



MASTER OF SCIENCE
IN ENGINEERING

UNIVERSITY OF APPLIED SCIENCES WESTERN SWITZERLAND
MSE - SOFTWARE ENGINEERING

Projet d'approfondissement Inphinity

Authors:
Déruaz Vincent

Supervisors:
Prof. Carlos Peña

Hes·SO

Haute Ecole Spécialisée
de Suisse occidentale

Fachhochschule Westschweiz

University of Applied Sciences and Arts
Western Switzerland

Lausanne, June 5, 2016

Contents

1	Introduction	3
1.1	foreword	3
1.2	phages Vs bacteria	3
2	State of the art	4
2.1	Biology	4
2.2	Bio-informatic	4
3	Methods	5
3.1	Docker	5
3.1.1	Functionment	5
3.1.2	Docker-machine installation	5
3.1.3	Docker commands	6
3.1.4	Inphinity, build & run	6
3.2	Phamerator	7
3.2.1	Installation	7
3.2.2	How it's works	8
3.3	PhamDB	9
3.3.1	Installation & Run	9
3.3.2	Utilisation	10
3.4	Database & Dataset	11
3.5	Phamily	12
3.5.1	Introduction	12
3.5.2	Phages selection	13
3.5.3	Displaying results	17
3.5.4	Data completion	17
4	Results & Analyse	18
4.1	First result	18
4.1.1	Tree	18
4.1.2	Hosts	18
4.2	Database	18
4.2.1	SEA	18
4.2.2	Phages list integration	18
5	Conclusion	19
5.1	Problems encountered	19
5.1.1	Phamerator Installation	19
5.1.2	SEA database	19
5.1.3	PhamDB Limitation jobs	19
5.1.4	PhageDB.org	19
5.2	Perspectives	19
5.2.1	Database population	19
5.2.2	Resultats validation	19
5.2.3	Results by host	19

6	List of figures	19
7	References	20
8	Annexes	22
.1	Fasta file	22

1 Introduction

1.1 foreword

This project falls within the context of a thesis proposed by Prof. Carlos Peña, YokAi Que, MDPhD and Grégory Resch, PhD entitled *In silico prediction of phagebacteria infection networks as a tool to implement personalized phage therapy* [3].

The official statment of the project is:

By using automated learning methods, explore alternate metodologies for bacteria and phages interaction modelisation on genomic informations or proteins.

1.2 phages Vs bacteria

In our modern world a challenging issues has apear concerning conventional antibiotics. In deed, some batceria have developpe resistance to antibiotics. To overcome this people are looking at phage therapy.

Phage therapy is the utilisation of phages, bacteriophage viruses, to threat infectious diseases of bacterial origin. This therapy is known to have only very few and only benign side effets. This last point make phagotherapy, not only useful to avoid antibiotic in case of resistance, but also to avoid their "toxicity".

Briefly, phage therapy was the only threatment solution in the before the 1930's. The apearence of the penicillin in the early 1940's and other modern drugs, releagate phage therapy to the past. But, in the slavic countries, phage therapy continued to be used as a current treatment.

Luckly for us, we don't have to start from nothing in phage therapy. However, we have the necessity to find a way, a methode to validate the phage selection. [4]

2 State of the art

2.1 Biology

TODO: genome

2.2 Bio-informatic

TODO: sequence alignment

TODO: explain phamily

TODO: explain genebank

3 Methods

In this section we will discuss about what we've used during this project. The Docker technology is used to build the different work environment. Phamator and PhamDB are used to compute genomes into family. Everything is stored into some SQL databases.

3.1 Docker

Docker allows to package applications with a functional system and every dependencies needed to run it, into a standardized container. [2]

3.1.1 Functionment

In a specific file called 'Dockerfile' you describe a system. You can build and run this system everywhere docker engine is installed. It will create a container, containing your application.

Containers are an isolated system from host or other containers. You can use every Linux distribution to run your container.

Docker builds images using layers, it allows docker to share those layers between containers, reducing disk usage at best.

3.1.2 Docker-machine installation

You have to install docker on your system, it can be used on Linux, Windows or MacOS.
<https://docs.docker.com/linux/>

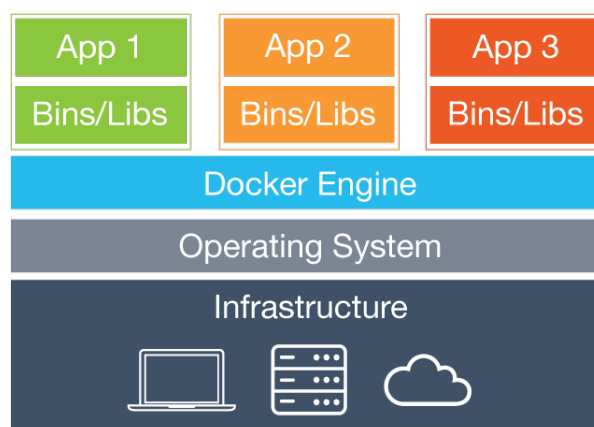


Figure 3.1: Docker harchitecture

Windows

If you want to use docker on Windows you need to do as following to ensure to have a docker-machine with more than 20Go.

```
$ docker-machine rm default
$ docker-machine create -d virtualbox --virtualbox-memory=4096 --virtualbox-cpu-count=2
→ --virtualbox-disk-size=50000 default
```

It will crate a docker-machine with 50Go of disk space.

3.1.3 Docker commands

Here are the basic commands you will need to manage docker. Attention, you will need to be in the same directory as the Dockerfile.

This command allow to build an image describe in the Dockerfile.

```
$ docker build -t "<image Name>" .
```

This command is used to run a container using a pre-build image, with a binding port.

```
$ docker run -it -d -p <host port>:<container port> <image name>
```

If you need to acces the container bash console, juste use this commande

```
$ docker exec -i -t <container ID> /bin/bash
```

You can list all the running containers and use a <container id> to stop it.

```
$ docker ps
$ docker stop <container ID>
```

This last command give you the ip of your docker-machine.

```
$ docker-machine ip
```

3.1.4 Inphinity, build & run

For the main code of this project we use python through Jupyter. To do so you can find a docker image that run Jupyter, python3 and some machin learning libraries. go to "dockers/jupyter_align_mysql" directory.

To build and run the environment type the following commands:

```
$ docker build -t "pa/inphinity" .
$ docker run -v <path to project dir>/jupyter_align_mysql/src:/home/pa/work/ -i -t -d -p 8888:8888
→ pa/inphinity
```

Replace <path to project dir> by the entire path to the directory. If you want to, you can change the host port. Just change "-p 8888:8888" to "-p <any port>:8888".

You can now access the jupyter interface and the project files using any browser you want using `http://<docker-machineip>:8888`

At this point you should see the interface figure 3.2

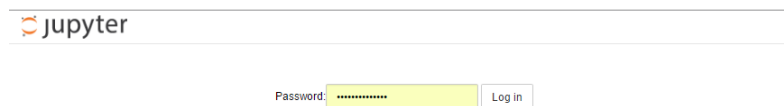


Figure 3.2: Jupyter login page

Rq: The password is "Inphinity-more"

When you're logged in you can access the "inphinity" directory. It contains most of this project results. We will discuss them later in this document.

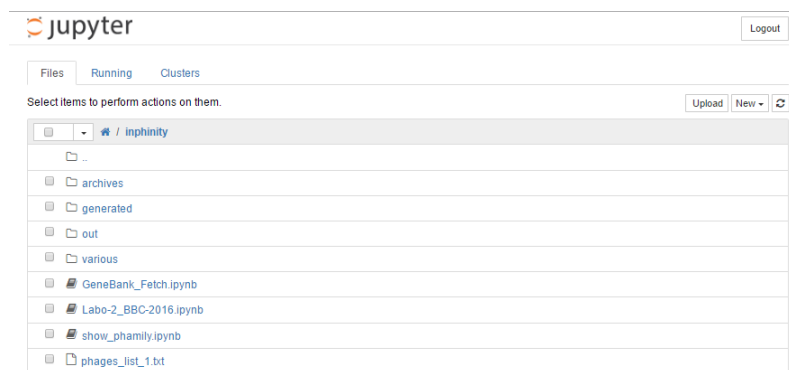


Figure 3.3: Inphinity jupyter directory

3.2 Phamerator

Phameroatr is a *bioinformatic tool for comparative bacteriophage genomics* [5]. Phamerator will allow us to compute and store phamily, in a database.

3.2.1 Installation

To use phamerator you have to install a linux environment. Thus copy the file "phamerator.sh" from the directory and execute it in order to install and run Phamerator.

Normally you Phamerator should start at the end.

The screenshot shows the phamerator interface. At the top, there are navigation links: Map, Pham Circle, and Google Maps. Below this is a 'Sources' section with a search bar. The main content area is titled 'The Actinobacteriophage Database at PhagesDB.org'. It features a navigation bar with links: Home, Phages, Hosts, Data, BLAST, Publications, Resources, Software, and Social. A search bar is present with the text 'Search PhagesDB.org'. Below the search bar, there is a 'Recently Added Phages' section listing: BossLady, Birchlyn, Attoomi, and Rowa. The main content area displays a table of phages with columns: Name, Length (bp), GC %, Number of Genes, and Cluster. The table lists several phages, including Corndog, Wildcat, Giles, Marvin-DRAFT, SendS13-DRAFT, Bxb1, and I12. Below the table, there is a section titled 'Mycobacterium phage Marvin' with two images. At the bottom, a status bar indicates 'Downloading BLAST (0% Complete)'.

Name	Length (bp)	GC %	Number of Genes	Cluster
Corndog	69777	65.384000	122	
Wildcat	78441	56.851600	148	
Giles	54512	67.267800	79	
Marvin-DRAFT	65089	63.419300	111	
SendS13-DRAFT	71547	55.998200	101	
Bxb1	50550	63.649900	86	A1
I12	51277	63.699100	81	A1

Figure 3.4: phamerator interface

3.2.2 How it's works

TODO: explain

The screenshot shows a detailed view of a phage family in the phamerator interface. The top navigation bar includes Map, Pham Circle, and Google Maps. Below this is a 'Sources' section with a search bar. The main content area is titled 'Mycobacterium phage Marvin'. It features a navigation bar with links: Home, Phages, Hosts, Data, BLAST, Publications, Resources, Software, and Social. A search bar is present with the text 'Search PhagesDB.org'. Below the search bar, there is a 'Recently Added Phages' section listing: BossLady, Birchlyn, Attoomi, and Rowa. The main content area displays a table of phage families with columns: Phamily, Number of Members, and Clusters. The table lists several phage families, including 7, 8, 9, 10, 11, 17, and 18. Below the table, there is a section titled 'Mycobacterium phage Marvin' with two images. At the bottom, a status bar indicates 'Downloading BLAST (0% Complete)'.

Phamily	Number of Members	Clusters
7	26	A1, A2, A3, A4, A5
8	26	A1, A2, A3, A4, A5
9	26	A1, A2, A3, A4, A5
10	26	A1, A2, A3, A4, A5
11	26	A1, A2, A3, A4, A5
17	16	A1, A3, A5
18	11	A1

Gene	Phage	Global % Identity	BLAST E-Value
16	Bxb1		
20	Bxz2		
gp17	U2		
gp17	Bethlehem		
20	D29		
gene20	L5		
22	Che12		
19	DD5		
19	Jasper		
KBG_19	KBG		
18	Lockley		
22	Pukovnik		
18	Solon		
19	Peaches		
20	SkiPole		
23	RedRock		

Figure 3.5: phamerator interface

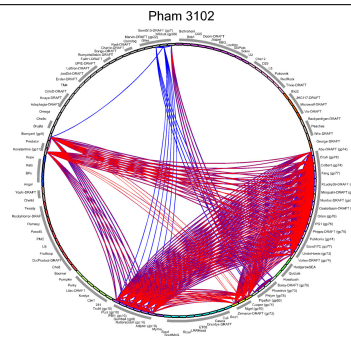


Figure 3.6: phamerator interface

3.3 PhamDB

To facilitate the construction of databases containing our phamily we will use PhamDb. In deed, with phamDb we can populate a database with new phage. At every addition of phage, it will recompute phamily accordingly to the new phage added.

We no longer need to access to Phamerator by GUI. In the future it will let us build a fully automated pipline of actions.

3.3.1 Installation & Run

To build and run the environment type the following commands:

```
$ docker build -t "pa/phamdb" .
$ docker run -v <path to project dir>/jupyter_align_mysql/src:/home/pa/work/ -i -t -d -p 81:80
  → pa/phamdb
```

Replace <path to project dir> by the entire path to the directory. If you want to, you can change the host port. Just change "-p 81:80" to "-p <any port>:80".

You can now acces the jupyter interface and the project files using any browser you want using <http://<docker-machineip>:80>

At this point you should see the interface figure 3.4, but with no database for the moment.

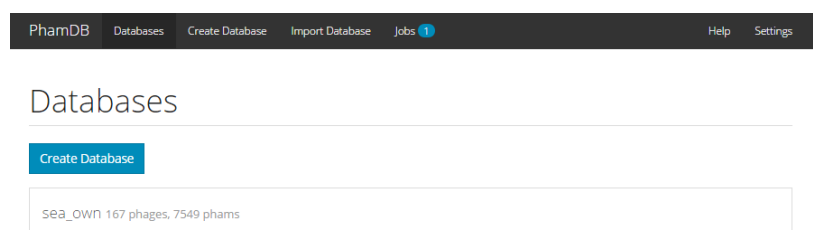


Figure 3.7: Phamdb interface

3.3.2 Utilisation

As you see from figure 3.4 you can access the list of all existing database and consult them.

PhamDB Databases Create Database Import Database Jobs 1 Help Settings

Database - sea_own

Edit Database Delete database

167 phages, 7549 phams

Description: sea_own first tests
Orphans: 61.5% (4646 / 7549)
Conserved domain search: Yes
Download: [sea_own.sql](#)

Connecting with Phamerator

To view this database in Phamerator, click **Edit > Settings**, then add the following settings:

Server	http://home.kamylh.ch:81/db/
Database	sea_own

Phages	
lambda (id: 10710, 73 genes)	Download
NF (id: 10753, 27 genes)	Download
B103 (id: 10778, 17 genes)	Download
B55 (id: 1126949, 272 genes)	Download
vB_AsaM-56 (id: 1127514, 83 genes)	Download
TA17A (id: 1131248, 94 genes)	Download
phage (id: 1131248, 94 genes)	Download

Figure 3.8: PhamDb database visualisation

You can create a new database from three different ways:

1. By importing phages from existing database on phamdb.
2. By uploading Genbank files.
3. By importing as an Sql file.

PhamDB

Databases

Create Database

Import Database

Jobs 1

Help

Settings

Create Database

Database Name

Enter a name for the database

Description

Enter a description

☐ Search Conserved Domain Database
Search for each gene in NCBI's conserved domain database. This will significantly increase runtime.

Import Phages from Database

Select a phage to import it from a database.

Databases

sea_own (id: 157 phages)

Phages

lambda (id: 10710, 73 genes)

NF (id: 10753, 27 genes)

B103 (id: 10778, 17 genes)

B55 (id: 1126949, 272 genes)

vB_AsaM-56 (id: 1127514, 63 genes)

TA17A (id: 1131248, 94 genes)

☒ **B103** (id: 10778, database: sea_own, 17 genes)

☒ **NF** (id: 10753, database: sea_own, 27 genes)

Upload Genbank Files

Sélectionner fichiers

Aucun fichier choisi

Select genbank files or drag and drop.

Import Database from SQL File

Database Name

Enter a name for the database

Description

Enter a description

Upload SQL File

Choisissez un fichier

Aucun fichier choisi

No file selected.

Submit

Figure 3.9: PhamDb database creation

3.4 Database & Dataset

We use the default database from phamerator for this phase of the project in order to gain some time. You can see the database structure on the figure 3.5 .

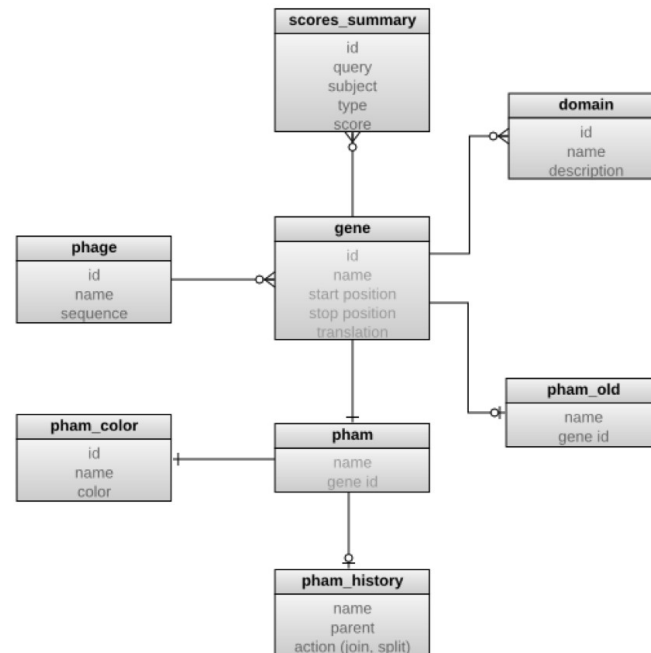


Figure 3.10: Docker harchitecture

From the dataset at my disposal, I've used the list of phage "phages_list_1.txt". You can find it in annexe of this document.

3.5 Phamily

Now we will discuss about phamily and the main script realise during this phase of the project. You can see and run the folowing code from the jupyter interface, Cf. chapter 3.1.4.

3.5.1 Introduction

Have to: You are now in your browser and have open the source file "inphinity/show_phamily.ipynb". The class "sea_inphinity()" have every methods that we use in this part. cf. file if you want to see it all.

First we have to create an Inphinity() object capable to access our data. We will use a modified database 'sea_own', we will discuss his creation later in the document (cf section 4.2.2).

```

1 inphinity = Inphinity('sea_own')
2 #Result: Number of phages loaded: 167

```

Figure 3.11: Inphinity object

You can display the list of every existing phamily in the database curently selected.

```
1 list_name = inphinity.get_list_name_pham(-1)
2 print(list_name)
3 #Result: [... ,2798,2799,2800,2801, ...]
```

Figure 3.12: Phame names listing

Note: A verbosity parameter allows to turn off console message from python execution workflow.

```
1 inphinity.verbosity = False
```

Figure 3.13: Verbosity attribut

3.5.2 Phages selection

Now we want to be able to get a phylogenetic tree for a phage. To do so we choose a phame, here pham '2799' and we call our method *inphinity.build_tree('2799')*.

```
1 def build_tree(self, pham):
2     genes = self.get_genes_from_a_pham(pham)
3     self.create_fasta(genes)
4     self.align_muscle()
5     self.compute_tree()
6     self.prepare_tree_fig()
```

Figure 3.14: Creation of phylogenetic tree

As you can see, this methods those a couple of different things. First we're retrieving all the genes composing our pham.

Now that we have selected only few genes, we have to store them into a FASTA file to be use with MUSCLE for align them.

```

1 def create_fasta(self, genes):
2     print('Creation of the FASTA file')
3     fasta = open("%sfasta.fa" % (self.out_dir), "w")
4     self.print_("Number of Genes: %d" % (len(genes)))
5     for gene in genes:
6         GeneID = gene['GeneID']
7         name = gene['Name']
8         description = ">%s - %s" % (GeneID, name)
9
10        translation = gene['translation']
11
12        self.print_(description)
13        self.print_(translation)
14
15        fasta.write(description)
16        fasta.write('\n')
17        fasta.write(translation)
18        fasta.write('\n')
19
20    fasta.close()

```

Figure 3.15: FASTA creation function

The FASTA file should look like figure 3.16. A gene takes two lines. First the gene identification number and his name. Then, the second line, the gene translation. You can find in annexe ()

```

1 >148603_11 - 11
2 MAETESIDPEKLRDQLDAFENKQNELKSSKAYYDAERRPDAIGLAVPLDMRKYLAVHGYPRTYVDAIAERQELGFRIPSAANGEEPESGGENDPASELHDMQANNLDIEAT
3 LGHTDALIYGTAYITISMPDPEVDFDVPDLIRVEPTALYAEVDPRTRKVLVAIRAIYGADGNEIVSATLYLPDPTMTWLRAGEWNEAPTSTPHGLEMPVPVIPISNRTRL
4 SDLYGTSESPELRSVTDAAQILMMQGTANLMAIPQRLIFGAKPEELGINAETGQRMFDAYMARILAFEGGEGAHAEQFSAELRNFDALDALDRKAASYSGLPPQYLS
5 SSONPASAEAIKAAESRLVKVVERKNKIFGGAWEQAMRLAYKMKVGGDIPEYRMEIVMDPSTPTAAKADAAKLFANGAGLIPRERGIVDMGYTIVEREQMRQNLQDDQ
6 KQGLGLIGSLYGASTPEGKPGEPAPVGEPPAPEPDA
7 >205870_14 - 14
8 MTSPLQKQENVDPKAREEHLNLFERTQDLGNTAYYESERRPDAVGTVPPQMQKLLAHVGYPRLYIDAIAERQELGFRILGGADKADEQLMDWMQANDLDIESTLGHDT
9 LVHGRSYITISKDPDNIDPQDPEVPIIRVEPTNLVAQIDPRTRQVMRAIRAEDEEGNEIVGATLYLPNNTVIMNREDGQVQVANVAHNLNEMVPVIPINRTRLSDLYGT
10 TEITPELRSVTDAAARTLMLQATAELMGVPQRLLFQVKGEEELGVDPEGTQTLFDAYLARILAFEDHESKAQFSAELRNFDALDALDRKAAYTGLPPYLSFSSENPA
11 AEAIRSESRLVKVVERKNKIFGGAWEQAMRVAYKVMGGDIPEYRMEIVMDPSTPTAAKADAAKLYNNGQGVIPKERARIDMGYSITEREEMRKWDEEQAGLGLM
12 GTMFGTDPSSGGNPNPETPEPQNPAAAA
13 >205879_166 - 166
14 MRDKVRDELAGFVIRAQYSGPGQTTDEIVDAILEKFDVTEKPGPAVPFGTIRVTASSOGRVWVIFISNGDGTWVTFHDKHGHEGLHMSAPWDGLDPGEKEVFRP
15 >260120_gp11 - gp11
16 MTTYHEHVERLQGLLARDLPNLLEAEAYRNGTRRLKTIIGAPPELAYLDVQPGWATYLRSLDRLDIEGFRISEDSEGLEELNMMQANDLDEESVLGHDDSLTFGRAYIT
17 VSHPDVESGDPAGIPLIRVESPLYMYAELDPNRRVTRAVRLYTRDDVAVPDRATLYLPDETVPPLRRNGGLNDQNVVDGDKHGLGVVPVPLTNDPRLGNRYGRSEISP
18 ELRKVTDAASRTLMNLQASQILGTPLRVISGVYTTDELNDGENTTLDIYGRILTLASEAAKISEFKAELRNFAEEMEVEFRKEAASITGLPPQYLSSENPAEAIAT
19 DSRIVKMAERKGRIFGGAWEQAMRIAMQIMGREVTEYTRLETVMRDPSTPTAAKADAVSKLYANGQGPVKEQARIDLGYTATQREQMRDMKQETEDMIDTLYSTTKAQA
20 DATPKPTVTEKTEQTSFGFNRTKTR
21 >260121_gp11 - gp11
22 MTTYHEHVERLQGLLARDLPNLLEAEAYRNGTRRLKTIIGAPPELAYLDVQPGWATYLRSLDRLDIEGFRISEDSEGLEELNMMQANDLDEESVLGHDDSLTFGRSYIT
23 VSHPDVESGDPAGIPLIRVESPLYMYAELDPNRRVTRAVRLYTRDDVAVPDRATLYLPDETVPPLRRNGGLNDQNVVDGDKHGLGVVPVPLTNDPRLGNRYGRSEISP
24 FIKVNTDASRTIMNLCASQITGDI RUTSGVTTDFI TMNGFMTTI DTVYGRTI TI ASFADKTSFFKASDFI RNFAFFMEVEFRKFAASITGI PPNVI SSCSFMDSASASATTTAT

```

Figure 3.16: PhamDb database creation

Now we are ready to use MUSCLE to align genes together. **TODO:** explain MUSCLE. In the future it will be interesting to take some time to customize our call to MUSCLE [1]

```
1 def align_muscle(self):
2     print('Alignment with MUSCLE')
3     muscle_loc = r'/home/pa/work/muscle3.8.31_i86linux64' # modifier si nécessaire
4
5     muscle_cline = MuscleCommandline(cmd=muscle_loc,input='%sfasta.fa' %
6                                     (self.out_dir),out='%sout.aln' % (self.out_dir),clwstrict=True)
7     stdout, stderr = muscle_cline()
8
9     muscle_align = AlignIO.read('%sout.aln' % (self.out_dir),'clustal')
10    self.print_(muscle_align)
```

Figure 3.17: FASTA creation function

MUSCLE takes our generated FASTA file to produce a .aln file who will contains the alignments.

Now we have our alignments ready to compute a phylogenetic tree. To realise the tree we use *FastTree* software. It will produce "approximately-maximum-likelihood phylogenetic trees" using our .aln file previously generated.

```
1 def compute_tree(self):
2     print('Compute tree')
3     AlignIO.convert('%sout.aln' % (self.out_dir),'clustal','%sintermediate.phy' \
4                   % (self.out_dir), 'phylip-relaxed')
5
6     cmd_fasttree = r'fasttree'
7     fasttree_cmdline = FastTreeCommandline(cmd=cmd_fasttree,fastest=True, \
8                                             input='%sintermediate.phy' % (self.out_dir),out='%stree.tre' % (self.out_dir))
9     out_log, err_log = fasttree_cmdline()
10
11    self.print_('Out Log:')
12    self.print_(out_log)
13
14    self.print_('Error Log')
15    self.print_(err_log)
16
17    self.tree = Phylo.read('%stree.tre' % (self.out_dir), 'newick')
```

Figure 3.18: FASTA creation function

Note: As for MUSCLE utilisation, maybe some modification to the call of FastTree can be used to improve the solution.

The following figure (3.19) show that we now have a file with our tree. Every genes have a score that will decide for its place in the tree.


```

1 (540068_15:0.07526,Turbido-DRAFT_gp14:0.05976,(((28369_14:0.03618,373405_14:0.04735)0.942:0.02920,(((George-
DRAFT_gp9:0.21862,148603_11:0.21628)0.997:0.13462,(((205870_14:0.0,JHC117-DRAFT_gp14:0.0,Microwolf-
DRAFT_gp14:0.0):0.00186,Vix-DRAFT_gp11:0.00623)0.991:0.09822,((663557_12:0.02764,(Backyardigan-DRAFT_gp12:0.0,Wile-
DRAFT_gp11:0.0):0.04929)1.000:0.23759,((((540064_13:0.0,540067_12:0.0):0.00604,540065_13:0.00000)0.912:0.00201,
((260120_gp11:0.0,Doom-DRAFT_gp14:0.0,SargentShorty9-DRAFT_gp12:0.0):0.00000,
(260121_gp11:0.00605,540066_KBG_13:0.00403)0.000:0.00000)0.581:0.00000)0.547:0.00201,
(555603_12:0.00000,701456_14:0.00401)0.440:0.00000)0.895:0.15062,
(701456_87:0.26122,205879_166:1.56348)0.830:0.17337)1.000:0.70950)0.955:0.11253)0.933:0.07263)1.000:0.24700,31757_ge
ne14:0.04763)0.493:0.01604)0.632:0.00340,(711470_16:0.04242,Trixie-DRAFT_gp17:0.04730)0.953:0.01654)0.867:0.02076);
2

```

Figure 3.19: PhamDb database creation

Now we can compute the visualisation of the tree using two function, *prepare_tree_fig()* and *draw_tree()*.

The function *compute_tree()* set an attribut, *self.tree*, to *Inphinity()* class. This is our tree, so we can used it after computation. For the selected Pham we have the tree from figure 3.19. We will discuss the result in the chapter 4.

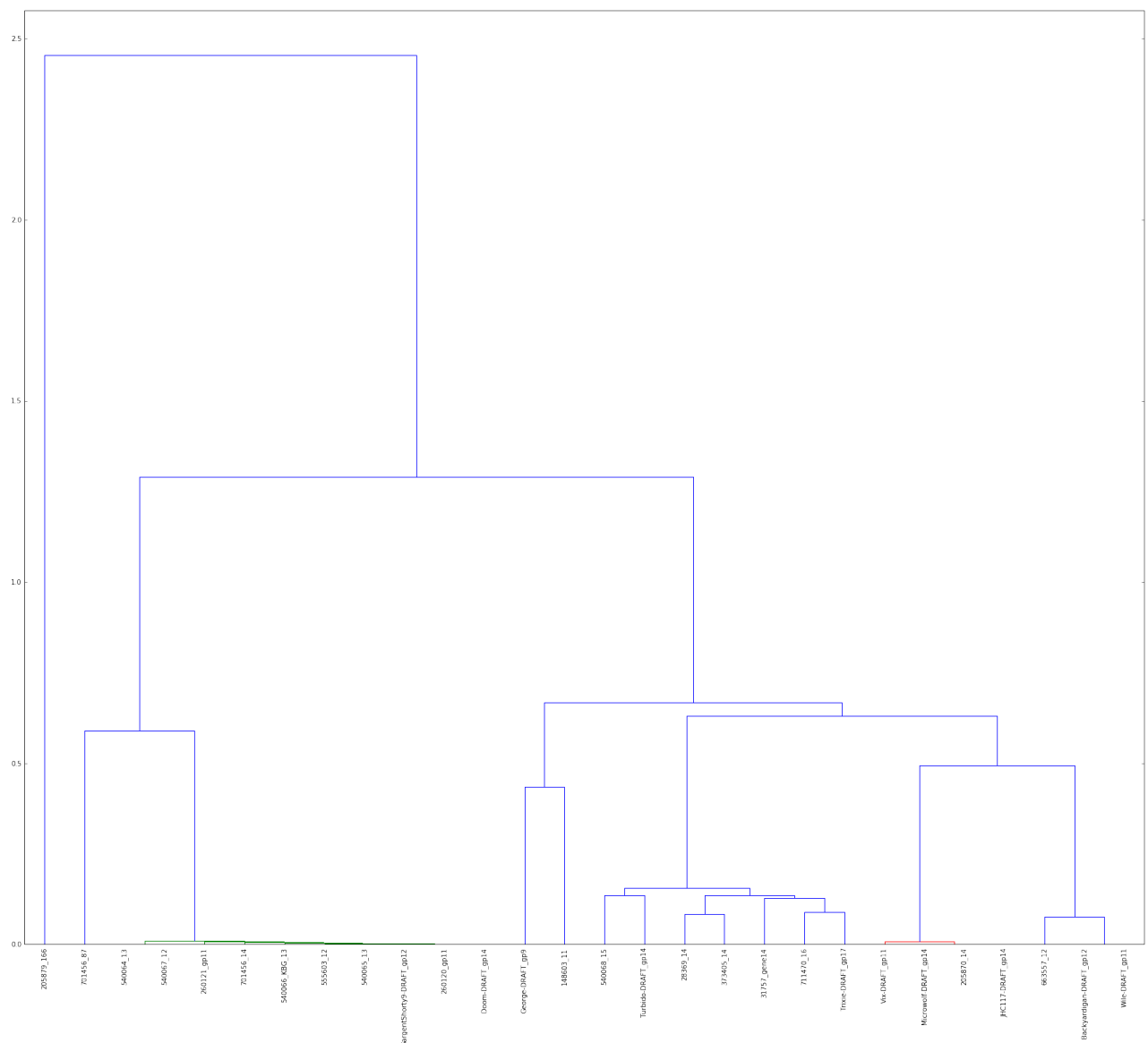


Figure 3.20: PhamDb database creation

3.5.3 Displaying results

```
1 inphinity = Inphinity('sea')
2 inphinity.verbose = False
3 inphinity.build_tree('2799')
4 inphinity.print_information_on_phages(inphinity.leaves)
```

Figure 3.21: FASTA creation function

3.5.4 Data completion

4 Results & Analyse

4.1 First result

4.1.1 Tree

4.1.2 Hosts

4.2 Database

4.2.1 SEA

4.2.2 Phages list integration

5 Conclusion

5.1 Problems encountered

5.1.1 Phamerator Installation

5.1.2 SEA database

5.1.3 PhamDB Limitation jobs

5.1.4 PhageDB.org

5.2 Perspectives

5.2.1 Database population

5.2.2 Resultats validation

5.2.3 Results by host

List of Figures

3.1	Docker harchitecture	5
3.2	Jupyter login page	7
3.3	Inphinity jupyter directory	7
3.4	phamerator interface	8
3.5	phamerator interface	8
3.6	phamerator interface	9
3.7	Phamdb interface	9
3.8	PhamDb database visualisation	10
3.9	PhamDb database creation	11
3.10	Docker harchitecture	12
3.11	Inphinity object	12
3.12	Phame names listing	13
3.13	Verbosity attribut	13
3.14	Creation of philogenetic tree	13
3.15	FASTA creation function	14
3.16	PhamDb database creation	14
3.17	FASTA creation function	15
3.18	FASTA creation function	15
3.19	PhamDb database creation	16
3.20	PhamDb database creation	16
3.21	FASTA creation function	17

Bibliography

- [1] by Robert C. Edgar. Muscle - multiple sequence comparison by log-expectation. <http://www.drive5.com/muscle/muscle.html>.
- [2] Docker. Docker official website. <https://www.docker.com/>.
- [3] YokAi Que MDPHD, Prof. Carlos Peña PhD, and Grégory Resch PhD. In silico prediction of phagebacteria infection networks as a tool to implement personalized phage therapy.
- [4] Ncbi. A historical overview of bacteriophage therapy as an alternative to antibiotics for the treatment of bacterial pathogens. <http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3916379/>.
- [5] Phamerator. Phamerator: a bioinformatic tool for comparative bacteriophage genomics. <http://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-12-395>.

8 Annexes

.1 Fasta file