Is the force with us?

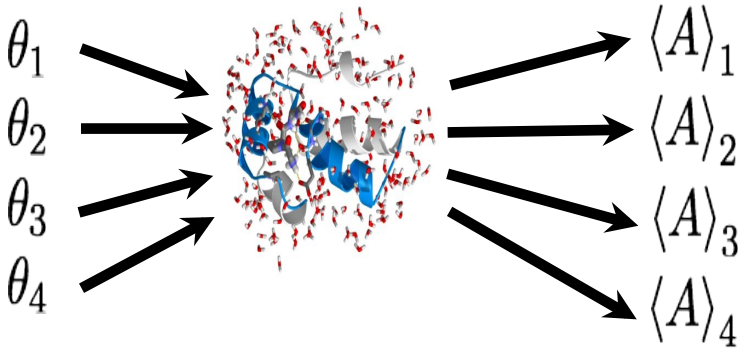
Bayesian inference in computational chemistry.

Chaya D. Stern
PyData NYC
Nov 29, 2017

https://github.com/ChayaSt/pydatanyc_cc

- **How are computational molecular models utilized in drug discovery?**

- **Bayesian inference of model parameters and model selection.**



- **Reweighting for error propagations.**

Biological Macromolecules are the molecular machines of life.

Structural

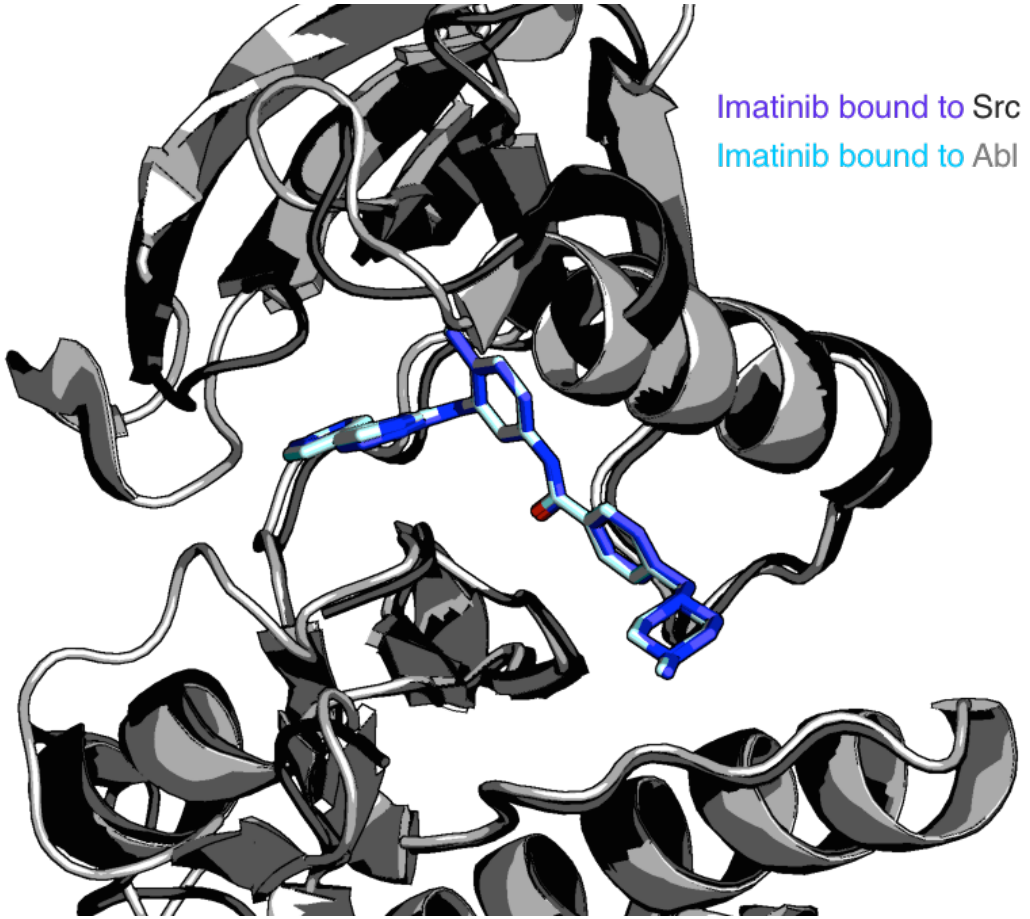
Signaling

Transport

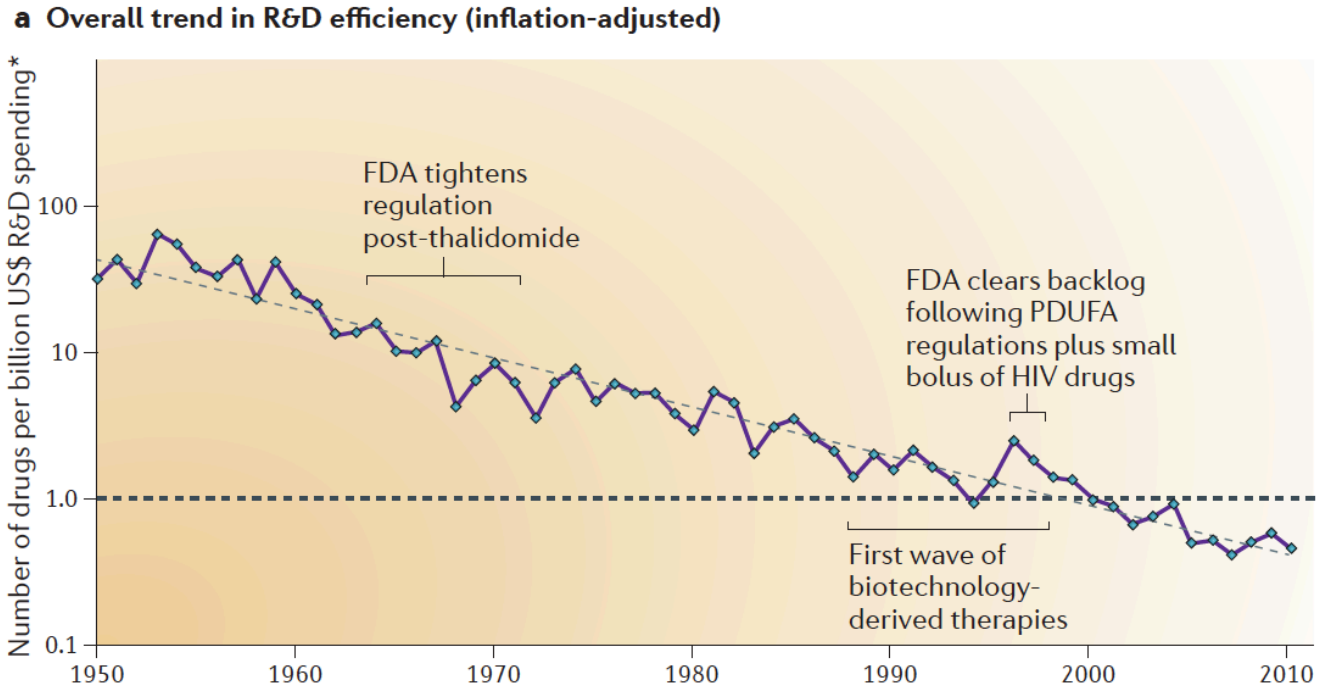
Catalysts

Regulation

Biosynthetic factories

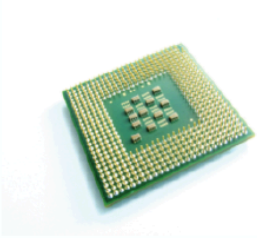


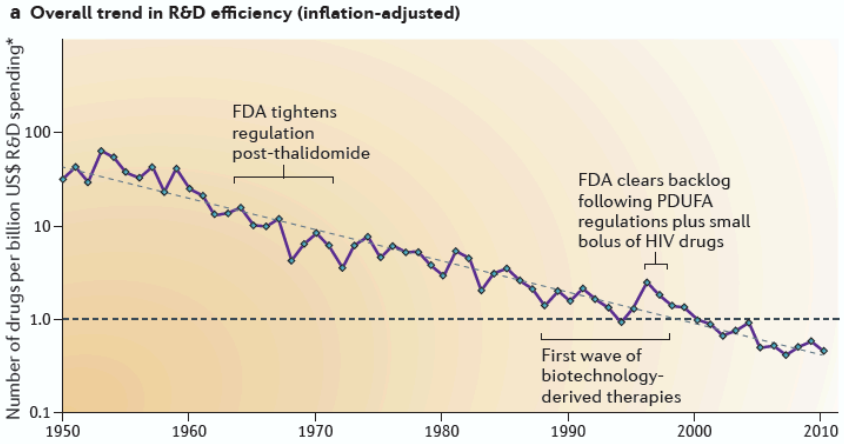
Drug discovery usually ends in failure

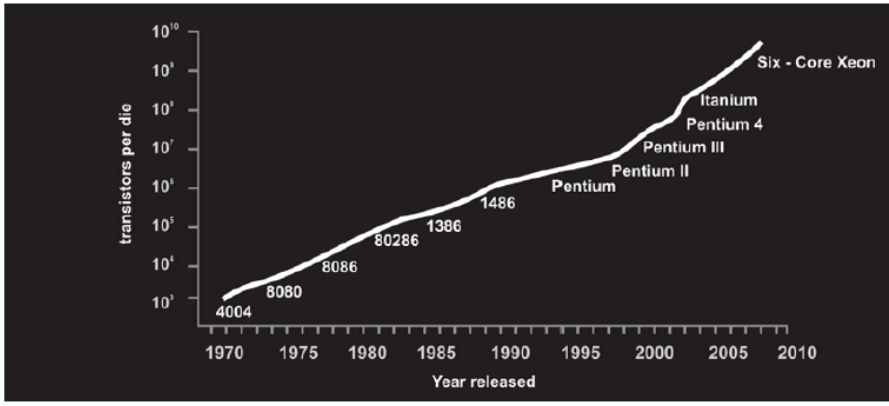


Scannell et al. Nature Rev Drug Disc 11:191, 2012

Drug discovery usually ends in failure



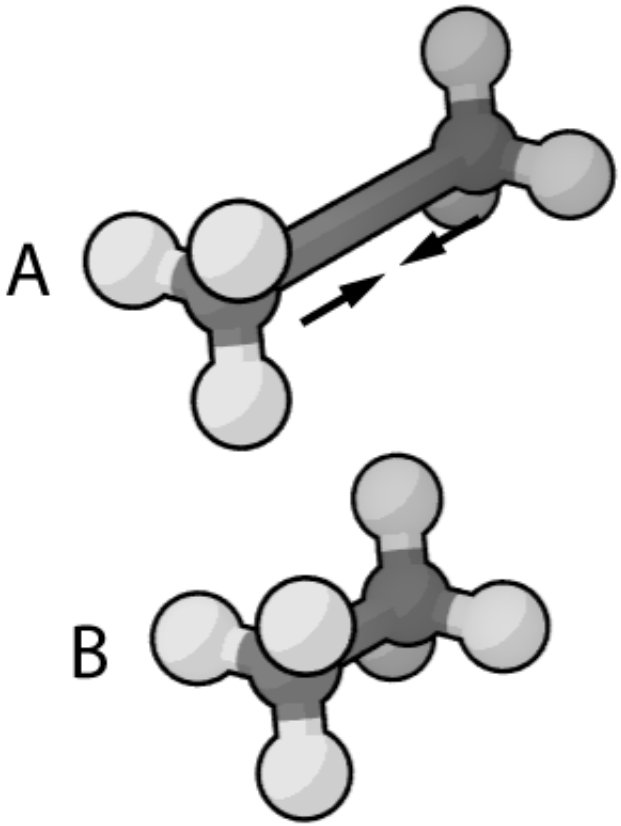




EROOM'S LAW **MOORE'S LAW**

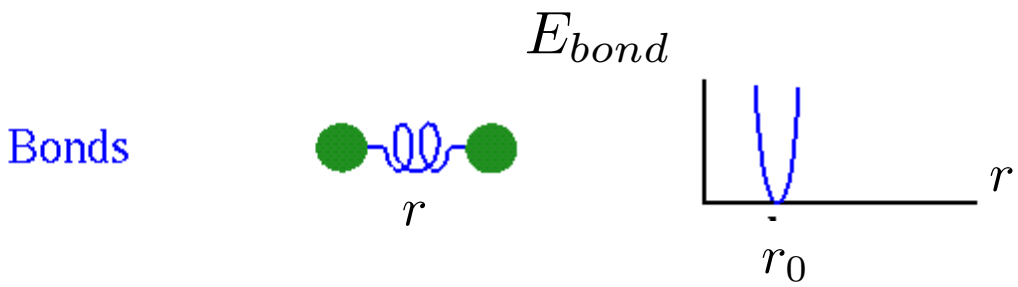
In molecular mechanics simulations, inter and intramolecular forces are modeled with a forcefield.

A Newtonian physics approximation of the underlying quantum mechanics.



$$E_{bond} = \sum_{bond} k_b (r - r_0)^2$$

k_b : Force constant
r_0 : equilibrium length

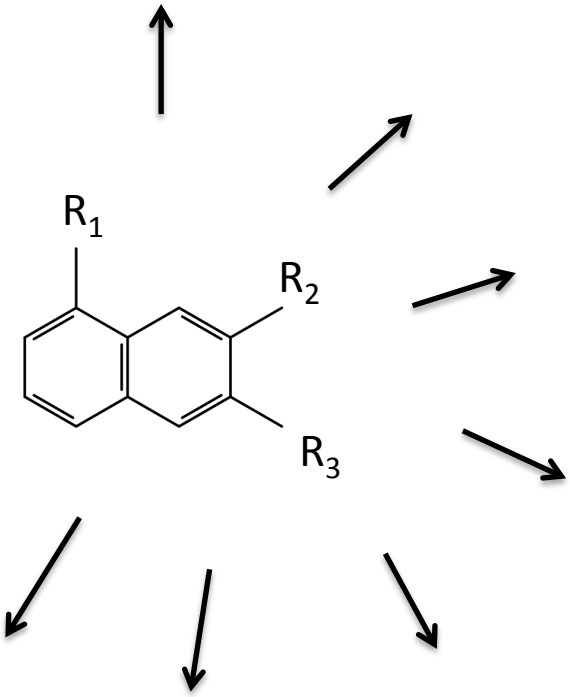


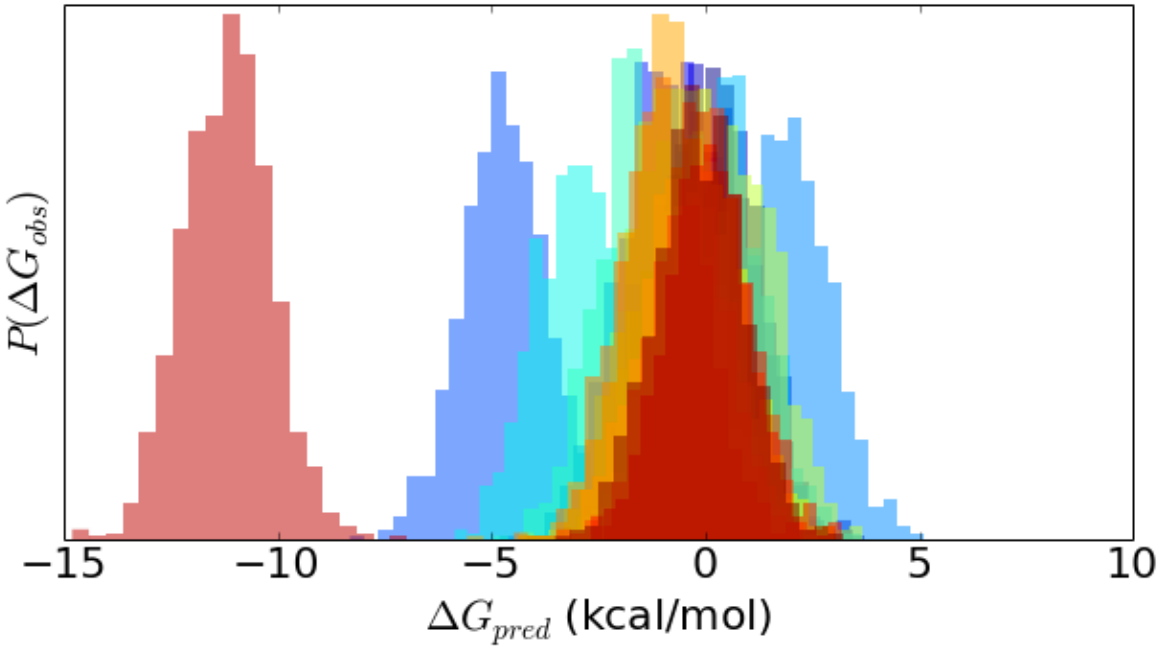
Free parameters need to be fit to reproduce experimental condensed phase data.

Molecular dynamics simulations provide atomic detail to dynamics that is usually difficult to observe.

To design a new drug, we need to predict which small molecule will bind to the target.

Uncertainty in computed properties are needed for designing the next generation of small-molecule drugs



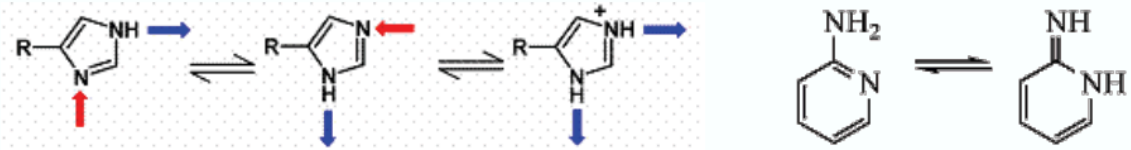


Three main sources of error.

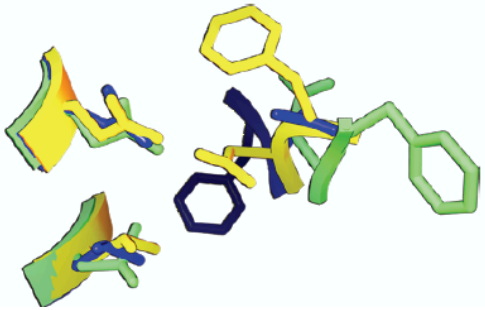
1. The forcefield does a poor job of modeling the physics of our system

$$V(\mathbf{q}) = \sum_{\text{bonds}} K_r(r - r_{eq})^2 + \sum_{\text{angles}} K_\theta(\theta - \theta_{eq})^2 + \sum_{\text{dihedrals}} \frac{V_n}{2}[1 + \cos(n\phi - \gamma)] + \sum_{i<j} \left[\frac{A_{ij}}{R_{ij}^{12}} - \frac{B_{ij}}{R_{ij}^{6}} + \frac{q_i q_j}{\epsilon R_{ij}} \right]$$

2. We're missing some essential chemical in our simulations
(e.g. protonation states, tautomers, covalent association)



3. We haven't sampled all of the relevant conformations



We expect binding free energies to be sensitive to forcefield error.

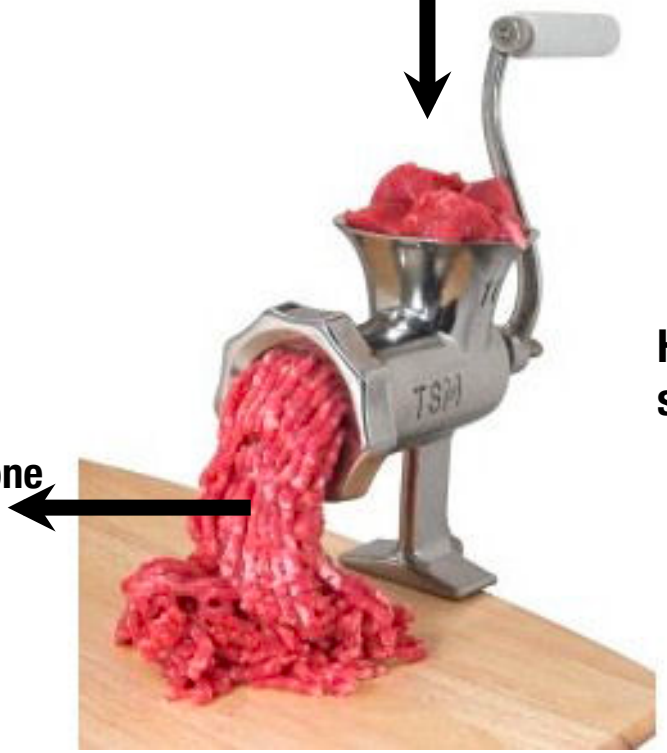
1. The forcefield does a poor job of modeling the physics of our system

$$V(\mathbf{q}) = \sum_{\text{bonds}} K_r(r - r_{eq})^2 + \sum_{\text{angles}} K_\theta(\theta - \theta_{eq})^2 + \sum_{\text{dihedrals}} \frac{V_n}{2}[1 + \cos(n\phi - \gamma)] + \sum_{i<j} \left[\frac{A_{ij}}{R_{ij}^{12}} - \frac{B_{ij}}{R_{ij}^{6}} + \frac{q_i q_j}{\epsilon R_{ij}} \right]$$

Currently we don't know how much uncertainty the forcefield contributes to simulation results.

How are forcefields made?

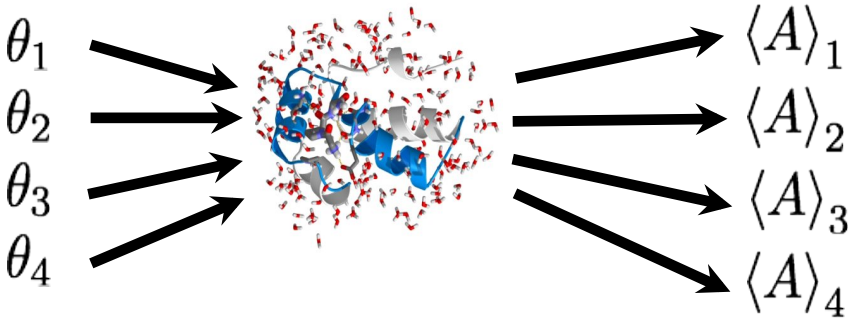
experimental data
quantum chemistry
keen chemical intuition



Heroic effort by graduate students and postdocs

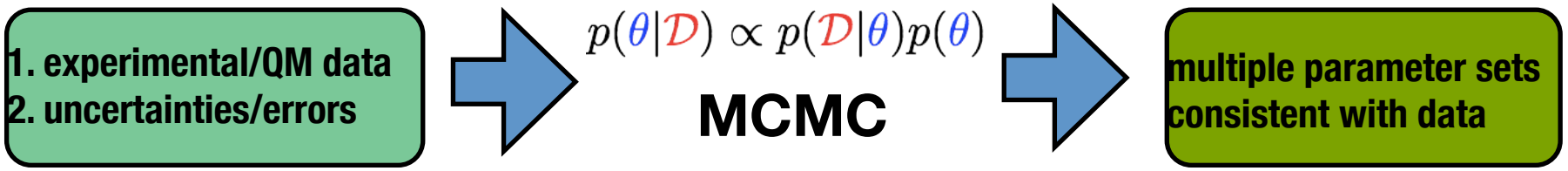
a parameter set we desperately hope someone actually uses

Can a Bayesian approach provide a better way to fit forcefields?



We cast the parameterization of force fields as a Bayesian inference problem.

Bayes rule provides a probability measure over unknown parameters given data:



$\mathcal{D}$ Data (QM and experimental)

θ Forcefield parameters

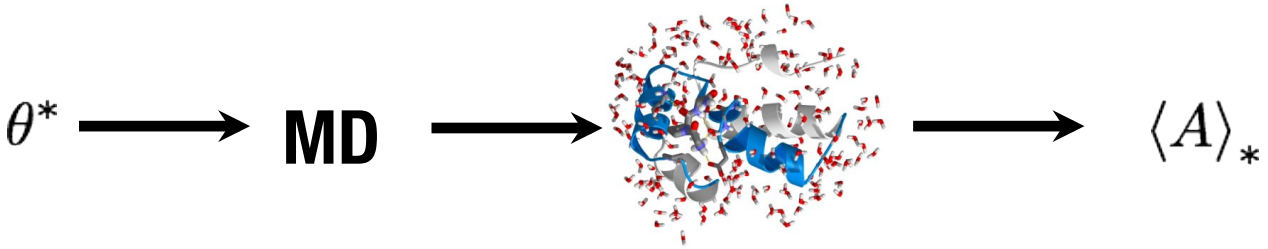
$p(\theta|\mathcal{D})$ posterior

$p(\mathcal{D}|\theta)$ error model for data

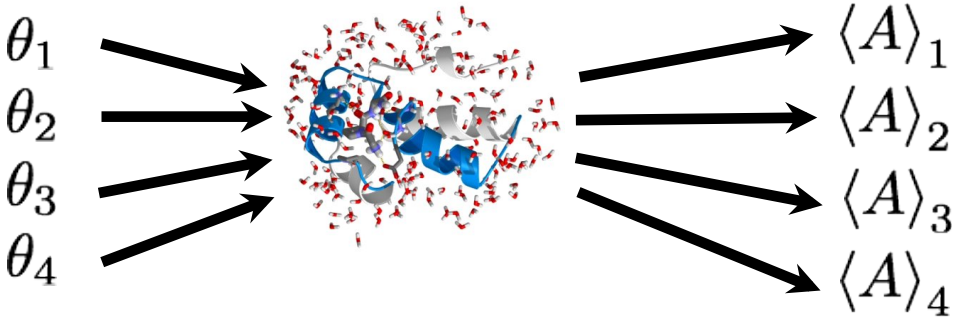
$p(\theta)$ prior on forcefield parameters

A Bayesian approach also provides a straightforward way to propagate uncertainty to computed properties.

Compute properties with one representative parameter set from collection

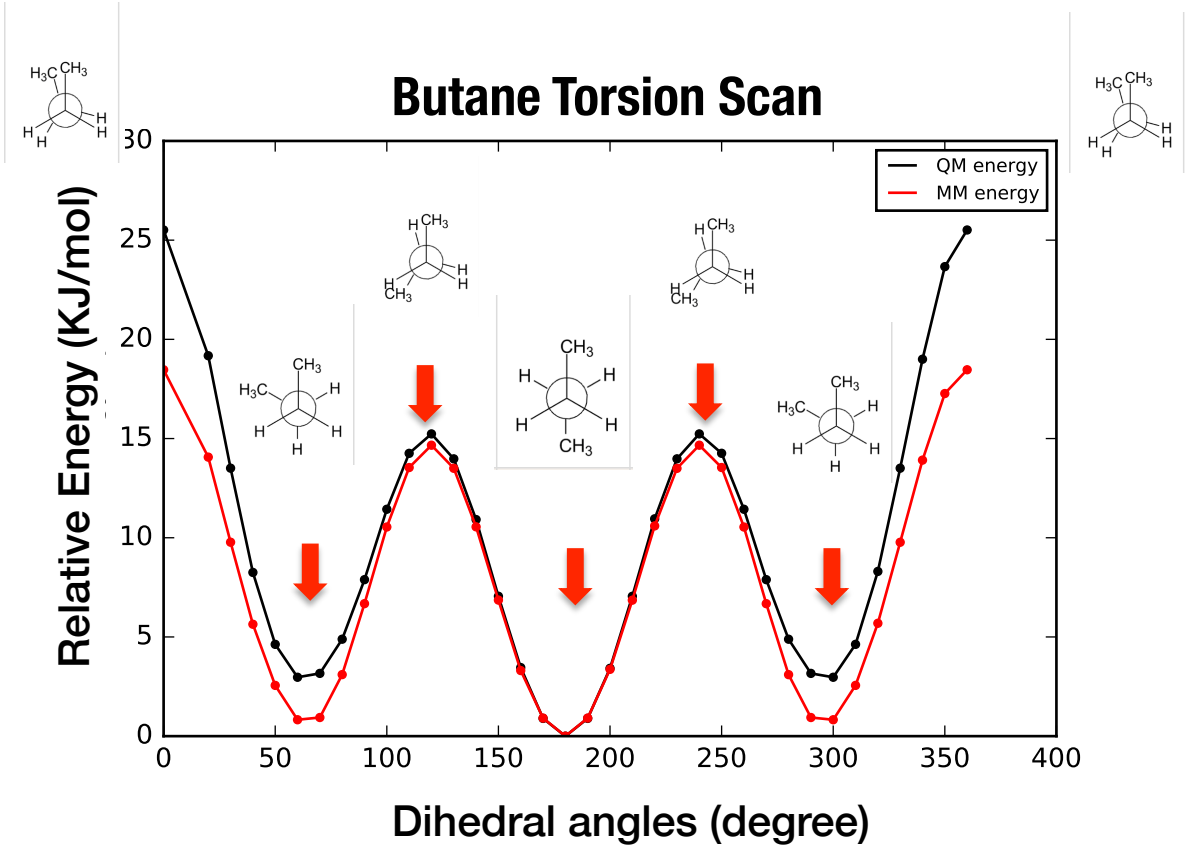


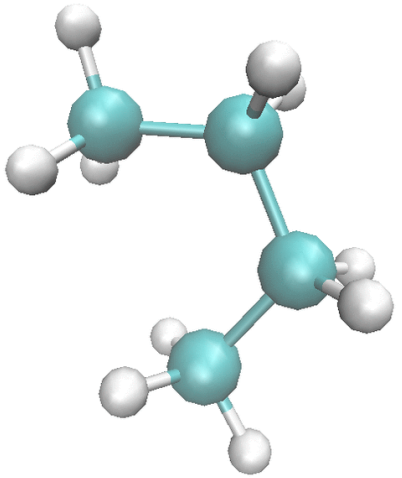
Estimate computed properties for other parameter sets from collection by reweighting (e.g. Zwanzig)



Can estimate both statistical and systematic components of computed property!

The dihedral angle energy is usually modeled with a Fourier series.





$$E_i^{dih} = \sum_m \sum_n = K_{m,n}\left(1 + cos(n\phi_i)\right)$$

The model is given by a Gaussian likelihood and uninformative priors.

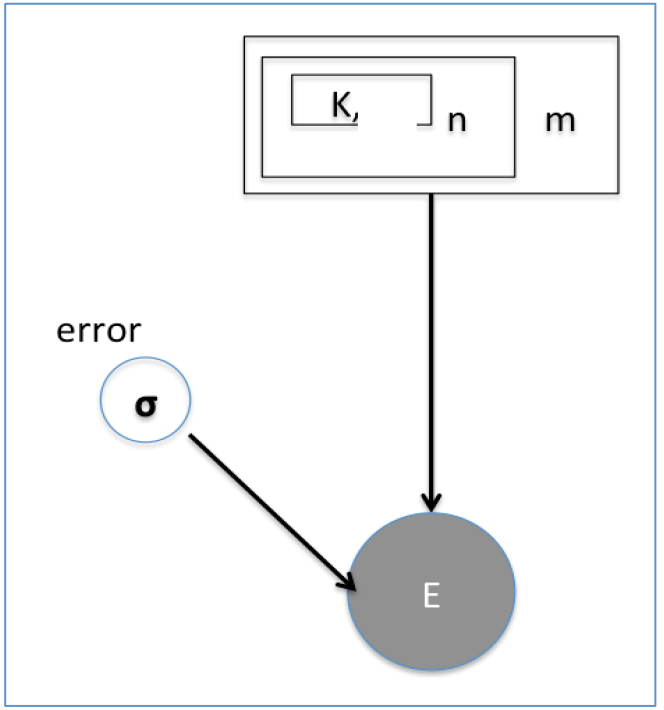
$$p(\theta|\mathcal{D}) \propto p(\mathcal{D}|\theta)p(\theta)$$

$$P(D|\Theta) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{\sum_i (E_i^{QM} - (E_i^{MM} + E_i^{dih}) + c)^2}{2\sigma^2}}$$

E_i^{QM} : QM energy for configuration i
E_i^{MM} : MM energy without dih for configuration i

$$E_i^{dih} = \sum_m \sum_n = K_{m,n}\left(1 + cos(n\phi_i)\right)$$

m: torsion type
n: periodicity
Φ_i: torsion angle for the i^{th} torsion.



Code (pre-alpha) is available on Github:
https://github.com/choderalab/torsionfit

The model is given by a Gaussian likelihood and uninformative priors.

```
class TorsionFitModel(object):
    """pymc model for sampling torsion parameters.
    This model provides the option to use reversible jump to sample over multiplicity terms

    Attributes:
    ----------
    pymc_parameters: dict() of pymc parameters
    fags: list of QMDatabasees for fragments to fit torsions to
    rj: bool. If True, model uses reversible jump to sample over multiplicity terms. If false, all terms are sampled.
    parameters_to_optimize: list of tuples (dihedrals to optimize)
    models: list of models to sample over.
    inner_sum: list of precalculated inner sum. This is also the gradient.

    """
```

Code (pre-alpha) is available on Github:
https://github.com/choderalab/torsionfit

The model is given by a Gaussian likelihood and uninformative priors.

Uniform prior on K (force constants):

```
for p in self.parameters_to_optimize:
    torsion_name = p[0] + '_' + p[1] + '_' + p[2] + '_' + p[3]

     for m in multiplicities:
         name = p[0] + '_' + p[1] + '_' + p[2] + '_' + p[3] + '_' + str(m) + '_K'
         k = pymc.Uniform(name, lower=0, upper=20, value=0)
         self.pymc_parameters[name] = k
```

Code (pre-alpha) is available on Github:
https://github.com/choderalab/torsionfit

The model is given by a Gaussian likelihood and uninformative priors.

$$E_i^{dih} = \sum_m \sum_n = K_{m,n}\left(1 + cos(n\phi_i)\right)$$

```
@pymc.deterministic
def torsion_energy(pymc_parameters=self.pymc_parameters):
    mm = np.ndarray(0)

    for i, mol in enumerate(self.frags):
        Fourier_sum = np.zeros((mol.n_frames))
        for t in inner_sum[i]:
            name = t[0] + '_' + t[1] + '_' + t[2] + '_' + t[3]
            K = pymc_parameters['{}_K'.format(name)]
            Fourier_sum += (K*inner_sum[i][t]).sum(1)
        Fourier_sum_rel = Fourier_sum - min(Fourier_sum)
        Fourier_sum_rel += pymc_parameters['{}_offset'.format(mol.topology._residues[0])]
        mm = np.append(mm, Fourier_sum)
    return mm
```

Code (pre-alpha) is available on Github:
https://github.com/choderalab/torsionfit

The model is given by a Gaussian likelihood and uninformative priors.

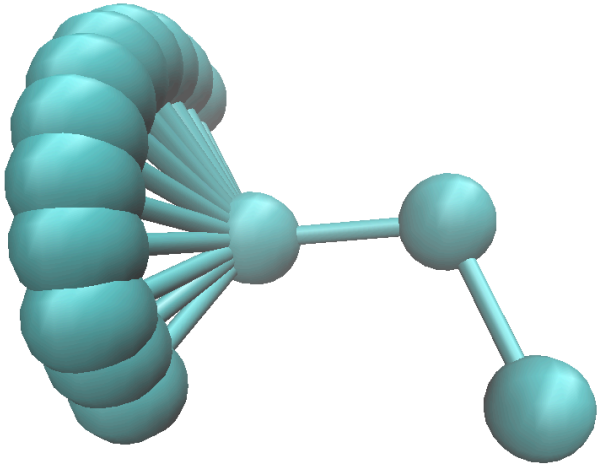
$$E_i^{dih} = \sum_m \sum_n = K_{m,n}\left(1 + cos(n\phi_i)\right)$$

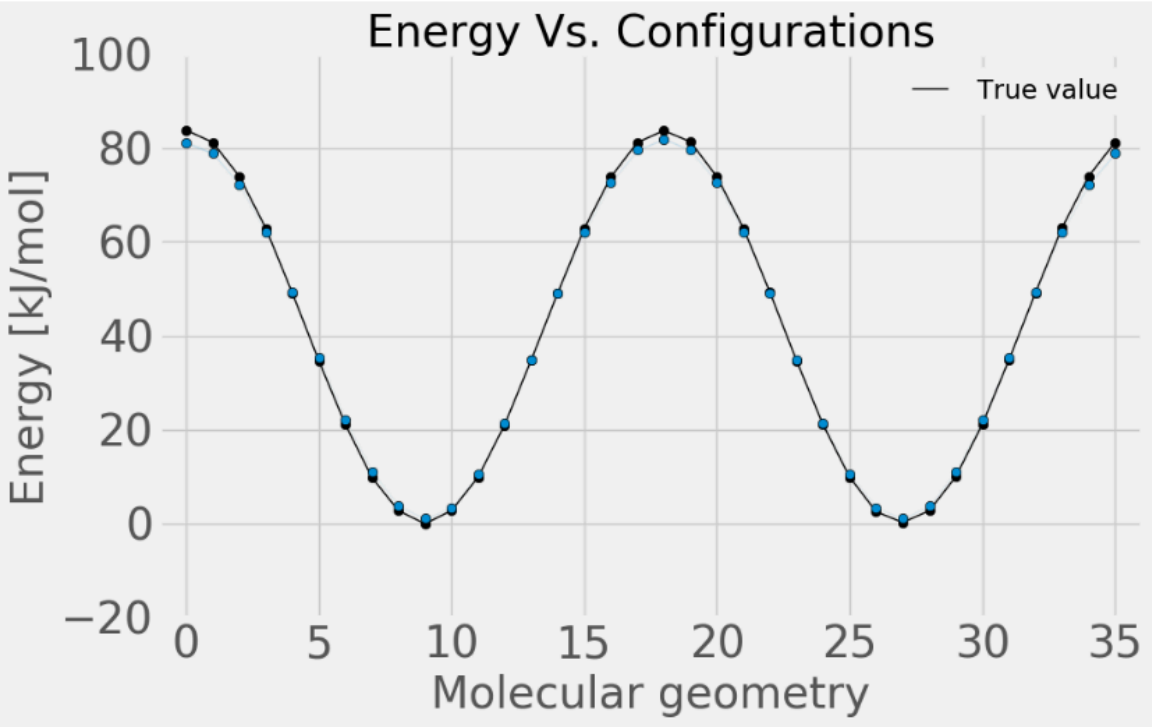
```
qm_energy = np.ndarray(0)
for i in range(len(frags)):
    qm_energy = np.append(qm_energy, frags[i].qm_energy)
self.pymc_parameters['mm_energy'] =  torsion_energy
self.pymc_parameters['qm_fit'] = pymc.Normal('qm_fit', mu=self.pymc_parameters['mm_energy'],
                                             tau=self.pymc_parameters['precision'], size=size, observed=True,
                                             value=qm_energy)
```

Code (pre-alpha) is available on Github:
https://github.com/choderalab/torsionfit

Toy model of 4 carbons

Jupyter notebook demo





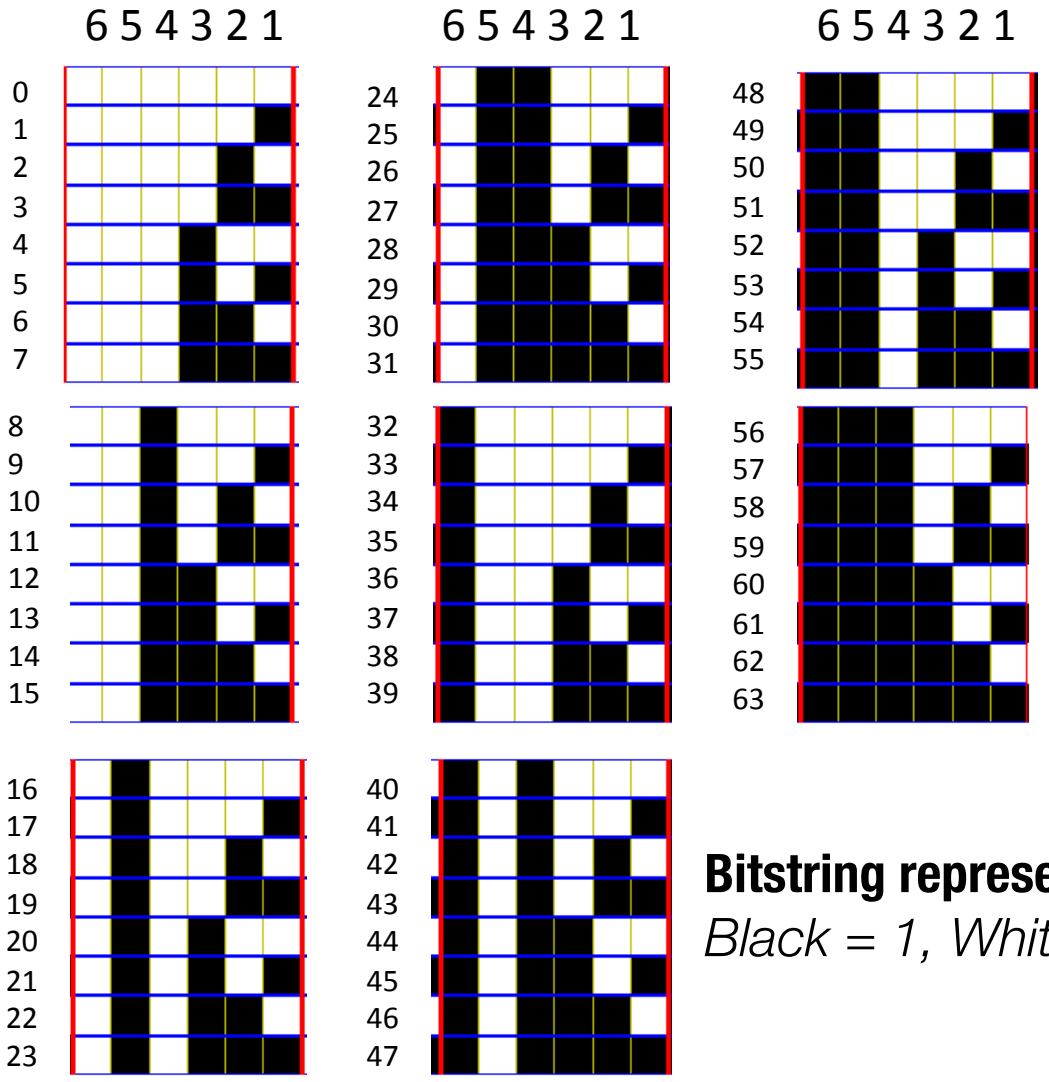
Model selection with Pymc

How many Fourier terms do we need to reproduce the PES?
Current tools rely on chemical intuition to choose periodicity terms.

$$2^6 = 64$$

For each torsion term!

Algorithm for sampling over models in Pymc



Bitstring representation of integers {0, 63}

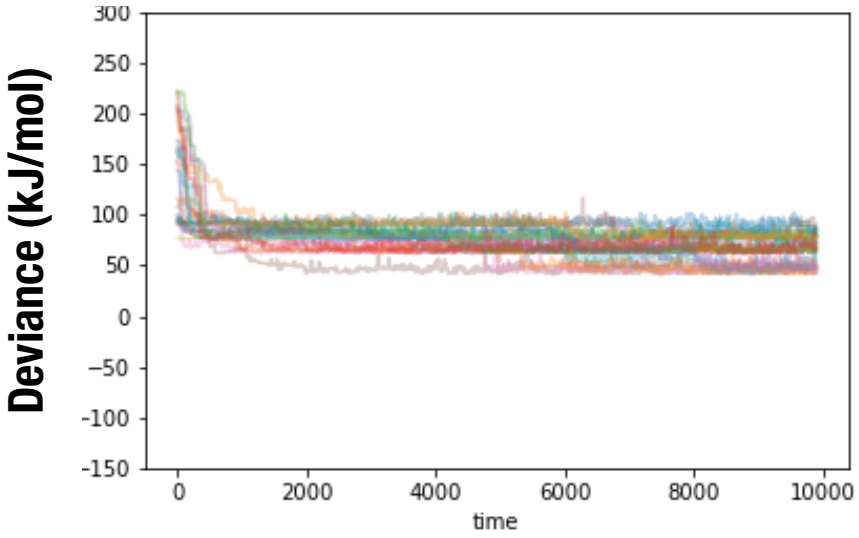
Black = 1, White = 0

```
bitstring = pymc.DiscreteUniform(name, lower=0, upper=63, value=multiplicity_bitstrings[torsion_name])
```

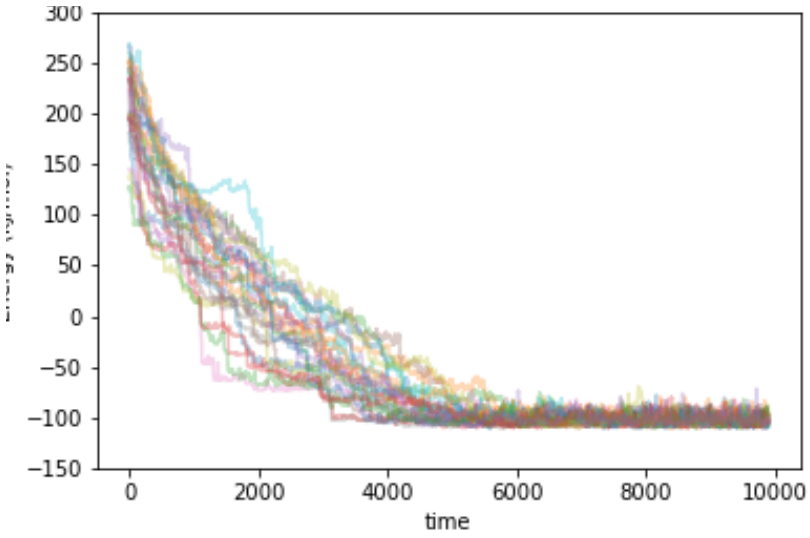
RJMC leads to poor mixing

$$-2\log[p(D|\theta)]$$

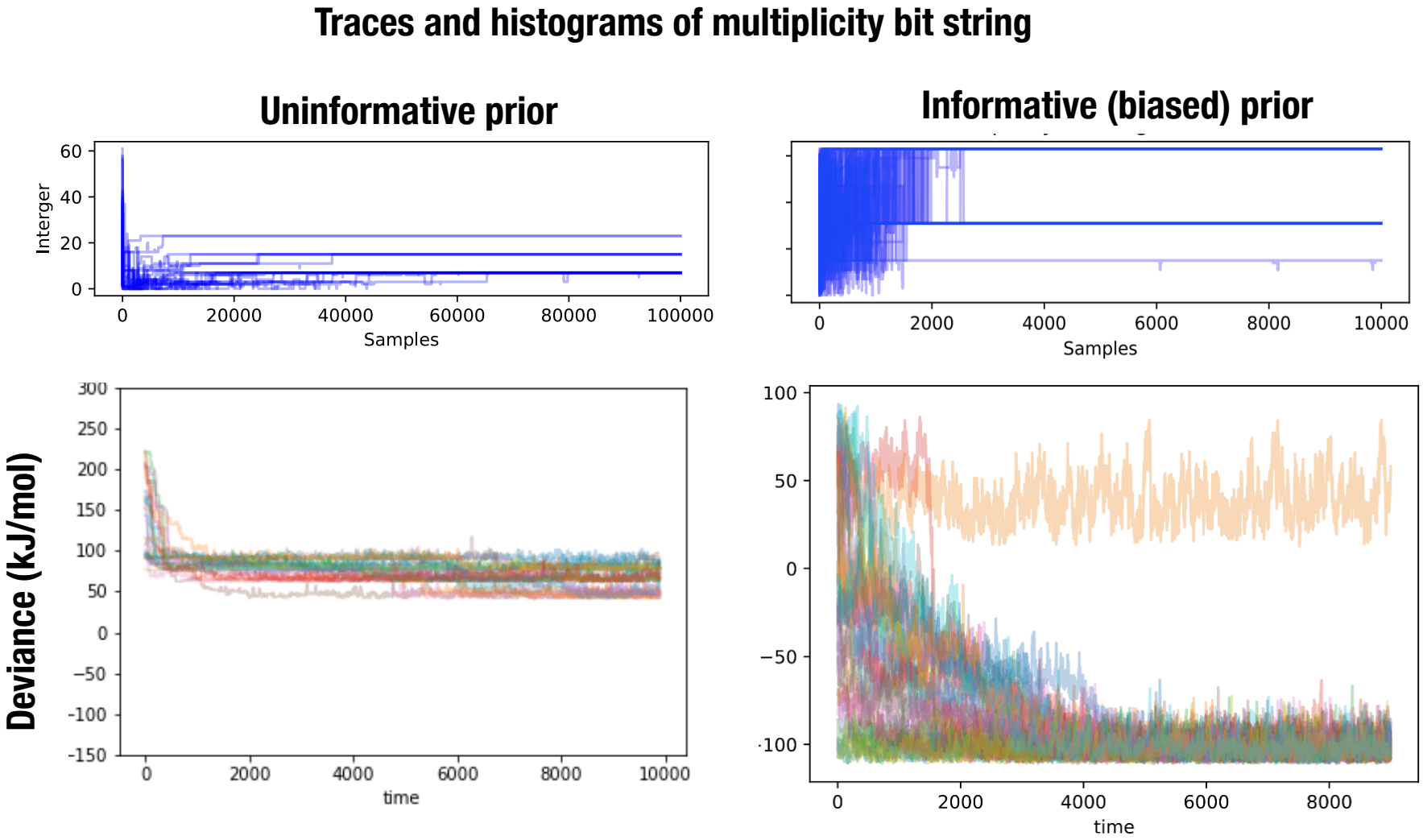
Reversible jump



Model 63 (includes all terms)

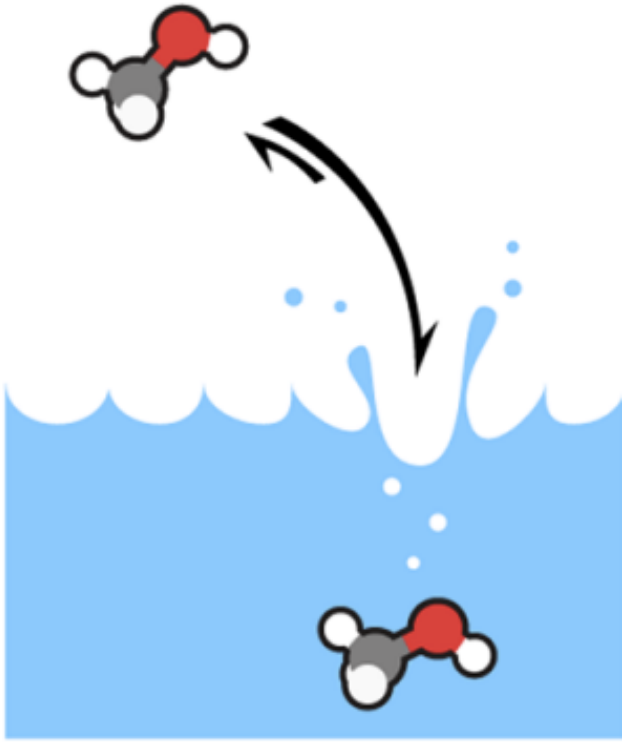


A biased prior was used to improve mixing between models.



Propagating the error to a computed property using the posterior.

Hydration Free Energy.

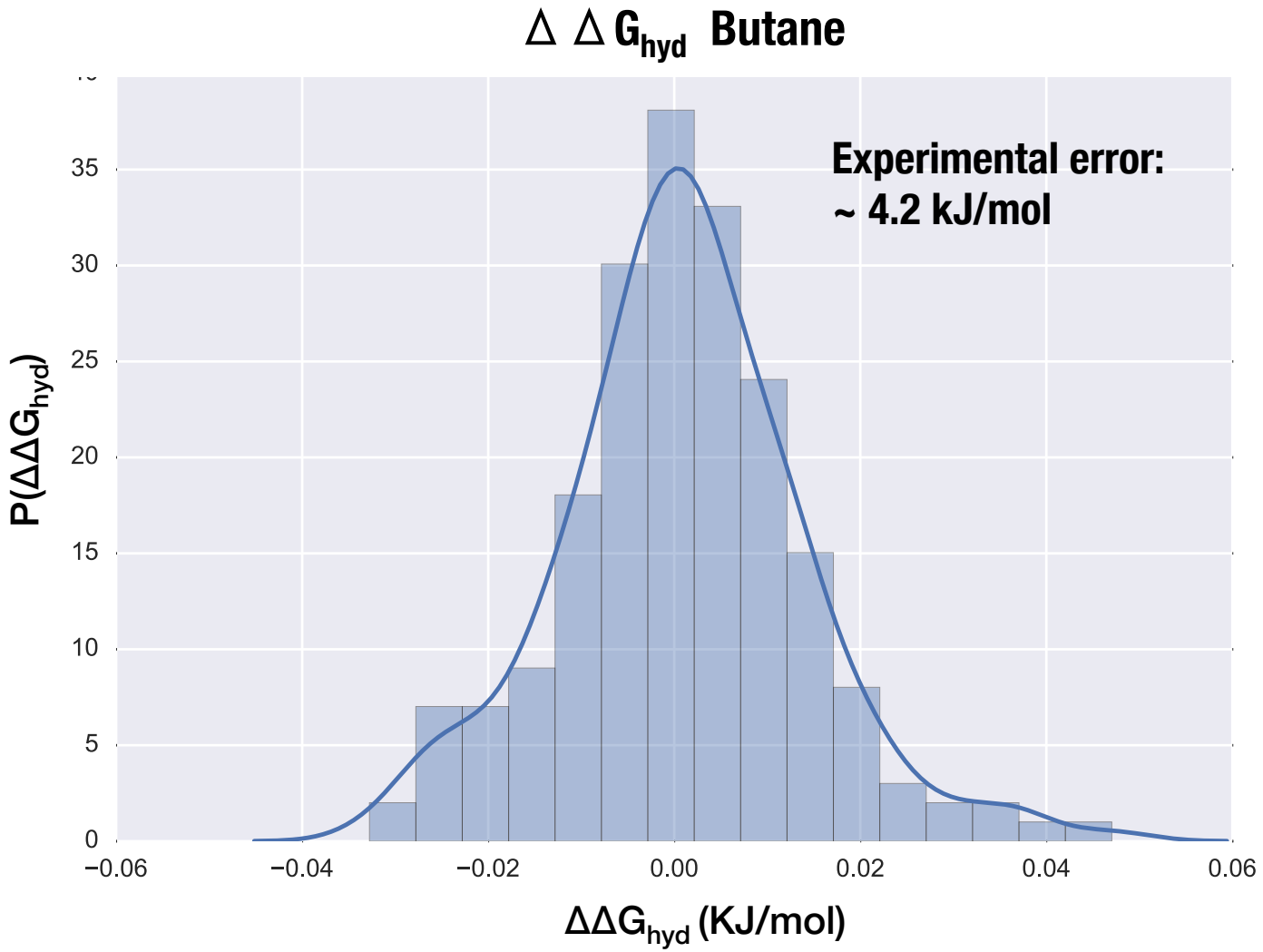


Fennell, et al. 2014.

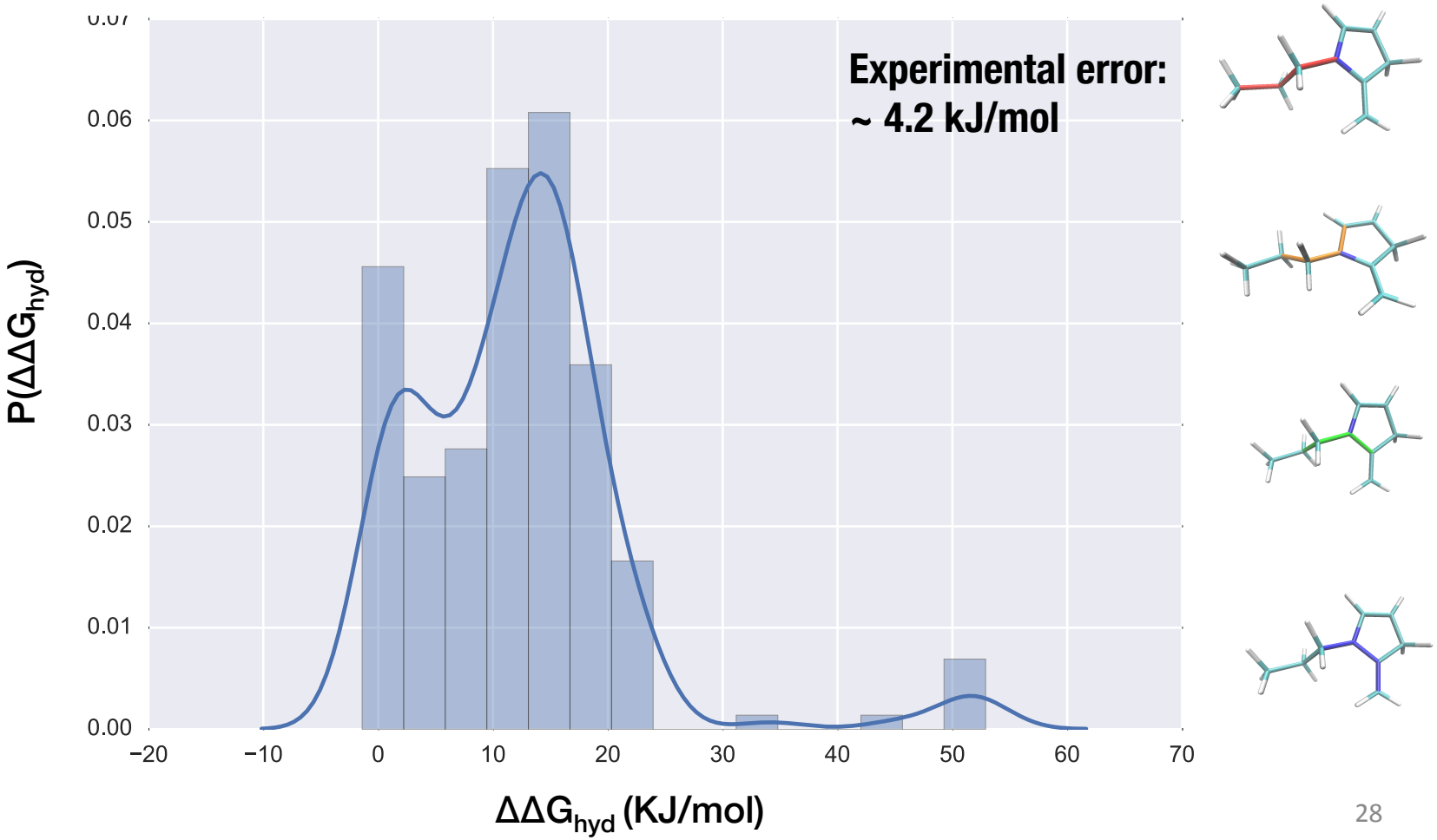
$$\Delta G_{hyd} = -k_B T ln \left[\frac{Z_s}{Z_v} \right]$$

k_B: Boltzmann constant
Z_s : Normalizing constant of molecule in solvent
Z_v: Normalizing constant of molecule in vacuum

For butane, error in $\Delta\Delta G$ is very small relative to experimental error.



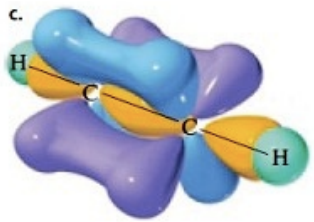
The error for more complex molecules is significant relative to experimental error.



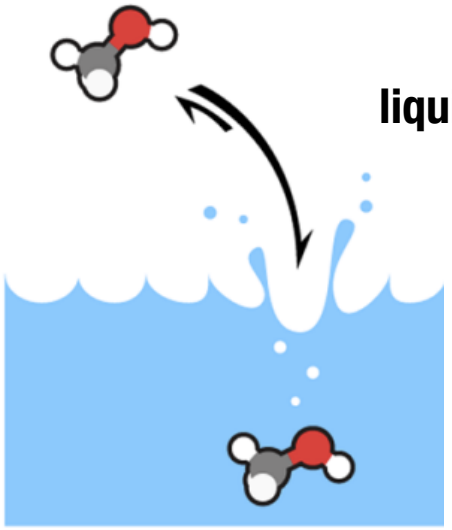
Why are the errors so large?

- Do we have enough data?
  - Is QM data enough?
  - Are we using the right kind of data?
  - Is the functional form adequate (model misspecification) ?

Gas phase



liquid phase



Acknowledgment

John Chodera
Gregory Ross

Funding

Tri-Institutional PhD Program
Chemical Biology

NSF