ProDy

Protein Dynamics & Sequence Analysis

PRS

Release

James Krieger

June 07, 2024

CHAPTER

ONE

INTRODUCTION

This tutorial demonstrates how to use perturbation response scanning (PRS) to determine sensors and effectors, which are important for allosteric signal transduction. The PRS approach is derived from linear response theory where perturbation forces are applied via a covariance matrix, which can be derived from elastic network models or MD simulations.

The example used in this tutorial is the AMPA-type ionotropic glutamate receptor (AMPAR; PDB 3kg2), which we studied using this method (see Figure 6 of *[AD15]*).

The theory was originally described in *[CA09]* and *[CA10]*, and extended to include sensors in *[IG14]*.

1.1 Required Programs

The latest version of **ProDy_** is recommended along with **NumPy_** and **Matplotlib_**. **IPython_** is highly recommended for interactive usage.

1.2 Getting Started

To follow this tutorial, you will not need any additional files.

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

```
$ ipython
```

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.

1.3 How to Cite

If you benefited from Perturbation Response Analysis in your research, please cite the following papers:

CHAPTER

TWO

CALCULATIONS

Here are the required imports again. You do not need to repeat them if you are still in the same python session.

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

In [2]: from pylab import *

In [3]: ion()
```

2.1 Loading a structure and applying the anisotropic network model

First, we parse a structure that we want to analyse with PRS. For this tutorial, we will fetch the near-intact full length AMPAR structure 3kg2 from the PDB. We import a subset containing the calpha atoms, which we will use in downstream steps.

```
In [4]: ampar_ca = parsePDB('3kg2', subset='ca')
```

Next, create an ANM instance and calculate modes from which the covariance matrix can be calculated. We ask for all modes rather than the default subset of the first 20 global modes to reproduce the published data.

```
In [5]: anm_ampar = ANM('AMPAR 3kg2')

In [6]: anm_ampar.buildHessian(ampar_ca)

In [7]: anm_ampar.calcModes('all')
```

PRS can also be performed with any other model from which a covariance matrix can be calculated including GNM and PCA. A PCA model can also be used with an external covariance matrix using its `PCA.setCovariance()` method. ANM is used here to reproduce the published data.

2.2 Showing the PRS matrix with effectiveness and sensitivity profiles

The PRS matrix is then calculated from the covariance matrix from the ANM of the AMPAR. This can be automatically carried out together with plotting using the following code. We supply atoms to allow us to delineate residue numbers and chains.

```
In [8]: show = showPerturbResponse(anm_ampar, atoms=ampar_ca)
```

2.3 Saving the model

You can also save the ANM model in a .npz file for loading again and write an .nmd for visualizing in the NMWiz[1]. Adding *matrices=True* allows us to save the covariance matrix so we can calculate the PRS faster in future analyses.

```
In [9]: saveModel(anm_ampar, '3kg2', matrices=True)
Out[9]: '3kg2.anm.npz'

In [10]: writeNMD('anm_3kg2.nmd', anm_ampar, ampar_ca)
Out[10]: 'anm_3kg2.nmd'
```

The next page illustrates some more detailed analyses.

[1] http://prody.csb.pitt.edu/nmwiz/

CHAPTER

THREE

ANALYSIS

Here are the required imports again. You do not need to repeat them if you are still in the same python session.

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

In [2]: from pylab import *

In [3]: ion()
```

You can load the model again if you are starting a new session as follows:

```
In [4]: anm_ampar = loadModel('3kg2.anm.npz')

In [5]: ampar_ca = parsePDB('3kg2', subset='ca')
```

3.1 Plotting effectiveness and sensitivity profiles

We can use showPerturbResponse with the matrix=False option to plot the effectiveness and sensitivity profiles colored by chain:

```
In [6]: show = showPerturbResponse(anm_ampar, atoms=ampar_ca,
   ...:                            show_matrix=False)
   ...:
```

3.2 Plotting residue-specific effectiveness and sensitivity profiles

To look at the effectiveness that perturbing a residue has in eliciting a response in individual residues (instead of its overall effectiveness) or to look at the sensitivity of a residue to perturbations of individual residues (instead of its overall sensitivity), we read out rows or columns from the perturbation response matrix. This can again be shown in a plot as in Figure 6D.

```
In [7]: show = showPerturbResponse(anm_ampar, atoms=ampar_ca,
   ...:                            show_matrix=False,
   ...:                            select='chain B and resnum 84')
   ...:
```

We can also calculate the PRS matrix and profiles separately from showPerturbResponse. This gives us more flexibility with what we show and enables us to do other things with the return values. For example, we could apply a cutoff to identify residues with particularly high effectiveness and sensitivity as effectors and sensors, or slice out individual rows or columns and write them into PDB files for visualization (see below).

```
In [8]: prs_mat, effectiveness, sensitivity = calcPerturbResponse(anm_ampar)
```

3.3 Writing effectiveness and sensitivity profiles to PDB for visualization

It is sometimes more helpful to understand what is happening using a colored structure. To achieve this we can overwrite the B-factor or occupancy column of PDB file and use PyMOL or VMD to color the structure by B-factor or occupancy.

To do this we modify the ampar_ca object and then write a PDB from it as follows:

```
In [9]: ampar_ca.setBetas(effectiveness)

In [10]: writePDB('3kg2_ca_effectiveness.pdb', ampar_ca)
Out[10]: '3kg2_ca_effectiveness.pdb'
```

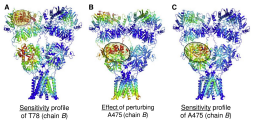
We can also calculate the PRS matrix and profiles separately from showPerturbResponse and slice out individual rows or columns and write them into PDB files for visualization. We slice rows using *axis=0* to obtain the effectiveness profile

```
In [11]: B_84_effectiveness = sliceAtomicData(prs_mat, atoms=ampar_ca, axis=0,
   ....:                                       select='chain B and resnum 84')
   ....:

In [12]: writePDB('3kg2_ca_B_84_effectiveness.pdb', ampar_ca,
   ....:          beta=B_84_effectiveness[0])
   ....:
Out[12]: '3kg2_ca_B_84_effectiveness.pdb'
```

and slice columns using *axis=1* to obtain the sensitivity profile

```
In [13]: B_84_sensitivity = sliceAtomicData(prs_mat, atoms=ampar_ca, axis=1,
   ....:                                    select='chain B and resnum 84')
   ....:

In [14]: writePDB('3kg2_ca_B_84_sensitivity.pdb', ampar_ca,
   ....:          beta=B_84_sensitivity)
   ....:
Out[14]: '3kg2_ca_B_84_sensitivity.pdb'
```

We generated our Figure 7 using this approach together with the *spectrum* command from PyMOL.



../template/acknowledgments.rst

[CA09] Atilgan C, Atilgan AR. Perturbation-response scanning reveals ligand entry-exit mechanisms of ferric binding protein. *PLoS Comput. Biol.* **2009** 5:e1000544.

[CA10] Atilgan C, Gerek ZN, Ozkan SB, Atilgan AR. Manipulation of conformational change in proteins by single-residue perturbations. *Biophys. J. 2010* 99(3):933-43

[IG14] General IJ, Liu Y, Blackburn ME, Mao W, Gierasch LM, Bahar I. ATPase subdomain IA is a mediator of interdomain allostery in Hsp70 molecular chaperones. *PLoS Comput. Biol.* **2014** 10:e1003624.

[AD15] Dutta A, Krieger J, Lee JY, Garcia-Nafria J, Greger IH, Bahar I. Cooperative Dynamics of Intact AMPA and NMDA Glutamate Receptors: Similarities and Subfamily-Specific Differences. *Structure* **2015** 23:1692-1704