Defining monoterpene cyclase theoretical chemical space and binding capacity using high-throughput *in silico* molecular docking

Aretas Gaspariūnas
Faculty of Biology, Medicine and Health, School of Biological Sciences, University of Manchester.

Supplementary Material

Contents

1 Introduction

2 Preparation of ligand libraries

3 Preparation of enzyme libraries

4 Molecular Docking

5 Analysis of result

1 Introduction

Here as presented and described full methods and novel high throughput molecular docking and virtual screening pipeline used in this study. In this study, the HTS_MD folder is considered as a home directory for all Python scripts, software tools, ligand libraries, enzyme libraries and MD results unless otherwise noted. Python scripts include command examples in their respective sections. The project repository can be found on GitHub: https://github.com/aretas2/High-throughput-molecular-docking.git

2 Preparation of ligand libraries

To perform molecular docking with AutoDock Vina (Trott and Olson, 2009) both, enzyme and ligand, files must be in the supported format – PDBQT. Here are present and described methods used to generate and prepare ligand libraries.

2.1 Control library

Control library consists of 39 unbranched linear alkanes of carbon lengths 2-40 (C_n; n = [2-40]). Initially, control library was generated progressively elongating the alkane chain by adding a symbol for carbon ("C") to a text file with SMILES for the compounds. This was achieved with a written script crt_smiles_list.py found in control library folder (control_scaffolds). The script accepts only one input – the maximum length of the alkane chain to be generated in SMILES and outputs in present working directory (pwd) a text file "list_control_smiles.txt" with a list of all the SMILES as well as individual files for each linear alkane SMILES. It can be run by:

```
control_scaffolds User$: python crt_smiles_list.py 40
```

The SMILES are then converted to PDB format using the Open Babel command (O'Boyle et al., 2011):

```
control_scaffolds User$: obabel -i SMILES *.smiles -o pdb -m --gen3d
```

The ligand PDB files are then manually moved to a new folder "Scaffolds_pdbs" for a subsequent energy minimisation (EMM) with AmberTools16 (Salomon-Ferrer et al., 2012). To automate energy minimisation and skip many commands that involve intermediate files a Python script EMM.py was written. First, for the script to work $AMBERHOME path must be set by specifying the directory of AmberTools16:

```
HTS_MD User$: export AMBERHOME=/User/RP1/amber16
```

After this step, the EMM.py can be utilised by providing ligand PDB file as input. The script outputs a folder with the EMM related files in the location of the input file. Default settings

are used for EMM which can be modified by the user with the use of appropriate flags (see EMM.py). Use this command to run EMM:

```
Scaffolds_pdbs User$: python ../../EMM.py –pdb scaffold_2.pdb
```

The command to run EMM on PDB files in bulk:

```
Scaffolds_pdbs User$: for f in *.pdb; do python ../../EMM.py –pdb $f; done
```

For the final step, the ligand files are converted to PDBQT. The scripts prepare_ligand4.py and prepare_receptor4.py used for the conversion are distributed with MGLTools software package (Morris et al., 2009). A command to run the script:

```
control_scaffolds User$ ../bin/pythonsh ../prepare_ligand4.py –l
Scaffolds_pdbs/scaffold_2/*.pdb –A hydrogens –U nphs
```

A command to run the script on PDB files in bulk:

```
control_scaffolds User$ for f in Scaffolds_pdbs/scaffold_*/*.pdb; do
../bin/pythonsh ../prepare_ligand4.py –l $f –A hydrogens –U nphs; done
```

PDBQT files are then manually moved to a separate folder in the control_scaffolds folder named “PDBQT_EMM”

2.2 Linear library

The linear library consists of isomers and enantiomers of decane. The total number of decane isomers is 75 and coupled it with stereoisomers results as a 143 linear ligand library. 75 IUPAC names of isomers of decane were obtained (www.kentchemistry.com/links/organic/isomersofalkanes.htm) and converted to SMILES using Opsin (Lowe et al., 2011). The chiral centres were determined with Open Babel (obchiral); a list of enantiomer SMILES was created using chirality.py.
linear_scaffold_list.txt was obtained by using knkt2clmn.py script to connect IUPAC names text file with their respective canonical SMILES (Opsin) into one file to be used with chirality.py.
The script accepts obchiral output (chi_info2.txt) and a file with: a ligand number in the library; IUPAC name; canonical SMILES (linear_scaffold_list.txt) as inputs and outputs linear_scaffold_list_ch.txt containing ligand number; IUPAC name and isomeric SMILES for each enantiomer.

```
Linear_scaffolds User$ python chirality.py linear_scaffold_list.txt
chi_info2.txt
```

Generation of the final list of linear library (linear_enantiomer_list.txt) and 2D sketches of every ligand in the library:

```
Linear_scaffolds User$ python smiles22D.py linear_scaffold_list.txt
```

Another script crt_smiles.py was utilised to generate individual SMILES file for each enantiomer in linear_enantiomer_list.txt file. The subsequent steps for further linear ligand preparation (conversion to PDB; EMM; PDBQT) were identical to the ones described in Section 2.1.

2.3 Product library

The product library located in the Natural_products folder consists of 28 various natural products of mTC/S, GPP and geranyl. SMILES structures were obtained from PubChem. For the details of the library preparation please see Section 2.1.

2.4 Cyclic library

The cyclic library located in the Cyclic_skeletons consists of 74 theoretically possible monoterpenoid skeletons sorted into 5 types: 1-ring + 1 double bond; 2-rings (spiro); 2-rings (bridge); 2-rings (fused); 2-rings (separated) (Tian et al., 2016). For details of the library preparation see Section 2.1.

3 Preparation of enzyme libraries

Receptor/enzyme libraries were prepared from 14 publicly available mTC/S crystal structures and 2 in-house provided structures. Two enzyme libraries were created: enzymes with no heterogeneous atoms (HETATM) and enzymes with only the pyrophosphate in the active site, NS (no substrate) and PHO libraries, respectively. NS and PHO libraries were prepared using protein_prep.py and prot_prep_phos.py (located in their respective folders). Finally, after the NS and PHO libraries are created a conversion to PDBQT format is done:

```
PHO User$ ../bin/pythonsh ../prepare_receptor4.py –r foo_NS_pho.pdb –A
hydrogens –U nphs
```

A command to run the script on protein PDB files in bulk:

```
PHO User$ for f in *.pdb; do ../bin/pythonsh ../prepare_receptor4.py –r $f
–A hydrogens –U nphs; done
```

4. Molecular Docking

Before MD can be done with the enzyme and ligand libraries a new folder for the MD results is created. The Python script vina_config.py automates the creation of Vina configuration files (Vina input). The script accepts as input: protein PDBQT file; central coordinates of the search window; the size of the search window; the folder to save configuration files to; the ligand library folder with PDBQT files; enzyme chain (A/B). The output consists of a folder named after a protein name and a ligand combination with a respective Vina configuration file. The protein PDB input must be in pwd:

```
PHO User$ python ../vina_config.py –f foo_NS_pho.pdbqt –xc 1 –yc 2 –zc 3 –
lo ../MD_foo_results –lo ../Linear_scaffolds/PDBQT_EMM
```

A command to run the script on protein PDBQT files in bulk if the coordinates for the search window are the same:

```
PHO User$ for f in 1n*.pdbqt; do python ../vina_config.py –f $f –xc 1 –yc 2
–zc 3 –lo ../MD_foo_results –lo ../Linear_scaffolds/PDBQT_EMM; done
```

A table 1 shows the central coordinate of the search window for each enzyme chain. The size of the search window was the same for all the enzyme chains: 20 x 20 x 16. Exhaustiveness, num_modes and seed parameters for Vina configuration file were the same for all molecular docking simulations: 15, 20, 314159, respectively.

Table 1. Search window centre coordinates used for all enzymes in the study.

PDB_ID	State	Chain	X central	Y central	Z central
2ON*	C	A	22	55	-50
2ON*	C	B	-22	57	-71
1N*	C	A	45	50	50
1N*	C	B	33	41	0
1N21	C	A	70	82	55
2J5C	C	A	4	-48	52
2J5C	C	B	24	-10	39
bCinS	C	A	-48	43	-7
bCinS	C	B	0	45	-9
5UV*	O	A	-10	12	-21
5C05	O	A	-2	9	64
5C05	O	B	26	32.6	40
bLinS	O	A	35	115	-28
bLinS	O	B	45	145	0

4.1 Autodock Vina

After the Vina configuration files are created a Python script autodock1.py is used to launch MD simulations. The script has two inputs – the configuration file and the location of Vina. Outputs a PDBQT file with the position and shape of the top 20 MD results in the same folder as the input configuration file:

```
HTS_MD User$ for f in MD_foo_results/*_final1_*/config*; do python
autodock1.py –i $f –vina ./vina; done
```

5 Analysis of results

After high throughput MD is done a series of scripts were used to perform high throughput virtual screening by extracting top docking scores for each enzyme-ligand docking combination and performing normalisation. The visual presentation of the MD results as histograms and normalisation were done using bst_norm_scr.py script. The script accepts multiple inputs: protein pdbqt (-f); protein chain (-ch; default = A); and name of the MD results folder (-lo). To normalise the MD data two additional flags must be used: choose the SMILES library (-lo_norm); folder with MD results with control library (-lo_norm2). The script outputs up to 3 folders with histograms in PNG and TXT files. The protein PDBQT file must be in the pwd.

```
PHO User$ for f in *.pdbqt; do python ../bst_norm_scr.py –f $f –ch B – lo
../MD_C_scaff_pho –lo_norm ../Cyclic_skeletons –lo_norm2 control_scaffolds;
done
```

The MD results analysis was continued by transferring all the data to Excel sheets with a series of scripts which utilise XlsxWriter package found in the excel-py folder.