Introduction to Metagenomics

Milot Mirdita, Wanwan Ge, Franco Simonetti, Annika Seidel, Eli Levy Karin, Johannes Söding

October 30th, 2018

1 Introduction

Microbial communities are major players of Earth's ecosystems. The study of their genomes could be of great importance not only for ecology and evolution, but also for the discovery of new potentially useful enzymes and metabolites. Most microorganisms are impossible to cultivate in laboratory conditions, but we can analyze their genome by directly sequencing samples from their natural habitats. Direct analysis of genomes contained in environmental samples is known as *metagenomics* and is a promising way to get the most information about microbial populations in specific ecosystems. Increasing usage of shotgun sequencing in metagenomics has led to rapid accumulation of data, used for discovery of new pathways, genes and species [1, 2, 3].

A typical metagenomic analysis starts with collecting the samples from an environment of interest followed by DNA extraction from these samples. Afterwards, the extracted DNA is sequenced, providing a mixture of billions of short sequences (so-called *reads*, with a length of about 250 bp using modern technology) over the A,C,G,T alphabet of fragments of different genomes of the microorganisms present in the sample.
The downstream analysis of these reads is only computational: quality control of the reads, assembling, annotation, etc. In a typical metagenomic project, there can be hundreds of different species and tens of millions of proteins.
The purpose of this hands-on tutorial is to get familiar with modern computational tools to handle metagenomic data and get some insight into the biology behind it :)

On the Internet, there are many public resources that store metagenomic data. MG-RAST[1] is a database providing most of the metagenomics reads publicly available, together with some basic data analysis.

[1] `http://metagenomics.anl.gov`

2 Playground

We used the MG-RAST server to download four different sets of protein sequences coming from different environments. *Your mission, if you choose to accept it*, is to identify those environments using our metagenomics toolkit! Be creative![2]

2.1 Linux

Throughout this tutorial you will work in a **Linux** environment. Briefly, Linux is a descendant of the UNIX operating systems family. It is popular because it is open-source, free and runs on everything from tiny micro controllers, to phones, computer clusters and even super computers. It has found wide adoption in the bioinformatics community. An operating system has many important roles, which include:

- managing a file system: information (generally: "files") is stored on the computer hard disk. The operating system manages the access to files. To do so, it represents their location as a tree hierarchy. Each file has a **path**, starting from the root and going through **directories**. For example:

  /most_important_project_ever/omg_important/seriously_important.txt

- managing resources: all software running on the computer cannot access its resources directly but rather, they get services from the operating system, which makes sure the resources are allocated fairly and safely. The same is true for us, **users** of the computer.

If we want to save a new file to the disk, we do it through the operating system. We usually do it using a graphical interface (press some button and save). Today we will communicate with the Linux operating system using **a textual interface**.

2.2 Bash

A "**Shell**" is a basic textual interface to communicate with the operating system. We do so by typing commands in a designated command window. These commands allow us for example, to create a new file or to navigate to some directory. Below you will get familiar with a few basic textual commands in a specific type of Linux Shell, called "**Bash**".

Now, in the Bash window, let's type the following commands:

```
# print working directory: the full path from the root of the current directory
pwd
```

This should result in your **home directory**:

```
# change directory: navigate to the data directory under your home directory
cd data
```

[2]You can later download the data yourself and reproduce the tutorial at home. On MG-RAST, there are currently 350,050 metagenomes (October 2018), so feel free to explore :)

Validate that your location (directory) has indeed changed.

```
# list files and sub-directories in the directory:
ls
```

You should see:

- crAssphage.fas
- metagenomic_dataset.faa
- useful_links.txt

Bash Tip 1 To avoid typos and save time, if you partially type a command or a file name, you can press the TAB key to get the automatic completion of your command or file. If what you are typing cannot be uniquely completed, you can press the TAB key twice to see a list of suggestions.

In this tutorial, whenever you see YourSomething it means you need to replace it with a value you choose.

```
# create a copy of a file:
cp useful_links.txt YourFileNameCopy

# print the first 5 lines of a file:
head -n 5 useful_links.txt

# print the entire content of a file to the screen:
cat useful_links.txt
```

Validate that useful_links.txt and YourFileNameCopy have the same content

```
# print the number of lines in a file:
wc -l useful_links.txt

# remove a file (permanently deletes it! Achtung!!!):
rm YourFileNameCopy
```

Now, let's play with directories.
In the commands below, instead of YourDirName, you can type any name you choose.

```
# make directory: create a directory in the current location.
mkdir YourDirName
```

Change directory to YourDirName and validate that you are indeed in the right location

```
# go back to the parent directory:
cd ..

# remove a directory (-r for "recursive"; permanently deletes it! Achtung!!!):
rm -r YourDirName
```

Several Bash commands can be carried out one after the other using pipes `|`. The output of one program is given ("piped") to the next one, that then processes this output.

```
# pipe example to count the number of files in the current directory:
pwd | ls | wc -l
```

How many files do you see?

Some more commands are described in the appendix 6.1.
There are many more features to Bash. Check out this resource to learn more: ryanstutorials.net/linuxtutorial

Today, we will use Bash to run metagenomics software.

Bash Tip 2: To cancel a running program you can press `Ctrl` + `C`.

2.3 File formats

Biological information is conventionally stored in specific textual formats. These formats indicate where, for example the name of the gene/protein is stored and where the sequence itself is stored. This way bioinformatic tools can extract the needed information from the files. One of the most common formats is called **FASTA**. It is used, for example, to store metagenomics sequence reads. In FASTA format, an identifier (a protein ID, for example) is written after the ">" symbol, and its corresponding sequence is written in the following lines.
The **tsv** (tab separated values) formatted files, with which you are going to work later, contain one record per line, with attributes about this record separated by "TAB" characters. This is a common representation of data in bioinformatics and easy to explore with standard Linux tools.

2.4 Plass

Plass is a protein-level assembly[3] [4]. It is designed to take sequenced DNA/RNA reads as input and assemble them together to full or partial proteins. It does so by translating each read in 6 frames and detecting overlapping reads ultra-fast. You can explore its usage by running its main command in the terminal:

```
plass
```

```
# create a directory for plass and navigate to it:
cd ~
mkdir plass
cd plass
```

```
# assemble the metagenomic dataset using plass:
plass assemble ~/data/metagenomic_dataset.faa metagenomic_dataset_assembled.faa temp
```

How many assembled proteins did Plass produce[4]?

[3] Larn more about this tool at github.com/soedinglab/plass

[4] Solution: *326 647*

2.5 MMseqs2

MMseqs2 (standing for *Many to Many Sequence Search*) is a suite of tools for searching, clustering, and filtering protein sequence sets, from single sequences to billions of sequences. Its speed and ability to process large volumes of data make it perfect for the analysis of metagenomic data. You can explore its usage by running its main command in the terminal[5]:

```
mmseqs
```

```
# create a directory for MMseqs2 analyses and navigate to it:
cd ~
mkdir mmseqs2
cd mmseqs2
```

2.5.1 Working with MMseqs2

FASTA files (such as the one produced by Plass) need to be converted to the MMseqs2 format to be able to used by MMseqs2 modules:

```
# create a db that MMseqs2 can read from a FASTA file
mmseqs createdb ~/plass/metagenomic_dataset_assembled.faa YourDBname
```

`YourDBname` is a placeholder name, give your files descriptive and concise names[6]!

You can check that the database files are actually created:

```
# lists the files in the directory
ls
```

If everything is fine, you should see four files:

```
YourDBname
YourDBname.index
YourDBname_h
YourDBname_h.index
YourDBname.dbtype
YourDBname.lookup
```

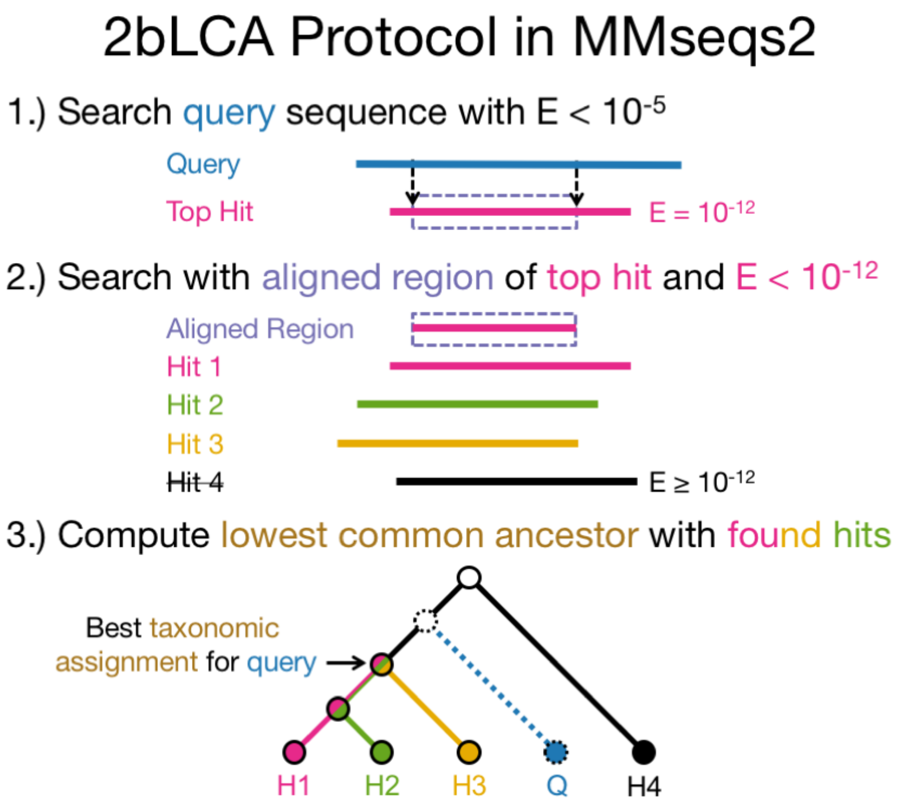
The `YourDBname_h` file contains the header descriptor of your sequences, each separated by a `NULL` byte. The `YourDBname` file contains the sequences, also separated by `NULL` bytes. The `.index` files contain descriptions of the corresponding data file for fast access.

[5]To learn more about this tool, the complete MMseqs2 documentation can be found at github.com/soedinglab/MMseqs2/wiki

[6]A wise grey beard once said: “There are two hard things in computer science: cache invalidation, naming things, and off-by-one errors.”

2.6 Identify taxonomy composition

To get an insight about an ecosystem, one can identify the taxonomy of the organisms in the samples. One way to do so is to search the sequences you have freshly packed into an MMseqs2 database against a target database of reference sequences for which we know the taxonomies. By identifying homologs through searches with taxonomy annotated reference databases, MMseqs2 can compute the lowest common ancestor. This lowest common ancestor is a robust taxonomic label for unknown sequences. By default, MMseqs2 implements the 2bLCA protocol [5] for choosing a robust LCA.


2bLCA Protocol in MMseqs2
1.) Search query sequence with E < 10^-5
Query
Top Hit
E = 10^-12
2.) Search with aligned region of top hit and E < 10^-12
Aligned Region
Hit 1
Hit 2
Hit 3
~~Hit 4~~
E ≥ 10^-12
3.) Compute lowest common ancestor with found hits
Best taxonomic assignment for query
H1
H2
H3
Q
H4

One of the most important manually curated reference databases is SwissProt [7][8]. We have already built the necessary databases (see how in 6.2 and 6.3) and prepared the taxonomy tree of life from the NCBI in your environment (see 6.4). You can find the resulting database files here:

```
# list the files in the directory
ls ~/databases/swissprot
```

Let's get the taxonomy of the sequences contained in your database:

[7] http://www.uniprot.org/uniprot/?query=reviewed%3Ayes

[8] For a more in-depth analysis you would, for example, use our Uniclust databases [6]. These are the bigger brothers of the SwissClust30 database we prepared for you. They were built by clustering all 126 million UniProt sequences.

```
# run MMseqs2 2bLCA taxonomy protocol
mmseqs taxonomy YourDBname ~/databases/swissprot/swissclust30_2018_10_seed_db
 ↪  ~/databases/swissprot/swissclust30_2018_10_OX.tsv ~/databases/ncbi/
 ↪  YourDBnameTaxa tmp -s 3

# create a tab-separated file of the search that is human-readable
mmseqs createtsv YourDBname YourDBnameTaxa YourDBnameTaxa.tsv
```

2.6.1 Tasks

- The taxonomic labels appear on the fourth column of the TSV file. Examine the first 50 most recurrent labels. Use the list of commands in the appendix 6.1 [9].
- Can you say anything about the taxonomic units you see?
- In the list you will find *Arabidopsis thaliana* as well as *Mus musculus* and *Escherichia coli* – what can be the reason for that?
- One of the labels is "Candidatus Pelagibacter ubique HTCC1062". Google it. What can you learn from it?

There are many databases on the web that you can use to learn more about specific species. The NCBI Taxonomy Browser [10] gathers all those resources in one place. Search for some of the most represented taxa in your tree. Wikipedia is another great taxonomy resource. Try to find out more about the organisms and their habitats in your dataset.

2.7 Looking for molecular functions

You can annotate (infer the function) of the sequences in your databases by searching with MMseqs2 for highly similar sequences in the reference database:

```
# learn about the possible search parameters by typing:
mmseqs search

# run a search with specific parameters:
mmseqs search YourDBname ~/databases/swissprot/swissclust30_2018_10_seed_db
 ↪  YourSearchResult tmp -s 2 -c 0.9 --min-seq-id 0.8

# creates a human-readable tab-separated file of the search results
mmseqs createtsv YourDBname ~/databases/swissprot/swissclust30_2018_10_seed_db
 ↪  YourSearchResult YourSearchResult.tsv
```

2.7.1 Tasks

- Can you find out the function of the most abundant proteins[11]?
- Look up some of the most abundant target identifiers in the UniProt website to get a feeling of the data.

[9]Solution (try yourself first): *cut -f4 YourDBnameTaxa.tsv | sort | uniq -c | sort -n -r | head -n 50*

[10]`https://www.ncbi.nlm.nih.gov/Taxonomy/Browser/wwwtax.cgi`

[11]Hint: introduce minor changes to solution above

3 Deep annotation with MMseqs2, example of a recently discovered virus in the human gut

To explore the sensitive search options of MMseqs2, let take a look at the genome of a recently discovered wide-spread bacteriophage, called “crAssphage” (named after the technique used to discover it, the cross-assembly). You can find its genome on the NCBI website[12]. We already placed the FASTA file in the `/data` folder:

```
# check if the crAssphage FASTA file is there
ls ~/data
```

You can create a MMseqs2 database from the FASTA file using the createdb command of MMseqs2. Then, let’s extract all the open reading frames (potential proteins) of this phage:

```
cd ~/mmseqs2

mmseqs createdb ~/data/crAssphage.fas YourCrassphageDB

# extract only potential proteins of at least 60 amino-acids:
mmseqs extractorfs YourCrassphageDB crassphageOrfs --min-length 60

# translate the ORFs to amino-acid sequences:
mmseqs translatenucs crassphageOrfs crassphageProts
```

Let’s check what a sensitive search will output as results:

```
# run a sensitive search (-s 7.5, similar to BLAST) and keep distantly related
 ↪  homologs (e-value of 1.0: -e 1) with MMseqs2:
mmseqs search crassphageProts ~/databases/swissprot/swissclust30_2018_10_seed_db
 ↪  crassPhageAnnotation tmp -s 7.5 -e 1
mmseqs createtsv crassphageProts ~/databases/swissprot/swissclust30_2018_10_seed_db
 ↪  crassPhageAnnotation crassPhageAnnotation.tsv
```

You can check the different hits you get:

```
cat crassPhageAnnotation
```

- What E-values do you get? Are they reliable?
- Check some of the UniProt identifiers on the UniProt website. Do they make sense?

To be more sensitive, we can use iterative searches:

```
# run an iterative search:
mmseqs search crassphageProts ~/databases/swissprot/swissclust30_2018_10_seed_db
 ↪  crassPhageAnnotationIterative tmp --num-iterations 3
mmseqs createtsv crassphageProts ~/databases/swissprot/swissclust30_2018_10_seed_db
 ↪  crassPhageAnnotationIterative crassPhageAnnotationIterative.tsv
```

What E-values do you get? Are they more reliable? Check the UniProt annotations associated to the best E-values. Do your results make sense?

[12] `https://www.ncbi.nlm.nih.gov/nuccore/NC_024711.1?report=fasta`

Filtering a result databases You can post-process the annotation file to retrieve only annotations of high confidence:

```
# check that the E-values are shown in column 4 of the search result file
head crassPhageAnnotationIterative

# create a new database containing
# only annotations of e-value <= 1e-5
mmseqs filterdb crassPhageAnnotationIterative crassPhageAnnotationIterative_good_eval
 ↪  --filter-column 4 --comparison-operator le --comparison-value 1e-5

mmseqs createtsv crassphageProts ~/databases/swissprot/swissclust30_2018_10_seed_db
 ↪  crassPhageAnnotationIterative_good_eval
 ↪  crassPhageAnnotationIterative_good_eval.tsv
```

4 Sequence Clustering

In the previous Plass section, you assembled the reads of a metagenomic sample. Since Plass assembles with replacement of reads, the assembled protein catalogue will contain some redundancy. You can reduce this redundancy by clustering the catalogue, for instance, to 90% of sequence identity, and demanding for the representative sequences that cover at least 95% of the members. For this, you can either use the `cluster` (sensitive clustering) or `linclust` (linear time fast clustering) workflows of MMseqs2. MMseqs2 also offers variants of the default workflows that take a FASTA input and return human readable results. One such workflow is the `easy-linclust` workflow:

```
mmseqs easy-linclust ~/plass/metagenomic_dataset_assembled.faa clusteredProts tmpDir
 ↪  --min-seq-id 0.9 -c 0.95 --cov-mode 1
```

Both the default MMseqs2 clustering and Linclust link two sequences by an edge based on three local alignment criteria:

- a maximum E-value threshold (option `-e`, default 10^{-3}) computed
- a minimum coverage (option `-c`), which is defined by the number of aligned residue pairs divided by either the maximum of the length of query/centre and target/non-centre sequences alnRes/max(qLen,tLen) (default mode, `--cov-mode` 0) or by the length of the target/non-centre sequence alnRes/tLen (`--cov-mode` 1)
- a minimum sequence identity (`--min-seq-id`) defined as the number of identical aligned residues divided by the number of aligned columns including internal gap columns

You can count the number of entries in your clustered FASTA file `clusteredProts_rep_seq.fasta` again using the previous `grep` command.

Learn how to deal with MMseqs2's indexed databases Let us do the long version of the previous `easy-linclust` command again. First we create the database:

```
mmseqs createdb ~/plass/metagenomic_dataset_assembled.faa YourDBname
```

Then re-run the clustering of the catalogue database with the database you just created:

```
mmseqs linclust YourDBname YourDBnameReduced tmpDir --min-seq-id 0.9 -c 0.95
 ↪  --cov-mode 1
```

This creates a *cluster database* where **each entry has the key of its representative sequence**, and whose data consists of the list of keys of its members:

```
# the index file contains entries whose
# keys are of those of their representative sequence
head YourDBnameReduced.index

# you will see the keys belonging to different clusters
# (one per line) and such that every cluster is
# separated by a null byte (shown as a ^@ in vim or using less)
less YourDBnameReduced
```

5 Build you own cascaded clustering workflow

We will create our own clustering workflow using MMseqs2's **modular architecture**. This workflow will be a bash script that calls MMseqs2 modules to deeply cluster a set of proteins.

Cascaded sequence clustering Let's first create a cascaded clustering workflow: after a first clustering step, the representative sequences of each of the clusters are searched against each other and the result of the search is again clustered. By repeating this procedure iteratively, one gets a deeper clustering of the original set.
Try to code complete the following script:

```
#!/bin/bash -e
inputdb="YourDBname"
clusteringresults=""
END=3
for (( step=0; step < END; step++ )); do
    mmseqs search $inputdb $inputdb searchStep$step tmpDir
    # here we use the clust module (different from cluster!). This module is only
     ↪  responsible
    mmseqs clust $inputdb searchStep$step clusteringStep$step
    mmseqs result2repseq ...
    inputdb=...
    clusteringresults="$clusteringresults clusteringStep$step"
done

# Then merge the clustering steps into the result database deepClusterDB
mmseqs mergeclusters YourDBname deepClusterDB $clusteringresults
```

Try your script with 3 steps, and check the clustering depth (number of clusters) at each step:

```
wc -l clusteringStep*.index
```

What do you notice ?

6 Appendix

6.1 Some useful Bash commands

```
# show a file inside the terminal
less myFile

# show only the second column from a TSV file
cut -f2 YourFile

# show the lexicographically sorted lines of a file
sort YourFile

# show the numerically sorted lines of a file
sort -n YourFile

# store in YourFileSorted, a sorted version of your file
sort YourFile > YourFileSorted

# show only unique elements in a file (the file needs to be sorted first)
uniq YourFileSorted

# show how often every unique element occurred in a file (file needs to be sorted)
uniq -c YourFileSorted

# another pipe example
# sort lines lexicographically, count appearances of each line and sort by the counts
 ↪  in reverse order
sort YourFile | uniq -c | sort -n -r
```

6.2 How we built the reference database

SwissClust30 is a clustered version of SwissProt to a minimal sequence identity of 30%, built with MMseqs2. We already prepared this database for you. This section is only meant as a reference.

```
# navigate to the data directory in the home directory
mkdir -p ~/databases/swissprot
cd ~/databases/swissprot

# get the swissprot database from the internet:

wget ftp://ftp.expasy.org/databases/uniprot/current_release/knowledgebase/complete/uniprot_sprot.fasta.gz

# rename it to the current release (2018_10, when we prepared this document)
mv uniprot_sprot.fasta.gz uniprot_sprot_2018_10.fasta.gz

# build the MMseqs2 database
mmseqs createdb uniprot_sprot_2018_10.fasta.gz uniprot_sprot_2018_10

# run the clustering
mmseqs cluster uniprot_sprot_2018_10 swissclust30_2018_10 tmp --min-seq-id 0.3 -c 0.8
 ↪  -s 6

# convert the clustering database into a Fasta file
```

```
mmseqs mergedbs swissclust30_2018_10 swissclust30_2018_10_seed uniprot_sprot_2018_10_h
 ↪  uniprot_sprot_2018_10 --prefixes ">"

tr -d '\000' < swissclust30_2018_10_seed > swissclust30_2018_10_seed.Fasta
```

6.3 How we built the taxonomy mapping

```
# navigate to the data directory in the home directory
mkdir -p ~/databases/swissprot
cd ~/databases/swissprot

# get the SwissProt knowledge base from the internet:
wget ftp://ftp.expasy.org/databases/uniprot/current_release/knowledgebase/complete/uniprot_sprot.dat.gz

# create a MMseqs2 database from the SwissClust30 Fasta file
mmseqs createdb swissclust30_2018_10_seed.fasta swissclust30_2018_10_seed_db

# map SwissClust30 IDs to Taxons (you can ignore the "Could not find accession XXXX
# in mapping!" prints)
mmseqs convertkb uniprot_sprot.dat.gz swissclust30_2018_10_seed_db.mapping
 ↪  --kb-columns OX --mapping-file swissclust30_2018_10_seed_db.lookup

# post-processing to get a TSV file with the format
# SwissClust30\_ID\textbackslash{}NCBI\_Taxon
mmseqs prefixid swissclust30_2018_10_seed_db.mapping_OX
 ↪  swissclust30_2018_10_seed_db.mapping_OX_pref
tr -d '\000' < swissclust30_2018_10_seed_db.mapping_OX_pref >
 ↪  swissclust30_2018_10_OX.tsv_tmp

gawk '{match($2, /=([^ ;]+)/, a); print $1"\t"a[1]; }' swissclust30_2018_10_OX.tsv_tmp
 ↪  > swissclust30_2018_10_OX.tsv
```

6.4 How to download the NCBI taxonomy data

```
# create the directory for the NCBI data and navigate to it
mkdir -p ~/databases/ncbi
cd ~/databases/ncbi

# download the taxonomy data from NCBI
wget ftp://ftp.ncbi.nlm.nih.gov/pub/taxonomy/taxdump.tar.gz
# extract
tar xzvf taxdump.tar.gz
```

References

[1] Gene W Tyson, Jarrod Chapman, Philip Hugenholtz, Eric E Allen, Rachna J Ram, Paul M Richardson, Victor V Solovyev, Edward M Rubin, Daniel S Rokhsar, and Jillian F Banfield. Community structure and metabolism through reconstruction of microbial genomes from the environment. *Nature*, 2004.

[2] J Craig Venter, Karin Remington, John F Heidelberg, Aaron L Halpern, Doug Rusch, Jonathan A Eisen, Dongying Wu, Ian Paulsen, Karen E Nelson, William Nelson, et al. Environmental genome shotgun sequencing of the sargasso sea. *Science*, 2004.

[3] Yasir Bashir, Salam Pradeep Singh, and Bolin Kumar Konwar. Metagenomics: an application based perspective. *Chin. J. Biol.*, 2014.

[4] Martin Steinegger, Milot Mirdita, and Johannes Söding. Protein-level assembly increases protein sequence recovery from metagenomic samples manyfold. *bioRxiv*, 2018.

[5] Pascal Hingamp, Nigel Grimsley, Silvia G Acinas, Camille Clerissi, Lucie Subirana, Julie Poulain, Isabel Ferrera, Hugo Sarmento, Emilie Villar, Gipsi Lima-Mendez, Karoline Faust, Shinichi Sunagawa, Jean-Michel Claverie, Hervé Moreau, Yves Desdevises, Peer Bork, Jeroen Raes, Colomban de Vargas, Eric Karsenti, Stefanie Kandels-Lewis, Olivier Jaillon, Fabrice Not, Stéphane Pesant, Patrick Wincker, and Hiroyuki Ogata. Exploring nucleo-cytoplasmic large DNA viruses in Tara Oceans microbial metagenomes. *ISME J.*, 2013.

[6] Milot Mirdita, Lars von den Driesch, Clovis Galiez, Maria J Martin, Johannes Söding, and Martin Steinegger. Uniclust databases of clustered and deeply annotated protein sequences and alignments. *Nucleic Acids Res.*, 2016.