**Yunhui Ge**

**2016.6**

**BICEPes users guide**

***Release 1.1***

Copyright 2014 Temple University

Contributors: Vincent Voelz, Yunhui Ge

Permission is hereby granted, free of charge, to any person obtaining a copy of this document (the “Document”), to deal in the Document without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Document, and to permit persons to whom the Document is furnished to do so, subject to the following conditions:

This copyright and permission notice shall be included in all copies or substantial portions of the Document.

**THE DOCUMENT IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS, CONTRIBUTORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE DOCUMENT OR THE USE OR OTHER DEALINGS IN THE DOCUMENT.**

**INTRODUCTION**

Bayesian inference of conformational populations (BICePs) uses Bayesian inference to optimally combine computational predictions and experimental measurements to estimate conformational state populations. Features of BICePs include:

* Correct use of reference priors
* MCMC sampling of the posterior distribution of states, along with nuisance parameters
* Estimation of Bayes Scores for model selection

Currently there is support for the following experimental observables:

1. NOE distances
2. J-coupling constants
3. Chemical shifts

**Theory**

BICePs uses a Bayesian approach to infer a distribution of conformational populations, by sampling a posterior probability function

P(X|D) P(D|X)P(X)

where X is one of a set of molecular conformations, and D

represents the experimental data.

P(X) is a prior probability function, estimated from computational methods as the predicted equilibrium populations. P(D|X) is a likelihood function representing experimental restraints. We assume normally-distributed errors in the experimental measurements, parameterized by a standard deviation σ

P(D|X, σ) = exp(-[rj(X) - rjexp]2/2 σ2)

Since σ is undetermined, we treat it as a nuisance parameter, including it in the joint posterior of possible X and σ.

**Reference potentials**

To correctly use the likelihood model as experimental restraints, a proper reference potential

Pref(r(X)) must be used for the experimental observables r(X)

P(X|D) [] P(X)

The bracketed term is a potential of mean force (PMF) along the reaction coordinates defined by r(X). The reference potential is needed to properly normalize the observed PMF. For distances rj, we conservatively model the reference potentials Pref(rj(X)|βj) as exponential distributions whose first moments βj are set to the estimated mean distance across all conformational clusters.

**MCMC sampling**

Markov Chain Monte Carlo is used to sample from the posterior distribution of conformations X as well as nuisance parameter(s) σ (and, for distances, scaling parameters γ′, see our Voelz JCC 2014 paper).

**Estimating BICePs Score**

A major advantage of BICePs is the quantitative estimatation of the BICePs Score, KBS, defined as

KBS = =

The BICePs Score represents the strength of evidence in support of model M1 over M2, and is analogous to the likelihood ratio test used in classical hypothesis testing. Each integral is simply the expectation value of the posterior probability

P(σ,X|D) for a given model over all values of σ,X. Thus, if we consider joint energy functions E1 =−lnP(σ,X|D,M1) and E2 =−lnP(σ,X|D,M2), the BICePs Score is given by

ln KBS = ln

To efficiently compute KBS, we use the Multistate Bennett Acceptance Ratio (MBAR) method of Shirts and Chodera to estimate gk = −ln⟨exp(−Ek)⟩ for any number of models Mk, k=1,..,K. The MBAR estimate gk is obtained by solving the self-consistent set of equations

gk = -ln

where xjn is the nth sample of (X,σ) from model Mk. We commonly want to compute KBF in the case where M2 is a "null distribution" with no information from computation (i.e. a uniform P(X)), and M1 is the full posterior model. In this case, we sample multiple models Mk using priors Pk(X)∝P(X)λk, for 0≤λk≤1.

**Online Resources**

You can find more documentation and other material at our github: https://github.com/vvoelz/nmr-biceps

**Citation**

The primary citation for BICePs is

Bayesian inference of conformational state populations from computational models and sparse experimental observables Vincent A. Voelz and Guangfeng Zhou. Journal of Computational Chemistry, 35(30):2215–2224 (2014)

**INSTALLING BICePs**

BICePs source code package is available on: <https://github.com/vvoelz/nmr-biceps> and the latest version is 1.1.

Other dependent software package:

Numpy ---> 1.10 or later

MSMbuilder ---> 3.0.0 or later

Yaml ---> 0.1.6 or later

Mdtraj ---> 1.5.1 or later

Pymbar ---> 2.1.0 or later

The older version package may work but not be guaranteed.

**Tutorial**

**PDB file preparation**

Let’s begin with an example of an BICePs script. First step is to prepare file formats which could be read and loaded by BICePs. In most cases you should have pdb files as your input files. It’s better to do a pre-operated step to clean up everything else and get a pdb file that only incudes ‘ATOM’ lines. To do this, just copy ‘Cleanpdb.py’ from /src to your work directory and use python to run it and you will be asked about your original input pdb file name:

Please tell me your pdb filename:

Just type the file’s name and it will return a new pdb file named ‘clean.pdb’. Also you can change its name to anything you want.

**Atom indices file preparation**

Then, the before you can work with BICePs you still need one step to generate atom index file. For example, suppose you have some NOE experimental data and in BICePs you will use atom pair distances to compare to your NOE data. So you need atom indices to specify which atom pairs you want to calculate interproton distances. Similar, copy make\_ind\_NOE.py to your working directory and run it with python command:

python make\_ind\_NOE.py

Normally, the pdb file will be the one we just ‘cleaned’ which is ‘clean.pdb’. If it is not, don’t forget to change the file name either in the script or the pdb file. (The default name in the script is ‘clean.pdb’)

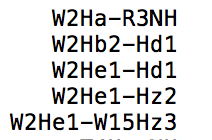
Then you will be asked specify the residue number and atom name like this:

Residue number (1-based index):

Atom name (capital form):

Pay attention the residue number should be 1-based (not 0-based!)

For example, you have the experimental NOE data of following atom pairs:



So for the first pair, type ‘2’ as residue number and ‘HA’ as atom name then repeat it by typing ‘3’ and ‘H’ to finish the first pair. After you finish typing each pair and the script will ask you how many pairs of your input:

How many atom pairs of your input?

Make sure you answer it with atom pairs but not atom numbers! If your input is wrong it will return:

Something is wrong about your input, please double check

I strongly suggest you finish this process carefully and it will save your time.

For double check, the script will check if the number of your input atoms is an even number (remember two atoms for one atom pair) and will tell you if there is a mistake (normally it will be a missing atom or more):

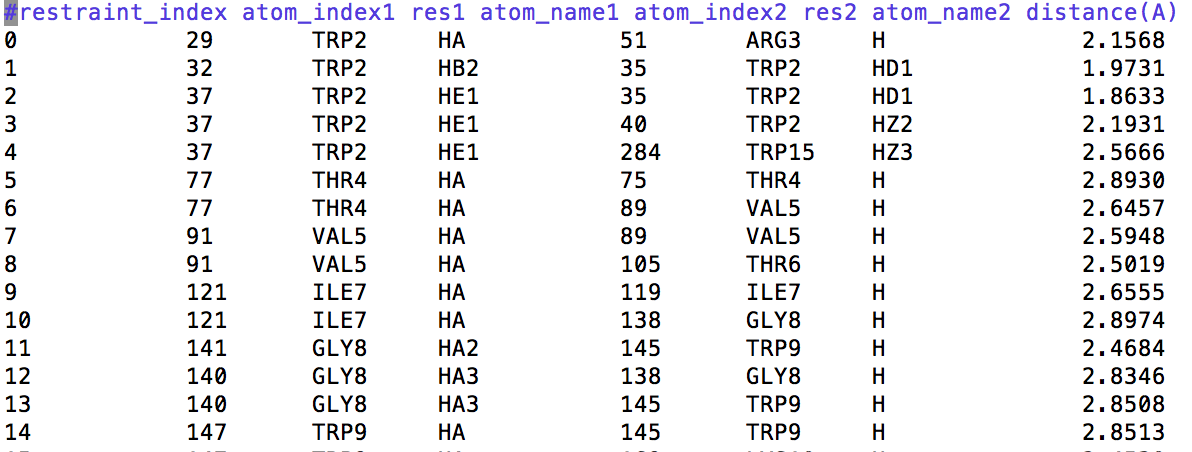
Something is wrong about your input, please double check

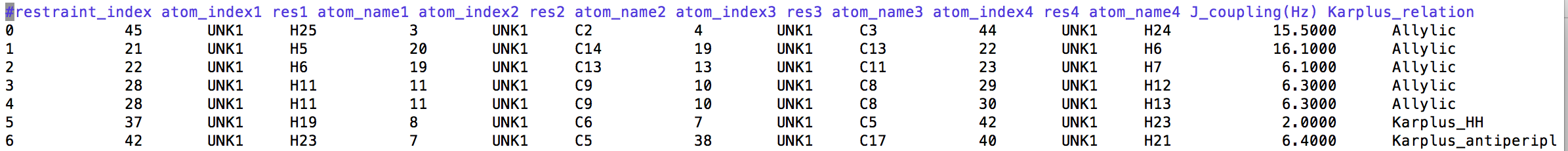
If everything is correct, you will get a new file named ‘Ind.npy’ which includes each atom pair indices (0-index based!) for you use in the next steps. Similarly, you will find ‘make\_ind\_CS.py’ and ‘make\_ind\_J.py’ in src/ folder for atom indices preparation for chemical shifts and J-couplings.

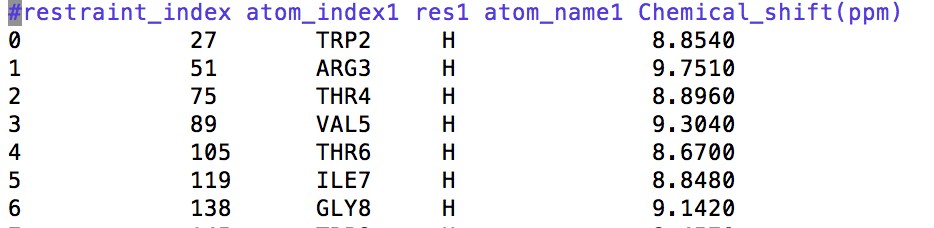
**Experiment restraints files preparation**

Next step, you could use ‘make\_NOE\_restraint\_file.py’ to make NOE experiment restraints file for BICePs. Similarly, ‘make\_J\_restraint\_file.py’ and ‘make\_cs\_restraint\_file.py’ are used to generate J-couplings and chemical shift experiment restraints files. These restraints files will be named as ‘your-sample-name.noe/ your-sample-name.Jcoupling/ your-sample-name.cs’. You can define the file name in scrips mentioned above.

An example of NOE, J-couplings and chemical shifts restraints files:





****

**Pre-computed data from simulation**

The latest version BICePs supports NOE, J-couplings and chemical shifts. So far, we finished experiment restraints files preparation and next step will be pre-computed these experimental data based on your simulation.

**NOE data**

We converge NOE peak intensity to atom pair distance and then we can compute atom pair distance by using mdtraj to compare the computational simulation with experimental data. Before that, to obtain comprehensive and diverse conformation of your simulation we suggest to use MSMbuilder 3.0 to cluster your trajectory into multiple microstates. For example, if you have a gro file and a trajectory from your simulation then you can use ‘cluster.py’ in tools/ folder to finish your cluster. You can set the number of states and which methods (KCenters, KMeans, etc) you prefer to do your cluster. To count how many frames does each state have, you can use ‘state.py’ in tools/ folder and then use ‘population.py’ in the same folder. Now you will have a population file name as ‘population.txt’. To get reduced free energy for each state, use ‘energy.py’ to generate ‘energy.txt’ file.

Use ‘distance.py’ to compute atom pair distances for each clustering states. This script will generate average ⟨r−6⟩−1/6 values (You can read more in paper about why we need this).

**Chemical shift**

Will be updated later…

**Running BICePs**

So far, we finished preparation job for running BICePs scripts. Below is a checklist for you to check if you get everything you need to run BICePs scripts:

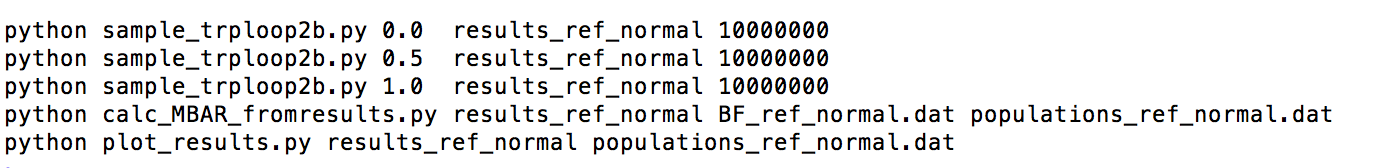
* energy.txt – reduced free energy for each clustered states
* your-sample-name.noe/.Jcoupling/.chemicalshift – experiment restraints files
* Gens – clustered states folder (pdb files from clustering)
* NOE – pre-computed atom pair distances for each state (⟨r−6⟩−1/6 values)
* Chemical\_shift – pre-computed chemical shift for each state

If you get everything on this list, then you are ready to run BICePs scripts. You also need ‘calc\_MBAR\_fromresults.py’ and ‘ploy\_results.py’ in your work directory. You can find a ‘runme’ file in examples/ folder as well:

Main BICePs script

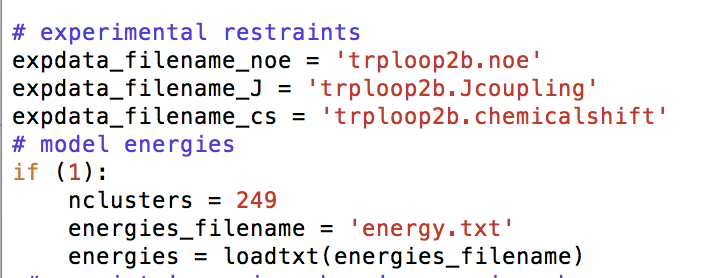
Monte Carlo running steps

values



You can set values and MC running steps by yourself. Basically, the more steps you set, the longer the script will run. In some cases, to get better converge, a big number of steps are necessary.

Now let’s have a look at the main BICePs script ‘sample\_xxx.py’:



You need change these file names as your file names. Also, set number of cluster states as your option.

Then you can run BICePs by ‘. runme’ and wait for the scripts finished.