



DIPARTIMENTO  
DI SCIENZE AZIENDALI  
MANAGEMENT  
& INNOVATION SYSTEMS



DIPARTIMENTO DI  
CHIMICA E BIOLOGIA "A. ZAMBELLI"



UNIVERSITÀ DEGLI STUDI  
DI SALERNO

## ALPHAMOD SUPPLEMENTARY FILE 2 USER MANUAL

### The Critical Assessment of Protein Structure Prediction Competition (CASP)

The [CASP](#) competition provides an independent mechanism for the assessment of methods of protein structure modeling. Specifically for CASP 14 starting from May through August 2020, CASP organizers have been posting on their website sequences of unknown protein structures for modeling. Protein models have been collected from May through mid-September, and evaluated as the experimental coordinates become available. In the summer and fall, the tens of thousands of models submitted by approximately 100 research groups worldwide are processed and evaluated. Independent assessors in each of the prediction categories bring independent insight into their assessment. Tools for viewing, comparison, and analysis of submitted models are available from this website.

# Contents

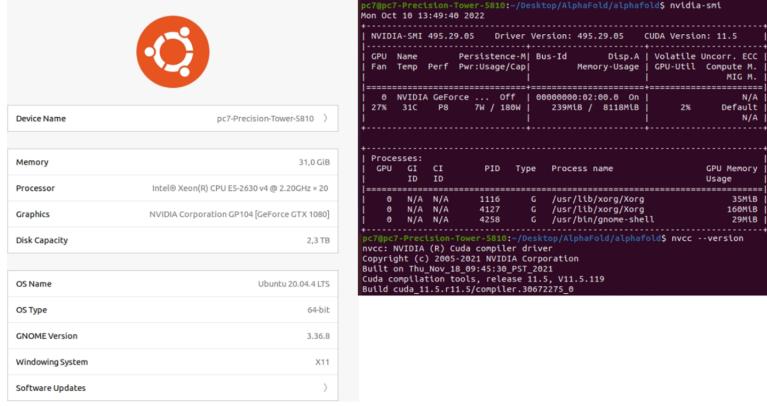
<b>1 Requirements</b>	<b>3</b>
1.1 Hardware requirements . . . . .	3
1.2 Software requirements . . . . .	3
1.2.1 NVIDIA, CUDA and CuDNN driver installation . . . . .	4
1.2.2 Python3 installation . . . . .	12
1.2.3 Pycharm Installation (Optional) . . . . .	12
1.2.4 Docker Installation . . . . .	12
1.2.5 AlphaFold Installation . . . . .	15
1.2.6 Modeller Installation (Optional) . . . . .	16
1.2.7 Phenix Installation (Optional: Required to calculate RMSD)	16
<b>2 Fasta Files</b>	<b>17</b>
2.1 CASP14 Target List . . . . .	17
2.2 Downloading Aminoacid Sequences . . . . .	18
2.3 Saving as fasta format . . . . .	20
2.4 Adjusting FastaFile to comply with AlphaMod guidelines . . . . .	20
<b>3 AlphaMod</b>	<b>23</b>
3.1 Pre-requisites . . . . .	23
3.2 Downloading AlphaMod's GitHub Repository . . . . .	23
3.3 Managing AlphaMod's configuration (config.json) . . . . .	24
3.3.1 AlphaFold_Prediction_List . . . . .	24
3.3.2 StructureAssessment . . . . .	25
3.4 Launching AlphaMod . . . . .	29

# 1 Requirements

## 1.1 Hardware requirements

*AlphaMod* was tested on a PC with the following specifications:

- **Processor:** Intel® Xeon(R) CPU E5-2630 v4 @ 2.20GHz × 20
- **RAM Memory:** 32GB
- **GPU:** GeForce GTX 1080
- **Disk Capacity:** 2,3 TB
- **Operating System:** Ubuntu 20.04.4 LTS / 64 bit



The figure shows a composite screenshot of a Linux desktop environment. On the left, there is a window titled 'pc7-Precision-Tower-5810' displaying various system statistics: Device Name (pc7-Precision-Tower-5810), Memory (31,0 GiB), Processor (Intel® Xeon(R) CPU E5-2630 v4 @ 2.20GHz × 20), Graphics (NVIDIA Corporation GF104 [GeForce GTX 1080]), Disk Capacity (2,3 TB), OS Name (Ubuntu 20.04.4 LTS), OS Type (64-bit), GNOME Version (3.36.8), Windowing System (X11), and Software Updates. On the right, there is a terminal window showing the output of two commands: 'nvidia-smi' and 'nvcc --version'. The 'nvidia-smi' command provides detailed information about the GPU, including its name (NVIDIA-SMI 495.29.05), driver version (495.29.05), CUDA version (11.5), and memory usage. The 'nvcc --version' command shows the NVIDIA (R) Cuda compiler driver version (2005-2021) and the Cuda compilation tools release 11.5, V11.5.119.

```
pc7@pc7-Precision-Tower-5810:~/Desktop/AlphaFold$ nvidia-smi
Mon Jul 10 13:16:20 2022
+-----+
| NVIDIA-SMI 495.29.05      Driver Version: 495.29.05    CUDA Version: 11.5 |
| Persistence-M | Bus-Id   Dlsp.A | Volatile Uncorr. ECC |
| Fan Temp Perf Pwr/Usage/Cap| Memory-Usage | GPU-Util  Compute M. |
|=====|
| 0  NVIDIA GeForce ... Off | 00000000:02:00.0 On | N/A       |
| 27% 31C P8    7W / 180W | 239M/LB / 8108MB | 2% Default |
|=====|
Processes:
| GPU ID CI ID PID Type Process name          GPU Memory Usage |
|=====|
| 0 N/A N/A 4116 G /usr/lib/xorg/Xorg           150MB |
| 1 N/A N/A 4127 G /usr/lib/xorg/Xorg           150MB |
| 2 N/A N/A 4258 G /usr/bin/gnome-shell          29MB |
+-----+
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2021 NVIDIA Corporation
Version 11.5.119 (2021-07-08T14:45:45Z)
Cuda compilation tools: release 11.5, V11.5.119
Build cuda_11.5.r11.5/compiler.30672275_0
```

Figure 1: Hardware specifications of the computer we used to run AlphaFold in our research.

## 1.2 Software requirements

To run *AlphaMod* you will need the following required software:

- NVIDIA, CUDA and CuDNN drivers.
- Python3 v3.8.10.
- PyCharm (Optional).
- Docker.
- AlphaFold.
- Modeller (Optional).
- Phenix (Optional).

### 1.2.1 NVIDIA, CUDA and CuDNN driver installation

*AlphaMod* uses CUDA Toolkit 11.5.1 (November 2021) as shown in Fig.2.

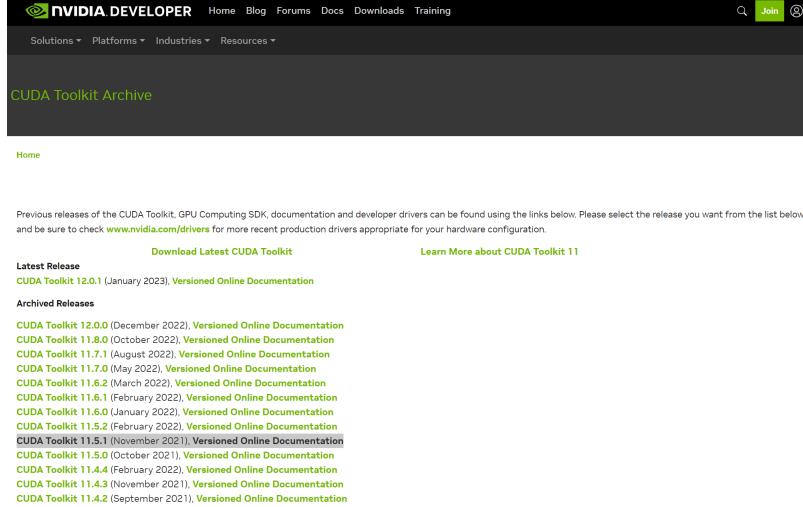


Figure 2: Screenshot of NVIDIA's tool kit archive, highlighting version 11.5.1

Please click on version CUDA Toolkit 11.5.1 (November 2021). According to our described hardware at Section 1, the following options were chosen as shown in Fig.3

- **Operating System:** Linux
- **Architecture:** x86\_64
- **Distribution:** Ubuntu
- **Version:** 20.04
- **Installer Type:** runfile (local)

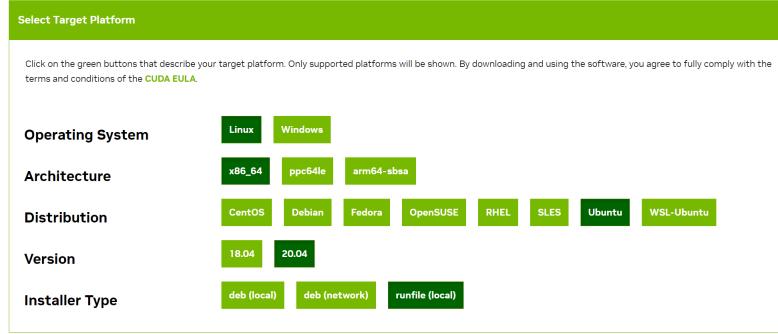


Figure 3: Screenshot of NVIDIA’s CUDA Toolkit 11.5.1 (November 2021) selected Target Platform.

The screen shown in Fig.4 will be prompted



Figure 4: Screenshot of NVIDIA’s CUDA Toolkit 11.5.1 (November 2021) installer download screen.

At the desired location (in our example Downloads folder) copy and paste the installation instructions from Fig.4 into your terminal as shown in Fig.5

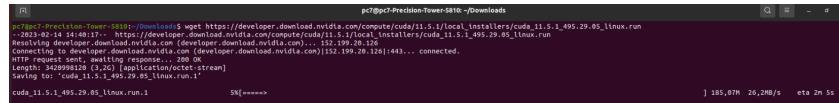


Figure 5: Screenshot of our terminal downloading NVIDIA’s CUDA Toolkit 11.5.1 (November 2021)

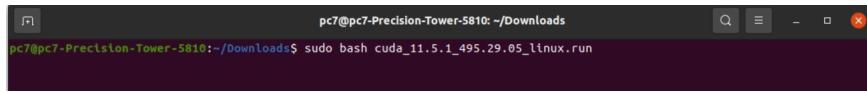
Once NVIDIA's CUDA Toolkit 11.5.1 (November 2021) has been downloaded, it is necessary to give executing permissions to the file with the following command `sudo chmod +x cuda_11.5.1_495.29.05_linux.run` as shown in figure 6.



```
pc7@pc7-Precision-Tower-5810: ~/Downloads
pc7@pc7-Precision-Tower-5810:~/Downloads$ sudo chmod +x cuda_11.5.1_495.29.05_linux.run
```

Figure 6: Screenshot of our terminal giving executing permissions to NVIDIA's CUDA Toolkit 11.5.1 (November 2021)

Now, to proceed with the installation of NVIDIA's CUDA Toolkit 11.5.1 (November 2021), please execute the file with the following command `sudo bash cuda_11.5.1_495.29.05_linux.run` as shown in figure 7.



```
pc7@pc7-Precision-Tower-5810: ~/Downloads
pc7@pc7-Precision-Tower-5810:~/Downloads$ sudo bash cuda_11.5.1_495.29.05_linux.run
```

Figure 7: Screenshot of our terminal executing NVIDIA's CUDA Toolkit 11.5.1 (November 2021)

To proceed with the installation it is necessary to *accept* the *End User License Agreement*, if you agree type *accept* to proceed with the installation as shown in figure 8.



```
pc7@pc7-Precision-Tower-5810: ~/Downloads
-----
End User License Agreement
-----
NVIDIA Software License Agreement and CUDA Supplement to
Software License Agreement. Last updated: October 8, 2021

The CUDA Toolkit End User License Agreement applies to the
NVIDIA CUDA Toolkit, the NVIDIA CUDA Samples, the NVIDIA
Display Driver, NVIDIA Nsight tools (Visual Studio Edition),
and the associated documentation on CUDA APIs, programming
model and development tools. If you do not agree with the
terms and conditions of the license agreement, then do not
download or use the software.

Last updated: October 8, 2021.

Preface
-----
Do you accept the above EULA? (accept/decline/quit):
accept
```

Figure 8: Screenshot of our terminal accepting end user licence of NVIDIA's CUDA Toolkit 11.5.1 (November 2021)

By default NVIDIA's CUDA Toolkit 11.5.1 (November 2021) has drivers and CUDA TOOLKIT for installation, in our research we made the default installation as shown in figure 9.



Figure 9: Screenshot of our terminal of the options used in the installation of NVIDIA's CUDA Toolkit 11.5.1 (November 2021)

After rebooting your pc you can confirm your NVIDIA drivers installation with the following command ***nvidia-smi*** as shown in figure 10.

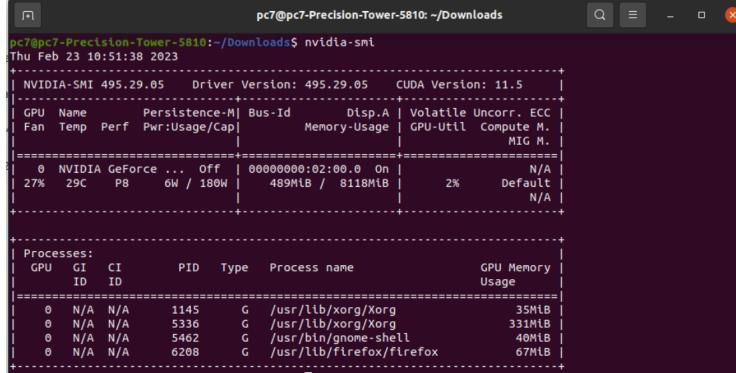


Figure 10: Screenshot of our terminal showing installed NVIDIA drivers

Now we need to activate the ***environmental variables*** for our Linux system to recognize NVIDIA's TOOLKIT. To activate the environmental variables, we will modify the ***bashrc*** file with the ***nano*** editor. Please open the ***bashrc*** file with ***nano*** text editor using the following command: ***nano ./bashrc*** as shown in figure 11.

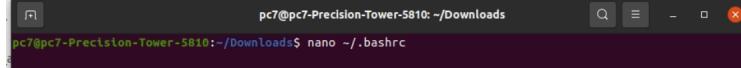


Figure 11: Screenshot of our terminal opening bashrc file with nano text editor

To let our Linux system recognize NVIDIA's TOOLKIT we will add three lines of text to the **bashrc** file using the **nano** editor:

- **EX1:** export PATH=\$PATH:/usr/local/cuda-11.5/bin
- **EX2:** export LD\_LIBRARY\_PATH=\$LD\_LIBRARY\_PATH:/usr/local/cuda-11.5/lib64
- **EX3:** export CUDADIR=/usr/local/cuda-11.5

At the end of the file **bashrc** please copy the lines **EX1**, **EX2** and **EX3** as shown in figure 12.

```

GNU nano 4.8                               /home/pc7/.bashrc
alias la='ls -A'
alias l='ls -C'

# Add an "alert" alias for long running commands.  Use like so:
# sleep 10; alert
alias alert='notify-send --urgency=low -t "$([ $? = 0 ] && echo terminal || echo error)" "$(history|tail -n 1)'

# Alias definitions.
# You may want to put all your additions into a separate file like
# ~/.bash_aliases, instead of adding them here directly.
# See /usr/share/doc/bash-doc/examples in the bash-doc package.

if [ -f ~/.bash_aliases ]; then
. ~/.bash_aliases
fi

# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
if ! shopt -q posix; then
if [ -f /usr/share/bash-completion/bash_completion ]; then
. /usr/share/bash-completion/bash_completion
elif [ -f /etc/bash_completion ]; then
. /etc/bash_completion
fi
fi

export PATH=$PATH:/usr/local/cuda-11.5/bin
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/cuda-11.5/lib64
export CUDADIR=/usr/local/cuda-11.5

```

Figure 12: Screenshot of our terminal copying EX1, EX2 and EX3 on our bashrc file with nano text editor

To verify NVIDIA's TOOLKIT installation please type the following command: ***nvcc –version*** as shown in figure 13.

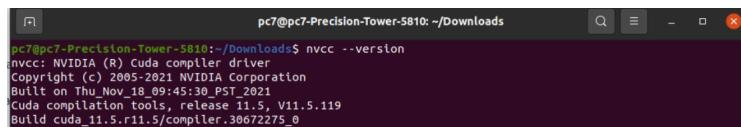


Figure 13: Screenshot of our terminal verifying the installation of NVIDIA's TOOLKIT

Furthermore, we activated cuDNN tools. To activate cuDNN we went to [NVIDIA's cuDNN official website](#) as shown in figure 14.



Figure 14: Screenshot of NVIDIA's cuDNN archive.

Please find the associated cuDNN version for your CUDA TOOLKIT version. In our case we selected it by using the following criteria:

- **C1:** cuDNN v8.3.1 (November 22nd, 2021), for CUDA 11.5
- **C2:** Local Installer for Linux x86\_64 (Tar)

An example can be seen in figure 15.



Figure 15: Screenshot of NVIDIA's cuDNN archive selecting criteria C1 and C2.

Once the file has been downloaded please **unzip** the file with the following command: `tar -xf cudnn-linux-x86_64-8.3.1.22_cuda11.5-archive.tar.xz` as shown in figure 16.

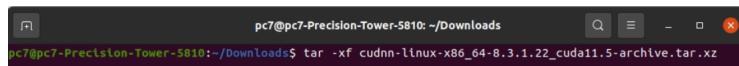
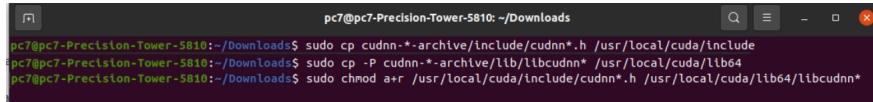


Figure 16: Screenshot of our terminal unzipping cuDNN.

At **Downloads** folder please copy the required files using the following commands:

- **CP1:** sudo cp cudnn-\*-archive/include/cudnn\*.h /usr/local/cuda/include
- **CP2:** sudo cp -P cudnn-\*-archive/lib/libcudnn\* /usr/local/cuda/lib64
- **CP3:** sudo chmod a+r /usr/local/cuda/include/cudnn\*.h /usr/local/cuda/lib64/libcudnn\*

An example can be seen in figure 17.



```
pc7@pc7-Precision-Tower-5810: ~/Downloads
pc7@pc7-Precision-Tower-5810: ~/Downloads$ sudo cp cudnn-*archive/include/cudnn*.h /usr/local/cuda/include
pc7@pc7-Precision-Tower-5810: ~/Downloads$ sudo cp -P cudnn-*archive/lib/libcudnn* /usr/local/cuda/lib64
pc7@pc7-Precision-Tower-5810: ~/Downloads$ sudo chmod a+r /usr/local/cuda/include/cudnn*.h /usr/local/cuda/lib64/libcudnn*
```

Figure 17: Screenshot of our terminal executing **CP1**, **CP2** and **CP3**.

To verify cuDNN's installation go to **NVIDIA\_CUDA-11.5\_Samples** located at **home** as seen in figure 18.



Figure 18: Screenshot of our **home** directory with focus on **NVIDIA\_CUDA-11.5\_Samples** folder.

To create the test examples to confirm cuDNN installation we run the following command: **sudo make -k** as shown in figure



```
pc7@pc7-Precision-Tower-5810: ~/NVIDIA_CUDA-11.5_Samples
pc7@pc7-Precision-Tower-5810: ~/NVIDIA_CUDA-11.5_Samples$ sudo make -k
```

Figure 19: Screenshot of our terminal executing **sudo make -k** command to create test samples for cuDNN.

Once the `sudo make -k` command has finished executing, we proceed to enter the `release` folder with the following command: `cd bin/x86_64/linux/release/` and then we execute the test query with `./deviceQuery` as show in figure 20.

```

pc7@pc7-Precision-Tower-5810:~/NVIDIA_CUDA-11.5_Samples/bin/x86_64/linux/release$ cd bin/x86_64/linux/release/
pc7@pc7-Precision-Tower-5810:~/NVIDIA_CUDA-11.5_Samples/bin/x86_64/linux/release$ ./deviceQuery
./deviceQuery Starting...

```

CUDA Device Query (Runtime API) version (CUDART static linking)

Detected 1 CUDA Capable device(s)

Device 0: "NVIDIA GeForce GTX 1080"

CUDA Driver Version / Runtime Version	11.5 / 11.5
CUDA Capability Major/Minor version number	6.1
Total amount of global Memory	8118 Mbytes (8512602112 bytes)
(620) Multiprocessors, (128) CUDA Cores/MP	2048 CUDA Cores
GPU Max Clock rate	1734 MHz (1.73 GHz)
Memory Clock rate	5095 MHz
Memory Bus Width	256-bit
L2 Cache Size	2097152 bytes
Maximum Texture Dimension Size (x,y,z)	1D=(131072), 2D=(131072, 65536), 3D=(16384, 16384, 16384)
Maximum Layered 1D Texture Size, (num) layers	1D=(32768), 2048 layers
Maximum Layered 2D Texture Size, (num) layers	2D=(32768, 32768), 2048 layers
Total amount of constant memory	65536 bytes
Total amount of shared memory per block	49152 bytes
Total shared memory per multiprocessor	98304 bytes
Total number of registers available per block	65536
Warp size	32
Maximum number of threads per multiprocessor	2048
Maximum number of threads per block	1024
Max dimension size of a thread block (x,y,z): (1024, 1024, 64)	
Max dimension size of a grid size (x,y,z): (2147483647, 65535, 65535)	
Maximum memory pitch	2147483647 bytes
Texture alignment	512 bytes
Concurrent copy and kernel execution	Yes with 2 copy engine(s)
Run time limit on kernels	Yes
Integrated/Discrete Host Memory	No
Support host page-locked memory mapping	Yes
Allignment requirement for surfaces	Yes
Device has ECC support	Disabled
Device supports Unified Addressing (UVA)	Yes
Device supports Managed Memory	Yes
Device supports Compute Preemption	Yes
Supports Cooperative Kernel Launch	Yes
Supports MultiDevice Co-op Kernel Launch	Yes
Device PCI Domain ID / Bus ID / location ID	0 / 2 / 0
Compute Mode:	< Default (multiple host threads can use ::cudaSetDevice() with device simultaneously) >

```

deviceQuery, CUDA Driver = CUDART, CUDA Driver Verstion = 11.5, CUDA Runtime Verstion = 11.5, NumDevs = 1
Result = PASS

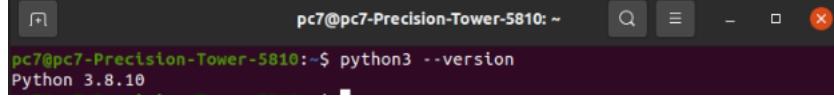
```

Figure 20: Screenshot of our terminal executing cuDNN test query.

The last row of the execution states: `Result = PASS`, confirming that our installation was done correctly.

### 1.2.2 Python3 installation

Before installing Python3 we can verify if the system has it already installed with the following command: ***python3 –version*** as shown in figure 21.

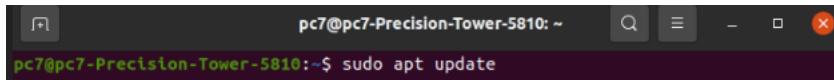


```
pc7@pc7-Precision-Tower-5810:~$ python3 --version
Python 3.8.10
```

Figure 21: Screenshot of our terminal checking python3's version.

If python3 version 3.8.10 is installed in your local machine we can proceed to [1.2.3](#), otherwise we will start with the installation process.

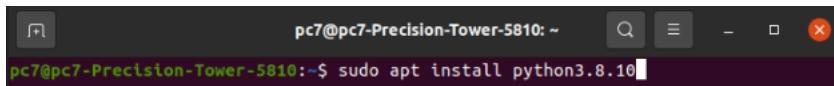
First, we will **update and refresh repository lists** with the following command: ***sudo apt update*** as shown in figure 22



```
pc7@pc7-Precision-Tower-5810:~$ sudo apt update
```

Figure 22: Screenshot of our terminal updating and refreshing repository lists.

Followed, by the python3 installation command ***sudo apt install python3.8.10*** as shown in figure 23.

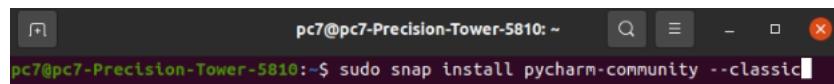


```
pc7@pc7-Precision-Tower-5810:~$ sudo apt install python3.8.10
```

Figure 23: Screenshot of our terminal installing Python v3.8.10.

### 1.2.3 Pycharm Installation (Optional)

To install PyCharm, please open a terminal and prompt the following command: ***sudo snap install pycharm-community --classic*** as shown in figure 24.



```
pc7@pc7-Precision-Tower-5810:~$ sudo snap install pycharm-community --classic
```

Figure 24: Screenshot of our terminal installing PyCharm community edition.

### 1.2.4 Docker Installation

For the installation of Docker we will use a bash script. First, we will create the bash script file with nano editor by prompting the following command into the terminal: ***nano docker\_install.sh*** as shown in figure 25

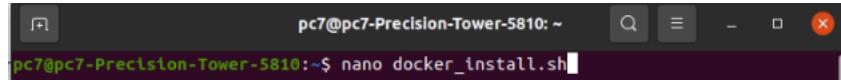


Figure 25: Screenshot of our terminal creating the docker installation bash script file.

Copy the following set of commands:

```
#!/bin/bash

sudo apt-get update

sudo apt-get install \
    ca-certificates \
    curl \
    gnupg \
    lsb-release

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg

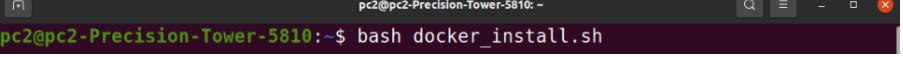
echo \
"deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu \$(lsb_release -cs) stable" | \
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

sudo apt-get update

sudo apt-get install docker-ce docker-ce-cli containerd.io

sudo docker run hello-world
```

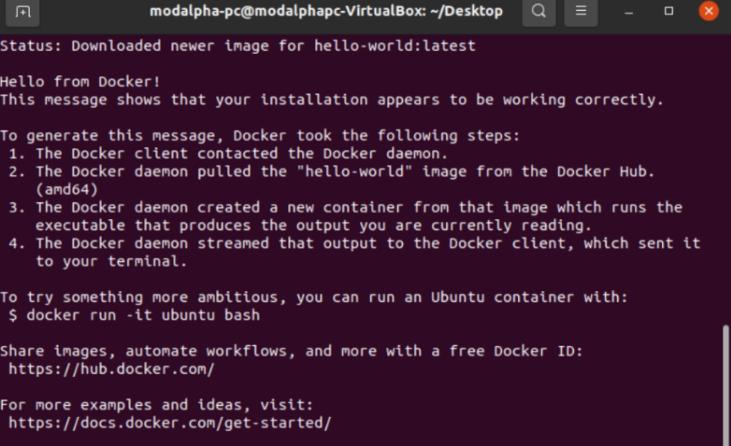
You may run the created script by typing bash `docker_install.sh` as shown in figure 26.



```
pc2@pc2-Precision-Tower-5810:~$ bash docker_install.sh
```

Figure 26: Screenshot of our terminal running docker installation bash script file.

You may confirm the installation was a success if you see the "*Hello World from docker*" as shown in figure 27.



```
modalpha-pc@modalphapc-VirtualBox:~/Desktop
Status: Downloaded newer image for hello-world:latest
Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
   (amd64)
3. The Docker daemon created a new container from that image which runs the
   executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it
   to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/
```

Figure 27: Screenshot of our terminal running docker "Hello World".

Finally we want to configure **docker as a non root user**, we can do it by executing the following commands:

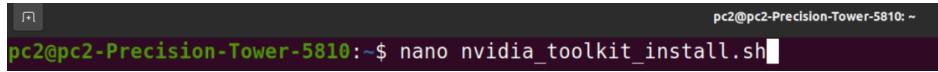
```
#!/bin/bash
sudo groupadd docker
sudo usermod -aG docker $USER
newgrp docker
```

### 1.2.5 AlphaFold Installation

According to [AlphaFold's](#) official website, we have already installed docker.

The next step is to install the **NVIDIA Container Toolkit**.

To install **NVIDIA Container Toolkit** create a bash script file as shown in figure 28.



```
pc2@pc2-Precision-Tower-5810:~$ nano nvidia_toolkit_install.sh
```

Figure 28: Screenshot of our terminal creating nvidia\_toolkit\_install.sh bash script.

Inside the file **nvidia\_toolkit\_install.sh** please copy and paste the following script:

```
#!/bin/bash

distribution=$(./etc/os-release;echo $ID$VERSION_ID) \
&& curl -s -L https://nvidia.github.io/nvidia-docker/gpgkey | sudo apt-key add - \
&& curl -s -L https://nvidia.github.io/nvidia-docker/$distribution/nvidia-docker.list | sudo tee /etc/apt/sources.list.d/nvidia-docker.list

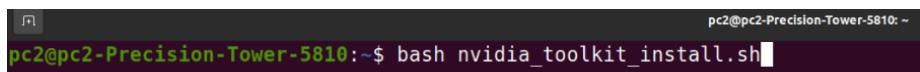
sudo apt-get update

sudo apt-get install -y nvidia-docker2

sudo systemctl restart docker

sudo docker run --rm --gpus all nvidia/cuda:11.0-base nvidia-smi
```

Finally, install **nvidia\_toolkit\_install.sh** by executing the script as shown in figure 29.



```
pc2@pc2-Precision-Tower-5810:~$ bash nvidia_toolkit_install.sh
```

Figure 29: Screenshot of our terminal executing nvidia\_toolkit\_install.sh bash script.

According to [AlphaFold's](#) official website you may run AlphaFold with the instructions shown in figure 30.

Run `run_docker.py` pointing to a FASTA file containing the protein sequence(s) for which you wish to predict the structure (`--fasta_paths` parameter). AlphaFold will search for the available templates before the date specified by the `--max_template_date` parameter; this could be used to avoid certain templates during modeling. `--data_dir` is the directory with downloaded genetic databases and `--output_dir` is the absolute path to the output directory.

```
python3 docker/run_docker.py \
--fasta_paths=your_protein.fasta \
--max_template_date=2022-01-01 \
--data_dir=$DOWNLOAD_DIR \
--output_dir=/home/user/absolute_path_to_the_output_dir
```

Figure 30: Screenshot of AlphaFold's official website. Instructions to run AlphaFold are shown.

### 1.2.6 Modeller Installation (Optional)

At [Modeller's official website](#) you may find the installation guide in case you find it necessary. **The code we have released includes a ready to use installation of Modeller, thus it is not necessary to install Modeller to run AlphaMod.**

### 1.2.7 Phenix Installation (Optional: Required to calculate RMSD)

You may follow [Phenix's Official website](#) for downloading and installing Phenix software.

## 2 Fasta Files

In this section we will describe how to acquire the same CASP14 fasta files we used in our research. These fasta files contain a string of characters representing the amino acid sequence of a protein structure.

### 2.1 CASP14 Target List

At the official [CASP](#) website proceed to the CASP14 [Target List](#) section by clicking as shown in Fig.31.

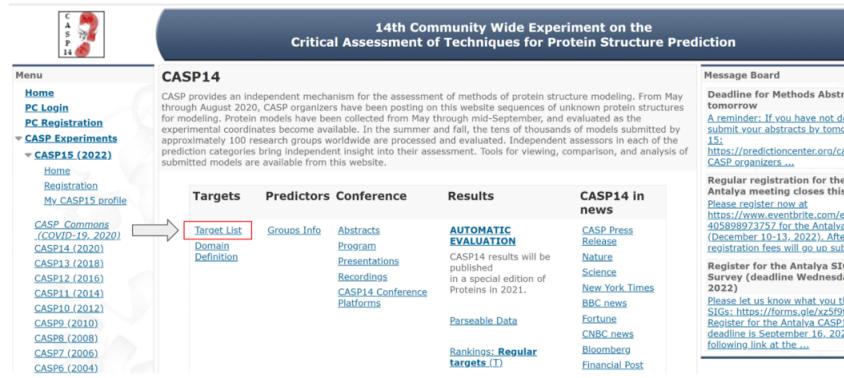


Figure 31: Screenshot of the official Critical Assessment of Techniques for Protein Structure Prediction (CASP), clicking on Target List

## 2.2 Downloading Aminoacid Sequences

At the [Target List](#) page, proceed to select the desired target by clicking as shown in Fig.32

The screenshot shows the 'Target List' page for the 14th Community Wide Experiment on the Critical Assessment of Techniques for Protein Structure Prediction. The page has a header with the experiment name and navigation icons. On the left is a sidebar with a menu and a list of previous experiments. The main content area displays a table of targets. An arrow points to the row for target T1024, which is highlighted with a red border. The table columns include: #, ID, Type, Res, Spec-chim., Entry Date, Server Expiration, QA Expiration, Human Expiration, and Description. The target T1024 is described as 'All groups' with ID A1, entry date 2020-05-18, and server expiration on 2020-05-27. The description notes it is LmrP, PDB code 6t1x.

#	ID	Type	Res	Spec-chim.	Entry Date	Server Expiration	QA Expiration	Human Expiration	Description
1.	<a href="#">T1024</a>	All groups	408	A1	2020-05-18	2020-05-21	m1: 2020-05-25 m2: 2020-05-27	2020-06-08	LmrP PDB code <a href="#">6t1x</a>
2.	<a href="#">T1025</a>	Server only	268	A1	2020-05-19	2020-05-22	m1: 2020-05-24 m2: 2020-05-28	2020-06-09	AtmN PDB code <a href="#">5uzx</a>
3.	<a href="#">T1026</a>	All groups	172	A1	2020-05-19	2020-05-22	m1: 2020-05-24 m2: 2020-05-28	2020-06-09	FBN5V PDB code <a href="#">6z44</a>
4.	<a href="#">T1027</a>	All groups	168	A1	2020-05-20	2020-05-23	m1: 2020-05-27 m2: 2020-05-29	2020-06-10	GLuc PDB code <a href="#">7d2o</a>
5.	<a href="#">T1028</a>	Server only	316	A1	2020-05-21	2020-05-24	m1: 2020-05-28 m2: 2020-05-30	2020-06-11	CalI17 PDB code <a href="#">6ggr</a>

Figure 32: Screenshot of the official Critical Assessment of Techniques for Protein Structure Prediction (CASP), Selecting a Target, specifically T1024, the same approach can be used for the other targets.

At the specific target page (in this example T1024) you will find the **amino acid sequence** and its corresponding **template** as shown in Fig.33.

**Target: T1024**

<b>Target:</b>	T1024
<b>Type:</b>	Human and Server
<b>Entry Date:</b>	2020-05-18
<b>Server Expiration Date:</b>	2020-05-21
<b>Human Expiration Date:</b>	2020-06-08

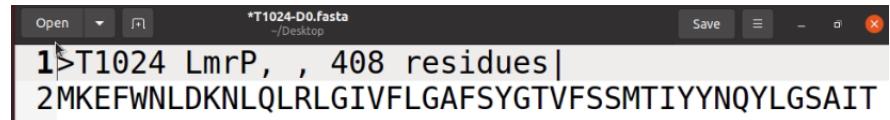
---

<b>Protein:</b>	LmrP
<b>Organism:</b>	
<b>Residues:</b>	408
<b>Method:</b>	X-RAY
<b>CAPRI target:</b>	No
<b>Sequence:</b> ( <a href="#">Plain text version</a> )	<pre>&gt;T1024 LmrP, , 408 residues MKEFWNLDKNLQLRLGIVFLGAFSYGTVFSSMTIYYNQYLGSAITGILLALSAVATFVAGILAGFFADRNGRKPVMVFGTIIQLLGAALP</pre>
<b>Template:</b> ( <a href="#">Plain text version</a> )	<pre>REMARK Template For Target T1024 ATOM      1  N   MET A  1    0.000   0.000   0.000   0.00  0.00  0.00 ATOM      2  CA  MET A  1    0.000   0.000   0.000   0.00  0.00  0.00 ATOM      3  C   MET A  1    0.000   0.000   0.000   0.00  0.00  0.00 ATOM      4  O   MET A  1    0.000   0.000   0.000   0.00  0.00  0.00 ATOM      5  CB  MET A  1    0.000   0.000   0.000   0.00  0.00  0.00 ATOM      6  CG  MET A  1    0.000   0.000   0.000   0.00  0.00  0.00 ATOM      7  SD  MET A  1    0.000   0.000   0.000   0.00  0.00  0.00 ATOM      8  CE  MET A  1    0.000   0.000   0.000   0.00  0.00  0.00 ATOM      9  N   LYS A  2    0.000   0.000   0.000   0.00  0.00  0.00 ATOM     10  CA  LYS A  2    0.000   0.000   0.000   0.00  0.00  0.00 ATOM     11  C   LYS A  2    0.000   0.000   0.000   0.00  0.00  0.00 ATOM     12  O   LYS A  2    0.000   0.000   0.000   0.00  0.00  0.00 ATOM     13  CB  LYS A  2    0.000   0.000   0.000   0.00  0.00  0.00 ATOM     14  CG  LYS A  2    0.000   0.000   0.000   0.00  0.00  0.00 ATOM     15  CD  LYS A  2    0.000   0.000   0.000   0.00  0.00  0.00 ATOM     16  CE  LYS A  2    0.000   0.000   0.000   0.00  0.00  0.00 ATOM     17  NZ  LYS A  2    0.000   0.000   0.000   0.00  0.00  0.00 ATOM     18  N   GLU A  3    0.000   0.000   0.000   0.00  0.00  0.00 ATOM     19  CA  GLU A  3    0.000   0.000   0.000   0.00  0.00  0.00 ATOM     20  C   GLU A  3    0.000   0.000   0.000   0.00  0.00  0.00 ATOM     21  O   GLU A  3    0.000   0.000   0.000   0.00  0.00  0.00 ATOM     22  CB  GLU A  3    0.000   0.000   0.000   0.00  0.00  0.00 ATOM     23  CG  GLU A  3    0.000   0.000   0.000   0.00  0.00  0.00 ATOM     24  CD  GLU A  3    0.000   0.000   0.000   0.00  0.00  0.00</pre>

Figure 33: Screenshot of Target T1024 displaying its sequence and the template, the arrows and red boxes shows that “Plain text version” is a clickable link to its raw version.

## 2.3 Saving as fasta format

After clicking on “plain text version” you will be prompted to the raw version, select the whole amino acid sequence ( CTRL + A), open a text editor of your preference and paste the sequence ( CTRL + V ) as shown in Fig.34, save it with fasta extension as shown in Fig.35.



```
1>T1024 LmrP, , 408 residues|
2MKEFWNLTKNLQLRLGIVFLGAFSYGTVFSSMTIYYNQYLGSAIT
```

Figure 34: Process for acquiring the raw text information of the Target T1024 copy its information and pasting it on a text editor.



Figure 35: Process for saving Target T1024 with fasta extension.

## 2.4 Adjusting FastaFile to comply with AlphaMod guidelines

Open the saved FastaFile, in this example T1024-D0 as shown in figures 36.

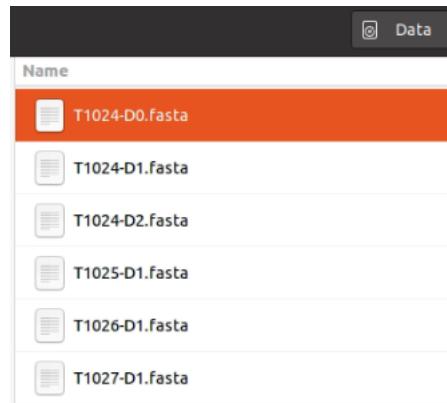


Figure 36: Opening Target T1024-D0.

We want to change the first line shown in figure 37.

```
T1024-D0.fasta
1>T1024 LmrP, , 408 residues
2MKEFWNLKQLRLGIVFLGAFSYGTVFSSMTIYYNQYLGSAI
```

Figure 37: Opening Target T1024-D0.

We want to change the first line with the correct number of residues that were evaluated by the GDT\_TS algorithm, also to put the correct extension (-D0, -D1, -D2 etc) and the word *Domain* all of this separated by commas. To do so, we go to the [CASP14 Results - Automatic Evaluation](#) official website, and we want to look for our protein target, in this case T1024-D0 as shown in figure 38 and click on it.

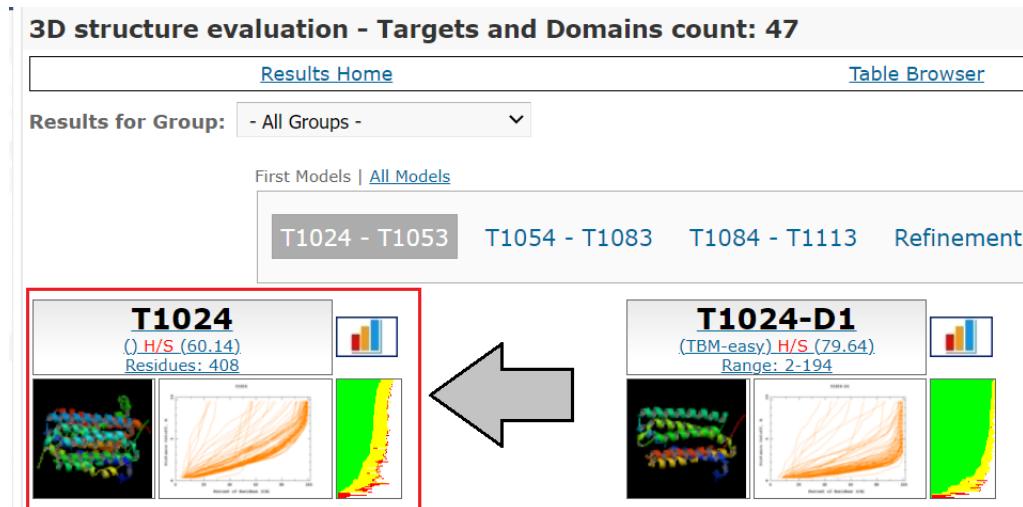
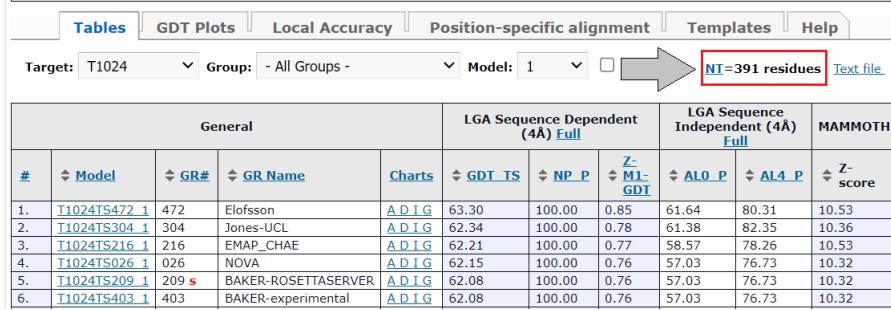


Figure 38: Screenshot of Target T1024-D0 at CASP's official website.

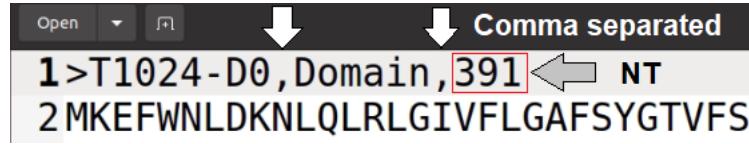
After clicking on the protein you will be prompted a list with the names of the competing groups and their respective scores of different metrics, nevertheless, we are interested in the value of NT as shown in figure 39.



General										LGA Sequence Dependent (4Å) Full		LGA Sequence Independent (4Å) Full		MAMMOTH
#	Model	GR#	GR Name	Charts	GDT_TS	NP_P	Z_M1_GDT	AL0_P	AL4_P	Z-score				
1.	T1024TS472_1	472	Elofsson	ADIG	63.30	100.00	0.85	61.64	80.31	10.53				
2.	T1024TS304_1	304	Jones-UCL	ADIG	62.34	100.00	0.78	61.38	82.35	10.36				
3.	T1024TS216_1	216	EMAP_CHAE	ADIG	62.21	100.00	0.77	58.57	78.26	10.53				
4.	T1024TS026_1	026	NOVA	ADIG	62.15	100.00	0.76	57.03	76.73	10.32				
5.	T1024TS209_1	209_s	BAKER-ROSETTASERVER	ADIG	62.08	100.00	0.76	57.03	76.73	10.32				
6.	T1024TS403_1	403	BAKER-experimental	ADIG	62.08	100.00	0.76	57.03	76.73	10.32				

Figure 39: List of results for Target T1024-D0 at CASP14's official website.

Finally, we update our file as shown in figure 40.



```
Open ↓ ↓ Comma separated
1>T1024-D0,Domain,391 ← NT
2 MKEFWNLKNLQLRLGIVFLGAFSYGTVFS
```

Figure 40: Screenshot of updated Target T1024-D0 fasta file to comply with AlphaMod guidelines.

## 3 AlphaMod

### 3.1 Pre-requisites

At section 1 (requirements) we mentioned that we must have installed:

- Nvidia drivers, CUDA and CuDNN installed in our local machine.
  - *Further information can be found at [NVIDIA's official website](#).*
- AlphaFold.
  - *For installation instructions please refer to [AlphaFold's GitHub](#)*
- Modeller
  - *For installation instructions please refer to [Modeller's official website](#) (Modeller's Installation is optional as our GitHub repository includes Modeller installed already.)*
- Phenix
  - *For installation instructions please refer to [Phenix's official website](#) (Phenix installation is optional, you may install it if you want to calculate RMSD)*

### 3.2 Downloading AlphaMod's GitHub Repository

At [AlphaMod's official repository](#) click on *code*, at *local, SSH* click on the two squares on the right to copy the link as shown on figures 41 and 42.

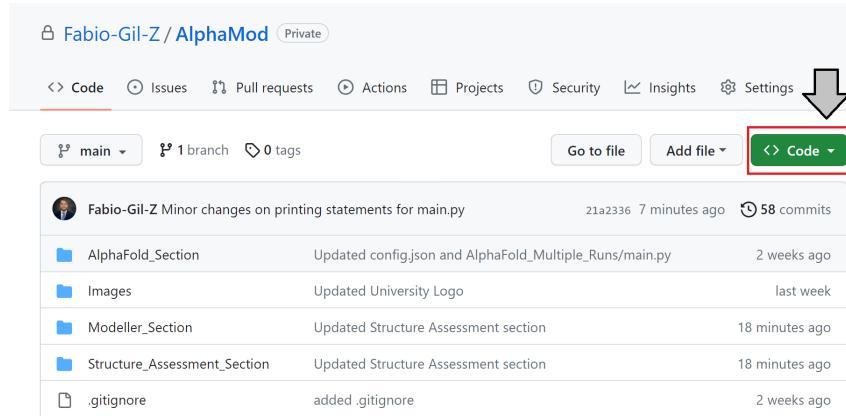


Figure 41: Screenshot of AlphaMod GitHub repository.

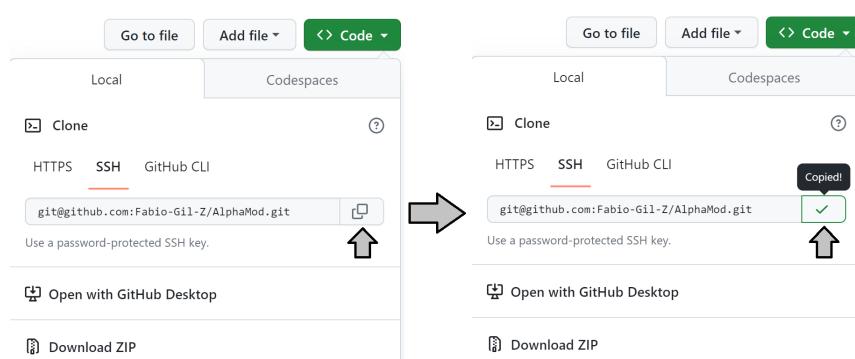


Figure 42: Screenshot of AlphaMod GitHub repository, copying link.

On your computer at the desired destination open a terminal and write:

**git clone git@github.com:Fabio-Gil-Z/AlphaMod.git**

As shown in figure 43.

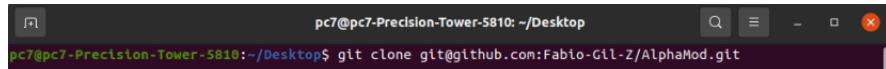


Figure 43: Screenshot of AlphaMod GitHub repository, copying link.

### 3.3 Managing AlphaMod's configuration (config.json)

The file **config.json** is the file that will control *AlphaMod*'s output. An overview of **config.json** can be seen in figure 44.

#### 3.3.1 AlphaFold\_Prediction\_List

This part of the file contains the names of the sequences we want to calculate. Please notice the format, the first sequence is **T1024-D0**, also it is inside quotation marks "T1024-D0" followed by a colon : and then also in quotation marks the number of residues (the **NT** explained at section 2.4). That would be for a single sequence, if we need to calculate multiple sequences then we add a comma (,) and then we proceed to repeat the process with the information of another sequence in figure 44 the second sequence we want to calculate is T1024-D1 (remember -D0 stands for the whole protein, while -D1 stands for a domain of the protein as it is presented on [CASP14 website](#)). Finally, the last sequence we will calculate is T1030-D2, notice it is in the last row and **the last row does not have a comma**.

```

1  {
2      "AlphaFold_Prediction_List": {
3          "T1024-D0": "391",
4          "T1024-D1": "193",
5          "T1024-D2": "204",
6          "T1025-D1": "257",
7          "T1026-D1": "146",
8          "T1027-D1": "99",
9          "T1028-D1": "292",
10         "T1029-D1": "125",
11         "T1030-D0": "273",
12         "T1030-D1": "154",
13         "T1030-D2": "119"
14     },
15     "StructureAssessment": {
16         "pLDDT": "TRUE",
17         "GDT_TS": "TRUE",
18         "DOPESCORE": "TRUE",
19         "QMEAN": "TRUE",
20         "PROSA": "TRUE",
21         "MOLPROBITY": "TRUE",
22         "PROCHECK": "TRUE",
23         "AlphaFold_Models": "TRUE",
24         "2best_supervised": "TRUE",
25         "2best_unsupervised": "TRUE",
26         "5ranked_unsupervised": "TRUE",
27         "first_run_flag": "TRUE"
28     },
29     "Install_Requirements": "TRUE",
30     "AlphaFold_Databases" : "/media/pc7/SeagateBasic/AlphaFold/alphafold_databases"
31 }

```

Figure 44: Screenshot of AlphaMod **config.json** file.

### 3.3.2 StructureAssessment

This part of the file controls two things:

- The metrics you want to calculate such as: pLDDT, GDT\_TS, QMEAN.
- How you want to generate the models, supervised or unsupervised

It is important to note, that certain models require specific metrics to be calculated. For instance, to be able to calculate 2best\_supervised we need the supervised metric GDT\_TS, otherwise would be impossible to calculate it. In a similar way, 2best\_unsupervised requires pLDDT and QMEAN to select the best models, whereas 5ranked\_unsupervised does not need anything as it uses all ranked models given as output by AlphaFold. A table of the required metrics in order to calculate each of the models is shown in figure 45.

	pLDDT	GDT_TS	DOPESCORE	QMEAN	PROSA	MOLPROBITY	PROCHECK
<b>AlphaFold_Models</b>	✗	✗	✗	✗	✗	✗	✗
<b>2best_supervised</b>	✗	✓	✗	✗	✗	✗	✗
<b>2best_unsupervised</b>	✓	✗	✗	✓	✗	✗	✗
<b>5ranked_unsupervised</b>	✗	✗	✗	✗	✗	✗	✗

Figure 45: Metrics required to calculate, 2best\_supervised and 2best\_unsupervised, AlphaFold's models and 5ranked\_unsupervised have no requirements.

Therefore, if we want to calculate only AlphaFold's models we set:

- "AlphaFold\_Models": "TRUE"
- "2best\_supervised": "FALSE"
- "2best\_unsupervised": "FALSE"
- "5ranked\_unsupervised": "FALSE"

If also, we want to calculate AlphaMod's 2best\_supervised, we set:

- "AlphaFold\_Models": "TRUE"
- "2best\_supervised": "TRUE"
- "2best\_unsupervised": "FALSE"
- "5ranked\_unsupervised": "FALSE"

Nevertheless, according to figure 45 to be able to calculate 2best\_supervised we need to also activate GDT\_TS. So we set, GDT\_TS to "TRUE" as shown in figure 46.

```

1 ▼ {
2 ▼   "AlphaFold_Prediction_List": {
3     "T1024-D0": "391",
4     "T1024-D1": "193",
5     "T1024-D2": "204",
6     "T1025-D1": "257",
7     "T1026-D1": "146",
8     "T1027-D1": "99",
9     "T1028-D1": "292",
10    "T1029-D1": "125",
11    "T1030-D0": "273",
12    "T1030-D1": "154",
13    "T1030-D2": "119"
14  },
15 ▼   "StructureAssessment": {
16     "pLDDT": "TRUE",
17     "GDT_TS": "TRUE", →
18     "DOPESCORE": "TRUE",
19     "QMEAN": "TRUE",
20     "PROSA": "TRUE",
21     "MOLPROBITY": "TRUE",
22     "PROCHECK": "TRUE",
23     "AlphaFold Models": "TRUE",
24     "2best_supervised": "TRUE", →
25     "2best_unsupervised": "TRUE",
26     "5ranked_unsupervised": "TRUE",
27     "first_run_flag": "TRUE"
28  },
29  "Install_Requirements": "TRUE",
30  "AlphaFold_Databases" : "/media/pc7/SeagateBasic/AlphaFold/alphafold_databases"
31 }

```

Figure 46: Screenshot of config.json configuration. AlphaFold's and AlphaMod (2best\_supervised) will be calculated. The metric GDT\_TS is set to "TRUE" as it is required to calculate supervised models.

Finnaly, according to figure 45 to calculate 2best\_unsupervised we need pLDDT and QMEAN, the configuration is shown in figure 47.

```

1 v {
2 ▼   "AlphaFold_Prediction_List": {
3     "T1024-D0": "391",
4     "T1024-D1": "193",
5     "T1024-D2": "204",
6     "T1025-D1": "257",
7     "T1026-D1": "146",
8     "T1027-D1": "99",
9     "T1028-D1": "292",
10    "T1029-D1": "125",
11    "T1030-D0": "273",
12    "T1030-D1": "154",
13    "T1030-D2": "119"
14  },
15 ▼   "StructureAssessment": {
16     ➔ "pLDDT": "TRUE",
17     ➔ "GDT_TS": "TRUE",
18     ➔ "DOPESCORE": "TRUE",
19     ➔ "QMEAN": "TRUE",
20     ➔ "PROSA": "TRUE",
21     ➔ "MOLPROBITY": "TRUE",
22     ➔ "PROCHECK": "TRUE",
23     ➔ "AlphaFold_Models": "TRUE",
24     ➔ "2best supervised": "TRUE",
25     ➔ "2best_unsupervised": "TRUE",
26     ➔ "5ranked_unsupervised": "TRUE",
27     ➔ "first_run_flag": "TRUE"
28  },
29  "Install_Requirements": "TRUE",
30  "AlphaFold_Databases" : "/media/pc7/SeagateBasic/AlphaFold/alphafold_databases"
31 }

```

Figure 47: Screenshot of config.json configuration. AlphaFold's, AlphaMod (2best\_supervised) and AlphaMod (2best\_unsupervised) will be calculated. The metrics pLDDT and QMEAN are set to "TRUE" as it is required to calculate unsupervised models.

The other unsupervised metrics:

- DOPESCORE
- PROSA
- MOLPROBITY
- PROCHECK

These metrics are optional and not required to calculate the models. They are used to provide additional information about the predicted models.

The row *first\_run\_flag* is used as flow control inside the program and it is recommended to leave it as "TRUE" if it is the first time you are running the program. In general do not change it.

The row "*Install\_Requirements*" reads *requirements.txt* text file and installs all the packages needed via pip if "TRUE".

Finally, the row "*AlphaFold\_Databases*" is asking for the AF2 databases path, these are the databases downloaded according to [AlphaFold's](#) official website. In our case we stored the databases in a folder called "alphafolddatabases".

### 3.4 Launching AlphaMod

Once **config.json** has been configured according to sections 3.3.1 and 3.3.2, we are ready to launch AlphaMod.

First, open a terminal at the top folder (AlphaMod). Followed by the command **python3 main.py** as shown in figure 48. In our example, our base folder is located at */Desktop/AlphaMod*.



```
Terminal: Local + 
(venv) pc7@pc7-Precision-Tower-5810:~/Desktop/AlphaMod$ python3 main.py
```

Figure 48: Screenshot of our terminal initializing AlphaMod.

The configuration we will use in this example is shown in figure 49. Additionally, if it is the first time you are launching AlphaMod you will be prompted with the installation of the required packages.

```
(venv) pc7@pc7-Precision-Tower-5810:/media/pc7/Data/AlphaMod$ python3 main.py
#####
###           CONFIGURATION          ###
#####
{
    "AlphaFold_Prediction_List": {
        "T1024-D0": "391",
        "T1024-D1": "193",
        "T1024-D2": "204",
        "T1025-D1": "257",
        "T1026-D1": "146",
        "T1027-D1": "99",
        "T1028-D1": "292",
        "T1029-D1": "125",
        "T1030-D0": "273",
        "T1030-D1": "154",
        "T1030-D2": "119"
    },
    "StructureAssessment": {
        "pLDDT": "TRUE",
        "GDT_TS": "TRUE",
        "DOPESCORE": "TRUE",
        "QMEAN": "TRUE",
        "PROSA": "TRUE",
        "MOLPROBITY": "TRUE",
        "PROCHECK": "TRUE",
        "AlphaFold_Models": "TRUE",
        "2best_supervised": "TRUE",
        "2best_unsupervised": "TRUE",
        "5ranked_unsupervised": "TRUE",
        "first_run_flag": "TRUE"
    },
    "Install_Requirements": "TRUE",
    "AlphaFold_Databases": "/media/pc7/SeagateBasic/AlphaFold/alphafold_databases"
}
First time launching AlphaMod, requirements will be installed...
Please press Enter...
```

Figure 49: Screenshot of our terminal executing AlphaMod for the first time.

Once all the packages have been installed you will be given a message that all the requirements have been installed as shown in figure 50.

```
Successfully installed Keras-Preprocessing-1.1.2 MarkupSafe-2.1.1 Pillow-9.4.0 PySocks-1.7.1 PyYAML-6.0 Sel
sl-py-0.13.0 astunparse-1.6.3 async-generator-1.10 attrs-22.2.0 biopython-1.79 cachetools-5.2.1 certifi-2022.12.7 charset-
21.6.0 dm-haiku-0.0.4 dm-tree-0.1.6 docker-5.0.0 exceptiongroup-1.1.0 flatbuffers-1.12 gast-0.4.0 google-auth-2.15.0 googl
grpcio-1.34.1 h11-0.14.0 h5py-3.1.0 idna-3.4 immutabledict-2.0.0 importlib-metadata-6.0.0 jax-0.2.14 jaxlib-0.3.15 keras-
ns-0.1.0 natsort-8.2.0 numpy-1.19.5 oauthlib-3.2.2 opt-einsum-3.3.0 outcome-1.2.0 pandas-1.3.4 protobuf-3.20.3 pyasn1-0.4.
ython-dateutil-2.8.2 pytz-2022.7 requests-2.28.1 requests-oauthlib-1.3.1 rsa-4.9 scipy-1.7.0 selenium-4.7.2 six-1.15.0 sni
te-0.9.0 tensorflow-2.11.0 tensorflow-data-server-0.6.1 tensorflow-plugin-wit-1.8.1 tensorflow-cpu-2.5.0 tensorflow-est
0 trio-0.22.0 trio-websocket-0.9.2 typing-extensions-3.7.4.3 urllib3-1.26.13 websocket-client-1.4.2 wrapt-1.12.1 wsproto-1

#####
### Requirements have been installed, press enter to continue, AlphaFold will be launched #####
#####
```

Figure 50: Screenshot of our terminal at the end of the installation of packages.

Moreover, if you are going to use *2best\_supervised* it is necessary to set GDT\_TS as "TRUE" and thus we will need to calculate the GDT\_TS score. The scripts necessary to calculate the GDT\_TS score needs administrator rights for execution. You will be informed which files are requesting this permission and it's location as shown in figure 51. Once you have entered your password AlphaMod will proceed to launch AlphaFold.

```
#####
### To be able to calculate the GDT_TS Score #####
### we need to give permission to Zemla's scripts #####
### please enter your 'sudo' password #####
### if no password is asked, is because #####
### you already gave the permission #####
### and this message can be ignored #####
### The files we are giving permission are: #####
### AlphaMod/Structure_Assessment_Section/GDT_TS/Zemla_GDT_TS_admin_permissions.sh #####
### AlphaMod/Structure_Assessment_Section/GDT_TS/superposition_and_GDT_TS.sh #####
### AlphaMod/Structure_Assessment_Section/GDT_TS/LGA_Zemla/lga #####
### AlphaMod/Structure_Assessment_Section/GDT_TS/LGA_Zemla/MOL2/collect_PDB.pl #####
### #####
[sudo] password for pc7:
```

Figure 51: Screenshot of our terminal entering administrator password for GDT\_TS scripts.

After giving administrator rights to Zemlas's scripts AlphaFold will be launched as shown in figure 52.

```

##### AlphaFold Parameters #####
fasta_paths: /media/pc7/Data/AlphaMod/AlphaFold_Section/AlphaFold_Files/Fasta_Files
max_template_date: 2020-05-14
data_dir: /media/pc7/SeagateBasic/AlphaFold/alphafold_databases
output_dir: /media/pc7/Data/AlphaMod/AlphaFold_Section/AlphaFold_Files/results
##### AlphaFold Parameters #####
##### RUNNING T1024-D0.fasta -#####
File location: /media/pc7/Data/AlphaMod/AlphaFold_Section/AlphaFold_Files/Fasta_Files/T1024-D0.fasta
run_docker.py:113] Mounting /media/pc7/Data/AlphaMod/AlphaFold_Section/AlphaFold_Files/Fasta_Files -> /mnt/fasta_path_0
run_docker.py:113] Mounting /media/pc7/SeagateBasic/AlphaFold/alphafold_databases/uniref90 -> /mnt/uniref90_database_path
run_docker.py:113] Mounting /media/pc7/SeagateBasic/AlphaFold/alphafold_databases/mgnify -> /mnt/mgnify_database_path
run_docker.py:113] Mounting /media/pc7/SeagateBasic/AlphaFold/alphafold_databases -> /mnt/data_dir
run_docker.py:113] Mounting /media/pc7/SeagateBasic/AlphaFold/alphafold_databases/pdb_mmcif/mmcif_files -> /mnt/template_mmc
run_docker.py:113] Mounting /media/pc7/SeagateBasic/AlphaFold/alphafold_databases/pdb_mmcif -> /mnt/obsolete_pdbs_path
run_docker.py:113] Mounting /media/pc7/SeagateBasic/AlphaFold/alphafold_databases/pdb70 -> /mnt/pdb70_database_path
run_docker.py:113] Mounting /media/pc7/SeagateBasic/AlphaFold/alphafold_databases/uniclust30/uniclust30_2018_08 -> /mnt/unic
run_docker.py:113] Mounting /media/pc7/SeagateBasic/AlphaFold/alphafold_databases/bfd -> /mnt/bfd_database_path
run_docker.py:255] I0426 12:19:38.123532 139965912745792 templates.py:857] Using precomputed obsolete pdbs /mnt/obsolete_pdb

```

Figure 52: Screenshot of our terminal running AlphaFold.

After AlphaFold2 has finished doing the prediction of the protein, the next step is to assess the quality of the predicted structure, be it on a supervised or unsupervised manner. In our example we selected *2best\_supervised*. Therefore, according to figure 45 we need the GDT\_TS score to calculate supervised models, the calculation of the GDT\_TS score is shown in figure 53.

```

#####
###      Calculating Superposition and GDT_TS of T1024-D0      ###
#####
Directory: /home/pc7/Desktop/AlphaMod/Structure_Assessment_Section,
Directory: /home/pc7/Desktop/AlphaMod/Structure_Assessment_Section,
superposition_and_GDT_TS of ranked_0.pdb finished
superposition_and_GDT_TS of ranked_1.pdb finished
superposition_and_GDT_TS of ranked_2.pdb finished
superposition_and_GDT_TS of ranked_3.pdb finished
superposition_and_GDT_TS of ranked_4.pdb finished

```

Figure 53: Screenshot of our terminal calculating the GDT\_TS score for Target T1024-D0.

In our example we have also selected *2best\_unsupervised*, for the unsupervised route we need to extract the *pLDDT* values and calculate *QMEAN* score. The pLDDT extraction is done in the background while the QMEAN score calculation is shown in figure 54.

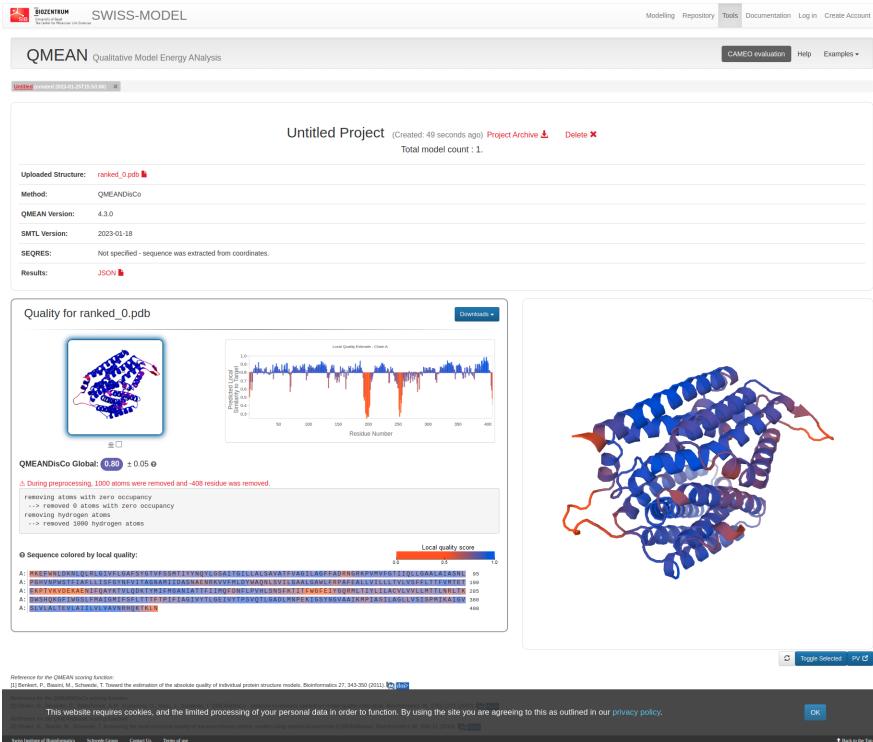


Figure 54: Screenshot of our terminal running python Ad-Hoc web-crawler. Shows QMEAN calculation for Target T1024, AF2 structure ranked\_0.pdb .