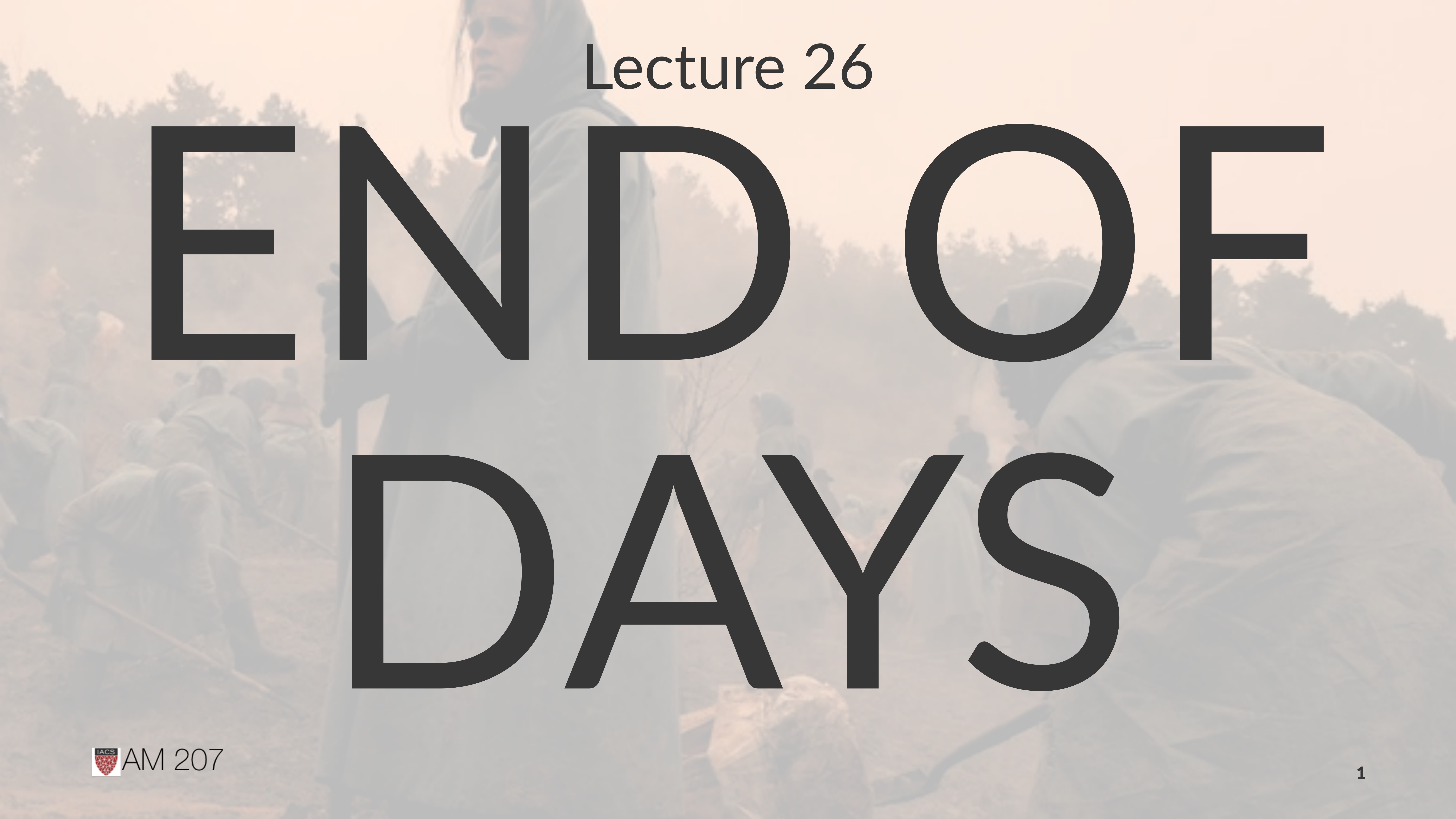


Lecture 26

END OF DAYS

AM 207

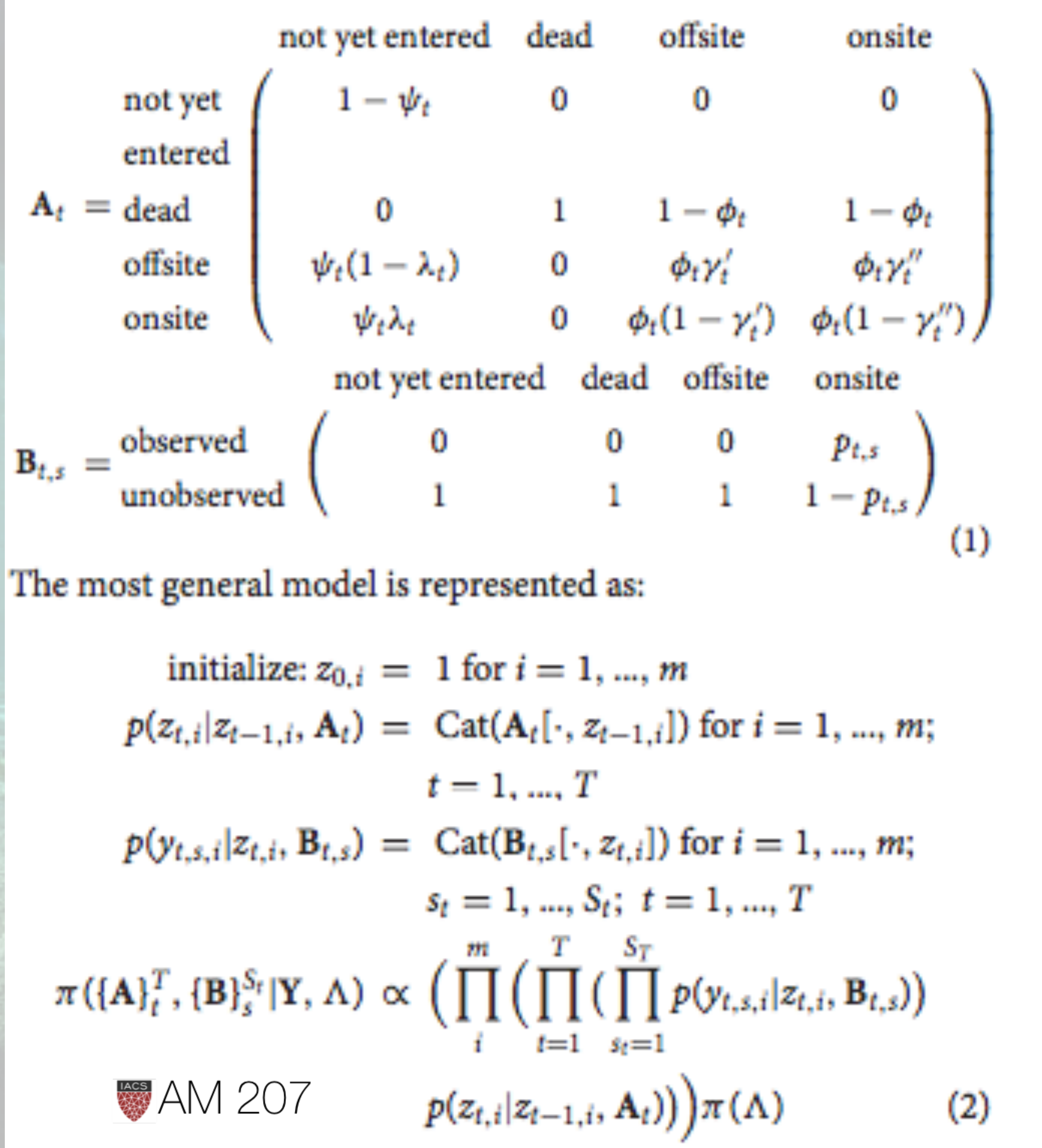
What is this course about?

- **Density Estimation**. (Also called unsupervised or representation learning)
- **Generative Models** in statistics and machine learning..a principled way of modeling (both supervised and unsupervised)
- **Being Bayesian**: a self-consistent process to carry out this modeling
- **Sampling and stochastic optimization**: the technology needed

Along the way we

- learn how to regularize models
- deal with data computationally large/small and statistically small/large
- learn how to optimize objective functions such as loss functions using Stochastic Gradient Descent and Simulated annealing
- Perform sampling and MCMC to solve a variety of problems
- Learn how to use interpretably parametric, non-interpretably parametric, and non-parametric methods

Bayesian Hierarchical Mark-Recapture Models (see https://www.frontiersin.org/articles/10.3389/fmars.2016.00025/full)

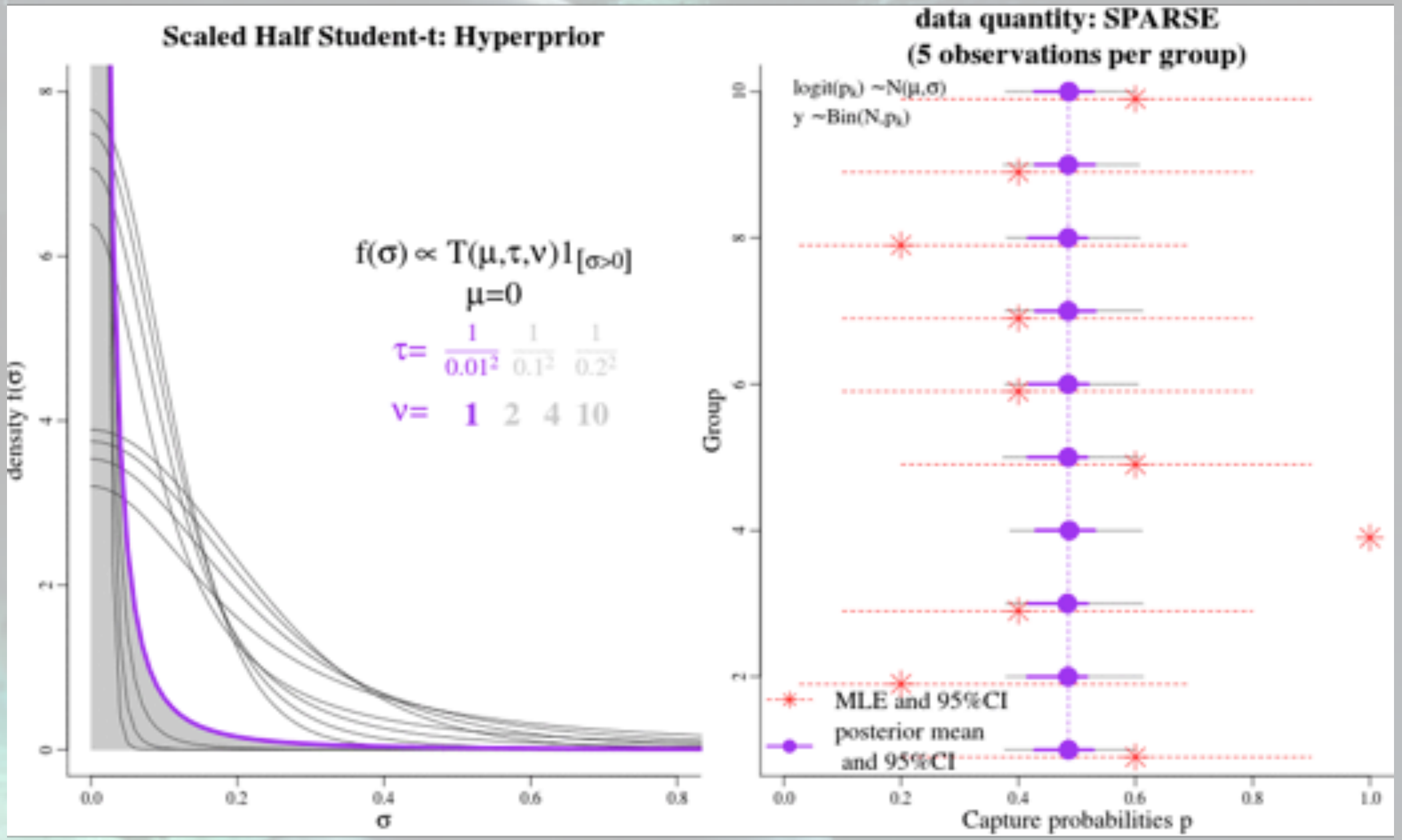


$$\mathbf{A}_t = \begin{array}{c|cccc} & \text{not yet entered} & \text{dead} & \text{offsite} & \text{onsite} \\ \hline \text{not yet entered} & 1-\psi_t & 0 & 0 & 0 \\ \text{dead} & 0 & 1 & 1-\phi_t & 1-\phi_t \\ \text{offsite} & \psi_t(1-\lambda_t) & 0 & \phi_t\gamma_t' & \phi_t\gamma_t'' \\ \text{onsite} & \psi_t\lambda_t & 0 & \phi_t(1-\gamma_t') & \phi_t(1-\gamma_t'') \end{array}$$

$$\mathbf{B}_{t,s} = \begin{array}{c|cccc} & \text{not yet entered} & \text{dead} & \text{offsite} & \text{onsite} \\ \hline \text{observed} & 0 & 0 & 0 & p_{t,s} \\ \text{unobserved} & 1 & 1 & 1 & 1-p_{t,s} \end{array} \qquad (1)$$

The most general model is represented as:

$$\begin{aligned} \text{initialize: } z_{0,i} &= 1 \text{ for } i = 1, \ldots, m \\ p(z_{t,i}|z_{t-1,i}, \mathbf{A}_t) &= \text{Cat}(\mathbf{A}_t[\cdot, z_{t-1,i}]) \text{ for } i = 1, \ldots, m; \\ & \quad t = 1, \ldots, T \\ p(y_{t,s,i}|z_{t,i}, \mathbf{B}_{t,s}) &= \text{Cat}(\mathbf{B}_{t,s}[\cdot, z_{t,i}]) \text{ for } i = 1, \ldots, m; \\ & \quad s_t = 1, \ldots, S_t;\ t = 1, \ldots, T \\ \pi(\{\mathbf{A}\}_t^T, \{\mathbf{B}\}_s^{S_t}|\mathbf{Y}, \Lambda) &\propto \Big(\prod_i^m \Big(\prod_{t=1}^T \Big(\prod_{s_t=1}^{S_T} p(y_{t,s,i}|z_{t,i}, \mathbf{B}_{t,s})\Big) \\ & \quad p(z_{t,i}|z_{t-1,i}, \mathbf{A}_t)\Big)\Big)\pi(\Lambda) \qquad (2) \end{aligned}$$



Concepts running through:

Hidden Variables, marginalized

Testing, testing, testing

Differentiation vs Integration

Frequentist vs Bayesian

Generative Models

Outline of the course summary

1. The Nature of learning
2. Frequentist stats and machine learning
3. Stochastic optimization
4. Sampling and MCMC: Metropolis to HMC and NUTS
5. Bayesian Stats
6. Hierarchical Modeling
7. Supervised Learning: Regression, GLMs and GPs
8. Model Checking
9. Model comparison and ensembling
10. Generative Models (adding latent variables which are "not parameters")
11. ELBO grease and Variational Inference

THE NATURE OF LEARNING

SMALL WORLD vs BIG WORLD

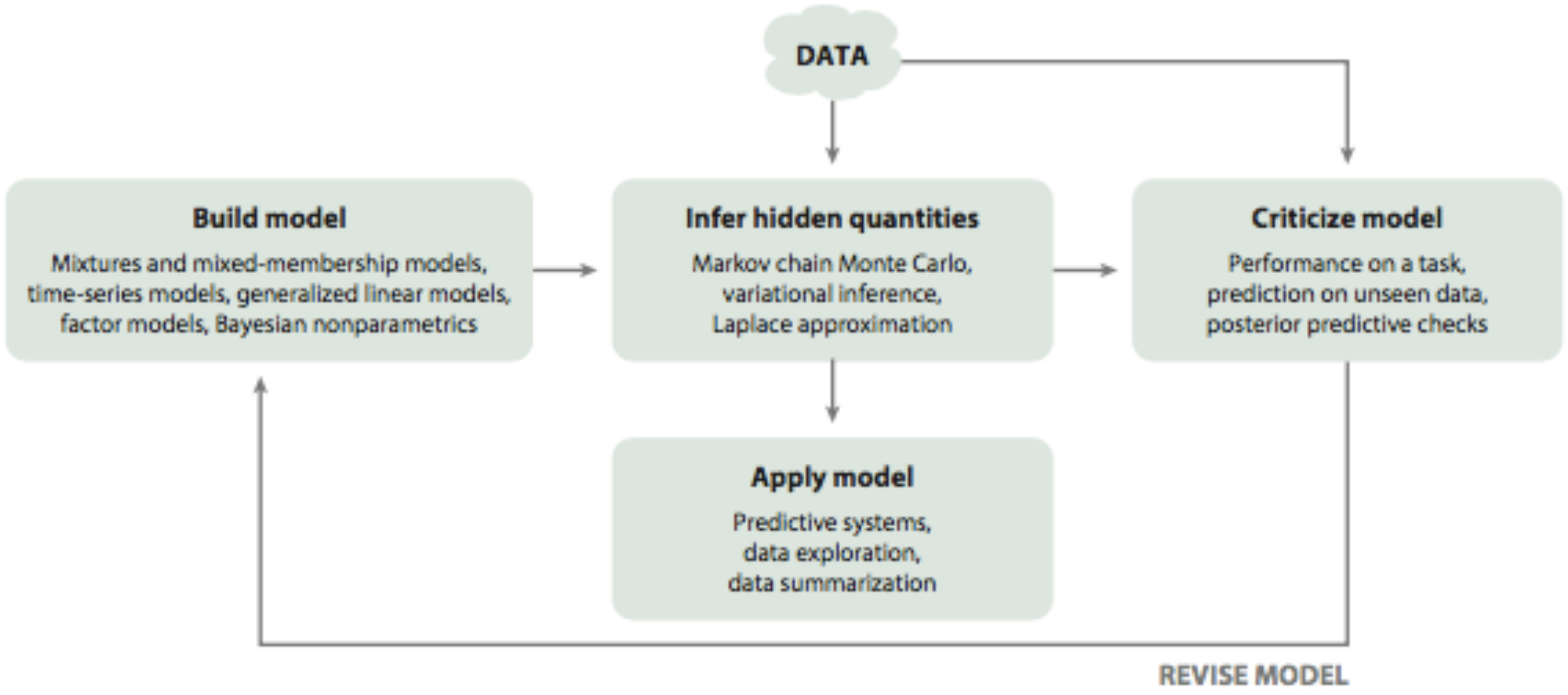
Small world:

$$P(\theta \mid D) = \frac{P(D \mid \theta) \times P(\theta)}{P(D)}$$

Big World:

$$P(M \mid D) = \frac{P(D \mid M) \times P(M)}{P(D)}$$

Box's loop



DATA

Build model

Mixtures and mixed-membership models, time-series models, generalized linear models, factor models, Bayesian nonparametrics

Infer hidden quantities

Markov chain Monte Carlo, variational inference, Laplace approximation

Criticize model

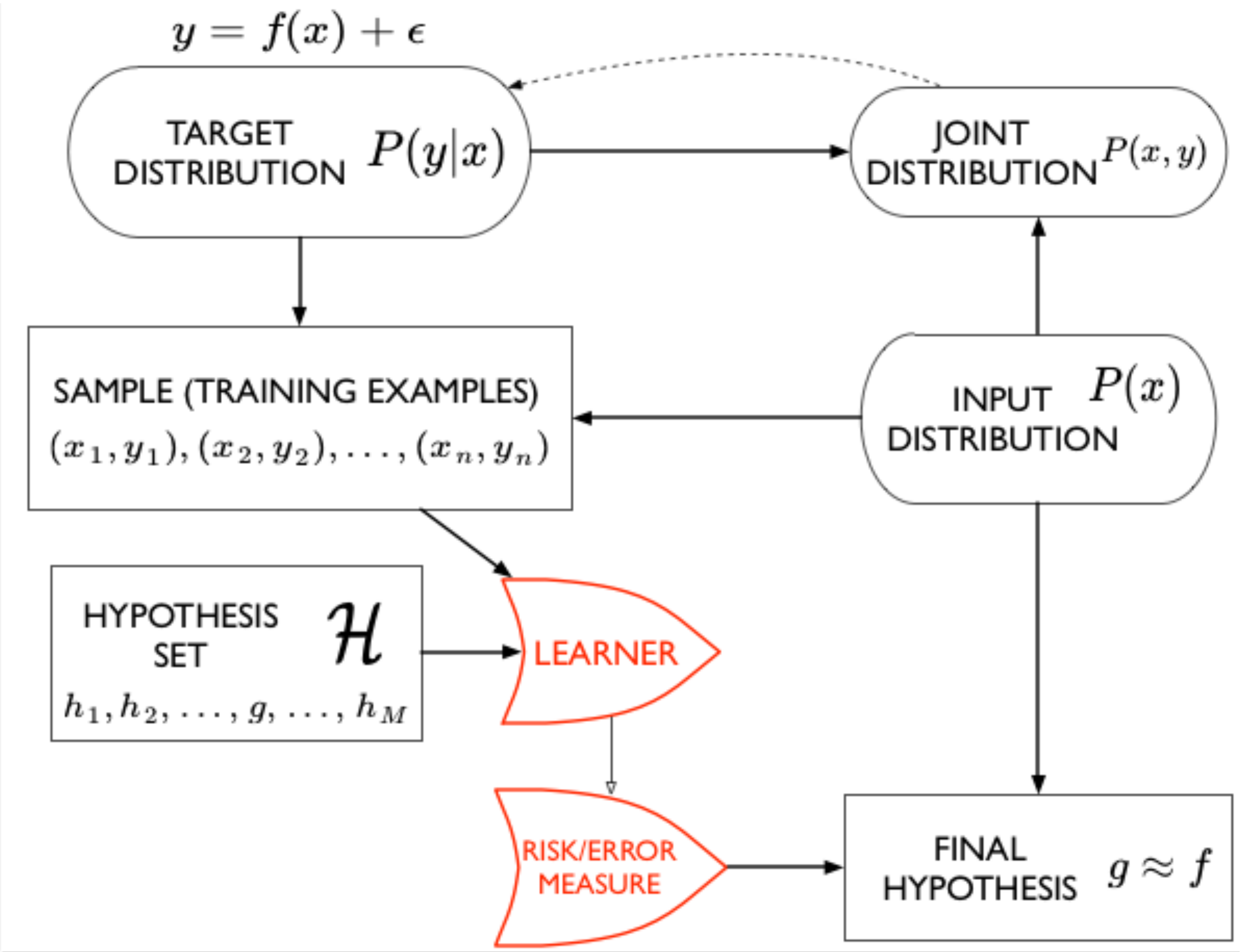
Performance on a task, prediction on unseen data, posterior predictive checks

Apply model

Predictive systems, data exploration, data summarization

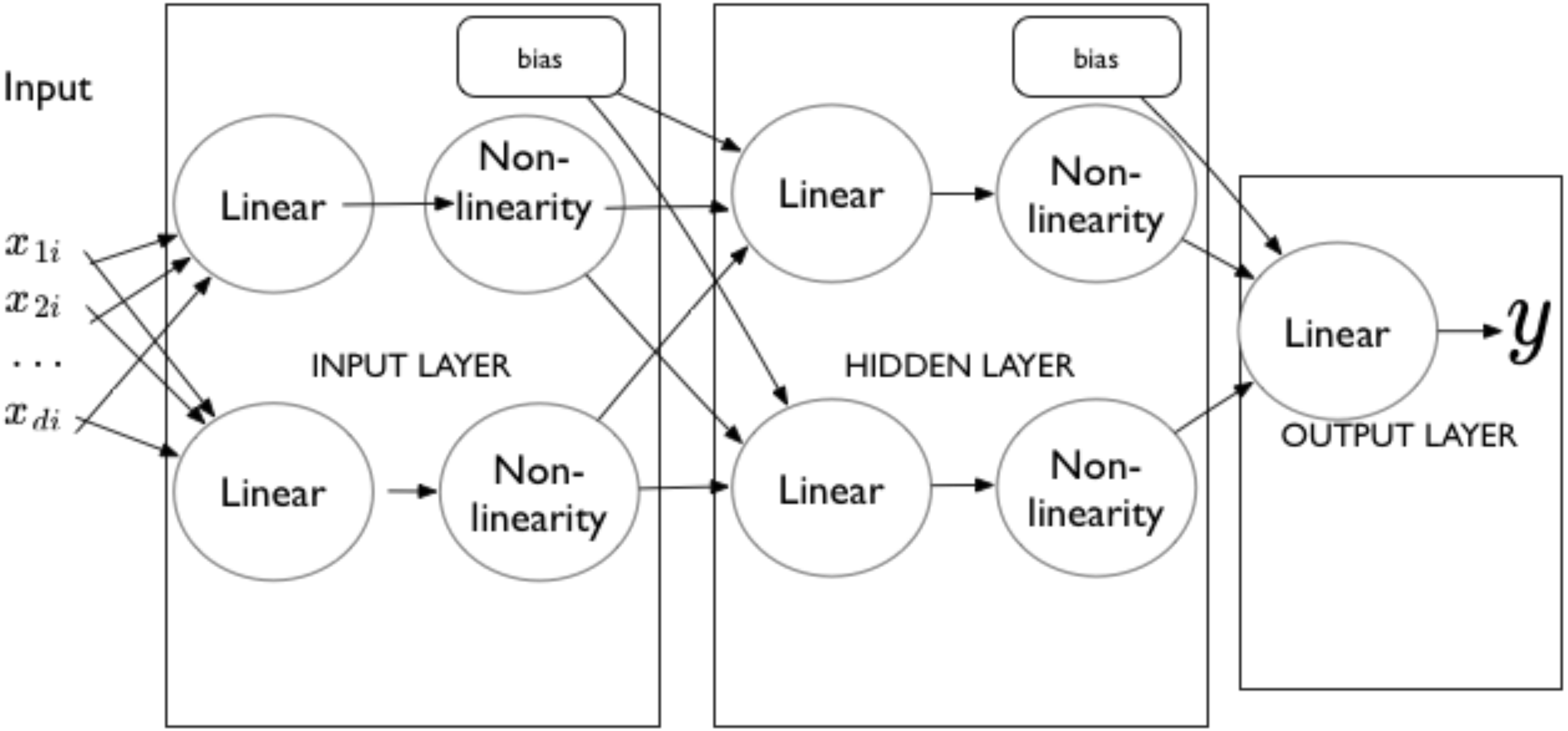
REVISE MODEL

The nature of learning via predictives



$y = f(x) + \epsilon$
TARGET DISTRIBUTION $P(y|x)$
JOINT DISTRIBUTION $P(x,y)$
SAMPLE (TRAINING EXAMPLES)
$(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$
INPUT DISTRIBUTION $P(x)$
HYPOTHESIS SET $\mathcal{H}$
$h_1, h_2, \ldots, g, \ldots, h_M$
LEARNER
RISK/ERROR MEASURE
FINAL HYPOTHESIS $g \approx f$

Interpretable vs Nets



Universal Approximation

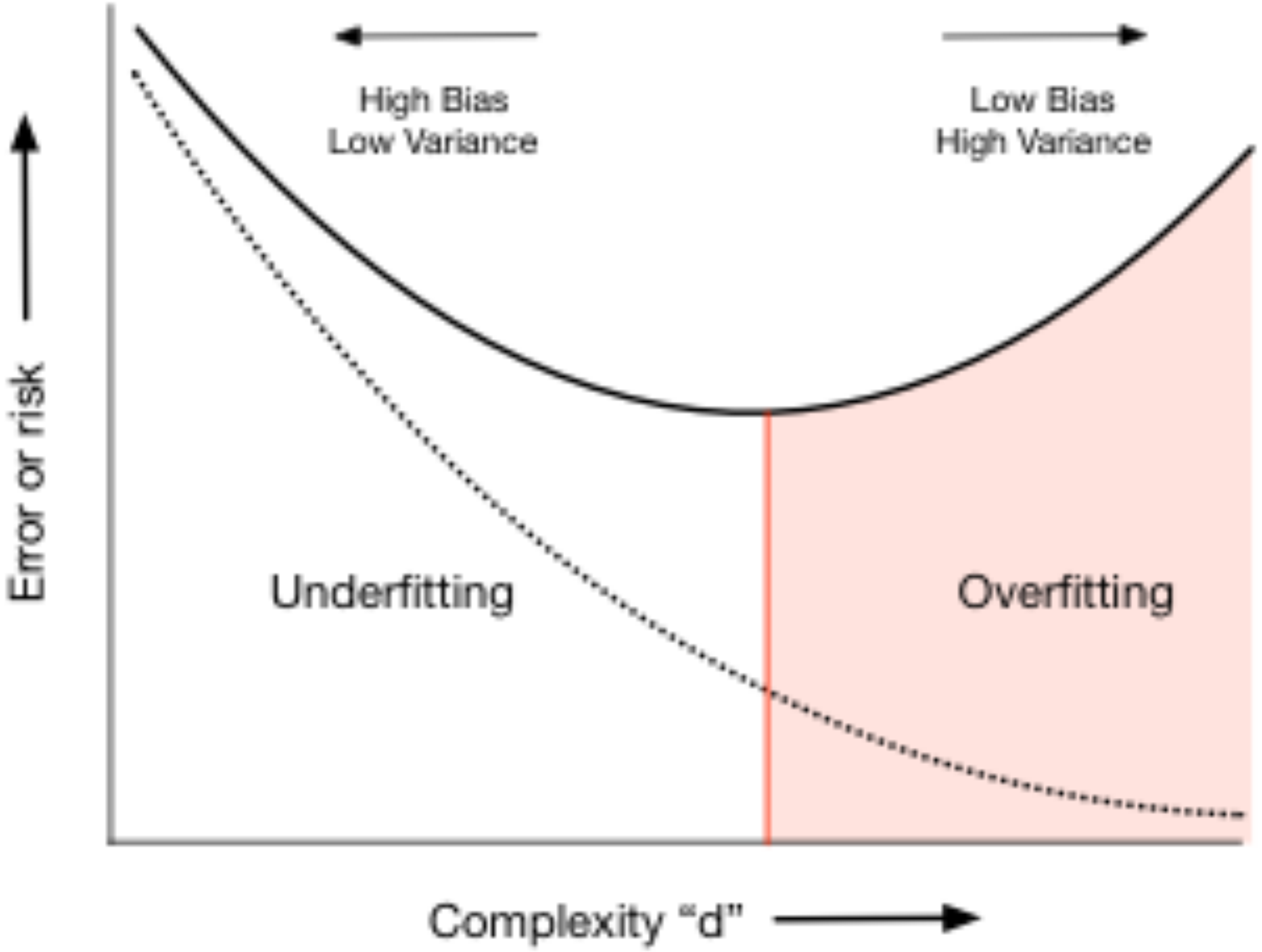
- any one hidden layer net can approximate any continuous function with finite support, with appropriate choice of nonlinearity

- under appropriate conditions, all of sigmoid, tanh, RELU can work

- but may need lots of units

- and will learn the function it thinks the data has, not what you

Dont Overfit



KL-Divergence: compare box to nature

$$D_{KL}(p, q) = E_p[log(p) - log(q)] = E_p[log(p/q)]$$
$$= \sum_i p_i log(\frac{p_i}{q_i}) \ or \int dPlog(\frac{p}{q})$$

$$D_{KL}(p, p) = 0$$

KL divergence measures distance/dissimilarity of the two distributions p(x) and q(x).

- used for VI, EM, a probabilistic loss function

FREQUENTIST STATS MACHINE LEARNING

Law of Large numbers, LOTUS, MC

Let $x_1, x_2, \ldots, x_n$ be a sequence of IID values from random variable X, which has finite mean μ. Let:

$$S_n = \frac{1}{n}\sum_{i=1}^{n} x_i, \text{ then } S_n \to \mu \, as \, n \to \infty.$$

- Expectations become sample averages. Convergence for large N.

$$E_f[g] = \int g(x)dF = \int g(x)f(x)dx = \lim_{n\to\infty} \frac{1}{N}\sum_{x_i \sim f} g(x_i)$$

- allows for monte-carlo

The Central Limit Theorem (CLT)

Let $x_1, x_2, \ldots, x_n$ be a sequence of IID values from a random variable X. Suppose that X has the finite mean μ AND finite variance σ^2. Then:

$$S_n = \frac{1}{n}\sum_{i=1}^{n} x_i, \text{ converges to}$$

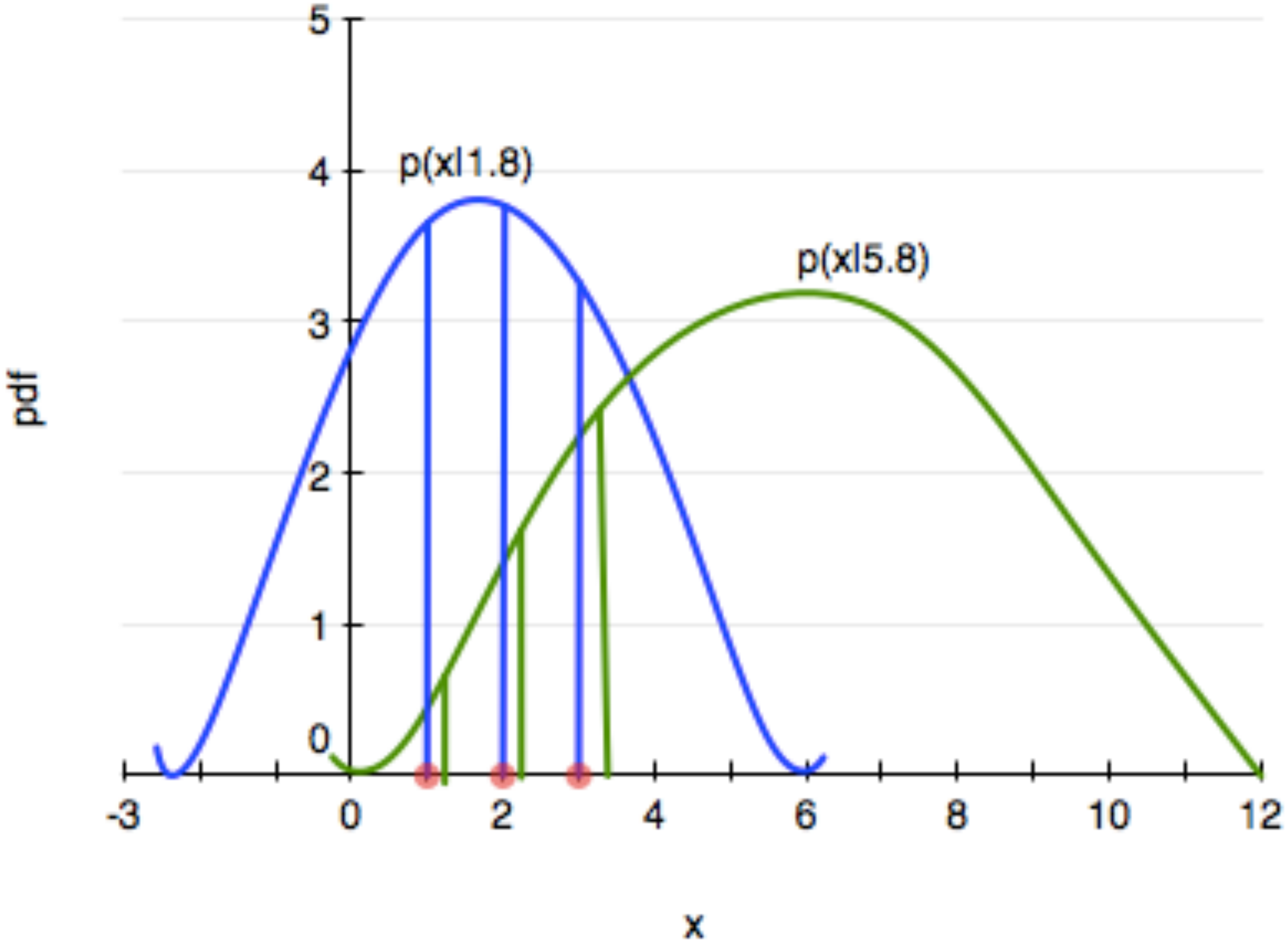
$$S_n \sim N(\mu, \frac{\sigma^2}{n})\, as\, n \rightarrow \infty.$$

Frequentist Statistics

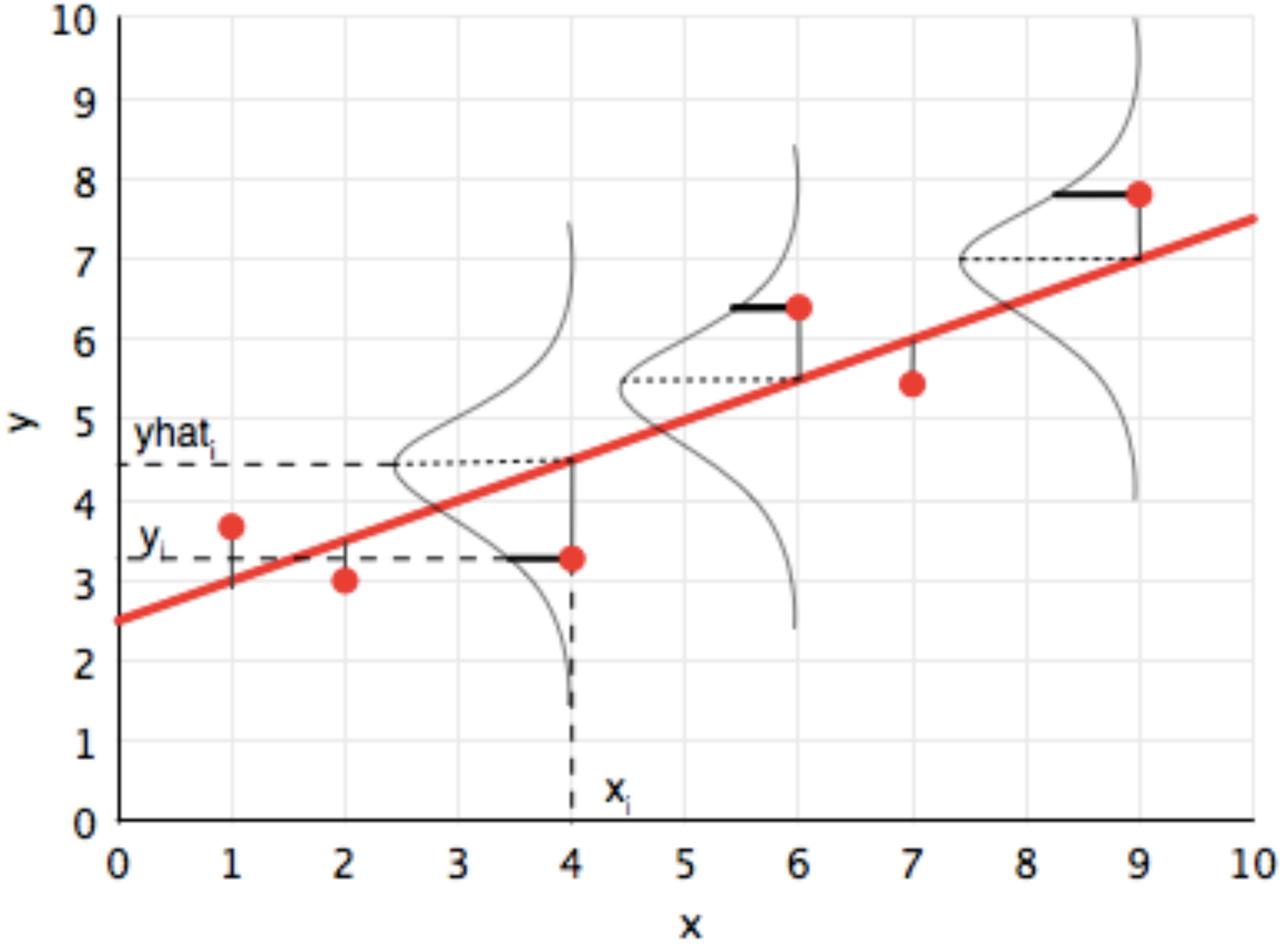
"data is a **sample** from an existing **population**"

- data is stochastic, variable; parameters fixed
- apply an estimator F to the sample data D, so $\hat{\mu} = F(D)$.
- If your model describes the true generating process for the data (not mis-specified), then there is some true μ^*.
- The best we can do is to estimate $\hat{\mu}$.

MLE



p(x|1.8)
p(x|5.8)
pdf
5
4
3
2
1
0
-3
0
2
4
6
8
10
12
x



yhat$_i$
y$_i$
x$_i$
y
10
9
8
7
6
5
4
3
2
1
0
0
1
2
3
4
5
6
7
8
9
10
x

Information Entropy and MAXENT

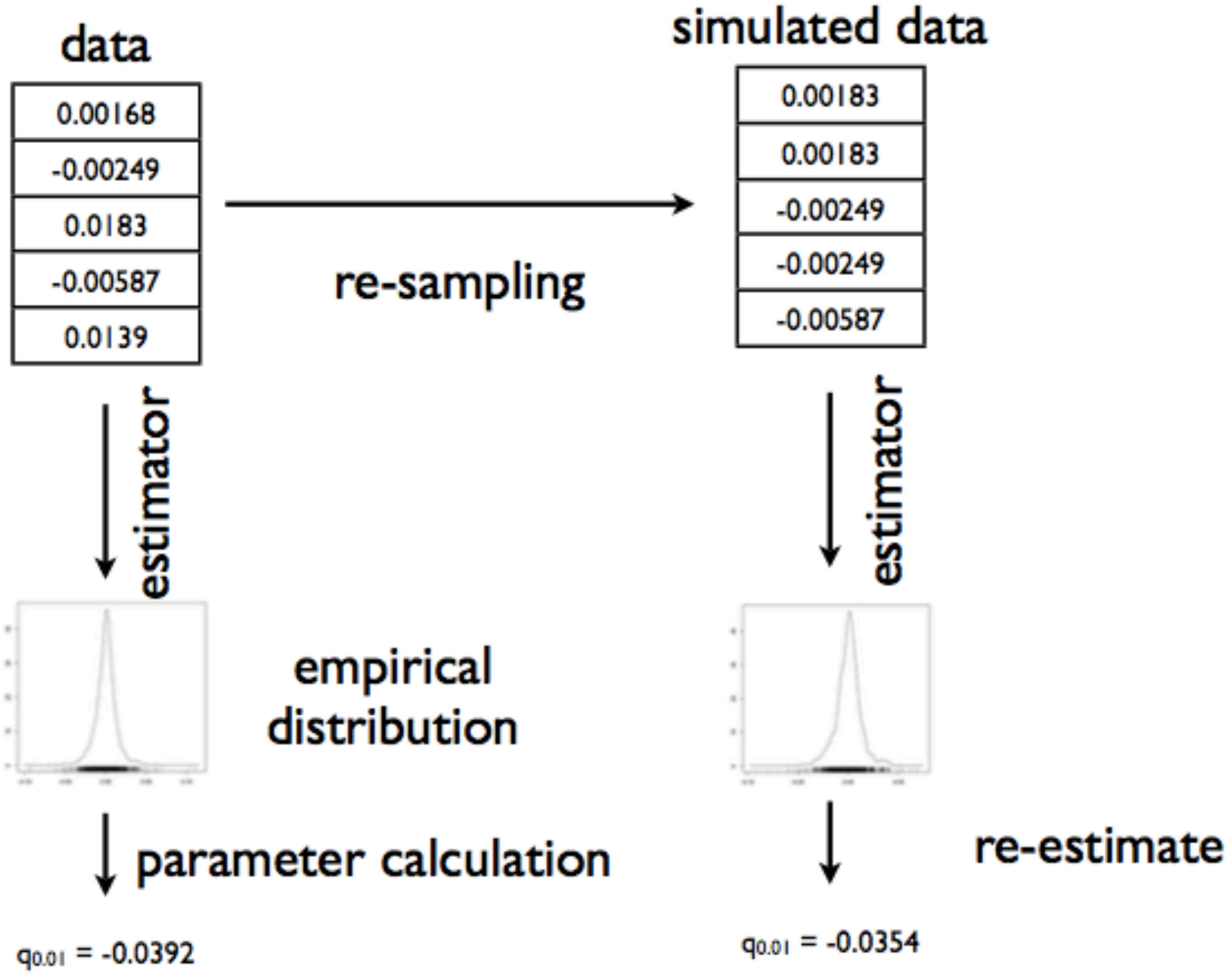
$$H(p) = -E_p[log(p)] = -\int p(x)log(p(x))dx \;\; OR \; -\sum_i p_i log(p_i)$$

- what would be the least surprising distribution, the one with the least additional assumptions (most conservative), the one that can happen in the most ways consistent with constraints

- most common distributions used as likelihoods (and priors) are in the exponential family, MAXENT subject to different constraints.

Sampling Distribution, bootstrap

- M data sets **drawn** from the population, thus M estimates
- As we let $M \to \infty$, the distribution induced on $\hat{\mu}$ is the empirical **sampling distribution of the estimator**.
- create data sets by BOOTSTRAP
- but we need samples



data

0.00168

-0.00249

0.0183

-0.00587

0.0139

re-sampling

simulated data

0.00183

0.00183

-0.00249

-0.00249

-0.00587

estimator

estimator

empirical distribution

parameter calculation

re-estimate

$q_{0.01}$ = -0.0392

$q_{0.01}$ = -0.0354

SAMPLE vs POPULATION

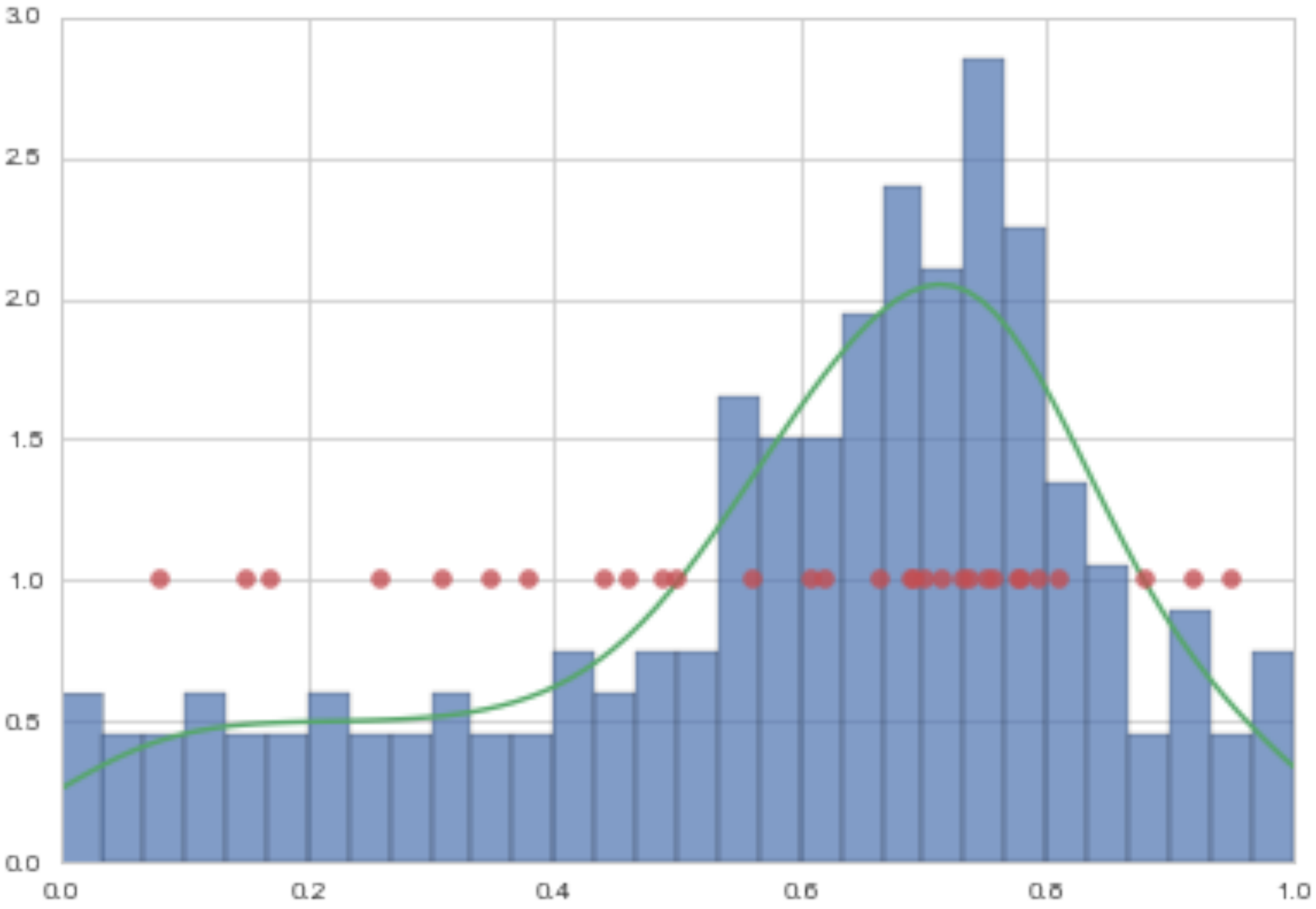
Want: $R_{out}(h) = E_{p(x)}[(h(x) - f(x))^2] = \int dx p(x)(h(x) - f(x))^2$

LLN:

$$R_{out}(h) = \lim_{n \to \infty} \frac{1}{n} \sum_{x_i \sim p(x)} (h(x_i) - f(x_i))^2 = \lim_{n \to \infty} \frac{1}{n} \sum_{x_i \sim p(x)} (h(x_i) - y_i)^2$$

$\mathcal{D}$ representative ($\mathcal{D} \sim p(x)$) $\implies \mathcal{R}_{\mathcal{D}}(h) = \sum_{x_i \in \mathcal{D}} (h(x_i) - y_i)^2$

Statement of the Learning Problem

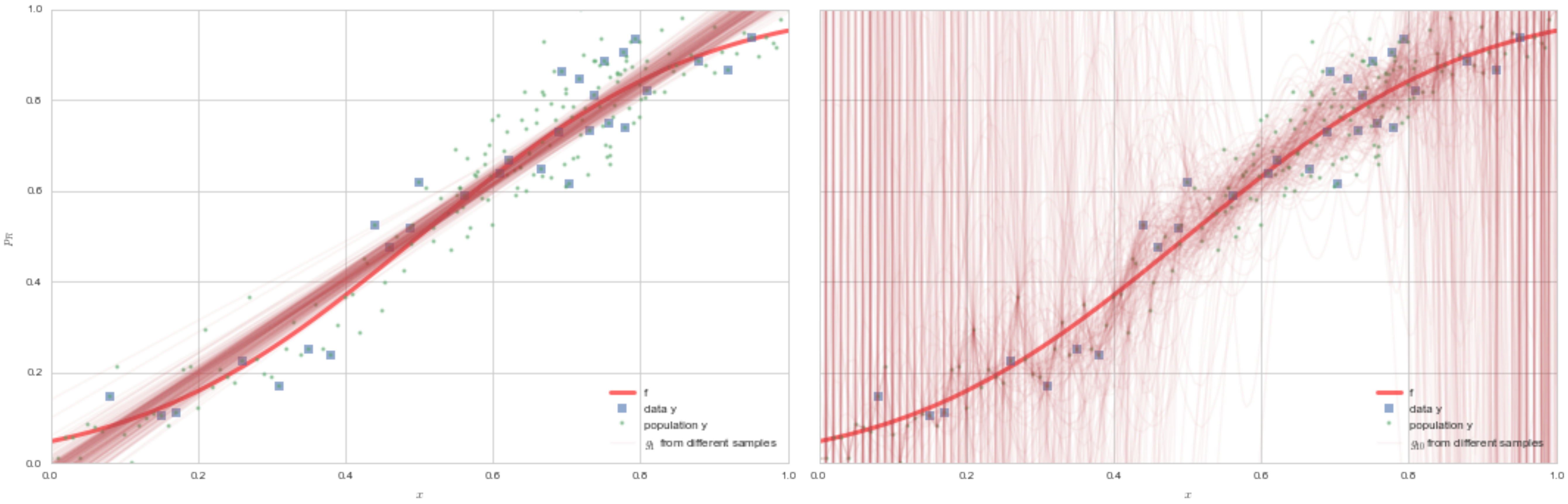


The sample must be representative of the population!

$$A: R_{\mathcal{D}}(g)\ \ smallest\, on\, \mathcal{H}$$
$$B: R_{out}(g) \approx R_{\mathcal{D}}(g)$$

A: Empirical risk estimates in-sample risk.
B: Thus the out of sample risk is also small.

UNDERFITTING (Bias)
vs OVERFITTING (Variance)



$$\langle R \rangle = E_{\mathcal{D}}[R_{out}(g_{\mathcal{D}})] = E_{\mathcal{D}} E_{p(x)}[(g_{\mathcal{D}}(x) - f(x) - \epsilon)^2]$$

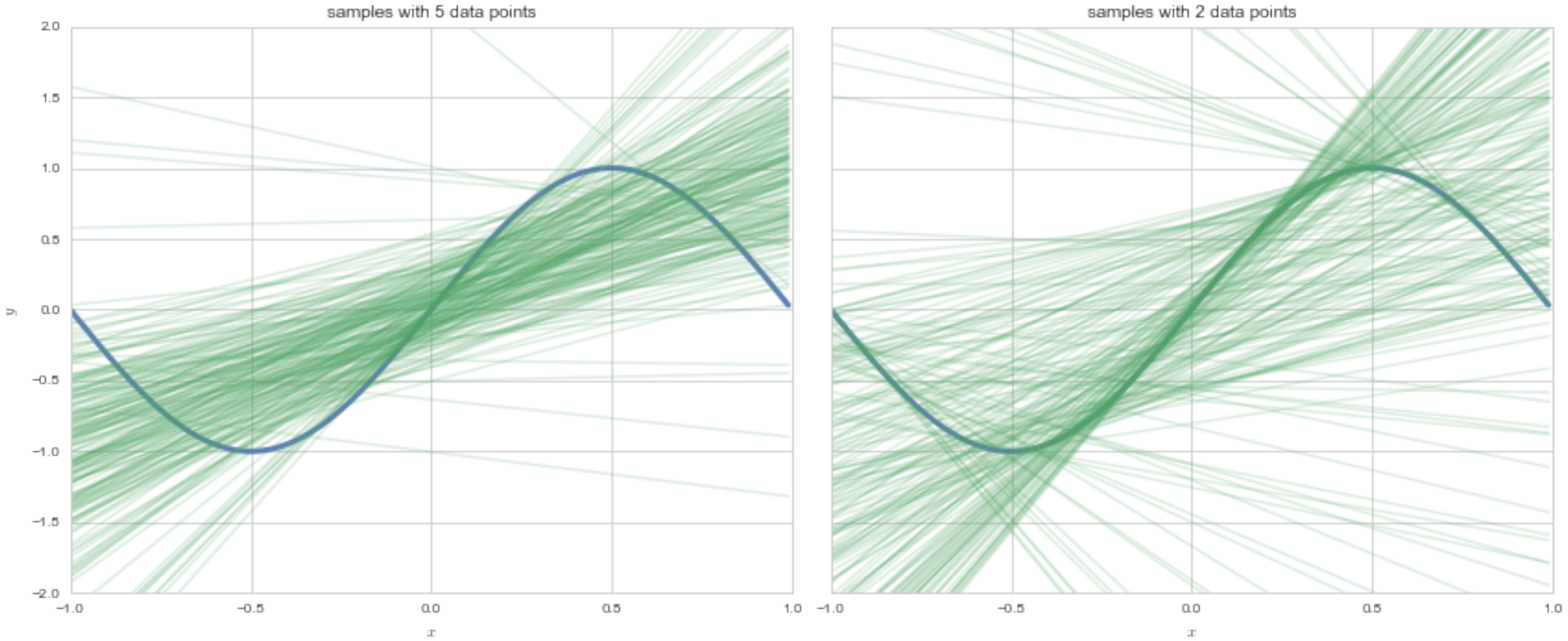
$$\bar{g} = E_{\mathcal{D}}[g_{\mathcal{D}}] = (1/M) \sum_{\mathcal{D}} g_{\mathcal{D}}$$

Then,

$$\langle R \rangle = E_{p(x)}[E_{\mathcal{D}}[(g_{\mathcal{D}} - \bar{g})^2]] + E_{p(x)}[(f - \bar{g})^2] + \sigma^2$$

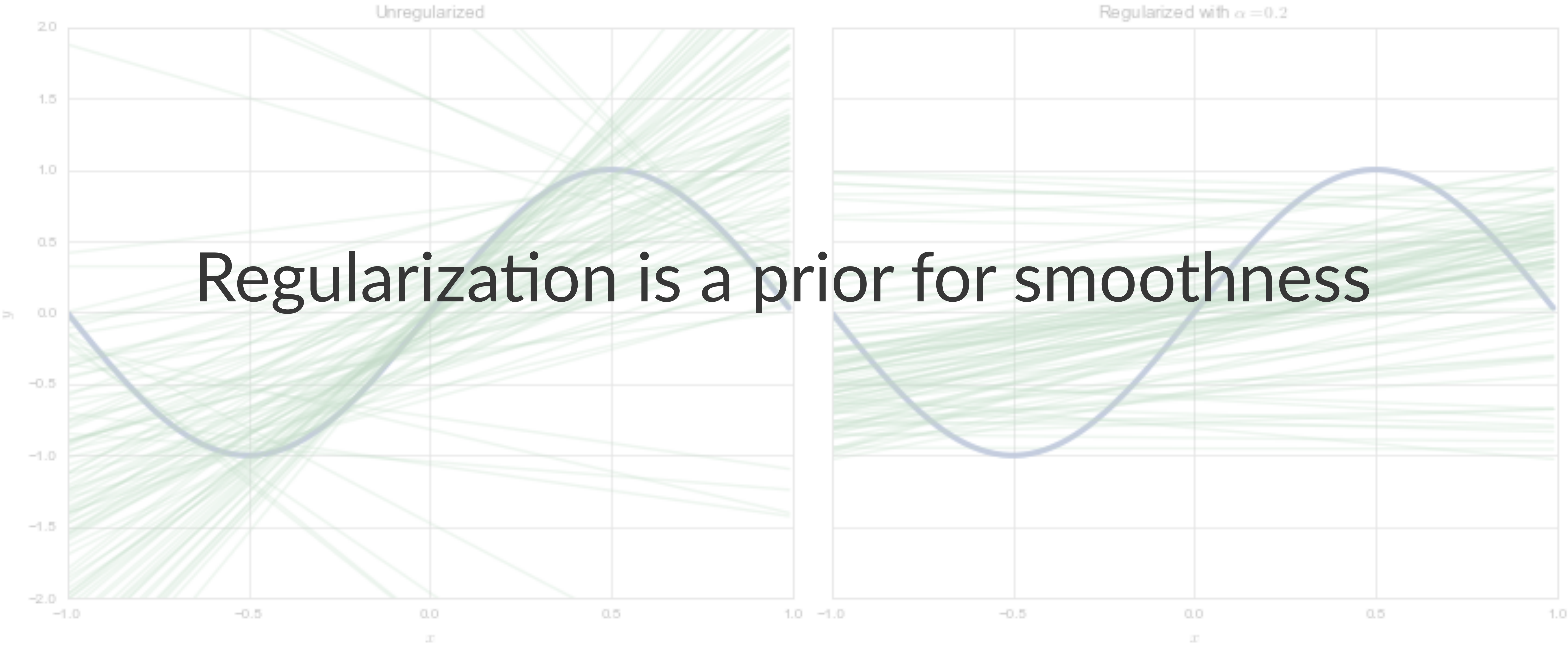
This is the bias variance decomposition for regression.

DATA SIZE MATTERS: straight line fits to a sine curve



Corollary: Must fit simpler models to less data!

Regularization is a prior for smoothness



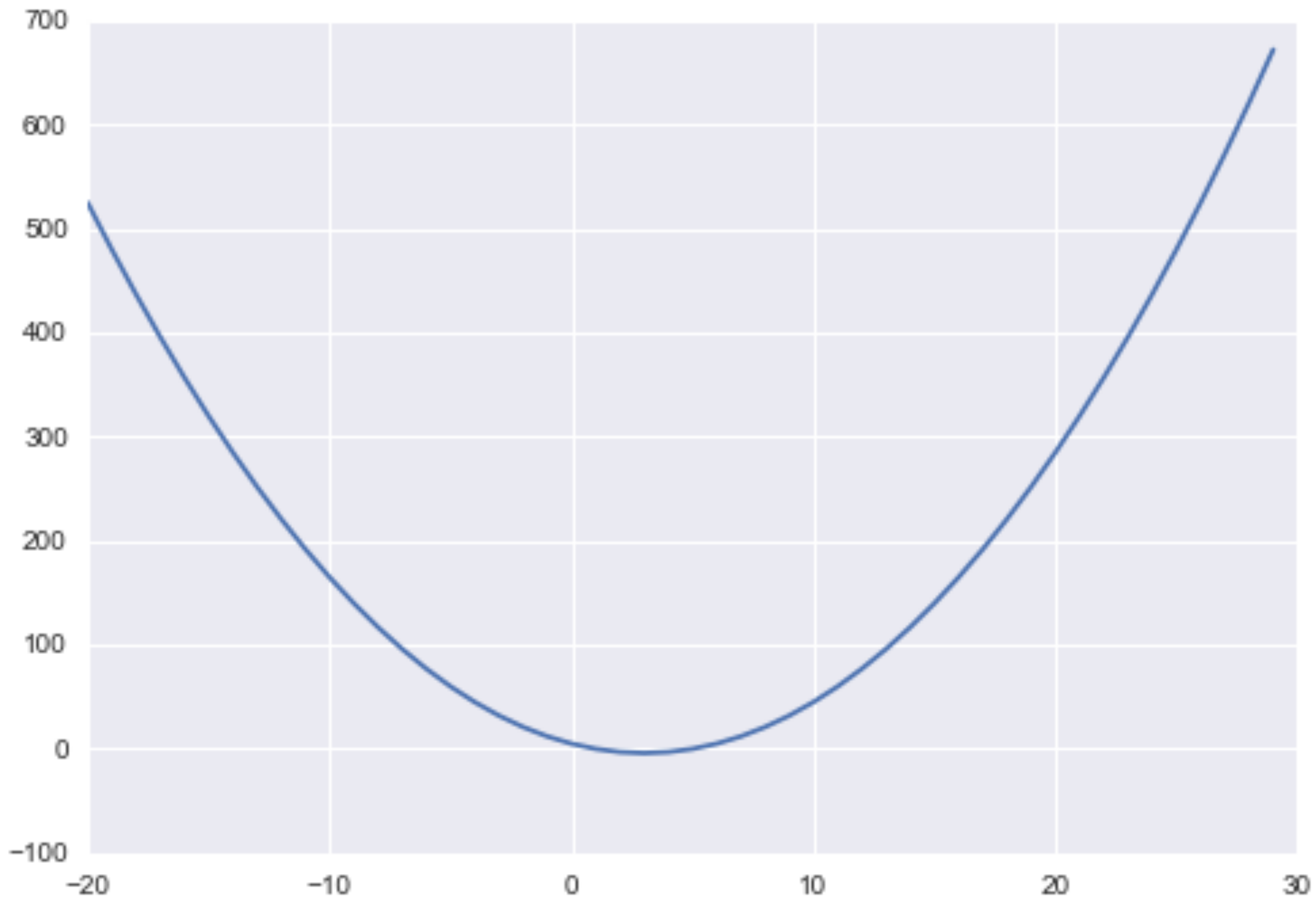
STOCHASTIC OPTIMIZATION

Gradient ascent (descent)

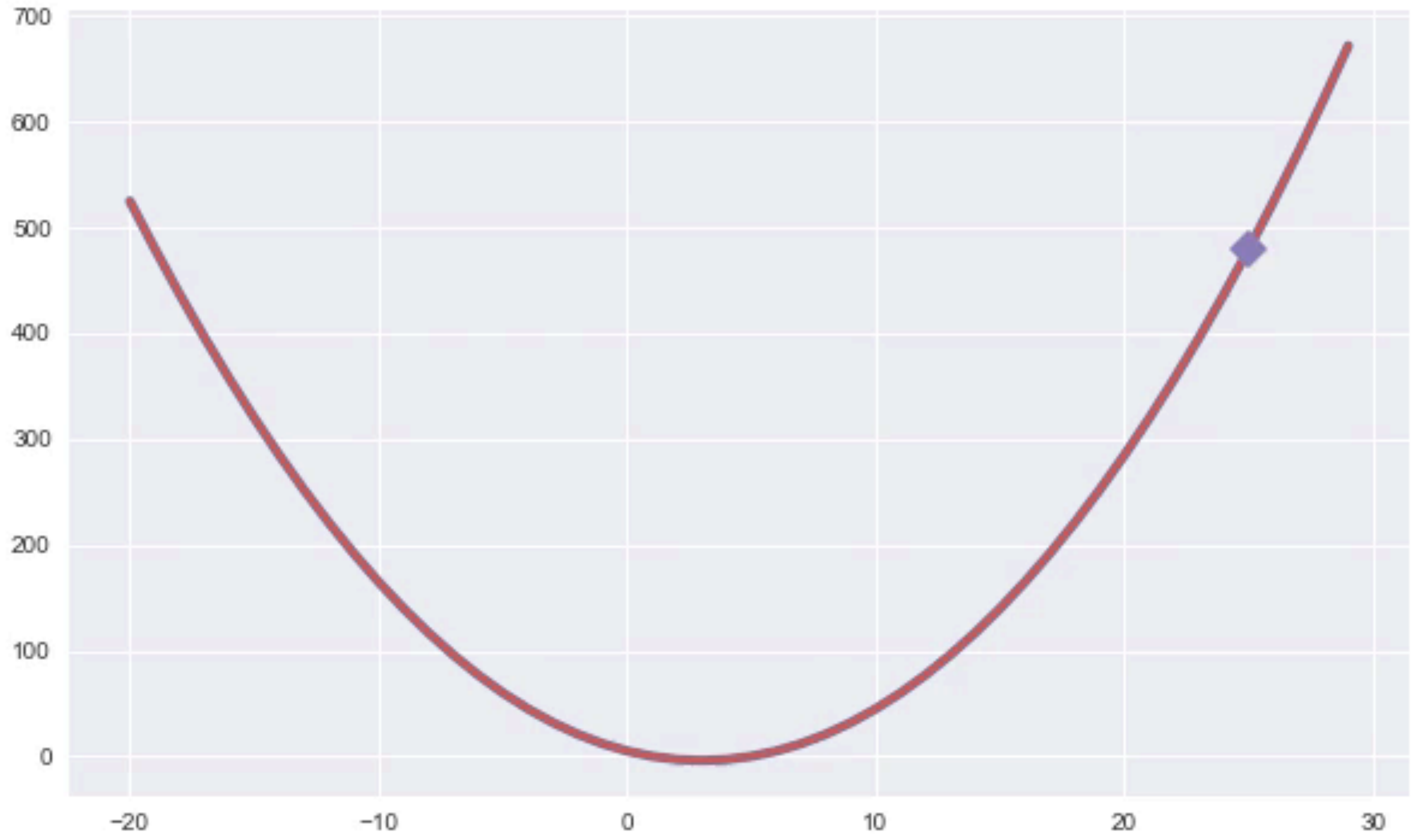
basically go opposite the direction of the derivative.

Consider the objective function:
$J(x) = x^2 - 6x + 5$

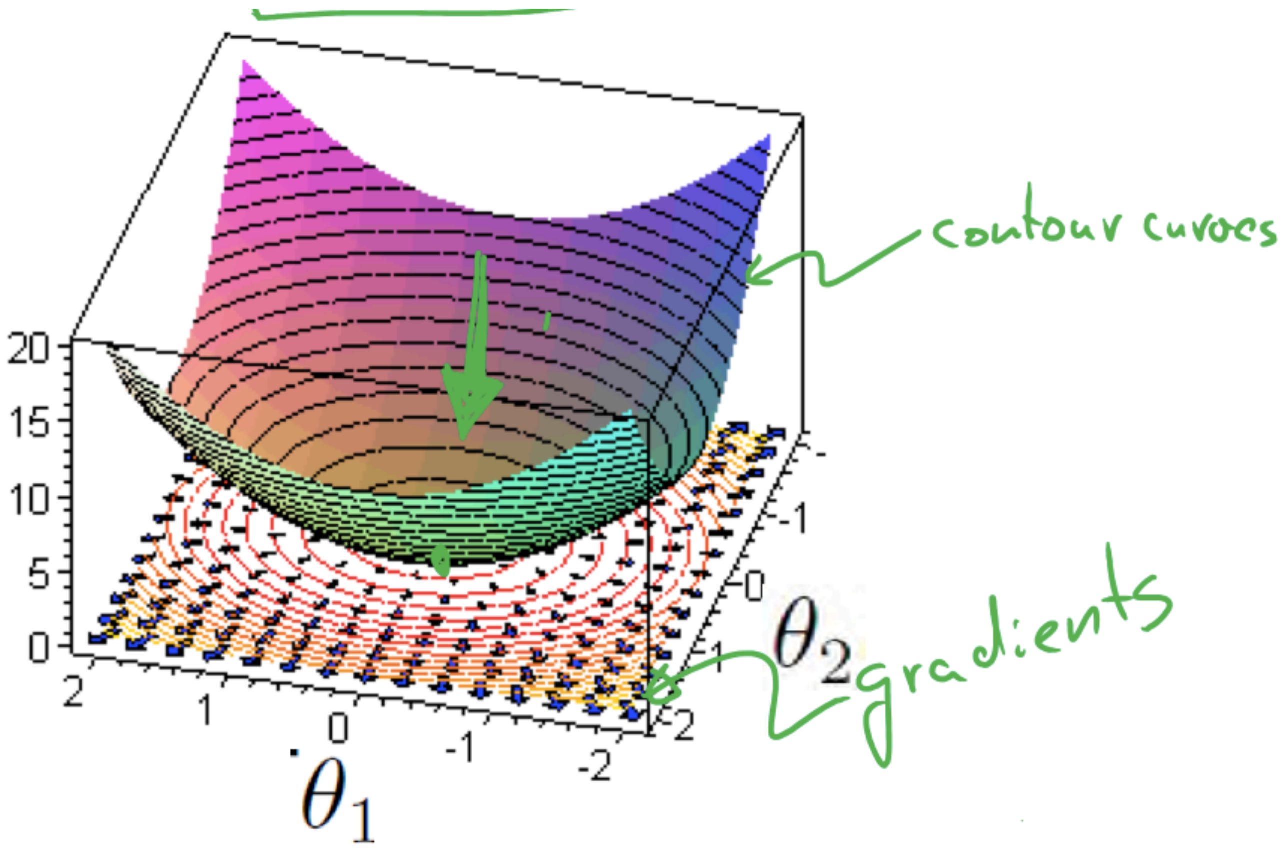
```
gradient = fprime(old_x)
move = gradient * step
current_x = old_x - move
```



good step size

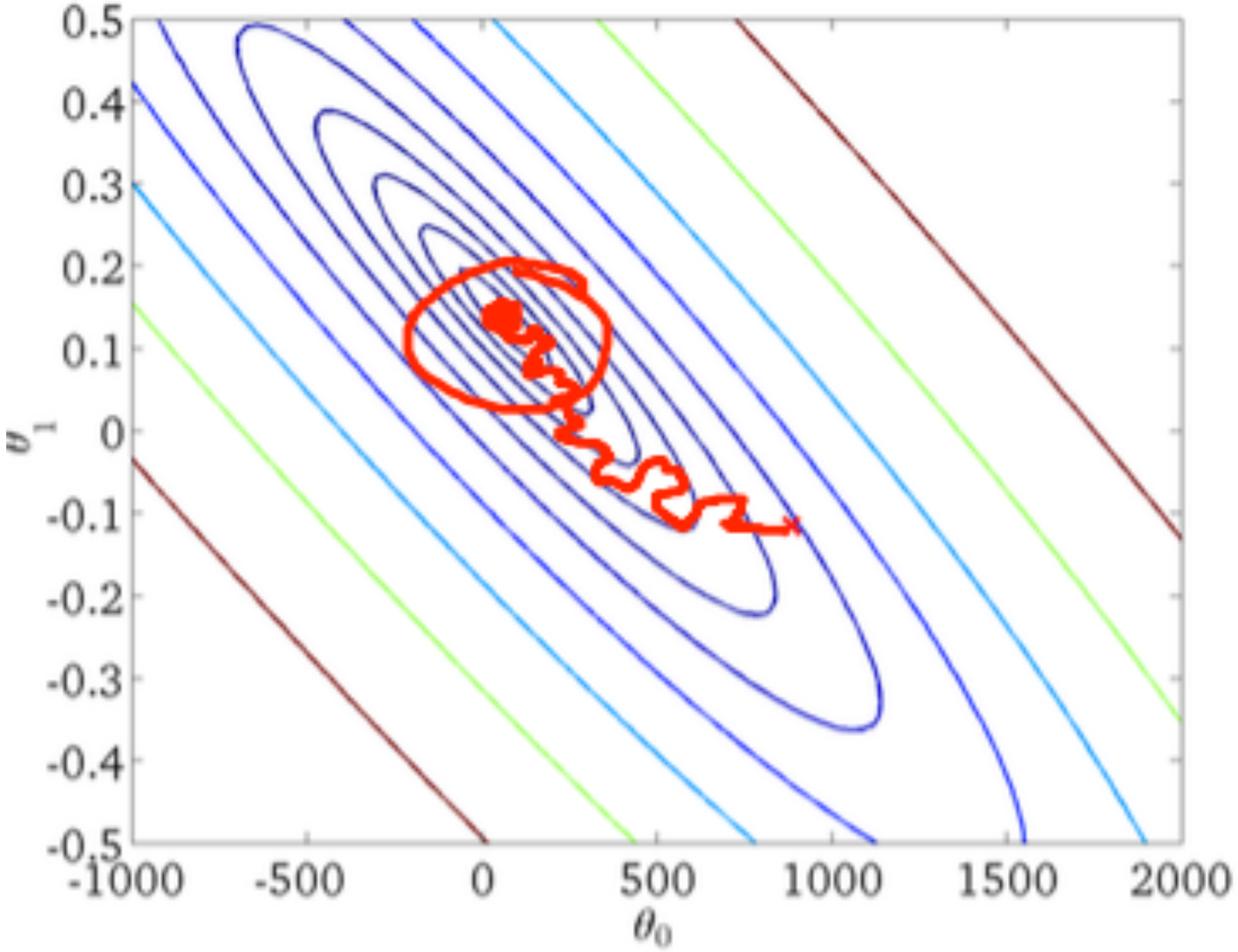


Gradient Descent and SGD



contour curves

gradients



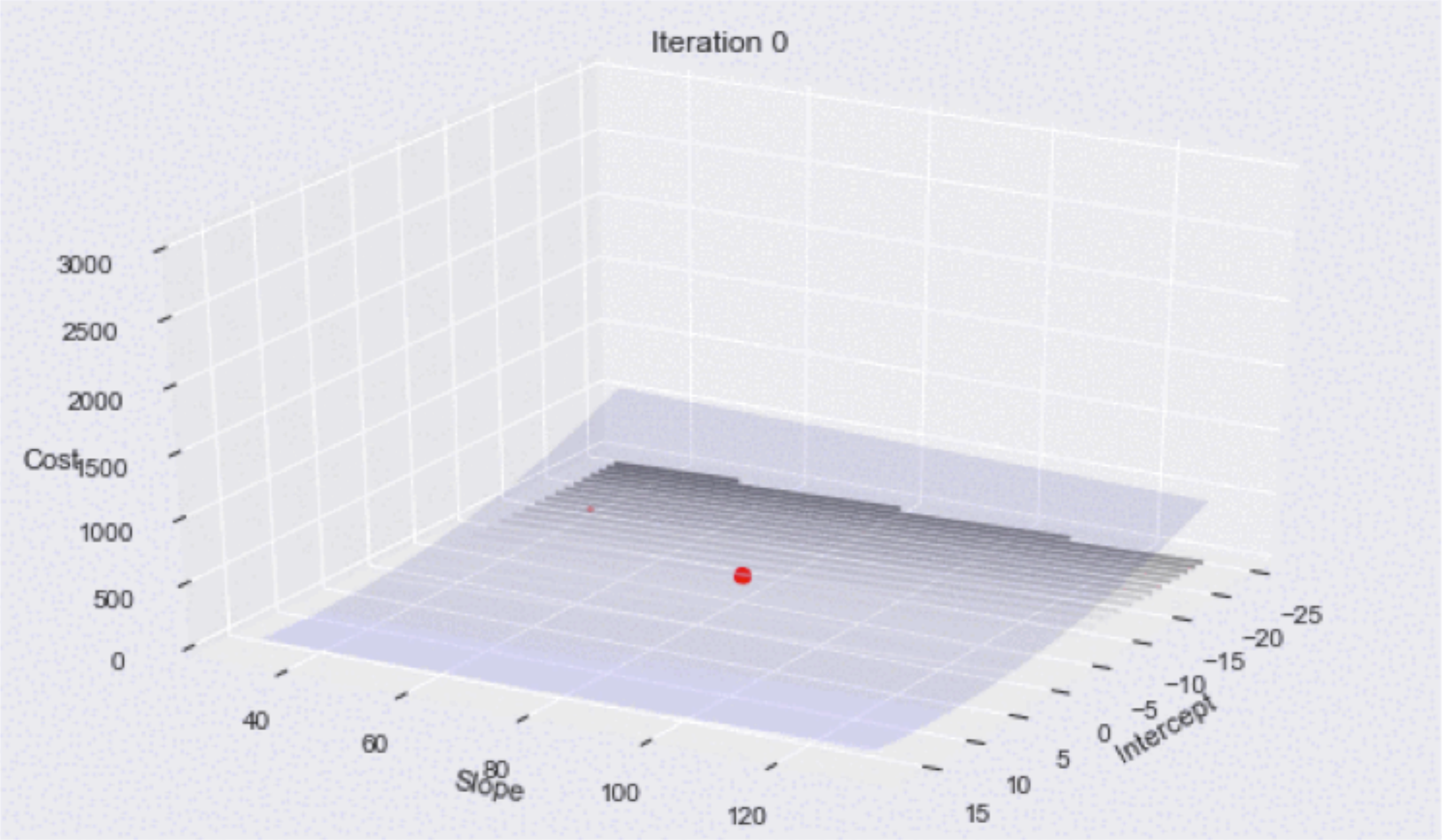
Stochastic Gradient Descent

$$\theta := \theta - \alpha \nabla_\theta J_i(\theta)$$

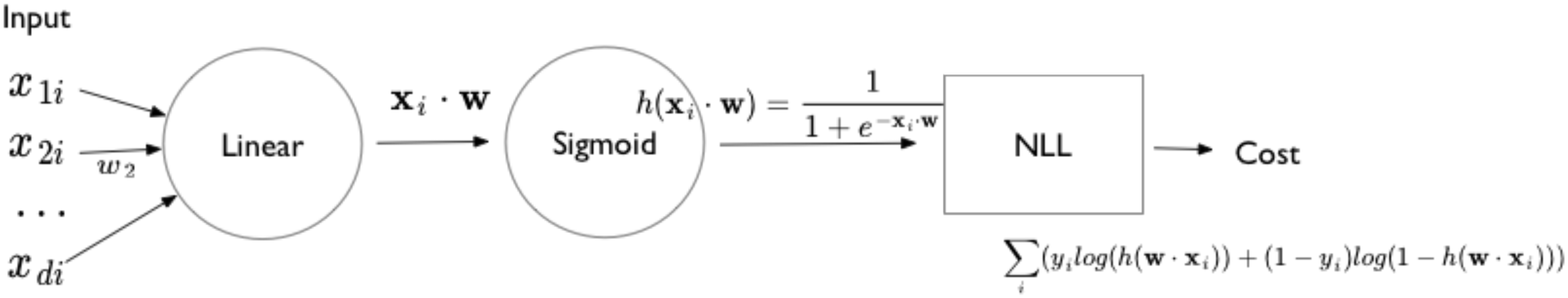
ONE POINT AT A TIME

```
for i in range(nb_epochs):
  np.random.shuffle(data)
  for example in data:
    params_grad = evaluate_gradient(loss_function, example, params)
    params = params - learning_rate * params_grad
```

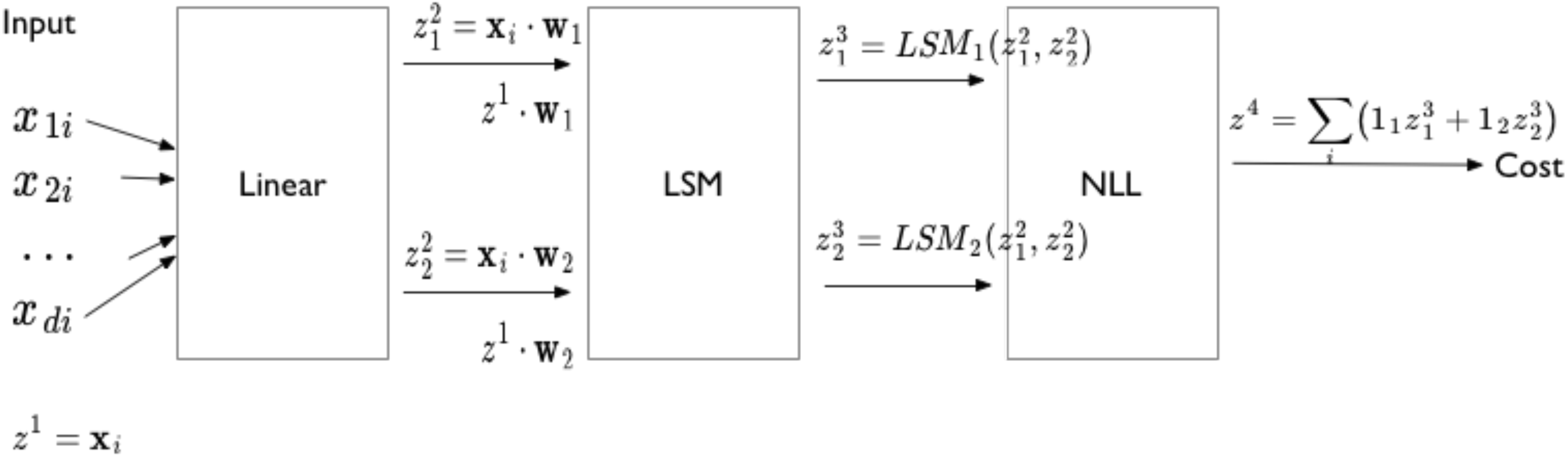
Mini-Batch: do some at a time


Iteration 0
Cost
3000
2500
2000
1500
1000
500
0
Slope
40
60
80
100
120
Intercept
-25
-20
-15
-10
-5
0
5
10
15

Logistic Regression Likelihood, graphically


Input
x_{1i}
x_{2i}
w_2
$\dots$
x_{di}
Linear
$\mathbf{x}_i \cdot \mathbf{w}$
Sigmoid
$h(\mathbf{x}_i \cdot \mathbf{w}) = \frac{1}{1+e^{-\mathbf{x}_i \cdot \mathbf{w}}}$
NLL
Cost
$\sum_i (y_i log(h(\mathbf{w} \cdot \mathbf{x}_i)) + (1-y_i) log(1-h(\mathbf{w} \cdot \mathbf{x}_i)))$

Softmax Formulation


Input
x_{1i}
x_{2i}
$\dots$
x_{di}
Linear
$z^2_1 = \mathbf{x}_i \cdot \mathbf{w}_1$
$z^1 \cdot \mathbf{w}_1$
$z^2_2 = \mathbf{x}_i \cdot \mathbf{w}_2$
$z^1 \cdot \mathbf{w}_2$
LSM
$z^3_1 = LSM_1(z^2_1, z^2_2)$
$z^3_2 = LSM_2(z^2_1, z^2_2)$
NLL
$z^4 = \sum_i (1_1 z^3_1 + 1_2 z^3_2)$
Cost
$z^1 = \mathbf{x}_i$

Backprop: Reverse Mode Differentiation

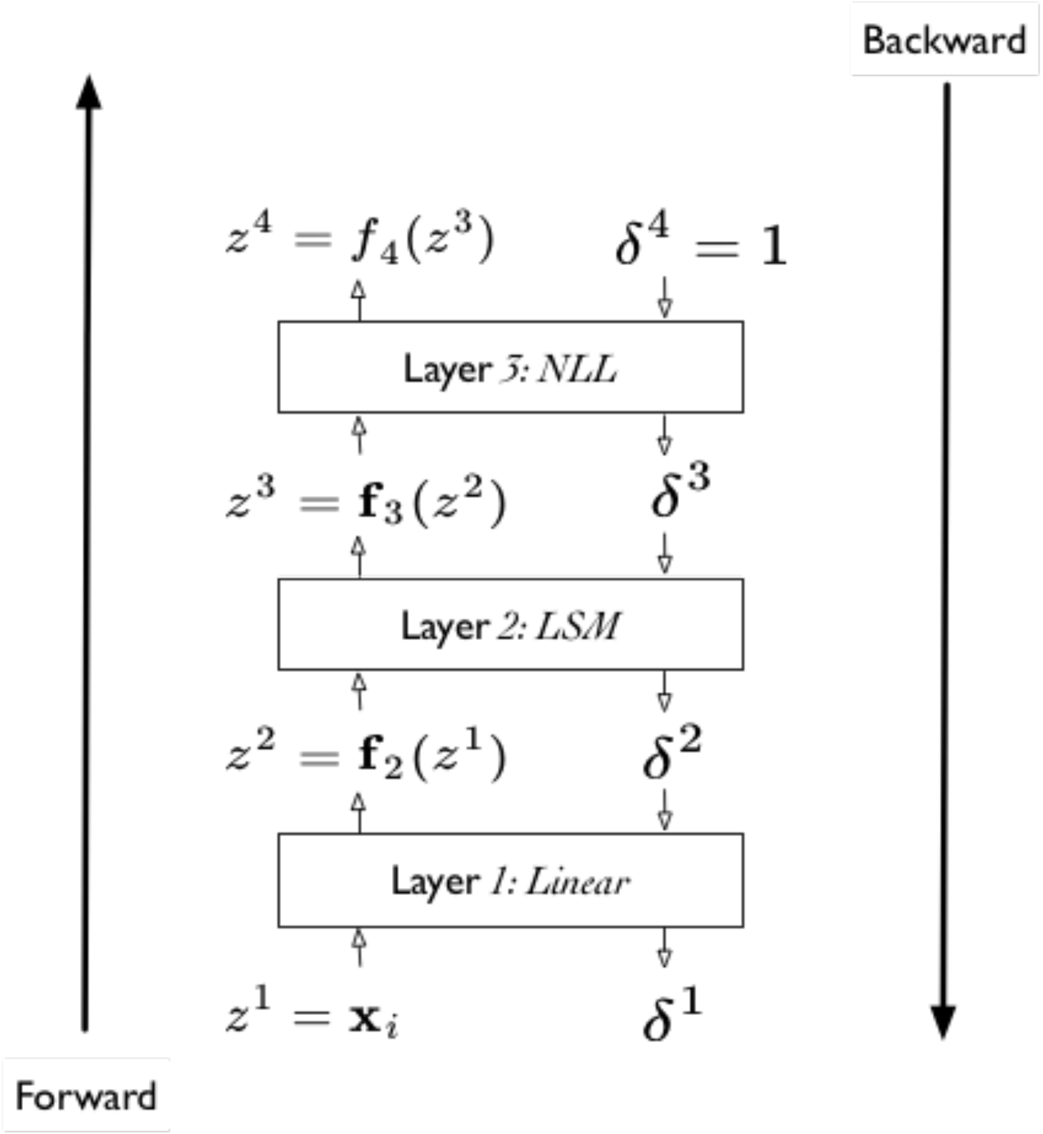
$$Cost = f^{Loss}(\mathbf{f^3}(\mathbf{f^2}(\mathbf{f^1}(\mathbf{x}))))$$

$$\nabla_{\mathbf{x}} Cost = \frac{\partial f^{Loss}}{\partial \mathbf{f^3}} \frac{\partial \mathbf{f}^3}{\partial \mathbf{f}^2} \frac{\partial \mathbf{f}^2}{\partial \mathbf{f}^1} \frac{\partial \mathbf{f}^1}{\partial \mathbf{x}}$$

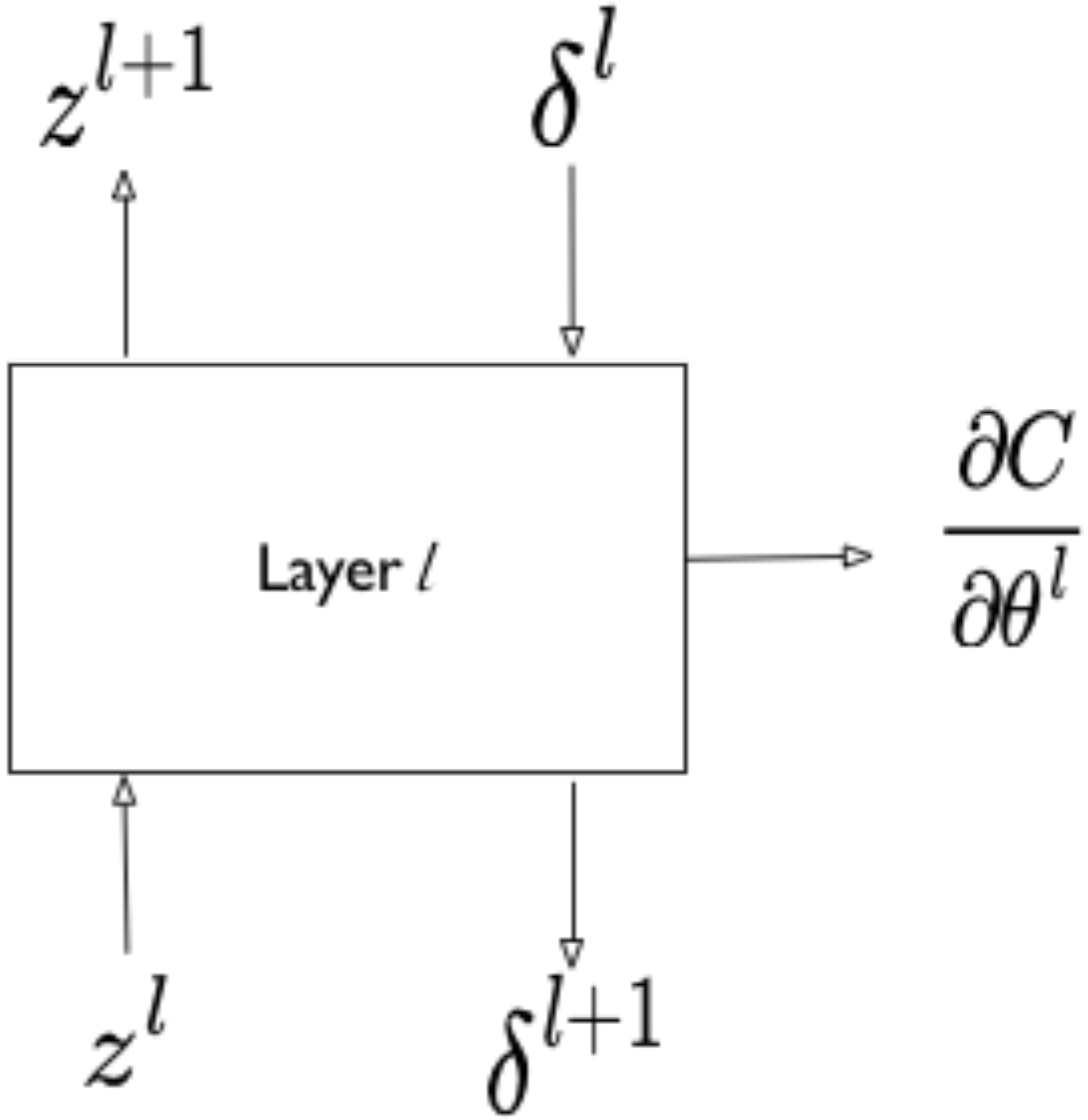
Write as:

$$\nabla_{\mathbf{x}} Cost = (((\frac{\partial f^{Loss}}{\partial \mathbf{f^3}} \frac{\partial \mathbf{f}^3}{\partial \mathbf{f}^2}) \frac{\partial \mathbf{f}^2}{\partial \mathbf{f}^1}) \frac{\partial \mathbf{f}^1}{\partial \mathbf{x}})$$

Layer Cake


Backward
$z^4 = f_4(z^3)$
$\delta^4 = 1$
Layer 3: *NLL*
$z^3 = \mathbf{f}_3(z^2)$
δ^3
Layer 2: *LSM*
$z^2 = \mathbf{f}_2(z^1)$
δ^2
Layer 1: *Linear*
$z^1 = \mathbf{x}_i$
δ^1
Forward

THATS IT! Write your Own Layer



Simulated Annealing

Minimize f by identifying with the energy of an imaginary physical system undergoing an annealing process.

Move from x_i to x_j via a **proposal**.

If the new state has lower energy, accept x_j.

If the new state has higher energy, accept with $A = \exp\left(-\Delta f/kT\right)$

Lowering temperature slowly, the system reaches "thermal equilibrium" at each temperature. Boltzmann's distribution:

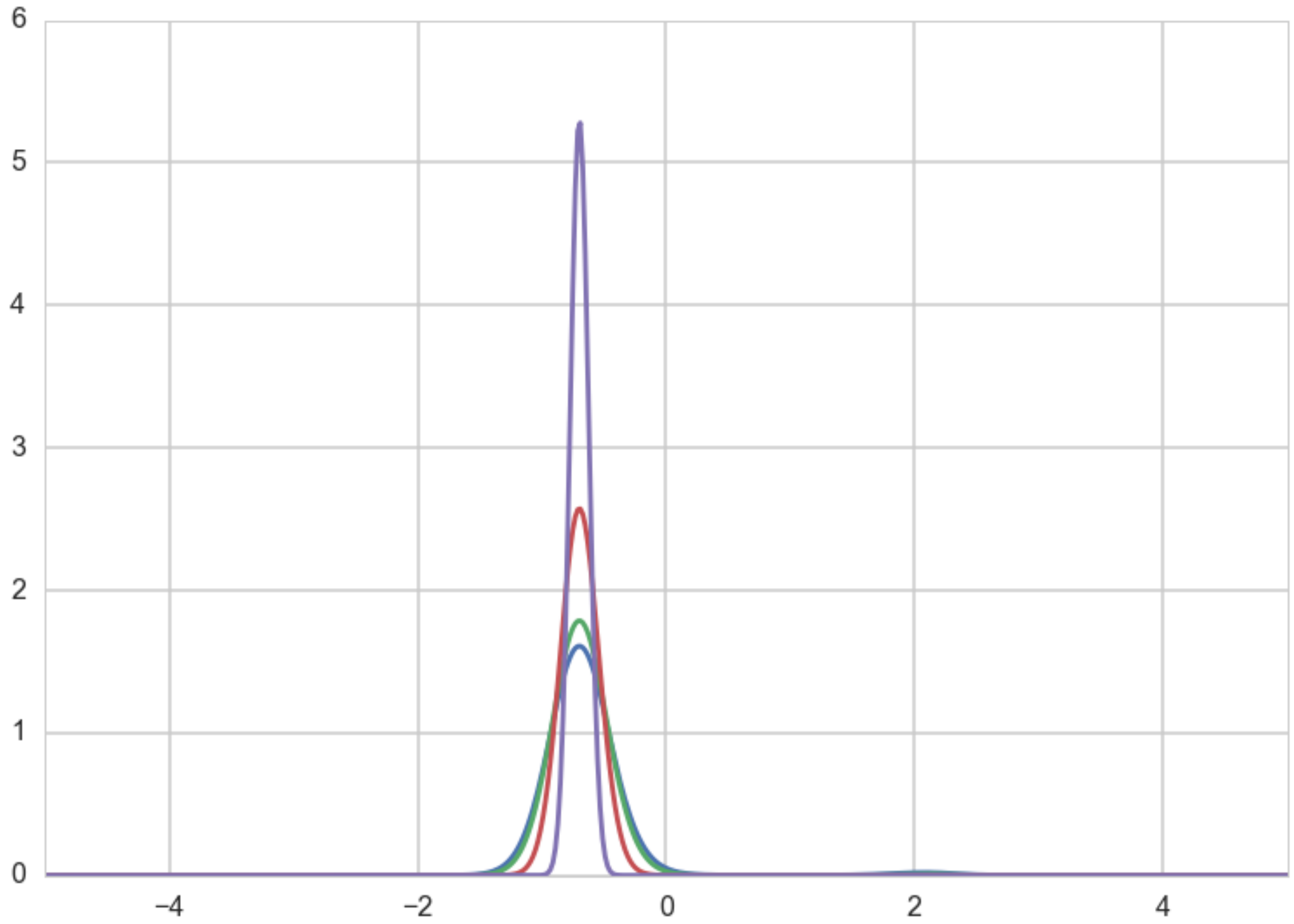
$$p(X=i) = \frac{1}{Z(T)} \exp\left(\frac{-E_i}{kT}\right)$$

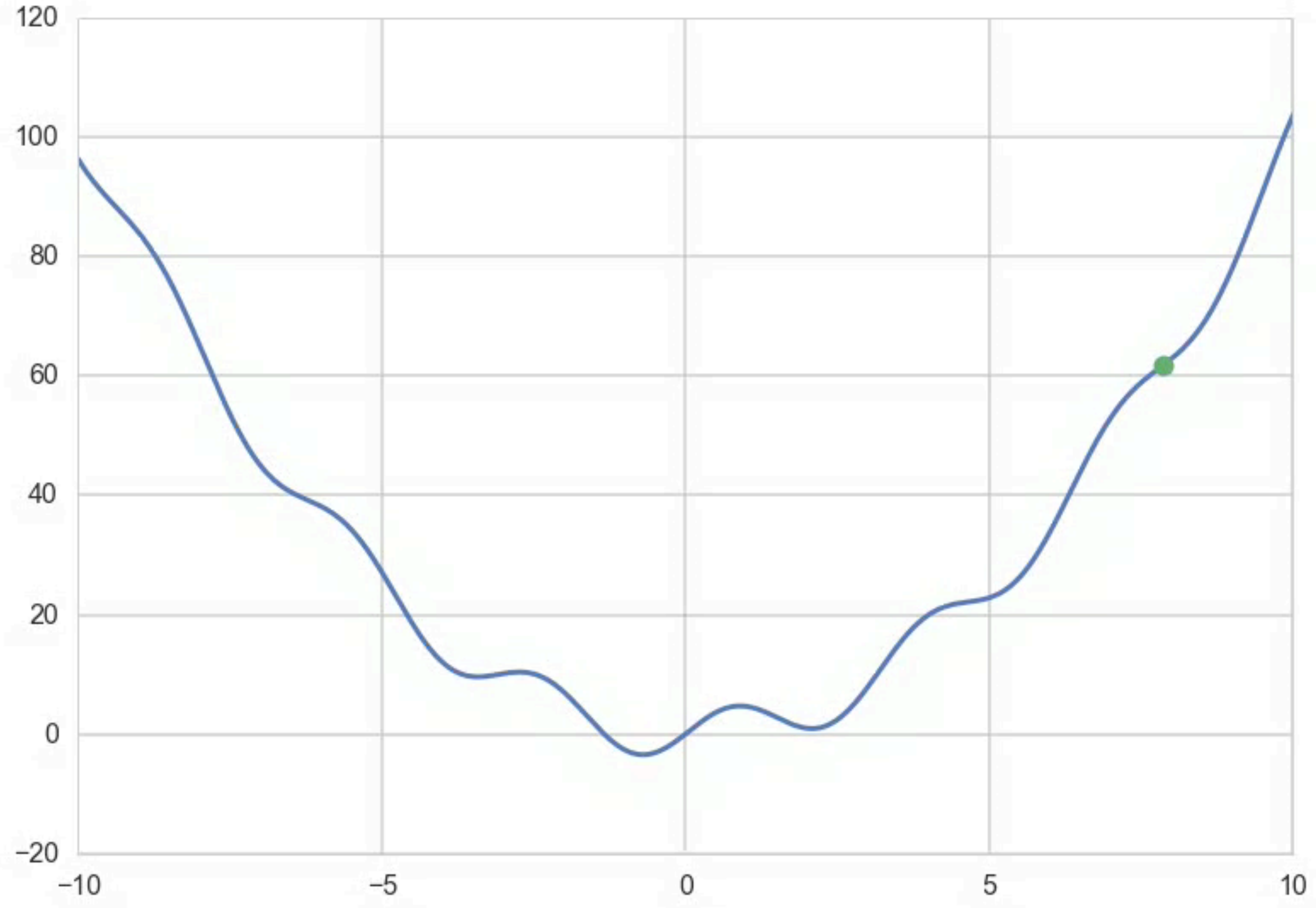
If you identify

$$p_T(x) = e^{-f(x)/T} \text{ and } p(x) = e^{-f(x)}$$

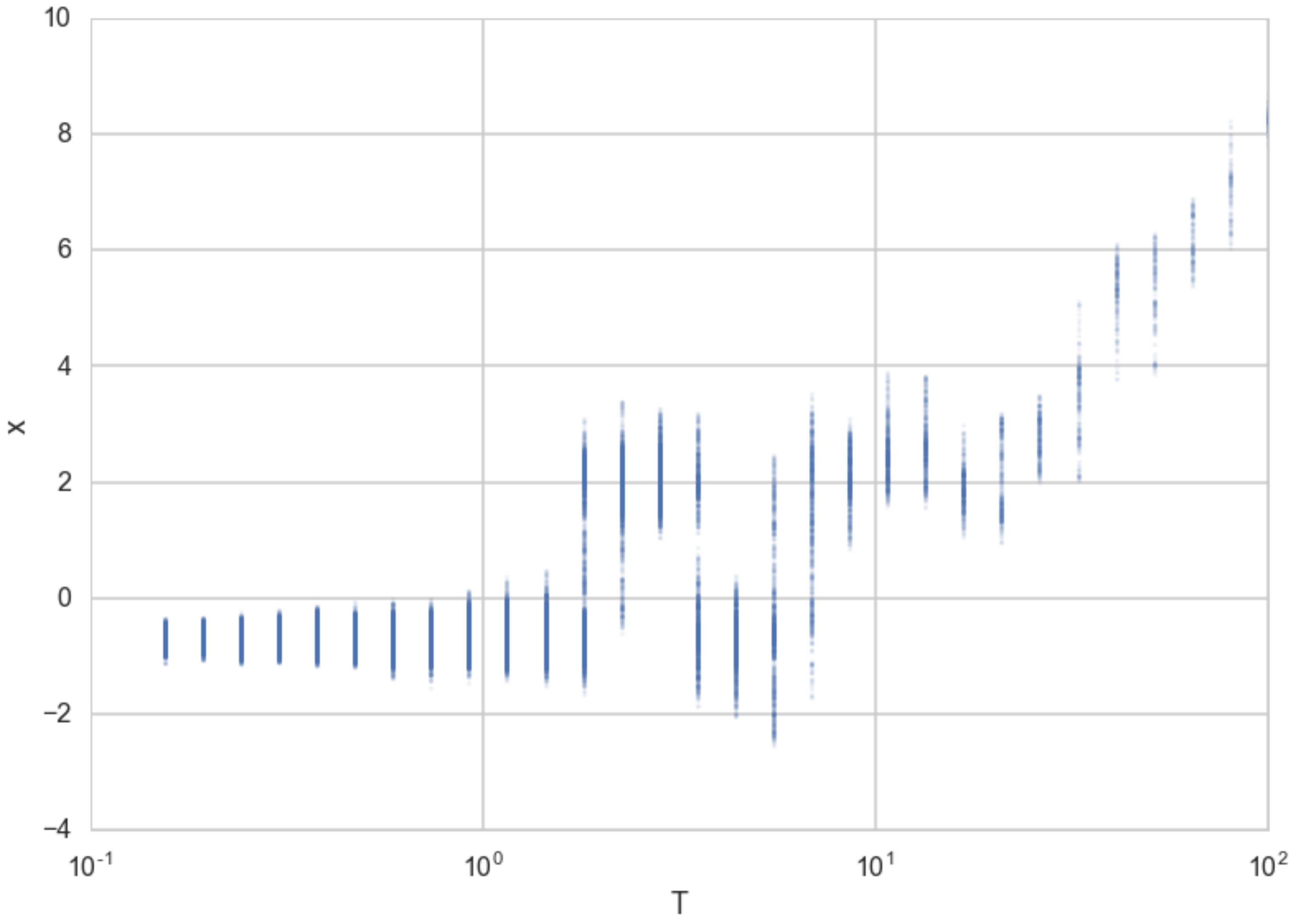
Then:

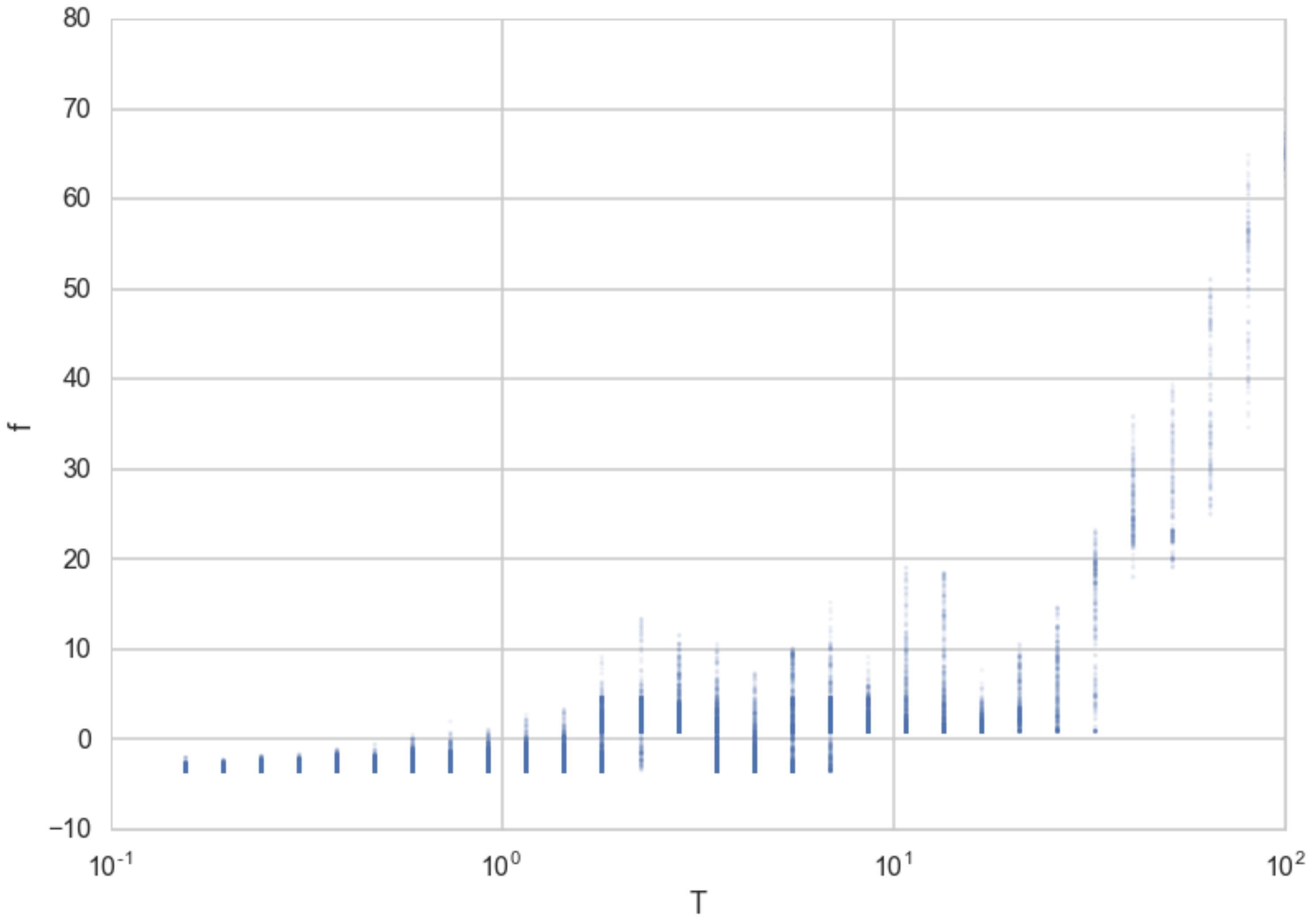
$$P_T(x) = P(x)^{1/T}$$




120
100
80
60
40
20
0
−20
−10
−5
0
5
10

SA Distributions

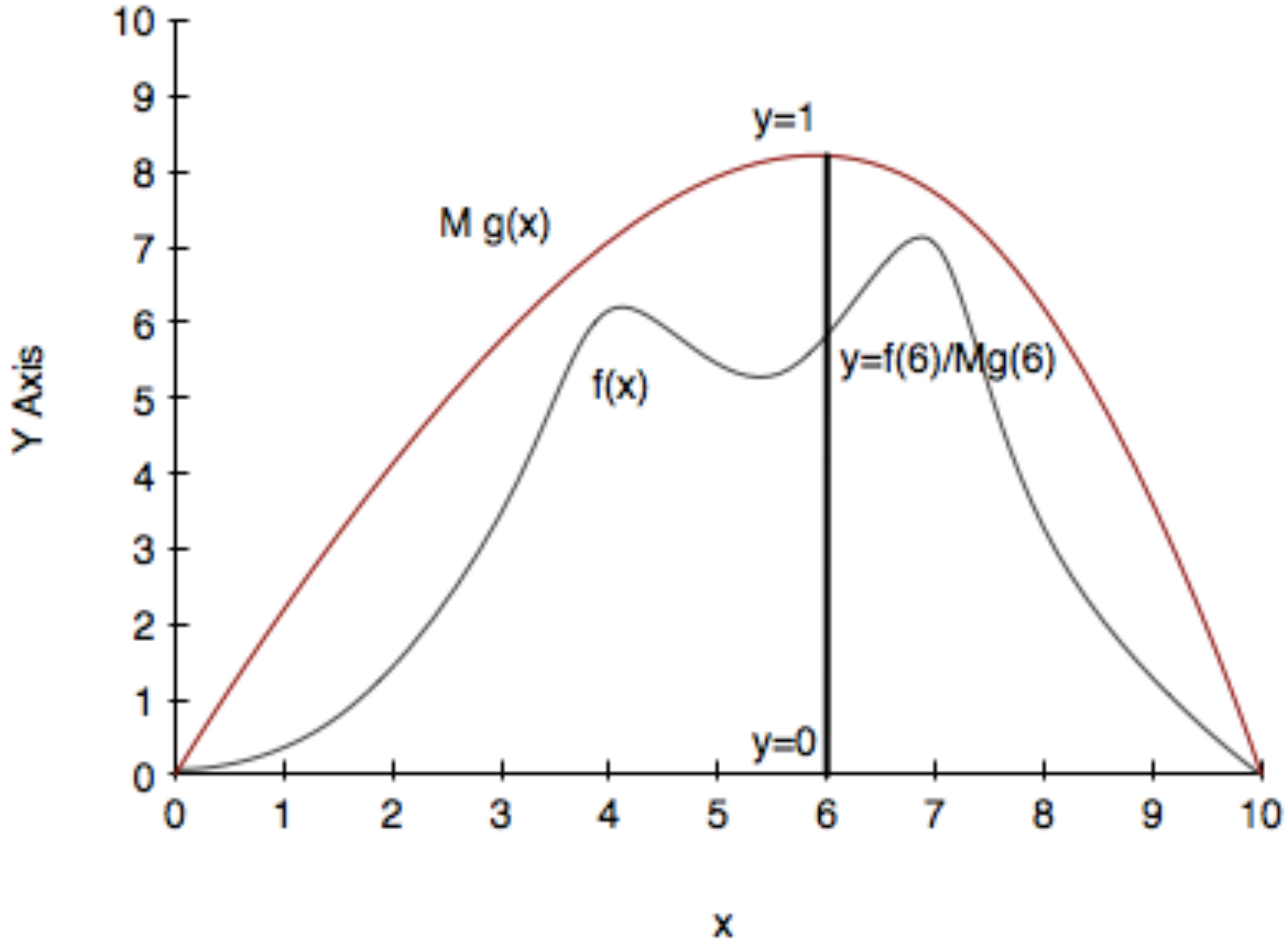




SAMPLING AND MCMC

GENERATE THEM!

- Inverse method, Rejection (on steroids)
- Stratification to reduce variance
- Importance (for expectations)
- MCMC, MH, HMC, Slice, ADVI, etc
- integrals (marginalize) by ignoring dimensions in histogram

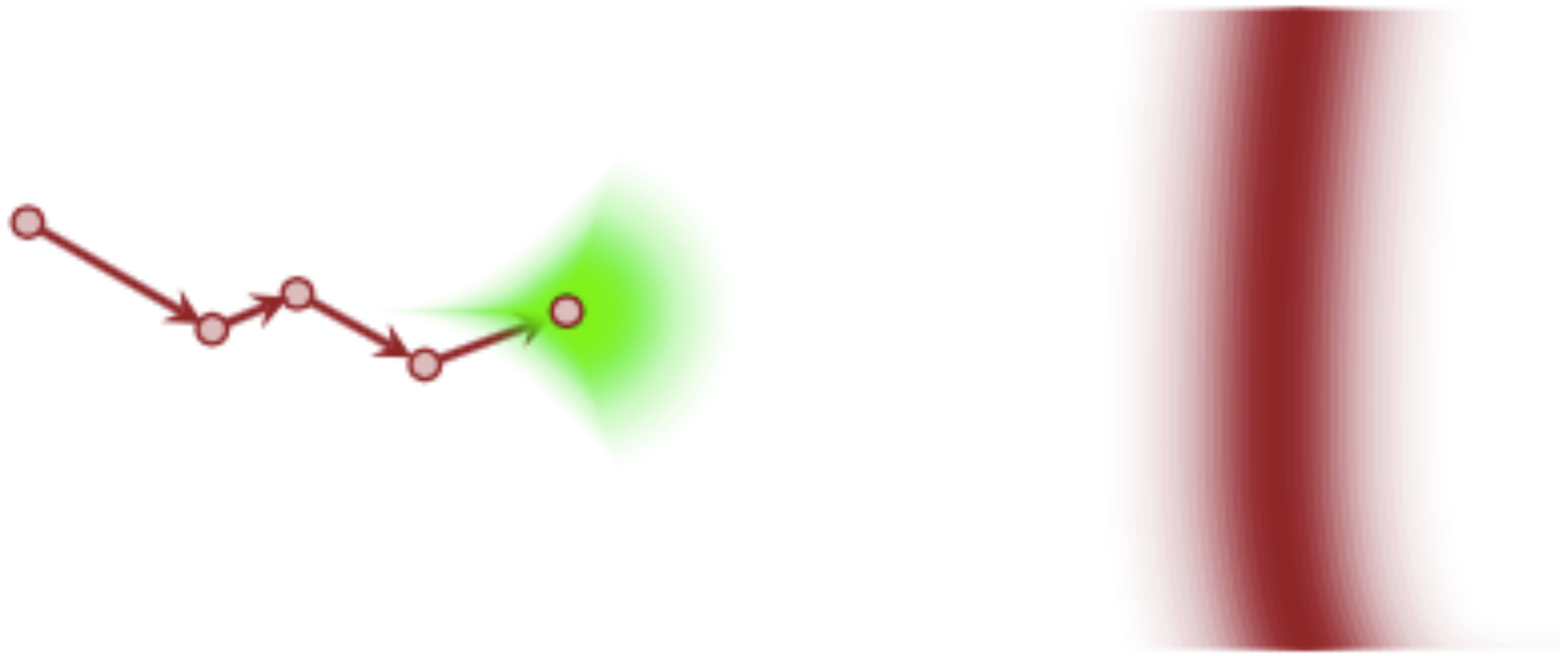


Sampling a Distribution

- Turn the annealing question on its head.
- Suppose we wanted to sample from a distribution $p(x)$ (corresponding to a minimization of energy $-log(p(x))$).
- keep our symmetric proposal (reversibility!). Need irreducibility to sample from full distribution
- set T=1, and use our simulated annealing method: **Metropolis**

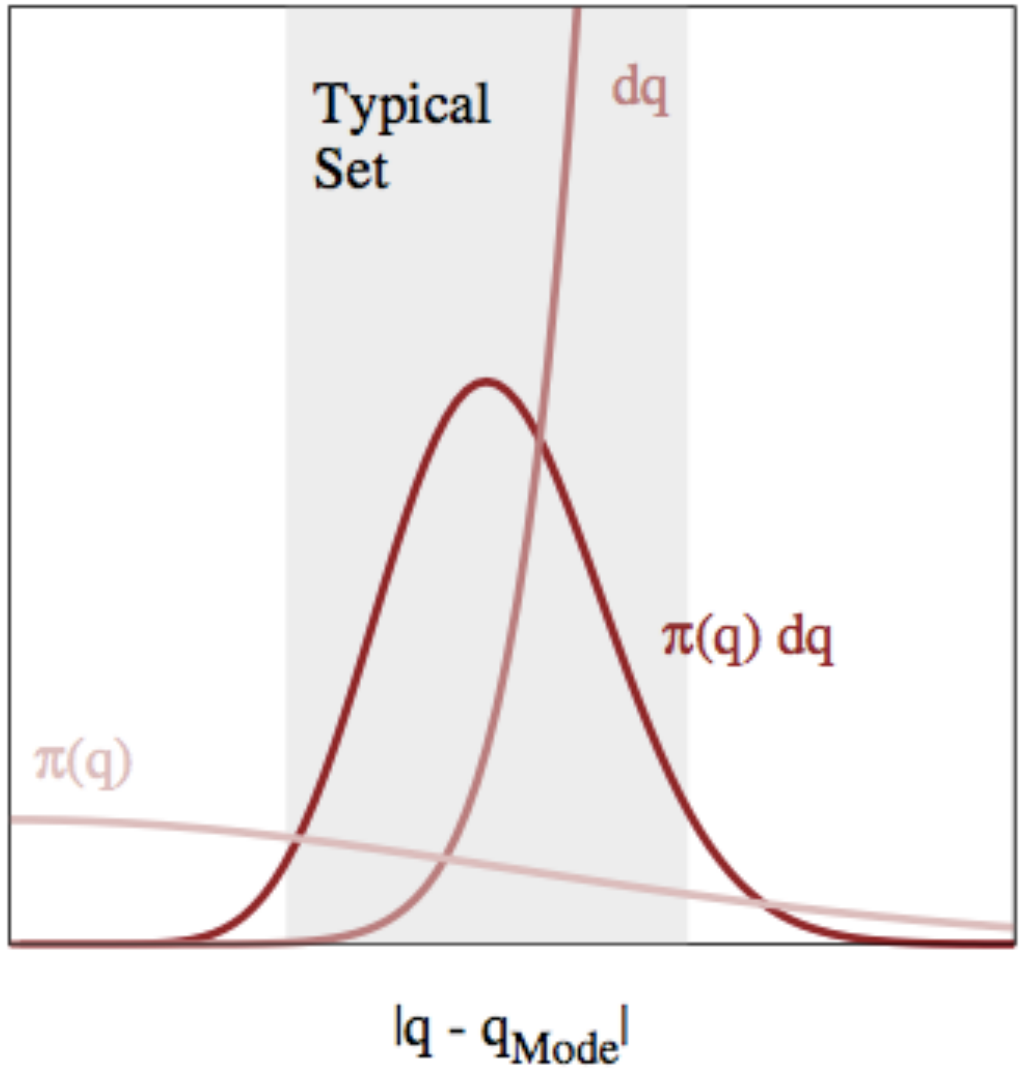
MCMC

Intuition: proposal approaches typical set



Instead of sampling p we sample q, yielding a new state, and a new proposal distribution from which to sample.

Critical: explore the typical set



Stationarity

$$sT = s \text{ or } \sum_i s_i T_{ij} = s_j \text{ or}$$

Continuous case: define T so that:

$$\int dx_i \, s(x_i) T(x_{i+1} | x_i) = s(x_{i+1}) \text{ then}$$

$$\int dx s(x) T(y|x) = \int p(y, x) dx = s(y)$$

AM 207

Detailed balance is enough for stationarity

$$s(x)T(y|x) = s(y)T(x|y)$$

If one sums both sides over x

$\int dx s(x)t(y|x) = s(y) \int dx T(x|y)$ which gives us back the stationarity condition from above.

Need Ergodicity

- “Ergodic” law of large numbers:

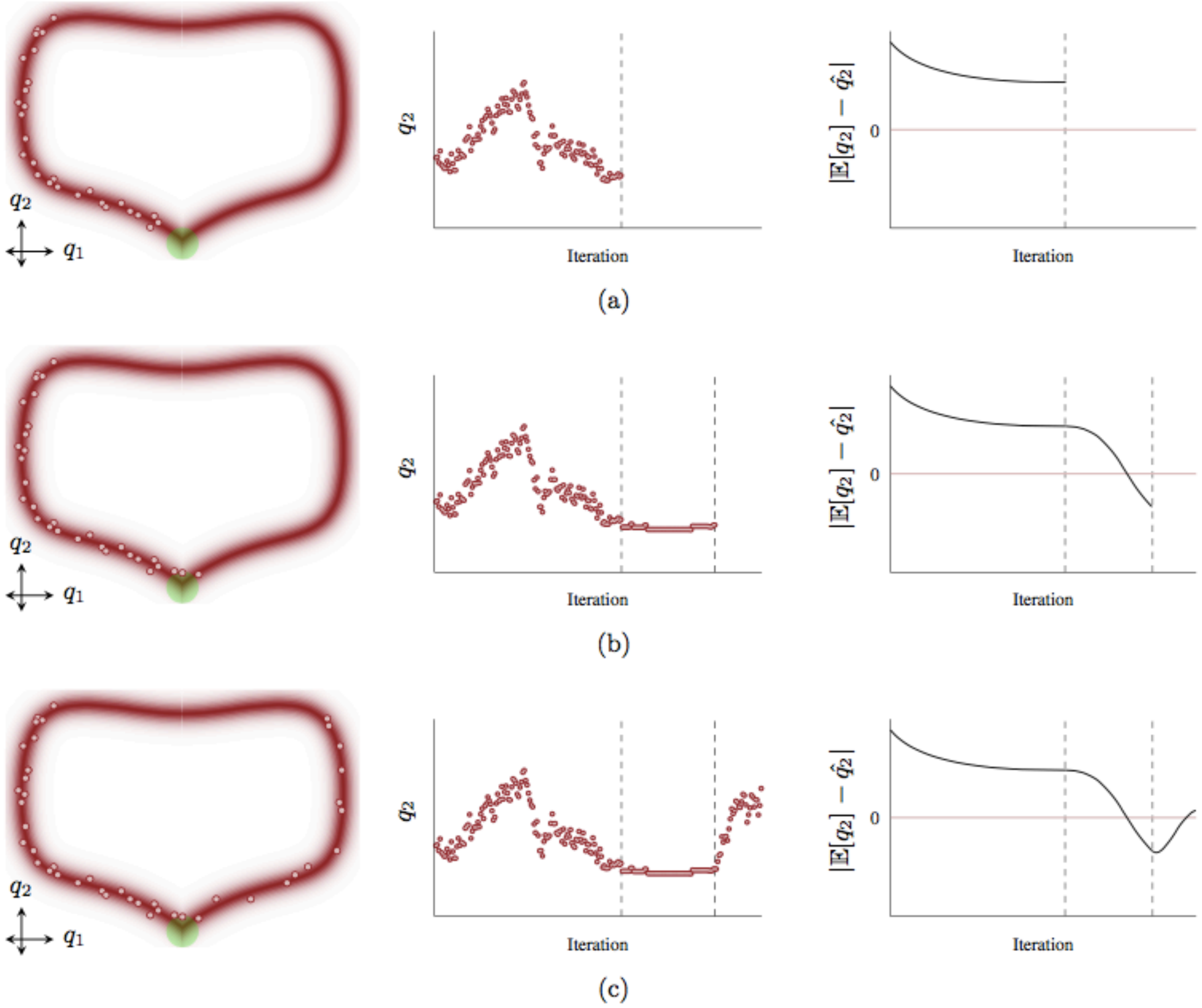
$$\int g(x)f(x)dx = \frac{1}{N}\sum_{j=B+1}^{B+N} g(x_j)$$

(the jury is out on thinning. Most dont think one needs it)

If there exists a stationary $s(x)$, you can construct a T such that $\lim_{t \to \infty} T^n$ is stationary and converges to s, and

- an ergodic law of large numbers exists
- an ergodic central limit theorem exists

Metropolis and MH

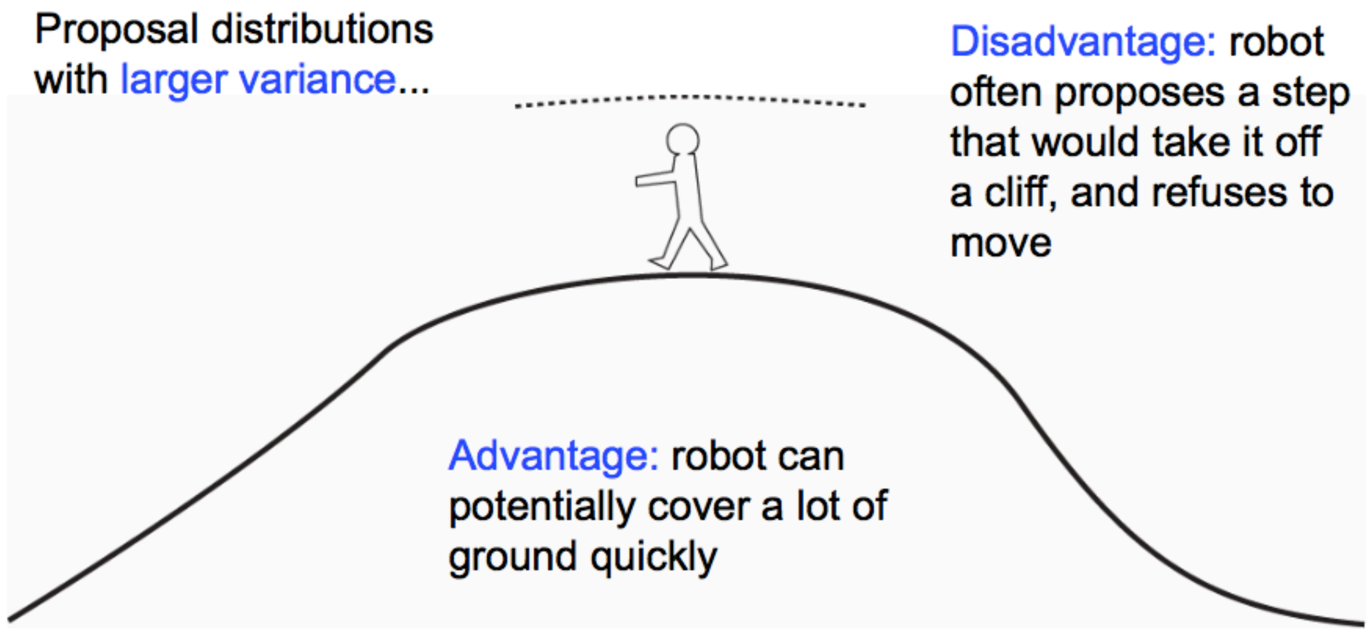


(a)

(b)

(c)

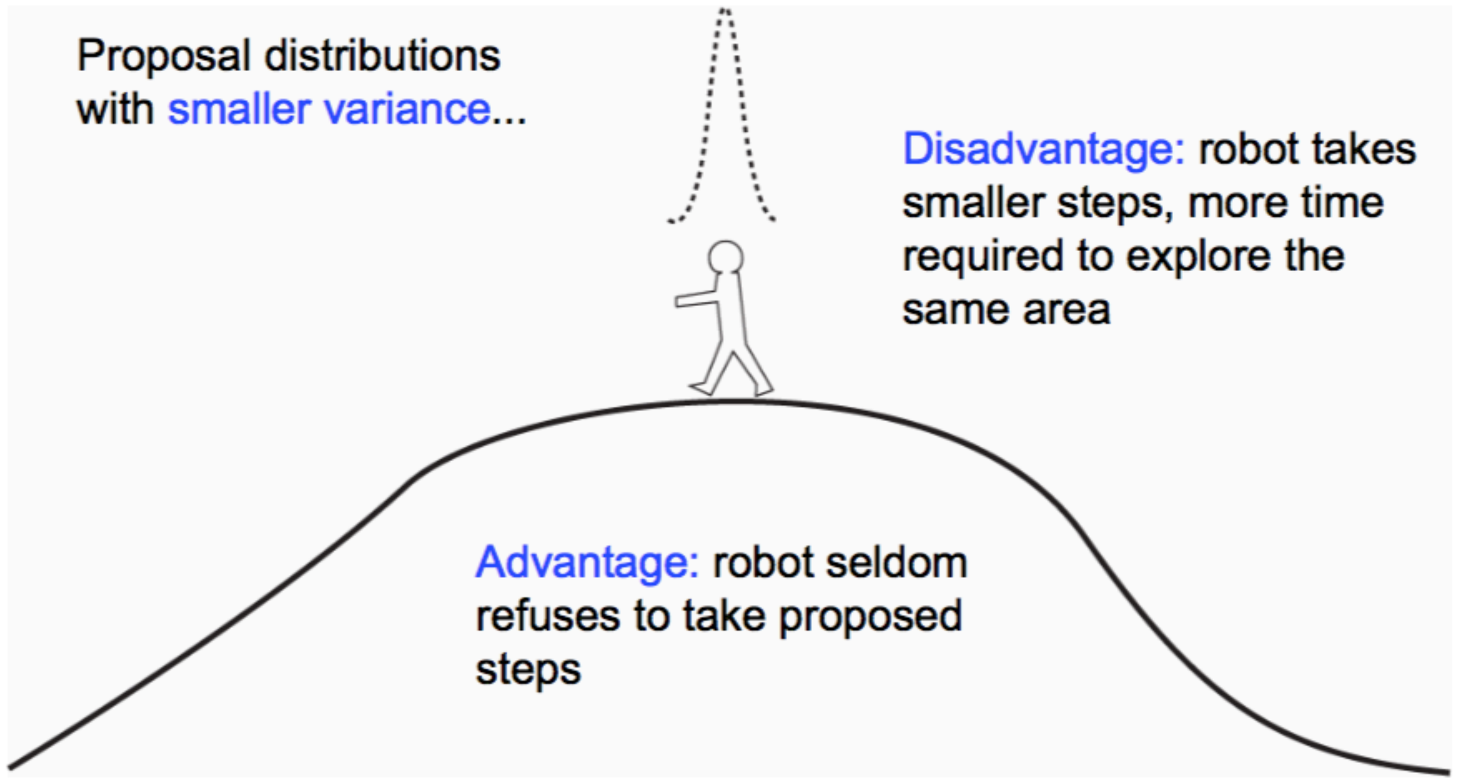
- overshoot and oscillate at pinches
- need to specify step step sizes
- large steps can go outside typical set and are not accepted
- but can cover ground quickly
- small steps accepted but go nowhere
- large correlations



Proposal distributions with larger variance...

Disadvantage: robot often proposes a step that would take it off a cliff, and refuses to move

Advantage: robot can potentially cover a lot of ground quickly



Proposal distributions with smaller variance...

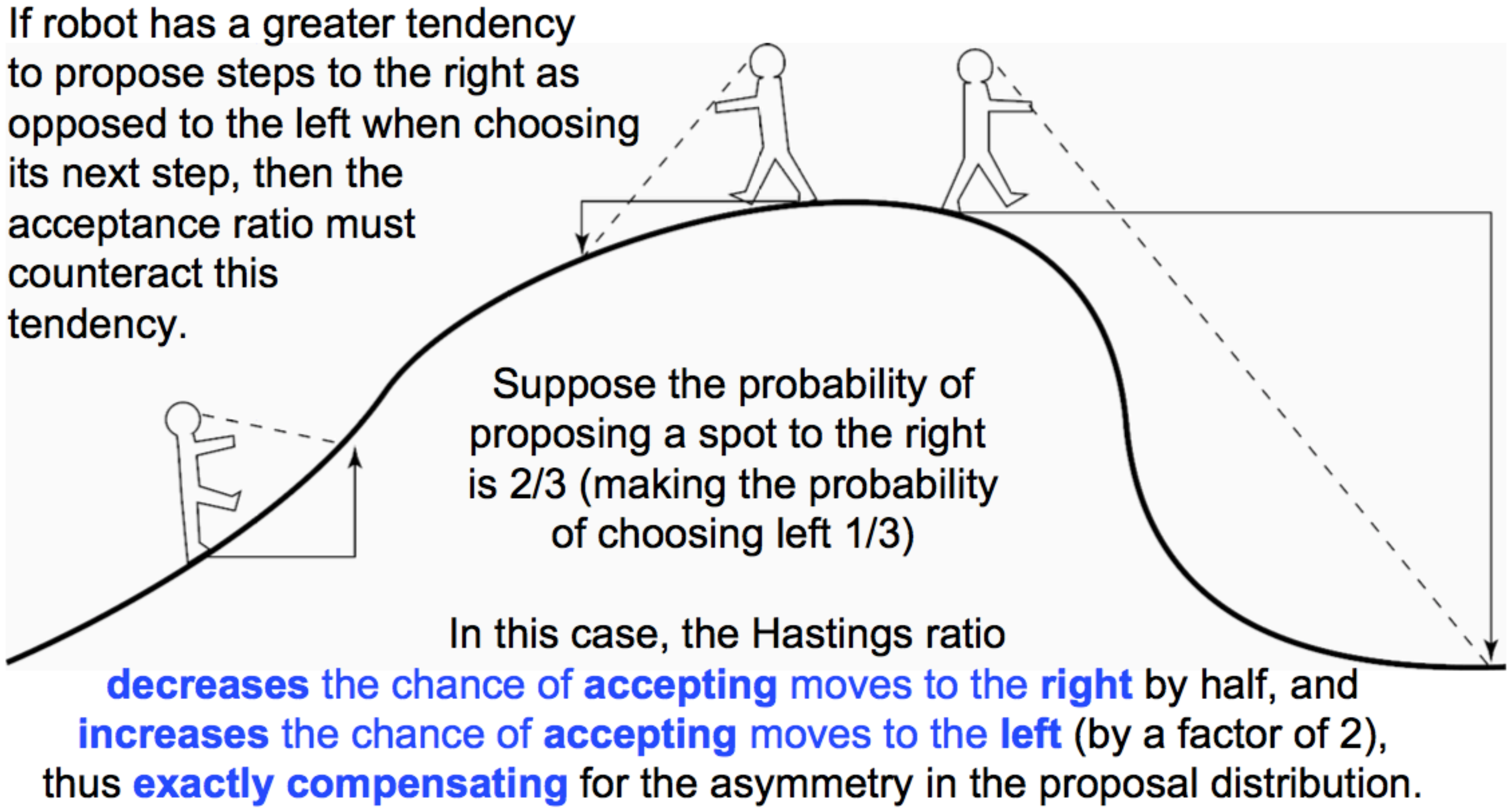
Disadvantage: robot takes smaller steps, more time required to explore the same area

Advantage: robot seldom refuses to take proposed steps

MH Acceptance

$$A(x_i, x_{i-1}) = min(1, \frac{s(x_i) \times q(x_{i-1}|x_i)}{s(x_{i-1}) \times q(x_i|x_{i-1})}).$$

- correct the sampling of q to match p, corrects for any asymmetries in the proposal distribution.
- A good rule of thumb is that the proposal has the same or larger support then the target, with the same support being the best.



If robot has a greater tendency to propose steps to the right as opposed to the left when choosing its next step, then the acceptance ratio must counteract this tendency.

Suppose the probability of proposing a spot to the right is 2/3 (making the probability of choosing left 1/3)

In this case, the Hastings ratio **decreases** the chance of **accepting** moves to the **right** by half, and **increases** the chance of **accepting** moves to the **left** (by a factor of 2), thus **exactly compensating** for the asymmetry in the proposal distribution.

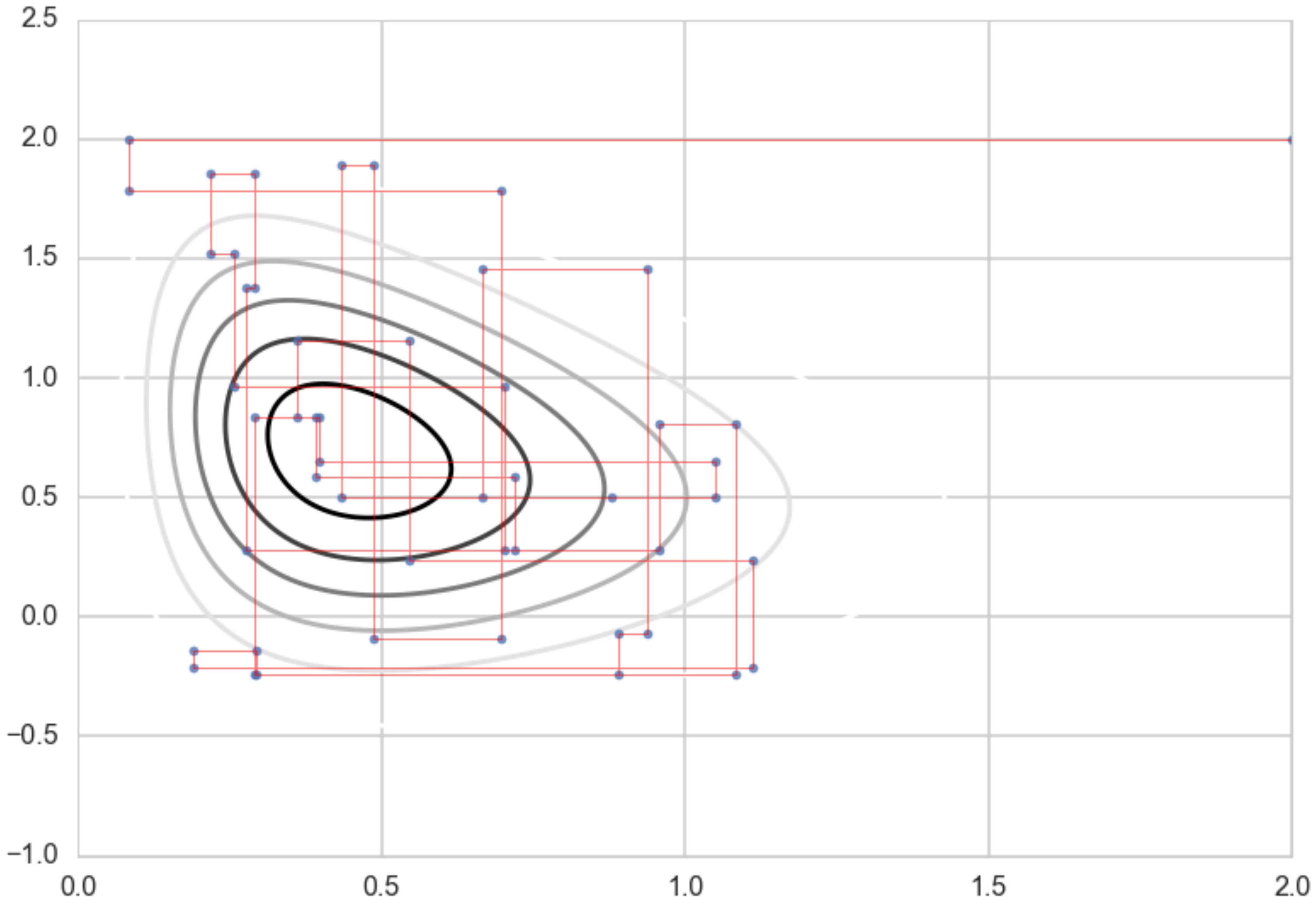
(from Paul Lewis)

The idea of Gibbs

$f(x_t) = \int h(x_t, x_{t-1}) f(x_{t-1}) dx_{t-1}$, a Stationary distribution.

$h(x, x') = \int dy f(x|y) f(y|x')$.: Sample alternately to get transitions.

Can sample x marginal and $x|y$ so can sample the joint x, y.



Data Augmentation

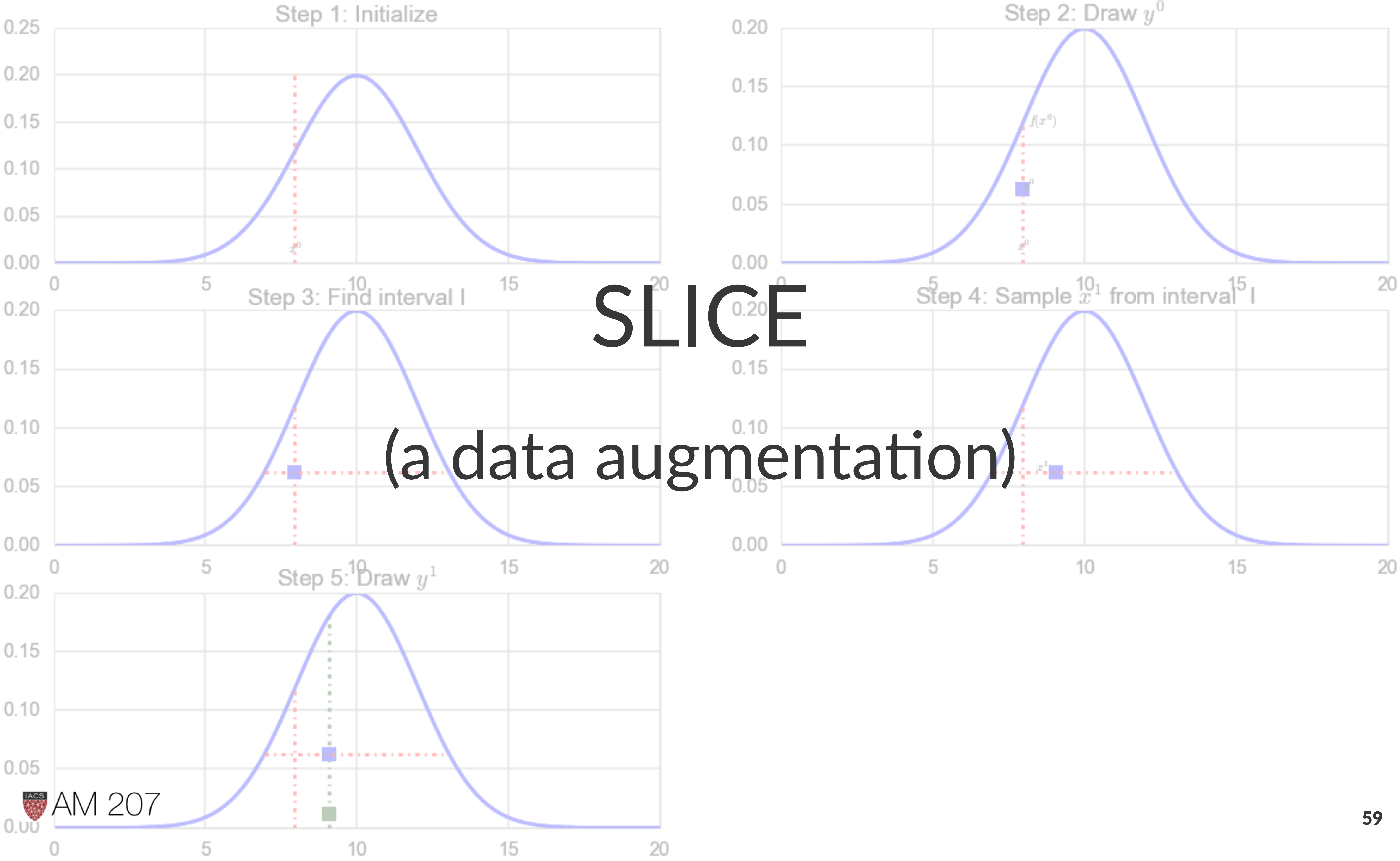
The difference from Gibbs Sampling: the other variable, say y, is to be treated as latent.

The game is to construct a joint $p(x, y)$ such that we can sample from $p(x|y)$ and $p(y|x)$, and then find the marginal

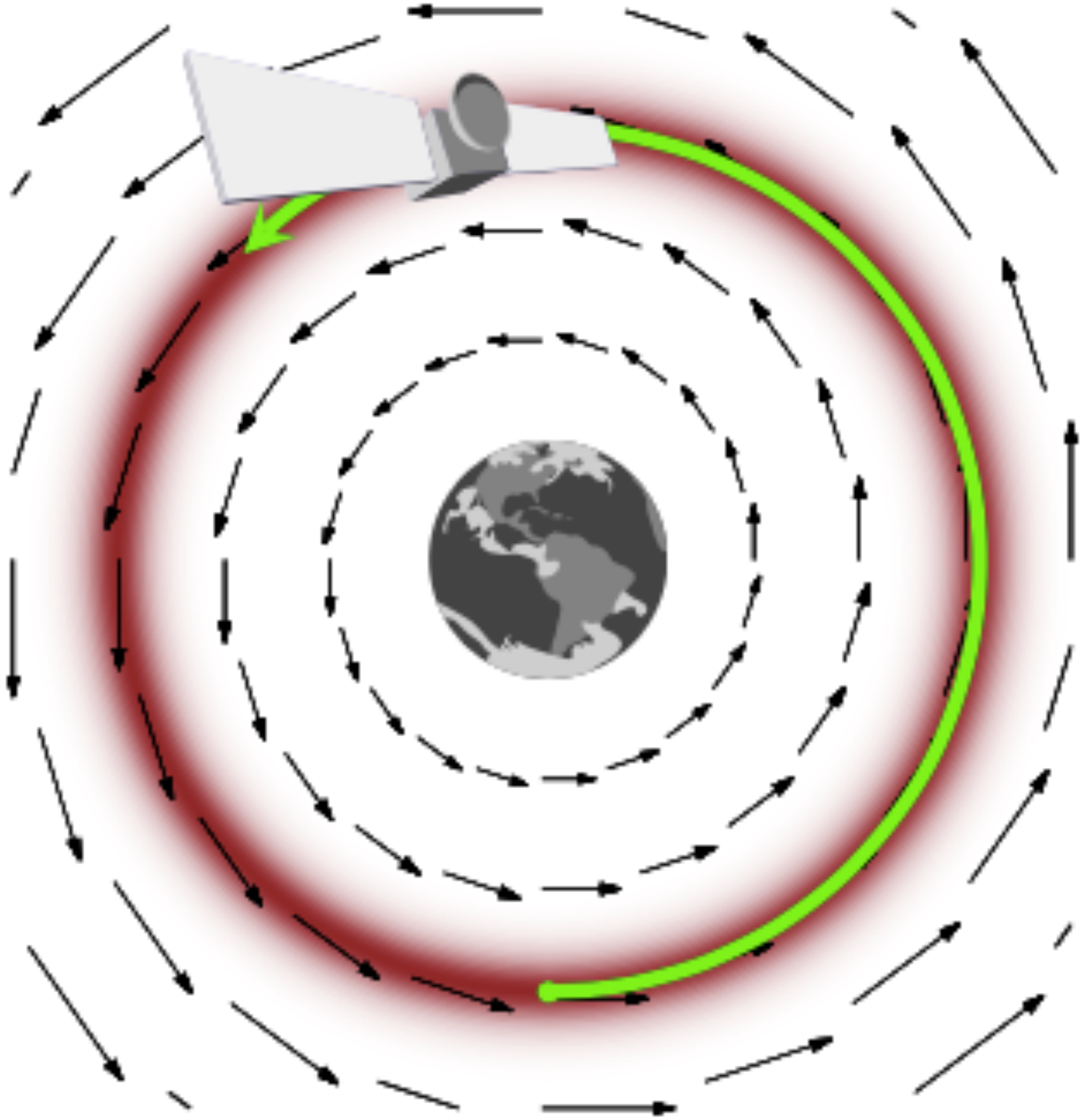
$$p(x) = \int dy\, p(x, y).$$

SLICE

(a data augmentation)



HMC & NUTS



HMC to the rescue: need glide

Now, like in annealing, let $p(p,q) = e^{-Energy}$

DATA AUGMENTATION: with an additional momentum gives energy

Hamiltonian $H(p,q) = \frac{p^2}{2m} + V(q)$

Hamiltonian flow: reversible, time-invariant, volume-preserving

Thrusters fire away

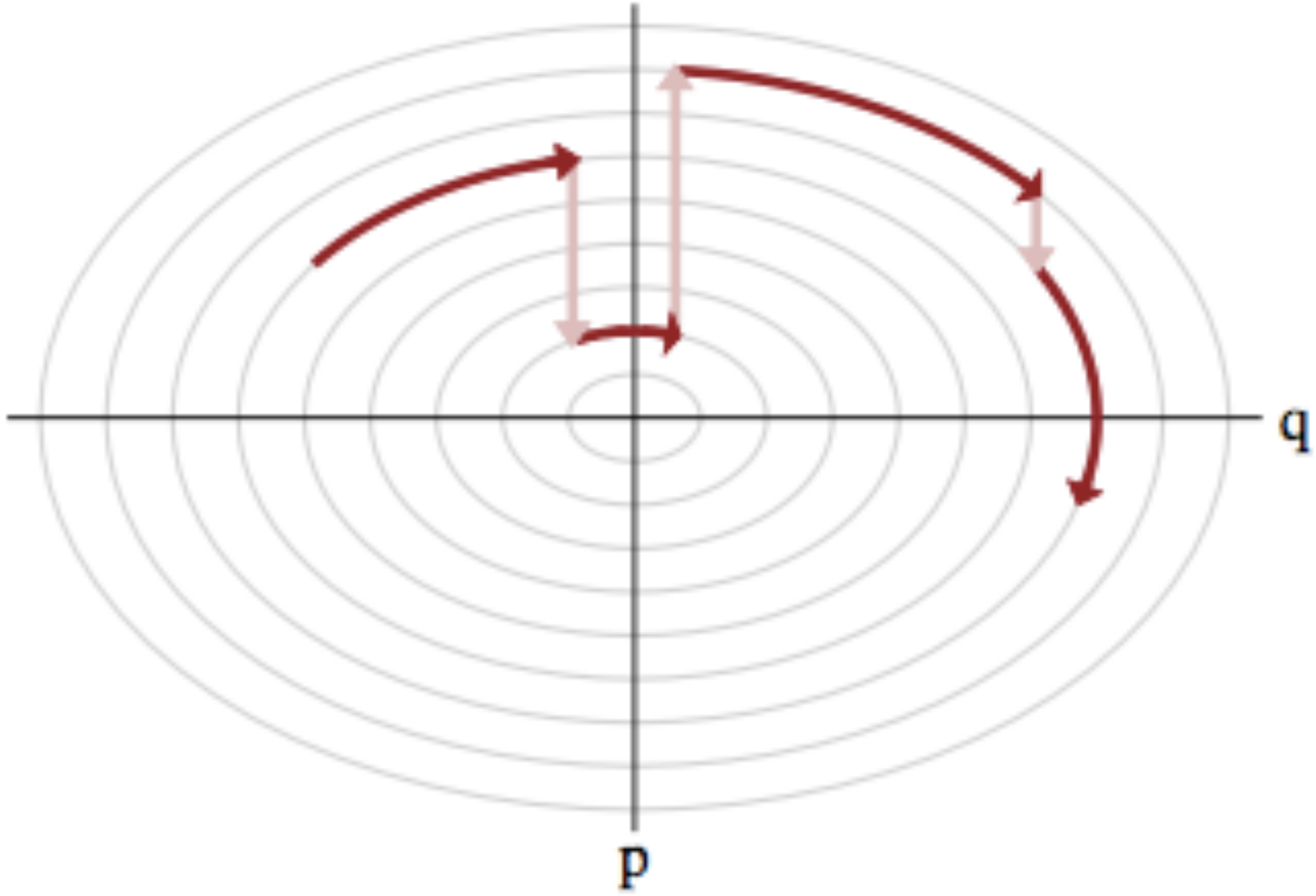
$$p(p,q) = e^{-H(p,q)} = e^{-K(p,q)}e^{-V(q)} = p(p|q)p(q)$$

$$H(p,q) = -log(p(p,q)) = -logp(p|q) - logp(q)$$

Choice of a kinetic energy term is choice of a conditional probability distribution over the "augmented" momentum such that:

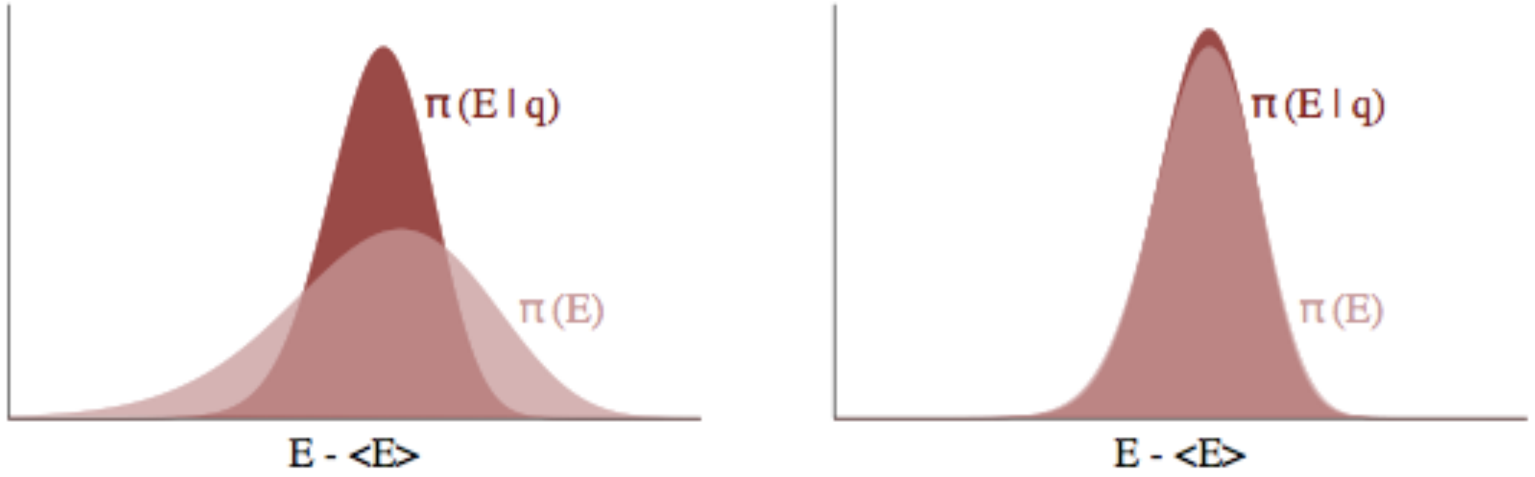
$$\int dpp(p,q) = \int dpp(p|q)p(q) = p(q)\int p(p|q)dp = p(q)$$

.



Tuning:

- The ideal kinetic energy interacts with target, in practice we often use $K(p) = p' M^{-1} p$
- Set inverse mass matrix to the covariance of the target distribution: maximally decorrelate the target. Do in warmup phase.
- use symplectic integration
- need to determine L and ϵ.
- generally static not good, under samples tails (high-energy microcanonicals). Estimate dynamically: NUTS (pymc3 and Stan)



Acceptance probability

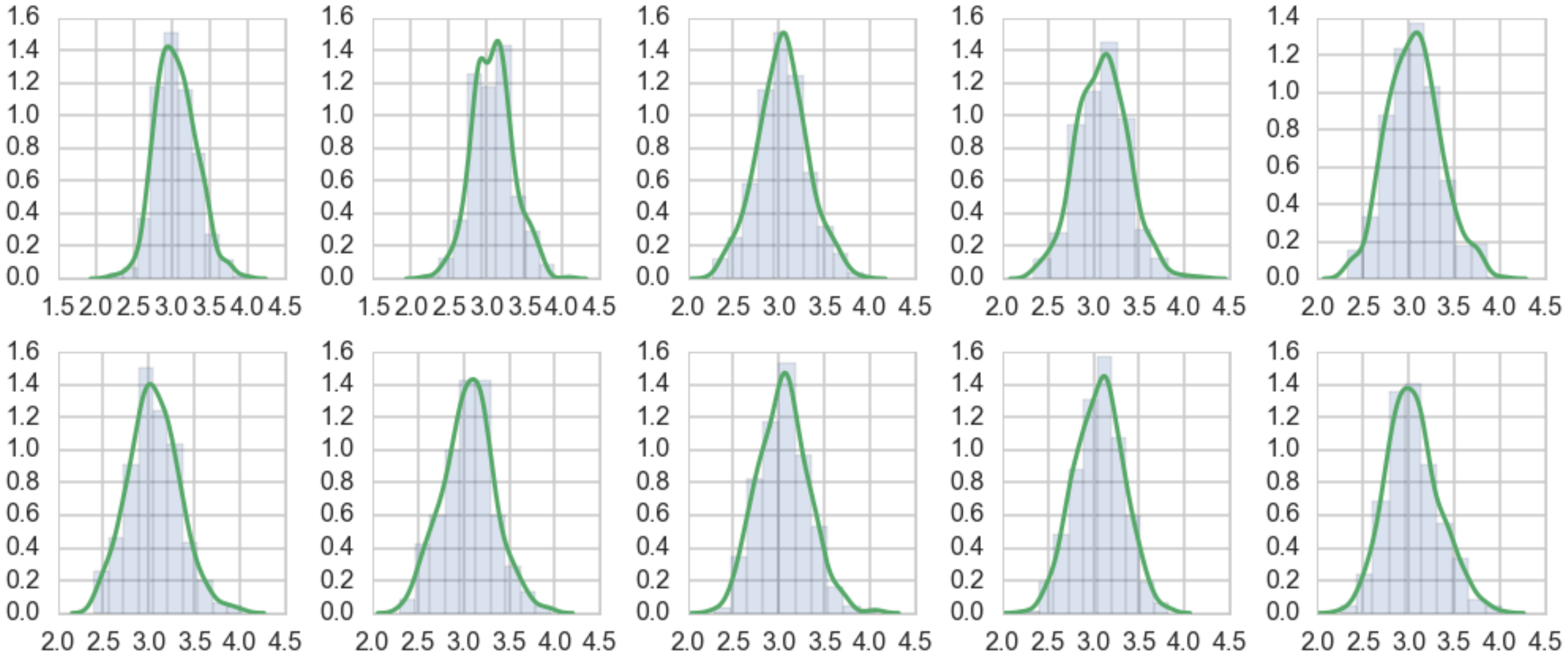
- small symplectic errors means H evolution only forward in time
- tack on sign change $(q, p) \to (q_L, -p_L)$. Superman to the rescue!
- Acceptance: $A = \min[1, \frac{p(q_L, -p_L)\delta(q_L - q_L)\delta(-p_L + p_L)}{p(q,p)\delta(q-q)\delta(p-p)}]$
- More general acceptance in NUTS, sum over all points in orbit

Model convergence

- traces white noisy
- diagnose autocorrelation, check parameter correlations

```
pm.trace_to_dataframe(trace).corr()
```

- visually inspect histogram every m samples
- traceplots from different starting points, different chains
- formal tests: Gewecke, Gelman-Rubin, Effective Sample Size



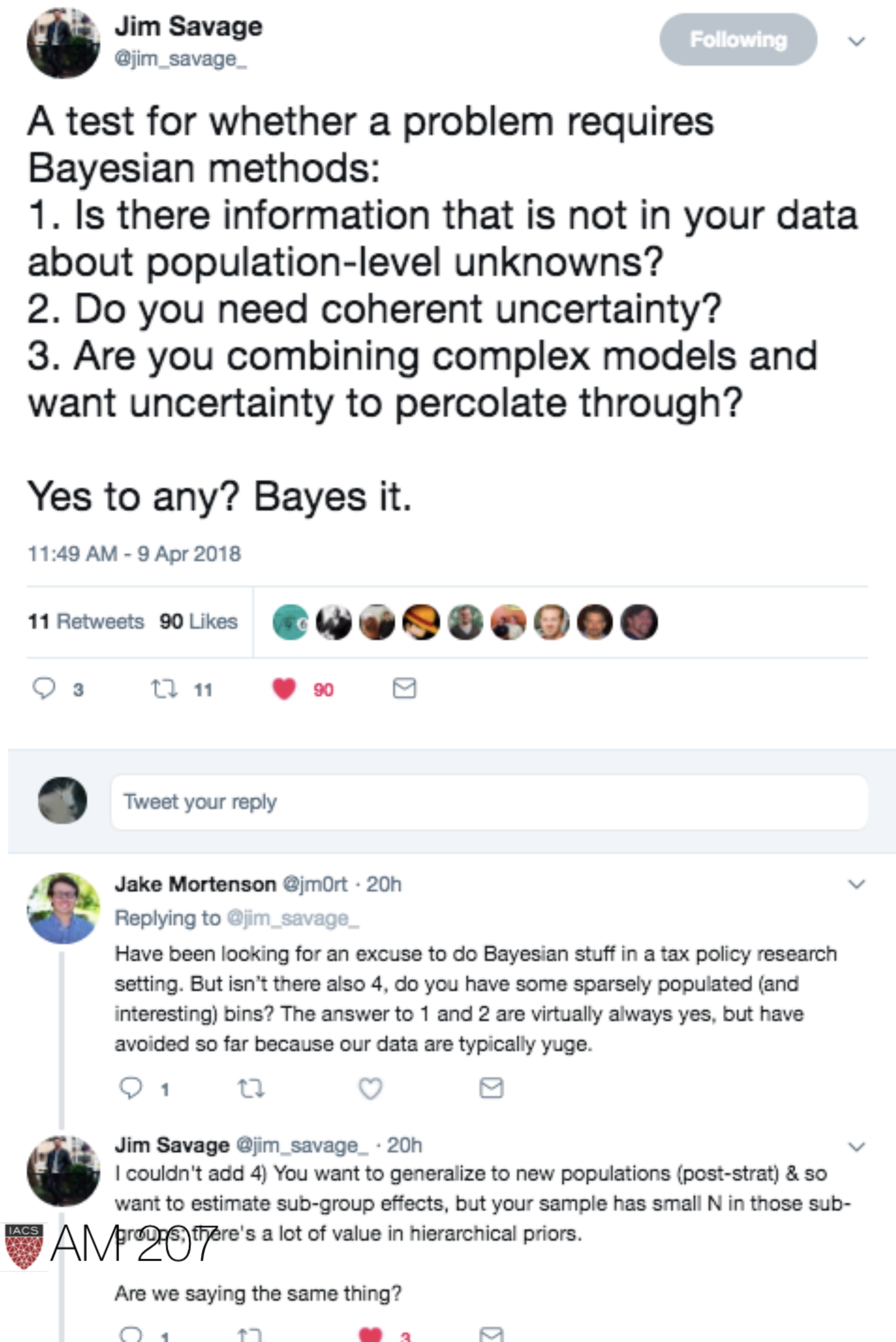
Thoughts on Diagnostics

- be paranoid, you only know you have not converged, not if you have
- what if you missed out an entire lobe? Thus multiple chains and multiple starting points.
- check posterior correlations, trace autocorrelation, effective n, the look of the trace, the acceptance rate
- check gewecke and gelman-rubin

STATS

WHEN BAYES

from Jim Savage



Jim Savage @jim_savage_

A test for whether a problem requires Bayesian methods:
1. Is there information that is not in your data about population-level unknowns?
2. Do you need coherent uncertainty?
3. Are you combining complex models and want uncertainty to percolate through?

Yes to any? Bayes it.

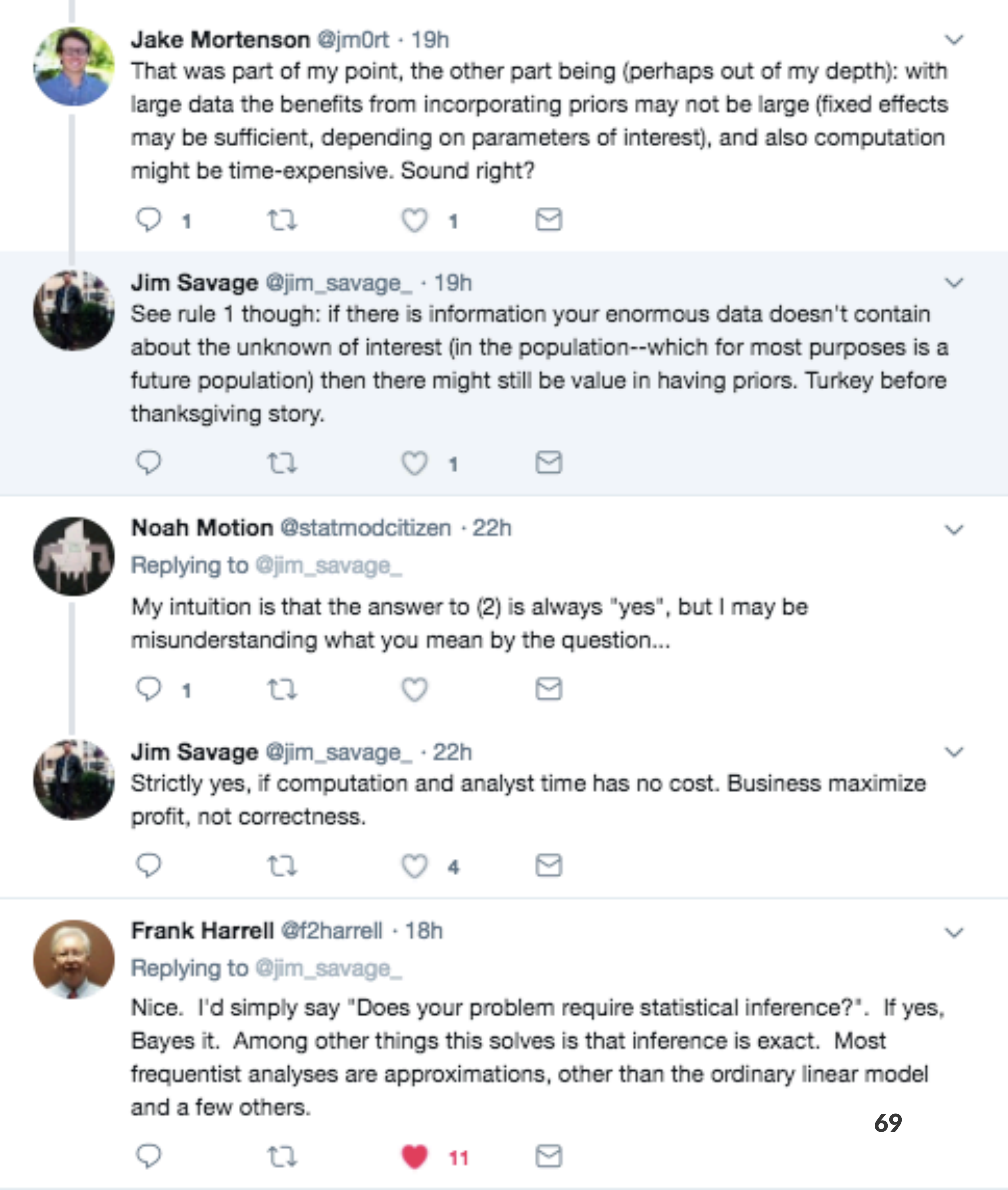
11:49 AM - 9 Apr 2018

Jake Mortenson @jm0rt · 20h
Replying to @jim_savage_

Have been looking for an excuse to do Bayesian stuff in a tax policy research setting. But isn't there also 4, do you have some sparsely populated (and interesting) bins? The answer to 1 and 2 are virtually always yes, but have avoided so far because our data are typically yuge.

Jim Savage @jim_savage_ · 20h
I couldn't add 4) You want to generalize to new populations (post-strat) & so want to estimate sub-group effects, but your sample has small N in those sub-groups, there's a lot of value in hierarchical priors.

Are we saying the same thing?



Jake Mortenson @jm0rt · 19h
That was part of my point, the other part being (perhaps out of my depth): with large data the benefits from incorporating priors may not be large (fixed effects may be sufficient, depending on parameters of interest), and also computation might be time-expensive. Sound right?

Jim Savage @jim_savage_ · 19h
See rule 1 though: if there is information your enormous data doesn't contain about the unknown of interest (in the population--which for most purposes is a future population) then there might still be value in having priors. Turkey before thanksgiving story.

Noah Motion @statmodcitizen · 22h
Replying to @jim_savage_

My intuition is that the answer to (2) is always "yes", but I may be misunderstanding what you mean by the question...

Jim Savage @jim_savage_ · 22h
Strictly yes, if computation and analyst time has no cost. Business maximize profit, not correctness.

Frank Harrell @f2harrell · 18h
Replying to @jim_savage_

Nice. I'd simply say "Does your problem require statistical inference?". If yes, Bayes it. Among other things this solves is that inference is exact. Most frequentist analyses are approximations, other than the ordinary linear model and a few others.

Latent Variables

- dont think of bayes/frequentist, think of observed x /Latent z
- anything unobserved is latent (this is the posterior predictive point of view, x as θ), thus standard bayesian viewpoint: nuisance parameters are latent
- latent factors in matrix factorization, mixtures, recommendations...cluster zs

Generative model

$$p(x, z) = p(x|z)p(z)$$

Bayesian

- sample is the data, and is fixed
- parameter is stochastic, has prior and posterior distribution
- posterior: $p(\theta|y) = \frac{p(y|\theta)\,p(\theta)}{p(y)}$, can summarize via MAP
- just bayes rule: $posterior = \frac{likelihood \times prior}{evidence}$

From `edwardlib`: $p(\mathbf{x} \mid \mathbf{z})$

describes how any data $\mathbf{x}$ depend on the latent variables $\mathbf{z}$.

- **The likelihood posits a data generating process**, where the data $\mathbf{x}$ are assumed drawn from the likelihood conditioned on a particular hidden pattern described by $\mathbf{z}$.
- The *prior* $p(\mathbf{z})$ is a probability distribution that describes the latent variables present in the data. **The prior posits a generating process of the hidden structure**.

- prior-predictive = evidence: $p(y) = E_{p(\theta)}[\mathcal{L}] = \int d\theta p(y|\theta)p(\theta)$ a normalization, irrelevant for sampling, useful for EB
- What if θ is multidimensional? Marginal posterior: $p(\theta_1|D) = \int d\theta_{-1} p(\theta|D)$.
- posterior predictive: the distribution of a future data point y^*:

$$p(y^*|D=\{y\}) = E_{p(\theta|D)}[p(y|\theta)] = \int d\theta p(y^*|\theta)p(\theta|\{y\}).$$

Marginalization

Marginal posterior:

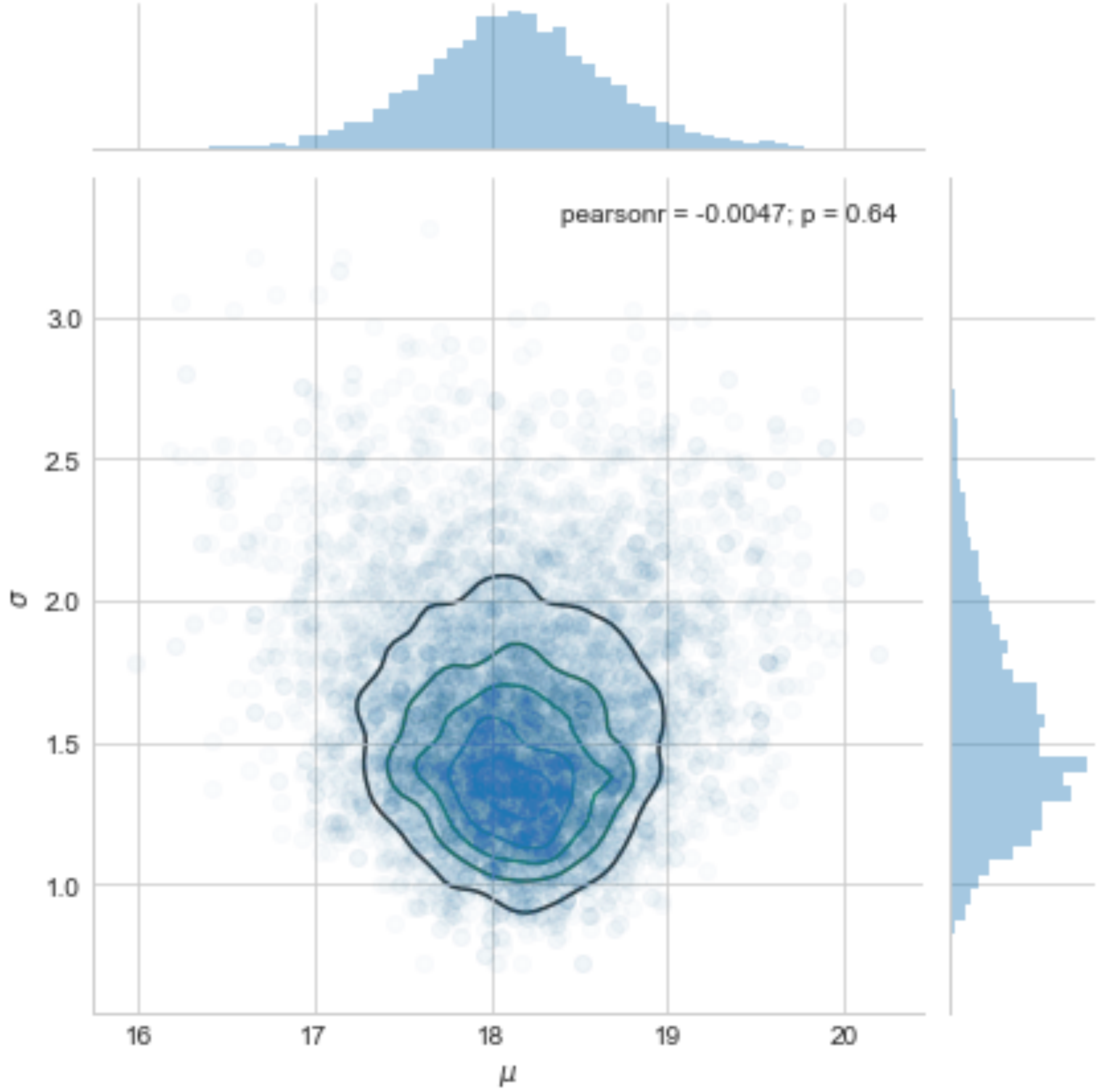
$$p(\theta_1|D) = \int d\theta_{-1} p(\theta|D).$$

```
samps[20000::,:].shape #(10001, 2)
```

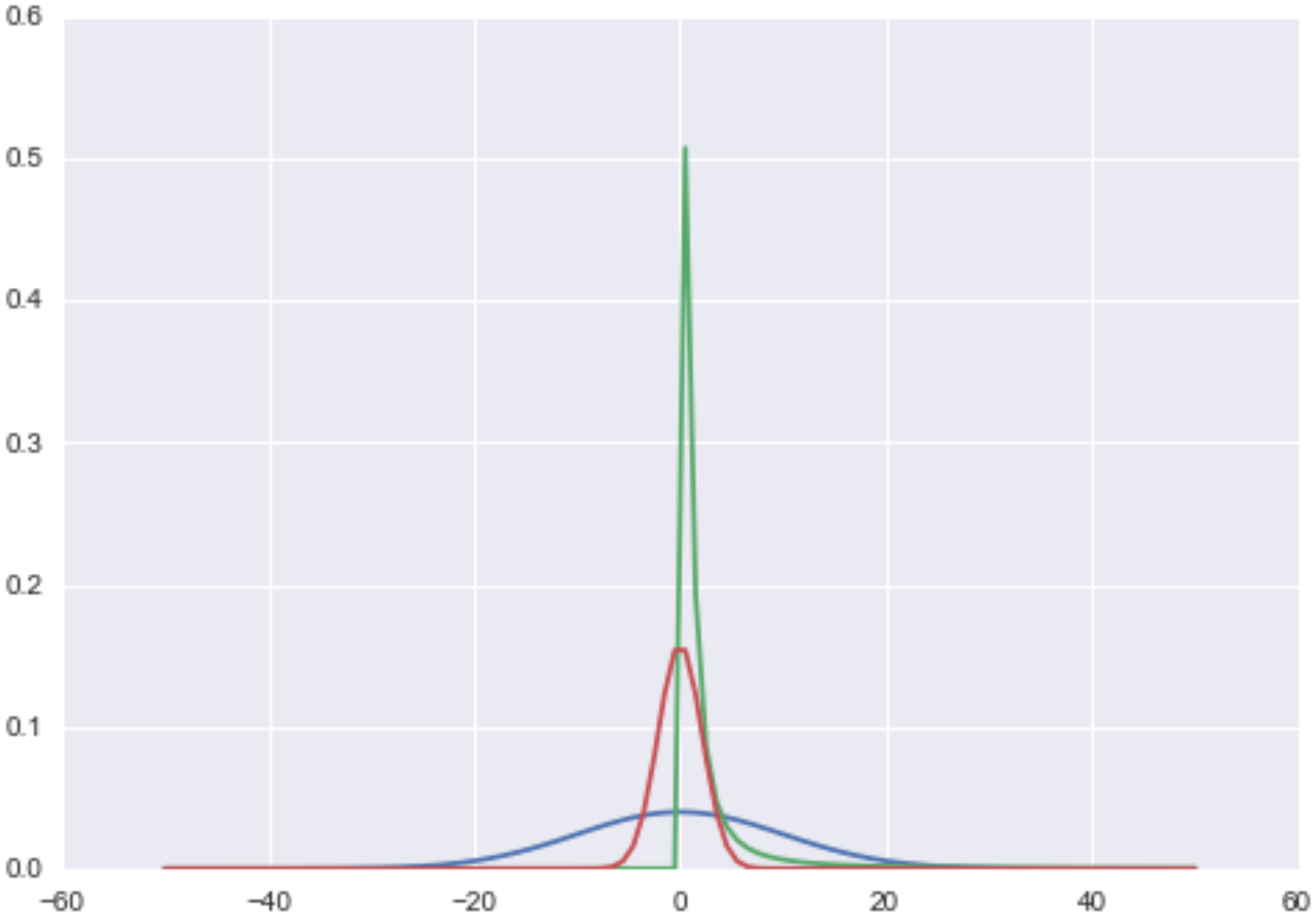
```
sns.jointplot(
    pd.Series(samps[20000::,0], name="$\mu$"),
    pd.Series(samps[20000::,1], name="$\sigma$"),
    alpha=0.02)
    .plot_joint(
        sns.kdeplot,
    zorder=0, n_levels=6, alpha=1)
```

Marginals are just 1D histograms

```
plt.hist(samps[20000::,0])
```



priors



- choose likelihoods with MAXENT
- choose priors as non-informative, e.g. uniform or Jeffreys
- better still: choose priors as weakly informative/regularizing
- helps with sampler performance
- see https://github.com/stan-dev/stan/wiki/Prior-Choice-Recommendations and Stan Manual

Normal-Normal Model

Posterior for a gaussian likelihood:

$$p(\mu, \sigma^2 | y_1, \ldots, y_n, \sigma^2) \propto \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}\sum (y_i - \mu)^2} p(\mu, \sigma^2)$$

What is the posterior of μ assuming we know σ^2?

Prior for σ^2 is $p(\sigma^2) = \delta(\sigma^2 - \sigma_0^2)$

$$p(\mu|y_1,\ldots,y_n,\sigma^2=\sigma_0^2) \propto p(\mu|\sigma^2=\sigma_0^2)\, e^{-\frac{1}{2\sigma_0^2}\sum(y_i-\mu)^2}$$

The conjugate of the normal is the normal itself.

Say we have the prior

$$p(\mu|\sigma^2) = \exp\left\{-\frac{1}{2\tau^2}(\hat{\mu}-\mu)^2\right\}$$

posterior: $p(\mu|y_1,\ldots,y_n,\sigma^2) \propto \exp\left\{-\frac{a}{2}(\mu-b/a)^2\right\}$

Here

$$a = \frac{1}{\tau^2} + \frac{n}{\sigma_0^2}, \qquad b = \frac{\hat{\mu}}{\tau^2} + \frac{\sum y_i}{\sigma_0^2}$$

Define $\kappa = \sigma^2/\tau^2$

$$\mu_p = \frac{b}{a} = \frac{\kappa}{\kappa+n}\hat{\mu} + \frac{n}{\kappa+n}\bar{y}$$

which is a weighted average of prior mean and sampling mean.

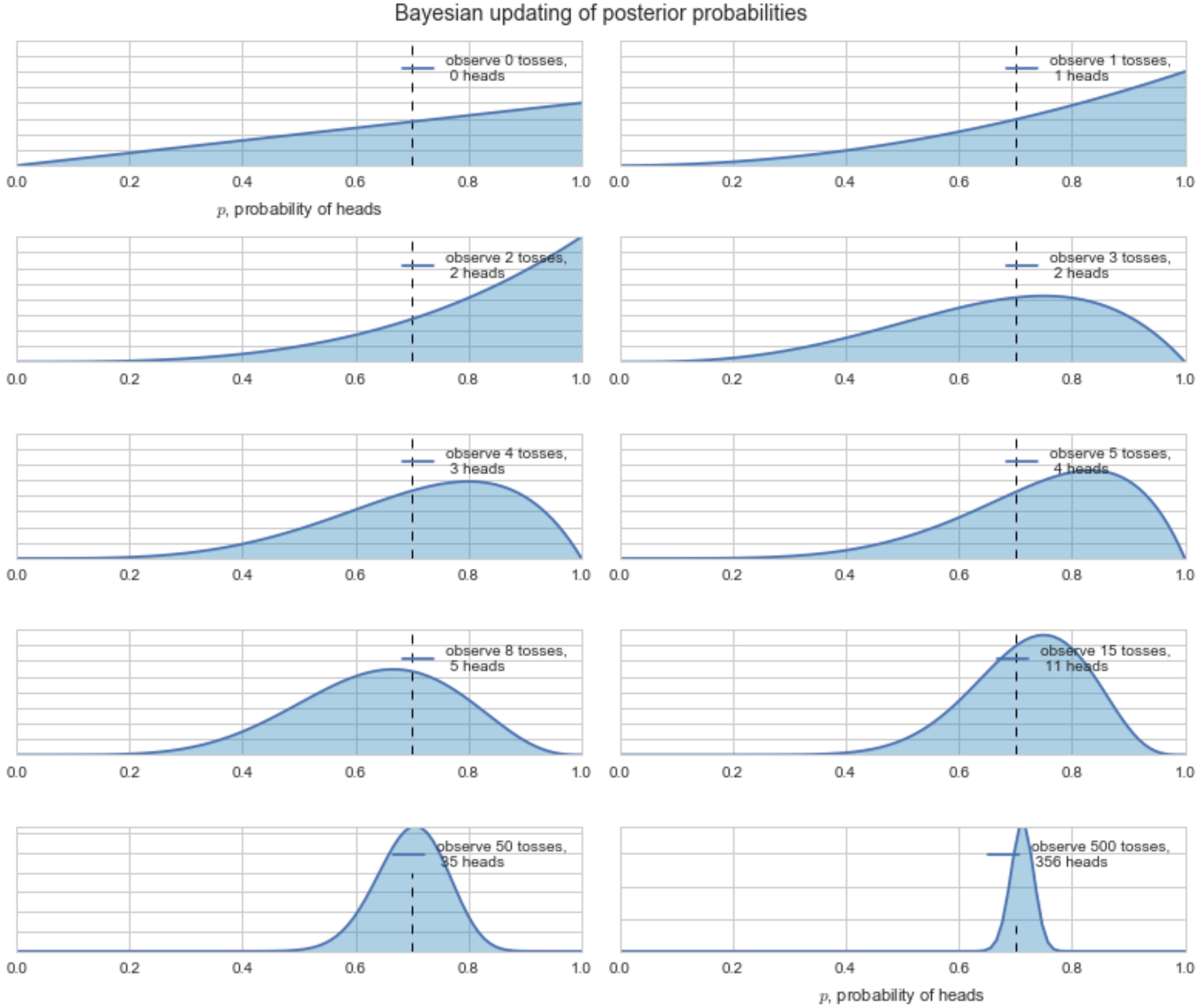
The variance is

$$\tau_p^2 = \frac{1}{1/\tau^2 + n/\sigma^2}$$

or better

$$\frac{1}{\tau_p^2} = \frac{1}{\tau^2} + \frac{n}{\sigma^2}.$$

as n increases, the data dominates the prior and the posterior mean approaches the data mean, with the posterior distribution narrowing...


Bayesian updating of posterior probabilities
observe 0 tosses, 0 heads
observe 1 tosses, 1 heads
observe 2 tosses, 2 heads
observe 3 tosses, 2 heads
observe 4 tosses, 3 heads
observe 5 tosses, 4 heads
observe 8 tosses, 5 heads
observe 15 tosses, 11 heads
observe 50 tosses, 35 heads
observe 500 tosses, 356 heads
0.0
0.2
0.4
0.6
0.8
1.0
p, probability of heads
p, probability of heads

Bayesian Updating "on-line"

- as each piece of data comes in, you update the prior by multiplying by the one-point likelihood.

- the posterior you get becomes the prior for our next step

$$p(\theta \mid \{y_1, \ldots, y_{n+1}\}) \propto p(\{y_1, \ldots, y_n\} \mid \theta) \times p(\theta \mid \{y_1, \ldots, y_n\})$$

- the posterior predictive is the distribution of the next data point!

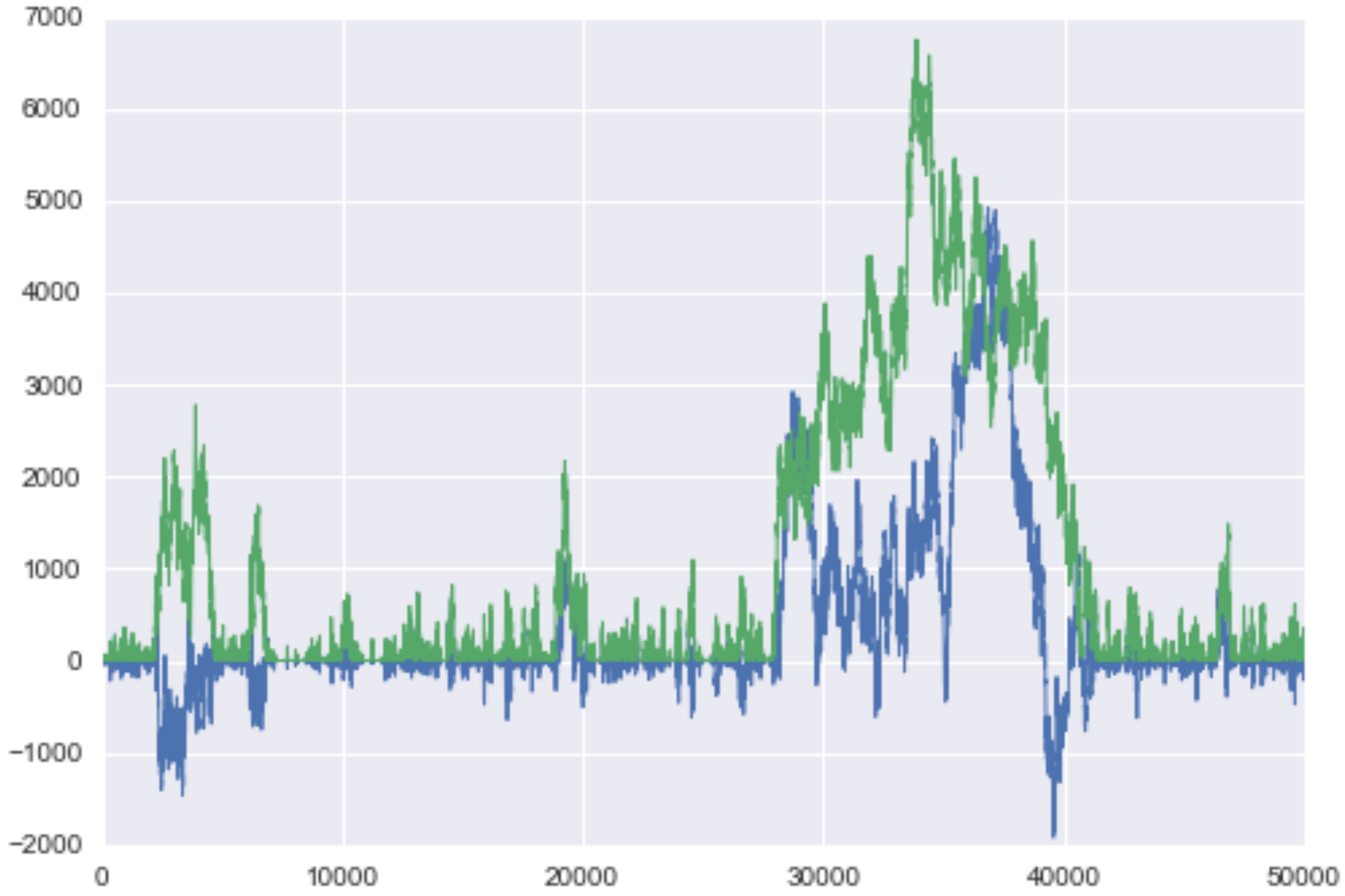
$$p(y_{n+1} | \{y_1, \ldots y_n\}) = E_{p(\theta|\{y_1, \ldots y_n\})}[p(y_{n+1}|\theta)] = \int d\theta p(y_{n+1}|\theta) p(\theta|\{y_1, \ldots y_n\})$$

Weakly informative or regularizing priors

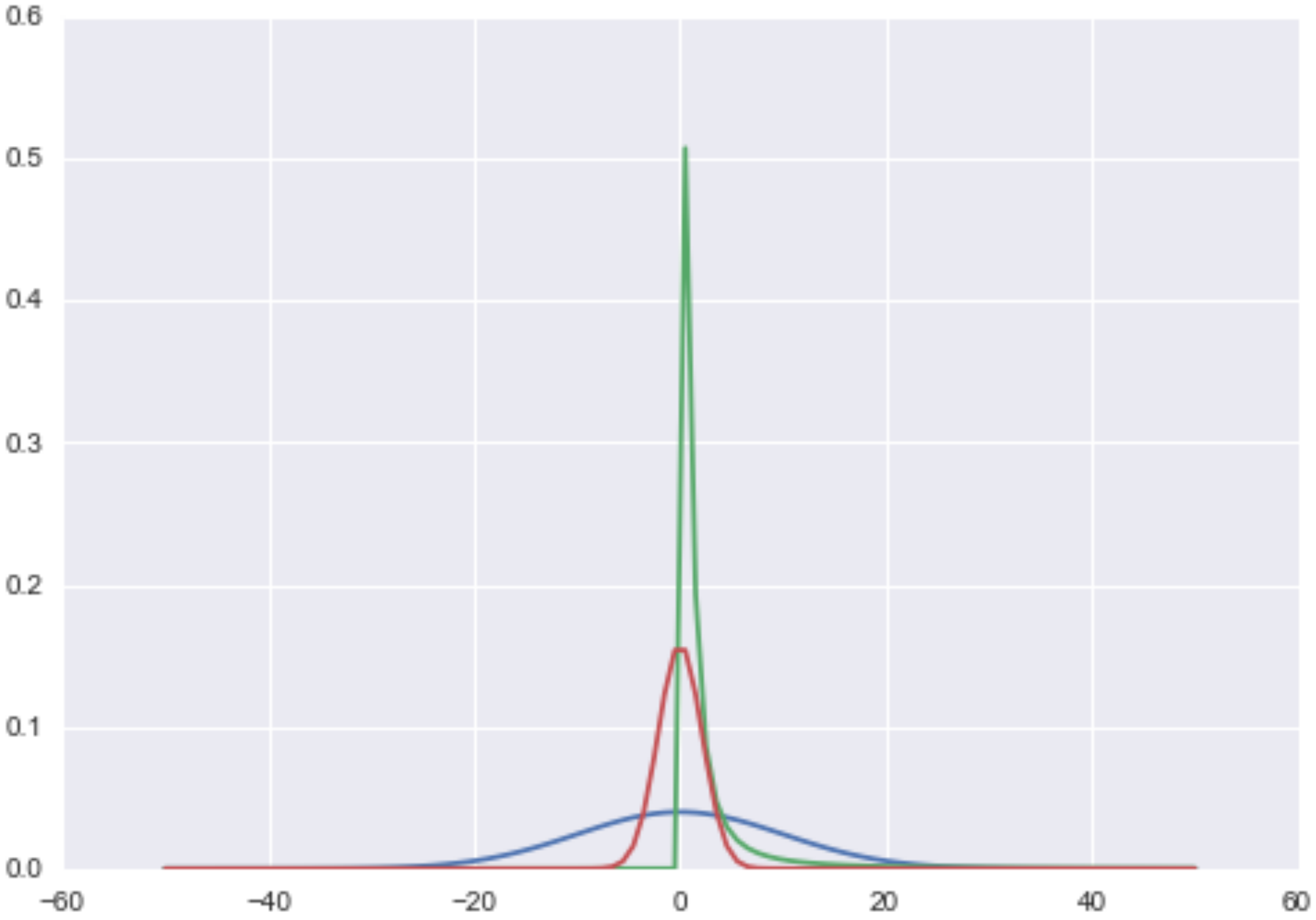
- these are the priors we will concern ourselves most with
- restrict parameter ranges
- help samplers
- regularizing priors may use the data "twice" as we shall see

Normal model Example

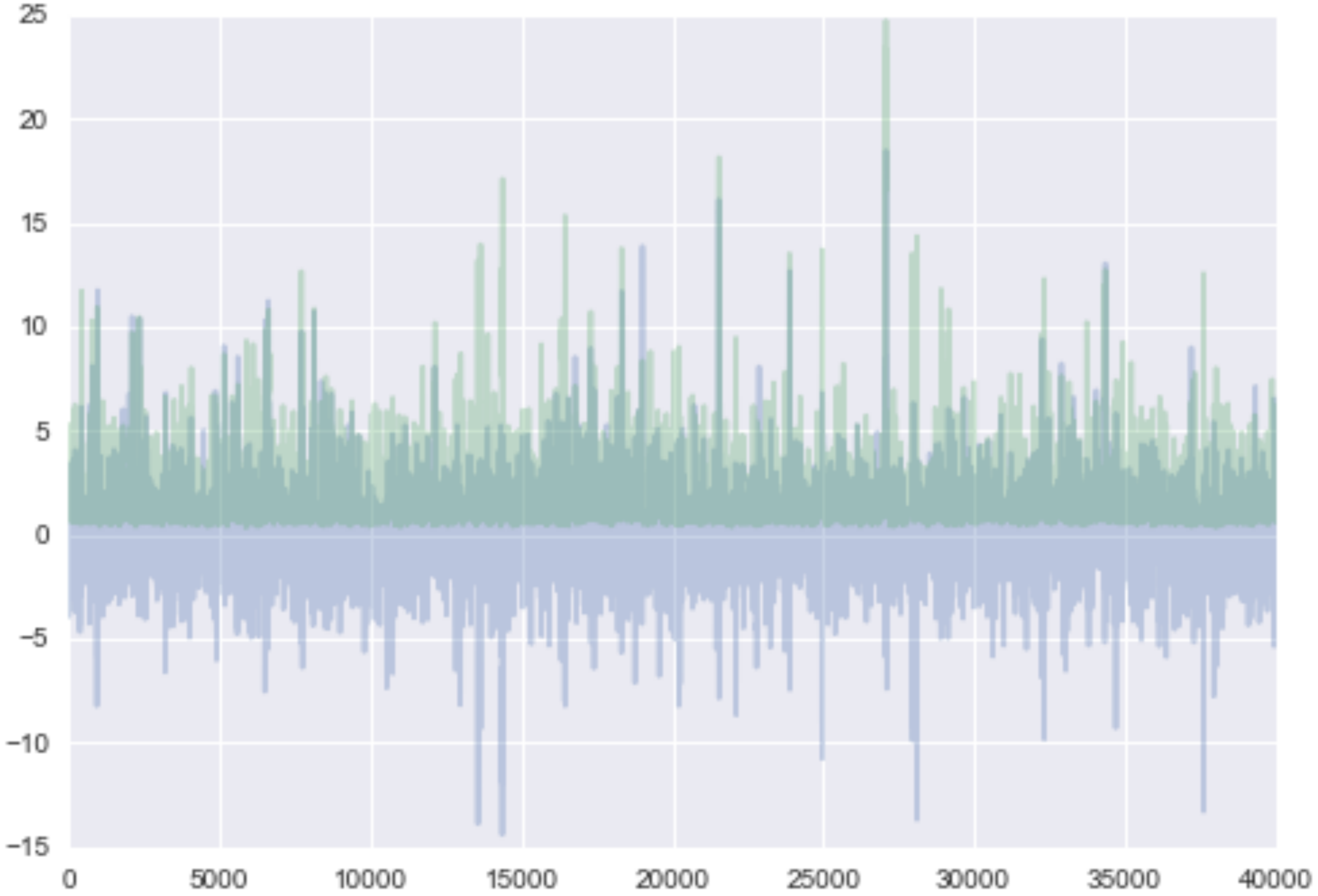
- two data points 1 and -1
- flat improper priors on $\mu, \sigma > 0$
- model drifts wildly as less data
- flat priors say extreme implausible values quite likely
- extreme drifts overwhelm chain

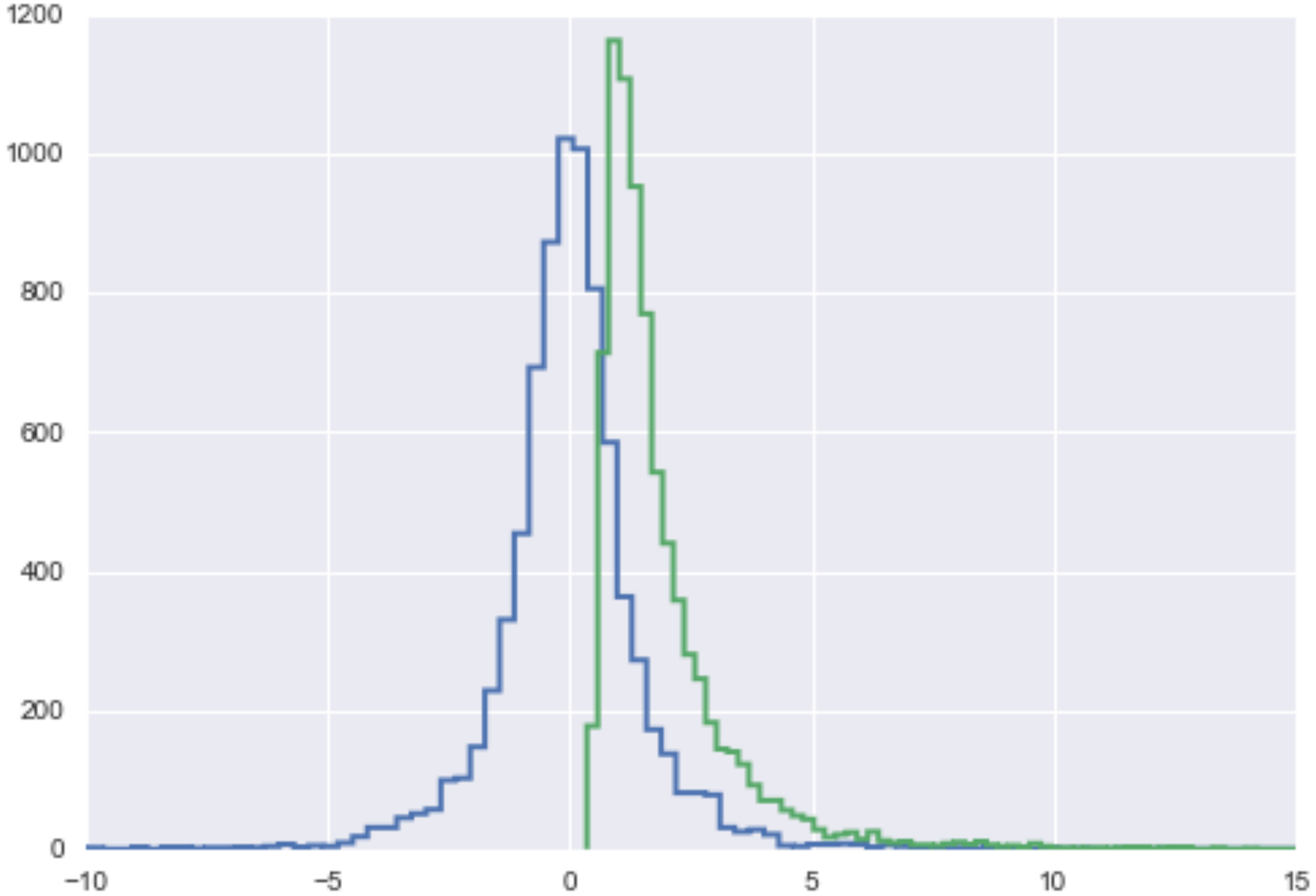


weakly regularizing priors



- choose $\mu \sum N(0, 10)$
- choose $\sigma \sim HalfCauchy(0, 1)$
- lets mean vary widely but not crazily
- HalfCauchy lets variance be positive and occasionally can have high value samples


25
20
15
10
5
0
−5
−10
−15
0
5000
10000
15000
20000
25000
30000
35000
40000


1200
1000
800
600
400
200
0
−10
−5
0
5
10
15

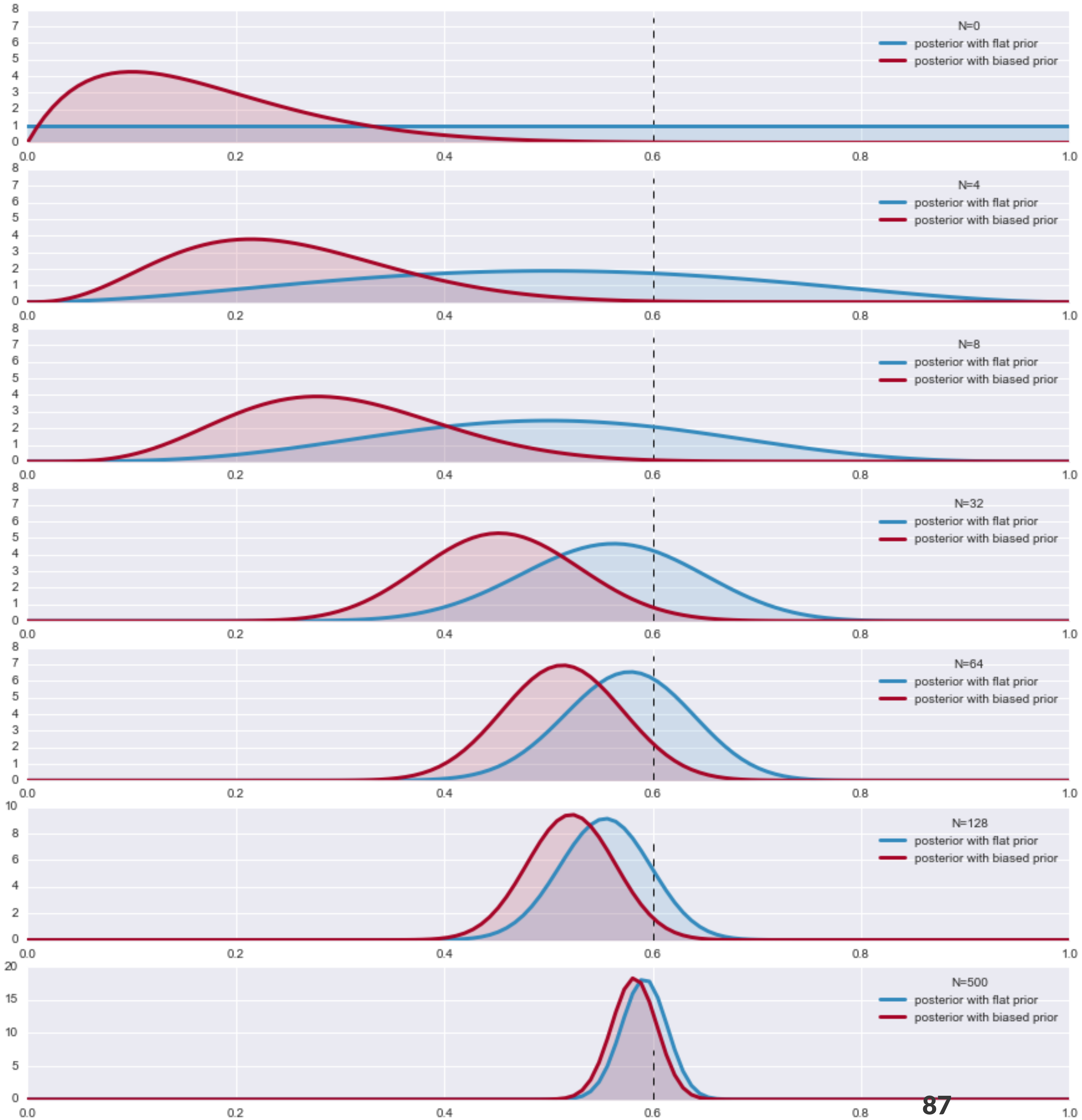
Data Overwhelms priors

Define $\kappa = \sigma^2/\tau^2$

$$\mu_p = \frac{b}{a} = \frac{\kappa}{\kappa + n}\hat{\mu} + \frac{n}{\kappa + n}\bar{y}$$

$$\frac{1}{\tau_p^2} = \frac{1}{\tau^2} + \frac{n}{\sigma^2}.$$

- priors regularize data for small data
- but large data overwhelms priors



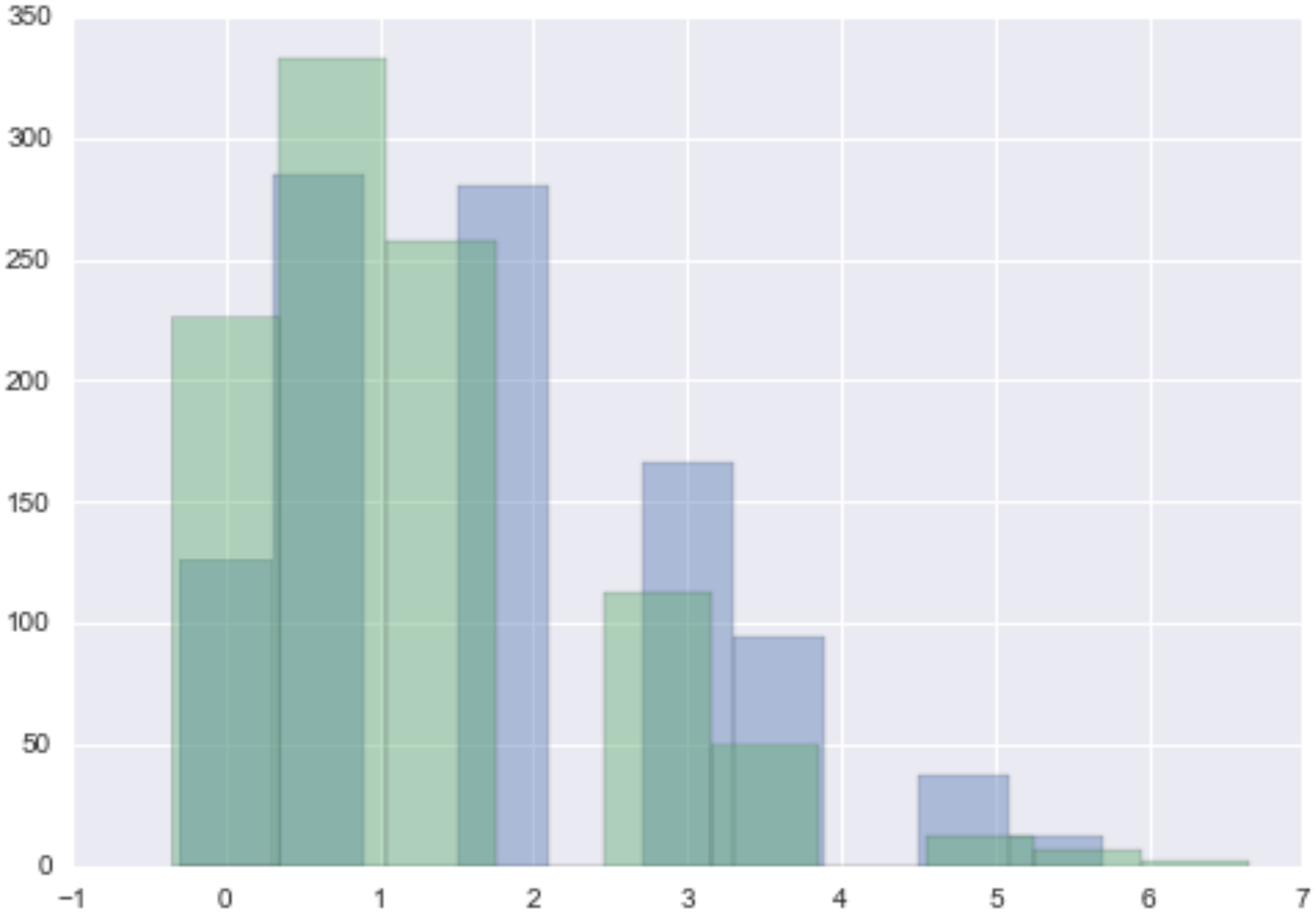
Exchangeability

Lets assume that the number of children of a women in any one of these classes can me modelled as coming from ONE birth rate.

The in-class likelihood for these women is invariant to a permutation of variables.

This is really a statement about what is IID and what is not.

It depends on how much knowledge you have...



Posterior Predictives

$$p(y^*|D) = \int d\theta p(y^*|\theta)p(\theta|D)$$

Sampling easy (mothers poisson-gamma):

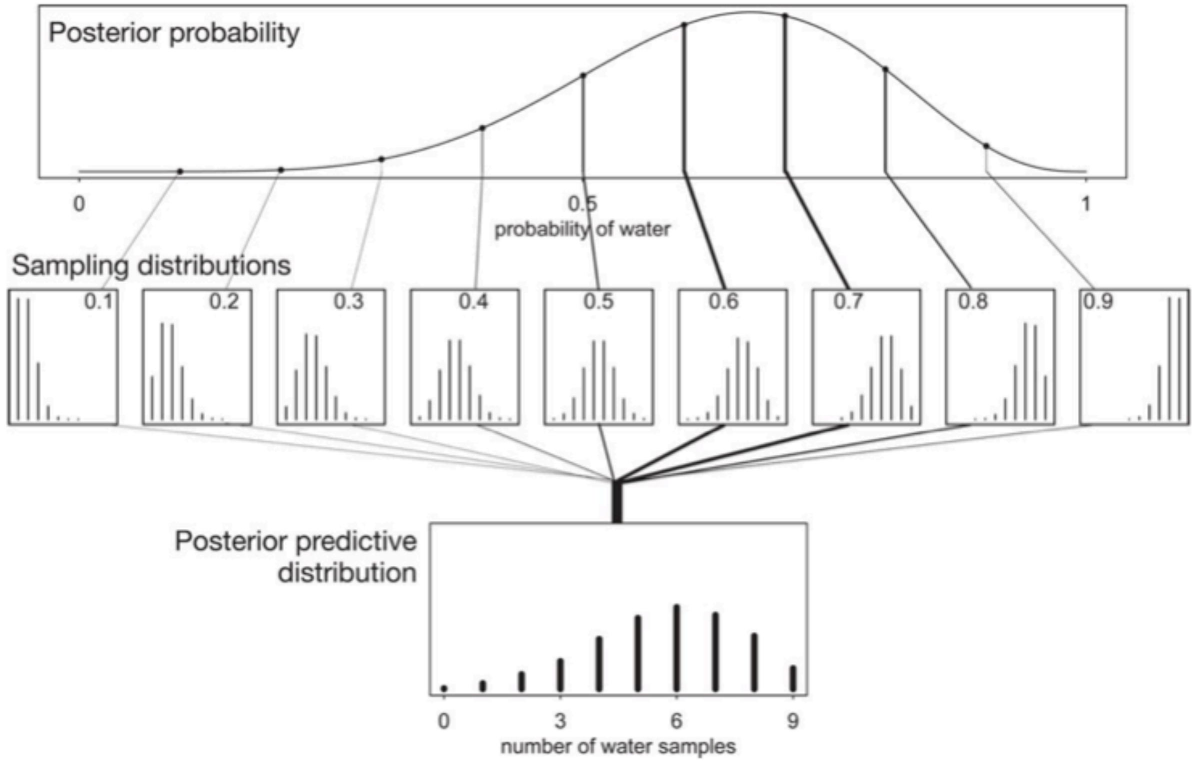
```
postpred1 = poisson.rvs(theta1trace)
postpred2 = poisson.rvs(theta2trace)
```

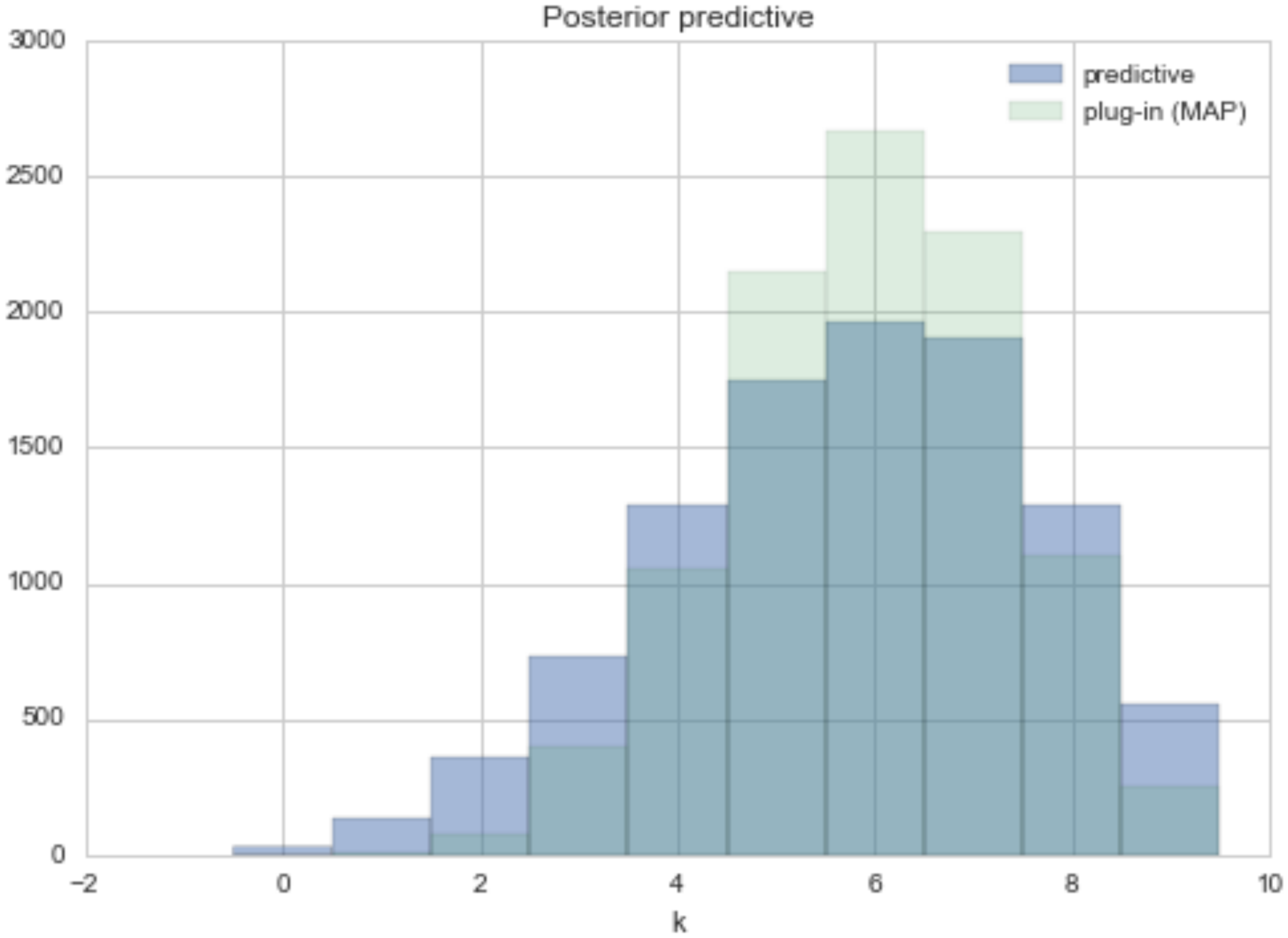
Exact: Negative Binomial (requires math):

$$E[y^*] = \frac{(a + \sum y_i)}{(b + N)}$$

$$var[y^*] = \frac{(a + \sum y_i)}{(b + N)^2}(N + b + 1).$$

Posterior Predictive Smear



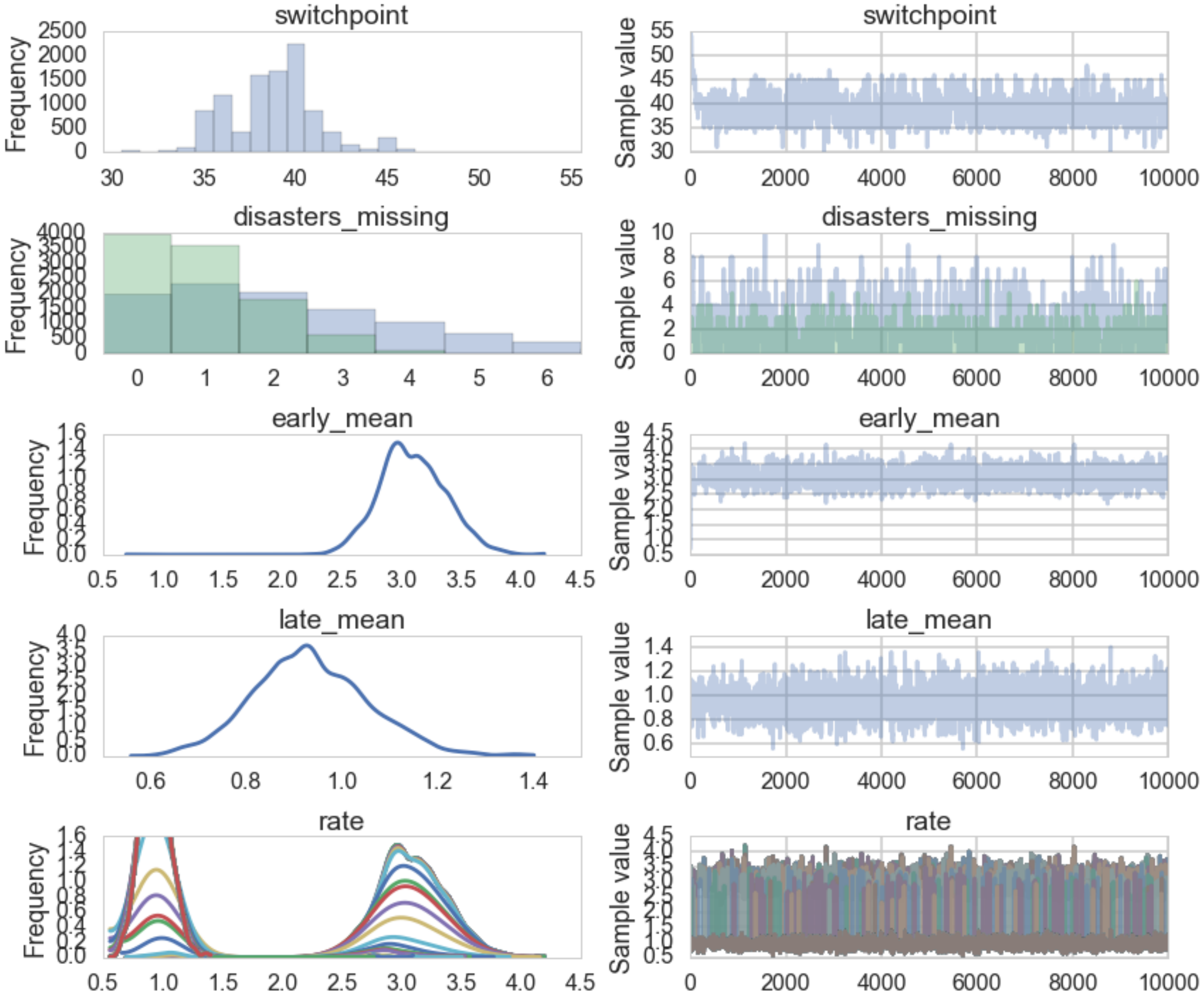


pp vs sampling distrib at MAP →

Missing Data can be imputed with the predictive

```
disasters_masked = np.ma.masked_values(disasters_missing, value=-999)
disasters = pm.Poisson('disasters', rate, observed=disasters_masked)
with missing_data_model:
    stepper=pm.Metropolis()
    trace_missing = pm.sample(10000, step=stepper)

pm.summary(trace_missing, varnames=['disasters_missing'])
```



Bayes Action

First define the distribution-averaged utility:

$$\bar{u}(a) = \int d\omega \, u(a, \omega) \, p(\omega | D)$$

We then find the a that maximizes this utility:

$$\hat{a} = \arg\max_a \bar{u}(a)$$

This action is called the **bayes action**.

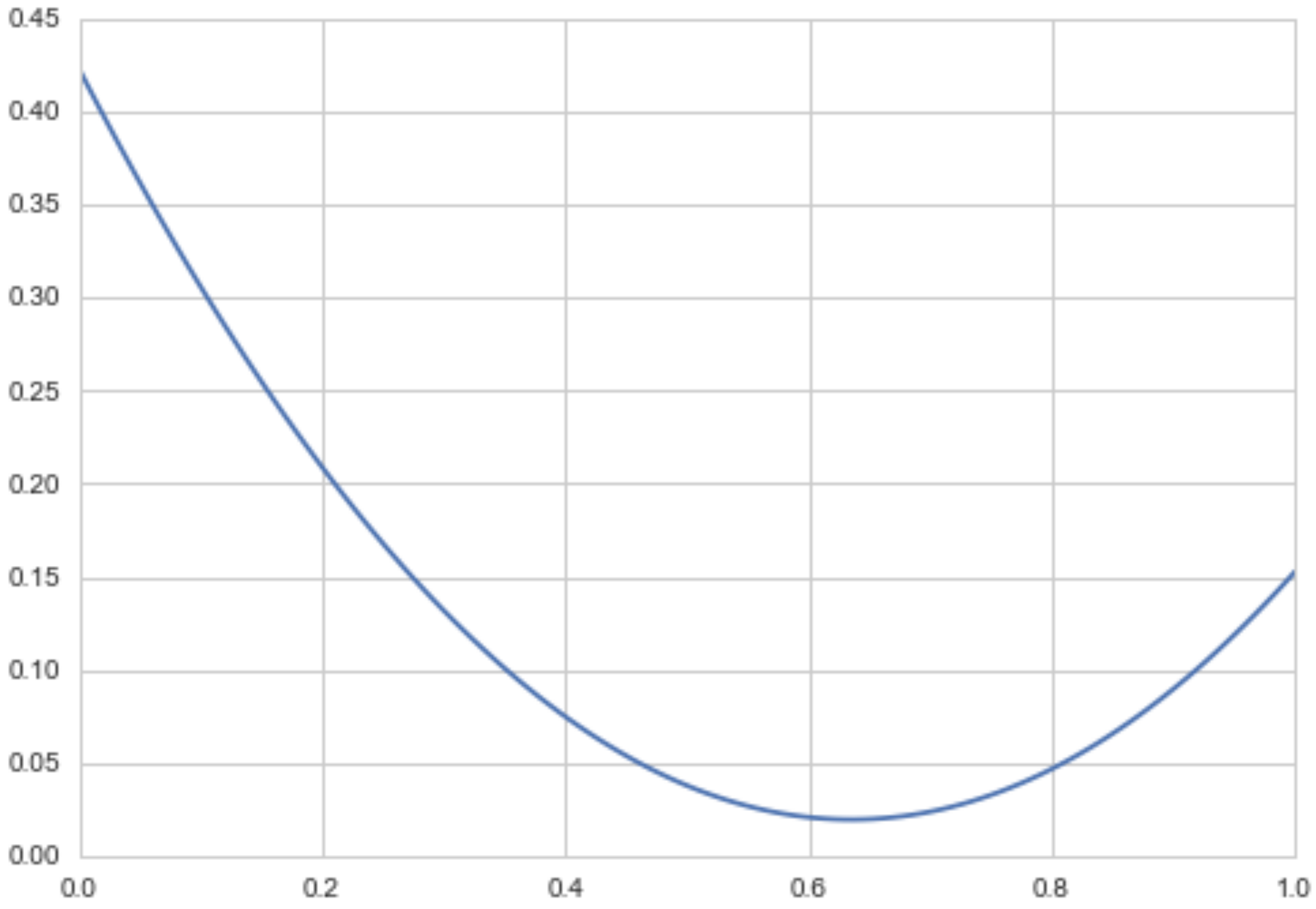
Posterior Mean minimizes squared loss

$$R(t) = E_{p(\theta|D)}[(\theta - t)^2] = \int d\theta (\theta - t)^2 p(\theta|D)$$

$$\frac{dR(t)}{dt} = 0 \implies t = \int d\theta \theta \, p(\theta|D)$$

```
mse = [np.mean((xi-samples)**2) for xi in x]
plt.plot(x, mse);
```

This is **Decision Theory**.

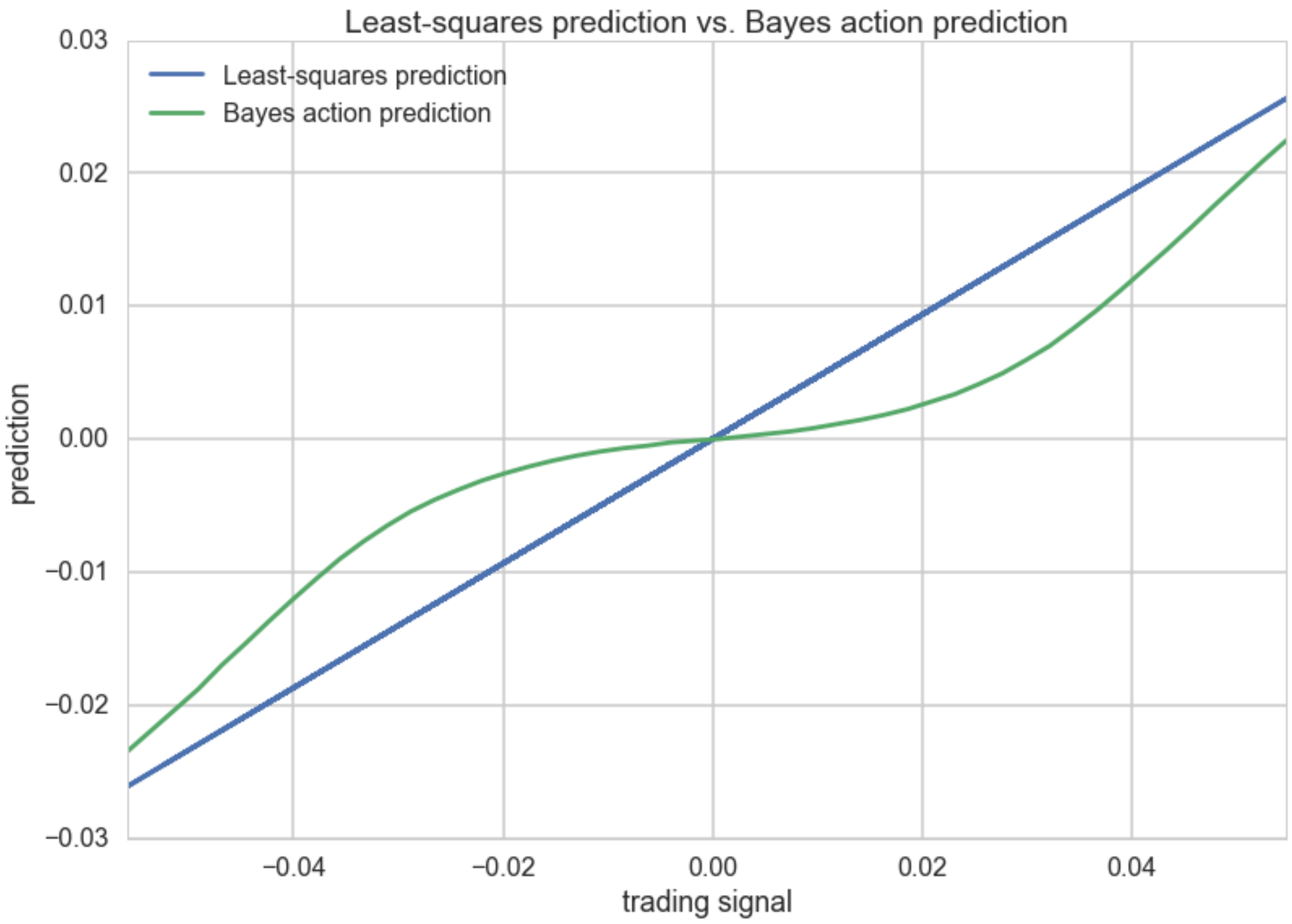


AM 207

Custom Loss

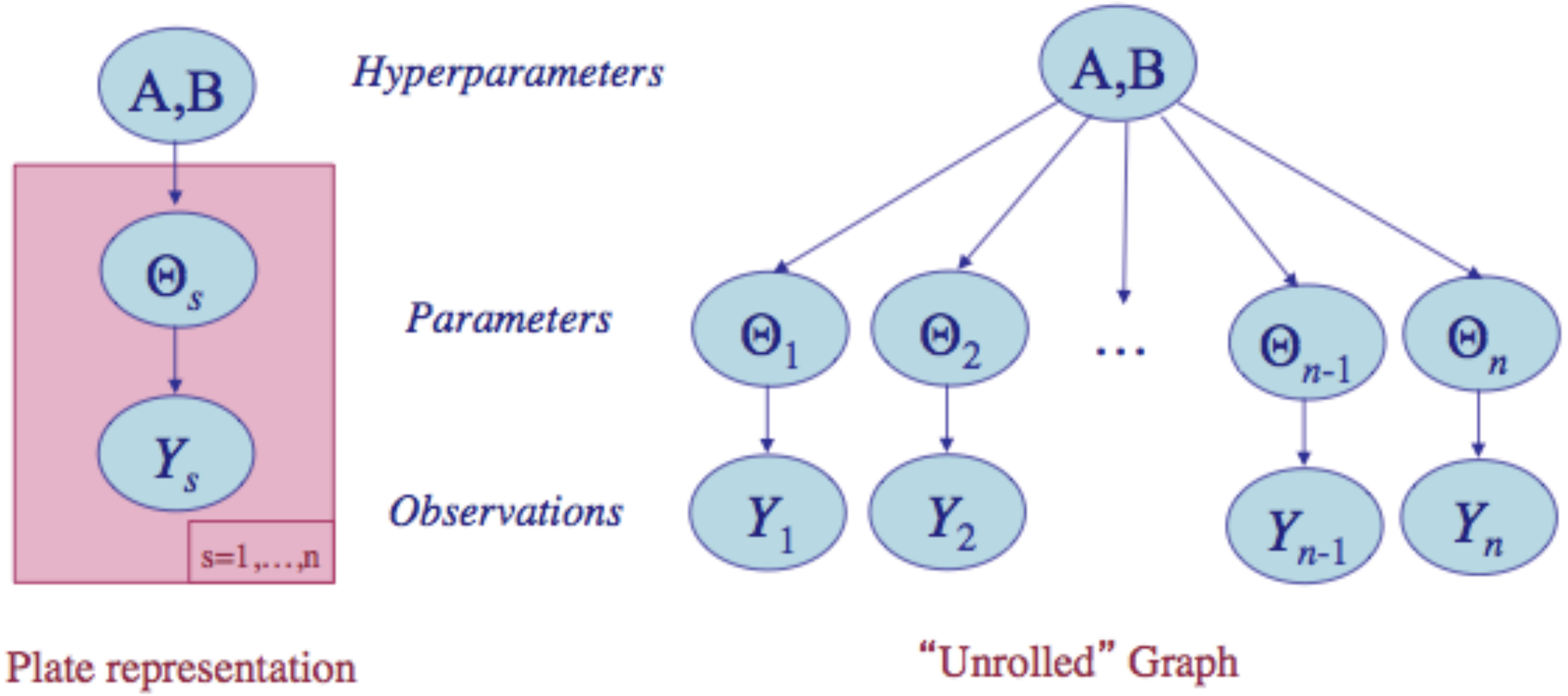
```
def stock_loss(stock_return, pred, alpha = 100.):
    if stock_return * pred < 0:
        #opposite signs, not good
        return alpha*pred**2 - np.sign(stock_return)*pred \
                        + abs(stock_return)
    else:
        return abs(stock_return - pred)
#posterior predictive samples at every x
possible_outcomes = lambda signal: alpha_samples + \
    beta_samples*signal + noise


opt_predictions = np.zeros(50)
trading_signals =  np.linspace(X.min(), X.max(), 50)
for i, _signal in enumerate(trading_signals):
        _possible_outcomes = possible_outcomes(_signal)
        #expected loss over posterior predictive
        tomin = lambda pred: stock_loss(_possible_outcomes, pred).mean()
        #bayes action minimizes expected loss
        opt_predictions[i] = fmin(tomin, 0, disp = False)
```



HIERARCHICAL MODELS

Partial pooling: Hierarchical Model



θ_is drawn from "population distribution" given by a conjugate Beta prior $Beta(\alpha, \beta)$ with **hyperparameters** α and β.

$$\theta_i \sim Beta(\alpha, \beta).$$

$$p(\Theta|\alpha, \beta) = \prod_{i=1}^{70} Beta(\theta_i, \alpha, \beta).$$

Priors from data

Where do α and β come from?

Why are we calling them hyperparameters?

So far have assumed α and β known in priors to be weakly informative.

New idea: estimate priors from data. Looks like a cross-validation like setup.

Key Idea: Share statistical strength

- Some **units** (experiments) statistically more robust
- Non-robust experiments have smaller samples or outlier like behavior
- Borrow strength from all the data as a whole through the estimation of the hyperparameters
- **regularized partial pooling model** in which the "lower" parameters (θs) tied together by "upper level" hyperparameters.

Empirical Bayes or Type-2 Likelihood

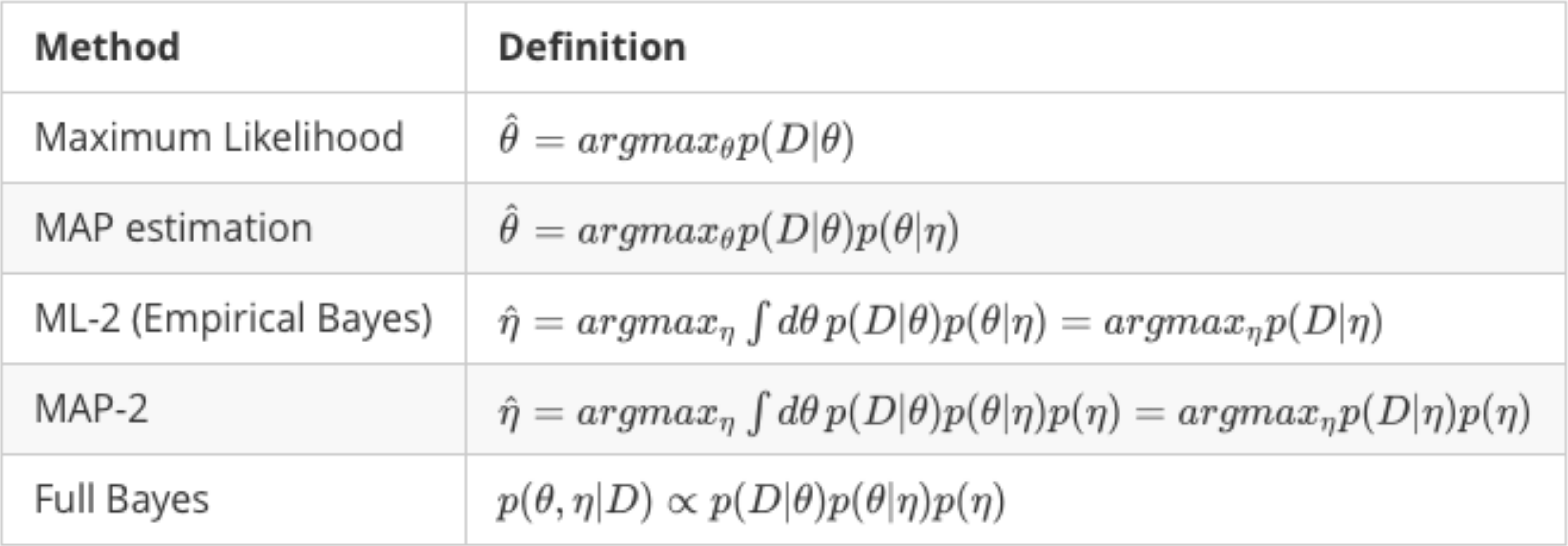
Posterior-predictive distribution, as a function of upper level parameters $\eta = (\alpha, \beta)$.

$$p(y^* | D, \eta) = \int d\theta \, p(y^* | \theta) \, p(\theta | D, \eta)$$

A likelihood with parameters η and simply use maximum-likelihood with respect to η to estimate these η using our "data" y^*

Used in GPs, even can be sampled from

Levels of Bayes



Method	Definition
Maximum Likelihood	$\hat{\theta} = argmax_{\theta} p(D\|\theta)$
MAP estimation	$\hat{\theta} = argmax_{\theta} p(D\|\theta) p(\theta\|\eta)$
ML-2 (Empirical Bayes)	$\hat{\eta} = argmax_{\eta} \int d\theta\, p(D\|\theta) p(\theta\|\eta) = argmax_{\eta} p(D\|\eta)$
MAP-2	$\hat{\eta} = argmax_{\eta} \int d\theta\, p(D\|\theta) p(\theta\|\eta) p(\eta) = argmax_{\eta} p(D\|\eta) p(\eta)$
Full Bayes	$p(\theta, \eta\|D) \propto p(D\|\theta) p(\theta\|\eta) p(\eta)$

Full Bayes

- Fix α and β, we have a Gibbs step for all of the θ_is
- For α and β, everything else fixed, use stationary metropolis step, as conditionals are not isolatable to simply sampled distributions
- when we sample for α, we will propose a new value using a normal proposal, while holding all the θs and β constant at the old value. ditto for β.

Howto Sampling

- a DAG, with observations at the bottom of a tree, next layer intermediate parameters, upper layers hyper-parameters
- sample conditionals from parents up the tree.
- general structure is sampling steps inside Gibbs
- stan, pymc3 all have this structure

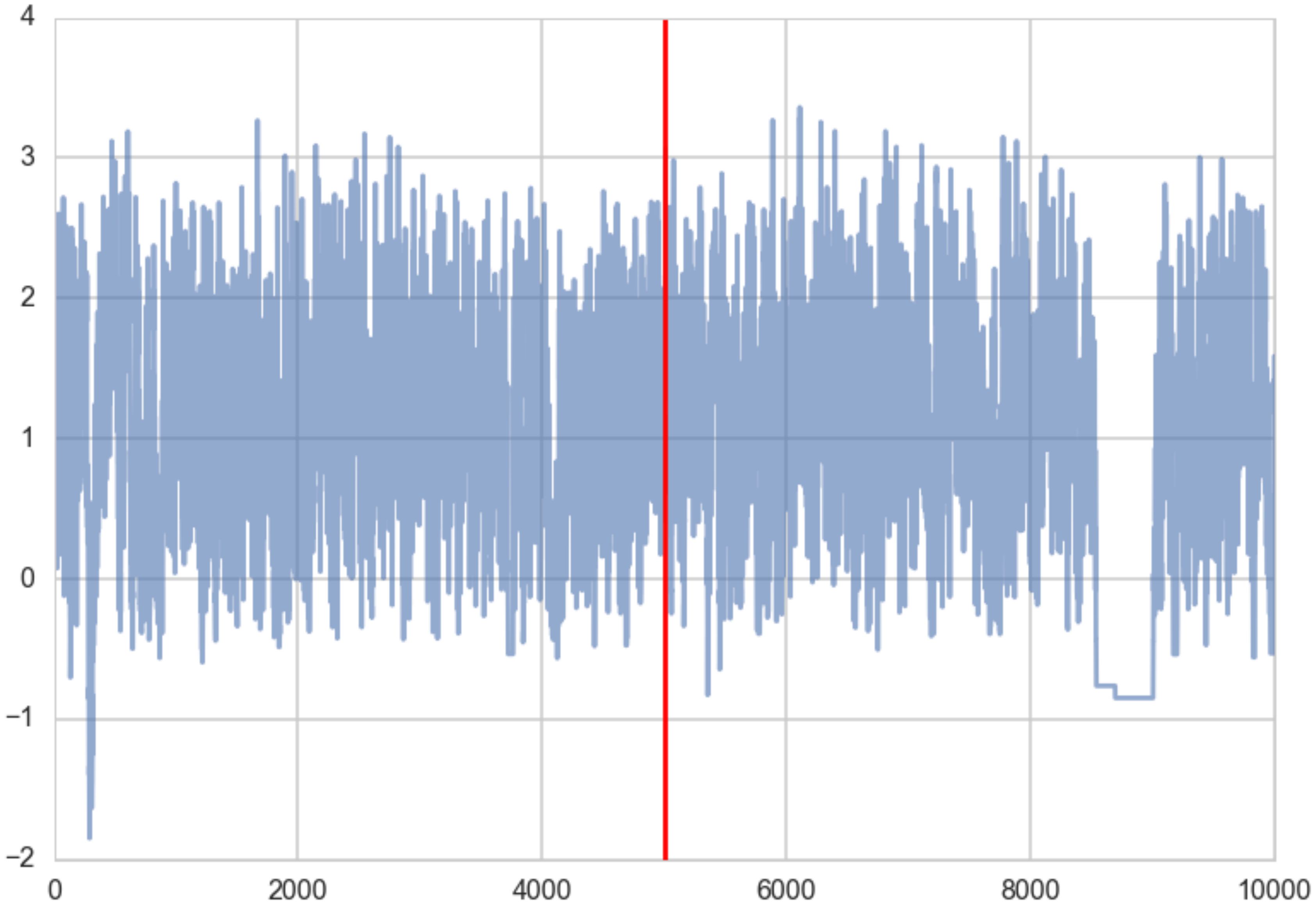
Hierarchy organizes exchangeability

- we use the notion of exchangeability at the level of 'units'.
- for our rats, the y_j were exchangeable since we had no additional information about experimental conditions.
- if specific groups of experiments came from specific laboratories, assume experiments interchangeable if from the same lab.
- lab specific α_{lab} and β_{lab} parameters
- add another level of hierarchy to draw these from hyperprior.

Centered Hierarchical Normal-Normal Model

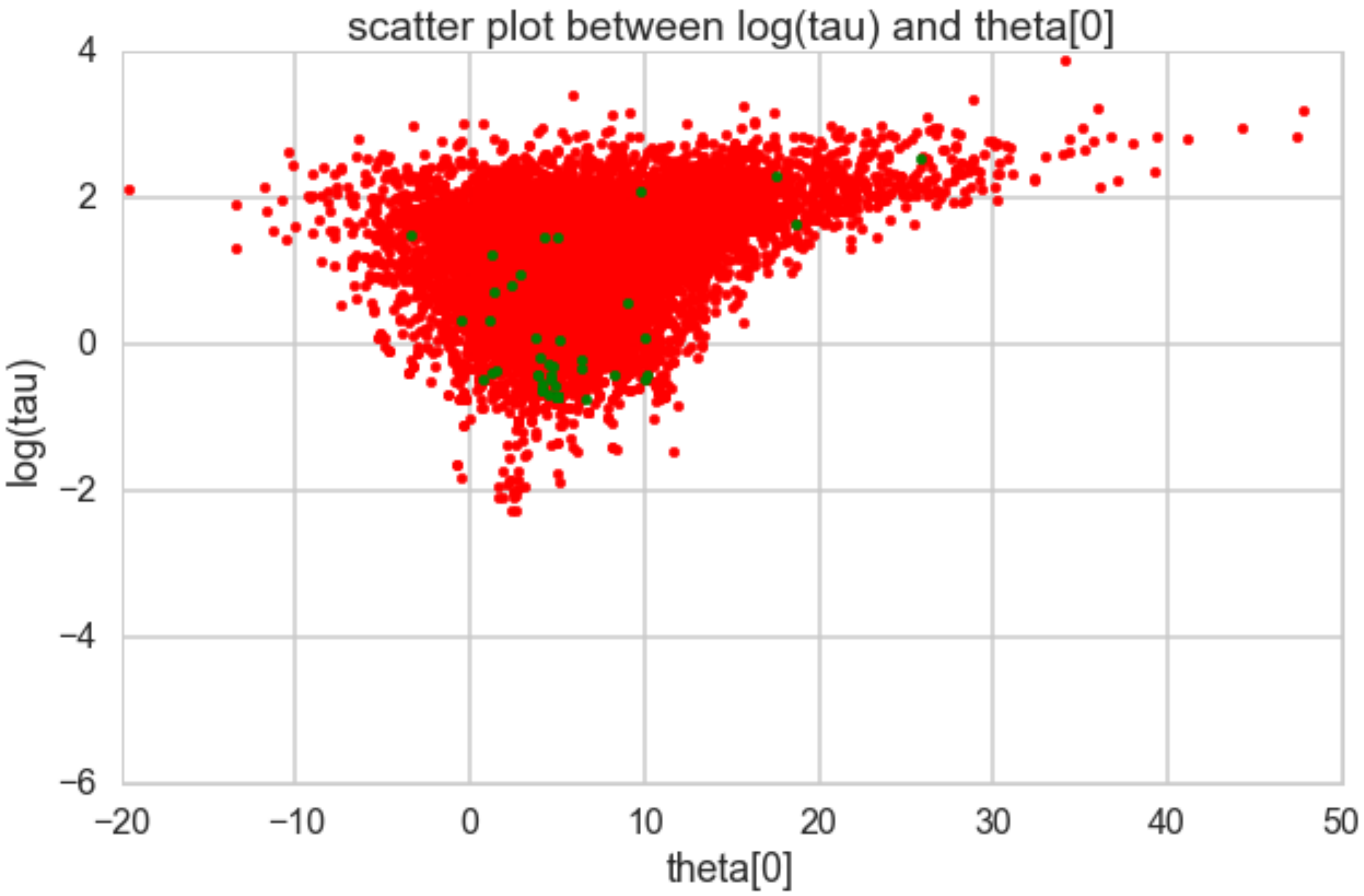
$$
\begin{gathered}
\mu \sim \mathcal{N}(0, 5) \\
\tau \sim \text{Half-Cauchy}(0, 5) \\
\theta_j \sim \mathcal{N}(\mu, \tau) \\
\bar{y_j} \sim \mathcal{N}(\theta_j, \sigma_j)
\end{gathered}
$$

problem: Small n_{eff}. Poor sampling.

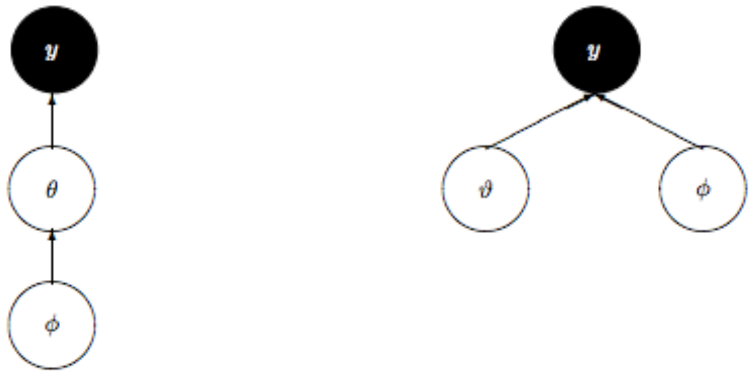


High Curvature Issues

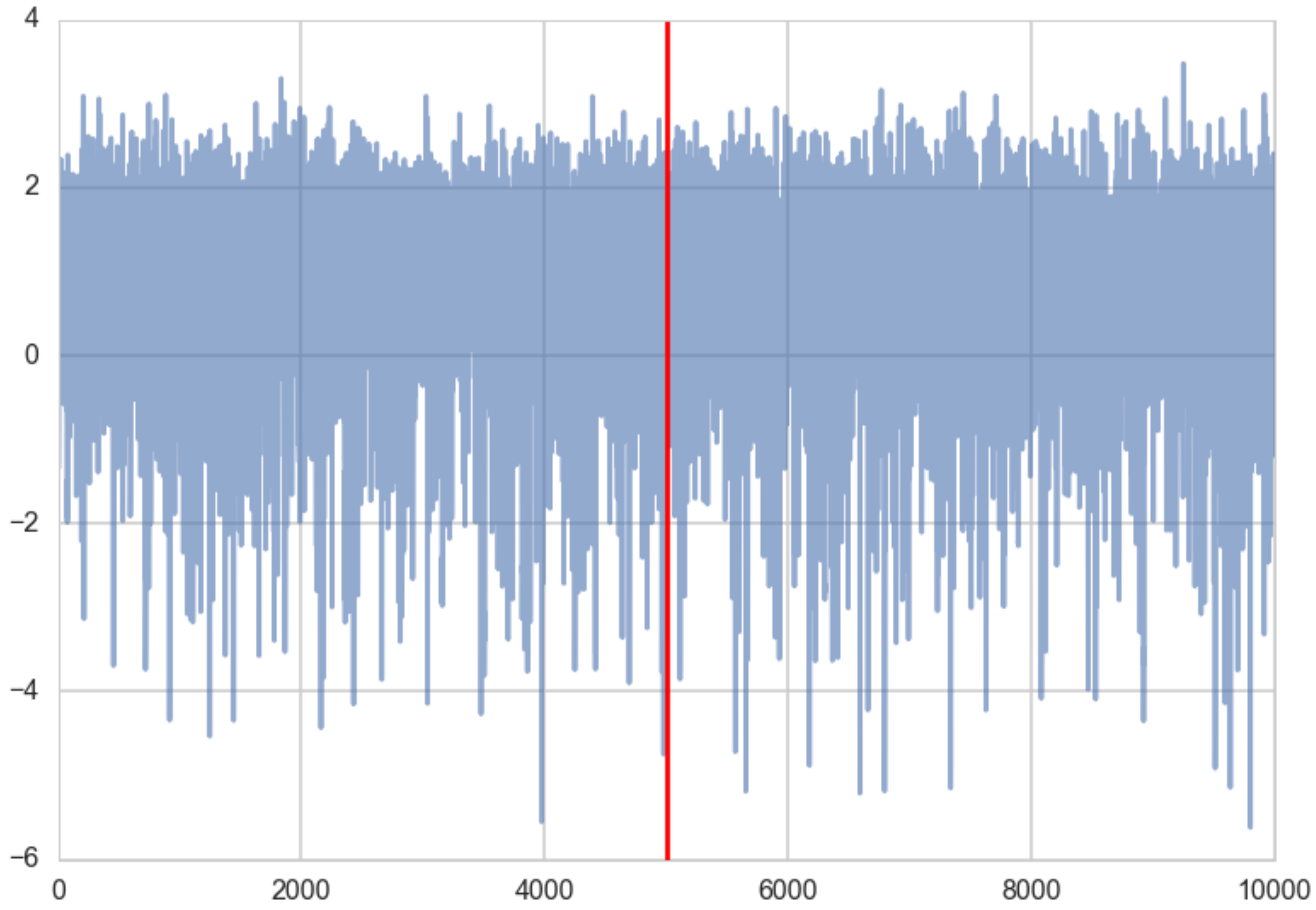
- symplectic integration diverges: good diagnostic
- sampler needs to have real small steps to not diverge, but then becomes sticky
- regions of high curvature often have high energy differences, causing trouble for microcanonical jump transitions.



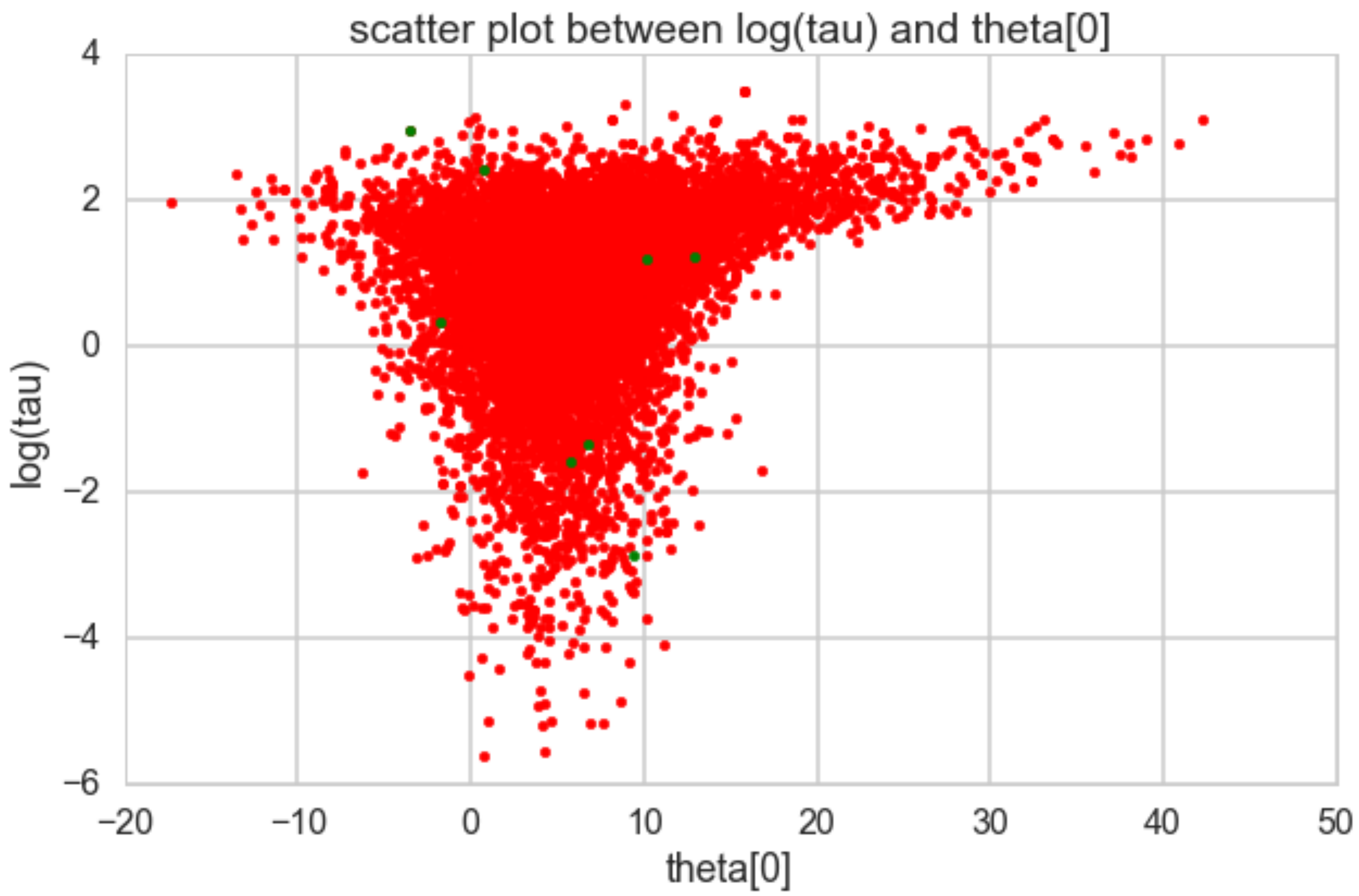
Non-centered model: Matt Trick

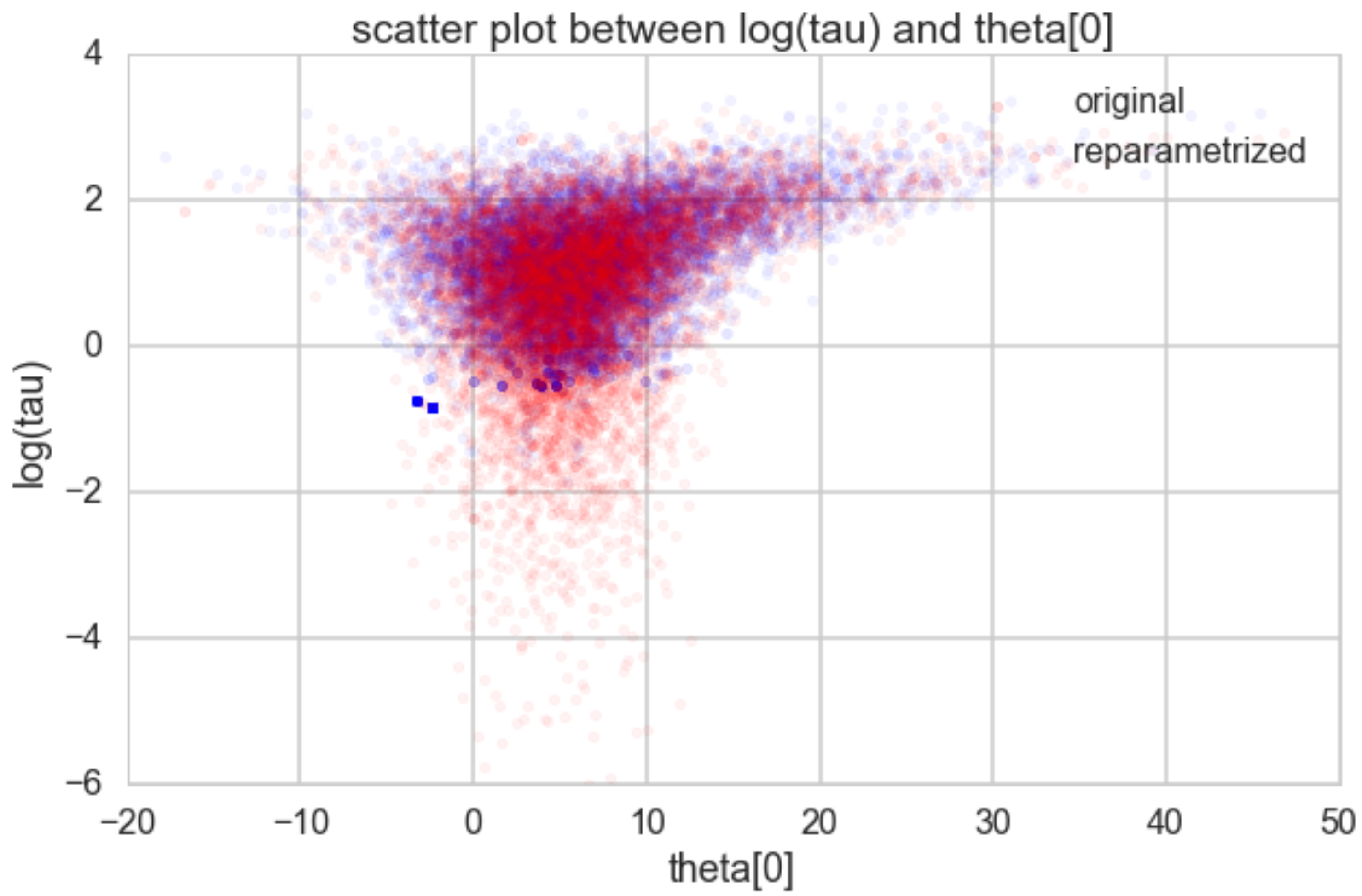


$$
\begin{gathered}
\mu \sim \mathcal{N}(0, 5) \\
\tau \sim \text{Half-Cauchy}(0, 5) \\
\nu_j \sim \mathcal{N}(0, 1) \\
\theta_j = \mu + \tau \nu_j \\
\bar{y_j} \sim \mathcal{N}(\theta_j, \sigma_j)
\end{gathered}
$$

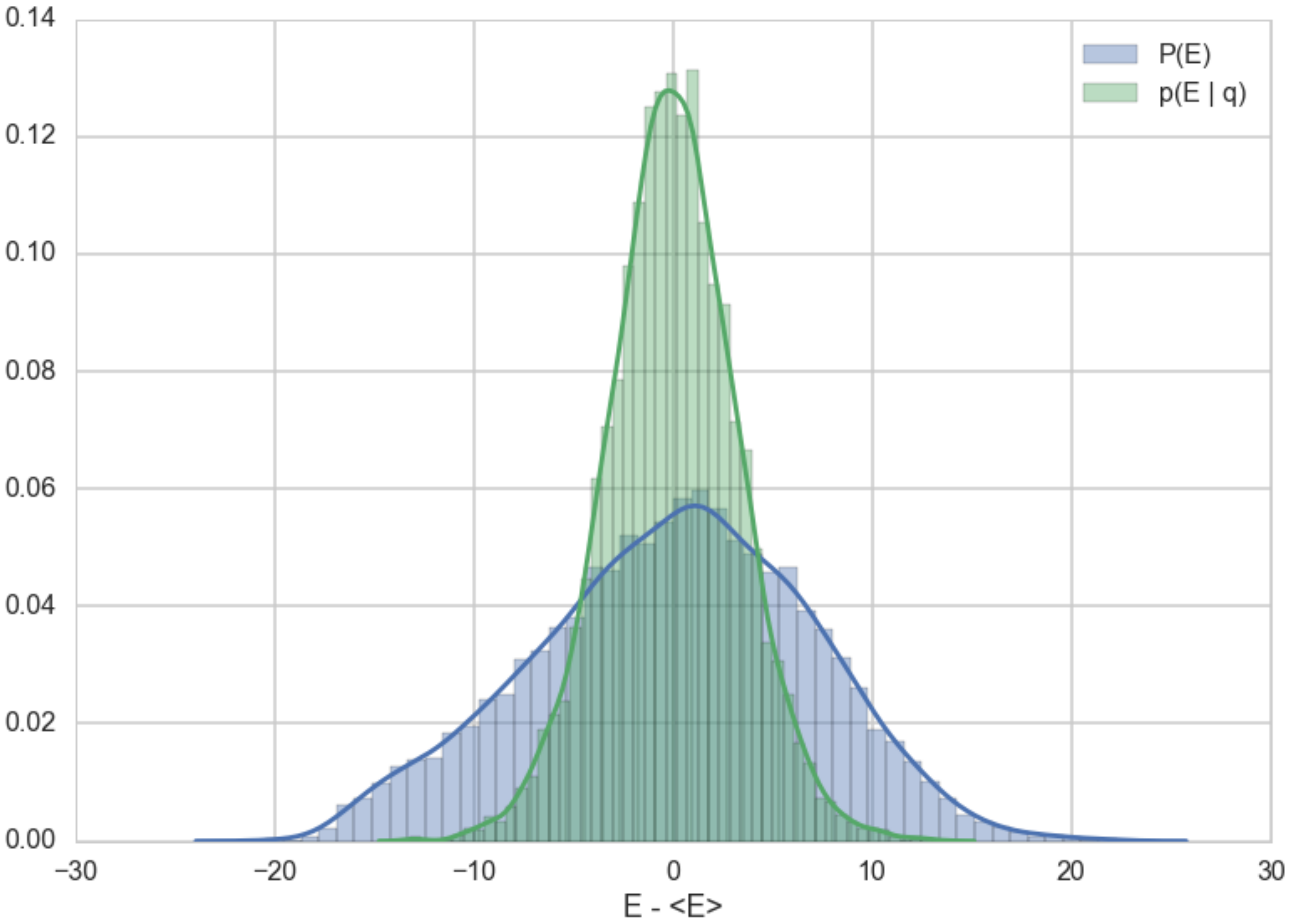


Divergences and true length of funnel





Step size effect



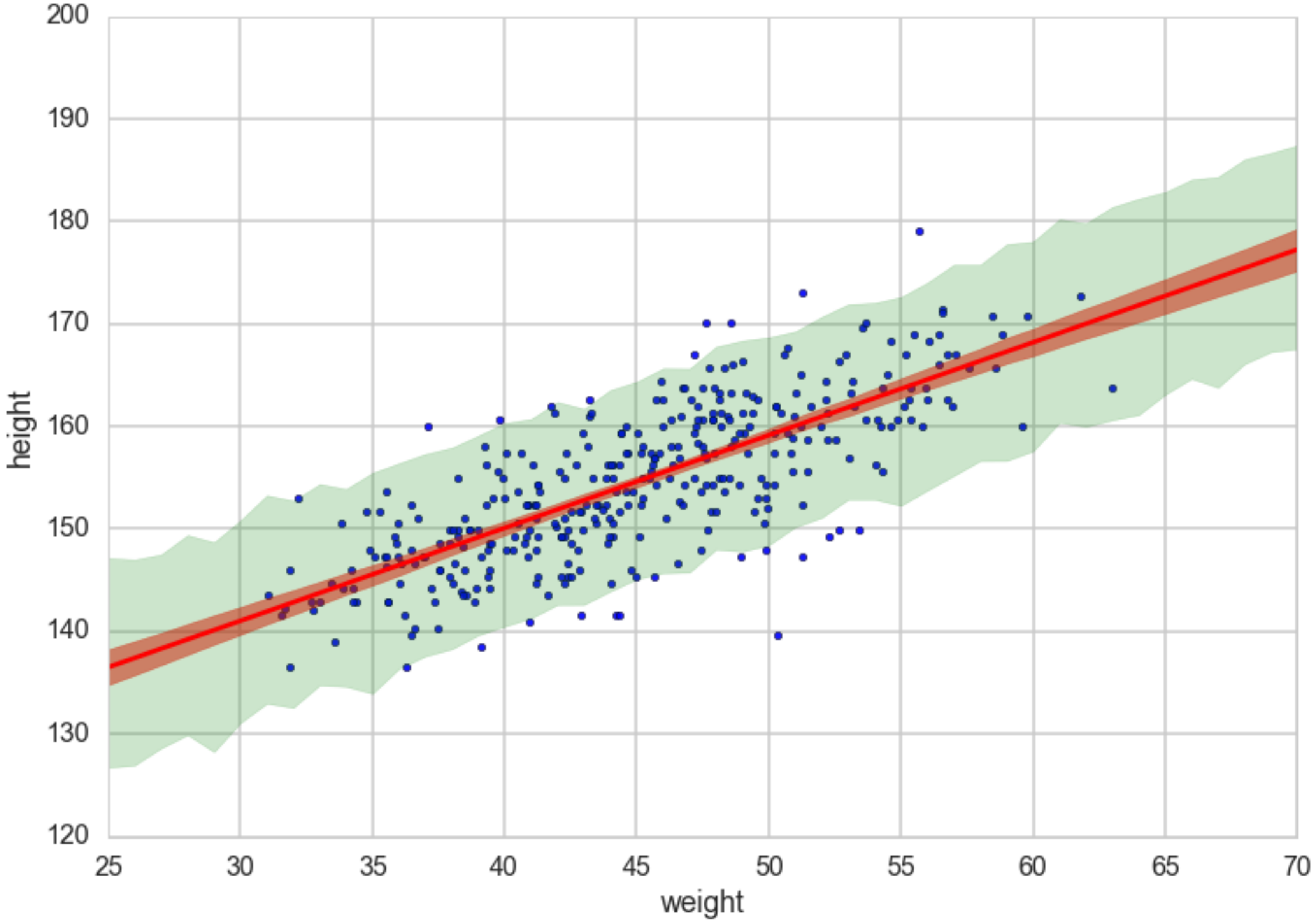
- lower step size ϵ better for symplectic integrators, especially in high curvature regions, but too small: return of the random walk

- if divergences persist on lowering step sizes, we are still too curved

- If Divergences infrequent, and all over. Mostly false positives. Lowering step sizes should make them go away

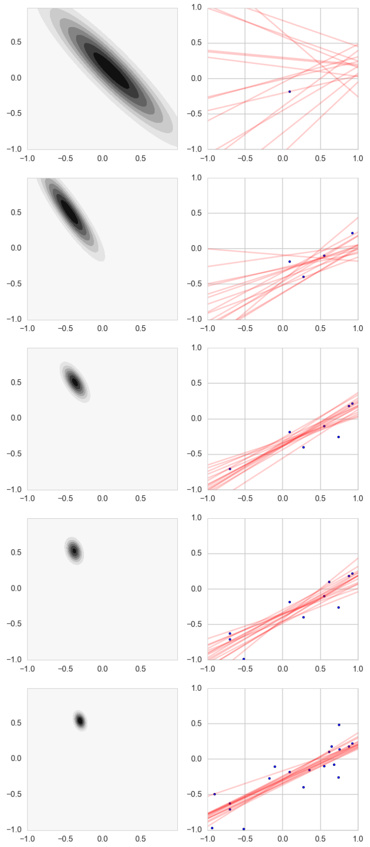
- check marginal energy: if has bigger tails, indicative of big energy changes in high-curvature regions not possible to boost to.

SUPERVISED LEARNING REGRESSION AND GLMs

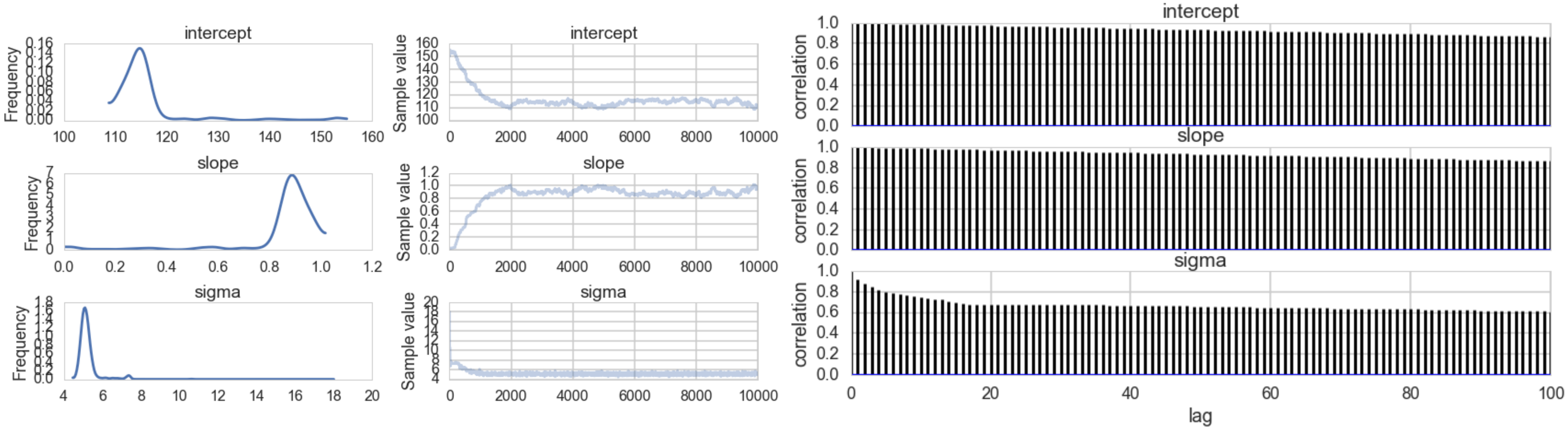


Bayesian Regression

- posterior narrower (μ spread) than PP
- supervised learning, a distrib at each x

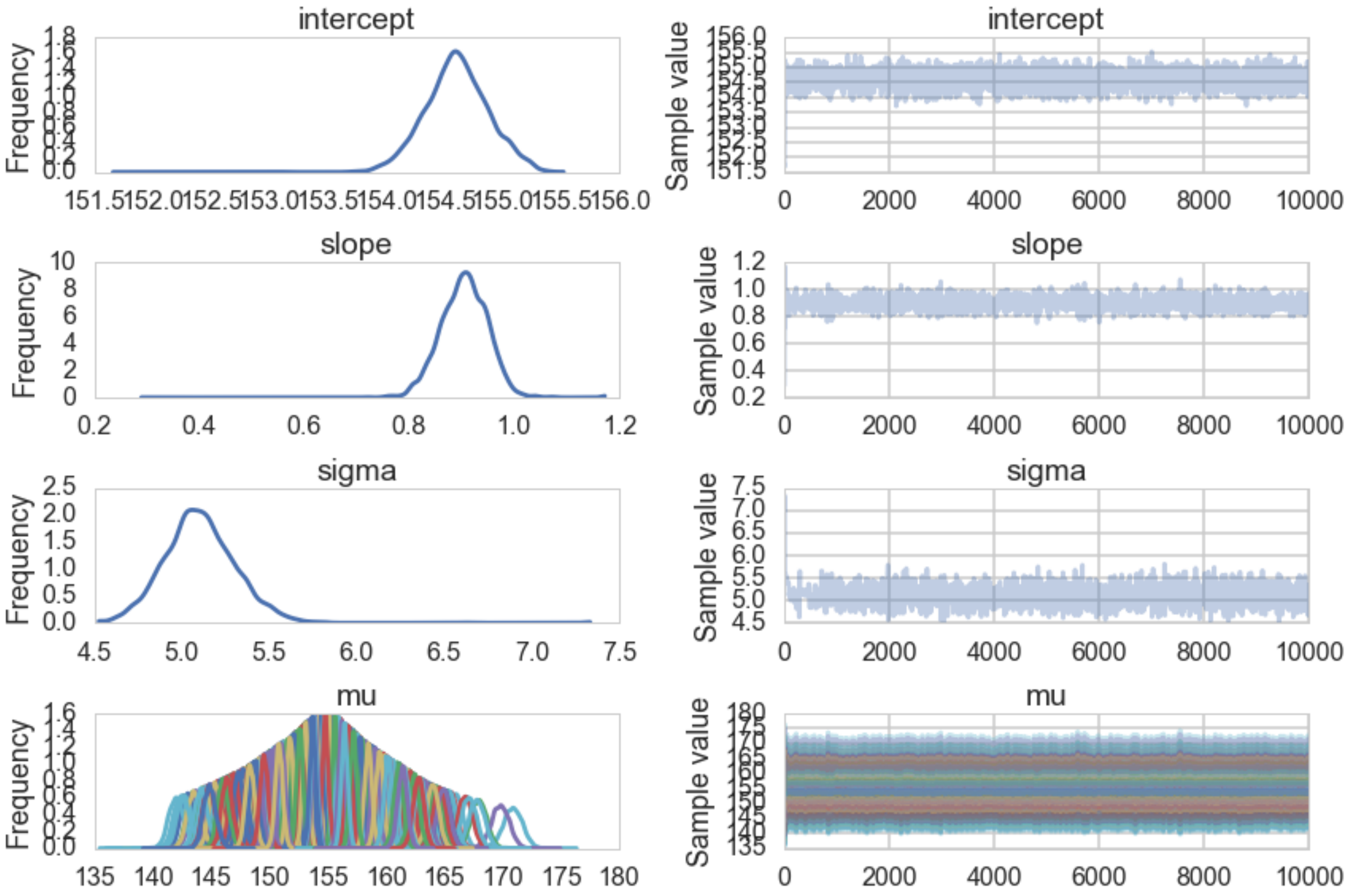


Traces for such a model are awful



The slope and intercept are very highly correlated: -0.99!

Center the weights



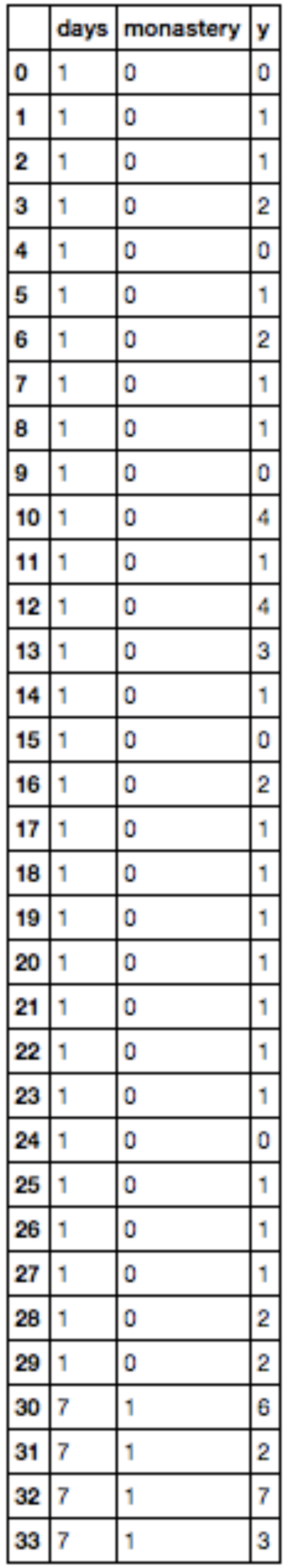
- symptom of shared information and identifiability
- fix by centering. intercept then gives response when predictor=mean.

glms

- linear regression with a link. likelihoods chosen MAXENT

$f(p_i) = \alpha + \beta x_i$ where p_i is the parameter at the ith data point.

For most GLMs, the common links we use are the *logit* link to model the space of probabilities, and the *log* link which you will use here to enforce positiveness on a parameter.



	days	monastery	y
0	1	0	0
1	1	0	1
2	1	0	1
3	1	0	2
4	1	0	0
5	1	0	1
6	1	0	2
7	1	0	1
8	1	0	1
9	1	0	0
10	1	0	4
11	1	0	1
12	1	0	4
13	1	0	3
14	1	0	1
15	1	0	0
16	1	0	2
17	1	0	1
18	1	0	1
19	1	0	1
20	1	0	1
21	1	0	1
22	1	0	1
23	1	0	1
24	1	0	0
25	1	0	1
26	1	0	1
27	1	0	1
28	1	0	2
29	1	0	2
30	7	1	6
31	7	1	2
32	7	1	7
33	7	1	3

$$y_i \sim Poisson(\lambda_i)$$

$$log(\lambda_i) = log(\frac{\mu_i}{\tau_i}) = \alpha + \beta x_i$$

λ_i is rate, μ_i is counts, τ_i is exposure.

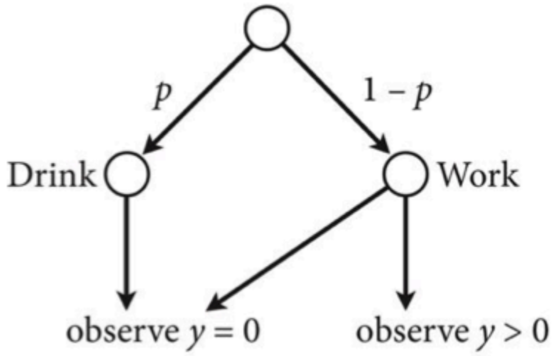
μ_i or λ_i constrained to be positive.

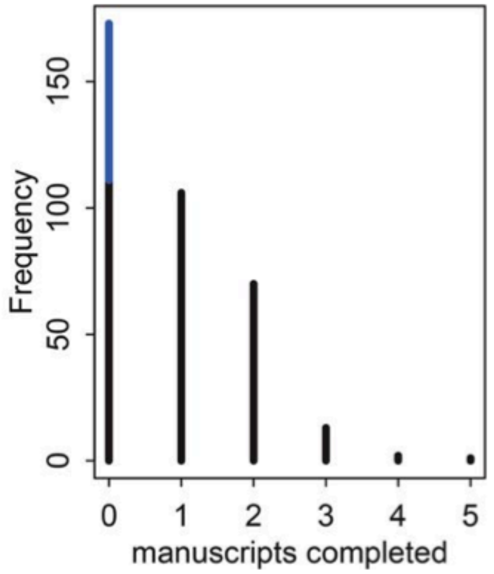
```
import theano.tensor as t
with pm.Model() as model1:
    alpha=pm.Normal("alpha", 0,100)
    beta=pm.Normal("beta", 0,1)
    logmu = t.log(df.days)+alpha+beta*df.monastery
    y = pm.Poisson("obsv", mu=t.exp(logmu), observed=df.y)
    lambda0 = pm.Deterministic("lambda0", t.exp(alpha))
    lambda1 = pm.Deterministic("lambda1", t.exp(alpha + beta))
```

Zero Inflated Poisson Mixture model

$$\mathcal{L}(y = 0) = p + (1 - p)e^{-\lambda},$$

$$\mathcal{L}(y \neq 0) = (1 - p)\frac{\lambda^y e^{-\lambda}}{y!}$$

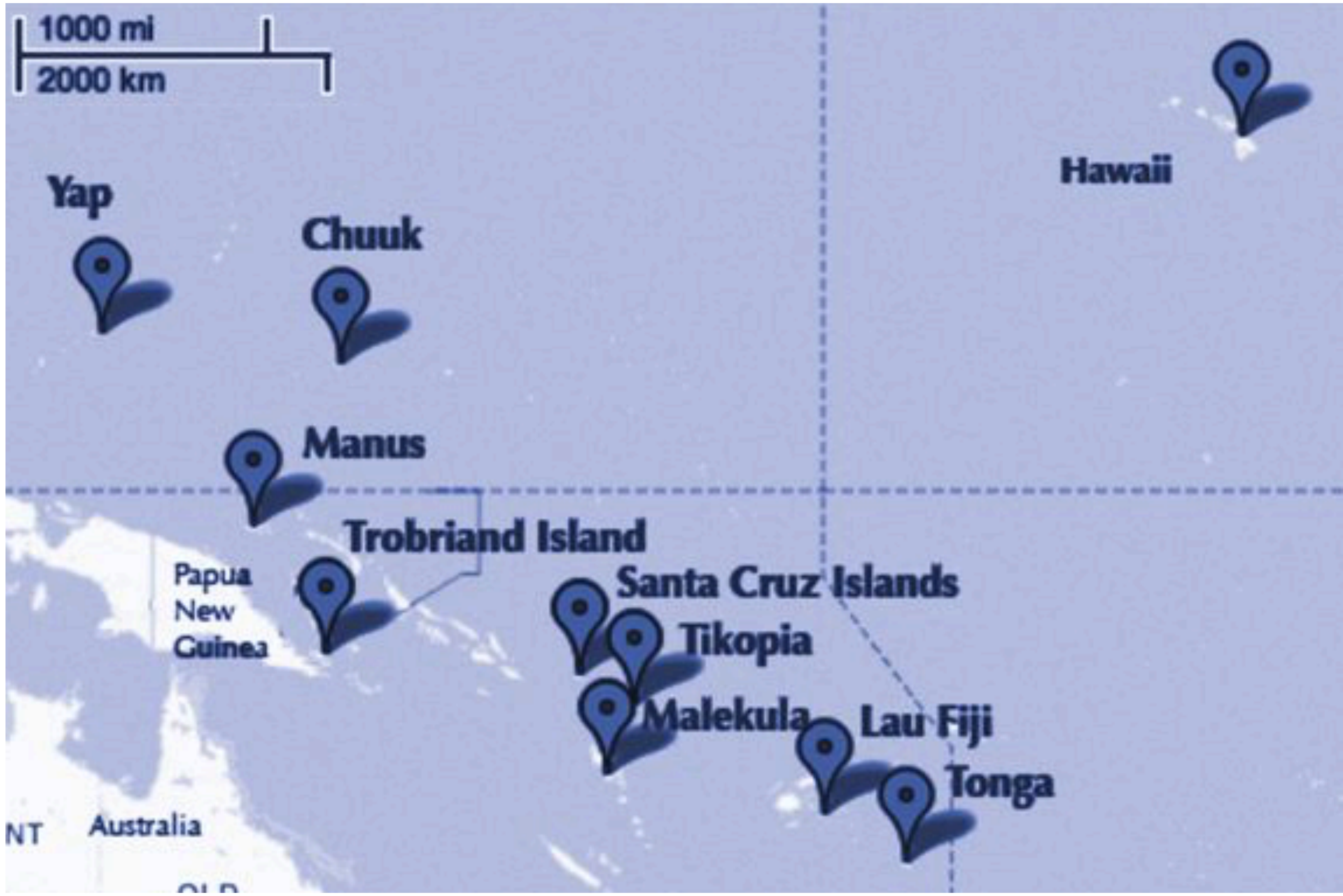




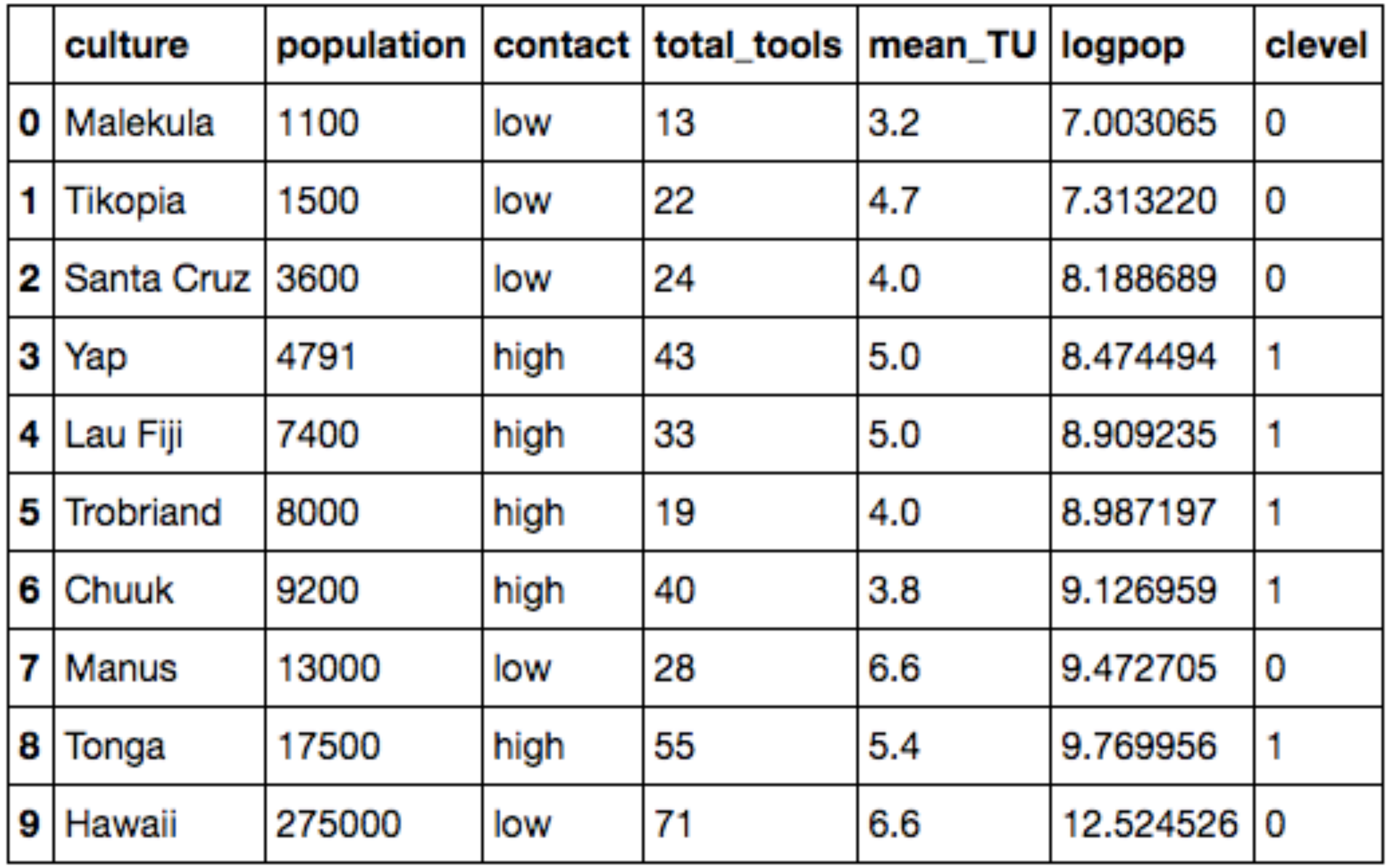
Oceanic tools

From Mcelreath:

The island societies of Oceania provide a natural experiment in technological evolution. Different historical island populations possessed tool kits of different size. These kits include fish hooks, axes, boats, hand plows, and many other types of tools. A number of theories predict that larger populations will both develop and sustain more complex tool kits. So the natural variation in population size induced by natural variation in island size in Oceania provides a natural experiment to test these ideas. It's also suggested that contact rates among populations effectively increase population size, as it's relevant to technological evolution. So variation in contact rates among Oceanic societies is also relevant. (McElreath 313)

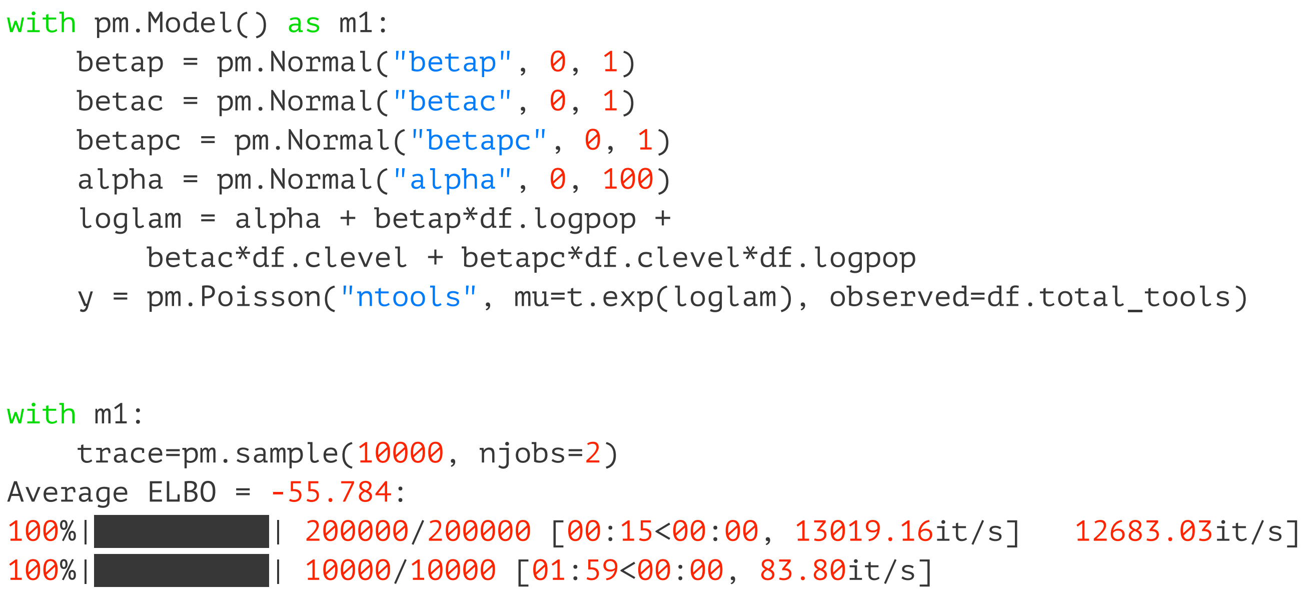


Model M1



	culture	population	contact	total_tools	mean_TU	logpop	clevel
0	Malekula	1100	low	13	3.2	7.003065	0
1	Tikopia	1500	low	22	4.7	7.313220	0
2	Santa Cruz	3600	low	24	4.0	8.188689	0
3	Yap	4791	high	43	5.0	8.474494	1
4	Lau Fiji	7400	high	33	5.0	8.909235	1
5	Trobriand	8000	high	19	4.0	8.987197	1
6	Chuuk	9200	high	40	3.8	9.126959	1
7	Manus	13000	low	28	6.6	9.472705	0
8	Tonga	17500	high	55	5.4	9.769956	1
9	Hawaii	275000	low	71	6.6	12.524526	0

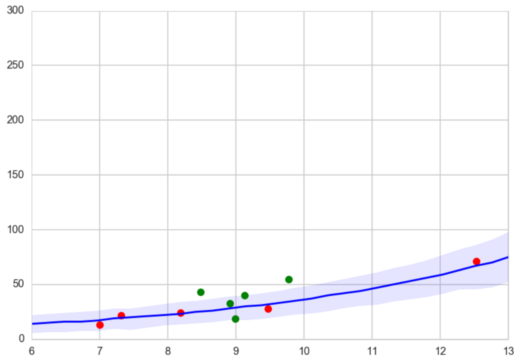
$$
\begin{aligned}
T_i &\sim Poisson(\lambda_i)\\
log(\lambda_i) &= \alpha + \beta_P log(P_i) + \beta_C C_i + \beta_{PC} C_i log(P_i)\\
\alpha &\sim N(0,100)\\
\beta_P &\sim N(0,1)\\
\beta_C &\sim N(0,1)\\
\beta_{PC} &\sim N(0,1)
\end{aligned}
$$



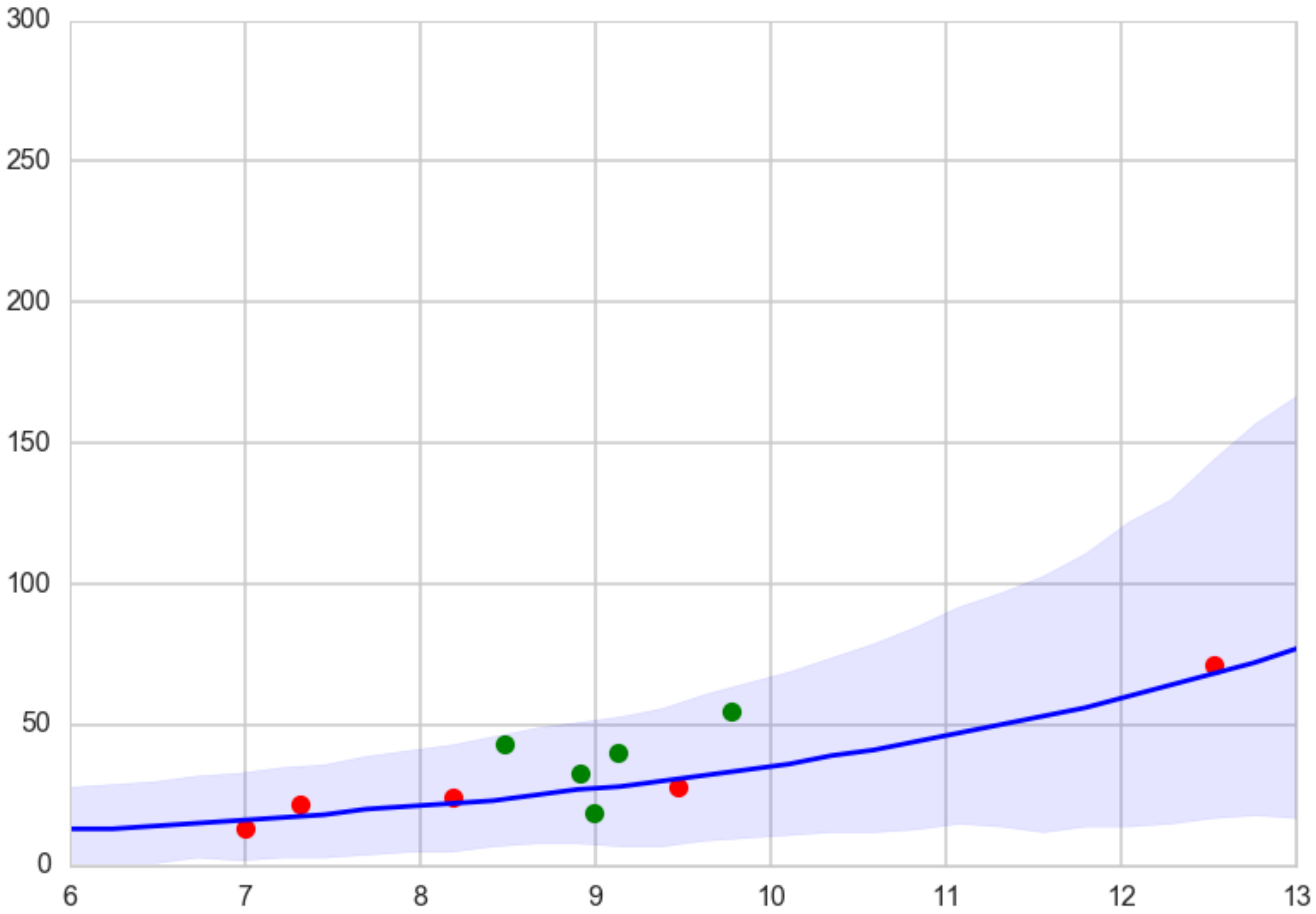
```
with pm.Model() as m1:
    betap = pm.Normal("betap", 0, 1)
    betac = pm.Normal("betac", 0, 1)
    betapc = pm.Normal("betapc", 0, 1)
    alpha = pm.Normal("alpha", 0, 100)
    loglam = alpha + betap*df.logpop +
        betac*df.clevel + betapc*df.clevel*df.logpop
    y = pm.Poisson("ntools", mu=t.exp(loglam), observed=df.total_tools)
```

```
with m1:
    trace=pm.sample(10000, njobs=2)
Average ELBO = -55.784:
100%|██████████| 200000/200000 [00:15<00:00, 13019.16it/s]   12683.03it/s]
100%|██████████| 10000/10000 [01:59<00:00, 83.80it/s]
```

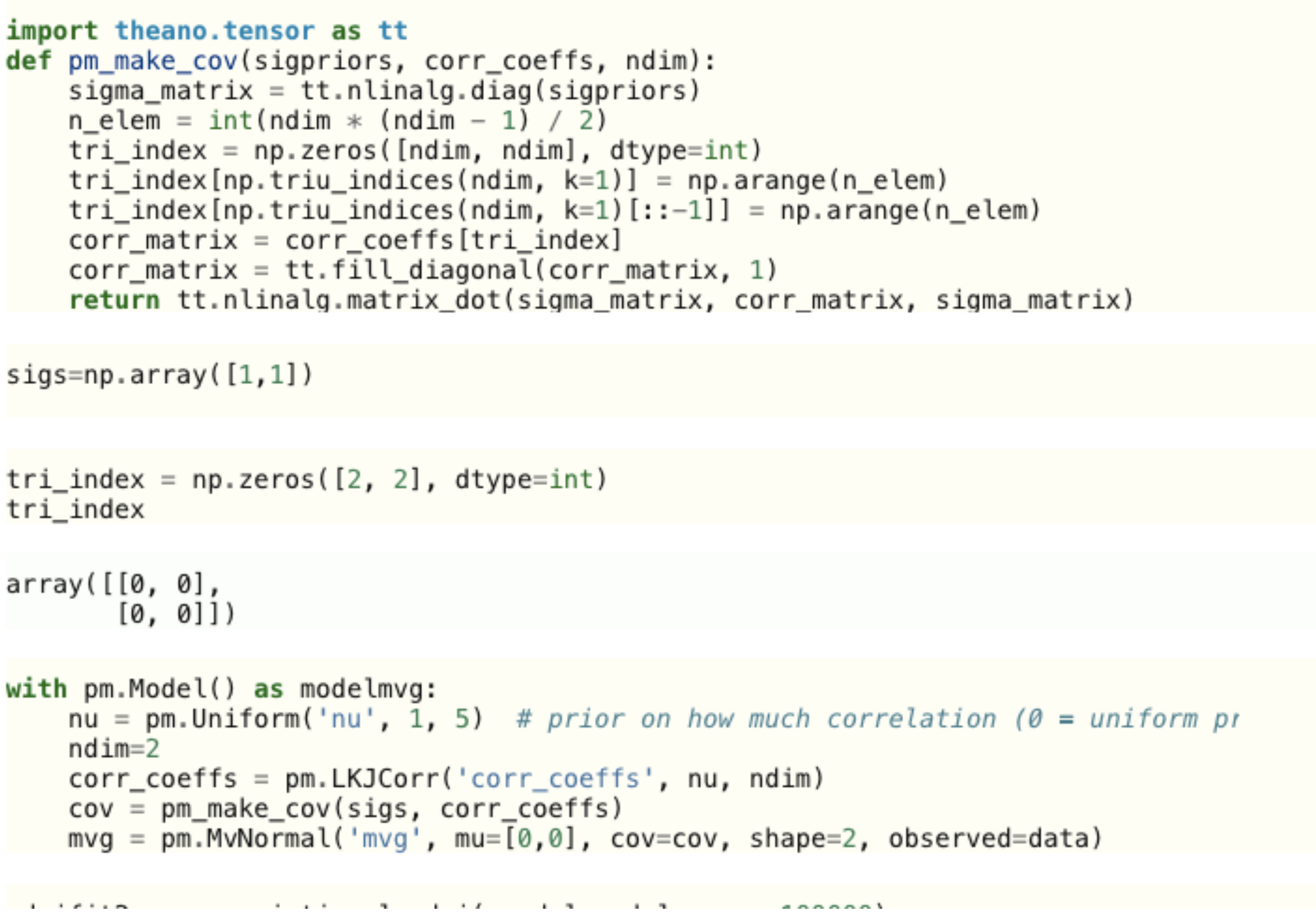
Hierarchical regression: Overdispersion fixed by varying intercepts



```
with pm.Model() as m3c:
    betap = pm.Normal("betap", 0, 1)
    alpha = pm.Normal("alpha", 0, 100)
    sigmasoc = pm.HalfCauchy("sigmasoc", 1)
    alphasoc = pm.Normal("alphasoc", 0, sigmasoc, shape=df.shape[0])
    loglam = alpha + alphasoc + betap*df.logpop_c
    y = pm.Poisson("ntools", mu=t.exp(loglam), observed=df.total_tools)
```



Intercept-slope Correlation Modeling



```
import theano.tensor as tt
def pm_make_cov(sigpriors, corr_coeffs, ndim):
    sigma_matrix = tt.nlinalg.diag(sigpriors)
    n_elem = int(ndim * (ndim - 1) / 2)
    tri_index = np.zeros([ndim, ndim], dtype=int)
    tri_index[np.triu_indices(ndim, k=1)] = np.arange(n_elem)
    tri_index[np.triu_indices(ndim, k=1)[::-1]] = np.arange(n_elem)
    corr_matrix = corr_coeffs[tri_index]
    corr_matrix = tt.fill_diagonal(corr_matrix, 1)
    return tt.nlinalg.matrix_dot(sigma_matrix, corr_matrix, sigma_matrix)
```

```
sigs=np.array([1,1])
```

```
tri_index = np.zeros([2, 2], dtype=int)
tri_index
```

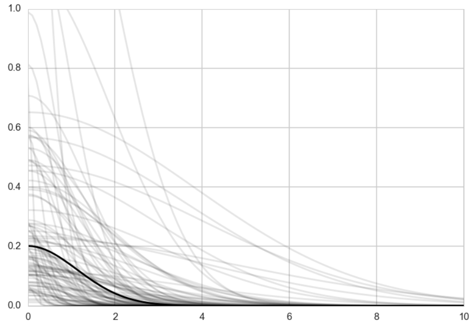
```
array([[0, 0],
       [0, 0]])
```

```
with pm.Model() as modelmvg:
    nu = pm.Uniform('nu', 1, 5)  # prior on how much correlation (0 = uniform pr
    ndim=2
    corr_coeffs = pm.LKJCorr('corr_coeffs', nu, ndim)
    cov = pm_make_cov(sigs, corr_coeffs)
    mvg = pm.MvNormal('mvg', mu=[0,0], cov=cov, shape=2, observed=data)
```

Oceanic Tools Correlations: Example of GP

We modeled society specific intercepts for oceanic tools as draws from a 0 mean multivariate gaussian and correlation function depending on distance: nearer societies have similar intercepts.

Covariance posteriors:



$$
\begin{aligned}
T_i &\sim \text{Poisson}(\lambda_i)\\
\log \lambda_i &= \alpha + \gamma_{\text{SOCIETY}[i]} + \beta_P \log P_i\\
\gamma &\sim \text{MVNormal}((0,\ldots,0),\mathbf{K})\\
\mathbf{K}_{ij} &= \eta^2 \exp(-\rho^2 D_{ij}^2) + \delta_{ij}(0.01)\\
\alpha &\sim \text{Normal}(0,10)\\
\beta_P &\sim \text{Normal}(0,1)\\
\eta^2 &\sim \text{HalfCauchy}(0,1)\\
\rho^2 &\sim \text{HalfCauchy}(0,1)
\end{aligned}
$$

Gaussian Formulae

$$JOINT: \;\; p(y, f^*) = \mathcal{N}\left(\begin{bmatrix}\mu_y \\ \mu_{f^*}\end{bmatrix}, \begin{bmatrix}\Sigma_{yy} & \Sigma_{yf^*} \\ \Sigma_{yf^*}^T & \Sigma_{f^*f^*}\end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix}\mu \\ \mu_*\end{bmatrix}, \begin{bmatrix}K+\sigma^2 I & K_* \\ K_*^T & K_{**}\end{bmatrix}\right)$$

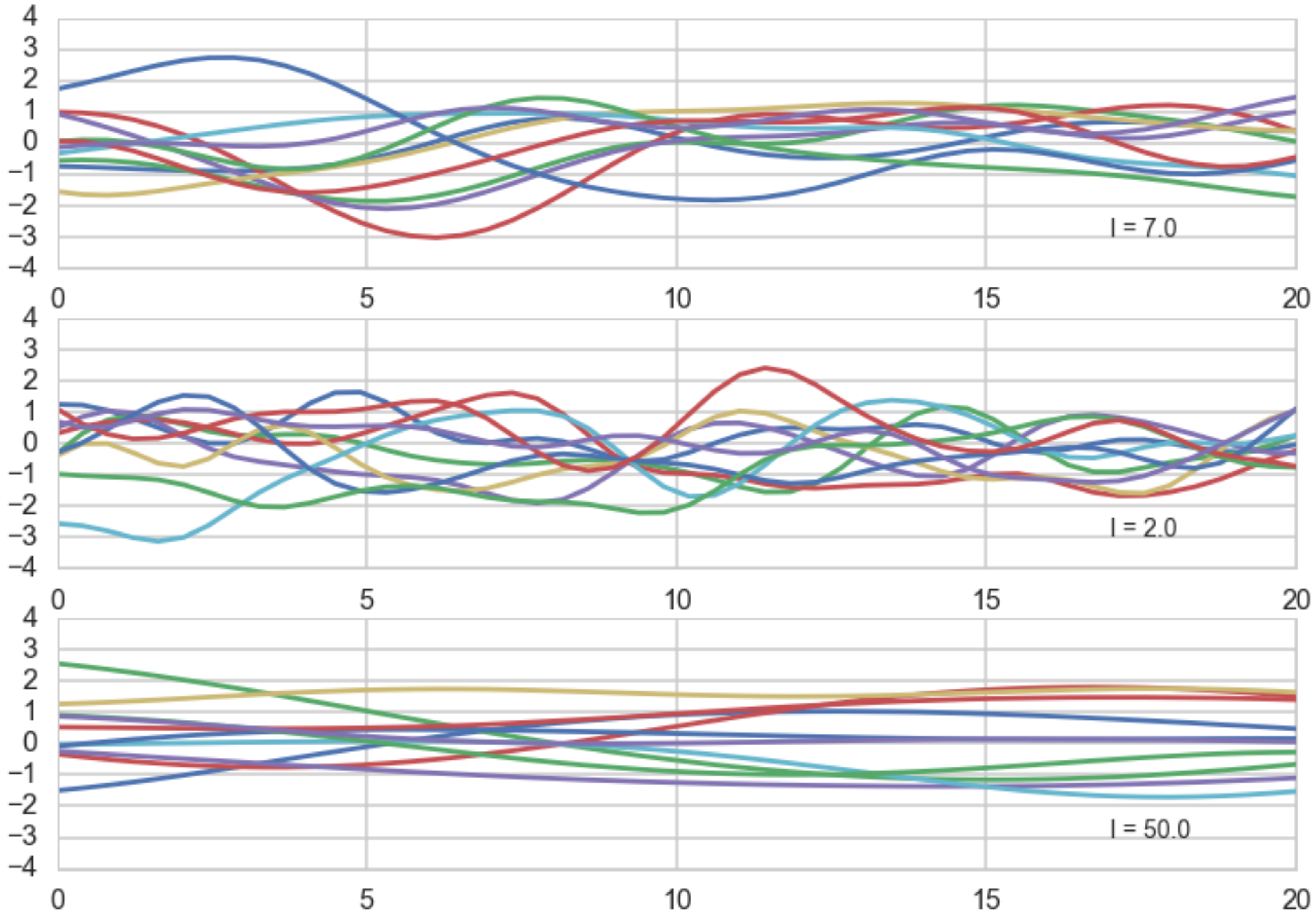
$$MARGINAL: \;\; p(f^*) = \int p(f^*, y)dy = \mathcal{N}(\mu_*, K_{**})$$

$$CONDITIONAL: \;\; p(f^* \mid y) = \mathcal{N}\left(\mu_* + K_*(K+\sigma^2 I)^{-1}(y-\mu), \;\; K_{**} - K_*(K+\sigma^2 I)^{-1} K_*^T\right)$$

Note here that:

$$K = K(x, x); K_* = K(x, x^*); K_{**} = K(x^*, x^*)$$

Generating curves from a kernel-based covariance



a Gaussian Process defines a prior distribution over functions!

Once we have seen some data, this prior can be converted to a posterior over functions, thus restricting the set of functions that we can use based on the data.

KEY INSIGHT:

MARGINAL IS DECOUPLED

...for the marginal of a gaussian, only the covariance of the block of the matrix involving the unmarginalized dimensions matters! Thus "if you ask only for the properties of the function (you are fitting to the data) at a finite number of points, then inference in the Gaussian process will give you the same answer if you ignore the infinitely many other points, as if you would have taken them all into account!"

-Rasmunnsen

Universal Approximation

- Exact correspondence between the gaussian process (direct usage of gaussians in space) to the basis function regression (in feature space with gaussians for prior parameters) in the kernelized representation, as long as we identify the GP covariance function k with the kernel function $\kappa(x, x') = \phi(x)^T \Sigma \phi(x')$. (Mercer's theorem)

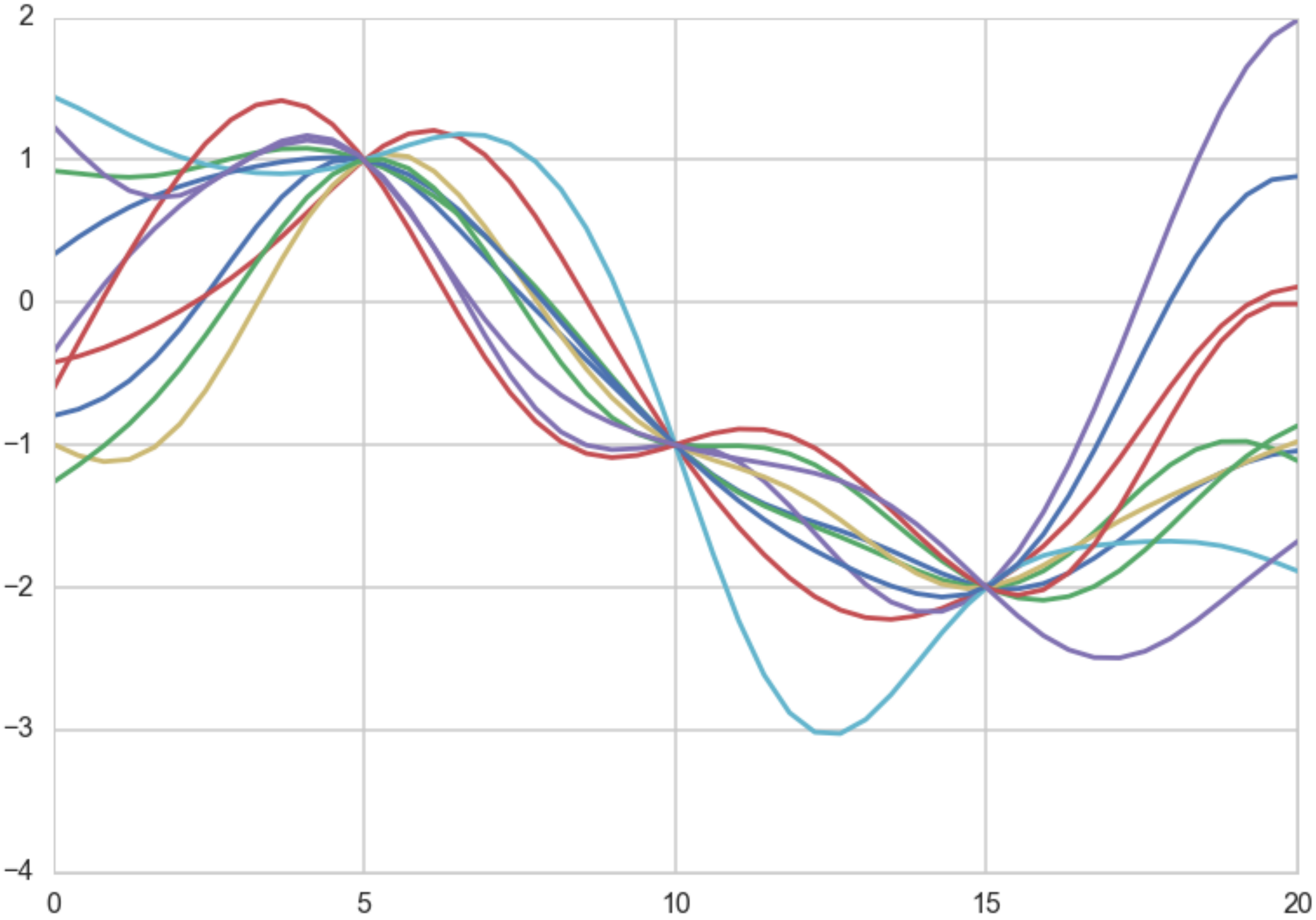
- We have seen such universal approximation in NN

- there is a connection for both single layer and deep NN

Conditional

$$p(f^* \mid y) = \mathcal{N}\left(\mu_* + K_*(K+\sigma^2 I)^{-1}(y-\mu),\ K_{**} - K_*(K+\sigma^2 I)^{-1}K_*^T\right)$$

EQUALS Predictive


2
1
0
−1
−2
−3
−4
0
5
10
15
20

INFERENCE

Use the marginal likelihood:

$$p(y|X) = \int_f p(y|f, X)p(f|X)df$$

The Marginal likelihood given a GP prior and a gaussian likelihood is:

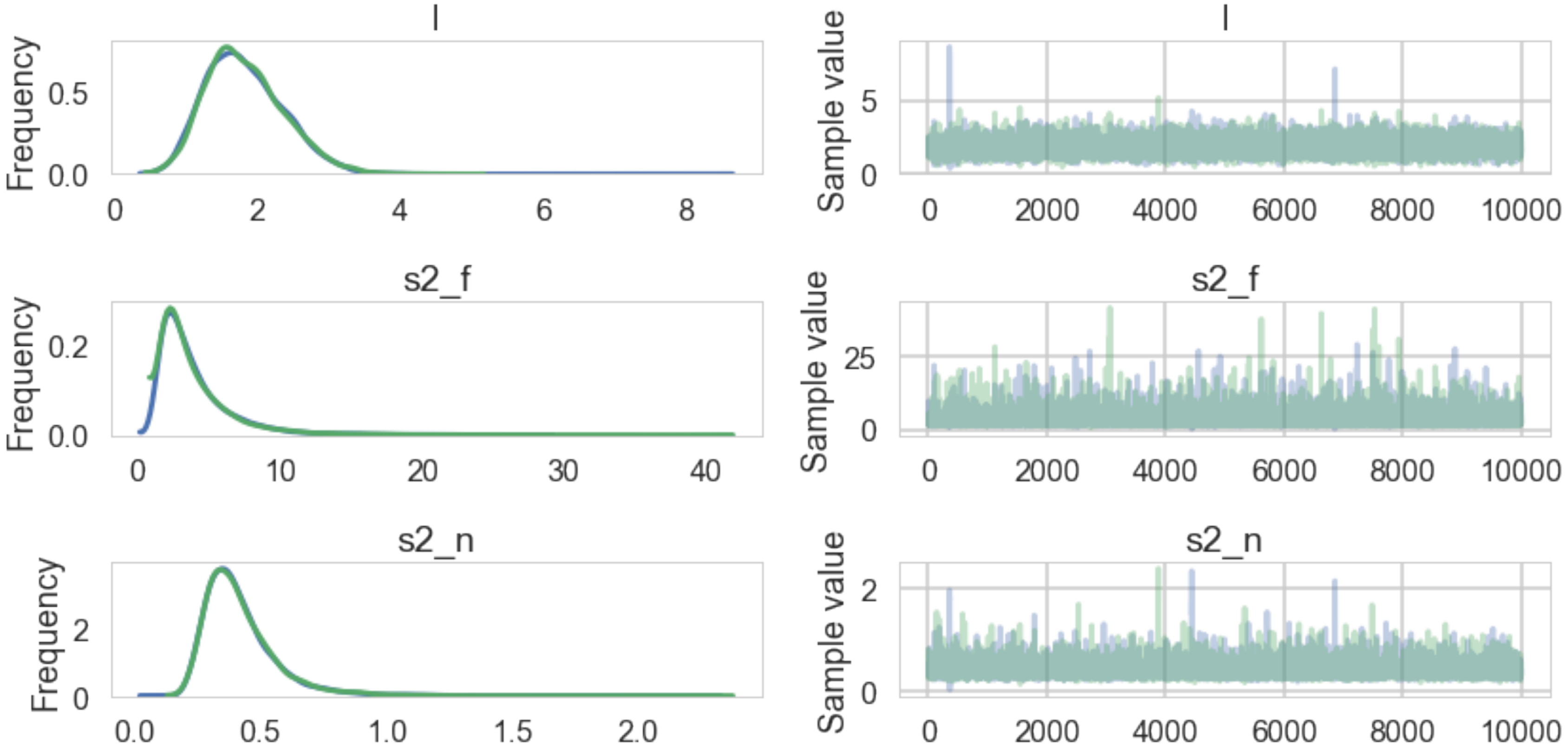
$$\log p(y|X) = -\frac{n}{2}\log 2\pi - \frac{1}{2}\log |K + \sigma^2 I| - \frac{1}{2}y^T (K + \sigma^2 I)^{-1} y$$

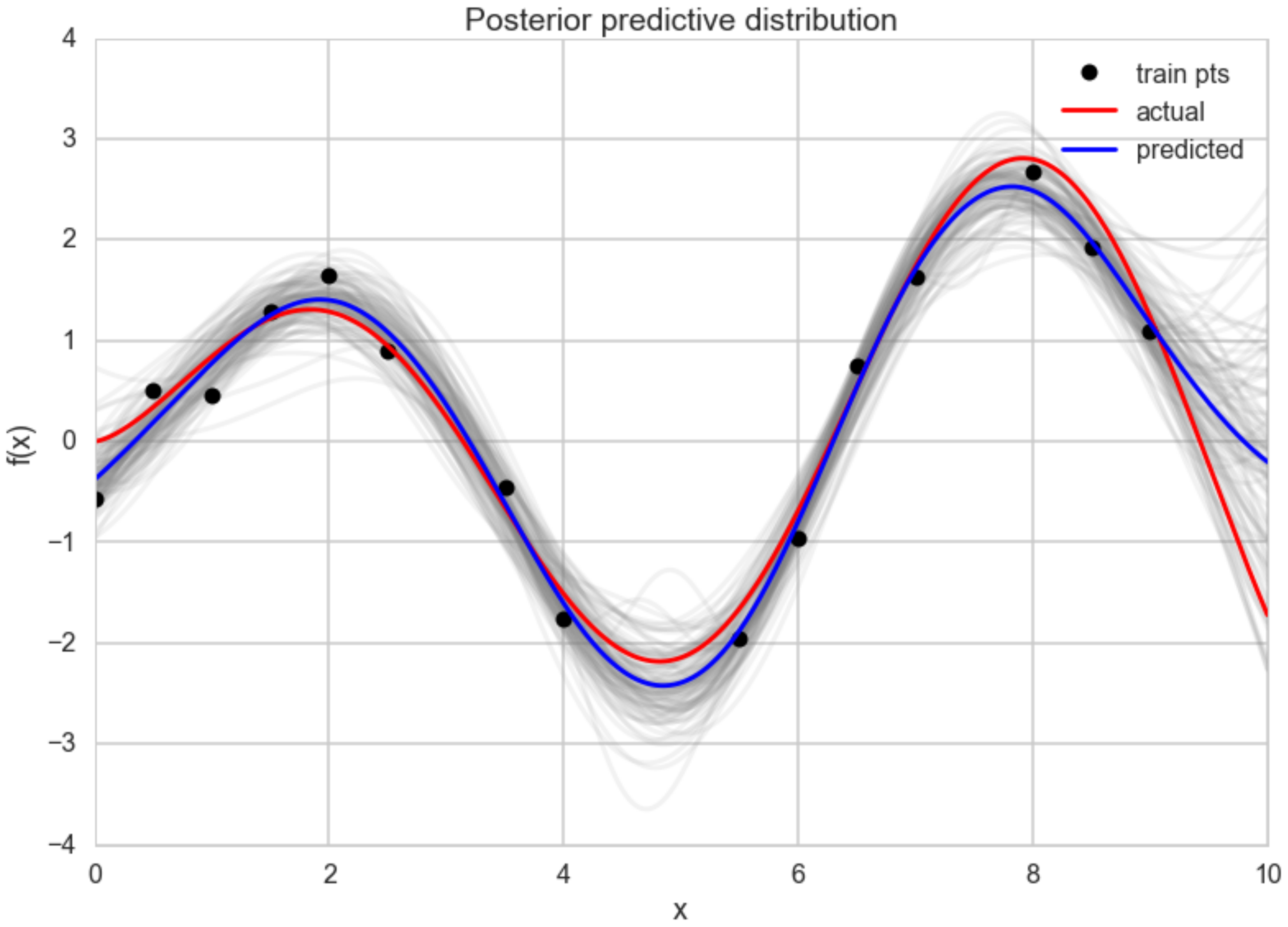
Setting up the model

```
with pm.Model() as model:
    # priors on the covariance function hyperparameters
    l = pm.Uniform('l', 0, 10)
    # uninformative prior on the function variance
    s2_f = pm.HalfCauchy('s2_f', beta=10)
    # uninformative prior on the noise variance
    s2_n = pm.HalfCauchy('s2_n', beta=5)
    # covariance functions for the function f and the noise
    f_cov = s2_f**2 * pm.gp.cov.ExpQuad(1, l)
    mgp = pm.gp.Marginal(cov_func=f_cov)
    y_obs = mgp.marginal_likelihood('y_obs',
        X=xtrain.reshape(-1,1), y=ytrain, noise=s2_n,
        is_observed=True)
```

MCMC

```
with model:
     trace = pm.sample(10000, tune=2000,
        nuts_kwargs={'target_accept':0.85})
 with model:
     fpred = mgp.conditional("fpred",
        Xnew = x_pred.reshape(-1,1),
        pred_noise=False)
     ypred = mgp.conditional("ypred",
        Xnew = x_pred.reshape(-1,1),
        pred_noise=True)
     gp_samples = pm.sample_ppc(trace,
         vars=[fpred, ypred],
         samples=200)
```




Posterior predictive distribution
4
3
2
1
0
−1
−2
−3
−4
f(x)
0
2
4
6
8
10
x
train pts
actual
predicted

MODEL CHECKING

Multiple replications of the posterior predictive

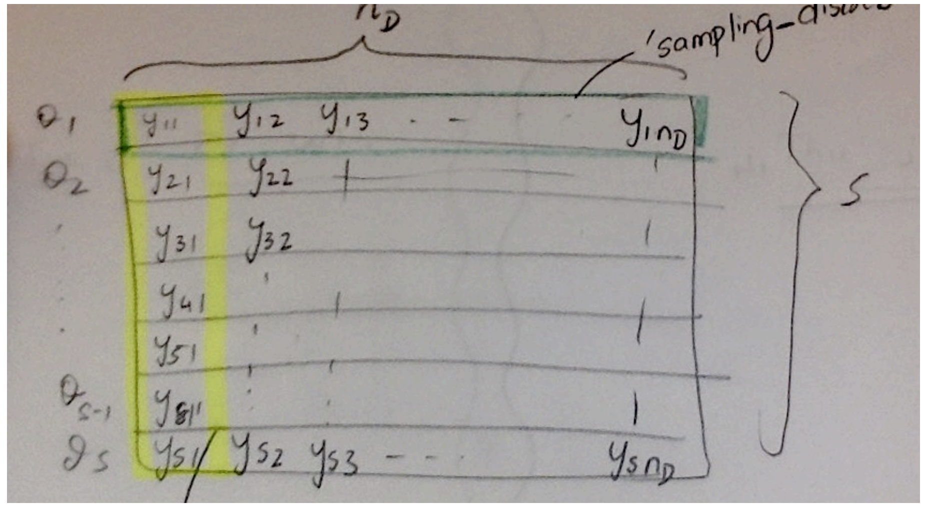
$$p(\{y^*\}) = \int p(\{y^*\}|\theta)p(\theta|\mathcal{D})d\theta, \text{ observed data: } \mathcal{D} = \{y\}$$

Replicated Data: $\{y_r\}$: data seen tomorrow if experiment replicated with same model and value of θ producing todays data $\{y\}$.

$\{y_r\}$ comes from posterior predictive, and if there are covariates $\{x^*\}$, then $\{y_r\}$ is calculated at those covariates only (`sample_ppc`).

Departure from usual predictive sampling

Sample an entire $\{y_r\}$ at each θ from trace.



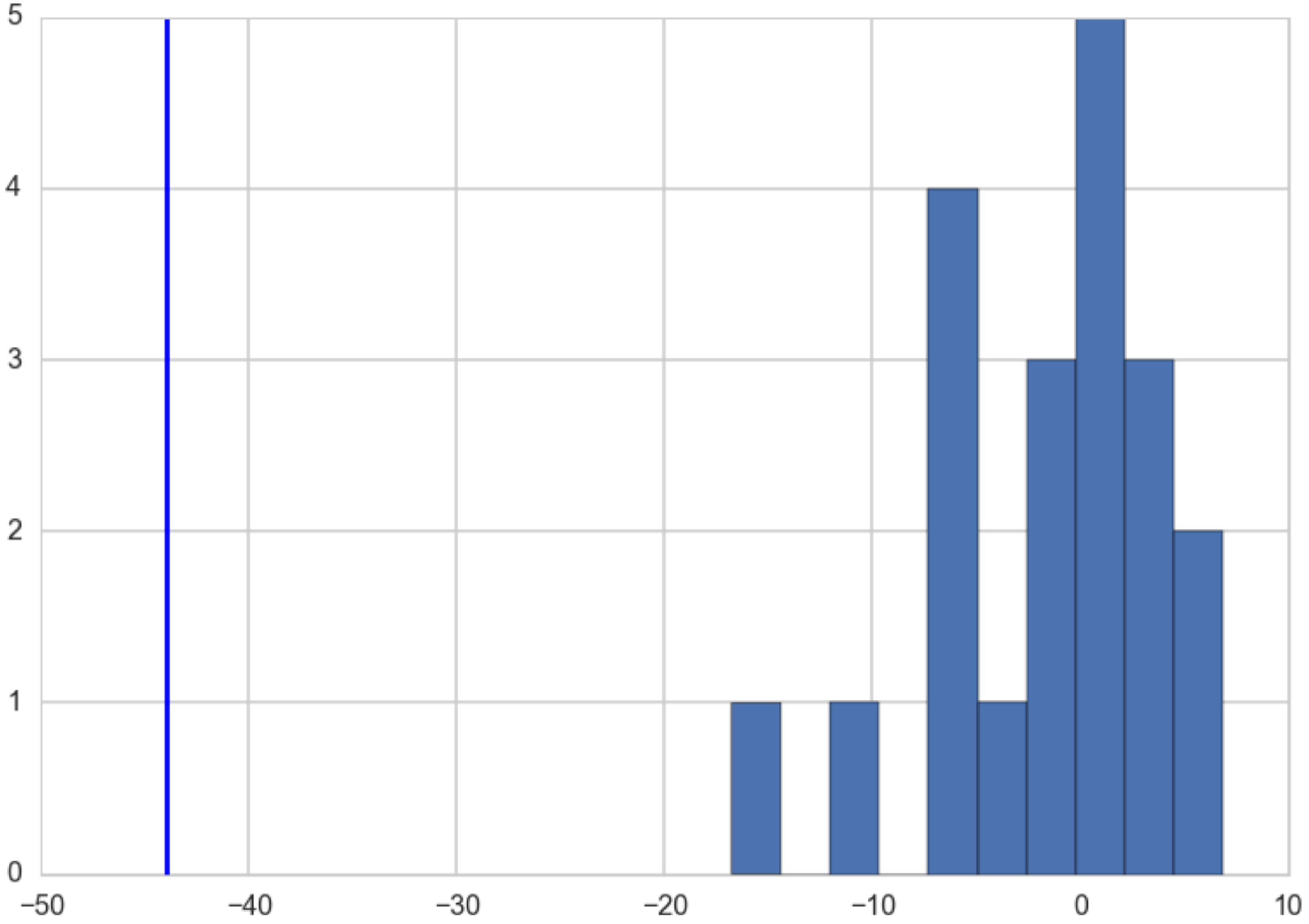
n_D

'sampling_distr

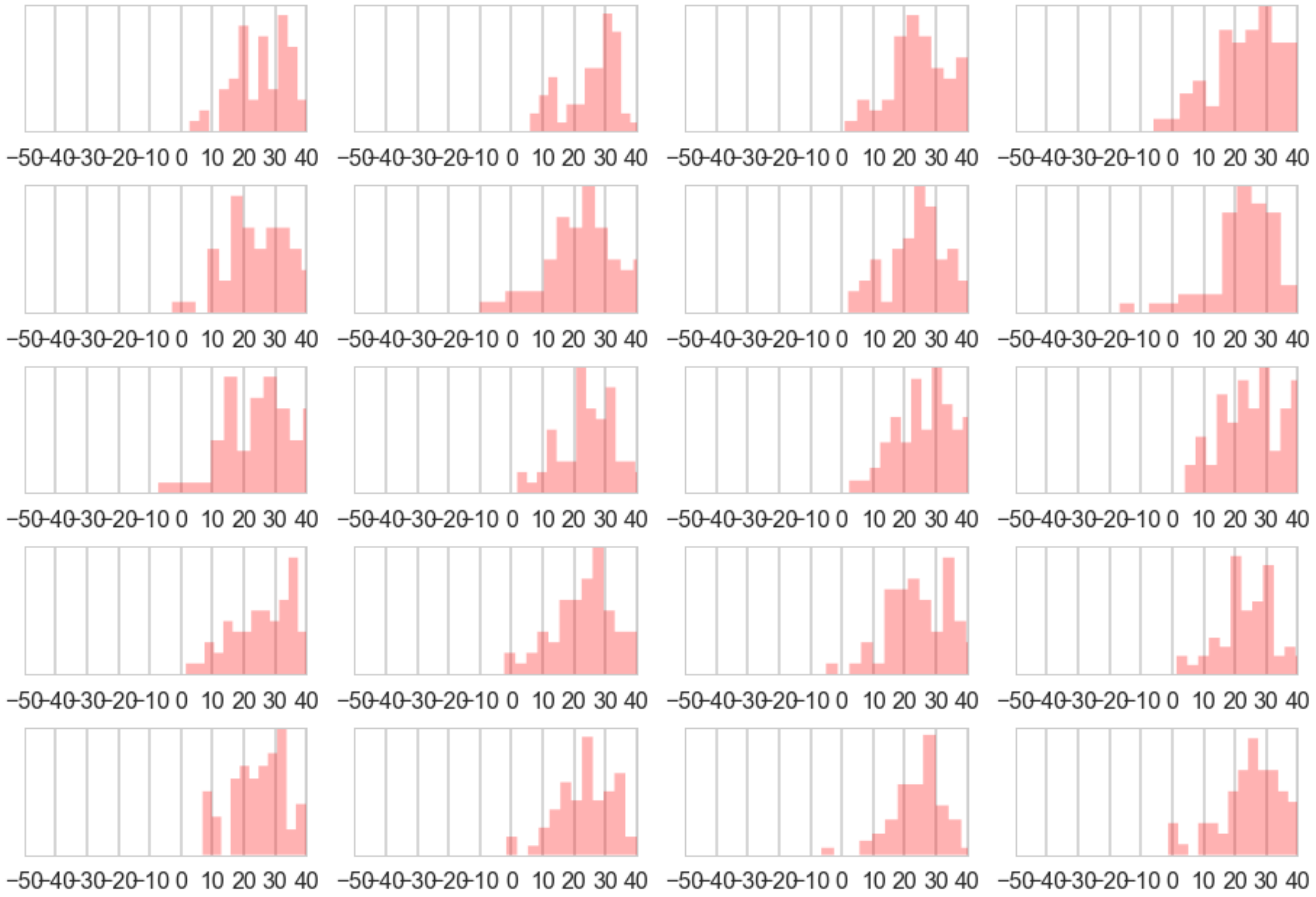
$$\begin{array}{cccccc} \theta_1 & y_{11} & y_{12} & y_{13} & \cdots & y_{1n_D} \\ \theta_2 & y_{21} & y_{22} & & & \\ \vdots & y_{31} & y_{32} & & & \\ & y_{41} & & & & \\ & y_{51} & & & & \\ \theta_{S-1} & y_{S1} & & & & \\ \theta_S & y_{S1} & y_{S2} & y_{S3} & \cdots & y_{Sn_D} \end{array}$$

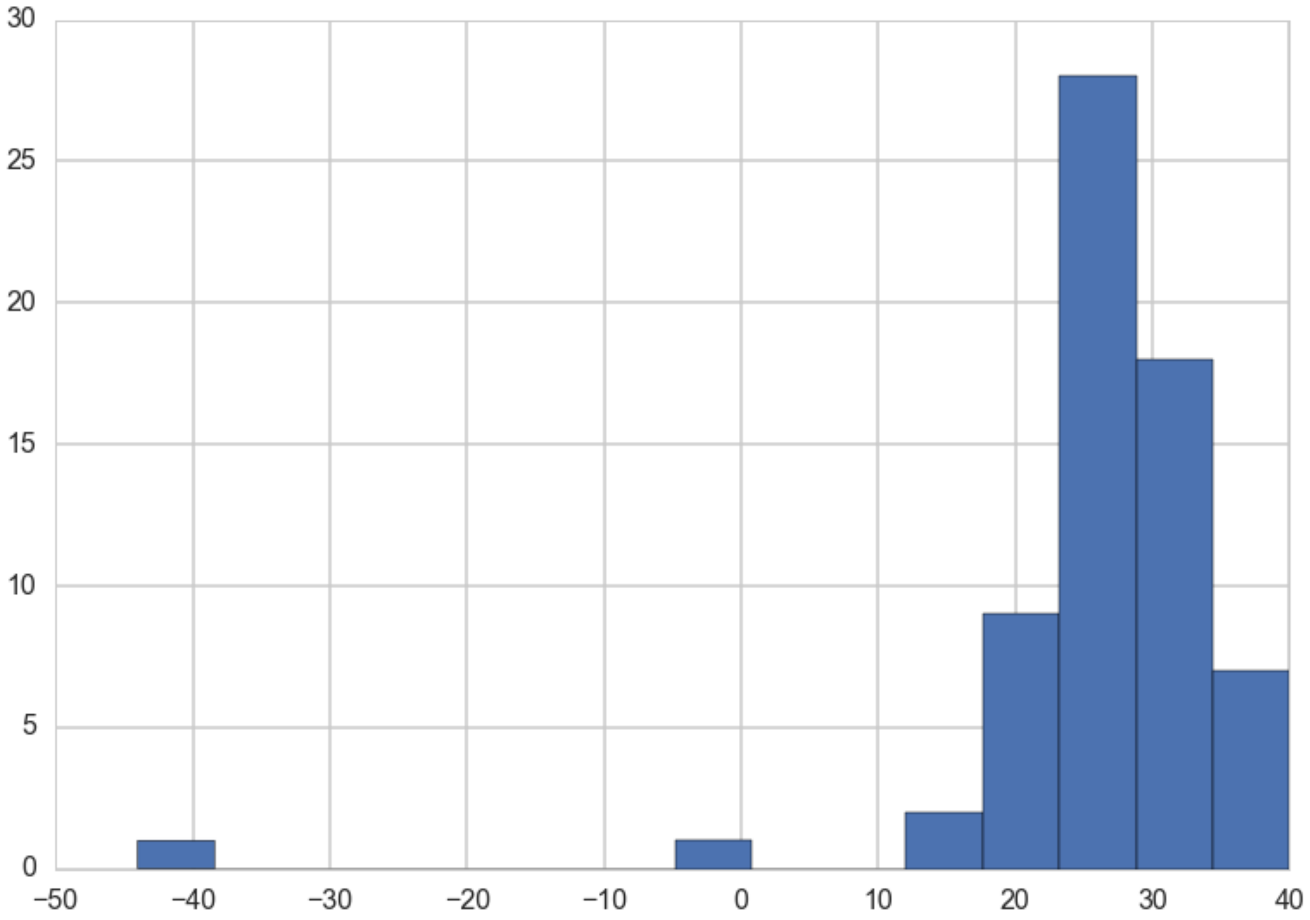
S

For example the minimum value of speed of light in 20 predictive replications.

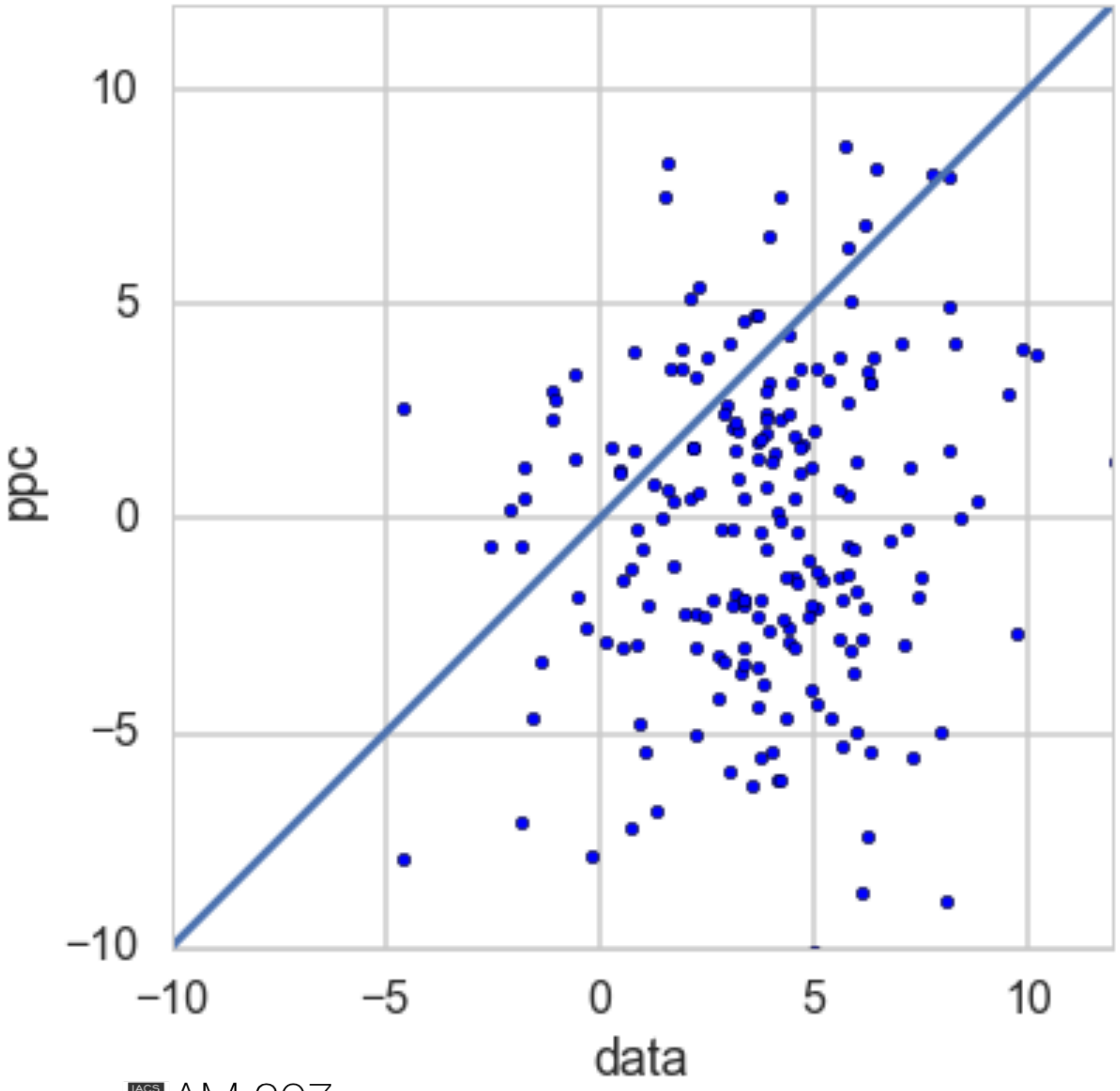


Visual Checking





Do these even look similar??



Bayesian p-values

$$p_B = Pr(T(\{y_r\},\theta) \geq T(\{y\},\theta)|\{y\}),$$

probability over the posterior and posterior predictive (that is, the joint distribution, $p(\theta, \{y_r\}|\{y\})$).

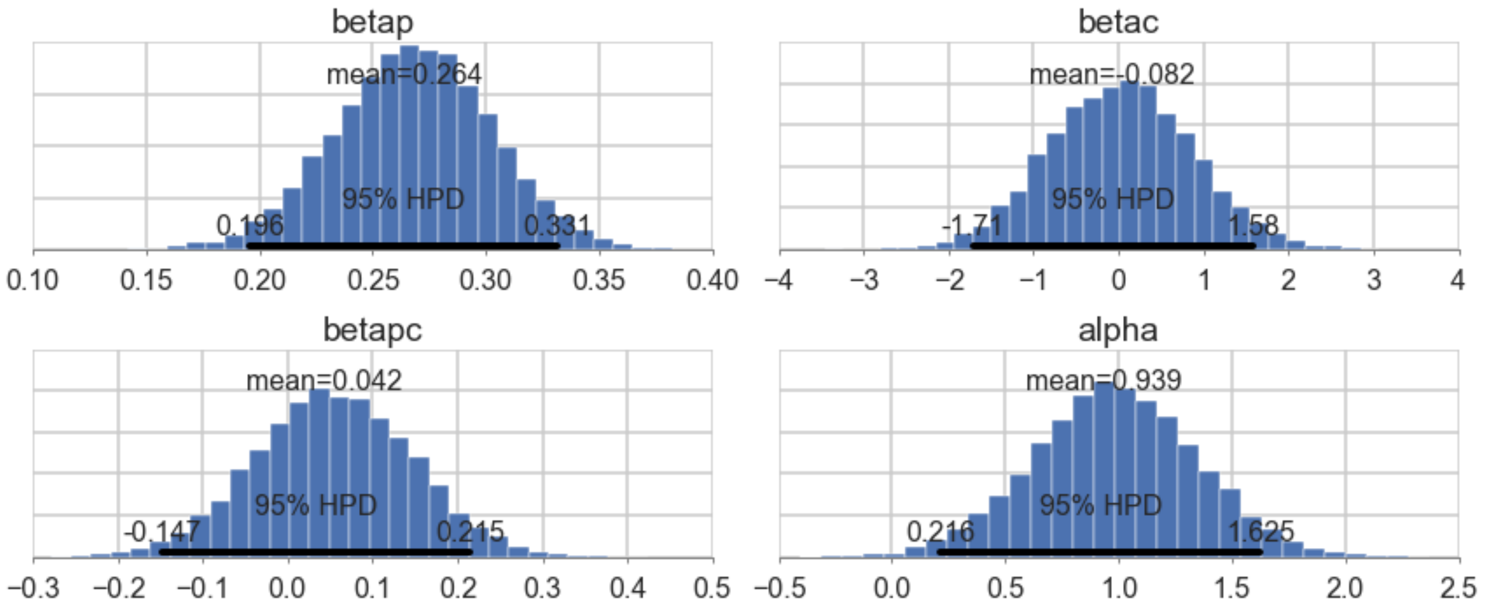
$$p_B = \int d\theta\, d\{y_r\}\, I(T(\{y_r\},\theta) \geq T(\{y\},\theta))\, p(\{y_r\}|\theta) p(\theta|\{y\})$$

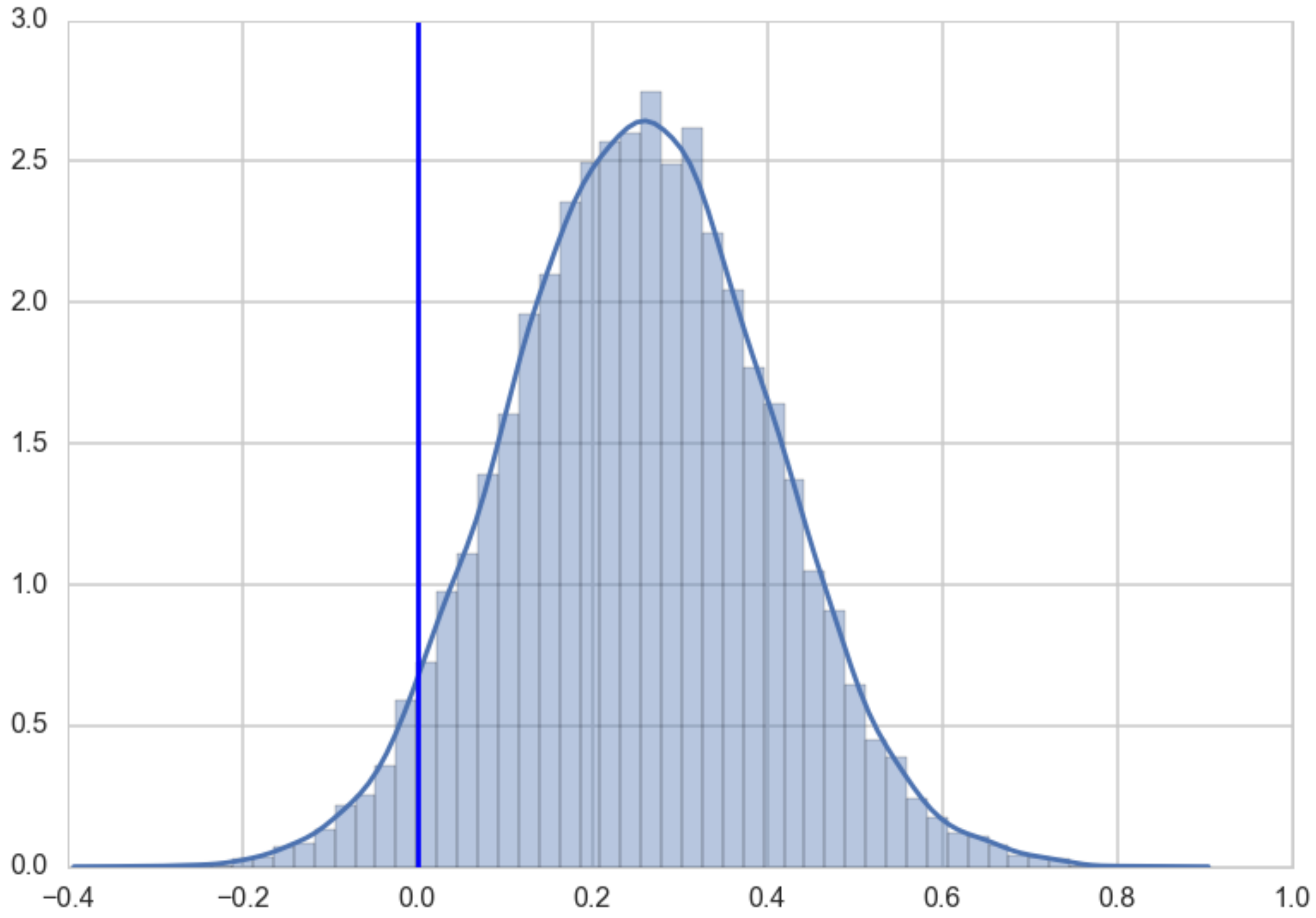
using $p(\{y_r\}|\theta, \{y\}) = p(\{y_r\}|\theta)$.

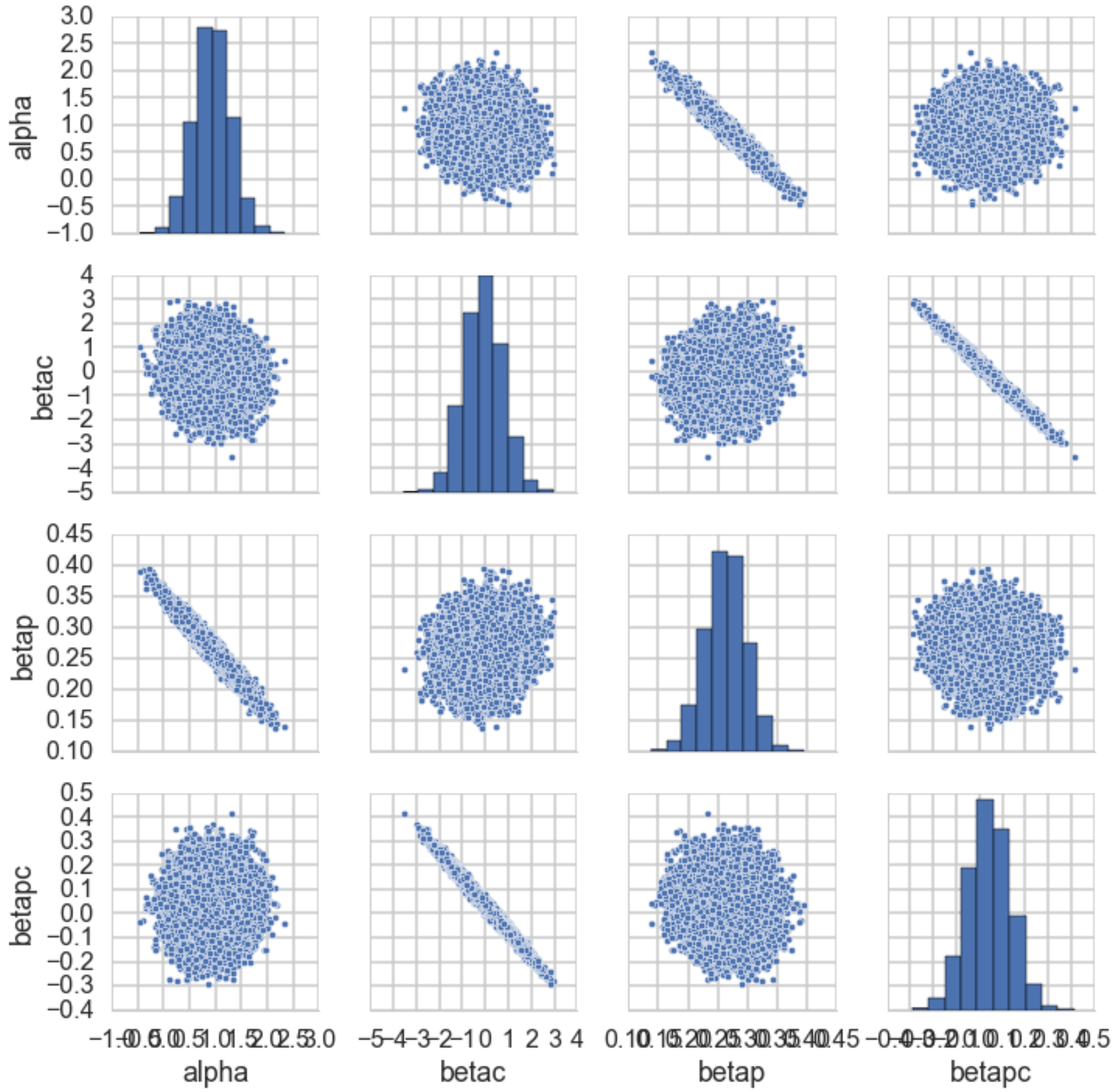
Appropriate usage

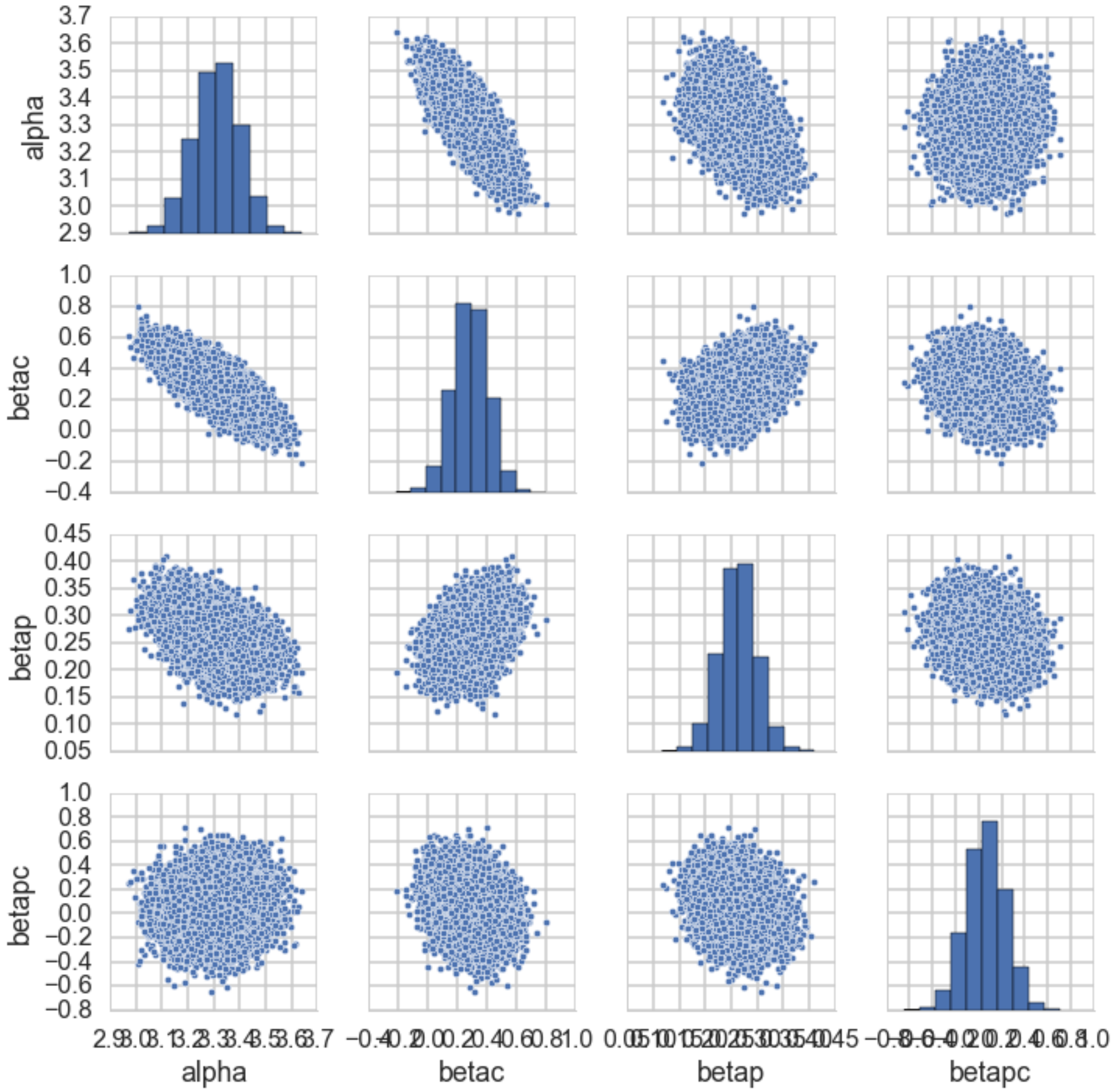
Gelman: *Finding an extreme p-value and thus ‘rejecting’ a model is never the end of an analysis; the departures of the test quantity in question from its posterior predictive distribution will often suggest improvements of the model or places to check the data, as in the speed of light example. Moreover, even when the current model seems appropriate for drawing inferences (in that no unusual deviations between the model and the data are found), the next scientific step will often be a more rigorous experiment incorporating additional factors, thereby providing better data.*

Oceanic Tools counterfactual checking









Solution is centering, see in n_{eff}, can use model comparison

MODEL COMPARISON AND ENSEMBLING

Model comparison

The key idea in model comparison is that we will sort our average utilities in some order. The exact values are not important, and may be computed with respect to some true distribution or true-belief distribution M_{tb}.

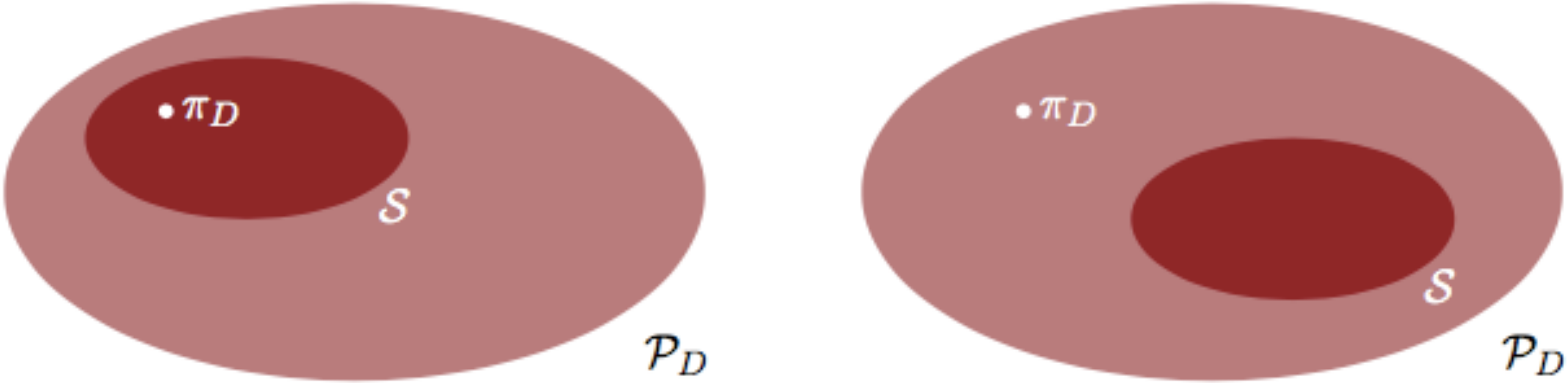
$$\bar{u}(M_k, \hat{a}_k) = \int dy^* u(\hat{a}_k, y^*) p(y^* | D, M_{tb})$$

where $\hat{a}_k$ is the optimal prediction under the model M_k. Now we compare the actions, that is, we want:

$$\hat{M} = \arg\max_k \bar{u}(M_k, \hat{a}_k)$$

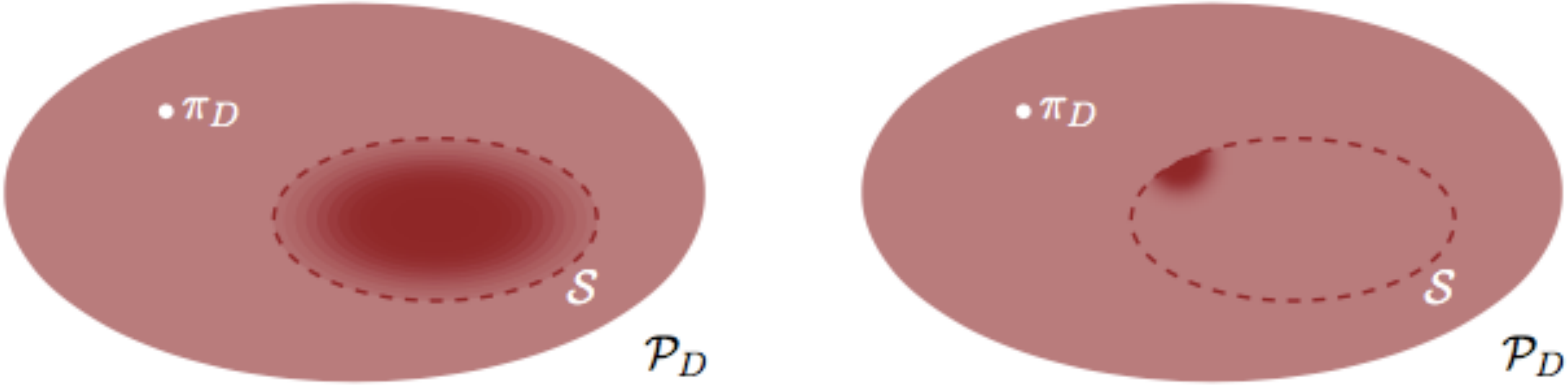
No calibration, but calculating the standard error of the difference can be used to see if the difference is significant, as we did with the WAIC score

Bayesian Inference works in the small world



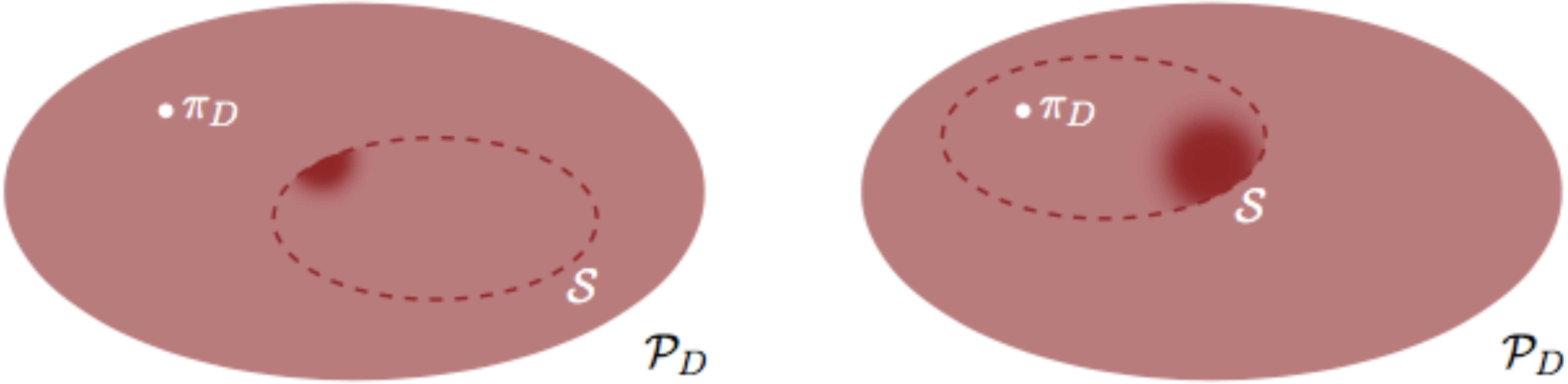
(some times includes the true generating process p_{tb})

Inference in the small world



we go from prior to posterior

Bias and Variance: Overfitting



Overfitting can occur even if the small world includes the true data generating process p_{tb}.

Deviance

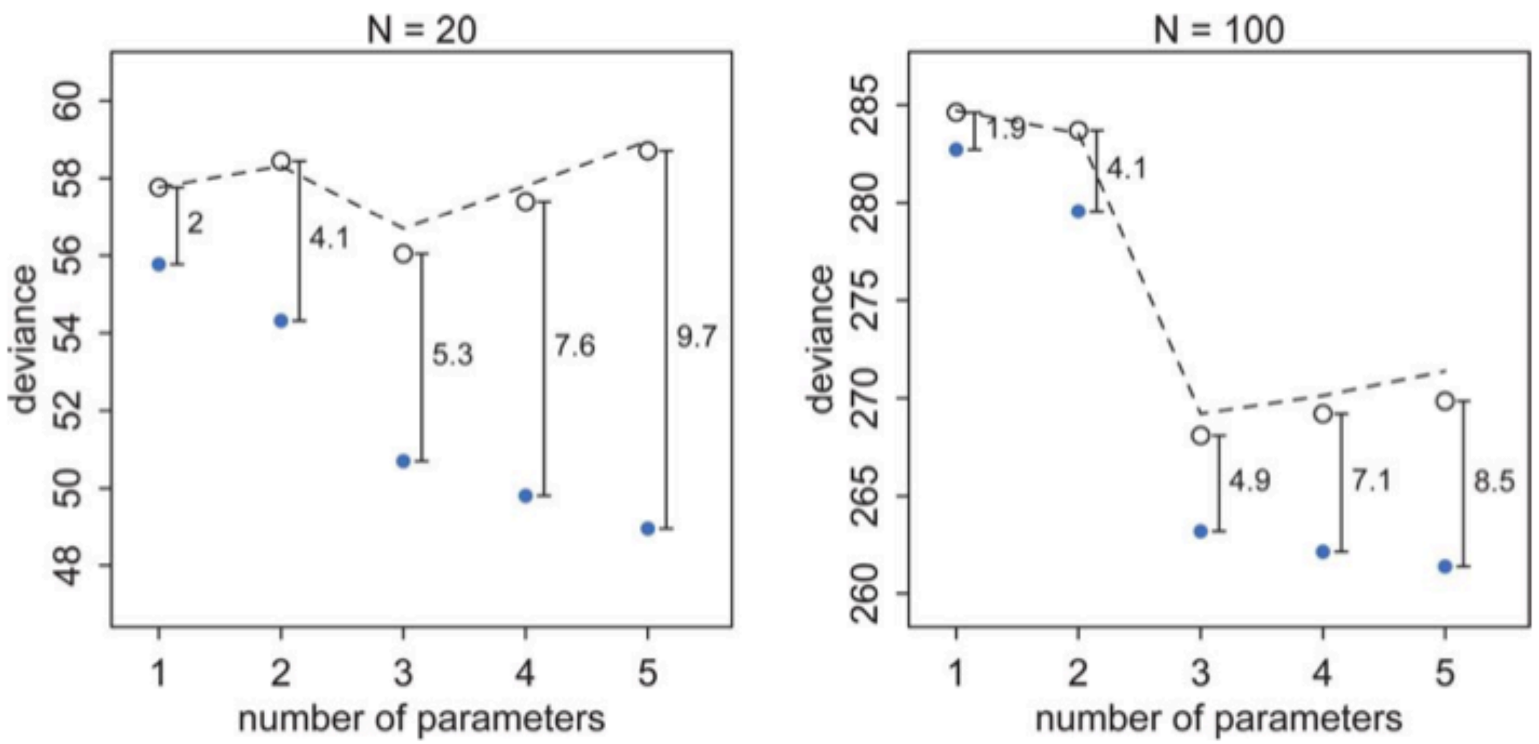
$$D(q) = -2 \sum_i log(q_i),$$

then

$$D_{KL}(p, q) - D_{KL}(p, r) = \frac{2}{N}(D(q) - D(r))$$

More generally: $D(q) = -\frac{N}{2} E_p[log(q)]$

AIC



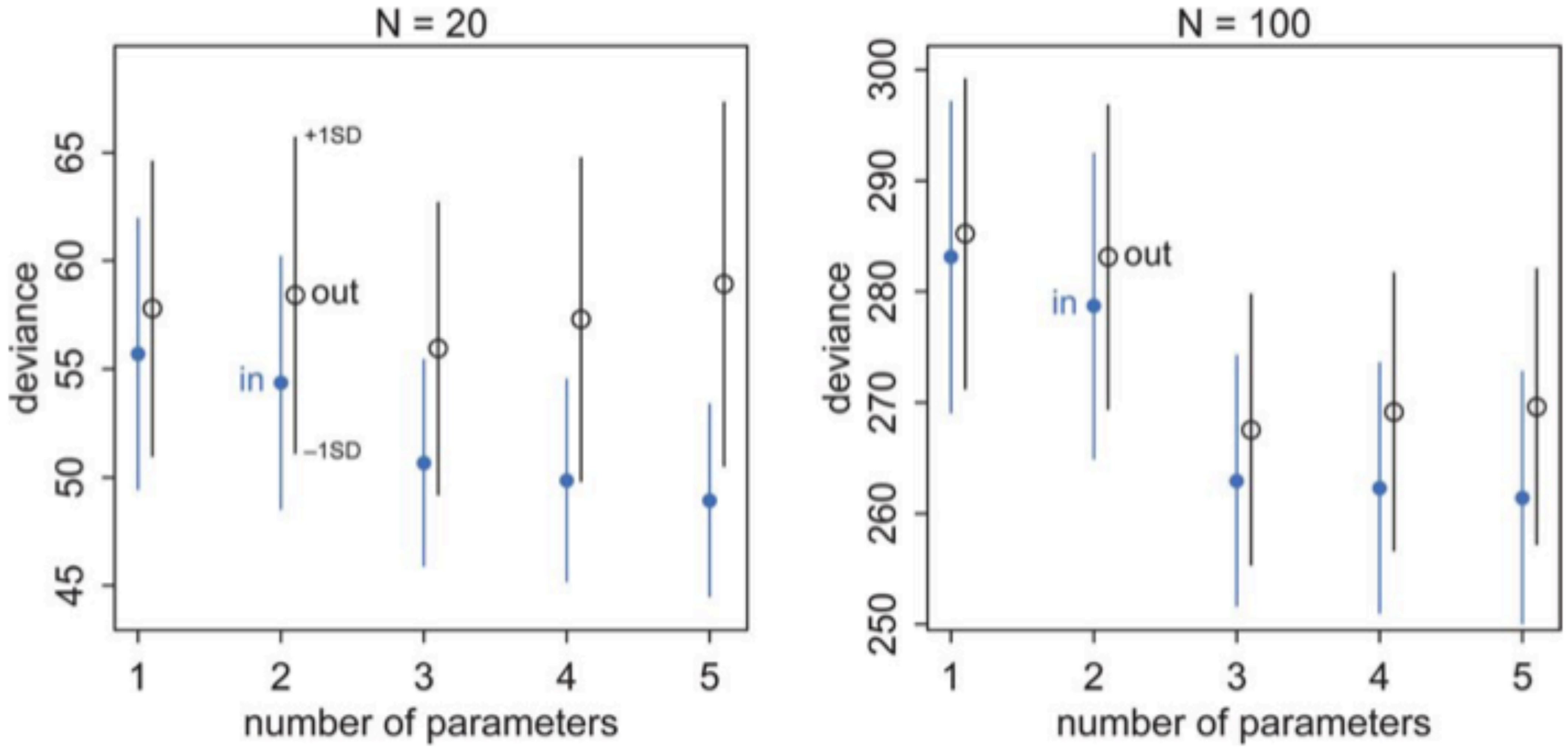
Akaike Information Criterion, or AIC:

$$AIC = D_{train} + 2p$$

$$D_{train} = -2 * log(p(y|\theta_{mle})$$

- multivariate gaussian posterior
- flat priors
- data >> parameters

Train to Test



DIC

- uses the posterior distribution, calculable from MCMC.
- multivariate gaussian posterior distribution.

$$D_{train} = -2 * log(p(y|\theta_{postmean})$$

$$DIC = D_{train} + 2p_D \text{ where}$$

$$p_{DIC} = 2 * (log(p(y|\theta_{postmean}) - E_{post}[log(p(y|\theta)]) \text{ (by monte carlo)}$$

alternative fomulation for p_D, guaranteed to be positive, is

$$p_D = 2 * Var_{post}[log(p(y|\theta_{postmean}))]$$

Bayesian deviance

- $D(q) = -\frac{N}{2} E_p[log(pp(y))]$ posterior predictive for points y on the test set or future data
- replace joint posterior predictive over new points y by product of marginals: ELPD: $\sum_i E_p[log(pp(y_i))]$
- Since we do not know the true distribution p, replace elpd: $\sum_i E_p[log(pp(y_i))]$ by the computed "log pointwise predictive density" (lppd) **in-sample**

$$\sum_j log \langle p(y_j|\theta) \rangle = \sum_j log \left(\frac{1}{S} \sum_s p(y_j|\theta_s) \right)$$

WAIC

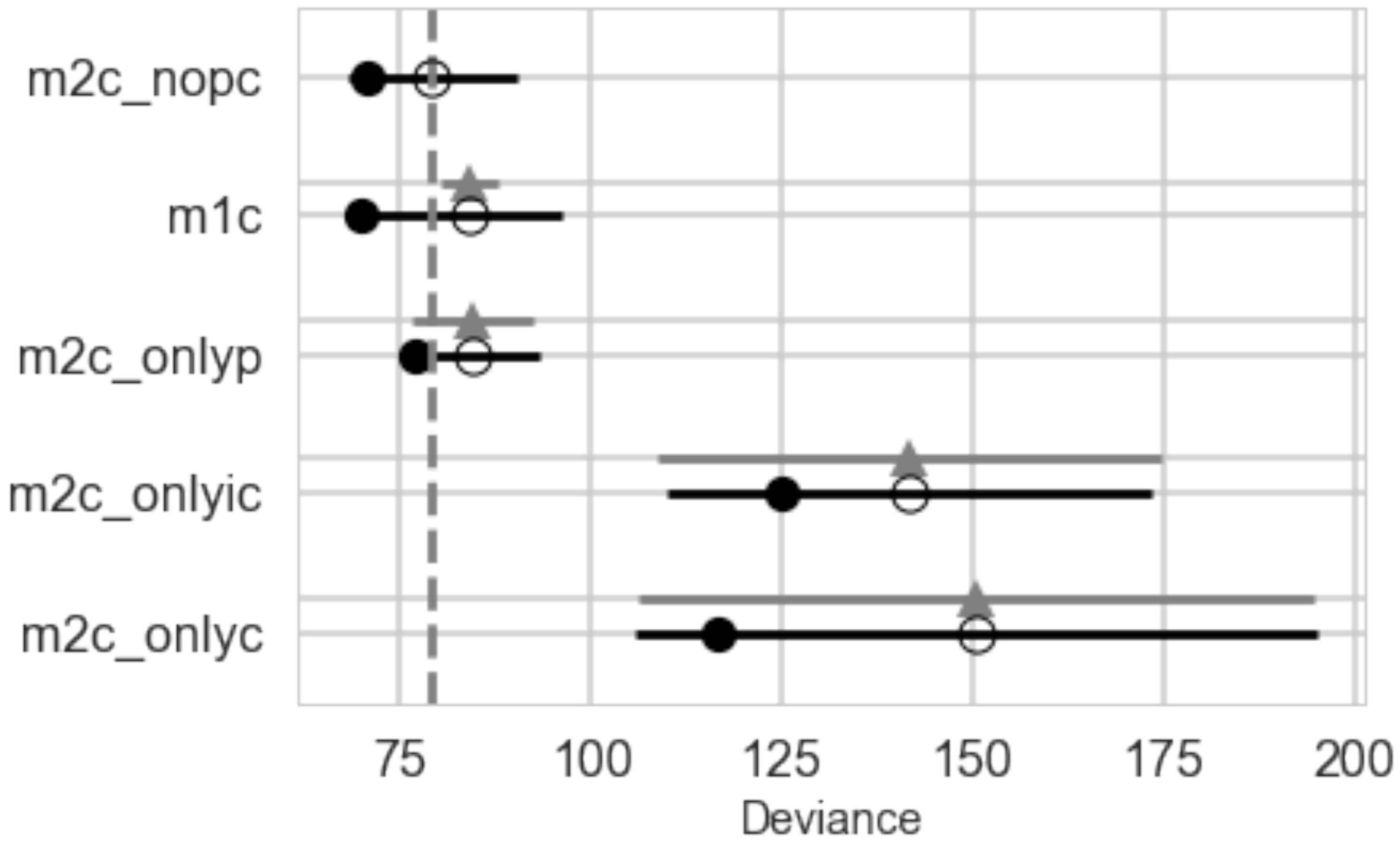
$$WAIC = lppd + 2p_W$$

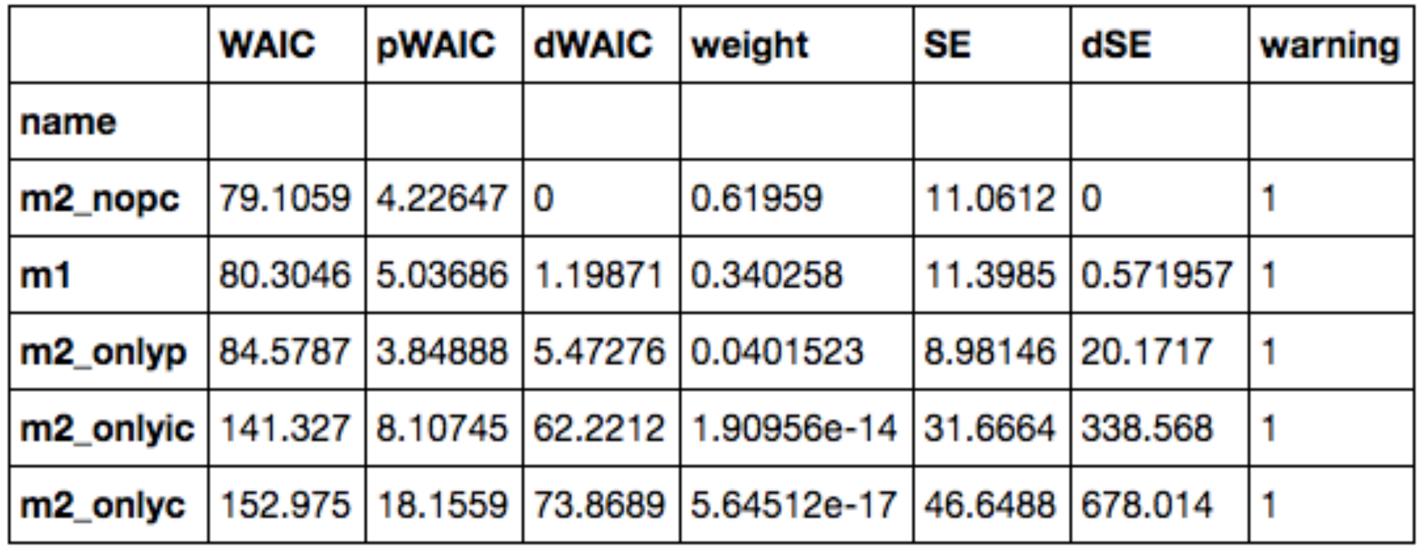
where

$$p_W = 2\sum_i \left(log(E_{post}[p(y_i|\theta)] - E_{post}[log(p(y_i|\theta))]\right)$$

Once again this can be estimated by $\sum_i Var_{post}[log(p(y_i|\theta))]$

Centered vs Uncentered





	WAIC	pWAIC	dWAIC	weight	SE	dSE	warning
name							
m2_nopc	79.1059	4.22647	0	0.61959	11.0612	0	1
m1	80.3046	5.03686	1.19871	0.340258	11.3985	0.571957	1
m2_onlyp	84.5787	3.84888	5.47276	0.0401523	8.98146	20.1717	1
m2_onlyic	141.327	8.10745	62.2212	1.90956e-14	31.6664	338.568	1
m2_onlyc	152.975	18.1559	73.8689	5.64512e-17	46.6488	678.014	1

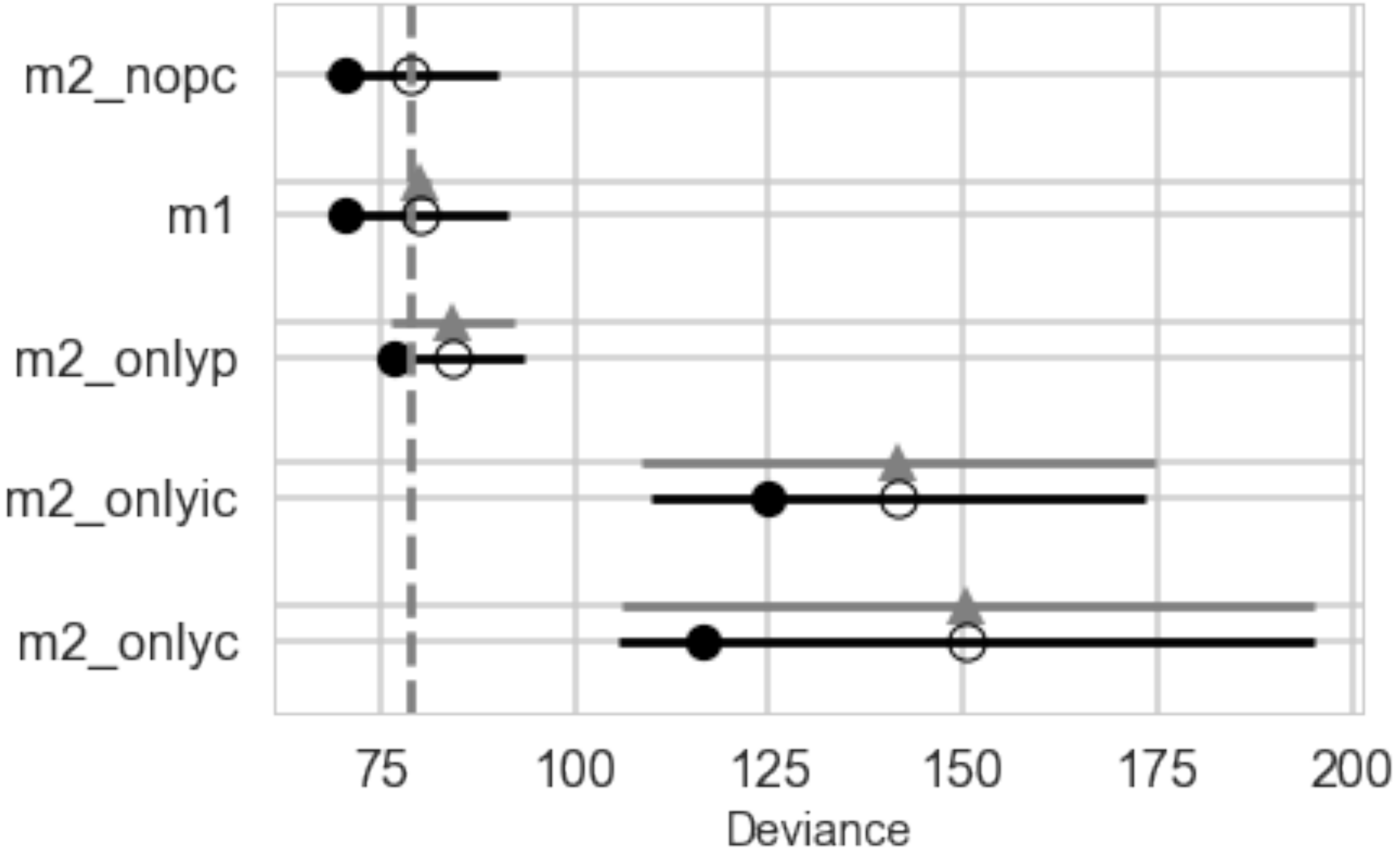
interaction is overfit. centering decorrelates

Definitions

- dWAIC is the difference between each WAIC and the lowest WAIC.
- SE is the standard error of the WAIC estimate.
- dSE is the standard error of the difference in WAIC between each model and the top-ranked model.

$$w_i = \frac{exp(-\frac{1}{2}dWAIC_i)}{\sum_j exp(-\frac{1}{2}dWAIC_j)}$$

read each weight as an estimated probability that each model will perform best on future data.



> it is tempting to use information criteria to compare models with different likelihood functions. Is a Gaussian or binomial better? Can't we just let WAIC sort it out?
> Unfortunately, WAIC (or any other information criterion) cannot sort it out. The problem is that deviance is part normalizing constant. The constant affects the absolute magnitude of the deviance, but it doesn't affect fit to data.
>
> — *McElreath*

Use Cross-Validation in such cases

LOOCV

- Fit a model on N-1 data points, and use the Nth point as a validation point.
- the N-point and N-1 point posteriors are likely to be quite similar, use importance sampling. Fit the full posterior once. Then we have

$$w_s = \frac{p(\theta_s | y_{-i})}{p(\theta_s | y)} \propto \frac{1}{p(y_i | \theta_s, y_{-i})}$$

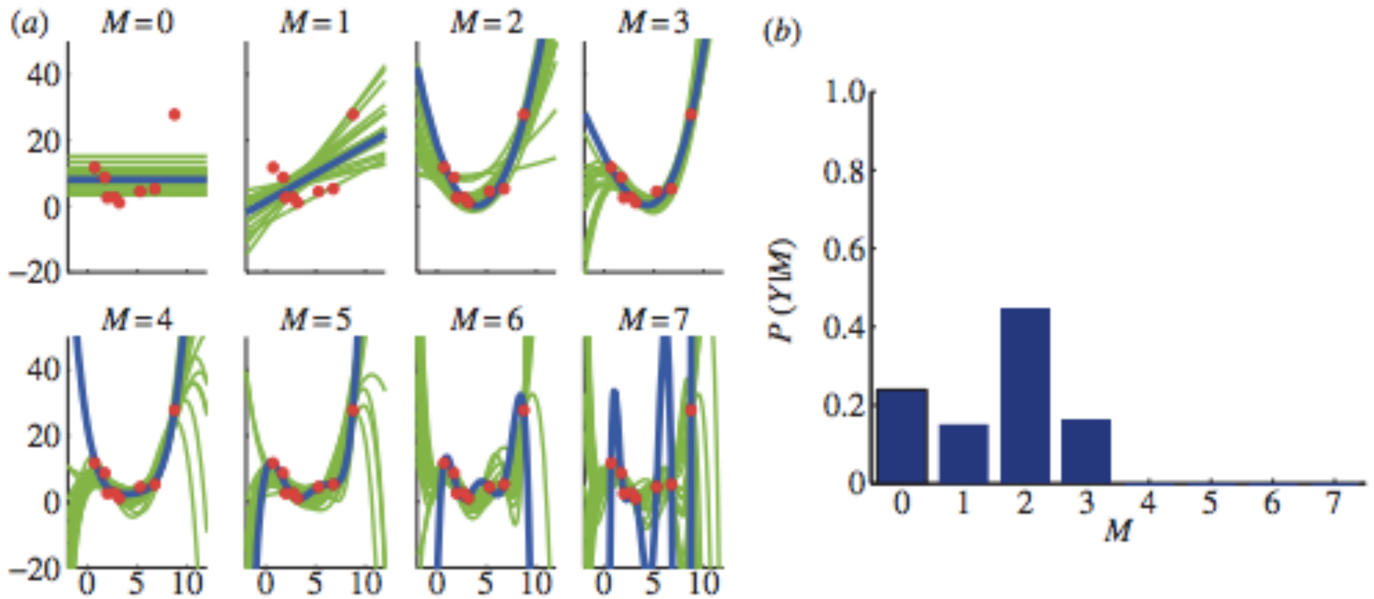
- the importance sampling weights can be unstablein the tails, pymc (`pm.loo`) fits a generalized pareto to the tail (largest 20% importance ratios) for each held out data point i (a MLE fit). Smooths out any large variations.

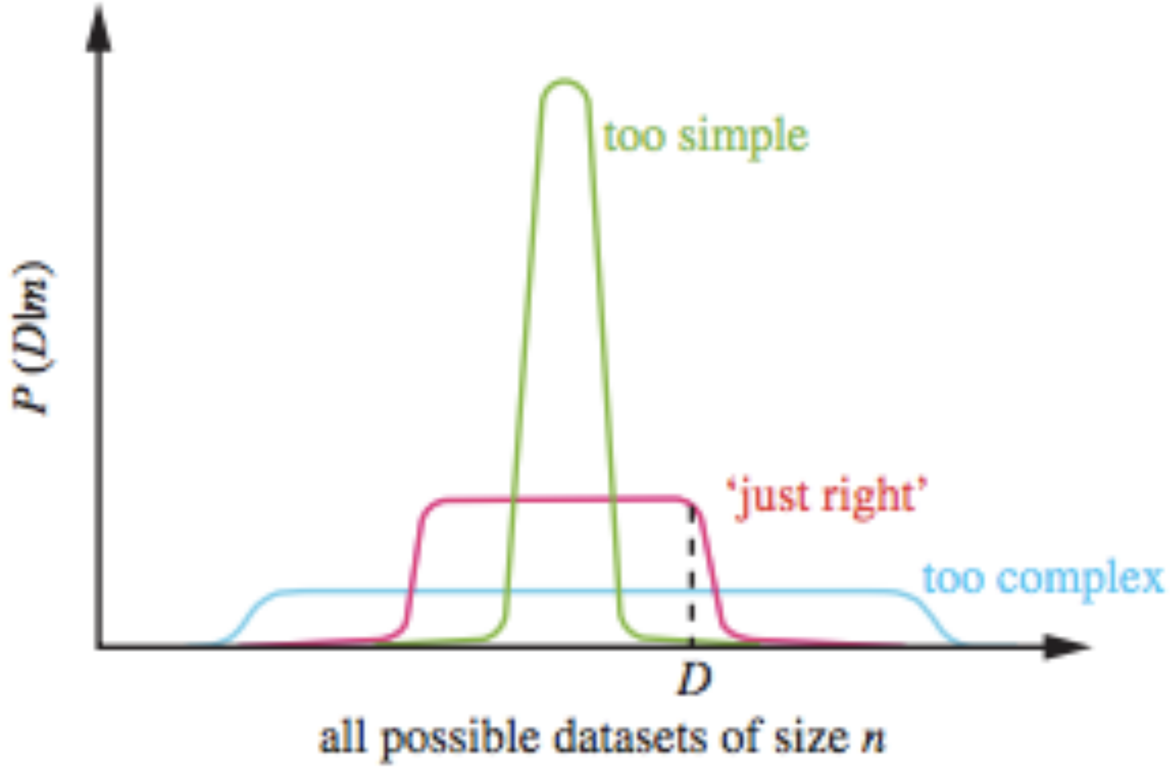
$$elpd_{loo} = \sum_i log(p(y_i | y_{-i})) = \sum_i log\left(\frac{\sum_s w_{is} p(y_i | \theta_s)}{\sum_s w_{is}}\right)$$

What should you use?

1. LOOCV and WAIC are fine. The former can be used for models not having the same likelihood, the latter can be used with models having the same likelihood.
2. WAIC is fast and computationally less intensive, so for same-likelihood models (especially nested models where you are really performing feature selection), it is the first line of attack
3. One does not always have to do model selection. Sometimes just do posterior predictive checks to see how the predictions are, and you might deem it fine.
4. For hierarchical models, WAIC is best for predictive performance within an existing cluster or group. Cross validation is best for new observations from new groups

Evidence for model comparison: Occams Razor





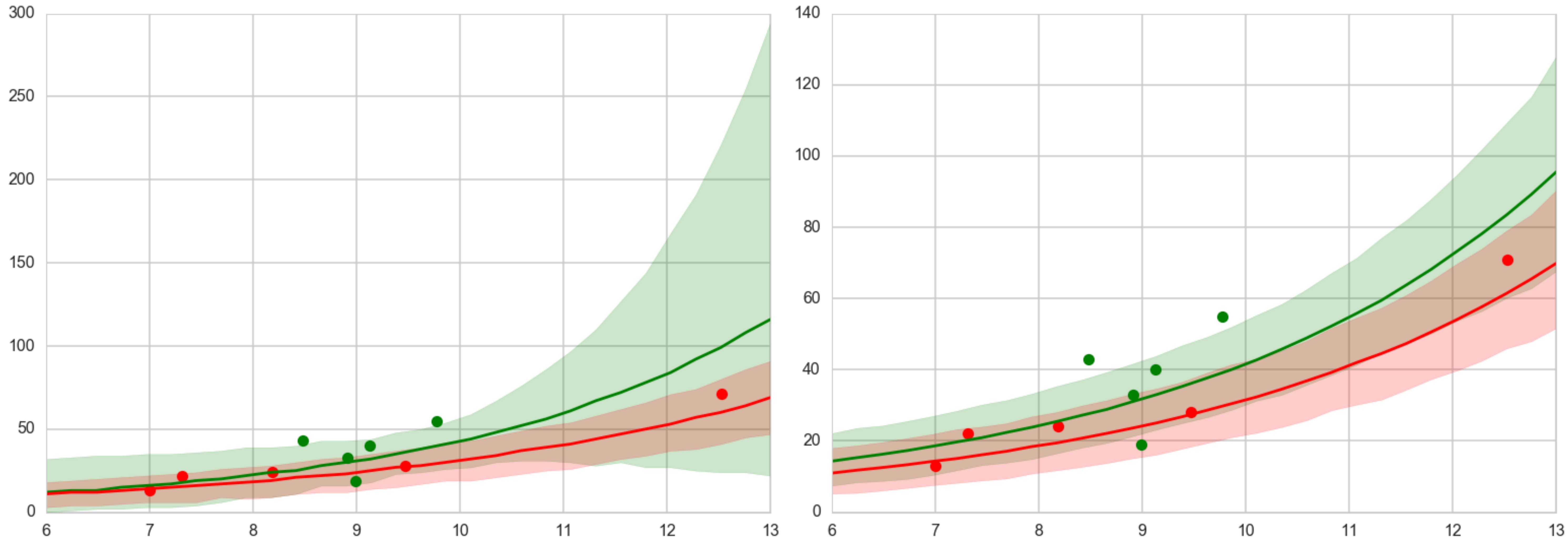
Bayesian Model Averaging

$$p_{BMA}(y^* | x^*, D) = \sum_k p(y^* | x^*, D, M_k) p(M_k | D)$$

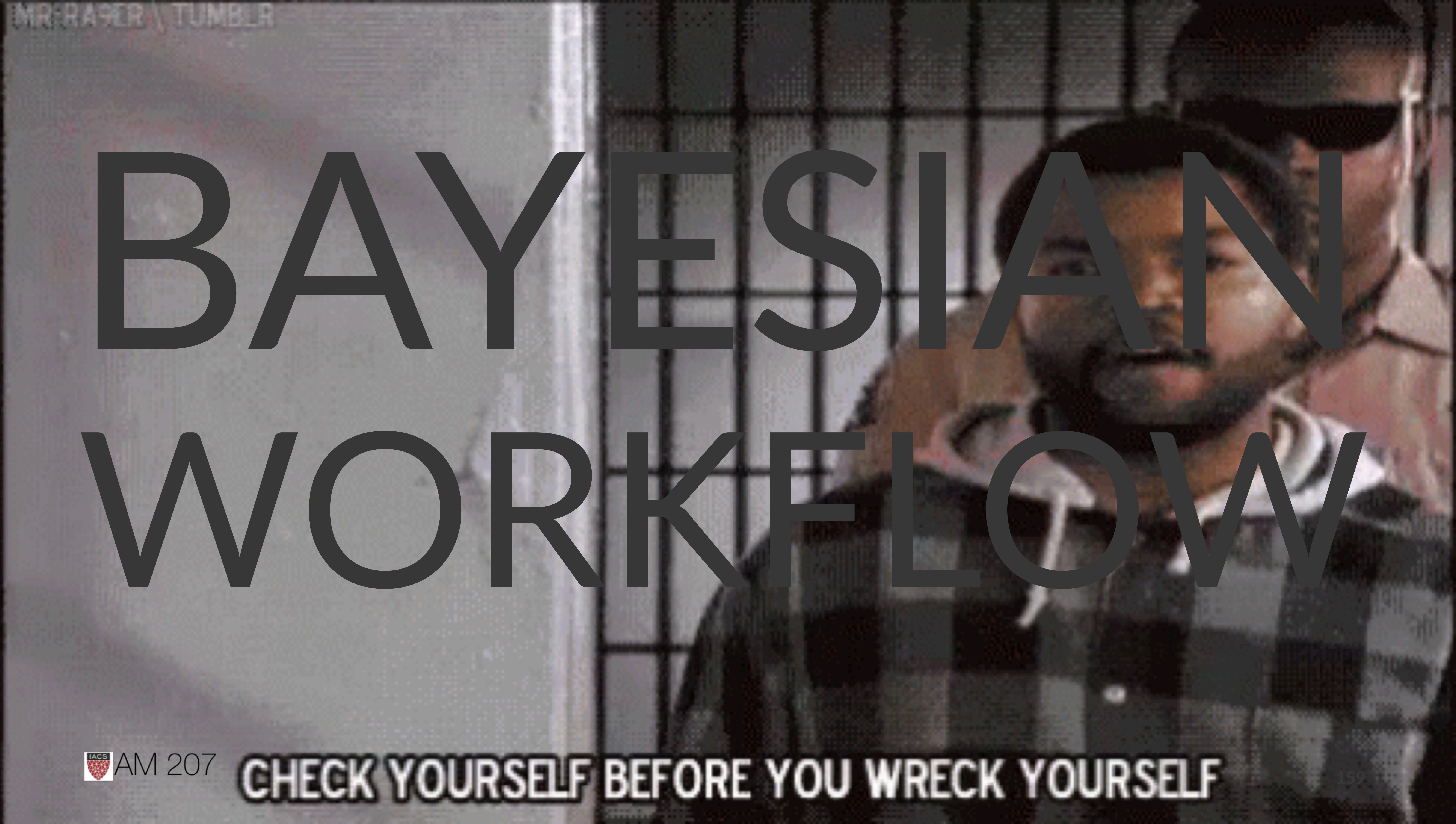
where the averaging is with repect to weights $w_k = p(M_k | D)$, the posterior probabilities of the models M_k.

Use the weights from the WAIC

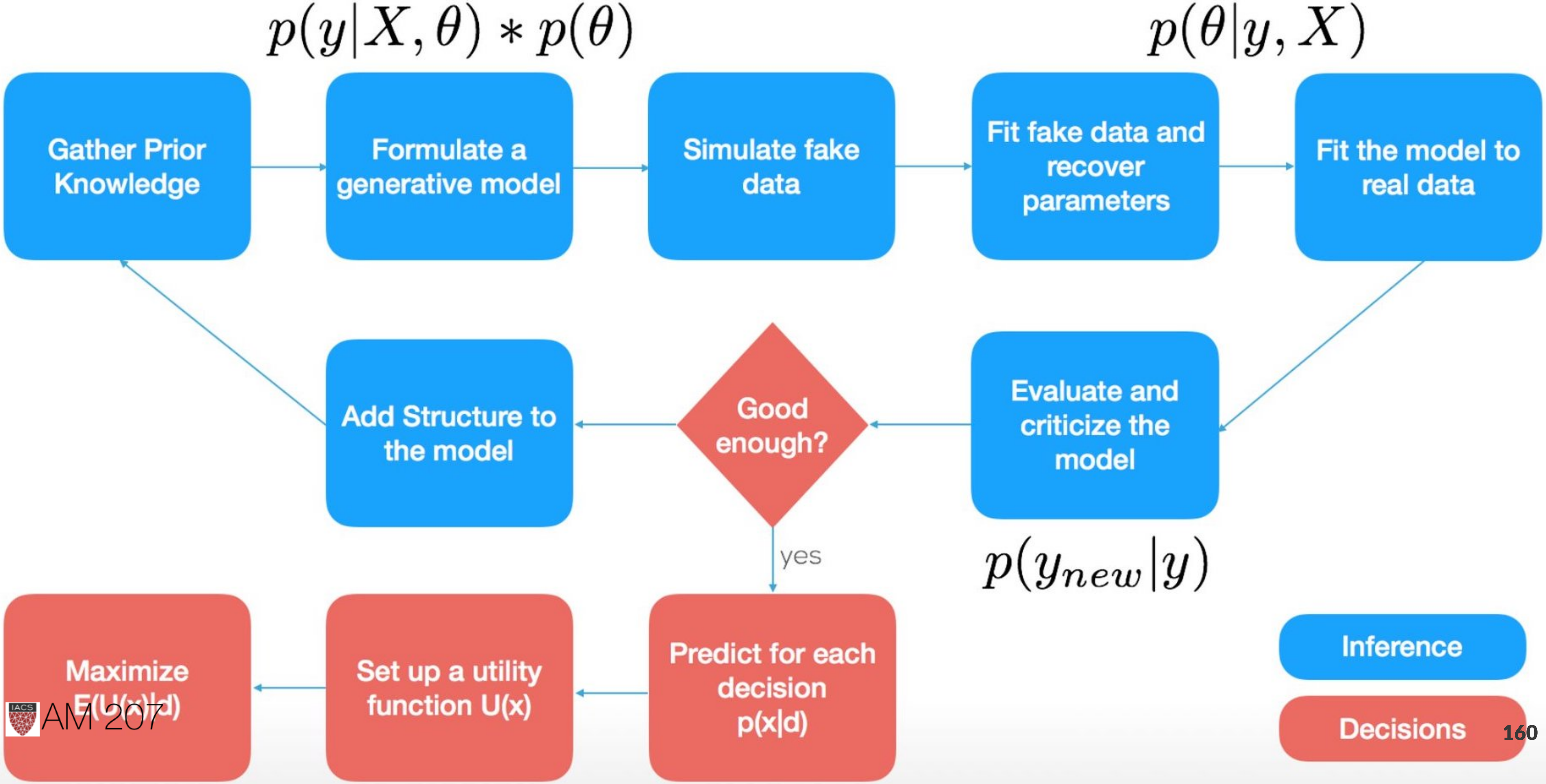
Counterfactual PP and ensembling via weights



BAYESIAN WORKFLOW

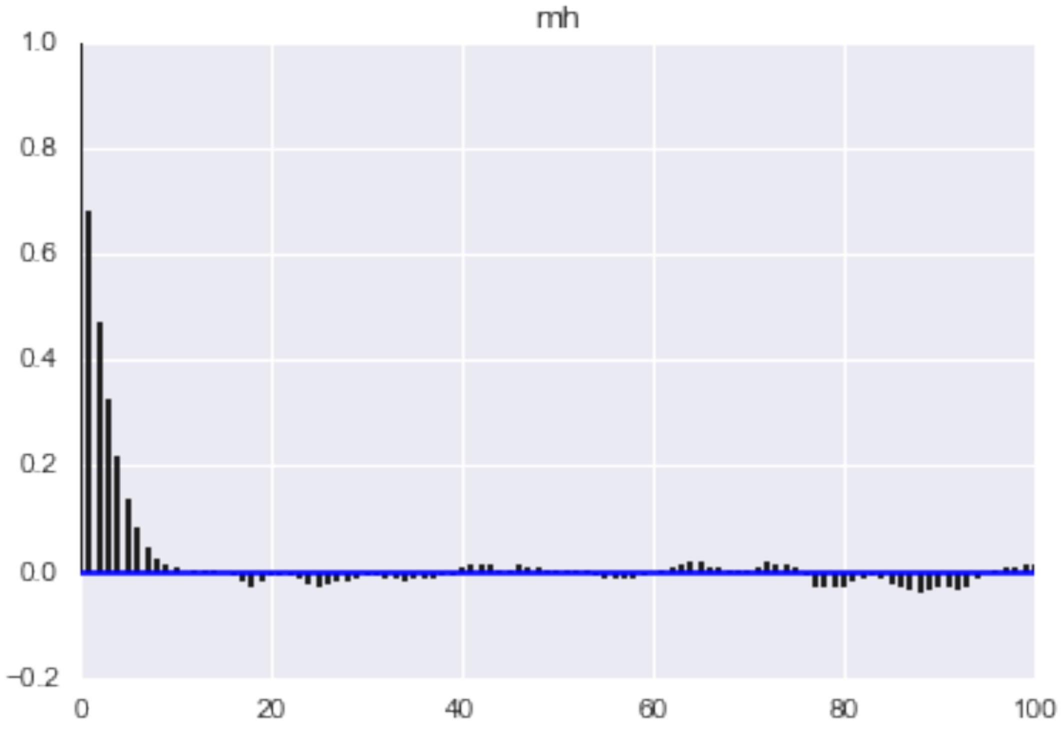


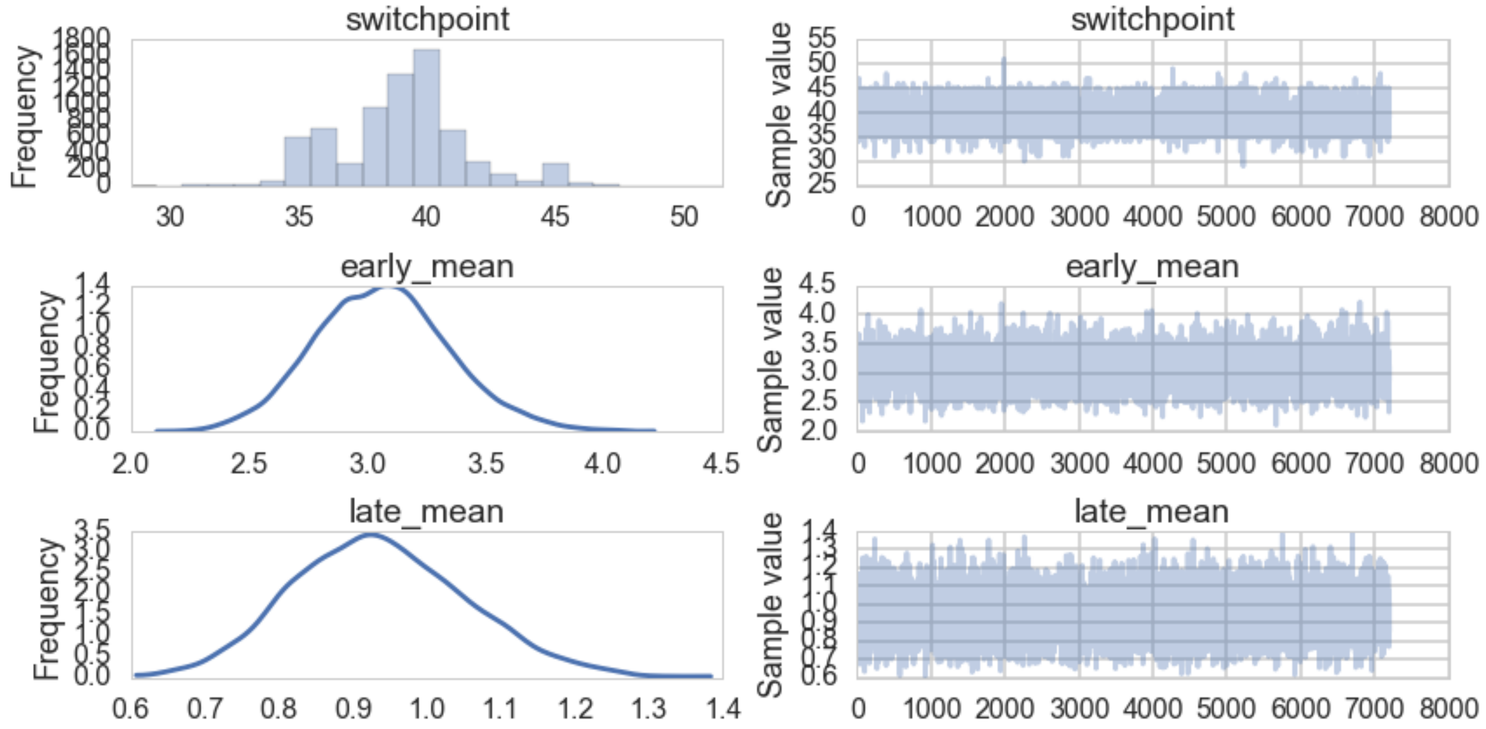
Bayesian Workflow
(from @ericnovik)

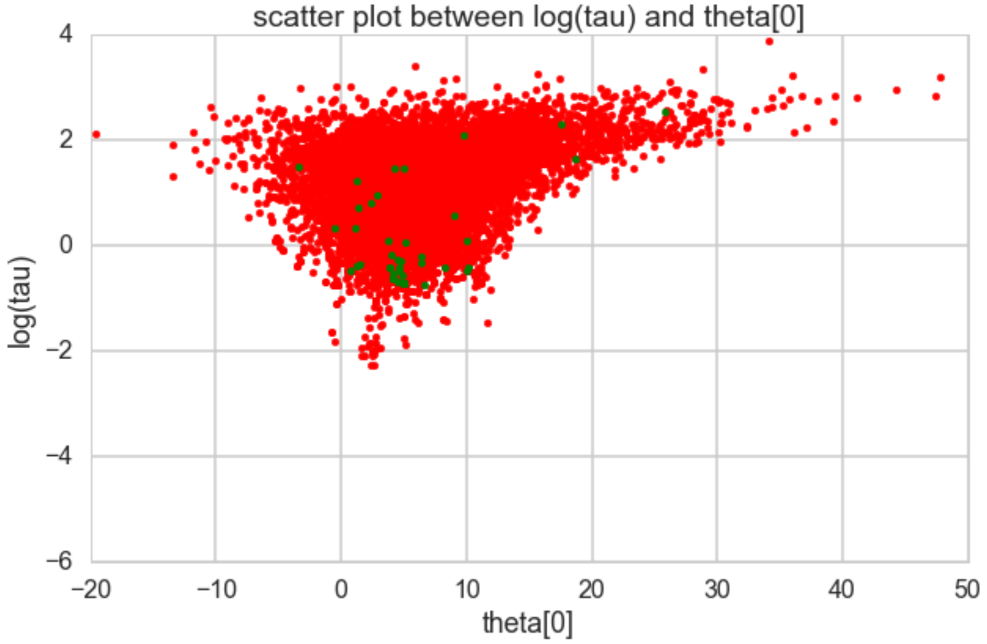

$p(y|X,\theta) * p(\theta)$
$p(\theta|y,X)$
Gather Prior Knowledge
Formulate a generative model
Simulate fake data
Fit fake data and recover parameters
Fit the model to real data
Add Structure to the model
Good enough?
Evaluate and criticize the model
yes
$p(y_{new}|y)$
Maximize E(U(x)|d)
Set up a utility function U(x)
Predict for each decision p(x|d)
Inference
Decisions

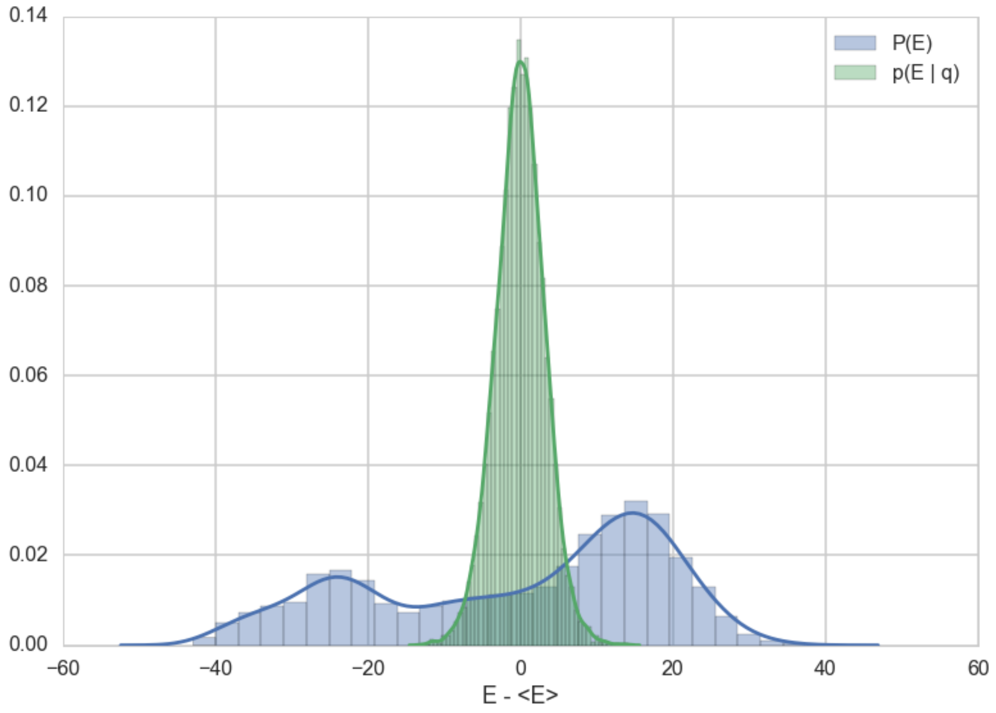
Battery of tests

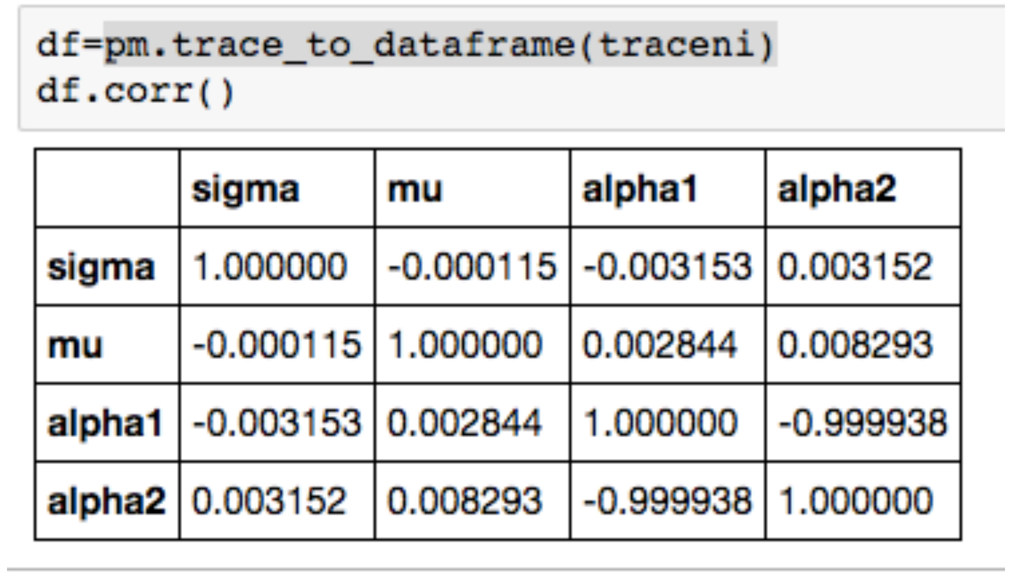
- Visual Inspection
- Gewecke, Gelman Rubin, Effective N, posteriors from various starts
- posterior plots, pairwise posterior plots
- Divergences, energygyplots







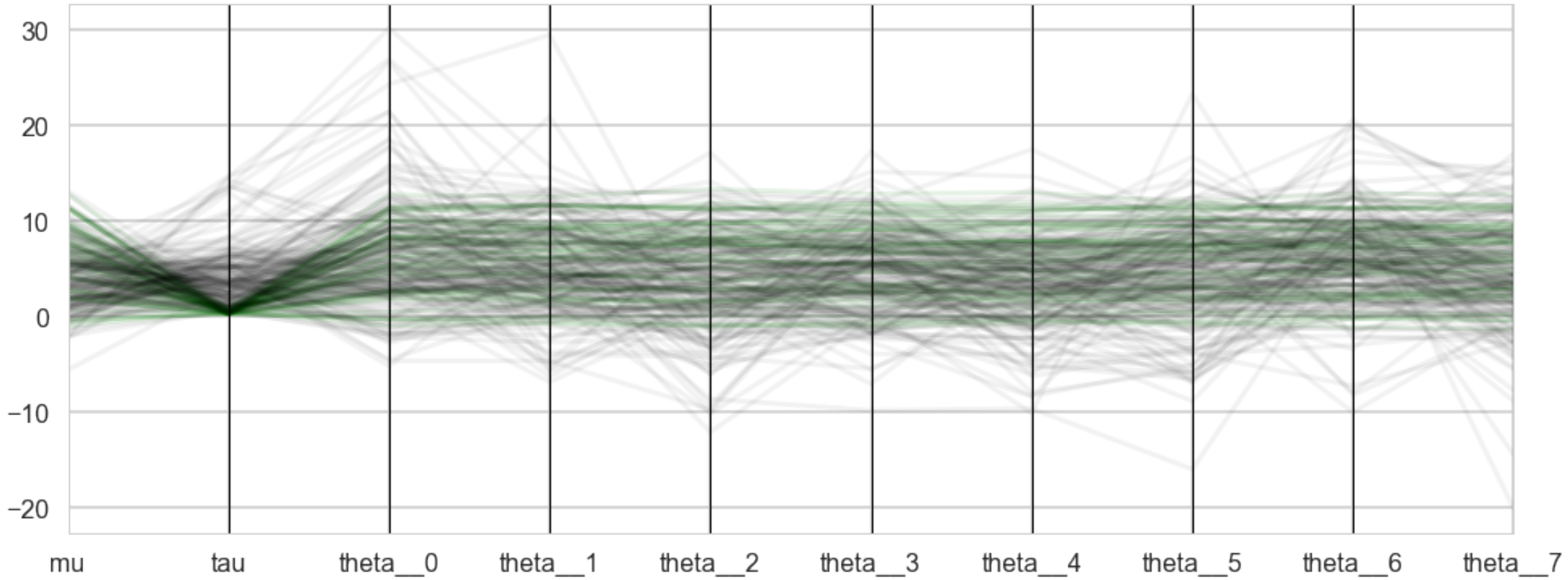



```
df=pm.trace_to_dataframe(traceni)
df.corr()
```

	sigma	mu	alpha1	alpha2
sigma	1.000000	-0.000115	-0.003153	0.003152
mu	-0.000115	1.000000	0.002844	0.008293
alpha1	-0.003153	0.002844	1.000000	-0.999938
alpha2	0.003152	0.008293	-0.999938	1.000000

Parallel Co-ordinates for divergences

see paper on bayesian viz



Model Calibration

Think about the **Bayesian Joint distribution**.

$$p(\theta, y) = p(y \mid \theta)p(\theta)$$

The prior predictive:

$$p(y) = \int d\theta \, p(\theta, y) = \int d\theta \, p(y \mid \theta)p(\theta)$$

How to choose priors?

- mild regularization
- un-informativity
- sensible parameter space
- should correspond to scales and units of process being modeled
- we should calibrate to them

Generate Artificial data sets

- from fixed params, but even better, from priors
- $\tilde{\theta} \sim p(\theta)$
- $\tilde{y} \sim p(y \mid \tilde{\theta})$
- callibrate inferences or decisions by analysing this data
- $U(a) = \int d\tilde{\theta} d\tilde{y} p(\tilde{y}, \tilde{\theta}) U(a(\tilde{y}), \tilde{\theta})$

Now fit a posterior to each generated dataset

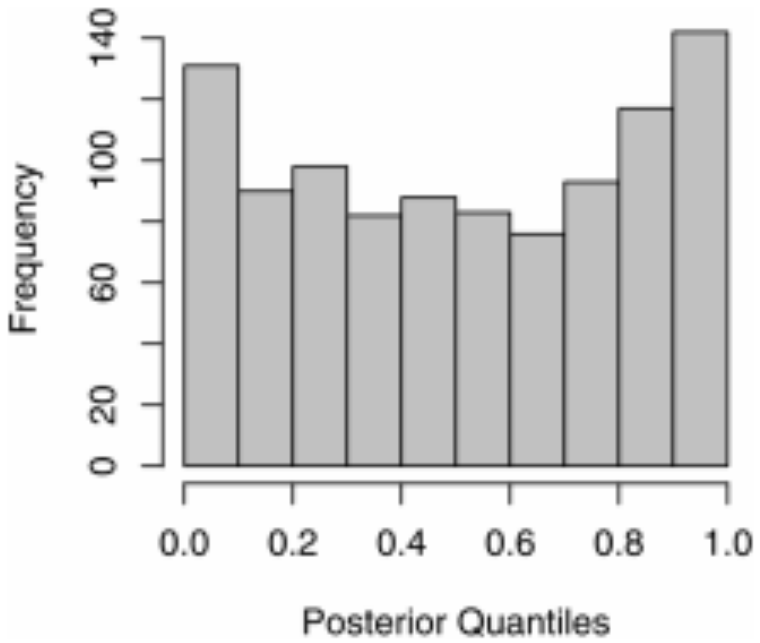
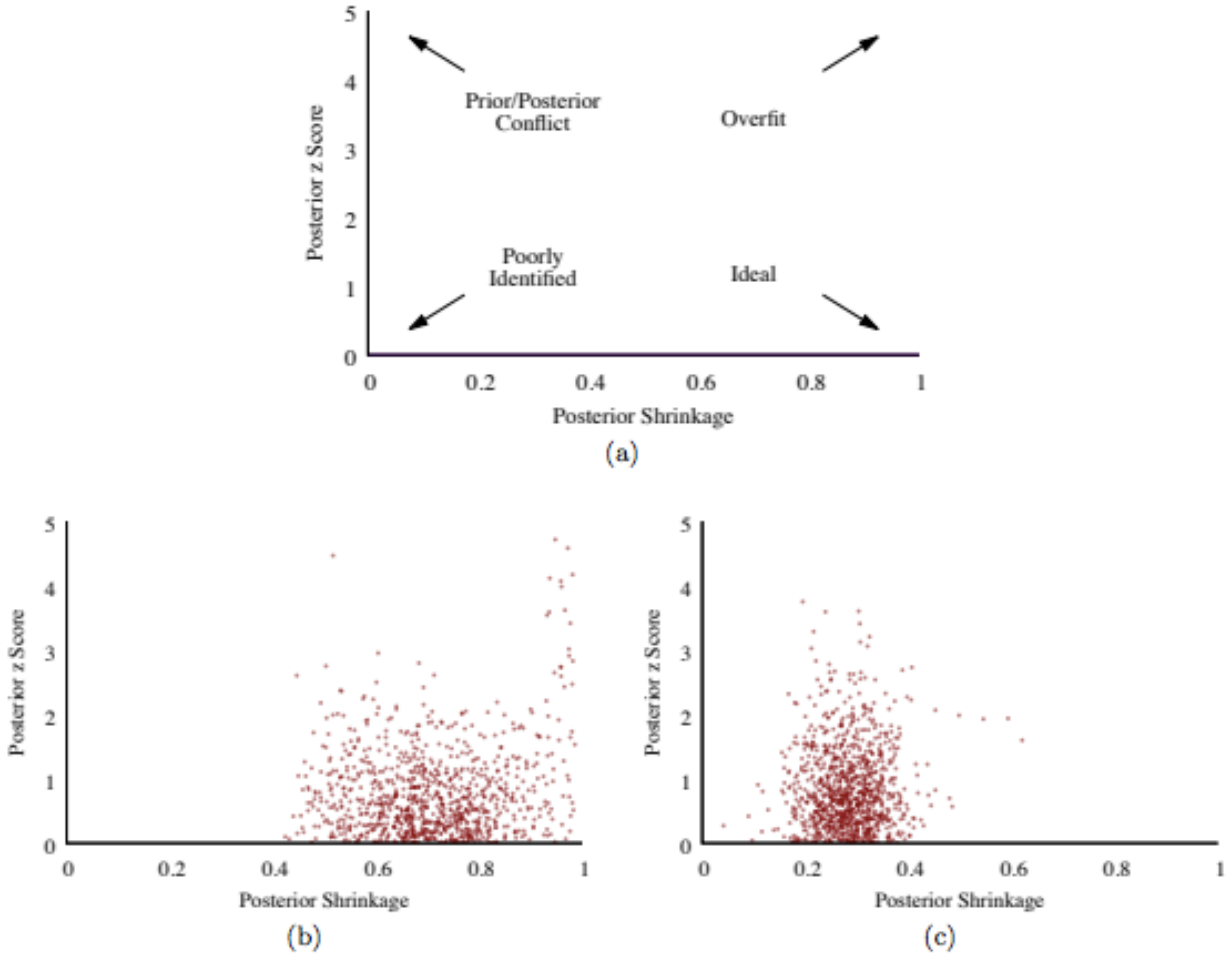


Figure 3. An example of posterior quantiles q from software with error. An effective summary for detecting the error should emphasize quantiles near 0 or 1, such as $h(q) = (\Phi^{-1}(q))^2$.

- see Cook et al
- take each $\tilde{y}$
- get a $\theta \mid \tilde{y}$ posterior
- find the rank of $\tilde{\theta}$ in "its" posterior
- a histogram of ranks should be uniform- this tests our sampling software

Sensitivity of posterior to range allowed by prior



(a)

(b)

(c)

$$z_n = \left| \frac{\mu_n(\theta_n|\tilde{y}) - \tilde{\theta}_n}{\sigma_n(\theta_n|\tilde{y})} \right|$$

$$s_n = 1 - \frac{\sigma_n(\theta_n|\tilde{y})^2}{\tau_n(\tilde{y})^2}$$

where μ and σ are generated-posterior quantities and τ is a prior one, and n indexes the parameters

Then move to the REAL DATA posterior

- now we do posterior predictive checks
- the prior checks have specified possible data distributions that can be generated
- the posterior predictive ought to be a subset of these. If not our model is mis-specified
- this may seem strange as we didnt think priors are data generating
- they are not but are defined with respect to the likelihood

The Workflow (from Betancourt, and Savage)

Prior to Observation

1. Define Data and interesting statistics
2. Build Model
3. Analyze the joint, and its data marginal (prior predictive) and its summary statistics
4. fit posteriors to simulated data to calibrate
    - check sampler diagnostics, and correlate with simulated data
    - use rank statistics to evaluate prior-posterior consistency
    - check posterior behaviors and behaviors of decisions

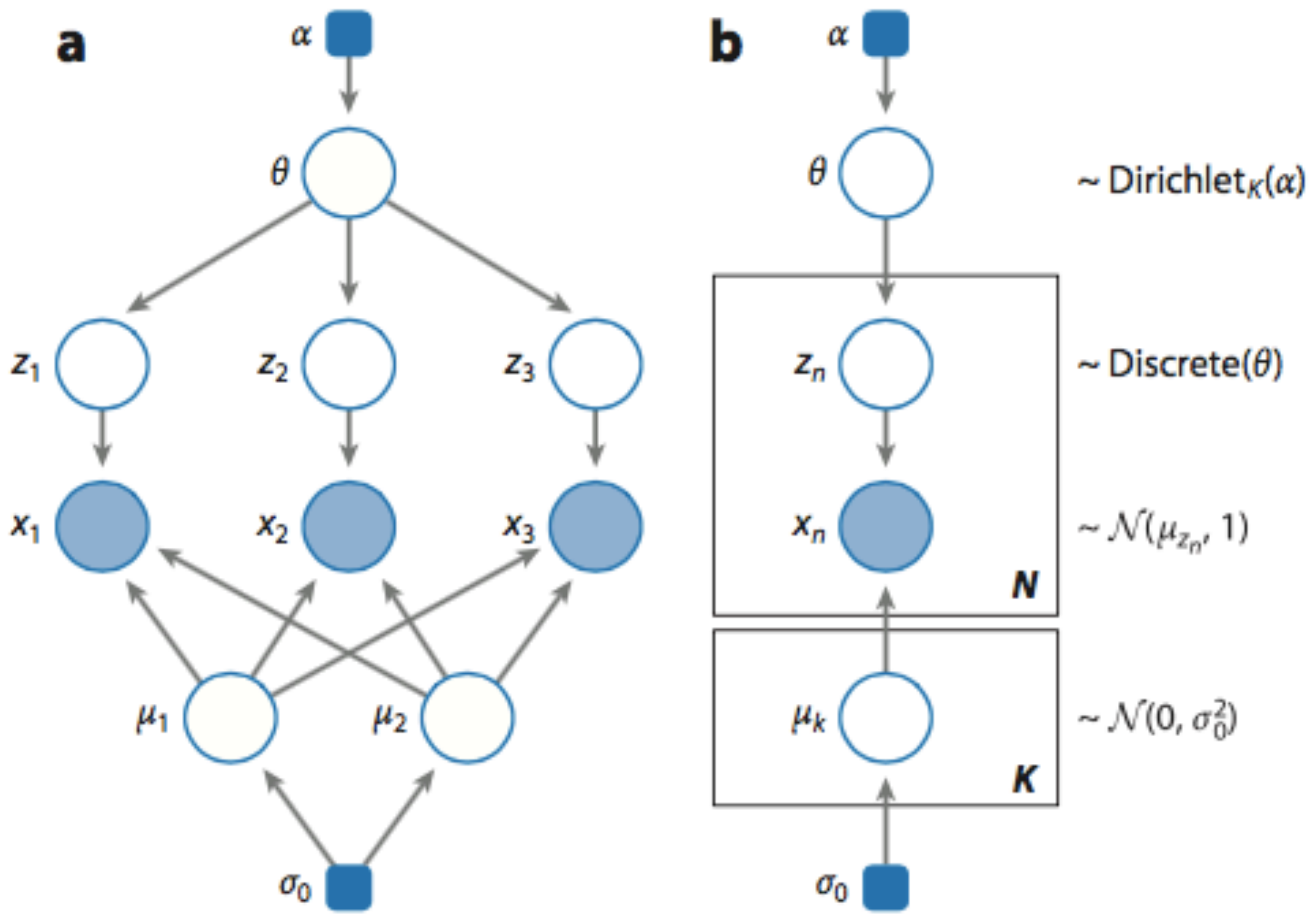
Posterior to Observation

1. Fit the Observed Data and Evaluate the fit
    - check sampler diagnostics, poor performance means generative model not consistent with actual data
2. Analyze the Posterior Predictive Distribution
    - do posterior predictive checks, now comparing actual data with posterior-predictive simulations
    - consider expanding the model
3. Do model comparison
    - usually within a nested model, but you might want to apply a different modeling scheme, in which case use loo
    - you might want to ensemble instead

GENERATIVE MODELS

(with latent variables for things like mixtures)

Mixture Models



A distribution $p(x|\{\theta_k\})$ is a mixture of K component distributions $p_1, p_2, \ldots p_K$ if:

$$p(x|\{\theta_k\}) = \sum_k \lambda_k p_k(x|\theta_k)$$

with the λ_k being mixing weights, $\lambda_k > 0$, $\sum_k \lambda_k = 1$.

Example: Zero Inflated Poisson

Generative Model: How to simulate from it?

$$Z \sim Categorical(\lambda_1, \lambda_2, \ldots, \lambda_K)$$

where Z says which component X is drawn from.

Thus λ_j is the probability that the hidden class variable $z = j$.

Then: $X \sim p_z(x|\theta_z)$ and general structure is:

$$p(x|\theta) = \sum_z p(x, z) = \sum_z p(z)p(x|z, \theta) \text{ where } \theta = \{\theta_k\}.$$

GMM supervised formulation

$$Z \sim \text{Bernoulli}(\lambda)$$
$$X|z=0 \sim \mathcal{N}(\mu_0, \Sigma_0),\, X|z=1 \sim \mathcal{N}(\mu_1, \Sigma_1)$$

Full-data loglike: $l(x, z|\lambda, \mu_0, \mu_1, \Sigma) = -\sum_{i=1}^{m} \log((2\pi)^{n/2} |\Sigma|^{1/2})$

$$-\frac{1}{2}\sum_{i=1}^{m}\sum_{i=1}^{m}(x-\mu_{z_i})^T\,\Sigma^{-1}(x-\mu_{z_i}) + \sum_{i=1}^{m}[z_i\ \log\lambda + (1-z_i)\log(1-\lambda)]$$

Concrete Formulation of unsupervised learning

Estimate Parameters by $\mathbf{x}$-MLE:

$$\begin{aligned} l(x|\lambda, \mu, \Sigma) &= \sum_{i=1}^{m} \log p(x_i|\lambda, \mu, \Sigma) \\ &= \sum_{i=1}^{m} \log \sum_{z} p(x_i|z_i, \mu, \Sigma)\, p(z_i|\lambda) \end{aligned}$$

Not Solvable analytically! EM and Variational. Or do MCMC.

Supervised vs Unsupervised Learning

In **Supervised Learning**, Latent Variables $\mathbf{z}$ are observed.

In other words, we can write the full-data likelihood $p(\mathbf{x}, \mathbf{z})$

In **Unsupervised Learning**, Latent Variables $\mathbf{z}$ are hidden.

We can only write the observed data likelihood:

$$p(\mathbf{x}) = \sum_{z} p(\mathbf{x}, \mathbf{z}) = \sum_{z} p(\mathbf{z}) p(\mathbf{x}|\mathbf{z})$$

Semi-supervised learning

We have some labels, but typically very few labels: not enough to form a good training set. Likelihood a combination.

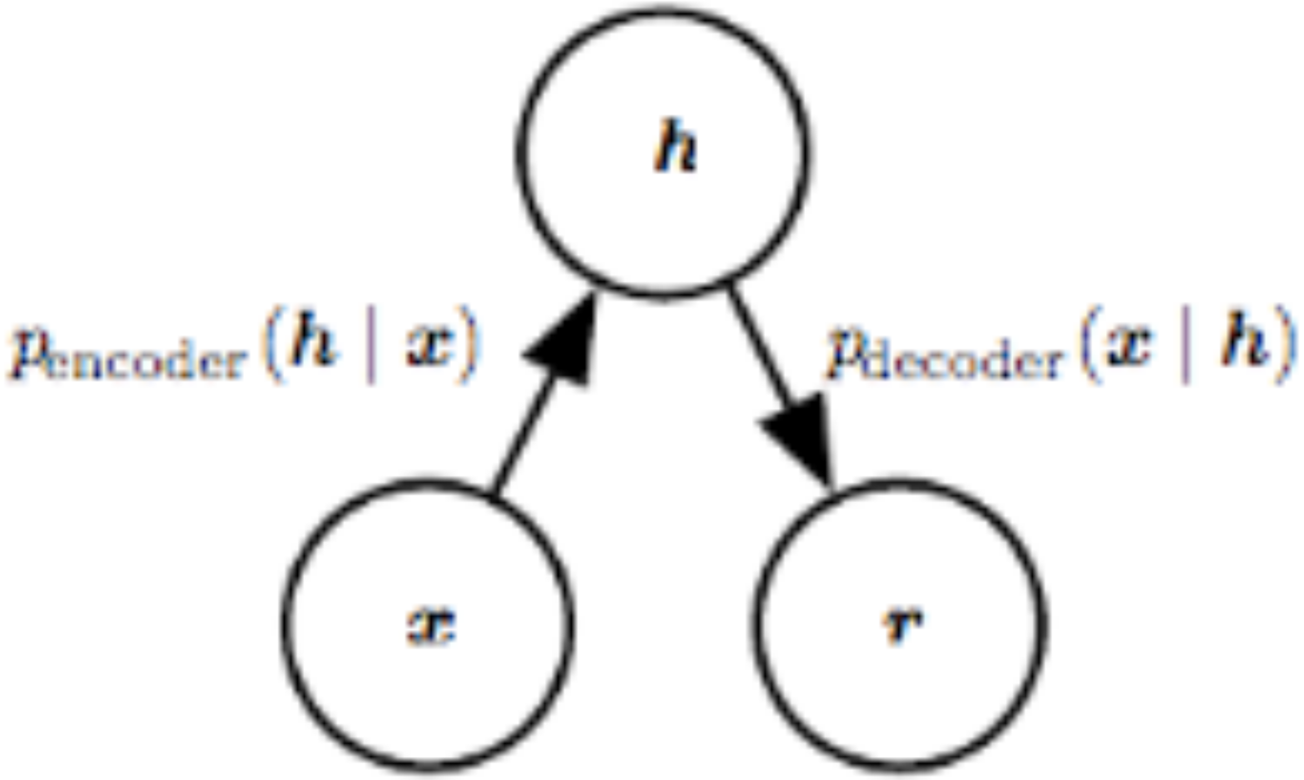
$$l(\{x_i\},\{x_j\},\{z_i\}|\theta,\lambda) = \sum_i log\, p(x_i, z_i|\lambda,\theta) + \sum_j log\, p(x_j|\lambda,\theta)$$

$$= \sum_i log\, p(z_i|\lambda)p(x_i|z,\theta) + \sum_j log\, \sum_z p(z_j|\lambda)p(x_j|z_j,\theta)$$

Here i ranges over the data points where we have labels, and j over the data points where we dont.

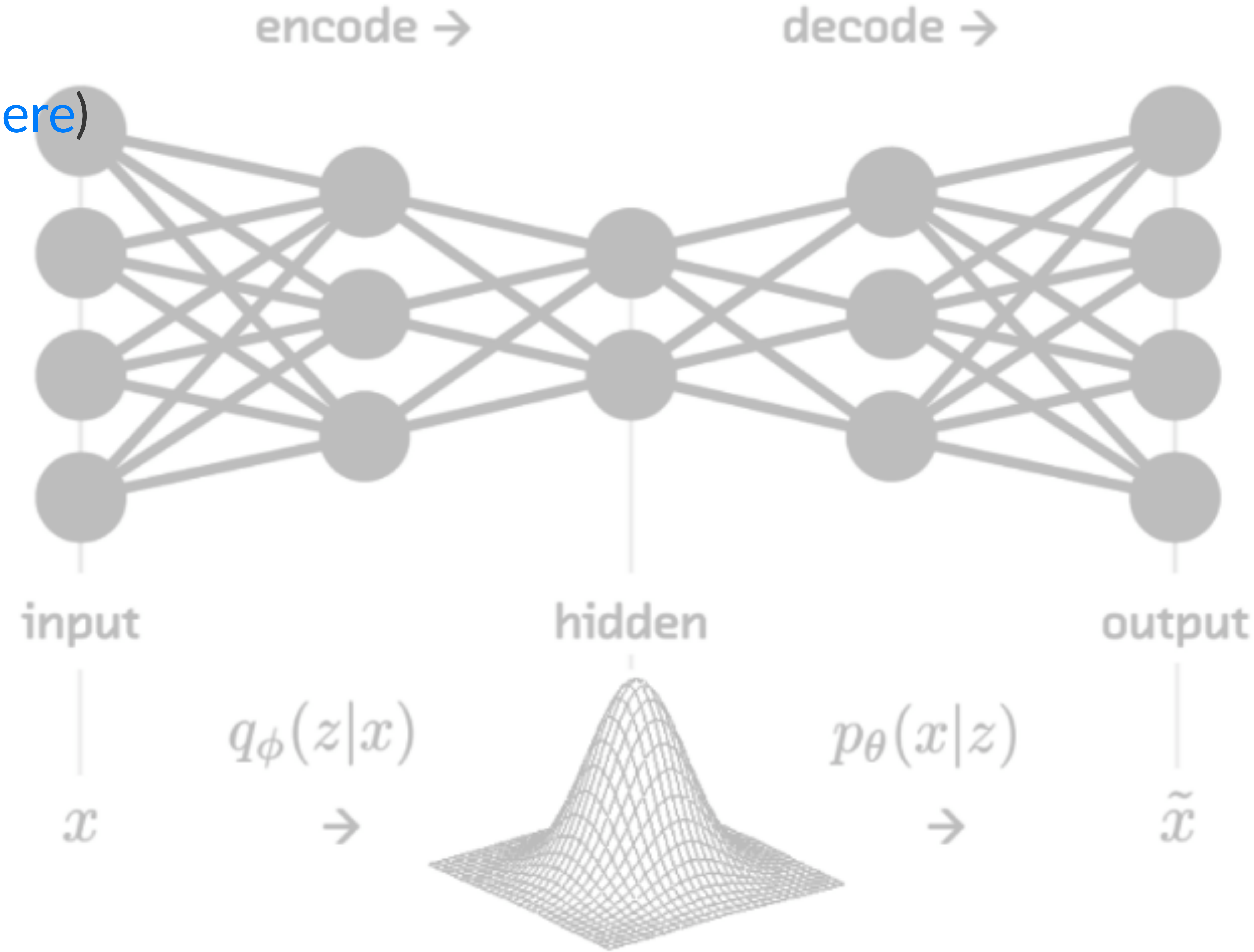
Autoencoders

- can think of an autoencoder as a way of approximately training a generative model.
- the features of the autoencoder describe the latent variables that explain the input
- can go deep!
- generalize to a stochastic autoencoder. The standard autoencoder then is a specific hidden state h or z



h
$p_{\text{encoder}}(\boldsymbol{h} \mid \boldsymbol{x})$
$p_{\text{decoder}}(\boldsymbol{x} \mid \boldsymbol{h})$
x
r

(from here)


encode →
decode →
input
hidden
output
$q_\phi(z|x)$
$p_\theta(x|z)$
x
→
→
$\tilde{x}$

ELBO GREASE

The EM algorithm

- iterative method for maximizing difficult likelihood (or posterior) problems, first introduced by Dempster, Laird, and Rubin in 1977
- Sorta like, just assign points to clusters to start with and iterate.
- Then, at each iteration, replace the augmented data by its conditional expectation (more precisely, compute the conditional expectation of the augmented or full data likelihood) given current observed data and parameter estimates. (E-step)
- Maximize the full-data likelihood (M-step).

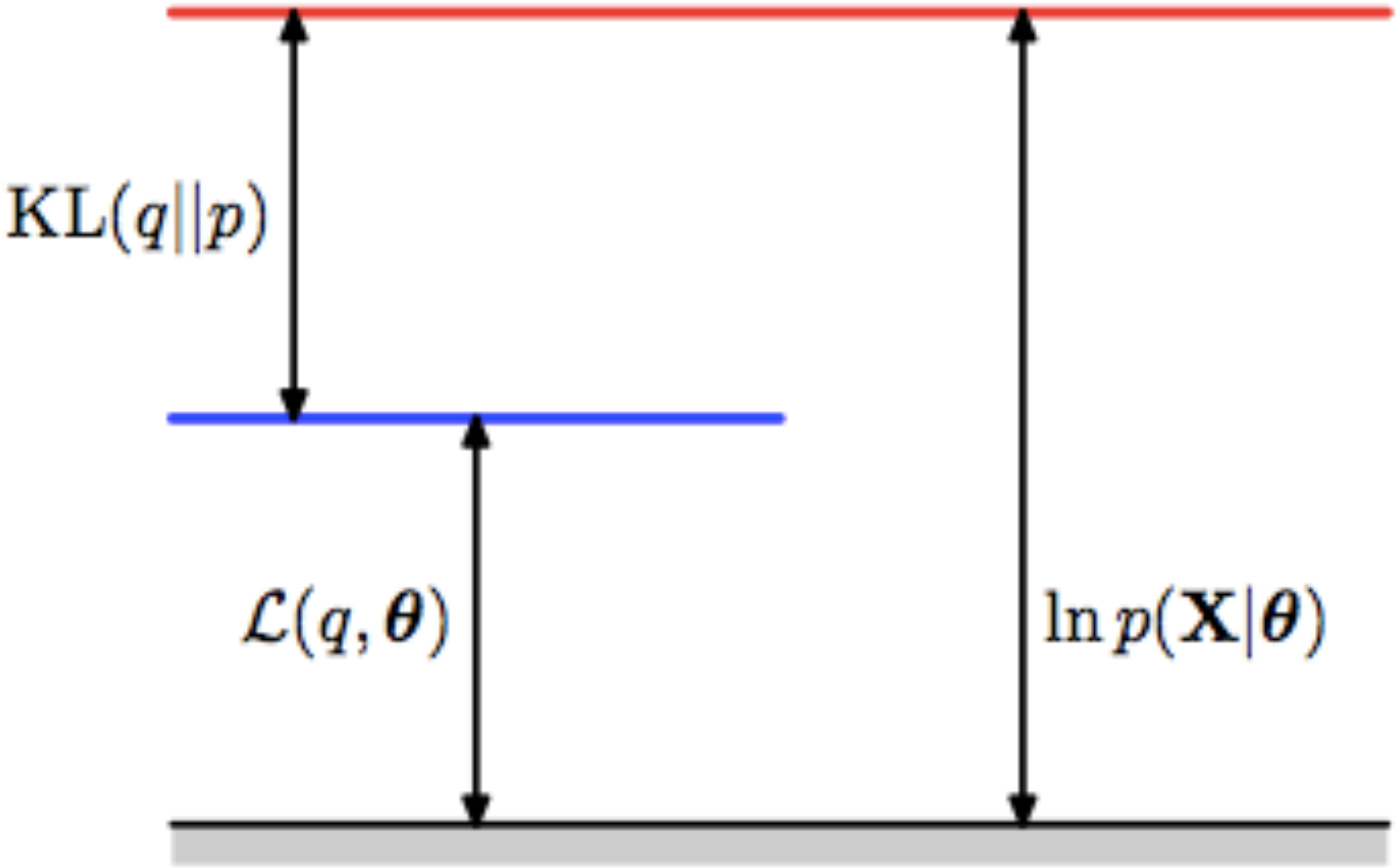
x-data likelihood

$$log\, p(x|\theta) = E_q[log \frac{p(x,z|\theta)}{q}] + D_{KL}(q,p)$$

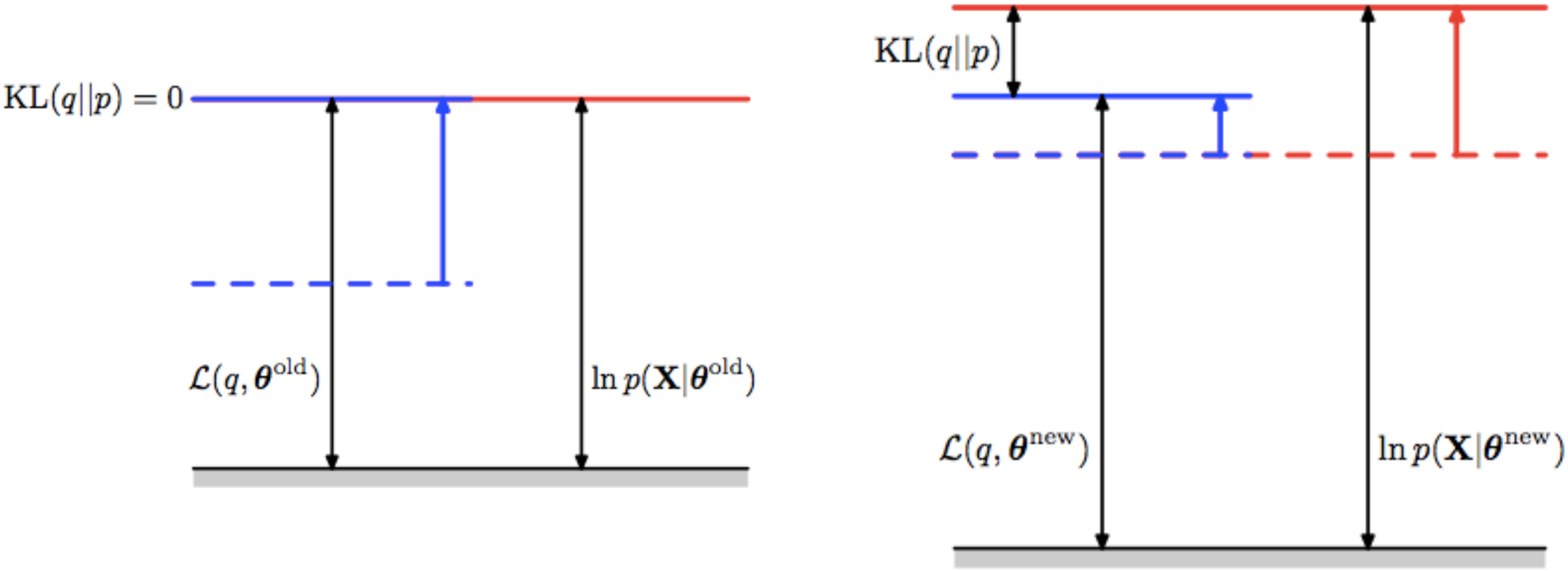
If we define the ELBO or Evidence Lower bound as:

$$\mathcal{L}(q,\theta) = E_q[log \frac{p(x,z|\theta)}{q}]$$

then $log\, p(x|\theta)$ = ELBO + KL-divergence

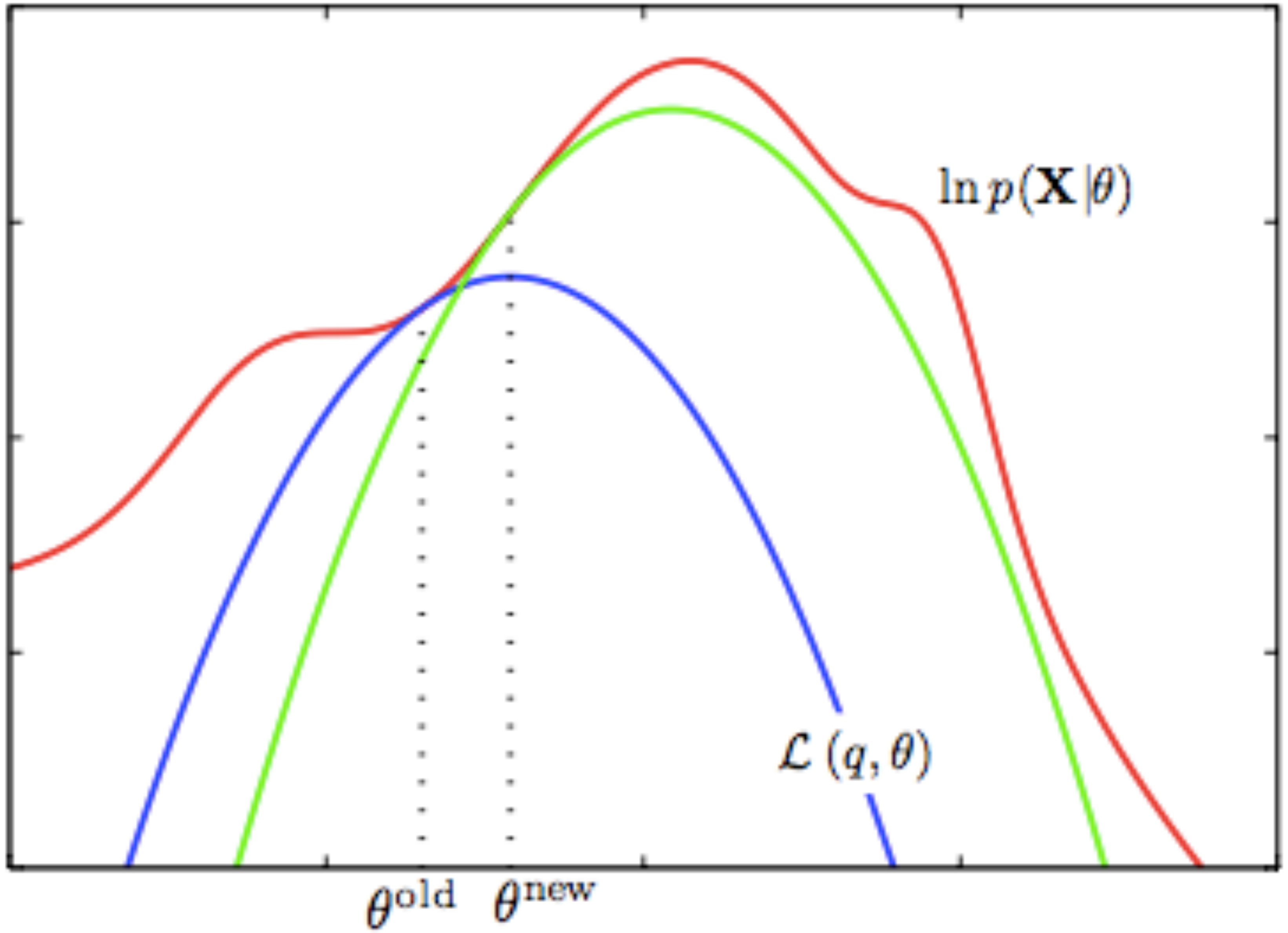

$\mathrm{KL}(q||p)$
$\mathcal{L}(q,\boldsymbol{\theta})$
$\ln p(\mathbf{X}|\boldsymbol{\theta})$

E-step and M-step


$\mathrm{KL}(q||p) = 0$
$\mathcal{L}(q, \boldsymbol{\theta}^{\text{old}})$
$\ln p(\mathbf{X}|\boldsymbol{\theta}^{\text{old}})$
$\mathrm{KL}(q||p)$
$\mathcal{L}(q, \boldsymbol{\theta}^{\text{new}})$
$\ln p(\mathbf{X}|\boldsymbol{\theta}^{\text{new}})$

Process

1. Start with $p(x|\theta)$(red curve), θ_{old}.
2. Until convergence:
   1. E-step: Evaluate $q(z, \theta_{old}) = p(z|x, \theta_{old})$ which gives rise to ELBO(θ): $\mathcal{L}(q(z, \theta_{old}), \theta)$(blue curve) whose value equals the value of $p(x|\theta)$ at θ_{old}.
   2. M-step: maximize ELBO (or Q func) wrt θ to get θ_{new}.
   3. Set $\theta_{old} = \theta_{new}$



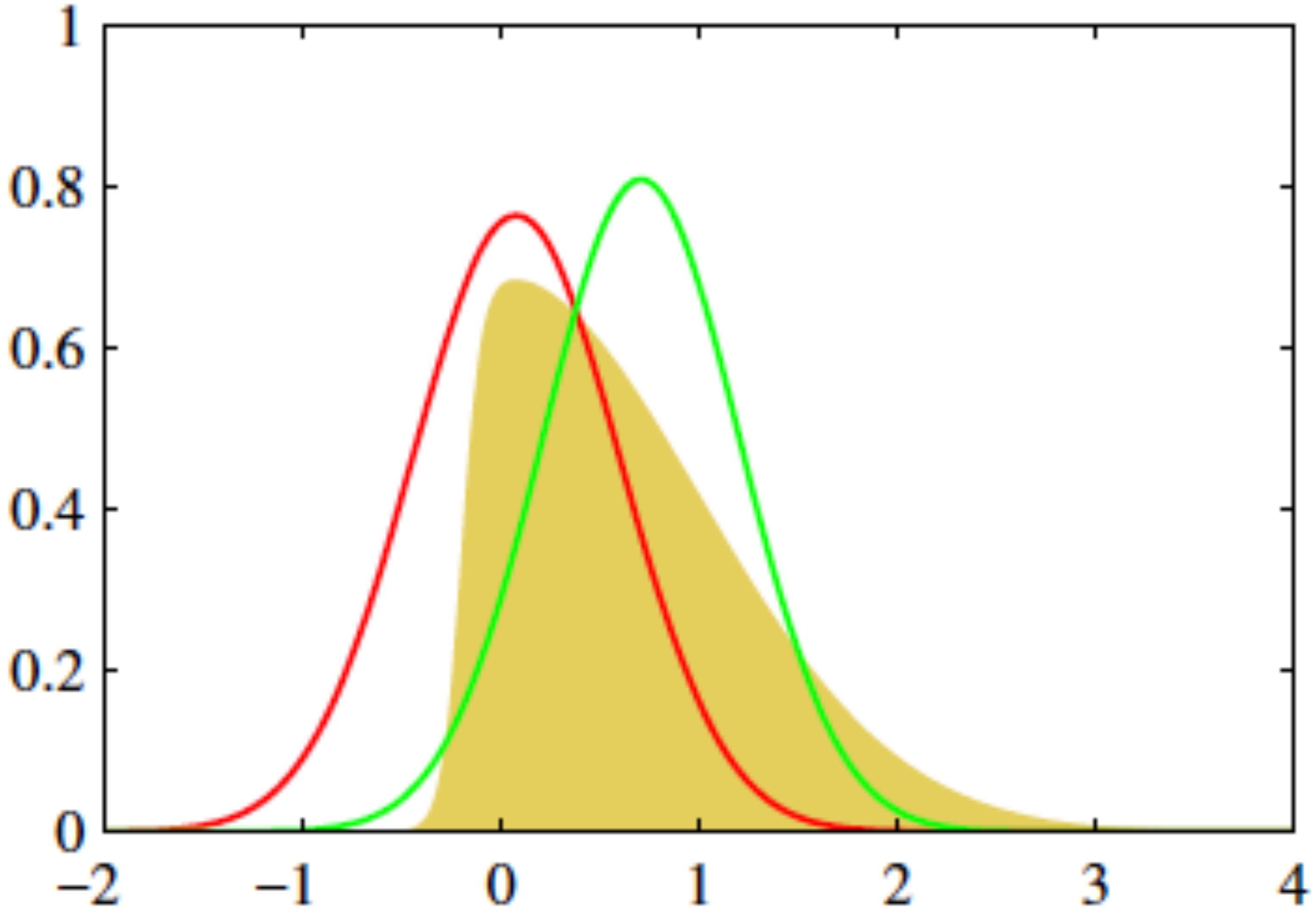
VARIATIONAL INFERENCE

Variational Inference Core Idea

z is now all parameters. Dont distinguish from θ.

Restricting to a family of approximate distributions D over z, find a member of that family that minimizes the KL divergence to the exact posterior. An optimization problem:

$$q^*(z) = \underset{q(z)\in D}{\arg\min} \quad KL(q(z)||p(z|x))$$



Basic Setup in VI

$KL + ELBO = log(p(x))$: ELBO bounds log(evidence)

$$ELBO(q) = E_q[log\,\frac{p(z,x)}{q(z)}] = E_q[log\,\frac{p(x|z)p(z)}{q(z)}] = E_q[log\,p(x|z)] + E_q[log\,\frac{p(z)}{q(z)}]$$

$$\implies ELBO(q) = E_{q(z)}[(log(p(x|z))] - KL(q(z)||p(z))$$

(likelihood-prior balance)

Mean Field: Find a q such that:

$KL + ELBO = log(p(x))$: KL minimized means ELBO maximized.

Choose a "mean-field" q such that:

$$q(z) = \prod_{j=1}^{m} q_j(z_j)$$

Each individual latent factor can take on any paramteric form corresponding to the latent variable.

ADVI

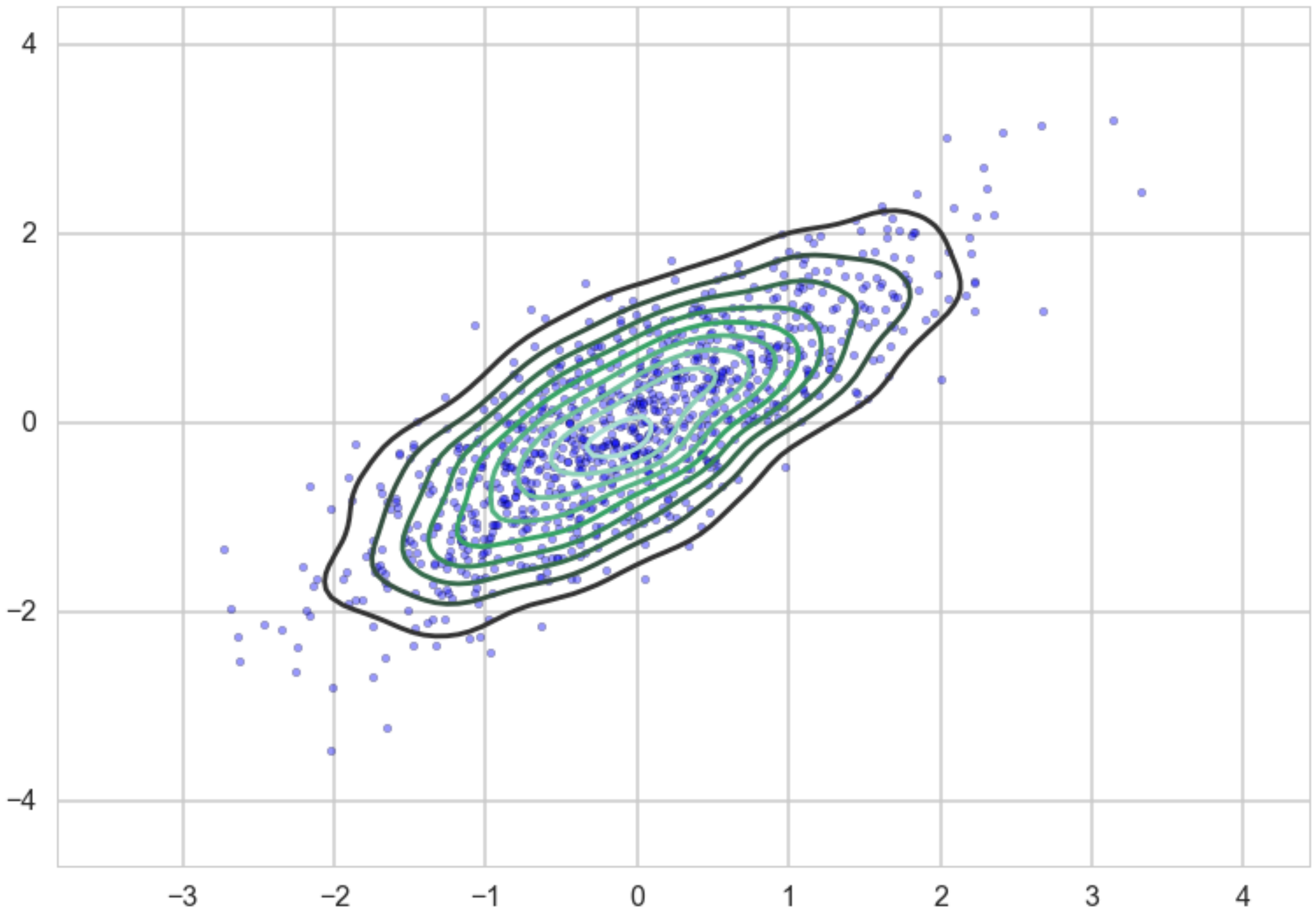
Core Idea:

- CAVI does not scale
- Use gradient based optimization, do it on less data
- do it automatically

What does ADVI do?

1. Transformation of latent parameters (**T** transform)
   - reparametrize mean field parameters to the real line
2. Standardization transform for posterior to push gradient inside expectation (**S** transform)
3. Monte-Carlo estimate of expectation
4. Hill-climb using automatic differentiation

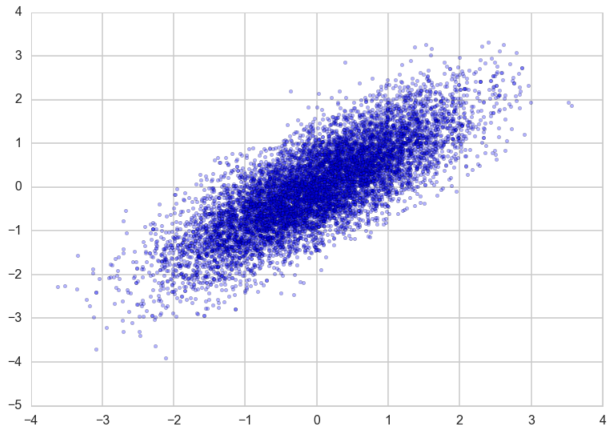
2D gaussian example



High correlation gaussian with sampler

```
cov=np.array([[0,0.8],[0.8,0]], dtype=np.float64)
data = np.random.multivariate_normal([0,0], cov, size=1000)
sns.kdeplot(data);
with pm.Model() as mdensity:
    density = pm.MvNormal('density', mu=[0,0],
    cov=tt.fill_diagonal(cov,1), shape=2)
with mdensity:
    mdtrace=pm.sample(10000)
```

Trace:



Two ideas from Yao et. al.

- pareto shape parameter k from PSIS tells you goodness of fit (see here for @junpenglao pymc3 implementation, WIP). The idea comes from the process of smoothing in LOOCV estimation
- VSBC (variational simulation based callibration) : Extends calibration from Bayesian Workflow to variational case. pymc3 experimentation by @junpenglao here, WIP

Why use VB: Deep Generative Models

- simply not possible to do inference in large models
- inference in neural networks: understanding robustness, etc
- hierarchical neural networks (perhaps on exam)
- Mixture density networks: mixture parameters are fitted using ANNs
- extension to generative semisupervised learning

Variational Autoencoder

- just as in ADVI, we want to learn an approximate "encoding posterior" $p(z|x)$
- note that we have now again gone back to thinking of z as a (possibly) deep latent variable, or "representation".

We know how to do this:

ELBO maximization

Why?

See pymc3 for e.g. for auto-encoding LDA

- variational auto-encoders algorithm which allows us to perform inference efficiently for large datasets
- use tunable and flexible encoders such as multilayer perceptrons (MLPs) as our variational distribution to approximate complex variational posterior
  -then its just ADVI with mini-batch on PyMC3 or pytorch. Can use for any posterior, example LDA, or custom for MNIST

Big Ideas

- learning is possible because there is a compressive manifold on which the data lives
- through SGD, HMC, etc we try tolearn about this manifold
- principled modeling can be done by combining known schemes such as poisson GLM with deep networks
- networks (which are just complex models) can be used at other places such as variational posteriors
- priors will regularise for us!

Interesting Times

- we progress by first predicting, and then understanding the robustness of our inference: posteriors and error bars
- MCMC/HMC, bayesian workflow, generative models, deep generative models and variational inference are at the cutting edge
- we have tried in this course to cover the basics and then be at this edge in places

What you have done and should do

- a lot of practice with lecture examples, labs, and homework
- been at the edge with your paper
- stay at the edge! Twitter is the place to be.
- follow folks like Andrew Gelman, Michael Betancourt, Jim Savage, Dan Simpson, Ian Goodfellow, Aki Vehtari, Dustin Tran, BayesGroup, Stephen Merity, Jeremy Howard, Roger Grosse, Ferenc Huszar, Alex D'Amour, Tom Wiecki, Colin Carrol, Tom Augsperger, Francios Chollet, Junpeng Lao, Richard McElreath

FIN