HAL
open science

# Proposer un workflow généraliste pour identifier les interactions protéine-protéine entre deux tissus. Exemple des PPIs entre le sécrétome et le surfaceome pour comprendre le dialogue entre tissus chez les mammifères et en particulier chez le bovin

Manon Connault, Nadia Goué, Muriel Bonnet

## To cite this version:

Manon Connault, Nadia Goué, Muriel Bonnet. Proposer un workflow généraliste pour identifier les interactions protéine-protéine entre deux tissus. Exemple des PPIs entre le sécrétome et le surfaceome pour comprendre le dialogue entre tissus chez les mammifères et en particulier chez le bovin. Bio-Informatique, Biologie Systémique [q-bio.QM]. 2021. hal-03332182

**HAL Id: hal-03332182**

**https://hal.inrae.fr/hal-03332182**

Submitted on 2 Sep 2021

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

**Master 2**

**Bio-informatique**

Parcours
Analyse et Modélisation des Données

*RAPPORT DE STAGE PRÉSENTÉ PAR :*

Manon CONNAULT

*SUJET :*

# Proposer un workflow généraliste pour identifier les interactions protéine-protéine entre deux tissus. Exemple des PPIs entre le sécrétome et le surfaceome pour comprendre le dialogue entre tissus chez les mammifères et en particulier chez le bovin

Responsable du Stage :

**Nadia GOUÉ**

**Université Clermont-Auvergne**
**DOSI – Mésocentre**
**7 avenue Blaise Pascal**
**63178 AUBIÈRE, FRANCE**

Co-encadrante :

**Muriel BONNET**

**INRAE – UMR1213 Herbivores Theix**
**Équipe Biomarqueurs**
**63122 SAINT-GENÈS-CHAMPANELLE, FRANCE**

**Juin 2021**

**Résumé du travail**

La prédiction des interactions protéine-protéine est étudiée dans différents domaines. Notamment, la connaissance des interactions entre le surfaceome et le sécrétome contribuerait à une meilleure compréhension du dialogue inter-tissus. Ce stage a pour objectif de développer un workflow en Snakemake, appelé Talkmine, pour identifier le dialogue moléculaire entre deux tissus biologiques résultant d'interactions protéine-protéine.

Le premier objectif a été d'identifier puis de tester des outils opensources pour prédire l'appartenance des protéines au sécrétome ou au surfaceome. Ils ont été classés selon trois catégories, la prédiction du peptide signal, de la localisation subcellulaire ou de la topologie. Dans un deuxième temps, le workflow a été développé. L'utilisateur donne en entrée une liste d'identifiants de gènes ou de protéines. L'outil g:Convert convertit les identifiants au format ENSP puis l'outil Entrez-Direct récupère les séquences dans un fichier multi-fasta. Les séquences protéiques sont injectées dans les outils de prédiction. Enfin, les protéines associées aux classes *Sécrétome* et *Surfaceome* sont envoyées dans l'outil PSICQUIC, pour identifier les interactions protéine-protéine. L'utilisateur a accès à la liste des interactions ainsi qu'aux résultats intermédiaires.

Le workflow Talkmine a été développé pour être appliqué à *Bos taurus*, afin de déterminer les interactions entre le muscle et le tissu adipeux, et il est applicable à d'autres espèces et domaines de recherche.

The prediction of the protein-protein interactions is studied in different domains. In particular, the interactions between surfaceome and secretome may strongly improved the understanding of inter-tissue crosstalk. This intership aims to develop a Snakemake-based workflow named

Talkmine for the identification of the molecular dialogue between two biological tissues resulting from protein-protein interactions.

The first objective was to identify and test some opensource tools known to predict proteins that belong to secretome or surfaceome. Publically available tools were classified according to three classes, i.e. peptide signal, subcellular location or topology prediction. Secondly, we developed the workflow. In brief, user gives a gene or protein identifiers list. g:Convert tool converts identifiers to a same format and then, Entrez-Direct tool generates a multi-fasta file. The protein sequences are then launched to the predictive tools. Finally, proteins tagged to secretome and surfaceome classes that are sent to PSICQUIC tool, which determines the protein-protein interactions. The user has access to the list of these interactions as well as to the intermediate results.

The workflow Talkmine was initiated to be applied to *Bos taurus* to determine the interactions between muscle and fat tissue, and it could also be applied to other species and research purposes.

**Remerciements**

Avant toute chose, je tiens à remercier Monsieur Laurent LEMOINE, directeur opérationnel de la DOSI, ainsi que Monsieur Antoine MAHUL, responsable de l'équipe de rattachement au Mésocentre, qui m'ont permis de réaliser mon stage au sein de la structure du Mésocentre et de la DOSI.

Je remercie ma maître de stage Madame Nadia GOUÉ (Mésocentre), qui a été présente et qui s'est rendue disponible malgré la distance et les contraintes. Nos échanges m'ont permis d'avancer dans ma vision du développement d'outils bio-informatiques, mes réflexions, mes travaux et la rédaction de ce rapport.

Je remercie ma co-encadrante de stage Madame Muriel BONNET (INRAE), qui s'est rendue présente le plus possible malgré les contraintes de la distance et des emplois du temps. Ses retours m'ont permis de progresser dans mes recherches et ma compréhension des thématiques biologiques de ce sujet de stage et dans la rédaction de ce rapport.

Je remercie Monsieur Jérémy TOURNAYRE (INRAE), qui m'a apporté un soutien bioinformatique pour la recherche d'outils et l'élaboration du jeu de données test.

Je remercie Madame Céline BOBY (INRAE), qui m'a permis de développer l'aspect pédagogique de ce sujet de stage et aidé dans la rédaction de ce rapport.

Enfin, je remercie l'équipe Système du pôle Infrastructures de la DOSI, qui m'a permis de participer aux réunions en visioconférence.

**Préambule : Présentation de la structure d'accueil**

La direction opérationnelle des systèmes d'information (DOSI)[1] est une infrastructure de l'Université Clermont-Auvergne (UCA) visant à harmoniser les services aux usagers, améliorer les processus métiers et porter les projets numériques au sein de l'Université. Il est divisé en trois pôles : le pôle Système d'Information, le pôle Infrastructures et le pôle Proximité. Le pôle Infrastructures gère les infrastructures numériques pour l'ensemble des structures de l'Université ainsi que les établissements partenaires. Il administre également le Datacenter universitaire et il s'occupe des infrastructures du Système d'Information et des services numériques de l'UCA, tels que l'ENT, la messagerie Zimbra, etc… Enfin, un rattachement au Mésocentre Clermont-Auvergne fournit des ressources informatiques pour la recherche.

Le Mésocentre[2,3] est une infrastructure qui mutualise les ressources humaines, les équipements de calcul, de traitement et de stockage de données ainsi que les applications scientifiques. Toutes ces ressources sont disponibles sur demande aux différents membres des laboratoires auvergnats et comptabilise un cluster de calcul, un calculateur à mémoire partagée, une plateforme de stockage distribuée et une plateforme cloud nommée OSCAR. En plus de cette mutualisation des ressources et le développement d'une expertise autour du calcul scientifique, le Mésocentre a pour mission d'accompagner les utilisateurs et de favoriser les échanges et les collaborations entre les différentes équipes de recherche mais aussi avec les mésocentres en région et à l'échelle nationale.

1 Site de la DOSI : **https://dsi.uca.fr/**
2 Site du Mésocentre : **https://mesocentre.uca.fr/**
3 Présentation du Mésocentre sur le site de l'Université Clermont-Auvergne : **https://www.uca.fr/recherche/plateaux-techniques/auvergrid**

**Liste des abréviations**

ABC : ATP Binding Cassette

API : Application Programming Interface / Interface de Programmation

AP-MS : Purification par Affinité couplée à la Spectrométrie de Masse

AuBi : plateforme Auvergne Bioinformatique

BiLSTM : Réseau de Neurones Récurrents Bidirectionnel à Mémoire Court et Long terme

BiRNN : Réseau de Neurones Récurrents Bidirectionnel

CNN : Réseau de neurones convolutifs

COPI : Complexe Protéique I

COPII : Complexe Protéique II

CRF : Champ Aléatoire Conditionnel

CUPS : Compartiment de Sécrétion de Protéines Non-conventionnelle

DOSI : Direction Opérationnelle des Systèmes d'Information

DNN : Réseau de Neurones Profond

Edirect : Entrez Direct

ENSP : ENSembl Peptide ID

ER : Réticulum Endoplasmique

E-utilities : Entrez Programming Utilities

FFN : Réseau de neurones de rétroaction

GPI : Glycosylphosphatidylinositol

GRU : Unité Récurrente Fermée

HAPPI-2 : Human Annotated and Predicted Protein Interactions

HPC2 : High Performance Computing 2

IMEx : International Molecular Exchange

JSON : JavaScript Object Notation

LSTM : Mémoire Court et Long terme

MIQL : Molecular Interaction Query Language

MVB : Corps Multivésiculaire

NCBI : National Center for Biotechnology Information

PM : Membrane Plasmique

PPI : Interaction Protéine-Protéine

PSI : Proteomics Standard Initiative

PSICQUIC : PSI Common QUery InterfaCe

RNN : Réseau de neurones récurrents

SLURM : Simple Linux Utility for Resource Management

SQL : Structured Query Language

SRP : Protéine de reconnaissance du Signal

STRING : Search Tool for the Retrieval of Interacting Genes/Proteins

TGN : Réseau Trans-Golgien

TM : Transmembranaire

UCA : Université Clermont-Auvergne

Y2H : Double Hybride

**Liste des figures**

Figure 1 | Schéma des voies de sécrétion conventionnelle et non conventionnelle.

Figure 2 | Schéma de la synthèse des protéines du surfaceome dans les cellules épithéliales.

Figure 3 | Schéma de l'élaboration d'un outil de prédiction.

Figure 4 | Algorigramme de l'étude de Bonnet et al. (2020).

Figure 5 | Répartition du jeu de données test au sein des compartiments subcellulaires.

Figure 6 | Algorigramme du script Python g:Convert.

Figure 7 | Ligne de commande des outils permettant de prédire la présence d'un peptide signal et la localisation subcellulaire.

Figure 8 | Algorigramme du workflow Talkmine.

**Liste des tableaux**

Tableau 1 | Exemples de protéines empruntant la voie de sécrétion non conventionnelle.

Tableau 2 | Motifs reconnus par les différents outils de prédiction au sein des séquences protéiques.

Tableau 3 | Termes utilisés sous UniProt pour répertorier les protéines d'intérêt.

Tableau 4 | Ensemble des organismes pris en charge par les différents outils de prédiction sélectionnés.

**Liste des annexes**

Figure Annexe 1 | Schéma de deux techniques à haut débit permettant l'identification des PPIs.

Figure Annexe 2.1.a | Schéma d'un réseau neuronal convolutif (Min et al. 2017, traduite).

Figure Annexe 2.1.b | Schéma d'un réseau neuronal récurent (Min et al. 2017, traduite).

Figure Annexe 2.1.c | Schéma d'un réseau neuronal récurent bidirectionnel (Min et al. 2017, traduite).

Figure Annexe 2.2 | Architecture de l'outil SignalP 5.0 (Almagro Armenteros, Tsirigos, et al. 2019, traduite).

Figure Annexe 2.3 | Architecture de l'outil TargetP 2.0 (Almagro Armenteros, Salvatore, et al. 2019, traduite).

Figure Annexe 2.4 | Architecture de l'outil DeepLoc (Almagro Armenteros et al. 2017, traduite).

Tableau Annexe 3 | Informations sur le jeu de données test.

Tableau Annexe 7.1 | Outils de prédiction du peptide signal répertoriés.

Tableau Annexe 7.2 | Outils de prédiction de la localisation subcellulaire répertoriés.

Tableau Annexe 7.3 | Outils de prédiction de la topologie répertoriés.

**Sommaire**

Résumé du travail .....

Remerciements .....

Préambule : Présentation de la structure d'accueil .....

Liste des abréviations .....

Liste des figures .....

Liste des tableaux .....

Liste des annexes .....

I. Bibliographie ..... 1

I. I. Contexte scientifique ..... 1

I. II. Les protéines du sécrétome et du surfaceome et les interactions protéine-protéine …. 2

I. II. I. Protéines du sécrétome : voie conventionnelle versus voie non conventionnelle 2

I. II. II. Protéines du surfaceome : deux classes localisées dans la membrane plasmique . 4

I. II. III. Interactions protéine-protéine ..... 6

I. III. Prédiction bioinformatique de la localisation subcellulaire et des interactions protéine-protéine ..... 7

I. III I. Prédiction du peptide signal et de la localisation subcellulaire ..... 7

I. III II. Bases de données répertoriant les interactions protéine-protéine ..... 9

I. IV. Objectifs du stage ..... 9

II. Réalisations ..... 11

II. I. Matériel et méthodes ..... 11

II. I. I. Jeu de données test ..... 11

II. I. II. Élaboration du workflow ..... 11

II. I. II. I. Environnement de développement informatique ..... 11

II. I. II. II. Préparation des données : g:Convert de g:Profiler ..... 13

II. I. II. III. Extraction des séquences protéiques : Entrez Direct ..... 15

II. I. II. IV. Sélection des outils de prédiction ..... 16

II. I. II. V. Récupération des données sur les interactions protéine-protéine : PSICQUIC ..... 18

II. II. Résultats ..... 19

II. II. I. Développement d'un workflow en Snakemake pour déterminer les interactions protéine-protéine ..... 19

II. II. II. Analyse du jeu de données ..... 20

II. II. II. I. Correction du jeu de données de test ..... 20

II. II. II. II. Étape 01 – Conversion des identifiants : g:Convert ..... 20

II. II. II. III. Étape 02 – Récupération des séquences protéiques : Entrez-Direct ..... 21

II. II. II. IV. Étape 03 – Prédiction du peptide signal (SignalP et TargetP), de la localisation subcellulaire (DeepLoc) et extraction des données ..... 21

II. II. II. V. Prédiction des interactions protéine-protéine : PSICQUIC ..... 21

III. Discussion ..... 22

Bibliographie .....

Annexes .....

Annexe 1 : Méthodes à haut débit pour la mise en évidence d'une interaction entre deux protéines .....

Annexe 2 : Apprentissage profond (ou *deep learning*) en Bioinformatique – Application à la prédiction et architectures .....

Annexe 2.1 : Généralités sur les architectures de *deep learning* .....

Annexe 2.2 : Architecture de SignalP 5.0 .....

Annexe 2.3 : Architecture de TargetP 2.0 .....

Annexe 2.4 : Architecture de DeepLoc 1.0 .....

Annexe 3 : Jeu de données test .....

Annexe 4 : Extrait du fichier *ReadMe.md* de la branche dev du dépôt Github .....

Annexe 5 : Script Python *gConvert.py* .....

Annexe 6 : Script Bash *EntreDirect.sh* .....

Annexe 7 : Tableau récapitulant les outils répertoriés .....

Annexe 7.1 : Outils de prédiction du peptide signal .....

Annexe 7.2 : Outils de prédiction de la localisation subcellulaire .....

Annexe 7.3 : Outils de prédiction de la topologie .....

Annexe 8 : Résumé du poster soumis pour l'évènement JOBIM 2021 (https://jobim2021.sciencesconf.org/, référence : 59) .....

# I. Bibliographie

## I. I. Contexte scientifique

L'UMR 1213 Herbivores de INRAE travaille sur des thématiques visant à élaborer un élevage d'herbivores économiquement et environnementalement performant, qui valorise des fourrages et d'autres ressources non compétitives avec l'alimentation humaine, tout en répondant aux différentes attentes sociétales[4]. Au sein de cette structure, l'équipe BIOMARQUEURS a notamment pour but de comprendre le dialogue inter-organes chez les ruminants producteurs de viande ou de lait élevés dans un contexte de diversification des pratiques d'élevage, tel que les environnements changeants, la diminution des intrants en compétition avec l'alimentation humaine et la transition vers l'agroécologie.

La masse de tissu adipeux par rapport à la masse du tissu musculaire est l'un des déterminants des performances de production et des capacités d'adaptation des bovins à des environnements changeants. Les connaissances actuelles suggèrent une forme de compétition entre les deux tissus (Bonnet et al. 2016) : une forte croissance du tissu musculaire provoque un ralentissement de celle du tissu adipeux et inversement. Cependant, les mécanismes impliqués n'ont pas été clairement définis. Ainsi, l'équipe BIOMARQUEURS a établi une première liste des protéines sécrétées par le tissu musculaire interagissant avec les protéines de surface du tissu adipeux et *vice-versa* (Bonnet et al. 2020).

4 UMR 1213 Herbivores : **https://umrh-bioinfo.clermont.inrae.fr/Intranet/web/UMRH**

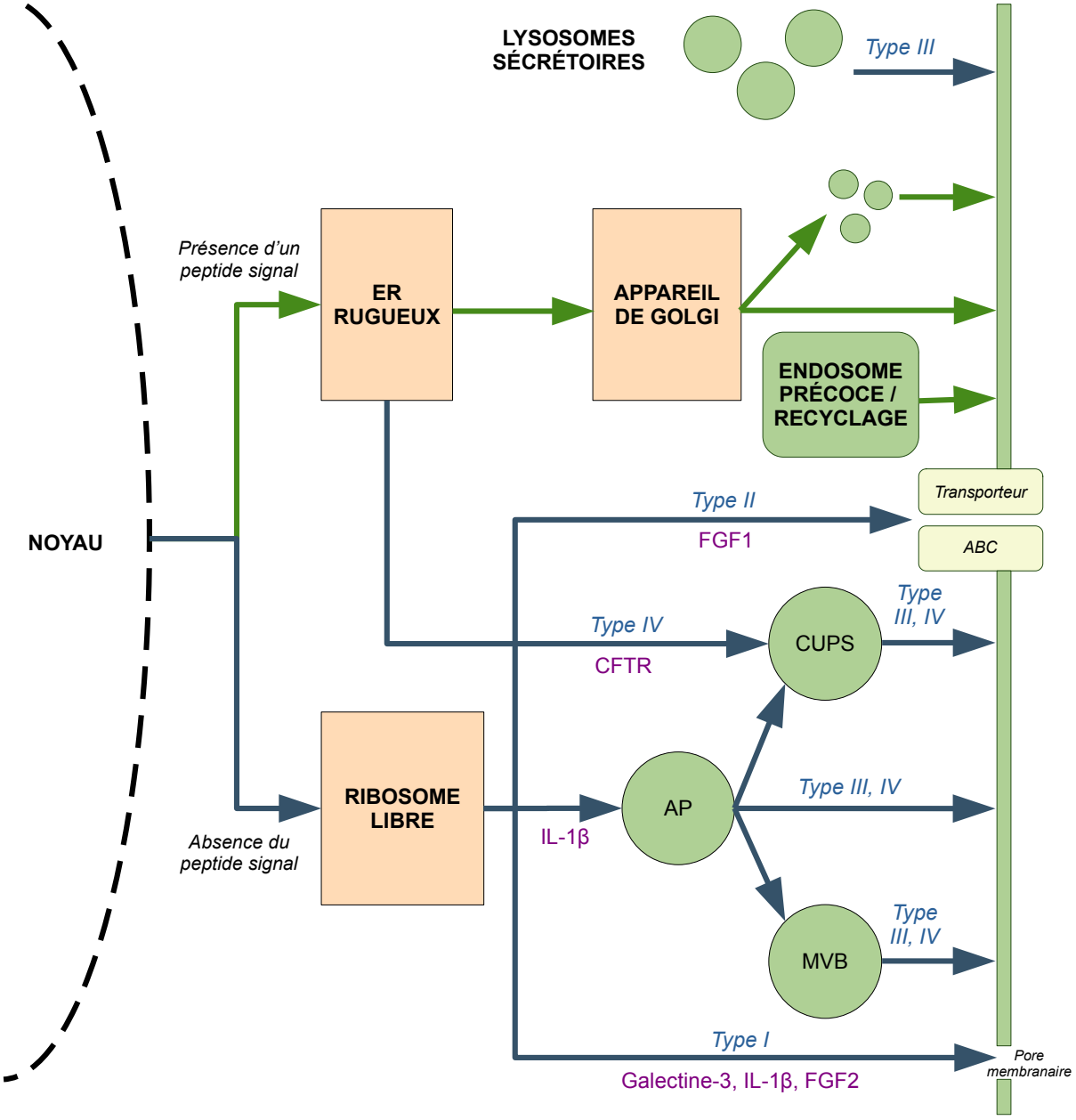


**Figure 1 | Schéma des voies de sécrétion conventionnelle et non conventionnelle.** La voie de sécrétion conventionnelle, empruntée par les protéines présentant un peptide signal, est représentée par les flèches vertes tandis que la voie non-conventionnelle, empruntée par les protéines ne présentant aucun peptide signal, est représentée par les flèches bleues (Transporteur ABC, Cassette Liant l'ATP). Les vésicules sont en vert (AP, Autophagosome ; CUPS, Compartiment de Sécrétion de Protéines Non-conventionnelle ; MVB, Corps Multivésiculaires) et les organites en orange (ER, Réticulum Endoplasmique). Pour les voies non-conventionnelles, les types sont indiqués en bleu et en italique. Des exemples de protéines empruntant ces voies sont indiqués en violet (CFTR, Régulateur de la conductance Transmembranaire de la Fibrose Kystique ; FGF1, Facteur de Croissance des Fibroblastes 1 ; FGF2, Facteur de Croissance des Fibroblastes 2 ; IL1-β, Interleukine-1 β).

## I. II. Les protéines du sécrétome et du surfaceome et les interactions protéine-protéine

### I. II. I. Protéines du sécrétome : voie conventionnelle versus voie non conventionnelle

Le sécrétome constitue l'ensemble des protéines sécrétées par les cellules, tissus ou organes vers les espaces extracellulaires incluant la matrice extracellulaire ainsi que des vésicules (exosomes et vésicules microsomales) (Caccia et al. 2013). Ces protéines jouent un rôle important notamment dans l'homéostasie métabolique, le développement, l'organisation de la matrice extracellulaire ou la réponse immunitaire (Caccia et al. 2013, Gee et al. 2018) Ainsi, le sécrétome est hautement dynamique et sa composition peut être influencée par des stimuli environnementaux, physiologiques ou pathologiques (Caccia et al. 2013).

La majorité des protéines sécrétées empruntent la voie de sécrétion conventionnelle (Figure 1) (Cohen et al. 2020, Ng and Tang 2016). Cette voie implique le passage de la protéine au travers des différents organites : le réticulum endoplasmique (ER), l'appareil de Golgi puis le réseau trans-golgien (TGN) (Ng and Tang 2016). Une fois internalisée dans une vésicule (Nickel 2003), elle est ensuite amenée au niveau de la membrane plasmique (PM) pour être excrétée dans la matrice extracellulaire ou intégrée dans d'autres organelles (peroxysome, lysosome, endosome) (Gee et al. 2018). Les protéines empruntant cette voie de sécrétion conventionnelle ont la particularité de présenter au niveau de leur extrémité N-terminale une séquence de 9 à 12 acide aminés hydrophobes (Whitley and Mingarro 2014), appelée le peptide signal (Cohen et al. 2020, Gee et al. 2018, Ng and Tang 2016, Nickel 2003). Cette séquence est reconnue par un complexe ribonucléoprotéique, la particule de reconnaissance du signal (SRP), qui induit l'adressage du complexe ribosome-chaîne protéique naissante à la membrane du ER. Les chaînes polypeptidiques nouvellement traduites traversent la lumière

du ER au travers d'un canal, appelé un translocon (Cohen et al. 2020, Whitley and Mingarro 2014). Une fois dans le ER, certaines protéines subissent des modifications post-traductionnelles, telles que la N-glycolysation (Ng and Tang 2016). La protéine est ensuite amenée au niveau des sites de sortie du ER, pour être internalisée dans des vésicules enrobées de complexe protéique II (COPII) (Gee et al. 2018, Ng and Tang 2016). Ces vésicules acheminent la protéine jusqu'à l'appareil de Golgi puis le TGN. Durant cette étape, d'autres modifications peuvent être apportées au niveau de l'extrémité N-terminale. Enfin, la protéine est internalisée dans une vésicule enrobée d'un complexe protéique I (COPI) (Ng and Tang 2016). Ces vésicules sont amenées directement à la PM, via la voie de sécrétion constitutive, ou restent dans le cytosol jusqu'à la perception d'un signal de sécrétion, via la voie de sécrétion régulée (Cohen et al. 2020).

Il existe également des protéines qui ne présentent aucun peptide signal et qui sont sécrétées par des voies non conventionnelles (Cohen et al. 2020, Dimou and Nickel 2018, Gee et al. 2018, Ng and Tang 2016, Nickel 2003). Quatre voies de sécrétion non conventionnelles ont pour le moment été répertoriées, dont trois concernent spécifiquement les protéines sécrétées dans l'espace extracellulaire (Figure 1) (Dimou and Nickel 2018, Gee et al. 2018) :

- La voie de type I, dans laquelle les protéines passent à travers un pore de la PM
- La voie de type II, dans laquelle les protéines sont excrétées dans le milieu extracellulaire via un transporteur ABC, qui est une protéine transmembranaire transportant activement des molécules en hydrolysant une molécule d'ATP
- La voie de type III, dans laquelle les protéines sont excrétées via un organite d'endocytose ou lié à la membrane (exemple, un corps multivésiculaire (MVB) ou un compartiment de sécrétion de protéines non-conventionnelles (CUPS))

| Protéine | Classe / Espèce | Voie non conventionnelle associée (Type) | Références |
|---|---|---|---|
| α-Synuclein | *Homo sapiens* | Endosomes sécrétoires et lysosomes (III) | Kim et al 2018, Ng and Tang 2016 |
| Annexin A1 | Mammifères | Translocation au travers de la membrane (I) | Cohen et al 2020 |
| Annexin A2 | Mammifères | Translocation au travers de la membrane (I) | Cohen et al 2020 |
| Annexin A5 | Mammifères | Translocation au travers de la membrane (I) | Cohen et al 2020 |
| CD45 (*receptor-type tyrosine-protein phosphatase C*) | *Homo sapiens* | Sécrétion transmembranaire indépendante du Golgi (IV) | Gee et al 2018 |
| (ΔF508) CFTR (*cystic fibrosis transmembrane conductance regulator*) | *Homo sapiens* | Sécrétion transmembranaire indépendante du Golgi (IV) | Dimou and Nickel 2018, Gee et al 2018, Kim et al 2018 |
| Connexin 26 | *Homo sapiens* | Sécrétion transmembranaire indépendante du Golgi (IV) | Gee et al 2018 |
| Connexin 30 | *Homo sapiens* | Sécrétion transmembranaire indépendante du Golgi (IV) | Gee et al 2018 |
| DKK3 (*Dickkopf-related protein 3*) | *Homo sapiens* | Autophagosome (III) | Kim et al 2018 |
| Enolase | Mammifères | Système endolysosomal (III) | Cohen et al 2020 |
| FABP4 (*fatty acid-binding protein*) | Mammifères | Endosomes sécrétoires et lysosomes (III) | Cohen et al 2020 |
| FADD (*fas Associated Death Domain*) | Mammifères | Microvésicules (III) | Cohen et al 2020 |
| FAM3C (*family with sequence similarity 3 member C*) | *Homo sapiens* | Autophagosome (III) | Kim et al 2018 |
| FGF1 (*fibroblast growth factor 1*) | Mammifères | Translocation au travers de la membrane (I) | Cohen et al 2020, Dimou and Nickel 2018 |
| FGF2 (*fibroblast growth factor 2*) | Mammifères | Translocation au travers de la membrane (I) | Cohen et al 2020, Dimou and Nickel 2018 |
| Galectin-1 | *Cricetulus griseus*<br>*Homo sapiens* | Translocation au travers de la membrane (I)<br>Endosomes sécrétoires et lysosomes (III) | Popa et al 2018 |
| Galectin-3 | *Homo sapiens* | Translocation au travers de la membrane (I) | Kim et al 2018, Popa et al 2018 |
| γ-Synuclein | *Homo sapiens* | Endosomes sécrétoires et lysosomes (III) | Ng and Tang 2016 |
| HMGB1 (*high-mobility group box 1*) | Mammifères | Système endolysomal (III) | Cohen et al 2020, Kim et al 2018 |
| Hsp70 (*heat shock protein 70*) | Mammifères | Système endolysomal (III) | Cohen et al 2020 |
| Hsc70 (*heat shock cognate 70*) | Mammifères | Système endolysomal (III) | Cohen et al 2020 |
| Hsp90 (*heat shock protein 90*) | Mammifères | Vésicules (III)<br>Exomères (III) | Cohen et al 2020 |
| IDE (*insulin-degrading enzyme*) | *Homo sapiens* | Système endolysomal (III) | Kim et al 2018 |
| IL1-β (*interleukin-1 β*) | Mammifères | Translocation au travers de la membrane (I)<br>Autophagosome (III)<br>Amphisome fusionné à la membrane plasmique (III) | Cohen et al 2020, Dimou and Nickel 2018 |
| LIF (*leukemia inhibitory factor*) | *Homo sapiens* | Autophagosome (III) | Kim et al 2018 |
| MPL (*myeloproliferative leukemia virus oncogene*) | *Homo sapiens* | Sécrétion transmembranaire indépendante du Golgi (IV) | Gee et al 2018 |
| Pannexin 1 | *Homo sapiens* | Sécrétion transmembranaire indépendante du Golgi (IV) | Gee et al 2018 |
| Pannexin 3 | *Homo sapiens* | Sécrétion transmembranaire indépendante du Golgi (IV) | Gee et al 2018 |
| Pendrin (GRASP-indépendante) | *Homo sapiens* | Sécrétion transmembranaire indépendante du Golgi (IV) | Gee et al 2018, Kim et al 2018 |
| Peripherin-2 (RDS, *retinal degeneration slow protein*) | *Homo sapiens* | Sécrétion transmembranaire indépendante du Golgi (IV) | Gee et al 2018, Kim et al 2018 |
| Phospholipid scramblase1 | *Homo sapiens* | Sécrétion transmembranaire indépendante du Golgi (IV) | Gee et al 2018 |
| Polycystin-2 | *Homo sapiens* | Sécrétion transmembranaire indépendante du Golgi (IV) | Gee et al 2018, Kim et al 2018 |
| Smoothened (M2) | *Homo sapiens* | Sécrétion transmembranaire indépendante du Golgi (IV) | Gee et al 2018 |
| Tau | *Homo sapiens* | Translocation au travers de la membrane (I)<br>Endosomes sécrétoires et lysosomes (III) | Dimou and Nickel 2018 |
| TGM2 (*protein-glutamine gamma-glutamyltransferase 2*) | *Homo sapiens*<br>*Mus musculus* | Translocation au travers de la membrane (I)<br>Endosomes sécrétoires et lysosomes (III) | Dimou and Nickel 2018 |

**Tableau 1 | Exemples de protéines empruntant la voie de sécrétion non conventionnelle.** Pour la colonne « *Protéine* », le nom complet des protéines est indiqué entre parenthèses en italique tandis que les informations complémentaires sont simplement entre parenthèses.

- La voie de type IV, qui concerne les protéines membranaires. Elle sera abordée dans le paragraphe suivant.

Différentes protéines sont connues pour emprunter une ou plusieurs de ces voies (Tableau 1). Un grand nombre de ces protéines présentent des fonctions extracellulaires et une sécrétion contrôlée (Cohen et al. 2020).

### I. II. II. Protéines du surfaceome : deux classes localisées dans la membrane plasmique

Le surfaceome est défini comme l'ensemble des protéines de la PM et qui ont au moins un acide aminé exposé du côté extracellulaire. Les protéines sont réparties en deux classes : les protéines rattachées à la bicouche lipidique via une ancre lipidique et les protéines contenant un domaine transmembranaire (TM) (Bausch-Fluck et al. 2018).

Les protéines à ancre glycosylphosphatidylinositol (GPI) est l'une des catégories de protéines à ancrage lipidique, dont le lipide sert à ancrer la protéine dans le feuillet externe de la PM. Le complexe GPI est lié à l'extrémité C-terminale de la protéine, par le complexe multienzymatique GPI-transamidase lors de la modification post-traductionnelle (Lebreton et al. 2018, Muniz and Zurzolo 2014). Ces protéines représentent 0,5 % des protéines totales chez les eucaryotes et plus de 150 protéines à ancre GPI ont été identifiées chez les mammifères (Lebreton et al. 2018).

Les protéines TM sont classées en deux catégories structurelles : les protéines avec un domaine TM sous la forme d'hélices α et les protéines dont la région TM est composée de feuillets β regroupés en tonneaux anti-parallèles (Tsirigos et al. 2018). Les protéines TM avec des hélices α constituent la majorité des protéines membranaires, représentant 25 % à 30 % des protéines codées par le génome (Tsirigos et al. 2018). Elles sont présentes dans toutes les

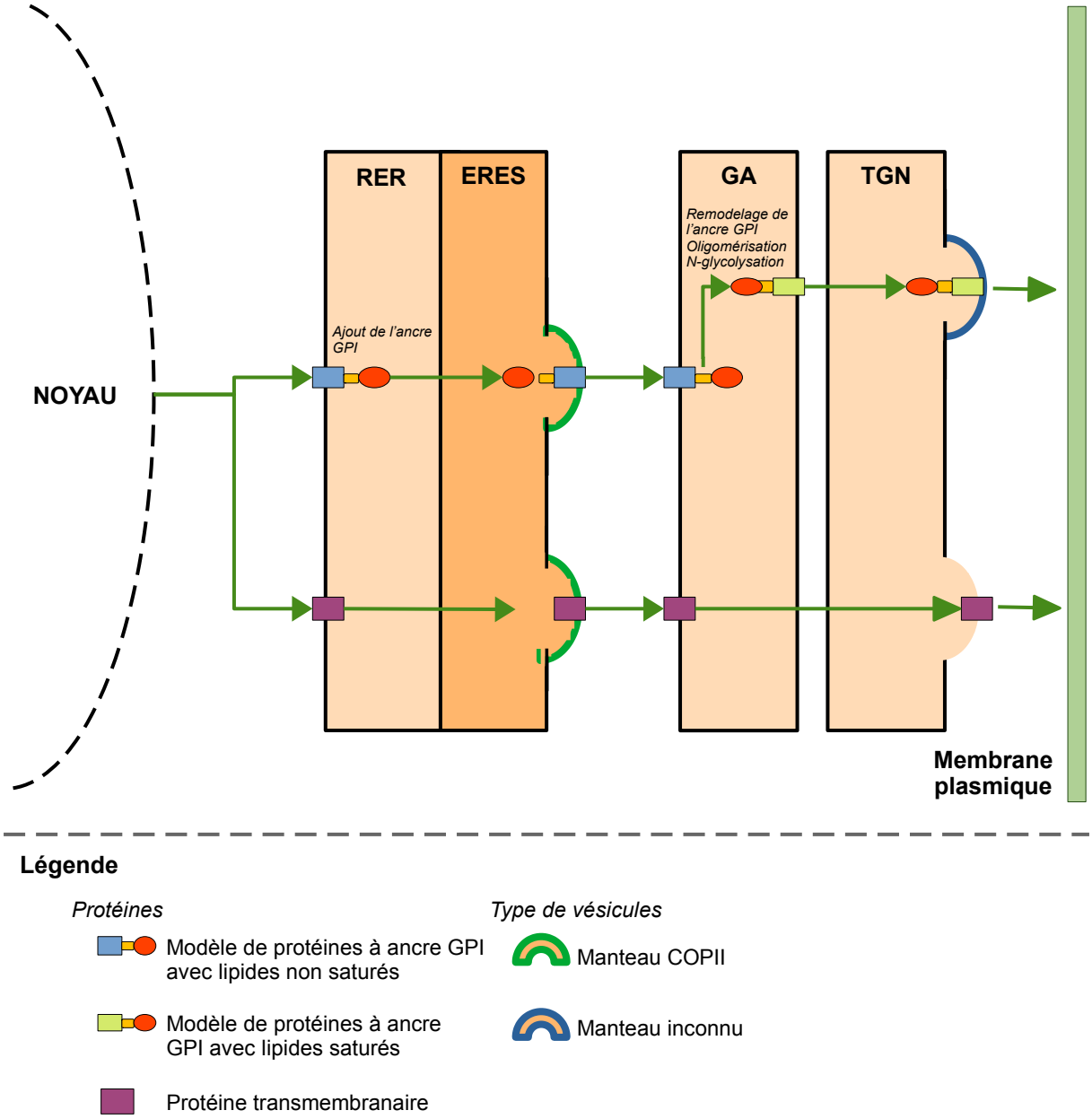


**Figure 2 | Schéma de la synthèse des protéines du surfaceome dans les cellules épithéliales.** COPII, Complexe Protéique II ; ERES, Sites de Sortie du Réticulum Endoplasmique ; GA, Appareil de Golgi ; GPI, Glycosylphosphatidylinositol ; RER, Réticulum Endoplasmique Rugueux ; TGN, Réseau Trans-Golgien.

membranes cellulaires. Les protéines avec des feuillets β n'existent que dans les membranes externes des bactéries à Gram négatif, dans les chloroplastes et les mitochondries (Tsirigos et al. 2018).

Parmi les voies d'acheminement des protéines du surfaceome, tout comme les protéines du sécrétome, les protéines de surface empruntent la voie ER/Golgi (Figure 2) mais, au lieu d'être excrétées dans la matrice extracellulaire, celles-ci sont internalisées dans la PM.

Chez les mammifères, les protéines sont triées selon leur destination au niveau du TGN. Dans les cellules épithéliales, les protéines à ancre GPI subissent un remodelage de l'ancre GPI, une oligomérisation et une N-glycolysation, afin d'être internalisées dans des vésicules spécialisées, puis acheminées au niveau de la PM apicale (Guo et al. 2014, Muniz and Zurzolo 2014). Ainsi, l'ancre GPI possède également un rôle de transport et de ciblage de la protéine pour la guider jusqu'à la PM (Muniz and Zurzolo 2014). D'autres signaux se trouveraient également au niveau des domaines transmembranaires (Guo et al. 2014).

Enfin, il existe une autre voie d'acheminement des protéines du surfaceome au niveau de la PM. Il s'agit de la voie de type IV des voies de sécrétion non conventionnelles (Figure 1) (Gee et al. 2018). Dans cette voie, les protéines transmembranaires et/ou présentant un peptide signal sont synthétisées dans le ER mais, au lieu de passer par l'appareil de Golgi, elles sont transportées directement au niveau de la PM via des vésicules (Dimou and Nickel 2018, Gee et al. 2018).

Au sein du surfaceome, les protéines peuvent être des récepteurs, des transporteurs, des canaux, des protéines d'adhésion à la cellule, des enzymes, des antigènes ou des éléments régulateurs (Bausch-Fluck et al. 2018, Lebreton et al. 2018).

### I. II. III. Interactions protéine-protéine

De nombreuses voies de régulations impliquent des interactions moléculaires, telles que des interactions gènes-protéines ou protéines-protéines (PPI) : l'ensemble des interactions constitue l'interactome (Koh et al. 2012). Dans ce rapport, nous nous intéresserons uniquement aux interactions protéine-protéine.

La caractérisation d'interaction physique entre protéines requiert une investigation expérimentale poussée, afin d'établir des caractéristiques telles que les propriétés d'affinité et de cinétique (Miura 2018). Il existe différentes méthodes pour détecter les PPIs (Koh et al. 2012, Miura 2018). Deux techniques à haut débit ont notamment contribué à la publication de PPI : le double hybride (Y2H) et la purification par affinité couplée à la spectrométrie de masse (AP-MS) (Koh et al. 2012). Pour deux protéines A et B, la méthode de Y2H (Annexe 1) se base sur la complémentation de deux parties d'un facteur de transcription. L'interaction de A avec B induit ainsi la transcription un gène rapporteur, comme la β-galactosidase (Koh et al. 2012). Quant à la méthode AP-MS (Annexe 1), la protéine A est capturée sur une matrice par affinité puis un mélange d'interacteurs potentiels passe au travers de la matrice. Seules les protéines interagissant avec la protéine A sont retenues (Koh et al. 2012). Il est important de noter que chaque méthode a des avantages et des inconvénients. Y2H identifie un grand nombre d'interactions binaires mais elle produit des faux positifs. AP-MS a pour avantage d'examiner les interactions entre plusieurs protéines. Cependant, certaines interactions sont manquées en raison d'une modification post-traductionnelle de l'interacteur, de faible concentration ou d'une interaction instable/transitoire avec la protéine d'intérêt (Koh et al. 2012).

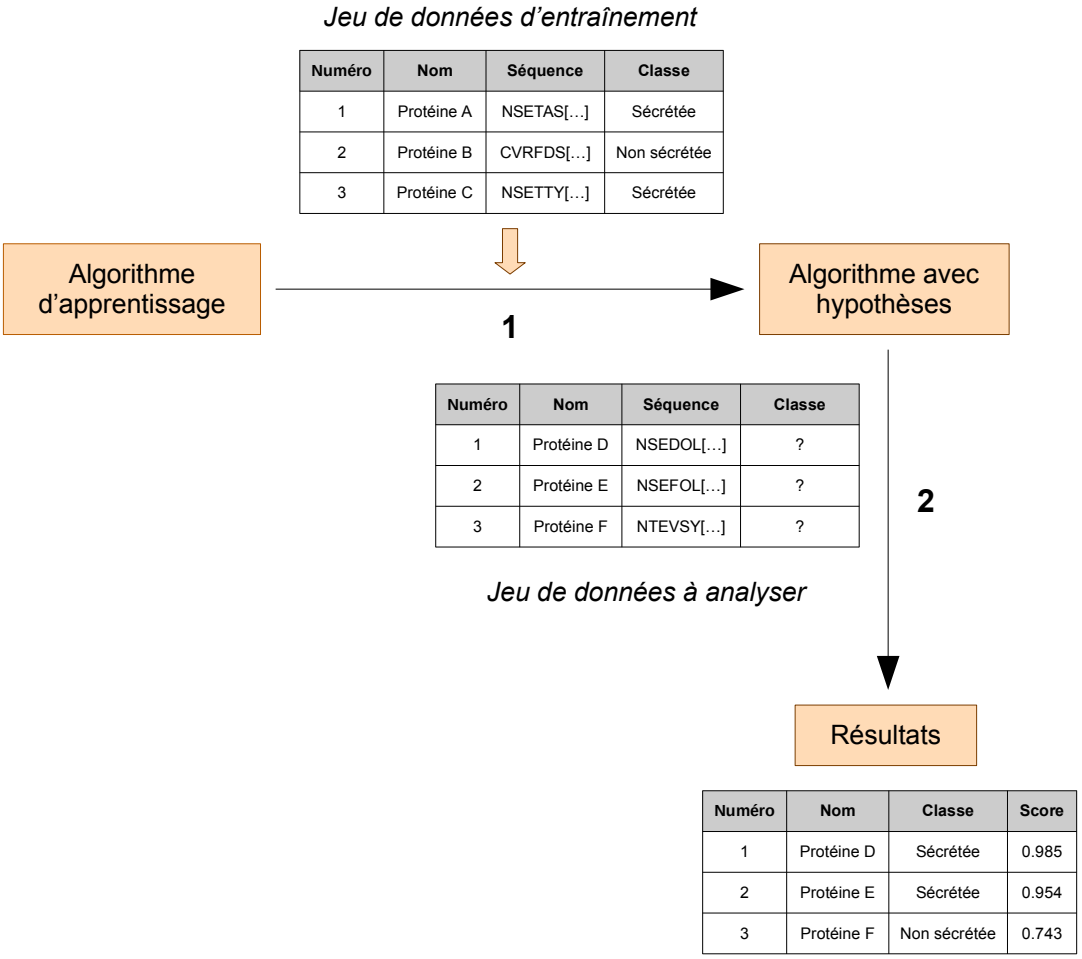


**Figure 3 | Schéma de l'élaboration d'un outil de prédiction.** (1) Entraînement de l'algorithme de l'outil de prédiction en se basant sur les séquences protéiques. (2) Analyse d'un jeu de données. Les résultats sont accompagnés d'un score mais cela peut être aussi une valeur de prédiction. Les valeurs affichées sur cette figure ne sont données qu'à titre d'exemple.

Afin de compenser les difficultés à identifier expérimentalement les PPIs, des méthodes informatiques prédisent des PPIs et leurs réseaux (Tanwar and George Priya Doss 2018). Les méthodes de prédiction se basent sur différentes approches : des approches structurales avec la recherche des séquences et des domaines protéiques et des structures tertiaires qui en découlent, ainsi que par des approches de génomique comparative et de topologie du réseau d'interactions (Kotlyar et al. 2017).

## I. III. Prédiction bioinformatique de la localisation subcellulaire et des interactions protéine-protéine

### I. III I. Prédiction du peptide signal et de la localisation subcellulaire

La prédiction du peptide signal et de la localisation subcellulaire à partir des séquences protéiques est un défi important en Bioinformatique et implique différentes approches : dans ce rapport, nous nous intéresserons aux approches utilisant le *machine learning* (Nielsen et al. 2019). En bref, le *machine learning* regroupe des algorithmes d'apprentissage nécessitant l'apport initial de données afin de produire des hypothèses générales, ce que nous qualifions d'apprentissage supervisé (Kotsiantis 2007).

Dans le cas de la prédiction du peptide signal, les principales tâches sont de différencier les protéines avec et sans peptide signal et de prédire la position du site de clivage du peptide (Nielsen et al. 2019). Pour ce faire, lors de l'apprentissage (Figure 3), des séquences protéiques regroupées en classes, telles que les protéines avec peptide signal et celles sans peptide signal, sont soumises à l'algorithme afin de générer des hypothèses de base selon les motifs présents dans les séquences. Ainsi, lors de l'analyse de nouvelles séquences, l'outil se base sur ces hypothèses et attribue un score à chaque séquence voire chaque acide aminé pour chaque classe (Almagro Armenteros, Tsirigos, et al. 2019, Savojardo et al. 2018). Selon

l'outil, l'algorithme d'apprentissage et la nature du score de prédiction peuvent varier. Par exemple, la cinquième version de l'outil de prédiction SignalP (Figure Annexe 2.2) (Almagro Armenteros, Tsirigos, et al. 2019) utilise des réseaux de neurones convolutifs (CNN) et un réseau de neurones récurrents bidirectionnel (BiRNN) à mémoire à courte et long terme (LSTM et BiLSTM respectivement) (Annexe 2.1), et attribue des probabilités marginales grâce à une classification par champ aléatoire conditionnel (CRF) (Annexe 2.2). La classe possédant le score ou la probabilité la plus élevée sera la classe du résultat final.

Cependant, il faut garder en tête que les protéines avec un peptide signal ne sont pas toujours sécrétées et que certaines protéines sécrétées ne possèdent pas de peptide signal. Afin de les différencier des protéines membranaires et des autres protéines, la localisation subcellulaire doit être considérée.

Il existe différents compartiments au sein d'une cellule dont le noyau, le cytoplasme, la PM et la mitochondrie (Almagro Armenteros et al. 2017). Bien que le peptide signal différencie les protéines membranaires et sécrétées des autres protéines, il existe d'autres pré-séquences pour trier les protéines selon leur localisation intracellulaire, tels que les peptides signaux mitochondriaux (Jose Juan Almagro Armenteros et al. 2019). De plus, pour différencier les protéines membranaires des protéines sécrétées, la présence d'une ancre GPI ou d'un domaine transmembranaire sont détectées à partir de la séquence protéique (Bausch-Fluck et al. 2018). Ainsi, un défi de la prédiction est de détecter les motifs au niveau des extrémités N-terminale et C-terminale (Almagro Armenteros et al. 2017). L'une des solutions est d'utiliser des algorithmes d'apprentissage pouvant lire la séquence protéique dans les deux sens : c'est le cas du BiLSTM (Annexe 2.1), qui est utilisé par la deuxième version de l'outil de prédiction des pré-séquences à l'extrémité N-terminale TargetP (Annexe 2.3, Figure Annexe 2.3)

| Catégories | Motif(s) reconnu(s) par l'outil | Références |
|---|---|---|
| Protéine de la matrice extracellulaire | Peptides signaux<br>Site de clivage | Almagro Armenteros, Tsirigos, et al. 2019 |
| Protéine de la membrane plasmique | Peptides signaux<br>Ancre GPI<br>Domaine TM | Almagro Armenteros et al. 2017<br>Savojardo et al. 2018 |
| Protéine de la mitochondrie | Peptide de transit mitochondrial | Jose Juan Almagro Armenteros et al. 2019 |
| Protéine du cytoplasme | Aucun peptide signal | |

**Tableau 2 | Motifs reconnus par les différents outils de prédiction au sein des séquences protéiques.**

(Almagro Armenteros, Salvatore, et al. 2019) et par l'outil de prédiction de localisation subcellulaire DeepLoc (Figure Annexe 2.4) (Almagro Armenteros et al. 2017), qui complète également son apprentissage par un CNN (Annexe 2.4). De plus, ces outils sont couplés à des modèles d'attention, qui sont des mécanismes qui se focalisent sur les motifs d'intérêt (Tableau 2), quelque soit sa position dans la séquence protéique (Almagro Armenteros et al. 2017, Almagro Armenteros, Salvatore, et al. 2019).

### I. III II. Bases de données répertoriant les interactions protéine-protéine

Actuellement, il existe un bon nombre de bases de données répertoriant les PPIs et qui sont interrogeables pour une ou un ensemble de protéines. Elles rapportent le type d'interactions, la méthode de détection et/ou le réseau résultant (Kotlyar et al. 2017). Par exemple, UniProt (The UniProt Consortium et al. 2021) et STRING (Szklarczyk et al. 2019) regroupent les informations de différentes espèces et HAPPI-2 regroupe les données des PPIs de *Homo sapiens* (Chen et al. 2017). Afin de faciliter l'accès à l'information, le consortium d'Échange Moléculaire International (IMEx) vise à collecter et à valider les PPIs publiées. Cette base de données intègre les interactions rapportées par de nombreuses études, utilise différents identifiants de gènes et de protéines pour la requête, filtre des données selon des critères (fiabilité, type d'interactions) et fournit des résultats dans des formats accessibles (Kotlyar et al. 2017, The IMEx Consortium Curators et al. 2019).

## I. IV. Objectifs du stage

Afin d'établir la liste des protéines sécrétées par le tissu musculaire interagissant avec les protéines de surface du tissu adipeux et inversement, l'équipe BIOMARQUEURS (Bonnet et al. 2020) a développé l'outil ProteINSIDE[5] (Kaspric et al. 2015) (sécrétome et PPI) et utilisé

5 ProteINSIDE : **https://www.proteinside.org/**

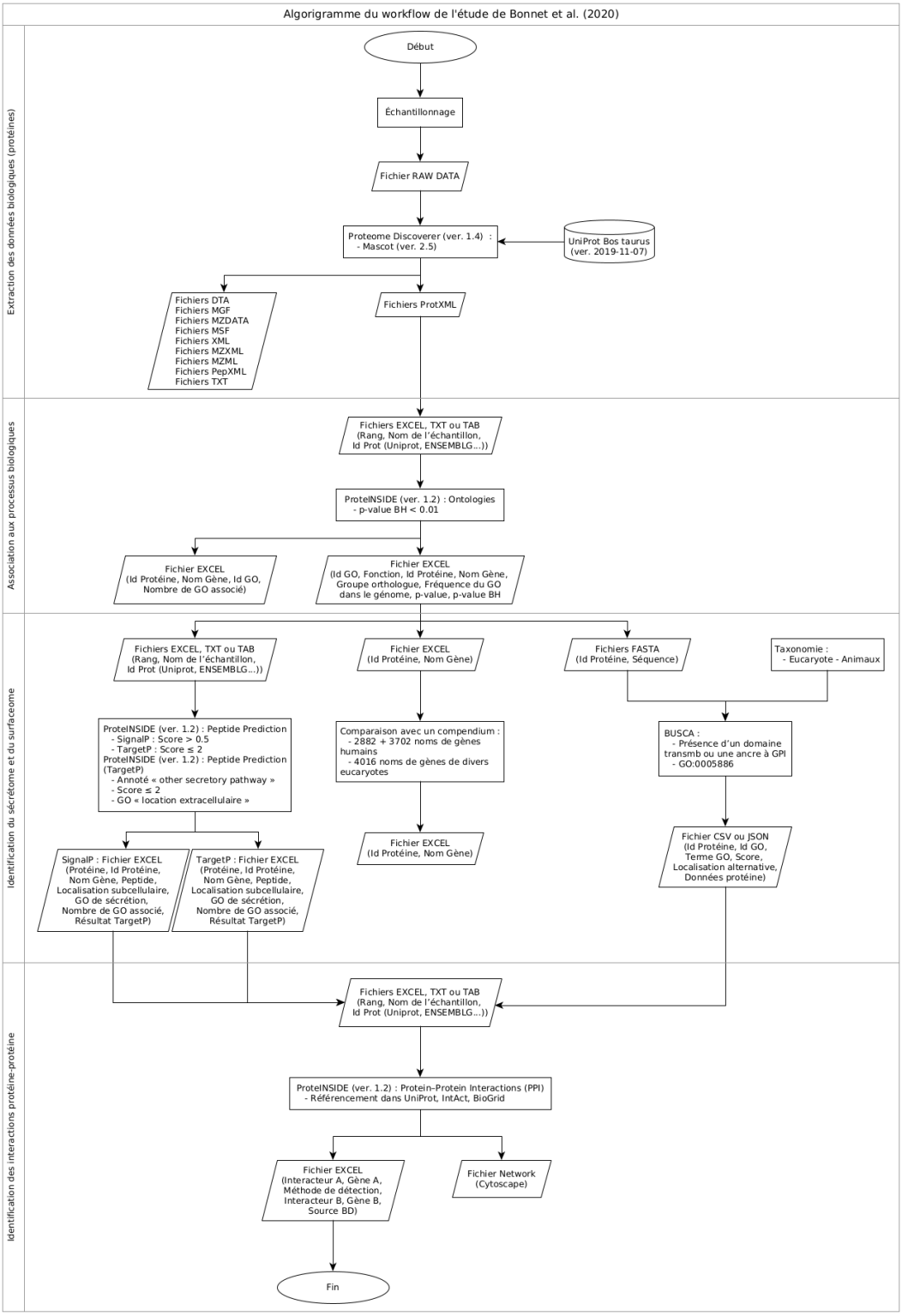


**Figure 4 | Algorigramme de l'étude de Bonnet et al. (2020).** Cette étude se divise en quatre étapes : l'extraction des données biologiques (protéines), l'association aux processus biologiques, l'identification du sécrétome et du surfaceome et l'identification des interactions protéine-protéine. Les données d'entrée et de sortie sont représentées par des parallélépipèdes, les outils et les processus par des rectangles, les bases de données par des cylindres.

l'application web BUSCA[6] (Savojardo, Pier Luigi Martelli, et al. 2018) (surfaceome) (Figure 4). Cependant, aucun outil bioinformatique ou workflow ne détermine de manière automatique l'appartenance des protéines au surfaceome et au sécrétome ainsi que leurs interactions.

Ainsi, l'objectif de ce stage est double. D'une part, il vise à développer un workflow pour identifier le dialogue moléculaire entre deux tissus biologiques résultant d'interactions protéine-protéine. Afin de répondre à la problématique, le développement de ce workflow s'est déroulé en trois étapes. La première étape a consisté à établir une liste d'outils bioinformatiques qui prédisent les séquences protéiques orientant les protéines dans les classes *Sécrétome* ou *Surfaceome*. La deuxième étape est d'intégrer ces outils dans un pipeline afin de les tester et de les évaluer. La dernière étape est de proposer un worflow en Snakemake qui intégrera les outils de prédiction ainsi que d'autres scripts « maison ». D'autre part, ce stage se déroulant au Mésocentre de l'UCA, le workflow développé doit fonctionner sur un cluster de calcul, être exécutable facilement et être le plus généraliste possible afin d'intéresser les laboratoires de la communauté Auvergne Bioinformatique (AuBi)[7].

6 BUSCA – Bologna Biocomputing Group : **http://busca.biocomp.unibo.it/**
7 Plateforme AuBi : **https://ressources.france-bioinformatique.fr/fr/plateformes/aubi**

| | Espèce | Révisée | Termes UniProt de localisation subcellulaire | Termes Uniprot clés (keyword) |
|---|---|---|---|---|
| Protéine à ancre GPI | *Homo sapiens* [9606] | Oui | GPI-anchor [SL-9902]<br>Cell membrane [SL-0039]<br>Apical cell membrane [SL-0015]<br>Basolateral cell membrane [SL-0026]<br>Endoplasmic reticulum [SL-0095]<br>Extracellular side [SL-9911] | |
| Protéine à domaine TM en hélice α | *Homo sapiens* [9606] | Oui | Cell membrane [SL-0039]<br>Single-pass type I membrane protein [SL-9905]<br>Single-pass type II membrane protein [SL-9906]<br>Single-pass type III membrane protein [SL-9907]<br>Single-pass type IV membrane protein [SL-9908]<br>Multi-pass membrane protein [SL-9909]<br>Apical cell membrane [SL-0015]<br>Basolateral cell membrane [SL-0026]<br>Microvillus [SL-0296]<br>Microvillus membrane [SL-0294]<br>Cilium [SL-0066]<br>Cilium membrane [SL-0305]<br>Lamellipodium [SL-0291]<br>Lamellipodium membrane [SL-0292]<br>Flagellum [SL-0117]<br>Flagellum membrane [SL-0115]<br>Membrane lipid raft [SL-0370]<br>Stereocilium [SL-0302]<br>Stereocilium membrane [SL-0303]<br>Ruffle [SL-0300]<br>Ruffle membrane [SL-0301] | Transmembrane helix [KW-1133] |
| Protéine à domaine TM en tonneau β | *Homo sapiens* [9606] | Oui | Mitochondrion [SL-0173]<br>Mitochondrion outer membrane [SL-0172]<br>Membrane [SL-0162]<br>Cell membrane [SL-0039] | Transmembrane beta strand [KW-1134] |

**Tableau 3 | Termes utilisés sous UniProt pour répertorier les protéines d'intérêt.** La colonne « *Révisée* » précise si l'entrée appartient à la section Swiss-Prot d'UniProtKB (oui) ou à la section annotée par ordinateur (non). KW, terme de la catégorie « Mot-clés » (KeyWord) ; SL, terme de la catégorie « Localisation subcellulaire » (*Subcellulaire Location*) .

# II. Réalisations

## II. I. Matériel et méthodes

### II. I. I. Jeu de données test

Afin de tester les outils déterminant l'appartenance des protéines au sécrétome et au surfaceome, un jeu de données test a été constitué à partir du jeu de données de ProteINSIDE. Ce jeu de données était composé de 133 identifiants nommés « entrées protéiques » de *Homo sapiens* provenant de la base de données UniProt[8] et appartenant à différents compartiments cellulaires : la matrice extracellulaire, le cytoplasme, la matrice et la membrane interne de la mitochondrie. Il a été mis à jour, avec la suppression de trois identifiants, l'ajout de quatre identifiants et l'actualisation d'un identifiant (134 identifiants).

Trente et une autres entrées de *Homo sapiens* ont été ajoutées pour les caractéristiques suivantes : 10 protéines à ancre GPI, 10 protéines avec un domaine transmembranaire en hélice α, 5 protéines avec un domaine transmembranaire en tonneau β et 6 protéines sécrétées par la voie non-conventionnelle. Les protéines membranaires ont été sélectionnées via UniProt en utilisant différents paramètres (Tableau 3). Quant aux protéines sécrétées par voie non-conventionnelle, elles ont été répertoriées à partir de la bibliographie (Tableau 1).

Ainsi, le jeu de données comptabilise un total de 165 identifiants UniProt (Annexe 3) et couvrent l'ensemble des compartiments cellulaires d'intérêt (Figure 5).

### II. I. II. Élaboration du workflow

#### II. I. II. I. Environnement de développement informatique

L'ensemble des travaux de ce stage a été réalisé sur un ordinateur Dell sous Ubuntu 20.04.2 LTS et un processeur Intel® Core™ i5-5200U. Les scripts développés, les fichiers

8 UniProt : **https://www.uniprot.org/**

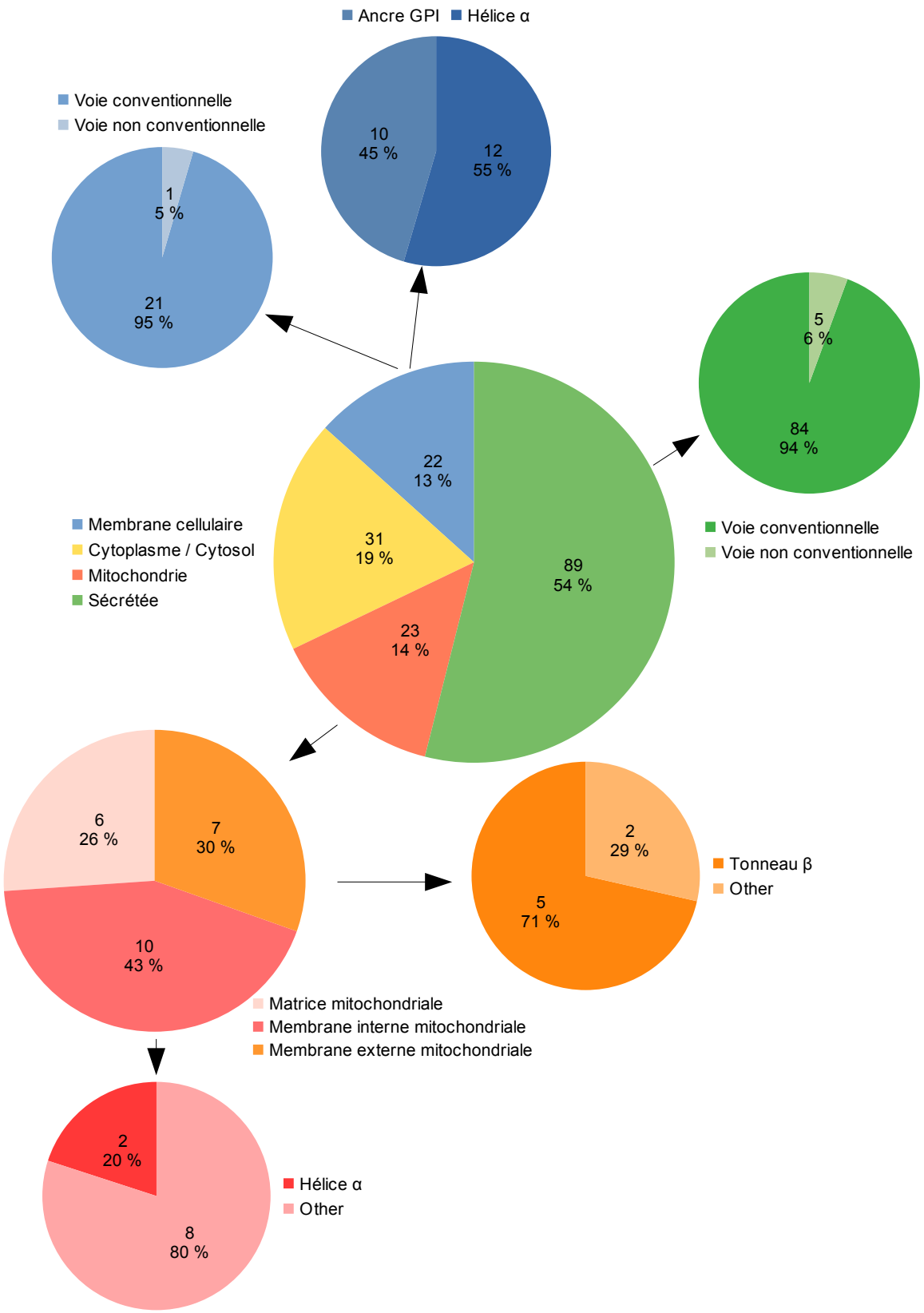


**Figure 5 | Répartition du jeu de données test au sein des compartiments subcellulaires.** Ces répartitions ont été établies à partir des données de UniProt relevées le 25 Mai 2021.

Snakemake et les fichiers contenant les données de tests ont été sauvegardés sur un dépôt privé GitHub[9]. Les scripts ont été testés en local sous environnement Conda et sur le cluster HPC2 du Mésocentre de l'Université Clermont-Auvergne.

Conda est un système de gestion de packages opensource et d'environnements virtuels[10]. Il permet de créer, de charger et de basculer entre différents environnements possédant chacun leurs propres packages dans une version précise. Ainsi, il installe les packages et leurs dépendances, les exécute et les met à jour. La version 4.10.1 a été utilisée dans le cadre de ce stage.

Le cluster de calcul HPC2 possède actuellement plus de 1100 cœurs répartis sur 58 nœuds. Chaque nœud est configuré de la même manière, avec les mêmes versions de logiciels, bibliothèques et applications (appelées des modules), ainsi les analyses appelées jobs sur un cluster sont exécutables sur n'importe quel nœud du cluster. De plus, chaque nœud est défini par un nombre de CPU et une quantité de RAM spécifique. Les jobs sont gérés par l'ordonnanceur SLURM[11], un gestionnaire et planificateur de tâches open-source, tolérant aux pannes et hautement évolutif pour toutes les tailles de clusters.

Une fois que les scripts ont été finalisés, un travail d'intégration de ces scripts en workflow via l'outil Snakemake (Mölder et al. 2021) a été initié. Il s'agit d'un langage de workflow pour créer des analyses de données reproductibles et évolutives[12]. Le workflow est codé en Python 3.7 et en Bash. Les dépendances liées aux outils utilisés sont décrites dans le fichier *Readme.md* (Annexe 4). Il est évalué avec l'espèce *Homo sapiens*.

9 GitHub : **https://github.com/**
10 Documentation de Conda : **https://docs.conda.io/projects/conda/en/latest/**
11 Slurm Workload Manager : **https://slurm.schedmd.com/overview.html**
12 Documentation de Snakemake : **https://snakemake.readthedocs.io/en/stable/**

Afin d'informer l'utilisateur du déroulement de l'exécution du script, l'ensemble des scripts Python utilise la librairie ***Logging*** pour écrire des messages ou des logs. Ainsi, les potentielles erreurs rencontrées au cours de son exécution sont également spécifiées. Le niveau des logs peut être défini via un argument en ligne de commande, sachant que l'outil affiche par défaut les logs d'informations (*info*), d'avertissement (*warning*), d'erreur (*error*) et d'erreur critique (*critical*). Ces logs sont écrits dans un fichier d'extension **`.log`**. L'utilisateur peut également définir si les résultats des outils sont écrits sur la sortie standard et/ou dans un fichier au format texte ou au format spécifié en entrée. Si un nom de fichier est spécifié en ligne de commande, les méthodes qui écrivent les résultats analysent ce nom afin de vérifier la présence d'un chemin menant à un répertoire et d'une extension grâce aux fonctions `path.dirname()` et `path.basename()` respectives de la librairie ***Os***. Si aucun chemin n'est spécifié, les fichiers sont écrits dans le répertoire courant. Concernant l'extension, les méthodes ajoutent l'extension texte **`.txt`** si aucune n'est détectée (ou **`.faa`** dans le cas des fichiers fasta). De plus, si un outil écrit plusieurs résultats, le nom de fichier entré se voit ajouter des caractères afin de spécifier le type de données qu'il contient.

Enfin, afin de vérifier que les scripts Python mis en ligne fonctionnent et que les résultats correspondent aux attentes, un workflow d'intégration continue disponible sur Github a été mis en place en utilisant le module Pytest[13].

#### II. I. II. II. Préparation des données : g:Convert de g:Profiler

g:Profiler[14] est un serveur web avec une interface de programmation (API) pour caractériser et manipuler des listes de gènes et/ou de protéines (Raudvere et al. 2019). Dans le cadre de ce

13 Documentation de Pytest : **https://docs.pytest.org/en/6.2.x/contents.html**
14 g:Profiler : **https://biit.cs.ut.ee/gprofiler/page/docs**

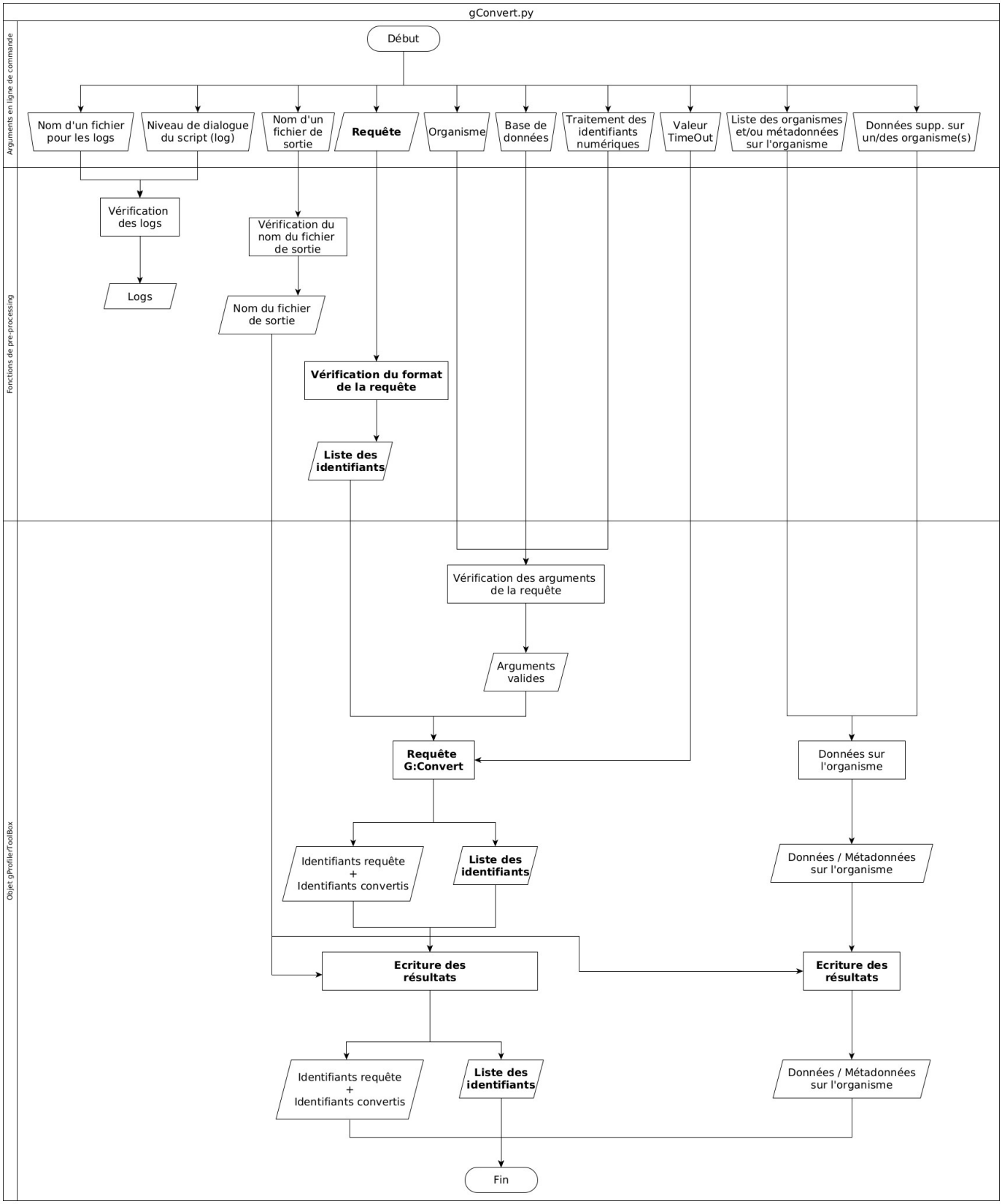


**Figure 6 | Algorigramme du script Python g:Convert.** Les étapes clés sont en gras. Les données d'entrée et de sortie sont représentées par des parallélépipèdes, les outils et les processus par des rectangles.

projet, un script a été développé en Python Orienté Objet en utilisant l'API de g:Profiler afin d'effectuer des requêtes POST avec une sortie au format JSON (Annexe 5).

Ce script utilise l'outil g:Convert[15], qui convertit des noms de protéines issus de la base de données UniProt au format des identifiants protéiques de la base de données Ensembl (ENSP) et ce en les faisant correspondre via les identifiants de gènes Ensembl Biomart comme référence. Cet outil donne en sortie un dictionnaire contenant les identifiants convertis en clé, et en valeurs, les noms des gènes associés avec une description des gènes correspondants ainsi que les paramètres utilisés pour la requête.

Il nécessite différents arguments en ligne de commande (Figure 6) : les identifiants à convertir, les paramètres nécessaires pour la requête de g:Convert et la sortie des résultats (sortie standard ou fichier).

Tout d'abord, une première fonction, `format_query`, formate les données en fonction de la nature de leur entrée (ligne de commande ou fichier), afin d'extraire une liste. Cette liste est envoyée dans l'objet `gProfilerToolBox`. En plus de la liste des identifiants, cet objet a pour attribut les paramètres de la requête : l'organisme dont sont issus les identifiants, l'espace de nom de conversion et un autre espace de nom de conversion si les identifiants sont numériques. Par défaut, le paramètre sur l'organisme est vide (`None`), celui de l'espace de nom de conversion `ENSP` et celui pour les identifiants numériques `ENTREZGENE_ACC`. Une fois les attributs initialisés, la méthode `gConvert` envoie une requête POST et récupère les résultats. Par la suite, la méthode `idList_gConvert` extrait uniquement les identifiants convertis. Selon l'argument de la sortie des données, les identifiants convertis sont écrits sur la sortie standard et/ou dans un fichier au format texte ou au format spécifié en entrée. Il est

15 Interface Web de g:Convert : **https://biit.cs.ut.ee/gprofiler/convert**

également possible de récupérer la liste des identifiants donnés en entrée et leurs résultats de conversion. Par défaut, ils sont écrits dans les logs mais, si un nom de fichier est spécifié, ils sont écrits dans ce fichier.

L'objet `gProfilerToolBox` comporte également une méthode pour obtenir les données sur l'organisme choisi dans la requête. Cette même méthode a également d'autres fonctionnalités. Tout d'abord, si aucun organisme n'est précisé, elle récupère l'intégralité des organismes pris en charge par g:Profiler. Ainsi, elle vérifie également l'argument entré pour le paramètre de l'organisme. Si aucun nom n'est entré ou si le nom est erroné, le nom complet des organismes et leur abréviation seront extraits puis écrits sur la sortie standard ou dans un fichier. Ensuite, une dernière méthode vérifie les deux autres paramètres de requête concernant les espaces de nom de conversion.

La sortie du script est la liste des identifiants convertis. Dans notre cas, une conversion des identifiants a été effectuée selon le format ENSP et la taxonomie *Homo sapiens* (*hsapiens*). Ces identifiants sont utilisés en entrée de la prochaine étape.

#### II. I. II. III. Extraction des séquences protéiques : Entrez Direct

Le script développé au cours de ce stage utilise les utilitaires Esearch et Efetch de programmation Entrez (E-utilities) du NCBI (Sayers 2010), via l'outil Bash Entrez Direct (Edirect) (Kans 2013). Une requête au format SQL est effectuée avec les identifiants convertis par g:Convert en utilisant Esearch, qui permet d'accéder aux données de la base de données protéique. Puis les séquences protéiques sont récupérées avec Efetch en chaînant les résultats de la commande précédente grâce à un « pipe ».

Le script Bash (Annexe 6), dans un premier temps, utilise la fonction de Edirect `join-into-groups-of` pour concaténer les identifiants issus de *gConvert.py* en une liste de

**A**

```
signalp -fasta [Nom-Fichier-1].faa -format short -org 'euk' -
    prefix 'output-SignalP_[Nom-Fichier-2]'
```

**B**

```
targetp -fasta [Nom-Fichier-1].faa -format short -org 'non-
    pl' -prefix 'output-TargetP_[Nom-Fichier-2]'
```

**C**

```
deeploc -f [Nom-Fichier-1].faa -o 'output-DeepLoc_[Nom-
    Fichier-2]'
```

**Figure 7 | Ligne de commande des outils permettant de prédire la présence d'un peptide signal (A, B) et la localisation subcellulaire (C).** Les trois outils utilisent le même fichier multi-fasta contenant les séquences protéiques en entrée (`-fasta` pour SignalP et TargetP, `-f` pour DeepLoc). Les fichiers de sortie contenant les résultats ont pour nom `output-NomOutil_[Nom-Fichier-2]` (`-prefix` pour SignalP et TargetP, `-o` pour DeepLoc). Des arguments autres que ceux qui ont été utilisés sont également disponibles. Leurs valeurs par défaut ont été dans ce cas utilisés. (A) Les valeurs choisies pour les arguments de `-format` et `-org` sont les valeurs par défaut de SignalP. L'argument `short` de `-format` permet d'obtenir uniquement les résultats de prédiction pour chaque protéine. La prédiction a été réalisée sur des organismes eucaryotes (`euk`). (B) Les valeurs choisies pour les arguments de `-format` et `-org` sont les valeurs par défaut de TargetP. L'argument `short` de `-format` a la même finalité que celui de l'outil SignalP. La prédiction a été réalisée sur des organismes non-plantes (`non-pl`).

caractères séparés par une virgule et par bloc de 5000 identifiants, qui est la taille pour optimiser la requête sur un serveur distant. Ensuite, afin d'adapter cette liste à la requête du programme Esearch, la liste est redirigée par un « pipe » dans la fonction Bash `sed`, qui remplace les virgules par la chaîne de caractère `' OR '`. Enfin, cette nouvelle liste est redirigée par un « pipe » dans un ensemble parallélisé. Cet ensemble envoie tout d'abord une requête via le programme Esearch, avec pour argument de base de données *protein* et d'organisme *human*, puis les résultats de celle-ci sont redirigés dans le programme Efetch, qui extrait les données au format fasta et rend donc disponible les séquences protéiques.

Cet ensemble parallélisé envoie au maximum un total de trois jobs à la fois, qui est le nombre de requêtes maximales par seconde autorisé par Entrez Direct.

Afin d'éviter la redondance au sein du jeu de données, le script Python *NonRedundantFasta.py* retire du fichier fasta toutes les séquences identiques et garde uniquement une séquence par protéine.

#### II. I. II. IV. Sélection des outils de prédiction

Afin de déterminer si une protéine appartient au sécrétome ou au surfaceome, un ensemble d'outils a été sélectionné selon les critères suivants : l'entrée doit être un fichier fasta ou multi-fasta, il doit prendre en charge les organismes eucaryotes dont *Homo sapiens* et *Bos taurus*, avoir une licence open-source, une interface en ligne de commande disponible et il doit pouvoir être installé en local et sur un cluster de calcul (Annexe 7). Ainsi, SignalP (Almagro Armenteros, Tsirigos, et al. 2019) et TargetP (Almagro Armenteros, Salvatore, et al. 2019) ont été sélectionnés pour la prédiction du peptide signal et DeepLoc (Almagro Armenteros et al. 2017) pour la localisation subcellulaire (Figure 7).

Il est important de noter que la prédiction de chaque outil correspond à la classe qui possède la valeur de score ou de probabilité la plus élevée : plus la valeur est proche de 1, plus la séquence protéique est associée à cette classe.

Utilisé sous sa version 5.0 et pour chaque séquence protéique, SignalP[16] donne le résultat de la prédiction, soit la présence d'un peptide signal (Sec/SPI pour les eucaryotes) ou non (Other), ainsi que la valeur de probabilité associée à chaque classe. Si un peptide signal est prédit, la position et la valeur de probabilité associée du site de clivage est précisée.

La version 2.0 de TargetP[17] prédit la localisation subcellulaire d'une séquence protéique en se basant sur les pré-séquences localisées dans sa partie N-terminale. Pour chaque séquence protéique, TargetP donne le résultat de la prédiction, la valeur de la probabilité pour la présence d'un peptide signal (SP), d'un peptide de transit mitochondrial (mTP) et pour l'absence de peptide signal (noTP). Si un peptide signal ou de transit est détecté, la position et la valeur de probabilité pour cette position du site de clivage sont indiqués.

Enfin, DeepLoc[18] prédit pour chaque séquence protéique la localisation subcellulaire : le noyau, le cytoplasme, le milieu extracellulaire, la mitochondrie, la membrane cellulaire, le réticulum endoplasmique, le plaste, l'appareil de Golgi, le lysosome, la vacuole ou le peroxisome. Les résultats regroupent la prédiction de l'outil ainsi que les valeurs des probabilités de prédiction pour chaque classe.

Ainsi, quatre classes comportent des données d'intérêt : les protéines de la classe *Sec/SPI* pour SignalP, les protéines de la classe *SP* pour TargetP, les protéines des classes *Milieu extracellulaire* et *Membrane cellulaire* pour DeepLoc. Un script Python, *DataToolExtract.py*,

16 Interface Web de SignalP 5.0 : **http://www.cbs.dtu.dk/services/SignalP/**
17 Interface Web de TargetP 2.0 : **http://www.cbs.dtu.dk/services/TargetP/**
18 Interface Web DeepLoc : **http://www.cbs.dtu.dk/services/DeepLoc/**

a été développé afin d'extraire les identifiants appartenant à ces classes et les valeurs des probabilités associées, en générant un fichier par outil et par classe.

Ensuite, un autre script Python, *SecretSurf.py*, compare les résultats de chaque outil selon la catégorie *Sécrétome* ou *Surfaceome*. Pour la catégorie *Sécrétome*, l'outil compare les valeurs de prédiction de SignalP puis de TargetP à une valeur seuil de prédiction et seules les protéines dont la valeur est supérieure ou égale à cette valeur sont ajoutées à une liste : dans ce rapport, la valeur seuil de prédiction choisie est de 0,90 et a été choisie de manière arbitraire, le choix pouvant être laissé à la discrétion de l'utilisateur. Par la suite, le même traitement est appliqué aux résultats des outils de prédiction de la localisation subcellulaire (dans ce cas, uniquement pour DeepLoc), pour obtenir une deuxième liste, contenant cette fois des protéines de la classe *Matrice extracellulaire*. Enfin, cette liste et la liste de protéines avec un peptide signal sont comparées : si une protéine apparaît dans les deux listes, elle est alors ajoutée à la liste du sécrétome. Le même procédé est appliqué à la catégorie *Surfaceome*, à la seule différence que les identifiants protéiques de SignalP et TargetP sont comparées à ceux de la classe *Membrane cellulaire* de DeepLoc. À la sortie de cet outil, deux fichiers contenant la liste de chaque catégorie sont générés.

#### II. I. II. V. Récupération des données sur les interactions protéine-protéine : PSICQUIC

Utilisé par l'IMEx, PSICQUIC[19] est un web-service d'accès aux ressources de données sur les interactions moléculaires en standardisant les requêtes (Aranda et al. 2011). Une requête est

19 Interface Web PSICQUIC : **http://www.ebi.ac.uk/Tools/webservices/psicquic/view/home.xhtml;jsessionid=2D24538EB1B620ED5152DBE6CD4EC8C9**

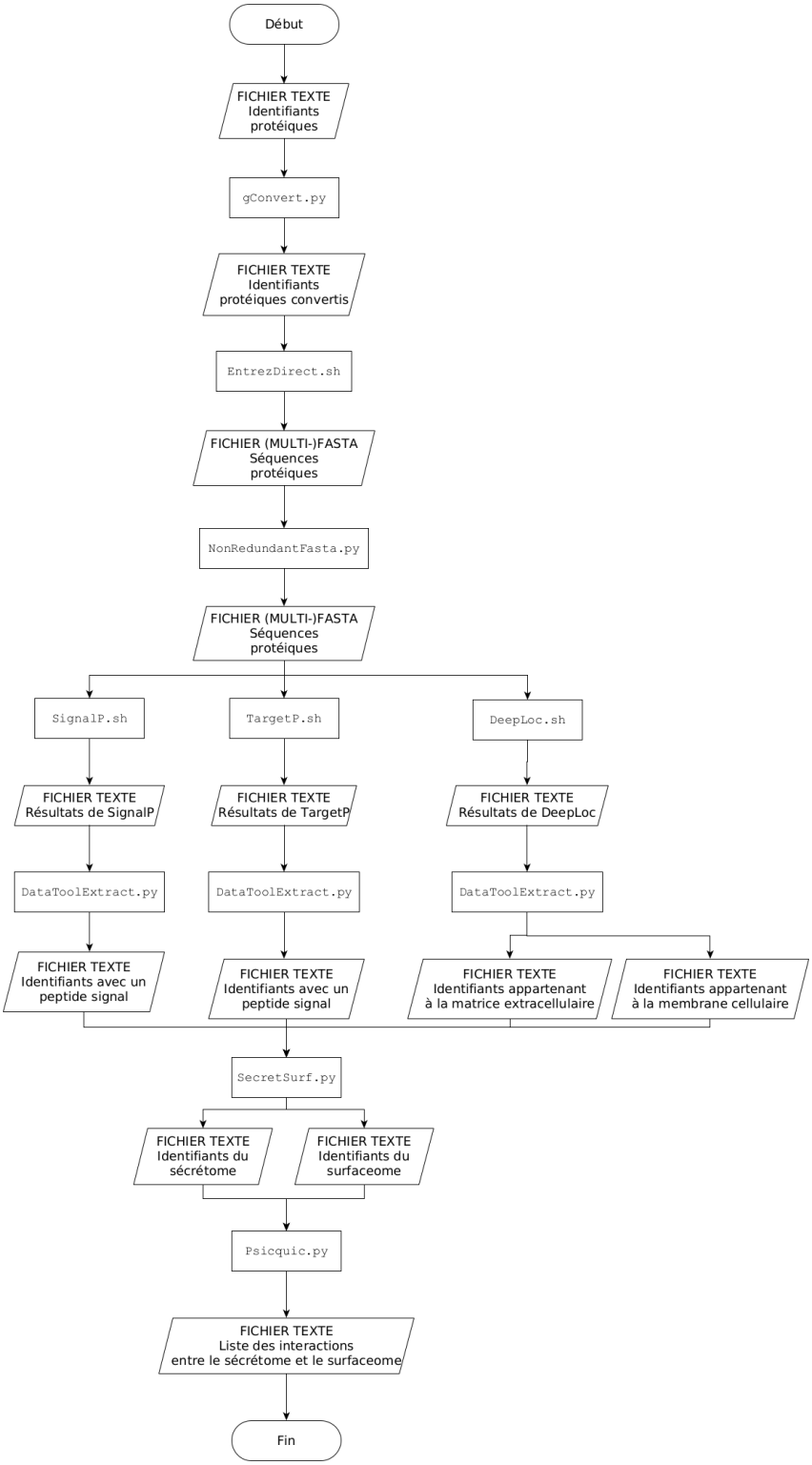


**Figure 8 | Algorigramme du workflow Talkmine.** Les données d'entrée et de sortie sont représentées par des parallélépipèdes, les outils et les processus par des rectangles. Les extensions des scripts présents dans le workflow sont les suivantes : `.py`, script Python ; `.sh`, script Shell (Bash). Le format des entrées et/ou des sorties sont précisées en capitales.

effectuée via une interface web ou via un service client accessible à différents langages de programmation tel que Python (Aranda et al. 2011).

Le script Python *Psicquic.py* développé effectue une requête à partir de deux listes fournies en entrée, une liste d'identifiants protéiques appartenant au sécrétome et une liste d'identifiants protéiques appartenant au surfaceome, et du nom d'un organisme. Selon les services disponibles du registre[20], qui sont les interactions binaires dispensées par différentes bases de données, il génère une liste des interactions possibles entre les identifiants donnés. L'utilisateur peut également choisir une base de données en particulier. Si le nom de cette base de données est erroné ou si elle n'est pas dans la liste des services disponibles, le script prévient l'utilisateur en soumettant la liste des services accessibles.

Pour l'analyse préliminaire présentée dans ce rapport, la base de données *IntAct*[21] (Orchard et al. 2014) et l'organisme *Homo sapiens* ont été sélectionnés.

## II. II. Résultats

L'accès à l'intégralité des scripts et fichiers déposés sur le GitHub est possible sur demande. Après avoir brièvement exposé le workflow, les résultats des scripts développés sont explicités.

### II. II. I. Développement d'un workflow en Snakemake pour déterminer les interactions protéine-protéine

L'utilisateur fournit en entrée une liste d'identifiants protéiques (Figure 8). Ces identifiants sont traités par l'outil *gConvert.py* et le résultat est envoyé dans l'outil *EntrezDirect.sh*, afin de récupérer les séquences protéiques correspondantes. Ces séquences sont injectées dans les

20 Registre des différentes bases de données accessibles via PSICQUIC : **http://www.ebi.ac.uk/Tools/webservices/psicquic/registry/registry?action=STATUS**

21 Base de données IntAct : **https://www.ebi.ac.uk/intact/**

différents outils de prédiction SignalP (*signalp.sh*), TargetP (*targetp.sh*) et DeepLoc (*deeploc.sh*). Les résultats de chaque outil sont traités par l'outil *DataToolExtract.py*, qui génère une liste des protéines possédant un peptide signal pour SignalP et TargetP et deux listes de protéines localisées dans le milieu extracellulaire ou dans la membrane cellulaire pour DeepLoc. L'intégralité des listes est alors soumise à l'outil *SecretSurf.py*, afin de comparer les résultats et d'établir la liste des protéines appartenant au sécrétome ou au surfaceome. La liste contenant les protéines du sécrétome et celle contenant les protéines du surfaceome sont envoyées dans l'outil *Psicquic.py*, qui génère la liste des PPIs prédites entre les protéines des deux compartiments.

### II. II. II. Analyse du jeu de données

#### II. II. II. I. Correction du jeu de données de test

Avant de soumettre le jeu de données test au workflow, les identifiants ont été entrés dans la base de données UniProt. Deux identifiants (P01233, P01243) ont été identifiés comme étant obsolètes et sont retrouvés comme accession secondaire au sein de quatre autres entrées (P0DN86 et P0DN87 pour P01233, P0DML2 et P0DML3 pour P01243) : ils ont été remplacés par ces entrées, localisées dans la matrice extracellulaire. Un autre identifiant, A4D1N9, a été identifié comme un numéro d'accession secondaire d'un autre identifiant présent dans le jeu de données, P07738 : il a donc été supprimé. Enfin, l'identifiant Q16106 est également une accession secondaire de l'identifiant P36871 mais celui-ci n'appartient pas au jeu de données. Ainsi, l'identifiant P36871 remplace l'identifiant Q16106.

#### II. II. II. II. Étape 01 – Conversion des identifiants : g:Convert

En injectant le jeu de données test dans l'outil *gConvert.py*, une liste de 855 identifiants protéiques Ensembl est obtenue pour un taux de conversion de 100 %, avec une médiane de 3

isoformes par identifiant, un premier et troisième quartiles de 2 et de 7 isoformes par identifiant, respectivement.

#### II. II. II. III. Étape 02 – Récupération des séquences protéiques : Entrez-Direct

La liste de 855 identifiants protéiques Ensembl récupère via le script *EntrezDirect.sh* un total de 325 séquences protéiques. Après soumission de ce fichier multi-fasta au script *NonRedundantFasta.py*, 171 séquences protéiques non redondantes ont été extraites, avec la suppression de 154 séquences protéiques.

#### II. II. II. IV. Étape 03 – Prédiction du peptide signal (SignalP et TargetP), de la localisation subcellulaire (DeepLoc) et extraction des données

Les résultats sortant des outils SignalP, TargetP et DeepLoc et soumis à l'outil *ExtractData.py* puis *SecretSurf.py* ont généré une liste de 62 protéines appartenant au sécrétome et une liste de 11 protéines appartenant au surfaceome, pour une valeur seuil de prédiction fixée ici à 0,90.

#### II. II. II. V. Prédiction des interactions protéine-protéine : PSICQUIC

Avec la liste d'identifiants protéiques prédits comme appartenant au sécrétome et au surfaceome, l'outil PSICQUIC a répertorié un total de 171 PPIs.

## III. Discussion

Le workflow Talkmine a été développé dans le but d'identifier les PPIs entre deux tissus biologiques. Pour cela, trois outils pour prédire les protéines appartenant au sécrétome ou au surfaceome ont été intégrés à un workflow sous Snakemake. Ils sont combinés à des scripts pour 1) utiliser des services afin de préparer les identifiants protéiques donnés en entrée (g:Convert), 2) récupérer les séquences protéiques correspondantes tout en évitant la redondance (eDirect), 3) extraire et formater les données sortantes des trois outils, et 4) obtenir la liste des PPIs (PSICQUIC). Le workflow produit des résultats préliminaires avec encore de nombreux points qui restent à améliorer.

Le premier point concerne les scripts. Tout d'abord, le script *gConvert.py* ne vérifie pas de manière optimale les arguments nécessaires à la requête pour l'outil g:Convert. Dans sa version actuelle 0.7.3, il réalise deux requêtes qui génèrent les identifiants convertis et la liste de l'ensemble des espèces prises en charge par g:Profiler. L'idéal serait de générer la liste une fois puis de l'actualiser en fonction des mises à jour de g:Profiler. De plus, bien que le script informe l'utilisateur du nombre et du pourcentage d'identifiants convertis, il n'informe pas quels identifiants n'ont pas été convertis et pour quelle raison. Par exemple, le jeu de données test a été utilisé avant sa correction et les quatre identifiants mentionnés dans la partie Résultats (*Correction du jeu de données*) n'ont pas été convertis. En utilisant l'API de la base de données UniProt, il serait possible d'informer l'utilisateur et de l'orienter dans la correction de son jeu de données. Le deuxième script, *EntrezDirect.sh*, ne possède pas d'arguments personnalisables. Cependant, il existe une version de Edirect sous le langage Python, Entrezpy (Buchmann and Holmes 2019). Pour la suite, il sera envisagé de poursuivre avec cette version, afin de faciliter la mise en place du workflow d'intégration continue sous

GitHub. Le script *SecretSurf.py* est encore à améliorer, notamment dans le recoupement des résultats : dans la première partie, les protéines sécrétées par voie non-conventionnelle ont été présentées comme des protéines ne possédant pas de peptide signal. Grâce aux outils de prédiction de la localisation subcellulaire, ces protéines sont prédites comme appartenant à la matrice extracellulaire. Cependant, le script recoupe les données de la manière suivante : si la protéine est prédite comme possédant un peptide signal et localisée dans la matrice extracellulaire, elle est classée comme appartenant au sécrétome. Ainsi, le script actuel exclut ces protéines, alors qu'elles sont à considérer pour leur appartenance à la classe *Matrice extracellulaire*. Afin de résoudre ce problème, il serait intéressant de comparer les résultats des outils prédisant la localisation subcellulaire avec une liste contenant les identifiants des protéines sécrétées par voie non-conventionnelle. Enfin, les filtres de l'outil *Psicquic.py* sont à améliorer car seuls les premiers identifiants sont vérifiés comme appartenant à l'une des deux listes.

Le deuxième point concerne les résultats de l'outil *gConvert.py* et *EntrezDirect.sh*. Le jeu de données test possède un total de 165 identifiants protéiques. Cependant, en le soumettant au premier script du workflow, *gConvert.py*, celui-ci a donné en sortie 855 identifiants protéiques Ensembl. En effectuant une requête sur la base de données UniProt, les identifiants supplémentaires correspondent à ceux des isoformes, qui ont été répertoriées dans des revues ou qui ont été informatiquement alignés comme séquences d'isoformes potentielles. Par contre, si ces identifiants sont donnés en entrée de l'outil *EntrezDirect.sh*, celui-ci donne un fichier multi-fasta contenant 325 séquences puis 177 séquences protéiques après élimination de la redondance. Il est donc nécessaire d'informer l'utilisateur de la correspondance à des isoformes ou non.

| | Version | Espèces | Références |
|---|---|---|---|
| | *Prédiction du peptide signal* | | |
| SignalP | 5.0 | Eucaryotes<br>Archées<br>Bactéries Gram +<br>Bactéries Gram – | Almagro Armenteros, Tsirigos, et al. 2019 |
| TargetP | 2.0 | Plantes<br>« Non plantes » | Almagro Armenteros, Salvatore, et al. 2019 |
| DeepSig | 1.0 | Eucaryotes<br>Bactéries Gram +<br>Bactéries Gram – | Savojardo et al. 2018 |
| | *Prédiction de la localisation subcellulaire* | | |
| DeepLoc | 1.0 | Eucaryotes | Almagro Armenteros et al. 2017 |
| MULocDeep | | Eucaryotes | Jiang et al. 2020 |
| | *Prédiction de la topologie* | | |
| TOPCONS | 2.0 | Eucaryotes<br>Archées<br>Procaryotes | Tsirigos et al. 2015 |
| | *Prédiction du sécrétome* | | |
| SecretSanta | 0.99.0 | Eucaryotes<br>Archées<br>Bactéries Gram +<br>Bactéries Gram – | Gogleva et al. 2018 |

**Tableau 4 | Ensemble des organismes pris en charge par les différents outils de prédiction sélectionnés.** La version de MULocDeep n'a pas été renseignée.

Le troisième point est l'évaluation des outils. De nouveaux outils seront prochainement intégrés au workflow. Dans ce rapport, seuls les outils SignalP, TargetP et DeepLoc ont été testés mais d'autres outils de prédictions ont été identifiés : DeepSig (Savojardo et al. 2018) pour la prédiction du peptide signal et d'un domaine transmembranaire, SecretSanta (Gogleva et al. 2018) pour la prédiction des protéines du sécrétome, TOPCONS2 (Tsirigos et al. 2015) pour la prédiction du peptide signal et de la topologie consensus des protéines membranaires et MULocDeep (Jiang et al. 2020) pour la prédiction de la localisation subcellulaire. Ainsi, il est nécessaire de mettre en place une veille et une évaluation régulière des outils de prédiction du peptide signal et de la localisation subcellulaire. Les outils d'extraction et de comparaison des données devront également être mis à jour afin de pouvoir traiter les résultats. Cette évaluation s'appliquera également aux résultats actuels.

Le quatrième et dernier point à améliorer est la normalisation de certains arguments tels que la taxonomie de l'étude : par exemple, pour *Homo sapiens*, la taxonomie est désignée par *hsapiens* pour le script *gConvert.py* alors qu'elle est entrée sous *human* pour le script *EntrezDirect.sh*. De plus, cette différence de désignation est retrouvée pour les outils de prédiction (Tableau 4). Il est donc nécessaire de développer un dictionnaire qui retrouverait les arguments selon l'outil.

Actuellement, le workflow sous Snakemake contient l'intégralité des étapes écrites sous forme de règles dans le fichier *snakefile* et est en développement. Les entrées et les sorties multiples sont à gérer, comme par exemple avec l'outil *DataToolExtract.py* qui écrit une première liste d'identifiants appartenant à la classe *Matrice extracellulaire* puis une deuxième liste d'identifiants appartenant à la classe *Membrane cellulaire*. De plus, outre les règles écrites dans le fichier *snakefile*, Snakemake requiert le fichier *config.yml*, qui permet

notamment de gérer les environnements de travail. Les outils requièrent différents modules et, dans certains cas, ils utilisent des modules de différentes versions : c'est le cas de l'outil DeepLoc, qui utilise la version 2.7.9 de Python, tandis que l'intégralité des scripts Python ont été développés sous la version 3.8.5 en local et exécutés sous la version 3.7.1 sur le cluster HPC2 (Annexe 4).

Ce workflow a été développé dans le but de répondre à la problématique qu'aucun outil bioinformatique ou workflow unique ne détermine de manière automatique les PPIs entre le surfaceome et le sécrétome. Au regard de l'étude de l'équipe BIOMARQUEURS (Bonnet et al. 2020), cet outil intervient au niveau de l'étape de l'identification du sécrétome et du surfaceome. Cependant contrairement à cette étude, notre workflow ne permet pas d'associer nos identifiants à des processus biologiques. Ce point est important à définir, notamment pour identifier de quels tissus proviennent les protéines. Enfin, il serait intéressant d'analyser les données de cette étude avec le workflow, afin de comparer les résultats.

Ainsi, le workflow Talkmine produit actuellement des résultats préliminaires en prédisant les PPIs entre le sécrétome et le surfaceome de deux tissus biologiques. Il est développé dans le cadre d'une application à *Bos taurus*, afin de déterminer les PPIs entre le tissu musculaire et le tissu adipeux. Cependant, il est applicable à d'autres espèces pluricellulaires, d'autres types cellulaires et surtout à différents domaines d'application qui concerne les eucaryotes, notamment la recherche biomédicale et alimentaire. Le développement et l'amélioration de ce workflow sera poursuivi dans les semaines qui suivent et fera l'objet d'un poster à l'évènement JOBIM 2021[22], dont le résumé est fourni en Annexe 8. L'objectif ultime est de rendre ce workflow disponible pour la communauté AuBi.

22 Site de l'évènement JOBIM 2021 : **https://jobim2021.sciencesconf.org/**

**Bibliographie**

Almagro Armenteros JJ, Salvatore M, Emanuelsson O, Winther O, von Heijne G, Elofsson A, Nielsen H. 2019. Detecting sequence signals in targeting peptides using deep learning. *Life Sci. Alliance* 2:e201900429.

Almagro Armenteros JJ, Sønderby CK, Sønderby SK, Nielsen H, Winther O. 2017. DeepLoc: prediction of protein subcellular localization using deep learning.Hancock J, editor. *Bioinformatics* 33:3387–3395.

Almagro Armenteros JJ, Tsirigos KD, Sønderby CK, Petersen TN, Winther O, Brunak S, von Heijne G, Nielsen H. 2019. SignalP 5.0 improves signal peptide predictions using deep neural networks. *Nat. Biotechnol.* 37:420–423.

Aranda B, Blankenburg H, Kerrien S, Brinkman FSL, Ceol A, Chautard E, Dana JM, De Las Rivas J, Dumousseau M, Galeota E, et al. 2011. PSICQUIC and PSISCORE: accessing and scoring molecular interactions. *Nat. Methods* 8:528–529.

Bausch-Fluck D, Goldmann U, Müller S, van Oostrum M, Müller M, Schubert OT, Wollscheid B. 2018. The in silico human surfaceome. *Proc. Natl. Acad. Sci. U. S. A.* 115:E10988–E10997.

Bonnet M, Kaspric N, Vonnahme K, Viala D, Chambon C, Picard B. 2020. Prediction of the Secretome and the Surfaceome: A Strategy to Decipher the Crosstalk between Adipose Tissue and Muscle during Fetal Growth. *Int. J. Mol. Sci.* 21:4375.

Bonnet M, Tournayre J, Cassar-Malek I. 2016. Integrated data mining of transcriptomic and proteomic datasets to predict the secretome of adipose tissue and muscle in ruminants. *Mol. Biosyst.* 12:2722–2734.

Buchmann JP, Holmes EC. 2019. Entrezpy: a Python library to dynamically interact with the NCBI Entrez databases.Wren J, editor. *Bioinformatics* 35:4511–4514.

Caccia D, Dugo M, Callari M, Bongarzone I. 2013. Bioinformatics tools for secretome analysis. *Biochim. Biophys. Acta BBA - Proteins Proteomics* 1834:2442–2453.

Chen JY, Pandey R, Nguyen TM. 2017. HAPPI-2: a Comprehensive and High-quality Map of Human Annotated and Predicted Protein Interactions. *BMC Genomics* 18:182.

Cohen MJ, Chirico WJ, Lipke PN. 2020. Through the back door: Unconventional protein secretion. *Cell Surf.* 6:100045.

Dimou E, Nickel W. 2018. Unconventional mechanisms of eukaryotic protein secretion. *Curr. Biol.* 28:R406–R410.

Gee HY, Kim J, Lee MG. 2018. Unconventional secretion of transmembrane proteins. *Semin. Cell Dev. Biol.* 83:59–66.

Gogleva A, Drost H-G, Schornack S. 2018. SecretSanta: flexible pipelines for functional secretome prediction.Hancock J, editor. *Bioinformatics* 34:2295–2296.

Guo Y, Sirkis DW, Schekman R. 2014. Protein Sorting at the trans-Golgi Network. *Annu. Rev. Cell Dev. Biol.* 30:169–206.

Jiang Y, Wang D, Yao Y, Eubel H, Künzler P, Møller I, Xu D. 2020. MULocDeep: A deep-learning framework for protein subcellular and suborganellar localization prediction with residue-level interpretation. In Review Available from: https://dx.doi.org/10.21203/RS.3.RS-40744/V1

Kans J. 2013. Entrez Direct: E-utilities on the Unix Command Line. In: Entrez Programming Utilities Help [Internet]. National Center for Biotechnology Information (US). Available from: https://www.ncbi.nlm.nih.gov/books/NBK179288/

Kaspric N, Picard B, Reichstadt M, Tournayre J, Bonnet M. 2015. ProteINSIDE to Easily Investigate Proteomics Data from Ruminants: Application to Mine Proteome of Adipose and Muscle Tissues in Bovine Foetuses.Lisacek F, editor. *PLoS One* 10:e0128086.

Koh GCKW, Porras P, Aranda B, Hermjakob H, Orchard SE. 2012. Analyzing Protein–Protein Interaction Networks. *J. Proteome Res.* 11:2014–2031.

Kotlyar M, Rossos AEM, Jurisica I. 2017. Prediction of Protein-Protein Interactions. *Curr. Protoc. Bioinforma.* 60:8.2.1-8.2.14.

Kotsiantis SB. 2007. Supervised Machine Learning: A Review of Classification Techniques. *Informatica* 31:249–268.

Krogh A, Larsson B, von Heijne G, Sonnhammer ELL. 2001. Predicting transmembrane protein topology with a hidden markov model: application to complete genomes. *J. Mol. Biol.* 305:567–580.

Lebreton S, Zurzolo C, Paladino S. 2018. Organization of GPI-anchored proteins at the cell surface and its physiopathological relevance. *Crit. Rev. Biochem. Mol. Biol.* 53:403–419.

Min S, Lee B, Yoon S. 2017. Deep learning in bioinformatics. *Brief. Bioinform.* 18:851–869.

Miura K. 2018. An Overview of Current Methods to Confirm Protein- Protein Interactions. *Protein Pept. Lett.* 25:728–733.

Mölder F, Jablonski KP, Letcher B, Hall MB, Tomkins-Tinch CH, Sochat V, Forster J, Lee S, Twardziok SO, Kanitz A, et al. 2021. Sustainable data analysis with Snakemake. *F1000Research* 10:33.

Muniz M, Zurzolo C. 2014. Sorting of GPI-anchored proteins from yeast to mammals - common pathways at different sites? *J. Cell Sci.* 127:2793–2801.

Ng F, Tang BL. 2016. Unconventional Protein Secretion in Animal Cells. In: Pompa A, De Marchis F, editors. Unconventional Protein Secretion. Vol. 1459. Methods in Molecular Biology. New York, NY: Springer New York. p. 31–46.

Nickel W. 2003. The mystery of nonclassical protein secretion: A current view on cargo proteins and potential export routes. *Eur. J. Biochem.* 270:2109–2119.

Nielsen H, Tsirigos KD, Brunak S, von Heijne G. 2019. A Brief History of Protein Sorting Prediction. *Protein J.* 38:200–216.

Orchard S, Ammari M, Aranda B, Breuza L, Briganti L, Broackes-Carter F, Campbell NH, Chavali G, Chen C, del-Toro N, et al. 2014. The MIntAct project—IntAct as a common curation platform for 11 molecular interaction databases. *Nucleic Acids Res.* 42:D358–D363.

Peters C, Tsirigos KD, Shu N, Elofsson A. 2016. Improved topology prediction using the terminal hydrophobic helices rule. *Bioinformatics* 32:1158–1162.

Pierleoni A, Indio V, Savojardo C, Fariselli P, Martelli PL, Casadio R. 2011. MemPype: a pipeline for the annotation of eukaryotic membrane proteins. *Nucleic Acids Res.* 39:W375–W380.

Pierleoni A, Martelli P, Casadio R. 2008. PredGPI: a GPI-anchor predictor. *BMC Bioinformatics* 9:392.

Pierleoni A, Martelli PL, Casadio R. 2011. MemLoci: predicting subcellular localization of membrane proteins in eukaryotes. *Bioinformatics* 27:1224–1230.

Raudvere U, Kolberg L, Kuzmin I, Arak T, Adler P, Peterson H, Vilo J. 2019. g:Profiler: a web server for functional enrichment analysis and conversions of gene lists (2019 update). *Nucleic Acids Res.* 47:W191–W198.

Savojardo C, Martelli Pier Luigi, Fariselli P, Casadio R. 2018. DeepSig: deep learning improves signal peptide detection in proteins.Valencia A, editor. *Bioinformatics* 34:1690–1696.

Savojardo C, Martelli Pier Luigi, Fariselli P, Profiti G, Casadio R. 2018. BUSCA: an integrative web server to predict subcellular localization of proteins. *Nucleic Acids Res.* 46:W459–W466.

Sayers E. 2010. A General Introduction to the E-utilities. In: Entrez Programming Utilities Help [Internet]. Available from: https://www.ncbi.nlm.nih.gov/books/NBK25497/

Szklarczyk D, Gable AL, Lyon D, Junge A, Wyder S, Huerta-Cepas J, Simonovic M, Doncheva NT, Morris JH, Bork P, et al. 2019. STRING v11: protein–protein association networks with increased coverage, supporting functional discovery in genome-wide experimental datasets. *Nucleic Acids Res.* 47:D607–D613.

Tanwar H, George Priya Doss C. 2018. Computational Resources for Predicting Protein–Protein Interactions. In: Advances in Protein Chemistry and Structural Biology. Vol. 110. Elsevier. p. 251–275.

The IMEx Consortium Curators, del-Toro N, Duesbury M, Koch M, Perfetto L, Shrivastava A, Ochoa D, Wagih O, Piñero J, Kotlyar M, et al. 2019. Capturing variation impact on molecular interactions in the IMEx Consortium mutations data set. *Nat. Commun.* 10:10.

The UniProt Consortium, Bateman A, Martin M-J, Orchard S, Magrane M, Agivetova R, Ahmad S, Alpi E, Bowler-Barnett EH, Britto R, et al. 2021. UniProt: the universal protein knowledgebase in 2021. *Nucleic Acids Res.* 49:D480–D489.

Tsirigos KD, Elofsson A, Bagos PG. 2016. PRED-TMBB2: improved topology prediction and detection of beta-barrel outer membrane proteins. *Bioinformatics* 32:i665–i671.

Tsirigos KD, Govindarajan S, Bassot C, Västermark Å, Lamb J, Shu N, Elofsson A. 2018. Topology of membrane proteins — predictions, limitations and variations. *Curr. Opin. Struct. Biol.* 50:9–17.

Tsirigos KD, Peters C, Shu N, Käll L, Elofsson A. 2015. The TOPCONS web server for consensus prediction of membrane protein topology and signal peptides. *Nucleic Acids Res.* 43:W401–W407.

Viklund H, Elofsson A. 2008. OCTOPUS: improving topology prediction by two-track ANN-based preference scores and an extended topological grammar. *Bioinformatics* 24:1662–1668.

Whitley P, Mingarro I. 2014. Stitching proteins into membranes, not sew simple. *Biol. Chem.* 395:1417–1424.

Zhao L, Poschmann G, Waldera-Lupa D, Rafiee N, Kollmann M, Stühler K. 2019. OutCyte: a novel tool for predicting unconventional protein secretion. *Sci. Rep.* 9:19448.

# Annexes

## Annexe 1 : Méthodes à haut débit pour la mise en évidence d'une interaction entre deux protéines

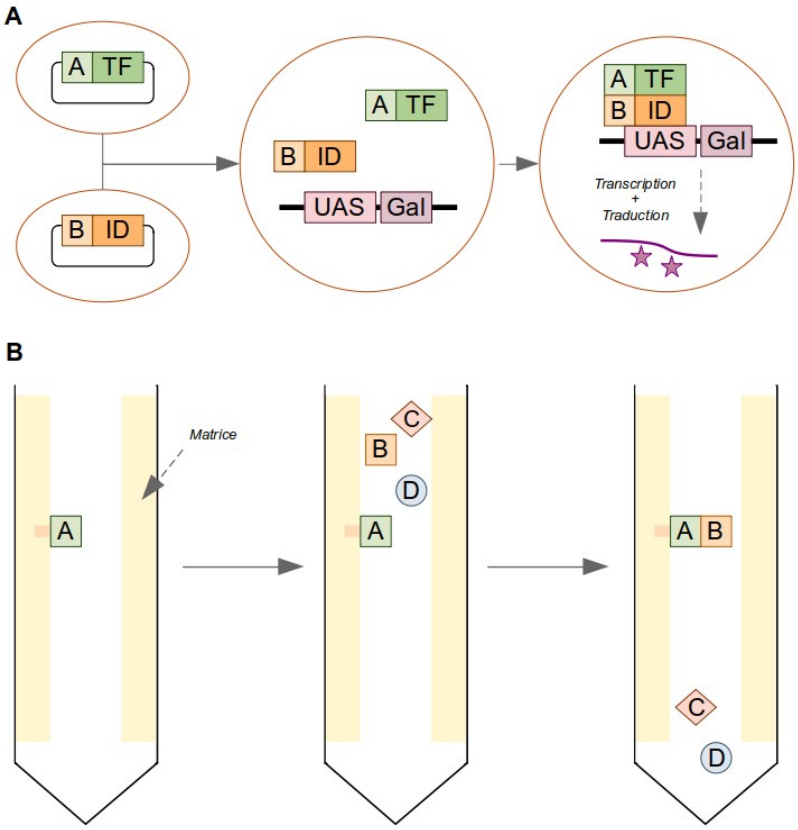


**Figure Annexe 1 | Schéma de deux techniques à haut débit permettant l'identification des PPIs.** (A) Technique Y2H : Deux constructions sont ajoutées à une cellule transformée. Seule l'interaction entre A et B entraîne la transcription et la traduction du gène rapporteur. (B) Technique AP-MS : Une protéine est capturée par la matrice. Après ajout d'un mélange de protéines, seules les protéines interagissant avec A resteront dans la colonne. A, B, C, D, protéines ; TF, facteur de transcription ; ID, domaine d'interaction permettant la transcription de la β-Galactosidase ; GAL, gène codant la β-Galactosidase ; UAS, séquence activatrice localisée en amont.

Soient deux protéines A et B interagissant ensemble.

Se basant sur la complémentation de deux parties d'un facteur de transcription, la méthode de Y2H se déroule de la manière suivante (Figure Annexe 1A). Dans un premier temps, deux constructions géniques sont réalisées : la première construction contient la protéine A et le domaine de liaison à l'ADN du facteur de transcription, tandis que la deuxième construction contient la protéine B et le domaine d'activation du facteur de transcription. Les constructions sont ajoutées par fusion à une cellule transformée (comme la levure par exemple), contenant un gène rapporteur, telle que la β-galactosidase (Koh et al. 2012). Le domaine de liaison à l'ADN se liera à une séquence localisée en amont du gène rapporteur, appelée la séquence activatrice en amont. Si les protéines interagissent, le domaine d'activation va permettre le recrutement du facteur de transcription et ainsi induire la transcription du gène rapporteur. Une autre variante consiste à fusionner la protéine A à un domaine de liaison de l'ADN de GAL4, un facteur de transcription régulateur de l'expression des gènes induits par le Galactose, et la protéine B au domaine d'activation d'ADN de GAL4 (Miura 2018). Si les levures sont ajoutées dans un milieu sélectif, lorsque la protéine A se lie à la protéine B, les cellules prolifèrent. *A contrario*, si elles n'interagissent pas, les cellules ne prolifèrent pas.

Quant à la méthode AP-MS (Figure Annexe 1B), elle se divise en deux grandes étapes. Dans un premier temps, la protéine A est capturée sur une matrice par affinité. Dans un deuxième temps, un mélange d'interacteurs potentiels est injecté dans la colonne et passe au travers de la matrice. Seules les protéines interagissant avec la protéine A sont retenues. Après plusieurs étapes de purification, les protéines sont séparées et identifiées par spectrométrie de masse (Koh et al. 2012, Miura 2018).

## Annexe 2 : Apprentissage profond (ou *deep learning*) en Bioinformatique – Application à la prédiction et architectures

Dans cette section d'annexe, nous nous intéressons aux algorithmes utilisés dans le cadre de l'apprentissage profond ou *deep learning* puis à l'architecture des outils de prédiction.

### Annexe 2.1 : Généralités sur les architectures de *deep learning*

Le *deep learning* ne se limite pas uniquement à la prédiction mais il est aussi utilisé dans l'imagerie et le traitement des images (Min et al. 2017). Son application peut se résumer en deux étapes, qui sont la construction et l'apprentissage d'une architecture ou d'un algorithme. Dans la revue de Min et al. (2017), les architectures sont classées en quatre catégories : les réseaux neuronaux profonds (DNNs), les réseaux neuronaux convolutifs (CNNs), les réseaux neuronaux récurrents (RNNs) et les algorithmes émergents. Les outils présentés dans ce rapport utilisent des CNNs et des RNNs.

Les CNNs sont des architectures principalement utilisés dans la reconnaissance d'images et sont constitués de trois couches différentes : des couches de convolution, des couches non linéaires et des couches de regroupement (Figure Annexe 2.1.a) (Min et al. 2017). La première couche, la couche de convolution, utilise des filtres issus de la phase d'apprentissage sur des cartes de caractéristiques, qui sont des regroupements des données. Chaque carte est spécifique d'un filtre et, dans le cas de la prédiction, un filtre peut correspondre à un motif d'acides aminés (Savojardo et al. 2018). Ensuite, la couche linéaire va réaliser un sous-échantillonnage maximal ou moyens des régions qui ne se chevauchent pas dans les cartes caractéristiques. Ce processus permet d'identifier des caractéristiques plus complexes. Enfin, la couche de regroupement va combiner les différentes couches afin d'en extraire des

caractéristiques, telle que la classe à laquelle appartient l'objet analysé (Min et al. 2017, Savojardo et al. 2018).

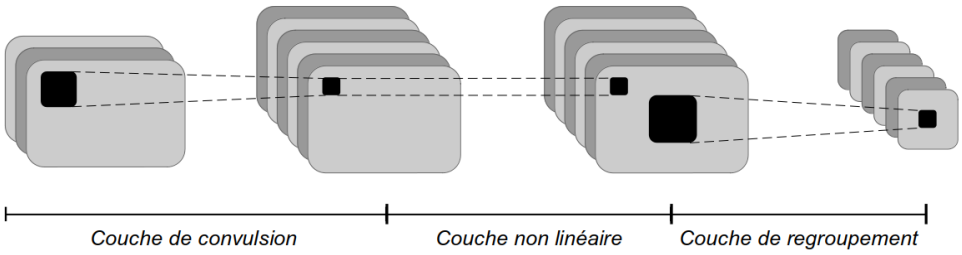


**Figure Annexe 2.1.a | Schéma d'un réseau neuronal convolutif (Min et al. 2017, traduite).** La structure de base consiste en une couche de convolution, une couche non linéaire et une couche de regroupement. La couche de convolution utilise des filtres issus de la phase d'apprentissage pour obtenir plusieurs cartes de filtres détectant les caractéristiques de bas niveau, la couche non linéaire extrait des caractéristiques plus complexe puis la couche de regroupement les combine dans une caractéristique de niveau supérieur.

Utilisés dans l'analyse de séquences, les RNNs sont des architectures composées d'une entrée, d'une sortie et d'une unité cachée, qui possède une connexion cyclique (Figure Annexe 2.1.b) (Min et al. 2017). Si cette organisation est « dépliée dans le temps », il s'agit d'une succession d'unités cachées avec chacune son entrée et sa sortie, qui reçoivent également les données des unités les précédant et qui les envoient dans les unités suivantes. Des variantes existent au sein de cette catégories : les RNNs bidirectionnels (BiRNNs), qui possèdent des unités cachées pouvant recevoir des données passées ou futures (Figure Annexe 2.1.c) (Min et al. 2017), et les RNNs à courte et longue mémoire (LSTM), dont les unités cachées possèdent des cellules de mémoire et sont appelées des unités récurrentes fermées (GRU), ce qui leur permettent de « retenir » des informations après plusieurs cycles (Almagro Armenteros et al. 2019, Min et al. 2017). La combinaison de ces deux catégories donne des

RNNs bidirectionnels à courte et longue mémoire (BiLTSM), qui sont des réseaux pouvant analyser des données dans les deux sens (par exemple, lire une séquence protéique de son extrémité N-terminale à son extrémité C-terminale et inversement) tout en gardant en « mémoire » les informations des motifs précédents.

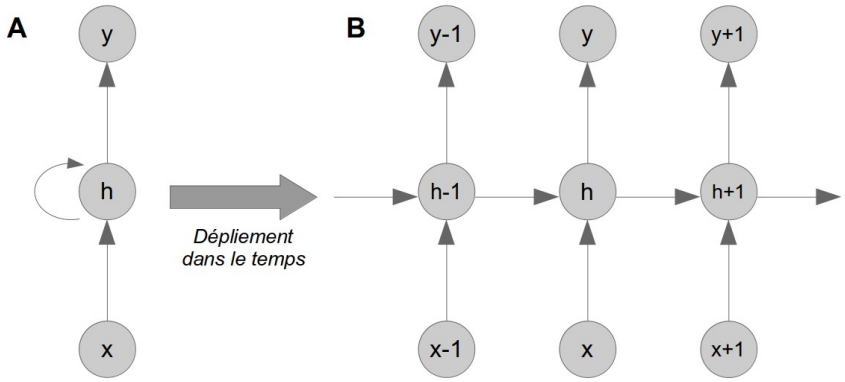


**Figure Annexe 2.1.b | Schéma d'un réseau neuronal récurent (Min et al. 2017, traduite).** (A) x est l'entrée du réseau, y la sortie et h l'unité cachée. (B) Si le réseau est « déplié dans le temps », il peut être représenté comme une succession d'unités cachées avec chacune son entrée et sa sortie, qui reçoivent les données des unités les précédant (h-1) et qui les envoient dans les unités suivantes (h+1).

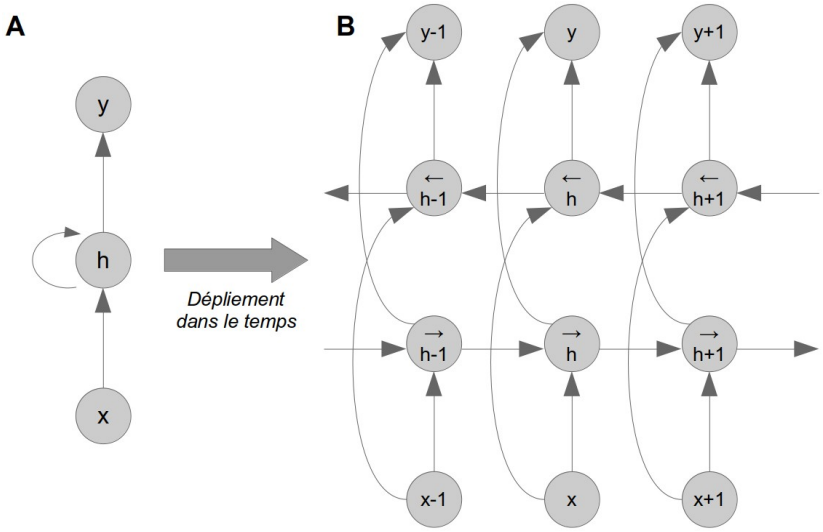


**Figure Annexe 2.1.c | Schéma d'un réseau neuronal récurent bidirectionnel (Min et al. 2017, traduite).** (A) x est l'entrée du réseau, y la sortie et h l'unité cachée. (B) Si le réseau est

« déplié dans le temps », il peut être représenté comme une succession d'unités cachées avec chacune son entrée et sa sortie, qui reçoivent les données des unités les précédent et qui les envoient dans les unités suivantes. Dans le cas d'un réseau bidirectionnel, chaque unité cachée peut recevoir des données de son entrée mais aussi des données d'autres unités, reflétant des données passées (si elles proviennent de h-1) ou des données futures (si elles proviennent de h+1).

## Annexe 2.2 : Architecture de SignalP 5.0

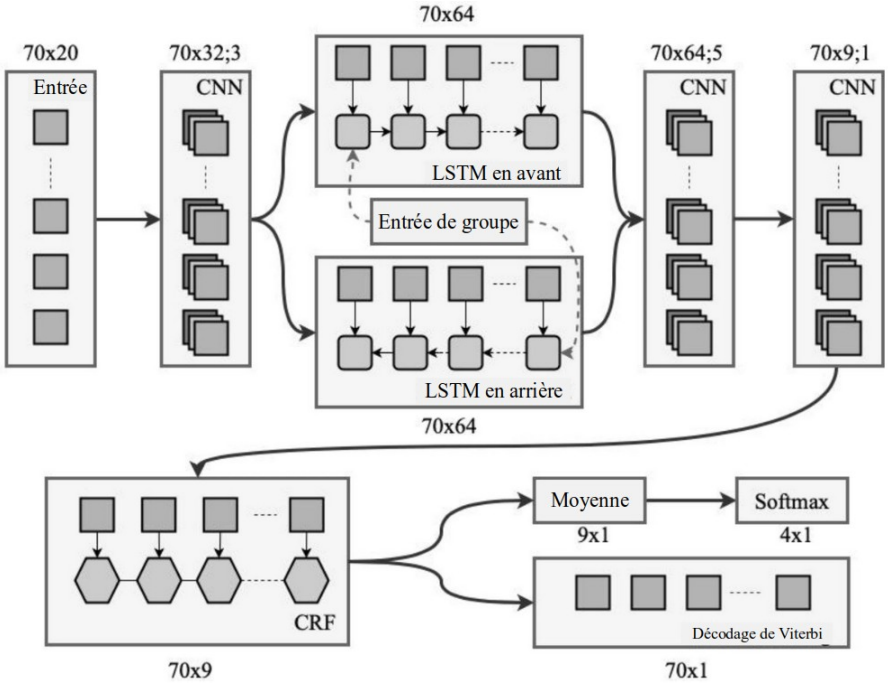


**Figure Annexe 2.2 | Architecture de l'outil SignalP 5.0 (Almagro Armenteros, Tsirigos, et al. 2019, traduite).** Les 70 acides aminées de la séquence N-terminale sont envoyés dans un CNN afin de capturer les petits motifs puis dans un BiLSTM pour les motifs plus grands. Après avoir été soumis à deux autres CNNs, dont le rôle est similaire au premier, les données sont traitées par un CRF pour établir les valeurs de prédiction. Ces valeurs sont attribuées à la séquence globale, avec une moyenne des probabilités pondérales suivie d'une application d'une fonction softmax, et à chacun des acides aminés, grâce au décodage avec l'algorithme de

Viterbi. Entrée 70 x 20, 70 positions décomposées sur 20 canaux en entrée ; CNN 70 x 32 ; 3, réseau neuronal convolutif avec 32 filtres pour une largeur de noyau de 3 ; LSTM en avant et en arrière 70 x 64, réseau neuronal récurrent bidirectionnel à mémoire courte et longue avec 64 unités cachées ; CNN 70 x 64 ; 5, réseau neuronal convolutif avec 64 filtres pour une largeur de noyau de 5 ; CNN 70 x 9 ; 1, réseau neuronal convolutif avec 9 filtres pour une largeur de noyau de 1 ; CRF, champ aléatoire conditionnel ; Moyenne 9 x 1, calcul de la moyenne de probabilités marginales produisant un vecteur 9 x 1 ; Softmax 4 x 1, fonction softmax produisant un vecteur 4 x 1 ; Décodage de Viterbi 70 x 1, algorithme de décodage de Viterbi produisant un vecteur 70 x 1.

SignalP 5.0 possède un modèle d'apprentissage avec trois principaux composants (Figure Annexe 2.2) : trois CNNs pour capturer les petits motifs, un BiLSTM pour capturer des motifs à longue portée et un CRF pour la classification (Almagro Armenteros, Tsirigos, et al. 2019). Il a été entraîné avec des séquences N-terminales de 70 acides aminés décomposés sur 20 canaux en entrée, correspondant aux 20 acides aminés.

Le premier CNN utilise 32 filtres pour une largeur de noyau de 3[23]. Il est suivi du BiLSTM, qui possède 64 unités cachées. Les données sortantes des composantes avant et arrière du LSTM sont envoyées dans deux CNNs, le premier possédant 64 filtres pour une largeur de noyau de 5 et le deuxième 9 filtres pour une largeur de noyau de 1. Enfin, les résultats sont envoyés dans le CRF. Durant cette dernière étape, il détermine à la fois la classification de la séquence globale en utilisant l'algorithme forward-backward, un algorithme permettant de déterminer la probabilité d'une séquence dans un contexte de modèle caché de Markov, et la classification de chaque acide aminé avec un décodage avec l'algorithme de Viterbi. Ce

23 Dans un CNN, un noyau de convolution correspond à un ensemble de neurones possédant les mêmes paramètres. Dans le cas annoté, un noyau de largeur 3 correspond à un ensemble de 3 neurones.

dernier algorithme produit un vecteur 70 x 1, qui correspond aux valeurs de probabilité pour les 70 acides aminés donnés en entrée. Le résultat de la séquence globale est déterminé via une moyenne des probabilités marginales produisant un vecteur 9 x 1, soit un ensemble de 9 valeurs correspondant à 9 classes (Signal Sec/SPI, signal Tat/SPI, signal Sec/SPII, région externe, région interne, domaine TM dans le sens entrée-sortie, domaine TM sortie-entrée, site de clivage Sec/SPI, site de clivage Tat/SPI et site de clivage Sec/SPII), qui est ensuite linéarisé par la fonction softmax. Cette fonction permet de faire la classification en classes multiples en produisant un vecteur 4 x 1, soit un ensemble de 4 valeurs qui correspondent aux valeurs de prédiction des 4 classes (Sec/SPI, Tat/SPI, Sec/SPII, *Others*) (Almagro Armenteros et al. 2019).

**Annexe 2.3 : Architecture de TargetP 2.0**

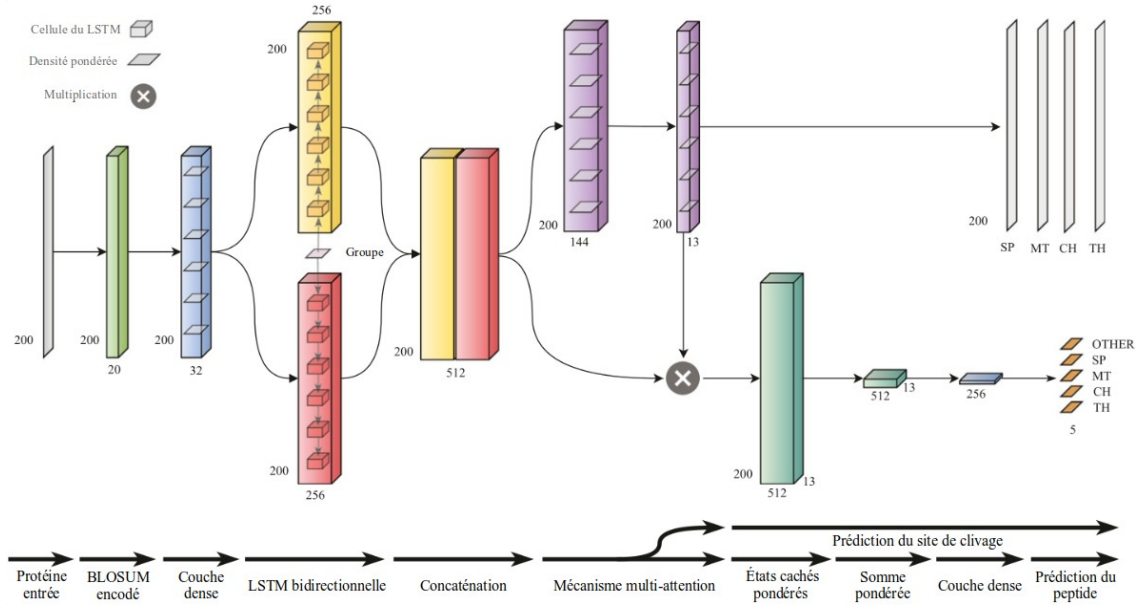


**Figure Annexe 2.3 | Architecture de l'outil TargetP 2.0 (Almagro Armenteros, Salvatore, et al. 2019, traduite).** Après un encodage des 200 premiers acides aminés par la matrice BLOSUM62, ils sont transformés par une couche dense avant d'être envoyé dans le BiLSTM.

Après une concaténation des résultats provenant des deux parties du LSTM, la matrice d'attention est calculée et 13 vecteurs sont extraits. Quatre sont utilisés pour la prédiction du site de clivage. Les autres sont utilisés afin d'encoder la séquence entière puis, après un traitement pour le transformer en un vecteur, le peptide est prédit grâce à une fonction softmax. BLOSUM, matrice de substitution de bloc ; LSTM, réseau neuronal récurrent à mémoire courte et longue ; SP, peptide signal ; MT, peptide de transit mitochondrial ; CH, peptide de transit pour le chloroplaste ; TH, peptide de transit luminal ; Other, protéine ne possédant aucun des peptides mentionnés précédemment.

Le modèle de TargetP 2.0 possède deux composants clés (Figure Annexe 2.3) : le BiLSTM et le mécanisme de multi-attention, permettant tous deux de prédire le type de peptide et la position du site de clivage (Almagro Armenteros, Salvatore, et al. 2019).

L'entrée de cet outil est les 200 premiers acides aminés d'une protéine, qui sont encodés en utilisant la matrice de substitution BLOSUM62 de taille 200 x 20[24]. La première couche du modèle, qui est une couche entièrement connectée, effectue une transformation de chaque caractéristique d'acide aminé donné en entré avec 32 unités cachées. Ces données sont envoyée dans le BiLSTM, composé de 256 unités cachées et qui va analyser dans les deux directions. Chaque sortie est concaténée dans une matrice de taille 200 x 512. Cette matrice est utilisée pour calculer la matrice multi-attention. Cette matrice obtenue est de 144 unités et produit un total de 13 vecteurs d'attention. Quatre vecteurs sont utilisés pour prédire les différentes positions du site de clivage pour le peptide signal, le peptide de transit mitochondrial, le peptide de transit du chloroplaste et le peptide de transit luminal. Cette matrice est également utilisée pour encoder la séquence entière dans une matrice de contexte

24 La taille d'une matrice s'écrit de la manière suivante : *ligne x colonne*. Ici, la matrice contient 200 lignes et 20 colonnes.

de taille 512 x 13. Elle est traitée par une couche entièrement connectée de 256 unités afin de la résumer en un vecteur. Ce vecteur est envoyé dans la couche de sortie, constitué d'une fonction softmax qui va permettre de classer la séquence protéique en produisant un vecteur de taille 5, correspondant aux valeurs de prédiction des 5 classes (peptide signal, peptide de transit mitochondrial, peptide de transit du chloroplaste, peptide de transit luminal et *other*, qui désigne les séquences protéiques ne possédant aucun des peptides mentionnés précédemment).

**Annexe 2.4 : Architecture de DeepLoc 1.0**

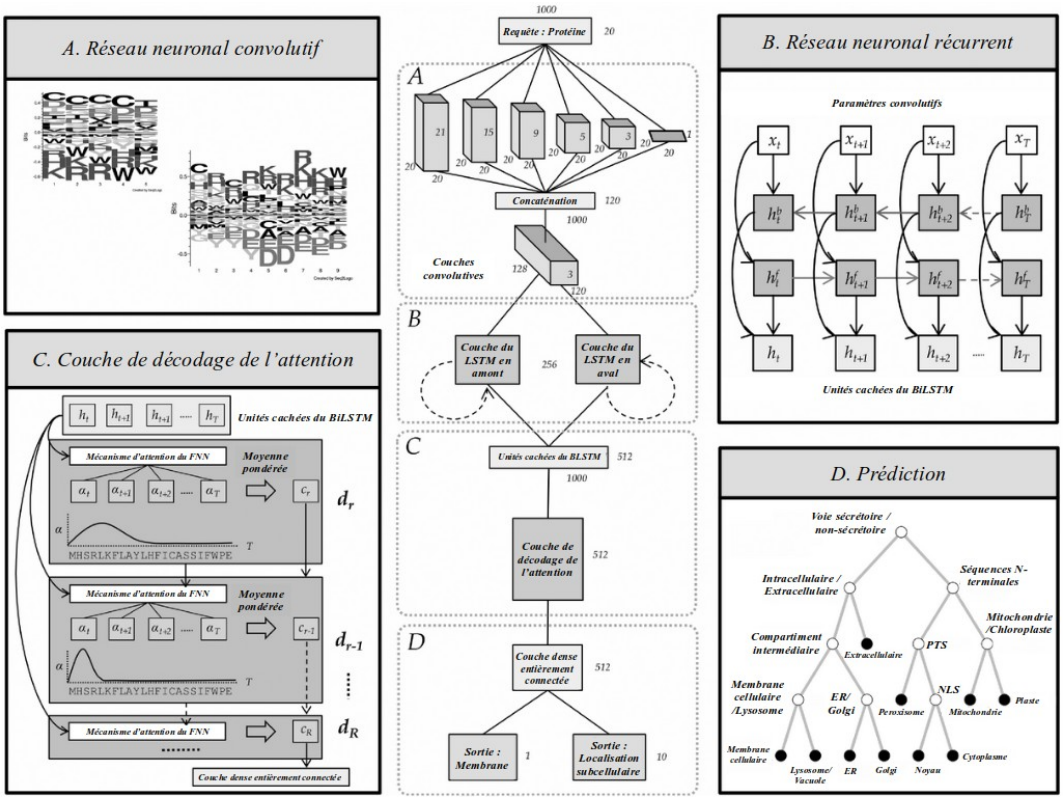


**Figure Annexe 2.4 | Architecture de l'outil DeepLoc (Almagro Armenteros et al. 2017, traduite).** L'entrée est ici une séquence protéique composée de 1000 acides aminés. (A) Le CNN extrait des informations des motifs en utilisant différentes tailles de ces motifs. (B) Le BiLSTM analyse la séquence dans les deux sens, pour extraire les dépendances spatiales entre

les acides aminés. (C) Le mécanisme d'attention attribue une importance plus élevée aux acides aminés qui sont pertinents pour la prédiction. (D) Toutes les informations recueillies à partir de la séquence protéique sont transmises à une fonction softmax et à un arbre hiérarchique pour calculer et définir la prédiction finale. BiLSTM, réseau neuronal récurrent à mémoire courte et longue bidirectionnel ; LSTM, réseau neuronal récurrent à mémoire courte et longue ; FNN, réseau de neurones de rétroaction ; PTS, signal de ciblage peroxysomal ; NLS, signal de localisation nucléaire ; ER, réticulum endoplasmique.

L'architecture de DeepLoc est composée de quatre principales composantes (Figure Annexe 2.4) : un CNN pour détecter les motifs, un BiLSTM pour analyser la séquence dans les deux sens, un mécanisme d'attention pour se focaliser sur les motifs d'intérêt et un arbre de décision pour déterminer la prédiction (Almagro Armenteros et al. 2017).

L'entrée est une matrice avec pour ligne la longueur de la séquence (ici, 1000) et pour largeur les 20 acides aminés standards. Dans un premier temps, le CNN extrait les informations sur les motifs à l'aide de 120 filtres de différentes tailles (avec 20 filtres pour les pas[25] de 1, 3, 5, 9, 15 et 21). Les résultats sont concaténés dans une carte de caractéristiques de taille 1000 x 120 puis envoyées dans une couche convolutive composée de 128 filtres, d'un pas de 3 et d'une marge de 120, afin d'obtenir une nouvelle carte de taille 1000 x 128. Cette carte est traitée par le BiLSTM, composée de 256 unités cachées dans les deux sens et donnant une sortie concaténée de dimension 1000 x 512. La couche de décodage d'attention utilise un autre LSTM avec 512 unités cachées à travers 10 étapes de décodage et un réseau de neurones de rétroaction (FFN). Les résultats sont traitées par une couche dense finale composée de

25 Dans un CNN, une couche de convolution est définie par trois paramètres : la profondeur, qui correspond au nombre de neurones de la couche ou de filtres (qui sont un ensemble de neurones avec les mêmes paramètres), le pas, qui désigne combien de colonnes vont être traités par les noyaux, et la marge, qui permet de contrôler le volume des données de sortie.

512 unités, avant d'être envoyés dans deux couches de sortie : une couche permettant de prédire si la protéine est liée à la membrane ou soluble, composée d'une unité, et une couche permettant de prédire la localisation subcellulaire, composée de 10 unités et utilisant un arbre de décision hiérarchique. Les valeurs de probabilité sont également déterminées par l'utilisation d'une fonction softmax.

## Annexe 3 : Jeu de données test

| Entrée UniProt | Entrée | Nom de la protéine | Localisation subcellulaire | GPI ? TM ? |
|---|---|---|---|---|
| Q02218 | ODO1_HUMAN | 2-oxoglutarate dehydrogenase, mitochondrial | Matrice mitochondriale | |
| P16118 | F261_HUMAN | 6-phosphofructo-2-kinase/fructose-2,6-bisphosphatase 1 (PFK/FBPase 1) | Cytosol | |
| O60825 | F262_HUMAN | 6-phosphofructo-2-kinase/fructose-2,6-bisphosphatase 2 (PFK/FBPase 2) | Cytosol | |
| Q16875 | F263_HUMAN | 6-phosphofructo-2-kinase/fructose-2,6-bisphosphatase 3 (PFK/FBPase 3) | Cytosol | |
| Q16877 | F264_HUMAN | 6-phosphofructo-2-kinase/fructose-2,6-bisphosphatase 4 (PFK/FBPase 4) | Cytosol | |
| Q01813 | PFKAP_HUMAN | ATP-dependent 6-phosphofructokinase, platelet type (ATP-PFK) (PFK-P) | Cytoplasme | |
| P17858 | PFKAL_HUMAN | ATP-dependent 6-phosphofructokinase, liver type (ATP-PFK) (PFK-L) | Cytoplasme | |
| P08237 | PFKAM_HUMAN | ATP-dependent 6-phosphofructokinase, muscle type (ATP-PFK) (PFK-M) | Cytoplasme | |
| Q15848 | ADIPO_HUMAN | Adiponectin (30 kDa adipocyte complement-related protein) (ACRP30) | Sécrétée | |
| Q7Z4H4 | ADM2_HUMAN | Protein ADM2 (Intermedin) | Sécrétée | |
| P35318 | ADML_HUMAN | Pro-adrenomedullin | Sécrétée | |
| Q9BRR6 | ADPGK_HUMAN | ADP-dependent glucokinase (ADP-GK) (ADPGK) | Sécrétée | |
| P06733 | ENOA_HUMAN | Alpha-enolase | Cytoplasme | |
| P01160 | ANF_HUMAN | Natriuretic peptides A (Atrial natriuretic factor prohormone) (proANF) | Sécrétée | |

| | | | | |
|---|---|---|---|---|
| P16860 | ANFB_HUMAN | Natriuretic peptides B (Brain natriuretic factor prohormone) (preproBNP) (proBNP) | Sécrétée | |
| Q9ULZ1 | APEL_HUMAN | Apelin (APJ endogenous ligand) | Sécrétée | |
| P13929 | ENOB_HUMAN | Beta-enolase | Cytoplasme | |
| P07738 | PMGE_HUMAN | Bisphosphoglycerate mutase (BPGM) | Cytosol | |
| P0C862 | C1T9A_HUMAN | Complement C1q and tumor necrosis factor-related protein 9A | Sécrétée | |
| P01258 | CALC_HUMAN | Calcitonin | Sécrétée | |
| P06881 | CALCA_HUMAN | Calcitonin gene-related peptide 1 (Alpha-type CGRP) | Sécrétée | |
| P10092 | CALCB_HUMAN | Calcitonin gene-related peptide 2 (Beta-type CGRP) | Sécrétée | |
| P06307 | CCKN_HUMAN | Cholecystokinin (CCK) | Sécrétée | |
| A6NKQ9 | CGB1_HUMAN | Choriogonadotropin subunit beta variant 1 | Sécrétée | |
| Q6NT52 | CGB2_HUMAN | Choriogonadotropin subunit beta variant 2 | Sécrétée | |
| P0DN86 | CGB3_HUMAN | Choriogonadotropin subunit beta 3 (Choriogonadotropin subunit beta) (CG-beta) | Sécrétée | |
| P0DN87 | CGB7_HUMAN | Choriogonadotropin subunit beta 7 | Sécrétée | |
| P01189 | COLI_HUMAN | Pro-opiomelanocortin (POMC) (Corticotropin-lipotropin) | Sécrétée | |
| P53621 | COPA_HUMAN | Coatomer subunit alpha (Alpha-coat protein) (Alpha-COP) | Cytoplasme | |
| P06850 | CRF_HUMAN | Corticoliberin (Corticotropin-releasing factor) (CRF) | Sécrétée | |
| P0DML2 | CSH1_HUMAN | Chorionic somatomammotropin hormone 1 | Sécrétée | |
| P0DML3 | CSH2_HUMAN | Chorionic somatomammotropin hormone 2 | Sécrétée | |
| Q14406 | CSHL_HUMAN | Chorionic somatomammotropin hormone-like 1 | Sécrétée | |
| P10515 | ODP2_HUMAN | Dihydrolipoyllysine-residue | Matrice | |

| | | | | |
|---|---|---|---|---|
| | | acetyltransferase component of pyruvate dehydrogenase complex, mitochondrial | mitochondriale | |
| O75356 | ENTP5_HUMAN | Ectonucleoside triphosphate diphosphohydrolase 5 (NTPDase 5) | Sécrétée | |
| P01588 | EPO_HUMAN | Erythropoietin (Epoetin) | Sécrétée | |
| P04075 | ALDOA_HUMAN | Fructose-bisphosphate aldolase A | Cytoplasme | |
| P05062 | ALDOB_HUMAN | Fructose-bisphosphate aldolase B | Cytoplasme | |
| P09972 | ALDOC_HUMAN | Fructose-bisphosphate aldolase C | Cytosol | |
| P01225 | FSHB_HUMAN | Follitropin subunit beta (Follicle-stimulating hormone beta subunit) (FSH-B) | Sécrétée | |
| P22466 | GALA_HUMAN | Galanin peptides | Sécrétée | |
| Q9UBC7 | GALP_HUMAN | Galanin-like peptide | Sécrétée | |
| P09104 | ENOG_HUMAN | Gamma-enolase | Cytoplasme | |
| P01350 | GAST_HUMAN | Gastrin | Sécrétée | |
| Q9UBU3 | GHRL_HUMAN | Appetite-regulating hormone (Protein M46) | Sécrétée | |
| P09681 | GIP_HUMAN | Gastric inhibitory polypeptide (GIP) (Glucose-dependent insulinotropic polypeptide) (Incretin hormone) | Sécrétée | |
| P01215 | GLHA_HUMAN | Glycoprotein hormones alpha chain | Sécrétée | |
| P01275 | GLUC_HUMAN | Pro-glucagon | Sécrétée | |
| P35557 | HXK4_HUMAN | Hexokinase-4 (HK4) | Cytoplasme | |
| P06744 | G6PI_HUMAN | Glucose-6-phosphate isomerase (GPI) | Cytoplasme | |
| P04406 | G3P_HUMAN | Glyceraldehyde-3-phosphate dehydrogenase (GAPDH) | Cytoplasme | |
| O14556 | G3PT_HUMAN | Glyceraldehyde-3-phosphate dehydrogenase, testis-specific | Cytoplasme | |
| P01148 | GON1_HUMAN | Progonadoliberin-1 (Progonadoliberin I) | Sécrétée | |
| O43555 | GON2_HUMAN | Progonadoliberin-2 | Sécrétée | |

| | | | | |
|---|---|---|---|---|
| | | (Progonadoliberin II) | | |
| Q96T91 | GPHA2_HUMAN | Glycoprotein hormone alpha-2 | Sécrétée | |
| Q86YW7 | GPHB5_HUMAN | Glycoprotein hormone beta-5 | Sécrétée | |
| P81172 | HEPC_HUMAN | Hepcidin (Liver-expressed antimicrobial peptide 1) (LEAP-1) (Putative liver tumor regressor) (PLTR) | Sécrétée | |
| P19367 | HXK1_HUMAN | Hexokinase-1 (HK I) | Membrane externe mitochondriale | |
| P52789 | HXK2_HUMAN | Hexokinase-2 (HK II) | Membrane externe mitochondriale | |
| P52790 | HXK3_HUMAN | Hexokinase-3 (HK III) | Cytosol | |
| P10997 | IAPP_HUMAN | Islet amyloid polypeptide (Amylin) | Sécrétée | |
| P01344 | IGF2_HUMAN | Insulin-like growth factor II (IGF-II) | Sécrétée | |
| P05111 | INHA_HUMAN | Inhibin alpha chain | Sécrétée | |
| P08476 | INHBA_HUMAN | Inhibin beta A chain (Activin beta-A chain) | Sécrétée | |
| P09529 | INHBB_HUMAN | Inhibin beta B chain (Activin beta-B chain) | Sécrétée | |
| P55103 | INHBC_HUMAN | Inhibin beta C chain (Activin beta-C chain) | Sécrétée | |
| P58166 | INHBE_HUMAN | Inhibin beta E chain (Activin beta-E chain) | Sécrétée | |
| P01308 | INS_HUMAN | Insulin | Sécrétée | |
| P51460 | INSL3_HUMAN | Insulin-like 3 | Sécrétée | |
| Q14641 | INSL4_HUMAN | Early placenta insulin-like peptide (EPIL) | Sécrétée | |
| Q9Y5Q6 | INSL5_HUMAN | Insulin-like peptide INSL5 | Sécrétée | |
| Q9Y581 | INSL6_HUMAN | Insulin-like peptide INSL6 | Sécrétée | |
| Q9UEF7 | KLOT_HUMAN | Klotho | Membrane cellulaire | Hélice alpha |
| P00338 | LDHA_HUMAN | L-lactate dehydrogenase A chain (LDH-A) | Cytoplasme | |
| P07195 | LDHB_HUMAN | L-lactate dehydrogenase B chain | Cytoplasme | |

| | | | | |
|---|---|---|---|---|
| | | (LDH-B) | | |
| P07864 | LDHC_HUMAN | L-lactate dehydrogenase C chain (LDH-C) | Cytoplasme | |
| P01229 | LSHB_HUMAN | Lutropin subunit beta (LSH-B) (LSH-beta) | Sécrétée | |
| P20382 | MCH_HUMAN | Pro-MCH | Sécrétée | |
| P12872 | MOTI_HUMAN | Promotilin | Sécrétée | |
| P01178 | NEU1_HUMAN | Oxytocin-neurophysin 1 (OT-NPI) | Sécrétée | |
| P01185 | NEU2_HUMAN | Vasopressin-neurophysin 2-copeptin (AVP-NPII) | Sécrétée | |
| P61366 | OSTN_HUMAN | Osteocrin (Musclin) | Sécrétée | |
| P18509 | PACA_HUMAN | Pituitary adenylate cyclase-activating polypeptide (PACAP) | Sécrétée | |
| P01298 | PAHO_HUMAN | Pancreatic prohormone (PP) | Sécrétée | |
| P36871 | PGM1_HUMAN | Phosphoglucomutase-1 (PGM 1) | Cytoplasme | |
| P00558 | PGK1_HUMAN | Phosphoglycerate kinase 1 | Cytoplasme | |
| P07205 | PGK2_HUMAN | Phosphoglycerate kinase 2 | Cytoplasme | |
| P18669 | PGAM1_HUMAN | Phosphoglycerate mutase 1 | Cytoplasme | |
| P15259 | PGAM2_HUMAN | Phosphoglycerate mutase 2 | Cytosol | |
| P01236 | PRL_HUMAN | Prolactin (PRL) | Sécrétée | |
| P81277 | PRRP_HUMAN | Prolactin-releasing peptide (PrRP) | Sécrétée | |
| P12272 | PTHR_HUMAN | Parathyroid hormone-related protein (PTHrP) | Sécrétée | |
| P01270 | PTHY_HUMAN | Parathyroid hormone (PTH) | Sécrétée | |
| P08559 | ODPA_HUMAN | Pyruvate dehydrogenase E1 component subunit alpha, somatic form, mitochondrial | Matrice mitochondriale | |
| P29803 | ODPAT_HUMAN | Pyruvate dehydrogenase E1 component subunit alpha, testis-specific form, mitochondrial | Matrice mitochondriale | |
| P11177 | ODPB_HUMAN | Pyruvate dehydrogenase E1 component subunit beta, mitochondrial (PDHE1-B) | Matrice mitochondriale | |
| P14618 | KPYM_HUMAN | Pyruvate kinase PKM | Cytoplasme | |

| | | | | |
|---|---|---|---|---|
| P30613 | KPYR_HUMAN | Pyruvate kinase PKLR | Cytoplasme | |
| P10082 | PYY_HUMAN | Peptide YY (PYY) | Sécrétée | |
| P04808 | REL1_HUMAN | Prorelaxin H1 | Sécrétée | |
| P04090 | REL2_HUMAN | Prorelaxin H2 | Sécrétée | |
| Q8WXF3 | REL3_HUMAN | Relaxin-3 (Prorelaxin H3) | Sécrétée | |
| Q9HD89 | RETN_HUMAN | Resistin | Sécrétée | |
| Q9BQ08 | RETNB_HUMAN | Resistin-like beta | Sécrétée | |
| P09683 | SECR_HUMAN | Secretin | Sécrétée | |
| P61278 | SMS_HUMAN | Somatostatin | Sécrétée | |
| P01242 | SOM2_HUMAN | Growth hormone variant (GH-V) | Sécrétée | |
| P01241 | SOMA_HUMAN | Somatotropin | Sécrétée | |
| P52823 | STC1_HUMAN | Stanniocalcin-1 (STC-1) | Sécrétée | |
| O76061 | STC2_HUMAN | Stanniocalcin-2 (STC-2) | Sécrétée | |
| P01266 | THYG_HUMAN | Thyroglobulin (Tg) | Sécrétée | |
| Q8N2E6 | TOR2X_HUMAN | Prosalusin | Sécrétée | |
| P40225 | TPO_HUMAN | Thrombopoietin | Sécrétée | |
| P20396 | TRH_HUMAN | Pro-thyrotropin-releasing hormone (Pro-TRH) | Sécrétée | |
| P60174 | TPIS_HUMAN | Triosephosphate isomerase (TIM) | Cytoplasme | |
| P01222 | TSHB_HUMAN | Thyrotropin subunit beta (TSH-beta) | Sécrétée | |
| P02766 | TTHY_HUMAN | Transthyretin | Sécrétée | |
| P55089 | UCN1_HUMAN | Urocortin | Sécrétée | |
| Q96RP3 | UCN2_HUMAN | Urocortin-2 | Sécrétée | |
| Q969E3 | UCN3_HUMAN | Urocortin-3 | Sécrétée | |
| O95399 | UTS2_HUMAN | Urotensin-2 | Sécrétée | |
| Q765I0 | UTS2B_HUMAN | Urotensin-2B | Sécrétée | |
| P01282 | VIP_HUMAN | VIP peptides | Sécrétée | |
| O00217 | NDUS8_HUMAN | NADH dehydrogenase [ubiquinone] iron-sulfur protein 8, mitochondrial | Membrane interne mitochondriale | |
| O75306 | NDUS2_HUMAN | NADH dehydrogenase | Membrane interne | |

| | | | | |
|---|---|---|---|---|
| | | [ubiquinone] iron-sulfur protein 2, mitochondrial | mitochondriale | |
| O75489 | NDUS3_HUMAN | NADH dehydrogenase [ubiquinone] iron-sulfur protein 3, mitochondrial | Membrane interne mitochondriale | |
| O75947 | ATP5H_HUMAN | ATP synthase subunit d, mitochondrial | Membrane interne mitochondriale | |
| O95299 | NDUAA_HUMAN | NADH dehydrogenase [ubiquinone] 1 alpha subcomplex subunit 10, mitochondrial | Matrice mitochondriale | |
| P06576 | ATPB_HUMAN | ATP synthase subunit beta, mitochondrial | Membrane interne mitochondriale | |
| P13073 | COX41_HUMAN | Cytochrome c oxidase subunit 4 isoform 1, mitochondrial | Mitochondrion inner membrane | Hélice alpha |
| P20674 | COX5A_HUMAN | Cytochrome c oxidase subunit 5A, mitochondrial | Membrane interne mitochondriale | |
| P31930 | QCR1_HUMAN | Cytochrome b-c1 complex subunit 1, mitochondrial | Membrane interne mitochondriale | |
| Q02221 | CX6A2_HUMAN | Cytochrome c oxidase subunit 6A2, mitochondrial | Membrane interne mitochondriale | Hélice alpha |
| Q5HYK3 | COQ5_HUMAN | 2-methoxy-6-polyprenyl-1,4-benzoquinol methylase, mitochondrial | Membrane interne mitochondriale | |
| O96008 | TOM40_HUMAN | Mitochondrial import receptor subunit TOM40 homolog (Protein Haymaker) | Membrane externe mitochondriale | Tonneau bêta* |
| P45880 | VDAC2_HUMAN | Voltage-dependent anion-selective channel protein 2 (VDAC-2) | Membrane externe mitochondriale | Tonneau bêta |
| P21796 | VDAC1_HUMAN | Voltage-dependent anion-selective channel protein 1 (VDAC-1) | Membrane externe mitochondriale | Tonneau bêta |
| Q9Y277 | VDAC3_HUMAN | Voltage-dependent anion-selective channel protein 3 (VDAC-3) | Membrane externe mitochondriale | Tonneau bêta |
| Q969M1 | TM40L_HUMAN | Mitochondrial import receptor subunit TOM40B (Protein TOMM40-like) | Membrane externe mitochondriale | Tonneau bêta* |
| Q8NFP4 | MDGA1_HUMAN | MAM domain-containing glycosylphosphatidylinositol anchor protein 1 (GPIM) | Membrane cellulaire | Ancre GPI† |

| | | | | |
|---|---|---|---|---|
| Q6ZVN8 | RGMC_HUMAN | Hemojuvelin | Membrane cellulaire | Ancre GPI† |
| Q12891 | HYAL2_HUMAN | Hyaluronidase-2 (Hyal-2) | Membrane cellulaire | Ancre GPI† |
| Q96B86 | RGMA_HUMAN | Repulsive guidance molecule A | Membrane cellulaire | Ancre GPI† |
| O95497 | VNN1_HUMAN | Pantetheinase | Membrane cellulaire | Ancre GPI† |
| Q14982 | OPCM_HUMAN | Opioid-binding protein/cell adhesion molecule (OBCAM) | Membrane cellulaire | Ancre GPI† |
| Q9UKY0 | PRND_HUMAN | Prion-like protein doppel (PrPLP) | Membrane cellulaire | Ancre GPI† |
| Q9BZM6 | ULBP1_HUMAN | UL16-binding protein 1 (ALCAN-beta) | Membrane cellulaire | Ancre GPI† |
| Q9BZM4 | ULBP3_HUMAN | UL16-binding protein 3 (ALCAN-gamma) | Membrane cellulaire | Ancre GPI† |
| P04216 | THY1_HUMAN | Thy-1 membrane glycoprotein (CDw90) | Membrane cellulaire | Ancre GPI† |
| P43629 | KI3L1_HUMAN | Killer cell immunoglobulin-like receptor 3DL1 (CD158 antigen-like family member E) | Membrane cellulaire | Hélice alpha |
| Q68D85 | NR3L1_HUMAN | Natural cytotoxicity triggering receptor 3 ligand 1 (B7-H6) | Membrane cellulaire | Hélice alpha |
| Q8N271 | PROM2_HUMAN | Prominin-2 (PROM-2) | Membrane cellulaire | Hélice alpha |
| O15031 | PLXB2_HUMAN | Plexin-B2 (MM1) | Membrane cellulaire | Hélice alpha |
| Q9GZY6 | NTAL_HUMAN | Linker for activation of T-cells family member 2 | Membrane cellulaire | Hélice alpha |
| P56373 | P2RX3_HUMAN | P2X purinoceptor 3 (P2X3) | Membrane cellulaire | Hélice alpha |
| Q13635 | PTC1_HUMAN | Protein patched homolog 1 (PTC) | Membrane cellulaire | Hélice alpha |
| Q9UBD6 | RHCG_HUMAN | Ammonium transporter Rh type C | Membrane cellulaire | Hélice alpha |
| P28566 | 5HT1E_HUMAN | 5-hydroxytryptamine receptor 1E (5-HT-1E) | Membrane cellulaire | Hélice alpha |

| | | | | |
|---|---|---|---|---|
| O43570 | CAH12_HUMAN | Carbonic anhydrase 12 | Membrane cellulaire | Hélice alpha |
| P13569 | CFTR_HUMAN | Cystic fibrosis transmembrane conductance regulator (CFTR) | Membrane cellulaire (non-conventionnelle) | Hélice alpha |
| P05230 | FGF1_HUMAN | Fibroblast growth factor 1 (FGF-1) | Sécrétée (non-conventionnelle) | |
| P09038 | FGF2_HUMAN | Fibroblast growth factor 2 (FGF-2) | Sécrétée (non-conventionnelle) | |
| P09382 | LEG1_HUMAN | Galectin-1 (Gal-1) | Sécrétée (non-conventionnelle) | |
| P17931 | LEG3_HUMAN | Galectin-3 (Gal-3) | Sécrétée (non-conventionnelle) | |
| P01584 | IL1B_HUMAN | Interleukin-1 beta (IL-1 beta) | Sécrétée (non-conventionnelle) | |

**Tableau Annexe 3 | Informations sur le jeu de données test.** Toutes les données ont été récupérées sur UniProt le 25 Mai 2021. Les données sur la localisation subcellulaire est un bilan des données apportées par les sous-sections « *Subcellular location [CC]* » de « *Subcellular location* » et « *Gene ontology (cellular component)* » de « *Gene Ontology (GO)* ». Les données de la colonne « *GPI ? TM ?* » (TM, transmembranaire) annotées * sont issues de la sous-section « *Keywords - Domain* » de « *Miscellaneous* », celles annotées † « *Subcellular location [CC]* » de « *Subcellular location* » et les autres proviennent de la sous-section « *Transmembrane* » de « *Subcellular location* ».

**Annexe 4 : Extrait du fichier *ReadMe.md* de la branche dev du dépôt Github**

# Manon Connault Internship

This repository will contain all files and scripts writting during the internship of Manon CONNAULT (Master Degree 2 in Bioinformatic, Clermont-Auvergne University).

The internship is supervised by Mrs. Nadia GOUÉ (UCA Mesocenter) and Mrs. Muriel BONNET (UMR Herbivores INRAE).

## In the `dev` branch

Currently, you are in the `dev` branch. This branch created for work on the different scripts. You can too find different test data files in the "tests-data/" folder and the test scripts in the "tests/" folder.

## How to use the different scripts

### Script "Slurm-Workflow_SignaP-TargetP-DeepLoc.slurm"

- Slurm environment

| Tool Name | Version | Installation |
|---|---|---|
| GCC: GNU Compiler Collection | 8.1.0 | `module load gcc/8.1.0` |
| Python | 3.7.1 | `module load python/3.7.1` |
| Python | 2.7.9 | `module load python/2.7.9` |
| Entrez Direct (EDirect) | 1.0 | `module load edirect/1.0` |
| GNU Parallel | 20151222 | `module load parallel/20151222` |
| SignalP | 5.0b | `module load signalp/5.0b` |
| TargetP | 2.0 | `module load targetp/2.0` |
| DeepLoc | 1.0 | `module load deeploc/1.0` |

Script launcher : `sbatch Slurm-Workflow_SignaP-TargetP-DeepLoc.slurm`

## Annexe 5 : Script Python *gConvert.py*

```
#!/usr/bin/env python3
""" g:Convert API (version 0.7.3). (c) 2021 CC-BY-NC-SA """
# coding: utf-8


##---TO DO---##
# Definition of the entry (string, list, file)
# Add other options for the results out (as to write a file)
# Definition of methods checking line commands
# Definition of a method checking URL code (ex. stop code of error 404)
# Add try -except commands
# Add a method for make a list of organisms available from g:Profiler
# Definition of the informations out
# Definition of how to write a file
# Add more commentaries
# Add code description with Logging library
# Add a documentation class and function


__all__=[]
__author__="Manon CONNAULT"
__date__="8th June 2021"
__version__="0.7.3"
__copyright__="(c) 2021 CC-BY-NC-SA"


# Library import
import argparse
import logging
import os
import re
import requests
import statistics
import sys
import time
from collections import defaultdict
from collections import OrderedDict
#from configparser import ConfigParser


# Global variables

logger = logging.getLogger(__name__)

# Global Data structures

# Functions

    # Pre-processing arguments 1

def parse_arguments():
    """Definition of arguments of the script
```

```
53
54      Parameters:
55      none
56
57      Returns:
58      Several arguments
59      Use -h for more details
60      """
61      parser=argparse.ArgumentParser(
62          description="This tool use the API of g:Convert from g:Profiler.",
63          formatter_class=argparse.ArgumentDefaultsHelpFormatter,
64          prefix_chars='-',
65          add_help=True)
66      parser.add_argument("-l", "--log-level", default='WARNING', dest="logLevel",
        type=str.upper,choices=['DEBUG', 'INFO', 'WARNING', 'ERROR', 'CRITICAL'],
        help="""Provide logging level. Example --log-level DEBUG. Default:
        %(default)s""")
67      parser.add_argument('-c', '--config-file', required=False, type=str,
        dest="configFile", help="path to configuration file. Default : the current
        directory")
68      parser.add_argument("-Q", "--query", required=False, type=str,
        help="""Definition of the query. If your query is a file, please use the -F or
        --queryFile argument.""")
69      parser.add_argument("-F", "--queryFile", required=False, help="""Definition of
        the query file.""")
70      parser.add_argument("-O", "--organism", type=str, default=None,
        help="""Definition of the organism name. For the list of organisms available
        from g:Profiler along with their associated namespaces, let the default value
        'None'.""")
71      parser.add_argument("-T", "--target", type=str, default="ENSP",
        help="""Definition of the namespace to convert to. The default value is
        ENSP.""")
72      parser.add_argument("-N", "--numericNS", type=str, default="ENTREZGENE_ACC",
        help="""Definition of namespace to use when input IDs are numeric. The default
        value is ENTREZGENE_ACC.""")
73      parser.add_argument("-L", "--organismsList", action='store_true',
        default=False, help="""Discover the species data along with the associated
        namespaces.""")
74      parser.add_argument("-X", "--extraData", action='store_true', default=False,
        help="""Boolean for the method for discovering the species available from
        g:Profiler and for choose whether to include lists of available namespaces for
        organisms.""")
75      parser.add_argument("-o", "--fileName", required=False, type=str, help="""Path
        to results file. Separation is a tabulation. Default is the standard out.
        Extension by default is textual (.txt).""")
76      parser.add_argument("-t", "--timeout", type=float ,default=None,
        help="""Timeout for requests operations. Default: None""")
77      return parser.parse_args()
78
79      # Pre-processing arguments 2
80
81  def format_query (query=None, queryFile=None):
```

```
82      listDataLines = []
83          # List for the 'file' type case
84      if query and not queryFile:
85          logger.info("Data input in standard entry.")
86          logger.debug("Extraction data from standard entry...")
87          allDataLines = query
88      elif not query and queryFile:
89          logger.info('Data input in the file: ' + str(queryFile))
90          try:
91              logger.debug("File opening...")
92              fileHandler = open(queryFile, "r")
93          except FileNotFoundError as fnf:
94              raise fnf
95          else:
96              logger.debug("... opening sucessfull")
97              logger.debug("Extraction data from file...")
98              listDataLines = [x.strip() for x in fileHandler.readlines()]
99              fileHandler.close()
100             if len(listDataLines) == 1:
101                 for listid in listDataLines:
102                     allDataLines = listid
103             else:
104                 allDataLines = listDataLines
105     elif not query and not queryFile:
106         raise EOFError("Missing argument. Please use -h for more informations")
107     elif query and queryFile:
108         raise Exception("String and file resquest both used. Please check or use ↵
            -h for more informations")
109         # Don't found the error term
110     logger.debug("... done")
111     return allDataLines
112
113     # Pre-processing arguments 3
114
115 def check_path_fileName (fileName=None, cwd=None, timestr=None):
116     outWD = cwd + '/gConvert-results_' + timestr
117     if fileName:
118         path = os.path.dirname(fileName)
119         filename = os.path.basename(fileName)
120         if path == "":
121             try:
122                 directory = os.mkdir(outWD)
123             except PermissionError as permErr:
124                 raise permErr
125             else:
126                 fileName = outWD + "/" + filename
127     return fileName
128
129 # Class
130
131 class gProfilerToolBox :
132
```

```
133     def __init__ (self, query, organism, target, numericNS, organismsList,
        extraData, timeout, fileName):
134         self.query = query
135         self.organism = organism
136         self.target = target
137         self.numericNS = numericNS
138         self.organismsList = organismsList
139         self.extraData = extraData
140         self.timeout = timeout
141         self.fileName = fileName
142         if self.organism == None:
143             logger.info("Demand of organism list sending to G:Profiler.")
144             self.fetch_organism(organism=self.organism, extraData=self.extraData,
                timeout=self.timeout)
145             logger.debug("Demand of organism list sucessfull")
146             organismsList = self.list_fetchOrganism()
147             self.out_ListOrganisms(fileName=self.fileName,
                listOrganisms=organismsList)
148         else :
149             self.check_APIArguments (organism=self.organism, target=self.target,
                numericNS=self.numericNS, timeout=self.timeout)
150             self.gConvert(organism=self.organism, query=self.query,
                target=self.target, numericNS=self.numericNS, timeout=self.timeout)
151             dicoAllId = self.allIdConverted_gConvert()
152             self.out_allIdList(fileName=self.fileName, dicoAllId=dicoAllId)
153             idResults = self.idList_gConvert()
154             self.out_idList(fileName=self.fileName, idConvertedList=idResults)
155             if self.organismsList:
156                 logger.info("Demand of organism data sending to G:Profiler.")
157                 logger.debug(str(self.organism) + ' : organism chosen')
158                 self.fetch_organism(organism=self.organism,
                    extraData=self.extraData, timeout=self.timeout)
159                 logger.debug("Demand of organism data sucessfull")
160                 organismData = self.organism_data()
161                 self.out_organismData (fileName=self.fileName,
                    organismData=organismData)
162
163     def __name__ (self):
164         return "gProfilerToolBox"
165
166     def fetch_organism (self, organism, extraData, timeout,
        organismsJson="Undefined"):
167         try:
168             r = requests.get(
169                 url = 'https://biit.cs.ut.ee/gprofiler/api/util/organisms_list/',
170                 json = {
171                     'organism': organism,
172                     'extra_data': extraData
173                     },
174                 timeout = timeout
175                 )
176         except requests.HTTPError as http_err:
```

```
177                 raise http_err
178             except requests.ConnectionError as connect_err:
179                 raise connect_err
180             except requests.Timeout as timeout_err:
181                 raise timeout_err
182             except requests.RequestException as req_err:
183                 raise req_err
184             except Exception as err:
185                 logger.error('Other error occurred: ' + str(err))
186             else:
187                 if r.status_code == requests.codes.ok:
188                     self.organismsJson = r.json()
189 
190         def list_fetchOrganism (self):
191             logger.debug("Extraction data from Json result...")
192             dicoResults = defaultdict(list)
193             for item in self.organismsJson:
194                 for key in item:
195                     if key == "id" :
196                         idValue = item[key]
197                     elif key == "scientific_name" :
198                         sctfNameValue = item[key]
199                 dicoResults[sctfNameValue]
200                 if idValue not in dicoResults[sctfNameValue] :
201                     dicoResults[sctfNameValue].append(idValue)
202             dicoResults = OrderedDict(sorted(dicoResults.items()))
203             logger.debug('... done')
204             return dicoResults
205 
206         def out_ListOrganisms (self, fileName, listOrganisms):
207             if not fileName:
208                 logger.info("Organisms list writting on standard exit.")
209                 for organismSp in listOrganisms:
210                     for organismId in listOrganisms[organismSp]:
211                         sys.stdout.write(str(organismSp) + "\t" + str(organismId) + "\n")
212             elif fileName:
213                 path = os.path.dirname(fileName)
214                 filename = os.path.basename(fileName)
215                 (file, ext) = os.path.splitext(filename)
216                 if ext == "":
217                     if path == "":
218                         fileName = file + "_list-organisms.txt"
219                     else:
220                         fileName = path + "/" + file + "_list-organisms.txt"
221                 else:
222                     if path == "":
223                         fileName = file + "_list-organisms" + ext
224                     else:
225                         fileName = path + "/" + file + "_list-organisms" + ext
226                 logger.info('Organisms list writting in the file: ' + str(fileName))
227                 try:
228                     logger.debug("Results file opening...")
```

```
229                 fileHandler = open(fileName, "w")
230             except FileNotFoundError as fnf:
231                 raise fnf
232             else:
233                 logger.debug("... opening sucessfull")
234                 logger.debug("Organisms list writting...")
235                 for organismSp in listOrganisms:
236                     for organismId in listOrganisms[organismSp]:
237                         fileHandler.write(str(organismSp) + "\t" + str(organismId)  ↲
                            + "\n")
238                 logger.debug("... done")
239
240     def check_APIArguments (self, organism, target, numericNS, timeout):
241         listOrganism = []
242         listTarget = []
243         listNumericNamespaces = []
244         logger.debug("Check of different arguments for G:Convert API...")
245         logger.debug("Organism list demand sending to G:Profiler.")
246         self.fetch_organism(organism=None, extraData=False, timeout=self.timeout)
247         listOrganism = self.organismsJson
248         for item in listOrganism:
249             for key in item:
250                 if key == "id" :
251                     listOrganism.append(item[key])
252         if organism not in listOrganism:
253             logger.error("Invalide \"organism\" argument. Please check the          ↲
                argument in the  following list:")
254             listOrganisms = self.list_fetchOrganism()
255             self.out_ListOrganisms(fileName=None, listOrganisms=listOrganisms)
256             raise SystemExit
257         logger.debug("Organism data demand sending to G:Profiler.")
258         self.fetch_organism(organism=organism, extraData=True, timeout=self.timeout)
259         listResultOrganism = self.organismsJson
260         for item in listResultOrganism:
261             for key in item:
262                 if key == "nonnumeric_namespaces":
263                     for idDB in item[key]:
264                         listTarget.append(idDB)
265                 elif key == "numeric_namespaces":
266                     for numIdDB in item[key]:
267                         listNumericNamespaces.append(numIdDB)
268         if target not in listTarget:
269             logger.error("Invalide \"target\" argument. Please check the argument   ↲
                in the following list:")
270             sys.stdout.write('\n'.join(listTarget) + "\n")
271             raise SystemExit
272         elif numericNS not in listNumericNamespaces:
273             logger.error("Invalide \"numeric_ns\" argument. Please check the        ↲
                argument in the following list:")
274             sys.stdout.write('\n'.join(listNumericNamespaces) + "\n")
275             raise SystemExit
276         logger.debug("... check of different arguments done.")
```

```
277
278     def gConvert (self, organism, query, target, numericNS, timeout,
        resultsJson="Undefined"):
279         logger.info("Query sending to G:Convert.")
280         listIdQuery = []
281         listIdNotConverted = []
282         try:
283             r = requests.post(
284                 url = 'https://biit.cs.ut.ee/gprofiler/api/convert/convert/',
285                 json = {
286                     'organism': organism,
287                     'target': target,
288                     'query': query,
289                     'numeric_ns': numericNS
290                     },
291                 timeout = timeout
292                 )
293             r.raise_for_status()
294         except requests.HTTPError as http_err:
295             raise http_err
296         except requests.ConnectionError as connect_err:
297             raise connect_err
298         except requests.Timeout as timeout_err:
299             raise timeout_err
300         except requests.RequestException as req_err:
301             raise req_err
302         except Exception as err:
303             logger.error('Other error occurred: ' + str(err))
304         else:
305             if r.status_code == requests.codes.ok:
306                 logger.debug("Query return sucessfull")
307                 for item in r.json()['result'] :
308                     for key in item:
309                         if key == "converted":
310                             if item[key] == "nan":
311                                 idQueryEmpty = item["incoming"]
312                                 listIdNotConverted.append(idQueryEmpty)
313                         elif key == "incoming" :
314                             idQuery = item[key]
315                             if idQuery not in listIdQuery:
316                                 listIdQuery.append(idQuery)
317                 if len(listIdNotConverted) == len(listIdQuery):
318                     logger.warning("Any identifier was converted. Please check
                        your query list and your arguments : \n-organism : " +
                        organism + "\n-target : " + target + "\n-numeric_ns : " +
                        numericNS + "\n-query :\n" + '\n'.join(listIdNotConverted))
319                     raise SystemExit
320                 else:
321                     self.resultsJson = r.json()
322
323     def allIdConverted_gConvert (self):
324         logger.debug("Extraction of all converted identifiers.")
```

```
325         listIdIncoming = []
326         listIdIncomingConverted = []
327         nbAllIdentifiers = 0
328         nbIdConverted = 0
329         dicoIdlist = defaultdict(list)
330         dicoQueryNb = defaultdict(int)
331         listNbIso = []
332         dicoAllId = defaultdict(list)
333         try:
334             logger.debug("Json file reading...")
335             for item in self.resultsJson['result']:
336                 for key in item:
337                     if key == "converted":
338                         idConverted = item[key]
339                     elif key == "incoming":
340                         idIncoming = item[key]
341                 dicoAllId[idConverted].append(idIncoming)
342                 if idIncoming not in listIdIncoming:
343                     listIdIncoming.append(idIncoming)
344                 if idConverted != "nan" and idIncoming not in                    ↲
                    listIdIncomingConverted:
345                     listIdIncomingConverted.append(idIncoming)
346                 nbAllIdentifiers = len(listIdIncoming)
347                 nbIdConverted = len(listIdIncomingConverted)
348         except KeyError as key_err:
349             raise key_err
350         except Exception as e:
351             raise e
352         else:
353             logger.debug("... done")
354             PercentIdConv = (nbIdConverted/nbAllIdentifiers)*100
355             PercentIdConv = "%.2f" % PercentIdConv
356             logger.info('For ' + str(nbAllIdentifiers) + ' gene names given, ' +    ↲
                str(PercentIdConv) + '% converted (as ' + str(nbIdConverted) + '        ↲
                identifier(s)).')
357             if nbIdConverted/nbAllIdentifiers < 0.25 :
358                 logger.warning('Be carefull, only '+ str(PercentIdConv) + '% of     ↲
                    identifiers was converted. Check your request. More informations    ↲
                    in the log file.')
359             for idConverted in dicoAllId :
360                 if idConverted != "nan":
361                     for idQuery in dicoAllId[idConverted]:
362                         dicoIdlist[idQuery].append(idConverted)
363             for query in dicoIdlist :
364                 dicoQueryNb[query]=len(dicoIdlist[query])
365             for query in dicoQueryNb :
366                 listNbIso.append(dicoQueryNb[query])
367             try:
368                 meanIso = statistics.mean(data=listNbIso)
369             except statistics.StatisticsError as statsErr:
370                 raise statsErr
371             else:
```

```
372                 try:
373                     stdevNbIso = statistics.pstdev(data=listNbIso, mu=meanIso)
374                 except statistics.StatisticsError as statsErr:
375                     raise statsErr
376                 else:
377                     try:
378                         medianNbIso =  statistics.median(data=listNbIso)
379                     except statistics.StatisticsError as statsErr:
380                         raise statsErr
381                     else:
382                         try:
383                             quantilesNbIso = statistics.quantiles(data=listNbIso,   ↲
                                n=4, method='inclusive')
384                             for value in range(len(quantilesNbIso)):
385                                 firstQt = quantilesNbIso[0]
386                                 thirdQt = quantilesNbIso[2]
387                         except statistics.StatisticsError as statsErr:
388                             raise statsErr
389                         else:
390                             logger.info('In the results, a mean of ' +              ↲
                                str(meanIso) + ' of isoforms is present, with a         ↲
                                standard deviation of ' + str(stdevNbIso) + ', and a    ↲
                                median of ' + str(medianNbIso) + ' with ' +             ↲
                                str(firstQt) + ' as first quantile and as ' +           ↲
                                str(thirdQt) + ' third quantile.')
391                     return dicoAllId
392
393         def out_allIdList (self, fileName, dicoAllId):
394             if fileName:
395                 path = os.path.dirname(fileName)
396                 filename = os.path.basename(fileName)
397                 (file, ext) = os.path.splitext(filename)
398                 if ext == "":
399                     if path == "":
400                         fileName = + file + "_all-id.txt"
401                     else:
402                         fileName = path + "/" + file + "_all-id.txt"
403                 else:
404                     if path == "":
405                         fileName = file + "_all-id" + ext
406                     else:
407                         fileName = path + "/" + file + "_all-id" + ext
408                 logger.info('All identifiers writting in the file: ' + str(fileName))
409                 try:
410                     logger.debug("Converted identifiers list file opening...")
411                     fileHandler = open(fileName, 'w')
412                 except IOError as ioe:
413                     raise ioe
414                 except Exception as e:
415                     raise e
416                 else:
417                     logger.debug("... opening sucessfull.")
```

```
418                 logger.debug("All identifiers writting...")
419                 for idConverted in dicoAllId:
420                     for idIncoming in dicoAllId[idConverted]:
421                         fileHandler.write(str(idIncoming) + '\t' +
                            str(idConverted)+ '\n')
422                 logger.debug("... writting done.")
423         else:
424             logger.info("All identifiers writting on the standard error exit.")
425             for idConverted in dicoAllId:
426                 for idIncoming in dicoAllId[idConverted]:
427                     logger.debug(str(idIncoming) + '\t' + str(idConverted))
428
429     def idList_gConvert (self):
430         logger.info("Extraction of the converted identifiers.")
431         dicoResults = defaultdict(lambda: defaultdict(str))
432         try:
433             logger.debug("Json file reading...")
434             for item in self.resultsJson['result']:
435                 for key in item:
436                     if key == "converted" and item[key]!="nan":
437                         dicoResults[item[key]] = item
438         except KeyError as key_err:
439             raise key_err
440         except Exception as e:
441             raise e
442         else:
443             logger.debug("... done")
444             return dicoResults.keys()
445
446     def out_idList (self, fileName, idConvertedList):
447         if fileName:
448             path = os.path.dirname(fileName)
449             fnWithoutPath = os.path.basename(fileName)
450             (file, ext) = os.path.splitext(fnWithoutPath)
451             if ext == "":
452                 if path == "":
453                     fileName = file + ".txt"
454                 else:
455                     fileName = path + "/" + file + ".txt"
456             else:
457                 fileName = fileName
458             logger.info('Results writting in the file: ' + str(fileName))
459             try:
460                 logger.debug("Converted identifiers list file opening...")
461                 fileHandler = open(fileName, 'w')
462             except IOError as ioe:
463                 raise ioe
464             except Exception as e:
465                 raise e
466             else:
467                 logger.debug("... opening sucessfull.")
468                 logger.debug("Converted identifiers writting...")
```

```
469                 fileHandler.write('\n'.join(idConvertedList))
470                 fileHandler.close()
471                 sys.stdout.write('\n'.join(idConvertedList))
472                 logger.debug("... writting done.")
473         else:
474             logger.info("Results writting on the standard exit.")
475             sys.stdout.write('\n'.join(idConvertedList) + "\n")
476 
477     def organism_data (self):
478         logger.debug("Extraction data from organism data...")
479         dicoResults = defaultdict(str)
480         for indice in self.organismsJson:
481             for key in indice:
482                 nameValue = key
483                 dataValue = indice[key]
484                 dicoResults[nameValue] = dataValue
485         logger.debug("... done")
486         return dicoResults
487 
488     def out_organismData (self, fileName, organismData):
489         if not fileName:
490             logger.info("Organism data writting on standard exit.")
491             for key in organismData:
492                 sys.stdout.write(str(key) + "\t" + str(organismData[key]) + "\n")
493         elif fileName:
494             path = os.path.dirname(fileName)
495             filename = os.path.basename(fileName)
496             (file, ext) = os.path.splitext(filename)
497             if ext == "":
498                 if path == "":
499                     fileName = file + "_data-organism.txt"
500                 else:
501                     fileName = path + "/" + file + "_data-organism.txt"
502             else:
503                 if path == "":
504                     fileName = file + "_data-organism" + ext
505                 else:
506                     fileName = path + "/" + file + "_data-organism" + ext
507             logger.info('Organisms data writting in the file: ' + str(fileName))
508             try:
509                 logger.debug("Results file opening...")
510                 fileHandler = open(fileName, "w")
511             except FileNotFoundError as fnf:
512                 raise fnf
513             else:
514                 logger.debug("... opening sucessfull")
515                 logger.debug("Organism results writting...")
516                 for key in organismData:
517                     fileHandler.write(str(key) + "\t" + str(organismData[key]) +    ↲
                        "\n")
518                 logger.debug("... done")
519 
```

```
520
521 if __name__ == "__main__":
522     args = parse_arguments()
523     cwd = os.getcwd()
524     timestr = time.strftime("%Y-%m-%d-%H-%M-%S")
525
526     try:
527         for entry in os.listdir(cwd):
528             if entry == 'config.yml':
529                 # Determination of the information to find
530                 if args.configFile:
531                     logfile = args.configFile
532                 else:
533                     logfile = '/results-gConvert_' + timestr + '.log'
534             else:
535                 if args.configFile:
536                     logfile = args.configFile
537                     path = os.path.dirname(args.configFile)
538                     filename = os.path.basename(args.configFile)
539                     (file, ext) = os.path.splitext(filename)
540                     if ext == "":
541                         if path == "":
542                             logfile = cwd + "/" + args.configFile + ".log"
543                         else:
544                             logfile = args.configFile + ".log"
545
546                     else:
547                         if path == "":
548                             logfile = cwd + "/" + args.configFile
549                         else:
550                             logfile = args.configFile
551                 else:
552                     logfile = cwd + '/results-gConvert_' + timestr + '.log'
553     except PermissionError as permErr:
554         raise permErr
555     else:
556         log_format = '%(asctime)s - %(name)s: %(process)d - %(levelname)s -          ↲
            %(message)s'
557         logging.basicConfig(format=log_format,
558             level=args.logLevel,
559             handlers=[
560                 logging.StreamHandler(sys.stderr),
561                 logging.FileHandler(logfile)
562                 ]
563             )
564
565     fileName = check_path_fileName(fileName=args.fileName, cwd=cwd, timestr=timestr)
566
567     dataQuery = format_query(query=args.query, queryFile=args.queryFile)
568     dataAnalyse = gProfilerToolBox(query=dataQuery, organism=args.organism,          ↲
        target=args.target, numericNS=args.numericNS,                                    ↲
        organismsList=args.organismsList, extraData=args.extraData,                      ↲
        timeout=args.timeout, fileName=fileName)
569
```

## Annexe 6 : Script Bash *EntreDirect.sh*

```bash
#!/bin/bash


# Program configuration

__author__='Manon Connault'
__date__='27th April 2021'
__version__='0.1.5'
__status__='Development'
__copyright__='(c) 2021 CC-BY-NC-SA'


# Arguments definition

# $1 = Identifiers list


# Data structure declarations


# Function declarations


# Entrez-Direct

echo 'Reading identifiers' >&2

echo 'Use of Entrez-Direct' >&2

cat $1 |
    join-into-groups-of 5000 |
    sed 's/,/ OR /g' |
    parallel -j 3 "esearch -db protein -organism human -query {} |
        efetch -format fasta"

```

# Annexe 7 : Tableau récapitulant les outils répertoriés

## Annexe 7.1 : Outils de prédiction du peptide signal

| Nom | Description | Développeur | Type | Espèce(s) | Site | Source(s) bibliographique(s) |
|---|---|---|---|---|---|---|
| DeepSig | Prédiction des peptides signaux et leurs sites de clivage | Castrense SAVOJARDO | Outil | Eucaryotes<br>Bactérie à Gram (positif, négatif) | https://deepsig.biocomp.unibo.it/welcome/default/index | Savojardo et al. 2018 |
| OutCyte | Prédiction des protéines en tant que protéine contenant un signal-peptide, un domaine transmembranaire, intracellulaire ou sécrétées de manière non conventionnelle | Kai STÜHLER | Outil | Eucaryotes | http://www.outcyte.com/ | Zhao et al. 2019 |
| SecretSanta | Ensemble de pipelines permettant une prédiction évolutive des protéines sécrétées | Anna GOGLEVA<br>Hajk-Georg DROST | Pipeline | Eucaryote (animal, plante, fungi)<br>Archées<br>Bactérie à Gram (positif, négatif) | https://github.com/gogleva/SecretSanta | Gogleva et al. 2018 |
| SignalP | Prédiction de la présence et de la localisation des sites de clivage du peptide signal dans les séquences d'acides aminés de différents organismes | Thomas Nordahl PETERSEN<br>Henrik NIELSEN | Outil | Eucaryotes<br>Archées<br>Bactérie à Gram (positif, négatif) | http://www.cbs.dtu.dk/services/SignalP/ | Almagro Armenteros, Tsirigos, et al. 2019 |

**Tableau Annexe 7.1 | Outils de prédiction du peptide signal répertoriés.** Toutes ces données ont été mises à jour le 27 Mai 2021.

## Annexe 7.2 : Outils de prédiction de la localisation subcellulaire

| Nom | Description | Développeur | Type | Espèce(s) | Site | Source(s) bibliographique(s) |
|---|---|---|---|---|---|---|
| BUSCA | Prédiction de la localisation subcellulaire d'une protéine à partir de sa séquence | Castrense SAVOJARDO | Workflow | Animaux<br>Plantes<br>Fungi<br>Autres eucaryotes<br>Bactéries Gram positif ou négatif<br>Autres procaryotes | http://busca.biocomp.unibo.it/ | Savojardo, Pier Luigi Martelli, et al. 2018 |
| DeepLoc | Prédiction de la localisation subcellulaire des protéines eucaryotes à l'aide du Deep learning | Henrik NIELSEN | Outil | Eucaryotes | http://www.cbs.dtu.dk/services/DeepLoc/ | Almagro Armenteros et al. 2017 |
| MemLoci | Prédiction de la localisation subcellulaire de protéines associées ou insérées dans les membranes eucaryotes (membrane plasmique, organelle, système endomembranaire…) | Andrea PIERLEONI<br>Pier Luigi MARTELLI | Outil | Eucaryotes | https://mu2py.biocomp.unibo.it/memloci/default/index | Pierleoni et al. 2011 |
| MemPype | Prédiction de la topologie et de la localisation subcellulaire des protéines membranaires eucaryotes | Andrea PIERLEONI<br>Pier Luigi MARTELLI | Pipeline | Eucaryotes | https://mu2py.biocomp.unibo.it/mempype | Pierleoni, Indio, et al. 2011 |
| MULocDeep | Deep-learning framework pour la prédiction de la localisation sous-cellulaire et sous-organellaire des protéines avec interprétation au niveau des résidus | Dong XU | Outil | Eucaryotes | http://www.mu-loc.org/ | (Non publié) Jiang et al. 2020 |
| TargetP | Prédiction de la localisation subcellulaire des protéines eucaryotes à l'aide du Deep Learning | Henrik NIELSEN | Outil | Plante<br>« Non plante » | http://www.cbs.dtu.dk/services/TargetP/ | Almagro Armenteros, Salvatore, et al. 2019 |

**Tableau Annexe 7.2 | Outils de prédiction de la localisation subcellulaire répertoriés.** Toutes ces données ont été mises à jour le 27 Mai 2021.

## Annexe 7.3 : Outils de prédiction de la topologie

| Nom | Description | Développeur | Type | Espèce(s) | Site | Source(s) bibliographique(s) |
|---|---|---|---|---|---|---|
| OCTOPUS | Prédiction de la topologie des protéines transmembranaires en intégrant la modélisation des régions réentrantes / membranaires et des épingles à cheveux transmembranaires dans la grammaire topologique | Arne ELOFSSON | Outil | *Non renseigné* | https://octopus.cbr.su.se/ | Viklund and Elofsson 2008 |
| PredGPI | Système de prédiction des protéines à ancre GPI | Andrea PIERLEONI Rita CASADIO Pier Luigi MARTELLI | Outil | Homo sapiens (NCBI 36.48) Arabidopsis thaliana (Integr8 v.75) | http:// gpcr2.biocomp.unibo.it/ gpipe/index.htm | Pierleoni et al. 2008 |
| PRED-TMBB2 | Prédiction de la topologie et de la détection des protéines de la membrane externe avec tonneau bêta (beta-barrel) | Pantelis BAGOS | Outil | *Non renseigné* | http://www.compgen.org/ tools/PRED-TMBB2 | Tsirigos et al. 2016 |
| SCAMPI2 | Prédiction de la topologie des protéines transmembranaires en hélice alpha avec la règle de la première et de la dernière hélice hydrophobe | Arne ELOFSSON | Outil | *Non renseigné* | https://scampi.cbr.su.se/ | Peters et al. 2016 |
| TMHMM | Prédiction d'hélices transmembranaires dans les protéines | Anders KROGH | Outil | *Non renseigné* | http://www.cbs.dtu.dk/ services/TMHMM/ | Krogh et al. 2001 |
| | | | | | https:// services.healthtech.dtu.dk/ service.php?TMHMM-2.0 | |
| | | Richel J.C. BILDERBEEK | | | https:// services.healthtech.dtu.dk/ service.php?TMHMM-2.0 | |
| TOPCONS / TOPCONS2 | Prédiction consensus de la topologie des protéines membranaires | Arne ELOFSSON | Workflow | Eucaryotes Archées Procaryotes | http://topcons.cbr.su.se/ | Tsirigos et al. 2015 |

**Tableau Annexe 7.3 | Outils de prédiction de la topologie répertoriés.** Toutes ces données ont été mises à jour le 27 Mai 2021.

**Annexe 8 : Résumé du poster soumis pour l'évènement JOBIM 2021 (https://jobim2021.sciencesconf.org/, référence : 59)**

# Talkmine, a workflow for the prediction of the interactions between secretome and surfaceome in the dialogue between cellular types

Manon CONNAULT[1], Jérémy TOURNAYRE[2], Céline BOBY[2], Muriel BONNET[2] and Nadia GOUÉ[1]

1 AuBi plateform, Mesocenter, Clermont-Auvergne University, Turing Building, 7 Avenue Blaise Pascal, 63170 AUBIERE, France
2 INRAE, Clermont-Auvergne University, Vetagro Sup, UMRH, 63122 SAINT-GENES-CHAMPANELLE, France

Corresponding Authors: nadia.goue@uca.fr, muriel.bonnet@inrae.fr

## 1 Context

The prediction of the protein-protein interactions is studied in different domains, for example for their roles in the interplay between cellular types or tissues[1]. In particular, the interactions between surfaceome and secretome may strongly improved the understanding of cellular crosstalk. This project aims to develop a Snakemake-based workflow[2] named Talkmine for the identification of molecular dialogue between two biologic tissues resulting from protein-protein interactions.

## 2 Materials and Methods

First objective was to identify and test some opensource tools known to predict proteins that belong to secretome or surfaceome. Publically available tools were classified according to three classes, i.e. peptide signal, subcellular location or topology prediction, and have been tested with a dataset of 165 UniProt protein entries whose subcellular location are known.

Secondly, we developed a workflow to connect the selected tools. In brief, user sends a gene or protein identifiers list to a python tool, g:Convert[3] which converts identifiers to a same format, according to a defined database. Then, Entrez-Direct tool[4] generates a multi-fasta file with all protein sequences listed in the NCBI database, injected to secretome and surfaceome tools. Finally, proteins tagged to secretome and surfaceome classes are send to PSICQUIC tool[5], which determines the interactions between proteins from these two classes.

## 3 Results

The resulting protein-protein interactions prediction from Talkmine is available both in standard output and in flat file. The user has access to the list of protein-protein interactions between the two tissues but also to the intermediate results.

## 4 Conclusion and perspectives

The workflow Talkmine is developed to predict the interactions between the proteins from the secretome and the surfaceome. This work was initiated to be applied to *Bos taurus* to determine the interactions between muscle and fat tissue but it is possible to apply it to other tissues and to application fields such as biomedical or food research.

## References

[1] Bonnet M., et al. Prediction of the Secretome and the Surfaceome: A Strategy to Decipher the Crosstalk between Adipose Tissue and Muscle during Fetal Growth. *Int J Mol Sci* 2020;21:4375. doi:10.3390/ijms21124375

[2] Mölder F., et al. Sustainable data analysis with Snakemake. *F1000Res* 2021;10:33. doi:10.12688/f1000research.29032.1

[3] Raudvere U., et al. g:Profiler: a web server for functional enrichment analysis and conversions of gene lists (2019 update). *Nucleic Acids Res* 2019;47:W191–W198. doi:10.1093/nar/gkz369

[4] Kans J. Entrez Direct: E-utilities on the Unix Command Line. In: *Entrez Programming Utilities Help [Internet]*. National Center for Biotechnology Information (US), 2013.

[5] Aranda B., et al. PSICQUIC and PSISCORE: accessing and scoring molecular interactions. *Nat Methods*, 2011;8:528–529. doi:10.1038/nmeth.1637