

Story time and puppet show with Mr. Bayes

Data, Models, and Decisions

John D. Chodera

Computational Biology Program, Memorial Sloan-Kettering Cancer Center

<http://www.choderalab.org>



Tweet your questions to @jchodera



CUP XIV - Wed 5 Mar - via The Interwebs



Ant: “This **is the 21st Century”**
We can have you attend by telepresence robot...

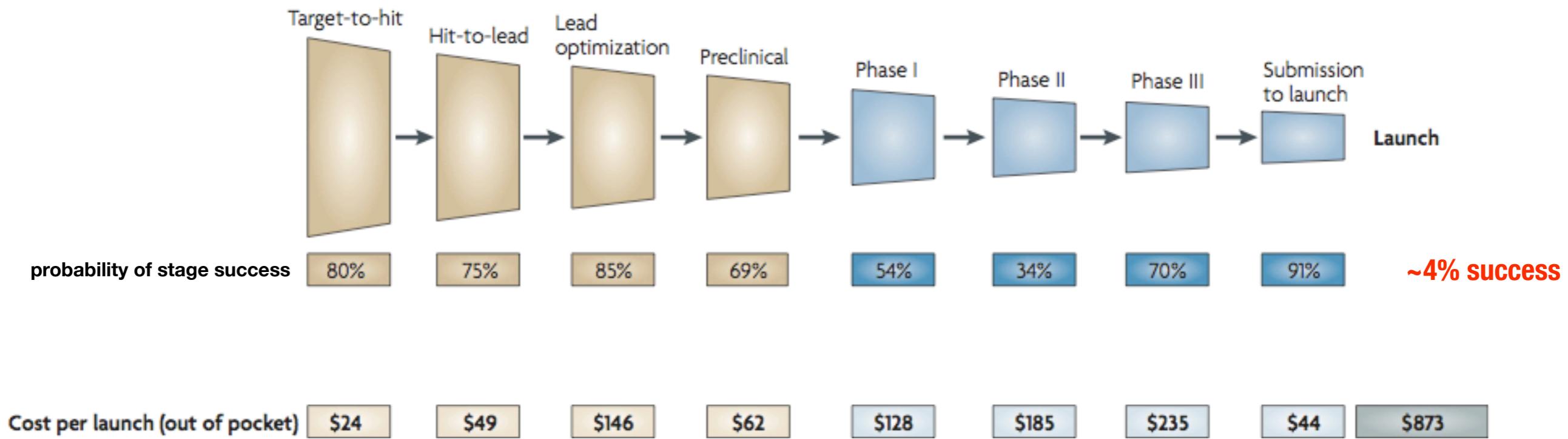


We are all here because we desperately hope modeling can improve the dismal success rates in drug discovery

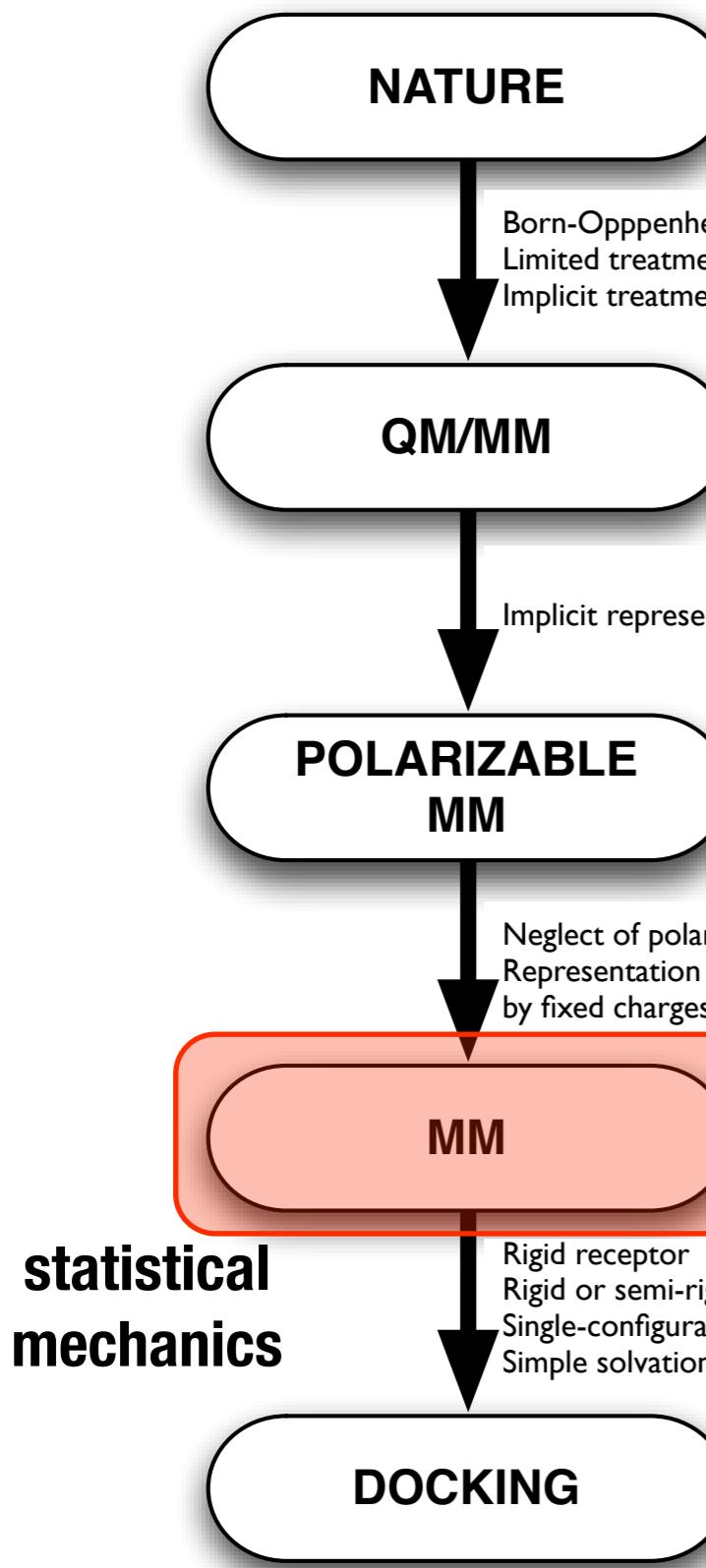
Total pharma research spending has **doubled** to \$65.3B in last decade

Number of new molecular entities approved by FDA 2005-2009 is **half** that from previous five years

Number of truly innovative new molecular entities has **remained constant** at 5-6/year



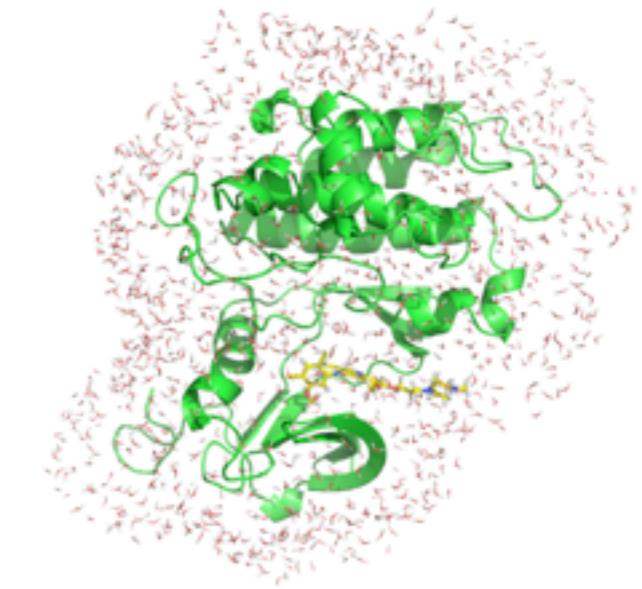
What details are crucial for useful accuracy in rational design? Our lab is trying to find out!



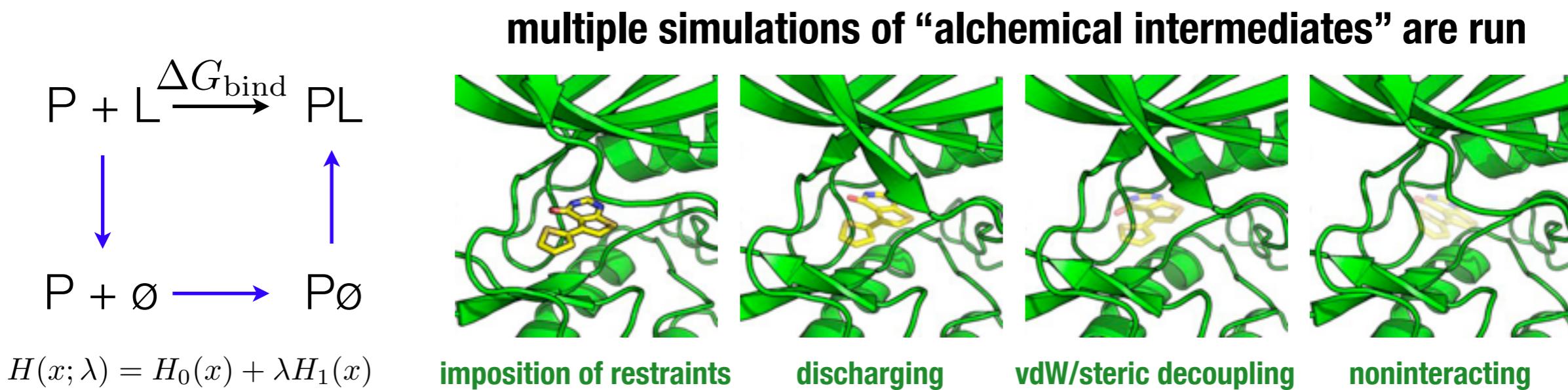
molecular mechanics potential energy function (e.g. AMBER)

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

if accurate enough, systematically remove detail



Alchemical free energy calculations provide a rigorous way to efficiently compute binding affinities for a given forcefield



Requires **orders of magnitude** less effort than simulating direct association process, but includes all enthalpic/entropic contributions to binding free energy.

Not be as fast as OpenEye tools, but will eventually give true binding free energy for a given forcefield (be it bad, or really really bad).

Learning (to not get depressed) from failure: Fail fast, fail cheap, learn something in the process

computational predictions



experimental confirmation

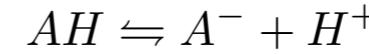
Predicting binding affinities computationally



\$500
1.5 TFLOP
50-500x speedup

GPU acceleration

$$\pi(x)K(x,y) = \pi(y)K(y,x)$$

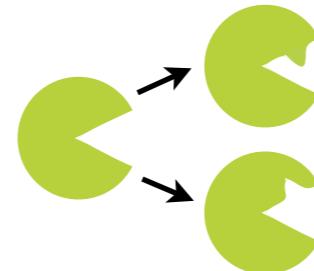


MCMC framework



enhanced algorithms

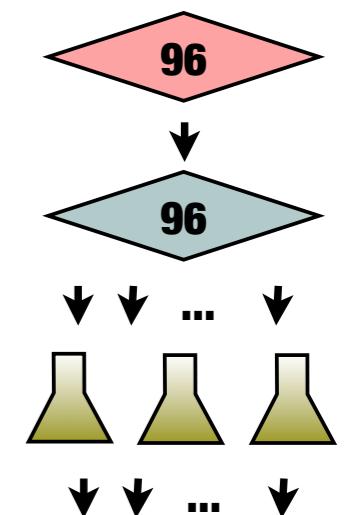
Testing predictions by mutating the target protein



mutate proteins instead of ligands

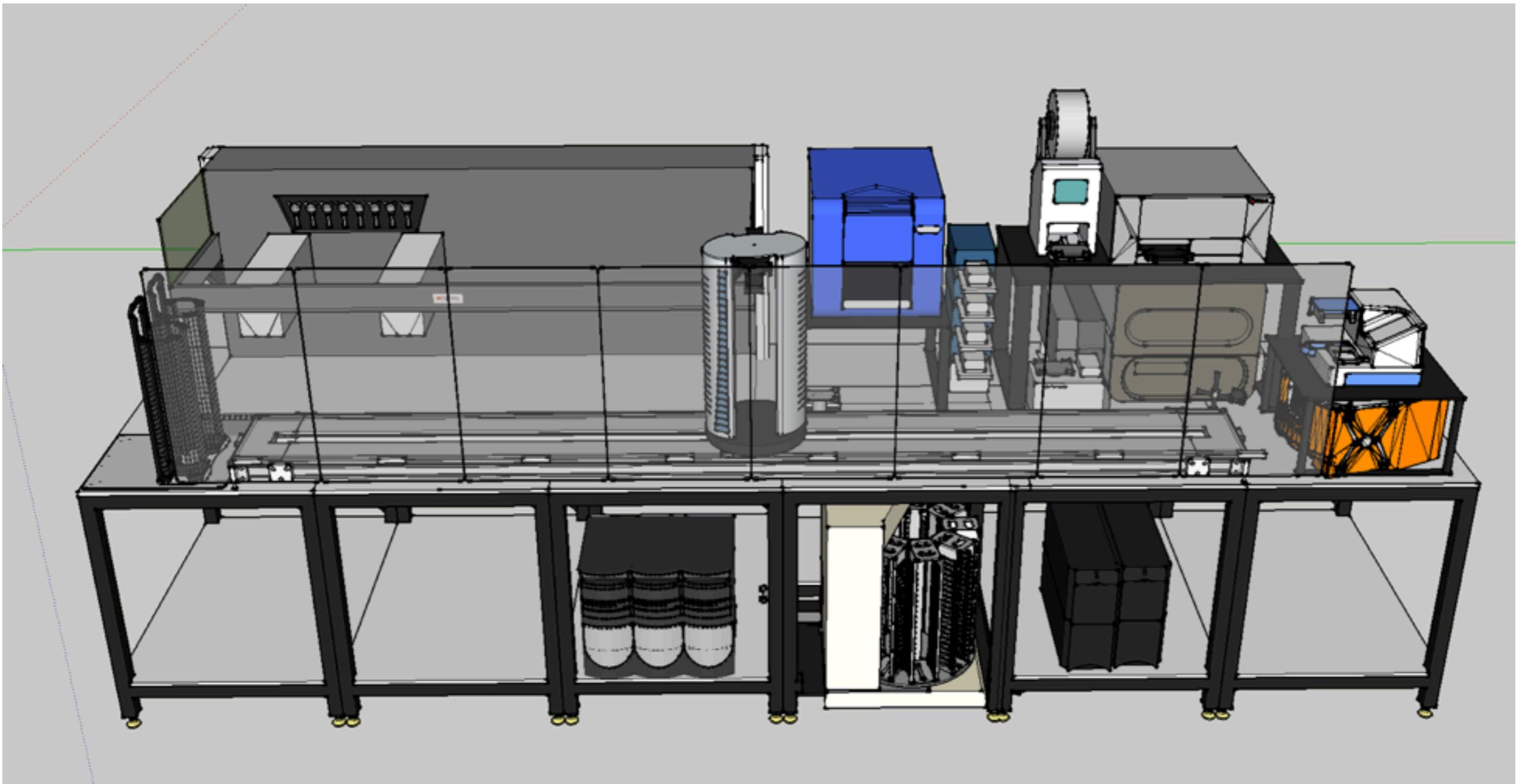


buy inexpensive ligands



high-throughput experiments

The last, best hope for data: RUG-1 (robotic undergraduate no. 1)



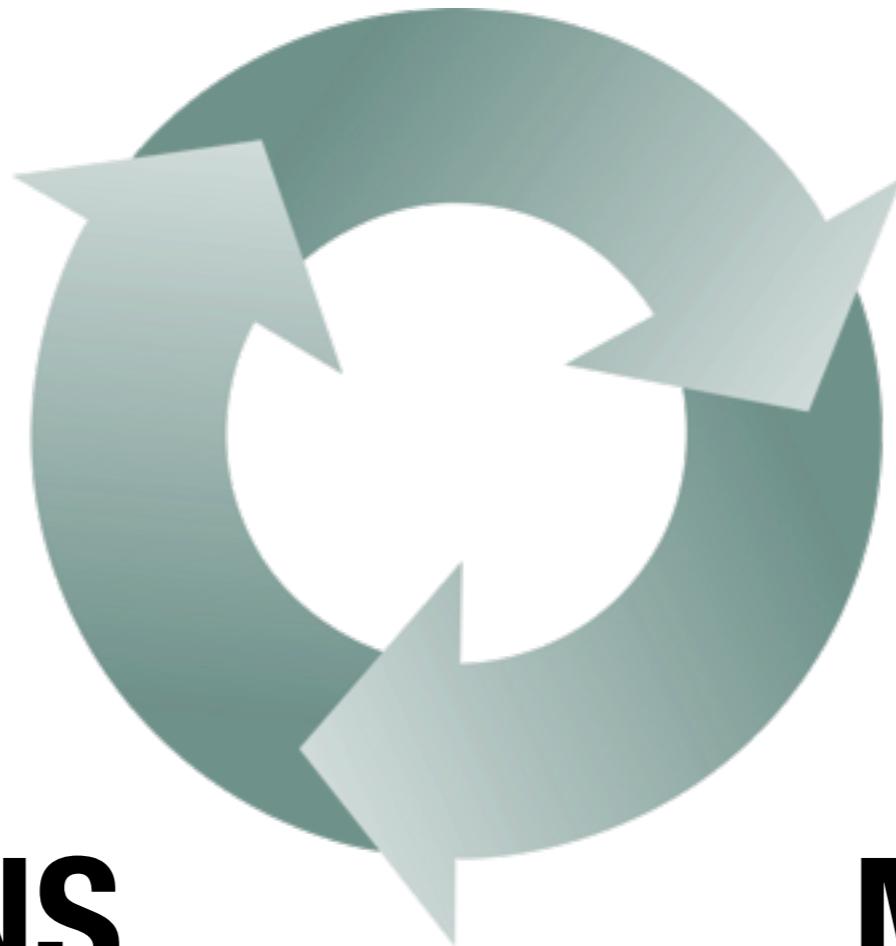
Automated platform for bacterial cloning, mutagenesis, expression, purification, and binding affinity measurement with 24/7 wallet-draining capability

The universal cycle of progress



The universal cycle of progress

DATA



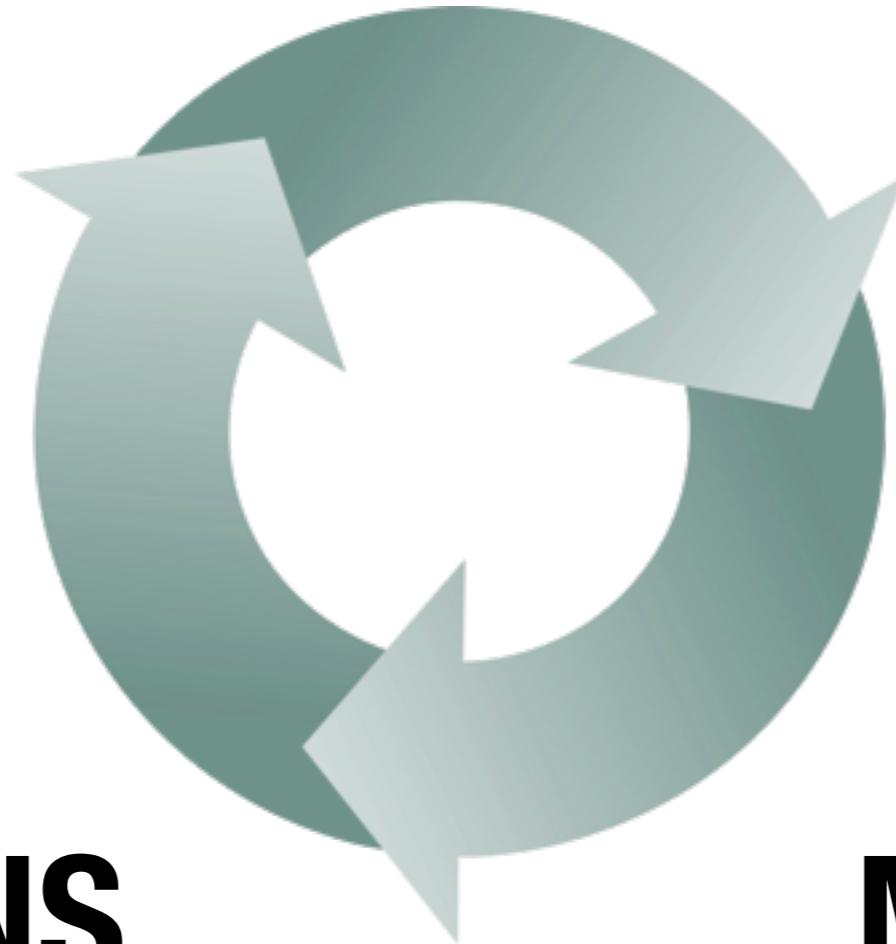
DECISIONS

MODELS



The universal cycle of progress

DATA

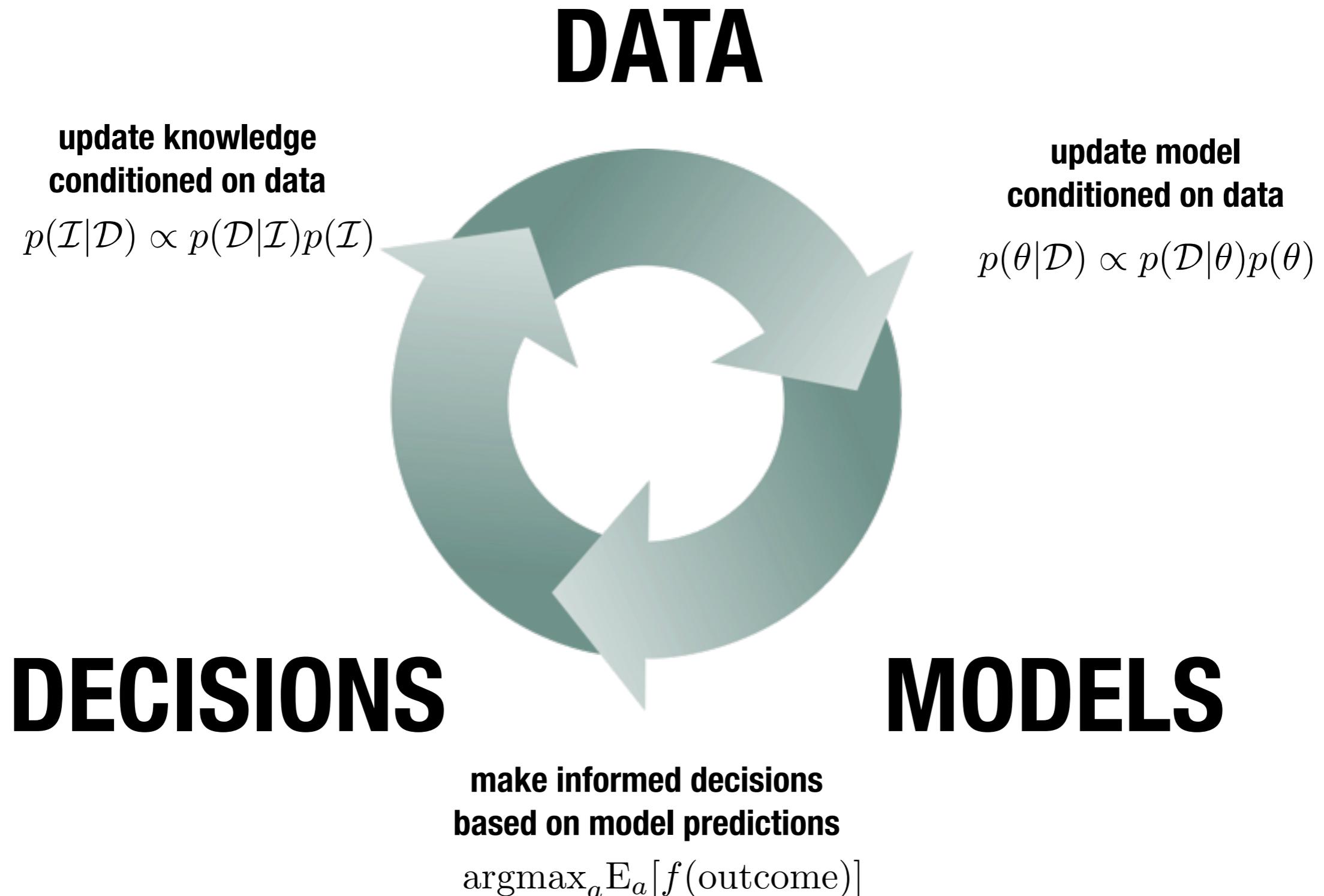


DECISIONS

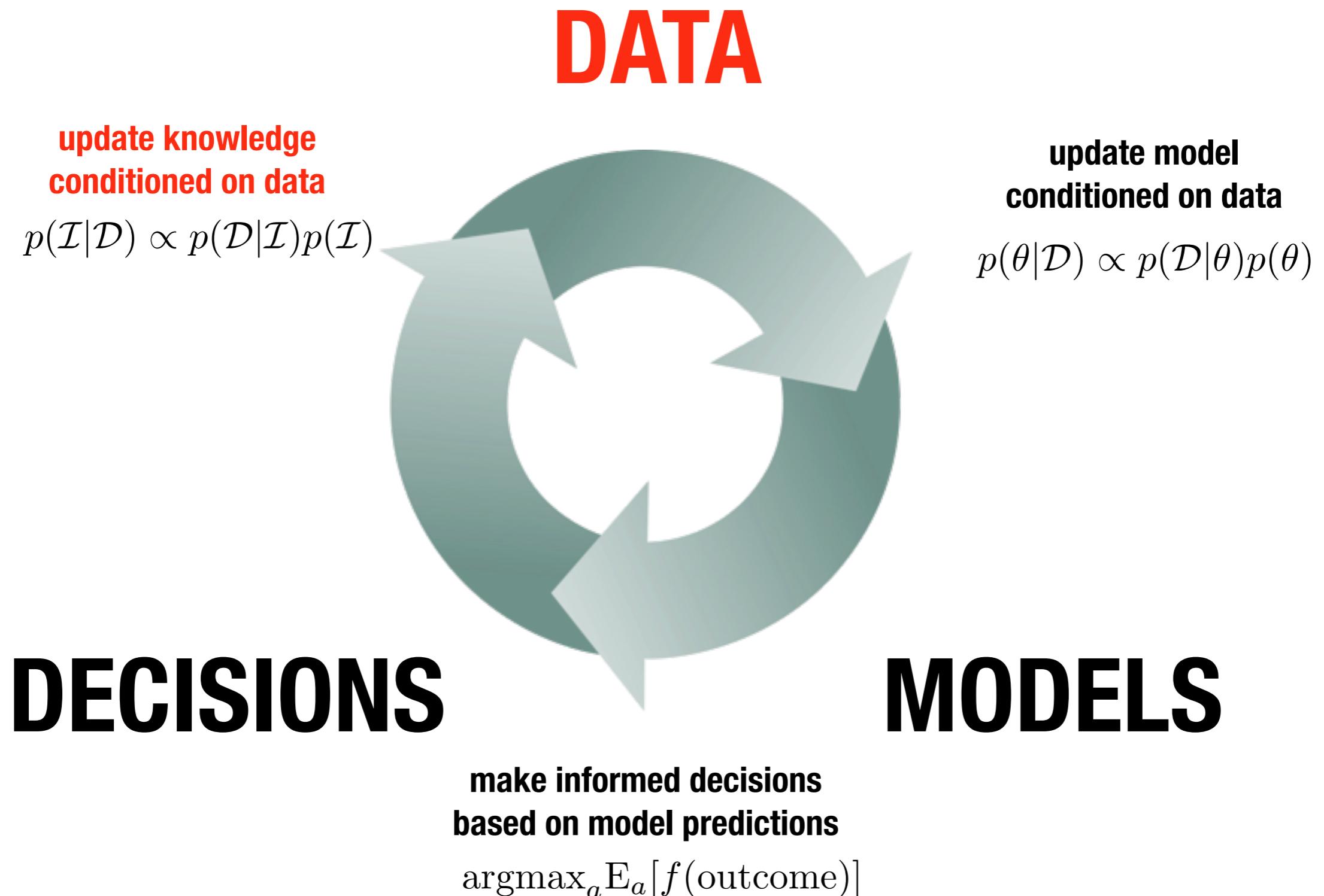
MODELS

How can we do all this without losing our minds?

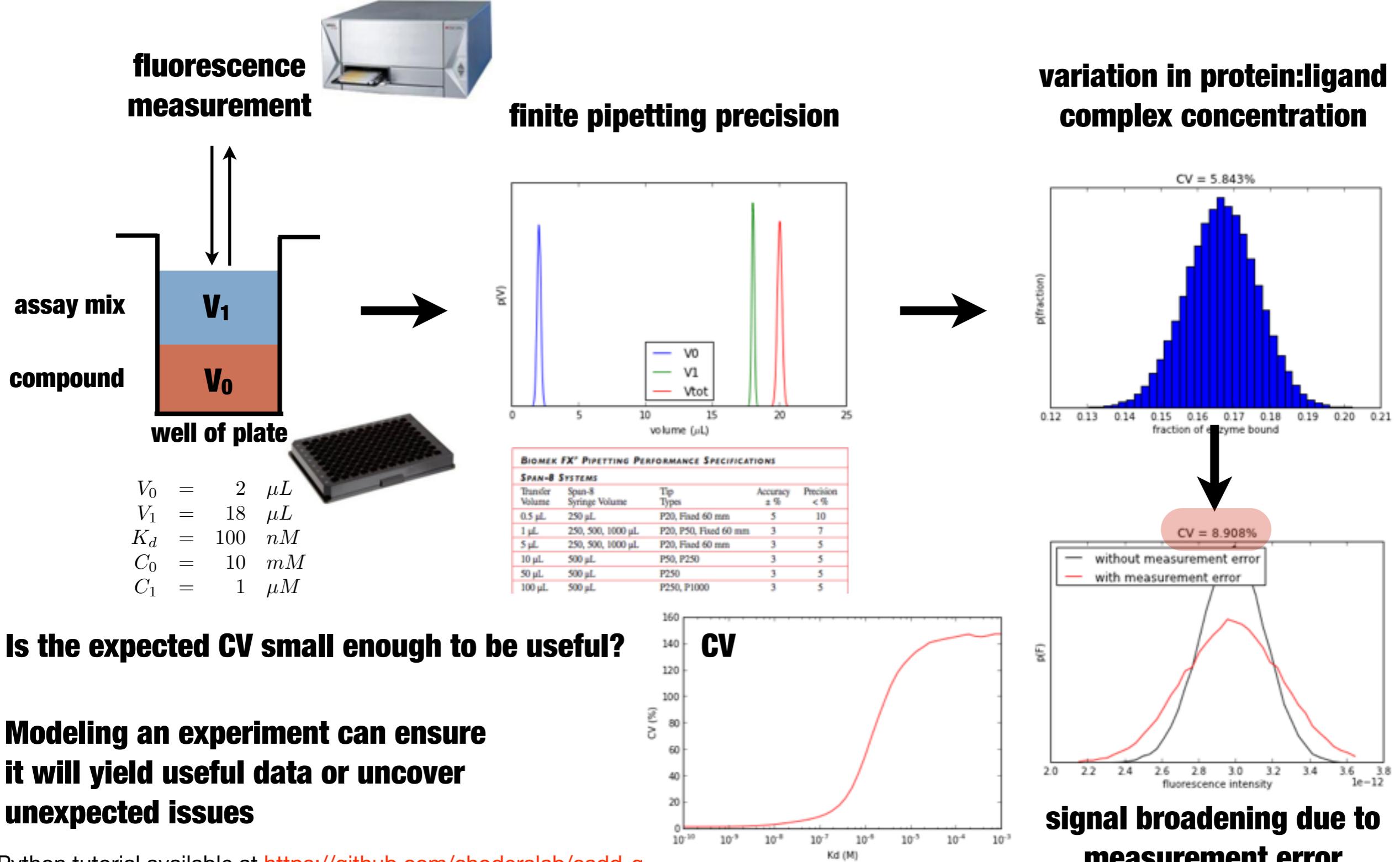
Bayesian inference can drive progress



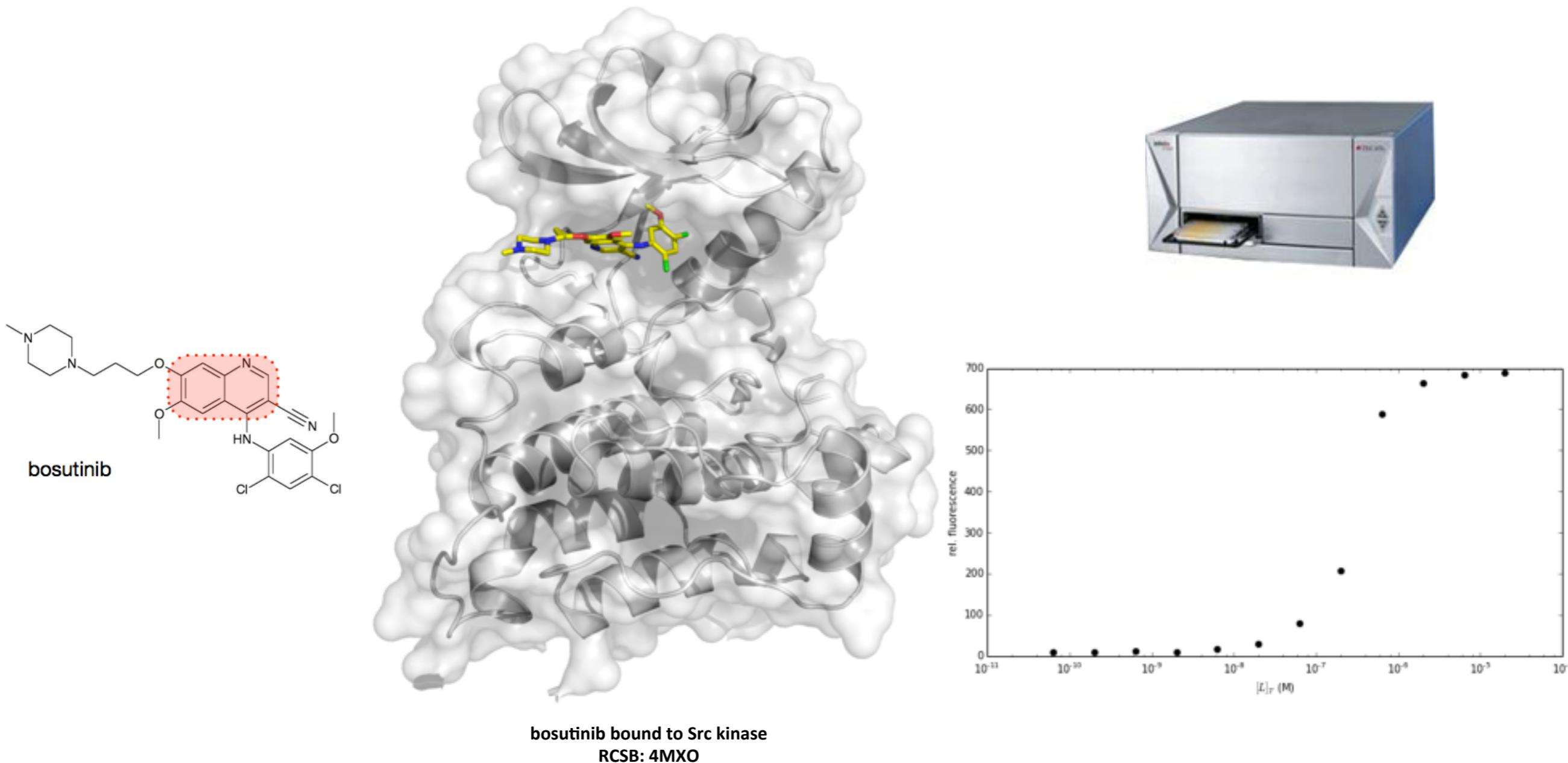
Bayesian inference can drive progress



What is experimental error and where does it come from?

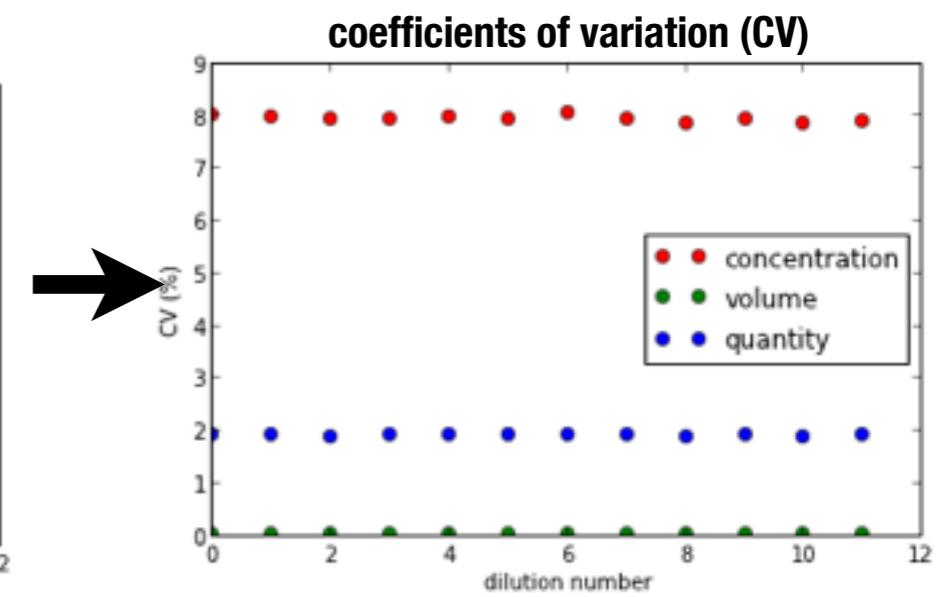
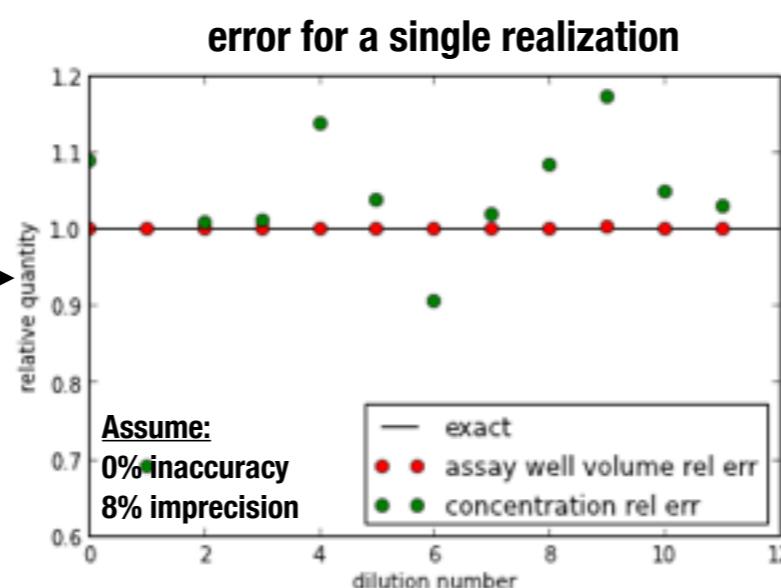
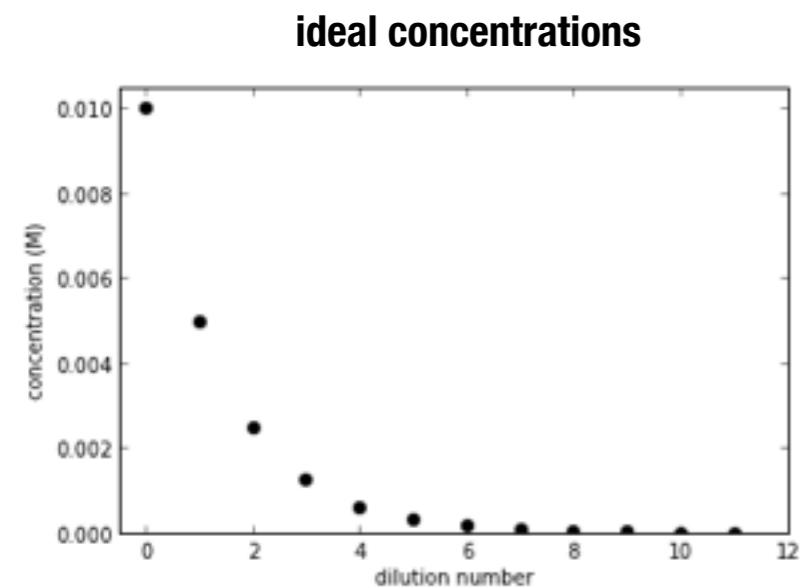
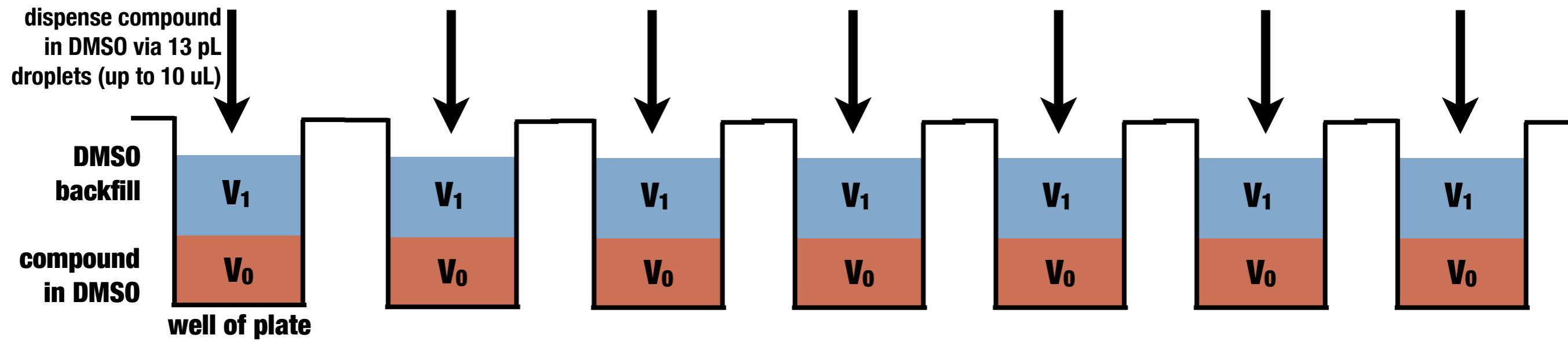


Suppose we have fluorescence data for a dilution series assay of kinase inhibitors



"Structural and Spectroscopic Analysis of the Kinase Inhibitor Bosutinib and an Isomer of Bosutinib Binding to the Abl Tyrosine Kinase Domain",
Nicholas M. Levinson, and Steven G. Boxer, PLoS ONE 7, e29828 (2012).

We can use simple Python modeling to propagate the uncertainties in the concentrations of species



Bayesian inference is easy: pymc is like ‘Bayes in a box’!

<https://pypi.python.org/pypi/pymc>



binding model

```
import pymc

# Two-component binding.
def two_component_binding(DeltaG, P, L):
    Kd = np.exp(-DeltaG)
    PL = 0.5 * ((P + L + Kd) - np.sqrt((P + L + Kd)**2 - 4*P*L)); # complex concentration (M)
    P = P - PL; # free protein concentration in sample cell after n injections (M)
    L = L - PL; # free ligand concentration in sample cell after n injections (M)
    return [P, L, PL]
```

pymc model

```
# Create a pymc model
def make_model(Pstated, dPstated, Lstated, dLstated, Fobs_i):
    N = len(Lstated)

    # Prior on binding free energies.
    DeltaG = pymc.Uniform('DeltaG', lower=-20, upper=+20, value=0.0) # binding free energy (kT)

    # Priors on true concentrations of protein and ligand.
    Ptrue = pymc.Lognormal('Ptrue', mu=np.log(Pstated**2 / np.sqrt(dPstated**2 + Pstated**2)), tau=np.sqrt(np.log(1.0)))
    Ltrue = pymc.Lognormal('Ltrue', mu=np.log(Lstated**2 / np.sqrt(dLstated**2 + Lstated**2)), tau=np.sqrt(np.log(1.0)))

    # Priors on fluorescence intensities of complexes (later divided by a factor of Pstated for scale).
    F_background = pymc.Gamma('F_background', alpha=0.1, beta=0.1, value=Fobs_i.min()) # background fluorescence
    F_PL = pymc.Gamma('F_PL', alpha=0.01, beta=0.01, value=Fobs_i.max()) # complex fluorescence
    F_L = pymc.Gamma('F_L', alpha=0.01, beta=0.01, value=Fobs_i.max()) # ligand fluorescence

    # Unknown experimental measurement error.
    log_sigma = pymc.Uniform('log_sigma', lower=-3, upper=+3, value=0.0)
    @pymc.deterministic
    def precision(log_sigma=log_sigma): # measurement precision
        return 1.0 / np.exp(log_sigma)

    # Fluorescence model.
    @pymc.deterministic
    def Fmodel(F_background=F_background, F_PL=F_PL, F_P=F_P, F_L=F_L, Ptrue=Ptrue, Ltrue=Ltrue, DeltaG=DeltaG):
        Fmodel_i = np.zeros([N])
        for i in range(N):
            [P, L, PL] = two_component_binding(DeltaG, Ptrue, Ltrue[i])
            Fmodel_i[i] = (F_PL*PL + F_L*L) / Pstated + F_background
        return Fmodel_i

    # Experimental error on fluorescence observations.
    Fobs_i = pymc.Normal('Fobs_i', mu=Fmodel, tau=precision, size=[N], observed=True, value=Fobs_i) # observed data

    # Construct dictionary of model variables.
    pymc_model = { 'Ptrue' : Ptrue, 'Ltrue' : Ltrue,
                   'log_sigma' : log_sigma, 'precision' : precision,
                   'F_PL' : F_PL, 'F_P' : F_P, 'F_L' : F_L, 'F_background' : F_background,
                   'Fmodel_i' : Fmodel, 'Fobs_i' : Fobs_i, 'DeltaG' : DeltaG }
    return pymc_model
```

pymc sampling

```
# Uncertainties in protein and ligand concentrations.
dPstated = 0.10 * Pstated # protein concentration uncertainty
dLstated = 0.08 * Lstated # ligand concentration uncertainty (due to gravimetric preparation and HP D300 dispensing)

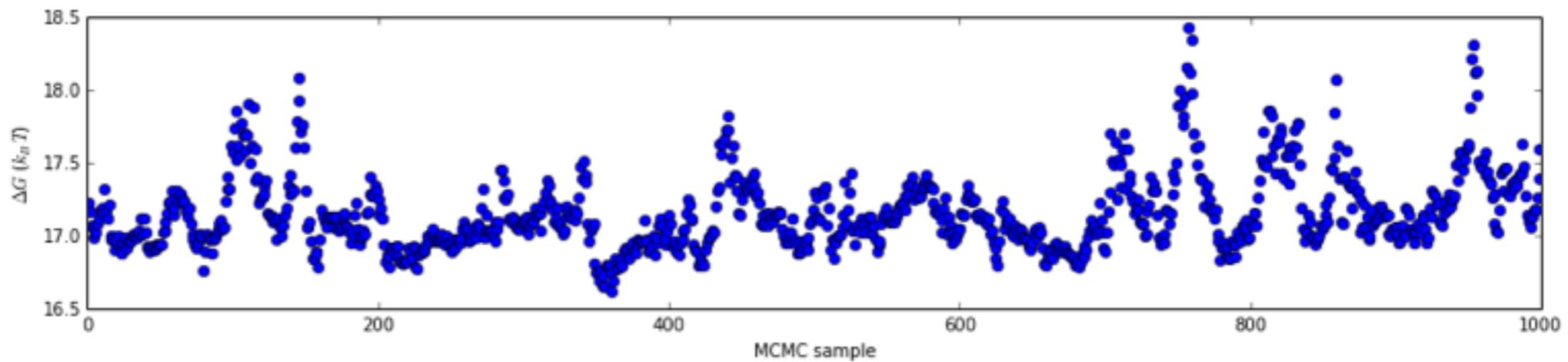
# Build model.
pymc_model = pymc.Model(make_model(Pstated, dPstated, Lstated, dLstated, F_i))

# Sample with MCMC
mcmc = pymc.MCMC(pymc_model, db='ram', name='Sampler', verbose=True)
mcmc.sample(iter=100000, burn=50000, thin=50, progress_bar=False)
```

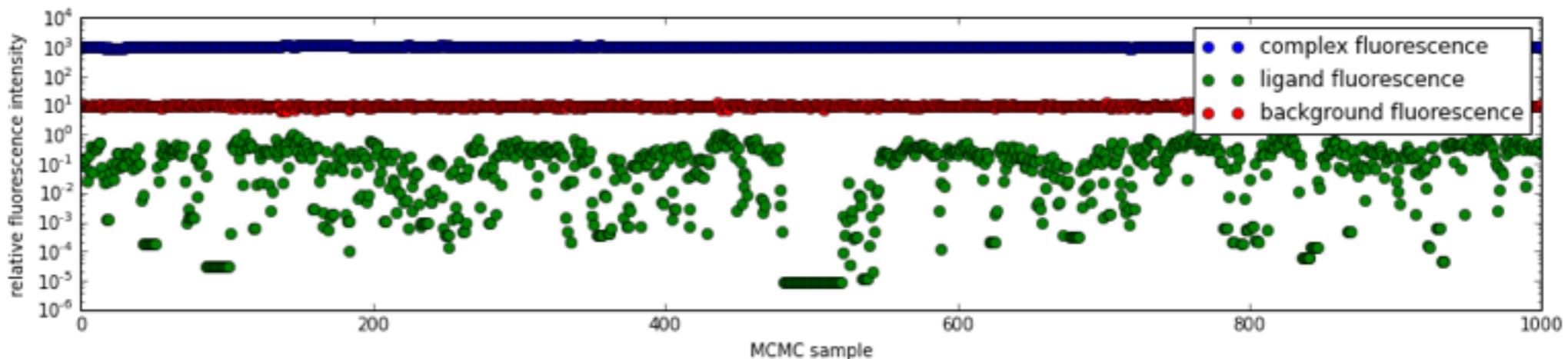
Code at <https://github.com/choderalab/cup-xiv>

Suppose we have fluorescence data for a dilution series assay of kinase inhibitors

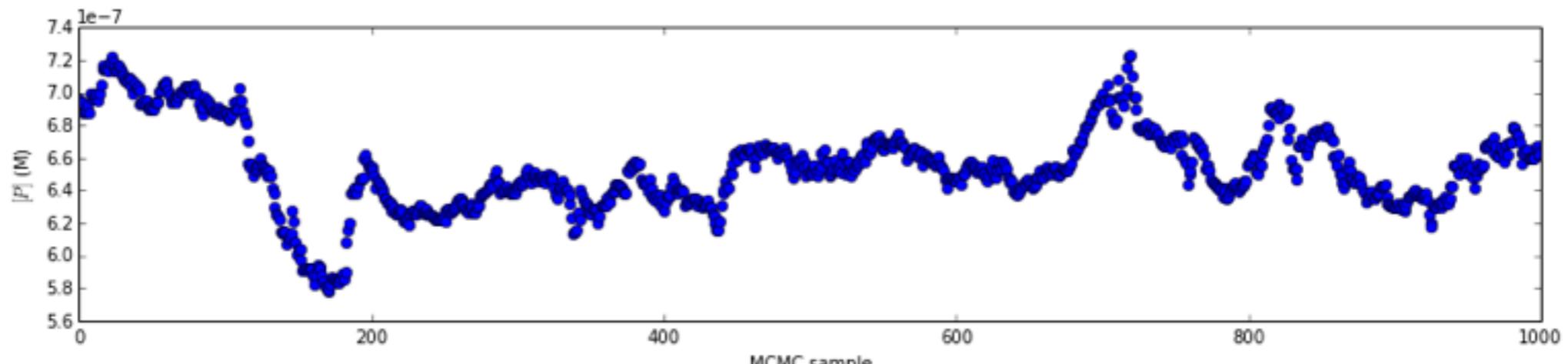
MCMC sampling
of free energies



MCMC sampling
of intrinsic
fluorescence

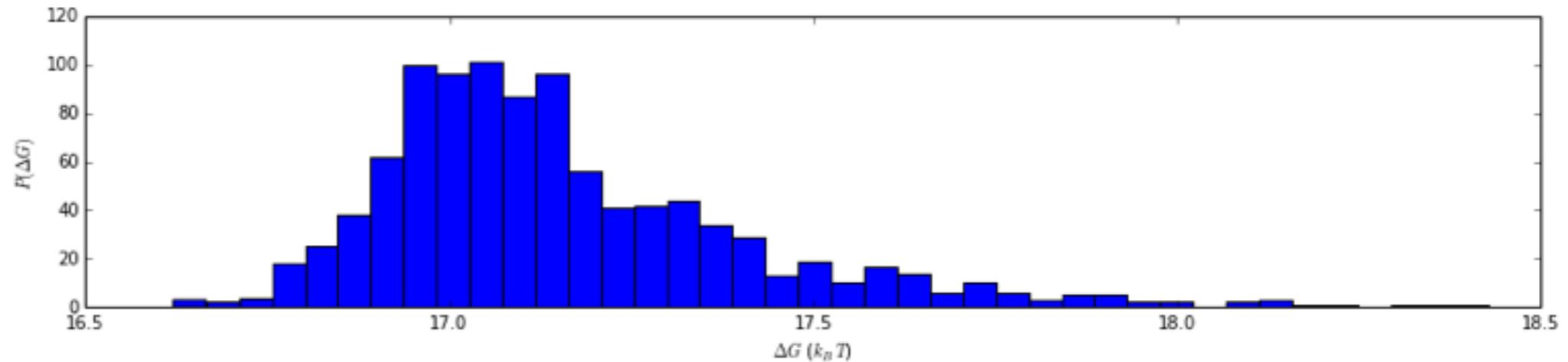


MCMC sampling
of true concentrations

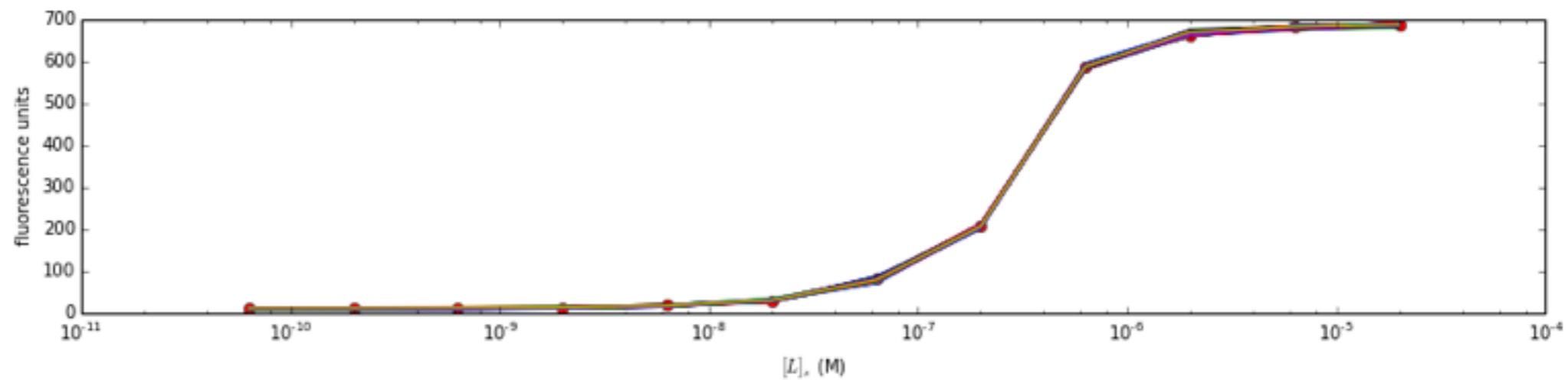


Suppose we have fluorescence data for a dilution series assay of kinase inhibitors

**posterior distribution
of binding free energies**

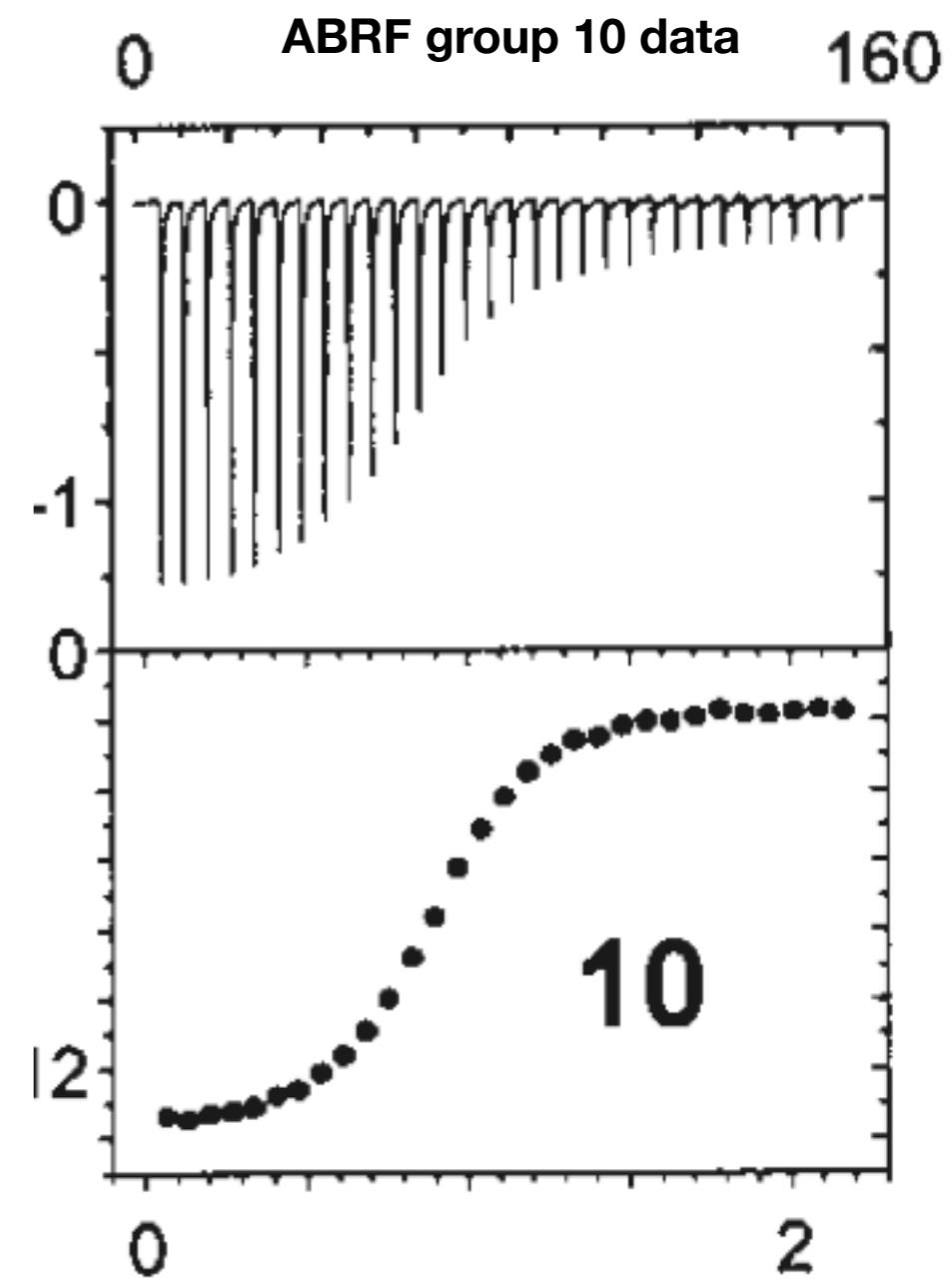
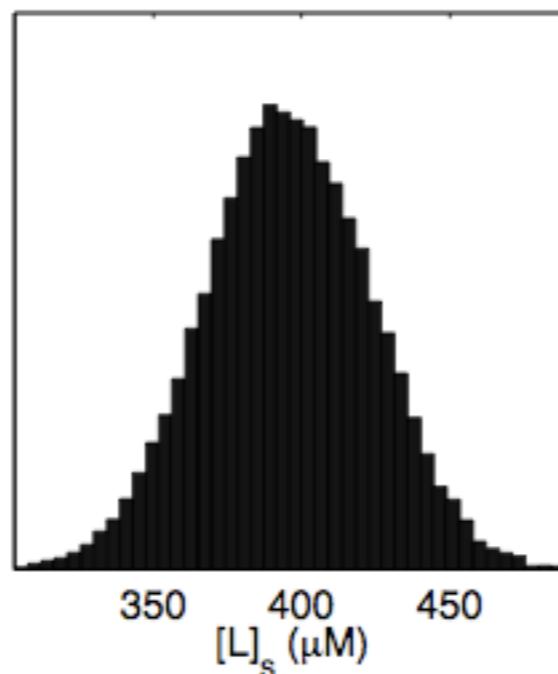
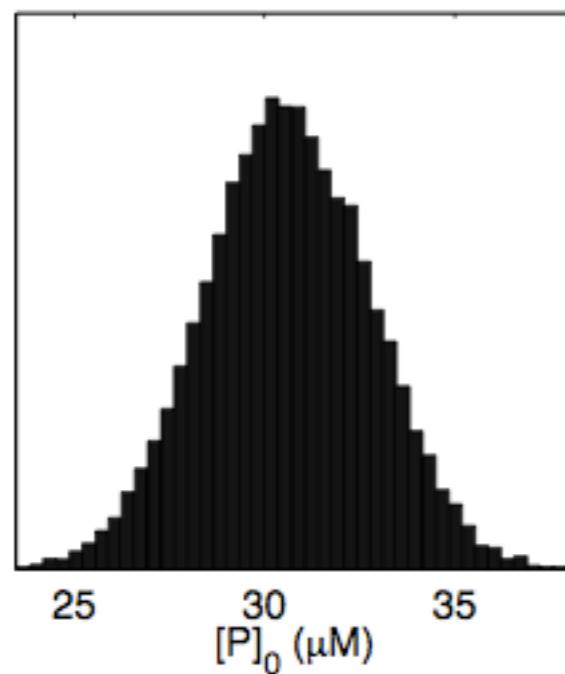
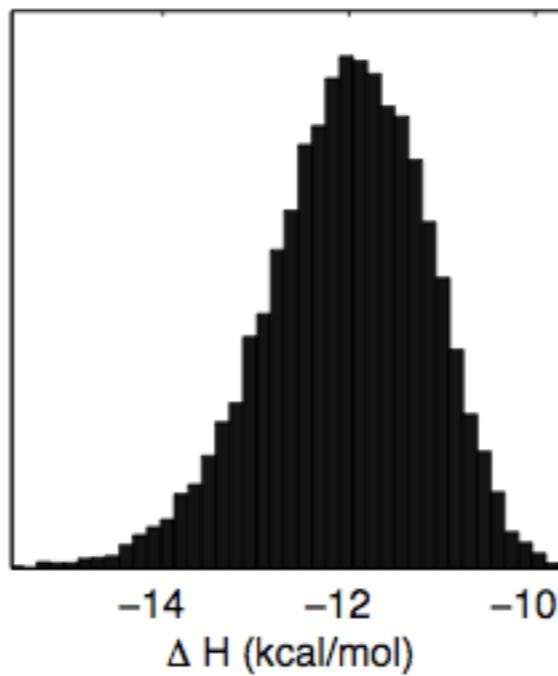
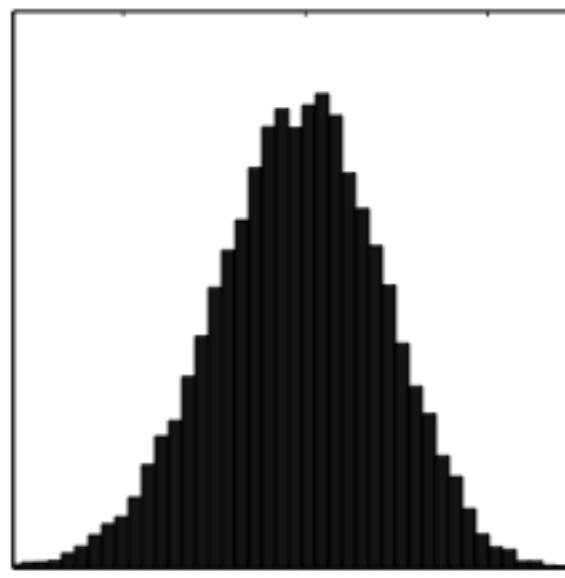


**family of models
fit to experimental
fluorescence data**



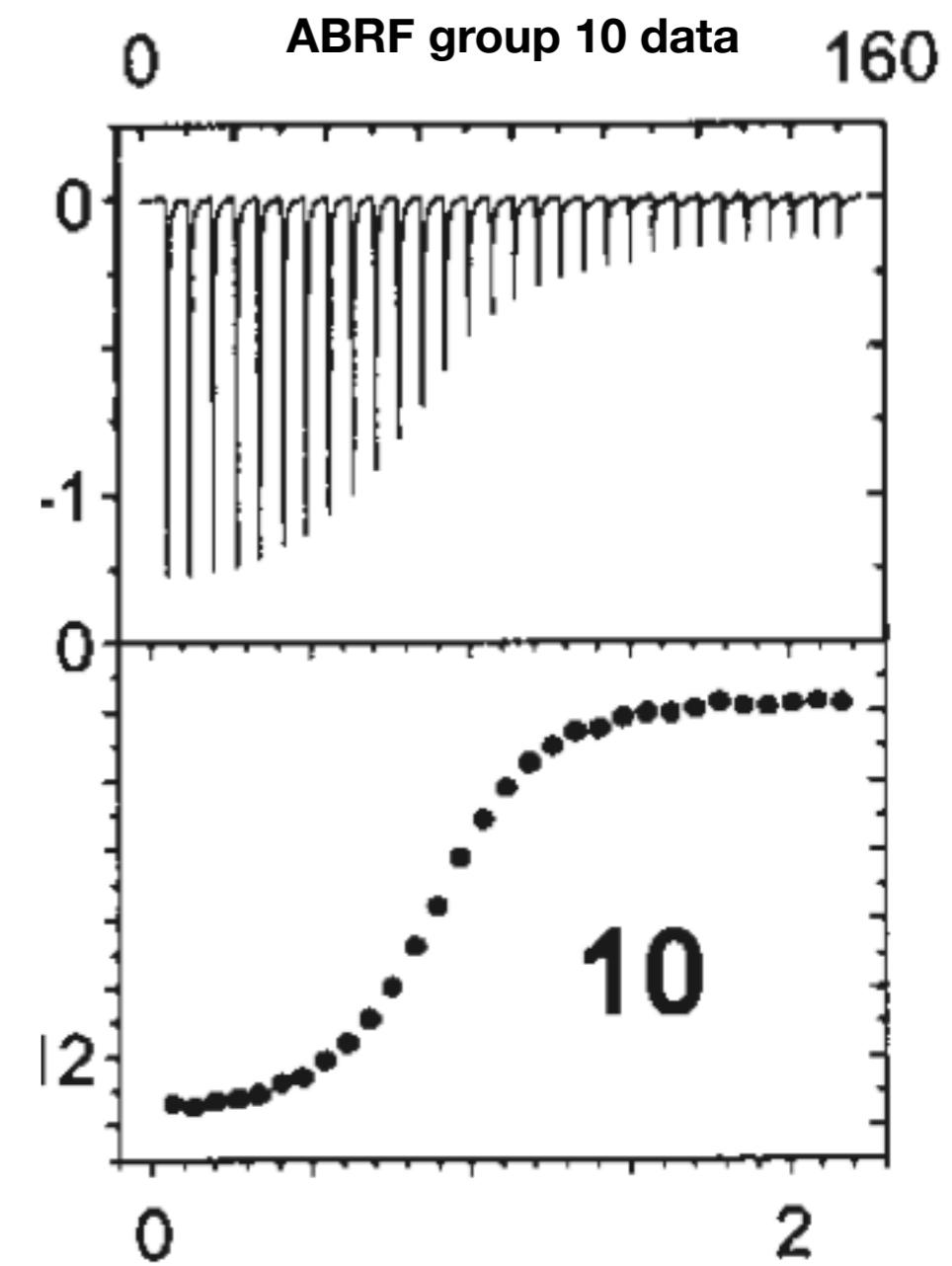
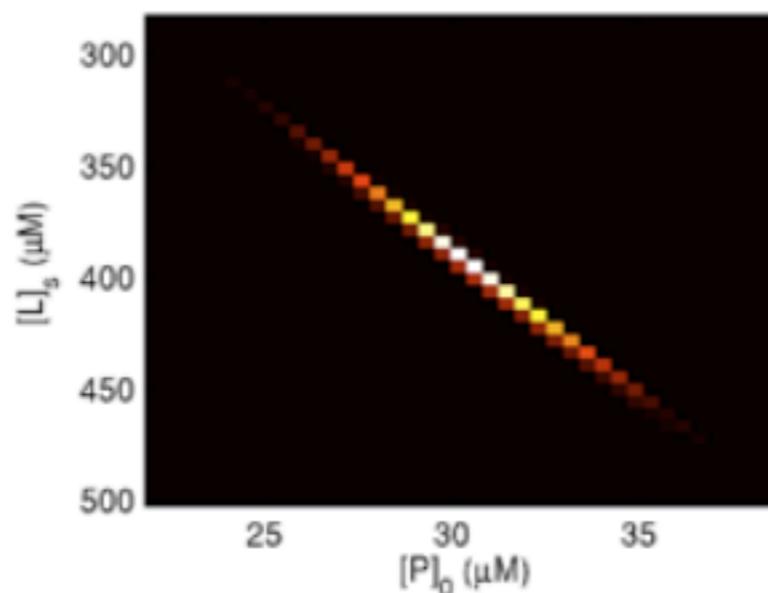
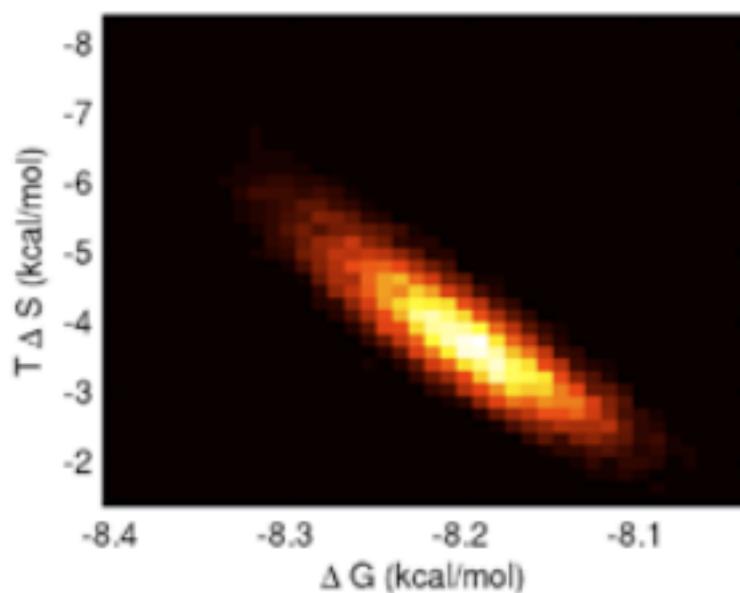
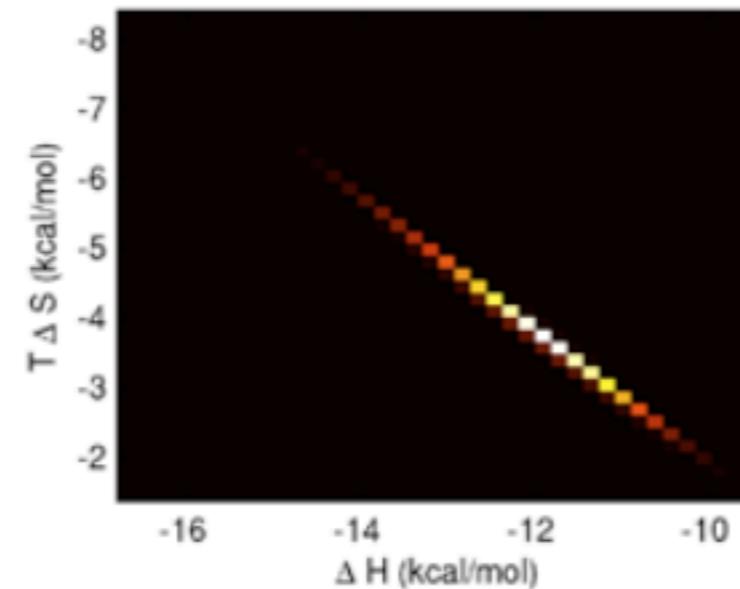
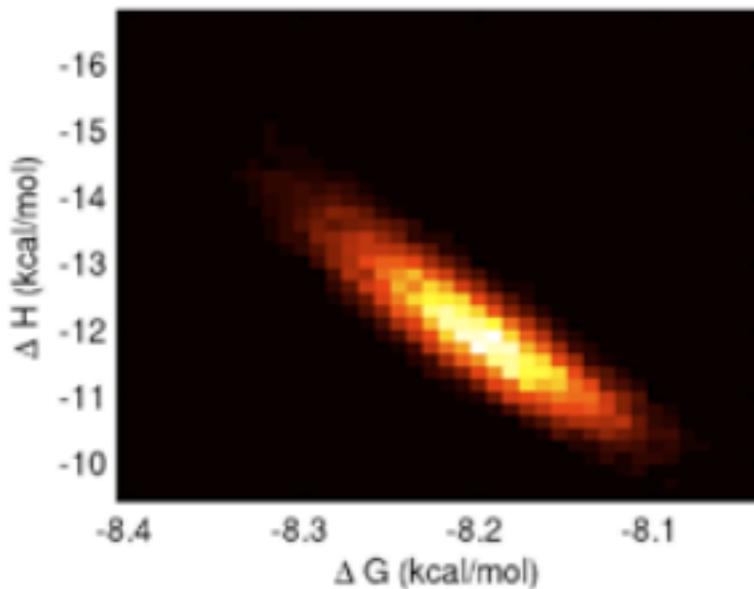
Analysis of ITC experiments: The Bayesian way

Marginal posterior probability distributions describe the uncertainty in each thermodynamic parameter inferred from the data.



Analysis of ITC experiments: The Bayesian way

Joint distribution functions describe the correlated uncertainty between any set of parameters.



Correlation in error in ΔH and $T\Delta S$ explains much of apparent entropy-enthalpy compensation behavior.

Myszka et al. J. Biomol. Tech. 14:247, 2003.

Bayesian data analysis has numerous advantages

Provides true posterior joint distribution of all thermodynamic parameters

Asymmetric confidence intervals and non-normal marginal distributions

Easy to “plug in” new binding models.

Make joint inferences from multiple experiments

Instrument parameters can be conditioned on calibration data (e.g. standard titrations)

Expected information content of new experiments can be estimated for protocol design

Experimental design:

“Will experiment X give me enough information to make it worthwhile?”

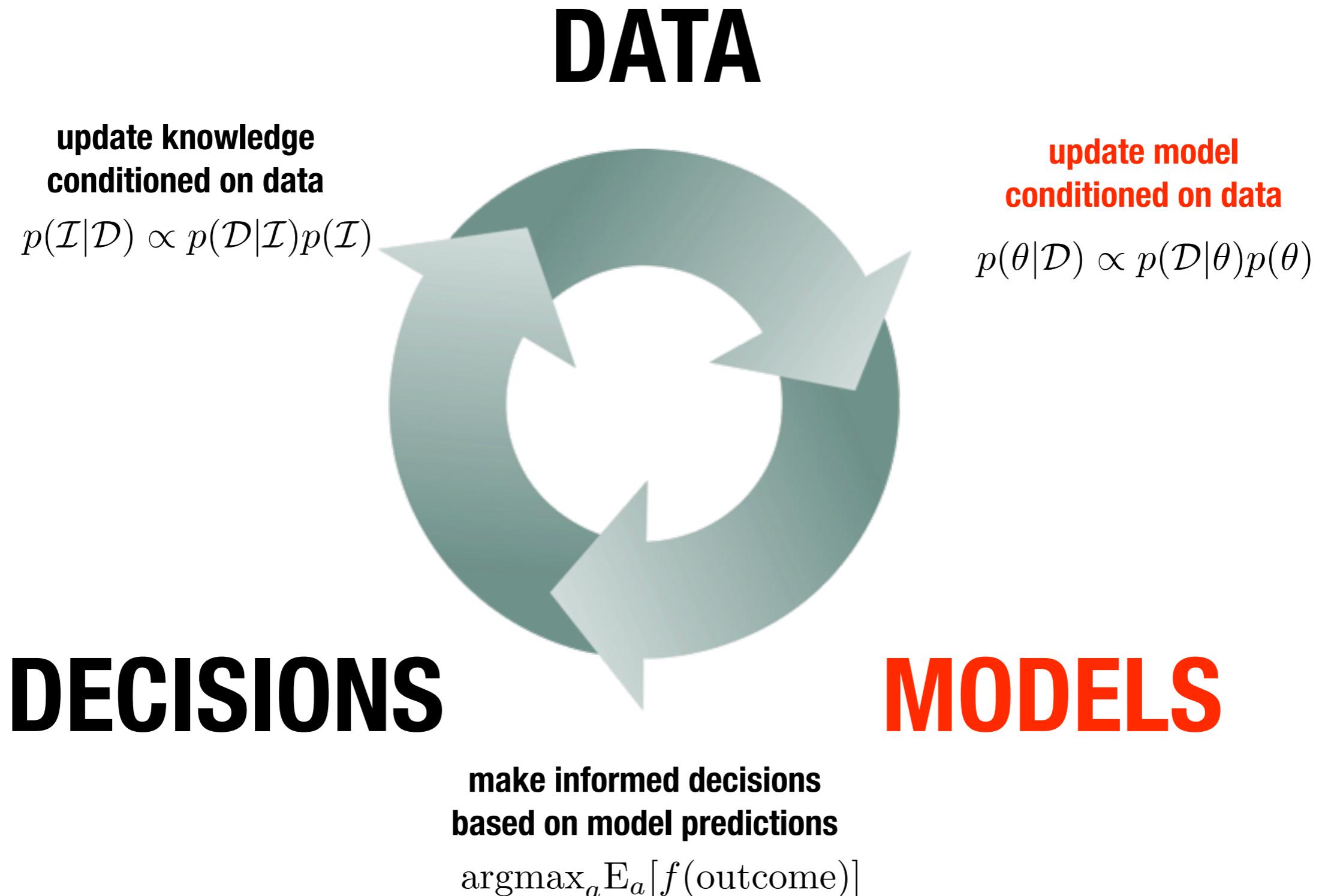
“What is the best experimental design to reduce the uncertainty in Z?”

“Do I have to run a baseline for sample X?”

Code will be available at <http://github.com/choderlab/bayesian-itc>

The universal cycle of progress

Bayesian inference can drive progress



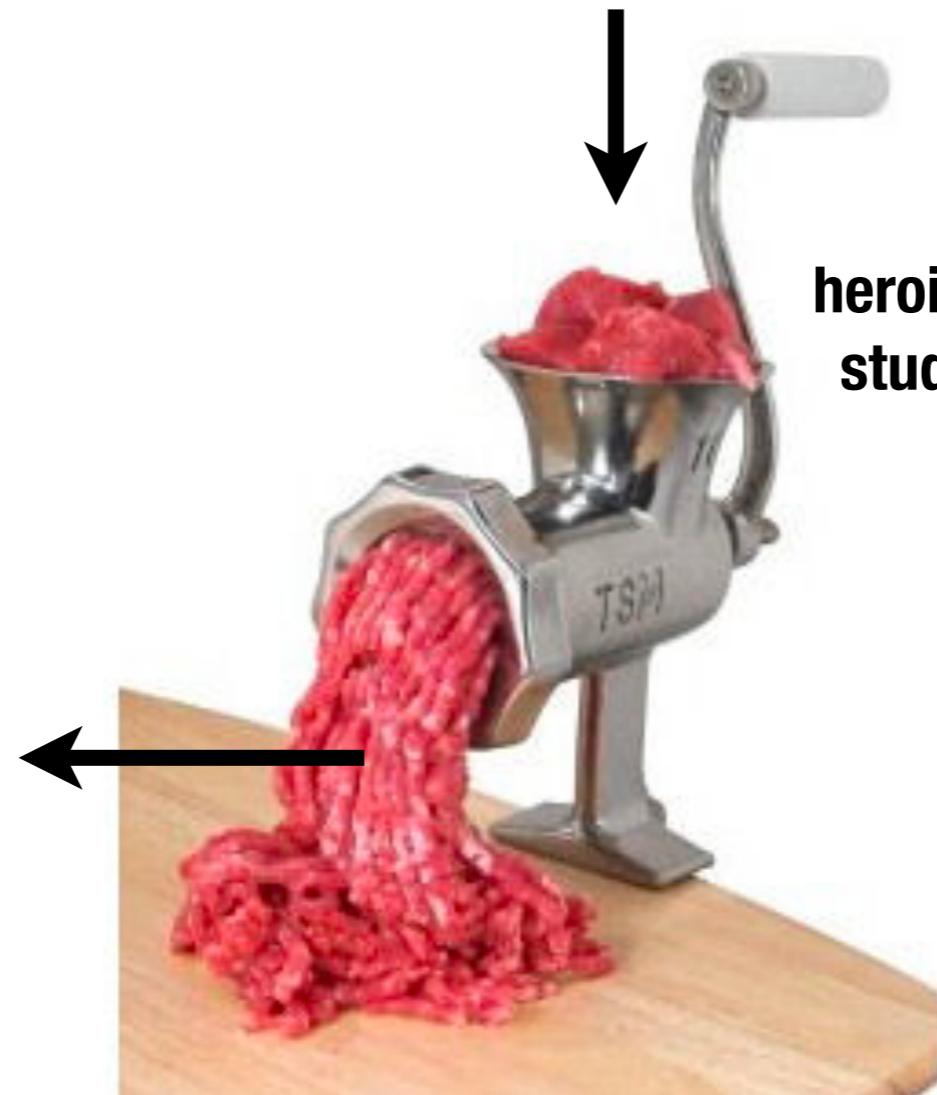
How are forcefields made?

**experimental data
quantum chemistry
keen chemical intuition**



**heroic effort by graduate
students and postdocs**

**a parameter set we
desperately hope someone
actually uses**

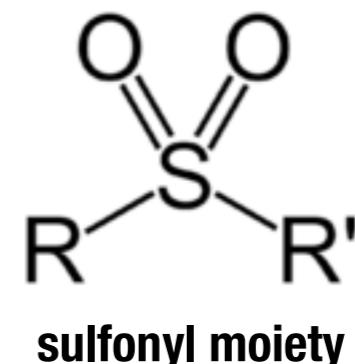


Poorly represented functional groups are problematic

Hydration free energy errors by functional group

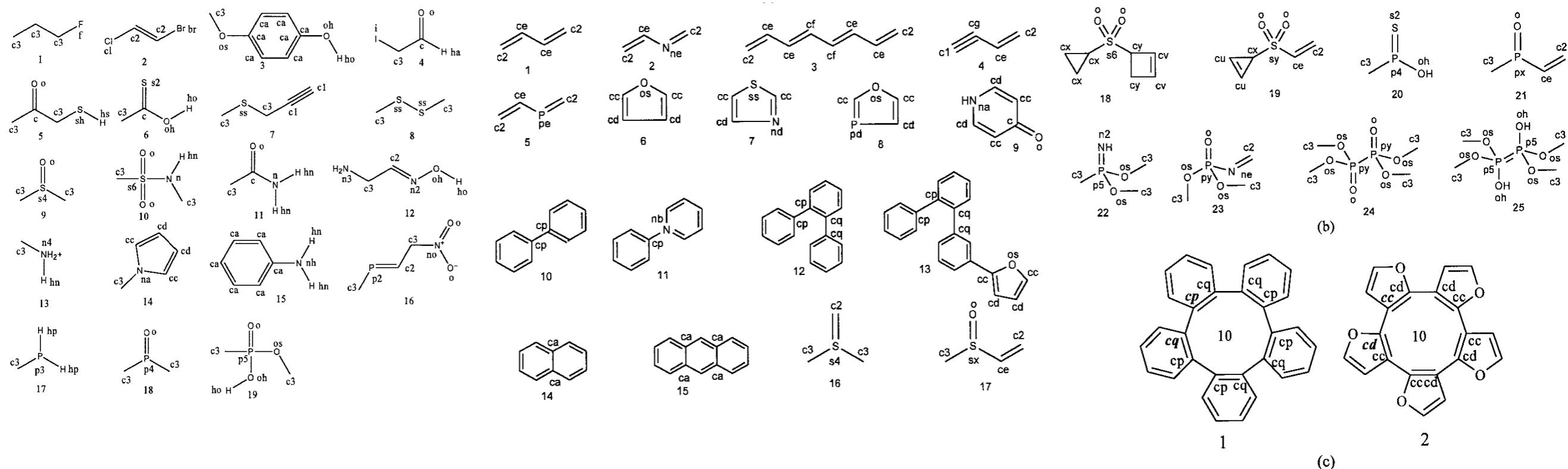
functional group	number	t-value	significance	mean error
acid	73	-7.43	4e-13	-0.34
alcohol	38	3.62	0.0003	1.29
aldehyde	20	-3.04	0.003	-0.07
alkanes	28	-1.69	0.09	0.31
alkene	35	2.34	0.02	1.07
alkyl bromide	17	3.31	0.001	1.50
alkyl chloride	31	2.31	0.02	1.09
alkyl iodide	9	0.59	0.6	0.86
alkyne	6	-0.38	0.7	0.49
amine	44	-0.65	0.5	0.55
aromatic compound	170	-1.05	0.3	0.55
aryl chloride	20	1.65	0.1	1.04
carbonitrile	12	3.22	0.001	1.63
cyclic hydrocarbon	8	-1.18	0.2	0.21
ester	8	-1.69	0.09	0.02
ether	42	2.18	0.03	1.01
halogen derivative	22	0.32	0.8	0.73
heterocyclic compound	48	2.38	0.02	1.02
hypervalent S	5	-4.55	7e-06	-1.50
ketone	25	-2.77	0.006	0.05
nitro compound	17	1.86	0.06	1.13
other	29	-0.48	0.6	0.55
phenol or hydroxyhetarene	33	2.72	0.007	1.16
thiol	5	0.51	0.6	0.89

Poorly represented functionalities in parameterization set lead to pathologies in use.



As drug discovery explores new parts of chemical space, how can forcefields keep up?

GAFF was parameterized for a limited diversity of compounds (and hence atom types):



Extension of GAFF by others “nontrivial” (effectively impossible for anyone but Junmei)

Our approach to parameterization has evolved over time, but it's still not completely automated by any measure

year	forcefield	parameter fitting	atom types
1990s	AMBER parm96	lots of “hand tweaking”	hand-picked
early 2000s	GAFF	genetic algorithm	hand-picked
mid 2000s	TIP4P-Ew	least-squares optimization	hand-picked

(an AMBER-centric view, but meant to be illustrative)

Torsion barrier for peptide bond from `parm96.dat`

```
X -C -N -X    4   10.00      180.0      2.      AA | check Wendy? & NMA
```

How can we move to entirely automated schemes that are easy to grow and refine?

What would we want out of a forcefield parameterization scheme (while we're dreaming)?

Everything is **automatic**; don't need to tweak things by hand.

There is only one **Christopher Bayly**, and he's pretty busy, so we shouldn't need to consult him too frequently for deep feats of chemical insight.

Automatically chooses optimal functional forms.

If we find ourselves in uncharted territory in chemical space, **can add more data**. (Maybe it could even tell us which new data would be useful!)

I am lazy and don't want to learn new algorithms.

Would give us an idea of **how reliable** it new predictions are expected to be.

Is there really a procedure that could fit these criteria?

Bayesian inference can help!

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

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

\mathcal{D} data

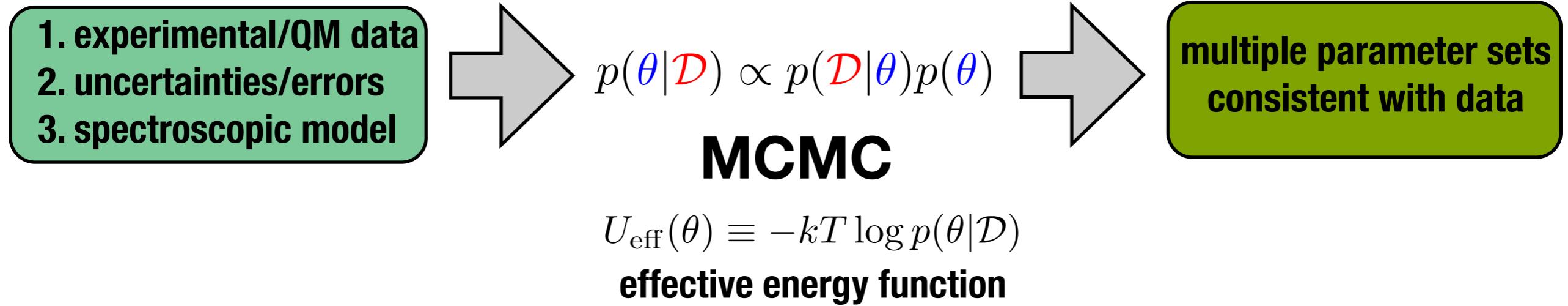
θ forcefield parameters

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

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

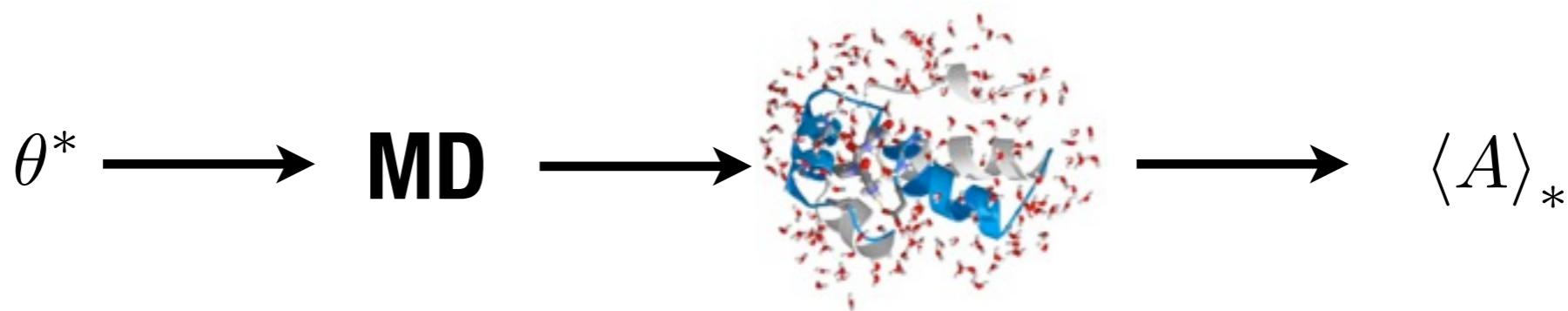
$p(\theta)$ prior on forcefield parameters

Bayesian parameterization can exploit the one tool we already know how to use well, Markov chain Monte Carlo

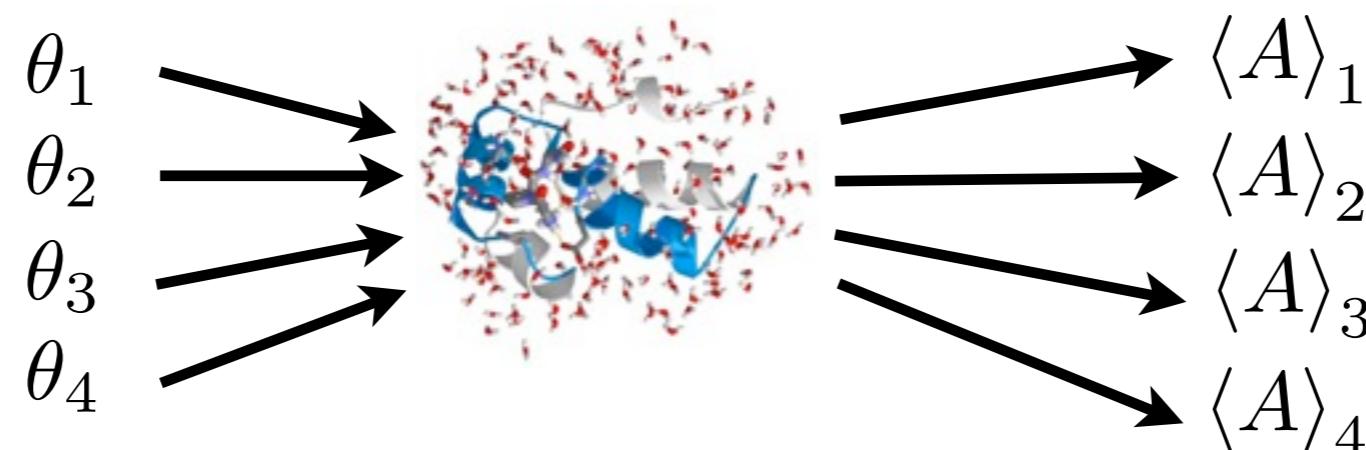


Evaluation of computed properties with multiple parameter sets gives an **estimate of systematic error** in forcefield

Compute properties with one representative parameter set from collection



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



Can estimate both **statistical** and **systematic** components of computed property!

Reparameterizing a simple GBSA model

OBC-GBSA model

Only change GB parameters, not SA

“Quick-and-dirty” (e.g. done last night)

% SMARTS atom match	typename	comment	% atomtype	radius (Å)	scalingFactor
[#6]	C	% carbon	HN	1.25	1.0000
[\$([#6]=[*]),\$([#6]#[*]),\$([#6]:[*])]	C(pi)	% carbon (pi-bonded)	HO	1.00	1.0000
[#7]	N	% nitrogen	HP	1.00	1.0000
[\$([#7]=[*]),\$([#7]#[*]),\$([#7]:[*])]	N(pi)	% nitrogen (pi-bonded)	HS	1.25	1.0000
[#8]	O	% oxygen	HC	1.25	1.0000
[\$([#8]=[*]),\$([#8]#[*]),\$([#8]:[*])]	O(pi)	% oxygen (pi-bonded)	C(pi)	2.00	1.0000
[\$([#1]~[#7])]	HN	% H (on N)	C	1.80	1.0000
[\$([#1]~[#8])]	HO	% H (on O)	N(pi)	1.65	1.0000
[\$([#1]~[#15])]	HP	% H (on P)	N	1.65	1.0000
[\$([#1]~[#16])]	HS	% H (on S)	O(pi)	1.40	1.0000
[\$([#1]~[#6])]	HC	% H (on C)	O	1.35	1.0000
[#15]	P	% phosphorus	P	2.15	1.0000
[#16]	S	% sulfur	S	1.95	1.0000
[#9]	F	% fluorine	F	1.50	1.0000
[#17]	Cl	% chlorine	Cl	1.80	1.0000
[#35]	Br	% bromine	Br	2.40	1.0000
[#53]	I	% iodine	I	2.60	1.0000

Train: ~200 molecules from Mobley et al. 504 molecule test set

Test: ~200 different molecules from same set

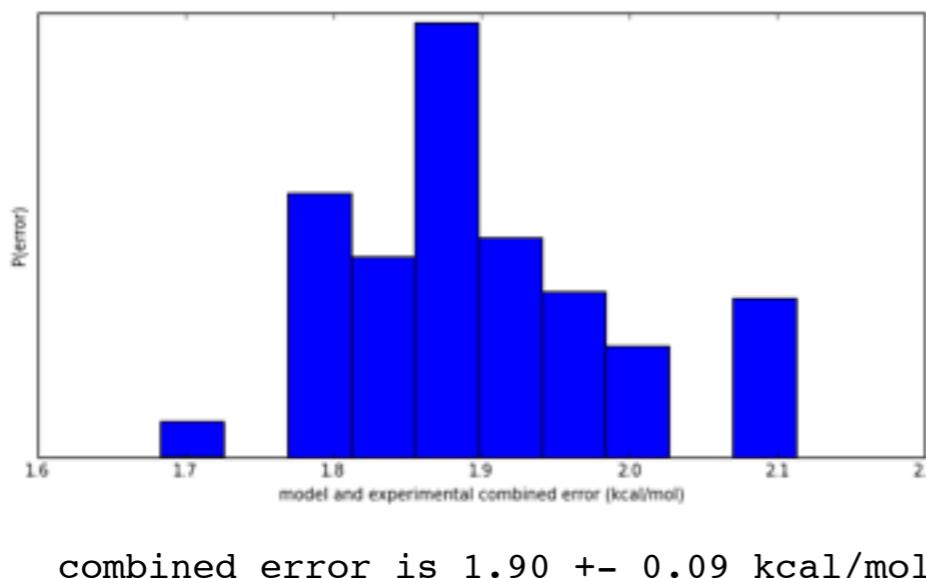
Reparameterizing a simple GBSA model

OBC-GBSA model

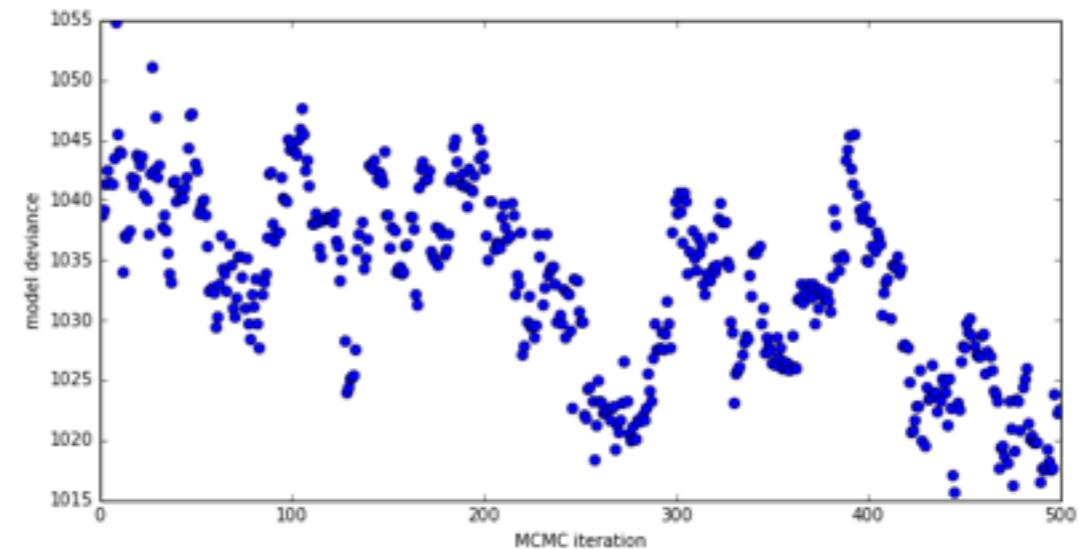
Only change GB parameters, not SA

“Quick-and-dirty” (e.g. done last night)

**combined estimate of experimental
error and model limitation error**



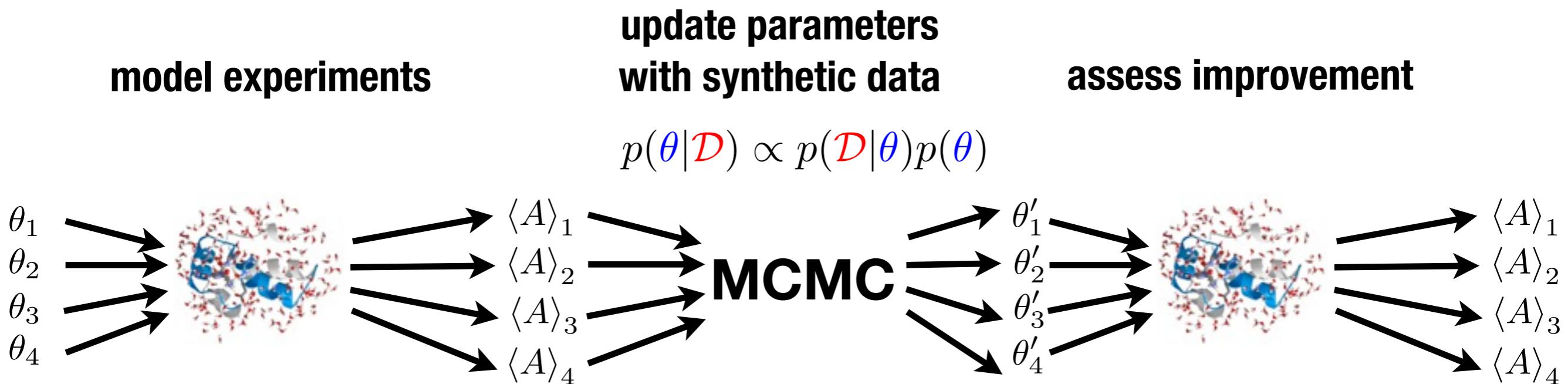
**model deviance
(I should have run it longer)**



For each predicted compound in the test set, we have three pieces of information:

Estimated experimental and model limitation error:	1.90 ± 0.09 kcal/mol
Uncertainty from model parameter uncertainty:	0.07 kcal/mol
Actual compound error:	1.94 kcal/mol

Bayesian experimental design can tell us which kinds of data would best help reduce systematic error

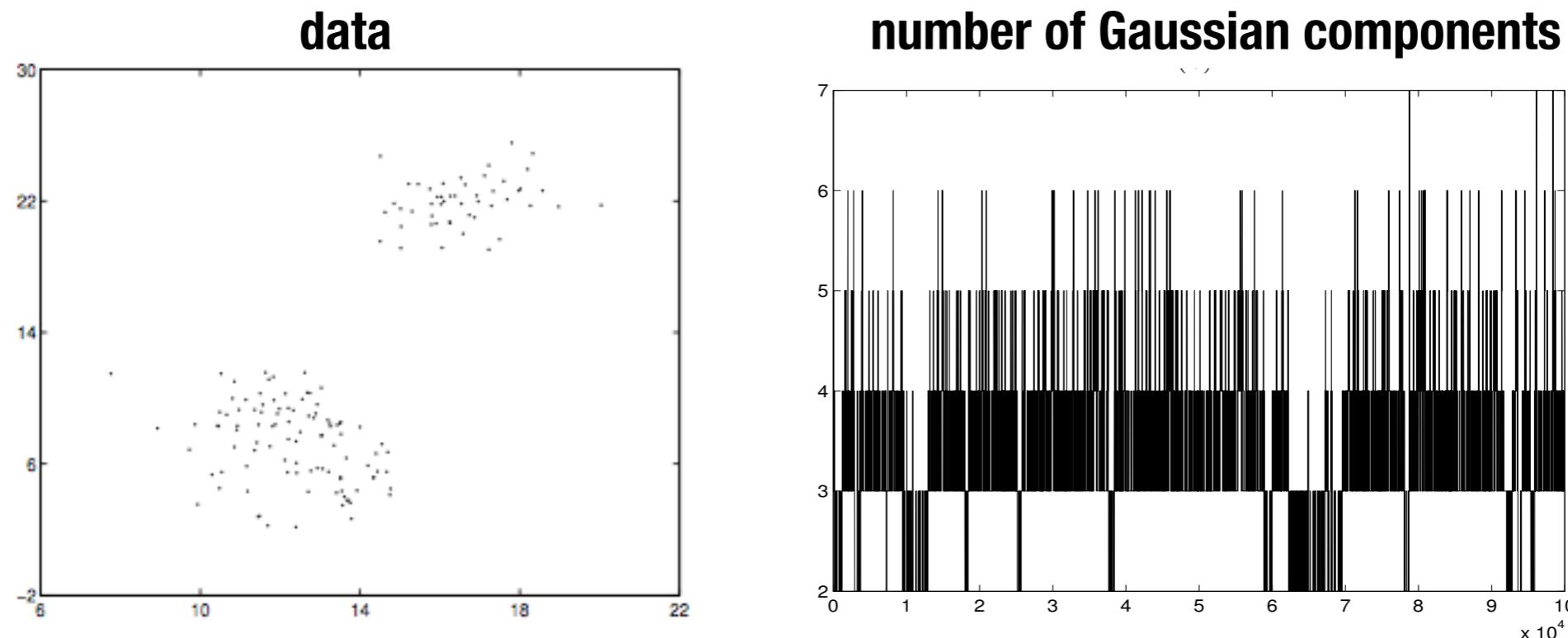


Modeling the outcome of new experiments can suggest which will be **most informative** in reducing systematic error

Bayesian model selection can be a powerful tool in forcefield development and parameterization

Classic Gaussian mixture model case:

- How many components do the data support?
- What are their parameters?



Bayesian inference penalizes complexity

Bayesian model selection can be a powerful tool in forcefield development and parameterization

Reversible-jump Monte Carlo (RJMC) allows jumps among models; models best supported by data are preferentially sampled. (Calculation is analogous to grand canonical MC.)

RJMC moves simply have to satisfy detailed balance in model space (e.g. via Metropolis-Hastings):

$$P_{\text{accept}} = \min \left\{ 1, \frac{P(\text{new})}{P(\text{old})} \cdot \frac{P(\text{old}|\text{new})}{P(\text{new}|\text{old})} \right\}$$

RJMC can automate selection of **most appropriate functional form** and/or **mixing rules** for vdW:

Lennard-Jones

exponential-6

Halgren buffer 14-7

Lorentz-Berthelot

Waldman-Hagler

Halgren HHG

RJMC can automate selection of **atom types**: only as many types as the data supports!

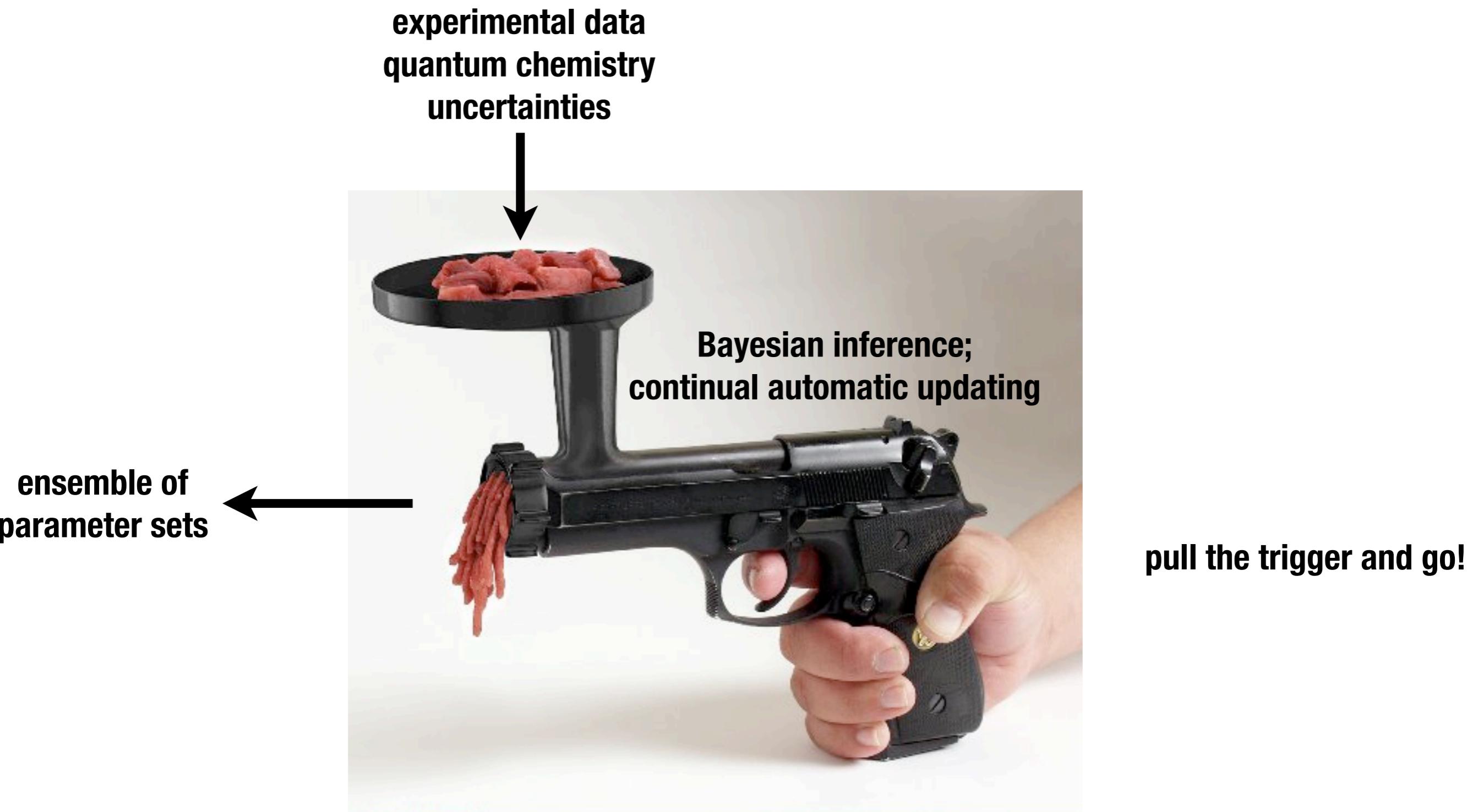
splitting moves that split off a subtype (e.g. sp₃ carbon split from generic carbon type)

subtype deletion moves that remove subtypes

What would we want out of a forcefield parameterization scheme?

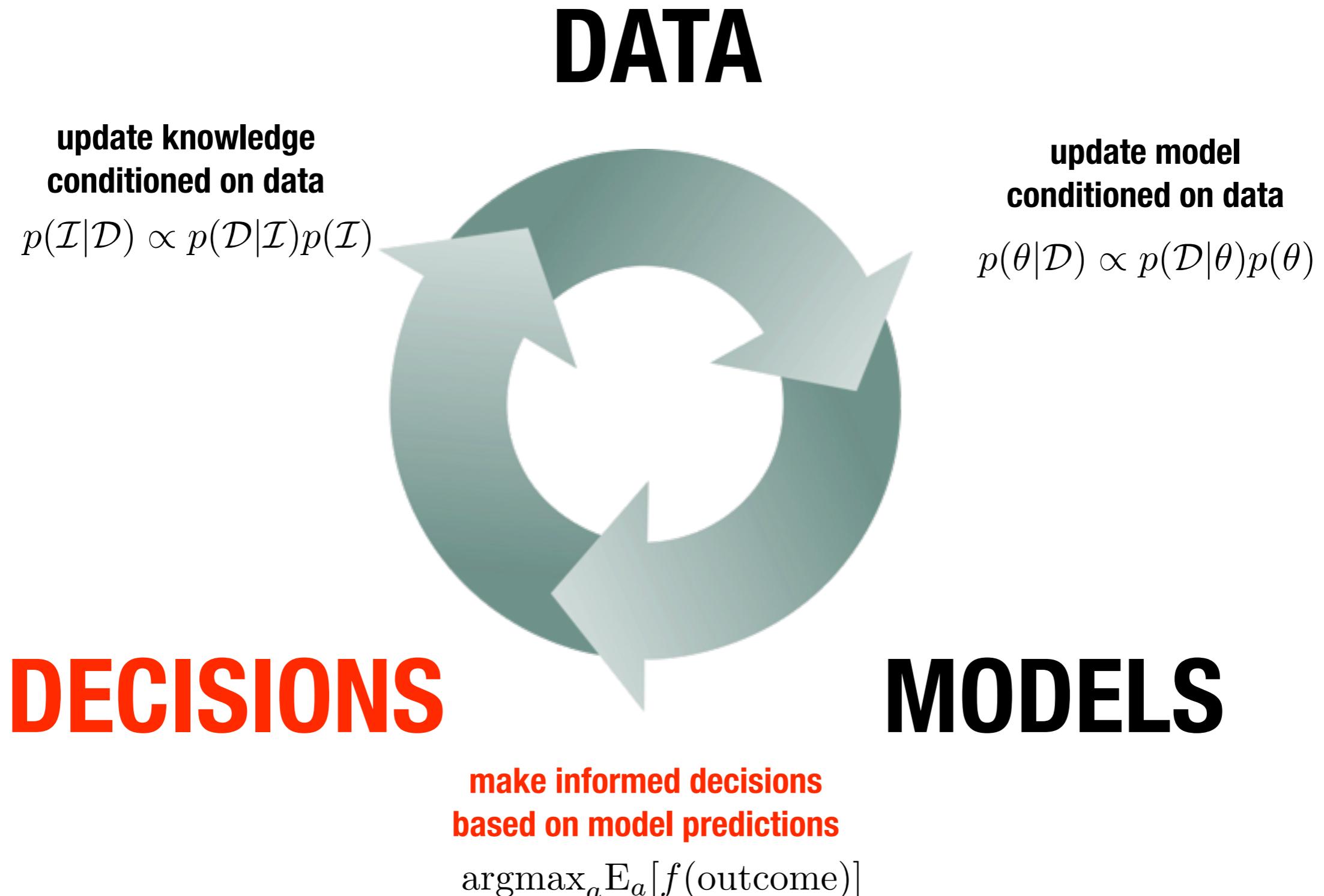
- ✓ Everything is automatic; don't need to tweak things by hand.
- ✓ Data goes in, parameters come out!
- ✓ Never need to choose atom types or perform immense feats of chemical intuition.
Bayesian model selection with RJMC!
- ✓ If we're uncertain about which functional forms are best, automatically chooses.
Bayesian model selection with RJMC!
- ✓ If we find ourselves in uncharted territory in chemical space, throw in more data.
(Maybe it could give us hints about which data would be useful to obtain?)
Bayesian updating and experimental design!
- ✓ Would be nice if parameterization procedure didn't require me to learn crazy new mathematics or algorithms (mainly because I'm lazy)
Isomorphic with statistical mechanics!
- ✓ Would give us an idea of how reliable it new predictions are expected to be.
Multiple parameter sets from Bayesian posterior

The future of forcefield parameterization?



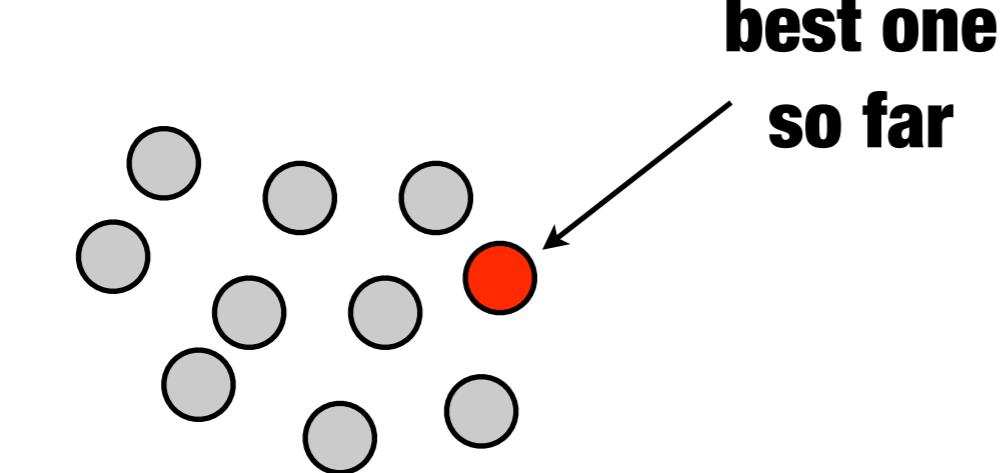
The universal cycle of progress

Bayesian inference can drive progress



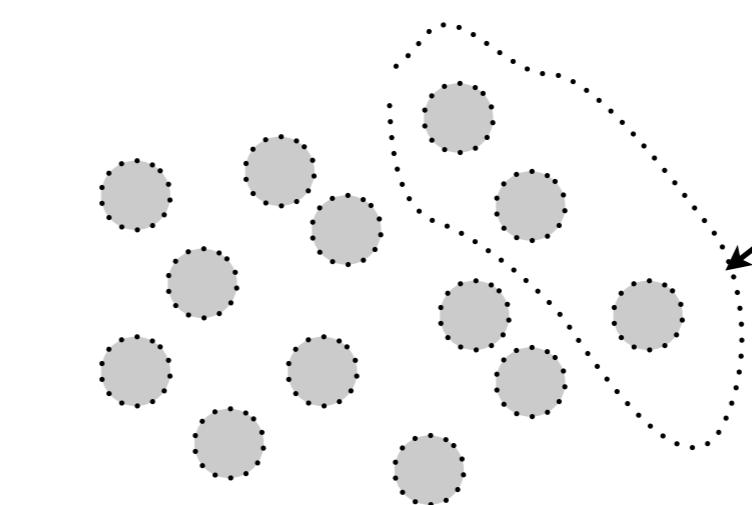
“Pick a winner”

Suppose we make predictions for 20 compounds the chemists can make.
They don't want to make all 20.



**some compounds
we've made**

**best one
so far**



**a bunch of compounds
we could make**

**how many
the chemists want
to make**

What should they make?



Let's pick a simple illustrative model

Simple model for affinity/potency/whatever:

$$\Delta G = \mathbf{x}^T \mathbf{A} \mathbf{x}$$

feature/fingerprint vector $\mathbf{x} = [x_1 \ x_2 \ x_3]^T$

model parameters $\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$ **true model** \mathbf{A}^*

error in measured affinities $\Delta G_{obs} = \Delta G + \xi$

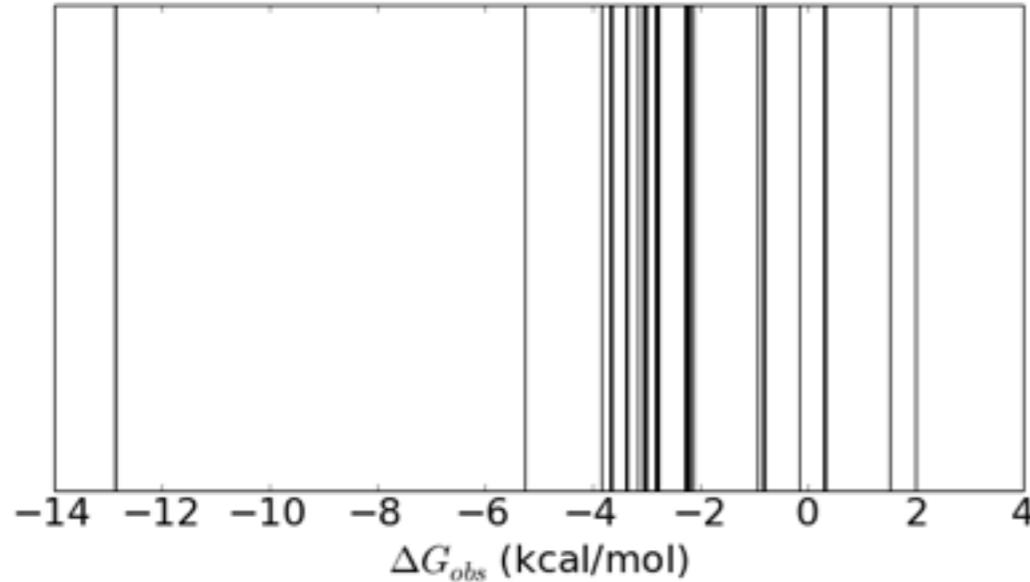
measurement error is Gaussian with zero mean $\xi \in \mathcal{N}(0, \sigma^2)$

σ **measurement error standard deviation**

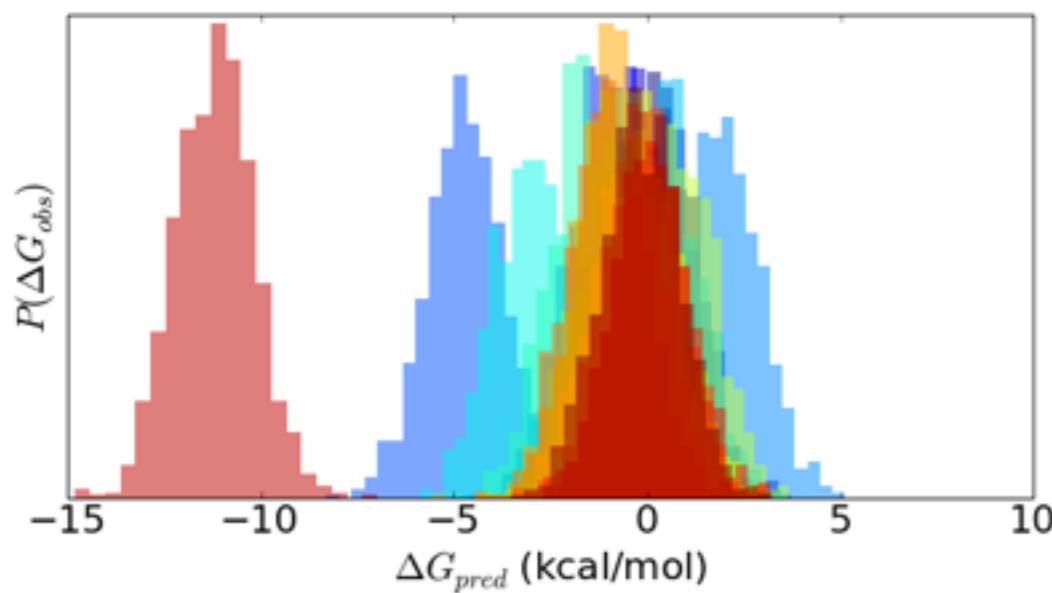
This model is simple, but these principles apply to whatever you want

Suppose we have already made some compounds

Measured affinities for 20 compounds



Predictions for next 20 compounds



Some pymc magic to sample models:

```
import pymc

# Create a pymc model
def make_model(x, DGobs_training):
    ntrain = len(x)

    A = pymc.Normal('A', mu=0.0, tau=1.0, size=(N,N))

    log_sigma = pymc.Uniform('log_sigma', lower=-3, upper=0, value=0.0)
    #pymc.deterministic
    def precision(log_sigma=log_sigma):
        return 1.0 / numpy.exp(log_sigma)

    #pymc.deterministic
    def DGmodel(A=A):
        DG = numpy.zeros([ntrain])
        for i in range(ntrain):
            DG[i] = model(x[i], A)
        return DG

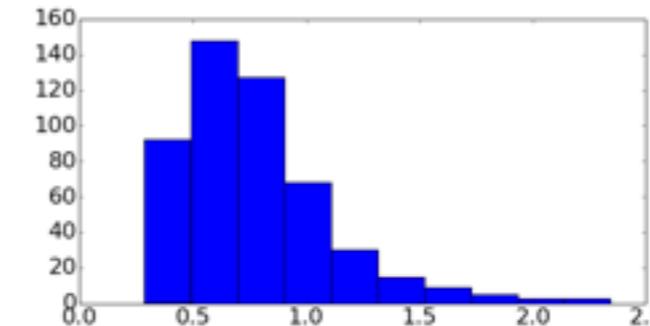
    DGobs = pymc.Normal('DGobs', mu=DGmodel, tau=precision, size=[ntrain], observed=True, value=DGobs_training) # observed data

    # Construct dictionary of model variables.
    pymc_model = { 'A' : A, 'log_sigma' : log_sigma, 'precision' : precision, 'DGmodel' : DGmodel, 'DGobs' : DGobs }

    return pymc_model

pymc_model = pymc.Model(make_model(x_training, DGobs_training))

# Sample with MCMC
mcmc = pymc.MCMC(pymc_model, db='ram', name='Sampler', verbose=True)
mcmc.assign_step_methods()
mcmc.sample(iter=10000, burn=5000, thin=10, progress_bar=False)
```



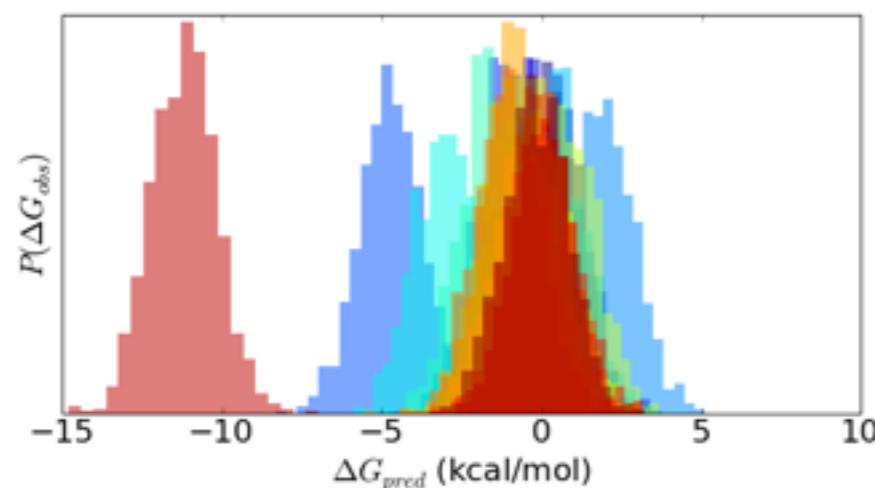
**predicted
experimental
error (actual: 1)**

Which ones should we make?

Code at <https://github.com/choderalab/cup-xiv>

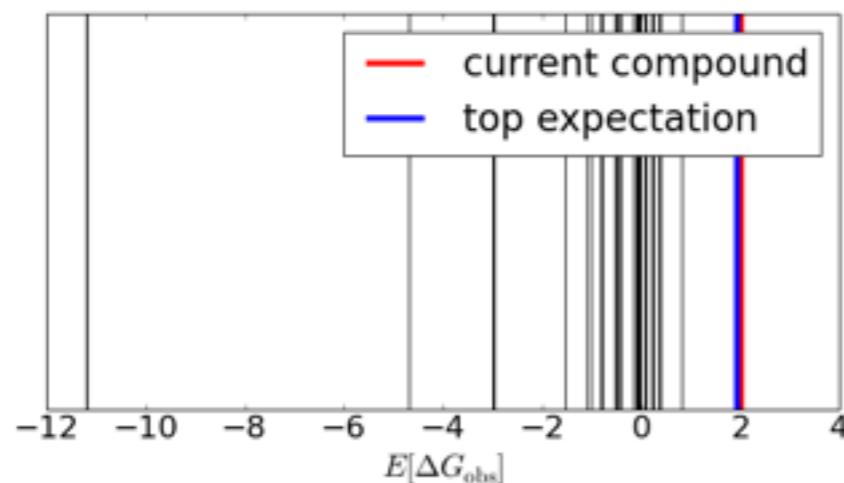
Maximizing the expectation (as in Bayesian bandits)

We could select a compound that **maximizes our expected affinity gain** (just like Bayesian Bandits)



P(one of 20 compounds has affinity gain) = 0.538
P(top predicted expectation has affinity gain) = 0.361

[Not unexpected for a model with 9 parameters fit to data for 10 compounds!]

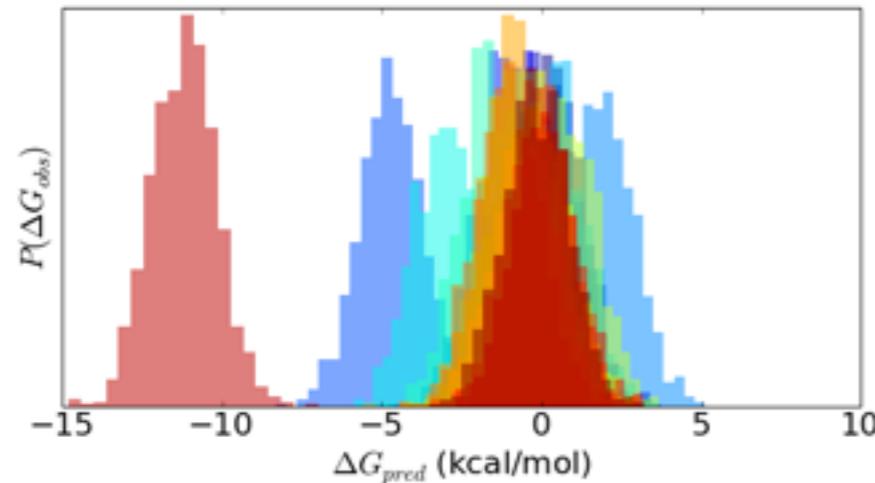


Uh oh. None of the 20 we were given have an expectation higher than the current best compound.

Code at <https://github.com/choderlab/cup-xiv>

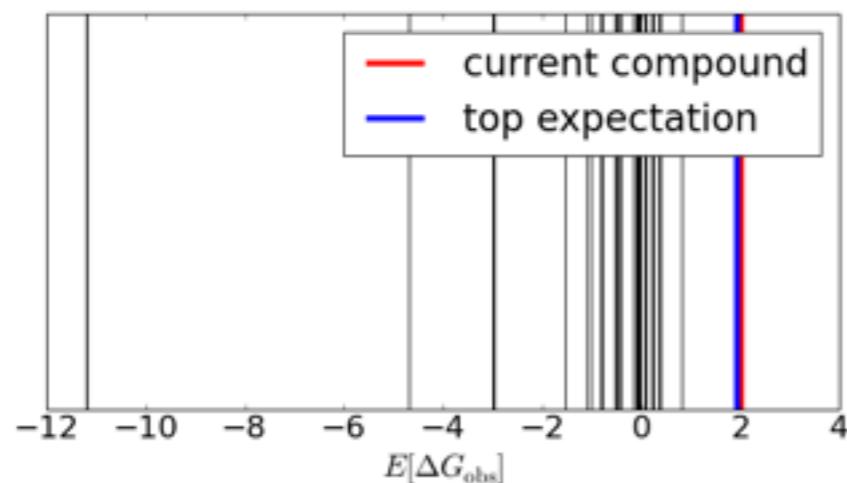
Maximizing the expectation (as in Bayesian bandits)

What if we selected the five compounds that maximize expected affinity gain?



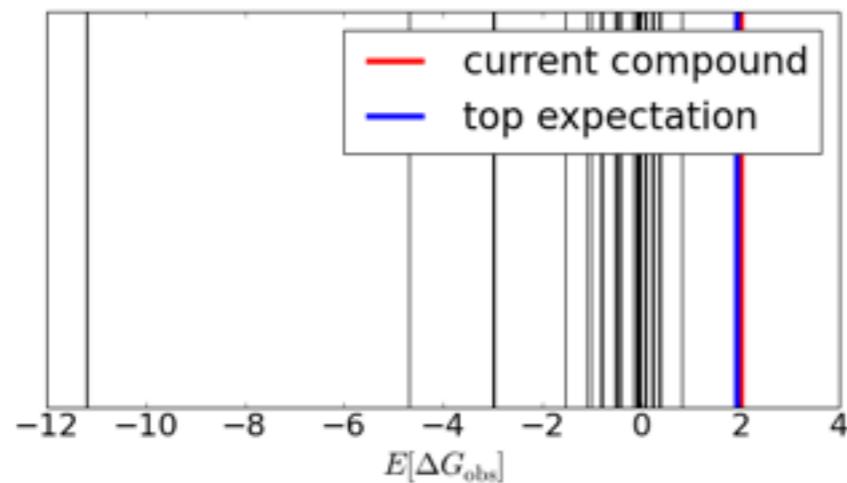
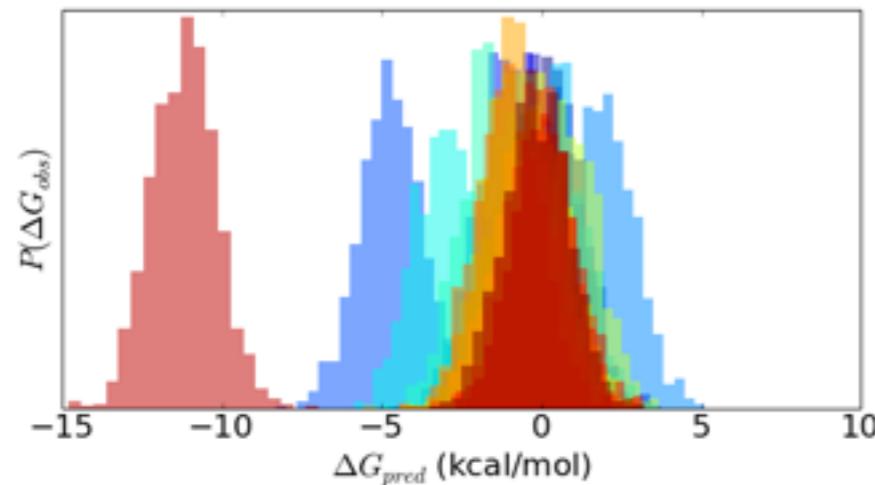
P(one of 20 compounds has affinity gain) = 0.538
P(top predicted expectation has affinity gain) = 0.361
P(affinity gain in top 5) = 0.502

[Doing better!]



Maximizing the expectation (as in Bayesian bandits)

What if we selected the five compounds that minimize probability of failure?



P(one of 20 compounds has affinity gain) = 0.538
P(top predicted expectation has affinity gain) = 0.361
P(affinity gain in top 5) = 0.502
P(affinity gain in “least worst” 5) = 0.539
[A bit better!]

Advanced topics:

Decision-making for the risk-averse
Societal issues often penalize failure much more than rewarding success.
“You only get to be wrong once.”
How can we incorporate risk-aversion into decision-making?

Decision-making for information gain
What experiments will maximize our information gain?

Decision-making with justification
Betting odds can help justify risks taken by decisions

(I really liked the Optibrium talk!)

Acknowledgments

Statistical fisticuffs

Ant Nicholls

Kim Branson

ITC and entropy-enthalpy compensation

**Kim Branson, Sarah Boyce, Paul Novick, Vijay Pande,
David Minh, David Mobley**

Telepresents

**Johnny 5 and his support staff
(Brian Cole, Craig Bruce, Ben Ellingson)**

