MutateX

Contents

1 Requirements and installation

MutateX and the associated scripts are written in Python, and requires having a working Python 2.x or 3.x (x >= 7) installation. A number of Python packages need also to be available. More in details, MutateX requires:

- setuptools
- biopython
- matplotlib
- numpy
- scipy
- six
- openpyxl

All the mentioned packages are available for free for Linux and macOS. Once these requirements have been satisfied, MutateX needs to be installed as a standard Python package, using the included setup.py file.

Please follow detailed the installation instructions in the INSTALL file included with the distribution to install the software.

Since MutateX works by running the FoldX software, the FoldX binary and the associated rotabase.txt file should be available. Please refer to the FoldX website (http://http://foldx.crg.es) to aquire it.

1.1 Supported FoldX versions

MutateX supports FoldX Suite version 4 and 5.

The distribution for version 4 includes a rotabase.txt file which si necessary for it to work; however, the FoldX Suite 5 binary generates its own rotabase file and it's not a requirement as such. See section 2.2 for details.

2 Running calculations with MutateX

2.1 The MutateX protocol

MutateX uses FoldX to systematically calculate the difference in free energy of folding ($\Delta\Delta G$) between mutated variants and the wild-type using the FoldX BuildModel command. More in details, one or more protein models provided by the user are mutated sequentially by indipendently changing the residue at each position of the protein sequence to another aminoacid. The list of residues that each position will be mutated to can be provided by the user, or defaults to the list of 20 canonical aminoacids found in proteins. For instance, the user

can provide a PDB file containing the structure of a 50-residues protein, and demand each of them to be changed to ALA, PHE and ASP. In this case, 150 (50*3) FoldX runs will be performed and the FoldX $\Delta\Delta G$ value calculated for each of them. Similarly, if more than two protein chains are provided, MutateX can calculate the change of free energy of interaction upon mutation for all the tested mutations, if requested. Note that MutateX also allows performing partial mutational scans, in which only few positions at a time are considered. The protocol followed by MutateX can be summarized as follows:

1. Load and check the structure(s). One (or more) PDB file containing one (or more) models may be employed. See 3.1 for details.
2. Repair. Each model is subject to a Repair run in which residues having bad torsion angles, Van der Waals clashes or bad total energy are modified to more reasonable conformations. This uses the RepairPDB command of FoldX.
3. Mutate. Each residue at every position is indipendently is mutated to a number of selected aminoacids as previously detailed, and $\Delta\Delta G$ values are calculated.
   (a) Optional: calculate $\Delta\Delta G$ of interaction between protein chains upon mutation, using the AnalyseComplex command.
4. Report. The $\Delta\Delta G$ are gathered in a user-friendly collection of text files. Also, $\Delta\Delta G$ values are averaged among the mutations calculated on different models and the values are reported.

2.2 Required files

The following files are required for a MutateX run:

- By default, MutateX reads the content of the FOLDX_BINARY and FOLDX_ROTABASE system variables in order to determine their location (see Section 1 for details). If any of those variables is empty, the location of the respective file can be supplied through the –foldx-binary or –rotabase command line options, respectively. If the command line options are supplied the content of the system variables is ignored. Notice that the rotabase file is a requirement for FoldX Suite 4 but not for FoldX suite 5, namely:
  - If you're using FoldX 4 the rotabase file needs to be supplied as described
  - If you're using FoldX 5, there's no need to supply a rotabase file as the software will generate one automatically when it's not present. However, if a rotabase file is specified in any of the options above, the custom rotabase file will be used instead
  - You should make MutateX aware of the version you intend to use by using the –foldx-version option, specifying either suite4 or suite5 accordingly

- One (or more) PDB file, containing one (or more) models of the protein to be mutated. The PDB files need to adhere to the following guidelines:
  - The models should all have the same sequence. If this not the case, MutateX will detect the problem and exit after repair
  - The chain identifier should be provided for all the chains of all the models. This is especially important if the supplied models have more than one chain
  - The PDB file should contain exclusively residues recognized by FoldX. See http://foldxsuite.crg.eu/allowed-residues for a complete list. Also notice that residues (even those with post-translational modifications such as phosphorilated residues) should be defined as whole residues under a single residues identifier. For instance, phosphorylated residues where the phosphate group is separated from the residue and defined as HETATM at the end of the PDB may not be recognized correctly.
  - The number of residues or atoms should monotonically increase progressing along the molecule (e.g., the first four residues of a chain should follow the order 1-2-3-4, and 1-3-2-4 is not ammissible.

- Two template files: one for the repair and one for the mutation runs. These should be named, respectively, repair_runfile_template.txt and mutate_runfile_template.txt. MutateX looks for these files in the working directory (i.e. where it is run). Nonetheless, it is also possible to specify custom filenames or locations using the –repair-runfile-template and –mutate-runfile-template options. Two examples of these template files are available within the MutateX package. These are normal FoldX RunFiles that perform a PDBRepair and BuildModel respectively, except for two details. In the two files, one "$PDBS$" and one "$NRUNS$" strings are present. These have to remain as such for MutateX to work correctly (i.e. the user should not modify the respective lines). Nonetheless, the user is free to modify all the other options as fit for the task at hand. If the $\Delta\Delta G$ of interaction needs to be calculated, a third template file needs to be provided, either for protein complex or complex with DNA. Examples for template files are available with the MutateX distribution.

- Optionally, one mutations list file (usually named mutations_list.txt). This file specifies the ordered list of residue types that each residue of the input models should mutated to in order to calculate the values of $\Delta\Delta G$ (see section 2.1 for details). The same order is also used in the output of the reporting of $\Delta\Delta G$. The file is a list of canonical aminoacids specified in the single-letter code, one per line. Each line should begin with the single-letter specifying the aminoacid type, and all the other characters are not considered. For instance, one could use:

```
A
C
D
E
```

as well as

```
A (Alanine)
C (Cysteine)
D (Aspartate)
E (Glutamate)
```

only the first letter matters. Be careful: this means that a mutations list file like

```
ALA
CYS
ASP
GLU
```

would be the same as

```
A (Ala)
C (Cys)
A (Ala)
G (Gly)
```

The mutation list file is provided by the user using the –mutlist command line option. If this option is not specified, the default mutation list will be used, which is:

```
G
A
V
L
I
M
F
W
P
S
T
C
Y
N
Q
D
E
K
R
H
```

2.3 Supported residue types

MutateX supports all protein residue types supported by FoldX as detailed in their user guide (see http://foldxsuite.crg.eu/allowed-residues). In particular,

PTM'd aminoacid	FoldX/MutateX single letter code	Residue name in PDB
Phophorylated threonine	p	TPO
Phosphorylated tyrosine	y	PTR
Phosphorylated serine	s	SEP
Hydroxiproline	h	HYP
Sulfortyrosine	z	TYS
Monomethylated lysine	k	MLZ
Dimethylated lisine	m	MLY
Trimethylated lisine	l	M3L
Charged ND1 histidine	o	H1S
Charged NE2 histidine	e	H2S
Neutral histidine	f	H3S

Table 1: supported post-translationally modified residues

MutateX considers all 20 natural aminoacid types plus a few post-translationally modified residues and specific tautomeric states for histidine. These are treated exactly in the same way as natural amino acids.

In user-defined mutation list files (see section 2.2) and in default plot labels the natural aminoacid types are represented by their respective single-letter code, while post-translationally modified amino acids are represented by lowercase letters, as detailed in table 1.

2.4 Running MutateX

2.4.1 Basic usage

If everything is set up as described in sections 1 and 2.2, the simplest MutateX command is simply:

```
mutatex single_model.pdb
```

More than one file may be supplied at once:

```
mutatex single_model1.pdb single_model2.pdb
```

Please notice that each PDB file is required to contain only one model, and MutateX will complain and exit if this not the case. This is because we require the user to acknowledge that he is using multi-model files, and that all the models in the provided PDB file will be used. This is performed by adding an option to the command:

```
mutatex single_model1.pdb multiple_models.pdb --multiple-models
```

If you have a multi-core machine (as you most probably do) you can greatly shorten the calculation times by allowing MutateX to run more than one instance of FoldX at the same time, using option –np, which specifies the number of FoldX processes to be run at the same time:

```
mutatex single_model.pdb --np 4
```

You can also specify a customized mutation list file:

```
mutatex single_model.pdb --np 4 --mutlist mutations_list.txt
```

2.4.2 Other options

In this section, several options available in MutateX are described.

```
--nruns NRUNS
```

This is the number of mutation runs to be performed by MutateX per position and residue type of mutation. This is important since for each run the algorithm might explore different conformations for mutagenesis. As to keep the FoldX run as informative as possible, we have decided to set this value to 5 by default.

```
--foldx-version {suite4, suite5}
```

This option is used to tell MutateX which FoldX version it should expect. Right now, FoldX Suite 4 or 5 are supported.

```
--verbose
```

Activate verbose mode, off by default.

```
--foldx-log
```

Write the standard output of the FoldX runs to a log file in the run directories.

```
--binding-interface
```

If more than two chains are present, calculate difference of free energy of interaction

```
--clean {partial,deep,none}
```

Clean the directories where runs are performed, after performing them. "none" performs no cleaning. Partial removes all the generated PDB files. Deep removes all files except for the fxout result files. Default is partial.

```
--compress
```

Compress mutations folder after the run is over

```
--no-multimers
```

By default, if more than one protein chain is present, MutateX automatically identifies if two protein chains have the same sequence and in that case they are considered multimers. This means that all the chains with the same sequence will be mutated at the same time, in order to simulate a multimer of mutants. By using this option, multimers won't be considered and all chains will be mutated independently.

```
--poslist
```

MutateX supports partial mutational scans if requested. This is possible by providing a list of desired positions in a text file, one per line. Positions are defined in a similar format as the output files under the "results" folder, so [One-letter wild-type residue type][Chain ID][Residue number]. Residue types are optional. If multimer mode is on, you just need to specificy one residue of a multi-chain complex to select the corresponding set of residues in all the chains. For instance, if you only specify AA35 and AB35 is also available, both will be mutated. Likewise, you can use the MutateX file name syntax and ask for AA35_AB35. As mentioned, residue types are optional, meaning that A35_B35 will work as well. A position list file could look something like this:

```
CA2
A4
B8
CA2_CB2
A6_B6
```

Plotting tools (see below) also support partial scans through a specific option.

```
--self-mutate
```

Turns on self-mutation mode. When this option is used, the mutation list (either provided by the user or the default one) is ignored. Instead, each residue is mutated to the type it already has, using templates and options as per a normal mutational scan. This is intended as a sanity check fo the original repaired structure - if the free energy differences are significant when mutating a residue to itself, it's possible it hasn't been properly optimized during repair and we should be careful in interpreting the results for that residue. The runs are performed in a folder called selfmutations, while results are available in the selfmutation_results folder. The results are available as a table in which, for each residue, the maximum, minimum, average and standard deviation of self-mutation free energy difference are reported, calculated over FoldX multiple runs. This is performed both for stability and interaction energy, if requested.

```
--skip-repair
```

When this option is used, the repair phase is not performed on the input models and the original PDB files are used instead. The calculation proceeds are requested fo the rest.

2.5 Interrupting and restarting a MutateX run

In order to interrupt a MutateX run just kill the MutateX script right away.

If a MutateX run is interrupted, you don't have to repeat it from scratch. Just enter the directory where the job was running and launch *the very same* MutateX command line (therefore, it is advisable to save it as a script that will be used to run the program). MutateX will automatically understand what

has been already done and what's missing, and will restart from where he had left. Be careful though; you shouldn't modify any of the input our outpur file of FoldX for the restart to work as expected. FoldX is able to recognize certain inconsistencies between the available input and the available finished FoldX runs, but mostly expects that the same command with same input has been issued.

The fact that MutateX supports smart restarting means that you should *never run two different MutateX calculations in the same working directory.*

3 MutateX Output

MutateX automatically prepares the FoldX runs necessary for the completion of the complete mutation scan, meaning that it writes a number of directories and files on disk. By default, the four FoldX phases (as described in section 2.1) are run one after the other, as each of the requires the output of the previous phase.

From this moment on, each phase will be illustrated using as example a generic PDB named "structure.pdb" as example. This generic PDB file contains a single model containing a single chain ("A") with six residues having a specific protein sequence (MATELE), and residues having residue number from 1 to 6.

In the following description, we will often refer to the "current working directory": this is the directory in which the input files are present and from which MutateX has been launched. It will be also referred as CWD for convenience.

3.1 Phase 1: Preliminary operations and model check

MutateX first reads the PDB file(s) it will work on and checks it can be read correctly. If this is the case, and if the PDB file contains a single model, it is used as such, while the program exits with an error if the PDB contains multiple models. If option -a is provided, however, any multi-model PDB file is split into one PDB file per model and all models are considered. This is done to ensure that the user is fully aware of the fact that they are using multiple models. A further check is performed after phase 2 (Repair) to ensure that all the PDB files under consideration have exactly the same primary sequence.

3.2 Phase 2: Repair

On each of the models saved as single PDB files in the previous section, a FoldX Repair run is performed, following the protocol detailed in the template file provided by the user. FoldX creates the "repair" directory, and each repair run is performed inside a subdirectory of such directory. For instance, in our example there would be only one repair directory under the "repair" one, which is named "repair_structure_model0_checked". Inside it, MutateX writes all the files that are necessary for the run and executes it automatically.

For instance, the following directory structure has to expected in our example:

```
CWD
└── repair
    └── repair_structure_model0_checked
```

3.3 Phase 3: Mutate

In this phase, the mutation runs are performed. A "mutations" directory is first created, under the CWD. Inside of it, a directory is created for every repair run which was succesfully completed in the previous phase. The directory name will derive from the output files written by FoldX in the repair runs - in our example, that would be "RepairPDB_structure_model0_checked". Finally, inside of these directories, several directories will be present, each corresponding to a protein residue and each containing a complete mutation runs. The naming of these directory will follow this convention: [Single-letter residue type of the wild-type residue][chain name][residue number]. For instance, the following directory structure has to expected in our example:

```
CWD
└── mutations
    └── structure_model0_checked_Repair
        ├── MA1
        ├── AA2
        ├── TA3
        ├── EA4
        ├── LA5
        └── EA6
```

3.3.1 Optional: Free energy of interaction

If the calculation of free energy of interaction has been requested, this is performed directly in the mutations directories.

3.4 Phase 4: Report

After running all the mutations runs as detailed in Phase 3, the values of $\Delta\Delta G$ gathered from the mutation runs are written in text files. This happens in the "results" directory under CWD. Inside this, one directory per model used is found, and each of them contains several text files, named as mutation directories in the Mutate phase. Each of them contains a list of $\Delta\Delta G$ values (in kcal/mol), one for each residue type specified in the mutation_list.txt (or the default list if a file has not been specified) and in the same order. These values are the average of the N FoldX run that have been carried out for each mutation, as specified by the –nruns command line option (see section 2.4.2).

Finally, inside the "results" diretory, a final_averages directory will also be present. This contains a number of text file, in the same exact format as the different result directories each referring to a single model. The values written in the files, however, represent the average $\Delta\Delta G$ across all the models analyzed.

```
CWD
└── results
    └── mutation_ddgs
        ├── structure_model0_checked_Repair
        └── final_averages
```

3.4.1 Optional: Free energy of interaction

If the calculation of free energy of interaction has been requested, the corresponding changes in free energy are stored similarly as in the mutations results directory, however in a different position. The interaction can be between different protein chains or between protein and DNA. See the different Interaction templates. If the interaction is between two protein chains, the interaction energies are calculated for pairs of chains (chains A and B in the below example). If a DNA complex is analyzed, the interaction will be illustrated based on the chain names in the pdb file, i.e A-BC, if BC is the two strands of DNA.

```
CWD
└── results
    └── interface_ddgs
        └── A-B
            ├── structure_model0_checked_Repair
            └── final_averages
CWD
└── results
    └── interface_ddgs
        └── A-BC
            ├── structure_model0_checked_Repair
            └── final_averages
```

4 Processing the output of MutateX

4.1 Visualizing the output of MutateX

Visualization of MutateX output can be performed using a set of supplied tools for graphical representation. These include scripts for generating: heatmaps, stemplots, boxplots and more to visualize the calcualted free energies, either of folding or interaction. Supporting these tools is a script that generates a .csv file that allows you to add custom labels to the residues used to generate plots. All the scripts detailed here can be run with a "-h" option to print a helpful description of their purpose and the supported options.

4.1.1 Using a position list

By default, all plotting tools assume that a full mutational scan has been performed. If only a partial one has been performed by using a position list file (option -q), the same position list file has to be provided to the plotting tools as an additional input, again with option -q. All our tools support position list files except for ddg2pdb, which expect a full mutational scan, ddg2histo and ddg2summary, both which support different mechanisms to select mutation sites or mutations (see below).

Providing a mutation list can also be useful to limit plotting and reporting to a certain set of residues, even though a larger a scan has been performed. You can see an example in section 4.1.3.

4.1.2 Labels

The pdb2labels tool may be used to generate a .csv file of the input ΔΔGs folder. This .csv file is used in all other plotting tools to add optional custom labels as residue names instead of those generated by FoldX. pdb2labels is run by specifying one of the PDB models used for the calculation:

```
pdb2labels -p single_model.pdb
```

pdb2labels will generate a comma seperated .csv file containing 2 columns: "Residue name" and "label". You can open the .csv file with Microsoft Excel or similar programs to add the labels in the second column on the same row as the corrrsponding residue. When the changes are saved the file can be included in all other command line plotting tools with the -b (--label) parameter and the file path to the .csv file.

```
-b 3i3c_labels.csv
```

Labels are optional. When the second column is left empty for any residue the plotting tools will simply use the residue name.

4.1.3 Heatmap

The ddg2heatmap tool may be used to generate heatmaps of the calculated mutation ΔΔGs. ddg2heatmap is run by specifying the path to one of the "results" directory. The final_averages can be used or, if a specific protein model is of interest, the name of the model file (see previous section). In addition to the results directory, the original model.pdb file and list of mutations must be supplied.

By default wildtype amino acids are represented on the y-axis and mutated-to amino acids on the x-axis, however, using the option -t (transpose) these may be swapped. Option -s defines the number of residues to be plotted per each plot, and -n and -x control the plotted range of free energy values. Values outside the range are changed to the limits of the range, and are simply to be interpreted as "very destabilizing" or "very stabilizing".

A typical example of commandline for this tool is:

```
ddg2heatmap -p 1D5R_clean_model0_checked.pdb
-l mutation_list.txt
-d results/mutation_ddgs/1D5R_clean_model0_checked_Repair
-s 25 -n -3.0 -x 5.0
```

The resulting plot is shown in figure 1

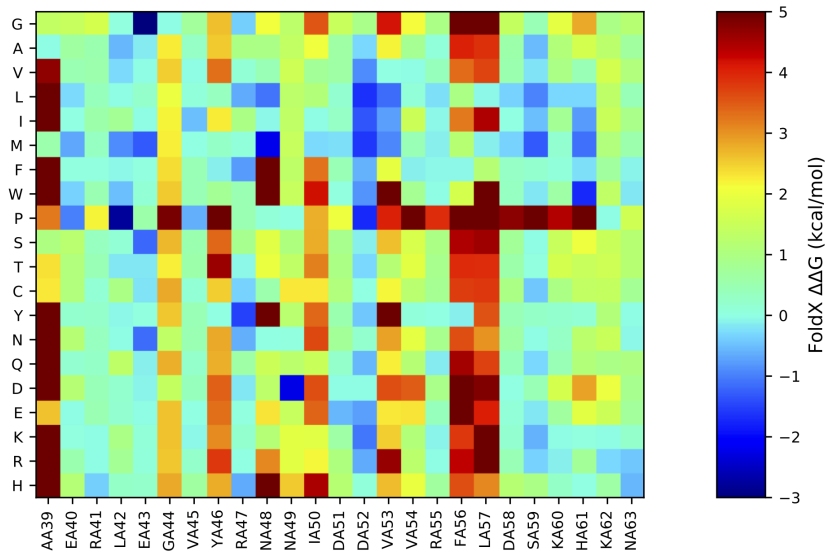


Figure 1: Heatmap generated using ddg2heatmap.

It is also possible to provide a custom position list to this and other tools (see section 4.1.1). In this case we provide a poslist file with the following content, therefore including 8 custom residues:

```
AA39
GA44
YA46
NA48
NA49
NA50
FA56
LA57
```

we then run ddg2heatmap providing the poslist file with option -q, obtaining the plot in figure 2:

```
ddg2heatmap -p 1D5R_clean_model0_checked.pdb
-l mutation_list.txt
-d results/mutation_ddgs/1D5R_clean_model0_checked_Repair
-n -3.0 -x 5.0
-q poslist.txt
```

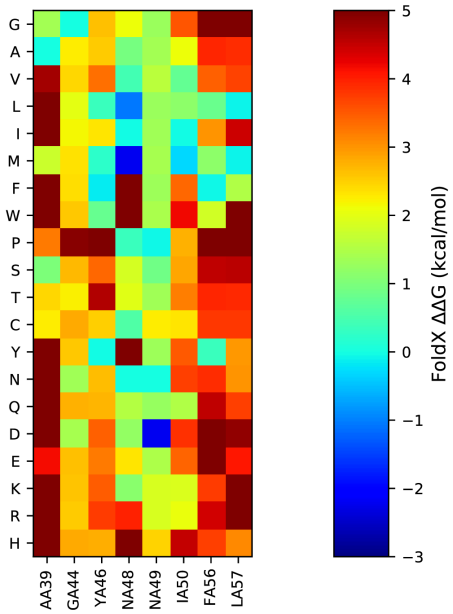


Figure 2: Heatmap generated using ddg2heatmap and poslist file.

4.1.4 Distribution plots

The ddg2distribution tool can be used to generate a number of different tpyes of plots each visualizing the distribution of the mean protein ΔΔG for any mutated wildtype amino acid. It can generate a stem plot, a box plot, a violin plot, average plot and scatter plot. The stem plot just plots the average ΔΔG for all the mutations of a given site, and the stems are set to the standard deviation of the mean. The boxplot shows the general distribution and outliers of the mean ΔΔG values for every mutation. The violinplot is similar to the boxplot but also shows a continue vertical probability density plot of the data on each side of the figures. The average plot shows the average per-site ΔΔG as a dot for each position, allowing to identify the outliers for the whole protein. The scatter plot shows the ΔΔG value for every single mutation, instead of a distribution.

Ddg2distribution may be run using some of the same inputs as for ddg2heatmap (see before). The path to the "results" directory (either final_averages or a specific protein model of interest), original pdb file and a list of mutated residue types are mandatory parameters:

```
ddg2distribution -d results/final_averages
-p single_model.pdb -l mutation_list.txt
```

Another mandatory parameter is the -T (--type) parameter. You use the -T (--type) parameter to select the type of plot you want to generate. You need to select at least one type.

```
-T {stem,box,violin,average,scatter}
```

The example below shows the command for running ddg2distribution for the stem plot - but all the available output plots can be seen in figures 3, 4, 5, 6 and 7.

```
ddg2distribution -p
3i3c_edit_model0_checked_Repair.pdb -d
/results/final_averages/ -l mutation_list.txt
-o distribution.png -s 25 -x 50 -n -10
-T stem
```

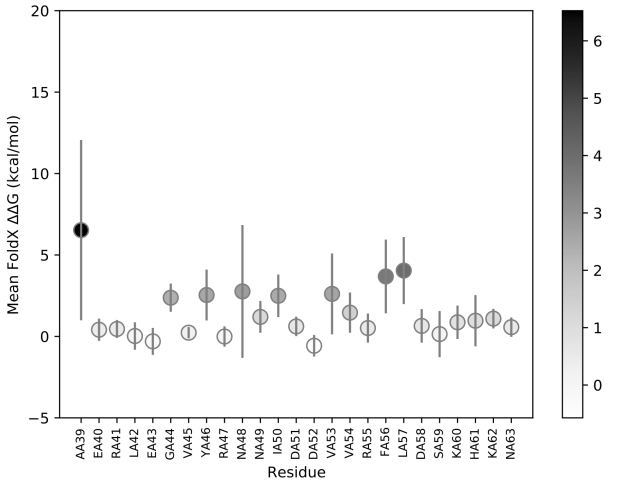


Figure 3: Stem plot generated from ddg2distribution.

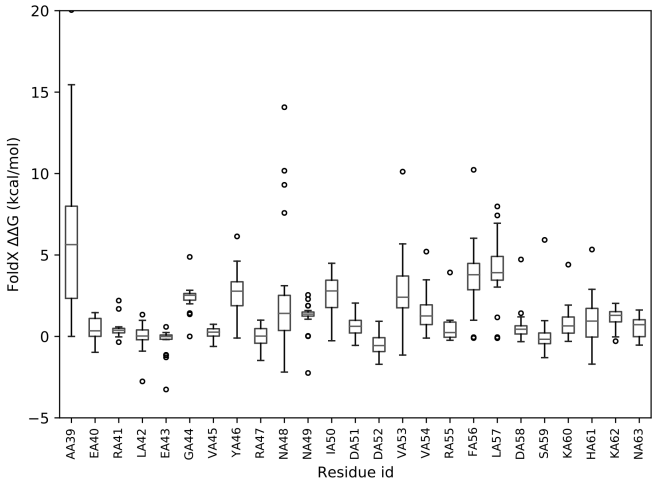


Figure 4: Boxplot generated from ddg2distribution.

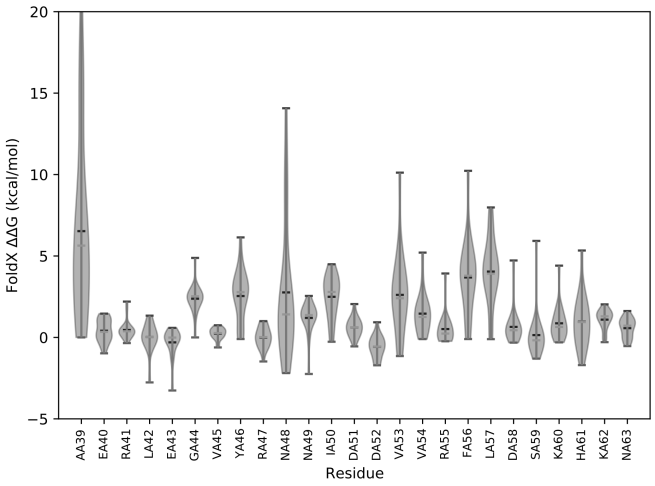


Figure 5: Violin plot generated from ddg2distribution.

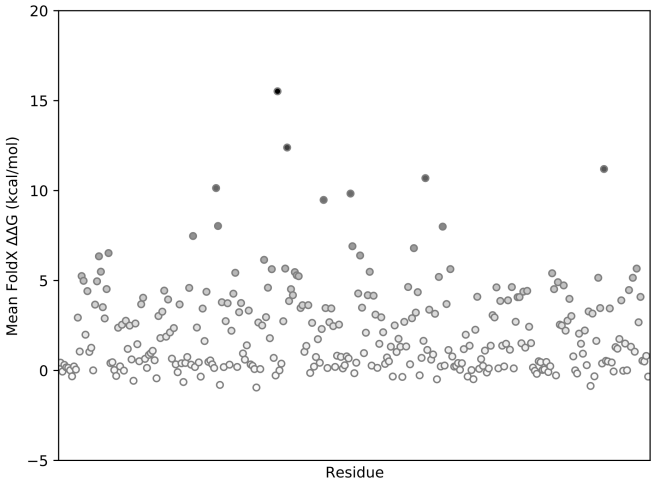


Figure 6: Average plot generated from ddg2distribution.

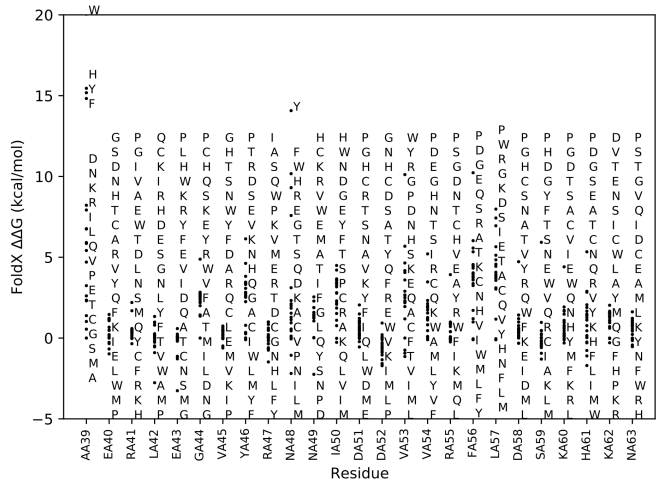


Figure 7: Scatter plot generated from ddg2distribution.

4.1.5 Histogram plots

The ddg2histo tool can be used to generate bar plots showing the distribution of mutatex $\Delta\Delta$G values for every mutation of aspecific wildtype amino acid. The user may define any number of wildtype amino acids to plot, as long as they are available in the calculated data set. Running the ddg2histo tool requires the same mandatory parameters as ddg2distribution and ddg2heatmap: the path to the "results" directory (either final_averages or a specific protein model of interest), original model.pdb file and a list of mutations.

```
ddg2histo -d results/final_averages
-p single_model.pdb -l mutation_list.txt
```

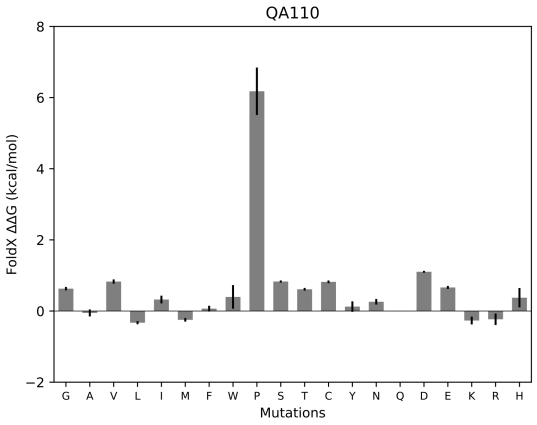
To specify for which residues you want to generate a plot you use the -r parameter and a comma-separeted list of residues and residue ranges. A residue range can be defined by specifying two residues separated by a '-'. The residues that specify the range have to be present in the dataset. The example below shows some valid residue selections.

```
-r AA16
-r AA16,AB31,AD55
-r AA16-AA68
-r AA68-AA16
-r AB31-WC44
-r AA16,AD31,AB16-AC50,AA68
```

The residues are ordered alphabetically, ignoring the first letter of the residue name (the amino acid). eg: DA14, IA15, AA16, EA24, IB15, EB24, KC61...

Residues have to be selected by simple single residue names. If a multimer contains atleast one of the selected residues, it will also be added.

The example below shows the command for running ddg2hist for residue Q110:




```
b -l mutation_list.txt
0_checked_Repair -r QA110
```

8.

Figure 8: Histogram of ΔΔG values for residue Q110 generated using ddg2hist.

4.1.6 Logo plot

ddg2logo generates pseudo-logo plots from ΔΔG values. While actual logo plots are based on letter frequency, these are based on the provided MutateX scan and are designed to highlight the mutation sites which specifically feature a limited number of stabilizing or destabilizing mutations, as chosen by the user (see below). For each site, the letters of the plot represent the (de)stabilizing mutations and their height is proportional to the (de)stabilizing effect (i.e. the absolute value of the respective change in free energy). If a certain site has more than a certain number of selected mutations, the respective column can be replaced by a X - this threshold value is controlled by the -x option. The behaviour of the tool is slightly different if stabilizing or destabilizing mutations are being plotted:

- To plot stabilizing mutations, the user needs to provide option -T with a free energy threshold in kcal/mol which must be zero or negative. If option -T is provided, before plotting:
  - all the ΔΔG values $\geq$ threshold are set to 0
  - the sign of all the ΔΔG values is changed (i.e. they are made positive)
- To plot destabilizing mutations, the user needs to provide option -t with a free energy threshold in kcal/mol which must be zero or positive. If option -t is provided, before plotting:
  - all the ΔΔG values $\leq$ threshold are set to 0

This allows to obtain a 0-based positive-only plot in which the ratio between the energy values is the same as the original data.

A typical command line for the tool is:

```
ddg2logo -p 1D5R_clean_model0_checked.pdb
-l mutation_list.txt
-d results/mutation_ddgs/1D5R_clean_model0_checked_Repair
-s 25 -D 300
-t 1.2
```

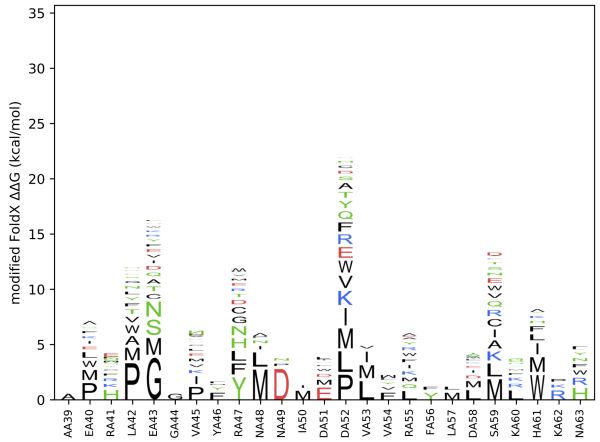


Figure 9: Logo plot obtained using ddg2logo.

4.1.7 PDB file and matrix file.

The ddg2pdb tool can be used to write the outcome of the mutational scan as b-factors in PDB files, as well as writing input files compatible with the xPyder PyMOL plugin. Running ddg2pdb requires the path to the "results" directory (either final_averages or a specific protein model of interest), the original model pdb file and list of mutations. Additionally, an upper and lower cut-off of protein $\Delta\Delta$G is needed, as the tool will save for each residue the number of mutations which are within the specified parameters. Default cut-off values are: lower = -99999.0 and upper = 99999.0. If the ddg2pdb option -t is set to "between" the tool will extract the values between upper and lower. Alternatively -t may be set to "outside" specifying that everything $<$ then the lower cut-off and/or everything $>$ than the upper sould be included.

For full list of user specified options see the ddg2pdb help option.

```
ddg2pdb -h
```

A typical command line for ddg2pdb is as follows. Cut-offs are set to between lower = 1.0 and upper = 99999.0 in order to obtain for each residue the number of destabilizing mutations (defined as $\Delta\Delta$G $>=$ 1.0).

```
ddg2pdb -p 1D5R_clean_model0_checked.pdb -l mutation_list.txt
-d results/mutation_ddgs/1D5R_clean_model0_checked_Repair
-a 3.0 -b 99999.0 -T between
```

4.1.8 Energy distribution of all the mutations

The ddg2density tool can produce a density plot of all the calculated differences in free energies, after removing self-mutations, by employing kernel density estimation as implemented in the scipy package. The typical command line is:

```
ddg2density -p 1D5R_clean_model0_checked.pdb
-l mutation_list.txt
-d results/mutation_ddgs/1D5R_clean_model0_checked_Repair
```

The output of the tool include the plot of the density, the density values and relative energy values as text files, as well as the energy values used for the calculation. The plot shows the calculated density and all the single energy values identified below the x-axis.

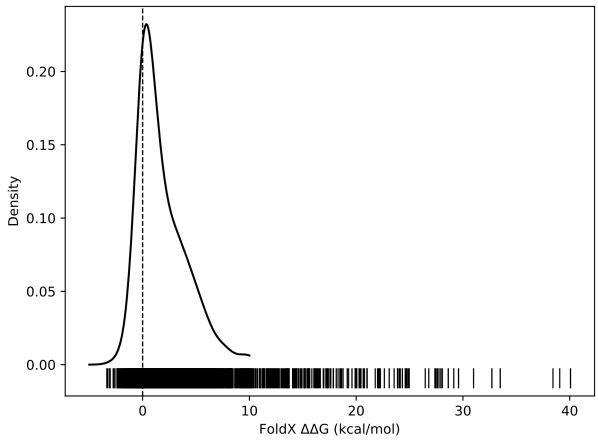


Figure 10: Density plot obtained using ddg2density.

4.2 Processing and converting the output of MutateX

4.2.1 Excel Sheet

ddg2excel is a tool for exporting results as a table in either an excel sheet or a comma-separated file. The excel file is the default format but can be changed by optionally adding -F csv. The table has the same basic structure as the ddg2heatmap tool. To get the table, a typical command line for this tool is:

```
ddg2excel -p 1D5R_clean_model0_checked.pdb
-l mutation_list.txt
-d 1D5R_clean_model0_checked_Repair
```

4.2.2 Linear correction of the energy values

Using the ddg2dg tool, it's possible to apply a linear correction to the calculated energy values, in the form:

$$\Delta\Delta G_{mod} = x + DDG_{orig} * m + y$$

Where $\Delta\Delta G_{orig}$ are the original energy values. x, m and y can be specified by options on the command line. The tool accepts a MutateX results directory with option -d, and writes another directory with exactly the same format and the modified values. For instance, the following command line can be used:

```
ddg2dg -p 1D5R_clean_model0_checked.pdb
-d results/mutation_ddgs/1D5R_clean_model0_checked_Repair
-m 1 -y 0 -x 0
```

4.2.3 Summary of interesting mutations

The ddg2summary tool writes a text file with the average $\Delta\Delta G$ values of specific mutations. The mutations are specified in a text file that is mandatory. The file contains one mutation per line, each of them needs to be in the form: [WT residue type][chain][residue number][mutant residue]. The wild-type residue type can be omitted. The following is a typical command line for the tool:

```
ddg2summary -p 1D5R_clean_model0_checked.pdb
-l mutation_list.txt
-d results/mutation_ddgs/1D5R_clean_model0_checked_Repair
```

```
-L mutations_of_interest.txt
```

5 FAQ

Q: Is it possible to include non-standard (for instance, phosphorilated) residues in the mutation list?

A: Yes it is! See the following web page for the list of supported residues: http://foldxsuite.crg.eu/allowed-residues