MASSIVEFOLD

Practical work

Monomer

IFB training - AlphaFold and beyond - 10th to 12th December 2025

AlphaFold3 weights

Copy your AlphaFold3 weights and create links to database in your home:

- create a ‘af3_datadir’ directory in your home and copy your weights there (af3.bin.zst)
- in this directory, create symbolic links to the AlphaFold3 database:

```
ln -s /lustre/fsmisc/dataset/Alphafold3/* ./
```

Install MassiveFold

MassiveFold works with 4 conda environments already installed on Jean Zay.

- Load MassiveFold 1.5

```
module load massivefold/1.5.0
```

N.B.: all MassiveFold 1.5.x versions work with the same conda environments (v1.5.0 on Jean Zay, for instance), you can see the environments used by massiveFold (mf-***) with `conda env list`

To avoid any problem of storage and inode, navigate to the scratch storage $CCFRSCRATCH

```
cd $CCFRSCRATCH
```

But the scripts have to be installed:

- Clone the git and install for Jean Zay

```
git clone https://github.com/GBLille/MassiveFold.git
cd MassiveFold
git checkout v1.5.6
./install.sh
```

Don’t hesitate to visualize the help for the scripts (bash or python) :

```
./install.sh --help
```

Architecture

```
MassiveFold
├── install.sh          => to setup massivefold pipeline (create the files to run massivefold)
├── ...
├── examples
├── massivefold         => contains the scripts for MassiveFold (inc gathers_runs.py & massivefold_plots.py)
└── massivefold_runs    => where to run MassiveFold
    ├── AFmassive_params.json       => parameter file for AFmassive runs
    ├── ColabFold_params.json       => parameter file for ColabFold runs
    ├── AlphaFold3_params.json      => parameter file for AlphaFold3 runs
    ├── headers/
        ├── alignment.slurm         => SLURM header for the alignments (on CPU nodes)
        ├── jobarray.slurm          => SLURM header for prediction batches (on GPU nodes)
        └── post_treatment.slurm    => SLURM headers for post-treatment
    ├── input/                => fasta files
    ├── log/            => log files for each job (useful to debug in case of crashes)
    ├── output/               => output files, organized in subfolders 1) fasta file name, 2) run name
    └── run_massivefold.sh    => to run MassiveFold (check ./run_masivefold.sh -h)
```

Prepare files 1/6

A0A2U7UDN4 is an unknown viral protein for which we are going to model the structure with AlphaFold2 (AFmassive), ColabFold and AFmassive, then choose the best prediction and try to identify a specific fold.

- Copy the fasta files from '$ALL_CCFRWORK/input' to your input directory

```
rsync -avx $ALL_CCFRWORK/training_Alphafold_202512/input/* $CCFRSCRATCH/MassiveFold/massivefold_runs/input/
```

It includes the sequence of the viral protein A0A2U7UDN4.fasta

```
>tr|A0A2U7UDN4|A0A2U7UDN4_9VIRU DUF5848 domain-containing protein OS=Pandoravirus neocaledonia OX=2107708 GN=pneo_cds_858 PE=4 SV=1
MRCAPGPFRAAMSAGASILTGPDCRLLIVGLYTSDSHGPLAADRQCTTRTSWWPKDVAQW
QSVSAAAAQCLLRNGLVLGTAKGSRDASLLGPVMYDAMGRTMEAADARLIGVAMATAPVV
PCYGGAETRATSDVTVVDKARTVHTFATDETADEWLAFCRLVALRDATLWRALSVDPSAY
GERGAHVATSILRSEEAIARRRRFDPALVDADEIAALRRAFVAAADGHRLATPLDLGF
```

- To avoid the Multiple Sequence Alignment (MSA) step, which may last several hours for some sequences, copy also the content of the '$ALL_CCFRWORK/training_Alphafold_202512/msas/' folder to your 'output' folder:

```
rsync -avx $ALL_CCFRWORK/training_Alphafold_202512/msas/* $CCFRSCRATCH/MassiveFold/massivefold_runs/output/
```

They are also available here (all_msas.zip):
https://nextcloud.univ-lille.fr/index.php/s/dnQLmBywbcQdx56

Prepare files 2/6

- Edit the ‘AFmassive_params.json’ file and check the parameters.

  In the “**massivefold**” section, modify the pkl format to “light”, let “models_to_use” to “”.

```
"massivefold": {
        "run_massivefold": "run_AFmassive.py",
        "run_massivefold_plots": "../massivefold/massivefold_plots.py",
        "data_dir": "$DSDIR/Alphafold-2024-04",
        "uniref_database": "",
        "jobfile_headers_dir": "./headers",
        "jobfile_templates_dir": "../massivefold/parallelization/templates",
        "output_dir": "./output",
        "logs_dir": "./log",
        "input_dir": "./input",
        "scripts_dir": "../massivefold/parallelization",
        "models_to_use": "",
        "pkl_format": "light"
    },
```

  pkl files can also be lighten later with the ‘lighten_output.py’ script

```
massivefold/parallelization/lighten_output.py --help
```

Prepare files 3/6

- Edit the ‘AFmassive_params.json’ file and check the parameters.

  In the “**custom_params**” section, set your project ID and the gpu to “h100”.

  Set "jeanzay_jobarray_time" to 1:00:00.

```
"custom_params": {
        "jeanzay_gpu": "h100",
        "jeanzay_project": "<project>",
        "jeanzay_account": "<project>@h100",
        "jeanzay_gpu_with_memory": "h100",
        "jeanzay_alignment_time": "10:00:00",
        "jeanzay_jobarray_time": "1:00:00"
    },
```

Prepare files 4/6

- Edit the ‘AFmassive_params.json’ file and check the parameters.

  In the “**AFM_runs**” section, set "model_preset" to “monomer_ptm”.

```
"AFM_run": {
        "db_preset": "full_dbs",
        "dropout": "false",
        "dropout_structure_module": "false",
        "dropout_rates_filename": "",
        "max_recycles": "20",
        "early_stop_tolerance": "0.5",
        "max_template_date": "2024-08-31",
        "bfd_max_hits": "100000",
        "mgnify_max_hits": "501",
        "uniprot_max_hits": "50000",
        "uniref_max_hits": "10000",
        "model_preset": "monomer_ptm",
        "templates": "true",
        "stop_recycling_below": "0",
        "min_score": "0",
        "max_score": "1"
},
```

Prepare files 5/6

- Edit the ‘AFmassive_params.json’ file and check the parameters.

  In the “**plots**” section, remove “recycles” (not available for monomers):

```
"plots": {
        "MF_plots_top_n_predictions": "25",
        "MF_plots_chosen_plots": "coverage,DM_plddt_PAE,CF_PAEs,score_distribution"
    }
```

- More plots can be generated later if necessary with the ‘massivefold_plots.py’ script:

```
massivefold/massivefold_plots.py --help
```

Prepare files 6/6

- Because we are going to run the jobs on H100 GPUs, edit the ‘jobarray.slurm’ header file in the ‘massivefold_runs/headers” folder and uncomment the line related to H100

```
# to run on H100:
module purge
module load arch/h100
```

N.B.: if you had to run on A100, you should uncomment these lines and let the H100 ones commented

```
# to run on A100:
module purge
module load arch/a100
```

To run on V100, let them all commented.

Run MassiveFold

- Run MassiveFold with AFmassive

```
./run_massivefold.sh -s input/A0A2U7UDN4.fasta -r afm_default -p 5 -b 5 -f AFmassive_params.json
```

- Show the running jobs

```
squeue --me -o "%.18i %.9P %.60j %.8u %.2t %.10M %.6D %R %.8Q %.10l %.10P" --sort=-p
```

- In case of crashes

```
scancel -u <user>
```

or

```
scancel <jobid>
```

and check the logs in

```
massivefold_runs/log/<fasta_name>/<run_name>
```

Same work for ColabFold

- Edit the ‘ColabFold_params.json’ file and check the parameters.
- Set “pkl_format” to “light”
- Set the “custom_params”
- In the “**CF_run**” params, set “model_preset” to “monomer_ptm”

```
"CF_run": {
        "model_preset": "monomer_ptm",
        "use_dropout": "false",
        "num_recycle": "20",
        "recycle_early_stop_tolerance": "0.5",
        "stop_at_score": "100",
        "disable_cluster_profile": "false",
        "pair_strategy": "greedy"
    },
```

- In “plots”, remove “recycles”
- Run MassiveFold with ColabFold:

```
./run_massivefold.sh -s input/A0A2U7UDN4.fasta -r cf_default -p 5 -b 5 -f ColabFold_params.json
```

Same work for AlphaFold3

- Edit the ‘AlphaFold3_params.json’ file and check the parameters.
- Set “pkl_format” to “light”
- Set the “custom_params”
- In the “**AF3_run**” params:
  - in “fasta_chains”, set the chain type (here “protein”) the same number of time as the number of chains in the fasta file (here one time for a monomer)
  - set “model_preset” to “monomer_ptm”

```
"AF3_run": {
        "fasta_chains": ["protein"],
        "ligand": [
            {"ccdCodes": [""], "smiles": ""}
        ],
        "modifications": [
            [{"type": "", "sequence": "", "positions": []}]
        ],
        "model_preset": "monomer_ptm",
        "max_template_date": "2024-11-28",
        "num_diffusion_samples": "5",
        "unpairedMsa": "true",
        "pairedMsa": "true",
        "templates": "true"
    },
```

Same work for AlphaFold3

Then run MassiveFold with AlphaFold3:

```
./run_massivefold.sh -s input/A0A2U7UDN4.fasta -r af3_default -p 5 -b 5 -f AlphaFold3_params.json
```

N.B.: Only the plddt and PAE plots are generated for AF3. The number can be set to 25.

Results

- Retrieve the results from the cluster to your local computer to visualize them
- Alternatively, you can get them here

  https://nextcloud.univ-lille.fr/index.php/s/dnQLmBywbcQdx56

N.B.: it is possible to relax a structure or a set of structures with ‘colabfold_relax’ available in the ‘mf-colabfold-1.5.5’ conda environment

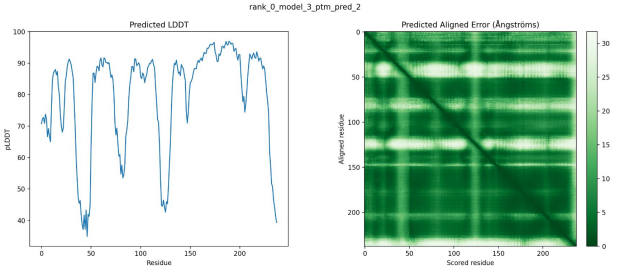
```
conda activate mf-colabfold-1.5.5
colabfold_relax --help
```

Results

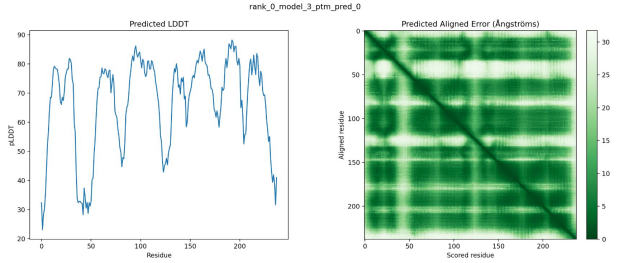
Compare the results for the 3 tools, looking at the ‘ranking_debug.json’, ‘ranking_ptm.json’ and the plots.

=> Which tool seems to give the best predictions ?

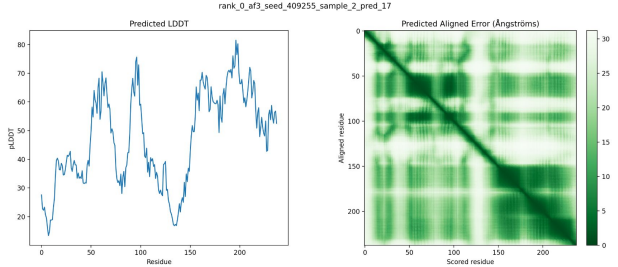
```
ranking confidence: 79.37
ranking ptm: 0.81
```




```
ranking confidence: 66.63
ranking ptm: 0.71
```




```
ranking confidence: 0.50
ranking ptm: 0.39
```



Foldseek

With the goal to identify a well known fold, take the most confident prediction and send it to FoldSeek, both using 3Di/AA or TM-align.

https://search.foldseek.com/search

Using in particular CATH50 and PDB100,
Which fold do you find ?

Find a PDB where this fold exists.