RosettaDDGPrediction User Guide

Rosetta version: 3.12

Last updated: December 10th, 2020

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 incorporated two different procedures in our package, each one having two variants, for a total of four protocols currently available in RosettaDDGPrediction.

`cartddg` protocols

Two of the aforementioned protocols belong to the `cartddg` family and serve the purpose of predicting of the ΔΔG of stability of a monomeric protein upon mutation, according to the procedure devised by Park et al. [1].

These protocols are named `cartddg_ref2015` and `cartddg_talaris2014`, where `ref2015` and `talaris2014` refer to the family of the scoring function used.

Both protocols include three steps:

1. `relax`: the preprocessing of the input PDB structure, that consists in a relaxation in Cartesian space of the structure, performed with the Rosetta `relax` application.
2. `structure_selection`: the selection of the lowest energy structure produced by `relax`, that will be fed to the next step. This step is performed internally by Python.
3. `cartesian`: 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 in order to produce the relaxed structure. Then, this structure would serve as input for as many runs as needed to predict the ΔΔG associated to the mutations of interest.

In order to perform only the `relax` or the `cartesian` step, use the `relax_*.yaml` and `cartesian_*.yaml` files in `config_run`. 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 other two protocols belong to the `flexddg` family, and are employed to predict the ΔΔG of binding of a protein complex upon mutation. Those protocols are named `flexddg_ref2015` and `flexddg_talaris2014`, both coming from the work of Barlow et al. [2]. As their `cartddg` counterparts, they differ in the Rosetta energy function used for the calculations.

These two protocols make use of a XML script implementing the procedure devised by Barlow et al. [2]. A slightly modified version of the XML script that the authors provide in their Flex ddG tutorial (which can be found here) is provided by RosettaDDGPrediction, inside the `RosettaScripts` directory.

Both protocol consist in only one step, `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 three steps:

1. Data generation, performed by the `rosetta_ddg_run` executable, where the ΔΔG prediction occurs.
2. Data aggregation, performed by the `rosetta_ddg_aggregate` executable, where raw data are aggregated in CSV files for all mutations included in a run.
3. Plotting, 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 behaviour 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 one defining 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, containing settings for the runs that do not relate to the specific protocol being run. These include options for parallelization, use of MPI, etc.
- `config_aggregate` files, which 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, containing both options for the format of the output file containing the plot and for the

plot aesthetics.

- `rosetta_ddg_run`

This 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

`$INSTALLDIR` indicates the directory where you installed RosettaDDGPrediction.

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 `$INSTALLDIR/RosettaDDGPrediction/config_run` .
`-cs` , `--configfile-settings`	Configuration file containing settings to be used for the run. If it is a name without extension, it is assumed to be the name of a YAML file in `$INSTALLDIR/RosettaDDGPrediction/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

Mutations/Positions 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, such as `flexddg` protocols, require you also to provide 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 refer to the Rosetta documentation and to the Flex ddG tutorial for more details). Such 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
```

When positions instead of mutations are specified in the file, you just have to omit the mutated residue. Please be aware that multiple simultaneous mutations are not supported when running saturation mutagenesis scans on positions. Extra pieces of information are passed exactly as you would do with a list of mutations.

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

Residue 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 (see here for more details).

```
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 mutation file as the first one presented in the previous section and you ran the `cartddg_ref2015` protocol with the default configuration file. 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 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. See here for more details about Roseta flags files.

`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. See here for an example of a Rosetta mutfile.

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

`flexddg` protocols

Suppose you ran the `flexddg_ref2015` protocol with the default configuration file, using the second mutations file shown in the previous section. This would be the directory tree you would find 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 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 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.

`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 three `*.pdb` files found in each folder, that are also intermediate files.

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]
```

Options

Option	Meaning
`-h` , `--help`	Show the help message and exit.
`-cr` , `--configfile-run`	Configuration file of the protocol that was run. If it is a name without extension, it is assumed to be the name of a YAML file in `$INSTALLDIR/RosettaDDGPrediction/config_run` .
`-ca` , `--configfile-aggregate`	Configuration file for data aggregation. If it is a name without extension, it is assumed to be the name of a YAML file in `$INSTALLDIR/RosettaDDGPrediction/config_aggregate` . Default is `$INSTALLDIR/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 aggregated data. Default is the current working directory.

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` 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) averaged over all the structures of the ensemble generated by the protocol.

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.

rosetta_ddg_plot

Command line

```
rosetta_ddg_plot [-h] -i INFILE -o OUTFILE [-ca CONFIGFILE_AGGREGATE]
                 -cp CONFIGFILE_PLOT [-d2m D2MFILE]
```

Options

Option	Description
`-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 used for data aggregation. If it is a name without extension, it is assumed to be the name of a YAML file in `$INSTALLDIR/RosettaDDGPrediction/config_aggregate` .
`-cp` , `--configfile-plot`	Configuration file for plotting. If it is a name without extension, it is assumed to be the name of a YAML file in `$INSTALLDIR/RosettaDDGPrediction/config_plot` .
`-d2m` , `--d2mfile`	File mapping the names of the directories containing mutations included in a saturation mutagenesis scan to the mutations themselves, represented as in the mutations' list file used to perform the scan. Required only if the plot is total_heatmap_saturation.

Plot types

Four different 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 : `config_plot/total_heatmap.yaml` .

- `total_heatmap_saturation`

  Notes : suitable for data produced by saturation mutagenesis scans.

  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 (see the `total_heatmap_saturation.yaml` configuration file in `config_plot` for more details).

  Example of configuration file : `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. These settings, together with other plot aesthetics, can be tuned in the `contributions_barplot.yaml` configuration file in `config_plot` .

  Example of configuration file : `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 : `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] 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. ↩ ↩