Protein Bioinformatics

Milot Mirdita Venket Raghavan Ruoshi Zhang
Johannes Söding

November 10 – 11, 2020

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. Below you will get familiar with a few basic textual commands in a specific type of Linux Shell, called **Bash**.

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://devNN.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

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

computer (e.g. for uploading it to a webserver) you can open `https://devNN.mmseqs.com/web` and download the files from there.

You should see something like the following image:

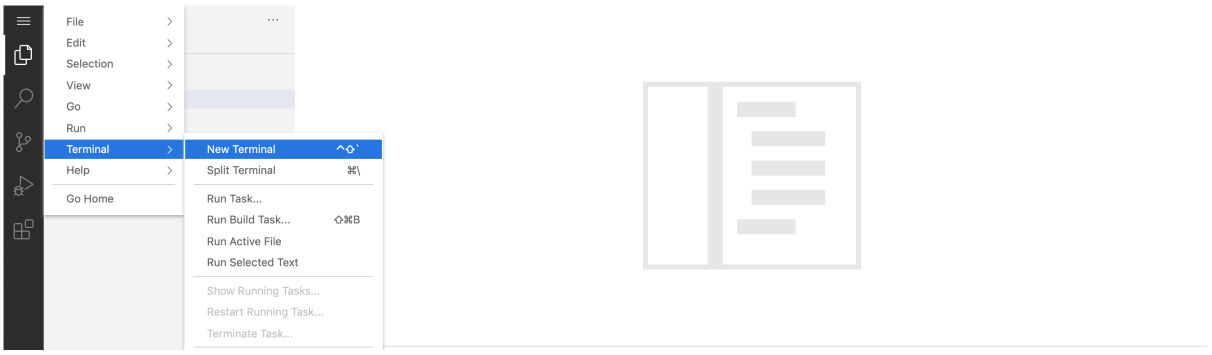


Figure 1.1: You can open a new terminal by clicking on the burger icon (≡) and then navigating to "Terminal -> New Terminal".

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

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`.

```
# open the file in a separate text editor window
code 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:
`cod`TAB 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.

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
```

```
# 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
```

```
# 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
```

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

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

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.

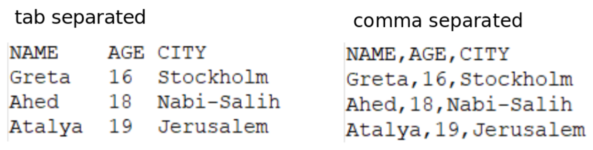
Note these commands usually have various command line options that will modify their behavior. Whenever you are not sure about what a command does or how to run it, you can always look up its manual page with the following command:

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

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



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

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

Thus far, commands were always entered into the terminal, and the output presented directly (also on the terminal). We also only entered a single command at a time. But what if we want to store the output (of a command) in a file, or need to perform some other actions on this output using other Bash commands?

Then we need to redirect the output using `>` (redirection operator) and `|` (pipe operator).

The **redirection operators** (`>` and `>` `>`) redirect the Standard Output (stdout) [3] of a command to a file or elsewhere:

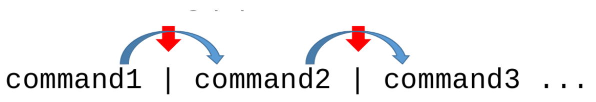
- > creates and/or overwrites(!) the file
- >> appends to the end of the file

[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`.

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

? From the file `molbio_2020.txt` print the country of origin to a file called `nationalities.txt`

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



```
command1 | command2 | command3 ...
```

? What do these commands do?

```
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
```

grep extracts 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

? Count number of students from *India*.

? Count number of students that are not from *Germany*.

? How many people contain the word fragment *an* in their names?

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

? What does this command do?

```
grep -E "^\w{5}\s" molbio_2020.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
```

[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.

1.4 Programming in Bash

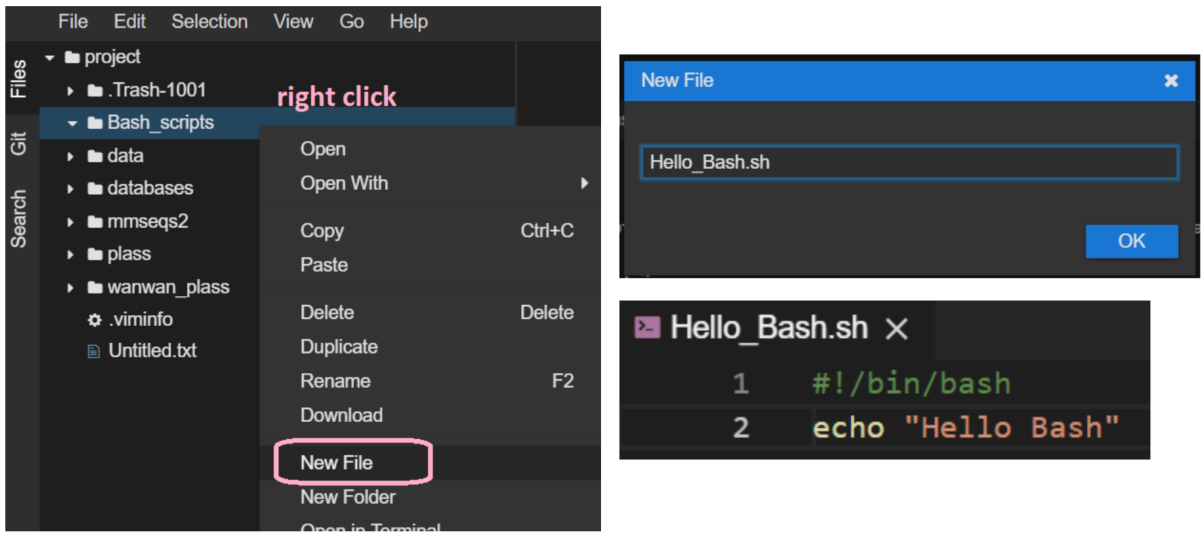
A Bash script is a plain text file which contains a series of commands. Bash programming is useful as it allows you to automate tasks (e.g., manipulating files and executing processes). In the MMseqs2 software suite, we also use Bash scripts to combine its modules and workflows, to create tailored computational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



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

[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.

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.

❢ **Create a Hello_Bash.sh script and run it.**

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

Bash Tip 4: `~` means your **home directory**. Try the following:

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

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.

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

```
#!/bin/bash
NAME="Eli"
NUMBER_OF_EYES=3
echo "Hello $NAME, you have $NUMBER_OF_EYES eyes"
```

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

1.4.3 Arithmetic evaluation

You can perform math operations on (and with) Bash variables. In order for Bash to treat the variable as numeric we need to use parentheses. E.g., if we were to follow up on the small script in sub-section 1.4.2 above, we could do something like this:

```
CORRECT_NUMBER_OF_EYES=$((NUMBER_OF_EYES - 1))
echo "Humans usually don't have more than $CORRECT_NUMBER_OF_EYES eyes"
```

Of course, two (or more) Bash variables can also be used together in arithmetic operations like so:

```
VAR_A=2
VAR_B=4
VAR_C=$((VAR_A + VAR_B))
echo "The sum of $VAR_A and $VAR_B is $VAR_C"
```

? **Create a Bash script with a variable AGE and assign it your current age. Print how old you will be in one year from now.**

1.4.4 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	

1.4.5 User interaction

If we would like to ask the user for input then we use a command called **read**. This command takes the user input and saves it to a variable. For example, this simple Bash script asks the user for their name and greets them personally:

```
#!/bin/bash
echo "Enter your name and press [ENTER]: "
# NAME is the variable here in which the user input obtained by read will be stored
read NAME
echo "Hi $NAME"
```

- **Create a script that asks for the user's age and serves beer only if the user is at least 18.**
- **What does the following code do?**

```
echo "Enter a directory name and press [ENTER]: "
read DIR
#-d True if FILE exists and is a directory.
if [ -d "$DIR" ]; then
    ls "$DIR"
else
    mkdir "$DIR"
fi
```

1.4.6 Repetitive execution of commands

Often we would like to perform the same thing more than once until a particular situation/condition is reached. Bash **for/while** loops allow us to do exactly that. For example, this Bash script says hello to all the students in the class (there are 24 of you!) using a **for** loop:

```
#!/bin/bash
START=1
END=24
for (( i=$START; i<=$END; i++ ))
do
        echo "$i. Hi, student!"
done
```

The **while** loop structure allows for repetitive execution of a list of commands, **while** an expression is true. The following Bash script greets all the students in the class using a **while** loop:

```
#!/bin/bash
i=1
END=24
while [[ $i -le $END ]]
do
    echo "$i. Oh hi there, student!"
    ((i = i + 1))
done
```

For loops are useful when you already know in advance how many times you would like to repeat the operations inside the loop. In contrast, you would use a **while** loop when you don't know in advance how many times the operations must be repeated, but do know that they should be stopped once a certain condition is met (which you keep checking for).

- **Create a script to compute the sum of the first 40 natural numbers: 1 + 2 + ... + 40.**
- **Create a script that will keep adding up the numbers the user provides until they provide a negative number. Can you also report how many numbers were input by the user, not including the negative number?**

Bash Tip 5: There are many, many more features to Bash! Check out this resource to learn more: `https://ryanstutorials.net/linuxtutorial`

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 metagenomics sequence reads.

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?**

‽ **Demanding exact k-mer matching in Kraken has benefits for removing human reads. Why?**

‽ **What is the SRA? How many samples are there in the SRA currently? How many bases are 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 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 Uniprot has a taxonomic label. Through a similarity search we will transfer the annotation of the reference protein to our unknown reads.

```
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:

```
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
```

? **What is the most common species in this dataset?**

? **Why are there so many different eukaryotic sequences? Were they really in the spinal fluid sample?**

[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].

2.2.2 Visualizing taxonomic results

MMseqs2 can also generate an interactive visualization of the data using Krona [5]. 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. This offers an interactive circular visualization where you can click on each label to zoom into different parts of the hierarchy. 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://devNN.mmseqs.com/web` in a new tab. There you will see your `report.html` file.

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 number(s) you just found. If you found multiple, concatenate them with a comma $\boxed{,}$ character.

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

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

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

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

```
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 to 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 method 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.

```
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

We will look for known protein domains to identify the proteins we found. Instead of the MMseqs2 command line, we use the MMseqs2 webserver, which will visualize the results. Paste the content of the FASTA file containing the representative sequences into the webserver and make sure to select all three target databases (PFAM, PDB, Uniclust): `https://search.mmseqs.com`

? Which of the expected proteins 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.

Discovering candidate Cas14 orthologs

3.1 Introduction

CRISPR-Cas9 systems provide bacteria and archaea with adaptive immunity to infectious nucleic acids (e.g., viruses), and is widely used as genome editing tools. Recently, Harrington and Burstein et al. [8] discovered CRISPR-Cas systems in archaea that consist of previously unreported Cas14 proteins. These proteins are compact RNA-guided nucleases (400 to 700 amino acids in length). Due to its compact size and special enzymatic property, it has potential to be exploited as a gene editing tool like Cas9. In their work, the authors identified a set of 45 Cas14 proteins by constructing and iteratively refining hidden Markov models (HMMs) of known Cas14 proteins, and using them to query public metagenomes from IMG/M.

3.2 Goal and motivation

We will examine candidate orthologs of Cas14 in order to enrich the authors' original set. This is very useful for improving HMMs, identifying taxa that have this system, as well as to better understand the diversity and functionality of the protein.

In this section you will learn how to:

- create and visualize multiple sequence alignments (MSAs) of protein sequences
- compute a phylogenetic tree
- visualize and interpret the phylogenetic tree

In the interest of time, we carried out some of the computational steps for you. Your tasks are marked in red.

3.3 Input

Our starting point will be the previously reported 45 sequences.[1]

[1] `https://science.sciencemag.org/highwire/filestream/716984/field_highwire_adjunct_files/1/aav4294_Data_S2.fasta`

1. Change to the exercise directory: Cas14
2. Download the sequences by using the command:
   `wget <url_to_sequences_see_footnote>`

? Using Bash commands: What is the average Cas14 length (in amino acids)?
A) 45 B) 563.2 C) 553.5 D) 626.4

There are several possible solutions. Here is one that doesn't require more than basic commands.

```
grep -c "^>" aav4294_Data_S2.fasta
# the number of sequences is: 45

grep -cv "^>" aav4294_Data_S2.fasta
# the number of sequence lines is: 438

grep -v "^>" aav4294_Data_S2.fasta | wc -c
# the number of characters (including \n) in sequence lines is: 25344
```

Solution: (25344 − 438)/45

3.4 How we searched for candidate orthologs

Our chances of finding highly diverse orthologs increase as we explore more comprehensive protein databases. We thus chose to use "**BFD**" (**B**ig well... let's just say... **F**antastic **D**atabase; `https://bfd.mmseqs.com/`), which was constructed from 2,500,000,000 protein sequences of various sources, including environmental samples of soil and ocean. The BFD has been clustered using **Linclust** [9] to 65,983,866 clusters of 30% sequence identity. A multiple sequence alignment (MSA) was computed from each of the BFD clusters.

Like Harrington and Burstein et al., it is useful to use HMMs to search for candidate orthologs of Cas14:

1. Align the previously reported 45 sequences using **MAFFT** [10].
2. Then, run **HHblits** [11] with this alignment as input and the BFD_MSAs as database. This conducts an iterative HMM to HMM search and results in a set of alignments computed from the constructed profiles.
3. Following such a search, we performed several steps and obtained the candidate sequences from the output and saved them in `cas14_bfd_candidates.fasta`.

We performed the search step for you because the BFD database is too large for you to have in your exercise environment.

3.5 Aligning known Cas14 and BFD candidates

The **cas14_bfd_candidates.fasta** file contains three types of sequences: previously reported Cas14 ("CAS" headers), sequences that were found in standard reference databases, such as UniProt ("REF" headers) and sequences that were found strictly in environmental metagenomic samples ("ENV" headers).

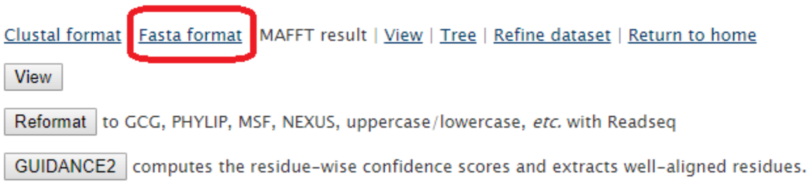
1. Using Bash commands, inspect `cas14_bfd_candidates.fasta` file.

   ❓ **How many sequences are there of each type?**

   Solution:

   ```
   grep ">CAS" cas14_bfd_candidates.fasta | wc -l        # 45
   grep ">REF" cas14_bfd_candidates.fasta | wc -l        # 25
   grep ">ENV" cas14_bfd_candidates.fasta | wc -l        #164
   ```

2. Download `cas14_bfd_candidates.fasta` from `https://devNN.mmseqs.com/web` (replace NN with your number). Align all these sequences using the MAFFT online server and save the result as `cas14_bfd_candidates_MSA.fasta`.[2]



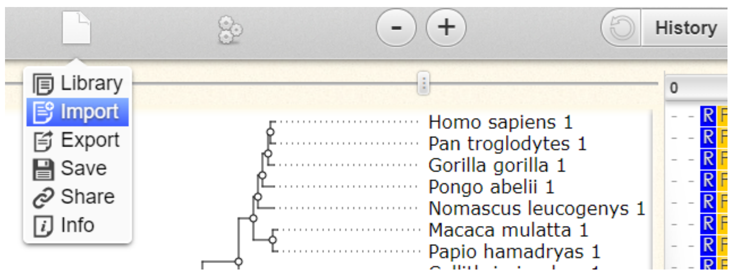
Clustal format | Fasta format | MAFFT result | View | Tree | Refine dataset | Return to home

View

Reformat to GCG, PHYLIP, MSF, NEXUS, uppercase/lowercase, *etc.* with Readseq

GUIDANCE2 computes the residue-wise confidence scores and extracts well-aligned residues.

3. Upload the MSA file to the Wasabi MSA viewer.[3]

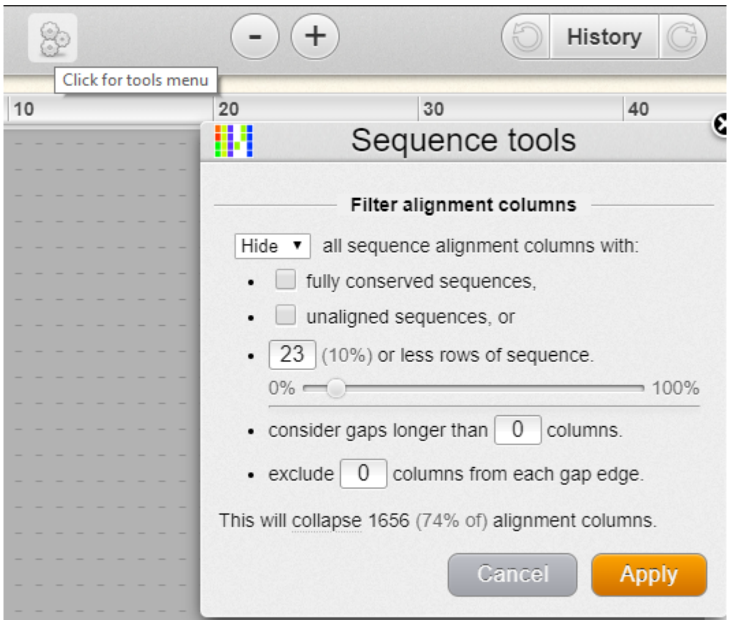


4. Scroll and zoom in and out to get an overall impression.

   ❓ **What can you say about the MSA? For example, what is its length?**

5. Use the "collapse gaps" option:

[2] `https://mafft.cbrc.jp/alignment/server/`
[3] `http://wasabiapp.org/`



Sequence tools

Filter alignment columns

Hide all sequence alignment columns with:

- ☐ fully conserved sequences,
- ☐ unaligned sequences, or
- 23 (10%) or less rows of sequence.
  0% — 100%
- consider gaps longer than 0 columns.
- exclude 0 columns from each gap edge.

This will collapse 1656 (74% of) alignment columns.

? What would be interesting things to consider?

Solution:
There are 2,217 columns in the MSA. The MSA has quite a lot of gap characters so it is hard to examine. After collapsing the gaps with default settings 26% of the columns remain. We can see that these columns correspond to the full length of the original MSA, which reassures us that it is not a single domain that matches. Also, we can see at least some of the ENV sequences resemble the CAS proteins, suggestive of true homologs.

3.6 Computing a phylogenetic tree

A phylogentic tree represents the reconstructed evolution leading to the sequences in a multiple sequence alignment (MSA). There are several ways[4] to infer phylogenetic trees based on MSAs. The likelihood criterion allows scoring each possible tree based on its probability to give rise to the sequences by using a statistical model of sequence evolution. This criterion is often used together with a search procedure to scan and score possible trees until the highest score is reached. Various software tools[5] implement this tree reconstruction strategy. Today, we will use FastTree[6][12], which approximates the maximum likelihood computation to achieve short running times.

Reconstruct a phylogenetic tree using FastTree. To move the MSA results to your environment, create an empty file named `cas14_bfd_candidates_MSA.fasta` and paste the results there.

```
FastTree cas14_bfd_candidates_MSA.fasta > cas14_bfd_candidates_MSA.nwc
```

[4]https://en.wikipedia.org/wiki/Phylogenetic_tree#Construction
[5]https://molbiol-tools.ca/Phylogeny.html
[6]http://www.microbesonline.org/fasttree/

3.7 Viewing the tree

By examining the tree we can learn of the divergence of the bfd candidates. We will use the interactive Tree Of Life (iTOL, Letunic and Bork 2019 Nucleic Acids Res) server to examine the tree. The server allows various tree displays, coloring branches and leaves, adding labels and exporting the tree to common formats, such as PDF.

Upload `cas14_bfd_candidates_MSA.nwc` to the iTOL server[7].

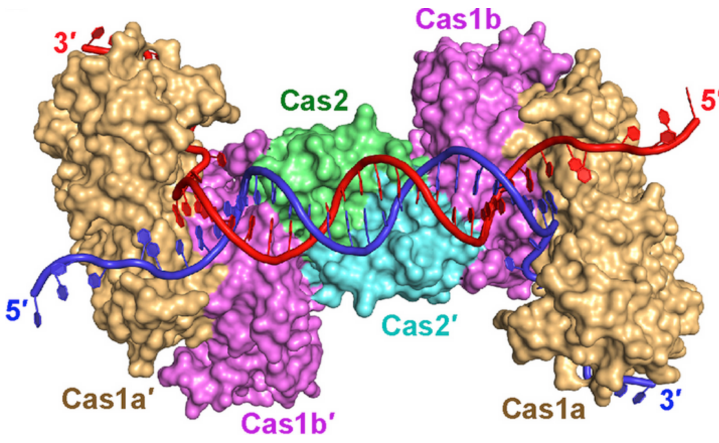
We have prepared an annotation file (`cas14_bfd_candidates_iTOL_leaves.txt`) based on the iTOL format of coloring leave labels. Drag and drop this file on your tree.

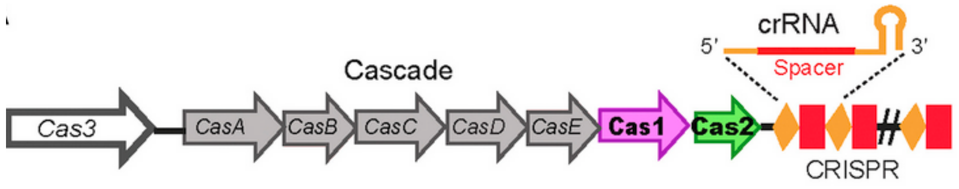
Questions:

- **What can you say about the diversity of sequences in the environmental metagenomic samples, compared to the standard reference database?**
- **Do you think all candidate sequences are Cas14 orthologs? Please explain.**
- **Where would you expect true Cas14 orthologs to be on the phylogenetic tree, and why?**
- **Can you think of other bioinformatic approaches you can take to verify if they are likely true orthologs?**
- **How can you validate those predictions by experimental approaches?**

[7]https://itol.embl.de/

Protein structure prediction





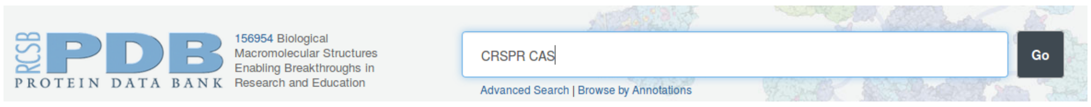
In this section you will learn how to:

1. Search for protein structures on the RCSB PDB[13] and UniProt websites[4];
2. Use visualization tools to explore protein structures and the interface of proteins and DNA;
3. Use homology modeling to predict protein structures and briefly evaluate the predicted models [14].

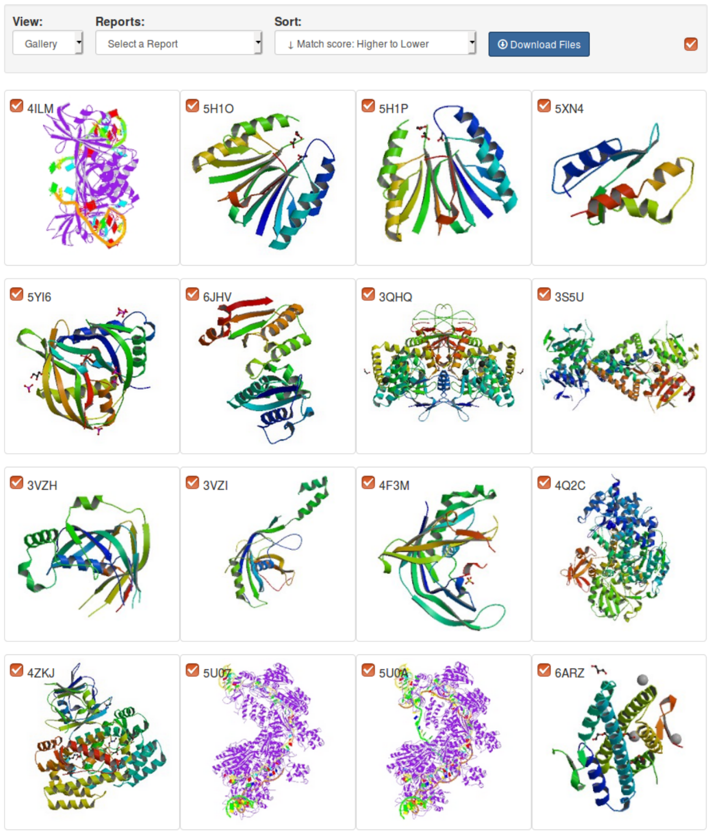
Have fun!

4.1 Finding solved Cas protein structures in the Protein Data Bank (PDB)

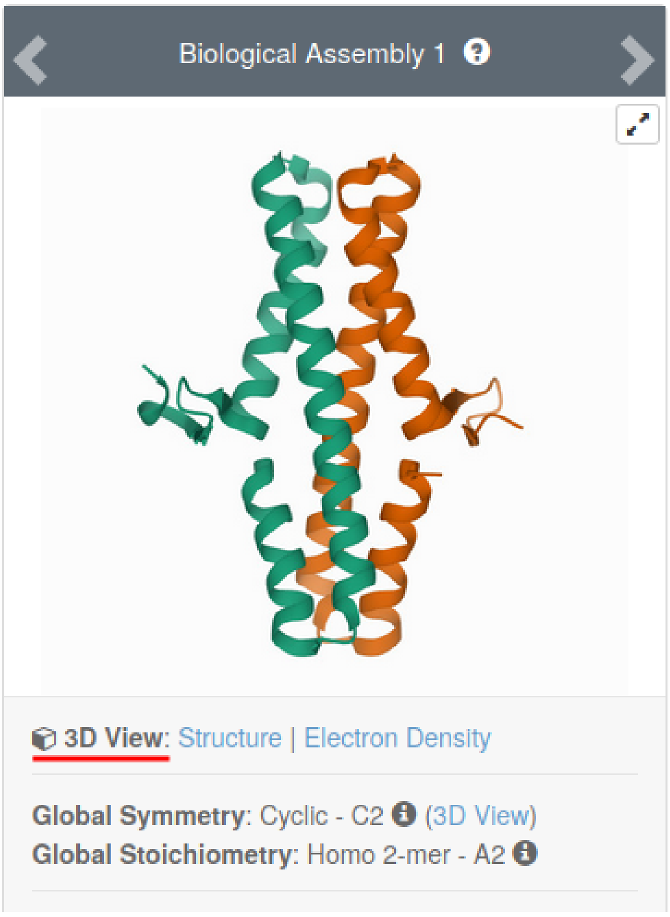
1. Go to the RCSB PDB website: `http://www.rcsb.org`
2. Search with a keyword such as "CRISPR", click the "Go" button:


RCSB PDB PROTEIN DATA BANK
156954 Biological Macromolecular Structures Enabling Breakthroughs in Research and Education
CRSPR CAS
Go
Advanced Search | Browse by Annotations

3. Explore the result page in the "Gallery" view, and you will see a collection of the CRISPR-related protein structures (note: the collection you'll find there today will differ from the ones depicted below):



4. You can click on any of the structures there and briefly explore its web page. In specific, try out the 3D viewer (click on the button highlighted in the image below):
5. For our structural shenanigans, however, we will not be selecting a structure from the PDB (not directly anyway).

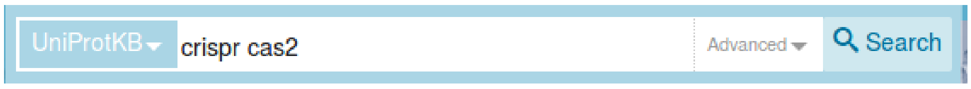


Biological Assembly 1

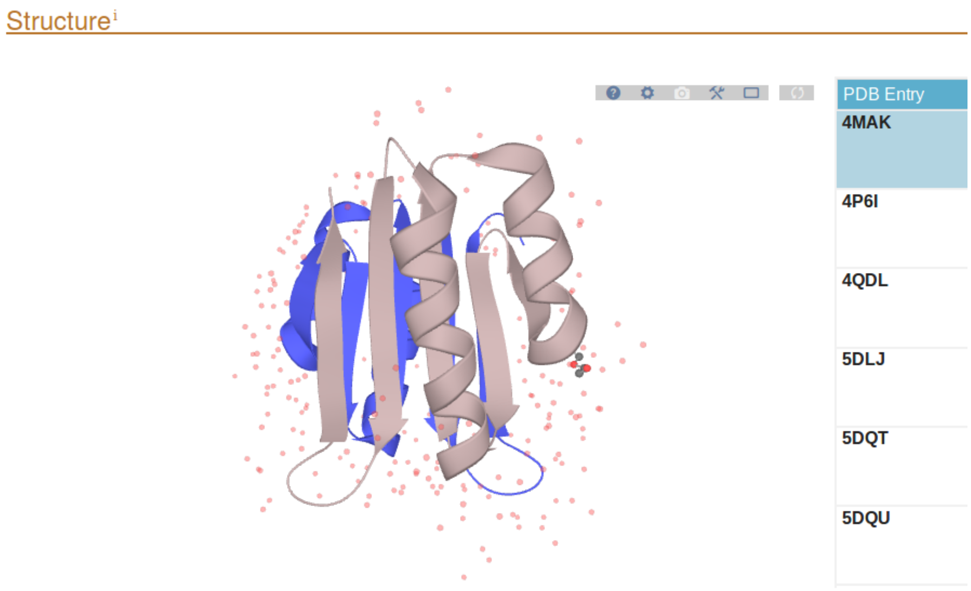
Global Symmetry: Cyclic - C2 (3D View)
Global Stoichiometry: Homo 2-mer - A2

4.2 Basic protein visualization in the UniProt database

1. Go to the UniProt website: `https://www.uniprot.org/`.
2. Search “crispr cas2”:



3. Click on the Entry **“P45956”** for the Cas2 protein from *Escherichia coli* and go to the summary page of this protein.
4. Scroll down the page and find the structure of the Cas2 protein, which shows a homo-dimer from PDB ID **4MAK**. Chain A is shown in pink and chain B is shown in blue



5. Click and drag the mouse on the structure and you will see the structure rotates.
6. Point the mouse on the structure, you will see an amino acid will be highlighted in yellow color and the position and the name of the amino acid will be shown on the top left corner.

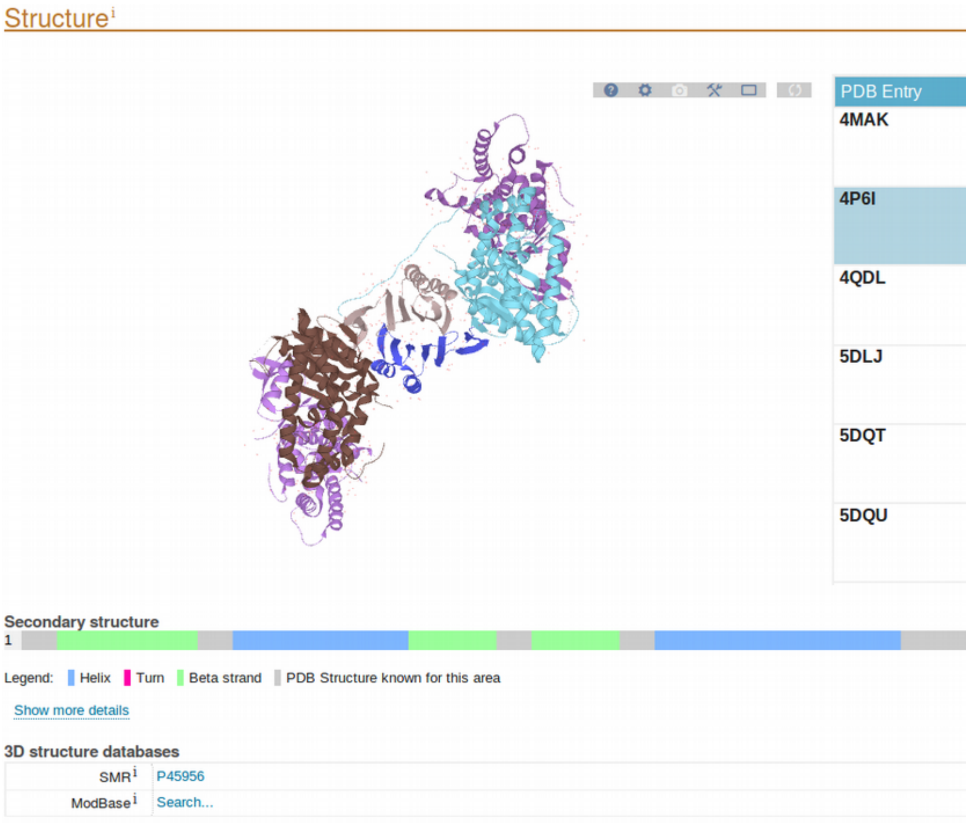
⁈ What is the amino acid of the first alpha-helix on chain A?

(Answer: Pro)

If you click on it, you can see the structure of that single amino acid. (It would zoom in upon clicking.)

4.3 Looking at a Cas1-Cas2 complex from different angles in SWISS-MODEL [15]

1. In the Structure section on the UniProt page `https://www.uniprot.org/uniprot/P45956`, choose the PDB entry **4P6I**, you will see a heteromer structure of Cas1-Cas2 complex:



Structure

PDB Entry
4MAK
4P6I
4QDL
5DLJ
5DQT
5DQU

Secondary structure

1

Legend: Helix Turn Beta strand PDB Structure known for this area

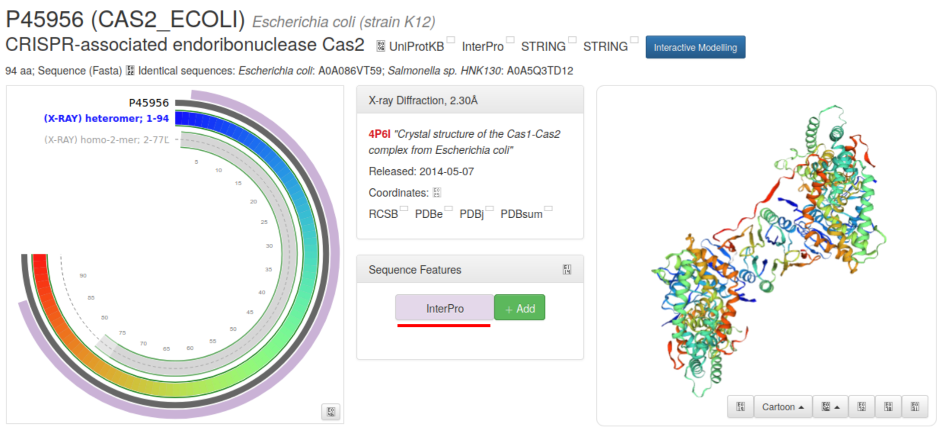
Show more details

3D structure databases

SMR	P45956
ModBase	Search...

2. Click on the SMR entry P45956 under "**3D structure databases**" and this leads you to the Swiss Model Repository.

3. On the left side of the page, it shows the protein sequence while on the right side, it shows the structure of the Cas1-Cas2 complex.



P45956 (CAS2_ECOLI) *Escherichia coli (strain K12)*

CRISPR-associated endoribonuclease Cas2 UniProtKB InterPro STRING STRING Interactive Modelling

94 aa; Sequence (Fasta) Identical sequences: *Escherichia coli*: A0A086VT59; *Salmonella sp. HNK130*: A0A5Q3TD12

P45956

(X-RAY) heteromer; 1-94

(X-RAY) homo-2-mer; 2-77

X-ray Diffraction, 2.30Å

4P6I *"Crystal structure of the Cas1-Cas2 complex from Escherichia coli"*

Released: 2014-05-07

Coordinates:

RCSB PDBe PDBj PDBsum

Sequence Features

InterPro + Add

Cartoon

Tips for playing with the model:

(a) You can rotate the structure by clicking and dragging the left button of the mouse over the structure;

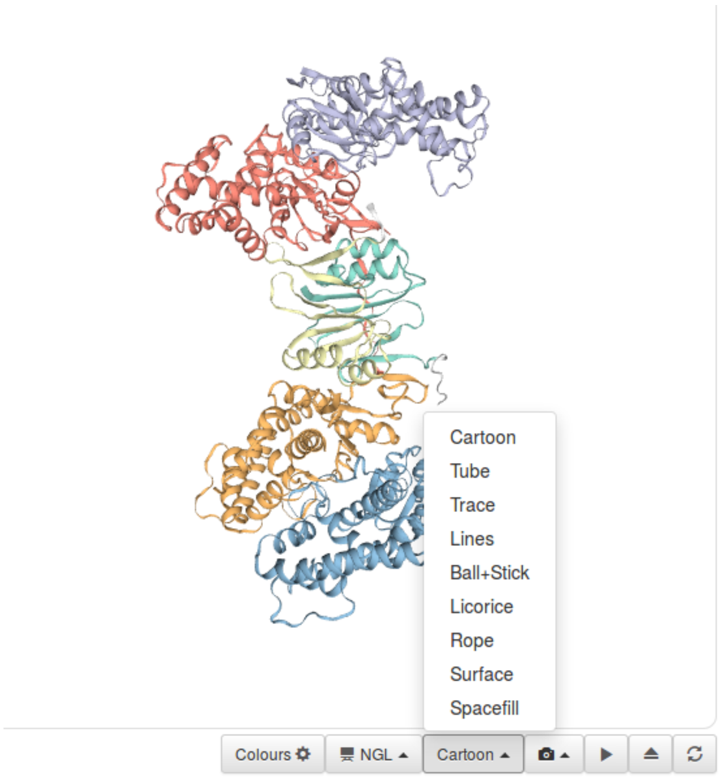
(b) You can zoom in and out of the structure by scrolling the middle button of the mouse;

(c) You can move the structure by clicking and dragging the right button of the mouse over the structure.

4. To find out where Cas2 is located in the complex, click on the "**InterPro**" button in the middle of the page.

5. Change colors: (Before doing this, don't forget to unclick the "InterPro" button)

   (a) By default, the structure is shown in rainbow color scheme. Change it now to "**Chain**". Now you see one of the two Cas2 chains is highlighted in green color, which is corresponding to the sequence on the left, and the other chain is shown in yellow. The other 4 chains are from Cas1 protein.



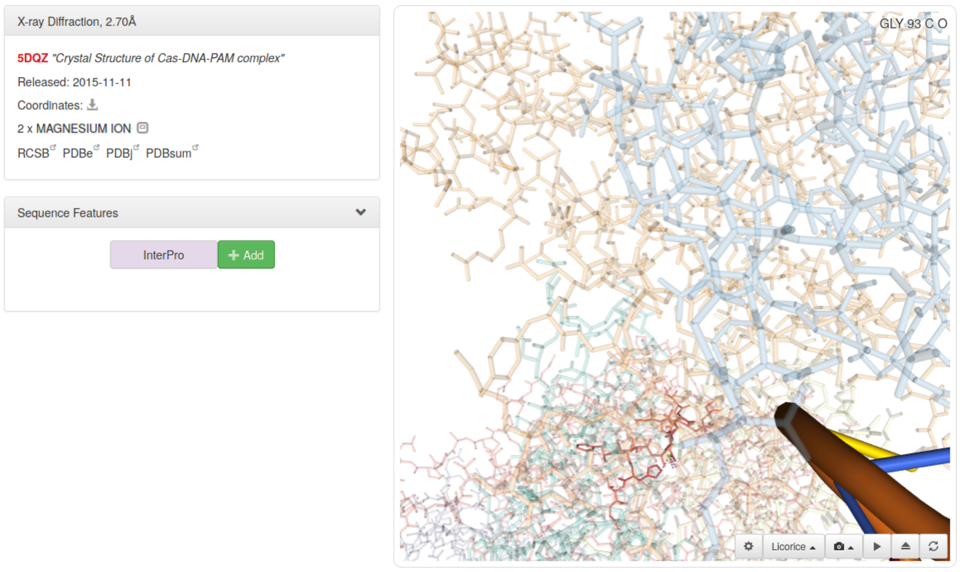
   (b) Change the display mode from Cartoon to Surface, and change the Colours to Charged. Now you will see the positive charged amino acids are shown in red while the negative charged ones are in blue.

6. Find out the interface of Cas1-Cas2 to DNA strands:

(a) On the bottom of the page, choose a different structure: the Cas-DNA-PAM complex with a PDB ID **5dqz** [16]. This structure illustrates the interface between the Cas1-Cas2 and the DNA strands.

(b) Set the Colours to **Chains** and change the display mode to **Spacefill**.

(c) Check the interactions between amino acids and ions. Notice there are 2 Magnesium ions in the structure. Change the display mode to **Licorice**. Click on "2x MAGNESIUM ION":



X-ray Diffraction, 2.70Å

5DQZ *"Crystal Structure of Cas-DNA-PAM complex"*

Released: 2015-11-11

Coordinates:

2 x MAGNESIUM ION

RCSB PDBe PDBj PDBsum

Sequence Features

It will highlight the regions where the Mg^{2+} ions bind to. Click on the little box and you will see the details of interactions.

Which amino acids do the Mg^{2+} ions bind to?

Click on **MG.1**, and you will see which amino acids it binds to. (Answer: E, 2xH, 2xD)

4.4 Homology modeling using SWISS-MODEL

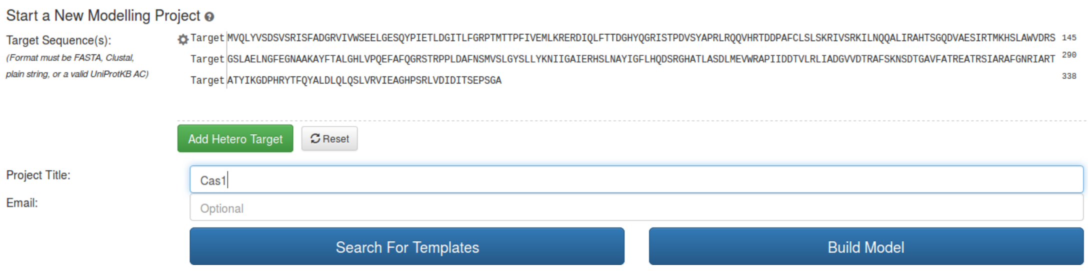
Homology modeling is one of the most useful computational methods for protein structure prediction. Given the evolutionary conservation of protein structures, we can use the known protein structure to predict the structures of proteins which come from a common ancestor.

1. Go to modeling page: `https://swissmodel.expasy.org/interactive`

   Target sequence:

```
>Cas1
MVQLYVSDSVSRISFADGRVIVWSEELGESQYPIETLDGITLFGRPTMTTPFI
VEMLKRERDIQLFTTDGHYQGRISTPDVSYAPRLRQQVHRTDDPAFCLSLS
KRIVSRKILNQQALIRAHTSGQDVAESIRTMKHSLAWVDRSGSLAELNGFEG
NAAKAYFTALGHLVPQEFAFQGRSTRPPLDAFNSMVSLGYSLLYKNIIGAIE
RHSLNAYIGFLHQDSRGHATLASDLMEVWRAPIIDDTVLRLIADGVVDTRA
FSKNSDTGAVFATREATRSIARAFGNRIARTATYIKGDPHRYTFQYALDLQL
QSLVRVIEAGHPSRLVDIDITSEPSGA
```

2. Copy our target sequences above and paste it to the "Target Sequence(s)" window, give a name to this project, and click "Search For Templates" button:



Start a New Modelling Project

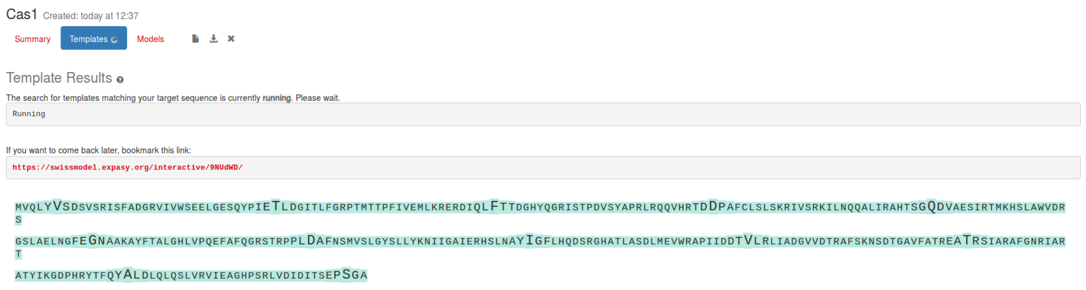
Target Sequence(s):
(Format must be FASTA, Clustal, plain string, or a valid UniProtKB AC)

```
Target MVQLYVSDSVSRISFADGRVIVWSEELGESQYPIETLDGITLFGRPTMTTPFIVEMLKRERDIQLFTTDGHYQGRISTPDVSYAPRLRQQVHRTDDPAFCLSLSKRIVSRKILNQQALIRAHTSGQDVAESIRTMKHSLAWVDRS 145
Target GSLAELNGFEGNAAKAYFTALGHLVPQEFAFQGRSTRPPLDAFNSMVSLGYSLLYKNIIGAIERHSLNAYIGFLHQDSRGHATLASDLMEVWRAPIIDDTVLRLIADGVVDTRAFSKNSDTGAVFATREATRSIARAFGNRIART 290
Target ATYIKGDPHRYTFQYALDLQLQSLVRVIEAGHPSRLVDIDITSEPSGA 338
```

Project Title: Cas1

Email: Optional

3. The process will run for a few minutes, and the intermediate page shows the real time progress:



Cas1 Created: today at 12:37

Template Results

The search for templates matching your target sequence is currently running. Please wait.

```
Running
```

If you want to come back later, bookmark this link:

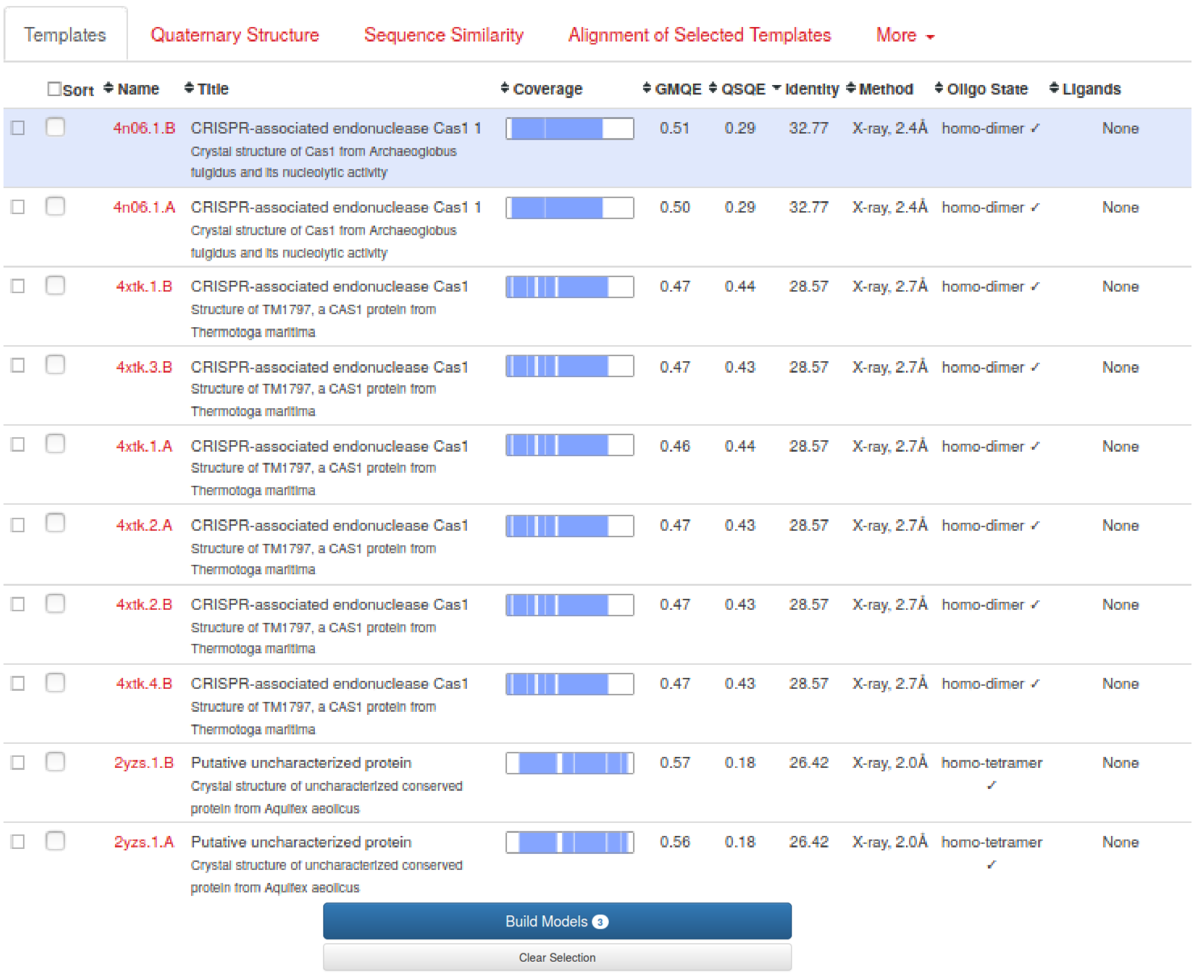
```
https://swissmodel.expasy.org/interactive/9NUdWD/
```

```
MVQLYVSDSVSRISFADGRVIVWSEELGESQYPIETLDGITLFGRPTMTTPFIVEMLKRERDIQLFTTDGHYQGRISTPDVSYAPRLRQQVHRTDDPAFCLSLSKRIVSRKILNQQALIRAHTSGQDVAESIRTMKHSLAWVDR
S
GSLAELNGFEGNAAKAYFTALGHLVPQEFAFQGRSTRPPLDAFNSMVSLGYSLLYKNIIGAIERHSLNAYIGFLHQDSRGHATLASDLMEVWRAPIIDDTVLRLIADGVVDTRAFSKNSDTGAVFATREATRSIARAFGNRIAR
T
ATYIKGDPHRYTFQYALDLQLQSLVRVIEAGHPSRLVDIDITSEPSGA
```

4. The results page shows different protein structure templates that can be used to predict the 3D structure for our target protein. These templates are ranked due to how well their sequences align with the target sequence.

   ⁈ To which structure does it have the most sequence identity?

   Note that sequences falling below a 20% sequence identity can have very different structure.

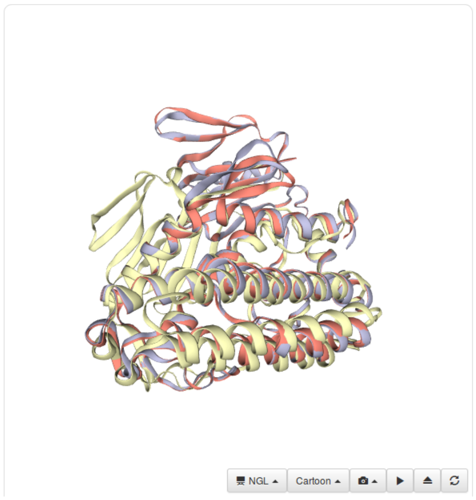


Templates | Quaternary Structure | Sequence Similarity | Alignment of Selected Templates | More

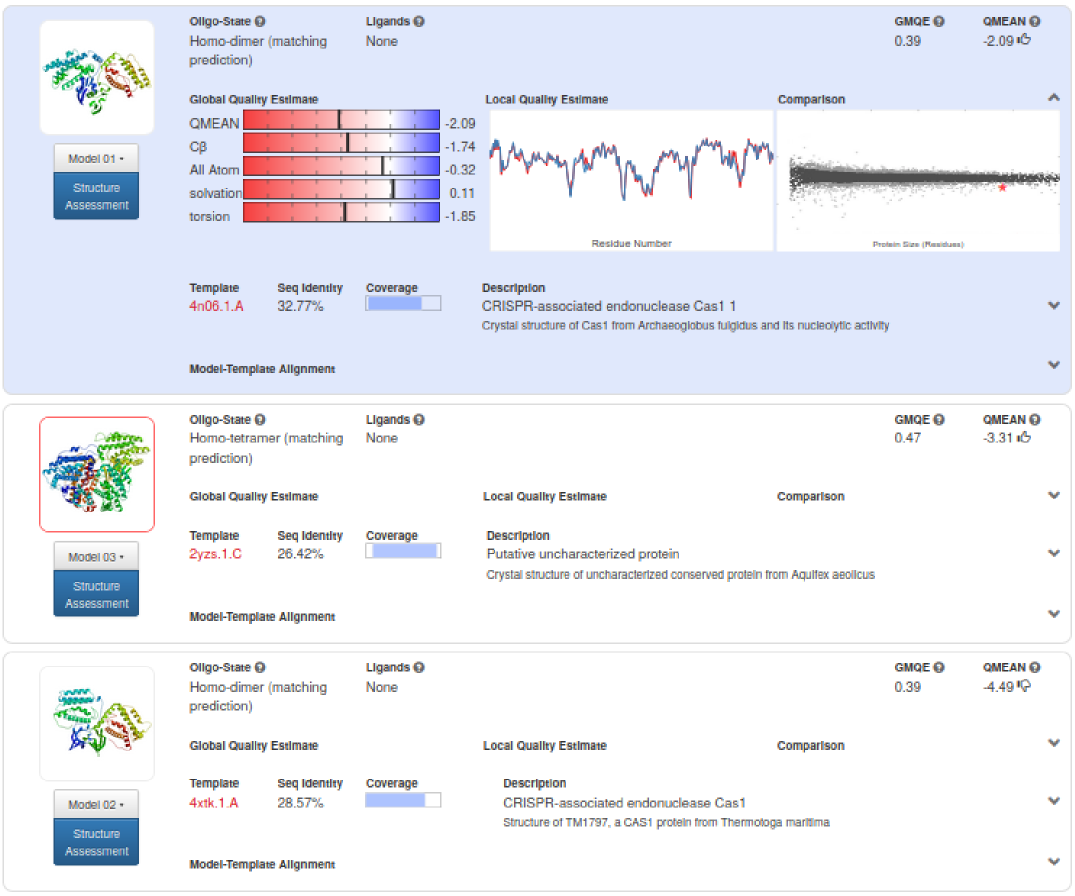
Sort	Name	Title	Coverage	GMQE	QSQE	Identity	Method	Oligo State	Ligands
	4n06.1.B	CRISPR-associated endonuclease Cas1 1 Crystal structure of Cas1 from Archaeoglobus fulgidus and its nucleolytic activity		0.51	0.29	32.77	X-ray, 2.4Å	homo-dimer ✓	None
	4n06.1.A	CRISPR-associated endonuclease Cas1 1 Crystal structure of Cas1 from Archaeoglobus fulgidus and its nucleolytic activity		0.50	0.29	32.77	X-ray, 2.4Å	homo-dimer ✓	None
	4xtk.1.B	CRISPR-associated endonuclease Cas1 Structure of TM1797, a CAS1 protein from Thermotoga maritima		0.47	0.44	28.57	X-ray, 2.7Å	homo-dimer ✓	None
	4xtk.3.B	CRISPR-associated endonuclease Cas1 Structure of TM1797, a CAS1 protein from Thermotoga maritima		0.47	0.43	28.57	X-ray, 2.7Å	homo-dimer ✓	None
	4xtk.1.A	CRISPR-associated endonuclease Cas1 Structure of TM1797, a CAS1 protein from Thermotoga maritima		0.46	0.44	28.57	X-ray, 2.7Å	homo-dimer ✓	None
	4xtk.2.A	CRISPR-associated endonuclease Cas1 Structure of TM1797, a CAS1 protein from Thermotoga maritima		0.47	0.43	28.57	X-ray, 2.7Å	homo-dimer ✓	None
	4xtk.2.B	CRISPR-associated endonuclease Cas1 Structure of TM1797, a CAS1 protein from Thermotoga maritima		0.47	0.43	28.57	X-ray, 2.7Å	homo-dimer ✓	None
	4xtk.4.B	CRISPR-associated endonuclease Cas1 Structure of TM1797, a CAS1 protein from Thermotoga maritima		0.47	0.43	28.57	X-ray, 2.7Å	homo-dimer ✓	None
	2yzs.1.B	Putative uncharacterized protein Crystal structure of uncharacterized conserved protein from Aquifex aeolicus		0.57	0.18	26.42	X-ray, 2.0Å	homo-tetramer ✓	None
	2yzs.1.A	Putative uncharacterized protein Crystal structure of uncharacterized conserved protein from Aquifex aeolicus		0.56	0.18	26.42	X-ray, 2.0Å	homo-tetramer ✓	None

Build Models 3

Clear Selection



5. We select 3 templates with high identity scores and build models upon them. Click the "Build Models" button on the top right. Note that there are multiple matches from the same organism (so account for this). The model results page is shown below. The predicted structures are ranked according to the quality of their models.


Oligo-State
Homo-dimer (matching prediction)
Ligands
None
GMQE
0.39
QMEAN
-2.09

Global Quality Estimate
QMEAN -2.09
Cβ -1.74
All Atom -0.32
solvation 0.11
torsion -1.85

Local Quality Estimate
Comparison

Template
4n06.1.A
Seq Identity
32.77%
Coverage
Description
CRISPR-associated endonuclease Cas1 1
Crystal structure of Cas1 from Archaeoglobus fulgidus and its nucleolytic activity

Model-Template Alignment

Model 01
Structure Assessment

Oligo-State
Homo-tetramer (matching prediction)
Ligands
None
GMQE
0.47
QMEAN
-3.31

Global Quality Estimate
Local Quality Estimate
Comparison

Template
2yzs.1.C
Seq Identity
26.42%
Coverage
Description
Putative uncharacterized protein
Crystal structure of uncharacterized conserved protein from Aquifex aeolicus

Model-Template Alignment

Model 03
Structure Assessment

Oligo-State
Homo-dimer (matching prediction)
Ligands
None
GMQE
0.39
QMEAN
-4.49

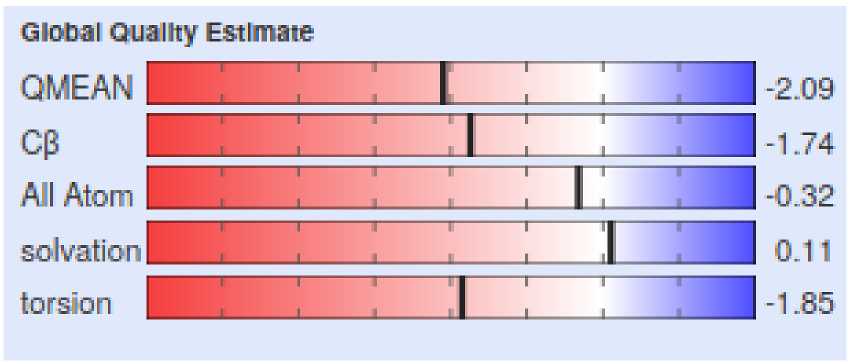
Global Quality Estimate
Local Quality Estimate
Comparison

Template
4xtk.1.A
Seq Identity
28.57%
Coverage
Description
CRISPR-associated endonuclease Cas1
Structure of TM1797, a CAS1 protein from Thermotoga maritima

Model-Template Alignment

Model 02
Structure Assessment

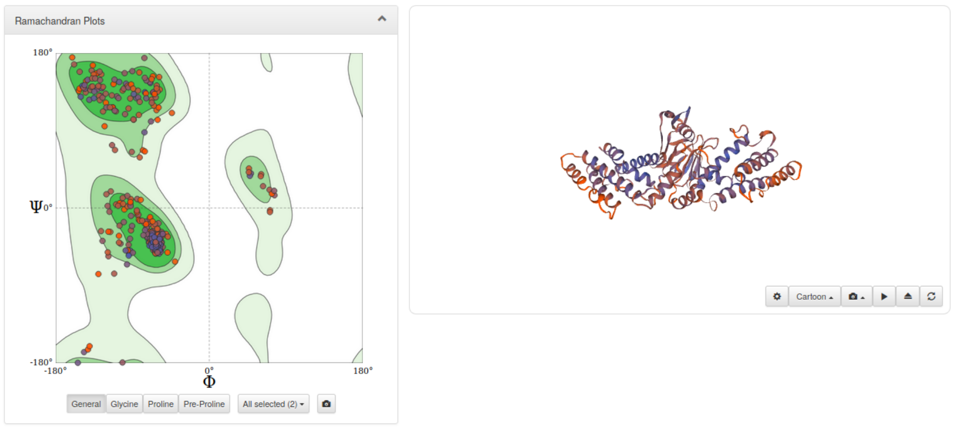
6. Model evaluation: the first model (built using 4n06.1.A as a template) has the best QMEAN score (just sort by QMEAN) and a fair GMQE score; note that the quality estimators mostly fall near the white region. Therefore, it is the most optimal model that we can get for the target sequence, given the available homologous structures. (See below for explanations of GMQE and QMEAN.)


Global Quality Estimate
QMEAN -2.09
Cβ -1.74
All Atom -0.32
solvation 0.11
torsion -1.85

GMQE(Global Model Quality Estimation) is a quality estimation which combines properties from the target–template alignment and the template search method. The resulting GMQE score is expressed as a number between 0 and 1, reflecting the expected accuracy of a model built with that alignment and template and the coverage of the target. Higher numbers indicate higher reliability.

QMEAN is a composite estimator based on different geometrical properties and provides both global (i.e. for the entire structure) and local (i.e. per residue) absolute quality estimates on the basis of one single model. Scores of -4.0 or below are an indication of models with low quality.

7. This brings us to the end of this short homology modeling tutorial! As a last "task" you can take a quick look at the Ramachandran plot for one of the models by clicking on "Structure Assessment" in the Model Results page:



Appendix

5.1 Some useful Bash commands

```bash
# 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

# 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
```

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

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

2.
   ```
   #!/bin/bash
   AGE=99
   AGE_NEXT_YEAR=$((AGE + 1))
   echo "Next year you will be $AGE_NEXT_YEAR"
   ```

3.
   ```
   #!/bin/bash
   echo "Enter your age and press [ENTER]: "
   read USER_AGE
   if [ $USER_AGE -ge 18 ]; then
           echo "Here is your beer"
   fi
   ```

4.
   ```
   #!/bin/bash
   START=1
   END=40
   SUM=0
   for ((i=$START; i<=$END; i++)) do
       SUM=$((SUM+i))
   done
   echo "The result is $SUM"
   ```

5.
```
#!/bin/bash
USER_NUMBER=0
NUM_NUMBERS=-1
SUM=0
while [[ $USER_NUMBER -ge 0 ]]
do
        SUM=$((SUM+USER_NUMBER))
        NUM_NUMBERS=$((NUM_NUMBERS+1))
        echo "Insert a new number [negative number to exit]:"
        read USER_NUMBER
done
echo "Final sum is $SUM and $NUM_NUMBERS numbers were summed"
```

6.
```
#Alternative solution for question 5. Don't copy this comment into the script!!

#!/bin/bash

SUM=0;
NUMENTRIES=0;
USERINP=0;

while [[ "$USERINP" -ge 0 ]]
do
    echo "Please enter a number (and press enter): ";
    read USERINP;

    if [ "$USERINP" -ge 0 ]; then
        SUM=$((SUM+USERINP));
        NUMENTRIES=$((NUMENTRIES+1));
    fi

done

echo "The sum is $SUM";
echo "The number of positive numbers entered is $NUMENTRIES";
```

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] Lucas B Harrington, David Burstein, Janice S Chen, David Paez-Espino, Enbo Ma, Isaac P Witte, Joshua C Cofsky, Nikos C Kyrpides, Jillian F Banfield, and Jennifer Doudna. Programmed dna destruction by miniature crispr-cas14 enzymes. *Science.*, 2018.

[9] Martin Steinegger and Johannes Söding. Clustering huge protein sequence sets in linear time. *Nat. Commun.*, 9(1):2542, 2018.

[10] Kazutaka Katoh and Daron M. Standley. MAFFT multiple sequence alignment software version 7: Improvements in performance and usability. *Mol. Biol. Evol.*, 30(4), 2013.

[11] Michael Remmert, Andreas Biegert, Andreas Hauser, and Johannes Söding. HHblits: lightning-fast iterative protein sequence searching by HMM-HMM alignment. *Nat. Methods*, 9(2):173–175, 2012.

[12] Morgan N. Price, Paramvir S. Dehal, and Adam P. Arkin. FastTree 2 - Approximately maximum-likelihood trees for large alignments. *PLoS One*, 5(3), 2010.

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

[14] Elmar Krieger, Sander B. Nabuurs, and Gert Vriend. Homology modeling. In *Dict. Genomics, Transcr. Proteomics*, volume 44, pages 1–1. 2015.

[15] Marco Biasini, Stefan Bienert, Andrew Waterhouse, Konstantin Arnold, Gabriel Studer, Tobias Schmidt, Florian Kiefer, Tiziano Gallo Cassarino, Martino Bertoni, Lorenza Bordoli, and Torsten Schwede. SWISS-MODEL: Modelling protein tertiary and quaternary structure using evolutionary information. *Nucleic Acids Res.*, 42(W1), 2014.

[16] Jiuyu Wang, Jiazhi Li, Hongtu Zhao, Gang Sheng, Min Wang, Maolu Yin, and Yanli Wang. Structural and mechanistic basis of pam-dependent spacer acquisition in crispr-cas systems. *Cell*, 163(4):840–853, 2015.