Structure prediction with deep-learning-based methods AlphaFold2, RoseTTAFold and ColabFold

Sevilay Gulesen, Hannes Junker, Eli McDonald and Hope Woods, PhD

Rosetta Workshop, July 22nd 2024, Vanderbilt University

Introduction to the tutorial

In this session, we will be predicting the structures of monomeric and multimeric proteins in four different cases. The first scenario is the prediction of a short peptide structure that is stapled by multiple disulfide bonds. The second scenario is the prediction of a monotopic protein structure with no close homologs to see the performance of different deep-learning methods in challenging cases. The third scenario uses the protein from the second one to assess the effects of point mutations on protein structure. The final scenario is calculation of protein complexes using a dimeric peptide as the example.

Case 1 - Simple structure prediction: Modeling of small proteins or peptides

Introduction

This first case was selected because the short sequence length makes calculations tenable within the timeframe of the tutorials. You will be introduced to the scripts used for AlphaFold2 (AF2)1 and RoseTTAFold (RF)2 calculations in the following steps as well. We will be predicting the structure of a cysteine-knot peptide with the PDB ID 2M2Q using AF2 and RF, followed by some analyses to set the foundation for more complex cases we will run in the following sections. Let's begin.

Understanding the scripts

AlphaFold2 Today's AF2 script contains a number of variables for the bash scripting language and AF2 itself. This script can be found in `ml_folding/AF2/2M2Q` directory as `run_alphafold_ACCRE_Turing.sb`.

Navigate to the directory containing the script:

```
cd ml_folding/AF2/2M2Q
```

Open the script with your preferred text editor. For gedit you can use the following command:

```
gedit run_alphafold_ACCRE_Turing.sb
```

Let's take a look at these commands, starting with the bash variables first:

```
# Set your input/output data path
CALCDIR='.'

# Your input fasta should be in the directory above:
FASTA=2M2Q.fasta

# Where is the AF2 miniconda environment
AF2_MINICONDA=/sb/apps/alphafold232/miniconda3

# Where is the AF2 Inference data
AF2_DATADIR=/sb/apps/alphafold-data.230

# Where is the AF2 Git?
AF2_REPO=/sb/apps/alphafold232/alphafold
```

What are these variables? `CALCDIR` points to the directory in which the calculations are going to be run. This is the directory you are currently working at. You can use the `pwd` command to print the folder location and replace the `CALCDIR` directory with that value. `FASTA` points to your fasta file. `AF2_MINICONDA` is the location of the miniconda environment for the AF2 calculations, `AF2_DATADIR` is the location of the AF2 parameter and database folders, and `AF2_REPO` is the location where the AF2 executables are. You should not touch these values. Next, we have AF2 specific keywords. The example below is for an AF2 run that involves the calculation of MSA values:

```
python $AF2_REPO/run_alphafold.py \
    --fasta_paths=$FASTA \
    --max_template_date=9999-12-31 \
    --data_dir=$AF2_DATADIR \
    --output_dir=$CALCDIR \
    --use_gpu_relax \
    --uniref90_database_path=$AF2_DATADIR/uniref90/uniref90.fasta \
    --mgnify_database_path=$AF2_DATADIR/mgnify/mgy_clusters_2022_05.fa \
    --uniref30_database_path=$AF2_DATADIR/uniref30/UniRef30_2021_03 \
    --bfd_database_path=$AF2_DATADIR/bfd/bfd_metaclust_clu_complete_id30_c90_final_seq.sorted_opt \
    --pdb70_database_path=$AF2_DATADIR/pdb70/pdb70 \
    --template_mmcif_dir=$AF2_DATADIR/pdb_mmcif/mmcif_files \
    --obsolete_pdbs_path=$AF2_DATADIR/pdb_mmcif/obsolete.dat
```

Some of these keywords are self-explanatory in the light of the variables explained in the section above. However, some keywords may look unfamiliar as they are part of the AF2 terminology, so let's go over those keywords. `--use_gpu_relax` tells AF2 to run the relax calculation following the structure prediction using GPUs rather than CPUS. `--uniref90_database_path`, `--mgnify_database_path`, `--uniref30_database_path`, `--bfd_database_path`, `--pdb70_database_path`, `--template_mmcif_dir`, and `--obsolete_pdbs_path` are all related to the MSA and template selection steps. In summary, AF2 uses all the available structures and the corresponding sequences in a number of databases to predict first contacts between the different backbone groups. MSA generation and template selection are parts of this procedure. Therefore, all of these keywords show AF2 the location of the different sequence sources and template structure databases. Normally, these all point to a location inside the `AF2_DATADIR`.

In today's example, we will use a slightly modified version of this script. Ours has the additional keyword `--use_precomputed_msas` added to the end of the script. What is the meaning of this? AF2 normally uses sequence alignments to build contacts as just mentioned. This step, however, can be time consuming even for shorter sequences like ours, at least too long for the purposes of this tutorial. Therefore, we ran the AF2 calculation before the class using the first script provided to generate all the MSA alignments. All you need to do is to make sure that the `2M2Q` folder is in the same directory as the slurm script you are going to run. Take a look at the `2M2Q/msas` folder.

You should be still be in the `AF2/2M2Q` directory. You can use the ls command to list the contents of a directory. Run the following command to check what is in the `2M2Q/msas` folder:

```
ls 2M2Q/msas
```

What do you see? Check the content of these files and try to understand which file may be relevant for what purpose.

For example, you can use gedit to open `pdb_hits.hhr` with the following command:

```
gedit 2M2Q/msas/pdb_hits.hhr
```

AF2, once provided with the `--use_precomputed_msas` command, will look for pre-generated MSA and template files and will automatically skip the generation step.

RoseTTAFold The RF script provided by the developers allow end-to-end prediction without the necessity to do significant changes by the users. In today's examples, we will use a slightly modified version of the script that basically allows running it outside the RoseTTAFold folders and with the conda environment that is at a different location than the canonical anaconda/miniconda installation on the machine.

Check your current directory with the `pwd` command. If you are still in the `ml_folding/AF2/2M2Q` directory, you can navigate back to `ml_folding` directory with:

```
cd ../../
```

Now, navigate to the `RoseTTAFold/2M2Q` directory and list the contents to find the RF script.

```
cd RoseTTAFold/2M2Q/
```

```
ls
```

You should see `run_e2e_ver_adapted.sh` in this directory. Open it with your preferred text editor.

```
gedit run_e2e_ver_adatped.sh
```

The parts of the script that are relevant for our purposes are the following:

```
if [! -s $WDIR/t000_.hhr]
then
    echo "Running hhsearch"
    HH="hhsearch -b 50 -B 500 -z 50 -Z 500 -mact 0.05 -cpu $CPU -maxmem $MEM -aliw 100000 -e 100 -p 5.
    cat $WDIR/t000_.ss2 $WDIR/t000_.msa0.a3m > $WDIR/t000_.msa0.ss2.a3m
    $HH -i $WDIR/t000_.msa0.ss2.a3m -o $WDIR/t000_.hhr -atab $WDIR/t000_.atab -v 0 > $WDIR/log/hhsearc
fi
```

This part of the script runs the HHblits search (https://pubmed.ncbi.nlm.nih.gov/15531603 that is necessary for the actual RF run through the hhsearch command (https://pubmed.ncbi.nlm.nih.gov/15531603/). The CPU, working directory `$WDIR`, and memory requirements `$MEM` are based on preset values defined previously in the script (not shown here). Next, the prediction is run by the following lines:

```
if [! -s $WDIR/t000_.3track.npz]
then
    echo "Running end-to-end prediction"
    python $PIPEDIR/network/predict_e2e.py \
    -m $PIPEDIR/weights \
    -i $WDIR/t000_.msa0.a3m \
    -o $WDIR/t000_.e2e \
    --hhr $WDIR/t000_.hhr \
    --atab $WDIR/t000_.atab \
    --db $DB 1> $WDIR/log/network.stdout 2> $WDIR/log/network.stderr
fi

echo "Done"
```

The code above basically runs the `predict_e2e.py` script that does the actual prediction by defining the weights directory, MSA files, and writes a number of outputs for the user. The `$PIPEDIR` parameter in this script stands for the RF installation folder, and the `$WDIR` stands for the current working directory. We will check the output files individually in the following sections.

Running the calculations

Go to the PDB (https://www.rcsb.org/), search for 2m2q, and download the fasta file corresponding to the 2M2Q structure, which is the model protein we are going to use for today's exercise. Once you open the 2M2Q page, you can select the "Download Files>FASTA Sequence" option to download the fasta file. This is the only input we are going to need, because the MSAs and the other necessary prediction files are generated internally by AF2 and RF.

AlphaFold2 Navigate back to the `AF2/2M2Q` directory we were in before.

```
cd ../../AF2/2M2Q
```

Move the fasta file you downloaded from the PDB to your working directory:

```
mv ~/Downloads/rcsb_pdb_2M2Q.fasta 2M2Q.fasta
```

We will be running AF on GPUs housed by ACCRE, our computing center on campus. ACCRE uses a job scheduler called slurm, which tracks and manages compute and memory resources to schedule submitted jobs. Therefore, we will be using slurm script to submit our AF jobs to the ACCRE GPUs. Lines that begin with #SBATCH are specific for slurm scripts and tell the scheduler details about resources you are requesting. You should not need to change any of these lines.

Open the slurm script in your preferred text editor.

```
gedit run_alphafold_ACCRE_Turing.sb
```

Check the variables discussed earlier to make sure that your slurm script `run_alphafold_ACCRE_Turing.sb` has all the right parameters (i.e. fasta name, working directory, etc.). Also, make sure you have the slurm script, the `2M2Q.fasta` file, and the `2M2Q` folder containing the msas in the same directory.

Unfortunately, once you log into ACCRE, you can no longer access your current directory on the machine you are logged onto. Therefore, we will need to copy the folder to a location ACCRE can access.

First, lets check that you can log into ACCRE. Run the following command, but replace USERNAME with the ACCRE username you have been provided.

```
ssh USERNAME@login.accre.vanderbilt.edu
```

You may be asked if you want to continue to connect. Type `yes` and press enter.

You will be prompted for a password. Please type the ACCRE password you have been provided and press enter.

You should see the following if you successfully logged in:

```
Vanderbilt University - Advanced Computing Center for Research and Education


    _    ____ ____ ____  _____    ____ _           _
   / \  / ___/ ___|  _ \| ____|  / ___| |_   _ ___| |_ ___ _ __
  / _ \| |  | |   | |_) |  _|   | |   | | | | / __| __/ _ \ '__|
 / ___ \ |__| |___|  _ <| |___  | |___| | |_| \__ \ ||  __/ |
/_/   \_\____\____|_| \_\_____|  \____|_|\__,_|___/\__\___|_|
================================================================


Go forth and compute!

This is a shared gateway node designed for interactive use and small test jobs.
Please restrict your total system memory usage to less than 31 GB,
and do not run individual processes exceeding 20 minutes of CPU-time.

To list useful cluster commands type: accre_help
To view your current storage type: accre_storage
To list basic Linux commands type: commands101

~$
```

You can see your current directory path with `pwd` and list the contents with `ls`.

Make a directory for running the workshop tutorials.

```
mkdir workshop
```

For now, log out of ACCRE to go back to your local directory.

`logout`

We will use `rsync` to copy the `AF2/2M2Q` directory to ACCRE. Navigate to the AF2 directory. You can check which directory you're in with `pwd`.

In the AF2 directory, run the following command replacing USERNAME with your ACCRE username:

```
rsync -avz 2M2Q USERNAME@login.accre.vanderbilt.edu:~/workshop/
```

You will be prompted for your password again, type your ACCRE password and press enter.

Log back into ACCRE using the ssh command from earlier. Navigate to the 2M2Q directory you just copied.

Run the slurm script using the following command:

```
sbatch run_alphafold_ACCRE_Turing.sb
```

You can check the status of your job on ACCRE with the following command:

```
squeue -u USERNAME
```

This should list a job with the job name being 2M2Q. If you just submitted your job and there is no job listed when running squeue please ask for assistance. If it has been 10-20 minutes, check to see if you have output PDBs in the 2M2Q directory.

This calculation should take 10-20 minutes to generate five models in total if AF2 indeed reads the right MSA files from the provided folders. The results will be under the `2M2Q` folder as multiple pdb files representing unrelaxed and relaxed AF2 structures along with additional information and a ranked version of the PDB outputs.

For now, logout of ACCRE so you can continue with the next part of the tutorial, in the RoseTTAFold section. You can log back onto ACCRE and check the status of your job after 10-20 minutes.

Once your job is complete, you will need to copy the output back to your local directory. Navigate to the `AF2/` directory on your local machine and run the following command:

```
rsync -avz USERNAME@login.accre.vanderbilt.edu:~/workshop/2M2Q ./
```

Resulting pdb files should now be found in `ml_folding/AF2/2M2Q/2M2Q`.

Pre-generated results can be found in `ml_folding/AF2/2M2Q/demo/2M2Q/`.

RoseTTAFold RoseTTAFold can be run locally for the purposes of this tutorial.

Navigate to the `ml_folding/RoseTTAFold/2M2Q` directory.

Copy the 2M2Q fasta file from the AF2 directory.

```
cp ../../AF2/2M2Q/2M2Q.fasta .
```

Create a folder named "results" as the output directory for the results.

```
mkdir results
```

Then to run RoseTTAFold use the following command:

```
bash run_e2e_ver_adapted.sh 2M2Q.fasta results > logfile &
```

Because you are provided with the MSAs for this step, it should take about 10-20 minutes to generate the RF model. Move on to the next section "An interesting test case: Human Caveolin-1", while this is running.

The version of RF we have on our machines does not generate side chains. Therefore, once you get your final model for the calculations (`t000_.e2e.pdb`), we will run the following command in the results directory to generate the sidechains:

```
~/rosetta_workshop/rosetta/main/source/bin/fixbb.default.linuxgccrelease \
     -in:file:s t000_.e2e.pdb -score:weights ref2015 -packing:repack_only
```

In about a minute, you will get another pdb file with the suffix `_0001`, which is your structure with all the sidechains built by Rosetta.

Pre-generated results can be found in `ml_folding/RoseTTAFold/2M2Q/demo/results/`.

Analysis of the generated structures

Contrary to the traditional Rosetta calculations, AF2 or RF calculations do not have scores corresponding to the thermodynamic stability of the structure. Instead, they use a metric called pLDDT to measure the "likeliness of a structure being realistic". The pLDDT value is a per-residue prediction, output in the B-factor column of the output PDB files. For AF, the average pLDDT across the whole structure for each model can be found in the `ranking_debug.json` file. Typically, the lowest-ranking AF2 structure is selected to be the "best" structure and is used for further modeling applications. For RF, using the `run_e2e.sh` script results in a single pdb file instead of five.

Open each AF and RF structure one by one, and color the structures based on B-factors with Chimera. The B-factors in this case correspond to the pLDDT values of each residue. Use Tools > Structure Analysis > Render by Attribute menu and select Attributes of: *Residues,* Attribute: *average > B-factor* options. Click Apply to apply changes to the image. The resulting menu will color-code each residue based on the corresponding pLDDT values. You can create a color key to see the range of values by clicking "Create corresponding color key" and using the mouse to create the key box. For viewing the pLDDT values of RF structures, open the RF output `t000_.e2e.pdb`, not the output from fixbb with the 0001 suffix. Also note, AF outputs pLDDT values on a scale from 0-100, while RF outputs pLDDT values on a scale from 0-1. If plotting pLDDT from B-factor for AF and RF structures in the same session, make sure each is scaled properly. *How consistent are these values within each model? How similar are the pLDDT values of the different models you generated?*

Align all the ranked AF2 structures with UCSF Chimera3 and look for similarities and differences. You can use the Tools > Structure Comparison > MatchMaker tool for this purpose. In the MatchMaker tool box, select the structure you want to align to in Reference structure box and select all other models in the Structure(s) to match box. Click Apply to align structures. *Does the lowest pLDDT structure significantly differ from the structures? What similarities and differences do you see between the two?*

Pay attention to the disulfide bonds of the generated models. *Are they consistent with the experimental structure? Do you see all disulfide bonds forming, or are there free cysteines laying around?*

Case 2 - Modifying the behavior of AlphaFold2 with ColabFold

Introduction

ColabFold is an open-source reimplementation of AlphaFold2 with additional functionality 4,5,6. This involves speed-up by replacing the homology serch with MMseq2. It further allows to control of several internal parameters such as the number of sequences to be used to create the MSA (i.e. its "depth") as well as providing a custom selection of structure templates.

ColabFold can be run locally on a cluster with GPUs, but, as the name suggests, it can also be run online through the Google Colab notebook system. See https://colab.research.google.com/github/sokrypton/ColabFold/blob/main/AlphaFold2.ipynb for the online version. Anyone with a Google Drive account can launch predictions using Google's GPU servers. However, as this is a free service by Google, the amount of computer and the number of jobs you can run at any one time is limited. If you want to use ColabFold for a number of predictions, a local install on your workstation/cluster is likely called for. This is what we'll use today. (Although you can also try the online version, if you want.)

One short-coming of AF2 is the fact that it robustly predicts one conformation of a protein in question. However, many proteins, such as transporters or receptors, adopt different conformations depending on the presence or absence of binding partners. Recent studies have shown that AF2 can be manipulated to predict alternative conformations by reducing the MSA depth and/or providing custom structural templates.

Figure 1: Dark blue part represents transmembrane helix 6.

In this section of the tutorial, we will investigate this at the example of the G protein-coupled receptor Angiotensin (AGTR1). G protein-coupled receptors (GPCRs) are the largest family of membrane proteins and essential for transfering extracellular signals into intracellular pathways. They can assume an 'inactive' and and 'active' conformation, that is marked be the outward movement of its transmembrane helix 6 (TM6; see Figure 1).

Prediction of AGTR1 with default parameters

In the `ColabFold/reference_structures/` directory you find two crytal structures of AGTR1, one in the inactive conformation (4yay.pdb) and one in the active conformation (6do1.pdb). Load both structures into pymol and align them to eachother to investigate their conformational difference. These will be the reference structures against which we compare the upcoming predictions.

First we will generate models with default parameters. Open the bash script/job submission file called `colab_default.job`.

```
gedit run_scripts/colab_default.job
```

This script will execute ColabFold with sequence of AGTR1 as input sequence. The flag `--random-seed 42` determines what seed the network should use. A seed is the starting point for calculations within the ColabFold code that involve randomization. Setting a specific seed ensures reproducibility of the predicions. The flag `--num-seeds 20` controls how many different seeds should be used. The combination of these two flags means that ColabFold will run the prediction 20 times with seeds 42 to 62. For each seed, two models are generated, as controlled with the flag `--num-models 2`. In total, this will therefore result in 40 generated models. For runtime reasons, we will not use internal relaxation protocol, since we are only interested in backbone orientations anyway.

Run the script. Use rsync as above to copy the files to the cluster.

```
rsync -avz run_scripts/ active_templates/ angiotensin.fa USERNAME@login.accre.vanderbilt.edu:~/workshop/Co
```

Log on to the cluster, change to the ColabFold directory and then launch the job

```
ssh USERNAME@login.accre.vanderbilt.edu cd ~/workshop/ColabFold/ sbatch run_scripts/colab_default.job
```

This will take approximately 10 minutes. When predicition is complete, copy the results back and load the resulting models together with the two reference structures into pymol: (Note: There are only a limited number of GPUs availible. If a large number of AlphaFold predictions are still running, your colabfold job may not launch immediately.
Check the status of your jobs with `squeue -u USERNAME`. You may want to cancel your AlphaFold job with `scancel JOBNUMBER` to free up a GPU for your ColabFold job. Note, though, that will cause you to lose all the progress you've made.)

```
rsync -avz USERNAME@login.accre.vanderbilt.edu:~/workshop/ColabFold/out_default ./
```

```
pymol ./reference_structures/*pdb ./out_default/default*.pdb
```

Align the models and color them by typing the following commands in the pymol command line:

```
alignto; color gray; color tv_red, 4yay; color marine, 6do1; center
```

The inactive reference will be red, the active reference will be blue and the ColabFold models will be gray. Identify TM6 and investigate the conformations of the predictions. *Where do the predicted models match the references? Where do they differ? Are there differences regarding the different seeds?*

Biasing prediciton of AGTR1 to adopt the active conformation

As can be seen, predictions with the default parameters resemble the the inactive conformation the most. We will now attempt to bias the prediction of AGTR1 towards the active conformation. For that purpose, 10 structures of different GPCR families in their active conformation are collected in the active_templates folder.

Open the bash script called colab_templates.job

```
gedit run_scripts/colab_templates.job
```

It is identical to the previous commands, with the addition of the flags `--templates` and `custom-template-path`. Note that, when providing custom templates, their name has to be the original PDB-ID in lower case.

Run the predition on the cluster. (The input files will already have been transfered in the previous step). This will again take approximately 10 minutes.

```
sbatch run_scripts/colab_templates.job
```

When the prediction is complete, copy the files back from the cluster and then load the models in pymol again.

```
rsync -avz USERNAME@login.accre.vanderbilt.edu:~/workshop/ColabFold/out_templates ./
```

```
pymol ./reference_structures/*pdb ./out_templates/templates*.pdb
```

Identify TM6 and inspect the models by aligning and coloring them.

```
alignto; color gray; color tv_red, 4yay; color marine, 6do1; center
```

Compared to the previous run, how do the predictions differ?

Impact of a reduced MSA

Lastly, we want to investigate the impact of a very shallow MSA on the predictions of AGTR1. By default, AlphaFold creates a MSA with up to 5120 sequences. Here, we limit the MSA size to mere 96 sequences. For this we extend the default command with the flag `--max-msa 48:96`, where `96` denotes the number of sequences and `48` denotes the number of randomly chosen sequence clusters provided to the neural network.

Run the corresponding script on the cluster. This will take approximately 5 minutes.

```
sbatch run_scripts/colab_msa.job
```

Once prediction is complete, copy the files back and inspect the results in pymol again with the commands used previously. Additionally, you can look at the pLDDT values by typing the following into the pymol command line:

```
rsync -avz USERNAME@login.accre.vanderbilt.edu:~/workshop/ColabFold/out_msa ./
```

```
pymol ./reference_structures/*pdb ./out_msa/templates*.pdb
```

```
spectrum b, rainbow_rev, msa_0_unrelaxed_rank_0*, minimum=0, maximum=100
```

How would you interpret these models?

Case 3 - An interesting test case: Human Caveolin-1

Caveolins are monotopic membrane proteins with a unique scaffold and no close homologs7. A member of this family is CAV-1 that plays a role in the formation of membrane invaginations called caveolae. In the nature, Caveolin-1 (CAV-1) is found as a stable 11-mer that is defined by a coiled region that locks the oligomers, a helical region that runs parallel to the membrane surface, and a parallel beta-barrel.

In this exercise, we will model CAV-1 using the full Uniprot sequence for this protein. Go to the Uniprot website (www.uniprot.org) and look for the entry corresponding to the search keyword "CAV1_HUMAN". Download the canonical fasta sequence from the page belonging to this protein. Save the sequence as `cav1_full.fasta` into your `cav1_WT` directory. Since the first 49 residues of the experimental CAV-1 structure are not present and are predicted to be unstructured, we will truncate the CAV-1 structure to speed up the calculations. With your preferred text editor, open the `cav1_full.fasta` file and delete all the residues before the IDL motif, which corresponds to the first 48 residues. Save the fasta file again.

In the next step, we will model this sequence using AF2 and RF separately to compare their performances in a challenging case whereby no close homologs exist. We would run the calculations the same way we did for the 2M2Q case. However, due to the larger size of CAV-1, we will simply proceed with the provided files and analyze them. Use Chimera to visualize and align the RF-generated structure and the five AF2-generated structures. *How do these two sets of structures compare? Do you see any differences?*

Next, close this session and open the best-ranking AF2 structure (`ranked_0.pdb`). Color the structure based on the pLDDT values that are written to the B-factor column of every AF2 output structure. More detailed instructions

for this can be found in the "Analysis of the generated structures" section above. *What part of the structure was predicted reliably? Are there any regions predicted to be disordered?* Repeat the same protocol with the RF structure in a separate Chimera window. *Are the two methods in agreement regarding the disordered regions?*

Although the CAV-1 structure was not part of the original AF2 training set, it was released in the past year, therefore a comparison with the experimental structure is possible8. Download the experimental 11-meric CAV-1 structure into each Chimera session by using the File > Fetch by ID option and input "7SC0" into the resulting window. Once the structure is loaded, you can compare the AF2 and RF structures with the experimental structure by using the "Matchmaker" tool of Chimera. *Which prediction method was more accurate based on this comparison?*

This test case allows us to see the differences between the algorithms of AF2 and RF in action. Both methods use the same sequence as input, generate MSA files in similar ways, but the resulting structures can be different because of the features used to train either model. This is a case where we see one method performs better over the other, but this not necessarily the general trend.

Case 4 - Sensitivity issues: Single point mutations

Several CAV-1 mutations disrupt the ability of CAV-1 to form oligomers, probably due to disruption of the structure at the monomeric level. As an example, the P132L mutation results in CAV-1s that are unable to form oligomers9. This is believed to be due to structural distortions caused by the proline residue that is absolutely conserved among all caveolins. In order to test the effect of single point mutations, we will follow the same protocol as above with a minor difference. Open the `cav1_P132L` folder for the corresponding AF2 and RF calculations and copy the fasta file and the slurm files from the previous CAV-1 calculations into here. View the fasta file with the command a text editor and look for the residue P132 (which is the residue P82 in the truncated fasta file). Replace the letter P at this position with an L to create the new sequence file. Save this new file as "cav1_P132L.fasta" and close it. Make the corresponding change to the fasta file parameter in your slurm file. You are ready to go, but we will not submit this job for the same reason as the WT caveolin. Check the results folder to see what the resulting structures look like.

Follow the same analysis protocol as before to check the pLDDT values and similarities/ differences both within the AF2-generated models and between the AF2 and RF models. *Is there a difference between the structures predicted by the two methods?* Next, compare the predicted structures with the WT CAV-1 structure from the previous step. *Did the mutation result in a structural change or pLDDT change as the result of this single point mutation?*

The results from these calculations show that AF2 was not sensitive to the changes caused by a single mutation in this case. For RF, there was a change in the arrangement of the helix alpha 3 and the structure downstream, which could be related to the mutation or just a general variation caused by the variations at the prediction stage. Because this structure is not consistent with the experimental CAV-1 structure, it is hard to draw conclusions. Therefore, the use of structure prediction methods like AF2 and RF may be limited in cases where only a single amino acid is substituted.

Case 5 - Complex cases: Modeling of multimers with AlphaFold2

Our final case is going to be modeling a dimeric complex with AF2. Although RF too supports modeling of multimeric complexes, the generation of the MSA files can be problematic and time consuming. For specific applications including protein-protein interactions (PPI) screening with the 2-track version of the algorithm, you can find more information here: https://www.science.org/doi/10.1126/science.abm4805.

Monomeric and multimeric modeling with AF2 involves some differences in terms of the used libraries for the MSA and template generation. Because of this, some of the input keywords to the main script are going to be different. View the slurm script we used to run 2M2Q and the current script. *What are the differences between the two?* Note that we now use the keyword `--multimer` to let AF2 know this is a multimeric run. You can create fasta files including multiple entries consisting of n repeating units in the case of a homomer, or you can input multiple sequences in the case of a heteromer. The order of the inputted sequences should not matter. Another difference since the release of AFv2.2 is that the default AF2 run with a monomer produces five output files, whereas the multimer mode generates five output structures starting with five seeds, resulting in a total of 25 structures.

Today's example is human beta defensin-2, a dimeric peptide (PDB ID: 1FD4). Just like we did for 2M2Q, go to the PDB database and download the fasta file. Because we want to model the dimeric form of this protein, we will have two entries in the fasta file in this way:

```
>sequence1
Seq1
>sequence2
Seq2
```

In other words, copy and paste the same input you have in the fasta file under the current two lines of text in the fasta file. The multimer jobs are submitted the same way as the monomers since all the changes have been made in the slurm file. We will not run this example due to time requirements either. Check the results folder for the generated models. Similar to the previous examples, compare the multiple structures generated with each other, the pLDDT values of the best-ranking structure, and compare the dimeric structure with the PDB entry 1FD4. *Do you think AF2 can be used as a tool to accurately model multimers?* This example uses a case that was already present in the AF2 training set, therefore we expect AF2 to be successful. However, AF2 is capable of predicting unique scaffolds like that of the CAV-1 complex as well10. The accuracy of the method is going to depend on your specific system, so you should always pay attention to the results. The 25 models generated by AF2 can differ a lot in certain cases, so consider these alternative configurations in your analyses as well.

Conclusion

In these tutorials, we went over the basics of running AF2 and RF through simple scripts that can be run locally or on HPCs. In our first case, we used a simple peptide to understand the basics of structure prediction and how to analyze the resulting data. In the second case, we predicted different conformation of GPCRs by using additional features of ColabFold and the impact of the shallow MSA on the predictions. In the third case, we worked with a more complex protein with no high-homology analogs to understand how these similar, yet different algorithms (AF2 and RF) can differ in terms of their performance. In our fourth case, we investigated the effect of single point mutations on protein structure in AF2 and RF calculations. In the fifth case, we used AF2 to model a multimer to better understand the differences from the monomeric case and to assess the performance of AF2 in predicting a simple complex structure. The take-home message from these calculations is that the challenge with deep-learning based structure prediction methods is not running the calculations, but rather analyzing them. Quality of the MSAs, speed considerations, monomeric vs. multimeric prediction cases may all have an effect on the performance of the prediction methods and require special care in some cases.

References

1. Jumper J, Evans R, Pritzel A, et al. *Highly accurate protein structure prediction with AlphaFold. Nature.* 2021;596(7873):583-589. doi:10.1038/s41586-021-03819-2

2. Baek M, Dimaio F, Anishchenko I, et al. *Accurate Prediction of Protein Structures and Interactions Using a Three-Track Neural Network.* Vol 373.; 2021. https://predictioncenter.org/casp14/

3. Pettersen EF, Goddard TD, Huang CC, et al. UCSF Chimera - *A visualization system for exploratory research and analysis. J Comput Chem.* 2004;25(13):1605-1612. doi:10.1002/jcc.20084

4. del Alamo, D., Sala, D., Mchaourab, H. S., & Meiler, J. (2022). *Sampling alternative conformational states of transporters and receptors with alphafold2. eLife*, 11. https://doi.org/10.7554/elife.75751

5. Mirdita, M., Schütze, K., Moriwaki, Y., Heo, L., Ovchinnikov, S., & Steinegger, M. (2022). *Colabfold: Making protein folding accessible to all. Nature Methods*, 19(6), 679–682. https://doi.org/10.1038/s41592-022-01488-1

6. Sokrypton. (n.d.). Sokrypton/Colabfold: *Making protein folding accessible to all!.* GitHub. https://github.com/sokrypton/ColabFold

7. Parton RG, Hanzal-bayer M, Hancock JF. *Biogenesis of caveolae: a structural model for caveolin-induced domain formation.* 2006;1. doi:10.1242/jcs.02853

8. Porta JC, Han B, Gulsevin A, et al. *Molecular architecture of the human caveolin-1 complex. Sci Adv.* 2022;8(19):q. doi:10.1126/sciadv.abn7232

9. Han B, Gulsevin A, Connolly S, et al. *Structural characterization of a breast cancer-associated mutation in caveolin-1. bioRxiv.* Published online January 1, 2022:2022.05.23.493104. doi:10.1101/2022.05.23.493104

10. Gulsevin A, Han B, Porta JC, Mchaourab HS, Meiler J, Kenworthy AK. *Template-free prediction of a new monotopic membrane protein fold and assembly by Alphafold2. Biophys J.* Published online November 17, 2022. doi:10.1016/j.bpj.2022.11.011