RosettaDDGPrediction User Guide

Rosetta version: 3.12
Last updated: June 19th, 2022

Introduction

`RosettaDDGPrediction` is a Python package relying on Rosetta-based protocols for the prediction of the ΔΔG of stability of a monomeric protein upon mutation and the ΔΔG of binding of a protein complex upon mutation.

So far, we have incorporated two different types of protocols in our package (`cartddg` and `flexddg` protocols), each one having different variants.

`cartddg` and `cartddg2020` protocols

The `cartddg` and `cartddg2020` protocols serve the purpose of predicting the ΔΔG of stability of a monomeric protein upon mutation, according to the procedure devised by Park et al. [1] and later revised by Franz and co-workers [2].

The protocols named `cartddg_talaris2014` and `cartddg_ref2015` refer to the work performed by Park and co-workers, while those named `cartddg2020_talaris2014` and `cartddg2020_ref2015` refer to the work of Franz and co-workers. Here, `talaris2014` and `ref2015` refer to the names of the energy function used by the protocol.

`cartddg` protocols include three steps:

1. The `relax` step (called `relax2020` in `cartddg2020` protocols) performs a pre-processing of the input PDB structure, which consists of relaxation in the Cartesian space of the structure, completed with the Rosetta `relax` application.
2. The `structure_selection` step corresponds to the selection of the lowest energy structure produced by the `relax` step, which will be later fed to the final step. The `structure_selection` step is performed by Python code.
3. The `cartesian` step deals with the actual ΔΔG prediction, performed with the Rosetta `cartesian_ddg` application. For a set of mutations to be performed on a single PDB structure, the relax step should only be run once to produce the relaxed structure. Then, this structure would serve as input for as many runs as needed to predict the ΔΔG associated with the mutations of interest. To perform only the `relax` or the `cartesian` step separately, use the `relax_*.yaml` and `cartesian_*.yaml` files in the `config_run` directory. In this case, the structure selection must be performed manually, and the selected structure must be passed as input PDB file when running the `cartesian` step.

`flexddg` protocols

The `flexddg` protocols are employed to predict the ΔΔG of binding of a protein complex upon mutation. Those protocols are named `flexddg_ref2015` and `flexddg_talaris2014`, and come from the work of Barlow and co-workers [3].

`flexddg` protocols make use of an XML script implementing the procedure devised by the original authors. A slightly modified version of the XML script that the authors provide in their Flex ddG tutorial (which can be found here) is supplied with `RosettaDDGPrediction`, inside the `RosettaScripts` directory.

Both protocols consist of only one step, named `flexddg`, which employs the `rosetta_scripts` executable to run the procedure laid out in the XML file.

How to run

The `RosettaDDGPrediction` pipeline consists of four steps:

1. Data generation is performed by the `rosetta_ddg_run` executable, where the ΔΔG prediction occurs.
2. A sanity check on the runs can be performed using the `rosetta_ddg_check` executable.
3. Data aggregation is performed by the `rosetta_ddg_aggregate` executable, where raw data are aggregated in CSV files for all mutations included in a run.
4. Plotting is performed by the `rosetta_ddg_plot` executable, where aggregated data can be plotted in different ways to visually inspect the data. Only aggregated data can be given as inputs.

Configuration files

`RosettaDDGPrediction` is designed as a set of executables whose behavior can be almost fully customized via YAML configuration files. You can use the default configuration files provided, modify them, change their names, add custom files to the corresponding folders inside the package or create customized files and pass them directly to the `RosettaDDGPrediction` executables.

There are four types of configuration files, examples of which can be found in the homonymous folders inside the package:

- `config_run` files each define a protocol to be run. Such files mark the steps included in the protocols, define the Rosetta options to be used to run each step, the names of the output files and the names of the directories where the steps should be run. They also define how the protocol treats input PDB files (for example whether it accepts multi-chains PDB files or not) and mutations (what residue numbering the protocol uses, whether it generates one structure for each mutation or an ensemble, etc.).
- `config_settings` files contain settings for the runs that are not related to the specific protocol being run. These include options for parallelization, use of MPI, etc.
- `config_aggregate` files contain both options to customize the name and format of the aggregated outputs and settings for the (optional) conversion of the Rosetta ΔΔG scores (normally expressed in Rosetta Energy Units) to kcal/mol.
- `config_plot` files define options for the format of the output file containing the plot and for the plot aesthetics.

`rosetta_ddg_run`

`rosetta_ddg_run` is the executable responsible for running the Rosetta protocols for ΔΔG prediction.

Command line

```
rosetta_ddg_run [-h] -p PDBFILE -cr CONFIGFILE_RUN -cs
                      CONFIGFILE_SETTINGS -r ROSETTAPATH [-d RUNDIR]
                      [-l LISTFILE] [-n NPROC] [--saturation]
                      [--reslistfile RESLISTFILE]
```

Options

Option	Meaning
`-h`, `--help`	Show the help message and exit.
`-p`, `--pdbfile`	PDB file of the wild-type structure.
`-cr`, `--configfile-run`	Configuration file of the protocol to be run. If it is a name without extension, it is assumed to be the name of a YAML file in `$INSTALLDIR/RosettaDDGPrediction/RosettaDDGPrediction/config_run`.
`-cs`, `--configfile-settings`	Configuration file containing settings to be used for the run. If it is a name without extension, it is assumed to be the name of a YAML file in `$INSTALLDIR/RosettaDDGPrediction/RosettaDDGPrediction/config_settings`.
`-r`, `--rosettapath`	Path to the Rosetta installation directory.
`-d`, `--rundir`	Directory where the protocol will be run. Default is the current working directory.
`-l`, `--listfile`	File containing the list of selected mutations (or positions to perform saturation mutagenesis).
`-n`, `--nproc`	Number of processes to be started in parallel. Default is one process (no parallelization).
`--saturation`	Perform saturation mutagenesis on selected positions.
`--reslistfile`	File containing the list of residue types to be included in the saturation mutagenesis. It is used only if `--saturation` is provided.

Input files

Non-saturation mutagenesis scans

Mutations list file

The file containing the mutations is a newline-separated file with a mutation per line, with multiple simultaneous mutations defined on the same line and separated by a comma. Chain, wild-type residue, residue number and mutated residue of each single mutation should be period-separated.

```
A.F.118.W
A.F.52.D,A.L.53.F
```

Some protocols require some extra information about each mutation in the mutation file.

For `flexddg` protocols, this information concerns the chain to be moved away from the complex when scoring each chain of the complex separately to compute the ΔΔG of binding (please to the Flex ddG tutorial for more details).

These extra pieces of information are whitespace-separated from the mutation definition and from each other (if more than one is required).

```
A.F.118.W A
B.G.1044.E,B.W.1040.C B
```

Saturation mutagenesis scans

Positions list file

When you run a saturation mutagenesis scan, the list file will include positions on which the scan needs to be performed.

The format is similar to that of the mutations list file, but you must omit the mutated residue.

Extra pieces of information are passed exactly as you would do with a list of mutations.

```
A.F.118 A
B.G.1044 B
```

Multiple simultaneous mutations are supported when running saturation mutagenesis scans on positions.

If more than one position needs to undergo saturation mutagenesis simultaneously, you can specify them on the same line, and all possible combinations of mutations on those positions will be performed. Positions can also be specified along "fixed" mutations that will be included in any instance of the saturation mutagenesis scan performed on those positions.

```
A.F.118,B.G.1044 A
A.F.118,B.G.1044.A A
B.G.1044 B
```

Residue types list file

When running a saturation mutagenesis scan, it is mandatory to provide a newline-separated file where the names of the residue types that will be part of the scan are specified.

Names must follow the one-letter convention for the 20 canonical residues, and the Rosetta resfile convention for non-canonical residues.

```
A
D
E
DALA
```

Outputs

The structure of the output will depend on the protocol that was run.

`cartddg` protocols

Suppose you performed three iterations of relaxation, you had a mutations file as the one presented below, and you ran the `cartddg_ref2015` protocol with the default configuration file.

```
A.F.118.W
A.F.52.D,A.L.53.F
```

This would be the directory tree you would find in your running directory.

```
relax/
|----| flags.txt
|----| relax_structure_0001.pdb
|----| relax_structure_0002.pdb
|----| relax_structure_0003.pdb
|----| relax.out
|----| scorefile.sc
cartesian/
|----| A-F118W/
|----|----| cartesian.out
|----|----| flags.out
|----|----| mutation.ddg
|----|----| mutation.mutfile
|----|----| MUT_118TRP_bj1.pdb
|----|----| MUT_118TRP_bj2.pdb
|----|----| MUT_118TRP_bj3.pdb
|----|----| WT_bj1.pdb
|----|----| WT_bj2.pdb
|----|----| WT_bj3.pdb
|----| A-F52D_A-L53F/
|----|----| cartesian.out
|----|----| flags.out
|----|----| mutation.ddg
|----|----| mutation.mutfile
|----|----| MUT_52ASP_53PHE_bj1.pdb
|----|----| MUT_52ASP_53PHE_bj2.pdb
|----|----| MUT_52ASP_53PHE_bj3.pdb
|----|----| WT_bj1.pdb
|----|----| WT_bj2.pdb
|----|----| WT_bj3.pdb
```

`relax.out` and `cartesian.out` are the Rosetta log files.

`flags.txt` is the Rosetta flags file used to run the protocol, containing all the Rosetta options, input and output files specified for the run. Flags files make it easier to restart a single run within a protocol in case it unexpectedly fails.

`scorefile.sc` is a file generated by Rosetta containing the energy scores of the relaxed structures. This is used in the `structure_selection` step to select the structure with the lowest score.

`mutation.mutfile` is a file following the Rosetta mutfile format containing the mutation performed in the folder where the file is.

`mutation.ddg` is a file generated by Rosetta containing the predicted ΔΔG scores for all the iterations performed by `cartesian_ddg`. Each wild-type/mutant pair of structures corresponds to an iteration, and the final ΔΔG score will be the arithmetic means of the ΔΔG scores calculated at each iteration.

The PDB files correspond to the wild-type/mutant pair of structures numbered after the iteration that generated them. If the `cleanlevel` option for the `cartesian` step in the protocol's configuration file is set to `pdb`, those files are removed at the end of the run and do not appear in the final directory tree.

`flexddg` protocols

Suppose you ran the `flexddg_ref2015` protocol with the default configuration file, using the mutations list file outlined below.

```
A.F.118.W A
```

This would be the directory tree in your running directory.

```
flexddg/
|----| A-F118W/
|----|----| 1/
|----|----|----| ddg.db3
|----|----|----| flexddg.out
|----|----|----| inputpdb_0001.pdb
|----|----|----| inputpdb_0001_last.pdb
|----|----|----| inputpdb_0001_low.pdb
|----|----|----| score.sc
|----|----|----| struct.db3
|----|----| 2/
(same content as 1/)
|----|----| ...
|----|----| 35/
(same content as 1/)
```

`flexddg.out` is the Rosetta log file.

`ddg.db3` is a Rosetta database file containing the energy scores of:

- The minimized wild-type and mutant structures used as starting points for the backrub trajectories.
- All structures generated along the backrub trajectories of both the wild-type and the mutant.
- The scores of the wild-type and mutant structures at the end of the final minimization, that will be used to compute the final ΔG and ΔΔG scores.

`struct.db3` is a Rosetta database file containing the structures generated along both the wild-type and the mutant backrub trajectories, saved at certain intervals. Since it includes all the structural information necessary to write all the saved structures to output PDB files, it is usually pretty big and can be safely removed if you are only interested in ΔΔG scores. If the `cleanlevel` option of the `flexddg` step in the protocol's configuration file is set to `structdbfile` or `all`, this file is removed at the end of the run and does not appear in the directory tree. Regardless of the chosen cleaning level, the user can extract the PDB structures from the different steps of the backrub trajectory generated by the protocol by setting the `extract` flag in the `extract_structures` sub-section of the `flexddg` step to `True` in the configuration file. In this case, the extraction will be performed before the removal of the `struct.db3` file even if the user has selected the `structdbfile` or `all` cleaning level.

`score.sc` is a Rosetta "scorefile" containing the score of the wild-type structure after the first minimization. This file can also be removed if there is no interest in inspecting the intermediate steps of the protocol, together with the PDB files found in each folder, that are also intermediate files. If the `cleanlevel` option of the `flexddg` step in the protocol's configuration file is set to `all`, both the `struct.db3` file and the PDB files are removed at the end of the run and do not appear in the directory tree.

rosetta_ddg_check_run

`rosetta_ddg_check_run` is the executable responsible for checking if the runs completed successfully.

If a run crashes due to a Python exception, it will be reported by `rosetta_ddg_run` at runtime. However, `rosetta_ddg_check_run` is needed for those runs that crashed due problems encountered by the Rosetta executables, since `rosetta_ddg_run` has no way to detect those crashes at runtime.

Therefore, `rosetta_ddg_check_run` can be used to log to the console (or to a file if the standard output has been redirected) whether any crashed runs were found. A run's crash is assessed by looking for a `ROSETTA_CRASH.log` file in the directory where the run was performed since its presence indicates that the Rosetta executables crashed.

Command line

```
rosetta_ddg_check_run [-h] -cr CONFIGFILE_RUN [-d RUNNING_DIR]
                      [-mf MUTINFOFILE]
```

Options

Option	Meaning
`-h`, `--help`	Show the help message and exit.
`-cr`, `--configfile-run`	Configuration file of the protocol to be run. If it is a name without extension, it is assumed to be the name of a YAML file in `$INSTALLDIR/RosettaDDGPrediction/RosettaDDGPrediction/config_run`.
`-d`, `--running-dir`	Directory where the protocol was run. Default is the current working directory.
`-mf`, `--mutinfofile`	File with info about the mutations (it is created when running). Not needed if the calculation to be checked did not perform any mutations.

rosetta_ddg_aggregate

This executable is used to aggregate data obtained when running one of the protocols.

Command line

```
rosetta_ddg_aggregate [-h] -cr CONFIGFILE_RUN [-ca CONFIGFILE_AGGREGATE]
                      [-d RUNNING_DIR] [-od OUTPUT_DIR] [-mf MUTINFOFILE]
                      [-n NPROC] [--mutatex-convert] [--mutatex-reslistfile
                      MUTATEX_RESLISTFILE] [--mutatex-dir MUTATEX_DIR]
```

Options

Option	Meaning
`-h`, `--help`	Show the help message and exit.
`-cr`, `--configfile-run`	Configuration file of the protocol to be run. If it is a name without extension, it is assumed to be the name of a YAML file in `$INSTALLDIR/RosettaDDGPrediction/RosettaDDGPrediction/config_run`.
`-ca`, `--configfile-aggregate`	Configuration file for data aggregation. If it is a name without extension, it is assumed to be the name of a YAML file in `$INSTALLDIR/RosettaDDGPrediction/RosettaDDGPrediction/config_aggregate`. Default is `$INSTALLDIR/RosettaDDGPrediction/RosettaDDGPrediction/config_aggregate/aggregate.yaml`.
`-d`, `--running-dir`	Directory where the protocol was run. Default is the current working directory.
`-od`, `--output-dir`	Directory where to store the aggregate data. Default is the current working directory.
`-mf`, `--mutinfofile`	File with info about the mutations (it is created when running).
`-n`, `--nproc`	Number of processes to be started in parallel. Default is one process (no parallelization).
`--mutatex-convert`	Request the conversion of the aggregate output files to MutateX-compatible outputs (the original files will be kept).
`--mutatex-reslistfile`	File containing the list of residue types used to sort the ones used in the saturation mutagenesis in the MutateX-compatible outputs.
`--mutatex-dir`	Name of the directory (inside the output directory) where the MutateX-compatible output files will be stored. Default is: 'mutatex_compatible'.

Outputs

Suppose you ran the `cartddg_ref2015` protocol as described above, used the default configuration file for data aggregation, and chose a directory called `aggregation_output_dir` as output directory. You would obtain the output files enumerated below.

N.B.: the aggregated files have the same format both for `cartddg`, `cartddg2020`, and `flexddg` protocols.

```
aggregation_output_dir/
|----| A-F52D_A-L53F_aggregate.csv
|----| A-F52D_A-L53F_structures.csv
|----| A-F118W_aggregate.csv
|----| A-F118W_structures.csv
|----| ddg_mutations_aggregate.csv
|----| ddg_mutations_structures.csv
```

Files whose names start with the name of a mutation and end in `*_aggregate.csv` contain energy scores for that mutation (both ΔG scores for wild-type structures and mutants and ΔΔG scores). These scores are averages over all the structures of the ensemble generated by the protocol for `cartddg` and `flexddg` protocols, while they correspond to the lowest ΔΔG score found for `cartddg2020` protocols.

Files whose names start with the name of a mutation and end in `*_structures.csv` contain per-structure energy scores for that mutation (both ΔG scores for wild-type structures and mutants and ΔΔG scores).

`ddg_mutations_aggregate.csv` contains the same data present in the `*_aggregate.csv` files for all mutations found in the directory where the protocol was run.

`ddg_mutations_structures.csv` contains the same data present in the `*_structures.csv` files for all mutations found in the directory where the protocol was run.

MutateX-compatible output files

For saturation mutagenesis scans, `rosetta_ddg_aggregate` is able to generate files compatible with the plotting system implemented in the MutateX software [4]. Several command-line options are available to specify whether the conversion of the aggregate files to a MutateX-compatible format is requested, a residue list file to use so that the residue types included in the saturation mutagenesis will be reported in the correct order in the MutateX-compatible outputs, and the name of the directory where these new outputs will be stored.

`rosetta_ddg_plot`

Command line

```
rosetta_ddg_plot [-h] -i INFILE -o OUTFILE [-ca CONFIGFILE_AGGREGATE]
                 -cp CONFIGFILE_PLOT [--saturation-on SATURATION_ON]
```

Options

Option	Meaning
`-h`, `--help`	Show the help message and exit.
`-i`, `--infile`	Input CSV file (aggregated data).
`-o`, `--outfile`	Output file containing the plot.
`-ca`, `--configfile-aggregate`	Configuration file for data aggregation. If it is a name without extension, it is assumed to be the name of a YAML file in `$INSTALLDIR/RosettaDDGPrediction/RosettaDDGPrediction/config_aggregate`. Default is `$INSTALLDIR/RosettaDDGPrediction/RosettaDDGPrediction/config_aggregate/aggregate.yaml`.
`-cp`, `--configfile-plot`	Configuration file for plotting. If it is a name without extension, it is assumed to be the name of a YAML file in `$INSTALLDIR/RosettaDDGPrediction/RosettaDDGPrediction/config_plot`.

Plot types

These plot types are supported so far:

- `total_heatmap`

  Expected input

  Any of the `*_aggregate.csv` files (including `ddg_mutations_aggregate.csv`).

  Description

  A one-row heatmap where all mutations are displayed on the x-axis, each one corresponding to a
  cell color-coded according to the ΔΔG score corresponding to that mutation.

  Example of configuration file

  `$INSTALLDIR/RosettaDDGPrediction/RosettaDDGPrediction/config_plot/total_heatmap.yaml`.

- `total_heatmap_saturation`

Notes

This plot only accepts data produced from saturation mutagenesis scans where no multiple simultaneous mutations were performed on any position.

Expected input

Any of the `*_aggregate.csv` files (including `ddg_mutations_aggregate.csv`).

Description

A 2D heatmap showing all positions scanned on the x-axis, all the residue types included in the mutagenesis on the y-axis and each cell color-coded according to the ΔΔG score corresponding to a specific position mutated to a specific residue type. In case of partially scanned positions, cells with missing ΔΔG scores can be masked.

Example of configuration file

`$INSTALLDIR/RosettaDDGPrediction/RosettaDDGPrediction/config_plot/total_heatmap_saturation.yaml`.

- `contributions_barplot`

  Expected input

  Any of the `*_aggregate.csv` files (including `ddg_mutations_aggregate.csv`).

  Description

  A bar plot showing a stacked bar for each mutation, where the components of the stacked bar are the different energy contributions to the ΔΔG score corresponding to that mutation. Positive contributions are stacked on the positive y semi axis while negative contributions are stacked on the negative y semi axis. By default, total ΔΔG scores are written above each bar, and a horizontal line is drawn to highlight the 0.0 position on the y axis.

  Example of configuration file

  `$INSTALLDIR/RosettaDDGPrediction/RosettaDDGPrediction/config_plot/contributions_barplot.yaml`.

- `dg_swarmplot`

  Expected input

  Any of the `*_structures.csv` files (including `ddg_mutations_structures.csv`).

  Description

  A swarmplot showing, for each mutation, the ΔG score of each wild-type and mutant structure present in the ensemble of structures generated by the protocol.

  Example of configuration file

  `$INSTALLDIR/RosettaDDGPrediction/RosettaDDGPrediction/config_plot/dg_swarmplot.yaml`.

Outputs

The output is always a single file, containing a plot which depends on the plot type and aesthetics specified in the plot configuration file and on the type of aggregated file passed.

References

1. Park, Hahnbeom, et al. "Simultaneous optimization of biomolecular energy functions on features from small molecules and macromolecules." Journal of chemical theory and computation 12.12 (2016): 6201-6212. ↩

2. Frenz, Brandon, et al. "Prediction of protein mutational free energy: benchmark and sampling improvements increase classification accuracy." Frontiers in bioengineering and biotechnology (2020): 1175. ↩

3. Barlow, Kyle A., et al. "Flex ddG: Rosetta ensemble-based estimation of changes in protein–protein binding affinity upon mutation." The Journal of Physical Chemistry B 122.21 (2018): 5389-5399. ↩

4. Tiberti, Matteo, et al. "MutateX: an automated pipeline for in silico saturation mutagenesis of protein structures and structural ensembles." Briefings in Bioinformatics 23.3 (2022): bbac074. ↩