Rosetta Guide for the iGEM beginner

Made by: iGEM Technion 2016
iGEM TU Eindhoven 2016

Table of contents

About this guide 3
About Rosetta 4
Do you need a local installation of Rosetta? 5
Online Services 5
Getting Rosetta 6
Before you start 7
What do you need? 7
Supporting software 7
Structure Databases 11
I want to do X 13
Add an unknown residue to my simulations 13
Preparing structures for use in Rosetta 13
Finding important Residues for a protein binding interface 13
Designing an orthogonal binding interaction 14
Design of a binding pocket/interface for a chemical compound 14
Protocols 15
Cleaning your pdb 15
Adding unknown residues 15
Relax 16
Backrub 17
Computational alanine scanning 18
Point mutant scanning 20
Extensive remodeling 23
Design of Ligand Binding Sites 23
Tips and tricks 24
Rosetta Energy - how to filter results 25
Scoring Proteins 25
Important links, support and more data 31
Thanks and Acknowledgements 32
iGEM Technion 2016 32
iGEM TU Eindhoven 2016 32

About this guide

This guide was written by iGEM Technion 2016 and iGEM TU Eindhoven 2016 to be used as a starting point for new users of the Rosetta software for protein modeling and design.

This document is in no way a complete guide to Rosetta but rather an organized collection of all the important information we came across while using this powerful tool – articles, protocols, forum posts and our own personal experience.

We hope future iGEM teams will find this guide useful during their very first steps with protein modeling and design using Rosetta.

About Rosetta

Rosetta is a software suite for macromolecular modeling. It was initially developed to predict protein folding and has since been greatly expanded to include dozens of other options. As of 2016 it has been used to predict protein structures, perform protein – protein and protein – ligand docking, design novel proteins and redesign existing ones just to name a few.

Today, Rosetta algorithms are able to predict, design and analyze almost every set of biomolecular systems: proteins, RNA, DNA, Peptides, small molecules and non-canonical amino acids.

All of the above is available to the end user in two ways:

- A set of nearly 300 premade functions that can be used to perform specific tasks.
- Two frameworks, PyRosetta and RosettaScripts, which allow customization and creation of protocols for yet undefined tasks.

It is worth noting that successfully designing a protein in Rosetta does not guarantee its successful function in *vivo*. That is the case with every biological computational design. Testing your designs is a crucial part of

Do you need a local installation of Rosetta?

Given the large scale of the "typical" iGEM project, almost every team can find themselves needing one or more functions available in Rosetta. In most cases, if the task is simple enough (structure or docking predictions for example) it can be completed without directly using the software, saving both precious time and resources.

Before diving into the official documentation (or the rest of this guide) to figure out how to run Rosetta, check the following online services to see if your task can be performed automatically.

Online Services

These servers were developed to give the biological community the ability to use the most common Rosetta functions easily and freely without the need to understand every minor detail in the program.

- ROSIE – **R**osetta **O**nline **S**erver that **I**ncludes **E**veryone http://rosie.rosettacommons.org/
  ROSIE is a web framework for Rosetta applications run by RosettaCommons. It provides users access to computer cluster resources and a common user interface for simple use of several Rosetta protocols. As of September 2016, 18 Rosetta protocols can be run in ROSIE.
- ROBETTA – Full chain Protein Structure Prediction Server http://robetta.bakerlab.org/
  ROBETTA is a full chain structure prediction server run by the University of Washington. Aside from structure prediction and 3d modeling, ROBETTA offers fragment library generation (pieces of experimentally determined structures that Rosetta uses in the structure prediction process) and interface Alanine scanning (estimate the energetic contribution to the binding energy provided by each residue at a protein-protein interface).
- FlexPepDock – High resolution modeling of peptide-protein interactions http://flexpepdock.furmanlab.cs.huji.ac.il/
  FlexPepDock is a high resolution peptide-protein docking server run by the University of Jerusalem.
- Phyre2 - Protein Homology/Analogy Recognition Engine http://www.sbg.bio.ic.ac.uk/phyre2/html/page.cgi?id=index
  Phyre2 is a protein structure prediction service run by Imperial College London. It is free for non-commercial users and extremely easy to use. Although it is not based on Rosetta, it provides fast and reliable results

and is considered one of the best structure prediction services online. (Only for proteins with less than 2000 amino acids)

Getting Rosetta

An academic license for Rosetta is freely available, in order to receive this license you have to fill in an application form at the site of the University of Washington, they will then evaluate whether you are suitable for an academic license:
https://els.comotion.uw.edu/express_license_technologies/rosetta
Be sure to check whether your institution/organisation doesn't already have an (academic) license before you apply, as the evaluation might take a while.

Unlike most windows programs, Linux and Mac do not work with executables, so you will need to compile the software yourself, if you're using your institutions cluster it is also possible that, Rosetta is already installed, in which case you can skip this step.

In order to compile Rosetta and use it you will need the Rosetta source code, Python and the Scons compiler.

Using your license, you can download the Rosetta source code at:
https://www.rosettacommons.org/software/license-and-download

To Install Rosetta, you must first download and install Scons (Software CONStructor) which will build the Rosetta installation automatically. Scons requires Python 2.4 or above (Python 3 currently not supported).

Python is freely available at: https://www.python.org/

Scons is freely available at: http://scons.org/

Rosetta's Documentation for compiling and testing for the software and troubleshooting is available at:
https://www.rosettacommons.org/docs/latest/build_documentation/Build-Documentatio n

If you cannot install one or more of these softwares because you have no access/administrative rights to parts of a cluster, consider using steps as defined in this

Before you start

What do you need?

If your task does require a local version of Rosetta, the first thing you need to secure is substantial computing power. The simpler tasks, such as structure prediction or energy analysis, can probably run on a simple PC but it is time consuming. The heavier tasks, such as protein design, require large computing resources so use of a computer cluster is extremely recommended.

Currently, Rosetta runs only on UNIX based systems – Linux/Mac.
Basic knowledge of how to work and navigate in these systems is a must.
How to use Rosetta on a windows computer is documented in the chapter - Necessary software.

A few linux guides to get started

- ✓ A short youtube playlist introducing navigation and basic commands: https://www.youtube.com/watch?v=uJ39gAaeJsw&list=PLRUeXoFmWO_3CwGQvTNV92U G3HG9VXDK2&index=1
- ✓ Online tutorials: https://www.digitalocean.com/community/tutorials/basic-linux-navigation-and-file-mana gement

Supporting software

The input/output files you'll be working with are mainly PDB (Protein Data Bank) files for protein structures and mol2/smiles/sdf files for chemical ligands.

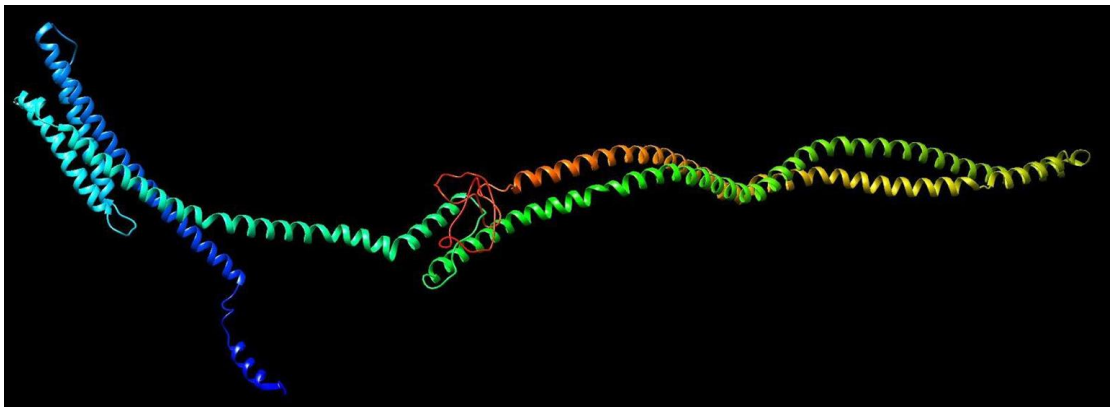
PDB files contain both the sequence and the 3D structure of the protein. Some proteins have more than one PDB file, each one showing the protein in a different conformation, for example bound and unbound states. PDB files may also sometimes contain chemical ligands which can be isolated from the file and saved separately if needed.

Ligand files contain the chemical structure and orientation of the ligand atoms and are similar to PDB files in purpose.

A dedicated software is needed to open and view all these file types correctly and perform actions on them. The two most popular programs to handle PDB and ligand files files are Pymol and Chimera UCSF.

Chimera UCSF

Chimera is a software for graphical display of proteins and small molecules. It can open any file type from online protein or ligand databases, present the 3D structure (if it exists), present and compare sequences and perform various other tasks like recording videos of proteins in motion (i.e. changing conformation or spinning to present them from all angles), running BLAST on a desired sequence and more.



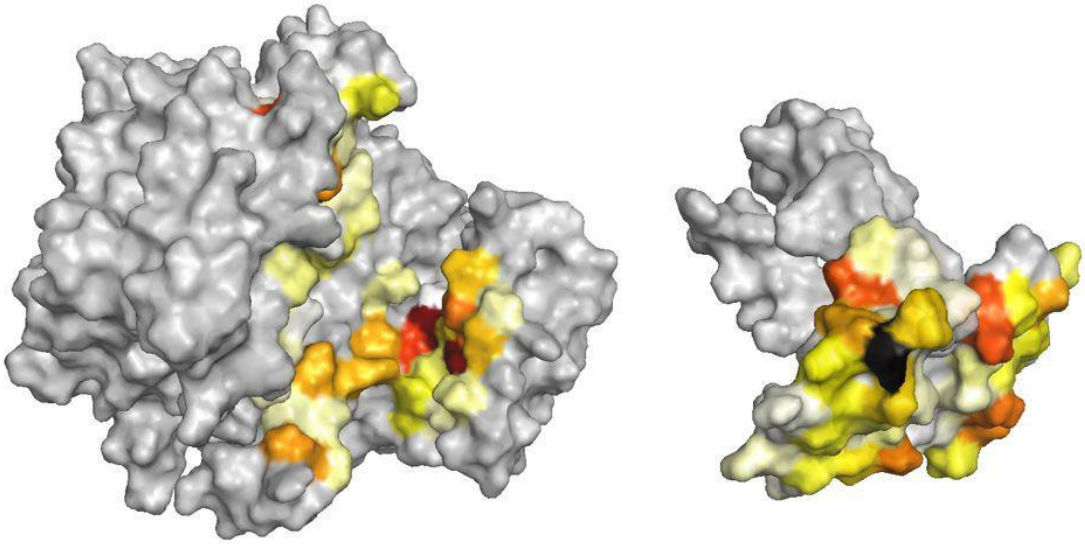
The structure of the native E. Coli Tar chemoreceptor which is the basis of iGEM team Technion 2016's project, As presented in Chimera UCSF

Chimera is available for free at:
https://www.cgl.ucsf.edu/chimera/
The site also offers extensive documentation on the software.

For a quick start guide, you can watch the following short playlist:
https://www.youtube.com/playlist?list=PLHib7JgKNUUeTZONxd0h0WBiZzAJmXmva

PyMol

PyMol is an open source molecular visualisation software that, as the name suggests, is integrated with Python, meaning that elaborate programs can be written to display results on your 3D structure with Python:



The Results of a computational alanine scan of iGEM TU Eindhoven 2016, showing the important residues on a T14-3-3 dimer and a dimerized CT52, darker colors indicate a stronger change in G values, and thus show more important residues.

Besides custom visualisation pyMol can perform various other tasks like creating a PDB from scratch, moviemaking, determining neighbouring residues, and much more. An extensive documentation on how to use PyMol is also available in the form of a wiki.

OpenBabel

OpenBabel is a chemical toolbox which has many applications in analysing and converting chemical data, one of these applications is to convert a PDB of a chemical ligand to a .mol2 file, which in turn Rosetta can turn into the files necessary for its simulations. You can get OpenBabel and find support for it in the official website.

BCL

BCL - the Biology and Chemistry Library Project is a C++ programming library designed to simulate biological molecules and chemicals. Certain Rosetta protocols demand use of BCL (The design of ligand binding sites presented later in this document for example). BCL is free for academic users and a licence for it can be obtained in the Meiler Lab website.

Virtualbox

If you have no access to a UNIX based system and instead have to use a windows PC, you need to take some extra steps to install and use Rosetta. Since Rosetta runs only on UNIX based systems, it is necessary to either dual boot your computer to have both Windows and Linux, or create a virtual PC which runs Linux. Such a virtual PC can be created using VirtualBox.

VirtualBox is available for free at: https://www.virtualbox.org/

In order to create your virtual PC you need to download the OS:
http://www.ubuntu.com/download

Documentation on how to create a virtual PC is available at:
https://www.virtualbox.org/manual/ch01.html#idm267

Another option for windows is the alternative software of PyRosetta. PyRosetta is an interface for Rosetta written in Python, which was made to make Rosetta modeling available for a broader public, since Python a widely used programming language. On the official PyRosetta website, www.pyrosetta.org, the software is free to download if you are from an academic institution. If you start with PyRosetta, it is recommended to download the graphic molecule visualization software PyMOL (in my institution it was already available).

These tutorials start off by introducing PyMOL, and guide the user step-by-step into using PyRosetta. If you know a bit of Python, the tutorials are easy. And even if you don't, getting to know a bit of Python is not the worst idea because it is a very simple and intuitive programming language (documentation of Python can be found onhttps://docs.python.org/3/) . However, we can imagine that the advantage of using PyRosetta over Rosetta is limited as long as you don't know any Python.

RECOMMENDATIONS

Most computers have trouble running two 64-bit operating systems at the same time, so it is recommended you download a 32-bit operating system.

Rosetta is a reasonably large software and you will need quite a bit of space to store all your data and generated PDBs, so we recommend creating a virtual drive with **at least** 60 GB of disk space.

Adding multiple processors is also recommended.

Structure Databases

Generally, every protocol in Rosetta require a protein structure or sequence file as a basis to work with. These files can be obtained freely from several online databases.

Protein Data Bank

Protein Data Bank (PDB) is the main database for three dimensional structures of biological molecules. The vast majority of data was obtained over the course of years using X-ray crystallography or NMR spectroscopy and submitted by scientists from around the world.

The data is accessible for free via one of three sites:

- PDBe
- PDBj
- RCSB PDB

A high resolution structure (better than 2 Å) obtained using X-ray crystallography is the

best input possible and can used with Rosetta with very few preparations. A structure with lower resolution, an NMR or a homology structure will make your modeling less effective and the results less accurate.
Be sure to search for the best possible inputs available before starting your work.

UniProt

Universal Protein Resource (UniProt) is the main resource for protein sequence and functional information. It contains data about thousands of proteins from hundreds of different organisms. All the data is accessible for free.

each protein page presents the amino acid sequence and a link to the relevant literature from which the information was taken. If a 3D structure of a protein exists, there will also be a link to the relevant entry on PDB.

UniProt

Since UniProt contains a massive amount of proteins, and is easier to navigate, it is recommended to start your search from it and find the relevant entries on PDB from the links on UniProt.

ZINC15

Zinc is a free database of commercially available chemical compounds (Ligands). It contains over 100 million compounds in ready to dock 3D formats - sdf/smiles/mol2

Aside from the chemical information and ligand files, each entry contains links to several suppliers from which the compound can be purchased.

ZINC15

I want to do X

When you are trying to design proteins with Rosetta you often know exactly what you want to achieve, but not how to achieve it. This chapter contains some of the common goals and goals we have personally strived for when using Rosetta and elaborates how to achieve these.

Add an unknown residue to my simulations

Sometimes your goal is to simulate the interaction between proteins and a non standard ligand/chemical compound, like sulfate or Fusicoccin, in this case you will need to create a
.params file for your ligand to tell Rosetta how to work with it.

1. *If your ligand is present in the PDB, extract it and generate a .mol2 file using OpenBabel*
2. *Make a .params file using Rosettas mol_to_params.py*
3. *Add to simulations using command -extra_res_fa <.params files>*

Preparing structures for use in Rosetta

input structures from PDB databases are rarely immediately suitable for use in Rosetta, the PDB has to be relaxed into the Rosetta scoring function to prevent steric clashes and produce more accurate results. This can be done with the Relax application. If your protein is flexible or contains flexible parts, it is recommended to also do a Backrub simulation.

1. *Cleaning*
2. *relax simulation*
3. *Backrub simulation*

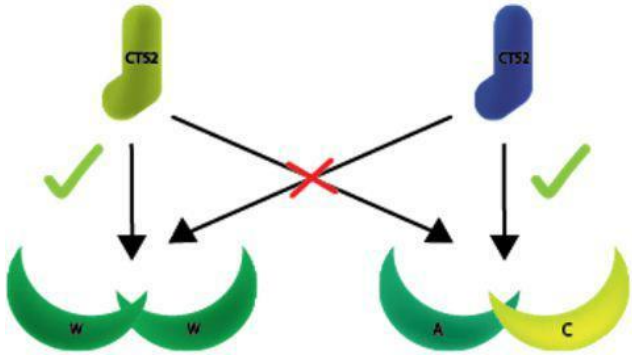
Finding important Residues for a protein binding interface

Not all residues in a protein are relevant for the binding interactions between 2 (or more) proteins, how relevant a residue can be seen with alanine scanning: With alanine scanning you mutate a residue to alanine, which is a relatively non-interactive amino acid, and determine the change of the free Gibbs energy (G), the larger the change*, the more relevant the residue. Alanine scanning can be done in the lab but also ***in silico*** (computational), which is far more cheap and fast, but does give less certainty about the results.

*Note that a positive change indicates that the binding interaction weakens, and a negative change the binding interaction strengthens.

1. *Prepare the structure for use in Rosetta*
2. *Computational alanine scan*

Designing an orthogonal binding interaction



Designing an orthogonal protein pair means introducing mutations in both interacting proteins and having no crosslinking between the wildtype pair and your designed pair (e.g. wildtype protein A only binds to wildtype protein B and designed protein A only to designed protein B, see figure). Designing an orthogonal pair is commonly used to ensure that your designed protein only interacts with the proteins you designed it to interact with and no others. iGEM team Eindhoven 2016 has designed these interactions to turn a homodimeric scaffold protein into a heterodimeric and a tetrameric scaffold protein. They have also written a more extensive protocol for the design of an orthogonal pair, including example scripts. This protocol is available here.

1. *prepare the structure for use in Rosetta*
2. *Find important residues*
3. *Mutate protein A (point mutant scanning)*
4. *Compensate by mutating protein B (point mutant scanning)*
5. *check orthogonality (point mutant scanning)*
6. *extensive remodeling*

Design of a binding pocket/interface for a chemical compound

Natural ligand binding proteins bind molecules that are necessary for the cell or are harmful to it in some way (i.e. to recognize danger), this implies that that the majority of chemical ligands are not recognized by any protein in nature.

Natural proteins can be redesigned computationally to bind different ligands. During the design process the expected binding pocket is mutated and the binding energy is re-evaluated to assess whether binding is more probable.

The redesign protocol was written by: Rocco Moretti, Brian J. Bender, Brittany Allison and Jens Meiler from Meiler lab, Vanderbilt University.

This Protocol is available here.

Please refer to the Protocols section of this guide for tips and a general outline of the protocol.

iGEM Technion 2016 have written and optimized scripts for every section of the protocol which are available here

Protocols

The protocols that are elaborated here are linked to the steps in the previous chapter. Note that these protocols are examples and recommendations, and might be incomplete for the simulation you want to run. Links to The official Rosetta documentation will be available for each protocol.

Cleaning your pdb

a pdb file has to be cleaned before use in Rosetta, because it has information Rosetta doesn't use and could even hinder your simulations.

Materials

- input pdb

Protocol

- if your pdb contains a wanted ligand, use: <path_to_Rosetta>/main/source/src/apps/public/relax_w_allatom_cst /clean_pdb _keep_ligand.py <input pdb> - ignorechain
- otherwise use: <path_to_Rosetta>/tools/protein_tools/clean_pdb.py <input pdb>

this outputs a pdb that can be used for further preparation, note that this renumbers the residues in your pdb.

Adding unknown residues

Often you want your protein to react to a chemical compound or another nonstandard residue which Rosetta is unfamiliar with, in this case you will need to add a file which contains information about your ligand to your simulations. Such a file can be found in structure databases like ZINC15,or in case your ligand is already present in the pdb, you can extract it from the pdb.

Materials

- OpenBabel software package
- pdb containing ligand
  OR
- ligand file (.smiles,.mol2,.sdf)

Protocol

- extract file from pdb:
  - make sure your ligand is in a separate chain from the rest of the pdb
  - use: grep '<residue> <chain>' <pdb> > <residue>.pdb
- making a Rosetta compatible file:

- generate .mol2 file:
  - babel <residue>.pdb <residue>.mol2
- generate .params files:
  - <path_to_Rosetta>/main/source/scripts/python/public/molfile_to _params.py -n <res> -p <res> <residue>.mol2
  - <res> has to be the same 3-letter code as the code the ligand is represented by in the input pdb.

To use the residue in your simulations, add the -extra_res_fa <res>.params flag to your flag file.

Relax

In order to use your input PDBs in Rosetta the first need to be 'relaxed' into the Rosetta score function to reduce minor steric clashes in the PDB. Not relaxing the PDB before using it for other simulations will almost certainly give inaccurate results.

Materials

- Input PDB
- (optional) .params files for unknown residues

Protocol

- create a flags file (a file that contains commands for the simulation)
  - Required flags:
    - -database <path to database>
    - -s <pdb> OR -l <list of pdbs>
    - - nstruct <n> #n should be at least 10
  - Recommended flags:
    - -relax:constrain_relax_to_start_coords
    - -relax:coord_constrain_sidechains
    - -relax:ramp_constrain false
    - -ex1
    - ex2
    - -use_inut_sc
    - -fli=_HNQ
    - no_optH false
    - mute basic core
  - for more options and flags see the official documentation.
- run the simulation
  - <path_to_Rosetta>/main/source/bin/relax.<OS>gccrelease @flags > <outputfile>

Post processing

The output of this simulation will be:

- a set of generated pdbs
- a score.sc file
- <outputfile> which contains the information normally printed to the terminal.

The most important files are the pdbs and the score file, the output file can be used to find errors in your simulation. The score file contains the scores of each generated pdb, use the best pdbs (>2 is recommended) to continue your work. Note that a **lower**score means a structure is more stable, however, this does not necessarily means that it will perform the function you want it to. Usually you determine your best pdbs on their total score, but in some case other scores are more relevant, see chapter Rosetta energy - how to filter results to see what all scores describe.

Backrub

In order to simulate backbone flexibility it is recommended to do a backrub simulation before continuing towards getting results. If you only have rigid/robust proteins to analyze you can skip this step. In the case that only some of your proteins are robust, you should run the simulation, but excluding the residues of the rigid proteins is recommended.

In a backrub simulation the backbones of residues are rotated respective to their neighbours in order to improve the score of the protein, and thus finding a more stable structure. It is also possible to repack (= change orientation) the side chains and introduce mutations using a resfile.

It is also possible to submit your pdb to the RosettaBackrubserver, however this limits your control over the simulation significantly, so running it on your own machine/cluster is recommended.

Materials

- Relaxed pdbs
- (optional) .params files
- (optional) resfile

Protocol

- create a flags file
  - required flags:
    - -database <path to database>
    - -s <pdb> or -l <pdblist>
    - -nstruct <n> # n > 10 recommended
    - -backrub:ntrials <n> #n = 10000 recommended
    - -in:file:fullatom
  - recommended flags:
    - -ex1
    - -ex2
    - -ex3
    - -ex4
    - -extrachi_cutoff 0
      (Optional) -resfile
    - <resfile> #for repacking and mutating

residues, see the official documentationon how to construct resfiles
(optional) -pivot_residues <resnr1> <resnr2> #determine etc.
which residues to backrub (default is all residues)
mute basic core
- for more options and flags see the official documentation.

Post-processing

the post-processing is similar to the post-processing of the relax simulation, however it outputs 2 pdbs per input pdb, namely the last generated and the pdb that scored lowest.

Computational alanine scanning

In order to determine the important residues in a binding interaction an alanine scan can be performed, this can be done on a local version of Rosetta, but the Robettaserver provides an excellent service for this application, therefore we recommend simply submitting your PDB to this server. We strongly recommend submitting multiple pdbs of your structure, as results may differ slightly from pdb to pdb

Materials

- pdbs
- (optional) mutations list

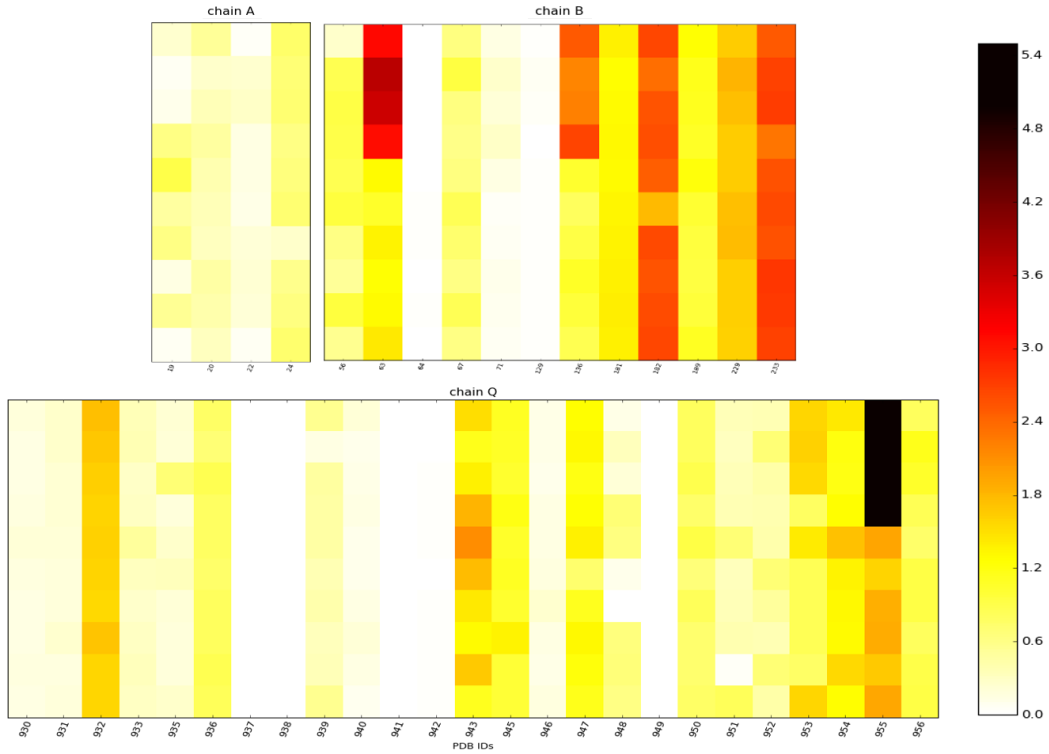
Protocol

- if you have knowledge/suspicion on which residues are important for the binding of your proteins, it is recommended to construct a mutations list file, as this tells the server which residues to process, this will make analyzing results easier and more consistent. If you do not submit a mutations list, the server will automatically determine which residues to process, and this may differ per pdb.
  - documentation on constructing a mutations list can be found here
- identify binding partners by chain id
- fill in the submit form

Post-processing

The server outputs a bunch of files, of which the .results file is the most important. The results file contains a line for each mutated residue. the columns you should look at are pdb#, which tells you the residue number in the submitted pdb, the chain id and the DDG(complex), these describe the change in Gibbs free energy when the residue is mutated to alanine.

We recommend that you visualize your results to make them easier to understand and generally more presentable. This can be done in several ways, for example in a heatmap (see below) or on the protein complex itself (see chapter 'Necessary software' - PyMol).



Computational alanine scanning results of iGEM TU Eindhoven 2016 represented on a heatmap.

Point mutant scanning

The point mutant scan application, as the name suggests, scans for point mutants for given residues and determines the change of Gibbs free energy the mutation causes. This scanning is a great way to design orthogonal binding interactions:

you start by introducing mutations in one of the proteins (say protein A) to destabilize the binding interaction. The highest positive changes mean the largest increase in the G values and the weaker the binding affinity between the mutated protein A and wildtype protein B, which is essential for the orthogonality of your designed pair.

Next you want to find mutations in protein B that strengthen the binding interaction with the **mutated**protein A, this is to ensure your orthogonal pair is still functional. in this case the G values should decrease, preferably even lower than the G values of your wildtype-wildtype binding interaction

Lastly, you need to check whether your mutated protein B does not/weakly interact with your **wildtype**protein A, you can do this in the same way as in the first step, but you mutate protein B instead of A.

Materials

- pdbs
- (optional but **strongly**recommended) mutations list
    - We recommend doing a computational alanine scan to determine which residues to mutate, and you should mutate to all amino acids except cysteine, because it forms disulfide bonds, and proline, because the point mutant scan application can not handle these correctly and could strongly disturb the structure of your protein.
    - a mutations list can contain single and double point mutations and is constructed as following.
        - <chain> <old AA> <residue number> <new AA>
        - for example:
            - A E 19 R
            - A E 19 R Q K 943 D

Protocol

- create a flags file
    - essential flags:
        - -database <path to database>
        - -s <pdb> or -l <pdblist>
        - -output_mutant_structures #Outputs pdbs that have a change in G values that passes a threshold, these output pdbs are necessary for the second step
        - -alter_spec_disruption_mode #This makes the application search

for destabilizing mutations instead of stabilizing ones, note that

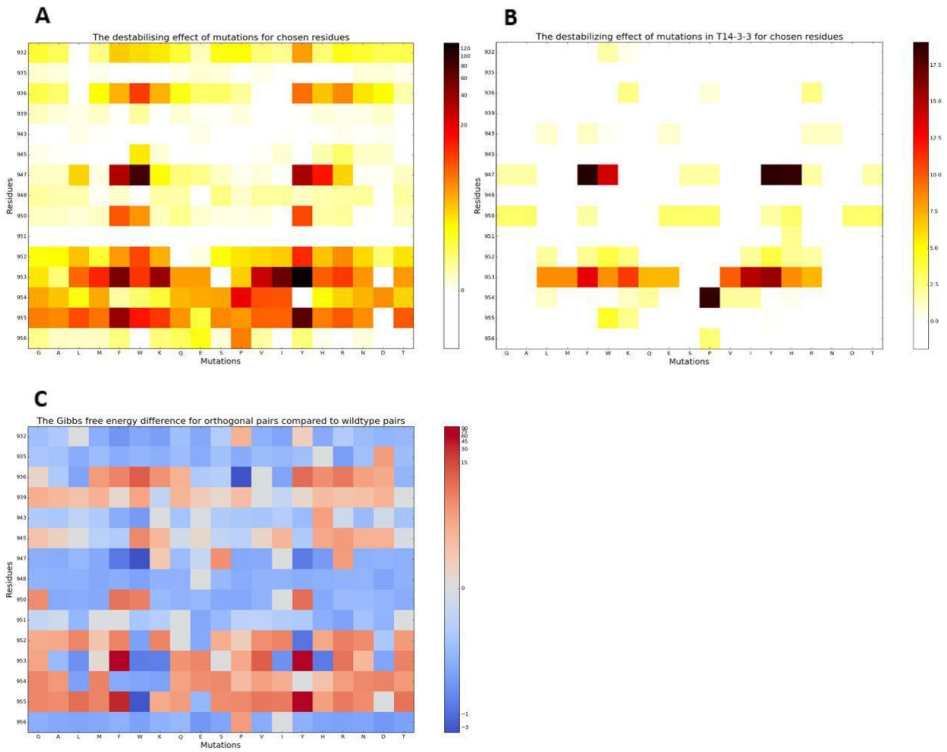
this is **not** reflected in the log file, so if this is set to true, changes
in the energy that are negative should be positive and vice versa.

- recommended flags:
  - -ex1
  - -ex2
  - -use_input_sc
  - -flip_HNQ
  - -no_opth false
  - -ignore_unrecognized_res
  - -no_his_his_pairE
  - -ddG_cutoff #determines the threshold for mutated structures to be approved (change should be below threshold, default = -1)
  - -double_mutant_scan #allows use of double point mutations

- run /path/to/Rosetta/main/source/bin/pmut_scan_parallell.<OS>gc crelease @flags > <logfile>

Post-processing

The output of the point mutant scan is a log file that contains, among other things, the tested mutation and the resulting average change in Gibbs free energy. As with computational alanine scanning we recommend visualizing your results. for example:



Point mutant scan results of iGEM TU Eindhoven 2016. A) destabilizing mutations in the CT52 residue numbers are shown along the y-axis, mutations along the x-axis. B) destabilizing mutations in the T14-3-3 protein, shown residue numbers and mutations are not on T14-3-3, but on the CT52 that the mutations in the T14-3-3 compensate for. C) The energy difference between the wildtype pair and mutated pair.

For the design of your orthogonal pair mutations should be chosen that are very destabilizing in both A and B (dark red), but preferably blue or light red in C in the figure above.

Extensive remodeling

After designing an orthogonal pair, or another reason, you might want to remodel your pdb to more accurately predict the likely structure of your mutated pair, Gregory T. Kapp et al. have written a protocol on how to to such an extensive remodeling which is available in the SI of their article.

Design of Ligand Binding Sites

*Based on the article"Rosetta and the Design of Ligand Binding Sites" by Rocco Moretti, Brian J. Bender, Brittany Allison and Jens Meiler. Available here

The protocol which is presented in the article is thoroughly explained so this section will not rewrite it, instead we will present a general outline of the procedure with tips and clarifications about the different steps from Technion iGEM's experience.

Materials

- input PDB of a protein ligand binding site
- input mol2 file of a ligand of your choice
- OpenBabel
- BCL

Protocol

(*The number in brackets is the relevant step in the article)

- (3.1) relax the input PDB into Rosetta using the written command line
- (3.2) convert the small molecule file into SDF format using OpenBabel
    - in our experience, mol2 files perform best in this step but every ligand file type (sdf,smiles) can be used.
    - A licence to use BCL is usually given for a week at first (and can be extended when needed). Make sure your licence is valid before running step 3.2.2
- (3.3) Identify the interaction pocket
    - This step is critical for a successful design and has to be done manually, please refer to our designated guide for this step which can be found here
- (3.4.1) Prepare a residue specification file
    - A resfile specifies which amino acids can be mutated for the design and which should not be touched. The example file given in the article works

      Very well however if you want to customize the file, you can refer to the documentationto learn about the various options.
- (3.4.2) Prepare a docking and design script
    - The docking and design script is the heart of the protocol, performing the actual mutations on the protein. The script in the article works very well, please refer to the article or to the

scripts provided by Technion iGEM 2016 if any problems arise (most likely due to typos)

- (3.4.3) Prepare a design options file according to the article.
- (3.4.4) Run Rosetta using the command provided in the article. This is the actual design step and should take some time to finish depending on your computer and on your input files.
- (3.5) Filter designs according to the article.
    - When preparing the metric_threshold.txt and metric_thresholds2.txt files which specify how to filter the results please refer to the next section of this guide - Rosetta Energy
- (3.7) run the design protocol again on your filtered outputs several times more
    - 3-5 runs of the protocol (which each running on the previous one's results) is a good start.

Tips and tricks

- You can sort your score files with the sort command. For example: sort <score file> or sort -k5 score file if you want to sort by the fifth column.
- if you have unrecognized residues in your pdb but you do not want them/ do not care about them, you can pass the ***-ignore_unrecognized_res*** flag
- if residues go missing during your simulations, try passing the ***-ignore_zero_occupancy false*** flag.
- if your pdb contains water you should either remove them, or pass the ***-ignore_waters false*** flag.

For more information on our take on the modeling process and how we used Rosetta visit our WIKI pages:

iGEM TU Eindhoven: http://2016.igem.org/Team:TU-Eindhoven

iGEM Technion Israel: http://2016.igem.org/Team:Technion_Israel

Rosetta Energy - how to filter results

At the end of your design or modelling run, you will likely have an output library of results. This library can range from dozens to thousands of proteins depending on your parameters. This makes the subject of filtering results an extremely crucial part of any design process.

Filtering in Rosetta is done by scoring every aspect of the protein complex, this includes bonding energy, interactions between amino acids of the same protein, backbone angles, clashes with the ligand (if exists) and many more. The scores are in arbitrary units of REU - Rosetta Energy Unit. After scoring the parameters of the protein, you can select the proteins with the best possible scores according to your desired parameter.

In many cases Rosetta can tell which proteins are more likely to fold properly (this does not mean they will function the way you want them to). It is your responsibility to give it the correct parameters for the filtering process. Correct filtering can reduce your outputs from hundreds to dozens or less.

Scoring Proteins

The Rosetta energy function is a combination of physics-based and statistics-based potentials. **The actual process of scoring a protein file is well documented and demonstrated in the official documentation found here.**

After scoring proteins you should have an output file similar to this:

SEQUENCE:																
SCORE:	total_score	ClassicGrid_grid_X	Transform_accept_ratio	angle_constraint	atom_pair_constraint	chainbreak	coordinate_constraint	dihedral_constraint	dslf_ca_dih	dslf_cs_ang	dslf_ss_dih	dslf_ss_dst	fa_atr	fa_dun	fa
SCORE:	-608.410	-33.000	0.732	0.000	0.000	0.000	1.572	0.000	0.000	0.000	0.000	0.000	-620.537	117.943	-14
SCORE:	-608.306	-35.000	0.540	0.000	0.000	0.000	1.377	0.000	0.000	0.000	0.000	0.000	-622.069	116.632	-14
SCORE:	-602.350	-34.000	0.686	0.000	0.000	0.000	1.516	0.000	0.000	0.000	0.000	0.000	-619.075	117.381	-13
SCORE:	-606.983	-33.000	0.514	0.000	0.000	0.000	1.175	0.000	0.000	0.000	0.000	0.000	-625.897	119.239	-14
SCORE:	-604.049	-33.000	0.524	0.000	0.000	0.000	1.323	0.000	0.000	0.000	0.000	0.000	-622.391	121.506	-14
SCORE:	-601.878	-35.000	0.574	0.000	0.000	0.000	0.907	0.000	0.000	0.000	0.000	0.000	-621.786	119.268	-14
SCORE:	-600.496	-34.000	0.592	0.000	0.000	0.000	1.634	0.000	0.000	0.000	0.000	0.000	-626.298	118.286	-13
SCORE:	-606.789	-35.000	0.628	0.000	0.000	0.000	1.167	0.000	0.000	0.000	0.000	0.000	-624.395	119.459	-13
SCORE:	-603.660	-35.000	0.740	0.000	0.000	0.000	1.126	0.000	0.000	0.000	0.000	0.000	-626.360	122.852	-14
SCORE:	-602.431	-34.000	0.678	0.000	0.000	0.000	1.419	0.000	0.000	0.000	0.000	0.000	-625.065	122.544	-14
SCORE:	-608.999	-34.000	0.714	0.000	0.000	0.000	1.237	0.000	0.000	0.000	0.000	0.000	-623.386	119.796	-14
SCORE:	-604.104	-34.000	0.650	0.000	0.000	0.000	2.066	0.002	0.000	0.000	0.000	0.000	-623.449	115.192	-13
SCORE:	-607.125	-34.000	0.662	0.000	0.000	0.000	1.228	0.000	0.000	0.000	0.000	0.000	-626.789	118.982	-14
SCORE:	-607.037	-33.000	0.650	0.000	0.000	0.000	1.113	0.000	0.000	0.000	0.000	0.000	-622.490	119.753	-14
SCORE:	-608.872	-34.000	0.678	0.000	0.000	0.000	1.990	0.000	0.000	0.000	0.000	0.000	-623.721	117.308	-14
SCORE:	-602.386	-35.000	0.730	0.000	0.000	0.000	0.850	0.000	0.000	0.000	0.000	0.000	-623.962	121.046	-14
SCORE:	-608.935	-35.000	0.668	0.000	0.000	0.000	1.298	0.000	0.000	0.000	0.000	0.000	-625.706	119.262	-14
SCORE:	-605.131	-35.000	0.752	0.000	0.000	0.000	1.344	0.000	0.000	0.000	0.000	0.000	-620.732	119.741	-14
SCORE:	-607.359	-35.000	0.712	0.000	0.000	0.000	1.130	0.000	0.000	0.000	0.000	0.000	-620.448	118.284	-14
SCORE:	-594.637	-33.000	0.604	0.000	0.000	0.000	2.932	0.000	0.000	0.000	0.000	0.000	-621.882	118.526	-14
SCORE:	-608.268	-35.000	0.746	0.000	0.000	0.000	1.282	0.000	0.000	0.000	0.000	0.000	-626.299	119.343	-14
SCORE:	-607.192	-35.000	0.676	0.000	0.000	0.000	1.395	0.000	0.000	0.000	0.000	0.000	-618.064	114.234	-13

The different parameters which are being scored are lacking in explanation, so we will focus on them here. The following table is not complete but it presents the majority of parameters available. Please read through it carefully before deciding how to filter your results.

After scoring a protein file(s) you should have a file with the filetype: .sc. The file should look similar to the above photo. You can open this file in Microsoft Excel and perform calculations on the scores. Use the table below to choose the parameters upon which you want to filter the results. For each

parameter select a threshold - a value such that any protein with a score higher (or lower, depend on your configuration) than it is discarded.

The table lists some parameters with known universal values. If no good universal values are known a good rule of thumb is to filter by sorting the scores of each parameter from lowest to highest and selecting the top 25% as a threshold for filtering (i.e dropping the highest 25% as lower results are always better).

You can change the parameters and thresholds as you like depending on how many results pass the filtering process and how many you want.

How to use this table: The table has three columns, The name of the parameter, its biological or statistical meaning and universal good values to filter by if it is known.

NAME OF PARAMETER	WHAT IT MEANS	GOOD VALUES (IF KNOWN)
TOTAL_SCORE	Weighted average of all results	(-2)*(number of residues in protein) **Should not be positive.**
CLASSICGRID_GRID_X	Useful in docking protocols. The energy of binding from the low resolution stage of the docking. (Use to see how well the low resolution stage ran)	N/A
TRANSFORM_ACCEPT_RATIO	How well the low resolution Monte Carlo stage worked. It is a number between 0 and 1 and is provided as a diagnostic.	Generally around 0.3 is ideal, but anything that is not 0 or 1 is ok. If 0 or 1 you need to change how you're doing the docking.
COORDINATE_CONSTRAINT	How well the design fits the coordinate constraints (atom A must be at coordinate x,y,z. This is a parameter you choose during the design)	N/A
FA_ATR	The attractive portion of the Lenard-Jones potential.Can be used to tell how good the protein is packed	N/A
FA_REP	The repulsive portion of the Lenard-Jones potential.If high then there are clashes in the protein. **Filtering designs which are high compared to the average is recommended.**	N/A
FA_DUN	"Dunbrack energy" - a statistical	80-150

	potential based on sidechain conformation preference assembled by Roland Dunbrack's lab.It is protein only term.	
FA_ELEC	Coulombic electrostatic potential.	N/A
FA_INTRA_REP	The repulsive energy within residues.	N/A
FA_PAIR	Statistical residue-residue interaction potential. Very useful	Negative results are ideal

	in protein-protein interactions	
FA_SOL	Solvationterm. Useful in protein-ligand interactions with particularly hydrophobic or hydrophilic ligands.	N/A
HBOND_BB_SC	Hydrogen bonding term between protein backbones and sidechains.	N/A
HBOND_SC	Hydrogen bonding between different sidechains.	N/A
IF_X_FA_ATR	The terms without if_X_ are for protein as a whole. The ones with if_X are for the interface of the protein with sidechain X (usually a ligand).	N/A
IF_X_FA_REP		
IF_X_FA_ELEC		
IF_X_FA_PARI	(Normally interface is defined to be 5 Å from the ligand)	
IF_X_FA_SOL		
INTERFACE_DELTA_X	The energy of interactions between the ligand (chain X) and the protein	N/A
LIGAND_IS_TOUCHING_X	1 if ligand is close to the protein, 0 otherwise.	Should be 1 in most cases.
OMEGA	How bad are the protein backbone	N/A

	omega angles. Useful if doing loop remodeling.	
RAMA	Ramachandran energy. Each amino acid has it's own preference for where it likes to sit in the Ramachandran plot. Useful during backbone remodeling or loop remodeling.	N/A
P_AA_PP	Probability of amino acid given phi and psi. rama looked at from a different angle. Useful during loop remodeling.	N/A
PRO_CLOSE	How bad the geometry of the proline rings are. Rosetta allows the proline sidechain ring to open up during modeling and uses pro_close to keep them closed. Useful only for extensive loop remodeling.	N/A
RES_TYPE_CONSTRAINT	How close is the results to the native sequence. Useful if you want to filter on this basis.	N/A

COMPLEX_NORMALIZED	Total score of the protein complex normalized by the number of residues.	N/A
DG_CROSS	The energy of interaction between the two sides of the complex, calculated in the holo state (protein bound to ligand)	N/A
DG_SEPARATED	The energy of interaction between the two sides of the complex, calculated by taking the score of the holo state, then separating the two sides of the interface, optionally repacking, and then calculating the score in the separated state. This is particularly useful if you think you have an "induced fit" type situation, and want to correct for repacking in the absence of the ligand.	N/A
DG_SEPARATED/DSASAX100	The energy density of the interface. This is a more direct measure if you have a small tight interface. Useful for protein-protein designs	N/A
DG_CROSS/DSASAX100		N/A
DSASA_INT	A rough gauge of how much the two sides of the interface are touching.	N/A
DELTA_UNSATHBONDS	How many unsatisfied hydrogen bonds are introduced by the design Unsatisfied hydrogen bonds are ones where the atom is able to make a hydrogen bond but doesn't because it's blocked by other residues.	N/A
NRES_ALL	The total number of residues in	Number of residues in the PDB

	the complex	file
NRES_INT	The total number of residues in	Number of residues in the
	the interface	interface
PACKSTAT	How well the protein is packed.	values between 0-1. 1 being
	values between 0-1 with 1 being	better.
	better	

Important links, support and more data

For more information and guides on Rosetta and its functions please visit the official documentation.

For troubleshooting you can check the fixing errorspage or the forums.

The forumsare a great source of knowledge about various aspects of Rosetta and are quite active.

Another great source of information for beginners is Vanderbilt University's Meiler Lab website.Under the tab "Rosetta Tutorials" you can find materials from several workshops about Rosetta, these include protocols, demos and scripts. You can also check out their youtube page

Thanks and Acknowledgements

iGEM Technion 2016

We would like to thank Rocco Moretti, Brian J. Bender, Brittany Allison and Jens Meiler of Vanderbilt University's Meiler Lab for the protocol: "Rosetta and the Design of Ligand Binding Sites".

A special thanks to Dr. Rocco Moretti for his extensive help, detailed and thorough answers and incredible patience.

Mr. David Cohen from the Technion Physics department for his help with the ATLAS computer cluster.

Dr. Fabian Glaser from the Technion Bioinformatics Knowledge Unit for his help with protein-ligand docking and explanations about Chimera UCSF.

Dr. Roee Amit, Mrs. Michal Brunwasser, Mrs. Noa Kats, Mrs. Beate Kaufmann, and Mrs. Alex Ereskovsky from the Technion faculty of Biotechnology for their guidance and patience.

iGEM TU Eindhoven 2016

We would like to thank Dr Ir. Tom de Greef for providing literature on which we could base our protein design. That said we would like to thank Gregory T. Kapp, Sen Liu, Amelie Stein, Derek T. Wong, Attila Reményi, Brian J. Yeh, James S. Fraser, Jack Taunton, Wendell A. Lim and Tanja Kortemme for their article about the computational design of an orthogonal pair and providing the protocolfor extensive remodeling to get more accurate results.

A special thanks Dr. Ir Bart Markvoort for advice on how to use the Tu/e biosim cluster, for giving feedback on our written protocol and giving tips on what further steps to take to get more accurate results.

Another big thank you to Job Roodhuizen for giving us a crash course on Linux and providing self written programs for analyzing the computational alanine scan data.

A thank you to iGEM team Wageningen for providing some information about PyRosetta.