ProDy

Protein Dynamics & Sequence Analysis

Conformational Sampling

Release 1.7

Ahmet Bakan, Cihan Kaya

November 12, 2015

CHAPTER

ONE

INTRODUCTION

This tutorial describes sampling alternate protein conformations along Bond-Bending Elastic Network Model (bbENM)[1] 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[2]), which is also known as p38 MAPK. The structure identifier is 1p38[3]. PDB and PSF files are provided in documentation files.

1.1 Required Programs

Latest version of ProDy[4], Matplotlib[5], and NAMD[6] are required.

1.2 Recommended Programs

IPython[7] is highly recommended for interactive usage.

1.3 Getting Started

To follow this tutorial, you will need the following files:

```
    471B Oct 29 20:15 min.conf
    437K Oct 29 20:15 p38.pdb
    1.3M Oct 29 20:15 p38.psf
```

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

```
$ ipython
```

or with pylab environment:

[1] http://prody.csb.pitt.edu/tutorials/enm_analysis/bbenm.html#anm
[2] http://en.wikipedia.org/wiki/MAPK14
[3] http://www.pdb.org/pdb/explore/explore.do?structureId=1p38
[4] http://prody.csb.pitt.edu
[5] http://matplotlib.org
[6] http://www.ks.uiuc.edu/Research/namd/
[7] http://ipython.org

```
$ 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.

CHAPTER

TWO

ANM CALCULATIONS

Required imports:

```
In [1]: from prody import *

In [2]: from pylab import *

In [3]: ion()
```

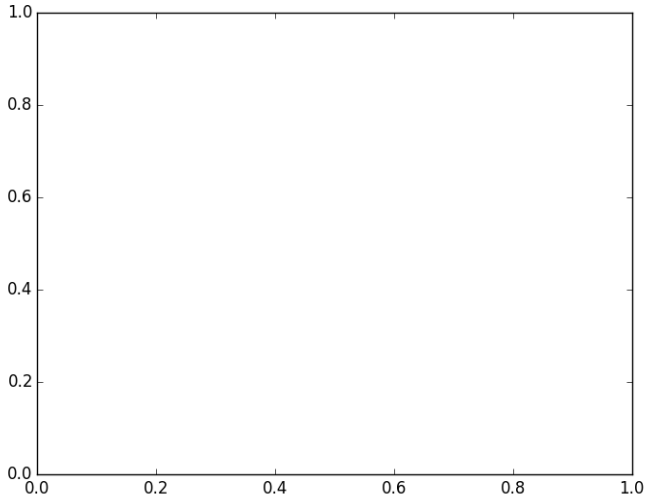
2.1 Atom Selection

First, we parse p38 structure `p38.pdb`:

```
In [4]: p38 = parsePDB('1p38.pdb')
---------------------------------------------------------------------------
IOError                                   Traceback (most recent call last)
<ipython-input-4-8144910c8366> in <module>()
----> 1 p38 = parsePDB('1p38.pdb')

/Users/cihank/anaconda/lib/python2.7/site-packages/prody/proteins/pdbfile.pyc in parsePDB(pdb, **kwa
    103         else:
    104             raise IOError('{0} is not a valid filename or a valid PDB '
--> 105                           'identifier.'.format(pdb))
    106     if title is None:
    107         title, ext = os.path.splitext(os.path.split(pdb)[1])

IOError: 1p38.pdb is not a valid filename or a valid PDB identifier.

In [5]: p38
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-5-76c327520983> in <module>()
----> 1 p38

NameError: name 'p38' is not defined
```

Let's take a look at the structure:

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

In [7]: legend();
```



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

```
In [8]: p38.numAtoms('hydrogen')
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-8-8a2590545cc8> in <module>()
----> 1 p38.numAtoms('hydrogen')

NameError: name 'p38' is not defined
```

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

```
In [9]: p38_ca = p38.ca
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-9-a272a513ca68> in <module>()
----> 1 p38_ca = p38.ca

NameError: name 'p38' is not defined

In [10]: p38_ca
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-10-08013b2d37a0> in <module>()
----> 1 p38_ca

NameError: name 'p38_ca' is not defined
```

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)
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-13-58dc032316ca> in <module>()
----> 1 p38_anm.buildHessian(p38_ca)

NameError: name 'p38_ca' is not defined

In [14]: p38_anm
Out[14]: <ANM: p38 ca (0 modes; 0 nodes)>
```

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

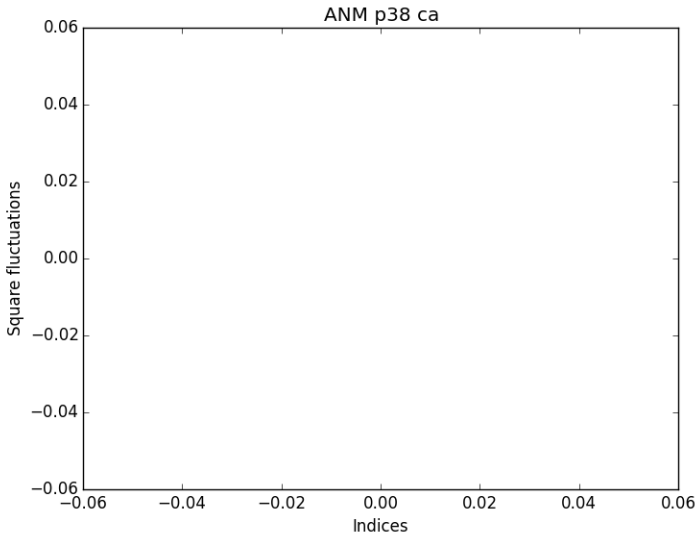
We will calculate only top ranking three ANM modes, since we are going to use only that many in sampling:

```
In [15]: p38_anm.calcModes(n_modes=3)
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-15-93720636aa56> in <module>()
----> 1 p38_anm.calcModes(n_modes=3)

/Users/cihank/anaconda/lib/python2.7/site-packages/prody/dynamics/anm.pyc in calcModes(self, n_modes,
    195
    196         if self._hessian is None:
--> 197             raise ValueError('Hessian matrix is not built or set')
    198         assert n_modes is None or isinstance(n_modes, int) and n_modes > 0, \
    199             'n_modes must be a positive integer'

ValueError: Hessian matrix is not built or set

In [16]: p38_anm
Out[16]: <ANM: p38 ca (0 modes; 0 nodes)>
```

2.3 Analysis & Plotting

Let's plot mobility of residues along ANM modes:

```
In [17]: showSqFlucts(p38_anm);
```



We can also calculate collectivity of these modes as follows:

```
In [18]: for mode in p38_anm:
   ....:     print('{}\tcollectivity: {}'.format(str(mode), calcCollectivity(mode)))
   ....:
```

2.4 Visualization

You can visualize ANM modes using Normal Mode Wizard[1]. You need to write an `.nmd` file using `writeNMD()` and open it using VMD:

```
In [19]: writeNMD('p38_anm.nmd', p38_anm, p38_ca)
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-19-a2a9a000beb5> in <module>()
----> 1 writeNMD('p38_anm.nmd', p38_anm, p38_ca)

NameError: name 'p38_ca' is not defined
```

For visualization, you can use `viewNMDinVMD()`, i.e. `viewNMDinVMD('p38_anm.nmd')`

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)
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-20-4b12cca159f5> in <module>()
----> 1 p38_anm_ext, p38_all = extendModel(p38_anm, p38_ca, p38, norm=True)
```

[1] http://prody.csb.pitt.edu/tutorials/nmwiz_tutorial/intro.html#nmwiz

```
NameError: name 'p38_ca' is not defined

In [21]: p38_anm_ext
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-21-63f563ebae06> in <module>()
----> 1 p38_anm_ext

NameError: name 'p38_anm_ext' is not defined

In [22]: p38_all
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-22-1ddfda9d25d9> in <module>()
----> 1 p38_all

NameError: name 'p38_all' is not defined
```

Note `p38_anm_ext` is an `NMA` model, which has similar features as an `ANM` object. Extended model has 3 modes, but 5668 atoms as opposed to 351 nodes in the original `ANM` 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 Cα 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)
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
```

```
<ipython-input-24-6b60dbbfc42d> in <module>()
----> 1 saveAtoms(p38)

NameError: name 'p38' is not defined

In [25]: saveModel(p38_anm)
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-25-e6b8b86e3cb6> in <module>()
----> 1 saveModel(p38_anm)

/Users/cihank/anaconda/lib/python2.7/site-packages/prody/dynamics/functions.pyc in saveModel(nma, fi
     38         raise TypeError('invalid type for nma, {0}'.format(type(nma)))
     39     if len(nma) == 0:
---> 40         raise ValueError('nma instance does not contain data')
     41
     42     dict_ = nma.__dict__

ValueError: nma instance does not contain data

In [26]: saveModel(p38_anm_ext, 'p38_ext')
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-26-7034f04e093d> in <module>()
----> 1 saveModel(p38_anm_ext, 'p38_ext')

NameError: name 'p38_anm_ext' is not defined
```

2.7 More Examples

We have performed a quick ANM calculation and extended the resulting model to all atoms of of the structure. You can see more examples on this in Elastic Network Models[2] tutorial.

[2] http://prody.csb.pitt.edu/tutorials/enm_analysis/index.html#enm-analysis

CHAPTER

THREE

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_anm = loadModel('p38_ca.anm.npz')

In [6]: p38_anm_ext = loadModel('p38_ext.nma.npz')
```

3.2 Sampling

We will use `sampleModes()` function:

```
In [7]: ens = sampleModes(p38_anm_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.

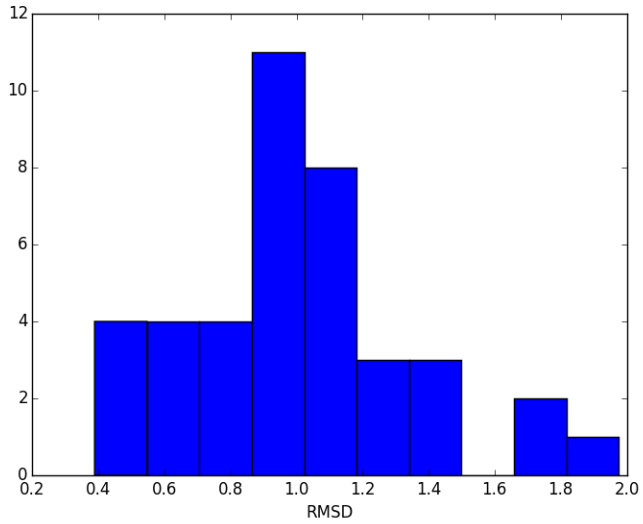
We can write this ensemble in `.dcd` for visualization in VMD:

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

3.3 Analysis

Let's analyze the `Ensemble` by plotting RMSD of conformations to the input structure:

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

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

In [12]: xlabel('RMSD');
```

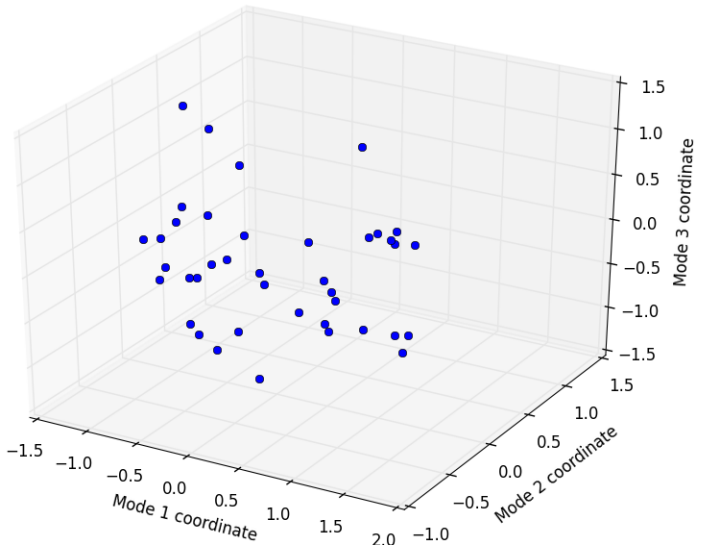


This histogram might look like a flat distribution due to the small size of the ensemble. For larger numbers of conformations it will get closer to a normal distribution. Let's calculate average and extremum RMSD values:

```
In [13]: rmsd.mean()
Out[13]: 1.001016667861981

In [14]: rmsd.max()
Out[14]: 1.9748865174145691

In [15]: rmsd.min()
Out[15]: 0.38921142516006302
```

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])
```



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[1] 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)
   ....:
```

[1] http://prody.csb.pitt.edu/manual/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
```

CHAPTER

FOUR

OPTIMIZE CONFORMATIONS

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]: '/Users/cihank/anaconda/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('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_1.conf > p38_1.log', 'namd2 p38_10.conf > p38_10.log']
```

We will run these commands using `multiprocessing`[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 origional folder as follows:

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

[1] http://docs.python.org/library/multiprocessing.html#module-multiprocessing

CHAPTER

FIVE

ANALYZE CONFORMATIONS

First, necessary imports:

```
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:

```
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)
   ...:
```

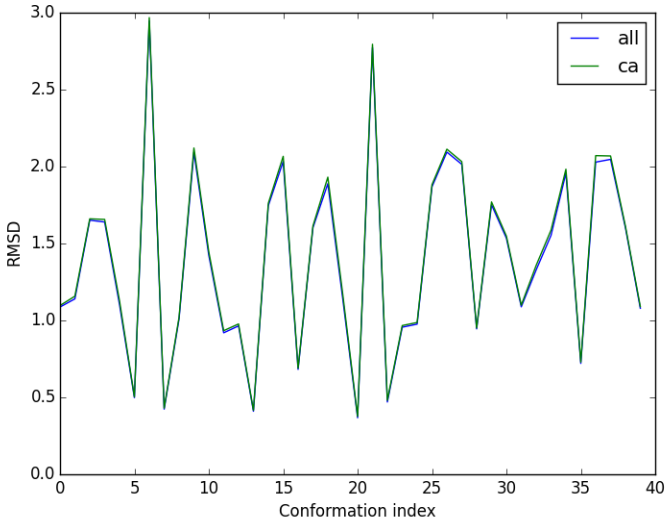
```
In [8]: initial
Out[8]: <AtomGroup: p38 initial (5658 atoms; active #0 of 40 coordsets)>

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:

```
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();
```

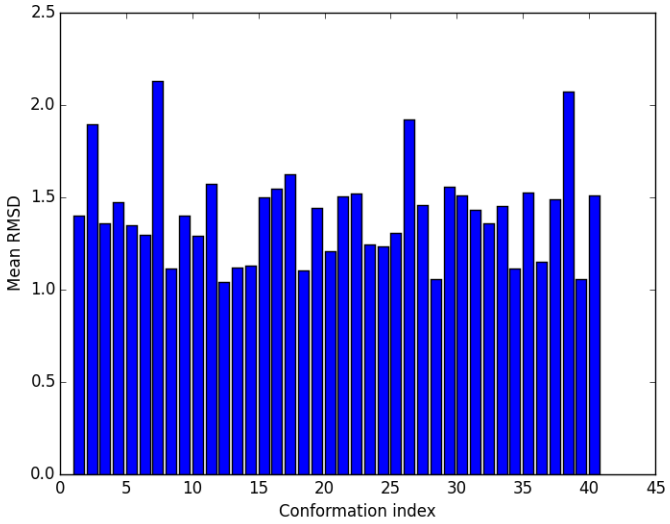


5.3 Select a diverse set

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))
   ....:
```

```
In [22]: bar(arange(1, len(rmsd_mean) + 1), rmsd_mean);

In [23]: xlabel('Conformation index');

In [24]: ylabel('Mean RMSD');
```

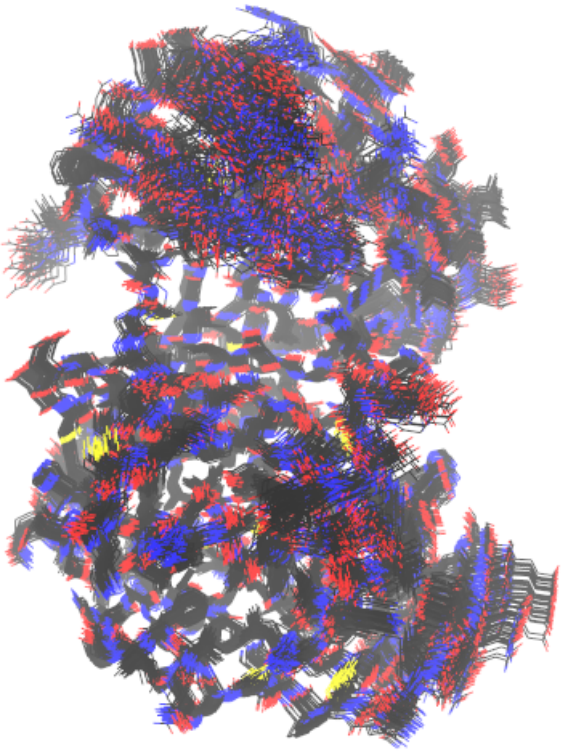


Let's select conformations that are 1.2 Å away from other on average:

```
In [25]: rmsd_mean = array(rmsd_mean)

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

In [27]: selected
Out[27]:
array([ 1,  2,  3,  4,  5,  6,  7,  9, 10, 11, 15, 16, 17, 19, 20, 21, 22,
       23, 24, 25, 26, 27, 29, 30, 31, 32, 33, 35, 37, 38, 40])

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

5.4 Visualization

When you visualize the refined ensemble, you should see something similar to this:



Acknowledgments

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/