 2014).

## 3.3 CONCLUSION

S-MART est une boîte à outils complète d'analyse de données de séquençage, particulièrement adaptée aux petits ARN. Il s'agit d'une boîte à outils que j'ai longuement enrichie au cours des années, et qui m'a servi à répondre à de nombreuses questions biologiques, comme en témoignent les publications que j'ai faites avec cet outil : Charbonnel, A. K. Niazi et al., 2017 ; Elvira-Matelot et al., 2016 ; Geering et al., 2014 ; Molitor, Latrasse, Zytnicki, P. Andrey et al., 2016 ; Parent, V. Jauvion et al., 2015 ; Shamandi, Zytnicki, Charbonnel, Elvira-Matelot, Bochnakian, Comella, A. C. Mallory et al., 2015 ; Toffano-Nioche et al., 2013 ; Zanni et al., 2013.

# 4 LES NC-LIST

Cette partie s'appuie sur l'article Zytnicki, Luo et Quesneville, 2013.

## 4.1 INTRODUCTION

Après l'alignement, la deuxième étape du sRNA-Seq (comme de beaucoup de données de séquençage), est en général la prédiction du transcrit qui a produit la lecture. Pour ce faire, on compare les intervalles génomiques des transcrits connus et d'une lecture. Si les intervalles coïncident, on peut attribuer la lecture à ce transcrit. D'une manière générale, la comparaison d'intervalles génomiques est l'opération la plus importante lorsque l'on compare deux jeux de données génomiques. Il sert par exemple à comparer des sites candidats de facteurs de transcriptions avec des gènes sur-exprimés, à trouver les polymorphismes situés dans des régions codantes, ou caractériser des nouveaux transcrits prédits par rapport aux transcrits connus.

Dans tous les cas présentés, il faut comparer des milliers, voire des centaines de millions d'intervalles, contre des milliers d'autres intervalles de référence. Il faut donc trouver des algorithmes optimisés, qui eux-mêmes utilisent des structures de données adaptées. Ces structures de données sont en général calculées sur les données de référence, car elles sont susceptibles d'être comparées plusieurs fois. Il peut s'agir d'un index sur des données triées (comme pour les fichiers BAM), un R-tree, ou des *nested containment lists* (NC-list, Alekseyenko et Lee, 2007). Même si le temps de création de ces structures est long, il est en général avantageux de les utiliser dès lors que l'on opère un certain nombre de comparaisons.

En général, les algorithmes existants cherchent à savoir quels sont les intervalles de référence qui chevauchent un seul intervalle donné, et ne proposent pas spécifiquement de méthode pour retrouver les chevauchements de plusieurs intervalles requête. Pour autant, l'information récoltée par une itération peut être réutilisée pour une autre itération, dès que les intervalles requête sont correctement structurés. L'unique méthode faite pour de retrouver l'ensemble des chevauchements semble être l'algorithme fjoin (Richardson, 2006).

Nous utilisons ici une NC-list, qui est une structure compacte permettant de stocker les intervalles génomiques. L'intérêt de la NC-list est le suivant. Si l'on souhaite trouver tous les intervalles de référence d'un intervalle requête, il est désirable de procéder par dichotomie sur l'ensemble des intervalles de référence. C'est en fait *a priori* impossible, car les intervalles peuvent être emboîtés les uns dans les autres, ce qui fait qu'il n'existe pas d'ordre total sur les intervalles permettant d'utiliser la dichotomie. Pour résoudre ce problème, le vecteur $L$ contient des listes d'intervalles non-emboîtés, que l'on peut requêter par dichotomie. Les intervalles emboîtés sont stockés dans une autre sous-liste, et le lien est stocké dans le vecteur $H$. Les NC-list peuvent être construites en temps quasi-linéaire (c'est-à-dire en $O(n \log(n))$) et la complexité spatiale est linéaire. Les concepteurs des NC-list présentent également un algorithme cherchant tous les intervalles de référence chevauchant un intervalle donné. Nous montrerons que la complexité temporelle de cet algorithme n'est pas exactement celle donnée, à savoir $O(\log(|R|) + |C|)$, où $R$ est l'ensemble des intervalles de référence, et $C$ les paires d'intervalles chevauchants.

Pour construire de façon efficace une NC-list (cf. figure 4.1), il faut avant tout que les intervalles aient été triés par position de début croissante, et en cas d'égalité, par position de fin décroissante. Comme les intervalles 3, 4, et 5 sont emboîtés dans l'intervalle 2, ils sont supprimés de la première liste, composée des intervalles 1, 2, et 6. Les intervalles 3 et

5 sont sont envoyés dans la sous-liste de 2. De la même manière, l'intervalle 4 est emboîté dans l'intervalle 3, et est donc envoyé dans une autre sous-liste. Lorsqu'un intervalle est emboîté dans deux intervalles (comme c'est le cas pour l'intervalle 7, qui est emboîté dans les intervalles 2 et 6), celui situé le plus à droite est choisi. Dans ce cas, c'est l'intervalle 6.

Une NC-list est composée des vecteurs $L$ et $H$. Chaque ligne de $L$ contient les positions de début et de fin d'intervalle, ainsi qu'une position sur une cellule du vecteur $H$. Les éléments de chaque liste apparaissent de façon contiguë dans $L$. Une sous-liste est représentée par une ligne de $H$, qui contient la position du premier intervalle dans $L$, et le nombre d'éléments de la sous-liste. La sous-liste de l'intervalle 2 a été grisée dans l'exemple. L'intervalle 2 est stocké dans la ligne 1 du vecteur $L$ (si l'on suppose que le premier indice est 0). Sa sous-liste est la ligne 1 du vecteur $H$. Elle commence à la position 3 du vecteur $L$, et contient 2 intervalles (3, et 5).

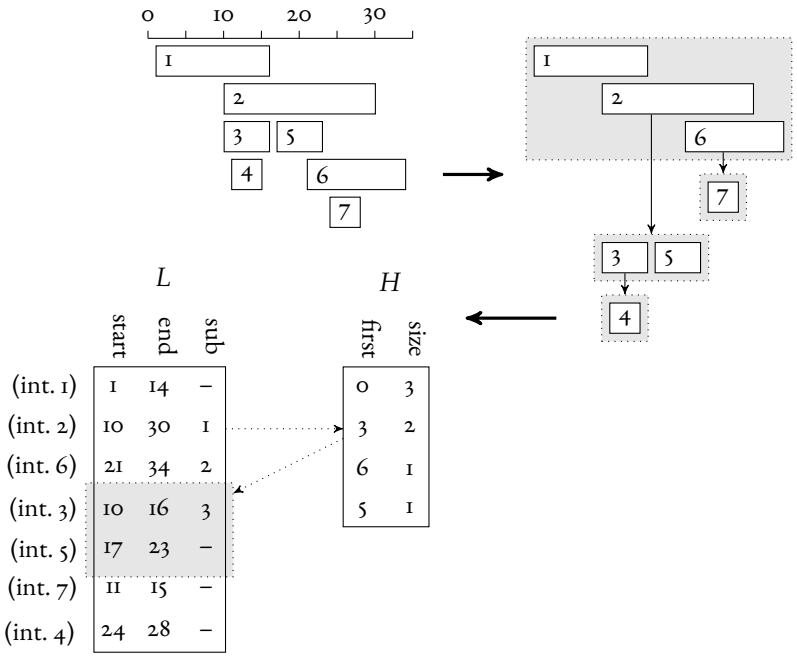



FIGURE 4.1 – Construction d'une NC-list.

Notre objectif est ici de produire un algorithme permettant de trouver tous les couples intervalles requête/intervalles de référence qui chevauchent. On supposera que les intervalles de référence et de requête ont été précédemment triés. Nous pourrons ensuite créer une NC-list sur les intervalles de référence. La phase de comparaison donne tous les couples d'intervalles chevauchants, en temps $O(|R| + |Q| + |C|)$, où $|R|$ est l'ensemble d'intervalles de référence, $|Q|$ est l'ensemble d'intervalles requête, et $|C|$ l'ensemble de couples chevauchants. Cette complexité temporelle est bien meilleure que celles des algorithmes ayant une complexité spatiale constante présentés jusqu'alors. En incluant la préparation des données, la complexité temporelle est de $O(|R|\log(|R|) + |Q|\log(|Q|) + |C|)$. Cette méthode est donc particulièrement adaptée lorsque l'on réalise plusieurs comparaisons sur de larges jeux de données.

## 4.2 MÉTHODE

### 4.2.1 *Algorithme originel*

L'article de ALEKSEYENKO et LEE, 2007 décrit un algorithme retrouvant tous les intervalles de référence, stockés dans une NC-list, chevauchant avec un seul intervalle requête.

*Définitions*

Dans ce cadre, on supposera que les brins n'ont pas d'importance, et on se permet de concaténer tous les chromosomes. On se ramène donc au cas le plus simple, où un intervalle est une simple paire $(a, b)$, où $a \leq b$. On dit que :

— l'intervalle $i$ est avant $j$ ($i < j$) ssi $i.b < j.a$ ;
— les intervalles $i$ et $j$ se chevauchent ($i <> j$) ssi $(i.a \leq j.b) \wedge (j.a \leq i.b)$ ;
— l'intervalle $i$ est inclus dans $j$ ($i \subset j$) ssi $(j.a \leq i.a) \wedge (i.b \leq j.b)$.

On se munit d'une relation d'ordre sur les intervalles : $i \prec j$ ssi $i.a < j.a$ ou $i.a = j.a \wedge i.b > j.b$.

Notons que deux intervalles identiques peuvent exister dans une liste. Dans ce cas, nous décidons que si deux intervalles sont $i$ et $j$ identiques, alors $i \prec j$ ou $j \prec i$, arbitrairement.

Une NC-list respecte les assertions suivantes. Les intervalles n'étant inclus dans aucun autre sont stockés dans la première liste. Chaque intervalle $i$ contient une sous-liste (éventuellement vide), des éléments qui lui sont inclus. Si la sous-liste n'est pas vide, on ajoute un lien entre l'intervalle père et la sous-liste dans dans $H$. Les intervalles d'une même liste sont stockés de façon contiguë dans $L$. Dans chaque liste, les intervalles sont triés par $\prec$. La première liste apparaît en premier. L'ordre respectif des autres listes les unes par rapport aux autres n'a pas d'importance.

### 4.2.2 *Complexité de l'algorithme originel*

L'algorithme 1 présente l'algorithme, tel que présenté dans ibid. Brièvement, l'algorithme cherche par dichotomie dans une sous-liste courante le premier élément qui chevauche l'intervalle-requête $i$. Lorsqu'il est trouvé, on procède récursivement dans les sous-intervalles, et on passe au prochain intervalle, s'il chevauche encore $i$.

**Algorithme 1 :** Algorithme originel

```
// i: intervalle requête
// h: indice dans H de la sous-liste courante
// M: ensemble des intervalles de référence chevauchants
Function Recherche NC(i, h, M)
    l ← Recherche premier(i, H[h])      // Recherche dichotomique
    while (l ≤ H[h].first + H[h].size) ∧ (L[l] <> i) do
        M ← M ∪ {L[l]} ∪ Recherche NC(i, L[l].sub)
        l ← l + 1
    return M
```

La fonction *Recherche premier*$(i, H[h])$ recherche de façon dichotomique le premier élément de la sous-liste $H[h]$ chevauchant $i$. Pour rechercher les intervalles chevauchant à un intervalle requête $i$, on doit appeler la fonction *Recherche NC*$(i, 0, \varnothing)$.

Dans l'article, les auteurs indiquent que l'algorithme a une complexité de $O(\log(|R|) + |C|)$, où $R$ est l'ensemble des intervalles requêtes, et $C$, le nombre de chevauchements. L'exemple de la figure 4.2 montre que ce n'est pas toujours exact.

On suppose que chaque liste contient $n$ intervalles. Tous les intervalles n'ont pas de fils, sauf le dernier de chaque sous-liste (excepté le tout dernier intervalle). Il y a $n$ sous-listes,

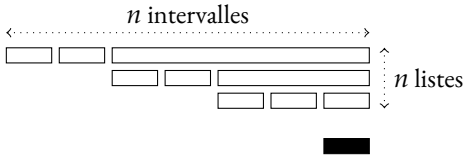



FIGURE 4.2 – Contre-exemple montrant l'erreur dans le calcul de la complexité de l'algorithme originel. L'intervalle noir est l'intervalle-requête, et les intervalles de référence sont placés dans une structure récursive.

nous avons donc en tout $n \times n = n^2$ intervalles. L'intervalle-requête chevauche le dernier intervalle de chaque sous-liste, soit $n$ intervalles.

Si l'on s'en réfère à l'algorithme 1 (équivalent à l'algorithme donné dans la publication), l'algorithme effectue une recherche dichotomique dans chaque sous-liste, pour un temps de $O(n \times \log(n))$. Ceci n'est pas conforme à la complexité annoncée, qui est de $O(\log(n^2) + n) = O(n)$.

### 4.2.3 *Nouvel algorithme*

Le but est ici de trouver tous les couples intervalles requête/intervalles de référence qui chevauchent. On lit donc tous les intervalles requête triés, et on les compare itérativement aux intervalles de référence. Chaque itération retourne à la fois les intervalles chevauchants, mais surtout le premier intervalle qui a chevauché. L'idée générale est décrite dans l'algorithme 2. Pour faciliter la lecture, quelques détails techniques ont été omis (tel quel, l'algorithme n'est pas linéaire dans quelques cas pathologiques).

**Algorithme 2 :** Algorithme utilisant l'information récoltée précédemment

```
// i: intervalle à comparer
// l: dernier intervalle de référence qui a chevauché le
   précédent intervalle requête
// M: ensemble des intervalles chevauchants
l' ← ∅
Function Recherche NC 2(i, l, M)
    if i <> L[l] then
        if l' == ∅ then l' ← l
        M ← M ∪ {L[l]} ∪ Recherche NC 2(i, H[L[l].sub].first, M)
        l ← succ(l)
    else if i < L[l] then
        l ← succ(l)
    else
        return (M, l')
```

La fonction *succ* prend l'élément suivant dans la liste. S'il s'agissait du dernier élément, on remonte à la sous-liste parent, et on prend le prochain élément (récursivement).

### 4.2.4 *Modélisation des transcrits*

Il est à noter que les transcrits, qui sont en général comparés aux lectures, ne sont en pratique pas des intervalles. Un transcrit est le plus souvent constitué d'exons qui, eux, sont des intervalles. *A priori*, les intervalles situés entre les exons, les introns, ne donnent pas lieu à des transcrits stables, et l'utilisateur, en général préfère annoter les lectures dans les introns d'un gène comme étant « autre chose » qu'un gène (un ARN anti-sens, par exemple).

Dans notre implémentation, nous avons stocké uniquement les gènes, modélisés comme des intervalles démarrant au premier nucléotide du premier exon, et finissant au dernier

nucléotide du dernier exon. Si une lecture chevauche un gène, elle est ensuite comparée à tous les exons de tous les transcrits.

## 4.3 RÉSULTATS

### 4.3.1 *Comparaison avec les autres méthodes*

Nous avons comparé notre travail avec plusieurs autres méthodes déjà publiées :

— l'algorithme originel sur les NC-list (ALEKSEYENKO et LEE, 2007), nommé « nc » par la suite, utilisé itérativement ;
— une méthode de binning (KENT et al., 2002) utilisant une table SQLite indexée, appelée « bin » ;
— une autre méthode de binning utilisant une table de hashage où les bins sont clefs, et les valeurs des listes d'intervalles, appelée « has » ;
— une dernière méthode de binning utilisant des *segment trees*, décrits dans RENAUD et al., 2011, appelée « seg » ;
— la méthode *fjoin* proposée par RICHARDSON, 2006, appelée « fj » ;
— notre méthode, nommée « new » ;

Parmi les algorithme précédents, seuls « bin », « nc », et « new » ont des complexités spatiales constantes. Dans « has », la table de hashage est stockée en mémoire, et dans « seg », les *segment trees* le sont également. De même, « fj », a une complexité spatiale linéaire, et ne fonctionne pas sur les gros jeux de données. Dans notre implémentation, « has » prend 4Go de RAM (mémoire disponible pour l'ordinateur effectuant les tests) lorsque le jeu de données contient trente millions d'intervalles.

### 4.3.2 *Exemple sur un jeu réel*

Les lectures ont été alignées avec bowtie (LANGMEAD, TRAPNELL et al., 2009) sur trois génomes de référence, et nous avons comparé les lectures alignées avec l'annotation. Nous avons utilisé les six méthodes précédemment décrites. Les temps sont données dans la table 4.1.

| Organisme | bin | has | seg | fj | nc | new |
|---|---|---|---|---|---|---|
| *S. cerevisiae* | 5.1 | 3.2 | 4.3 | * | 4.8 | 3.4 |
| *D. melanogaster* | 2.5 | 1.3 | 1.9 | 1.1 | 2.1 | 1.4 |
| *A. thaliana* | 17 | 9.2 | 13 | * | 14 | 9.1 |

TABLE 4.1 – Temps passé à effectuer la comparaison en utilisant six méthodes différentes. * : Le programme a été arrêté car il demandait plus de 4Go de RAM.

Comme attendu, les méthodes « has » et « fj » sont particulièrement adaptées à ces jeux de données car les intervalles peuvent être intégralement stockés en mémoire. Notre méthode fait également partie des plus rapides.

En revanche, l'étape de *pre-processing* des données est la plus lente pour notre méthode (voir table 4.2). C'est une balance classique entre temps de *pre-processing* et d'analyse.

| Organisme | bin | has | seg | fj | nc | new |
|---|---|---|---|---|---|---|
| *S. cerevisiae* | 2 | 1 | 1 | * | $2.10^3$ | $7.10^3$ |
| *D. melanogaster* | 44 | 29 | 23 | 33 | $1.10^3$ | $7.10^3$ |
| *A. thaliana* | 68 | 44 | 50 | * | $7.10^3$ | $2.10^4$ |

TABLE 4.2 – Temps de *preprocessing* en secondes. * : Le programme a été arrêté car il demandait plus de 4Go de RAM.

| Organisme | bin | has | seg | fj | nc | new |
|---|---|---|---|---|---|---|
| *S. cerevisiae* | 12 | 8 | 8 | * | 32 | 376 |
| *D. melanogaster* | 12 | 40 | 12 | $4.10^4$ | 292 | 236 |
| *A. thaliana* | 12 | 56 | 12 | * | 236 | 176 |

TABLE 4.3 – Consommation en RAM, en ko. * : Le programme a été arrêté car il demandait plus de 4Go de RAM.

### 4.3.3 *Données simulées*

Nous avons également généré des données simulées pour comparer ces méthodes dans le détail. Les intervalles vont de 36 à 100 nucléotides. Il y a entre 100 et 100 000 intervalles de référence, et entre 100 et 100 millions d'intervalles requête. Le génome ne contient qu'un seul chromosome, allant de 100 nucléotides à 200 millions. Chaque configuration a été générée cinq fois. Les temps passés pour construire les structures de données et résoudre les instances sont donnés dans la figure 4.3.

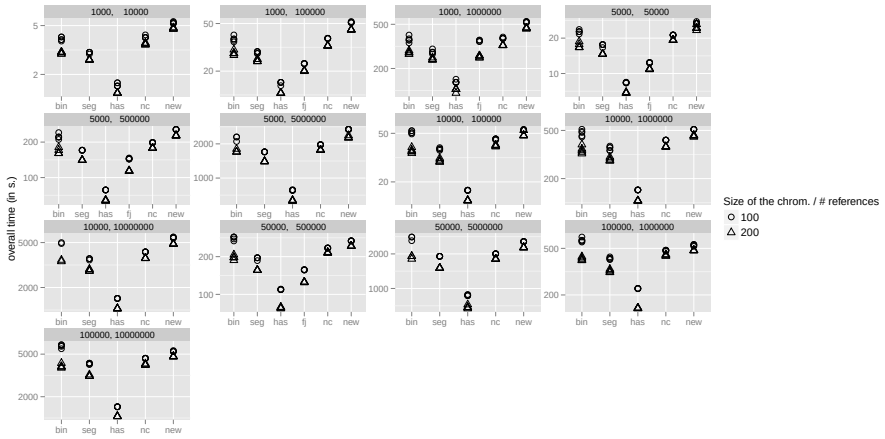



FIGURE 4.3 – Temps passé à construire les structures de données et résoudre des instances simulées.

Notre méthode est la plus rapide, parmi celles qui ont une complexité spatiale constante. La méthode « fj » demandait trop de mémoire pour les instances trop grandes.

La mémoire utilisée est donné dans la figure 4.4.

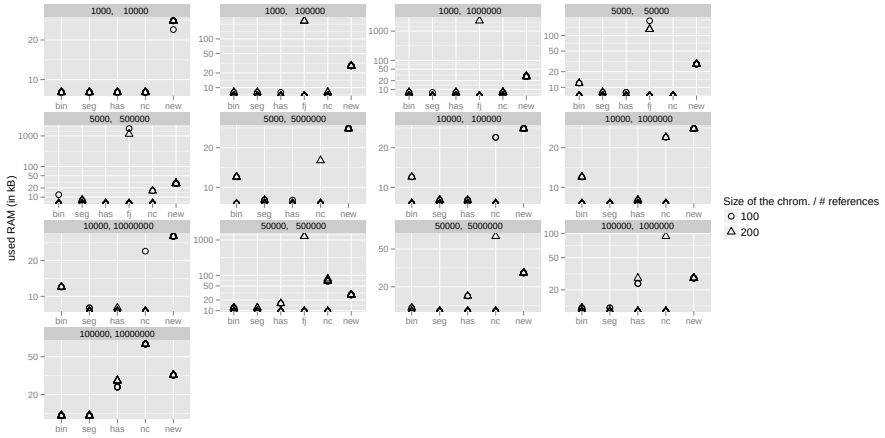



FIGURE 4.4 – Mémoire utilisée pour résoudre les instances simulées.

Notre méthode est toujours la plus longue lors de l'étape de *pré-processing* (cf. figure 4.5).

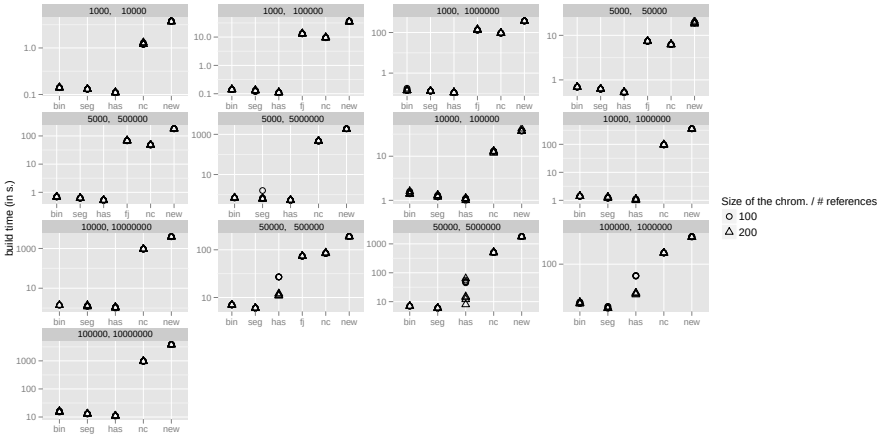



FIGURE 4.5 – Temps passé à construire les structures de données pour les instances simulées.

En revanche, après trois analyses de comparaison, notre méthode s'avère en général la plus rapide lorsque le temps total est considéré (cf. figure 4.6).


FIGURE 4.6 – Temps passé à construire les structures de données et résoudre trois fois des instances simulées.

# 5 MMANNOT

Cet partie est issue de (Zytnicki et Gaspin, 2020).

## 5.1 INTRODUCTION

Après l'alignement des lectures de sRNA-Seq, une des tâches habituelles est de quantifier l'abondance de chacune des classes de petits ARN. Pour ce faire, la méthode la plus simple consiste à comparer les lectures alignées avec un fichier d'annotation, qui contient l'ensemble des petits ARN connus, leur classe, et leur intervalle génomique. Il s'agit de l'étape d'*annotation*, qui est exécutée pour chaque lecture. Les classes d'ARN sont ainsi quantifiés en comptant le nombre de lectures chevauchantes avec les membres de chaque classe.

Si le principe de la quantification des classes d'ARN est en apparence simple, il existe pourtant quelques cas complexes (cf. figure 5.1).

1. Une lecture s'aligne à plusieurs endroits : si deux régions dans le génome sont identiques ou presque, comme c'est le cas après une duplication génomique, une lecture peut s'aligner aux différents *loci* (figure 5.1A).
2. Deux gènes différents se chevauchent et la lecture chevauche les deux gènes : la lecture peut être attribuée à chacun des gènes (figure 5.1B).
3. Une lecture chevauche deux gènes qui ne se chevauchent pas : la lecture se situe en général à la frontière des gènes (figure 5.1C).

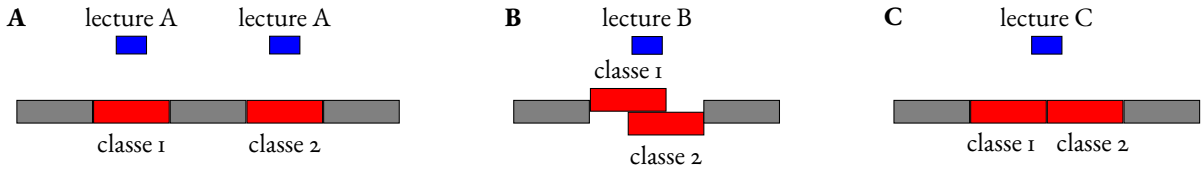



Figure 5.1 – Cas ambigus posant problème lors de la quantification. Dans chaque figure, la lecture est en bleu, les régions intergéniques en gris, et les gènes en rouge. **(A)** La lecture A s'aligne à deux endroits différents, et chaque alignement chevauche un gène. **(B)** La lecture B chevauche deux gènes qui se chevauchent. **(C)** La lecture C chevauche deux gènes qui ne se chevauchent pas entre eux.

Tout d'abord (cas 1), beaucoup de petits ARN sont intrinsèquement liées à des régions répétées. Les piARN et les siARN sont, pour partie, formés par des régions répétées et régulent l'expression des éléments transposables. Les miARN et les tsARN proviennent parfois de familles de gènes, dont les membres sont presque identiques. Ceci pose un problème majeur lors de l'étape de quantification, car une lecture, appelée *multi-alignée*, s'aligne aussi bien sur chaque *locus* dupliqué. De la même manière, il existe des gènes chevauchants, et les lectures qui s'alignent sur l'intersection de ces gènes chevauchant ne peuvent pas, sans ambiguïté, être attribuées à l'un ou l'autre gène. Plusieurs méthodes ont été proposées pour résoudre ces problèmes.

La pratique la plus courante consiste en général à ignorer ces lectures. Ceci constitue bien sûr un biais de l'approche, car tous les gènes venant de régions répétées seront systématiquement omis. Une autre approche consiste à prendre un *hit* au hasard, parmi ceux possibles. Cette solution est d'autant plus facile à mettre en œuvre qu'elle est en général implémentée par les outils d'alignement. Le biais de l'approche est que l'on ne peut absolument

pas savoir si le bon *locus* a été choisi. En règle générale, cela revient à sous-estimer l'expression du bon *locus*, et à sur-estimer l'expression des mauvais *loci*. Une méthode globalement équivalent consiste à attribuer un poids de $1/n$ lecture à chaque *locus*, où $n$ est le nombre d'alignements. Les erreurs d'estimation sont identiques à la méthode précédente. On peut y ajouter une quatrième stratégie, implémentée par des outils comme MMR (Kahles, Behr et Rätsch, 2015). MMR prédit le *locus* optimal pour chaque lecture, en se basant sur l'expression des régions adjacentes, qui sont supposées appartenir au même transcrit, exprimé de manière plus ou moins uniforme. Ces hypothèses sont difficilement vérifiables dans le cadre des petits ARN, car les transcrits ont souvent la taille d'une lecture.

ShortStack (Johnson et al., 2016) implémente une autre stratégie, particulièrement adaptée aux petits ARN. ShortStack part du même principe que précédemment, mais il est capable d'estimer le meilleur *locus* d'une lecture même à partir d'un profil d'expression très irrégulier.

Parallèlement, Vaz et al., 2015 développe une tout autre stratégie. Tout d'abord, la séquence nucléotidique des petits ARN connus est extraite, et compilée dans des bases de données, en général une base par classe d'ARN. Les lectures sont ensuite alignées non pas sur le génome, mais sur cette base de données. Le fait qu'une lecture s'aligne sur une base suggère que la lecture provient de la classe d'ARN correspondante. En revanche, cette méthode ne résout pas le problème de l'affectation multiple des lectures, car une lecture peut s'aligner sur plusieurs bases.

Parmi les autres ambiguïtés, on a également cité le cas où deux annotations se chevauchent. Ceci est parfois dû à des incohérences entre annotations. Par exemple, certains *loci* ont été annotés à la fois comme des miARN et des snoARN. Ceci traduit en général le fait que les outils de détection de miARN et de snoARN ne sont pas parfaitement spécifiques, et l'un des deux (au moins) peut se tromper car les signatures qu'ils recherchent sont relativement similaires.

Par ailleurs, les annotations peuvent également se chevaucher « naturellement ». En effet, certaines miARN sont exprimés dans des introns. Les annotations « miARN » et « intron » se chevauchent donc. En revanche, l'utilisateur préférera, dans ce cas, annoter la lecture comme un miARN, et non comme un intron.

D'une manière générale, pour répondre à la question de l'annotation des lectures, les outils classiques de quantification comme featureCounts (Liao, Smyth et Shi, 2014) ne peuvent pas être utilisés tels quels. Tout d'abord, ces outils de quantification rapportent le nombre de lecture par gène, et non par classe. De plus, certaines régions produisant des petits ARN, telles que les introns ou bien les régions flanquantes de gènes, ne sont pas présents dans les fichiers d'annotation, et ne sont donc pas utilisés pour l'annotation. Troisièmement, les outils de quantification s'attendent à ce que la librairie ait été préparée en prenant en compte le brin du transcrit (*stranded*) ou pas (*unstranded*). Dans notre cas, nous pouvons demander l'un ou l'autre, en fonction de l'annotation : il est important de repérer les ARN dans le sens contraire des gènes (*a priori* éléments régulateurs), mais le sens n'a pas d'importance dans la quantification des éléments trouvés dans les régions flanquantes.

## 5.2 MÉTHODE

### 5.2.1 *Les différentes sources d'ambiguïté*

Afin de résoudre la première source d'ambiguïté, nous nous inspirons de la méthode présentée par Robert et Watson, 2015, que nous adaptons à l'annotation des lectures. Notre méthode lit tous les *hits* d'une lecture donnée. Si tous les alignements chevauchent la même classe donnée, et/ou si les alignements restants ne chevauchent aucune annotation, alors on attribue la lecture à cette classe. Sinon, la lecture est déclarée ambiguë, et toutes les annotations chevauchantes sont renseignées. Cette méthode a plusieurs avantages :

— elle est moins biaisée,

— elle utilise toutes les lectures,
— elle utilise tous les alignements,
— comme les annotations ambiguës sont renseignées, les *loci* correspondant à ces annotations peuvent être inspectées en cas de doute.

La seconde source d'ambiguïté peut être introduite par le fait que plusieurs annotations se chevauchent. Dans ce cas, notre stratégie ordonne les différents types d'annotation. Un utilisateur peut, par exemple, décider que les miARN sont plus prioritaires que les introns, et cette ambiguïté peut ainsi être résolue dans ce cas. Le même utilisateur, peut, en revanche, décider que les miARN et les snoARN ont la même priorité. Dans ce cas, la lecture sera déclarée ambiguë.

Une dernière source d'ambiguïté peut être observée lorsqu'une lecture chevauche plusieurs annotations. Dans ce cas, l'utilisateur peut donner un seuil minimum de nucléotides chevauchants, ou bien un pourcentage minimum exprimé par rapport à la taille de la lecture. Si le nombre de nucléotides chevauchants est inférieur aux seuils donnés, l'annotation n'est pas considérée. Si plusieurs annotations peuvent toujours être attribuées à une lecture, celle-ci est déclarée ambiguë.

### 5.2.2 *Stratégie d'annotation*

mmannot a besoin de trois fichiers en entrée, et suit les trois étapes suivantes (voir Fig. 5.2) :

— mmannot lit le fichier de configuration.
— mmannot lit le fichier d'annotation et le stocke en mémoire.
— mmannot lit le fichier des lectures, et annote chacun d'entre elles.

Il est à noter que certaines annotations ne figurent en général pas dans un fichier d'annotation standard. Les introns et les régions proximales ne sont en général pas mentionnées, car elles peuvent être inférées directement des annotations. En conséquence, mmannot extrait les introns des annotations sélectionnées par l'utilisateur, et les ajoute aux annotations stockées en mémoire. De la même manière, les régions codantes, les régions non-traduites 5' et 3', les régions amont et aval (avec une taille donnée par l'utilisateur), sont également ajoutées. L'utilisateur peut également spécifier l'orientation par rapport aux annotations (sens, ou anti-sens). La figure 5.2A donne l'exemple d'un fichier GTF simple, contenant un gène codant, et cinq miARN. Dans cet exemple, le champ *source* (le second champ) est omis, et le champ *feature* (le troisième champ) donne le type d'annotation. Le gène est automatiquement reconstruit par les champs `exon`.

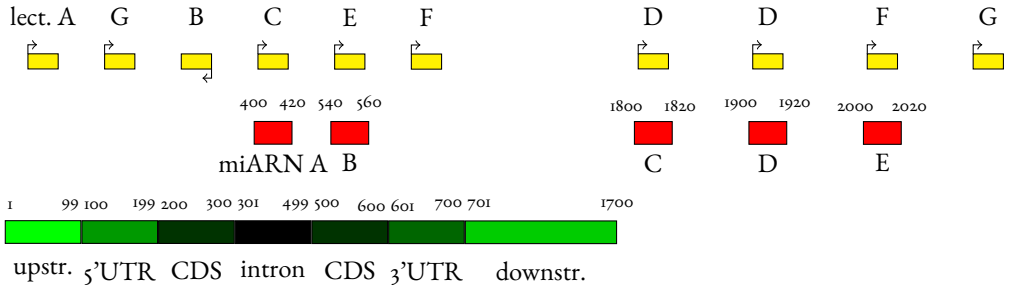
Un fichier de configuration (voir Fig. 5.2B) est également nécessaire afin de sélectionner les annotations à quantifier. La première section, `Introns`, donne la liste des éléments dont les introns doivent être extraits. La section suivante, `Vicinity`, donne la liste des éléments dont les régions amont et aval doivent être extraites. La dernière section, `Order`, donne la liste des annotations qui doivent être quantifiées. Le deux-points (`:`) est le séparateur entre le champ *source* et *feature*. En l'occurrence, la ligne `.:CDS` indique qu'il faut compter le nombre de lectures qui chevauchent les annotations qui ont pour champ *feature* `CDS` dans le fichier GTF. Si l'élément est suivi du signe « plus » (`+`), mmannot ne compte que les lectures qui sont dans le même sens que l'annotation. De même, si l'élément est suivi du signe « moins » (`-`), mmannot ne compte que les lectures qui sont dans le sens contraire. Les annotations sont données par ordre de priorité. Dans l'exemple donné, `CDS +` est avant `intron`, il est donc plus prioritaire. En revanche, `CDS +` et `5'UTR`, étant sur la même ligne, sont également prioritaires.

L'annotation se divise en deux étapes. La première étape trouve les annotations chevauchantes à tous les alignements d'une lectures (voir Fig. 5.2C). Si un alignement chevauche plusieurs annotations, l'annotation ayant la plus grande priorité sont gardée (voir la lecture C dans l'exemple). Si plusieurs annotations ont la même priorité, celles-ci sont toutes gardées, et la lecture est déclarée ambiguë (voir la lecture E). Dans l'exemple, les annotations

**A**
```
1  .  exon    100   300  .  +  .  gene_id "geneA"
1  .  exon    500   700  .  +  .  gene_id "geneA"
1  .  CDS     200   600  .  +  .  gene_id "geneA"
1  .  miRNA   400   420  .  +  .  gene_id "mirnaA"
1  .  miRNA   540   560  .  +  .  gene_id "mirnaB"
1  .  miRNA  1800  1820  .  +  .  gene_id "mirnaC"
1  .  miRNA  1900  1920  .  +  .  gene_id "mirnaD"
1  .  miRNA  2000  2020  .  +  .  gene_id "mirnaE"
```

**B**
```
Introns:
.:gene
Vicinity:
.:gene
Order:
.:CDS +, .:5'UTR +, .:3'UTR +, .:miRNA
.:intron
.:gene -
.:upstream, .:downstream
```

**C**




**D**

| | |
|---|---|
| A | upstream |
| B | gene (−) |
| C | miRNA |
| D | miRNA |
| E | CDS (+)—miRNA |
| F | 3'UTR (+)—miRNA |
| G | 5'UTR (+) |

**E**

| | |
|---|---|
| CDS (+)—miRNA | 1 |
| 3'UTR (+)—miRNA | 1 |
| 5'UTR (+) | 1 |
| miRNA | 2 |
| gene (−) | 1 |
| upstream | 1 |

**F**

| | |
|---|---|
| geneA (−) | 1 |
| geneA :3'UTR (+)—mirnaE | 1 |
| geneA :5'UTR (+) | 1 |
| geneA :CDS (+)—mirnaB | 1 |
| geneA :upstream | 1 |
| mirnaA | 1 |
| mirnaC—mirnaD | 1 |

FIGURE 5.2 – **Étapes de l'annotation. (A)** Fichier au format GTF, contenant un gène codant, et cinq miARN. **(B)** Fichier de configuration. **(C)** Localisation génomique des éléments annotés (la figure n'a pas été réalisée à l'échelle). **(D)** Annotation des lectures. **(E)** Quantification. **(F)** Quantification des éléments.

liées au gène sont colorées en nuances de verts et noir. Les miARN sont en rouge, les lectures

en jaune, étiquetées de A à G. Les lectures D, F, et G apparaissent plusieurs fois, car elles peuvent s'aligner sur plusieurs *loci*. Les lectures sont orientées, et la flèche indique le sens.

La deuxième étape résout les ambiguités et annote chaque lecture. Elle se subdivise en plusieurs sous-étapes.

— Si une lecture s'aligne uniquement, sans ambiguïté, le nombre de fois que l'annotation est vue est incrémenté.
— Si une lecture s'aligne sur plusieurs *loci*, mais tous les alignements chevauchent la même annotation, la lecture est déclarée « récupérée », et le nombre de fois que l'annotation est vue est également incrémenté (lecture D). De la même manière, si une lecture s'aligne sur une annotation, et une ou plusieurs régions intergéniques, alors la lecture est récupérée de la même manière.
— Si une lecture chevauche plusieurs annotations, les annotations avec la plus grande priorité est gardée. Toutefois, s'il n'y a qu'une seule annotation avec la plus grande priorité, la lecture n'est pas ambiguë.
— Dans les autres cas, il y a ambiguïté. Un nouveau type d'annotation, appelé « annotation groupée », est créée, et son décompte est incrémenté. Il s'agit de la concaténation des annotations les plus prioritaires chevauchantes.

L'annotation individuelle de chaque lecture (voir Fig. 5.2D) peut être produite par mmannot.

Dans l'exemple, la lecture A ne chevauche qu'une seule région, qui, elle-même, ne chevauche que la partie amont d'un gène. La lecture peut donc être attribuée de façon non ambiguë à cette annotation. La lecture B ne chevauche que la région codante du gène. En revanche, la lecture est sur le brin −, alors que le gène est sur le brin +. Le fichier de configuration spécifie que seuls les lectures dans le sens du gène ne peuvent être être attribuées aux régions codantes ; cette ligne du fichier ne s'applique donc pas ici. La ligne `gene -` du fichier de configuration peut en revanche être utilisée, et nous attribuons la lecture à cette annotation. La lecture C s'aligne uniquement, et chevauche un miARN et un intron. La lecture D s'aligne sur deux *loci*, et ces deux régions chevauchent des miARN. La lecture est donc attribuée à un miARN, et elle est déclarée « récupérée ». La lecture E chevauche une région codante, et un miARN. Ces deux annotations ont la même priorité. L'ambiguïté ne peut ici pas être résolue, et on attribue E à `CDS (+)-miRNA`. La lecture F s'aligne sur plusieurs *loci* : un *locus* chevauchant avec une région non-codante 3', et un autre chevauchant un miARN. L'ambiguïté ne peut ici pas être non plus résolue, et on attribue F à `3'UTR (+)-miRNA`. Enfin, la lecture G chevauche une une région non-codante 5' et une région intergénique non annotée. Dans ce cas, on affecte G à `3'UTR`, et on déclare la lecture récupérée.

Après avoir lu toutes les lectures, la table de quantification est construite (voir Fig. 5.2E).

La table de quantification de chacun des éléments (Fig. 5.2F) peut également être produite par mmannot. Cette table compte le nombre de lecture par élément donné dans le fichier d'annotation, ainsi que les éléments groupés. Grâce à cette table, l'utilisateur peut par exemple noter que le miARN vu au travers de la lecture E peut avoir pour cible potentielle la région région non-codante 3'. On peut également noter une ambiguïté entre la région codante et le miARN B, ce qui peut être la conséquence d'une erreur dans l'annotation du génome.

### 5.2.3 *Implémentation*

mmannot procède comme suit. Il lit tout d'abord le fichier d'annotation, extrait les intervalles correspondant à chaque annotation, les trie, et les stocke dans des *bins* non chevauchants. mmannot lit ensuite le fichier contenant les lectures alignées, extrait les *bins* correspondants, et compare les annotations avec chaque lecture. Les annotations correspondantes sont gardées en mémoire tant que tous les alignements de chaque lecture ne sont pas lus. Lorsque tous les alignements d'une lecture sont analysés, le décompte

de l'annotation correspondante (potentiellement groupé) est incrémenté. En pratique, la complexité temporelle est linéaire par rapport au nombre de lectures et d'annotations, même si certains cas (rarement rencontrés dans des cas réels) peuvent donner une complexité temporelle quadratique. mmannot a une complexité spatiale linéaire par rapport au nombre d'annotations, et potentiellement, par rapport au nombre de lectures (si une lecture est alignée à de multiples endroits). En pratique, mmannot peut être lancé sur n'importe quel ordinateur. Chaque fichier de lecture peut être lu par un *thread* indépendant.

Afin de trouver tous les alignements d'une lecture donnée, mmannot lit soit le *tag* `NH` du fichier SAM/BAM, qui compte le nombre d'alignements par lecture, ou bien le *tag* `XA`, qui donne directement tous les alignements d'une lecture. Malheureusement, certains outils d'alignement comment bowtie (Langmead, Trapnell et al., 2009) ne remplissent aucun des deux champs. Un outil accessoire est donc également fourni. Il lit un fichier SAM, et ajoute le *tag* `NH`, avec la valeur correcte.

### 5.2.4 *Données*

Nous avons comparé notre méthodes à d'autres stratégies, sur plusieurs organismes complexes, mais nous ne présenterons ici que les résultats sur *A. thaliana* (Fig. 5.3). Nous avons également généré un jeu de données simulées, en reprenant une méthode développée par l'outil ShortStack (Johnson et al., 2016), qui génère à la fois les lectures, ainsi que les annotations de miARN et de piARN (Fig. 5.4).

Nous avons comparé notre méthode avec les autres méthodes précédemment mentionnées : « unique », « random », et « ratio », ainsi qu'avec les outils ShortStack et MMR. Nous avons utilisé MMR avec bowtie, car il ne semble pas interpréter correctement le fichier SAM produit par bwa. Il est à noter que MMR demande beaucoup plus de mémoire vive que les autres outils, et s'est terminé prématurément avec un message d'erreur sur le jeu de données humaines, après avoir demandé plus de 32Go de mémoire.

Nous avons également utilisé featureCounts (Liao, Smyth et Shi, 2014) dans la stratégie nommée « featureCounts ». Pour cela, nous avons extrait les annotations des petits ARN du fichier d'annotation, ajouté les annotations manquantes (introns et régions flanquantes), et généré un nouveau fichier d'annotation au format SAF (lisible par featureCounts). featureCounts donne la possibilité de supprimer les lectures ambiguës, qui est l'option par défaut, ou bien de compter un alignement par gène chevauchant. Nous avons laissé l'option par défaut.

Nous avons finalement implémenté une dernière stratégie (stratégie « sequence »). En utilisant le génome de référence, nous avons extrait les séquences correspondant au fichier d'annotation utilisé. Nous avons donc constitué une base nucléotidique de référence. Nous avons ensuite aligné les lectures sur cette base. Nous avons toutefois noté que certaines lectures, qui s'alignaient sans erreur sur le génome (en observant l'alignement produit pour mmannot), s'alignaient avec erreur sur la base de données. Ceci conduit à attribuer une lecture à un petit ARN, à tort. Pour éviter ce bais, nous avons également également ajouté le génome entier à notre base de données, et adopté une nouvelle stratégie. Si une lecture s'aligne avec le même nombre d'erreurs sur un petit ARN et le génome, la lecture est attribuée au petit ARN. Si une lecture s'aligne sur le génome avec moins d'erreur que sur un petit ARN, ou qu'elle s'aligne sur le génome sans s'aligner sur un petit ARN, la lecture est déclarée non-annotée. En revanche, cette stratégie ne résout pas les problèmes de lectures ambiguës, car une lecture peut s'aligner sur plusieurs petits ARN. Dans ce cas, nous attribuons la lecture à la première classe sur laquelle l'outil (ici, bwa) a aligné la lecture. Cette méthode demande beaucoup plus de mémoire vive, notamment pour aligner à la fois sur le génome et les petits ARN. En conséquence, nous n'avons pas pu utiliser cette méthode sur le jeu de données humain avec un ordinateur de 32Go de mémoire.

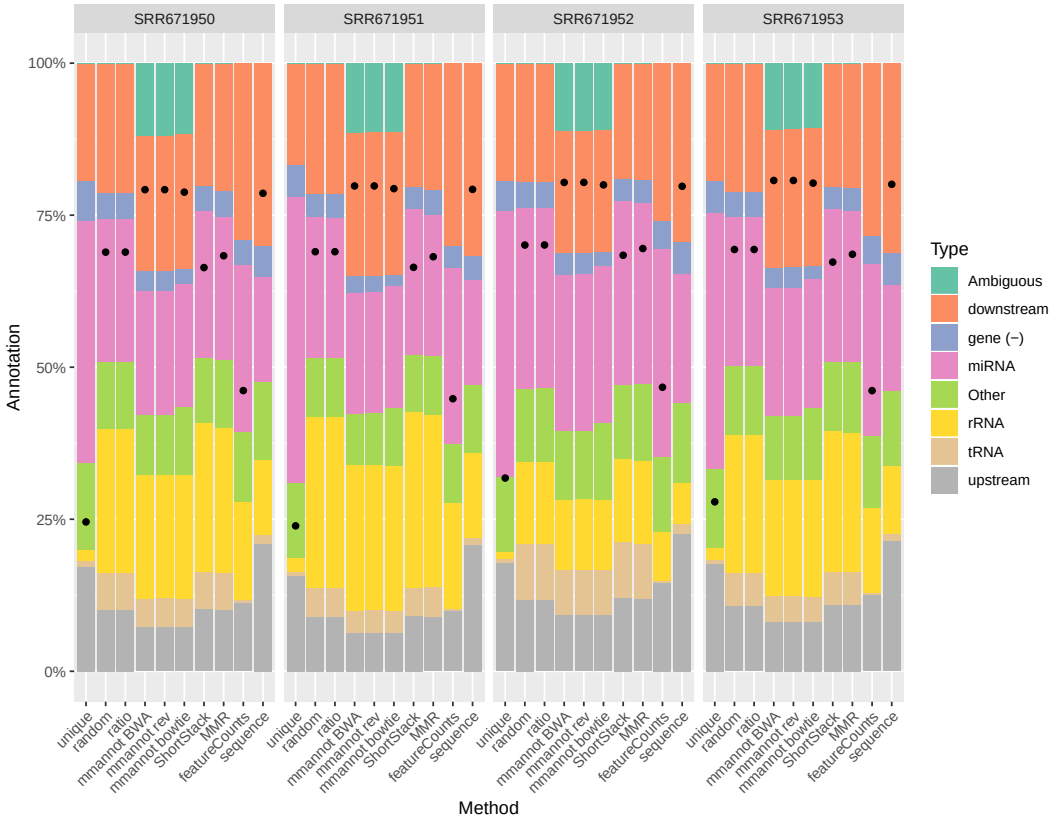



FIGURE 5.3 – Comparaison des différentes stratégies sur le jeu de données *A. thaliana*. Chaque sous-figure donne les résultats d'un séquençage. La stratégie « unique » n'utilise que les lectures ne s'alignant qu'à une seule position. La stratégie « random » utilise un alignement au hasard pour chaque lecture. La stratégie « ratio » attribue $1/n$ à chaque alignement, où $n$ est le nombre d'alignements. Notre méthode utilise bwa pour l'alignement, et est nommée « mmannot BWA ». Dans les figures « mmannot rev », les lignes du fichier de configuration ont été inversées. La stratégie « mmannot bowtie » utilise bowtie à la place de bwa. Les stratégies « ShortStack », « MMR » et « featureCounts » utilisent les stratégies implémentées par ces outils. Enfin, la stratégie « sequence » consiste à aligner les lectures sur des banques de petits ARN (et pas sur le génome). Chaque colonne donne le nombre de lecture s'alignant sur chaque annotation, en pourcentage. La catégorie « ambiguë » donne le nombre de lectures s'alignant à plusieurs endroits. Les classes exprimant peu d'ARN sont groupées dans la catégorie « other ». Les points, dans chaque colonne, donnent le pourcentage de lectures alignées qui ont été annotées.

### *La restriction aux lectures s'alignant qu'à un seul locus donne une fausse image du répertoire des petits ARN*

Les résultats en terme de pourcentages présence de chaque classe donnés stratégie « unique » sont très biaisés. Dans le jeu de données *A. thaliana* (figure 5.3), cette stratégie ne trouve pas les lectures issues de l'ARN ribosomique, qui est très répété dans le génome. De plus, la stratégie « unique » donne environ 40% de miARN, alors que les autres stratégies, qui considèrent les lectures alignées plusieurs fois, prédisent que ces miARN représentent plutôt 20% des lectures. La stratégie « unique » trouve entre 126 et 137 miARN exprimés, alors que mmannot en trouve plutôt entre 149 et 159. Même si la différence ne semble pas grande, mmannot trouve par exemple qu'une famille, *miR157*, qui est présente en trois copies identiques, produit entre 35 000 et 110 000 lectures. La stratégie « unique » ne peut pas annoter ces lectures, et ignorer une seule famille d'ARN (mais plusieurs *loci*) revêt une importance majeure.

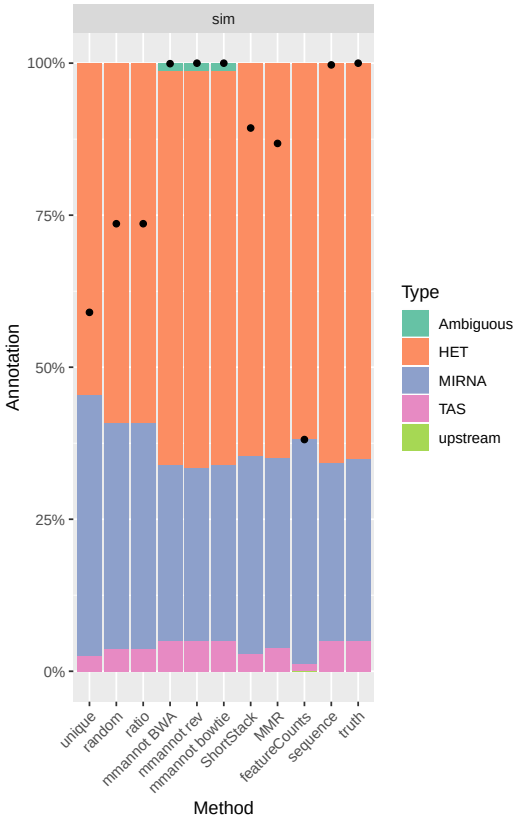



FIGURE 5.4 – Comparaison des différentes stratégies sur le jeu de données simulées. La légende est similaire à la figure 5.3, et la colonne « truth » donne la fraction de petits ARN générée.

*Utiliser les lectures multi-alignées augmente le nombre de lectures annotées*

Pour tous les jeux de données, utiliser les lectures augmente considérablement le pourcentage de lectures annotées. Ceci montre que le répertoire des régions exprimées est, pour une part importante, associée aux régions répétées. Dans le jeu de données *A. thaliana* (figure 5.3), la stratégie « unique » annote entre 0,8 et 1,8 millions de lectures. À l'opposé, les stratégies mmannot et « séquence » en annotent entre 2,5 et 5,2, les autres stratégies se situant entre les deux extrêmes.

Certains des alignements des lectures multi-alignées peuvent ne pas chevaucher avec une annotation. Ces lectures peuvent alors ne pas être annotées par la stratégie « random », et le poids associé dans la stratégie « ratio » est perdu. De même, ces lectures peuvent être mal placées par les méthodes MMR et ShortStack. En conséquence, ces lectures peuvent ne pas être utilisées pour l'annotation. mmannot, en revanche, utilise tous les alignements possibles pour chaque lecture, et les utilise pour annoter cette lecture.

*mmannot contrôle le nombre de lecture mal positionnées*

La stratégie « random » place aléatoirement les lectures. La figure 5.5 montre que cette méthode donne de nombreux faux négatifs. ShortStack et MMR peuvent être vus comme des amélioration de l'approche, car ils tendent de placer la lecture à l'endroit le plus vraisemblable. Toutefois, la même figure montre que ces prédictions sont parfois fausses. Environ 10% de ces lectures sont placées sur un *locus* contenant une autre annotation, ou bien placées hors annotation. mmannot, en revanche, donne toujours la bonne annotation.

La méthode « sequence » aligne beaucoup de lecture, mais elle attribue également beaucoup de lectures multi-alignées.

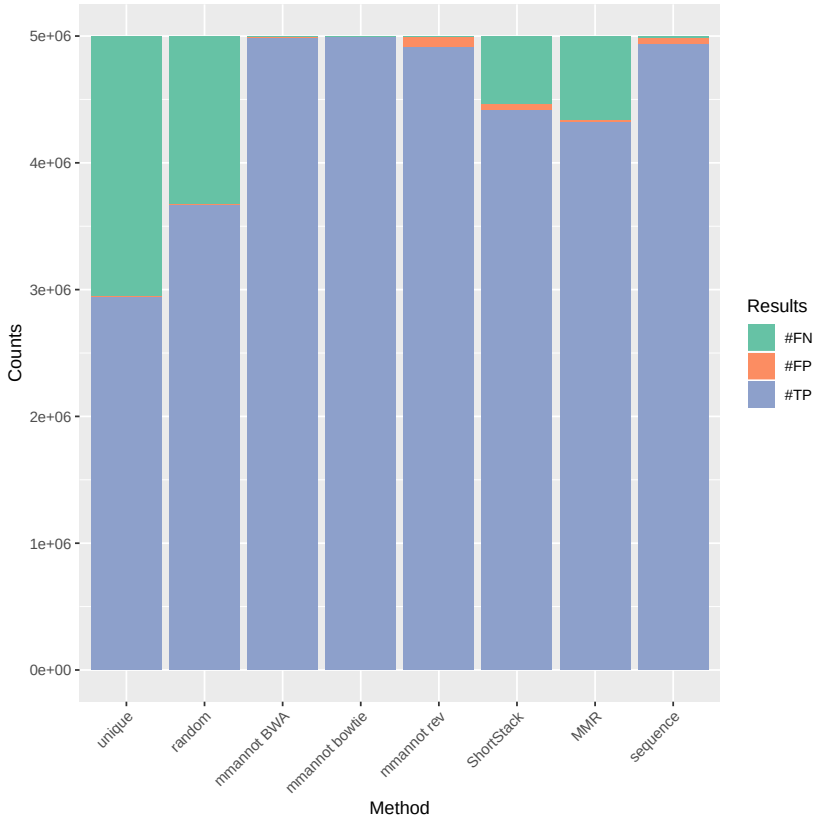



FIGURE 5.5 – Résultat sur des données simulées. Les nombres de faux positifs (#TP), faux positifs (#FP), et faux négatifs (#FN), sont donnés pour chaque méthode. Les lectures multi-alignées sont classifiées comme vrai positifs si au moins une annotation donnée par mmannot est correcte.

*Les prédictions de mmannot sont plus précises*

La figure 5.5 montre que seuls mmannot et la stratégie « sequence » ne donne pas de faux négatifs. La figure 5.4 montre également que les quantifications MMR et ShortStack sont en général correctes, même si quelques lectures sont attribuées à la mauvaise classe. Ceci est dû au fait MMR et ShortStack fois parfois un mauvaise prédiction du placement de certaines lectures, ce qui peut occasionnellement poser problème.

*Toutes les stratégies ne parviennent pas à annoter une partie des lectures*

Quelle que soit la stratégie utilisée, ou l'organisme, toutes les jeux de données présente un pourcentage élevé de lectures (jusqu'à 20% chez *A. thaliana*) qui sont localisées dans des régions intergéniques, loin des régions régions aval et amont de gènes, où aucune annotation n'est renseignée.

*Analyse des lectures ambiguës de A. thaliana*

Nous avons analysé plus finement les lectures ambiguës de *A. thaliana* afin de tenter de comprendre la cause de cette ambiguïté. La plupart des ambiguïtés viennent du fait que ces lectures peuvent se positionner sur des régions aval de gène et sur des régions amont de gène (mais à une autre position). Ceci est sans doute le résultats d'une duplication génomique. Le deuxième le plus fréquent de lectures ambiguës sont des lectures qui s'alignent à la fois sur un transcrit, et sur une région aval, amont, ou intronique (encore une fois, à une autre position). Ce peut être dû à une duplication d'un gène en un élément qui n'est plus fonc-

tionnel. L'ambiguïté la plus représentée qui contient uniquement des régions transcrites est constituée par l'ensemble des lectures qui s'alignent à la fois sur un miARN, et sur un gène anti-sens. Chez les plantes, le miARN et sa cible peuvent être identiques à 100%, ce qui pose problème lors de l'annotation des lectures. Parmi ces cas ambigus, nous nous sommes concentrés sur les associations qui étaient supportées par au moins cent lectures. Nous y avons trouvé des interactions miARN/gène déjà documentées : *miR156/miR157* avec *SPL* (Gandikota et al., 2007), *miR163* avec *PXMT1* (Allen, Xie, Adam M Gustafson et al., 2004), *miR171* avec *ATHAM* (Brodersen et al., 2008), *miR400* avec *PPR1* (Park et al., 2014), *miR403* avec *Ago2* (Allen, Xie, Adam M. Gustafson et al., 2005), et *miR824* avec *AGL* (Kutter et al., 2007). Ces lectures, très intéressantes, ne peuvent pas être correctement annotées par les autres méthodes, et l'expression des miARN est donc sous-estimée.

## 5.3 CONCLUSION

Une large proportion de lectures produites par le sRNA-Seq s'aligne à plusieurs endroits, et ceci pose problème lors de l'étape d'annotation et de quantification, car les lectures ne peuvent pas être annotées de façon non-ambiguë. À ce jour, la plupart des outils d'annotation n'utilise pas ces lectures, ou les distribue suivant différents modèles, qui peuvent être vérifiés, ou pas. En traitant de façon efficace ces lectures, mmannot améliore l'annotation des petits ARN, ainsi que la précision de la quantification. Par rapport aux autres stratégies décrites précédemment, les avantages de mmannot sont les suivants. Tout d'abord, mmannot n'introduit pas de biais dans l'annotation, et ne défavorise pas les classes les plus dupliquées. Il utilise plus de lectures que les autres méthodes pour l'annotation. Il donne la proportion de lectures effectivement ambiguës. Il a une meilleure sensibilité et spécificité que les autres outils. Il permet de trouver les annotations concernées par ces lectures ambiguës, comme les miARN et leur cible. Enfin, il permet de trouver des annotations suspectes, qui sont révélées par des classes associées de façon inattendues. Ces avantages permettront peut-être de comprendre un peu plus la « matière noire » répétée dans les génomes.

# 6 MMQUANT

L'article est issu de la publication Zytnicki, 2017.

## 6.1 INTRODUCTION

Nous supposons ici que nous avons à disposition un génome de référence, des données de séquençage, et une annotation. Le but est ici de quantifier l'expression de chacun des gènes et non pas, comme précédemment, l'expression des classes de petits ARN. Il est à noter que problème s'applique aussi bien aux petits ARN qu'aux ARN messagers.

En général, la quantification de l'expression de gènes ne fait sens que dans un contexte différentiel, où une condition d'étude est comparée à une condition de référence. Les différences d'expression sont alors supposées être liées à la condition d'étude.

L'analyse d'expression différentielle est en général découpée en plusieurs étapes : l'alignement des lectures sur le génome (ou le transcriptome), la quantification de l'expression des gènes, et l'expression différentielle (voir figure 6.1).

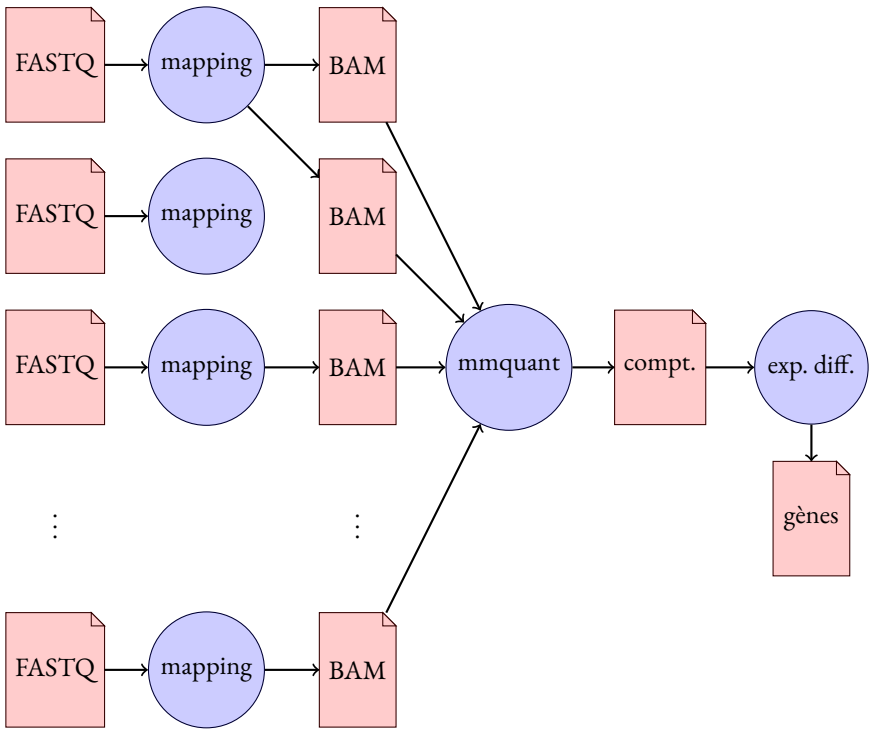



Figure 6.1 – Étapes possibles pour l'analyse différentielle en RNA-Seq. L'analyse commence avec plusieurs FASTQ, comprenant deux conditions, et plusieurs répliques. Lorsqu'un génome est accessible, les lectures sont alignées et les alignements sont stockés dans un fichier au format BAM. Un outil de quantification, comme mmquant, compte le nombre de lectures par gène. Une analyse statistique est ensuite conduite sur les comptage, par exemple en utilisant DESeq2 (Love, Huber et Anders, 2014).

Dans le cas des lectures multi-alignées, les trois stratégies de comptage évoquées précédemment (supprimer les lectures multi-alignées, prendre un *locus* au hasard, ou bien pondérer les *loci*) peuvent également être utilisées.

Les méthodes comme RSEM (B. Li et Dewey, 2011) estiment l'expression de chaque gène, en utilisant un algorithme EM (*expectation-maximization*), en se basant notamment sur le comptage des parties qui ne sont pas dupliquées. Ces estimations tiennent également compte de la position sur le transcrit (les extrémités 5' et 3' sont en général moins couvertes) et du contenu nucléotidique (les régions à fort GC% sont plus difficiles à séquencer), car on sait que le séquençage dépend beaucoup de ces paramètres. Toutefois, la qualité de l'estimation reste difficile à évaluer, car il n'existe pas réellement de jeu de test de référence validé.

Les auteurs de featureCounts (Liao, Smyth et Shi, 2014) proposent, pour certains génomes, des annotations légèrement modifiées, où les gènes chevauchants sont groupés. La quantification s'opère donc non pas sur les gènes renseignés dans l'annotation, mais sur les gènes groupés.

Nous appliquerons ici la même méthode que développée dans le chapitre précédent (cf. figure 6.2). Lorsque une lecture s'aligne sur deux gènes différents, A et B, on crée un nouveau « gène groupé » nommé A–B, et on attribue la lecture à ce nouveau « gène ». La méthode donne le nombre de lectures chevauchant à la fois les vrais gènes, et les gènes groupés. Ces gènes groupés peuvent être utilisés par la suite comme des gènes classiques pour l'étape de recherche d'expression différentielle. L'intérêt de l'approche est qu'elle exploite toutes les données du RNA-Seq, sans supposition ni inférence.

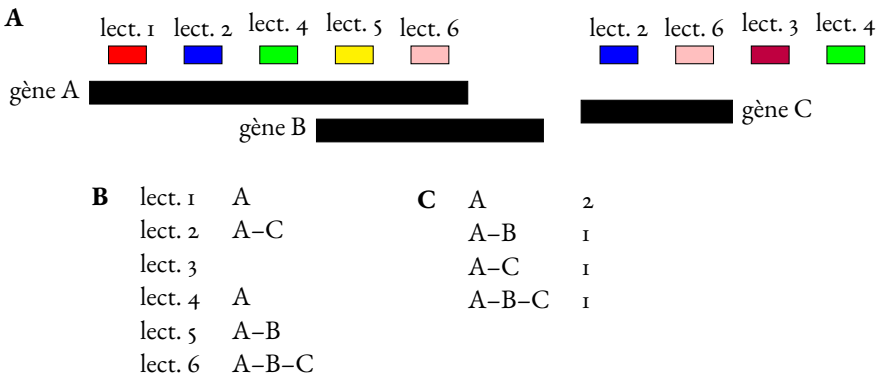



Figure 6.2 – Étape de résolution des ambiguïtés. **A** : Dans cet exemple, on lit trois gènes en noir : A, B (qui se chevauchent), et C. Six lectures sont alignées, dont trois (2, 4, et 6) s'alignent à plusieurs endroits. Si une lecture s'aligne de façon non-ambiguë, et ne chevauche qu'un seul gène (comme la lecture 1), on attribue cette lecture au gène. Si une lecture, comme la lecture 2, chevauche deux gènes, on crée un nouveau gène groupé, ici A–C, et on attribue cette lecture à ce gène. Une lecture qui ne chevauche aucun gène (comme la lecture 3) n'est pas utilisée. Si une lecture (lecture 4) chevauche un gène et une région intergénique, elle est attribué au gène uniquement. Si une lecture (lecture 5) chevauche deux gènes, car ils se chevauchent entre eux, on crée également un nouveau gène groupé (ici, A–B). De la même manière, si une lecture (lecture 6) chevauche deux gènes chevauchants, et un troisième gène, alors on crée un nouveau gène groupé (ici, A–B–C). La table **B** donne les gènes attribués pour chaque lecture. La table **C** donne le résultat de la quantification, et est le résultat principal de l'outil. Les noms sont classés par ordre lexicographique, si bien que le gène groupé comprenant A et B sera toujours nommé A–B, et jamais B–A.

Les concepteurs de cette méthode (Robert et Watson, 2015) ont conçu un prototype qui implémente la méthode. Toutefois, les lectures et l'annotation doivent avoir un format particulier, non standard. Des paramètres essentiels pour contrôler les résultats de la méthode ne sont pas proposés (comme le type de la librairie), Enfin, la méthode, implémentée en Perl, est lente, et peu adaptée au jeux de données modernes qui stockent plusieurs gigabases de données. Nous proposons ici une méthode qui étend ces travaux, implémente plusieurs fonctionnalités essentielles proposées par les outils habituellement utilisés tel que featureCounts (Liao, Smyth et Shi, 2014) ou htseq-count (Anders, Pyl et

HUBER, 2015), avec une complexité temporelle et spatiale comparable. Cet outil se propose comme un remplaçant des méthodes actuelles, plus informatif, et sans désavantage.

Nous appliquons également d'autres règles pour les autres cas d'ambiguïté. Une première règle décide si un *hit* d'une lecture chevauche un gène donné. Ceci se fait en fonction de la valeur d'un paramètre donné par l'utilisateur. Si ce paramètre est négatif, la lecture doit être emboîté dans le gène (hypothèse conservatrice). Si ce paramètre est un entier positif $n$, alors la lecture et le gène doivent avoir au moins $n$ bases communes. Si ce paramètre est un entier décimal $x$, plus petit que 1, alors lecture et le gène doivent avoir au moins $\frac{x}{100} \times t$ bases communes, où $t$ est la taille de la lecture.

La figure 6.3 donne l'ensemble des règles utilisées lorsque deux annotations se chevauchent.

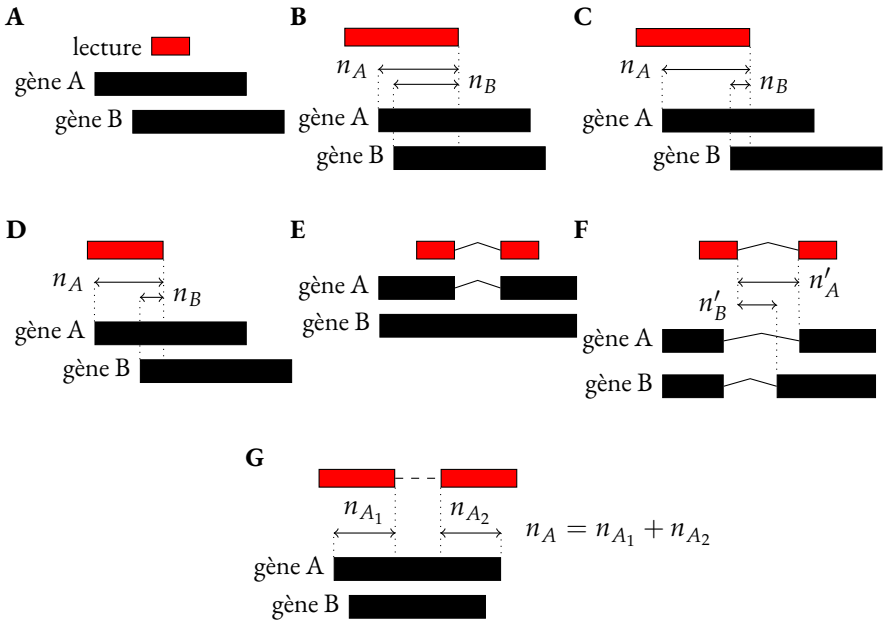



FIGURE 6.3 – Résolution des ambiguïtés. **A** : La lecture est incluse dans les gènes A et B. Ici, l'ambiguïté ne peut être résolue, et la lecture sera attribuée au groupe de gènes A–B. **B** : La lecture n'est pas totalement incluse dans le gène A, ni dans le gène B. La lecture a $n_A$ nucléotides en commun avec A, $n_B$ avec B, et $n_A > n_B$. Si $n_A \gg n_B$, on peut attribuer la lecture à A uniquement. En revanche, si if $n_A \approx n_B$, l'ambiguïté ne peut être résolue et la lecture sera attribuée à A–B. Dans cette approche, nous avons utilisé les deux règles suivantes. **C** : Si $n_A > n_B + N$, où $N$ est un paramètre contrôlable par l'utilisateur (par défaut, 30), mmquant attribuera la lecture à A uniquement. **D** : Si $n_A > n_B \times P$, où $P$ est un autre paramètre contrôlable par l'utilisateur (par défaut, 2), la lecture sera attribuée à A uniquement. **E** : Ici, la lecture *single end* contient un intron. En se basant exclusivement sur l'information exonique, la lecture peut être attribuée indifféremment à A ou à B. En cas d'ambiguïté, les introns sont comparés. L'intron de la lecture correspond à celui de du gène A, tandis que B n'a pas d'intron à cet endroit. On peut attribuer de façon unique la lecture à A. **F** : Sur la base des exons, la lecture est ambiguë. On calcule $n'_A$ et $n'_B$, le nombre de nucléotides communs entre les introns de la lecture, et les introns des gènes A et B. L'ambiguïté est résolue en appliquant les règles vues dans **C** et **D** sur $n'_A$ et $n'_B$. **G** : La lecture est *pair ends*. En cas d'ambiguïté, $n_A$ et $n_B$ sont calculés comme la somme des bases chevauchantes entre les deux lectures et les gènes A et B. On applique ensuite les règles vues dans **C** et **D**.

L'outil prend en charge les lectures pairées, et vérifie que les deux lectures s'alignent bien sur le même transcrit, de façon cohérente avec le protocole utilisé lors de la création de la librairie (la première lecture peut être dans le sens du transcrit, et la seconde, sur l'autre brin, mais toutes les stratégies sont supportées par l'outil). Les fragments sont alors comptés lors de la quantification.

Pour comparaison, htseq-count a trois modes : « union », « intersection-strict », et « intersection-nonempty ». mmquant peut fonctionner de façon similaire du mode « union » (préconisé par la documentation de l'outil), ainsi que le mode « intersection-strict », mais pas le mode « intersection-nonempty ». Dans htseq-count, les lectures ambiguës ne sont pas utilisées. Dans featureCounts, la stratégie par défaut supprime les lectures multi-alignées, mais il est possible d'utiliser tous les alignements, et d'utiliser la méthode de pondération. De plus, une option permet d'attribuer une lecture à gène avec lequel il chevauche le plus (en nombre de bases). Cette stratégie peut aussi être utilisée avec mmquant.

## 6.2 RÉSULTATS

Nous avons comparé notre méthode sur des données présentées par Akula et al., 2014. Dans cette étude, des données de RNA-Seq ont été produites à partir de biopsies de cerveaux humains, afin de connaître les gènes qui sont différentiellement exprimés entre patients souffrant de troubles bipolaires, et personnes saines. Notre but n'est pas ici de reproduire l'analyse (les paramètres et les versions des outils n'étant d'ailleurs pas fournis), mais de comprendre les différences entre les outils de quantification. Les données ont été séquencées avec une machine HiSeq, produisant environ 200 millions de lectures de 100 nucléotides, en *pair ends*. Il est à noter qu'il s'agit d'un cas d'étude complexe, car il est connu que des gènes dupliqués jouent un rôle important dans l'expression du cerveau humain (Geschwind et Konopka, 2012).

Notre pipe-line utilise STAR (Dobin et al., 2013) avec les paramètres recommandés par le consortium ENCODE, et DESeq2 (Love, Huber et Anders, 2014), avec les options par défaut et une p-valeur ajustée de 5%. La publication originelle utilise TopHat, htseq-count and DESeq, qui étaient les outils de référence à la date de publication. Dans leur article, les auteurs trouvent uniquement 11 gènes différentiellement exprimés (également avec une p-valeur ajustée de 5%). Nous retrouvons ces gènes, excepté un seul, un pseudo-gène, probablement dû à un défaut de TopHat, qui, dans ses premières versions, favorisait les pseudo-gènes sans introns aux transcrits matures).

htseq-count a été utilisé en mode « union », recommandé. En conséquence, si une lecture chevauche deux gènes, il est déclaré ambigu, et n'est pas utilisé pour la quantification. featureCounts a été utilisé avec les options par défaut pour le *pair ends*, et mmquant a été utilisé avec l'option permettant de déclarer qu'une lecture et un gène se chevauchent dès lors qu'ils ont une lecture en commun.

Une première observation est que la distribution des p-valeurs varie notablement en fonction de l'outil de quantification utilisé (cf. figure 6.4). htseq-count, featureCounts et mmquant (en excluant les gènes groupés) trouvent 734, 835 et 763 gènes différentiellement exprimés, respectivement. La différence vient principalement de la manière dont les gènes attribués aux gènes. htseq-count et mmquant attribuent la lecture au gène dès qu'un nucléotide est commun entre les deux, alors que featureCounts exige que les lectures soient inclues dans le gène. En conséquence, htseq-count et mmquant comptent plus de lectures par gène que featureCounts. De plus mmquant applique certaines règles pour les lectures multi-alignées (notamment le fait qu'une lecture s'alignant à la fois sur un gène et dans l'intergénique sera attribué uniquement au gène), qui sont différentes des règles appliquées par les autres outils, qui ne prennent pas en compte ces lectures. On peut ainsi vérifier que mmquant utilise plus de lectures que htseq-count. Enfin, mmquant teste plus de gènes (non-groupés, et groupés). Pour une même p-valeur, la p-valeur ajustée sera donc plus faible que pour les autres méthodes. Il s'agit d'une balance classique entre sensibilité et spécificité. Ceci se traduit directement sur les résultats. 133 des 158 gènes identifiés comme différentiellement exprimés par featureCounts et mmquant ont une p-valeur inférieure à 10%, lorsque l'on utilise mmquant.

mmquant trouve que cinq à six pour cent des lectures peuvent être attribués à plusieurs gènes. En conséquence, il trouve 254 groupes de gènes différentiellement exprimés, qui

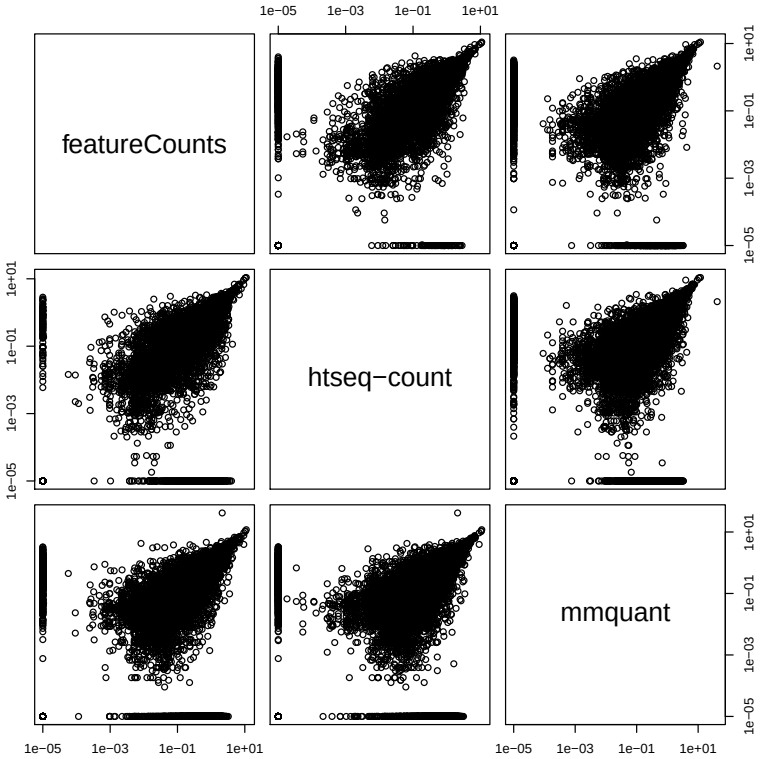



FIGURE 6.4 – Les carrés hors de la diagonale sont des comparaisons entre deux outils. Dans chaque carré, un point représente un gène. Son abscisse est le log de la p-valeur d'un outil, et son ordonnée, le log de la p-valeur de l'autre outil. Un pseudo-compte de $10^{-5}$ est ajouté pour permettre de visualiser les valeurs nulles.

regroupent 516 gènes non détectés comme différentiellement exprimés. Nous avons essayé de trouver une fonction dans ces gènes en utilisant DAVID (HUANG, SHERMAN et LEMPICKI, 2009). Aucune fonction enrichie n'est prédite. La raison se trouve sans doute dans le fait que la moitié d'entre eux n'a pas de fonction connue, certainement car ils sont dupliqués. Ceci confirme le besoin de continuer à travailler sur ces gènes dupliqués.

Nous avons ensuite étudié les 33 groupes de gène ayant une p-valeur ajustée inférieure à 1%. Ces groupes contiennent 75 gènes qui n'ont pas été détectés par ailleurs : ni par les autres outils de quantification, ni dans la recherche des gènes non-groupés différentiels trouvés par mmquant. Cette liste fournit d'excellent candidats avec des potentiellement lié aux troubles bipolaires, dont *ADK* (BOISON, 2006), *GTF2I* (SAKURAI et al., 2011), *hnRNP-A1* (X.-Y. LIU et al., 2015), *HTRA2* (PATTERSON et al., 2014), *PKD1* (WODARCZYK et al., 2009) et *RERE* (FREGEAU et al., 2017), qui sont tous liés à différentes troubles du cerveau. Certains de ces gènes présentent potentiellement des mécanismes complexes de régulation en *cis* : *ADK* and *HTRA2* chevauchent des pseudo-gènes anti-sens, et mmquant les groupe à la volée. D'autres gènes, comme *GTF2I*, *hnRNP-A1*, *PKD1*, et *RERE*, sont dupliqués, ou ont produit des pseudo-gènes en *trans*.

Concernant, le temps, featureCounts est le plus rapide des outils : il prend entre 8 à 11 minutes par échantillon. mmquant est deuxième, avec 21 à 29 minutes (plus 1 à 3 minutes si les lectures ne sont pas triées). htseq-count, écrit en Python, prend quatre à plus de cinq heures. mmquant est plus lent que featureCounts car il doit stocker (et rechercher) les lectures qui sont alignées plusieurs fois. Ces résultats ont été obtenus en utilisant un *thread*

par fichier BAM, mais featureCounts peut en utiliser plus, et accélérer ainsi d'autant plus la quantification. Cette option n'est pas possible pour mmquant et htseq-count.

Afin de confirmer ces résultats, nous avons également comparé les outils sur d'autres jeux de données, accessibles sur Gene Expression Omnibus (BARRETT et al., 2013), qui sont résumées dans le tableau 6.1. Nous avons comparé plusieurs organismes modèles eucaryotes multicellulaires, en utilisant des jeux de données incluant plusieurs stratégies de séquençage, avec des profondeurs variées. Les résultats confirment ceux trouvés précédemment. Le nombre de gènes différentiellement exprimés trouvé par chaque méthode est comparable. featureCounts et htseq-count donnent des résultats équivalents pour les jeux de données *single end*. featureCounts est l'outil le plus rapide. mmquant arrive deuxième, et htseq-count est en général beaucoup plus lent. Dans le jeux de données « souris », nous avons remarqué que les comptages et les p-valeurs brutes des gènes non-groupés sont presque identiques pour les trois méthodes. Toutefois, comme mmquant ajoute un grand nombre de gènes groupés, les p-valeurs ajustées augmentent, et donc seulement trois gènes différentiellement exprimés sont trouvés par mmquant. Dans le jeux de données « levure », nous avons remarqué que l'inclusion de lecture multi-alignées changeait drastiquement le nombre de lectures utilisées. En conséquence, la taille des librairies et les coefficients de sur-dispersion (calculés par DESeq2), sont sensiblement différents entre les résultats fournis par mmquant et les autres. Ici encore, mmquant trouve moins de gènes différentiellement exprimés. Toutefois, nous pensons que notre méthode est justifiée par le fait qu'elle utilise plus de lectures, et donc plus d'information. Il est à noter que mmquant constitue toujours des gènes groupés.

| Organisme | *D. melanogaster* | *M. musculus* | *A. thaliana* | *S. cerevisiae* |
|---|---|---|---|---|
| identifiant | GSE80323 | GSE86865 | GSE89850 | GSE83827 |
| article | HATELEYA et al., 2016 | X. LIU et al., 2017 | TANG et al., 2017 | LOSH et al., 2015 |
| nb. gènes exp. | 16383 | 24516 | 25170 | 6722 |
| nb. genes groupés | 7237 | 25302 | 5903 | 4782 |
| type | *pair ends* | *single end* | *pair ends* | *single end* |
| nb. répliques | 4 | 3 | 3 | 2 |
| nb. lectures | 27M–33M | 39M–47M | 12M–16M | 10M–14M |
| nb. alignements | 28M–34M | 53M–70M | 12M–15M | 13M–17M |
| nb. gènes fc | 1446 | 13 | 4622 | 546 |
| nb. gènes htsc | 1432 | 13 | 4557 | 546 |
| nb. genes mm | 1441 | 10 | 4599 | 388 |
| nb. fc et htsc | 1420 | 13 | 4503 | 546 |
| nb. fc et mm | 1415 | 10 | 4534 | 387 |
| nb. htsc et mm | 1399 | 10 | 4458 | 387 |
| nb. mm groupés | 191 | 2 | 394 | 97 |
| temps fc | 1mn28–1mn56 | 1mn25–2mn38 | 0mn46–1mn10 | 0mn16–0mn29 |
| temps htsc | 36mn–46mn | 21mn–25mn | 20mn–26mn | 4mn11–5mn58 |
| temps mm | 2mn35–3mn31 | 2mn06–2mn42 | 1mn19–1mn43 | 0mn20–0mn28 |
| temps mm non trié | 3mn31–4mn18 | 2mn22–2mn53 | 1mn28–2mn11 | 0mn24–0mn31 |

TABLE 6.1 – Résultats des autres jeux de données. Abréviations utilisées. nb. gènes exp. : nombre de gènes avec au moins une lecture chevauchante ; nb. genes groupés : nombre de gènes groupés trouvés par mmquant ; nb. gènes fc (resp. htsc, mm) : nombre de gènes différentiellement exprimés trouvés par featureCounts (resp. htseq-count, mmquant) ; nb. mm groupés : nombre de gènes inclus dans les groupes de gènes trouvés par mmquant.

## 6.3 CONCLUSION

La quantification est souvent perçue comme une étape simple et automatique. Toutefois, elle comprend un grand nombre de choix assez arbitraires, comme le traitement des lectures multi-alignées. mmquant est un outil simple de quantification, qui permet la quantification de gènes dupliqués. mmquant ne recourt pas à de l'inférence, ce qui est un avantage dans la mesure où la quantification des gènes dupliquée est mal estimée, y compris avec des méthodes classiques comme la qRT-PCR. Dans notre cas de test, nous montrons que mmquant peut fournir jusqu'à 25% de gènes différentiellement exprimés en plus.

# 7 SRNADIFF

L'article est issu de la publication (Zytnicki et González, 2021).

## 7.1 INTRODUCTION

La régulation de l'expression des petits ARN est en général comprise *via* un protocole d'expression différentielle, impliquant par exemple des sauvages, comparés à des mutants. Dans le protocole usuel, les expressions des petits ARN sont tout d'abord quantifiées. On construit ensuite une matrice où les gènes sont les lignes, les échantillons sont les colonnes, et l'expression des gènes, les cellules. Cette matrice est ensuite donnée à un outil d'analyse différentielle (en général conçu pour les ARN messagers), tel que DESeq2 (Love, Huber et Anders, 2014) et edgeR (Robinson, McCarthy et Smyth, 2010), qui trouve les gènes différentiellement exprimés.

Cette méthode fonctionne bien lorsque les petits ARN sont connus et correctement annotés. Malheureusement, beaucoup des ARN sont mal identifiés, et mal caractérisés, surtout s'ils ont une structure secondaire peu définie, comme c'est le cas des piARN.

*A priori*, cela réduit l'analyse des petits ARN différentiellement exprimés aux seuls bien caractérisés. Certains travaux essaient de trouver des unités transcriptionnelles à partir des lectures non annotées, en utilisant ShortStack (Johnson et al., 2016) ou S-MART. Le résultat est parfois décevant, car la clusterisation agrège souvent des petits ARN qui, quoi que proches, sont en fait deux unités transcriptionnelles différentes. En conséquence, la quantification agrège également les comptages, et fait perdre une partie importante de l'information.

Récemment, Collado-Torres et al., 2017 a présenté un outil pour la découverte *de novo* de gènes (codants) différentiellement exprimés, sans utiliser d'annotation de référence. Cette méthode fonctionne mal pour les petits ARN, car leurs profils d'expression sont très différent des profils d'expression des ARN messagers, eux-mêmes relativement uniformes. Dans ce travail, nous présentons une adaptation de cette idée, à savoir une méthode de détection des petits ARN différentiellement exprimés, sans utiliser d'annotation.

## 7.2 MÉTHODE

### 7.2.1 *Description*

La méthode est divisée en deux étapes principales, décrites sur la figure 7.1. Brièvement, la première étape lance plusieurs méthodes qui détectent des régions putatives. La seconde étape compare les régions trouvées précédemment, et sélectionne les meilleures.

*Étape 1 : trouver les régions candidates*

Cette étape transforme tout d'abord les données de séquençage en profils d'expression, encodés en *run-length encoding*, et normalisés en utilisant la procédure *counts per million* (CPM), comme dans edgeR (Robinson, McCarthy et Smyth, 2010) (cf. figure 7.2A).

Plusieurs méthodes complémentaires sont ensuite utilisées pour trouver des régions putatives. Les petits ARN ont en effet des profils d'expression extrêmement différents, selon leur classe. Les miARN, par exemple, s'accumulent sur des régions très bien définies, alors que les piARN peuvent être répartis sur plusieurs milliers de nucléotides.

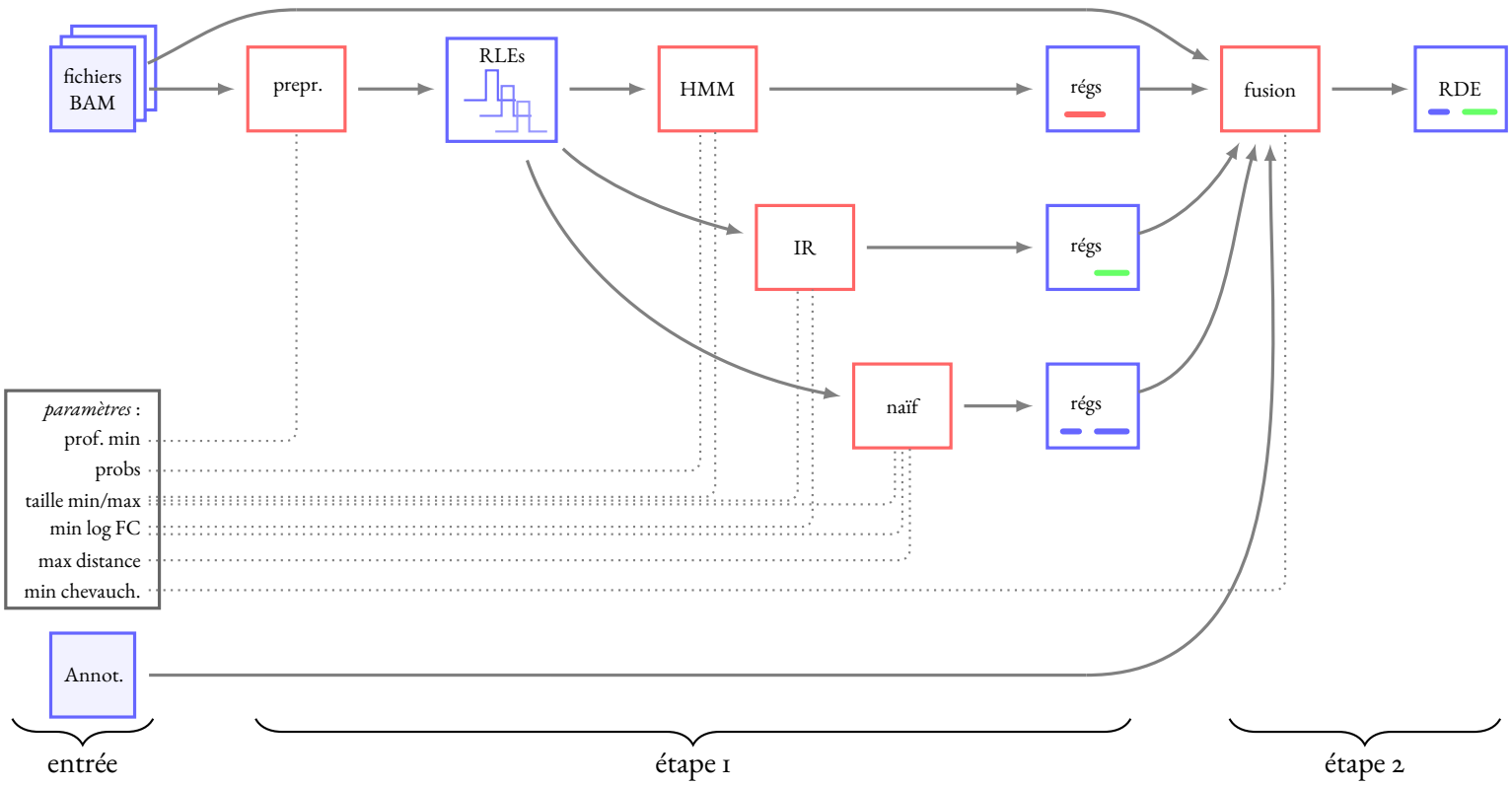



FIGURE 7.1 – Description de la méthode. Les fichiers et les données intermédiaires sont donnés dans les rectangles bleus, les algorithmes sont en rectangles rouges, et les paramètres sont dans le rectangle noir. Les flèches montrent les données utilisées par chaque algorithme. La première étape transforme les fichiers BAM en profils de couverture, encodés en *run-length encoding* (RLE). Le *run-length encoding* comprime les intervalles génomiques de profondeur identiques en signalant le nombre de fois qu'une profondeur donnée est observée. Les couvertures sont ensuite utilisées par les différentes méthodes (naïf, HMM, et IR), qui produisent des régions différentiellement exprimées, stockées dans des listes d'intervalles génomiques. L'utilisateur peut éventuellement ajouter des annotations, qui sont transformées en régions putatives. Dans une seconde étape, les régions les plus pertinentes sont sélectionnées. Ces régions peuvent se chevaucher, car elles sont proposées par des méthodes indépendantes. On calcule les p-valeurs des expressions différentielles, et, en cas de chevauchement, on choisit les régions avec les p-valeurs les plus faibles. Les paramètres utilisés sont donnés en dans les lignes en pointillé. Abréviations : prof. : profondeur ; chevauch. : chevauchement ; prepr. : pré-processing ; RLE : *run-length encoding* ; régs : régions ; RDE : régions différentiellement exprimées.

Nous avons donc implémenté trois méthodes : une méthode naïve (simple, mais souvent utilisée en pratique), une méthode basée sur les modèles de Markov cachés (*hidden Markov model*, ou HMM), et une méthode basée sur les régions irréductibles (*irreducible regions*, ou IR). Ces méthodes sont décrites ci-après.

ANNOTATION Cette étape ajoute simplement l'annotation donnée par l'utilisateur comme régions potentiellement différentiellement exprimées.

NAÏF La méthode est décrite dans la figure 7.2B. Elle implémente la stratégie habituellement utilisée.

HMM La méthode construit tout d'abord une p-valeur pour chacun des nucléotides du génome (cf. figure 7.2C). Pour cela, nous donnons, comme matrice d'entrée à DE-

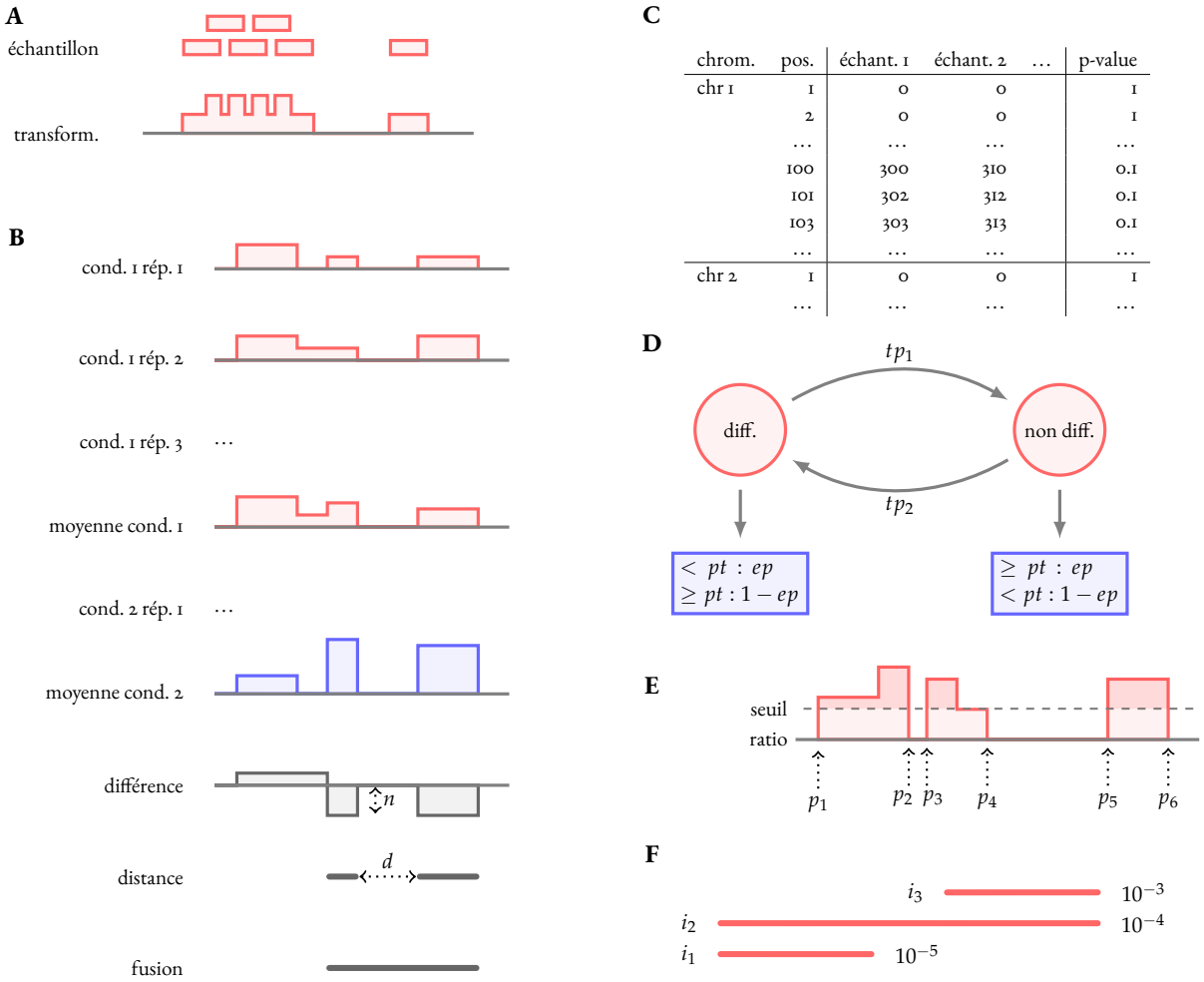



FIGURE 7.2 – Descriptions des étapes. **A** : Transformation des lectures en *run-length encoding*. **B** : Méthode naïve. On calcule la moyenne des profondeurs des échantillons, pour chaque condition, puis le $\log_2$ du ratio. Toutes les régions dont la valeur absolue du ratio est supérieure à un seuil donné sont gardées, puis éventuellement fusionnées aux régions voisines. **C** : Première partie de la méthode HMM. La profondeur de chaque échantillon est calculée pour chaque position. On calcule une p-valeur pour chacune de ces profondeurs. **D** : Seconde partie de la méthode HMM. Un HMM est utilisé sur chaque chromosome. Il contient deux états : « différentiellement exprimé » (cercle rouge « diff »), et « non différentiellement exprimé » (cercle rouge « non diff »). Les probabilités de transitions sont $tp_1$ et $tp_2$. L'état « diff » émet une p-valeur inférieure ou égale à $pt$ avec la probabilité $ep$. L'état « non diff » émet une p-valeur supérieure ou égale à $pt$ avec la probabilité $ep$. **E** : La méthode IR. Le $\log_2$ du ratio est donné en rouge, le zéro est donné en trait continu gris, et la ligne discontinue est le seuil donné par l'utilisateur. Chaque région au-dessus du seuil est une région putative, et la méthode IR se propose de les fusionner, sans utiliser de paramètre supplémentaire. Brièvement, la méthode considère chaque intervalle (par exemple $p_1$–$p_6$), calcule l'aire au-dessus du seuil, et la divise par la taille (ici, $p_6 - p_1$). On considère ensuite chaque point à l'intérieur de l'intervalle (par exemple $p_3$). Si l'aire moyenne entre $p_1$ et $p_4$, ou l'aire moyenne entre $p_4$ et $p_6$ est inférieure au seuil, l'intervalle est coupé en deux, au point $p_3$. Dans l'exemple, l'aire moyenne entre $p_4$ et $p_6$ est inférieure au seuil, l'intervalle est donc coupé. En revanche, l'intervalle n'est pas coupé entre $p_1$ et $p_4$. **F** : La fusion. On effectue l'étape de quantification et de test pour chaque intervalle, afin de donner une p-valeur. Une méthode naïve éliminerait $i_2$ et $i_3$, car ces intervalles ont une valeur supérieure à $i_1$ et $i_2$ respectivement. Dans notre méthode, nous gardons $i_3$.

Seq2 (LOVE, HUBER et ANDERS, 2014), l'ensemble des comptages trouvés sur chaque position du génome. Un HMM est ensuite construit, comme indiqué figure 7.2D. L'algorithme

de Viterbi est ensuite passé sur chacun des chromosomes. Il utilise une implémentation originale, en exploitant le fait que seule une petite partie du génome est exprimée. En conséquence, les p-valeurs sont très majoritairement 1 (en cas de comptes nuls). Lorsque la probabilité de l'état « non diff » est très grande devant l'état « diff », on peut directement passer au prochain nucléotide dont la p-valeur n'est pas 1. Ceci ne change pas les résultats de la méthode de Viterbi, tout en accélérant grandement l'implémentation.

IR La méthode est décrite dans la figure 7.2E, suit la présentation de LEŚNIEWSKA et OKONIEWSKI, 2011. L'algorithme est linéaire en temps.

*Étape 2 : fusion des régions*

La méthode est décrite dans la figure 7.2F. Il faut ici sélectionner les intervalles les plus pertinents, si certains se chevauchent entre eux. On compte donc le nombre de lectures pour chaque intervalle, et on utilise DESeq2 (LOVE, HUBER et ANDERS, 2014) pour faire les tests d'expression différentielle pour chaque intervalle. On dit qu'un intervalle $i_1$ domine un autre $i_2$ si $i_1$ et $i_2$ se chevauchent, et la p-valeur de $i_1$ est inférieure à celle de $i_2$. Notre méthode supprime tous les intervalles qui sont à la fois dominés par des intervalles, et dominent d'autres intervalles. À la fin de la procédure, ne restent que des éléments non dominés. Cela permet de garder plus d'intervalles qu'une méthode naïve supprimant tous les intervalles dominés.

### 7.2.2 *Comparaison*

Nous avons comparé nos résultats avec des données humaines (JORGE et al., 2017; KIM et al., 2013), d'*Arabidopsis thaliana* (MAY et al., 2013), de *D. melanogaster* (DHAHBI et al., 2016). Pour évaluer les méthodes, nous avons suivi le pipe-line classique d'expression différentielle, en utilisant une annotation (incluant gènes, miARN, piARN, etc.), une étape de quantification avec featureCounts (LIAO, SMYTH et SHI, 2014), et de test avec DESeq2 (LOVE, HUBER et ANDERS, 2014). Ces régions sont considérées comme des vrais positifs. Nous avons comparé notre méthode à derfinder (COLLADO-TORRES et al., 2017), et à une méthode clusterisant les lectures en petits ARN, nommée ShortStack (JOHNSON et al., 2016), suivie de DESeq2.

Nous avons également généré des données simulées, avec 44 régions différentiellemnt exprimées.

## 7.3 RÉSULTATS

Les données réelles ont été analysées par un ou deux (dans le cas humain) articles. La figure 7.3 donne le nombre de régions trouvées par chaque méthode, et montre que srnadiff trouve beaucoup plus de régions différentiellement exprimées que les autres méthodes.

Nous donnons également la sensibilité des approches, en se basant sur les vrais positifs, dans la figure 7.4. Notre méthode donne en général les meilleurs résultats, même si ShortStack est parfois plus exhaustif.

Sur nos données simulées, la courbe ROC évaluant la qualité des prédicteurs est donnée dans la figure 7.5. Cette courbe indique que la classification donnée par notre outil est bien meilleure.

Concernant les complexités temporelle et spatiale, la figure 7.6 montre que tous les outils analysent les données en moins de 15 minutes. derfinder est l'outil le plus rapide, et ShortStack (écrit en Perl), le plus lent. srnadiff est plus lent que derfinder car il lit les données une fois par méthode qu'il utilise (une fois pour HMM et une fois pour IR). De plus, derfinder utilise des données déjà transformées (au format bigWig), alors que srnadiff lit directement les données au format BAM.

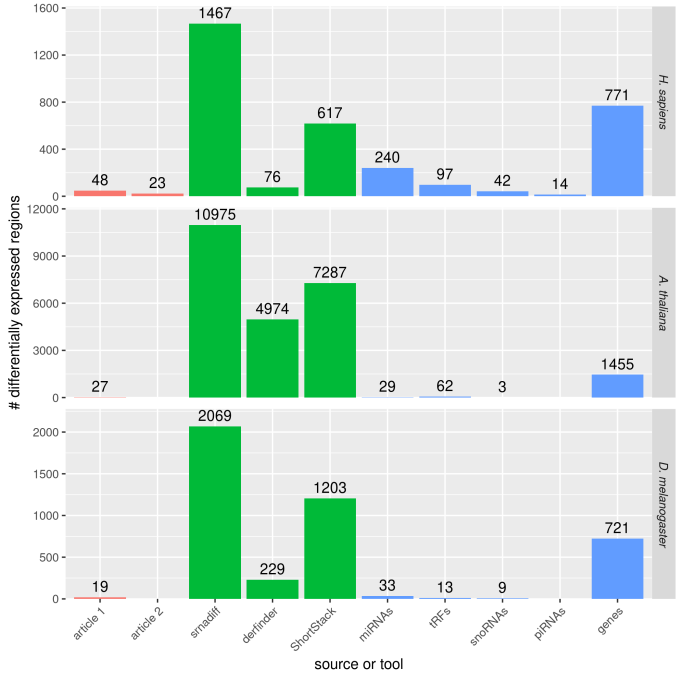



FIGURE 7.3 – Nombre de régions différentiellement exprimées, trouvées par les différentes méthodes. Les barres rouges sont le nombre de régions trouvées par les auteurs des articles (KIM et al., 2013 et JORGE et al., 2017 pour les données *H. sapiens*, MAY et al., 2013 pour les données *A. thaliana*, et DHAHBI et al., 2016 pour les données *D. melanogaster*). Les barres vertes donnent les nombres de régions trouvées par les trois méthodes comparées. Les dernières barres donnent les vrais positifs, tels que définis précédemment.

## 7.4 CONCLUSION

Nous présentons une nouvelle méthode, srnadiff, pour la détection *de novo* de petits ARN différentiellement exprimés. La méthode présente plusieurs avantages :

— elle ne suppose pas de connaissance *a priori* de petits ARN ;
— elle peut être appliquée à tout type de petits ARN ;
— elle donne de meilleurs résultats que des méthodes *ad hoc*.

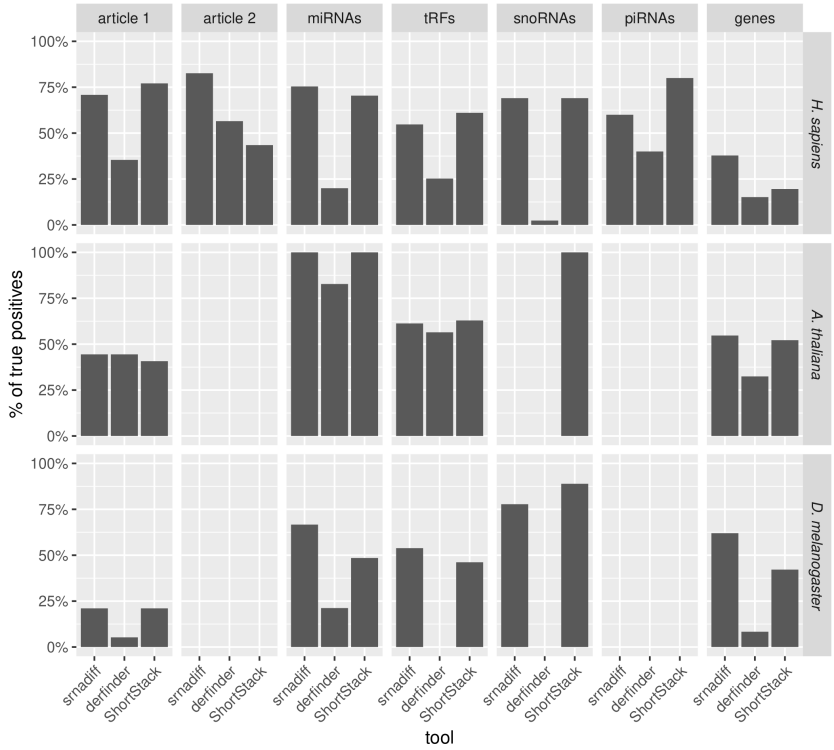



FIGURE 7.4 – Sensibilité des méthodes sur les données réelles.

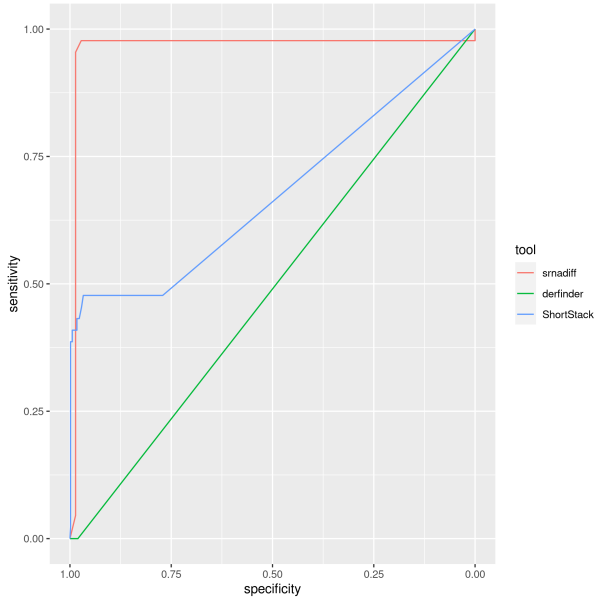



FIGURE 7.5 – Courbes ROC appliquées sur les résultats des trois méthodes comparées, en utilisant des données simulées.

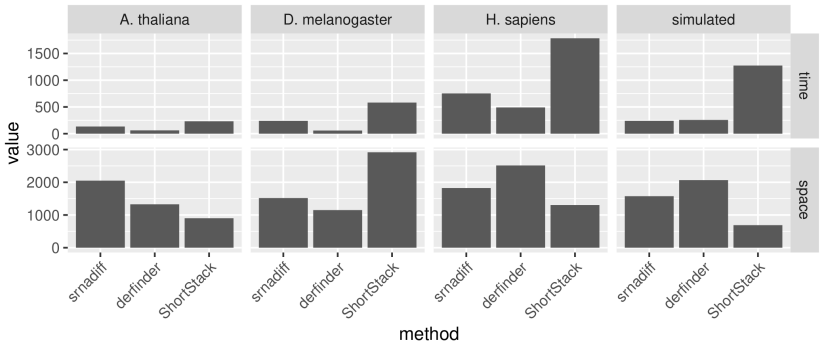



FIGURE 7.6 – Complexité temporelle (en secondes) et spatiale (en Mo) des outils évalués.

# CONCLUSION

Les travaux présentés précédemment sont le fruit des analyses de données que j'ai eu plaisir à mener avec des collègues biologistes. Il m'a semblé que les outils, développés pour les ARN messagers, n'étaient pas toujours parfaitement adaptés aux petits ARN. Plus précisément, les outils classiques permettent d'obtenir des résultats, mais ils ne sont pas optimaux. J'ai ainsi pensé qu'il fallait trouver des méthodes adaptées pour traiter correctement les petits ARN multi-alignés, ce qui constitue une de leurs caractéristiques majeures. Ne pas prendre en compte ces petits ARN dupliqués dans le génome peut fausser une analyse. Et il est dommage que ce soit une pratique courante.

Pour autant, les méthodes que j'ai proposées (avec mmannot et surtout mmquant) ne font pas encore l'unanimité. Le fait de regrouper des gènes à la volée, puis d'utiliser ces groupes de gènes en entrée de la recherche de gènes différentiellement exprimés semble heurter les habitudes. Malgré tout, il me semble que le résultat de l'analyse, à savoir « le petit ARN A, ou B, ou C » est différentiellement exprimé, est l'unique interprétation possible d'une analyse de sRNA-Seq où les gènes sont dupliqués.

Un autre aspect spécifique aux petits ARN est leur taille, et le fait que les séquences sont fortements dupliquées. Il me semble que cet aspect n'avait pas été pris en compte dans les outils d'alignement. Sur le principe, il est possible de supprimer la redondance avant l'alignement, avec une série de scripts *ad hoc*, ce qui est souvent fait en pratique par les pipelines d'analyse de petits ARN. Toutefois, la qualité des lectures est en général perdue, et ceci prend souvent plus de temps que d'aligner les lectures. Lorsque j'ai développé srnaMapper, j'ai également souhaité exploiter les informations extraites pendant l'alignement d'une lecture pour aligner une lecture presque identique, où l'on ajoute simplement un nucléotide à la fin de la lecture (ce qui arrive souvent, par exemple chez les miARN). Une autre aspect intéressant de l'approche est qu'il réduit l'alignement de lectures en un problème général, celui de la recherche de sous-arbre approchée, qui n'avait, semble-t-il pas été abordé jusqu'alors. Ces aspects sont, de fait, totalement hors de propos pour les lectures d'ARN messagers, où l'on ne s'attend pas à de la redondance (elle est même suspecte), et où les lectures, de plus en plus longues, tiennent difficilement en mémoire.

Une dernière spécificité des petits ARN que j'ai souhaité aborder ici est qu'ils sont souvent mal définis. Même les petits ARN connus sont difficiles à retrouver, car certains n'ont pas de caractéristique claire. Il existe également de nombreux petits ARN inconnus, comme en témoigne le nombre de lectures alignées qui ne colocalisent avec aucune annotation. Enfin, les bases de données sont parcellaires, même pour les espèces non-modèles, et inexistantes pour les autres.

En conséquence, l'analyse différentielle s'arrête souvent aux seuls miARN, relativement bien caractérisés. Rechercher les petits ARN différentiellement exprimés directement dans les lectures, sans passer par l'annotation, m'a semblé être une manière simple de résoudre la difficulté. Il reste toute fois la question de caractériser les petits ARN différentiellement exprimés, mais c'est chose plus aisée dès lors que l'on a significativement réduit le nombre de *loci* à analyser.

J'espère que ces méthodes permettront d'analyser plus finement les petits ARN, qui restent moins bien connus que les ARN messagers. Les outils que j'ai conçus peuvent se voir comme une boîte à outils complète d'analyse, qui peuvent s'enchaîner logiquement. Je suis d'ailleurs en train de concevoir un pipe-line d'analyse de petits ARN autour de ces outils avec NextFlow (Di Tommaso et al., 2017), dans le cadre d'un stage que je co-encadre avec Sarah Djébali.

Que reste-t-il à faire pour l'analyse des petits ARN ?
Beaucoup de choses, assurément.

Je n'ai pas mentionné un travail non publié que j'ai fait autour du clustering de miARN. Il s'agit ici de regrouper les miARN en familles, sur la base de leur similarité dans la graine (une région conservée de 6 à 8 nucléotides dans la partie 5' du miARN). Cet outil peut également retrouver les *isomiR* dans les lectures, c'est-à-dire les miARN qui sont subi une édition, voire tenter de regrouper toutes les lectures données en familles.

Un autre outil que j'aurais également aimé faire est un visualisateur d'expression de miARN. Il s'agit d'un outil simple, qui, sur un précurseur de miARN, crée une image du repliement, et montre où sont les lectures sur ce repliement. Cette visualisation permettrait notamment de montrer si l'on a affaire à un vrai miARN, et serait sans doute d'intérêt pour la communauté.

Toujours au sujet des petits ARN, il me semble qu'il n'existe encore pas d'outil permettant de trouver, dans un contexte différentiel, l'ensemble des miARN qui changent leur profil expression au sein d'un même miARN précurseur. Il est en effet connu que, dans certaines conditions, c'est le miARN mature qui est le plus exprimé. Dans d'autres conditions, c'est le *star*. Il me semble qu'il n'existe toujours pas d'outil permettant de quantifier cette différence d'expression. De même, lorsqu'un miARN de 21 nucléotides (chez les plantes) ou 22 nucléotides (chez les animaux) est très exprimé, on trouve en général des miARN un peu plus courts, et d'autres un peu plus longs. Ces variants sont-ils constants dans toutes les conditions ? Peut-on observer un changement ?

Il existe certainement beaucoup d'autres questions que les biologistes se posent, et qui n'ont pas trouvé de réponse en bio-informatique…

# PERSPECTIVE : L'ASSEMBLAGE

Il est tout aussi difficile de faire un résumé cohérent des activités passées que des activités futures. Par souci de clarté, je ne parlerai que d'un projet, celui de l'assemblage.

Toutefois, il existe d'autres thématiques sur lesquelles je souhaite continuer de travailler, avec un investissement moins important.

ÉPIGÉNÉTIQUE DU CERVEAU J'ai déjà collaboré avec des collègues Canadiens sur une protéine, *MeCP2*, notamment responsable du syndrome de Rett, une maladie génétique rare, mais très sévère. Cette protéine possède deux isoformes, qui ont toutes deux la possibilité de se fixer sur la chromatine. Sur le modèle murin, les collègues ont réalisé des ChIP-Seq de cerveaux prélevés à minuit (période d'éveil des souris) et à midi (période de repos). Les analyses que j'ai menées montrent tout d'abord la dynamique de l'expression, mais aussi la spécialisation des isoformes. Elles ont également permis d'identifier des cibles intéressantes, notamment des microtubules, qui pourraient être une piste de compréhension de la maladie (Martínez de Paz et al., 2019).

Je vais continuer ma collaboration avec l'équipe autour de l'analyse d'autres données de ChIP-Seq sur MacroH2A (mH2A), un variant de l'histone H2A, et plus spécifiquement les isoformes mH2A.1 et mH2A.2. Parmi ses fonctions connues, le variant a été identifié pour modeler la chromatine lors du développement des cerveaux des mammifères. Son absence entraînerait des maladies similaires au syndrome de Rett, ou à l'autisme. Dans ce projet en cours, je me propose de d'analyser des données ChIP-Seq du variants d'histone, ainsi que le RNA-Seq, pré- et post-naissance, dans le cortex et le cervelet de souris.

Ces analyses ne nécessitent pas, pour l'instant, de développement particulier, mais le domaine d'application me semble fascinant, et m'entraîneront peut-être vers d'autres sujets d'analyse.

HI-C Le Hi-C a été développé afin de connaître la structure tridimensionnelle de la chromatine. Le protocole est complexe, mais du point de vue informatique, une lecture qui a passé tous les filtres qualité informe d'un contact, c'est-à-dire qu'une paire de *loci* donnée sont très proches l'un de l'autre dans le noyau. Les interactions que l'on observe sont très diverses. On distingue les interactions très localisées, que l'on appelle des boucles. Les TAD (*topologically associated domains*) séparent le génome en domaines, dont chacun a type de régulation qui lui est propre. Enfin, on distingue les compartiments, qui sont de vastes zones ouvertes et exprimées (compartiments de type A) ou fermées et réprimées (type B).

Les données étant très bruitées (on observe beaucoup beaucoup de contacts qui semblent des faux positifs), il est nécessaire de les traiter et les filtrer afin de révéler le signal pertinent. *In fine*, ces données doivent être mises en lien avec d'autres types de données, comme le RNA-Seq : une prédiction des compartiments doit révéler que les gènes dans les compartiments A sont plus exprimés que les gènes des compartiments B.

J'ai donc contribué à analyser ces données dans un travail réalisé avec GenPhySE, autour de la maturation du muscle porcin (Foissac et al., 2019 ; Marti-Marimon et al., 2021). Le but était ici de se confronter aux données, d'extraire les TAD et les compartiments, et les valider par rapport à d'autres sources d'annotation.

Cette étude nous a donné l'idée de démarrer un développement autour de l'analyse de ces données, afin de trouver les régions différentiellement compartimentées, c'est-à-dire passant des compartiments A au B, ou inversement. L'outil que nous sommes en train de développer prend en compte les répliques, et propose une méthode originale afin de trouver les éléments différentiels. Des résultats préliminaires sur nos données nous donnent des résultats encourageants.

PLATE-FORME BIO-INFO GENOTOUL La perspective sans doute la plus marquante est que je vais reprendre, en 2023, l'animation de la plate-forme bio-info GenoToul. Après sollicitation de l'actuelle animatrice, j'ai accepté avec plaisir d'approfondir mes liens avec la plate-forme, avec qui je collabore déjà beaucoup. Ceci répond à mon souhaite de m'investir dans le collectif, mais également de trouver de nouveaux projets situés sur le front de science. Toutefois, il est clair que le temps passé à l'animation diminuera mon temps effectif de recherche.

# 8 PERSPECTIVES

## 8.1 INTRODUCTION

Après m'être investi dans les techniques d'analyse de petits ARN, j'ai souhaité me rediriger vers un autre projet, particulièrement ambitieux, qui est l'assemblage de génomes complets.

L'assemblage peut se définir comme la reconstitution d'un génome entier, à partir de lectures, qui peuvent être plus ou moins longues. Jusqu'à il y a peu, ces lectures étaient relativement courtes, ne dépassant pas les 200 nucléotides. Sont maintenant disponibles des lectures longues, appelées PacBio ou ONT, qui contiennent une dizaine de milliers de nucléotides, pour aller jusqu'à cent mille nucléotides environ. Ces lectures ont permis d'améliorer grandement l'assemblage. Toutefois, ces données sont extrêmement bruitées, et contiennent en général plus de 10% d'erreur. Récemment, des lectures, appelées HiFi, sont à la fois relativement longues (plus d'une dizaine de milliers de nucléotides) et sans erreur ou presque. Ce sont ces lectures qui constituent maintenant la référence pour l'assemblage, même si elles restent relativement coûteuses à produire.

Le processus d'assemblage se décompose en général en deux étapes principales : le contigage et le scaffolding. Le contigage consiste en la fusion des lectures, basée sur le chevauchement de celles-ci, de manière à donner des séquences plus longues, appelées contigs. Cette étape s'appuie en général sur un graphe d'assemblage (voir figure 8.1). Les nœuds du graphe représentent en général une séquence, et les arcs, les possibles successeurs (en supposant que l'on lit le génome de la partie 5' vers la partie 3'). Lorsqu'un nœud a plusieurs successeurs, ou plusieurs prédécesseurs, on a affaire à une ambiguïté. Cette ambiguïté peut être de trois ordres. Tout d'abord, il peut s'agir d'une erreur de séquençage. On a donc, dans les lectures, la « vraie » séquence, et une séquence erronée qui cohabitent. Les méthodes de correction de lectures peuvent être utilisées pour cela, mais elles sont coûteuses en temps et en mémoire, et font parfois des erreurs. La deuxième source d'ambiguïté vient de l'hétérozygotie. Dans ce cas, les allèles maternel et paternel sont différents sur un certain nombre de positions. La plupart des outils de contigage ne traite pas correctement les organismes diploïdes, et choisissent en général un allèle au hasard. Le dernier cas d'ambiguïté vient de la présence de larges répétitions. S'il existe deux *loci* distincts qui contiennent exactement la même séquence, on ne peut connaître les séquences qui se trouvent en amont et en aval de la séquence dupliquée. Lorsqu'il fait face à des ambiguïtés qu'il ne sait pas résoudre, l'assembleur préfère en général ne pas fusionner les séquences, et donne les contigs séparément.

En pratique, cette étape n'est pas suffisante pour obtenir des tailles de contig importantes. C'est pourquoi on complète l'assemblage avec une deuxième étape, appelée scaffolding. Cette étape utilise d'autres types de données, qui permettent de définir la proximité relative de chaque contig entre eux. En principe, le scaffolding peut utiliser n'importe quelle donnée permettant de suggérer la disposition, voire la distance, entre deux contigs. Les données *paired ends* ou *mate pairs* ont été les premières à être utilisées. Par construction, elles donnent une bonne idée de la distance relative de deux contigs. Toutefois, elles ne peuvent pas être utilisées pour des contigs distants de plusieurs milliers de nucléotides. Les données *linked read* de type 10X sont produites à partir d'un long fragment d'ADN, appelés molécules. Ces longs fragments sont ensuite clivés, et on ajoute à chaque fragment un identifiant (presque) unique, avant séquençage. Ces données offrent un compromis intéressant, car elles permettent de scaffolder des contigs relativement lointains, avec une relativement bonne estimation de la distance entre contigs. À l'extrémité, le Hi-C permet de scaffolder

FIGURE 8.1 – Exemple de graphe d'assemblage. Dans cette visualisation, chaque trait est un nœud, et le fait que plusieurs traits partent d'un même point indique une ambiguïté, comme dans l'exemple cerclé de rouge.

des contigs très lointains, pour peu qu'ils soient suffisamment longs. C'est la technologie de choix pour avoir des scaffolds qui se rapprochent de la taille d'un chromosome.

Sur le principe, le scaffolding se base également sur un graphe d'assemblage. Il est en général souvent plus simple, dans la mesure où il contient moins de nœuds.

L'assemblage est connu pour être une tâche difficile. D'une part, il faut générer beaucoup de données, et ces données sont bruitées. D'autre part, le but à atteindre est algorithmiquement complexe : les régions répétées rendent difficile l'assemblage de certaines régions. Il est à noter que les espèces d'intérêt agronomique, comme les animaux d'élevage, les céréales, etc. ont des génomes complexes, dépassant pour la plupart le milliard de paires de bases, et culminant à dix-sept milliards.

Par essence, l'assemblage se base beaucoup sur des concepts d'algorithmique du texte, sujet avec lequel je m'étais déjà familiarisé. Outre l'outil d'alignement, srnaMapper, j'avais déjà conçu quelques algorithmes *ad hoc* dans le cadre de ma thèse (ZYTNICKI, 2007). Brièvement, il s'agissait de trouver une hélice (c'est-à-dire une suite continue d'interactions intra-moléculaires) dans une région délimitée donnée. Cette hélice pouvait contenir des mésappariements, ou des nucléotides non-appariés, mais on favorisait également l'hélice la plus longue. J'avais dû mettre au point un algorithme original, basé sur la programmation dynamique.

J'avais également déjà produit un premier assembleur, mais spécifique aux éléments transposables, nommé tedna (ZYTNICKI, AKHUNOV et QUESNEVILLE, 2014). Cet outil se propose de construire des éléments transposables consensus à partir des lectures courtes de séquençage. Cet assembleur détecte tout d'abord les séquences répétées en étudiant la distribution en *k*-mers, puis effectue les tâches habituelles de contigage et de scaffolding. La difficulté réside dans le fait que les séquences sont répétées, et divergentes. Le graphe d'assemblage est donc très enchevêtré, et il faut trouver des méthodes permettant de trouver la séquence majoritaire la plus longue.

Dans ce cadre, le projet « SeqOccIn » s'est présenté comme une chance. Il s'agit d'un projet FEDER à destination des plates-formes, attribué à GenoToul-Bioinfo et GeT-PlaGe. Parmi les divers aspects abordés, il se propose de mettre au point des méthodes optimisées pour l'assemblage de génomes d'intérêt agronomique. Ces espèces n'ont en effet pas toutes reçu l'intérêt des espèces modèles. En conséquence, elles sont moins bien assemblées que les espèces les plus connues, et l'ensemble des variations génétiques n'est pas aussi bien caractérisé. De plus, les lignées étudiées sont complexes : souvent très hétérozygotes, et parfois polyploïdes, ce qui constitue deux verrous méthodologiques supplémentaires.

La question de l'assemblage est poussée par une dynamique locale forte. Tout d'abord, les partenaires industriels souhaitent connaître précisément le génome de leur cultivar/race d'intérêt, voire de leur génome « élite ». Au niveau biotechnologique, l'unité GeT-PlaGe dispose d'une capacité de séquençage très importante, avec des séquenceurs de dernière

génération. Du point de vue ingénierie, les plates-formes GenoToul-bioinfo et Sigenae ont une grande connaissance métier, et ont déjà réalisé plusieurs assemblages complets.

## 8.2 PROJET EN COURS : SCAFFOLDING EN UTILISANT LES DONNÉES 10X

Parmi les méthodes utilisées pour le scaffolding, le 10X est une des plus intéressantes. Toutefois, il existe encore relativement peu d'outils capable d'utiliser efficacement cette technique. Au moment du début du projet, Tigmint (Jackman et al., 2018) et Arcs (Yeo et al., 2017) étaient les deux seuls outils à exploiter ces données. Le premier se propose de détecter les erreurs lors de l'étape de contigage. Lorsque les données de 10X contredisent l'assemblage, Tigmint casse l'assemblage courant. Le second outil est le scaffolder proprement dit.

Avec Andreea Dréau, que j'encadre dans le cadre de son post-doctorat, nous avons remarqué que, sur nos données, les deux outils n'étaient ni très sensibles, ni très spécifiques. Nous sommes donc en train de proposer deux nouveaux outils qui donnent de meilleurs résultats sur nos données, et permettent, dans la majorité des cas, d'obtenir un meilleur scaffolding. Le principe en théorie assez simple. Concernant le découpage des contig, nous calculons plusieurs métriques, par fenêtre de taille constante, sur l'ensemble des données. Les métriques calculées sont la couverture moyenne des molécules, la densité moyenne des lectures par molécule, la longueur moyenne des molécules, ainsi que le nombre de molécules débutant ou finissant dans cette fenêtre. Si, à un endroit sur le contig, les distributions s'éloignent de la distribution attendue, nous coupons le contig.

Concernant le scaffolding, nous détectons les identifiants se situant aux extrémités des contigs. Les identifiants situé sur le corps du contig (c'est-à-dire à la fois à une extrémité, et à l'intérieur du contig) sont éliminés. Si deux contigs partagent un grand nombre d'identifiants communs, ils sont connectés. Cette connexion ne sera utilisée par le scaffolding uniquement s'il n'y a pas d'ambiguïté, c'est-à-dire si le contig gauche ne peut être connecté qu'à un seul contig, et vice-versa. Ceci nous permet d'écarter les fausses liaisons.

## 8.3 PROJET EN DÉMARRAGE : SCAFFOLDING ET L'INTÉGRATION DES DONNÉES

Dans un second temps, je me propose d'utiliser de façon plus efficace l'intégration de données pour le scaffolding. Dans le projet SeqOccIn, nous disposons à la fois de données 10X et Hi-C. Nous avons observé que ces données se contredisaient parfois. En l'occurrence, les données Hi-C pouvaient parfois prédire une cassure, alors que les données 10X suggèrent fortement le contraire. La raison de cette contradiction est souvent difficile à expliquer. Il peut s'agit de la variabilité observée dans les données, ou bien d'un biais technique. En tout état de cause, il est important d'utiliser l'intégralité des données avant d'effectuer une action sur le graphe de scaffolding.

Pour cela, nous nous proposons de développer une méthode intégrative, qui modélise en même temps les données Hi-C, 10X, mais également les données de séquençage de type *long reads*. Cette méthode discrétise tout d'abord l'ensemble des données en *bins* de taille donnée, et compte le nombre de lectures joignant chaque paire de *bins* (voir figure 8.2). Pour obtenir ces comptages, il est nécessaire d'aligner les lectures sur les contigs. Ensuite, nous comptons le nombre de contacts pour les données Hi-C. Pour les données 10X, il s'agit du nombre d'identifiants communs. Nous considérons que c'est le nombre de lectures qui chevauchent deux *bins* différents pour les données de séquençage.

Bien évidemment, les données sont fortement hétérogènes. Les données de séquençage ne donneront des informations que sur des paires de *bins* très proches, alors que les données Hi-C pourront connecter des *bins* très lointains. De plus, chaque type de données a une profondeur différente. On peut s'attendre à une très grande profondeur pour les données

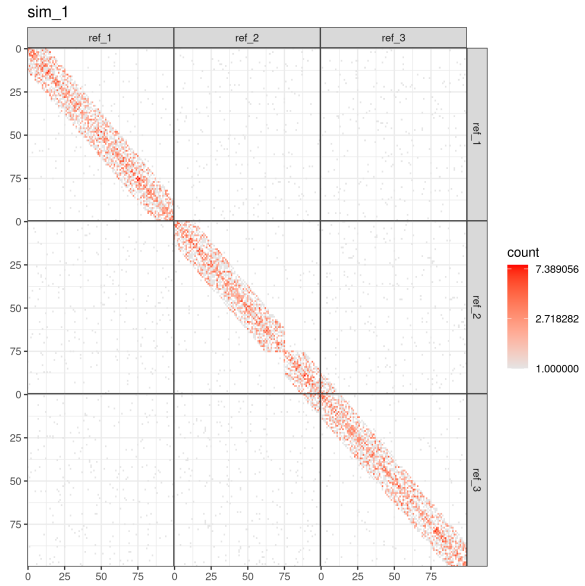



FIGURE 8.2 – Exemple de carte d'interaction sur données simulées. Le génome contient ici trois contigs, qui font chacun 100 *bins*. Chaque point donne le nombre d'interactions entre le *bin* situé en abscisse, et celui donné en ordonnée. Il y a en général plus d'interactions autour de la diagonale. Dans l'exemple, on peut noter une cassure autour du *bin* 75 du deuxième contig, ainsi qu'une connexion entre le deuxième et le troisième contig.

de séquençage, et à une profondeur moindre pour les données Hi-C. Il est donc impossible de fusionner les données.

L'idée est ici d'estimer les paramètres des distributions de comptage pour chaque type de données. Tout d'abord, on peut remarquer que, aux points où il faut couper un contig, la distribution des comptages n'est pas celle attendue. En effet, on peut observer comme une sorte de triangle blanc qui entrerait dans la diagonale (voir le deuxième contig de la figure 8.2). Afin de trouver ces points de découpe, nous estimons donc, sur toute la diagonale, la distribution de comptage de ces triangles. Nous calculons ensuite la différence entre la distribution en un point et la distribution globale, que nous utiliserons comme statistique. Les points où la différence est la plus affirmée (un seuil adéquat est encore à déterminer) sont alors retenus.

Nous avons alors un ensemble de points potentiels, chacun obtenu par une technologie, qu'il s'agit de réconcilier. Il est à noter que, dans l'étape précédente, nous avons calculé la statistique à chaque *bin* de chaque contig. On vérifie donc la valeur de la statistique des autres technologiques au point de cassure potentiel calculé par une technologie donnée. Si la statistique, pour toutes les autres technologies, n'est pas significative, ou va dans le sens de la coupure, alors on procède à la découpe. Sinon, on suppose que l'on avait un faux positif.

On utilise une méthode légèrement similaire pour le scaffolding. Pour chaque type de données, nous estimons donc la distribution de distance des interactions, par rapport à la distribution du nombre de comptages, appelée en général *MD plot*. On compare ensuite, pour chaque coin de chaque paire de contig, la distribution des comptages par rapport à la distribution précédemment calculée (voir les deuxième et troisième contigs de la figure 8.2). Si la différence est proche de zéro, on propose une fusion. Cette fusion est ensuite comparée aux autres fusions comparées par les autres technologies. Si une autre technologie propose une autre fusion pour l'un ou l'autre des contigs (sur les mêmes extrémités), avec une meilleure statistique, la fusion sera abandonnée.

Des résultats préliminaires semblent indiquer de bons résultats en pratique sur nos données.

## 8.4 PROJET SOUMIS : INTÉGRATION DE DONNÉES SUR LE GRAPHE D'ASSEMBLAGE

Les précédents assembleurs construisaient leur graphe de contigage utilisant les lectures courtes (sans erreur) ou les lectures longues (avec beaucoup d'erreurs). Les graphes étaient très branchés, avec beaucoup d'ambiguïté, notamment à cause des répétitions, ou des erreurs d'assemblage.

À cause de ces erreurs, il est impossible de traverser le graphe de façon exhaustive, car cela mène à une explosion combinatoire. Toutefois, l'utilisation des données HiFi a changé notablement la topologie des graphes de contigage, qui deviennent beaucoup plus simples. Sur nos données SeqOccIn, le graphe de contigage produit par les assembleurs classiques passe de 6000-8000 en utilisant les données ONT, à 1500 en utilisant les données HiFi. Ces nouvelles données permettent également de résoudre des ambiguïtés dues à des répétitions.

Le fait d'avoir des graphes plus simples, et contenant moins d'erreurs, permet de répondre à des questions plus ambitieuses. Nous avons donc déposé un projet en 2020 à l'ANR avec Antoine Limasset (CNRS, Lille) et Thierry Lecroq (Université de Rouen), refusé, puis resoumis en 2021. Il se propose d'améliorer les graphes de contigage en utilisant les données de scaffolding.

La première étape consiste à modéliser finement les distribution des différentes données utilisées pour le scaffolding. Les distributions incluent la longueur des interactions, la distribution de couverture, mais aussi la précision. Ces paramètres seraient estimés sur les données elles-mêmes, et nous permettraient notamment de classer les informations d'interactions en « sûres » ou « peu sûres », mais aussi d'estimer si un contig est homozygote, hétérozygotes, ou une région répétée.

Dans une deuxième étape, nous souhaitons concevoir une nouvelle structure de graphe d'assemblage qui contienne tous les types d'informations : séquençage (lectures courtes ou longues, bruitées ou pas), mais aussi les données de scaffolding. En utilisant les résultats de l'étape précédente, nous pourrons supprimer les données vraisemblablement erronées, et pondérer les liens en fonction de la confiance que l'on a. Ces informations additionnelles nous permettront de résoudre les grandes répétitions, séparer les haplotypes et, quand l'information de séquence est absente, donner une estimation de la distance entre les contigs.

Les scaffolders actuels se basent en général sur des algorithmes glouton et des critères de choix locaux. Notre but est d'utiliser des méthodes exactes qui optimisent un critère global. Ceci permet non seulement de résoudre de façon optimale les informations potentiellement contradictoires entre les différentes technologies, mais également de ne pas utiliser de méthode simplificatrice *ad hoc* comme l'utilisation de *bins*. Quelques articles, comme Gao et al., 2016, ont déjà proposé des méthodes basées sur la programmation par contraintes, ou la programmation linéaire en nombres entiers. Ces méthodes permettent d'obtenir de bons résultats sur des petits problèmes. De plus, les récentes avancées faites autour des solveurs comme Gurobi (Gurobi Optimization, LLC, 2021) ou Toulbar2 (Hurley et al., 2016) montrent qu'il est possible de poser des questions plus difficiles aux solveurs.

Étant donné que l'objectif est relativement ambitieux, nous nous proposons d'abord de commencer par appliquer quelques méthodes heuristiques classiques. Tout d'abord, nous dupliquerons les régions monozygotes, et les connecterons aux régions hétérozygotes adjacentes, de manière à simplifier le graphe. De plus, nous ferons de même sur les régions répétées. Enfin, nous supprimerons les liens vraisemblablement erronés.

Dans une dernière étape, nous allons appliquer notre méthode à plusieurs organismes dont les séquences sont fournies par des collaborateurs. En l'occurrence, nous souhaitons proposer un génome amélioré pour le bovin (diploïde) et *Poa annua* (tétraploïde). Nous essaierons également utiliser nos données sur des données de lymphome, également fournies par des collaborateurs. Le cancer est un cas évidemment plus compliqué, car les néo-répétions sont nombreuses. Ce serait un défi intéressant pour notre outil.

Pour résumer, notre objectif est de proposer un outil exhaustif et exact pour le contigage et le scaffolding, qui intègre plusieurs technologies.

## 8.5 PROJETS EN RÉFLEXION

Une des caractéristiques des technologies de séquençage est qu'elles évoluent très vite, et plutôt par à-coups. Par exemple, les lectures longues ont été annoncées depuis 2008, mais elles ne sont largement utilisées que depuis quelques années seulement. *A contrario*, les lectures longues non bruitées sont arrivées peu après la précédente technologie. Il est extrêmement difficile de se projeter sur le sujet de l'assemblage. Dans ce cadre, il est très difficile de se projeter sur un projet à long terme.

Une des évolutions possibles est la production de lectures très longues, sans bruit, accompagnées par des techniques fiables de type *linked reads*. L'assemblage de génome serait alors grandement simplifié. D'autres questions restent pourtant ouvertes. La première est la modélisation d'une population de génomes, à base d'un graphe (en utilisant notamment les développements présentés précédemment). Les *variation graphs* (Garrison et al., 2018) sont une source d'inspiration intéressante, mais l'extension aux cas polyploïdes est encore à construire. Il ne fait pas de doute que l'ajout de chromosomes homéologues rend le graphe plus complexe. Du point de vue agronomique, la question est d'importance cruciale, car le but de l'assemblage d'un individu est de le comparer aux autres, afin d'identifier des *loci* d'intérêt. Un graphe permet d'effectuer cette comparaison de manière élégante. Il peut, de manière efficace, montrer les régions différentes entre deux individus, mais également entre deux populations. Il permet ainsi d'éviter les étapes de comparaisons génome à génome, qui sont longues, et perdent de l'information, notamment au niveau des régions répétées.

Une autre extension possible est la méta-génomique. Brièvement, il s'agit d'assembler toutes les espèces d'un milieu donné : virus, bactéries, ou eucaryote. Il s'agit d'un sujet complexe, car on a affaire à une population de génomes, plus ou moins distincts (plusieurs souches peuvent cohabiter), avec des occurrences plus ou moins importantes. Il s'agit d'une thématique que je connais moins, mais qui me semble un des sujets difficiles, et donc intéressants, liés à l'assemblage.

*In fine*, le travail sur les séquences a ceci de passionnant que la recherche est sans cesse renouvellée. Chaque avancée, notamment en biotechnologie, qui permet de résoudre un problème (comme l'apparition de lectures longues a permis d'assembler de façon exacte un petit génome haploïde), ouvre une autre question plus complexe (comment assembler parfaitement un grand génome polyploïde). Je suis donc heureux de penser que cette problématique nous fournira de nombreuses questions, toujours plus complexes, durant de longues années.

# BIBLIOGRAPHIE


Affymetrix/Cold Spring Harbor Laboratory ENCODE Transcriptome Project (2009). « Post-transcriptional processing generates a diversity of 5'-modified long and short RNAs ». In : *Nature* 457, p. 1028-1032.

Ahmadi, Athena et al. (2011). « Hobbes: optimized gram-based methods for efficient read alignment ». In : *Nucleic Acids Research* 40, e41.

Akula, N et al. (2014). « RNA-sequencing of the brain transcriptome implicates dysregulation of neuroplasticity, circadian rhythms and GTPase binding in bipolar disorder ». In : *Molecular Psychiatry* 19.11, p. 1179-1185.

Alekseyenko, Alexander V. et Christopher J. Lee (2007). « Nested Containment List (NCList): a new algorithm for accelerating interval query of genome alignment and interval databases ». In : *Bioinformatics* 23, p. 1386-1393.

Allen, Edwards, Zhixin Xie, Adam M Gustafson et al. (2004). « Evolution of microRNA genes by inverted duplication of target gene sequences in *Arabidopsis thaliana* ». In : *Nature Genetics* 36, p. 1282-1290.

Allen, Edwards, Zhixin Xie, Adam M. Gustafson et al. (2005). « microRNA-Directed Phasing during *Trans*-Acting siRNA Biogenesis in Plants ». In : *Cell* 121, p. 207-221.

Anders, Simon, Paul Theodor Pyl et Wolfgang Huber (2015). « HTSeq–a Python framework to work with high-throughput sequencing data ». In : *Bionformatics* 31.2, p. 166-169.

Barrett, T et al. (2013). « NCBI GEO: archive for functional genomics data sets–update ». In : *Nucleic Acids Research* 41, p. D991-995.

Bernard-Valnet, Raphaël et al. (accepté pour publication). « Influenza vaccination induces autoimmunity against orexinergic neurons in a mouse model for narcolepsy ». In : *Brain*.

Blow, Matthew J. et al. (2006). « RNA editing of human microRNAs ». In : *Genome Biology* 7.4, R27.

Boison, Detlev (2006). « Adenosine kinase, epilepsy and stroke: mechanisms and therapies ». In : *Trends in Pharmacological Sciences* 27, p. 652-658.

Borges, Filipe et Robert A. Martienssen (2015). « The expanding world of small RNAs in plants ». In : *Nature Reviews Molecular Cell Biology* 16, p. 727-741.

Brennecke, Julius et al. (2007). « Discrete Small RNA-Generating Loci as Master Regulators of Transposon Activity in Drosophila ». In : *Cell* 128, p. 1089-1103.

Brodersen, Peter et al. (2008). « Widespread Translational Inhibition by Plant miRNAs and siRNAs ». In : *Science* 320, p. 1185-1190.

Bussotti, Giovanni et al. (2011). « BlastR-fast and accurate database searches for non-coding RNAs ». In : *Nucleic Acids Research* 39, p. 6886-6895.

Campo, Sonia et al. (2013). « Identification of a novel microRNA (miRNA) from rice that targets an alternatively spliced transcript of the Nramp6 (Natural resistance-associated macrophage protein 6) gene involved in pathogen resistance ». In : *New Phytologist* 199, p. 212-227.

Charbonnel, Cyril, Adnan Niazi et al. (2017). « The siRNA suppressor RTL1 is redox-regulated through glutathionylation of a conserved cysteine in the double-stranded-RNA-binding domain ». In : *Nucleic Acids Research* 45, p. 11891-11907.

Charbonnel, Cyril, Adnan K. Niazi et al. (2017). « The siRNA suppressor RTL1 is redox-regulated through glutathionylation of a conserved cysteine in the double-stranded-RNA-binding domain ». In : *Nucleic Acids Research* 45.20, p. 11891-11907.

CHENG, H., Y. ZHANG et Y. XU (2019). « BitMapper2: A GPU-Accelerated All-Mapper Based on the Sparse q-Gram Index ». In : *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 16.3, p. 886-897.

COLLADO-TORRES, Leonardo et al. (2017). « Flexible expressed region analysis for RNA-seq with derfinder ». In : *Nucleic Acids Research* 45, e9.

COOPER, Martin et al. (2008). « Virtual arc consistency for weighted CSP ». In : AAAI, p. 6.

DARD-DASCOT, Cloelia et al. (2018). « Systematic comparison of small RNA library preparation protocols for next-generation sequencing ». In : *BMC Genomics* 19, p. 118.

DENNENMOSER, Stefan et al. (2019). « Genome-wide patterns of transposon proliferation in an evolutionary young hybrid fish ». In : *Molecular Ecology* 28.6, p. 1491-1505. DOI : 10.1111/mec.14969.

DHAHBI, Joseph M. et al. (2016). « MicroRNAs circulate in the hemolymph of Drosophila and accumulate relative to tissue microRNAs in an age-dependent manner ». In : *Genomics Insights* 9, GEI.S38147.

DI TOMMASO, Paolo et al. (2017). « Nextflow enables reproducible computational workflows ». In : *Nature Biotechnology* 35, p. 316-319.

DOBIN, Alexander et al. (2013). « STAR: ultrafast universal RNA-seq aligner ». In : *Bioinformatics* 29.1, p. 15-21.

ELVIRA-MATELOT, Emilie et al. (2016). « Arabidopsis RNASE THREE LIKE2 Modulates the Expression of Protein-Coding Genes via 24-Nucleotide Small Interfering RNA-Directed DNA Methylation ». In : *The Plant Cell* 28, p. 406-425.

FOISSAC, Sylvain et al. (2019). « Multi-species annotation of transcriptome and chromatin structure in domesticated animals ». In : *BMC Biology* 17, p. 25.

FREGEAU, Brieana et al. (2017). « De Novo Mutations of *RERE* Cause a Genetic Syndrome with Features that Overlap Those Associated with Proximal 1p36 Deletions ». In : *Trends in Pharmacological Sciences* 98, p. 963-970.

GANDIKOTA, Madhuri et al. (2007). « The miRNA156/157 recognition element in the 3' UTR of the Arabidopsis SBP box gene SPL3 prevents early flowering by translational inhibition in seedlings ». In : *The Plant Journal* 49, p. 683-693.

GAO, Song et al. (2016). « OPERA-LG: efficient and exact scaffolding of large, repeat-rich eukaryotic genomes with performance guarantees ». In : *Genome Biology* 17, p. 102.

GARRISON, Erik et al. (2018). « Variation graph toolkit improves read mapping by representing genetic variation in the reference ». In : *Nature biotechnology* 36, p. 875-879.

GEERING, Andrew D. W. et al. (2014). « Endogenous florendoviruses are major components of plant genomes and hallmarks of virus evolution ». In : *Nature Communications* 5, p. 5269.

GESCHWIND, Daniel H et Genevieve KONOPKA (2012). « Neuroscience: Genes and human brain evolution ». In : *Nature* 486, p. 481-482.

GROSSHANS, Helge et Witold FILIPOWICZ (2008). « The expanding world of small RNAs ». In : *Nature* 451, p. 941-416.

GUROBI OPTIMIZATION, LLC (2021). *Gurobi Optimizer Reference Manual*.

HATELEYA, Shannon et al. (2016). « Transcriptomic response of Drosophila melanogaster pupae developed in hypergravity ». In : *Genomics* 108, p. 158-167.

HERAS, Federico et al. (2005). « Existential arc consistency: getting closer to full arc consistency in weighted CSPs ». In : *IJCAI*.

HUANG, Da Wei, Brad T SHERMAN et Richard A LEMPICKI (2009). « Systematic and integrative analysis of large gene lists using DAVID bioinformatics resources ». In : *Nature Protocols* 4, p. 44-57.

HURLEY, Barry et al. (2016). « Multi-language evaluation of exact solvers in graphical model discrete optimization ». In : *Constraints* 21, p. 413-434.

International Wheat Genome Sequencing Consortium (2014). « A chromosome-based draft sequence of the hexaploid bread wheat (Triticum aestivum) genome ». In : *Science* 345, p. 1251788.

Jackman, Shaun D. et al. (2018). « Tigmint: correcting assembly errors using linked reads from large molecules ». In : *BMC Bioinformatics* 19, p. 393.

Johnson, Nathan R. et al. (2016). « Improved Placement of Multi-mapping Small RNAs ». In : *G3: Genes, Genomes, Genetics* 6, p. 2103-2111.

Jorge, Natasha Andressa Nogueira et al. (2017). « snoRNA and piRNA expression levels modified by tobacco use in women with lung adenocarcinoma ». In : *PLoS ONE* 12, p. 1-18.

Kahles, André, Jonas Behr et Gunnar Rätsch (2015). « MMR: a tool for read multi-mapper resolution ». In : *Bioinformatics* 32, p. 770-772.

Kent, W James et al. (2002). « The human genome browser at UCSC ». In : *Genome Research* 12, p. 996-1006.

Kim, Sang Cheol et al. (2013). « A High-Dimensional, Deep-Sequencing Study of Lung Adenocarcinoma in Female Never-Smokers ». In : *PLoS ONE* 8, p. 1-10.

Kutter, Claudia et al. (2007). « MicroRNA-Mediated Regulation of Stomatal Development in Arabidopsis ». In : *The Plant Cell* 19, p. 2417-2429.

Langmead, Ben et Steven L Salzberg (2012). « Fast gapped-read alignment with Bowtie 2 ». In : *Nature Methods* 9, p. 357-359.

Langmead, Ben, Cole Trapnell et al. (2009). « Ultrafast and memory-efficient alignment of short DNA sequences to the human genome ». In : *Genome Biology* 10, R25.

Leśniewska, Anna et Michał J Okoniewski (2011). « rnaSeqMap: a Bioconductor package for RNA sequencing data exploration ». In : *BMC Bioinformatics* 12, p. 200.

Li, Bo et Colin N. Dewey (2011). « RSEM: accurate transcript quantification from RNA-Seq data with or without a reference genome ». In : *BMC Bioinformatics* 12, p. 323.

Li, Heng et Richard Durbin (2009). « Fast and accurate short read alignment with Burrows–Wheeler transform ». In : *Bioinformatics* 25, p. 1754-1760.

Liao, Yang, Gordon K. Smyth et Wei Shi (2014). « featureCounts: an efficient general purpose program for assigning sequence reads to genomic features ». In : *Bionformatics* 30.7, p. 923-930.

Liu, Xiao-Yan et al. (2015). « Regulation of RAGE splicing by hnRNP A1 and Tra2$\beta$-1 and its potential role in AD pathogenesis ». In : *Journal of Neurochemistry* 133, p. 187-198.

Liu, Xin et al. (2017). « Regulation of mitochondrial biogenesis in erythropoiesis by mTORC1-mediated protein translation ». In : *Nature Cell Biology* 19, p. 626-638.

Losh, Jillian S. et al. (2015). « Interaction between the RNA-dependent ATPase and poly(A) polymerase subunits of the TRAMP complex is mediated by short peptides and important for snoRNA processing ». In : *Nucleic Acids Research* 43, p. 1848-1858.

Love, Michael I, Wolfgang Huber et Simon Anders (2014). « Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2 ». In : *Genome Biology* 15, p. 550.

Marcussen, T. et al. (2014). « Ancient hybridizations among the ancestral genomes of bread wheat ». In : *Science* 345, p. 1250092-1250092.

Marti-Marimon, Maria et al. (2021). « Major Reorganization of Chromosome Conformation During Muscle Development in Pig ». In : *Frontiers in Genetics* 12, p. 1895.

Martínez de Paz, Alexia et al. (2019). « MeCP2-E1 isoform is a dynamically expressed, weakly DNA-bound protein with different protein and DNA interactions compared to MeCP2-E2 ». In : *Epigenetics and Chromatin* 12.

May, Patrick et al. (2013). « The effects of carbon dioxide and temperature on microRNA expression in Arabidopsis development ». In : *Nature Communications* 4, p. 2145.

Molitor, Anne M., David Latrasse, Matthias Zytnicki, Philippe Andrey et al. (2016). « The Arabidopsis hnRNP-Q Protein LIF2 and the PRC1 Subunit LHP1 Function in

Concert to Regulate the Transcription of Stress-Responsive Genes ». In : *The Plant Cell* 28, p. 2197-2211.

MOLITOR, Anne M., David LATRASSE, Matthias ZYTNICKI, Philippe P. ANDREY et al. (2016). « The Arabidopsis hnRNP-Q Protein LIF2 and the PRC1 subunit LHP1 function in concert to regulate the transcription of stress-responsive genes ». In : *The Plant cell* 28.

ØROM, Ulf Andersson et al. (2011). « Long noncoding RNAs with enhancer-like function in human cells ». In : *Cell* 143, p. 46-58.

PARENT, Jean-Sébastien, Vincent JAUVION et al. (2015). « Post-transcriptional gene silencing triggered by sense transgenes involves uncapped antisense RNA and differs from silencing intentionally triggered by antisense transgenes ». In : *Nucleic Acids Research* 43, p. 8464-8475.

PARENT, Jean-Sébastien, Vincent V. JAUVION et al. (2015). « Post-transcriptional gene silencing triggered by sense transgenes involves uncapped antisense RNA and differs from silencing intentionally triggered by antisense transgenes ». In : *Nucleic Acids Research* 43, p. 8464-8475.

PARK, Young Ju et al. (2014). « MicroRNA400-Guided Cleavage of Pentatricopeptide Repeat Protein mRNAs Renders Arabidopsis thaliana More Susceptible to Pathogenic Bacteria and Fungi ». In : *Plant and Cell Physiology* 55, p. 1660-1668.

PATTERSON, Victoria L. et al. (2014). « Neural-Specific Deletion of *Htra2* Causes Cerebellar Neurodegeneration and Defective Processing of Mitochondrial OPA1 ». In : *PLOS ONE* 9, p. 1-24.

PFEIFER, Matthias et al. (2014). « Genome interplay in the grain transcriptome of hexaploid bread wheat ». In : *Science* 345.6194, p. 287-287.

QUINLAN, Aaron R. (2014). « BEDTools: The Swiss-Army Tool for Genome Feature Analysis ». In : *Current Protocols in Bioinformatics* 47, p. 11.12.1-11.12.34.

RAMÍREZ, Fidel et al. (2014). « deepTools: a flexible platform for exploring deep-sequencing data ». In : *Nucleic Acids Research* 42, W187-W191.

RENAUD, Gabriel et al. (2011). « Segtor: Rapid Annotation of Genomic Coordinates and Single Nucleotide Variations Using Segment Trees ». In : *PLOS ONE* 6, p. 1-10.

RICHARDSON, Joel E. (2006). « fjoin: Simple and Efficient Computation of Feature Overlaps ». In : *Journal of Computational Biology* 13, p. 1457-1464.

ROBERT, Christelle et Mick WATSON (2015). « Errors in RNA-Seq quantification affect genes of relevance to human disease ». In : *Genome Biology* 16, p. 177.

ROBINSON, Mark D., Davis J. MCCARTHY et Gordon K. SMYTH (2010). « edgeR: a Bioconductor package for differential expression analysis of digital gene expression data ». In : *Bioinformatics* 26, p. 139-140.

SAKURAI, Takeshi et al. (2011). « Haploinsufficiency of *Gtf2i*, a gene deleted in Williams Syndrome, leads to increases in social interactions ». In : *Autism Research* 4, p. 28-39.

SHAMANDI, Nahid, Matthias ZYTNICKI, Cyril CHARBONNEL, Emilie ELVIRA-MATELOT, Aurore BOCHNAKIAN, Pascale COMELLA, Allison MALLORY et al. (2015). « Plants Encode a General siRNA Suppressor That Is Induced and Suppressed by Viruses ». In : *PLoS Biology* 13, e1002326.

SHAMANDI, Nahid, Matthias ZYTNICKI, Cyril CHARBONNEL, Emilie ELVIRA-MATELOT, Aurore BOCHNAKIAN, Pascale COMELLA, Allison C. MALLORY et al. (2015). « Plants Encode a General siRNA Suppressor That Is Induced and Suppressed by Viruses ». In : *PLOS Biology* 13, p. 1-28.

SIRAGUSA, Enrico, David WEESE et Knut REINERT (2013). « Fast and accurate read mapping with approximate seeds and multiple backtracking ». In : *Nucleic Acids Research* 41.7, e78-e78.

STEFANI, Giovanni et Frank J. SLACK (2008). « Small non-coding RNAs in animal development ». In : *Nature Reviews Molecular Cell Biology* 9, p. 219-230.

Storz, Gisela (2002). « An expanding universe of noncoding RNAs ». In : *Science* 296, p. 1260-1263.

Tang, Yang et al. (2017). « *Arabidopsis* NF-YCs Mediate the Light-Controlled Hypocotyl Elongation via Modulating Histone Acetylation ». In : *Molecular Plant* 10, p. 260-273.

Toffano-Nioche, Claire et al. (2013). « RNA at 92°C ». In : *RNA Biology* 10, p. 1211-1220.

Vaz, Candida et al. (2015). « Deep sequencing of small RNA facilitates tissue and sex associated microRNA discovery in zebrafish ». In : *BMC Genomics* 19, p. 950.

Willing, Eva-Maria et al. (2015). « Genome expansion of Arabis alpina linked with retrotransposition and reduced symmetric DNA methylation ». In : *Nature Plants* 1, p. 14023-14028.

Wodarczyk, Claas et al. (2009). « A Novel Mouse Model Reveals that Polycystin-1 Deficiency in Ependyma and Choroid Plexus Results in Dysfunctional Cilia and Hydrocephalus ». In : *PLOS ONE* 4, p. 1-14.

Yeo, Sarah et al. (2017). « ARCS: scaffolding genome drafts with linked reads ». In : *Bioinformatics* 34, p. 725-731.

Zanni, Vanessa et al. (2013). « Distribution, evolution, and diversity of retrotransposons at the flamenco locus reflect the regulatory properties of piRNA clusters ». In : *Proceedings of the National Academy of Sciences* 110, p. 19842-19847.

Zhang, Haowen et al. (2018). « Fast and efficient short read mapping based on a succinct hash index ». In : *BMC Bioinformatics* 19, p. 92.

Ziemann, M, A Kaspi et A El-Osta (2016). « Evaluation of microRNA alignment techniques ». In : *RNA* 22, p. 1120-1138.

Zytnicki, Matthias (2007). « Localisation d'ARN non-codants par réseaux de contraintes pondérées ». Thèse de doct. Université Paul Sabatier.

— (2017). « mmquant: how to count multi-mapping reads? » In : *BMC Bioinformatics* 18, p. 411.

Zytnicki, Matthias, Eduard Akhunov et Hadi Quesneville (2014). « Tedna: a transposable element de novo assembler ». In : *Bioinformatics* 30, p. 2656-2658.

Zytnicki, Matthias et Christine Gaspin (2020). « mmannot: How to improve small–RNA annotation? » In : *PLOS ONE* 15, p. 1-19.

Zytnicki, Matthias, Christine Gaspin, Simon De Givry et al. (2009). « Bounds arc consistency for weighted CSPs ». In : *Journal of Artificial Intelligence Research* 35, p. 593-621.

Zytnicki, Matthias, Christine Gaspin et Thomas Schiex (2008). « DARN! A weighted constraint solver for RNA motif localization ». In : *Constraints* 13, p. 91-109.

Zytnicki, Matthias et Ignacio González (2021). « Finding differentially expressed sRNA-Seq regions with srnadiff ». In : *PLOS ONE* 16, p. 1-21.

Zytnicki, Matthias, YuFei Luo et Hadi Quesneville (2013). « Efficient comparison of sets of intervals with NC-lists ». In : *Bioinformatics* 29, p. 933-939.

Zytnicki, Matthias et Hadi Quesneville (2011). « S-MART, A Software Toolbox to Aid RNA-seq Data Analysis ». In : *PLOS ONE* 6, p. 1-3.

# *CURRICULUM VITAE*

# Matthias Zytnicki

*Chargé de Recherche*


30 décembre 1980 | matthias.zytnicki@inrae.fr
INRAE MIAT | BP 52627 31326, Castanet-Tolosan
05.61.28.54.93 | https://miat.inra.fr/site/Matthias_ZYTNICKI


## Formation Scientifique

2000 – 2003 | **École Nationale Supérieure d'Informatique et de Mathématique Appliquées de Grenoble**
*2002 : ERASMUS à la Danmarks Tekniske Universitet (Copenhague)*

2003 – 2004 | **École Normale de Cachan**
*DEA : Localisation d'ARN non-codants par réseaux de contraintes pondérées.*
*Encadrants : Christine Gaspin, Thomas Schiex*

2004 – 2007 | **Doctorat de l'Université de Toulouse, Mention Informatique, dans l'Unité Biométrie et Intelligence Artificielle, INRA Toulouse**
*Localisation d'ARN non-codants par réseaux de contraintes pondérées.*
*Encadrants : Christine Gaspin, Thomas Schiex*
*Rapporteurs : Christian Besssière, François Major*
*Examinateurs : Martin Cooper, Alain Denise*

2008 | **Post-doctorat au Centre de Regulació Genòmica (Barcelone)**
*Laboratoire de Cédric Notredame.*
**Qualification aux CNU 27 (informatique) et 65 (biologie cellulaire)**

2009 – 2013 | **Chargé de recherche dans l'Unité de Recherche Génomique Info, INRA Versailles**

2013 – présent | **Chargé de recherche dans l'Unité de Mathématiques et Informatique Appliquées de Toulouse, INRAE Occitanie-Toulouse**

## Contributions à l'animation et à la diffusion de l'information scientifique

### Animation collective

| | |
|---|---|
| 2004 – 2005 | Représentant des membres non-titulaires au Conseil de Service de BIA. |
| 2012 – 2016 | Membre du Conseil du Gestion du département BAP. |
| 2016 – 2020 | Membre du Conseil du Gestion du département MIA. |
| 2018 – 2020 | Animateur du séminaire hebdomadaire de l'unité MIAT. |
| 2015 – présent | Membre du Conseil de Service de l'unité MIAT. |

### Animation de la science

| | |
|---|---|
| 2010 – 2013 | Animateur du groupe « Alignement de lectures de séquençage haut-débit » d'APLIBIO, plate-forme francilienne de bio-informatique. |
| 2019 – présent | Vice-président du bureau de la Société Française de Bioinformatique. |

### Enseignements

| | |
|---|---|
| 2004 – 2007 | Programmation et bureautique en Licence. |
| 2016 – 2019 | Modèles graphiques en Master. |
| 2017 – 2019 | Complexité en Master. |
| 2019 – présent | Introduction à Linux en Master.<br>Analyse de données de séquençage « petits ARN » en Master.<br>Jury du Master 2 « Bioinformatique et biologie des systèmes ». |

## Formations organisées

| | |
|---|---|
| 2011 | Formation d'une journée sur l'assemblage de génomes. |
| 2014 | Formation de 2 jours sur l'analyse de petits ARN. |
| 2011 – présent | Formation annuelle de 5 jours Aviesan–IFB sur l'analyse de données de séquençage de type « variants », « RNA-Seq », « ChIP-Seq », niveau 1. |
| 2015 | Formation de 3 jours sur l'analyse de données RNA-Seq avec la Formation Nationale permanente de l'INRA. |
| 2013 – présent | Formation annuelle de 2 jours sur l'analyse de données RNA-Seq avec la GenoToul-Bioinfo. |
| 2021 | Formation annuelle de 4 jours Aviesan–IFB sur l'analyse de données de séquençage de type « variants », « RNA-Seq », « ChIP-Seq », niveau 2. |
| 2022 | Formation annuelle de 5 jours Aviesan–IFB sur l'analyse de données de séquençage pour l'assemblage de génomes. |

## Comités d'organisation

| | |
|---|---|
| 2016 | Principles and Practice of Constraint Programming |
| 2019 | useR |

## Comités de programme

| | |
|---|---|
| 2008, 2014 | Journées Francophones de Programmation par Contraintes |
| 2016 | European Conference on Artificial Intelligence |
| 2018 | IEEE International Conference on Tools with Artificial Intelligence |
| 2018 | European Conference on Computational Biology |
| 2020 – 2021 | Journées Ouvertes de Biologie, Informatique et Mathématiques |

## Bourses

| | |
|---|---|
| 2010 | Appel à projet département BAP |
| 2009 – 2012 | ANR tripartite TransNet |
| 2011 – 2015 | ANR EpiRNAseIII (responsable de *work package*) |
| 2016 – 2019 | Projet IDEX (Lyon) Antiselfish |

## Relectures

Bioinformatics, BioMed Research International (Hindawi), Giga Science, Molecular Ecology, Nucleic Acids Reasearch, PLOS ONE, World Rabbit Science.

## Post-doctorat encadré

| | |
|---|---|
| 2018 – 2022 | Andreea Dréau : Conception de méthodes d'assemblage avec données hétérogènes. |

## Thèses encadrées

| | |
|---|---|
| 2016 – 2018 | Walid Ben Saoud Benjerri : Conception d'un outil d'alignement de petits ARN (*thèse non achevée*) |
| 2017 – 2021 | Leila Khajavi : Profilage transcriptionnel des lymphocytes T pathogènes dans la narcolepsie de type I |

**Article co-publié :**
Alexia Martínez de Paz, Leila Khajavi, Hélène Martin, Rafael Claveria-Gimeno, Susanne Tom Dieck, Manjinder Cheema, Jose Sanchez-Mut, Malgorzata Moksa, Annaick Carles, Nick Brodie, Taimoor Sheikh, Melissa Freeman, Evgeniy Petrotchenko, Christoph Borchers, Erin Schuman, Matthias Zytnicki, Adrian Velazquez-Campoy, Olga Abian, Martin Hirst, Manel Esteller, John Vincent, Cécile Malnou, Juan Ausió (2019). « MeCP2-E1 isoform is a dynamically expressed, weakly DNA-bound protein with different protein and DNA interactions compared to MeCP2-E2. » In : *Epigenetics and Chromatin*, 12.1.
**Article accepté :**
Raphaël Bernard-Valnet, David Frieser, Xuan-Hung Nguyen, Leila Khajavi, Clémence Quériault, Silvia Melzi, Frederick Masson, Matthias Zytnicki, Abdelhadi Saoudi, Yves Dauvilliers, Christelle Peyron, Jan Bauer, Roland Liblau « Influenza vaccination induces autoimmunity against orexinergic neurons in a mouse model for narcolepsy » In : *Brain*.

**Article en préparation :**
Leila Khajavi, Xuan-Hung Nguyen, Clémence Queriault, Marianne Chabod, Lucie Barateau, Yves Dauvilliers, Matthias Zytnicki, Roland Liblau. « The Transcriptomic Profiling of Blood CD4 and CD8 T-cells. »

## Comités de suivi de thèse

| | |
|---|---|
| 2013 – 2017 | Jérôme Mariette : Apprentissage statistique pour l'intégration de données omiques |
| 2014 – 2018 | Franck Cerutti : Évolution et coévolution des petits ARNs régulateurs et des gènes codants chez les bactéries |
| 2016 – 2019 | Franklin Delehelle : ASGART – Cartographie de novo des duplications segmentaires à l'échelle génomique |
| 2021 – | Cheryn Ali : Integrating omics data |
| 2021 – | Thomas Caetano : Méthodes quantitatives pour l'analyse de données RNA-seq en lien avec l'état de phosphorylation des ARN et la régulation de leur dégradation chez *S. aureus* |

## Stages encadrés

| | |
|---|---|
| 2016 | Aurélien Cottin : Comparaison d'outils d'alignement du sRNA-Seq |
| 2017 | Leila Khajavi : Analyse de données d'épigénétique neuronale |
| 2017 | Francisco Ribeiro : Relaxation lagrangienne et réseaux de fonctions de coût |
| 2019 | Cyril Kurylo : Conception d'un outil de recherche de domaines chromatiques différents. |
| 2020 | Paul Simon (étudiant M1) : Analyse de graphes d'assemblage |
| 2021 | Pierre Guenzi-Tibéri (étudiant M1) : Conception d'un pipe-line d'analyse de petits ARN avec NextFlow. |

## Programmes développés

2008 | DARN : un programme de recherche d'ARN non-codants
`https://sourcesup.renater.fr/projects/darn/`

2011 | S-MART : un outil d'analyse de données (s)RNA-Seq
`http://urgi.versailles.inra.fr/Tools/S-Mart`

2014 | tedna : un assembleur d'éléments transposables
`https://github.com/mzytnicki/tedna`

2016 | mmquant : un outil de quantification de données (s)RNA-Seq
`https://bitbucket.org/mzytnicki/multi-mapping-counter`
Également sous BioConda et R BioConductor

2020 | mmannot : un outil de classification de données sRNA-Seq
`https://github.com/mzytnicki/mmannot`

2021 | srnadiff : un outil de recherche de petits ARN
`https://github.com/mzytnicki/srnadiff`

2021 | srnaMapper : un outil d'alignement de petits ARN
`https://github.com/mzytnicki/srnaMapper`

## Vulgarisation

2018 | Sciences ouvertes : l'intelligence artificielle pour la biologie, à destination des lycéens

2019 | Maths en Scène : à destination des collégiens, deux jours

2019 | Carrefour des métiers : lycée

## Publications

Seules sont données ici les publications avec un comité de lecture sélectif. Les ateliers, *workshops*, posters, ne sont donc pas mentionnés.

Allouche, David, Sophie Barbe et al. (2019). « Cost Function Networks to Solve Large Computational Protein Design Problems ». In : *Operations Research and Simulation in healthcare*. Springer, p. 81-102.

Allouche, David, Simon De Givry et al. (2015). « Anytime hybrid best-first search with tree decomposition for weighted CSP ». In : *CP 2015 - 21st International Conference on Principles and Practice of Constraint Programming*. Cork, Ireland, 17 p.

Bussotti, Giovanni et al. (2011). « BlastR-fast and accurate database searches for non-coding RNAs ». In : *Nucleic Acids Research* 39.16, p. 6886-6895.

Campo, Sonia et al. (2013). « Identification of a novel microRNA (miRNA) from rice that targets an alternatively spliced transcript of the Nramp6 (Natural resistance-associated macrophage protein 6) gene involved in pathogen resistance. » In : *New Phytologist* 199.1, p. 212-27.

Charbonnel, Cyril et al. (2017). « The siRNA suppressor RTL1 is redox-regulated through glutathionylation of a conserved cysteine in the double-stranded-RNA-binding domain ». In : *Nucleic Acids Research* 45.20, p. 11891-11907.

Cooper, Martin, Simon De Givry, Marti Sanchez, Thomas Schiex et Matthias Zytnicki (2008). « Virtual arc consistency for weighted CSP ». In : *Twenty-third AAAI Conference on Artificial Intelligence*. Chicago, United States : AAAI - Association for the Advancement of Artificial Intelligence, p. 6.

Cooper, Martin, Simon De Givry, Marti Sanchez, Thomas Schiex, Matthias Zytnicki et Tomas Werner (2010). « Soft arc consistency revisited ». In : *Artificial Intelligence* 174, p. 449-478.

Dennenmoser, Stefan et al. (2019). « Genome-wide patterns of transposon proliferation in an evolutionary young hybrid fish ». In : *Molecular Ecology* 28.6, p. 1491-1505.

Elvira-Matelot, Emilie et al. (2016). « Arabidopsis RNASE THREE LIKE2 Modulates the Expression of Protein-Coding Genes via 24-Nucleotide Small Interfering RNA-Directed DNA Methylation ». In : *The Plant cell* 28.2, p. 406-425.

Foissac, Sylvain et al. (2019). « Multi-species annotation of transcriptome and chromatin structure in domesticated animals ». In : *BMC Biology* 17.1, 25 p.

Gaspin, Christine, Olivier Rué et Matthias Zytnicki (2016). « Ingredients for in silico miRNA identification and annotation ». In : *JSM Biotechnology and Biomedical Engineering* 3.5, p. 1-5.

Geering, Andrew D W et al. (2014). « Endogenous florendoviruses are major components of plant genomes and hallmarks of virus evolution ». In : *Nature Communications* 5.

Heras, Federico et al. (2005). « Existential arc consistency : getting closer to full arc consistency in weighted CSPs ». In : *IJCAI 2005 - International Joint Conference on Artificial Intelligence*. Edimbourg, United Kingdom.

Hurley, Barry et al. (2016). « Multi-language evaluation of exact solvers in graphical model discrete optimization ». In : *Constraints* 21.3, p. 413-434.

International Wheat Genome Sequencing Consortium (2014). « A chromosome-based draft sequence of the hexaploid bread wheat (Triticum aestivum) genome ». In : *Science* 345.6194, p. 1251788.

Marcussen, T. et al. (2014). « Ancient hybridizations among the ancestral genomes of bread wheat ». In : *Science* 345.6194, p. 1250092-1250092.

Marti-Marimon, Maria et al. (2021). « Major Reorganization of Chromosome Conformation During Muscle Development in Pig ». In : *Frontiers in Genetics* 12, p. 1895.
Martínez de Paz, Alexia et al. (2019). « MeCP2-E1 isoform is a dynamically expressed, weakly DNA-bound protein with different protein and DNA interactions compared to MeCP2-E2 ». In : *Epigenetics and Chromatin* 12.1.
Molitor, Anne M. et al. (2016). « The Arabidopsis hnRNP-Q Protein LIF2 and the PRC1 subunit LHP1 function in concert to regulate the transcription of stress-responsive genes ». In : *The Plant cell* 28.9.
Ørom, Ulf Andersson et al. (2010). « Long noncoding RNAs with enhancer-like function in human cells ». In : *Cell* 143.1, p. 46-58.
Parent, Jean-Sébastien et al. (2015). « Post-transcriptional gene silencing triggered by sense transgenes involves uncapped antisense RNA and differs from silencing intentionally triggered by antisense transgenes ». In : *Nucleic Acids Research* 43.17, p. 8464-8475.
Pfeifer, Matthias et al. (2014). « Genome interplay in the grain transcriptome of hexaploid bread wheat ». In : *Science* 345.6194, p. 287-287.
Schbath, Sophie et al. (2012). « Mapping reads on a genomic sequence : an algorithmic overview and a practical comparative analysis ». In : *Journal of Computational Biology* 19.6, p. 796-813.
Shamandi, Nahid et al. (2015). « Plants Encode a General siRNA Suppressor That Is Induced and Suppressed by Viruses ». In : *PLoS Biology* 13.12, e1002326.
Toffano-Nioche, Claire et al. (2013). « Detection of non-coding RNA in bacteria and archaea using the DETR'PROK Galaxy pipeline ». In : *Methods* 63.1, p. 60-65.
Willing, Eva-Maria et al. (2015). « Genome expansion of Arabis alpina linked with retrotransposition and reduced symmetric DNA methylation ». In : *Nature Plants* 1.2, p. 14023-14028.
Zanni, Vanessa et al. (2013). « Distribution, evolution, and diversity of retrotransposons at the flamenco locus reflect the regulatory properties of piRNA clusters ». In : *Proceedings of the National Academy of Sciences of the United States of America* 110, p. 19842-19847.
Zytnicki, Matthias (2017). « mmquant : how to count multi-mapping reads ? » In : *BMC Bioinformatics* 18.1, p. 411.
Zytnicki, Matthias, Eduard Akhunov et Hadi Quesneville (2014). « Tedna : a transposable element de novo assembler ». In : *Bioinformatics* 30, p. 2656-2658.
Zytnicki, Matthias et Christine Gaspin (2020). « mmannot : How to improve small–RNA annotation ? » In : *PLoS ONE* 15.5.
Zytnicki, Matthias, Christine Gaspin et al. (2009). « Bounds arc consistency for weighted CSPs ». In : *Journal of Artificial Intelligence Research* 35, p. 593-621.
Zytnicki, Matthias et Ignacio González (2021). « Finding differentially expressed sRNA-Seq regions with srnadiff ». In : *PLoS ONE* 16.8, e0256196.

Zytnicki, Matthias, Yufei Luo et Hadi Quesneville (2013). « Efficient comparison of sets of intervals with NC-lists ». In : *Bioinformatics* 29.7, p. 933-939.
Zytnicki, Matthias et Hadi Quesneville (2011). « S-MART, a software toolbox to aid RNA-seq data analysis ». In : *PLoS ONE* 6.10, p. 1-3.