Nouvelle stratégie d'annotation des génomes par l'utilisation d'algorithmes d'intelligence artificielle

Nicolas Scalzitti

UNIVERSITÉ DE STRASBOURG

ÉCOLE DOCTORALE DES SCIENCES DE LA VIE ET DE LA SANTÉ

Laboratoire ICube – UMR7357

THÈSE présentée par :

Nicolas SCALZITTI

soutenue le : 29 septembre 2021

Pour obtenir le grade de : *Docteur de l'Université de Strasbourg*

Discipline/Spécialité : *Sciences de la vie et de la santé, spécialité Bioinformatique*

Nouvelle stratégie d'annotation des génomes par l'utilisation d'algorithmes d'intelligence artificielle

THÈSE dirigée par :

Mme THOMPSON Julie	*Directeur de recherche, CNRS*
M COLLET Pierre	*Professeur, Université de Strasbourg*

RAPPORTEURS EXTERNES :

Mme LAINE Élodie	*Maitre de conférences, Sorbonne Université*
M LEGRAND Pierrick	*Maitre de conférences, Université de Bordeaux*

EXAMINATRICE INTERNE :

Mme LECOMPTE ODILE	*Professeur, Université de Strasbourg*

EXAMINATEUR EXTERNE :

M BANZHAF Wolfgang	*Professeur, Michigan State University*

Remerciements

Avant tout, je tiens à exprimer mes remerciements les plus sincères au Dr LAINE Élodie, Dr LEGRAND Pierrick, Pr BANZHAF Wolfgang, Pr LECOMPTE Odile et Dr SMAIL-TABBONE Malika, d'avoir accepté de participer à mon jury, ainsi que pour l'honneur qu'ils me font à examiner et juger mes travaux de thèse.

Ce manuscrit a pu voir le jour malgré de nombreuses péripéties et chaque membre de l'équipe du CSTB (et plus encore) y a contribué. Je tiens ainsi, par ces quelques lignes, à leur adresser mes plus sincères remerciements.

CSTB, quatre lettres qui ne veulent probablement pas dire grand-chose à la majorité des gens, cependant, pour moi, elles resteront gravées dans ma mémoire. Chacune a une signification particulière. C pour Communication, car il est crucial de pouvoir discuter avec son entourage, tant autour d'un café, que dans les salles de réunion où certaines discussions ont été cruciales et m'ont permis d'avancer, de comprendre, d'avoir des déclics et d'enrichir ma culture. S pour Science, parce que c'est ce qu'on fait au laboratoire, bien que j'ai pu avoir des doutes dans certains cas. Hormis les compétences et la rigueur des membres de ce laboratoire, il en découle une vraie philosophie de vie. La science est la quintessence de ce laboratoire animé par des gens passionnés. T pour Travail, parce que pour faire une thèse il en faut, et beaucoup. J'ai appris énormément de choses en travaillant avec les membres du laboratoire. La maîtrise des techniques et le savoir de chacun ont été des bibliothèques scientifiques qu'ils n'ont pas hésité à me faire découvrir. Pour B je voulais d'abord mettre "boisson à base de *Saccharomyces cerevisiae"* mais ça faisait trop scientifique, alors j'ai choisi Bonne humeur, parce qu'il n'y a pas un jour où je me suis levé en me disant que je ne voulais pas aller au labo. Je ne vais pas faire mon philosophe et citer Confucius, mais il avait raison. Chaque jour, avec les chanceux (ou pas) qui ont partagé mon bureau, ma table à midi ou simplement un morceau de couloir, a été un bonheur grâce à l'ambiance et l'esprit de partage. Ce fut un réel plaisir et un honneur de travailler avec vous, au sein de la maison CSTB.

Ces quelques lignes ne seront jamais représentatives de votre contribution, mais j'avais à cœur de les écrire. J'étais obligé de commencer par toi, hé oui ! Toto, mon "collègue" de bureau. Pour ta bonne humeur, ton soutien, ton enthousiasme, ta tchatche, ta sagacité, ta générosité. Pour les cours de trading et d'art pythonique. Pour tes conseils et ta bienveillance, merci pour tout Thomas. Merci également à Kirsley, avec qui j'ai partagé pendant un petit bout de temps notre bureau. Tes connaissances sont aussi grandes que ton cœur. Merci de m'avoir tant appris, d'avoir partagé tes secrets (notamment celui de la recette du mojito). Enfin, pour compléter la liste des "blaireaux" du bureau 132, il n'en manque plus qu'un. Pour nos discussions intenses et enrichissantes, pour ton humour et ton aide, merci Merguez, euh Luc. S'il y a bien un petit malin qui a vite compris qu'il fallait fuir ce bureau, c'est Romain. Mais cela ne l'a pas empêché de partager, à une autre époque, un autre superbe bureau. Merci en tout cas pour ton aide, ton altruisme, et pour les GAF. Je n'oublie bien sûr pas un des membres les plus importants du laboratoire, car il gère d'une poigne de fer le système informatique, donc merci Arnaud pour ton support technique. J'ai peut-être oublié de préciser que ta gentillesse, ta bonne humeur, ton investissement, ton aide, ton savoir et ta passion ont été un atout primordial pour la réussite de ma thèse, alors merci pour tout.

À l'ensemble de tous les "grands" du labo, qui ont participé de près ou de loin à cette thèse, merci. Merci Pierre P., Rabih, Jean-Sébastien, Christian, d'avoir partagé les connaissances de vos domaines. Merci à toi Raymond, pour m'avoir encadré, appris la programmation web, tenté de me convertir au tcl, ainsi que pour toutes les photos qui permettent de garder des souvenirs inestimables. J'aurais dû t'écouter…cette partie n'est vraiment pas facile à écrire et j'aurais dû m'y prendre bien plus

tôt ! Merci également Laetitia pour ton encadrement en stage, tu m'as fait découvrir le monde des réseaux qui est passionnant.

On dit souvent que la première impression est la bonne, mais pour le coup, je me suis bien trompé et j'en suis bien content, car j'ai pu faire la connaissance d'une personne merveilleuse et pétillante de vie. Alors pour tout ce que tu as fait, pour nos discussions et pour avoir été le meilleur public à mes blagues, merci pour tout Claudine. Merci Anne N. pour ton soutien et ton aide, tu nous as sorti des méandres administratifs, et sans toi, je nagerais encore en eaux inconnues. Enfin, merci Anne J., pour ton apport scientifique et ton encadrement. Tes connaissances en IA ont été une ressource essentielle à la mise en place de cette thèse.

Je tenais également à remercier tous les "djeuns" qui sont en train de vivre (ou finir) cette superbe aventure qu'est la thèse. Merci à Corentin, Anna, Hiba, Christelle, Lalla, Amani et Soumaya. Bon courage pour la suite. Je tiens également à remercier les anciens "djeuns" qui étaient là au début de ma thèse. Merci à Julio, Gopal, Yannis, Camille et Audrey. Une pensée également à Sarah, avec qui j'ai commencé au CSTB, à Bastien pour les soirées, ainsi qu'à Lucille, avec qui j'ai pu mettre en place certains de mes travaux. Merci également à mes anciens collègues Cyril et Stéphane, qui m'ont fait découvrir l'art de la paillasse plus jeune, le chemin parcouru aurait été probablement différent sans eux.

Je tiens aussi à remercier la fondation Bioniria et ses membres, qui m'ont permis d'initier cette thèse.

Je souhaite maintenant remercier trois personnes qui ont joué un rôle déterminant durant ces trois ans de thèse, et sans eux, rien n'aurait été possible. Je tiens donc à te remercier Pierre, pour avoir accepté d'être un de mes directeurs de thèse, pour m'avoir permis de mettre un pied à l'étrier, de m'avoir fait confiance et de m'avoir partagé ton savoir. Au travers de tes connaissances, j'ai découvert l'univers de l'évolution artificielle qui m'a fasciné. Je voudrais ensuite remercier un homme, sans qui le CSTB ne serait ce qu'il est. Cher Olivier, c'est au travers de ces mots que j'aimerais t'exprimer toute ma gratitude et la profonde affection que je te porte. Merci d'avoir été là quand tout s'est écroulé, de m'avoir fait détester les poires, de m'avoir guidé et d'avoir été une force de proposition. Merci d'avoir été plus qu'un chef et de m'avoir fait découvrir le monde parallèle des alignements. Ta joie de vivre, ton enthousiasme, ton énergie contagieuse, tes mimiques, ton humour (quoique) ont été le carburant de ma thèse. Merci pour ta patience, ta passion, tes conseils et ta culture. Merci coach, ou aigle 4, je ne sais plus. Tu as été et tu resteras un acteur emblématique de ma vie. Merci infiniment. Allez, à l'attaque !!

Enfin, j'aimerais remercier Julie. Lorsque je t'ai envoyé ma candidature de stage de M1, il y a de ça quelques années déjà, je ne m'attendais absolument pas à cette aventure. Grâce à toi, j'ai pu concrétiser mon projet. Merci de m'avoir soutenu, d'avoir été là au pire moment. Merci de m'avoir fait confiance et d'avoir cru en moi. Tu m'as permis de me dépasser, de repousser mes limites pour donner le meilleur de moi-même, je t'en suis extrêmement reconnaissant. Ton calme et ta sagesse lors des corrections (notamment pour mon anglais…) ont été indispensables, même si parfois c'était difficile lorsque tu dégainais le katana. Ton encadrement et tes conseils avisés ont participé à l'aboutissement de cette thèse, alors merci d'avoir été une directrice de thèse irréprochable. Tu es un exemple à suivre, donc pour tout ce que tu as fait, et tout ce que tu m'as apporté, merci !

Pour finir, j'aimerais remercier mes proches, qui étaient là dès le début et qui m'ont soutenu. Merci à mes deux sœurs Marina et Laura de m'avoir encouragé, et merci à vous deux, maman et papa, d'avoir cru en moi. Cette thèse est aussi la vôtre. Merci également aux potos, Fred, Bibi, Alex, Bastien et Gigi, qui m'ont sorti de ma bulle de temps à autre. Bien sûr je ne t'oublie pas, toi qui as été là pour m'encourager, me soutenir, me relever. Tu as été la clé de voûte de ma thèse. Pour tout le bonheur que tu m'as apporté et pour le futur qui nous attend, je te remercie du plus profond de mon cœur. Merci pour tout Célia.

SOMMAIRE

RÉSUMÉ DE LA THESE (VERSION FRANÇAISE)

Les approches d'intelligence artificielle

L'intelligence artificielle (IA) regroupe de nombreux algorithmes, notamment les systèmes experts, les algorithmes de *machine learning* comme les arbres de décision ou les réseaux de neurones, ou encore les algorithmes évolutionnaires (AE). Dans ces travaux de thèse, je me suis focalisé sur ces deux dernières approches, pouvant être complémentaires, afin de développer une nouvelle stratégie d'annotation des génomes eucaryotes.

Depuis des décennies, les algorithmes de *machine learning* ont été exploités dans de nombreux domaines. Ils se focalisent sur l'apprentissage des données d'entrée, pour ensuite réaliser des prédictions sur de nouvelles données inconnues. En ce sens, plusieurs méthodes d'apprentissage existent. Si les données sont étiquetées, c'est-à-dire que l'on connaît leur catégorie et donc le résultat que l'on doit obtenir, on parle d'apprentissage supervisé. Il s'agit communément de tâches de régression ou de classification, par exemple classer une variation génétique comme "pathogène" ou "bénigne". Au contraire, si les données ne sont pas étiquetées, on parle d'apprentissage non supervisé, le programme cherche à identifier les motifs, les éléments ou les structures sous-jacentes aux données d'entrée. On peut par exemple identifier des groupes de patients en fonction de corrélations génomiques.

Ces deux méthodes sont parmi les plus répandues et de nombreux algorithmes les utilisant ont été développés. Pour les méthodes d'apprentissage non supervisé, il existe des algorithmes de clustering tels que K-*means*, DBSCAN ou l'Analyse en Composantes Principales (ACP). Pour les méthodes d'apprentissage supervisé, on citera par exemple les régressions (linéaires ou logistiques), les Machines à Vecteurs de Support (SVM), les arbres de décision, les k-plus proches voisins ou les réseaux de neurones artificiels. Ce sont finalement les réseaux de neurones artificiels et notamment les réseaux de neurones profonds (*deep neural networks*) qui ont su se démarquer et qui sont couramment utilisés aujourd'hui. Les réseaux de neurones artificiels sont une bio-inspiration des neurones du cerveau des mammifères. L'unité centrale de ces algorithmes est un neurone artificiel qui s'interconnecte à ces homologues. Le processus d'apprentissage consiste à modifier les poids associés à chaque neurone afin de réduire l'erreur. Ce processus est réalisé automatiquement en se basant sur les données.

Un autre domaine de l'IA qui s'inspire également de la biologie est l'évolution artificielle. En effet, l'évolution artificielle reproduit certains mécanismes de l'évolution, tels

que le croisement de deux individus, les mutations spontanées et la sélection des meilleurs individus, dans le but d'optimiser une solution. Il existe plusieurs algorithmes d'évolution artificielle comme la stratégie d'évolution, les algorithmes génétiques ou encore la programmation génétique (PG). Parmi les algorithmes évolutionnaires, la PG s'inscrit dans une stratégie d'apprentissage des données pour résoudre les problèmes.

Finalement, les données sont la pierre angulaire de ces méthodes et l'arrivée du *Big Data* ainsi qu'une puissance de calcul plus importante ont permis à ces algorithmes de manifester leur performance.

IA et *Big Data*

Le *Big Data* a ouvert la voie à de nouvelles opportunités et a révolutionné notre approche du monde. Les nouvelles technologies du numérique produisent une quantité astronomique de données. Cette datasphère s'étend quotidiennement, générant de nouveaux défis pour stocker, traiter et exploiter ces données hétérogènes. Ces enjeux peuvent être définis par ce qui est appelé les "3V" du *Big Data* : Volume, Vélocité et Variété. Cela soulève également des problèmes liés à la confidentialité et à la protection des données, ainsi qu'à l'éthique ou l'échange entre pairs (dans le cadre de la médecine par exemple). En 2016, le Règlement Général sur la Protection des Données (RGPD) a été instauré pour protéger les données à caractère personnel symbolisant leur valeur et leur fragilité.

Cependant, le *Big Data* n'a pas que des défauts, il a ouvert la porte à de nombreuses applications dans de multiples domaines, notamment en IA. Dans le domaine de la biologie, les nouvelles technologies de séquençage ont par exemple permis l'acquisition d'une quantité considérable de données brutes. Grâce à ces technologies, des avancées majeures ont été réalisées, telles que le séquençage du génome humain, fondement de l'ère des *"*omiques*"*. Depuis, de nombreux algorithmes basés sur l'IA et exploitant les données biologiques multi-omiques ont vu le jour comme des programmes d'annotation de génome basés sur des prédicteurs de structures et de fonctions des protéines, de variants pathogènes, de gènes ou encore de sites d'épissage. Ainsi, les données biologiques sont l'exemple parfait de la représentation des «3V» du *Big Data*.

Annotation des génomes eucaryotes

Les données biologiques brutes générées par les technologies à haut débit sont une source d'informations cruciales pour étudier et comprendre le vivant. Cependant, sans une étape d'ajout d'informations appelée "annotation", ces séquences brutes sont difficiles à exploiter et parfois même superflues. De plus, le rythme d'annotation des données n'est pas le même que celui du séquençage des génomes. Par conséquent, l'annotation des génomes est un des défis majeurs en bio-informatique et se scinde en deux parties : i) l'annotation structurale consiste à localiser les éléments importants du génome tels que les gènes, les éléments régulateurs, les promoteurs, etc. puis à identifier leur structure interne, notamment celle des gènes codant pour des protéines . Ensuite, ii) l'annotation fonctionnelle consiste à définir la fonction de chaque élément localisé.

Lors de ma thèse, je me suis intéressé à l'amélioration de l'annotation des génomes eucaryotes en me focalisant principalement sur la prédiction des sites d'épissage, frontières entre les régions exoniques et introniques. J'ai mis en place une nouvelle stratégie d'annotation en construisant un jeu de données original spécifiquement établi pour mettre en avant l'importance de la qualité des données. J'ai ensuite exploité la puissance des algorithmes de *deep learning* et d'évolution artificielle afin d'obtenir des performances de prédiction élevées.

Le premier objectif de la thèse était de mettre en évidence les difficultés rencontrées lors de l'annotation des génomes eucaryotes. Pour cela, nous avons réalisé une étude comparative de cinq programmes de prédiction de gènes couramment utilisés. Afin d'évaluer leurs performances, nous avons élaboré un jeu de données de référence appelé G3PO (*Gene and Protein Prediction PrOgrams*). G3PO a été spécifiquement conçu pour être représentatif du monde vivant et inclut donc des séquences d'une large gamme d'organismes eucaryotes allant de l'Homme aux protistes. De plus, nous avons porté une attention particulière sur la qualité des données que nous avons incluses en créant deux catégories, l'une comprenant des séquences sans erreur, l'autre des séquences avec au moins une erreur. En utilisant ce jeu de référence, nous avons mis en avant les forces et les faiblesses de chaque programme selon différents scénarios (complexité de la carte exonique, taille du gène, etc.). Chaque programme de l'étude a révélé des défauts, attestant que la problématique d'annotation des gènes codant pour des protéines n'est que partiellement résolue malgré 30 ans de recherche. Les erreurs engendrées lors des prédictions peuvent être induites par le modèle en lui-même ou alors par un biais lié aux données d'entraînement potentiellement erronées. Une des causes principales d'erreurs est la présence de nucléotides indéterminés (notés 'N') reflétant des problèmes lors

du séquençage ou de l'assemblage. Une autre cause est liée aux caractéristiques inhérentes des gènes telles que leur longueur atypique, leur complexité structurale ou le chevauchement de deux gènes.

Dans une deuxième étude, nous avons analysé plus en profondeur les causes potentielles des erreurs chez une dizaine de primates, qui ont induit la prédiction de régions protéiques incorrectes. Plusieurs facteurs sont impliqués tels que : i) la présence de nucléotides indéterminés comme précisé précédemment, ii) de multiples petits exons de taille peu conventionnelle, iii) des décalages du cadre de lecture, iv) la prise en compte d'un mauvais site d'épissage ou iv) la présence d'un site non-canonique non détecté. Ces erreurs ont été identifiées sur les protéomes de primates issus de banques de données publiques. Par conséquent, la manipulation de ces données peut avoir un impact néfaste sur les expériences réalisées en aval.

Annotation des sites d'épissage multi-espèces par des approches d'IA, basées sur des données de haute qualité

Une des principales étapes de l'annotation structurale consiste à déterminer avec précision l'architecture interne des gènes codant pour des protéines en mettant en évidence les jonctions entre les éléments qui les composent, à savoir les exons et les introns. Ces jonctions sont appelées des sites d'épissage (SE) et inclut le site donneur (5') et le site accepteur (3'). Ils sont caractérisés par la présence d'un dinucléotide GT (donneur) et AG (accepteur) qui sont intégrés dans un motif plus divergent d'une dizaine de nucléotides. Aujourd'hui, de nombreux algorithmes d'IA se focalisent sur la prédiction des SE. Malheureusement, le manque de données d'organismes non-modèles restreint ces programmes à l'analyse d'organismes modèles bien étudiés uniquement, pour lesquels de nombreuses données expérimentales sont disponibles. De plus, les erreurs de prédiction sont encore trop présentes, souvent liées à la qualité des données utilisées lors du processus d'apprentissage.

Nous avons donc décidé de nous orienter vers l'amélioration de la prédiction des gènes eucaryotes codant pour des protéines, avec un nouveau programme d'identification des SE multi-espèces appelé Spliceator. Spliceator est basé sur un réseau de neurones convolutif, appartenant à la catégorie des algorithmes de *deep learning*. Il utilise comme données d'entrée des séquences génomiques afin d'en extraire des caractéristiques sous-jacentes. Notre stratégie a été de transformer ces séquences génomiques en un vecteur linéaire que le modèle analyse pour localiser des régions (ou caractéristiques) déterminantes pour la reconnaissance de SE.

Pour entraîner notre modèle, nous avons exploité nos connaissances et les résultats des deux premières études, ainsi que le jeu de données G3PO. En effet, comme précisé antérieurement, G3PO est constitué de deux catégories de séquences, une avec erreurs et une autre sans erreurs. Ainsi, nous avons mis en évidence l'impact de la qualité des données lors du processus d'apprentissage, mais nous avons également montré qu'un modèle entraîné sur des données de qualité multi-espèces permet d'obtenir des performances équivalentes voire supérieures aux modèles espèce-spécifique, dont l'Homme. Ces résultats nous ont permis de mettre en œuvre un programme de prédiction des SE capable d'égaliser, voire de surpasser, les meilleurs programmes de prédiction de SE actuels, mais également de mettre en évidence l'universalité du mécanisme d'épissage.

En parallèle, nous avons développé une méthode complémentaire nommée SpliceSLEIA, basée sur la PG à l'aide de la plateforme EASEA, qui permet de mettre en œuvre des algorithmes évolutionnaires multi-parallélisés. L'idée de base est de faire évoluer des automates à états finis sous forme d'expressions régulières, afin qu'ils soient capables de reconnaître des SE sur une séquence génomique. Les opérateurs évolutionnaires (mutation, croisement et sélection) permettent à ce type d'algorithme de concevoir des solutions originales que l'Homme n'a pas (ou n'aurait pu) inventées. Cette méthode, encore à un stade prototypique, nous a déjà permis d'obtenir des résultats prometteurs, engendrant un engouement supplémentaire pour ce type d'approche faiblement exploitée en biologie. Ces résultats pourraient être améliorés avec la parallélisation sur une architecture GPU, ce qui permettra de réduire significativement les temps de calcul nécessaires. Cette mise en œuvre sur GPU permettra également d'établir une population de solutions initiales plus importante engendrant un espace de recherche plus grand et donc un spectre plus large de solutions possibles. De plus, nous prévoyons de modifier notre méthode de sélection des meilleures solutions en appliquant d'autres fonctions d'évaluation.

Conclusions et perspectives

Bien que les avancées des technologies de séquençage de l'ARN permettent actuellement d'avoir des informations précises sur la structure et l'expression des ARN messager, il n'est pas encore évident d'obtenir l'ensemble de *l'isoformome* (*i.e.* l'ensemble de toutes les isoformes, à tous les âges et pour tous les tissus d'un organisme). Par conséquent, l'utilisation d'un programme de prédiction de SE multi-espèces permettra d'obtenir des informations sur les ARN messagers et leurs isoformes éventuelles non détectées

expérimentalement, ou bien pourra être utilisé pour des projets de séquençage de génomes sans données de transcriptomiques. Au cours de l'évolution, le processus d'épissage a été conservé et Spliceator est l'un des premiers programmes permettant de déterminer des sites d'épissages au sein de gènes codant pour des protéines d'espèces eucaryotes modèles et non-modèles en tirant parti de ces caractéristiques universelles préservées. Grâce à l'entraînement avec G3PO, un jeu de données de haute qualité spécialement conçu pour ce projet, nous obtenons des résultats qui surpassent les meilleurs algorithmes de prédiction de sites d'épissage actuels.

L'ajout de données supplémentaires issues d'autres espèces justifie d'une amélioration des performances de Spliceator. Finalement, nous envisageons de fusionner les approches de *deep learning* et de PG pour obtenir des résultats complémentaires et augmenter la précision des prédictions. Cette nouvelle stratégie d'annotations des génomes s'inscrit d'une part dans une approche qualitative, où l'amélioration des prédictions à partir des génomes est le point névralgique, et d'autre part dans une approche quantitative, afin de réduire l'écart entre la production des données annotées et des données brutes, notamment pour des organismes non-modèles. De plus, nous avons comme ambition d'exploiter les résultats de ce projet pour des études de médecine translationnelle, où nous pourrions identifier des variants d'épissage responsables de pathologies, telles que des cancers ou des maladies génétiques rares.

THESIS SUMMARY

The different techniques of artificial intelligence

Artificial intelligence (AI) incorporates many algorithms, including expert systems, machine learning algorithms such as decision trees or neural networks, or evolutionary algorithms. In this thesis, I focused on the last two approaches, which are complementary, in order to develop a new strategy for eukaryotic genome annotation.

For decades, machine learning algorithms have been exploited in many different fields. They initially focused on learning from input data, and then realized predictions on new unknown data. Different learning methods exist, for instance if the input data are labeled, *i.e.* we know their category and therefore the result we should obtain, this approach is called "supervised learning". It is commonly used for regression or classification tasks, for example to classify a genetic variation as "pathogenic" or "benign". In contrast, if the data are not labeled, this method is called "unsupervised learning", and the program tries to identify patterns, elements or structures underlying the input data. For example, groups of patients can be identified on the basis of genomic correlations.

These two methods are among the most widespread and many algorithms have been developed. The unsupervised learning methods include clustering algorithms such as K-means, DBSCAN or principal component analysis. Supervised learning methods include linear or logistic regressions, Support Vector Machines (SVM), decision trees, k-nearest neighbors or artificial neural networks. Currently, artificial neural networks, and especially deep neural networks, are probably the most commonly used. Artificial neural networks are a bio-inspiration of mammalian brain neurons. The central unit of these algorithms is an artificial neuron interconnected with its counterparts. The learning process consists of modifying the weights associated with each neuron in order to reduce the error. This process is performed automatically, based on the data.

Another AI field that is also inspired by biology is artificial evolution. Indeed, artificial evolution reproduces several mechanisms of evolution, such as the crossover between two individuals, mutations and the selection of the best individuals, in order to optimize a solution. There are several artificial evolution algorithms, such as evolutionary strategy, genetic algorithms or genetic programming. Among the artificial evolution algorithms, genetic programming is part of a strategy of learning from data to solve problems.

Finally, data is the cornerstone of these methods and the advent of Big Data and greater computing power have allowed these algorithms to demonstrate their performance.

AI and Big Data

Big Data has opened new opportunities and revolutionized our world view. For example, new digital technologies are now producing an astronomical amount of data. The resulting 'datasphere' is daily expanding, generating new challenges to store, process and exploit this heterogeneous data. The challenges can be defined by what is called the "3V" of Big Data: Volume, Velocity and Variety. In addition, issues related to privacy and data protection, ethics or peer-to-peer exchange (in medicine for example) have been raised and in 2016, the General Data Protection Regulation was introduced to protect personal data symbolizing its value and fragility.

However, Big Data does not have only defects, it has opened the way to diverse applications, in multiple fields and especially in AI. In the field of biology, the new sequencing technologies for example have allowed the acquisition of huge amounts of raw data. Thanks to these technologies, major advances have been made, such as the sequencing of the human genome, which was the foundation of the "omics" era. Since then, many AI-based algorithms exploiting multi-omics biological data have emerged, such as genome annotation programs, pathogen variants, genes or splice site identification. Thus, biological data is the perfect example of the "3V" representation of Big Data.

Annotation of eukaryotic genomes

The raw biological data generated by high-throughput technologies are a crucial source of information for studying and understanding living organisms. However, without a step called "annotation", these raw sequences are difficult to exploit and sometimes even useless. Moreover, the pace of annotation is not the same as genome sequencing. Currently, genome annotation is one of the major challenges in bioinformatics and can be divided into two parts. i) Structural annotation consists in locating the important elements of the genome such as genes, regulatory elements, promoters, etc. and then identifying their internal structure, especially for protein-coding genes. Then, ii) functional annotation consists of defining the function of each localized element.

During my thesis, I was interested in improving the annotation of eukaryotic genomes, focusing mainly on the prediction of splice sites, the boundaries between exonic and intronic regions. I implemented a new annotation strategy by building an original dataset specifically designed to emphasize the importance of data quality. I then exploited the power of deep learning and artificial evolution algorithms to achieve high prediction performance.

The first objective of the thesis aimed to highlight the difficulties encountered when annotating eukaryotic genomes. For this purpose, we conducted a comparative study of five commonly used gene prediction programs. In order to evaluate their performance, we developed a reference dataset called G3PO (Gene and Protein Prediction PrOgrams). G3PO was specifically designed to be representative of the living world and includes sequences from a wide range of eukaryotic organisms from humans to protists. In addition, we paid particular attention to the quality of the data we included by creating two categories, including either sequences with no errors or sequences with at least one error. Using this benchmark set, we highlighted the strengths and weaknesses of each program under different scenarios (exonic map complexity, gene size, etc.). Each program in the study revealed flaws, attesting to the fact that the problem of annotating protein-coding genes is only partially solved despite 30 years of research. The errors generated during prediction can be induced by the model itself or by a bias linked to the potentially erroneous training data. One of the main causes of these errors is the presence of indetermined nucleotides (noted 'N') reflecting problems during sequencing or assembly. Another cause is related to the inherent characteristics of the genes, such as their atypical length, their structural complexity or the overlap of two genes.

In a second study, we further analyzed the potential causes of errors in twelve primates, resulting in the prediction of incorrect protein regions. We identified several factors involved such as: i) the presence of indeterminate nucleotides as previously mentioned, ii) multiple small exons of unconventional size, iii) ORF frameshifts, iv) the inclusion of a wrong splice site or iv) the presence of an undetected non-canonical site. These errors were identified on primate proteomes from public databases. Therefore, manipulation of these data can have a detrimental impact on downstream experiments.

Multi-species splice site annotation by AI approaches based on high quality data

One of the main steps in structural annotation is to accurately determine the internal architecture of protein-coding genes by highlighting the junctions between their component parts, namely exons and introns. These junctions are called splice sites and include the donor

site (5') and the acceptor site (3'). They are characterized by the presence of a GT (donor) and AG (acceptor) dinucleotide that are integrated in a more divergent motif of about ten nucleotides. Today, many AI algorithms focus on splice site prediction. Unfortunately, the lack of non-model organism data has restricted these programs to the analysis of well-studied model organisms for which extensive experimental data are available. In addition, prediction errors are still frequent, often related to the quality of the data used in the learning process.

We therefore decided to focus on improving the prediction of eukaryotic protein-coding genes with a new multi-species splice site identification program called Spliceator. Spliceator is based on a convolutional neural network, belonging to the category of deep learning algorithms, and uses raw genomic sequences as input data, in order to extract underlying features. Our strategy was to transform these genomic sequences into a linear vector, so that the model could analyze and localize regions (or features) that are crucial for splice site recognition. To train our model, we exploited our knowledge and results from the first two studies, as well as the G3PO dataset. Indeed, as previously mentioned, G3PO consists of two categories of sequences, one with errors and another without errors. Thus, we highlighted the impact of the data quality during the learning process, but we also showed that a model trained on multi-species quality data allows to obtain equivalent or even superior performance to species-specific models, including humans. These results allowed us to implement a splice site prediction program able to outperform the best current splice site prediction programs, but also to highlight the universality of the splicing mechanism.

In parallel, we developed a complementary method named SpliceSLEIA, based on genetic programming using the EASEA platform, which can be used to implement multi-parallelized evolutionary algorithms. The idea is to evolve finite state automata as regular expressions, so that they are able to recognize splice sites on a genomic sequence. The evolutionary operators (mutation, crossover and selection) allow this type of algorithm to design original solutions that humans have not (or could not) invent. This method, still at a prototypical stage, has already allowed us to obtain promising results, and should generate additional interest in this type of approach, which has not been widely used in biology. The results could be improved by parallelization of the method on a GPU architecture, which will significantly reduce the necessary computation time. This GPU implementation will also allow us to establish a larger population of initial solutions, generating a larger search space and thus a larger panel of possible solutions. Moreover, we plan to modify our method of selecting the best solutions by applying other evaluation functions.

Conclusions and perspectives

Although advances in RNA sequencing technologies currently allow for more precise information on the structure and expression of some messenger RNA, it is not yet obvious how to obtain entire isoformoms (*i.e.* the set of all isoforms, at all ages and for all tissues of an organism). A multi-species splice site prediction program could be used to obtain information on messenger RNA and their possible isoforms not detected experimentally or could be used for genome sequencing projects without transcriptomic data. During the evolution, the splicing process has been conserved and Spliceator is one of the first programs to determine splice sites within protein-coding genes of both model and non-model eukaryotic species by taking advantage of these conserved universal features. By training with G3PO, a high-quality dataset specifically designed for this project, we achieved results that outperform the best current splice site prediction algorithms.

Adding more data from other species should improve the performances of Spliceator. Finally, we plan to merge the deep learning and genetic programming approaches to obtain complementary results and increase prediction accuracy. This new strategy for genome annotation is part of a qualitative approach, where the improvement of genome predictions is the main focus, and part of a quantitative approach, in order to reduce the gap between the production of raw and annotated data, especially for non-model organisms. Furthermore, we aim to exploit the results of this project for translational medicine studies, where we want to identify splice variants responsible for pathologies, such as cancers or rare genetic diseases.

LISTE DES FIGURES

LISTE DES TABLEAUX

ABRÉVIATIONS

AA	Acide Aminé
ACP	Analyse en Composantes Principales
ADN	Acide DésoxyriboNucléique
ADNc	ADN complémentaire
AE	Algorithme Évolutionnaire
AJAX	*Asynchronous JavaScript and XML*
ARN	Acide RiboNucléique
AS	*All Sequences*
BAM	*Binary Alignement Map*
BBS	*Bardet-Biedl Syndrom*
BLAST	*Basic Local Alignment Search Tool*
CASP	*Critical Assessment of Structure Prediction*
CDS	*Coding DNA Sequence*
CNN	*Convolutional Neural Network*
CPU	*Central Processing Unit*
EASEA	*EAsy Specification of Evolutionary Algorithms*
ENCODE	*ENCyclopedia Of Dna Elements*
ESE	*Exonic Splicing Enhancer*
ESS	*Exonic Splicing Silencer*
EST	*Expressed Sequence Tag*
FN	Faux Négatif
FP	Faux Positif
G3PO	*Gene and Protein Prediction PrOgrams*
GAN	*Generative Adversarial Network*
GAR	*Gene ARchitecture*
GFF	*General Feature Format*
GPU	*Graphics Processing Units*
GS	*Gold Standard*
HMM	*Hidden Markov Model*
HS^{3}D	*Homo Sapiens Splice Sites Dataset*
IA	Intelligence Artificielle
ISE	*Intronic Splicing Enhancer*
ISS	*Intronic Splicing Silencer*
LECA	*Last Eukaryotic Common Ancestor*
LINE	*Long Interspersed Nuclear Elements*
LSTM	*Long Short-Term Memory*
MNIST	*Mixed National Institute of Standards and Technology*
MSA	*Multiple Sequence Alignment*
ORF	*Open Reading Frame*
pb	paire de bases
PEAR	*Protein Elements with Aberrant Regions*
PG	Programmation Génétique
PMC	Perceptron Multi-Couches

Protéine SR	Protéine Sérine-Arginine
regex	Expression régulière
ReLU	*Rectified Linear Units*
RGPD	Règlement Général sur la Protection des Données
RN	Réseau de neurones
RNA-seq	Séquençage de l'ARN
RNN	*Recurrent Neural Network*
SE	Site d'Épissage
SINE	*Short Interspersed Nuclear Elements*
snRNA	*small nuclear RNA*
snRNP	*small nuclear RiboNucleoProtéine*
SpliceSLEIA	*Splice Site Localization by EvolutIonnary Algorithm*
SVM	*Support Vector Machines*
TSS	*Transcription Start Site*
UTR	*UnTranslated Region*
VN	Vrai Négatif
VP	Vrai Positif

INTRODUCTION

Chapitre 1 - L'intelligence artificielle d'hier à aujourd'hui

1.1 La genèse des machines "intelligentes"

De tout temps, l'être humain a imaginé des créatures façonnées à son image, tant sur l'aspect physique que psychique. Cette vision anthropocentriste est le reflet de nombreuses interrogations sur nos aptitudes, notre existence et la fatalité de notre destin. Au travers de ces évasions de l'esprit, l'Homme a inventé le concept de la "vie artificielle", qui selon Christopher Langton, pionnier dans ce domaine, se définit par : "un système construit par l'Homme plutôt que par la nature, qui présente des comportements caractéristiques des systèmes vivants naturels" (Langton, 1997). En conséquence, ces systèmes sont dotés de capacités physiques remarquables, mais également de facultés de jugement et de réflexion. Elles s'apparentent à une intelligence "artificielle", généralement supérieure à celle de l'Homme, qui elle, est contrainte par des limites biologiques.

Les préludes d'une trace d'intelligence "artificielle" sont apparus dans la Grèce antique, où les histoires s'entremêlent avec la mythologie. Hésiode (VIII^{e} siècle av. J-C), poète grec, mentionne l'existence d'un géant nommé Talos, capable d'avoir des émotions. Il aurait été forgé par Héphaïstos, dieu du feu et de la métallurgie et lui aurait insufflé la vie au travers d'un conduit allant de sa tête à son talon où circulait le liquide des dieux, l'Ichor. Par la suite, de nombreuses inventions tentant de simuler une intelligence ont traversé les siècles, fascinant le monde, telles que le chevalier de Leonardo da Vinci (XVI^{e} siècle) ou encore le "Turc mécanique"[1] de l'Autrichien Wolfgang Von Kempelen (XVIII^{e} siècle) (Levitt, 2000). Ce dernier est constitué d'un mannequin jouant aux échecs et d'un meuble composé du mécanisme qui déplace les pièces. Cependant, le meuble était construit de telle sorte à pouvoir dissimuler un homme, qui maniait le mannequin et donc qui jouait à sa place.

La notion de machines "intelligentes" ou "programmables" apparaît réellement avec Basile Bouchon en 1725, quand il met au point un métier à tisser semi-automatique (Davis and Davis, 2005). La machine est programmée grâce à un ruban perforé qui contrôle l'aiguille en

1 Le "Turc" a fait le tour du monde, et a joué de nombreuses parties d'échecs qu'il a bien souvent gagnées, notamment contre de célèbres adversaires tels que Benjamin Franklin ou Napoléon Bonaparte.

fonction de la présence ou de l'absence de trous dans le ruban. Il s'agit d'une des premières machines qui traite de l'information stockée. Les années suivantes voient apparaître des améliorations importantes, comme le remplacement du ruban par une série de cartes perforées, qui sont ensuite rendues indépendantes (elles ne sont plus intégrées directement à la machine) par Joseph Marie Jacquard (Davis and Davis, 2005). Ces travaux ont inspiré l'ingénieur Charles Babbage, qui a utilisé les cartes perforées pour lire des données avec sa machine à calculer automatique, faisant de cette dernière une machine programmable (Qin *et al.*, 2018). Babbage est ainsi considéré comme l'un des précurseurs de l'informatique.

En 1880, Herman Hollerith invente la mécanographie, qui consiste à exploiter des machines électromécaniques dans le but de réaliser des calculs et traiter des informations à l'aide de cartes perforées. Il fondera plus tard la société *International Business Machines Corporation* (Blodgett and Schultz, 1969), plus connue sous le sigle IBM. La mécanographie a permis à Konrad Zuse en 1941 de concevoir un calculateur électromécanique, le Zuse 3 (Rojas, 1997), premier calculateur automatique programmable stockant les informations sur des cartes perforées. En 1944, aux Etats-Unis, le Mark I est élaboré (Cohen, 2003), une machine analogue au Zuse 3.

Une évolution majeure a été le passage des calculateurs électromécaniques aux calculateurs électroniques, avec le Atanasoff-Berry Computer (Gustafson, 2000), le Colossus en 1944 (Copeland, 2004), puis en 1945, le *Electronic Numerical Integrator And Computer*, plus communément connu sous l'acronyme ENIAC (Burks, 1947). Ce dernier possède des caractéristiques architecturales impressionnantes (30 tonnes et plus de 18 000 tubes à vide, pour une fréquence d'horloge de 100 kHz) ce qui fait de lui le calculateur le plus rapide de son époque.

S'il y a bien un nom à retenir dans l'histoire de l'informatique, c'est celui du génie britannique Alan Turing. En 1936, il publie un article "*on computable numbers, with an application to the entscheidungsproblem*" (Turing, 1937), où il démontre que l'emploi d'opérations élémentaires telles que l'addition, la soustraction ou la multiplication pourrait résoudre tous les problèmes algorithmiques sous forme binaire. Il met ainsi en place le concept de calculateur programmable universel. Dans ce sens, il conçoit virtuellement la "machine de Turing", fondement des ordinateurs modernes. Pendant la deuxième guerre mondiale, de nombreuses avancées ont été réalisées, notamment l'architecture de Von Neumann (1945) ou le transistor (1947), qui ont augmenté la vitesse et la puissance des calculateurs. Ainsi, le premier "vrai" ordinateur basé sur une architecture de Von Neumann et donc intégrant le code

du programme dans sa mémoire, est le Manchester Baby (ou *Small-Scale Experimental Machine*) (Burton, 1998), créé en 1948, nous amenant à l'ère de l'informatique moderne.

1.2 L'avènement de l'intelligence artificielle

Au sortir de la guerre, Turing se pose une nouvelle question : "*Can machines think*?" (Turing, 1950). Il se demande si les machines peuvent avoir une intelligence semblable à celle de l'homme et met en place le célèbre "test de Turing". Ainsi, en associant le concept de l'intelligence humaine avec celle d'une machine, Alan Turing pose les fondements d'une des inventions les plus remarquables de l'histoire de l'humanité après la roue : l'intelligence artificielle.

Les travaux précurseurs de Turing ont ravivé la flamme d'une intelligence artificielle ancrée depuis longtemps dans l'esprit des Hommes. L'idée d'avoir des machines aussi intelligentes, voire plus intelligentes, a toujours été présente. Ainsi, des chercheurs se sont focalisés sur l'analogie entre les capacités cognitives, le cerveau et les neurones biologiques, et les machines. En 1943, le premier modèle mathématique d'un neurone biologique, appelé neurone formel, est conceptualisé (McCulloch and Pitts, 1943). En 1951, Marvin Minsky construit le SNARC (*Stochastic Neural Analog Reinforcement Calculator*), un réseau de neurones électroniques, *i.e.* sous la forme d'une machine, composée de 40 neurones.

Un domaine où l'intelligence artificielle est précurseur est le jeu, notamment les échecs, comme nous l'avons vu avec le "Turc mécanique" de Kempelen. Ainsi, Alan Turing avait déjà imaginé un programme de jeu d'échecs (avant même que les ordinateurs ne soient construits), appelé *Turochamp* (Björk, 2013). Il faudra attendre les travaux de Dietrich Prinz pour qu'un programme de jeu d'échecs soit exécuté sur un ordinateur (le Ferranti Mark I, descendant du Manchester baby), suivie par le développement du jeu de dames de Christopher Strachey en 1952. En s'inspirant des travaux de Strachey, Arthur Samuel met en place une nouvelle version du jeu de dames, notamment en modifiant l'intelligence artificielle et son mode d'apprentissage. De plus, Samuel inventa le terme "*machine learning*", que nous connaissons en français sous le nom d'apprentissage automatique (Samuel, 1959). Parallèlement, en 1954, des travaux ont été effectués sur la possibilité de simuler l'évolution sur ordinateur pour exploiter les forces des mécanismes de l'évolution dans le but de résoudre des problèmes (Barricelli, 1957). Ces recherches marquent le début d'une nouvelle branche de l'intelligence artificielle appelée "évolution artificielle".

Le terme "Intelligence Artificielle" ou "IA" est proposé pour la première fois en 1956 par John McCarthy lors d'une conférence au Dartmouth College (Moor, 2006). Les protagonistes se sont notamment intéressés à la manière d'exploiter le potentiel des calculateurs (ordinateurs) automatiques et de comment les programmer. Ils se sont accordés sur le fait que, si une machine peut réaliser une tâche intelligente, alors un ordinateur programmé peut simuler la machine et réaliser cette tâche (Moor, 2006).

Les premiers travaux sous la dénomination d'IA sont ceux de Alan Newell, avec la création du langage IPL (*Information Processing Language*) et l'implémentation de deux programmes codés en IPL : le "*general problem solver*" (Newell, 1958) et le "*logic theory machine*", ce dernier est souvent défini comme le premier programme informatique basé sur une IA. Ces programmes devaient résoudre des problèmes tels que "les tours de Hanoï" ou proposer des solutions de théorèmes.

En 1958, John McCarthy invente le langage de programmation Lisp (McCarthy, 1960), qui deviendra le langage dédié à l'IA dans les années 70-80. Il permet de manipuler des éléments, comme des symboles ainsi que des listes, sous forme d'expressions. Après les travaux préliminaires de McCulloch, Pitts et Minsky sur les réseaux de neurones, Frank Rosenblatt développe un des algorithmes phares de l'IA, le perceptron (Rosenblatt, 1958). Les perceptrons obtiennent généralement de bonnes performances si le problème est linéairement séparable et ont été améliorés par la suite, en ajoutant plus de neurones ou plus de couches.

En 1963, James Morgan met au point un algorithme basé sur des arbres de décision (Morgan and Sonquist, 1963). Ces derniers sont de puissants outils pour l'aide à la décision et sont encore largement utilisés aujourd'hui en médecine et en biologie par exemple, car ils sont automatisables et explicables (Gunning and Aha, 2019).

Un an plus tard, le premier programme "interlocuteur" est développé, nommé ELIZA (Weizenbaum, 1966). Le programme simule un psychothérapeute et se base sur les réponses du patient, c'est-à-dire qu'il récupère des mots clés pour faire des associations afin de formuler une réponse cohérente, souvent à partir de modèles. Lorsque ELIZA a été confrontée au test de Turing, le programme a réussi à tromper quelques humains, par conséquent c'est le premier programme "validant" cet exercice (Haenlein and Kaplan, 2019). Cependant, certaines questions, souvent incohérentes, trahissent rapidement ELIZA qui n'est pas capable de prendre en compte la sémantique de la langue.

L'année suivante, les premiers algorithmes de stratégie d'évolution (*Evolutionary Strategy*) (Rechenberg, 1965) sont utilisés pour résoudre des problèmes d'optimisation. Ils ont la particularité d'utiliser un opérateur de mutation afin de visiter un plus large espace de

recherche. Puis un an plus tard, Lawrence Fogel met au point les algorithmes de programmation évolutionnaire (*Evolutionary Programming*) permettant l'évolution d'automates à états finis (Fogel and Fogel, 1996).

Malgré ces avancées, la recherche sur l'IA stagne, les résultats obtenus n'atteignent pas les attentes des financeurs et la puissance des machines limite les performances, au point que beaucoup de travaux sur ce sujet ont été abandonnés pendant plus de 10 ans, marquant le début du premier hiver de l'IA en 1974. Parmi les rares travaux entrepris durant cette période, certains ont quand même eu un impact important tel que ceux portant sur l'algorithme de rétropropagation (Werbos, 1994). Cet algorithme est un des fondements de l'IA actuelle. En 1975, un nouvel algorithme basé sur l'EA est développé, appelé algorithme génétique (Holland, 1992). Ce sont les algorithmes d'EA les plus utilisés. Ils incluent des opérateurs de mutation, mais également des événements de recombinaison. Les systèmes experts ont été développés parallèlement la même année, afin de reproduire l'expertise d'un domaine spécifique tel que le système expert MYCIN (capable de détecter des maladies du sang provoquées par une bactérie) (Buchanan, 1984). Ils sont constitués d'une base de faits, d'une base de règles et d'un moteur d'inférence. L'avènement des systèmes experts va peu à peu redorer l'image de l'IA, aboutissant à la fin de l'hiver en 1980. Ainsi, on observe un accroissement du nombre de programmes voués à la résolution d'objectifs précis et dotés de hautes performances (Fuchs *et al.*, 1999), relançant la recherche en IA, avec notamment les réseaux de neurones et le connexionnisme (Hopfield, 1982).

En dépit de l'engouement pour ces nouveaux algorithmes, la recherche en IA ralentit à nouveau, la plongeant dans les abîmes de son deuxième hiver. En dépit de cela, Geoffrey Hinton et Yann LeCun, pionniers dans leur domaine, effectuent des travaux sur l'algorithme de rétropropagation (Rumelhart *et al.*, 1986) et l'améliorent (LeCun, 1985). Au début des années 90, John Koza s'inspire des travaux de Cramer (Cramer, 1985) pour conceptualiser les algorithmes de programmation génétique (Koza, 1994). La spécificité de ces algorithmes est d'utiliser une structure d'arbre pour représenter les données.

Les progrès technologiques tels que les capacités de calcul et de stockage des ordinateurs ainsi que l'arrivée des premiers jeux de données massifs ont relancé la recherche en IA, mettant fin à ce second hiver. Yann LeCun remet sous le feu des projecteurs les réseaux de neurones. Il établit les premiers réseaux de neurones convolutifs pour réaliser des tâches de reconnaissance de caractères (Lecun and Bengio, 1995). Il a également travaillé sur la mise en place du célèbre jeu de données MNIST (Lecun *et al.*, 1998) qui aujourd'hui est l'archétype du tutoriel en IA. Vladimir Vapnik a élaboré un algorithme pour résoudre des problèmes de

classification et de régression appelée les machines à vecteurs de support (SVM) (Cortes and Vapnik, 1995). Les SVM ont l'avantage de résoudre des problèmes non-linéaires comme le problème du XOR. Les travaux sur les algorithmes évolutionnaires continuent à se développer, avec un nouvel algorithme performant appelé évolution différentielle (Storn and Price, 1997).

En 1996, un événement hors du commun va bousculer la communauté scientifique : le duel d'échecs entre *Deep Blue* et Garry Kasparov. *Deep Blue* est un superordinateur construit par IBM spécialisé dans les parties d'échecs (Campbell *et al.*, 2002). Son algorithme est basé sur une recherche arborescente et ce sont ses capacités de calculs gigantesques (32 cœurs, 1Gb de RAM et 4Gb de ROM) qui permettent à *Deep Blue* de vérifier plus de 200 millions de positions par seconde. Lors du match contre Kasparov, l'ordinateur bat le champion du monde dès la première partie[2], démontrant la puissance de l'IA et la propulsant vers une ère nouvelle.

Cependant le tournant de l'IA est situé au début des années 2010. Plusieurs facteurs ont permis son avènement, notamment l'explosion des performances de calcul des ordinateurs qui suivent la loi de Moore (*i.e.* le nombre de transistors double tous les deux ans) (Moore, 1965), le développement des *Graphics Processing Units* (GPU) (Cano, 2018), les masses de données (*Big Data*) et des algorithmes plus performants. L'IA s'est ainsi vue dotée d'une nouvelle branche, l'apprentissage profond (ou *deep learning*). Ces algorithmes ont permis l'obtention de nombreux succès dans de multiples domaines tels que la reconnaissance de signaux ou d'images ou encore le jeu. Au début des années 2010, une autre IA d'IBM, nommée Watson, bat les champions du jeu *Jeopardy!* (Lally and Fodor, 2011). En 2012, Google sort son assistant intelligent Google Now, cinq ans après le programme de commande vocale Siri de la firme Apple, puis en 2014 c'est au tour de Microsoft, qui développe Cortana. En 2016, AlphaGo, une IA spécialisée dans le jeu de GO, bat les meilleurs joueurs de GO. En 2017, le programme Libratus gagne un tournoi de poker de type Texas Hold'em. En 2019, l'IA AlphaStar bat des joueurs humains sur StarCraft II. Un des derniers exploits en date, qui a été très médiatisé, est celui de AlphaFold, qui a obtenu des performances historiques (92,4% de précision) au concours de prédiction des structures tridimensionnelles de protéines CASP14 (Jumper *et al.*, 2021).

1.3 Le paysage de l'intelligence artificielle

2 La partie fut finalement remportée par Kasparov 4 à 2 et ce n'est que l'année suivante que *Deep Blue* parvient à battre Kasparov sur un score de 3 ½ contre 2 ½.

L'IA a été développée pour seconder l'Homme et surpasse déjà largement certaines de ses capacités telles que la mémoire ou la puissance et la vitesse de calcul. L'IA est de plus en plus présente dans de nombreux domaines comme l'ingénierie, l'art ou encore les jeux vidéo. Par ses capacités, elle est jugée plus fiable, comme dans le domaine de la finance et du trading (Bahrammirzaee, 2010), ou plus impartiale dans certains jugements comme l'assistance vidéo à l'arbitrage (VAR) (Spitz *et al.*, 2021). Cependant, pour certains métiers, l'IA ne secondera pas les humains, mais les remplacera, probablement comme dans certains métiers impliquant la conduite de véhicules. Quant au domaine médical, l'IA est déjà présente dans les salles d'opération (Panesar *et al.*, 2019) ou pour le diagnostic de certains cancers. Mais son impact sera probablement encore plus important avec les objets connectés (*Internet of Things* (IoT)), en prédisant par exemple quelques heures avant, la survenue d'un AVC. Les données seront directement récoltées par les objets connectés, alarmant ainsi les services médicaux compétents. Ce sont des spéculations, mais au vu des progrès actuels, cela ne serait pas étonnant que le médical 3.0 (Galland, 2020) arrive prochainement. L'IA s'est ainsi démocratisée et est devenue accessible à tous. Cependant des chercheurs se sont inquiétés de son potentiel ainsi que des problèmes éthiques que l'on pourrait rencontrer. Bien souvent, l'IA est vue comme une source de problèmes induisant des scénarios catastrophes. Cette idée est probablement à l'origine d'une réticence partielle de la société actuelle à accepter certains concepts ou idées en lien avec l'IA. Mais de plus en plus, l'IA est aussi vue comme quelque chose d'utile, capable d'aider l'humain dans son quotidien.

1.4 Définir l'intelligence artificielle, un problème nébuleux

L'IA est un domaine qui s'est démocratisé ces dernières années, et on a tendance à l'utiliser sans en avoir conscience, mais en réalité, qu'est-ce que concrètement l'IA ? Dans "intelligence artificielle", le mot "intelligence" symbolise les capacités cognitives exceptionnelles que devrait atteindre l'IA. Le mot "artificielle" fait référence au support exploité *i.e.* l'informatique, les machines, etc. Cependant, les chercheurs ne s'accordent pas encore sur une définition unique du terme. Selon la définition de l'encyclopédie Larousse, l'IA est :

"*L'ensemble des théories et de techniques mises en œuvre en vue de réaliser des machines capables de simuler l'intelligence humaine*".

Marvin Minsky parle de l'IA comme :

> "*La construction de programmes informatiques qui s'adonnent à des tâches qui sont, pour l'instant, accomplies de façon plus satisfaisante par des êtres humains, car elles demandent des processus mentaux de haut niveau tels que : l'apprentissage perceptuel, l'organisation de la mémoire et le raisonnement critique*".

Pour Yann LeCun, pionnier du *deep learning*, l'IA est :

> "*Un ensemble de techniques permettant à des machines d'accomplir des tâches et de résoudre des problèmes normalement réservés aux humains et à certains animaux*".

Enfin, pour Geoffrey Hinton, un des chercheurs les plus respectés et influant de ce domaine :

> "*L'IA moderne s'inspire des idées sur le fonctionnement du cerveau. [...] Au lieu de programmer l'ordinateur, vous lui montrez de nombreux exemples, il modifie les forces de connexion et apprend à produire les bonnes réponses sans que vous ayez à programmer*".

Ces définitions s'entrecoupent et survolent de manière très large l'IA. Cependant, ce sont des définitions d'experts qui mettent en avant le côté technique et ces visions diffèrent les unes des autres sans qu'aucune définition n'ait su satisfaire l'ensemble de la communauté. Définir l'IA est peut-être une tâche que seule l'IA elle-même pourrait résoudre. Il y a cependant une divergence entre deux notions, l'IA faible et l'IA forte.

L'IA forte fait référence aux concepts même d'intelligence, de réflexion, de sentiments et bien sûr d'esprit. Aujourd'hui, cette IA forte est encore utopique et inaccessible, hormis au 7^{e} art, mais certains chercheurs estiment son arrivée imminente aux alentours de 2030. Le concept d'IA faible est probablement celui qui se rapproche le plus de l'IA actuelle, c'est-à-dire une science appliquée. L'IA faible est dépourvue d'une conscience ou d'une quelconque intelligence. Elle se base sur des algorithmes et/ou des données afin de résoudre une ou plusieurs tâches bien définies. L'IA faible simule l'intelligence, mais ne l'exploite pas.

Le terme d'IA recouvre une multitude de sous-domaines qui incluent plusieurs technologies et algorithmes. Parmi les plus répandus, nous retrouvons l'apprentissage automatique, plus communément appelée *machine learning* (Baştanlar and Ozuysal, 2014), les

systèmes experts (Feigenbaum, 1992), les algorithmes évolutionnaires (AE) (Back and Schwefel, 1996) ou encore le traitement naturel du langage (*Natural Language Processing*) (Chowdhury, 2003). La figure 1 expose de manière non exhaustive le paysage de l'IA. Dans le cadre de ces travaux de thèse, j'ai eu l'occasion de mettre en pratique les approches d'évolution artificielle et de *machine learning*.

Figure 1 - *Paysage (non exhaustif) de l'intelligence artificielle. (CNN : Réseau de neurones convolutif, LSTM : Réseau de neurones récurrent).*

1.5 L'intelligence artificielle dans la culture populaire

L'IA actuelle est encore loin de l'homme-machine imaginé par les Grecs, ou du programme surpassant l'intelligence humaine et capable de réflexion, convoité par les pionniers du domaine. Cependant, lorsque Alan Turing, publie son article "*computing machinery and intelligence*" en 1950, nul n'aurait pu imaginer que 70 ans après, le monde serait régi par les ordinateurs et l'IA. Aujourd'hui nous utilisons l'IA quotidiennement et demain, nous serons probablement assistés par des robots intelligents.

Cet engouement pour une "intelligence artificielle" s'est retranscrit dans notre culture. Par exemple, de nombreuses fictions se sont inspirées de cet univers en imaginant des robots avec une conscience, très intelligents et souvent belliqueux. Dans le monde cinématographique, de nombreuses IA ont cultivé le mythe des robots tueurs (Terminator, I-Robot) ou des IA prenant le contrôle comme HAL (2001 : L'odyssée de l'espace). A l'inverse, d'autres fictions

ont mis en avant l'utilité des robots assistants comme Wall-e, Jarvis des Avengers ou encore R2D2 et C-3PO de l'univers de Star Wars.

Ce mouvement démontre un engouement pour l'IA, mais remet cependant en cause cette avancée et les risques qu'elle peut engendrer si un jour elle surpasse les capacités cognitives de l'Homme. En attendant, dans ces travaux de thèse, je me suis concentré sur des techniques de *machine learning* dans le but de résoudre des problématiques biologiques. La suite de l'introduction est organisée comme suite. Le Chapitre 2 est consacré au *machine learning* et au *deep learning*, le Chapitre 3 à l'évolution artificielle, et enfin le Chapitre 4 au *Big Data* et à l'annotation des génomes eucaryotes.

Chapitre 2 - Le *machine learning*

2.1 Les concepts du *machine learning*

Le *machine learning* est un domaine de l'IA fondé sur la capacité d'une machine à inférer les caractéristiques des données. En exploitant ces données, les algorithmes réalisent un "apprentissage" en appliquant des approches mathématiques et statistiques afin de générer un modèle (Baştanlar and Ozuysal, 2014). Les modèles sont alors capables de réaliser des tâches de prédiction et de classification sans qu'ils aient été initialement programmés pour les résoudre.

Avant d'entraîner un modèle, il est bon d'en donner une définition. Un modèle en *machine learning* est une équation mathématique qui représente les données d'apprentissage à la sortie d'un algorithme. Le modèle est entraîné avec les données d'apprentissage et bien que formé sur ces dernières, son but n'est pas de les imiter à l'identique (sinon on parle de sur-apprentissage), mais de les généraliser. Ainsi, la phase d'apprentissage consiste à ce que le modèle modifie/ajuste ses paramètres, pour qu'il généralise au mieux et obtienne de meilleures performances.

Au sein même du *machine learning*, les algorithmes sont scindés en différentes catégories relatives à leur mode d'apprentissage : l'apprentissage supervisé, non supervisé, semi-supervisé, par transfert et par renforcement.

2.1.1 Apprentissage supervisé

L'apprentissage supervisé se consacre à l'élaboration d'un modèle qui apprend à partir de données dites étiquetées. Ces dernières forment un binôme entre les données en soi, incluant des informations appelées descripteurs, caractérisant un même objet (on retrouve également le terme de *features* ou X), et les étiquettes associées (aussi appelée *labels* ou Y) qui définissent la classe ou la catégorie de chaque donnée d'entrée. Les données sont donc représentées sous la forme d'un vecteur $\{(X_1, Y_1), \dots, (X_n, Y_n)\}$ avec X pouvant être une valeur unique ou un vecteur : $(X_{i,1}, X_{i,2}, \dots, X_{i,n})$, représentant l'ensemble des descripteurs, le nombre de descripteurs *n* correspond à sa dimension. Y est un vecteur comprenant le nombre de classes :

$(1, \dots, m)$. Cette méthode permet de guider le modèle lors de son processus d'apprentissage à partir des données étiquetées. L'objectif est donc d'entraîner un modèle qui sera capable de généraliser les données d'entrée, pour ensuite réaliser des prédictions sur des données inconnues. L'apprentissage supervisé intègre communément des tâches de régression (pour des variables quantitatives) ou de classification (pour des variables qualitatives). Parmi les algorithmes d'apprentissage supervisé, on retrouve par exemple :

- les régressions linéaire et logistique,
- les arbres de décision et les forêts aléatoires,
- les machine à vecteurs de support (SVM),
- les k-plus proches voisins,
- les réseaux de neurones.

Pour illustrer l'apprentissage supervisé, prenons l'exemple des variations génétiques faux sens (*missense variation*). On dispose d'un jeu de données incluant de nombreuses variations caractérisées par plusieurs descripteurs, comme la fréquence allélique mineure ou des mesures de conservation (Chennen *et al.*, 2020). Chaque variation est étiquetée selon la classe à laquelle elle appartient ("pathogène" ou "bénigne"). Le processus d'entraînement du modèle va établir des relations entre les descripteurs des données afin de les généraliser. Une fois le modèle entraîné, on peut lui donner d'autres variants inconnus à prédire comme appartenant à la classe "bénigne" ou "pathogène".

L'apprentissage supervisé est donc dépendant des données d'entrée et plus précisément, d'un volume important de données. Cependant, l'étiquetage des données est une tâche réalisée la plupart du temps par des experts, ce qui en fait une ressource rare. Ainsi, sans une quantité de données suffisante, le modèle rencontrera des difficultés pour généraliser les données (on parle de sous-apprentissage). Les données à manipuler doivent également être cohérentes avec le problème posé et de bonne qualité.

2.1.2 Apprentissage non supervisé

L'apprentissage non supervisé, tout comme le supervisé, se base sur les données (X), cependant avec cette approche, elles ne sont pas étiquetées. Ainsi les données sont représentées par le vecteur $\{X_i, \dots, X_n\}$. Le modèle essaie alors d'extraire des structures sous-jacentes et détermine des relations entre les données, afin de générer des classes. Chaque classe intègre une partie des données ayant des similitudes, des structures ou motifs que le modèle aurait pu

identifier. De plus, l'apprentissage non supervisé nécessite aussi une quantité de données importante et donc les problématiques des petits jeux de données ou de mauvaise qualité sont similaires à celles de l'apprentissage supervisé.

Pour exemplifier l'apprentissage non supervisé, on peut imaginer un jeu de données incluant des informations sur des patients (poids, taux de glycémie, pourcentage de gonadotrophine chorionique humaine, etc.). Le modèle va alors extraire des corrélations entre les données. A partir de ces relations, l'algorithme peut établir des classes et isoler les caractéristiques importantes de chaque catégorie de patients. Il existe quatre grandes catégories d'algorithmes d'apprentissage non supervisé :

- les méthodes de regroupement (*clustering*), telles que la méthode des k-moyens (*k-means*), DBSCAN (*Density-based spatial clustering of applications with noise*), ou le regroupement hiérarchique,
- la détection d'anomalie, comme les SVM à une classe (*one-class SVM*) ou les forêts isolées (*isolation forests*),
- la réduction de dimension, comme l'analyse en composantes principales (ACP), le kernel ACP ou le t-SNE (*t-distributed stochastic neighbor embedding*),
- l'apprentissage de règles d'association, par exemple Apriori, FP-Growth (*frequent pattern-growth*), Eclat ou OPUS.

2.1.3 Apprentissage semi-supervisé

Le terme semi-supervisé fait référence à un type d'apprentissage qui mélange les concepts de l'apprentissage supervisé et non supervisé. En effet, cette catégorie d'apprentissage exploite des données non étiquetées et étiquetées, ce qui est intéressant quand ces dernières sont rares. En partant de l'axiome que si les données non étiquetées partagent des caractéristiques similaires aux données étiquetées, alors le modèle pourra extraire des structures sous-jacentes et sera capable d'améliorer ses performances. En contrepartie, si les données divergent, les performances du modèle seront diminuées (Zhu and Goldberg, 2009).

2.1.4 Apprentissage par transfert

L'apprentissage par transfert (ou *transfer learning*) est un type d'apprentissage qui se consacre à l'échange d'informations entre deux domaines étroitement liés. Cette approche

inspirée des facultés de transmission du savoir et de l'expérience humaine, cherche à transférer les connaissances d'une référence vers une cible, telle que la valeur des paramètres du modèle par exemple, afin d'améliorer sa fonction prédictive (Zhuang *et al.*, 2020). L'apprentissage par transfert est utilisé lorsque les données étiquetées du domaine cible sont rares et qu'un modèle déjà entraîné provenant d'un domaine proche est disponible. On parle d'apprentissage par transfert homogène si les domaines sont proches, ou hétérogène si les domaines sont différents (Pan and Yang, 2010).

2.1.5 Apprentissage par renforcement

L'apprentissage par renforcement est un tout autre type d'apprentissage. Il se base sur un conditionnement du modèle. A l'instar du chien de Pavlov qui était conditionné par le son d'une cloche, ici le modèle (appelé agent) réagit à son environnement sur la base d'un système de récompense. L'agent effectue des actions de manière répétée et apprend de son expérience. Si l'action réalisée a été bénéfique, il est récompensé, cela peut se traduire par exemple par l'augmentation d'un score. A l'inverse, si l'action est néfaste, le score diminue. L'agent cherche donc à optimiser son score au travers de ses expériences. Ainsi, il ne se base pas sur des données externes, mais sur celles qu'il récolte en fonction de son environnement. Ce type d'apprentissage est souvent présent en robotique et dans les jeux vidéo (Kaelbling *et al.*, 1996), par exemple le jeu de dames de Samuel est basé sur de l'apprentissage par renforcement (Samuel, 1959). Parmi les algorithmes d'apprentissage par renforcement on retrouve notamment SARSA (*State-Action-Reward-State-Action*) et *Q-learning* (Watkins and Dayan, 1992).

2.1.6 Construction des jeux de données

Comme énoncé précédemment, les données sont la pierre angulaire des algorithmes de *machine learning*. Sans un jeu de données important, incluant des données cohérentes et de bonne qualité, il est fort probable que les performances seront réduites, peu importe l'algorithme utilisé. L'élaboration du jeu de données, appelé prétraitement, est une étape longue et fastidieuse, cependant il ne faut pas la négliger. En général, il s'agit de l'étape où le *data scientist* consacre la majorité du temps. Dans le cadre d'un apprentissage supervisé ou non, il est important de judicieusement choisir les descripteurs. En effet, au sein des données il peut y

avoir des informations bruitées ou manquantes, mais il peut également y avoir des informations redondantes ou superflues. Des données corrélées peuvent être redondantes, comme la longueur d'une séquence génomique et sa masse moléculaire. Ces données se recoupent et l'une d'entre elles peut donc être retirée. Pour mettre en évidence des corrélations entre les données, on peut réaliser une ACP qui réduira la dimensionnalité du jeu de données (Wold *et al.*, 1987), *i.e.* simplifier les données sans perdre trop d'informations. La pertinence des données est aussi importante. Selon la littérature (John *et al.*, 1994), les données peuvent être classées en trois catégories : fortement pertinentes, faiblement pertinentes et non pertinentes. La première catégorie est indispensable, car sa suppression induit une diminution des performances, à l'inverse, la dernière catégorie est inutile et sa suppression peut même améliorer les performances. Quant à la seconde catégorie, le choix de la garder ou non est difficile et se fait en général selon l'expertise du *data scientist*.

Lorsque les descripteurs des données d'entrées sont établis, il est indispensable de correctement étiqueter les données selon leur classe (dans le cadre d'un apprentissage supervisé). Cette étape, capitale et chronophage, peut sembler simpliste, cependant de nombreux jeux de données de référence intègrent un nombre important de données mal étiquetées (Northcutt *et al.*, 2021). Étiqueter les données peut s'avérer être une tâche plus complexe encore, dans la mesure où ces données sont fortement similaires (figure 2). Comprendre les données est donc la première étape pour réussir à les modéliser correctement.

Figure 2 - *Chihuahua ou muffin ? La mise en forme des données est l'étape la plus importante en machine learning, et peut s'avérer complexe quand les données se ressemblent. Dans cet exemple, quelque peu fantaisiste, il est important d'étiqueter correctement les données selon la classe 'chihuahua' ou 'muffin aux myrtilles', afin de ne pas introduire de biais lors de l'apprentissage (source : shyrobotics.com/cookie-dog-ai/).*

Une étape supplémentaire importante est la normalisation des données. Cette étape de prétraitement permet de transformer les données afin qu'elles soient comparables entre elles. Cela consiste par exemple à redéfinir les valeurs dans l'ensemble [0,1], permettent également de réaliser des calculs plus rapides.

2.1.6.1 Jeu d'entrainement

Lorsque le jeu de données initial est établi et vérifié, il faut construire différents sous-ensembles qui auront chacun un rôle spécifique lors du processus d'apprentissage du modèle. Le premier, et probablement le plus important, est le jeu d'entraînement. Il comporte la majorité des données qui seront utilisées par le modèle lors de son processus d'apprentissage. Le nombre de données à injecter dans le jeu d'entraînement est dépendant de la taille du jeu initial. Classiquement, il comprend 80% des données, mais s'il en contient peu, le modèle aura du mal à les généraliser. La figure 3A résume la répartition des différents jeux de données.

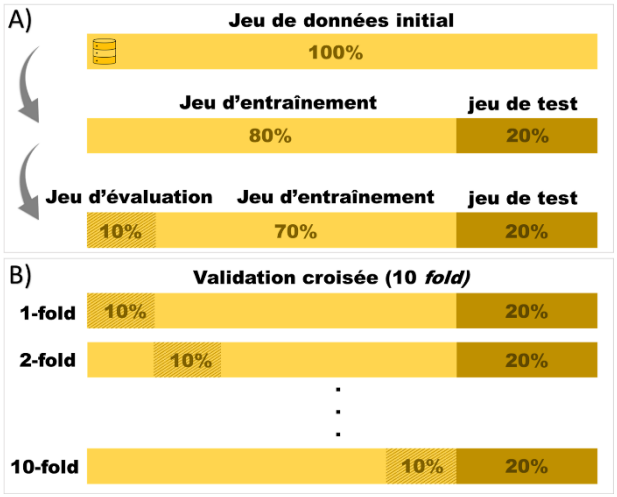


Figure 3 - *Les jeux de données du machine learning et la validation croisée.* ***A)*** *Composition des différents jeux de données.* ***B)*** *Principe de la validation croisée, avec ici 10-Fold, c'est-à-dire, que le jeu d'entraînement est découpé 10 fois pour construire 10 jeux d'évaluation contenant 10% des données.*

2.1.6.2 Jeu d'évaluation

Un des dangers lors de l'apprentissage en *machine learning* est le sur-apprentissage. Ce phénomène apparaît quand le modèle apprend par cœur les données, mais cette notion est reprise plus en détail dans la section 2.1.8. Pour lutter contre ce phénomène, il est important d'évaluer le modèle lors de l'étape d'apprentissage, avec un jeu d'évaluation pour estimer sa fiabilité et s'il généralise bien. Pour cela, on construit un jeu d'évaluation en récupérant entre 10 à 20% des données du jeu d'entraînement. Cependant, ce jeu peut être sous représenté et

biaiser les résultats. Pour éviter cela, on utilise une méthodologie de validation croisée (Stone, 1974) comme le *K-fold*. Cela consiste à découper *k* fois le jeu d'entraînement, afin d'obtenir *k* jeux de validation. Le modèle est ensuite entraîné puis évalué *k* fois et les résultats sont ensuite moyennés. Les autres techniques de validation croisées sont le *leave one out* ou le *holdout*. La figure 3B résume le fonctionnement de la validation croisée.

2.1.6.3 Jeu de test

Une fois le modèle entraîné, il faut évaluer ses performances. Pour cela on utilise un jeu de test, qui contient des données que le modèle n'a jamais vues. Il se complète au jeu d'entraînement et est par conséquent composé de 20% des données (si le jeu d'entraînement en contient 80%). Pour évaluer les performances du modèle sur le jeu de test, il existe plusieurs métriques mettant en avant certaines forces et faiblesses du modèle (voir section 7 du Matériels et méthodes).

2.1.7 Entraînement d'un modèle par apprentissage supervisé

Lorsque l'on parle d'entraînement d'un modèle, on parle de réglage de ses paramètres. L'objectif est de faire en sorte que les paramètres soient ajustés au mieux, pour que le modèle réalise une meilleure généralisation des données, en faisant attention de ne pas faire de sur-apprentissage. Cet ajustement est réalisé par le modèle lui-même et l'utilisateur n'intervient pas directement. Une fois que le modèle est entraîné, il a pour objectif de réaliser les tâches confiées telles que des prédictions ou de la classification. Pour entraîner un modèle, il faut donc utiliser les données du jeu d'entraînement. Le modèle calculé est une fonction, qui à partir des descripteurs, défini une sortie *y*.

Prenons un exemple simple, le cas d'une régression linéaire où le modèle peut être représenté par : $y = ax + b$ (figure 4). Comme il s'agit d'un apprentissage supervisé, par conséquent nous connaissons la valeur de *y* (étiquette) et la valeur de *x* (descripteurs). La phase d'apprentissage du modèle consiste alors à trouver les meilleures valeurs des paramètres *a* et *b* pour que la droite (le modèle) corresponde au mieux aux données.

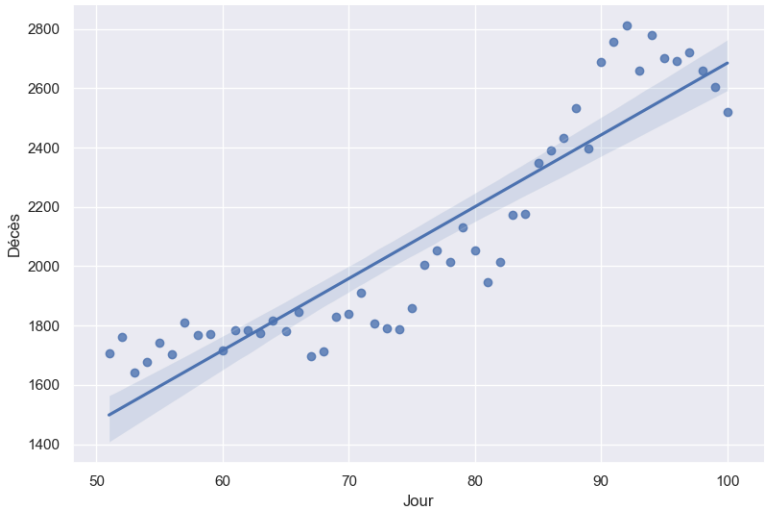


Figure 4 - *Représentation d'une régression linéaire. La figure décrit le nombre de décès en France lors de la première vague (fin février-début mars) de la pandémie liée à la COVID-19. Les données sont représentées par le nuage de point et le modèle est représenté par la droite (source des données : INSEE).*

2.1.7.1 La fonction de coût

Afin de guider l'apprentissage du modèle, il est important de lui donner quelques indications telles que sa performance. Ainsi, en modifiant les paramètres, si les performances diminuent cela signifie que la voie choisie pour ajuster les paramètres n'est pas la bonne. La différence (ou erreur) entre le modèle et les données s'appelle la fonction de coût. Dans le cas d'une régression, on peut par exemple utiliser le *Root Mean Square Error* (RMSE) défini par :

$$RMSE(X,h) = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(h(x_i) - y_i)^2}$$

avec m le nombre d'observations, x la matrice composée des différentes valeurs des descripteurs, y la valeur de la sortie désirée et h la fonction de prédiction hypothétique. Il s'agit de l'écart entre la donnée observée et le modèle, plus la somme des écarts est faible plus le modèle « généralise » bien les données.

Dans le cadre d'une tâche de classification, le modèle doit séparer les données selon leur classe. Pour calculer l'erreur, on peut compter le nombre d'éléments mal classés parmi le nombre total d'éléments. Le processus d'apprentissage a donc pour objectif de réduire le nombre de données mal classées et donc la fonction de coût en modifiant les paramètres du modèle.

2.1.7.2 La descente de gradient

L'ajustement des paramètres (*e.g. a* et *b* pour une régression linéaire) du modèle constitue la phase d'apprentissage. Cependant ces réglages ne peuvent se faire de manière exhaustive, il est donc nécessaire de l'automatiser avec un algorithme. En *machine learning*, un des principaux algorithmes utilisés est la descente de gradient. Il s'agit d'un algorithme d'optimisation qui va chercher un minimum d'une fonction de coût. Son principe consiste à modifier les paramètres du modèle de manière itérative afin de minimiser la fonction de coût. Pour cela, l'algorithme calcule le gradient (*i.e.* le vecteur représentant la variation d'une fonction par rapport aux paramètres établis) de cette fonction. S'il est négatif alors l'algorithme se dirige vers un optimum local. Un des paramètres importants est le pas (ou *learning rate*) lors de chaque itération, s'il est trop petit, l'algorithme sera très long pour trouver l'optimum local, voire trop long et ne convergera pas. A l'inverse, s'il est trop grand, le gradient peut osciller et diverger.

Lorsque le jeu de données a une taille importante, l'exécution de l'algorithme peut être relativement longue. Dans ce cas, on réalise un échantillonnage aléatoire des données à chaque itération, qu'on exploite ensuite. On parle également de lot (ou de *batch)*. Cette alternative se nomme la descente de gradient stochastique et est plus rapide que sa version initiale. L'hypothèse émise lorsque l'on utilise cet algorithme est que le lot de données choisi aléatoirement est représentatif de l'ensemble des données. Cependant, on observe un phénomène d'oscillation qui peut induire du bruit et ralentir la convergence.

Actuellement, différentes approches ont été développées pour limiter ces biais, telles que le *momentum* (Qian, 1999) qui permet de garder en mémoire le gradient précédent, ou encore le gradient d'accélération de Nesterov (Ruder, 2017) qui exploite le gradient en ayant pris en considération le gradient au temps *t+1*.

2.1.8 Sur-apprentissage et sous-apprentissage

Lorsque l'on parle d'apprentissage en *machine learning*, il n'est pas rare de voir les termes sur-apprentissage (*overfitting*) ou sous-apprentissage (*underfitting*). Il s'agit de deux phénomènes opposés, mais qui sont tous deux sources de mauvaises performances. Nous avons vu précédemment que lors de l'apprentissage, le modèle doit généraliser au mieux les données d'entraînement pour ensuite réaliser des prédictions sur des données inconnues. Mais que se passe-t-il quand un modèle généralise trop, ou pas assez ?

Le sur-apprentissage est le phénomène qui se produit quand le modèle a “trop” bien appris les données, il les représente parfaitement. Ainsi, lors de l’évaluation sur le jeu de test, les performances sont souvent exceptionnelles, mais dès qu’on l’utilise sur des données inconnues, par exemple sur un benchmark indépendant, les performances s’effondrent. En effet, le modèle va extraire toutes les caractéristiques et variations des données sans les généraliser (Hawkins, 2004). La figure 5A représente une situation de sur-apprentissage.

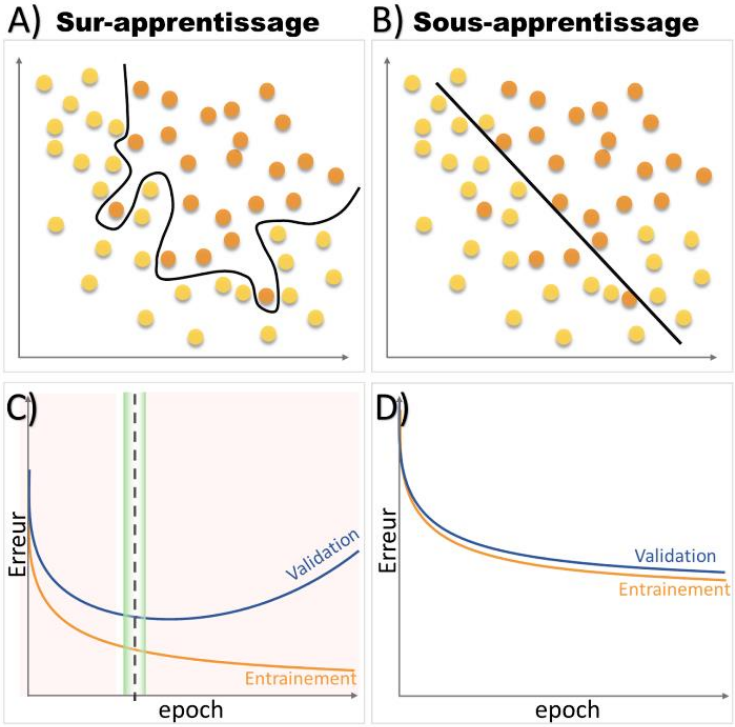


Figure 5 - *Phénomènes de sur et sous-apprentissage.* ***A)*** *Cas d’un sur-apprentissage : les données sont parfaitement classées, car le modèle a appris par cœur leurs caractéristiques.* ***B)*** *Cas d’un sous-apprentissage où le modèle ne généralise pas assez les données et n’est pas précis.* ***C)*** *Détection du sur-apprentissage lorsque la valeur de l’erreur sur le jeu d’entraînement continue de baisser alors que celle sur le jeu de validation augmente. L’entraînement doit être stoppé au moment où la courbe de validation commence à augmenter (barre en pointillé) = early stopping.* ***D)*** *Détection de sous-apprentissage, la valeur de l’erreur diminue très faiblement, car le modèle n’arrive pas à généraliser les données.*

A l’inverse, le phénomène de sous-apprentissage (figure 5B) se produit quand le modèle a “mal” appris, il n’arrive pas à extraire les caractéristiques des données de manière efficace et donc ne généralise pas les données non plus (Sehra *et al.*, 2021). Le sous-apprentissage est souvent induit par un modèle trop simple. Lorsque ce phénomène se produit, les performances sur le jeu de test sont souvent faibles et l’erreur reste assez élevée.

Ces phénomènes sont donc des “bornes” qu’il faut éviter pour qu’un entraînement soit efficace. La difficulté réside donc dans le fait de trouver un juste milieu (Van der Aalst *et al.*,

2008). Ce compromis est connu sous le terme “biais versus variance” (Geman *et al.*, 1992). Le biais correspond aux mauvaises prédictions (ou l’erreur) qui se produisent quand le modèle est en sous-apprentissage. La variance est l’erreur due à la sensibilité du modèle lors du sur-apprentissage. Le compromis biais-variance est un des facteurs difficiles à maîtriser, mais crucial pour tirer les meilleures performances des modèles. Il existe cependant une astuce pour nous guider et détecter ces phénomènes. En effet, lors de l’apprentissage, on peut analyser l’erreur (de la fonction de coût) sur le jeu d’entraînement et sur le jeu d’évaluation. Lors de la phase d’apprentissage, l’erreur va diminuer sur les données d’entraînement ainsi que sur le jeu d’évaluation. Lorsque le modèle va commencer à faire du sur-apprentissage, il va trop représenter les données d’entraînement, donc l’erreur sur le jeu d’entraînement continuera à diminuer, mais elle augmentera de plus en plus sur le jeu de validation parce que le modèle ne généralise plus. La figure 5C représente la détection de sur-apprentissage et la figure 5D de sous-apprentissage.

2.1.9 Evaluation des modèles de classification binaire

A l’instar d’une machine avec divers capteurs et alarmes, le *machine learning* possède également des éléments permettant de contrôler et évaluer la fiabilité du modèle. Dans le cas d’une classification et d’un apprentissage supervisé, il existe plusieurs métriques qui sont fondamentalement basées sur une matrice de confusion. Cette dernière correspond à une matrice répertoriant le nombre d’éléments correctement identifiés ou non. Elle se compose des éléments réels (ou observés) que l’on devrait identifier et des éléments que le modèle prédit. Ainsi, on parle de vrais positifs (VP) quand la donnée observée appartient à la classe positive et que le modèle l’a correctement prédit, de faux positifs (FP) quand la prédiction est positive mais l’observation est négative, de vrais négatifs (VN) quand la prédiction est négative et la vraie classe est négative, et enfin on parle de faux négatifs (FN) quand la prédiction est négative mais l’observation est positive. La figure 6 illustre une matrice de confusion binaire.

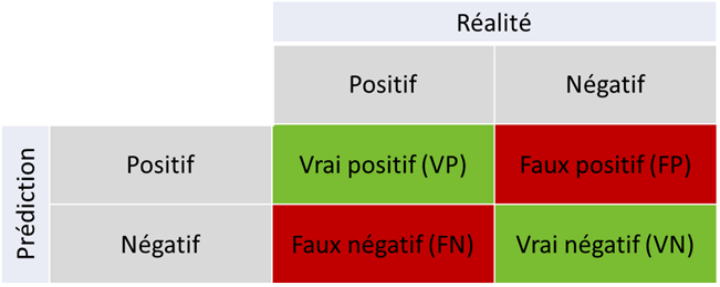


Figure 6 - *Matrice de confusion pour une classification binaire. Elle met en évidence les quatre catégories d'éléments (VP, FP, VN, FN) qui permettent de calculer différentes métriques et évaluer la qualité et la performance d'un modèle.*

A partir de ces quatre valeurs, on peut établir plusieurs métriques visant à contrôler et évaluer les performances du modèle, et ainsi mettre en avant ses forces et ses défauts. Les métriques les plus utilisées en *machine learning* et que j'ai appliqué sont présentées dans la section 7 du Matériels et méthodes.

2.2 Les réseaux de neurones

Parmi l'ensemble des algorithmes d'apprentissage automatique supervisé permettant de faire de la classification, je me suis focalisé sur les réseaux de neurones artificiels. Ces derniers ont été largement utilisés dans de nombreux domaines, dont la biologie et la bio-informatique grâce à leurs performances exceptionnelles (AlQuraishi, 2019; Tang *et al.*, 2019; Koumakis, 2020; Li *et al.*, 2020), mais également grâce à la disponibilité des données et la puissance de calcul adéquate.

2.2.1 La bio-inspiration des neurones formels

Le neurone formel est l'unité élémentaire des réseaux de neurones actuels. Des travaux préliminaires (McCulloch and Pitts, 1943) ont été réalisés au début des années 40 et ont bouleversé le domaine de l'IA. Ils se sont inspirés des neurones biologiques que l'on observe dans le cerveau des mammifères. Un neurone est une cellule nerveuse excitable qui transmet un signal (influx nerveux). Tout comme le neurone formel, le neurone biologique est l'unité centrale d'un système complexe qu'est le système nerveux. Le neurone biologique est composé de trois éléments fondamentaux : le corps cellulaire (péricaryon), les dendrites et l'axone (Freeman, 1992). Le péricaryon contient le noyau, les informations génétiques ainsi que les organelles nécessaires pour le maintien de l'homéostasie cellulaire. L'axone est un long

prolongement cellulaire qui conduit le signal vers les autres cellules, enfin les dendrites sont des prolongements plus courts qui présentent une structure arborescente, permettant de recouvrir une surface plus importante et ainsi de capter de nombreux signaux externes. A l'interface entre deux neurones, on retrouve les synapses, qui établissent une interaction physico-chimique à l'aide de neurotransmetteurs ou d'ions, permettant la transmission du signal. La figure 7A schématise un neurone biologique.

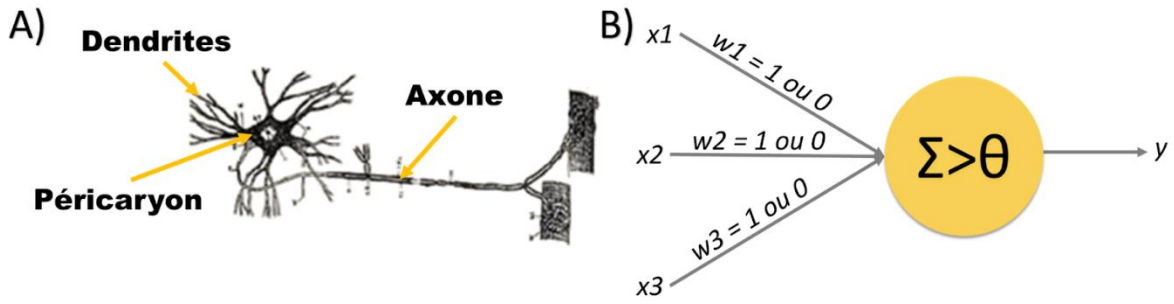


Figure 7 - *Du neurone biologique au neurone formel.* ***A)*** *Représentation schématique d'un neurone biologique composé d'un péricaryon, de plusieurs dendrites et d'un axone afin de transmettre un influx nerveux.* ***B)*** *Neurone formel composé de plusieurs entrées x (dendrites), un "centre de calcul" (péricaryon), et une sortie y (axone). Les valeurs d'entrées sont binaires (0 ou 1) et sont prises en compte par le centre de calcul. Si la somme de ces valeurs est supérieure au seuil, alors le neurone est activé (source* ***A)*** *(Wingate and Kwint, 2006)).*

Le neurone formel est donc un modèle mathématique inspiré d'un neurone biologique. Il a été conceptualisé en intégrant les éléments structuraux décrits précédemment. Classiquement, le neurone formel est constitué de plusieurs entrées *x* (dendrites), un "centre de calcul" (péricaryon), et une sortie *y* (axone) (figure 7B). Les signaux (sous la forme d'un vecteur) proviennent des différentes entrées et sont pris en compte dans le "centre de calcul" qui va les sommer. Cette somme va activer ou non le neurone dès lors que le résultat dépasse un seuil θ. Si le neurone reste actif, il envoie alors un signal à d'autres neurones via sa sortie. Les neurones de McCulloch et Pitts fonctionnaient initialement selon des règles logiques établies au XIX^{e} siècle par George Boole (Boole, 1847). Les entrées et sorties étaient binaires, soit 0 soit 1, il était alors possible d'exploiter des portes logiques pour activer les neurones. Cependant, comme les valeurs restent constantes, il n'est pas possible pour ce type de neurone d'apprendre (*i.e.* de modifier les valeurs pour réduire l'erreur).

Le neurone formel a ensuite été amélioré pour donner un neurone "artificiel". Ce dernier est basé sur les mêmes caractéristiques que son prédécesseur, cependant quelques subtilités viennent le complexifier. En effet, chaque valeur de signal (entrée ou sortie) n'est plus une valeur binaire, mais un nombre décimal. Les signaux d'entrée sont également associés à un

poids, on obtient donc un vecteur de taille *m*, où *m* correspond au nombre de binômes $(x_i w_i)$ avec *x* la valeur du signal et *w* la valeur du poids associé. Le flux d'entrée correspond donc a :

$$input = w_m x = \sum (x_i * w_i)$$

Le résultat est ensuite soumis à une fonction d'activation $f(input)$ (Abraham, 2005). Cette dernière a pour objectif de transformer le signal et d'activer ou non le neurone (en fonction d'un seuil θ) selon le modèle mathématique qu'elle applique. Un dernier paramètre est attribué au neurone, il s'agit du biais noté *b*, qui est une valeur constante pouvant impacter la valeur du signal final. Ainsi, le modèle final correspond donc à l'équation :

$$y = f(\sum ((x_i * w_i) + b)$$

Un neurone seul est un modèle non linéaire capable d'effectuer une classification binaire, à l'instar des régressions. Si le neurone est activé *i.e.* la somme des entrées pondérées a dépassé le seuil θ, alors l'entrée est classée dans la catégorie positive. Ainsi, la magie de l'intelligence artificielle réside dans la faculté à modifier les poids et le biais des neurones artificiels, pour optimiser les performances du modèle et réduire l'erreur.

2.2.2 Les fonctions d'activation

Une fonction d'activation correspond à une fonction mathématique. Elle permet de transformer un signal d'entrée en signal de sortie et de le transmettre vers d'autres neurones (Sharma *et al.*, 2020).

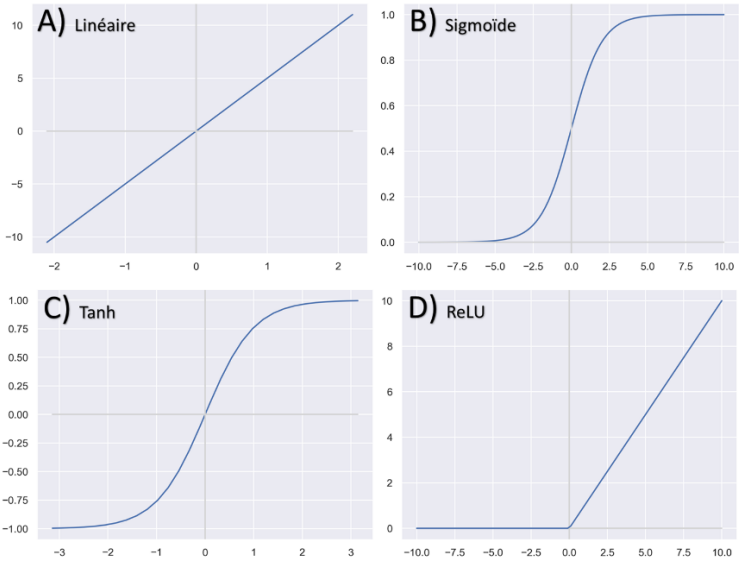


Figure 8 - *Principales fonctions d'activation des neurones artificiels.* ***A)*** *La fonction d'activation linéaire est la plus simple, elle est définie entre -∞ et +∞.* ***B)*** *La fonction sigmoïde est efficace lorsque l'on réalise une étape de classification, elle est définie entre 0 et 1.* ***C)*** *La fonction tangente hyperbolique (Tanh) converge plus rapidement car certaines valeurs sont plus susceptibles d'être proches de 0, du fait de son intervalle compris entre -1 et 1.* ***D)*** *La fonction ReLU est souvent utilisée, car elle permet d'inactiver des neurones si la valeur d'entrée est inférieure à 0.*

Parmi les fonctions d'activation les plus connues (figure 8), on retrouve :

- La *fonction d'activation linéaire* qui est la plus basique. Elle est représentée par une droite avec la valeur du signal de sortie proportionnelle au signal d'entrée. Cette fonction est linéaire et se caractérise par l'équation : $f(x) = ax$, avec a étant une constante (figure 8A).
- La *fonction d'activation sigmoïde* est plus couramment utilisée et permet au modèle de mieux généraliser car elle est non linéaire. Elle transforme les valeurs d'entrée en valeurs comprises entre 0 et 1. Cette fonction a la particularité d'avoir un impact important quand les valeurs des entrées sont proches de 0, et un impact moindre quand elles sont proches de 1, *i.e.* le gradient sera très faible (Feng and Lu, 2019). Cette fonction se définit par : $f(x) = \frac{1}{1+e^{xi}}$ (figure 8B).
- La *fonction tangente hyperbolique* (Tanh) ressemble à la fonction sigmoïde, il s'agit aussi d'une fonction non linéaire, cependant elle modifie les valeurs d'entrée en valeurs incluses dans l'intervalle [-1,1] en étant centrée sur 0. Ainsi, les valeurs de sortie peuvent être proches de 0, ce qui permettra de converger plus rapidement que la fonction sigmoïde car le gradient sera plus important (Lecun *et al.*, 1998) (figure 8C). La fonction Tanh correspond à : $f(x) = \left(\frac{2}{1+e^{2x}}\right) - 1$

- La fonction *unité de rectification linéaire* (ou *rectified linear units*, ReLU) est une des fonctions d'activation les plus utilisées dans les réseaux de neurones. La particularité de cette fonction est l'activation d'un certain nombre de neurones et l'inactivation d'autres, la rendant plus efficace que Tanh (Sharma *et al.*, 2020). ReLU n'est pas une fonction linéaire et n'a pas de limite finie (figure 8D), elle se définit par l'équation : $f(x) = \max(0, x)$ avec une valeur de sortie x si $x \geq 0$ et 0 si $x < 0$. Il existe d'autres variantes de ReLU telles que Leaky ReLU, PReLU ou ELU, qui consiste à modifier la pente lorsque les valeurs sont inférieures à 0.
- La fonction *softmax* est souvent utilisée sur les neurones de la dernière couche lorsque l'on effectue de la classification. En effet, un des avantages de cette fonction sigmoïde est sa capacité à générer une distribution de probabilités en renvoyant des valeurs entre 0 et 1 et la somme de ces valeurs est égale à 1. Ainsi, en définissant x neurones sur la dernière couche, correspondant donc à x classes, on obtient des probabilités pour chaque classe. La fonction softmax se définit par l'équation suivante : $\sigma(z)_j = \frac{e^{zj}}{\sum_{k=1}^{K} e^{zk}}$, avec K le nombre de classes.

2.2.3 Le perceptron

Les réseaux de neurones (RN) artificiels sont inspirés des réseaux de neurones biologiques présents dans l'encéphale de mammifères. Il s'agit d'un modèle mathématique constitué d'une architecture spécifique, où chaque nœud du réseau correspond à un neurone artificiel et chaque lien entre ces neurones correspond aux synapses. Les RN artificiels sont organisés en couches de neurones où chaque neurone est interconnecté avec ceux de la couche suivante. L'information se transmet de la première jusqu'à la dernière couche par l'intermédiaire des synapses et les poids synaptiques exercent une influence sur l'activation ou non de chaque neurone.

Les premiers RN ont été développés sur la base du perceptron (Rosenblatt, 1958). Il s'agit d'un algorithme d'apprentissage automatique supervisé réalisant des tâches de classification binaire, remarquablement efficace lorsqu'il s'agit de traiter des problèmes linéaires. Son architecture, schématisée par la figure 9, est composée d'une zone de projection qui capte les signaux issus des données d'entrée. Selon certaines définitions, cette zone est considérée comme la première couche du réseau, appelée couche d'entrée. Le signal est transmis vers les neurones via les synapses et est pondéré en fonction des poids synaptiques et

une valeur de biais y est additionnée. Par la suite, l'ensemble des signaux sont réceptionnés et traités par l'unique neurone de la couche de sortie. La fonction d'activation de ce dernier participe à la traduction du résultat final. Cette particularité permet au perceptron de réaliser la classification binaire (Abraham, 2005). Afin d'obtenir de meilleures performances, une étape d'apprentissage est nécessaire pour ajuster les paramètres du RN au moyen de l'algorithme de la descente de gradient. Malheureusement les limites des perceptrons sont atteintes lorsque le problème se complexifie et qu'il n'est pas linéaire, comme le problème du XOR (Yanling *et al.*, 2002).

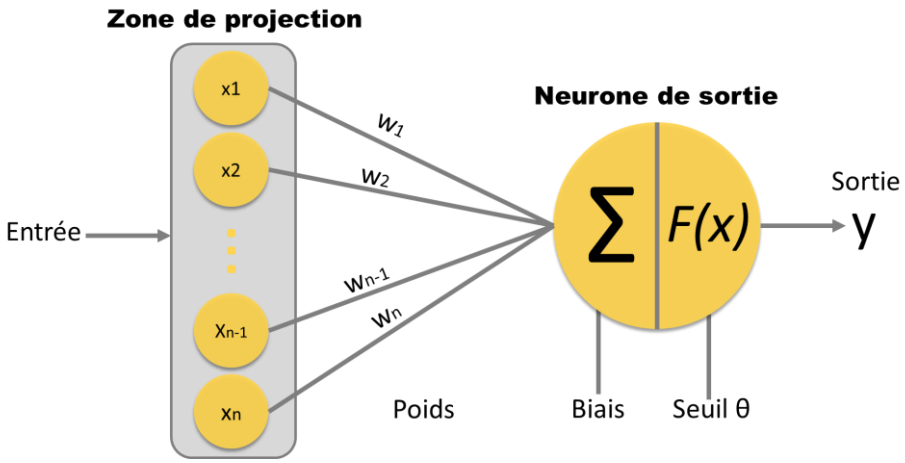


Figure 9 - *Représentation schématique d'un perceptron. Ce dernier est composé d'une zone de projection qui capte le signal d'entrée et d'un neurone de sortie qui le traite. Le signal est ensuite pondéré par les poids synaptiques puis sommé. Si la somme dépasse le seuil θ le neurone est activé et le résultat est traduit selon la fonction d'activation.*

2.2.4 Le perceptron multi-couche

Le perceptron de Rosenblatt a été amélioré notamment par l'adjonction de nouvelles couches de neurones, afin de former un réseau de neurones. On parle ainsi de perceptron multi-couche (*multilayer perceptron*) ou PMC. Le signal ne se transmet plus directement de la zone de projection vers le neurone de la couche de sortie, mais à des neurones d'une nouvelle couche appelée couche cachée. Chaque neurone de la couche *n* est ainsi connecté avec l'ensemble des neurones de la couche cachée *n+1*. Le signal va alors se propager dans les couches suivantes de manière unidirectionnelle sans retour arrière possible (il est donc acyclique). On appelle ce type d'architecture, un réseau à propagation avant (*feed-forward*). Dans cette architecture, il n'existe aucun neurone connecté à d'autres neurones de la même couche. De plus, le nombre de couches cachées n'est pas limité, cependant la complexité de l'architecture intervient dans les performances finales du réseau. Un PMC trop complexe risque de réaliser du sur-

apprentissage et donc de mal généraliser (Murtagh, 1991). L'ajout des couches cachées a pour avantage de transformer les données d'entrée dans un autre espace. Plus on ajoute de couches cachées, plus le nombre de frontières de décision augmente. Ainsi, dans le cadre du problème du XOR, cela consiste à réduire la problématique non linéaire en plusieurs problématiques linéaires. *In fine*, le PMC est capable de séparer les données en deux, en combinant les résultats des couches précédentes, tel que représenté dans la figure 10. Bien que le PMC soit capable d'obtenir des performances élevées, il est limité par les problèmes de mise à l'échelle et de l'explosion du nombre de paramètres. Cela peut avoir comme conséquence un apprentissage très long et une mauvaise généralisation des données. Une des avancées majeures qui a permis d'obtenir des performances très élevées avec les PMC est la méthode d'apprentissage, appelée algorithme de rétropropagation du gradient de l'erreur.

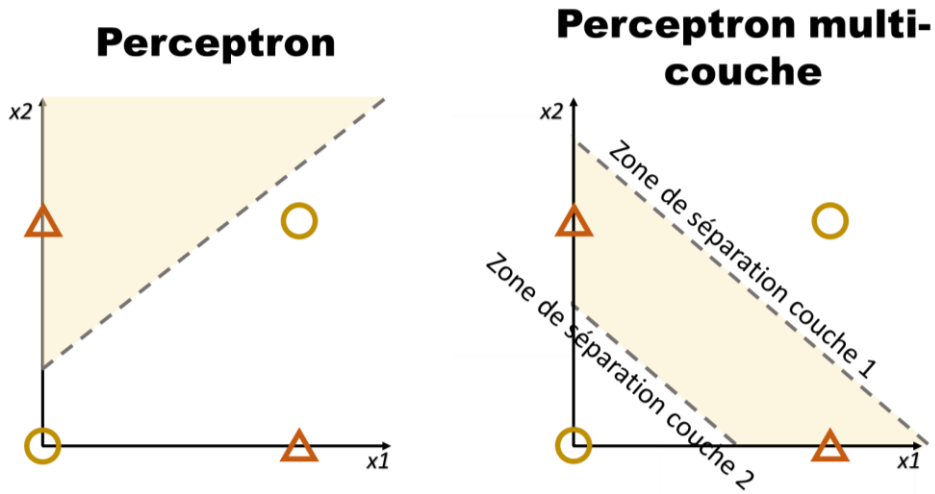


Figure 10 - *Résolution du problème du XOR à l'aide d'un perceptron multi-couche. La couche cachée permet d'ajouter une nouvelle dimension, modélisée par des frontières de décision. Le perceptron classique ne peut résoudre ce problème, car il n'est pas linéairement séparable.*

2.2.5 L'algorithme de rétropropagation du gradient de l'erreur

Initialement conceptualisé par Werbos (Werbos, 1994), puis indépendamment repris par LeCun (Lecun, 1985) et Hinton (Rumelhart *et al.*, 1986), l'algorithme de rétropropagation du gradient de l'erreur (en anglais *backpropagation algorithm*) est un algorithme d'optimisation largement utilisé pour minimiser l'erreur lorsque l'on utilise des RN multi-couche.

Habituellement, le RN effectue une prédiction en fonction des signaux qui se sont propagés au travers des neurones des différentes couches et qui ont fini par atteindre la couche de sortie. Cette première étape est appelée "passe avant" ou *forward pass*. L'erreur est ensuite calculée en utilisant la fonction de coût. La deuxième étape consiste à revenir en arrière en

propageant le gradient de l'erreur, pour identifier l'impact de chaque paramètre (poids et biais) des neurones de chaque couche dans la prédiction et les modifier pour réduire l'erreur en utilisant la descente de gradient. En connaissant l'impact des paramètres des neurones les plus en aval, on peut revenir en arrière et identifier l'impact des paramètres des neurones en amont. Les paramètres sont alors actualisés proportionnellement à la dérivée partielle de la fonction d'erreur par rapport au poids. Cette étape est appelée "passe arrière" ou *backward* (LeCun *et al.*, 2015). Le processus est ensuite répété plusieurs fois et chaque itération est appelée une epoch . La figure 11 résume cet algorithme.

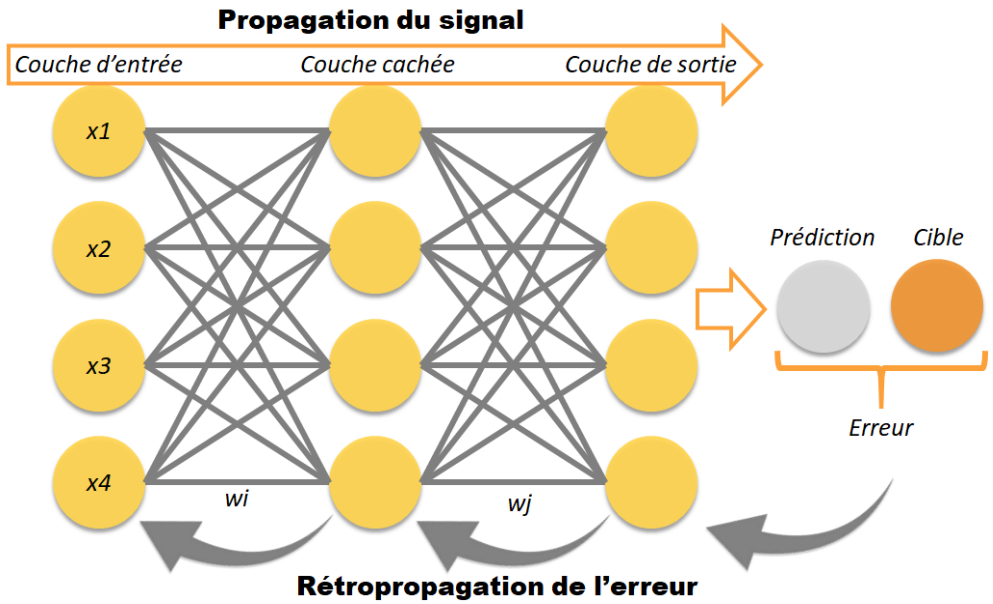


Figure 11 - *Représentation de l'algorithme de rétropropagation du gradient de l'erreur. Les signaux captés par la couche d'entrée vont se répandre au travers des synapses (liens gris) et subir une étape de transformation dans chaque neurone. Lorsque le signal atteint la couche de sortie, il est traduit en résultat. Ce dernier est comparé avec le résultat attendu (la cible) et l'erreur est calculée. La suite de l'algorithme consiste à revenir en arrière pour évaluer l'impact de chaque paramètre des neurones, puis de les modifier, jusqu'à atteindre la première couche de neurone. Le processus est itératif et chaque passage est appelé epoch.*

Bien que cet algorithme soit très performant, il comporte certains défauts qui engendrent divers problèmes. On peut noter par exemple la disparition des gradients (*gradient vanishing*) qui se produit lorsque les gradients des premières couches sont très faibles et approchent 0, induisant un ralentissement voire une inhibition de l'entraînement. A l'inverse, le gradient peut exploser si la modification des poids est trop importante (Philipp *et al.*, 2018).

2.2.6 Les différentes architectures de réseaux de neurones artificiels

Le perceptron et son amélioration le PMC, ne sont que la face visible de l'iceberg. Sa partie submergée est un monde encore plus vaste, composé de nombreuses autres architectures (figure 12). Parmi les plus répandus, on retrouve les réseaux de neurones récurrents (*Recurrent Neural Networks* - RNN), les réseaux antagonistes génératifs (*Generative adversarial Networks* - GAN) ou encore les réseaux de neurones convolutifs (*Convolutional Neural Networks* - CNN).

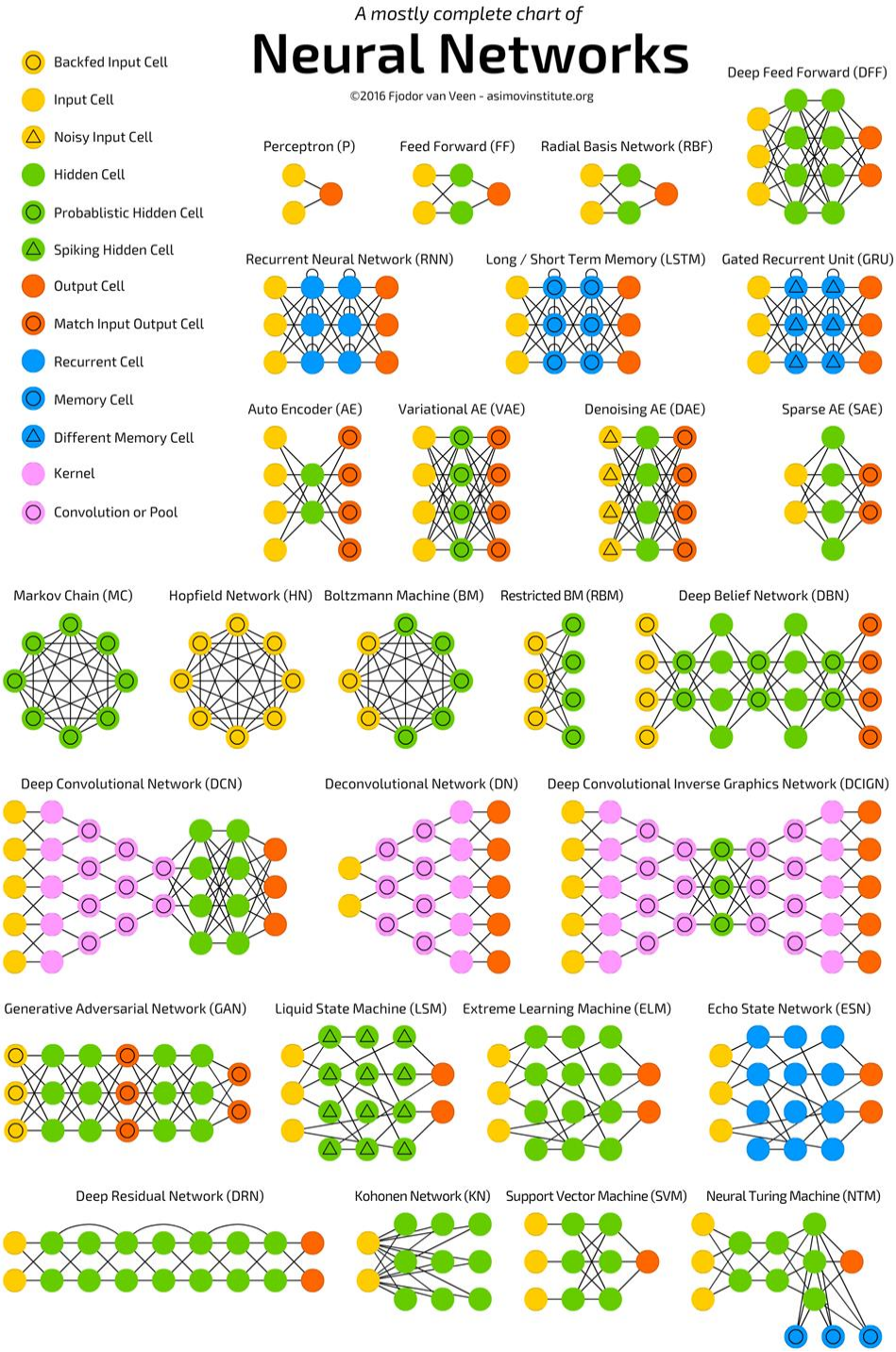


Figure 12 - *Catalogue des différentes architectures de réseaux de neurones artificiels (source : (Leijnen and Veen, 2020)).*

2.2.6.1 Les réseaux de neurones récurrents

Les RNN sont un type particulier de RN artificiel, en effet, ils ne reposent pas sur une architecture *feed-forward* (*i.e.* le sens du signal est unidirectionnel et acyclique), mais ils intègrent des cycles, c'est-à-dire des connexions récurrentes entre d'autres neurones ou sur le neurone lui-même. Cette particularité permet de garder en mémoire une information antérieure puis de prendre une décision en fonction des éléments mémorisés. Par conséquent, ces réseaux de neurones sont particulièrement adaptés pour le traitement de signaux ou de données séquentielles (Lipton *et al.*, 2015). L'état de l'art des RNN est représenté par les réseaux de neurones à base de cellules *long short-term memory* (LSTM) (Hochreiter and Schmidhuber, 1997). Leur capacité de mémorisation est plus importante et ils sont capables de prendre en compte un contexte plus étendu en limitant la perte du gradient. Les LSTM ont été largement exploités pour traiter des processus du langage naturel et différentes applications ont été réalisées telles que la traduction automatique (Sutskever *et al.*, 2014) ou la description d'images (Vinyals *et al.*, 2015). Les assistants vocaux tels que Siri ou Google assistant se sont également imprégnés de cette technologie, et demain les traducteurs instantanés nous aideront à communiquer dans n'importe quelle langue.

2.2.6.2 Les réseaux antagonistes génératifs

Les GAN sont des algorithmes d'apprentissage automatique non supervisé. A l'instar du test de Turing, ces réseaux mettent en compétition deux modèles, un qui génère une donnée (appelé le générateur) et un autre (le discriminateur) qui doit prédire si la donnée est réelle ou si elle a été produite par le générateur (Goodfellow *et al.*, 2014). Ces réseaux se sont distingués dans de nombreuses applications (Creswell *et al.*, 2018) telles que la synthèse d'images, avec des résultats remarquables, au point qu'il devient très difficile de discerner une image réelle d'une image générée par un GAN (figure 13) (Wang *et al.*, 2020). Une autre application des GAN est l'amélioration de la résolution des images, par exemple en 2020, le célèbre film des frères Lumière de 1896 "L'Arrivée d'un train en gare de La Ciotat" a été amélioré en une version 4K avec 60 images par seconde. Enfin, une autre application des GAN est la création d'œuvres d'art (Elgammal *et al.*, 2017).

Figure 13 - *Performance d'un réseau antagoniste génératif. Comparaison entre une image issue d'un jeu de données et une image générée par un GAN. Il est (quasiment) impossible de déterminer quelle image est celle produite par le réseau de neurones RelGan (source : (Wu et al., 2019)).*

2.2.6.3 Les réseaux de neurones convolutifs

Les CNN sont un type de RN *feed-forward*, inspirés des travaux de Hubel et Wiesel sur le cortex visuel de chats et de primates (Hubel and Wiesel, 1959). Ils ont démontré que les neurones du cortex ne réagissent qu'à des stimuli locaux. Le potentiel des CNN est réellement dévoilé avec les travaux de LeCun (LeCun *et al.*, 1989; Lecun *et al.*, 1998). Ils sont organisés en couches, capables d'extraire des informations localement comme le démontre la figure 14 qui représente l'architecture du réseau LeNet-5. Les CNN ont été largement utilisés pour la classification d'images comme dans le cadre de diagnostic médical (Spanhol *et al.*, 2016; Amin *et al.*, 2018; Reda *et al.*, 2018), pour la conduite autonome (Hadsell *et al.*, 2009) ou encore en bio-informatique pour la détection de sites d'épissage (Wang *et al.*, 2019). Ils peuvent exploiter des images, c'est-à-dire, des données en 2 dimensions, mais ils sont aussi capables de tirer profit des données à une dimension telle que des signaux d'électrocardiogramme (Yıldırım *et al.*, 2018) ou des séquences génomiques (Umarov and Solovyev, 2017). De par leur haute performance d'extraction des informations spatiales et contextuelles, les CNN sont un très bon choix d'algorithmes pour détecter des motifs particuliers. Les CNN sont les réseaux de neurones que j'ai utilisés lors de mes travaux de thèse, c'est pourquoi je reviens plus en détail sur leur fonctionnement dans la section 5 du Matériels et méthodes.

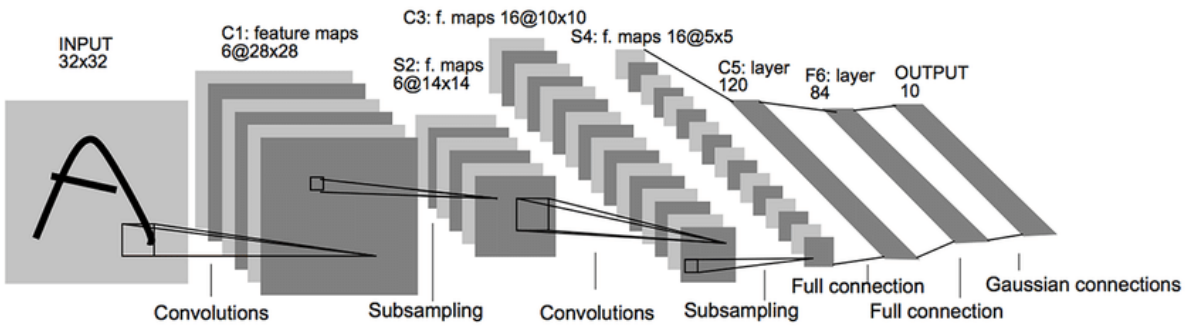


Figure 14 - *Architecture du réseau de neurone convolutif LeNet-5 établit par LeCun. Le CNN est capable d'extraire des motifs, dans une zone particulière. C'est le cas ici, où la branche droite de la lettre "A" est extraite, générant une caractéristique (source : (Lecun et al., 1998)).*

2.3 L'apprentissage profond ou *deep learning*

L'apprentissage profond, plus connu sous le terme anglais "*deep learning*" est une branche du *machine learning* et plus généralement de l'IA. Ses méthodes sont principalement basées sur des RN artificiels qui ont la particularité d'être profonds. Cela signifie que l'architecture est composée d'un nombre important de couches cachées. Définir la limite entre un RN artificiel profond ou non est encore flou, certains considèrent que 10 couches le caractérisent, d'autres une centaine. Les réseaux de neurones cités précédemment (PMC ≥ 3 couches, RNN, CNN) font partie des algorithmes inclus dans le *deep learning*.

Hormis leur particularité architecturale, les RN artificiels profonds modélisent les données avec un haut niveau d'abstraction, *i.e.* on part de données brutes, comme une image, et on les réduit à un niveau très abstrait (figure 15). De plus, pour certains algorithmes de *deep learning* comme les CNN, il n'est pas nécessaire de définir les caractéristiques des données, car ils sont capables d'extraire par eux-mêmes celles qui leur semblent pertinentes.

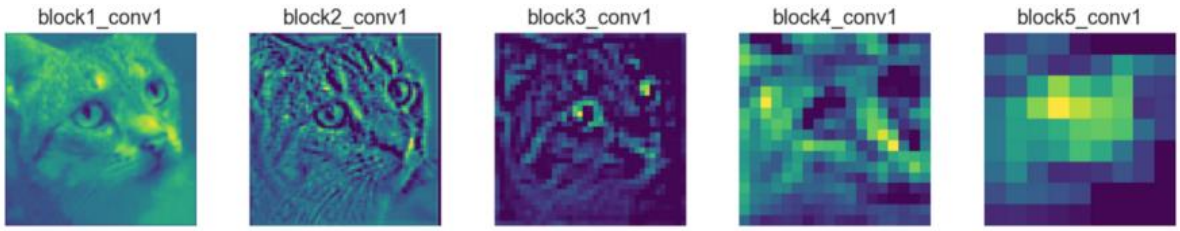


Figure 15 - *Représentation du niveau d'abstraction en fonction de la profondeur de l'architecture des réseaux de neurones. Les premières couches du réseau vont extraire des motifs simples tels que des droites ou des courbes pour déterminer les contours. Une droite est donc mathématiquement simple et peu abstraite. Plus les couches sont profondes, plus les motifs vont être abstraits, car ils seront formés par plusieurs combinaisons de motifs simples. Le résultat est une fonction mathématique complexe et abstraite qui permet au CNN de reconnaître des motifs sophistiqués tels que des oreilles de chat (source :towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2).*

Les CNN sont des algorithmes de *deep learning très* utilisés et des plus compétitifs aujourd'hui. La puissance de calcul et la quantité importante de données issues du *Big Data* ont permis d'obtenir des performances hors normes, faisant du *deep learning* et des CNN un atout majeur pour les chercheurs et les entreprises.

Finalement, les rebondissements historiques du *machine learning*, ainsi que les évolutions technologiques, ont induit l'émergence du *deep learning* qui est utilisé indirectement au quotidien par des millions, sinon des milliards d'individus. Mais l'IA n'est pas uniquement

composée du *machine learning* et du *deep learning*. Il existe d'autres approches tout aussi séduisantes et passionnantes. Dans le prochain chapitre, je présente des algorithmes d'évolution artificielle qui sont également bioinspirés et qui partagent une histoire parallèle à celle des algorithmes de *machine learning*.

Chapitre 3 - Les algorithmes évolutionnaires

"If variations useful to any organic being do occur, assuredly individuals thus characterized will have the best chance of being preserved in the struggle for life [...]. This principle of preservation, I have called, for the sake of brevity, Natural Selection."

- Charles Darwin -

Les algorithmes basés sur des concepts évolutionnaires sont différents des algorithmes d'apprentissage. Pour les distinguer, on peut les opposer selon leur objectif et leur approche principale pour résoudre leur tâche. En ce sens, on peut faire la distinction élémentaire entre un algorithme qui apprend en définissant des règles et un algorithme qui optimise. Ainsi, les algorithmes évolutionnaires (AE) appartiennent à la deuxième catégorie. Ce sont des algorithmes qui se focalisent sur la résolution de problèmes d'optimisation (McDowell and Popa, 2010). Ils sont dits "approximatifs", car ils vont trouver une (ou plusieurs) bonne(s) solution(s) en un temps raisonnable, grâce aux opérateurs évolutifs. Par leur approche stochastique, les algorithmes d'évolution artificiel sont capables d'imaginer des solutions originales en balayant un large espace de recherche.

3.1 Les algorithmes évolutionnaires dans le paysage des métaheuristiques

Les AE appartiennent à la grande famille des algorithmes stochastiques et sont considérés comme des métaheuristiques (Glover, 1986; Boussaïd *et al.*, 2013). Ces derniers sont conçus pour résoudre automatiquement des problèmes d'optimisation complexes (McDowell and Popa, 2010), notamment les problèmes dits "NP-hard", fréquemment rencontrés en biologie (Wang and Jiang, 1994; Hart and Istrail, 1997; Berger and Leighton,

1998; Pevzner, 2000; Nikoloski *et al.*, 2008). Les AE sont capables de trouver une “bonne” solution en un temps de calcul raisonnable (Bianchi *et al.*, 2009) à l’inverse des algorithmes dits "exacts" (ou déterministes) qui sont utilisés pour trouver les solutions optimales, lorsqu’elles peuvent être trouvées en un temps raisonnable. Par conséquent, l’utilisation de métaheuristiques s’est répandue dans de nombreux domaines, notamment l’ingénierie, la logistique, les télécommunications, l’économie et la bio-informatique (Gogna and Tayal, 2013).

Les approches métaheuristiques peuvent être classées en deux grandes catégories. Premièrement, les algorithmes "locaux", sont basés sur une solution unique de l’espace de recherche, comme les algorithmes gloutons, les algorithmes de descente de gradient ou les algorithmes de recherche locale guidée. Deuxièmement, les algorithmes "globaux", sont principalement basés sur une population de solutions, comme les algorithmes d’essaim ou les AE. Deux concepts principaux définissent les métaheuristiques, à savoir : i) l’exploitation (ou intensification), impliquant la recherche des meilleures solutions dans le voisinage de la solution actuelle et ii) l’exploration (ou diversification), visant à identifier de nouvelles zones de l’espace de recherche contenant des solutions potentiellement satisfaisantes (Blum and Roli, 2003). Un algorithme métaheuristique efficace tente généralement de mettre en œuvre un équilibre entre ces deux concepts. Si l’on observe un excès d’exploitation, l’algorithme risque de converger prématurément. A l’inverse, si l’on observe un excès d’exploration, l’algorithme risque de ne pas converger. Parmi les AE, on dénombre cinq sous-classes principales : les stratégies d’évolution (Rechenberg, 1965), la programmation évolutionnaire (Fogel and Fogel, 1996) les algorithmes génétiques (Holland, 1992), la programmation génétique (Koza, 1992, 1994; Koza *et al.*, 1999) et l’évolution différentielle (Storn and Price, 1997).

3.2 Concepts de base des algorithmes évolutionnaires

Les AE sont des méthodes d’optimisation stochastiques inspirées des processus évolutifs darwiniens (Darwin, 1859; McDowell and Popa, 2010) et qui ont emprunté le vocabulaire à la biologie. Ils sont conçus pour explorer un large espace de recherche. Ainsi, les AE sont capables de produire des solutions originales que les humains n’auraient pas nécessairement trouvées ou imaginées (Miikkulainen, 2020) sans connaissances spécifiques des potentielles solutions. Ces solutions originales inhabituelles sont engendrées par les mécanismes évolutifs inhérents à tous les AE, comme la reproduction (croisement ou *crossover*), les mutations et la sélection (Mirjalili, 2019). En faisant évoluer une population

d'individus et en tirant parti de ces mécanismes évolutifs, les AE sont capables d'obtenir des individus (solutions) plus adaptés au problème initial. La figure 16 représente le fonctionnement classique des AE.

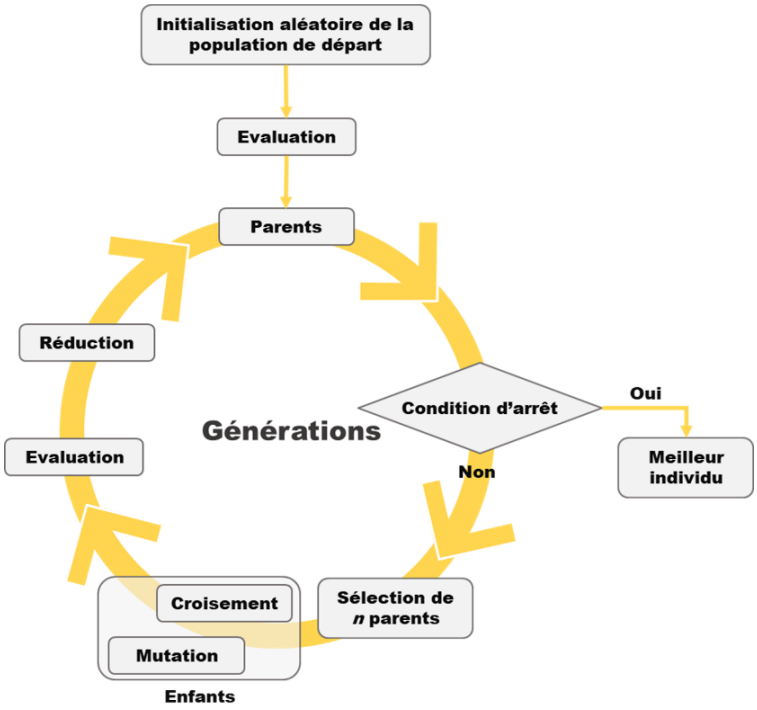


Figure 16 - *Fonctionnement général du processus des algorithmes évolutionnaires. Une population d'individus (représentant des solutions possibles au problème cible) est générée aléatoirement. Une première évaluation est effectuée sur ces individus dépendant de la fonction de fitness qui en fait des parents potentiels (un parent est un individu dont la valeur de fitness est connue). Des opérateurs évolutifs sont ensuite appliqués pour éviter de converger trop rapidement vers une solution non satisfaisante. Le croisement simule la reproduction biologique et permet à deux individus de se croiser afin que le nouvel individu hérite des caractéristiques des deux parents. Par la suite, des mutations sont appliquées à l'enfant pour introduire de la diversité. Les enfants sont ensuite évalués et une nouvelle génération est produite à partir des meilleurs individus. Si une condition d'arrêt est remplie, la meilleure solution au problème est récupérée, sinon le cycle est répété, jusqu'à ce qu'une condition d'arrêt soit atteinte. (Schéma inspiré de (Banzhaf, 2013)).*

En général, un AE initialise aléatoirement une population P de I individus (chaque individu de la population représente une solution candidate pour le problème cible), établissant ainsi la génération G_0. Il est important de noter que l'initialisation aléatoire ne signifie pas que tout peut être fait : certaines contraintes peuvent être appliquées aux valeurs aléatoires possibles, comme des intervalles. Une première évaluation est effectuée sur l'ensemble des individus (parents) dépendant de la fonction d'évaluation (aussi appelée *fitness*) mise en œuvre. Les meilleurs individus de la génération G_0 sont ensuite sélectionnés (généralement parmi ceux ayant la *fitness* la plus élevée), puis des mécanismes de croisement et de mutation (appelés opérateurs évolutionnaires ou de variation stochastique) sont appliqués, afin d'obtenir de

nouveaux individus (enfants) constituant la génération suivante G_{+1}. Le croisement combine aléatoirement des parties des parents P (avec $P \geq 2$) pour générer un nouvel individu, tandis que la mutation modifie aléatoirement une ou plusieurs caractéristiques de l'individu. Finalement, les individus ayant les caractéristiques les plus intéressantes pour le problème seront sélectionnés et persisteront dans le processus itératif. Le cycle est répété jusqu'à ce qu'une condition d'arrêt soit atteinte (nombre maximal d'itérations, solutions satisfaisantes, temps CPU (*Central Processing Unit*) consommé, etc.)

Le cœur de chaque AE est similaire et s'inspire des processus évolutifs présentés ci-dessus. Cependant, il existe des différences selon l'AE, celles-ci sont brièvement décrites dans le tableau 1 en termes de structure des individus, des opérateurs évolutionnaires (croisement et mutation) et de la méthode de sélection des meilleurs individus. Bien que de nombreuses autres stratégies aient été développées en utilisant d'autres types d'opérateurs ou de sélection (Mirjalili, 2019; Katoch *et al.*, 2021), seuls les algorithmes de programmation génétique (PG) seront présentés par la suite, car ils ont été exploités durant ces travaux de thèse.

	Stratégie d'évolution	**Programmation évolutionnaire**	**Algorithme génétique**	**Programmation génétique**	**Évolution différentielle**
Encodage des individus	Nombres réels ou vecteurs	Machine à états finis	*Bit-string* Vecteur (binaire/lettre)	Structure d'arbre	Vecteurs
Croisement	*One-point k-point*	-	*One-point*	*One-point*	Vecteur trial + vecteur aléatoire
Mutation	Adaptative	Méta-mutation	*Bit-flip*	Sous-arbre ponctuelle	3 vecteurs = 1 vecteur trial
Sélection	Déterministe	Tournoi	Rang Roulette Tournoi	Tournoi	Déterministe

Tableau *1* - *Comparaison des différentes approches d'évolution artificielle. Ce tableau a pour objectif de présenter un schéma classique des différents paramètres possibles, bien que d'autres stratégies puissent être appliquées.*

3.3 Les algorithmes de programmation génétique

Les travaux de Cramer en 1985 (Cramer, 1985) sur l'évolution d'individus avec des structures particulières telles que des arbres, des expressions ou des programmes, ont inspiré John Koza qui continua ses recherches et proposa une quatrième classe d'AE, la programmation génétique (Koza, 1992). La PG a été conçue pour faire évoluer des programmes (ou du code informatique) afin d'obtenir une solution satisfaisante (Banzhaf, 2013), comme la

détermination de l'équation mathématique d'une fonction. La PG manipule les programmes et les équations (Sloss and Gustafson, 2019) avec des opérations arithmétiques ou des fonctions mathématiques, mais ils peuvent également utiliser des expressions régulières (Bakker and Jansen, 2018). La PG est capable de générer une population de solutions qui s'améliorent grâce à l'apprentissage qu'elle réalise sur des données. Par conséquent, la PG appartient également au domaine du *machine learning* (Banzhaf *et al.*, 1998).

La représentation classique des individus en PG se présente sous la forme d'arbres (Poli *et al.*, 2008). L'opérateur de croisement le plus populaire est le croisement à un point (*one point*), qui échange au hasard une branche du parent 1 avec une autre issue du parent 2 (Banzhaf *et al.*, 1998). L'opérateur de mutation le plus couramment utilisé est la mutation de sous-arbre, qui remplace une branche de l'arbre par une autre (Banzhaf *et al.*, 1998). Enfin, la méthode la plus courante pour sélectionner les meilleurs individus est la sélection par tournoi (Poli *et al.*, 2008).

3.3.1 La fonction d'évaluation

La fonction d'évaluation (ou de *fitness*) est le centre névralgique des AE. Il s'agit d'une fonction mise en place par le développeur et qui répond au mieux à la problématique. En PG, il s'agit notamment de la faculté à prédire correctement les sorties à partir des données d'entrée. Lors de chaque étape d'évaluation, tous les individus sont confrontés à la fonction de *fitness* qui leur affecte une valeur numérique. Ainsi, les meilleurs individus se verront assigner une valeur élevée et à l'inverse les individus qui répondent le moins bien à la problématique auront une valeur faible. A l'initialisation de la population, en général, les individus ont une valeur de *fitness* très faible. Cependant, plus le processus évolutionnaire est long, plus les individus se spécialisent pour répondre au mieux à la problématique et par conséquent la valeur obtenue après application de la fonction de *fitness* augmente. Les meilleurs individus auront plus de chances d'être conservés pour la génération suivante lors de l'étape de sélection.

Définir une bonne fonction de *fitness* est une tâche ardue et nécessite une bonne connaissance du problème à résoudre. Il existe donc autant de fonctions de *fitness* que de problèmes d'optimisation. Cependant, on retrouve plusieurs méthodes classiques telles que le calcul de la précision, du gain, de classification ou de l'identité.

3.3.2 Initialisation de la population

En PG, les individus (*e.g.* un programme ou une expression régulière) sont générés aléatoirement afin que l'algorithme explore un large espace de recherche et qu'il n'ait pas d'*a priori* sur les résultats à obtenir. Cette population nécessite qu'elle soit diversifiée et de taille conséquente, afin que l'algorithme ne converge pas prématurément. En PG, les individus ont très souvent une architecture d'arbre (*i.e.* un graphe acyclique), mais d'autres travaux ont été réalisés en représentant les individus linéairement, c'est-à-dire en manipulant directement des instructions d'un langage de programmation tel que C (Brameier and Banzhaf, 2007) ou Lisp.

Les individus sous la forme d'arbres sont constitués de nœuds, reliées par des liens. Le nœud le moins profond est appelée racine, c'est lui qui est à l'origine de l'ensemble de la structure de l'individu. On dénombre différents types de nœuds : ceux qui sont composées des variables ou des constantes présentent à la couche terminale de l'arbre (également appelés feuilles) et les nœuds internes composées d'opérations. Parmi ces opérations on retrouve les fonctions arithmétiques (+, -, * et /), les fonctions mathématiques (*sin*, *cos* ou *tan*), les fonctions booléennes (*and*, *or* etc.) ou encore les fonctions conditionnelles (*if*, *then*, *else*). Un des paramètres essentiels lors de la mise en place d'un algorithme de PG est la profondeur des arbres. Une profondeur trop importante peut conduire à des solutions trop complexes et donc l'algorithme ne convergera pas. Un autre paramètre est la taille de la population qui impactera directement la taille de l'espace de recherche et donc les résultats finaux.

3.3.2.1 Initialisation par méthode grow

La méthode *grow* est une méthode d'initialisation de population en PG où les arbres sont irréguliers. Chaque individu est limité par une profondeur maximale, mais les branches ont une profondeur variable. Cette méthode induit une population initiale hétérogène. La figure 17A représente un arbre initialisé selon la méthode *grow*.

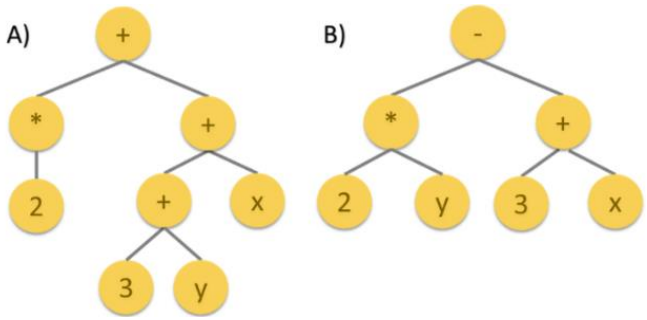


Figure 17 - *Représentation d'un individu sous la forme d'un arbre,* ***A)*** *initialisé par la méthode grow et* ***B)*** *par la méthode full.*

3.3.2.2 Initialisation par méthode full

La méthode *full* génère des arbres où les branches atteignent inéluctablement la profondeur maximale déterminée préalablement. Par conséquent, les individus initiaux constituent une population homogène. La figure 17B représente un arbre initialisé selon la méthode *full*.

3.3.2.3 Initialisation par méthode ramped half-and-half

L'initialisation de la population avec les méthodes *grow* et *full* induit une trop forte ou une trop faible diversité de la population, or comme nous l'avons vu précédemment, cela peut empêcher la convergence de l'algorithme ou l'accélérer. Par conséquent, Koza a développé une nouvelle méthode d'initialisation de population plus diversifiée : la méthode *ramped half-and-half* (**Koza 1992**), qui combine les deux approches *grow* et *full*. Cette approche génère la moitié des arbres selon la méthode *full* et l'autre moitié selon la méthode *grow*. *Ramped* signifie que la profondeur des arbres créés croit.

3.3.3 Sélection des meilleurs individus

Le processus de sélection des algorithmes de PG est inspiré de la sélection naturelle. La majeure partie des individus les plus adaptés vont intervenir dans la génération suivante. Cette adaptabilité se calcule par rapport à la fonction de *fitness*. Plus un individu obtient des performances élevées, plus il a de chances d'être sélectionné. Cependant, ce processus est stochastique et de nombreuses méthodes ont été développées afin de ne pas privilégier uniquement les meilleurs individus, ce qui pourrait limiter l'algorithme à trouver d'autres solutions encore meilleures. Ainsi, la sélection affecte directement la convergence de l'algorithme. De nombreuses méthodes de sélection ont été développées, cependant je ne présenterai que les plus répandues.

3.3.3.1 Sélection par tournoi

La sélection par tournoi (Brindle, 1980) est la méthode la plus utilisée en PG. Elle consiste à échantillonner aléatoirement k individus de la population, puis de sélectionner le meilleur individu de cette sous-population pour qu'il devienne un individu parent. On réalise ainsi j tournois de sous-population jusqu'à ce que l'ensemble des individus soit évalué. Le nombre d'individus (appelé pression de sélection) dans une sous-population est généralement

de $k = 2, 4,$ ou 7 (Fang and Li, 2010). Plus k est grand et moins les individus avec une *fitness* faible ont de chance d'être sélectionné, cependant cette méthode permet la sélection d'individus avec une *fitness* moyenne, selon le groupe aléatoire dans lequel il est sélectionné. De plus, cette méthode fonctionne aussi bien sur des architectures non parallélisables que parallélisables (Miller *et al.*, 1995). La figure 18 schématise une sélection par tournoi.

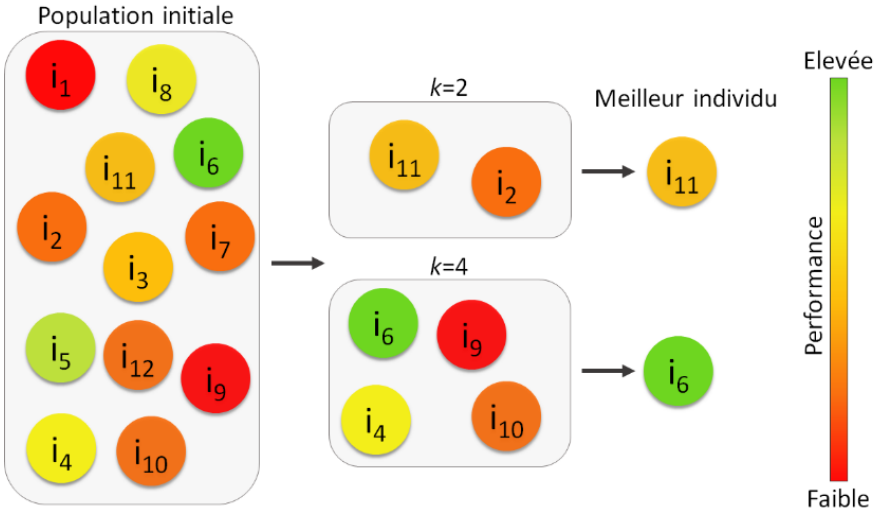


Figure 18 - *Sélection des individus par tournoi. La population initiale contient plusieurs individus avec un score de performance dépendant de la fonction de fitness (plus le score de l'individu est élevé plus sa couleur s'approche du vert). Une sélection aléatoire est effectuée sur la population initiale afin de générer des groupes de taille k. Avec k=2, on observe que les individus avec une performance moyenne peuvent être sélectionnés. En augmentant la valeur de k (pression de sélection), la probabilité de tirer aléatoirement des individus avec de hautes performances est plus élevée, limitant ainsi la sélection des individus moyens.*

3.3.3.2 Sélection par rang

La sélection par rang (Kumar and Jyotishree, 2012) dépend de la performance de chaque individu par rapport à la fonction de *fitness*. Plus la performance de l'individu est faible plus son rang est faible, ainsi l'individu le plus faible aura le rang 1 et l'individu avec la performance la plus élevée aura le rang n (où n correspond à la taille de la population). La probabilité d'être sélectionnée est ensuite calculée en fonction du rang de l'individu selon la formule :

$$probabilité\ de\ sélection\ = \left(\frac{rang\ de\ l'individu}{taille\ de\ la\ population}\right)$$

Ensuite, k individus (avec k la pression de sélection) sont sélectionnés en fonction de leurs probabilités. A l'instar de la sélection par tournoi, cette méthode de sélection favorise les meilleurs individus, mais permet aussi aux individus moyens d'être sélectionnés ce qui évite à

l'algorithme de converger trop rapidement. Un défaut de cette méthode est son coût de calcul important dû au tri de la population (Razali and Geraghty, 2011).

3.3.3.3 Sélection par roulette

La méthode de la roulette est un tirage aléatoire favorisant également les individus avec un score de *fitness* élevé. Cependant, cette méthode laisse plus de chances aux individus faibles d'être sélectionnés par rapport aux autres méthodes. Classiquement, une roulette virtuelle est générée et contient autant de segments que d'individus. La taille de ces segments est représentative du score de *fitness* des individus. Ainsi, un individu avec un score élevé aura un grand segment et un individu avec un score faible se verra attribuer un petit segment. La roulette virtuelle est ensuite activée et l'individu sélectionné est celui sur lequel le pointeur de la roulette s'est arrêté (Kumar and Jyotishree, 2012). La figure 19 représente la sélection des individus par la méthode de la roulette.

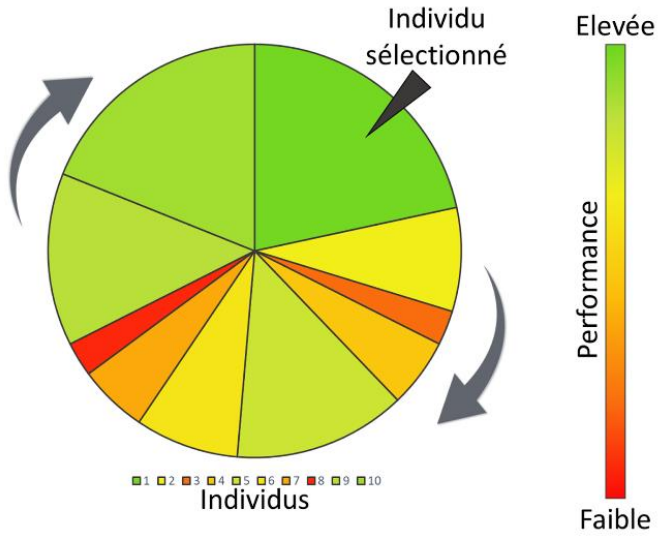


Figure 19 - *Sélection des individus par roulette. Un individu est sélectionné pour la génération suivante selon la méthode de la roulette. La taille de chaque segment de la roulette est proportionnelle au score de fitness des individus. La sélection par roulette favorise ainsi les meilleurs individus, mais n'empêche pas la sélection d'individus plus faibles.*

3.3.4 L'opérateur de croisement

L'encodage des individus par des arbres facilite la mise en place des opérateurs évolutionnaires, dont le croisement qui est un des plus utilisés en PG. En général, 90% des individus de la population réalisent un croisement avec un autre individu (Banzhaf *et al.*, 1998). L'opérateur de croisement consiste à combiner deux sous-arbres de deux parents différents. Il existe plusieurs méthodes de croisement (*k-point crossover*, *shuffle crossover*, *uniform*

crossover) (Umbarkar and Sheth, 2015), mais nous aborderons principalement celle à un point (*one-point*) qui est la plus utilisée en PG. La méthode de croisement à un point, sélectionne aléatoirement une région de l'arbre du parent 1, afin d'extraire un sous-arbre et de l'échanger avec un sous-arbre du parent 2. Les deux individus résultants après l'étape de croisement sont les individus enfants. Le croisement permet notamment d'enrichir la population en générant des individus plus diversifiés. De plus, avec la sélection des meilleurs individus, les caractéristiques importantes qui favorisent ces individus sont préservées. Cependant, bien que cette méthode puisse explorer de manière globale l'espace de recherche initial, elle tend au cours du processus évolutif à devenir locale et n'explore donc plus de manière générale l'espace de recherche (Poli and Langdon, 1998). La figure 20 représente la méthode de croisement à un point entre deux individus parents.

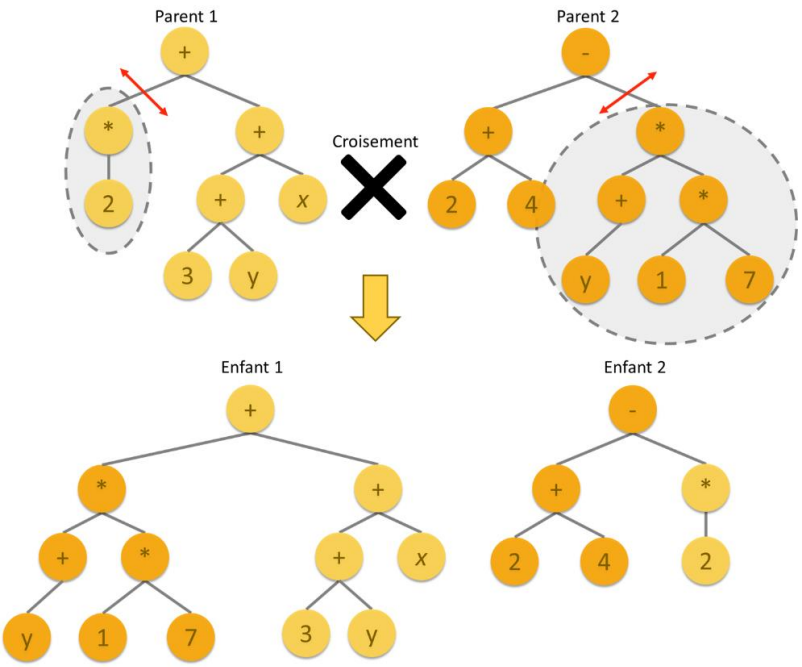


Figure 20 - *Processus de croisement à un point. Un sous-arbre du parent 1 et du parent 2 est sectionné de l'arbre initial au niveau d'un point de coupure (flèche rouge) choisi aléatoirement. Le sous-arbre du parent 1 va ensuite s'intégrer au niveau du point de coupure du parent 2 (et vice versa) générant ainsi deux individus enfants, avec des caractéristiques héritées des individus parents. Ces caractéristiques peuvent être importantes et favoriser les individus lors de la sélection, la réciproque est aussi vraie.*

3.3.5 L'opérateur de mutation

La mutation consiste à générer un individu enfant en induisant une mutation aléatoire à un individu parent. L'opérateur est cependant moins présent dans la population que l'opérateur

de croisement. Environ 1% des individus sont susceptibles de subir une mutation à chaque nouvelle génération. La mutation permet d'explorer l'espace de recherche de manière plus précise. En effet, elle induit l'apparition de nouvelles caractéristiques qui peuvent potentiellement surpasser les meilleures caractéristiques qui sont transférées de génération en génération par l'opérateur de croisement. On dénombre également de nombreuses techniques de mutation comme la méthode par permutation, la *hoist mutation*, la *shrink mutation*, mais dans cette partie nous allons uniquement parler de la mutation de sous arbres (*sub-tree mutation)* et de la mutation à un point (*one-point mutation*) (Poli *et al.*, 2008).

La mutation de sous-arbre, introduite par Koza (Koza, 1992), est la méthode de mutation la plus utilisée en PG. Elle consiste à remplacer un sous-arbre par un autre (figure 21A). La sélection du point de coupure du sous-arbre se fait aléatoirement. De plus, chaque nouveau sous-arbre de l'individu, à une profondeur (en général) équivalente à la profondeur du sous-arbre qu'il remplace. La mutation ponctuelle (ou remplacement de nœud) consiste à sélectionner aléatoirement un nœud de l'arbre et de le modifier (figure 21B).

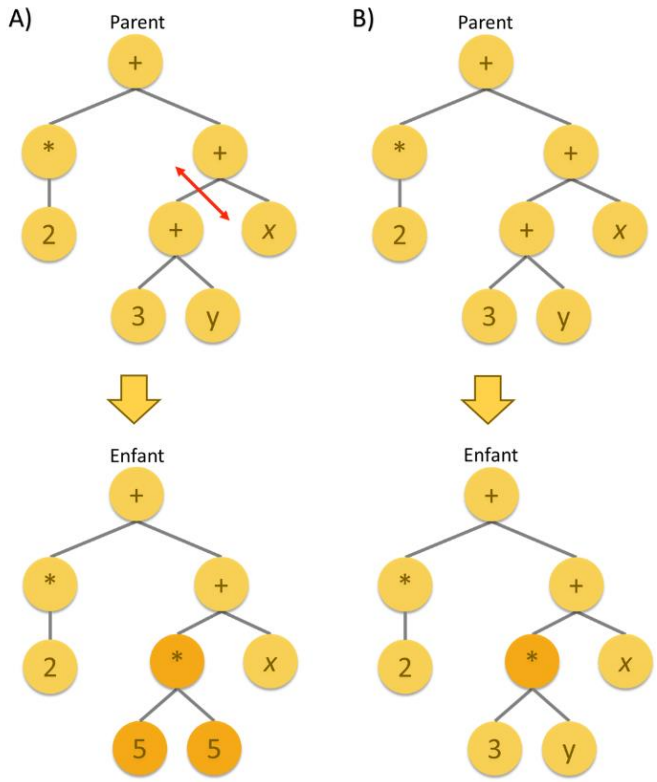


Figure 21 - *Représentation de deux méthodes de mutation en programmation génétique.* ***A)*** *La mutation de sous-arbre consiste à sectionner un sous-arbre de l'individu parent (flèche rouge) et à le remplacer par un autre sous-arbre de taille équivalente générant un nouvel individu enfant.* ***B)*** *La mutation ponctuelle sélectionne aléatoirement un nœud de l'individu parent et le modifie, ce qui induit la création d'un nouvel individu enfant.*

3.4 Utilisation des algorithmes évolutionnaires

De nombreux domaines utilisent aujourd'hui des AE en raison de leur nature stochastique, imprévisible et inventive. Parmi ces domaines on retrouve la finance (Duyvesteyn and Kaymak, 2005), les jeux vidéo (Azaria and Sipper, 2005; Martínez-Arellano *et al.*, 2017), l'ingénierie (Ebrahimzade *et al.*, 2018; Hien *et al.*, 2020) ou la bio-informatique (Liu and Xu, 2009; Niazkar and Reza, 2020). Mais l'application la plus célèbre et médiatisée est probablement l'évolution de la forme d'une antenne (figure 22) pour la mission *Space Technology 5* de la NASA (Lohn *et al.*, 2004).

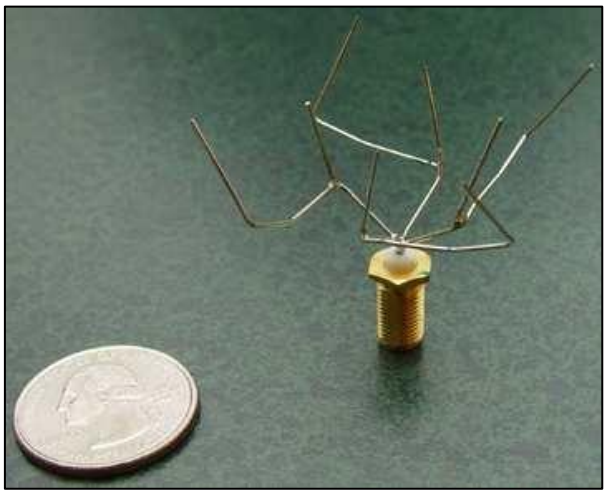


Figure 22 - *Photo du prototype de l'antenne du projet ST5 de la NASA déterminé par évolution artificielle (source : (Lohn et al., 2004)).*

En bio-informatique de nombreux travaux ont été réalisés notamment pour la prédiction de structure de protéines (Pedersen and Moult, 1996; Rout *et al.*, 2017; Boumedine and Bouroubi, 2019), les alignements multiples de séquences (Notredame and Higgins, 1996; L. Cai *et al.*, 2000; Nizam *et al.*, 2011; Chowdhury and Garai, 2017; Behera *et al.*, 2017), la prédiction de gènes (Chowdhury *et al.*, 2017) ou d'éléments fonctionnels (Wang and Lichodzijewski, 2005), la prédiction de pathologie (Paul and Iba, 2009) ou encore la prédiction de structure des ARN (Lee and Han, 2003). Bien que peu exploités, les AE sont une solution intéressante et bien adaptée pour des problèmes de bio-informatique, car ils nécessitent très peu de connaissances initiales de ce problème. J'ai donc décidé d'exploiter la puissance et les avantages des algorithmes de PG lors de mes travaux de thèse afin d'améliorer un des problèmes majeurs de la bio-informatique : l'annotation des génomes eucaryotes.

Chapitre 4 - L'annotation des génomes : un cas d'application pour l'IA en bio-informatique

Comme nous l'avons vu dans les Chapitres 2 et 3, l'IA est utilisée dans de nombreux champs d'applications dont la biomédecine et la bio-informatique. Les succès réalisés dans ces domaines sont dus à l'explosion de la capacité de calcul (Moore, 1965), de la parallélisation sur GPU, des nouvelles technologies, mais surtout à l'avènement du *Big Data*. Aujourd'hui, les données sont la pierre angulaire des algorithmes d'IA et sont devenues le nouvel "or noir" du XXI^{e} siècle. Posséder les données, c'est pouvoir comprendre et prédire les éléments de son environnement. Pour illustrer ces propos, il suffit de voir que les quatre entreprises du GAFA (Google, Amazon, Facebook et Apple) sont parmi les plus puissantes et sont des entreprises exploitant des données massives.

4.1 L'univers du *Big Data*

Le *Big Data* (Stephens *et al.*, 2015) est le regroupement massif de données qui nécessite des technologies de pointe pour son exploitation. Ce phénomène a ouvert la voie à de nouvelles opportunités et a révolutionné notre approche du monde. Les nouvelles technologies du numérique (web, *cloud computing*, *Internet of Things*, séquenceurs, etc.) produisent une quantité colossale de données. Cette datasphère s'étend quotidiennement de plusieurs milliers de tera octets (10^{12}) (Altaf-Ul-Amin *et al.*, 2014; Pal *et al.*, 2020), générant de nouveaux défis pour stocker, traiter et exploiter ces données hétérogènes. Initialement, le *Big Data* s'est caractérisé par ce qu'on appelle les 3V : Volume (pour la quantité importante des données), Variété (pour la diversité du type de donnée) et Vélocité (pour la vitesse de production et de traitement des données) (Sagiroglu and Sinanc, 2013) (figure 23). Cependant, aujourd'hui la définition peut s'étendre aux 4V avec l'ajout de la Véracité (pour la qualité des données), et même aux 5 ou 6V avec la Valeur et la Visualisation. Certains cherchent même à ajouter le plus de V possible (Panimalar, 2017), ce qui tend à dégrader la définition même du *Big Data*.

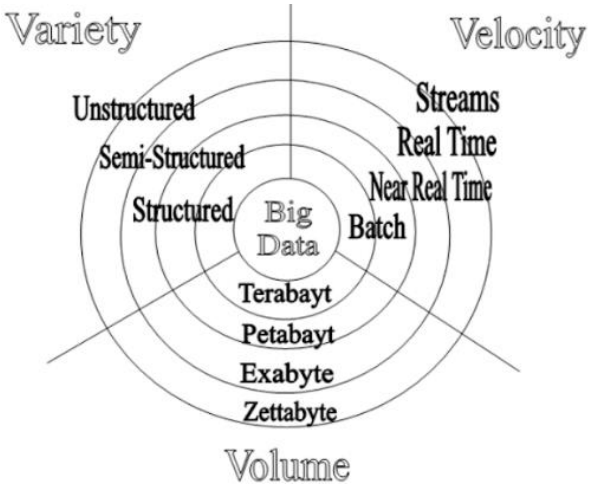


Figure 23 - *Les différentes strates du Big Data. Initialement le Big Data a été caractérisé par les 3V : La Variété pour l'hétérogénéité des données, la Vélocité pour la vitesse d'acquisition et de traitement des données, et le Volume pour la quantité astronomique de données (source : (Sagiroglu and Sinanc, 2013)).*

4.1.1 L'ère des -omiques

Des premières séquences du bactériophage φX174 par Frederick Sanger en 1977 (Sanger *et al.*, 1977) aux nouvelles technologies de séquençage ou technologies à haut débits, tel que le séquenceur ultra-portable SmidgION de Oxford Nanopore (nanoporetech.com/products/smidgion), la quantité de données produite n'a cessé d'augmenter en biologie. Il s'agit d'une des sciences les plus récentes qui arborent fièrement l'étendard du *Big Data*, depuis que le séquençage du génome humain (Venter *et al.*, 2001; Lander *et al.*, 2001), qui fête son 20^{e} anniversaire cette année (Gates *et al.*, 2021), a été réalisé. Cette quantité massive de données s'étend aujourd'hui sur toutes les strates biologiques, ce qui a propulsé la biologie à l'ère des "omiques".

Le suffixe "omique" désigne "l'ensemble de", ainsi la génomique (mère de toutes les omiques) signifie l'ensemble du matériel génétique dont les gènes. S'en sont suivi toutes les déclinaisons permettant d'étudier les systèmes biologiques. On retrouve ainsi la métagénomique, l'épigénomique, la transcriptomique, la protéomique, la métabolomique, la lipidomique, ainsi que l'interactomique, la mitointeractomique (interactome des protéines mitochondriales) (Reja *et al.*, 2009), mais également la variomique, la diseasomique ou la phénomique. Dans la suite de ce chapitre, je décrirai plus en détail les données génomiques et leur traitement, puisque j'ai pris comme cas d'étude pendant cette thèse, l'annotation des génomes eucaryotes.

L'ère des omiques a déclenché le lancement d'une pléthore de projets de séquençage afin d'étudier et de comprendre plus finement les organismes et les systèmes biologiques, de déterminer les enjeux écologiques, et d'étudier les relations génotype-phénotype[3]. Le séquençage quasi-complet du premier génome humain, aujourd'hui considéré comme patrimoine de l'humanité, a duré plus de 13 ans et a coûté environ 3 milliards de dollars (soit presque 1$/nucléotide) (Li and Chen, 2014). Ce projet a réuni plus de 350 laboratoires de 18 pays différents. Vingt ans après, le génome complet a été annoncé (les régions centromériques complexes ont notamment été résolues) (Nurk *et al.*, 2021) et séquencer un génome coûte maintenant la modeste somme de 1 000$ et prend quelques jours voire quelques heures. Cette réduction drastique du coût et l'augmentation de la vitesse de séquençage ont initié le lancement d'autres projets tels que le projet 1 000 génomes (Durbin *et al.*, 2010) qui vise à analyser les variants présents dans le génome de 1 000 individus afin d'identifier des relations génotype-phénotype. Un projet encore plus ambitieux est celui des Britanniques, le projet 100 000 génomes (Samuel and Farsides, 2017) qui a pour objectif de séquencer 100 000 individus et d'identifier les relations génotype-phénotype impliquées dans des maladies rares ou des cancers. En dehors du domaine médical, d'autres projets ont été initiés sur plusieurs espèces ou clades, tels que le projet G10K pour le séquençage du génome de 10 000 vertébrés (Genome 10K Community of Scientists, 2009), le projet GIGA qui a pour objectif de séquencer 7 000 espèces marines invertébrées (GIGA Community of Scientists, 2014), le projet i5K qui se focalise sur les espèces d'arthropodes (The i5K consortium, 2013), le projet 1 000 fungi est spécifique aux champignons (Grigoriev *et al.*, 2014) ou le projet 10KP, qui à long terme séquencera le génome de 10 000 plantes (Cheng *et al.*, 2018). D'autres projets ont également été lancés comme le projet Cephseq (pour les céphalopodes) (Albertin *et al.*, 2012) ou le projet 10 000 protistes (Miao *et al.*, 2020).

La principale "omique" exploitée durant ces travaux est la génomique. Comme précisé précédemment, la génomique est la science qui étudie le génome. Son histoire est couplée à celle de la découverte de la structure de l'ADN et du séquençage. En 1953, James Watson et Francis Crick (Watson and Crick, 1953) ont déterminé la structure en double hélice de l'ADN, défini comme le support de l'information génétique quelques années auparavant (Avery *et al.*, 1944; Hershey and Chase, 1952). La structure en double hélice a notamment pu être résolue grâce au cliché de diffraction numéro 51 (figure 24) de Rosalind Franklin (Braun *et al.*, 2011).

3 Le génotype correspond à la composition allélique de tous les gènes d'un individu et le phénotype correspond à l'expression du génotype de cet individu, *i.e.* tous les caractères observables.

Aujourd'hui, la génomique se scinde en différentes approches, la génomique structurale qui définit l'organisation du génome et la structure des gènes, la génomique fonctionnelle qui a pour objectif de déterminer la fonction des gènes ou encore la génomique comparative qui compare le génome d'organismes différents pour identifier des mécanismes évolutifs par exemple. L'ajout d'informations sur un génome est un processus appelé annotation sur lequel nous reviendrons dans la partie 4.4 de ce chapitre, car il s'agit d'un point essentiel de mes travaux.

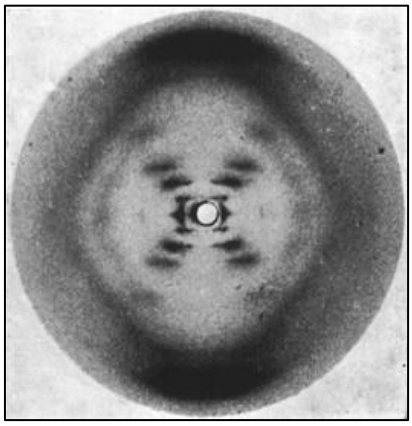


Figure 24 - *Cliché de diffraction numéro 51, pris en 1951 par Rosalind Franklin qui a été déterminant pour l'identification de la structure en double hélice de l'ADN (source : (Braun et al., 2011)).*

4.1.2 Les défis du *Big Data*

L'avènement des technologies à haut débit a propulsé la biologie dans l'ère du *Big Data* et de nombreux progrès ont pu être réalisés. Cependant aujourd'hui encore nous faisons face à de nombreux défis. Ces données doivent être traitées, analysées et stockées, mais leur volume important rend ces tâches complexes. Un des principaux défis consiste à extraire les données en sélectionnant la source la plus pertinente. En effet de nombreuses bases de données sont aujourd'hui disponibles telles que la DDBJ (Kaminuma *et al.*, 2011), Genbank (Benson *et al.*, 2009), Gene expression omnibus (Clough and Barrett, 2016), *Genotype-Tissue Expression* (Lonsdale *et al.*, 2013), Uniprot (The UniProt Consortium, 2017), Prosite (Sigrist *et al.*, 2013), la *Protein Data Bank* (Bernstein *et al.*, 1977), *Kyoto Encyclopedia of Genes and Genomes* (Kanehisa and Goto, 2000) ou String (Von Mering *et al.*, 2003). Certaines infrastructures actuelles peuvent favoriser le stockage massif et la rapidité d'accès des données au détriment de l'organisation et de l'homogénéité de ces dernières. C'est le cas des lacs de données (*datalake*) par exemple. De plus, de nombreuses ressources disponibles lors de la publication sont obsolètes au bout de deux ans. Un autre défi que l'on rencontre est le traitement, l'analyse,

l'interprétation ainsi que la visualisation de ces données massives. Cette étape nécessite de nombreux outils qui demandent une puissance de calcul et des ressources en mémoire importantes. De plus, ce processus peut être extrêmement chronophage. Après l'extraction et le traitement des données, il faut prendre en compte leur stockage. Outre le fait que le téléchargement peut être relativement long en fonction de la technologie de transfert, la capacité de stockage peut être importante et devenir un facteur limitant. Bien que le prix de la mémoire reste abordable de nos jours, la facture peut rapidement augmenter avec des données massives. De nouvelles technologies comme le stockage massif sur *cloud* (Wu *et al.*, 2010) offrent d'autres perspectives, mais ces services ont un coût non négligeable. L'ADN va également jouer un rôle, non plus pour être le support de l'information génétique, mais pour stocker des données telles que des vidéos, des images etc. (Church *et al.*, 2012). Une autre solution pour lutter contre l'augmentation croissante des données est de les compresser (Hsi-Yang Fritz *et al.*, 2011), par exemple en génomique le format BAM (*Binary Alignment Map*) inclut des données binaires brutes compressées (Li *et al.*, 2009), mais cette compression/décompression peut être chronophage. Un autre problème, plus éthique, lié aux données est leur sécurité. En effet, les données biomédicales sont souvent sensibles et nécessitent une attention particulière en termes d'anonymisation et de sécurisation. La fuite de données est un problème important, et on dénombre de nombreuses attaques, même sur de célèbres réseaux sociaux où les données ont fuité[4]. Cela soulève donc des problèmes liés à la confidentialité et à la protection des données, à l'éthique ou à l'échange entre pairs (dans le cadre de la médecine par exemple). En 2016, le Règlement Général sur la Protection des Données (RGPD) a été instauré pour protéger les données à caractère personnel symbolisant leur valeur et leur fragilité. Enfin, un problème moins tangible et qui a tendance à passer inaperçu est la consommation énergétique et la pollution engendrée par les calculs et le stockage intensif (Wu *et al.*, 2016).

Le dernier problème sur lequel j'aimerai revenir, car il a joué un rôle prépondérant durant ma thèse, est la qualité des données. La production massive de données brutes engendre forcément des erreurs, comme lors du séquençage où l'on estime un taux d'erreurs de 0,01% à 0,1% (Lou *et al.*, 2013). Les erreurs générées peuvent ensuite se propager dans les bases de données (Gilks *et al.*, 2002; Promponas *et al.*, 2015). On estime, par exemple, que 25 à 60% des données présentes dans les bases de données publiques sont erronées (Devos and Valencia, 2001; Schnoes *et al.*, 2009) et ces erreurs peuvent ensuite impacter les expériences en aval (Gilks *et al.*, 2005; Meyer *et al.*, 2020; Zhang *et al.*, 2020). Il est donc indispensable de prendre

4 siecledigital.fr/2021/04/04/facebook-donnees-533-millions-utilisateurs

en considération que les données peuvent être erronées même si elles sont extraites de bases de données publiques. De plus, identifier puis corriger ces erreurs est à ce jour un des grands défis de la bio-informatique bien que des experts tentent de répondre à cette problématique par l'intermédiaire de bases de données de haute qualité comme Swiss-Prot. Malheureusement, ce processus est long et atteint qu'une faible portion des données, en effet Swiss-Prot ne correspond qu'à 0,3% d'Uniprot.

Malgré ses défauts, le *Big Data*, a ouvert la porte à de nombreuses applications, dans de multiples domaines et notamment en IA. De nombreux algorithmes basés sur l'IA et exploitant les données biologiques multi-omiques ont vu le jour comme des programmes d'annotation de génome basés sur des prédicteurs de structures et de fonction des protéines, de variants pathogènes, de gènes ou encore de sites d'épissage. Ainsi, la bio-informatique et notamment l'annotation des génomes est un exemple parfait qui combine les données multi-omiques du *Big Data* et les algorithmes d'IA capables d'exploiter ces données massives.

4.2 Le génome eucaryote

Dans la suite de ce chapitre je reprends les éléments fondamentaux du génome eucaryote et principalement ceux des sites d'épissage (SE).

Le génome est l'ensemble du matériel génétique d'un individu encodé dans son ADN et correspond au support de l'information génétique transmise de génération en génération. Chez les organismes eucaryotes, le génome est divisé en deux, on retrouve le génome nucléaire et le génome non nucléaire, incorporé dans les organites (mitochondries et/ou chloroplastes). Le génome nucléaire est organisé en chromosomes linéaires et le nombre de chromosomes (correspondant à la ploïdie) est dépendant de chaque espèce. Ainsi, l'humain est diploïde, c'est-à-dire que son génome est composé de $2n$ chromosomes (22 paires de chromosomes autosomes et 1 paire de chromosomes sexuelles, soit 46 chromosomes au total). Certaines espèces peuvent être haploïdes (un seul exemplaire), dont principalement des champignons comme *Plectosphaerella cucumerina* (Uecker, 1993), ou polyploïde ($n > 2$ exemplaires) comme certaines espèces de blé (Kerby and Kuspira, 1987).

4.2.1 Taille des génomes

La taille du génome est propre à chaque espèce et n'est pas corrélée avec sa complexité. Par exemple, le génome d'*Arabidopsis thaliana* fait 119,75 millions de paires de bases (pb) (ncbi.nlm.nih.gov/genome/?term=txid3702) et encode environ 25 000 gènes codant pour des protéines, tandis que le génome humain à une taille de 3,3 milliards de pb et encode environ 20 000 gènes codant pour des protéines (Nurk *et al.*, 2021). L'organisme eucaryote avec le plus petit génome connu est actuellement *Encephalitozoon intestinalis* avec une taille de 2,25 millions de pb et le plus grand est probablement *Paris japonica* avec une taille de génome de 148,8 milliards de pb (Hidalgo *et al.*, 2017), bien que des estimations du génome de l'amibe *Amoeba dubia* ont suggéré que sa taille atteindrait près de 700 milliards de pb (Hidalgo *et al.*, 2017; Friz, 1968). Bien souvent les organismes végétaux ont un très grand génome, car ils ont subi des événements de duplication durant l'évolution (Ibarra-Laclette *et al.*, 2013). La figure 25 présente une liste non exhaustive de la taille des génomes de plusieurs organismes de clades différents.

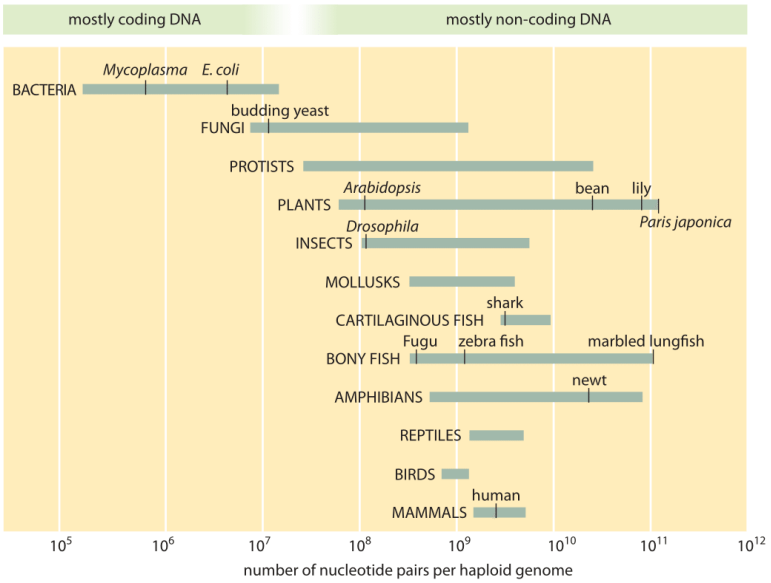


Figure 25 - *Représentation de la taille des génomes de quelques organismes eucaryotes et procaryotes. Les angiospermes et les protozoaires se caractérisent par un large spectre de tailles de génomes différents (source : book.bionumbers.org/how-big-are-genomes/).*

4.2.2 Composition des génomes

Les génomes eucaryotes sont extrêmement complexes, ils contiennent de nombreux signaux encodés, ou d'autres qui sont dépendants de sa conformation ou de facteurs de

régulation. Le génome est organisé par des enchaînements de régions dites "intergéniques" et "géniques", bien que cette définition soit discutable dans certains cas. Les régions "géniques" sont composées de gènes qui peuvent coder pour des protéines (ARN messager (ARNm)) ou pour d'autres ARN (Djebali *et al.*, 2012) tels que des ARN de transfert (ARNt), des ribozymes, des ARN ribosomiques (ARNr), des petits ARN (*snoRNA*), des micro ARN (*miRNA*), ou encore des ARN interférents (*siRNA*). D'autres éléments participent à la composition du génome comme des séquences répétées telles que les SINE (*short interspersed elements*), les LINE (*long interspersed elements*) ou les transposons. Ces éléments répétés participent à l'extension du génome et on estime qu'au moins 40% du génome humain dérive d'éléments transposables (Smit, 1999). Par exemple, une séquence répétée nommée *Alu* est fortement présente dans le génome humain et correspond à 11% de ce dernier (Lander *et al.*, 2001). Les éléments fonctionnels et régulateurs font également partie intégrante des génomes. Le génome est donc un chef-d'œuvre du vivant et une encyclopédie nommée ENCODE (*ENCyclopedia Of Dna Elements*) (Dunham *et al.*, 2012) a été établie répertoriant les connaissances actuelles du génome et de ces différents éléments. Il serait beaucoup trop long de détailler l'ensemble de ces éléments, c'est pourquoi je continue mon exposé uniquement sur la présentation des gènes codant pour des protéines, qui ont été au cœur de mon travail.

4.3 Les gènes codant pour des protéines

Les gènes codant pour des protéines sont les principaux éléments impliqués dans le phénotype de l'individu. Comme leur nom l'indique, ces gènes codent pour des protéines et ces dernières sont issues de la traduction d'ARNm. Chez l'humain cela représente ~2-3% (exome uniquement) de son génome, chez la drosophile 17% du génome, chez *C. elegans* 25% du génome et chez la levure 70% du génome (Cooper, 2000). La figure 26 représente l'architecture d'un gène typique codant pour une protéine.

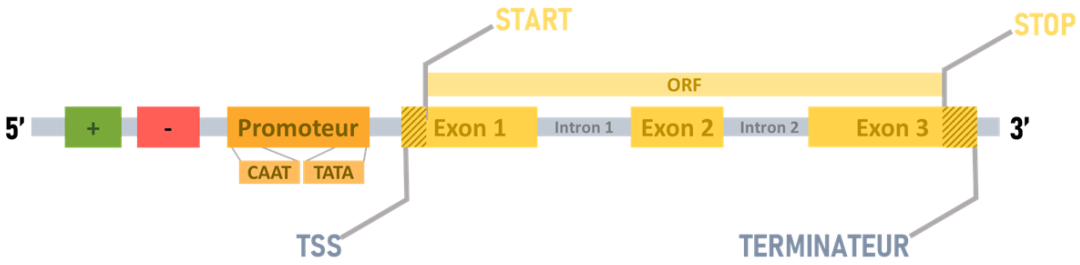


Figure 26 - *Architecture d'un gène eucaryote codant pour une protéine. Les éléments activateurs sont représentés par la boîte verte et les éléments inhibiteurs par la boîte rouge. Le cadre de lecture ouverte*

(ORF) représente la région entre un codon start et stop. Les régions hachurées représentent les régions 5' et 3' UTR.

4.3.1 Structure des gènes codant pour des protéines

4.3.1.1 Le promoteur

Les gènes codant pour des protéines présentent généralement une région non transcrite appelée promoteur à l'extrémité 5' (avant le site d'initiation de la transcription (TSS)). Son rôle est de fixer l'ARN polymérase II (Hahn, 2004), afin d'initier la transcription grâce à plusieurs co-facteurs appelés facteurs de transcription (Orphanides *et al.*, 1996). La composition du promoteur est modulable et peut varier en fonction du gène et de l'espèce. On y retrouve différents éléments signalétiques principaux tels que la boîte TATA (séquences d'ADN TATAAAA) (Pedersen *et al.*, 1999) qui se trouve à environ 30 nucléotides en amont du TSS (Breathnach and Chambon, 1981) et qui a pour rôle de guider l'ARN polymérase II en fixant le facteur de transcription TFIID (Horikoshi *et al.*, 1990). On retrouve aussi la boîte CAAT (séquence consensus GGCCAATCAG) (Dolfini *et al.*, 2009) qui se situe entre 50 et 120 nucléotides en amont du TSS (Hatamochi *et al.*, 1988). Ce motif permet également de fixer des facteurs de transcription pour guider l'ARN polymérase II. Un troisième élément du promoteur est la boîte GC qui est généralement présente entre la boîte TATA et CAAT. Elle peut être présente en une ou plusieurs copies et a pour rôle de fixer le facteur de transcription *Sp1* (Jankowski and Dixon, 1987). Les derniers éléments que j'aimerais présenter sont les îlots CpG. Ils correspondent à une zone riche en nucléotides G et C, généralement organisés en petits groupes et ils sont principalement situés à l'extrémité 5' du gène, bien que certains soient présents en 3' (Gardiner-Garden and Frommer, 1987). Les îlots CpG sont présents dans environ 70% des promoteurs (Saxonov *et al.*, 2006) et on en retrouve environ 30 000 dans le génome humain. Ces régions sont fortement résistantes à la méthylation (Cooper *et al.*, 1983) et sont souvent associées à une activité transcriptionnelle élevée (Papin *et al.*, 2021).

Comme précisé précédemment, le promoteur est modulable (Butler and Kadonaga, 2002) et l'ensemble de ces éléments ne sont pas nécessaires pour l'initiation de la transcription. Par exemple, la boîte TATA qui est un des éléments principaux du promoteur est absente chez certains gènes humains (Suzuki *et al.*, 2001).

4.3.1.2 Les exons et les introns

Le cœur des gènes codant pour des protéines est situé entre le TSS et le terminateur, élément qui interrompt la transcription. Ces bornes définissent la région transcrite qui se composent de deux éléments principaux : les exons (*EXpressed regiONs*) et les introns (*INTRagenic regiONs*) (Gilbert, 1978). Ces derniers sont principalement présents chez les organismes eucaryotes, bien que l'on retrouve des introns particuliers chez certains procaryotes (Ferat and Michel, 1993; LaRoche-Johnston *et al.*, 2018). Les gènes codant pour des protéines sont ainsi organisés en mosaïque d'exons et d'introns.

Les exons sont les segments que l'on retrouve généralement dans l'ARNm à la suite du processus de transcription de l'ADN (Crick, 1958), puis du processus d'épissage. La fonction des exons, découverte fin des années 70 (Gilbert, 1978), est de coder, selon une suite de codons (triplet de nucléotides), l'information qui sera traduite par le ribosome afin de générer une protéine. L'ensemble de la séquence traduite qui permet la production d'une protéine est appelée CDS (pour *Coding DNA Sequence*). Cette dernière est bornée par un codon start (AUG) également appelé séquence Kozak (Kozak, 1981) et un codon stop (ocre: UAA ; opale : UGA ou ambre : UAG). Douze catégories ont été établies afin de classer les exons (tableau 2).

Nom	Description
5uexon	Exon en 5' entièrement non traduit (5' UTR)
3uexon	Exon en 3' entièrement non traduit (3' UTR)
5utexon	Exon en 5' avec un segment 5' UTR puis une CDS
3tuexon	Exon en 3' avec une CDS puis un segment 3' UTR
iutexon	Exon interne ayant un segment 5' UTR suivie d'une CDS
ituexon	Exon interne ayant une CDS suivie d'un segment 3' UTR
iuexon	Exon interne entièrement non traduit
itexon	Exon interne entièrement traduit
5utuexon	Exon mixte en 5' avec un segment 5' UTR puis une CDS puis un segment 3' UTR
3utuexon	Exon mixte en 3' avec un segment 5' UTR puis une CDS puis un segment 3' UTR
5-3utuexon	Mono-exon avec un segment 5' UTR puis une CDS puis un segment 3' UTR
iutuexon	Exon mixte interne avec un segment 5' UTR puis une CDS puis un segment 3' UTR

Tableau 2 - *Classification des exons en fonction de leurs caractéristiques de traduction (inspirée de (Zhang, 1998))*

Une des particularités des gènes codant pour des protéines est la présence de régions non traduites (UTR pour *UnTranslated Region* en anglais) qui jouent un rôle dans la régulation, la stabilité et l'efficacité de la traduction des ARNm (Pesole *et al.*, 2000). Ainsi, en raison de la présence des UTR, il ne faut pas considérer les exons comme des séquences codantes uniquement. Ces régions sont situées aux extrémités 5' (proche du TSS) et 3' (proche du terminateur) du gène. Elles peuvent recouvrir partiellement un exon ou recouvrir entièrement

un ou plusieurs exons (induisant alors le recouvrement d'intron), ou être absentes. En moyenne chez l'Homme, on retrouve 8,8 exons par gène (Tazi *et al.*, 2009), avec des gènes ayant un seul exon comme le gène *CDR1* (Jorquera *et al.*, 2018), ou jusqu'à 363 exons pour le gène de la titine (Bang *et al.*, 2001).

Les introns sont les deuxièmes éléments qui composent le cœur des gènes avec les exons. Historiquement, les introns étaient considérés comme de l'ADN "junk" ou poubelle, vestige de l'évolution (Ohno, 1972). Cependant, nous savons aujourd'hui qu'ils jouent un rôle essentiel au niveau de la transcription (en augmentant ou diminuant son efficacité), de l'exportation des ARNm et de leur stabilité (Shaul, 2017). Les introns sont en outre porteurs de signaux importants pour le processus d'épissage tel que les SE (Shapiro and Senapathy, 1987) ou le point de branchement (Mercer *et al.*, 2015). De plus, ils régulent ce processus par l'intermédiaire de sites spécifiques guidant des protéines inhibitrices ou activatrices de SE (Tange *et al.*, 2001; Lee and Rio, 2015). Les introns sont des segments transcrits en pré-ARNm puis excisés, ils sont donc dits "non-codants" bien que lors du processus d'épissage une partie de l'intron peut ne pas être supprimé (rétention d'intron) et participer à la construction du transcrit mature et donc de la protéine (Vanichkina *et al.*, 2018). De plus, certains introns peuvent coder pour d'autres gènes : on parle alors de gènes imbriqués (ou *nested genes*) (Makałowski, 2001; Ben Khalaf *et al.*, 2019).

4.3.1.3 Les sites d'épissage (SE)

Les SE correspondent aux frontières entre les exons et les introns. Ils sont reconnus par le spliceosome, un complexe ribonucléoprotéique, qui excise les introns et suture les exons (Matera and Wang, 2014). On dénombre deux types de SE : le donneur (5') et l'accepteur (3'). Le SE donneur se situe à la jonction entre l'exon et l'intron, tandis que le SE accepteur se situe à la jonction entre l'intron et l'exon. Ces SE se caractérisent par la présence d'un dinucléotide : le dinucléotide GT spécifique aux SE donneurs et le dinucléotide AG spécifique aux SE accepteurs. Ces dinucléotides dits "canoniques" sont hautement conservés (Burset *et al.*, 2000) et représentent plus de 98,3% des SE chez les animaux, 98,7% chez les champignons et 97,9% chez les plantes (Frey and Pucker, 2020). Ces dinucléotides sont également incorporés dans un motif plus large : aG**GT**AAGT pour le SE donneur et (Y)6N(C/t)**AG**(g/a)t pour le SE accepteur (figure 27A et C) (Nguyen *et al.*, 2018). La région qui précède le SE accepteur est riche en pyrimidines (C/T) et est souvent désignée par le terme "tractus polypyrimidique" (Will and Lührmann, 2011). Dans certains cas, on retrouve des dinucléotides divergents du consensus

alors appelés "non-canoniques". La découverte d'un petit nombre d'introns contenant des dinucléotides non-canoniques aux jonctions d'épissages a conduit à la caractérisation d'un spliceosome variant (Turunen *et al.*, 2013).

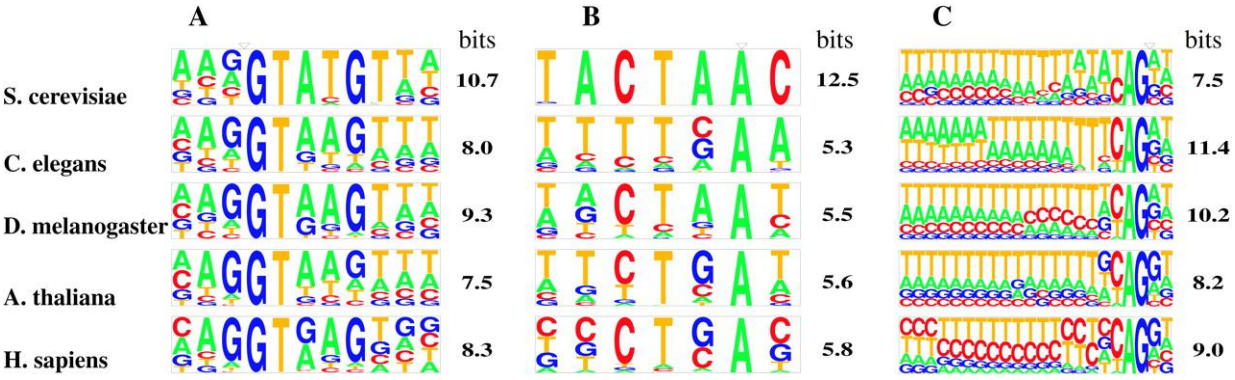


Figure 27 - *Séquences logos des sites d'épissage **A)** donneur et **C)** accepteur consensus de quatre organismes modèles et de l'Homme. **B)** Représentation des séquences logos du point de branchement (Source : (Lim and Burge, 2001)).*

4.3.1.4 Le point de branchement

Le point de branchement est un signal présent au niveau des introns entre 15 et 50 pb en amont du SE accepteur (Matera and Wang, 2014). La particularité de ce signal est la présence d'une adénine incorporée dans un motif très divergent semblable à : CU[A/G]**A**[C/U] (figure 27B). Cette adénine joue un rôle crucial lors de l'épissage, car elle est reconnue par une sous-unité du spliceosome induisant la formation du lasso (Green, 1986).

4.3.2 L'épissage des gènes codant pour des protéines

Lors de la transcription des gènes codant pour des protéines, la polymérase II génère un pré-ARNm. Ce dernier comprend l'ensemble des éléments présents entre le TSS et le terminateur. On retrouve les régions 5' et 3' UTR, ainsi que les exons et les introns (figure 28A). L'étape suivante consiste à produire un ARNm mature en excisant les introns et en liant les exons entre eux (Berget *et al.*, 1977). Cette étape s'appelle l'épissage (Jurica and Moore, 2003) et est réalisée par le spliceosome. L'ARNm mature produit, est alors constitué d'exons interconnectés (hors rétention d'intron) et des régions UTR. Il est également protégé par une coiffe à l'extrémité 5' et par une suite d'adénosines, appelée queue polyA à l'extrémité 3' (figure 28B).

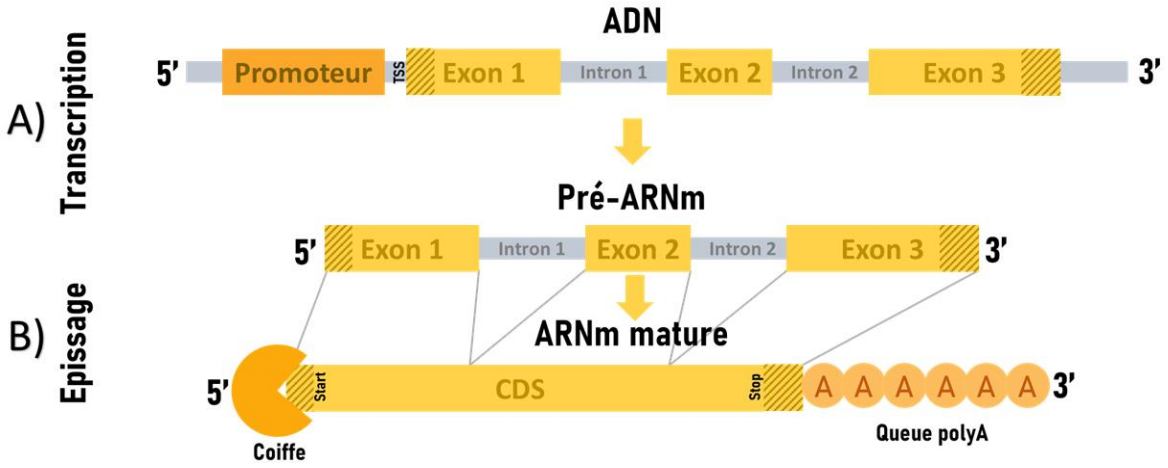


Figure 28 - *Processus de transcription et d'épissage.* ***A)*** *Mécanisme de transcription induisant la formation d'un pré-ARNm à partir d'un gène codant pour une protéine.* ***B)*** *Génération d'un ARNm mature via le processus d'épissage. Lors de ce processus, les introns sont excisés et les exons sont ligaturés entre eux. L'ARNm mature est ensuite protégé par une coiffe en 5' et une queue polyA en 3'.*

4.3.2.1 Le spliceosome

Le spliceosome réalise l'épissage des exons et l'excision des introns, en s'assemblant sur des segments du gène. Il est présent chez tous les organismes eucaryotes (Frey and Pucker, 2020) (environ 100 000/cellule humaine (Chen and Moore, 2015)) et sa structure est fortement conservée (Wahl *et al.*, 2009). On estime qu'il était présent au sein de LECA (*Last Eukaryotic Common Ancestor*) (Poverennaya and Roytberg, 2020). On retrouve deux types de spliceosome, le majeur qui excise les introns de U2 et le mineur qui excise les introns de type U12 (Patel and Steitz, 2003).

i) Le spliceosome majeur est un complexe ribonucléoprotéique composé de 5 petites ribonucléoprotéines nucléaires (snRNP) (U1, U2, U4, U5 et U6) constitués d'ARN non-codants (snRNA) ainsi qu'une trentaine de protéines et plus de 200 co-facteurs (Jurica and Moore, 2003) (figure 29). Il est appelé "majeur" car on le retrouve dans 99,5% des cas chez la plupart des organismes eucaryotes (Yoshida and Ogawa, 2014). Ce complexe est très dynamique et change de conformation au cours de son cycle d'assemblage. Cependant, le *core* du spliceosome reste stable et est composé de la sous-unité U6, la majeure partie de la sous-unité U5, une partie de la sous-unité U2 et une vingtaine de protéines (Shi, 2017). Le spliceosome contient également des hélicases *DexD/H*, qui ont pour rôle de médier les changements conformationnels lors du processus d'assemblage (Liu and Cheng, 2015).

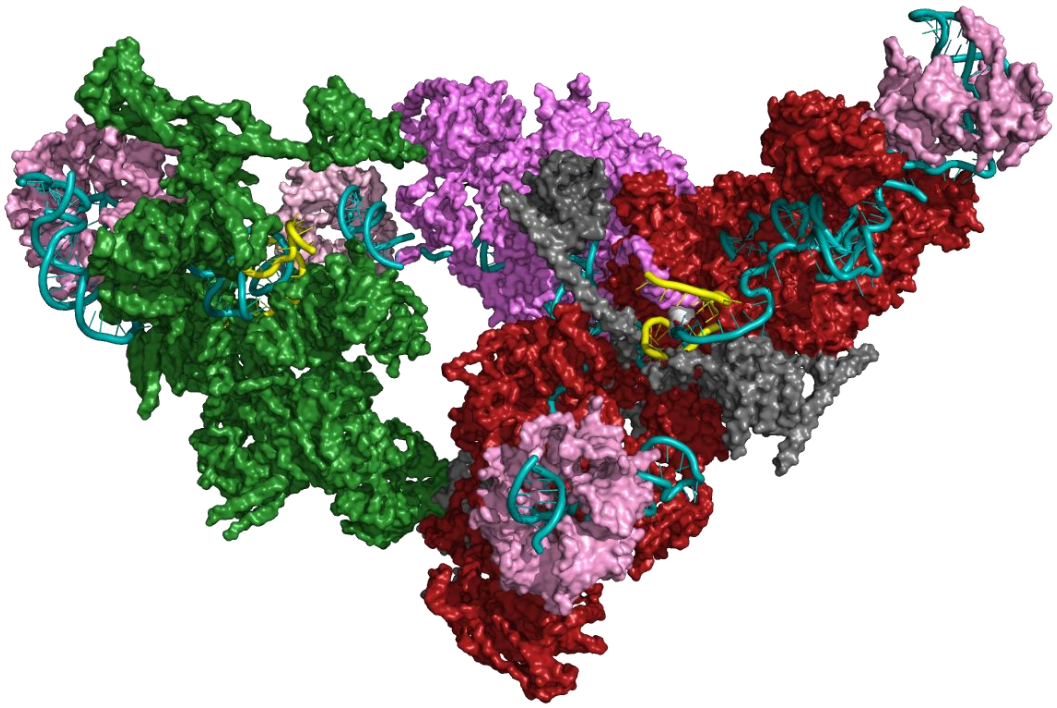


Figure 29 - *Structure tridimensionnelle d'un spliceosome humain (complexe B). La structure a été réalisée avec la technique de cryo-microscopie électronique à une résolution moyenne de 3.8 Å. Les segments introniques du pré-ARNm sont coloriés en jaune avec le site donneur (GT) représenté par la sphère blanche, les snRNA en turquoises, la sous-unité U2 en vert, U5 en rouge, les protéines impliquées dans le tri-snRNP en violet, les protéines SM en rose et d'autres protéines du complexe B en gris (Source :6AHD (Zhan et al., 2018), illustration : C. Mayer).*

ii) Le spliceosome mineur, aussi appelé U12, est présent chez la grande majorité des espèces eucaryotes (Turunen *et al.*, 2013), mais est très faiblement représenté (seulement 0,5% des cas (Yoshida and Ogawa, 2014)). Sa particularité est d'exciser des introns spécifiques avec des SE non-canoniques comme la paire AT-AC. Cependant, il a été démontré que certains introns avec des SE canoniques peuvent être reconnues par un spliceosome mineur (Poverennaya and Roytberg, 2020). Ce spliceosome est aussi composé de 5 snRNP (U11, U12, U5 et U4atac/U6atac), qui sont analogues à ceux du spliceosome majeur, à l'exception de la sous-unité U5 qui est identique (Patel and Steitz, 2003). Du fait de la divergence des séquences des SE non-canoniques, les introns de type U12 sont identifiés de manière moins efficace, induisant une régulation des gènes spécifique lorsque ce type d'introns est présent dans le gène.

4.3.2.2 Le mécanisme d'épissage

Le processus d'épissage des gènes codant pour des protéines se déroule au sein du noyau des cellules eucaryotes et est initié par le spliceosome. Des études ont décrit le mécanisme d'épissage comme un processus co-transcriptionnel principalement, ce qui signifie que le spliceosome agit alors que le transcrit naissant est encore accroché à la polymérase II (Bentley,

2014). Cependant, l'épissage peut également être post-transcriptionnel (Gazzoli *et al.*, 2015) notamment pour les introns alternatifs (Vargas *et al.*, 2011). Le rôle de l'épissage est de regrouper et lier les exons entre eux en supprimant les introns, induisant une régulation de l'expression des gènes. L'épissage est un mécanisme multi-étape complexe qui implique l'ensemble des sous-unités du spliceosome et de nombreux co-facteurs. La première étape (1) de l'épissage consiste à séparer le segment intronique (*branching*) et la deuxième étape (2) permet la ligation de deux exons entre eux (*exon ligation*) (figure 30) (Burge *et al.*, 1999). Ces étapes sont régies par une réaction chimique de transestérification (Will and Lührmann, 2011) (transformation d'un ester (R-COO-R') par un autre, via un groupement alcool (R-OH)) secondée par deux ions métalliques (magnésium) qui stabilisent le nucléophile ainsi que l'élément éliminé (Yean *et al.*, 2000; Fica *et al.*, 2013).

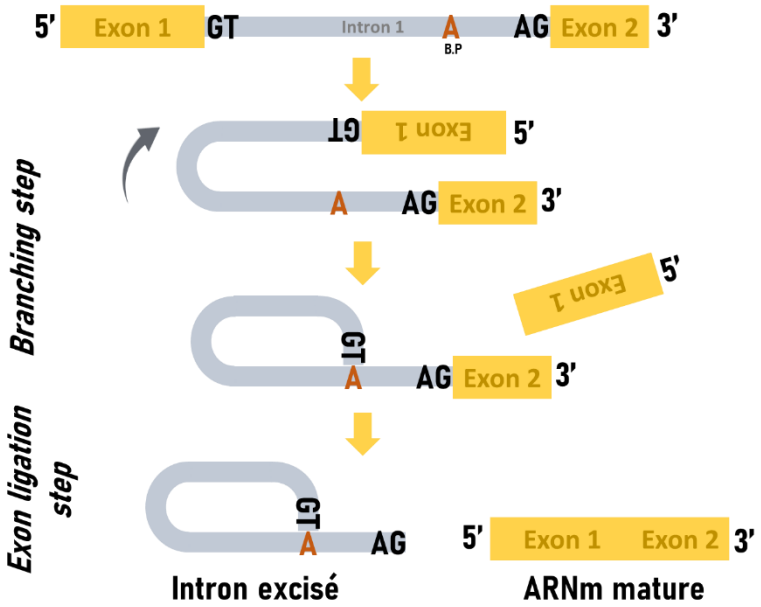


Figure 30 - *Représentation des deux étapes du processus d'épissage. Deux exons séparés par un intron vont être regroupés, par l'élimination de ce dernier. Dans un premier temps, des protéines vont induire le repliement du premier exon afin de rapprocher le site donneur (5') vers l'adénine (A) du point de branchement (B.P). L'extrémité 5' est ensuite clivée, libérant le premier exon. Le site donneur va ensuite s'associer à l'adénine du point de branchement afin de former une structure en forme de lasso. Cette première étape constitue le "branching". Dans un deuxième temps, l'intron va être clivé au niveau du site accepteur (3') libérant le deuxième exon. Les deux exons peuvent ensuite se lier, lors de l'étape dite "exon ligation", afin de former un segment constant qui permettra la formation de l'ARNm mature lorsque tous les introns sont excisés.*

(1) Deux exons successifs sont rapprochés par l'intermédiaire des sous-unités U1 et U2 du spliceosome. Le groupe 2' OH de l'adénine du point de branchement va ensuite réaliser une attaque nucléophile sur le SE donneur (5') de l'exon en amont. Une première

réaction de transestérification va avoir lieu, séparant l'exon de l'intron au niveau du site donneur. L'extrémité 5' de l'intron va alors se lier à l'adénine du point de branchement créant un lasso (*lariat*).

(2) : Le groupe 3′ OH de l'exon libre qui a été clivé va ensuite fusionner avec l'exon en aval via une deuxième réaction de transestérification sur le SE accepteur, ce qui va entièrement libérer le fragment intronique.

Finalement, les exons sont ligaturés entre eux et les introns, sous forme de lasso, sont éliminés. Cependant, avant de déclencher le mécanisme d'épissage, il est nécessaire d'identifier les SE. Ce processus est modulé par des signaux d'épissage primaires comme les SE ou le point de branchement agissant en *cis*, ainsi que d'autres éléments comme des inhibiteurs ou des activateurs agissant en *trans* (Matera and Wang, 2014). Par exemple, le SE donneur (5') est spécifiquement reconnu par la sous-unité U6 par l'intermédiaire d'appariement entre des nucléotides de l'intron et ceux de cette sous-unité (figure 31A). Le point de branchement est également reconnu par une sous-unité du spliceosome (la sous-unité U2), formant un duplex U2-point de branchement, via des appariements nucléotidiques (figure 31B) (Shi, 2017).

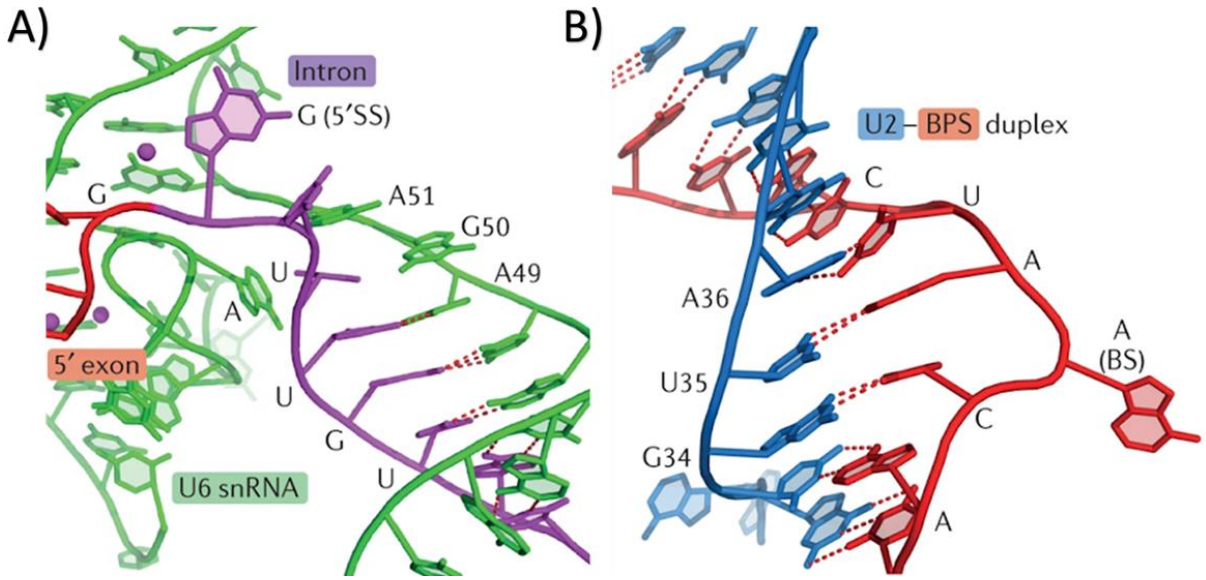


Figure 31 - *Identification des éléments propices au déclenchement du mécanisme d'épissage par des snRNA du spliceosome.* ***A)*** *Reconnaissance du SE donneur (5'), les nucléotides UGU introniques s'apparient avec les nucléotides de la sous-unité snRNA U6.* ***B)*** *Formation de duplex entre la séquence de point de branchement (BPS) et la sous-unité snRNA U2 chez la levure (source : (Shi, 2017)).*

En raison de sa complexité, le spliceosome intégral ne se fixe pas directement sur le pré-ARNm mais s'assemble dessus, tout au long du processus d'épissage. Ainsi, le processus d'assemblage du spliceosome et d'épissage sont intimement liés. Dans un premier temps, la sous-unité U1 reconnaît le site donneur (5') par l'intermédiaire d'une interaction ARN-ARN secondée par des protéines riches en acides aminés sérine et arginine (SR) (Cho *et al.*, 2011). Le complexe intermédiaire formé est le complexe E. Dans un second temps, la sous-unité U2 se fixe au niveau de point de branchement ainsi qu'à la sous-unité U1, formant le complexe A (Reed, 2000). Par ce mécanisme, les deux exons qui vont se lier se rapprochent. Lors de la troisième étape, la sous-unité U4 se lie à U6, formant un binôme qui va ensuite se lier à la sous-unité U5, induisant la formation d'un trinôme. Ce dernier va se rapprocher des sous-unités U1 et U2 formant le complexe B. Ce réarrangement conformationnel permet aux sous-unités U2 et U6 d'interagir et de catalyser l'étape de *branching*, libérant les sous-unités U1 et U4 permettant la formation du complexe C. Après des réarrangements au sein de ce complexe, la deuxième étape de catalyse (*exon ligation*) à lieu générant un complexe post-spliceosomal composé des exons liés et de l'intron en forme de lasso. Les sous-unités U2, U5 et U6 sont ensuite relarguées. L'assemblage du spliceosome est très dynamique et découle d'une succession d'associations entre de l'ARN et des protéines.

La figure 32 résume l'ensemble du processus de formation du spliceosome ainsi que le mécanisme d'épissage. Pour une lecture plus précise de ces mécanismes, voir (Staley and Woolford, 2009; Will and Lührmann, 2011; Matera and Wang, 2014; Shi, 2017; Wan *et al.*, 2019; Wilkinson *et al.*, 2020)

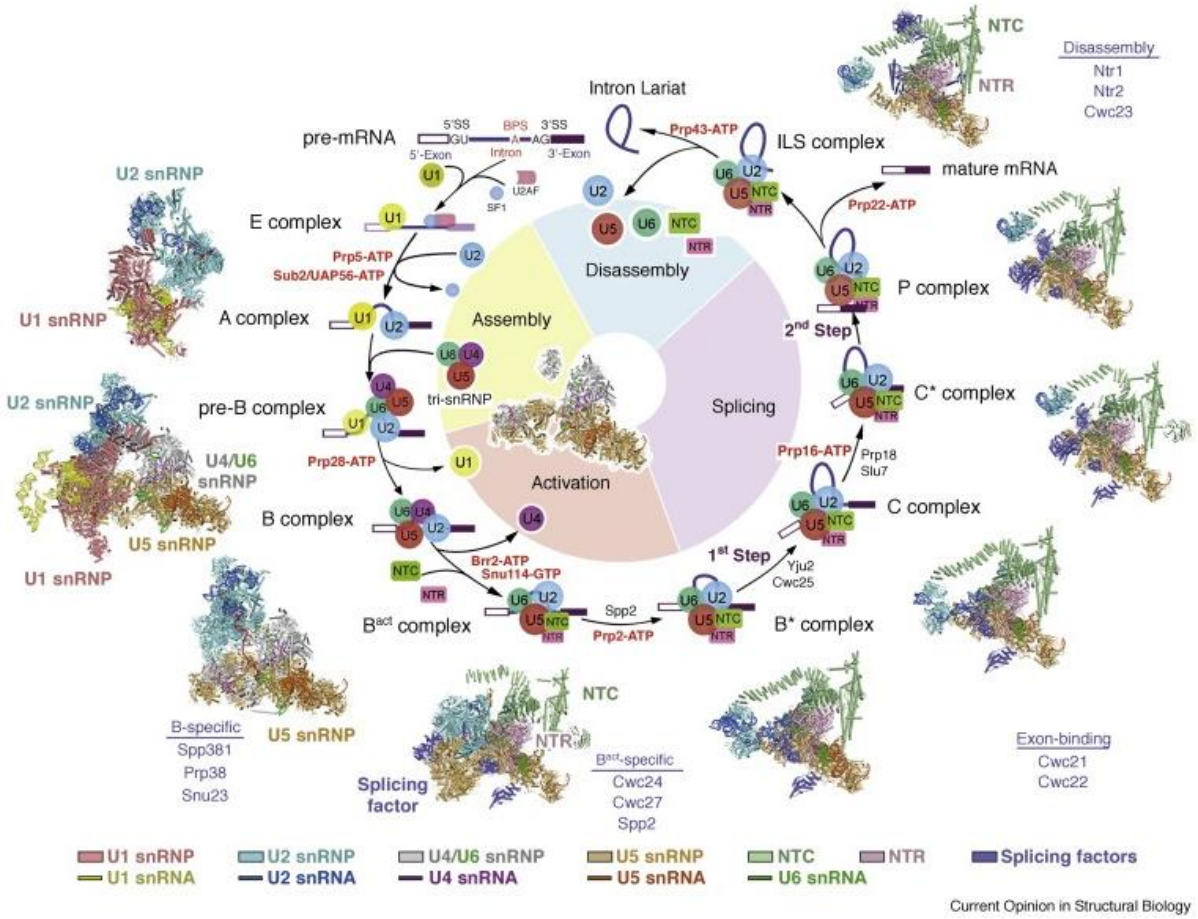


Figure 32 - *Représentation du cycle du mécanisme d'épissage. Différentes étapes sont schématisées : l'assemblage du spliceosome sur le pré-ARNm, l'activation, le processus d'épissage (branching et exon ligation step) et enfin le désassemblage des sous-unités du spliceosome (source :(Wan et al., 2019)).*

4.3.2.3 L'épissage alternatif

Comme son nom l'indique, l'épissage alternatif (Matlin *et al.*, 2005; Blencowe, 2006) participe à la formation d'ARNm matures alternatifs et par conséquent de protéines différentes (appelées isoformes) (Leff *et al.*, 1986). L'épissage alternatif est le mécanisme qui permet d'étendre le protéome (en multipliant par 5 le nombre de protéines chez l'Homme par exemple) par rapport au nombre de gènes codant pour des protéines. En effet, grâce à ce phénomène, un gène codant pour des protéines induit la production de plusieurs isoformes en fonction d'une combinaison exonique différente. On estime à 95% le nombre de gènes humains soumis au mécanisme d'épissage alternatif (Pan *et al.*, 2008). En moyenne, on dénombre entre 3 et 4 isoformes par gène humain codant pour des protéines (communication personnelle – Weber 2021) (Nurk *et al.*, 2021), bien qu'en général il n'y ait qu'une seule isoforme majoritaire (Ezkurdia *et al.*, 2015). L'épissage alternatif est également modulaire, la population d'isoformes produite varie en fonction du type de tissu, ainsi que durant le développement de l'organisme (Stamm *et al.*, 2000) ou encore en fonction de la cinétique de la polymérase II

(Bentley, 2014) ou de la structure secondaire du pré-ARNm (Warf and Berglund, 2010). Les différents signaux, facteurs et conditions induisent un épissage alternatif hautement spécifique. Ces nombreuses caractéristiques ont engendré la création d'un code d'épissage (Barash *et al.*, 2010; Baralle and Baralle, 2018) soulignant les éléments de régulation de l'épissage alternatif dans un environnement spécifique. Cette diversité protéomique a ainsi de grandes conséquences sur le phénotype de l'individu.

i) Les sites d'épissage consensus, divergents et cryptiques

Parmi les SE, on dénombre trois principales catégories : les SE consensus (avec un dinucléotide GT pour le SE donneur et un dinucléotide AG pour le SE accepteur), les SE non-consensus qui n'ont ni le dinucléotide GT ni AG et qui s'éloignent plus ou moins de la séquence des SE consensus, et enfin les SE cryptiques.

Les SE consensus (ou canoniques) sont présents dans 99% des cas et sont reconnus par le spliceosome dit "majeur" ou U2 (Matera and Wang, 2014). En général, ce sont des sites dits "forts", car ils sont facilement identifiés par le spliceosome en raison de leurs séquences consensus et des éléments régulateurs proximaux. Les SE non-consensus (ou non-canoniques) divergent de la séquence des SE consensus, notamment au niveau des dinucléotides GT et AG. On retrouve par exemple les couples GC-AG (1%), AT-AC (0,15%) et même GA-AG (1-3 cas connus) (Sheth *et al.*, 2006; Pucker and Brockington, 2018). Ces sites sont généralement dits "faibles", car leur affinité avec les éléments du spliceosome est atténuée du fait de leurs motifs divergents et probablement d'autres éléments régulateurs plus distants. La reconnaissance entre les sites canoniques et non-canoniques dépend d'une compétition entre les différents signaux et éléments régulateurs présents au niveau cellulaire, voire tissulaire. Il existe bien évidemment des SE canoniques faibles et la réciproque est également valable. Enfin, certains SE sont dits cryptiques. Ces derniers ont un motif généralement proche des SE consensus, mais sont très faiblement identifiés par le spliceosome voir pas du tout. Ils sont dits "dormants" et peuvent être reconnus de manière systématique lorsqu'ils sont modifiés par une variation (Padgett *et al.*, 1986; Green, 1986) induisant généralement un phénotype pathologique (Singh and Cooper, 2012; de Boer *et al.*, 2019; Montes *et al.*, 2019).

ii) Les éléments régulateurs

De nombreux autres signaux sont présents sur les séquences introniques et exoniques afin de compenser la divergence des principaux éléments régulateurs (SE et point de branchement) et aider au choix de l'exon à épisser. La présence de signaux supplémentaires accentue (ou diminue) la reconnaissance et l'assemblage du spliceosome par l'intermédiaire de

protéines (Smith and Valcárcel, 2000). On retrouve deux familles de protéines, les protéines sérine-arginine (protéines SR) qui sont souvent activatrices et les complexes nucléaires hétérogènes (hnRNP), souvent inhibiteurs (Stamm *et al.*, 2000; Matlin *et al.*, 2005). Ces sites de fixations sont présents au niveau des exons (Wang and Burge, 2008) : lorsqu'ils renforcent la reconnaissance, on parle de *Exonic Splicing Enhancer* (ESE) (Cartegni *et al.*, 2003) et s'il la réduit on parle de *Exonic Splicing Silencer* (ESS). Les introns ont également des signaux similaires (Wang and Burge, 2008) : les *Intronic Splicing Enhancer* (ISE) et les *Intronic Splicing Silencer* (ISS) participent à la régulation de l'épissage alternatif (figure 33). Ces motifs sont généralement petits et variables, mais riches en purines ce qui favorise la fixation des protéines SR (Kelemen *et al.*, 2013).

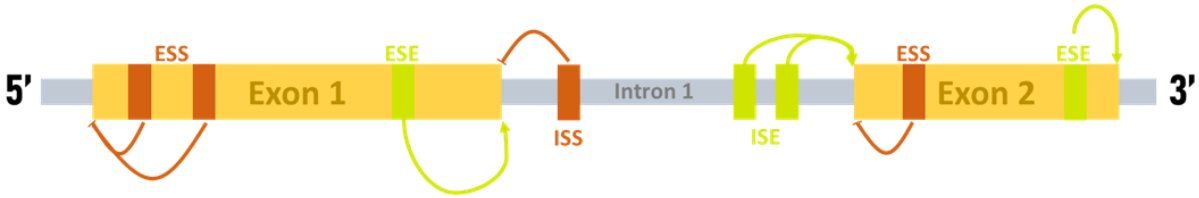


Figure 33 - *Représentation de la répartition des éléments régulateurs introniques et exoniques au sein d'un gène codant pour des protéines. Les éléments ESE et ISE intensifient la reconnaissance du SE, alors que les éléments ESS et ISS réduisent la reconnaissance.*

iii) La structure secondaire des pré-ARNm

Il a été démontré que la structure secondaire des pré-ARNm influence le mécanisme d'épissage (Warf and Berglund, 2010). En effet, certaines structures peuvent stimuler ou inhiber l'épissage ce qui peut affecter le SE localement et induire un phénotype pathologique (Soemedi *et al.*, 2017). Par exemple, la structure tige-boucle à proximité du site donneur du dixième exon du gène *Tau* (MAPT) impacte directement l'utilisation du site donneur (Lisowiec *et al.*, 2015).

L'identification d'exons alternatifs est un mécanisme complexe dépendant de plusieurs facteurs. Certains exons sont systématiquement présents dans l'ensemble des isoformes, ils sont dits "constitutifs", et ceux qui sont au moins absents une fois sont dits "alternatifs". Plusieurs stratégies ont été mises en place pour réaliser l'épissage alternatif (figure 34) :

- Exon cassette : l'exon est entièrement ignoré et supprimé, à l'instar d'une séquence intronique
- Exons mutuellement exclusifs : la présence d'un exon exclut obligatoirement un autre exon. La présence des deux n'est pas possible

- Extrémité 5` alternative : une partie de l'exon est rognée ou une partie de l'intron est ajoutée en 5' en raison d'un SE alternatif
- Extrémité 3` alternative : une partie de l'exon est rognée ou une partie de l'intron est ajoutée en 3' en raison d'un SE alternatif
- Rétention d'intron : un segment intronique n'est pas épissé et participe à la formation de l'ARNm mature
- Promoteur alternatif : la présence d'un autre promoteur dans le gène peut induire un épissage alternatif

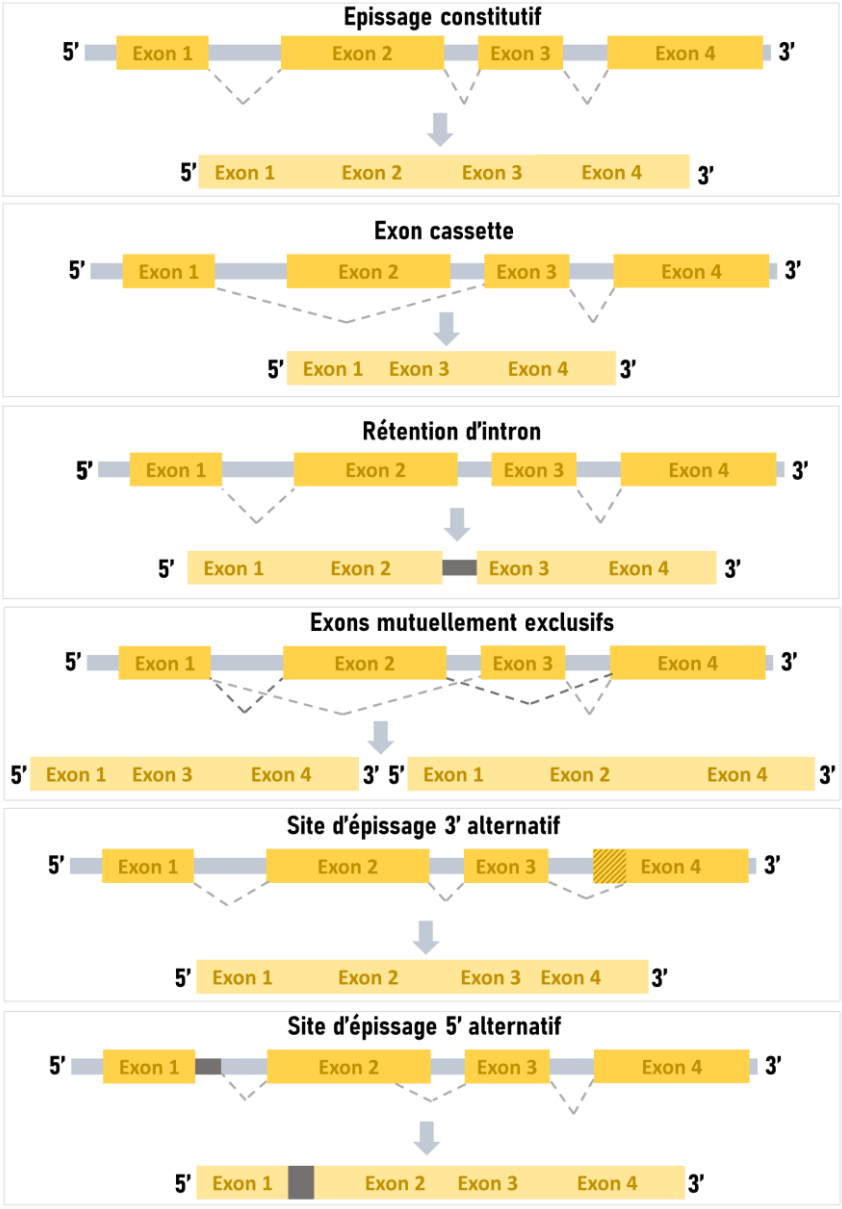


Figure 34 - *Les différentes stratégies mises en place pour effectuer l'épissage alternatif. Chaque combinaison permet la formation d'une isoforme différente.*

4.3.2.4 Autres mécanismes d'épissage

Un autre mécanisme d'épissage existe, principalement au sein des organelles eucaryotes, ainsi que chez les procaryotes (Robart and Zimmerly, 2005), il s'agit d'introns du groupe II capable de s'auto-épisser (Pyle, 2016). On parle aussi de ribozymes *i.e.* des ARN capables de catalyser une réaction chimique. Le processus d'épissage des introns du groupe II est très proche de celui initier par le spliceosome (Brown *et al.*, 2014). Il y a l'étape de *branching* ainsi que l'étape de ligation des exons, secondés par deux ions magnésium. Ainsi, l'ARN et les ions sont les éléments principaux permettant l'autocatalyse. Cependant, dans certains cas, des protéines comme les maturases viennent assister la réaction en optimisant la structure secondaire des introns du groupe II (Zimmerly and Semper, 2015). Il a également été observé que la structure des introns du groupe II est complexe et bien conservée tandis que sa séquence primaire l'est moins (Pyle, 2016).

Un dernier mécanisme d'épissage connu est celui guidé par des nucléases. Il concerne principalement les exons des ARNt (Abelson *et al.*, 1998; Schmidt and Matera, 2020).

4.3.2.5 Dysfonctionnement de l'épissage: causes et conséquences

Comme tout mécanisme biologique, l'épissage peut être sujet à un dysfonctionnement (Cáceres and Kornblihtt, 2002). Malheureusement, lorsque le mécanisme d'épissage est dérégulé, cela peut avoir de lourdes conséquences induisant généralement un phénotype pathologique (Faustino and Cooper, 2003; Singh and Cooper, 2012; Padgett, 2012). Différentes perturbations peuvent affecter le mécanisme d'épissage: elles sont soit *cis*, soit *trans*.

i) Les perturbations *cis* affectent le mécanisme d'épissage au travers de variations dans les séquences des SE ou des éléments régulateurs. Les variations (souvent silencieuses, *i.e.* elles n'affectent pas la CDS (Cooper and Mattox, 1997)) peuvent modifier le SE directement (Cartegni *et al.*, 2002) et réduire leur reconnaissance par le spliceosome induisant une isoforme différente et souvent pathologique. A l'inverse, une variation peut activer un SE cryptique et induire une autre isoforme aberrante pathologique (Nissim-Rafinia and Kerem, 2002).

ii) Les perturbations *trans* affectent les éléments du spliceosome ou les facteurs liés à l'épissage. Les variations peuvent perturber la génération et/ou l'expression des éléments constituant le spliceosome ou bien des co-facteurs jouant un rôle dans l'épissage (Tazi *et al.*, 2009).

De nombreuses maladies génétiques (López-Bigas *et al.*, 2005) sont les conséquences d'une perturbation du processus d'épissage. La maladie de Menkes est une maladie neurodégénérative progressive liée à un dysfonctionnement du métabolisme du cuivre (Tümer

and Møller, 2010). Il a été démontré que la variation du SE donneur du 6^{e} exon du gène *ATP7A* induit la maladie (Møller *et al.*, 2000). Cette perturbation *cis* est un des nombreux exemples retrouvés dans la littérature. Les perturbations *trans* sont également bien représentées, par exemple l'inhibition du facteur d'épissage TDP43, à la suite d'une variation ou d'un dérèglement de l'expression, entraîne des pathologies lourdes comme la sclérose latérale amyotrophique (Neumann *et al.*, 2006; Tazi *et al.*, 2009). D'autres maladies peuvent être liées à un dysfonctionnement du processus d'épissage. On retrouve par exemple la dysautonomie (variation dans le SE donneur), le syndrome de Hutchinson-Gilford (variation intronique activant un site d'épissage cryptique), la dystrophie musculaire de Duchenne, l'amyotrophie spinale ou les tauopathies comme la maladie d'Alzheimer (variations dans les séquences régulatrices) (Tazi *et al.*, 2009).

Plusieurs exemples de dysfonctionnement de l'épissage ont également été démontrés dans les cas de cancers (David and Manley, 2010; Auboeuf *et al.*, 2012). Une variation affectant le gène *U2AF1* est liée au développement de leucémie (Yoshida and Ogawa, 2014; Waterfall *et al.*, 2014). Cependant, la présence d'isoformes aberrantes induisant une pathologie peut être exploitée à des fins de diagnostic et pour la mise en place d'une thérapie contre certaines maladies comme le cancer.

4.4 L'annotation du génome eucaryote

L'ensemble des éléments cités précédemment jouent un rôle clé dans de nombreux mécanismes biologiques. Il est donc important de les localiser sur le génome, d'identifier leurs architectures et de caractériser leurs fonctions (Mudge and Harrow, 2016). En bio-informatique, cette étape s'intitule l'annotation des génomes et est un champ de recherche à part entière (Yandell and Ence, 2012; Ejigu and Jung, 2020). Malheureusement, le rythme de l'annotation des génomes n'est pas le même que celui du séquençage des génomes. En effet, l'annotation est un processus long et fastidieux (Danchin *et al.*, 2018), dépendant de nombreux facteurs comme une puissance de calcul suffisante, des outils adaptés ou des données de qualité. L'annotation automatique des génomes eucaryotes est donc un cas d'application idéale pour l'IA. En effet, il s'agit d'un problème complexe qui n'a pas encore été entièrement résolu (Salzberg, 2019) et il symbolise parfaitement les 3V du *Big Data*. De plus, en raison de son automatisation, l'annotation est temporaire et la ré-annotation des génomes est un processus sans fin (Siezen and Hijum, 2010). Elle est également sujette à de nombreuses erreurs (sur

lesquelles nous reviendrons tout au long de ces travaux de thèse) et les services d'annotation peuvent générer des résultats différents (Bakke *et al.*, 2009). Ainsi, en couplant l'énorme quantité de données produites (pour entraîner des modèles), les nouveaux algorithmes performants de l'IA et la puissance de calcul, il sera possible de faire d'énormes progrès pour l'annotation des génomes.

De nombreux efforts ont été réalisés récemment pour mettre en place des ressources et développer des outils facilitant l'annotation des génomes. Par exemple, on retrouve l'encyclopédie ENCODE (Dunham *et al.*, 2012) dont nous avons déjà parlé précédemment, le projet GENCODE qui est une ressource publique basée sur ENCODE, répertoriant des annotations de qualité chez l'humain et la souris pour la recherche biomédicale (Frankish *et al.*, 2019). D'autres ressources sont également largement exploitées comme Ensembl (Howe *et al.*, 2020), l'UCSC (Kent *et al.*, 2002), Gnomon du NCBI (Souvorov *et al.*, 2010), RefSeq (Pruitt *et al.*, 2007), Uniprot (The UniProt Consortium, 2017), WormBase (Harris *et al.*, 2020), FlyBase (Larkin *et al.*, 2020) et bien d'autres encore. A l'instar du concours CASP pour la prédiction de la structure des protéines, un concours a été organisé, le EGASP, avec pour objectif d'évaluer l'état de l'art de l'annotation du génome (Guigó *et al.*, 2006).

L'annotation d'un génome eucaryote est principalement une étape de prédiction et consiste à localiser les éléments fonctionnels que nous avons cité précédemment, comme les gènes codant pour des protéines ou pour des ARN non-codants, les éléments régulateurs (promoteur, TSS, terminateur, point de branchement, ESE/ESS/ISE/ISS, etc.), les SE (donneur et accepteur), les pseudogènes, les éléments répétés, les transposons ou encore les sites de méthylation de l'ADN, les îlots CpG, la configuration de la chromatine et les variants. De plus, il est important d'identifier le cadre de lecture ouverte (ORF), la CDS ainsi que l'ensemble des combinaisons exoniques biologiquement viables (*i.e.* réalisé par le spliceosome) amenant à une ou plusieurs isoformes non aberrantes. Enfin, il est indispensable d'associer une fonction à tous les éléments identifiés. On retrouve donc deux branches principales dans le processus d'annotation : l'annotation structurale et l'annotation fonctionnelle. Dans le cadre de ma thèse, je me suis consacré à l'annotation structurale des gènes codant pour des protéines, par conséquent j'expose uniquement cette étape.

4.4.1 L'annotation structurale

Cette étape consiste à localiser les éléments fonctionnels et importants du génome, ainsi qu'à identifier leur structure interne. L'effort majeur est concentré sur la localisation des gènes codant pour des protéines ainsi que l'identification de leur architecture organisée en mosaïque d'introns et d'exons. Ces éléments sont les plus représentatifs du phénotype et jouent un rôle clé dans la compréhension des mécanismes biologiques. Par exemple, le séquençage et l'annotation de l'écrevisse *Astacus astacus* réalisés au laboratoire, permettra de comprendre des mécanismes impliqués dans leur fragilité face aux pathogènes *Aphanomyces astaci*, induisant la peste de l'écrevisse (Boštjančić *et al.*, 2021).

L'annotation structurale se décompose en deux principales phases, la phase de calcul et la phase d'annotation à proprement parler (Yandell and Ence, 2012). Ces étapes regroupent généralement plusieurs outils indépendants. En effet de nombreux pipelines d'annotation ont été développés, englobant une suite d'outils et de méthodes qui s'enchaînent. Parmi ces pipelines, on peut citer ceux issus de grands consortia comme Ensembl, Gnomon ou l'UCSC et d'autres comme Braker (Hoff *et al.*, 2016, 2019) et Braker2 (Brůna *et al.*, 2020), Maker (Cantarel *et al.*, 2008; Holt and Yandell, 2011) ou PASA (Haas *et al.*, 2003).

4.4.2 La phase de calcul

La phase de calcul est l'étape de prédiction des éléments du génome par des approches d'homologie et *ab initio* ou à l'aide de données expérimentales. Cependant avant de traiter les données disponibles, une étape cruciale est nécessaire. Elle consiste à masquer les éléments répétés comme les SINE ou LINE, et les régions de faible complexité (*i.e.* les séquences avec peu de variation nucléotidique comme "AGAGAGAGA"). Classiquement, cette étape consiste à remplacer ces régions par des caractères 'N' (masquage dur), ou par des caractères minuscules dans certains cas (masquage doux) (Yandell and Ence, 2012). Ces régions complexifient la détection des éléments importants ce qui peut provoquer des erreurs d'annotations si elles ne sont pas masquées. RepeatMasker (repeatmasker.org) est probablement l'outil le plus utilisé pour réaliser cette tâche, bien que de nombreux autres outils sont disponibles, comme Tandem repeats finder (Benson, 1999), Mreps (Kolpakov *et al.*, 2003) ou Piler (Edgar and Myers, 2005).

i) Les approches basées sur des preuves expérimentales

Les données expérimentales sont une source importante d'information. Parmi ces données hétérogènes, aussi appelées preuves, on retrouve les EST (*expressed sequence tag*) qui

sont de petits fragments d'ADN complémentaires permettant d'identifier des transcrits (Adams *et al.*, 1991), les données de RNA-seq (sous la forme d'ADN complémentaire (ADNc)), ainsi que les séquences d'acides aminés des protéines. Les données de RNA-seq sont devenues les preuves les plus prépondérantes, car elles permettent d'améliorer l'annotation notamment au niveau de la délimitation des SE (Mudge and Harrow, 2016) et des régions UTR (Keilwagen *et al.*, 2016) (figure 35A), cependant la disponibilité des données et leur extraction peuvent restreindre cette approche, notamment pour des transcrits faiblement exprimés. Les preuves sont ensuite alignées sur le génome à annoter afin d'identifier les fragments exoniques et ainsi pouvoir construire une structure de gène (figure 35B).

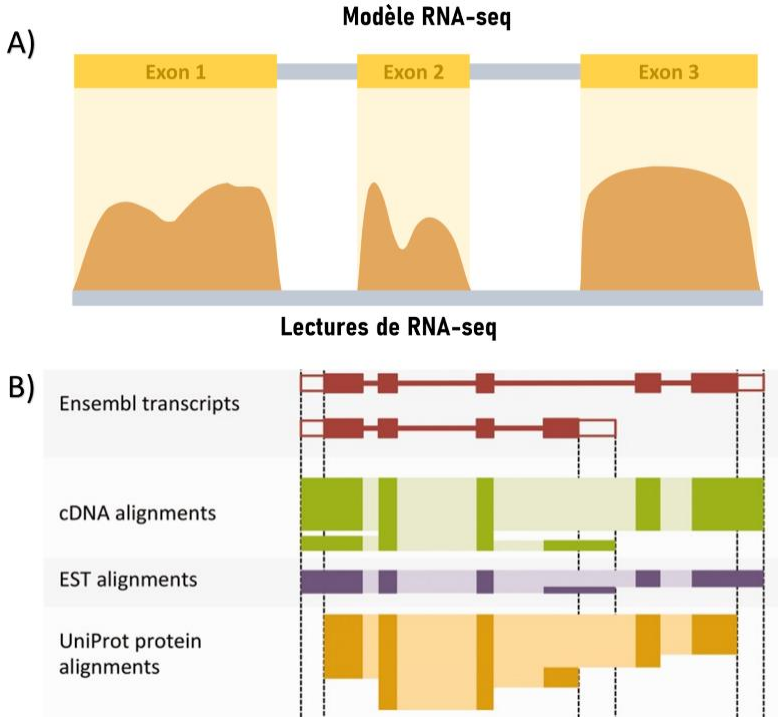


Figure 35 - *Phase de calcul de l'annotation structurale à l'aide de preuves expérimentales.* ***A)*** *Alignement des lectures de RNA-seq pour la construction d'un modèle de structure de gène.* ***B)****Différents types de preuves sont utilisés afin de construire une structure de gène ou de transcrit. Un gène (rouge) est prédit d'après la combinaison des preuves apportées par les différents types de données : RNA-seq (vert), EST (violet) ou protéines (orange). Les blocs plus foncés correspondent aux exons. Les blocs blancs aux extrémités correspondent aux régions UTR. (source : (Aken et al., 2016)).*

ii) Les approches par homologie

Afin de prédire une structure de gène, on peut utiliser les approches par homologie, appelées approches extrinsèques. Cela consiste à extrapoler la structure de gènes ou de transcrits de l'organisme cible à l'aide des gènes d'organismes phylogénétiquement proches dont le génome a déjà été annoté. Cette méthode se base sur l'hypothèse que les séquences conservées

entre des organismes proches sont relativement similaires. Cette stratégie inclut notamment les gènes orthologues quand ces derniers sont disponibles. Les données d'orthologie peuvent notamment être extraites de bases de données d'orthologie comme OrthoInspector (Nevers *et al.*, 2019) ou OrthoDB (Kriventseva *et al.*, 2015). On peut également identifier et aligner les différentes régions homologues sur un génome de référence proche à l'aide du programme BLAST (Altschul *et al.*, 1990) et ces déclinaisons. De plus, il existe de nombreux autres programmes ayant chacun leurs particularités en termes de performance, de vitesse de calcul, etc. Parmi les plus populaires, on retrouve : Twinscan (Korf *et al.*, 2001), GeneWise et GenomeWise (Birney, 2004), GenomeThreader (Gremme *et al.*, 2005), Exonerate (Slater and Birney, 2005) et ProSplign (Kiryutin *et al.*, 2021).

iii) Les approches ab initio

Les méthodes *ab initio*, ou intrinsèques, sont des méthodes de prédiction utilisant des modèles probabilistes (Stanke *et al.*, 2006), des algorithmes d'évolution artificielle (Chowdhury *et al.*, 2017), de *machine learning* (Yip *et al.*, 2013) et plus récemment de *deep learning* (Jaganathan *et al.*, 2019). Dans le contexte de la prédiction de gènes codant pour des protéines, deux stratégies sont possibles. La première consiste à prédire la structure interne globale du gène et la deuxième consiste à prédire uniquement un élément de cette structure. Ces méthodes s'appuient sur les informations intrinsèques de la séquence génomique (Yandell and Ence, 2012). Nous avons vu que de nombreux signaux étaient encodés au sein de ces séquences génomiques (promoteur, SE, éléments régulateurs, etc). Il existe aussi d'autres informations qui sont plus implicites, comme la taille des exons ou des introns, la composition en nucléotide (comme le pourcentage en GC) pour caractériser le segment (Kalari *et al.*, 2006), l'utilisation des codons (*i.e.* respect du cadre de lecture). Ces deux types d'informations participent à la conception des approches *ab initio* qui sont ainsi basées sur un couplage de méthodes de capteurs de signaux et de méthodes de capteurs de contenus (Huang *et al.*, 2016).

Les méthodes de prédictions *ab initio* sont encore indispensables, notamment quand les données de l'organisme cible ou d'organismes phylogénétiquement proches sont absentes ou de mauvaise qualité. De plus, elles sont relativement rapides et leurs performances tendent à s'accroître actuellement. Ainsi, de nombreux outils ont été développés à l'aube de l'annotation comme GeneID (Guigó *et al.*, 1992) ou Genscan (Burge and Karlin, 1997), puis plus récemment, à la suite du séquençage du génome humain, comme Augustus (Stanke and Waack, 2003), GlimmerHMM (Majoros *et al.*, 2004), GeneMark-ES (Lomsadze, 2005), ou Snap (Korf, 2004), permettant d'obtenir une annotation de la structure complète des gènes codant pour des

protéines. Malheureusement, 20 ans après, nous utilisons encore les mêmes technologies (Salzberg, 2019), *e.g.* Braker2, un des pipelines les plus performants (Mudge and Harrow, 2016), se base sur Augustus. De ce fait, la tendance actuelle est le développement de programmes exploitant de nouvelles technologies comme le *deep learning*, qui cherchent à prédire un élément spécifiquement, comme le promoteur (Singh *et al.*, 2015), le TSS (Solovyev *et al.*, 2010), les gènes d'ARN de transfert (Lowe and Eddy, 1997; Gao *et al.*, 2018) ou les SE (Wang *et al.*, 2019).

4.4.3 La phase d'annotation

Une fois les différents éléments identifiés, il est nécessaire de faire une synthèse de l'ensemble des résultats obtenues par les différentes approches afin de créer un modèle de gène consensus (figure 36) et d'annoter le génome.

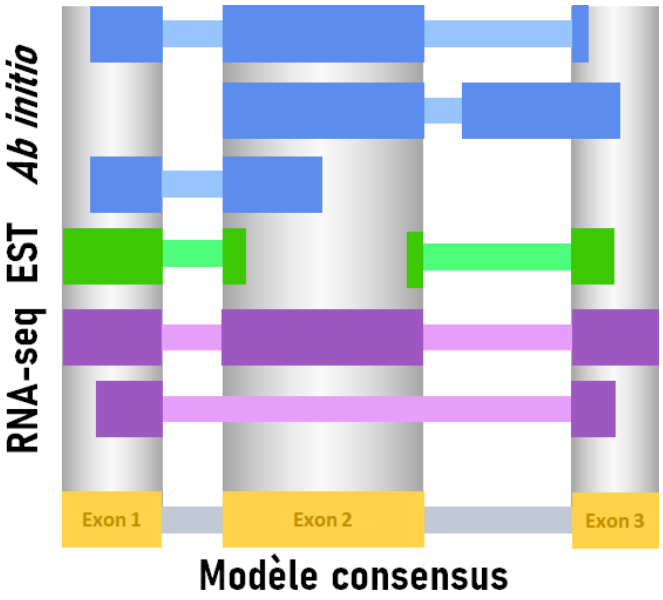


Figure 36 - *Étape finale de l'annotation, où l'ensemble des preuves sont regroupées et un modèle de gène consensus est généré. Dans certains cas, les données RNA-seq ou EST ne sont pas disponibles, donc seules les preuves ab initio sont prises en compte*.

Cette phase combine les résultats des différentes méthodes, tirant parti des avantages et des inconvénients de chacune (Yandell and Ence, 2012). Cependant, dans certains cas, seul un programme a été utilisé et le résultat de ce dernier participe à la totalité du modèle final. Dans le cas d'une combinaison des résultats, deux stratégies sont généralement adoptées par ce type d'approche. Premièrement, le programme essaye de construire le modèle le plus pertinent en

fonction des données expérimentales et du résultat obtenu par le prédicteur *ab initio/*homologie. Deuxièmement, une recherche *ab initio/*homologie est effectuée et le résultat peut être modulé par les données expérimentales. Par exemple, l'intégration de données RNA-seq avec des prédictions de gènes basées sur l'homologie (Keilwagen *et al.*, 2018). Parmi les programmes qui combinent les approches, on retrouve Augustus, qui a été mis à jour et prend dorénavant en compte des données de RNA-seq, ainsi que Jigsaw (Allen and Salzberg, 2005), Glean (Elsik *et al.*, 2007) ou EvidenceModeler (Haas *et al.*, 2008). En général, l'ensemble du processus d'annotation est réalisé par les pipelines d'annotation. De plus, la majorité des programmes de prédiction utilisent des formats standards. Ainsi, on retrouve trois formats majeurs : *Generic Feature Format* (GFF/GFF3), *General Transfer Format* (GTF) et *Browser Extensible Data* (BED).

4.5 Projet de thèse

C'est dans ce contexte que s'inscrivent mes travaux de thèse. Durant ces trois années, je me suis attelé à l'élaboration d'une nouvelle stratégie d'annotation des gènes codant pour des protéines, en me focalisant sur certains défis soulignés lors de mon exposé. Un des points principaux sur lequel j'ai porté une grande attention est la qualité des données. Ainsi la stratégie a été d'identifier les erreurs dans les données, les caractériser puis les supprimer afin de travailler avec des données de haute qualité et montrer leur importance. Le deuxième point a été d'exploiter ces données pour étudier leur pertinence lors de leur utilisation avec deux approches : *deep learning* et évolution artificielle, pour la prédiction des SE sur un large panel d'organismes eucaryotes dans le but d'améliorer l'annotation des gènes codant pour des protéines. Le premier outil développé se base sur un réseau de neurones convolutif, un des algorithmes d'intelligence artificielle les plus performants du moment. De plus, un prototype basé sur la programmation génétique a été développé afin de le combiner avec l'algorithme de *deep learning*, dans le but d'augmenter la précision des prédictions des SE.

En définitive, mon objectif était de simuler le spliceosome afin d'obtenir de hautes performances lors de la prédiction des SE chez les organismes eucaryotes, ainsi que d'identifier et comprendre des mécanismes sous-jacents de l'épissage.

MATÉRIELS ET MÉTHODES

Les technologies de séquençage à haut débit ont été le détonateur du *Big Data* en biologie, positionnant le génome à l'épicentre des travaux subséquents. Ces avancées sont notamment essentielles pour l'exploration de nouveaux génomes et leur annotation structurale et fonctionnelle. Ainsi, l'annotation des génomes eucaryotes et plus particulièrement des gènes codant pour des protéines est une des tâches essentielles en bio-informatique.

Dans ce contexte, je me suis intéressé pendant mes trois ans de thèse à améliorer l'annotation des génomes eucaryotes en me spécialisant dans l'identification des sites d'épissage (SE). De plus, l'identification, la caractérisation et la correction des erreurs induites par les programmes de prédiction ont été le credo de mes travaux.

Je présenterai donc dans un premier temps les ressources bio-informatiques utilisées pour extraire et traiter les données. Puis les programmes, outils et librairies qui ont participé à l'élaboration de mon projet. Enfin, je décrirai plus en détail les algorithmes d'IA que j'ai utilisés ainsi que les méthodes d'analyses.

1 - Ressources bio-informatiques

Durant mes travaux de thèse, j'ai eu l'occasion de parcourir un certain nombre de ressources afin d'exploiter celles qui me semblaient les plus pertinentes.

1.1 Banques de données de référence

1.1.1 UniProt

UniProt (The UniProt Consortium, 2017) est une base de connaissances regroupant des informations sur les protéines. Elle inclut par exemple les séquences des protéines et de certaines de leurs isoformes, les annotations structurales et fonctionnelles ainsi que de multiples références croisées. UniProt est composé d'un ensemble de plusieurs bases de données :

- *UniProt KnowledgeBase* (UniProtKB) : il s'agit de la base de données principale. Elle est constituée de la base TrEMBL et Swiss-Prot. TrEMBL répertorie les données protéiques annotées automatiquement et non révisées par des experts. Swiss-Prot intègre des données de haute qualité annotées manuellement par des experts. Malheureusement, les séquences incluses dans Swiss-Prot ne représentent qu'environ 0,3% de UniProtKB.

- *UniProt Reference Clusters* (UniRef) : regroupe des protéines ayant 100%, 90% ou 50% d'identité de séquence en commun.
- *UniProt Archive* (UniParc) : contient la plupart des séquences de protéines accessibles au public dans le monde. Un effort est réalisé quotidiennement afin de ne pas intégrer de données redondantes.

UniProt inclut également un nombre important de protéomes (plus de 300 000) complets ou non, issus de multiples organismes eucaryotes et procaryotes ainsi que des archées et des virus. UniProt est le résultat d'une collaboration entre l'*European Bioinformatics Institute* (EBI), le *Swiss Institute of Bioinformatics* (SIB) et le *Protein Information Resource* (PIR). Les données d'UniProt sont accessibles via le site web (uniprot.org/) ainsi qu'au travers d'une *Application Programming Interface* (API) (ebi.ac.uk/proteins/api/doc/). L'ensemble des données protéiques exploitées durant mes travaux de thèse ont été extraites de la base de connaissances UniProt.

1.1.2 Le projet Ensembl

Le projet Ensembl est une ressource web largement exploitée par la communauté scientifique qui se décline en deux points : Ensembl et Ensembl Genomes.

Ensembl (ensembl.org) (Howe *et al.*, 2020) annote les génomes et diffuse les données génomiques des espèces vertébrées. Il permet également la manipulation, la visualisation et l'analyse de données. Il propose de nombreuses informations sur les données omiques comme la séquence des gènes, leur structure détaillée et les différents transcrits. De plus, il permet l'analyse de l'effet des variants avec l'outil *Variant Effect Predictor* (VEP) (McLaren *et al.*, 2016). Les données sont accessibles *via* l'interface web ainsi qu'une API (rest.ensembl.org/) et un site FTP. Le pipeline d'annotation automatique d'Ensembl (Aken *et al.*, 2016) est basé sur de nombreux outils dont certains ont déjà été mentionnés comme Genscan, tRNAscan-SE, BLAST ou RepeatMasker. Ensembl est régulièrement mis à jour (quatre fois par an) et depuis 2005, un article est publié annuellement présentant les nouveautés.

Ensembl Genomes (ensemblgenomes.org) est l'analogue d'Ensembl, mais pour les organismes invertébrés. Cinq déclinaisons sont incluses dans Ensembl Genomes : EnsemblPlants, EnsemblMetazoa, EnsemblFungi, EnsemblProtists et EnsemblBacteria. Les interfaces ainsi que l'accès aux données sont similaires à Ensembl.

Dans le cadre de mes travaux de thèse, j'ai exploité cette ressource afin d'extraire les données comme les cartes exoniques (architectures des gènes) et les séquences génomiques des gènes d'intérêts. Un des avantages que j'ai pu tirer d'Ensembl par rapport aux autres ressources est sa facilité d'accès aux données ainsi que l'ajout de séquences supplémentaires en amont et en aval des gènes pour récupérer un contexte génomique étendu.

1.1.3 Le NCBI

Le *National Center for Biotechnology Information* (NCBI) regroupe un nombre important de ressources comme la base de données *Reference Sequence* (RefSeq), GenBank ou Taxonomy. Des informations et données supplémentaires ont pu être récupérées *via* ce portail en utilisant *E-utilities*. Ce dernier est un regroupement de neuf programmes côté serveur, qui au travers d'une syntaxe particulière, permet d'extraire les données. De plus, le NCBI intègre la base de données PubMed permettant l'accès à de nombreux articles en biomédecine et en sciences de la vie. A l'heure actuelle, il référence plus de 32 millions de documents. Cette ressource a été fortement mise à contribution lors de mes recherches.

1.2 Les jeux de référence ou *benchmarks*

En informatique, les jeux de référence (ou *benchmarks*) sont cruciaux lorsqu'il s'agit de comparer les performances des programmes entre eux. Il est important d'utiliser un *benchmark* communautaire comme "*gold standard*", afin d'être impartial dans l'évaluation des programmes.

Dans le contexte de l'annotation des génomes et plus spécifiquement lors de la prédiction *ab initio*, les *benchmarks* sont indispensables pour comparer les performances des programmes de prédictions des SE. Un des *benchmarks* de référence et probablement le plus utilisé est HS^{3}D (*Homo Sapiens Splice Sites Dataset*) (Pollastro and Rampone, 2002). Ce dernier contient 2 796 régions avec un SE donneur dont 65 non-canoniques et 2 880 régions avec un SE accepteur dont 74 non-canoniques. Il inclut également 271 937 séquences incluant un GT faux positif et 332 296 séquences incluant un AG faux positif. Le jeu de données HS^{3}D a été écarté de notre étude car il n'inclut que des SE humains. Par conséquent, d'autres jeux de référence ont été utilisés afin d'élargir le spectre des organismes. Les données ont été récupérées sur public.bmi.inf.ethz.ch/user/behr/splicing/. Quatre jeux de référence d'organismes modèles ont été récupérés : *A. thaliana*, *D. melanogaster*, *D. rerio* et *C. elegans*, ainsi qu'un cinquième

jeu de données incluant des données humaines (Sonnenburg *et al.*, 2007). Ces cinq jeux ont la particularité d'avoir été réalisés de manière homogène. Le tableau 3 référence les données de ces cinq *benchmarks*.

Organismes	Donneur		Accepteur	
	SE	Non-SE	SE	Non-SE
H.sapiens	160 601	76 335 126	158 217	54 469 623
D. rerio	143 495	33 175 785	143 509	19 350 438
D. melanogaster	29 788	4 126 777	29 501	2 126 899
C. elegans	64 844	2 846 598	64 838	1 777 912
A. thaliana	76 659	3 311 934	76 871	2 157 898

Tableau 3 - *Description des cinq jeux de références pour la prédiction des sites d'épissage*

1.3 Formats de fichiers

En bio-informatique on dénombre un grand nombre de formats de fichiers, souvent spécifiques à une application. Dans le cadre de mes travaux de thèse, j'ai principalement utilisé les formats de fichiers de type i) GFF3, ii) FASTA ainsi que le format iii) GAR qui est un format personnel.

i) Le format GFF3 (*General Feature Format version 3*) est un format permettant de décrire les gènes, leur structure interne ainsi que d'autres éléments de séquences nucléotidiques ou protéiques. Sa structure inclut 9 champs : le nom de la séquence, son origine, le type d'élément du gène, sa coordonnée de début, sa coordonnée de fin, un score de confiance, son brin, sa phase du cadre de lecture et enfin toutes autres informations relatives à l'élément du gène qui pourraient être utiles aux utilisateurs. Chaque ligne correspond à un élément de la structure interne de l'objet décrit.

ii) Le format FASTA est probablement le plus répandu et ne contient que 2 champs, l'en-tête précédé d'un chevron et la séquence biologique en elle-même (protéique ou nucléique). De plus, il est possible d'avoir plusieurs séquences (précédées de leur en-tête) dans un même fichier, on parle alors de fichier au format multi-fasta.

iii) Le format GAR (*Gene ARchitecture*) n'est pas un format standard, car je l'ai conçu afin de faciliter sa lecture et d'adapter certains de mes travaux. Il comporte cinq champs : le numéro de l'exon, la position de départ de l'exon, la position de fin de l'exon, la taille de l'exon et son brin. Chaque ligne correspond à un exon. Son en-tête est composé de différents éléments tels que les identifiants du gène ou du transcrit, le nombre d'exons et la longueur totale.

2 - Les programmes, outils et librairies

2.1 Alignements de protéines

2.1.1 OrthoInspector

OrthoInspector v3 (Nevers *et al.*, 2019) est un logiciel pour l'inférence des relations d'orthologie (*i.e.* un lien évolutif entre deux espèces issues d'un même ancêtre commun ayant subi un événement de spéciation) entre les gènes codant pour les protéines et une ressource en ligne pour accéder et interroger trois bases de données d'orthologie précalculées. Elles contiennent les séquences protéiques de 711 eucaryotes, 179 archées et 3 863 bactéries. OrthoInspector a principalement été utilisé pour extraire l'ensemble des séquences protéiques orthologues de séquences humaines pour *in fine*, construire des alignements multiples de séquences (MSA).

2.1.2 PipeAlign2

PipeAlign2 est un outil développé au laboratoire pour l'analyse automatisée de familles de protéines, par la construction d'alignements multiples de séquences complètes (Plewniak *et al.*, 2003). Il est disponible sur le web à l'adresse suivante : lbgi.fr/pipealign. Il intègre un processus en plusieurs étapes allant de la recherche de séquences homologues dans les bases de données de protéines et de structures 3D à l'annotation structurale et fonctionnelle. Dans le cadre de mes travaux, je l'ai exploité afin de réaliser des MSA à l'aide de MAFFT (Katoh *et al.*, 2002) qui a remplacé DbClustal (Thompson *et al.*, 2000), car il est plus adapté aux projets à haut débit. Deux programmes de détection et de correction d'erreurs ont été utilisés : Rascal (Thompson *et al.*, 2003) et SIBIS (Khenoussi *et al.*, 2014). Rascal a été utilisé afin de réaliser une correction préliminaire des MSA et SIBIS (v1.0) l'a secondé pour identifier les segments du MSA erronées

2.1.3 OrdAlie

OrdAlie (*Ordered Alignment Information Explorer*) est développé au laboratoire par Luc Moulinier. Il s'agit d'un programme en Tcl/Tk conçu pour explorer et analyser les protéines au niveau de leurs séquences ainsi que de leurs structures, fonctions et relations évolutives.

OrdAlie a notamment été utilisé pour visualiser les MSA et son éditeur a permis l'identification et la correction manuelle des séquences, mais nous reviendrons sur ce point dans la partie 6.

2.2 L'annotation des gènes codant pour des protéines

De nombreux programmes de prédiction de gènes *ab initio* ont été utilisés et comparés afin d'identifier leurs forces et leurs faiblesses au niveau des prédictions de la structure interne des gènes codant pour des protéines, mais également des séquences protéiques potentiellement traduites.

- Augustus v3.3.2 (Stanke and Waack, 2003) : prédit les gènes à partir de séquences génomiques eucaryotes ainsi que les protéines résultantes. Il est basé sur des modèles de Markov caché (HMM) du 4^{e} ordre. Augustus a la particularité d'inclure un grand nombre de modèles (>100) pour des organismes différents (métazoaires, alvéolata, plantes, algues, fungi, bactéries et archées).
- Genscan v1.0 (Burge and Karlin, 1997) : identifie la structure de gène complète au sein des séquences génomiques. Son algorithme est basé sur un HMM du 3^{e} ordre et il n'y a que 3 modèles disponibles (Humain, *A. thaliana* et *Z. mays*).
- Snap v2006/07/28 (Korf, 2004) : programme de recherche de gènes adapté aux génomes eucaryotes et procaryotes. Il est basé sur un HMM du 4^{e} ordre et inclut 11 modèles différents.
- GlimmerHMM v3.02 (Majoros *et al.*, 2004) : prédicteur de gènes basé sur un modèle de Markov caché généralisé (GHMM). Il inclut aussi un modèle de prédiction de SE issue de GeneSplicer (Pertea *et al.*, 2001) et d'un arbre de décision adapté de GlimmerM (Salzberg *et al.*, 1999). Il inclut 5 modèles différents.
- GeneID v1.4 (Guigó *et al.*, 1992) : identifie la structure des gènes avec un HMM du 5^{e} ordre de manière hiérarchique et est basé sur 66 modèles. GeneID localise d'abord les TSS et les SE, puis il construit des séquences exoniques les plus probables et finalise la construction en assemblant tous les exons.

Des programmes plus spécifiques ont également été utilisés et évalués, ils ont la particularité de prédire les SE.

- SpliceFinder v1 (Wang *et al.*, 2019) : outil de prédiction de SE basé sur un réseau de neurones convolutif à une couche. Lors de son entraînement, il utilise un processus de

reconstruction du jeu d'entraînement en injectant les faux positifs obtenus dans le jeu négatif lors d'une seconde phase d'entraînement.

- MaxEntScan v1 (Yeo and Burge, 2004) : programme de prédiction de SE basé sur le principe d'entropie maximale. Une de ces particularités est d'exploiter un très petit contexte génomique pour les prédictions (9 bases pour les SE donneur et 23 bases pour les SE accepteur).
- DSSP v1 (Naito, 2018) : système de prédiction de SE basé sur une architecture de réseau de neurones hybride incluant un CNN à 2 couches et un RNN. Il prend en compte des entrées d'une longueur de 140 nucléotides.
- NNSplice v0.9 (Reese *et al.*, 1997) : prédicteur de SE basé sur un réseau de neurones, il s'agit d'une sous-couche au programme de prédiction de gènes Genie.

2.3 Intelligence artificielle

2.3.1 Deep learning

Dans le cadre de ma thèse, j'ai utilisé Tensorflow v2.1.4 lorsque j'ai voulu exploiter la puissance de l'IA. Tensorflow (Abadi *et al.*, 2016) est une boîte à outils *open source* idéale pour le développement et l'entraînement de modèles basés sur des algorithmes de *deep learning* dont les réseaux de neurones convolutifs. L'avantage de Tensorflow est sa prise en main facile, sa communauté grandissante et la quantité importante de ressources et de tutoriaux. Tensorflow peut être utilisé avec le langage python grâce à son API en *front-end* Keras. Cependant, l'exécution des calculs est réalisée en C++ afin d'optimiser les performances. Tensorflow a également l'énorme avantage de pouvoir être exécuté sur plusieurs CPU et même des GPU, car il a été optimisé avec des architectures comme CUDA, ce qui lui permet d'avoir une puissance de calcul importante et donc d'augmenter ses performances.

Keras v2.3.1 (github.com/fchollet/keras) est l'API *front-end* de haut niveau de Tensorflow pour le langage python. Cela signifie que Keras fournit un niveau d'abstraction supérieur, afin que l'utilisateur ne se préoccupe que du développement de l'architecture du RN et des éléments autour, et non pas de la mise en place des calculs. Keras a été conçu pour être utilisé par l'humain et non par une machine, en ce sens, il est simple d'utilisation et flexible.

La librairie Scikit-learn v0.23.2 (Pedregosa *et al.*, 2011) est une librairie spécialisée dans le *machine learning* et exploitable avec python. Elle inclut de nombreuses fonctionnalités

comme la conceptualisation et l'entraînement de modèle de *machine learning*. Dans le cadre de ma thèse, je l'ai principalement utilisé lors de l'étape de pré-traitement des données. Par exemple, en utilisant la fonctionnalité qui réalise la distribution aléatoire des données pour former le jeu d'entraînement et de test.

2.3.2 Programmation génétique

Pour manipuler la programmation génétique, la plateforme logicielle EASEA (easea.unistra.fr), développée et maintenue au laboratoire, a été utilisée. EASEA (*EAsy Specification of Evolutionary Algorithms*) (Collet *et al.*, 2000) est une plateforme pour la mise en place d'AE, permettant aux chercheurs d'exploiter la puissance de calcul massivement parallèle (sur CPU et GPU) pour résoudre des problèmes complexes. La plateforme EASEA est *open source* et multi-système (Linux, MacOS et Windows). Elle embarque un compilateur et met à disposition des templates facilitant l'implémentation d'AE en C++.

En programmation génétique, les individus sont représentés par des arbres (réseau acyclique composé de nœuds reliés par des liens). Chaque arbre est constitué de noeuds internes (composées des opérateurs) et de feuilles terminales (composées des variables ou des constantes). Pour exploiter pleinement cette structure, nous avons conceptualisé des automates à états finis représentés par des expressions régulières (ou expression rationnelle). Ainsi, nous avons supposé que le spliceosome puisse être considéré comme un automate à états finis, acceptant ou rejetant des mots (ou motifs) écrits à l'aide de l'alphabet $\{A, C, G, T\}$ suivant qu'ils soient détectés comme séquences introniques ou exoniques.

Selon son article (Kleene, 1951), Kleene propose un algorithme transformant tout automate à états finis en expression régulière et inversement, prouvant ainsi l'équivalence parfaite entre expressions régulières et automates à états finis. Comme il est possible de représenter une expression régulière comme une fonction mathématique composée d'opérateurs et opérandes (figure 37), cela signifie qu'on peut utiliser la programmation génétique pour faire évoluer des automates à états finis, et donc, possiblement, un équivalent artificiel de spliceosome capable de déterminer les frontières entre les régions exoniques et intronique.

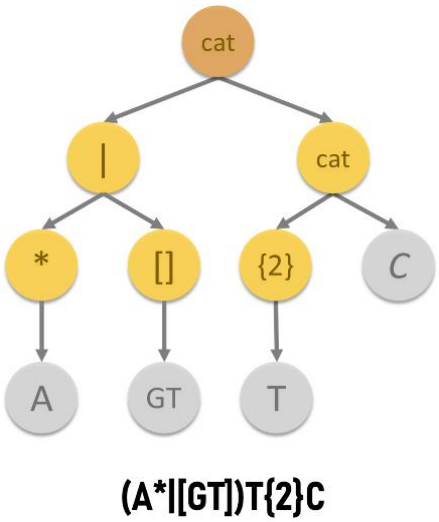


Figure 37 - *Représentation d'une expression régulière sous forme d'arbre. Le nœud orange représente la racine de l'arbre, les nœuds jaunes sont les nœuds internes composées des opérateurs et les nœuds gris sont les feuille terminales composées des constantes/variables.*

Les résultats obtenus par PG sont décrits dans le Chapitre 8 : SpliceSLEIA - Stratégie évolutive pour la prédiction de sites d'épissage.

2.3.3 Les expressions régulières

Une expression régulière est une chaîne de caractères (incluant plusieurs opérateurs et constantes) qui décrit, selon une syntaxe précise, un ensemble de chaînes de caractères possibles. Les éléments principaux inclus dans cette syntaxe sont décrits dans le tableau 4. La librairie utilisée pour implémenter les expressions régulières est Google RE2 (github.com/google/re2) écrite en C++.

Opérateurs	Symboles	Description
Concaténation	cat	Concatène deux éléments
Choix alternatif	|	Opérateur de choix entre plusieurs solutions alternatives
Quantifieur 0,1	?	Définit un groupe présent une seule fois, ou qui n'est jamais présent
Quantifieur 0,*n*	*	Définit un groupe qui est présent *n* fois ou qui n'est jamais présent
Quantifieur *n*	+	Définit un groupe qui est présent *n* fois (avec $n \geq 1$)
Accolades	{ }	Définit le nombre de fois que le motif doit être répété
Crochets	[ ]	Définit une liste de choix entre les éléments présent entre les crochets
Crochets exclusifs	[^]	Définit une liste de choix entre les éléments qui ne sont pas présent entre les crochets
Parenthèses	( )	Définit un groupe de capture (un motif particulier)‡
Joker	.	Correspond à n'importe quelle éléments terminaux (ici : A, C, G ou T)

Tableau 4 - *Opérateurs classiques des expressions régulières. (‡ dans notre étude, nous avons utilisé les parenthèses comme groupe non-capturant, afin de faciliter la lisibilité et pour respecter les priorités de chaque opérateur).*

3 - Développement logiciel

3.1 Python

En dehors de la programmation génétique écrite avec le langage EASEA, L'ensemble des scripts et programmes développés durant ma thèse ont été écrits en python v3.6 (et plus). Python est un langage de programmation interprété, polyvalent avec un typage fort et largement utilisé par la communauté des bio-informaticiens. Il a été conçu de manière à mettre l'accent sur la lisibilité du code ainsi que sur une syntaxe facilement abordable. Python a de nombreux avantages comme la gestion automatique de la mémoire, il supporte les multiples paradigmes de programmation comme la programmation orientée objet, ou le style procédural. De plus, de nombreuses librairies sont disponibles notamment en biologie et en IA (comme Keras et Scikit-learn).

3.2 Librairies python

Dans le cadre de mes travaux, j'ai eu l'occasion de manipuler de nombreuses librairies dont certaines très répandues, comme Pandas, Matplotlib et Seaborn. Pandas (pandas.pydata.org) est une librairie utilisée pour la science des données et permet d'organiser et manipuler facilement les données grâce à des structures de données flexibles. Les principaux objets manipulés sont des *DataFrames*, il s'agit de tableaux plus perfectionnés. L'avantage de cette librairie est sa compatibilité avec de nombreuses autres, notamment les librairies de visualisation comme Matplotlib (matplotlib.org) et Seaborn (seaborn.pydata.org). Ces dernières permettent de visualiser les données avec une multitude de graphiques différents.

La boîte à outils *open source* Biopython (biopython.org) a également été utilisée, elle permet la manipulation et le traitement de données biologiques. Biopython inclut de nombreuses fonctionnalités comme la prise en charge de séquences génomiques ou la gestion des fichiers spécifiques à la biologie. De plus, il permet d'interagir avec des bases de données comme celles du NCBI.

4 - Développement web

Afin de partager plus aisément les données et outils avec la communauté scientifique, j'ai eu recours au développement web. Ainsi, la méthode CGI (*Common Gateway Interface*) a été utilisée afin de créer une interface entre le programme (écrit en python) et le serveur.

4.1 HTML et CSS

Le site web a été conçu en utilisant le principal langage balisé HTML5 (*HyperText Markup Language*) afin de structurer la sémantique des différentes pages et de les afficher. En parallèle, le langage CSS3 (*Cascading Style Sheets*) a été utilisé pour mettre en forme les pages web au niveau de leur apparence et de l'ergonomie. Ces deux langages sont normés par le *World Wide Web Consortium* (W3C). Les pages web sont ensuite affichées et consultables *via* un navigateur web comme Google Chrome ou Safari.

4.2 JavaScript, jQuery et AJAX

Pour que l'utilisateur puisse exploiter pleinement le potentiel des outils développés, les pages web sont rendues dynamiques et interactives à l'aide du langage de programmation web JavaScript, compatible avec les principaux navigateurs web. JavaScript est un langage de script multiparadigme, qui prend en charge les styles de programmation orientée objet. Il a la particularité d'être exécuté côté client (à l'inverse d'autres langages de programmation web comme PHP qui sont exécutés côté serveur), cela signifie que c'est le navigateur web qui va prendre en charge l'exécution des scripts. JavaScript permet d'interagir avec les éléments de la page HTML *via* le DOM (*Document Object Model*).

jQuery est une bibliothèque JavaScript *open-source* multi-navigateurs. Elle facilite le développement des scripts par la gestion des éléments du DOM. De nombreuses fonctionnalités sont disponibles comme l'accessibilité des classes et des attributs, la modification du CSS, la mise en place d'animations pour rendre les pages web plus dynamiques, ainsi qu'une prise en main plus pratique du protocole AJAX.

Le protocole AJAX (*Asynchronous JavaScript and XML*) permet une communication asynchrone entre le client et le serveur sans interférer avec l'affichage et les opérations de la page web en cours. AJAX permet de communiquer par l'intermédiaire d'un système de requête-réponse. Ainsi, lorsque l'utilisateur envoie une requête sur le site web, elle passe par le

protocole AJAX et est renvoyée sur les serveurs afin de procéder aux calculs. Le résultat est ensuite retransmis au client sur la page web *via* le protocole AJAX. Cette mécanique évite d'exploiter les ressources du client qui peuvent être limitées. La figure 38 représente le protocole AJAX.

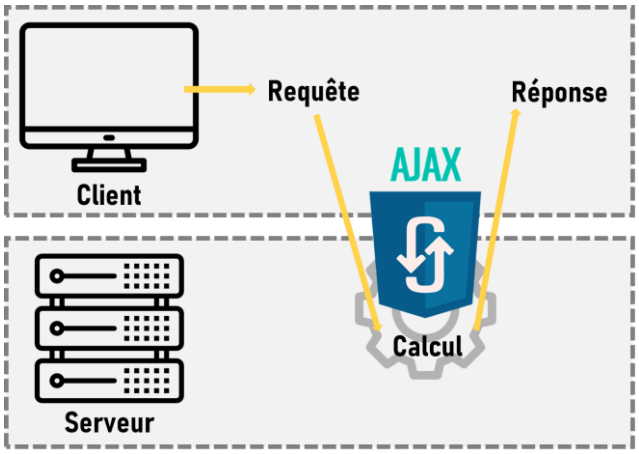


Figure 38 - *Protocole AJAX utilisé pour interagir entre le client et le serveur. Le client envoie une requête qui est prise en charge par le protocole AJAX et envoyée aux serveurs. Ce dernier effectue les calculs puis envoie les résultats côté client.*

5 - Architectures des réseaux de neurones convolutifs

L'intelligence artificielle a joué un rôle prépondérant durant cette thèse. J'ai eu l'opportunité de travailler sur des algorithmes de *deep learning* dont les réseaux de neurones convolutifs. Ces derniers ont été présentés dans le chapitre d'introduction, par conséquent, dans cette partie, j'aborderai les aspects algorithmiques et l'architecture des CNN développés dans cette thèse.

5.1 Les couches de convolution

Les couches de convolution sont composées de noyaux de convolution (ou filtre ou *kernel*) qui opèrent localement. Chaque noyau parcourt la couche précédente en effectuant des convolutions, c'est-à-dire une opération mathématique combinant deux signaux d'entrées pour fournir un signal de sortie. La figure 39 représente une convolution. Généralement, les noyaux de convolution ont une taille de $3 * 3$ ou $5 * 5$ lorsque les données sont en deux dimensions (image en noir et blanc par exemple). Ainsi, pour un noyau de $3 * 3$, le nombre de poids sera de 9 et pour un noyau de $5 * 5$, il y aura 25 poids. Leur organisation spécifique fait en sorte que

les poids sont partagés lors de chaque convolution, ce qui réduit drastiquement le nombre de paramètres. De plus, chaque noyau se focalise sur la détection d'un motif ou d'une caractéristique particulière (détection de contours avec un filtre Sobel, de courbes, etc.) (Albawi *et al.*, 2017), dans l'exemple de la figure 39, le noyau détecte des diagonales. L'ajustement des poids est réalisé par l'algorithme de rétropropagation du gradient de l'erreur, par conséquent, le type de motif à apprendre est défini par le réseau lui-même. Ainsi, si un motif est détecté à un endroit de l'image, il pourra être détecté ailleurs dans l'image (LeCun *et al.*, 2015).

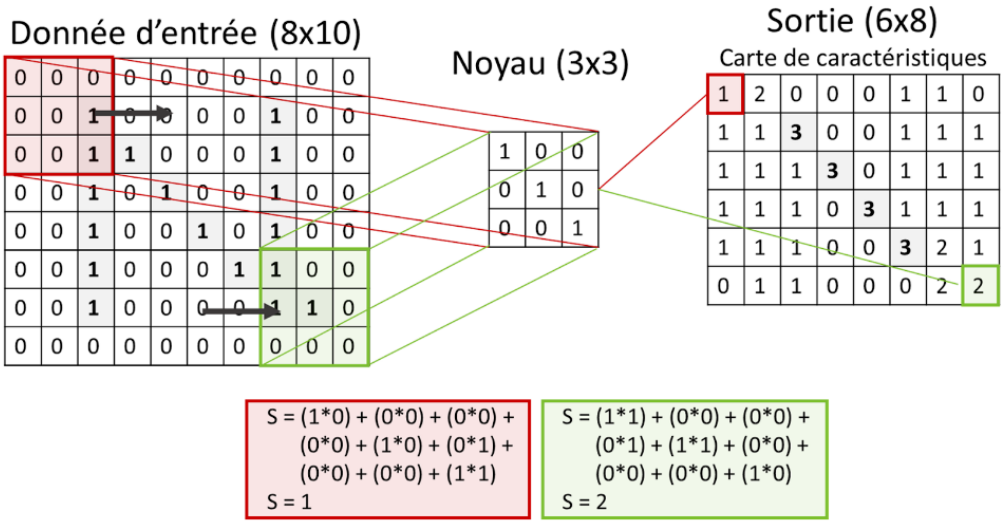


Figure 39 - *Représentation d'une opération de convolution. L'entrée est composée d'un tableau en 2 dimensions de 8 par 10 pixels. Le noyau de 3x3 est composé de 9 poids, caractérisant un motif particulier, ici une diagonale. L'opération consiste à calculer la convolution sur chaque surface de 3x3 de la donnée d'entrée puis de se déplacer de 1 pixel à chaque fois (si le décalage (stride) est de 1). La première convolution est effectuée sur le carré en rouge, après l'opération de convolution, le résultat obtenu est 1. Lorsque l'ensemble de la donnée d'entrée a été parcouru par le noyau, on obtient une carte de caractéristiques censée accentuer un motif. Dans notre cas, on observe que la diagonale est bien représentée (cases en gris). La carte des caractéristiques a augmenté le niveau d'abstraction.*

Le pas (*stride*) est une technique développée pour accélérer les calculs, il s'agit de la valeur de décalage entre chaque étape de convolution par un noyau. Un pas de 1 signifie que le noyau se déplace de 1 pixel à chaque fois, un pas de 2 se déplacera de 2 pixels et aura comme effet de réduire la dimension (O'Shea and Nash, 2015). En effet, si un pas de 2 était appliqué sur l'exemple de la figure 39, une carte de caractéristiques de $4 * 5$ en résulterait. Cependant, cela induirait un problème de bornes, car lorsque le noyau atteint la fin d'une ligne, il ne prend en compte que 6 pixels sur 9, la dernière colonne est hors cadre, car l'image est trop petite. Pour pallier cela, une autre technique a été développée, le zéro *padding*. Cela consiste à ajouter des zéros partout où le noyau dépasse de l'image.

Une fois que l'opération de convolution est réalisée, il en résulte des cartes de caractéristiques (*feature maps*) qui augmentent le niveau d'abstraction de la donnée d'entrée. Il y a une carte de caractéristiques par noyau, où chaque pixel représente un neurone avec les mêmes paramètres (poids et biais). Ces paramètres varient seulement pour des neurones qui ne sont pas de la même carte de caractéristiques. L'étape suivante consiste à réaliser un *pooling*.

5.2 Les couches de *pooling*

Les couches de *pooling* (ou de regroupement) sont une agrégation d'un ensemble de valeurs. Elles consistent à extraire les caractéristiques importantes réduisant la dimension des cartes de caractéristiques en prenant en compte la moyenne ou la valeur maximale d'une zone locale. Le *pooling* peut éliminer environ 75% de l'information tout en assurant le maintien des motifs pertinents (Akhtar and Ragavendran, 2020). La figure 40 représente l'étape de *pooling*. Elle permet également de réduire le nombre de paramètres à calculer et d'augmenter le niveau d'abstraction et la vitesse de calcul. En général, les zones de *pooling* sont petites (*e.g.* 2x2) (Chen *et al.*, 2017) pour limiter le sur-apprentissage et le pas est en général de 2.

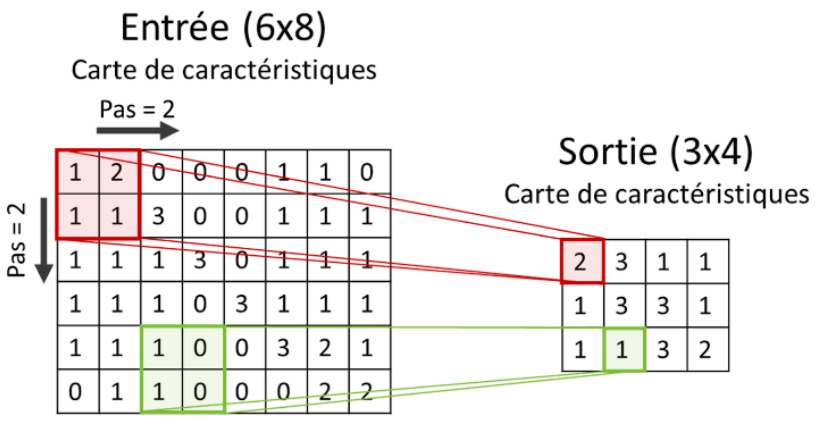


Figure 40 - *Représentation du processus de pooling. La zone de pooling est un carré de 2x2. Elle se déplace d'un pas de 2, réduisant ainsi la dimension de la carte de caractéristiques de moitié. Le processus de pooling sélectionné ici correspond au max pooling, c'est-à-dire que pour chaque surface explorée par la zone de pooling, on récupère uniquement la valeur la plus élevée.*

Parmi les méthodes de *pooling* les plus répandues, on retrouve le max *pooling* et le *pooling* moyen (*average pooling*). La première méthode consiste à prendre la valeur la plus élevée dans la zone de pooling. Elle se caractérise par la fonction $maxpool(x) = max(x_i)$ avec $i \in \{1, n\}$, n étant le nombre de neurones pris en compte par la zone de *pooling*. Le max *pooling* permet d'accentuer les caractéristiques importantes. La deuxième méthode, proche de

la première, calcule la moyenne des valeurs prises en compte par la zone de pooling elle est calculée par $avgpool(x) = \frac{1}{N}\sum_{i=1}^{N} x_i$ avec $i \in \{1, n\}$ (Suárez-Paniagua and Segura-Bedmar, 2018). Cette méthode permet de mettre en avant toutes les caractéristiques locales. D'autres méthodes de *pooling* sont disponibles comme le *mixed pooling*, *LEAP pooling*, *Kernel pooling*, *Stochastic pooling* (Akhtar and Ragavendran, 2020), cependant elles sont moins utilisées que le max *pooling*.

5.3 Le *dropout*

Le *dropout* est une technique de régularisation qui permet d'améliorer la généralisation des données et de limiter le sur-apprentissage (Hinton *et al.*, 2012). La méthode consiste à sélectionner aléatoirement un certain pourcentage (en général 20%) de neurones dans une couche donnée et de les inactiver provisoirement. Cependant, cette méthode augmente le temps d'apprentissage (Srivastava *et al.*, 2014). Le *dropout* est généralement intégré après les couches de convolution et de *pooling*.

5.4 Les couches entièrement connectées

Les couches entièrement connectées (ou *fully connected*) sont les dernières couches des CNN. Elles nécessitent une étape de mise à plat (*flatten layer*) des cartes de caractéristiques sous la forme d'un vecteur linéaire (figure 41). Le nombre de couches entièrement connectées est dépendant du problème, mais il n'y a pas de restrictions, si ce n'est que la dernière couche du réseau participe aux résultats finaux. Ainsi, dans le cadre d'une classification, il est nécessaire que le nombre de neurones de la dernière couche corresponde au nombre de classes et que la fonction d'activation soit une sigmoïde de type *softmax* pour établir des probabilités.

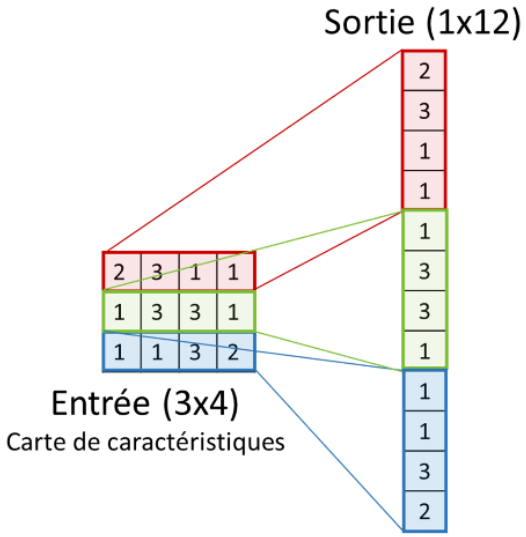


Figure 41 - *Processus de mise à plat des cartes de caractéristiques afin de créer un vecteur linéaire. Ce vecteur participe à la prédiction finale.*

5.5 Exploitation des CNN sur des données multi-canaux

Les CNN peuvent traiter des formats de données différents comme des tenseurs du premier ordre (vecteurs = 1D) ou du second (matrice = 2D). Dans les paragraphes précédents, nous avons pris comme exemple une image en noir et blanc convertie en une matrice. Dans cette partie, nous allons intégrer une nouvelle dimension en prenant en compte des images en couleur. En informatique, les couleurs sont codées par un triplet de valeurs (rouge, vert, bleu (RVB)), compris entre 0 et 255, la synthèse de ces valeurs définit la couleur du pixel. Ainsi, pour que les CNN exploitent cette information, il est nécessaire d'ajouter une nouvelle dimension appelée profondeur, qui est caractérisée par le nombre de canaux (*channels*) (LeCun *et al.*, 2015). Lorsque l'on traite une image en couleur avec un CNN, le nombre de canaux est de trois, un pour chaque couleur RVB. Les noyaux appliqués sont donc à la même dimension que la donnée d'entrée et sont caractérisés par $H * L * C$, avec H la hauteur du filtre, L sa largeur et C le nombre de canaux. La figure 42 représente le processus de convolution à partir d'une image en couleur.

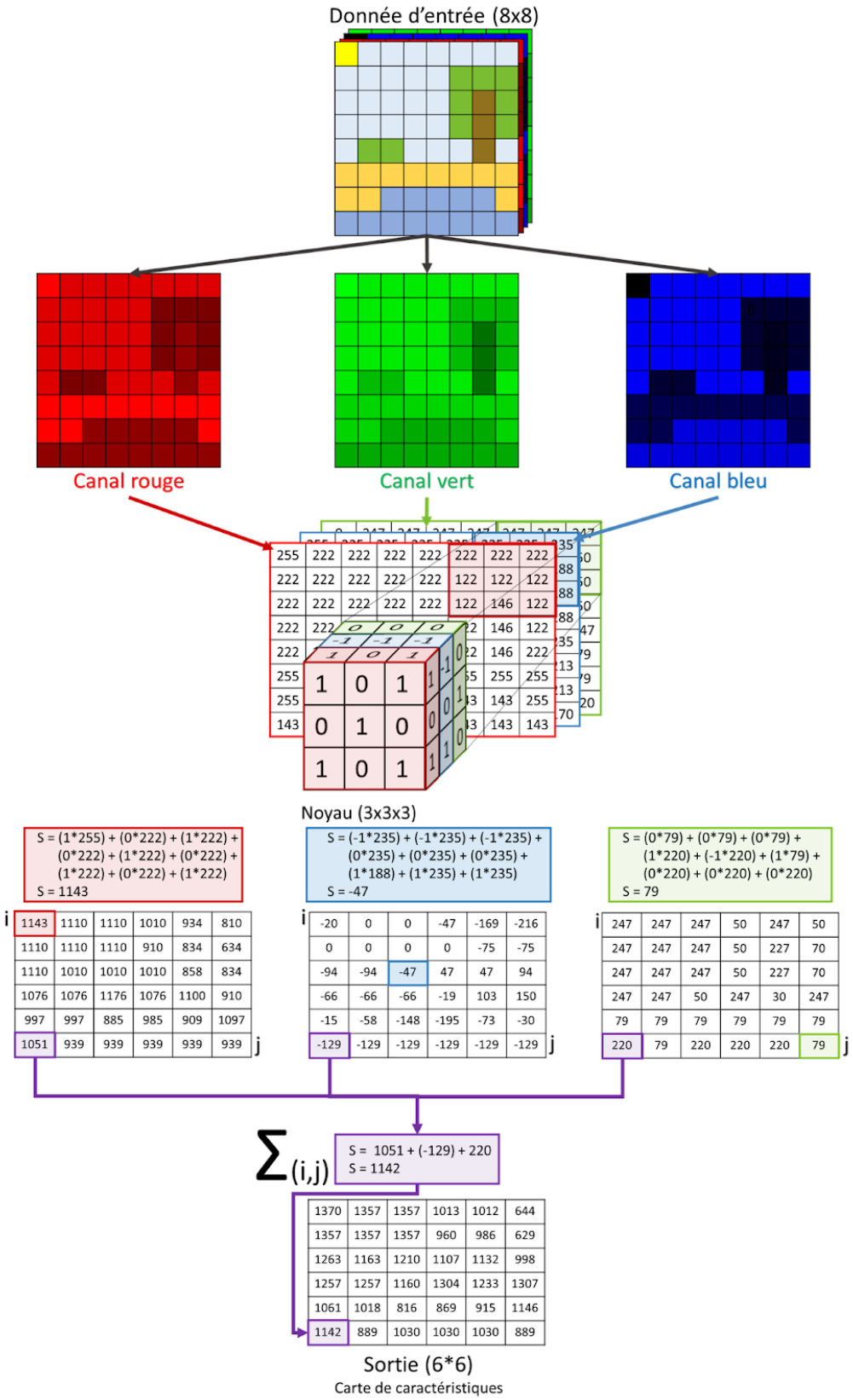


Figure 42 - *Fonctionnement du processus de convolution sur une image en couleur. La première étape consiste à scinder la donnée initiale en 3 canaux, un pour chaque couleur RVB. Chaque noyau est ensuite appliqué sur les différents canaux afin d'obtenir des cartes de caractéristiques temporaires. Les valeurs des éléments (i,j) sont sommées pour finalement obtenir la valeur (i,j) de la carte de caractéristiques.*

Ce système de canaux est aussi utilisé lorsque l'on traite des tenseurs du premier ordre. Dans le cadre de séquences d'acide désoxyribonucléique (ADN) par exemple, il est important de réaliser une étape initiale d'encodage appelé *one-hot*, afin de transformer les données d'entrée en données interprétables par l'ordinateur. Chaque élément du vocabulaire (ici les nucléotides A, C, G, T) est converti en un vecteur de taille équivalent à la cardinalité du vocabulaire. L'ensemble des valeurs du vecteur est vide, à l'exception de la position *i* correspondant au $i^{éme}$ élément du vocabulaire. L'encodage *one-hot* des nucléotides est présenté dans la figure 43. Malheureusement, ce type d'encodage est restreint à un vocabulaire de petite taille, mais certains travaux tentent de résoudre ce problème (Zhang and LeCun, 2017).

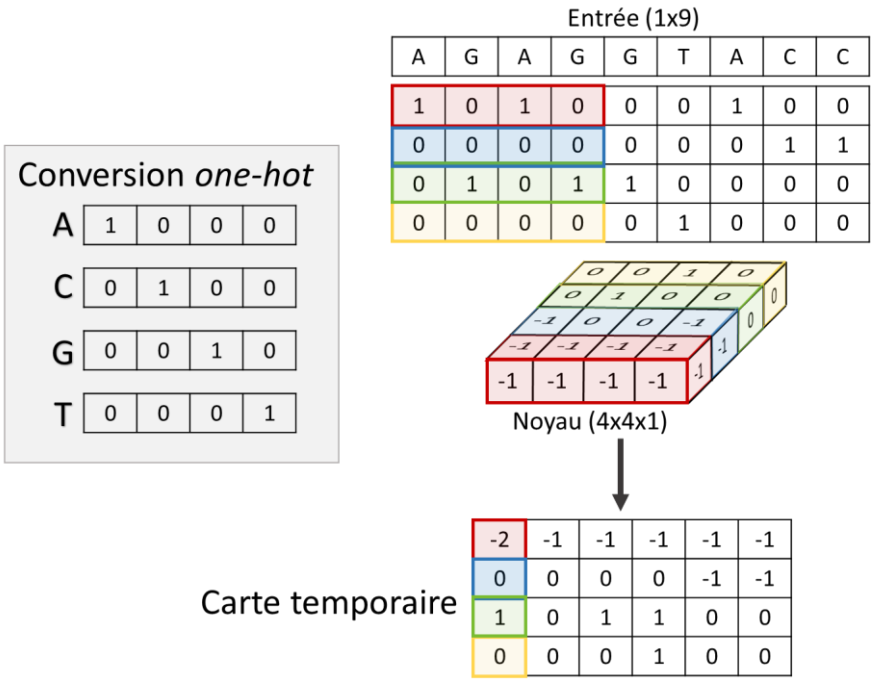


Figure 43 - *Processus de convolution sur des tenseurs du premier ordre de type séquence génomique. Dans un premier temps, les données sont encodées au format one-hot, puis un noyau de profondeur n est appliqué sur les n canaux.*

6 - Analyse de séquences et caractérisation des erreurs

Une grande partie des travaux de cette thèse est basée sur les données (qui vont de pair avec les algorithmes d'IA) et notamment leur qualité. Ainsi, pour identifier les séquences erronées issues de bases de données publiques de référence, un protocole robuste de reconnaissance d'erreurs dans les séquences protéiques a été élaboré.

6.1 Extraction des données

La première étape consiste à extraire les séquences protéiques de gènes humains d'intérêt. Dans notre étude, nous avons initié le protocole avec des gènes étudiés au sein de l'équipe dont la complexité en termes de longueur, composition, distribution phylogénétique étendue... était importante et connue. Ces gènes correspondent à 20 gènes humains responsables d'une maladie génétique rare : le BBS (*Bardet-Biedl Syndrom*) ainsi que 25 gènes humains impliqués dans des myopathies. Les protéines orthologues de ces 45 gènes de référence ont ensuite été extraites avec l'outil OrthoInspector. 8 594 protéines (dont les 45 protéines humaines) composent le jeu initial. Ce dernier intègre de nombreuses espèces issues de plusieurs clades eucaryotes, allant des primates aux protistes, afin d'être le plus représentatif du domaine eucaryote et incluant les métazoaires, les fungi, les amœbozoaires, les viridiplantae, les straménopiles ou encore les alvéolata.

6.2 Réalisation des alignements multiples de séquences

Pour chaque gène humain de référence, un MSA a été construit avec l'ensemble des orthologues récupérés, en utilisant le protocole de PipeAlign2. Ainsi, 45 MSA ont été construits, la figure 44 représente le nombre de séquences pour chaque alignement. SIBIS et RASCAL ont établi un premier pré-traitement pour supprimer les séquences les plus aberrantes.

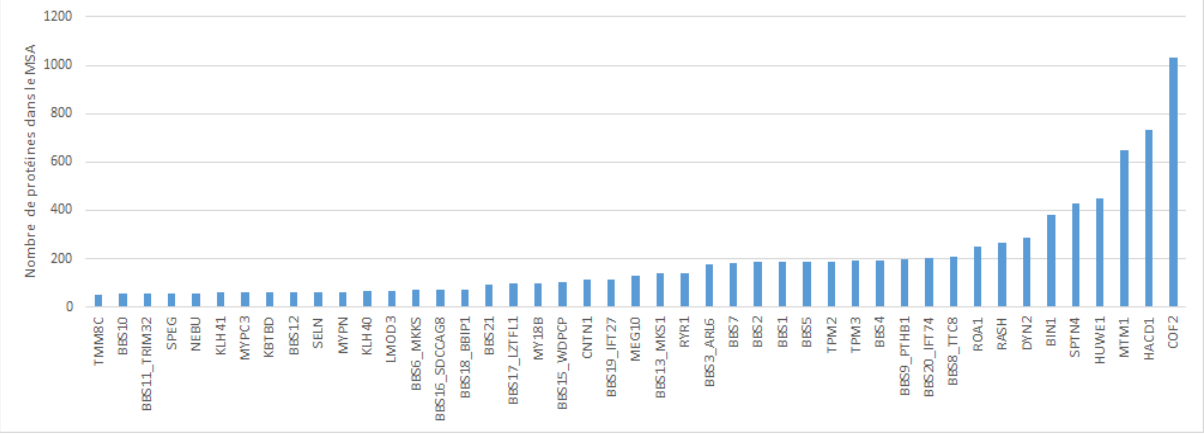


Figure 44 - *Nombre de protéines constituant chaque alignement multiple de séquences.*

6.3 Classification des erreurs

La nomenclature de classification des erreurs de SIBIS a été utilisée. Chaque erreur est classée selon son type : insertion, délétion ou *mismatch* (correspond à un fragment de séquence erroné). Les erreurs sont ensuite caractérisées en fonction de leur localisation au sein d'une séquence, elles peuvent se situer soit en position N-terminale, C-terminale ou interne. Par conséquent il y a 9 catégories possibles : délétion N-terminale, insertion N-terminale, *mismatch* N-terminal, délétion C-terminale, insertion C-terminale, *mismatch* C-terminal, délétion interne, insertion interne et *mismatch* interne.

6.4 Identification des erreurs

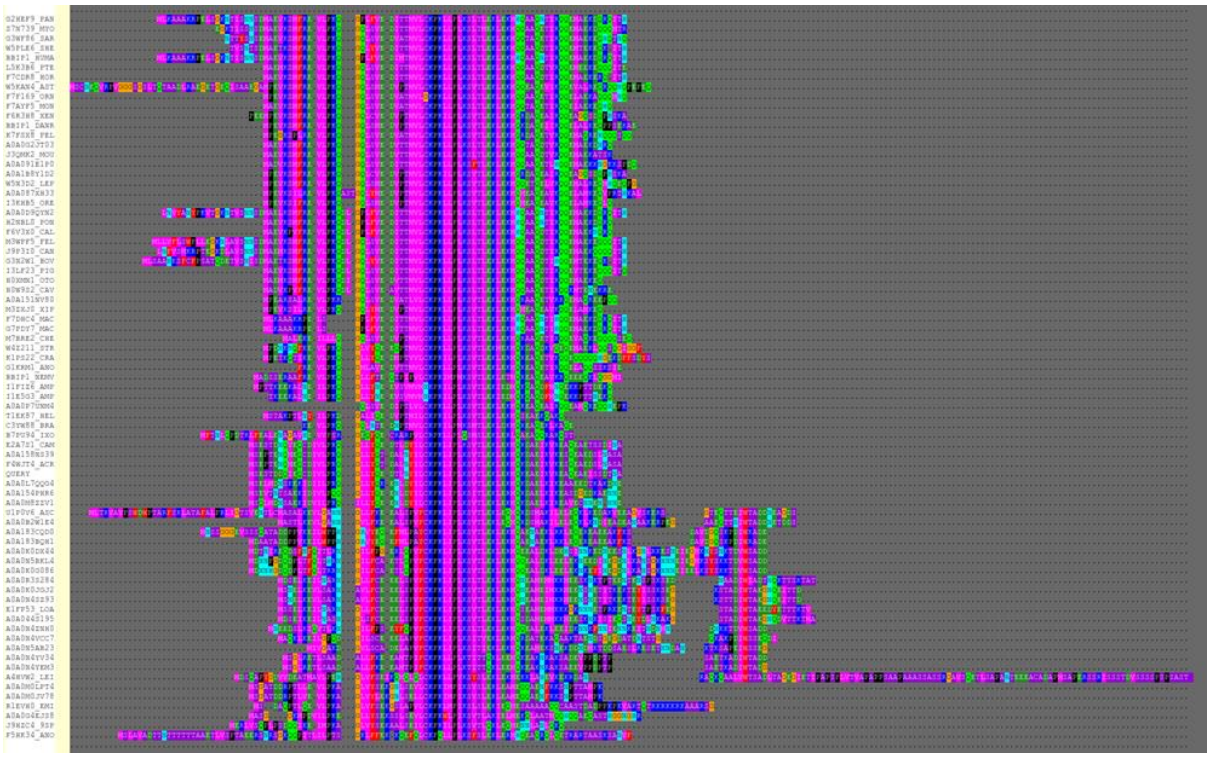


Figure 45 - *Alignement multiple de la séquence protéique du gène BBS18 visualisé avec l'outil OrdAlie. Le MSA contient 76 séquences orthologues d'organismes différents. A partir de cet MSA, on peut identifier les séquences contenant des erreurs ou non.*

Afin d'analyser et visualiser ces MSA pour identifier les erreurs, l'outil OrdAlie a été utilisé (figure 45). Les erreurs sont des événements discordants dans les séquences protéiques et on observe rarement la même erreur sur différentes séquences du MSA (à l'exception des erreurs dues à l'homologie qui se sont propagées dans les banques de données). Ainsi, une même erreur n'est pas observée chez les séquences orthologues d'organismes

phylogénétiquement proches. Par conséquent différents critères sont utilisées pour identifier les erreurs :

- La séquence protéique : certains enchaînements d'acides aminés (AA) sont peu probables comme des répétitions. La figure 46A représente des répétitions d'AA improbables notamment lorsqu'elles sont présentes en position C-ter.
- La présence d'une méthionine (M) alternative qui aboutit à une insertion ou délétion N-terminales (figure 46B).
- L'absence d'AA ou motif quasi-invariants (figure 46C), (figure 46D) (*e.g.* sites catalytiques). Sur la figure 46C, compte tenu des séquences proches et du haut taux de conservation de la proline, il est très peu probable que l'absence de la proline ne soit pas une erreur.

Schématiquement, pour identifier une séquence protéique erronée, on s'appuie sur les séquences des plus proches voisins. Ainsi, si on observe une position présentant une divergence importante et non observée chez les proches voisins, on suppose que la séquence à cette position est fausse et on répartit les erreurs selon les types prédéfinis : insertion (figure 46E), délétions (figure 46F) et *mismatch* (figure 46G). Cependant, il est important de prendre en compte la notion de divergence évolutive. En effet, la variabilité au sein des séquences peut être due à l'évolution et non pas à une erreur. Cela est illustré par la figure 46H. Dans cet exemple, il ne s'agit pas d'une erreur, mais d'une divergence évolutive. Un argument supplémentaire qui complète cette caractérisation est que les permutations S ↔ T ↔ N sont très fréquentes dans les tables de substitutions (BLOSUM, PAM…) car il s'agit d'AA de la même famille physico-chimique (*i.e.* des AA avec une chaîne apolaire).

Une fois que les séquences ont été caractérisées, elles sont classées comme "*Confirmed*" si aucune erreur n'a été identifiée et "*Unconfirmed*" si elles possèdent au moins une erreur. Cependant, dans certains cas, il est trop difficile de différencier une erreur d'une divergence évolutive. Dans ce cas, pour assurer la haute qualité finale, nous avons décidé d'exclure les séquences incertaines.

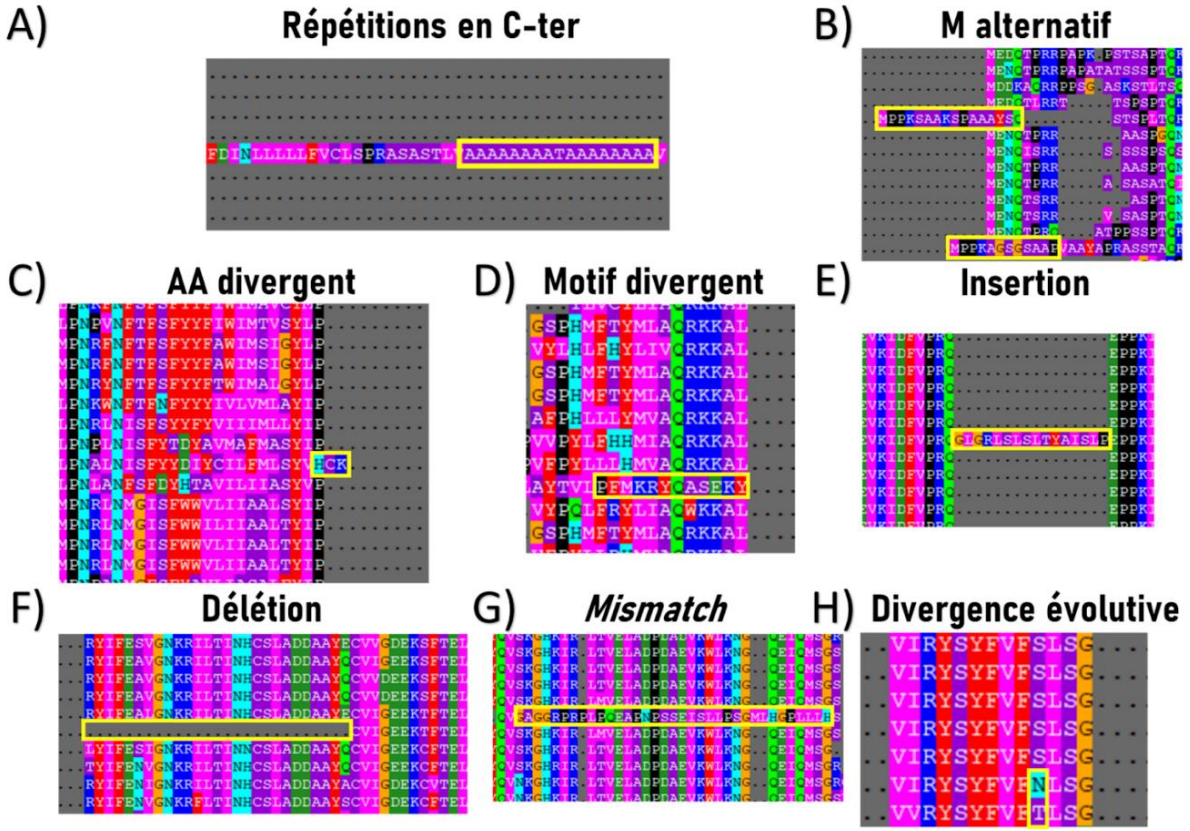


Figure 46 - *Caractérisation de segments protéiques et identification des erreurs.* ***A)*** *Les séquences présentent un nombre important d'AA qui se répètent. Dans le cas d'une répétition d'alanine (A) en position C-ter, cela peut être suspect, car les codons induisant cet AA ont un fort taux en GC donc propice aux répétitions.* ***B)*** *Identification d'AA méthionine (M) alternatifs, dû à un codon start précoce, induisant un fragment en N-ter supplémentaire et générant donc une protéine anormale.* ***C)*** *L'ensemble des protéines du MSA présentent une Proline, et une seule séquence n'en possède pas, il est donc fortement probable qu'il s'agisse d'une erreur.* ***D)*** *Cette erreur ressemble fortement à la précédente à l'exception qu'il s'agit d'un segment et non plus d'un AA unique.* ***E)*** *Insertion d'un fragment* ***F)*** *Suppression d'un fragment, à l'instar des insertions, ils sont facilement identifiables et peuvent être plus ou moins long.* ***G)*** *Séquence alignée, mais divergente. Ce type d'erreur peut être compliqué à identifier, car il peut s'agir d'une divergence évolutive comme présentée en* ***H)****.*

6.5 Identification des isoformes

L'identification des isoformes est un problème important, car, à titre d'exemple, l'épissage alternatif concerne 95% des protéines humaines (Pan *et al.*, 2008). Ainsi, pour déterminer si le fragment ajouté (ou supprimé) est dû à la présence d'une isoforme, il faut analyser l'ensemble de l'alignement et vérifier si d'autres séquences ont été prédites avec ce segment. Dans certains cas, cela peut être fastidieux, car les erreurs peuvent se propager dans les bases de données, dès lors, en cas de doute, les séquences sont supprimées. La figure 47 présente le cas d'isoformes.

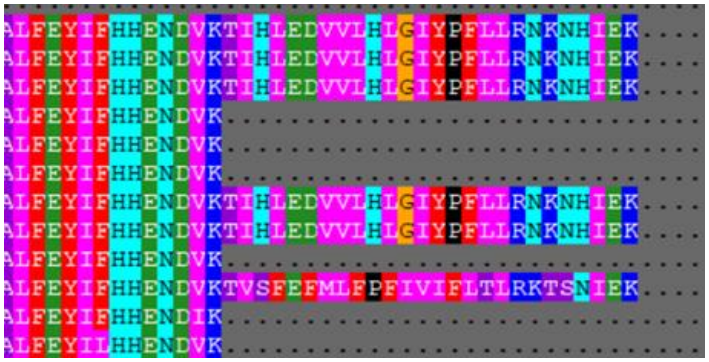


Figure 47 - *Identification d'isoformes. Dans certains cas, il y a une séquence supplémentaire et dans d'autres elle est absente. Il ne s'agit pas d'une insertion ou d'une délétion provoquée par une erreur, mais bien d'une isoforme, car des espèces proches présentent également ce même fragment.*

6.6 Mise en place du *benchmark* G3PO

L'ensemble des séquences "*Confirmed*" et "*Unconfirmed*" ont été utilisés pour construire un *benchmark* multi-espèces appelé G3PO (pour *Gene and Protein Prediction PrOgrams*). Ce dernier, décrit plus précisément dans la partie Contributions et résultats, est basé sur une stratégie stricte d'identification des erreurs afin de caractériser le plus précisément les séquences incluses dans G3PO. Une base de données a été construite afin de répertorier un ensemble de caractéristiques décrivant les séquences de G3PO. Par exemple, on retrouve le nom de l'organisme, l'identifiant Uniprot et Ensembl, le nombre d'erreurs identifiées, le nombre d'exons, la taille moyenne des exons, la présence de régions UTR, la présence de nucléotides indéterminés, etc. Le fichier complet est téléchargeable sur zenodo.org/record/4081640.

7 - Les mesures de performances

Les mesures de performance, ou métriques, sont basées sur une matrice de confusion que nous avons vue dans la section 2.1.9 du Chapitre 2. La métrique la plus connue est probablement l'exactitude (ou précision), cependant on préférera utiliser l'anglicisme *accuracy* pour éviter de confondre avec l'autre métrique nommée précision *(precision)*, qui évalue les performances globales du modèle, qui évalue les performances globales du modèle. Elle se calcule selon la formule :

$$Accuracy = (VP + VN)/(VP + VN + FP + FN), \text{ avec } accuracy \in [0,1]$$

L'*accuracy* calcule le nombre de bonnes prédictions sur le total des prédictions. Elle est souvent exploitée lorsque les données sont équilibrées. Dans le cas d'un jeu déséquilibré, l'*accuracy* n'est pas représentative, car elle peut être favorisée par une classe majoritaire. Par exemple, si le jeu de données contient 9 chats et 1 grimpoteuthis, le modèle peut prédire à chaque fois des chats et aura une *accuracy* de 90%, or il sera très mauvais pour détecter les grimpoteuthis. Dans ce cas, on utilise d'autres métriques tels que la précision, le rappel (ou sensibilité) et la spécificité.

La précision correspond au nombre de bonnes prédictions parmi les prédictions dans la classe positive. Elle est utilisée pour évaluer la rigueur du modèle à bien prédire les éléments positifs et se calcule selon la formule :

$$Précision = \frac{VP}{(VP+FP)}, \text{ avec précision} \in [0,1]$$

Le rappel (*recall*), également appelé sensibilité (*sensitivity*), est la proportion d'éléments positifs correctement classés, parmi tous les éléments correctement classés, on peut considérer qu'il évalue la quantité de bonnes prédictions. La formule est présentée ci-dessous :

$$Rappel = \frac{VP}{(VP+FN)}, \text{ avec rappel} \in [0,1]$$

La spécificité (*specificity*) représente la proportion de vrais négatifs parmi les éléments prédits comme négatifs. Il met en évidence l'impact des faux positifs, en ce sens on qualifie la spécificité comme évaluateur de la qualité de la prédiction. Elle se calcule selon la formule :

$$Spécificité = \frac{VN}{(VN+FP)}, \text{ avec specificité} \in [0,1]$$

Ces métriques sont les principales que l'on retrouve dans de nombreux articles. Cependant il existe d'autres métriques mettant en évidence d'autres éléments. Par exemple, le F1-Score représente la moyenne harmonique du rappel et de la précision, il évalue donc l'équilibre entre ces 2 métriques. Il est tout à fait possible d'avoir un modèle avec une bonne sensibilité, mais une mauvaise précision et donc un mauvais F1-Score. Cela signifie que le modèle a trouvé beaucoup d'éléments, mais qu'il y a beaucoup de FP. Le F1-Score se calcule comme tel :

$$\text{F1-Score} = 2 * \frac{(précision * rappel)}{(précision + rappel)}, \text{ avec F1-Score} \in [0,1]$$

La courbe ROC pour *Receiver Operating Characteristic*, mesure les performances d'un modèle de classifieur binaire en évaluant le taux de vrais positifs (TVP ou *TPR: True Positive Rate*) par rapport aux taux de faux positifs (TFP ou *FPR: False Positive Rate*), en fonction d'un seuil. En faisant varier ce seuil, on obtient des valeurs de TVP et TFP permettant de générer la courbe ROC. Le TVP et TFP se calcule selon la formule suivante :

$$TVP = \frac{VP}{(VP+FN)} \text{ et } TFP = \frac{FP}{(FP+VN)}, \text{ avec TVP et TFP} \in [0,1]$$

Pour évaluer ensuite les performances du modèle à l'aide de la courbe ROC, on calcule l'aire sous la courbe (AUC). Plus l'aire est grande, plus elle s'éloigne de la courbe théorique aléatoire et donc plus le modèle est performant. Une AUC de 0,7 par exemple, signifie que le modèle à 70% de chances de correctement classer les éléments négatifs et positifs, avec une AUC=0,5, on a une chance sur deux, donc totalement aléatoire de prédire correctement la classe. La figure 48 représente des courbes ROC en fonction des performances d'un classifieur.

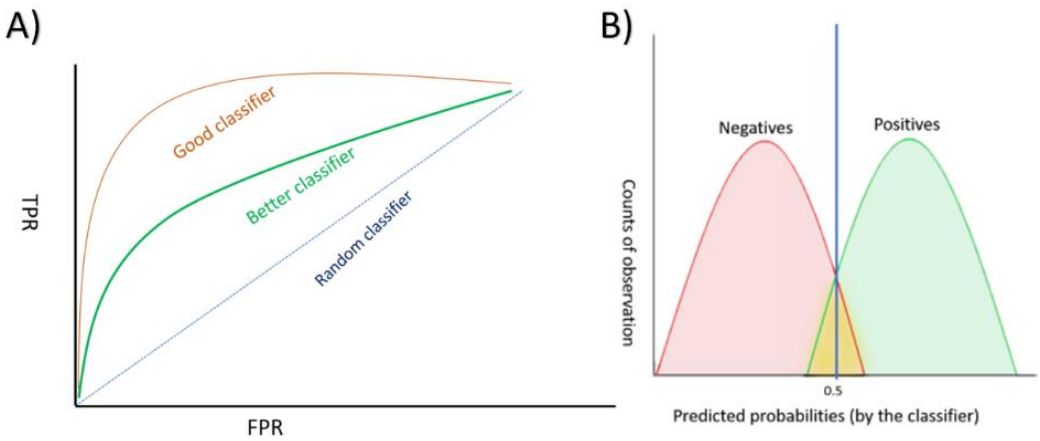


Figure 48 - *A)* *Représentation de différentes courbes ROC. La courbe bleue signifie que le classifieur obtient des résultats totalement aléatoires, car il y a autant de VP que de FP. La courbe verte représente les performances d'un classifieur moyen et la courbe orange représente les performances d'un bon classifieur.* ***B)*** *Représentation des performances du classifieur. En faisant varier le seuil, on obtient différentes valeurs utilisées pour la construction de la courbe ROC (source : ichi.pro/fr/courbe-roc-et-auc-comprehension-detaillee-et-paquet-r-proc-148233331028369).*

CONTRIBUTIONS ET RÉSULTATS

Chapitre 5 - Analyse comparative des programmes *ab initio* de prédiction de gènes codant pour des protéines et construction de G3PO

Des génomes d'organismes eucaryotes et procaryotes sont séquencés quotidiennement provoquant un déluge de données. Ces données volumineuses nécessitent une annotation précise afin de pouvoir les exploiter et extraire de l'information biologique. Dans la majorité des cas, l'annotation est réalisée par l'intermédiaire de programmes de prédiction et non par un expert, car ce travail est extrêmement chronophage et fastidieux. Ces programmes sont principalement basés sur des données expérimentales comme les données de RNA-seq afin de fournir des preuves directes des gènes et isoformes exprimés, ainsi que sur des approches par homologie.

Malheureusement, ces méthodes possèdent certaines limites comme le manque de données expérimentales de l'organisme à annoter ou d'organismes phylogénétiquement proches ou encore des génomes mal assemblés. De plus, les approches par homologie sont susceptibles d'intégrer des données erronées et d'entraîner leur propagation au sein des bases de données (Promponas *et al.*, 2015). Par conséquent, elles sont secondées par des programmes de prédiction *ab initio*.

Les programmes d'annotation *ab initio* ont été exploités bien avant le séquençage du génome humain (Fleischmann *et al.*, 1995), mais c'est à l'aube du XXI^{e} siècle qu'ils sont devenus indispensables, avec l'arrivée du *Big Data*, et aujourd'hui ces mêmes programmes sont encore utilisés (Salzberg, 2019). Bien que leur utilité et leur efficacité ne soient plus à démontrer sur les données humaines et certains organismes modèles, ces programmes souffrent d'une technologie surannée comme les HMM et d'une absence de modèles pour des organismes exotiques. A l'ère des études systémiques et des nombreux enjeux environnementaux et écologiques, il est indispensable de prendre en considération l'étude d'organismes non modèles, à des fins de compréhension des mécanismes biologiques mais, également de préservation des espèces.

Dans ce contexte, j'ai réalisé une analyse comparative de cinq programmes *ab initio* les plus utilisés pour prédire des gènes codant pour des protéines : Augustus, Genscan, GeneID, GlimmerHMM et Snap. L'objectif était d'évaluer leurs performances afin de : i) caractériser

les forces et les faiblesses de chaque programme, ii) de les mutualiser pour extraire des caractéristiques pertinentes qui pourraient être utilisées pour améliorer la prédiction des gènes et iii) de mettre en évidence les difficultés actuelles de l'annotation des génomes. Au travers de différents scénarios, nous avons évalué les performances de ces programmes sur 3 échelles : nucléotidique, structure du gène et protéique.

Pour comparer ces programmes, j'ai élaboré un *benchmark* multi-espèces appelé G3PO, pour *Gene and Protein Prediction PrOgrams*. G3PO inclut deux particularités, la première, quantitative, est liée à l'intégration d'un grand nombre de clades et d'espèces différentes (allant de l'homme aux protistes) et la deuxième, qualitative, est liée à sa haute qualité, car il inclut des données validées manuellement selon des règles très strictes excluant, autant que possible, les erreurs dans le jeu G3PO.

5.1 Diversité phylogénétique de G3PO

Notre objectif était d'analyser les performances des programmes de prédiction *ab initio* sur des données humaines et d'organismes modèles (*D. rerio*, *D. melanogaster*, *C. elegans* ou encore *A. thaliana*) ainsi que sur de nombreux organismes non modèles balayant un large spectre de clades. La totalité des séquences présentes dans G3PO est de 1 793, incluant ainsi une forte hétérogénéité. La figure 49 représente l'arbre phylogénétique des 147 organismes utilisés dans notre étude.

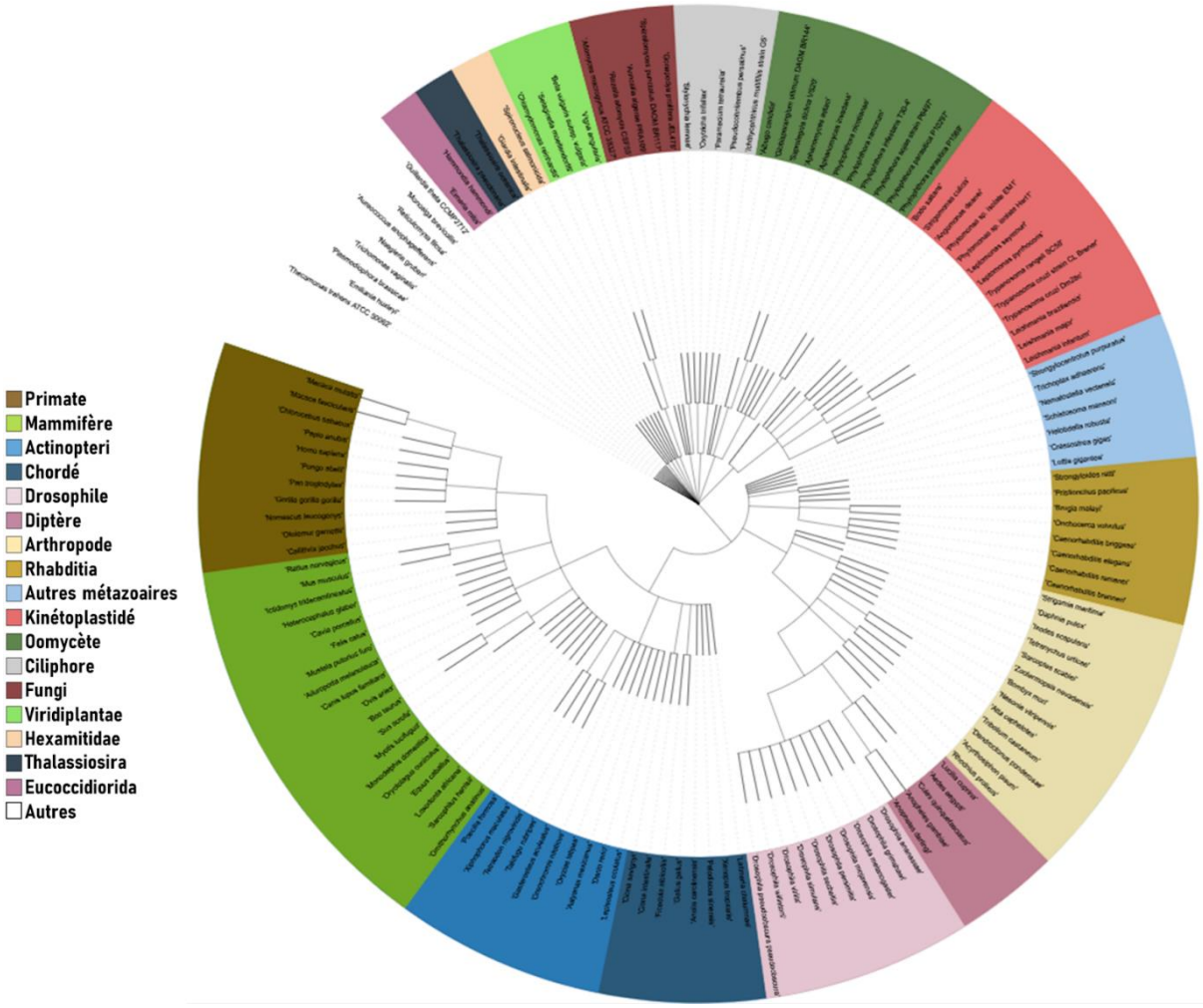


Figure 49 - *Arbre phylogénétique des 147 espèces utilisées pour la construction de G3PO. De nombreux organismes non-modèles sont présents et participent à l'hétérogénéité du benchmark. Ce dernier balaie un large éventail de clades eucaryotes (généré avec iTol (Letunic and Bork, 2021)).*

G3PO inclut 92 espèces métazoaires (69% des séquences totales), 16 stramenopiles (9,6%), 13 euglenozoa (8,3%), 11 alvéolata (5,5%), 8 fungi (1,4%), 4 viridiplantae (0,67%), 2 rhizaria (1,2%), 2 choanoflagellés (1,2%), 1 apusozoa (0,5%), 1 diplomonadida (0,3%), 1 heterolobosea (0,67%), 1 parabasalia (0,73%), 1 cryptophyta (0,67%), 1 haptophyceae (0,33%) et 1 diplomonadida (0,28%). Par exemple, la distribution de la taille moyenne des gènes par organisme est représentative de cette hétérogénéité (figure 50). On retrouve des gènes extrêmement longs (*BBS15* humain avec une taille de 708 285 nucléotides) et d'autres très courts (*BBS18* chez la sangsue *Helobdella robusta* avec une taille de 171 nucléotides).

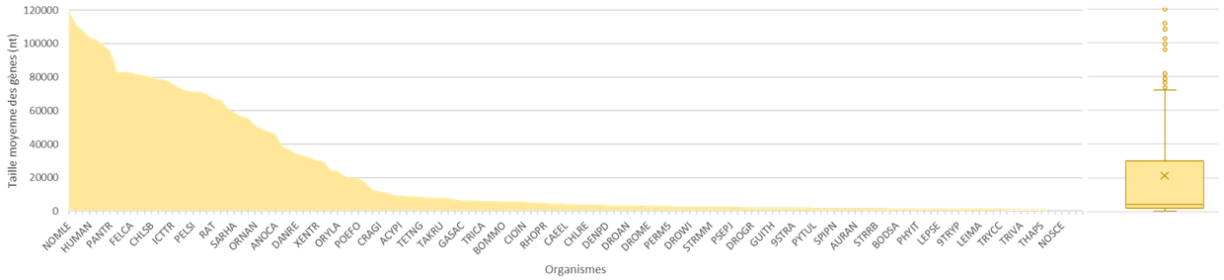


Figure 50 - *Tailles moyennes des gènes de chaque organisme de G3PO. Le gibbon Nomascus leucogenys (NOMLE) est l'organisme avec une taille moyenne des gènes la plus élevée, avec 119 904 nucléotides et la fougère Selaginella moellendorffii (SELML) à une taille moyenne des gènes la plus faible avec 201 nucléotides. La boîte à moustaches représente la distribution des tailles moyennes des gènes pour l'ensemble des organismes de G3PO, la moyenne est proche de 21 200 et la médiane de 4 300.*

Cette diversité au sein de G3PO est une force, car les études comparatives l'utilisant pourront désormais se concentrer sur d'autres organismes non modèles, afin de détecter par exemple de potentielles failles au sein des programmes. Cette hétérogénéité permet également de construire un nombre important de scénarios tels que l'étude de la performance d'un programme sur les primates avec de petits gènes ou sur des arthropodes ayant plus de 5 exons. Cette richesse fait de G3PO une ressource essentielle pour de nombreuses futures études.

5.2 G3PO : spécialiste de la qualité

En plus d'être diversifié, G3PO a également été conçu dans une démarche de qualité. En effet, l'ensemble des données ont été extraites des bases de données UniProt et Ensembl, et comme nous l'avons précisé auparavant, celles-ci peuvent intégrer un certain nombre de séquences erronées. Par conséquent, évaluer des programmes sur ce type de données serait contre-productif. Nous avons donc décidé de vérifier manuellement l'ensemble des séquences protéiques afin de les classer en deux catégories. Toutes les séquences sans erreurs identifiées sont classées comme "*Confirmed*" et toutes les séquences avec au moins une erreur sont classées comme "*Unconfirmed*". Le protocole d'identification des erreurs a été décrit dans la partie 6 du Matériels et méthodes. G3PO est donc composé de 889 séquences "*Confirmed*" et 904 séquences "*Unconfirmed*", soit environ 50% des séquences dans chaque catégorie. D'après la figure 51, le nombre d'organismes du clade métazoaire de bonne qualité, *i.e.* sans erreurs, compose ¾ des données "*Confirmed*". A l'inverse, il y a une plus grande hétérogénéité au sein des séquences erronées, bien que les métazoaires soient également le clade majoritaire.

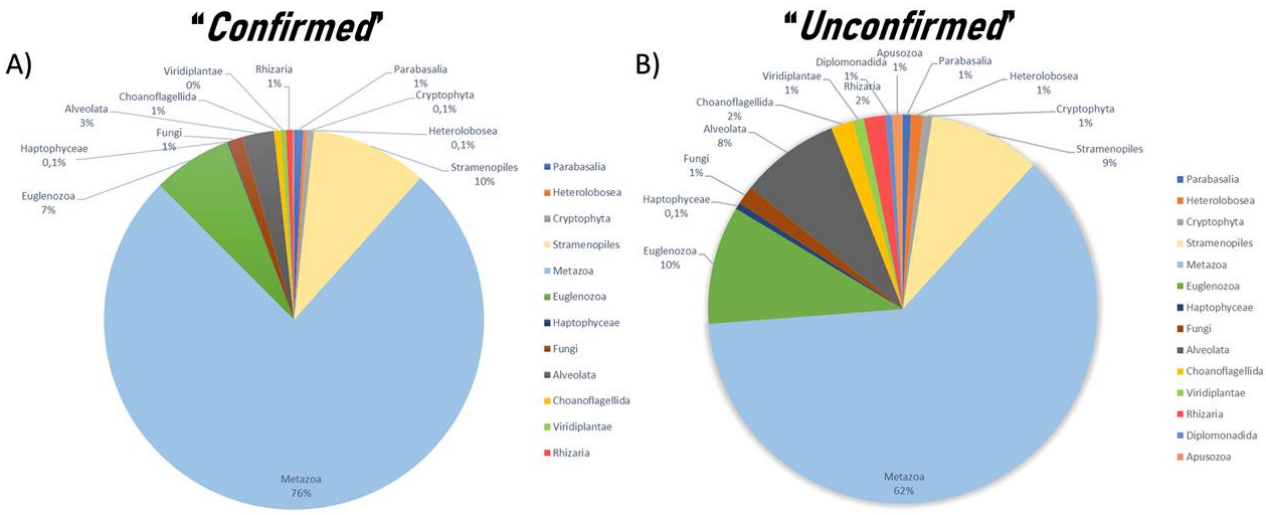


Figure 51 - *Répartition des clades inclus dans G3PO en fonction de la qualité des données.* ***A)*** *la majorité des séquences de bonne qualité sont des métazoaires, straménopiles ou euglenozoa.* ***B)*** *Les séquences incluant des erreurs sont plus hétérogènes, cependant on retrouve les clades métazoaires, straménopiles et euglenozoa qui sont toujours majoritaires, avec l'addition des alvéolata.*

Il est important de noter que si une séquence est considérée comme "*Unconfirmed*", cela ne signifie pas que la séquence entière est erronée, mais uniquement une région de cette séquence. Dans certains cas, il se peut qu'une seule erreur soit identifiée et que le gène soit très long avec beaucoup d'exons, comme c'est le cas pour le gène *EKC33772* de *Crassostrea gigas* qui a une taille de 25 000 nucléotides et est composé de 40 exons (figure 52). En effet, la séquence protéique du gène ne contient qu'une erreur en C-terminal.

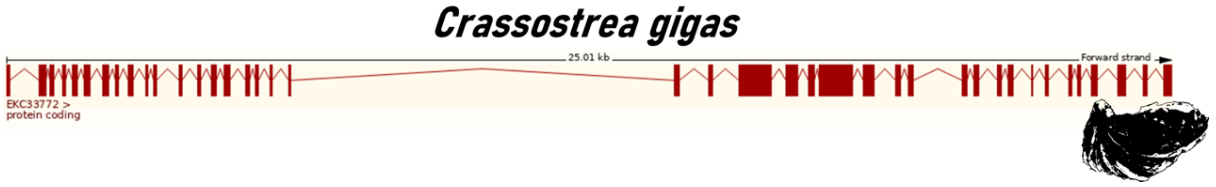


Figure 52 - *Carte génomique de Crassostrea gigas (huître). Le gène EKC33772 est d'une taille de 25 000 nucléotides et se compose de 40 exons et 39 introns, dont un très grand (8 245 nucléotides) au centre du gène.*

Nous avons également caractérisé les clades les plus sensibles à l'erreur. 166 (9,3%) séquences ont au moins 3 erreurs et le nombre maximal d'erreurs identifiées est de 8, pour les séquences de *Giardia intestinalis* (diplomonadida) et de *Aureococcus anophagefferens* (stramenopiles). La figure 53 représente le pourcentage de séquences avec au moins 3 erreurs parmi le nombre total de séquences. 100% des séquences de diplomonadida contiennent au moins 3 erreurs et plus de 50% des séquences des organismes issus du clade apusozoa. Ce sont donc principalement les organismes protistes qui sont très mal prédits. Les métazoaires sont globalement bien prédits, avec seulement 7% des séquences ayant au moins 3 erreurs.

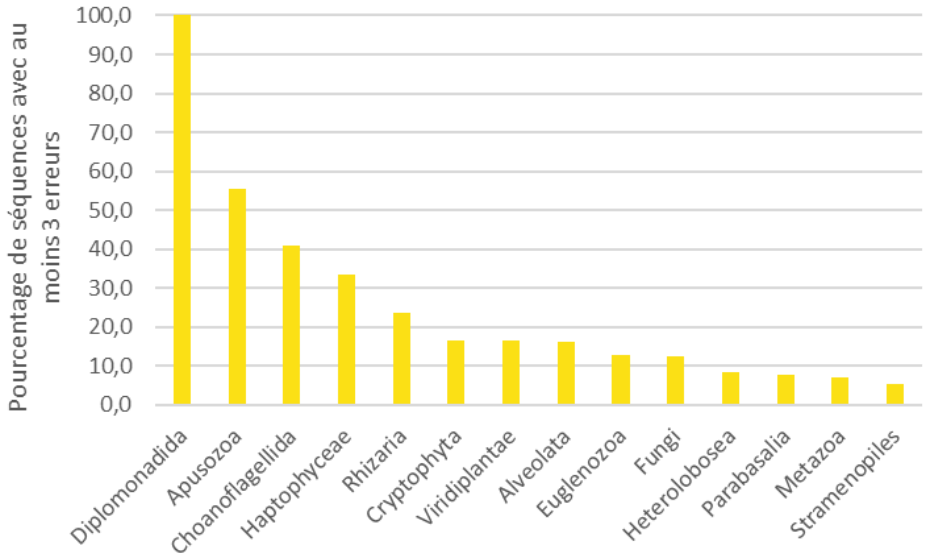


Figure 53 - *Pourcentage de séquences ayant au moins 3 erreurs sur l'ensemble des séquences du clade de G3PO. Les clades les plus sensibles aux erreurs sont ceux des protistes*.

A ce jour, nous n'avons pas identifié de corrélation, dans G3PO, entre le nombre d'erreurs et des caractéristiques génomiques comme la taille du gène, le nombre d'exons ou le pourcentage en GC.

5.3 Applications

Il nous semblait indispensable de développer un *benchmark* hétérogène de haute qualité afin de comparer les performances des programmes de prédictions de gènes *ab initio*. Ainsi, à partir de G3PO nous avons pu évaluer les performances et identifier les forces et faiblesses de chaque programme en construisant différents scénarios, par exemple :

- la présence de régions génomiques avec des nucléotides indéterminés, notés 'N',
- des petites ou grandes protéines,
- un biais en composition génomique, mesuré par le pourcentage de nucléotides GC et bien d'autres encore décrits dans le manuscrit présenté au paragraphe 5.5. G3PO a donc été conçu pour représenter un grand nombre de défis rencontrés lors de la prédiction des gènes codant pour des protéines. Bien qu'en général les programmes de prédiction *ab initio* obtiennent de bonnes performances, il y a encore des cas particuliers qui résistent à une identification efficace.

5.4 De G3PO à G3PO+

Un des avantages de G3PO est qu'il est extensible. De nombreuses autres espèces peuvent y être ajoutées, à condition de respecter le protocole d'identification des erreurs, car la force de G3PO est sa qualité. Ainsi, nous avons procédé à une première mise à jour appelée G3PO+. 948 nouvelles séquences ont été ajoutées (issues de 25 gènes impliqués dans les myopathies) dont 472 "*Confirmed*" et 476 "*Unconfirmed*", soit toujours environ 50% des séquences par catégorie. Ces données ont la particularité d'être issues de 47 organismes du clade des métazoaires uniquement (figure 54). Ainsi, G3PO+ est composé de 2 741 séquences, dont 1 361 "*Confirmed*" et 1 380 "*Unconfirmed*".

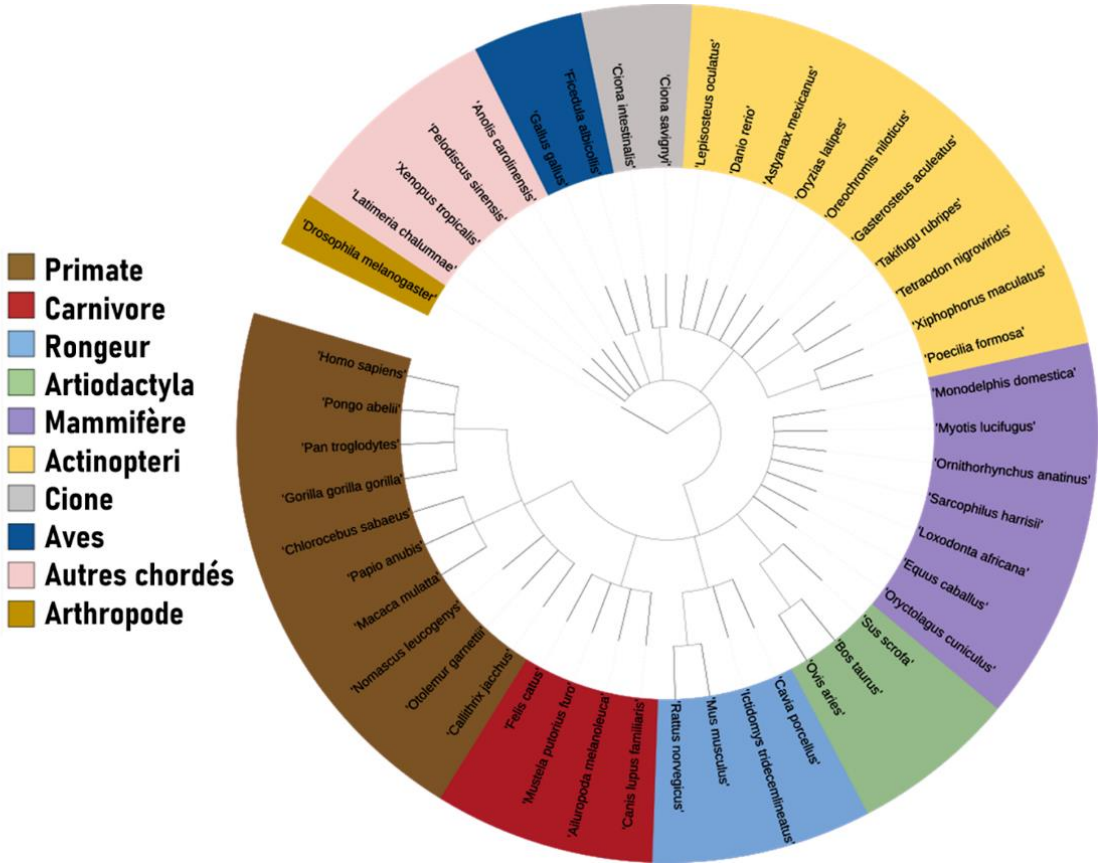


Figure 54 - *Arbre phylogénétique des 47 espèces ajoutées pour la construction de G3PO+. Il ne s'agit que d'organismes métazoaires, allant de l'Homme à la drosophile (généré avec iTol (Letunic and Bork, 2021)).*

L'intérêt de G3PO+ pour mes travaux de thèse est qu'il contient de nombreuses séquences avec un nombre très important d'exons (figure 55A), par exemple le gène *ENSG00000183091* (version 87 d'Ensembl) humain contient 180 exons (figure 55B). Le

nombre moyen d'exons de G3PO+ est de 25,7, par conséquent cela permet de récupérer de nombreuses régions incluant un SE.

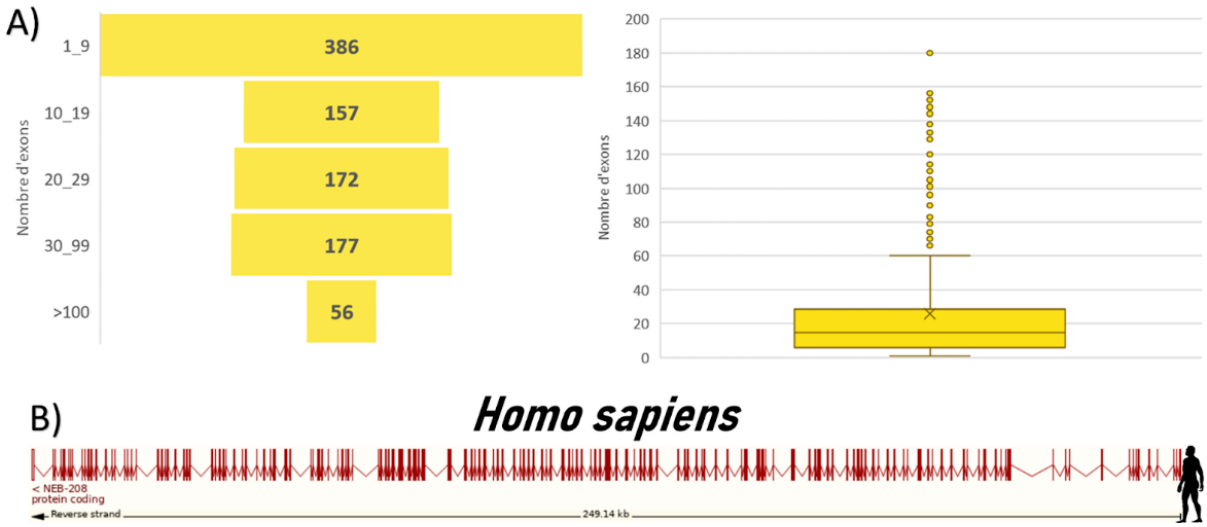


Figure 55 - *Nombre d'exons présent dans les séquences de métazoaires ajoutés dans G3PO+. **A)** Le nombre d'exons est très important dans ces séquences, avec une moyenne de 25,7 exons par gène et plus de 50 séquences ont au moins 100 exons comme **B)** le gène humain ENSG00000183091 qui en possède 180.*

5.5 Publication

L'ensemble des données sont accessibles sur Zenodo : zenodo.org/record/4081640. Les outils et scripts de comparaison sont également accessibles via le git : git.lbgi.fr/scalzitti/Benchmark_study.

Scalzitti et al. BMC Genomics (2020) 21:293
https://doi.org/10.1186/s12864-020-6707-9

BMC Genomics

RESEARCH ARTICLE Open Access

A benchmark study of ab initio gene prediction methods in diverse eukaryotic organisms

Nicolas Scalzitti, Anne Jeannin-Girardon, Pierre Collet, Olivier Poch and Julie D. Thompson*

Abstract

Background: The draft genome assemblies produced by new sequencing technologies present important challenges for automatic gene prediction pipelines, leading to less accurate gene models. New benchmark methods are needed to evaluate the accuracy of gene prediction methods in the face of incomplete genome assemblies, low genome coverage and quality, complex gene structures, or a lack of suitable sequences for evidence-based annotations.

Results: We describe the construction of a new benchmark, called G3PO (benchmark for Gene and Protein Prediction PrOgrams), designed to represent many of the typical challenges faced by current genome annotation projects. The benchmark is based on a carefully validated and curated set of real eukaryotic genes from 147 phylogenetically disperse organisms, and a number of test sets are defined to evaluate the effects of different features, including genome sequence quality, gene structure complexity, protein length, etc. We used the benchmark to perform an independent comparative analysis of the most widely used ab initio gene prediction programs and identified the main strengths and weaknesses of the programs. More importantly, we highlight a number of features that could be exploited in order to improve the accuracy of current prediction tools.

Conclusions: The experiments showed that ab initio gene structure prediction is a very challenging task, which should be further investigated. We believe that the baseline results associated with the complex gene test sets in G3PO provide useful guidelines for future studies.

Keywords: Genome annotation, Gene prediction, Protein prediction, Benchmark study

Background

The plunging costs of DNA sequencing [1] have made de novo genome sequencing widely accessible for an increasingly broad range of study systems with important applications in agriculture, ecology, and biotechnologies amongst others [2]. The major bottleneck is now the high-throughput analysis and exploitation of the resulting sequence data [3]. The first essential step in the analysis process is to identify the functional elements, and in particular the protein-coding genes. However, identifying genes in a newly assembled genome is challenging, especially in eukaryotes where the aim is to establish accurate gene models with precise exon-intron structures of all genes [3–5].

Experimental data from high-throughput expression profiling experiments, such as RNA-seq or direct RNA sequencing technologies, have been applied to complement the genome sequencing and provide direct evidence of expressed genes [6, 7]. In addition, information from closely related genomes can be exploited, in order to transfer known gene models to the target genome. Numerous automated gene prediction methods have been developed

* Correspondence: thompson@unistra.fr
Department of Computer Science, ICube, CNRS, University of Strasbourg, Strasbourg, France

that incorporate similarity information, either from transcriptome data or known gene models, including GenomeScan [8], GeneWise [9], FGENESH [10], Augustus [11], Splign [12], CodingQuarry [13], and LoReAN [14].

The main limitation of similarity-based approaches is in cases where transcriptome sequences or closely related genomes are not available. Furthermore, such approaches encourage the propagation of erroneous annotations across genomes and cannot be used to discover novelty [5]. Therefore, similarity-based approaches are generally combined with ab initio methods that predict protein coding potential based on the target genome alone. Ab initio methods typically use statistical models, such as Support Vector Machines (SVMs) or hidden Markov models (HMMs), to combine two types of sensors: signal and content sensors. Signal sensors exploit specific sites and patterns such as splicing sites, promotor and terminator sequences, polyadenylation signals or branch points. Content sensors exploit the coding versus non-coding sequence features, such as exon or intron lengths or nucleotide composition [15]. Ab initio gene predictors, such as Genscan [16], GlimmerHMM [17], GeneID [18], FGENESH [10], Snap [19], Augustus [20], and GeneMark-ES [21], can thus be used to identify previously unknown genes or genes that have evolved beyond the limits of similarity-based approaches.

Unfortunately, automatic ab initio gene prediction algorithms often make substantial errors and can jeopardize subsequent analyses, including functional annotations, identification of genes involved in important biological process, evolutionary studies, etc. [22–25]. This is especially true in the case of large “draft” genomes, where the researcher is generally faced with an incomplete genome assembly, low coverage, low quality, and high complexity of the gene structures. Typical errors in the resulting gene models include missing exons, non-coding sequence retention in exons, fragmenting genes and merging neighboring genes. Furthermore, the annotation errors are often propagated between species and the more “draft” genomes we produce, the more errors we create and propagate [3–5]. Other important challenges that have attracted interest recently include the prediction of small proteins/peptides coded by short open reading frames (sORFs) [26, 27] or the identification of events such as stop codon recoding [28]. These atypical proteins are often overlooked by the standard gene prediction pipelines, and their annotation requires dedicated methods or manual curation.

The increased complexity of today’s genome annotation process means that it is timely to perform an extensive benchmark study of the main computational methods employed, in order to obtain a more detailed knowledge of their advantages and disadvantages in different situations. Some previous studies have been performed to evaluate the performance of the most widely used ab initio gene predictors. One of the first studies [29] compared 9 programs on a set of 570 vertebrate sequences encoding a single functional protein, and concluded that most of the methods were overly dependent on the original set of sequences used to train the gene models. More recent studies have focused on gene prediction in specific genomes, usually from model or closely-related organisms, such as mammals [30], human [31, 32] or eukaryotic pathogen genomes [33], since they have been widely studied and many gene structures are available that have been validated experimentally. To the best of our knowledge, no recent benchmark study has been performed on complex gene sequences from a wide range of organisms.

Here, we describe the construction of a new benchmark, called G3PO – benchmark for Gene and Protein Prediction PrOgrams, containing a large set of complex eukaryote genes from very diverse organisms (from human to protists). The benchmark consists of 1793 reference genes and their corresponding protein sequences from 147 species and covers a range of gene structures from single exon genes to genes with over 20 exons. A crucial factor in the design of any benchmark is the quality of the data included. Therefore, in order to ensure the quality of the benchmark proteins, we constructed high quality multiple sequence alignments (MSA) and identified the proteins with inconsistent sequence segments that might indicate potential sequence annotation errors. Protein sequences with no identified errors were labeled ‘Confirmed’, while sequences with at least one error were labeled ‘Unconfirmed’. The benchmark thus contains both Confirmed and Unconfirmed proteins (defined in Methods: Benchmark test sets) and represents many of the typical prediction errors presented above. We believe the benchmark allows a realistic evaluation of the currently available gene prediction tools on challenging data sets.

We used the G3PO benchmark to compare the accuracy and efficiency of five widely used ab initio gene prediction programs, namely Genscan, GlimmerHMM, GeneID, Snap and Augustus. Our initial comparison highlighted the difficult nature of the test cases in the G3PO benchmark, since 68% of the exons and 69% of the Confirmed protein sequences were not predicted with 100% accuracy by all five gene prediction programs. Different benchmark tests were then designed in order to identify the main strengths and weaknesses of the different programs, but also to investigate the impact of the genomic environment, the complexity of the gene structure, or the nature of the final protein product on the prediction accuracy.

Results

The presentation of the results is divided into 3 sections, describing (i) the data sets included in the G3PO

benchmark, (ii) the overall prediction quality of the five gene prediction programs tested and (iii) the effects of various factors on gene prediction quality.

Benchmark data sets

The G3PO benchmark contains 1793 proteins from a diverse set of organisms (Additional file 1: Table S1), which can be used for the evaluation of gene prediction programs. The proteins were extracted from the Uniprot [34] database, and are divided into 20 orthologous families (called BBS1–21, excluding BBS14) that are representative of complex proteins, with multiple functional domains, repeats and low complexity regions (Additional file 1: Table S2). The benchmark test sets cover many typical gene prediction tasks, with different gene lengths, protein lengths and levels of complexity in terms of number of exons (Additional file 1: Fig. S1). For each of the 1793 proteins, we identified the corresponding genomic sequence and the exon map in the Ensembl [35] database. We also extracted the same genomic sequences with additional DNA regions ranging from 150 to 10,000 nucleotides upstream and downstream of the gene, in order to represent more realistic genome annotation tasks. Additional file 1: Fig. S2 shows the distribution of various features of the 1793 benchmark test cases, at the genome level (gene length, GC content), gene structure level (number and length of exons, intron length), and protein level (length of main protein product).

Phylogenetic distribution of benchmark sequences

The protein sequences used in the construction of the G3PO benchmark were identified in 147 phylogenetically diverse eukaryotic organisms, ranging from human

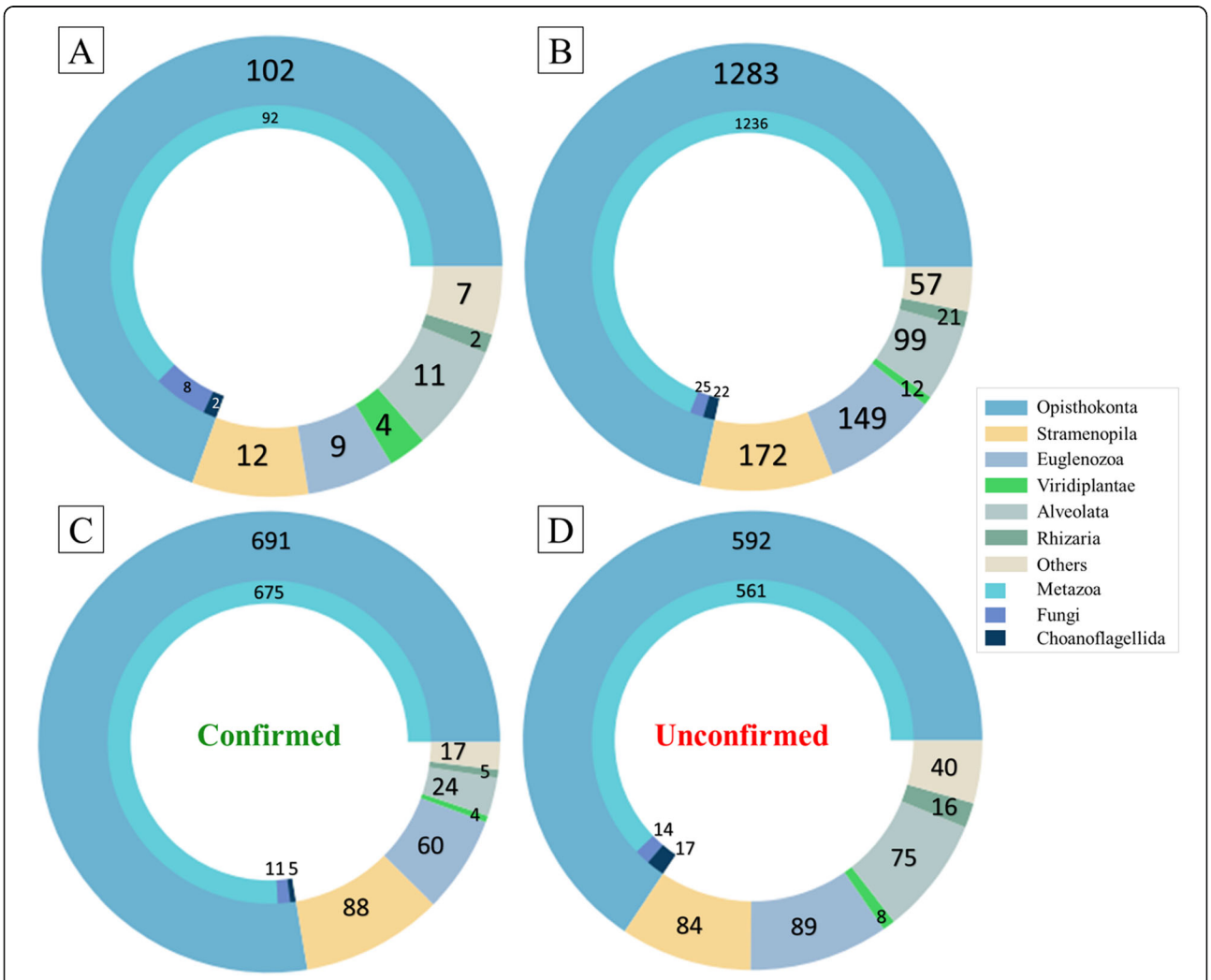


Fig. 1 Phylogenetic distribution of the 1793 test cases in the G3PO benchmark. **a** Number of species in each clade. **b** Number of sequences in each clade. **c** Number of sequences in each clade in the Confirmed test set. **d** Number of sequences in each clade in the Unconfirmed test set. The 'Others' group corresponds to: Apusozoa, Cryptophyta, Diplomonadida, Haptophyceae, Heterolobosea, Parabasalia

to protists (Fig. 1a and Additional file 1: Table S3). The majority (72%) of the proteins are from the Opisthokonta clade, which includes 1236 (96.4%) Metazoa, 25 (1.9%) Fungi and 22 (1.7%) Choanoflagellida sequences (Fig. 1b). The next largest groups represented in the database are the Stramenopila (172), Euglenozoa (149) and Alveolata (99) sequences. More divergent species are included in the 'Others' group, containing 57 sequences from 6 different clades, namely Apusozoa, Cryptophyta, Diplomonadida, Haptophyceae, Heterolobosea and Parabasalia.

Exon map complexity

The benchmark was designed to cover a wide range of test cases with different exon map complexities, as encountered in a realistic complete genome annotation project. The test cases in the benchmark range from single exon genes to genes with 40 exons (Additional file 1: Fig. S2). In particular, the different species included in the benchmark present different challenges for gene prediction programs. To illustrate this point, we compared the number of exons in the human genes to the number of exons in the orthologous genes from each species (Fig. 2). Three main groups can be distinguished: i) Chordata, ii) other Opisthokonta (Mollusca, Platyhelminthes, Panarthropoda, Nematoda, Cnidaria, Fungi and Choanoflagellida) and iii) other Eukaryota (Amoebozoa, Euglenozoa, Heterolobosza, Parabasalia, Rhodophyta, Viridiplantae, Stramenopila, Alveolata, Rhizaria, Cryptophyta, Haptophyceae). As might be expected, the sequences in the Chordata group generally have a similar number of exons compared to the Human sequences. The sequences in the 'other Opisthokonta' group have greater heterogeneity, as expected due to their phylogenetic divergence, although some classes, such as the insects are more homogeneous. The genes in this group have three times fewer exons on average, compared to the Chordata group. The 'other Eukaryota' group includes diverse clades ranging from Viridiplantae and Protists, although the exon map complexity is relatively homogeneous within each clade. For example, in the Euglenozoa clades, all sequences have less than 20% of the number of exons compared to human.

Quality of protein sequences

The protein sequences included in the benchmark were extracted from the public databases, and it has been shown previously that these resources contain many sequence errors [22–25]. Therefore, we evaluated the quality of the protein sequences in G3PO using a homology-based approach (see Methods), similar to that used in the GeneValidator program [23]. We thus identified protein sequences containing potential errors, such as inconsistent insertions/deletions or mismatched sequence segments (Additional file 1: Fig. S3 and Methods). Of the 1793 proteins, 889 (49.58%) protein sequences had no identified errors and were classified as 'Confirmed', while 904 (50.42%) protein sequences had from 1 to 8 potential errors (Fig. 3a) and were classified as 'Unconfirmed'. The 904 Unconfirmed sequences contain a total of 1641 errors, i.e. each sequence has an average of 1.8 errors. Additional file 1: Table S4 shows the number of Unconfirmed sequences and the total number of errors identified for each species included in the benchmark.

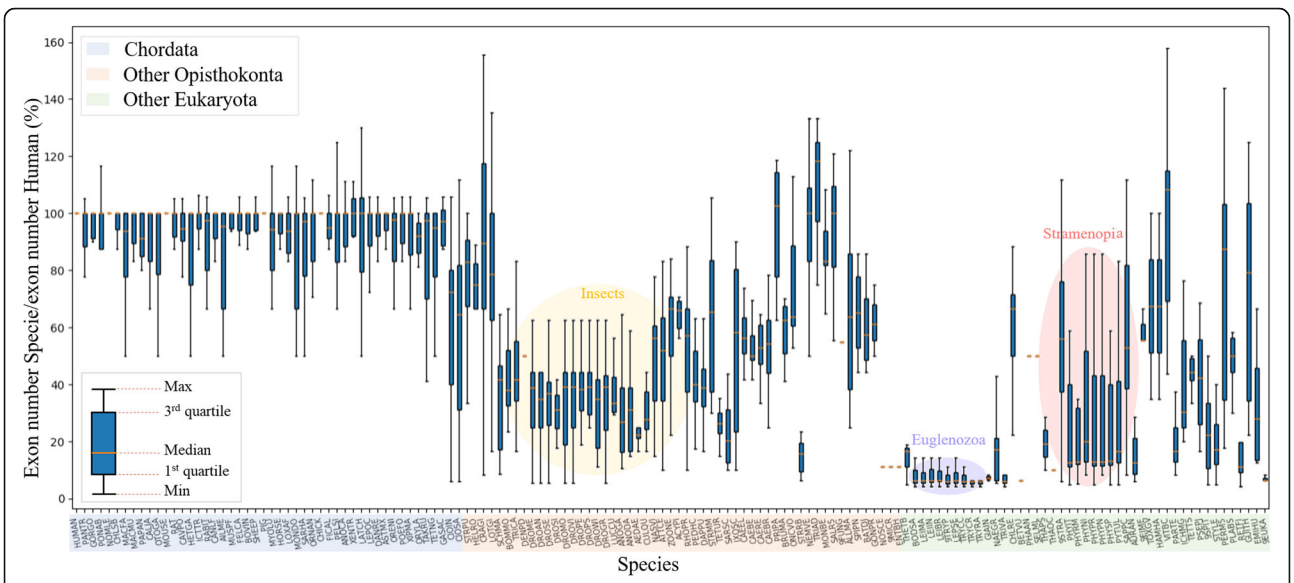


Fig. 2 Exon map complexity for each species. Each box plot represents the distribution of the ratio of the number of exons in the gene of a given species (Exon Number Species), to the number of exons in the orthologous human gene (Exon number Human), for all genes in the benchmark. Notable clades include Insects (BOMMO to PEDHC), Euglenozoa (BODSA to TRYRA) or Stramenopila (THAPS to AURAN)

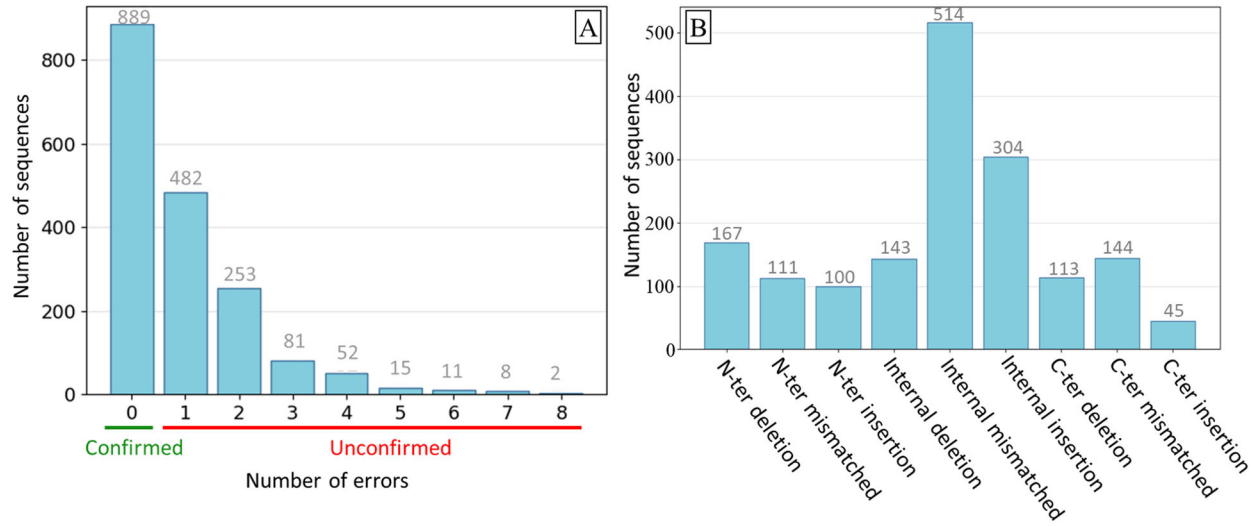


Fig. 3 **a** Number of identified sequence errors in the 1793 benchmark proteins. **b** Number of 'Unconfirmed' protein sequences for each error category

We further characterized the Unconfirmed sequences by the categories of error they contain (Fig. 3b) and by orthologous protein family (Additional file 1: Fig. S4A and B). All the protein families contain Unconfirmed sequences, regardless of the number or length of the sequences, although the ratio of Confirmed to Unconfirmed sequences is not the same in all families. For example, the BBS6, 11, 12, 18 families, that are present mainly in vertebrate species, have more Confirmed sequences (68.5, 80.0, 52.3, 61.1% respectively). Inversely, the majority of sequences in the BBS8 and 9 families, that contain many phylogenetically disperse organisms, are Unconfirmed (68.8, 73.3% respectively). The majority of the 1641 errors (58.4%) are internal (i.e. do not affect the N- or C-termini) and 31% are internal mismatched segments, while N-terminal errors (378 = 23.0%) are more frequent than C-terminal errors (302 = 18.4%). At the N- and C-termini, deletions are more frequent than insertions (280 and 145, respectively), in contrast to the internal errors, where insertions are more frequent (304 compared to 143).

The distributions of various features are compared for the sets of 889 Confirmed and 904 Unconfirmed sequences in Additional file 1: Fig. S2. There are no significant differences in gene length (p-value = 0.735), GC content (p-value = 0.790), number of exons (p-value = 0.073), and exon/intron lengths (p-value = 0.690 / p-value = 0.949) between the Confirmed and Unconfirmed sequences. The biggest difference is observed at the protein level, where the Confirmed protein sequences are 13% shorter than the Unconfirmed proteins (p-value = 8.75×10^{-9}). We also compared the phylogenetic distributions observed in the Confirmed and Unconfirmed sequence sets (Fig. 1c and d). Two clades had a higher proportion of Confirmed sequences, namely Opisthokonta (691/1283 = 54%) and Stramenopila (88/172 = 51%). In contrast, Alveolata (24/99 = 24%), Rhizaria (5/21 = 24%) and Choanoflagellida (5/22 = 22%) had fewer Confirmed than Unconfirmed sequences.

Quality of genome sequences

The genomic sequences corresponding to the reference proteins in G3PO were extracted from the Ensembl database. In all cases, the soft mask option was used (see Methods) to localize repeated or low complexity regions. However, some sequences still contained undetermined nucleotides, represented by 'n' characters, probably due to genome sequencing errors or gaps in the assembly. Undetermined (UDT) nucleotides were found in 283 (15.8%) genomic sequences from 58 (39.5%) organisms, of which 281 sequences (56 organisms) were from the metazoan clade (Additional file 1: Fig. S5). Of these 283 sequences, 133 were classified as Confirmed and 150 were classified as Unconfirmed.

We observed important differences between the characteristics of the sequences with UDT regions and the other G3PO sequences, for both Confirmed and Unconfirmed proteins (Additional file 1: Table S5). The average length of the 283 gene sequences with UDT regions (95,584 nucleotides) is 6 times longer than the average length of the 1510 genes without UDT (15,934 nucleotides), although the protein sequences have similar average lengths (551 amino acids for UDT sequences compared to 514 amino acids for non UDT sequences). Sequences with UDT regions have twice as many exons,

three times shorter exons and five times longer introns than sequences without UDT.

Evaluation metrics

The benchmark includes a number of different performance metrics that are designed to measure the quality of the gene prediction programs at different levels. At the nucleotide level, we study the ability of the programs to correctly classify individual nucleotides found within exons or introns. At the exon level, we applied a strict definition of correctly predicted exons: the boundaries of the predicted exons should exactly match the boundaries of the benchmark exons. At the protein level, we compare the predicted protein to the benchmark sequence and calculate the percent sequence identity (defined as the number of identical amino acids compared to the number of amino acids in the benchmark sequence). It should be noted that, due to their strict definition, scores at the exon level are generally lower. For example, in some cases, the predicted exon boundary may be shifted by a few nucleotides, resulting in a low exon score but high nucleotide and protein level scores.

Evaluation of gene prediction programs

We selected five widely used gene prediction programs: Augustus, Genscan, GeneID, GlimmerHMM and Snap. These programs all use Hidden Markov Models (HMMs) trained on different sets of known protein sequences and take into account different context sensors, as summarized in Table 1. Each prediction program was run with the default settings, except for the species model to be used. As the benchmark contains sequences from a wide range of species, we selected the most pertinent training model for each sequence, based on their taxonomic proximity (see Methods). The genomic sequences for the 1793 test cases in the G3PO benchmark were used as input to the selected gene prediction programs and a series of tests were performed (outlined in Fig. 4), in order to identify the strong and weak points of the different algorithms, as well as to highlight specific factors affecting prediction accuracy.

Gene prediction accuracy

In order to estimate the overall accuracy of the five gene prediction programs, the genes predicted by the programs were compared to the benchmark sequences in G3PO. At this stage, we included only the 889 Confirmed proteins, and used the genomic sequences corresponding to the gene region with 150 bp flanking sequence upstream and downstream of the gene (Fig. 4 – Initial tests) as input. Figure 5(a-c) and Additional file 1: Table S6 show the mean quality scores at different levels: nucleotide, exon structure and final protein sequence (defined in Methods).

At the nucleotide level (Fig. 5a), most of the programs have higher specificities than sensitivities (with the exception of GlimmerHMM), meaning that they tend to underpredict. F1 scores range from 0.39 for Snap to 0.52 for Augustus, meaning that it has the best accuracy.

At the exon level (Fig. 5b left), Augustus and Genscan achieve higher sensitivities (0.27, 0.23 respectively) and specificities (0.30, 0.28 respectively) than the other programs. Nevertheless, the number of mis-predicted exons remains high with 65 and 74% Missing Exons and 62 and 69% Wrong Exons respectively for Augustus and Genscan. At this level, GeneID and Snap have the lowest sensitivity and specificity, indicating that the predicted splice boundaries are not accurate. We also investigated whether the

Table 1 Main characteristics of the gene prediction programs evaluated in this study. GHMM: Generalized hidden Markov model; UTR: Untranslated regions

Gene predictor	Signal sensors	Content sensors	Algorithm model	Organism-specific models
Genscan (version 1.0)	Promoter (15 bp), cap site (8 bp), TATA to cap site distance of 30 to 36 bp, donor (– 3 to + 6 bp)/ acceptor (– 20 to + 3) splice sites, polyadenylation, translation start/stop sites	Intergenic, 5′–/3′-UTR, exon/introns in 3 phases, forward/reverse strands	3-periodic fifth-order Markov model (GHMM)	3 models
GlimmerHMM (version 3.02)	Donor (16 bp)/ acceptor (29 bp) splice sites, start/ stop codons	Exon/intron in one frame,intron length 50–1500 bp, total coding length > 200 bp	Hidden Markov model (GHMM)	5 models
GeneID (version 1.4)	Donor/acceptor splice sites (– 3 to + 6 bp), start/stop codons	First/initial/last exon, single-exon gene, intron, intron length > 40 bp, intergenic distance > 300 bp	Fifth-order Markov model (HMM)	66 models
SNAP (version 2006-07-28)	Donor (– 3 to + 6 bp) /acceptor (– 24 to + 3) splice sites, translation start (– 6 to + 6 bp) /stop (– 6 to + 3 bp) sites	intergenic, single-exon gene, first/initial/last exon, introns in 3 phases	Fourth-order Markov model (GHMM)	11 models
Augustus (version 3.3.2)	Donor (– 3 to + 6 bp) /acceptor (– 5 to + 1 bp) splice sites, branch point (32 bp), translation start (– 20 to + 3)/stop (3 bp) sites	intergenic, single exon gene, first/initial/last exon, short/long introns in 3 phases and forward/reverse strands, isochore boundaries	Fourth-order Interpolated Markov model (GHMM)	109 models

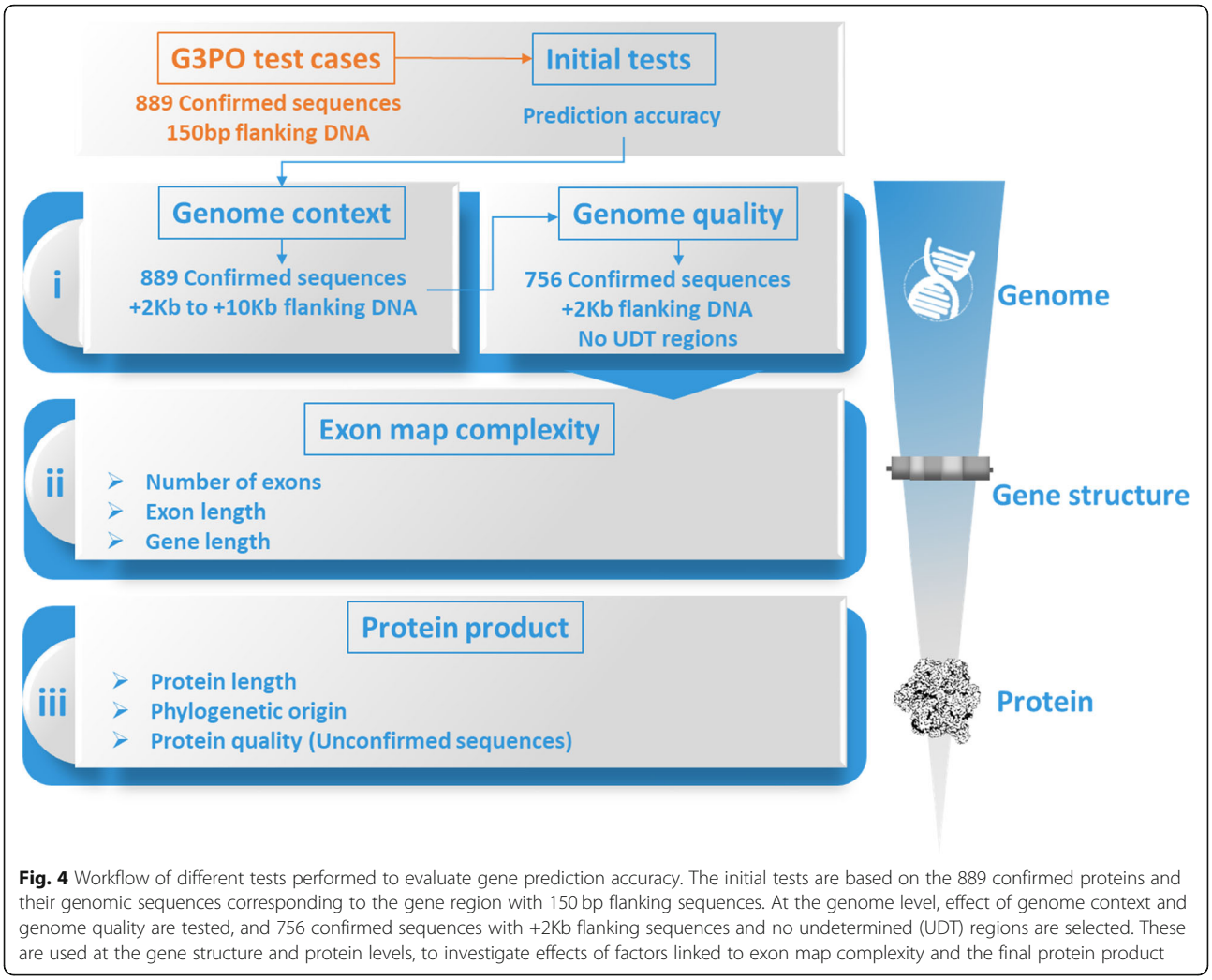


Fig. 4 Workflow of different tests performed to evaluate gene prediction accuracy. The initial tests are based on the 889 confirmed proteins and their genomic sequences corresponding to the gene region with 150 bp flanking sequences. At the genome level, effect of genome context and genome quality are tested, and 756 confirmed sequences with +2Kb flanking sequences and no undetermined (UDT) regions are selected. These are used at the gene structure and protein levels, to investigate effects of factors linked to exon map complexity and the final protein product

exon position had an effect on prediction accuracy, by comparing the percentage of well predicted first and last exons with the percentage of well predicted internal exons (Fig. 5b right). The internal exons are predicted better than the first and last exons. In addition, for all exons, the 3′ boundary is generally predicted better than the 5′ boundary. To further investigate the complementarity of the different programs, we plotted the number of Correct Exons (i.e. both 5′ and 3′ exon boundaries correctly predicted) identified by at least one of the programs (Fig. 6a). A total of 167 exons were found by all five programs, suggesting that they are relatively simple to identify. More importantly, 689 exons were correctly predicted by only one program, while 5461 (68.4%) exons were not predicted correctly by any of the programs.

As might be expected, the nucleotide and exon scores are reflected at the protein level (Fig. 5c), with Augustus again achieving the best score, obtaining 75% sequence identity overall and predicting 209 of the 889 (23.5%) Confirmed proteins with 100% accuracy. GeneID and Snap have the lowest scores in terms of perfect protein predictions (52.6, 46.6% respectively). Again, we investigated the complementarity of the programs, by plotting the number of proteins that were perfectly predicted (100% identity) by at least one of the programs (Fig. 6b). Only 32 proteins are perfectly predicted by all five programs, while 108 proteins were predicted with 100% accuracy by a single program. These were mostly predicted by Augustus (61), followed by GlimmerHMM (17). 611 (69%) of the 889 benchmark proteins were not predicted perfectly by any of the programs included in this study.

Computational runtime

We also compared the CPU time required for each program to process the benchmark sequences (Additional file 1: Table S7). Using the gene sequences with 150 bp flanking regions (representing a total length of 51,699, 512 nucleotides), Augustus required the largest CPU time (1826 s), taking > 3.4 times as long as the second slowest program, namely GlimmerHMM (540 s). GeneID

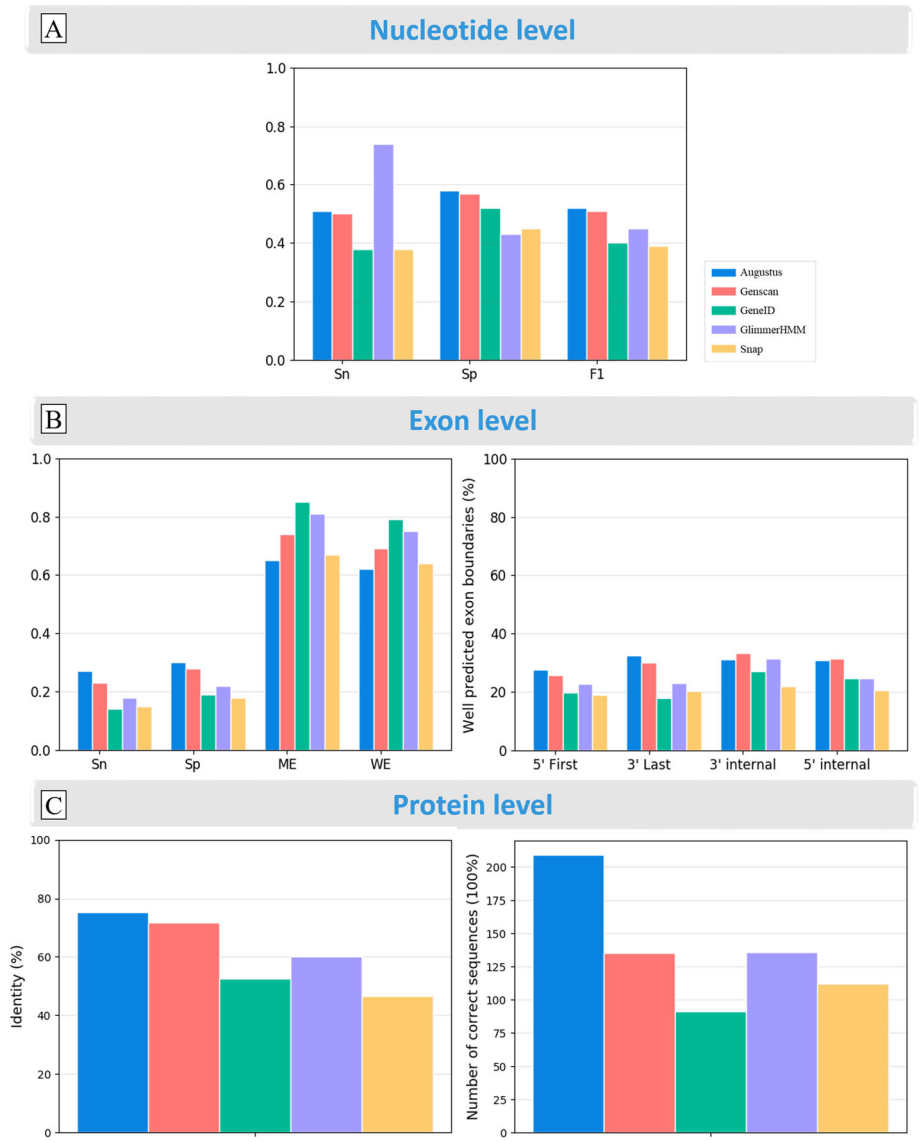


Fig. 5 Overall performance of the five gene prediction programs, using the 889 Confirmed sequences with 150 bp flanking sequences, at the **a** nucleotide, **b** exon and **c** protein levels. Sn = sensitivity; Sp = specificity; F1 = F1 score; ME = Missing Exon; WE = Wrong Exon; 5′ First = percentage of correctly predicted 5′ boundaries of first exons only; 3′ Last = percentage of correctly predicted 3′ boundaries of last exons; 3′ and 5′ internal are the percentage of correctly predicted 3′ and 5′ internal exon boundaries. %Identity indicates the average sequence identity observed between the predicted proteins and the Confirmed benchmark sequences

was the fastest program and completed the gene prediction for the 1793 genomic regions, including 10Kb upstream/downstream flanking nucleotides (total length of 86,970,612 nucleotides), in 260 s.

Analysis of factors affecting gene prediction quality

Based on the results of our initial comparison of gene prediction accuracy, and particularly the complementarity of the programs highlighted in Fig. 6, we decided to investigate further the different factors that may influence the performance of the prediction programs. Figure 4 provides an overview of the different tests performed, including: i) factors associated with the input genomic sequence, ii) factors associated with the gene structure, and iii) factors associated with the protein product.

Factors associated with the input genomic sequence

We first evaluated the genome context and the effect of adding flanking sequences upstream and downstream of

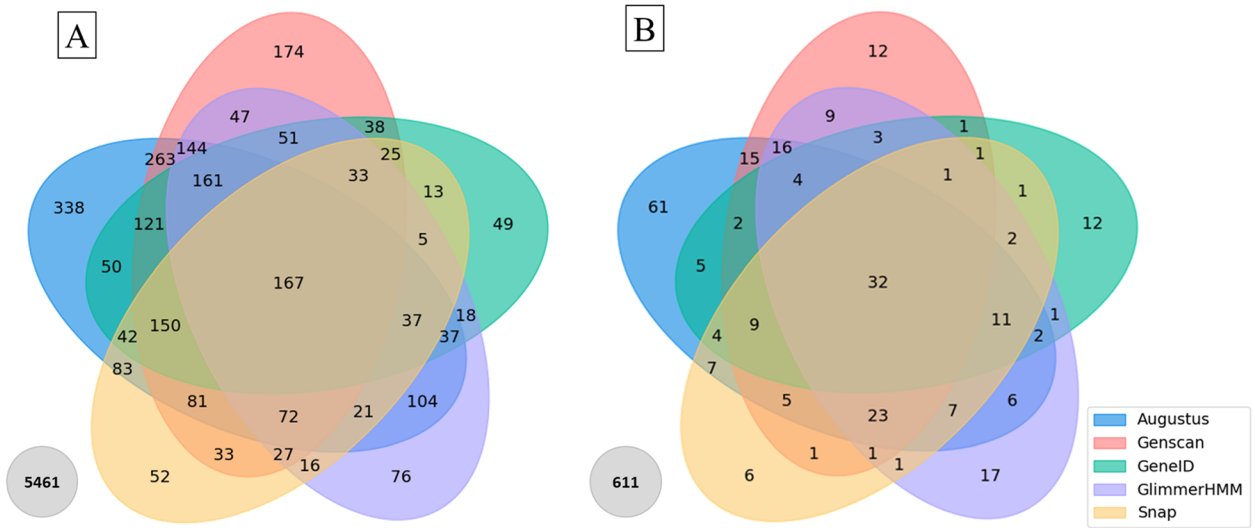


Fig. 6 Venn diagrams representing **a** the number of correct exons predicted by each program, and **b** the number of perfectly predicted proteins by each program. The grey circles indicate the number of exons/proteins badly predicted by all programs

the benchmark gene sequence used as input to the prediction programs, using the 889 Confirmed benchmark tests. We added different flanking sequence lengths ranging from 150 bp to 10Kb, and calculated the same quality scores as above, at the nucleotide, exon and protein levels (Fig. 7 and Additional file 1: Table S8).

At the nucleotide level, the sensitivity of Augustus, Genscan, GeneID and Snap is not significantly affected by the addition of the flanking sequences. For GlimmerHMM (p-value = 4.8×10^{-20}), a significant increase in sensitivity is observed when 2Kb flanking sequences are added, compared to the gene sequences with 150 bp only. In terms of specificity, the addition of 2Kb flanking sequences increases significantly the quality of all the programs (Augustus: p-value = 2.87×10^{-7}, Genscan: p-value = 1.27×10^{-9}, GeneID: p-value = 8.46×10^{-5}, GlimmerHMM: p-value = 2.78×10^{-7}, Snap: p-value = 1.03×10^{-17}). This is probably due to the addition of specific signals in the genomic environment of the gene (further than 150 bp from the gene boundaries), such as the promoter, enhancers/silencers, etc. that are taken into account in the program prediction models. At the exon level, the effect of the flanking sequences is not the same for the different programs. For example, the sensitivity of Augustus (p-value = 4.06×10^{-4}), Genscan (p-value = 1.59×10^{-8}) and GeneID (p-value = 2.98×10^{-2}) is highest when the input sequence has 150 bp flanking regions and significantly decreases when 2Kb flanking nucleotides are added, while for GlimmerHMM (p-value = 0.54) and Snap (p-value = 0.62) no significant difference is observed. Similar results are observed in terms of specificity. At the protein level, for all five programs, the sequence identity compared to the benchmark protein sequence decreases as the length of the flanking sequences increases.

For Augustus, Genscan and GeneID, the addition of the flanking sequences also reduces the number of proteins perfectly predicted (100% identity). This is especially true for Genscan, where we observe a loss of more than 24% of perfectly predicted proteins between 150 bp and 2Kb. On the other hand, for GlimmerHMM and Snap, the number of perfectly predicted proteins increases, especially when 2-4Kb flanking DNA is provided.

Since the greatest effect of adding upstream/downstream flanking sequences was generally observed for a length of 2Kb, the remaining analyses described in this work are all based on the gene sequences with 2Kb upstream/downstream flanking regions.

Next, we studied the relative robustness of the programs to the presence of UDT regions in the genomic sequences, generally due to genome sequencing errors or assembly gaps. This test was limited to the Confirmed sequences from the metazoan clade, since the sequences with UDT regions were almost exclusively found in this clade. Of the 675 metazoan sequences, 133 were found to have UDT regions. We therefore compared the 542 Confirmed sequences without UDT (−UDT) regions with the 133 Confirmed sequences with UDT regions (+UDT). Figure 8 and Additional file 1: Table S9 show the average scores obtained for these two sequence sets, at the nucleotide, exon and protein levels. As might be expected, a reduction in sensitivity and specificity was observed at the nucleotide and exon levels for almost all programs (except exon level specificity and 5′/3′

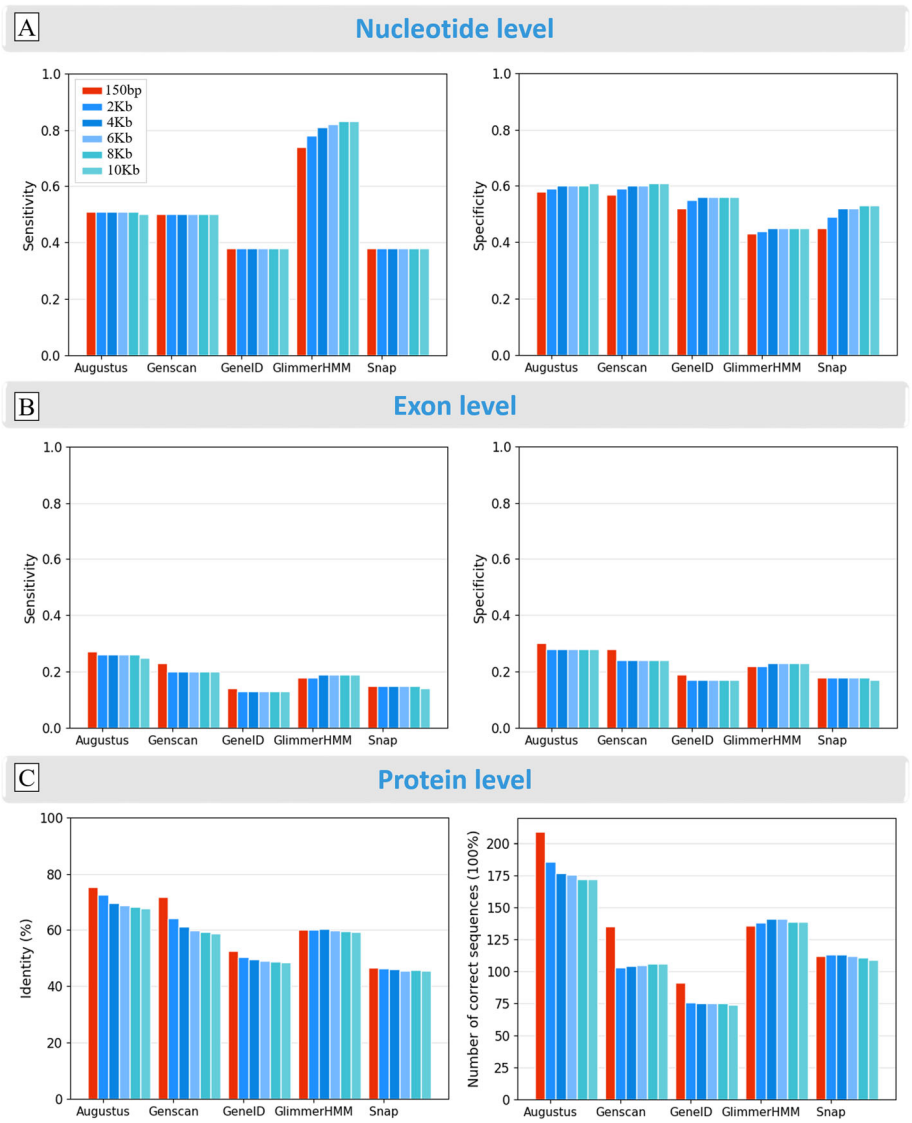


Fig. 7 Effect of the genomic context based on the different lengths of upstream/downstream flanking genomic sequences on the performance of the five gene prediction programs. **a** sensitivity and specificity of prediction of coding nucleotides. **b** sensitivity and specificity of exon prediction. **c** accuracy of protein sequence prediction (% identity) and number of proteins correctly predicted with 100% identity

internal exon boundaries of Augustus) for the +UDT sequences, and at the protein level, very few +UDT proteins are predicted with 100% accuracy. Overall, Augustus and Genscan perform better, although GlimmerHMM predicts the highest number of proteins with 100% accuracy for the +UDT sequences.

Since the UDT regions affected the programs to different extents, the analyses described in the following sections are all based on the set of 756 Confirmed sequences that have no UDT regions.

Finally, we investigated how the GC content of the genes influences the gene finders (Additional file 1: Fig. S6). As might be expected, genes with high GC content are predicted better than genes with high AT content. The GC content of the genome is more difficult to test independently of the other factors, but could contribute to the species-dependent differences observed, shown in Fig. 12.

Factors associated with the gene structure

We first evaluated the effect of the Exon Map Complexity (EMC), represented by the number of exons in the Confirmed benchmark tests (Additional file 1: Fig. S7). Figure 9 shows the quality scores at the exon and protein levels, for sequences with the number of exons ranging from 1 to 20. Overall, we observed a tendency for the five programs to achieve better sensitivity and specificity for the genes with more exons. This may be

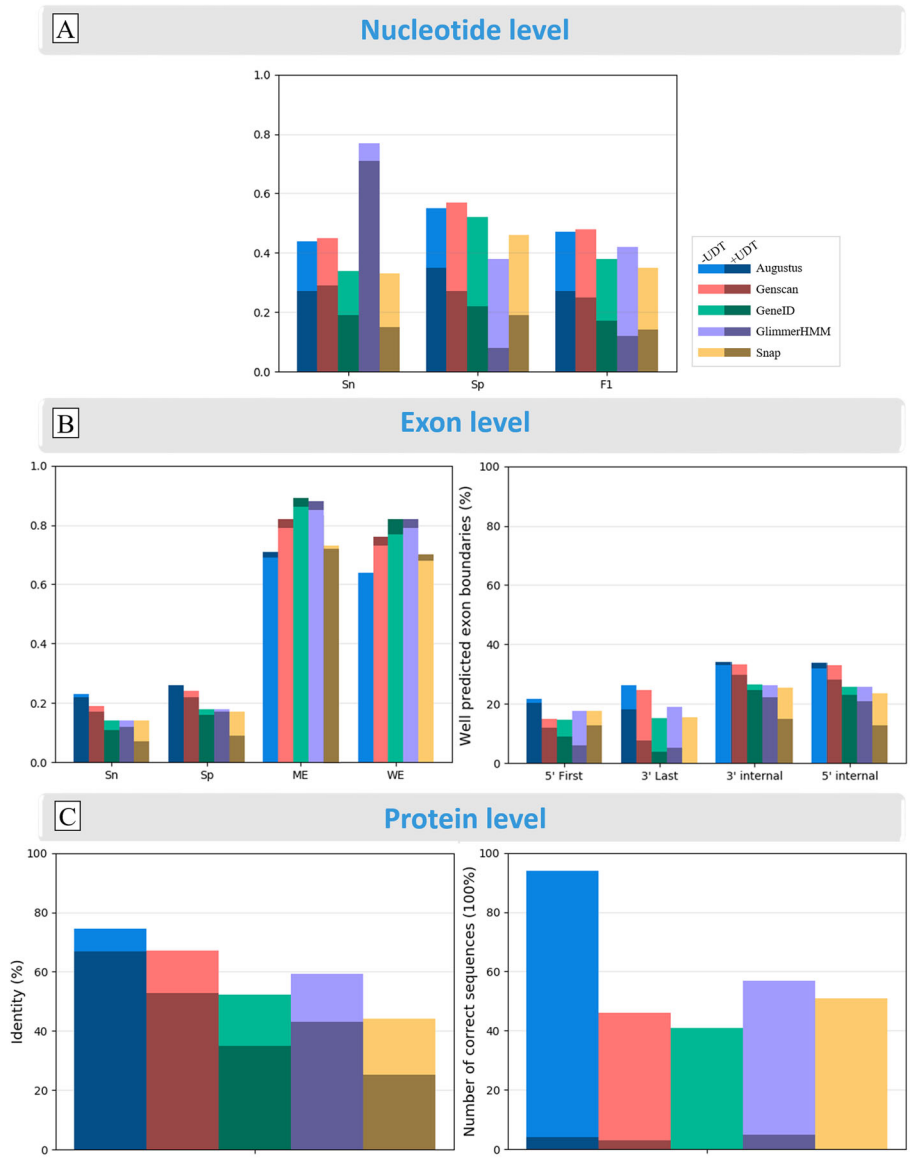


Fig. 8 Effect of undetermined sequence regions (UDT) on prediction performance of the five gene prediction programs, using Confirmed benchmark sequences from Metazoa, where 542 sequences have no undetermined regions (–UDT: light colors) and 133 sequences have undetermined regions (+UDT: dark colors). **a** sensitivity and specificity of nucleotide prediction. **b** sensitivity and specificity of exon prediction **c** accuracy of protein sequence prediction (% identity) and number of proteins correctly predicted with 100% identity. Sn = sensitivity; Sp = specificity; F1 = F1 score; ME = Missing Exons; WE = Wrong Exons; 5′ First = percentage of correctly predicted 5′ boundaries of first exons only; 3′ Last = percentage of correctly predicted 3′ boundaries of last exons; 3′ and 5′ internal are the percentage of correctly predicted 3′ and 5′ internal exon boundaries. %Identity indicates the sequence identity observed between the predicted proteins and the Confirmed benchmark sequences

because most of these more complex sequences are from well-studied vertebrate genomes. For very complex exon maps (≥20 exons), all the programs seem to perform less well, although this may be an artifact due to the small number of these sequences in the benchmark (Additional file 1: Fig. S7A). For single exon genes, all the programs tend to perform worse, although the 3′ internal exon boundary of the cDNA is predicted better than the 5′ internal exon boundary. Similarly, the 3′ internal exon boundaries are generally predicted better than the 5′ internal exon boundaries by all the programs, for genes with a small number of exons. At the protein level, Augustus and GlimmerHMM achieve higher sequence identity for genes with ≤7 exons, while Augustus and Genscan are more accurate for genes with more exons. Most of the perfectly predicted proteins (with 100% sequence identity) have less than 3 exons.

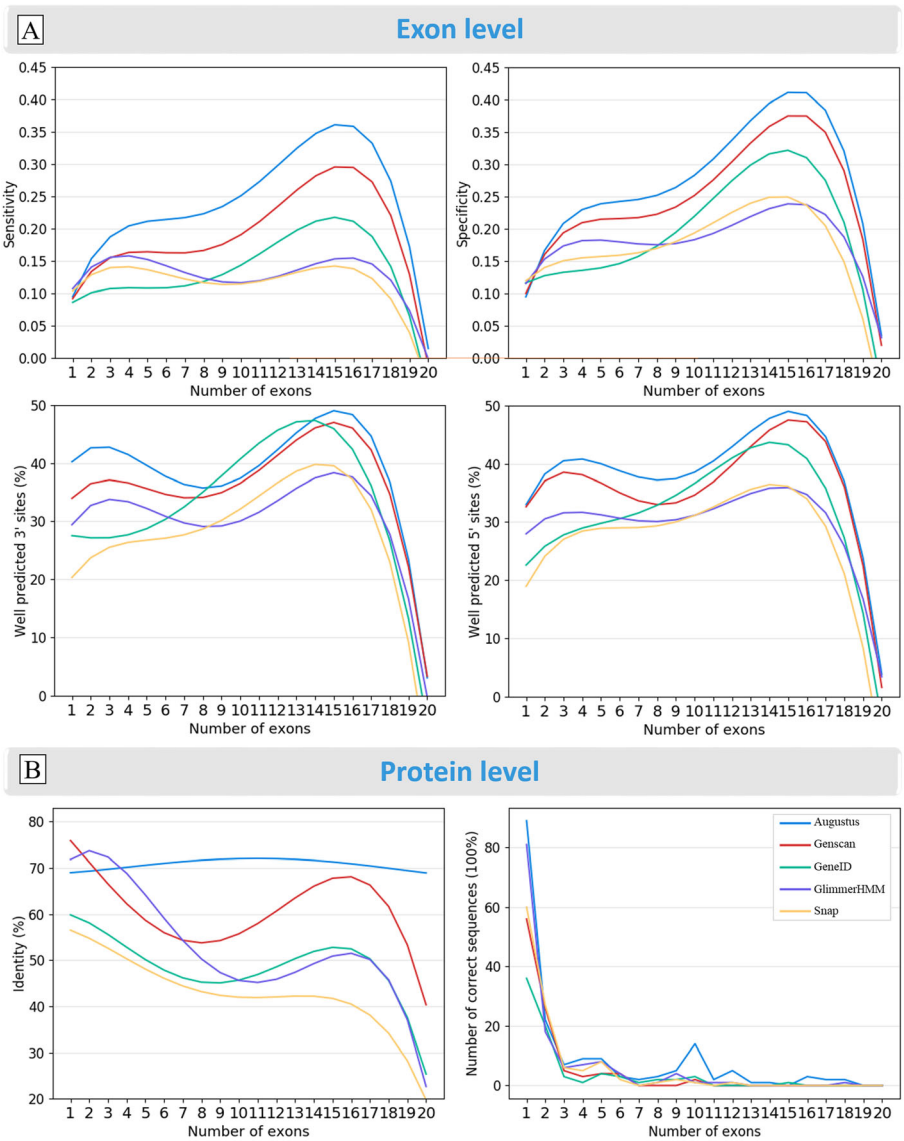


Fig. 9 Effect of exon map complexity on prediction quality at the **a** exon and **b** protein levels. A 4th degree polynomial curve fitting was used to represent the results more clearly. Sequences with 21–24 exons were not included, due to the low number of sequences in the benchmark with these exon counts. 3′ and 5′ are the proportion of correctly predicted 3′ and 5′ internal exon boundaries respectively. %Identity indicates the sequence identity observed between the predicted proteins and the Confirmed benchmark sequences

We then assessed the effect of exon lengths on the prediction quality of the five programs, using the 756 Confirmed sequences without UDT regions. Figure 10a and Additional file 1: Table S10A show the proportion of Correct exons (both 5′ and 3′ exon boundaries correctly predicted) depending on the exon length. The short exons (< 50 nucleotides) are generally the least accurate, with the best program, Augustus, achieving only 18% Correct short exons. Medium length exons (50–200 nucleotides) are predicted better than longer exons (> 200 nucleotides) for Augustus and Genscan.

To further investigate the exon prediction, each exon predicted by a gene prediction program was classified as 'Correct' if both exon boundaries were correctly predicted, 'Wrong (5')' or 'Wrong (3′)' if the 5′ or 3′ exon boundary was badly predicted respectively, and 'Wrong' if both boundaries were badly predicted. In some cases, the predicted exon has good 5′ and 3′ exon boundaries, however they correspond to 2 different benchmark exons, so these exons are classed as 'Wrong (Fusion)'. Figure 10b and Additional file 1: Table S10B show the number of Correct, Wrong, Wrong (5′), Wrong (3′) and Wrong (Fusion) exons, according to the exon lengths.

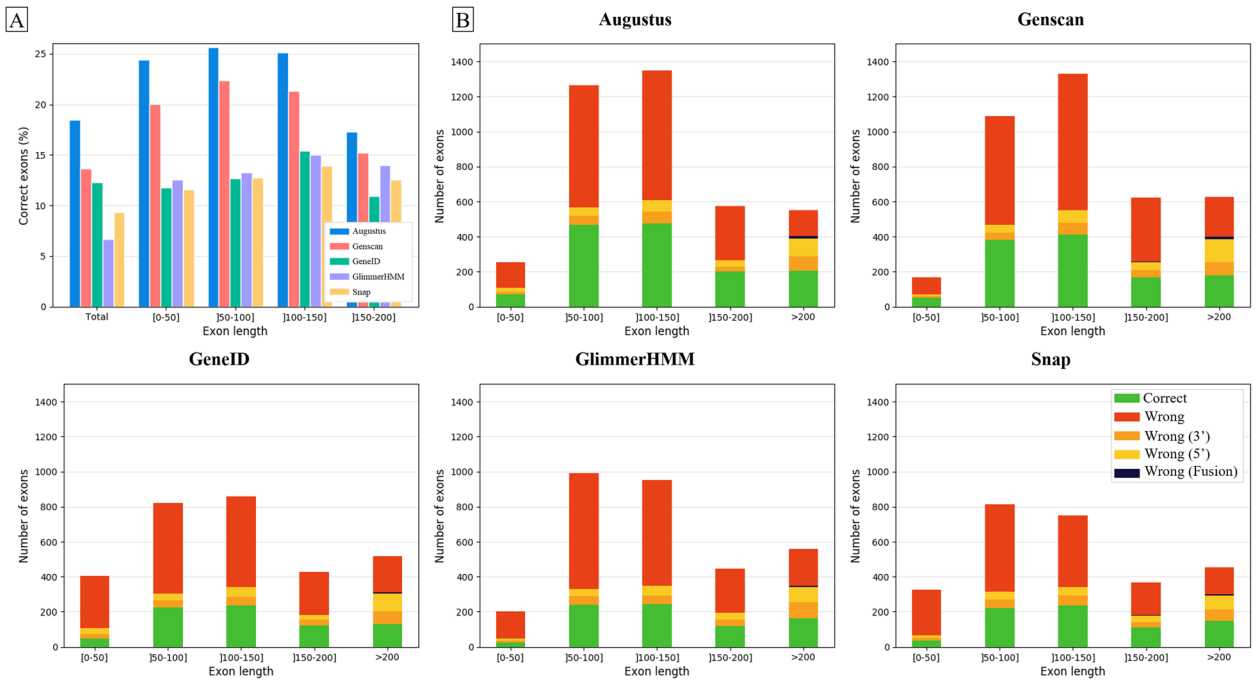


Fig. 10 Effect of exon length on exon prediction quality. **a** Proportion of benchmark exons correctly predicted depending on the exon length. **b** Number of exons predicted correctly, with one of the 5′ or 3′ exon boundaries correct, or with both boundaries wrongly predicted, for each of the five programs

Overall, there are more 'Wrong' exons than 'Correct' exons for all exon lengths and for all the programs. Interestingly, the number of predicted exons with only one boundary correctly predicted, i.e. Wrong (5′) or Wrong (3′), is small for all exon lengths, except for exons with > 200 nucleotides.

Factors associated with the protein product

In this section, prediction accuracy is measured at the protein level and is estimated by the percent sequence identity of the predicted protein compared to the benchmark protein.

First, we investigated the effect of protein length on protein prediction quality. We divided the 756 Confirmed sequences without UDT regions into five groups, with different protein lengths ranging from 50 to 1000 amino acids (Additional file 1: Fig. S8). Note that the very large proteins (> 1000 amino acids) in the benchmark are all classified as Unconfirmed and are therefore not included in this study. Figure 11 and Additional file 1:

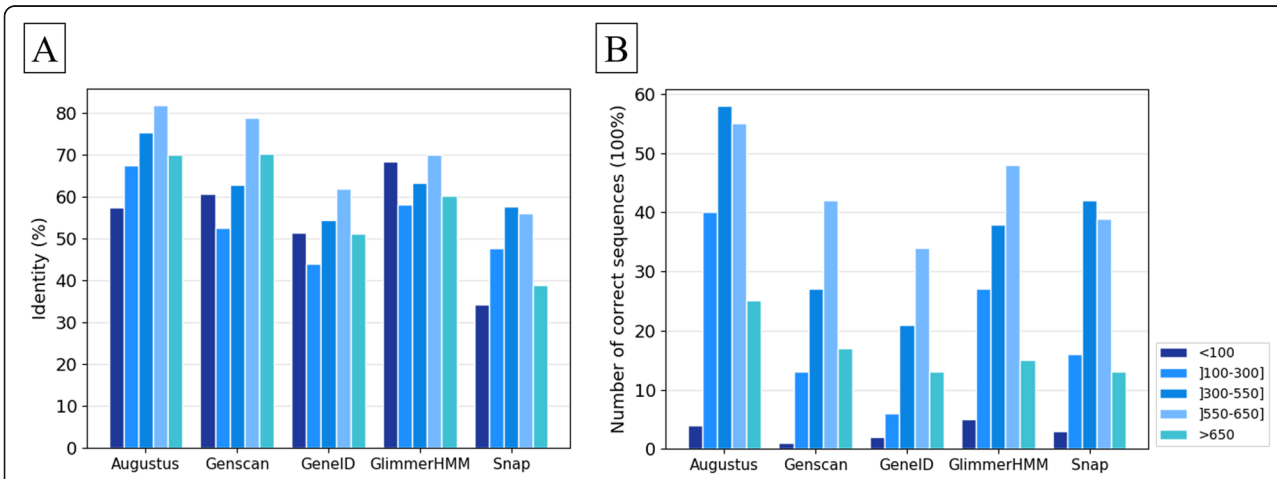


Fig. 11 Effect of protein length on prediction accuracy: **a** average percent identity between the predicted and the benchmark protein sequences, **b** number of proteins perfectly predicted with 100% sequence identity

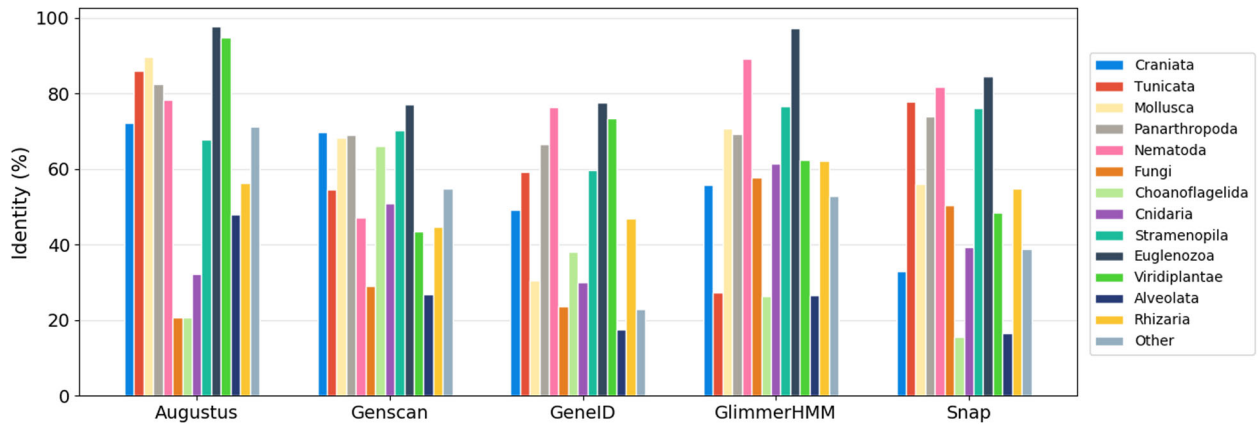


Fig. 12 Prediction performance for sequences from different clades. The 'Other' group contains the Apusozoa, Cryptophyta, Diplomonadida, Haptophyceae, Heterolobosea, Parabasalia clades, as well as Placozoa, Annelida and urchin. % Identity indicates the average percent identity between the predicted and the benchmark protein sequences

Table S11 show the mean accuracies obtained by the five programs for the different length proteins. The prediction accuracy generally decreases for shorter proteins and for protein lengths > 650 amino acids. For proteins with < 100 amino acids, GlimmerHMM achieves the best results with 68% sequence identity and five (25%) perfectly predicted proteins (100% identity), while Augustus obtains only 57% sequence identity and four perfectly predicted proteins.

We then studied the phylogenetic origin of the proteins and the availability of suitable species models in the different programs. Figure 12 and Additional file 1: Table S12 show the performance of the five gene prediction programs for the sequences in the different clades in G3PO. The accuracy of each program is highly variable between the different clades, probably due to the availability of suitable prediction models for some species. For the sequences in the Craniata clade, Augustus and Genscan achieve the highest accuracy (72 and 70% respectively), while Snap has the lowest accuracy (33%). In contrast, Augustus obtains lower accuracy (21%) for Fungi proteins, compared to the highest accuracy obtained by GlimmerHMM (58%). The proteins in the Euglenoezoa clade are predicted with the highest accuracy by all the programs, although this might be explained by their low EMC. Choanoflagellida and Cnidaria proteins are the least well predicted (except for Genscan), but these clade contain only a few sequences (5 and 6 sequences respectively) and this result remains to be confirmed.

Effect of protein sequence errors

Finally, we investigated the performance of the prediction programs for the 904 Unconfirmed sequences, where potential sequence errors were observed in the benchmark sequences. As mentioned above, the G3PO benchmark sequences were extracted from the Uniprot database, which means that many of the proteins are not supported by experimental evidence. In this test, we wanted to estimate the prediction accuracy of the five gene prediction programs for the Unconfirmed benchmark sequences. Since the Unconfirmed sequences could not be used as a ground truth, here we measured prediction accuracy based on a closely related Confirmed sequence (see Methods). Table 2 shows the prediction accuracies achieved by each program for the sets of Confirmed and Unconfirmed sequences. As might be expected, the Unconfirmed sequences are predicted with lower accuracy than the Confirmed sequences by all five programs. Augustus and Genscan achieved the highest accuracy (56, 50% respectively) for the Unconfirmed sequences. For comparison purposes, we also calculated the accuracy scores for the Unconfirmed benchmark

Table 2 Effect of protein sequence quality measured at the protein level. %Identity indicates the average sequence identity observed between the predicted and benchmark protein sequences for the test sets of Confirmed and Unconfirmed proteins

	Confirmed proteins (%Identity)	Unconfirmed proteins (%Identity)
Augustus	74.44	56.22
Genscan	67.13	49.86
GeneID	52.26	38.52
GlimmerHMM	59.36	45.60
Snap	44.20	41.70

proteins. The benchmark proteins had higher accuracy (76%) than any of the methods tested here, implying that the more complex pipelines used to curate proteins in Uniprot can effectively improve the results of ab initio methods.

Discussion

Several recent reviews [3, 22, 23] have highlighted the fact that automated genome annotation strategies still have difficulty correctly identifying protein-coding genes. This failure might be explained by the quality of the draft genome assemblies, the complexity of eukaryotic exon maps, high levels of genetic sequence divergence or deviations from canonical genetic characteristics [36]. Consequently, it is essential to benchmark the existing different gene prediction strategies to assess their reliability, to identify the most promising approaches, but also to limit the spread of errors in protein databases [37]. An ideal benchmark for gene prediction programs should include proteins encoded by real genomic sequences. Unfortunately, most of the protein sequences in the public databases have not been verified by experimental means, with the exception of the manually annotated Swiss-Prot sequences (representing only 0.3% of UniProt), and contain many sequence annotation errors. It is therefore dangerous to use them to estimate the accuracy of the prediction programs.

G3PO is a new gene prediction benchmark containing 1793 orthologous sequences from 20 different protein families, and designed to be as representative as possible of the living world. It includes sequences from phylogenetically diverse organisms, with a wide range of different genomic and protein characteristics, from simple single exon genes to very long and complex genes with over 20 exons. The quality of the protein sequences in the benchmark was ensured by excluding sequences containing potential annotation errors, including deletions, insertions and mismatched segments. We also characterized the test sets in the benchmark using different features at the genome, gene structure and protein levels. This in-depth characterization allowed us to investigate the impact of these features on gene prediction accuracy.

One of the main limitations of the benchmark concerns the fact that the protein sequences were extracted from the Uniprot database, where a 'canonical' protein isoform is defined based on cross-species conservation and the conservation of protein structure and function. Consequently, programs that predicted more minor isoforms created by alternative splicing events were penalized in our evaluations. Unfortunately, there is currently no ideal solution to this. In the future, gene prediction programs will need to evolve to predict all isoforms for a gene. Another limitation of the benchmark concerns the evaluation of the gene prediction results with respect to a single benchmark sequence. It is possible that the flanking regions used in some tests covered more than one gene, and that some programs successfully predicted one or more exons from these neighboring genes in addition to the reference gene.

The ab initio gene prediction programs included in the benchmark study are based on statistical models that are trained using known proteins and genes, and typically perform well at predicting conserved or well-studied genes [33, 38]. However, ab initio prediction accuracy has been previously shown to decrease in some special cases, such as small proteins [39], organism-specific genes or other unusual genes [40–42]. Our goal was therefore to identify the strengths and weaknesses of the programs, but also to highlight genomic and protein characteristics that could be incorporated to improve the prediction models.

In terms of overall quality, the gene prediction programs were generally ranked in agreement with previous findings, with Augustus and Genscan achieving the best overall accuracy scores. However, it should be noted that Augustus is also the most computationally expensive method, taking over 1 h to process the 87 Mb corresponding to the 1793 benchmark sequences, compared to the fastest program, GeneID, which required only 4 min.

We then performed a more in-depth study of the different factors affecting prediction accuracy. At the genome level, an increase in accuracy was generally observed when at least 2Kb flanking regions were added, reflecting the fact that all the programs try to model in vivo gene translation systems to some extent by taking into account the different regulatory signals found within and outside the gene [43]. In contrast, undetermined regions in the gene sequences had a negative effect on the accuracy of all the prediction programs, even when they occur outside the coding exons of the genes. Since undetermined or ambiguous regions are likely to occur more often in low coverage genomes, this is an important issue that needs to be addressed by the developers of gene prediction software.

At the gene structure level, we found that the number of exons affects the accuracy of all the programs and that gene prediction is generally more difficult for complex exon maps, as might be expected. Concerning the effect of exon length, the programs appear to be optimized for intermediate length exons (50–200 nucleotides), since none of the programs was able to reliably predict exons that were shorter (< 50 nucleotides) or longer (> 200 nucleotides). Protein length had a similar effect to that observed for exon length, since the programs seem to be optimized for intermediate length proteins (300–650 amino acids). This result confirms previous findings that smaller proteins (less than 100 amino acids) are often missed in genome annotations

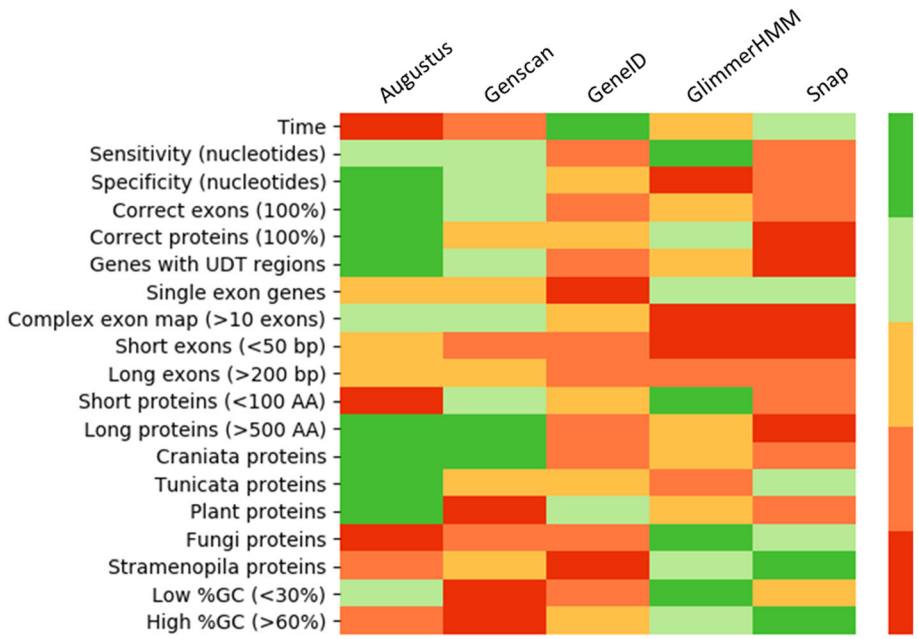


Fig. 13 Strengths and weaknesses of the gene prediction programs evaluated in this study. Heatmap colors are: dark green = best program, light green = 2nd best program, yellow = 3rd best program, orange = 4th best program, red = 5th best program

[39], although we also demonstrated that long proteins are also more likely to be badly predicted. Finally, the phylogenetic origin of the benchmark sequences had a large effect on prediction accuracy, with different programs producing the best results depending on the specific species. The two best scoring programs, Augustus and Genscan use different strategies, since Augustus includes > 100 different species models, while Genscan has only three models.

Each of the analyses performed here highlights different strengths or weaknesses of the prediction programs, as summarized in the heat map shown in Fig. 13. The in-depth characterization of the benchmark sequences and the detailed information extracted from the analyses provide essential elements that could be used to improve model training and therefore gene prediction. It may be interesting to further analyze the weaknesses identified, including small proteins, very long proteins, proteins coded by a large number of exons, proteins from non-model organisms, etc.

Finally, the Unconfirmed sequences identified in this study represent a goldmine for the identification of atypical gene features, for example atypical regulatory signals or splice sites, that are not fully taken into account in the current prediction models. More than 50% of the original reference protein sequences extracted from public databases were found to contain at least one error. They therefore represent very challenging test cases that were not resolved by the combined ab initio and similarity-based curation processes used to annotate these proteins. We accurately located the errors within these badly predicted sequences and classified them into 9 groups. Here, we performed a preliminary analysis using the erroneous sequences that confirmed our idea that all the prediction programs are less accurate for these proteins. A more comprehensive analysis of these proteins will be published elsewhere.

Conclusions

The complexity of the genome annotation process and the recent activity in the field mean that it is timely to perform an extensive benchmark study of the main computational methods employed, in order to obtain a more detailed knowledge of their advantages and disadvantages in different situations. Currently, most of the programs used for gene prediction are based on statistical approaches and perform relatively well in intermediate cases. However, they have difficulty identifying more extreme cases, such as very short or very long proteins, complex exon maps, or genes from less well studied species. Recently, artificial intelligence approaches have been applied to some specific tasks, for example DeepSplice [44] or SpliceAI [45] for the prediction of splice sites. The further development of these approaches should contribute to production of high quality gene predictions that can be leveraged downstream to improve functional annotations, evolutionary studies, prediction of disease genes, etc.

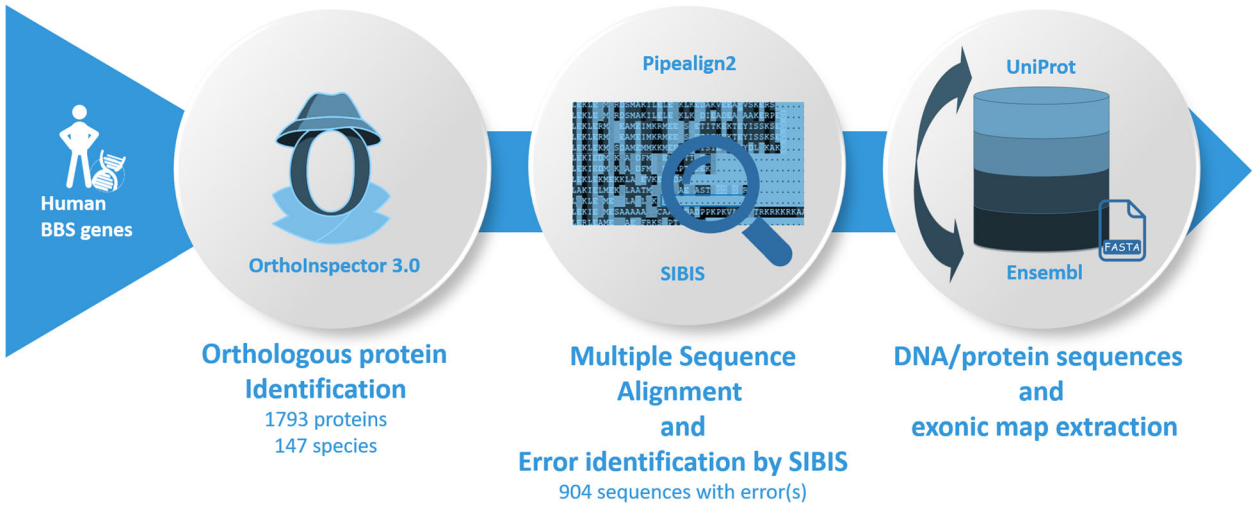


Fig. 14 Schematic view of the pipeline used to construct the benchmark

Methods

Benchmark test sets

To construct a benchmark set of eukaryotic genes, we selected the 20 human Bardet-Biedl Syndrome (BBS) proteins (Additional file 1: Table S2). Based on this initial gene set, we extended the test sets using the pipeline shown in Fig. 14 and described in detail below.

(i) For each of the 20 human proteins, orthologous proteins were identified in 147 eukaryotic organisms (Additional file 1: Table S1) using OrthoInspector version 3.0 [46], which was built using proteins from the Uniprot Reference Proteomes database [34] (Release 2016_11). For each species, we selected one ortholog sharing the highest percent identity with the human sequence. This resulted in a total of 1793 protein sequences, of which 65 (3.6%) were found in the curated Swissprot database. The number of proteins in each BBS family is provided in Additional file 1: Table S2. BBS 6,10,11,12,15, 16 and 18 are specific to Metazoa (with some exceptions), and therefore contain fewer sequences than the other families.

(ii) Since the reference protein sequences extracted from the Uniprot database may contain errors, we identified potentially unreliable sequences based on multiple sequence alignments (MSA). MSAs were constructed for each protein family using the Pipealign2 tool (http://www.lbgi.fr/pipealign) and manually refined to identify and correct misaligned regions. The SIBIS (version 1.0) program [47] using a Bayesian framework combined with Dirichlet mixture models and visual inspection, was used to identify inconsistent sequence segments. These segments might indicate that different isoforms are defined as the canonical sequence for different organisms, or they might indicate a badly predicted protein (Additional file 1: Fig. S3). SIBIS classifies the potential sequence errors into 9 categories: N-terminal deletion, N-terminal extension, N-terminal mismatched segment, C-terminal deletion, C-terminal extension, C-terminal mismatched segment, internal deletion, internal insertion and internal mismatched segment. Of the 1793 protein sequences identified in step (i), 889 proteins had no errors (called “Confirmed”) and 904 proteins had at least one potential error (called “Unconfirmed”). At this stage, the BBS14 protein was excluded from the benchmark because the MSA contained too many misalignments.

(iii) For each orthologous protein, the genomic sequence was extracted from the Ensembl database [35]. Genomic sequences were extracted with the ‘soft mask’ option, i.e. repeated or low complexity regions are replaced by lower case nucleotides. These are generally ignored by gene prediction programs. We also found regions with ‘n’ characters, which are used to indicate undetermined or ambiguous nucleotides (IUPAC nomenclature) probably caused by genome sequencing errors or assembly gaps. A sequence segment with a run of n characters was defined as an undetermined (UDT) region. Additional file 1: Table S5 summarizes the general statistics of these 283 sequences with UDT regions. Finally, we identified the Ensembl transcript corresponding to the Uniprot protein sequence, (generally the ‘canonical transcript’ from APPRIS [48]) in order to construct the exon map by extracting the positions of all exons/introns, including the 5′/3′ untranslated regions when available.

(iv) For the baseline tests, we included flanking sequences of length 150 bases upstream and downstream of the gene. To make the benchmark set more challenging, we also extracted genomic sequences corresponding to 2Kb, 4Kb, 6Kb, 8Kb, 10Kb upstream and downstream of the gene sequence.

Gene prediction methods

The programs tested are listed in Table 1 with the main features, including the HMM model used to differentiate intron/exon regions, and the specific signal sensors used to detect the presence of functional sites. Transcriptional signal sensors include the initiator or cap signal located at the transcriptional start site and the upstream TATA box promoter signal, as well as the polyadenylation signal (a consensus AATAAA hexamer) located downstream of the coding region and the 3′ UTR. Translational signals include the "Kozak sequence" located immediately upstream of the start codon [49]. For higher eukaryotes, splice site signals are also incorporated, including donor and acceptor sites (GT-AG on the intron sequence) and the branch point [yUnAy] [50] (underlined A is the branch point at position zero and y represents pyrimidines, n represents any nucleotide) located 20–50 bp upstream of the AG acceptor.

The command lines used to run the programs are:

augustus --species=<species> --softmasking=1 --gff3=off <sequence.fasta>

genscan <species> <sequence.fasta>

geneid -A -P <species> <sequence.fasta>

glimmerhmm <sequence.fasta> -d <species> -g

snap -gff -quiet -lcmask <species> <sequence.fasta> --a protein.fasta

where <species> indicates the species model used and <sequence.fasta> contains the input genomic sequence.

All programs were run on an Intel(R) Xeon(R) CPU E5–2695 v2 @ 2.40Ghz, 12 cores, with 256 Go RAM. Each prediction program was run with the default settings, except for the species model to be used. As the benchmark contains sequences from a wide range of species, we selected the most pertinent training model for each target species, based on the taxonomic proximity between the target and model species. For each program, we compared the taxonomy of the target species with the taxonomy for each model species available, where taxonomies were obtained from the NCBI Taxonomy database (https://www.ncbi.nlm.nih.gov/taxonomy). We then selected the model species that was closest to the target in the taxonomic tree.

Evaluation metrics

The performance of the gene prediction programs is based on the measures used in [29], calculated at three different levels: nucleotides, exons and complete proteins. The significance of pairwise comparisons of the evaluation metrics was evaluated using the paired t-test.

At the nucleotide level, we measure the accuracy of a gene prediction on a benchmark sequence by comparing the predicted state (exon or intron) with the true state for each nucleotide along the benchmark sequence. Nucleotides correctly predicted to be in either an exon or an intron are considered to be True Positives (TP) or True Negatives (TN) respectively. Conversely, nucleotides incorrectly predicted to be in exons or introns are considered to be False Positives (FP) or False Negatives (FN) respectively. We then calculated different performance statistics, defined below.

Sensitivity measures the proportion of benchmark nucleotides that are correctly predicted:

$$Sn = \frac{TP}{TP + FN.}$$

The specificity measure that is most widely used in the context of gene prediction is the proportion of nucleotides predicted in exons that are actually in exons:

$$Sp = \frac{TP}{TP + FP}$$

The F1 score represents the harmonic mean of the sensitivity and specificity values:

$$F1 = 2 * \frac{Sp * Sn}{Sp + Sn}$$

At the exon structure level, we measure the accuracy of the predictions by comparing predicted and true exons along the benchmark gene sequence. An exon is considered correctly predicted (TP), when it is an exact match to the benchmark exon, i.e. when the 5′ and 3′ exon boundaries are identical. All other predicted exons are then considered FP. Sensitivity and specificity are then defined as before.

Since the definition of TP and TN exons above is strict, we also calculated two additional measures similar to those defined in [29] (Additional file 1: Fig. S9). First, true exons with or without overlap to predicted exons are considered to be Missing Exons (ME) and the MEScore is defined as:

$$MEScore = \frac{ME}{Total\ number\ of\ true\ exons}$$

Second, predicted exons with or without overlap to true exons are considered Wrong Exons (WE). The WEScore is defined as:

$$WEScore = \frac{WE}{Total\ number\ of\ predicted\ exons}$$

We also determined the proportion of correctly predicted 5′ and 3′ exon boundaries, as follows:

$$5' = \frac{\text{number of true } 5' \text{ exon boundaries correctly predicted} * 100}{\text{number of correct predicted exons} + \text{number of wrong exons}}$$

$$3' = \frac{\text{number of true } 3' \text{ exon boundaries correctly predicted} * 100}{\text{number of correct predicted exons} + \text{number of wrong exons}}$$

At the protein level, we measure the accuracy of the protein products predicted by a program. Since a program may predict more than one transcript for a given gene sequence in the benchmark, we calculate the percent identity between the benchmark protein and all predicted proteins and the predicted protein with the highest percent identity score is selected. To calculate the percent identity score between the benchmark protein and the predicted protein, we construct a pairwise alignment using the MAFFT software (version 7.307) [51] and the percent identity is then defined as:

$$\%Identity = \frac{Number\ of\ identical\ amino\ acids * 100}{Length\ of\ benchmark\ protein}$$

Evaluation metric for unconfirmed benchmark proteins

Since the Unconfirmed proteins in the benchmark are badly predicted and have at least one identified sequence error, the %Identity score defined above for the Confirmed sequences cannot be used. Instead, we compare the protein sequences predicted by the programs with the most closely related Confirmed sequence found in the corresponding MSA. Thus, for a given Unconfirmed sequence, E, we calculated the sequence identity between E (excluding the sequence segments with predicted errors) and all the orthologous sequences in the corresponding MSA. If a Confirmed orthologous sequence, V, was found that shared ≥50% identity with E, then the sequence V was used as the reference protein to evaluate the program prediction accuracy.

As before, a pairwise alignment between the prediction protein and sequence V was constructed using MAFFT and the %Identity score was calculated. Finally, the accuracy score was normalized by the sequence identity shared between the E and V benchmark sequences.

$$Accuracy = \frac{\%Identity(P, V) * 100}{\%Identity(E, V)}$$

Supplementary information

Supplementary information accompanies this paper at https://doi.org/10.1186/s12864-020-6707-9.

Additional file 1: Tables S1–11, Figures S1–9.

Abbreviations

AA: Amino acid; BBS: Bardet-Biedl syndrome; Bp: Base pair; DNA: Deoxyribonucleic acid; EMC: Exon map complexity; F1: F1 score; FN: False negative; FP: False positive; HMM: Hidden Markov Model; Kb: Kilobase; ME: Missing exon; MSA: Multiple Sequence Alignment; RNA: Ribonucleic acid; Sn: Sensitivity; Sp: Specificity; TN: True negative; TP: True positive; UDT: Undetermined region; UTR: Untranslated region; WE: Wrong exon

Acknowledgements

The authors would like to thank the BISTRO and BICS Bioinformatics Platforms for their assistance.

Authors' contributions

NS developed the benchmark, performed the program benchmarking, and produced all graphical presentations. AJG and PC advised on the feature content of the test sets and supervised the comparative analyses. OP and JDT supervised the production and exploitation of the benchmark. All authors participated in the definition of the original study concept. All authors read and approved the final manuscript.

Funding

NS was supported by funds from the Swiss foundation BIONIRIA. This work was also supported by the ANR projects Elixir-Excelerate: GA-676559 and RAinRARE: ANR-18-RAR3–0006-02, and Institute funds from the French Centre National de la Recherche Scientifique, the University of Strasbourg.

Availability of data and materials

The DNA and protein sequences used in the G3PO benchmark, the scripts used to produce the results and the outputs of the gene prediction programs are available at http://git.lbgi.fr/scalzitti/Benchmark_study.

Ethics approval and consent to participate

Not applicable. All data presented in this article was extracted from publicly available sources.

Consent for publication

Not applicable

Competing interests

The authors declare that they have no competing interests.

Received: 29 November 2019 Accepted: 30 March 2020
Published online: 09 April 2020

References

1. DNA Sequencing Costs: Data | NHGRI. https://www.genome.gov/about-genomics/fact-sheets/DNA-Sequencing-Costs-Data. Accessed 30 Oct 2019.
2. Matz MV. Fantastic beasts and how to sequence them: ecological genomics for obscure model organisms. Trends Genet. 2018;34:121–32.
3. Salzberg SL. Next-generation genome annotation: we still struggle to get it right. Genome Biol. 2019;20:92 s13059–019–1715–2.
4. Mudge JM, Harrow J. The state of play in higher eukaryote gene annotation. Nat Rev Genet. 2016;17:758–72.
5. Danchin A, Ouzounis C, Tokuyasu T, Zucker J-D. No wisdom in the crowd: genome annotation in the era of big data - current status and future prospects. Microb Biotechnol. 2018;11:588–605.
6. Ozsolak F, Platt AR, Jones DR, Reifenberger JG, Sass LE, McInerney P, et al. Direct RNA sequencing. Nature. 2009;461:814–8.
7. Workman RE, Tang AD, Tang PS, Jain M, Tyson JR, Zuzarte PC, et al. Nanopore native RNA sequencing of a human poly(a) transcriptome. Nat Methods. 2019; (in press).
8. Yeh R-F, Lim LP, Burge CB. Computational inference of homologous gene structures in the human genome. Genome Res. 2001;11:803–16.
9. Birney E. GeneWise and Genomewise. Genome Res. 2004;14:988–95.
10. Solovyev V, Kosarev P, Seledsov I, Vorobyev D. Automatic annotation of eukaryotic genes, pseudogenes and promoters. Genome Biology. 2006:12.
11. Stanke M, Schöffmann O, Morgenstern B, Waack S. Gene prediction in eukaryotes with a generalized hidden Markov model that uses hints from external sources. BMC Bioinform. 2006;7:62.
12. Kapustin Y, Souvorov A, Tatusova T, Lipman D. Splign: algorithms for computing spliced alignments with identification of paralogs. Biol Direct. 2008;3:20.

13. Testa AC, Hane JK, Ellwood SR, Oliver RP. CodingQuarry: highly accurate hidden Markov model gene prediction in fungal genomes using RNA-seq transcripts. BMC Genomics. 2015;16:170.
14. Cook DE, Valle-Inclan JE, Pajoro A, Rovenich H, Thomma BPHJ, Faino L. Long-read annotation: automated eukaryotic genome annotation based on long-read cDNA sequencing. Plant Physiol. 2019;179:38–54.
15. Huang Y, Chen S-Y, Deng F. Well-characterized sequence features of eukaryote genomes and implications for ab initio gene prediction. Comput Struct Biotechnol J. 2016;14:298–303.
16. Burge C, Karlin S. Prediction of complete gene structures in human genomic DNA. J Mol Biol. 1997;268:78–94.
17. Salzberg SL, Pertea M, Delcher AL, Gardner MJ, Tettelin H. Interpolated Markov models for eukaryotic gene finding. Genomics. 1999;59:24–31.
18. Guigó R, Knudsen S, Drake N, Smith T. Prediction of gene structure. J Mol Biol. 1992;226:141–57.
19. Korf I. Gene finding in novel genomes. BMC Bioinform. 2004;5:59.
20. Stanke M, Waack S. Gene prediction with a hidden Markov model and a new intron submodel. Bioinformatics. 2003;19(Suppl 2):ii215–25.
21. Lomsadze A. Gene identification in novel eukaryotic genomes by self-training algorithm. Nucleic Acids Res. 2005;33:6494–506.
22. Simão FA, Waterhouse RM, Ioannidis P, Kriventseva EV, Zdobnov EM. BUSCO: assessing genome assembly and annotation completeness with single-copy orthologs. Bioinformatics. 2015;31:3210–2.
23. Drăgan M-A, Moghul I, Priyam A, Bustos C, Wurm Y. GeneValidator: identify problems with protein-coding gene predictions. Bioinformatics. 2016;32:1559–61.
24. Nishimura O, Hara Y, Kuraku S. Evaluating genome assemblies and gene models using gVolante. In: Kollmar M, editor. Gene prediction. New York: Springer New York; 2019. p. 247–56.
25. Kemena C, Dohmen E, Bornberg-Bauer E. DOGMA: a web server for proteome and transcriptome quality assessment. Nucleic Acids Res. 2019;47:W507–10.
26. Delcourt V, Staskevicius A, Salzet M, Fournier I, Roucou X. Small proteins encoded by Unannotated ORFs are rising stars of the proteome, Confirming Shortcomings in Genome Annotations and Current Vision of an mRNA. Proteomics. 2018;18:1700058.
27. Mat-Sharani S, Firdaus-Raih M. Computational discovery and annotation of conserved small open reading frames in fungal genomes. BMC Bioinform. 2019;19:551.
28. Rajput B, Pruitt KD, Murphy TD. RefSeq curation and annotation of stop codon recoding in vertebrates. Nucleic Acids Res. 2019;47:594–606.
29. Burset M, Guigó R. Evaluation of gene structure prediction programs. Genomics. 1996;34:353–67.
30. Rogic S, Mackworth AK, Ouellette FBF. Evaluation of gene-finding programs on mammalian sequences. Genome Res. 2001;11:817–32.
31. Guigo R. An assessment of gene prediction accuracy in large DNA sequences. Genome Res. 2000;10:1631–42.
32. Guigó R, Flicek P, Abril JF, Reymond A, Lagarde J, Denoeud F, et al. EGASP: the human ENCODE Genome Annotation Assessment Project. Genome Biol. 2006:31.
33. Goodswen SJ, Kennedy PJ, Ellis JT. Evaluating high-throughput Ab initio gene finders to discover proteins encoded in eukaryotic pathogen genomes missed by laboratory techniques. PLoS One. 2012;7:e50609.
34. The UniProt Consortium. UniProt: the universal protein knowledgebase. Nucleic Acids Res. 2017;45:D158–69.
35. Hubbard T, Barker D, Birney E, Cameron G, Chen Y, Clark L, et al. The Ensembl genome database project. Nucleic Acids Res. 2002;30:38–41.
36. Wilbrandt J, Misof B, Panfilio KA, Niehuis O. Repertoire-wide gene structure analyses: a case study comparing automatically predicted and manually annotated gene models. BMC Genomics. 2019;20:753.
37. Schnoes AM, Brown SD, Dodevski I, Babbitt PC. Annotation error in public databases: Misannotation of molecular function in enzyme Superfamilies. PLoS Comput Biol. 2009;5.
38. Yandell M, Ence D. A beginner's guide to eukaryotic genome annotation. Nat Rev Genet. 2012;13:329–42.
39. Sberro H, Fremin BJ, Zlitni S, Edfors F, Greenfield N, Snyder MP, et al. Large-scale analyses of human microbiomes reveal thousands of small, novel genes. Cell. 2019;178:1245–1259.e14.
40. Ter-Hovhannisyan V, Lomsadze A, Chernoff YO, Borodovsky M. Gene prediction in novel fungal genomes using an ab initio algorithm with unsupervised training. Genome Res. 2008;18:1979–90.
41. Reid I, O'Toole N, Zabaneh O, Nourzadeh R, Dahdouli M, Abdellateef M, et al. SnowyOwl: accurate prediction of fungal genes by using RNA-Seq and homology information to select among ab initio models. BMC Bioinformatics. 2014;15:229.
42. Hoff KJ, Lange S, Lomsadze A, Borodovsky M, Stanke M. BRAKER1: unsupervised RNA-Seq-based genome annotation with GeneMark-ET and AUGUSTUS: table 1. Bioinformatics. 2016;32:767–9.
43. Matera AG, Wang Z. A day in the life of the spliceosome. Nat Rev Mol Cell Biol. 2014;15:108–21.
44. Zhang Y, Liu X, MacLeod J, Liu J. Discerning novel splice junctions derived from RNA-seq alignment: a deep learning approach. BMC Genomics. 2018;19. https://doi.org/10.1186/s12864-018-5350-1.
45. Jaganathan K, Kyriazopoulou Panagiotopoulou S, McRae JF, Darbandi SF, Knowles D, Li YI, et al. Predicting splicing from primary sequence with deep learning. Cell. 2019;176:535–548.e24.
46. Nevers Y, Kress A, Defosset A, Ripp R, Linard B, Thompson JD, et al. OrthoInspector 3.0: open portal for comparative genomics. Nucleic Acids Res. 2019;47(Database issue):D411–8.
47. Khenoussi W, Vanhoutrève R, Poch O, Thompson JD. SIBIS: a Bayesian model for inconsistent protein sequence estimation. Bioinformatics. 2014;30:2432–9.
48. Rodriguez JM, Maietta P, Ezkurdia I, Pietrelli A, Wesselink J-J, Lopez G, et al. APPRIS: annotation of principal and alternative splice isoforms. Nucleic Acids Res. 2013;41(Database issue):D110–7.
49. Kozak M. Possible role of flanking nucleotides in recognition of the AUG initiator codon by eukaryotic ribosomes. Nucleic Acids Res. 1981;9:5233–52.
50. Gao K, Masuda A, Matsuura T, Ohno K. Human branch point consensus sequence is yUnAy. Nucleic Acids Res. 2008;36:2257–67.
51. Katoh K, Standley DM. MAFFT multiple sequence alignment software version 7: improvements in performance and usability. Mol Biol Evol. 2013;30:772–80.

Publisher's Note

Chapitre 6 - Identification et caractérisation des erreurs chez des organismes eucaryotes

6.1 La face cachée du *Big Data*

L'arrivée massive des données, produites par les technologies à haut débit, est une opportunité inestimable pour la recherche en biomédecine et en bio-informatique. Les nombreuses banques de données s'approvisionnent constamment en données, ouvrant la voie vers la découverte et la compréhension des mécanismes et des systèmes biologiques. Cependant, la vitesse de production des données et la concurrence révèlent la partie submergée de l'iceberg. De nombreux génomes sont partiellement séquencés et/ou assemblés, on parle dans ce cas de "*draft genomes*". Ces derniers sont incomplets, dans certains cas même incohérents (Denton *et al.*, 2014) et incorporent de nombreux fragments de séquences indéterminées comprenant des nucléotides non identifiés (notés 'N'). L'annotation des gènes codant pour des protéines est alors un défi sur ces génomes fragmentés car la sur/sous-prédiction de gènes induit en erreur les chercheurs. Plus inquiétant encore, les erreurs générées se propagent dans les banques de données affectant les conclusions des recherches exploitant ces données (Prosdocimi *et al.*, 2012; Deutekom *et al.*, 2019). Un effort supplémentaire doit donc être fourni afin d'identifier, caractériser et corriger ces erreurs.

Les analyses comparatives de G3PO, décrites dans le chapitre précédent, ont permis de mettre en avant certaines caractéristiques des séquences erronées ainsi que les failles des programmes de prédiction, cependant aucune information sur l'origine ou la cause des erreurs n'a été décrite.

6.2 L'univers des erreurs

Au travers des analyses avec G3PO, nous avons identifié qu'en moyenne une séquence sur deux, issue des bases de données publiques, compte au moins une erreur. La classification de ces erreurs repose sur la nomenclature du programme SIBIS, qui établit 9 catégories différentes (voir Matériels et méthodes section 6.3). D'après les analyses comparatives des programmes de prédiction *ab initio* susmentionnées, 30% des erreurs sont du type *mismatch*

interne, ce qui représente la catégorie la plus importante. Ces résultats démontrent que les meilleurs programmes et pipelines d'annotations de gènes codant pour des protéines commettent encore énormément d'erreurs notamment pour les organismes non-modèles. L'origine de ces erreurs peut être multifactorielle et on dénombre plusieurs types d'anomalies dont les causes sont soit les données, soit le modèle de prédiction. Ainsi, elles peuvent être dû au :

- mauvais séquençage : présence de régions indéterminées (caractérisée par des nucléotides 'N'),
- mauvais assemblage : présence de fragments de séquences incorrectes et de régions indéterminées,
- absence de données : pour des approches par homologie,
- limites du modèle : le modèle a du mal à généraliser en raison d'un manque de données, de données erronées ou non représentatives de l'espèce cible, *e.g.* un modèle primate peut être très bon pour l'humain et très mauvais pour *Pongo pygmaeus*. De plus, certaines caractéristiques génomiques rares peuvent induire en erreur le modèle comme des SE non canoniques ou des exons/introns très petits (<30 nucléotides).

Les erreurs induites par ces anomalies sont diverses et se caractérisent par :

- l'absence de prédiction d'une protéine,
- des gènes mal délimités car leurs bornes 5' et/ou 3' sont aberrantes,
- un décalage du cadre de lecture,
- des SE faux,
- une rétention d'intron au sein de la CDS,
- une fragmentation ou une fusion de deux gènes.

6.3 Identification et correction des erreurs

Après avoir mis en évidence la proportion importante d'erreurs, nous avons continué à investiguer l'origine de ces erreurs. Afin d'assurer la qualité des données, nous avons restreint notre étude à une groupe d'organismes très proches, les primates, où l'Homme représente l'organisme modèle d'excellence considéré comme de haute qualité. Nous avons récupéré les protéomes de 10 primates à partir de la base de données UniProt et les avons comparés avec le protéome humain. Les protéines orthologues aux gènes humains ont été alignées pour obtenir

des MSA. Notre étude a été focalisée sur l'identification des erreurs de type *mismatch*, qui pour ces travaux ont été définies comme un segment d'au moins 20 AA et partageant moins de 50% d'identité avec la protéine humaine de référence. Nous nous sommes intéressés aux erreurs de type *mismatch* car leur identification dans un MSA est plus aisée. Si une séquence est fortement divergente des autres séquences des organismes phylogénétiquement proches, alors il est fort probable qu'il s'agisse d'une erreur. De plus, ce type d'erreur peut introduire des biais au cœur même des protéines modifiant directement des domaines protéiques importants, ce qui peut modifier les prédictions de la structure tridimensionnelle ou de la fonction de la protéine. Parmi les 82 305 erreurs identifiées sur les 176 478 protéines, 11 015 (13,4%) sont des *mismatchs*. Des caractéristiques particulières ont été identifiées confirmant que ces segments *mismatch* sont bien des erreurs. Ainsi, ce type d'erreur peut être dû à :

- la présence de nucléotides indéterminés (dans quasiment 50% des cas),
- des petits exons/introns (<30 nucléotides),
- des SE non-canoniques,
- des régions ne correspondant pas à un SE, mais considérées comme tel par les programmes de prédiction.

Lorsque les régions erronées ont été identifiées, un protocole (décrit dans la publication en 6.4) de correction a été appliqué afin de tenter d'améliorer la prédiction. Cependant, ce protocole est une "preuve de concept" nécessitant une élaboration plus robuste par la suite. Bien qu'il s'agisse d'un prototype, les résultats sont très encourageants et suggèrent qu'un segment plus 'cohérent' (classiquement un exon très proche du gène modèle humain et possédant des SE détectables) existe dans le gène du primate incriminé et qu'il sera possible à l'avenir de corriger beaucoup d'erreurs. En effet, les améliorations ont réduit le nombre global d'erreurs de *mismatch* et ont augmenter le pourcentage moyen d'identité et de couverture pour plus de 600 séquences. Des projets exploitant cette approche sont actuellement en cours de développement au laboratoire.

6.4 Publication

BMC Bioinformatics

RESEARCH ARTICLE

Open Access

Understanding the causes of errors in eukaryotic protein-coding gene prediction: a case study of primate proteomes

Corentin Meyer, Nicolas Scalzitti, Anne Jeannin-Girardon, Pierre Collet, Olivier Poch and Julie D. Thompson*

*Correspondence: thompson@unistra.fr
Department of Computer Science, ICube, CNRS, University of Strasbourg, Strasbourg, France

Abstract

Background: Recent advances in sequencing technologies have led to an explosion in the number of genomes available, but accurate genome annotation remains a major challenge. The prediction of protein-coding genes in eukaryotic genomes is especially problematic, due to their complex exon–intron structures. Even the best eukaryotic gene prediction algorithms can make serious errors that will significantly affect subsequent analyses.

Results: We first investigated the prevalence of gene prediction errors in a large set of 176,478 proteins from ten primate proteomes available in public databases. Using the well-studied human proteins as a reference, a total of 82,305 potential errors were detected, including 44,001 deletions, 27,289 insertions and 11,015 mismatched segments where part of the correct protein sequence is replaced with an alternative erroneous sequence. We then focused on the mismatched sequence errors that cause particular problems for downstream applications. A detailed characterization allowed us to identify the potential causes for the gene misprediction in approximately half (5446) of these cases. As a proof-of-concept, we also developed a simple method which allowed us to propose improved sequences for 603 primate proteins.

Conclusions: Gene prediction errors in primate proteomes affect up to 50% of the sequences. Major causes of errors include undetermined genome regions, genome sequencing or assembly issues, and limitations in the models used to represent gene exon–intron structures. Nevertheless, existing genome sequences can still be exploited to improve protein sequence quality. Perspectives of the work include the characterization of other types of gene prediction errors, as well as the development of a more comprehensive algorithm for protein sequence error correction.

Keywords: Genome annotation, Primates, Gene prediction, Protein sequence errors, Error correction

Background

An unprecedented number of genomes are being sequenced, offering a unique view of the specific characteristics of individual organisms and new opportunities to analyze life on a larger scale. An essential first step in the genome annotation process is

the identification of all the coding regions of the genome. However, discovering genes in the new genome assemblies is challenging, especially in eukaryotes where the aim is to establish accurate gene models with precise exon–intron structures of all genes [1, 2]. While model organisms such as human or mouse are well-studied and experimental evidence is available for many genes [3–5], many sequences for non-model organisms are predicted by computational pipelines and may thus be incomplete or incorrect [6–8]. Furthermore, erroneous gene predictions are often propagated across genomes by homology-based genome annotation strategies. As a result, nucleotide and protein repositories, such as UniProt [9], RefSeq [10] or Ensembl [11], contain an increasing number of sequence errors or inaccuracies [12, 13].

Commonly encountered errors include missed proteins [14, 15], wrong exon and gene boundaries [16], non-coding nucleotide sequence retention in coding exons [17], as well as fragmentation or fusion of gene models [18]. These errors can be propagated in downstream applications, leading to incoherent or incorrect conclusions [19, 20]. Errors can severely affect studies of individual proteins, including transcript quantification, phylogenetic tree inference, and protein structure or function predictions. It has also been shown that gene prediction errors can lead to biases in large-scale statistical studies [19, 21, 22]. Therefore, DNA and protein sequence quality is essential and sequence information needs to be accurate, reliable, and accessible in a clear and consistent manner [23, 24].

Various strategies have been developed to help identify and correct errors in gene annotations, using information from comparative or evolutionary analyses [25–27], experimental evidence from expressed sequence tags (ESTs) or RNA-sequencing (RNA-seq) [22, 28], or dedicated sequence datasets such as the G3PO benchmark [29]. These studies generally agree that up to 50% of the protein sequences in the public databases have an least one error. However, very few studies have attempted to identify the underlying reasons for the badly predicted sequences.

Here, we use a large set of 176,478 proteins from a representative set of ten primates in order to characterize different types of gene prediction errors and investigate their causes. The protein sequences were downloaded from the Uniprot reference proteomes and RefSeq databases. To identify potential sequence errors, including insertions, deletions and mismatched segments, we compared the primate sequences to the canonical isoforms of all known human proteins in the UniProt database. The UniProt canonical sequence for a given human protein generally corresponds to the most frequent or most conserved protein isoform in orthologous species. We then focused on the primate protein sequences containing mismatched segment errors, i.e. where part of the correct protein sequence is replaced with an alternative erroneous sequence, as shown in the example in Fig. 1. These errors cause particular problems for downstream applications, since the badly predicted amino acids can significantly affect structural and functional annotations, as well as conservation and phylogenetic analyses.

Results

Identification of sequence errors in primate proteins

To estimate the frequency of gene prediction errors, a large set of primate protein sequences was extracted from the Uniprot reference proteome database. The human

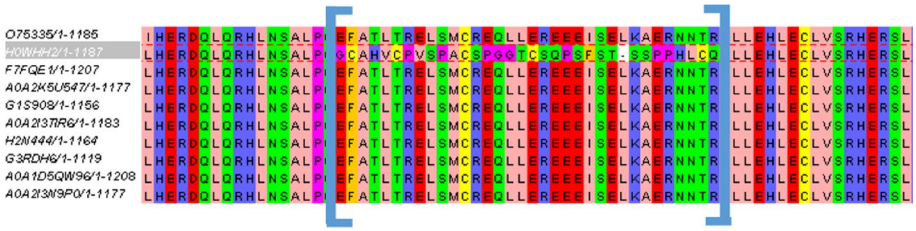


Fig. 1 Part of the MSA of the Human Liprin-Alpha-4 protein (first from top) with orthologous proteins in nine other primates. The sequence from *Otolemur garnettii* (H0WHH2) contains a 'mismatched segment' in the region highlighted by blue square brackets

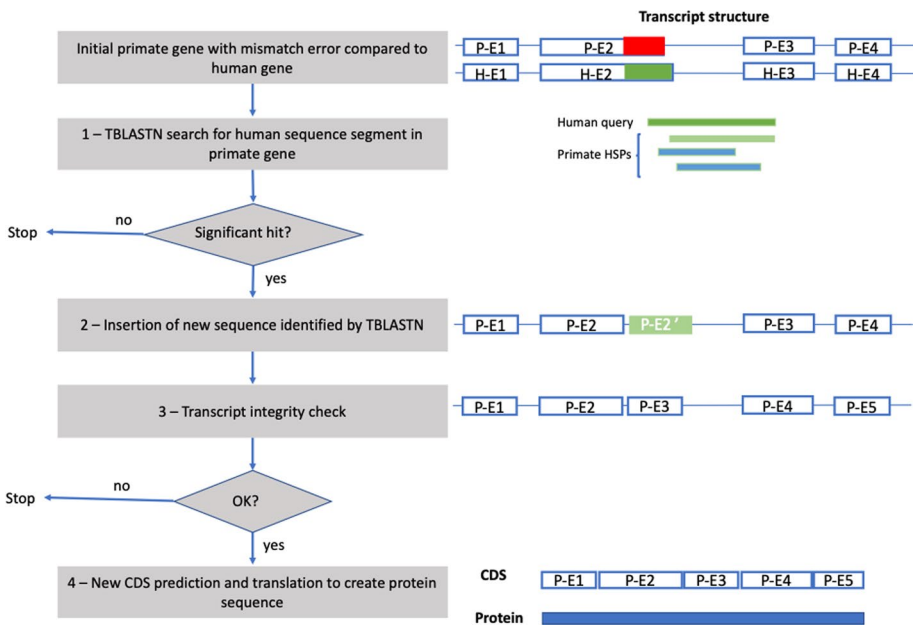


Fig. 2 Schema of transcript correction protocol. Blue boxes represent exons and are labeled P-E(*i*) for primate exon number *i* and H-E(*i*) for human exon number *i*. Horizontal thin blue lines represent introns. Red box represents a primate sequence segment coding for a mismatch, compared to the human sequence (dark green box). Light green box represents a significant TBLASTN hit or high-scoring segment pair (HSP), which will be inserted in the primate gene as a new exon

proteome was used as a reference, since the proteins have been very widely studied and for the purposes of this study, are assumed to be accurate. By comparing closely related primate sequences with the human proteins, we could identify potential gene prediction errors in the other primate proteomes (Fig. 2). Orthologous relationships were predicted using BLASTP searches, where each human protein sequence was used as a query to search each primate proteome. Starting with the 20,595 human proteins, we thus obtained between 14,000 and 19,000 orthologous protein sequences for each primate (Table 1). In total, 176,478 orthologous proteins were retrieved from the primate proteomes, with an average of 85.7% of the human sequences found in each primate.

Potential gene prediction errors were detected based on the MSAs of the orthologous proteins. For the 176,478 primate sequences extracted from Uniprot, a total

Table 1 Primate proteomes, number of orthologous proteins and human orthology rates

Organism	TAXID	Uniprot proteome	No. of predicted orthologs	% Human sequences with a primate ortholog
Homo sapiens (Human)	9606	UP000005640	20,595	100.0
Pan Troglodytes (Chimpanzee)	9598	UP000002277	19,010	92.3
Gorilla gorilla gorilla (Gorilla)	9595	UP000001519	18,540	90.0
Macaca mulatta (Macaque)	9544	UP000006718	18,327	89.0
Macaca fascicularis (Crab-eating macaque)	9541	UP000233100	17,976	87.3
Chlorocebus Sabaeus (Vervet-AGM)	60,711	UP000029965	17,948	87.1
Papio Anubis (Olive baboon)	9555	UP000028761	17,904	86.9
Pongo abelii (Orangutan)	9601	UP000001595	17,814	86.5
Nomascus leucogenys (Gibbon)	61,853	UP000001073	17,478	84.9
Callithrix jacchus (Marmoset)	9483	UP000008225	17,110	83.1
Otolemur garnettii (Bushbaby)	30,611	UP000005225	14,371	69.8

Table 2 Number of protein sequence errors detected in Uniprot primate sequences, for each of the error types

Primate	N-terminal extension	N-terminal deletion	C-terminal extension	C-terminal deletion	Internal insertion	Internal deletion	Mismatched segment	Total errors
Callithrix jacchus	1427	398	532	274	1315	1593	694	6233
Chlorocebus Sabaeus	914	1870	480	766	1840	2901	992	9763
Gorilla gorilla gorilla	820	1104	389	392	828	3211	1043	7787
Macaca fascicularis	918	667	448	295	964	2016	703	6011
Macaca mulatta	1657	402	661	235	1702	1403	561	6621
Nomascus leucogenys	757	1446	478	554	1186	6744	2641	13,806
Otolemur garnettii	603	1879	271	682	1417	2352	1887	9091
Papio Anubis	1134	434	500	286	1108	2342	1091	6895
Pongo abelii	1183	1673	370	981	1280	4877	802	11,166
Pan troglodytes	867	391	444	227	796	1606	601	4932
TOTAL	10,280	10,264	4573	4692	12,436	29,045	11,015	82,305

of 82,305 sequence errors were identified, including insertions, deletions and mismatched segments (Table 2). In other words, 47% of the protein sequences are expected to have at least one error in agreement with previous studies. Internal deletions were the most commonly detected errors (29,045), followed by internal insertions (12,436) and mismatched segments (11,015). More errors were detected at the N-terminus than at the C-terminus for both sequence extensions (10,280 and 4573, respectively) and deletions (10,264 and 4672, respectively), although this could be due at least in part to the error detection algorithm used.

As shown in Fig. 3, the frequency of gene prediction errors (normalized by the number of orthologous proteins detected) is not the same for all primate proteomes tested. While all the primates have globally similar distributions of the seven types of protein sequence error, some proteomes had significantly more errors than others. The number of errors per sequence ranges from 0.26 for *P. troglodytes* proteins to 0.79 for *N. leucogenys* proteins.

Mismatch gene prediction errors

We then focused our analyses on the 11,015 potential mismatch sequence errors, as these cause specific problems for subsequent analyses. Mismatches were identified in all of the primate proteomes, with the lowest number of mismatches occurring in the *M. mulatta* protein sequences (561 mismatches, or 0.03 per protein) and the highest number of mismatches in the *N. leucogenys* sequences (2641 mismatches, or 0.15 per protein).

To determine whether the mismatch error rate was linked to the evolutionary distance of the primates from the human reference, the percentage of sequences with at least one mismatch error was calculated for each primate proteome (Fig. 4a). While most of the primates have a mismatch rate between 2 and 5%, we identified two primates (*O. garnettii, N. leucogenys*) with as many as 11–12% of their sequences containing at least one mismatch. Surprisingly, the primate showing the highest mismatch rate (*N. leucogenys*) is not the most distant from human according to the tree of life (Fig. 4b). Furthermore, *G. g. gorilla*, which is very close to human in the tree of life, has a medium mismatch rate of 5.5%. These results taken together show that mismatch detection is not only dependent on the phylogenetic distance between primates and human, and there must be other factors to explain these observations.

Finally, to investigate whether the mismatch error rates were an issue related specifically to protein sequences from the Uniprot database, the same detection protocol was applied using the RefSeq protein database as the source for the primate reference proteomes. A similar number of orthologous proteins (181,223 primate proteins) were extracted from the RefSeq database, however 5971 mismatch errors were detected, compared to 11,015 for the Uniprot sequences. We conclude that mismatches are present in both databases, although the error rates seem to be lower in RefSeq than in Uniprot.

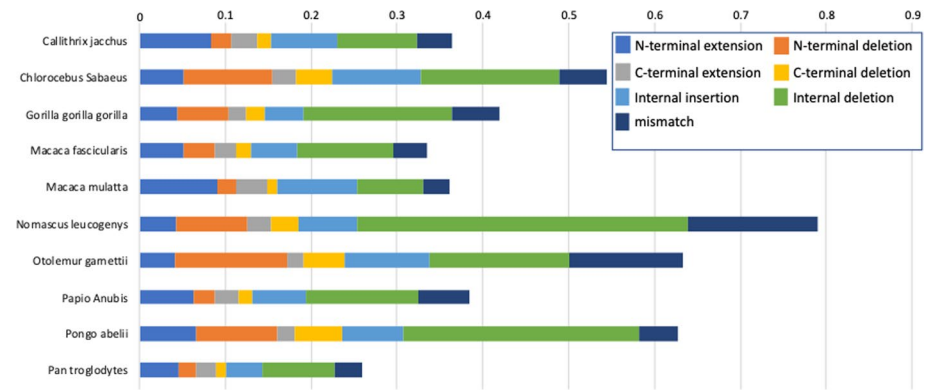


Fig. 3 Frequency of gene prediction errors per protein sequence, in each of the Uniprot primate proteomes

Characterization of mismatch errors

To understand why mismatch errors were detected in the Uniprot protein sequences, each mismatch error was characterized using nine different features. These features can be divided into three categories: (i) features suggesting that our mismatch identification protocol identified a real gene misprediction, (ii) features suggesting that our mismatch identification protocol identified a false positive mismatch (for example, a mismatch resulting from a wrong ortholog prediction or a MSA error), and (iii) features that were uncertain (not clear whether the detected mismatch corresponds to a gene misprediction or a false positive). Out of 11,015 mismatches, 7401 (67.2%) could be associated with one or more features, leaving one third of the mismatches so far unexplained (Table 3).

Almost half of the potential mismatches (5446 mismatches, 49.4%) were associated uniquely with evidence of a gene misprediction error (details are provided in Additional file 1). The most frequent feature associated with misprediction is the presence of undefined genomic regions (represented by N characters) in introns or exons, which were found in 47.7% of all mismatches. Introns shorter than the minimum length of 30 nucleotides required for a functional intron [30], are found in 8.5% of all mismatches. The presence of short introns is also sometimes linked to the presence of several small primate exons, corresponding to a single exon in the human gene (5.5%). Figure 5a shows an example of a badly predicted sequence from *G. gorilla* (G3S2R3_GORGO), which has an exon in the wrong frame compared to the human gene and an intron of length 21 nucleotides. A deletion of 2 bases compared to the exon of the human reference sequence A7F2E4_HUMAN (golgin A8 family member A) is probably due to a genome sequencing error. The sequence could be improved by creating

Table 3 Characterization of 11,015 mismatched sequence segments in primate sequences, according to nine different features

Class	Feature	No. (%) of errors
Evidence of gene prediction error	Genomic sequence contains N characters (introns or exons)	5256 (47.7%)
	Primate sequence contains short introns (< 30 nucleotides)	937 (8.5%)
	1 Human exon aligned with ≥ 3 primate exons	611 (5.5%)
	Non-canonical splice sites in human sequence	237 (2.2%)
	Frameshift in primate exon sequence	138 (1.3%)
Evidence of false positive error	Human isoform exists that matches primate sequence	1194 (10.8%)
	Multiple alignment error	244 (2.2%)
	In a repeated protein region	232 (2.1%)
Mixed evidence	Mismatch associated with evidence of both gene prediction error and false positive error	341 (3.1%)
Unconfirmed	Conserved in ≥ 4 primates	1054 (9.6%)
	Mismatch associated with evidence of gene prediction error only	5446 (49.4%)
	Mismatch associated with evidence of false positive error only	4174 (37.9%)
	Mismatch associated with at least 1 feature	7401 (67.2%)
	Mismatch associated with 0 features	3614 (32.8%)

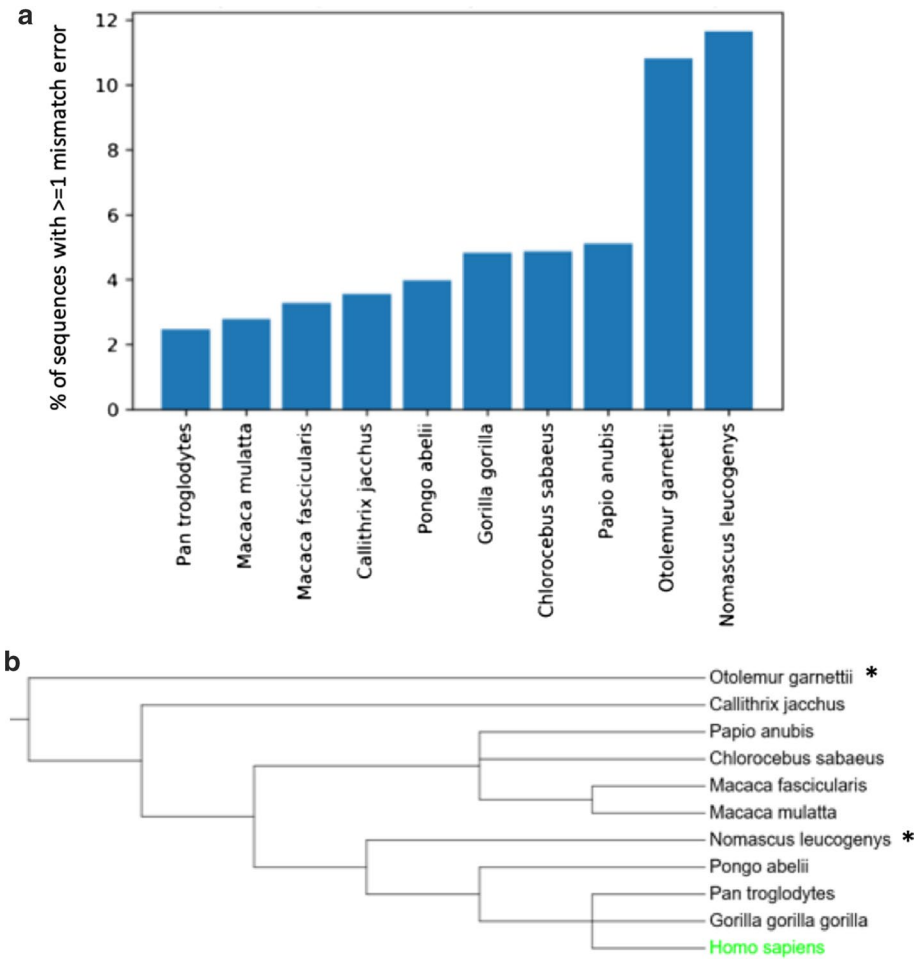


Fig. 4 Comparison of primate sequence error rates and evolutionary distance from the human reference proteome. **a** Percentage of sequences with at least one mismatched segment for each primate. **b** Phylogenetic tree of the 11 primates included in the study, including the reference *Homo sapiens*. Asterisks indicate the species with high protein sequence error rates

an alternative longer exon in the correct reading frame, although this means introducing a short intron of only 7 nucleotides.

Other factors causing gene mispredictions include the presence of frameshifts in the primate exon sequences (1.3%) or the use of non-canonical splice sites in human sequences (2.2%). Figure 5b shows an example of a badly predicted primate sequence (A0A096NZ82_PAPAN) from *P. Anubis,* which is orthologous to the human sequence Q15858_HUMAN (Sodium channel protein type 9 subunit alpha) with non-canonical splice sites. A completely corrected primate sequence can be proposed if non-canonical splice sites are allowed in the gene model.

A number of false positive mismatches (37.9%) were found that could not be reliably classified as gene prediction errors. For 10.8% of all mismatches, an alternative human isoform could be found in the Ensembl database corresponding to the primate sequence. In this case, the mismatch error was probably caused by the definition of different canonical isoforms for human and primate sequences in the Uniprot database.

Table 4 Number of sequence errors identified before and after correction of 603 primate sequences

Error type	No. of errors before correction	No. of errors after correction	Difference
Internal insertion	340	132	−208
Internal deletion	816	785	−31
Mismatch	833	263	−570
N-terminal insertion	32	32	0
N-terminal deletion	80	90	+10
C-terminal insertion	29	26	−4
C-terminal deletion	54	52	−2
TOTAL	2184	1379	−805
Mean % identity	91.9%	95.0%	+3.1%
Mean % coverage	89.5%	90.7%	+1.2%

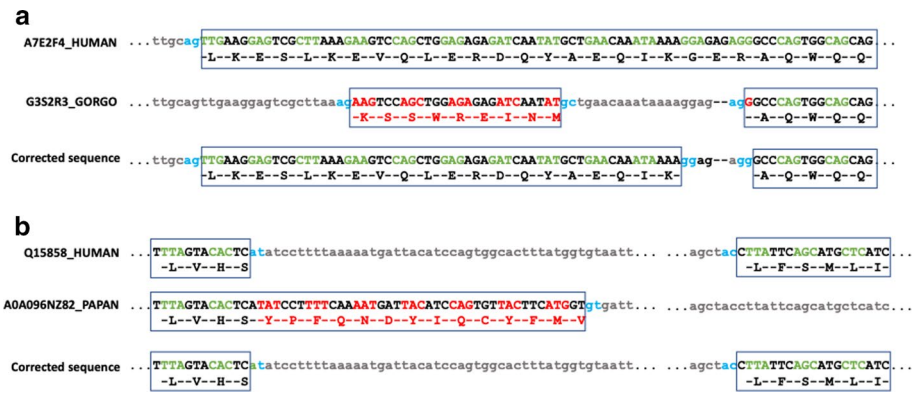


Fig. 5 Example mismatched segments corresponding to gene prediction errors. **a** A deletion of 2 bases in the primate sequence G3S2R3 results in misprediction of 5′ part of exon. **b** Non-canonical splice sites in the human sequence Q15858 and misprediction of the primate sequence A0A096NZ82. Black boxes represent exons with nucleotide sequences shown in upper case, and protein sequences shown below the translated exon sequence. For the correctly predicted exons, alternate codons are shown in green. For the badly predicted exons, alternate codons are shown in red. Intron sequences are in lower case, with splice sites in blue

Finally, 12.7% of the mismatches were associated with at least one feature, but the cause of the error remains unconfirmed. These unconfirmed cases were mainly due to the detection of the mismatch in four or more primates (9.6% of all mismatches), which suggests that either the same misprediction was propagated across multiple primates or the human protein sequence is in fact significantly different from the non-human primates and the mismatch is a false positive. The remaining 3.1% of mismatches were associated with both misprediction evidence and false positive evidence and were also classed as 'Unconfirmed'.

Correction of mismatch errors

For the primate sequences containing mismatches linked to gene misprediction, we then tried to correct the mismatch errors using the correction protocol defined in the Methods section. The protocol first identifies the human sequence segment corresponding to

the primate mismatch error and then uses the human segment to perform a TBLASTN search for a conserved segment in the primate genome. Out of 5446 mismatches associated with misprediction, refined sequences were proposed for 603 of them (~11%). The remaining mismatches could not be improved due to either a lack of significant TBLASTN hits, or issues with the integrity of the refined primate transcript.

To validate the proposed error corrections, the refined sequences were realigned with the human reference sequence and the error detection protocol was again performed on the full-length protein sequences. The number of sequence errors detected for the original and the refined proteins are shown in Table 4. A significant decrease in gene prediction errors can be observed, especially for the number of mismatched segments as might be expected. For the 603 proteins, 833 mismatch errors were detected in the original sequences, while only 263 were detected after refinement, i.e. 570 mismatch errors were fully corrected. The remaining mismatches (263) could not be corrected for several reasons: the refined sequence contains false positive mismatch errors due to MSA errors for example, or the correction protocol badly predicted the new sequence.

Interestingly, we also observed a decrease in the number of internal insertion errors, since the mismatch refinement could also affect nearby internal insertions. In contract, a small increase in the number of N-terminal deletions was observed, due to the new CDS prediction during the correction protocol. When the original start codon cannot be mapped onto the new transcript (for example, when the mismatch error occurs at the beginning of the protein), an alternative start codon is used, leading to possible N-terminal deletion errors. Taking into account all the error types, the correction protocol leads to a decrease of 37% in the number of detected sequence errors.

We then estimated the quality of the refined primate sequences, by calculating the percent sequence identity and coverage for the original and refined primate sequences, compared to the human reference. On average, a small increase in both scores is observed after refinement, of +3.1% and +1.2% respectively (Table 4). However, the differences between the scores before and after correction between are not the same for the 603 sequences, as shown in Fig. 6. Some sequences had an increase in % sequence identity of >50% and in coverage of >45%, indicating the correction of very long mismatch errors. In contrast, a decrease in coverage was observed for a small number of sequences. This is probably be due to over-alignment of the mismatch-containing sequence, since the mismatched segment is aligned to the human reference even when it is not homologous, artificially boosting the original coverage score.

For most of the primates, higher quality sequences were proposed for between 20 and 80 proteins, while for *N. leucogenys*, 262 sequences were improved, representing 43% of the 603 corrections. This is not surprising since this primate is closely related to humans, and had the highest percentage of sequences with mismatch errors (Fig. 4).

Discussion

DNA and protein sequence databases are increasingly useful research tools, but to maximize their potential, the errors in them need to be addressed. Contrary to expectations, advances in genome sequencing technologies have led to an increased number of protein sequence errors in public databases, including incomplete, incorrect or inconsistent

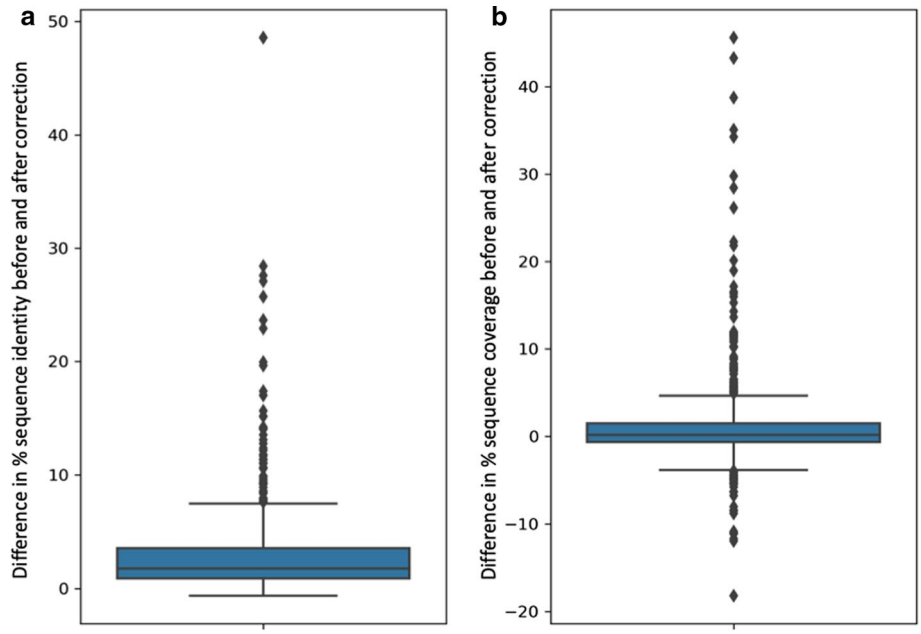


Fig. 6 Boxplots of the difference before and after correction for **a** percent sequence identity and **b** percent sequence coverage, for each of the 603 primate protein sequences

sequences. Here, we used MSAs of closely related sequences to detect potential errors in primate proteins and investigate the underlying causes.

We used a simple protocol based on BLASTP searches (using very strict thresholds to limit the number of false positives) to identify potential orthologs of human proteins in ten primate proteomes. This method was chosen because it is time-efficient, convenient and allows identification of sequences in the most recent protein databases. More accurate tools for orthology detection have been developed, such as eggNOG [31], OrthoInspector [32], or OrthoFinder [33]. However, these methods are computationally costly and are not always updated with each release of the Uniprot, Ensembl or RefSeq databases. The strict thresholds used in our protocol mean that we eliminated many primate orthologs, for example for *Otolemur garnettii*, orthologs were identified for only 69% of the human reference sequences (Table 1). In the future, it would be useful to incorporate other methods, such as Reciprocal Blast Hits, in order to extend the method to allow analysis of more orthologs, as well as to other groups of organisms. Nevertheless, it is important to point out that our protocol relies on the presence of a well-studied organism with characterized proteins that can be used as a reference.

We then chose to focus our analyses on the sequence errors involving mismatched segments, since mismatch errors can have a serious effect on downstream analyses. Mismatched segments generally arise when an exon is translated in the wrong reading frame, or the exon boundaries are badly predicted and a non-coding nucleotide sequence is translated instead. To ensure that the mismatched segments detected by our protocol were indeed caused by such errors in gene prediction, we eliminated the mismatch errors that were associated with the possible identification of primate paralogs instead of orthologs, or misalignments during the MSA construction process. This resulted in a total of 11,015 mismatch errors in Uniprot protein sequences, and 5971 mismatch errors

in RefSeq proteins. The difference in the number of mismatches can be partly explained by the fact that, in the Uniprot reference proteomes only one 'canonical' protein is provided for each gene, while in RefSeq all protein isoforms are listed, reducing the number of false positive mismatches due to the definition of alternative isoforms (with different exon–intron structures) for different species.

By mapping the Uniprot proteins to their genomic sequences in the Ensembl database, we investigated the underlying reasons for the predicted errors. Unfortunately, the extraction of the same information for the RefSeq sequences could not be fully automated, but we intend to address this issue in the future. Almost half (47.7%) of the 11,015 mismatch errors in the Uniprot sequences could be attributed to the presence of undetermined regions in the genomic sequences (represented by 'N' characters), causing misprediction of the gene exon–intron structures. Surprisingly, the undetermined region was not always located within the mismatched exon, and in these cases our error correction protocol was able to identify the correct exon sequence and improve the quality of the translated protein sequence.

Many of the remaining mismatch errors could be linked to other issues in the primate genome sequencing and/or assembly processes. For example, a large number of primate proteins (937) had abnormal introns of length < 30 nucleotides. In Human, it has been suggested previously that the minimal length for an intron to be processed is 30 nucleotides [30]. A number of mismatches were also linked to the presence of insertions or deletions of a small number of nucleotides (non-multiple of 3) within the exon sequence, meaning that part of the protein was predicted in the wrong frame or several short exons were predicted corresponding to one human exon. These results taken together clearly suggest that more robust gene prediction algorithms are needed that incorporate strategies for genome error detection or quality control.

Finally, a number of mismatch errors (237) were associated with the presence of non-canonical splice sites in the human reference gene, suggesting that the gene model used in the original primate genome annotation was incomplete and might be improved by taking into account non-canonical splice sites.

To test whether strategies could be developed to correct gene prediction errors, we implemented a simple error correction protocol as a proof-of-concept. A total of 603 protein sequences were refined and led to a significant reduction in the number of sequence errors, from 2184 errors before correction to 1379 errors after correction. The protocol involves the insertion of new exons in the primate transcripts, although the new exons are not guaranteed to have canonical splicing sites. This was done in order to produce a more accurate protein sequence, at the expense of transcript logic where splicing rules are not respected. This is only a first step in the improvement of gene prediction algorithms, but it demonstrates that the available primate genomes could be exploited to correct the primate proteins in public sequence databases.

Conclusions

To conclude, by comparing ten primate proteomes with the human reference proteome, we showed that mismatch errors are frequent in primate proteomes, where the percentage of sequences having a mismatch error ranges from 2 to 12% depending on the primate species. Based on the features we identified, the reasons causing two thirds of

these mismatches could be identified. Half of all potential mismatches were associated with evidence of gene misprediction. Interestingly, we showed that the existing primate genomes contain signals allowing improvement of these sequences.

The error detection protocol used here could be improved by the addition of new characterization features to understand the reasons for the detected mismatch errors that were not explained in our analyses, and to highlight the unexplained errors that may in fact be biological deviations representing organism specificities or innovations. It would also be interesting to extend our quality control protocol to other misprediction error types, such as insertions or deletions, as well as to the study of genomes from other species.

We hope that the results of this work will provide useful information for the development of more reliable gene prediction algorithms, as well as for downstream analyses that rely on high quality protein sequences. For example, the analysis protocol is currently being integrated in a new variant prediction system, called MISTIC [34], which uses the information in MSAs to predict the pathogenicity of human genome variants.

Methods

Human reference protein sequences

Human protein sequences are well studied and for the purposes of this study, are assumed to be accurate. The *Homo sapiens* reference proteome (UP000005640) from Uniprot (downloaded in Dec. 2019) was used for the reference protein sequences and we selected the canonical isoform for each human gene. A total of 20,596 human protein sequences, annotated as 'Reviewed' in the Uniprot database, were extracted.

Orthologous sequences from primate proteomes

Potential orthology relationships between human proteins and all (Reviewed and Unreviewed) proteins from ten primates (*Pan Troglodytes, Gorilla gorilla gorilla, Macaca mulatta, Macaca fascicularis, Chlorocebus Sabaeus, Papio Anubis, Pongo abelii, Nomascus leucogenys, Callithrix jacchus, Otolemur garnettii*) were predicted using a simple protocol based on BLASTP [35] searches in the Uniprot reference proteome database. For each human protein sequence, the best hit in each of the ten primates was considered as an orthologous protein if it has an e-value $< 10^{-50}$ and sequence identity > 80%. Table 1 shows details of the primate proteomes used and the number of predicted orthologous sequences.

Orthologous sequences were also identified using the same protocol in the proteomes corresponding to the ten primates in the RefSeq database (release 98). In this case, protein isoforms of the same gene are not grouped in the same database entry, we retained all protein sequences identified in the BLASTP searches with the same thresholds as above, i.e. an e-value $< 10^{-50}$ and sequence identity > 80%.

Multiple alignments and detection of sequence errors

For each of the 20,596 human proteins, a multiple sequence alignment (MSA) of the primate orthologous sequences was constructed using the MAFFT program [36]. Each of MSA was analyzed to detect potential insertions, deletions and mismatches in the orthologous sequences using the SIBIS program [37]. SIBIS uses a Bayesian framework

combined with Dirichlet mixture models to identify inconsistent sequence segments representing potential sequence errors. The Bayesian approach provides a strong theoretical foundation for modeling the amino acid frequencies found at a specific alignment position. SIBIS combines a prior distribution of amino acid probabilities, with observed amino acid frequencies at homologous positions within the related proteins, in order to calculate scores for new sequences. The predicted errors are classified into seven categories: N-terminal deletion, N-terminal extension, C-terminal deletion, C-terminal extension, internal deletion, internal insertion and mismatched segment.

For this study, we focused on potential sequence mismatches, where a mismatch was defined as a sequence segment that is at least 20 amino acids long with less than 50% sequence identity between the human reference and the primate sequences. In addition, to exclude false positive mismatch errors due to wrong ortholog detection or MSA errors, mismatching sequences in primates that were significantly longer or shorter than the human sequence (primate sequence length three times longer or shorter than human) were excluded.

Phylogenetic trees

Phylogenetic relations between organisms were determined using the LifeMap tool [38] by extracting a subtree based on the list of organisms of interest. The subtree was visualized using iTOL [39] generate the tree figures.

Intron–exon maps and genomic sequences

To perform more in-depth analysis of mismatch errors, Uniprot protein sequences were mapped to their transcript IDs in the Ensembl database (release 99) using the Uniprot Retrieve/ID mapping tool. For each transcript, the CDS and genomic sequence were retrieved, as well as intron and exon positions on the genomic sequence.

Mismatch characterization

For each potential mismatch error detected by SIBIS, the mismatch sequence segment was located on the exon/intron map, CDS and genomic region. To identify the potential causes of the mismatch error, nine features were defined and classified into three categories (Gene misprediction, False positive, Undetermined) as follows:

Evidence of a Gene prediction error involves five possible features:

iA low quality genomic sequence, containing 'N' characters indicating undetermined or ambiguous nucleotides (IUPAC nomenclature) probably caused by genome sequencing errors or assembly gaps,

j Short introns in the primate sequence (< 30 nucleotides), since in Human, it has been shown previously that the minimal length for an intron to be processed is 30 nucleotides [30],

k A mismatch coded by a single exon in human compared to > 3 exons in the primate sequence,

l Non-canonical splice sites (not GT/AG) in the human mismatch sequence,

m The presence of a small nucleotide insertion in the primate exon sequence causing a frameshift.

Evidence that the detected mismatch is in fact a False positive (i.e. the mismatch is due to inaccuracies in the quality control protocol) involves three possible features:

i The existence of alternative human isoform in the Ensembl database that does not Generate a mismatch at that position,
j MSA alignment errors,
k Mismatches in a sequence region with repeats.

The mismatch was characterized as Undetermined if one feature was detected, namely that the same mismatch is identified in four or more primates in the MSA, or a mixture of Gene misprediction and False positive evidence was identified.

Mismatch error correction protocol

For the mismatch errors classified as gene mispredictions, a simple protocol was implemented to try to improve the quality of the protein sequence. Given a primate protein sequence with a mismatch error compared to the human sequence, the error correction protocol involves four steps (outlined in Fig. 2) as follows:

1 Using the human sequence segment as a query, perform a TBLASTN search in the primate genomic region corresponding to the transcript.
2 If a hit is found with an e-value < 0.001 and sequence identity > 80%, it is inserted into the primate transcript as a new exon.
3 Verify the integrity of the new transcript, by checking for overlapping or inverted exons, frameshifts and stop codons in the inserted sequence.
4 Create a new CDS and protein sequence.

To estimate the quality of the new protein sequence, the modified primate sequence was realigned with the human reference protein. Two scores were used to evaluate the quality of the primate sequences:

(i) Percent sequence identity I was defined as $I = i/N \times 100$, where i is the number of identical amino acids in the alignment and N is the total number of non-gap positions in the alignment.
(ii) Percent sequence coverage C was defined as $C = N/N_{tot} \times 100$, where N is the number of non-gap positions in the alignment and N_{tot} is the total length of the alignment.

Implementation

The analyses were performed using an Sqlite3 database, in-house bash and Python (version 2.7) scripts, and numpy (numpy.org), pandas (pandas.pydata.org) and matplotlib (matplotlib.org) Python libraries. The Biopython (biopython.org) package was used to perform pairwise alignments during the mismatch characterization step. To retrieve large amounts of data from the Ensembl database, the grequests (pypi.org) package was used to perform asynchronous application programming interface (API) requests. The Json (www.json.org) library was used to process Ensembl API output.

Supplementary information

Supplementary information accompanies this paper at https://doi.org/10.1186/s12859-020-03855-1.

Additional file 1: Characterization of the 5446 mismatch errors associated with gene mispredictions.

Acknowledgements
The authors would like to thank the BigEst and AICS Platforms for their assistance.

Authors' contributions
CM developed the analysis protocols, and CM and NS analyzed the data regarding gene prediction errors. AJG and PC advised on characterization features of the datasets and supervised the analyses. OP and JDT supervised the production of the datasets and validated the results. All authors participated in the definition of the original study concept. All authors read and approved the final manuscript.

Funding
This work was supported by the Institut Français de Bioinformatique (IFB), ANR-11-INBS-0013, the ANR projects Elixir-Excelerate: GA-676559 and RAinRARE: ANR-18-RAR3-0006-02, and Institute funds from the French Centre National de la Recherche Scientifique, the University of Strasbourg. The funding bodies played no role in the design of the study and collection, analysis, and interpretation of data and in writing the manuscript.

Availability of data and materials
All sequences used in this study are available in the public Uniprot, RefSeq and Ensembl databases. The results of the characterization of the mismatch errors associated with gene mispredictions are available in Additional file 1.

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Received: 28 July 2020 Accepted: 30 October 2020
Published online: 10 November 2020

References

1. Mudge JM, Harrow J. The state of play in higher eukaryote gene annotation. Nat Rev Genet. 2016;17:758–72.
2. Danchin A, Ouzounis C, Tokuyasu T, Zucker J-D. No wisdom in the crowd: genome annotation in the era of big data-current status and future prospects. Microb Biotechnol. 2018;11:588–605.
3. Pertea M, Shumate A, Pertea G, Varabyou A, Breitwieser FP, Chang YC, Madugundu AK, Pandey A, Salzberg SL. Genome Biol. 2018;19:208.
4. Alliance of Genome Resources Consortium. The alliance of genome resources: building a modern data ecosystem for model organism databases. Genetics. 2019;213:1189–96.
5. Zahn-Zabal M, Michel PA, Gateau A, et al. The neXtProt knowledgebase in 2020: data, tools and usability improvements. Nucleic Acids Res. 2020;48(D1):D328–34.
6. Norgren RB. Improving genome assemblies and annotations for nonhuman primates. ILAR J. 2013;54:144–53.
7. Bick JT, Zeng S, Robinson MD, Ulbrich SE, Bauersachs S. Mammalian annotation database for improved annotation and functional classification of omics datasets from less well-annotated organisms. Database (Oxford). 2019;baz086.
8. Hart AJ, Ginzburg S, Xu MS, et al. EnTAP: bringing faster and smarter functional annotation to non-model eukaryotic transcriptomes. Mol Ecol Resour. 2019. https://doi.org/10.1111/1755-0998.13106.
9. UniProt Consortium. UniProt: a worldwide hub of protein knowledge. Nucleic Acids Res. 2019;47:D506–15.
10. O'Leary NA, Wright MW, Brister JR, et al. Reference sequence (RefSeq) database at NCBI: current status, taxonomic expansion, and functional annotation. Nucleic Acids Res. 2016;44:D733–45.
11. Yates AD, Achuthan P, Akanni W, et al. Ensembl 2020. Nucleic Acids Res. 2020;48(D1):D682–8.
12. Salzberg SL. Next-generation genome annotation: we still struggle to get it right. Genome Biol. 2019;20:s13059-019-1715-2.
13. Tørresen OK, Star B, Mier P, Andrade-Navarro MA, Bateman A, Jarnot P, Gruca A, Grynberg M, Kajava AV, Promponas VJ, Anisimova M, Jakobsen KS, Linke D. Tandem repeats lead to sequence assembly errors and impose multi-level challenges for genome and protein databases. Nucleic Acids Res. 2019;47:10994–1006.
14. Goodswen SJ, Kennedy PJ, Ellis JT. Evaluating high-throughput ab initio gene finders to discover proteins encoded in eukaryotic pathogen genomes missed by laboratory techniques. PLoS ONE. 2012;7:e50609.
15. Denton JF, Lugo-Martinez J, Tucker AE, Schrider DR, Warren WC, Hahn MW. Extensive error in the number of genes inferred from draft genome assemblies. PLoS Comput Biol. 2014;10:e1003998.
16. Guigó R, Agarwal P, Abril JF, Burset M, Fickett JW. An assessment of gene prediction accuracy in large DNA sequences. Genome Res. 2000;10:1631–42.
17. Drăgan M-A, Moghul I, Priyam A, Bustos C, Wurm Y. GeneValidator: identify problems with protein-coding gene predictions. Bioinformatics. 2016;btw015.

18. Guigó R, Flicek P, Abril JF, Reymond A, Lagarde J, Denoeud F, et al. EGASP: the human ENCODE genome annotation assessment project. Genome Biol. 2006;7:S2.
19. Prosdocimi F, Linard B, Pontarotti P, Poch O, Thompson JD. Controversies in modern evolutionary biology: the imperative for error detection and quality control. BMC Genom. 2012;13:5.
20. Deutekom ES, Vosseberg J, van Dam TJP, Snel B. Measuring the impact of gene prediction on gene loss estimates in Eukaryotes by quantifying falsely inferred absences. PLoS Comput Biol. 2019;15:e1007301.
21. Hadley C. Righting the wrongs. EMBO Rep. 2003;4:829–31.
22. Söllner JF, Leparc G, Zwick M, et al. Exploiting orthology and de novo transcriptome assembly to refine target sequence information. BMC Med Genom. 2019;12:69.
23. Schnoes AM, Brown SD, Dodevski I, Babbitt PC. Annotation error in public databases: misannotation of molecular function in enzyme superfamilies. PLoS Comput Biol. 2009;5(12):e1000605.
24. Bouadjenek MR, Verspoor K, Zobel J. Literature consistency of bioinformatics sequence databases is effective for assessing record quality. Database (Oxford). 2017:bax021.
25. Nagy A, Patthy L. FixPred: a resource for correction of erroneous protein sequences. Database (Oxford). 2014;bau032.
26. Vanhoutreve R, Kress A, Legrand B, Gass H, Poch O, Thompson JD. LEON-BIS: multiple alignment evaluation of sequence neighbours using a Bayesian inference system. BMC Bioinform. 2016;17:271.
27. Dunne MP, Kelly S. OMGene: mutual improvement of gene models through optimisation of evolutionary conservation. BMC Genom. 2018;19:307.
28. Venturini L, Caim S, Kaithakottil GG, Mapleson DL, Swarbreck D. Leveraging multiple transcriptome assembly methods for improved gene structure annotation. GigaScience. 2018;7:1–15.
29. Scalzitti N, Jeannin-Girardon A, Collet P, Poch O, Thompson JD. A benchmark study of ab initio gene prediction methods in diverse eukaryotic organisms. BMC Genom. 2020;21:293.
30. Piovesan A, Caracausi M, Ricci M, Strippoli P, Vitale L, Pelleri MC. Identification of minimal eukaryotic introns through GeneBase, a user-friendly tool for parsing the NCBI Gene databank. DNA Res. 2015;22:495–503.
31. Huerta-Cepas J, Szklarczyk D, Heller D, Hernández-Plaza A, Forslund SK, Cook H, Mende DR, Letunic I, Rattei T, Jensen LJ, von Mering C, Bork P. eggNOG 5.0: a hierarchical, functionally and phylogenetically annotated orthology resource based on 5090 organisms and 2502 viruses. Nucleic Acids Res. 2019;47:D309–14.
32. Nevers Y, Kress A, Defosset A, Ripp R, Linard B, Thompson JD, Poch O, Lecompte O. OrthoInspector 3.0: open portal for comparative genomics. Nucleic Acids Res. 2019;47:D411–8.
33. Emms DM, Kelly S. OrthoFinder: phylogenetic orthology inference for comparative genomics. Genome Biol. 2019;20:238.
34. Chennen K, Weber T, Lornage X, Kress A, Böhm J, Thompson JD, Laporte J, Poch O. MISTIC: a prediction tool to reveal disease-relevant deleterious missense variants. PLoS One (in press).
35. Camacho C, Coulouris G, Avagyan V, Ma N, Papadopoulos J, Bealer K, Madden TL. BLAST+: architecture and applications. BMC Bioinform. 2009;10:421.
36. Nakamura T, Yamada KD, Tomii K, Katoh K. Parallelization of MAFFT for large-scale multiple sequence alignments. Bioinformatics. 2018;34:2490–2.
37. Khenoussi W, Vanhoutrève R, Poch O, Thompson JD. SIBIS: a Bayesian model for inconsistent protein sequence estimation. Bioinformatics. 2014;30:2432–9.
38. de Vienne DM. Lifemap: exploring the entire tree of life. PLoS Biol. 2016;14:e2001624.
39. Letunic I, Bork P. Interactive tree of life (iTOL) v4: recent updates and new developments. Nucleic Acids Res. 2019;47(W1):W256–9.

Publisher's Note

A partir de ces résultats, ainsi que de l'analyse comparative initiée à l'aide de G3PO, il était devenu évident qu'un effort supplémentaire doit être fait pour l'annotation des génomes eucaryotes et notamment des gènes codant pour des protéines. Ainsi, de nouveaux outils de prédiction, plus performant, doivent être développés en s'appuyant sur la masse de données disponible et en tenant compte de leur qualité.

Chapitre 7 – Spliceator : la revanche des sites

"Don't Blame Me! I'm An Interpreter. I'm Not Supposed To Know A Power Socket From A Computer Terminal"

- C-3PO -

7.1 La prédiction *ab initio* des sites d'épissage

Il y a 25 ans, quand l'annotation des génomes était à son commencement, la stratégie des programmes *ab initio* de prédiction des gènes codant pour des protéines était de localiser le maximum d'éléments fonctionnels comme le promoteur ou les SE, puis de construire un modèle d'architecture de gènes probable basée sur des statistiques. Aujourd'hui, les programmes tendent à se spécialiser afin d'effectuer une seule tâche, mais de manière beaucoup plus robuste, précise et fiable. L'intégration de plusieurs programmes dans un pipeline complet d'annotation des gènes codants pour des protéines semble ainsi être une meilleure stratégie. Dans ce contexte, mes travaux de thèse se sont principalement axés sur la détection des SE par des approches *ab initio*.

La prédiction des SE a également commencé vers les années 90, où des premiers travaux basés sur des réseaux de neurones ont été effectués (Moody *et al.*, 1991; Brunak *et al.*, 1991; Reese *et al.*, 1997). D'autres programmes ont ensuite exploité des approches probabilistes comme le principe d'entropie maximale (Yeo and Burge, 2004), les chaînes de Markov (Zhang *et al.*, 2010; Pashaei, Yilmaz, *et al.*, 2016) ou encore les réseaux bayésiens (D. Cai *et al.*, 2000; Castelo and Guigó, 2004; Chen *et al.*, 2005). Les algorithmes de *machine learning* sont aussi utilisés, comme les SVM (Sun *et al.*, 2003; Degroeve *et al.*, 2005; Sonnenburg *et al.*, 2007; Wei *et al.*, 2013; Bari *et al.*, 2013; Maji and Garg, 2014; Li *et al.*, 2017), les arbres de décision (Pashaei, Ozen, *et al.*, 2016; Meher *et al.*, 2016) ou des approches d'évolution artificielle (Nasibov and Tunaboylu, 2010; Kamath *et al.*, 2012). Plus récemment, le *deep learning* est

venu renforcer la prédiction des SE (Lee *et al.*, 2015; Naito, 2018; Bretschneider *et al.*, 2018; Zuallaert *et al.*, 2018; Wang *et al.*, 2019; Jaganathan *et al.*, 2019).

A l'heure actuelle, la prédiction *ab initio* des SE est encore un problème majeur pour l'annotation des gènes codant pour des protéines. Le nombre de dinucléotides GT/AG n'étant pas impliqué dans les SE est astronomique, et on estime que seul 0,1% de ces dinucléotides seraient de vrais SE (Sonnenburg *et al.*, 2007). Ainsi, bien que le RNA-seq semble ouvrir une nouvelle voie vers l'amélioration de la prédiction des SE, il semble encore indispensable de développer des outils pour identifier ces SE.

Sur la base du code d'épissage, les chercheurs ont développé des outils capables de détecter des signaux afin de prédire avec la plus grande précision les SE et donc la structure interne des gènes codant pour des protéines. Malheureusement, certains problèmes persistent comme :

- identifier sur la base de signaux *cis*, les vrais SE et éviter la prédiction de faux positifs,
- identifier les SE non-canoniques,
- prédire les isoformes biologiquement vrais,
- identifier les variants d'épissage, notamment ceux impliqués dans des pathologies.

7.2 Spliceator

Afin de répondre à certaines de ces problématiques, je me suis concentré sur le développement d'un nouvel outil, appelé Spliceator, pour la prédiction de SE. Spliceator présente quatre atouts majeurs :

- utilisation d'un des algorithmes de *deep learning* les plus performants : le réseau de neurones convolutif,
- entraînement sur un jeu de données de haute qualité avec des données validées manuellement : G3PO+,
- modèle multi-espèces : plus de 100 organismes,
- utilisation facile.

7.2.1 Architecture du CNN de Spliceator

Spliceator est un programme d'intelligence artificielle. Il exploite un réseau de neurones convolutif, un des algorithmes de *deep learning* les plus performants pour la reconnaissance de

motifs locaux. L'architecture du CNN de Spliceator est peu profonde et contient seulement 3 couches de convolution, car le modèle généralise moins bien les données lorsqu'il est trop profond (figure 56). L'architecture complète est présentée dans la publication, voir 7.2.5. Deux modèles ont été entraînés séparément, un pour prédire les SE donneurs et un autre pour prédire les SE accepteurs, afin de les rendre plus spécifiques. J'ai utilisé le *framework* Tensorflow et la librairie Keras pour implémenter le réseau de neurones.

Les entrées du CNN sont des séquences génomiques de taille variable (allant de 20 à 600 nucléotides en fonction du modèle), encodées par un système *one-hot* à 5 éléments : A=(1,0,0,0), C=(0,1,0,0), G=(0,0,1,0), T=(0,0,0,1) et N=(0,0,0,0). La dernière couche du réseau est composée de deux neurones pour définir les deux catégories : 0 si la donnée d'entrée est classée comme non-SE et 1 si la donnée d'entrée est classée comme SE. La fonction d'activation est la fonction *softmax* afin de fournir une probabilité pour chaque classe.

	Accuracy		Sur-apprentissage	
	Donneur	**Accepteur**	**Donneur**	**Accepteur**
1 couche	96,7	95,5	+	+
2 couches	95,6	94,9	+/-	+/-
3 couches	95,09	94,41	-	-
4 couches	92,72	93	-	-
6 couches	91,2	91,4	+/-	-
3 couches sans *dropout*	97	95,6	+	+
3 couches sans *pooling*	95,5	94,3	+	+

Figure 56 - *Évaluation des performances des modèles de Spliceator sur différentes architectures de CNN. Plus l'architecture se complexifie en accumulant des couches de convolution, plus l'accuracy diminue. Les régions jaunes représentent les architectures où un phénomène de sur-apprentissage a lieu. Ainsi, l'architecture 3 a été notre meilleure option. Les architectures sans dropout ou pooling sont également favorables à un événement de sur-apprentissage. (+signifie la présence de sur-apprentissage, - indique qu'il n'y en a pas et +/- signifie qu'un évènement de sur-apprentissage a peut-être eu lieu.*

7.2.2 Apprentissage des modèles

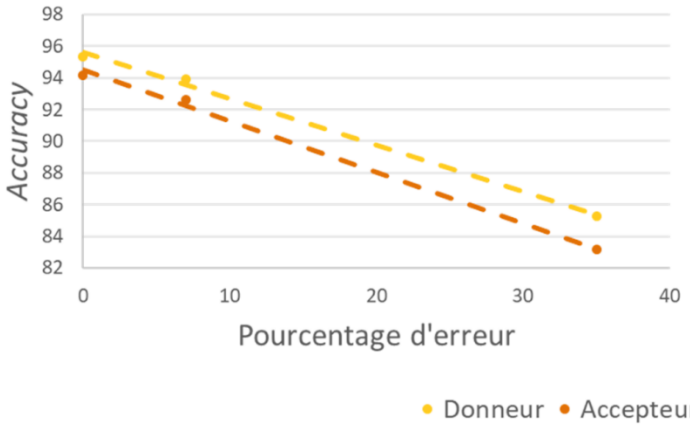
7.2.2.1 Apprentissage via des données de haute qualité

Comme tous les algorithmes de *deep learning*, les CNN s'entraînent sur des données. Ces dernières vont guider l'apprentissage du modèle, par conséquent, leur cohérence avec la problématique et leur qualité impactent directement les performances des modèles. De nombreux programmes de prédiction de SE sont aujourd'hui basés sur des algorithmes de *deep learning*. Cependant, la majorité de ces programmes ont des modèles espèce-spécifiques. Hormis les données humaines, qui sont considérées comme de haute qualité, la majeure partie des autres organismes, même modèles, inclut des données erronées comme nous l'avons vu précédemment. Par conséquent, si un modèle s'entraîne avec un jeu de données contenant des erreurs, il peut être fortement impacté. On parle également du concept "*garbage-in, garbage-out*" (Kilkenny and Robinson, 2018).

Spliceator outrepasse cette problématique, car nous avons utilisé G3PO+ qui inclut spécifiquement des données de haute qualité, pour entraîner les modèles. Les régions incluant un SE (donneur et accepteur) ont été extraites de chaque séquence "*Confirmed*". Au total 10 973 régions avec un SE donneur et 11 179 régions avec un SE accepteur ont été extraites formant ainsi les sets positifs (appelé GS = "*Gold Standard")* du jeu d'entraînement de Spliceator. Les sets négatifs ont été formés par l'extraction de régions de gènes sans SE. Le protocole de construction des jeux de données est présenté en détail dans la publication.

En s'entraînant sur des données robustes, le modèle gagne en performance. En effet, nous avons évalué les performances de modèles entraînés avec un jeu incluant peu de séquences erronées (appelé AS = "*All Sequences*"), un autre avec un nombre conséquent de séquences erronées (appelé PEAR = "*Proteic Elements with Aberrant Regions*") et le jeu GS incluant uniquement des séquences de bonne qualité. Nous avons estimé que le pourcentage de séquences erronées était proche de ~7% dans le jeu AS et de ~30% dans le jeu PEAR. Les performances des modèles donneur et accepteur sont répertoriées dans la figure 57. Les droites représentent les courbes de tendance pour le SE donneur (jaune) et accepteur (orange). On observe une forte corrélation entre le pourcentage d'erreurs et la performance des modèles (R^2 donneur et accepteur = 99,7). Ainsi, plus le nombre de séquences erronées est important, plus les performances diminuent. Il est donc indispensable de limiter au maximum le nombre de données erronées au sein du jeu d'entraînement afin de réduire l'effet "*garbage-in, garbage-out*". Finalement, en exploitant un jeu de données de qualité (GS), nous avons augmenté les

performances d'environ 1,5% par rapport au jeu AS et de plus de 8% par rapport au jeu PEAR. Cette stratégie a donc porté ses fruits.



Jeux	% Erreur	Donneur	Accepteur
GS	0	95,34	94,19
AS	7	93,9	92,62
PEAR	35	85,3	83,15

Figure 57 - *Performances des modèles de Spliceator en fonction du pourcentage de séquences erronées dans le jeu d'entraînement. Les droites de régression mettent en évidence une forte corrélation entre la qualité des données et la performance des modèles.*

7.2.2.2 Vérification de l'apprentissage

Le sur-apprentissage est un des fléaux du *machine learning*, mais des méthodes ont été développées pour lutter contre, comme le *dropout*. Cependant, une architecture de réseau ou des données peu adaptées peuvent accentuer ce phénomène. Si le modèle sur-apprend, les performances seront impactées. Lors des expérimentations avec Spliceator, nous avons testé différents jeux de données afin de mettre en évidence des caractéristiques pouvant impacter ces performances. Une des caractéristiques que nous avons testées est le déséquilibre du jeu d'entraînement. En effet, la prédiction des SE est un problème déséquilibré, car il y a beaucoup plus de dinucléotides GT/AG qui ne sont pas impliqués dans les SE. Nous avons donc construit un jeu de données déséquilibré avec un ratio 1:10 (contenant seulement 10% de vrais SE). Lors de la phase d'entraînement des modèles, ces derniers obtenaient des performances très élevées (*accuracy* > 97%) sur le jeu de test. Cependant, lorsqu'ils ont été testés sur des *benchmarks* indépendants, les performances se sont effondrées. Ce phénomène s'est produit car les modèles ont réalisé du sur-apprentissage sur les données d'entraînement. En effet, en raison du déséquilibre, les modèles ont appris à reconnaître parfaitement les SE négatives, c'est-à-dire, la quasi-totalité d'un gène codant pour des protéines, sans prendre en compte les cas positifs.

Pour éviter le phénomène de sur-apprentissage, j'ai évalué l'apprentissage des deux modèles lors de chaque epoch à l'aide d'un jeu d'évaluation. L'ensemble des valeurs ont été répertoriées sur un graphique présenté sur la figure 58. D'après ces graphiques, on observe que les courbes d'erreur sur le jeu d'évaluation (rose) suivent la trajectoire des courbes d'erreur sur

le jeu d'entraînement (rouge). Il n'y a également pas de zone de fracture *i.e.* les performances sur le jeu d'entraînement baissent et celles sur le jeu de validation augmentent. Nous pouvons donc conclure que nos modèles finaux n'ont pas effectué de sur-apprentissage.

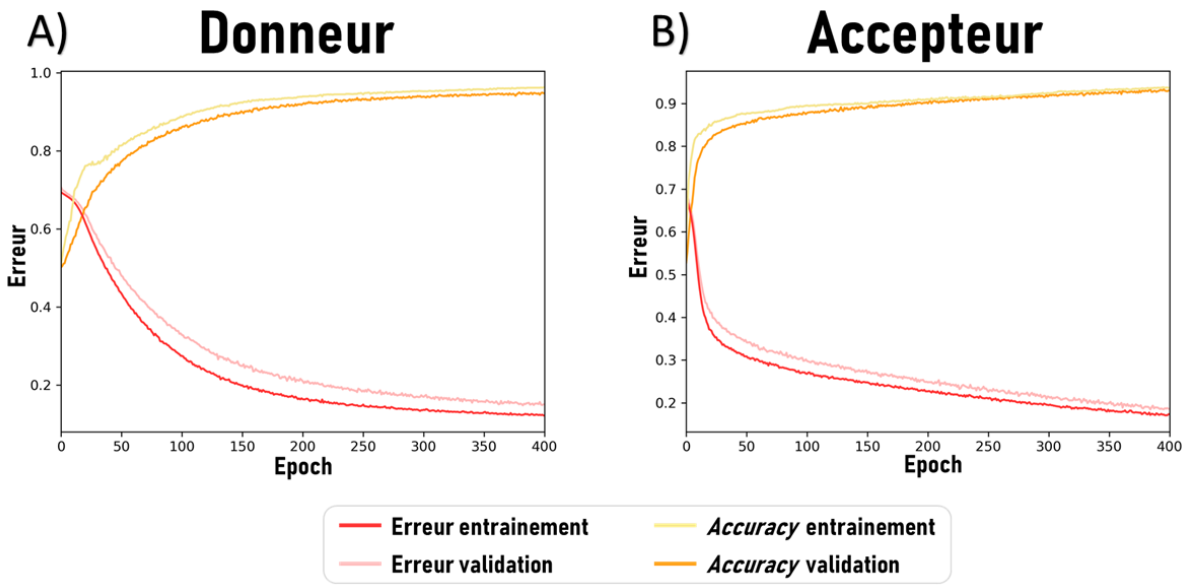


Figure 58 - *Courbe d'apprentissage du modèle du site d'épissage **A)** donneur et **B)** accepteur de Spliceator. Les courbes rose et orange représentent les valeurs obtenues sur le jeu d'évaluation. Elles permettent de confirmer que le modèle n'a pas réalisé de sur-apprentissage.*

7.2.3 Modèle universel

Comme G3PO+ inclut un grand nombre de séquences d'organismes différents et que Spliceator est entraîné sur ces données, il est de rigueur d'admettre qu'il s'agit d'un programme de prédiction de SE multi-espèces. En effet, plus de 100 organismes composent G3PO+ allant de l'homme aux protistes. Pour confirmer cette caractéristique originale, nous avons évalué les performances de Spliceator sur plusieurs *benchmarks* d'organismes différents : *D. Rerio* (poisson), *D. melanogaster* (drosophile), *C. elegans* (ver), *A. thaliana* (plante) et bien évidemment l'humain. En moyenne, Spliceator obtient 94,2% d'*accuracy* avec le modèle donneur ainsi qu'un F1-Score de 94,4%. Pour le modèle accepteur il obtient une *accuracy* de 90,1% et un F1-Score de 90,6%. Une des explications de l'écart de performance entre les deux modèles est probablement la plus grande variabilité qui existe au niveau du motif du SE accepteur.

Nous avons ensuite investigué plus en détail les performances de Spliceator sur un *benchmark* contenant différents protistes et viridiplantae. Spliceator obtient une accuracy de 86,0% et un F1-Score de 86,6% pour le modèle donneur et une accuracy de 83,5% et un F1-

Score de 83,6% pour le modèle accepteur. Ces résultats surpassent ceux des programmes de l'état-de-l'art (voir publication).

7.2.4 Utilisation de Spliceator

7.2.4.1 Site internet

Un site internet a été développé afin de faciliter l'accès et l'utilisation de Spliceator par des biologistes (lbgi.fr/spliceator). L'interface a été conçue pour que l'utilisateur ait très peu d'opérations à réaliser. En effet, l'utilisation la plus simple est de rentrer une séquence génomique, puis de lancer l'analyse en utilisant les paramètres par défaut. Il est également envisageable de modifier ces paramètres, comme la taille du contexte à prendre en compte, ou la fiabilité des résultats. Choisir une haute fiabilité réduirait le nombre de SE trouvés, mais les résultats seront plus précis, tandis que choisir une faible fiabilité permettrait de trouver plus de SE mais serait moins précis. Une valeur idéale de ce paramètre est aux alentours de 98%. En dessous de 90%, les résultats seront peu précis et génèrent trop de FP. Les résultats sont ensuite affichés de manière dynamique, afin d'optimiser leur visualisation. La figure 59 représente une capture d'écran de l'interface web de Spliceator.

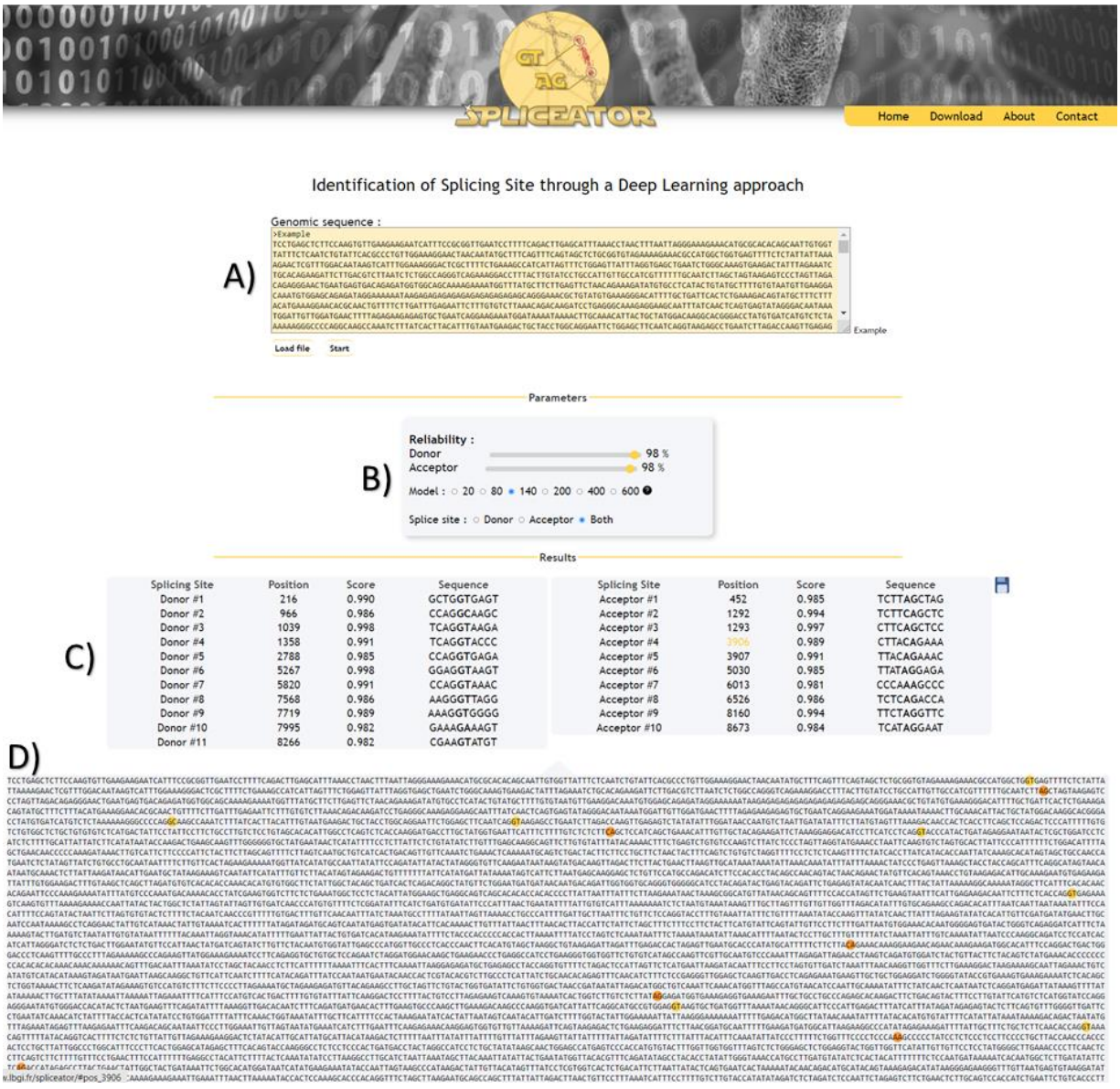


Figure 59 - *Capture d'écran de l'interface web de Spliceator. **A)** Zone d'insertion de la séquence génomique à analyser **B)** Choix des paramètres (fiabilité, choix du modèle en fonction de la taille du contexte et type de SE). **C)** Résultats sous forme de tableau, la position, le score de fiabilité et la séquence des SE identifiés sont représentés. Ces résultats peuvent être sauvegardés en cliquant sur la disquette à droite. **D)** Séquence annotée avec les SE identifiés (jaune = donneur et orange = accepteur).*

7.2.4.2 Programme en ligne de commande

Un programme en ligne de commande a également été développé. Il a été conçu afin d'être plus efficace et adapté aux analyses haut débit. Il est disponible sur le site de Spliceator ainsi que sur le git (git.lbgi.fr/scalzitti/Spliceator). Le programme prend en entrée un fichier fasta ou multi-fasta et renvoie les prédictions des SE dans le terminal ou dans un fichier de sortie. Différentes options sont disponibles, comme le choix du modèle avec les options ***-md*** (donneur) et ***-ma*** (accepteur), ou le seuil de fiabilité avec ***-ta*** ou ***-td***. L'option ***-o*** permet d'écrire

les résultats dans un fichier de sortie et ***-a*** de calculer le score de tous les nucléotides de la séquence génomique d'entrée.

7.2.4.3 Prédiction des SE non-canoniques

En dehors de la prédiction des SE canoniques, Spliceator peut prédire des SE non-canoniques, car il a été entraîné avec ce type de données. Environ 2% des séquences incluses dans le jeu d'entraînement sont divergentes du dinucléotide GT ou AG. Ce rapport est celui que l'on observe chez les organismes eucaryotes (Frey and Pucker, 2020). Des tests ont été réalisés sur des séquences intégrant un SE non-canonique de plusieurs organismes et les résultats sont présentés dans le tableau 5.

Organisme	Séquence	SE	Ensembl_ID	Fiabilité (%)	Position
Homo sapiens	TGTTTACTGTTACAGGTTTCAAATTCTCTCCTGTTTGTTGTGACATGGAGGCTCAAGCTGGGTCTAC TCAAG**GC**AAGTGTTTTCAAATATTTGATCACAAATTATTTTTACTTCACCCCATCCCTGGCACCCTA CCCCCACAAAATGGAA	GC	ENST00000371130.7	98,1	Intron 3
Ovis aries	AGAAGACATCAACTACGCCTCCCAGAAGCAGGTGTACCGCGACATCGGCGAGGAGATGCTACAGC ACGCCTTCGAAGGCTACAACGTCTGCATCTTCGCCTACGGACAGACTGGGGCTGGCAAGTCCTAC ACCATGATGGGCAAGCAGGAGAAGGACCAGCAGGGCATCATCCCGCAGGCACGCGGCGGCGGGG CGGGGCCTG	GC	ENSOART00020001179.1	93,2	Intron 4
Ovis aries	TGCCTCATCTCTGCTTCCTTCTAGAACCTTTTACCAGTTTGAGGCTGCGTGGGACAGCTCCATGCAC AACTCCCTCCTGCTGAACCGGGTCACCCCTTACAGAGAGAAGATTTACATGACCCTCTCTGCTTAT ATCGAGGCAAGAAACCGATGCCCAGGGCCTC	GC	ENSOART00020001179.1	96,9	Intron 8
Philomachus pugnax	AAAAGCTCGTTTCATTAAAATGGGACACCTTAACTCTATGAATATTTTTCTTCGGCAAGAAATTGA CCGAATGCAAAAGTTGATCACAATAGTCCGCAACAGTTTGAATGATCTTAAATTAGCCATAGAAG GAACTATTGTTATGAGTGAGGCAAGTAAAAGCATAATTACTGCAT	GC	XM_014950520	92	Intron 86
Capsaspora owczarzaki	GATCAAATGCTCAAGACGGCTGTCGAGTTGCTCACCTTTGCGCGCCCAATGCTGGAGCGCCTGTTG CCACCGCCCACCGCACTCACACCGGCGCAAACGGCACTCCCTGCCAATCAGAGGCCCGGTTCGCT GCATGGCACCTGGAAGCACATTACATGCCTGCCAGACATGTGGATG	GC	KJE95720	98,6	Intron 7
Capsaspora owczarzaki	GCCTCCTCTTGTTTTTGTTGCACAGATCGCCCACCGCAAACCCAGCGAAATGTGGCAACATCTCGA CGTGCAAGCGATACCCCGAACGCAGCAAATGCAAGTCGTTGGTTTCTTGTTGCGT	GC	KJE95741	92,1	Intron 2
Apis melifera	GCAATTTATTGTTACACACCAGATGGTAGAGGTCCTTGTTATGCAGTAGAAGGTCAAAAAATTATG TTAGAATGGTTCAGAAGTTATCTAGTTATTATAGCAAAAGAAGCTGCTAATGTTCCAAGAACAAC AACTACTATTTCTGCGAAACCAAGGCAAGAATATTTCATATGTTTATAA	GC	XM_026444963	95,9	Intron 6
Homo sapiens	GTTTGTCTTAATCCTTTCTTTAAAGTATTACTATTTCAGAATTCTTTGATATACCCAAATTCTCCTTA ACCTGTTGCGGACATTGACCAAATGGTGATTGATGGGTGTAAGAAATACATGCGAAAAACGTGTG GCGATGTCTTAGACAATCTTAAAGGAGACTGCTATCAGGT	AC	ENST00000379404.5	85,8	Intron 4

Tableau 5 - *Identification de SE non-canoniques sur différents organismes eucaryotes.*

7.2.4.4 Analyse de variants

Les variants sont des permutations nucléotidiques (ponctuelle si elle concerne un seul nucléotide ou structurale si elle concerne un fragment de séquence). Il existe trois principales catégories de variants ponctuels :

- faux-sens : permutation d'un nucléotide qui induit un changement d'AA dans la CDS,
- non-sens : permutation d'un nucléotide générant un codon stop,
- silencieuse : variation qui n'a pas d'impact au niveau de la protéine.

Chez l'humain, de nombreuses variations sont observées au niveau des SE (appelées "variants d'épissage"). Elles peuvent activer un SE cryptique ou réduire sa reconnaissance par le spliceosome. Dans le cadre de diagnostic génétique, il est important d'identifier ce type de variants lorsqu'ils sont impliqués dans des maladies mendéliennes.

Spliceator peut prédire l'impact d'un variant sur le SE, en comparant la séquence *wild type* (WT) avec celle qui inclut le variant (*disease causing* (DC)). La différence de score de prédiction, appelée delta (Δ) est calculé selon :

$$\Delta = \left|\frac{(WT - DC)}{Score\ max}\right|$$

inspiré de Riepe *et al.*, (Riepe *et al.*). Plus Δ est grand, plus l'impact de la variation est important (figure 60).

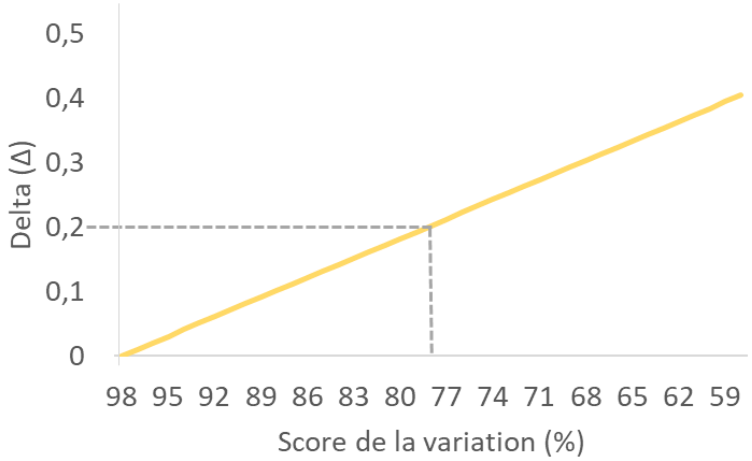


Figure 60 - *Variation du delta (Δ) en fonction du score de prédiction du variant. Si Spliceator prédit un SE WT avec une précision de 98% et s'il prédit le SE variant avec une précision de 78%, alors la valeur de Δ sera de 0,2. Plus cette valeur est élevée, plus l'impact de la variation est important.*

Dans certains cas, en raison de la variation, Spliceator peut ne plus reconnaître le SE, signifiant que la variation inhibe totalement le SE. Pour le moment, ces expériences sont réalisées manuellement, mais très prochainement, une mise à jour sera effectuée sur le site web (figure 61) afin d'ajouter un module pour la prédiction de variant d'épissage.

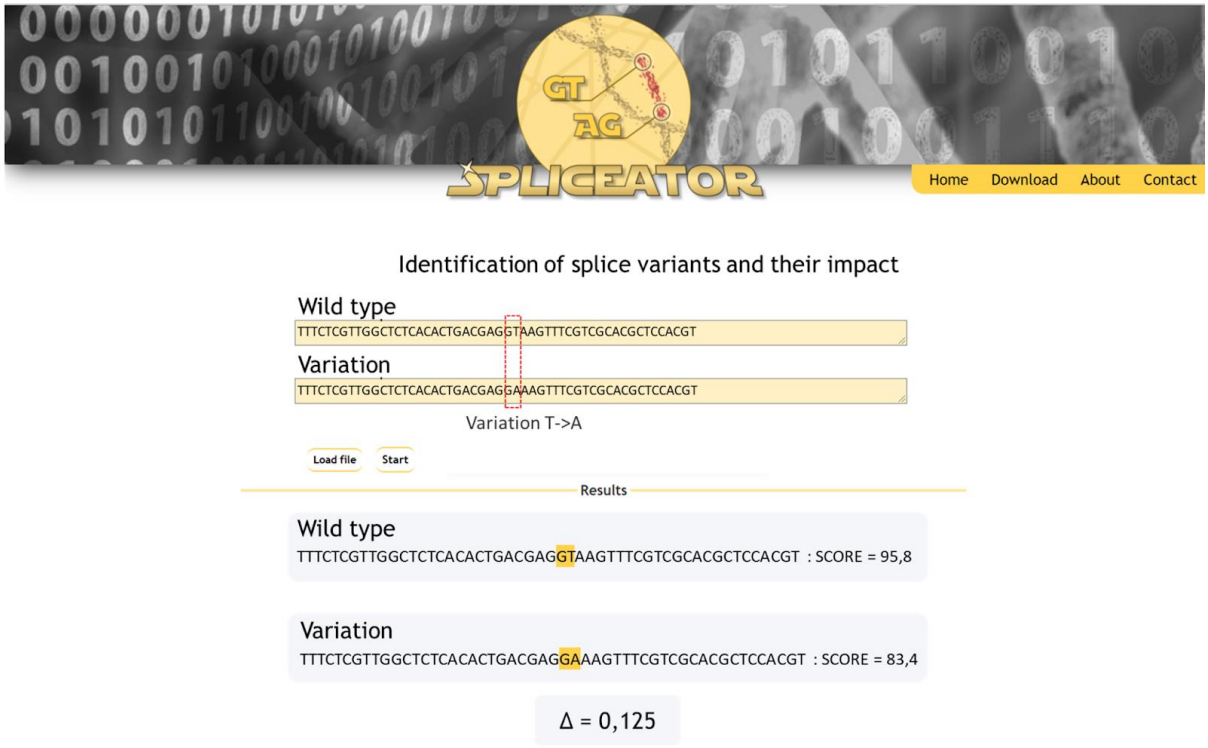


Figure 61 - *Maquette de la mise à jour du site web de Spliceator pour la prédiction de variants d'épissage. Deux séquences sont comparées (wild type vs disease causing) afin d'évaluer l'impact d'une variation.*

Des tests ont été effectués sur plusieurs variants d'épissage induisant une pathologie et les résultats sont répertoriés dans le tableau 6. Les programmes capables de prédire l'impact des variants de SE sont une bonne alternative, car pour le moment il est encore trop difficile de tester un grand nombre de variants en pratique pour un clinicien (Jian *et al.*, 2014).

ID	Nom	chromosome_ Position_brin	Type	Séquence	Score	Δ	Pathologie
1	NM_002074.5(GNB1):c.700-1G>T	1_1789270_-1	WT	CATGATGTCTTGATTTGCTTTCTCTTTCCAgTTCTTTCCAAATGGCAATGCATTTGCCACT	98,1	0,991	non repertorié
			DC	CATGATGTCTTGATTTGCTTTCTCTTTCCAtTTCTTTCCAAATGGCAATGCATTTGCCACT	0,00		
2	NM_014704.4(CEP104):c.2572-2A>G	1_3816372_1	WT	GATTAACACTCTGTGTGCCTCCTCTGCTGCaGGCATGGAAAGCTCACCTGATGGGCCCAGC	97,8	0,065	Joubert syndrome 25
			DC	GATTAACACTCTGTGTGCCTCCTCTGCTGCgGGCATGGAAAGCTCACCTGATGGGCCCAGC	91,4		
3	NM_000249.3(MLH1):c.1667+2T>C	3_37040296_-1	WT	TTATACCTTCTCAACACCACCAAGCTTAGGtAAATCAGCTGAGTGTGTGAACAAGCAGAGC	99	0,229	Hereditary nonpolyposis colorectal neoplasms|Hereditary cancer-predisposing syndrome|Lynch syndrome II|Lynch syndrome
			DC	TTATACCTTCTCAACACCACCAAGCTTAGGcAAATCAGCTGAGTGTGTGAACAAGCAGAGC	76,3		
4	NM_006118.3(HAX1):c.317-2A>G	1_154273772_1	WT	CATCCCAGCAAACACCTGCCACCTTTCTGCaGAACTTCCAGGTCCTGAGTCAGAGACACCT	92,7	0,184	Kostmann syndrome
			DC	CATCCCAGCAAACACCTGCCACCTTTCTGCgGAACTTCCAGGTCCTGAGTCAGAGACACCT	74,5		
5	NM_003921.5(BCL10):c.57+1G>A	1_85276295_-1	WT	GAGGAGGACCTCACTGAAGTGAAGAAGGACgTGAGTAACGCAGCTGTGCCCAGGGCGGGCG	98,8	0,115	Immunodeficiency 37
			DC	GAGGAGGACCTCACTGAAGTGAAGAAGGACaTGAGTAACGCAGCTGTGCCCAGGGCGGGCG	87,4		
6	NM_000277.3(PAH):c.441+1G>A	2_102877461_-	WT	TATGGAGCGGAACTGGATGCTGACCACCCTgTGAGTCCATGGCCCGTAGGATGAGATTTTT	98,4	0,179	not provided|Phenylketonuria
			DC	TATGGAGCGGAACTGGATGCTGACCACCCTaTGAGTCCATGGCCCGTAGGATGAGATTTTT	80,7		

Tableau 6 - *Résultats des performances de Spliceator sur l'analyse de l'impact des variants d'épissage. Dans le premier exemple, la variation inhibe complètement le SE, il n'est plus reconnu par Spliceator. Les autres variations sont moins impactées et le delta moyen est de 15.*

7.2.5 Publication

Spliceator: multi-species splice site prediction using convolutional neural networks

Nicolas Scalzitti[1], Arnaud Kress[1,2], Romain Orhand[1], Thomas Weber[1], Luc Moulinier[1,2], Anne Jeannin-Girardon[1], Pierre Collet[1], Olivier Poch[1] and Julie D. Thompson[1]*

[1] *Complex Systems and Translational Bioinformatics (CSTB), ICube laboratory, UMR7357, University of Strasbourg, 1 rue Eugène Boeckel, 67000, Strasbourg, France*

[2] *BiGEst-ICube platform, ICube laboratory, UMR7357,1 rue Eugène Boeckel, 67000, Strasbourg, France*

nicolas.scalzitti2@etu.unistra.fr; akress@unistra.fr; romain.orhand@etu.unistra.fr; thomas.weber@etu.unistra.fr; luc.moulinier@unistra.fr; jeanningirardon@unistra.fr; collet@unistra.fr; olivier.poch@unistra.fr; thompson@unistra.fr

*To whom correspondence should be addressed.

Abstract

Ab initio prediction of splice sites is an essential step in eukaryotic genome annotation. Recent predictors have exploited Deep Learning algorithms and reliable gene structures from model organisms. However, Deep Learning methods for non-model organisms are lacking. We developed Spliceator to predict splice sites in a wide range of species, including model and non-model organisms. Spliceator uses a convolutional neural network and is trained on carefully validated data from over 100 organisms. We show that Spliceator achieves consistently high accuracy (89-92%) compared to existing methods on independent benchmarks from human, fish, fly, worm, plant and protist organisms.

Keywords: splice site prediction, genome annotation, data quality, deep learning, convolutional neural network

Background

The raw genomic sequences generated by next generation sequencing (NGS) are an important source of data for studying and understanding organisms and biological mechanisms. However, without a crucial step of extracting biological knowledge from the raw data, a process called 'genome annotation', the sequences are difficult to exploit and may even be useless. A critical step in the annotation process involves the location of genes (*i.e.* 'structural annotation'), in particular the protein-coding genes and the characterization of their intron/exon structures. A large number of automatic annotation pipelines have been developed to identify protein-coding genes, such as Braker2 [1], Maker [2] or PASA [3], as well as dedicated resources, such as Ensembl [4] or NCBI [5]. Automatic annotation methods are generally based on a combination of empirical evidence, *e.g.* mRNA sequencing (RNA-seq) data or known gene structures from closely related organisms, and *ab initio* gene prediction programs, such as Augustus [6], Genscan [7], Snap [8] or GlimmerHMM [9]. Despite these developments, the annotation of

gene structure remains a major challenge, especially for eukaryotic organisms [10–13] due to their complex exon-intron mosaics [14] (Fig. 1).

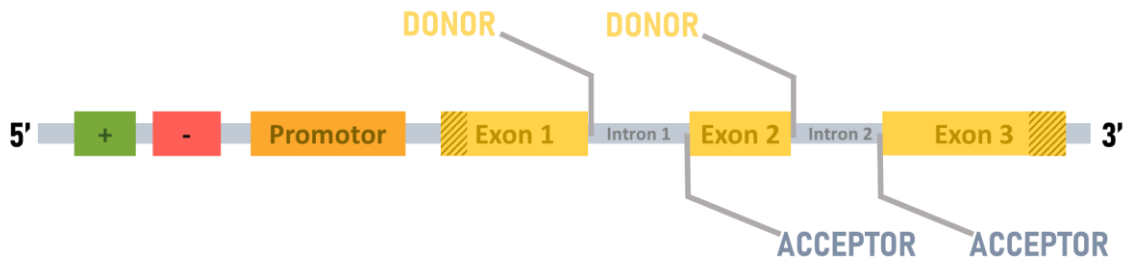


Fig. 1 – Typical architecture of a eukaryotic protein-coding gene. Green (enhancer) and red (silencer) boxes represent the regulatory elements. The mosaic of exons (labelled yellow boxes) and introns (labelled grey boxes) is usually preceded by a promotor (orange box). The brown diagonal stripes represent the untranslated regions (UTR). The boundaries between exons and introns are called donor splice sites and between introns and exons are acceptor splice sites.

Eukaryotic gene prediction involves determining the internal architecture of each gene, including the start and stop codons, and the boundaries between each exon and intron, called splice sites (SS). The SS are specifically recognized by the spliceosome, a ribonucleoprotein complex [15], and play an important role in the diversity of the proteome [16,17]. There are two types of SS, the 5' site (also called donor site) and the 3' site (or acceptor site), located respectively at the exon-intron and intron-exon junctions. SS are generally characterized by the presence of the dinucleotide GT at the 5' site and AG at the 3' site, called canonical sites [18]. The dinucleotides are embedded in longer, consensus motifs: aG**GT**AAGT (Donor) and (Y)6N(C/t)**AG**(g/a)t (Acceptor) [19]. Although the canonical SS are highly conserved [20] and represent more than 98.3% of SS in animals, 98.7% in fungi and 97.9% in plants [21], there are some exceptions, such as the presence of the dinucleotides AT-AC or GC-AG [22,23], described as non-canonical sites. Thus, the challenges in accurately predicting all SS in a genome are twofold. First, the huge number of GT and AG dinucleotides that are not located at

SS can generate a high rate of false positives. Second, the presence of non-canonical SS can lead to false negative predictions if they are not taken into account [24].

A number of methods have been developed to identify SS by exploiting recent high-throughput RNA-seq data, for instance MapSplice [25], TopHat [26] or SplitSeek [27]. However, this approach depends on the availability of high quality data and a minimal depth of sequencing to be able to detect all SS, in particular those in low-expressed isoforms [28]. As a consequence, alternative approaches are needed to identify SS based solely on the genome sequence. Most of them exploit machine learning (ML) algorithms and use several features to describe SS, covering the consensus motifs or other nucleotides in proximity to the SS [29]. The most widely used ML algorithms include Support Vector Machines [30–32], Markov models [33,34], Random Forest [35,36] and Bayesian networks [37]. However, these methods are limited by the lack of knowledge about the input sequence (patterns, secondary structures, etc.), complex biological processes [38], a weak genomic context (the region around the SS) and the construction and selection of pertinent feature sets [29], which is often time-consuming. More recently, programs using deep learning (DL) algorithms have been introduced, such as DSSP [39], SpliceRover [40] or SpliceFinder [41]. The DL approaches are based on convolutional neural networks (CNN) and do not require the manual definition of a feature set, because they automatically extract the most pertinent characteristics to classify elements (here, splice sites) in different classes determined by the initial problem [42]. Another advantage of these algorithms is that they are able to find correlations between features in a larger region (*i.e.* in the genomic sequence). In the context of SS detection, this characteristic is important, as several elements are involved, such as the branchpoint site (BPS), intronic splicing enhancers (ISE), intronic splicing silencers (ISS), exonic splicing enhancers (ESE) and exonic splicing silencers (ESS). Moreover, CNN use fewer parameters than classical multi-layer perceptrons, reducing the risk of overfitting [43], and they also share these parameters to extract local features.

DL methods rely on the availability of high quality data that is pertinent to the problem being solved, in order to train accurate models. For this reason, most of the current SS predictors have been trained on data restricted to humans or other model organisms. To our knowledge, there are no SS prediction tools trained on data from a large range of less well studied organisms, such as insects (except fruit fly), fungi or protists.

In this context, we have developed Spliceator, a new tool for *ab initio* prediction of eukaryotic multi-species splice sites. Spliceator is based on the CNN technology and more importantly, is trained on an original high quality dataset [44] containing genomic sequences from organisms ranging from human to protists. The training dataset has been rigorously established and validated to reduce the number of errors in the input data and avoid introducing bias in the learning process. This dataset allows us to limit the 'garbage-in, garbage-out' effect [45], meaning that poor quality data lead to less reliable results. Based on several benchmark experiments, we show that Spliceator achieves overall high accuracy compared to other state-of-the-art programs, including the neural network-based NNSplice [46], MaxEntScan [47] that models SS using the maximum entropy distribution, and two CNN-based methods: DSSP [39] and SpliceFinder [41]. More importantly, Spliceator performance is robust and remains consistently high for sequences from diverse organisms ranging from human to protists.

Results

Design of training and test datasets for multi-species SS prediction

Since we employ a supervised learning approach, the careful construction of the positive and negative datasets used for training the CNN models is essential. We designed eight strategies to build different datasets, where each strategy highlights a parameter that can influence the model performance, such as the input sequence length, the data quality, the type of negative sequences (only false positives (FP) or exon, intron and FP sequences) and the dataset

composition, *i.e.* the effect of balanced or unbalanced datasets with different ratios between the number of positive and negative sequences. Each dataset was then split into separate training and test sets in order to build prediction models for donor and acceptor SS using CNN.

The first dataset, called All Sequences (AS), includes sequences from the 1361 'Confirmed' (error-free) gene sequences available in the G3PO+ dataset (see "Methods"), as well as the 1380 'Unconfirmed' sequences that contain potential gene prediction errors. The AS dataset is designed to represent real-world problems, in the sense that the data is extracted directly from public databases. For the second dataset, called Gold Standard (GS), we exploited only the 'Confirmed' sequences implying that this set is error-free. The resulting AS and GS positive subsets include donor and acceptor SS from human, as well as from a diverse set of 147 eukaryotic organisms, ranging from primates to protists. In order to construct a robust negative subset of non-SS sequences, we again exploited both 'Confirmed' and 'Unconfirmed' sequences for the AS dataset and only 'Confirmed' sequences for the GS dataset. To do this, we randomly selected sequences from the exon/intron regions of the G3PO+ genomic sequences, as well as sequences containing GT/AG dinucleotides that are not SS.

In order to verify that there is no over-representation of sequences from specific organisms or clades, we calculated the mean pairwise percent identity for the input sequences with a length of 600 nucleotides (nt) (Table 1) and showed that the majority (>90%) of the sequences in each positive subset (AS and GS) for both donor and acceptor SS share between 20% and 30% identity. We also calculated the mean pairwise percent identity for sequences with a length of 20 nt, *i.e.* specifically the short region around SS. Again, the pairwise percent identity is similar for AS and GS positive datasets, however the majority of these sequences share between 20% and 60%, showing that the context close to the SS is more conserved. Interestingly, the donor sequences of length 20 nt are more conserved than acceptor sequences, *e.g.* in the GS dataset,

37.17% of donor sequences share 40-50% mean identity compared to 34.09% of acceptor sequences.

Table 1 Distribution of sequences according to the mean percent identity

Pairwise sequence identity	Sequence length: 600 nt				Sequence length: 20 nt			
	Donor		Acceptor		Donor		Acceptor	
	AS	GS	AS	GS	AS	GS	AS	GS
0-10%	0.0%	0.0%	0.0%	0.0%	0.04%	0.03%	0.04%	0.03%
10-20%	0.35%	0.27%	0.33%	0.25%	1.0%	0.71%	1.65%	1.32%
20-30%	**92.77%**	**94.88%**	**92.45%**	**94.7%**	10.27%	8.93%	13.92%	12.44%
30-40%	6.83%	4.78%	7.18%	4.99%	32.62%	31.57%	**34.05%**	33.20%
40-50%	0.03%	0.04%	0.03%	0.04%	**36.24%**	**37.17%**	32.89%	**34.09%**
50-60%	0.01%	0.01%	0.01%	0.01%	16.28%	17.47%	14.22%	15.34%
60-70%	0.0%	0.0%	0.0%	0.0%	3.2%	3.65%	2.9%	3.2%
70-80%	0.0%	0.0%	0.0%	0.0%	0.3%	0.39%	0.29%	0.33%
80-90%	0.0%	0.0%	0.0%	0.0%	0.03%	0.04%	0.03%	0.03%
90-100%	0.0%	0.0%	0.0%	0.0%	0.02%	0.03%	0.02%	0.02%

Pairwise sequence percent identity of positive subsets (AS: All Sequences and GS: Gold Standard) for sequences with a length of 600 nt and 20 nt for donor and acceptor SS.

Impact of genomic context

To evaluate the impact of the genomic context around the SS on the prediction performance of our CNN method, we constructed subsets of sequences for the AS and GS datasets having different lengths, ranging from 20-600 nt. The sequence segments upstream and downstream of the SS dinucleotide contain information allowing the discrimination of SS and non-SS, such as the BPS, polypyrimidine tract (PPT) or regulatory *cis*-elements including exon/intron splicing enhancers or silencers (ESE/ISE or ESS/ISS) [48]. Determining a pertinent sequence length is important because too short genomic regions would prevent the model from using important discriminatory sites, while too large genomic regions may introduce noise-inducing features and loss of accuracy [49]. We then built CNN prediction models for donor and acceptor SS, using these different sequence lengths. Fig. 2 (and Additional file 2: Table S1 and S2) summarizes the prediction accuracies obtained by the models for each SS on the test sets.

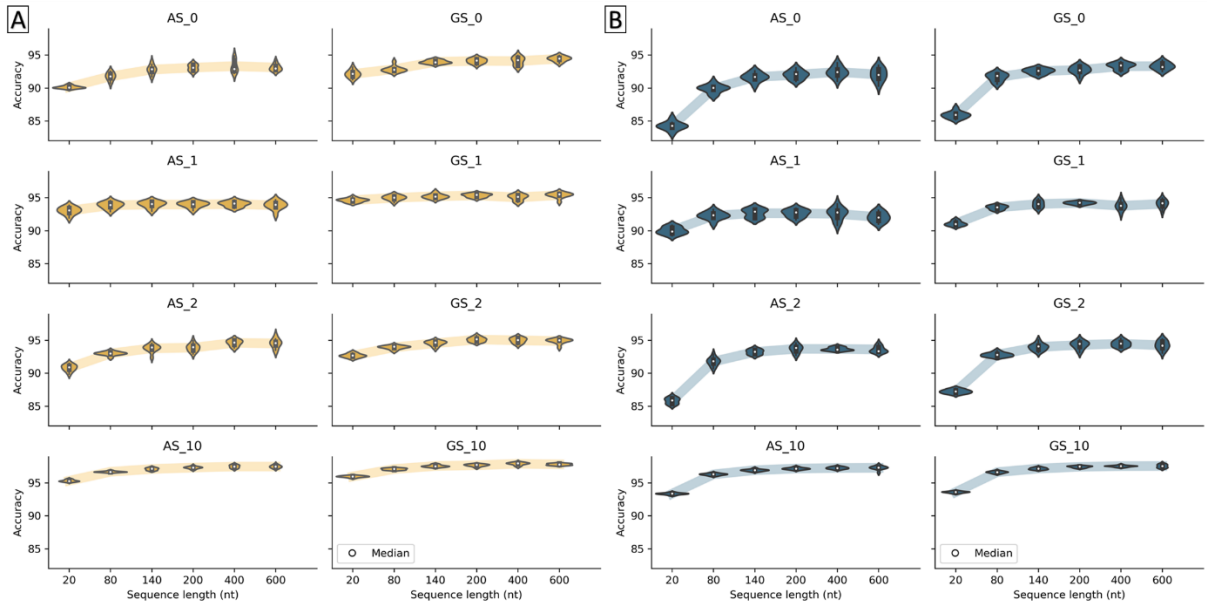


Fig. 2 – Prediction accuracy according to input sequence length for each dataset (AS: All Sequences and GS: Gold Standard) for A) donor and B) acceptor SS.

We observe similar trends for the prediction of donor and acceptor SS with the AS and GS datasets. The average prediction accuracy increases for sequence lengths ranging from 20 nt to 200 nt and then generally levels off, indicating that the model cannot find relevant genomic context features beyond this length. However, there are some differences between the datasets with different compositions (described in detail below). For example, for the AS_10 and GS_10 datasets (ratio 1:10 of positive to negative examples), the prediction accuracies are more homogeneous and higher than the other datasets. Interestingly, the sequence length has less effect for the AS_1 and GS_1 datasets, compared to AS_0 and GS_0 respectively. AS_1 (respectively GS_1) has the same balanced ratio of positive to negative examples as AS_0 (respectively GS_0), but the negative examples are more heterogeneous, consisting of exon, intron and FP sequences.

Based on this initial analysis, in the following experiments, we used a sequence length of 200 nt for the prediction of donor and acceptor sites with the AS and GS dataset, to consider a genomic context that is neither too small nor too large.

Impact of data quality

As described above, the GS dataset contains only true SS from the 'Confirmed' gene sequences, while the AS dataset includes some noise (*i.e.* false SS) from 'Unconfirmed' sequences. To estimate the impact of this noise on model prediction, we compared the average accuracy of the AS models with the corresponding GS models for each SS (donor and acceptor) and for different dataset compositions, as shown in Fig. 3 (Additional file S1: Table S1).

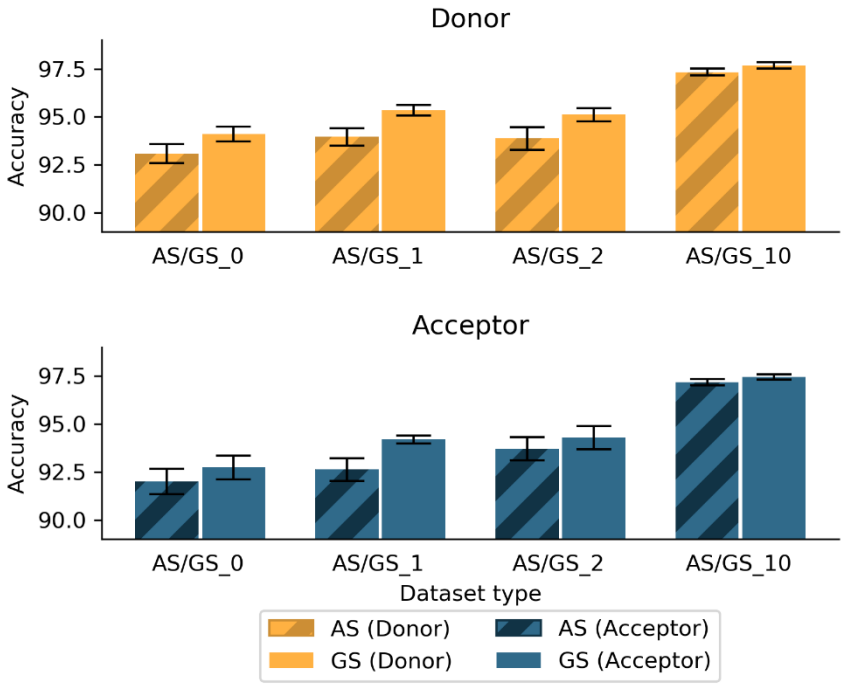


Fig. 3 – Average prediction accuracy for donor and acceptor SS, using the AS and GS datasets (AS/GS_0 = positive/negative ratio of 1:1 with only FP sequences in negative subset; AS/GS_1 = positive/negative ratio of 1:1 with exon, intron and FP sequences; AS/GS_2 = positive/negative ratio of 1:2 with only FP sequences in negative subset; AS/GS_10 =

positive/negative ratio of 1:10 with only FP sequences in negative subset). Standard deviations are indicated by black bars.

As expected, the GS models achieved the best accuracy with an average of 94.11%, 95.34%, 95.11% and 97.68% respectively for the GS_0, GS_1, GS_2 and GS_10 donor datasets and 92.73%, 94.19%, 94.59%, 97.45% respectively for the acceptor datasets. In comparison, the AS models obtained an average accuracy of 93.09% (-1.02%), 93.95% (-1.39%), 93.88% (-1.23 %) and 97.33% (-0.35%) respectively for the AS_0, AS_1, AS_2 and AS_10 donor datasets, and 91.99% (-0.74%), 92.62% (-1.57%), 93.70% (-0.49%) and 97.17% (-0.28%) for the acceptor datasets. The results of unpaired t-tests (Additional file S1: Table S2) show that all differences between AS and GS datasets are statistically significant. Interestingly, the difference between AS_1 and GS_1 is the largest for both donor and acceptor models. Based on these results, we selected only the GS models for the following experiments.

Impact of negative dataset composition

While the definition of reliable positive examples is clearly essential, the construction of the negative dataset will also have an impact on the ability of CNN methods to distinguish between positive and negative examples. The prediction of SS is an intrinsically unbalanced problem, since in a protein coding gene the SS represent only a small proportion of the total nucleotide length. Therefore, to investigate the impact of the negative subset on prediction performance, we constructed a number of datasets with different types of negative sequences and different ratios of positive and negative examples. We designed two balanced datasets, both with a ratio 1:1 of positive to negative examples, but with either homogeneous (GS_0) or heterogeneous (GS_1) negative examples, as well as two unbalanced datasets with ratios of 1:2 (GS_2) and 1:10 (GS_10) of positive to negative examples. The unbalanced datasets both have

heterogeneous negative examples. We then computed different metrics to evaluate the performance of each model on the test set, as shown in Fig. 4.

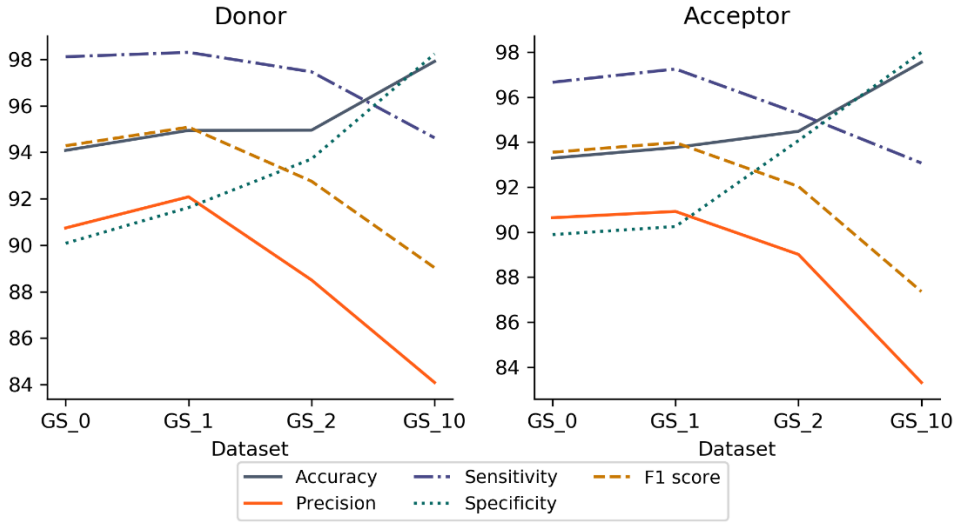


Fig. 4 - Average values of the 5 performance metrics (accuracy, precision, sensitivity, specificity and F1 score) for each dataset composition and for each type of SS (donor or acceptor). GS_0 = positive/negative ratio of 1:1 with only FP sequences in negative subset, GS_1 = positive/negative ratio of 1:1 with exon, intron and FP sequences in negative subset, GS_2 = positive/negative ratio of 1:2 and GS_10 = positive/negative ratio of 1:10.

The overall best performance for the test set is obtained using the GS_1 dataset, with a balanced number of positive and negative sequences and heterogeneous negative examples (exon, intron and FP sequences). Although the average accuracy is lower than for the unbalanced GS_10 dataset (positive/negative ratio of 1:10), the other metrics including the average F1 score and precision are higher for the GS_1 balanced dataset.

To confirm these results, the GS models were also evaluated on a set of 5 independent benchmarks (human, fish, fly, worm, plant) and using different metrics. The results are shown in Additional file S1: Figure S1A and B. The GS_1 dataset again obtains better overall

performance metrics for both donor and acceptor SS, notably for fly, worm and plant species. Based on these results, we chose to consider only the GS_1 models in the following experiments.

Performance of optimized CNN model

Based on our initial analyses, we determined the optimal training set for the CNN models to predict donor and acceptor SS, namely the GS_1 dataset: a high quality balanced dataset with an equal number of positive and negative sequences, heterogeneous negative examples containing exon, intron and FP sequences and an input sequence length of 200 nt. For this optimized model, we further characterized the prediction performance of Spliceator averaged over a total of 10 experiments due to the random selection of negative sequences. The results are shown in Fig. 5 (additional file S1: Table S3).

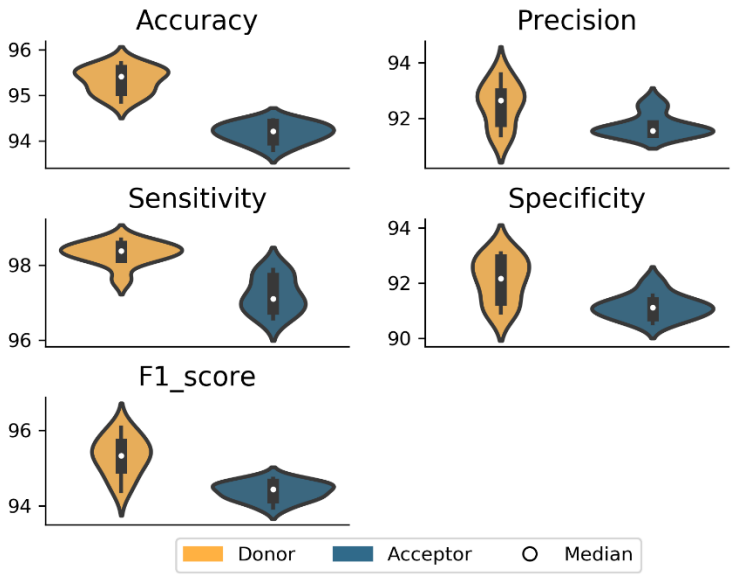


Fig. 5 – Performance of optimized model (GS_1 dataset, positive/negative ratio of 1:1 with heterogeneous negative examples and input sequence length = 200 nt) averaged over 10 experiments.

The average accuracy is 95.34% for the optimized donor model and 94.19% for the acceptor model. The precision of the models is similar, ranging from 92.50% (donor) to 91.73% (acceptor). We observed high sensitivity for both models, with 98.31% (donor) and 97.20% (acceptor), although the specificity is slightly lower, with 92.11% (donor) and 91.14% (acceptor). Finally, the F1 scores are similar for both donor and acceptor with 95.32% for donor and 94.39% for acceptor SS. Thus, the average performance for the donor model is slightly higher than for the acceptor model, which might be explained by the fact that the donor SS consensus motif is more conserved than the acceptor SS motif (Table 1). The acceptor SS also contains a low complexity PPT sequence, which may complicate predictions.

Model interpretability

In this section, we focus on the nucleotide regions that influenced the models during the learning step. Using the Grad-CAM method (see "Methods"), we measured the impact of each nucleotide position in the input sequences, thus allowing us to highlight the most important regions of these sequences that are determining factors in the learning step of the model. We calculated 10000 heatmaps per class and the average heatmap for each SS model (donor and acceptor) and each class (non-SS, SS) is shown in Fig. 6. Therefore, this representation shows only the most important features that the models use.

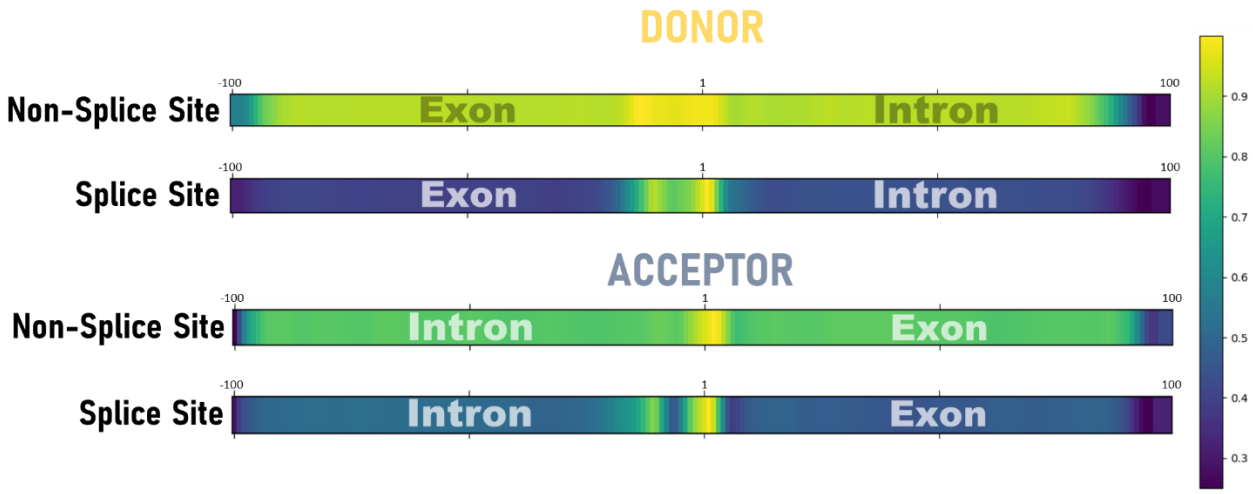


Fig. 6 – Average heatmap of the two classes, non-Splice Site and Splice Site, for donor and acceptor SS, with colors ranging from yellow (very important nucleotide position) to dark blue (not important position). The dinucleotide characterizing the SS is located at positions 101-102 for the donor and acceptor SS.

The heatmaps show that in order to classify sequences as non-SS, both models are based on elements of the whole sequence (score >0.8) with the exception of the 5' and 3' extremities (score <0.4, probably due to the CNN processing), although we observe a higher score (>0.9) close to the positions 1-2. The heatmaps for sequences containing donor or acceptor SS are more specific than for the non-SS sequences. For the donor sequences, the region around the SS (~10 nt upstream and downstream) seems to be more influential, with a predominance for the upstream exonic side. The positions 1 and 2, representing the GT dinucleotide, have the highest scores as expected. For the acceptor sequences, the central region around the AG dinucleotide is also the most important, although it is less well delineated than for donor SS. We observe an upstream intronic region of about 10 nt that seems to slightly impact learning (score >0.6), which probably corresponds to the PPT. A second upstream region from position -50 is also influential, possibly covering the BPS known to be generally located around 40 nt upstream of the acceptor site [50], although some BPS may be more distant up to a distance of 400 nt [51,52]. Interestingly, the downstream exonic region also seems to play a role in the training process for the acceptor SS (score >0.5).

Comparison with existing SS prediction methods

In order to compare the performance of the Spliceator models with other state-of-the-art methods, namely NNSplice, MaxEntScan, DSSP and SpliceFinder, we used six independent

292 benchmarks from a wide range of organisms (see “Methods”). The performance metrics are

293 shown in Table 2.

294

295 **Table 2** Performance of Spliceator and state-of-the-art programs on six independent
296 benchmarks.

	Accuracy		Precision		Sensitivity		Specificity		F1 Score	
	Donor	Acceptor	Donor	Acceptor	Donor	Acceptor	Donor	Acceptor	Donor	Acceptor
Human										
Spliceator	92.7	89.5	90.0	86.1	96.1	94.4	89.4	84.7	**93.0**	90.0
SpliceFinder	91.5	**91.1**	76.9	73.3	**97.0**	**99.8**	89.6	88.2	85.8	84.5
DSSP	**93.2**	91.0	**96.9**	**94.4**	89.2	87.1	**97.2**	**94.8**	92.9	**90.6**
MaxEntScan	91.8	83.7	88.8	77.3	95.8	95.6	87.9	71.9	92.1	85.5
NNSplice	68.4	65.9	62.3	61.5	89.2	86.7	48.5	44.8	73.4	72.0
Fish										
Spliceator	**95.0**	91.3	91.6	86.0	**99.0**	98.7	90.9	84.0	**95.1**	91.9
SpliceFinder	92.8	**94.9**	81.5	85.0	96.4	**99.8**	91.4	93.0	88.3	91.8
DSSP	94.6	93.6	**97.5**	**94.2**	91.5	92.9	**97.6**	**94.3**	94.4	**93.6**
MaxEntScan	93.9	83.3	90.6	75.4	98.1	99.0	89.8	67.7	94.2	85.6
NNSplice	71.0	68.9	64.3	63.9	90.8	85.6	52.2	52.5	75.3	73.2
Fly										
Spliceator	**94.6**	90.4	91.6	86.2	**98.3**	96.4	91.0	84.5	**94.8**	91.0
SpliceFinder	92.1	90.9	79.8	72.8	95.8	**99.8**	90.7	88.0	87.1	84.2
DSSP	94.0	**91.6**	**96.6**	**93.1**	91.3	89.8	**96.7**	**93.4**	93.9	**91.4**
MaxEntScan	94.5	86.0	91.8	79.5	97.6	97.0	91.3	75.0	94.6	87.4
NNSplice	71.7	71.1	65.3	64.8	91.4	92.1	52.3	50.3	76.2	76.1
Worm										
Spliceator	**93.9**	**88.8**	91.4	86.9	**96.9**	91.4	90.9	86.2	**94.1**	**89.1**
SpliceFinder	88.7	87.9	71.2	64.1	93.3	**99.3**	87.1	84.8	80.8	77.9
DSSP	90.4	84.1	**97.0**	**93.1**	83.4	73.6	**97.4**	**94.5**	89.7	82.2
MaxEntScan	92.9	82.0	91.6	76.8	94.5	92.0	91.3	72.1	93.0	83.7
NNSplice	65.8	57.8	61.9	58.0	80.3	56.4	51.7	59.2	69.9	57.1
Plant										
Spliceator	**94.7**	**90.6**	91.8	89.1	**98.1**	92.7	91.3	88.6	**94.9**	**90.8**
SpliceFinder	88.9	87.6	72.2	63.1	92.9	**99.3**	87.5	84.4	81.3	77.2
DSSP	88.2	85.1	**96.5**	**94.2**	79.3	74.8	**97.1**	**95.4**	87.1	83.3
MaxEntScan	92.4	85.3	90.5	79.2	94.6	95.9	90.1	74.8	92.5	86.7
NNSplice	59.8	59.2	57.2	57.1	78.4	71.3	41.0	47.3	66.2	63.4
PVP										
Spliceator	86.0	83.5	82.9	83.1	**90.8**	84.0	81.2	82.9	**86.6**	**83.6**
SpliceFinder	**87.3**	**89.3**	67.9	58.8	77.6	**98.6**	89.9	87.7	72.4	73.8
DSSP	84.3	76.6	**92.5**	**84.7**	74.6	65.0	**93.9**	**88.2**	82.6	73.5
MaxEntScan	84.5	80.0	81.0	75.4	90.1	89.4	79.1	70.4	85.2	81.8
NNSplice	64.8	63.7	60.4	61.4	85.8	73.9	43.6	53.5	70.9	67.1
Average										
Spliceator	**92.82**	89.02	89.88	86.23	**96.53**	92.93	89.12	85.15	**93.08**	**89.40**
SpliceFinder	90.22	**90.28**	74.92	69.52	92.17	**99.43**	89.37	87.68	82.62	81.57

DSSP	90.78	87.00	**96.17**	**92.28**	84.88	80.51	**96.65**	**93.43**	90.10	85.77
MaxEntScan	91.67	83.38	89.05	77.27	95.12	94.82	88.25	71.98	91.93	85.12
NNSplice	66.92	64.43	61.90	61.12	85.98	77.67	48.22	51.27	71.98	68.15

Performance metrics for Spliceator, SpliceFinder, DSSP, MaxEntScan, and NNSplice, using six independent benchmarks from model and non-model organisms: human, fish, fly, worm, plant and PVP (Protists+Viridiplantae).

For the donor SS prediction, Spliceator obtains the best average accuracy of 92.82%, with an average increase of +25.9, +2.6, +2.04 and +1.15% compared to NNSplice, SpliceFinder, DSSP and MaxEntScan respectively. For the acceptor SS prediction, Spliceator obtains the second best average accuracy, with 89.02% (-1.26%) compared to SpliceFinder with 90.28%, although Spliceator is more accurate on the Worm and Plant benchmarks. To further investigate the reasons for the different performances, we considered four other metrics, including the precision, sensitivity, specificity and F1 score. A high precision indicates that the program predicts few FP. Spliceator obtains the second best average precision of 89.88% for the donor SS and 86.23% for the acceptor SS, behind DSSP (96.17% for donor and 92.28% for acceptor SS). SpliceFinder, which has generally good accuracy, obtains lower precision (74.92% for the SS donor and 69.52% for the SS acceptor). Sensitivity and specificity are two inseparable metrics. They describe the proportion of well predicted elements and their quality, *i.e.* if elements have been correctly predicted. A high sensitivity and low specificity indicates that many non-SS elements are predicted as SS. The F1 score combines these two metrics and provides a more global view of the number of correctly predicted elements. Spliceator obtains the best average F1 score for both donor and acceptor SS, with 93.08% (+1.15%) and 89.4% (+3.83%) compared to the second best program for donor SS, MaxEntScan with an F1 score of 91.93% or acceptor SS, DSSP with a F1 score of 85.57%. Fig. 7 shows the accuracy and F1 score for each program on the individual benchmarks containing SS from different organisms. While most of the programs tested achieve high scores on the vertebrate sequences (human and

fish), reflecting their training sets focused on human SS, Spliceator performance is generally better on the more distant organisms (worm, plant and PVP).

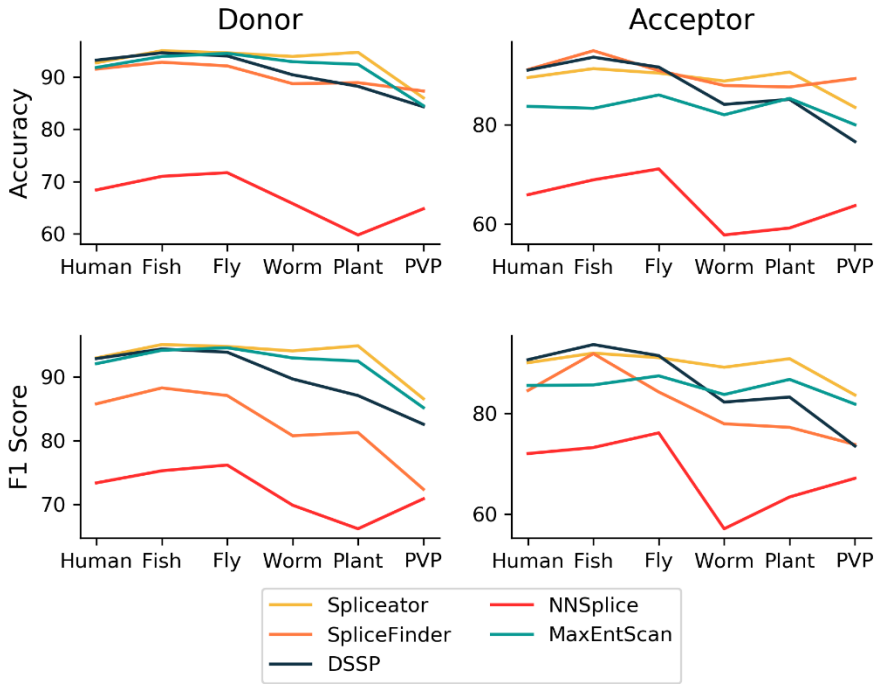


Fig. 7 - Accuracy and F1 score for each program and for each independent benchmark representing diverse organisms.

Finally, for each model, we observe that the predictions of the donor SS are slightly better than those of the acceptor SS. This difference is probably due to the divergence of the acceptor SS motif, but also to the genomic context around it which seems to be more complex.

Discussion

Thanks to high-throughput technologies, as well as the development of computing power, huge amounts of data can now be exploited by DL algorithms and produce remarkable results [53–56]. In particular, CNN are increasingly used in the field of bioinformatics [57–59], for example to detect specific patterns in a genomic sequence [60] where the reduced number of parameters allows for better generalization compared to other ML methods. Moreover, maxpooling techniques allows the algorithm to focus on the local features that it considers most important.

In this context, we have developed a SS prediction program called Spliceator, based on a three-layer convolutional CNN. Despite the recent use of RNA-seq to accurately identify SS, it is currently impossible to obtain experimental data for the full panel of transcripts for all tissues and all developmental stages. Thus, *ab initio* SS prediction programs that rely only on the genomic sequence remain essential. Clearly, it would be ideal to couple *ab initio* prediction programs such as Spliceator with RNA-seq based programs in genome annotation tools or workflows.

The accuracy of *ab initio* algorithms is dependent on the quality of the data used during the training step. Indeed, although neural networks are getting deeper and more complex [61,62], the fact remains that data is the cornerstone of artificial intelligence. Consequently, the majority of SS prediction programs use sequence data from humans (such as GRCh38 or HS3D [63]) or other model organisms [36,39,64], where high quality, expert-refined data is available. For non-model organisms, it is more difficult to find accurate training data and very few CNN methods have been designed specifically to predict SS in non-model organisms. Since SS and other regulatory motifs may be conserved across similar species [20], some work has been done to try to transfer models trained on model organisms to related organisms, for example between different vertebrate genomes [65]. Others have built cross-species models for specific clades, such as animals or plants using Helixer [66], but unfortunately the source code for this program is not yet stable (according to the authors). The aim of our work was to extend the idea of cross-species models to conceive universal SS prediction models (one for each SS), that are applicable to a wider range of organisms.

For the training of the Spliceator models, we focused on the construction of a multi-species dataset that is as representative as possible of the eukaryotic domain (from primates to protists). This dataset is based on an extension (G3PO+) of the gene prediction benchmark G3PO. Since high-quality, genome-wide annotations are not available for the 147 species in this dataset, we

developed a protocol based on expert-guided comparative sequence analysis in order to identify reliable SS in a subset of genes. Since the G3PO+ gene sequences are evolutionarily related, we eliminated redundant sequences, which could cause potential bias of sequence over-representation and thus a risk of overfitting. We also made an effort to respect the proportions of non-canonical SS (2.2% donor and 1.3% acceptor) found in real-world data [21].

To investigate the impact of the quality of the initial training data on the CNN models, we extracted data from public databases such as Ensembl [4] and UniProt [67], where it has been estimated that many proteins (with the exception of Swiss-Prot, which represents 0.3% of UniProt) have errors [68]. We then built a dataset called 'All Sequences' (AS), that includes some badly predicted gene sequences [44] and thus introduces noise in the form of wrong or missing SS. We compared the CNN model trained on the AS dataset with a second model trained on a 'Gold Standard' (GS) dataset, which was cleaned by removing all error-prone sequences. Since our results showed that the quality of the data had a significant impact on the accuracy of the models, we conclude that quality control and data cleaning steps are essential in order to obtain better results.

We also tested the impact of other parameters, such as the length of the input sequences. It is important to carefully select the size of the genomic sequence in order to take into account different important elements such as regulatory elements (ESE, ESS, ISE or ISS [69]), the BPS [70] and the PPT [22] that can be kept and help the algorithm to generalize. All these elements constitute intrinsic signals that are indispensable for the spliceosome to accurately recognize the SS. In order to include enough *cis* elements without introducing too much noise, we chose an input sequence length of 200 nt for both donor and acceptor models. Unfortunately, many other external signals impacting SS recognition by the spliceosome cannot be detected by current methods such as the secondary structure of RNA [71], or the transcription speed of polymerase II [72].

Finally, we investigated the impact of the negative examples and the use of balanced or unbalanced datasets, in terms of the ratio of positive to negative examples. SS prediction is an inherently unbalanced problem, because the number of nucleotides involved in a SS is much smaller than the number of non-SS nucleotides. The results confirmed the hypothesis that an unbalanced dataset was more prone to overfitting because one of the classes is overrepresented [65]. Moreover, the heterogeneous negative examples provided better performance. All these tests allowed us to optimize our method and improve prediction performance.

In order to estimate the performance of Spliceator on independent genome data, we used six different benchmarks from diverse organisms and compared Spliceator with a number of state-of-the art programs, including two other CNN-based methods. As expected, the more recent CNN-based methods generally achieved higher performance metrics than the older prediction methods that used either neural networks or maximum entropy distributions approaches. We calculated a number of different performance metrics, since the most suitable metric to measure 'good' performance will depend on the specific user application. For example, accuracy is useful when the true positives and true negatives are more important, while the F1 score is used when the false positives and false negatives are crucial. Spliceator achieved the highest accuracy (92.82%) for the donor SS, and the second best accuracy (89%) for the acceptor SS. In terms of F1 score, Spliceator outperformed the current state-of-the art programs for both donor (93.08%) and acceptor (89.40%) SS. Interestingly, Spliceator performed very well on the human benchmark even though it was trained with only 45 human genes. However, a major strength of Spliceator is that it maintains good performance over a wide range of organisms, from human to protists. Our results thus showed that a universal SS prediction program is feasible, and hopefully performance can be further increased in the future by including more divergent species data in the model.

Conclusions

Here, we present a new approach to train Spliceator, a universal splice site prediction program based on a high-quality dataset from diverse eukaryotic organisms (from primates to protists). We highlighted the inherent link between data quality and the performance of prediction programs based on machine learning algorithms. We also showed that including high quality multi-species data can result in accuracy equivalent to other state-of-the-art SS prediction programs. In the future, it would be interesting to include more data from other species, but also to test other types of network architecture in order to extract new high-level features. Moreover, as some of the extracted features are highly conserved, it would be interesting to use our model to perform transfer learning for gene annotation of other organisms.

Methods

Data collection

Initial datasets were constructed for each type of SS, in order to establish two separate models: one to predict donor SS and one to predict acceptor SS. The models developed in this study are based on supervised learning allowing the classification of entries in two classes (0: nucleotide not involved in SS, and 1: nucleotide involved in SS). Thus, each dataset is constructed from a positive subset containing SS sequences and a negative subset containing non-SS sequences.

To build the positive and negative subsets, gene sequences from the multi-species benchmark G3PO [44] were used. G3PO is based on 147 phylogenetically disperse organisms and contains 1793 sequences including 20 human Bardet-Biedl Syndrome (BBS) genes (Additional file 1: Table S4) and their orthologous sequences (ranging from primates to protists) extracted from the OrthoInspector database v3.0 [73]. Following the same methodology implemented in G3PO, we extended the original dataset by adding 948 sequences from 25 human genes responsible for myopathies and their orthologs from 47 metazoans (Additional file 1: Figure

S2). The 948 protein sequences in the extended dataset (called G3PO+) were analyzed according to the G3PO protocol in order to classify them into two categories: those without gene prediction errors (called 'Confirmed') and those that contain at least one error (called 'Unconfirmed'). Errors include insertions, deletions and mismatches in the N-terminal, C-terminal or internal regions. This protocol allows to verify the quality of the data and ensures that the SS present in the 'Confirmed' sequences are biologically true, *i.e.* they are recognized by the spliceosome. Table 3 summarizes the composition of the G3PO+ dataset.

Table 3 Composition of the original G3PO and extended G3PO+ datasets

	G3PO	**Extension**	**G3PO+**
Confirmed	889	472	1361
Unconfirmed	904	476	1380
Total	1793	948	2741

To build the G3PO+ dataset, we retrieved orthologous sequences for 45 human genes and performed multiple sequence alignments. Each sequence was then checked to identify those that contained no errors, called 'Confirmed', and those that contained at least one error, called 'Unconfirmed'.

Construction of a series of training and test sets

Based on the gene sequences in the G3PO+ benchmark, we constructed a series of datasets used to train the Spliceator models and estimate the effect of various parameters on their prediction performance. Fig. 8 shows an overview of the dataset construction process. The positive and negative subsets are described in the following sections.

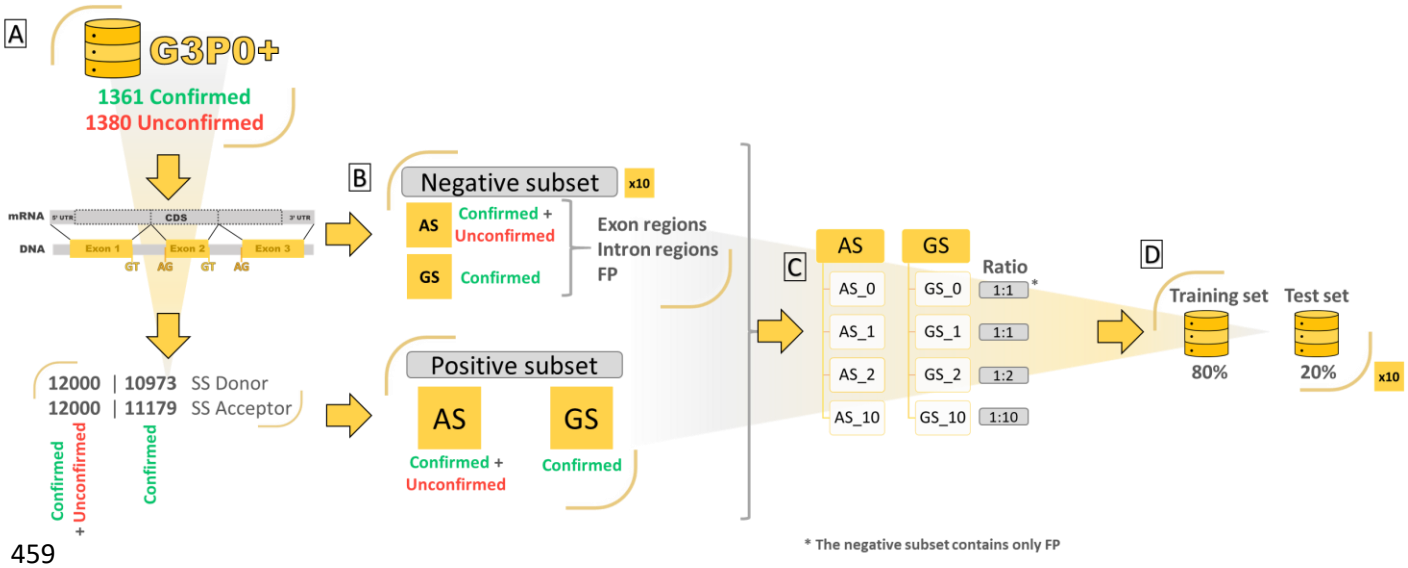


Fig. 8 – Overview of the construction of the training and test sets. A) DNA sequences and exon maps are recovered for each G3PO+ gene. B) The AS (All Sequences) positive subset includes the SS of all G3PO+ 'Confirmed' and 'Unconfirmed' sequences. The GS (Gold Standard) positive subset includes only the SS of the 'Confirmed' sequences. Ten negative AS subsets and ten negative GS subsets are then constructed by random sampling of the exon, intron and FP regions of the corresponding genomic sequences. C) Four AS and four GS datasets are then constructed with different ratios of positive and negative SS (described in table 4). D) Finally, the training and test sets are formed by shuffling the positive and negative sequences (10 times for each AS and GS dataset).

Positive subsets

The 2741 G3PO+ sequences are classified into 2 categories, either 'Confirmed' (1361 sequences) because they were annotated 'error-free', or 'Unconfirmed' (1380 sequences) because they contained at least one gene prediction error. For each G3PO+ sequence, genomic sequences and exon maps were retrieved from the Ensembl database [4] release 87, and the SS were extracted, flanked by a +/- 300 nt environment. A verification was made to ensure that no sequences containing undetermined nucleotides (noted 'N') were selected. The GS datasets

contain only SS from the 1361 'Confirmed' sequences. Thus, the same positive subset for each GS dataset (GS_0, GS_1, GS_2 and GS_10) contains 10973 donor and 11179 acceptor SS sequences, where each sequence is of length 600 nt with the GT (donor) or AG (acceptor) dinucleotide in the central position (301 and 302). In contrast, the AS datasets contain SS from all 2741 G3PO+ sequences, including both 'Confirmed' and 'Unconfirmed' sequences. Thus, the AS datasets are representative of the data present in the public databases, as no pre-processing has been performed on the data, and therefore they include a certain number of errors. To eliminate any bias from the size of the datasets, an equivalent number of SS sequences were used (12000 donor and 12000 acceptor) to form the same positive subset for each AS dataset (AS_0, AS_1, AS_2 and AS_10).

In order to test the impact of the genomic context, each dataset is provided in six different versions, according to the defined length of the input sequences. Sequence lengths selected for this study are 20, 80, 140, 200, 400 and 600 nt, where the SS is always in the central position. To reduce redundancy, for each sequence length, duplicate sequences are removed. As the length of the sequences decreases, the number of duplicates increases, reducing the size of the data sets (especially for 20 nt sequences). Fig. 9 (Additional file 1: Table S5) summarizes the composition of the AS and GS positive subsets according to sequence length. Each SS is described according to its type, either canonical (*i.e.* GT for donor site and AG for acceptor site) or non-canonical. The number of non-canonical donor and acceptor SS present in each AS and GS dataset for each sequence length is also shown in Gig. 9 (Additional file 1: Table S5). In addition, Fig. 10 shows the sequence logos of the canonical and non-canonical donor and acceptor SS motifs from the AS and GS dataset sequences. The sequence logos were made with the program WebLogo v3.7.4 [74].

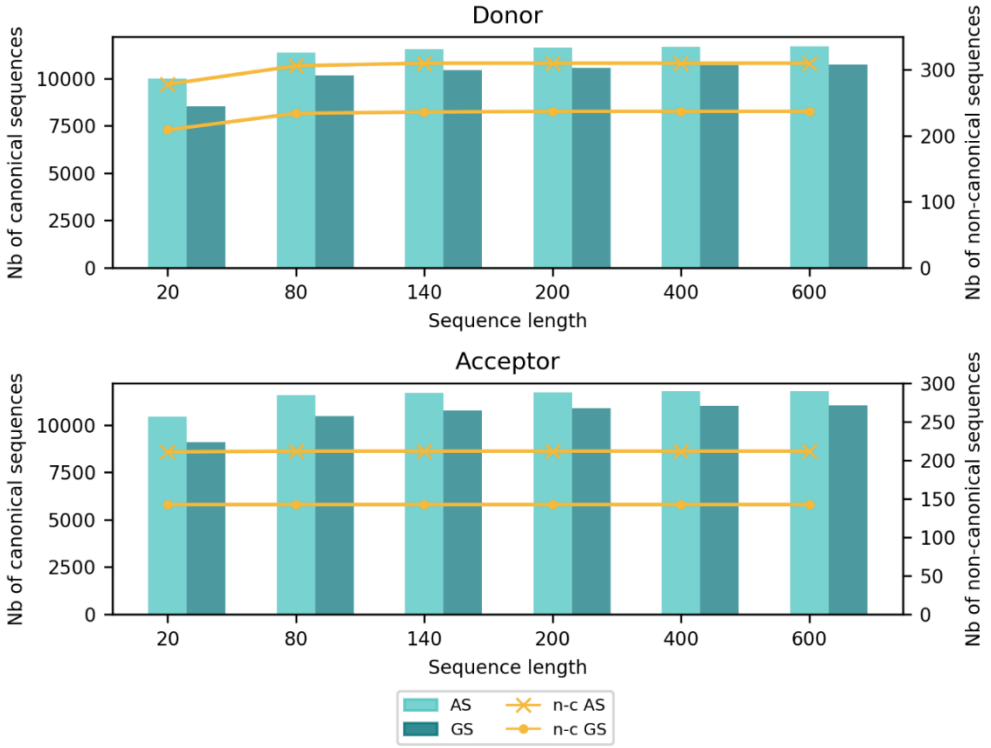


Fig. 9 – Number of canonical (bar) and non-canonical (n-c) (line) sequences for each positive subset (AS and GS) and for each sequence length.

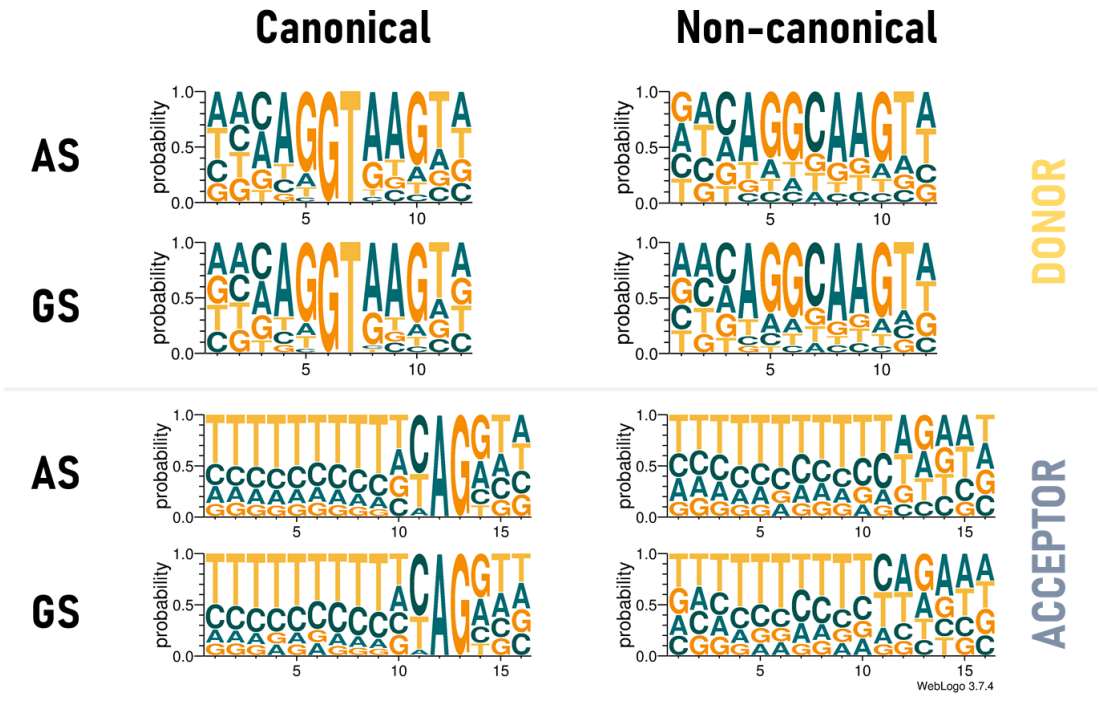


Fig. 10 - Sequence logos for canonical and non-canonical SS for each SS type (donor or acceptor) and each positive subset (AS and GS).

Negative subsets

Two negative subsets were first constructed. The first one is composed only of FP sequences, *i.e.* randomly selected regions in the G3PO+ sequences in both 'Confirmed' and 'Unconfirmed' sequences (for AS dataset) or only 'Confirmed' sequences (for GS dataset), with a GT or AG dinucleotide (depending on the type of SS), in the central position (*e.g.* 301–302 for length = 600 nt) that do not correspond to a SS identified in the positive subsets. The second type of negative subset is composed of 3 categories of sequences extracted from the G3PO+ dataset:

- Exon sequences: randomly selected exon regions,
- Intron sequences: randomly selected intron regions,
- False positive SS: randomly selected GT or AG dinucleotides.

Negative subsets were also constructed with different numbers of sequences depending on the size of the positive subset: (i) with a ratio 1:1 we have the same number of positive and negative sequences, (ii) with a ratio 1:2 we have twice as many negative sequences as positive sequences and (iii) with a ratio 1:10 we have ten times more negative sequences than positive sequences. As for the positive subsets, identical redundant sequences and sequences containing undetermined 'N' characters were removed. Finally, as the selection of the negative sequences is random, 10 random selections were made, in order to obtain 10 different negative subsets and to eliminate potential random bias due to a specific data sampling.

Data composition strategies

By combining the same positive subset with different negative subsets, a number of datasets were constructed in order to measure the impact of different parameters, including the type of negative examples used (heterogeneous = exons, introns and FP or homogeneous = only FP), the use of balanced or unbalanced datasets defined by the ratio of positive to negative examples, and data quality (AS vs. GS). In total, eight datasets were established, summarized in Table 4.

Table 4 Composition of the 8 datasets

Dataset	Quality of sequences	No. of positive sequences		No. of negative sequences	Type of negative sequences	Ratio
		Donor	Acceptor			
AS_0	Unconfirmed & Confirmed	12000	12000	12000	FP only	1:1
AS_1				12000	4000 exons, 4000 introns and 4000 FP	
AS_2				24000	FP only	1:2
AS_10				120000	FP only	1:10
GS_0	Confirmed	10973	11179	11000	FP only	1:1
GS_1				11000	3650 exons, 3650 introns and 3700 FP	
GS_2				22000	FP only	1:2
GS_10				110000	FP only	1:10

Composition of the 8 datasets used to study the impact of (i) the type of negative examples (only FP sequences vs. heterogeneous data with exons, introns and FP sequences), (ii) the ratio of positive to negative examples (1:1, 1:2 and 1:10), (iii) data quality ('Confirmed' and 'Unconfirmed' sequences in the AS datasets vs. only Confirmed sequences in the GS datasets. FP = False Positive, GS = Gold Standard, AS = All Sequences.

Sequence identity

A sequence similarity search was performed on the whole sequences with a length of 600 nt and 20 nt from the AS and GS positive subsets. Each sequence was compared to all the others and the pairwise percent identity was defined by:

$$\%\, Identity = \left(\frac{Number\ of\ identical\ nucleotide}{Length\ of\ sequence}\right) * 100$$

Preparation of datasets for CNN

Training and test sets

For each dataset described above, a random selection was performed to form a training set containing 80% of the sequences and a test set containing 20% of the sequences. Since there are 10 different negative datasets, there are 10 different training and test sets. Fig. 11

(Additional file 1: Table S6 A and B) summarizes the average number of negative and positive sequences in all training and test sets for each dataset (AS and GS) and for donor and acceptor SS.

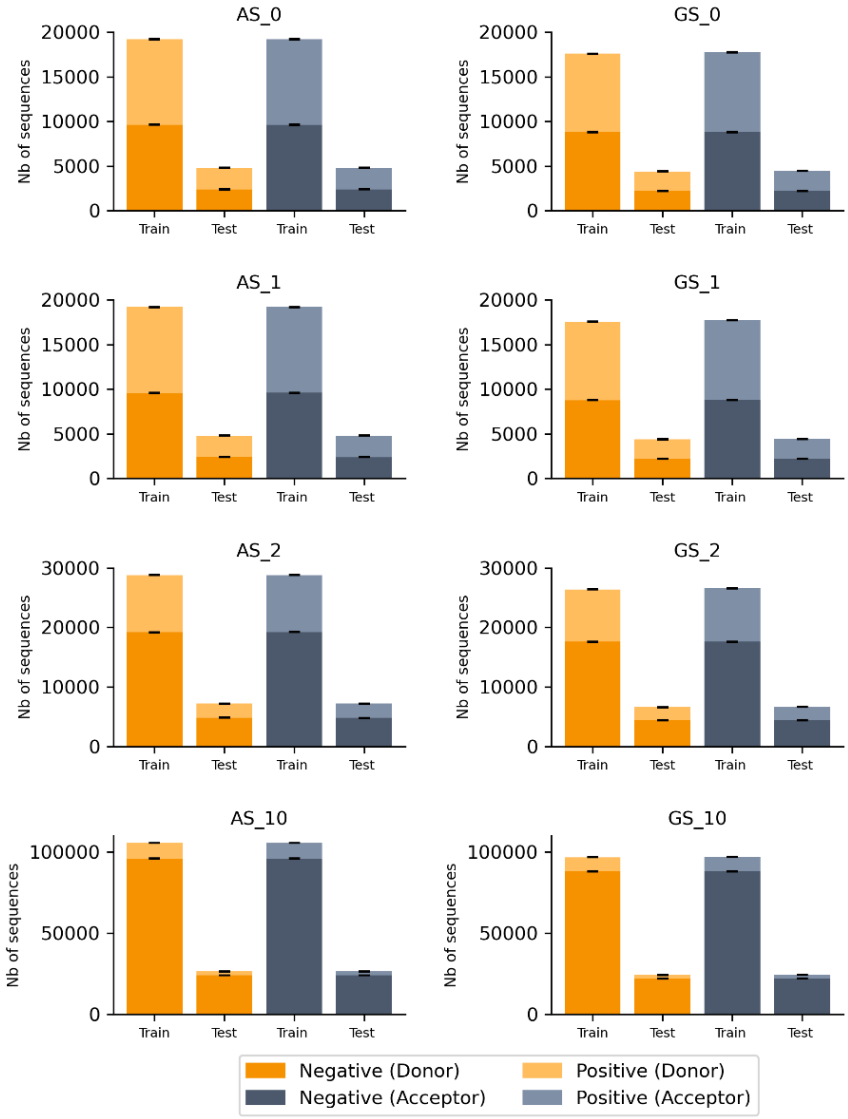


Fig 11 - Average number of positive and negative sequences in training and test sets, for all AS and GS datasets, according to SS type. Standard deviations are indicated by black bars.

Data encoding

For efficient exploitation of the genomic sequences in the training and test sets, a one-hot encoding step was performed. Each nucleotide of an input sequence was converted into a binary

vector of size 4. Adenine is encoded by the vector (1,0,0,0), Cytosine by (0,1,0,0), Guanine by (0,0,1,0) and finally Thymine is encoded by (0,0,0,1). In the case where an external sequence contains indeterminate 'N' nucleotides, *e.g.* when users test their own sequences, the vector (0,0,0,0) is used. Thus, each output sequence is a first order tensor (vector) of size W, where W is the length of the input sequence, with 4 channels representing the one-encoding. Finally, the shape of the input is: S x H x W x C, where S is the number of input sequences, H is the height of the 1D vector (so H = 1), W is the width of the vector corresponding to the length of the input sequences and C is the number of channels from one-hot encoding. Fig. 12 summarizes the data encoding.

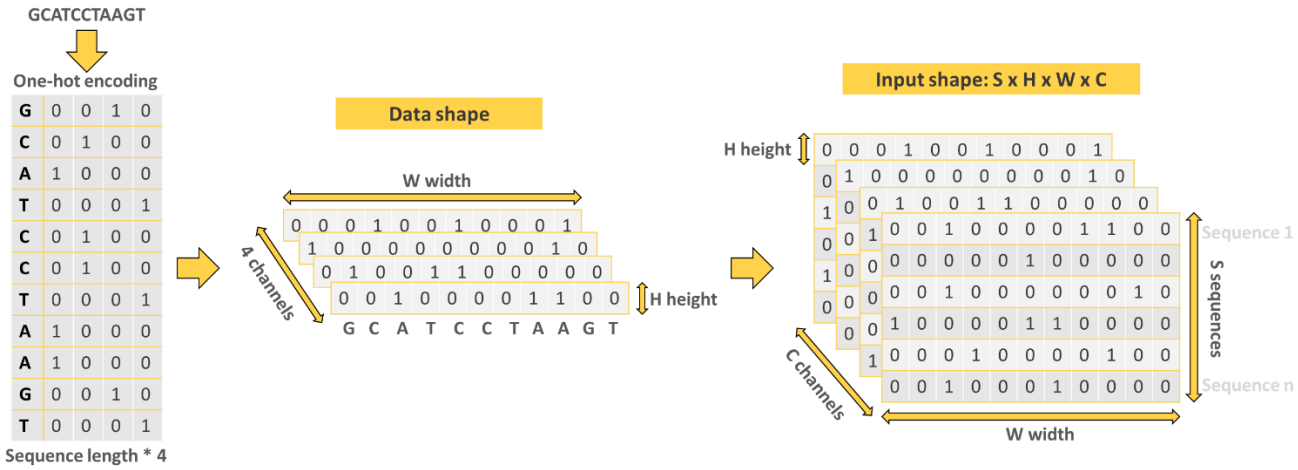


Fig. 12 – Data pre-processing. Input sequences are converted in one-hot encoding. The result is a 1D vector of size W, where W is the length of the input sequences, with 4 channels.

Finally, the shape of the input is S x H x W x C (S = number of input sequences, H= height of the vector (here equal to 1 because the vector is 1 dimensional), W is the length of each input sequence and C is the number of channels).

Convolutional Neural Network

Models were constructed and trained for each type of SS (donor or acceptor) independently. The models result from supervised learning, where genomic sequences (input) are coupled with

class labels 0: non-SS and 1: SS. The CNN then applies filters on each input sequence and tries to modify the weights of these filters to improve the predictions thanks to the back-propagation algorithm. The filters allow to extract pertinent features/patterns within the input data. The output of the CNN is a vector of size 2, corresponding to the non-SS (0)/SS classes (1). The implementation of the CNN, as well as the training of the models, was done in Python v3.7, with Tensorflow v2.4.1 [75], the API Keras v2.3.1 and the Scikit-learn library v0.23.2 [76].

Architecture

We constructed a CNN architecture for donor or acceptor prediction, composed of a series of three convolutional layers over a single spatial dimension. The layers are composed of 16, 32 and 64 filters of sizes 7, 6 and 6 respectively, with a stride of 1. Between each convolution layer, a maxpooling layer of size 2x1 with a stride of 2 is added. Maxpooling allows to under-sample the data by reducing their size, while preserving the features that seem important. A dropout layer [77] is also added between each convolution layer to inactivate 20% of neurons. Then, a data flattening step (flatten layer) is performed in order to generate a vector exploitable by the fully connected layer containing 100 neurons. The last layer is the final output layer, containing two neurons that return the results of the classification. The neurons of each layer are activated by a Rectified Linear Unit (ReLU) activation function, except for the last layer where the activation function is Softmax in order to establish probabilities for each neuron and thus to predict a class according to the highest probability. Fig. 13 summarizes the CNN architecture.

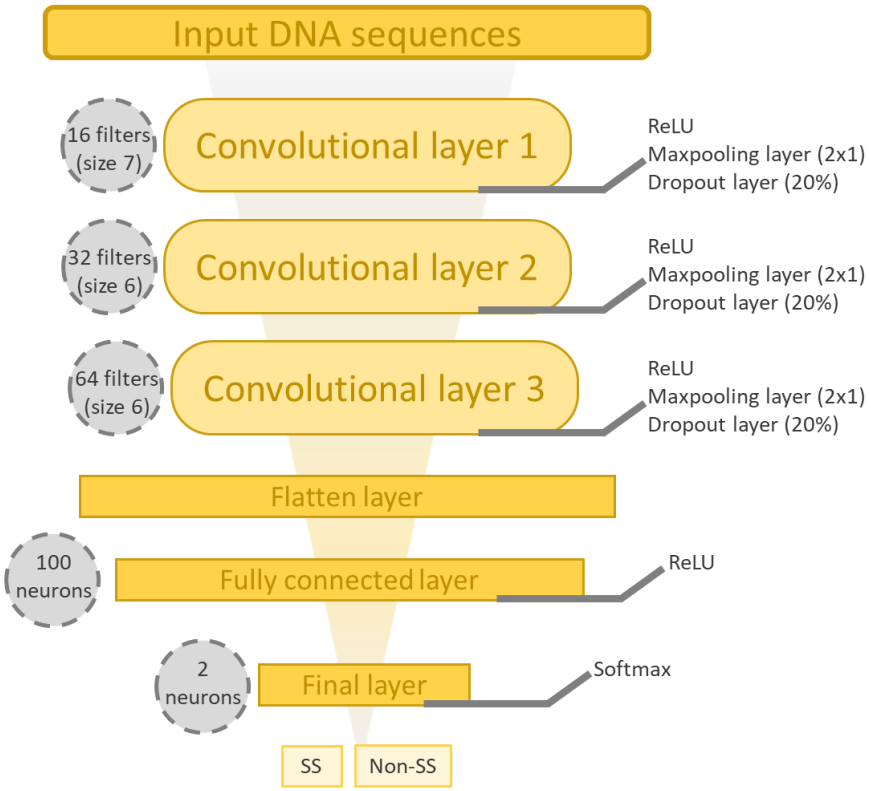


Fig. 13 – Representation of the CNN architecture. The architecture is composed of 2 convolutional layers, each followed by a dropout step and maxpooling layer. Then, a flatten layer is added to flatten the input. The output layer consists of 2 neurons activated by the Softmax function.

Training process

During the training process, the training set is divided into two parts, to generate an evaluation set (containing 15% of the sequences) that allows to control the learning of the network and avoid overfitting. The cross-entropy function is used as a loss function and the Adamax optimization algorithm [78] is applied with a learning-rate of $1e^{-5}$. Finally, the training is performed during 400 epochs with a batch-size of 32.

Evaluation

Metrics

SS are considered as true positives (TP) if they are correctly predicted and false positives (FP) otherwise. Nucleotides that do not correspond to a SS are considered as True Negatives (TN) if they are not predicted to be SS and False Negatives (FN) otherwise. To evaluate the performance of the CNN models, five metrics were used:

Accuracy is the ratio of the number of correct predictions to the total number of predictions:

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)}$$

Precision is the ratio of the number of correctly predicted SS to the total number of predicted SS:

$$Precision = \frac{TP}{(TP + FP)}$$

Sensitivity (also known as recall) is the ratio of the number of correctly predicted SS to the total number of SS:

$$Sensitivity = \frac{TP}{(TP + FN)}$$

Specificity is the ratio of the number of correctly predicted non-SS sequences to the total number of non-SS sequences:

$$Specificity = \frac{TN}{(TN + FP)}$$

F1 Score is the harmonic mean of the precision and sensitivity and shows a balance between these two metrics:

$$F1\ score = 2 * \frac{Precision * Sensitivity}{Precision + Sensitivity}$$

Independent benchmarks of SS from model and non-model organisms

To estimate the reliability and robustness of the CNN models, they were evaluated on 5 large-scale benchmarks, including sequences from: Human, *D. rerio* (Fish), *D. melanogaster* (Fly), *C. elegans* (Worm) and *A. thaliana* (Plant), that were downloaded from

https://public.bmi.inf.ethz.ch/user/behr/splicing/ [31]. A selection of 10000 SS and 10000 non-SS sequences was performed for each benchmark, including a number of non-canonical SS (human: 307; fish: 85; fly:120; worm: 67 and plant: 122).

To evaluate the performance of the models on non-model organisms, we constructed one other independent benchmark called 'PVP' (Protist and ViridiPlantae), containing sequences from protists and viridiplantae. The sequence selection process is similar to that used in G3PO. The reference sequences are the cytoplasmic tryptophanyl-tRNA synthetase of *Paramecium tetraurelia* (A0D783_PARTE) and the tryptophan-tRNA ligase of *Arabidopsis thaliana* (SYWM_ARATH). All orthologs were extracted from the OrthoInspector database version 3 [73], multiple sequence alignments were obtained with PipeAlign version 2 [79], and manually analyzed to identify 62 'Confirmed' sequences (33 plants and 29 protists). Finally, the benchmark contains 692 (with 21 non-canonical) donor SS and 714 (with 18 non-canonical) acceptor SS, and the same number of non-SS sequences to balance the dataset.

The benchmarks were used to compare Spliceator with other existing SS prediction methods, including NNSplice, MaxEntScan, DSSP and SpliceFinder. Note that SpliceRover was not included in these large-scale benchmark tests, since the method is only available as a web server.

Explicability

The visualization heatmaps of the nucleotides most used by the models were generated using the Grad-CAM (Gradient Class Activation Map) technique [80]. The maps were generated from the training sets: the higher the score, the warmer the color (yellow) and the lower the score, the colder the color (deep blue). In order to highlight the most representative patterns identified during the training process, the heatmaps were averaged from 10000 samples for each class of each SS.

Abbreviations

NGS: Next generation sequencing

RNA-seq: mRNA-sequencing

UTR: UnTranslated Region

SS: Splice Site

ML: Machine Learning

DL: Deep Learning

CNN: Convolutional Neural Network

BPS: BranchPoint site

ISE: Intronic Splicing Enhancer

ISS: Intronic Splicing Silencer

ESE: Exonic Splicing Enhancer

ESS: Exonic Splicing Silencer

FP: False Positive

AS: All Sequences

GS: Gold Standard

nt: nucleotide

PPT: PolyPyrimidine Tract

PVP: Protists and ViridiPlantea

BBS: Bardet-Biedl Syndrome

ReLU: Rectified Linear Unit

TP: True Positive

FP: False Positive

TN: True Negative

FN: False Negative

DECLARATIONS

Ethics approval and consent to participate

Not applicable

Consent for publication

Not applicable

Availability of data and materials

Spliceator source code (python) and the datasets analyzed or generated are available at: http://git.lbgi.fr/scalzitti/Spliceator, and a web service is available at: www.lbgi.fr/spliceator/.

Competing interests

The authors declare they have no competing interests.

Funding

This work was supported by French Infrastructure Institut Français de Bioinformatique (IFB) ANR-11-INBS-0013, and the ANR projects Elixir-Excelerate: GA-676559 and RAinRARE: ANR-18-RAR3–0006-02, and Institute funds from the French Centre National de la Recherche Scientifique, the University of Strasbourg.

Authors' contributions

NS, OP and JDT conceived the study and designed analyses. AJG and PC contributed to designing and implementing the CNN method. NS and OP generated and curated training and test datasets. NS performed predictions, evaluations and benchmarks, with support from AK, RO, TW and LM. NS, OP and JDT analyzed the data, with input from all authors. NS, AJG and

JDT wrote the manuscript, with input from all authors. All authors read and approved the final manuscript.

Acknowledgements

The authors would like to thank the BiGEst-ICube bioinformatics platform for assistance.

SUPPLEMENTARY INFORMATION

Additional file 1. Figures S1-S2 and Tables S1-S6

Additional file 2. Tables S1-S3, performance results of donor and acceptor models

References

1. Brůna T, Hoff KJ, Lomsadze A, Stanke M, Borodovsky M. BRAKER2: Automatic Eukaryotic Genome Annotation with GeneMark-EP+ and AUGUSTUS Supported by a Protein Database. bioRxiv. Cold Spring Harbor Laboratory; 2020;2020.08.10.245134.
2. Campbell MS, Holt C, Moore B, Yandell M. Genome Annotation and Curation Using MAKER and MAKER-P. Curr Protoc Bioinformatics. 2014;48:4.11.1-4.11.39.
3. Haas BJ, Delcher AL, Mount SM, Wortman JR, Smith Jr RK, Hannick LI, et al. Improving the Arabidopsis genome annotation using maximal transcript alignment assemblies. Nucleic Acids Research. 2003;31:5654–66.
4. Yates AD, Achuthan P, Akanni W, Allen J, Allen J, Alvarez-Jarreta J, et al. Ensembl 2020. Nucleic Acids Res. 2020;48:D682–8.
5. Thibaud-Nissen F, DiCuccio M, Hlavina W, Kimchi A, Kitts PA, Murphy TD, et al. P8008 The NCBI Eukaryotic Genome Annotation Pipeline. Journal of Animal Science. 2016;94:184–184.
6. Stanke M, Schöffmann O, Morgenstern B, Waack S. Gene prediction in eukaryotes with a generalized hidden Markov model that uses hints from external sources. BMC Bioinformatics. 2006;7:62.
7. Burge C, Karlin S. Prediction of complete gene structures in human genomic DNA. Journal of Molecular Biology. 1997;268:78–94.
8. Korf I. Gene finding in novel genomes. BMC Bioinformatics. 2004;9.
9. Majoros WH, Pertea M, Salzberg SL. TigrScan and GlimmerHMM: two open source ab initio eukaryotic gene-finders. Bioinformatics. 2004;20:2878–9.
10. Yandell M, Ence D. A beginner's guide to eukaryotic genome annotation. Nature Reviews Genetics. 2012;13:329–42.
11. Salzberg SL. Next-generation genome annotation: we still struggle to get it right. Genome Biol. 2019;20:92, s13059-019-1715–2.
12. Meyer C, Scalzitti N, Jeannin-Girardon A, Collet P, Poch O, Thompson JD. Understanding the causes of errors in eukaryotic protein-coding gene prediction: a case study of primate proteomes. BMC Bioinformatics. 2020;21:513.
13. Zhang D, Guelfi S, Garcia-Ruiz S, Costa B, Reynolds RH, D'Sa K, et al. Incomplete annotation has a disproportionate impact on our understanding of Mendelian and complex neurogenetic disorders. Sci

760 Adv [Internet]. 2020 [cited 2021 Jan 6];6. Available from:
761 https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7286675/
762 14. Rogozin IB, Carmel L, Csuros M, Koonin EV. Origin and evolution of spliceosomal introns. Biol
763 Direct. 2012;7:11.
764 15. Matera AG, Wang Z. A day in the life of the spliceosome. Nat Rev Mol Cell Biol. 2014;15:108–21.
765 16. Pan Q, Shai O, Lee LJ, Frey BJ, Blencowe BJ. Deep surveying of alternative splicing complexity in
766 the human transcriptome by high-throughput sequencing. Nature Genetics. Nature Publishing Group;
767 2008;40:1413–5.
768 17. Ben-Dov C, Hartmann B, Lundgren J, Valcárcel J. Genome-wide Analysis of Alternative Pre-mRNA
769 Splicing. J Biol Chem. American Society for Biochemistry and Molecular Biology; 2008;283:1229–33.
770 18. Burset M, Seledtsov IA, Solovyev VV. SpliceDB: database of canonical and non-canonical
771 mammalian splice sites. Nucleic Acids Res. 2001;29:255–9.
772 19. Nguyen H, Das U, Wang B, Xie J. The matrices and constraints of GT/AG splice sites of more than
773 1000 species/lineages. Gene. 2018;660:92–101.
774 20. Burset M, Seledtsov IA, Solovyev VV. Analysis of canonical and non-canonical splice sites in
775 mammalian genomes. Nucleic Acids Res. 2000;28:4364–75.
776 21. Frey K, Pucker B. Animal, Fungi, and Plant Genome Sequences Harbor Different Non-Canonical
777 Splice Sites. Cells. Multidisciplinary Digital Publishing Institute; 2020;9:458.
778 22. Sheth N, Roca X, Hastings ML, Roeder T, Krainer AR, Sachidanandam R. Comprehensive splice-site
779 analysis using comparative genomics. Nucleic Acids Res. 2006;34:3955–67.
780 23. Pucker B, Brockington SF. Genome-wide analyses supported by RNA-Seq reveal non-canonical
781 splice sites in plant genomes. BMC Genomics. 2018;19:980.
782 24. Pucker B, Holtgräwe D, Weisshaar B. Consideration of non-canonical splice sites improves gene
783 prediction on the Arabidopsis thaliana Niederzenz-1 genome sequence. BMC Research Notes.
784 2017;10:667.
785 25. Wang K, Singh D, Zeng Z, Coleman SJ, Huang Y, Savich GL, et al. MapSplice: Accurate mapping of
786 RNA-seq reads for splice junction discovery. Nucleic Acids Research. 2010;38:e178–e178.
787 26. Trapnell C, Pachter L, Salzberg SL. TopHat: discovering splice junctions with RNA-Seq.
788 Bioinformatics. 2009;25:1105–11.
789 27. Ameur A, Wetterbom A, Feuk L, Gyllensten U. Global and unbiased detection of splice junctions
790 from RNA-seq data. Genome Biol. 2010;11:R34.
791 28. Ozsolak F, Milos PM. RNA sequencing: advances, challenges and opportunities. Nat Rev Genet.
792 Nature Publishing Group; 2011;12:87–98.
793 29. Degroeve S, De Baets B, Van de Peer Y, Rouzé P. Feature subset selection for splice site
794 prediction. Bioinformatics. 2002;18 Suppl 2:S75-83.
795 30. Degroeve S, Saeys Y, De Baets B, Rouzé P, Van de Peer Y. SpliceMachine: predicting splice sites
796 from high-dimensional local context representations. Bioinformatics. 2005;21:1332–8.
797 31. Sonnenburg S, Schweikert G, Philips P, Behr J, Rätsch G. Accurate splice site prediction using
798 support vector machines. BMC Bioinformatics. 2007;8:S7.
799 32. Maji S, Garg D. Hybrid Approach Using SVM and MM2 in Splice Site Junction Identification.
800 Current Bioinformatics. 2014;9:76–85.
801 33. Pashaei E, Yilmaz A, Ozen M, Aydin N. A novel method for splice sites prediction using sequence
802 component and hidden Markov model. Annu Int Conf IEEE Eng Med Biol Soc. 2016;2016:3076–9.
803 34. Zhang Q, Peng Q, Zhang Q, Yan Y, Li K, Li J. Splice sites prediction of Human genome using length-
804 variable Markov model and feature selection. Expert Syst Appl. 2010;37:2771–82.
805 35. Pashaei E, Ozen M, Aydin N. Splice site identification in human genome using random forest.
806 Health and Technology. 2016;1:141–52.
807 36. Meher PK, Sahu TK, Rao AR. Prediction of donor splice sites using random forest with a new
808 sequence encoding approach. BioData Mining. 2016;9:4.
809 37. Chen T-M, Lu C-C, Li W-H. Prediction of splice sites with dependency graphs and their expanded
810 bayesian networks. Bioinformatics. 2005;21:471–82.

811 38. Saeys Y, Degroeve S, Van de Peer Y. Digging into Acceptor Splice Site Prediction: An Iterative
812 Feature Selection Approach. In: Boulicaut J-F, Esposito F, Giannotti F, Pedreschi D, editors.
813 Knowledge Discovery in Databases: PKDD 2004. Berlin, Heidelberg: Springer; 2004. p. 386–97.
814 39. Naito T. Human Splice-Site Prediction with Deep Neural Networks. Journal of Computational
815 Biology. Mary Ann Liebert, Inc., publishers; 2018;25:954–61.
816 40. Zuallaert J, Godin F, Kim M, Soete A, Saeys Y, De Neve W. SpliceRover: interpretable
817 convolutional neural networks for improved splice site prediction. Bioinformatics. 2018;34:4180–8.
818 41. Wang R, Wang Z, Wang J, Li S. SpliceFinder: ab initio prediction of splice sites using convolutional
819 neural network. BMC Bioinformatics. 2019;20:652.
820 42. LeCun Y, Bengio Y, Hinton G. Deep learning. Nature. Nature Publishing Group; 2015;521:436–44.
821 43. Li Y, Huang C, Ding L, Li Z, Pan Y, Gao X. Deep learning in bioinformatics: Introduction, application,
822 and perspective in the big data era. Methods. 2019;166:4–21.
823 44. Scalzitti N, Jeannin-Girardon A, Collet P, Poch O, Thompson JD. A benchmark study of ab initio
824 gene prediction methods in diverse eukaryotic organisms. BMC Genomics. 2020;21:293.
825 45. Kilkenny MF, Robinson KM. Data quality: “Garbage in – garbage out.” HIM J. SAGE Publications
826 Ltd STM; 2018;47:103–5.
827 46. Reese MG, Eeckman FH, Kulp D, Haussler D. Improved Splice Site Detection in Genie. Journal of
828 Computational Biology. Mary Ann Liebert, Inc., publishers; 1997;4:311–23.
829 47. Yeo G, Burge CB. Maximum Entropy Modeling of Short Sequence Motifs with Applications to RNA
830 Splicing Signals. Journal of Computational Biology. Mary Ann Liebert, Inc., publishers; 2004;11:377–
831 94.
832 48. Cartegni L, Chew SL, Krainer AR. Listening to silence and understanding nonsense: exonic
833 mutations that affect splicing. Nat Rev Genet. 2002;3:285–98.
834 49. Zeng Y, Yuan H, Yuan Z, Chen Y. A high-performance approach for predicting donor splice sites
835 based on short window size and imbalanced large samples. Biology Direct. 2019;14:6.
836 50. Mercer TR, Clark MB, Andersen SB, Brunck ME, Haerty W, Crawford J, et al. Genome-wide
837 discovery of human splicing branchpoints. Genome Res. 2015;25:290–303.
838 51. Anna A, Monika G. Splicing mutations in human genetic disorders: examples, detection, and
839 confirmation. J Appl Genet. 2018;59:253–68.
840 52. Gooding C, Clark F, Wollerton MC, Grellscheid S-N, Groom H, Smith CW. A class of human exons
841 with predicted distant branch points revealed by analysis of AG dinucleotide exclusion zones.
842 Genome Biol. 2006;7:R1.
843 53. Campbell M, Hoane AJ, Hsu F. Deep Blue. Artificial Intelligence. 2002;134:57–83.
844 54. Silver D, Huang A, Maddison CJ, Guez A, Sifre L, van den Driessche G, et al. Mastering the game of
845 Go with deep neural networks and tree search. Nature. Nature Publishing Group; 2016;529:484–9.
846 55. AlQuraishi M. AlphaFold at CASP13. Bioinformatics. 2019;35:4862–5.
847 56. Senior AW, Evans R, Jumper J, Kirkpatrick J, Sifre L, Green T, et al. Improved protein structure
848 prediction using potentials from deep learning. Nature. Nature Publishing Group; 2020;577:706–10.
849 57. Li H, Tian S, Li Y, Fang Q, Tan R, Pan Y, et al. Modern Deep Learning in Bioinformatics. J Mol Cell
850 Biol. 2020;
851 58. Koumakis L. Deep learning models in genomics; are we there yet? Computational and Structural
852 Biotechnology Journal. 2020;18:1466–73.
853 59. Tang B, Pan Z, Yin K, Khateeb A. Recent Advances of Deep Learning in Bioinformatics and
854 Computational Biology. Front Genet [Internet]. Frontiers; 2019 [cited 2020 Nov 10];10. Available
855 from: https://www.frontiersin.org/articles/10.3389/fgene.2019.00214/full
856 60. He Y, Shen Z, Zhang Q, Wang S, Huang D-S. A survey on deep learning in DNA/RNA motif mining.
857 Briefings in Bioinformatics [Internet]. 2020 [cited 2021 Jan 6]; Available from:
858 https://doi.org/10.1093/bib/bbaa229
859 61. Krizhevsky A, Sutskever I, Hinton GE. ImageNet classification with deep convolutional neural
860 networks. Commun ACM. 2017;60:84–90.

62. Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, et al. Going Deeper with Convolutions. arXiv:14094842 [cs] [Internet]. 2014 [cited 2021 Jan 6]; Available from: http://arxiv.org/abs/1409.4842
63. Pollastro P, Rampone S. Hs3d, a dataset of homo sapiens splice regions, and its extraction procedure from a major public database. Int J Mod Phys C. World Scientific Publishing Co.; 2002;13:1105–17.
64. Pertea M, Lin X, Salzberg SL. GeneSplicer: a new computational method for splice site prediction. Nucleic Acids Res. 2001;29:1185–90.
65. Khodabandelou G, Routhier E, Mozziconacci J. Genome annotation across species using deep convolutional neural networks. PeerJ Comput Sci [Internet]. 2020 [cited 2021 Jun 14];6. Available from: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7924482/
66. Stiehler F, Steinborn M, Scholz S, Dey D, Weber APM, Denton AK. Helixer: cross-species gene annotation of large eukaryotic genomes using deep learning. Bioinformatics. 2020;36:5291–8.
67. The UniProt Consortium. UniProt: the universal protein knowledgebase in 2021. Nucleic Acids Research. 2021;49:D480–9.
68. Tørresen OK, Star B, Mier P, Andrade-Navarro MA, Bateman A, Jarnot P, et al. Tandem repeats lead to sequence assembly errors and impose multi-level challenges for genome and protein databases. Nucleic Acids Research. 2019;47:10994–1006.
69. Zhang C, Li W-H, Krainer AR, Zhang MQ. RNA landscape of evolution for optimal exon and intron discrimination. PNAS. National Academy of Sciences; 2008;105:5797–802.
70. Gao K, Masuda A, Matsuura T, Ohno K. Human branch point consensus sequence is yUnAy. Nucleic Acids Res. 2008;36:2257–67.
71. Soemedi R, Cygan KJ, Rhine C, Glidden DT, Taggart AJ, Lin C-L, et al. The Effects of Structure on pre-mRNA Processing and Stability. Methods. 2017;125:36–44.
72. Tellier M, Maudlin I, Murphy S. Transcription and splicing: A two-way street. WIREs RNA. 2020;11:e1593.
73. Nevers Y, Kress A, Defosset A, Ripp R, Linard B, Thompson JD, et al. OrthoInspector 3.0: open portal for comparative genomics. Nucleic Acids Res. 2019;47:D411–8.
74. Crooks GE, Hon G, Chandonia J-M, Brenner SE. WebLogo: A Sequence Logo Generator. Genome Res. 2004;14:1188–90.
75. Abadi M, Barham P, Chen J, Chen Z, Davis A, Dean J, et al. TensorFlow: A system for large-scale machine learning. arXiv:160508695 [cs] [Internet]. 2016 [cited 2021 Jan 6]; Available from: http://arxiv.org/abs/1605.08695
76. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research. 2011;12:2825–30.
77. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research. 2014;15:1929–58.
78. Kingma DP, Ba J. Adam: A Method for Stochastic Optimization. arXiv:14126980 [cs] [Internet]. 2017 [cited 2021 Jan 6]; Available from: http://arxiv.org/abs/1412.6980
79. Plewniak F, Bianchetti L, Brelivet Y, Carles A, Chalmel F, Lecompte O, et al. PipeAlign: a new toolkit for protein family analysis. Nucleic Acids Research. 2003;31:3829–32.
80. Selvaraju RR, Cogswell M, Das A, Vedantam R, Parikh D, Batra D. Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization. Int J Comput Vis. 2020;128:336–59.

Chapitre 8 – SpliceSLEIA : Stratégie évolutive pour la prédiction de sites d'épissage

"All models are wrong, but some are useful"

– George Box –

Après avoir exploité les réseaux de neurones, je me suis intéressé aux algorithmes de programmation génétique (PG) afin d'améliorer le processus d'annotation des génomes. Ce type d'algorithme est très peu utilisé dans le domaine de la bio-informatique, cependant le laboratoire dispose de la plateforme EASEA, ce qui nous a permis de développer une approche hautement parallélisable, améliorant grandement les performances et la vitesse de calcul. De plus, les algorithmes de PG sont stochastiques, ce qui leur permet de trouver des solutions originales et surprenantes qu'un humain n'aurait pas forcément imaginé (Miikkulainen, 2020). Ces points forts ont conforté notre choix pour l'utilisation de la PG pour la prédiction de SE.

Dans ce contexte, nous avons élaboré un prototype nommé SpliceSLEIA (pour *Splice Site Localization by EvolutIonary Algorithm*), permettant d'identifier des SE en exploitant la PG. Nous avons fait évoluer une population d'automates à état finis (individus) représentés par des expressions régulières (regex) sous forme d'arbres, qui devront reconnaître les motifs des SE (d'une taille de 10 ou de 20 nucléotides) après leur entraînement et les processus évolutifs. Quelques travaux ont déjà exploité l'évolution de regex (Bartoli *et al.*, 2012; Bakker and Jansen, 2018), cependant ils ne sont pas appliqués à des problèmes biologiques comme la prédiction de SE.

8.1 Mise en place de la stratégie de programmation génétique

De nombreux tests ont été réalisés afin de sélectionner les meilleurs hyperparamètres de l'algorithme. Cependant, en raison du caractère stochastique de ces derniers, il a été très difficile

d'établir une stratégie optimale pour la recherche paramétrique. Ainsi, lors de la mise en place de l'algorithme de PG, nous avons initialisé aléatoirement une population de 1 000 individus selon la méthode *ramped half-and-half*, avec une profondeur maximale des arbres de 7. Les opérateurs évolutionnaires utilisés sont le croisement à un point et tous les individus sont concernés. La mutation de sous-arbres impacte 1% des individus lors de chaque nouvelle génération. La sélection des individus pour la génération *n+1* est réalisée par une sélection par tournoi avec une pression de sélection de 3 et un seul meilleur individu sortant (*i.e.* avec le score d'évaluation le plus élevé). Enfin, le nombre de générations total est de 10 000, qui est une des conditions d'arrêt de l'algorithme, la deuxième étant le temps limité à 24h.

A partir de la syntaxe des expressions régulières décrite dans la section 2.3.3 du Matériels et méthodes, nous avons mis en place un algorithme de construction des individus qui consiste à réaliser un parcours préfixé de l'arbre afin d'obtenir l'expression régulière. La première étape consiste à définir le nœud racine puis d'autres éléments de la syntaxe sont ajoutés qui ont une arité (*i.e.* nœuds avec enfants) >0, jusqu'à arriver à la profondeur maximale de l'arbre. Une fois cette profondeur atteinte, des feuilles terminales (*i.e.* nœuds sans enfant) sont insérées, avec la présence de constantes comme 'A', 'C', 'G', 'T' ou '.'. De plus, chaque individu a été construit afin de respecter les conditions de chaque opérateur. Par exemple, si des crochets sont ouverts, il faut ajouter une feuille terminale afin d'insérer sa constante entre les crochets. Ce système à base d'arbre a permis une implémentation plus facile des individus, afin de générer des expressions régulières cohérentes et exploitables pour un problème de reconnaissances de motifs comme les SE.

8.2 Entraînement de SpliceSLEIA

La première étape de la mise en place de SpliceSLEIA est de réaliser un entraînement afin de guider le processus évolutif. Chaque individu est confronté à une fonction d'évaluation afin de sélectionner de préférence les individus les plus aptes *i.e.* avec les scores les plus élevés, à l'instar de la sélection naturelle décrite par Charles Darwin.

Pour cela, nous avons évalué chaque individu à l'aide d'un jeu d'entraînement constitué d'environ 7 700 séquences issues de G3PO, d'une taille de 200 nucléotides où un site d'épissage est présent au milieu. La performance (ou *fitness*) d'un individu a été estimée par la distance de Levenshtein (Levenshtein, 1965) qui mesure la différence entre deux chaînes de caractères, exprimée comme le nombre d'opérations à réaliser pour que les deux chaînes soient identiques. Il existe 3 types d'opérations :

- substitution : remplacement d'un caractère par un autre,
- insertion : ajout d'un caractère,
- délétion : suppression d'un caractère.

Ainsi, une distance de Levenshtein élevée signifie que les deux chaînes divergent fortement, à l'inverse si la distance est égale à 0, cela signifie qu'elles sont identiques. L'expression régulière va ainsi correspondre avec certains fragments X_i des séquences d'entraînement de taille 200. Chacun de ces fragments X_i va alors être comparé avec le segment génomique *Y* (d'une taille de 10 ou de 20 nucléotides) correspondant au SE à identifier. La distance de Levenshtein est ensuite calculée entre chaque fragment X_i et le segment *Y*. Le score est normalisé (entre 0 et 1) en fonction de la longueur maximale selon *:*

$$Normalized_{score} = \frac{Levenshtein\_distance}{\max(t.size, s.size)}$$

avec *t* la longueur du fragment X_i et *s* la longueur du segment génomique Y. L'ensemble des scores normalisés pour chaque fragment X_i obtenus sont ensuite moyennées :

$$Mean_{score} = mean(\sum_{i=1}^{N} Normalized\_score_i)$$

avec *N* le nombre de fragments identifié. Enfin, la moyenne finale est calculée selon la formule :

$$Final_{score} = mean(\sum_{j=1}^{M} Mean\_score_j)$$

Avec *M* le nombre de séquences du jeu d'entraînement. Elle correspond à la moyenne de toutes les identifications de segment sur toutes les séquences du jeu d'entraînement. Si un individu n'identifie aucun fragment, son $Mean_{score}$ est équivalent à 1. La figure 62 résume ce protocole. Notre stratégie pénalise les individus qui n'identifient aucun segment sur le fragment *Y*, cependant cela induit un fort taux de FP. Ce taux a tendance à diminuer durant tout le processus évolutif. Une deuxième solution envisageable serait de pénaliser les individus qui identifient plusieurs segments sur une même séquence d'entraînement. Afin de balayer un espace de recherche assez large, 128 processus évolutifs différents ont été exécutés pour chaque SE, permettant l'obtention de 128 meilleurs individus par SE.

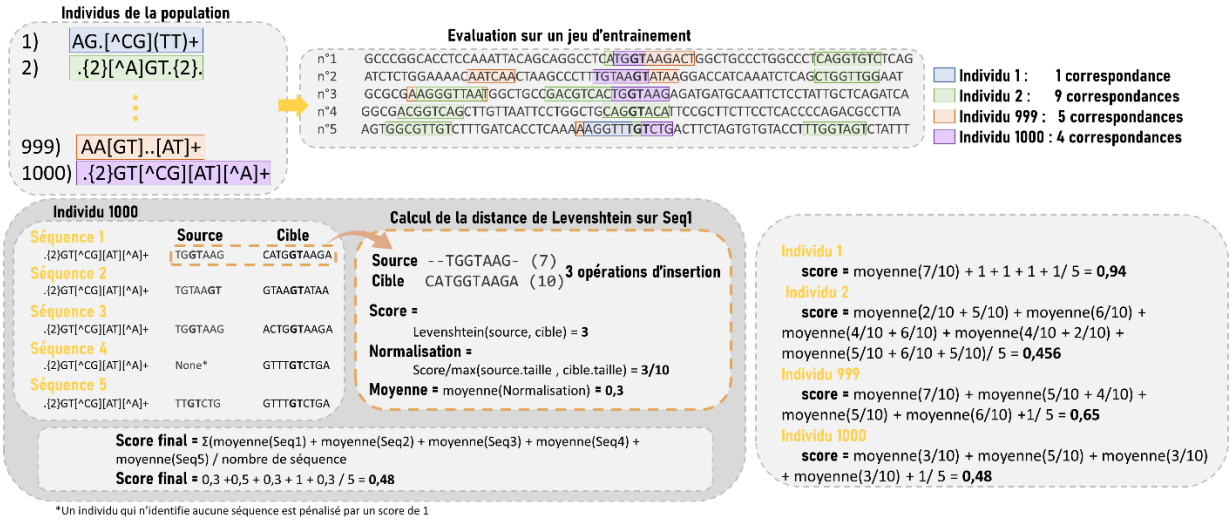


Figure 62 - *Protocole de guidage du processus évolutif par la distance de Levenshtein. Une population contenant n individus est initialisée aléatoirement. Pour sélectionner les meilleurs individus de la génération suivante, chaque individu est soumis à la fonction d'évaluation. Dans un premier temps, l'expression régulière (qui représente l'individu) est évaluée sur chaque séquence du jeu d'entraînement. Chaque correspondance est ensuite testée avec la distance de Levenshtein sur les segments cibles (i.e. la bonne réponse), plus le score est faible, meilleur est l'individu. Les scores sont ensuite normalisés, moyennés puis sommés. Ainsi dans notre exemple, l'individu 2 est le meilleur individu avec un score de 0,456. Un individu est défavorisé s'il ne reconnaît aucun segment et se voit imputer d'un score de 1.*

8.3 Evaluation des performances des individus en programmation génétique

La deuxième étape évalue les performances de ces individus sur le jeu de test (également construit à partir des données de G3PO). Pour chaque SE, nous avons établi 2 jeux tests composés de 2 252 séquences chacun également issu de G3PO, un avec les fragments à reconnaître d'une taille de 10 nucléotides et l'autre avec une taille de 20 nucléotides. La figure 63 décrit le processus de calcul des performances. Si le segment identifié par l'expression régulière correspond au SE, il s'agit d'un VP sinon d'un FP et si aucun segment n'est trouvé alors il s'agit d'un FN.

Figure 63 - *Evaluation des performances de chaque individu, par l'identification des VP, FP et FN sur le jeu de test avec une taille de fragment égale à 10. Un segment correctement identifié est un VP, un segment ne correspondant pas au SE est un FP et un segment correspondant à un SE non identifié est un FN. A partir de ces valeurs, différentes métriques sont calculées comme la précision, la sensibilité et le F1-Score (source : (Herbay et al., 2021))*

L'évaluation sur les jeux tests nous permet d'obtenir des métriques comme le F1-Score (figure 64). Les résultats obtenus sur les jeux de test avec une taille de fragment de 10 nucléotides sont meilleurs que ceux avec une taille de 20 nucléotides. On observe également de mauvais résultats (bien qu'il y ait des points atypiques >0,7) pour le SE donneur sur le jeu de test avec une taille de 20 nucléotides. Cela est probablement dû au fait que l'algorithme n'arrive pas à trouver d'optimum global. Nous supposons qu'une taille de 20 nucléotides est trop importante et qu'au-delà de 10 nucléotides, le contexte du SE donneur est trop divergent rendant leur identification trop compliquée avec notre approche. De manière intéressante, ce phénomène ne se produit pas avec le SE accepteur, malgré un contexte génomique divergent notamment à cause du tractus polypyrimidique.

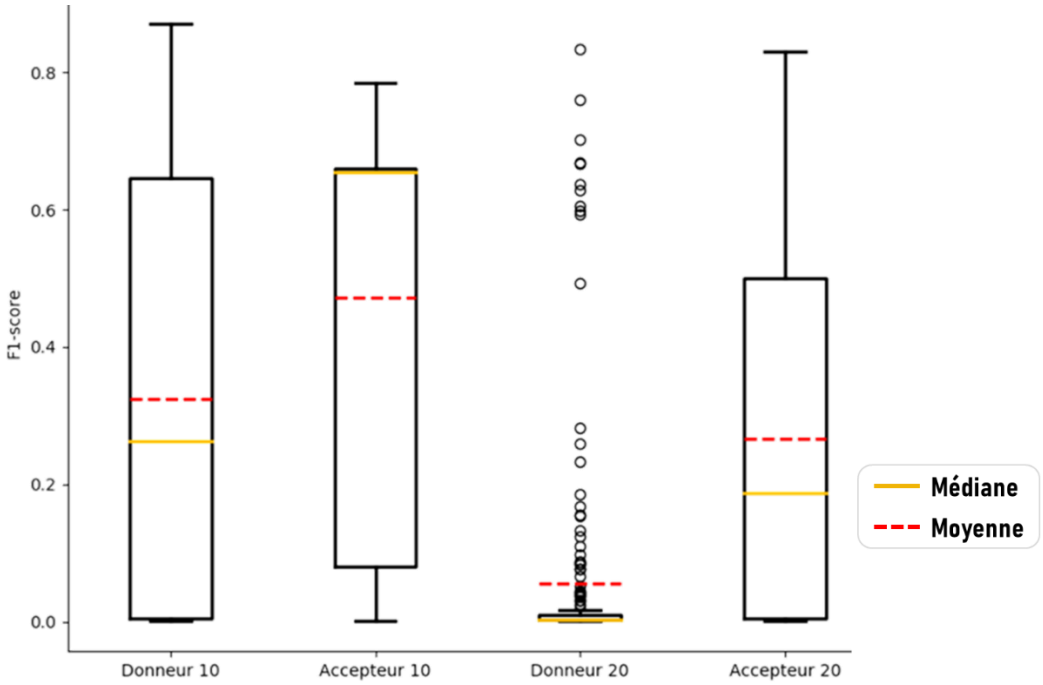


Figure 64 - *Représentation du F1-Score de chaque expérience préliminaire. L'ensemble des 128 regex a été évalué sur le jeu de test (taille 10 ou 20 nucléotides) pour chaque SE et les résultats sont représentés par la boîte à moustaches. Les résultats pour le SE donneur avec une taille de 20 sont vraiment très faibles et laisse supposer que l'algorithme n'a pas convergé.*

A partir de ces résultats, nous pouvons classer les individus du meilleur au moins bon et les X (avec $X \in [1-10]$) premiers individus (sur le jeu test de 10 nucléotides) ont été sélectionnés pour être intégrés au programme final.

8.4 Application de SpliceSLEIA pour la prédiction des SE

SpliceSLEIA, disponible sur le dépôt git : git.unistra.fr/nscalzitti/splicesleia, identifie les SE présents dans une séquence génomique à l'aide des meilleures regex précalculées comme cité précédemment. Chacune d'entre elles va donc générer une collection de SE prédits. L'ensemble de ces collections est comparé afin de sélectionner les SE à l'intersection *i.e.* les SE similaires à chaque collection, ou présentant au moins 90% d'identité. La figure 65 résume le protocole de SpliceSLEIA.

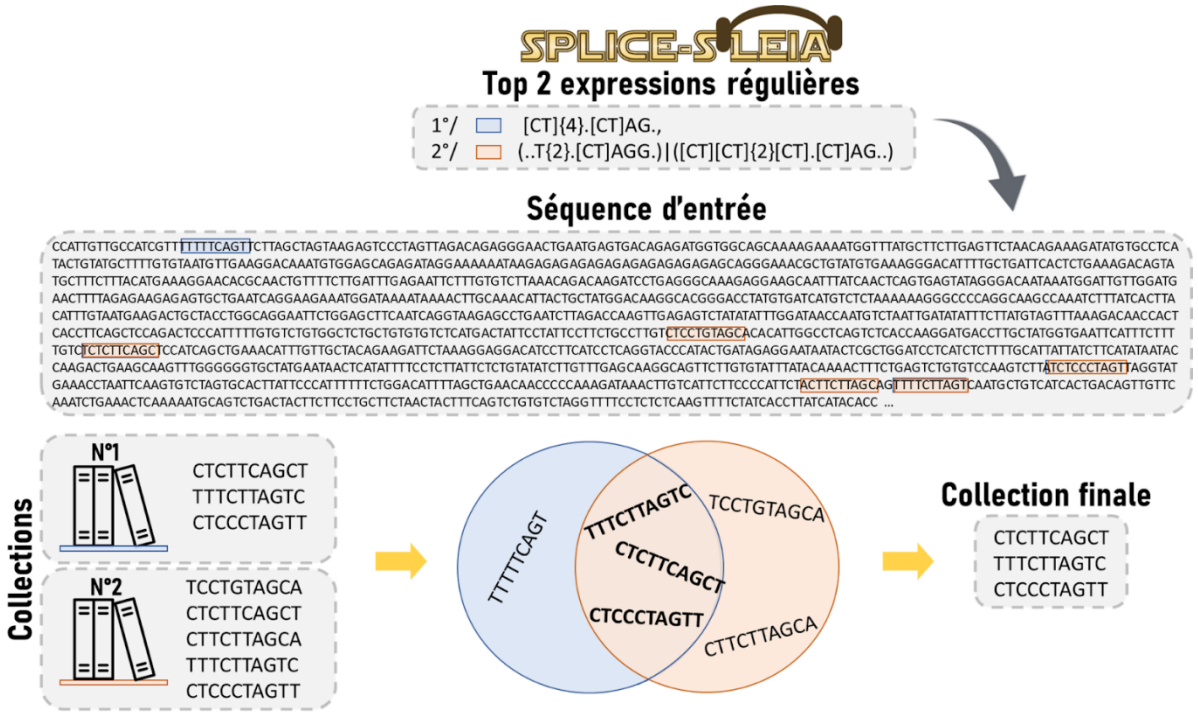


Figure 65 - *Protocole de prédiction des SE par SpliceSLEIA. Les meilleures expressions régulières vont identifier des fragments de séquences sur la région génomique d'entrée. Chaque fragment constitue un élément de la collection de l'expression régulière. L'ensemble des collections sont ensuite mis en commun et les expressions régulières similaires sont extraites afin de composer la collection finale.*

8.5 Résultats préliminaires

Nous avons classé les 128 regex de chaque SE en fonction de leur performance (F1-Score) sur le jeu test (taille de séquence=10). Les trois meilleures regex sélectionnées sont les suivantes :

Donneur :

1. *.{2}((.[ACG]GT){2})|((ATTAT)|((A[AG][AG]T)|((.A)|(.[ACG])))))(AGGT[AG]..{2})|((AGA){5})*, F1-Score = 87,7%
2. *(....[^ACT]T[^CT]A[^ACT].)|(.[^T][^GT][^ACT][^ACT]TA...)*, F1-Score = 87.1%
3. *(.[^T][^GT]G{2}T[A](((.|A)|(T.))|([^T]TG|.))){3})|(.(G|.){3}GT[^CT][A]G.)*, F1-Score = 87.1%

Accepteur :

1. *(..T{2}.[CT]A[^ACT][^ACT].)|([CT][CT]{2}[CT].[CT]A[^ACT]..)*, F1-Score = 78,4%
2. *([CT]|(([^CT]T)|T)){5}AG(G[^G])|(.TT{3}AG)(.T|[^CT]G)|(..T{3}CAG)(T{3}G{3}.|(..))|([CT][CT]{3}.[CT]AG)(..)|((.[CT]){3}AGGT)*, F1-Score = 77,1%
3. *(.[AGT][^AG]{4}AGG[^AG])|([^AG]{4}.[^AG]AG..)*, F1-Score = 74,7%

De manière intéressante ces regex ne sont pas trop longues, leur permettant de bien généraliser les données, mais cela risque également de générer des FP.

Pour commencer, nous avons d'abord étudié l'impact du nombre de regex sélectionnées dans la collection finale (figure 66 et 67A). Il s'avère que plus le nombre de regex est grand,

moins les performances globales (dont le F1-Score) sont élevées. Cependant, on observe une zone de fracture entre 2 et 3 regex pour le SE donneur et entre 6 et 7 regex pour le SE accepteur. Cette zone indique une balance équitable entre les différentes métriques. Nous avons ensuite étudié l'impact qu'avait le seuil *i.e.* le nombre minimal de regex différentes qui doit reconnaître un même SE pour que ce dernier soit validé. Les figures 66 (donneur) et 67 (accepteur) B, C et D représentent respectivement 3, 4 et 5 regex retenues. Un seuil de 1 signifie qu'un SE doit être vu au minimum 1 fois pour être incorporé dans la collection finale. S'il y a 5 regex en tout et que le seuil est de 5, cela signifie que le SE doit être vu par toutes les regex pour être sélectionné. Dans les 3 exemples présentés sur les figures 66 et 67, on observe une corrélation entre l'augmentation du nombre de faux positifs et de vrais positifs. Cependant lorsque le seuil est équivalent à *n-1* ou *n* est le nombre de regex, les valeurs semblent être meilleures.

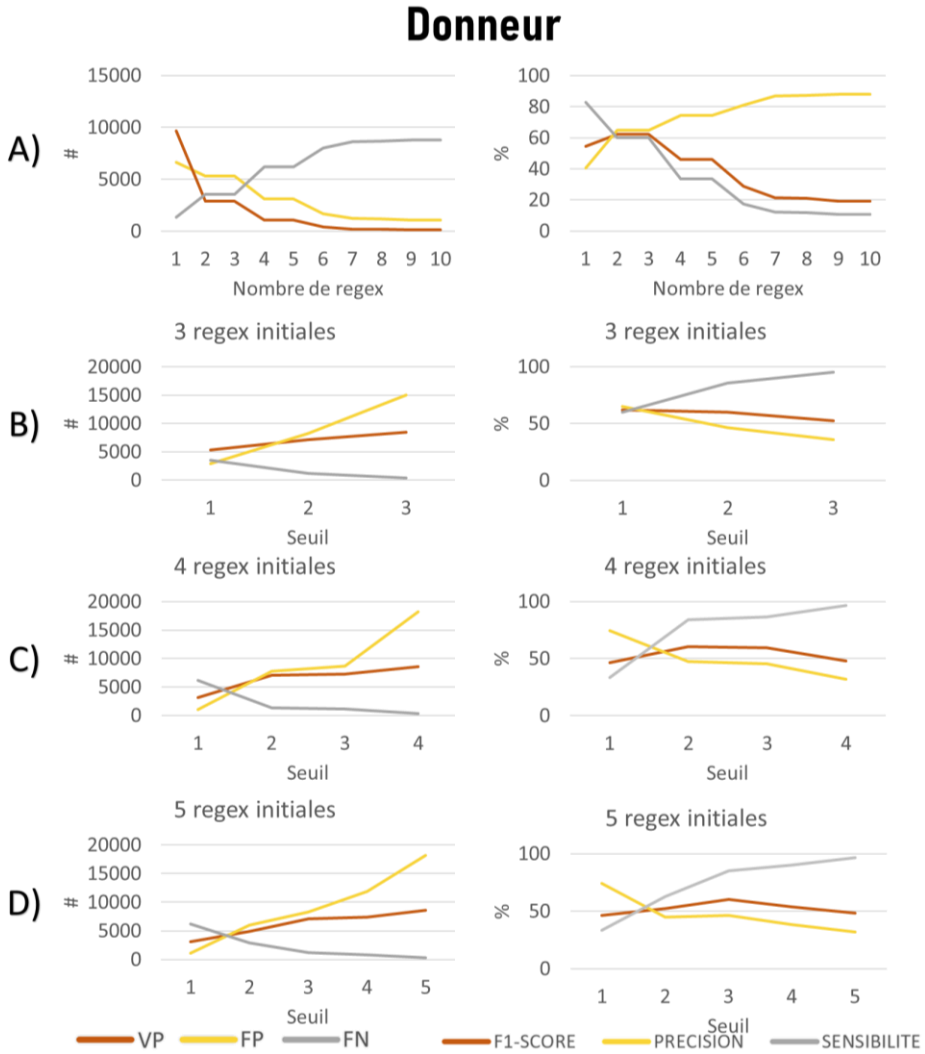


Figure 66 - *Etude de l'impact du nombre de regex pour la prédiction de SE donneur. **A)** Représentation du nombre de vrais positifs, faux positifs et faux négatifs et des différentes métriques en fonction du nombre de regex. Représentations des mêmes éléments en fonction du nombre de regex initiales, **B)** 3,*

C) 4 et D) 5 et du seuil. Le seuil représente le nombre minimal de regex qui identifie le même SE pour qu'il soit validé. Par exemple, pour 5 regex et un seuil de 3, il faut que le site soit vu au minimum par 3 regex différents pour qu'il soit validé.

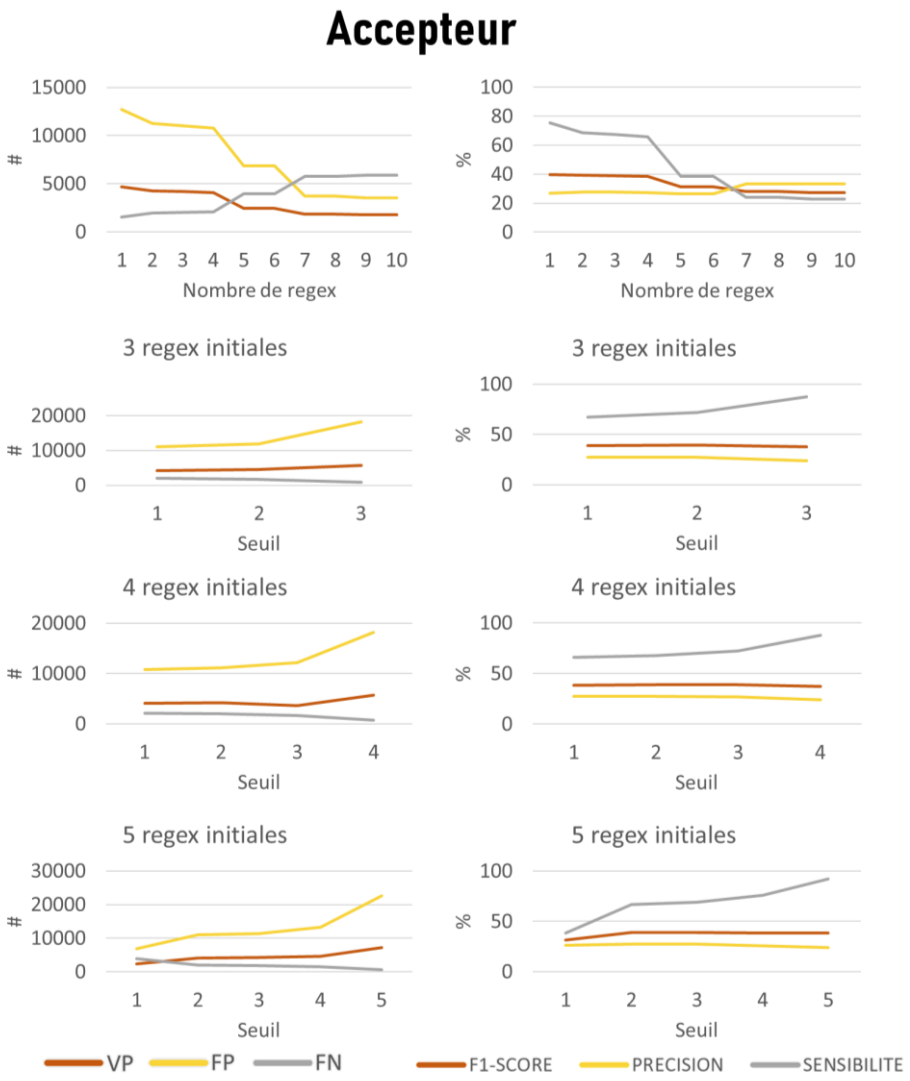


Figure 67 - *Etude de l'impact du nombre de regex pour la prédiction de SE accepteur. **A)** Représentation du nombre de vrais positifs, faux positifs et faux négatifs et des différentes métriques en fonction du nombre de regex. Représentations des mêmes éléments en fonction du nombre de regex initiales, **B)** 3, **C)** 4 et **D)** 5 et du seuil. Le seuil représente le nombre minimal de regex qui identifie le même SE pour qu'il soit validé.*

Les regex de SpliceSLEIA ont ensuite été testées sur 10 000 séquences de 5 *benchmarks* décrits dans la partie 1.2 du Matériels et méthodes. Bien que SpliceSLEIA trouve entre 50 et 70% (~7 000) des SE donneurs et entre 30 et 40% (~4 000) des SE accepteurs présents dans les *benchmarks*, il génère pour le moment encore un nombre trop important de faux positifs, ce qui impacte fortement les valeurs de F1-Score (figure 67). Nous observons également que les résultats sur les SE donneurs sont globalement meilleurs que les SE accepteurs, à l'instar des résultats de Spliceator, nous laissant entrevoir une forte divergence pour les SE accepteur. De

plus, les deux modèles semblent être plus performants sur l'espèce *D. melanogaster*, car il génère moins de faux positifs et le F1-Score est élevé. A l'inverse, les modèles sont moins performants sur les séquences de *A. thaliana*, car le F1-Score est le plus faible.

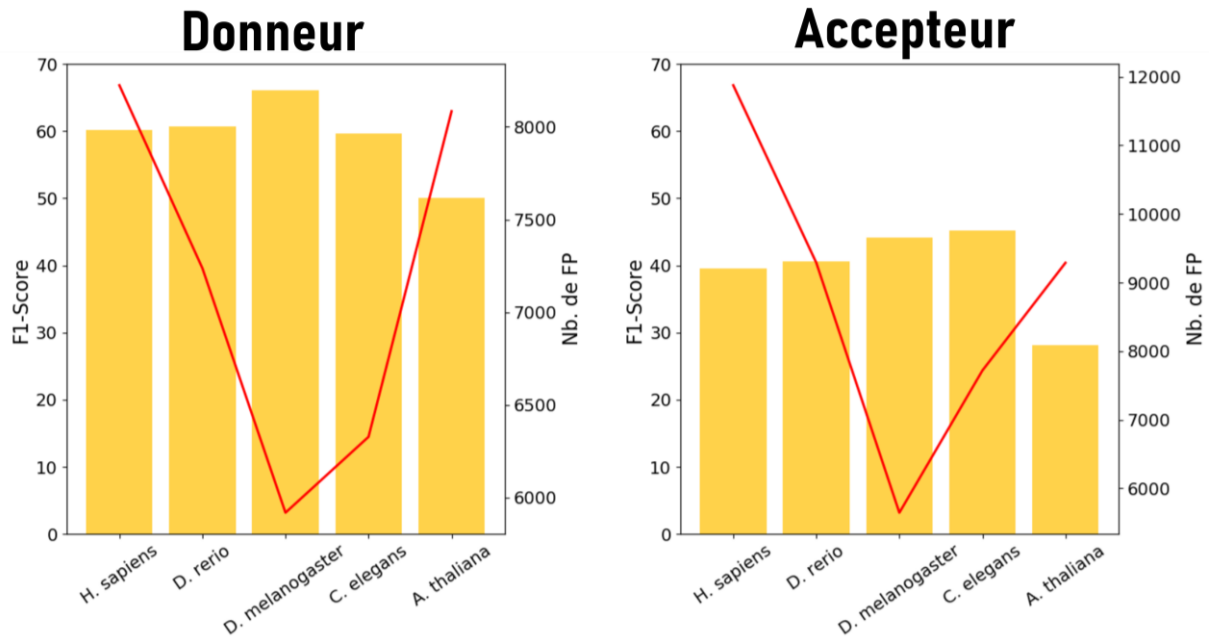


Figure 68 - *Résultats préliminaires de SpliceSLEIA sur les benchmarks H. sapiens, D. rerio, D. melanogaster, C. elegans et A. thaliana. Les SE prédits sont le résultat de la réunion des collections des 3 meilleures regex sélectionnées avec un seuil de 2. On observe un nombre important de faux positifs (courbe rouge), ce qui réduit les valeurs du F1-Score.*

SpliceSLEIA est encore en phase de développement et de nombreuses améliorations sont encore possibles. Les résultats préliminaires, bien qu'assez faibles, sont très encourageants pour la suite de notre travail. En effet, en augmentant la taille de la population nous pourrons explorer un espace de recherche encore plus grand afin d'obtenir des regex encore plus précises. Cependant bien que le nombre de FP soit important, SpliceSLEIA est capable de détecter des SE de manière assez efficace. Nous avons donc décidé de coupler SpliceSLEIA avec Spliceator.

8.6 Hybridation réseaux neuronaux / programmation génétique

La stratégie finale pour améliorer la prédiction des SE dans les gènes codant pour des protéines est de combiner les deux algorithmes : SpliceSLEIA et Spliceator pour obtenir un algorithme hybride, bénéficiant des avantages de chaque approche. Suite aux prédictions indépendantes des SE de chaque programme sur une séquence génomique d'entrée, les résultats sont mis en commun. Une étape de sélection est ensuite appliquée afin d'extraire les SE les plus

probables, générant ainsi les prédictions finales. L'ensemble du processus combinatoire est présenté dans la figure 69. D'autres méthodes sont à envisager tel qu'un système de vote, l'ajout de probabilités ou de statistiques en fonction des scores, etc.

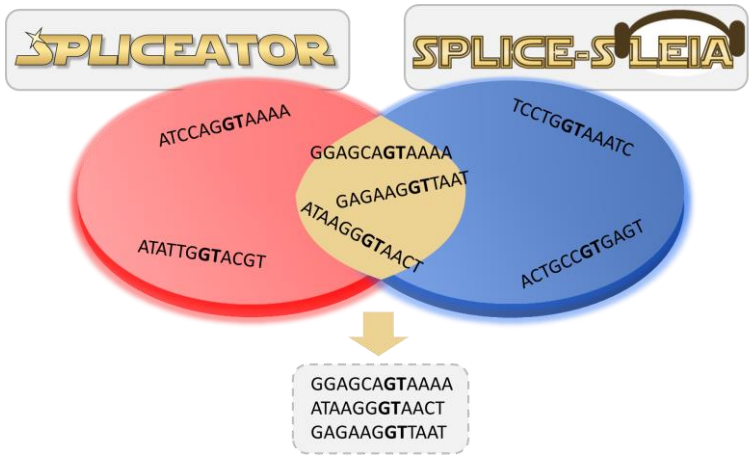


Figure 69 - *Stratégie combinatoire des programmes Spliceator et SpliceSLEIA pour la prédiction des SE dans les gènes codant pour des protéines. Chaque programme effectue des prédictions sur une séquence génomique d'entrée puis les résultats sont mis en commun. Les SE prédit par les deux programmes sont conservés.*

En raison du développement en cours de SpliceSLEIA, cette stratégie hybride est également un prototype. Des résultats préliminaires ont cependant été réalisés sur différents gènes d'organismes différents. Les résultats sont présentés dans le tableau 7. Les prédictions des SE donneurs semblent encore une fois meilleures que celles des accepteurs.

A)

ID Ensembl	Organisme	Nbr exon	Donneur			
			SE prédit (VP/FP)		Combinaison des VP	Combinaison des FP
			Spliceator	SpliceSLEIA		
ENSPTRT00000012383.4	*Pan troglodytes*	6	4/30	4/226	4	2
Aco011873.1	*Ananas comosus*	13	9/1	2/8	2	0
VIT_07s0151g00630	*Vitis vinifera*	5	1/2	1/9	1	2
CAD51656	*Plasmodium falciparum*	6	4/0	3/0	3	0
CAX63992	*Plasmodium falciparum*	10	8/11	1/1	1	0

B)

ID Ensembl	Organisme	Nbr exon	Accepteur			
			SE prédit (VP/FP)		Combinaison des VP	Combinaison des FP
			Spliceator	SpliceSLEIA		
ENSPTRT00000012383.4	*Pan troglodytes*	6	3/28	1/231	1	4
Aco011873.1	*Ananas comosus*	13	4/8	2/21	2	0
VIT_07s0151g00630	*Vitis vinifera*	5	1/2	1/5	1	1
CAD51656	*Plasmodium falciparum*	6	4/0	1/1	1	0
CAX63992	*Plasmodium falciparum*	10	6/1	2/0	1	0

Tableau 7 - *Résultats préliminaires de la combinaison de Spliceator et SpliceSLEIA. Les programmes ont été testé sur différents organismes de clades différentes (métazoaire, plante et protiste) et leurs résultats ont été combinés. Le tableau* ***A)*** *décrit les résultats pour le SE donneur et le tableau* ***B)*** *pour le SE accepteur. Cette combinaison semble très efficace pour réduire le nombre de faux positifs mais possède encore des lacunes pour obtenir des prédictions de vrais positifs efficaces.*

D'après ces résultats, la combinaison des deux programmes permet de réduire drastiquement le nombre de faux positifs, cependant, cela réduit également le nombre de vrais positifs. Une bonne solution serait d'évaluer avec plus de précision les SE prédits afin de leur attribuer un poids ou des probabilités, notamment ceux prédits par Spliceator, qui semble être plus performant pour le moment. Cette stratégie initiale nous a permis d'obtenir des résultats très prometteurs et nous continuerons à améliorer les modèles des programmes afin de développer un prédicteur de SE multi-espèces encore plus performant. En combinant nos deux approches, nous pourrons augmenter le nombre de vrais positifs et diminuer le nombre de faux positifs.

Actuellement, un article fondamental est en cours de rédaction pour présenter la proposition novatrice d'évolution artificielle d'automates à états finis par évolution d'expressions régulières en utilisant la programmation génétique avec SpliceSLEIA comme application. Il sera soumis prochainement à la communauté des chercheurs en évolution artificielle.

DISCUSSION

1- L'annotation des génomes eucaryotes est vulnérable

Aujourd'hui, de nombreux domaines exploitent l'IA, dont la bio-informatique, qui malgré sa jeune existence, est confronté à de nombreux défis (Yandell and Ence, 2012). Parmi ces défis, il y a celui sur lequel j'ai consacré mes trois ans de thèse : l'annotation des génomes eucaryotes. Je me suis principalement focalisé sur les gènes codant pour des protéines, en identifiant les sites d'épissage, frontières entre les exons et les introns.

L'avènement du *Big Data* a généré une quantité importante de données, malheureusement encore trop impactées par les erreurs (Promponas *et al.*, 2015), réduisant leur qualité. Ainsi, pour améliorer l'annotation des génomes, j'ai mis au point une nouvelle stratégie visant à affiner la qualité des données, puis à les exploiter pour entraîner et évaluer des algorithmes d'IA, afin de prédire les SE. L'ensemble de mes travaux est donc basé sur cette triade :

Annotation ⇔ données de qualité ⇔ algorithmes d'IA

Chez les organismes eucaryotes, l'annotation des gènes codant pour des protéines est une tâche compliquée en raison de leur faible présence dans le génome (seulement 2-3% chez l'humain), ainsi que de leur structure interne complexe, clairsemée de segments exoniques et introniques (Salzberg, 2019). La prédiction de cette structure est encore difficile lorsqu'il s'agit d'un nouvel organisme eucaryote (Danchin *et al.*, 2018). De plus, les approches de prédiction *ab initio* prédisent en générale uniquement la CDS la plus probable (Yandell and Ence, 2012). Elles omettent les régions UTR ou les isoformes alternatives. La présence de régions répétées ou de faible complexité alourdit également la balance en défaveur d'une annotation précise. Ces caractéristiques rendent les prédictions de gènes codant pour des protéines difficiles, se traduisant par une production sempiternelle d'erreurs au sein des données générées. En effet, plus il y a de génomes annotés automatiquement, plus les erreurs sont probables. Hormis le fait que ces erreurs soient la cause de données aberrantes, elles peuvent également se propager de manière exponentielle (Gilks *et al.*, 2002) au sein des bases de données, engendrant une contamination discrète de ces dernières. Lorsqu'une nouvelle annotation est réalisée, il est probable qu'elle se base sur des méthodes d'homologie incluant une donnée erronée, perpétuant ainsi la propagation (Promponas *et al.*, 2015). Ce phénomène est appelé "percolation d'erreurs" (Gilks *et al.*, 2002). Malencontreusement, si une erreur est détectée puis corrigée, il n'existe

actuellement aucune méthode capable de retracer sa propagation et de corriger les autres données impactées (Salzberg, 2019).

2- Elaboration de G3PO pour une caractérisation des données erronées

Ce problème m'amène à mettre l'accent sur la qualité des données. Pour réaliser une bonne annotation, il est nécessaire de travailler avec un génome assemblé de qualité. Le logiciel BUSCO (Seppey *et al.*, 2019) a été développé pour évaluer la complétude des génomes assemblés et de nombreuses métriques permettent de caractériser sa qualité, dont une des plus importantes qui est le N50 (contig et scaffold). Le N50 contig de l'assemblage correspond à la longueur du contig le plus court parmi la meilleure moitié (contigs les plus longs) de l'ensemble des contigs. Cela signifie que 50% des contigs ont une taille supérieure à cette valeur. Un N50 élevé est un gage de qualité, car le génome est moins fragmenté, par exemple le génome humain actuelle a un N50 contig égale à 57,9 Mbp et un N50 scaffold de 67,8 Mpb[5]. La fragmentation des génomes entraîne la présence de gaps et donc de nucléotides indéterminés notés 'N', biaisant les prédictions lors du processus d'annotations.

Conscient de la conjoncture de ce phénomène, nous avons élaboré G3PO et G3PO+, afin de travailler avec des données de haute qualité qui ont été nettoyées préalablement. Cette stratégie nous permet de tirer des conclusions robustes sur les recherches réalisées en aval. Cependant, lorsque l'on travaille avec des données issues de bases de données, il est de coutume de faire face à des difficultés et la construction de G3PO n'a pas dérogé à la règle. Un point sensible qui a rendu mes travaux fastidieux est la disponibilité des données, par exemple il n'est pas rare qu'une séquence protéique soit disponible dans Uniprot, mais pas dans Ensembl. De plus, dans certains cas, la séquence génomique est disponible via l'API d'Ensembl, mais pas la carte exonique et *vice versa*. Le deuxième point délicat est la version des données. G3PO+ intègre des données issues de la version 87 d'Ensembl, et la version la plus récente est la 104, il y a donc eu un peu moins d'une vingtaine de nouvelles versions. Cette activité intense complique la recherche inter-bases lorsque les banques divergent ou ne sont pas encore mises à jour, car elles incorporent des données obsolètes (Bakke *et al.*, 2009). J'ai aussi rencontré des problèmes de cohérence au sein des données, comme un décalage de la séquence génomique de quelques nucléotides, ce qui avait pour conséquence de ne plus correspondre avec la carte

5 ncbi.nlm.nih.gov/assembly/GCF_000001405.39/

exonique. De manière intéressante, ce décalage n'était présent que sur des séquences extraites via l'API et non pas sur le site d'Ensembl. Ainsi, lors de la manipulation de données issues de bases de données publiques, il est important de vérifier manuellement certaines étapes. Ces difficultés sont les causes directes de la réduction du nombre de séquences présentes dans G3PO. Initialement, les 45 MSA comprenaient plus de 8 500 séquences, mais la version finale de G3PO+ n'en comprend que 2 741 (~32%).

Malgré cela, nous avons caractérisé un nombre important d'erreurs et les avons classées en neuf catégories. Ces erreurs ont un impact sur la prédiction des gènes générés par les programmes, en réduisant leur performance. Notre étude sur l'identification des causes de ces erreurs chez une dizaine de primates a permis une meilleure compréhension. A partir des connaissances extraites de cette étude, ainsi que de l'étude comparative de G3PO, nous avons identifié des caractéristiques et des points importants à prendre en compte pour améliorer l'annotation des gènes codant pour des protéines. La présence de nucléotides indéterminés est un facteur important, ainsi que la présence de petits exons/introns (<30 nucléotides) et de SE non-canoniques qui peuvent aussi perturber les prédictions. En outre, certains organismes sont plus enclins à de mauvaises prédictions comme les protistes, les cnidaires ou les fungi.

Ainsi, la force de G3PO+ réside dans la qualité de ces données, mais également dans l'hétérogénéité des organismes eucaryotes (issues de plusieurs clades) qu'il intègre.

3- La technologie de pointe au secours de l'annotation

Afin de réduire la quantité d'erreurs dans les séquences biologiques, la communauté scientifique continue de chercher des solutions pour améliorer les performances des programmes de prédiction des gènes codant pour des protéines. Initialement, les nombreux outils développés étaient basés sur des modèles probabilistes comme les modèles de Markov ou des algorithmes de *machine learning* (SVM, arbre de décision, etc.). Cependant, ces méthodes sont limitées par la taille du contexte qu'elles peuvent prendre en compte, elles sont dépendantes des connaissances des données d'entrée (*i.e.* elles n'identifient pas de nouvelles caractéristiques) et la construction et la sélection des caractéristiques pertinentes pour l'entraînement du modèle est complexe et chronophage (Degroeve *et al.*, 2002). L'entraînement de ces modèles est également un facteur limitant, car il faut prendre en compte les différentes caractéristiques génétiques de l'espèce et bien souvent, seuls les organismes modèles sont bien étudiés et disposent d'un modèle (Yandell and Ence, 2012). Les organismes plus exotiques et

phylogénétiquement éloignés des organismes modèles sont donc voués à être moins (et mal) annotés.

Ainsi, de nombreux efforts ont été réalisés sur le plan expérimental, notamment grâce aux technologies à haut débit comme le RNA-seq (Ozsolak and Milos, 2011). Cette technique présage une amélioration significative de l'annotation du génome, notamment au niveau de la prédiction des SE (Au *et al.*, 2010), comme avec l'utilisation de l'outil MapSplice (Wang *et al.*, 2010). Mais le RNA-seq n'est pas la panacée, car cette technique est incapable de bien distinguer les ARN codant pour des protéines des autres ARN, le rapport signal sur bruit est donc élevé. Une des particularités de l'ARN est son instabilité et sa tendance à une dégradation rapide, par conséquent certains transcrits peuvent déjà être dégradés (Kovaka *et al.*, 2019). De plus, les transcrits faiblement exprimés ne sont pas détectés par cette technique (Westoby *et al.*, 2020) ainsi que les transcrits qui sont tissu-spécifiques et les petits exons (Sahlin and Mäkinen, 2021). Dans certains cas, obtenir des échantillons peut être trop invasif, voire impossible. Obtenir le transcriptome à saturation, ou "isoformome" (les isoformes issues de tous les stades de la vie, tous les tissus, toutes les conditions, etc.) est pour le moment utopique.

Par conséquent, la meilleure stratégie est probablement la combinaison de ces approches avec des méthodes *ab initio*. Cependant comme nous venons de le voir, les méthodes *ab initio* actuelles sont de plus en plus limitées. C'est pourquoi depuis 2010, avec l'arrivée du *deep learning*, de nouvelles opportunités sont apparues et aujourd'hui une pléthore de domaines exploite cette technologie, dont la bio-informatique (Angermueller *et al.*, 2016; Min *et al.*, 2017; Tang *et al.*, 2019; Li *et al.*, 2020; Cao *et al.*, 2020), mais chaque révolution amène ses inquiétudes et ses défis.

4- Les limites et défis du *deep learning*

Une des principales critiques que l'on peut imputer au *deep learning* est son manque de transparence. En effet, le terme "boîte noire" est une expression courante lorsque l'on travaille avec des réseaux de neurones (RN) profonds. Cela signifie que le modèle est devenu tellement complexe, qu'il est difficilement compréhensible et donc interprétable. L'explicabilité des RN est un enjeu crucial pour comprendre les raisons des choix effectués par le modèle. Dans le cadre médical, il est important de comprendre et d'expliquer les choix effectués par le modèle (de Bruijne, 2016), au risque d'effectuer une erreur dommageable. Certains algorithmes de *machine learning* sont plus explicables que d'autres (figure 70), tels que les algorithmes à base

d'arbres ou encore les systèmes experts. Un autre problème proche de l'explicabilité est celui de la fiabilité (Jain *et al.*, 2020). Bien que de nombreux modèles obtiennent de très hautes performances, il est nécessaire d'estimer leur fiabilité afin de pouvoir estimer leur degré de confiance. Certains travaux ont commencé à émerger afin d'estimer cette fiabilité (Amini *et al.*, 2020), mais nous ne sommes qu'au début de cette nouvelle ère.

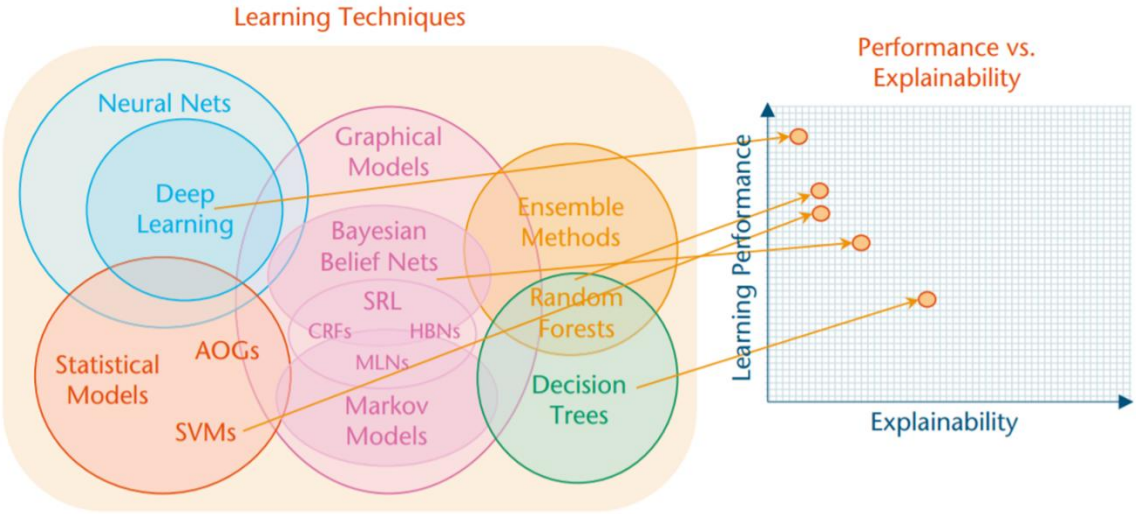


Figure 70 - *Représentation de l'explicabilité en fonction de la performance de différents algorithmes de machine learning. Les algorithmes de deep learning obtiennent les meilleures performances, mais sont très peu explicables, tandis que les algorithmes à base d'arbres sont plus facilement explicables mais, moins performants (source : (Gunning and Aha, 2019)).*

Hormis les défis éthiques engendrés par l'émergence du *deep learning*, les chercheurs font aussi face à des problèmes techniques. En effet, le compromis entre le biais et la variance est compliqué à maîtriser. Un biais trop important (trop de mauvaises prédictions) suite à un sous-apprentissage, ou une variance trop élevée (modèle trop sensible) dû à un sur-apprentissage peut impacter fortement les performances du modèle. Il est donc crucial de contrôler ce phénomène. Ensuite, il n'existe pour le moment aucune théorie ou stratégie pour mettre en place une architecture optimale d'un RN pour une problématique définie. Du fait de la combinaison infinie des composants architecturaux, ainsi que des hyperparamètres (nombre d'epochs, pas d'apprentissage, taille des noyaux, pourcentage de *dropout* etc.) il est difficile d'optimiser un RN et sa performance est fortement dépendante des choix opérés. Cependant, certaines approches originales exploitant les algorithmes évolutionnaires ont été utilisées afin de rechercher une architecture de RN et ses paramètres optimaux (Angeline *et al.*, 1994; Leung *et al.*, 2003; Suganuma *et al.*, 2017). Un autre défi est lié à la problématique posée. Dans le cadre d'une régression linéaire, la descente de gradient s'effectue aisément sur une fonction convexe, mais dans le cadre des RN les fonctions sont bien plus complexes. Ainsi, en fonction

de l'initialisation aléatoire des poids, il n'est pas garanti d'atteindre l'optimum global, mais plutôt un optimum local. La disponibilité des ressources tant sous la forme de données que de la puissance de calcul est aussi un problème important. Les RN sont d'énormes consommateurs de ressources et il est nécessaire d'avoir du matériel optimal afin que le processus d'entraînement des RN ne soit pas trop long. Heureusement, au milieu des années 2000, le problème commence à se résoudre avec à l'arrivée des GPU et du calcul parallélisable. Aujourd'hui, grâce à ces GPU de pointe on peut facilement entraîner un RN avec des millions de paramètres en quelques minutes. En somme, s'il y a bien un élément à prendre en compte en *deep learning* et plus généralement en IA ce sont les données. Ces dernières sont les fondements de chaque modèle. Sans un jeu de données de taille conséquente, de qualité et en lien avec la problématique, on aura beau avoir optimisé l'architecture et les hyperparamètres du RN, les performances seront médiocres. Aujourd'hui d'énormes quantités de données sont disponibles notamment grâce aux technologies à haut débit. Cependant, il ne s'agit pas de posséder uniquement des données, il est également important que ces données soient étiquetées par des experts afin qu'elles puissent être exploitables. De plus, même si les données sont disponibles il peut y avoir un problème de déséquilibre entre les classes, car la biologie comprend un certain nombre de problèmes disproportionnés. Par exemple, pour l'identification de variant ou de SE, il y a beaucoup plus de données de cas négatifs que de positifs. Ainsi, un défaut au niveau des données peut engendrer un problème de sur/sous-apprentissage atténuant les performances des modèles. Malgré cela, les algorithmes de *deep learning* se sont révélés très performants en biologie, comme c'est le cas par exemple d'AlphaFold lors du CASP14 (Jumper *et al.*, 2021).

5- Spliceator : modèle universel pour les sites d'épissage

Aujourd'hui, les algorithmes de *deep learning* sont de plus en plus utilisés notamment pour la détection d'éléments fonctionnels du génome tel que les promoteurs, les sites de fixations à l'ADN ou les SE, car ils sont capables de détecter des caractéristiques dans les données brutes. Ainsi, de nombreux outils ont été développés pour identifier les SE, malheureusement leurs modèles sont souvent limités aux organismes modèles : SpliceAI (Jaganathan *et al.*, 2019), SpliceFinder (Wang *et al.*, 2019) et DSSP : *H. sapiens* (Naito, 2018), SpliceRover (Zuallaert *et al.*, 2018) : *H. sapiens* et *A. thaliana*, Splice2Deep (Albaradei *et al.*, 2020) : *H. sapiens*, *A. thaliana, O. sativa japonica, D. melanogaster* et *C. elegans*. Dans ce contexte, en tirant partie de l'hétérogénéité phylogénétique des données de G3PO+, nous avons

développé et entraîné Spliceator, un programme de prédiction multi-espèces de SE. Cette nouvelle dimension nous a permis d'élargir notre compréhension des caractéristiques universelles de l'épissage et de confirmer que certains motifs sont hautement conservés (Frey and Pucker, 2020), mais également des spécificités propres à chaque espèce. Cela permet de caractériser les organismes les plus divergents et d'identifier des points faibles des prédictions pour certains clades, puis de les exploiter pour améliorer les prédictions. Cette hétérogénéité rend l'annotation des organismes exotiques également plus accessibles, comme les protistes ou les fungi, qui souffrent d'un manque de modèles et d'un taux d'erreur important.

Spliceator exploite également les données de haute qualité de notre jeu de données G3PO+ ce qui nous a permis de confirmer le concept du "*garbage-in, garbage-out*" et de démontrer que la qualité des données avait un impact sur l'entraînement du modèle et donc sur la performance de ce dernier. Malheureusement, les données de qualité sont rares, ce qui se manifeste dans notre étude par un jeu d'entraînement restreint à ~10 000 séquences. A noter qu'une quantité limitée de données peut empêcher le modèle de les généraliser correctement. Nous avons observé ce phénomène lors de la mise en place de Spliceator. Différentes architectures ont été construites et lorsqu'elles étaient trop simples ou trop profondes, le modèle réalisait respectivement du sous et sur-apprentissage. Ainsi, quand de nouvelles données seront disponibles, il sera important de réadapter l'architecture du modèle et de le réentraîner, pour éviter ces problèmes d'apprentissage. L'ajout de nouvelles données dans le jeu d'entraînement permettra notamment d'augmenter les performances du modèle. En ce sens, deux stratégies s'offrent à nous :

i) le *batch learning* est la stratégie adoptée actuellement. Cela consiste à traiter les données, puis entraîner le modèle. A chaque nouvelle donnée, le modèle doit être réentraîné, ce qui à long terme peut être chronophage, mais permet de garder le contrôle sur les données.

ii) le *cloud learning* permet aux modèles de s'entraîner incrémentalement. Cette stratégie est intéressante lorsque les données arrivent en flux continu. Cependant, il faut maîtriser leur ajout car si elles sont erronées, elles risquent de diminuer les performances du modèle.

Cette dernière méthode d'apprentissage est envisageable pour Spliceator si les nouvelles données ajoutées ont été analysées et validées selon le même protocole d'identification des erreurs décrit dans la section 6 du Matériels et méthodes.

Bien que Spliceator dispose de peu de données, il a obtenu de très bons résultats sur les *benchmarks* des différents organismes et a notamment surpassé les programmes les plus performants actuellement. Malheureusement, Spliceator souffre de deux principaux problèmes.

Le premier, plutôt mineur, est la redondance des prédictions. En effet, dans certains cas, Spliceator prédit plusieurs positions très proches pour un SE. Par exemple, si le SE est en position 200, Spliceator peut le prédire en position 199, 200 et 201, ce qui génère des FP, bien qu'il soit très proche du véritable SE. On peut cependant tirer un avantage à cette sur-prédiction, lorsque l'on observe un certain nombre de redondances, on peut supposer que Spliceator est quasiment sûr d'identifier un SE dans cette région. Le deuxième problème est la prédiction encore trop importante de FP. Cela pourra être corrigé avec l'ajout de nouvelles données, mais pour le moment, nous avons mis au point une autre stratégie : la combinaison des résultats de Spliceator et de SpliceSLEIA.

6- Les algorithmes évolutionnaires, révolutionnaires

Le programme SpliceSLEIA a été conçu avec un algorithme de programmation génétique qui est très peu utilisé pour des problèmes d'identification d'éléments génomiques. Notre stratégie, visant à faire évoluer des automates à états finis, sous forme d'expression régulière, nous a permis de réduire drastiquement les FP (>90%) lorsque nous combinons les résultats de SpliceSLEIA avec Spliceator. Comme il s'agit pour le moment d'un prototype, cela limite également la prédiction des VP, c'est pourquoi nous envisageons un système de poids (ou de probabilités) qui accentuerait les bonnes prédictions de Spliceator, en plus de l'amélioration des modèles des deux programmes. Nous envisageons également d'améliorer la recherche paramétrique ainsi que l'augmentation de l'espace de recherche. En effet, un petit espace de recherche induit une convergence prématurée de l'algorithme. Il est donc important d'augmenter la taille de la population et surtout d'y inclure de la diversité, grâce notamment aux opérateurs évolutionnaires. Parallèlement, la convergence tardive est un autre goulet d'étranglement, elle peut être induite par une mauvaise initialisation des individus, une taille de population trop faible ou un problème de conceptualisation en lien avec les données. Lors de la mise en place de SpliceSLEIA, nous avons été restreints au niveau de la taille de la population, réduisant ainsi l'espace de recherche. Cette restriction a provoqué des problèmes de non-convergence sur les modèles donneurs, lorsque nous les avons évalués sur le jeu de test avec des séquences d'une taille de 20 nucléotides. En effet, nous étions limités en termes de puissance de calcul car le processus évolutif peut être très long et très coûteux. Cependant, la parallélisation (Alba and Tomassini, 2002; Tsutsui and Collet, 2013; Maitre *et al.*, 2017) de ces algorithmes sur des infrastructures GPU permettra d'améliorer ces performances ainsi que la

vitesse de calcul et de nombreux travaux ont déjà été effectués (Harding and Banzhaf, 2007; Langdon, 2011). La plateforme EASEA permet des calculs sur GPU, cependant notre projet n'a pas encore été adapté pour une exécution sur ce type d'architecture, mais des travaux sont en cours de développement pour l'implémenter.

Bien que l'avantage des AE soit leur nature stochastique, car ils peuvent échapper aux minima locaux (que les algorithmes déterministes ne peuvent pas réaliser), elle ne garantit pas de trouver une bonne solution au problème initial si l'espace de recherche est trop petit. De plus, il est fortement probable qu'avec les mêmes conditions, on ne retrouve jamais les mêmes résultats entre deux expériences, ce qui pose des problèmes de reproductibilité, qui sont cependant gérables en calculant le *Computational Effort* défini par John Koza (Koza, 1994). Les résultats de la PG sont des fonctions symboliques (ici des expressions régulières) elles sont donc bien plus explicables que le résultat des boîtes noires du *deep learning* ce qui en fait un atout majeur.

Actuellement, la communauté qui s'est construite autour des AE est petite mais de nombreux chercheurs emblématiques ont enrichi, par leurs travaux et leurs expériences, ce domaine de recherche. Ce dernier a malheureusement été rapidement éclipsé par les résultats médiatiques du *deep learning*, alors que ce sont des approches différentes, apportant des résultats qualitativement différents. Cependant, le fait que leur mode de fonctionnement soit différent signifie qu'ils ne sont pas exclusifs : on peut les combiner pour tirer parti des avantages de chacune de ces méthodes. On voit donc l'importance de continuer à exploiter plusieurs types d'algorithmes et de ne pas se concentrer uniquement sur ceux qui subissent un effet de mode. Ce constat a été effectué par Marvin Minsky dans les années 50, quand il travailla sur les réseaux de neurones et qu'il abandonna ces recherches pour d'autres approches plus populaires. L'avenir des AE est dans une explicabilité complémentaire au *deep learning*, qui fournit des résultats intéressants mais non symboliques. L'augmentation de la puissance de calcul procurée par la parallélisation sur GPU va probablement favoriser son essor (Sloss and Gustafson, 2019). De plus, de nombreuses librairies se développent dans plusieurs langages de programmation populaires (java (Kronfeld *et al.*, 2010), python (Fortin *et al.*, 2012), C++ (eodev.sourceforge.net)), facilitant l'utilisation des AE (Corne and Lones, 2018), ces avancées annoncent un bel avenir pour les AE, notamment en bio-informatique.

SpliceSLEIA a donc été conçue dans cet élan, afin de tirer parti des avantages des AE, comme leur nature stochastique et leur faculté à imaginer des solutions originales et leur capacité unique à fournir des solutions symboliques. Nous sommes persuadés qu'en exploitant les mécanismes évolutionnaires, nous trouverons une bonne solution qui parviendra à

augmenter de manière significative les performances des prédictions. SpliceSLEIA est ainsi capable de prédire les SE sans connaissances préalables des mécanismes biologiques, mais en exploitant la puissance des mécanismes évolutifs comme la mutation et le croisement. Par conséquent, les seules limites et biais de notre approche sont les ressources : soit par les données qu'elle exploite, soit par la puissance de calcul et le temps nécessaire.

CONCLUSIONS ET PERSPECTIVES

La bio-informatique a subi un bouleversement important ces vingt dernières années avec le développement de nouvelles technologies à haut débit, l'avènement du *Big data* et l'émergence du *deep learning* qui ont entièrement redessiné son paysage. Ces révolutions ont ainsi soulevé de nouveaux défis en bio-informatique dont l'annotation des génomes eucaryotes. Mes travaux de thèse se sont inscrits dans ce contexte, en utilisant des algorithmes d'IA pour développer une nouvelle stratégie d'annotation des génomes. Cette stratégie repose sur quatre niveaux. Les premier et deuxième niveaux sont consacrés à l'étude des données en identifiant et caractérisant les erreurs présentes dans ces dernières, puis en construisant un jeu de données de haute qualité. Le troisième niveau est l'exploitation de ce jeu de données pour entraîner des algorithmes d'IA à la reconnaissance de SE et enfin le quatrième niveau est la mise en place d'une méthode combinatoire pour la prédiction de SE.

L'analyse comparative que nous avons effectuée a été déterminante pour la suite du projet. Elle a pointé les failles d'un système gangrené par une prolifération d'erreurs générées par des programmes de prédiction, puis diffusé par les bases de données publiques. Il nous était inconcevable de travailler avec des données erronées et nous avons donc développé G3PO+. La force de G3PO+ réside ainsi dans la qualité de ses données, mais également dans l'hétérogénéité des 147 organismes eucaryotes (originaires de plusieurs clades : des primates aux protistes) qu'il intègre.

G3PO+ a ensuite pu être exploité par des algorithmes d'IA, dont leurs besoins primaires sont la consommation massive de données pour pouvoir s'entraîner. Basé sur un réseau de neurones convolutif, Spliceator a été conçu pour identifier les SE donneur et accepteur, en tirant parti des informations qu'il a apprises des séquences de haute qualité de G3PO+. Les performances obtenues par Spliceator sur les *benchmarks* de référence de différents organismes ont surpassé celles des programmes les plus performants actuellement, faisant de lui un des meilleurs prédicteurs de SE universel.

En parallèle, nous avons développé SpliceSLEIA, un second programme de prédiction de SE basé sur un algorithme évolutionnaire novateur, car faisant évoluer des automates à états finis par programmation génétique. En exploitant la richesse et l'inventivité du processus évolutif, nous avons mis au point des modèles capables d'identifier des SE. *In fine*, notre stratégie est de proposer un algorithme hybride combinant ces deux techniques en extrayant l'intersection des résultats. Cette stratégie, bien qu'actuellement au stade de prototype, a dévoilé

des résultats préliminaires très prometteurs et de nombreuses améliorations sont envisageables pour augmenter les performances de nos programmes.

Tout d'abord, dans l'optique d'une extension de G3PO+, nous allons développer un programme capable de localiser automatiquement et avec une précision élevée les erreurs de prédiction dans les séquences, lorsque l'ensemble des règles d'identification de ces erreurs seront établies. Il serait également intéressant de développer un site web avec l'ensemble des données accessibles facilement, ainsi que les statistiques descriptives de G3PO+, afin que cette ressource devienne un "*gold standard*" pour les études multi-espèces.

A moyen terme, les perspectives d'amélioration de Spliceator et SpliceSLEIA sont multiples. Une des plus importantes est la faculté à identifier toutes les isoformes biologiquement possibles afin d'obtenir *l'isoformome* (*i.e.* l'ensemble de toutes les isoformes d'un organisme donné, à tous les âges et pour tous les tissus). Cet aspect sera une plus-value par rapport au RNA-seq, qui rencontre encore des difficultés à identifier les isoformes minoritaires. D'autre part, SpliceSLEIA est un prototype, ce qui signifie qu'il reste de nombreuses améliorations à développer comme la recherche paramétrique ou la fonction de *fitness*, qui peut être perfectionnée soit en utilisant des variantes de la distance de Levenshtein comme la distance de Jaro-Winkler, soit en changeant totalement de stratégie.

Finalement, les pipelines d'annotation de demain intégreront probablement de plus en plus d'outils indépendants basés sur des algorithmes plus performants comme ceux du *deep learning* ou les AE, combinés avec des approches expérimentales comme le RNA-seq spécifique à une cellule et capable de spatialiser l'information cellulaire. Ces outils seront spécifiques à la prédiction d'un seul élément du génome comme les SE, et les preuves rassemblées seront combinées afin de construire la structure du gène la plus probable en fonction des éléments récoltés par chaque outil. Cette stratégie combinatoire, déjà décrite par Bakke *et al.,* (Bakke *et al.,* 2009), semble la plus efficace pour réduire les erreurs d'annotation.

L'objectif final serait d'étendre l'approche de modélisation de Spliceator et SpliceSLEIA vers une méthode complète de prédiction de structure de gène spécifiquement conçue pour optimiser le processus d'annotation du génome et réduire le niveau d'erreurs. Pour y parvenir, nous envisageons de développer un nouveau pipeline intégrant la combinaison des deux programmes et une analyse comparative des séquences, selon cinq étapes importantes :

- Prédire les modèles initiaux de structure des gènes en utilisant des méthodes traditionnelles basées sur l'homologie.
- Construire un alignement multiple des protéines cibles avec les orthologues correspondants.

- Identifier les sites d'épissage potentiels avec Spliceator et SpliceSLEIA pour les gènes présentant des erreurs potentielles.
- Prédire les isoformes de protéines à l'aide d'algorithmes évolutionnaires.
- Construire un programme de prédiction de la séquence de Kozak (codon AUG), afin de pouvoir identifier une CDS en exploitant les séquences de G3PO+.

Ce pipeline, mais aussi l'ensemble de cette thèse, s'intègre dans une volonté d'amélioration de l'annotation complète des gènes, car il est devenu capital que les données ne soient pas systématiquement erronées. En effet, l'utilisation de données erronées peut amener les chercheurs à tirer des conclusions incorrectes, ce qui peut être désastreux pour la science. Avec une meilleure prédiction de la structure des gènes, mais également de leur fonction, nous améliorerons notre compréhension des organismes et de leurs mécanismes biologiques sous-jacents. Ainsi, il sera plus aisé de comprendre le comportement cellulaire d'une espèce, d'étudier les interactions et les relations (interactome), et à plus grande échelle, les systèmes complexes. De plus, une bonne annotation permettra également une meilleure compréhension des mécanismes pathologiques humains, facilitant la mise en place d'une thérapie adéquate, l'identification des relations génotype-phénotype, l'étude de l'histoire évolutive des organismes ou encore la préservation de la biodiversité.

RÉFÉRENCES

Abadi,M. *et al.* (2016) TensorFlow: A system for large-scale machine learning. *arXiv:1605.08695 [cs]*.
Abelson,J. *et al.* (1998) tRNA Splicing. *Journal of Biological Chemistry*, **273**, 12685–12688.
Abraham,A. (2005) Artificial Neural Networks. In, *Handbook of Measuring System Design*. American Cancer Society.
Adams,M.D. *et al.* (1991) Complementary DNA sequencing: expressed sequence tags and human genome project. *Science*, **252**, 1651–1656.
Aken,B.L. *et al.* (2016) The Ensembl gene annotation system. *Database (Oxford)*, **2016**.
Akhtar,N. and Ragavendran,U. (2020) Interpretation of intelligence in CNN-pooling processes: a methodological survey. *Neural Computing and Applications*, **32**.
Alba,E. and Tomassini,M. (2002) Parallelism and evolutionary algorithms. *Evolutionary Computation, IEEE Transactions on*, **6**, 443–462.
Albaradei,S. *et al.* (2020) Splice2Deep: An ensemble of deep convolutional neural networks for improved splice site prediction in genomic DNA. *Gene: X*, **5**, 100035.
Albawi,S. *et al.* (2017) Understanding of a convolutional neural network. In, *2017 International Conference on Engineering and Technology (ICET)*., pp. 1–6.
Albertin,C.B. *et al.* (2012) Cephalopod genomics: A plan of strategies and organization. *Stand. Genomic Sci.*, **7**, 175–188.
Allen,J.E. and Salzberg,S.L. (2005) JIGSAW: integration of multiple sources of evidence for gene prediction. *Bioinformatics*, **21**, 3596–3603.
AlQuraishi,M. (2019) End-to-End Differentiable Learning of Protein Structure. *Cell Systems*, **8**, 292-301.e3.
Altaf-Ul-Amin,M. *et al.* (2014) Systems Biology in the Context of Big Data and Networks. *BioMed Research International*, **2014**, e428570.
Altschul,S.F. *et al.* (1990) Basic local alignment search tool. *Journal of Molecular Biology*, **215**, 403–410.
Amin,J. *et al.* (2018) Big data analysis for brain tumor detection: Deep convolutional neural networks. *Future Generation Computer Systems*, **87**, 290–297.
Amini,A. *et al.* (2020) Deep Evidential Regression. *arXiv:1910.02600 [cs, stat]*.
Angeline,P.J. *et al.* (1994) An evolutionary algorithm that constructs recurrent neural networks. *IEEE Transactions on Neural Networks*, **5**, 54–65.
Angermueller,C. *et al.* (2016) Deep learning for computational biology. *Molecular Systems Biology*, **12**, 878.
Au,K.F. *et al.* (2010) Detection of splice junctions from paired-end RNA-seq data by SpliceMap. *Nucleic Acids Res*, **38**, 4570–4578.
Auboeuf,D. *et al.* (2012) Alternative splicing and cancer. *J Nucleic Acids*, **2012**, 363809.
Avery,O.T. *et al.* (1944) Studies on the chemical nature of the substance inducing transformation of pneumococcal types. *J Exp Med*, **79**, 137–158.
Azaria,Y. and Sipper,M. (2005) GP-Gammon: Genetically Programming Backgammon Players. *Genet Program Evolvable Mach*, **6**, 283–300.
Back,T. and Schwefel,H.-P. (1996) Evolutionary computation: an overview. In, *Proceedings of IEEE International Conference on Evolutionary Computation*., pp. 20–29.
Bahrammirzaee,A. (2010) A comparative survey of artificial intelligence applications in finance: artificial neural networks, expert system and hybrid intelligent systems. *Neural Comput & Applic*, **19**, 1165–1195.
Bakke,P. *et al.* (2009) Evaluation of Three Automated Genome Annotations for Halorhabdus utahensis. *PLOS ONE*, **4**, e6291.
Bakker,R. and Jansen,F. (2018) Evolving Regular Expression Features for Text Classification with Genetic Programming.
Bang,M.-L. *et al.* (2001) The Complete Gene Sequence of Titin, Expression of an Unusual ≈700-kDa Titin Isoform, and Its Interaction With Obscurin Identify a Novel Z-Line to I-Band Linking System. *Circulation Research*, **89**, 1065–1072.
Banzhaf,W. (2013) Evolutionary Computation and Genetic Programming.
Banzhaf,W. *et al.* (1998) Genetic Programming: An Introduction on the Automatic Evolution of computer programs and its Applications.

Baralle,M. and Baralle,F.E. (2018) The splicing code. *Biosystems*, **164**, 39–48.
Barash,Y. *et al.* (2010) Deciphering the splicing code. *Nature*, **465**, 53–59.
Bari,A.T.M. *et al.* (2013) Effective DNA encoding for splice site prediction using SVM. *Match (Mulheim an der Ruhr, Germany)*, **71**, 241–258.
Barricelli,N.A. (1957) Symbiogenetic Evolution Processes Realized by Artificial Methods.
Bartoli,A. *et al.* (2012) Automatic generation of regular expressions from examples with genetic programming. In, *Proceedings of the 14th annual conference companion on Genetic and evolutionary computation*, GECCO '12. Association for Computing Machinery, New York, NY, USA, pp. 1477–1478.
Baştanlar,Y. and Ozuysal,M. (2014) Introduction to machine learning. *Methods Mol Biol*, **1107**, 105–128.
Behera,N. *et al.* (2017) Higher accuracy protein multiple sequence alignments by genetic algorithm. *Procedia Computer Science*, **108**, 1135–1144.
Ben Khalaf,N. *et al.* (2019) The mouse intron-nested gene, Israa, is expressed in the lymphoid organs and involved in T-cell activation and signaling. *Molecular Immunology*, **111**, 209–219.
Benson,D.A. *et al.* (2009) GenBank. *Nucleic Acids Res*, **37**, D26–D31.
Benson,G. (1999) Tandem repeats finder: a program to analyze DNA sequences. *Nucleic Acids Res*, **27**, 573–580.
Bentley,D.L. (2014) Coupling mRNA processing with transcription in time and space. *Nat Rev Genet*, **15**, 163–175.
Berger,B. and Leighton,T. (1998) Protein Folding in the Hydrophobic-Hydrophilic (HP) Model is NP-Complete. *Journal of Computational Biology*, **5**, 27–40.
Berget,S.M. *et al.* (1977) Spliced segments at the 5′ terminus of adenovirus 2 late mRNA. *PNAS*, **74**, 3171–3175.
Bernstein,F.C. *et al.* (1977) The protein data bank: A computer-based archival file for macromolecular structures. *Journal of Molecular Biology*, **112**, 535–542.
Bianchi,L. *et al.* (2009) A survey on metaheuristics for stochastic combinatorial optimization. *Nat Comput*, **8**, 239–287.
Birney,E. (2004) GeneWise and Genomewise. *Genome Research*, **14**, 988–995.
Björk,S. (2013) On The Foundations of Digital Games.
Blencowe,B.J. (2006) Alternative Splicing: New Insights from Global Analyses. *Cell*, **126**, 37–47.
Blodgett,J.H. and Schultz,C.K. (1969) Herman hollerith: data processing pioneer. *American Documentation*, **20**, 221–226.
Blum,C. and Roli,A. (2003) Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Comput. Surv.*, **35**, 268–308.
de Boer,M. *et al.* (2019) Activation of cryptic splice sites in three patients with chronic granulomatous disease. *Mol Genet Genomic Med*, **7**, e854.
Boole,G. (1847) The Mathematical Analysis of Logic Philosophical Library.
Boštjančić,L.L. *et al.* (2021) Comparative transcriptome analysis of noble crayfish and marbled crayfish immune response to Aphanomyces astaci challenges. *bioRxiv*, 2021.05.25.445163.
Boumedine,N. and Bouroubi,S. (2019) A new hybrid genetic algorithm for protein structure prediction on the 2D triangular lattice. *arXiv:1907.04190 [cs, math]*.
Boussaïd,I. *et al.* (2013) A survey on optimization metaheuristics. *Information Sciences*, **237**, 82–117.
Brameier,M.F. and Banzhaf,W. (2007) Linear Genetic Programming Springer US.
Braun,G. *et al.* (2011) How Rosalind Franklin Discovered the Helical Structure of DNA: Experiments in Diffraction. *The Physics Teacher*, **49**, 140–143.
Breathnach,R. and Chambon,P. (1981) Organization and Expression of Eucaryotic Split Genes Coding for Proteins. *Annu. Rev. Biochem.*, **50**, 349–383.
Bretschneider,H. *et al.* (2018) COSSMO: predicting competitive alternative splice site selection using deep learning. *Bioinformatics*, **34**, i429–i437.
Brindle,A. (1980) Genetic algorithms for function optimization. *ERA*.
Brown,G.G. *et al.* (2014) Group II intron splicing factors in plant mitochondria. *Front. Plant Sci.*, **0**.
de Bruijne,M. (2016) Machine learning approaches in medical image analysis: From detection to diagnosis. *Medical Image Analysis*, **33**, 94–97.

Brůna,T. *et al.* (2020) BRAKER2: Automatic Eukaryotic Genome Annotation with GeneMark-EP+ and AUGUSTUS Supported by a Protein Database. *bioRxiv*, 2020.08.10.245134.
Brunak,S. *et al.* (1991) Prediction of human mRNA donor and acceptor sites from the DNA sequence. *Journal of Molecular Biology*, **220**, 49–65.
Buchanan,B.G. and S. (1984) Rule- Based Expert Systems : The MYCIN Experiments of the Stanford Heuristic Programming Project. In, *xix, 748 p. Reading, Mass.: Addison-Wesley Pub. Co., 1984. includes bibliography: p. 717-738 and subject index*. CUMINCAD.
Burge,C. and Karlin,S. (1997) Prediction of complete gene structures in human genomic DNA. *Journal of Molecular Biology*, **268**, 78–94.
Burge,Christopher B. *et al.* (1999) Splicing of Precursors to mRNAs by the Spliceosomes. *2354 Nucleic Acids Research*.
Burks,A.W. (1947) Electronic Computing Circuits of the ENIAC. *Proceedings of the IRE*, **35**, 756–767.
Burset,M. *et al.* (2000) Analysis of canonical and non-canonical splice sites in mammalian genomes. *Nucleic Acids Res*, **28**, 4364–4375.
Burton,C. (1998) The Manchester baby reborn. *IEE Review*, **44**, 113–117.
Butler,J.E.F. and Kadonaga,J.T. (2002) The RNA polymerase II core promoter: a key component in the regulation of gene expression. *Genes Dev.*, **16**, 2583–2592.
Cáceres,J.F. and Kornblihtt,A.R. (2002) Alternative splicing: multiple control mechanisms and involvement in human disease. *Trends in Genetics*, **18**, 186–193.
Cai,D. *et al.* (2000) Modeling splice sites with Bayes networks. *Bioinformatics*, **16**, 152–158.
Cai,L. *et al.* (2000) Evolutionary computation techniques for multiple sequence alignment. In, *Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No.00TH8512)*., pp. 829–835 vol.2.
Campbell,M. *et al.* (2002) Deep Blue. *Artificial Intelligence*, **134**, 57–83.
Cano,A. (2018) A survey on graphic processing unit computing for large-scale data mining. *WIREs Data Mining and Knowledge Discovery*, **8**, e1232.
Cantarel,B.L. *et al.* (2008) MAKER: An easy-to-use annotation pipeline designed for emerging model organism genomes. *Genome Res*, **18**, 188–196.
Cao,Y. *et al.* (2020) Ensemble deep learning in bioinformatics. *Nat Mach Intell*, **2**, 500–508.
Cartegni,L. *et al.* (2003) ESEfinder: a web resource to identify exonic splicing enhancers. *Nucleic Acids Res*, **31**, 3568–3571.
Cartegni,L. *et al.* (2002) Listening to silence and understanding nonsense: exonic mutations that affect splicing. *Nat Rev Genet*, **3**, 285–298.
Castelo,R. and Guigó,R. (2004) Splice site identification by idlBNs. *Bioinformatics*, **20**, i69–i76.
Chen,J. *et al.* (2017) A Convolutional Neural Network with Dynamic Correlation Pooling. In, *2017 13th International Conference on Computational Intelligence and Security (CIS)*., pp. 496–499.
Chen,T.-M. *et al.* (2005) Prediction of splice sites with dependency graphs and their expanded bayesian networks. *Bioinformatics*, **21**, 471–482.
Chen,W. and Moore,M.J. (2015) Spliceosomes. *Current Biology*, **25**, R181–R183.
Cheng,S. *et al.* (2018) 10KP: A phylodiverse genome sequencing plan. *Gigascience*, **7**, giy013.
Chennen,K. *et al.* (2020) MISTIC: A prediction tool to reveal disease-relevant deleterious missense variants. *PLOS ONE*, **15**, e0236962.
Cho,S. *et al.* (2011) Interaction between the RNA binding domains of Ser-Arg splicing factor 1 and U1-70K snRNP protein determines early spliceosome assembly. *PNAS*, **108**, 8233–8238.
Chowdhury,B. *et al.* (2017) An optimized approach for annotation of large eukaryotic genomic sequences using genetic algorithm. *BMC Bioinformatics*, **18**, 460.
Chowdhury,B. and Garai,G. (2017) A review on multiple sequence alignment from the perspective of genetic algorithm. *Genomics*, **109**, 419–431.
Chowdhury,G.G. (2003) Natural language processing. *Annual Review of Information Science and Technology*, **37**, 51–89.
Church,G.M. *et al.* (2012) Next-Generation Digital Information Storage in DNA. *Science*, **337**, 1628–1628.

Clough,E. and Barrett,T. (2016) The Gene Expression Omnibus database. *Methods Mol Biol*, **1418**, 93–110.
Cohen,I.B. (2003) Mark I, Harvard. In, *Encyclopedia of Computer Science*. John Wiley and Sons Ltd., GBR, pp. 1078–1080.
Collet,P. *et al.* (2000) Take it EASEA., pp. 891–901.
Cooper,D.N. *et al.* (1983) Unmethylated domains in vertebrate DNA. *Nucleic Acids Res*, **11**, 647–658.
Cooper,G.M. (2000) The Cell 2nd ed. Sinauer Associates.
Cooper,T.A. and Mattox,W. (1997) The regulation of splice-site selection, and its role in human disease. *Am J Hum Genet*, **61**, 259–266.
Copeland,B.J. (2004) Colossus: its origins and originators. *IEEE Annals of the History of Computing*, **26**, 38–45.
Corne,D.W. and Lones,M.A. (2018) Evolutionary Algorithms. *arXiv:1805.11014 [cs]*, 1–22.
Cortes,C. and Vapnik,V. (1995) Support-vector networks. *Mach Learn*, **20**, 273–297.
Cramer,N.L. (1985) A Representation for the Adaptive Generation of Simple Sequential Programs. In, *Proceedings of the 1st International Conference on Genetic Algorithms*. L. Erlbaum Associates Inc., USA, pp. 183–187.
Creswell,A. *et al.* (2018) Generative Adversarial Networks: An Overview. *IEEE Signal Process. Mag.*, **35**, 53–65.
Crick,F.H. (1958) On protein synthesis. *Symp Soc Exp Biol*, **12**, 138–163.
Danchin,A. *et al.* (2018) No wisdom in the crowd: genome annotation in the era of big data - current status and future prospects. *Microb. Biotechnol.*, **11**, 588–605.
Darwin,C. (1859) The origin of species. 313.
David,C.J. and Manley,J.L. (2010) Alternative pre-mRNA splicing regulation in cancer: pathways and programs unhinged. *Genes Dev*, **24**, 2343–2364.
Davis,M. and Davis,V. (2005) Mistaken Ancestry: The Jacquard and the Computer. *TEXTILE*, **3**, 76–87.
Degroeve,S. *et al.* (2002) Feature subset selection for splice site prediction. *Bioinformatics*, **18 Suppl 2**, S75-83.
Degroeve,S. *et al.* (2005) SpliceMachine: predicting splice sites from high-dimensional local context representations. *Bioinformatics*, **21**, 1332–1338.
Denton,J.F. *et al.* (2014) Extensive Error in the Number of Genes Inferred from Draft Genome Assemblies. *PLoS Comput Biol*, **10**, e1003998.
Deutekom,E.S. *et al.* (2019) Measuring the impact of gene prediction on gene loss estimates in Eukaryotes by quantifying falsely inferred absences. *PLoS Comput Biol*, **15**, e1007301.
Devos,D. and Valencia,A. (2001) Intrinsic errors in genome annotation. *Trends in Genetics*, **17**, 429–431.
Djebali,S. *et al.* (2012) Landscape of transcription in human cells. *Nature*, **489**, 101–108.
Dolfini,D. *et al.* (2009) A perspective of promoter architecture from the CCAAT box. *Cell Cycle*, **8**, 4127–4137.
Dunham,I. *et al.* (2012) An integrated encyclopedia of DNA elements in the human genome. *Nature*, **489**, 57–74.
Durbin,R.M. *et al.* (2010) A map of human genome variation from population-scale sequencing. *Nature*, **467**, 1061–1073.
Duyvesteyn,K. and Kaymak,U. (2005) Genetic programming in economic modelling. *The 2005 IEEE Congress on Evolutionary Computation*, 1025–1031.
Ebrahimzade,H. *et al.* (2018) A novel predictive model for estimation of cobalt leaching from waste Li-ion batteries: Application of genetic programming for design. *Journal of Environmental Chemical Engineering*, **6**, 3999–4007.
Edgar,R.C. and Myers,E.W. (2005) PILER: identification and classification of genomic repeats. *Bioinformatics*, **21**, i152–i158.
Ejigu,G.F. and Jung,J. (2020) Review on the Computational Genome Annotation of Sequences Obtained by Next-Generation Sequencing. *Biology (Basel)*, **9**, 295.
Elgammal,A. *et al.* (2017) CAN: Creative Adversarial Networks, Generating 'Art' by Learning About Styles and Deviating from Style Norms. *arXiv:1706.07068*.
Elsik,C.G. *et al.* (2007) Creating a honey bee consensus gene set. *Genome Biology*, **8**, R13.

Ezkurdia,I. *et al.* (2015) Most Highly Expressed Protein-Coding Genes Have a Single Dominant Isoform. *J Proteome Res*, **14**, 1880–1887.
Fang,Y. and Li,J. (2010) A Review of Tournament Selection in Genetic Programming. In, Cai,Z. *et al.* (eds), *Advances in Computation and Intelligence*, Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, pp. 181–192.
Faustino,N.A. and Cooper,T.A. (2003) Pre-mRNA splicing and human disease. *Genes Dev.*, **17**, 419–437.
Feigenbaum,E.A. (1992) Expert Systems: Principles and Practice.
Feng,J. and Lu,S. (2019) Performance Analysis of Various Activation Functions in Artificial Neural Networks. *J. Phys.: Conf. Ser.*, **1237**, 022030.
Ferat,J.-L. and Michel,F. (1993) Group II self-splicing introns in bacteria. *Nature*, **364**, 358–361.
Fica,S.M. *et al.* (2013) RNA catalyses nuclear pre-mRNA splicing. *Nature*, **503**, 229–234.
Fleischmann,R.D. *et al.* (1995) Whole-genome random sequencing and assembly of Haemophilus influenzae Rd. *Science*, **269**, 496–512.
Fogel,D.B. and Fogel,L.J. (1996) An introduction to evolutionary programming. In, Alliot,J.-M. *et al.* (eds), *Artificial Evolution*, Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, pp. 21–33.
Fortin,F.-A. *et al.* (2012) DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research, Machine Learning Open Source Software*, **13**, 2171–2175.
Frankish,A. *et al.* (2019) GENCODE reference annotation for the human and mouse genomes. *Nucleic Acids Res*, **47**, D766–D773.
Freeman,W.J. (1992) Tutorial on neurobiology: from single neurons to brain chaos. *Int. J. Bifurcation Chaos*, **02**, 451–482.
Frey,K. and Pucker,B. (2020) Animal, Fungi, and Plant Genome Sequences Harbor Different Non-Canonical Splice Sites. *Cells*, **9**, 458.
Friz,C.T. (1968) The biochemical composition of the free-living Amoebae Chaos chaos, amoeba dubia and Amoeba proteus. *Comparative Biochemistry and Physiology*, **26**, 81–90.
Fuchs,J. *et al.* (1999) CaDet, a computer-based clinical decision support system for early cancer detection. *Cancer Detect Prev*, **23**, 78–87.
Galland,J. (2020) La Médecine Interne 3.0. *La Revue de Médecine Interne*, **41**, 149–151.
Gao,X. *et al.* (2018) tRNA-DL: A Deep Learning Approach to Improve tRNAscan-SE Prediction Results. *HHE*, **83**, 163–172.
Gardiner-Garden,M. and Frommer,M. (1987) CpG Islands in vertebrate genomes. *Journal of Molecular Biology*, **196**, 261–282.
Gates,A.J. *et al.* (2021) A wealth of discovery built on the Human Genome Project — by the numbers. *Nature*, **590**, 212–215.
Gazzoli,I. *et al.* (2015) Non-sequential and multi-step splicing of the dystrophin transcript. *RNA Biol*, **13**, 290–305.
Geman,S. *et al.* (1992) Neural Networks and the Bias/Variance Dilemma. *Neural Computation*, **4**, 1–58.
Genome 10K Community of Scientists (2009) Genome 10K: A Proposal to Obtain Whole-Genome Sequence for 10 000 Vertebrate Species. *J Hered*, **100**, 659–674.
GIGA Community of Scientists (2014) The Global Invertebrate Genomics Alliance (GIGA): Developing Community Resources to Study Diverse Invertebrate Genomes. *J Hered*, **105**, 1–18.
Gilbert,W. (1978) Why genes in pieces? *Nature*, **271**, 501–501.
Gilks,W.R. *et al.* (2002) Modeling the percolation of annotation errors in a database of protein sequences. *Bioinformatics*, **18**, 1641–1649.
Gilks,W.R. *et al.* (2005) Percolation of annotation errors through hierarchically structured protein sequence databases. *Mathematical Biosciences*, **193**, 223–234.
Glover,F. (1986) Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, **13**, 533–549.
Gogna,A. and Tayal,A. (2013) Metaheuristics: review and application. *Journal of Experimental & Theoretical Artificial Intelligence*, **25**, 503–526.
Goodfellow,I.J. *et al.* (2014) Generative Adversarial Networks. *arXiv:1406.2661 [cs, stat]*.

Green,M.R. (1986) PRE-mRNA SPLICING. *Annu. Rev. Genet.*, **20**, 671–708.
Gremme,G. *et al.* (2005) Engineering a software tool for gene structure prediction in higher organisms. *Information and Software Technology*, **47**, 965–978.
Grigoriev,I.V. *et al.* (2014) MycoCosm portal: gearing up for 1000 fungal genomes. *Nucleic Acids Res*, **42**, D699–D704.
Guigó,R. *et al.* (2006) EGASP: the human ENCODE Genome Annotation Assessment Project. *Genome Biology*, 31.
Guigó,R. *et al.* (1992) Prediction of gene structure. *Journal of Molecular Biology*, **226**, 141–157.
Gunning,D. and Aha,D. (2019) DARPA's Explainable Artificial Intelligence (XAI) Program. *AI Magazine*, **40**, 44–58.
Gustafson,J. (2000) Reconstruction of the Atanasoff-Berry computer. *Conference: The first computers*.
Haas,B.J. *et al.* (2008) Automated eukaryotic gene structure annotation using EVidenceModeler and the Program to Assemble Spliced Alignments. *Genome Biol*, **9**, R7.
Haas,B.J. *et al.* (2003) Improving the Arabidopsis genome annotation using maximal transcript alignment assemblies. *Nucleic Acids Research*, **31**, 5654–5666.
Hadsell,R. *et al.* (2009) Learning long-range vision for autonomous off-road driving. *Journal of Field Robotics*, **26**, 120–144.
Haenlein,M. and Kaplan,A. (2019) A Brief History of Artificial Intelligence: On the Past, Present, and Future of Artificial Intelligence. *California Management Review*, **61**, 5–14.
Hahn,S. (2004) Structure and mechanism of the RNA Polymerase II transcription machinery. *Nat Struct Mol Biol*, **11**, 394–403.
Harding,S. and Banzhaf,W. (2007) Fast Genetic Programming and Artificial Developmental Systems on GPUs. In, *21st International Symposium on High Performance Computing Systems and Applications (HPCS'07)*., pp. 2–2.
Harris,T.W. *et al.* (2020) WormBase: a modern Model Organism Information Resource. *Nucleic Acids Research*, **48**, D762–D767.
Hart,W.E. and Istrail,S. (1997) Robust Proofs of NP-Hardness for Protein Folding: General Lattices and Energy Potentials. *Journal of Computational Biology*, **4**, 1–22.
Hatamochi,A. *et al.* (1988) A CCAAT DNA binding factor consisting of two different components that are both required for DNA binding. *Journal of Biological Chemistry*, **263**, 5940–5947.
Hawkins,D.M. (2004) The Problem of Overfitting. *J. Chem. Inf. Comput. Sci.*, **44**, 1–12.
Herbay,L. *et al.* (2021) Prédiction de sites d'épissage par une approche d'Algorithme Evolutionnaire.
Hershey,A.D. and Chase,M. (1952) Independent functions of viral protein and nucleic acid in growth of bacteriophage. *J Gen Physiol*, **36**, 39–56.
Hidalgo,O. *et al.* (2017) Is There an Upper Limit to Genome Size? *Trends in Plant Science*, **22**, 567–573.
Hien,N.T. *et al.* (2020) Genetic Programming for storm surge forecasting. *Ocean Engineering*, **215**, 107812.
Hinton,G.E. *et al.* (2012) Improving neural networks by preventing co-adaptation of feature detectors. *arXiv:1207.0580*.
Hochreiter,S. and Schmidhuber,J. (1997) Long Short-Term Memory. *Neural Computation*, **9**, 1735–1780.
Hoff,K. *et al.* (2019) Whole-Genome Annotation with BRAKER. *Methods Mol Biol*, **1962**, 65–95.
Hoff,K.J. *et al.* (2016) BRAKER1: Unsupervised RNA-Seq-Based Genome Annotation with GeneMark-ET and AUGUSTUS: Table 1. *Bioinformatics*, **32**, 767–769.
Holland,J.H. (1992) Genetic Algorithms. *Scientific American*, **267**, 66–73.
Holt,C. and Yandell,M. (2011) MAKER2: an annotation pipeline and genome-database management tool for second-generation genome projects. *BMC Bioinformatics*, **12**, 491.
Hopfield,J.J. (1982) Neural networks and physical systems with emergent collective computational abilities. *PNAS*, **79**, 2554–2558.
Horikoshi,M. *et al.* (1990) Analysis of structure-function relationships of yeast TATA box binding factor TFIID. *Cell*, **61**, 1171–1178.
Howe,K.L. *et al.* (2020) Ensembl 2021. *Nucleic Acids Res*, **49**, D884–D891.

Hsi-Yang Fritz,M. *et al.* (2011) Efficient storage of high throughput DNA sequencing data using reference-based compression. *Genome Res*, **21**, 734–740.
Huang,Y. *et al.* (2016) Well-characterized sequence features of eukaryote genomes and implications for ab initio gene prediction. *Computational and Structural Biotechnology Journal*, **14**, 298–303.
Hubel,D.H. and Wiesel,T.N. (1959) Receptive fields of single neurones in the cat's striate cortex. *The Journal of Physiology*, **148**, 574–591.
Ibarra-Laclette,E. *et al.* (2013) Architecture and evolution of a minute plant genome. *Nature*, **498**, 94–98.
Jaganathan,K. *et al.* (2019) Predicting Splicing from Primary Sequence with Deep Learning. *Cell*, **176**, 535-548.e24.
Jain,S. *et al.* (2020) Trustworthiness of Artificial Intelligence. In, *2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)*., pp. 907–912.
Jankowski,J.M. and Dixon,G.H. (1987) The GC box as a silencer. *Biosci Rep*, **7**, 955–963.
Jian,X. *et al.* (2014) In silico tools for splicing defect prediction: a survey from the viewpoint of end users. *Genet Med*, **16**, 497–503.
John,G.H. *et al.* (1994) Irrelevant Features and the Subset Selection Problem. In, *Machine Learning: Proceedings of the Eleventh International*. Morgan Kaufmann, pp. 121–129.
Jorquera,R. *et al.* (2018) Improved ontology for eukaryotic single-exon coding sequences in biological databases. *Database (Oxford)*, **2018**, bay089.
Jumper,J. *et al.* (2021) Highly accurate protein structure prediction with AlphaFold. *Nature*, 1–11.
Jurica,M.S. and Moore,M.J. (2003) Pre-mRNA Splicing: Awash in a Sea of Proteins. *Molecular Cell*, **12**, 5–14.
Kaelbling,L.P. *et al.* (1996) Reinforcement Learning: A Survey. *arXiv:cs/9605103*.
Kalari,K.R. *et al.* (2006) First Exons and Introns – A Survey of GC Content and Gene Structure in the Human Genome. *In Silico Biology*, **6**, 237–242.
Kamath,U. *et al.* (2012) An Evolutionary Algorithm Approach for Feature Generation from Sequence Data and Its Application to DNA Splice Site Prediction. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, **9**, 1387–1398.
Kaminuma,E. *et al.* (2011) DDBJ progress report. *Nucleic Acids Res*, **39**, D22–D27.
Kanehisa,M. and Goto,S. (2000) KEGG: Kyoto Encyclopedia of Genes and Genomes. *Nucleic Acids Res*, **28**, 27–30.
Katoch,S. *et al.* (2021) A review on genetic algorithm: past, present, and future. *Multimed Tools Appl*, **80**, 8091–8126.
Katoh,K. *et al.* (2002) MAFFT: a novel method for rapid multiple sequence alignment based on fast Fourier transform. *Nucleic Acids Res*, **30**, 3059–3066.
Keilwagen,J. *et al.* (2018) Combining RNA-seq data and homology-based gene prediction for plants, animals and fungi. *BMC Bioinformatics*, **19**, 189.
Keilwagen,J. *et al.* (2016) Using intron position conservation for homology-based gene prediction. *Nucleic Acids Res*, **44**, e89.
Kelemen,O. *et al.* (2013) Function of alternative splicing. *Gene*, **514**, 1–30.
Kent,W.J. *et al.* (2002) The Human Genome Browser at UCSC. *Genome Res*, **12**, 996–1006.
Kerby,K. and Kuspira,J. (1987) The phylogeny of the polyploid wheats Triticum aestivum (bread wheat) and Triticum turgidum (macaroni wheat). *Genome*, **29**, 722–737.
Khenoussi,W. *et al.* (2014) SIBIS: a Bayesian model for inconsistent protein sequence estimation. *Bioinformatics*, **30**, 2432–2439.
Kilkenny,M.F. and Robinson,K.M. (2018) Data quality: "Garbage in – garbage out". *HIM J*, **47**, 103–105.
Kiryutin,B. *et al.* (2021) ProSplign – Protein to Genomic Alignment Tool.
Kleene,S.C. (1951) Representation of Events in Nerve Nets and Finite Automata.
Kolpakov,R. *et al.* (2003) mreps: efficient and flexible detection of tandem repeats in DNA. *Nucleic Acids Res*, **31**, 3672–3678.
Korf,I. (2004) Gene finding in novel genomes. *BMC Bioinformatics*, 9.
Korf,I. *et al.* (2001) Integrating genomic homology into gene structure prediction. *Bioinformatics*, **17**, S140–S148.

Koumakis,L. (2020) Deep learning models in genomics; are we there yet? *Computational and Structural Biotechnology Journal*, **18**, 1466–1473.
Kovaka,S. *et al.* (2019) Transcriptome assembly from long-read RNA-seq alignments with StringTie2. *Genome Biology*, **20**, 278.
Koza,J.R. (1994) Genetic programming as a means for programming computers by natural selection. *Stat Comput*, **4**, 87–112.
Koza,J.R. *et al.* (1999) Genetic Programming III: Darwinian Invention and Problem Solving Morgan Kaufmann.
Koza,J.R. (1992) Genetic Programming: On the Programming of Computers by Means of Natural Selection MIT Press.
Kozak,M. (1981) Possible role of flanking nucleotides in recognition of the AUG initiator codon by eukaryotic ribosomes. *Nucleic Acids Res*, **9**, 5233–5252.
Kriventseva,E.V. *et al.* (2015) OrthoDB v8: update of the hierarchical catalog of orthologs and the underlying free software. *Nucleic Acids Res*, **43**, D250–D256.
Kronfeld,M. *et al.* (2010) The EvA2 Optimization Framework. In, Blum,C. and Battiti,R. (eds), *Learning and Intelligent Optimization*, Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, pp. 247–250.
Kumar,R. and Jyotishree (2012) Blending Roulette Wheel Selection & Rank Selection in Genetic Algorithms. *International Journal of Machine Learning and Computing*, 365–370.
Lally,A. and Fodor,P. (2011) Natural Language Processing With Prolog in the IBM Watson System.
Lander,E.S. *et al.* (2001) Initial sequencing and analysis of the human genome. *Nature*, **409**, 860–921.
Langdon,W.B. (2011) Graphics processing units and genetic programming: an overview. *Soft Comput*, **15**, 1657–1669.
Langton,C.G. (1997) Artificial Life: An Overview MIT Press.
Larkin,A. *et al.* (2020) FlyBase: updates to the Drosophila melanogaster knowledge base. *Nucleic Acids Res*, **49**, D899–D907.
LaRoche-Johnston,F. *et al.* (2018) Bacterial group II introns generate genetic diversity by circularization and trans-splicing from a population of intron-invaded mRNAs. *PLOS Genetics*, **14**, e1007792.
LeCun,Y. *et al.* (1989) Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation*, **1**, 541–551.
LeCun,Y. *et al.* (2015) Deep learning. *Nature*, **521**, 436–444.
Lecun,Y. *et al.* (1998) Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, **86**, 2278–2324.
Lecun,Y. (1985) Une procedure d'apprentissage pour reseau a seuil asymmetrique (A learning scheme for asymmetric threshold networks). *Proceedings of Cognitiva 85, Paris, France*, 599–604.
Lecun,Y. and Bengio,Y. (1995) Convolutional networks for images, speech, and time-series. *The handbook of brain theory and neural networks*.
Lee,B. *et al.* (2015) DNA-Level Splice Junction Prediction using Deep Recurrent Neural Networks. *arXiv:1512.05135 [cs, q-bio]*.
Lee,D. and Han,K. (2003) A Genetic Algorithm for Predicting RNA Pseudoknot Structures. In, Sloot,P.M.A. *et al.* (eds), *Computational Science — ICCS 2003*, Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, pp. 130–139.
Lee,Y. and Rio,D.C. (2015) Mechanisms and Regulation of Alternative Pre-mRNA Splicing. *Annu Rev Biochem*, **84**, 291–323.
Leff,S.E. *et al.* (1986) Complex transcriptional units: diversity in gene expression by alternative rna processing. *Annu. Rev. Biochem.*, **55**, 1091–1117.
Leijnen,S. and Veen,F. van (2020) The Neural Network Zoo. *Proceedings*, **47**, 9.
Letunic,I. and Bork,P. (2021) Interactive Tree Of Life (iTOL) v5: an online tool for phylogenetic tree display and annotation. *Nucleic Acids Research*.
Leung,F.H.F. *et al.* (2003) Tuning of the structure and parameters of a neural network using an improved genetic algorithm. *IEEE Transactions on Neural Networks*, **14**, 79–88.
Levenshtein,V. (1965) Binary codes capable of correcting deletions, insertions, and reversals. *undefined*.
Levitt,G.M. (2000) The Turk, Chess Automaton McFarland, Incorporated, Publishers.

Li,H. *et al.* (2020) Modern Deep Learning in Bioinformatics. *J Mol Cell Biol*.
Li,H. *et al.* (2009) The Sequence Alignment/Map format and SAMtools. *Bioinformatics*, **25**, 2078–2079.
Li,Wen *et al.* (2017) Prediction of Splice Site using Support Vector Machine with Feature Selection. In, *Proceedings of the International Conference on Bioinformatics and Computational Intelligence*, ICBCI 2017. Association for Computing Machinery, New York, NY, USA, pp. 1–5.
Li,Y. and Chen,L. (2014) Big Biological Data: Challenges and Opportunities. *Genomics Proteomics Bioinformatics*, **12**, 187–189.
Lim,L.P. and Burge,C.B. (2001) A computational analysis of sequence features involved in recognition of short introns. *PNAS*, **98**, 11193–11198.
Lipton,Z.C. *et al.* (2015) A Critical Review of Recurrent Neural Networks for Sequence Learning. *arXiv:1506.00019 [cs]*.
Lisowiec,J. *et al.* (2015) Structural determinants for alternative splicing regulation of the MAPT pre-mRNA. *RNA Biol*, **12**, 330–342.
Liu,K.-H. and Xu,C.-G. (2009) A genetic programming-based approach to the classification of multiclass microarray datasets. *Bioinformatics*, **25**, 331–337.
Liu,Y.-C. and Cheng,S.-C. (2015) Functional roles of DExD/H-box RNA helicases in Pre-mRNA splicing. *Journal of Biomedical Science*, **22**, 54.
Lohn,J.D. *et al.* (2004) Evolutionary design of an X-band antenna for NASA's Space Technology 5 mission. In, *IEEE Antennas and Propagation Society Symposium, 2004.*, pp. 2313-2316 Vol.3.
Lomsadze,A. (2005) Gene identification in novel eukaryotic genomes by self-training algorithm. *Nucleic Acids Research*, **33**, 6494–6506.
Lonsdale,J. *et al.* (2013) The Genotype-Tissue Expression (GTEx) project. *Nat Genet*, **45**, 580–585.
López-Bigas,N. *et al.* (2005) Are splicing mutations the most frequent cause of hereditary disease? *FEBS Letters*, **579**, 1900–1903.
Lou,D.I. *et al.* (2013) High-throughput DNA sequencing errors are reduced by orders of magnitude using circle sequencing. *Proc Natl Acad Sci U S A*, **110**, 19872–19877.
Lowe,T.M. and Eddy,S.R. (1997) tRNAscan-SE: a program for improved detection of transfer RNA genes in genomic sequence. *Nucleic Acids Res*, **25**, 955–964.
Maitre,O. *et al.* (2017) Parallelizing Evolutionary Algorithms on GPGPU Cards with the EASEA Platform. In, *Programming multi-core and many-core computing systems*. John Wiley & Sons, Ltd, pp. 301–319.
Maji,S. and Garg,D. (2014) Hybrid Approach Using SVM and MM2 in Splice Site Junction Identification. *Current Bioinformatics*, **9**, 76–85.
Majoros,W.H. *et al.* (2004) TigrScan and GlimmerHMM: two open source ab initio eukaryotic gene-finders. *Bioinformatics*, **20**, 2878–2879.
Makałowski,W. (2001) The human genome structure and organization. *Acta Biochim Pol*, **48**, 587–598.
Martínez-Arellano,G. *et al.* (2017) Creating AI Characters for Fighting Games Using Genetic Programming. *IEEE Transactions on Computational Intelligence and AI in Games*, **9**, 423–434.
Matera,A.G. and Wang,Z. (2014) A day in the life of the spliceosome. *Nat Rev Mol Cell Biol*, **15**, 108–121.
Matlin,A.J. *et al.* (2005) Understanding alternative splicing: towards a cellular code. *Nat Rev Mol Cell Biol*, **6**, 386–398.
McCarthy,J. (1960) Recursive functions of symbolic expressions and their computation by machine, Part I. *Commun. ACM*, **3**, 184–195.
McCulloch,W.S. and Pitts,W. (1943) A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, **5**, 115–133.
McDowell,J.J. and Popa,A. (2010) Toward a Mechanics of Adaptive Behavior: Evolutionary Dynamics and Matching Theory Statics. *J Exp Anal Behav*, **94**, 241–260.
McLaren,W. *et al.* (2016) The Ensembl Variant Effect Predictor. *Genome Biology*, **17**, 122.
Meher,P.K. *et al.* (2016) Prediction of donor splice sites using random forest with a new sequence encoding approach. *BioData Mining*, **9**, 4.

Mercer,T.R. *et al.* (2015) Genome-wide discovery of human splicing branchpoints. *Genome Res.*, **25**, 290–303.
Meyer,C. *et al.* (2020) Understanding the causes of errors in eukaryotic protein-coding gene prediction: a case study of primate proteomes. *BMC Bioinformatics*, **21**, 513.
Miao,W. *et al.* (2020) Protist 10,000 Genomes Project. *The Innovation*, **1**, 100058.
Miikkulainen,R. (2020) Creative AI Through Evolutionary Computation. *arXiv:1901.03775 [cs]*.
Miller,B.L. *et al.* (1995) Genetic Algorithms, Tournament Selection, and the Effects of Noise. *Complex Systems*, **9**, 193–212.
Min,S. *et al.* (2017) Deep learning in bioinformatics. *Briefings in Bioinformatics*, **18**, 851–869.
Mirjalili,S. (2019) Genetic Algorithm. In, Mirjalili,S. (ed), *Evolutionary Algorithms and Neural Networks: Theory and Applications*, Studies in Computational Intelligence. Springer International Publishing, Cham, pp. 43–55.
Møller,L.B. *et al.* (2000) Similar Splice-Site Mutations of the ATP7A Gene Lead to Different Phenotypes: Classical Menkes Disease or Occipital Horn Syndrome. *The American Journal of Human Genetics*, **66**, 1211–1220.
Montes,M. *et al.* (2019) RNA splicing and disease: animal models to therapies. *Trends Genet*, **35**, 68–87.
Moody,J. *et al.* (1991) Training Knowledge-Based Neural Networks to Recognize Genes in DNA Sequences.
Moor,J. (2006) The Dartmouth College Artificial Intelligence Conference: The Next Fifty Years. *AI Magazine*, **27**, 87–87.
Moore,G.E. (1965) Cramming more components onto integrated circuits. **38**, 4.
Morgan,J.N. and Sonquist,J.A. (1963) Problems in the Analysis of Survey Data, and a Proposal. *Journal of the American Statistical Association*, **58**, 415–434.
Mudge,J.M. and Harrow,J. (2016) The state of play in higher eukaryote gene annotation. *Nat Rev Genet*, **17**, 758–772.
Murtagh,F. (1991) Multilayer perceptrons for classification and regression. *Neurocomputing*, **2**, 183–197.
Naito,T. (2018) Human Splice-Site Prediction with Deep Neural Networks. *Journal of Computational Biology*, **25**, 954–961.
Nasibov,E. and Tunaboylu,S. (2010) Classification of splice-junction sequences via weighted position specific scoring approach. *Computational Biology and Chemistry*, **34**, 293–299.
Neumann,M. *et al.* (2006) Ubiquitinated TDP-43 in Frontotemporal Lobar Degeneration and Amyotrophic Lateral Sclerosis. *Science*, **314**, 130–133.
Nevers,Y. *et al.* (2019) OrthoInspector 3.0: open portal for comparative genomics. *Nucleic Acids Res*, **47**, D411–D418.
Newell,A. (1958) Report on a general problem-solving. *Carnegie Institute of Technology*.
Nguyen,H. *et al.* (2018) The matrices and constraints of GT/AG splice sites of more than 1000 species/lineages. *Gene*, **660**, 92–101.
Niazkar,M. and Reza,H. (2020) COVID-19 Outbreak: Application of Multi-gene Genetic Programming to Country-based Prediction Models. *ELECTRON J GEN MED*, **17**, em247.
Nikoloski,Z. *et al.* (2008) Metabolic networks are NP-hard to reconstruct. *Journal of Theoretical Biology*, **254**, 807–816.
Nissim-Rafinia,M. and Kerem,B. (2002) Splicing regulation as a potential genetic modifier. *Trends in Genetics*, **18**, 123–127.
Nizam,A. *et al.* (2011) Cyclic Genetic Algorithm for Multiple Sequence Alignment. *Science Academy Publisher International Journal of Research and Reviews in Electrical and Computer Engineering*, **1**, 2046–5149.
Northcutt,C.G. *et al.* (2021) Pervasive Label Errors in Test Sets Destabilize Machine Learning Benchmarks. *arXiv:2103.14749 [cs, stat]*.
Notredame,C. and Higgins,D.G. (1996) SAGA: Sequence Alignment by Genetic Algorithm. *Nucleic Acids Research*, **24**, 1515–1524.
Nurk,S. *et al.* (2021) The complete sequence of a human genome. *bioRxiv*, 2021.05.26.445798.
Ohno,S. (1972) So much 'junk' DNA in our genome. *Brookhaven Symp Biol*, **23**, 366–370.

Orphanides,G. *et al.* (1996) The general transcription factors of RNA polymerase II. *Genes Dev.*, **10**, 2657–2683.
O'Shea,K. and Nash,R. (2015) An Introduction to Convolutional Neural Networks. *arXiv:1511.08458 [cs]*.
Ozsolak,F. and Milos,P.M. (2011) RNA sequencing: advances, challenges and opportunities. *Nat Rev Genet*, **12**, 87–98.
Padgett,R.A. (2012) New connections between splicing and human disease. *Trends Genet*, **28**, 147–154.
Padgett,R.A. *et al.* (1986) Splicing of messenger rna precursors. *Annu. Rev. Biochem.*, **55**, 1119–1150.
Pal,S. *et al.* (2020) Big data in biology: The hope and present-day challenges in it. *Gene Reports*, **21**, 100869.
Pan,Q. *et al.* (2008) Deep surveying of alternative splicing complexity in the human transcriptome by high-throughput sequencing. *Nature Genetics*, **40**, 1413–1415.
Pan,S.J. and Yang,Q. (2010) A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering*, **22**, 1345–1359.
Panesar,S. *et al.* (2019) Artificial Intelligence and the Future of Surgical Robotics. *Annals of Surgery*, **270**, 223–226.
Panimalar,A. (2017) The 17 V's of Big Data. *International Research Journal of Engineering and Technology*, **4**, 329–333.
Papin,C. *et al.* (2021) CpG Islands Shape the Epigenome Landscape. *Journal of Molecular Biology*, **433**, 166659.
Pashaei,E., Yilmaz,A., *et al.* (2016) A novel method for splice sites prediction using sequence component and hidden Markov model. *Annu Int Conf IEEE Eng Med Biol Soc*, **2016**, 3076–3079.
Pashaei,E., Ozen,M., *et al.* (2016) Splice site identification in human genome using random forest. *Health and Technology*, **1**, 141–152.
Patel,A.A. and Steitz,J.A. (2003) Splicing double: insights from the second spliceosome. *Nat Rev Mol Cell Biol*, **4**, 960–970.
Paul,T.K. and Iba,H. (2009) Prediction of cancer class with majority voting genetic programming classifier using gene expression data. *IEEE/ACM Trans Comput Biol Bioinform*, **6**, 353–367.
Pedersen,A.G. *et al.* (1999) The biology of eukaryotic promoter prediction—a review. *Computers & Chemistry*, **23**, 191–207.
Pedersen,J.T. and Moult,J. (1996) Genetic algorithms for protein structure prediction. *Current Opinion in Structural Biology*, **6**, 227–231.
Pedregosa,F. *et al.* (2011) Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, **12**, 2825–2830.
Pertea,M. *et al.* (2001) GeneSplicer: a new computational method for splice site prediction. *Nucleic Acids Res*, **29**, 1185–1190.
Pesole,G. *et al.* (2000) The untranslated regions of eukaryotic mRNAs: Structure, function, evolution and bioinformatic tools for their analysis. *Briefings in Bioinformatics*, **1**, 236–249.
Pevzner,P. (2000) Computational Molecular Biology: An Algorithmic Approach MIT Press.
Philipp,G. *et al.* (2018) The exploding gradient problem demystified - definition, prevalence, impact, origin, tradeoffs, and solutions. *arXiv:1712.05577 [cs]*.
Plewniak,F. *et al.* (2003) PipeAlign: a new toolkit for protein family analysis. *Nucleic Acids Research*, **31**, 3829–3832.
Poli,R. *et al.* (2008) A Field Guide to Genetic Programming.
Poli,R. and Langdon,W.B. (1998) On the Search Properties of Different Crossover Operators in Genetic Programming. In, *University of Wisconsin*. Morgan Kaufmann, pp. 293–301.
Pollastro,P. and Rampone,S. (2002) Hs3d, a dataset of homo sapiens splice regions, and its extraction procedure from a major public database. *Int. J. Mod. Phys. C*, **13**, 1105–1117.
Poverennaya,I.V. and Roytberg,M.A. (2020) Spliceosomal Introns: Features, Functions, and Evolution. *Biochemistry Moscow*, **85**, 725–734.
Promponas,V.J. *et al.* (2015) Annotation inconsistencies beyond sequence similarity-based function prediction – phylogeny and genome structure. *Stand Genomic Sci*, **10**, 108.

Prosdocimi,F. *et al.* (2012) Controversies in modern evolutionary biology: the imperative for error detection and quality control. *BMC Genomics*, **13**, 5.
Pruitt,K.D. *et al.* (2007) NCBI reference sequences (RefSeq): a curated non-redundant sequence database of genomes, transcripts and proteins. *Nucleic Acids Res*, **35**, D61–D65.
Pucker,B. and Brockington,S.F. (2018) Genome-wide analyses supported by RNA-Seq reveal non-canonical splice sites in plant genomes. *BMC Genomics*, **19**, 980.
Pyle,A.M. (2016) Group II Intron Self-Splicing. *Annu. Rev. Biophys.*, **45**, 183–205.
Qian,N. (1999) On the momentum term in gradient descent learning algorithms. *Neural Networks*, **12**, 145–151.
Qin,Z. *et al.* (2018) The History of Computing. In, Qin,Z. *et al.* (eds), *Fundamentals of Software Culture*. Springer, Singapore, pp. 1–36.
Razali,N. and Geraghty,J. (2011) Genetic Algorithm Performance with Different Selection Strategies in Solving TSP.
Rechenberg,I. (1965) Cybernetic Solution Path of an Experimental Problem Ministry of Aviation.
Reda,I. *et al.* (2018) A new CNN-based system for early diagnosis of prostate cancer. In, *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*., pp. 207–210.
Reed,R. (2000) Mechanisms of fidelity in pre-mRNA splicing. *Current Opinion in Cell Biology*, **12**, 340–345.
Reese,M.G. *et al.* (1997) Improved Splice Site Detection in Genie. *Journal of Computational Biology*, **4**, 311–323.
Reja,R. *et al.* (2009) MitoInteractome: Mitochondrial protein interactome database, and its application in 'aging network' analysis. *BMC Genomics*, **10**, S20.
Riepe,T.V. *et al.* Benchmarking deep learning splice prediction tools using functional splice assays. *Human Mutation*, **n/a**.
Robart,A.R. and Zimmerly,S. (2005) Group II intron retroelements: function and diversity. *CGR*, **110**, 589–597.
Rojas,R. (1997) Konrad Zuse's legacy: the architecture of the Z1 and Z3. *IEEE Annals of the History of Computing*, **19**, 5–16.
Rosenblatt,F. (1958) The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, **65**, 386–408.
Rout,S.B. *et al.* (2017) Protein secondary structure prediction of PDB 4HU7 using Genetic Algorithm (GA). In, *2017 International Conference on Computer Communication and Informatics (ICCCI)*., pp. 1–6.
Ruder,S. (2017) An overview of gradient descent optimization algorithms. *arXiv:1609.04747 [cs]*.
Rumelhart,D.E. *et al.* (1986) Learning representations by back-propagating errors. *Nature*, **323**, 533–536.
Sagiroglu,S. and Sinanc,D. (2013) Big data: A review. In, *2013 International Conference on Collaboration Technologies and Systems (CTS)*., pp. 42–47.
Sahlin,K. and Mäkinen,V. (2021) Accurate spliced alignment of long RNA sequencing reads. *bioRxiv*, 2020.09.02.279208.
Salzberg,S.L. *et al.* (1999) Interpolated Markov Models for Eukaryotic Gene Finding. *Genomics*, **59**, 24–31.
Salzberg,S.L. (2019) Next-generation genome annotation: we still struggle to get it right. *Genome Biol*, **20**, 92, s13059-019-1715–2.
Samuel,A.L. (1959) Some Studies in Machine Learning Using the Game of Checkers. *IBM Journal of Research and Development*, **3**, 210–229.
Samuel,G.N. and Farsides,B. (2017) The UK's 100,000 Genomes Project: manifesting policymakers' expectations. *New Genet Soc*, **36**, 336–353.
Sanger,F. *et al.* (1977) DNA sequencing with chain-terminating inhibitors. *Proc Natl Acad Sci U S A*, **74**, 5463–5467.
Saxonov,S. *et al.* (2006) A genome-wide analysis of CpG dinucleotides in the human genome distinguishes two distinct classes of promoters. *Proc Natl Acad Sci U S A*, **103**, 1412–1417.
Schmidt,C.A. and Matera,A.G. (2020) tRNA introns: Presence, processing, and purpose. *WIREs RNA*, **11**, e1583.

Schnoes,A.M. *et al.* (2009) Annotation Error in Public Databases: Misannotation of Molecular Function in Enzyme Superfamilies. *PLoS Comput Biol*, **5**.
Sehra,S. *et al.* (2021) Undecidability of Underfitting in Learning Algorithms. *arXiv:2102.02850 [cs, math]*.
Seppey,M. *et al.* (2019) BUSCO: Assessing Genome Assembly and Annotation Completeness. In, Kollmar,M. (ed), *Gene Prediction: Methods and Protocols*, Methods in Molecular Biology. Springer, New York, NY, pp. 227–245.
Shapiro,M.B. and Senapathy,P. (1987) RNA splice junctions of different classes of eukaryotes: sequence statistics and functional implications in gene expression. *Nucleic Acids Res*, **15**, 7155–7174.
Sharma,Siddharth *et al.* (2020) Activation functions in neural networks. *International Journal of Engineering Applied Sciences and Technology*, **04**, 310–316.
Shaul,O. (2017) How introns enhance gene expression. *The International Journal of Biochemistry & Cell Biology*, **91**, 145–155.
Sheth,N. *et al.* (2006) Comprehensive splice-site analysis using comparative genomics. *Nucleic Acids Res*, **34**, 3955–3967.
Shi,Y. (2017) Mechanistic insights into precursor messenger RNA splicing by the spliceosome. *Nat Rev Mol Cell Biol*, **18**, 655–670.
Siezen,R.J. and Hijum,S.A.F.T.V. (2010) Genome (re-)annotation and open-source annotation pipelines. *Microbial Biotechnology*, **3**, 362–369.
Sigrist,C.J.A. *et al.* (2013) New and continuing developments at PROSITE. *Nucleic Acids Res*, **41**, D344–D347.
Singh,R.K. and Cooper,T.A. (2012) Pre-mRNA splicing in disease and therapeutics. *Trends Mol Med*, **18**, 472–482.
Singh,S. *et al.* (2015) A Review of Computational Intelligence Methods for Eukaryotic Promoter Prediction. *Nucleosides Nucleotides Nucleic Acids*, **34**, 449–462.
Slater,G.S.C. and Birney,E. (2005) Automated generation of heuristics for biological sequence comparison. *BMC Bioinformatics*, **6**, 31.
Sloss,A.N. and Gustafson,S. (2019) 2019 Evolutionary Algorithms Review. *arXiv:1906.08870 [cs]*.
Smit,A.F. (1999) Interspersed repeats and other mementos of transposable elements in mammalian genomes. *Current Opinion in Genetics & Development*, **9**, 657–663.
Smith,C.W.J. and Valcárcel,J. (2000) Alternative pre-mRNA splicing: the logic of combinatorial control. *Trends in Biochemical Sciences*, **25**, 381–388.
Soemedi,R. *et al.* (2017) The Effects of Structure on pre-mRNA Processing and Stability. *Methods*, **125**, 36–44.
Solovyev,V.V. *et al.* (2010) Identification of promoter regions and regulatory sites. *Methods Mol Biol*, **674**, 57–83.
Sonnenburg,S. *et al.* (2007) Accurate splice site prediction using support vector machines. *BMC Bioinformatics*, **8**, S7.
Souvorov,A. *et al.* (2010) Gnomon – NCBI eukaryotic gene prediction.
Spanhol,F.A. *et al.* (2016) Breast cancer histopathological image classification using Convolutional Neural Networks. In, *2016 International Joint Conference on Neural Networks (IJCNN)*., pp. 2560–2567.
Spitz,J. *et al.* (2021) Video assistant referees (VAR): The impact of technology on decision making in association football referees. *J Sports Sci*, **39**, 147–153.
Srivastava,N. *et al.* (2014) Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, **15**, 1929–1958.
Staley,J.P. and Woolford,J.L. (2009) Assembly of ribosomes and spliceosomes: complex ribonucleoprotein machines. *Curr Opin Cell Biol*, **21**, 109–118.
Stamm,S. *et al.* (2000) An Alternative-Exon Database and Its Statistical Analysis. *DNA and Cell Biology*, **19**, 739–756.
Stanke,M. *et al.* (2006) Gene prediction in eukaryotes with a generalized hidden Markov model that uses hints from external sources. *BMC Bioinformatics*, **7**, 62.
Stanke,M. and Waack,S. (2003) Gene prediction with a hidden Markov model and a new intron submodel. *Bioinformatics*, **19**, ii215–ii225.

Stephens,Z.D. *et al.* (2015) Big Data: Astronomical or Genomical? *PLOS Biology*, **13**, e1002195.
Stone,M. (1974) Cross-Validatory Choice and Assessment of Statistical Predictions. *Journal of the Royal Statistical Society. Series B (Methodological)*, **36**, 111–147.
Storn,R. and Price,K. (1997) Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. *Journal of Global Optimization*, **11**, 341–359.
Suárez-Paniagua,V. and Segura-Bedmar,I. (2018) Evaluation of pooling operations in convolutional architectures for drug-drug interaction extraction. *BMC Bioinformatics*, **19**, 209.
Suganuma,M. *et al.* (2017) A Genetic Programming Approach to Designing Convolutional Neural Network Architectures. *arXiv:1704.00764 [cs]*.
Sun,Y.-F. *et al.* (2003) Identifying splicing sites in eukaryotic RNA: support vector machine approach. *Computers in Biology and Medicine*, **33**, 17–29.
Sutskever,I. *et al.* (2014) Sequence to Sequence Learning with Neural Networks. *arXiv:1409.3215 [cs]*.
Suzuki,Y. *et al.* (2001) Identification and Characterization of the Potential Promoter Regions of 1031 Kinds of Human Genes. *Genome Res*, **11**, 677–684.
Tang,B. *et al.* (2019) Recent Advances of Deep Learning in Bioinformatics and Computational Biology. *Front. Genet.*, **10**.
Tange,T.Ø. *et al.* (2001) The hnRNP A1 protein regulates HIV-1 tat splicing via a novel intron silencer element. *EMBO J*, **20**, 5748–5758.
Tazi,J. *et al.* (2009) Alternative splicing and disease. *Biochim Biophys Acta*, **1792**, 14–26.
The i5K consortium (2013) The i5K Initiative: Advancing Arthropod Genomics for Knowledge, Human Health, Agriculture, and the Environment. *J Hered*, **104**, 595–600.
Thompson,J.D. *et al.* (2000) DbClustal: rapid and reliable global multiple alignments of protein sequences detected by database searches. *Nucleic Acids Res*, **28**, 2919–2926.
Thompson,J.D. *et al.* (2003) RASCAL: rapid scanning and correction of multiple sequence alignments. *Bioinformatics*, **19**, 1155–1161.
Tsutsui,S. and Collet,P. eds. (2013) Massively Parallel Evolutionary Computation on GPGPUs Springer-Verlag, Berlin Heidelberg.
Tümer,Z. and Møller,L.B. (2010) Menkes disease. *Eur J Hum Genet*, **18**, 511–518.
Turing,A.M. (1950) Computing machinery anf intelligence. *Mind*, **LIX**, 433–460.
Turing,A.M. (1937) On Computable Numbers, with an Application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, **s2-42**, 230–265.
Turunen,J.J. *et al.* (2013) The significant other: splicing by the minor spliceosome. *Wiley Interdiscip Rev RNA*, **4**, 61–76.
Uecker,F.A. (1993) Development and Cytology of Plectosphaerella Cucumerina. *Mycologia*, **85**, 470–479.
Umarov,R.K. and Solovyev,V.V. (2017) Recognition of prokaryotic and eukaryotic promoters using convolutional deep learning neural networks. *PLOS ONE*, **12**, e0171410.
Umbarkar,Dr.A. and Sheth,P. (2015) Crossover operators in genetic algorithms; a review. *ICTACT Journal on Soft Computing (Volume: 6 , Issue: 1)*, **6**.
The UniProt Consortium (2017) UniProt: the universal protein knowledgebase. *Nucleic Acids Res*, **45**, D158–D169.
Van der Aalst,W.M.P. *et al.* (2008) Process mining: a two-step approach to balance between underfitting and overfitting. *Softw Syst Model*, **9**, 87.
Vanichkina,D.P. *et al.* (2018) Challenges in defining the role of intron retention in normal biology and disease. *Seminars in Cell & Developmental Biology*, **75**, 40–49.
Vargas,D.Y. *et al.* (2011) Single-Molecule Imaging of Transcriptionally Coupled and Uncoupled Splicing. *Cell*, **147**, 1054–1065.
Venter,J.C. *et al.* (2001) The Sequence of the Human Genome. *Science*, **291**, 1304–1351.
Vinyals,O. *et al.* (2015) Show and Tell: A Neural Image Caption Generator. *arXiv:1411.4555 [cs]*.
Von Mering,C. *et al.* (2003) STRING: a database of predicted functional associations between proteins. *Nucleic Acids Res*, **31**, 258–261.
Wahl,M.C. *et al.* (2009) The Spliceosome: Design Principles of a Dynamic RNP Machine. *Cell*, **136**, 701–718.

Wan,R. *et al.* (2019) Molecular choreography of pre-mRNA splicing by the spliceosome. *Current Opinion in Structural Biology*, **59**, 124–133.
Wang,K. *et al.* (2010) MapSplice: Accurate mapping of RNA-seq reads for splice junction discovery. *Nucleic Acids Research*, **38**, e178–e178.
Wang,L. *et al.* (2020) A State-of-the-Art Review on Image Synthesis With Generative Adversarial Networks. *IEEE Access*, **8**, 63514–63537.
Wang,L. and Jiang,T. (1994) On the Complexity of Multiple Sequence Alignment. *Journal of Computational Biology*, **1**, 337–348.
Wang,R. *et al.* (2019) SpliceFinder: ab initio prediction of splice sites using convolutional neural network. *BMC Bioinformatics*, **20**, 652.
Wang,S.X.J. and Lichodzijewski,P. (2005) Boolean genetic programming for promoter recognition in eukaryotes., pp. 683-690 Vol.1.
Wang,Z. and Burge,C.B. (2008) Splicing regulation: From a parts list of regulatory elements to an integrated splicing code. *RNA*, **14**, 802–813.
Warf,M.B. and Berglund,J.A. (2010) The role of RNA structure in regulating pre-mRNA splicing. *Trends Biochem Sci*, **35**, 169–178.
Waterfall,J.J. *et al.* (2014) High prevalence of MAP2K1 mutations in variant and IGHV4-34 expressing hairy-cell leukemia. *Nat Genet*, **46**, 8–10.
Watkins,C.J.C.H. and Dayan,P. (1992) Q-learning. *Mach Learn*, **8**, 279–292.
Watson,J.D. and Crick,F.H.C. (1953) Molecular Structure of Nucleic Acids: A Structure for Deoxyribose Nucleic Acid. *Nature*, **171**, 737–738.
Wei,D. *et al.* (2013) A Novel Splice Site Prediction Method using Support Vector Machine ⋆.
Weizenbaum,J. (1966) ELIZA: a computer program for the study of natural language communication between man and machine. *Commun. ACM*, **9**, 36–45.
Werbos,P.J. (1994) The Roots of Backpropagation: From Ordered Derivatives to Neural Networks and Political Forecasting John Wiley & Sons.
Westoby,J. *et al.* (2020) Obstacles to detecting isoforms using full-length scRNA-seq data. *Genome Biology*, **21**, 74.
Wilkinson,M.E. *et al.* (2020) RNA Splicing by the Spliceosome. *Annu. Rev. Biochem.*, **89**, 359–388.
Will,C.L. and Lührmann,R. (2011) Spliceosome Structure and Function. *Cold Spring Harb Perspect Biol*, **3**, a003707.
Wingate,R. and Kwint,M. (2006) Imagining the brain cell: the neuron in visual culture. *Nat Rev Neurosci*, **7**, 745–752.
Wold,S. *et al.* (1987) Principal component analysis. *Chemometrics and Intelligent Laboratory Systems*, **2**, 37–52.
Wu,J. *et al.* (2016) Big Data Meet Green Challenges: Big Data Toward Green Applications. *IEEE Systems Journal*, **10**, 888–900.
Wu,J. *et al.* (2010) Cloud Storage as the Infrastructure of Cloud Computing. *Intelligent Computing and Cognitive Informatics, International Conference on*, **0**, 380–383.
Wu,P.-W. *et al.* (2019) RelGAN: Multi-Domain Image-to-Image Translation via Relative Attributes. *arXiv:1908.07269 [cs]*.
Yandell,M. and Ence,D. (2012) A beginner's guide to eukaryotic genome annotation. *Nature Reviews Genetics*, **13**, 329–342.
Yanling,Z. *et al.* (2002) Analysis and study of perceptron to solve XOR problem. In, *The 2nd International Workshop on Autonomous Decentralized System, 2002.*, pp. 168–173.
Yean,S.-L. *et al.* (2000) Metal-ion coordination by U6 small nuclear RNA contributes to catalysis in the spliceosome. *Nature*, **408**, 881–884.
Yeo,G. and Burge,C.B. (2004) Maximum Entropy Modeling of Short Sequence Motifs with Applications to RNA Splicing Signals. *Journal of Computational Biology*, **11**, 377–394.
Yip,K.Y. *et al.* (2013) Machine learning and genome annotation: a match meant to be? *Genome Biology*, **14**, 205.
Yıldırım,Ö. *et al.* (2018) Arrhythmia detection using deep convolutional neural network with long duration ECG signals. *Computers in Biology and Medicine*, **102**, 411–420.
Yoshida,K. and Ogawa,S. (2014) Splicing factor mutations and cancer. *Wiley interdisciplinary reviews. RNA*, **5**, 445–459.

Zhan, X, *et al.* (2018) Structures of the human pre-catalytic spliceosome and its precursor spliceosome. *Cell Res*, **28**, 1129-1140.
Zhang,D. *et al.* (2020) Incomplete annotation has a disproportionate impact on our understanding of Mendelian and complex neurogenetic disorders. *Science Advances*, **6**, eaay8299.
Zhang,M.Q. (1998) Statistical Features of Human Exons and Their Flanking Regions. *Human Molecular Genetics*, **7**, 919–932.
Zhang,Quanwei *et al.* (2010) Splice sites prediction of Human genome using length-variable Markov model and feature selection. *Expert Systems with Applications*, **37**, 2771–2782.
Zhang,X. *et al.* (2017) An Atomic Structure of the Human Spliceosome. *Cell*, **169**, 918-929.e14.
Zhang,X. and LeCun,Y. (2017) Which Encoding is the Best for Text Classification in Chinese, English, Japanese and Korean? *arXiv:1708.02657 [cs]*.
Zhu,X. and Goldberg,A.B. (2009) Introduction to Semi-Supervised Learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, **3**, 1–130.
Zhuang,F. *et al.* (2020) A Comprehensive Survey on Transfer Learning. *arXiv:1911.02685 [cs, stat]*.
Zimmerly,S. and Semper,C. (2015) Evolution of group II introns. *Mob DNA*, **6**, 7.
Zuallaert,J. *et al.* (2018) SpliceRover: interpretable convolutional neural networks for improved splice site prediction. *Bioinformatics*, **34**, 4180–4188.

Université de Strasbourg

Nicolas SCALZITTI

École Doctorale des Sciences de la Vie et de la Santé STRASBOURG

Nouvelle stratégie d'annotation des génomes par l'utilisation d'algorithmes d'intelligence artificielle

Résumé :

Les projets de séquençage à haut débit produisent quotidiennement une énorme quantité de données biologiques brutes. Cependant, ces données sont difficilement exploitables si elles ne sont pas annotées. Afin de pouvoir traiter ces données massives, des programmes d'annotation de génomes ont été développés, malheureusement ces derniers sont encore trop sujet aux erreurs de prédiction, faisant de l'annotation des génomes un des défis majeurs en bio-informatique. Dans ce contexte, mes travaux de thèse s'organisent autour d'un trinôme indissociable : i) l'amélioration de la prédiction des gènes eucaryotes codant pour des protéines en se focalisant spécifiquement sur les sites d'épissage ii) en exploitant des algorithmes d'intelligence artificielle (réseaux de neurones convolutif et algorithmes évolutionnaires), iii) entraînés avec des données de haute qualité incluant une forte hétérogénéité d'organismes eucaryotes (de l'Homme aux protistes). La stratégie mise au point consiste à combiner l'ensemble des données traitées et validées (G3PO) avec les programmes développés (Spliceator et SpliceSLEIA) afin d'améliorer la prédiction des gènes en diminuant le taux d'erreurs afin qu'elles ne se propagent plus dans les bases de données publiques. De plus, ces travaux permettront une meilleure compréhension des organismes et de leurs mécanismes biologiques.

Mots-clés : Intelligence artificielle, génomique, sites d'épissage, évolution artificielle, deep learning, programmation génétique, annotation de génome

Summary:

High-throughput genome sequencing projects produce a huge amount of raw biological data on a daily basis. However, the data are difficult to exploit if they are not annotated. Unfortunately, the currently available genome annotation programs are still too prone to prediction errors, making genome annotation one of the major challenges in bioinformatics. In this context, my thesis is organized around three connected topics: i) improving the prediction of eukaryotic protein-coding genes by focusing on splice sites ii) exploiting artificial intelligence algorithms (convolutional neural networks and evolutionary algorithms), iii) training with high quality data from a wide range of eukaryotic organisms (from humans to protists). The strategy developed consists in combining the processed and validated data set (G3PO) with the developed programs (Spliceator and SpliceSLEIA) in order to improve gene prediction by decreasing the error rate so that they no longer propagated in public databases. This work should contribute to a better understanding of organisms and their biological mechanisms.

Keywords: Artificial intelligence, genomics, splice site, artificial evolution, deep learning, genetic programming, genome annotation