Likelihoods and Inference

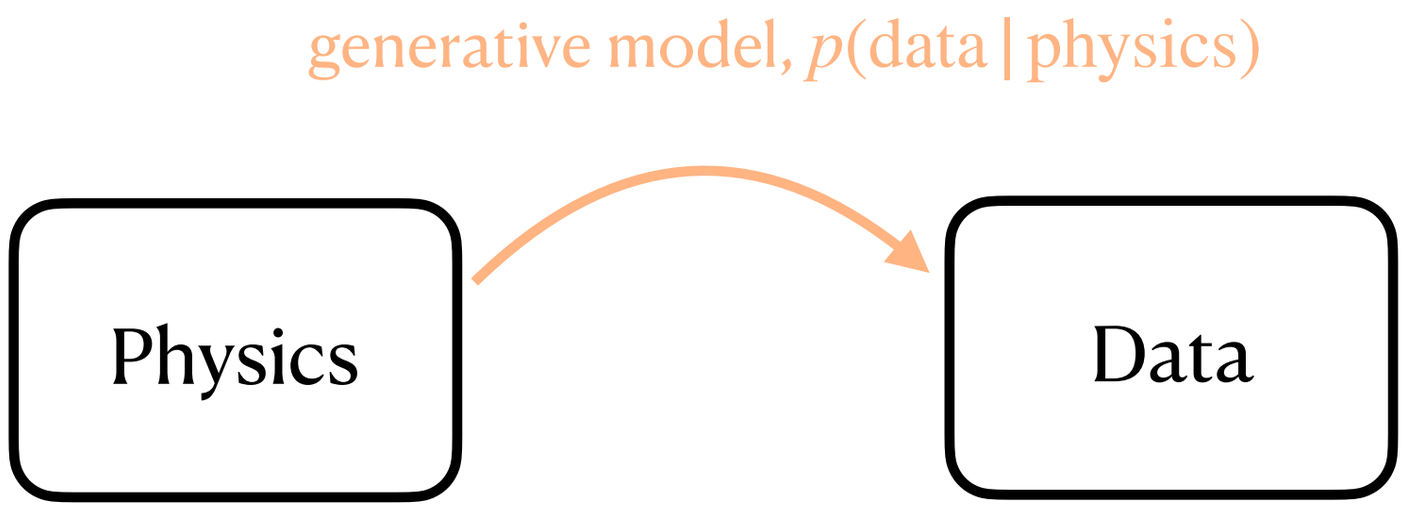
Or: when and how to use MCMC

Seshadri Nadathur, Euclid Advanced School, 17 June 2020

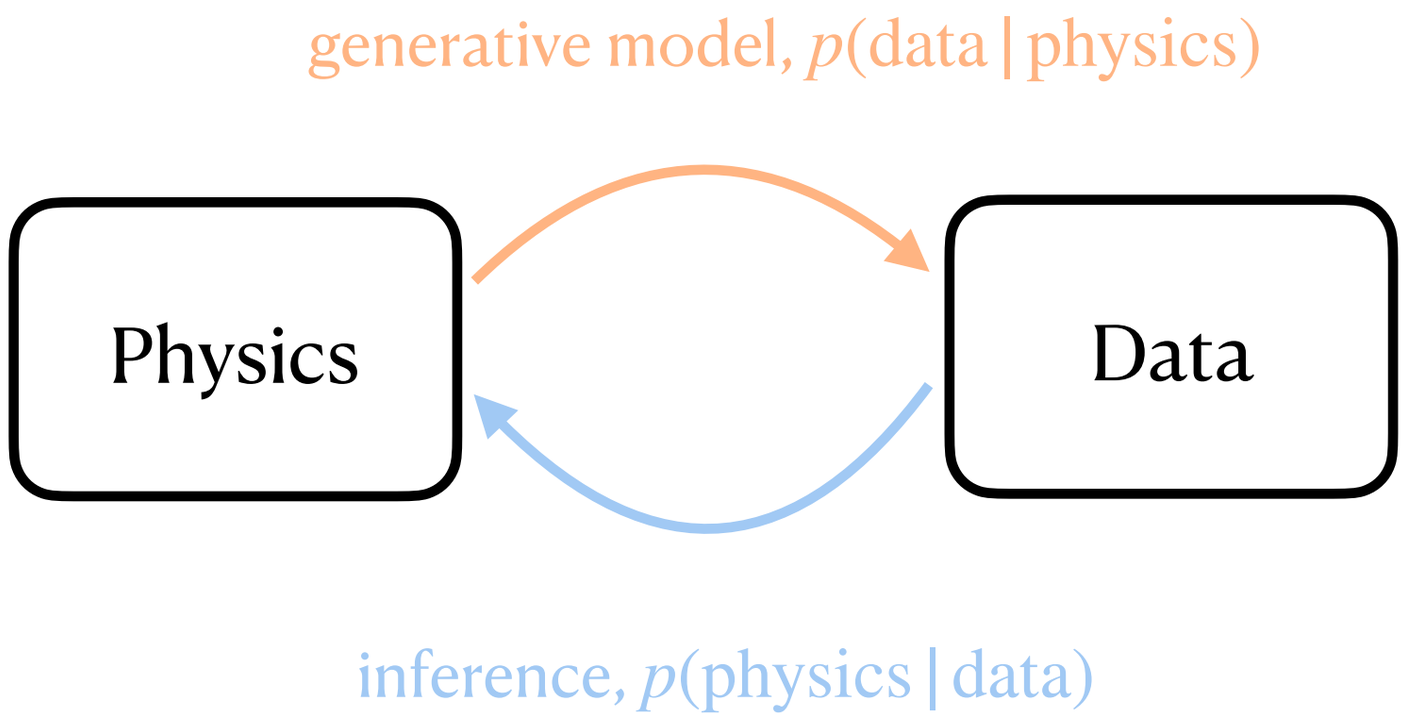
What this lecture covers

- Inferring physics from data
- Sampling and Markov Chain Monte Carlo
- Why/when should you use MCMC? When should you NOT use it?
- Practical introduction to MCMC via Metropolis-Hastings
    - algorithm, tuning, troubleshooting, convergence, reporting results ...
- Advanced MCMC techniques
    - other sampling algorithms
- Common MCMC and related codes for cosmology

The basic picture



The basic picture



Bayes' Theorem

likelihood function
(depends on model of specific scenario)

$$p(\text{physics}\,|\,\text{data}) \propto p(\text{data}\,|\,\text{physics})p(\text{physics})$$

posterior probability
(what we are interested in)

prior (everything we knew before seeing the data)

Bayes' Theorem

likelihood function
(depends on model of specific scenario)

$$p(\theta|D) = \frac{1}{Z} p(D|\theta) p(\theta)$$

posterior probability
(what we are interested in)

prior (everything we knew before seeing the data)

Bayes' Theorem

likelihood function
(depends on model of specific scenario)

$$p(\theta|D) = \frac{1}{Z} p(D|\theta) p(\theta)$$

posterior probability
(what we are interested in)

prior (everything we knew before seeing the data)

"evidence"/"marginal likelihood"/"Bayes integral" etc
(generally hard/impossible to calculate!)

What is a sampling?

For:

- parameters θ – in general, a vector in d-dimensional space
- some pdf $p(\theta)$ satisfying

$$p(\theta) \geq 0 \quad \forall \theta$$

$$\int p(\theta) d\theta = 1$$

a *sampling* is a set $\{\theta_k\}_{k=1}^{K}$ of draws from the pdf such that

$$E_{p(\theta)}[g(\theta)] \equiv \int g(\theta) p(\theta) d\theta \approx \frac{1}{K} \sum_{k=1}^{K} g(\theta_k)$$

What is a sampling?

For:

- parameters θ – in general, a vector in d-dimensional space
- some pdf $p(\theta)$ satisfying

$$p(\theta) \geq 0 \quad \forall \theta$$

$$\int p(\theta) d\theta = 1$$

a *fair sampling* is a set $\{\theta_k\}_{k=1}^{K}$ of draws from the pdf such that

$$E_{p(\theta)}[g(\theta)] \equiv \int g(\theta) p(\theta) d\theta \approx \frac{1}{K} \sum_{k=1}^{K} g(\theta_k)$$

expectation value

any function over θ

What is a sampling?

$$E_{p(\theta)}[g(\theta)] \equiv \int g(\theta)p(\theta)d\theta \approx \frac{1}{K}\sum_{k=1}^{K} g(\theta_k)$$

A good sampling allows us to perform integrals easily

What is a sampling?

$$E_{p(\theta)}[g(\theta)] \equiv \int g(\theta)p(\theta)d\theta \approx \frac{1}{K}\sum_{k=1}^{K} g(\theta_k)$$

Markov Chain Monte Carlo (MCMC):
a method for *generating a good sampling* from the posterior

What is a sampling?

holds if $\theta_k \sim p(\theta)$

$$E_{p(\theta)}[g(\theta)] \equiv \int g(\theta)p(\theta)d\theta \approx \frac{1}{K}\sum_{k=1}^{K} g(\theta_k)$$

Markov Chain Monte Carlo (MCMC):
a method for *generating a good sampling* from the posterior

Sampling the posterior

Remember, we don't usually know the normalisation:

$$p(\theta \mid D) = \frac{1}{Z} p(D \mid \theta) p(\theta)$$

Instead of $p(\theta)$ we have some $f(\theta)$:

$$f(\theta) \geq 0 \;\; \forall \theta \qquad , \qquad \int f(\theta) d\theta = Z$$

But that's OK!

Sampling the posterior

$$E_{p(\theta)}[g(\theta)] \equiv \frac{\int g(\theta) f(\theta) d\theta}{\int f(\theta) d\theta} \approx \frac{1}{K} \sum_{k=1}^{K} g(\theta_k)$$

We can sample from the badly normalised pdf

Can still approximate all integrals the same way!

Sampling the posterior

$$E_{p(\theta)}[g(\theta)] \equiv \frac{\int g(\theta) f(\theta) d\theta}{\int f(\theta) d\theta} \approx \frac{1}{K} \sum_{k=1}^{K} g(\theta_k)$$

error in the integral approximation, $\Delta \propto \frac{1}{\sqrt{N}}$

where $N < K$ is number of *independent* samples from posterior

When do you want an MCMC?

If (and only if) you want to compute an *integral!*

(Mean, median, quantiles etc of any function over posterior)

(**NOT** to find the “best-fit”!)

Generating a sampling

numerical integration
by randomly sampling
integrand

generate new
samples based
on existing ones

Markov Chain Monte Carlo

step θ_{k+1} depends only on state at θ_k

Generating a sampling

numerical integration
by randomly sampling
integrand

generate new
samples based
on existing ones

Markov Chain Monte Carlo

step θ_{k+1} depends only on state at θ_k

biased random walk

Metropolis-Hastings MCMC

From θ_k, to move chain to θ_{k+1}:

- pick $\theta' \sim q(\theta' | \theta_k)$

proposal pdf, must be symmetric,
$q(\theta' | \theta_k) = q(\theta_k | \theta')$

- pick $r \sim U(0,1)$
- if $\dfrac{f(\theta')}{f(\theta_k)} > r$, assign $\theta_{k+1} \leftarrow \theta'$ (**accept** proposal)

  else assign $\theta_{k+1} \leftarrow \theta_k$ (**reject**, repeat step)

Metropolis-Hastings MCMC

From θ_k, to move chain to θ_{k+1}:

- pick $\theta' \sim q(\theta' | \theta_k)$

proposal pdf, must be symmetric,
$q(\theta' | \theta_k) = q(\theta_k | \theta')$

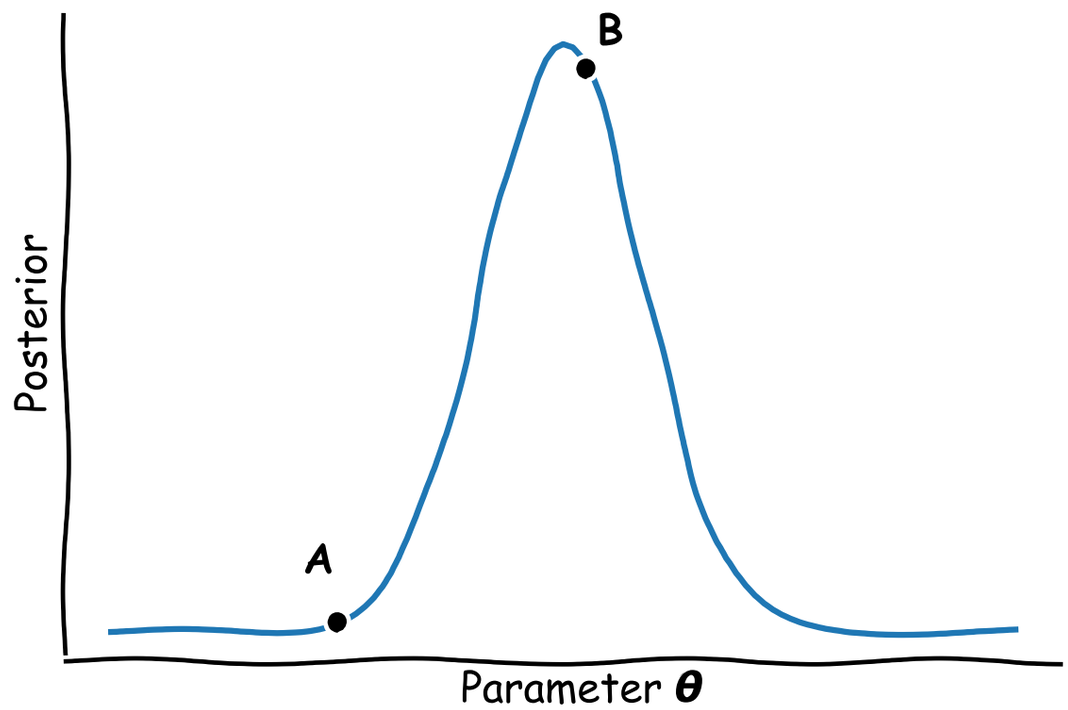
- pick $r \sim U(0,1)$
- if $\dfrac{f(\theta')}{f(\theta_k)} > r$, assign $\theta_{k+1} \leftarrow \theta'$ (**accept** proposal)

  else assign $\theta_{k+1} \leftarrow \theta_k$ (**reject**, repeat step)

$$\text{i.e., } p(\text{accept}) = \min\left(1, \frac{f(\theta')}{f(\theta_k)}\right)$$

Metropolis-Hastings MCMC

"Detailed balance"



$$p(\text{start at } A)p(\text{accept } A \to B)) = p(\text{start at } B)p(\text{accept } B \to A)$$

Metropolis-Hastings MCMC

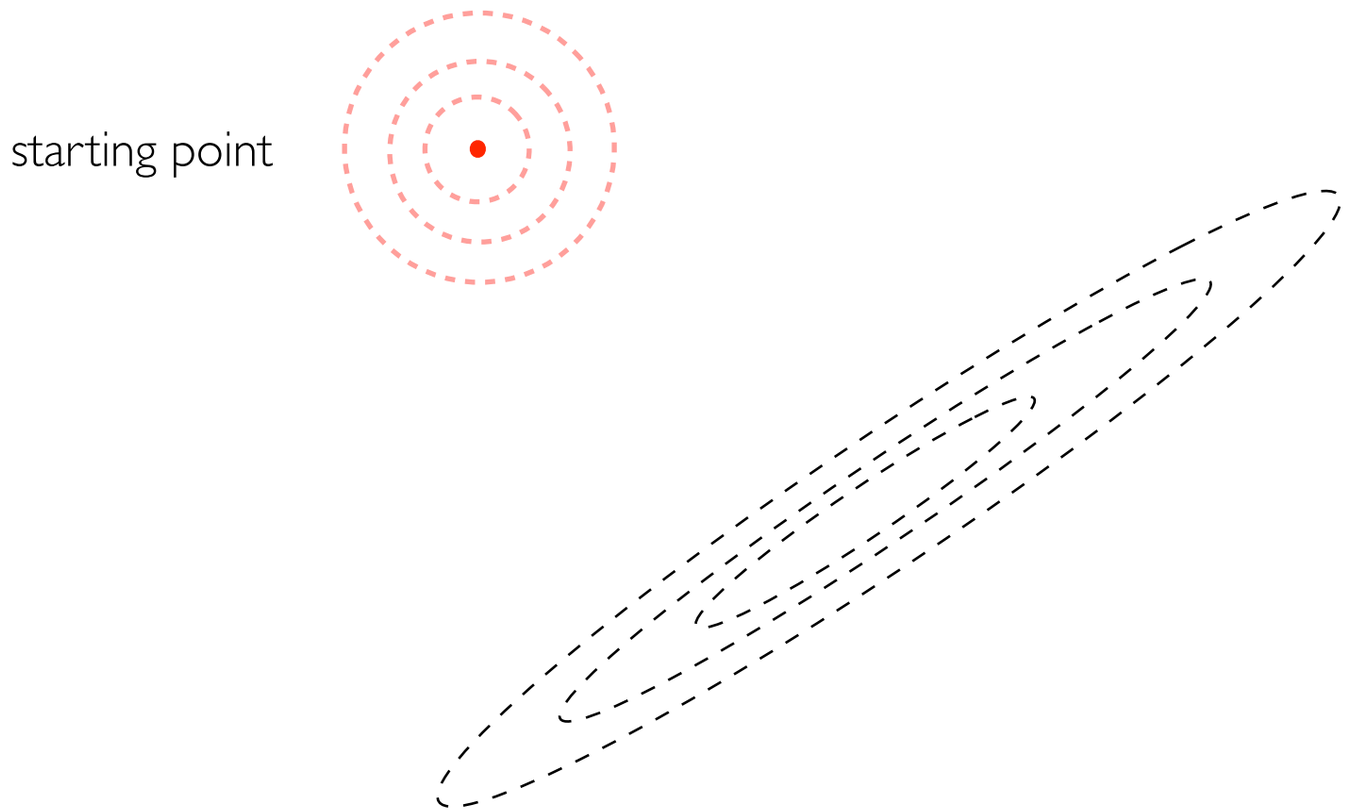
From θ_k, to move chain to θ_{k+1}:

modified acceptance, so proposal need not be symmetric

- pick $\theta' \sim q(\theta' | \theta_k)$
- $p(\text{accept}) = \min\left(1, \frac{f(\theta')}{f(\theta_k)} \frac{q(\theta_k | \theta')}{q(\theta' | \theta_k)}\right)$

in practice, nearly always use symmetric multivariate Gaussian proposal

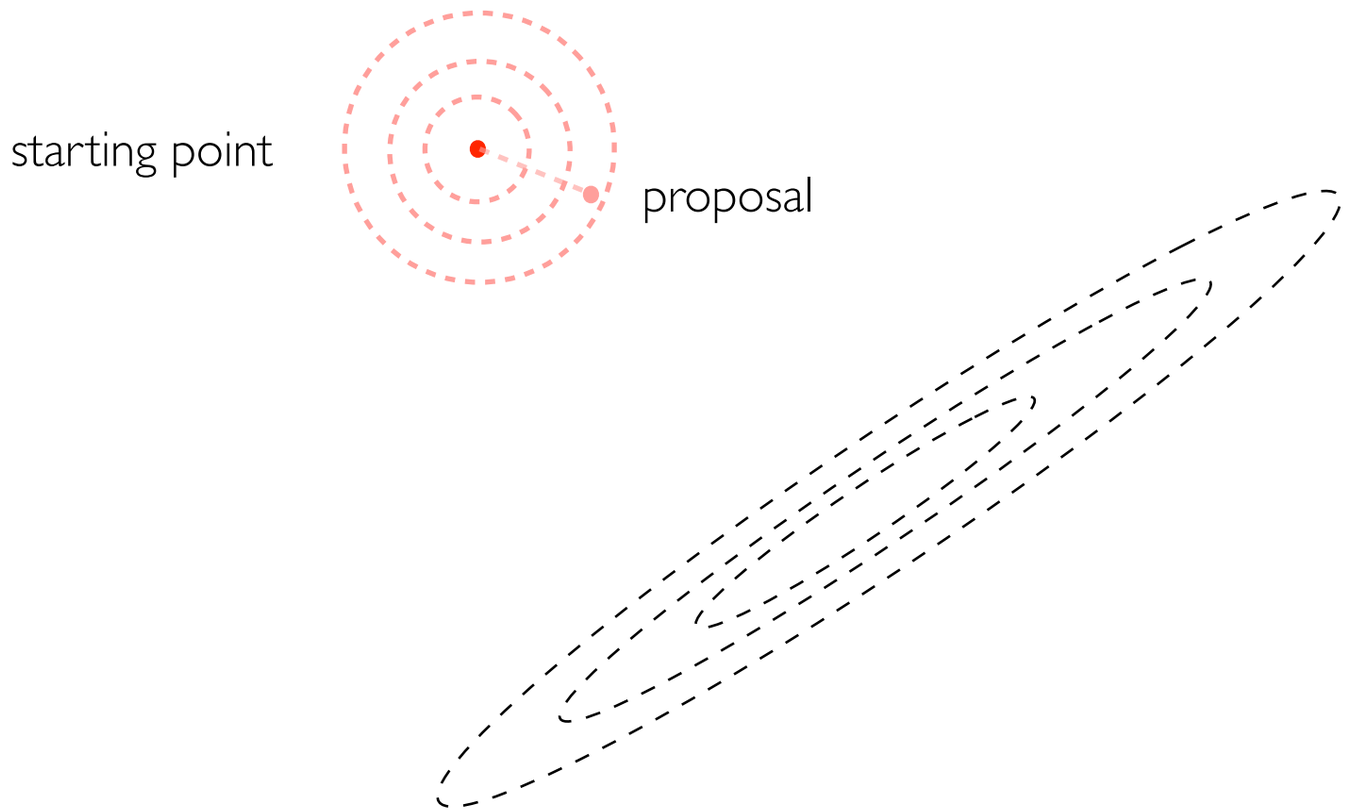
Achieving a good sampling



what we want to happen

slides inspired by D. Foreman-Mackey

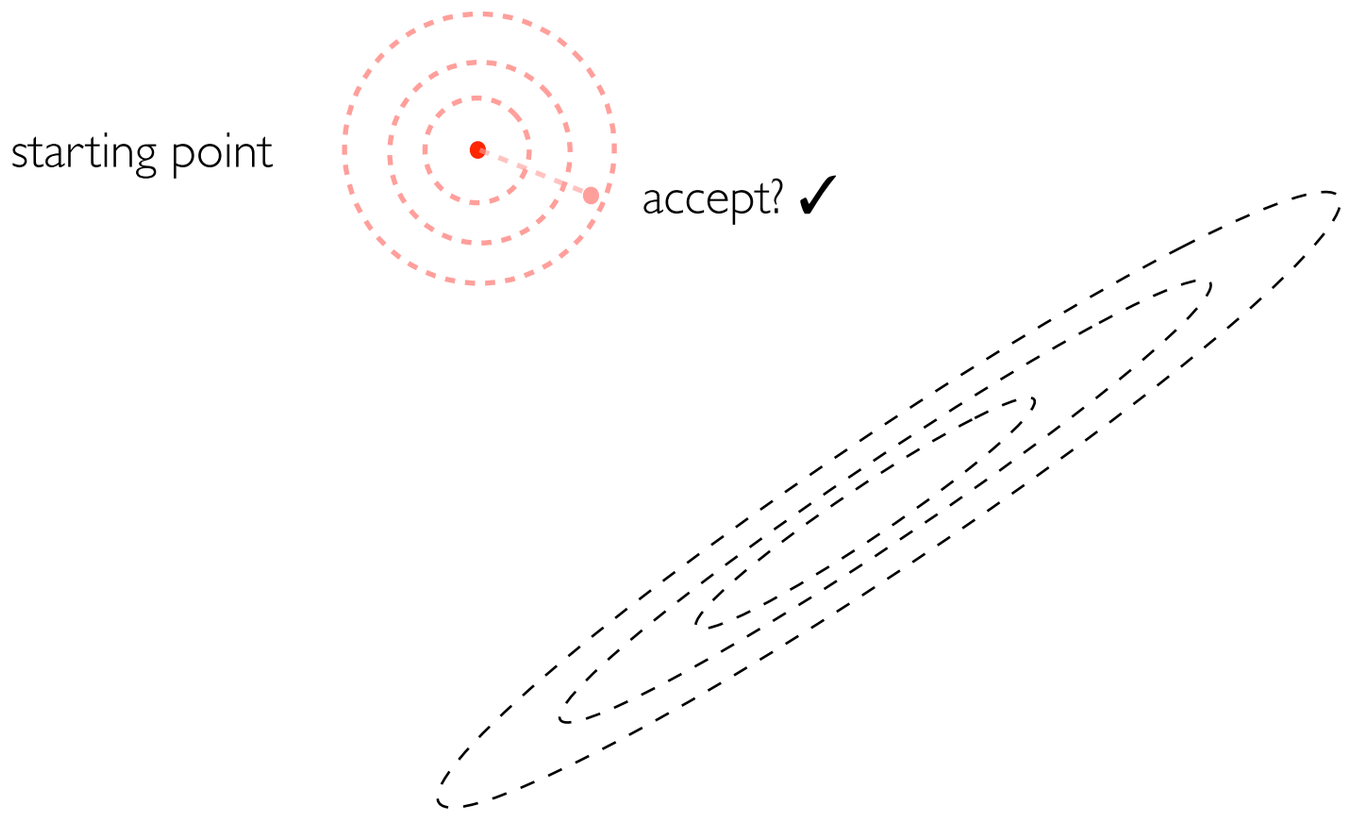
Achieving a good sampling



what we want to happen

slides inspired by D. Foreman-Mackey

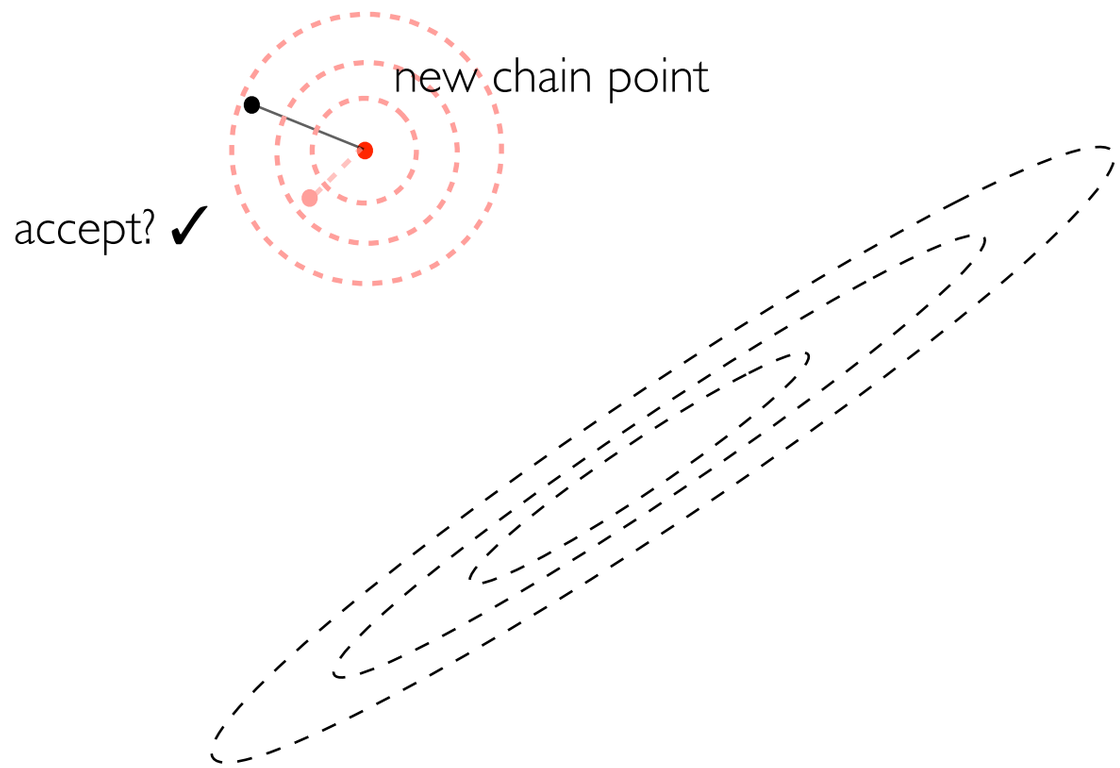
Achieving a good sampling



what we want to happen

slides inspired by D. Foreman-Mackey

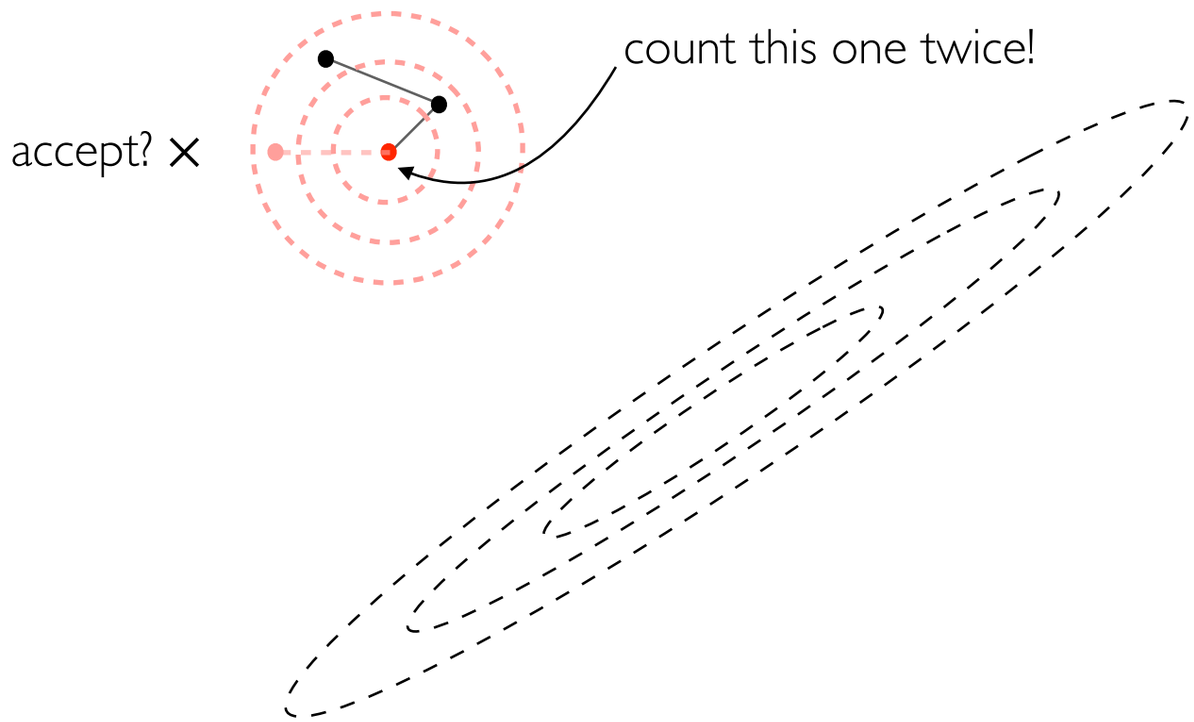
Achieving a good sampling



what we want to happen

slides inspired by D. Foreman-Mackey

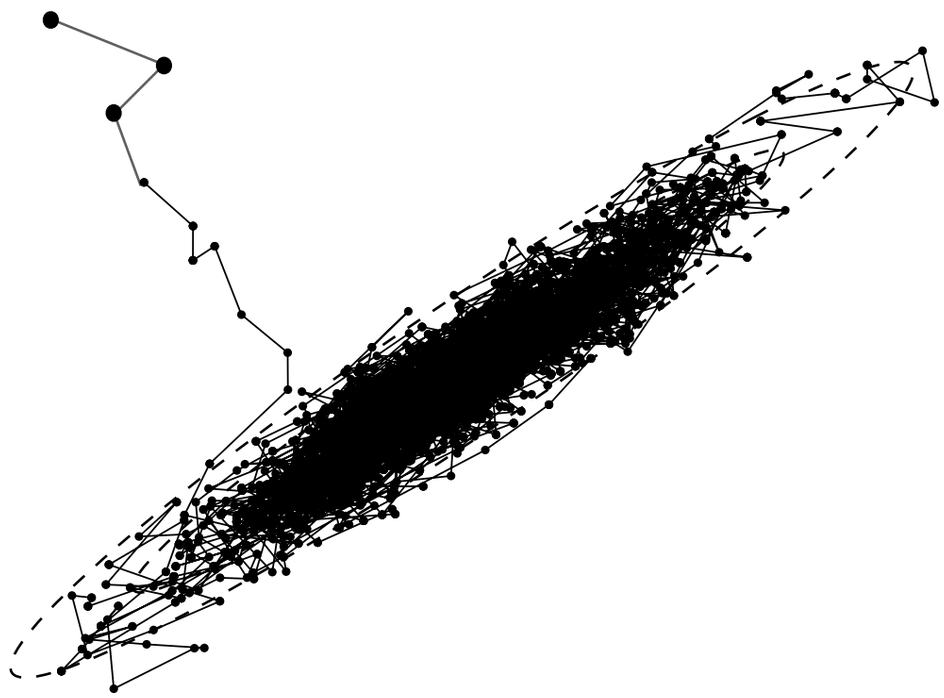
Achieving a good sampling



what we want to happen

slides inspired by D. Foreman-Mackey

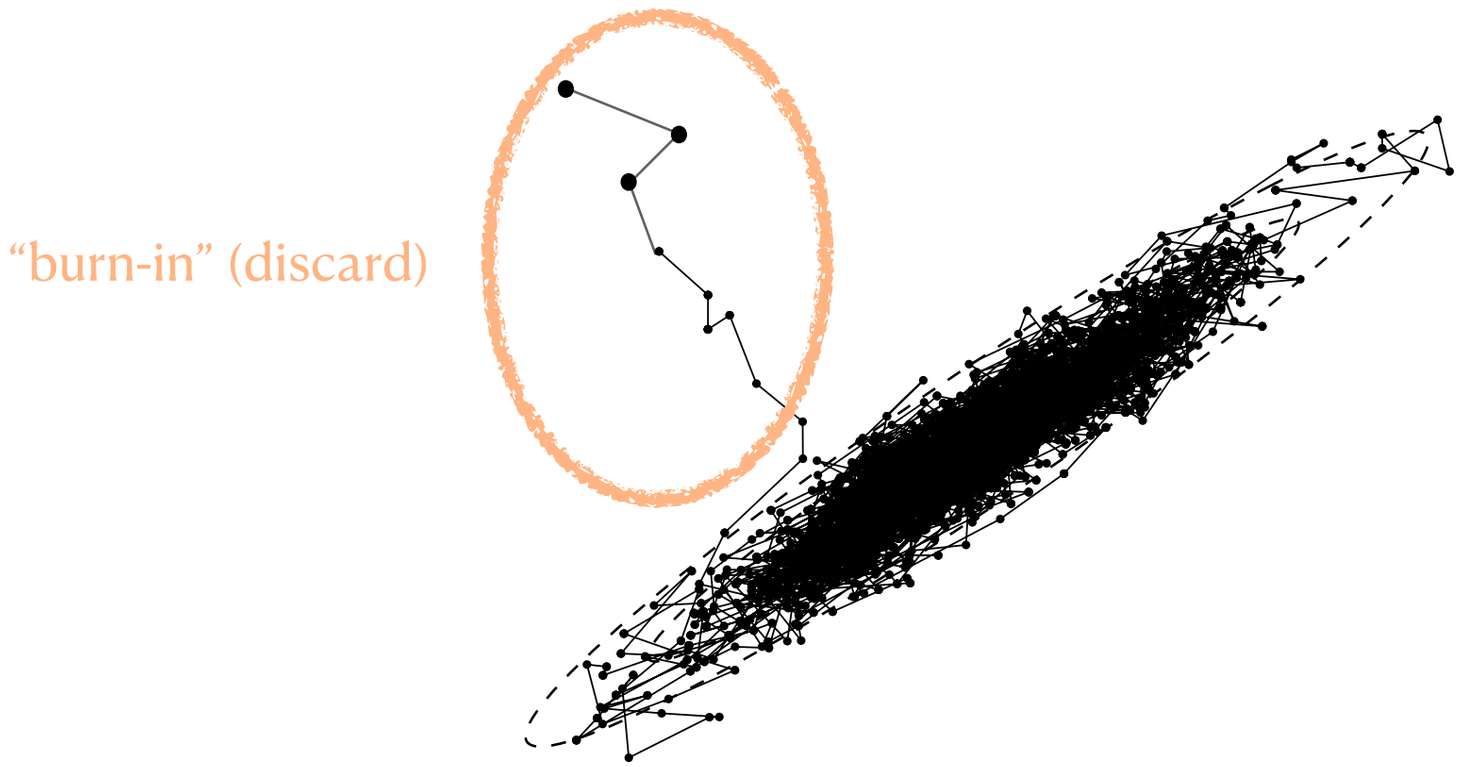
Achieving a good sampling



what we want to happen

slides inspired by D. Foreman-Mackey

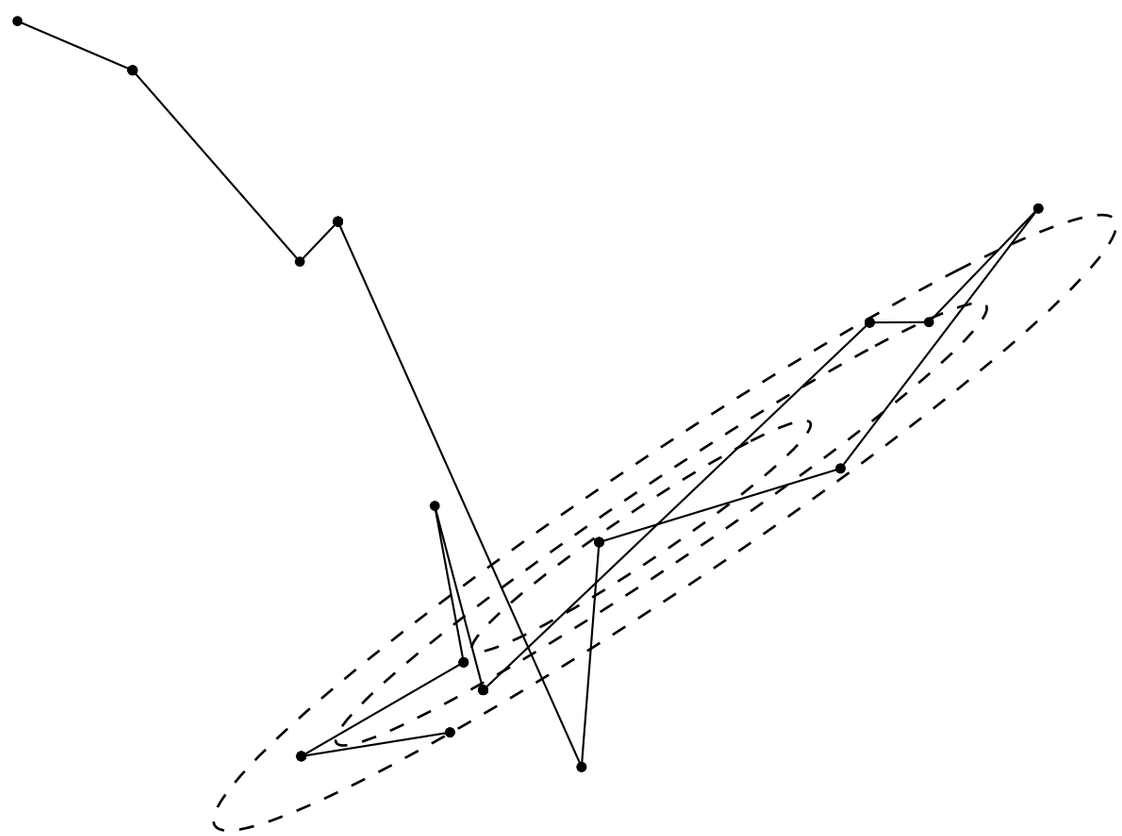
Achieving a good sampling



what we want to happen

slides inspired by D. Foreman-Mackey

Achieving a good sampling

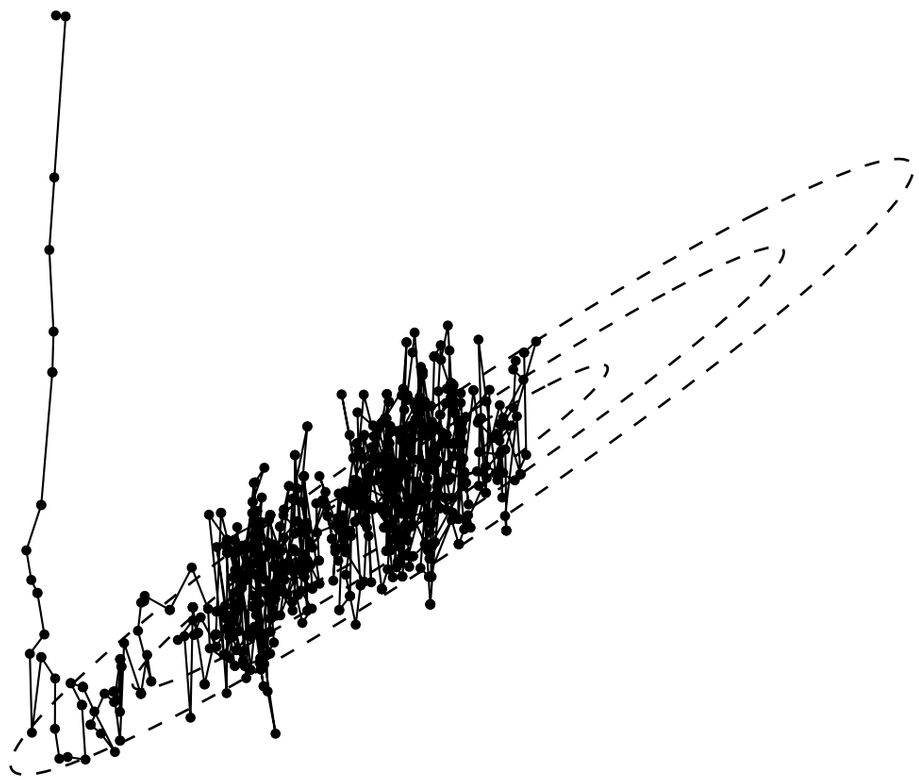


proposed steps too big
most steps rejected
chain very slow

what often happens

slides inspired by D. Foreman-Mackey

Achieving a good sampling



proposed steps too small
most steps accepted
chain very slow

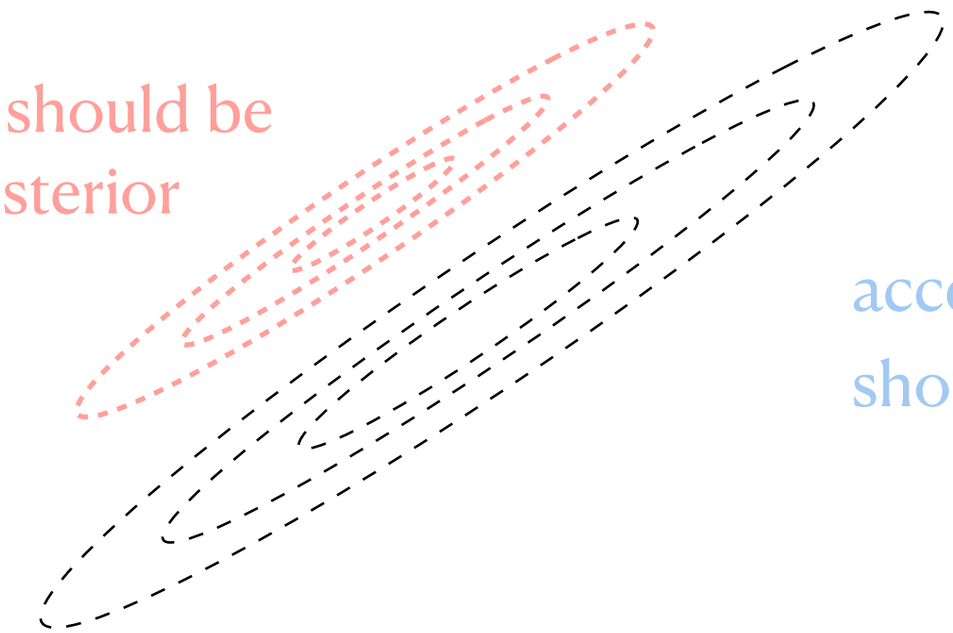
what often happens

slides inspired by D. Foreman-Mackey

Achieving a good sampling

for ideal operation:

proposal shape should be similar to posterior



acceptance fraction should be ~ 0.24

but this requires tuning $D(D+1)/2$ parameters!

slides inspired by D. Foreman-Mackey

Tuning Metropolis-Hastings

In practice:

- calculate covariance matrix of samples so far
- use this to estimate proposal pdf on-the-fly
- after N steps, stop updating proposal
- discard first N samples in chain (burn-in)

Tuning Metropolis-Hastings

In practice:

- calculate covariance matrix of samples so far
- use this to estimate proposal pdf on-the-fly
- after N steps, stop updating proposal
- discard first N samples in chain (burn-in)

for large dimensional problems, this is very expensive!

Go and code!

Write a simple M-H sampler yourself* in your favourite language!

* after the lecture

if you get stuck, try notes at https://github.com/seshnadathur/IntroductionToMCMC

Convergence

Or: how do we know when to stop?

Resources are not infinite; we can’t run chains for ever

Convergence

Or: how do we know when to stop?

Remember, fundamentally:

$$E_{p(\theta)}[g(\theta)] \equiv \frac{\int g(\theta) f(\theta) d\theta}{\int f(\theta) d\theta} \approx \frac{1}{K} \sum_{k=1}^{K} g(\theta_k)$$

error in the integral approximation, $\Delta \propto \frac{1}{\sqrt{N}}$

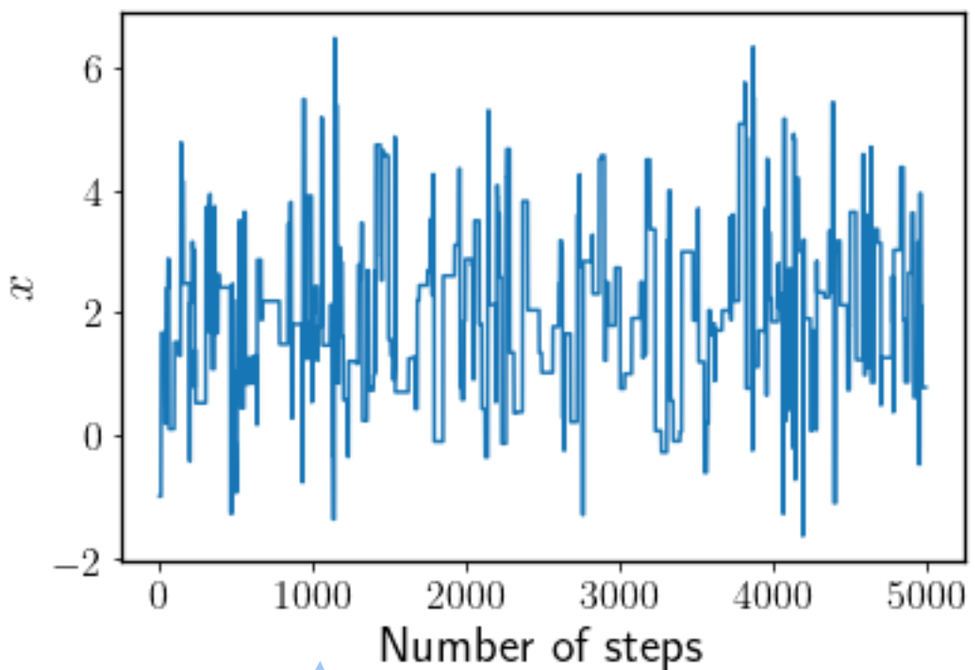
where $N < K$ is number of *independent* samples from posterior

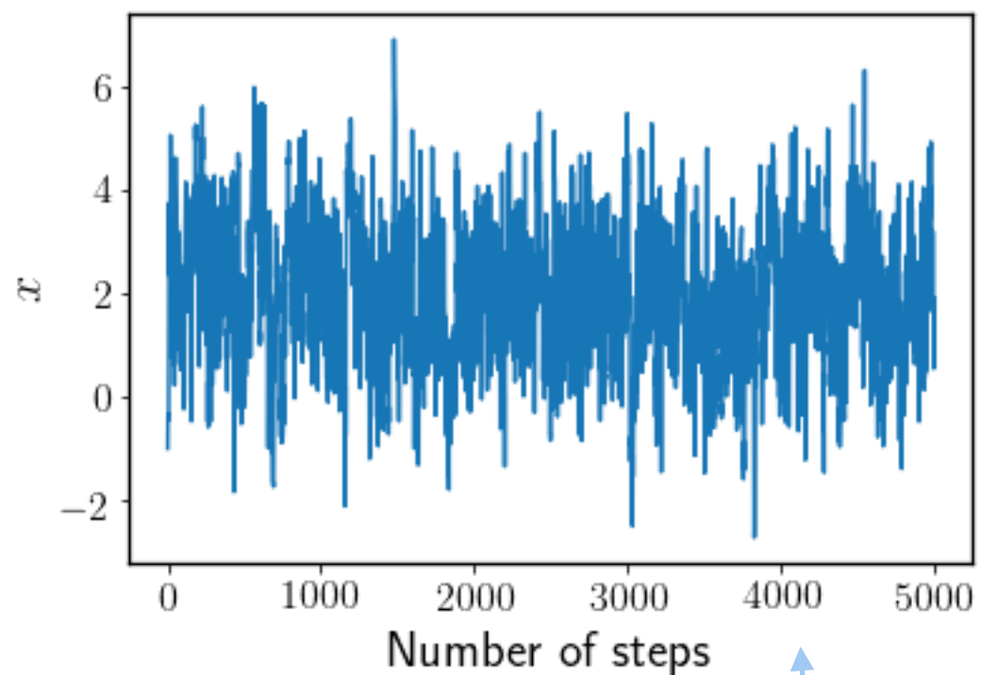
Convergence

Or: how do we know when to stop?

So, Heuristic #1:

- calculate *integrated autocorrelation time*, τ, for chain





τ is large

τ is small

(these are known as "trace plots")

Convergence

Or: how do we know when to stop?

So, Heuristic #1:

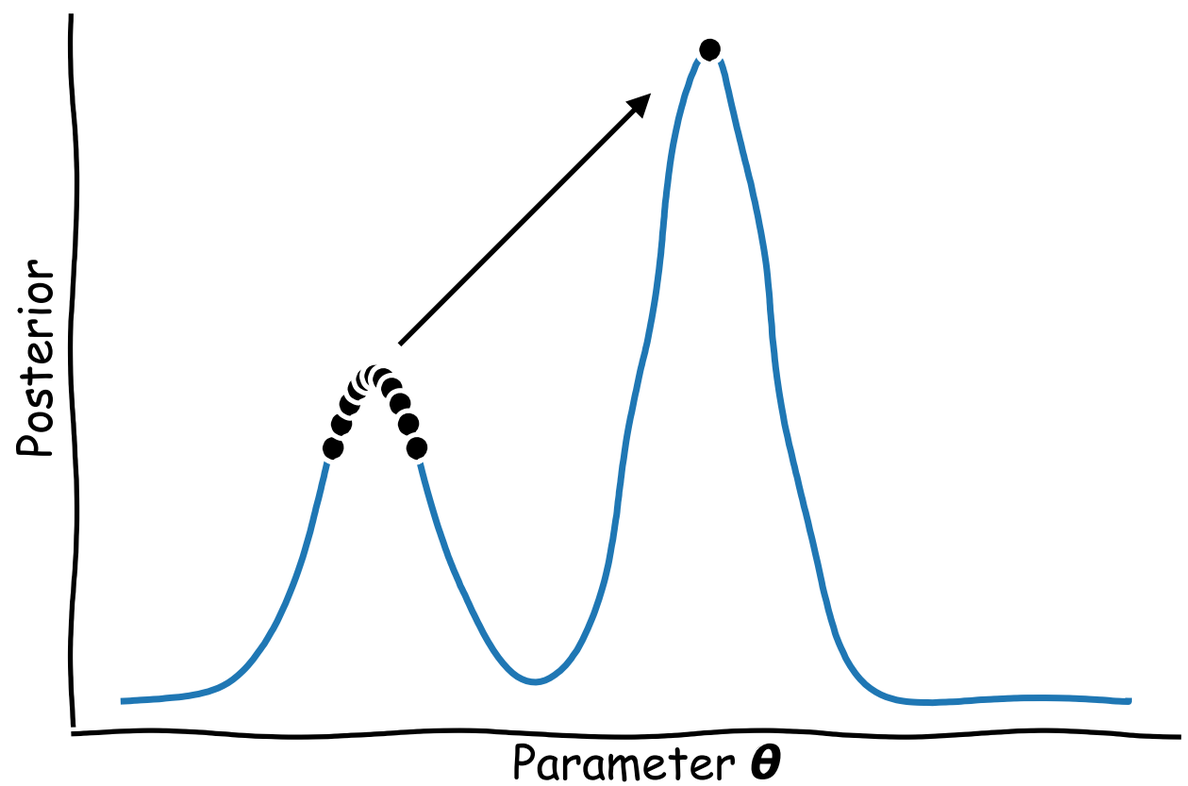
- calculate *integrated autocorrelation time*, τ, for chain
- keep running chain until #steps $K \gg \tau$
- error in integral will be

$$\sigma^2 = \frac{\tau}{K} \mathrm{Var}_p(\theta)\left[g(\theta)\right]$$

* estimating τ is tricky! see, e.g., https://emcee.readthedocs.io/en/stable/tutorials/autocorr/#autocorr

Convergence

Or: how do we know when to stop?



could get stuck in a local minimum!

Convergence

Or: how do we know when to stop?

So, Heuristic #2:

- run several chains with well-spaced starting points
- check they all converge to same answer

Convergence

Or: how do we know when to stop?

Gelman-Rubin diagnostic:

R, compares variance within single chain to variance between all chains

usually require $R - 1 < 0.1$ (or < 0.01, or < 0.001)

* $R = \sqrt{\frac{K-1}{K}W + \frac{N+1}{NK}B}$,

where W = mean variance of parameter θ within individual chains, B = variance of mean of θ between chains, K = length of each chain, and N = number of chains

Convergence

Or: how do we know when to stop?

But in principle, we can

NEVER*

100% prove chain has fully sampled posterior

$\Longrightarrow$ all diagnostics are indicators, not guarantees

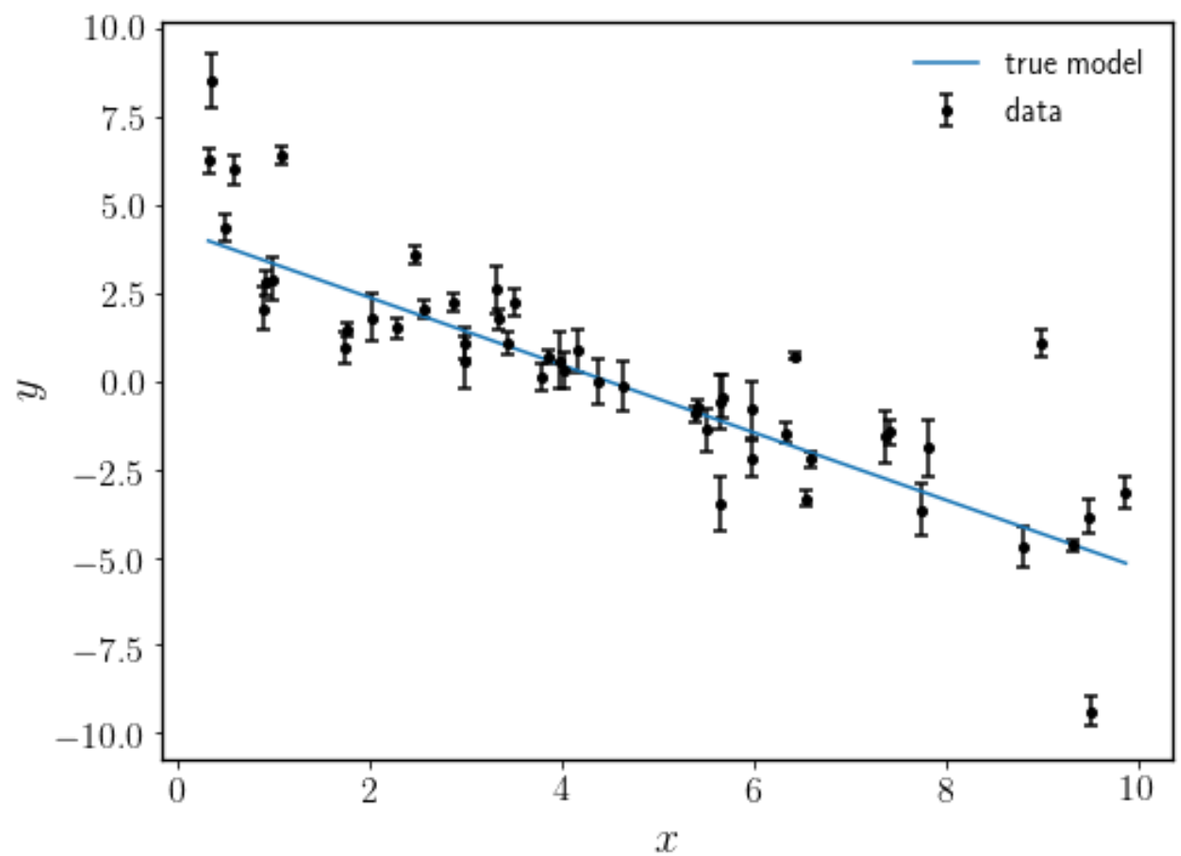
$\Longrightarrow$ always inspect chains carefully

* except in special cases where posterior satisfies some mathematical properties

A simple example

Simple linear generative model, with systematically underestimated random errors

$y = m_{\text{true}}x + b_{\text{true}} + \sigma_2(y)\sigma_1$, where $\sigma_1 \sim \mathcal{N}(0.1, 0.5)$ and $\sigma_2(y) \sim \mathcal{N}(0, f_{\text{true}}y)$

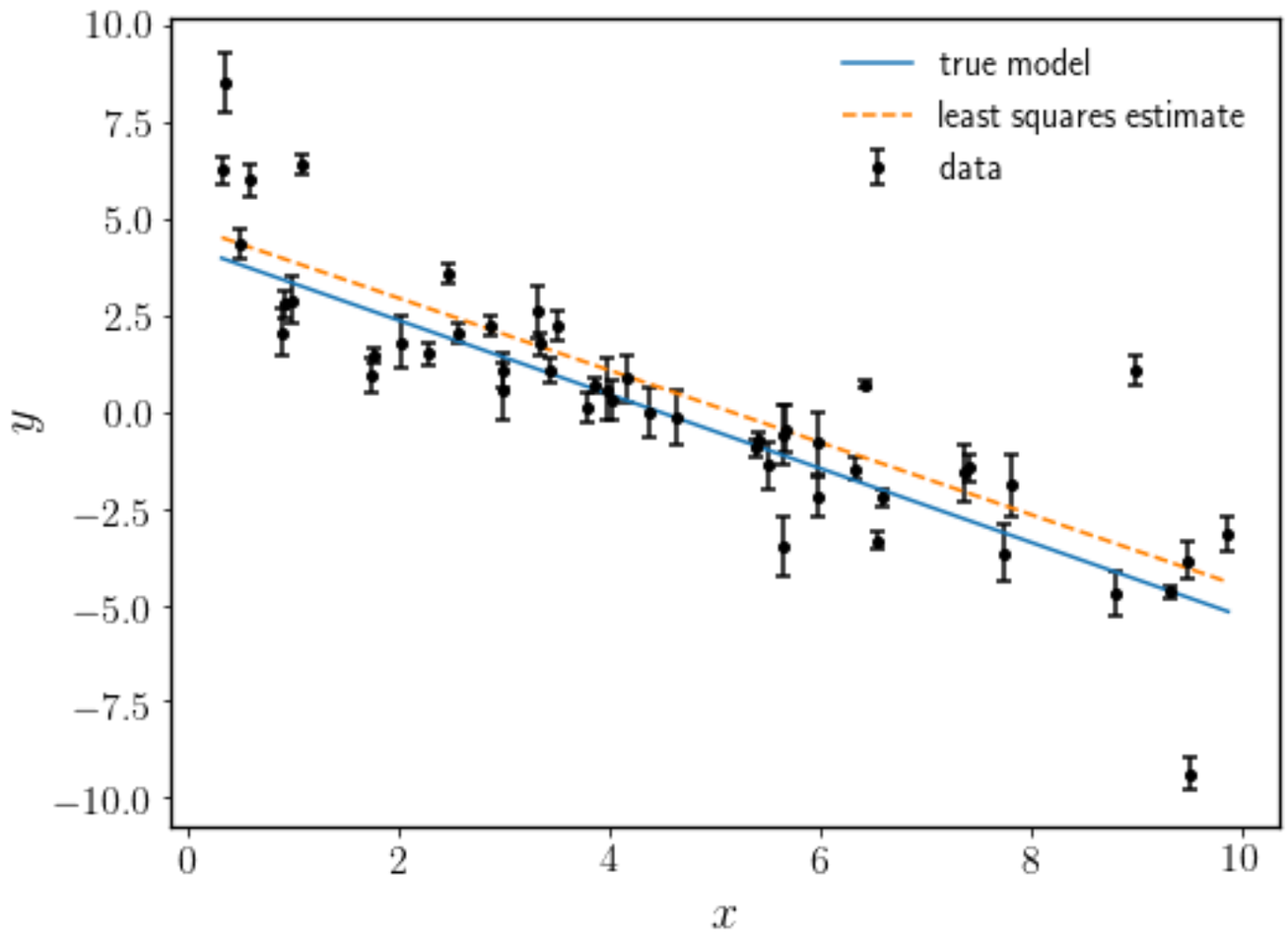


"known" errors shown here are just σ_1

example taken from emcee documentation

A simple example

We chose this model because a native least-squares fit doesn't do well!



example taken from emcee documentation

A simple example

Let's write code for likelihood and posterior:

```
def lnlike(theta, data, covmat):
    m, b, lnf = theta
    x = data[0]
    y = data[1]
    yerr = np.sqrt(np.diag(covmat))

    model = m * x + b
    inv_sigma2 = 1.0/(yerr**2 + model**2*np.exp(2*lnf))

    return -0.5*(np.sum((y - model)**2 *inv_sigma2 - np.log(inv_sigma2)))

def lnprior(theta):
    m, b, lnf = theta
    if -5.0 < m < 0.5 and 0.0 < b < 10.0 and -10.0 < lnf < 1.0:
        return 0.0
    return -np.inf

def lnprob(theta, data, covmat):

    x = data[0]
    y = data[1]
    yerr = np.sqrt(np.diag(covmat))

    lp = lnprior(theta)
    if not np.isfinite(lp):
        return -np.inf
    return lp + lnlike(theta, data, covmat)

data = np.vstack([x, y])
covmat = np.diag(yerr * yerr)
```

we ALWAYS sample from posterior, not the likelihood!
so define your priors explicitly
(and better to have "proper" priors)

posterior = prior × likelihood

example taken from emcee documentation

A simple example

Try to sample this posterior with your M-H code*

* that you will write after this lecture!

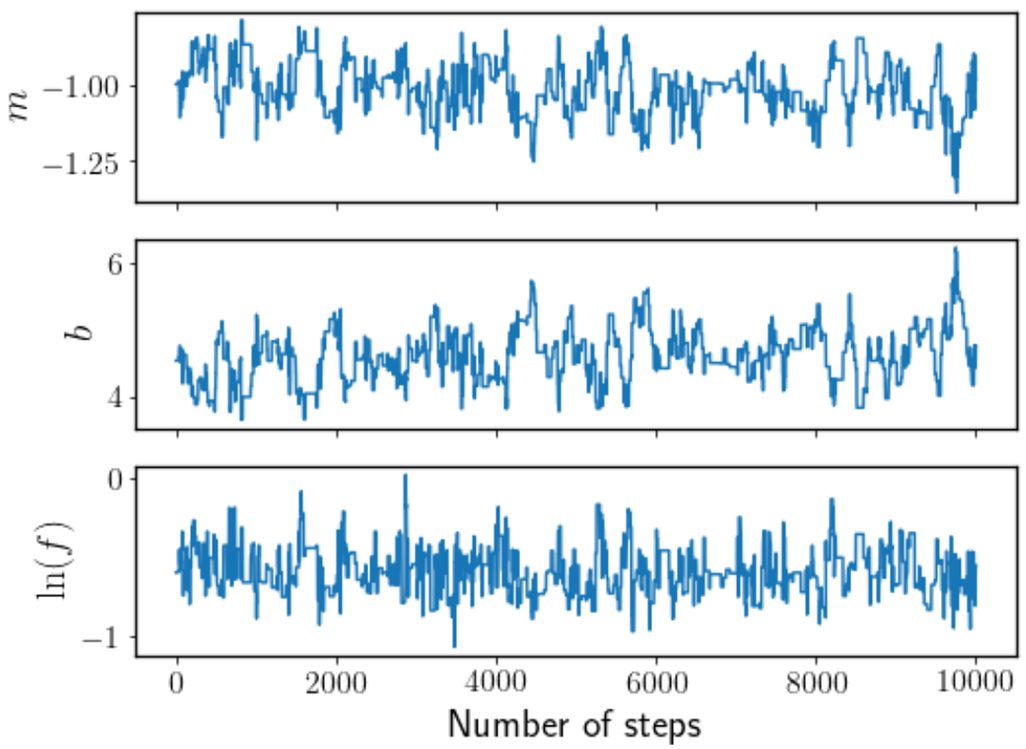
With no advance knowledge, use a diagonal proposal matrix, run for 10,000 steps

A simple example

Try to sample this posterior with your M-H code*

* that you will write after this lecture!

With no advance knowledge, use a diagonal proposal matrix, run for 10,000 steps




```
Current acceptance fraction, a = 0.07
```

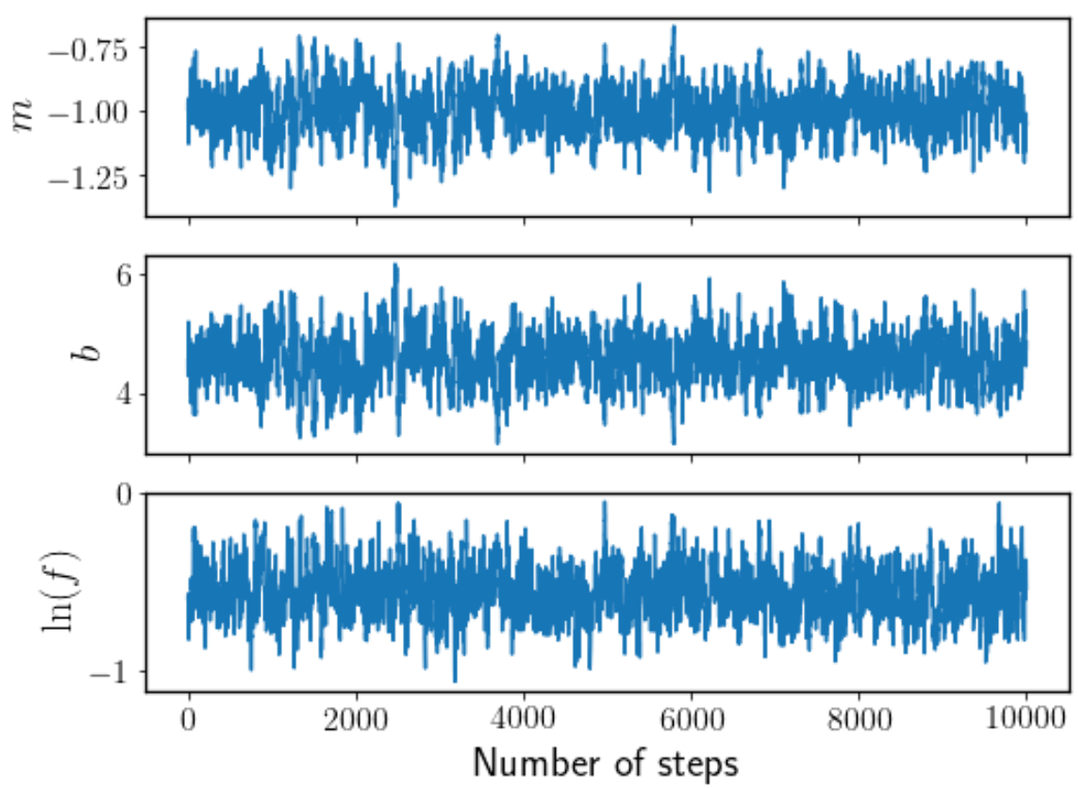
these two heuristics tell us convergence is poor!

why?

A simple example

Now tune your proposal based on covariance of samples from first run

Then re-run chain for 10,000 steps:

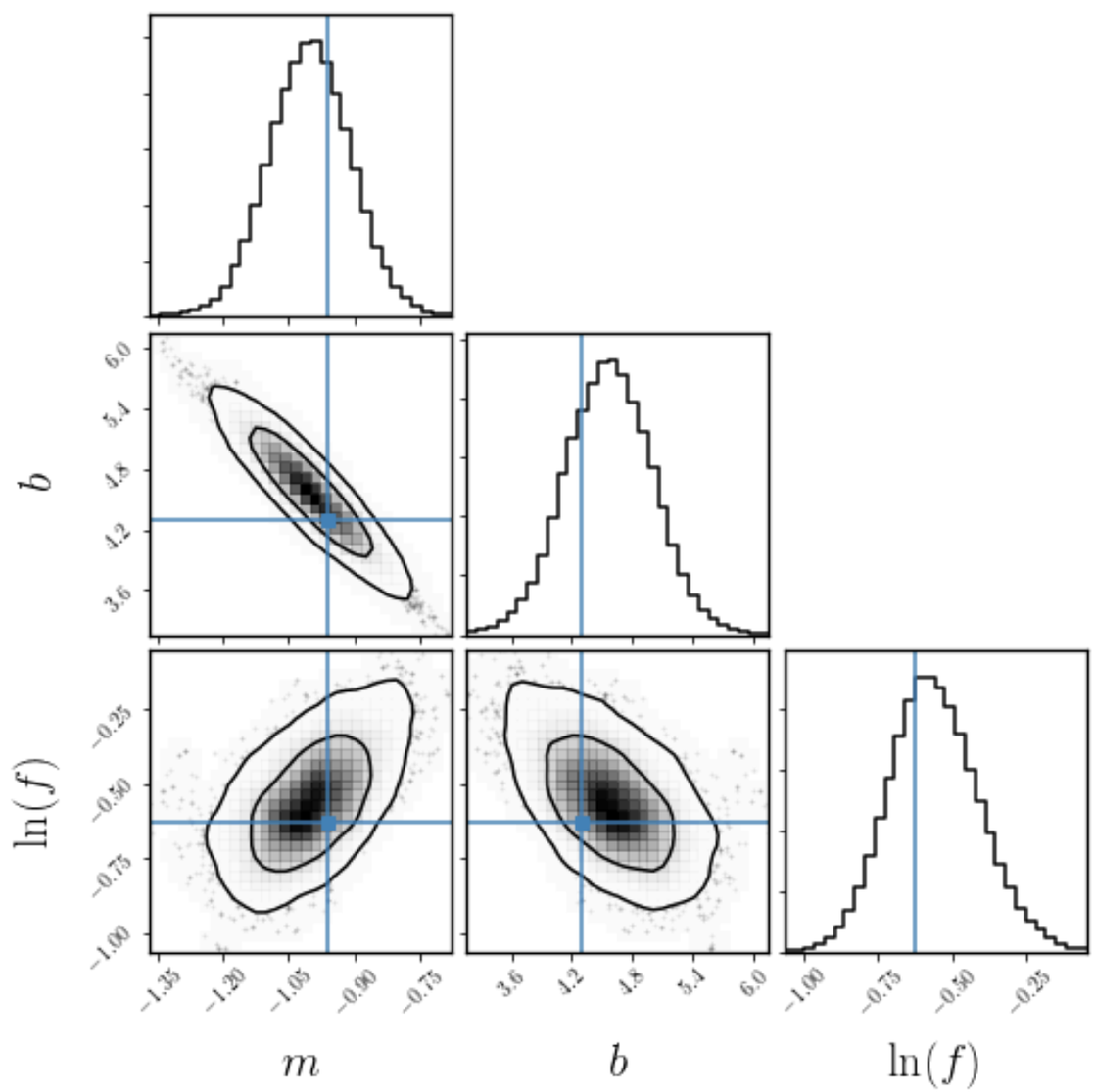



```
Current acceptance fraction, a = 0.66
```

much better!

A simple example: reporting results

1. Plot marginalised posteriors for parameter combinations using “triangle plots”



this example made using corner.py (https://corner.readthedocs.io/en/latest/)

another very common code in cosmology is GetDist (https://getdist.readthedocs.io/en/latest/)

A simple example: reporting results

2. Report marginalised mean or median value of each parameter; indicate uncertainties using quantiles of posterior

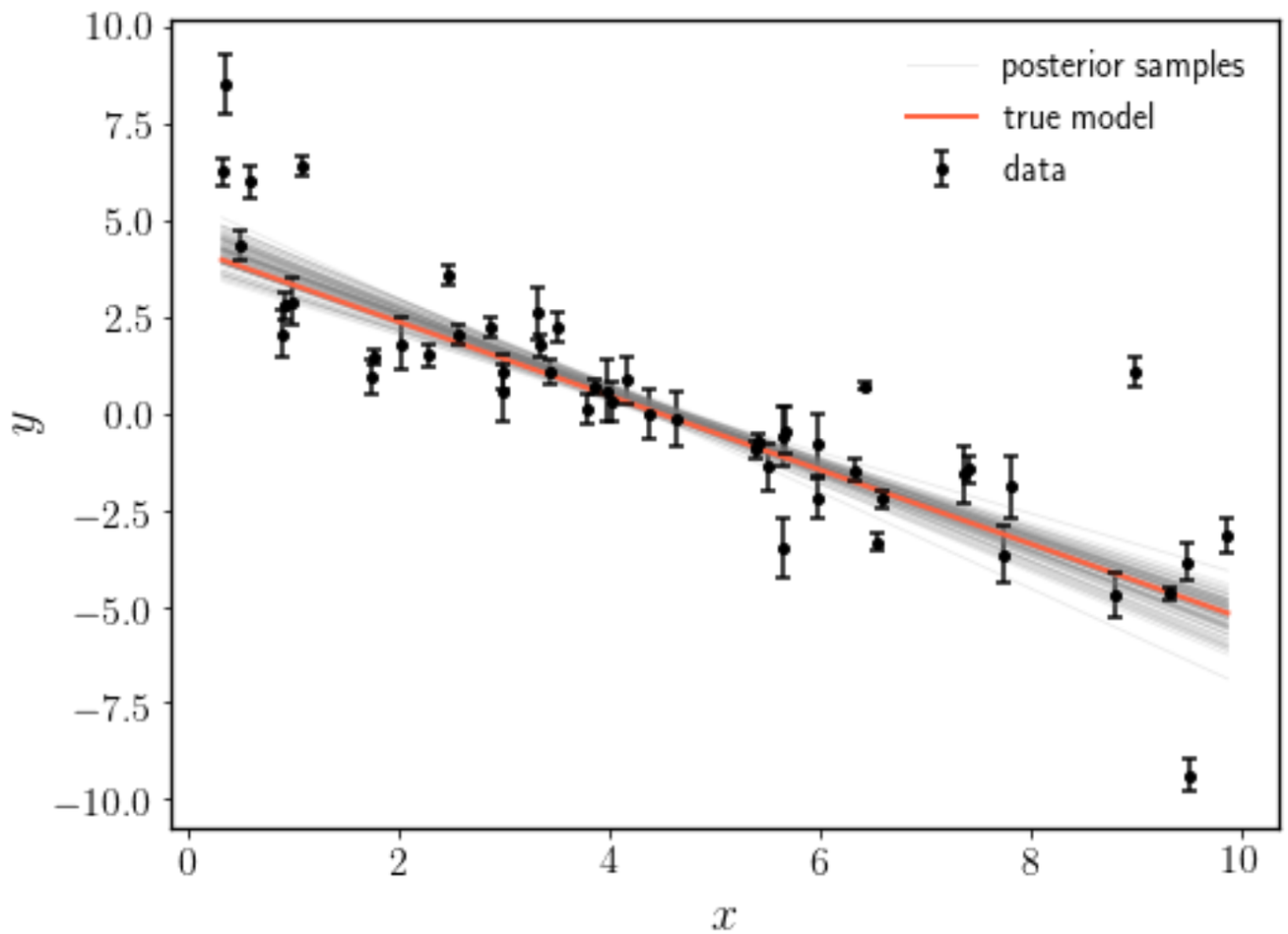
$$m = -1.00^{+0.09}_{-0.09}$$

$$b = 4.57^{+0.42}_{-0.41}$$

$$f = 0.57^{+0.09}_{-0.07}$$

A simple example: reporting results

3. Show a posterior predictive plot:



Now that we have covered the basics ...

Practical MCMC in cosmology

Lots of MCMC tools available, easy to use if you write your own likelihood

e.g., emcee https://emcee.readthedocs.io/en/stable/

based on a *affine invariant ensemble sampling* method (not M-H)
not covered in this lecture

Practical MCMC in cosmology

Lots of MCMC tools available, easy to use if you write your own likelihood

e.g., emcee https://emcee.readthedocs.io/en/stable/

In Python, ~ few lines of code and use it out of the box:

If you wanted to draw samples from a 5 dimensional Gaussian, you would do something like:

```
import numpy as np
import emcee

def log_prob(x, ivar):
    return -0.5 * np.sum(ivar * x ** 2)

ndim, nwalkers = 5, 100
ivar = 1. / np.random.rand(ndim)
p0 = np.random.randn(nwalkers, ndim)

sampler = emcee.EnsembleSampler(nwalkers, ndim, log_prob, args=[ivar])
sampler.run_mcmc(p0, 10000)
```

Practical MCMC in cosmology

Lots of MCMC tools available, easy to use if you write your own likelihood

e.g., cobaya https://cobaya.readthedocs.io/en/latest/

includes (highly-optimised) M-H sampler, nested sampler etc

can use for own likelihoods

also integrated with specific cosmological likelihoods/data and Boltzmann codes like `CAMB`, `CLASS`

Practical MCMC in cosmology

If you want to test a common model (e.g. $w_0 - w_a$ DE equation of state) against latest cosmological data:

- you probably want to use existing Boltzmann codes for theory
- you probably want to integrate with existing likelihoods (e.g from Planck)
- you want to use optimised algorithms + highly-tuned proposals for high-dimensional posterior

then use CosmoMC or MontePython (or either via cobaya)

integrates with CAMB Boltzmann code

integrates with CLASS

Note on cosmological MCMC:

If you are using `CosmoMC` or `MontePython` out-of-the-box, some things to bear in mind:

1. run many (4-8) chains in parallel
2. `CosmoMC` checks Gelman-Rubin convergence on-the-fly; `MontePython` runs for fixed number of steps
3. both update proposal matrix during the run – this means *not truly Markov!* Therefore must stop updating at some point + discard burn-in
4. make sure to correctly account for priors

Advanced methods

Importance sampling

Sometimes it is very hard to sample from the posterior we want

Instead of sampling from $p_{\text{hard}}(\theta)$, sample from a different distribution $p_{\text{easy}}(\theta)$ ...

... then reweight each sample in chain by

$$w(\theta_k) = \frac{p_{\text{hard}}(\theta_k)}{p_{\text{easy}}(\theta_k)}$$

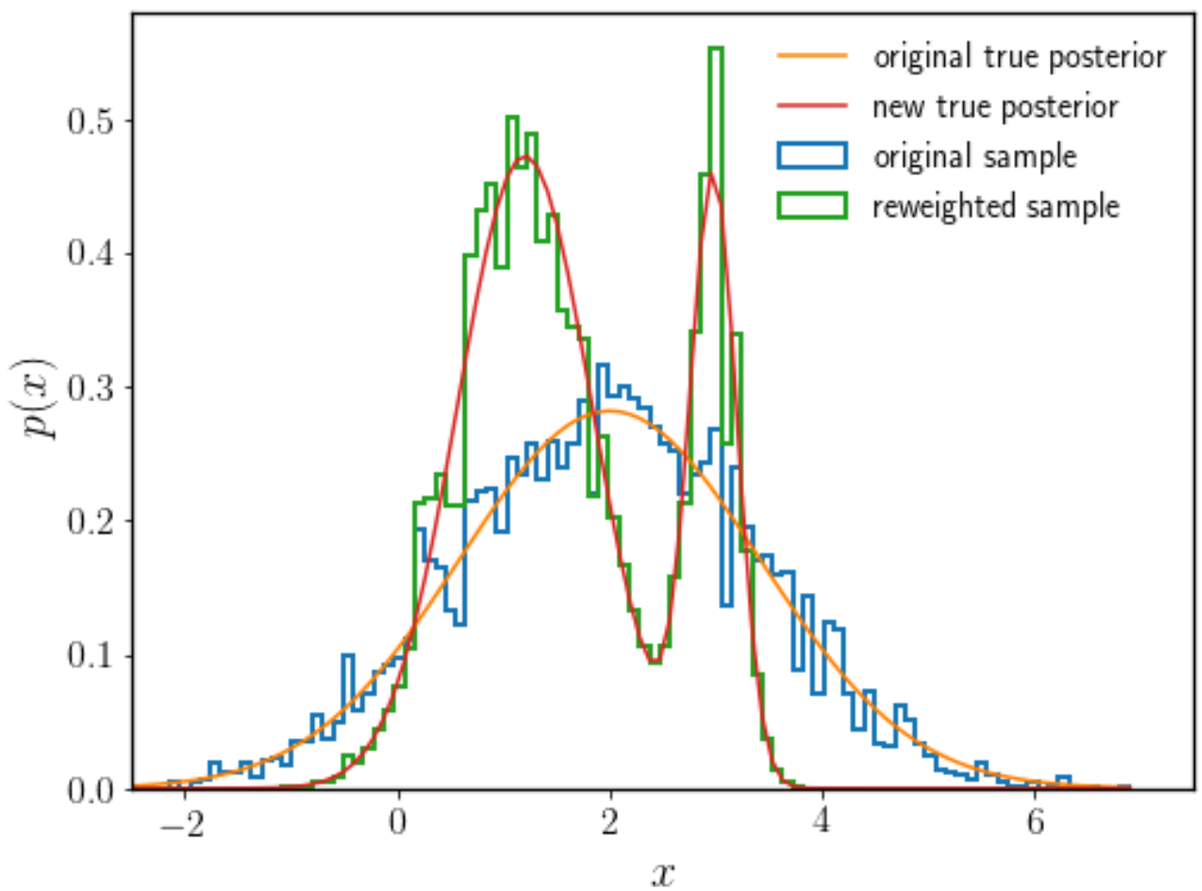
... and re-calculate means using these weights

Important: $p_{\text{easy}}(\theta)$ must have a broader basis than $p_{\text{hard}}(\theta)$

Advanced methods

Importance sampling

an example:



Advanced methods

Importance sampling

another cosmology-relevant example:

You have chains from fit of ΛCDM + m_ν model to Planck CMB data ...

... you want to fit to Planck CMB + BAO

... and re-running the chains is long and expensive

... but evaluating BAO likelihood at existing chain points is fast

$\Longrightarrow$ reweight existing chains with additional BAO likelihood

Advanced methods

Importance sampling

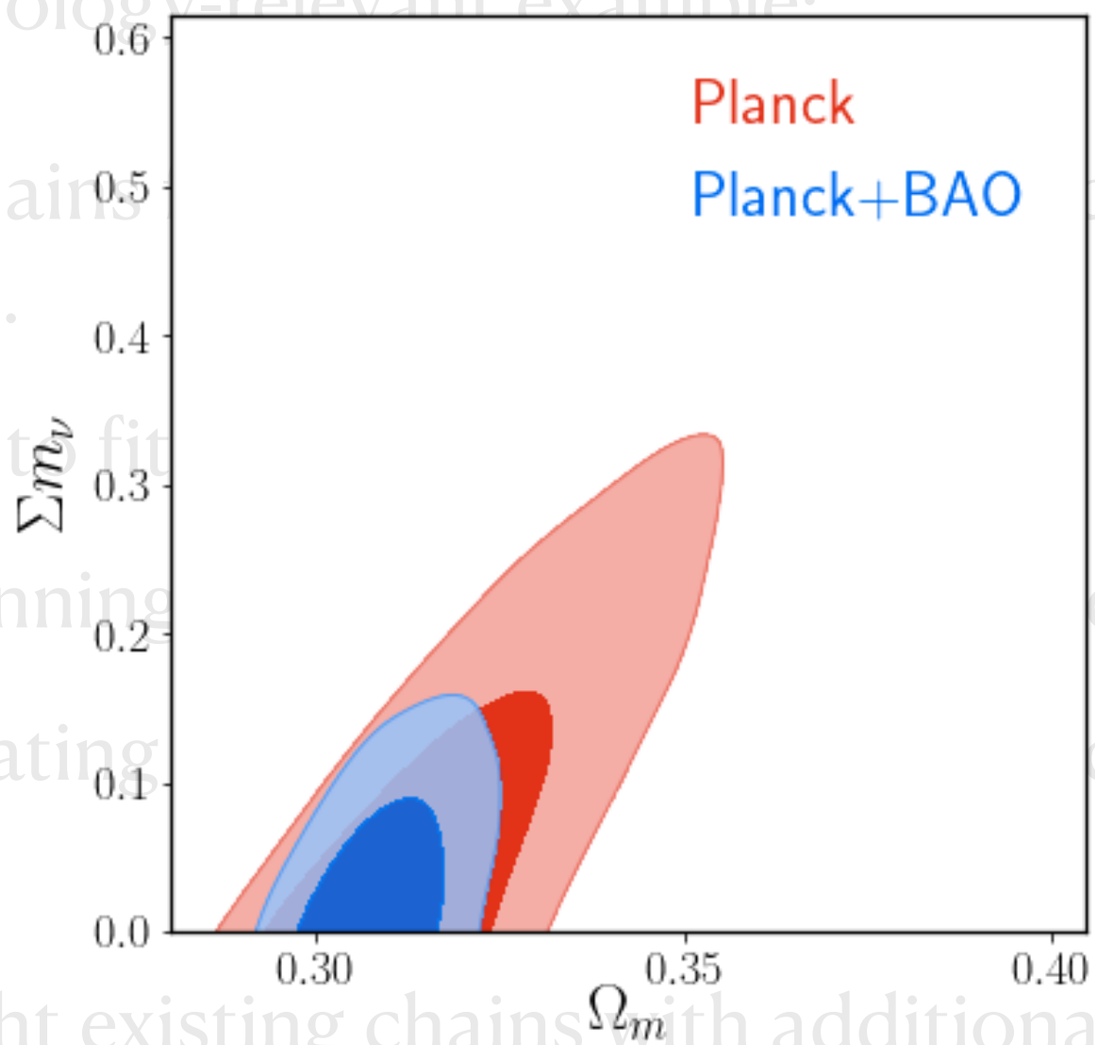
another cosmology-relevant example:

You have chains del to Planck CMB data ...

... you want to fit

... and re-running ensive

... but evaluating chain points is fast



$\Longrightarrow$ reweight existing chains with additional BAO likelihood

Advanced methods

Things not covered in this lecture ...

... but that may be very useful:

1. ensemble samplers
2. Hamiltonian MC
3. Nested sampling
4. fast-slow decomposition to speed up M-H MCMC

Useful resources

1. Hogg & Foreman-Mackey 2017 (arXiv:1710.06068) – great pedagogical introduction to MCMC, heavily used here + lots of references therein

2. `emcee`, `cobaya`, `CosmoMC`, `MontePython`, `corner`. `GetDist` codes (linked on earlier slides)

3. https://m-clark.github.io/docs/ld_mcmc/ – a list + short description of many many MCMC algorithms

4. https://gabriel-p.github.io/pythonMCMC/ – a list of public Python-based MCMC packages

5. https://github.com/seshnadathur/IntroductionToMCMC – notes and Jupyter notebook on which this lecture was based