



UNIVERSITÉ DE NANTES

UNIVERSITÉ DE NANTES

MASTER 1 DE BIO-INFORMATIQUE BIostatistique

RAPPORT DE STAGE

Développement d'outils de criblage de génomiques de moustiques pour la recherche d'une famille de protéines.

Auteur :

Amandine Lecerf Defer

Encadrant :

Pr. Bernard OFFMANN



Du 26 Mars 2018 au 18 Mai 2018
amandine.lecerf-defer@etu.univ-nantes.fr

Sommaire

1	Remerciements	4
2	Introduction	5
2.1	Contexte	5
2.2	Les OBP	5
2.3	La base de données mOBPdb	5
2.4	Objectifs du stage	6
3	Cahier des charges techniques	7
3.1	Connaissances sur les OBPs	7
3.2	Récupération des données de génomes	7
3.3	Recherche d'OBPs par homologie	7
3.4	Recherche des orthologues	7
3.5	Récupération d'informations	8
3.6	Intégration des informations dans la base de données déjà existante	8
3.7	Les outils	8
3.7.1	<i>Psi Blast</i>	8
3.7.2	phpMyAdmin	8
3.7.3	Wordpress	9
3.8	Langages utilisés : Bash et Python	9
3.8.1	Bash	9
3.8.2	Python	9
3.9	Recherche nom d'entrée Uniprot	9
4	Ce qui a été fait	10
4.1	Mise en place de l'environnement de travail	10
4.2	Recherche d'OBPs dans un génome inconnu	11
4.3	Recherche d'orthologues pour les nouvelles OBPs trouvées	13
4.4	Récupération des diverses informations	14
4.4.1	Quel format ?	14
4.4.2	Localisation	15
4.4.3	Longueur	15

4.4.4	Début et fin de séquence	16
4.4.5	L'identifiant de la séquence sur le site Uniprot	16
4.4.6	L'orthologie	17
4.4.7	Class et Cluster	17
4.4.8	Assemblage d'informations	19
4.5	Insertion des données dans la base	19
4.6	Mise à jour du site web	19
4.7	Conclusion au sujet des réalisations	20
5	Ce qu'il reste à faire pour une analyse complète	21
5.1	Analyse phylogénétique	21
5.2	Modélisation d'un modèle 3D par homologie.	21
5.3	Docking moléculaire.	22
6	Connaissances acquises	23
7	Abstract	24
8	Résumé	24
9	Référence	25

Table des figures

4.1	Schéma global des étapes à réaliser	10
4.2	Code Bash pour créer un virtualenv	11
4.3	Code du script search_obp.sh	11
4.4	Schéma expliquant le script bash search_obp.sh avant amélioration du code	12
4.5	Schéma expliquant le script bash search_obp.sh après amélioration du code	12
4.6	Schéma expliquant le script bash rbh.sh avant amélioration du code	13
4.7	Schéma expliquant le script bash rbh.sh après amélioration du code	14
4.8	Exemple de fichier csv	14
4.9	Code Bash permettant de récupérer la localisation	15
4.10	Code Bash permettant de récupérer la longueur	15
4.11	Code Bash pour récupérer le début de chaque séquence	16
4.12	Code Bash pour récupérer le fin de chaque séquence	16
4.13	Code Python récupération des entrée Uniprot version 1	16
4.14	Code Python récupération des entrée Uniprot version 2	16
4.15	Code Python pour la récupération des orthologues	17
4.16	Algorithme expliquant le script pour la récupération de la classe et du cluster	18
4.17	Exemple de fiche d'identité créée	19
4.18	Schéma global des étapes réalisées	20
5.1	Analyse génétique	21

1 Remerciements

Avant d'exposer mon expérience professionnelle dans ce laboratoire, je voudrais tout d'abord remercier les personnes travaillant dans cette structure pour leur bon accueil et leur gentillesse, ainsi que les étudiants réalisant leurs thèses et qui étaient présents avec moi. Ils ont su me mettre à l'aise et permettre que mon stage se déroule dans une ambiance conviviale. Leur gentillesse a permis de faire de ce stage un moment très profitable.

Je remercie en particulier Monsieur Bernard Offmann, mon maître de stage qui m'a formée et accompagnée tout au long de l'apprentissage sur l'utilisation des outils informatiques et dans cette expérience professionnelle.

2 Introduction

2.1 Contexte

Les moustiques sont des insectes qui transmettent facilement de nombreuses maladies qui peuvent être fatales pour les Hommes ; on peut citer par exemple le paludisme transmis par le moustique *Anopheles gambiae*. Les moustiques sont attirés par leur cible grâce, entre autre, aux molécules odorantes et aux gaz qui sont libérés par les humains. Les molécules odorantes sont hydrophobes et ont donc besoin d'être transportés aux récepteurs à la surface des membranes plasmiques des neurones olfactifs. Le blocage de ce transport permettrait d'éviter l'attraction entre les moustiques et leur cible et ainsi éviter la transmission de maladies [1] [2].

2.2 Les OBP

Les OBPs (odorant binding protein) sont des molécules essentielles dans la reconnaissance et le transport des molécules odorantes. Ce type de protéines possède 6 segments hélicoïdaux et trois ponts disulfures très conservés. Ces protéines sont regroupées en trois familles avec chacune leurs caractéristiques : *Classic* (avec six cystéines qui sont typiques de la famille des OBPs), *PlusC* (avec six cystéines supplémentaires dont deux très conservées) et *Atypical* (composées de deux domaines classic) [1] [2].

Afin de consolider les connaissances sur les OBPs, l'UFIP a développé une base de données pour répertorier toutes les OBPs qui puissent exister chez les moustiques. Elle est connue sous le nom mOBPdb pour mosquito Odorant Binding Protein database [3].

2.3 La base de données mOBPdb

La base de données a été développée en 2014. Au début du stage, elle comprenait 69 séquences d'OBPs pour l'espèce d'*Anopheles gambiae*, 111 OBPs de *Aedes aegypti* et 109 OBPs de *Culex quinquefasciatus*. On y trouve diverses informations sur les OBPs telles que la localisation des gènes, la phylogénie, l'orthologie et éventuellement, un modèle structural. La visualisation de cette base de données est permise via le site web mosquito odorant binding protein database aussi appelé mOBPdb (<http://www.bo-protscience.fr/mobpdb/>).

L'objectif de cette base de données est de fournir des informations aux biologistes permettant de comprendre l'évolution des gènes de l'olfaction et les adaptations des moustiques à l'environnement.

2.4 Objectifs du stage

Le premier objectif est donc de comprendre et d'analyser le fonctionnement de cette base de données pour mieux se l'approprier. Cette étape est essentielle et primordiale dans la réalisation de ce stage car elle permet de mieux comprendre le contexte et ainsi pouvoir répondre au mieux aux besoins. Le second objectif de ce stage est d'analyser et d'améliorer un script existant pour le criblage de génomes de moustiques à l'aide d'outils standards de recherche de séquences par homologie. Le troisième consiste à améliorer un script pour la recherche d'orthologues des OBPs identifiés précédemment. Le quatrième objectif concerne l'intégration des données obtenues dans la base de données. Enfin, le dernier consiste à rendre visible le contenu de la base de données via une interface web s'appuyant sur une technologie CGI et PHP.

Le but final est donc d'alimenter cette base de données par l'analyse de 16 nouveaux génomes d'insecte. Ces nouveaux génomes ont été séquencés et publiés en 2016 dans Science [6].

3 Cahier des charges techniques

Suite à un entretien avec Monsieur OFFMANN, les différents besoins de ce projet ont été définis.

3.1 Connaissances sur les OBPs

Une bonne connaissance des OBPs du point de vue structurale et fonctionnelle est possible par la consultation d'articles scientifiques. On peut citer en particulier les articles de Pelletier J, Leal WS (2009) [1] et de Manoharan et al (2014) [2].

3.2 Récupération des données de génomes

Des génomes complets sont disponibles sur le site Vectorbase (<https://www.vectorbase.org>). Ils y sont téléchargeables sous différentes formes allant du transcrit aux peptides. La forme la plus adaptée pour ce projet et la forme peptides qui représente le protéome complet d'un organisme. Après téléchargement, le protéome de l'organisme se situe dans un dossier compressé comprenant un fichier unique sous l'extension .fa (représentant le format fasta) qui comprend toutes les séquences qui composent le protéome de l'organisme. Le format fasta est un format de fichier texte utilisé pour stocker des séquences biologiques de nature nucléique ou protéique. Chaque séquence est représentée par une suite de lettres représentant une suite d'acides aminés ou une suite d'acides nucléiques. Cette séquence est précédée par une ligne démarrant par le signe ">" (signe obligatoire) suivi immédiatement de l'identifiant de la séquence.

3.3 Recherche d'OBPs par homologie

L'autre besoin consiste à analyser et améliorer un script déjà existant pour trouver de nouvelles OBPs dans un génome nouvellement séquencé à l'aide d'un jeu de séquences d'OBP connues dit de référence. Ce script est codé en Bash et utilise la suite BLAST (Makeblast et PSI-Blast). Pour appliquer ce script à ce projet, il m'a été demandé de le faire fonctionner en ayant regroupé toutes les séquences de référence dans un seul et même fichier.

3.4 Recherche des orthologues

Il m'a également été demandé d'analyser et d'améliorer un script déjà existant pour trouver des orthologues aux nouvelles OBPs d'un génome nouvellement séquencé. Ce script est codé en Bash et utilise la suite BLAST (Makeblast et PSI-Blast). Pour appliquer ce script à ce projet, il m'a été demandé de gérer différemment les fichiers de séquences.

3.5 Récupération d'informations

Une autre demande a été de développer des scripts de récupération des informations associées à chaque OBP détectée. La demande concerne la récupération d'informations précises : localisation des gènes, longueur de la séquence, l'entrée Uniprot, la class et le cluster de chaque séquence.

3.6 Intégration des informations dans la base de données déjà existante

La demande finale a été d'une part, l'intégration des données dans la base de données du site web, et d'autre part, la mise à jour du site web avec des informations correspondant aux nouvelles espèces.

3.7 Les outils

Il m'a été demandé de me familiariser avec divers outils informatiques qui m'ont été essentiels dans la réalisation de mon stage.

3.7.1 *Psi Blast*

PSI-BLAST est un outil qui fait partie de la suite BLAST. BLAST est un outil de bioinformatique qui permet de comparer des séquences biologiques. C'est un puissant outil fonctionnant sur le principe de l'homologie entre deux séquences. Une séquence protéique ou nucléotidique (dite cible) sera comparée à des séquences de même nature, présentes dans une base de données, par alignement local. Lors d'un alignement local, chaque lettre formant la séquence cible (l'ordre des lettres est conservé) sera comparée et associée à une autre lettre de la chaîne de la base de données. Le résultat de la correspondance entre ces deux séquences est unique et correspond au score maximal de correspondance entre ces séquences.

PSI-BLAST est un dérivé de BLAST car il permet une succession d'alignements locaux. Contrairement à BLAST, PSI-BLAST continue à faire les itérations d'alignements locaux tant qu'il y a des correspondances entre les séquences ce qui permet de corriger le score de correspondance. Cet outil est disponible en ligne sur le site du NCBI mais il est aussi possible de télécharger le package BLAST pour l'utiliser en ligne de commande.

3.7.2 *phpMyAdmin*

PhpMyAdmin est une application Web gratuite écrit en PHP. Cette application permet la gestion sécurisée de base de données MySQL, notamment à distance.

3.7.3 Wordpress

WordPress est un système de gestion de contenu de site web (SGC ou content management system (CMS)) gratuit et libre. Ce logiciel est écrit en PHP et repose sur une base de données MySQL. WordPress permet de créer et gérer différents types de sites Web : blog, site e-commerce, site vitrine, ...

3.8 Langages utilisés : Bash et Python

Durant mon stage, j'ai dû élaborer des fonctions à l'aide des langages de scripts bash et python. A cette fin, il m'a été demandé de consolider mes connaissances pour ces deux langages.

3.8.1 Bash

Le langage Bash est commun à tous les environnements Unix et il est un langage de commande qui interagit principalement avec les dossiers, fichiers et les éléments présents dans un ordinateur.

3.8.2 Python

Contrairement au Bash, mes connaissances en Python au début du stage étaient très faibles. Il a donc fallu que je me documente à propos de ce langage pour le découvrir et ainsi mieux le comprendre. Ce langage est très utilisé dans le domaine de l'informatique mais également dans la bioinformatique. Python est un langage extensible par de nombreux modules. Il possède donc de nombreuses capacités utilisables dans tous les domaines (de la biologie à la gestion des fichiers et à l'interaction avec le web).

3.9 Recherche nom d'entrée Uniprot

Il m'a également été demandé de récupérer le code d'identification Uniprot pour chaque séquence. Pour effectuer ce script, il a été nécessaire de coder l'API (service fourni par Uniprot) pour retrouver le nom d'entrée de chaque séquence trouvée à l'aide de Blast. Uniprot fournit une ébauche de ce script dans différents langages. J'ai décidé d'utiliser la version en python de cette API pour développer et approfondir mes connaissances dans ce langage.

4 Ce qui a été fait

Ce stage (Figure : 4.1) a été réalisé dans un environnement Linux (version 16 d'Ubuntu).

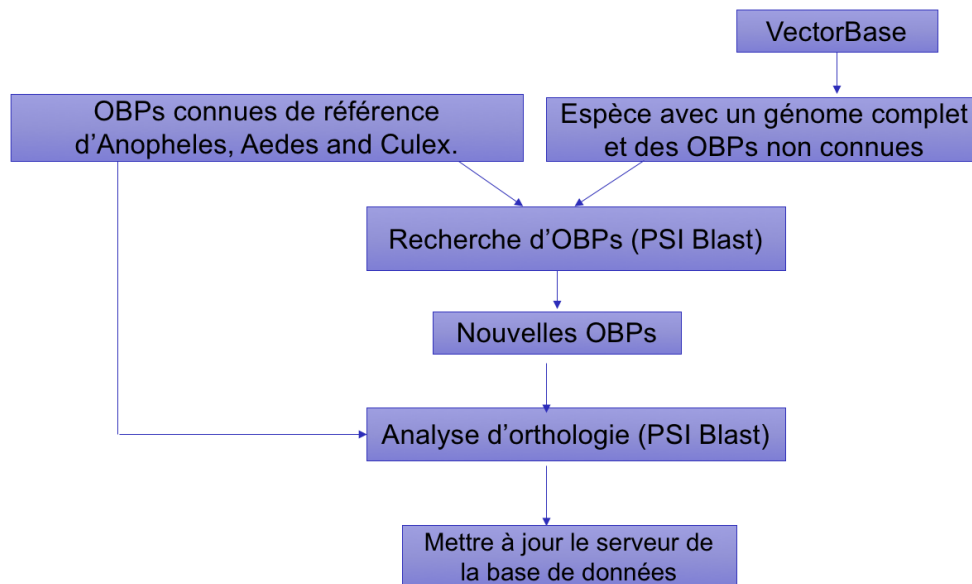


FIGURE 4.1 – Schéma global des étapes à réaliser pour permettre la mise à jour de la base de données. Les différents génome viennent du site Vectorbase.

4.1 Mise en place de l'environnement de travail

Pour faciliter la gestion des programmes et des packages associés, il est possible d'utiliser l'outil virtualenv (Figure : 4.2). La création d'un environnement virtuel permet également d'installer des packages particuliers, qui ne sont pas encore installés, en évitant les problèmes de permissions. Pour mon projet, l'environnement est surtout utilisé pour le langage Python. L'outil virtualenv crée un nouvel environnement Python en tant que dossier qui contient l'ensemble des packages dont a besoin Python pour exécuter les scripts.

```
# Installation d'un virtualenv de python par le terminal
virtualenv name
# activation/entrée dans l'environnement virtuel
source name/bin/activate
# Installation des packages nécessaire pour
# utiliser python et BLAST
pip install numpy
pip install biopython
# Installation de la suite BLAST+
sudo apt-get install ncbi-blast+
```

FIGURE 4.2 – Code Bash pour créer un virtualenv et ainsi permettre l'installation de la suite BLAST.

Cependant, l'installation de BLAST n'a pas pu être effectuée dans le virtualenv. C'est pourquoi, avant chaque script, il faut charger le module Blast installé sur le serveur de l'UFIP grâce à la commande "module load blast".

4.2 Recherche d'OBPs dans un génome inconnu

J'ai analysé un script Bash existant qui permet la recherche d'OBPs par homologie (search_obp.sh). Ce script nécessite l'utilisation d'un ensemble d'OBPs connues de référence ainsi qu'un seul et unique fichier qui contient toutes les protéines provenant du génome complet d'insecte. Ce script nécessite l'installation de la suite BLAST car utilisant MAKEBLAST (création de base de données) et PSI-BLAST (itération d'alignement local) (Figure : 4.3). Pour limiter le nombre de résultats peu plausibles, une e-value de $3e-10$ et un taux de recouvrement avec la séquence de référence supérieur à 75% ont été choisis.

```
#generate a database for PSIBLAST analysis for the
#new fasta file of target proteome
makeblastdb -in $2/$2_new.fasta -dbtype prot

# Perform psiblast. Output is named according to the first 2 arguments
psiblast -query $1/$1.fasta -db $2/$2_new.fasta -outfmt "7 std qcovs"
-evalue 3e-10 -out $2/$1_$2/$1_$2.txt -num_iterations 10 -num_threads $mythreads
```

FIGURE 4.3 – Une partie du code du script search_obp.sh montrant l'utilisation de makeblast et psi-blast ainsi que la contrainte de e-value.

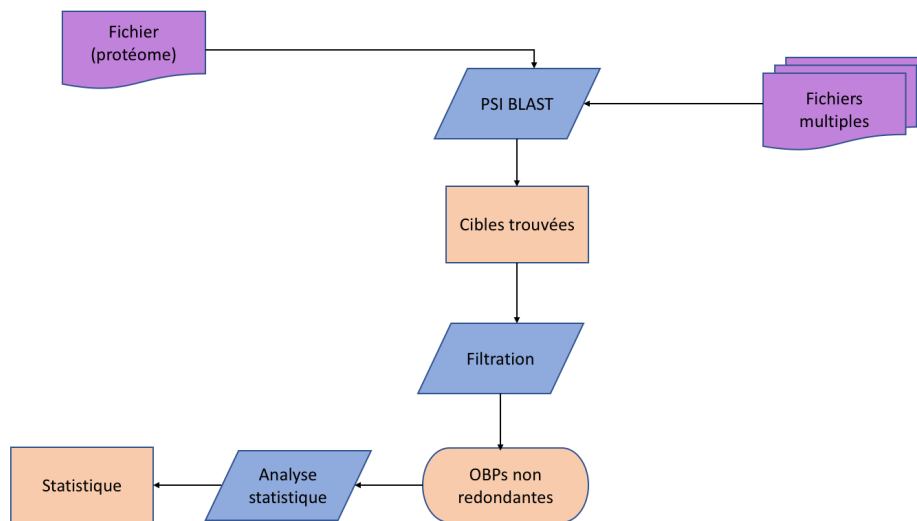


FIGURE 4.4 – Schéma expliquant le script bash search_obp.sh avant amélioration du code. En violet, il s’agit des fichiers mis en entrée. En bleu, il s’agit des divers processus. Et en jaunes, il s’agit des résultats obtenus à la sortie de chaque processus.

J’ai amélioré des éléments de ce script pré-existant (Figure : 4.4). Il a tout d’abord fallu simplifier la gestion des séquences de référence. Ainsi, uniquement un fichier contenant toutes les séquences de référence de tous les organismes (renommées de façon à avoir avant le nom de chaque séquence le préfixe "REF_") est nécessaire et non un fichier par organisme. J’ai également créé des scripts qui facilitent la décompression du génome complet et la bonne présentation des séquences. Grâce à ces scripts intermédiaires, les fichiers sont prêts pour pouvoir exécuter le script de recherche d’OBPs (Figure : 4.5).

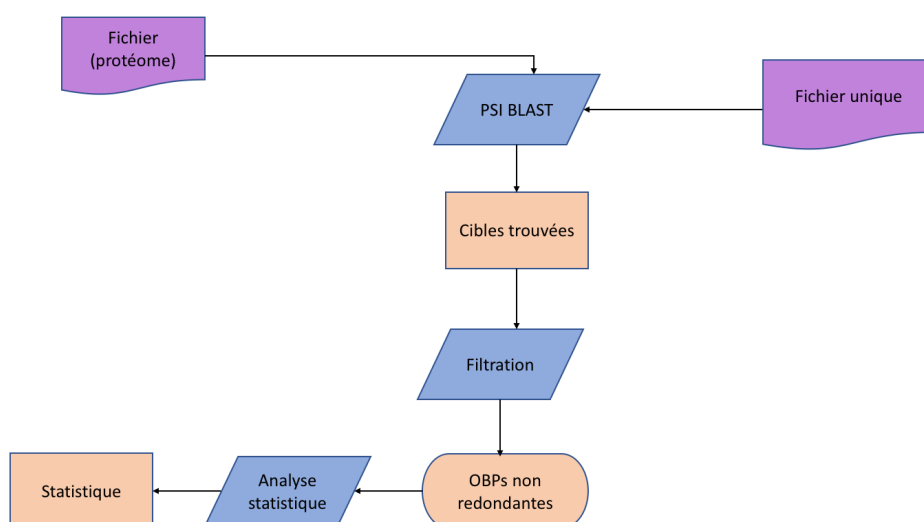


FIGURE 4.5 – Schéma expliquant le script bash search_obp.sh après amélioration du code. En violet, il s’agit des fichiers mis en entrée. En bleu, il s’agit des divers processus. Et en jaunes, il s’agit des résultats obtenus à la sortie de chaque processus.

4.3 Recherche d'orthologues pour les nouvelles OBPs trouvées

J'ai analysé un script bash existant qui permet la recherche d'orthologues entre chaque nouvelle OBP trouvée et les OBPs de référence (rbh.sh). Le langage utilisé est aussi du bash. Ce script utilise aussi un fichier contenant toutes les séquences de tous les organismes mais il a également besoin que chaque organisme de référence soit dans des dossiers différents ainsi que chaque séquence dans un fichier distinct. Les nouvelles OBPs trouvées par le script précédent doivent toutes être dans un seul et unique fichier. Ce script a aussi besoin de la suite BLAST pour MAKEBLAST et PSI-BLAST et il doit être exécuté autant de fois qu'il y a d'organismes modèles.

La version initiale de ce script prend toutes les séquences de référence dans un seul et même fichier quelque soit les organismes (Figure : 4.6). Cependant, lors de l'exécution, il s'est avéré que seul le premier organisme modèle était comparé et donc que le programme s'arrêtait à la première correspondance. Ainsi, c'est pourquoi une division par organisme de référence permet la comparaison de chaque organisme avec chaque OBPs et des orthologues peuvent être trouvés pour chaque espèce de référence.

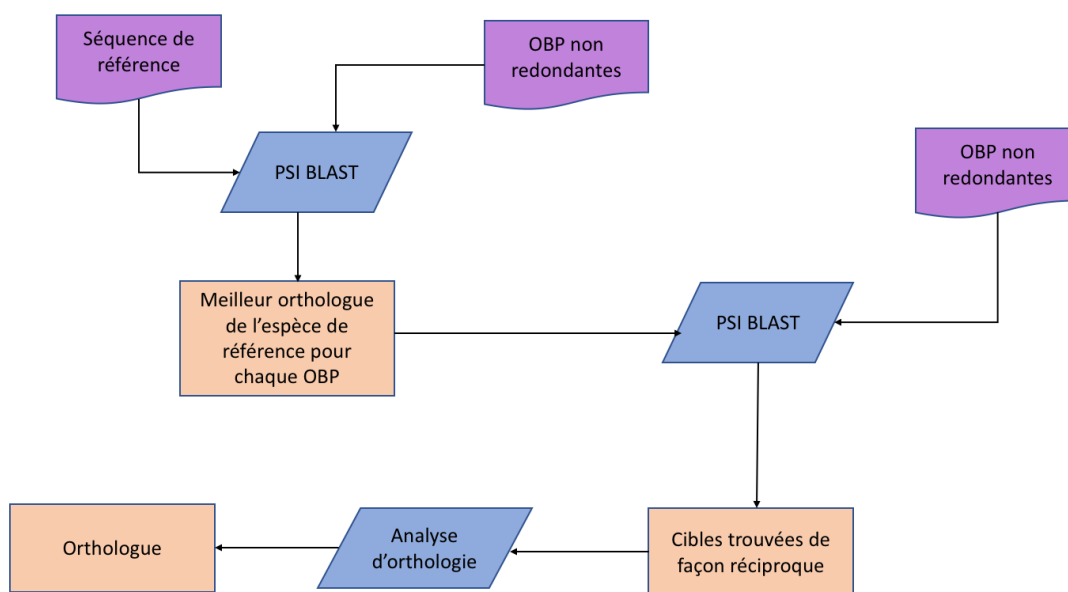


FIGURE 4.6 – Schéma expliquant le script bash rbh.sh avant amélioration du code. En violet, il s'agit des fichiers mis en entrée. En bleu, il s'agit des divers processus. Et en jaunes, il s'agit des résultats obtenus à la sortie de chaque processus.

J'ai donc amélioré le script rbh.sh en créant un script qui permet le découpage du fichier contenant les cibles trouvées par de search_obp.sh pour au final avoir une séquence par fichier qui a été renommé par le nom de la séquence (Figure : 4.7). J'ai également créé un script qui permet de mettre chaque séquence de référence dans un fichier.

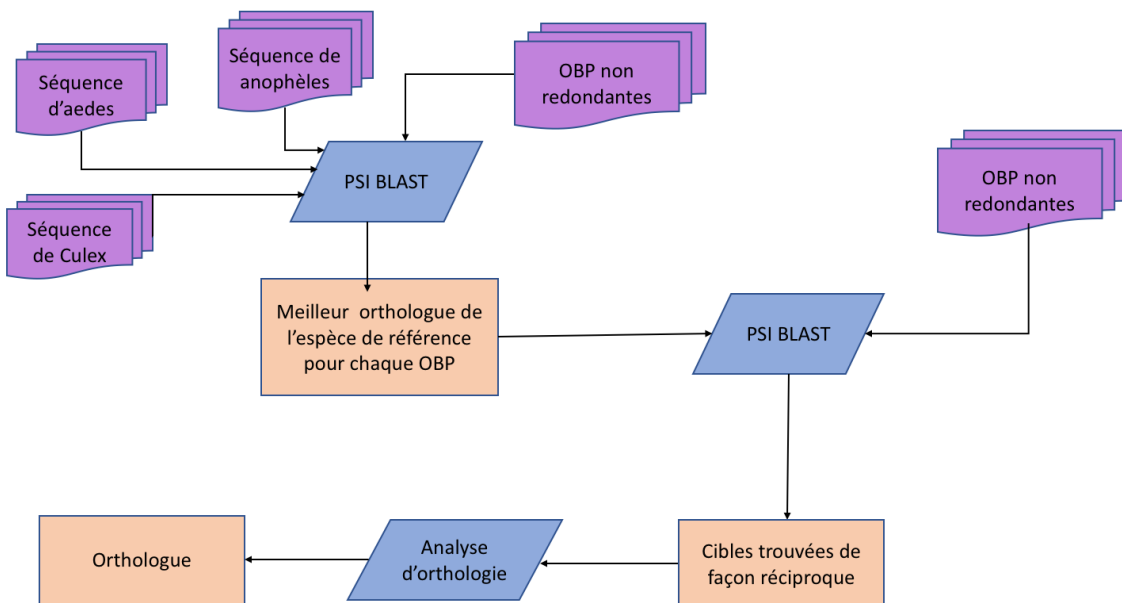


FIGURE 4.7 – Schéma expliquant le script bash rbh.sh après amélioration du code. En violet, il s'agit des fichiers mis en entrée. En bleu, il s'agit des divers processus. Et en jaunes, il s'agit des résultats obtenus à la sortie de chaque processus.

4.4 Récupération des diverses informations

4.4.1 Quel format ?

Pour que les informations sur chaque séquences puissent être entrées dans la base, il faut les présenter dans un fichier csv respectant le format d'entrée de la base de données (Figure : 4.8).

ID	name	entry	length	localisation	phylogeny	orthology					
ID	Name	Uniprot	Length	Chr	Start	End	Class	Cluster	Orth_Anopeles	Orth_Aedes	Orth_Culex
AGAP001189	agamobp10	F7IW19	131	2R	1034139	1169444	Classic	mclassic6	NA	aaegobp10	cquiobp24
AGAP002025	agamobp11	Q8I8Q9	192	2R	14069095	14069749	Classic	mclassic9	NA	NA	NA
AGAP002189	agamobp14	Q8I8T3	188	2R	17331871	17332553	Classic	mclassic9	NA	aaegobp18	cquiobp63
AGAP008398	agamobp21	Q8I8S3	131	3R	10317255	10317835	Classic	mclassic5	NA	aaegobp8	cquiobp23
AGAP010409	agamobp22	Q7PGA3	144	3L	2853087	2853645	Classic	mclassic8	NA	aaegobp81	cquiobp44
AGAP012318	agamobp23	Q8I8R9	131	3L	40168852	40169329	Classic	mclassic3	NA	aaegobp9	cquiobp22
AGAP012319	agamobp24	Q8I8R8	176	3L	40171315	40172237	Classic	mclassic2	NA	aaegobp77	cquiobp21
AGAP012320	agamobp25	Q7Q088	142	3L	40209434	40210326	Classic	mclassic3	NA	aaegobp11	cquiobp19
AGAP012321	agamobp26	Q8I8R6	131	3L	40213816	40214477	Classic	mclassic3	NA	aaegobp35	cquiobp20
AGAP012323	agamobp27	Q5TN65	134	3L	40218226	40218764	Classic	mclassic1	NA	aaegobp65	cquiobp16
AGAP012325	agamobp28	Q8I8R4	134	3L	40221011	40221620	Classic	mclassic3	NA	aaegobp12	cquiobp18
AGAP012322	agamobp63	Q6H900	135	3L	40217381	40217853	Classic	mclassic4	NA	aaegobp61	cquiobp17

FIGURE 4.8 – Exemple de fichier (contenant les informations des trois génomes déjà présents pour les OBPS Classics) montrant le format d'entrée de la base de données avec les diverses informations.

4.4.2 Localisation

J'ai tout d'abord créé un script pour récupérer la localisation du gène sur le chromosome. Pour cela, il est nécessaire d'avoir deux fichiers : le fichier original contenant les séquences ainsi que leurs informations et le fichier contenant les OBPs trouvées par le script de recherche d'OBPs. Ainsi, il s'agit de rechercher dans le fichier contenant toutes les séquences du génome, le nom des OBPs pour permettre de récupérer l'information sur la localisation de chaque gène (Figure : 4.9).

```
# The file containing all the hits found by search_obp.sh is used
fichier=`cat uniqhits`

# Get back lines containing all the information in the file fasta on hits found
#by search_obp.sh
for ligne in $fichier; do # For all the hits of the file
cat $1.fasta | grep $ligne # Search for the name of sequence in the file and the recovery
#of the line containing the information
done >total.txt # This information are saved in a text file

# Recovery only of the line where is present all the information on sequences
cut -d '|' -f 3 total.txt >info.txt

# Recovery position in chromosome
cut -d ':' -f 1 info.txt >non.txt
sed 's/supercont//g' non.txt >localisation.txt

# Erase text files intermediary which do not serve us any more
rm total.txt
rm non.txt
```

FIGURE 4.9 – Code Bash permettant de récupérer la localisation de chaque séquence à partir d'un fichier créé lors de ce script et qui contient toutes les informations sur chaque séquence d'une espèce.

4.4.3 Longueur

Ensuite, j'ai créé un script pour récupérer la longueur du gène. Pour cela, il est nécessaire d'avoir le fichier csv de sortie du script de recherche d'OBPs et de récupérer l'information (Figure : 4.10).

```
#Recovery of the column contening length of sequences
cut -d ';' -f 2 statistics.csv >length.txt
```

FIGURE 4.10 – Code Bash permettant de récupérer la longueur de chaque séquence à partir du fichier statistics.csv créé par le script search_obp.sh et qui contient les informations liées à chaque séquence d'une espèce.

4.4.4 Début et fin de séquence

Puis j'ai créé un script pour récupérer le début et la fin de séquence du gène sur les chromosomes ou contigs à partir du fichier d'origine contenant toutes les séquences (Figures : 4.11 et 4.12).

```
#Recovery of the start of the sequence
cut -d ':' -f 2 info.txt >start.txt
```

FIGURE 4.11 – Code Bash pour récupérer le début de chaque séquence à partir du fichier créé par le script localisation.sh et qui contient les informations liées à chaque séquence d'une espèce.

```
#Recovery of the end of the sequence
cut -d ':' -f 3 info.txt >end.txt
```

FIGURE 4.12 – Code Bash pour récupérer le fin de chaque séquence à partir du fichier créé par le script localisation.sh et qui contient les informations liées à chaque séquence d'une espèce.

4.4.5 L'identifiant de la séquence sur le site Uniprot

En utilisant le langage Python et un outils (API) du site internet Uniprot, j'ai créé un script qui permet de récupérer les identifiants sous lesquels sont connues nos séquences sur le site Uniprot. Il s'agissait donc de coder cet outil tout en l'adaptant à notre situation. La contrainte principale était de modifier le code pour qu'il renvoie NA si aucune entrée Uniprot n'est disponible pour une séquence.

Cependant, avec les paramètres de la version 1 (Figure : 4.13), peu d'entrées étaient trouvées. Après de nombreuses recherches, de nouveaux paramètres ont été choisis (Figure : 4.14). La dernière contrainte est de renommer les séquences et remplacer les suffixe -PA, -PB en -RA, -RB. Par exemple : remplacer AALF000521-PA en AALF000521-RA.

```
import urllib,urllib2

url = 'http://www.uniprot.org/uploadlists/'

fichier = open("hits.txt", "r")

#hits=fichier.readline()

for hit in fichier:
    params = {
        'from':'GENENAME',
        'to':'ACC',
        'format':'list',
        'query': hit }
    data = urllib.urlencode(params)
    request = urllib2.Request(url, data)
    contact = "amandine.lecerf-defer@etu.univ-nantes.fr"
    request.add_header('User-Agent', 'Python %s' % contact)
    response = urllib2.urlopen(request)
    page = response.read(200000)
    if not page:
        #print "%s %s"%(hit,"NA")
        print (hit,"NA")
    else :
        #print "%s %s" % (hit,page)
        print (hit,page)
fichier.close()
```

FIGURE 4.13 – Code Python récupération des entrée Uniprot version 1.

```
import urllib,urllib2

url = 'http://www.uniprot.org/uploadlists/'

fichier = open("hits.txt", "r")

#hits=fichier.readline()

for hit in fichier:
    params = {
        'from':'VECTORBASE_ID',
        'to':'ID',
        'format':'list',
        'query': hit }
    data = urllib.urlencode(params)
    request = urllib2.Request(url, data)
    contact = "amandine.lecerf-defer@etu.univ-nantes.fr"
    request.add_header('User-Agent', 'Python %s' % contact)
    response = urllib2.urlopen(request)
    page = response.read(200000)
    if not page:
        #print "%s %s"%(hit,"NA")
        print (hit,"NA")
    else :
        #print "%s %s" % (hit,page)
        print (hit,page)
fichier.close()
```

FIGURE 4.14 – Code Python pour la récupération des entrée Uniprot version 2.

La conception de ce script a été possible en améliorant et en adaptant l'API de Uniprot mise à disposition pour retrouver le nom d'identification de séquence contenues dans le site Uniprot. Cette API peut être utilisée en ligne de code ou directement sur le site internet de Uniprot [5].

API signifie Application Programming Interface, c'est donc un moyen mis en place par un logiciel afin que d'autres logiciels puissent interagir avec. Dans notre cas, cette API permet de faire le lien entre notre terminal, qui exécute le script python, et le site internet Uniprot sur lequel on veut récupérer des informations.

4.4.6 L'orthologie

J'ai ensuite créé un script Python qui permet de récupérer les orthologues d'une séquence chez les organismes de référence. Si aucun lien d'orthologie n'est retrouvé entre une séquence et un autre organisme, le script doit renvoyer NA. Il faut lancer le script autant de fois qu'il y a d'organismes de référence (Figure : 4.15).

```
import os
os.system("sed 's/-RA/-PA/g; s/-RB/-PB/g; s/-RC/-PC/g' hits.txt >hits2.txt")
os.system("cut -d ' ' -f 1,2 palpilis_culex_orthology_detected.txt >ortho.txt")
#avant : faire la commande : sed 's/-RA//g; s/-RB//g' hits.txt >hits2.txt
# faire cut -d ' ' -f 1,2 dirus_aedes_orthology_detected.txt >ortho.txt

#from __future__ import with_statement, print_function

with open("hits2.txt") as fhits, open("ortho.txt") as fortho:
    hits = [hit.strip() for hit in fhits.readlines()]
    ortho = dict()
    for line in fortho.readlines():
        k, v = line.split(' REF_')
        ortho[k.strip()] = v.strip()

    for hit in hits:
        print(ortho.get(hit, 'NA'))

os.system("rm hits2.txt")
os.system("rm ortho.txt")
```

FIGURE 4.15 – Code Python pour la récupération des orthologues à partir du fichier contenant les OBP trouvées par search_obp.sh et du fichier contenant les orthologues trouvés par rbh.sh.

4.4.7 Class et Cluster

J'ai conçu un script Bash qui compare les noms des orthologues des séquences présentes dans le fichier csv, créé précédemment, avec une base de données comportant toutes les informations pour toutes les séquences de références déjà présentes sur le site. Ainsi, par homologie, il est possible de dire qu'une nouvelle OBP et son orthologues possèdent la même classe et le même cluster. Il sera nécessaire de refaire le script précédent en prenant en compte ces nouvelles informations récupérées pour que le tableau soit complet (Figure : 4.16).

Après avoir récupéré toutes ces informations et réalisé le tableau csv au format souhaité, il est possible de rentrer toutes ces données sur le serveur du site web.

```

DEBUT ALGORITHME
TANT QUE (la dernière ligne du fichier n'a pas été lue) FAIRE
    RECUPERER "L'orthologue chez Anophles "
    SI (L'orthologue différent de "NA") ALORS
        RECHERCHER "L'orthologue dans la base de données."
        SI (L'orthologue est trouvé) ALORS
            AFFICHER "La classe et le cluster."
        SINON
            RECUPERER " L'orthologue chez Aedes"
            SI (L'orthologue est différent de "NA") ALORS
                RECHERCHER " L'orthologue dans la base de données."
                SI (L'orthologue est trouvé) ALORS
                    AFFICHER " La classe et le cluster."
                SINON
                    RECUPERER " L'orthologue chez Culex"
                    SI (L'orthologue est différent de "NA") ALORS
                        RECHERCHER " L'orthologue dans la base de données."
                        SI (L'orthologue est trouvé) ALORS
                            AFFICHER " La classe et le cluster."
                        SINON
                            AFFICHER "NA;NA ; "
                        FIN SI
                    SINON
                        AFFICHER "NA;NA ; "
                    FIN SI
                FIN SI
            SINON
                RECUPERER " L'orthologue chez Culex"
                SI (L'orthologue est trouvé) ALORS
                    RECHERCHER " L'orthologue dans la base de données."
                    SI (L'orthologue est trouvé) ALORS
                        AFFICHER " La classe et le cluster."
                    SINON
                        AFFICHER "NA;NA ; "
                    FIN SI
                SINON
                    AFFICHER "NA;NA ; "
                FIN SI
            FIN SI
        FIN SI
    SINON
        RECUPERER " L'orthologue chez Aedes"
        SI (L'orthologue est différent de "NA") ALORS
            RECHERCHER " L'orthologue dans la base de données."
            SI (L'orthologue est trouvé) ALORS
                AFFICHER " La classe et le cluster."
            SINON
                RECUPERER " L'orthologue chez Culex"
                SI (L'orthologue est différent de "NA") ALORS
                    RECHERCHER " L'orthologue dans la base de données."
                    SI (L'orthologue est trouvé) ALORS
                        AFFICHER " La classe et le cluster."
                    SINON
                        AFFICHER "NA;NA ; "
                    FIN SI
                SINON
                    AFFICHER "NA;NA ; "
                FIN SI
            FIN SI
        SINON
            RECUPERER " L'orthologue chez Culex"
            SI (L'orthologue est différent de "NA") ALORS
                RECHERCHER " L'orthologue dans la base de données."
                SI (L'orthologue est trouvé) ALORS
                    AFFICHER " La classe et le cluster."
                SINON
                    AFFICHER "NA;NA ; "
                FIN SI
            SINON
                AFFICHER "NA;NA ; "
            FIN SI
        FIN SI
    FIN SI
FIN TANT QUE
FIN ALGORITHME

```

FIGURE 4.16 – Algorithme expliquant le script pour la récupération de la classe et du cluster à partir des orthologues d'une séquence.

4.4.8 Assemblage d'informations

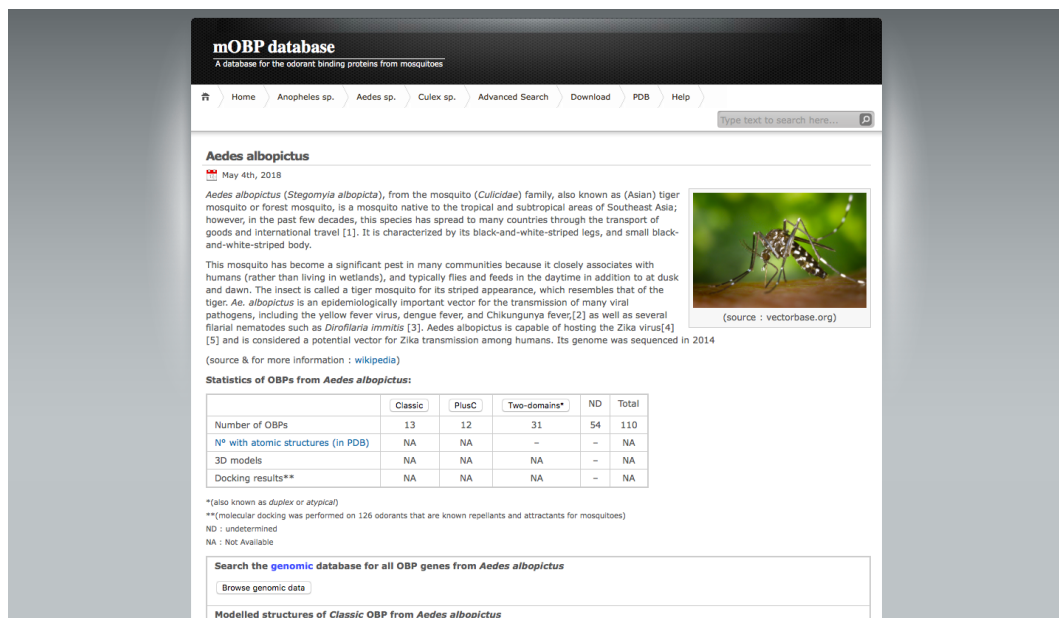
J'ai également créé un script bash qui permet de regrouper toutes ces informations pour donner en sortie un fichier csv.

4.5 Insertion des données dans la base

L'application phpMyAdmin permet d'intégrer les fichiers csv créés précédemment et contenant toutes les informations pour chaque organisme. Cette étape permet de mettre à jour la base de données avec de nouvelles informations sur de nouvelles OBPs.

4.6 Mise à jour du site web

J'ai recherché des informations sur chaque espèce nouvellement séquencée (surtout sur Vectorbase) afin de créer une fiche d'identité pour chaque espèce (Figure : 4.17). Cette fiche d'identité comprend une bibliographie de l'espèce, une photo, un tableau reprenant les informations obtenues sur les OBPs et enfin, la possibilité de télécharger le fichier csv contenant les OBPs trouvées. L'ajout de ces fiches est possible grâce à Wordpress qui permet la gestion des pages web.



mOBP database
A database for the odorant binding proteins from mosquitoes

Home Anopheles sp. Aedes sp. Culex sp. Advanced Search Download PDB Help

Type text to search here...

Aedes albopictus

May 4th, 2018

Aedes albopictus (*Stegomyia albopicta*), from the mosquito (*Culicidae*) family, also known as (Asian) tiger mosquito or forest mosquito, is a mosquito native to the tropical and subtropical areas of Southeast Asia; however, in the past few decades, this species has spread to many countries through the transport of goods and international travel [1]. It is characterized by its black-and-white-striped legs, and small black-and-white-striped body.

This mosquito has become a significant pest in many communities because it closely associates with humans (rather than living in wetlands), and typically flies and feeds in the daytime in addition to at dusk and dawn. The insect is called a tiger mosquito for its striped appearance, which resembles that of the tiger. *Ae. albopictus* is an epidemiologically important vector for the transmission of many viral pathogens, including the yellow fever virus, dengue fever, and Chikungunya fever [2] as well as several filarial nematodes such as *Dirofilaria immitis* [3]. *Aedes albopictus* is capable of hosting the Zika virus [4] [5] and is considered a potential vector for Zika transmission among humans. Its genome was sequenced in 2014

(source & for more information : wikipedia)

Statistics of OBPs from *Aedes albopictus*:

	Classic	PlusC	Two-domains*	ND	Total
Number of OBPs	13	12	31	54	110
N° with atomic structures (in PDB)	NA	NA	-	-	NA
3D models	NA	NA	NA	-	NA
Docking results**	NA	NA	NA	-	NA

*(also known as duplex or atypical)
**(molecular docking was performed on 126 odorants that are known repellents and attractants for mosquitoes)
ND : undetermined
NA : Not Available

Search the **genomic** database for all OBP genes from *Aedes albopictus*

Browse genomic data

Modelled structures of Classic OBP from *Aedes albopictus*

FIGURE 4.17 – Exemple de fiche d'identité créée contenant une image de espèce, une bibliographie et les statistiques sur les OBPs de cette espèce.

4.7 Conclusion au sujet des réalisations

Les scripts utilisés ici, permettant la réalisation des diverses étapes, ainsi que l'utilisation d'outils utiles à la bioinformatique, comme par exemple PSIBLAST, ont permis de mettre à jour le site web mOBPdb avec des espèces de moustique nouvellement séquencées (Figure : 4.18).

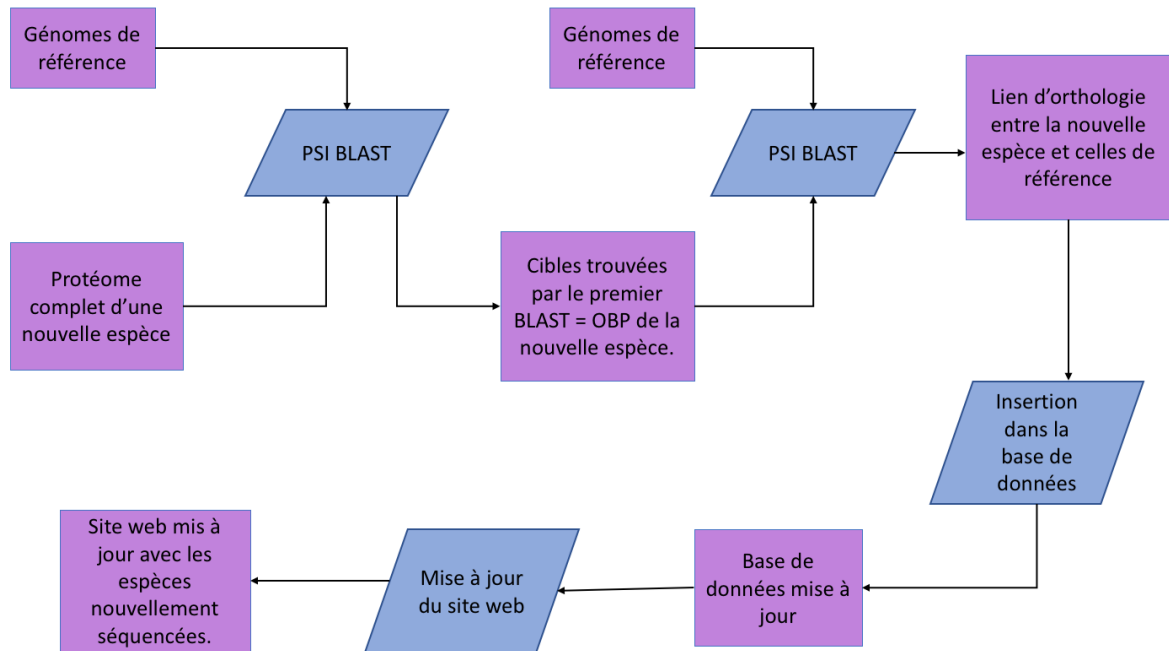


FIGURE 4.18 – Schéma global des étapes effectuées lors de ce stage pour la réalisation du projet.

5 Ce qu'il reste à faire pour une analyse complète

5.1 Analyse phylogénétique

Après avoir déterminé les liens d'orthologie avec ces nouvelles OBPs, il serait nécessaire d'analyser la phylogénie des orthologues et des paralogues d'OBPs. Cette analyse par alignement de séquences et reconstruction phylogénétique permettra de voir l'évolution des OBPs ainsi que des différentes sous-familles (Figure : 5.1).

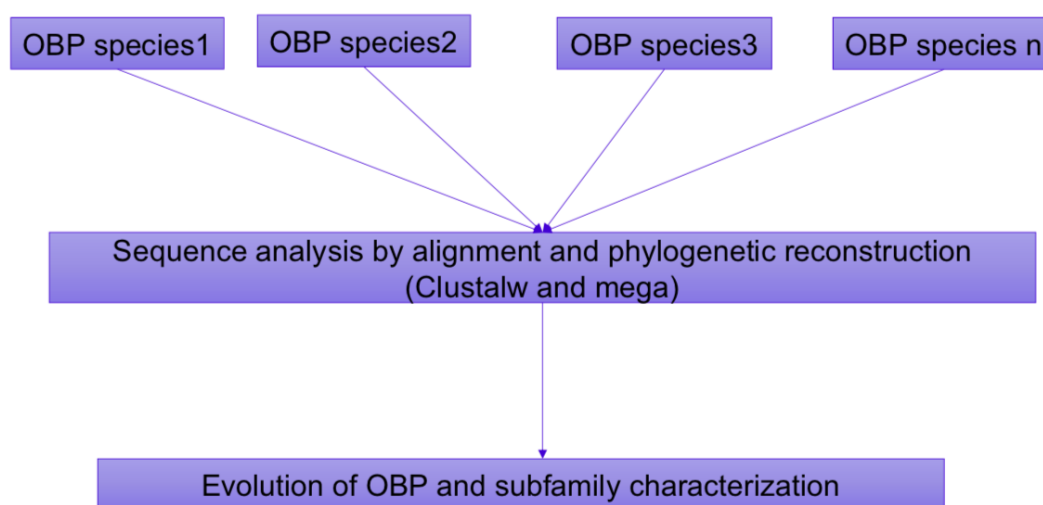


FIGURE 5.1 – Schéma des étapes qui peuvent compléter mon étude sur les OBPs et ainsi réaliser une analyse génétique complète.

5.2 Modélisation d'un modèle 3D par homologie.

Après avoir fait l'étude des séquences, il est envisagé de modéliser la structure 3D des protéines. Cette détermination de structure peut se faire par homologie. C'est à dire qu'un modèle 3D connu nous servira de base. Ici ce serait les modèles 3D des séquences de référence. Des logiciels tels que Modeller pourront donc modéliser les nouvelles OBPs à partir de la base en tenant compte du nombre de cystéines mais aussi des contraintes de repliement dues aux ponts disulfures.

5.3 Docking moléculaire.

Pour répondre à notre problématique, le docking permettrait de faire une analyse fonctionnelle de nos OBPs. Ainsi, il serait possible de déterminer le meilleur site de liaison entre deux molécules ainsi que leur affinité. Cette technique permettrait donc, pour notre problématique, de voir quelles sont les molécules qui se lient à nos nouvelles OBPs et ainsi trouver les molécules quiaturent les OBPS, ce qui permettrait de bloquer l'olfaction des moustiques et ainsi limiter la transmission de maladies.

6 Connaissances acquises

Durant ce stage de deux mois, de nombreuses compétences ont été acquises dans plusieurs domaines. J'ai tout d'abord fait de nombreux progrès au niveau de la programmation grâce aux Shell surtout dans la manipulation de fichiers. L'utilisation de l'environnement Ubuntu m'a permis de progresser surtout dans le langage Bash qui était une petite difficulté pour moi.

La polyvalence et la liberté d'utilisation de langages différents m'a permis de découvrir le langage Python qui pour moi était inconnu. J'ai donc pu apprendre les bases essentielles de Python et donc d'améliorer mes connaissances sur ce langage.

La création de ce rapport a été réalisé par Latex. De part sa taille et sa forme plus travaillée, j'ai pu accroître mes connaissances de ce logiciel car de nombreuses recherches ont été nécessaires pour arriver à ce résultat.

Enfin, ma méthode de travail s'est améliorée. En effet, j'ai essayé d'être aussi indépendante que possible en faisant de nombreuses recherches sur Internet et plus particulièrement sur des forums (openclasserom, developper, commentçamarche) pour résoudre mes problèmes de programmations.

Sans m'en rendre compte, mon niveau d'anglais s'est aussi amélioré car l'anglais est principalement utilisée dans le laboratoire d'accueil.

7 Abstract

Bioinformatics allows the analysis and interpretation of biological data and therefore represents a very varied field.

Nowadays, the olfactory system of mosquitoes is a subject of numerous studies. These studies focus more specifically on proteins that bind with odor molecules to allow for olfactory recognition. These proteins are called odorant-binding proteins. These proteins can be present in the proteome of an organism and their identification is possible thanks to the BLAST alignment tool which allows the search for protein by homology. Previously, this search was not automated and was done on a case-by-case basis for each sequence. A considerable amount of research time was therefore required, which caused a lot of time wasted.

This loss of time is limited to date by many tools such as the BLAST suite with makeblast and Psiblast (for protein search by homology) and computer language such as Python and BASH. Thanks to the BLAST tool, homology searches are possible by comparing one or more sequences to a database (database of reference protein sequences) by performing a blast cycle.

The work of this internship aims to deepen the knowledge of these proteins for different organisms by analyzing and improving programs allowing the search of the OBPs within a whole proteome. The collected data will be available on the website, dedicated to mosquito OBPs, developed by UFIP. This program allowed the identification of new OBP sequences among 16 newly sequenced mosquito genomes.

Keywords : mosquito, OBP, BLAST, PSI-BLAST, MAKEBLAST, homology search, orthology search.

8 Résumé

La bio-informatique permet l'analyse et l'interprétation de données biologiques et représente donc un domaine très varié.

De nos jours, le système olfactif des moustiques est un sujet propice à de nombreuses études. Ces études se portent plus précisément sur les protéines qui se lient avec les molécules odorantes pour permettre la reconnaissance olfactive. Ces protéines sont appelées des odorant-binding proteins. Ces protéines peuvent être présentes dans le protéome d'un organisme et leur identification est possible grâce à l'outil d'alignement BLAST qui permet la recherche de protéines par homologie. Auparavant, cette recherche n'était pas automatisée et s'effectuait au cas par cas pour chaque séquence. Un temps de recherche considérable était donc demandé.

Cette perte de temps est limitée de nos jours par de nombreux outils tel que la suite BLAST avec makeblast et Psiblast (pour la recherche de protéine par homologie) et langage informatique tel que Python, BASH. Grâce à l'outil BLAST, ces recherches par homologie sont possibles en confrontant une ou plusieurs séquences à une database (base de données de séquences de protéine de référence) en réalisant un cycle de blast.

Le travail de ce stage avait pour objectif d'approfondir les connaissances des protéines odorantes pour différents organismes en analysant et améliorant des programmes permettant la recherche des OBPs au sein d'un protéome entier. Les données recueillies devaient être consultables sur le site internet dédié aux OBPs de moustiques, développé par l'UFIP. Ce programme a permis l'identification de nouvelles séquences d'OBPs parmi 16 génomes de moustique nouvellement séquencés.

Mots-clés : moustique, OBP, BLAST, PSI-BLAST, MAKEBLAST, recherche par homologie, recherche d'orthologie.

9 Référence

- [1] Pelletier J, Leal WS (2009) génome Analysis and Expression Patterns of Odorant-Binding Proteins from the Southern House Mosquito *Culex pipiens quinquefasciatus*. PLoS ONE 4(7) : e6237. doi :10.1371/journal.pone.0006237
- [2] Malini Manoharan, Matthieu Ng Fuk Chong, Aurore Vaïtinadapoule, Etienne Frumence, Ramanathan Sowdhamini and Bernard Offmann, génome Biol. Evol (2013) Comparative Genomics of Odorant Binding Proteins in *Anopheles gambiae*, *Aedes aegypti*, and *Culex quinquefasciatus*. 163 ?180 doi :10.1093/gbe/evs131 Advance Access publication January 3, 2013
- [3] <http://www.bo-protscience.fr/mobpdb/> , UFIP, dernière mise à jour : mai 2018
- [4] http://www.uniprot.org/help/api_idmapping, dernière modification : septembre 2017
- [5] <https://www.uniprot.org/uploadlists/>, dernière modification : mars 2018
- [6] Highly evolvable malaria vectors : The génomes of 16 *Anopheles* mosquitoes par Daniel E. Neafsey, Robert M. Waterhouse et al