# Markov Chain Monte Carlo

Dustin Lang

2019-09-17

# Markov Chain Monte Carlo for data analysis

- MCMC generates samples from a probability distribution function
- For data analysis: you have some *measurements* $X$, with uncertainties
- have a *model* for the thing you're measuring
- with unknown *parameters* $\theta$
- the model is *generative* – predicts measurements given parameters
- MCMC can put constraints on the parameters given your measurements

# Markov Chain Monte Carlo for data analysis

- can think of it as: *model* is a function mapping parameters to observables $t$: $m(\theta) \to t$
- and *measurement process* gives probability of observing $X$ given $t$: $P(X|t)$
- (eg, maybe $P(X|t)$ is a Gaussian with zero mean and standard deviation $\sigma$)
- Then, can chain $P(X|t) = P(X|m(\theta))$
- Usually, we fold $m$ and $P$ together to write $P(X|\theta)$

# Markov Chain Monte Carlo for data analysis

- $P(X|\theta)$ (the "likelihood") is not what you want!
- You want $P(\theta|X)$ – the "posterior" – constraints on / measurements of the parameters
- Good thing you're Bayesian!
- "posterior" $P(\theta|X) = \frac{P(X|\theta)P(\theta)}{P(X)}$
- $P(\theta)$ is the "prior"
- $P(X)$ is the "evidence", and in this setting is constant (ignored)

# Markov Chain Monte Carlo for data analysis

- We'll use MCMC to draw samples from the *posterior* probability distribution
- $P(\theta|X) = P(X|\theta)P(\theta)$
- The samples are a set of $\theta$ values drawn from your parameter constraints distribution

# What's in a name

- *Monte Carlo* – a famous casino in Monaco – alluding to the use of *randomness* in the algorithm
- *Chain* – a list (here, a list of samples / steps $\theta_1$, $\theta_2$, $\theta_3$)
- *Markov Chain* – a list of "steps" where the decision to step from $\theta_i$ to $\theta_{i+1}$ depends only on the value of $\theta_i$

- *Metropolis–Hastings* – first authors of 1953 and 1970 papers (resp) with the simplest, most commonly taught algorithm

# Using MCMC

- "Just write down the probability distributions"
- hah, the "model to predictions" part, $m(\theta) \rightarrow t$, can be *very* difficult
- eg, in cosmology, the software *CAMB* does this, in 25,000 lines of FORTRAN
- writing down the *likelihood* might take some work too!

# MCMC: the idea

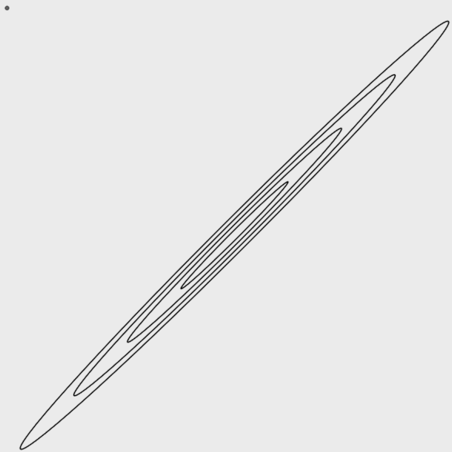
- move a "particle" or "walker" around in the parameter space $\theta$, based on the value of the probability function (for us: the posterior probability $P(\theta|X)$)
- choose new parameters according to a *proposal distribution* – often Gaussian: $\theta_{new} = \theta + \mathcal{N}(0, \Sigma)$
- compute probability at new $\theta_{new}$ and decide whether to move ("jump") to new parameters
- *always* keep improvements, *sometimes* keep decreases
- in the long run, the set of walker positions $\{\theta_1, \theta_2, \theta_3, ...\}$ are your samples

# Finally, M–H MCMC code

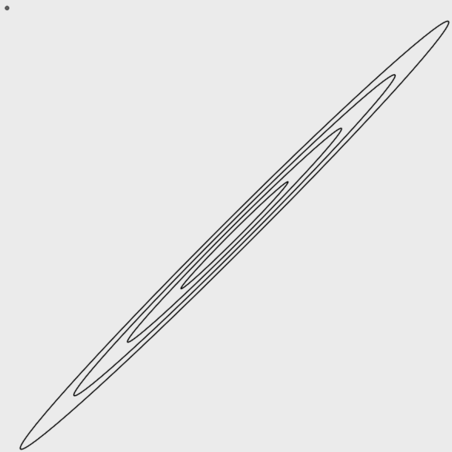
```
fun MCMC(prob_function, sample_proposal, params, Nsteps):
    chain = []
    prob = prob_function(params)
    for i in 1 to Nsteps:
        # sample from proposal distribution
        params_new = sample_proposal(params)
        prob_new = prob_function(params_new)
        # accept?
        if ((prob_new > prob) OR
            (prob_new / prob) > uniform_random()):
            params = params_new
            prob = prob_new
        chain.append(params)
    return chain
```

# Demo

Credit: Dan Foreman–Mackey
*https://speakerdeck.com/dfm/data-analysis-with-mcmc*

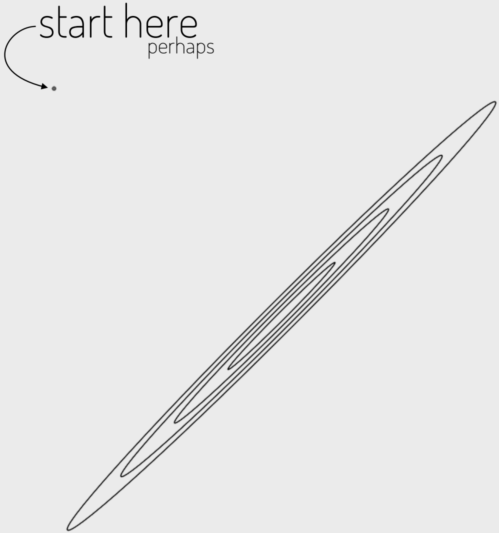


# Metropolis–Hastings

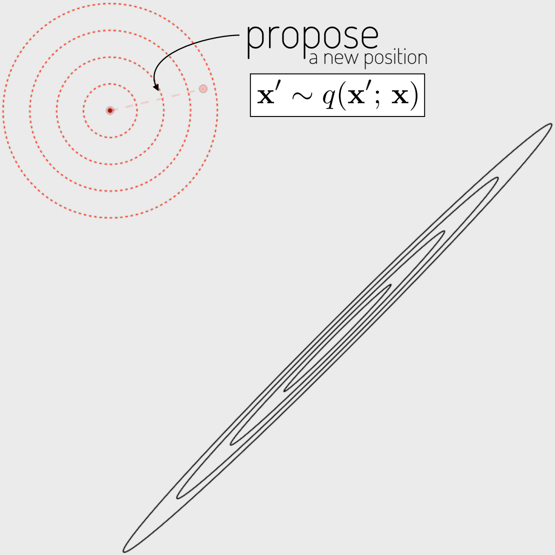


# Metropolis–Hastings

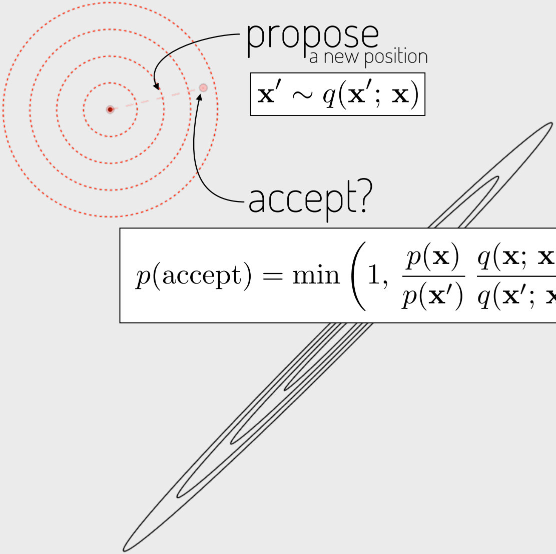
in an ideal world




Metropolis–Hastings
in an ideal world




# Metropolis–Hastings
in an ideal world

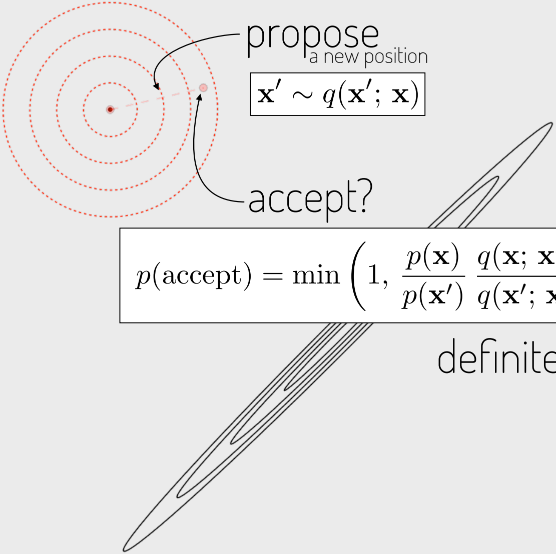



$$\mathbf{x}' \sim q(\mathbf{x}'; \mathbf{x})$$

$$p(\text{accept}) = \min\left(1, \frac{p(\mathbf{x})}{p(\mathbf{x}')}\,\frac{q(\mathbf{x}; \mathbf{x}')}{q(\mathbf{x}'; \mathbf{x})}\right)$$

# Metropolis–Hastings

in an ideal world

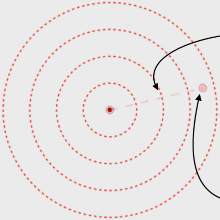



$$\boxed{\mathbf{x}' \sim q(\mathbf{x}'; \mathbf{x})}$$

$$\boxed{p(\text{accept}) = \min\left(1, \frac{p(\mathbf{x})}{p(\mathbf{x}')}\,\frac{q(\mathbf{x}; \mathbf{x}')}{q(\mathbf{x}'; \mathbf{x})}\right)}$$

# Metropolis–Hastings

in an ideal world

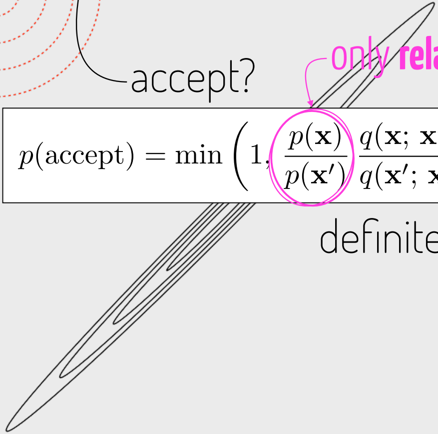


propose
a new position

$$\boxed{\mathbf{x}' \sim q(\mathbf{x}';\, \mathbf{x})}$$

accept?

only **relative** probabilities

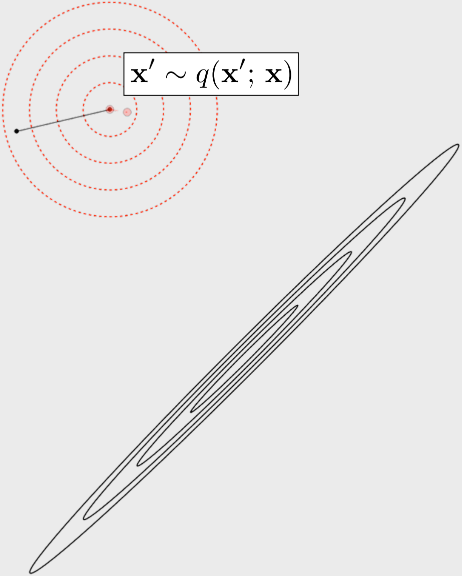


$$p(\text{accept}) = \min\left(1, \frac{p(\mathbf{x})}{p(\mathbf{x}')}\frac{q(\mathbf{x};\, \mathbf{x}')}{q(\mathbf{x}';\, \mathbf{x})}\right)$$

definitely.

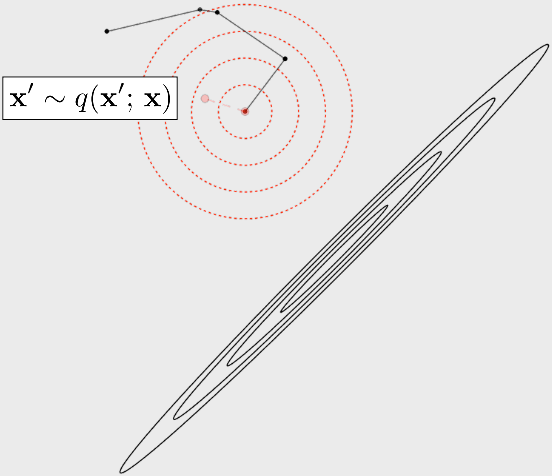
# Metropolis–Hastings

in an ideal world



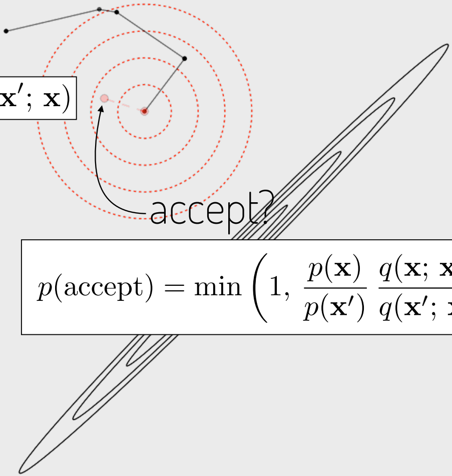

# Metropolis–Hastings

in an ideal world




# Metropolis–Hastings

in an ideal world

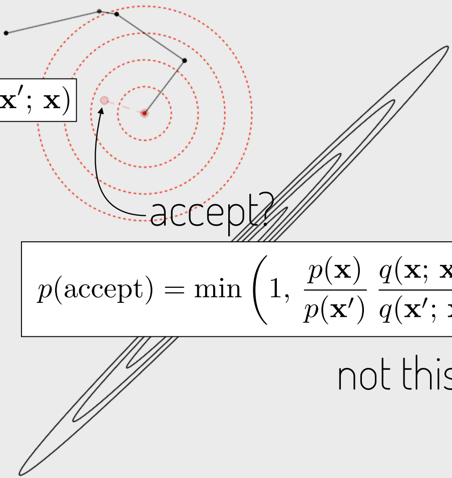



$$\mathbf{x}' \sim q(\mathbf{x}'; \mathbf{x})$$

$$p(\text{accept}) = \min\left(1, \frac{p(\mathbf{x})}{p(\mathbf{x}')} \frac{q(\mathbf{x}; \mathbf{x}')}{q(\mathbf{x}'; \mathbf{x})}\right)$$

# Metropolis–Hastings

in an ideal world



$$\boxed{\mathbf{x}' \sim q(\mathbf{x}';\, \mathbf{x})}$$

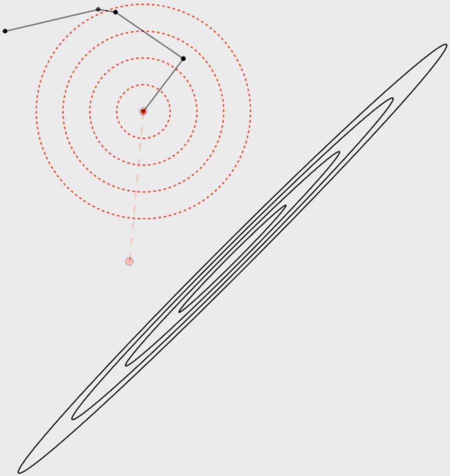
accept?

$$\boxed{p(\text{accept}) = \min\left(1,\ \frac{p(\mathbf{x})}{p(\mathbf{x}')}\ \frac{q(\mathbf{x};\, \mathbf{x}')}{q(\mathbf{x}';\, \mathbf{x})}\right)}$$

not this time.

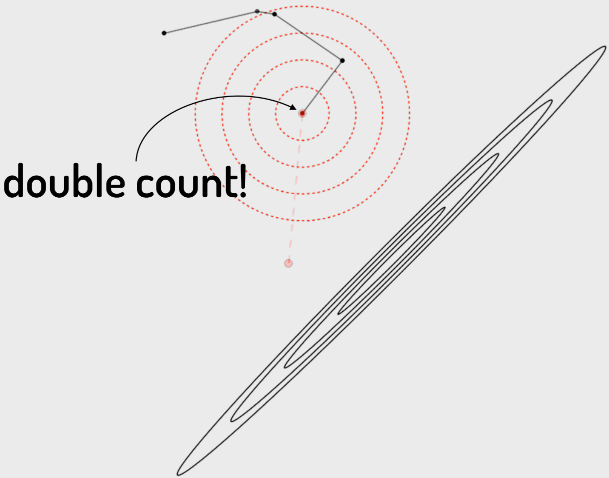
# Metropolis–Hastings

in an ideal world

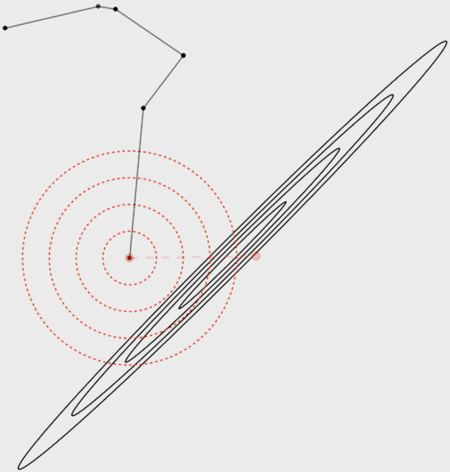


# Metropolis–Hastings

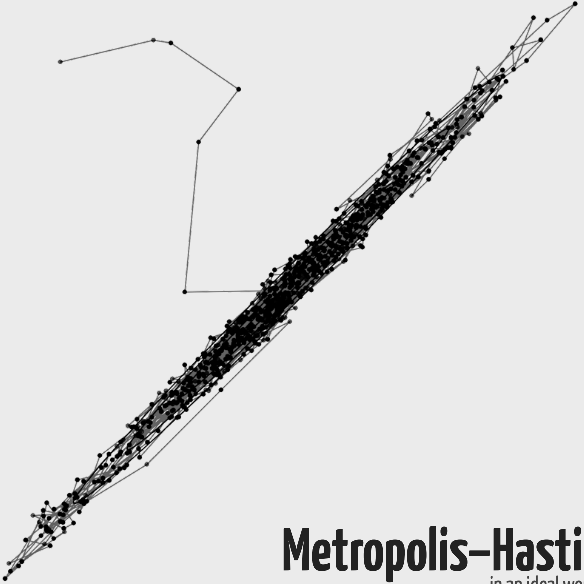
in an ideal world




# Metropolis–Hastings
in an ideal world

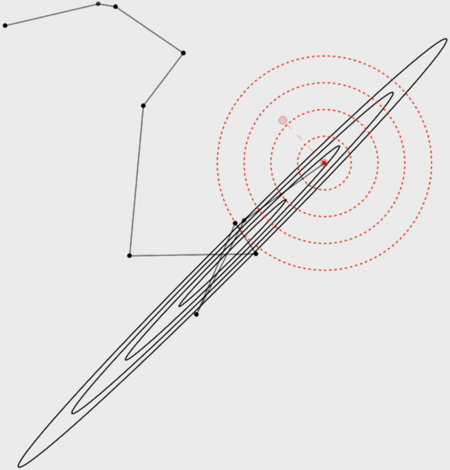


# Metropolis–Hastings
in an ideal world



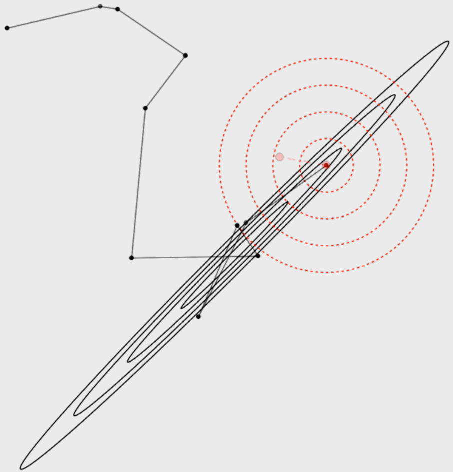
# Metropolis–Hastings

in an ideal world



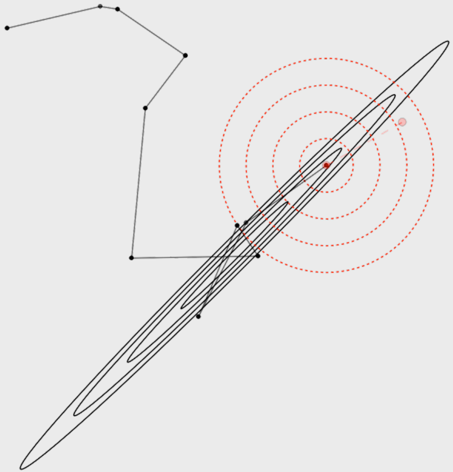
# Metropolis–Hastings

in the real world



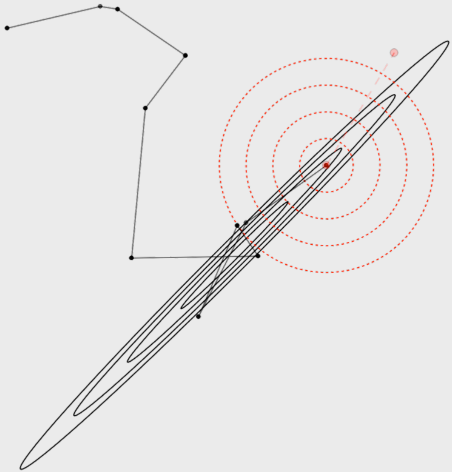
# Metropolis–Hastings

in the real world



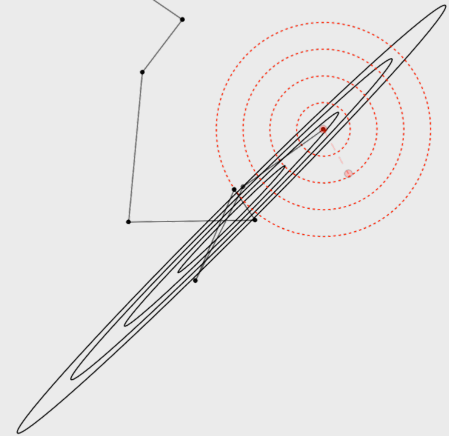
# Metropolis–Hastings

in the real world



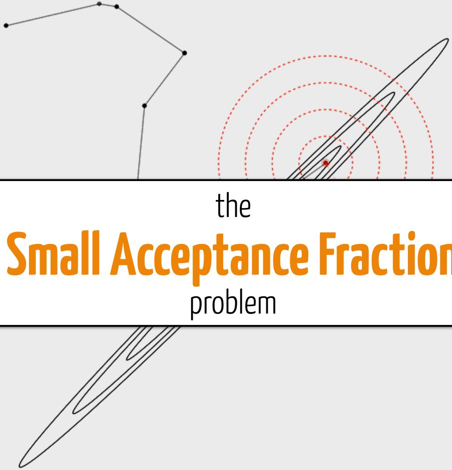
# Metropolis–Hastings

in the real world



# Metropolis–Hastings

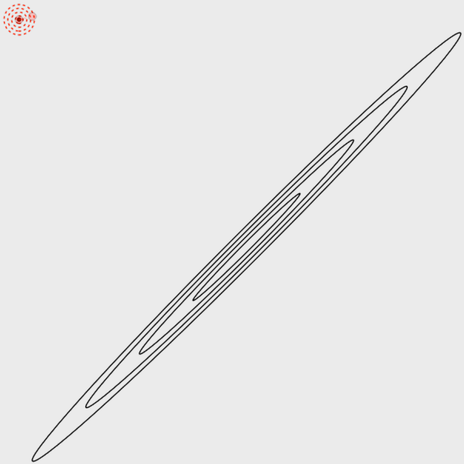
in the real world



the

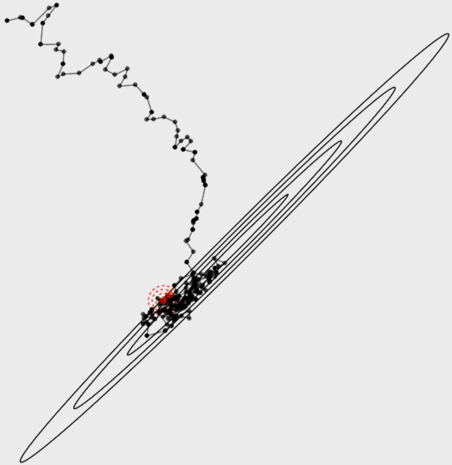
# Small Acceptance Fraction

problem

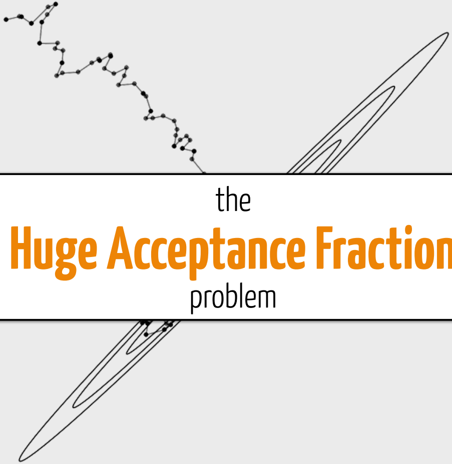


# Metropolis–Hastings

in the real world



# Metropolis–Hastings
in the real world



the

# Huge Acceptance Fraction

problem

# MCMC: summary

Things MCMC can do for you

- produce constraints on model parameters, including covariances
- include (but ignore) *nuisance parameters* in your data analysis

Things MCMC doesn't do for you

- tell you if it has converged!
- tell you if your model is good
- (easily) allow comparisons between models
- handle multi-modal distributions (try *nested sampling*)
- Dan Foreman–Mackey has good notes on advanced sampling methods