## CONTENTS

1 Introduction 1
   1.1 Required Programs .............................................. 1
   1.2 Recommended Programs ........................................ 1
   1.3 Getting Started ............................................. 1

2 ANM Calculations 3
   2.1 Atom Selection ............................................... 3
   2.2 ANM Calculation ............................................. 4
   2.3 Analysis & Plotting .......................................... 5
   2.4 Visualization ................................................ 5
   2.5 Extend Model ................................................. 6
   2.6 Save Results ................................................ 6
   2.7 More Examples .............................................. 7

3 Sample Conformations 8
   3.1 Load results ................................................. 8
   3.2 Sampling .................................................... 8
   3.3 Analysis .................................................... 9
   3.4 Write conformations ....................................... 10
   3.5 Visualization ............................................... 11

4 Optimize Conformations 12
   4.1 Configuration ............................................... 12
   4.2 Optimization ................................................ 13

5 Analyze Conformations 14
   5.1 Parse conformations ....................................... 14
   5.2 Calculate RMSD change ..................................... 14
   5.3 Select a diverse set ....................................... 15
   5.4 Visualization ............................................... 16

A Acknowledgment 18
CHAPTER
ONE

INTRODUCTION

This tutorial describes sampling alternate protein conformations along Anisotropic Network Model (ANM) modes, then optimizing them using a molecular dynamics program. Conformations obtained in this way can be useful in, for example, docking studies when the target binding site is flexible and can be affected by motions of protein along collective modes.

We will use a structure of mitogen-activated protein kinase 14 (MAPK14), which is also known as p38 MAPK. The structure identifier is 1p38. PDB and PSF files are provided in documentation files.

1.1 Required Programs

Latest version of ProDy, Matplotlib, and NAMD are required.

1.2 Recommended Programs

List any recommended programs, such as IPython, Scipy, etc.

1.3 Getting Started

To follow this tutorial, you will need the following files which can be downloaded from Tutorials.

471 min.conf
437K p38.pdb
1.3M p38.psf

We recommend that you will follow this tutorial by typing commands in an IPython session, e.g.:

$ ipython

1http://csb.pitt.edu/ProDy/tutorials/enm_analysis/anm.html#anm
2http://en.wikipedia.org/wiki/MAPK14
3http://www.pdb.org/pdb/explore/explore.do?structureId=1p38
4http://csb.pitt.edu/ProDy/getprody.html
5http://matplotlib.org/
6http://www.ks.uiuc.edu/Research/namd/
7http://ipython.org/
8http://scipy.org/
9http://csb.pitt.edu/ProDy/tutorials/index.html#tutorials
or with pylab environment:

```
$ ipython --pylab
```

First, we will make necessary imports from ProDy and Matplotlib packages.

```
In [1]: from prody import *
In [2]: from pylab import *
In [3]: ion()
```

We have included these imports in every part of the tutorial, so that code copied from the online pages is complete. You do not need to repeat imports in the same Python session.
ANM CALCULATIONS

2.1 Atom Selection

First, we parse p38 structure:

```python
In [4]: p38 = parsePDB('conformational_sampling_files/p38.pdb')
```

```bash
In [5]: p38
Out[5]: <AtomGroup: p38 (5658 atoms)>
```

Let’s take a look at the structure:

```bash
In [6]: showProtein(p38);
```

```bash
In [7]: legend();
```
Conformational Sampling, Release 1.4.10

Note that this structure has hydrogen atoms which are added using PSFGEN that comes with NAMD:

In [8]: p38.numAtoms('hydrogen')
Out[8]: 2824

We will perform ANM calculations for 351 Cα atoms of the structure:

In [9]: p38_ca = p38.ca
In [10]: p38_ca
Out[10]: <Selection: 'ca' from p38 (351 atoms)>

2.2 ANM Calculation

First, let's instantiate an ANM object:

In [11]: p38_anm = ANM('p38 ca')
In [12]: p38_anm
Out[12]: <ANM: p38 ca (0 modes; 0 nodes)>

Now, we can build Hessian matrix, simply by calling ANM.buildHessian() method:

In [13]: p38_anm.buildHessian(p38_ca)
In [14]: p38_anm
Out[14]: <ANM: p38 ca (0 modes; 351 nodes)>

We see that ANM object contains 351 nodes, which correspond to the Cα atoms.

We will calculate only top ranking three ANM modes, since we are going to use only that many in sampling:

Note: ANM calculation can be found at:

1. http://csb.pitt.edu/ProDy/reference/dynamics/anm.html#ANM
2. http://csb.pitt.edu/ProDy/reference/dynamics/anm.html#ANM.buildHessian
3. http://csb.pitt.edu/ProDy/reference/dynamics/anm.html#ANM
In [15]: p38_anm.calcModes(n_modes=3)

In [16]: p38_anm
Out[16]: <ANM: p38 ca (3 modes; 351 nodes)>

2.3 Analysis & Plotting

Let's plot mobility of residues along ANM modes:

In [17]: showSqFlucts(p38_anm);

![ANM Mobility Plot](image)

We can also calculate collectivity of these modes as follows:

In [18]: for mode in p38_anm:
   ....:    print(' Mode {} from ANM p38 ca collectivity: {:.4f}'.format(str(mode), calcCollectivity(mode)))
   ....:
Mode 1 from ANM p38 ca collectivity: 0.618084923455
Mode 2 from ANM p38 ca collectivity: 0.585153336557
Mode 3 from ANM p38 ca collectivity: 0.634434845453

2.4 Visualization

You can visualize ANM modes using Normal Mode Wizard\(^4\). You need to write an .nmd file using writeNMD()\(^5\) and open it using VMD:

In [19]: writeNMD('p38_anm.nmd', p38_anm, p38_ca)
Out[19]: 'p38_anm.nmd'

For visualization, you can use viewNMDinVMD()\(^6\), i.e. viewNMDinVMD('p38_anm.nmd')

---

\(^4\)http://csb.pitt.edu/ProDy/tutorials/nmwiz_tutorial/intro.html#nmwiz
\(^5\)http://csb.pitt.edu/ProDy/reference/dynamics/nmdfile.html#writeNMD
\(^6\)http://csb.pitt.edu/ProDy/reference/dynamics/nmdfile.html#viewNMDinVMD
2.5 Extend Model

We want to use ANM model to sample all atoms conformations of p38 MAPK, but we have a coarse-grained model. We will use `extendModel()` function for this purpose:

```
In [20]: p38_anm_ext, p38_all = extendModel(p38_anm, p38_ca, p38, norm=True)
```

```
In [21]: p38_anm_ext
Out[21]: <NMA: Extended ANM p38 ca (3 modes; 5658 atoms)>
```

```
In [22]: p38_all
Out[22]: <AtomMap: AtomGroup p38 from p38 (5658 atoms)>
```

Note `p38_anm_ext` is an NMA\(^8\) model, which has similar features as an ANM\(^9\) object. Extended model has 3 modes, but 5668 atoms as opposed to 351 nodes in the original ANM\(^10\) model.

Let's plot mobility of residues again to help understand what extending a model does:

```
In [23]: showSqFlucts(p38_anm_ext);
```

As you see, shape of the mobility plot is identical. In the extended model, each in the same direction as the C\(\alpha\) atoms of the residues that they belong to. The mobility profile is scaled down, however, due to renormalization of the mode vectors.

2.6 Save Results

Now let's save the original and extended model, and atoms:

```
In [24]: saveAtoms(p38)
Out[24]: 'p38.ag.npz'
```

```
In [25]: saveModel(p38_anm)
```

---

\(^7\)http://csb.pitt.edu/ProDy/reference/dynamics/editing.html#extendModel
\(^8\)http://csb.pitt.edu/ProDy/reference/dynamics/nma.html#NMA
\(^9\)http://csb.pitt.edu/ProDy/reference/dynamics/anm.html#ANM
\(^10\)http://csb.pitt.edu/ProDy/reference/dynamics/anm.html#ANM
2.7 More Examples

We have performed a quick ANM calculation and extended the resulting model to all atoms of the structure. You can see more examples on this in Elastic Network Models\textsuperscript{11} tutorial.

\textsuperscript{11}http://csb.pitt.edu/ProDy/tutorials/enm_analysis/index.html#enm-analysis
SAMPLE CONFORMATIONS

In this part, we will sample conformations along ANM modes.

In [1]: from prody import *

In [2]: from pylab import *

In [3]: ion()

3.1 Load results

First, we load results produced in the previous part. If you are in the same Python session, you don’t need to do this.

In [4]: p38 = loadAtoms('p38.ag.npz')

In [5]: p38_ann = loadModel('p38_ca.anm.npz')

In [6]: p38_ann_ext = loadModel('p38_ext.nma.npz')

3.2 Sampling

We will use sampleModes() function:

In [7]: ens = sampleModes(p38_ann_ext, atoms=p38.protein, n_confs=40, rmsd=1.0)

In [8]: ens
Out[8]: <Ensemble: Conformations along NMA Extended ANM p38 ca (40 conformations; 5658 atoms)>

We passed extended model, p38 structure, and two other parameters. This will produce 40 (n_confs) conformations. The conformations will have an average 1.0 Å RMSD from the input structure.

We can write this ensemble in .dcd for visualization in VMD:

In [9]: writeDCD('p38all.dcd', ens)
Out[9]: 'p38all.dcd'

1http://csb.pitt.edu/ProDy/reference/dynamics/sampling.html#sampleModes
3.3 Analysis

Let’s analyze the Ensemble\(^2\) by plotting RMSD of conformations to the input structure:

In [10]: rmsd = ens.getRMSDs()

In [11]: hist(rmsd, normed=False);

In [12]: xlabel(‘RMSD’);

![Histogram of RMSD](image)

This histogram might look like a flat distribution due to the small size of the ensemble. For larger numbers of conformations it will get closer to a normal distribution. Let’s calculate average and extremum RMSD values:

In [13]: rmsd.mean()
Out[13]: 1.0002313557559686

In [14]: rmsd.max()
Out[14]: 2.1898300210653279

In [15]: rmsd.min()
Out[15]: 0.23592822325178625

Let’s see the projection of these conformations in the ANM slow mode space:

In [16]: showProjection(ens, p38_anm_ext[:3], rmsd=True);

In [17]: proj = calcProjection(ens, p38_anm_ext[:3])

\(^2\)http://csb.pitt.edu/ProDy/reference/ensemble/ensemble.html#Ensemble
3.4 Write conformations

We will write them in p38_ensemble folder:

In [18]: mkdir -p p38_ensemble

Let's add the conformations to the AtomGroup object and set beta values of Cα atoms to 1 and of other atoms to 0:

In [19]: p38.addCoordset(ens.getCoordsets())

In [20]: p38
Out[20]: <AtomGroup: p38 (5658 atoms; active #0 of 41 coordsets)>

In [21]: p38.all.setBetas(0)

In [22]: p38.ca.setBetas(1)

In the next step, we will place a harmonic constraint on atoms with beta values 1. The optimization is aims for refining covalent geometry of atoms. We do not want the new Cα to change much to keep the refined ensemble diverse. We can easily verify that only Cα atoms have beta values set to 1:

In [23]: p38.ca == p38.beta_1
Out[23]: True

Now we write these conformations out:

In [24]: import os

In [25]: for i in range(1, p38.numCoordsets()):  # skipping 0th coordinate set
       fn = os.path.join('p38_ensemble', 'p38_' + str(i) + '.pdb')
       writePDB(fn, p38, csets=i)

---

3 http://csb.pitt.edu/ProDy/reference/atomic/atomgroup.html#AtomGroup
4 http://csb.pitt.edu/ProDy/reference/atomic/fields.html#term-beta
3.5 Visualization

You can visualize all of these conformations using VMD as follows:

$ vmd -m p38_ensemble/*pdb
In this part we will optimize the geometries of conformations generated in the previous step using NAMD.

### 4.1 Configuration

Let’s find the location of NAMD executable:

```
In [1]: from prody.utilities import which

In [2]: namd2 = which('namd2')

In [3]: namd2
Out[3]: '/usr/local/bin/namd2'
```

We will need a force field file for energy minimization. VMD ships with CHARMM force field files. We can find their location as follows:

```
In [4]: with open('where_is_charmmpar.tcl', 'w') as inp:
    ...:     inp.write('''
    ...:     global env;
    ...:     puts $env(CHARMMPARDIR);
    ...:     exit;
    ...:'
    ...

When you run the following command, you will see more output than the following, but the line that you need will be at the end:

```
In [5]: !vmd -e where_is_charmmpar.tcl
/home/abakan/Programs/vmd-1.9.1/plugins/noarch/tcl/readcharmmpar1.2
Info) VMD for LINUXAMD64, version 1.9.1 (February 1, 2012)
Info) Exiting normally.
```

```
In [6]: import os

In [7]: par = os.path.join('/home/abakan/Programs/vmd-1.9.1/
    ...:     'plugins/noarch/tcl/readcharmmpar1.2',
    ...:     'par_all27_prot_lipid_na.inp')
    ...
Let’s make a folder for writing optimization input and output files:

```
In [8]: mkdir -p p38_optimize
```

We will write an NAMD configuration file for each conformation based on `min.conf` file:
In [9]: import glob

In [10]: conf = open('conformational_sampling_files/min.conf').read()

In [11]: for pdb in glob.glob(os.path.join('p38_ensemble', '*.pdb')):
       ...
       fn = os.path.splitext(os.path.split(pdb)[1])[0]
       ...
       pdb = os.path.join('..', pdb)
       ...
       out = open(os.path.join('p38_optimize', fn + '.conf'), 'w')
       ...
       out.write(conf.format(
       ...
           out=fn, pdb=pdb,
       ...
           par=par))
       ...
       out.close()
       ...

4.2 Optimization

Now we will run NAMD to optimize each of these conformations. We make a list of commands that we
want to execute:

In [12]: os.chdir('p38_optimize') # we will run commands in this folder

In [13]: cmds = []

In [14]: for conf in glob.glob('*.conf'):
       ...
       fn = os.path.splitext(conf)[0]
       ...
       cmds.append('namd2 ' + conf + ' > ' + fn + '.log')
       ...

In [15]: cmds[:2]
Out[15]: ['namd2 p38_18.conf > p38_18.log', 'namd2 p38_36.conf > p38_36.log']

We will run these commands using multiprocessing\textsuperscript{1} module. We will allocate 3 processors for the job:

In [16]: from multiprocessing import Pool

In [17]: pool = Pool(3) # number of CPUs to use

In [18]: signals = pool.map(os.system, cmds)

signals will collect the output from execution of NAMD. If everything goes right, we should have only
0s.

In [19]: set(signals)
Out[19]: {34304}

All NAMD output should be in p38_optimize folder. We go back to original folder as follows:

In [20]: os.chdir('..')

\textsuperscript{1}\url{http://docs.python.org/library/multiprocessing.html#multiprocessing}

4.2. Optimization 13
First, necessary imports:

```python
In [1]: from prody import *
In [2]: from pylab import *
In [3]: ion()
In [4]: import os, glob
```

### 5.1 Parse conformations

Now, let’s read initial and refined conformations:

```python
In [5]: initial = AtomGroup('p38 initial')
In [6]: refined = AtomGroup('p38 refined')
In [7]: for pdb in glob.glob('p38_ensemble/*pdb'):
   ...:     fn = os.path.splitext(os.path.split(pdb)[1])[0]
   ...:     opt = os.path.join('p38_optimize', fn + '.coor')
   ...:     parsePDB(pdb, ag=initial)
   ...:     parsePDB(opt, ag=refined)
```

```text
In [8]: initial
Out[8]: <AtomGroup: p38 initial (5658 atoms; active #0 of 40 coordsets)>
In [9]: refined
Out[9]: <AtomGroup: p38 refined (5658 atoms; active #0 of 40 coordsets)>
```

### 5.2 Calculate RMSD change

We can plot RMSD change after refinement as follows:

```python
In [10]: rmsd_ca = []
In [11]: rmsd_all = []
```
In [12]: initial_ca = initial.ca

In [13]: refined_ca = refined.ca

In [14]: for i in range(initial.numCoordsets()):
   ....:     initial.setACSIndex(i)
   ....:     refined.setACSIndex(i)
   ....:     initial_ca.setACSIndex(i)
   ....:     refined_ca.setACSIndex(i)
   ....:     rmsd_ca.append(calcRMSD(initial_ca, refined_ca))
   ....:     rmsd_all.append(calcRMSD(initial, refined))

In [15]: plot(rmsd_all, label='all');

In [16]: plot(rmsd_ca, label='ca');

In [17]: xlabel('Conformation index');

In [18]: ylabel('RMSD');

In [19]: legend();

5.3 Select a diverse set

To select a diverse set of refined conformations, let’s calculate average RMSD for each conformation to others:

In [20]: rmsd_mean = []

In [21]: for i in range(refined.numCoordsets()):
   ....:     refined.setACSIndex(i)
   ....:     alignCoordsets(refined)
   ....:     rmsd = calcRMSD(refined)
   ....:     rmsd_mean.append(rmsd.sum() / (len(rmsd) - 1))
Let’s select conformations that are 1.2 Å away from other on average:

```
In [25]: rmsd_mean = array(rmsd_mean)
In [26]: selected = (rmsd_mean >= 1.2).nonzero()[0] + 1
```

```
[1, 2, 3, 5, 6, 7, 8, 10, 11, 12, 13, 15, 16, 18, 19, 20, 21, 23, 24, 25, 26, 27, 28, 30, 32, 33, 34, 35, 36, 37, 40]
```

```
In [28]: len(selected)
Out [28]: 31
```

### 5.4 Visualization

When you visualize the refined ensemble, you should see something similar to this:
5.4. Visualization
ACKNOWLEDGMENT

Continued development of Protein Dynamics Software ProDy is supported by NIH through R01 GM099738 award. Development of this tutorial is supported by NIH funded Biomedical Technology and Research Center (BTRC) on High Performance Computing for Multiscale Modeling of Biological Systems (MMBios\(^1\)) (P41 GM103712).

\(^1\)http://mmbios.org/