PANEL implementation

Authors: Nandhini Rajagopal and Shikha Nangia, Syracuse University

Prerequisite:

1. In addition to standard python packages such as numpy, scipy, pandas, matplotlib, please make sure to install MDAnalysis package.
2. The scripts are written to suit python2.7 and Gromacs versions above 5.
3. The CG Martini forcefield is used for all simulations.

Prep:

1. Please extract the files from PANEL.tar.gz
2. Add the PANEL folder to your path.
3. Obtain CG protein structure using *martinize.py*

Computing d_s

P_sep.py

For homomeric PANEL,

Inputs: protein gro/pdb file in CG

Input path to file when prompted
Example `'/home/user/PANEL/prot1.pdb'`

Run:
./P_sep.py
Output: D^{90} Å
(90th percentile. The percentile value can be edited in the code,
line 23: `np.percentile(D, 90)`)

Obtain d_s value:
$d_s = 2D^{90} + 2$ Å

For heteromeric PANEL,

Inputs: proteinA and protein_B gro file in CG, number of beads in each protein

Run dsep.py separately with each of the protein gro/pdb file path edited.

$d_s = D_A^{90} + D_B^{90} + 2$ Å

Generating uniformly distributed geometries

P_setup.py

(Prep:

Make sure to center the protein using editconf before generating initial geometries

Before running *P_setup.py*, it is recommended to test the *rand_geom.py* code, which is called from within the *P_setup.py* for generating fewer geometries to make sure everything is working fine. This is because *P_setup.py* may take a couple hours to finish and it is better to run it after checking the geometries generated are placed well in the membrane.

Before using *rand_geom.py*, please change the lines 590, 591 and 592 to point to the path of your martini forcefield libraries. These paths will be writing on the topology files.

```
echo '#include "/home/user/local/lib/martini_v2.2.itp"'
echo '#include "/home/user/local/lib/martini_v2.0_lipids.itp"'
echo '#include "/home/user/local/lib/martini_v2.0_ions.itp"'
```

If you have the include statements already in your prot1_CG.top (and prot2_CG.top in case of hetero), then you remove these lines and proceed.

You may use the sample command line below to perform test run.

For homomers:

rand_geom.sh -cg test=prot1.pdb,prot1.top -c 2 -c 2! -dsep *ds_value* -n 5 --insane{-pbc=rectangular,-l=DOPC:2,-l=DPPC:2,-l=CHOL:1,-sol=W:9,-sol=WF:1,-salt=0.15,-x=10,-y=10,-z=9,-center}

For heteromers:

rand_geom.sh -cg testA=prot1.pdb,prot1.top -cg testB=prot2.pdb,prot2.top -c 2! -dsep *ds_value* -n 5 --insane{-pbc=rectangular,-l=DOPC:2,-l=DPPC:2,-l=CHOL:1,-sol=W:9,-sol=WF:1,-salt=0.15,-x=10,-y=10,-z=9,-center}

Sometimes the proteins may not sit correctly in the membrane, in which case you might have to reorient the position of the proteins using *gmx editconf* and try again.)

Inputs:

- Input parameters that need to be mandatorily provided will be prompted while running the code, such as “path (`path`)”, “Number of CG beads in each of the proteins (`anum1, anum2`)”, “Reference group residues for defining zero angle in rotational space (`ref1, ref2`)” . In our work with claudin-5, the zero angle residues were chosen from the TM1. (please refer to the paper-SI for rotational angle schematic), “Initial separation distance (`ds`)”, “pdb and top file names of the proteins (`P1_pdb, P1_top, P2_pdb, P2_top`)”, “Specify dimer type (`dimer_type`)”.
- Other parameters such as Ω bin size (`gs`), number of geometries per each Ω bin (`num_per_grid`), number of random samples added in each iteration (`add_samp`), and tolerance to stop adding more sample (`tol`), have already been assigned the

recommended values. We have tested these values and provided the best choice and would suggest to use the same values. However, it is quite straightforward to adjust these values in the beginning of the code.

```
t0 = time.time()
gs = 10 # recommended size of each omega bin = 10x10 degrees
num_per_grid = 2 # recommended number of geometries per omega bin = 2

grids = np.zeros((360/gs,360/gs)) # obtaining grids based on omega bin
size
add_samp = 250 # number of random samples aded in each iteration
total_samp = ((360/gs)**2)*num_per_grid # total number of geometries
added to the system uniformly

path = input('Insert path to file (within qoutes): \nSample path
"/home/user/workdir/" \n\n ')
count = 0
a = 1
b = add_samp
tol = 0.9 # seting 90% tolerance for total number of geometries generated

anum1 = int(input('Number of beads in protein-1  '))
anum2 = int(input('Number of beads in protein-2  '))
ref1 = input('Insert group of reference residues for 0 angle position for
prot1 (within quotes): \nSample "4-25" \n\n')
ref2 = input('Insert group of reference residues for 0 angle position for
prot2 (within quotes, enter the same as prot1 if homomer): \nSample "4-
25" \n\n')
ds = int(input('Enter ds value (nm): '))
dimer_type = int(input('1. Homomer  2. Heteromer \nEnter 1 or 2 to choose
dimer type '))-1

init_geom_commands = ["rand_geom.sh -cg A=prot1.pdb,prot1.top -c 2 -c 2!
-dsep "+str(ds)+" -n "+str(add_samp)+" --insane{-pbc=rectangular,-
l=DOPC:2,-l=DPPC:2,-l=CHOL:1,-sol=W:9,-sol=WF:1,-salt=0.15,-x=10,-y=10,-
z=9}","rand_geom.sh -cg A=prot1.pdb,prot1.top -cg B=prot2.pdb,prot2.top -
c 2! -dsep "+str(ds)+" -n "+str(add_samp)+" --insane{-pbc=rectangular,-
l=DOPC:2,-l=DPPC:2,-l=CHOL:1,-sol=W:9,-sol=WF:1,-salt=0.15,-x=10,-y=10,-
z=9}"] # similar command-line as daft method

file_ext = ["A-A/conf/","A-B/conf/"]
```

Also, please feel free to change the membrane lipid compositions in the included insane.py commands in `rand_geom.sh` to suite your desired membrane setup.

Run:

./P_setup.py

(This takes a while to run and finish because of multiple iterations to obtain a uniform distribution. There may be some “warnings: Failed to guess atom type…” displayed by the MDAnalysis package, please disregard those. Please note down the total number of geometries generated that is displayed at the end of the run, we will need it later in P_analysis. It looks like shown below:

…

```
Grids sampled $\boxed{2374.0}$/2592


Iteration: 15
```

)

grids.py can be used to generate a plot to visualize the initial geometry distribution on the landscape.

Run MD simulations on all the geometries

Input: mdp files, run script, run_setup script, etc

- Example mdp files (`em.mdp, NVT.mdp, NPT.mdp, PROD.mdp`) with suggested length of simulations, sample run script (`run.sh`) and example bulk job submission script (`run_setup.py`) have been provided which can be edited to suit the cluster specifications you are using.
- Make sure to change the energygrps index names on the mdp files*** (Example shown below)

```
define                        = -DPOSRES
integrator                    = md
tinit                         = 0.0
dt                            = 0.020
nsteps                        = 1250000

nstcomm                       = 100
energygrps                    = A_A B_B Membrane Solvent
comm-grps                     = Solute Membrane Solvent
```

Set `energygrps                    = A_A B_B Membrane Solvent` for Heteromer

`energygrps                    = A_A B_A Membrane Solvent` for Homomer

Analysis

P_analysis.py

Inputs:

- The input parameters are similar to the ones prompted by *P_setup.py.*
- Make sure to provide the appropriate the energy group for obtaining interaction energies using "*gmx energy*" in line 58 of the code. The energy group may not necessarily be "47" depending on the gromacs version, please run the energy command for one geometry to know the correct index number for LJ interaction between either of the proteins.

```
os.system("echo 47|gmx energy -f PROD.edr -s PROD.tpr -o energy.xvg")
```

- Reference group residues (same as in *P_setup.py*)

Run:

./P_analysis.py

Outputs:

- Processed trajectories
- Interaction energies over the trajectory
- Rotational angles if the two proteins over the trajectory
- Distance between the two protein over the trajectory
- Text file listing all the configurations that qualify as good geometries (described in the paper). This text file will later be used by *panel.py* to generate PANEL plots.

Results

panel.py

Inputs: As prompted by the code while running.

Run:

./panel.py

Outputs:

- Minimum energy landscape
- Average energy landscape
- Population landscape
- Coverage
- Text files containing data for "minimum energies", "average energies", "frequencies" over the rotational space and configurations corresponding to the minimum energies.

```
Plots: Min_en_grid.png, Pop_grid.png, Avg_en_grid.png, Grid_cov.png
```

```
Text data: min_en_grids.txt, max_en_grids.txt, av_en_grids.txt,
pop_grids.txt, min_conf.txt, max_conf.txt, min_snap.txt, max_snap.txt
```

Please feel free to make changes to the specification for generating plots or wherever you feel necessary.

Please cite us:

Rajagopal, N.; Nangia, S. Obtaining Protein Association Energy Landscape for Integral Membrane Proteins. *Journal of Chemical Theory and Computation* 2019. DOI: 10.1021/acs.jctc.9b00626