Rosetta ddG pipeline

From SBiNLab-wiki

Contents

Overview

This is a guide that explains how to setup and run ΔΔG calculations using the stability pipeline. The pipeline uses Rosetta to create variants and predict stability.

About the pipeline:

- The pipeline takes a .pdb structure as input.
- The main purpose of the pipeline is to do saturation mutagenesis, which mean protein wide calculation of variants
- The pipeline first relaxes the structure and then follows with the ΔΔG energy calculations for each variant.
- To get the ΔΔs the formula 'ΔΔG = Variant - Wildtype' is used. For soluble proteins, the parsed result is giving in units kcal/mol which means the ΔΔG have been divided by a factor of 2.9. This is not the case for membrane proteins, the unit there is REU.

Please be aware of the following limitations before running the pipeline as noted at the github README [1] (https://github.com/KULL-Centre/PRISM/blob/main/software/rosetta_ddG_pipeline/README.md#recommendations--caveats)

Current installations

DeiC

v0.2:

version has a lot of bugfixes from structure preparation. outputs E_res file for quality control & performs other consistency checks. proceed now checks automatically what is missing. MP aligns now on the TM region. Performance is not influenced.

```
/groups/sbinlab/software/PRISM_tools/rosetta_stability-v0.2/software/rosetta_ddG_pipeline
```

No pyrosetta (+conda environment) needed when running MP-ΔΔG_cart.

v0.1.1:

version allows now automatic restart if calculations stopped due to cluster timeout. Includes also MP-ΔΔG_cart option. Performance is not influenced.

```
/groups/sbinlab/software/PRISM_tools/rosetta_stability-v0.1.1/software/rosetta_ddG_pipeline
```

to run membrane proteins (which used pyrosetta), use

```
conda activate /groups/sbinlab/software/PRISM_tools/py3_ros_ddG_env
#run calculations
conda deactivate
```

v0.1:

version with more options, including membrane protein calculations (see more below). Performance is not influenced but output optimized.

```
/groups/sbinlab/software/PRISM_tools/rosetta_stability-v0.1/software/rosetta_ddG_pipeline
```

to run membrane proteins (which used pyrosetta), use

```
conda activate /groups/sbinlab/software/PRISM_tools/py3_ros_ddG_env
#run calculations
conda deactivate
```

v0.0:

last verion edited by Anders, used for most previous calculations (as of May 2021)

```
/groups/sbinlab/software/PRISM_tools/rosetta_stability-v0.0/software/rosetta_ddG_pipeline
```

or

```
/groups/sbinlab/software/PRISM/software/rosetta_ddG_pipeline/
```

v0.0.1-beta.1:

some intermediate version

```
/groups/sbinlab/software/PRISM_tools/PRISM-v0.0.1-beta.1/software/rosetta_ddG_pipeline
```

Installing the pipeline

The pipeline can be found at GitHub: https://github.com/KULL-Centre/PRISM/tree/master/software/rosetta_ddG_pipeline

To install the pipeline, clone the pipeline or a release and add the following paths to your .bashrc

```
# Path to muscle
export muscle_exec='/groups/sbinlab/software/muscle/muscle3.8.31_i86linux64'
# Path to TMalign
export TMalign_exec='/groups/sbinlab/software/TMalign/TMalign'
# Path to prism parser
export prism_parser='/groups/sbinlab/software/PRISM_tools/prism_parser/scripts'
# Path to your installation of the pipeline
export ddG_pipeline='/groups/sbinlab/software/PRISM_tools/rosetta_stability-v0.2/software/rosetta_ddG_pipeline'
# Path to Rosetta extensions and tools
export Rosetta_extension='linuxgccrelease'
export Rosetta_main_path='/groups/sbinlab/software/Rosetta_2021_Aug_c7009b3/source/'
export Rosetta_database_path='/groups/sbinlab/software/Rosetta_2021_Aug_c7009b3/database/'
export Rosetta_tools_path='/sbinlab/software/Rosetta_tools/tools/'
# Add the command line as an alias. (A nice to have not a need to have)
alias run_ddG_pipeline='python3 /groups/sbinlab/software/PRISM_tools/rosetta_stability-v0.2/software/rosetta_ddG_pipeline/
```

The pipeline uses Python3 and a few third-party extension such as:

- Biopython
- Numpy
- Pandas
- scipy

Third-party software

- Muscle
- Rosetta
- PyRosetta
- git

Rosetta

When updating the default Rosetta version on the respective cluster, the first step with respect to the ΔΔG pipeline is to verify that there were no major changes - or, if major changes are expected due to changes in the ΔΔG code, verify that performance on our gold standard set GB1 is still good. For more details read up on the testing README at github [2] (https://github.com/KULL-Centre/PRISM/blob/main/software/rosetta_ddG_pipeline/test/README.md)

Once performance is assured, the Rosetta location should be updated both in the environment variables above as well as in https://github.com/KULL-Centre/PRISM/blob/main/software/rosetta_ddG_pipeline/rosetta_paths.py

Pipeline flags

v0.2:

New flags & changes introduced in this version

General flags

```
# This flag dictates the runmode
-i                  Default = create
    proceed         #Starts the calculations based on the files in the folder - now actually working without recalculating
```

Flags for membrane proteins:

```
#span file calculation method, now default DSSP
--mp_calc_span_mode          default=DSSP
                    False
                    struc
                    DSSP
                    octopus
                    bcl
                    Boctopus
#check if structural alignment based on TM region only (default) or on the complete structure
-spTM               default=True
```

Flags for modifying the protocol/pipeline (mostly development, partly not visible with the --help option)

```
#Scaling of ΔΔG to kcal/mol - default is automatic set for soluble proteins to x/2.9 and for membrane proteins to x/1.
--scale             default=-999 (internally 2.9 or 1)

--mp_energy_func_weights #only used when mp_cart_ddg is set to 1
# calculates the ΔΔGs in cartesian space for membrane proteins. If set to true, the according mp_energy_func_weights are se
--mp_cart_ddg       default=1
```

v0.1 & v0.1.1:

General flags

```
# path to structure flag
-s 1PGA.pdb        Required!
# Output path
-o run/             Default = ./run/
# This flag dictates the runmode
-i                  Default = create
    create          #Create all files but dont start the calculations
    proceed         #Starts the calculations based on the files in the folder
    relax           #Creates files and run relaxation
    ddg_calculation  #Creates files and run ddG calculation
    fullrun         #Creates files and run the full calculation

# mutation mode: do saturation mutagenesis [all] or use input mutfile
-mm                 Default = all
    all
    mut_file
# mutation input file, can be a rosetta_mutfile, mutfile_dir or pipeline_file (Syntax showed later) - can be generated by p
-m mutation_input.txt

# Chooses which chain to work on
```

```
--chainid A        Default = A
# Choose additional structures to keep. This is run in cooperation with chainid
--run_struc AB
# Choose whether to keep or remove ligands in structure
--ligand True      Default = False
# Allows overwriting of files in folder. Usefull for multiple calls to same folder
--overwrite_path True         Default = False
# Choose slurm_partition
--slurm_partition sbinlab_id  Default = sbinlab
--verbose #increase verbose level
--gapped_output #output pism and pdb file with shifted residue numbering according to input
```

Flags for membrane proteins:

```
--mp #needs to be set to true if running membrane protein pipeline
--mp_span #path to span file - if not present, choose calculation method in mp_calc_span_mode
#span file calculation method, prefered DSSP
--mp_calc_span_mode          default=False, prefered DSSP
                     False
                     struc
                     DSSP
                     octopus
                     bcl
                     Boctopus
#transfer protein into membrane plane, prefered OPM
--mp_prep_align_mode         default=OPM
                     False
                     OPM
                     PDBTM
                     TMDET
                     MemProtMD
--mp_align_ref #reference PDBid and chain for alignment, e.g. 3sn6_R
```

Flags for modifying the protocol/pipeline (mostly development, partly not visible with the --help option)

```
# Path to ddg flags
-ddgflags ddgflags.txt    Default = ddG_pipeline/rosetta_parameters/cartesian_ddg_flagfile
# Path to relax flags
-relaxflags relaxflags.txt  Default = ddG_pipeline/rosetta_parameters/relax_flagfile
--mp_relax_xml path to relaxxml file for membrane protein pipeline
--uniprot not implemented

--mp_thickness
--mp_lipids
--mp_temperature
--mp_pH
--benchmark_mp_repack
--benchmark_mp_repeat
--benchmark_mp_relax_repeat
--benchmark_mp_relax_strucs
--mp_ignore_relax_mp_flags
--mp_energy_func
--mp_repack_protocol
--mp_multistruc_protocol
```

v0.0:

```
# path to structure flag
-s 1PGA.pdb        Required!
# Output path
-o run/            Default = run/
# This flag dictates the runmode
-i fullrun         Default = Create
    create           #Create all files but dont start the calculations
    proceed          #Starts the calculations based on the files in the folder
    relax            #Creates files and run relaxation
    ddg_calculation  #Creates files and run ddG calculation
    fullrun          #Creates files and run the full calculation
    analysis         #Not working
# mutation input file. This file is used to specify which exact positions that should be mutated (Syntax showed later)
```

```
-m mutation_input.txt
# path to prism file for creating mutfiles
-p path_to_prism_file
# Path to ddg flags
-d ddgflags.txt    Default = ddG_pipeline/rosetta_parameters/cartesian_ddg_flagfile
# Path to relax flags
-r relaxflags.txt  Default = ddG_pipeline/rosetta_parameters/relax_flagfile
# Chooses which chain to work on
--chainid A        Default = A
# Choose whether to keep or remove ligands in structure
--ligand True      Default = False
# Choose additional structures to keep. This is run in cooperation with chainid
--run_struc AB
# Choose slurm_partition
--slurm_partition sbinlab_id  Default = sbinlab
# Allows overwriting of files in folder. Usefull for multiple calls to same folder
--overwrite_path True         Default = False
```

The default Rosetta flags can be found in : ddG_pipeline/rosetta_parameters/ or with the help command -h

Pipeline workflow

The pipeline workflow consist of 3 main parts which result in 4 types of slurm jobs.

Structure preparation The first part of the pipeline cleans the pdb so it only contains the information used in the ddG calculations and saves some of the removed information. This step also creates the mutation_input.txt, mutfiles and all necessesary files to do the calculations

Relaxation The relaxation step, queues the relaxation. Relaxation is done to relax the structure thus preparing it for energy calculations. Relaxation also runs the relaxation parser which from the 20 structures generated by relaxation picks the lowest energy one Thus a total of 2 jobs are queued in this step

ddG_calculation The ddG takes a relaxed structure as input and queues the ddG calculations. One job is made for each position that are run. Regardless of number of variants in that position After all jobs have been completed the pipeline will parse the results to a datafile. This result in a total of (#positions+1) jobs queued in this step

Run stability calculations

To run the pipeline we first need to have a starting structure. The pipeline takes .pdb files as input and should also be able to take homology models as input.

Running saturation mutagenesis Saturation mutagenesis will run all protein variants for your chosen protein. The example below runs saturation mutagenesis on chain A in the structure 1PGA.pdb

Preparing the structure for Rosetta Rosetta can deal with some special residues and molecules well, and less so with others. It will also remove any residues that are missing backbone heavy atoms. This needs to happen before any reference sequences are extracted from the coordinates. The easiest way to get all atoms that are compatible with current Rosetta is

```
/path/to/rosetta/source/bin/score.linuxgccrelease -s 1PGA.pdb -out::output
```

This will create a coordinate file named 1PGA_0001.pdb that should be used in the subsequent steps.

```
run_ddG_pipeline -s 1PGA.pdb -o run/ -i fullrun
```

If we want to do it on chain B we change the command to

```
run_ddG_pipeline -s 1PGA.pdb -o run/ -i fullrun --chainid B
```

To run specific variants you can use the -m flag and create a mutation_input.txt file. The file need the following format

```
#Wildtype, ResNumber, Variants
G 10 DEA
H 11 TW
A 12 ACDEFGHIKLMNPQRSTVWY
```

Alternatively, mutfile following the Rosetta scheme can be used. Also, prism files can be transferred into mutfiles following the ../script/prism2mutfile.py script. This script takes also a pdb file and renumbers it to rosetta numbering automatically.

After this file have been made, run the ddG calculation

```
run_ddG_pipeline -s 1PGA.pdb -o run/ -i fullrun -m mutation_input.txt
```

If you want run a structure while keeping another part of the structure

```
run_ddG_pipeline -s 1PGA.pdb -o run/ -i fullrun --chainid A --runstruc AB
```

After the run

Parsed files can be found in run/ddG/output

Run stability calculations for structure with ligand

To do stability calculations of a structure with its ligand(s), the relaxation step has to be run outside the pipeline so that the ligand will stay in the structure.

To create the files for the manual relaxation we can use the pipeline with the flag -i create

```
run_ddG_pipeline -s 1U72.pdb -o run/ -i create
```

Notice that running this command will produce an input.pdb file in run/relax/input/, where the ligand has been removed, therefore remove or replace this file with your PDB file containing the ligand.

To do the manual relaxation, correct the relaxation command in the rosetta_relax.sbatch file, also located in run/relax/input/, to point to the PDB file with the ligands. Then submit the job

```
sbatch rosetta_relax.sbatch
```

The output, i.e. the 20 relaxed structures and their corresponding score files, will be in the folder where you executed the command. All structures will still contain the ligand. Then select the lowest energy structure over the 20 relaxation runs. This can be done with the command

```
sort -k1,1 -k2n *.sc
```

For the ddG calculations you can copy the manually relaxed structure with lowest energy to the run/ddG/input/ folder. The ddG calculations can be performed using the pipeline with the flag -i ddg_calculation on the relaxed structure and using --overwrite_path True to run from the same folder again

```
run_ddG_pipeline -s run/ddG/input/12-1U72_bn15_calibrated_0001.pdb -o run/ -i ddg_calculation --overwrite_path True
```

This will result in ddG values for the structure containing its ligand.

Run stability calculations for a membrane protein

To run a protein through the membrane protein pipeline, -mp must be specified and a prepared protein provided. If the protein is not yet aligned, add --mp_align_ref which switches --mp_prep_align_mode to OPM. If no span file is provided, specify --mp_calc_span_mode with DSSP to calculate it:

```
run_ddG_pipeline -s 6xro.pdb -o run -i fullrun --chainid A -mp 1 --mp_calc_span_mode DSSP --mp_align_ref 6xro_A
```

Run stability calculations for a homo/hetero-multimer

To calculate ΔΔGs for homodimers, the pipeline & preprocessing has been initially tested. More documentation to come. You can check out the dedicated inputs from the unittest test_create_homodimer_prov_flag [3] (https://github.com/KULL-Centre/PRISM/blob/main/software/rosetta_ddG_pipeline/test/test_mp_pipeline.py#L403) or get in touch with Johanna Tiemann or Matteo Cagiada.

Here are some examples how you can do it:

Homo-oligomer:

- first run pdb_to_prism[4] (https://github.com/KULL-Centre/PRISM/blob/main/software/scripts/pdb_to_prism.py) on the monomer (or just one chain) to get the prism-file info

- use this prism-file and your multi-mere as input for homo_dimer[5] (https://github.com/KULL-Centre/PRISM/blob/main/software/scripts/homo_dimer.py) or homo_oligomers[6] (https://github.com/KULL-Centre/PRISM/blob/main/software/scripts/homo_oligomer.py) to generate the "final" mutation input for the pipeline. Along this, also a freshly renumbered (according to rosetta numbering starting with 1) and re-chained (all chains to A) pdb will be generated that you should use as your input for the pipeline

- run the pipeline with chainid=A, the mutations=pipeline_mutfile.txt and mutate_mode=mut_file

Homo/hetero-oligomer with ligands:

- ligands are now supported/kept during the process. You need to add the flags (and ligand names) accordingly during the process (in all scripts)

hetero-oligomer:

- hetero oligomers do not need a special preparation, just add run_struc=ABC (with all the chains you want to be kept in the run).

Practical notes

- All values in the parsed files are divided by 2.9. This scaling factor brings ddGs from the beta_nov16_cart sfxn to roughly kcal/mol. (Frank DiMaio, personal communication).
- For large structures, be sure to adjust the allocated memory used in the relax step. The actual ddG step uses local modelling only and is thus independent of the protein size.
- To continue a simulation that has run into the walltime simply rerun the ddg/input/rosetta_cartesian.sbatch while being in the ddg/run/ folder. It is recommended to check before *why* the walltime was exceeded and if there are any problems with the input, Rosetta installation, state of the cluster etc.
- More information or examples can also be found here: https://github.com/KULL-Centre/PRISM/tree/master/software/rosetta_ddG_pipeline

Current local expert(s)

Johanna, Helena (ligands)

Categories: Pages with syntax highlighting errors | KLL | AS

- This page was last modified on 19 August 2022, at 10:45.