De l'intérêt des modèles grammaticaux pour la reconnaissance de motifs dans les séquences génomiques

Aymeric Antoine-Lorquin

ANNÉE 2016

UNIVERSITÉ DE RENNES 1

UNIVERSITE BRETAGNE LOIRE

THÈSE / UNIVERSITÉ DE RENNES 1
sous le sceau de l'Université Bretagne Loire

pour le grade de

DOCTEUR DE L'UNIVERSITÉ DE RENNES 1

Mention : Informatique

Ecole doctorale MATISSE

présentée par

Aymeric Antoine-Lorquin

préparée à l'unité de recherche Inria/Irisa - UMR6074
Institut de Recherche en Informatique et Système Aléatoire
Composante universitaire : ISTIC

De l'intérêt des modèles grammaticaux pour la reconnaissance de motifs dans les séquences génomiques

Thèse soutenue à Rennes
le 1er décembre 2016

devant le jury composé de :

Olivier Ridoux
Professeur à l'Université de Rennes 1
/ *Président de jury*

Hélène Touzet
Directrice de Recherche au CNRS, Lille
/ *Rapporteure*

Jacques van Helden
Professeur à l'Université d'Aix-Marseille
/ *Rapporteur*

Yves Bigot
Directeur de recherche au CNRS, Nouzilly
/ *Examinateur*

Anne Siegel
Directrice de recherche au CNRS, Rennes
/ *Directrice de thèse*

Catherine Belleannée
Maître de conférence à l'Université de Rennes 1
/ *Co-directrice de thèse*

Remerciements

Je remercie Catherine Belleannée et Anne Siegel pour avoir dirigé cette thèse. Merci à Hélène Touzet et Jacques van Helden pour avoir accepté de rapporter cette thèse. Merci à Yves Bigot et Olivier Ridoux pour leur participation au jury.

Je remercie tout particulièrement Catherine Belleannée pour m'avoir guidé tout au long de ces trois années et sans qui cette thèse ne se serait jamais faite. Merci pour ton optimisme forcené, ton implication constante et pour m'avoir fait confiance tout ce temps.

Je remercie aussi Stéphanie Raphäel, sans qui j'aurai probablement fuit la soutenance. Merci pour ton aide inestimable qui m'a permis d'appréhender plus sereinement les prises de paroles en public.

Merci à l'ensemble de l'équipe Symbiose, tant aux anciens et qu'aux actuels, pour ces trois formidables années pleines de bonne humeur. C'était un réel plaisir de travailler dans un tel cadre, aussi bien sur le plan scientifique que personnel. Vous êtes juste géniaux.

Je remercie sincèrement mes différents co-bureaux, Julie Laniau, Charles Bettembourg et Chloé Rioux pour m'avoir supporté, moi et mon éloquence légendaire (un jour, je m'améliorerai, promis). Merci à Jean Coquet et Victorien Delannée pour m'avoir embarqué dans la réalisation d'un court-métrage scientifique, un projet aussi fou qu'enrichissant et que je ne regrette pas d'avoir mené. Merci à Sylvain Prigent, Gaelle Garet, Fanny Ruhland, Coraline Lafon et Sébastien Cordillet pour nos diverses aventures associatives qui sont surtout beaucoup de bons moments passés ensemble. Merci à Guillaume Collet pour avoir beaucoup insisté pour que je mette les pieds dans la "Bashtucada" et merci à tous les batuqueiros qui s'y sont également impliqués pour la faire vivre : toutata tou ta tou ta... ;)

Un grand merci à Anne Siegel qui m'a convaincu de faire de l'enseignement et à Olivier Dameron qui m'a fait confiance pour le remplacer. C'était une expérience très instructives et je suis finalement très heureux d'avoir pu le faire.

Et un très grand merci à Marie Le Roïc, Isobelle Kelly et Marie-Noëlle Georgeault pour leur aide inestimable concernant les méandres des documents administratifs et grâce à qui des drames affreux ont été évités de justesse.

Je tiens aussi à remercier tous mes amis qui m'ont supporté, dans tous les sens du terme, pendant ces trois années et ont particulièrement pris soin de moi durant la dernière ligne droite. Merci pour vos coups de pouce répétés au moral.

Enfin, je tiens surtout à remercier l'ensemble de ma famille, qui m'a toujours soutenu et encouragé tout au long de cette aventure. Un grand merci à mes parents et à leur soutien sans faille. J'arrive enfin au bout de mes études ;)

Et pour terminer, merci aux trop nombreuses personnes que je n'ai pas citées, qui m'ont permis de vivre trois années intenses et inoubliables.

Table des matières

Index

Introduction

La recherche de motifs dans les séquences biologiques est une étape clef dans la compréhension du fonctionnement des gènes et des organismes. Par exemple, si on connait l'enchaînement de nucléotides qu'une protéine utilise pour se fixer à l'ADN, alors on peut prédire la fixation de cette protéine sur différents gènes selon que ceux-ci possèdent ou non cet enchaînement de nucléotides. Ce champ de la bio-informatique qui consiste à rechercher dans les séquences biologiques les instances d'un motif connu est appelé reconnaissance de motifs, mais il est plus souvent désigné sous le vocable anglais *pattern matching*.

Les phénomènes biologiques admettent souvent une certaine dose de variabilité : il ne s'agit donc pas juste de rechercher l'exacte réplique d'une référence connue, mais aussi de considérer des variants plus ou moins ressemblants de cette instance. En outre, les phénomènes biologiques sont bien souvent complexes et ne peuvent pas toujours se résumer à la simple présence d'un mot dans une séquence : des contraintes supplémentaires s'ajoutent, telles que la présence de plusieurs éléments se structurant de façon particulière selon des règles biologiques. De plus, certains des éléments peuvent eux-mêmes être définis en termes de structure et non de contenu, tel que la recherche d'un palindrome (i.e. tige-boucle) ou d'une répétition. Il existe actuellement plusieurs méthodes pour modéliser un motif simple, même variable, telles que, par exemple, l'utilisation de matrices de score, les expressions régulières ou les modèles de Markov cachés. Ces méthodes sont bien connues de la communauté et largement employées. Il n'existe par contre que très peu de méthodes permettant de modéliser des motifs complexes, et il s'agit souvent de méthodes dédiées spécifiquement à une problématique biologique particulière, et non de méthodes génériques.

Pourtant, depuis les années 80, à l'initiative notamment de David Searls [SD93], un certain nombre de travaux a été mené sur les grammaires et sur leur capacité à modéliser des motifs complexes [Sea02, BB84]. Ces travaux montrent qu'en théorie des grammaires de haut niveau offrent suffisamment d'expressivité pour permettre la description de motifs biologiques complexes, notamment par le biais d'une nouvelle classe de grammaire dédiée à la biologie, les grammaires à variables de chaînes (SVG, String Variable Grammar) [Sea95]. Cependant, le concept des SVG n'a donné lieu qu'à des prototypes, rarement maintenus à l'heure actuelle, qui sont majoritairement restés méconnus de la communauté scientifique. L'équipe DYLISS, où a eu lieu cette thèse, a repris le concept des SVG et l'a développé plus avant, avec pour objectif d'offrir à la communauté scientifique un langage formel dont l'expressivité est suffisamment puissante pour permettre d'exprimer un grand nombre des motifs qu'un biologiste pourrait être amené à modéliser en étudiant la réalité des mécanismes biologiques. Le langage s'appelle Logol [BSN14]. C'est un langage de description de motifs d'une grande expressivité. Il est en effet basé sur un formalisme situé au-delà des grammaires algébriques (les grammaires à variables de chaînes), complété d'opérateurs dédiés aux motifs biologiques (permettant par exemple de traduire des notions de répétition ou de composition en "gc") et d'un formalisme de contraintes (permettant par exemple de préciser des taux d'erreurs autorisés sur

certaines parties du motif). Logol est disponible sur la plateforme logicielle Genouest (http ://-logol.genouest.org/web/app.php/logol), avec un accès aux bases de données bioinformatiques.

Ce nouveau langage soulève de nombreuses questions qui ont formé le point de départ de cette thèse. En effet, si les approches grammaticales permettent une grande expressivité dans la description des modèles, cette expressivité est-elle réellement nécessaire pour la recherche des motifs communément rencontrés en biologie ou des alternatives plus classiques sont-elles suffisantes ? Et lorsque cette expressivité est nécessaire pour exprimer des motifs biologiques complexes, l'analyse est-elle réellement possible ? En particulier, peut-on conduire la recherche de tels motifs en des temps raisonnables ? Par extension, comment se positionne la reconnaissance de motifs par grammaire par rapport à d'autres formes de reconnaissance de motifs ? Ces différentes méthodes peuvent-elles exprimer les mêmes choses ou ont-elles des spécificités propres les rendant plus ou moins aptes à gérer certaines catégories de motifs ?

Pour répondre à ce type de questions, nous avons adopté une démarche exploratoire concrète, visant à expérimenter, au travers de plusieurs cas réels, différents genres de modèles recherchés par les biologistes. Cette approche ne pouvant être exhaustive, nous avons diversifié au maximum les problématiques biologiques pour tenter de couvrir un champ d'utilisation représentatif de la réalité. Au cours de cette démarche exploratoire, nous avons parallèlement cherché à comprendre quels étaient les clefs et les verrous d'une "bonne" reconnaissance de motifs, en nous intéressant notamment à la façon de construire un modèle valide et valable, au moyen d'améliorer sa spécificité ou à la façon de faire coopérer différentes méthodes de recherche. Notre approche s'est également faite avec le souci de l'aspect "réaliste" des problématiques rencontrées. Nous avons essayé autant que faire se peut de travailler sur des données réelles, avec les aléas qu'elles peuvent contenir, et de collaborer avec des biologistes spécialistes des problématiques traitées afin de mieux saisir les principaux enjeux des modèles manipulés. Enfin, nous nous sommes attachés à pousser l'exploration et l'analyse suffisamment loin pour, d'une part, tenter d'aboutir à des résultats exploitables pour les biologistes, et d'autre part, en tirer des enseignements sur la reconnaissance de motifs sur des séquences biologiques. Ainsi, six applications ont été étudiées. Elles concernent la reconnaissance de différents types de motifs sur des génomes ou sur des séquences d'ARN. Deux d'entre elles ont été plus poussées et sont décrites dans des chapitres spécifiques (chapitre 3 et 4). L'ensemble des six applications est confronté à la modélisation grammaticale via Logol dans le chapitre 5.

Les chapitres de cette thèse s'organisent de la façon suivante : dans le chapitre 1, nous introduisons des informations de base liées aux séquences biologiques, ainsi qu'à la reconnaissance de motifs et présentons différentes alternatives de reconnaissance de motifs couramment employées en biologie. Nous y présentons ensuite les grammaires et Logol. Dans le chapitre 2, nous comparons les approches matricielle et grammaticale dans le cas d'une recherche d'un motif simple, construit à partir de références. Dans le chapitre 3, nous détaillons en profondeur un premier cas d'étude : la recherche sur le génome des sites de fixation d'un facteur de transcription et la façon de filtrer les faux-positifs pour aboutir à une liste de candidats à une régulation directe par ce facteur de transcription. Dans le chapitre 4, nous montrons comment la recherche des variants mutés d'amorces de PCR peut contribuer à identifier de nouvelles espèces présentes dans des analyses de métagénomique ciblée. Dans le chapitre 5, nous présentons les six applications étudiées dans la thèse, dans un tour d'horizon des avantages et difficultés d'une mise en œuvre de leur motif avec le langage Logol. Enfin, dans le chapitre 6, nous tirons les enseignements de cette étude, et discutons de l'intérêt des grammaires et d'un langage tel que Logol au service de la recherche de motifs dans des séquences biologiques. Le dernier chapitre conclura sur l'apport de cette thèse et abordera les différentes perspectives.

CHAPITRE 1

Reconnaissance de motifs dans les séquences biologiques

Dans ce chapitre, nous présentons les notions utilisées dans cette thèse. Tout d'abord, nous rappellerons le vocabulaire lié à la reconnaissance de motifs. Ensuite, nous détaillerons plusieurs éléments de base liés à la biologie et ce que cela peut impliquer concernant la modélisation. Nous passerons en revue différents outils de reconnaissance de motifs communément employés en biologie, qui illustrent différentes grandes approches en la matière. Nous enchaînerons ensuite sur les modèles grammaticaux, en présentant tout d'abord différents principes qui y sont liés : les notions de grammaire et de langage, les différents niveaux de grammaires et l'expressivité qui en découlent et enfin l'évocation de la façon dont les grammaires peuvent intervenir dans la recherche de motifs biologiques, notamment grâce aux grammaires dédiées que sont les grammaires à variables de chaînes (notées SVG, pour String Variable Grammar). Enfin, nous terminerons par une présentation du langage Logol, un langage grammatical à haut niveau d'expressivité, basé sur les SVG, qui constitue le point de départ et le fil rouge de la thèse.

1.1 Vocabulaire lié à la reconnaissance de motif

Pour commencer, voici quelques définitions du vocabulaire de base utilisé en reconnaissance des motifs [Wol06, EZ11] en biologie. Par ailleurs, plusieurs mots liés à la reconnaissance de motifs peuvent avoir des significations différentes selon les communautés (tel que motif et modèle). Nous donnons ici la signification que nous avons choisie d'associer à ces mots dans la suite de ce manuscrit.

Modèles et motifs

Tout d'abord, les termes motifs et modèles, au cœur de cette thèse, prêtent souvent à confusion selon les communautés et le contexte. Dans ce manuscrit, nous utiliserons le mot **motif** pour désigner la réalité biologique d'une notion ou d'un signal qu'on cherche à identifier sur une séquence. Il peut s'agir d'un signal simple, tel que, par exemple, un site de fixation, qui est un petit segment d'un brin d'ADN où se fixe une protéine ; mais il peut s'agir également d'un signal complexe, tel que, par exemple, le signal de "frameshift-1" (cf page 22), qui indique qu'un ARN est susceptible d'être traduit en deux protéines différentes, suite au glissement de la machinerie de traduction.

Nous utiliserons le mot **modèle** pour désigner la représentation concrète d'un motif dans un langage formel (sous la forme d'une expression régulière, d'une matrice, d'une grammaire Logol...). Ainsi, le modèle d'un site de fixation peut être décrit comme un ensemble de mots

possibles, alors que le modèle d'un motif "frameshift-1" peut être décrit comme la présence sur la séquence de plusieurs signaux successifs (une fenêtre glissante suivie d'une structure en pseudo-noeud etc.).

- **Alphabet** : un alphabet est un ensemble fini de symboles.
  Par exemple, L={A, C, G, T} est l'un des alphabets des séquences d'ADN ;
- **Mot** : un mot défini sur un alphabet est une séquence finie d'éléments de cet alphabet.
  Par exemple, "AACGA" est un mot défini sur l'alphabet L ;
- **Mot vide** : un mot peut ne contenir aucun caractère. Ce mot vide sera représenté ici par le symbole ϵ ;
- **Référence** : une instance déjà connue d'un motif, souvent prouvée biologiquement, utilisée pour définir les spécificités d'un modèle.
  Par exemple, les séquences dont on sait déjà que la protéine P peut s'y fixer constituent les références du motif du site de fixation P ;
- **Modèle** : un ensemble de règles et de contraintes permettant de définir les cibles recherchées lors d'une recherche par reconnaissance de motifs.
  Par exemple, si les références du motif de fixation de la protéine P vérifient une taille de 10, commencent par A et finissent par T, le modèle du site de fixation pourra être le cumul de ces trois règles ;
- **Reconnaissance de motifs ou *Pattern matching*** :
  Reconnaissance exacte de motifs : pour un modèle M et un texte T, localisation de toutes les occurrences de M dans T.
  Reconnaissance approchée de motifs approchée : pour un modèle M, un texte T, une fonction de similarité basée sur une distance d et un paramètre seuil k, localisation des segments I du texte T tels que $d(M, I) \leq k$.
- ***Background*** : proportion d'apparition des différentes lettres dans un texte. Les backgrounds les plus détaillés donne la proportion d'apparition d'une lettre en fonction des *k* lettres précédentes (Chaînes de Markov d'ordre *k* [PP97]). Le background permet de juger de la rareté d'un mot par rapport au hasard en prenant en compte d'éventuels biais de composition des séquences. *Dans un background riche en A, la localisation d'un motif AAATAA est moins significative que dans un background avec une équi-répartition des lettres ;*
- ***Match* ou *Hit*** : une instance dans un texte identifiée par un modèle.
  Par exemple : ATGCGCACGT est une instance reconnue par le modèle du site de fixation de la protéine P : elle commence par A, fini par T et fait une taille de 10 ;
- ***Insertion*** : Ajout d'un élément dans une instance par rapport à une référence.
  *Par exemple : ATG**C**ATG contient une insertion en position 4 par rapport à la référence ATGATG*
- ***Délétion*** : Suppression d'un élément dans une instance par rapport à une référence.
  *Par exemple : ATG**_**TG contient une délétion en position 4 par rapport à la référence ATGATG*
- ***Substitution*** : Remplacement d'un élément par un autre dans une instance par rapport à une référence.
  *Par exemple : ATG**T**TG contient une substitution en position 4 par rapport à la référence ATGATG*
- ***Gap*** : Symbole d'une insertion ou d'une délétion entre une séquence et une référence, notamment lors d'un alignement.
  Par exemple, le hit ATGCA possède une délétion par rapport au motif ATAGCA. Cette délétion est symbolisée par "-" lors de l'alignement entre ATGCA et ATAGCA :

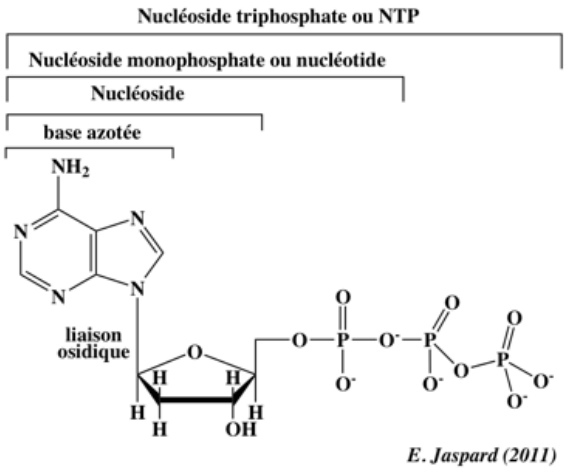


FIGURE 1.1 – Détail d'un nucléoside triphosphate [Jas16]

ATAGCA
AT-GCA

L'expression gap peut aussi désigner un espacement entre deux éléments de motifs.

1.2 Éléments de biologie

Nous allons indiquer dans cette section des éléments biologiques utiles à la compréhension de la recherche de motifs dans les séquences.

1.2.1 Séquence d'ADN

L'ADN (Acide DésoxyriboNucléique) est formé de deux brins s'enroulant en une double-hélice. Les deux brins d'ADN sont de sens opposés : ils sont dits antiparallèles. Un brin d'ADN est formé d'une succession de désoxyribose-phosphates, portant chacun une base azotée. Les bases azotées portées par les désoxyriboses dans l'ADN sont au nombre de 4, réparties en deux catégories : les purines (Adénosine et Guanine, formant respectivement la désoxy-adénosine et la désoxy-guanosine) et les pyrimidines (Cytosine et Thymine, formant respectivement la désoxy-cytidine et la désoxy-thymidine). On trouve une cinquième base azotée présente dans l'ARN : l'Uracyle (formant la désoxy-uridine), qui remplace la Thymine. Les bases azotées sont communément symbolisées par leur initiale : A, C, G ou T. Les désoxyriboses sont liés entre eux de façon asymétrique au niveau de leur structure cyclique. La liaison covalente est établie entre l'atome numéro 3 (3') du cycle d'un désoxyribose avec l'atome numéro 5 (5') du cycle du désoxyribose suivant. Pour cette raison, les brins d'ADN ont donc un sens. Les brins sont synthétisés de l'extrémité 5' vers l'extrémité 3' et sont donc usuellement représentés en respectant ce sens. Chaque base nucléique est capable de former des liaisons hydrogènes (appelées liaisons Watson-Crick) avec une autre base spécifique. L'Adénosine peut se lier ainsi à la Thymine, la Guanine à la Cytosine et respectivement réciproquement. Concrètement, puisque la liaison entre les deux brins est assurée par les liaisons Watson-Crick partagées par les nucléotides en vis-à-vis, cela implique que la séquence d'un brin sens correspond à la séquence inverse et complémentaire du brin antisens. On appelle usuellement cette séquence complémentaire en sens inverse la séquence complémentaire inverse (*reverse complement*). En bioinformatique, seul le brin direct est habituellement représenté, le brin anti-sens pouvant être déduit à partir du brin sens en appliquant le principe de complémentation inverse.

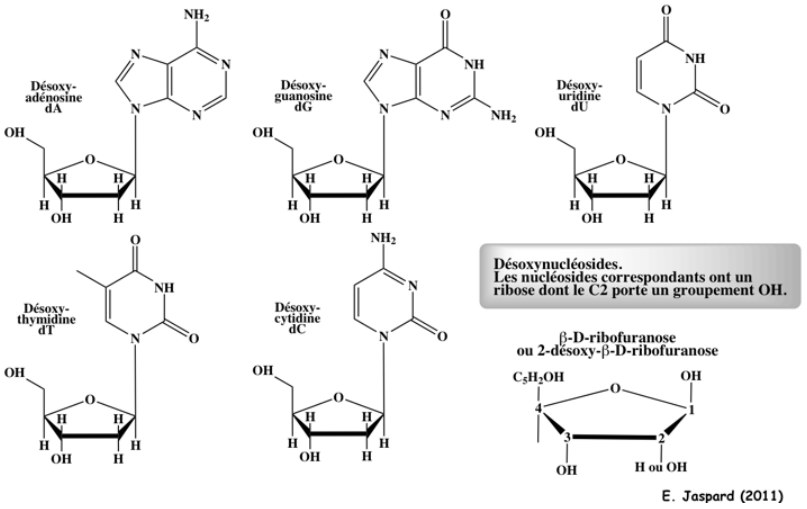


FIGURE 1.2 – Structure des différentes bases azotées [Jas16]

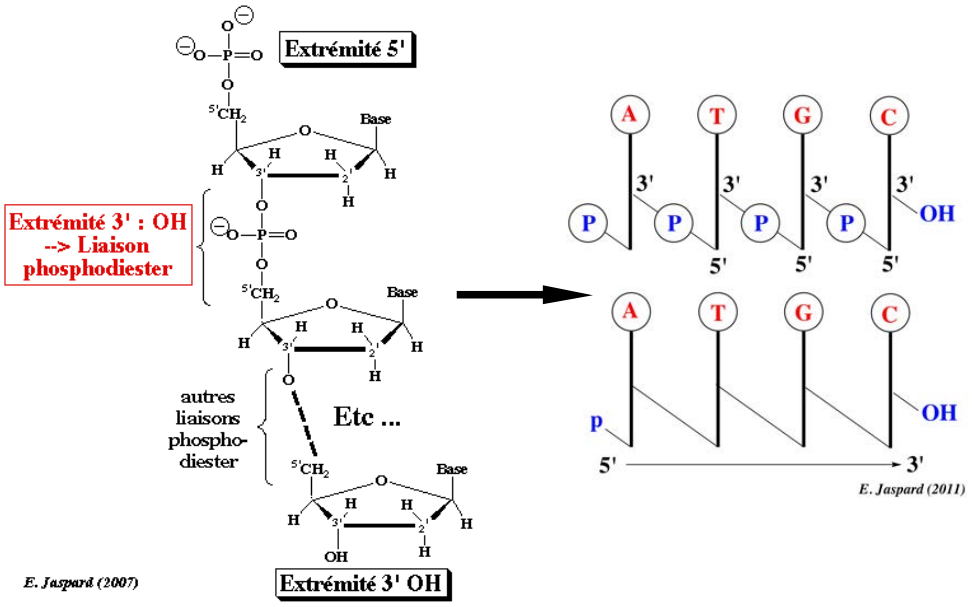


FIGURE 1.3 – Sens de lecture d'un enchaînement de nucléotides [Jas16]

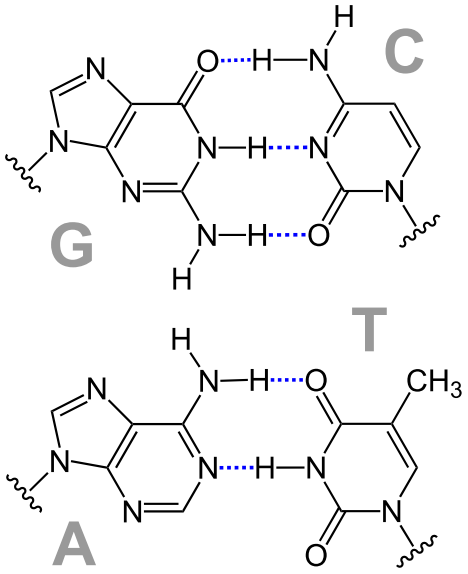


FIGURE 1.4 – Liaison Watson-Crick [Jas16]. Les liaisons G-C comptent davantage de liaisons hydrogènes (3 liaisons) que les liaisons A-T (2 liaisons), ce qui les rend relativement plus solides.

1.2.2 Organismes eucaryotes et procaryotes

Les deux grands types d'organismes cellulaires sont les eucaryotes et les procaryotes. Les génomes de l'humain ou de la levure, utilisés dans cette thèse, sont des génomes eucaryotes. Le génome de la bactérie E. Coli, également utilisé dans cette thèse, est un génome procaryote.

Les cellules eucaryotes sont plus grosses et possèdent la particularité de stocker leur matériel génétique dans une structure appelée le noyau, contrairement aux cellules procaryote. L'une des conséquences de ce cloisonnement réside dans la synthèse des ARN et protéines : chez les cellules eucaryotes, l'ARN est synthétisé dans le noyau puis est exporté dans le cytoplasme pour la synthèse de la protéine. Chez les cellules procaryotes, l'ARN est directement synthétisé dans le cytoplasme.

L'ADN des eucaryotes est globalement plus gros que l'ADN des procaryotes. Les procaryotes ne disposent généralement que d'un chromosome, généralement de forme circulaire. Les eucaryotes disposent de chromosomes linéaires, enroulés autour de protéines histones, contribuant à sa compaction. Les sections d'ADN compactées sont inaccessibles et les gènes qu'elles contiennent ne peuvent s'exprimer tant que la compaction demeure. En raison de cette compaction, à un instant T, l'intégralité de l'information génétique d'une cellule n'est pas complètement accessible.

Dans ce manuscrit, nous avons principalement travaillé sur les eucaryotes. Pour cette raison, les concepts biologiques suivants seront présentés de façon spécifique à ce type d'organisme.

1.2.3 Décodage de l'information génétique chez les eucaryotes

L'information génétique est portée par l'ADN au niveau des gènes. Ces gènes sont transcrits sous forme d'ARN messager (ARNm) dans le noyau avant d'être exportés dans le cytoplasme de la cellule. Les ribosomes présents dans le cytoplasme parcourent les brins d'ARNm et synthétisent les protéines correspondantes. Les protéines assurent une multitude de fonctions au

Caractéristiques	Procaryotes	Eucaryotes
Taille typique d'une cellule	1-10 μm	10-100 μm
Noyau	non	oui
Nombre de chromosomes	généralement 1	>1 (*23 paires chez l'humain*)
Chromosome circulaire	oui	non
Taille du génome	4,64 Méga-paires de base (*E. Coli*)	3 400 Méga-paire de base (*Humain*)
Histones	non	oui
Synthèse des ARN	dans le cytoplasme	dans le noyau
Synthèse des protéines	dans le cytoplasme	dans le cytoplasme

TABLE 1.1 – Propriétés des cellules procaryotes et eucaryotes [Inf]

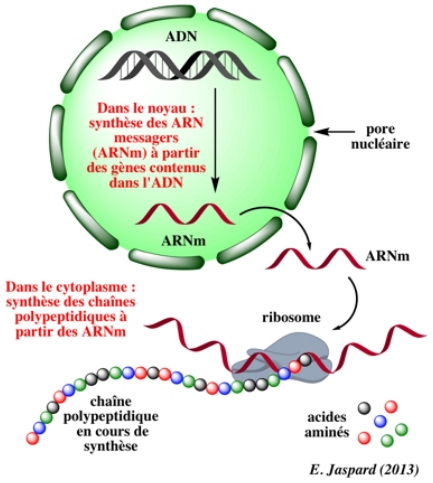


FIGURE 1.5 – Schéma simplifié de la synthèse d'une protéine à partir d'un ADN [Jas16]

sein de la cellule vivante.

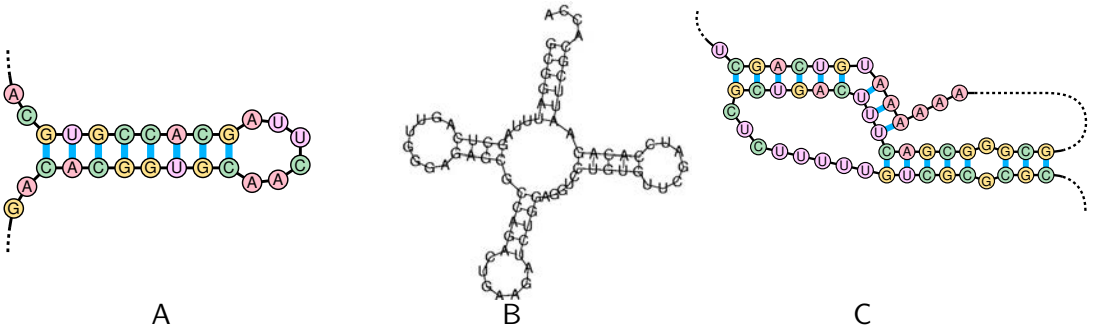
De l'ADN à l'ARNm

Un gène est une séquence de l'ADN susceptible d'être transcrite sous la forme d'un ARN. La partie du gène copiée sous la forme d'un ARN (par transcription), puis transformée en protéine (par traduction) est appelée partie codante.

L'expression d'un gène (la quantité d'ARN produite à partir d'un gène) est soumise à de nombreux facteurs de régulation (accessibilité de l'ADN...). Parmi ces facteurs existe une classe de protéines appelées Facteurs de Transcription (FT), qui ont la capacité de se fixer sur des séquences spécifiques de l'ADN. Ces sites de fixation des facteurs de transcription (TFBS, Transcription Factor Binding Site) sont généralement portés sur la partie de la séquence située en amont du gène (appelée zone upstream).

Un gène codant pour une protéine est constitué de segments pouvant potentiellement composer un ARNm mature, les exons, et de segments éliminés au cours de la maturation de l'ARNm, les introns. Le mécanisme qui permet d'obtenir un ARNm à partir d'un gène est appelé transcription et se déroule en deux temps.

Dans un premier temps, un pré-ARNm est synthétisé à partir de la séquence du gène. Il



A B C

FIGURE 1.6 – Illustration de différentes structures secondaires : A) Structure en tige-boucle B) Structre en feuille de trèfles C) Structure en pseudo-nœud [Jas16]

s'agit d'une réplique exacte, à l'exception de l'utilisation d'Uraciles à la place des Thymines. Ce pré-ARNm contient des exons et des introns.

Dans un second temps, le pré-ARNm subit une étape appelée épissage, au cours de laquelle les introns sont enlevés et les exons sont raboutés entre eux. Il existe plusieurs façons d'épisser un pré-ARNm, notamment en supprimant certains exons en même temps que les introns, qui aboutissent toutes à un ARNm capable de coder pour une protéine. Ce mécanisme d'**épissage alternatif** permet ainsi de synthétiser plusieurs ARNm différents, et donc plusieurs protéines différentes, à partir d'une même séquence génomique.

L'ARNm subit ensuite une étape de maturation, où il reçoit une coiffe et une **queue polyA** (mécanisme de poly-adénylation) pour le protéger et augmenter sa durée de vie.

L'ARNm est simple brin. En raison de la complémentarité entre les bases, l'ARNm est capable d'adopter des structures secondaires. Ainsi, si l'ARNm porte un segment qui est le reverse-complément d'un autre segment de cet ARNm, la séquence va se replier pour que ces deux segments s'apparient de façon anti-parallèle. Cette structure s'appelle une tige-boucle (cf figure 1.6 A).

De l'ARNm à la protéine

La traduction est le mécanisme qui permet la synthèse d'un enchaînement d'acides aminés, la protéine, à partir de la séquence d'un ARNm. Le complexe biologique permettant cette synthèse est le ribosome.

Le ribosome "lit" les nucléotides de l'ARNm par groupes de trois, appelés codons. Chaque codon correspond exactement à un acide aminé spécifique (mais un même acide aminé peut correspondre à plusieurs codons). Le ribosome assemble les acides aminés en fonction de la séquence ARNm et forme une protéine.

Ce découpage de l'ARNm en codons est appelé cadre de lecture ouvert ou ORF (Open Reading Frame). Pour être valide, un ORF doit commencer par un codon start (AUG), qui initie la synthèse d'une protéine, et se terminer par un codon stop (UAA, UAG ou UGA), qui conclut la synthèse d'une protéine.

Frameshift -1 : un mécanisme de recodage au cours de la traduction et un motif complexe

Les motifs Frameshift -1 sont des motifs présents sur certains ARNm, qui permettent de faire glisser le ribosome d'un cran en arrière lors de la synthèse de la protéine. Ce recul entraîne un décalage de phase de l'ORF et modifie le découpage des codons [Bri95]. Cela permet de synthétiser deux protéines différentes à partir d'un même brin d'ARNm. Ce recul a lieu sur

une zone de la séquence appelée "site de glissement", formée d'une structure NNX XXY YYZ, située juste en amont d'une structure secondaire contre laquelle vient butter le ribosome, généralement un pseudo-nœud.

La structure du Frameshift-1 fait intervenir de nombreuses contraintes et donne donc lieu à un motif très complexe : un codon start, un site de glissement positionné à une distance particulière d'une structure en pseudo-noeud et un codon stop positionné sur un cadre de lecture -1 (cf figure 1.8 et 1.9).

1.2.4 Amplification artificielle de l'ADN par PCR

La PCR (*Polymerase Chain Reaction*, réaction en chaîne de la polymérase [CVZ15]) permet d'amplifier une séquence génétique d'intérêt, c'est-à-dire qu'elle permet de multiplier le nombre d'exemplaire de séquences d'intérêt présentes dans un échantillon. Une PCR nécessite les éléments suivants :

- Une séquence ADN contenant le fragment d'intérêt
- Des amorces dites *forward*, correspondant au complémentaire d'un morceau de séquence positionné directement en amont du fragment d'intérêt
- Des amorces dites *reverse*, correspondant à un morceau de séquence positionné directement en aval du fragment d'intérêt
- Une polymérase qui est une protéine chargée de répliquer l'ADN dans les cellules.
- Des désoxyribonucléotides libres, en quantité non limitante

La PCR repose sur un cycle de réplication de l'ADN en trois étapes :

- **1** : La dénaturation. Les liaisons hydrogènes entre les deux brins d'ADN sont brisées par chauffage. Les brins d'ADN sont séparés à l'issu de cette étape.
- **2** : L'hybridation. La température redescend, autorisant de nouveau la formation de liaisons hydrogènes. Les amorces étant complémentaires de la séquence ADN, elles vont s'hybrider ensemble. L'amorce se fixe donc sur la séquence ADN.
- **3** : L'élongation. Le complexe formé par l'amorce hybridée sur la séquence est reconnu par la polymérase, qui utilise les désoxyribonucléotides présents pour allonger la séquence dans le sens 5'-3'.

Ce cycle est répété plusieurs dizaines de fois, aboutissant à l'obtention d'un grand nombre de copies du fragment d'intérêt (cf figure 1.10).

Dans le domaine de la métagénomique, il est possible d'amplifier en même temps plusieurs séquences d'espèces différentes si elles possèdent des amorces communes. Dans ce cas, l'amplification est compétitive : les espèces les plus abondantes sont davantage amplifiées que les espèces les plus rares. Selon le déséquilibre initial, il peut n'y avoir qu'une dizaine de séquences de l'espèce rare au terme de la PCR, contre plusieurs milliers pour les espèces abondantes [CVZ15].

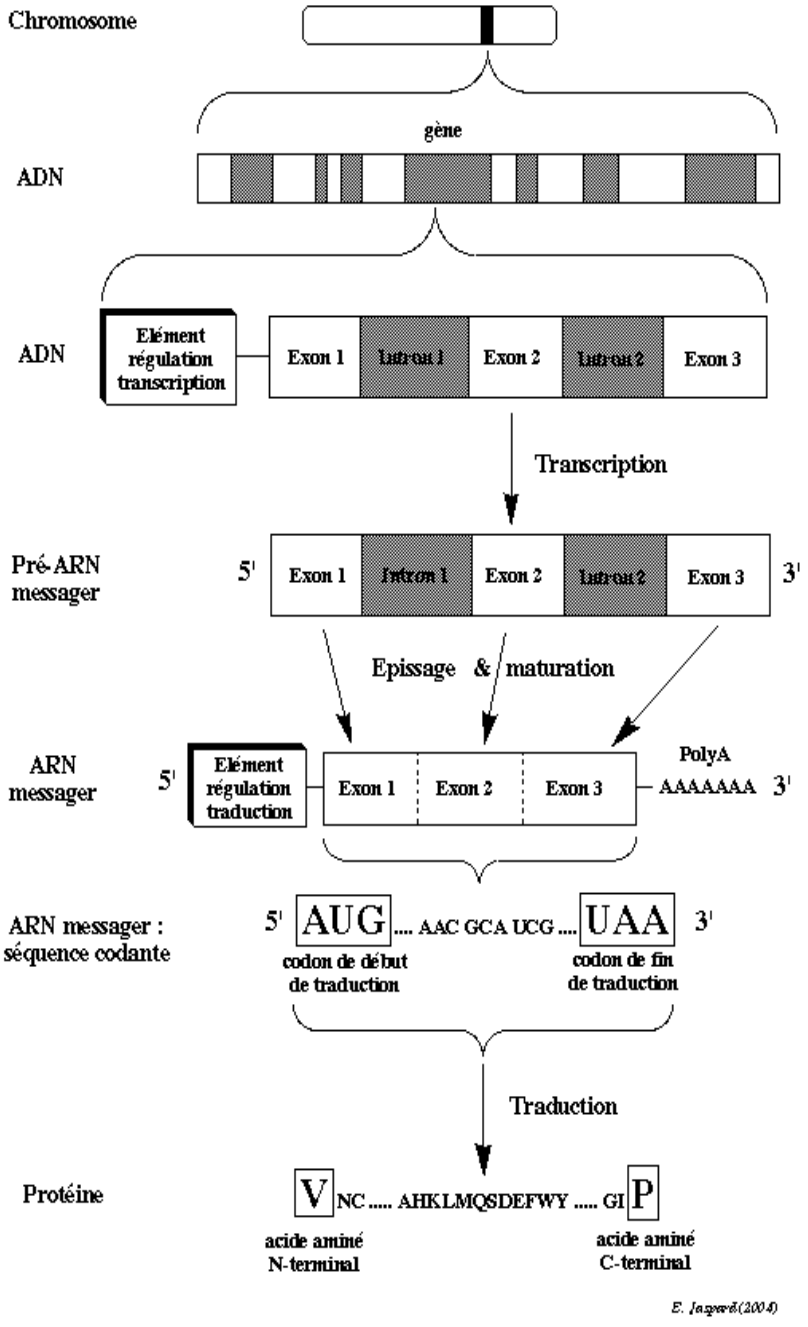


FIGURE 1.7 – Schéma récapitulatif de la synthèse d'une protéine à partir d'un fragment d'ADN (gène) [Jas16]

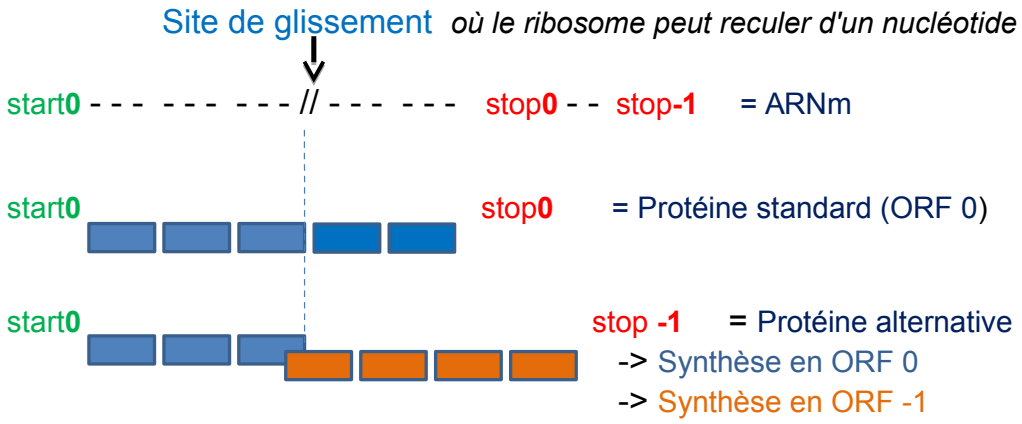


FIGURE 1.8 – Évènement de Frameshift-1 : production d'une protéine alternative.

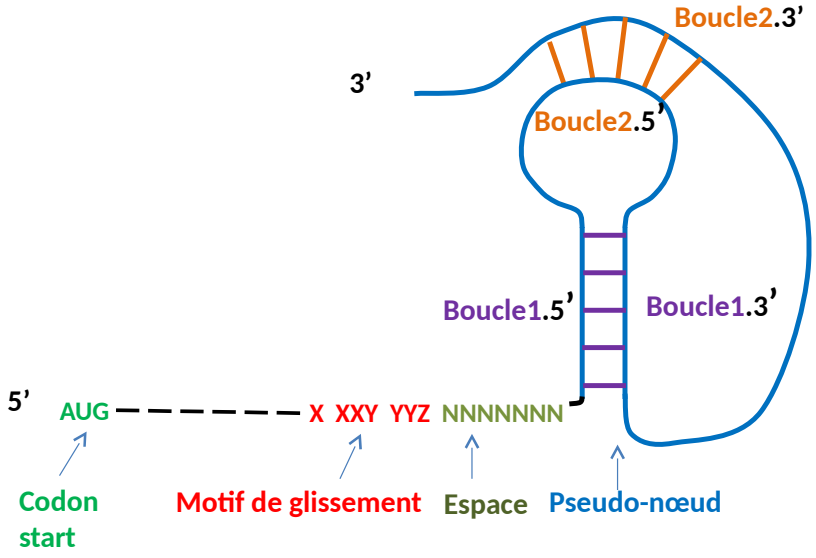


FIGURE 1.9 – Production alternative d'une protéine par l'effet d'une structure frameshift-1

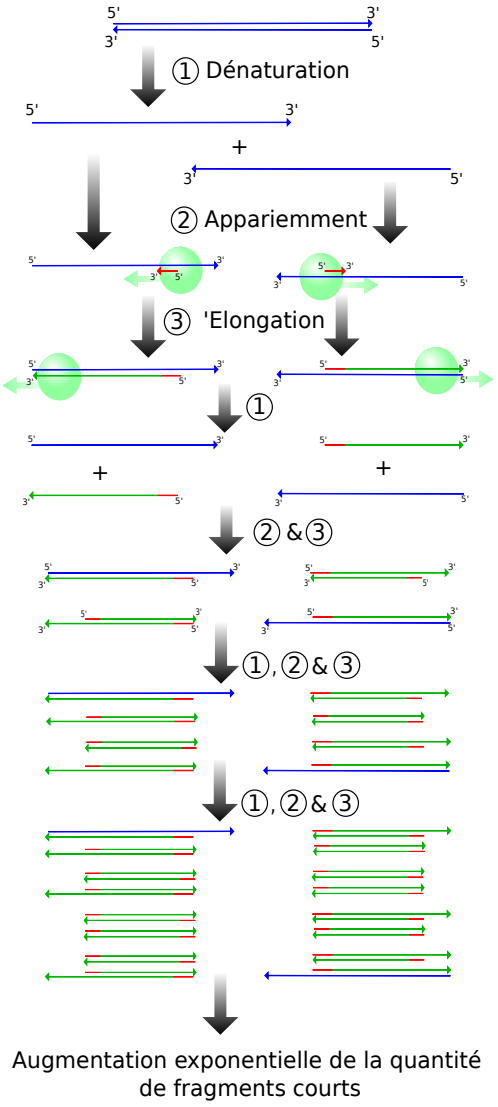


FIGURE 1.10 – Cycles de PCR. Les amorces sont représentées en rouge

1.3 Stratégies standards de reconnaissance de motifs en biologie

Il existe un trop grand nombre d'outils de reconnaissance de motifs, plus ou moins connus et possédant chacun leur spécificité, pour pouvoir les présenter tous en détail. Dans cette section, nous présenterons différentes stratégies de reconnaissance de motifs couramment utilisées par la communauté des biologistes, et auxquelles nous avons été confrontées pendant la thèse.

1.3.1 Recherche de motifs exacts : expressions régulières

Les expressions régulières sont l'une des formes de reconnaissance de motifs les plus basiques et faciles à utiliser. En effet, elles reposent sur un modèle explicite décrit par l'utilisateur. Ainsi, si un motif admet une alternative à certaines positions, la position où cette alternative intervient doit être directement spécifiée dans le modèle. Les opérateurs définis sont, par ordre de priorité croissante, l'union ensembliste (+), la concaténation (. ou rien) et la fermeture de Kleene ou l'étoile (*) [Lie08].

Le champ de recherche d'une expression régulière est donc strictement défini par le concepteur du modèle. Les expressions régulières sont parfaitement adaptées pour la recherche de mots définis, mais la nécessité de définir toutes les possibilités s'avère fastidieuse lorsqu'il s'agit de rechercher des mots approchés (par exemple, rechercher un mot de taille 20 possédant 1 substitution nécessite d'écrire 20 modèles différents, un pour chaque position possible de la substitution).

Enfin, l'expression régulière renvoie une réponse binaire : une instance correspond ou ne correspond pas au modèle.

En biologie, l'un des formalismes les plus connus d'expression régulière est celui proposé par l'outil scanProsite [dCSG+06] de la base de données PROSITE [SCH+02], une base de données sur les protéines pouvant être interrogée via des expressions régulières. Bien que le formalisme puisse différer en fonction du langage utilisé, la logique est souvent la même.

Ci-dessous figure le formalisme de PROSITE :

— Chaque position du modèle est séparée de la suivante par un trait d'union "-".
— Chaque caractère explicite définit la présence de l'acide aminé correspond à exactement cette position du motif.
— Lorsque plusieurs alternatives sont possibles à une même position, celles-ci sont indiquées entre crochets "**{}**".
— La notation "**x(4,7)**" signifie qu'entre 4 et 7 acides aminés peuvent s'intercaler à cette position
— La notation "**x(2)**" signifie qu'exactement 2 acides aminés peuvent s'intercaler à cette position
— La notation "**{GT}**" signifie que tous les acides aminés sont autorisés à cette position, à l'exception des deux acides aminés correspondant aux lettres entre accolades.

Ainsi, l'expression régulière : `A-T-x(1,2)-T` reconnait tous les motifs de taille 4 et 5 commençant par "AT" et finissant par "T".

Recherche d'expressions régulières

Il n'y a pas réellement d'outils de recherche dédiés car les expressions régulières font partie intégrante de nombreux langages de programmation (Bash, Python, ...).

1.3.2 Recherche de motifs par alignement

L'une des méthodes classiques de recherche d'une séquence proche d'une référence consiste à aligner cette séquence contre la référence et constater à quel point cet alignement est possible [Hen96].

Dans un alignement, les insertions et délétions d'une séquence sont présentées sur la séquence opposée par le symbole "-", appelé gap. Les gaps font l'objet d'un traitement spécifique selon les méthodes d'alignement.

Il existe un grand nombre d'algorithmes d'alignement possibles, qui dérivent de trois algorithmes fondamentaux basé sur la programmation dynamique :

Needleman-Wunsh [NW70] : Il s'agit d'un algorithme d'alignement global, c'est-à-dire qu'il essaie d'aligner la séquence contre l'intégralité de la référence. Des scores sont associés pour les différents cas de figure de l'alignement (match exact, insertion, délétion, substitution) et l'algorithme permet de tester toutes les possibilités pour fournir l'alignement avec le score le plus élevé. Le calcul des possibilités se fait via un tableau

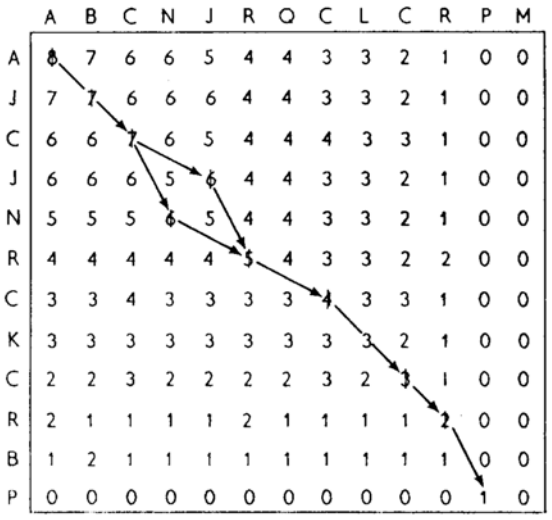


FIGURE 1.11 – Tableau de scores pour alignement de Needleman-Wunsh [SW81]. Un score $S_{i,j}$ est obtenu en ajoutant le score de l'alignement de la position i de la séquence avec la position j de la référence à $max(S_{i-1,j}, S_{i-1,j-1}, S_{i,j-1})$

à deux dimensions (cf figure 1.11) et la solution débute à partir du score le plus haut obtenu sur l'un des côtés opposés au point de départ, fixé en haut à gauche.

Smith-Waterman [SW81] : Il s'agit d'un algorithme d'alignement local, c'est-à-dire qu'il essaie d'obtenir le meilleur alignement possible, même s'il ne concerne qu'une fraction de la séquence ou de la référence. Il reprend la même logique que celle de Needleman-Wunsh, mais débute la solution à la position avec le score le plus haut et remonte l'alignement jusqu'à atteindre une extrémité du tableau ou arriver à un score nul (cf figure 1.12).

Alignement semi-global : Il s'agit d'un algorithme qui donne de meilleurs résultats lorsque les tailles des séquences et de la référence diffèrent. Il est basé sur l'algorithme de Needleman-Wunsh mais ne fixe pas de pénalité au score pour les gaps en amont et en aval de la plus petite séquence. Ceci permet d'aligner localement la totalité de la plus petite séquence (cf figure 1.13).

Analyse de séquences par alignement : BLASTN

BLAST [AGM+90] est l'outil d'alignement le plus connu et utilisé par la communauté des biologistes. Il permet de rechercher des séquences contre différentes bases de données de référence. BLAST permet d'effectuer des alignements entre des séquences et des références de même nature (nucléotides contre nucléotides, protéines contre protéines) mais aussi des alignements hybrides (nucléotides contre protéines et réciproquement). BLAST utilise une heuristique de recherche basée sur le principe de graines. Les séquences sont préalablement fragmentées en k-mers (mots de taille k) chevauchants. BLAST effectue d'abord une recherche des k-mers correspondant entre la séquence et les références pour limiter le nombre de candidats. Pour les séquences disposant de k-mers similaires, BLAST effectue un alignement en allongeant progressivement la séquence autour des k-mers.

	⊿	C	A	G	C	C	U	C	G	C	U	U	A	G
⊿	0·0	0·0	0·0	0·0	0·0	0·0	0·0	0·0	0·0	0·0	0·0	0·0	0·0	0·0
A	0·0	0·0	1·0	0·0	0·0	0·0	0·0	0·0	0·0	0·0	0·0	0·0	1·0	0·0
A	0·0	0·0	1·0	0·7	0·0	0·0	0·0	0·0	0·0	0·0	0·0	0·0	1·0	0·7
U	0·0	0·0	0·0	0·7	0·3	0·0	1·0	0·0	0·0	0·0	1·0	1·0	0·0	0·7
G	0·0	0·0	0·0	1·0	0·3	0·0	0·0	0·7	1·0	0·0	0·0	0·7	0·7	1·0
C	0·0	1·0	0·0	0·0	2·0	1·3	0·3	1·0	0·3	2·0	0·7	0·3	0·3	0·3
C	0·0	1·0	0·7	0·0	1·0	3·0	1·7	1·3	1·0	1·3	1·7	0·3	0·0	0·0
A	0·0	0·0	2·0	0·7	0·3	1·7	2·7	1·3	1·0	0·7	1·0	1·3	1·3	0·0
U	0·0	0·0	0·7	1·7	0·3	1·3	2·7	2·3	1·0	0·7	1·7	2·0	1·0	1·0
U	0·0	0·0	0·3	0·3	1·3	1·0	2·3	2·3	2·0	0·7	1·7	2·7	1·7	1·0
G	0·0	0·0	0·0	1·3	0·0	1·0	1·0	2·0	3·3	2·0	1·7	1·3	2·3	2·7
A	0·0	0·0	1·0	0·0	1·0	0·3	0·7	0·7	2·0	3·0	1·7	1·3	2·3	2·0
C	0·0	1·0	0·0	0·7	1·0	2·0	0·7	1·7	1·7	3·0	2·7	1·3	1·0	2·0
G	0·0	0·0	0·7	1·0	0·3	0·7	1·7	0·3	2·7	1·7	2·7	2·3	1·0	2·0
G	0·0	0·0	0·0	1·7	0·7	0·3	0·3	1·3	1·3	2·3	1·3	2·3	2·0	2·0

FIGURE 1.12 – Tableau de scores pour alignement de Smith-Waterman [SW81]. Le meilleur alignement local est surligné en vert : $GCCUCG$

```
Alignement Semi-global   Alignement global
CAGCA-CTTGGATTCTCGG      CAGCA-CTTGGATTCTCGG
---CAGCGTGG--------      CAGC------G-T----GG
```

FIGURE 1.13 – L'alignement semi-global permet d'identifier plus facilement que l'alignement global des alignements pertinents entre des séquences de tailles différentes

Les résultats fournis par BLAST indiquent à la fois le pourcentage de couverture (à quel point les deux séquences sont chevauchantes) et le pourcentage d'identité (à quel point les séquences sont parfaitement alignées) d'un alignement.

BLAST autorise de nombreux paramètres influant sur les résultats d'alignement, notamment la taille des graines nécessaires pour étendre un alignement, les scores et pénalités en cas d'alignements exacts ou inexacts et les pénalités d'ouverture et de prolongement des gaps. Les paramètres ne permettent pas de spécifier exactement un nombre ou un taux d'erreurs maximum précis pour l'alignement.

Analyse de séquences par alignement : FASTA

FASTA [PL88] est une suite logicielle historiquement apparue avant BLAST. Elle a été initialement conçue pour la comparaison de séquences protéiques, avant de s'ouvrir à la comparaison de séquences nucléotidiques puis de permettre des alignements protéines/nucléotides et réciproquement. À l'heure actuelle, FASTA propose de nombreuses fonctionnalités d'alignements similaires à BLAST.

L'heuristique de FASTA fonctionne sur le même principe que celle de BLAST, c'est-à-dire à partir de graines issues de la séquence.

Analyse de séquences par alignement : CutAdapt

CutAdapt [Mar11] est un outil initialement mis au point pour la recherche et la suppression des adaptateurs dans les données de séquençage haut-débit. Les adaptateurs sont des fragments de séquences qui assurent la fixation de l'ADN séquencé sur son support et ne présentent pas d'intérêt biologique. Néanmoins, l'utilisation de CutAdapt a rapidement dépassé son cadre initial et il est aussi utilisé pour la recherche de petits motifs pouvant contenir des erreurs

(par exemple, la recherche d'amorces de séquençage). CutAdapt est basé sur une variante de l'algorithme d'alignement semi-global.

CutAdapt fonctionne avec des paramètres plus explicite que BLAST. Ces paramètres font notamment intervenir un taux de mutations maximum autorisé (ce taux englobe indifféremment les insertions, les délétions et les substitutions) et un taux minimum de chevauchement entre la référence et le hit, pour éventuellement autoriser les hits à être tronqués à leurs extrémités, sans que cela compte comme des délétions (par exemple, pour une référence de taille 10, un chevauchement de 7 implique que le hit peut ne pas posséder jusqu'à 3 nucléotides à ses extrémités).

Concrètement, CutAdapt renvoie une réponse binaire hit/absence de hit, même s'il est possible de déterminer une distance par rapport à la référence en fonction des mutations présentes dans le hit.

Par exemple, la recherche du modèle "ATGCATAGCA" autorisant 20% d'erreurs (soit 2 erreurs pour une taille de 10) et un chevauchement de 80% (soit 8 positions pour une taille de 10) peut identifier les mots GCAAATCA ("-GCAAATCA" : deux substitutions et un chevauchement de 8/10) ou ATCATGC ("AT-CAT-GC-", 2 délétions et un chevauchement de 9/10).

1.3.3 Recherche de motifs à partir de références

Il est commun qu'un ensemble de références, de mêmes tailles mais présentant de légères variations, soit connu pour décrire un motif biologique. Il existe au moins quatre stratégies possibles pour traiter ce genre de motifs :

Consensus : établir le consensus des références, c'est-à-dire, pour chaque position présentant des choix, ne conserver que le choix majoritaire. Ceci permet d'obtenir une séquence unique, qui peut par exemple servir de référence pour un alignement, mais occulte toutes les subtilités du motif.

Expression régulière : conserver la mémoire de chaque choix, c'est-à-dire établir une expression régulière faisant apparaître toutes les possibilités à chaque position. Dans ce cas, chaque alternative a le même poids, ce qui ne reflète pas exactement la réalité. Par exemple, dans la figure 1.14, la position 14 de l'expression régulière admet un `C` et un `T`, alors que la fréquence du T à cette position est très rare (1 fois sur 13).

Matrice : conserver la mémoire et le poids relatif de chaque choix, par exemple en construisant une matrice [Hen96]. Les matrices permettent de conserver le rapport de force qui peut exister entre les différentes alternatives possibles.

Profil HMM : Les profils HMM permettent de modéliser un motif sous forme de probabilité de transitions entre les positions. Ceci permet d'autoriser des insertions et des délétions ou de prendre en compte des dépendances entre les positions, contrairement aux matrices standards.

Les modèles matriciels ont la particularité de définir un score pour toutes les séquences de même taille que le modèle. Plus la séquence est similaire au consensus de la matrice, plus le score est élevé. Il revient à l'utilisateur de fixer un score seuil de validation, qui déterminera quels sont les matchs valides.

Analyse de séquences par matrice : Matrix-scan de RSAT

La suite logicielle RSAT (Regulatory Sequence Analysis Tools, [MRDS+15]) propose une série de logiciels spécifiquement conçus pour la détection des signaux de régulation dans les séquences non-codantes du génome. Elle permet notamment la découverte de motifs et elle est particulièrement appropriée pour analyser des données plein-génome (c-à-d pour l'analyse de

Id	Sequence
01	TGAACTTGGGTGACCA
02	TGAACTTGAGTGACCA
03	TGACCGGTAGTAACCC
04	TGACCGGTAGTAACCC
05	TGACCTGTGGTAACCT
06	TGACCGGCAGTAACCC
07	CGACCGGGAGTAACCT
08	CGACCGAGAGTAACCT
09	CGACCGAGAGTAACCT
10	TGCCCGCGAGTAACCC
11	TGCCCGACAATGACCC
12	TGACCTCAGGTGATCC
13	TGACCACAGGTAACCT

Consensus IUPAC : `YGMMCDNNRETRAYCH`

Consensus : `TGACCGGAGTAACCC` Expression régulière :

`[CT]G[AC][AC]C[AGT]NN[AG][AG]T[AG]A[CT]C[ACT]`

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
A	0	0	11	2	0	1	3	2	9	1	0	9	13	0	0	2
C	3	0	2	11	13	0	3	2	0	0	0	0	0	12	13	6
G	0	13	0	0	0	8	5	6	4	12	0	4	0	0	0	0
T	10	0	0	0	0	4	2	3	0	0	13	0	0	1	0	5

FIGURE 1.14 – Modélisation sous forme de consensus, d'expression régulière et de matrice de fréquence non normalisée d'un même motif défini par 13 séquences différentes

génomes complets) telles que les pics ChIP-seq (expérimentation permettant d'identifier des zones de fixation d'une protéine d'intérêt sur le génome) ou pour analyser des sites de fixation de facteurs de transcription (évaluation de la qualité, comparaisons et regroupements) ou lors d'études de génomique comparative.

Parmi les nombreux programmes composant RSAT figure le logiciel matrix-scan [TTCDvH08] qui permet d'analyser une séquence avec un modèle exprimé sous forme de matrice.

Matrix-scan présente l'avantage de prendre en compte l'impact du background pour pondérer les scores des matchs obtenus, en autorisant différents types de modèles de background (Bernoulli, chaîne de Markov...). Il propose aussi de nombreuses options avancées permettant par exemple la recherche de régions enrichies en hits sur les séquences analysées.

Analyse de séquences par matrice : FIMO de MEME

MEME (Multiple Em for Motif Elicitation, [BBB$^+$09]) est une autre suite logicielle permettant la découverte et la reconnaissance de motifs dans les séquences, très utilisée par la communauté des biologistes.

Parmi les programmes de la suite MEME figure FIMO (Find Individual Motif Occurrences), qui permet de rechercher des motifs sous forme de matrices dans des séquences. FIMO est capable de prendre en compte des backgrounds nucléotidiques, mais uniquement avec la version en ligne de commande.

Analyse de séquences par HMM : Hmmer

```
>TigeBoucle
NNNNNNNNNNCUGNNNNNNNNN
(((((((((.....)))))))))
```

FIGURE 1.15 – Modèle d'une tige boucle pour Structator

Hmmer [Edd95] est une suite logicielle permettant de rechercher des motifs sous forme de profils HMM (Hidden Markov Model [Edd98]) dans des séquences. Parmi eux, nhmmer [WE13] est spécialisé dans la recherche de motifs nucléiques.

1.3.4 Recherche de motifs structurés

La recherche d'appariements par complémentarité de régions dans les ARN peut se faire d'au moins deux façons : d'une part en appliquant les règles de thermodynamique, c'est-à-dire en mesurant si les liens formés entre les deux régions de la tige sont suffisamment forts en terme d'énergie libre pour maintenir la structure formée, d'autre part, en codifiant les règles qui permettent la formation de structure.

Analyse de séquences par règle de thermodynamique : RNAfold et mfold

RNAfold [LBHZS$^+$11] et mfold [JTZ90] sont deux outils qui recherchent des repliements dans les séquences en s'appuyant sur la thermodynamique.

Tous deux permettent de fixer des paramètres pour la recherche de modèles (température, force ionique, taille des tiges, etc.).

Il n'est pas garanti qu'une structure identifiée pour une molécule ARN par mfold soit exacte, en raison des limitations des modèles, du bruit induit par les paramètres, l'impossibilité de définir des pseudo-noeuds, l'interférence d'autres molécules se complexant à l'ARN ou la possibilité de conformations alternatives des ARN [GG04]. En effet, mfold identifie la structure présentant le minimum d'énergie libre, donc la plus stable possible, sans influence extérieure.

Ces deux logicielles identifient plusieurs structures possibles et les classent par probabilité, proportionnellement au minimum d'énergie libre de chaque structure : la plus basse, donc la plus stable, obtient la probabilité de formation la plus haute.

Analyse de séquences par complémentarité des chaînes : Structator

Structator [MKB$^+$11] est un logiciel qui permet de rechercher des structures secondaires de type tige-boucle décrites sous la forme d'une séquence et d'une structure représentant les interactions sous forme de couples de parenthèses (cf figure 1.15).

La séquence peut contenir un nombre quelconque de nucléotides indéfinis, voir ne contenir aucun nucléotide défini. Des options permettent de définir la présence d'insertion en amont ou en aval de la boucle, ainsi que des insertions en amont et en aval du modèle (mais devant respecter la complémentarité de façon à allonger la tige). Il n'est pas possible de permettre des indels ou des substitutions en d'autres points de la séquence. Par défaut, seuls les liaisons Watson-Crick sont considérées, mais il est possible de spécifier l'ensemble des binômes permettant de vérifier un lien de complémentarité.

1.3.5 Recherche de motifs complexes

Les outils précédemment abordés permettent la recherche de motifs biologiques sous forme de mots, voire de palindromes. Même en incluant la recherche de variants, ce sont des motifs de faible complexité. Il existe d'autres motifs qui font intervenir la recherche de nombreuses propriétés ou de nombreuses contraintes de types différents. Ces motifs ne peuvent généralement pas être modélisés par les outils classiques. Pour palier ce problème, des outils dédiés sont élaborés, spécialisés dans la recherche d'un unique type de motif complexe. Ils procèdent généralement par enchaînement de filtres successifs.

Par exemple, **KnotInFrame** [TRG08] est un outil spécialisé dans la recherche des motifs Frameshift-1 (décrits en section 1.2.3). Cette recherche s'effectue en 3 étapes :

- Phase de recherche : le programme analyse la séquence pour identifier des sites de glissement sur le cadre de lecture 0. La région upstream directement en aval de ce site est analysée pour vérifier sa compatibilité avec une structure en pseudo-noeud. Cette analyse de la structure est faite par une approche thermodynamique (basé sur pknotsRG-mfe [RSG07] et RNAfold [LBHZS$^+$11])
- Phase de filtre : trois critères basés sur l'énergie libre et les contraintes de repliement sont appliqués pour réduire le nombre de candidats
- Phase de classement : les candidats restant sont classés par une fonction d'évaluation basée sur la dominance normalisée de l'énergie du pseudo-nœud.

1.4 Modèles grammaticaux

Les premiers travaux visant à utiliser des langages formels pour décrire des structures du génome datent des années 80 [BB84]. L'idée de base est que la composition d'un génome résulte de nombreuses contraintes variées et structurées entre elles. Ce format structuré et ces différentes contraintes peuvent être modélisés sous la forme d'un langage formel, défini par une grammaire [Sea02]. David Searls a ainsi défini un formalisme grammatical dédié aux motifs biologiques, les *grammaires à variables de chaînes* ou SVG (String Variable Grammars).

Dans un premier temps, nous allons définir ce que sont les grammaires et leurs différents niveaux d'expressivité, puis nous présenterons ce que sont les grammaires SVG et ce qu'elles apportent à la modélisation de motifs biologiques.

1.4.1 Grammaire et hiérarchie de Chomsky

Une grammaire est un formalisme permettant de définir un langage formel, c'est-à-dire un ensemble de mots admissibles sur un alphabet donné. Appliquée à la reconnaissance de motifs, une grammaire définit ainsi un ensemble de mots cibles, qui seront recherchés au sein des séquences analysées [Wol06].

Définition 1. ***Grammaire*** *: une grammaire est définie par un quadruplet (X,V,S,P) où :*

- *X est un alphabet dit terminal*
- *V est un alphabet disjoint de X dit non-terminal*
- *S est une lettre distinguée de V dit axiome*
- *P est un ensemble fini de couples de (X∪V)*V(X∪V)*×(X∪V)* dit règles de production Exemple : G3 = ({a,b},{S,A},S,{S → aS, S → A, A → bA, A → ϵ}).*

Définition 2. ***Dérivation d'une grammaire*** *: un mot u de (X∪V)* se dérive directement en un mot v de (X∪V)* ssi*
$\exists (g,d) \in P \ \exists \ u_1, u_2 \in (X \cup V)^*$ *tels que* $= u_1 g u_2$ *et v* $= u_1 d u_2$.

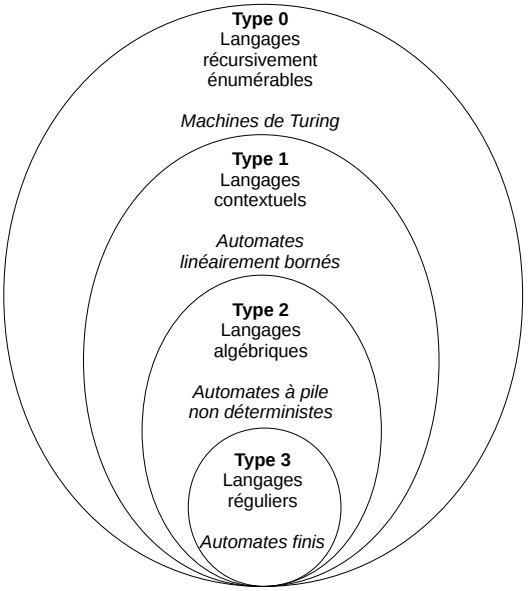


FIGURE 1.16 – Hiérarchie de Chomsky pour les différentes classes de langages. Les classes d'automates associés à chaque langage sont indiquées en italique

On note u $\underset{G}{\rightarrow}$ v cette relation

Exemple : Dans G3, les règles de production de la grammaire autorisent la dérivation de A en bA.

Définition 3. ***Langage engendré*** *: On appelle langage engendré par une grammaire G d'axiome S l'ensemble { f ∈ X* | s $\underset{G}{\rightarrow}$ f } et on le note L(G,S) (ou parfois L(G)).*

Exemple : L(G3) = {"a","ab","aab","aabbbbb", ... }

Le Langage engendré par G3 désigne l'ensemble des mots dérivant de l'axiome S par G3.

Les grammaires permettent de décrire des langages plus ou moins complexes en fonction de la forme de leurs règles. Noam Chomsky a décrit en 1956 [Cho56] une hiérarchie des langages associant la forme de la grammaire à son expressivité. Cette hiérarchie compte quatre échelons de langage, inclusifs, allant du type 3 au type 0, respectivement du plus limité au plus expressif (cf figure 1.16).

1.4.1.1 Grammaire de type 3 : grammaire régulière

Définition 4. ***Grammaire régulière*** *Une grammaire est régulière si toutes ses productions ont une des formes :*

$$A \rightarrow wB$$
$$A \rightarrow w$$

Où A,B ∈ X et w ∈ X. Toute production d'une grammaire régulière a donc un membre de gauche constitué d'un seul non-terminal et un membre de droite constitué d'un mot de symboles terminaux suivi éventuellement d'un seul non-terminal.*

Les grammaires les plus simples sont les grammaires de type 3, ou grammaires régulières. Les grammaires de type 3 sont suffisantes pour décrire par exemple des motifs de type $a^n b^m$ et correspondent aux langages reconnus par des automates d'états finis.

G3=($\{a,b\}$, $\{S,A\}$, S, P), avec P:

$$\begin{cases} S \rightarrow aS \\ S \rightarrow A \\ A \rightarrow bA \\ A \rightarrow \epsilon \end{cases}$$

Exemple de dérivation pour le mot aabbb :
S $\rightarrow$ aS $\rightarrow$ aaS $\rightarrow$ aaA $\rightarrow$ aabA $\rightarrow$ aabbA $\rightarrow$ aabbb

FIGURE 1.17 – Exemple de grammaire de type 3 engendrant le langage $\{a^n b^m; (n,m) \in \mathbb{N}\}$

Les expressions régulières peuvent être exactement écrites sous forme de grammaire régulière.

1.4.1.2 Grammaire de type 2 : grammaire algébrique

Définition 5. ***Grammaire de type 2 ou grammaire algébrique (context-free)*** *: Une grammaire est algébrique si toutes ses productions ont la forme :*

$$A \rightarrow \beta$$

Où A $\in$ X et où il n'y a pas de restriction sur β ($\beta \in$ (X $\cup$ V)). Une grammaire est donc context-free si le membre de gauche de chacune de ses règles de production est constitué d'un seul symbole non-terminal.*

Les grammaires de type 2 se situent au-delà des grammaires de type 3. On les appelle les grammaires algébriques (ou hors-contexte, *context-free* en anglais). Du point de vue de la grammaire, celle-ci autorise la substitution de symboles non-terminaux par des symboles terminaux, indépendamment du contexte entourant ledit symbole. Elles permettent ainsi de décrire par exemple des motifs de type $a^n b^n$ et correspondent aux langages reconnus par des automates à pile.

G2=($\{a,b\}$, $\{S\}$, S, P), avec P:

$$\begin{cases} S \rightarrow aSb \\ S \rightarrow \epsilon \end{cases}$$

Exemple de dérivation pour le mot mot aaabbb :
S $\rightarrow$ aSb $\rightarrow$ aaSbb $\rightarrow$ aaabbb

FIGURE 1.18 – Exemple de grammaire de type 2 engendrant le langage $\{a^n b^m; n \in \mathbb{N}\}$

Ces grammaires sont adaptées pour reconnaître notamment des langages parenthésés. Par exemple, la grammaire Gp = ({(,)},{S},S,{S $\rightarrow$ SS, S $\rightarrow$ (S), S $\rightarrow \epsilon$}) accepte les mots : `(())()` ou `((()()))`. Par contre, elle ne peut pas reconnaître les mots `(()()` ou `())` car la grammaire Gp ne permet pas de former des mots contenant autre chose que des couples de parenthèses inclusifs.

Par ailleurs, les grammaires de type 2 autorisent la reconnaissance de palindromes ($w\tilde{w}$, tel que le mot "serres" avec w :"ser"), mais elles ne permettent pas la reconnaissance de répétitions (ww).

Ainsi le langage $L_{doublet} = \{w \in X^* | \exists m \in X^*, w = mm\}$ n'est pas algébrique.

1.4.1.3 Grammaires de type 1 : grammaire contextuelle (*context-sensitive*)

Définition 6. ***Grammaire contextuelle*** *: Une grammaire est contextuelle (context-sensitive) si toutes ses productions ont la forme :*

$$\alpha \ \rightarrow \ \beta$$

Tel que la condition suivante soit satisfaite :

$$| \alpha | \ \leq \ | \beta |$$

Où il n'y a pas de restriction sur α et β (avec α, $\beta \in$ (X $\cup$ V)). Néanmoins, pour être une grammaire contextuelle, les membres de droite des règles de productions doivent contenir au moins autant de symboles que les membres de gauches. Toutefois, pour permettre aux grammaires de type 1 de générer le mot vide, on permet une exception à la restriction précédente pour la règle de production :*

$$S \ \rightarrow \ \epsilon$$

ssi S n'apparaît pas dans le membre de droite d'une règle de production.

Les grammaires de type 1 se situent au-delà des grammaires de type 2 et 3. Du point de vue de la grammaire, celle-ci autorise la substitution de symbole non-terminaux par différents symboles terminaux en fonction du contexte autour de ce symbole. Elles permettent ainsi de décrire par exemple des motifs de type $a^n b^n c^n$ et correspondent aux langages reconnus par des automates linéairement bornés.

G1=($\{a,b,c\}$, $\{S,A,B,C,D,E\}$, S, P), avec P:

$$\begin{cases} S \to abc \\ S \to aABc \\ S \to \epsilon \\ AB \to AC \\ AC \to BC \\ BC \to BA \\ Ac \to DBcc \\ BD \to BE \\ BE \to EB \\ DE \to DB \\ aD \to aaA \\ aD \to aa \\ B \to b \end{cases}$$

Exemple de dérivation pour le mot aaabbbccc :
S → aABc → aACc → aBCc → aBAc → aBDBcc →
aBEBcc → aDEBcc → aDBBcc → aaABBcc → aaACBcc
→ aaBCBcc → aaBBAcc → aaBBDBccc → aaBBEBccc
→ aaBEBBccc → aaDEBBccc → aaDBBBccc →
aaaBBBccc → aaabBBccc → aaabbBccc →
aaabbbccc

FIGURE 1.19 – Exemple de grammaire de type 1 engendrant le langage $\{a^n b^n c^n; n \in \mathbb{N}\}$

Les grammaires de type 1 sont adaptées pour la reconnaissance de répétitions (ww, cf le langage $L_{doublet}$).

1.4.1.4 Grammaires de type 0

Définition 7. ***Grammaire de type 0*** *: Les grammaires de type 0 n'ont pas de restriction sur les règles de productions.*

Enfin, les grammaires de type 0 rassemblent les grammaires générales, capable de reconnaître les langages récursivement énumérables. Elles permettent par exemple la définition de motifs reconnaissant tous les mots qui contiennent autant de "a" que de "b" et que de "c", quelque soit leur combinaison. Ce sont les langages exactement reconnus par les machine de Turing.

$$\texttt{G0=(}\{a,b,c\}\texttt{, }\{S,A,B,C\}\texttt{, S, P), avec P:}$$
$$\begin{cases} S \to SABC \\ S \to \epsilon \\ AB \to BA \\ BA \to AB \\ AC \to CA \\ CA \to AC \\ BC \to CB \\ CB \to BC \\ A \to a \\ B \to b \\ C \to c \end{cases}$$

```
Exemple de dérivation pour le mot aababcbcc :
S → SABC → SABCABC → SABCABCABC → ABCABCABC →
   ABACBCABC → AABCBCABC → AABCBACBC →
   AABCABCBC → AABACBCBC → AABABCCBC →
   AABABCBCC → aABABCBCC → aaBABCBCC → ... →
   aababcbcc
```

FIGURE 1.20 – Exemple de grammaire de type 0 engendrant le langage $X; |X|_a = |X|_b = |X|_c$

1.4.2 Analyse syntaxique : vérifier si un mot appartient à un langage

L'analyse syntaxique est la procédure qui, pour un mot donné, détermine s'il appartient ou non au langage engendré par une grammaire. Il s'agit donc de construire une suite des dérivations qui permettent d'aboutir au mot en partant de l'axiome.

Il existe deux grandes façons de construire un arbre de dérivation : les méthodes d'analyse descendante (top-down) et les méthodes d'analyse ascendante (bottom-up). Dans le cas des méthodes d'analyse descendante, l'arbre de dérivation est construit à partir de la racine (l'axiome) en allant jusqu'aux feuilles (les non-terminaux). Pour les analyses ascendantes, l'arbre est construit en remontant des feuilles jusqu'à la racine. Les grammaires permettent de construire différents mots regroupés au sein d'un langage. Appliqué à la reconnaissance de motifs, un motif est défini par une grammaire. Une séquence pour laquelle l'analyseur syntaxique vérifie qu'elle appartient au langage défini par la grammaire est une instance du motif.

L'ambiguïté des grammaires

Une grammaire est dite ambigüe si pour un même mot, plusieurs arbres de dérivations sont possibles.

1.4.3 Quel niveau d'expressivité pour des motifs biologiques ? Au-delà des grammaires algébriques : les SVG

Les grammaires engendrent des langages qui définissent une liste de mots. Au moyen de l'analyse syntaxique, il est possible de déterminer si un mot donné appartient au langage d'une grammaire. Par ailleurs, nous avons vu qu'il existe différents types de grammaire, chacune ayant son propre degré d'expressivité. On peut se demander quel type de grammaire employer pour exprimer des motifs biologiques.

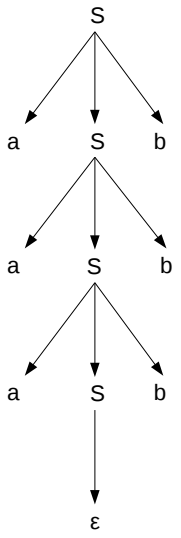


FIGURE 1.21 – Arbre de dérivation par méthode d'analyse syntaxique descendante du mot aaabbb par la grammaire G2

En 1997, dans son article *"Linguistic approaches to biological sequences"* [Sea97], David Searls a résumé l'expressivité permise par les différents types de grammaire et les concepts biologiques qu'elles permettent de modéliser dans le tableau de la figure 1.22. Ainsi, les langages réguliers, issus des grammaires de type 3 et qui se situent au plus bas niveau de la hiérarchie de Chomsky, ont une expressivité très limitée, qui les empêche par exemple de décrire un palindrome. Ceci est dû au fait que les grammaires régulières n'ont pas de notions de "mémoire" et ne peuvent donc pas décrire des dépendances entre différents morceaux de la grammaire.Elles permettent par exemple de décrire la structure globale d'un ORF.

Les langages algébriques, issus des grammaires de type 2, ont une plus grande expressivité que les langages réguliers. L'avantage des grammaires algébriques réside entre autre dans leur capacité à pouvoir créer des dépendances imbriquées (cf 1.18, la dérivation permet d'injecter l'axiome *entre* des éléments non-terminaux). Cette expressivité s'avère suffisante pour modéliser des palindromes : appliqué à la biologie, cela signifie pouvoir chercher des tiges-boucles.

Les langages contextuels, issus des grammaires de type 1, sont encore plus expressifs que les langages algébriques. En effet, l'utilisation des grammaires de type 1 permet de créer des dépendances croisées. Cette particularité permet ainsi la modélisation de répétitions, mais aussi d'imbriquer des palindromes : l'application de ce dernier point à la biologie permet notamment la recherche des pseudo-noeuds, composés de tiges-boucles imbriquées.

Néanmoins, l'expressivité croissante des langages dans la hiérarchie de Chomsky va de paire avec une augmentation de la puissance de calcul nécessaire pour déterminer si un mot appartient ou non au langage spécifié par une grammaire donnée. À ce titre, on constate que si les grammaires contextuelles (de type 1) sont celles qui détiennent une expressivité suffisante pour exprimer l'ensemble des principaux motifs biologiques, ce sont les grammaires algébriques qui bénéficient d'une analyse syntaxique pouvant être menée en temps raisonnable. Le problème de la reconnaissance de l'appartenance d'un mot au langage est $NP - complet$

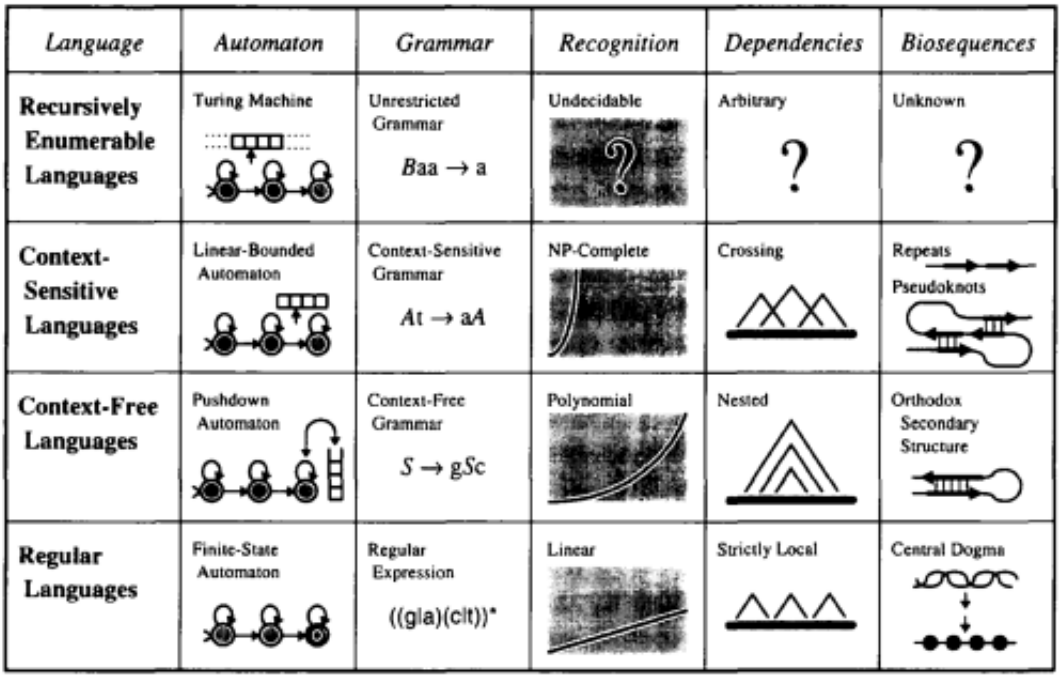


FIGURE 1.22 – Types de grammaires et propriétés associées [Sea97]

pour les grammaires *context-sensitive* (évolution exponentielle du temps de calcul dans le pire des cas) contre $P-complet$ pour les grammaires algébriques (évolution en temps polynomiale).

Les grammaires à variable de chaînes (SVG) : des grammaires faiblement sensibles au contexte (*mildly context-sensitive*)

Modéliser les concepts biologiques par des grammaires a fait l'objet d'un certain nombre de travaux [BB84, Hea87, EJM80, JM84], parmi lesquels ceux de David Searls qui a largement contribué à poser les fondations de ce domaine de recherche.

David Searls a énoncé le postulat que les signaux biologiques présents dans le génome obéissaient à des contraintes d'organisation et de structure comparables à ce qu'on peut observer dans les langages naturels : les signaux biologiques formeraient ainsi le "langage des gènes". Par extension, ce langage pouvait donc être décrit et modélisé par le biais de grammaires appropriées.

Il a été le premier à développer des méthodes permettant aux utilisateurs de concevoir des "grammaires biologiques" pour les employer lors d'analyses à grande échelle sur des séquences génomiques [SD93, Sea95, DS94].

Dans l'idée de permettre d'atteindre une partie de l'expressivité des grammaires contextuelles, David Searls a imaginé un nouveau formalisme appelé les grammaires à variables de chaînes (String Variable Grammar : SVG) [Sea88]. Les SVG permettent l'utilisation de variables logiques en tant que terminaux non instanciés dans les grammaires.

Cet ajout permet d'exprimer de façon simple différentes propriétés nécessaires à l'expression de motifs biologiques. Ainsi, il est possible d'exprimer le concept de copie directe $(X...X)$ qui recherche deux occurrences d'une même chaîne de caractères, sans qu'il soit nécessaire de définir ladite chaîne autrement que par sa taille. La copie reverse $(X... \sim X)$ ajoute la notion de reverse-complément à la copie et permet ainsi de modéliser des palindromes, notamment les

tiges-boucles (`STEM, LOOP, ~STEM`) ou les pseudo-nœuds (`STEM1, LOOP, STEM2, LOOP, ~STEM1, LOOP, ~STEM2`).

Bien que basées sur les grammaires algébriques, les SVG se situent à un niveau d'expressivité supérieur, entre les grammaires algébriques et les grammaires contextuelles. Cette position leur permet d'exprimer toute la richesse du langage biologique. Néanmoins, étant basées sur les grammaires algébriques, les SVG ne nécessitent pas autant de puissance de calcul que les grammaires contextuelles concernant l'analyse syntaxique de l'appartenance d'un mot à un langage SVG.

Les grammaires SVG semblent donc être une bonne réponse pour exprimer grammaticalement puis rechercher des motifs biologiques dans des séquences.

1.4.4 Recherche de motifs par modélisation grammaticale

Il existe un certain nombre d'outils permettant de faire de la reconnaissance de motifs en utilisant les modèles grammaticaux. Nous en présentons quelques uns dans cette section.

Analyse de séquence par les grammaires : GenLang

GenLang [DS94] est l'outil de reconnaissance de motifs mis au point par David Searls et Shan Dong pour exploiter les SVG (cf partie 1.4.3). Il permet d'analyser des séquences à la recherche de motifs structurés dans les séquences nucléotidiques (prédiction de gènes, épissage alternatif, inversion de génomes...). Les SVG sont utilisées pour définir des structures pouvant recevoir des contraintes basées sur la taille et le contenu [RBW06].

GenLang n'est plus maintenu à l'heure actuelle. En raison de la complexité de calcul, les analyses étaient limitées aux séquences de quelques megabases [NDR+05].

Analyse de séquence par les grammaires : PatScan/PatSearch

PatScan [DLO97] permet de modéliser des motifs sous forme de grammaire utilisant des variables de chaînes (cf partie 1.4.3).

PatScan permet de rechercher des modèles définis par une séquence précise ou par une fourchette de taille, pouvant fixer une quantité maximum d'évènements de mutations propre à chaque type (insertion, délétion et substitution). Les variables de chaîne lui permettent de rechercher des phénomènes de répétitions ou de palindromes. En outre, il autorise l'utilisation des matrices de comptage dans le modèle, en spécifiant le score minimum nécessaire pour valider un match de la matrice.

PatScan permet par ailleurs de définir les règles d'appariemments des nucléotides. Enfin, il permet de définir des ancres, notamment rechercher un motif spécifiquement au début ou à la fin d'une séquence.

```
p1 = 20...28, TTGACA[2,0,0], {(6,3,4,87)
   ,(8,4,5,83),(4,7,76,13),(64,12,6,18),
   (24,51,8,17),(53,9,11,27)} > 350 , ~p1
```

FIGURE 1.23 – Grammaire PatScan recherchant une tige de taille 20-28, le mot "ttgaca" avec jusqu'à 2 substitutions, 0 insertion, 0 délétion, une séquence définie par une matrice devant totaliser un score d'au moins 350 et enfin la séquence complémentaire de la tige. Les poids des positions de la matrice sont respectivement donnés pour A, C, G et T

Un inconvénient de PatScan réside dans la non-exhaustivité de ces résultats. Notamment, dans le cadre de motifs complexes, lorsque PatScan identifie une solution à une position d'une séquence, il ne vérifie pas s'il existe d'autres alternatives possibles à cette même position [NDR+05].

PatSearch [PLD00, GLL+03] est un outil du même style que PatScan et fonctionnant sur le même principe.

Analyse de séquences par les grammaires : RNAMotif

RNAMotif [MEG+01] est un outil de reconnaissance de motifs spécialisé dans la recherche de motifs intégrant des structures secondaires dans les ARN.

RNAMotif permet de rechercher des modèles basés sur les tiges dans les ARNm. Seule la boucle d'une tige peut être définie de façon spécifique : les palindromes ne sont définis que par contraintes de tailles et de mésappariemment.

RNAMotif ne permet pas la recherche de motifs n'incluant pas de palindrome.

1.5 Logol : modèles grammaticaux au-delà des SVG

La communauté biologiste ne disposait pas d'un outil capable d'exprimer une architecture globale de motifs ou de vérifier différentes hypothèses sur la façon dont leurs motifs sont structurés. C'est pour cette raison que l'équipe DYLISS a mis au point Logol[BSN14], un langage générique permettant de construire des modèles biologiques expressifs, pouvant intégrer à la fois des informations sur le contenu et sur la structure des séquences.

Le nom Logol désigne à la fois un langage et l'analyseur dédié permettant de lire les grammaires écrites dans ce langage et d'analyser des séquences avec ces grammaires. Cette partie présente successivement ces deux formes, avant de donner les clefs permettant de lire une grammaire Logol. Elle se conclura par une présentation de quelques exemples permettant de mieux saisir l'expressivité que permet ce langage.

1.5.1 Logol : un langage et un analyseur

Un langage

Logol est un langage conçu pour l'analyse des séquences biologiques, qui s'appuie sur les grammaires SVG (cf 1.4.3) tout en améliorant leur expressivité. En effet, le langage Logol

s'appuie sur les variables de chaînes mais va au-delà, en y autorisant des erreurs, des chevauchements en ajoutant des opérations de contraintes et des opérateurs dédiés. Ceci étend les possibilités des SVG tout en se rapprochant de la réalité des structures biologiques.

Le langage Logol a pour vocation d'exprimer tant des motifs basés sur une séquence (ex : mots) que des motifs basés sur une structure (ex : palindromes, répétitions). Ces motifs peuvent être stricts ou approchés (indels, substitutions) et éventuellement n'être définis que par une longueur.

La modélisation avec Logol est basée sur une approche par contraintes. Les nombreuses contraintes qui peuvent s'appliquer à un modèle se répartissent en deux catégories : les contraintes de séquence et les contraintes de structure.

Les contraintes de séquence définissent des éléments tels que la position de début, de fin, le contenu ou la taille des matchs.

Les contraintes de structure définissent les contraintes de coûts (avec un compteur spécifique aux substitutions ('$')et un compteur spécifique aux indels ('$$')) et les contraintes de composition (tel le pourcentage minimum de nucléotides "G" et "C", par exemple).

Au-delà de ces contraintes figurent en plus les opérateurs, tels que les négations (pour interdire une séquence en particulier), les morphismes (permettant notamment la recherche automatique du reverse-complément d'une séquence) ou les répétitions.

Enfin, Logol donne la possibilité de sauver des instances de séquences pour les rechercher en d'autres points du modèle. Typiquement, dans le cas d'une recherche de tige-boucle, cette propriété permet d'identifier une tige particulière en autorisant des erreurs tout en s'assurant que la partie complémentaire tient compte de ces erreurs.

Toutes ces propriétés font que Logol est un langage dont l'expressivité va au-delà des grammaires SVG strictes. Par ailleurs, toutes ces contraintes sont directement définies dans le modèle, permettant des modèles explicites, sans aucun paramètre caché.

Un analyseur

L'analyseur prend en charge une grammaire Logol et effectue la reconnaissance de motifs sur les séquences.

L'analyseur renvoie une liste exhaustive des hits appartenant au langage engendré par la grammaire Logol et présents dans une séquence. Par défaut, l'ensemble de ces solutions est encapsulé dans un fichier XML qui contient l'arbre de dérivation de chacune des solutions. À chaque étape d'une dérivation, la solution indique quelle partie du hit a vérifié quelle partie de la grammaire et, le cas échéant, quelles sont les contraintes de coût qui sont intervenues (substitutions, indels).

De la même façon que le langage Logol vise à être un langage explicite sans paramètres cachés, l'analyseur Logol fournit une solution dans un format qui explicite clairement la façon dont elle a été trouvée.

L'analyseur est écrit en Java et Prolog. Il analyse les grammaires Logol et les traduit en Prolog pour effectuer la recherche des éléments de la grammaire. Au chargement de la séquence, celle-ci est indéxée par tableau de suffixe avec Vmatch [Kur03] (ou avec l'alternative locale Cassiopée, en cas d'utilisation non-académique). Dans le cas où la grammaire contient une recherche de mot après un gap (section de séquence non-définie entre différents éléments de la grammaire, par l'opérateur '.*'), l'analyseur fait appel à l'index.

Logol est disponible sur la plate-forme Genouest :
(http ://tools.genouest.org/tools/logolanalyser/submit.php).

1.5.2 Lire et comprendre une grammaire Logol

Point de départ d'une grammaire Logol

Le formalisme des grammaires Logol a été conçu pour pouvoir être facilement compris, quel que soit le degré de complexité. Les grammaires se présentent basiquement de la façon suivante :

```
def{
}
mod1()==>"aaa"
mod1()==*>SEQ1
```

FIGURE 1.24 – Grammaire Logol basique reconnaissant le motif "AAA"

L'entête "def" est la zone de la grammaire où peuvent être définies des contraintes générales aux modèles, telles que les morphismes ou les contrôles. Elle peut être vide, car elle n'est utilisée que lorsqu'il est nécessaire de vérifier des contraintes concernant plusieurs zones non contigües de la grammaire ou lorsqu'on veut définir un nouveau morphisme, par exemple les liaisons wobbles("A"→"T", "C"→"G", "G"→"C", "G"→"T", "T"→"A", "T"→"G"). La complémentarité Watson-Crick étant un morphisme de base en génomique, il est implémenté par défaut et ne nécessite par d'être réécrit par l'utilisateur.

De la même façon qu'une grammaire formelle définit des dérivations d'un non-terminal vers une suite d'éléments terminaux/non-terminaux, la grammaire Logol dérive des modèles vers des contenus. Ces contenus peuvent être explicites (i.e. des contenus terminaux), comme ici avec "aaa", ou bien être des modèles définis ailleurs dans la grammaire (i.e des non-terminaux), par exemple mod1()==>mod2(). On parle de "modèle hiérarchique". Un non-terminal peut pointer vers une suite de contenus, qui sont alors séparés par des virgules. Par exemple mod1()==>"aaa",mod2(),"actg" signifie que le mod1() est constitué de "aaa" immédiatement suivi d'un contenu décrit par mod2(), lui-même immédiatement suivi de "actg".

Enfin, la dernière ligne de la grammaire est celle qui indique à l'analyseur quelle ligne du modèle va être le point de départ de la recherche dans la séquence (représentée par ==*>SEQ1), c'est-à-dire l'axiome.

Ajouter des contraintes au modèle

Les contraintes afférentes à un élément de la grammaire sont définies dans un cartouche formé par deux accolades tel que "`contenu:{contrainte}`", comme dans la grammaire ci-dessous.

```
def{
}
mod1()==>"aaa",mod2(),"actg":{$[0,1]}
mod2()==>X1:{#[5,5]}
mod1()==*>SEQ1
```

FIGURE 1.25 – Grammaire Logol reconnaissant le motif "aaa", suivi d'une chaîne de nucléotides quelconques de taille 5, suivi du motif "actg" à 0 ou 1 substitution près. Elle accepte par exemple le mot "AAATTCATACGG"

Comme déjà indiqué, les contraintes se divisent en deux types : les contraintes de séquence et les contraintes de structure. Lorsqu'il est nécessaire de définir ces deux types de contraintes pour un même motif, il faut un double cartouche `contenu:{contraintes de séquence}:{contraintes de structure}`, comme dans l'exemple de la figure 1.27.

Au sein du cartouche formé par les accolades, une contrainte est classiquement définie par un symbole suivi entre crochets de l'effet minimum et l'effet maximum. Chaque symbole est défini et correspond à une contrainte particulière, ainsi $ désigne les évènements de substitutions, $$ désigne les évènements d'indels, @ désigne les contraintes de début, @@ désigne les contraintes de fin et # désigne la taille. Dans la figure 1.25, le motif "acgt" peut contenir entre 0 et 1 substitution, comme précisé entre crochet après le symbole $, tandis que la variable X1 peut avoir une taille comprise entre 5 et 5, comme précisé entre crochets après le symbole #. La liste des différents symboles et de leur signification est donnée en annexe A.1.

Recherche exhaustive des solutions possibles pour les grammaires ambigües

Une grammaire ambigüe est une grammaire qui admet plusieurs façons d'analyser un même mot. Les grammaires Logol peuvent être des grammaires ambigües. Par exemple :

— Soit la séquence : `atgaca`

— Soit la grammaire reconnaissant le mot "atg" avec au plus un indel suivit du mot "aca" avec au plus un indel :

```
def:{
}
mod1()==>"atg":{$$[0,1]},"aca":{$$[0,1]}
mod1()==*>SEQ1
```

L'analyse de la séquence par cette grammaire aboutit à trois solutions différentes reconnaissant l'ensemble de la séquence :

— Cas 1 : les deux éléments de la grammaire sont complets : "ATG" / "ACA"

— Cas 2 : Le premier élément est tronqué et le second est rallongé : "AT_" / "TACA"

— Cas 3 : Le premier élément est rallongé et le second est tronqué : "ATGT" / "_ACA"

Toutes ces solutions sont reconnues par la grammaire et sont donc détaillées dans le fichier de résultats fourni par l'analyseur.

1.5.3 Exemples de grammaires Logol

Afin de donner une idée plus précise de l'expressivité permise par Logol, voici quelques exemples de motifs à exprimer.

Motif imparfaitement répété

```
def:{
}
mod1()==>"tgaccg":{$[0,1],_PART1},GAP1:{#[4,4]},
   ?PART1:{$[0,1]}
mod1()==*>SEQ1
```

FIGURE 1.26 – Grammaire Logol reconnaissant le motif "TGACCG" avec jusqu'à 1 substitution, répétée, avec jusqu'à 1 substitution, 4 nucléotides plus loin. Utilise la "sauvegarde d'instance"

Cette grammaire présente la recherche du motif "TGACCG" pour lequel on autorise jusqu'à 1 substitution, suivie quatre nucléotides plus loin du motif tel qu'il a été trouvé (c'est-à-dire potentiellement muté) avec jusqu'à 1 substitution. Cela signifie que si "TGACCG" est trouvé de façon exacte, sa répétition aura jusqu'à 1 substitution par rapport à la référence (on identifiera par exemple TGACCGnnnnTGACCG si la répétition est parfaite ou bien TGACCGnnnnTG**T**CCG si la répétition est imparfaite). Si TGACCG est identifié de façon imparfaite, alors la répétition aura entre 1 à 2 substitutions par rapport à la référence (on identifiera par exemple TGACC**T**nnnnTGACC**T** si la répétition est parfaite avec la première partie du motif muté ou bien TGACC**T**nnnn**A**GACC**T** si la répétition est imparfaite en plus d'une première partie du motif muté). Concrètement, cette grammaire définit un premier motif "TGACCG" qui autorise jusqu'à 1 substitution et stocke l'instance telle qu'elle sera trouvée (donc potentiellement avec la substitution) dans une variable appelé PART1. Après une variable GAP1 dont le contenu n'est pas défini mais dont la taille est fixée à 4, la grammaire réinjecte la variable PART1 (grâce à "?PART1") et lui applique de nouvelles contraintes. On notera au passage que le symbole en préfixe de PART1 désigne quel élément de l'instance est utilisé : ainsi, '?' renvoie à la séquence, tandis que '@' renverrait à sa position de début, etc.

Principe de sauvegarde d'instance : on a ici d'une part le fait de rechercher une répétition approximative (l'élément répété peut avoir des contraintes supplémentaires), mais d'autre part une répétition qui prend en compte ce qui a été préalablement trouvé et non la référence parfaite.

Motif structuré

```
def:{
}
mod1(PE1)==>"aataaa":{@[120,135],_PE1}:{$[0,2]}
mod2(PE1)==>!"gataaa":{@[@PE1,@PE1]}
mod3(PE1)==>"ca":{@[16 + @@PE1,27 + @@PE1]}
mod4()==>repeat("atatata":{@[90,120],#[6,6]}:                    {
   $$[1,1],$[0,1]};[0,10])+ [5,5]
mod4(). mod1(PE1). mod2(PE1). mod3(PE1)==*>SEQ1
```

FIGURE 1.27 – Grammaire Logol reconnaissant une répétition de 5 motifs "ATATATA" contenant forcément une délétion et avec jusqu'à 1 substitution, positionnée en 90-120 de la séquence, suivi d'un motif "AATAAA" positionné en 120-135 de la séquence, avec jusqu'à 2 substitutions mais qui ne forme pas le motif "GATAAA", et un dernier motif "CA" positionné entre 16 et 27 nucléotides en aval du motif "AATAAA"

.

La grammaire de la figure 1.27 présente la recherche d'un motif composite, dont les différents éléments sont positionnés de manière précise entre eux.

Opérateur de répétition : le premier élément est défini dans mod4(). C'est une répétition d'un motif "ATATATA", avec nécessairement une délétion pour correspondre à une taille de 6, jusqu'à 1 substitution, et positionné en 90-120 de la séquence. Le repeat définit l'intervalle entre les différents éléments répété ([0,10]) ainsi que le nombre de répétitions autorisées ([5,5]). Les 5 motifs "ATATATA" peuvent être différents entre eux du moment qu'ils respectent les contraintes de mutations définies.

Passage de paramètres, négation : le second élément est défini dans mod1() et mod2(). mod1() recherche le motif "AATAAA", avec jusqu'à 2 substitutions, et positionné en 120-135 de la séquence. L'instance qui sera exactement trouvée sera stockée dans la variable PE1. Cette variable PE1 (et toutes ces caractéristiques) va naviguer entre les différentes règles de grammaires pour que les autres puissent s'y référer. mod2() analyse justement la portion de séquence qui commence là où commence PE1 (donc le motif "AATAAA" tel que trouvé) et vérifie qu'il ne s'y trouve pas le motif "GATAAA". La combinaison de mod1() et mod2() recherche donc toutes les variantes du motif "AATAAA" avec jusqu'à deux substitutions, exception faite du motif "GATAAA".

Contrainte de position : le dernier élément est défini dans le mod3(). Il recherche un motif "ca", dont le début est positionné entre 16 et 27 nucléotides plus loin que le début du motif exactement obtenu dans mod1(). On recherche donc 5 motifs ressemblant à des "ATATATA" sur la portion 90-120 de la séquence, suivi d'un motif approché de "AATAAA" qui ne peut pas être "GATAAA" sur la portion 120-135, suivi d'un motif "CA" positionné entre 16 et 27 nucléotides plus loin que le motif approché de "AATAAA".

Analyses multiple d'une même séquence avec passage de paramètre : Ici, l'ensemble de la recherche effectuée par Logol se fait par des analyses successives de la séquence en fonction des mod(), dont l'ordre est donné dans la dernière ligne de la grammaire. Ce procédé permet notamment de revenir analyser des portions ciblées de la séquence éventuellement chevauchantes, notamment grâce à la transmission de la variable PE1 entre les différents éléments et le positionnement des motifs suivants par rapport à la position de PE1.

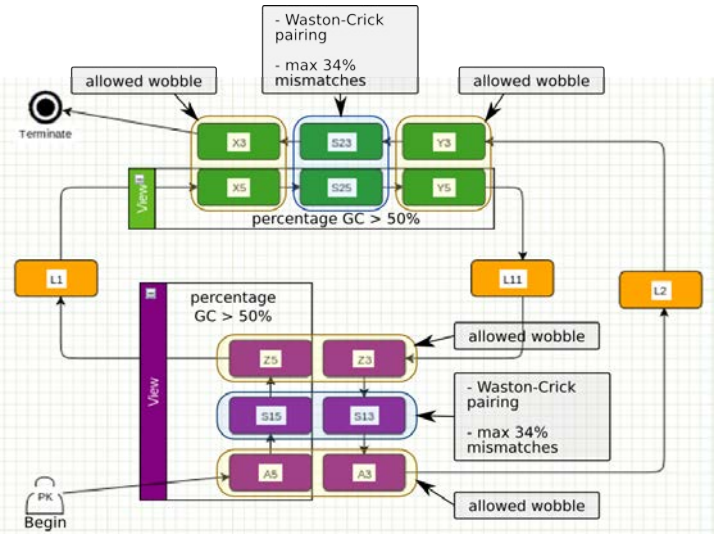


FIGURE 1.28 – Schéma des contraintes exprimées en Logol pour modéliser le pseudo-nœud

Motif complexe : le pseudo-nœud d'une structure Frameshift-1

Le motif complexe de Frameshift-1 a été traduit en Logol [BSN14] car il sollicite un grand éventail de son expressivité (cf figure 1.9 en partie 1.2). Ces différentes particularités nécessitent souvent la mise en œuvre d'outils de détection dédiés procédant habituellement par calculs thermodynamiques (cf section 1.3.5). Il s'agit donc d'un modèle emblématique de la puissance de l'expressivité de Logol puisque Logol est capable d'exprimer ce motif sans avoir été spécifiquement conçu pour le faire. En particulier, la traduction de la structure de pseudo-nœud présente dans le motif Frameshift-1, fait appel à de nombreuses contraintes.

La particularité du pseudo-nœud utilisé dans le motif Frameshift-1 réside dans les contraintes de type thermodynamiques qu'on y a inclu et qui sont :

— L'ensemble d'une tige doit contenir un minimum de 50% de liaison "GC", qui sont les liaisons les plus stables.
— Les parties extérieures de la tige (amont et aval) peuvent accepter des liaisons wobbles, c'est-à-dire des liaisons "GT".
— La partie centrale de la tige n'accepte que des liaisons Watson-Crick, mais admet jusqu'à 34% de mésappariemment entre les deux brins (i.e. cela inclut potentiellement des liaisons wobbles).

Ces différentes contraintes sont modélisées en grammaire Logol dans la figure 1.29.

```
def:{
morphism(wcw,a,t)
morphism(wcw,t,a)
morphism(wcw,c,g)
morphism(wcw,g,c)
morphism(wcw,g,t)
morphism(wcw,t,g)
}
controls:{
% "c"[mod3.SA5,mod3.TIGE15,mod3.SZ5,mod3.SZ3,mod3.TIGE13,
   mod3.SA3]>=25
% "c"[mod3.SX5,mod3.TIGE25,mod3.SY5,mod3.SY3,mod3.TIGE23,
   mod3.SX3]>=25
}
mod1()==>(A5:{#[1,1],_SA5},S15:{#[2,14],_TIGE15},Z5
   :{#[1,1],_SZ5}):{%"gc":50},
          L1:{#[1,5]},
          (X5:{#[1,1],_SX5},S25:{#[1,6],_TIGE25},Y5
             :{#[1,1],_SY5}):{%"gc":50},
          L11:{#[0,4]},
          -"wcw" Z3:{?SZ5,_SZ3},-"wc" ?TIGE15:{_TIGE13}:{p$
             [0,34]},-"wcw" A3:{?SA5,_SA3},
          L2:{#[4,40]},
          -"wcw" Y3:{?SY5,_SY3},-"wc" ?TIGE25:{_TIGE23}:{p$
             [0,34]},-"wcw" X3:{?SX5,_SX3}
mod2()==*>SEQ1
```

FIGURE 1.29 – Grammaire Logol modélisant le pseudo-nœud des structures Frameshift-1

Où :

- Le morphisme wcw définit les liaisons Watson-Crick et wobbles (on y retrouve les 4 liaisons standards Watson-Crick, plus la relation réciproque entre "G" et "T").
- mod3() détaille le pseudo-noeud, selon le schéma présenté en figure A.19. Chaque tige-boucle a été séparée en trois sections. Les sections aux extrémités autorisent des liaisons wobbles, sans limite, tandis que la section centrale n'autorise que des appariements Watson-Crick et 34% de mésappariemment. Enfin, l'ensemble des 3 sections de la tige-boucle doivent compter au moins 50% de liaisons "GC". Ce compte est vérifié sur un seul brin de la tige en se basant sur les "C" (les "G" ne sont pas représentatifs à cause des liaisons wobbles).
- Le contrôle défini un pourcentage de "C" minimum pour l'ensemble des éléments constituant la tige 1 ainsi que pour l'ensemble des éléments définissant la tige 2 (au moins 25% de C).

Une spécificité intéressante de ce modèle réside dans la vérification des proportions des liaisons "GC" dans les tige-boucles du modèle. En effet, la contrainte de 50% de "C" dans les

tiges n'est pas suffisante car la partie centrale de la tige-boucle autorise des mésappariements, donc des "C" sans nucléotides complémentaires en vis-à-vis. C'est pour cela que la grammaire définit un contrôle qui vérifie au moins 25% de "C" dans les deux des brins d'une tige. Bien que cette façon de compter considère un "G" mésapparié et un "C" mésapparié comme une même liaison "GC", cela constitue déjà une première approximation de la notion de stabilité d'une tige-boucle. Cette méthode permet donc de "mimer" dans une certaine mesure les calculs d'énergie libre des structures secondaires pour les intégrer dans des grammaires.

Ce modèle utilise donc les mécanismes Logol suivants :
— Composition d'un segment (pourcentage de "GC")
— Définition d'un morphisme personnalisé (la liaison wobble)
— Contraintes globales sur un ensemble de segments non-contigus (pourcentage de "C" dans l'ensemble de la tige)

1.6 Bilan à propos de la reconnaissance de motifs dans les séquences biologiques

On a perçu au travers de ce chapitre que de nombreux motifs biologiques existent, plus ou moins complexes. On a aussi vu qu'il existe de nombreux outils de reconnaissance de motifs dans la communauté des biologistes, dont très peu sont basés sur les grammaires. Par ailleurs, on a vu que les grammaires SVG, de par leur expressivité (au-delà des grammaires algébriques) proposent un formalisme *a priori* adapté aux motifs biologiques. Et, on a présenté Logol, un langage et un analyseur basé sur les SVG, mais qui étend l'expressivité des SVG au-delà des précédents outils grammaticaux.

Les chapitres suivants étudieront différentes applications biologiques pour essayer de comprendre quel types de modèles, grammaticaux ou non, sont les plus adaptés pour rechercher leurs motifs, pour chercher à comprendre quelle est la place des modèles grammaticaux et en particulier de Logol dans le panorama des outils de recherche de motifs.

CHAPITRE 2

Différence de cibles entre les matrices de score et les expressions régulières approchées

La question posée dans ce chapitre est de comparer la reconnaissance de motifs par expression régulière approchée avec les matrices de score, afin de déterminer quelle méthode est la plus adaptée pour la recherche de sites de fixation de facteurs de transcription, au travers de l'exemple de LXRα

Nous utilisons donc ici l'application LXRα (explorée au chapitre 3 et en section 5.1) pour faire une étude méthodologique.

Nous montrons tout d'abord que les matrices de comptage sont capables de cibler plus spécifiquement les séquences qui nous intéressent en limitant plus efficacement le nombre de séquences non-pertinentes par rapport aux expressions régulières approchées. Nous montrons ensuite une méthode pour fixer un seuil de validation pertinent pour notre recherche de séquences voisines des séquences de référence.

Les travaux présentés dans ce chapitre ont fait l'objet d'une présentation orale "Comparison of the targets obtained by a scoring matrix and by a regular expression" *et ont été publiés dans les actes de JOBIM 2015 [ALLL$^+$15] (Journées Ouvertes en Biologie, Informatique et Mathématiques).*

Dans le chapitre précédent, nous avons posé des bases sur la reconnaissance de motifs dans les séquences biologiques avec un focus sur la reconnaissance grammaticale.

Dans ce chapitre, nous cherchons à cerner s'il existe ou non une différence importante entre un modèle par matrice et une modélisation de type grammatical qu'est l'expression régulière approchée. Pour cela, nous nous intéressons à une problématique récurrente en bio-informatique qui consiste en la recherche d'éléments similaires ou voisins d'une série d'occurrences connues.

Une des deux méthodes de modélisation étudiée ici est celle des expressions régulières, qui consiste à indiquer quelles sont les bases permises à chaque position de la séquence, par le biais de formats plus ou moins élaborés (ex : dreg de EMBOSS [RLB00], grappe [KR95], scripts NRgrep [Nav01], Sequence Searcher [BSTU06]).

La seconde méthode étudiée est celle de type matrice de comptage (ex : TESS [SO97], matrix-scan de RSAT [TTCDvH08], oPOSSUM [KAWHW12]), qui consiste globalement à affecter un poids à chaque lettre pour chaque position en fonction de sa représentativité dans les séquences de référence.

Ces deux types de méthodes semblant utilisés indifféremment dans la pratique, nous nous sommes demandé si choisir l'une ou l'autre avait ou non un impact sur les résultats : les scores permis par les matrices donnent-ils des résultats plus fins que les expressions régulières approchées ? En pratique, quelles sont les occurrences retenues par la matrice ? Comment fixer un score utilisateur pertinent pour valider ou rejeter une occurrence ? De bons candidats (i.e. proches des références) peuvent-ils passer à travers les mailles de l'une ou l'autre méthode ? Le fait d'autoriser des substitutions sur les expressions régulières permet-il le même genre de graduation dans les résultats trouvés que le fait de faire varier un score seuil sur une matrice ?

Nous avons cherché à répondre à ces questions au travers d'une approche pratique pour laquelle nous disposons des conseils d'experts biologistes : la recherche des sites de fixation du facteur de transcription LXRα (cf Chapitre 3).

Afin d'analyser quantitativement et qualitativement les occurrences acceptées par chacune de ces deux méthodes, nous avons créé deux jeux de données. La première série, ERx, correspond à l'ensemble des séquences reconnues par l'expression régulière à deux substitutions près. Nous cherchons à constater dans quelle mesure ces séquences sont considérées par la matrice. La deuxième série, Refx, correspond aux séquences proches des séquences de référence (les sites avérés), c'est-à-dire l'ensemble des possibilités qui découlent des références jusqu'à 2 substitutions près. Elle représente les séquences qu'on cherche à cibler et nous chercherons, là encore, à constater dans quelle mesure ces séquences sont considérées par la matrice.

Bien que construites à partir des mêmes références, les spécificités propres aux modèles grammaticaux et aux modèles matriciels impliquent que ceux-ci ne ciblent pas exactement les mêmes séquences. Nous montrons ici que les matrices s'avèrent plus efficaces que les expressions régulières approchées pour détecter un motif de fixation de type LXRα et nous proposons une approche pour choisir un seuil de validation pour les hits.

2.1 Modélisation du TFBS LXRα

2.1.1 Propriété du modèle

Nous disposons de 13 séquences de référence validées expérimentalement (cf figure 2.1), dont le consensus est : $TGACCGnnnnTAACCC$. D'après l'expertise biologique, le facteur de transcription est de type DR4 [MMYG06], c'est-à-dire que les 4 nucléotides centraux n'influenceraient pas la fixation du facteur de transcription. Pour cette raison, ces 4 positions centrales sont considérées comme étant non-définies dans le consensus des 13 séquences de référence.

L'objectif de cette recherche est d'identifier les séquences voisines du site de fixation du facteur de transcription (Transcription Factor Binding Site, TFBS) LXRα. Par séquence voisine, nous entendons arbitrairement toutes les séquences à 2 substitutions près d'une référence. La taille de la séquence reconnue par le facteur de transcription LXRα étant de taille fixe, les insertions et délétions ne sont pas autorisées dans le modèle de mutation.

2.1.2 Modèle matriciel

La matrice de comptage LXRα (cf figure 2.2) a été construite en prenant en compte les spécificités du modèle. Dans notre cas, cela signifie que les quatre positions centrales (7 à 10) ont reçu un poids identique : la nature des nucléotides à ces positions n'aura donc aucune influence sur le score final d'un hit.

Id	Ensembl Id	Species	Sequence	Origine	valeur-p
01	Cyp7Alpha1	Souris	TGAACTtgggTGACCA	[CKS01]	$2e^{-5}$
02	Cyp7Alpha1	Rat	TGAACTtgagTGACCA	[CKS01]	$2,1e^{-5}$
03	FASN	Souris	TGACCGgtagTAACCC	[DDD^{+}09]	$8,4e^{-9}$
04	FASN	Rat	TGACCGgtagTAACCC	[JLP^{+}02]	$1,1e^{-8}$
05	FASN	Poule	TGACCTgtggTAACCT	[JLP^{+}02]	$4,8e^{-7}$
06	FASN	Humain	TGACCGgcagTAACCC	[JLP^{+}02]	$2,4e^{-8}$
07	LPCAT3	Humain	CGACCGggagTAACCT	[DLD^{+}11]	$2,4e^{-8}$
08	LPCAT3	Souris	CGACCGagagTAACCT	[DLD^{+}11]	$1,8e^{-8}$
09	LPCAT3	Rat	CGACCGagagTAACCT	[DLD^{+}11]	$2,2e^{-8}$
10	LPCAT3	Poule	TGCCCGcgagTAACCC	[DLD^{+}11]	$7,3e^{-8}$
11	CETP	Humain	TGCCCGacaaTGACCC	[LT00]	$7,8e^{-7}$
12	CYP51a	Humain	TGACCTcaggTGATCC	[WRS^{+}08]	$8,3e^{-6}$
13	SCD1	Souris	TGACCAcaggTAACCT	[CMMN06]	$2,4e^{-6}$
	Consensus		TGACCGnnnnTAACCC	[CMMN06]	

TABLE 2.1 – Gène, espèce et hit des 13 séquences de référence LXRα. Les nucléotides en minuscule représentent les 4 nucléotides centraux qui, en principe, n'interviennent pas dans la fixation du TF LXRα et dont la valeur n'est donc pas sensée avoir d'impact sur la reconnaissance du site

Position	**01**	**02**	**03**	**04**	**05**	**06**	**07**	**08**
A	0	0	11	2	0	1	3,25	3,25
C	3	0	2	11	13	0	3,25	3,25
G	0	13	0	0	0	8	3,25	3,25
T	10	0	0	0	0	4	3,25	3,25
Position	**09**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
A	3,25	3,25	0	9	13	0	0	2
C	3,25	3,25	0	2	0	12	13	6
G	3,25	3,25	0	0	4	0	0	0
T	3,25	3,25	13	0	0	1	0	5

TABLE 2.2 – Matrice LXRE13, les poids majoritaires sont soulignés

```
[CT]G[AC][AC]C[AGT]NNNNT[AC][AG][CT]C[ACT]
"Substitutions : [0,2]"
```

FIGURE 2.1 – Grammaire LXRE13, autorisant jusqu'à 2 substitutions au total

2.1.3 Modèle grammatical

La grammaire de LXRα (cf figure 2.1) a été construite en prenant en compte les spécificités du modèle. Dans son cas, cela signifie que les quatre positions centrales (7 à 10) sont indéfinies, afin de ne pas limiter les hits reconnus par la grammaire. Par ailleurs, la grammaire autorise jusqu'à 2 substitutions, conformément au périmètre que nous avons souhaité pour la détection des voisins.

À l'instar du modèle matriciel, les quatre positions centrales sont donc neutralisées. Elles sont modélisées pas des N, indiquant que toutes les possibilités sont acceptées. Ces positions n'entrent donc jamais dans le décompte des substitutions d'une séquence.

2.2 Jeux de séquences analysées

2.2.1 Les séquences ciblées par la grammaire LXRE13 : les séries ER

Les séries ER constituent l'ensemble des séquences qui peuvent être reconnues par la grammaire LXRE13.

Le jeu de données ER0 regroupe toutes les séquences reconnue par la grammaire sans utiliser de substitutions (par exemple : *CGCACAacgtTCATCA*). Elle rassemble 73 728 séquences.

Le jeu de données ER1 regroupe toutes les séquences reconnues par la grammaire en utilisant exactement 1 substitution (par exemple : ***A****GCACAacgtTCATCA*). Le jeu ER1 regroupe un total de 1 523 712 séquences différentes, toutes différentes du jeu ER0.

Le jeu de données ER2 regroupe toutes les séquences reconnues par la grammaire en utilisant exactement 2 substitutions (par exemple : ***A****GCAC****C****acgtTCATCA*). Le jeu ER2 regroupe un total de 15 893 632 séquences, toutes différentes des jeux ER0 et ER1.

La série ER0 correspond au produit cartésien de l'ensemble des alternatives possibles pour chaque position observées dans les séquences de référence. Cela implique que des séquences n'étant pas des séquences de référence figurent au sein de cette série. Par extension, cela signifie que les variantes de ces séquences à 2 substitutions près (et donc présentent dans ER2) se situent au-delà de 2 substitutions des séquences de référence elles-mêmes (cf figure 2.2).

2.2.2 Les séquences cibles : les séries Ref

En parallèle du jeu de séquences ciblées par la grammaire LXRE13, nous avons construit un jeu de séquences rassemblant spécifiquement les séquences voisines des séquences de référence. Nous avons ainsi créé une autre série de 3 autres jeux de données : Ref0, Ref1 et Ref2.

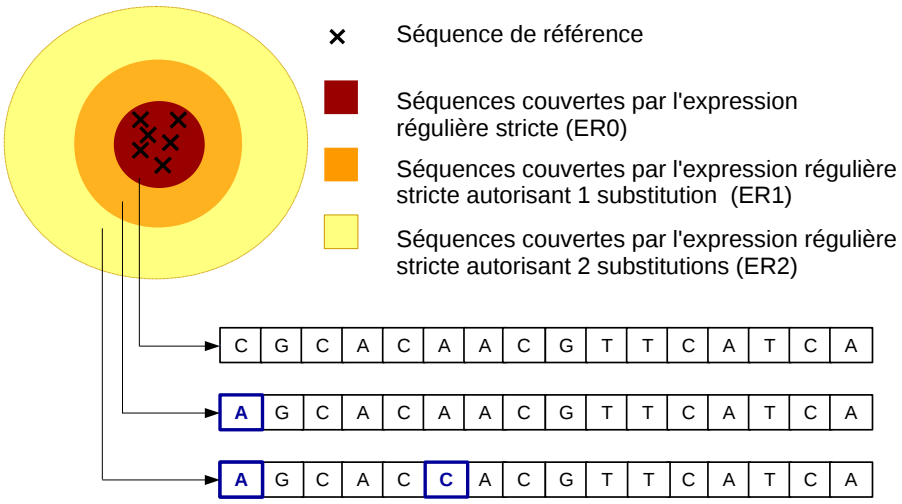


FIGURE 2.2 – Schéma des Séries ERX (ER0, ER1 et ER2). La fusion des références aboutit à une expression régulière autour de laquelle les séries rayonnent en fonction du nombre croissant de substitutions autorisées

Le jeu de données Ref0 rassemble simplement les 13 séquences de référence (exemple : *TGAACTtgggTGACCA*).

Le jeu de données Ref1 rassemble toutes les possibilités de variants d'une séquence Ref0 avec une substitution, si ce variant n'appartient pas déjà au jeu Ref0 (exemple : *TGA**G**CTtgggTGACCA*).

De la même façon, le jeu Ref2 rassemble toutes les possibilités de variants d'une séquence Ref1 avec une substitution, si ce variant n'appartient pas déjà au jeu Ref0 ou Ref1, ce qui aboutit à toutes les variants de Ref0 avec 2 substitions (exemple : *TGA**G**CTtgg**a**TGACCA*).

Le jeu de données Ref0 contient 13 séquences, le jeu Ref1 en contient 648 et le jeu Ref2 en contient 17 842 (cf figure 2.3).

Modalité de test d'un jeu de données

Les jeux de données des différentes séries ont été testés en utilisant le logiciel RSAT (cf p 27) pour les analyser avec la matrice LXRE13, avec un background nucléotide neutre (équiprobabilité de présence d'un nucléotide). Par défaut, le filtre du logiciel ne conserve que les hits dont la valeur p est inférieure ou égale à 10^{-4}.

2.3 Répartition des séquences de la série RefX dans la série ERX ?

Par construction, toutes les séquences des jeux Ref1 et Ref2 sont forcément incluses dans la série ERX. Il est néanmoins intéressant d'observer de quelle façon les RefX se répartissent au sein des jeux ERX (cf table 2.3).

L'analyse de la répartition des séquences de la série RefX permet de remarquer que seule

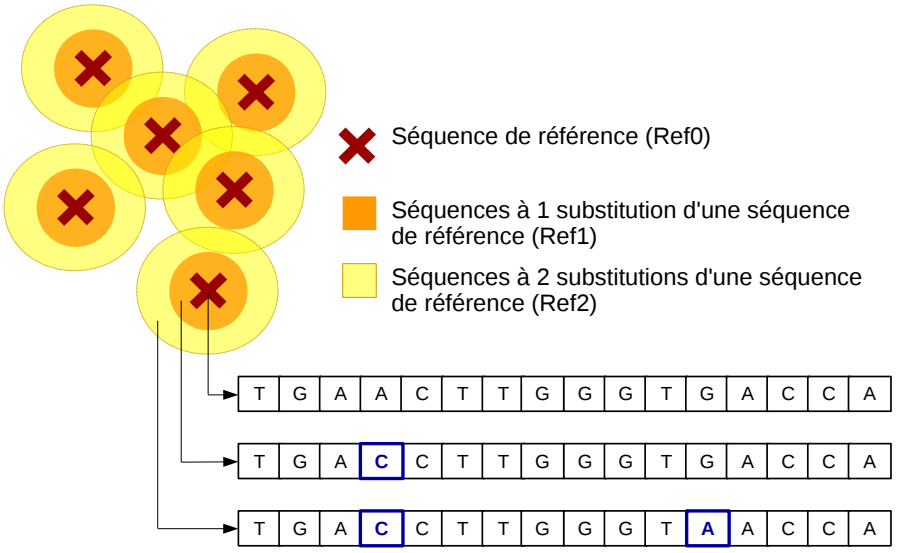
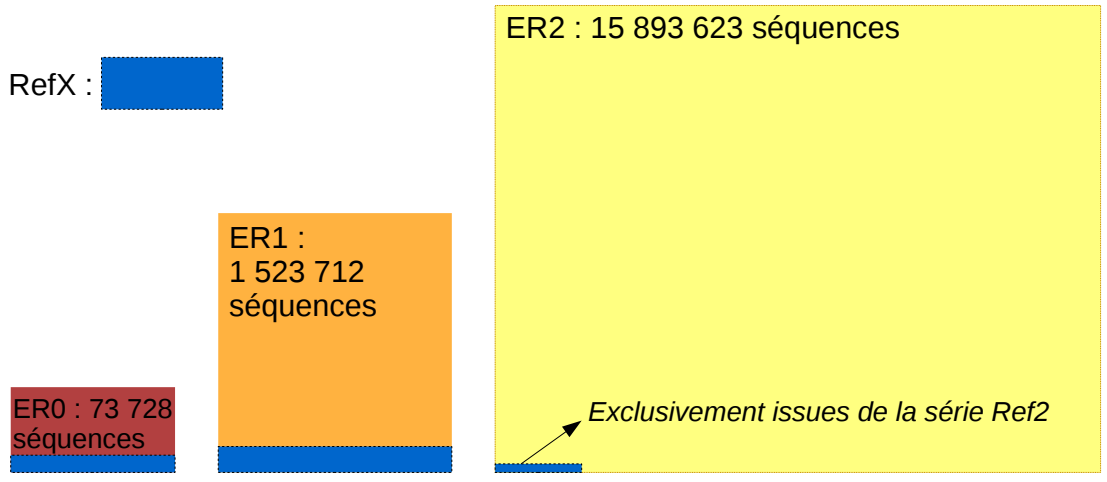


FIGURE 2.3 – Schéma des Séries RefX (Ref0, Ref1 et Ref2). Les variants rayonnent depuis les références par nombre croissant de mutations



	ER0	ER1	ER2
Ref0 (13 sequences)	100% (13 sequences)	-	-
Ref1 (648 sequences)	54,2% (351 séquences)	45,8% (297 séquences)	-
Ref2 (17 842 sequences)	29,05% (5 183 séquences)	50,55% (9 018 séquences)	20,4% (3 641 séquences)

TABLE 2.3 – Répartition des séquences de la série RefX au sein de la série ERX

	Jeu ER0	Jeu ER1	Jeu ER2
Séquences en entrée	73 728	1 523 712	15 893 632
Séquences en sortie	63 488	351 744	3 584
Pourcentage d'acceptation	86,11%	23,08%	0,02%
Pourcentage de la population passant le filtre	15,15%	83,9%	0,8%
	Jeu Ref0	Jeu Ref1	Jeu Ref2
Séquences en entrée	13	648	17 842
Séquences en sortie	13	585	12 103
Pourcentage d'acceptation	100%	90%	67,8%
Pourcentage de la population passant le filtre	0,1%	4,6%	95,2%

TABLE 2.4 – Quantité de séquences acceptées par la matrice LXRE13 pour les séries ERX et RefX

la moitié (54%) des séquences Ref1 est acceptée par l'expression régulière (ER0). Par ailleurs, en autorisant une substitution dans l'expression régulière (ER0 $\cup$ ER1), on constate que la majorité des séquences proches des références est acceptée. Ainsi, l'ensemble des séquences Ref0 et Ref1 et 80% des Ref2 sont acceptées par ER0 ou ER1.

Résultat 1

Sur l'exemple de LXRα, l'expression régulière avec 1 substitution paraît suffisante pour accepter un voisinage raisonnable des séquences de référence.

2.4 Combien de séquences des séries ERX et RefX sont acceptées par la matrice LXRE13 ?

Les matrices calculent un score pour tous les hits possibles, même les plus éloignés des références. Pour cette raison, les outils de reconnaissance de motifs matriciel proposent un filtre par défaut duquel les hits non-pertinents sont automatiquement rejetés. L'outil utilisé ici, RSAT, permet de filtrer les hits obtenus en fonction de leur valeur-p. Par défaut, ce filtre est automatiquement fixé à 10^{-4} : les hits trop éloignés de la matrice et ayant une valeur-p au-dessus de ce seuil ne sont pas retenus.

Une première question est donc de savoir combien d'éléments de chacun des jeux des différentes séries sont retenus avant toute application d'un filtre par l'utilisateur.

Résultat 2

Il ressort de nos tests (cf 2.4) que les séquences ER0 passent presque toutes le filtre automatique de la matrice mais que 14% des séquences sont tout de même rejetées. Les ER1 sont minoritairement (23%) acceptées et quasiment aucun ER2 (0,02%). Par ailleurs, sur la quantité globale de séquences acceptées, les ER0 ne représentent qu'une faible partie (15%) en raison des volumes de départ différents des jeux de données ERX.

On remarque par ailleurs qu'en toute logique, les séquences proches des références (série RefX) passent facilement le filtre automatique de la matrice. Cela dit, toutes les séquences très proches des références ne passent pas le filtre. Ainsi, 10% des Ref1 (63 séquences) sont directement rejetées (exemple de séquence rejetée par la matrice LXRE13 : *TGAACTtgagT-GAC**T**A*)

2.5 À quoi ressemblent les séquences des séries ERX et RefX acceptées par la matrice LXRE13 ?

Les séquences des jeux de données ER0, ER1 et ER2 ont été analysées afin de caractériser la raison pour laquelle elles passaient ou non le filtre de valeur-p de 10^{-4}. Ce sont logiquement les poids associés à chaque position de la matrice qui ont permis de déterminer le destin de chaque séquence (cf table 2.2).

Les séquences qui cumulent de nombreux choix faibles, par exemple : *CGCACAnnnnT-GATCA*, et qui ressemblent finalement peu aux séquences de référence se retrouvent automatiquement rejetées. De façon plus précise, les séquences cumulant plus de 3 choix faibles (ayant un poids de 2/13 ou moins) n'ont pas été retenues.

Inversement, les ER1 et les rares ER2 ayant passé le filtre sont ceux qui compensent la présence d'un hit ne respectant pas les choix imposés de la matrice par la présence du meilleur choix possible à de nombreuses autres positions. Par exemple, *TGACCGacgtTAACC**G*** chez les ER1 ou *TGACCGacgtTAAC**AG*** chez les ER2.

Ces résultats montrent qu'*a contrario* de l'expression régulière, qui accepte toutes les possibilités permises, la matrice poids-position filtre raisonnablement les mauvais hits (càd les hits qui s'écartent fortement des références). Ainsi 20% des séquences ER1 et la quasi-totalité des séquences ER2 sont éliminées par le filtre automatique de la matrice LXRE13 alors que dans le même temps, seuls 10% des séquences Ref1 et et 30% des séquences Ref2 sont rejetées, or la série ERX représente toutes les séquences reconnues par l'expression régulière tandis que la série RefX représente les séquences véritablement proche des références. Les séquences RefX, proches des références, passent donc finalement mieux le filtre automatique que l'ensemble des séquences ERX.

Au vu de ces résultats, il est intéressant de constater quelle est la répartition des séquences RefX ayant passé le filtre au sein des séquences ERX (cf table 2.5).

Ref acceptées par la matrice	**ER0** (62 879 seq.)	**ER1** (341 647 seq.)	**ER2**(3 242 seq.)
Ref0 (13/13 seq., 100%)	100% (13 seq.)	-	-
Ref1 (585/648 seq., 90%)	60% (351 séq.)	40% (234 séq.)	-
Ref2 (12 103/17 842 seq., 67%)	42,77% (5 177 séq.)	56,95% (6 892 séq.)	0,28% (34 séq.)

TABLE 2.5 – Répartition des sous-séquences de la série RefX acceptées par la matrice LXRE13 (i.e. passant le filtre de valeur-p de 10^{-4}) parmi les différentes séries ERX

Résultat 3

En comparant avec la répartition initiale (cf table 2.3), on constate que la plupart des Ref2 de type ER2 ont été rejetées par la matrice, ce qui implique qu'après ce filtre par défaut, la série ER2 ne contient quasiment plus de séquences d'intérêts (34 séquences sur un total de 12 700 séquences d'intérêt restantes). Par ailleurs, on constate que la quasi-totalité des RefX de type ER0 ont été conservées.

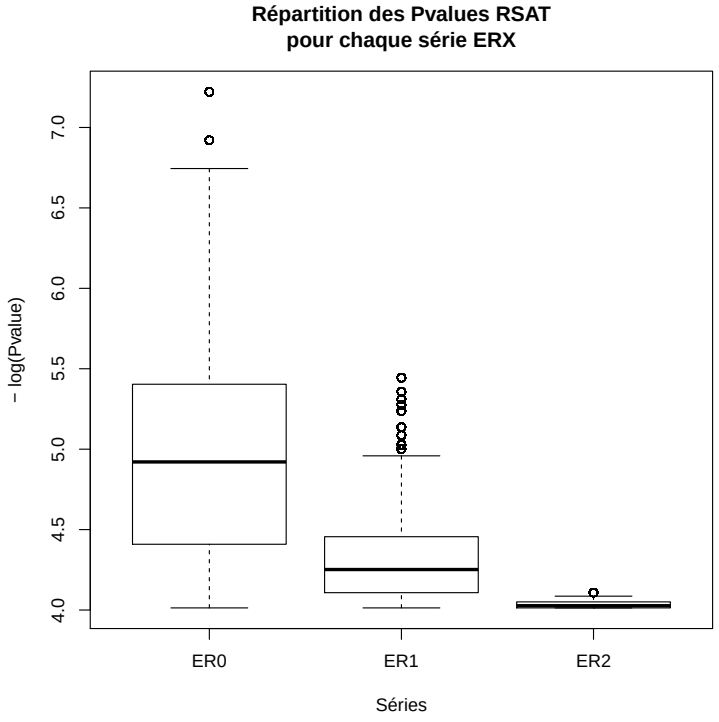


FIGURE 2.4 – Boxplot des valeurs-p calculées par RSAT pour les séquences des jeux de données ER0, ER1 and ER2.

2.6 Comment se répartissent les séquences des séries ERX et RefX en termes de valeur-p ?

Bien que chaque jeu de données ER0, ER1 et ER2 ait des éléments acceptés par la matrice LXRE13 et donc ayant passé le filtre automatique de RSAT, la question demeure de savoir si les hits acceptés de l'ensemble ER2 sont acceptés avec la même failité que ceux de l'ensemble ER0. Autrement dit, il s'agit de regarder si les hits possèdent le même type de valeur-p en fonction de leur nature (respectant la matrice ou s'en écartant) et s'ils sont davantage groupés autour de valeurs-p clefs ou, à l'inverse, répartis sur tout le spectre possible des valeurs-p. Pour le vérifier, l'ensemble des valeurs-p de hits ERX a été porté sur un boxplot [PHKD06] (cf 2.4) qui indique la répartition de la population de chaque série autour des différentes valeurs-p (exprimées selon un échelle logarithmique pour une meilleure lisibilité).

Ce diagramme nous montre que les valeurs-p du jeu de données ER0 couvrent l'ensemble des valeurs-p possibles, tandis que les valeurs-p des jeux de données ER1 et ER2 sont davantage ramassées vers les valeurs-p plus mauvaises. On s'attendait à ce que du fait de leurs mutations, qui affectent négativement le calcul des séquences des jeux ER1 et ER2, les séquences ER1 et ER2 aient de mauvaises valeurs-p. Par contre, la répartition régulière des scores ER0 est plus étonnante : on aurait pu la penser concentrée autour des meilleures valeurs-p.

La question sous-jacente qui se pose est de savoir à quoi ressemblent les hits aux frontières de ce box-plot. Si on prend la médiane du jeu de données ER0 (-log(valeur-p) de 4,9), une étude des hits situés aux environs de ce score indique que ceux-ci peuvent cumuler jusqu'à 4 choix non-optimaux voire 2 choix faibles (poids de 2 ou 1) par rapport à la matrice, tel que

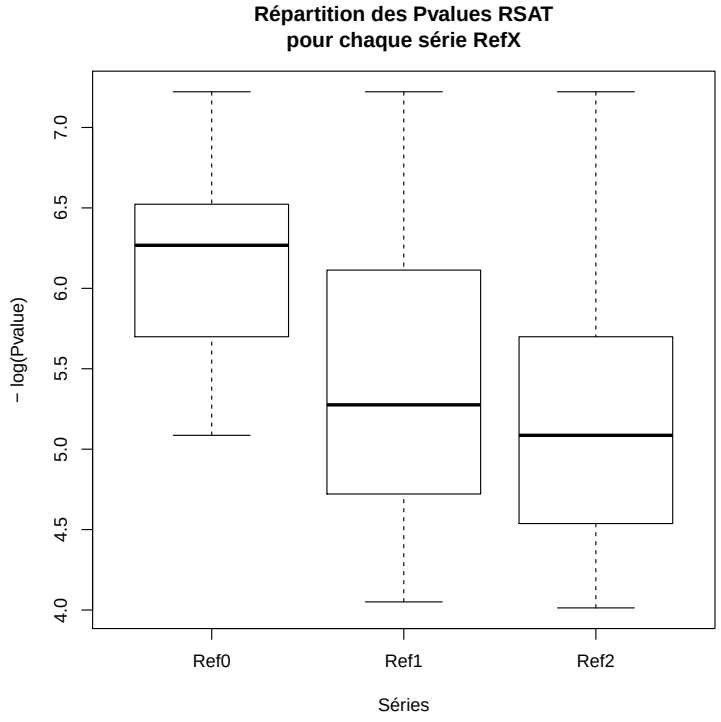


FIGURE 2.5 – Boxplot des valeurs-p calculées par RSAT pour les séquences des jeux de données Ref0, Ref1 and Ref2.

C*GA****A****C****A****acgtTGA****T****CT*. Dans le même temps, on obtient aussi des hits ER1, composés d'une mutation et d'un choix faible (poids de 1/13), tel que *TGACC****A****gaagTAACC****G***, ou bien une mutation et deux choix non-optimaux, tel que *TGACCGacgtT****G****A****A****C****T***.

Si on remonte jusqu'à la valeur du 3ème quartile (-log(valeur-p) de 5,4), alors les séquences comptent jusqu'à 3 choix non-optimaux, tel que ***C****GA****A****CAacgtTAACCT*.

Au final, en remontant le seuil de validation à 5,4 (quartile3 du jeu ER0), la majorité des hits obtenus respecte les choix de la matrice et ces hits ressemblent donc fortement aux références. Inversement, en descendant à un seuil de validation de 4,4 (quartile1 du jeu ER0), on obtient des hits plus divergents de la matrice, y compris de nombreux hits possédant un choix non-autorisé par la matrice (25% des hits ER1 ont une valeur -log(valeur-p) supérieure à 4,45). La connaissance des types de hits obtenus en fonction du score permet donc de choisir plus efficacement un seuil de coupure pour sélectionner les séquences par rapport à leur proximité à la matrice.

De la même façon, on regarde la répartition des scores des séquences dérivant directement des références (Ref1 et Ref2). À nouveau, toutes les valeurs-p sont représentés (cf figure 2.5). Néanmoins, les médianes des valeurs sont respectivement de 5,2 et 5,0 pour Ref1 et Ref2 : elles sont donc plus élevées que pour les jeux de données ER1 et ER2, donc leurs séquences sont globalement plus proches de la matrice.

Résultat 4

La valeur de 4,9 correspond à la valeur maximum du boxplot du jeu ER1 et seules les séquences ER0 ont de meilleurs scores, ce qui veut dire que les meilleures séquences à 2 degrés de liberté des références sont intégralement comprises dans ce qui est permis par l'expression régulière (cf figure 2.1).

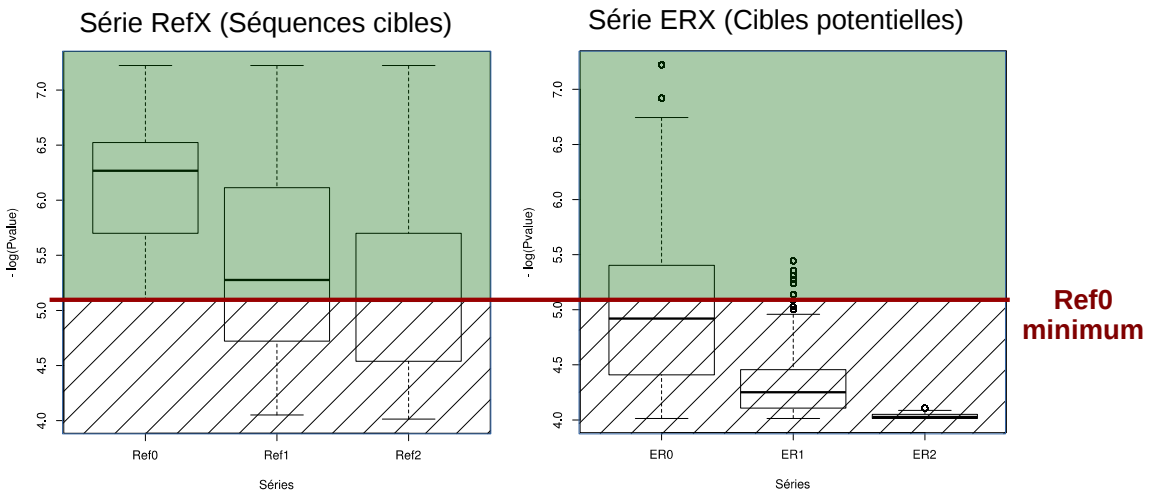


FIGURE 2.6 – Boxplot des valeurs-p calculées par RSAT pour les séquences des jeux de données RefX et ERX

2.7 Comment fixer un score utilisateur pertinent ?

Le filtre automatique appliqué par RSAT élimine par défaut les hits les moins pertinents vis-à-vis de la matrice, mais il revient à l'utilisateur de déterminer où placer la limite pour valider ou non un hit en fonction de ses besoins, càd de décider à quel point il accepte de s'éloigner de ses références. Dans notre cas, il s'agit de favoriser la sélection des séquences de la série RefX tout en limitant la quantité de séquences des séries ER1 et ER2 qui contiennent massivement des séquences ne correspondant pas à nos cibles (cf tableau 2.3)

Si on regarde la répartition des valeurs-p en fonction des différentes séries, on constate que la limite inférieure de la série Ref0 (c-est-à-dire des séquences de référence) est à 5,1. Cette valeur est inférieure à la médiane de la série Ref1 (5,2) et est identique à la médiane de la série Ref2. Dans le même temps, elle est strictement supérieure au dernier quartile des séries ER1 et ER2 et se situe au-dessus de la médiane de la série ER0.

Résultat 5

Avec la matrice LXRE13, via RSAT, la plus mauvaise valeur des références (Ref0) permet donc de sélectionner la majorité des séquences RefX tout en limitant de façon stringente l'ajout de faux-positifs issus des séries ERX.

2.8 Influence du background

Le modèle matriciel via RSAT présente l'avantage de pouvoir prendre en compte l'influence du background nucléotidique des séquences lors du calcul de la valeur-p. Tout au long de cette étude, nous avons utilisé un background neutre où chaque nucléotide possède une fréquence de 25%. Une telle répartition n'étant pas réaliste, nous avons voulu savoir à quel point le background pouvait impacter les résultats que nous avons mis en évidence dans l'étude précédente.

Pour cela, nous avons étudié deux backgrounds particuliers : l'humain et la souris.

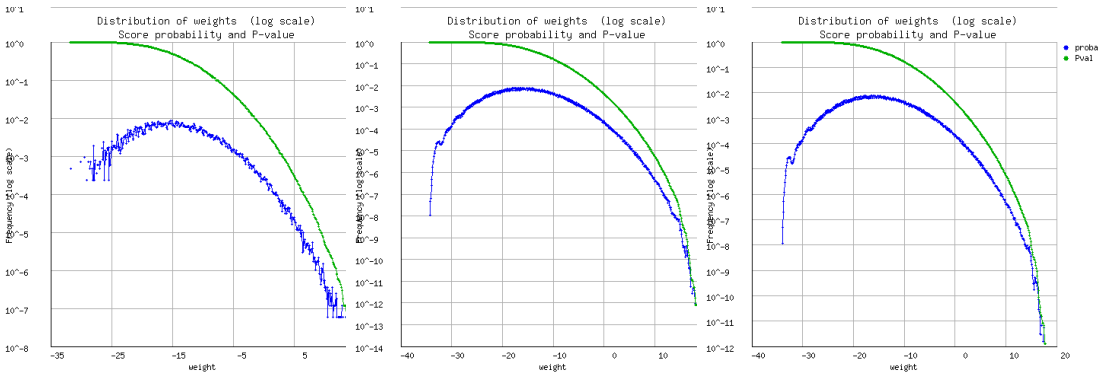


FIGURE 2.7 – Amplitude de la valeur-p possible pour la matrice LXRE13 en fonction du background nucléotidique utilisé *(A : 25%, B : Homo sapiens, C : Mus Musculus)*

2.8.1 Les valeurs-p sont-elles modifiées en prenant un background réel ?

L'une des questions qui nous a intéressés concerne l'amplitude des valeurs-p des hits. Les résultats sont-ils influencés par le background choisi ? Pour le savoir, nous avons utilisé le logiciel matrix-distrib de RSAT (cf p 27).

Résultat 6

On constate tout d'abord que le background à 25% aboutit à un profil nettement différent de celui de l'humain et de la souris, avec des valeurs-p atteignant au minimum 10^{-8} contre 10^{-12} dans le cas des backgrounds réalistes. Le background neutre présente donc l'inconvénient de tasser les valeurs par rapport à la réalité (cf figure 2.7).

2.8.2 Peut-on fixer un seuil satisfaisant avec lesbackgrounds réels ?

La différence d'amplitude observée suite à l'utilisation de différents backgrounds nous a conduit à vérifier de quelle façon cela se traduisait en terme de répartition des valeurs-p au sein des différentes séries RefX et ERX (cf figure 2.8 pour le background de l'humain et figure 2.9 pour le background de la souris).

Tout d'abord, on constate que de façon générale, les différents degrés de substitution d'une même série ont tendance à avoir des valeurs-p plus chevauchantes : il n'y a plus de séparation bien nette en fonction du nombre de substitutions.

Cette différence engendrée par les backgrounds réels posent la question de savoir si notre façon de fixer un score de validation raisonnable est toujours valable ou non.

Seuil de validation avec un background humain

Le -log(valeur-p) le plus bas de la série Ref0 est à 5,1 en utilisant un background humain. Cette valeur est inférieure au troisième quartile des Ref1 (5,2) et à la médiane des Ref2 (5,4) : elle permet donc toujours d'obtenir la majorité des séquences RefX, qui constituent les cibles qu'on souhaite retrouver. Par rapport aux séries ERX, cette valeur est légèrement inférieure à la médiane de la série ER0 (5,3) et est exactement identique au dernier quartile de la série ER2 (5,1). Elle est par contre légèrement supérieure au troisième quartile de la série ER1 (5), ce qui implique qu'on obtient presque 25% des séquences ER1.

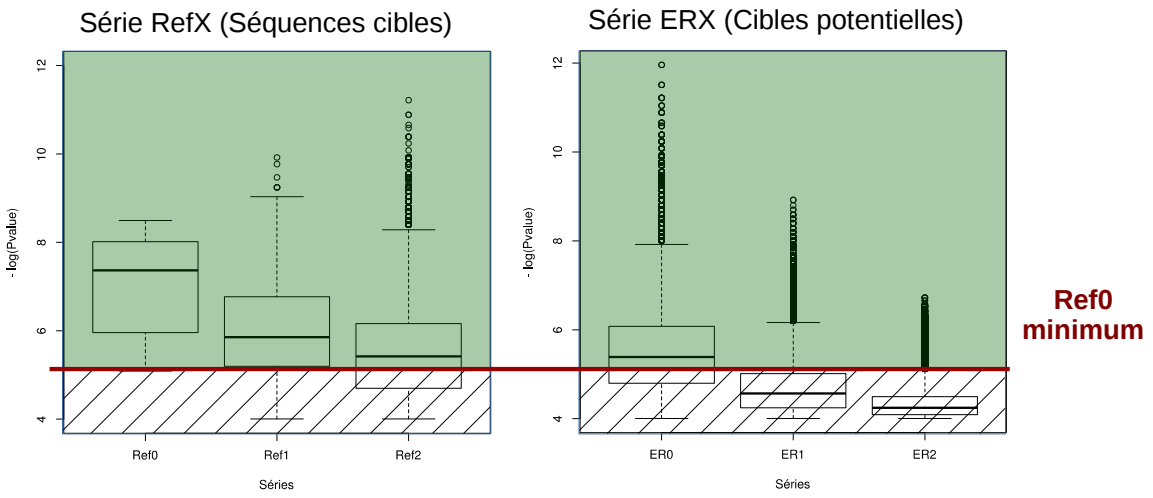


FIGURE 2.8 – Boxplot des valeurs-p calculées par RSAT avec un background humain pour les séquences des jeux de données RefX et ERX

Seuil de validation avec un background souris

Le -log(valeur-p) le plus bas de la série Ref0 est à 5,19 en utilisant un background souris. Cette valeur est supérieure au troisième quartile des Ref1 (5,10) et est légèrement inférieure à la médiane des Ref2 (5,24) : elle permet donc toujours d'obtenir la majorité des séquences RefX, qui constituent les cibles qu'on souhaite retrouver. Par rapport aux séries ERX, cette valeur est légèrement inférieure à la médiane de la série ER0 (5,24) et est supérieure au dernier quartile de la série ER2 (4,36) et ER1 (4,82).

Résultat 7

L'utilisation de différents backgrounds réels engendre des différences dans la valeur-p relative des hits de la matrice LXRE13. Notre méthode, qui consiste à se baser sur la plus mauvaise valeur-p de nos références (Ref0) pour fixer un seuil de validation raisonnable, permet d'obtenir la majorité des séquences cibles désirées. Néanmoins, dans certains cas, cela va de pair avec une partie des séquences ER1, qui abritent de nombreuses séquences bruitées. Sans être parfaite, cette façon de fixer un seuil de validation aboutit tout de même à une bonne séparation des séquences bruitées et des séquences cibles.

2.9 Conclusion

Dans ce chapitre, au travers de l'exemple de la recherche du site de fixation du facteur de transcription LXRα et de l'étude statistique de la répartition des hits obtenus, nous avons souhaité propager une méthode pratique pour sélectionner des hits raisonnables, c-à-d proches des références mais permettant suffisamment de flexibilité pour identifier des variants à deux degrés de substitution.

Pour cela nous avons comparé deux approches : la méthode classique, par l'utilisation de matrices, et une approche basée sur les modèles grammaticaux, telles que les expressions régulières approchées. Pour cela nous avons déterminé quelles étaient les cibles de chaque

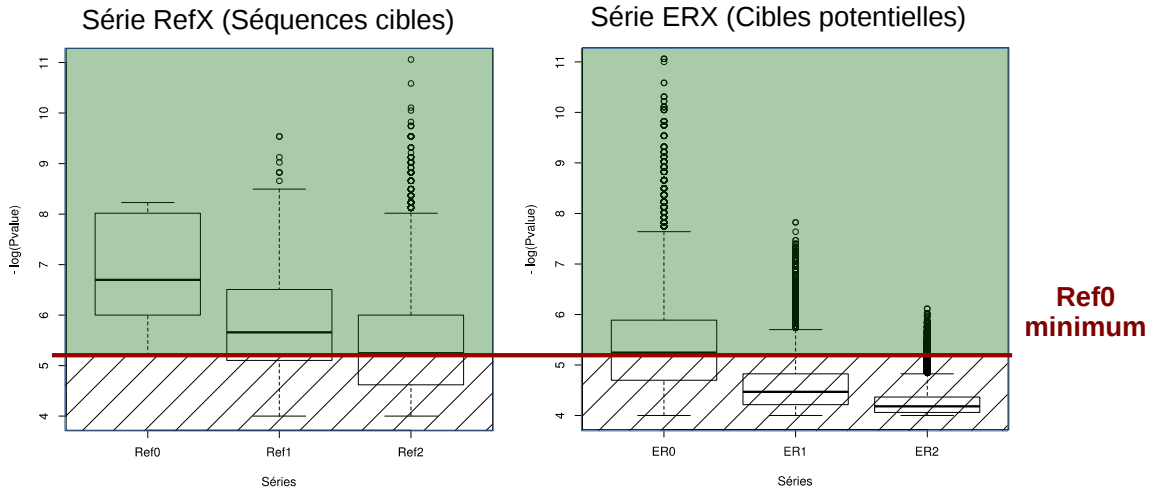


FIGURE 2.9 – Boxplot des valeurs-p calculées par RSAT avec un background souris pour les séquences des jeux de données RefX et ERX

méthode et à quel point ces cibles étaient conformes aux variants que nous souhaitions isoler.

Dans ce but, nous avons construit deux jeux de données : la première série, ERX, regroupe toutes les séquences reconnues par l'expression régulière, jusqu'à deux substitutions près, càd toutes les séquences qui possèdent des nucléotides plus ou moins attendus à chaque position, tandis que la seconde série, RefX, regroupe toutes les séquences proches des 13 références validées biologiquement du site de fixation LXRα, jusqu'à deux substitutions près, càd toutes les séquences que nous voudrions réellement cibler.

Sur le cas d'étude, la comparaison de ces deux jeux de données a mis en évidence que la grande majorité des séquences RefX, y compris à 2 substitutions d'une référence, est couverte par l'expression régulière avec au plus 1 substitution : **il n'est donc pas pertinent de se préoccuper des séquences proches de l'expression régulière avec 2 substitutions**. Globalement, autoriser 1 substitution permet de cibler la majorité des variants des références (Cf Résultat 1) et **l'expression régulière stricte permet de cibler les meilleurs variants**, càd ceux qui possèdent une meilleure valeur-p, d'après les résultats de la matrice (cf Résultat 4).

Il apparaît par ailleurs que les séquences RefX, plus proches des références, passent plus facilement le filtre de base 10^{-4} de RSAT que les séquences ERX : ainsi, si seules 20% des séquences couvertes par l'expression régulière avec 1 substitution sont conservées, 90% et 66% des séquences dérivant des références avec respectivement 1 et 2 substitutions sont malgré tout conservées. En outre, les séquences RefX disposent globalement de valeurs-p plus élevées que les séquences ERX, ce qui signifie qu'il est plus facile de sélectionner ces cibles. **L'utilisation d'un filtre égal à la valeur-p la plus mauvaise parmi les références permet ainsi d'isoler jusqu'à 75% des variants des références avec 2 substitutions**, tout en limitant fortement la sélection des séquences potentielles bruitées (rejet de la totalité des séquences ER2 et de la majorité des séquences ER1) (cf Résultats 2 et 3).

Ainsi, en comparant les résultats obtenus sur les séquences potentiellement ciblées par la matrice et les séquences proches des références qu'on souhaiterait cibler, nous avons pu

déterminer un score seuil (cf Résultat 5) qui nous permet de resserrer de façon efficace les séquences conservées par la matrice autour des séquences d'intérêt en minimisant la quantité de séquences potentiellement moins intéressantes pour notre problématique (l'identification de séquences proches de la fusion des références).

Ainsi, **l'approche matricielle s'avère plus pertinente que l'approche par les expressions régulières approchées pour détecter les variants de séquences de références un peu éloignées**. En effet, la matrice permet une meilleure granularité des résultats par l'intermédiaire des valeurs-p que les expressions régulières par l'intermédiaire du nombre de substitutions. Cette granularité permet de fixer de façon plus fine un seuil de validation pertinent. Néanmoins, si l'expression régulière approchée n'est pas la solution la plus efficace pour la recherche de variants, **une approche grammaticale par boule de mots** (c'est-à-dire rechercher tous les variants à deux substitutions près de chaque référence prise individuellement) pourrait donner d'excellents résultats, puisqu'elle ciblerait strictement notre définition des séquences voisines des références (les RefX). Cette stratégie n'est cependant pas sans contrepartie : les séquences voisines ne constituent que $\sim$8% des séquences reconnues par l'expression régulière stricte, or $\sim$25% des séquences reconnues par l'expression régulière ont de bons scores d'après la matrice et pourraient être des candidats vraisemblables, qui seraient ignorés par les boules de mots. Les boules de mots sont donc un modèle alternatif intéressant mais très spécifique.

Dans le chapitre suivant, nous allons poursuivre l'étude de l'application LXRα. Il ne s'agira alors plus de comparer deux méthodes sur des cibles artificielles, mais plutôt de tenter de résoudre la problématique biologique qui est la recherche de gènes candidats forts pour une régulation directe par le facteur de transcription LXRα. Dans un premier temps, nous montrerons qu'en pratique, il n'est pas pertinent de modéliser le motif DR4 (Direct Repeat) LXRα avec une grammaire. Dans un second temps, nous nous intéresserons aux résultats d'un modèle matriciel du motif LXRα et montrons comment limiter l'avalanche de faux-positifs dans les upstreams des gènes humains en mettant en œuvre trois filtres différents basés sur l'orthologie et sur un principe de conservation des séquences au cours de l'évolution.

CHAPITRE 3

Identification de cibles d'un facteur de transcription : limiter les faux-positifs

Dans ce chapitre, nous nous intéressons à la façon de limiter le nombre de faux-positifs dans une recherche de sites de fixation d'un facteur de transcription dans des régions amonts des gènes eucaryotes, afin de mettre en évidence un petit nombre de gènes potentiellement régulés par le facteur.

Dans la section 5.1, nous montrons que le motif LXRα, bien qu'étant un DR4 (Direct Repeat), n'est pas un motif sous forme répétée et qu'il n'est pas pertinent de le modéliser à l'aide de grammaire. Nous faisons alors le choix de le modéliser sous la forme d'une matrice, motivé par l'étude du chapitre 2. Dans le présent chapitre, nous montrons comment limiter l'avalanche de faux-positifs par l'intermédiaire de différents filtres basés sur l'orthologie et la conservation des séquences fonctionnelles (à l'aide de filtres basés sru la valeur-p, l'orthologie et l'homogénéité) pour arriver à une liste de gènes de confiance suffisamment courte pour pouvoir faire l'objet d'une validation biologique.

Les travaux présentés dans ce chapitre font l'objet d'un article en cours de soumission dans la revue BMC Bioinformatics : "Orthocis : A browser of genomic cis-regulatory elements using orthology for any number of species"

L'interface web Orthocis permettant d'employer les différents filtres sur les résultats de différents Facteurs de Transcriptions (FT) est disponible à l'adresse suivante :

https ://steamcat.genouest.org/hosted/orthocis2/

Dans ce chapitre, nous présentons une des deux applications que nous avons développées de façon poussée pendant cette thèse : la détection des sites de fixation d'un facteur de transcription dans le but de prédire de nouveaux gènes régulés directement par le facteur.

Comme pour chacune des applications explorées, le point de départ a été la possibilité d'utiliser un modèle grammatical via Logol. Bien que Logol ne se soit finalement pas révélé adapté, cette étude a été poussée jusqu'à obtenir un outil opérationnel : Orthocis. C'est ce développement qui est présenté dans ce chapitre. Les modèles testés en Logol sont présentés dans la partie 5.1.

La problématique étudiée ici est moins la reconnaissance de motifs elle-même que le filtrage des hits. En effet, la recherche de sites de fixation de facteur de transcription (Transcription Factor Binding Site, TFBS) génère de très nombreux hits dans les séquences, dont beaucoup

peuvent s'avérer être des faux-positifs [Bul04, MKPW13], or les biologistes ont besoin de candidats fiables pour effectuer la validation biologique. L'objectif est de limiter ces faux-positifs en filtrant les résultats sur la base de nouvelles contraintes. Les contraintes proposées ici exploitent l'hypothèse de conservation de la régulation des gènes au cours de l'évolution : un hit pour un gène d'une espèce d'intérêt sera donc considéré comme valide si on le retrouve de manière similaire dans d'autres espèces voisines.

Par l'utilisation conjointe des informations d'orthologie et de l'homogénéité des hits obtenus pour des gènes orthologues parmi un set d'espèces d'intérêt, nous parvenons à obtenir une sous-liste de gènes possédant un hit valide pour une régulation directe par le facteur de transcription LXRα

Ces travaux ont été menés en collaboration avec Sandrine Lagarrigue et Frédéric Lecerf de l'unité PEGASE de l'INRA de Rennes. Ils ont été initiés dans le cadre de l'ANR Fatinteger (2012-2015) dirigée par Florence Gondret, qui porte sur l'étude du métabolisme des lipides chez les Eucaryotes.

3.1 Modélisation du TFBS LXRα

L'objectif du projet est de parvenir à prédire des gènes régulés de façon directe par des facteurs de transcription dans les espèces eucaryotes. Puisque la reconnaissance de motifs de TFBS génère un grand nombre de faux-positifs, les biologistes souhaitent obtenir une liste suffisamment restreinte de candidats fiables afin de mener des validations expérimentales. Nous avons travaillé sur le facteur de transcription LXRα, car il est l'un des facteurs pour lequel nos collaborateurs biologistes possèdent une expertise poussée. LXRα est un facteur fortement impliqué dans la régulation du métabolisme des lipides, dont une partie des cibles est déjà connues. Il nous a servi d'exemple pour mettre au point une stratégie de prédiction que nous voulons applicable à tous les facteurs de transcription pour lesquels on connaît des références du site de fixation.

3.1.1 Séquences de référence du motif LXRα

Pour établir le modèle du site de fixation de LXRα, une recherche de l'ensemble des sites de fixation connus au travers de la littérature a été effectuée. Un ensemble de 13 séquences de référence, validées expérimentalement, a été retenu. D'après la littérature [DLD$^+$11], la protéine LXRα reconnaît un motif de type DR4, c'est-à-dire un motif composé d'une répétition directe, séparée par 4 nucléotides non-définis. L'élément répété de LXRα est long de 6 nucléotides. Le motif TFBS totalise donc une taille de 16 nucléotides.

3.1.2 Jeu de gènes contrôle pour LXRα : gènes différentiellement exprimés

Nous disposons d'une liste de 840 gènes très fortement différentiellement exprimés (DE) entre des souris « sauvages » et des souris chez qui le gène LXRα a été « éteint ».

Une simple expression différentielle d'un gène n'implique pas forcément la présence d'un motif du TBFS de LXRα dans l'upstream de ces gènes, puisqu'un gène peut être régulé de façon indirecte par le FT. Néanmoins, nous avons posé l'hypothèse que la proportion de séquences directement régulées par LXRα est plus forte dans l'ensemble des gènes DE que dans l'ensemble des gènes non DE.

Id	Ensembl Id	Species	Sequence	Origine	valeur-p
01	Cyp7Alpha1	Souris	TGAACTtgggTGACCA	[CKS01]	$2e^{-5}$
02	Cyp7Alpha1	Rat	TGAACTtgagTGACCA	[CKS01]	$2,1e^{-5}$
03	FASN	Souris	TGACCGgtagTAACCC	[DDD$^+$09]	$8,4e^{-9}$
04	FASN	Rat	TGACCGgtagTAACCC	[JLP$^+$02]	$1,1e^{-8}$
05	FASN	Poule	TGACCTgtggTAACCT	[JLP$^+$02]	$4,8e^{-7}$
06	FASN	Humain	TGACCGgcagTAACCC	[JLP$^+$02]	$2,4e^{-8}$
07	LPCAT3	Humain	CGACCGggagTAACCT	[DLD$^+$11]	$2,4e^{-8}$
08	LPCAT3	Souris	CGACCGagagTAACCT	[DLD$^+$11]	$1,8e^{-8}$
09	LPCAT3	Rat	CGACCGagagTAACCT	[DLD$^+$11]	$2,2e^{-8}$
10	LPCAT3	Poule	TGCCCGcgagTAACCC	[DLD$^+$11]	$7,3e^{-8}$
11	CETP	Humain	TGCCCGacaaTGACCC	[LT00]	$7,8e^{-7}$
12	CYP51a	Humain	TGACCTcaggTGATCC	[WRS$^+$08]	$8,3e^{-6}$
13	SCD1	Souris	TGACCAcaggTAACCT	[CMMN06]	$2,4e^{-6}$
	Consensus		TGACCGnnnnTAACCC	[CMMN06]	

TABLE 3.1 – Gène, espèce et hit des 13 séquences de référence LXRα. Les nucléotides en minuscule représentent les 4 nucléotides centraux qui, en principe, n'interviennent pas dans la fixation du facteur LXRα et dont la nature n'est donc pas sensée avoir d'impact sur la reconnaissance du site

3.1.3 Modèle du TFBS LXRα

Bien que LXRα soit considéré comme étant un motif de type DR4, dans les faits, le motif répété n'est pas conservé et se prête mal à une modélisation sous forme de grammaire exploitant ce concept de répétition. Cette modélisation est discutée plus en détail dans la partie 5.1.

Par ailleurs, nous avons vu dans le chapitre 2 que les modèles grammaticaux sont peu pertinents pour modéliser un motif dont les références présentent autant de positions variables et que les matrices étaient une approche préférable. Nous avons donc fait le choix de modéliser le TFBS LXRα sous la forme d'une matrice de score.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
A	0	0	11	2	0	1	3,25	3,25	3,25	3,25	0	9	13	0	0	2
C	3	0	2	11	13	0	3,25	3,25	3,25	3,25	0	0	0	12	13	6
G	0	13	0	0	0	8	3,25	3,25	3,25	3,25	0	4	0	0	0	0
T	10	0	0	0	0	4	3,25	3,25	3,25	3,25	13	0	0	1	0	5

FIGURE 3.1 – Matrice LXRE13 : matrice de comptage établie à partir des 13 séquences de références. Les 4 nucléotides centraux ont reçu des valeurs identiques afin que ces positions n'influent pas sur les scores des hits

3.2 Limitation des faux-positifs par la contrainte de conservation au cours de l'évolution

La recherche du TFBS avec la matrice sur un génome eucaryote, même en se limitant aux zones upstreams des gènes, rapporte de trop nombreux hits, le motif TFBS étant un signal faible. Nous avons défini un filtrage basé sur le principe de conservation au cours de l'évolution pour limiter le nombre de faux-positifs. Sa mise en œuvre se fait en deux temps : dans un

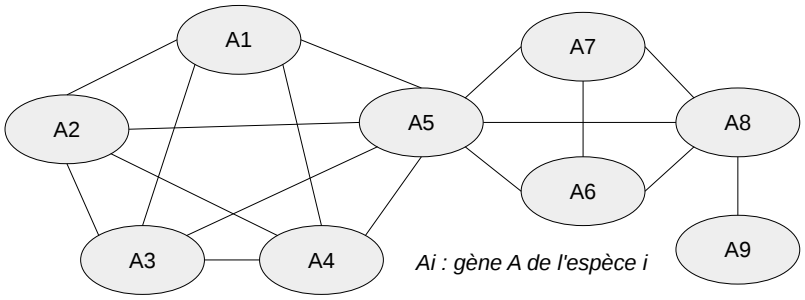


FIGURE 3.2 – Exemple de relations d'orthologies pour le gène A parmi différentes espèces i. {A1, A2 A3, A4, A5}, {A5, A6, A7, A8} et {A8, A9} forment les plus grandes cliques d'orthologie parmi ces relations.

premier temps, une sélection de gènes se fait au niveau des espèces, par l'orthologie. Les gènes retenus sont ceux qui sont suffisamment conservés pour conserver un lien d'orthologie direct avec d'autres espèces, c'est-à-dire, si l'espèce possède un gène commun hérité d'une même espèce ancestrale et ayant suffisamment peu évolué pour rester similaire [VSUV$^+$09]. Dans un second temps, la sélection se fait au niveau des TFBS identifiés pour un gène parmi plusieurs espèces, qui doivent alors posséder un minimum de similarité pour respecter le principe de conservation.

3.2.1 Conservation au niveau des espèces : filtre d'orthologie

Une première conservation est vérifiée au travers des liens d'orthologie : un gène est conservé pour un ensemble d'espèces si l'ensemble des espèces forment une clique d'orthologie pour le gène considéré, c-à-d si chaque espèce partage un lien d'orthologie one-to-one pour le gène considéré avec chaque autre espèce.

Par exemple, la figure 3.2 représente les liens d'orthologie one-to-one pour le gène A parmi 9 espèces. Les espèces 1, 2, 3, 4 et 5 forment une clique d'orthologie : chacune des espèces possèdent un lien d'orthologie direct deux à deux avec chaque autre membre de la clique. De la même façon, les espèces 5, 6, 7 et 8 forment une clique d'orthologie. Inversement, les espèces 1, 5 et 7 ne partagent pas de liens d'orthologie deux-à-deux avec chacun des autres et ne forment donc pas une clique d'orthologie.

Étant donné un ensemble d'espèces, les cliques présentes sur ces espèces pointent des gènes bien conservés sur cet ensemble d'espèces.

3.2.2 Conservation au niveau du TFBS : filtre d'homogénéité

Nous avons posé l'hypothèse que si un FT est conservé au cours de l'évolution, alors les sites de fixations des gènes qu'il cible sont probablement conservés eux aussi en termes de séquences.

Ceci nous a conduit à élaborer un filtre basé sur la similarité des hits, qu'on appelle l'homogénéité. L'idée étant que les hits positifs au sein de gènes orthologues doivent former un ensemble relativement homogène (i.e. qu'ils n'ont pas ou peu divergé).

Le calcul de l'homogénéité est basé sur celui de l'entropie relative de chaque position de l'alignement des hits, qui quantifie l'information détenue par chaque position de l'alignement,

selon la formule de la figure 3.3.

(1)
$$I^i = \sum_{b\in\{A,T,C,G\}} P(i) log_2 \frac{P(i)}{Q(i)}$$

(2) $$I^i = \sum_{b\in\{A,T,C,G\}} \frac{n_b^i}{n} log_2 \frac{4n_b^i}{n}$$

FIGURE 3.3 – (1) Formule de l'entropie relative. Avec P(i) : probabilité d'apparition des nucléotides à la position "i" et Q(i) : probabilité théorique d'apparition des nucléotides à la position "i". (2) Application de la formule pour un background théorique avec une équi-répartition de la fréquence entre 4 nucléotides A, T, C et G et "b" : nucléotide considéré, "i" : position au sein du motif et "n" : nombre de références alignées.

La valeur d'entropie pour chaque position varie de 2 pour la présence d'un unique nucléotide à 0 pour la présence de 4 nucléotides représentés de façon égale.

Dans le cas de notre motif de taille 16, l'entropie est donc comprise entre 0 (pour chacune des 16 positions, chaque nucléotide est équitablement représenté parmi les 4 possibilités) et 32 (pour chacune des 16 positions, un seul nucléotide est représente parmi les 4 possibilités).

Un seuil d'entropie fixé à 28 correspond par exemple au score d'une séquence pour laquelle 2 positions ne donnent aucune information. Cela signifie que plusieurs positions de l'alignement peuvent différer du consensus (Cf figure 3.4).

```
CGCNCCNCGTTGACCT        CGCACCACGTTGACCT
CGCNCCNCGTTGACCT        CGCACCACGTTGACCT
CGCNCCNCGTTGACCT        CGCACCACGTTGACCT
CGCNCCNCGTTGACCT        CGCACTACGTTGACCT
CGCNCCNCGTTGACCT        CGCACCACGTTGACCT
CGCNCCNCGTTGACCT        CGAACAATCCTGACCT
   $  $                    $  $ $$$
```

FIGURE 3.4 – L'alignement théorique de gauche totalise un score d'entropie de 28, avec deux positions qui n'apportent aucune information. L'alignement réel de droite totalise un score d'entropie de 28.15, avec 5 positions variables. Les positions où l'alignement n'est pas parfait sont symbolisées le symbole $

3.3 Résultats biologiques : détection de nouveaux gènes candidats pour une régulation directe par le facteur de transcription LXRα

3.3.1 Trop de faux-positifs lors d'une reconnaissance de motifs sur une espèce

La matrice a été utilisée pour analyser les 10 000 nucléotides en amont de chaque gène (l'upstream, tel que défini par Ensembl) chez l'humain.

Conformément à la stratégie évoquée dans le chapitre 2, les résultats ont été filtrés en utilisant comme seuil la plus mauvaise valeur-p des références. Ce seuil est de $2,1e^{-5}$ (cf Tableau 3.1). On obtient alors 19 059 gènes positifs (76%), dont 1 806 gènes DE (7,7% des positifs).

Cette quantité est considérée comme étant trop élevée par les biologistes et majoritairement due à la présence de faux-positifs.

3.3.2 Réduction des faux-positifs par la vérification de la conservation entre les espèces

Conservation au niveau des espèces

La première application de la conservation au cours de l'évolution consiste à déterminer un ensemble d'espèces et à ne conserver que les gènes qui forment une clique d'orthologie entre ces espèces (i.e. application du filtre d'orthologie).

Dans notre cas, nous avons finalement sélectionné l'ensemble d'espèces suivant (ensemble A) : l'humain, le chimpanzé, le macaque, la souris, le rat et la poule. Cette sélection totalise 3 902 gènes partageant une clique d'orthologie entre toutes les espèces de l'ensemble.

Si on sélectionne ensuite les gènes possédant un "bon" hit (seuil de validation de la valeur-p fixée à $2,1e^{-5}$) en zone upstream. Pour un seuil de validation à $2,1e^{-5}$ et un filtre d'orthologie appliqué à l'ensemble A, on obtient une liste de 288 gènes, dont 44 gènes DE (15,3%). Ce premier filtre permet déjà d'obtenir un enrichissement en gène DE.

Conservation au niveau des gènes

La seconde application de la conservation au cours de l'évolution consiste à vérifier que les hits obtenus au sein des gènes d'une clique d'orthologie sont relativement similaires entre eux. Nous avons fixé le seuil d'entropie à 28 (cf 3.2.2).

Pour un seuil de validation à $2,1e^{-5}$, un filtre d'orthologie appliqué à l'ensemble A et un filtre d'homogénéité de 28, on obtient désormais une liste de 4 gènes, dont 2 DE. Les 2 gènes DE sont LPCAT3, qui fait partie des références, et STOML2 , qui est un nouveau candidat fort.

Les deux autres gènes ayant passé les filtres sont BHLHE40 et MIRLET7D. BHLHE40 est connu pour être un gène intervenant dans le métabolisme des lipides et être régulé par LXRα [NUK+09]. Il n'apparait pas dans notre liste de gènes DE car la différence d'expression du gène entre les deux types de souris n'était pas assez fort. MIRLET7D ne possède pas de lien connus avec le métabolisme des lipides.

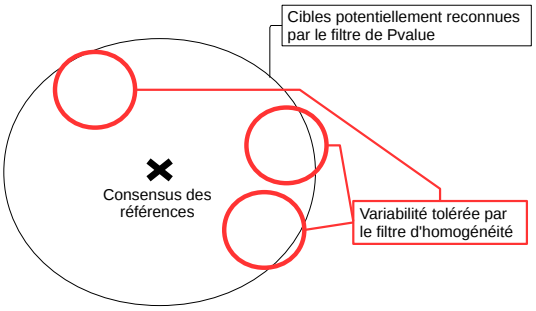


FIGURE 3.5 – Complémentarité des filtres valeur-p et homogénéité.

3.4 Orthocis, une base de données pour exploiter l'hypothèse de conservation au cours de l'évolution

Afin de pouvoir mener ce type de recherche facilement sur tout type de motifs, une base de données dédiée a été construite : Orthocis (cf Annexe A.4). Cette base de données regroupe les upstreams de l'ensemble des 65 espèces présentes dans Ensembl [YAA+16] pour tous les gènes possédant des relations one-to-one d'après Ensembl Compara [HMB+16].

Orthocis utilise l'annotation HGNC [GST+16] (HUGO (HUman Genome Organisation) Gene Nomenclature Committe) pour caractériser les gènes lorsque c'est possible : l'annotation d'un gène humain est répercutée sur l'ensemble des gènes capables de former une clique d'orthologie avec lui.

Par ailleurs, Orthocis contient par défaut l'ensemble des hits sur chaque upstream de tous les TFBS recensés dans la base de données JASPAR [MFA+16] (une base de données qui contient une large gamme de matrices de TFBS, non-redondantes et vérifiées, issues de données publiées ayant fait l'objet d'une validation expérimentale chez les eucaryotes).

Orthocis bénéficie enfin d'une interface web permettant de choisir facilement les ensembles d'espèces, les matrices TFBS et les seuils de valeurs-p ou d'homogénéité. Cette interface permet d'automatiser les requêtes SQL complexes nécessaires pour obtenir les listes de gènes candidats.

3.4.1 Rôle respectif de la sélection sur la valeur-p et sur l'homogénéité des hits

L'intérêt de la sélection sur l'homogénéité des hits dépend grandement de la stringence de la sélection sur la valeur-p. En effet, le filtre d'homogénéité a été mis en place pour déterminer si les hits au sein d'une clique de gènes orthologues vérifient un minimum de conservation. Or, si le seuil de validation de valeur-p est trop élevé, il va naturellement sélectionner les hits les plus similaires au consensus, ce qui réduit alors l'intérêt de filtrer sur l'homogénéité des hits obtenus.

L'utilisation du filtre d'homogénéité n'a donc d'intérêt que si le filtre de valeur-p est un minimum relâché. Dans ce cas de figure, le filtre de valeur-p valide les hits en autorisant un certain degré de variabilité par rapport au consensus. Le filtre d'homogénéité assure alors que, tout en autorisant cette variabilité, les hits d'une clique de gènes orthologues restent relativement conservés entre eux.

Le filtre de valeur-p permet de rechercher plus ou moins largement autour du consensus des références, tandis que le filtre d'homogénéité vérifie que les hits validés ne soient pas éparpillés au sein du champs des séquences visées par le filtre de valeur-p.

3.5 Conclusion

Dans le cadre de la recherche des gènes régulés par un FT, nous avons développé une méthode de filtrages des sites de fixations putatifs, ainsi qu'une base de données couplée à un web-service. Le cas particulier étudié est celui du facteur LXRα.

Résultats biologiques

En nous basant sur l'hypothèse de conservation des signaux fonctionnels au travers de l'évolution, nous avons mis au point un filtre basé sur l'orthologie des gènes entre différentes espèces : on ne retient un gène que s'il y a des orthologues dans les espèces voisines. En étendant cette hypothèse de conservation aux sites de fixation, nous avons mis au point un deuxième filtre basé sur l'homogénéité des hits pour un ensemble de gènes orthologues : on ne retient le gène que si l'ensemble des hits sur les orthologues de ce gène se ressemblent.

L'utilisation conjointe de filtres basés sur la valeur-p, l'orthologie et l'homogénéité a permis de restreindre la liste de gènes candidats à une régulation directe par LXRα a un nouveau candidat, STOML2, dont on sait déjà qu'il est différentiellement exprimé entre des souris "sauvage" et souris pour lesquelles l'expression du gène LXRα a été bloquée. Ce gène fait actuellement l'objet d'une validation biologique.

Création de la base de données et du web-service Orthocis

Cette étude a aussi été l'occasion de mettre en place une base de données dédiée à cette problématique, permettant d'explorer les résultats de n'importe quel facteur de transcription présent dans JASPAR vis-à-vis de n'importe quelle espèce présente dans Ensembl et permettant de filtrer les hits en tenant compte de la conservation au travers des espèces (filtres basés sur l'orthologie). Cette base de données dispose d'une interface web permettant d'utiliser aisément les différents filtres de sélection. Cette interface est disponible sur la plate-forme Genouest à l'adresse : https ://steamcat.genouest.org/hosted/orthocis2/

De nombreuses autres applications proposent ainsi une aide à la recherche des sites de fixation d'un facteur de transcription dans le génome, avec des fonctionnalités très variées. Un certain nombre détecte par exemple la colocalisation de sites de plusieurs facteurs de transcriptions, les CRMs (cis-Regulatory Modules) (ex. oPOSSUM-3 [KAWHW12],TFM-Explorer[TTV10]). Parmi les applications les plus proches d'Orthocis se situent Cotrasif [TGP$^+$09] et Target-Ortho [GWMH14]. Ainsi Target-Ortho permet également de tester la conservation des sites TFBS sur des espèces orthologues, mais il est plus spécialisé qu'Orthocis dans la mesure où il est dédié à deux types d'organismes, la mouche et le ver, sur chacun desquels il met à disposition cinq espèces. Cotrasif met quant à lui vingt espèces à disposition, mais il ne permet par de comparer les sites de TFBS eux-mêmes : il situe la comparaison au niveau des gènes impactés.

Filtrage des faux-positifs

Orthocis propose 3 niveaux de filtrage des faux-positifs : la valeur-p, pour contrôler la distance par rapport au motif TFBS tel que décrit dans la matrice, la sélection d'espèces, pour réduire les gènes ciblés à ceux conservés sur ces espèces, et le critère d'homogénéité, pour pouvoir choisir des sites conservés sur un même gène.

Le site web permet de faire varier la sensibilité en jouant sur ces 3 curseurs. Une autre

façon de filtrer des hits TFBS peut consister à croiser les résultats de *pattern matching* avec des résultats d'expérimentations biologiques (tels que les positions de pics ChIP-Seq ou les positions relatives à l'accessibilité de la chromatine). Nous avons d'ailleurs expérimenté de tels croisements avec des données de ChIP-Seq [PWSD+12].

Bonus : Réflexion sur les grammaires et les boules de mots

Le motif LXRα ne se prête pas à une modélisation sous la forme d'une grammaire globale en raison de la trop grande variété des références. Pour cette raison, cette étude s'est majoritairement faite en utilisant les modèles matriciels. Néanmoins, de premiers essais ont été menés pour une recherche par "boules de mots", qui consiste à rechercher individuellement chaque référence jusqu'à 2 substitutions près, et se sont avérés plutôt concluants : sur l'ensemble d'espèces A (Humain, macaque, chimpanzé, souris, rat, poule), les boules de mots couplées au concept de clique d'orthologie a permis de restreindre la sélection à deux gènes : LPCAT3 et BHLHE40, deux gènes déjà connus pour être effectivement régulés par LXRα. Ces premiers résultats semblent prometteurs et il serait intéressant de creuser d'avantage dans cette direction pour comprendre quels points du modèle ou de la méthode peuvent donner du jeu à la sélection.

Bilan

Dans ce chapitre, nous avons mis au point une méthode permettant de diminuer le nombre de faux-positifs obtenus à l'issu de la reconnaissance de motifs d'un facteur de transcription pour prédire plus précisément des candidats solides. Nous avons par ailleurs développé une base de données et un web-service adaptés pour proposer aux biologistes l'exploitation de cette méthode.

Dans le chapitre suivant, nous nous intéresserons à la deuxième application poussée que nous avons étudiée. Cela concerne la métagénomique et la façon dont la recherche d'amorces de PCR avec erreurs, habituellement ignorées par les biologistes, peut influer sur les résultats. Les nouveaux modèles d'amorce ont été écrit en Logol. Nous montrerons que cette recherche permet d'augmenter le nombre de séquences exploitables et que ces nouvelles séquences permettent d'augmenter l'abondance et la quantité des espèces détectées. Nous monterons enfin que ces nouvelles séquences permettent l'identification d'espèces qui ne seraient pas détectées autrement.

CHAPITRE 4

Reconnaissances d'amorces mutées en métagénomique : détection de nouvelles espèces

La question posée dans ce chapitre est de savoir si rechercher des séquences avec des amorces mutées dans des données métagénomiques améliore la détection d'espèces présentes (par rapport à une recherche standard qui ne conserve que les séquences ayant des amorces strictes). La recherche porte sur un fragment de la séquence 18S pour la détection d'espèces eucaryotes unicellulaires.

Nous montrons tout d'abord que la recherche d'amorces mutées permet d'obtenir un plus grand nombre de séquences exploitables. Ensuite, nous montrons que la majorité de ces nouvelles séquences sont similaires à celles qui possèdent des amorces exactes et ne bruitent pas les données. Enfin, nous montrons qu'une partie des nouvelles séquences forment des signatures d'espèces qui n'étaient pas détectées en utilisant seulement les séquences avec des amorces exactes.

Les travaux présentés dans ce chapitre font l'objet d'un article en cours de soumission dans la revue PeerJ : Detection of mutated primers and impact on targeted metagenomics results

Ces travaux ont aussi fait l'objet d'une présentation orale "Detection of mutated primers and impact on targeted metagenomics results" *et ont été publiés dans les actes de JOBIM 2016 [ALMDB16] (Journées Ouvertes en Biologie, Informatique et Mathématiques) et du workshop RCAM 2016 (Recent Computational Advances in Metagenomics).*

Dans ce chapitre, nous nous intéressons à la deuxième application qui a été développée en profondeur pendant la thèse. Elle concerne cette fois des données de métagénomique ciblée, utilisées pour estimer la biodiversité des sols. Nous allons y utiliser un modèle grammatical, en Logol, pour rechercher des amorces mutées.

En effet, dans une analyse métagénomique, lors de l'étape de séquençage des génomes présents, des erreurs peuvent survenir, aboutissant à l'obtention de séquences où les amorces de PCR sont mutées [HHM+07]. Dans les workflows standards en vigueur en métagénomique, les séquences qui possèdent des amorces mutées sont directement rejetées [dVAH+15, LSHM+14,

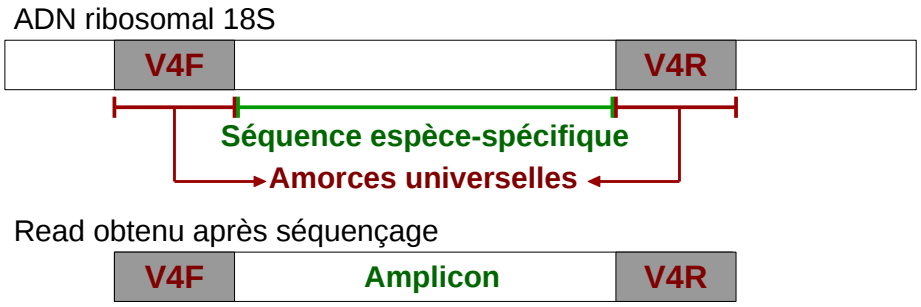


FIGURE 4.1 – Organisation de la région V4 sur l'ADN de la sous-unité ribosomale 18S

TBP+14]. Nous montrons qu'intégrer ces séquences dans les analyses de métagénomique permet d'augmenter l'abondance et la quantité des unités taxonomiques opérationnelles (*Operationnal Taxonomic Unit*, OTU) détectées. Ainsi sont détectés de nouveaux OTUs qui n'étaient pas détectés par les workflows classiques.

Ce travail a été effectué en collaboration avec Frédéric Mahé et Micah Dunthorn, du département d'écologie de l'université de Kaiserslautern, en Allemagne, et s'inscrit dans un projet de métabarcoding [met] visant à faire l'inventaire de la biodiversité des eucaryotes unicellulaires présents dans les sols de différents milieux tropicaux [MMB+15] (Costa Rica, Panama et Equateur). Leur demande était d'accroître le nombre de séquences exploitables dans leurs échantillons en récupérant les séquences ayant des amorces mutées, dans les limites d'un profil de mutations qu'ils avaient établi.

Ce profil de mutations , étant assez fin, nécessitait un outil capable de discriminer les différentes contraintes de mutations tout en autorisant l'absence d'une partie du motif sous certaines conditions. Logol offrant un contrôle complet sur les contraintes du modèle ; il semblait pertinent pour permettre d'exprimer ce modèle.

4.1 Marqueur de biodiversité : la région V4 de la sous-unité 18S de l'ADN ribosomal

L'estimation de la biodiversité d'un échantillon est basée ici sur le gène qui code pour la sous-unité 18S du ribosome, sur la région V4. En effet, cette région possède deux particularités : d'une part, elle contient une portion de séquence qui est extrêmement variable et spécifique à chaque espèce, voire sous-espèce. D'autre part, cette portion de séquence variable est positionnée entre deux segments de séquences extrêmement conservés au cours de l'évolution et présents chez toutes les espèces.

Ces séquences conservées servent donc d'amorces universelles dans le cadre de la métagénomique ciblée : leur relative universalité implique qu'elles peuvent servir d'amorces de PCR pour l'ensemble des espèces eucaryotes unicellulaires présentes dans l'échantillon. La séquence obtenue entre les deux amorces est appelée l'amplicon (cf figure 4.1).

Le gène codant pour la sous-unité 18S du ribosome est long d'environ 1 800 paires de bases, mais le séquençage ne porte que sur la sous-région V4, qui encadre un fragment d'une longueur moyenne d'environ 380 paires de bases. L'amorce universelle forward (V4F) est longue de 20 nucléotides tandis que l'amorce universelle reverse (V4R) est longue de 18 nucléotides.

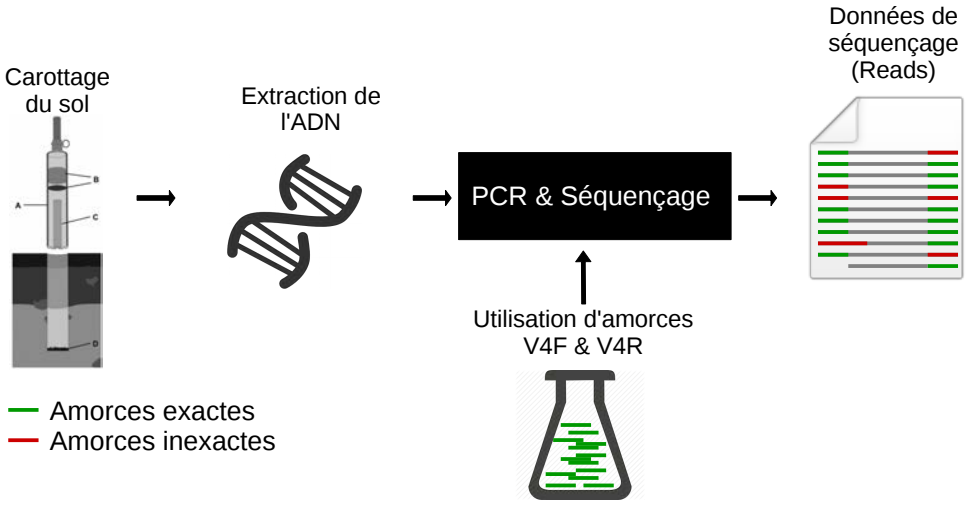


FIGURE 4.2 – Workflow d'analyse métagénomique, première partie : obtention des reads

4.1.1 Workflow standard d'analyse métagénomique : rejet des séquences portant des amorces mutées

Pour estimer la biodiversité d'un échantillon, les biologistes procèdent traditionnellement de la façon suivante (cf figure 4.2 et 4.3) :

- Récupération de l'ADN des espèces eucaryotes unicellulaires présent dans l'échantillon
- Amplification et séquençage de la sous-région 18S de l'ADN par PCR (utilisation des amorces V4 forward et reverse)
- Recherche des amorces dans les reads pour isoler les amplicons. Rejet des reads qui ne contiennent pas les séquences parfaites des amorces V4F et V4R.
- *Clustering* des amplicons sur un critère de proximité (c'est-à-dire regroupement des amplicons ayant des séquences proches).
- Validation des clusters sur un critère quantitatif. Un cluster valide devient un OTU et est considéré comme le marqueur d'une espèce.

Lors de l'étape de séquençage, une fraction des reads obtenus possède des séquences contenant des amorces mutées. Ces séquences sont habituellement rejetées, ce qui conduit potentiellement à une perte d'information sur l'échantillon séquencé. Or, une analyse de la biodiversité cherche à être la plus exhaustive possible dans la détection d'espèces.

4.1.2 Nouveau workflow d'analyse proposé : récupérer les séquences avec des amorces mutées

Les objectifs de notre recherche sont d'identifier les amorces mutées dans les reads issus du séquençage pour augmenter le nombre de séquences analysées et de déterminer dans quelle mesure ces séquences supplémentaires apportent de l'information ou du bruit dans l'analyse de l'estimation des espèces présentes. Il s'agit donc d'ajouter au module "Détection des amorces exactes" (cf figure 4.3) une détection des amorces mutées. Au-delà de cet aspect analytique, notre objectif est aussi de déterminer si le modèle de mutation des amorces proposé par le

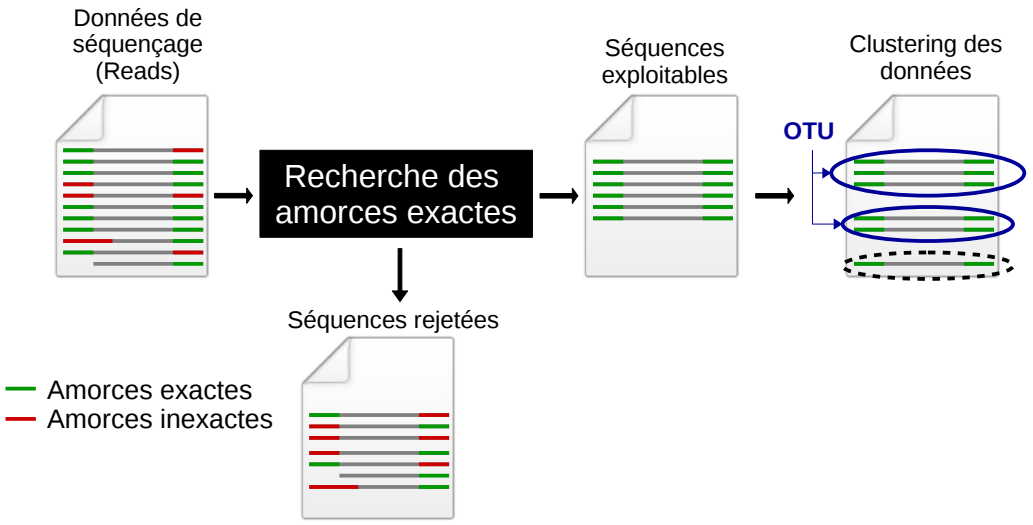


FIGURE 4.3 – Workflow standard d'analysemétagénomique, deuxième partie : obtentions des OTUs

biologiste peut être amélioré pour être plus exhaustif et précis.

4.2 Données disponibles : amorces V4 et reads 454/Roche et Illumina/MiSeq

4.2.1 Amorces universelles et profil de mutation

Les amorces V4F & V4R [SBN+10] : amorce exactes

L'amorce V4F est longue de 20 nucléotides et est définie par l'expression régulière suivante :

```
CCAGCA[CG]C[CT]GCGGTAATTCC
```

L'amorce V4R est longue de 18 nucléotides et est définie par l'expression régulière suivante :

```
T[CT][AG]ATCAAGAACGAAAGT
```

Les nucléotides entre crochets indiquent les différentes alternatives possible à une même position.

Le modèle de mutation : amorces mutées

En vérifiant la nature des reads rejetés au cours de la détection des amorces V4, Frédéric Mahé a pu constater que de nombreux reads possèdent une version légèrement mutée de l'amorce. Il a donc établi un profil de mutation à partir des amorces mutées qu'il a constaté de visu :

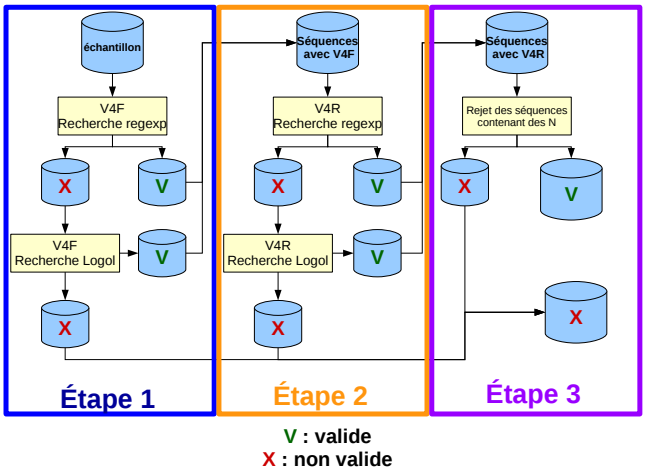


FIGURE 4.4 – Nouveau workflow d'analyse métagénomique couplant une recherche des amorces exactes (par expression régulière) à une recherche des amorces mutées (par Logol)

« 2 mutations ou 1 insertion/délétion possible. S'il y a une insertion/délétion, elle doit être interne (ie, ne pas affecter un nucléotide aux extrémités). L'amorce V4F peut être tronquée d'un ou deux nucléotides terminaux, sans que cela ne modifie les contraintes de mutation autorisées. »

Le modèle de mutation a été traduit en Logol. Les grammaires Logol qui découlent de ces contraintes sont présentées dans la partie 5.2.

4.2.2 Données de séquençage

Les données étudiées sont constituées de 9 échantillons issus de sols tropicaux. Chaque échantillon a été séquencé à la fois avec la technologie 454/Roche et la technologie Illumina/-MiSeq, à des fins de comparaison des 2 technologies.

— Les données 454/Roche rassemblent 310 375 séquences d'une taille moyenne de 397 nucléotides.

— Les données Illumina/MiSeq rassemblent 5 223 138 séquences d'une taille moyenne de 404 nucléotides.

4.3 Recherche de reads contenant des amorces mutées : détection de nouveaux OTUs

Le workflow habituellement employé par les biologistes a été repris en y ajoutant une étape de recherche des amorces mutées avec Logol pour les séquences où la recherche par les expressions régulières de l'amorce exacte ne donne pas de résultat, tel que présenté dans la figure 4.4.

Les reads passent par trois étapes de sélection successives :

— L'étape 1 sélectionne les reads munis de l'amorce V4F (exacte ou mutée). Les amorces identifiées sont supprimées des reads.

— L'étape 2 sélectionne les reads munis de l'amorce V4R (exacte ou mutée). Les amorces identifiées sont supprimées des reads.

	jeu de données 454	**jeu de données Illumina**
Séquences initiales	**310 375**	**5 223 138**
Séquences avec un V4F exact	302 505 (97,46%)	4 592 349 (87,93%)
Séquences avec un V4F muté	7 001 (2,25%)	109 792 (2,1%)
Séquences avec un V4F	**309 506 (99,72%)**	**4 702 141 (90,03%)**
Séquences sans V4F	*869*	*520 997*
Séquences avec un V4R exact	287 360 (92,59%)	4 412 349 (84,48%)
Séquences avec un V4R muté	19 311 (6,22%)	273 185 (5,23%)
Séquences avec un V4R	**306 671 (98,81%)**	**4 685 619 (89,71%)**
Séquences sans V4R	*2 835*	*516 522*
Sequences sans N	**305 693 (98,49%)**	**4 685 575 (89,71%)**
Séquences avec N	*978*	*44*
Séquences avec un couple d'amorces exactes	280 074 (90,24%)	4 317 315 (82,66%)
Séquences avec au moins une amorce mutée	25 619 (8,25%)	368 270 (7,05%)
Séquences rejetées	4 682 (1,51%)	537 563 (10,29%)

FIGURE 4.5 – Rappel dans les jeux de données 454/Roche et Illumina/MiSeq à chaque étape du workflow. Les pourcentages sont calculés en fonction de la quantité de séquences initiales.

— L'étape 3 sélectionne les reads sans nucléotide ambigüe (N). Cette étape est effectuée après la recherche des amorces, afin de conserver les reads qui ne présenteraient des N qu'au niveau des amorces.

Au terme de ce workflow, les reads obtenus forment ce qu'on appellera des **amplicons**.

4.3.1 Résultat 1 : la détection d'amorces mutées augmente le nombre d'amplicons obtenus

Sur les échantillons 454/Roche, la recherche d'amorces mutées dans les données de séquençage permet d'augmenter le rappel de +8,3 % (25 619 amplicons supplémentaires détectés), pour un rappel total de 98,5 %. Sur les échantillons Illumina/MiSeq, la recherche d'amorces mutées dans les données de séquençage permet d'augmenter le rappel de +7,1 % (368 260 amplicons supplémentaires détectés), pour un rappel total de 89,7 %.

4.3.2 Résultat 2 : les nouveaux amplicons obtenus sont majoritairement similaires aux amplicons précédemment obtenus

Bien que la détection des amorces mutées permette d'obtenir de nouvelles séquences, il reste à déterminer si ces séquences ajoutées sont effectivement des séquences biologiques vraisemblables ou s'il s'agit de bruit ajouté aux données. Pour cela, nous avons poursuivi le workflow de traitement de données métagénomiques en nous intéressant au devenir des nouvelles séquences dans le résultat final.

L'étape suivant la recherche des reads est l'étape de *clustering*, qui vise à grouper les amplicons relativement similaires au sein d'un même cluster et à éliminer les séquences isolées. Dans notre cas, nous avons utilisé l'outil de *clustering* SWARM [MRQ+15]. SWARM prend en

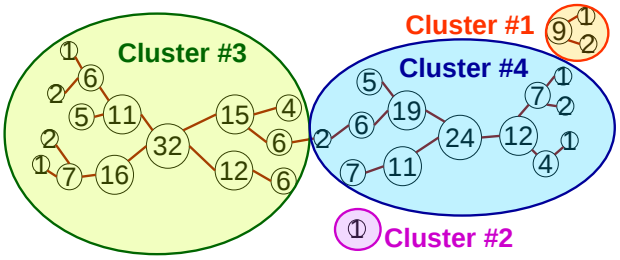


FIGURE 4.6 – Exemple de *clustering* basé sur l'abondance avec SWARM. Bien qu'il existe un chemin entre le noeud d'abondance 32 et le noeud d'abondance 24, l'abondance le long de ce chemin décroît jusqu'au noeud d'abondance 2 avant de recroître : SWARM en déduit la présence de deux clusters différents

entrée les amplicons dérépliqués après suppression des amorces (ainsi, la variabilité des amorces n'influencera pas le *clustering*) et compare chaque amplicon contre chacun des autres. Si deux amplicons sont similaires à un faible nombre de mutations près, SWARM crée un lien entre les deux. SWARM construit donc un graphe où les nœuds sont les amplicons dérépliqués, dotés d'un poids égal au nombre de répliques, et où les arcs sont un lien de similarité. À partir de ce graphe, SWARM définit les clusters à partir des nœuds présentant les plus grandes abondances. Dans le cas où il est possible d'établir un chemin entre deux nœuds de grande abondance, SWARM vérifie s'il existe sur ce chemin un nœud d'abondance minimum (dont l'abondance est inférieure à tous ses voisins). Si c'est le cas, ce nœud est considéré comme formant la frontière entre deux clusters et le chemin est coupé pour obtenir deux clusters indépendants.

SWARM attribue chaque séquence à un cluster, même si ce cluster ne contient qu'une seule séquence faute de voisins similaires. Au final, les clusters trop petits seront éliminés. En effet, un cluster est validé s'il vérifie l'une des deux règles suivantes :

— Contenir au minimum 3 séquences
— Contenir des séquences d'au moins 2 échantillons différents

Les clusters validés forment les OTUs (Operational Taxonomic Units).

Les règles de validation des clusters ont été empiriquement établies à partir du taux d'erreurs des technologies de séquençage. Traditionnellement, les biologistes considèrent qu'en raison de la taille des séquences, il est fortement improbable que les erreurs de séquençage aléatoires puissent aboutir à 3 séquences similaires (Huse et al. [HWMS10] ont montré que ce n'était pas le cas concernant des échantillons 454).

Nous souhaitons comparer les nouvelles séquences obtenues (les amplicons avec au moins une amorce mutée) avec les séquences obtenues de façon standard (i.e. avec les amorces exactes). Cela nous conduit à distinguer 5 types de clusters différents (cf Figure 4.7) :

— **1** Cluster invalide : la quantité de séquences présentes dans le cluster est insuffisante pour permettre sa validation
— **2** Cluster précédemment valide : le cluster ne contient que des séquences avec des amorces exactes. L'ajout des nouvelles séquences n'a pas modifié sa composition
— **3** Cluster mixte précédemment valide : le cluster contient à la fois des séquences avec des amorces exactes et des séquences avec des amorces mutées, mais la quantité de séquences avec amorces exactes est suffisante pour valider le cluster même sans les

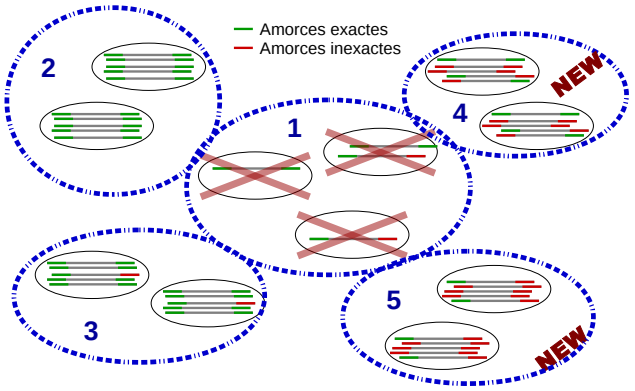


FIGURE 4.7 – Schéma des différents types de clusters possibles.

Technologie	**454/Roche**		**Illumina/MiSeq**	
Type d'amplicons	*Précédents*	*Nouveaux*	*Précédents*	*Nouveaux*
Quantité d'amplicons	*280 074*	*25 619*	*4 317 315*	*368 270*
1 Amplicons rejetés	4,67%	9,01%	19,15%	24,15%
2 Amplicons dans des OTUs précédemment valides	5,38%	∅	2,39%	∅
3 Amplicons dans des OTUs mixtes précédemment valides	89,89%	80,63%	78,46%	70,03%
4 Amplicons dans des OTUs mixtes précédemment non-valides	0,05%	1,31%	0,01%	0,39%
5 Amplicons dans des OTUs précédemment non-existant	∅	9,04%	∅	5,62%

FIGURE 4.8 – Répartition des différents effectifs des jeux de données entre les 5 types de cluster possibles. Les catégories 4 et 5 forment des clusters qui n'étaient pas détectés par le workflow standard des biologistes

nouvelles séquences.

— **4** Cluster mixte précédemment non-valide : le cluster contient à la fois des séquences avec des amorces exactes et des séquences avec des amorces mutées, mais la quantité de séquences avec amorces exactes est insuffisante pour valider le cluster. Sans les nouvelles séquences, le cluster aurait été rejeté : c'est donc un **nouveau cluster**.
— **5** Cluster précédemment non-existant : le cluster ne contient que des séquences avec des amorces mutées. Sans les nouvelles séquences, le cluster n'existerait pas : c'est donc un **nouveau cluster**.

L'analyse de la répartition des nouvelles séquences parmi les OTU aboutit aux résultats suivants (cf Figure 4.8) :

— La majorité des nouvelles séquences rejoint des clusters valides, tous types confondus, c'est-à-dire qu'elles sont dans une OTU nouvelle ou pré-existante (91% en 454/Roche et 76% en Illumina/MiSeq)
— La majorité des nouvelles séquences rejoint des clusters valides pré-existants, c-à-d formés par des séquences avec amorces exactes (80% en 454/Roche et 70% en Illumi-

na/MiSeq)
— Une minorité des nouvelles séquences rejoignent des clusters pré-existants mais qui étaient non valides, augmentant suffisamment l'abondance de ces clusters pour les valider (1,3% en 454/Roche et 0,4% en Illumina/MiSeq)
— Une minorité des nouvelles séquences forment des clusters complètement nouveau, qui ne contiennent donc pas de séquences avec des amorces exactes (9% en 454/Roche et 5,6% en Illumina/MiSeq)

Le fait que la majorité des nouvelles séquences rejoignent des clusters valides pré-existants donne du crédit à la méthode et aux nouveaux amplicons obtenus. En effet, le *clustering* montre que la majorité de ces nouveaux amplicons (82% en 454 er 70% en Illumina) sont similaires à l'existant : la présence d'amorces mutées dans un read n'est donc pas forcément synonyme d'amplicons dégénérés.

On notera qu'en 454, 9% (2 308) des séquences avec amorces mutées sont rejetées (cas 1), tandis que 9% (2 316) se regroupent dans des clusters sans séquences avec amorces exactes (cas 5). En Illumina, 24% (88 938) des séquences avec amorces mutées sont rejetées (cas 1), tandis que 6% (20 697) se regroupent dans des clusters sans séquences avec amorces exactes (cas 5).

4.3.3 Résultat 3 : la détection d'amorces mutées augmente le nombre d'OTU détectés

Les clusters de type 4 et 5 du tableau 4.8 correspondent à des OTUs qui ne sont pas détectés par le workflow standard des biologistes. Sur notre analyse, ces clusters regroupent 10,3% des amplicons en 454/Roche et 6% en Illumina/MiSeq.

En 454/Roche, ces nouveaux OTUs sont au nombre de 304, soit 6,85% des OTUs détectés. En Illumina/MiSeq, ces nouveaux OTUs sont au nombre de 1 190, soit 4,19% des OTUs détectés.

4.3.4 Résultat 4 : les nouveaux OTUs obtenus semblent biologiquement pertinents

La présence de nouveaux clusters valides (donc d'OTUs) soulève encore la question de savoir si l'ajout des amplicons aux amorces mutées améliore la sensibilité de la détection et permet la détection d'espèces peu représentées ou bien s'il introduit du bruit dans les résultats.

Pour y répondre, nous avons essayé d'identifier si ces nouveaux OTUs ressemblaient à des séquences 18S déjà connues dans les banques de données.

L'identification de ces nouvelles espèces s'est notamment faite en utilisant BLAST [AGM$^+$90] pour rechercher les séquences similaires pour chacune des séquences représentatives de ces nouveaux OTUs. Les critères de validation ont été fixés à au moins 90% de couverture et au moins 80% d'homologie.

Au final, 5% des nouveaux OTUs obtenus en 454/Roche et 11% des nouveaux OTUs obtenus en Illumina/MiSeq ont pu être identifiés comme étant proches de séquences 18S connues. En comparaison, l'identification par BLAST des séquences disposant d'amorces exactes s'élève à respectivement 4,6% en 454/Roche et 1,6% en Illumina/MiSeq. Le faible taux d'identification semble donc surtout imputable au manque de connaissances sur les espèces présentes dans les sols tropicaux.

Ainsi, comparés aux banques de séquences 18S connues, ces nouveaux amplicons semblent aussi vraisemblables que les amplicons issus de la détection standard.

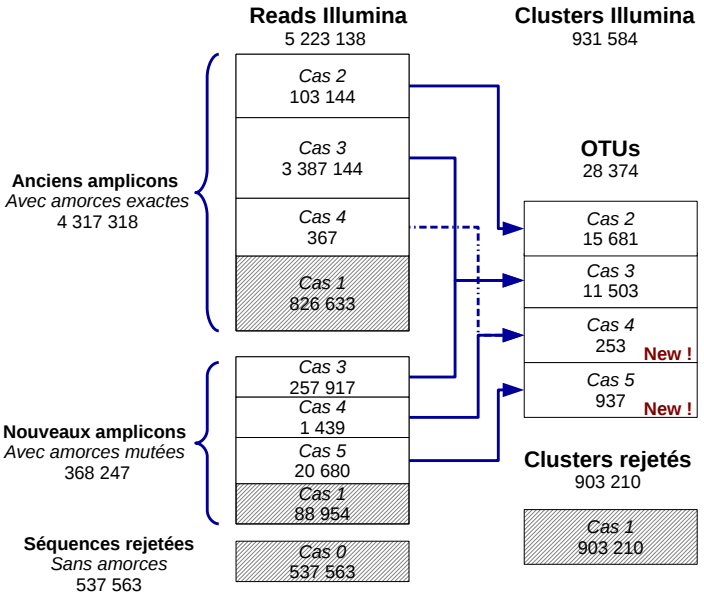


FIGURE 4.9 – Distribution des reads Illumina à l'issu du workflow de recherche des amorces mutées

4.4 Résultat 5 : amélioration du modèle de mutation

En dépit du modèle de mutation proposé par les biologistes (cf partie 4.2.1), il restait encore des séquences pour lesquelles nous ne parvenions pas à détecter les amorces. Nous avons donc essayé de comprendre pourquoi les amorces de ces séquences échappaient à nos filtres.

Pour cela, nous avons entrepris une ultime analyse appelée "la repêche par l'amplicon", dont le workflow est présenté en figure 5.8. Dans cette analyse, les amplicons identifiés avec des amorces (exactes ou mutées) ont été utilisés comme référence pour tenter d'identifier les amplicons non-détectés parmi les séquences rejetées. En connaissant la position de l'amplicon dans les séquences rejetées, il devient alors possible de déduire exactement les bornes des amorces non-détectées et de les isoler. Les amorces non-détectées ainsi obtenues ont été alignées contre leur référence V4F ou V4R, afin d'identifier leur profil de mutation et comprendre pourquoi ces profils n'étaient pas captés par le modèle de mutation actuel.

L'analyse n'a porté que sur un seul échantillon, pour lequel le rappel était de 99% en 454/Roche et 90% en Illumina/MiSeq, pour respectivement 431 et 32 923 séquences rejetées. La détection d'amplicons dans les séquences rejetées nous a permis d'identifier des amplicons dans 133 séquences en 454/Roche et 436 en illumina/MiSeq, que nous appellerons les **amplicons repêchés**.

Cette vérification a permis de se rendre compte d'une réalité qui était passée inaperçue : une partie non-négligeable des amplicons rejetés ne possèdent tout simplement pas d'amorces. Le read débute ou se termine par l'amplicon lui-même, empêchant automatiquement toute détection via la recherche des amorces. Ce cas touche 60% des amplicons repêchées en 454/Roche et 16% en Illumina/MiSeq.

L'alignement entre les références des amorces non-détectées et l'expression régulière des

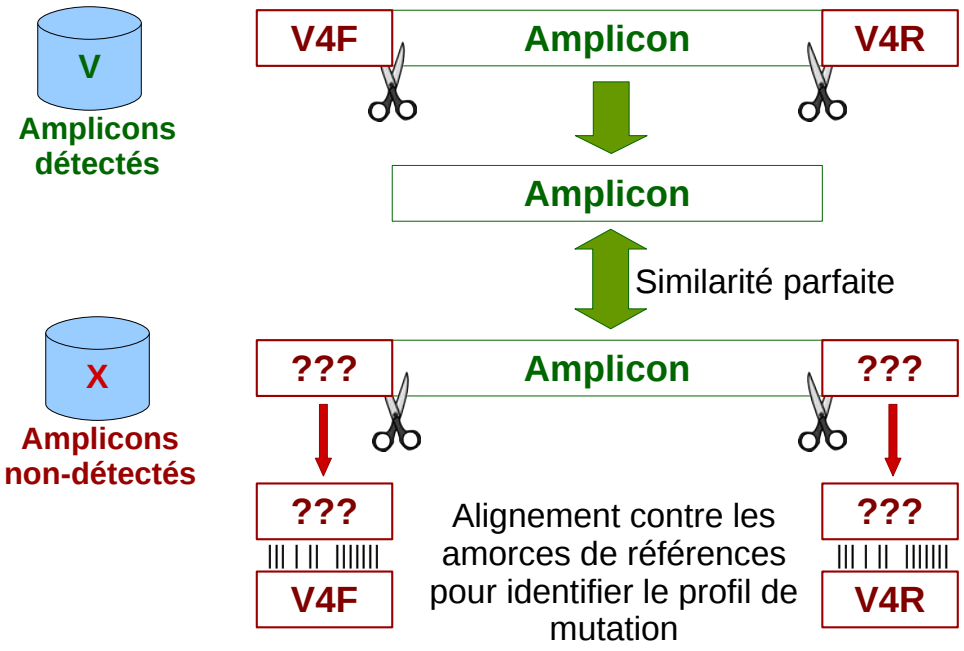


FIGURE 4.10 – Workflow de "la repêche par l'amplicon" : recherche d'amorces mutantes

amorces a été fait en utilisant le logiciel LALIGN [PL88].

La majorité des amorces mutantes possèdent l'un des deux profils de mutation suivant (62% en 454/Roche et 30% en Illumina/MiSeq) :

— Deux délétions
— Une insertion/délétion ET une substitution

Les autres profils de mutations sont des formes très dégénérées de l'amorce dont chaque cas ne concerne que de 1 à 3 séquences. Intégrer plusieurs de ces cas dans le modèle lui ferait perdre sa spécificité or le nombre de séquence concernées est trop faible pour que cela présente un intérêt (cf annexe A.5 pour plus de détails).

Ce résultat a permis d'améliorer le modèle de mutation qui avait été proposé par le biologiste de la façon suivante :

> *« Jusqu'à **2 mutations (substitutions et/ou indels)**. Dans le cas de l'amorce V4R, une absence d'un ou deux nucléotides terminaux est autorisée, sans que cela modifie les contraintes de mutation autorisées. »*

Ce nouveau modèle ne fait plus la distinction entre les différents types de mutations. Il autorise notamment des mutants présentant à la fois un évènement d'indel et un évènement de substitution.

4.5 Conclusion

Dans ce chapitre, nous avons montré que rechercher des amorces mutées dans les données de séquençage métagénomique permettait d'augmenter le nombre d'amplicons détectés, i.e. le nombre de reads retenus au terme du workflow de détection des amorces. Nous avons aussi montré que la majorité de ces nouveaux amplicons étaient similaires aux amplicons avec des amorces exactes : **la présence d'amorces mutées dans les reads n'implique donc pas forcément un amplicon plus muté que la normale** et les inclure dans une analyse semble pertinent. Pour finir, nous avons montré qu'une partie des nouveaux OTUs obtenus par l'intermédiaire des nouveaux amplicons n'étaient pas détectées par le workflow standard qui ne recherche que les amorces exactes : ainsi intégrer les amplicons issus de reads avec des amorces mutées permet la validation de +7% d'OTUs (+304) en 454/Roche et de +4% d'OTUs (+1 190) en Illumina/MiSeq. Par ailleurs, une partie de ces nouveaux OTUs a pu être identifiée comme étant la signature d'espèces connues (5% en 454/Roche, 11% en Illumina/MiSeq) et permet donc de **détecter de nouvelles espèces présentes dans les échantillons**.

Ainsi, la recherche des amorces mutées permet d'exploiter plus complètement un échantillon, ce qui peut être particulièrement utile lorsque les prélèvements ne peuvent être dupliqués. Néanmoins, pour pouvoir conclure de façon plus affirmative, il serait nécessaire de refaire l'étude sur des populations d'espèces mieux connues que celles des sols tropicaux, dans l'esprit des analyses faites par Huse ([HWMS10]) afin de vraiment pouvoir démontrer si les nouveaux OTUs détectés rajoutent majoritairement de l'information ou du bruit.

Cette étude a par ailleurs été l'occasion de préciser le profil de mutation des amorces. Les échantillons de l'étude avaient été analysés à la fois par deux technologies de séquençages (Illumina/MiSeq et 454/Roche). On a pu constater que bien que les deux technologies de séquençages employées soient chacune encline à provoquer un type de mutations de façon préférentielle lors des erreurs de séquençages (la technologie 454/Roche favorise la présence d'indels tandis que la technologie Illumina/MiSeq favorise les substitutions), il n'est pas nécessaire de prendre en compte ces particularités dans le profil de mutation. Dans les faits, le type d'erreur favorisé n'atteint pas une proportion suffisante pour permettre d'occulter les autres types d'erreurs.

Dans cette étude, la **capacité du langage Logol a pouvoir discriminer les paramètres des différents types de mutations** s'est avérée décisive pour permettre de modéliser le profil de mutation constaté par les experts biologistes. En effet, avant cette collaboration, ils n'avaient pas trouvé d'outil satisfaisant autorisant ce contrôle (notamment CutAdapt, l'outil de référence des analyses métagénomiques, n'autorise qu'un paramètre commun pour toutes les mutations).

Dans le chapitre suivant, nous présentons un panorama des six cas de reconnaissance de motifs étudiés faisant chacun appel à différentes subtilités de modélisation. À partir de ce panorama, nous dressons ensuite le bilan des grandes classes de motifs rencontrées pour discuter si les modèles grammaticaux s'avèrent ou non utiles pour exprimer les motifs.

CHAPITRE 5

Exploration de l'expressivité nécessaire dans différents modèles biologiques

La question posée dans ce chapitre est de savoir si l'expressivité des modèles grammaticaux est suffisante pour exprimer la richesse et la variété des motifs biologiques, mais aussi si cette expressivité est finalement réellement nécessaire pour la recherche de tels motifs. Pour cela, nous passons en revue 6 applications, dont les 2 étudiées du chapitre 3 et 4, reprises ici sous leur volet grammatical via Logol.

Il en découle que les grammaires, via Logol, sont capables de modéliser les motifs simples, mais que les approches alternatives sont généralement moins coûteuses. Par ailleurs, les grammaires s'avèrent particulièrement adaptées à la recherche de motifs complexes, qui ne peuvent être modélisés par les approches usuelles des biologistes. Nous dressons ensuite le bilan de ce panorama en positionnant les grammaires par rapport aux méthodes usuelles des biologistes et en pointant les types de motifs où les grammaires apportent une réelle plus-value.

Dans ce chapitre, nous explorons six problèmes biologiques pour lesquels la reconnaissance de motifs grammaticaux peut sembler profitable et nous tentons de les résoudre en Logol. Pour cela, nous sommes allés à la rencontre de biologistes aux prises avec une problématique de reconnaissance de motifs concrète pour laquelle une approche grammaticale semblait pouvoir apporter une solution pratique et efficace. Chacun de ces cas a constitué une occasion de confronter les modèles grammaticaux à de nouveaux types de contraintes à exprimer, mais aussi une occasion de se demander s'il existe d'autres méthodes plus classiques pour résoudre le problème et de les comparer avec le langage Logol. Bien que ces différents cas ne puissent prétendre être un aperçu exhaustif des besoins de la communauté des biologistes en termes de modélisation, nous espérons qu'ils couvrent un champ suffisamment représentatif pour s'en faire une idée.

La dernière partie de ce chapitre rassemble les enseignements tirés de ces différents cas d'étude pour en faire une analyse globale des besoins en termes de modélisation et des réponses qu'y apporte la modélisation grammaticale via Logol.

5.1 DR4 : Modélisation de sites de fixation de type DR4

Problématique

Nous faisons un retour sur l'application développée au chapitre 3 par l'intermédiaire des matrices. Nous montrons ici l'expérience de la modélisation grammaticale via le langage Logol.

Dans cette étude, nous nous intéressons à la modélisation du site de fixation du facteur de transcription (Transcription Factor Binding Site, TBFS) de type "direct répété 4" (Direct Repeat 4, DR4). Le DR4 est un motif répété avec un espace de 4 nucléotides entre les deux répétitions. Dans le cas de LXRα, qui est le facteur de transcription (FT) étudié par nos collaborateurs biologistes, le motif répété est un hexamère [MMYG06], ce qui donne la structure $[Hexamère]NNNN[Hexamère]$. Par ailleurs, dans les sites validés, la répétition n'est pas exacte et, en pratique, le second membre de la répétition diffère sensiblement du premier membre.

L'objectif de la modélisation est de rechercher les séquences approchées de la séquence consensus d'un TFBS de type DR4, tel que celui de LXRα.

Un autre objectif, plus ambitieux et à l'origine du projet, est de détecter les mots de type DR4 présents dans les séquences pour faire de la découverte *de novo* de sites de fixation.

La capacité à modéliser une répétition constitue l'une des plus-values des grammaires SVG par rapport aux grammaires de plus bas niveau. Par ailleurs, Logol possède spécifiquement la capacité de pouvoir modéliser des répétitions approchées. Ce cas d'étude nous a donc semblé particulièrement adapté pour une modélisation par une approche grammaticale.

Ces travaux ont fait partie des développements d'Orthocis présentés dans le chapitre 3. Ils ont été menés en collaboration avec Sandrine Lagarrigue et Frédéric Lecerf de l'équipe PEGASE de l'INRA à Rennes.

Enjeux biologiques

— Modéliser un TFBS de type DR4 (par exemple, LXRα) en vu d'une recherche plein-génome pour identifier des sites de fixation et prédire de nouveaux gènes ciblés.
— Découvrir de nouveaux TFBS de type DR4, sans référence.

Enjeu bioinformatique

— Définir un modèle et un filtrage suffisamment spécifiques pour limiter les faux-positifs.

Données disponibles

Les références sont les 13 séquences de référence de TFBS de LXRα utilisées dans le chapitre 3 (cf. figure 3.1) et qui correspondent à des TFBS de LXRα avérés. Pour rappel, la séquence consensus associée à ces 13 séquences est : $TGACCGnnnnTAACCC$.

Modèle Logol

Le motif du TFBS LXRα a été modélisé sous forme de grammaire Logol (cf figure 5.1).

Le modèle peut être subdivisé en 3 parties : le premier membre, le gap de taille 4 et le second membre.

```
def:{
}
mod1()==>"TGACCG":{$[0,3],_PART1},GAP1:{#[4,4]},?PART1:{$
    [0,3]}
mod1()==*>SEQ1
```

FIGURE 5.1 – Grammaire Logol pour le TFBS LXRα

— Le premier membre est basé sur le consensus $TGACCG$, auquel on autorise jusqu'à 3 substitutions. En effet, parmi les références, le premier membre le plus éloigné du consensus compte deux différences (par exemple $TGAACT$ sur le Cyp7Alpha1 chez la souris). Or, puisqu'on recherche les motifs approchés, il a été décidé de permettre une substitution supplémentaire.
L'instance qui sera identifiée par la grammaire sera stockée dans la variable PART1.
— Le gap est une séquence de taille 4 sans aucune contrainte de contenu.
— Le second membre est l'instance sauvegardée dans PART1, pour laquelle on autorise jusqu'à 3 substitutions. En effet, dans les références, un second membre peut posséder jusqu'à 3 différences par rapport au premier membre (par exemple $CGACCG/TAACCT$ sur le LPCAT3 chez l'humain).

Ainsi, le modèle utilise le mécanisme de sauvegarde de l'instance : la troisième partie du modèle est comparée à la première partie telle qu'elle a été identifiée et non comparée au consensus. Cela permet d'exprimer la structure répétée "MnnnnM'" avec M' correspondant à M avec au plus 3 substitutions.

Résultats biologiques

Le modèle ainsi mis au point n'est pas spécifique de LXRα. En effet, le modèle est capable de reconnaître trop de séquences différentes, souvent très éloignées des séquences de références (par exemple : $GGACGGacgtGGTGGG$). La divergence observée au sein des références n'est finalement pas adaptée à une recherche de variants via des modèles grammaticaux (cf. chapitre 2). En conclusion, un modèle grammatical ne semble pas la meilleure stratégie pour rechercher des TFBS de type DR, surtout quand la répétition est assez déformée : chaque élément de la paire peut-être éloigné du consensus et les deux copies ne sont pas semblables.

Réflexion sur le modèle Logol

Pour rappel, l'intérêt initial de cette problématique était de modéliser des motifs de type DR4. Les DR étant des répétions directes d'un mot, une approche grammaticale avec une grammaire de type SVG semblait tout à fait indiquée pour les modéliser. Or l'approche a échoué sur le facteur LXRα.

Le problème vient du fait qu'en pratique, le motif LXRα n'est pas un DR4 parfait, ce qui complique la modélisation. Ainsi, la répétition elle-même compte en moyenne deux à trois différences entre les deux éléments répétés. Par ailleurs, les références sont trop éloignées et ne permettent pas de définir un consensus solide. Ainsi, le premier membre ne compte que deux positions qui ne peuvent être mutées (le G en position 2 et le C en position 5).

Cette variabilité pose problème, car elle ne peut être modélisée par un modèle grammatical général que de deux façons différentes, toutes deux non-satisfaisantes :

— En exprimant les différents choix, comme pour une expression régulière. On a vu dans le chapitre 2 que cette solution aboutissait à une importante combinatoire de cibles, globalement peu pertinentes.

— En autorisant des substitutions à partir du consensus. C'est cette solution qui a été utilisée ici. Dans le cas du premier membre, cela implique d'autoriser jusqu'à deux substitutions. Or si on souhaite pouvoir identifier des variants proches, il faudrait autoriser une substitution supplémentaire (pour espérer identifier les variants des références elle-même situées à deux substitutions du consensus). On atteint alors 50% de substitutions sur un hexamère, ce qui réduit mécaniquement sa spécificité. Mais puisque le second membre n'est jamais une répétition exacte du premier membre, il faut encore rajouter des substitutions supplémentaires (un minimum de 3 ne serait-ce que pour identifier les références), ce qui aboutit à un motif de taille 16 avec jusqu'à 10 nucléotides potentiellement non-définis (soit 63% de la séquence).

La possibilité, permise par une grammaire, de structurer un motif n'apparaît donc pas comme une approche pertinente pour représenter un TFBS de type LXRα, en dépit de son appellation "DR4".

Ainsi qu'il a été montré dans le chapitre 2, les matrices forment une alternative simple et efficace sur ce type de motif sans indels.

Du point de vue des grammaires, une approche par boule de mots, c'est-à-dire rechercher les variants directs à partir de chaque référence au lieu de rechercher les variants à partir du consensus, devrait probablement permettre de gagner en spécificité. Des tests ont été fait en recherchant les boules de mots constitués autour des références avec jusqu'à 2 substitutions chacune. Appliqué à la recherche du TFBS LXRα, sur le même ensemble d'espèce que le cas d'étude du chapitre 5.1, l'analyse permet de retrouver la référence LPCAT3. Des travaux plus poussés devrait permettre de régler plus efficacement le rayon de ces boules de mots pour obtenir des résultats moins stringent.

5.2 Amorces mutées : Recherche d'amorces mutées dans des données métagénomiques

Problématique

Cette section détaille la modélisation des amorces mutées employées dans le chapitre 4. Elle s'inscrit dans le cadre d'une analyse métagénomique de la biodiversité dans des échantillons séquencés. Des erreurs peuvent survenir durant le séquençage, aboutissant à des reads avec des amorces mutées. Ces reads sont habituellement rejetés par le workflow d'analyse ([HHM$^+$07]). La question biologique était de savoir si la détection d'amorces mutées pouvait permettre d'augmenter le nombre de reads analysables et si cela pouvait permettre d'identifier de nouvelles espèces présentes dans les échantillons.

Notre objectif était de modéliser les amorces selon le modèle de mutation défini par les biologistes. Ce profil demandait un contrôle fin des paramètres du modèles, qu'ils ne réussissaient pas à obtenir avec les outils usuels. Nous avons donc essayé de modéliser leurs contraintes sous la forme d'un modèle grammatical via Logol.

```
def:{
}
mod1()==>("ccagca":{$[0,2]},("c":{$[0,1]}|"g":{$[0,1]}),"c
    ":{$[0,1]},("c":{$[0,1]}|"t":{$[0,1]}),"gcggtaattcc":{$
    [0,2]}):{$[0,2]}
mod2()==>("c","cagca":{$$[0,1]},("c":{$$[0,1]}|"g":{$$
    [0,1]}),"c":{$$[0,1]},("c":{$$[0,1]}|"t":{$$[0,1]}),"
    gcggtaattc":{$$[0,1]},"c"):{$$[0,1]}
mod3()==>(mod1()|mod2())
mod3()==*>SEQ1
```

FIGURE 5.2 – Grammaire Logol pour l'amorce V4F mutée

Ce travail a été effectué en collaboration avec Frédéric Mahé et Micah Dunthorn, du département d'écologie de l'université de Kaiserslautern, en Allemagne, et s'inscrit dans un projet de métabarcoding visant à faire l'inventaire de la biodiversité de sols de différents milieux tropicaux [MMB+15] (Costa Rica, Panama et Equateur).

Données disponibles

Les amorces exactes V4F & V4R

Pour rappel (cf section 4.2.1), les amorces V4F et V4R sont respectivement décrites par les expressions régulières suivantes :

`CCAGCA[CG]C[CT]GCGGTAATTCC` `T[CT][AG]ATCAAGAACGAAAGT`

Amorce V4F mutée

Les contraintes à modéliser étaient les suivantes :

« 2 mutations ou 1 insertion/délétion possible. S'il y a une insertion/délétion, elle doit être interne (ie, ne pas affecter un nucléotide aux extrémités) »

L'amorce V4F mutée a été modélisée avec une grammaire Logol (cf figure 5.2) où :
- mod1 représente la grammaire V4F en autorisant jusqu'à deux substitutions ($[0,2])
- mod2 représente la grammaire V4F en autorisant jusqu'à une insertion/délétion ($$[0,1]) et les nucléotides aux extrémités ne sont pas sujet à la possibilité de mutations.
- mod3 constitue l'axiome et permet la recherche de l'un ou l'autre modèle

Dans le cas du modèle autorisant jusqu'à deux substitutions ("mod1"), chaque élément constitutif de la grammaire a reçu la possibilité de totaliser jusqu'à deux substitutions, à l'exception des positions alternatives de taille 1. Le contrôle total du nombre de substitutions est effectué par une vue (symbolisée par des parenthèses incluant l'ensemble des éléments constitutifs) qui a elle-même la possibilité de totaliser jusqu'à deux substitutions. Ainsi, la somme des substitutions des différents éléments constitutifs dans la vue ne doit pas dépasser

```
def:{
}
mod1()==>("t":{$[0,1]},("c":{$[0,1]}|"t":{$[0,1]}),("a":{$
    [0,1]}|"g":{$[0,1]}),"atcaagaacgaa":{$[0,2]},("agt":{$
    [0,2]}|"ag":{$[0,2]}|"a":{$[0,1]})):{$[0,2]}
mod2()==>("t",("c":{$$[0,1]}|"t":{$$[0,1]}),("a":{$$
    [0,1]}|"g":{$$[0,1]}),"atcaagaacgaa":{$$[0,1]},(("ag":{
    $$[0,1]},"t")|(("a":{$$[0,1]},"g"))|"a")):{$$[0,1]}
mod3()==>(mod1()|mod2())
mod3()==*>SEQ1
```

FIGURE 5.3 – Grammaire Logol pour l'amorce V4R mutée

le nombre de substitutions associées à la vue. Hormis cette restriction, toutes les répartitions des substitutions entres les différents éléments constitutifs sont possibles.

Le même principe se retrouve pour contrôler le nombre d'indels dans le modèle autorisant jusqu'à un indel ("mod2"). Les nucléotides à chaque extrémité du modèle n'ont pas reçu de possibilité d'indels, conformément aux spécificités des contraintes proposées par les biologistes.

Amorce V4R mutée

Les propriétés à modéliser étaient les suivantes :

« 2 mutations ou 1 insertion/délétion possible. S'il y a une insertion/délétion, elle doit être interne (ie, ne pas affecter un nucléotide aux extrémités). Une absence d'un ou deux nucléotides terminaux est autorisée, sans que cela modifie les contraintes de mutation autorisées. »

La dernière contrainte est nouvelle par rapport au modèle de mutation proposé pour V4F.

L'amorce V4R mutée a été modélisée avec une grammaire Logol (cf. figure 5.3) où :

- mod1 représente la grammaire V4R en autorisant jusqu'à deux mutations ($[0,2])
- mod2 représente la grammaire V4R en autorisant jusqu'à une insertion/délétion ($$[0,1]) et les nucléotides aux extrémités ne sont pas sujet à la possibilité de mutation.
- mod3 permet la recherche de l'ensemble des modèles

Le contrôle du nombre de mutations dans chaque modèle repose sur le même principe que pour la grammaire V4F (cf figure 5.2).

La possibilité de tronquer la fin de la séquence a été modélisée par un choix représentant les différentes possibilités de fin : `("AGT"|"AG"|"A")`, auxquelles ont été ensuite associées les possibilités de mutation. Conformément aux contraintes proposées par les biologistes, le dernier nucléotide n'autorise pas d'indels.

Résultats biologiques

Les résultats biologiques sont présentés plus en détail dans la section 4.3, page 79.

Sur les échantillons 454/Roche, la recherche d'amorces mutées dans les données de séquençages permet d'augmenter le rappel de +8,3 % (25 619 amplicons supplémentaires détectés), pour un rappel total de 98,5 %. Sur les échantillons Illumina/MiSeq, la recherche d'amorces mutées dans les données de séquençages permet d'augmenter le rappel de +7,1 % (368 260 amplicons supplémentaires détectés), pour un rappel total de 89,8 %.

L'analyse plus fine du devenir de ces séquences au terme du workflow d'analyse des biologistes a permis de démontrer que les séquences avec des amorces mutées ne sont pas elle-mêmes davantage mutées que les séquences avec des amorces exactes (similarité suffisamment conservée pour intégrer des clusters similaires). Par ailleurs, la détection des amorces mutées permet d'accroître la sensibilité des résultats en identifiant la présence de nouvelles espèces dans les échantillons séquencés (détection de +205 espèces potentielles en 454/Roche et +937 espèces potentielles en Illumina/Seq).

Réflexion sur le modèle Logol

Le langage Logol

Du point de vue du langage, Logol s'est avéré très satisfaisant de par sa capacité à laisser un contrôle total de la grammaire à l'utilisateur. Ceci a notamment permis de contrôler où pouvait ou non s'exprimer les mutations au sein du modèle, ce que ne permettent généralement pas la plupart des méthodes.

L'analyseur Logol

Du point de vue de l'analyseur, Logol a posé de nombreux soucis au niveau du temps de calcul, qui s'avérait tout simplement trop long (environ 2 semaines sur le cluster de la plateforme Genouest pour analyser un seul des 9 échantillons), ce qui semble montrer que l'expressivité permise par le langage n'est pas sans contrepartie.

Afin de revenir à des temps de calcul plus raisonnables, nous avons finalement opté pour une stratégie en deux temps. Pour chaque recherche d'amorces :

— Une recherche des amorces exactes est effectuée à l'aide des expressions régulières,
— Une recherche des amorces mutées est effectuée à l'aide de Logol sur les séquences ne disposant pas des amorces exactes.

Procéder de cette façon a permis de ne déployer les analyses Logol que sur les séquences qui nécessitent vraiment une recherche d'amorces mutées. Cette stratégie a permis de traiter un échantillon Illumina en 36 heures sur le cluster de la plateforme Genouest. Bien que cette durée puisse sembler énorme au regard d'autres outils spécialisés (tel que CutAdapt, qui traite le même échantillon en une dizaine de minutes sur un ordinateur de bureau), elle reste acceptable pour les biologistes : si cela permet d'avoir des matériaux plus fiables pour leurs analyses, alors c'est un surcoût qu'ils sont prêts à accepter.

Les modèles grammaticaux

L'utilisation des grammaires de type SVG s'est avérée pertinente vis-à-vis des contraintes souhaitées par Frédéric Mahé et Micah Dunthorn. En effet, il était nécessaire de construire un modèle dans lequel il fallait :

— Que les substitutions et les indels soient mutuellement exclusifs : la présence de l'une empêchant la présence de l'autre,
— Définir des zones où les mutations n'étaient pas autorisées,
— Autoriser le motif à être potentiellement tronqué.

Il a été possible de modéliser l'ensemble de ces contraintes dans une unique grammaire, là où d'autres méthodes auraient nécessité la combinaison de plusieurs modèles ou l'utilisation de filtres de post-traitement. Les grammaires de type SVG, via Logol, ont donc montré ici une expressivité suffisante pour exprimer les différentes contraintes fines du modèle.

Néanmoins, l'exploration des cas de figure non pris en compte par le modèle de mutation proposé par le biologiste nous a permis d'améliorer le modèle et de le rendre plus facilement transposable pour d'autres outils (par exemple avec CutAdapt).

En conséquence, même si les grammaires SVG présentent l'avantage d'être suffisamment expressives et d'autoriser un contrôle assez fin des modèles pour permettre de modéliser les contraintes souhaitées par les biologistes, il s'avère que cette caractéristique n'était pas indispensable dans ce cas de figure.

5.3 Signal PolyA : Modélisation du signal de polyadénylation des ARNm

Problématique

D'après un certain nombre d'études [GCMS99, vHdOPO00, RLH$^+$91, LBMD05], le signal de clivage de la polyadénylation sur les ARN messagers (ARNm) se présente sous la forme d'un motif composite, formé de plusieurs sous-motifs espacés de distances plus ou moins contraintes. Ce cas d'étude est apparu comme étant l'occasion de tester la capacité des grammaires à structurer les sections d'un motif le long d'une séquence, c'est-à-dire à rechercher un motif composé d'une succession de mots dont on connaît plus ou moins les espacements.

Par ailleurs, les différents motifs formant ce signal sont assez mal connus. C'était l'occasion d'essayer de coupler deux processus qui semblent logiquement complémentaires : la découverte de motif (*pattern discovery*) et la reconnaissance de motif (*pattern matching*). Dans notre cas, l'idée est de progresser selon une démarche itérative : les motifs fréquents découverts sont recherchés dans les séquences pour les observer à l'échelle individuelle et en affiner la définition. Ces différents sous-motifs sont ensuite associés dans un modèle global et la qualité du modèle (en particulier sa spécificité) est alors testée sur les séquences.

Éléments de biologie

Au cours de la maturation d'un ARNm, une queue polyA (constituée d'une succession de nucléotides Adénosine d'une longueur comprise entre 50 et 250) est ajoutée à la molécule à son extrémité 3' [EA60]. La queue polyA possède de multiples fonctions importantes pour l'ARNm (stabilité, transport nucléocytoplasmique, traduction etc.) [DKMR04]. Des travaux effectués en 2000 sur la levure [vHdOPO00, GCMS99] ont permis de mettre en évidence plusieurs mots surreprésentés autour du site de clivage, affinant ainsi la connaissance des signaux de clivage.

Le signal de polyadénylation est connu pour posséder trois signaux distincts : le **Positionning Element** (PE) [LBMD05] sur lequel se fixe le complexe de clivage, l'**Efficiency Element** (EE) [RLH$^+$91] dont la présence accroît la stabilité du complexe, et enfin le site de clivage proprement dit, composé d'un dinucléotide CA, où le complexe coupe la molécule d'ARNm et commence la synthèse de la queue polyA.

L'idée de base de cette étude était d'établir un modèle unique incluant les trois signaux, qui soit suffisamment spécifique pour permettre une détection fiable des sites de polyadénylation lors de l'analyse d'un génome complet.

Enjeux Biologiques

— Reconnaître le site de clivage de la polyadénylation dans des gènes de levure
— Détecter de potentiels sites de clivage au cours d'analyses plein génome de la levure, ou d'autres espèces.

Enjeux Bioinformatiques
— Regrouper dans un modèle des signaux distincts et détachés (PE, EE et site de clivage), incluant des informations de positions relatives
— Lier une approche de *pattern discovery* à l'élaboration d'un motif de *pattern matching* (c'est-à-dire rassembler les résultats de sur-représentation de k-mers, les transposer en un modèle grammatical global et procéder à la validation du modèle).

Données disponibles

Jeu de données positif

Le jeu de donnée positif initial est constitué de 1 352 EST (Expressed Sequence Tag) utilisés par Graber [GCMS99] et van Helden [vHdOPO00] dans leurs études respectives. Ces 1352 séquences sont des fragments d'ADN complémentaires, d'une longueur de 230 nucléotides, qui vont de -150 à +80 (noté [-150 ;+80]) du site de clivage de la polyA polymérase.

L'analyse complète du modèle dans toute sa complexité aurait nécessité plus de temps que nous ne disposions. Nous avons donc décidé de nous concentrer dans un premier temps sur un sous-ensemble de ce jeu de données composé de 606 séquences ayant toute la particularité de posséder effectivement le site de clivage CA sur la zone [-10 ;+10] et de laisser les cas particuliers, sans site de clivage, pour un second temps.

Jeux de données négatifs

Afin d'établir la pertinence des motifs créés, plusieurs jeux de données négatifs ont été utilisés, afin de refléter différents cas de figure :
— **Jeu Shuffle** : le premier jeu de données négatif a été construit par la méthode shuffle, en mélangeant aléatoirement les nucléotides formant une séquence positive, pour chaque séquence. Ce jeu permet d'estimer un motif par rapport au hasard « brut ».
— **Jeu MM6** : le second jeu de données négatif a été construit suivant un modèle de Markov d'ordre 6 (c-à-d que la nature d'un nucléotide à une position donnée dépend de la nature des 5 nucléotides précédents). Ce jeu permet d'estimer un motif par rapport à un hasard qui maintiendrait un enchaînement de nucléotides comparable à celui des séquences d'origines.
— **Jeu Biologique** : le dernier jeu de données négatif a été construit suivant le même esprit que décrit dans l'article *"Statistical analysis of yeast genomic downstream sequences reveals putative polyadenylation signals"* ([vHdOPO00]) en sélectionnant aléatoirement des portions de génome de la levure de taille identique aux séquences positives, dans des régions présumées négatives (typiquement, il s'agit de zones en downstream de gènes). Ce jeu permet d'estimer un motif par rapport à la réalité biologique négative.

Motifs du signal de polyadénylation

À partir des résultats des études précédentes, nous avons distingué les trois motifs différents composant le signal de polyadénylation. Cette connaissance *a priori* nous a aidé à analyser les différents résultats obtenus lors de la découverte de motifs.

Motif PE : AATAAA Ce motif est positionné en [-10 ;-35] du site de clivage [ZSSW86]. D'après les travaux de Sherman et Guo [GS95], tous les variants à une mutation près peuvent fixer le complexe de polyadénylation, exception faite du motif GATAAA.

Motif EE : TATATA/ATATAT Ce motif est positionné en [-60 ;-30] du site de clivage [RLH+91].

Site de clivage : CA

L'ensemble de ces trois motifs s'articulerait de la façon suivante sur la séquence [Pro11, LD14] :

```
[U-rich]-[A-rich]-[U-rich]-[A-rich]-[U-rich]
   EE       PE                CA
```

Modélisation

Stratégie

La modélisation du site de clivage s'est faite en plusieurs aller-retours entre les résultats de la reconnaissance de motifs sur les différents jeux de données et les résultats de différents logiciels de découverte de motifs de la suite RSAT (oligo-analysis, dyad-analysis, position-analysis [vHACV98, vHRCV00, vHdOPO00]). La stratégie était d'analyser les séquences positives non-détectées par le modèle pour essayer d'y distinguer des caractéristiques particulières non encore prises en compte (par exemple, en agrandissant la fourchette de position d'un motif ou en repérant des variantes récurrentes d'un motif qui n'étaient pas prises en compte par le modèle).

Les modèles du PE et du EE ont tout d'abord été mis au point de façon séparée, en cherchant à optimiser le rapport rappel/spécificité en favorisant le rappel. Les deux parties ont ensuite été réunies dans un même modèle, légèrement modifié pour optimiser le rapport rappel/spécificité en favorisant la spécificité.

Modèle "final"

```
def:{
}
mod1(PE1)==>"aataaa":{@[120,135],_PE1}:{$[0,2]}
mod2(PE1)==>!"gataaa":{@[@PE1,@PE1]}
mod3(PE1)==>"ca":{@[16 + @@PE1,27 + @@PE1]}
mod4()==>repeat("atatata":{@[90,120],#[6,6]}:{$$[1,1],$
   [0,1]};[0,10])+ [5,5], HALT1
mod4(). mod1(PE1). mod2(PE1). mod3(PE1)==*>SEQ1
```

FIGURE 5.4 – Grammaire Logol pour le motif du site de polyadénylation

Dans le modèle de la figure 5.4 :

— Logol effectue 4 analyses successives de la séquence d'entrée, chaque analyse supplémentaire étant liée à la réussite de l'étape précédente. L'ordre des analyses est apparent dans la dernière ligne de la grammaire : il s'agit respectivement de "mod4", "mod1", "mod2" puis "mod3".

— "mod4" effectue une recherche du motif EE, décrit sous la forme ATATATA avec nécessairement une délétion (mot de taille 7 pour une contrainte de taille 6 (#[6,6]) avec un évènement d'indel imposé ($$[1,1]). On autorise en plus un évènement facultatif de substitution dans le motif ($[0,1]). Les différents tests ont montré que la quantité de motifs EE au sein de la zone [90,120] était la caractéristique la plus à même de distinguer les séquences positives des séquences négatives. On recherche donc un motif répété 5 fois, avec des répétitions potentiellement chevauchantes mais jamais espacées de plus de 10 nucléotides (`repeat(EE;[0,10])+[5,5]` l'utilisation du point-virgule autorise le chevauchement des répétitions). Enfin, il n'est pas intéressant ici d'obtenir toute la combinatoire possible de ces 5 répétitions : la présence d'une seule possibilité est suffisante pour passer à la suite de l'analyse. Le mot-clef HALT1 impose l'arrêt de la recherche à l'obtention du premier résultat valide.
— "mod1" effectue une recherche du motif PE, décrit sous la forme AATAAA dans lequel on autorise jusqu'à 2 évènements de substitution. Le contenu exactement obtenu est conservé dans la variable PE1, qui sera réutilisée dans les autres analyses.
— "mod2" effectue la vérification que la séquence PE1 est différente de GATAAA, l'unique variant du PE dont on sait qu'il ne peut biologiquement pas fixer le complexe de clivage. Cette vérification se fait en imposant le start (@) du "mod2" exactement au niveau du start de la variable PE1 (@[@PE1,@PE1]) et en imposant un contenu qui soit différent de GATAAA (!"gataaa").
— "mod3" effectue enfin la recherche du site de clivage CA de telle sorte que le début de ce site de clivage soit positionné entre 16 et 27 nucléotides à partir de la fin (@@) de la variable PE1.

Résultats biologiques

Le modèle présenté reconnaît une minorité des séquences positives (25%) et permet d'éliminer la majorité des séquences négatives (95% pour les shuffles, 97% pour les HMM6 et 98% pour les Biologiques), pour une précision d'environ 61% (59% pour les shuffles, 61% pour les HMM6 et 62% pour les Biologiques) et une valeur prédictive positive minimum de 79% (79% pour les shuffles, 89% pour les HMM6 et 93% pour les Biologiques). S'il est suffisamment stringent pour limiter en partie les faux-positifs, ce modèle présente l'inconvénient de ne pas reconnaître un grand nombre de positifs.

Les résultats montrent que les deux motifs EE et PE tendent à être indépendants, la présence de l'un n'influençant pas les chances d'obtenir l'autre au sein d'une même séquence. Par ailleurs, la recherche du PE et du EE, y compris dans les zones élargies, ne donne aucun résultat dans de nombreuses séquences du jeu de données positif. Il faudrait donc poursuivre le travail d'analyse pour tenter d'établir un motif fiable de détection.

Réflexion sur Logol et la pertinence d'utilisation des modèles grammaticaux

L'une des plus-values apportées par l'approche grammaticale et par Logol au cours de la mise au point du modèle sur un petit jeu de données réside dans la stratégie d'essai-erreur qu'il a été possible de mettre en place. En effet, Logol permet de très nombreuses possibilités de modélisation, ce qui permet de changer d'orientation pour le modèle sans forcément avoir besoin de changer d'outil. Par exemple, dans notre cas, il n'était initialement pas prévu de passer d'un modèle EE défini par un mot fixe à un modèle EE défini par une concentration de hits sur plage donnée. Logol nous a permis d'adapter très facilement le modèle en ce sens, là où d'autres outils ou approches auraient nécessité de mettre au point des post-filtres pour

rassembler tous les hits obtenus et vérifier la présence/absence de la quantité de hits à l'endroit voulu.

Cette étude a aussi été l'occasion de constater que paradoxalement, la recherche exhaustive des solutions peut s'avérer contre-productive. Ainsi, l'ambiguïté d'une grammaire basée sur la reconnaissance "d'au moins X motifs présents sur la zone Y" peut conduire à une explosion du nombre de solutions différentes possibles en fonction de la quantité de motifs X présents sur la zone Y. Cela se traduit alors par un temps de recherche plus long et une avalanche de résultats qui n'est pas forcément pertinente. Concrètement, dans notre cas, la recherche de 5 motifs ATATATA peut très vite s'emballer si la zone analysée est riche en motif AT. Ce problème a été à l'origine de la création du mot-clef HALT1 pour les grammaires Logol, qui limite la recherche d'un élément (un "mod" Logol) à sa première instance positive. À noter que l'introduction de ce mot-clef dans la grammaire fait passer l'analyse des 606 séquences de 6 heures à 20 minutes.

Enfin, les grammaires sont particulièrement adaptées pour décrire des motifs composés de mots structurés entre eux, puisque les modèles grammaticaux reposent justement sur une structure des informations entre elles. Elles permettent ainsi de rechercher d'un seul tenant de tels motifs, là où d'autres méthodes classiques recherchent les mots individuellement et nécessitent de croiser les résultats postérieurement.

5.4 MiHsmar1 : Recherche d'éléments transposables dans le génome humain

Problématique biologique

Dans cette étude, nous nous sommes intéressés aux cibles génomiques de la protéine SETMAR. La protéine SETMAR est l'un des 52 néogènes actuellement inventoriés dans le génome humain qui provient d'un transposon d'ADN (un élément génétique mobile capable de transposer à une autre position du génome [PBAB15]). SETMAR résulte de la fusion d'une partie des gènes SET et HSMAR1 [CUBF06]. HSMAR1 possèdent un élément ITR (Inverted Terminal Repeat, répétitions terminales inversées) sur lequel peut se fixer la protéine SETMAR. Cette protéine peut aussi se fixer sur un élément miniature qui dérive de l'ITR d'HSMAR1, appelé MADE1 (ou miHsmar1) long de 80pb [SR96]. Le but de cette étude est d'améliorer l'annotation de l'ITR d'HSMAR1 et de MADE1 dans le génome humain.

Cette recherche a été effectuée en collaboration avec Yves Bigot et Benoît Piégu, de l'UMR INRA-CNRS 7247 de l'équipe plasticité génotypique et expression phénotypique, et Isabelle Stevant, stagiaire de Master 2 à l'IRISA en 2010.

De précédentes études ont permis d'établir le consensus des miHsmar1 ainsi que la structure tige-boucle qu'ils forment [ROL+07]. En 2010, un stage effectué au sein de l'équipe Dyliss avait déjà démontré qu'il était possible de modéliser les miHsmar1 sous forme de grammaire Logol et d'effectuer les analyses plein-génome en un temps raisonnable.

Depuis, Yves Bigot et Benoît Piégu ont poursuivi leur recherche sur ce motif et ont affiné leur modèle. La grammaire Logol a donc été mise à jour et utilisée de nouveau pour analyser le génome humain.

Enjeu biologique

— Identifier de nouveaux sites miHsmar1 dans le génome humain

Enjeu Bioinformatique

— Identifier un motif fortement dégénéré mais avec un contrôle sur la répartition des mutations le long du motif, en vue d'une analyse plein génome.

Données disponibles

Modèle de miHsmar1 de 2010

Le modèle de base des miHsmar1 est assez simple et peut être vu comme la succession directe de 5 éléments structurés décrivant un mot de taille variable (entre 60 et 90 nucléotides de long) de la façon suivante :

— Un premier motif relativement conservé "TATTAGGTT". Faiblement muté, il n'admet qu'un indel et jusqu'à deux substitutions.
— Un second motif plus lourdement dégénéré "GGTGCAAAAGTAATTGCGGTT". Il admet un indel et jusqu'à dix substitutions.
— Un troisième motif quelconque d'une taille comprise entre 0 et 30 nucléotides, qui forme la boucle de la structure tige-boucle.
— Le complément du second motif, formant ainsi la tige de la structure tige-boucle.
— Le complément du premier motif, formant ainsi la tige de la structure tige-boucle.

L'étude de 2010 avait par ailleurs montré que seuls les hits disposant d'au moins 80% d'homologie avec le consensus étaient conservés.

Modèle de miHsmar1 de 2016

La modification à tester par rapport aux données de 2010 réside dans le deuxième élément, ainsi que son complément, où deux nucléotides peuvent être spécifiquement dégénérés sans incidence, aboutissant à "GGTGCAAAAGTAATTG**NN**GTT" [CUBF06].

En se basant sur les résultats de l'étude de 2010, seuls les hits disposant d'au moins 80% d'homologie avec le consensus sont considérés comme valides.

Génome humain

Les travaux de 2010 portaient sur la version Hg19 du génome humain. Cette version a été réutilisée pour comparer les résultats suite au changement de modèles, puis la version Hg38 du génome a été utilisée.

Modèle miHsmar1 2010

Le modèle des miHsmar1 utilisé en 2010 était le suivant :

```
def:{
}
mod1()==>"tattaggtt":{$[0,2],$$[0,1]},"
   ggtgcaaaagtaattgcggtt":{$[0,10],$$[0,1]},SPACER1
   :{#[0,30]}, -"wc" "ggtgcaaaagtaattgcggtt":{$[0,10],$$
   [0,1]},-"wc" "atttaggtt":{$[0,2],$$[0,1]}
mod1()==*>SEQ1
```

FIGURE 5.5 – Modèle miHsmar1 de 2010. Le morphisme -wc indique que la grammaire recherche le complément inverse du mot qui suit.

Il est intéressant de noter que l'utilisation du morphisme "wc" est facultative puisque le contenu qui suit est une constante : il était donc tout autant possible d'écrire directement le résultat de la complémentation.

Il est aussi intéressant de noter que la grammaire ne recherche pas réellement une tige boucle, en ce sens que les deux parties de la tige peuvent être chacune mutée de façon indépendante. Si la présence de la tige-boucle avait eu plus d'importance dans le modèle, il aurait été possible de stocker l'instance exacte de la tige pour rechercher un complément basé sur cette instance exacte, tel que :

```
def:{
}
mod1()==>"tattaggtt":{$[0,2],$$[0,1],_TIGE1},"
   ggtgcaaaagtaattgcggtt":{$[0,10],$$[0,1],_TIGE2},SPACER1
   :{#[0,30]}, -"wc" ?TIGE2:{$[0,2],$$[0,1]},-"wc" ?TIGE1:{
   $[0,1],$$[0,1]}
mod1()==*>SEQ1
```

FIGURE 5.6 – Exemple de modèle miHsmar1 insistant sur la présence tige-boucle en utilisant la capacité de la grammaire à stocker et réutiliser des variables. Le nombre de mutations associées aux parties complémentaires est donné à titre indicatif pour autoriser une complémentarité imparfaite.

Cette variante présente l'avantage d'assurer que les deux parties de chaque mot TIGEx sont relativement complémentaires. Par exemple, le mot définissant la seconde partie de la tige mesure 21 bases, dont 10 peuvent être mutées. Le modèle 2010 permet au deux parties de la tige de combiner indépendamment ces 10 substitutions, ce qui peut remettre en question la présence de la tige. Le modèle alternatif impose un base complémentaire au motif tel qu'il a été muté, avant de permettre si nécessaire quelques mutations pour admettre des tiges imparfaites.

Modèle miHsmar1 2016

Le nouveau modèle miHsmar1 est extrêmement proche du modèle précédent : le second élément de la tige ne diffère que sur les bases numéro 19 et 20, qui deviennent indéfinies et sont modélisé par la lettre N de l'alphabet IUPAC. La partie complémentaire de la tige a aussi été modifiée en conséquence.

```
def:{
}
mod1()==>"tattaggtt":{$[0,2],$$[0,1]},"
   ggtgcaaaagtaattgnngtt":{$[0,10],$$[0,1]},SPACER1
   :{#[0,30]}, -"wc" "ggtgcaaaagtaattgnngtt":{$[0,10],$$
   [0,1]},-"wc" "atttaggtt":{$[0,2],$$[0,1]}
mod1()==*>SEQ1
```

FIGURE 5.7 – Modèle miHsmar1 de 2016

Résultats biologiques

Bien que la modification entre le modèle de 2010 et le modèle de 2016 semble mineure (quatre nucléotides indéterminés dans un motif de taille 80), elle conduit a une perte totale de la spécificité du modèle, qui reconnaît alors de trop nombreuses séquences qui n'ont absolument rien à voir de près ou de loin avec les miHsmar1.

Ainsi, pour le chromosome I d'Hg38, on passe de 66 445 hits avec le modèle 2010 à 302 189 hits avec le modèle 2016 (x4,5). Cette avalanche de résultats sature le pipeline d'analyse sans aboutir à davantage de hits passant le filtre d'homologie (80% du consensus). La version 2016 du modèle a donc été jugée contre-productive et est actuellement abandonnée.

A posteriori, l'échec du modèle 2016 ne parait pas si étonnante. En effet, le motif miHsmar1 adopte une structure en tige-boucle, pour laquelle la boucle n'est pas définie, soit une tige définie de taille $\sim$30 et une boucle indéfinie de taille 0-30. La spécificité du modèle repose donc essentiellement sur l'identification de la tige. Le modèle 2010 n'exploite pas réellement cette information, puisqu'il permet aux éléments de la tige d'être mutées de façon non complémentaire. Néanmoins, lors de l'étude de 2010, cette absence de contrôle était contrebalancée par un post-filtre vérifiant l'homologie par rapport au consensus du modèle.

Dans le cas du modèle de 2016, deux positions ont été fixées comme étant indéfinies. Sur la séquence elle-même, cette modification n'est pas anodine : on passe de 10 substitutions autorisées et réparties entre 21 positions de la séquence à 10 substitutions autorisées et réparties entre 19 positions de la séquence. On obtient donc moins de la moitié des positions conservées par rapport au consensus, d'où une perte de spécificité. Cette perte de spécificité est encore amplifiée par le fait qu'on ne recherche pas réellement les deux brins de la tige : d'une part, en l'absence de contraintes, les positions mutées n'auront de complémentaires sur l'autre brin que par le hasard des mutations, et d'autre part, les positions non-mutées étant en minorité, les chances pour que deux positions complémentaires restent simultanément non-mutées sont faibles. Le modèle 2016 recherche donc une tige-boucle, avec une tige définie de taille $\sim$9 et une boucle de séquence très variable de taille $\sim$42-72.

La version 2010 du modèle a permis de fournir des résultats très intéressants sur Hg38 puisqu'elle a permis d'annoter 2 000 nouveaux miHsmar1, pour un total de 7 832 annotations identifiées avec une forte confiance (plus de 80% d'identité avec le consensus).

Réflexion sur Logol et les modèles grammaticaux

La recherche des miHsmar1 est typiquement l'un des cas de figure qui montre l'intérêt de pouvoir construire des motifs structurés : ici, il était nécessaire de définir des contraintes de mutations différentes en fonction de la région du motif, ce qui n'est pas permis par les outils standards (CutAdapat, Blast...). Dans notre cas, cela permet de s'assurer que les mutations ne peuvent pas se concentrer sur la partie du début de la tige.

Si on compare les résultats obtenus avec les annotations fournies par RepeatMasker [TGC02] (issues des bases de données Repbase [BKK15] et Dfam[HFC$^+$16]) et Logol, on s'aperçoit que RepeatMasker restitue 8 000 annotations connues de miHsmar1 sur Hg38. De son côté, Logol est capable d'identifier 6 000 annotations connues plus 2 000 nouveaux éléments putatifs.

Les annotations supplémentaires obtenues le sont grâce à la plus grande flexibilité du modèle Logol par rapport aux méthodes classiques, ce qui lui permet de détecter plus aisément les variants. Par ailleurs, si Logol n'identifie pas 2 000 annotations connues, c'est à cause des contraintes choisies pour le modèle. En effet, il avait été décidé d'avoir une conservation

relativement forte sur le début de la tige boucle. Or il s'avère que les miHsmar1 manquant sont effectivement abimés sur cette partie. Il serait donc nécessaire d'intégrer cette information dans le modèle pour permettre leur détection.

Pour ce cas d'étude précis, BLAST+ offre une complémentarité intéressante avec Logol. En effet, BLAST+ donne d'excellents résultats pour la recherche du motif miHsmar1 même fortement dégénéré. De la même façon que Logol permet l'identification de nouvelles annotations, BLAST+ aboutit lui aussi à 1000 annotations nouvelles, dont 600 sont communes aux nouvelles annotations Logol.

Chaque stratégie de recherche est donc capable d'identifier des motifs "à la marge" du motif idéal, mais de natures différentes en fonction de la méthode employée. Une recherche exhaustive d'un motif fortement variable semble donc demander de combiner différentes méthodes de recherche.

5.5 REP : Recherche de tige-boucles avec mésappariemment

Problématique

Les séquences REP (Repeated Extragenic Palindromes) ont initialement été découvertes chez les entérobactéries, notamment chez Escherichia coli K12 où elles composent environ 1% du génome [BSP+94]. Ces séquences sont très répandues dans le règne bactérien [TR05].

Les REP sont majoritairement retrouvées sous forme de structures extragéniques fortement répétées appelées BIMEs (Bacterial Interpersed Mosaic Element). Un BIME est composée de deux REP en orientation inversées, reliées par une séquence variable. De nombreuses propriétés sont associées aux REP, bien que les fonctions qui en découlent demeurent toujours mal connues [THSQ+12a, MTHH+12].

- Leur structure en tige-boucle intervient dans la stabilisation des brins d'ARNm, conduisant à une régulation de l'expression génétique. Leur présence en partie 3' des ARNm ralentit significativement la progression des exonucléases Rnase II et PNPase et nécessite le recrutement d'autres enzymes (polymérase, phosphorylase, hélicase) [EMB01].
- La structure palindromique joue un rôle de terminaison lors de la transcription (50 % d'efficacité d'un terminateur rho-indépendant) [KC04]
- Au niveau de l'ADN, ces séquences sont capables de lier plusieurs facteurs protéiques (ADN Gyrase, Polymerase I) [EB97, GPH90]
- Les BIMEs sont des sites contribuant à des évènements de recombinaisons homologues [CWB+99, TP06a, TP06b]

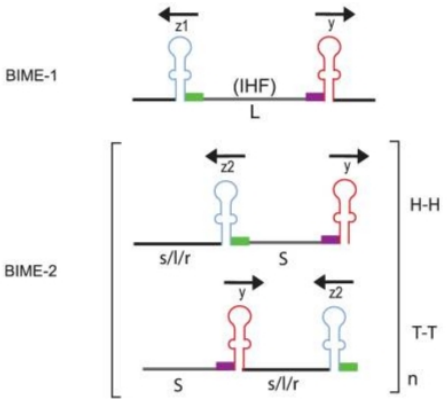


FIGURE 5.8 – Schéma des différents types de BIMEs, encadrés par des structures REP [THSQ$^+$12b]

Il existe différents types de REP (Y, z1, z2) mais dans le cadre de ce projet, seuls les REP de type Y ont été étudiés.

Les REP sont donc des motifs présentant un profil atypique : en effet, leur séquence peut être très fortement dégénérée alors que leur structure en tige-boucle avec un mésappariemment est toujours vérifiée. L'information de structure est ici plus importante que l'information de séquence elle-même.

Les grammaires présentent l'avantage de pouvoir modéliser simultanément des informations de structure et de séquence. Les REP constituent une excellente occasion d'en constater l'intérêt.

Enjeux Biologiques

— Détecter les REP dans un génome bactérien (analyse plein génome).

Enjeu Bioinformatique

— Modéliser une information de contenu liée à une information de structure.

Données disponibles

Bibliothèque de REP

Une bibliothèque recensant les différents REP identifiés manuellement dans le génome d'*E. Coli* a été mise en ligne par Sophie Bachelier-Bassi, sur le site de l'institut Pasteur [1].

Cette bibliothèque contient 58 REP de type Y, qui ont constitué nos cibles.

Modèle des REP Y

D'après la publication de Ton-Hoang et al [THSQ$^+$12a], le modèle d'un REP Y est le suivant :

1. http ://www.pasteur.fr/recherche/unites/pmtg/repet/tableauBIMEcoli.html (Page devenue inaccessible)

```
>Consensus
GTAGGACGGATAAGGCGTTCACGCGCATCCGGCA
>REP-Y
GTAGACCTGGTCAGGCGTTCAC__ __AT______
>REP-Y-mésappariement-réel
GTAGACCTGGTCAGGCGTTCAC____AT______
>REP-Y-repliement-réel
GTAGACCTGGTCAG
nnnnTACACTTGCG
```

FIGURE 5.10 – Exemple de REP Y fortement dégénéré

REP	Type	iREP
GTAGGACGGATAAGGCGTTCACGCCGCATCCGGCA	y	TGCCGGATGCGGCGTGAACGCCTTATCCGTCCTAC
GTAGGCCTGATAAGCGTAGCGCATCAGGCA	z1	TGCCTGATGCGCTACGCTTATCAGGCCTAC
GTAGGCATGATAAGACGCGCCAGCGTCGCATCAGGCA	z2	TGCCTGATGCGACGCTGGCGCGTCTTATCATGCCTAC

iREP : inverterd-REP, des REP en orientation inverse.

GTAG : Tétranucléotide conservé
AA : Nucléotides formant le mésappariemment de la tige
GAT : Nucléotides relativement conservés
ACG : Nucléotides très faiblement conservés
→ : nucléotides appariés dans le cadre de la tige

FIGURE 5.9 – Consensus de référence pour les différents types de REP

Néanmoins, dans les faits, les séquences des REP Y peuvent être très dégénérées. L'exemple de la figure 5.10 montre une séquence REP Y par rapport au consensus, et le décalage du mésappariemment que cette dégénerescence engendre. Bien que la complémentarité des tiges soit imparfaite, la structure reste globalement présente.

Modélisation

Bien qu'un BIME soit constitué de deux REP, nous nous sommes concentrés dans un premier temps sur la modélisation d'un REP unique.

En observant les séquences REP Y, il apparaît que le respect de la séquence importe moins à l'identité du REP que le respect rigoureux de la structure secondaire : une tige-boucle avec mésappariemment.

Le motif du REP Y a été modélisé en grammaire Logol de la façon suivante :

```
def:{
}
mod1(STRUCT2)==>("gtag",X1:{#[5,8],_TIGE1},X2:{#[1,1],
   _MESAP1},X3:{#[1,1],_MESAP2},X4:{#[2,6],_TIGE2},X6
   :{#[2,2]},-"wc" ?TIGE2:{$[0,2]},!+"wc" ?MESAP2,!+"wc" ?
   MESAP1,-"wc" ?TIGE1:{$[0,2]}):{_STRUCT2}
mod2(STRUCT2)==>("gtag","ggataagcgcacgcgcatcggca":{$$
   [0,10],$[0,2]}):{@@[@@STRUCT2,5 + @@STRUCT2]}
mod1(STRUCT2). mod2(STRUCT2)==*>SEQ1
```

FIGURE 5.11 – Grammaire Logol pour la REP Y

Où :

— Logol effectue deux analyses successives de la séquence d'entrée. L'ordre des analyses est apparent dans la dernière ligne de la grammaire : il s'agit respectivement de "mod1" puis "mod2"

— "mod1" modélise la structure en tige-boucle avec mésappariement. La grammaire recherche tout d'abord la présence du tétranucléotide invariant $GTAG$ caractéristique des REP. La grammaire recherche ensuite une tige-boucle telle que la tige soit composée d'une première section de 5 à 8 nucléotides de long, stockée dans la variable TIGE1, suivi de 2 nucléotides non-appariés, puis une seconde section de 2 à 6 nucléotides, stockée dans la variable TIGE2. La boucle qui suit fait 2 nucléotides de long. Puis vient la partie complémentaire de TIGE2, le mésappariement de taille 2 et enfin la partie complémentaire de TIGE1. En raison de la variabilité qui peut exister au niveau des séquences, la complémentarité de TIGE1 et TIGE2 admet un peu de jeu, avec respectivement chacun 2 substitutions autorisées en partie complémentaire. Par ailleurs, afin d'être certain de conserver le mésappariemment dans la structure, les deux nucléotides au sein de la tige en sens direct sont stockés dans des variables (respectivement MESAP1 et MESAP2), et le mésappariement est défini dans la partie complémentaire par "tout sauf le complémentaire de la variable" ($! + "wc"?MESAP$). Enfin, l'ensemble de l'instance vérifiant cette structure secondaire est stocké dans la variable STRUCT2.

— "mod2" compare la séquence avec le consensus du REP. Le motif qu'elle définit est délimité au début par la présence du tétranucléotide invariable, comme mod1, et à la fin par la borne de fin de la variable STRUCT2, définie dans mod1, à plus ou moins cinq nucléotides près. C'est-à-dire que la séquence de mod2 doit couvrir la même région que la structure de mod1 pour aboutir à un motif valide. Le motif en lui-même contient le consensus du REP, dans lequel on autorise jusqu'à 10 délétions et 2 substitutions. En effet, les différentes analyses des REP montrent que si la séquence est souvent conservée, elle peut aussi s'avérer extrêmement dégénérée, y compris concernant les nucléotides dit « conservés ». La dégénérescence est telle qu'il est finalement impossible d'assigner un nucléotide de la séquence à un élément précis de la structure. Un nucléotide théoriquement positionné dans le mésappariement peut finalement être présent dans l'une ou l'autre partie de la tige, par exemple.

Résultats biologiques

Sur les 58 REP testés, 49 sont identifiés par la grammaire. Parmi les 9 REP manquants, 8 ne possèdent pas le tétranucléotide invariant, dont la présence est obligatoire dans notre grammaire. Il faudrait modifier cette obligation pour pouvoir les détecter.

Pour l'heure, ce modèle n'est que théorique. En effet, aucune étape de validation n'a été faite, que ce soit pour estimer la spécificité du modèle vis-à-vis de séquences négatives, mais aussi plus finement par rapport aux autres types de REP (Z1 et Z2), qui possèdent une structure identique et une séquence dégénérée très proche (cf figure 5.9).

Réflexion sur Logol et les modèles grammaticaux

Dans ce cas de figure, l'outil Logol présente un avantage par sa capacité à lier différentes parties de la grammaire par une même variable. Cette particularité permet ainsi de découper proprement la grammaire en deux parties : l'une dédiée à la structure et l'autre dédiée à la séquence. La variable échangée entre les non-terminaux transmet à l'un le fragment de séquence analysé par l'autre. Cette approche est très appréciable puisque le motif REP présente un consensus "déconnecté" de la structure, qui ne permet pas d'associer réellement un nucléotide précis de la séquence à une région particulière de la structure.

Même si d'un point de vue plus général, rien n'empêche de faire des analyses séparées puis d'utiliser un filtre pour faire l'intersection des positions des résultats, cette particularité de Logol a le mérite de pouvoir justement restreindre les analyses successives aux positions d'intérêts (et donc éventuellement de gagner du temps) mais aussi de rendre les détails des deux modèles (la structure recherchée et la séquence acceptée) complètement explicites.

Du point de vue de l'utilisation des grammaires, on constate que dans ce cas précis, elles s'avèrent particulièrement adaptées pour modéliser des motifs possédant des contraintes de natures différentes. De façon générale, les grammaires SVG ont la capacité de pouvoir mélanger aisément des informations de structure et des informations de séquence.

Il est compliqué de mener ce genre de recherche par des approches de reconnaissance de motifs classiques. Notamment à cause de la faible conservation de la séquence [WSTH$^+$13]. Il existe néanmoins un outil spécialisé dans la recherche d'homologues d'ARN conservant davantage leur structure que leur séquence : Infernal [NKE09].

5.6 Introns/Exons : Recherche de tous les ARNm viables lors d'un épissage alternatif

Problématique biologique

Un gène eucaryote contient une succession d'exons, qui contiennent des parties codantes du gène, et d'introns, qui contiennent des parties non-codantes du gène. La transcription d'un gène aboutit à une séquence pré-ARNm contenant les exons et les introns (cd figure 1.2.3). Au cours d'un procédé biologique appelé l'épissage, les introns du pré-ARNm sont supprimés pour ne conserver qu'une succession d'exons formant un cadre de lecture (ORF, Open Reading Frame) viable : c'est-à-dire que la chaîne d'exons forme une succession de triplets de nucléotides, appelés codons, et que cette chaîne commence par un codon start et se termine par un codon stop.

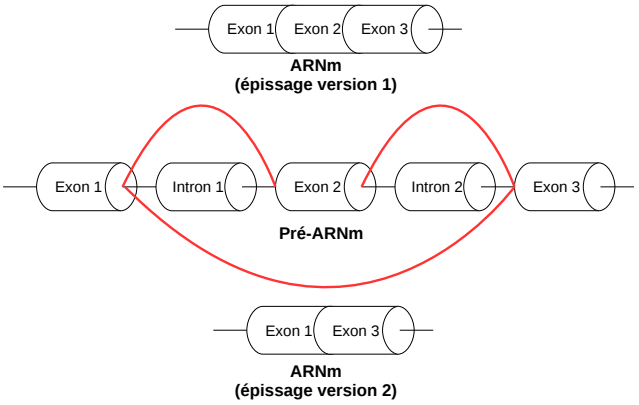


FIGURE 5.12 – Exemple d'épissage alternatif pour un pré-ARNm constitué de 3 exons et 2 introns

Au cours de l'épissage interviennent des possibilités d'épissage alternatif, c-à-d que des exons peuvent être supprimés en même temps que des introns, tout en conservant un cadre de lecture viable. C'est l'une des méthodes qui permet à un organisme de générer des ARNm alternatifs à partir d'un même gène, appelés isoformes.

Les signaux d'épissage sont des petits motifs (dinucléotides) portés par les introns. En les modélisant, il devient possible de prédire l'ensemble des ARNm alternatifs possibles d'un gène donné. Cette prédiction de l'ensemble des ARNm isoformes issus d'un même gènes est l'objet de notre étude.

Ce travail a été fait en collaboration avec Samuel Blanquart de l'équipe BONSAI de l'INRIA. Samuel Blanquart s'intéresse à la compréhension et à la modélisation de l'évolution des séquences moléculaires et travaille notamment sur différentes applications liées à l'inférence d'arrangements de génomes ancestraux [LLB09, BL06]. Il est intéressé par la prédiction des découpages introns/exons dans le cadre de comparaison de l'évolution de gènes orthologues entre différentes espèces.

L'épissage alternatif est la possibilité de construire différents ARNm en fonction de l'épissage des introns. Ce sont les signaux d'épissage qui délimitent les bornes des introns : un donneur d'épissage, le plus souvent "GT", définit le point de départ d'un intron potentiel, tandis que le accepteur d'épissage, le plus souvent "AG", définit la fin d'un intron potentiel. Néanmoins, un donneur d'épissage n'interagit pas exclusivement avec le accepteur d'épissage qui le suit : il lui est possible d'interagir avec des accepteurs d'épissage positionnés plus en aval, ce qui implique alors l'épissage des exons positionnés entre les deux. Le nombre d'ARNm alternatifs possibles pour un gène est donc en théorie égal à l'ensemble des combinaisons possibles des signaux d'épissage qui aboutissent à un ARNm possédant un cadre de lecture valide.

Deux points nous ont paru très intéressants dans ce projet : tout d'abord, l'objectif qui consiste à rechercher tous les résultats possibles obéissant à certaines règles semble une gageure, or il s'agit justement de la notion d'ambiguïté des grammaires. En effet, une grammaire ambiguë est une grammaire qui permet plusieurs façons d'analyser une séquence dans le respect des règles fixées. Dans notre cas, cette ambiguïté nous permettra d'identifier de façon

```
>Artificial_mRNA
ATGAAAAAAGTAAAGCCCCCCCCCCCCCCCCGTCCAGGGGGGGGGGTAA
>Artificial_mRNA_2Stop
ATGAAAAAAGTAAAGCCCCCCCCCCCCCCCCGTCCAGGGGGGGGGGTAAGGTAAG
>Artificial_mRNA_2Stop2Start
ATGAAGATGAAAAAAGTAAAGCCCCCCCCCCCCCCCCGTCCAGGGGGGGGGGTAAGGTAAG
>Artificial_mRNA_2Stop2Start_350nt
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAATGAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAGATG
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAGTAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAGCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCGTCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCAGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGTAAGGGG
GGGGGGGGGGGGGGGGGGGGGGGGGGGGGGTAAGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGG
```

FIGURE 5.13 – Séquences de test artificielles présentant plusieurs possibilités d'épissage différentes

exhaustive les différentes possibilités d'épissage alternatif pour une séquence.

Par ailleurs, l'objectif est d'identifier des ARNm viables, c'est-à-dire commençant par un codon start, suivi d'une succession de codon non-stop et se terminant par un codon stop. Outre l'identification des codons start/stop, cela implique d'obtenir un ARNm formé d'un multiple de trois nucléotides formant les codons. Or, les parties excisées ne sont pas forcément des multiples de 3 et peuvent provoquer un décalage de phase, c'est-à-dire la formation de nouveaux codons dans l'ARNm final, dont il faut vérifier la nature. Il nous a semblé intéressant de voir comment modéliser ce type de contrôle simultanément à la détection des signaux d'épissages.

Enjeu biologique

— Prédiction des découpages introns/exons possibles sur un gène, i.e reconnaissance *in silico* de l'ensemble des ARNm alternatifs possibles

Enjeux bioinformatiques

— Recherche d'un modèle combinatoire non déterministe
— Vérification intrinsèque de l'ORF dans un modèle fragmenté

Données disponibles

Set de séquences artificielles

Afin d'établir et d'étudier le comportement de notre modèle grammatical, nous avons commencé par travailler sur une série de séquences artificielles aux propriétés connues.

Les codons start sont surlignés en orange, les codons stop sont surlignés en vert, les accepteurs et donneurs d'épissages sont indiqués en rouge et les introns les plus courts sont soulignés. Note : Les donneurs et accepteurs d'épissage peuvent se combiner de différentes façons entre eux, qui peuvent produire plusieurs introns différents. Ceux-ci n'ont pas été indiqués sur les séquences par souci de lisibilité.

— La séquence Artificial_mRNA est le cas de figure le plus simple : individuellement, aucun intron ne perturbe le cadre de lecture (chaque intron débute après un codon complet). N'enlever qu'un seul intron conduit à l'obtention d'un ARNm invalide (pas de codon stop sur le cadre de lecture).
— La séquence Artificial_mRNA_2Stop reprend le cas de figure précédent mais ajoute un codon stop décalé par rapport au cadre de lecture standard. Ce codon stop permet de valider les cas où un seul intron est enlevé puisqu'il fournit un codon stop sur le cadre de lecture alternatif obtenu.
— La séquence Artificial_mRNA_2Stop2Start reprend le cas de figure précédent mais ajoute un codon start décalé par rapport au cadre de lecture standard. L'introduction de ce nouveau cadre de lecture double le nombre d'ARNm valides possibles sur cette séquence.
— La séquence Artificial_mRNA_2Stop2Start_350nt reprend le cas de figure précédent mais intercale des codons entre les différents signaux start/stop et donneur/accepteur d'épissage. Cette séquence permet notamment de vérifier si la taille de la séquence peut impacter le temps de calcul.

Gène test : fragment de TRPM8

Pour tester «en condition réelle» notre grammaire, Samuel Blanquart nous a fourni une séquence biologique constituée d'un extrait de 30 000 nucléotides issu du gène TRPM8. Il s'agit du gène TRPM8, tronqué de façon à être plus court et recelant 4 formes d'épissage alternatif valides. Cet extrait est bordé par du génome non-codant en aval et en amont, afin de simuler les conditions d'une véritable analyse plein génome.

Modèle du site d'épissage

Un signal d'épissage est constitué d'un donneur d'épissage, qui marque le début de l'intron, et un accepteur d'épissage, qui marque la fin de l'intron. Le couple donneur/accepteur d'épissage le plus courant est le donneur « GT » et l'accepteur « AG » [Zha98, BBO$^+$78]. Un intron est donc délimité de la façon suivante : [GT ... AG].

Il n'existe pas de taille spécifique aux introns, qui peuvent couvrir une distance de quelques nucléotides à plusieurs milliers [LLB$^+$01].

Concernant les exons, pour qu'ils forment un ARNm valide, ils doivent respecter les trois contraintes suivantes :

— Le premier exon doit commencer par un codon start « ATG »
— Le dernier exon doit terminer par un codon stop « TGA », « TAG » ou « TAA »
— La concaténation de l'ensemble des exons ne doit pas contenir un codon stop (exceptions faites du dernier codon du dernier exon)

De même que pour les introns, les exons n'ont pas spécialement de taille bornée et peuvent faire un ou deux nucléotides de long jusqu'à plusieurs centaines.

Recherche exhaustive des solutions possibles par des grammaires ambigües

Une grammaire ambigüe est une grammaire qui admet plusieurs façons d'analyser un même mot. En l'occurrence, l'une des forces de l'analyseur Logol réside dans sa capacité à rechercher de façon exhaustive toutes les solutions possibles lorsque l'analyse est effectuée avec une grammaire ambigüe. L'analyseur produira donc la liste exhaustive de tous les découpages introns/exons vérifiant les règles d'épissage indiquées dans le modèle Logol.

Grammaire conçue pour faciliter les post-traitements

Puisque la grammaire inclut la recherche simultanée des introns et des exons, cela signifie

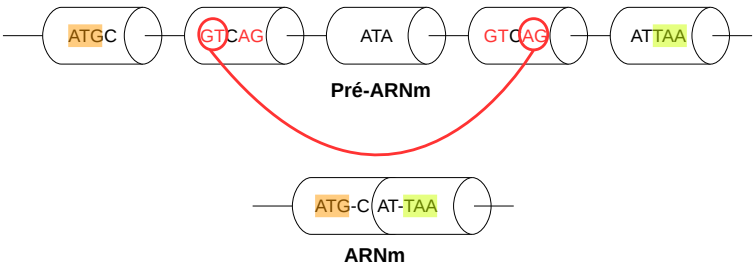


FIGURE 5.14 – Exemple d'épissage alternatif pour un pré-ARNm constitué de 3 exons et 2 introns. Le codon start est surligné en orange, le codon stop est surligné en vert, les accepteurs et donneurs d'épissages sont indiqués en rouge et l'intron épissé est figuré par un lien entre l'accepteur et le donneur employé.

que le résultat fournit par Logol n'est pas directement exploitable tel quel pour obtenir les ARNm valides. Un parsing du fichier XML de résultat est nécessaire pour séparer les introns et reconstruire les chaînes d'exons aboutissant aux ARNm retenus. Ceci implique de construire la grammaire de façon à faciliter au maximum cette étape de parsing. Dans notre cas, cela s'est traduit par une "fragmentation" de la grammaire au travers des différents "mod", afin de les utiliser en tant que mot-clef pour l'étape de post-filtrage.

Le modèle intron/exon

Le modèle Logol défini est le suivant :

```
def:{
morphism(nonstop,a,c)
morphism(nonstop,a,t)
morphism(nonstop,c,a)
morphism(nonstop,c,c)
morphism(nonstop,c,g)
morphism(nonstop,c,t)
morphism(nonstop,g,c)
morphism(nonstop,g,g)
morphism(nonstop,g,t)
morphism(nonstop,t,a)
morphism(nonstop,t,c)
morphism(nonstop,t,g)
morphism(nonstop,t,t)
}
mod1()==>"gt",.*:{#[0,_]},"ag"
mod2()==>!"t",(mod3()|mod4())
mod3()==>mod1(),"nn"
mod4()==>"n",mod1(),"n"
mod5()==>"t",(mod6()|mod7())
mod6()==>mod1(),"n":{_NUC2},+"nonstop" ?NUC2
mod7()==>"n":{_NUC2},mod1(),+"nonstop" ?NUC2
mod8()==>(mod1()|mod2()|mod5())
mod9()==>"atg"
mod10()==>"t",("aa"|"ga"|"ag")
mod11()==>((!"t","nn")|("t","n":{_NUC2},+"nonstop" ?NUC2))
mod12()==>(repeat(mod11(),[0,0])+ [0,_]|X1:{#[0,0]})
mod13()==>mod8(),mod12()
mod14()==>mod9(),mod12(),(repeat(mod13(),[0,0])+ [0,_]|X2
   :{#[0,0]}),mod10()
mod14()==*>SEQ1
```

FIGURE 5.15 – Grammaire Logol recherchant tout les découpages introns/exons aboutissant à un ARNm valide

Avec :

- mod9() : codon start
- mod10() : codon stop
- mod11() : codon non-stop
- mod8() : ensemble des différents cas d'introns
- mod1() : cas d'introns dont le donneur GT est en phase avec l'ORF : ... NNN GTN ...
- mod2() : cas d'introns dont le donneur GT n'est pas en phase avec l'ORF : NNN NGT ... ou NNG TNN ... Le codon chimérique raboutant les deux exons ne peut donner de codon stop car il ne commence pas par « t »

- mod3() : suite du cas d'introns dont le donneur GT n'est pas en phase avec l'ORF : NNN NGT mais ne formant pas un codon stop
- mod4() : suite du cas d'introns dont le donneur GT n'est pas en ohase avec l'ORF : NNG TNN mais ne formant pas un codon stop
- mod5() : cas d'introns dont le donneur GT n'est pas en phase avec l'ORF : NNN NGT ... ou NNG TNN ... Le codon chimérique raboutant les deux exons peut donner un codon stop car il commence par « t »
- mod6() : suite du cas d'introns dont le donneur GT n'est pas en phase avec l'ORF : NNN NGT et vérifiant que le codon chimérique ne forme pas un codon stop.
- mod7() : suite du cas d'introns dont le donneur GT n'est pas en phase avec l'ORF : NNG TNN et vérifiant que le codon chimérique ne forme pas un codon stop.
- mod12() : succession de codons non stop, possiblement absent
- mod13() : recherche d'un intron + une succession de codons non stop
- mod14() : recherche d'un codon start, puis d'une succession de codons non stop, puis d'une recherche d'intron + succession de codons non stop, les deux pouvant être possiblement absents, puis d'un codon stop.

Le pré-ARNm en lui-même est modélisé dans la partie mod14 de la grammaire.

La grammaire commence donc par le mod9, qui définit le codon start, et se termine par le mod10, qui définit la liste des différents codons stop.

Après le codon start vient la grammaire d'un exon, définie dans le mod12. Le mod12 est un choix entre une répétition de taille non-définie de nucléotides non-stop, défini dans le mod11, ou rien (plus exactement, un élément indéfini de taille 0) dans le cas où un intron commencerait directement après le codon start.

Les codons non-stop décrits dans le mod11 se présentent sous deux formes. Dans la première alternative, le premier nucléotide du codon est un codon non-T. Auquel cas, peu importe la nature des deux nucléotides suivants, le codon est forcément un codon non-stop puisque ceux-ci commencent forcément par T. D'où la recherche de deux codons indéfinis "NN". L'autre alternative intervient si le codon commence par un T. La nature du second nucléotide est libre mais est stockée dans la variable NUC2. La nature du troisième nucléotide dépend du contenu de la variable NUC2, à laquelle on applique le morphisme « nonstop ». Un morphisme est une règle de transition qui transforme un nucléotide recherché en un ou plusieurs autres à partir d'une référence donnée. Le complément Watson-Crick est le morphisme le plus utilisé et n'admet qu'une unique transition pour chaque nucléotide (A↔T, C↔G). Dans notre cas, le morphisme « non-stop » autorise de multiples transitions (A→C, A→T, etc.) décrites directement dans la grammaire et formant le produit cartésien des 4 nucléotides, à l'exception des trois transitions (A→A, A→G et G→A) pour lesquelles le triplet aboutirait à un codon stop.

L'avant-dernière partie de la grammaire du mod14 contient la recherche d'alternance d'introns/exons sous la forme d'un repeat du mod13, de façon facultative grâce à une alternative de taille vide (`X2:{#[0,0]}`).

Le mod13 contient donc le modèle de l'intron (mod8) directement suivi de l'exon (réutilisation de la grammaire du mod12).

La recherche de l'intron se définit selon 3 cas de figures, dépendant de leur position au sein du cadre de lecture puisque le cadre de lecture considère les nucléotides par groupe de 3. La position où débute un intron au sein d'un codon implique des stratégies différentes :

Si un intron débute en position 1, il s'agit du cas le plus simple : l'intron commence juste après un codon complet et n'entraînera donc pas de décalage de l'ORF une fois qu'il sera enlevé. Dans ce cas-là, on ne recherche que l'intron lui-même (mod1), défini par son accepteur

et son donneur d'épissage, séparés entre eux par un nombre quelconque de nucléotides.

Si un intron débute en position 2, il entrainera un décalage de l'ORF : l'excision de l'intron va conduire à la création d'un codon qui n'existe pas tel quel dans la séquence et dont il faut vérifier la nature.

Séquences avant l'épissage : `Nuc1-[séquence intronique]-NucX-NucY`
Séquences après l'épissage : `Nuc1-NucX-NucY`

Si Nuc1 n'est pas un "T", alors le nouveau codon formé ne peut pas être un codon stop (cas de figure décrit dans le mod3). Sinon, il faut vérifier que le codon obtenu ne forme pas un codon stop en vérifiant la nature de NucX et NucY en utilisant une fois encore le morphisme (cas de figure décrit dans le mod6). La recherche de l'intron et du codon chimérique obtenu est décrite dans le mod2.

Si un intron débute en position 3 (`Nuc1-Nuc2-[séquence intronique]-NucX`), la problématique est la même qu'en position 2, si ce n'est que la position des nucléotides à considérer pour la création du nouveau codon est légèrement différentes. La vérification de la nature du codon chimérique en fonction de la nature du premier nucléotide du codon est décrite dans les mod4 et mod7 (respectivement si Nuc1 n'est pas "T" ou s'il l'est). La recherche de l'intron et du codon chimérique obtenu est décrite dans le mod3.

Au final, cette grammaire permet de couvrir l'ensemble des cas de figures et donc de détecter l'ensemble des combinaisons possibles d'introns/exons d'une séquence aboutissant à la formation d'un ARNm viable.

Résultats biologiques

Mise au point du modèle sur les séquences artificielles

La grammaire Logol a donné de bons résultats sur les séquences artificielles en permettant de retrouver l'ensemble des combinaisons d'ARNm valides présentes dans les séquences, y compris des solutions «accidentelles» dont nous ne nous étions pas douté lorsque nous avons conçu ces séquences artificielles. L'exhaustivité de l'analyseur s'avère donc très utile pour ce genre de recherche.

L'analyse des séquences artificielles a été l'occasion de constater une dérive du temps de calcul :

Identifiant	Nombre de solutions	Temps
Artificial_mRNA	3 hits	8 secondes
Artificial_mRNA_2Stop	8 hits	11 secondes
Artificial_mRNA_2Stop2Start	16 hits	22 secondes
Artificial_mRNA_2Stop2Start_350nt	16 hits	3min 8s

On constate que le passage à la séquence artificielle de 350nt multiplie par 7 le temps de calcul. Cette augmentation peut s'expliquer par le fait que lorsque l'analyseur rencontre un codon start, il est alors obligé d'analyser (et de stocker) chaque triplet de nucléotides qui suit. Bien évidemment, plus la séquence est longue et plus cela prend du temps. Or, puisqu'il est obligé de recommencer l'analyse autant de fois qu'il y a de solutions potentielles, l'impact de cette analyse augmente de façon exponentielle. Il serait intéressant d'explorer les facteurs

d'augmentation du temps de calcul, en faisant notamment des tests sur des séquences de tailles graduellement croissantes.

Analyse du fragment de TRPM8

Malheureusement, la grammaire n'est pas utilisable en condition réelle, même sur un fragment de gène.

Le premier problème provient de l'analyseur. Comme on pouvait s'en douter en voyant l'évolution du temps de calcul pour les séquences artificielles, l'analyse sur une séquence de taille 30 000 prend un temps "infini". Plus exactement, le processus a été abandonné après une semaine de calcul, alors que l'analyseur n'en était encore qu'aux 500 premières positions de la séquence. L'analyseur n'est donc pas assez puissant pour permettre une recherche aussi ambitieuse sur des séquences de grandes tailles.

Le second problème provient du modèle lui-même. En effet, les 500 premiers nucléotides de TRPM8 sont constitués de gènes non-codants. Or, il s'avère que la grammaire a été capable de trouver une centaine de solutions répondant aux critères proposés par les experts biologistes. En définitive, les contraintes incluses dans le modèle s'avèrent très insuffisantes pour permettre de discriminer efficacement le codant du non-codant. Ceci joue notamment dans l'explosion du temps de calcul constatée car l'analyseur perd énormément de temps sur des éléments qui ne sont pas du tout ce qu'on souhaiterait cibler.

Une deuxième tentative a été essayée en améliorant la grammaire à l'aide de nouvelles données [BPS98, WS14]. Cette amélioration repose principalement sur l'ajout du modèle de la boîte de branchement, qui intervient dans la mécanique de suppression de l'intron. La boîte de branchement est une séquence présente dans l'intron qui intervient dans le mécanisme d'épissage en permettant notamment de replier le pré-ARNm et de "brancher" la partie 5' de l'intron pour initier le raboutage des exons.

Ceci nous a amené à réécrire la grammaire de l'intron (i.e. à redéfinir le non-terminal "mod1") de la façon suivante :

```
mod1()==>"gt",.*:{#[10,_]},("c"|"t"),.*:{#[1,1]},("c"|"t")
    ,("c"|"t"),("a"|"g"),"a",("c"|"t"),.*:{#[0,20]},X1
    :{#[8,12]}:{% "ct":85},"ag"
```

FIGURE 5.17 – Grammaire de l'intron intégrant le motif de la boîte de branchement

Dans le modèle de la figure ci-dessus, la boîte de branchement est positionnée au minimum à plus de 10 nucléotides du signal donneur "GT" et est constituée du consensus suivant [CT]N[CT][CT][AG]A[CT], positionné au maximum à 20 nucléotides du signal accepteur "AG".

Le signal accepteur a lui aussi été modifié : il s'agit maintenant d'une plage de 8 à 12 nucléotides riches en "C" et en "T" (au minimum 85%) se terminant par le dinucléotide "AG".

Le reste de la grammaire est inchangé. Mécaniquement, cela implique une taille minimum d'intron fixée à 30 nucléotides.

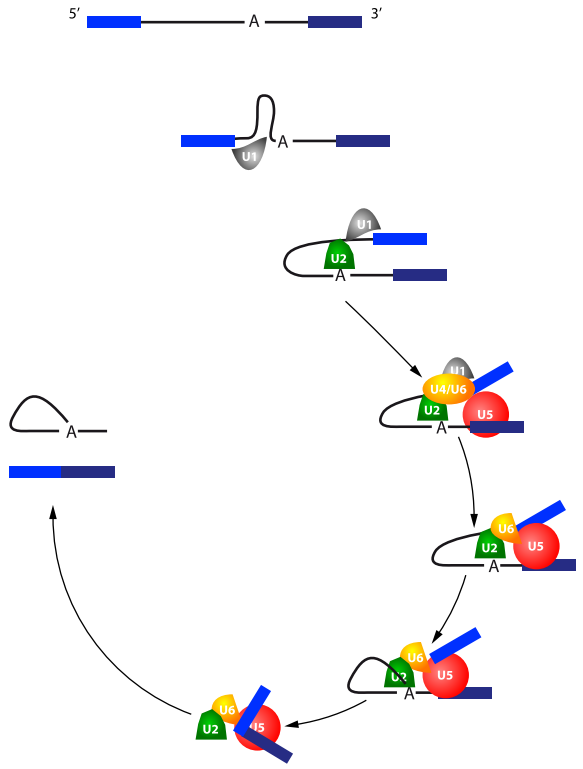


FIGURE 5.16 – Mécanisme d'épissage d'un intron

Malheureusement, cette modification n'a pas été suffisante pour gagner suffisamment de spécificité et la grammaire trouve encore de trop nombreux hits en partie non-codante.

Il est donc nécessaire d'obtenir davantage de contraintes spécifiques aux introns/exons et aux régions codantes pour pouvoir établir une grammaire capable de prédire les introns/exons.

Réflexion sur Logol

Le langage

Le langage Logol possède suffisamment de flexibilité et d'expressivité pour permettre d'établir un modèle non déterministe capable de reconnaître toutes les combinaisons possibles d'introns/exons, sans a priori de quantité ou de taille, et aboutissant à un ORF valide, y compris en prenant en compte les décalages de phase que peuvent engendrer les introns.

Il existe de nombreux outils de recherche d'épissage alternatif, tel que, par exemple, Genesplicer [PLS01] ou Spliceport[DGWM07]. Ces outils sont spécialisés dans la recherche des donneurs/accepteurs d'épissage à partir d'un set d'apprentissage. A partir de références, ils établissent une liste de k-mers surreprésentés dans le cas de références positives et négatives, sur une plage de [-80 ;+80] autour du donneur/accepteur. Ils utilisent ces listes pour établir une probabilité qu'un donneur/accepteur putatif soit positif en fonction des k-mers identifiés entre -80 et +80 de sa position.

Dans le cas de Genesplicer, l'ensemble d'apprentissage est fourni par l'utilisateur, tandis que Spliceport se réfère à un ensemble interne. Dans les deux cas, les matchs sont fournis avec un score, à charge de l'utilisateur de fixer un seuil de validation.

Logol permet une approche très différente, puisque le modèle grammatical effectue la

recherche des ARNm valides, alors que les autres outils ne font que donner des bornes possibles pour les introns, sans chercher à vérifier lesquels permettent d'obtenir des ARNm valides.

L'analyseur

D'un point de vue pratique, l'analyseur supporte difficilement la charge et voit son temps de calcul exploser rapidement en fonction de la séquence et de la quantité de codons et de solutions à traiter. Même si l'absence de spécificité de la grammaire ne l'aide pas, on constate tout de même cet écueil lors de l'analyse des séquences artificielles. L'analyseur ne semble donc pas capable d'effectuer une recherche plein génome (ni même sur de longs gènes), ce qui s'avère tout de même limitant.

D'un autre côté, l'analyseur remplit parfaitement son rôle de recherche exhaustive de toutes les solutions possibles satisfaisant la grammaire. Dans un cas comme celui-ci, où on recherche toutes les possibilités viables, cette particularité constitue un atout très intéressant : tant que la grammaire est correctement définie, aucune solution n'échappera à l'analyse.

En théorie, le découpage intron/exon avec son côté non-déterministe est adapté à une modélisation grammaticale : sous réserve d'avoir les règles générales de découpage, l'ambiguïté de la grammaire permet une identification exhaustive de l'ensemble des cas de figure respectant d'une façon ou d'une autre ces règles. Par ailleurs, l'expressivité de Logol est suffisante pour pouvoir coupler à ces règles de découpages des contraintes de contenu générique tels que la présence de codons non-stop.

Sur le papier, les modèles grammaticaux semblent tout à fait appropriés pour ce genre de recherche, a contrario des méthodes de reconnaissances de motifs classiques, dont les modèles doivent être complètement définis. Dans notre cas, les modèles grammaticaux permettent même d'aller au-delà des outils spécialisés en prenant en charge la vérification de la validité des ARNm alternatifs.

Malheureusement, dans la pratique, le langage se retrouve limité par les performances de l'analyseur, qui n'est pas capable de prendre en charge efficacement ce genre de recherche.

À l'heure actuelle, nous ignorons encore d'où provient ce mur dans la capacité d'analyse de Logol. Bien que des problèmes de lenteurs aient déjà été constatés, c'est la première fois qu'une analyse se retrouve réellement bloquée par les limites du logiciel.

5.7 Bilan des motifs rencontrés

Nous allons établir ici un bilan des différents types de motifs rencontrés dans les 6 applications décrites dans ce chapitre et de la réponse que la modélisation grammaticale via Logol a pu ou non apporter, en termes d'expressivité et d'analyse.

5.7.1 Résumé des différents cas d'études

Pour commencer, nous proposons un résumé des différentes applications abordées. Pour chaque application sont indiqués son nom de code, la thématique, les caractéristiques de l'application, les caractéristiques des modèles Logol utilisés et les résultats obtenus.

1- DR4 LXRα : Recherche des sites de fixation d'un facteur de transcription.
- Caractéristique du motif générique DR4 : structure répétée (RnnnnR)
- Taille : 16 nucléotides
- Modèle Logol : stockage d'une instance de taille 6 dans une variable puis recherche approchée de la variable (cf. page 89).

- — Cibles : 25 800 séquences d'une taille moyenne de 10 000 nucléotides
- — Coût : environ 8h 30min d'analyse sur un nœud du cluster Genouest doté de 16 cœurs.
- — Résultat : la structure répétée se traduit bien en Logol. Cela dit, les sites réels observés ont une répétition très imparfaite (2 à 3 substitutions d'écart entre les 2 hexamères sur les sites avérés du facteur LXRα). Un modèle basé sur la recherche de répétitions doit donc ici autoriser des répétitions très inexactes, et devient trop peu spécifique, cf. chapitre 2, page 49), ce qui génère de trop nombreux hits.
- — Observation : l'analyse a finalement été effectuée par l'intermédiaire d'une matrice (cf chapitre 3).

2- Amorces mutées : Recherche d'amorces mutées dans des données issues de séquençage haut-débit pour une étude métagénomique.

- — Caractéristique des amorces mutées : motifs de taille 18 (V4R) ou 20 (V4F) avec chacun deux positions avec deux alternatives. Évènements de mutation mutuellement exclusifs (respectivement indels et substitutions), motif partiellement tronqué.
- — Taille : 18-20 nucléotides
- — Modèle Logol : modélisation de règles de grammaires chacune dédiée à un type de mutation et distinction de zones sujettes ou non aux mutations dans le motif (cf. page 91).
- — Cibles : 920 500 séquences d'une taille moyenne de 400 nucléotides (jeux de données dérépliqué Illumina avec pré-filtre regex)
- — Coût : environ 20h d'analyse avec une parallélisation massive sur l'ensemble des nœuds du cluster Genouest et un pré-traitement des amorces exactes par les expressions régulières.
- — Résultat : détection de nouveaux amplicons au terme du workflow. Détection de nouveaux OTUs (1190 soit 4.2% des OTUs détectés sur l'échantillon Illumina/-MiSeq) et potentiellement de nouvelles espèces présentes dans les échantillons au terme de l'analyse métagénomique. Mise en évidence du modèle de mutation des amorces.
- — Observation : rechercher au préalable les amorces exactes par une expression régulière standard permet de n'employer la grammaire Logol que sur les reads sans amorces exactes et donc de limiter le nombre de séquences analysées par Logol (réduction du temps d'analyse de 2 semaines à 20 heures).

3- Signal PolyA : Modélisation du signal de polyadénylation des ARNm.

- — Caractéristique du motif polyA : trois signaux agencés entre eux
- — Taille : respectivement 6, 6 et 2 nucléotides, pour une emprise sur la séquence de taille maximum 75 nucléotides.
- — Modèle Logol : positionnement relatif de 3 sous-motifs. Définition d'un sous-motif répété par sa concentration sur une plage donnée. Définition d'un sous-motif par une exception ("tous les variants sauf un") (cf page 96).
- — Cibles : 600 séquences d'une taille moyenne de 230 nucléotides
- — Coût : environ 20min d'analyse sur un cœur d'une machine de bureau.
- — Résultat : le modèle permet de reconnaître 25% des séquences positives tout en rejetant au moins 95% des différents jeux négatifs. La spécificité du modèle s'est faite au détriment du rappel.
- — Observation : mise au point incrémentale du modèle en association avec une stratégie de découverte de motif (avec oligo-analysis [vHACV98], qui recherche des k-mers sur-représentés, position-analysis [vHdOPO00], qui recherche des k-mers sur-représentés à des plages de positions identiques, et dyad-analysis [vHRCV00],

qui recherche des dyad sur-représentés, tous de la suite RSAT).

4- MiHsmar1 : Recherche d'éléments transposables plein génome.

- Caractéristique du motif miHsmar1 : une séquence consensus dégénérée évoquant une structure en tige-boucle, avec des zones plus ou moins conservées.
- Taille : Entre 60 et 90 nucléotides (longueur de tige 30 nucléotides).
- Modèle Logol : Distinction de zones sujettes ou non aux mutations dans le motif (cf page 99).
- Cibles : 23 chromosomes humains Hg38 (3.10^9 nt)
- Coût : environ 24h d'analyse avec une parallélisation massive sur l'ensemble des nœuds du cluster Genouest.
- Résultat :nouvelles annotations du génome vis-à-vis de miHsmar1.
- Observation : contraintes trop permissives sur le modèle 2010, nécessité de post-filtrer sur l'identité par rapport au consensus ($>$80% d'identité : 7 832 annotations conservées). Complémentarité des méthodes de détection (46 hits spécifiques à la recherche de RepeatMasker, 493 avec Blastn, 1460 avec Logol).

5- REP : Recherche de structures en tige-boucle avec mésappariemment

- Caractéristique du motif REP : une séquence consensus alternant des régions plus ou moins conservées formant une structure en tige-boucle présentant systématiquement un mésappariemment dans la tige. La structure est plus conservée que le consensus.
- Taille : 36 nucléotides.
- Modèle Logol : modélisation d'une règle de grammaire dédiée à la structure et d'une règle de grammaire dédiée à la séquence. Passage de paramètre (transfert de variable) entre les règles pour coordonner l'analyse.
- Cibles : 58 séquences REPy d'une taille moyenne d'environ 150 nucléotides.
- Coût : environ 30min d'analyse sur un cœur d'une machine de bureau.
- Résultat : identification de l'ensemble des REPy pour lesquels le tétranucléotide invariant n'est pas muté.
- Observation : mise au point d'un morphisme dédié pour empêcher la complémentarité des nucléotides au niveau du mésappariemment de la tige. Analyse effectuée sans expert et restant à valider vis-à-vis de données négatives.

6- Intron/Exon : Recherche de tous les ARNm viables à partir d'un gène

- Caractéristique du motif Intron/Exon : combinaison de signaux donneurs et accepteurs d'épissage et respect d'une ORF valide
- Taille : Inconnue (pas même de bornes envisageables)
- Modèle Logol : modèle défini à partir des règles d'épissage pour identifier différents découpages viables (ambiguïté du découpage). Vérification de l'ORF (comptage modulo 3) en prenant en compte les décalages de phases induit par les introns.
- Cibles : Quelques séquences tests + 1 séquence longue de 30 000 nucléotides
- Coût : 3min sur les séquences tests, ∞ sur la séquence longue.
- Résultat : le modèle n'est pas suffisamment spécifique et identifie d'innombrables hits en tous points de la séquence, y compris hors de la région codante
- Observation : l'ambiguïté des grammaires permet d'identifier toute les solutions alternatives possibles pour un même segment, soit tous les découpages introns/exons formant un ARNm valides pour un même gène. Le test sur données réelles a mis en évidence le manque de spécificité du modèle proposé. Le comptage modulo 3 sur des segments non-contigü peut se faire directement dans la grammaire (i.e. prendre en compte les interférences des introns dans un cadre de lecture).

Modèles Grammaticaux	caractéristique des séquences	Cluster de calcul	Temps d'analyse
DR4	25 805 seq. ($\sim$ 10 000nt.)	✓	8h 30min
Amorces mutées	90 600 seq. ($\sim$400nt)	✓	48h
Signal PolyA	600 seq. (230nt.)		20min
MiHsmar1	génome humain (3,4.10^9nt.)	✓	24h
REP	58 séq. ($\sim$ 150nt)		30min
Introns/Exons	1 seq. (30 000nt.)	✓	∞
Frameshift-1	1 seq. (4 215 600nt.)	✓	2h

TABLE 5.1 – Temps de calcul en fonction du modèle

5.7.2 Temps de calcul des cas d'étude

La grande expressivité des modèles grammaticaux n'est pas sans contrepartie. Celle-ci se traduit notamment par un temps de calcul nettement plus long que la plupart des outils alternatifs (Blast, CutAdapt, matrices, etc). En se concentrant sur des types de motifs, plus ou moins spécifiques, les autres méthodes peuvent mettre en place des stratégies dédiées pour une recherche optimisée. La souplesse d'un langage comme Logol nécessite de tout rechercher, ce qui se traduit par des performances moindre, même sur des motifs basiques. Ainsi, Logol met presque autant de temps à rechercher un mot strict explicite qu'à rechercher un mot non-défini répété de façon approximative (cf. Annexe A.2), alors que la plupart des outils de reconnaissance de motifs identifierait le mot strict de façon quasi-instantanée.

Actuellement, le temps de calcul est donc une contrainte propre aux modèles grammaticaux Logol, qu'il est nécessaire de prendre en compte avant de mener une analyse.

5.7.3 Besoin en expressivité pour les différents cas d'étude

Pour les différents cas d'études, des modèles de types variés ont été élaborés, faisant intervenir de nombreuses fonctionnalités. Ces modèles peuvent être classés en plusieurs catégories.

Motifs exprimés par des recherches de mots

Dans le cas de motifs exprimés par recherche de mots, une contrainte globale de mutation pour l'ensemble du modèle s'est avérée suffisante dans certains modèles (REP, signal PolyA), mais deux d'entre eux (amorces mutées page 90 et miHsmar1 page 98) ont nécessité la mise en place d'une distinction des contraintes de mutations en fonction de la région du modèle. Ainsi, pour miHsmar1, cette distinction a permis de modéliser la notion de pression de sélection : certaines zones du motif sont plus facilement sujettes à des mutations.

Motifs exprimés par des recherches de structures

Trois motifs contenaient une structure secondaire (cf. DR4 page 90, et REP page 102 et miHsmar1 page 98) mais en pratique, seul le modèle REP a nécessité la description d'une structure.

— Dans le cas des DR4, en théorie, on recherche un motif répété (`RnnnnR`), de façon exacte ou approchée. Dans les faits, l'élément répété tel qu'il apparaît dans les données réelles est trop dégénéré et, dans le cas du LXRα, il est plus pertinent de cibler sur la séquence un consensus du site de fixation que son information de structure.

— Dans le cas des REP, on observe le cas contraire : le consensus est très dégénéré et n'apporte finalement qu'une information mineure, moins importante que le maintien de la structure elle-même. C'est donc la structure en tige-boucle qui a été modélisée. De

plus, il a fallu indiquer dans le modèle la contrainte de dépendance entre les positions : en effet, pour le maintien de la tige, un nucléotide modifié sur un brin implique une modification complémentaire du nucléotide sur le brin opposé, selon le phénomène dit de "mutation concertée".

Motifs exprimés par des agencements de mots

Quelques modèles ont nécessité de structurer entre eux les différents mots qui les composaient : PolyA en page 94 et Intron/Exon en page 106. Il s'agit des modèles les plus complexes, prenant en compte plusieurs signaux à la fois. Pour les motifs polyA, trois signaux (EE, PE et CA) sont à rechercher, dans l'ordre, avec des espacements contraints. Pour les introns/exons, une succession d'introns et d'exons est à rechercher dans la séquence par l'intermédiaire des signaux donneurs et accepteurs d'épissage. La taille de la répétition n'est pas prédéfinie, ni même bornée. Cette recherche a été mise en œuvre par l'opérateur de répétition.

Motifs décrivant différentes solutions sur une même zone

Le modèle d'épissage alternatif décrit dans le motif intron/exon a nécessité l'ambiguïté des solutions autorisées par les grammaires, c'est-à-dire avoir accès au détail de l'ensemble des solutions possibles satisfaisant le modèle sur un même segment.

Le tableau 5.2 récapitule l'ensemble de ces informations.

5.7.4 Bilan sur les approches alternatives en fonction de l'expressivité

L'expressivité qu'ont nécessité les différents cas d'étude a été comparée à certaines approches alternatives aux modèles grammaticaux.

Motifs exprimés par des recherches de mots

Les expressions régulières et donc les grammaires s'avèrent adaptées pour la recherche de motifs basés sur des références assez bien conservées (cf. amorces mutées, chapitre 4). Pour des variants de mots de taille fixe un peu divergents, les matrices constituent une meilleure alternative (cf. motif LXRα "réel" aux chapitres 2 et 3). Enfin, les méthodes d'alignement sont efficaces pour la recherche de mots de taille variable par rapport à un consensus (cf. MiHsmar1 avec Blast+ en partie 5.4 ou la recherche d'amorces mutées avec CutAdapt). Les motifs HMM n'ont pas été testés sur ces exemples, mais constituent une alternative pertinente de plus en plus utilisée.

Motifs exprimés par des recherches de structures

Il est apparu dans plusieurs des cas étudiés qu'une structure annoncée dans le motif n'était en fait pas vraiment conservée. C'est le cas des motifs miHsmar1 et DR4-LXRα où le modèle décrivant une séquence consensus, sans ajout de contrainte de structure, s'est avéré le plus adapté. Quand la séquence du motif est relativement bien connue, il est donc possible de ne finalement pas tenir compte de l'information de structure. Dans ce cas, toutes les méthodes de recherche d'après un consensus sont adaptées à la recherche.

Quand la recherche de structure est incontournable, il faut se tourner vers des outils dédiés, chaque outil ayant son champs d'application particulier (comme les outils de calcul thermodynamique pour la recherche de motifs en tige-boucle dans les séquences, ou les outils d'alignement pour les recherches de répétitions [JK09]).

Motifs exprimés par des agencements de mots

À notre connaissance, il n'existe pas réellement d'alternatives permettant de prendre en

Type de motifs	DR4 LXRα	Amorces mutées	Signal PolyA	MiHsmar1	REP	Introns/Exons
Ambiguïté des solutions						✓
Variant de mot de taille fixe	✓	✓	✓			
Variant de mot de taille variable		✓		✓	✓	✓
Variant de mots avec des contraintes de mutations variables en fonction de la région					✓	
Structure secondaire associée à un consensus (tige-boucle, répétition)	✓			✓	✓	
Structure secondaire sans séquence de référence (tige-boucle, pseudo-noeud)	✓				✓	
Motifs composites (agencements de sous-motifs)			✓			✓

TABLE 5.2 – Types de modèles utilisés dans chaque cas d'étude

compte ce type de contraintes. Néanmoins, l'intersection des résultats d'une reconnaissance de motifs individuels pour vérifier si l'ensemble respecte une contrainte d'agencement précise est une solution de post-traitement simple et facile à mettre en œuvre avec des langages de programmation usuels. Cependant, l'utilisation d'un pipeline d'analyse complique la vision globale du motif et de ses paramètres.

Motifs nécessitant des modèles non-déterministes

Pour certaines applications, on souhaite avoir l'ensemble des instances possibles d'un motif sur une séquence. C'est par exemple le cas des découpages d'un gène en zones introns/exons (cf. page 106), où on souhaite avoir tous les découpages possibles afin de prédire les transcrits alternatifs (les différents ARNm) possibles. C'est aussi le cas lors de l'analyse d'une séquence ARN à la recherche des potentiels sites de frameshift-1 (cf page 22), des emplacements différents des signaux du motif conduisant à la production de protéines différentes.

Cela dit, de nombreux outils ne délivrent pas toutes les solutions. En particulier, dans le cas de la recherche d'alignement, il existe souvent plusieurs solutions (cf figure 5.18). Néanmoins, la plupart de ces méthodes gèrent ces égalités de façon à ne produire qu'un seul et unique alignement possible [Gie00].

L'utilisation d'un certain nombre d'outils ne sera donc pas possible dans un pipeline d'analyse cherchant l'ensemble des instances d'un motif.

```
Alignement A      Alignement B      Alignement C
aaacc--ttaa       aaa--ccttaa       aaac--cttaa
aaa--ggttaa       aaagg--ttaa       aaa-gg-ttaa
```

FIGURE 5.18 – Possibilité d'ambiguïté sur un alignement entre deux séquences aboutissant à un coût de 4

Un panorama d'outils alternatifs à Logol pour chaque type de modèle est présenté en table 5.3.

Type de motifs	Grammaire simple (Regex)	Grammaire avancée (Logol)	Matrice (Pwm)	Alignement (Blast)	Outils dédiés
Ambiguïté des solutions	✓	✓			
Variants de mot de taille fixe	✓	✓	✓	✓	
Variants de mot de taille variable	✓	✓		✓	
Variants de mots avec gradient de mutations		✓		✓	
Structure secondaire avec séquence de référence (tige-boucle)		✓		✓	
Structure secondaire sans séquence de référence (tige-boucle)		✓			✓ (Mfold)
Multiple structure secondaire sans séquence de référence (pseudo-nœud)		✓			✓ (KnotInFrame)
Mots agencés entre eux sur une séquence		✓			✓ (Post-script) [a]

TABLE 5.3 – Type de motifs des cas d'études et méthodes permettant ou non de les prendre en charge. La colonne "Outils dédiés" indique entre parenthèse un exemple d'outil prenant en charge ce type de spécificité.

a. L'intersection des résultats d'une reconnaissance de motifs individuelle de chaque séquence permet de vérifier si l'ensemble respecte une contrainte d'agencement précise.

CHAPITRE 6

Bilan et résultats

Les six applications explorées pendant la thèse ont permis de juger de l'intérêt réel qu'il peut y avoir à utiliser des modèles grammaticaux tels que Logol au service du *pattern matching*. Au passage, ces explorations ont permis de produire des résultats biologiques sur les données étudiées (cf section 6.1) de mettre en oeuvre des procédures pour traiter les données (cf section 6.1) et de tirer des enseignements généraux sur l'approche du pattern matching sur des données réelles (ce section 6.3).

Pour ce qui est de la question principale "peut-il y avoir un intérêt à utiliser des modèles grammaticaux ?", la est réponse est positive. Plus précisément, nous avons identifier 3 propriétés pour lesquelles les modèles grammaticaux sont précieux :

1. pour traduire l'ambiguïté des modèles réels (et proposer ainsi l'ensemble des découpages alternatifs)
2. pour traduire par la négative certaines conditions (et ainsi interdire exclure certaines valeurs spécifiques)
3. pour traduire la finesse de certains modèles (et poser des contraintes différentes sur des zones précises des motifs)

Dans cette thèse, nous avons cherché à évaluer l'utilité des modèles grammaticaux de haut niveau (de type String Variables Grammar (SVG)) en cherchant à résoudre différentes applications de reconnaissance de motifs sur des séquences d'ADN ou d'ARN. Ces applications ont été choisies pour leur diversité et parce qu'elles semblaient pouvoir bénéficier d'une modélisation grammaticale via Logol. Deux d'entre elles, la recherche de sites de fixation sur le génome et la détection d'amorces mutées en métagénomique, ont été développées de façon plus poussée. Pour la recherche de sites de fixation, les matrices de comptage se sont révélées finalement plus adaptées que les modèles grammaticaux, alors que pour la détection d'amorces, Logol a permis de récupérer des séquences qui avaient été écartés.

Quatre applications supplémentaires ont aussi été explorées au travers du prisme de Logol (cf Chapitre 5).

Ce chapitre fait état des résultats de ces explorations. En particulier, les travaux étant menés sur des données réelles avec des spécialistes, ils ont parfois abouti à des résultats biologiques et méthodologiques. Le fait de multiplier les cas d'étude a permis également d'observer certains phénomènes récurrents et de tirer quelques enseignements sur de bonnes pratiques concernant le *pattern matching* sur des données réelles. Pour finir, l'étude a permis d'apporter des éléments

de réponses positifs à la question de départ : les modèles grammaticaux, et notamment Logol, sont-ils réellement utiles pour faire de la reconnaissance de motifs ?

6.1 Résultats biologiques

6.1.1 Prédiction d'un nouveau gène candidat à une régulation directe par le facteur de transcription LXRα

La reconnaissance de motifs appliquée à la recherche des sites de fixation d'un facteur de transcription (TFBS) génère par nature un grand nombre de faux-positifs, le TFBS étant un signal faible, c-à-d peu discriminant. Nous nous sommes basés sur l'hypothèse de conservation des TFBS au cours de l'évolution, que nous avons traduite sous la forme d'un filtre d'orthologie (un hit est potentiellement valide s'il est retrouvé au sein d'une clique d'orthologie formée par un groupe d'espèces, c-à-d si le site est également présent sur le même gène dans les espèces voisines) et d'un filtre d'homogénéité (le gène d'une clique d'orthologie est d'autant plus susceptible d'être régulé par le facteur de transcription que ses sites de fixation se ressemblent). Cette approche nous a permis d'identifier un nouveau gène, STOML2, différentiellement exprimé entre des souris sauvages et des souris dont le facteur LXRα a été neutralisé, et qui possède un TFBS de LXRα conservé chez l'homme, le macaque, le chimpanzé, la souris, le rat et la poule. Ce gène fait actuellement l'objet d'une validation expérimentale.

6.1.2 Détection de nouvelles espèces présentes dans des échantillons de sols tropicaux

En métagénomique, les reads issus de séquençage possédant des amorces mutées sont habituellement automatiquement rejetés et les analyses ne portent que sur les reads avec des amorces exactes. La recherche d'amorces mutées dans les reads de séquençage de données tropicales nous a permis d'augmenter le nombre de séquences exploitables (amplicons). L'étape de regroupement des amplicons similaires (*clustering*) a abouti à la formation de groupes d'amplicons, les OTUs, dont la validité repose sur la quantité. Nous avons montré qu'intégrer les amplicons issus de reads avec des amorces mutées permettait la validation de nouveaux OTUs ($+7\%$ ($+304$ OTUs) en 454/Roche, $+4\%$ ($+1\,190$ OTUs) en Illumina/MiSeq). Par ailleurs, en dépit du manque de connaissance sur les populations présentes dans ce type d'échantillon, une partie de ces nouveaux OTUs a pu être identifiée comme étant la signature d'espèces connues (5% en 454/Roche, 11% en Illumina/MiSeq), qui n'avaient pas été détectées lorsque les reads avec des amorces mutées étaient rejetées.

6.1.3 Amélioration du modèle de mutation des amorces V4 utilisées en métagénomique

Dans le cadre de la recherche d'amorces mutées dans les reads de séquençage en métagénomique, l'expert biologiste a proposé un modèle de mutation pour les amorces V4. Ce modèle indiquait une exclusivité des types de mutations présentes dans les amorces mutées valides : soit elles contiennent des substitutions, soit elles contiennent des indels. Afin d'améliorer ce modèle, nous avons analysé les reads rejetés pour lesquels au moins une amorce, exacte ou mutée, n'avait pu être identifiée. Nous avons recherché des amplicons déjà connus au sein de ces reads pour déduire les séquences des amorces non-détectées par les précédents modèles. L'analyse de ces amorces non-détectées a permis de proposer un modèle de mutation plus fin, autorisant le mélange de différents types de mutations (une amorce mutée peut contenir à la fois une substitution et une délétion).

6.1.4 Augmentation de l'annotation des éléments miHsmar1 dans le génome Humain

La protéine SETMAR reconnaît les éléments miniatures qui dérivent d'HSMAR1, les miHsmar1. Ces éléments sont modélisés en Logol sous la forme d'une structure fortement dégénérée, mais dont la dégenérescence est plus faible aux extrémités du motif. La recherche du motif miHsmar1 s'est faite par une recherche de la séquence dégénérée, avec le modèle Logol, puis une sélection des hits passant un filtre de 80% d'homologie avec le consensus. Ce modèle a permis de retrouver 5 796 annotations déjà connues (81%) et d'annoter 2 036 nouvelles positions du génome avec une forte confiance (+29% de nouvelles annotations).

6.2 Résultats méthodologiques

6.2.1 Création de la base de données et du serveur web Orthocis

Afin de faciliter la prédiction d'un nouveau gène candidat à une régulation directe par un facteur de transcription à l'aide de nos filtres, nous avons conçu la base de données Orthocis.

Elle contient les séquences upstreams (10 000 nucélotides en amont du gène) des gènes orthologues de 65 espèces eucaryotes de la base Ensembl, plus des informations d'orthologie des gènes (relations one-to-one issues d'Ensembl COMPARA) et les résultats non-filtrés d'analyse de ces séquences par 1 050 motifs TFBS de la base JASPAR.

Différents filtrages sont alors disponibles (en ligne de commande ou via l'interface Web) pour naviguer dans l'ensemble des hits TFBS à la recherche de gènes ciblés par un facteur de transcription. L'interface web permet de requêter la base de façon dynamique et simplifiée, en permettant d'ajuster librement les différents filtres (seuil de validation pour la valeur-P des hits, ensemble d'espèces considérées pour la clique d'orthologie, seuil de validité pour l'homogénéité des hits d'une clique d'orthologie).

6.2.2 Comparaison de cibles entre les expressions régulières approchées, les matrices et les boules de mots

Nous avons mis en évidence les différences de cibles entre trois types de modélisation du concept des "séquences au voisinage de références connues, à 2 substitutions près", au travers d'un exemple, les sites de fixation du facteur de transcription LXRα, ayant un motif DR4 de taille 16.

Il apparaît qu'avec des références légèrement divergentes (entropie de 20,96 / 32), le nombre de séquences reconnues par une expression régulière approchée devient très vite explosif (17 491 000 cibles reconnues par une expression régulière autorisant 2 substitutions pour 18 500 (0,1%) cibles d'intérêts) et qu'il suffit d'autoriser 1 substitution pour reconnaître la majorité des cibles (14 862 (80%) cibles d'intérêt reconnues sur 1 597 440 séquences reconnues (précision 0,9%)).

Les matrices sont plus pertinentes pour établir un modèle opérationnel à partir de références légèrement divergentes, car elles permettent de cibler efficacement les séquences les plus proches du consensus, donc de filtrer les faux-positifs. Elles autorisent suffisamment de variabilité pour reconnaître des séquences situées au-delà de 2 substitutions des références mais à condition qu'elles restent proches du consensus (ce qui est le cas quand les mutations interviennent sur les positions les plus floues du consensus).

Les boules de mots constituent les modèles les plus restrictifs, qui ne reconnaissent que les voisins directs de chaque référence et n'autorisent rien au-delà.

6.3 Retour d'expérience

Dans les différents cas d'étude que nous avons traités, nous avons pu constater plusieurs "évidences" dont nous n'avions pas pleinement conscience avant de commencer cette thèse, tel que le potentiel du croisement d'informations ou la nécessité de remise en question des motifs. Ces enseignements nous semblent aussi faire partie des résultats de cette thèse.

6.3.1 Au-delà de la reconnaissance de motifs : croisement d'informations

Dans certains cas, raffiner un modèle ne suffit pas pour discriminer efficacement de vrais positifs de faux positifs, un même hit pouvant s'avérer positif ou négatif en fonction de caractéristiques qui ne peuvent être contenues dans un modèle.

C'est par exemple le cas pour la recherche de sites de fixation : parmi les très bons hits retrouvés dans les séquences (i.e. pouvant même avoir la même séquence qu'une référence) certains peuvent malgré tout être des faux positifs.

L'une des façons de lever cette ambiguïté des hits consiste à rajouter de l'information complémentaire. Ainsi, des hits au contenu identique seront ou non retenus au regard d'informations complémentaires. Il s'agit donc de faire du croisement de données avec des sources différentes.

Dans le cadre d'Orthocis (cf chapitre 3), le croisement des résultats de reconnaissance de motifs avec des informations dépendant de la conservation au travers de l'évolution (homogénéité des hits sur des gènes orthologues) s'est avéré un bon moyen pour filtrer les hits les plus vraisemblables. D'autres types de données peuvent être utilisés tels que les pics issus d'analyses CHiP-seq, indiquant une fixation réelle de la protéine, des informations sur l'accessibilité de la chromatine ou encore les termes GO (Gene Ontology).

Dans le cas des amorces mutées (cf chapitre 4), le croisement des sources a également permis de donner du crédit à des hits isolés. En effet, d'une part, nous avons pris en compte des amplicons issus de deux types de séquençage différents, tirant profit du fait que la probabilité qu'un artefact obtenu lors d'un séquençage soit retrouvé à l'identique lors d'un autre séquençage est extrêmement faible ; d'autre part, nous avons travaillé sur des données issues de différents échantillons, tirant profit du fait qu'un élément isolé obtenu en deux endroits différents est probablement la signature d'une espèce présente.

Globalement, cette observation milite en faveur de l'utilisation de réplicats (dans le cas de la métagénomique, un réplicat peut être un même échantillon séquencé plusieurs fois, plusieurs prises d'échantillons au même endroit ou encore plusieurs échantillons sélectionnés en plusieurs endroits différents).

Ainsi, dans certains cas, la question qui importe n'est pas tant "Comment améliorer le modèle ?" que "Quelles sont les bonnes informations à croiser ?".

6.3.2 Complémentarité des méthodes : union des hits

Cumuler différentes stratégies pour obtenir différents hits marginaux

Il existe généralement plusieurs méthodes possibles pour modéliser un motif biologique. Selon le degré de complexité des motifs, l'alternative peut être soit une approche générique, soit un outil spécialisé sur la recherche de ce type de motif en particulier.

Ces alternatives constituent une chance concernant les motifs complexes. En effet, chaque approche met en œuvre ses propres méthodes et stratégies. Or selon la stratégie de recherche,

les limites de la méthode vont différer et la détection des cas les plus extrêmes (parce que plus dégénérés, moins typiques, à la frontière...) aussi. Ainsi, si une stratégie suffit généralement à identifier la majorité des hits éligibles pour un motif complexe, multiplier les stratégies permet d'approcher l'exhaustivité des hits possibles.

Par exemple, dans le cadre de la recherche du motif MiHsmar1, 3 sources de hits étaient à disposition : les hits Logol, les hits Blastn et les annotations déjà existantes du motif dans le génome et obtenues via RepeatMasker [source : Y. Bigot et B. Piégu]. La couverture de chaque source est donnée dans le diagramme de Venn de la figure 6.1. On constate que Logol découvre spécifiquement 1 460 hits alors que Blastn en identifie 493 autres. Et l'union des deux stratégies permet de reconnaître 2 529 nouveaux hits.

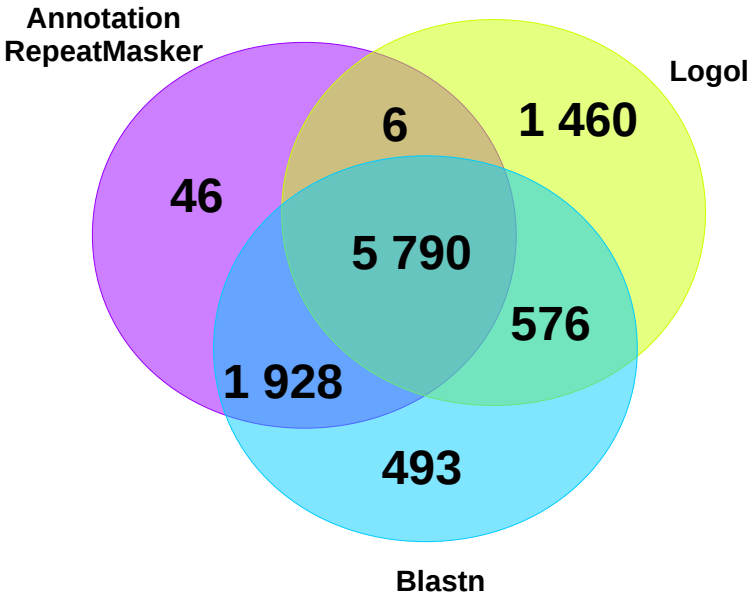


FIGURE 6.1 – Répartition des miHsmar1 détectés par RepeatMasker, Logol et Blastn, pour un total de 10 299 annotations [source : Y. Bigot et B. Piégu]

Dans les faits, il s'avère que les hits obtenus avec Blastn sont des motifs pour lesquelles les extrémités du motifs sont partiellement tronquées, ce qui n'avait pas été envisagé lors de la conception du modèle MiHsmar1, qui ne recherchait donc pas ce genre de hit.

Les hits spécifiques à Logol ont éventuellement une partie centrale de taille éloignée de celle du consensus, ce qui les rend difficile à atteindre par BlastN.

Rationalisation du coût : pipeline d'analyse pour éliminer les cas simples

Les modèles les plus complexes peuvent être coûteux à rechercher. Lorsque ce coût s'avère gênant, il peut être judicieux de faire collaborer un modèle complexe avec un modèle simplifié, capable de prendre en charge les cas moins subtiles et de réserver l'utilisation du modèle complexe aux cas nécessitant réellement sa puissance.

Ainsi, lors de la recherche sur les amorces mutées dans les données de séquençage, nous avons travaillé sur 9 échantillons Illumina totalisant 5 223 138 séquences, chaque séquence devant être analysée avec deux modèles d'amorces : le forward et le reverse. Or l'analyse avec le modèle Logol d'un seul échantillon Illumina prend plus de deux semaines sur le cluster de

calcul de GenOuest.

Pour palier ce problème, nous avons séparé le problème en deux parties. Pour chaque amorce, l'analyse s'est faite en deux temps : d'abord, toutes les séquences sont analysées à l'aide d'une expression régulière pour identifier les amorce exactes (i.e. méthode usuelle de nos collaborateurs). Ensuite, les séquences pour lesquelles une amorce parfaite n'a pas pu être détectée sont rassemblées et dérépliquées. Ces séquences dérépliquées sont ensuite analysées avec la grammaire Logol pour rechercher les amorces mutées.

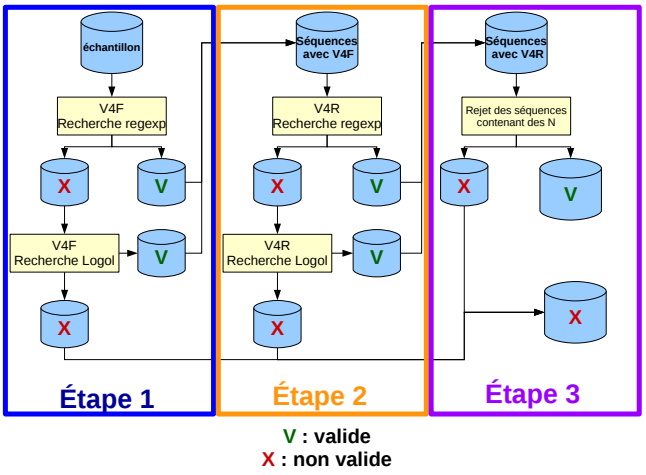


FIGURE 6.2 – Workflow d'analyse métagénomique couplant une recherche des amorces par expression régulière et par Logol

Le fait de ne déployer Logol que sur les séquences qui en ont vraiment besoin a permis de diminuer le temps d'analyse à $\sim$ 20h, ce qui, d'après les biologistes, constitue un temps acceptable pour obtenir des données plus précises.

6.3.3 Remise en question des motifs

L'un des constats qui est régulièrement revenu au cours de nos différents cas d'étude est le fait que les motifs initiaux sont rarement définitifs. Parfois, il s'agit de modifications (mineures) par rapport à l'axe de modélisation envisagé. Par exemple, dans le cas des amorces mutées (cf section 5.2, page 90), il s'est agi simplement d'autoriser le mélange de mutations qui étaient auparavant mutuellement exclusives. De la même façon, dans le cas des miHsmar1 (cf section 5.4, page 98), il s'agissait seulement de rajouter des nucléotides ambigus dans une séquence dégénérée.

Dans d'autres cas, la modification peut s'avérer plus importante, comme dans le cas des REP (cf 5.5) : les REP sont définis par une séquence consensus, composée de sections variables et de sections conservées. Mais dans les faits, les sections conservées se sont révélées aussi très dégénérées. Le modèle a finalement été orienté vers une séquence très dégénérée de façon globale et c'est la structure qui est devenue la clef du modèle. Dans le cas du site de fixation du facteur de transcription LXRα, la modification a été nécessaire dès le départ : bien que désigné sous le terme de DR4 (Direct Repeat 4) par les biologistes, il s'est avéré que ce motif n'était pas modélisable sous la forme d'une séquence répétée.

Un autre exemple est le cas, à nouveau, des miHsmar1, puisqu'à l'épreuve du croisement de données (dont des données CHIp-Seq), il s'est avéré que la partie fortement conservée du

motif n'était finalement pas les extrémités mais une partie plus interne. On pourrait donc maintenant intégrer ces nouvelles données au modèle et relancer l'analyse sur tout le génome à la recherche de nouveaux sites.

Il faut donc toujours garder un œil critique sur le motif de départ et envisager de le faire évoluer.

6.4 Intérêt des modèles grammaticaux de type Logol

Les travaux précédents ont permis de mettre en évidence trois atouts que possèdent les modèles grammaticaux de type Logol. Ces atouts sont respectivement la finesse de description, l'ambiguïté des solutions et la possibilité de définir un modèle par la négative.

6.4.1 Modèles grammaticaux de type Logol : finesse de description

Les modèles grammaticaux en général, et Logol en particulier, présentent l'avantage de pouvoir préciser des propriétés différentes en certains endroits du modèle. Par exemple, plutôt que d'associer un nombre de mutations autorisées à la globalité d'un motif, il est possible de préciser que ces mutations ne peuvent avoir lieu dans une région précise du motif ou au contraire imposer une répartition de ces mutations le long du motif (cf exemple en figure 6.3).

1) Motif brut : `ATCGATCGATCG` avec 3 sub.
2) Motif affiné : `ATCG` + `ATCGATCG` avec 3 sub.
3) Motif affiné : `ATCG` avec 1 sub. + `ATCGATCG` avec 2 sub.

FIGURE 6.3 – Exemple de modèles "fins" de mutations. 1) Modèle avec mutations globales 2) Modèle avec mutations concentrés sur la fin du motif 3) Modèle avec répartition des mutations le long de la séquence

Cette particularité s'est ainsi avérée très intéressante pour la modélisation des amorces mutées (cf cas d'étude 5.2 p90) : en subdivisant le motif en deux alternatives, il a ainsi été possible de produire un modèle recherchant simultanément un motif avec jusqu'à 2 substitutions ou un motif avec jusqu'à 1 insertion/délétion. Cette modélisation plus fine du profil de mutation permet ainsi d'identifier plus de reads exploitables et détecter davantage d'espèces présentes dans les échantillons.

Cette particularité a été par ailleurs important pour la modélisation des motifs miHsmar1 (cf cas d'étude 5.4 p98) : il a ainsi été possible de prendre en compte l'information de conservation de la tige du motif miHsmar1, qui est plus conservé à la base et plus dégénéré au sommet, en associant des valeurs de mutations maximum différentes à ces deux régions. Cette prise en compte plus précise des caractéristiques du profil a permis d'identifier spécifiquement de nouvelles annotations du motif miHsmar1 présentes dans le génome humain.

Les modèles grammaticaux permettent de passer de modèles monolithiques à des modèles fragmentés pour lesquels il est possible d'associer des propriétés distinctes entre chaque fragment, plus à même de représenter certaines particularités des motifs biologiques. C'est cette finesse qui permet notamment d'associer des informations de séquences et des informations de structure au sein d'un même modèle.

6.4.2 Modèles grammaticaux de type Logol : ambiguïté des solutions

Les modèles ambigus sont des modèles qui proposent plusieurs solutions satisfaisantes pour un même segment de séquence (cf exemple en figure 6.4).

Séquence : `ATGACA`

Modèle : `ATG` avec au plus 1 indel + `ACA` avec au plus 1 indel

Solution 1 : `ATG` + `ACA`
Solution 2 : `AT_` + `GACA`
Solution 3 : `ATGA` + `_CA`

FIGURE 6.4 – Exemple de solutions pour une séquence analysée avec un modèle ambigu

Par corollaire, cela implique que pour une séquence donnée, un modèle ambigu permet de déterminer toutes les façons possibles de satisfaire aux contraintes imposées. Cette particularité s'est avérée utile pour la recherche exhaustive des ARNm alternatifs productibles à partir de la séquence d'un gène (cas d'étude 5.6 p106) : en modélisant uniquement les règles d'épissage aboutissant à l'obtention d'un ARNm valide, l'analyse d'une séquence permet de retrouver l'ensemble des combinaisons d'introns/exons différentes, donc l'ensemble des ARNm issus de cette séquence.

Néanmoins, cette particularité peut s'avérer à double tranchant : parfois, il n'est pas souhaitable, du fait de l'explosion combinatoire générée, d'obtenir l'ensemble des solutions possibles d'un modèle, comme ç'a été le cas pour la recherche du signal de polyadénylation (cf cas d'étude 5.3 p94). Il est donc nécessaire de laisser le choix à l'utilisateur de bénéficier ou non de l'ambiguïté.

Dans le cadre des grammaires, l'ambiguïté permet de définir un modèle non plus par une information de séquence ou une information de structure mais aussi par des informations liées à des règles de production, tel que les règles d'épissage dans les ARNm.

6.4.3 Modèles grammaticaux de type Logol : définition par la négative

Les grammaires permettent de définir des éléments de modèles via la négation, c'est-à-dire en définissant ce que le modèle ne doit pas reconnaître.

Cette propriété s'est avérée nécessaire pour la modélisation du signal de polyadénylation ainsi que pour la recherche d'ARNm alternatif dans les introns/exons. En effet, dans le cas du signal de polyadénylation, on sait que la protéine est capable de se fixer sur l'ensemble des variants de AATAAA à une mutation près, excepté précisément GATAAA. De la même façon, dans le cas des motifs exons, il fallait vérifier que les codons présents ne forment pas de codon stop, donc qu'ils soient différents de TAA, TGA et TAG. Dans ces deux cas, il est possible de spécifier ce à quoi ne doit pas correspondre l'élément concerné, via l'utilisation d'un opérateur de négation (« ! » pour le langage Logol).

L'opérateur de négation présente l'inconvénient d'imposer une négation d'un seul tenant. Il n'est donc pas possible de prendre en compte des influences distantes, comme cela s'avère

nécessaire pour vérifier qu'un codon résultant de l'épissage d'un intron ne soit pas un codon stop (cf exemple de la figure Z), ou encore pour imposer un phénomène de mésappariemment dans une tige (cf cas d'étude des REP, partie 5.5 p102). Pour couvrir ces cas de figure, Logol propose la définition de morphisme, qui permet de définir des dépendances entre deux éléments, même si ces éléments sont « éclatés » le long de la séquence ou du modèle.

Séquence réelle : `ATG TCC TGT TGA AGC GAA CAA ...`
Séquence « théorique » après retrait de l'intron : `ATG TCC T__ ___ __C GAA CAA ...`
Codons résultant du retrait de l'intron : `ATG TCC TCG AAC AA...`

FIGURE 6.5 – Exemple de nécessité de vérifier la dépendance entre des points distants d'une séquence : le codon TCG qui sera formé après épissage de l'intron n'existe pas tel quel sur la séquence lors de l'analyse et doit donc être considéré en deux parties distinctes dans le modèle

6.4.4 Bilan de l'utilité des modèles grammaticaux

Après étude approfondie de différentes applications biologiques, il apparait que les modèles grammaticaux ont vraiment une utilité pour modéliser des motifs biologiques et permettre l'analyse des séquences. Nous avons identifié trois intérêts majeurs. Le premier est la grande finesse permise par ces modèles, qui permet de distinguer des parties de modèles plus ou moins sujettes aux mutations. Le deuxième est la possibilité d'exclure des cas de figure grâce à la négation. La troisième est la possibilité de définir des modèles ambigus permettant de capter l'ensemble des solutions possibles d'une séquence pour un modèle donné. Ces trois propriétés traduisent la souplesse des modèles grammaticaux qui permettent de retranscrire plus fidèlement la réalité biologique (propriétés variables selon les régions, dépendances entre des éléments mêmes distants, interdiction d'éléments particuliers, etc.). Bien entendu, cette souplesse n'est pas sans contrepartie et peut entraîner un surcoût de calcul par rapport aux méthodes plus classiques (cf Annexe A.2). Cependant, cette modélisation représente une véritable chance pour certains aspects de la reconnaissance de motifs, et tout particulièrement pour la mise au point des motifs ou la recherche de motifs composites.

CHAPITRE 7

Conclusion et perspectives

Bien que le concept des modèles grammaticaux appliqués à la reconnaissance de motifs biologiques ait pris son essor à partir des années 80 [BB84], notamment grâce à David Searls qui a introduit le formalisme des grammaires à variable de chaines (SVG, [Sea88]) son utilisation par la communauté des biologistes est restée très marginale et méconnue. Pourtant, ces modèles grammaticaux avancés (tels que Logol [BSN14] développé dans l'équipe Dyliss où a lieu cette thèse) offrent une grande expressivité des modèles. Pendant cette thèse, au cours d'une démarche exploratoire aussi variée qu'il nous a été possible de le faire, nous avons cherché à savoir si de tels modèles constituaient effectivement une solution adaptée pour modéliser la réalité biologique, et s'ils permettaient de réaliser une reconnaissance de motif opérationnelle.

Pour cela, nous avons tenté de résoudre six applications de reconnaissance de motifs sur des génomes ou des séquences ARN, en les confrontant à la modélisation grammaticale au travers du langage Logol.

A l'occasion de ces travaux, nous avons produits quelques résultats biologiques ou méthodologiques, tels que la prédiction d'un gène candidat à une régulation directe par le facteur de transcription LXRα, la production d'une application web adossée à une base de données de gènes eucaryotes, l'ajout d'annotations d'éléments répétés sur le génome humaine ou la mise en valeur d'amorces mutées en métagénomique.

La question au cœur de cette thèse était de déterminer si les modèles grammaticaux tels que Logol pouvaient avoir un intérêt pour la modélisation de motifs biologiques.

La réponse que nous apportons est positive. Nous avons notamment identifié trois éléments d'expressivité particulièrement utiles, apportés par les formalismes grammaticaux à haut niveau d'expressivité. Un premier atout est la finesse permise dans les modèles, qui permettent de définir des motifs avec des zones différemment mutés ou à faire cohabiter des contraintes de séquence et de structure. Un deuxième atout réside dans l'ambiguïté des solutions permise par la recherche de ces grammaires, qui permet d'identifier les différentes façons possibles d'appliquer un modèle sur une même séquence, ce qui permet de définir des règles générales de production de motifs pour obtenir des solutions exhaustives (e.g. règles d'épissage pour la recherche d'ARNm alternatifs associés à la séquence d'un gène). Un troisième atout est la possibilité de définir par la négative le contenu d'un modèle, c'est-à-dire par ce qu'il ne doit pas reconnaître (i.g. mésappariemment, "codon non-stop", etc).

Ces diverses particularités font que les modèles grammaticaux peuvent exprimer une plus large gamme de contraintes et d'informations que d'autres outils et permettent d'approcher ainsi au mieux « le langage des gènes », comme l'a exprimé David Searls [Sea02].

De plus, les travaux menés durant cette thèse ont montré que les modèles de départ étaient par nature évolutifs. Le fait d'utiliser un langage de modélisation généraliste permet de faire évoluer un modèle dans n'importe quel direction sans être contraint par la capacité d'expression du modèle.

Ainsi, les modèles grammaticaux de type Logol gagneraient à prendre toute leur place dans le panorama des outils de reconnaissance de motifs au service des biologistes.

Perspectives

L'intérêt du langage Logol ayant été montré, une perspective immédiate est de permettre que Logol soit réellement utilisé.

Une action dans ce sens serait de rendre les grammaires plus accessibles aux biologistes, notamment au travers du développement de l'ergonomie et de la visibilité de Logol. En l'état actuel, la prise en main de Logol n'est pas assez intuitive pour être aisément utilisée par les biologistes. Cette thèse nous a permis de faire un pas dans cette direction en identifiant un certain nombre de voies d'évolution.

Un autre facteur limitant de l'utilisation de Logol réside dans le temps de recherche de l'analyseur. Après expertise des points faibles de l'analyseur, il faudra peut-être reconsidérer complètement sa mise en œuvre ou a minima apporter des points d'amélioration. En particulier, puisque le langage a été conçu pour pouvoir tout rechercher, l'analyseur actuel se prive de certains raccourcis de recherche, qui ne seraient pas adaptés de façon généraliste. On pourrait mettre en place une pré-analyse de la grammaire qui permettrait de déterminer quels mécanismes de recherche appliquer en fonction du modèle. Par exemple, il est possible de se mettre dans une logique de pipeline, et d'orienter l'analyse vers différentes méthodes de recherche en fonction de la complexité de la grammaire, de façon à rechercher les motifs les plus simples avec des outils optimisés pour cette tâche. Par ailleurs, on pourrait s'appuyer sur les applications pour lesquels Logol s'est avéré le plus utile, pour rendre leurs types de motifs particulièrement opérationnels.

Dans le même ordre d'idée, on a constaté qu'une bonne question à se poser n'était pas seulement « Quelle est La bonne méthode de recherche à utiliser ? » mais aussi « Comment faire coopérer des méthodes complémentaires ? ». Il s'est d'ailleurs montré déterminant de faire fonctionner Logol en coopération avec d'autres méthodes de recherche afin de pré-traiter les cas simples d'un modèle pour n'effectuer les recherches complexes que sur les séquences qui le nécessitent réellement. Pour cela, il faudrait augmenter la suite logicielle Logol en la dotant de facilités d'interconnection avec d'autres analyseurs.

Une autre perspective serait d'ajouter une dimension au formalisme grammatical. En effet, un handicap des méthodes grammaticales est le fait de ne pas classer les solutions. On a pu voir dans le cas de la recherche des sites de fixation (cf Chapitre 2) que le poids associé aux hits des matrices était un facteur permettant de mieux discriminer les meilleurs résultats. On pourrait par exemple, suivant le principe des grammaires stochastiques, envisager d'ajouter des possibilités de scores dans les grammaires Logol pour permettre cette distinction des hits entre eux.

La thèse a été l'occasion d'explorer plusieurs applications biologiques, qui portent chacune des perspectives qui leur sont propres. Parmi les plus intéressantes, on notera :

Orthocis : Comme indiqué page 61, l'approche par boules de mots peut aider à la recherche des sites de fixation d'un facteur de transcription. Contrairement aux matrices, cette approche ne permet pas d'identifier des motifs qui ne dérivent pas directement des séquences de références (on a vu qu'en fonction du seuil de validation, les matrices

peuvent autoriser des voisins à plus de deux substitutions d'une référence si tant est que les mutations interviennent sur des positions peu conservées parmi les références), mais présente l'avantage d'être exhaustive dans la recherche des voisins d'une référence. La recherche par boule de mots peut être mise en place par modèle grammatical ou par d'autres méthodes de recherche des variants d'un mot.

Dans un autre ordre d'idée, il serait utile de permettre de multiples croisements de données. Ces données peuvent être externes (tels que des pics ChIP-Seq ou des données d'accessibilité de la chromatine) ou bien liées à la séquence. On peut aussi par exemple post-filtrer les hits des matrices pour conserver ceux ressemblant suffisamment à une structure répétée de type DR4, par exemple ceux contenant une répétition de l'hexamère à 3 substitutions près).

De telles possibilités, et bien d'autres encore (tel qu'étendre la zone de recherche de hits en ajoutant la partie des "upstreams alternatifs" à la base de données ou bien cibler la co-occurrence de sites de fixation de plusieurs protéines ou encore intégrer les orthologies one-to-many, cf. Annexe 4) pourraient être utilement ajoutées à l'application Orthocis.

MiHsmar1 : un certain nombre de hits putatifs de miHsmar1 ont été validés par croisement avec des pics ChIP-Seq. Parmi ceux-là, certains ne sont pas acceptés par le modèle Logol (ils ont été identifiés par RepeatMasker ou Blastn). On souhaiterait maintenant identifier les manquements dans les propriétés du modèle qui expliquent l'absence de ces annotations dans les résultats Logol. Les premiers travaux indiquent par exemple que les hits spécifiques à Blast présentent une dégénérescence plus forte qu'envisagée sur l'extérieur de la tige du motif miHsmar1. En relançant le nouveau modèle sur le génome humain, on peut espérer découvrir de nouvelles annotations du miHsmar1.

Inversement, il serait intéressant d'identifier quelles sont les propriétés des annotations spécifiquement identifiés avec Logol (1500 hits de haute confiance ne sont trouvés que par Logol) pour comprendre finement quelles sont les caractéristiques du motif miHsmar1 et ainsi adopter de meilleures stratégies pour le rechercher.

Introns/exons : l'identification à partir d'un gène de tous les ARNm alternatifs potentiels présente un réel intérêt. Les modèles grammaticaux semblent une réponse adaptée à cette problématique (par la gestion de l'ambiguïté par les grammaires), or actuellement, le modèle manque de spécificité, produisant une avalanche de hits sur toutes les parties d'une séquence génomique, même non géniques.

Au-delà d'une meilleure caractérisation des signaux d'épissage, on peut envisager d'améliorer la spécificité en indiquant à la grammaire certains sites (tels que starts, stops, donneurs ou accepteurs d'épissage) qui seraient déjà identifiés par ailleurs. L'idée serait donc de pouvoir annoter les séquences analysées et de permettre d'utiliser ces annotations dans la grammaire pour orienter la recherche. Il s'agirait donc d'enrichir Logol en lui donnant la capacité de prendre en compte des séquences annotées.

Une autre piste de réflexion serait de générer des grammaires « au cas par cas » pour chaque gène, en prenant en compte des informations issues de gènes orthologues (recherche d'ARNm alternatifs partagés par plusieurs espèces). Cette dernière piste est actuellement à l'étude en collaboration avec Samuel Blanquart.

Recherche d'amorces mutées : la recherche d'amorces mutées dans les données de métagénomiques pour l'estimation de la biodiversité est une problématique très intéressante. Les travaux que nous avons menés avec Frédéric Mahé et Micah Dunthorn semblent être une preuve de concept de l'intérêt qu'il y a à aller chercher ces amorces mutées dans les données : ceci permet d'obtenir davantage de reads exploitables et, *in*

fine, de détecter la présence de nouvelles espèces présentes dans les échantillons. Malheureusement, dans notre cas de figure, les espèces eucaryotes présentes dans les sols tropicaux sont trop peu connues et seules une minorité des séquences obtenues, que ce soit avec des amorces exactes ou mutées, peuvent être caractérisées par les banques de données. Ainsi, s'il est possible de démontrer que nos nouvelles données contiennent effectivement des séquences issues d'espèces présentes, il n'est actuellement pas possible d'établir clairement la proportion d'informations et de bruits ajoutés par nos nouvelles données.

Une perspective intéressante serait donc de reprendre ce type d'analyse en la transposant sur un milieu actuellement mieux connu, tel que par exemple la flore intestinale chez l'homme.

Bibliographie

[AGM+90] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3) :403–410, October 1990.

[ALLL+15] Aymeric Antoine-Lorquin, Sandrine Lagarrigue, Frédéric Lecerf, Jacques Nicolas, and Catherine Belleannée. Comparison of the targets obtained by a scoring matrix and by a regular expression. Application to the search for LXR binding sites. In *JOBIM 2015- 16e Journées Ouvertes en Biologie, Informatique et Mathématiques*, Clermont-Ferrand, France, July 2015.

[ALMDB16] Aymeric Antoine-Lorquin, Frédéric Mahé, Micah Dunthorn, and Catherine Belleannée. Impact de la recherche d'amorces mutées sur les résultats d'analyses métagénomiques. In *Journées Ouvertes en Biologie, Informatique et Mathématiques (JOBIM)*, Jobim-2016, Rennes, France, June 2016. Société Française de Bioinformatique (SFBI).

[BB84] V. Brendel and H. G. Busse. Genome structure described by formal languages. *Nucleic Acids Research*, 12(5) :2561–2568, March 1984.

[BBB+09] Timothy L. Bailey, Mikael Boden, Fabian A. Buske, Martin Frith, Charles E. Grant, Luca Clementi, Jingyuan Ren, Wilfred W. Li, and William S. Noble. MEME Suite : tools for motif discovery and searching. *Nucleic Acids Research*, 37(suppl 2) :W202–W208, January 2009.

[BBO+78] R Breathnach, C Benoist, K O'Hare, F Gannon, and P Chambon. Ovalbumin gene : evidence for a leader sequence in mRNA and DNA sequences at the exon-intron boundaries. *Proceedings of the National Academy of Sciences of the United States of America*, 75(10) :4853–4857, October 1978.

[BFZ+10] Michaël Bekaert, Andrew E. Firth, Yan Zhang, Vadim N. Gladyshev, John F. Atkins, and Pavel V. Baranov. Recode-2 : new design, new search tools, and many more genes. *Nucleic Acids Research*, 38(Database issue) :D69–D74, January 2010.

[BJI92] I. Brierley, A. J. Jenner, and S. C. Inglis. Mutational analysis of the "slippery-sequence" component of a coronavirus ribosomal frameshifting signal. *Journal of Molecular Biology*, 227(2) :463–479, September 1992.

[BKK15] Weidong Bao, Kenji K. Kojima, and Oleksiy Kohany. Repbase Update, a database of repetitive elements in eukaryotic genomes. *Mobile DNA*, 6 :11, 2015.

[BL06] Samuel Blanquart and Nicolas Lartillot. A Bayesian compound stochastic process for modeling nonstationary and nonhomogeneous sequence evolution. *Molecular Biology and Evolution*, 23(11) :2058–2071, November 2006.

[BPS98] Christopher B. Burge, Richard A. Padgett, and Phillip A. Sharp. Evolutionary Fates and Origins of U12-Type Introns. *Molecular Cell*, 2(6) :773–785, December 1998.

[Bri95] I. Brierley. Ribosomal frameshifting viral RNAs. *The Journal of General Virology*, 76 (Pt 8) :1885–1892, August 1995.

[BSN14] Catherine Belleannée, Olivier Sallou, and Jacques Nicolas. Logol : Expressive Pattern Matching in sequences. Application to Ribosomal Frameshift Modeling. In Matteo Comin, Lukas Kall, Elena Marchiori, Alioune Ngom, and Jagath Rajapakse, editors, *PRIB2014 - Pattern Recognition in Bioinformatics, 9th IAPR International Conference*, volume 8626, pages 34–47, Stockholm, Sweden, August 2014. Springer International Publishing.

[BSP+94] S. Bachellier, W. Saurin, D. Perrin, M. Hofnung, and E. Gilson. Structural and functional diversity among bacterial interspersed mosaic elements (BIMEs). *Molecular Microbiology*, 12(1) :61–70, April 1994.

[BSTU06] Marina Barsky, Ulrike Stege, Alex Thomo, and Chris Upton. A New Algorithm for Fast All-Against-All Substring Matching. In Fabio Crestani, Paolo Ferragina, and Mark Sanderson, editors, *String Processing and Information Retrieval*, number 4209 in Lecture Notes in Computer Science, pages 360–366. Springer Berlin Heidelberg, 2006.

[Bul04] Martha L Bulyk. Computational prediction of transcription-factor binding site locations. *Genome Biology*, 5(1) :201, 2004.

[Car98] Michael Carey. The Enhanceosome and Transcriptional Synergy. *Cell*, 92(1) :5–8, January 1998.

[Cho56] N Chomsky. Three models for the description of language. *IRE Transactions on Information Theory*, 2(3) :113–124, September 1956.

[CKS01] J. Y. Chiang, R. Kimmel, and D. Stroup. Regulation of cholesterol 7alpha-hydroxylase gene (CYP7a1) transcription by the liver orphan receptor (LXRalpha). *Gene*, 262(1-2) :257–265, January 2001.

[CLEC95] T. Chi, P. Lieberman, K. Ellwood, and M. Carey. A general mechanism for transcriptional synergy by eukaryotic activators. *Nature*, 377(6546) :254–257, September 1995.

[CMMN06] Kiki Chu, Makoto Miyazaki, Weng Chi Man, and James M. Ntambi. Stearoyl-coenzyme A desaturase 1 deficiency protects against hypertriglyceridemia and increases plasma high-density lipoprotein cholesterol induced by liver X receptor activation. *Molecular and Cellular Biology*, 26(18) :6786–6798, September 2006.

[CUBF06] Richard Cordaux, Swalpa Udit, Mark A. Batzer, and Cédric Feschotte. Birth of a chimeric primate gene by capture of the transposase gene from a mobile element. *Proceedings of the National Academy of Sciences of the United States of America*, 103(21) :8101–8106, May 2006.

[CVZ15] Marie-Christine Champomier-Vergès and Monique Zagorec. *La métagénomique : développements et futures applications*. Éd. Quae, Versailles, 2015. OCLC : 904552149.

[CWB+99] J. M. Clément, C. Wilde, S. Bachellier, P. Lambert, and M. Hofnung. IS1397 is active for transposition into the chromosome of Escherichia coli K-12 and inserts specifically into palindromic units of bacterial interspersed mosaic elements. *Journal of Bacteriology*, 181(22) :6929–6936, November 1999.

[dCSG+06] Edouard de Castro, Christian J. A. Sigrist, Alexandre Gattiker, Virginie Bulliard, Petra S. Langendijk-Genevaux, Elisabeth Gasteiger, Amos Bairoch, and Nicolas Hulo. ScanProsite : detection of PROSITE signature matches and ProRule-associated functional and structural residues in proteins. *Nucleic Acids Research*, 34(Web Server issue) :W362–365, July 2006.

[DDD+09] O. Demeure, C. Duby, C. Desert, S. Assaf, D. Hazard, H. Guillou, and S. Lagarrigue. Liver X receptor alpha regulates fatty acid synthase expression in chicken. *Poultry Science*, 88(12) :2628–2635, December 2009.

[DGWM07] Rezarta Islamaj Dogan, Lise Getoor, W. John Wilbur, and Stephen M. Mount. SplicePort—An interactive splice-site analysis tool. *Nucleic Acids Research*, 35(Web Server issue) :W285–W291, July 2007.

[DIW91] J. D. Dinman, T. Icho, and R. B. Wickner. A -1 ribosomal frameshift in a double-stranded RNA virus of yeast forms a gag-pol fusion protein. *Proceedings of the National Academy of Sciences of the United States of America*, 88(1) :174–178, January 1991.

[DKMR04] Ken Dower, Nicolas Kuperwasser, Houra Merrikh, and Michael Rosbash. A synthetic A tail rescues yeast nuclear accumulation of a ribozyme-terminated transcript. *RNA (New York, N.Y.)*, 10(12) :1888–1899, December 2004.

[DLD+11] O. Demeure, F. Lecerf, C. Duby, C. Desert, S. Ducheix, H. Guillou, and S. Lagarrigue. Regulation of LPCAT3 by LXR. *Gene*, 470(1-2) :7–11, January 2011.

[DLO97] M. Dsouza, N. Larsen, and R. Overbeek. Searching for patterns in genomic data. *Trends in genetics : TIG*, 13(12) :497–498, December 1997.

[DS94] S. Dong and D. B. Searls. Gene structure prediction by linguistic methods. *Genomics*, 23(3) :540–551, October 1994.

[dVAH+15] Colomban de Vargas, Stéphane Audic, Nicolas Henry, Johan Decelle, Frédéric Mahé, Ramiro Logares, Enrique Lara, Cédric Berney, Noan Le Bescot, Ian Probert, Margaux Carmichael, Julie Poulain, Sarah Romac, Sébastien Colin, Jean-Marc Aury, Lucie Bittner, Samuel Chaffron, Micah Dunthorn, Stefan Engelen, Olga Flegontova, Lionel Guidi, Aleš Horák, Olivier Jaillon, Gipsi Lima-Mendez, Julius Lukeš, Shruti Malviya, Raphael Morard, Matthieu Mulot, Eleonora Scalco, Raffaele Siano, Flora Vincent, Adriana Zingone, Céline Dimier, Marc Picheral, Sarah Searson, Stefanie Kandels-Lewis, Tara Oceans Coordinators, Silvia G. Acinas, Peer Bork, Chris Bowler, Gabriel Gorsky, Nigel Grimsley, Pascal Hingamp, Daniele Iudicone, Fabrice Not, Hiroyuki Ogata, Stephane Pesant, Jeroen Raes, Michael E. Sieracki, Sabrina Speich, Lars Stemmann, Shinichi Sunagawa, Jean Weissenbach, Patrick Wincker, and Eric Karsenti. Ocean plankton. Eukaryotic plankton diversity in the sunlit ocean. *Science (New York, N.Y.)*, 348(6237) :1261605, May 2015.

[DW92] J. D. Dinman and R. B. Wickner. Ribosomal frameshifting efficiency and gag/gag-pol ratio are critical for yeast M1 double-stranded RNA virus propagation. *Journal of Virology*, 66(6) :3669–3676, June 1992.

[EA60] Mary Edmonds and Richard Abrams. Polynucleotide Biosynthesis : Formation of a Sequence of Adenylate Units from Adenosine Triphosphate by an Enzyme from Thymus Nuclei. *Journal of Biological Chemistry*, 235(4) :1142–1149, January 1960.

[EB97] O. Espéli and F. Boccard. In vivo cleavage of Escherichia coli BIME-2 repeats by DNA gyrase : genetic characterization of the target and identification of the cut site. *Molecular Microbiology*, 26(4) :767–777, November 1997.

[Edd95] S. R. Eddy. Multiple alignment using hidden Markov models. *Proceedings / ... International Conference on Intelligent Systems for Molecular Biology ; ISMB. International Conference on Intelligent Systems for Molecular Biology*, 3 :114–120, 1995.

[Edd98] S. R. Eddy. Profile hidden Markov models. *Bioinformatics*, 14(9) :755–763, January 1998.

[EJM80] Werner Ebeling and Miguel A. Jiménez-Montaño. On grammars, complexity, and information measures of biological macromolecules. *Mathematical Biosciences*, 52(1) :53–71, November 1980.

[EMB01] O. Espéli, L. Moulin, and F. Boccard. Transcription attenuation associated with bacterial repetitive extragenic BIME elements. *Journal of Molecular Biology*, 314(3) :375–386, November 2001.

[EZ11] Mourad Elloumi and Albert Y. Zomaya. *Algorithms in Computational Molecular Biology : Techniques, Approaches and Applications*. Wiley-Blackwell, 1 edition edition, February 2011.

[GCMS99] J. H. Graber, C. R. Cantor, S. C. Mohr, and T. F. Smith. Genomic detection of new yeast pre-mRNA 3'-end-processing signals. *Nucleic Acids Research*, 27(3) :888–894, February 1999.

[GG04] Paul P Gardner and Robert Giegerich. A comprehensive comparison of comparative RNA structure prediction approaches. *BMC Bioinformatics*, 5(1) :140, 2004.

[Gie00] Robert Giegerich. Explaining and Controlling Ambiguity in Dynamic Programming. In Raffaele Giancarlo and David Sankoff, editors, *Combinatorial Pattern Matching*, number 1848 in Lecture Notes in Computer Science, pages 46–59. Springer Berlin Heidelberg, June 2000. DOI : 10.1007/3-540-45123-4_6.

[GLL+03] Giorgio Grillo, Flavio Licciulli, Sabino Liuni, Elisabetta Sbisà, and Graziano Pesole. PatSearch : A program for the detection of patterns and structural motifs in nucleotide sequences. *Nucleic Acids Research*, 31(13) :3608–3612, July 2003.

[GPH90] E. Gilson, D. Perrin, and M. Hofnung. DNA polymerase I and a protein complex bind specifically to E. coli palindromic unit highly repetitive DNA : implications for bacterial chromosome organization. *Nucleic Acids Research*, 18(13) :3941–3952, July 1990.

[GS95] Z. Guo and F. Sherman. 3'-end-forming signals of yeast mRNA. *Molecular and Cellular Biology*, 15(11) :5983–5990, November 1995.

[GST+16] Kristian A. Gray, Ruth L. Seal, Susan Tweedie, Mathew W. Wright, and Elspeth A. Bruford. A review of the new HGNC gene family resource. *Human Genomics*, 10 :6, 2016.

[GWMH14] Lori Glenwinkel, Di Wu, Gregory Minevich, and Oliver Hobert. Targetortho : A phylogenetic footprinting tool to identify transcription factor targets. *Genetics*, 197(1) :61–76, 2014.

[Hea87] Tom Head. Formal language theory and DNA : An analysis of the generative capacity of specific recombinant behaviors. *Bulletin of Mathematical Biology*, 49(6) :737–759, January 1987.

[Hen96] Steven Henikoff. Scores for sequence searches and alignments. *Current Opinion in Structural Biology*, 6(3) :353–360, June 1996.

[HFC+16] Robert Hubley, Robert D. Finn, Jody Clements, Sean R. Eddy, Thomas A. Jones, Weidong Bao, Arian F. A. Smit, and Travis J. Wheeler. The Dfam database of repetitive DNA families. *Nucleic Acids Research*, 44(D1) :D81–D89, April 2016.

[HHM+07] Susan M. Huse, Julie A. Huber, Hilary G. Morrison, Mitchell L. Sogin, and David Mark Welch. Accuracy and quality of massively parallel DNA pyrosequencing. *Genome Biology*, 8(7) :R143, 2007.

[HMB+16] Javier Herrero, Matthieu Muffato, Kathryn Beal, Stephen Fitzgerald, Leo Gordon, Miguel Pignatelli, Albert J. Vilella, Stephen M. J. Searle, Ridwan Amode, Simon Brent, William Spooner, Eugene Kulesha, Andrew Yates, and Paul Flicek. Ensembl comparative genomics resources. *Database*, 2016 :bav096, January 2016.

[HROS07] Thomas M. Hansen, S. Nader S. Reihani, Lene B. Oddershede, and Michael A. Sørensen. Correlation between mechanical strength of messenger RNA pseudoknots and ribosomal frameshifting. *Proceedings of the National Academy of Sciences of the United States of America*, 104(14) :5830–5835, April 2007.

[HWMS10] Susan M Huse, David Mark Welch, Hilary G Morrison, and Mitchell L Sogin. Ironing out the wrinkles in the rare biosphere through improved OTU clustering. *Environmental Microbiology*, 12(7) :1889–1898, July 2010.

[Inf] Dictionnaire de bactériologie vétérinaire. Infothèque francophone.

[Jas16] Emmanuel Jaspard. Biochimie enzymologie bioinformatique Enseignement recherche, 2016.

[JK09] Julien Jorda and Andrey V. Kajava. T-REKS : identification of Tandem REpeats in sequences with a K-meanS based algorithm. *Bioinformatics*, 25(20) :2632–2638, October 2009.

[JLP+02] Sean B. Joseph, Bryan A. Laffitte, Parthive H. Patel, Michael A. Watson, Karen E. Matsukuma, Robert Walczak, Jon L. Collins, Timothy F. Osborne, and Peter Tontonoz. Direct and indirect mechanisms for regulation of fatty acid synthase gene expression by liver X receptors. *The Journal of Biological Chemistry*, 277(13) :11019–11025, March 2002.

[JM84] M. A. Jiménez-Montaño. On the syntactic structure of protein sequences and the concept of grammar complexity. *Bulletin of Mathematical Biology*, 46(4) :641–659, 1984.

[JTZ90] J. A. Jaeger, D. H. Turner, and M. Zuker. Predicting optimal and suboptimal secondary structure for RNA. *Methods in Enzymology*, 183 :281–306, 1990.

[JV85] T. Jacks and H. E. Varmus. Expression of the Rous sarcoma virus pol gene by ribosomal frameshifting. *Science (New York, N.Y.)*, 230(4731) :1237–1242, December 1985.

[KAWHW12] Andrew T. Kwon, David J. Arenillas, Rebecca Worsley Hunt, and Wyeth W. Wasserman. oPOSSUM-3 : advanced analysis of regulatory motif over-representation across genes or ChIP-Seq datasets. *G3 (Bethesda, Md.)*, 2(9) :987–1002, September 2012.

[KC04] Vanessa Khemici and Agamemnon J. Carpousis. The RNA degradosome and poly(A) polymerase of Escherichia coli are required in vivo for the degradation of small mRNA decay intermediates containing REP-stabilizers. *Molecular Microbiology*, 51(3) :777–790, February 2004.

[KNB01] H. Kontos, S. Napthine, and I. Brierley. Ribosomal pausing at a frameshifter RNA pseudoknot is sensitive to reading phase but shows little correlation with frameshift efficiency. *Molecular and Cellular Biology*, 21(24) :8657–8670, December 2001.

[KR95] Gregory Kucherov and Michaël Rusinowitch. Matching a set of strings with variable length don't cares. In Zvi Galil and Esko Ukkonen, editors, *Combinatorial Pattern Matching*, number 937 in Lecture Notes in Computer Science, pages 230–247. Springer Berlin Heidelberg, 1995.

[Kur03] Stefan Kurtz. The vmatch large scale sequence analysis software. *Ref Type : Computer Program*, pages 4–12, 2003.

[LBHZS+11] Ronny Lorenz, Stephan H. Bernhart, Christian Höner Zu Siederdissen, Hakim Tafer, Christoph Flamm, Peter F. Stadler, and Ivo L. Hofacker. ViennaRNA Package 2.0. *Algorithms for molecular biology : AMB*, 6 :26, 2011.

[LBMD05] Mr Harvey Lodish, Mr Arnold Berk, Paul Matsudaira, and James Darnell. *Biologie moléculaire de la cellule*. De Boeck Supérieur, March 2005. Google-Books-ID : gSFbGLVFwMEC.

[LD14] Xiu-Qing Li and Donglei Du. Motif types, motif locations and base composition patterns around the RNA polyadenylation site in microorganisms, plants and animals. *BMC evolutionary biology*, 14 :162, 2014.

[Lie08] Aude Liefooghe. *Matrices score-position, algorithmes et propriétés*. phdthesis, Université des Sciences et Technologie de Lille - Lille I, July 2008.

[LLB+01] Eric S. Lander, Lauren M. Linton, Bruce Birren, Chad Nusbaum, Michael C. Zody, Jennifer Baldwin, Keri Devon, Ken Dewar, Michael Doyle, William Fitz-Hugh, Roel Funke, Diane Gage, Katrina Harris, Andrew Heaford, John Howland, Lisa Kann, Jessica Lehoczky, Rosie LeVine, Paul McEwan, Kevin McKernan, James Meldrim, Jill P. Mesirov, Cher Miranda, William Morris, Jerome Naylor, Christina Raymond, Mark Rosetti, Ralph Santos, Andrew Sheridan, Carrie Sougnez, Nicole Stange-Thomann, Nikola Stojanovic, Aravind Subramanian, Dudley Wyman, Jane Rogers, John Sulston, Rachael Ainscough, Stephan Beck, David Bentley, John Burton, Christopher Clee, Nigel Carter, Alan Coulson, Rebecca Deadman, Panos Deloukas, Andrew Dunham, Ian Dunham, Richard Durbin, Lisa French, Darren Grafham, Simon Gregory, Tim Hubbard, Sean Humphray, Adrienne Hunt, Matthew Jones, Christine Lloyd, Amanda McMurray, Lucy Matthews, Simon Mercer, Sarah Milne, James C. Mullikin, Andrew Mungall, Robert Plumb, Mark Ross, Ratna Shownkeen, Sarah Sims, Robert H. Waterston, Richard K. Wilson, LaDeana W. Hillier, John D. McPherson, Marco A. Marra, Elaine R. Mardis, Lucinda A. Fulton, Asif T. Chinwalla, Kymberlie H. Pepin, Warren R. Gish, Stephanie L. Chissoe, Michael C. Wendl, Kim D. Delehaunty, Tracie L. Miner, Andrew Delehaunty, Jason B. Kramer, Lisa L. Cook, Robert S. Fulton, Douglas L. Johnson, Patrick J. Minx, Sandra W. Clifton, Trevor Hawkins, Elbert Branscomb, Paul Predki, Paul Richardson, Sarah Wenning, Tom Slezak, Norman Doggett, Jan-Fang Cheng, Anne Olsen, Susan Lucas, Christopher Elkin, Edward Uberbacher, Marvin Frazier, Richard A. Gibbs, Donna M. Muzny, Steven E. Scherer, John B. Bouck, Erica J. Sodergren, Kim C. Worley, Catherine M. Rives, James H. Gorrell, Michael L. Metzker, Susan L. Naylor, Raju S. Kucherlapati, David L. Nelson, George M. Weinstock, Yoshiyuki Sakaki, Asao Fujiyama, Masahira Hattori, Tetsushi Yada, Atsushi Toyoda, Takehiko Itoh,

Chiharu Kawagoe, Hidemi Watanabe, Yasushi Totoki, Todd Taylor, Jean Weissenbach, Roland Heilig, William Saurin, Francois Artiguenave, Philippe Brottier, Thomas Bruls, Eric Pelletier, Catherine Robert, Patrick Wincker, André Rosenthal, Matthias Platzer, Gerald Nyakatura, Stefan Taudien, Andreas Rump, Douglas R. Smith, Lynn Doucette-Stamm, Marc Rubenfield, Keith Weinstock, Hong Mei Lee, JoAnn Dubois, Huanming Yang, Jun Yu, Jian Wang, Guyang Huang, Jun Gu, Leroy Hood, Lee Rowen, Anup Madan, Shizen Qin, Ronald W. Davis, Nancy A. Federspiel, A. Pia Abola, Michael J. Proctor, Bruce A. Roe, Feng Chen, Huaqin Pan, Juliane Ramser, Hans Lehrach, Richard Reinhardt, W. Richard McCombie, Melissa de la Bastide, Neilay Dedhia, Helmut Blöcker, Klaus Hornischer, Gabriele Nordsiek, Richa Agarwala, L. Aravind, Jeffrey A. Bailey, Alex Bateman, Serafim Batzoglou, Ewan Birney, Peer Bork, Daniel G. Brown, Christopher B. Burge, Lorenzo Cerutti, Hsiu-Chuan Chen, Deanna Church, Michele Clamp, Richard R. Copley, Tobias Doerks, Sean R. Eddy, Evan E. Eichler, Terrence S. Furey, James Galagan, James G. R. Gilbert, Cyrus Harmon, Yoshihide Hayashizaki, David Haussler, Henning Hermjakob, Karsten Hokamp, Wonhee Jang, L. Steven Johnson, Thomas A. Jones, Simon Kasif, Arek Kaspryzk, Scot Kennedy, W. James Kent, Paul Kitts, Eugene V. Koonin, Ian Korf, David Kulp, Doron Lancet, Todd M. Lowe, Aoife McLysaght, Tarjei Mikkelsen, John V. Moran, Nicola Mulder, Victor J. Pollara, Chris P. Ponting, Greg Schuler, Jörg Schultz, Guy Slater, Arian F. A. Smit, Elia Stupka, Joseph Szustakowki, Danielle Thierry-Mieg, Jean Thierry-Mieg, Lukas Wagner, John Wallis, Raymond Wheeler, Alan Williams, Yuri I. Wolf, Kenneth H. Wolfe, Shiaw-Pyng Yang, Ru-Fang Yeh, Francis Collins, Mark S. Guyer, Jane Peterson, Adam Felsenfeld, Kris A. Wetterstrand, Richard M. Myers, Jeremy Schmutz, Mark Dickson, Jane Grimwood, David R. Cox, Maynard V. Olson, Rajinder Kaul, Christopher Raymond, Nobuyoshi Shimizu, Kazuhiko Kawasaki, Shinsei Minoshima, Glen A. Evans, Maria Athanasiou, Roger Schultz, Aristides Patrinos, and Michael J. Morgan. Initial sequencing and analysis of the human genome. *Nature*, 409(6822) :860–921, February 2001.

[LLB09] Nicolas Lartillot, Thomas Lepage, and Samuel Blanquart. PhyloBayes 3 : a Bayesian software package for phylogenetic reconstruction and molecular dating. *Bioinformatics (Oxford, England)*, 25(17) :2286–2288, September 2009.

[LNKB99] J. Liphardt, S. Napthine, H. Kontos, and I. Brierley. Evidence for an RNA pseudoknot loop-helix interaction essential for efficient -1 ribosomal frameshifting. *Journal of Molecular Biology*, 288(3) :321–335, May 1999.

[LSHM+14] Simon Lax, Daniel P. Smith, Jarrad Hampton-Marcell, Sarah M. Owens, Kim M. Handley, Nicole M. Scott, Sean M. Gibbons, Peter Larsen, Benjamin D. Shogan, Sophie Weiss, Jessica L. Metcalf, Luke K. Ursell, Yoshiki Vázquez-Baeza, Will Van Treuren, Nur A. Hasan, Molly K. Gibson, Rita Colwell, Gautam Dantas, Rob Knight, and Jack A. Gilbert. Longitudinal analysis of microbial interaction between humans and the indoor environment. *Science (New York, N.Y.)*, 345(6200) :1048–1052, August 2014.

[LT00] Y. Luo and A. R. Tall. Sterol upregulation of human CETP expression in vitro and in transgenic mice by an LXR element. *The Journal of Clinical Investigation*, 105(4) :513–520, February 2000.

[Mar11] Marcel Martin. Cutadapt removes adapter sequences from high-throughput sequencing reads. *EMBnet.journal*, 17(1) :pp. 10–12, May 2011.

[MEG+01] Thomas J. Macke, David J. Ecker, Robin R. Gutell, Daniel Gautheret, David A. Case, and Rangarajan Sampath. RNAMotif, an RNA secondary structure definition and search algorithm. *Nucleic Acids Research*, 29(22) :4724–4735, November 2001.

[met] metabarcoding.org.

[MFA+16] Anthony Mathelier, Oriol Fornes, David J. Arenillas, Chih-yu Chen, Grégoire Denay, Jessica Lee, Wenqiang Shi, Casper Shyr, Ge Tan, Rebecca Worsley-Hunt, Allen W. Zhang, François Parcy, Boris Lenhard, Albin Sandelin, and Wyeth W. Wasserman. JASPAR 2016 : a major expansion and update of the open-access database of transcription factor binding profiles. *Nucleic Acids Research*, 44(Database issue) :D110–D115, January 2016.

[MKB+11] Fernando Meyer, Stefan Kurtz, Rolf Backofen, Sebastian Will, and Michael Beckstette. Structator : fast index-based search for RNA sequence-structure patterns. *BMC bioinformatics*, 12 :214, 2011.

[MKPW13] Georgi K. Marinov, Anshul Kundaje, Peter J. Park, and Barbara J. Wold. Large-Scale Quality Analysis of Published ChIP-seq Data. *G3 : Genes|Genomes|Genetics*, 4(2) :209–223, December 2013.

[MMB+15] Frédéric Mahé, Jordan Mayor, John Bunge, Jingyun Chi, Tobias Siemensmeyer, Thorsten Stoeck, Benjamin Wahl, Tobias Paprotka, Sabine Filker, and Micah Dunthorn. Comparing High-throughput Platforms for Sequencing the V4 Region of SSU-rDNA in Environmental Microbial Eukaryotic Diversity Surveys. *Journal of Eukaryotic Microbiology*, 62(3) :338–345, May 2015.

[MMYG06] Kimihiko Matsusue, Aya Miyoshi, Shigeru Yamano, and Frank J. Gonzalez. Ligand-activated PPAR B efficiently represses the induction of LXR-dependent promoter activity through competition with RXR. *Molecular and cellular endocrinology*, 256(1-2) :23–33, August 2006.

[MRDS+15] Alejandra Medina-Rivera, Matthieu Defrance, Olivier Sand, Carl Herrmann, Jaime A. Castro-Mondragon, Jeremy Delerce, Sébastien Jaeger, Christophe Blanchet, Pierre Vincens, Christophe Caron, Daniel M. Staines, Bruno Contreras-Moreira, Marie Artufel, Lucie Charbonnier-Khamvongsa, Céline Hernandez, Denis Thieffry, Morgane Thomas-Chollier, and Jacques van Helden. RSAT 2015 : Regulatory Sequence Analysis Tools. *Nucleic Acids Research*, page gkv362, April 2015.

[MRQ+15] Frédéric Mahé, Torbjørn Rognes, Christopher Quince, Colomban de Vargas, and Micah Dunthorn. Swarm v2 : highly-scalable and high-resolution amplicon clustering. *PeerJ*, 3 :e1420, 2015.

[MTHH+12] Simon A. J. Messing, Bao Ton-Hoang, Alison B. Hickman, Andrew J. McCubbin, Graham F. Peaslee, Rodolfo Ghirlando, Michael Chandler, and Fred Dyda. The processing of repetitive extragenic palindromes : the structure of a repetitive extragenic palindrome bound to its associated nuclease. *Nucleic Acids Research*, 40(19) :9964–9979, October 2012.

[Nav01] Gonzalo Navarro. NR-grep : a fast and flexible pattern-matching tool. *Software : Practice and Experience*, 31(13) :1265–1312, November 2001.

[NDR+05] Jacques Nicolas, Patrick Durand, Grégory Ranchy, Sébastien Tempel, and Anne-Sophie Valin. Suffix-tree analyser (STAN) : looking for nucleotidic and peptidic patterns in chromosomes. *Bioinformatics*, 21(24) :4408–4410, December 2005.

[NKE09] Eric P. Nawrocki, Diana L. Kolbe, and Sean R. Eddy. Infernal 1.0 : inference of RNA alignments. *Bioinformatics*, page btp157, March 2009.

[NUK+09] Mitsuhide Noshiro, Emiko Usui, Takeshi Kawamoto, Fuyuki Sato, Ayumu Nakashima, Taichi Ueshima, Kiyomasa Honda, Katsumi Fujimoto, Sato Honma, Ken-ichi Honma, Makoto Makishima, and Yukio Kato. Liver X receptors (LXRalpha and LXRbeta) are potent regulators for hepatic Dec1 expression. *Genes to Cells : Devoted to Molecular & Cellular Mechanisms*, 14(1) :29–40, January 2009.

[NW70] Saul B. Needleman and Christian D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3) :443–453, March 1970.

[PBAB15] Benoît Piégu, Solenne Bire, Peter Arensburger, and Yves Bigot. A survey of transposable element classification systems–a call for a fundamental update to meet the challenge of their diversity and complexity. *Molecular Phylogenetics and Evolution*, 86 :90–109, May 2015.

[PHKD06] Kristin Potter, Hans Hagen, Andreas Kerren, and Peter Dannenmann. Methods for Presenting Statistical Information : The Box Plot. *Visualization of Large and Unstructured Data Sets*, S-4 :97–106, 2006.

[PL88] W. R. Pearson and D. J. Lipman. Improved tools for biological sequence comparison. *Proceedings of the National Academy of Sciences of the United States of America*, 85(8) :2444–2448, April 1988.

[PLD00] Graziano Pesole, Sabino Liuni, and Mark D'Souza. PatSearch : a pattern matcher software that finds functional elements in nucleotide and protein sequences and assesses their statistical significance. *Bioinformatics*, 16(5) :439–450, January 2000.

[PLS01] Mihaela Pertea, Xiaoying Lin, and Steven L. Salzberg. GeneSplicer : a new computational method for splice site prediction. *Nucleic Acids Research*, 29(5) :1185–1190, March 2001.

[PP97] Prof. Charles Grinstead and Prof. Laurie Snell. *Introduction to Probability*. American Mathematical Society, 1997.

[Pro11] Nick J. Proudfoot. Ending the message : poly(A) signals then and now. *Genes & Development*, 25(17) :1770–1782, September 2011.

[PWSD+12] Petri Pehkonen, Lynn Welter-Stahl, Janine Diwo, Jussi Ryynänen, Anke Wienecke-Baldacchino, Sami Heikkinen, Eckardt Treuter, Knut R. Steffensen, and Carsten Carlberg. Genome-wide landscape of liver X receptor chromatin binding and gene regulation in human macrophages. *BMC Genomics*, 13 :50, 2012.

[RB12] Antoine Rocheteau and Catherine Belleannée. Recherche d'éléments structurés dans les génomes par modèles logiques. report, June 2012.

[RBW06] Francesca Rossi, Peter van Beek, and Toby Walsh. *Handbook of Constraint Programming*. Elsevier, August 2006. Google-Books-ID : Kjap9ZWcKOoC.

[RLB00] P. Rice, I. Longden, and A. Bleasby. EMBOSS : the European Molecular Biology Open Software Suite. *Trends in genetics : TIG*, 16(6) :276–277, June 2000.

[RLH+91] P Russo, W Z Li, D M Hampsey, K S Zaret, and F Sherman. Distinct cis-acting signals enhance 3' endpoint formation of CYC1 mRNA in the yeast Saccharomyces cerevisiae. *The EMBO Journal*, 10(3) :563–571, March 1991.

[ROL+07] Yaritzabel Roman, Masahiko Oshige, Young-Ju Lee, Kristie Goodwin, Millie M. Georgiadis, Robert A. Hromas, and Suk-Hee Lee. Biochemical characterization of a SET and transposase fusion protein, Metnase (SETMAR) for its DNA binding and DNA cleavage activity. *Biochemistry*, 46(40) :11369–11376, October 2007.

[RSG07] Jens Reeder, Peter Steffen, and Robert Giegerich. pknotsRG : RNA pseudoknot folding including near-optimal structures and sliding windows. *Nucleic Acids Research*, 35(Web Server issue) :W320–W324, July 2007.

[SBN+10] Thorsten Stoeck, David Bass, Markus Nebel, Richard Christen, Meredith D. M. Jones, Hans-Werner Breiner, and Thomas A. Richards. Multiple marker parallel tag environmental DNA sequencing reveals a highly complex eukaryotic community in marine anoxic water. *Molecular Ecology*, 19 Suppl 1 :21–31, March 2010.

[SCH+02] Christian J. A. Sigrist, Lorenzo Cerutti, Nicolas Hulo, Alexandre Gattiker, Laurent Falquet, Marco Pagni, Amos Bairoch, and Philipp Bucher. PROSITE : a documented database using patterns and profiles as motif descriptors. *Briefings in Bioinformatics*, 3(3) :265–274, September 2002.

[SD93] David Searls and SHAN DONG. *A Syntactic Pattern Recognition System For DNA Sequences*. 1993.

[Sea88] David Searls. Representing genetic information with formal grammars. *Proceedings of the 7th National Conference on Artificial Intelligence*, pages 386–291, 1988.

[Sea95] David B. Searls. String variable grammar : A logic grammar formalism for the biological language of DNA. *The Journal of Logic Programming*, 24(1) :73–102, July 1995.

[Sea97] David B. Searls. Linguistic approaches to biological sequences. *Computer applications in the biosciences : CABIOS*, 13(4) :333–344, January 1997.

[Sea02] David B. Searls. The language of genes. *Nature*, 420(6912) :211–217, November 2002.

[SO97] J. Schug and G.C. Overton. TESS : Transcription Element Search Software. *WWW Technical Report CBIL-TR-1997-1001-v0.0, of the Computational Biology and Informatics Laboratory*, 1997.

[SR96] A. F. Smit and A. D. Riggs. Tiggers and DNA transposon fossils in the human genome. *Proceedings of the National Academy of Sciences of the United States of America*, 93(4) :1443–1448, February 1996.

[SW81] T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147(1) :195–197, March 1981.

[TBP+14] Leho Tedersoo, Mohammad Bahram, Sergei Põlme, Urmas Kõljalg, Nourou S. Yorou, Ravi Wijesundera, Luis Villarreal Ruiz, Aída M. Vasco-Palacios, Pham Quang Thu, Ave Suija, Matthew E. Smith, Cathy Sharp, Erki Saluveer, Alessandro Saitta, Miguel Rosas, Taavi Riit, David Ratkowsky, Karin Pritsch, Kadri Põldmaa, Meike Piepenbring, Cherdchai Phosri, Marko Peterson, Kaarin Parts, Kadri Pärtel, Eveli Otsing, Eduardo Nouhra, André L. Njouonkou, R. Henrik Nilsson, Luis N. Morgado, Jordan Mayor, Tom W. May, Luiza Majuakim, D. Jean Lodge, Su See Lee, Karl-Henrik Larsson, Petr Kohout, Kentaro Hosaka, Indrek Hiiesalu, Terry W. Henkel, Helery Harend, Liang-dong Guo, Alina Greslebin, Gwen Grelet, Jozsef Geml, Genevieve Gates, William Dunstan, Chris

Dunk, Rein Drenkhan, John Dearnaley, André De Kesel, Tan Dang, Xin Chen, Franz Buegger, Francis Q. Brearley, Gregory Bonito, Sten Anslan, Sandra Abell, and Kessy Abarenkov. Global diversity and geography of soil fungi. *Science*, 346(6213) :1256688, November 2014.

[TGC02] Maja Tarailo-Graovac and Nansheng Chen. Using RepeatMasker to Identify Repetitive Elements in Genomic Sequences. In *Current Protocols in Bioinformatics*. John Wiley & Sons, Inc., 2002.

[TGP+09] Bogdan Tokovenko, Rostyslav Golda, Oleksiy Protas, Maria Obolenskaya, and Anna El'skaya. COTRASIF : conservation-aided transcription-factor-binding site finder. *Nucleic Acids Research*, 37(7) :e49, April 2009.

[THSQ+12a] Bao Ton-Hoang, Patricia Siguier, Yves Quentin, Séverine Onillon, Brigitte Marty, Gwennaele Fichant, and Mick Chandler. Structuring the bacterial genome : Y1-transposases associated with REP-BIME sequences. *Nucleic Acids Research*, 40(8) :3596–3609, April 2012.

[THSQ+12b] Bao Ton-Hoang, Patricia Siguier, Yves Quentin, Séverine Onillon, Brigitte Marty, Gwennaele Fichant, and Mick Chandler. Structuring the bacterial genome : Y1-transposases associated with REP-BIME sequences†. *Nucleic Acids Research*, 40(8) :3596–3609, April 2012.

[TP06a] Raquel Tobes and Eduardo Pareja. Bacterial repetitive extragenic palindromic sequences are DNA targets for Insertion Sequence elements. *BMC Genomics*, 7 :62, March 2006.

[TP06b] Raquel Tobes and Eduardo Pareja. Bacterial repetitive extragenic palindromic sequences are DNA targets for Insertion Sequence elements. *BMC Genomics*, 7 :62, March 2006.

[TR05] Raquel Tobes and Juan-Luis Ramos. REP code : defining bacterial identity in extragenic space. *Environmental Microbiology*, 7(2) :225–228, February 2005.

[TRG08] Corinna Theis, Jens Reeder, and Robert Giegerich. KnotInFrame : prediction of -1 ribosomal frameshift events. *Nucleic Acids Research*, 36(18) :6013–6020, October 2008.

[TTCDvH08] Jean-Valery Turatsinze, Morgane Thomas-Chollier, Matthieu Defrance, and Jacques van Helden. Using RSAT to scan genome sequences for transcription factor binding sites and cis-regulatory modules. *Nature Protocols*, 3(10) :1578–1588, 2008.

[TTV10] Laurie Tonon, Hélène Touzet, and Jean-Stéphane Varré. TFM-Explorer : mining cis-regulatory regions in genomes. *Nucleic Acids Research*, 38(suppl 2) :W286–W292, January 2010.

[vHACV98] J. van Helden, B. André, and J. Collado-Vides. Extracting regulatory sites from the upstream region of yeast genes by computational analysis of oligonucleotide frequencies. *Journal of Molecular Biology*, 281(5) :827–842, September 1998.

[vHdOPO00] J. van Helden, M. del Olmo, and J. E. Pérez-Ortín. Statistical analysis of yeast genomic downstream sequences reveals putative polyadenylation signals. *Nucleic Acids Research*, 28(4) :1000–1010, February 2000.

[vHRCV00] J. van Helden, A. F. Rios, and J. Collado-Vides. Discovering regulatory elements in non-coding sequences by analysis of spaced dyads. *Nucleic Acids Research*, 28(8) :1808–1818, April 2000.

[VSUV+09] Albert J. Vilella, Jessica Severin, Abel Ureta-Vidal, Li Heng, Richard Durbin, and Ewan Birney. EnsemblCompara GeneTrees : Complete, duplication-aware phylogenetic trees in vertebrates. *Genome Research*, 19(2) :327–335, February 2009.

[WE13] Travis J. Wheeler and Sean R. Eddy. nhmmer : DNA homology search with profile HMMs. *Bioinformatics*, 29(19) :2487–2489, January 2013.

[Wol06] Pierre Wolper. *Introduction à la Calculabilité*. Dunod, 2006.

[WRS+08] Yongjun Wang, Pamela M. Rogers, Chen Su, Gabor Varga, Keith R. Stayrook, and Thomas P. Burris. Regulation of cholesterologenesis by the oxysterol receptor, LXRalpha. *The Journal of Biological Chemistry*, 283(39) :26332–26339, September 2008.

[WS14] Somsakul Pop Wongpalee and Shalini Sharma. The pre-mRNA splicing reaction. *Methods in Molecular Biology (Clifton, N.J.)*, 1126 :3–12, 2014.

[WSTH+13] Mathias Weyder, Patricia Siguier, Bao Ton-Hoang, Mick Chandler, Gwennaele Fichant, and Yves Quentin. Evolution of repeated extragenic palindromic sequences. In *Journées Ouvertes en Biologie, Informatique et Mathématiques (JOBIM)*, Jobim-2013, Toulouse, France, 2013. Société Française de Bioinformatique (SFBI).

[YAA+16] Andrew Yates, Wasiu Akanni, M. Ridwan Amode, Daniel Barrell, Konstantinos Billis, Denise Carvalho-Silva, Carla Cummins, Peter Clapham, Stephen Fitzgerald, Laurent Gil, Carlos García Girón, Leo Gordon, Thibaut Hourlier, Sarah E. Hunt, Sophie H. Janacek, Nathan Johnson, Thomas Juettemann, Stephen Keenan, Ilias Lavidas, Fergal J. Martin, Thomas Maurel, William McLaren, Daniel N. Murphy, Rishi Nag, Michael Nuhn, Anne Parker, Mateus Patricio, Miguel Pignatelli, Matthew Rahtz, Harpreet Singh Riat, Daniel Sheppard, Kieron Taylor, Anja Thormann, Alessandro Vullo, Steven P. Wilder, Amonida Zadissa, Ewan Birney, Jennifer Harrow, Matthieu Muffato, Emily Perry, Magali Ruffier, Giulietta Spudich, Stephen J. Trevanion, Fiona Cunningham, Bronwen L. Aken, Daniel R. Zerbino, and Paul Flicek. Ensembl 2016. *Nucleic Acids Research*, 44(D1) :D710–D716, April 2016.

[Zha98] M. Q. Zhang. Statistical Features of Human Exons and Their Flanking Regions. *Human Molecular Genetics*, 7(5) :919–932, January 1998.

[ZSSW86] D. Zarkower, P. Stephenson, M. Sheets, and M. Wickens. The AAUAAA sequence is required both for cleavage and for polyadenylation of simian virus 40 pre-mRNA in vitro. *Molecular and Cellular Biology*, 6(7) :2317–2323, July 1986.

ANNEXE A

Annexes

A.1 Logol : détails du langage

Voici quelques éléments supplémentaires du langage Logol. Davantage de détails sont accessibles dans l'article présentant Logol (*"Logol : Expressive Pattern Matching in sequences. Application to Ribosomal Frameshift Modeling"*, [BSN14]) ainsi que dans la documentation présente sur le site web à l'adresse :
http ://logol.genouest.org/web/app.php/logol

Les grammaires Logol permettent de décrire différents types de contraintes pour un modèle, avec les opérateurs suivants :

Contraintes de séquence :

- Contrainte de taille #[tailleMinimum,TailleMaximum]
- Contrainte de début de position @[DebutMinimum,DebutMaximum]
- Contrainte de fin de position @@[FinMinimum,FinMaximum]
- Contrainte de stockage _NomVariable, qui permet de stocker l'instance exacte du match obtenu. Un nom de variable est toujours constitué de majuscules et doit se terminer par un chiffre.

Contraintes de structure :

- Contrainte de substitutions $[NombreMinimum,NombreMaximum]
- Contrainte de taux de substitutions "%" $:TauxMinimum
- Contrainte d'insertion/délétion $$[NombreMinimum,NombreMaximum]
- Contrainte de taux d'insertion/délétion "%" $$:TauxMinimum
- Contrainte de taux de nucléotides "%" chaîne :TauxMinimum, où chaîne est la concaténation des nucléotides ciblés (ex : `"%" "gc":50`)

Ces différentes contraintes s'appliquent à un élément du modèle de la façon suivante :

```
ElementDeModèle:{contrainteSeq1,...,contrainteSeqN}:{contrainteStr1,...,contrainteStrK}
```

Il est également possible de définir des vues, c'est-à-dire des regroupements d'éléments entre parenthèses. Par exemple : ("aaa","atg") constitue une vue regroupant la recherche d'"aaa" suivi d'"atg". Les contraintes qui s'appliquent à une vue forment une limite maximum que la somme des contraintes des éléments individuels ne doit pas dépasser.

Soit par exemple la grammaire : ("aaa" :$[0,2],"atg" :$[0,1]) :$[0,2]

"aaa" et "atg" sont regroupés au sein d'une vue (symbolisée par les parenthèses). Individuellement, "aaa" :$[0,2] peut accepter jusqu'à 2 substitutions tandis que "atg" :$[0,1]) n'en

accepte qu'une maximum. Néanmoins, la vue n'accepte elle-même que deux substitutions maximum (Vue) :$[0,2], ce qui signifie que la grammaire n'autorisera que les solutions où "aaa" et "atg" ne totalisent pas à eux deux plus de deux substitutions, peu importe la façon dont sont réparties ces substitutions entre les éléments de la vue (par exemple, la vue acceptera les solutions tel que "$a$atg" et "$aaat$" mais pas "$$aat$").

À ces différentes contraintes s'ajoutent les opérateurs :
- Inversion de l'élément `-Element1`. La grammaire recherche l'élément en sens indirect.
- Négation de l'élément `!Element1`. La grammaire recherche "tout sauf l'élément1".
- Insertion de l'instance exacte d'une variable stockée en amont de la grammaire ?NomVariable. Un nom de variable est toujours constitué de majuscules et doit se terminer par un chiffre.
- Choix multiple (Element1|Element2|...|ElementN)
- Répétition non-chevauchante repeat(Element, [IntervalleMinimum,IntervalleMaximum]) +[NbRépétitionMinimum,NbRépétitionMaximum]
- Répétition potentiellement chevauchante repeat(Element ; [IntervalleMinimum,IntervalleMaximum]) +[NbRépétitionMinimum,NbRépétitionMaximum]
- Impose l'arrêt de la recherche à la première solution obtenue HALT1. HALT1 se positionne à la fin d'une ligne de la grammaire.

Logol permet par ailleurs de lier différentes grammaires entre elles :
- Analyses successives `mod1(). mod2()==*>SEQ1`. La grammaire ne recherche `mod2()` qu'à la condition que le `mod1()` ait abouti à au moins un résultat valide.
- Transfert de variable `mod1(VAR1),mod2(VAR1)`. Une variable `VAR1` est définie dans le `mod1()`, son contenu est transféré au `mod2()` pour pouvoir y être utilisé.

Enfin, Logol permet de définir des contraintes au niveau global de la grammaire :
- Les morphismes sont définis dans la partie def{} de la grammaire. Ils décrivent les règles de transition d'un nucléotide vers un autre lorsqu'on utilise ce morphisme. Par exemple, le morphisme "wc" (pour Watson-Crick) défini la transition "complément", ainsi, "wc" "atg" aboutira à la recherche de la séquence complémentaire "tac". Ce morphisme est présent de base dans Logol et n'a pas besoin d'être décrit systématiquement dans les grammaires qui l'emploient, mais il est possible de définir d'autres morphismes.
- Les contrôles permettent de définir des contraintes qui s'appliquent à différentes variables sauvegardées. Un contrôle est défini dans une partie `controls:{}`, positionnée juste après le `def{}`. Par exemple, un contrôle vérifiant que le pourcentage en nucléotides "c" est supérieur ou égal à 30% lorsqu'on considère globalement les variables `VAR1` et `VAR2` du `mod1()` s'écrit de façon suivante :

```
controls:{
% "c"[mod1.VAR1,mod1.VAR2]>=30
}
```

A.1.1 Exemple de lancement de Logol

L'analyseur Logol utilise deux lignes de commandes différentes en fonction de la nature du fichier fasta. La commande LogolExec est à utiliser si le fichier fasta ne contient qu'une seule séquence. La sortie sera renvoyée sous la forme d'un fichier XML contenant les résultats de l'analyse sur cette séquence. La commande MultiLogolExec est à utiliser si le fichier fasta contenant plusieurs séquence. La sortie sera renvoyée sous la forme d'une archive .zip rassemblant l'ensemble des fichiers XML de résultat, un pour chaque séquence.

Ligne de commande :

```
LogolExec -s FastaFilePath -g GrammarFilePath -output OutputXMLpath -conf ConfFilePath
LogolMultiExec.sh -s FastaFilePath -g GrammarFilePath -out OutputZIPpath -conf
ConfFilePath
```

A.1.2 Exemple de fichier de configuration pour l'argument ConfFilePath

```
# Minimum size to use to split a file to parallelize treatments
minSplitSize=200000
# Maximum size to display a solution
#maxResultSize=20000
maxResultSize=0
# Maximum size of a solution
#maxMatchSize=30000
maxMatchSize=0
# Temporary directory used for the analysis. Should be local to the node
workingDir=/home/genouest/dyliss/USER/FOLDER
#workingDir=/tmp
# Directory where to place the results. In case of cluster usage, result must be a shar
between nodes
dir.result=/home/genouest/dyliss/USER/FOLDER
# Maximum length of a spacer when looking forward for a match
#maxSpacerLength=10000
maxSpacerLength=0
# Maximum length of a variable in a match
#maxLength=1000
maxLength=0
# Minimum length of a variable in a match
minLength=2
# Default strategy to use, 1 must be keep by default
parentStrategy=1
# Number of processor on computer running the analysis
nbProcessor=1
# Max number of job per sequence
nbJobs=10
# Default number to limit number of results
maxSolutions=10000
# Minimum size of tree index
#minTreeIndex=4
minTreeIndex=1
# Path to logol software
#installPath=C:\\Web\\eclipse\\workspace\\LogolMatch
# Host where is smtp server
smtp.host=genogrid-data
# Mail user for smtp host
mail.user=bioinfo
mail.from=do.not.reply@irisa.fr
# DRM queue command if a specific queue is to be used
# Example for SGE: drm.queue= -q long
```

```
drm.queue= -q sgi64g.q
# Use Vmatch [1] or internal tool [0] (default)
suffix.tool = 1
# Vmatch path, if used and not in path
suffix.path=/local/vmatch/vmatch.distribution
```

A.2 Logol : évolution du temps de calcul en fonction de la complexité du modèle

Le temps de calcul est le point critique de l'analyseur Logol. Pour donner une estimation de ce critère, nous avons mesuré l'évolution du temps de calcul sur des modèles de complexité croissante. Ces modèles ont été testés sur les 25 805 upstreams de taille 10 000 du génome humain (cf chapitre 3).

L'ensemble des recherches a été fait sur un nœud du cluster de Genouest doté de 24 cœurs.

Identifiant	Grammaire Logol	Temps
1	`"tgaccggcagtaaccc"`	8h 18min
2	`"tgaccggcagtaaccc":{$[0,1]}`	8h 26min
3	`"tgaccggcagtaaccc":{$[0,2]}`	7h 43min
4	`"tgaccg",GAP4:{#[4,4]},"taaccc"`	7h 53min
5	`("tgaccg":{$[0,1]},GAP4:{#[4,4]},"taaccc":{$[0,1]}):{$[0,1]}`	7h 46min
6	`("tgaccg":{$[0,2]},GAP4:{#[4,4]},"taaccc":{$[0,2]}):{$[0,2]}`	8h 17min
7	`"tgaccg":{_PART1},GAP4:{#[4,4]},?PART1`	8h 42min
8	`"tgaccg":{_PART1},GAP4:{#[4,4]},?PART1:{$[0,1]}`	8h 20min
9	`"tgaccg":{_PART1},GAP4:{#[4,4]},?PART1:{$[0,2]}`	8h 28min
10	`"tgaccg":{_PART1}:{$[0,1]},GAP4:{#[4,4]},?PART1`	8h 33min
11	`"tgaccg":{_PART1}:{$[0,1]},GAP4:{#[4,4]},?PART1:{$[0,1]}`	8h
12	`"tgaccg":{_PART1}:{$[0,1]},GAP4:{#[4,4]},?PART1:{$[0,2]}`	8h 6min
13	`HEXA1:{#[6,6],_PART1},GAP4:{#[4,4]},?PART1`	13h 7min
14	`HEXA1:{#[6,6],_PART1},GAP4:{#[4,4]},?PART1:{$[0,1]}`	13h 11min
15	`HEXA1:{#[6,6],_PART1},GAP4:{#[4,4]},?PART1:{$[0,2]}`	15h 51min

Avec :

- **1** : La référence stricte du TFBS LXRα pour le gène FASN chez l'humain
- **2** : Tous les variants de la grammaire **1** à 1 substitution près
- **3** : Tous les variants de la grammaire **1** à 2 substitution près
- **4** : La référence du TFBS LXRα, avec les 4 nucléotides centraux indéfinis
- **5** : Tous les variants de la grammaire **4** à 1 substitution près. Pour cela, on autorise jusqu'à 1 substitution de part et d'autre du gap, mais en n'autorisant pas plus d'1 substitution dans l'ensemble de la vue
- **6** : Tous les variants de la grammaire **4** à 2 substitution près
- **7** : Un motif DR4 dont la répétition est constituée de la première partie du motif du TFBS LXRα pour le gène FASN chez l'humain
- **8** : Tous les variants de la grammaire **7** à 1 substitution près dans la répétition
- **9** : Tous les variants de la grammaire **7** à 2 substitutions près dans la répétition
- **10** : Un motif DR4 dont la répétition est constituée d'une variante à 1 substitution près de la première partie du motif du TFBS LXRα pour le gène FASN chez l'humain
- **11** : Tous les variants de la grammaire **10** à 1 substitution près dans la répétition
- **12** : Tous les variants de la grammaire **11** à 2 substitutions près dans la répétition
- **13** : Un motif DR4 dont le motif de base de taille 6 n'est pas défini
- **14** : Tous les variants de la grammaire **13** à 1 substitution près dans la répétition
- **15** : Tous les variants de la grammaire **13** à 2 substitutions près dans la répétition

Tout d'abord, il est important de noter que l'analyseur Logol emploie deux stratégies de recherche différentes selon les grammaires employées. Pour les grammaires 1 à 12, la première partie du motif est fixe, ce qui permet à l'analyseur d'effectuer une recherche grâce à l'index pour accélérer le traitement (tableau de suffixe via Vmatch ou arbre de suffixe via Cassiopée). Dans le cas des grammaires 13 à 15, l'absence de motif défini oblige l'analyseur a parser successivement toutes les positions de la séquence pour vérifier la présence d'un match. Par ailleurs, pour des raisons vraisemblablement liés aux contraintes techniques du cluster de calcul, les temps de calcul possèdent des aléas de l'ordre de quelques dizaines de minutes, indépendant de l'analyseur lui-même. Une même recherche refaite dans les mêmes conditions n'aboutit pas exactement au même temps de calcul. Ainsi, les temps des grammaires 1 à 12 peuvent être considérés comme similaires.

Dans le cas des grammaires 1 à 12, on remarque que le temps est globalement identique quelque soit la grammaire, pour une durée moyenne de 8h. Étrangement, rechercher un mot fixe (grammaire 1) prend autant de temps que rechercher une répétition approximative d'un mot approché (grammaire 12) alors que cela semble de prime abord plus simple à mettre en œuvre. Bien que le temps de base pour la grammaire la plus élevé soit énorme, il s'avère que tant que le motif est partiellement défini, aucune des complexifications de la grammaire testées n'entraîne réellement un surcoût du temps de calcul. Par extension, rechercher ou non les variants à quelques substitutions près d'un modèle n'a pas d'impact sur le temps de calcul.

Dans le cas des grammaires 13 à 15, on recherche des grammaires qui ne sont définies que par des contraintes, sans aucune information de contenu. Ceci oblige l'analyseur à adopter une autre stratégie de recherche qui ne permet pas de raccourci dans le parcours de la séquence analysée. Les temps de recherche augmentent sensiblement pour les grammaires 13 et 14 par rapport aux grammaires précédentes, tout en restant proches l'un de l'autre. Ce n'est étonnamment pas le cas de la grammaire 15, pour laquelle le temps de calcul atteint le double de celui des grammaires 1 à 12.

En conclusion, on peut noter que sur ce cas de figure, l'analyseur présente un "coût de base" important qui pénalise le temps de calcul, même pour le cas le plus élémentaire possible. Néanmoins, il est suffisamment optimisé pour pouvoir prendre en charge la majorité des cas complexes qu'on lui soumet sans entraîner de surcout de temps de calcul.

A.3 Différents types de matrice de score

Cette annexe vise à présenter de façon succincte et simplifiée les différents types de matrices existantes. Pour plus de précision, il est possible de se référé à la thèse d'Aude Liefooghe "Matrice score-position, algorithmes et propriétés" [Lie08].

L'une des façons les plus courantes de modéliser un motif en biologie est l'utilisation des matrices. Les matrices synthétisent l'ensemble des données de référence d'un modèle sous la forme d'un tableau où chaque colonne représente une position et chaque ligne représente un nucléotide. La modélisation d'un motif sous la forme d'une matrice présuppose deux contraintes fortes : les références ne contiennent pas d'évènements d'indels et chaque position est indépendante des autres.

Les matrices ci-dessous sont construites à partir des références du motif LXRα (cf 3).

La matrice de comptage

La matrice de comptage, qui indique le nombre d'occurrence de chaque nucléotide pour chaque position dans les séquences de références. Elle présente l'avantage de conserver l'information du nombre de séquences de référence.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
A	0	0	11	2	0	1	3,25	3,25	3,25	3,25	0	9	13	0	0	2
C	3	0	2	11	13	0	3,25	3,25	3,25	3,25	0	0	0	12	13	6
G	0	13	0	0	0	8	3,25	3,25	3,25	3,25	0	4	0	0	0	0
T	10	0	0	0	0	4	3,25	3,25	3,25	3,25	13	0	0	1	0	5

FIGURE A.1 – Exemple de matrice de comptage pour le motif LXRα, établi à partir de 13 références. Les 4 nucléotides centraux (7-8-9-10) n'ont pas d'incidence sur le motif et ont reçu une répartition équitable.

La matrice de fréquence

La matrice de fréquence, qui indique la fréquence de chaque nucléotide pour chaque position dans les séquences de références.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
A	0	0	0,85	0,15	0	0,08	0,25	0,25	0,25	0,25	0	0,69	1	0	0	0,15
C	0,23	0	0,15	0,85	1	0	0,25	0,25	0,25	0,25	0	0	0	0,92	1	0,46
G	0	1	0	0	0	0,62	0,25	0,25	0,25	0,25	0	0,31	0	0	0	0
T	0,77	0	0	0	0	0,31	0,25	0,25	0,25	0,25	1	0	0	0,08	0	0,38

FIGURE A.2 – Exemple de matrice de fréquence pour le motif LXRα, établi à partir de 13 références

La matrice de fréquence corrigée

La matrice de fréquence corrigée, qui ajoute à la matrice de comptage un certain nombre de valeurs (le pseudo-compte) qui suivent une fréquence particulière de répartition entre les 4 nucléotides. Concrètement, ce pseudo-compte mime l'ajout de séquences ne faisant pas partie des références et introduit de la souplesse dans le modèle : il n'existe plus de position avec une valeur de 0, il devient donc plus facile au modèle de reconnaître des motifs substitués.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
A	0,02	0,02	0,8	0,16	0	0,02	0,25	0,25	0,25	0,25	0,02	0,59	0,95	0,02	0,02	0,16
C	0,23	0,02	0,16	0,8	0,95	0,02	0,25	0,25	0,25	0,25	0,02	0,02	0,02	0,88	0,95	0,45
G	0,02	0,95	0,02	0,02	0,02	0,59	0,25	0,25	0,25	0,25	0,02	0,3	0,02	0,02	0,02	0,02
T	0,73	0,02	0,02	0,02	0,02	0,3	0,25	0,25	0,25	0,25	0,95	0,02	0,02	0,09	0,02	0,3

FIGURE A.3 – Exemple de matrice de fréquence corrigée avec un pseudo-compte de 1 et une fréquence équivalente pour les 4 nucléotides pour le motif LXRα. La correction revient à ajouter une 14ème référence avec une valeur de 0,25 entre chaque nucléotide pour chaque position

La matrice de fréquence relative corrigée

La matrice de fréquence relative corrigée, qui pondèrent les fréquences de la matrice de fréquence corrigée en fonction d'un contexte. Elle permet d'adapter le score aux particularités des séquences analysées.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
A	0,08	0,08	3,2	0,64	0,08	0,36	1,08	1,08	1,08	1,08	0,08	2,64	3,8	0,08	0,08	0,64
C	0,92	0,08	0,64	3,2	3,8	0,08	1,08	1,08	1,08	1,08	0,08	0,08	0,08	3,52	3,8	1,8
G	0,08	3,8	0,08	0,08	0,08	8	1,08	1,08	1,08	1,08	0,08	1,2	0,08	0,08	0,08	0,08
T	2,92	0,08	0,08	0,08	0,08	1,2	1,08	1,08	1,08	1,08	3,8	0,08	0,08	0,36	0,08	1,52

FIGURE A.4 – Exemple de matrice de fréquence corrigée relative pour le motif LXRα, à partir de la matrice de fréquence corrigée de la figure A.3 pour un contexte avec une fréquence de 0,25 pour chaque nucléotide.

La matrice d'entropie

La matrice d'entropie donne le contenu informationnel d'une matrice de fréquence corrigée. Une position très conservée dans les références apporte beaucoup d'informations, tandis qu'une position où les nucléotides sont équi-répartis n'apporte que peu d'informations. Les matrices d'entropie permettent une bonne mesure de la correspondance entre un motif et une séquence.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
A	-0,08	-0,08	-0,18	-0,29	-0,08	-0,22	-0.35	-0.35	-0.35	-0.35	-0,08	-0,27	-0,05	-0,08	-0,08	-0,29
C	-0,34	-0,08	-0,29	-0,18	-0,05	-0,08	-0.35	-0.35	-0.35	-0.35	-0,08	-0,08	-0,08	-0,11	-0,05	-0,36
G	-0,08	-0,05	-0,08	-0,08	-0,08	-0,31	-0.35	-0.35	-0.35	-0.35	-0,08	-0,36	-0,08	-0,08	-0,08	-0,08
T	-0,23	-0,08	-0,08	-0,08	-0,08	-0,36	-0.35	-0.35	-0.35	-0.35	-0,05	-0,08	-0,08	-0,22	-0,08	-0,37

FIGURE A.5 – Exemple de matrice d'entropie pour le motif LXRα, établi à partir de 13 références.

La matrice d'entropie relative pondérée

La matrice d'entropie relative pondère le contenu informationnelle par rapport au contexte nucléotidique d'un background.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
A	-0,05	-0,05	0,93	-0,07	-0,05	-0,09	0,02	0,02	0,02	0,02	-0,05	0,64	1,27	-0,05	-0,05	-0,07
C	-0,02	-0,05	-0,07	0,93	1,27	-0,05	0,02	0,02	0,02	0,02	-0,05	-0,05	-0,05	1,11	1,27	0,26
G	-0,05	1,27	-0,05	-0,05	-0,05	0,51	0,02	0,02	0,02	0,02	-0,05	0,05	-0,05	-0,05	-0,05	-0,05
T	0,78	-0,05	-0,05	-0,05	-0,05	0,05	0,02	0,02	0,02	0,02	1,27	-0,05	-0,05	-0,09	-0,05	0,16

FIGURE A.6 – Exemple de matrice d'entropie pour le motif LXRα, établi à partir de 13 références pour un contexte avec une fréquence de 0,25 pour chaque nucléotide.

La matrice score-position

La matrice score-position est la forme la plus aboutie, qui prend en compte à la fois le contenu informationnel du modèle mais aussi l'influence du contexte. Elle attribue des poids à chaque nucléotide pour chaque position, qui seront positif si le nucléotide apparait plus souvent qu'attendue dans les références et négatifs dans le cas inverse. La matrice ainsi obtenue fonctionne de manière additive : lors d'une analyse, la matrice est alignée contre chaque position de la séquence, et le score final est la somme des poids de chaque lettre alignée à chaque position (Hertz-Stomo99).

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
A	-2,53	-2,53	1,16	-0,45	-2,53	-1,02	0,08	0,08	0,08	0,08	-2,53	0,97	1,34	-2,53	-2,53	-0,45
C	-0,45	-2,53	-1,02	1,26	1,34	-2,53	0,08	0,08	0,08	0,08	-2,53	-2,53	-2,53	1,26	1,34	0,59
G	-2,53	1,34	-2,53	-2,53	-2,53	0,86	0,08	0,08	0,08	0,08	-2,53	0,18	-2,53	-2,53	-2,53	-2,53
T	-1,02	-2,53	-2,53	-2,53	-2,53	0,18	0,08	0,08	0,08	0,08	1,34	-2,53	-2,53	-1,02	-2,53	0,42

FIGURE A.7 – Exemple de matrice de score-position pour le motif LXRα

Notion de distance entre un match et une référence

A contrario des modèles grammaticaux, les matrices ne déterminent pas intrinsèquement si une séquence correspond ou non au modèle. Quelque soit la séquence rencontrée, une matrice aboutira toujours à un score, plus ou moins grand en fonction de l'adéquation de la séquence avec le consensus des références formant la matrice. C'est à l'utilisateur de décider quel est le seuil à partir duquel un hit est valide ou non.

A.4 Base de données Orthocis

La base de données Orthocis réunit quatre types d'informations différents [1].

A.4.1 Information d'orthologie des espèces

Pour les 63 espèces présentes dans la base de données Ensembl Compara [HMB+16], Orthocis recueille un ensemble de données phylogéniques qui permettent d'établir le treillis des gènes orthologues « one2one » (cf Figure A.8 pour un exemple du treillis pour 8 espèces).

L'expression « one2one » désigne les relations d'orthologies entre deux espèces pour lesquelles un gène n'est orthologue qu'à un seul autre. Pour des raisons techniques et pratiques, les relations « one2many » (un gène d'une espèce pointe plusieurs gènes d'une autre espèce) sont actuellement exclues de l'étude. Cela signifie plus précisément que les gènes qui n'ont aucun orthologue dans aucune autre espèce ou que les gènes paralogues (qui ont une orthologie intra-espèce, càd que le gène y est dupliqué) sont exclus de la base.

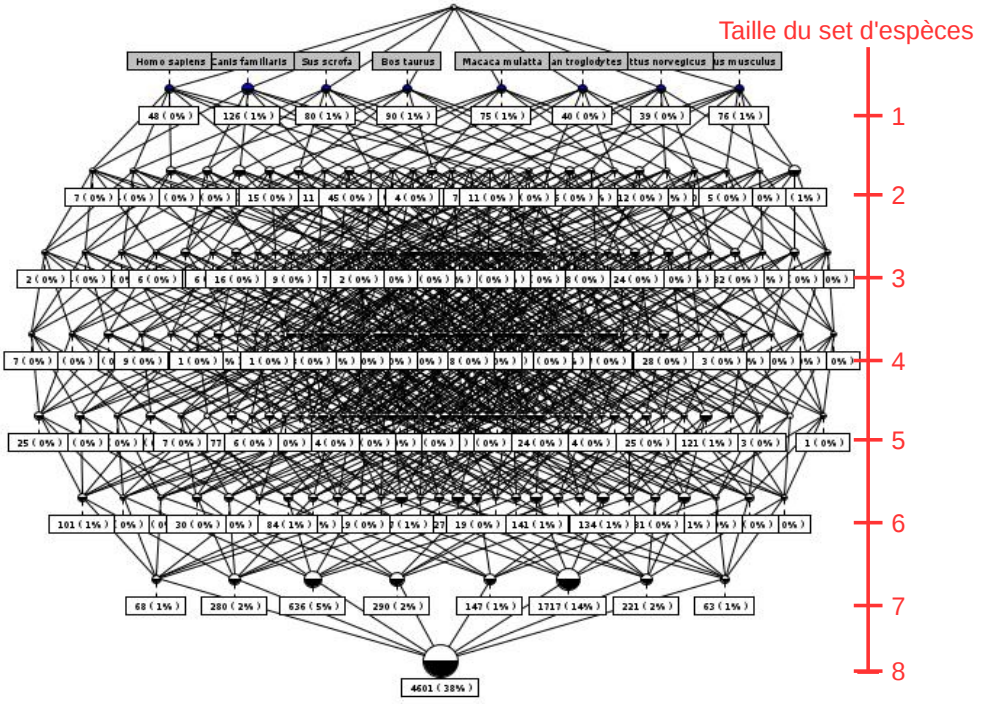


FIGURE A.8 – Treillis de gènes pour une sélection de 8 espèces.

Chaque ligne du treillis donne les différentes combinaisons possibles de la sélection pour un nombre d'espèces donné. Chaque case du treillis donne le nombre de gènes orthologues exclusivement partagés par le set d'espèces considéré. Par exemple, la première case en partant de la gauche pour 7 espèces indique qu'il existe 68 gènes orthologues spécifiquement partagés par l'humain, le chien, le cochon, la vache, le macaque, le chimpanzé et le rat mais pas par la souris. Les quantités sont cumulatives du bas vers le haut : les 4 601 gènes orthologues spécifiquement partagés par les 8 espèces s'ajoutent aux 68 gènes sus-mentionnés pour déterminer

1. Les travaux présentés ici datent de la première version d'Orthocis, basée sur la version 75 d'Ensembl et ne reflètent pas la version actuelle d'Orthocis qui utilisent une version plus récente d'Ensembl.

le nombre total de gènes orthologues partagés par l'humain, le chien, le cochon, la vache, le macaque, le chimpanzé et le rat.

A.4.2 Information génétique des espèces

Orthocis est en grande partie basée sur la base de données Ensembl [YAA+16], version 75, qui contient 63 espèces de vertébrés.

Pour chacune de ces espèces, à partir de la liste des gènes « one2one » précédemment établi, nous avons récupéré les 10 000 nucléotides upstreams de chaque gène (c-à-d les nucléotides positionnés avant le start du gène).

A.4.3 Information d'annotation des gènes

Orthocis intègre les annotations HGNC [GST+16] (HUGO (HUman Genome Organisation) Gene Nomenclature Committe) afin de faciliter l'analyse humaine des gènes qui peuvent être sélectionnés.

Bien que les annotations HGNC ne concernent que le génome humain, puisque nous avons a disposition les liens d'orthologie inter-espèces, nous avons répercuté ces annotations sur l'ensemble des gènes orthologues des autres espèces.

Toutefois, il existe des gènes présents dans la base qui n'ont pas d'orthologues chez l'humain. Tous les gènes ne disposent donc pas automatiquement d'une annotation HGNC.

A.4.4 Information des TFBS putatifs

Les sites de fixation de tous les facteurs de transcription présents dans la base sont recherchés sur l'ensemble des upstreams présents. Orthocis permet donc d'obtenir la liste de tous les gènes contenant une instance valide d'un TFBS. Pour chacune de ces instances, sa nature, sa position et éventuellement son score sont accessibles.

Dans le cadre de hits avec un score, seul le filtre par défaut a été employé pour rejeter les hits les moins probables. Aucun autre filtre n'a été fixé, de façon à conserver un maximum d'informations brutes. Ainsi, l'utilisateur de la base dispose de l'ensemble des données disponibles et peut librement fixer ses propres filtres pour raffiner les résultats.

A.4.5 Conséquence du filtre d'orthologie

Le filtre d'orthologie fonctionne de la façon suivante : pour qu'un gène soit valide, il faut qu'il possède un orthologue dans chacune des espèces sélectionnées pour le filtre. Cela entraîne deux conséquences qui peuvent affecter les résultats.

Influence de la conception

En premier lieu, la base de données Orthocis ne contient que les orthologues "one2one". Si une espèce possède des gènes paralogues (présents en plusieurs copies dans le génome d'une même espèce), ces gènes auront des relations "one2many" ou "many2many" avec les autres orthologues et n'auront pas été intégrés dans la base.

En conséquence, le filtre d'orthologie d'Orthocis peut rejeter un gène qui ne possède pas de gènes orthologues dans la base, alors qu'en réalité, il existe plusieurs gènes orthologues dans cette espèce.

L'intégration des gènes "one2many" est donc l'une des prochaines étapes d'amélioration de la base de données Orthocis. Cette intégration n'est cependant pas anodine, puisqu'elle pose

de nouvelles questions. Par exemple, si une espèce possède deux gènes paralogues et que l'un est positif pour un motif et l'autre négatif, satisfait-t-elle ou non au principe de conservation ?

Cette problématique demande donc davantage de travail pour comprendre toutes les subtilités qu'elle implique et l'intégrer correctement dans notre méthode de sélection.

Influence du set d'espèces

En second lieu, comme présenté dans la partie 3.2.1, le filtre d'orthologie ne valide que les cliques d'orthologie. Il suffit que l'une des espèces présentes ne puisse former une clique avec l'ensemble des autres pour invalider le gène considéré.

Deux conséquences peuvent en découler :

— Pour le gène considéré, deux espèces ont suffisamment divergés pour ne plus pouvoir former une clique. Bien que les gènes soient présents dans la base, bien qu'ils possèdent des hits valides, l'absence de clique invalide la sélection du gène.
— Une des espèces du set considéré ne possède pas d'orthologue pour le gène considéré, alors que le reste du set possède ce gène avec un hit valide. L'absence de clique invalide la sélection du gène.

Ces deux cas de figure expliquent pourquoi notre méthode ne parvient pas à retrouver la totalité de séquences de référence avec notre méthode. En effet, notre set d'espèces intègre un grand nombre d'espèces (7 au total) dont certaines sont très éloignées en terme d'évolution (ex : le poulet et l'homme). Ces deux facteurs compliquent d'autant l'obtention de gènes satisfaisant une clique d'orthologie parfaite. Mais inversement, employer des espèces trop peu nombreuses et/ou trop proche limite l'intérêt de vérifier la conservation au cours de l'évolution.

Déterminer l'impact du set d'espèces sur la sélection de gènes et déterminer la nécessité ou non d'être aussi strict sur la nécessité de former une clique parfaite d'orthologues sont donc deux chantiers du projet Orthocis qui nécessitent des études plus poussées.

A.4.6 Rôle respectif des filtres de Pvalue et d'homogénéité

L'intérêt du filtre d'homogénéité dépend grandement de la stringence du filtre de Pvalue. En effet, le filtre d'homogénéité a été mis en place pour déterminer si les hits au sein d'une clique de gènes orthologues vérifient un minimum de conservation. Or, si le filtre de Pvalue est trop élevé, il va naturellement sélectionner les hits les plus similaires au consensus, ce qui réduit alors l'intérêt de filtrer sur l'homogénéité des hits obtenus.

L'utilisation du filtre d'homogénéité n'a donc d'intérêt que si le filtre de Pvalue est un minimum relâché. Dans ce cas de figure, le filtre de Pvalue valide les hits en autorisant un certain degré de variabilité du consensus. Le filtre d'homogénéité assure alors que, tout en autorisant cette variabilité, les hits d'un clique de gènes orthologues restent relativement conservés entre eux.

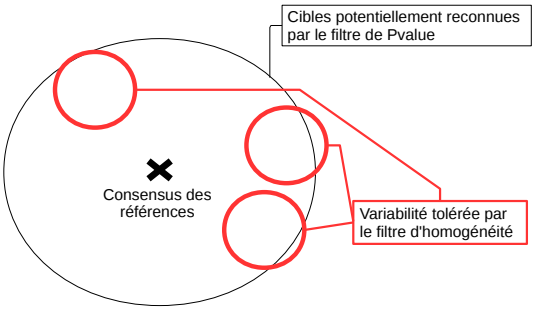


FIGURE A.9 – Complémentarité des filtres Pvalues et homogénéité : le filtre de Pvalue permet de rechercher plus ou moins largement autour du consensus des références, tandis que le filtre d'homogénéité vérifier que les hits validés ne soient pas éparpillés au sein du champs des séquences visées par le filtre de Pvalue

A.4.7 Perspective pour Orthocis

Les interactions avec différents membres de la communauté scientifiques à propos d'Orthocis ont permis de dégager plusieurs pistes supplémentaires de recherche qu'il serait intéressant d'exploiter.

Meilleur *design* des upstreams

Dans le cadre de cette étude, les upstreams des séquences font une taille de -10 000 à partir du début du gène tel que défini par Ensembl. Or, Ensembl détermine la taille d'un gène par la fusion de l'ensemble des transcrits possibles, du premier exon du transcrit le plus en amont au denier exon du transcrit le plus en aval. En conséquence, nos upstreams ne valent que pour un transcrit du gène : il nous manque la séquence entre ce "start" théorique du gène et les autres transcrits, qui peuvent contenir des sites de fixation des facteurs de transcription.

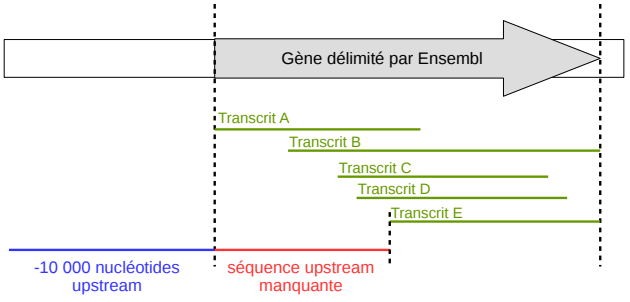


FIGURE A.10 – Définition de l'upstream en fonction du start du gène

C'est ce biais explique en partie pourquoi nous ne retrouvons pas l'ensemble de nos références. Le nouveau calcul des upstreams sera intégré à la prochaine mise à jour d'Orthocis.

Prise en compte des paralogues

Comme déjà indiqué, pour le moment, la méthode s'est restreinte aux gènes one-2-one, pour lesquels on est certain de la conservation. Elle exclut automatiquement les gènes provoquant des orthologies one-2-many, c'est-à-dire les gènes orthologues présents en plusieurs exemplaires dans un même organisme. L'intégration des gènes "one2many" constitue l'une des prochaines étapes d'amélioration de la base de données Orthocis. Elle pose néanmoins de nouvelles questions qui devront être analysées pour en estimer l'impact sur les résultats (Par exemple, si une espèce possède deux gènes paralogues et que l'un est positif pour un motif et l'autre négatif, comment la considère-t-on au vu du principe de conservation ?).

Prise en compte la co-occurence des TFBS

On sait qu'il existe des facteurs de transcription qui travaillent de façon complémentaire [Car98, CLEC95]. Pour le moment, Orthocis ne traite les TFBS que de façon individuelle, mais puisque toutes les informations d'une reconnaissance de motifs sont disponibles, il serait facile de passer à une recherche "multi-TFBS".

Utilisation de données biologiques

Enfin, une autre piste intéressante serait de permettre de coupler des résultats d'autres types d'analyse aux filtres d'Orthocis, tel que par exemple les données ChIP-Seq. Le ChIP-Seq (ChIP(CHromatin ImmunoPrecipitation)-sequencing) est une méthode utilisée pour analyser les interactions Protéines/ADN. Elle permet d'identifier les séquences où la protéine se fixe effectivement. L'utilisation de données ChIP-Seq pourrait donc aussi intervenir pour limiter la sélection de candidats.

A.5 Recherche d'amorces mutées en métagénomique : données complémentaires

A.5.1 Définition du pattern de mutation des amorces selon la technologie employée

Objectif : Déduire le pattern de mutation propre à chaque amorce, pour chaque technologie

Hypothèse : On suppose qu'il existe une différence dans les types de mutations de séquençages produites par 454 par rapport à l'Illumina. Dans cette optique, les amorces mutées ont été rassemblées (càd, les amorces identifiées par Logol mais aussi les fragments amonts et avals des amplicons repêchés, censés contenir les amorces) et analysé dans le but d'identifier quelles étaient les mutations qui les affectaient et quelles étaient celles qui étaient les plus récurrentes.

Méthode : L'identification des différents patterns a été fait en prenant le meilleur alignement de l'amorce muté contre l'une des quatre formes possibles de l'amorce de référence. Les différentes mutations ont été comptabilisées, en distinguant le cas des mutations aux extrémités des cas de mutations internes (en effet, ceux-ci font l'objet de spécifications précises de la part de l'expert). Les patterns similaires ont ensuite été regroupés pour quantifier leur présence au sein de l'ensemble.

Résultats et conclusions :

Mutation prépondérante pour V4F-454	**Mutation prépondérante pour V4F-Illumina**
78,24 % - Amorces Logol	87,03 % - Amorces Logol
dt 34.25% - 1 Délétion interne dt 30.7% - 1 Substitution dt 12.14% - 1 Insertion interne dt 1,15 % - Autres	dt 37.92% - 1 Substitution dt 34.99% - 1 Délétion interne dt 8,72% - 1 Insertion interne dt 5.4% - Autres
9,28 % - Pas d'amorces	0,82 % - Pas d'amorces
0 % - Amorces sans rapport	0,35 % - Amorces sans rapport
9,05 % - Patterns importants non-couverts	8,54 % - Patterns importants non-couverts
dt 5,96 % - Délétion à l'extrémité 5' dt 3,09 % - 2 Délétions internes	dt 7,09 % - Délétion à l'extrémité 5' dt 1,45 % - 2 Délétions internes
Total : 96,57 %	Total : 96,74 %

TABLE A.1 – Profil de mutations des amorces V4F (amorces regex, amorces Logol et amorces déduites par la repêche aux amplicons)

On remarque que les deux technologies possèdent le même ratio de délétion unique au sein de leurs amorces mutées. La technologie Illumina favorise davantage les substitutions au détriment des insertions, contrairement à la technologie 454. On remarque aussi que pour les deux technologies, les principales amorces loupées par Logol suivent les mêmes mutations : il faudrait permettre une délétion à l'extrémité 5' et permettre jusqu'à 2 délétions internes.

Enfin, en regroupant tout ces cas, on constate qu'on a récupéré la grande majorité des amorces mutés, les $\sim$3 % restant se subdivisant en une multitude de cas uniques.

Résultats et conclusions :

Mutation prépondérante pour V4R-454	**Mutation prépondérante pour V4R-Illumina**
94,36 % - Amorces Logol	93,42 % - Amorces Logol
dt 27,37% - 1 Insertion interne dt 26,76% - 1 Délétion interne dt 23,72% - Tronqué de 2 nucléotides dt 10,64 % - 1 Substitution dt 5,87% - Autres	dt 74,91% - 1 Substitution dt 7,01% - 2 Substitutions dt 6,2% - 1 Délétion interne dt 3,41% - 1 Insertion interne dt 1,89% - Autres
0 % - Pas d'amorces	0,06 % - Pas d'amorces
0,56 % - Amorces sans rapport	0,18 % - Amorces sans rapport
1,57 % - Patterns importants non-couverts	4,8 % - Patterns importants non-couverts
dt 0,81 % - 2 Insertions internes dt 0,76 % - 2 Délétions internes	dt 4,8 % - 2 Délétion internes
Total : 96,57 %	Total : 96,74 %

TABLE A.2 – Profil de mutations des amorces V4R (amorces regex, amorces Logol et amorces déduites par la repêche aux amplicons)

Contrairement à l'amorce V4F, les profils de mutations de V4R diffèrent nettement selon la technologie employée. Ainsi le 454 ampute facilement l'amorce à l'extrémité 5', ou bien oscille entre des cas d'insertions ou de délétions internes, tandis que l'écrasante majorité des mutations Illumina sont dues à des substitutions. Enfin, s'il est difficile d'améliorer les résultats sur 454, mêmes les cas les plus importants ne concernant qu'un faible pourcentage de séquences, les amorces mutées manquées en Illumina sont principalement rattachées au même pattern, à savoir 2 délétions internes.

Enfin, en regroupant tous ces cas, on constate qu'on a récupéré la grande majorité des amorces mutés, les $\sim$1-3 % restant se subdivisant en une multitude de cas uniques.

Modèle final :

Final_mutated_V4F :
`CCAGCA[GC]C[CT]GCGGTAATTCC` avec au plus 2 mutations

Final_mutated_V4R :
`CTTTCGTTCTTGAT[CT][AG]A` avec au plus 2 mutations

A.6 Vérification des découpages introns/exons et comparaisons des ARNm alternatifs entre gènes orthologues

Rechercher l'ensemble des ARNm potentiels dans un gène s'est avéré trop ambitieux pour Logol. Néanmoins, suite à ce revers, un second projet a émergé en collaboration avec Samuel Blanquart. Lui et son équipe sont parvenus à déduire les découpages introns/exons des ARNm alternatifs des gènes chez l'humain et la souris et ont démontré une relative conservation entre les gènes orthologues[2]. L'idée est donc devenu de construire des modèles d'ARNm connues (nombre d'introns/exons, éventuellement leurs tailles) afin de tester s'ils sont conservés dans des espèces homologues. En ajoutant davantage d'informations, nous espérons pouvoir simplifier l'analyse et pouvoir passer à l'échelle.

Problématique biologique

Samuel Blanquart et ses collaborateurs ont analysé 1 892 gènes orthologues chez la souris et l'humain ainsi que tous les transcrits associés identifiés. Ils ont ensuite décrit la composition de ces transcrits selon un formalisme grammatical figurant les codons start/stop, les donneurs/accepteurs d'épissage et la taille des introns/exons. Ils ont ainsi pu mettre en évidence des conservations entre les ARNm alternatifs de l'homme et de la souris, parfois avec des tailles strictement identiques.

Dans un premier temps, l'idée est de « traduire » ces grammaires en grammaire Logol afin de pouvoir scanner les gènes ciblés et vérifier s'il n'y a pas d'erreur dans le transcrit décrit par Samuel Blanquart. Puis, dans un second temps, d'utiliser les grammaires des transcrits humains sur les gènes de la souris afin d'identifier des transcrits alternatifs potentiels, actuellement non-identifié biologiquement, dans une espèce orthologue à partir d'une espèce de référence proche.

Enjeux biologiques

— Identifier un ARNm alternatif à partir d'une signature établie en terme d'enchaînement de tailles d'introns/exons par le biologiste (validation de la signature)
— Rechercher la présence de cette signature dans des gènes orthologues (uniquement si les exons sont de tailles identiques dans les deux espèces)

Enjeu bioinformatiques

— Vérification intrinsèque de l'ORF dans un modèle fragmenté
— Traduction d'une signature formelle en un modèle Logol

Données disponibles

1 892 Gènes orthologues humain/souris

Dans cette étude, nous disposons de 1 892 gènes orthologues entre l'humain et la souris. Ces gènes contiennent un fragment de séquence non-codante en amont et en aval du gène proprement dit.

10 875 modèles de transcrits

La liste de gènes est accompagnée des 6 043 transcrits humains et 4 832 transcrits souris qui dérivent de ces gènes. Chacun de ces transcrits est décrit par Samuel Blanquart sous la

2. Travaux actuellement non publiés

forme d'une grammaire, tel que celle-ci :

```
[ATG C:927 <GT .:1120 AG> D:90 <GT .:3577 AG> E:70 <GT
.:13760 AG> F:74 <GT .:32660 AG> G:9 H:66 <GT .:4161 AG>
I:61 <GT .:4458 AG> K:267 <GT .:586 AG> L:161 <GT .:1299
AG> M:98 <GT .:1235 AG> N:43 <GT .:140 AG> O:82 <GT .:243
AG> P:86 <GT .:380 AG> Q:147 <GT .:3466 AG> R:96 <GT .:825
AG> S:255 <GT .:2961 AG> T:75 <GT .:1546 AG> U:98 <GT
.:3208 AG> V:130 <GT .:784 AG> W:126 TAA]
```

FIGURE A.11 – Grammaire de l'intron intégrant le motif de la boîte de branchement

Où :
- Les codons start sont représentés par [ATG
- Les codons stop sont représentés par TGA]
- Les accepteurs d'épissage sont représentés par AG>
- Les donneurs d'épissage sont représentés par <GT
- Les exons sont représentés par une lettre suivie de la taille de l'exon (ex : C :927). Cette taille inclut les codons start/stop le cas échéant.
- Les introns sont représentés par un point suivi de la taille de l'intron (ex : . :1120). Cette taille inclut les donneurs/accepteurs d'épissage.

Ainsi, l'expression `ATG C:4 <GT .:5 AG> D:2 <GT .:6 AG> W:6 TAA` représente une séquence de taille 23 nucléotides (4+5+2+6+6) de la forme :

ATGNGTNAGNNGTNNNNAGNNNTAA

Le codon start est surligné en orange, le codon stop sont surligné en vert, les accepteurs et donneurs d'épissages sont indiqués en rouge et les introns les plus courts sont soulignés. Note : Les donneurs et accepteurs d'épissage peuvent se combiner de différentes façons entre eux, qui peuvent produire plusieurs introns différents. Ceux-ci n'ont pas été indiqué sur les séquences par soucis de lisibilité.

Dans un premier temps, l'idée était d'abord de tester la capacité de Logol à effectuer une recherche du transcrit exact dans un gène. Deux stratégies ont été envisagées.

Première stratégie : grammaire G1, regroupant toutes les contraintes

La première vise à rechercher directement le transcrit tout en vérifiant le maintient d'un ORF valide au terme de l'épissage (i.e vérification que la somme de la taille de l'ensemble des exons forment un multiple de trois et qu'aucun codon stop n'apparait avant la fin du dernier exon). On ne vérifie pas que les introns sont encadrés par des accepteurs/donneurs d'épissage.

Grammaire G1 :

```
def:{
morphism(nonstop,a,c)
morphism(nonstop,a,t)
morphism(nonstop,c,a)
morphism(nonstop,c,c)
morphism(nonstop,c,g)
morphism(nonstop,c,t)
morphism(nonstop,g,c)
morphism(nonstop,g,g)
morphism(nonstop,g,t)
morphism(nonstop,t,a)
morphism(nonstop,t,c)
morphism(nonstop,t,g)
morphism(nonstop,t,t)
}
mod1()==>"atg",repeat(!("t",("aa"|"ga"|"ag")):{#[3,3]},[0,0])
    +[308,308],INTRON1:{#[1120,1120]},repeat(!("t",("aa"|"ga"|"ag"))
    :{#[3,3]},[0,0])+[30,30],INTRON2:{#[3577,3577]},repeat(!("t",("aa
    "|"ga"|"ag")):{#[3,3]},[0,0])+[23,23],((!"t",INTRON3
    :{#[13760,13760]},END1:{#[2,2]})|("t",INTRON3:{#[13760,13760]},END1
    :{#[1,1],_NUC1},+"nonstop" ?NUC1)),repeat(!("t",("aa"|"ga"|"ag"))
    :{#[3,3]},[0,0])+[24,24],INTRON4:{#[32660,32660]},repeat(!("t",("aa
    "|"ga"|"ag")):{#[3,3]},[0,0])+[22,22],INTRON5:{#[4161,4161]},repeat
    (!("t",("aa"|"ga"|"ag")):{#[3,3]},[0,0])+[20,20],((!"t",INTRON6
    :{#[4458,4458]},END2:{#[2,2]})|("t",INTRON6:{#[4458,4458]},END2
    :{#[1,1],_NUC2},+"nonstop" ?NUC2)),repeat(!("t",("aa"|"ga"|"ag"))
    :{#[3,3]},[0,0])+[88,88],((!"t",INTRON7:{#[586,586]},END3:{#[2,2]})
    |("t",INTRON7:{#[586,586]},END3:{#[1,1],_NUC3},+"nonstop" ?NUC3)),
    repeat(!("t",("aa"|"ga"|"ag")):{#[3,3]},[0,0])+[53,53],INTRON8
    :{#[1299,1299]},repeat(!("t",("aa"|"ga"|"ag")):{#[3,3]},[0,0])
    +[32,32],((!"t",START4:{#[1,1]},INTRON9:{#[1235,1235]},END4
    :{#[2,2]})|("t",START4:{#[1,1],_NUC4},INTRON9:{#[1235,1235]},+"
    nonstop" ?NUC4)),repeat(!("t",("aa"|"ga"|"ag")):{#[3,3]},[0,0])
    +[14,14],INTRON10:{#[140,140]},repeat(!("t",("aa"|"ga"|"ag"))
    :{#[3,3]},[0,0])+[27,27],((!"t",INTRON11:{#[243,243]},END5
    :{#[2,2]})|("t",INTRON11:{#[243,243]},END5:{#[1,1],_NUC5},+"nonstop
    " ?NUC5)),repeat(!("t",("aa"|"ga"|"ag")):{#[3,3]},[0,0])+[28,28],
    INTRON12:{#[380,380]},repeat(!("t",("aa"|"ga"|"ag"))
    :{#[3,3]},[0,0])+[49,49],INTRON13:{#[3466,3466]},repeat(!("t",("aa
    "|"ga"|"ag")):{#[3,3]},[0,0])+[32,32],INTRON14:{#[825,825]},repeat
    (!("t",("aa"|"ga"|"ag")):{#[3,3]},[0,0])+[85,85],INTRON15
    :{#[2961,2961]},repeat(!("t",("aa"|"ga"|"ag")):{#[3,3]},[0,0])
    +[25,25],INTRON16:{#[1546,1546]},repeat(!("t",("aa"|"ga"|"ag"))
    :{#[3,3]},[0,0])+[32,32],((!"t",START6:{#[1,1]},INTRON17
    :{#[3208,3208]},END6:{#[2,2]})|("t",START6:{#[1,1],_NUC6},INTRON17
    :{#[3208,3208]},+"nonstop" ?NUC6)),repeat(!("t",("aa"|"ga"|"ag"))
    :{#[3,3]},[0,0])+[43,43],INTRON18:{#[784,784]},repeat(!("t",("aa"|"
    ga"|"ag")):{#[3,3]},[0,0])+[41,41],"taa"
mod1()==*>SEQ1
```

FIGURE A.12 – Grammaire de l'intron intégrant le motif de la boîte de branchement

Dans cette grammaire, les exons ont été traduits sous la forme d'un repeat d'un élément de

taille 3 qui ne soit pas un codon stop (exprimé via `!("t",("aa"|"ga"|"ag")):{\#[3,3]}`).

Par ailleurs, puisque la taille des exons est connue, il est possible de déduire à quel endroit de l'ORF s'intercalent les introns et de vérifier plus facilement que leur retrait ne provoquera pas un codon stop. Lorsqu'un intron s'intercale dans un codon, l'intron est modélisé par un choix entre deux sous-grammaires qui vérifie la nature du premier nucléotide du codon. Si ce n'est pas un T, la nature des deux nucléotides suivants importe peu. Autrement, on interdit la présence d'un codon stop en utilisant le morphisme non-stop.

Par exemple, si on considère la recherche de l'intron numéro 6 : `((!"t",INTRON6:{#[4458,4458]} ,END2:{#[2,2]})|("t",INTRON6:{#[4458,4458]},END2:{#[1,1],_NUC2},+"nonstop" ?NUC2))`. Par rapport à l'ORF qui considère les nucléotides par groupe de 3, l'intron débute en position 2. C'est-à-dire qu'il est positionné entre le nucléotide numéro 1 et le nucléotide numéro 2 du codon qui se formera après son épissage. Contrairement à la grammaire présentée en 5.6, il n'est donc pas nécessaire de vérifier pour chaque intron si celui-ci débute en position 2 ou en position 2 et de prévoir chacun de ces cas de figure dans la grammaire.

Dans un soucis d'alléger la grammaire dans un premier temps, la grammaire n'inclut pas de vérification de la présence des donneurs/receveurs d'introns dans les introns. Si cette première version de la grammaire se comporte de façon raisonnable dans l'analyse des séquences, cette vérification sera intégrée à la grammaire dans un second temps.

La traduction de la grammaire de l'équipe de Samuel Blanquart en une grammaire Logol résulte d'un traitement automatique spécifiquement mis au point pour l'occasion.

Deuxième stratégie : coopération entre une grammaire G2 et G3

L'analyseur a passé un temps "infini" à scanner le gène cible avec la grammaire G1. Nous avons émis l'hypothèse que cette grammaire était finalement trop lourde pour l'analyseur et avons tenté une deuxième approche visant à employer successivement deux grammaires (G2 et G3) pour analyser une séquence. La grammaire G2 a pour objectif de rechercher le prédécoupage intron/exon. Un filtre traite les résultats de ce découpage pour reconstruire l'ARNm (c-à-d en raboutant l'ensemble des exons) et la grammaire G3 se chargerait d'analyser le respect d'une ORF valide.

Grammaire G2 :

```
def:{
}
mod1()==>"atg",EXON1:{#[924,924]},INTRON1:{#[1120,1120]},EXON2
    :{#[90,90]},INTRON2:{#[3577,3577]},EXON3:{#[70,70]},INTRON3
    :{#[13760,13760]},EXON4:{#[74,74]},INTRON4:{#[32660,32660]},EXON5
    :{#[66,66]},INTRON5:{#[4161,4161]},EXON6:{#[61,61]},INTRON6
    :{#[4458,4458]},EXON7:{#[267,267]},INTRON7:{#[586,586]},EXON8
    :{#[161,161]},INTRON8:{#[1299,1299]},EXON9:{#[98,98]},INTRON9
    :{#[1235,1235]},EXON10:{#[43,43]},INTRON10:{#[140,140]},EXON11
    :{#[82,82]},INTRON11:{#[243,243]},EXON12:{#[86,86]},INTRON12
    :{#[380,380]},EXON13:{#[147,147]},INTRON13:{#[3466,3466]},EXON14
    :{#[96,96]},INTRON14:{#[825,825]},EXON15:{#[255,255]},INTRON15
    :{#[2961,2961]},EXON16:{#[75,75]},INTRON16:{#[1546,1546]},EXON17
    :{#[98,98]},INTRON17:{#[3208,3208]},EXON18:{#[130,130]},INTRON18
    :{#[784,784]},EXON19:{#[123,123]},"taa"
mod1()==*>SEQ1
```

FIGURE A.13 – Grammaire de l'intron intégrant le motif de la boîte de branchement

La grammaire G2 est une grammaire très simple, qui se contente de vérifier qu'il est possible de découper la séquence en une succession de portion de tailles définies.

Grammaire G3 :

```
def:{
morphism(nonstop,a,c)
morphism(nonstop,a,t)
morphism(nonstop,c,a)
morphism(nonstop,c,c)
morphism(nonstop,c,g)
morphism(nonstop,c,t)
morphism(nonstop,g,c)
morphism(nonstop,g,g)
morphism(nonstop,g,t)
morphism(nonstop,t,a)
morphism(nonstop,t,c)
morphism(nonstop,t,g)
morphism(nonstop,t,t)
}

mod1()==>"atg",repeat(!("t",("aa"|"ga"|"ag")):{#[3,3]},[0,0])
    +[982,982],"taa"
mod1()==*>SEQ1
```

FIGURE A.14 – Grammaire de l'intron intégrant le motif de la boîte de branchement

La grammaire G3 vérifie que la séquence d'introns reconstruite d'après les résultats de G2

vérifie une ORF valide : elle recherche littéralement un codon start, une succession de codon non-stop et pour terminer un codon stop.

Résultats biologiques

Les deux stratégies ont été tentées sur le gène ENSMUSG00000032582, à partir du découpage intron/exon du transcrit ENSMUST00000035201 (constitué de 18 introns et 19 exons, cf grammaires G1 et G2). Le gène ENSMUSG00000032582 fait 99 679 nucléotides de long.

Malheureusement, là encore, bien que la grammaire soit théoriquement fonctionnelle, dans les faits, elle s'avère trop lourde pour l'outil, qui ne parvient pas à scanner les séquences. Le temps de calcul explose de façon complètement anormale, même lors de l'utilisation de la grammaire G2, qui semble pourtant très basique.

Réflexion sur Logol

L'explosion du temps de calcul est tout à fait anormale au regard de la simplicité de la grammaire : dans le dernier cas, il s'agit uniquement de chercher un start, puis un stop à n positions plus loin, ce qui semble être une tâche simple. Pour essayer d'estimer à quel moment le temps de calcul cesse de devenir raisonnable, nous avons testé la grammaire G2 en augmentant progressivement le nombre d'éléments introns et d'exons dans la séquence. Le test a été mené sur la séquence ENSMUSG00000032582 entière.

Eléments Grammaticaux (inclusifs)	Temps d'analyse
Codon start	1 seconde
+ Exon1 (924nt)	12 seconde
+ Intron1 (1 120nt)	21 seconde
+ Exon2 (90nt)	25 seconde
+ Intron2 (3 557nt)	3 min 9 seconde
+ Exon3 (70nt)	3 min 5 seconde
+ Intron3 (13 760nt)	Abandonné après 24h

On remarque que le temps croît de façon disproportionnée en fonction de la taille des éléments ajoutés à la grammaire. C'est d'autant plus anormal qu'il s'agit d'élément non-défini, qui ne nécessite aucune vérification quant à leur nature : on recherche ATG, suivit d'un bloc de taille 924 quel qu'il soit. Ce problème complètement inattendu pose la question de l'efficacité de l'outil, qui devra être revu pour pouvoir être réellement utilisable.

Comparé à la recherche de l'ensemble des ARNm alternatifs présentée en 5.6, la vérification d'un découpage possible constitue un cas bien plus simple, que peut facilement prendre en charge une grammaire.

L'incapacité à rechercher ce type de découpage avec une grammaire de haut niveau est donc d'autant plus frustrant que ce cas d'étude a été, à dessein, simplifié : une simple expression régulière, tel que `"atg([atcg]{924})([atcg]{1120})([atcg]{90})taa"` en python, est tout à fait capable de faire l'analyse du découpage. En l'état, bien que le langage Logol soit d'un niveau d'expressivité largement supérieur aux langages réguliers, il s'avère "technologiquement" plus limité sur ce cas de figure.

Bien que l'expressivité des grammaires constitue un véritable point fort pour le pattern matching, il ne faut pas oublier que l'implémentation de cette expressivité peut engendrer un surcoût en termes de temps de recherche, qui peut s'avérer finalement rédhibitoire, comme

c'est le cas ici. Dans l'optique d'une diffusion large au sein de la communauté scientifique, cette balance entre possibilité et efficacité est à garder en tête.

A.7 Modélisation d'un structure composite permettant une lecture alternative des ARNm

Problématique biologique

L'un des mécanismes biologiques permettant de multiplier le nombre d'ARNm obtenu à partir d'une unique séquence ADN est le décalage de phase ribosomal (Ribosomal framshifting). Les structures à l'origine de ces décalages de phase sont appelés Frameshift. Les motifs Frameshift-1 sont des motifs présents sur certains ARNm, qui permettent de faire glisser le ribosome d'un cran en arrière lors de la synthèse de la protéine. Ce recul entraîne un décalage de l'ORF et donc des codons. Cela permet de synthétiser deux protéines différents à partir d'un même brin d'ARNm. Ce recul a lieu sur une séquence appelée "site de glissement", formée d'un structure NNX XXY YYZ, situé juste en amont de la structure secondaire du frameshift-1 [Bri95, RB12].

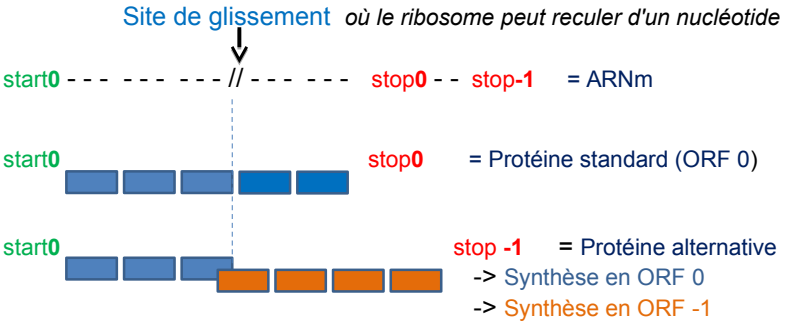


FIGURE A.15 – Production alternative d'une protéine par l'effet d'une structure frameshift-1

La structure du Frameshift-1 fait intervenir de nombreuses contraintes : un codon start, un site de glissement positionné à une distance précise d'une structure en pseudo-noeud et un codon stop positionné sur un cadre de lecture -1.

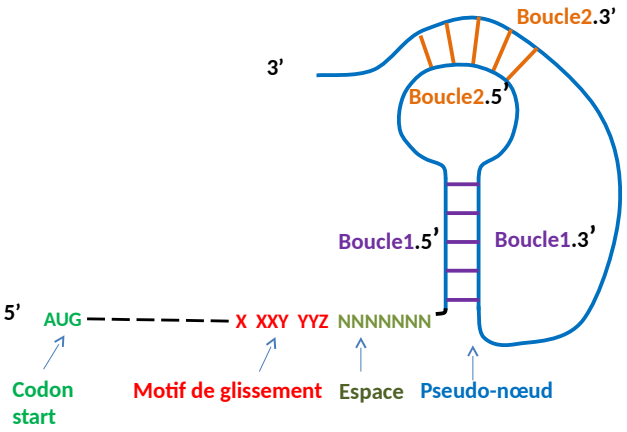


FIGURE A.16 – Production alternative d'une protéine par l'effet d'une structure frameshift-1

Il n'existe pas d'outils génériques permettant de rechercher les Frameshift-1. Or, l'ensemble des contraintes qui interviennent dans un Frameshift sont des contraintes qu'il est possible de modéliser sous forme de grammaire. Il était donc intéressant de voir si ce genre de motif complexe était à la portée de Logol.

Enjeu Biologique

— Reconnaître in silico le potentiel d'un ARNm a subir un évènement de frameshift-1

Enjeux bioinformatiques

— Mise en place d'un modèle composite complexe (pseudo-nœud, motif de glissement, etc).

Données disponibles

Structure du frameshift-1

La structure du Frameshift-1 lui-même suit la logique suivante :

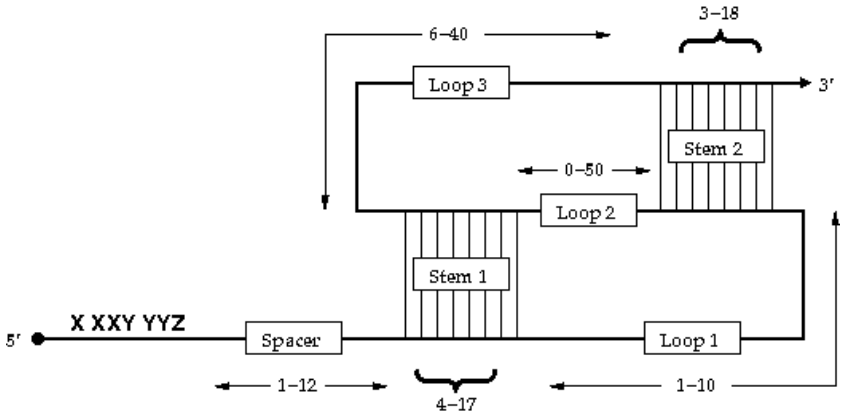


FIGURE A.17 – Production alternative d'une protéine par l'effet d'une structure frameshift-1 [TRG08]

Où :

— Le site de glissement est constitué d'un heptamère XXXYYYZ, avec X pouvant être n'importe nucléotide, Y étant soit un A soit un U et Z étant tout sauf un G [DW92, BJI92, DIW91, JV85].
— Le spacer fait une taille inférieure à 12 [KNB01] (généralement comprise entre 6 et 9 [HROS07]).
— Le pseudo-noeud est composé de deux boucles imbriquées acceptant des mismatchs et liaisons wobbles (appariement GU, plus faible que les liaisons wobble)[LNKB99]. Les tailles respectives des éléments du pseudo-noeud sont indiqués en figure A.17. La section Loop2 est facultative, d'où une taille minimum de 0.

Données positives

Un jeu de référence a été construit à partir de données issues de la base Recode-2 [BFZ$^+$10], qui recense les évènements de recodage, dont les frameshift-1.

Cette base contient 68 évènements de Frameshift-1, dont 30 sont associés à un pseudo-nœud.

Modélisation

Frameshift-1, modèle complet

```
def:{
morphism(wcw,a,t)
morphism(wcw,t,a)
morphism(wcw,c,g)
morphism(wcw,g,c)
morphism(wcw,g,t)
morphism(wcw,t,g)
}
controls:{
% "c"[mod3.SA5,mod3.STEM15,mod3.SZ5,mod3.SZ3,mod3.STEM13,
   mod3.SA3]>=25
% "c"[mod3.SX5,mod3.STEM25,mod3.SY5,mod3.SY3,mod3.STEM23,
   mod3.SX3]>=25
}

mod4()==>(("aaa")|("ccc")|("ttt")|("ggg"))

mod2()==>mod4(),(("aaa")|("ttt")),! "g":{#[1,1]}

mod3()==>(A5:{#[1,1],_SA5},S15:{#[2,14],_STEM15},Z5
   :{#[1,1],_SZ5}):{%"gc":50},
          L1:{#[1,5]},
          (X5:{#[1,1],_SX5},S25:{#[1,6],_STEM25},Y5
             :{#[1,1],_SY5}):{%"gc":50},
          L11:{#[0,4]},
          -"wcw" Z3:{?SZ5,_SZ3},-"wc" ?STEM15:{_STEM13}:{p$
             [0,34]},-"wcw" A3:{?SA5,_SA3},
          L2:{#[4,40]},
          -"wcw" Y3:{?SY5,_SY3},-"wc" ?STEM25:{_STEM23}:{p$
             [0,34]},-"wcw" X3:{?SX5,_SX3}

mod1()==>CC1:{#[2,2]},mod2(),SPACER2:{#[1,10]},mod3()

mod1()==*>SEQ1
```

FIGURE A.18 – Grammaire de l'intron intégrant le motif de la boîte de branchement

Où :

— Le morphisme wcw définit les liaisons Watson-Crick et Wobble (c'est-à-dire que le G aussi complémentaire du T).
— le contrôle défini un pourcentage de C minimum pour l'ensemble des éléments définissant la tige 1 ainsi que pour l'ensemble des éléments définissant la tige 2.
— Le mod1 définit les 2 premiers nucléotides du codon dans lequel débute le motif de glissement détaillé dans le mod2, un spacer de taille compris entre 1 et 10 et le pseudonoeud détaillé dans le mod3.

— Le mod2 définit le motif de glissement XXXYYYZ. Il commence par le mod4, qui définit le triplet XXX, constitué d'une répétition identique de n'importe quel nucléotide. Le triplet YYY est modélisé par un choix entre une triple répétition de "a" ou une triple répétition de "t". Enfin, le Z est représenté un séquence de taille 1 qui ne peut pas être un "g".
— Le mod3 détaille le pseudo-noeud, selon le schéma présenté en figure A.19. Chaque tige-boucle a été est séparée en trois section. Les sections aux extrémités autorisent des liaisons wobbles, sans limite, tandis que la section centrale n'autorise que des appariements Watson-Crick et 34% de mésappariemment. Enfin, l'ensemble des 3 sections de la tige-boucle doivent compter au moins 50% de liaisons GC. Ce compte est vérifié sur un seul brin de la tige en se basant sur les C (les G ne sont pas représentatifs à cause des liaisons Wobbles).

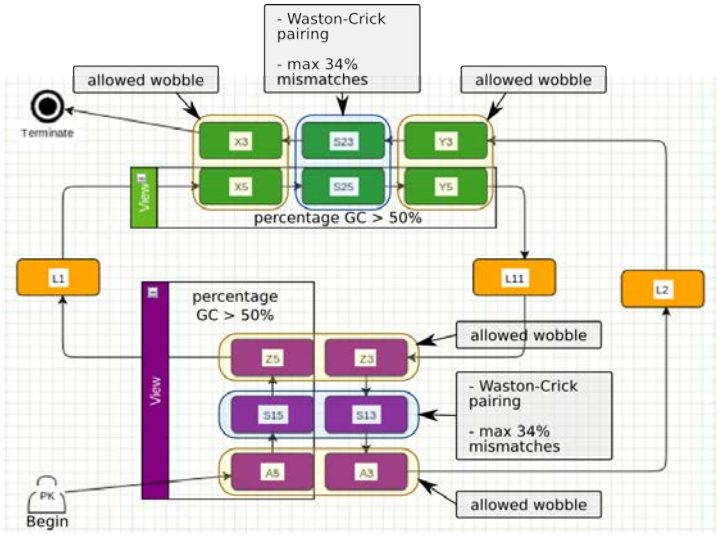


FIGURE A.19 – Détail du pseudo-noeud dans la grammaire Logol

Il est intéressant de noter que la grammaire ne décompte pas réellement le pourcentage de liaison GC. En effet, le contrôle ne vérifie que la quantité de C présent dans les trois sections d'une tige, hors la tige centrale autorise des mésappariemments. Un mésappariement faisant intervenir un C sur le brin contrôlé est tout de même compté comme une liaison GC.

Résultats biologiques

Le modèle Logol du Frameshift-1 permet d'identifier un hit identique à la référence de Recode-2 pour 26 des 30 séquences du jeu de test.

Le modèle Logol a permis de scanner le génome de Bacillus Subtilis (4 215 600 nuclétides) et a aboutit à 7 000 hits en deux heures (sur une machine Intel X5550, 144Go RAM).

Réflexion sur Logol et les modèles grammaticaux

Le motif de Frameshift-1 basé sur un pseudo-nœud fait intervenir de nombreuses contraintes tel que des motifs structures, des tiges-boucles, des liaisons wobble ou des pourcentages GC. Ces contraintes sont faciles à modéliser sous forme de grammaire SVG et fonctionne bien ensemble. Il est donc possible de définir un modèle de Frameshift-1 fonctionnel sous forme de grammaire.

Même si ce n'est qu'un exemple, cette étude montre que Logol, un outil de pattern matching générique, possède une expressivité suffisante pour rechercher des motifs complexes habituellement recherché par des outils spécialisés.

Un dernier point intéressant réside dans la proportion des liaisons "GC" dans les tige-boucles du modèle. La contrainte de 50% de C dans les tiges n'est pas suffisante car la partie centrale de la tige-boucle autorise des mésappariements, donc des C sans nucléotides complémentaires en vis-à-vis. C'est pour cela que la grammaire définit un contrôle qui vérifie au moins 25% de GC dans l'ensemble des brins d'une tige. Bien que cette façon de compter considère un G mésapparié et un C mésapparié comme une même liaison GC, cela constitue déjà une première approximation de la notion de stabilité d'une tige boucle. Il est donc possible de "mimer" dans une certaine mesure les calculs d'énergie libre des structures secondaires pour les intégrer dans des grammaires.

Résumé

Cette thèse en bioinformatique étudie l'intérêt d'utiliser des modèles grammaticaux pour rechercher des motifs dans des séquences génomiques.
Depuis les années 80, à l'initiative notamment de David Searls, des travaux ont montré qu'en théorie, des grammaires de haut niveau offrent suffisamment d'expressivité pour permettre la description de motifs biologiques complexes, notamment par le biais d'une nouvelle classe de grammaire dédiée à la biologie : les grammaires à variables de chaîne (SVG, String Variable Grammar).
Ce formalisme a donné lieu à Logol, qui est un langage grammatical et un outil d'analyse développé dans l'équipe Dyliss où a lieu cette thèse. Logol est un langage conçu pour être suffisamment flexible pour se plier à une large gamme de motifs qu'il est possible de rencontrer en biologie.
Le fait que les grammaires restent inutilisée pour la reconnaissance de motifs pose question. Le formalisme grammatical est-il vraiment pertinent pour modéliser des motifs biologiques ?
Cette thèse tente de répondre à cette question à travers une démarche exploratoire. Ainsi, nous étudions la pertinence d'utiliser les modèles grammaticaux, via Logol, sur six applications différentes de reconnaissance de motifs sur des génomes.
Au travers de la résolution concrète de problématiques biologiques, nous avons mis en évidence certaines caractéristiques des modèles grammaticaux. Une de leurs limites est que leur utilisation présente un coût en termes de performance. Un de leurs atouts est que leur expressivité couvre un large spectre des motifs biologiques, contrairement aux méthodes alternatives, et d'ailleurs certains motifs modélisés par les grammaires n'ont pas d'autres alternatives existantes. Il s'avère en particulier que pour certains motifs complexes, nécessitant une finesse de contrôle, l'exploitation de l'ambiguïté ou la définition par la négative, l'approche grammaticale est la plus adaptée.
Pour finir, l'une des conclusions de cette thèse est qu'il n'y a pas réellement de compétition entre les différentes approches, mais plutôt qu'il y a tout à gagner d'une coopération fructueuse.

Mots-clefs : Bioinformatique, reconnaissance de motifs, grammaire SVG, modèle grammatical, séquence génomique

Abstract

This thesis studies the interest to look for patterns in genomic sequences using grammars.
Since the 80s, work has shown that, in theory, high level grammars offer enough expressivity to allow the description of complex biological patterns. In particular David Searls has proposed a new grammar dedicated to biology : string variable grammar (SVG).
This formalism has resulted in Logol, a grammatical language and an analysis tool developed by Dyliss team where this thesis is taking place. Logol is a language designed to be flexible enough to express a wide range of biological patterns.
The fact that the grammars remain unknown to model biological patterns raises questions. Is the grammatical formalism really relevant to the recognition of biological patterns ? This thesis attempts to answer this question through an exploratory approach. We study the relevance of using the grammatical patterns, by using Logol on six different applications of genomic pattern matching.
Through the practical resolution of biological problems, we have highlighted some features of grammatical patterns. First, the use of grammatical models presents a cost in terms of performance. Second the expressiveness of grammatical models covers a broad spectrum of biological patterns, unlike the others alternatives, and some patterns modeled by grammars have no other alternative solutions. It also turns out that for some complex patterns, such as those combining sequence and structure, the grammatical approach is the most suitable.
Finally, a thesis conclusion is that there was no real competition between different approaches, but rather everything to gain from successful cooperation.

Keywords : Bioinformatics, pattern matching, string variable grammar, grammatical patterns, genomic sequences