Protein Bioinformatics

Yazhini Alexandra Kolodyazhnaya Hong Su
Michel van Kempen Johannes Söding

November 28 – 29, 2022

Contents

Introduction to Linux and Bash

1.1 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:

  `/home/coder/project/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**.

1.2 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. In bioinformatics, most tools are accessible via the command line (e.g., blast, mmseqs2). Using shell commands, we can execute those tools with the desired parameters (which is often not possible with the web interfaces) and process output files. Below you will get familiar with a few basic textual commands in a specific type of Linux Shell, called **Bash** (short for Bourne Again SHell).

You will work remotely on one of our servers, where we have prepared an integrated

development environment[1] for you that contains a text editor and a shell. We will assign a number NN to each of you. Replace NN with your number in this URL `https://tutorialNN.mmseqs.com` and open it in your browser.

We recommend Firefox, but any browser should work. If you want to download any of the files you produce to your own computer (e.g. for uploading it to a webserver) you can open `https://tutorialNN.mmseqs.com/web` and download the files from there.

You should see something like the following image:

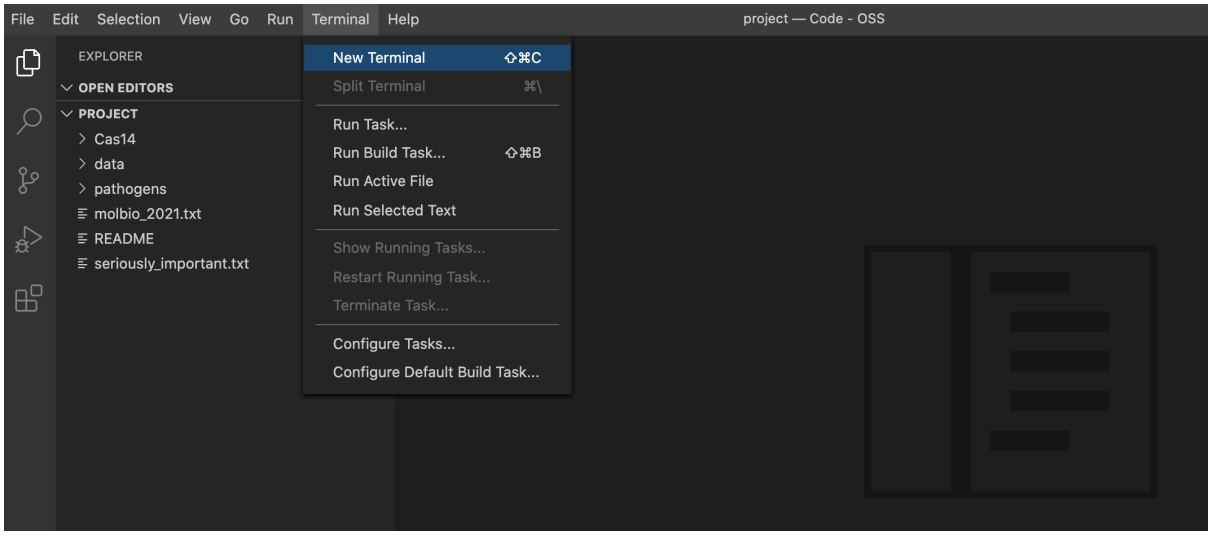


Figure 1.1: You can open a new terminal by clicking "Terminal -> New Terminal".

Now, in the Bash window, let's type the following commands (Lines that start with # are comments and will not be executed if entered):

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

This should result in navigating to a sub-folder of your **home directory**:

```
/home/coder/project
```

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

Validate that your location (directory) has indeed changed.

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

You should see:

- useful_links.txt

[1] `https://github.com/cdr/code-server`

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

Try the following keystrokes:
`cat` SPACE `u` TAB
It should get expanded to the same command as above (as long as you are in the correct directory). You should liberally use TAB-expansion as it will reduce the number of typos you will make.

Bash Tip 2: Use the ↑ ↓ arrow keys to navigate to the previous commands you executed.

Today we will use the integrated text editor to make changes to files instead of also using a shell based text editor. When you have some time you should try to familiarize yourself with one of the popular shell based editors such as `nano`, `vim` or `emacs`.

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

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

# create an empty file:
touch YourFileName

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

# print the last 5 lines of a file:
tail -n 5 useful_links.txt
```

Visually confirm that useful_links.txt and YourFileNameCopy have the same contents.

```
# lists the files in more detail
ls -lah

# 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

# print history of commands that you used
history
```

Later today, we will use Bash to run metagenomics software.

Bash Tip 3: To cancel a running program you can press CTRL + C .

Bash Tip 4: Whenever you are not sure about what a command does or how to run it, you can always look up its manual page with the following command:

```
# show the manual page of a command (quit by pressing 'q')
man <commandtolookup>
# E.g., man mkdir
```

1.3 Text processing in Bash

In Bash, we can take textual data and transform it in a particular way that is more useful for us. We will introduce a few text processing commands in this section.

Note these commands usually have various command line options that will modify their behavior. Some more commands used in this section are described in the appendix 5.1.

The **cut** command lets you select certain columns from a text file if your content is separated into columns.
Options (flags [2]):

- `-f`: indicates columns to print (e.g.: 1,4-9,12-)
- `-d`: specifies column separator character (e.g.: ,), the default separator is the tab character

tab separated

```
NAME    AGE CITY
Greta   16  Stockholm
Ahed    18  Nabi-Salih
Atalya  19  Jerusalem
```

comma separated

```
NAME,AGE,CITY
Greta,16,Stockholm
Ahed,18,Nabi-Salih
Atalya,19,Jerusalem
```

? Print the first column of `molbio_2022.txt` to the terminal with `cut`

[2] A flag is an (optional) input or parameter that is passed to a command to extend or modify its functionality. For example, we pass the `-l` flag to `wc` in order to show only the count of lines in a file like so:
`wc -l yourfile`.

Thus far, commands were always entered into the terminal, and the output presented directly (also on the terminal). What if we want to store the output (of a command) in a file?

The **redirection operators** (`>` and `>` `>`), as the name suggests, route the Standard Output (stdout) [3] of a command to a location of the user's choosing.

There are two types of redirections at your disposal:

- > creates and/or overwrites(!) the file
- >> appends to the end of the file

? From the file `molbio_2022.txt` print the country of origin to a file called `nationalities.txt`

We also only entered a single command at a time. But what if we need to perform some other actions on this output using other Bash commands?

The **pipe operator** (`|`) passes the output of a command as input to another command.

? What do these commands do? Guess the function of `uniq` and `sort`.

```
uniq nationalities.txt
sort nationalities.txt | uniq
```

? What do these commands do? Can you find out from the `man`-page what these flags mean: `-l`, `-c`, `-nrk1`?

```
sort nationalities.txt | uniq | wc -l
sort nationalities.txt | uniq -c
sort nationalities.txt | uniq -c | sort -nrk1
```

What if we want to extract certain information from the text file?

grep finds and prints all the lines that match a specific pattern or string in the file(s):

- `-c`: counts occurrences of the pattern
- `-v`: print only the lines that DO NOT contain the pattern
- `-i`: case insensitive flag

[3] The standard output is default place where the Bash command presents its output.

? Try the following command. What does it do?

```
grep "China" molbio_2022.txt
```

? Count the number of students from *India*.

? Count the number of international students (not from *Germany*).

? How many people contain the substring `an` in their names?

- `-E`: let's you use *regular expressions* [4]

? What does this command do?

```
grep -E "^\w{5}\s" molbio_2022.txt
```

```
'^'  : the beginning of a line
'\w' : any word character(alphanumeric & underscore)
'{5}': exact nº of occurrences of last element
'\s' : any white space character
```

1.4 Programming in Bash (Advanced)

A Bash script is a plain text file which contains a series of commands. Bash programming is useful as it allows you to automate tasks (e.g., manipulating files and executing processes). In the MMseqs2 software suite, we also use Bash scripts to combine its modules and workflows, to create tailored computational tools.

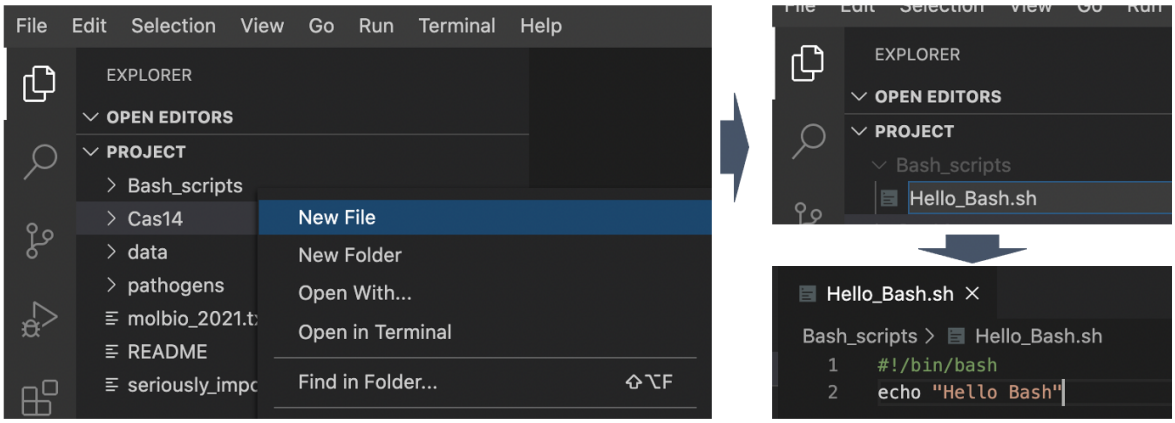
1.4.1 The script file

Now, let's try and print something to the terminal using a self-written Bash script.

Under your home directory, create a new directory called `Bash_scripts`. We will create our Bash scripts here.

Create a new file and rename your file as `Hello_Bash.sh`, similar to the following image. This will be the file where we will enter our Bash commands.

[4] A regular expression is a pattern of meta-characters that is used to describe one or more strings of interest. For instance, think about how you would generically describe to someone–verbally–the way the date is written here: 20-04-2020. It would probably be something along the lines of "day hyphen month hyphen year", or to be more precise "zero-leading-day hyphen zero-leading-month hyphen four-digit-year". The programmatic equivalent `[0-9]{2}-[0-9]{2}-[0-9]{4}` would be one possible regular expression.


File Edit Selection View Go Run Terminal Help
EXPLORER
OPEN EDITORS
PROJECT
Bash_scripts
Cas14
data
pathogens
molbio_2021.t
README
seriously_impo
New File
New Folder
Open With...
Open in Terminal
Find in Folder...
EXPLORER
OPEN EDITORS
PROJECT
Bash_scripts
Hello_Bash.sh
Hello_Bash.sh
Bash_scripts > Hello_Bash.sh
1 #!/bin/bash
2 echo "Hello Bash"

The first line of a Bash script is usually:

```
#!/bin/bash
```

This indicates this file is a Bash script [5]. Add this as the first line in the script.

Our Bash script here will contain a single command that will print "Hello Bash" to the terminal. The command for that is illustrated below. Go ahead and add this command to your script, and then save it.

```
# to print into the terminal
echo "Hello Bash"
```

Now the script can be executed. Almost.

To run your Bash script, you first need to give your script permission to execute:

```
chmod +x ~/project/Bash_scripts/Hello_Bash.sh
```

Now you can run it from the terminal.

Bash Tip 5: ~ means your **home directory**. Try the following:

```
echo $HOME
echo ~
cd ~
```

? Create a Hello_Bash.sh script and run it.

Hint: to run your Bash script, you can run either using the path based on your home directory:

```
~/project/Bash_scripts/Hello_Bash.sh
```

or first `cd` to the directory where the script is, and run it:

```
cd ~/project/Bash_scripts
./Hello_Bash.sh
```

1.4.2 Bash variables

Like any other programming language, Bash also provides variables to store values. There are no variable types in Bash. A variable in Bash can contain a number, a character, or a string of characters.

The assignment of a value to a variable is done by = ; note there should be no space around the = sign in variable assignment.

[5] Note: the `#!/bin/bash` sequence is called a **shebang** and is not an ordinary comment. By convention, every script that gets executed, first gets checked for a shebang. If one exists, the script is executed through the program mentioned in it (here: `/bin/bash`). Refer to this Stack Overflow discussion (`https://stackoverflow.com/q/3009192` and links therein) for more details regarding shebangs.

Then the value of this variable can be retrieved by putting a `$` before the variable name.

```
#!/bin/bash
NAME="Yazhi"
AGE=10
echo "Hello $NAME, you are $AGE old"
```

? Modify the Hello_Bash.sh script you created earlier to include a variable, and re-run it.

1.4.3 Conditional execution

`If` statements allow us to make decisions in our Bash scripts, and to execute commands only in certain cases.

```
AGE=20
if [ "$AGE" -eq 20 ]; then
    echo "Wow, you are exactly 20!"
fi
```

Anything between **then** and **fi** (**if** spelled backwards) will be executed only if the test condition (between the square brackets) is true. Some commonly used conditional operators are listed here:

Description	Numeric	String
less than	-lt	<
greater than	-gt	>
equal	-eq	=
not equal	-ne	!=
less or equal	-le	
greater or equal	-ge	

1.4.4 User Input

User can give input to bash script in terminal using **read** command.

```
echo "enter your name"
read NAME
echo "Hi" $NAME
```

? Edit Hello_bash script.sh to get input from user and month of birth with variable `NAME` and `MONTH_OF_BIRTH`. Apply a condition on month and serve an additional cake if the `MONTH_OF_BIRTH` is 11 or November.

Hint: to read more than one input, use `read NAME and MONTH_OF_BIRTH`

Bash Tip 6: There are many, many more features to Bash! Check out this resource to learn more: `https://ryanstutorials.net/linuxtutorial`

https://linuxconfig.org/bash-scripting-tutorial-for-beginners
https://towardsdatascience.com/basics-of-bash-for-beginners-92e53a4c117a

1.5 File formats

Biological information is conventionally stored in specific textual formats. The contents of such files are arranged in such a way that each unique kind of data within the file(s) is indicated clearly and unambiguously[6]. For example, there are file formats that store the name and polypeptide sequence of proteins. The data is demarcated in such a way that the name string can be disambiguated from the sequence string. This way bioinformatic tools can extract the needed information from the files efficiently, without confusion and/or mistakes.

One of the most common bioinformatics file formats is called **FASTA**. FASTA-formatted files are typically identified by the filename extensions `.fa` or `.fasta` (e.g., myproteins.fasta). In the FASTA format, an identifier (a protein name, for example) is written after the ">" symbol, and its corresponding sequence is written in the lines following it. This format is used, for example, to store protein sequences.

```
>sp|P01308|INS_HUMAN Insulin OS=Homo sapiens OX=9606 GN=INS PE=1 SV=1
MALWMRLLPLLALLALWGPDPAAAFVNQHLCGSHLVEALYLVCGERGFFYTPKTRREAED
LQVGQVELGGGPGAGSLQPLALEGSLQKRGIVEQCCTSICSLYQLENYCN
```

Another popular bioinformatics file format is the **TSV** (tab separated values) format. TSV-formatted usually files have the extension `.tsv` after the filename (e.g., mysamples.tsv). TSV files contain one record per line, with the contents of each line itself being separated by TAB characters. This file format is commonly used to represent tabular data in bioinformatics (e.g., a set of samples, species identities for each sample, and the rRNA sequence of each sample). TSV files are very popular as they are easy to explore with standard Linux tools (and most bioinformatics tools themselves are often Linux-based). This is a file format you will be working with later in the tutorial.

We will present examples of both FASTA files and TSV files later in the tutorial.

[6] uhm, yeah right

Metagenomic pathogen detection

2.1 The Patient

A 61-year-old man was admitted in December 2016 with bilateral headache, gait instability, lethargy, and confusion. Because of multiple tick bites in the preceding 2 weeks, he was prescribed the antibiotic doxycycline for presumed Lyme disease. Over the next 48 hours, he developed worsening confusion, weakness, and ataxia. He returned to the referring hospital and was admitted. He lived in a heavily wooded area in New Hampshire, had frequent tick exposures, and worked as a construction contractor in basements with uncertain rodent and bat exposures. His symptoms were diagnosed as Encephalitis and the causative agent — not known.

? Your task will be to identify the pathogenic root cause of the disease.

This pathogen is usually confirmed by a screening antibody test, followed by a plaque reduction neutralization test. However, this takes 5 weeks, which was too slow to affect the patient's care. As traditional tests done in the first week of the patient's hospital stay did not reveal any conclusive disease cause, the doctors were running out of options. Therefore a novel metagenomic analysis was performed.

2.1.1 The Dataset

Metagenomic sequencing from cerebrospinal fluid was performed on hospital day 8. It returned 14 million short nucleotide sequences (reads).

The authors of the study removed all human reads using Kraken [1] and released a much smaller set of 226,908 reads on the SRA (`https://trace.ncbi.nlm.nih.gov/Traces/sra/sra.cgi`). Kraken extracts short nucleotide subsequences of length k, also called k-mers, and compares them to a reference database where k-mers point to taxonomic labels. In case of exact matching it is able to assign taxonomy.

? Why didn't the authors release the complete dataset of the patient?

? What is the SRA? How many bases are currently publicly available on the SRA in total?

2.2 Metagenomic pathogen detection using MMseqs2

We will use the sequence search tool MMseqs2 [2] to find the cause of this patient's disease. MMseqs2 translates the nucleotide reads to putative protein fragments, searches against a protein reference database and assigns taxonomic labels based on the found reference database hits.

? Why might a protein-protein search be useful for finding bacterial or viral pathogens? How does this compare with Kraken's approach?

2.2.1 Assigning taxonomic labels

We already placed a FASTA file at `pathogens/reads.fasta` containing the reads.[1]

First, change to the exercise directory: `cd pathogens`. Here you will see the previously mentioned `reads.fasta` file and a couple of files starting with `uniprot_sprot`. This contains all the reference proteins from Swiss-Prot which is the manually curated, high-quality part of the Uniprot[4] protein reference database. We are using this smaller subset of about 500,000 proteins, since the full Uniprot with over 175,000,000 sequences requires too many computational resources. Each protein in Swiss-Prot has a taxonomic label. Through a similarity search we will transfer the annotation of the reference protein to our unknown reads. That would be done with the command "taxonomy". Before running this command, we have to convert the fasta file containing the reads to a MMseqs2 database with `createdb`.

```
mmseqs createdb reads.fasta reads
mmseqs taxonomy reads uniprot_sprot lca_result tmp -s 2
```

MMseqs2 will create a result database in your current working directory. This database consists of files, whose names start with `lca_result`. We can convert this database into a human readable tab separated values file (TSV), a common format in bioinformatics.

? Using help for the following command (`mmseqs command -h`), replace "<>" with the required arguments in the command:

```
mmseqs createtsv <> <> <>
```

```
mmseqs createtsv reads lca_result lca.tsv
```

In this file you see for every read a numeric taxonomic identifier, a taxonomic rank and a taxonomic label. However, due to the large number of reads, it is hard to gain insight by skimming the file. MMseqs2 offers a module to summarize the data into a single file `report.txt`:

```
mmseqs taxonomyreport uniprot_sprot lca_result report.txt
```

[1] The sequencing machine returns paired-end reads where sequencing starts in opposite directions from two close-by points to cover the same genomic region. Some of these paired reads overlap enough to be merged into a single read with FLASH [3].

In this file you see a summarized view of the data with the following columns: (1) the percent of reads covered by the clade rooted at this taxon, (2) number of reads covered by the clade rooted at this taxon, (3) number of reads assigned directly to this taxon, (4) rank, (5) taxonomy identifier, and (6) scientific name.

? Based on `report.txt`, what is the most common species in this dataset?

? Why are there so many different eukaryotic sequences?

2.2.2 Visualizing taxonomic results

MMseqs2 can also generate an interactive visualization of the data using Krona [5]. This offers an interactive circular visualization where you can click on each label to zoom into different parts of the hierarchy.

Adapt the previous call to generate a Krona report:

```
mmseqs taxonomyreport uniprot_sprot lca_result report.html --report-mode 1
```

This generates a `HTML` file that can be opened in a browser. Since your editor only display the content of the HTML file and not render it. You have to first navigate to it. Open the URL `https://tutorialNN.mmseqs.com/web` in a new tab. There you will see your `report.html` file. (Don't forget to replace the NN with the number assigned to you.)

2.2.3 What is the pathogen?

Look up the following encephalitis causing agents in Wikipedia.

1. Borrelia bacterium
2. Herpes simplex virus
3. Powassan virus
4. West Nile virus
5. Mycoplasma
6. Angiostrongylus cantonensis

? Based on the literature, which one is the most likely pathogen?

? For which species do you find evidence in the metagenomic reads?

? Approximately how many reads belong to the pathogen?

? Based on this number, how would you determine if it is significant evidence for an actual presence of this agent?

2.3 Investigating the pathogen

We now want to take a closer look only at the reads of the pathogen. To filter the result database, we will need the pathogen's numeric taxonomic identifier. Use the NCBI Taxonomy Browser to find it, by searching for its name:
`https://www.ncbi.nlm.nih.gov/Taxonomy/Browser/wwwtax.cgi`.

? What is the taxon identifier of the pathogen? Did you find one or more?

Now we can call a different MMseqs2 module to retrieve only the reads that belong to this pathogen. Replace **XXX** with the taxonomic identifier(s) you just found. If you found multiple identifiers, concatenate them with a comma `,` character.

```
mmseqs filtertaxdb uniprot_sprot lca_result lca_only_pathogen --taxon-list XXX
```

We now get a list of all queries (i.e., reads) that were **filtered out**, meaning they were annotated as pathogenic.

With a few more commands we can convert our taxonomic labels back into a FASTA file:

```
grep -Pv '\t1$' lca_only_pathogen.index > pathogenic_read_ids
```

```
mmseqs createsubdb pathogenic_read_ids reads reads_pathogen
```

```
mmseqs convert2fasta reads_pathogen reads_pathogen.fasta
```

? How many reads of the pathogen are in this resulting FASTA file?

2.4 Assembling reads into proteins

We want to try to recover the protein sequences of the pathogen.

? Which proteins do you expect to find in the pathogen you discovered? Search the internet.

We will use the protein assembly tool Plass [6] to find overlapping reads and generate whole proteins out of the best matching ones.

```
plass assemble reads_pathogen.fasta pathogen_assembly.fasta tmp
```

Take a look at the generated FASTA file `pathogen_assembly.fasta`.

? How many sequences were assembled?

? Do some of the sequences look similar to each other?

2.5 Clustering to find representative proteins

Plass will uncover a lot of variation in the reads and output many similar proteins. We can use the sequence clustering module in MMseqs2 to get only representative sequences.

? Using help for the following command (`mmseqs command -h`), replace "<>" with the required arguments in the command:

```
mmseqs easy-cluster <> <> <>
```

```
mmseqs easy-cluster pathogen_assembly.fasta assembly_clustered tmp
```

You will see three files starting with `assembly_clustered`:

1. `assembly_clustered_all_seqs.fasta`
2. `assembly_clustered_cluster.tsv`
3. `assembly_clustered_rep_seq.fasta`

Take a look at the last one `assembly_clustered_rep_seq.fasta`. This file contains all representative sequences, meaning the sequence that the algorithm chose as the most representative within this cluster.

? How many sequences remain now? How well does this agree with what you expected according to your internet search?

2.6 Annotating the proteins

Proteins are generally comprised of one or more functional regions, called **domains**. Identifying the domains in a protein provides insights to the function of the protein. We will look for known protein domains to identify the proteins we found.

For this, we will use MMseqs2 search module to search all the representative sequences contained in `assembly_clustered_rep_seq.fasta` against the Pfam database. The Pfam database is a large collection of protein domain families. Each family is represented by multiple sequence alignments (MSAs). The Pfam MSA database was downloaded, and the MSAs have been converted to sequence profile database with MMseqs2. The Pfam profile database is stored as `pfamAfull` in the `pathogens` directory.

```
mmseqs easy-search assembly_clustered_rep_seq.fasta pfamAfull pfam_result.html tmp
↪  --format-mode 3
```

The search results are generated as a `HTML` file that can be opened in a browser. Download the `pfam_result.html` from the URL `https://tutorialNN.mmseqs.com/web` in a new tab. (Don't forget to replace the NN with the number assigned to you.) Open `pfam_result.html`. You can navigate through the representative protein sequences to find out about the matched PFAM domains and visualize how they are aligned with the query proteins.

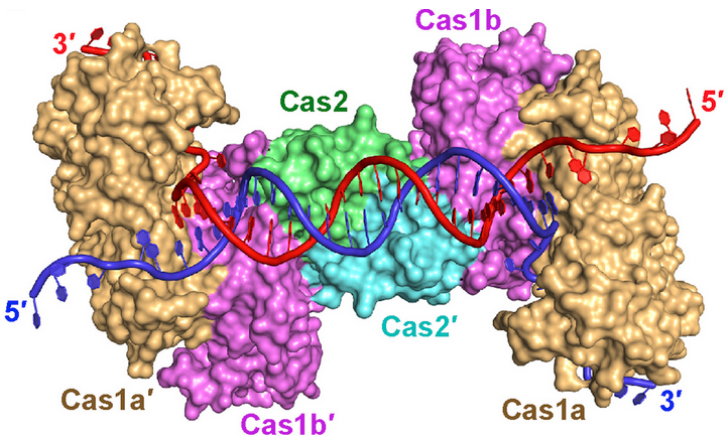
? Look up some of the PFAM domain entries that were matched. Which of the expected protein (domains) do you find?

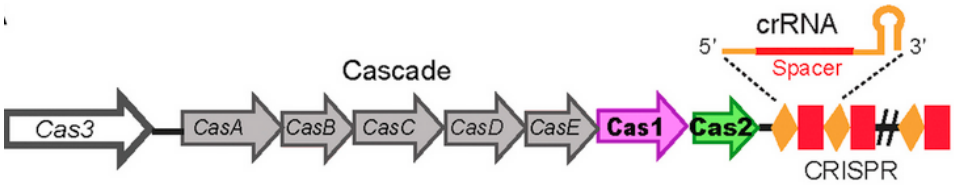
2.7 Aftermath

Despite being able to identify the causative agent, the pathogen is very hard to treat. The patient had minimal neurological recovery and was discharged to an acute care facility on hospital day 30. Seven months after discharge, he was reportedly able to nod his head to questions and slightly move his upper extremities and toes.

You can find the publication about this patient and dataset here [7].
Please look at it only after trying to answer the questions yourself.

Protein structure prediction





In this section you will learn how to:

1. Predict the 3-D structure of a protein (Cas1) with ColabFold
2. Search for protein structures on the websites UniProt[4] and RCSB PDB[8]
3. Use visualization tools to explore protein structures

Have fun!

3.1 Prediction of Cas1 protein structures using ColabFold

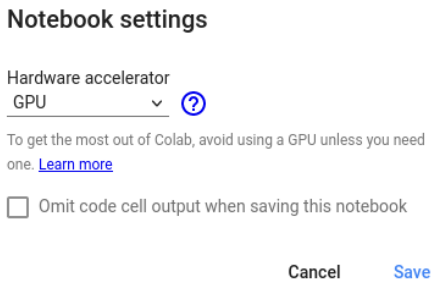
Cas1: CRISPR-associated protein 1 (Cas1) is a widely conserved component of the CRISPR adaptive immune system. It functions as a metal-dependent, DNA-specific endonuclease. It forms a complex with Cas2 to integrate phage DNA into the CRISPR array of the host (bacterial) genome. In this tutorial, we will work with Cas1 from *E. coli* strain K12.

ColabFold:

ColabFold is an easy-to-use, Google Colab-based implementation of the AlphaFold2 structure prediction suite. ColabFold [9] makes use of both to offer a simple, user friendly and fast tool to predict 3-D structures of proteins. AlphaFold2 predicts protein 3-D structures based on MSAs. Google Colab offers free CPU and, importantly, free GPU resources for running Jupyter Notebooks.

Tips for Colab:
- You can show/hide the code with **View** → **Show/hide code**, or click on the ▷ button left from the code cell.

1. Open the ColabFold Notebook[1] in Google Colab and sign in with your Google account. The usage of Google Colab is free, but requires a Google account.
2. A GPU is required for the structure prediction, therefore configure the notebook to use a GPU: **Runtime** → **Change runtime type**

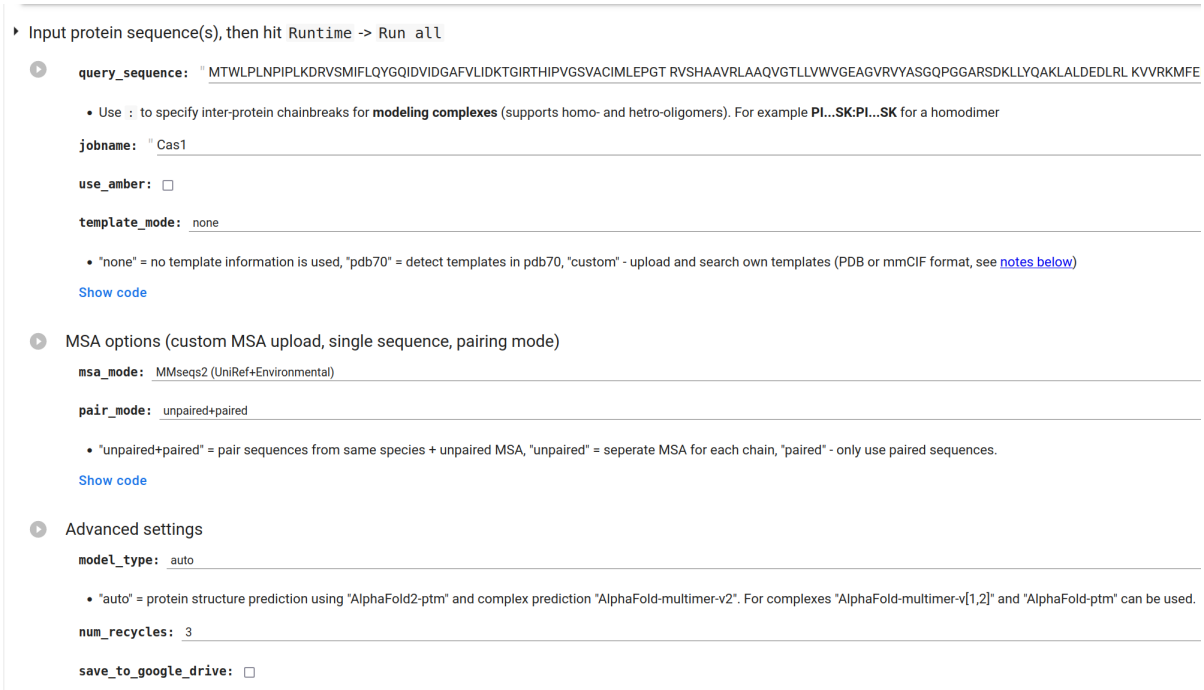


[1] https://colab.research.google.com/github/sokrypton/ColabFold/blob/main/AlphaFold2.ipynb

3. First enter the amino acid sequence of the protein into the field `query_sequence`. Then select the `template_mode`("none").

   Then select `msa_mode` as MMseqs(UniRef+Environmental). By default, AlphaFold2 predicts five different structures and we can choose the best model afterwards. You can give any jobname as you prefer. We used "Cas1" here.

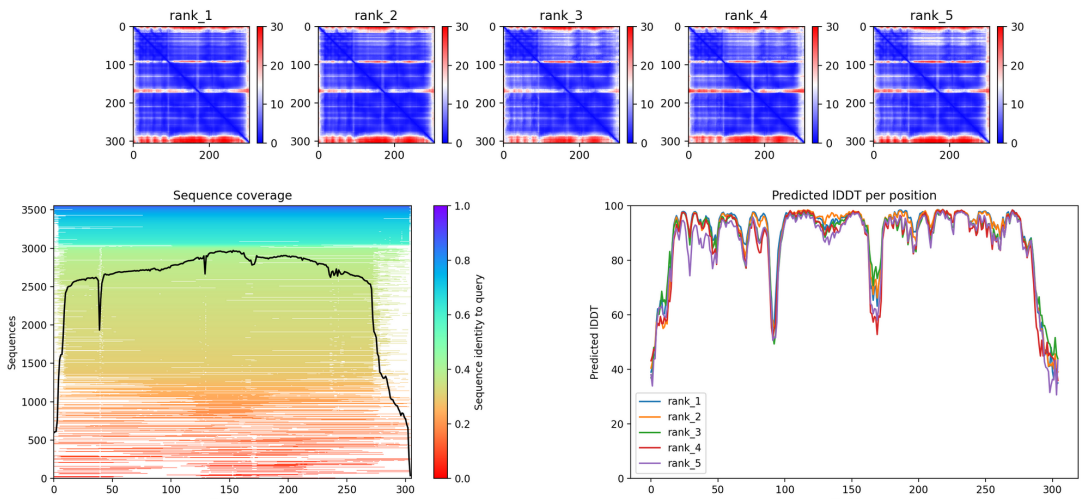
```
>sp|Q46896|CAS1_ECOLI CRISPR-associated endonuclease Cas1
MTWLPLNPIPLKDRVSMIFLQYGQIDVIDGAFVLIDKTGIRTHIPVGSVACIMLEPGTRVSHAAVRLA
AQVGTLLVWVGEAGVRVYASGQPGGARSDKLLYQAKLALDEDLRLKVVRKMFELRFGEPAPARRSVEQ
LRGIEGSRVRATYALLAKQYGVTWNGRRYDPKDWEKGDTINQCISAATSCLYGVTEAAILAAGYAPAI
GFVHTGKPLSFVYDIADIIKFDTVVPKAFEIARRNPGEPDREVRLACRDIFRSSKTLAKLIPLIEDVL
AAGEIQPPAPPEDAQPVAIPLPVSLGDAGHRSS
```



4. To start the prediction hit **Runtime** → **Run all** (This will take some minutes...)

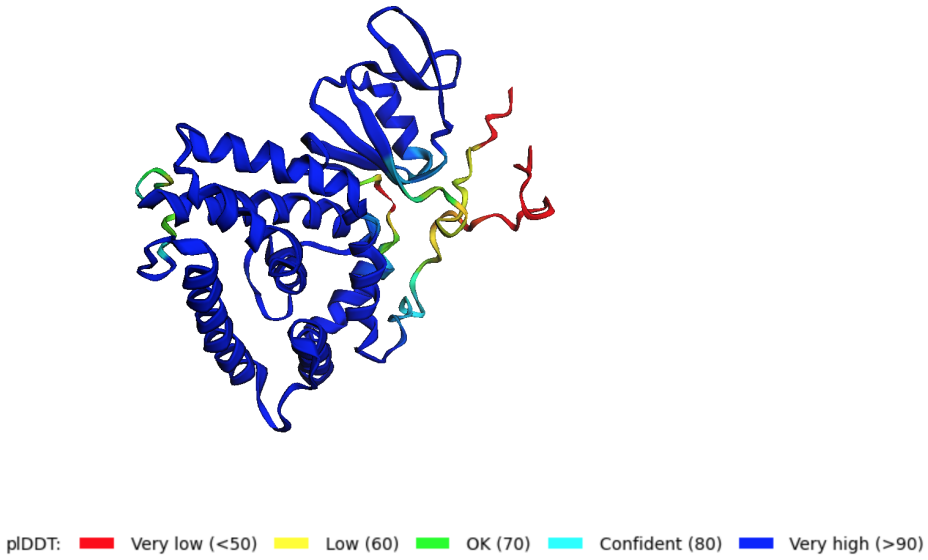
5. The prediction results can be visualized with the plots below. The five predicted models are ranked by confidence from high (rank 1) to low (rank 5).

   ? How confident is AlphaFold2 in its prediction and how good is the input MSA? Interpret the prediction quality by checking the plots (lDDT = local Distance Difference Test).

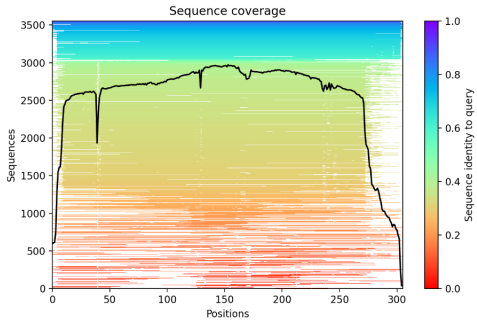


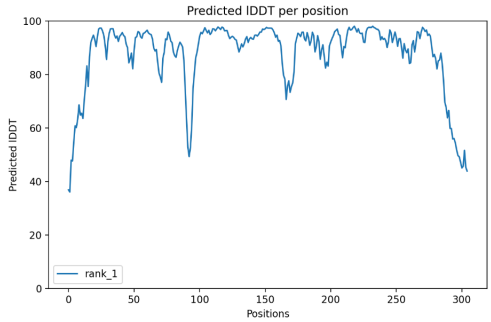
6. Check the predicted Cas1 3-D structure (rank 1). Have fun playing with the cartoon view (ribbon representation).

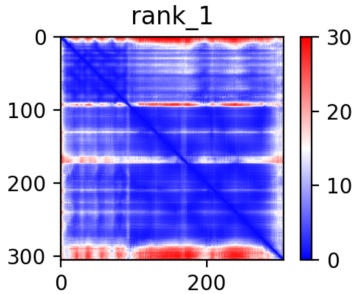
Show code



7. Take a closer look at the confidence and quality measures of the rank 1 model.





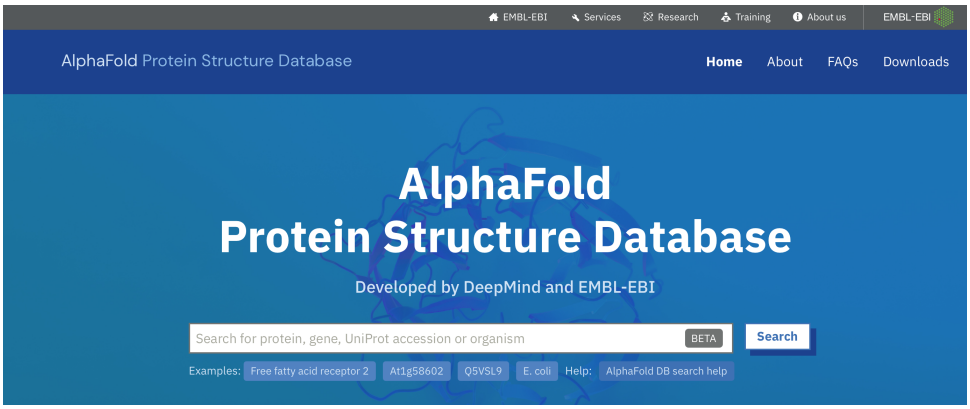


8. You can download the resulting structures as PDB files.

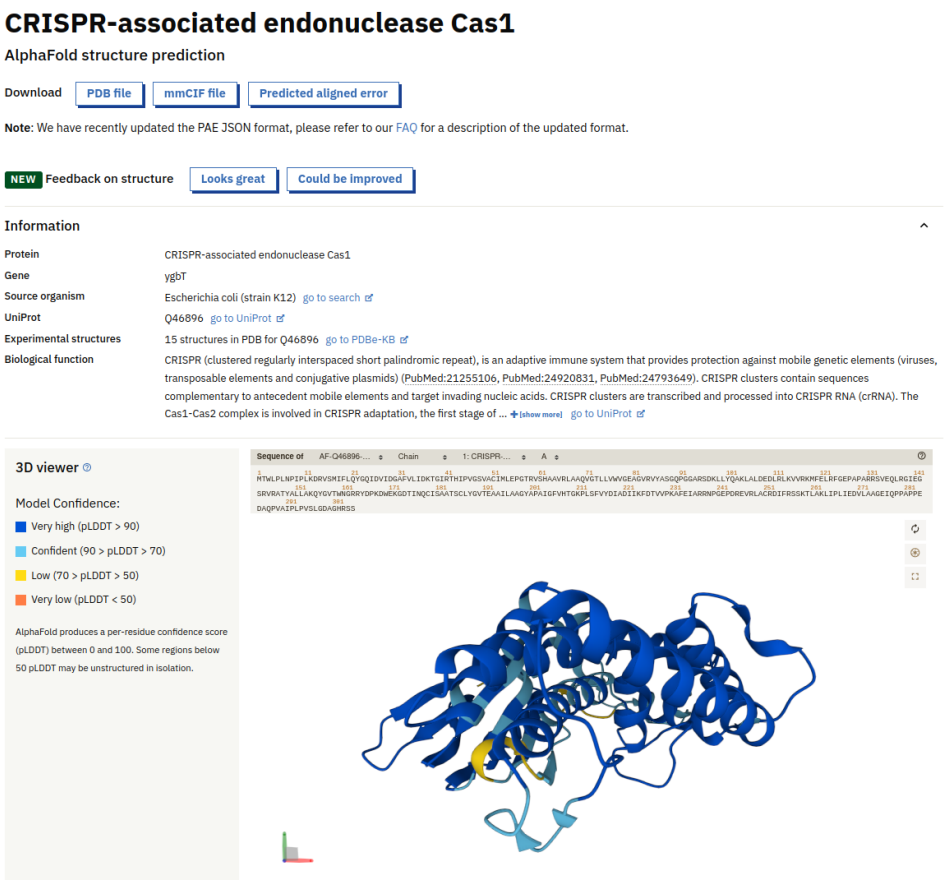
   Note: Further instructions for how to use ColabFold, descriptions about the results and acknowledgements can be found at the bottom of the Colab page.

3.2 AlphaFold Protein Structure Database

EMBL-EBI and DeepMind have together developed a database for protein structure models predicted by AlphaFold (`https://alphafold.ebi.ac.uk`). Currently, it has the 3-D models for the complete human proteome and 47 other reference organisms such as *Arabidopsis thaliana*, *Caenorhabditis elegans*, *Danio rerio*, and *Rattus norvegicus*. It also contains predictions for most UniProt sequences, resulting in more than 200 million entries. You can retrieve predicted protein 3-D structures using keywords such as protein name, Gene ID, UniProt ID or species name.



Search for Cas1 protein using UniProt ID **Q46896** in the search box. You will find the details of Gene name, Source Organism, PDBe-KB link (if experimental structure is available). You can also find the predicted model and other information (e.g. biological function) by clicking the first blue entry (CRISPR-associated endonuclease Cas1).

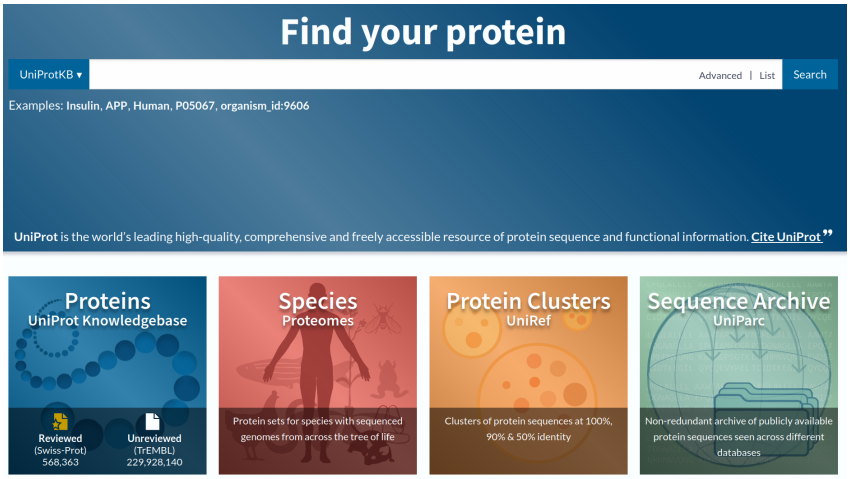


You can also find the models for all proteins in the proteome of the 47 species that they have covered so far.

In the coming months, the database will provide 3-D models for a large proportion of all catalogued proteins in the UniProt.

3.3 Understand more about the protein in UniProt Database

1. **UniProt** is a comprehensive, high-quality and freely accessible resource for protein sequence and functional information. Go to the UniProt website: `https://www.uniprot.org/`.

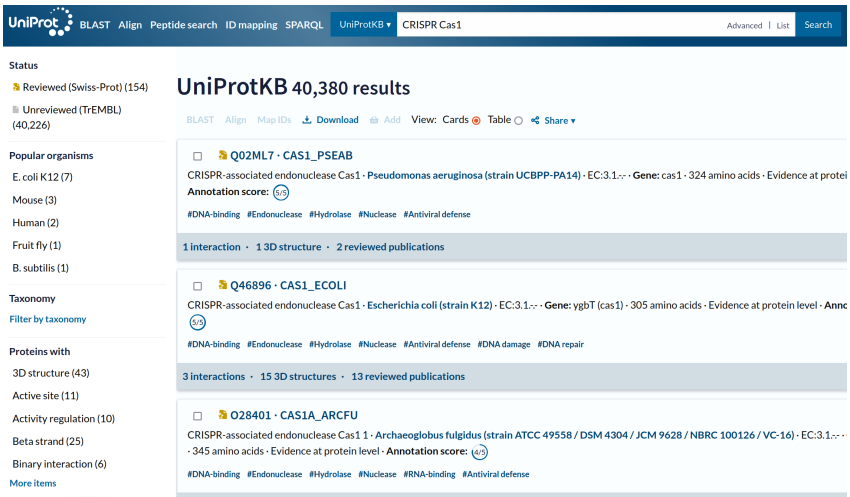


2. Search for **CRISPR Cas1**.

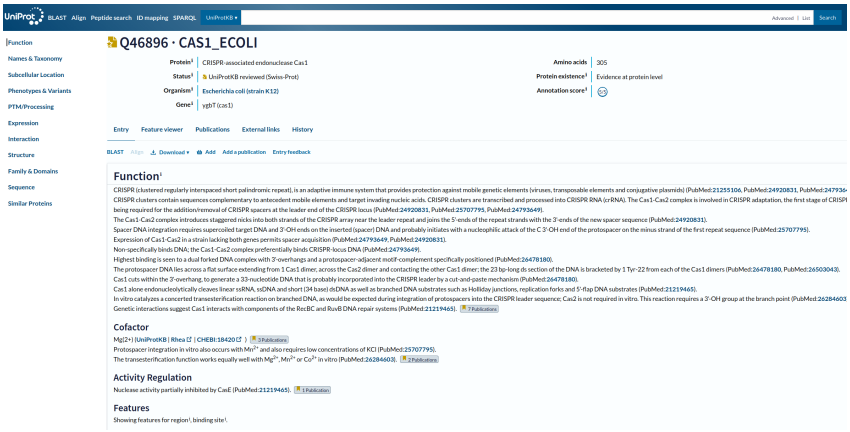
   ? How many entries do you get in the result table? How many of them are manually curated reviewed entries?

(Answer: **40,380; 154**)

3. Select the second entry (**Q46896**) corresponding to *E. coli* (strain K12).



4. Find the functional description about the protein at the left top. Other comprehensive details can be seen by navigating through various sections in the left panel.



What is the sequence length of *E. coli* Cas1 protein? Click on the *Sequence* section in the left panel.

(Answer: 305)

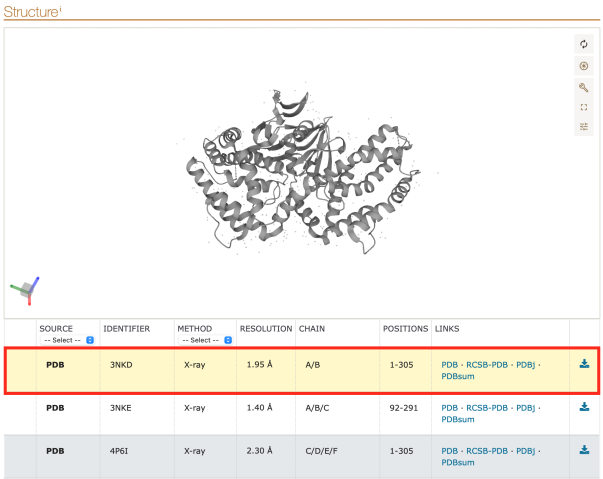
Where is this Cas1 protein expressed inside the *E. coli*? Click on the *Subcellular location* section.

(Answer: Cytoplasm and Cytosol)

Does this protein has a experimentally solved structure? Click on the *Structure* section.

(Answer: Yes)

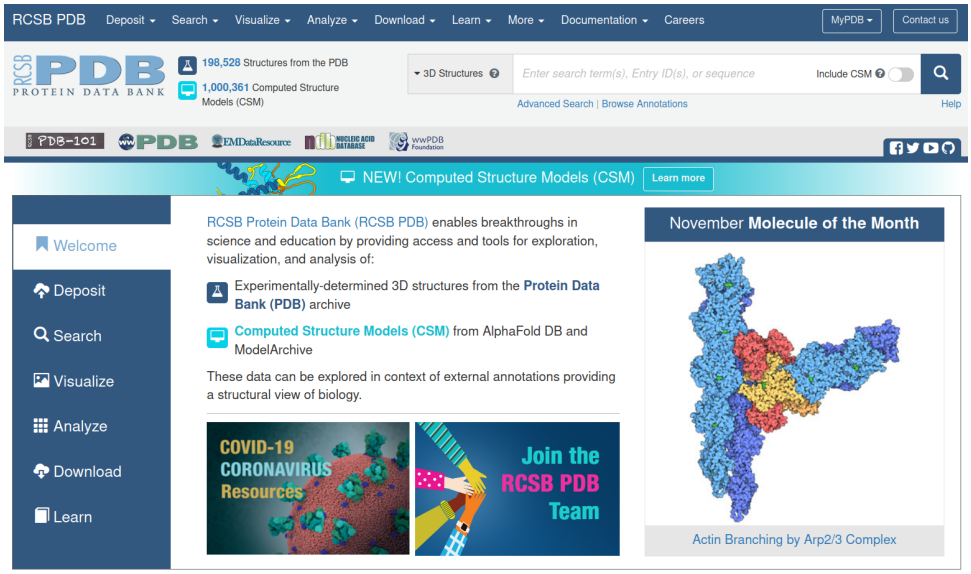
5. As the table shows, the protein has 15 experimentally solved structures and one predicted model from AlphaFold. In this tutorial we will focus on the first PDB entry **3NKD**.



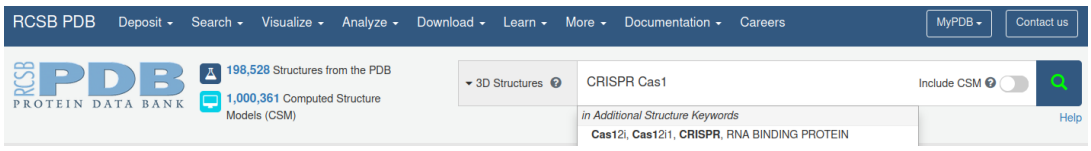
For interested candidates, check out the recently constructed UniProt **beta** version `https://beta.uniprot.org`

3.4 Searching for experimentally solved Cas1 protein structures in the Protein Data Bank (PDB)

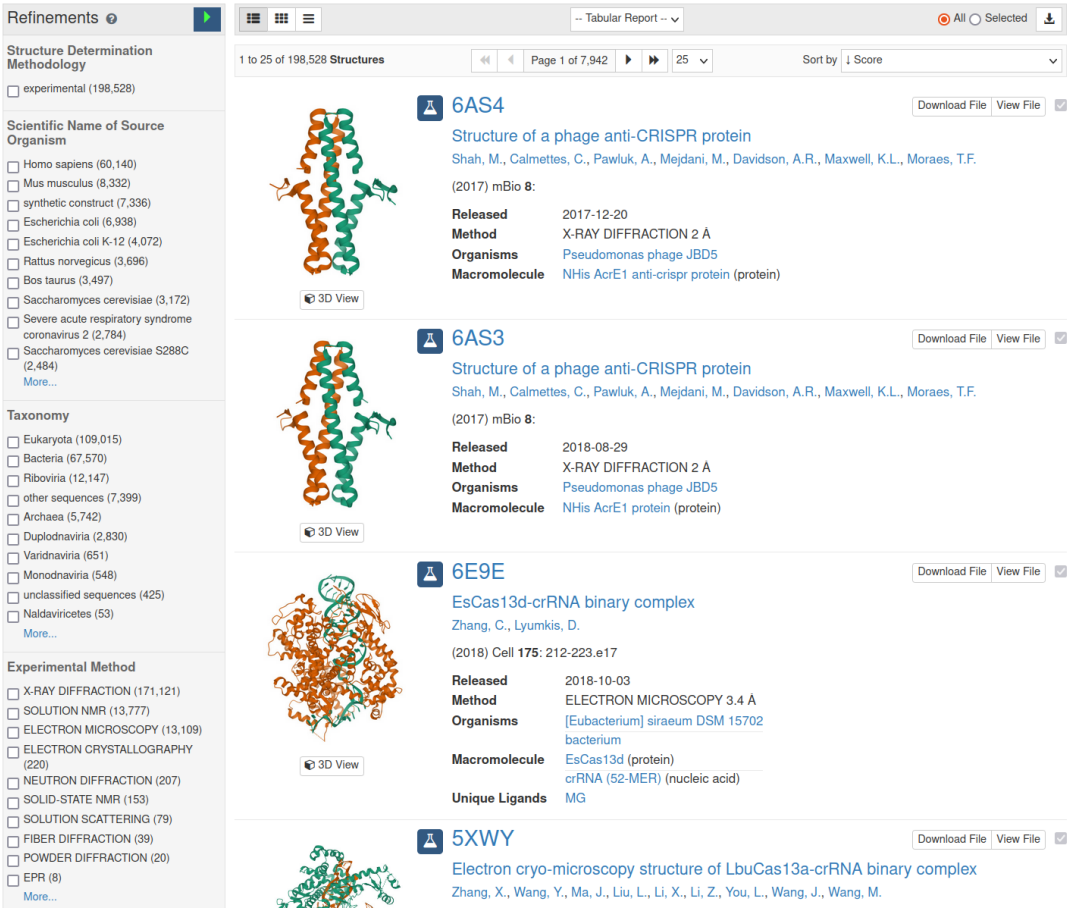
1. **RCSB PDB** is a repository for 3-D macromolecular structures (Proteins, nucleic acids and macromolecular complexes). Go to the RCSB PDB website: `http://www.rcsb.org`



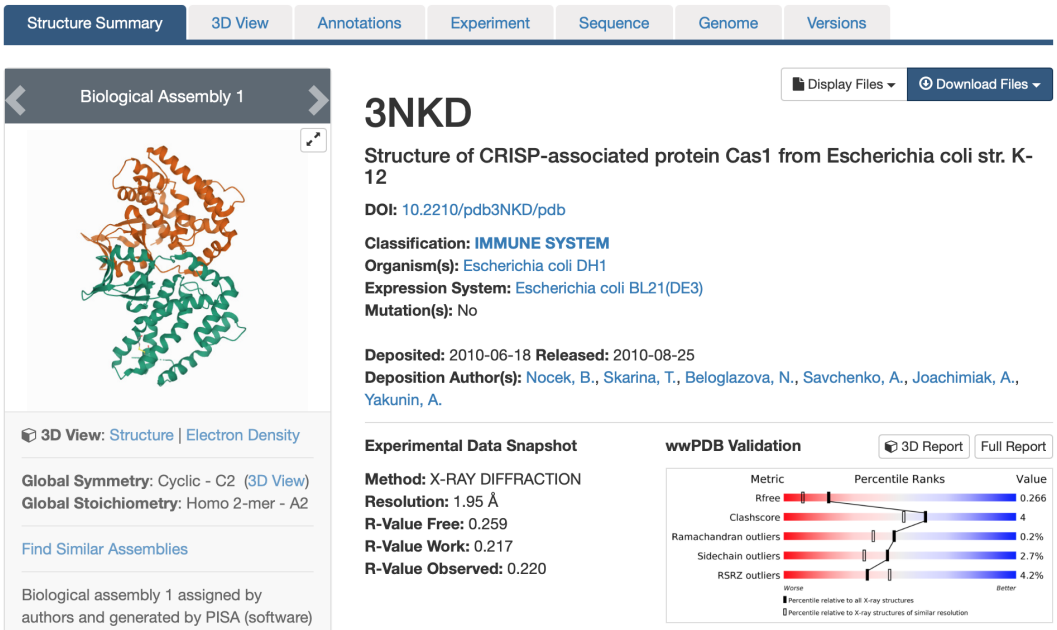
2. Search with the keyword **CRISPR Cas1**.



3. Explore the result page with different **Refinements** options and the summary of the results.



4. You can click on any of the structures and briefly explore its web page.
5. Let's analyze the PDB entry **3NKD** further here.



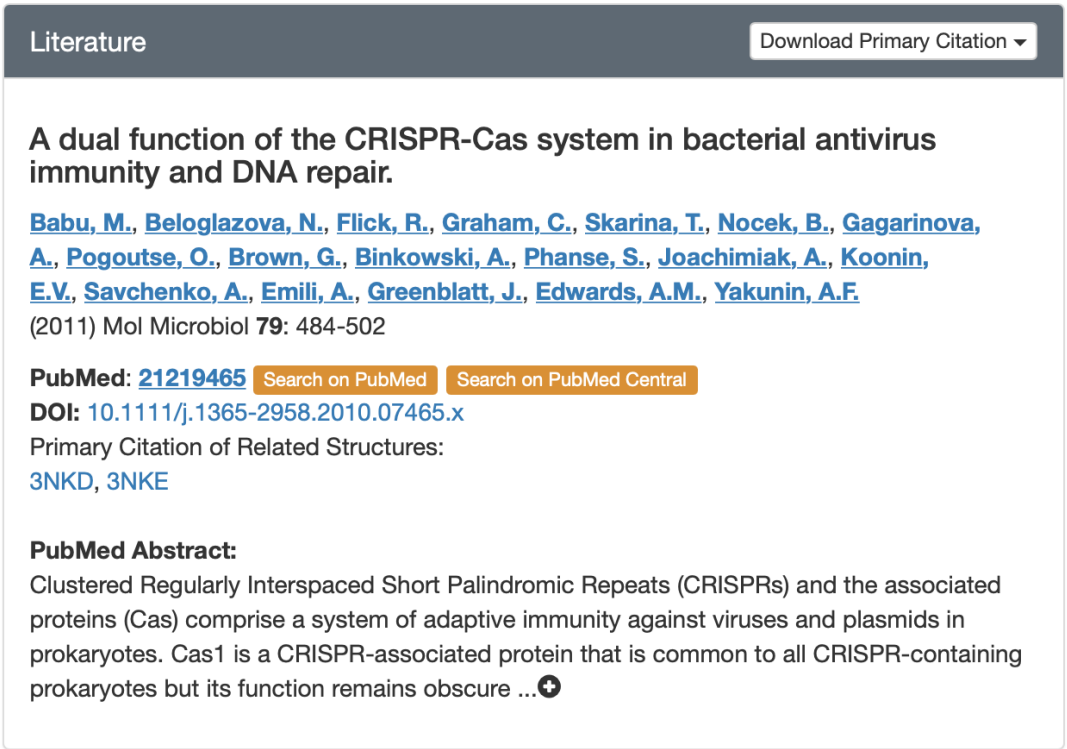
? What is the resolution of the structure?

(Answer: 1.95Å)

? Does this structure belong to a wild-type protein or does it have mutated residues?

(Answer: Wild-type, no mutations)

6. The details of the research article that has published this structure is given in the **Literature** section.



7. Residue-level secondary structural states and sequence annotations (mapped from UniProt) are provided in a graphical representation for an easy interpretation.

Entity ID: 1

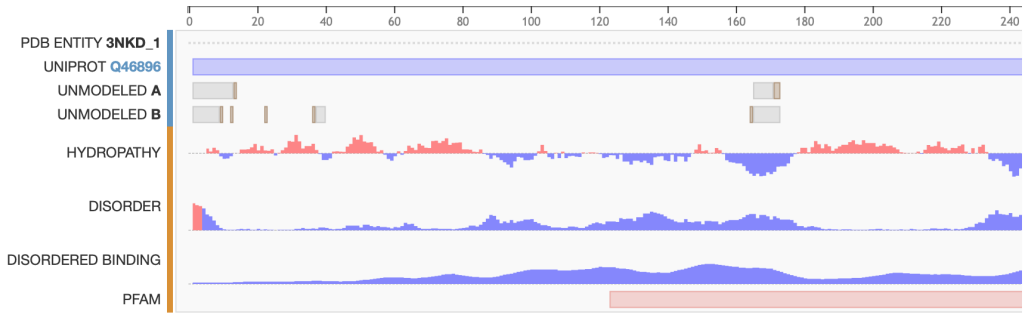
Molecule	Chains	Sequence Length	Organism	Details
CRISPR-associated protein Cas1	A, B	305	Escherichia coli DH1	**Mutation(s):** 0 **Gene Names:** EcDH1_093 3 **EC:** 3.1

UniProt

Find proteins for Q46896 *(Escherichia coli (strain K12))* Explore Q46896

Protein Feature View

Reference Sequence 3NKD_1

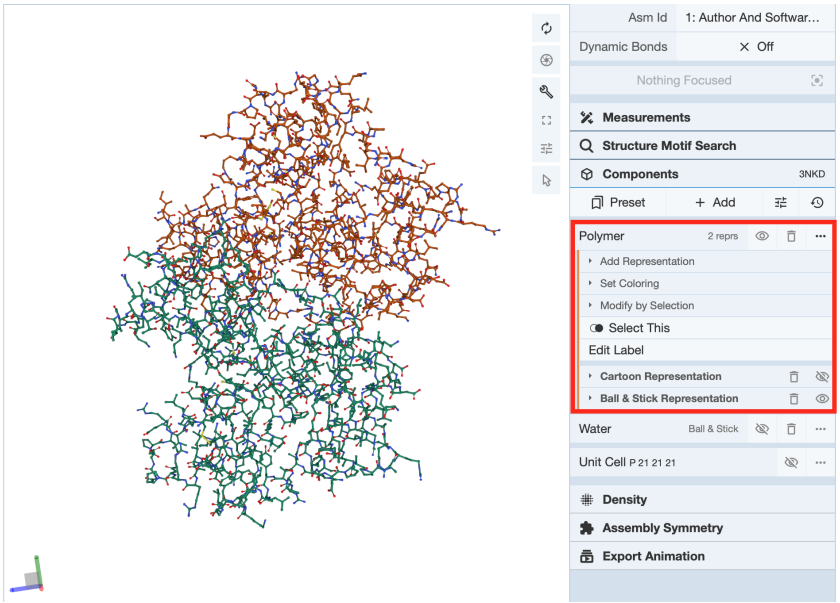


8. Go to **3D view**.

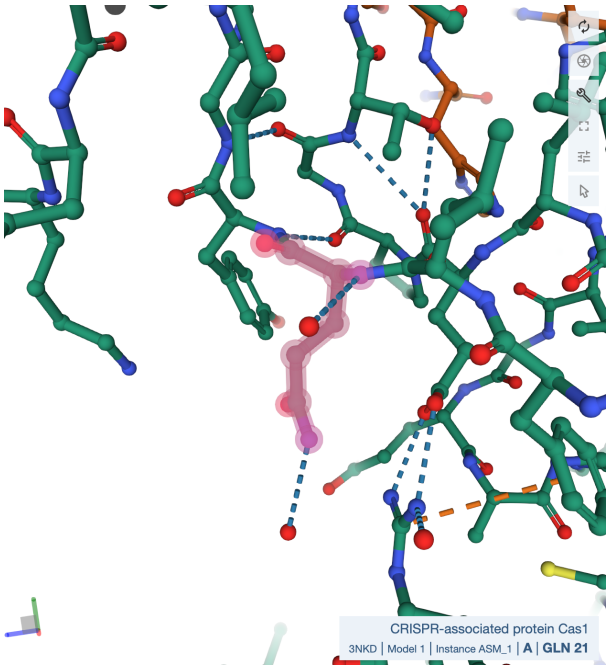
? Why do we see two colors in the cartoon view?

(Answer: It is a homodimer)

9. Have fun with adding different representation in the **Polymer** drop-down menu. Click on the **Add representation** to view multiple representation options. Shown below is the Ball & Stick representation.



10. Select residue **Q21** in the sequence shown at the top panel. The cartoon automatically focuses on the local region around this residue. Interactions between Q21 and other residues are shown by dashed lines.



11. If you want to explore more sophisticated tools for protein structure visualization and analysis, have a look at **Pymol, Chimera(X) or VMD**. They are GUI-based (graphical user interface) tools and offers several options to examine single as well as multiple protein structures.

Protein structure search

In this section you will learn how to:

- Discover similar protein structures with Foldseek.
- Annotate a protein by transferring the functional annotations of the best hits.

4.1 The dataset: A sponge proteome

Sponges are of interest, because they consist only out of (about) 20 cell types, and can provide insights about the early history of animals. The proteome of the freshwater sponge *Spongilla lacustris* was obtained by RNA isolation and sequencing. The subsequent transcriptome assembly resulted in 41 945 protein sequences [10] (published here [1]). The goal is to find functional annotations for these sequences (e.g. the protein functions) to learn more about the sponge.

4.2 Sequence-based annotation (using EggNOG-mapper)

The authors of [10] already performed a sequence-based annotation using EggNOG-mapper, which searches each sequence against a database of annotated sequences and transfers the annotation of the best hits.

How to use EggNOG-mapper: (not part of this task) The EggNOG-mapper web service at eggnog-mapper.embl.de allows uploading even large fasta files. After entering the email address and optionally adjusting search parameters, the job can be submitted. Next, start the job through the link sent to your email address. For the sponge genome, it took about 15 minutes and returned an annotation list for the submitted sequences.

This sequence-based search succeeded for around 40% of the sponge sequences to find an annotation. So, for the rest around 60% of sponge sequences, how to annotate them?

? Why can't the sequence-based methods annotate all sequences?

[1] `https://zenodo.org/record/6821244`

4.3 Structure-based annotation (using Foldseek)

As the protein structure determines its function, and as the structure can also be better conserved than its sequence, the idea is to search with the protein structure instead of its sequence.

? Your task will be to find possible annotations of the protein no. 101753, one of the unannotated proteins, using Foldseek.

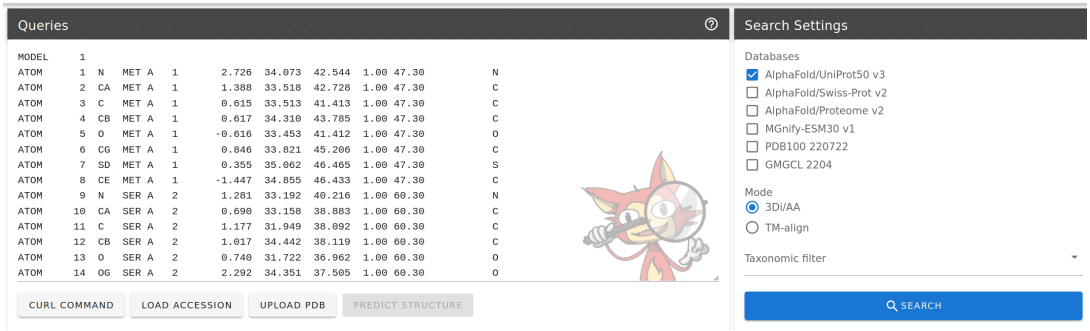
```
>unknown protein no.  101753
VSAARSSQTCTCTQTHAHTKPMLTQGGFGKSLNSLGKVLDSAVKDVDKTVTQAVSTSPLELLKNGYIVQII
SRVGGKCLRVLENGQADCLGDVGTSSQFEVVVPRPGIVKLRNVAMPKYYIAITGGYLIGYGQGGPDCDFVP
CDFVPSMIVGNYVVFESAMSPGGVIGALPSGLISAPMQTQKTCDAAHFGIKYINSVRR
```

Foldseek:

"Foldseek enables fast and sensitive comparisons of large structure sets. It reaches sensitivities similar to state-of-the-art structural aligners while being at least 20,000 times faster. To facilitate access to Foldseek, we developed a user-friendly webserver optimized to quickly return results for single queries." [11]

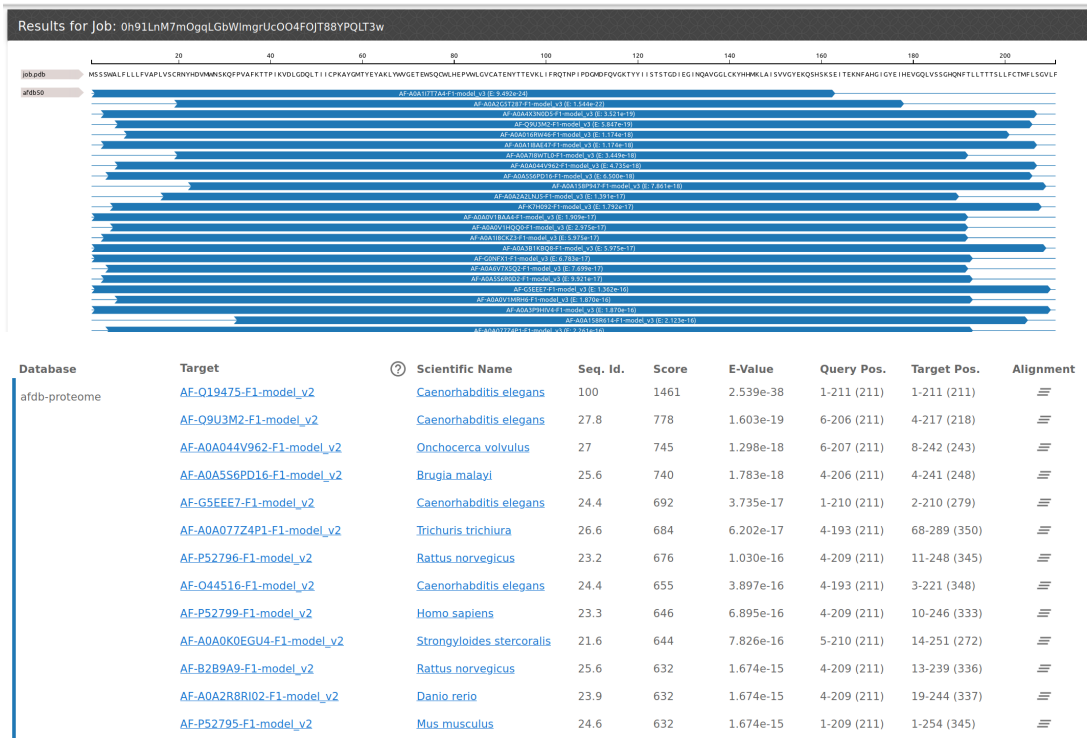
1. Predict the structure of our unknown protein no. 101753 using ColabFold. At the end of the job a download modal box will pop up with a **jobname.result.zip** file. Go to the directory where the **.zip** is located and extract **.zip** file. We can choose the best model **.pdb** file (rank 1) as the query structure.
2. Go to search.foldseek.com and upload your query structure.
3. Before starting the search, you can select on the right side **Search Settings** in which databases to search.

   For now, you can select all databases such as `AlphaFold/UniProt50 v3`, which contains structures predicted from all UniProt sequences clustered by 50 percent sequence identity.



4. Now, hit **SEARCH** to start it (It will only take a few seconds...).

5. The search result page is divided into the alignment visualization and below that (scroll down) a list of hits for each database:



? How many structural hits were found?

6. Click the button, to show the alignment of a hit. Examine the hit through the 3-D viewer, and check if the structures are similar.

? How well do the aligned structures match? Could they be homologous?

7. For searches against the AlphlfoldDB, you can open the AlphaFoldDB entry by clicking on the target name in the **Database**:afdb-proteome.

? What is the annotation of the protein sequence?

Fibroblast growth factor (FGF)

Tips: EggNOG also provides an annotation method using Hidden Markow Models (HMM) at eggnog5.embl.de, which is much more sensitive. For example, it finds an annotation for the *S. lacustris* FGF (try it with the sequence). However, this approach is much slower than EggNOG-mapper, and annotating the entire genome would take several hours.

Appendix

5.1 Some useful Bash commands

```
# show a file inside the terminal (hint: use q to exit again)
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

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

# 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

# get file name you created in previous command
history| grep 'touch'
```

5.2 Letter codes for amino acids in a protein chain

A	Alanine	Ala
C	Cysteine	Cys
D	Aspartic Acid	Asp
E	Glutamic Acid	Glu
F	Phenylalanine	Phe
G	Glycine	Gly
H	Histidine	His
I	Isoleucine	Ile
K	Lysine	Lys
L	Leucine	Leu
M	Methionine	Met
N	Asparagine	Asn
P	Proline	Pro
Q	Glutamine	Gln
R	Arginine	Arg
S	Serine	Ser
T	Threonine	Thr
V	Valine	Val
W	Tryptophan	Trp
Y	Tyrosine	Tyr

5.3 Exercise solutions for section 1.4.4

1. ```
   #!/bin/bash
   echo "Hello Bash"
   ```

2. ```
   #!/bin/bash
   echo "Hello! enter your name and month of birth"
   read NAME MONTH_OF_BIRTH
   if [ $MONTH_OF_BIRTH -eq 11 ] || [ $MONTH_OF_BIRTH = "november" ];
   then
   echo "Hi $NAME, This month is your birth of month "$MONTH_OF_BIRTH". We present
   ↪  you a birthday cake"
   fi
   ```

Bibliography

[1] Derrick E Wood and Steven L Salzberg. Kraken: ultrafast metagenomic sequence classification using exact alignments. *Genome Biol.*, 15(3):R46, 2014.

[2] Martin Steinegger and Johannes Söding. MMseqs2 enables sensitive protein sequence searching for the analysis of massive data sets. *Nat. Biotechnol.*, 35(11):1026–1028, 2017.

[3] Tanja Magoc and Steven L. Salzberg. Flash: fast length adjustment of short reads to improve genome assemblies. *Bioinformatics*, 27(21):2957–2963, 2011.

[4] Alex Bateman. UniProt: a worldwide hub of protein knowledge. *Nucleic Acids Res.*, 47(D1):D506–D515, 2019.

[5] Brian D Ondov, Nicholas H Bergman, and Adam M Phillippy. Interactive metagenomic visualization in a Web browser. *BMC Bioinform.*, 12(1):385, 2011.

[6] Martin Steinegger, Milot Mirdita, and Johannes Söding. Protein-level assembly increases protein sequence recovery from metagenomic samples manyfold. *Nat. Methods*, 16(7):603–606, 2019.

[7] Anne Piantadosi, Sanjat Kanjilal, Vijay Ganesh, Arjun Khanna, Emily P Hyle, Jonathan Rosand, Tyler Bold, Hayden C Metsky, Jacob Lemieux, Michael J Leone, Lisa Freimark, Christian B Matranga, Gordon Adams, Graham McGrath, Siavash Zamirpour, III Telford, Sam, Eric Rosenberg, Tracey Cho, Matthew P Frosch, Marcia B Goldberg, Shibani S Mukerji, and Pardis C Sabeti. Rapid Detection of Powassan Virus in a Patient With Encephalitis by Metagenomic Sequencing. *Clin. Infect. Dis.*, 66(5):789–792, 2017.

[8] Andrei Kouranov. The RCSB PDB information portal for structural genomics. *Nucleic Acids Res.*, 34(D1):D302–D305, 2006.

[9] Milot Mirdita, Konstantin Schütze, Yoshitaka Moriwaki, Lim Heo, Sergey Ovchinnikov, and Martin Steinegger. Colabfold: making protein folding accessible to all. *Nature Methods*, 19(6):679–682, Jun 2022.

[10] Fabian Ruperti, Nikolaos Papadopoulos, Jacob Musser, and Detlev Arendt. Beyond sequence similarity: cross-phyla protein annotation by structural prediction and alignment. *bioRxiv*, 2022.

[11] Michel van Kempen, Stephanie Kim, Charlotte Tumescheit, Milot Mirdita, Johannes Söding, and Martin Steinegger. Foldseek: fast and accurate protein structure search. *bioRxiv*, 2022.