Lecture 16

Gibbs, Metropolis, and MH Samplers

Fully Bayesian Rat tumors

Joint Posterior:

$$p(\Theta, \alpha, \beta | Y, \{n_i\}) \propto p(\alpha, \beta) \prod_{i=1}^{70} Beta(\theta_i, \alpha, \beta) \prod_{i=1}^{70} Binom(n_i, y_i, \theta_i)$$

Need to move ALL OVER this posterior. BUT, it is not easy to sample from.

Ancestral Sampling

- sample from graph, that is from prior, then conditional prior, then data-distribution
- but not enough to sample from posterior! What we want to do is to to restrict all possible samples we got this way by:
- CONDITIONING on the data, and getting only those samples consistent with this conditioning

Basic Idea of Markov Chain Monte Carlo (MCMC)

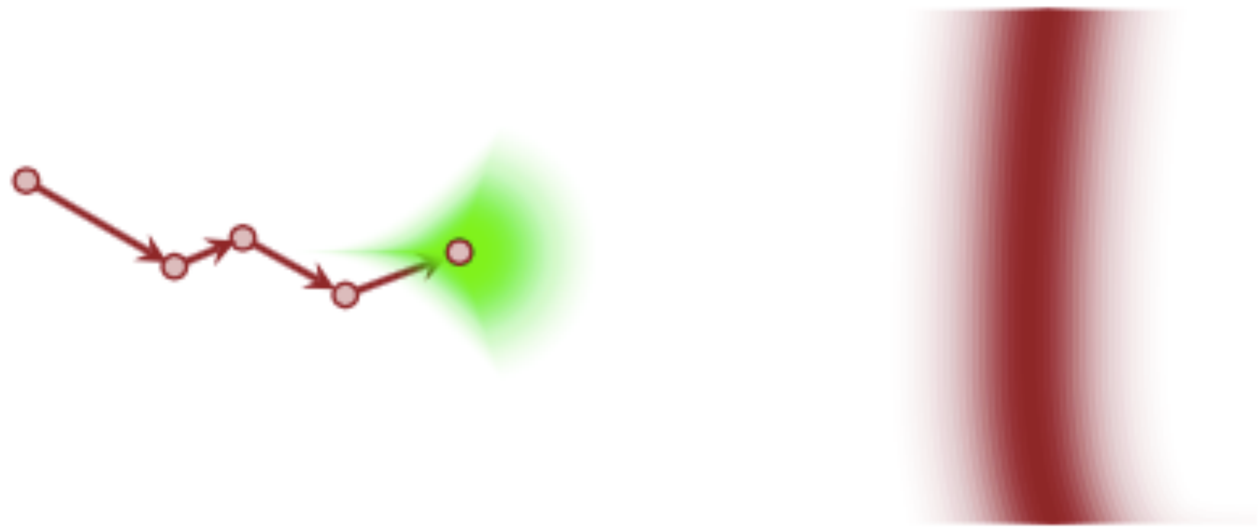
Move all over p by identifying $E = -log(p)$ with the energy of an imaginary physical system. Thus $p = exp(-E)$.

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

If the new state has lower energy, or higher probability, accept x_j.

If the new state has higher energy, accept x_j, with probability proportional to the ratio $p(x_j)/p(x_i)$ or $exp(-(E_j - E_i))$.

Intuition

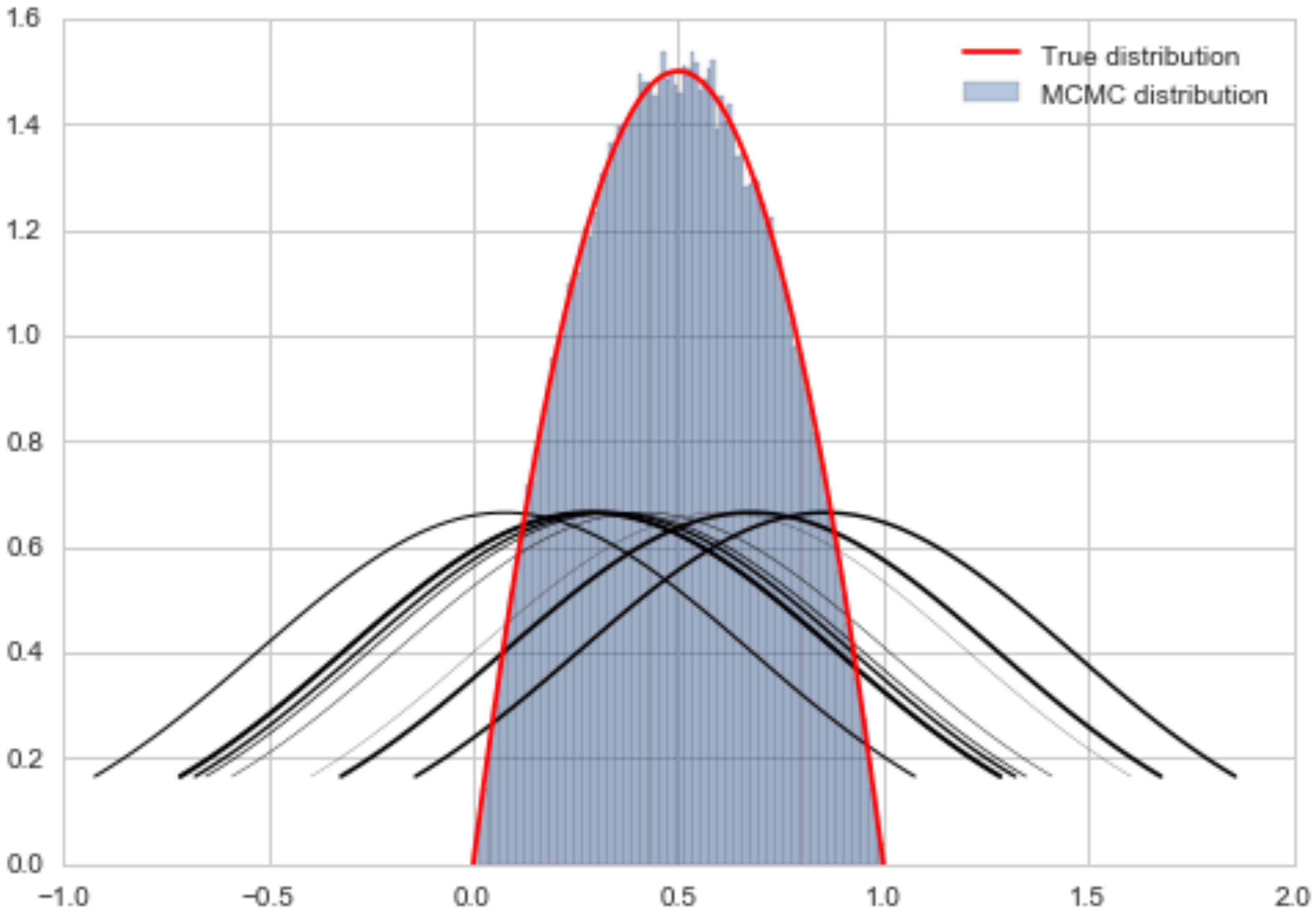


a particle approaches typical set and then gets stuck in it

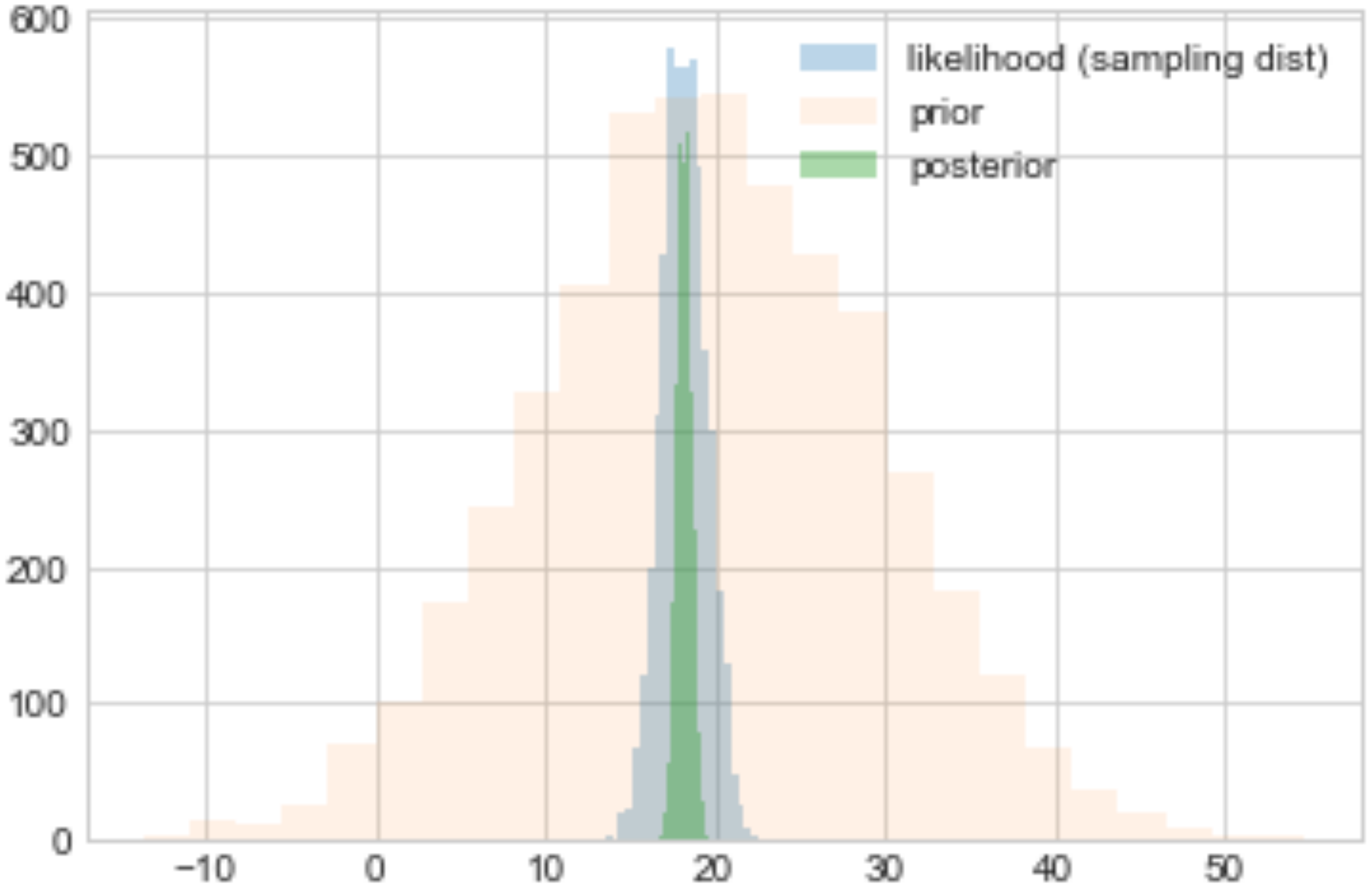
Instead of sampling p we sample q, yielding a new state, and a new proposal distribution from which to sample.

Sampling $6x(1-x)$

```python
def metropolis(p, qdraw, nsamp, xinit):
    samples=np.empty(nsamp)
    x_prev = xinit
    for i in range(nsamp):
        x_star = qdraw(x_prev)
        p_star = p(x_star)
        p_prev = p(x_prev)
        pdfratio = p_star/p_prev
        if np.random.uniform() < min(1, pdfratio):
            samples[i] = x_star
            x_prev = x_star
        else:#we always get a sample
            samples[i]= x_prev
    return samples

def prop(x):
    return np.random.normal(x, 0.6)

f = lambda x: 6*x*(1-x)
x0=np.random.uniform()
samps = metropolis(f, prop, 1000000, x0)
```



Bayesian Normal-Normal Model



```
logprior = lambda mu: norm.logpdf(mu, loc=mu_prior, scale=std_prior)
loglike = lambda mu: np.sum(norm.logpdf(Y, loc=mu, scale=np.std(Y)))
logpost = lambda mu: loglike(mu) + logprior(mu)
def prop(x, step):
    return np.random.normal(x, step)
x0=np.random.uniform()
nsamps=40000
samps, acc = metropolis(logpost, prop, 1, nsamps, x0)
```

Markov Chain

$$T(x_n|x_{n-1}, x_{n-1} \ldots, x_1) = T(x_n|x_{n-1})$$

- non IID, stochastic process
- but one step memory only
- widely applicable, first order equations

T is a transition probability, so that $\int T(x_n|x_{n-1})dx_{n-1} = 1.$

Stationarity

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

Continuous case: define T so that:

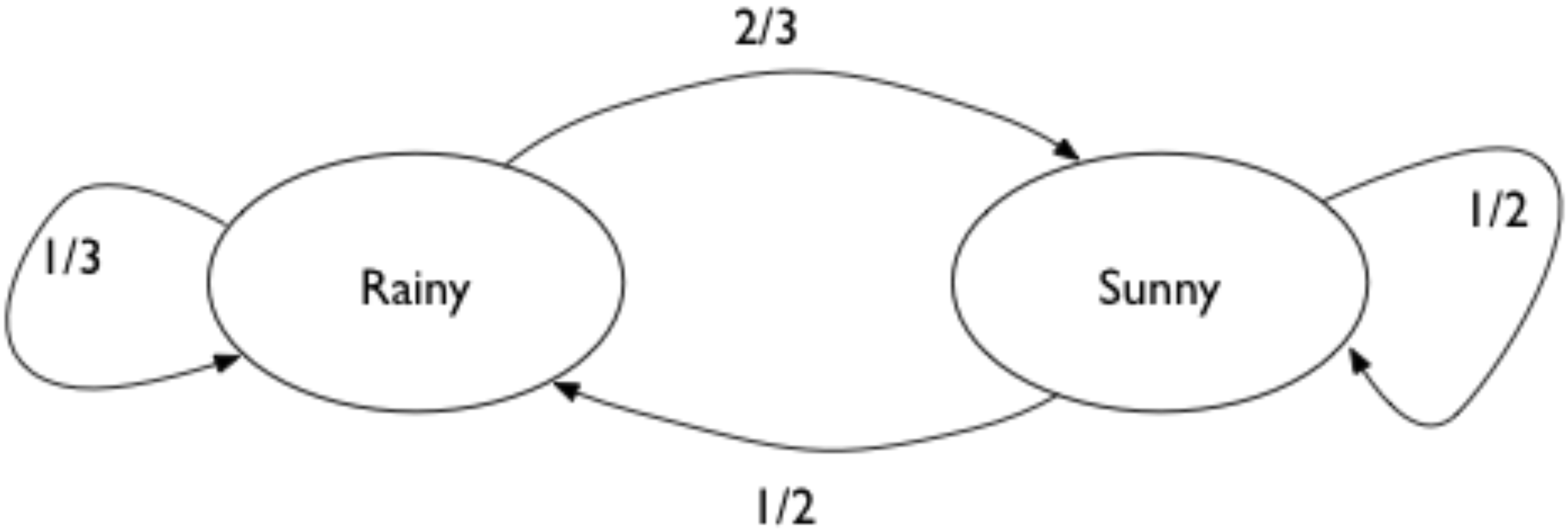
$$\int dx_i s(x_i) T(x_{i+1} | x_i) = s(x_{i+1}) \text{ then}$$

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

Jargon

- **Irreducible**: can go from anywhere to everywhere
- **Aperiodic**: no finite loops
- **Recurrent**: visited repeatedly. Harris recurrent if all states are visited infinitely as $t \to \infty$.

Rainy Sunny Markov chain



aperiodic and irreducible

Transition matrix, applied again and again

```
array([[ 0.33333333,  0.66666667],
       [ 0.5       ,  0.5       ]])
```

```
[[ 0.44444444  0.55555556]
 [ 0.41666667  0.58333333]]
-----------------
[[ 0.42592593  0.57407407]
 [ 0.43055556  0.56944444]]
-----------------
[[ 0.42901235  0.57098765]
 [ 0.42824074  0.57175926]]
-----------------
[[ 0.42849794  0.57150206]
 [ 0.42862654  0.57137346]]
-----------------
[[ 0.42858368  0.57141632]
 [ 0.42856224  0.57143776]]
```

Stationary distribution can be solved for:

Assume that it is $s = [p, 1-p]$

Then: $sT = s$

gives us

$$p \times (1/3) + (1-p) \times 1/2 = p$$

and thus $p = 3/7$

```
np.dot([0.9,0.1], tm_before): array([ 0.42858153,  0.57141847])
```

Detailed balance is enough for stationarity

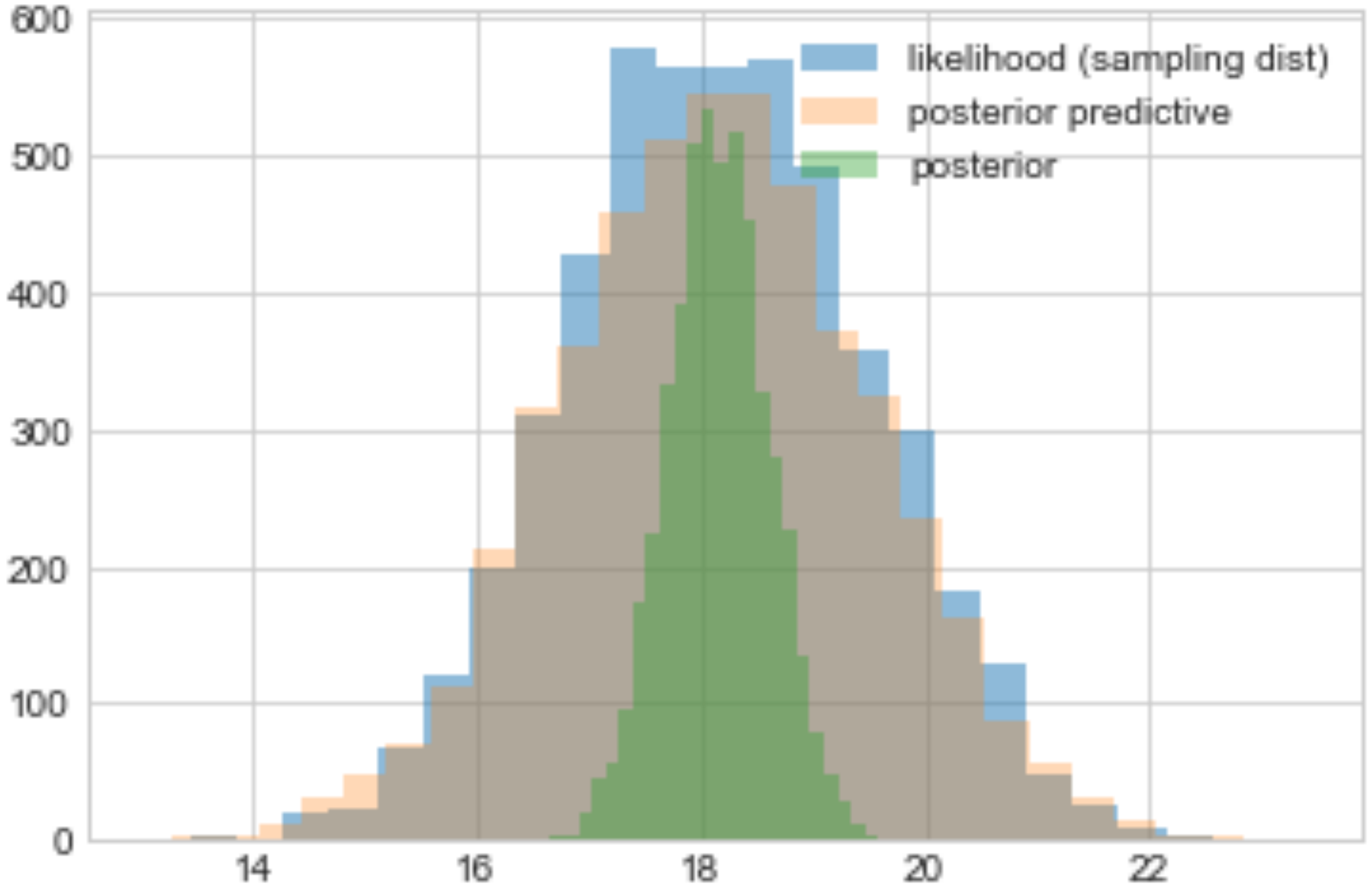
$$s(x)T(y|x) = s(y)T(x|y)$$

If one sums both sides over x

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

Posterior predictive from sampling

- first draw the thetas from the posterior
- then draw y's from the likelihood
- and histogram the likelihood
- these are draws from joint y, θ



Gibbs Sampling

What did Gibbs do?

He determined the energy states of gases at equilibrium by cycling through all the particles, drawing from each one of them conditionally given the energy levels of the others, taking the time average.

Geman and Geman used this idea to denoise images.

The idea of Gibbs

$$f(x) = \int f(x,y)dy = \int f(x|y)f(y)dy = \int dy f(x|y) \int dx' f(y|x')f(x')$$

Thus: $f(x) = \int h(x,x')f(x')dx'$ integral fixed point equation

where $h(x,x') = \int dy f(x|y)f(y|x')$.

Iterative scheme in which the "transition kernel" $h(x, x')$ is used to create a proposal for metropolis-hastings moves:

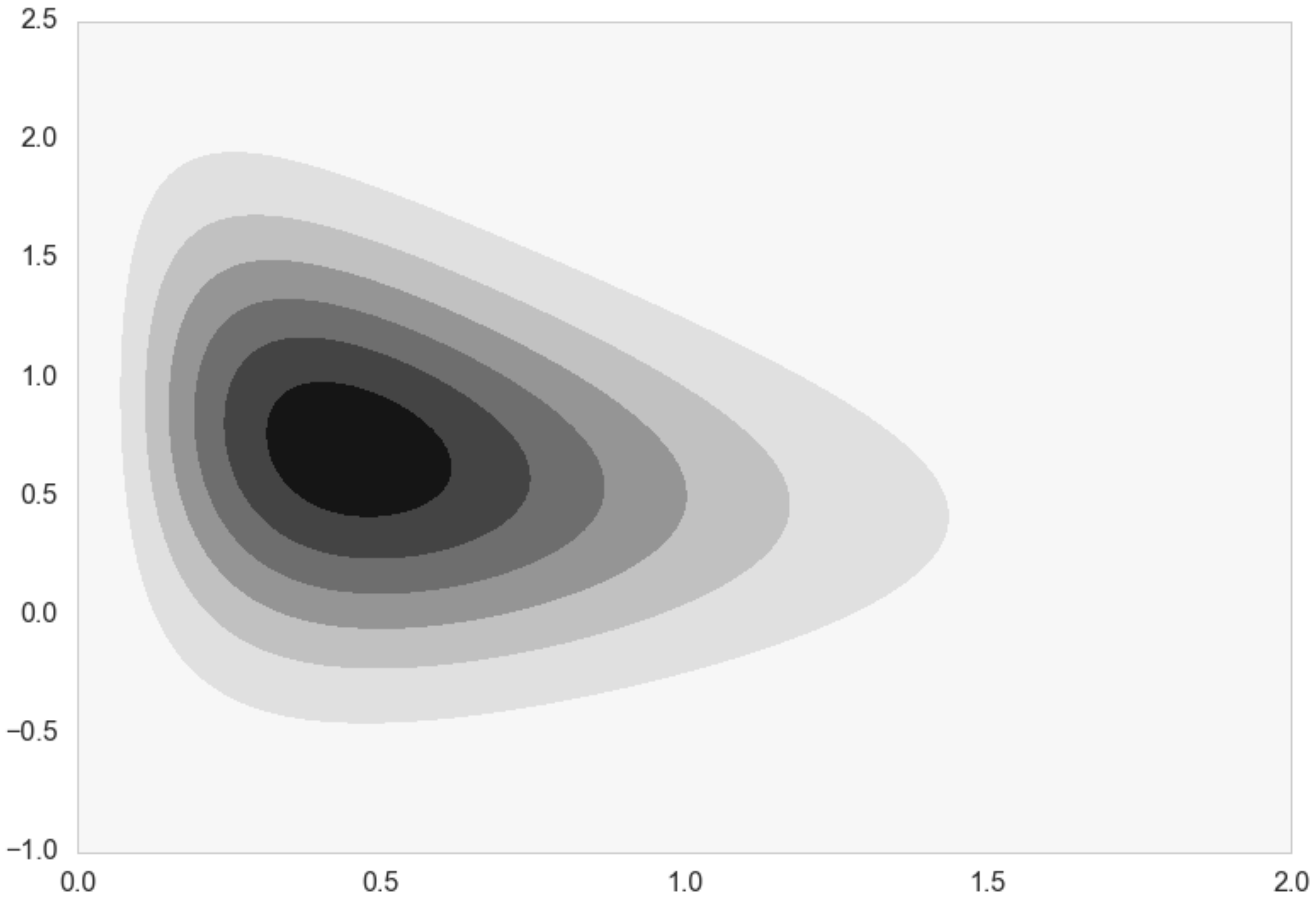
$$f(x_t) = \int h(x_t, x_{t-1}) f(x_{t-1}) dx_{t-1},\text{ a Stationary distribution.}$$

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

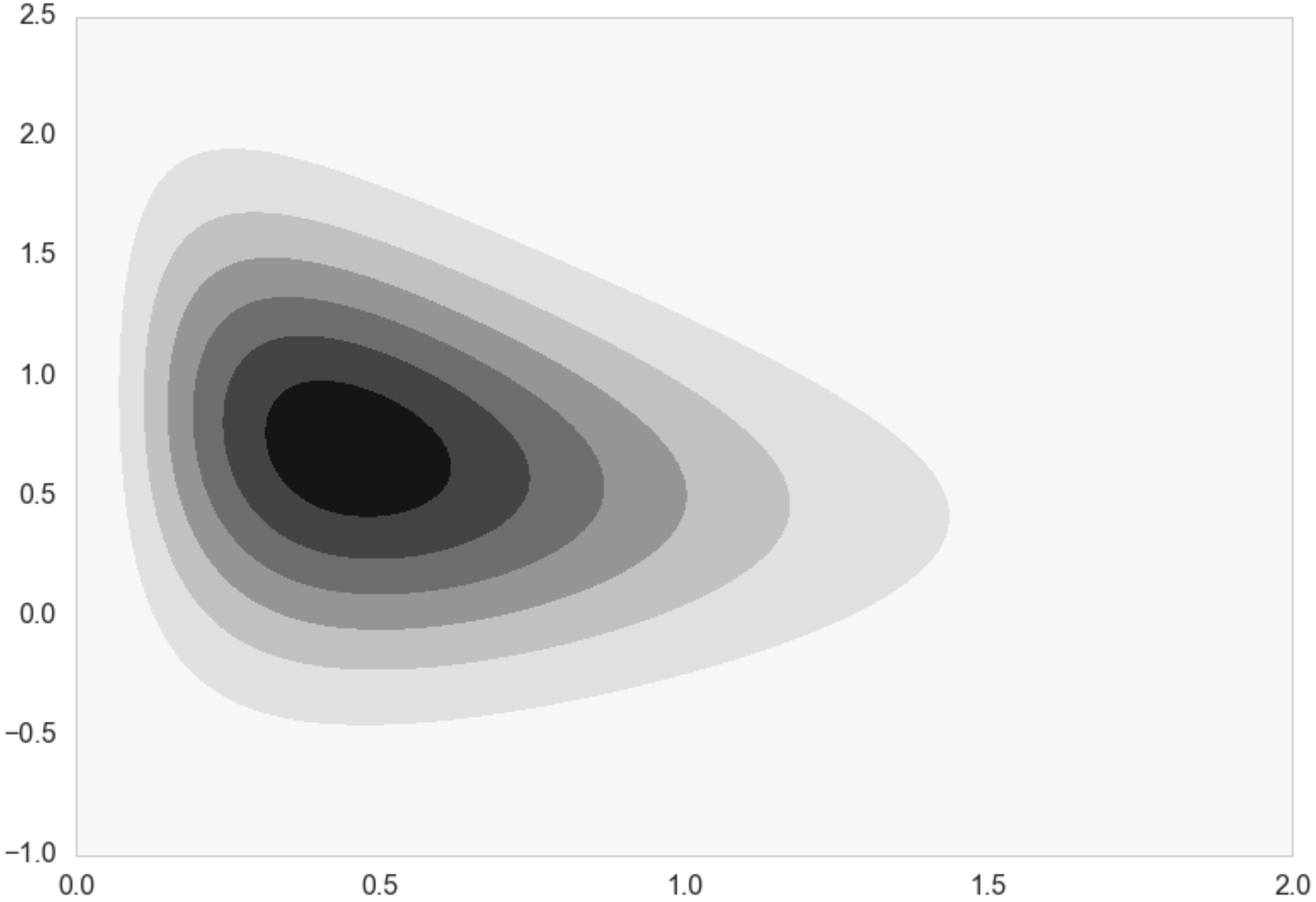
Can sample x marginal and $y|x$ so can sample the joint x, y.

Example

Sample from $f(x, y) = x^2 \exp[-xy^2 - y^2 + 2y - 4x]$



Example:



$$f(x,y) = x^2 \exp[-xy^2 - y^2 + 2y - 4x]$$

$$= x^2 \exp[-x(y^2+4)]\exp[-y^2+2y]$$

$$= g(y)\mathrm{Gamma}(x, 3, y^2+4)$$

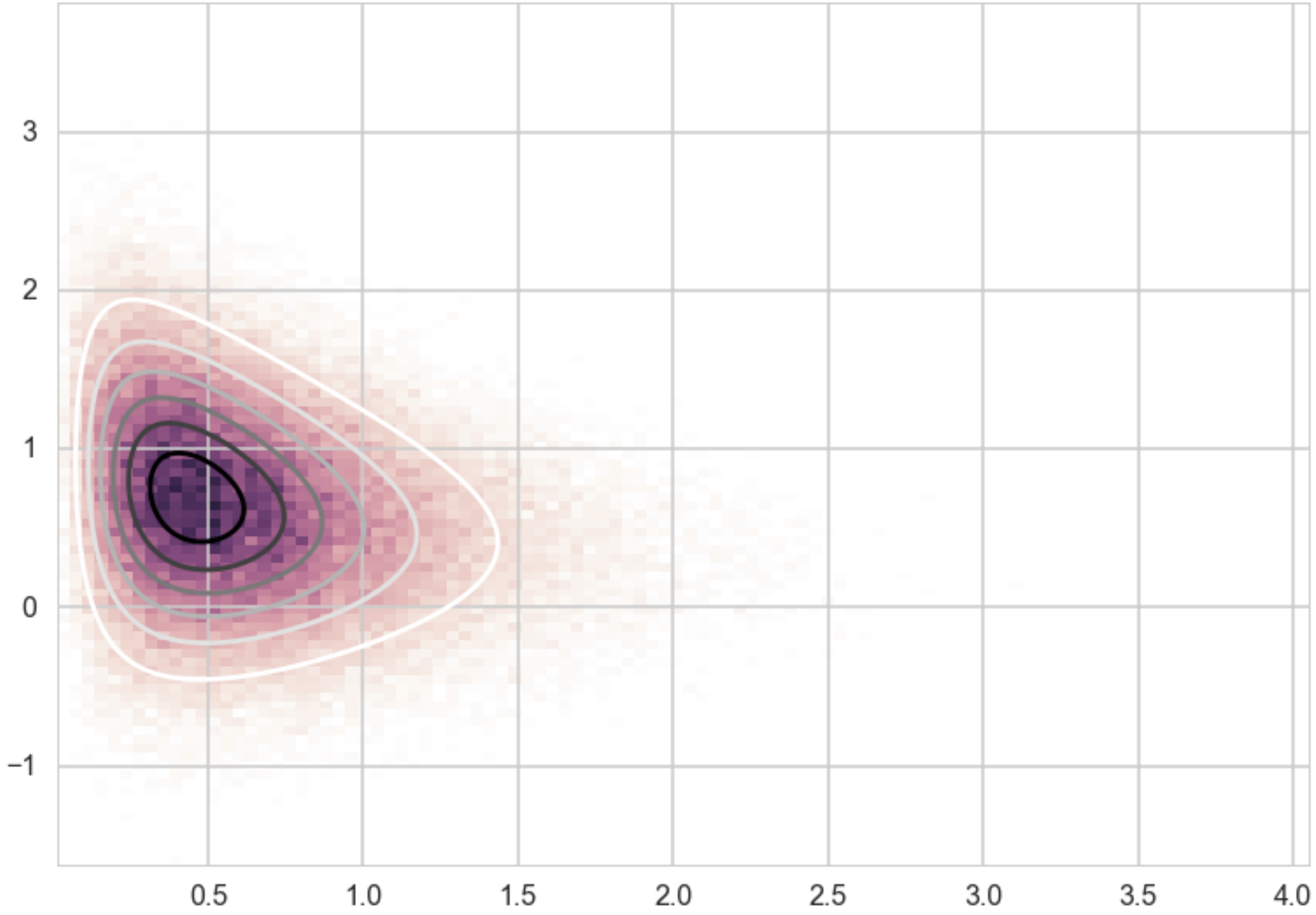
$$\implies f(x|y) \sim \mathrm{Gamma}(3, y^2+4)$$

$$f(x,y) = x^2 \exp[-y^2(1+x) + 2y]\exp[-4x]$$

$$\implies f(y|x) \sim N(\frac{1}{1+x}, \frac{1}{\sqrt{(2(1+x))}})$$

Sampler

```python
def xcond(y):
    return gamma.rvs(3, scale=1/(y*y + 4))
def ycond(x):
    return norm.rvs(1/(1+x), scale=1.0/np.sqrt(2*(x+1)))
def gibbs(xgiveny_sample, ygivenx_sample, N, start = [0,0]):
    x=start[0]
    y=start[1]
    samples=np.zeros((N+1, 2))
    samples[0,0]=x
    samples[0,1]=y
    for i in range(1,N,2):
        x=xgiveny_sample(y)
        samples[i,0]=x
        samples[i, 1]=y
        ######################
        y=ygivenx_sample(x)
        samples[i+1,0]=x
        samples[i+1,1]=y
    return samples
out=gibbs(xcond, ycond, 100000)
```



Ergodicity and Stationarity

- These are not the same concept
- detailed balance implies stationarity. Needs irreducibility.
- aperiodic, irreducible, harris recurrent markov chain $\implies$ ergodic
- what is ergodic?

Ergodicity

- Aperiodic, irreducible, positive Harris recurrent markov chains are ergodic
- i.e., in the limit of infinite (many) steps, the marginal distribution of the chain is the same. This means that if we take largely spaced about (some thinning T) samples from a stationary markov chain (after burnin B), we can draw independent samples.

- “Ergodic” law of large numbers:

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

- equivalent, for very large N:

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

- the jury is out on thinning. Most dont think one needs it
- you can get a similar central limit theorem as well

Sketch of proof (here and here for details)

- by Perron-Frobenius theorem, irreducible, aperiodic stochastic matrices (rows sum to 1 with non-negative elements) have one eigenvalue $\lambda_0 = 1$ and positive eigenvector $e_0 > 0$. All other eigenvalues have absolute value less than 1.
- $p^{(t)} = T^n p^{(0)}$ where $p^{(0)} = \sum_i \alpha_i e_i$
- Then $p^(t) = \sum_i \alpha_i \lambda_i^n e_i = \alpha_0 e_0 = e_0$

Metropolis

- probability increases, accept. decreases, accept some of the time.
- get aperiodic, irreducible, harris recurrent markov chain $\implies$ ergodic but takes a while to reach the **stationary distribution**

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

- arrange transition matrix(kernel) to get desired stationary distribution

Transition matrix for Metropolis:

$T(x_i|x_{i-1}) = q(x_i|x_{i-1})\,A(x_i,x_{i-1}) + \delta(x_{i-1} - x_i)r(x_{i-1})$ where

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

is the Metropolis acceptance probability and

$r(x_i) = \int dy q(y|x_i)(1 - A(y, x_i))$ is the rejection term.

Papers

Milestone 1: Getting Started

Due Date: Nov 10th

Deliverables: Pick a Paper, form groups

Milestone 2: Stay on Track

Due Date: Nov 10th thru 22nd.

EARLIER THE BETTER

Deliverables: Present your own short summary of the paper, add your plan of attack for creating the tutorial, and discuss it with and gain approval from at least one member of the teaching staff

Milestone 3: Turn in the final product

Due Date: December 10th

Deliverable: Complete Final Product Tutorial in a notebook

Which papers and how to work

Link for Papers: Papers

You can propose your own.

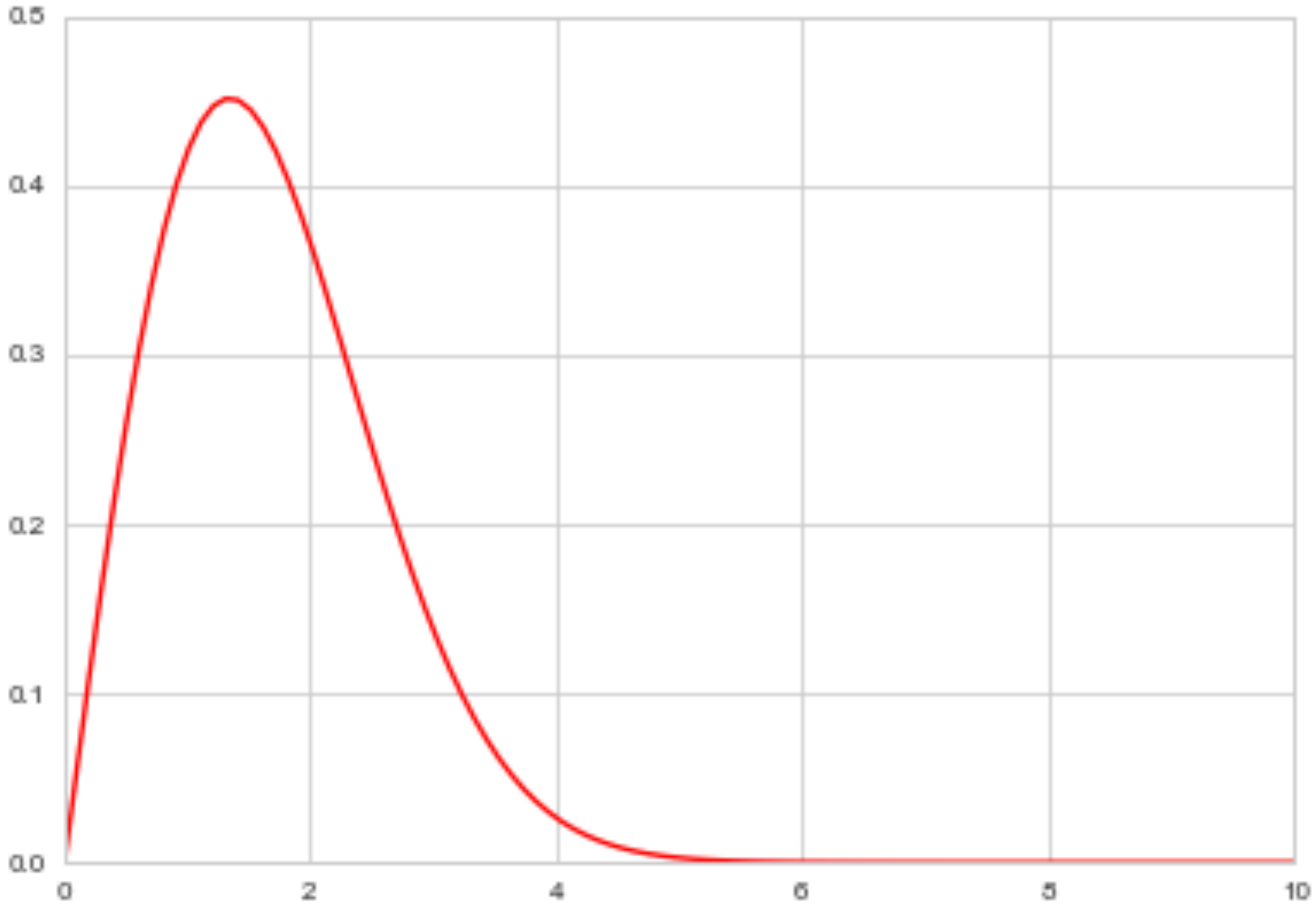
Work in teams of 3-4.

Metropolis-Hastings

- want to handle distributions with limited support
- proposal like normal leads to a lot of wasteful comparisons
- building in rejection breaks symmetry or proposal, the distribution needs to be normalized by some part of cdf.
- you might want to sample from a asymmetric distribution which matches targets support

Metropolis-Hastings

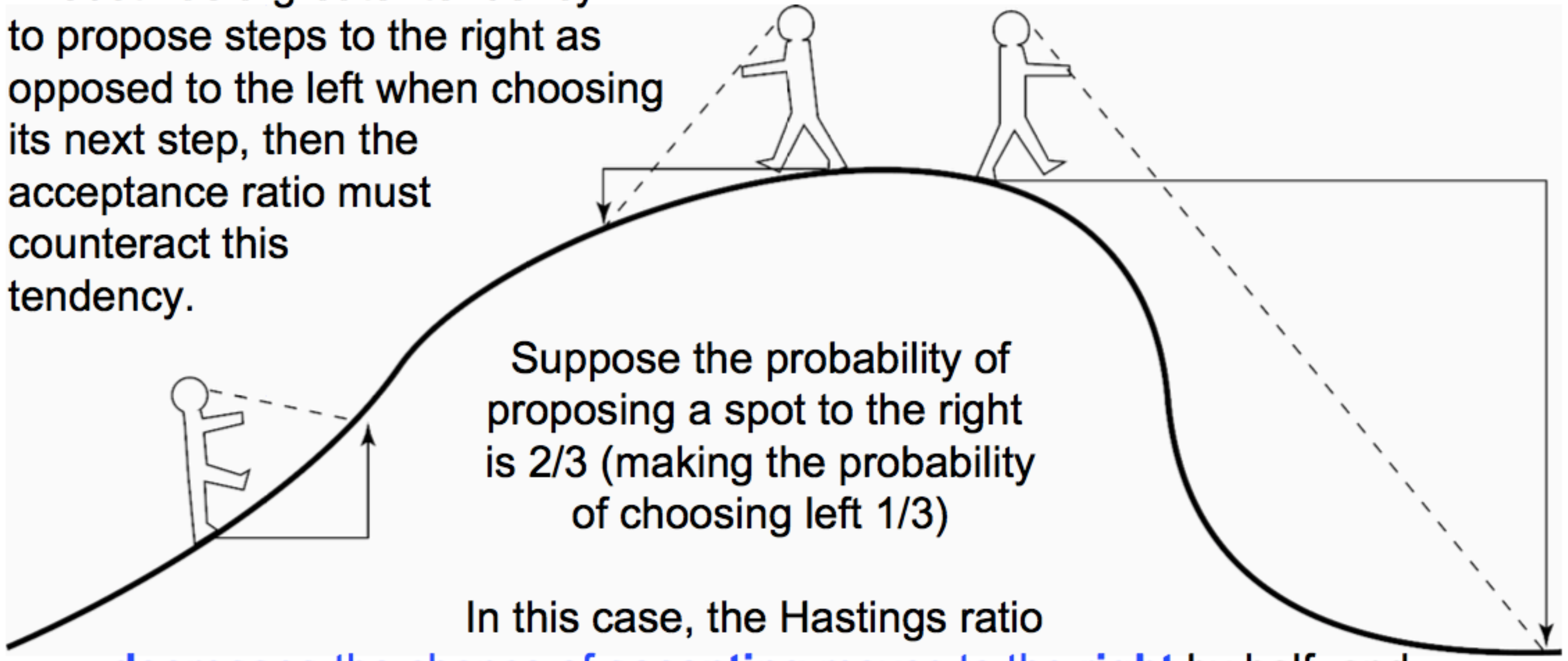
```
def metropolis_hastings(p,q, qdraw, nsamp, xinit):
    samples=np.empty(nsamp)
    x_prev = xinit
    for i in range(nsamp):
        x_star = qdraw(x_prev)
        p_star = p(x_star)
        p_prev = p(x_prev)
        pdfratio = p_star/p_prev
        proposalratio = q(x_prev, x_star)/q(x_star, x_prev)
        if np.random.uniform() < min(1, pdfratio*proposalratio):
            samples[i] = x_star
            x_prev = x_star
        else:#we always get a sample
            samples[i]= x_prev

    return samples
```



Acceptance is now

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

- 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)

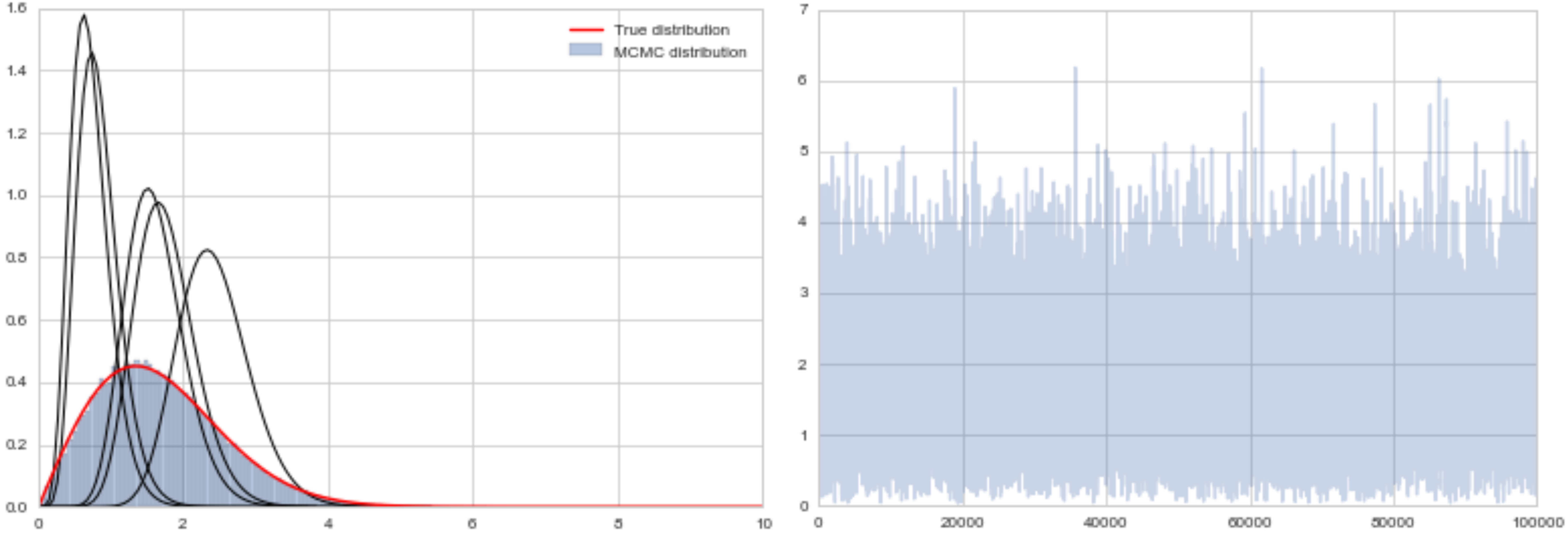
In this case, the Hastings ratio **decreases** the chance of **accepting** moves to the **right** by half, and **increases** the chance of **accepting** moves to the **left** (by a factor of 2), thus **exactly compensating** for the asymmetry in the proposal distribution.

(from Paul Lewis)

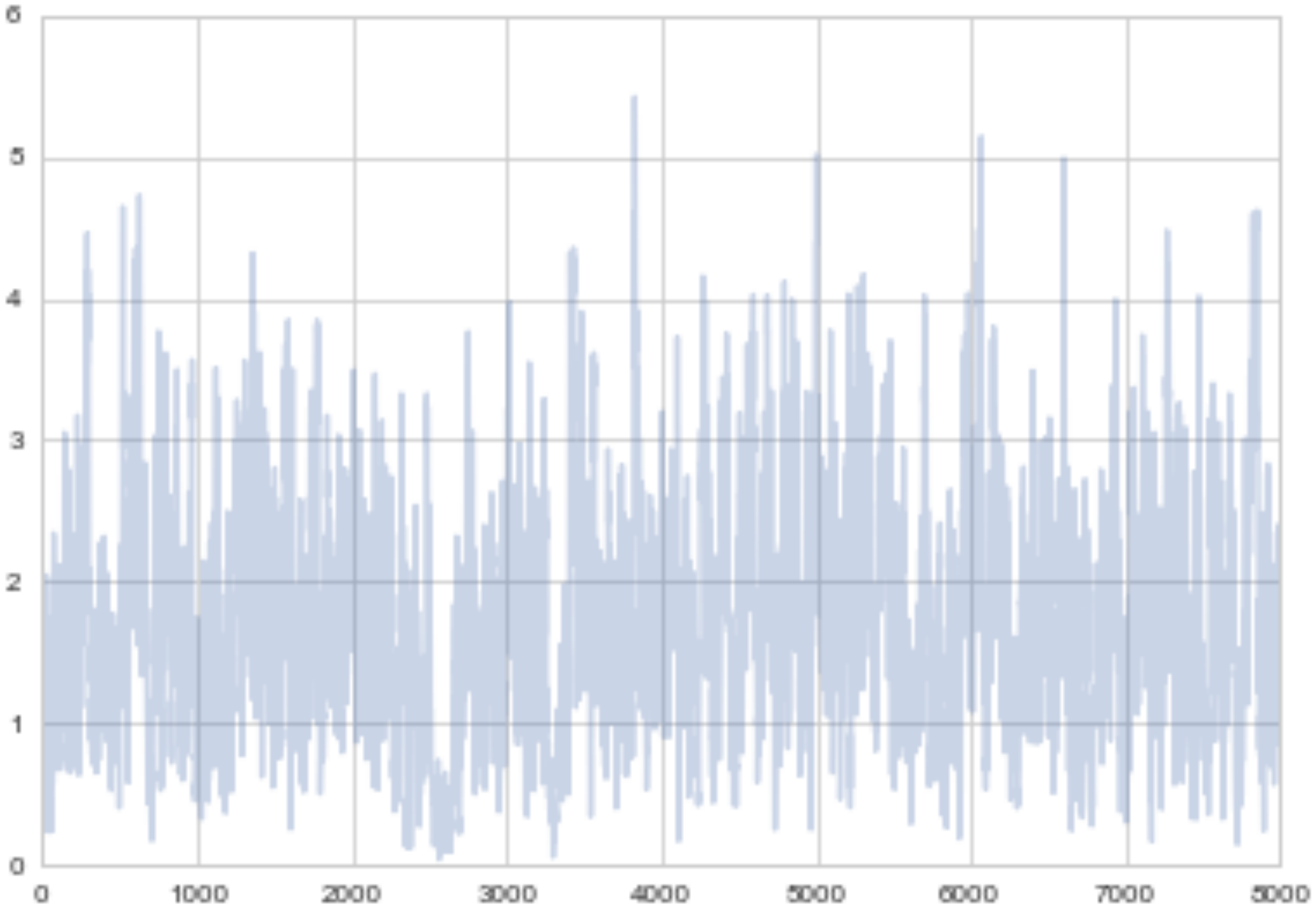
Choice of Proposal

- Our Weibull is: $0.554xe^{-(x/1.9)^2}$
- A rule of thumb for choosing proposal distributions is to parametrize them in terms of their mean and variance/precision since that provides a notion of "centeredness" which we can use for our proposals
- Use a Gamma Distribution with parametrization $Gamma(x\tau, 1/\tau)$ in the shape-scale argument setup.

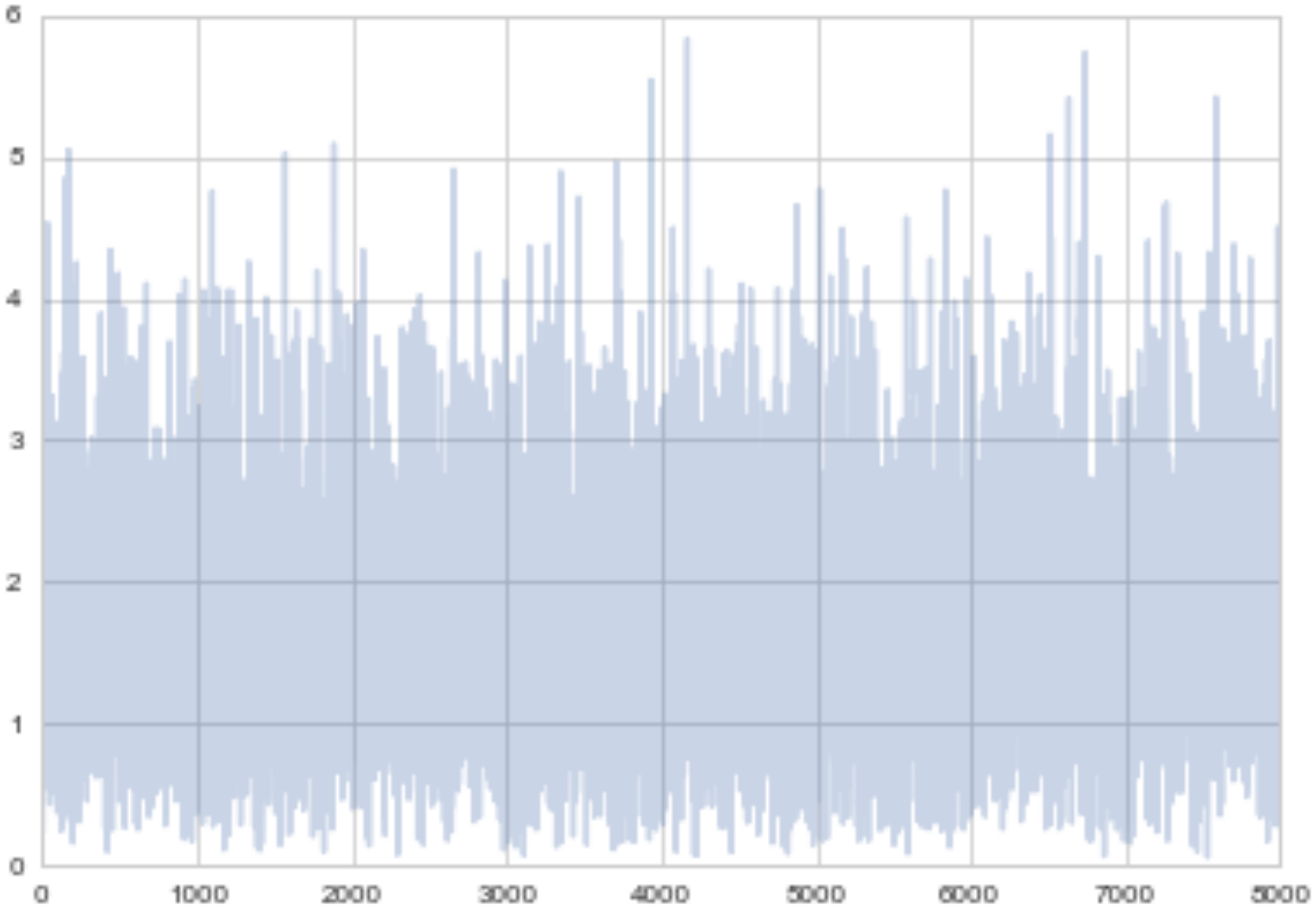
Gamma-Weibull with traceplot



Traceplot after burnin but without thinning



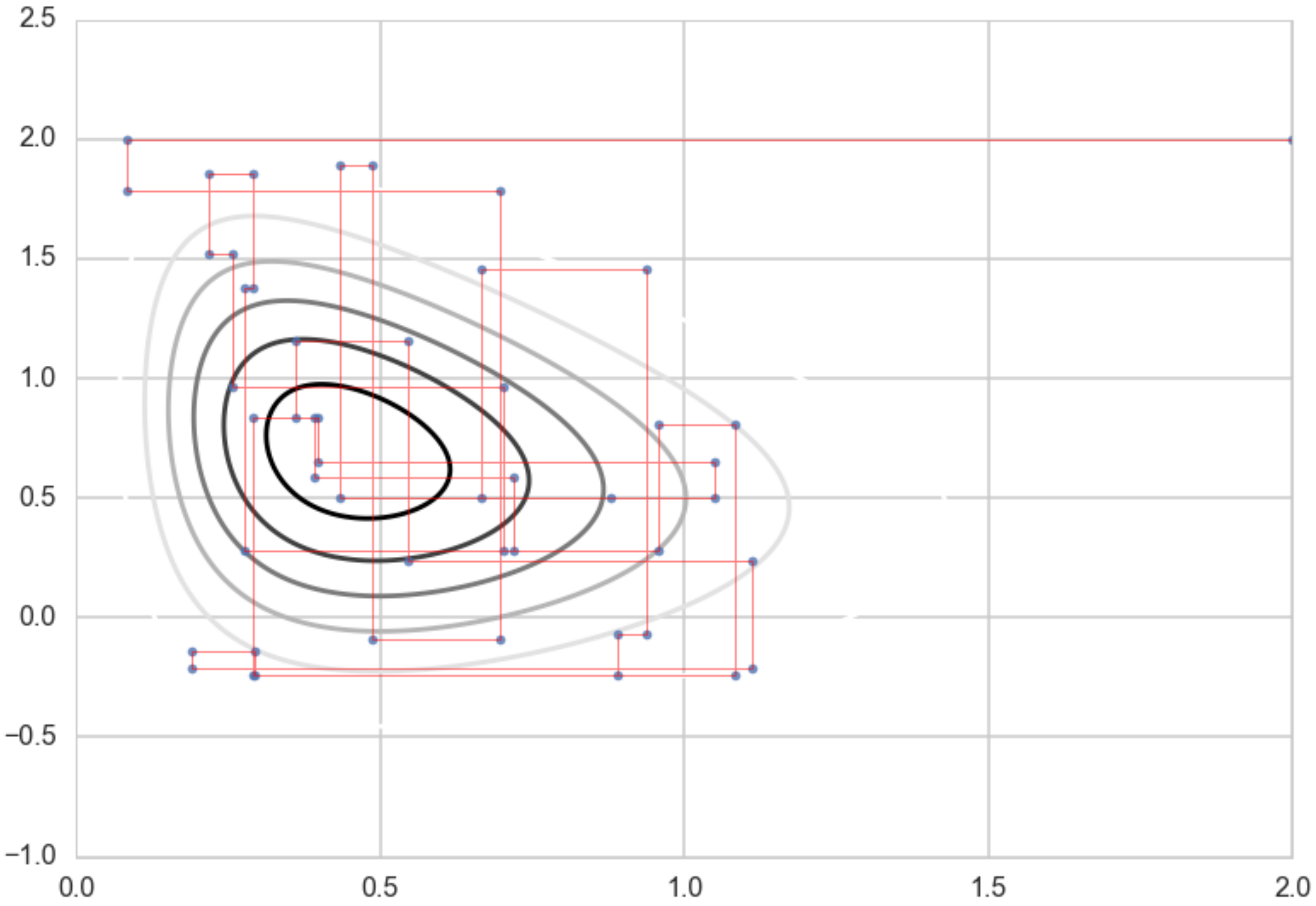
Traceplot after burning and thinning



Is thinning needed?

- jury is out but current thought is no
- does reduce space requirements and remove autocorrelation
- but removing autocorrelation is strictly not needed by ergodicity
- but how much burnin do we need? And how many effective samples?
- soon...

More about gibbs



- easiest is to know how to sample directly from conditionals: **no need for locality**
- moves one component (or one block) at a time
- all is not lost if thats not the case: can use a MH-step once stationarity has been reached
- this makes gibbs a very general idea

Fully Bayesian Rat tumors

Joint Posterior:

$$p(\Theta, \alpha, \beta | Y, \{n_i\}) \propto p(\alpha, \beta) \prod_{i=1}^{70} Beta(\theta_i, \alpha, \beta) \prod_{i=1}^{70} Binom(n_i, y_i, \theta_i)$$

Conditionals:

$$p(\theta_i | y_i, n_i, \alpha, \beta) = Beta(\alpha + y_i, \beta + n_i - y_i)$$

More Conditionals

$$p(\alpha|Y, \Theta, \beta) \propto p(\alpha, \beta) \left(\frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)} \right)^N \prod_{i=1}^{N} \theta_i^{\alpha}$$

$$P(\beta|Y, \Theta, \alpha) \propto p(\alpha, \beta) \left(\frac{\Gamma(\alpha + \beta)}{\Gamma(\beta)} \right)^N \prod_{i=1}^{N} (1 - \theta_i)^{\beta}$$

These depend on Y and $\{n\}$ via the θ's

Sampling (sampler done in lab)

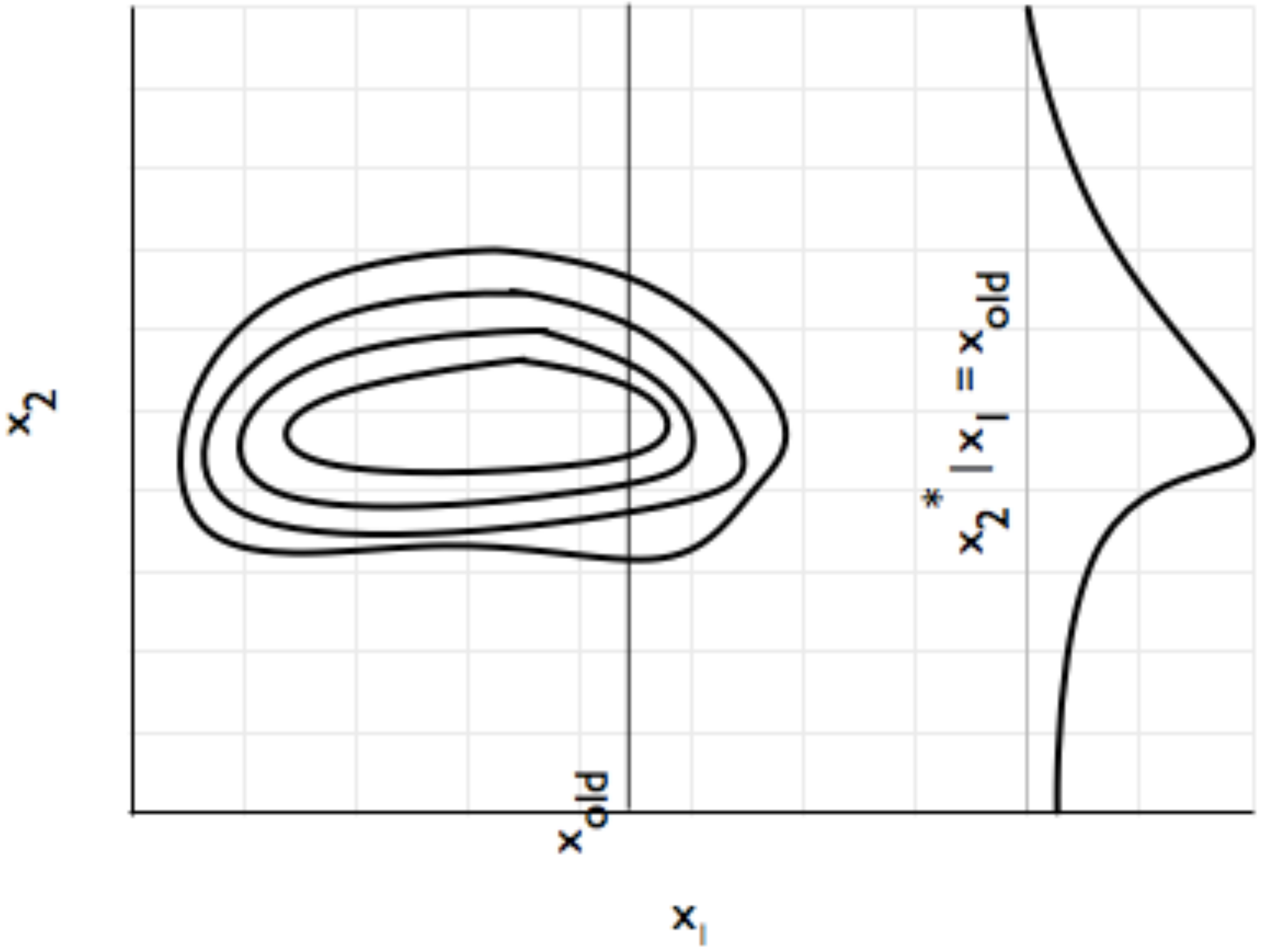
- Fix α and β, we have a Gibbs step for all of the θ_is
- For α and β, everything else fixed, use stationary metropolis step, as conditionals are not isolatable to simply sampled distributions
- when we sample for α, we will propose a new value using a normal proposal, while holding all the θs and β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

More Gibbs Theory

The transition kernel corresponds to this proposal:

$$q_k(x^*|x^i) = \begin{cases} p(x^*_k|x^i_{-k}) & for\ x^*_{-k} = x^i_{-k}, \\ 0 & otherwise \end{cases}$$

where x^i_k is the kth component (or block) of x at ith step, while x^i_{-k} is all other components of x at the same step



Gibbs=MH with no rejection

$$A = min(1, \frac{p(x^*)}{p(x^i)} \frac{q_k(x^i|x^*)}{q_k(x^*|x^i)})$$

$$p(x^*) = p(x^*{}_{-k}, x^*_k) = p(x^*_k|x^*{}_{-k})p(x^*{}_{-k})$$

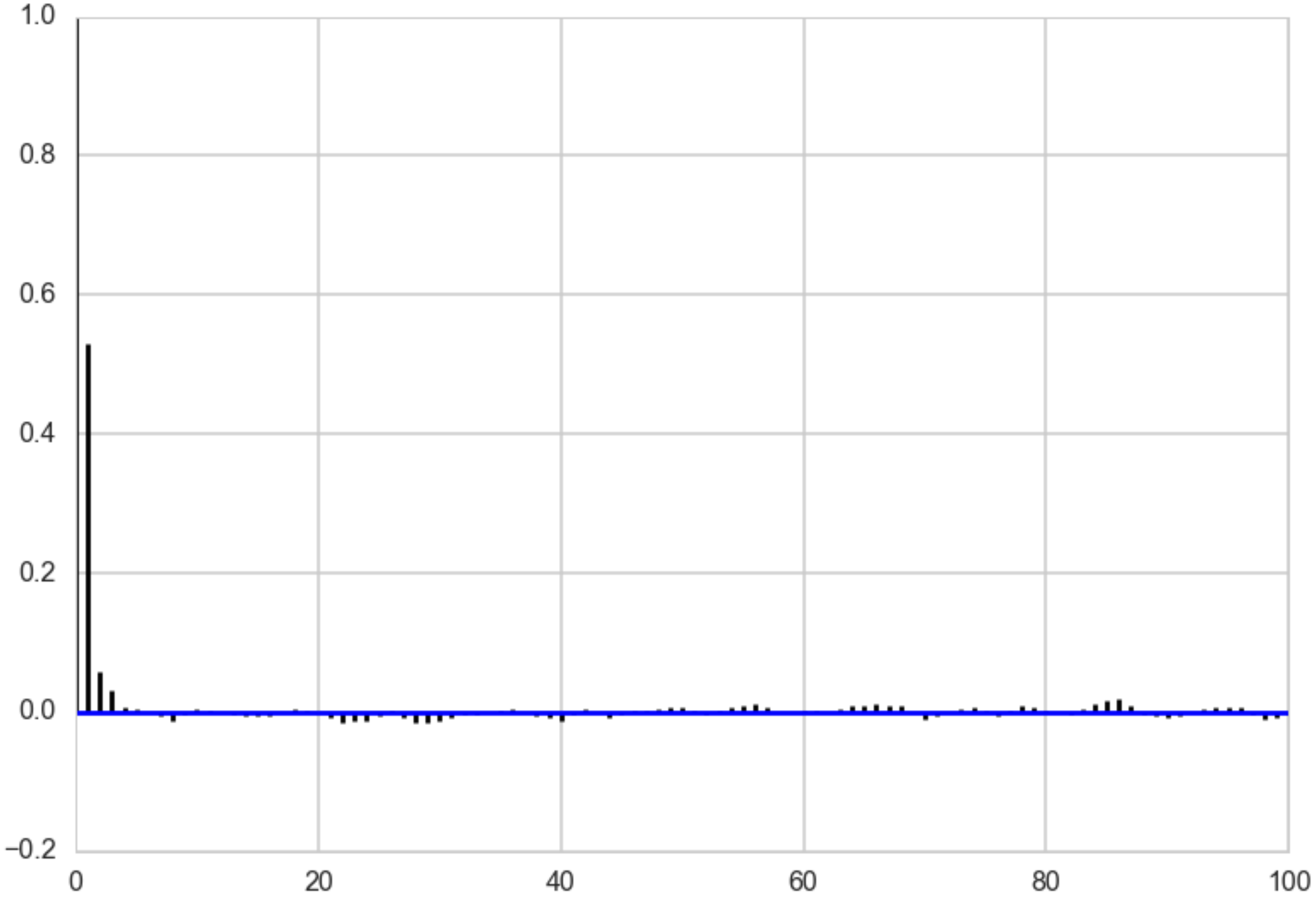
$$A = min(1, \frac{p(x^*_k|x^*{}_{-k})p(x^*{}_{-k})}{p(x^i_k|x^i{}_{-k})p(x^i{}_{-k})} \frac{q_k(x^i|x^*)}{q_k(x^*|x^i)}) = min(1, \frac{p(x^*_k|x^*{}_{-k})p(x^*{}_{-k})}{p(x^i_k|x^i{}_{-k})p(x^i{}_{-k})} \frac{p(x^i_k|x^*{}_{-k})}{p(x^*_k|x^i{}_{-k})})$$

Componentwise update, $\implies x^*{}_{-k} = x^i{}_{-k}$ and A is 1!

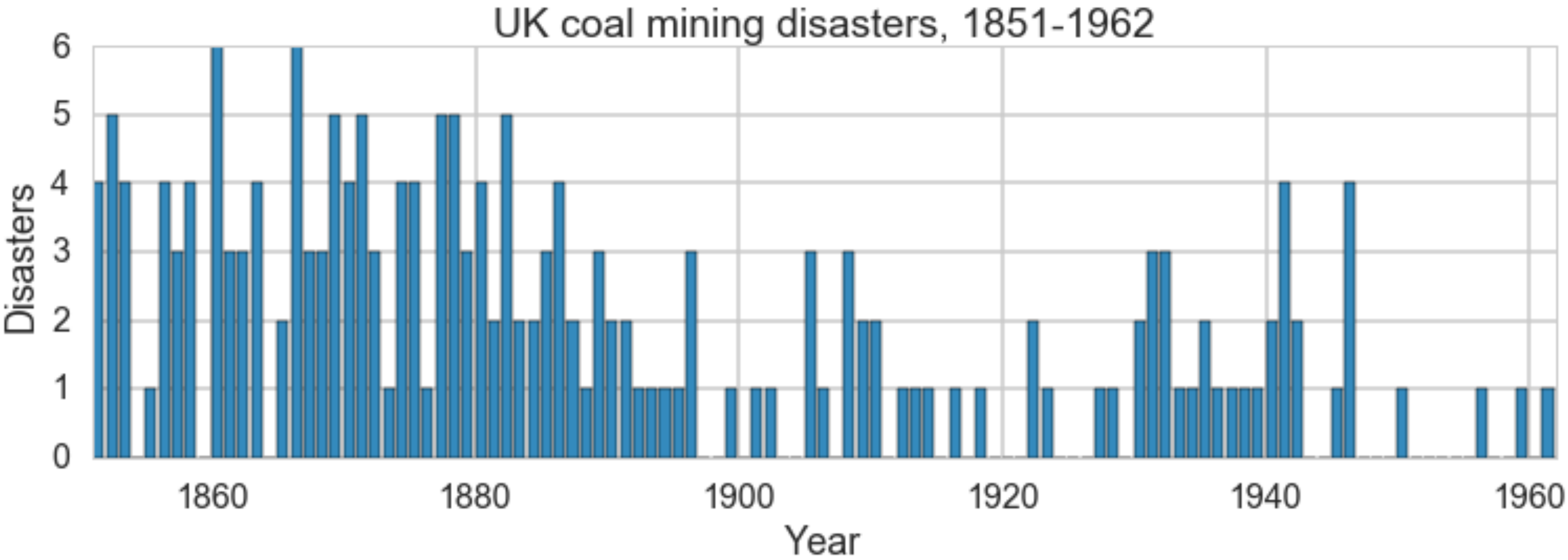
Sampling with pymc3

Diagnostics

Traceplots and Autocorrelation



- the gibbs joint has very little autocorrelation
- highly correlated joints will have lots of autocorrelation
- thinning/longer chains may be required, but as usual it depends on what you are trying to calculate.
- expectations require far fewer samples
- complete posterior characterization require many more
- can come from various sources


UK coal mining disasters, 1851-1962
Disasters
6
5
4
3
2
1
0
1860
1880
1900
1920
1940
1960
Year

Model

$$y|\tau, \lambda_1, \lambda_2 \sim Poisson(r_t)$$

$$r_t = \lambda_1 \text{ if } t < \tau \text{ else } \lambda_2 \text{ for } t \in [t_l, t_h]$$

$$\tau \sim DiscreteUniform(t_l, t_h)$$

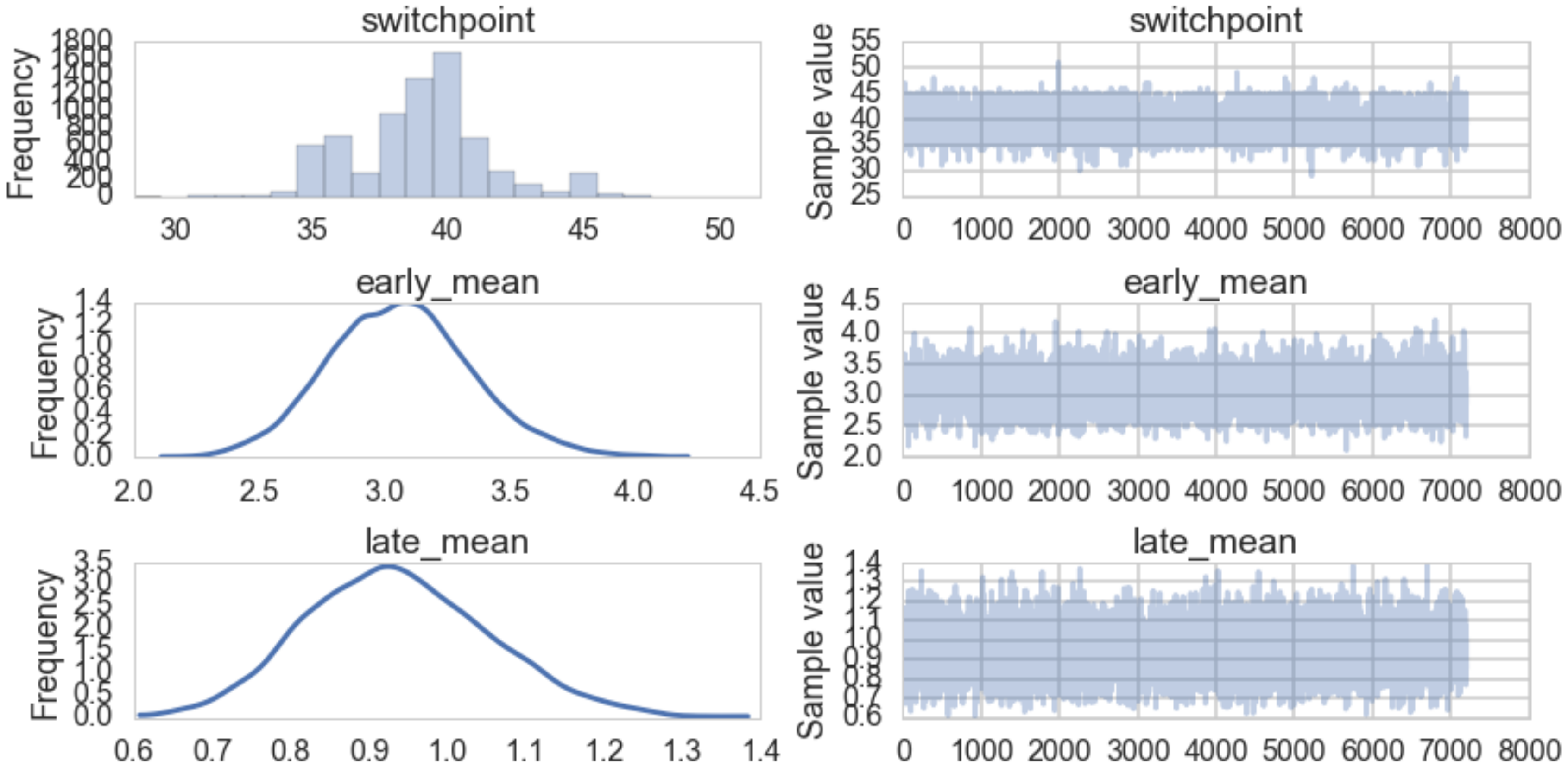
$$\lambda_1 \sim Exp(a)$$

$$\lambda_2 \sim Exp(b)$$

```
from pymc3.math import switch
with pm.Model() as coaldis1:
    early_mean = pm.Exponential('early_mean', 1)
    late_mean = pm.Exponential('late_mean', 1)
    switchpoint = pm.DiscreteUniform('switchpoint', lower=0, upper=n_years)
    rate = switch(switchpoint >= np.arange(n_years), early_mean, late_mean)
    disasters = pm.Poisson('disasters', mu=rate, observed=disasters_data)
```

```
with coaldis1:
    stepper=pm.Metropolis()
    trace = pm.sample(40000, step=stepper)
```

```
100%|██████████| 40000/40000 [00:12<00:00, 3326.53it/s] | 229/40000 [00:00<00:17, 2289.39it/s]
```


switchpoint
Frequency
switchpoint
Sample value
early_mean
Frequency
early_mean
Sample value
late_mean
Frequency
late_mean
Sample value