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

Haute Ecole Spécialisée
de Suisse occidentale
Fachhochschule Westschweiz
University of Applied Sciences and Arts
Western Switzerland

Lausanne, June 8, 2016

Contents

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* [5].

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 (Section 2.2.2), bacteriophage viruses, to threat infectious diseases of bacterial origin. Phage, also called bacteriophage, are a specific type of virus infecting only bacteria. 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. [7]

2 Theories

This chapter is made to ensure that every reader understand basics needed. It contains some simplifications and it's not, in any case, a biology guide.

2.1 Biology

Before entering in the project content, let say some word about genomic. Almost all living organisms use **Deoxyribonucleic acid, DNA,** [8]. DNA contains the necessary imformation for the developpment and activities of living creatures.

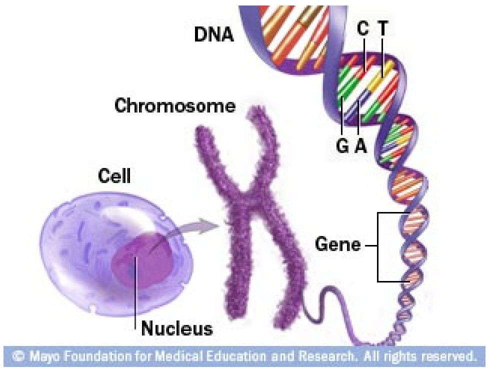
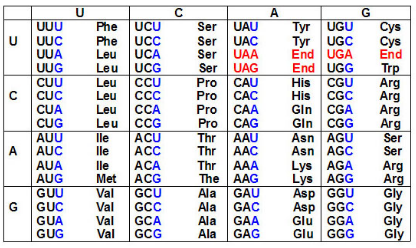

DNA
C T
G A
Chromosome
Cell
Gene
Nucleus
© Mayo Foundation for Medical Education and Research. All rights reserved.

Figure 2.1: A figure



	U		C		A		G	
U	UUU	Phe	UCU	Ser	UAU	Tyr	UGU	Cys
	UUC	Phe	UCC	Ser	UAC	Tyr	UGC	Cys
	UUA	Leu	UCA	Ser	UAA	End	UGA	End
	UUG	Leu	UCG	Ser	UAG	End	UGG	Trp
C	CUU	Leu	CCU	Pro	CAU	His	CGU	Arg
	CUC	Leu	CCC	Pro	CAC	His	CGC	Arg
	CUA	Leu	CCA	Pro	CAA	Gln	CGA	Arg
	CUG	Leu	CCG	Pro	CAG	Gln	CGG	Arg
A	AUU	Ile	ACU	Thr	AAU	Asn	AGU	Ser
	AUC	Ile	ACC	Thr	AAC	Asn	AGC	Ser
	AUA	Ile	ACA	Thr	AAA	Lys	AGA	Arg
	AUG	Met	ACG	The	AAG	Lys	AGG	Arg
G	GUU	Val	GCU	Ala	GAU	Asp	GGU	Gly
	GUC	Val	GCC	Ala	GAC	Asp	GGC	Gly
	GUA	Val	GCA	Ala	GAA	Glu	GGA	Gly
	GUG	Val	GCG	Ala	GAG	Glu	GGG	Gly

Figure 2.2: Another figure

Dna is composed of 4 units, called nucleotide bases, adenine (A), thymine (T), guanine (G), and cytosine (C). Those bases are stored in chromosome, in human cells (Fig 2.1). There are responsable for coding information used to create proteins (Fig. 2.2). Organisms structural organisation and every biological activity are made up from proteins.

2.2 Bio-informatic

2.2.1 DNA sequencing

Today we are capable of sequencing genetics code of any organism. When DNA is sequence, only one half is recorded, because every bases work in pairs (C-G | T-A).

Note: For more information abour sequencing see article in reference [8].

2.2.2 Phamily

A pham is a regroupment of protein-coding genes where related sequence, determined by pairwise analysis, are used for comparaison. Sequence alignment is used to determin conserved domaine in DNA of phage. A phamily contains all related genes.

2.2.3 Genbank

Most of the genetics information used in this project can be found on GenBank [6]. GenBank is a public database regrouping avalaible DNA sequences and information.

3 Methods

In this section we will disscuss about what we've used during this project. The Docker technology is used to build the differents work enviornment. Phamerator and PhamDB are used to compute genomes into phamily. Everythings is stored into some Sql databases.

3.1 Docker

Docker allows to package applications with a fonctionnal 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.

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

Docker build images using layers, it allows docker to share those layer 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/`

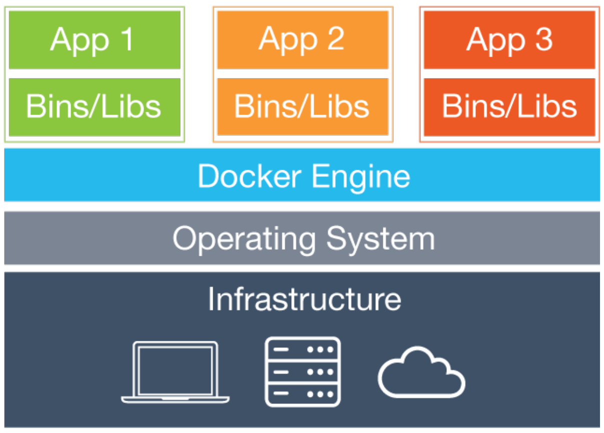


Figure 3.1: Docker harchitecture

Windows

If you want to use docker on Windows you need to do as folowing 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 acces 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 figue 3.2

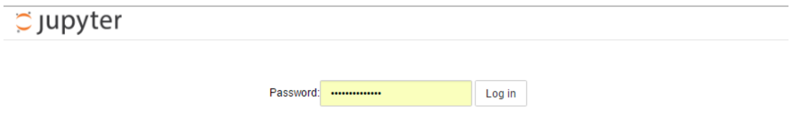
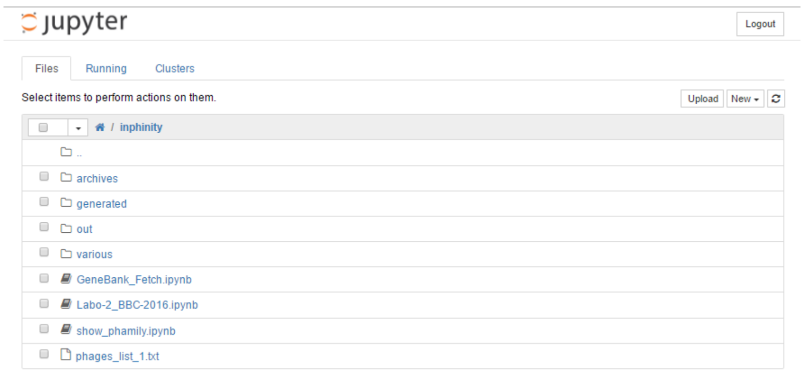


Figure 3.2: Jupyter login page

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 disscuss them later in this document.



Select items to perform actions on them.

..
archives
generated
out
various
GeneBank_Fetch.ipynb
Labo-2_BBC-2016.ipynb
show_phamily.ipynb
phages_list_1.txt

Figure 3.3: Inphinity jupyter directory

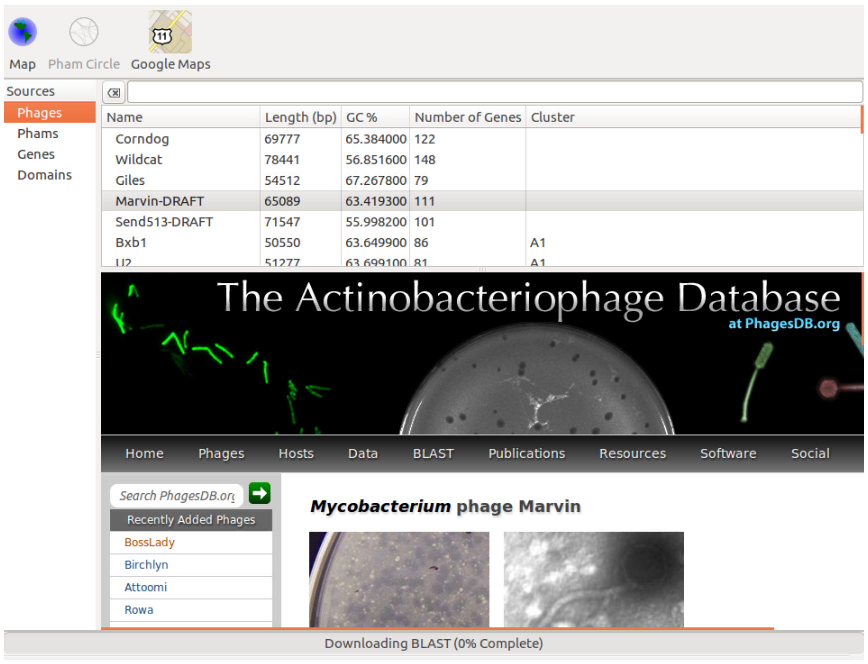
3.2 Phamerator

Phameraotr is *a bioinformatic tool for comparative bacteriophage genomics* [9]. 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 enviornment. 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.



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	
Send513-DRAFT	71547	55.998200	101	
Bxb1	50550	63.649900	86	A1
U2	51277	63.699100	81	A1

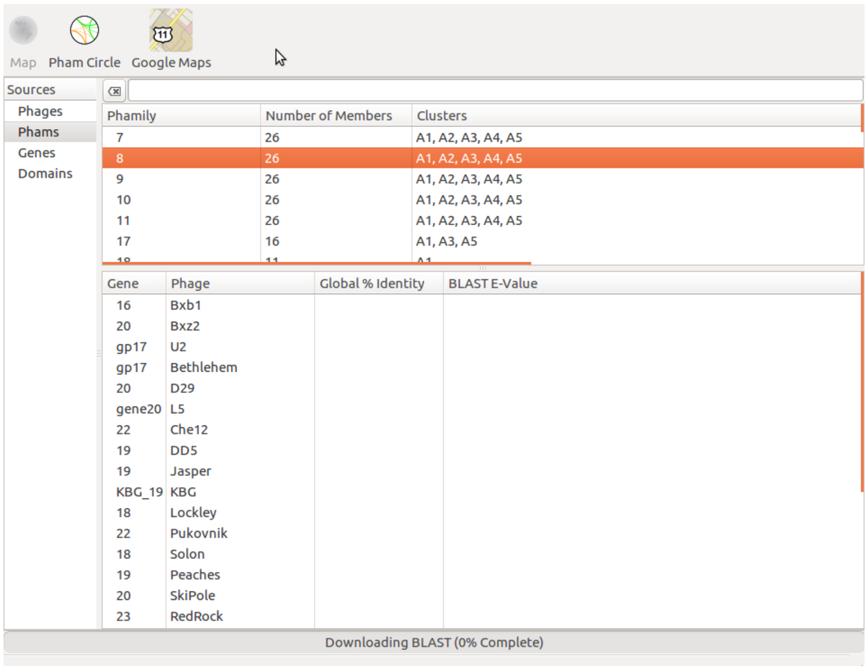
***Mycobacterium* phage Marvin**

Downloading BLAST (0% Complete)

Figure 3.4: phamerator interface

3.2.2 How it's works

TODO: explain



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		

Downloading BLAST (0% Complete)

Figure 3.5: phamerator interface

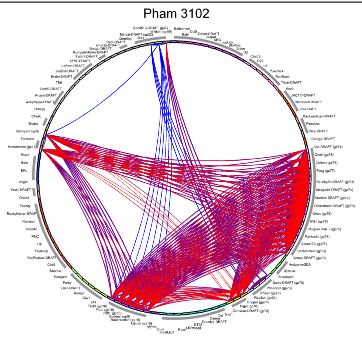


Figure 3.6: phamerator interface

3.3 PhamDB

To facilited 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

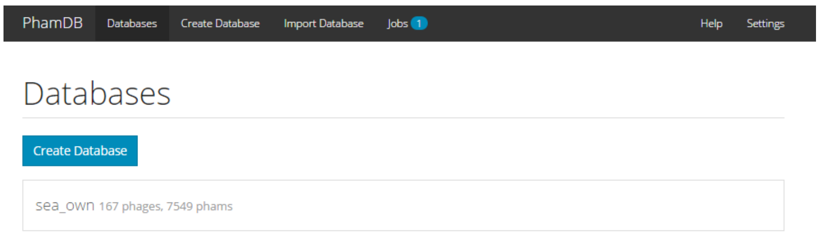
To build and run the environment type the following commands:

```
$ 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 figue 3.4, but with no database for the moment.

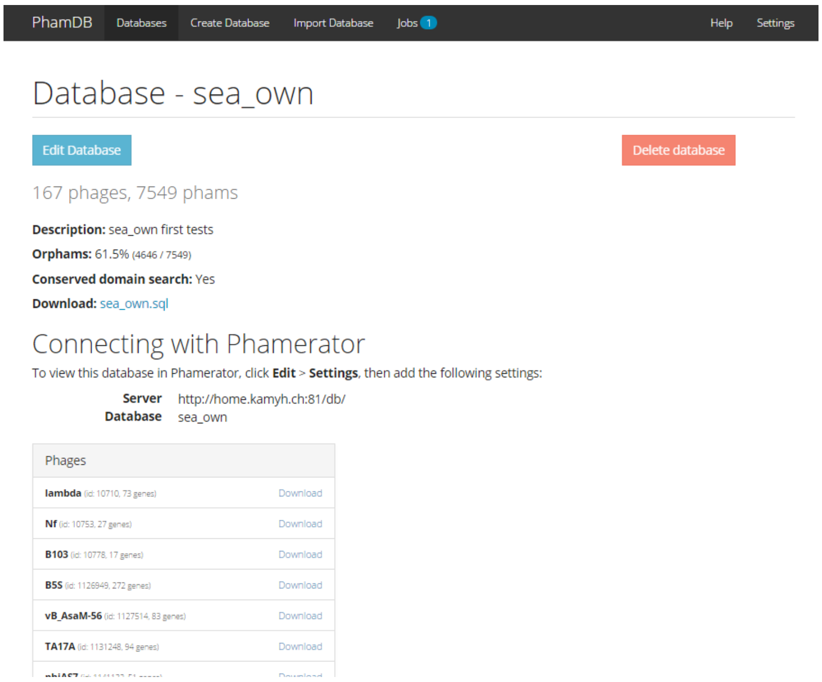


sea_own 167 phages, 7549 phams

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.



Database - sea_own

167 phages, 7549 phams

Description: sea_own first tests
Orphams: 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:

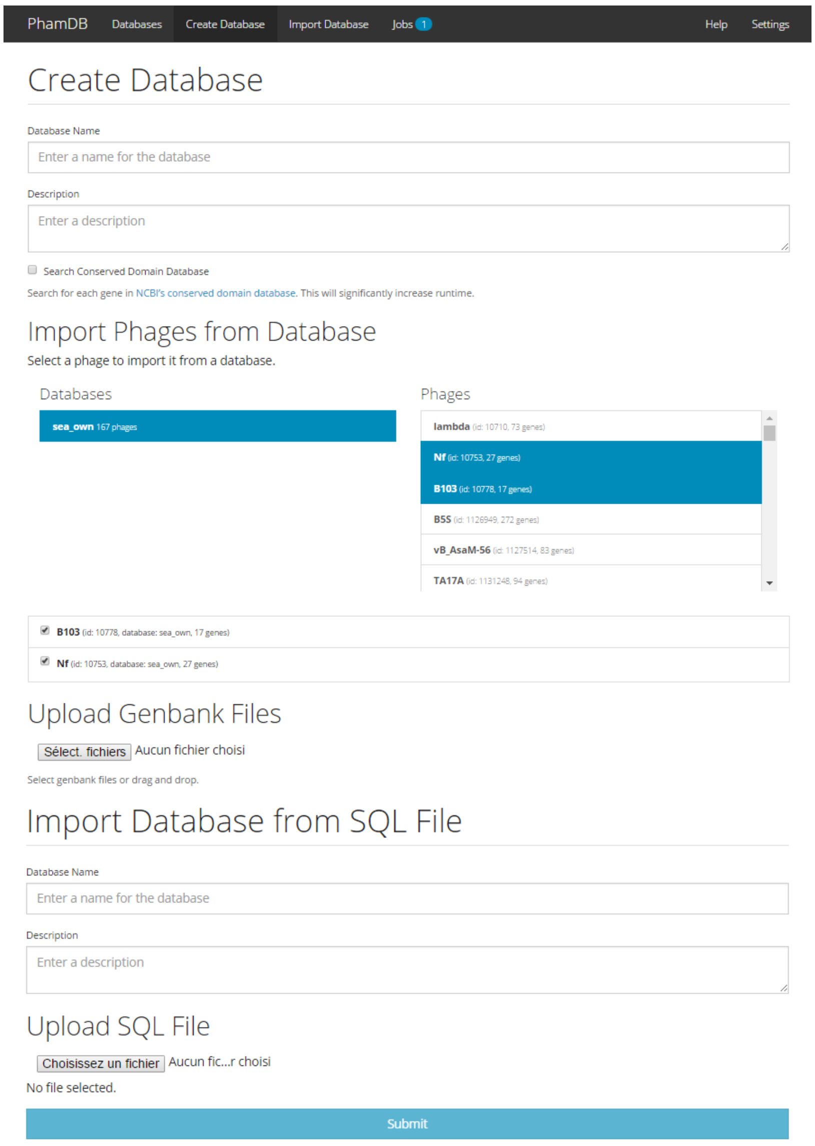
Server	http://home.kamyh.ch:81/db/
Database	sea_own

Phages	
lambda (id: 10710, 73 genes)	Download
Nf (id: 10753, 27 genes)	Download
B103 (id: 10778, 17 genes)	Download
B5S (id: 1126949, 272 genes)	Download
vB_AsaM-56 (id: 1127514, 83 genes)	Download
TA17A (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 uploding Genbank files.
3. By importing as an Sql file.



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 167 phages

Phages

lambda (id: 10710, 73 genes)
Nf (id: 10753, 27 genes)
B103 (id: 10778, 17 genes)
B5S (id: 1126949, 272 genes)
vB_AsaM-56 (id: 1127514, 83 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

Sélect. 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 fic...r choisi

No file selected.

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 .

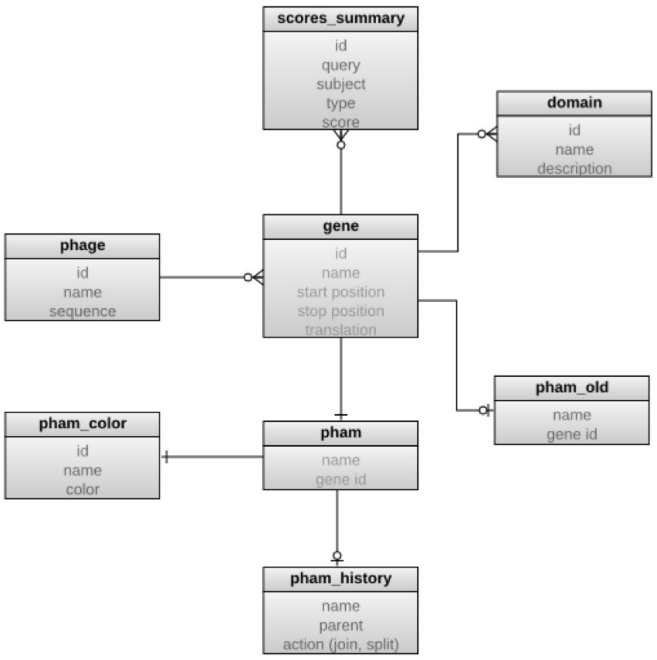


Figure 3.10: Docker harchitecture

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).

```
inphinity = Inphinity('sea_own')
#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.

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

Figure 3.12: Phame names listing

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

```
inphinity.verbose = False
```

Figure 3.13: Verbosity attribut

3.5.2 Phages selection

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

```
def build_tree(self,pham):
    genes = self.get_genes_from_a_pham(pham)
    self.create_fasta(genes)
    self.align_muscle()
    self.compute_tree()
    self.prepare_tree_fig()
```

Figure 3.14: Creation of philogenetic tree

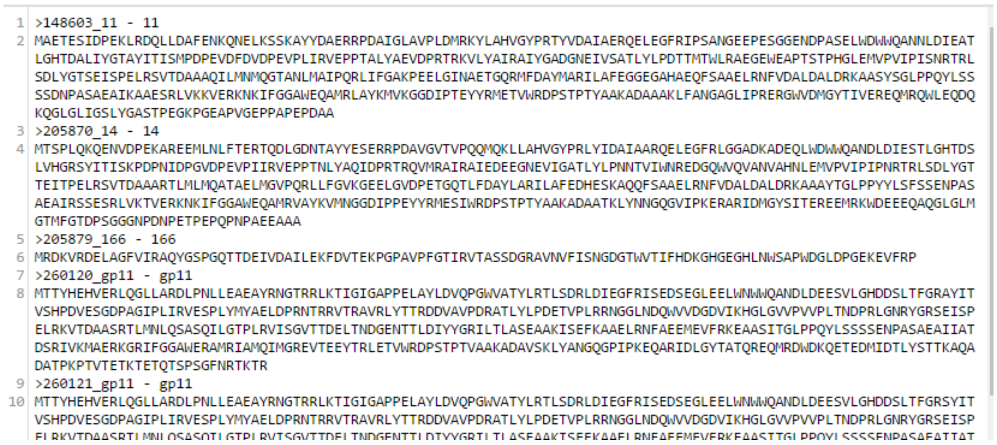
As you can see, this methods those a couple of different things. First we're retriving all the genes composing our pham.

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.

```python
def create_fasta(self, genes):
    print('Creation of the FASTA file')
    fasta = open("%sfasta.fa" % (self.out_dir), "w")
    self.print_("Number of Genes: %d" % (len(genes)))
    for gene in genes:
        GeneID = gene['GeneID']
        name = gene['Name']
        description = ">%s - %s" % (GeneID, name)

        translation = gene['translation']

        self.print_(description)
        self.print_(translation)

        fasta.write(description)
        fasta.write('\n')
        fasta.write(translation)
        fasta.write('\n')

    fasta.close()
```

Figure 3.15: FASTA creation function

The FASTA file should looks 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 fine in annexe ()



```
>148603_11 - 11
MAETESIDPEKLRDQLLDAFENKQNELKSSKAYYDAERRPDAIGLAVPLDMRKYLAHVGYPRTYVDAIAERQELEGFRIPSANGEEPESGGENDPASELWDWWQANNLDIEAT
LGHTDALIYGTAYITISMPDPEVDFDVDPEVPLIRVEPPTALYAEVDPRTRKVLYAIRAIYGADGNEIVSATLYLPDTTMTWLRAEGEWEAPTSTPHGLEMVPVIPISNRTRL
SDLYGTSEISPELRSVTDAAAQILMNMQGTANLMAIPQRLIFGAKPEELGINAETGQRMFDAYMARILAFEGGEGAHAEQFSAAELRNFVDALDALDRKAASYSGLPPQYLSS
SSDNPASAEAIKAAESRLVKKVERKNKIFGGAWEQAMRLAYKMVKGGDIPTEYYRMETVWRDPSTPTYAAKADAAAKLFANGAGLIPRERGWVDMGYTIVEREQMRQWLEQDQ
KQGLGLIGSLYGASTPEGKPGEAPVGEPPAPEPDAA
>205870_14 - 14
MTSPLQKQENVDPEKAREEMLNLFTERTQDLGDNTAYYESERRPDAVGVTVPQQMQKLLAHVGYPRLYIDAIAARQELEGFRLGGADKADEQLWDWWQANDLDIESTLGHTDS
LVHGRSYITISKPDPNIDPGVDPEVPIIRVEPPTNLYAQIDPRTRQVMRAIRAIEDEEGNEVIGATLYLPNNTVIWNREDGQWVQVANVAHNLEMVPVIPIPNRTRLSDLYGT
TEITPELRSVTDAAARTLMLMQATAELMGVPQRLLFGVKGEELGVDPETGQTLFDAYLARILAFEDHESKAQQFSAAELRNFVDALDALDRKAAAYTGLPPYYLSFSSENPAS
AEAIRSSESRLVKTVERKNKIFGGAWEQAMRVAYKVMNGGDIPPEYYRMESIWRDPSTPTYAAKADAATKLYNNGQGVIPKERARIDMGYSITEREEMRKWDEEEQAQGLGLM
GTMFGTDPSGGGNPDNPETPEPQPNPAEEAAA
>205879_166 - 166
MRDKVRDELAGFVIRAQYGSPGQTTDEIVDAILEKFDVTEKPGPAVPFGTIRVTASSDGRAVNVFISNGDGTWVTIFHDKGHGEGHLNWSAPWDGLDPGEKEVFRP
>260120_gp11 - gp11
MTTYHEHVERLQGLLARDLPNLLEAEAYRNGTRRLKTIGIGAPPELAYLDVQPGWVATYLRTLSDRLDIEGFRISEDSEGLEELWNWWQANDLDEESVLGHDDSLTFGRAYIT
VSHPDVESGDPAGIPLIRVESPLYMYAELDPRNTRRVTRAVRLYTTRDDVAVPDRATLYLPDETVPLRRNGGLNDQWVVDGDVIKHGLGVVPVVPLTNDPRLGNRYGRSEISP
ELRKVTDAASRTLMNLQSASQILGTPLRVISGVTTDELTNDGENTTLDIYYGRILTLASEAAKISEFKAAELRNFAEEMEVFRKEAASITGLPPQYLSSSSENPASAEAIIAT
DSRIVKMAERKGRIFGGAWERAMRIAMQIMGREVTEEYTRLETVWRDPSTPTVAAKADAVSKLYANGQGPIPKEQARIDLGYTATQREQMRDWDKQETEDMIDTLYSTTKAQA
DATPKPTVTETKTETQTSPSGFNRTKTR
>260121_gp11 - gp11
MTTYHEHVERLQGLLARDLPNLLEAEAYRNGTRRLKTIGIGAPPELAYLDVQPGWVATYLRTLSDRLDIEGFRISEDSEGLEELWNWWQANDLDEESVLGHDDSLTFGRSYIT
VSHPDVESGDPAGIPLIRVESPLYMYAELDPRNTRRVTRAVRLYTTRDDVAVPDRATLYLPDETVPLRRNGGLNDQWVVDGDVIKHGLGVVPVVPLTNDPRLGNRYGRSEISP
ELRKVTDAASRTLMNLQSASQILGTPLRVISGVTTDELTNDGENTTLDIYYGRILTLASEAAKISEFKAAELRNFAEEMEVFRKEAASITGLPPQYLSSSSENPASAEAIIAT
```

Figure 3.16: PhamDb database creation

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

```python
def align_muscle(self):
        print('Alignment with MUSCLE')
        muscle_loc = r'/home/pa/work/muscle3.8.31_i86linux64' # modifier si nécessaire

        muscle_cline = MuscleCommandline(cmd=muscle_loc,input='%sfasta.fa' %
                (self.out_dir),out='%sout.aln' % (self.out_dir),clwstrict=True)
        stdout, stderr = muscle_cline()

        muscle_align = AlignIO.read('%sout.aln' % (self.out_dir),'clustal')
        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.

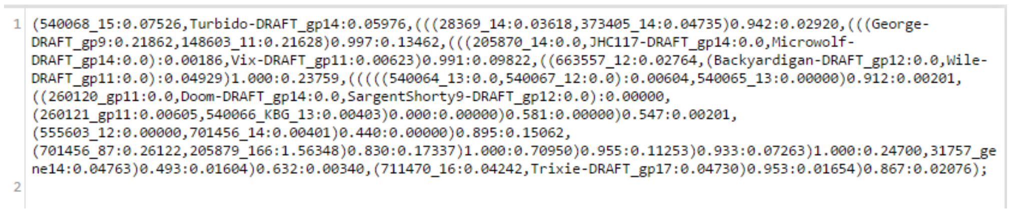
Now we have our alignments ready to compute a phylogenetic tree. To realise the tree we use *FastTree* software. It will produce "approximately-maximum-likelihood phylogenetic trees" using our .aln file proviously generated.

```python
def compute_tree(self):
    print('Compute tree')
    AlignIO.convert('%sout.aln' % (self.out_dir),'clustal','%sintermediate.phy' \
            % (self.out_dir), 'phylip-relaxed')

    cmd_fasttree = r'fasttree'
    fasttree_cmdline = FastTreeCommandline(cmd=cmd_fasttree,fastest=True, \
            input='%sintermediate.phy' % (self.out_dir),out='%stree.tre' % (self.out_dir))
    out_log, err_log = fasttree_cmdline()

    self.print_('Out Log:')
    self.print_(out_log)

    self.print_('Error Log')
    self.print_(err_log)

    self.tree = Phylo.read('%stree.tre' % (self.out_dir), 'newick')
```

Figure 3.18: FASTA creation function

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

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



```
(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);
```

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 disscuss the result in the chapter 4.

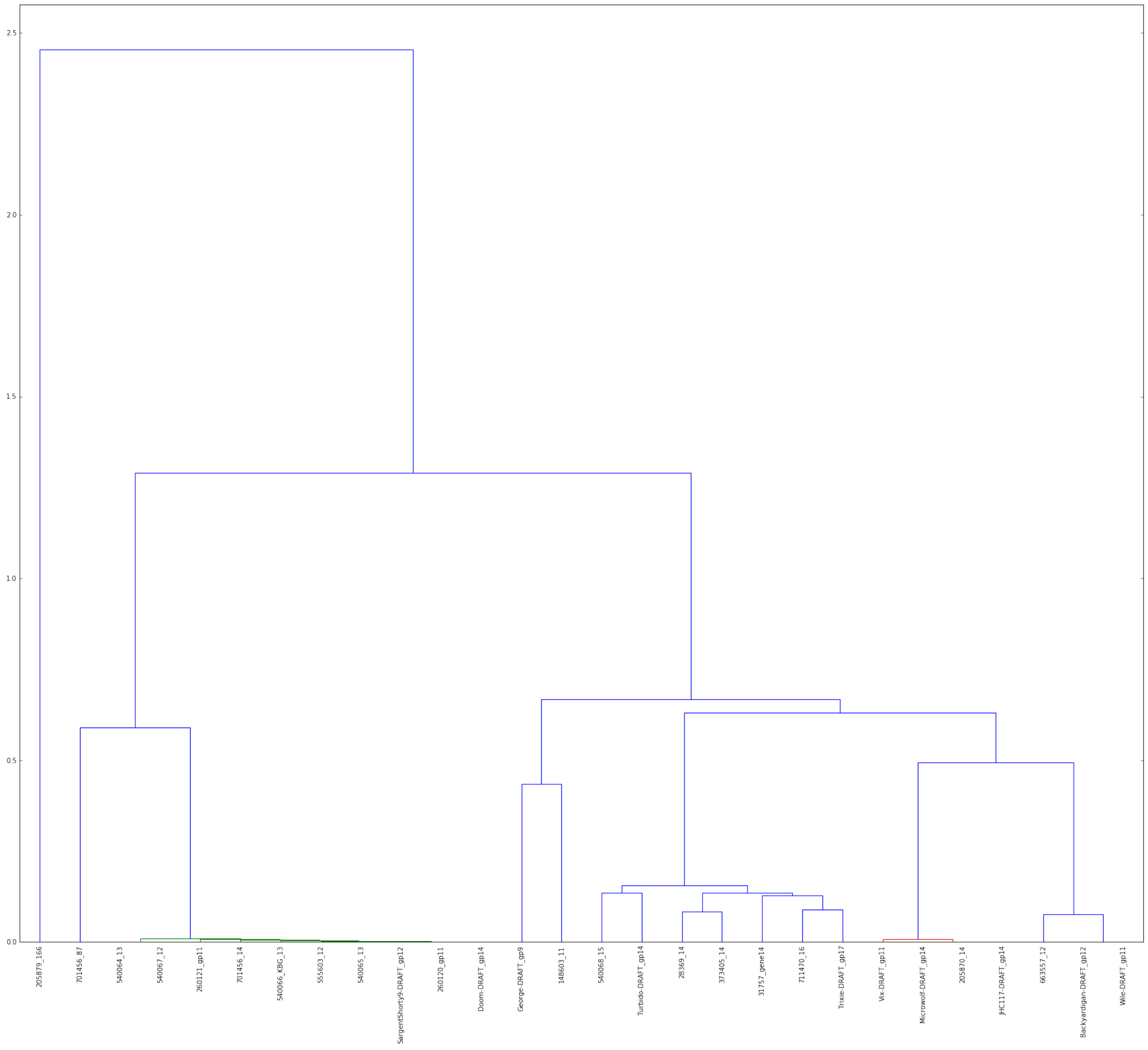


Figure 3.20: PhamDb database creation

3.5.3 Displaying results

you can display information concerning all pages of the selected Pham in text format.

```
inphinity = Inphinity('sea')
inphinity.verbose = False
inphinity.build_tree('2799')
inphinity.print_informations_on_phages(inphinity.leaves)
```

Figure 3.21: FASTA creation function

We will disscuss the content of those information in section 4.1.2. The avalaible informations are the phage gene id who's been use to build the tree, the phage identification number, the phage name, the page accession identifier on Genbank and the phage host strain.

The host strain is labeled in three colors. Blue mean that the host strain is known from the loaded database. Red mean that host strain informtaion comes from GenBank access. Finally, informations are retrieve using webscraping on Phagedb.org.

3.5.4 Data completion

As said in previous section, 3.5.3, the original database was not fully populate. Informtaion about host strain was sometimes missing. In the future we will complete this database but for now the software retrieve them every time it's needed.

```python
def get_host_from_genbank(self, genome_id):

    try:
        record = Entrez.efetch(db="nuccore", id=genome_id, rettype="gb", retmode="text")

        filename = 'out/genBankRecord.gb'
        with open(filename, 'w') as f:
            f.write(record.read())
        parsed_gb_file = next(SeqIO.parse(filename, "genbank"))

        #print(parsed_gb_file)
        return parsed_gb_file.annotations["source"]

    except:
        return 'Not Found'

def get_informations_from_phage_db(self, phage_name):
    page = requests.get('http://phagesdb.org/phages/%s' % (phage_name))
    tree = html.fromstring(page.content)

    host = tree.xpath('//*[@id="phageDetails"]/tbody/tr[2]/td[2]/a/em/text()')[0]

    return host
```

Figure 3.22: FASTA creation function

4 Results & Analyse

In this chapter we'ill take about the results produce during this project.

4.1 First result

I've not had lots of time to produce and analyze many results and the missing host strain in the data as slowed the work.

We can produce phylogenetic tree for any calculated Pham in database. For the moment we treat all gene from a tree the same way. See in section 5.2.4 for future developpment.

For better lisibility, scores using in the tree building phase are not directly draw on it. To display scores, to analyze results use the methode from figure 4.1. It give pairwise distances between two leaves.

```
def display_scores(self):
    print('Display Scores')

    self.leaves = [str(cladit) for k,cladit in enumerate(self.tree.get_terminals())]
    for l1,leave1 in enumerate(self.leaves):
        for l2,leave2 in enumerate(self.leaves):
            distance = self.tree.distance(leave1,leave2)
            if distance > 0:
                print('----------------------------')
                print('%s - %s' % (leave1,leave2))
                print(distance)
```

Figure 4.1: FASTA creation function

For pham *2799* we can split the pham into parts using those distances. On figure 3.20 those part are colored differentlly depending on ther appartenence.

Using a figure tree is not really practicle to make things automatic. So funtion *prepare_tree_fig* set the attribute *self.leaves*. This attribute store all gene composing a pham. Using the code from figure 3.21 it show for all gene, the phage and his host strain.

As you can see in annexe *res_pham_2799_host_strain.pdf*, every host strain for this pham is **Mycobacterium smegmatis**.

Note: See issue at section 5.2.3 about phage host strain in the current database.

4.2 Database

4.2.1 SEA

For this phase of the project have used the *SEA* database [4]. This db is populated with 113 phages and 2771 phams.

Note: See in annexes, file *SEA.sql* for the db sql dump.

We've used PhamDb to add some more phage to the database. For now the new database, *SEA_own*, contains 167 phages and 7549 phams. An issue with PhamDb, cf. section 5.1.3, is responsable for the fact that we only add 84 new phages.

We have generated a list of phage from *Ebi* [3]. A file containing this list has been generated, cf annexes *phages_list_1.txt.* Using the code from *inphinity/GeneBank_Fetch.ipynb* we can store in GenBank files informations and sequences about our new phages.

5 Conclusion

5.1 Problems encountered

5.1.1 Phamerator Installation

Phamerator installation has been difficult. Some packages used in Phamerator are not all listed on their installation procedure. In addition, BioPython package last version seems not working sith Phamerator.

For not loosing your time, you can find a installation bash in annexes, *phamerator.sh*. This is the procedure I've came up with for installing and running Phamerator on Linux Ubuntu.

5.1.2 SEA database

As already said (section 3.5.4) the SEA database taken from Phamerator was missing some information. Furthermore we will certainly add permanantly to the db those missing information in the future.

5.1.3 PhamDB Limitation jobs

PhamDB is an excellent tool but for an unknown reason, if you load to many import in one job, it is uncapable to finish the job.

5.1.4 PhageDB.org

Some of the information were not present on GenBank. Thier are retrieve from PhageDB.org. But this website does not have an API to get data. For this particuliar reason, information are retrieve using webscraping.

Webscraping consiste of getting an internet (HTML) page and parsing it. We use HTML balises to locate information in the document. Because we use document structure to locate information, it is not garanteed to work in the future. Indeed, if the page structure changes the retrive methods will not work anymore.

5.2 Perspectives

5.2.1 Database population

In the purpose to realise a full automated pipeline, it will be interesting to take the PhamDb execution code without GUI elements. Indeed, at the end we can parse data from GenBank or FASTA files and inject them into PhamDB. When PhamDb will have process those data, the code from *inphinity/show_phamily.ipynb*.

5.2.2 Resultats validation

At the moment, we are not sure if our assomption is right, *Is genes of a Pham are from phages infecting the same bateria ?*. To answer this question we need to add more phage to the database.

After adding more phage we can make a script to iterate on all generated Pham and check if host strain are the same for all phage in a Pham.

5.2.3 Sub-tree selection

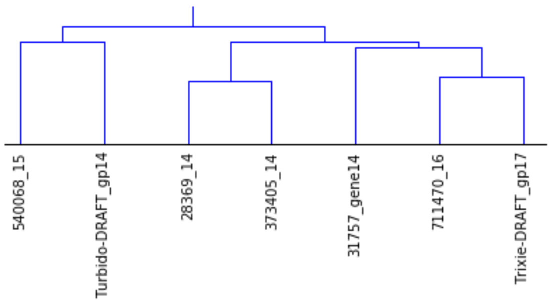


Figure 5.1: PhamDb database creation

The figure 5.1 is a selection from figure 3.20. It will be interesting to extend the function in figure 4.1 to select those sub-parts. Wether listing all the sub-parts, wether by selecting a part from a specific gene.

List of Figures

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] ENA. European nucleotide archive. `http://www.ebi.ac.uk/genomes/phage.html`.

[4] http://seaphages.org/. Sea database. `http://phamerator.csm.jmu.edu/sea/`.

[5] 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.

[6] NCBI. Genbank overview. `http://www.ncbi.nlm.nih.gov/genbank/`.

[7] 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/`.

[8] NIH. A brief guide to genomics. `https://www.genome.gov/18016863/`.

[9] Phamerator. Phamerator: a bioinformatic tool for comparative bacteriophage genomics. `http://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-12-395`.

8 Annexes

1. **phages_list_1.txt**
2. **phamerator.sh**
3. **respham2799hoststrain.pdf**
4. **SEA.sql**
5. **show_phamily.ipynb.pdf**
6. **GeneBank_Fetch.ipynb.pdf**