GAUTHAM NARAYAN

ASTR 496: FOUNDATIONS OF DATA SCIENCE IN ASTRONOMY

MCMC, WEEK 5

RECAP

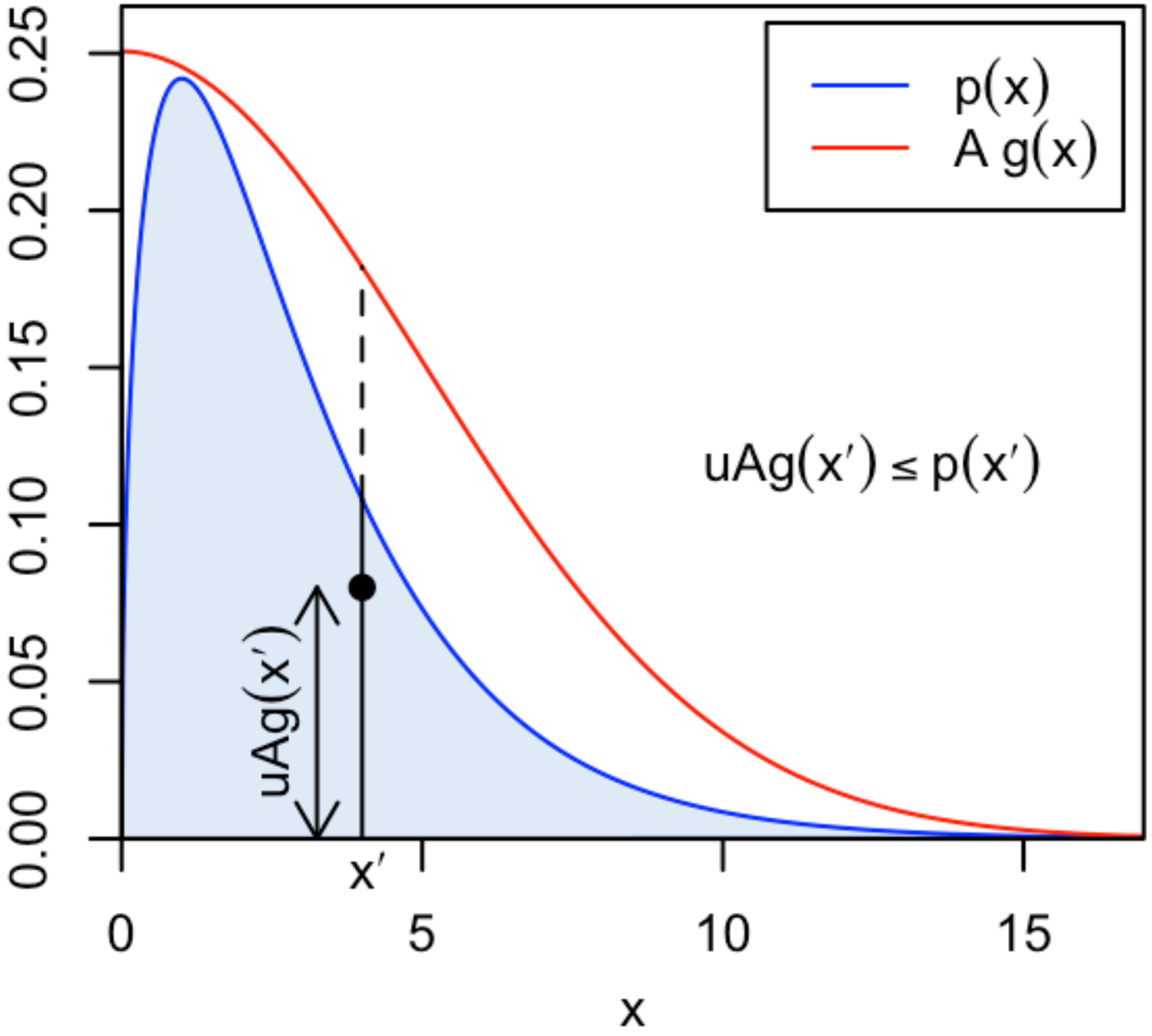
- We've come up with a model, an objective/loss/likelihood function and some priors
  - Now we actually want to evaluate the posterior $P(\theta|D)$
  - analytically is too hard, so we resort to numerical techniques
  - Option 1: Evaluate the function on some grid of parameter values - doesn't scale
  - Option 2: we draw samples (i.e. Monte Carlo)
    - **convert messy integrals to sums over the samples**

REJECTION SAMPLING

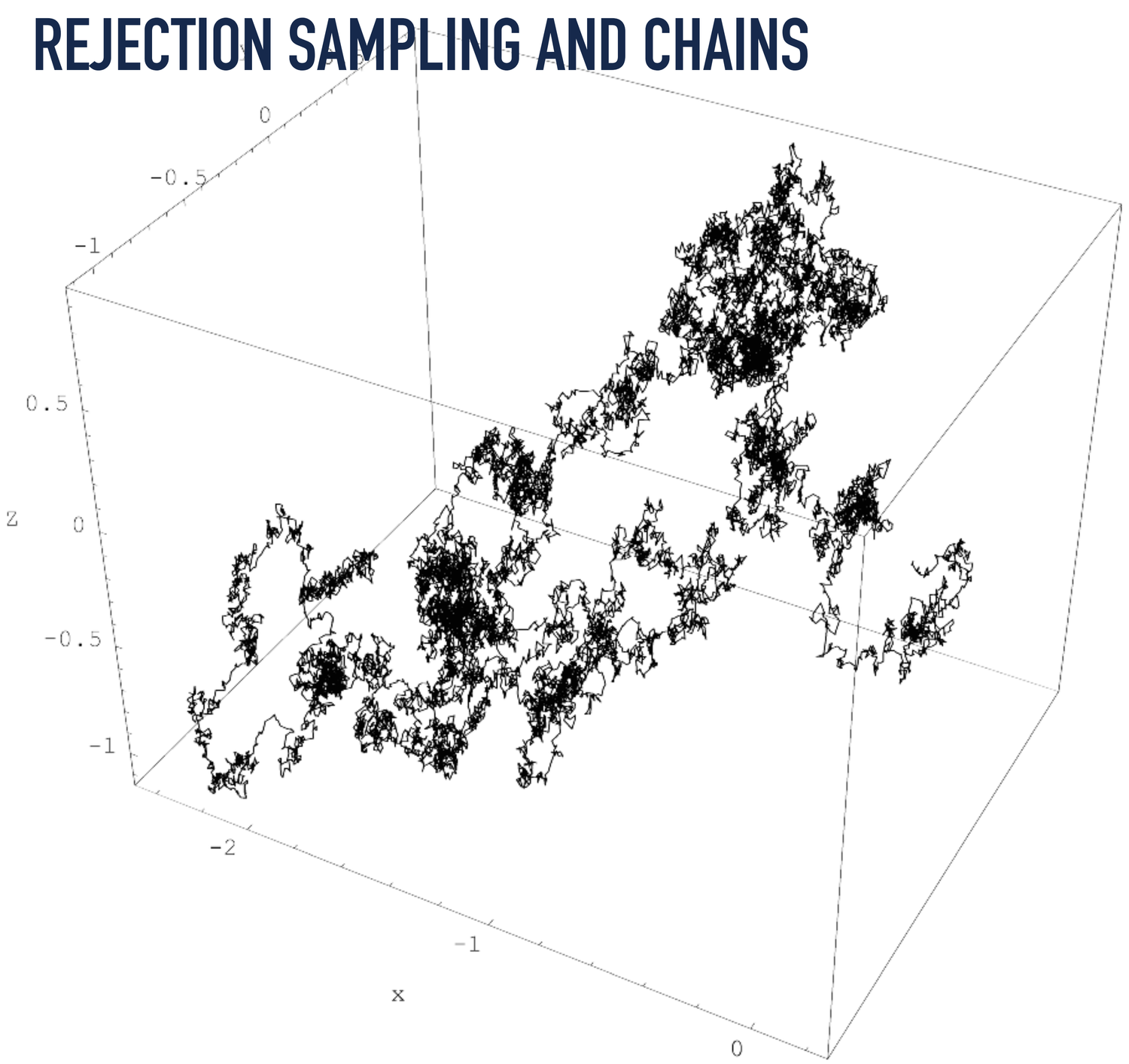
- Define an envelope function which everywhere exceeds the target PDF, $p(x)$, and can be sampled. Let this be $A.g(x)$ where A is a scaling factor and $g(x)$ is a PDF we know.

Then the algorithm is:

- while we want more samples
  - draw a random value for x from **g(x)**
  - draw u from Uniform(0,1)
  - if **u** <= **p(x)/A*g(x)**, keep the sample x
  - otherwise, reject x



REJECTION SAMPLING AND CHAINS



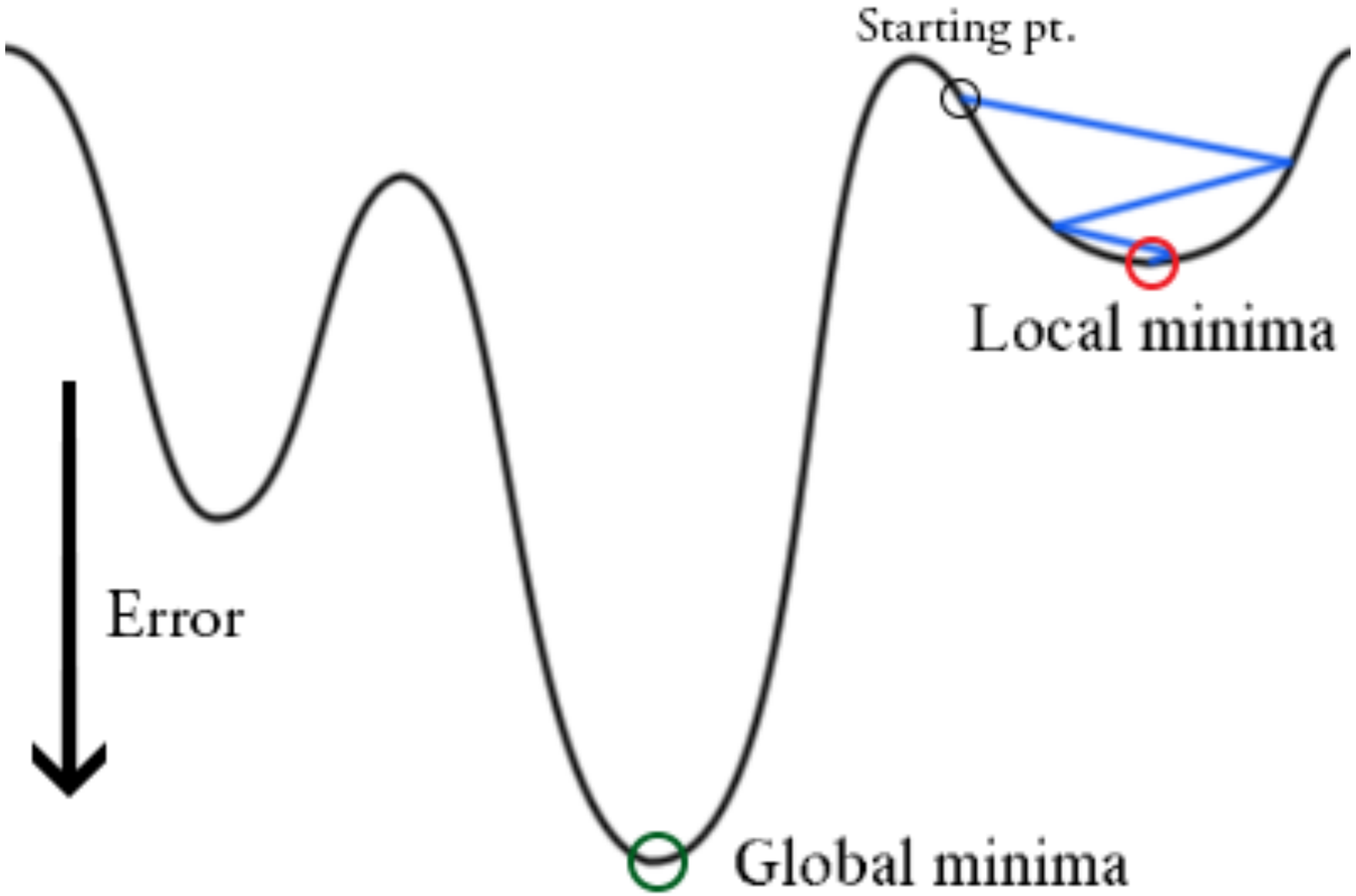
- Start sampling in your parameter space following a prior distribution at ***x***
- Compute the likelihood at this location ***p(x)***
- Move to a new location ***x'***
- Compute the likelihood at the new location ***p(x')***
- If it's higher at the new location, keep the new sample ***x'***
- If it's lower, check against a random Uniform number draw ***u***, and maybe keep the sample
- The list of all samples is a **chain** - but this isn't enough to make a *Markov chain*

This comparing against a random number may seem… random

But the stochasticity is the point!

Stochasticity allows us to explore the whole surface but spend more time in interesting spots

This will avoid the issues with local minima that plagued us with optimizers



RECAP

- We've come up with a model, an objective/loss/likelihood function and some priors
  - Now we actually want to evaluate the posterior P(θ|D)
  - analytically is too hard, so we resort to numerical techniques
  - Option 1: Evaluate the function on some grid of parameter values - doesn't scale
  - Option 2: we draw samples (i.e. Monte Carlo)
    - **convert messy integrals to sums over the samples**
- Simple Monte Carlo is wasteful because we don't want to draw samples over all parameter space
- one way to make it more efficient was make **samples that are correlated with each other**
  - keep sampling in regions of high probability, don't sample in regions with low probability
  - a) If we are lucky enough to have nice functions, we can use **inverse transform sampling**
  - b) alternately we can use **rejection sampling**
  - Markov Chains are series of samples where every sample **only** depends on the previous one
  - **If we build our Markov Chain using rejection sampling, we're doing Markov Chain Monte Carlo (the particular strategy here is called Metropolis-Hastings)**

MARKOV CHAIN PROPERTIES

- **Ergodic** - given enough time* the entire parameter space will be sampled

$$X_1, X_2, \ldots, X_n, X_{n+1}, \ldots$$

X_{n+1} *depends* *only* *on* X_n

(and not on $X_1, X_2, \ldots, X_{n-1}$*)*

MARKOV CHAIN PROPERTIES

- **Stationary** - As long as the Markov chain is **positive recurrent** (i.e. you can get to any parameter in a finite number of steps) and is **irreducible** (you can get to every parameter value from every other parameter value) then it has another nice property - it is **stationary**

$$X_1, X_2, \ldots, X_n, X_{n+1}, \ldots, X_m, X_{m+1}, \ldots$$

$$P(X_{n+1} | X_n) = P(X_{m+1} | X_m)$$

$$\sim time\ invariant$$

MARKOV CHAIN PROPERTIES

- **Reversible** - If the probability of getting from point x to x' is the same as the probability of getting from x' to x, then the chain is reversible. This happens if a condition called **detailed balance is satisfied**

$$X_1, X_2, \ldots, \underbrace{X_n, X_{n+1}, X_n}_{=x}, \ldots$$

$$P(X_{n+1} \mid X_n = x) = P(X_{n+2} = x \mid X_{n+1})$$

$$\sim \textit{time-reversal invariant}$$

DETAILED BALANCE

- **Detailed balance: the probability of getting from point x to x′ is the same as the probability of getting from x′ to x - how do we go about getting this property?**
- The Metropolis-Hastings algorithm consists of these steps:
  - given some x, $p(x)$
  - and a transition matrix probability $T(x'|x)$, draw a proposed value for x'
  - compute probability $p(x')$
  - draw a random number **u** between 0 and 1 from a uniform distribution; if it smaller than $p(x')$, then accept x'
- if x' is accepted added it to the chain, if not, add x' to the chain.
- ***This process is NOT stationary*** - $T(x'|x)$ and $T(x|x')$ don't have to be the same in principle

BUILDING A CHAIN OF SAMPLES

- The probability of an arbitrary point from such a chain being located at x' is (marginalizing over the possible immediately preceding points)

$$p(x') = \int dx\, p(x)\, T(x' \mid x)$$

- where $T(x'|x)$ is the transition probability of a step from x to x'.

- **We want to have detailed balance**

$$p(x)T(x' \mid x) = p(x')T(x \mid x')$$

REARRANGING

- We'll break the transition **T(x'|x)** into two steps:
- A proposal, **g(x'| x)** and
- An acceptance ratio, **A(x'|x)**
- i.e.

$$T(x' \mid x) = A(x' \mid x)g(x' \mid x)$$

$$\frac{A(x' \mid x)}{A(x \mid x')} = \frac{p(x')g(x \mid x')}{p(x)g(x' \mid x)}$$

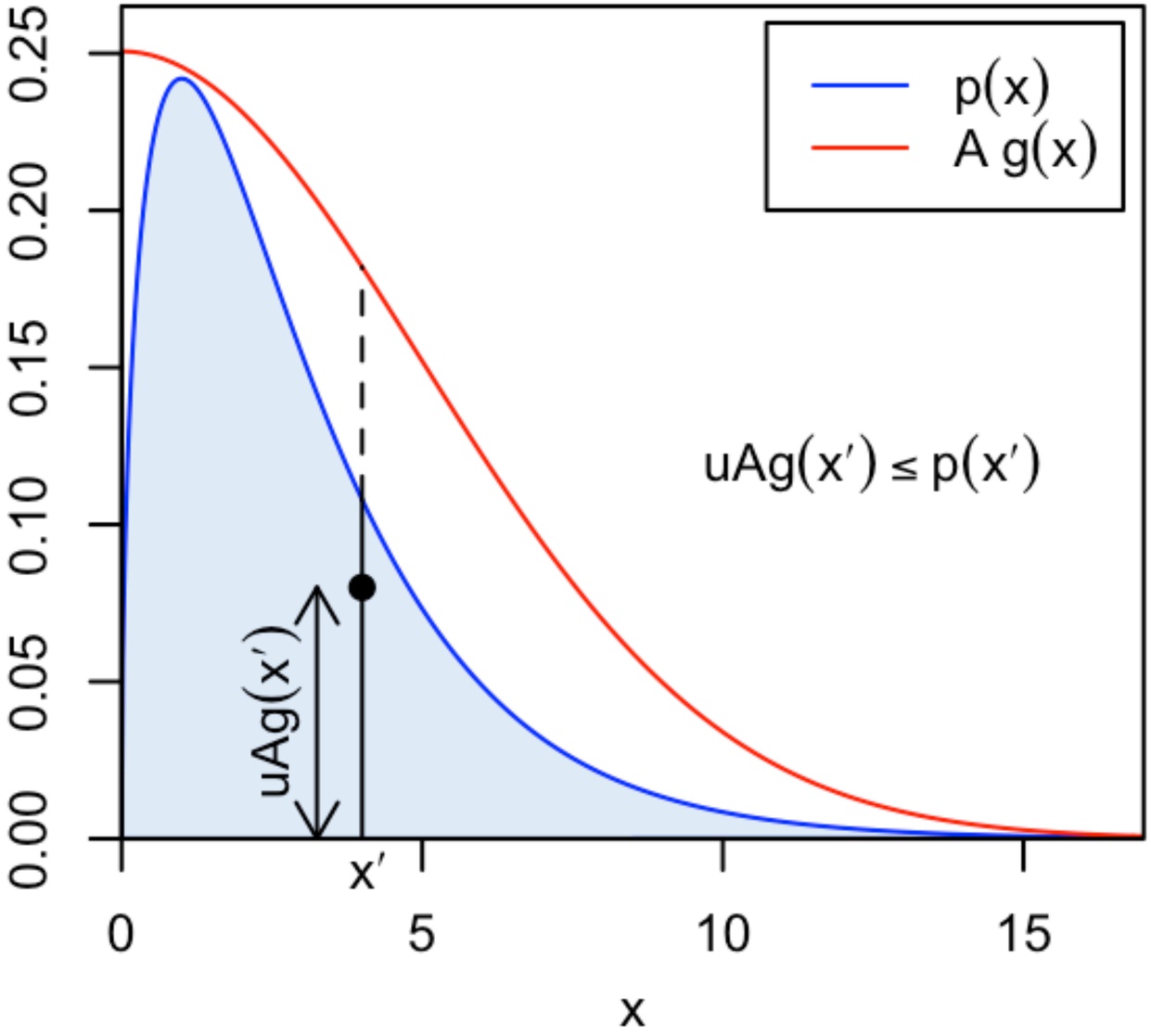
HEY THIS IS JUST REJECTION SAMPLING

- This notation of **p(x)**, **A(x)**, **g(x)** again is deliberate - this is just rejection sampling at each position in the chain.

$$\frac{A(x' \mid x)}{A(x \mid x')} = \frac{p(x')g(x \mid x')}{p(x)g(x' \mid x)}$$

- The probability of accepting a proposed step from *x* to *x′* then

$$A(x', x) = \min\left[1, \frac{p(x')g(x \mid x')}{p(x)g(x' \mid x)}\right]$$



HEY THIS IS JUST REJECTION SAMPLING

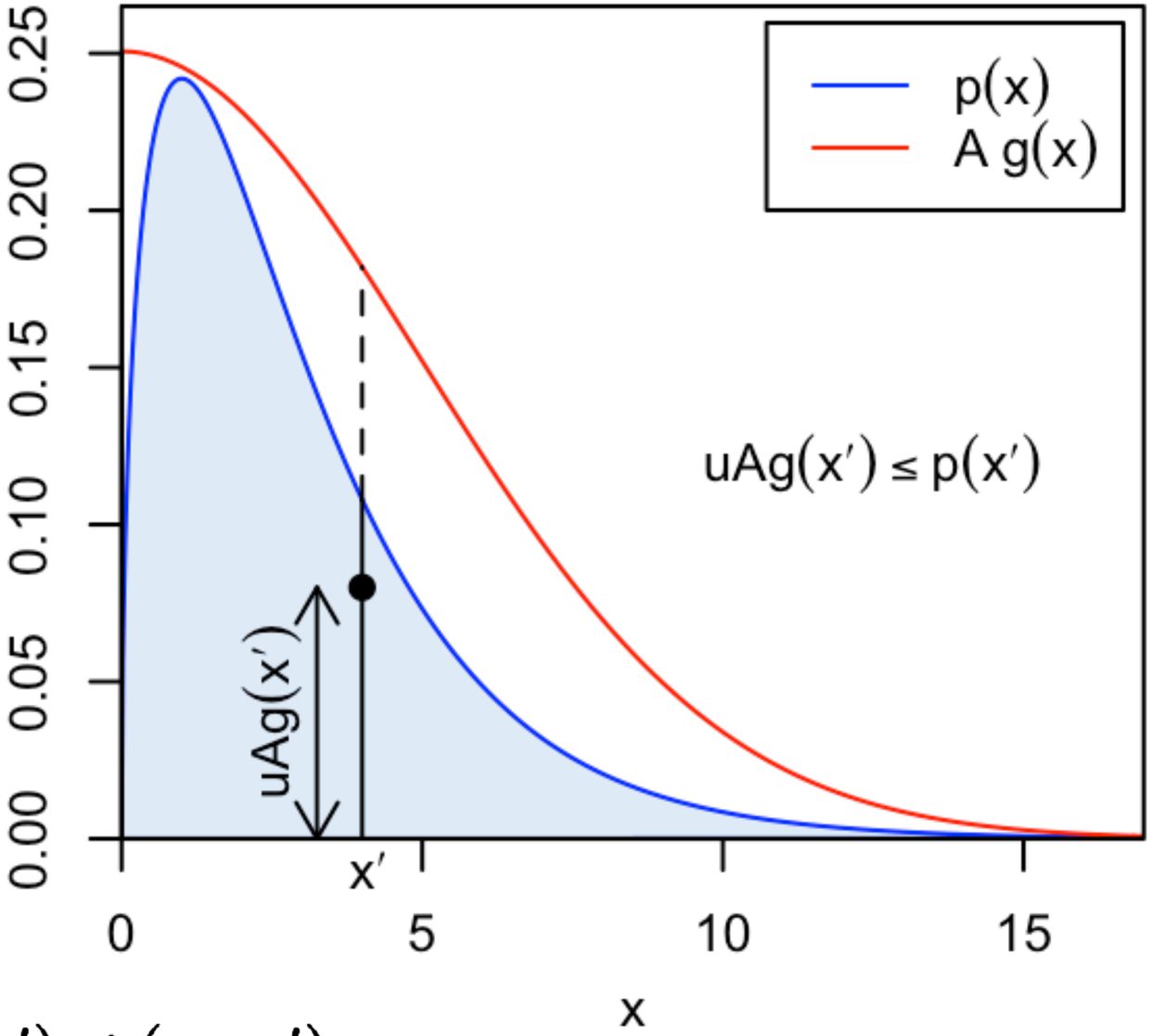
So if we identify:

$p(x)$ is the posterior density (probability of being at x, if we're sampling P properly)

$g(x'|x)$ is the proposal distribution (probability of attempting a move to x' from x)

$A(x', x)$ is the probability of accepting the proposed move

With this definition of A, detailed balance is automatically satisfied, and we can **guarantee our chain is time-reversal invariant**



$$p(x)g(x' \mid x)A(x', x) \equiv p(x')g(x \mid x')A(x, x')$$

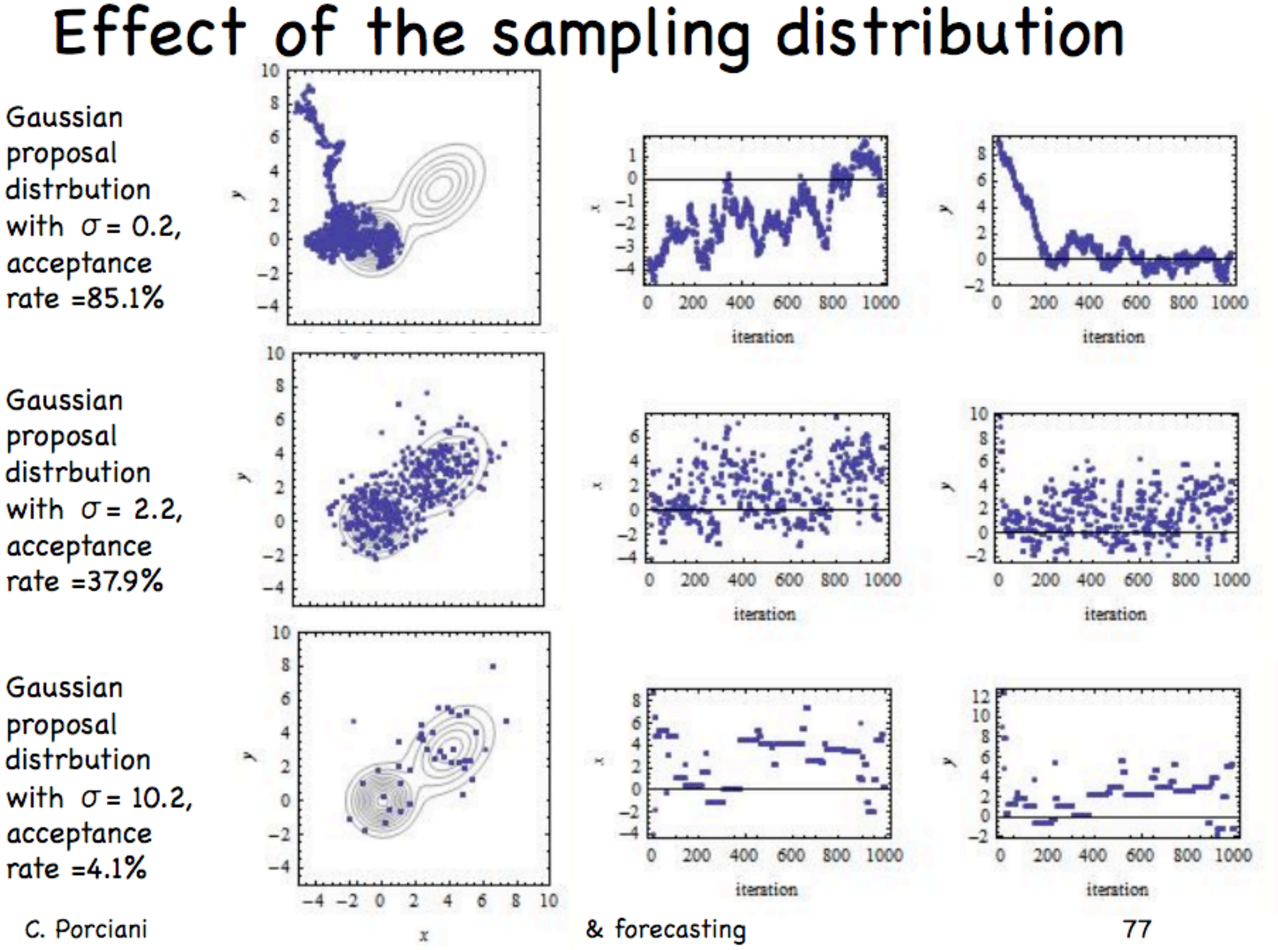
DETAILED BALANCE GIVES YOU REVERSIBLE MARKOV CHAINS

$$p(x)g(x' \mid x)A(x', x) \equiv p(x')g(x \mid x')A(x, x')$$

IMPORTANT:

- Even if a step is rejected, we still keep ***a*** sample (i.e. the original state ***x***, without moving).

- **The difficulty of finding a temptingly better point is important information!**



MCMC PROPERTIES

- The equilibrium state of this chain is the stationary distribution of samples we desire - the posterior
- If the chain isn't in equilibrium, well tough luck. You're guaranteed it'll get there eventually… you need to tune your chain so it gets to equilibrium reasonably **efficiently**.
- So we need some metrics to diagnose if:
  - the chain has converged to the posterior distribution - **i.e. is the chain stationary?**
  - the chain provides enough effectively independent samples to characterize the posterior

THIS BOILS DOWN TO:

- A) How do you choose **g(x, x')**
  - This effectively determines what the algorithm's name is - many options…
  - Gibbs sampling - move along one parameter at a time but you know conditional distribution of each variable and always accept every sample
  - Simulated annealing - start one chain exploring, but have it's step size depend on a "temperature" of the system that cools - initially starts in high temperature - bad moved accepted, but eventually cools, and moves that don't increase the posterior are less likely to be accepted
  - Parallel tempering - start many parallel chains at once, run some hot (large steps) and some cold (https://www.youtube.com/watch?v=J6FrNf5__G0&list=PLgArfv_fOU5dwjeP_57NO_jnRJ7cWCe6J - you might want to mute this)

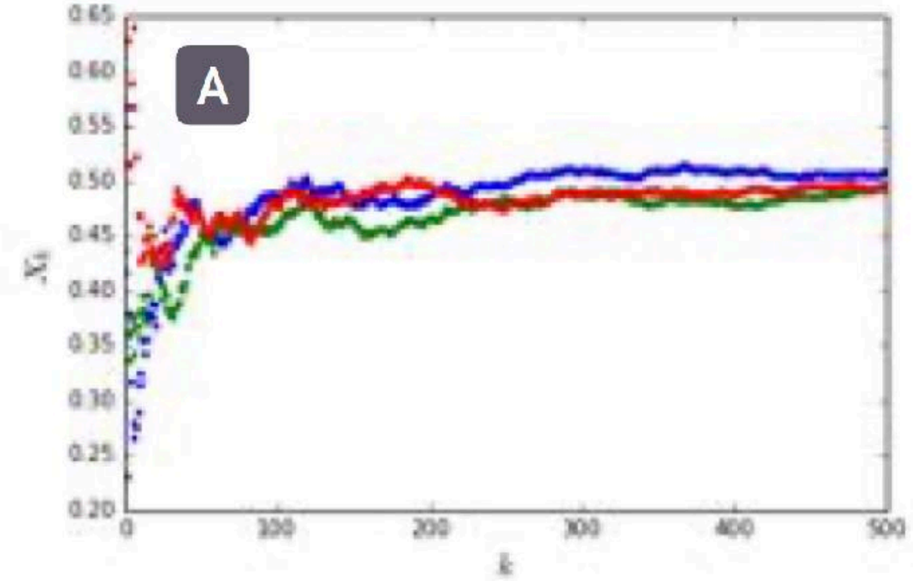
MCMC FLAVORS

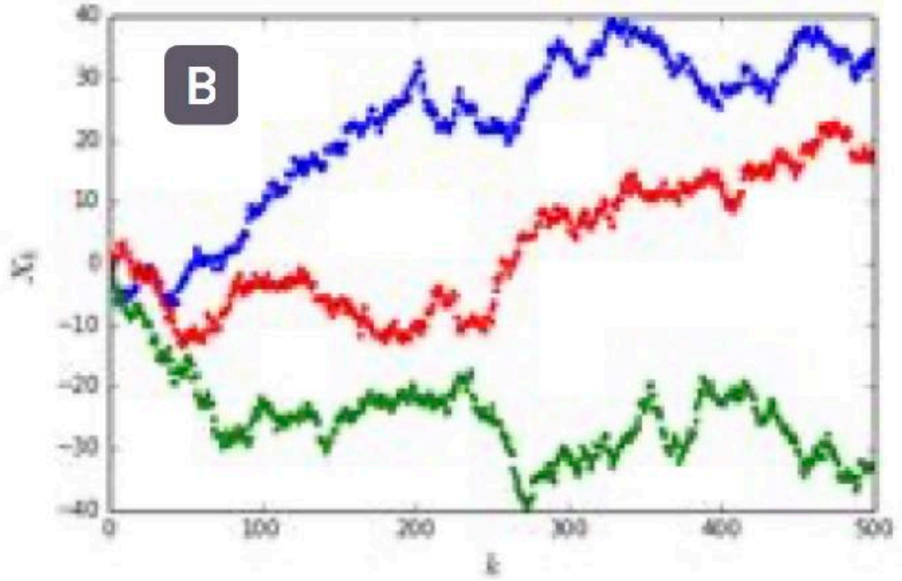
- http://chi-feng.github.io/mcmc-demo/
  - Fiddle on your own time
  - Sampling efficiency often starts as the primary concern - your code needs to work in a reasonable amount of time
  - But once you've got something acceptable, the primary concern becomes **Diagnosis** - How do you know that your MCMC is providing you reasonable, independent samples drawn from the posterior

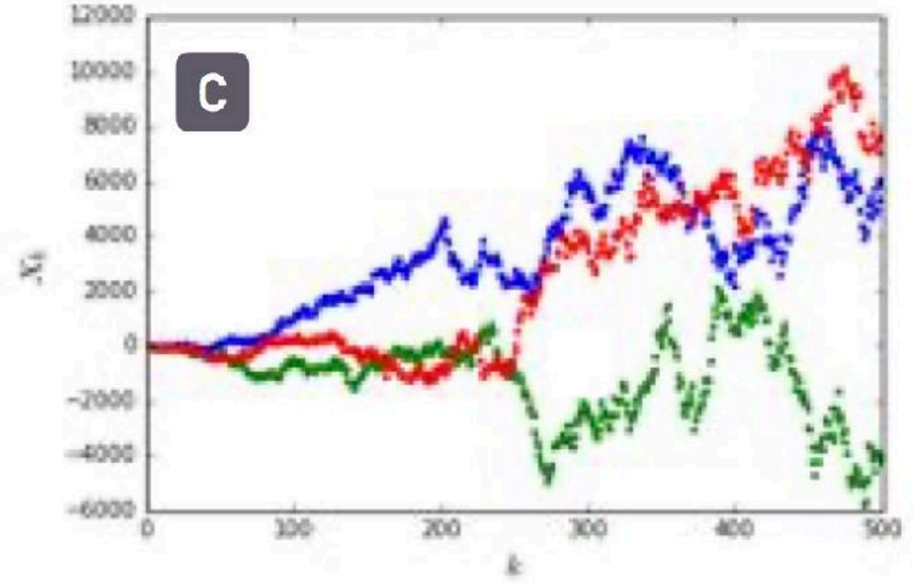
MCMC DIAGNOSTICS

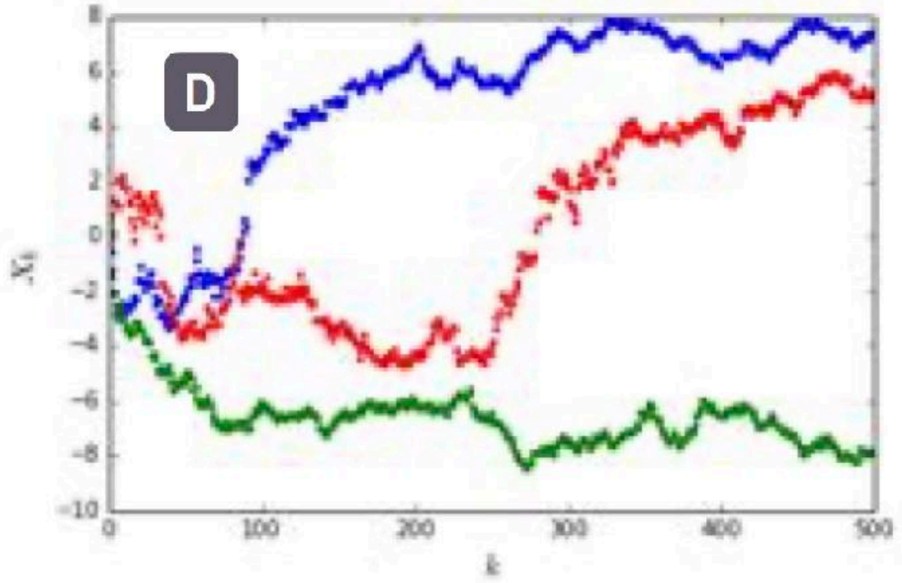
- What would make us confident of convergence?
  - Is the chain stationary?
  - Do independent chains started from overdispersed positions end up in the same stationary state?
- There's also a trick here - we wanted i.i.d samples from the posterior, but to make our Markov chain efficient, we violated the "independence" criteria a little
  - Each sample of the chain depends on the previous sample through **g(x, x')**
- So how do we guess the number of independent samples?
  - Check how well the chain appears to exploring the distribution
  - Compare the autocorrelation length scale of the samples with the chain length

ACTIVITY: STATIONARY / NON-STATIONARY / NON-REVERSIBLE?



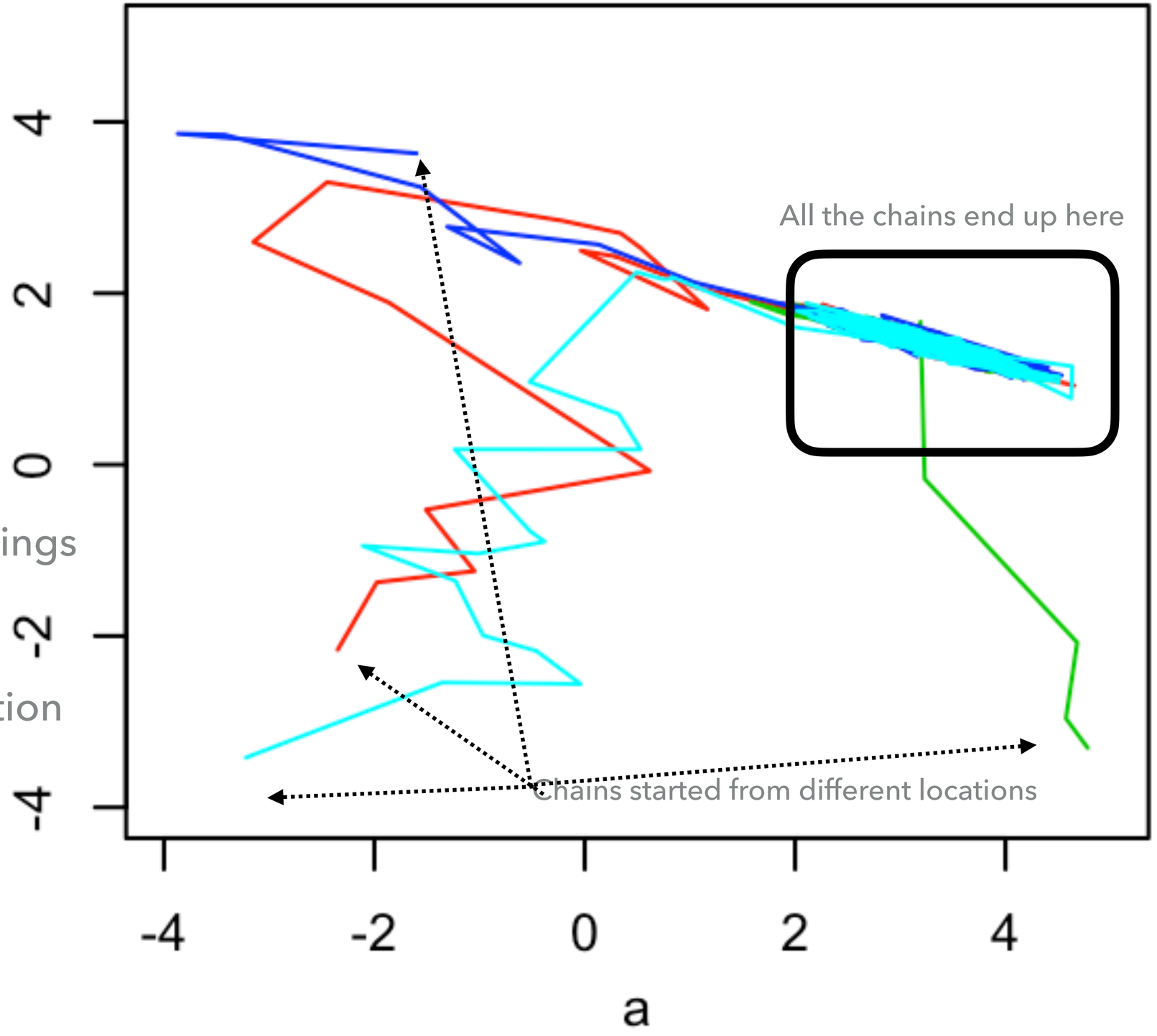






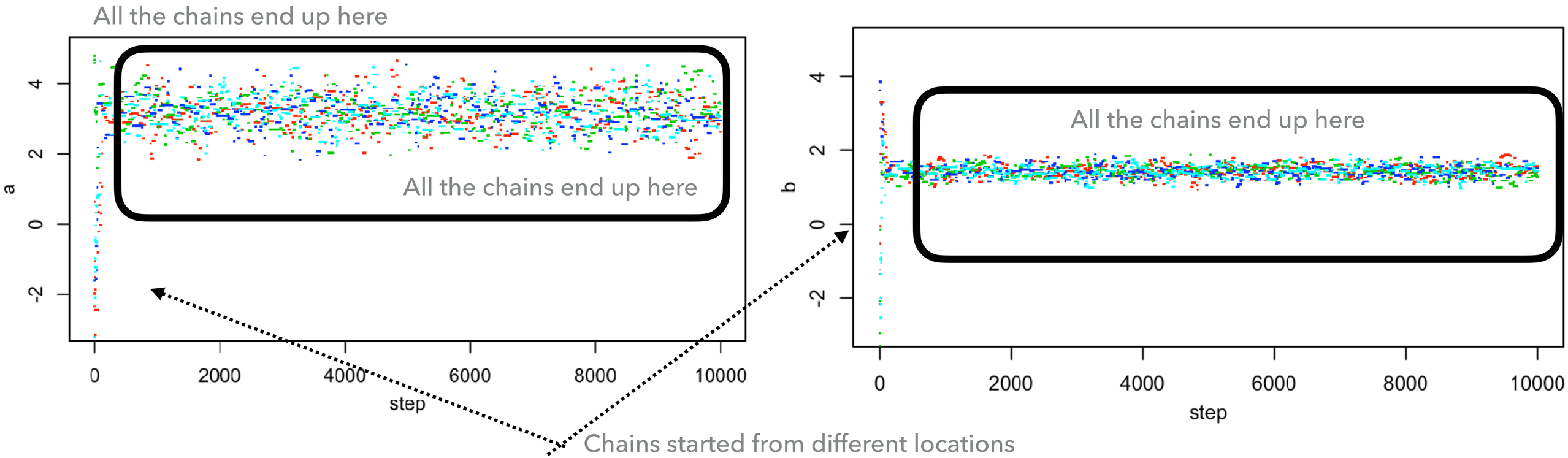
Setup: here's some chains from a simple linear problem

I've started 4 Metropolis-Hastings chains (colors) from different locations, and had them all sample the posterior distribution



MCMC DIAGNOSTICS

- So you can tell by eye when things are behaving reasonably - how do we take that intuition and express it with mathematics



- **Gelman-Rubin convergence statistic:** This approach tests the similarlity of independent chains intended to sample the same PDF. To be meaningful, they should start from different locations and burn-in should be removed.

GELMAN-RUBIN

- For a given parameter, θ, the R statistic compares the variance across chains with the variance within a chain.
  - Intuitively, if the chains are random-walking in very different places, i.e. not sampling the same distribution, R will be large
- **Step 1:** Get the variance between each chain's estimate of the parameter and the global estimate, where θj^{-} is the average θ for chain j and θ^{-} is the global average.

$$B = \frac{n}{m-1} \sum_{j} \left(\bar{\theta}_j - \bar{\theta} \right)^2$$

GELMAN-RUBIN

- For a given parameter, θ, the R statistic compares the variance across chains with the variance within a chain.
  - Intuitively, if the chains are random-walking in very different places, i.e. not sampling the same distribution, R will be large
- **Step 2:** Get the average variance of the individual-chain variances for θ, where $s^2{}_j$ is the estimated variance of θ within chain j.

$$W = \frac{1}{m} \sum_j s_j^2$$

GELMAN-RUBIN

- For a given parameter, θ, the R statistic compares the variance across chains with the variance within a chain.
  - Intuitively, if the chains are random-walking in very different places, i.e. not sampling the same distribution, R will be large
- **Step 3:** Get the overall estimate for the variance of θ

$$V = \frac{n-1}{n}W + \frac{1}{n}B$$

GELMAN-RUBIN

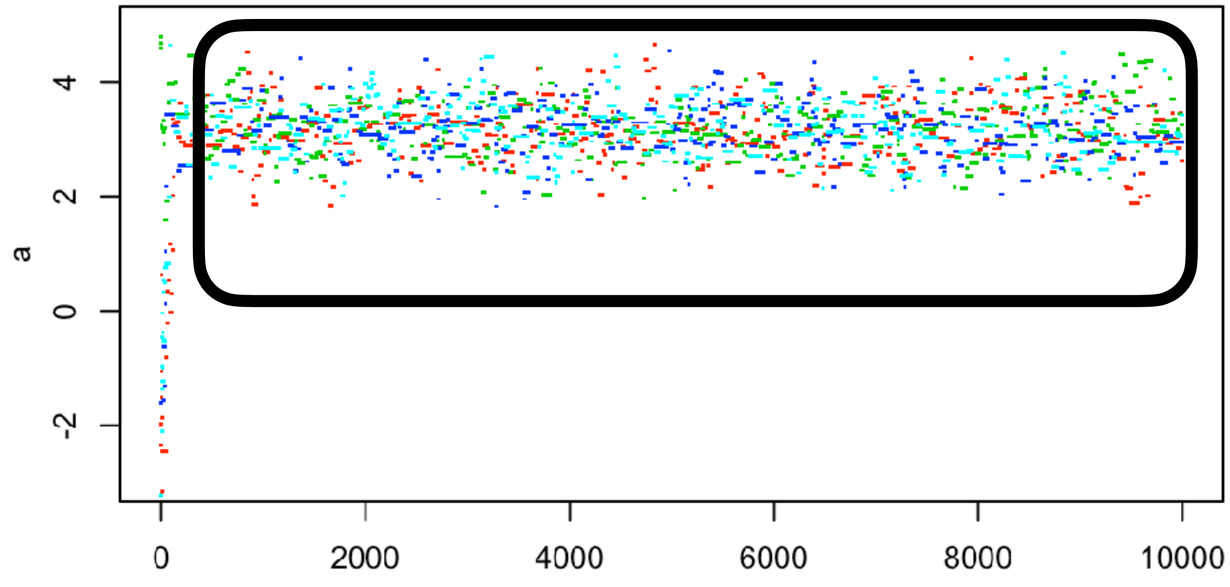
- For a given parameter, θ, the R statistic compares the variance across chains with the variance within a chain.
  - Intuitively, if the chains are random-walking in very different places, i.e. not sampling the same distribution, R will be large
- **Step 4:** Finally

$$R = \sqrt{\frac{V}{W}}$$

- We'd like to see $R \approx 1$ (e.g. $R < 1.1$ is often used). Note that this calculation can also be used to track convergence of combinations of parameters, or anything else derived from them. **Iff $R \approx 1$, then the chains are described as well mixed.**

CORRELATION TESTS

- We also gave up a nice property of Simple Monte Carlo moving to Markov Chain Monte Carlo.
  - Our samples are now correlated - i.e. you literally took a position and adjusted it by a small amount.
  - This means that when you request 10,000 samples from Metropolis Hastings, you aren't actually getting 10,000 i.i.d samples precisely **because they aren't independent.**



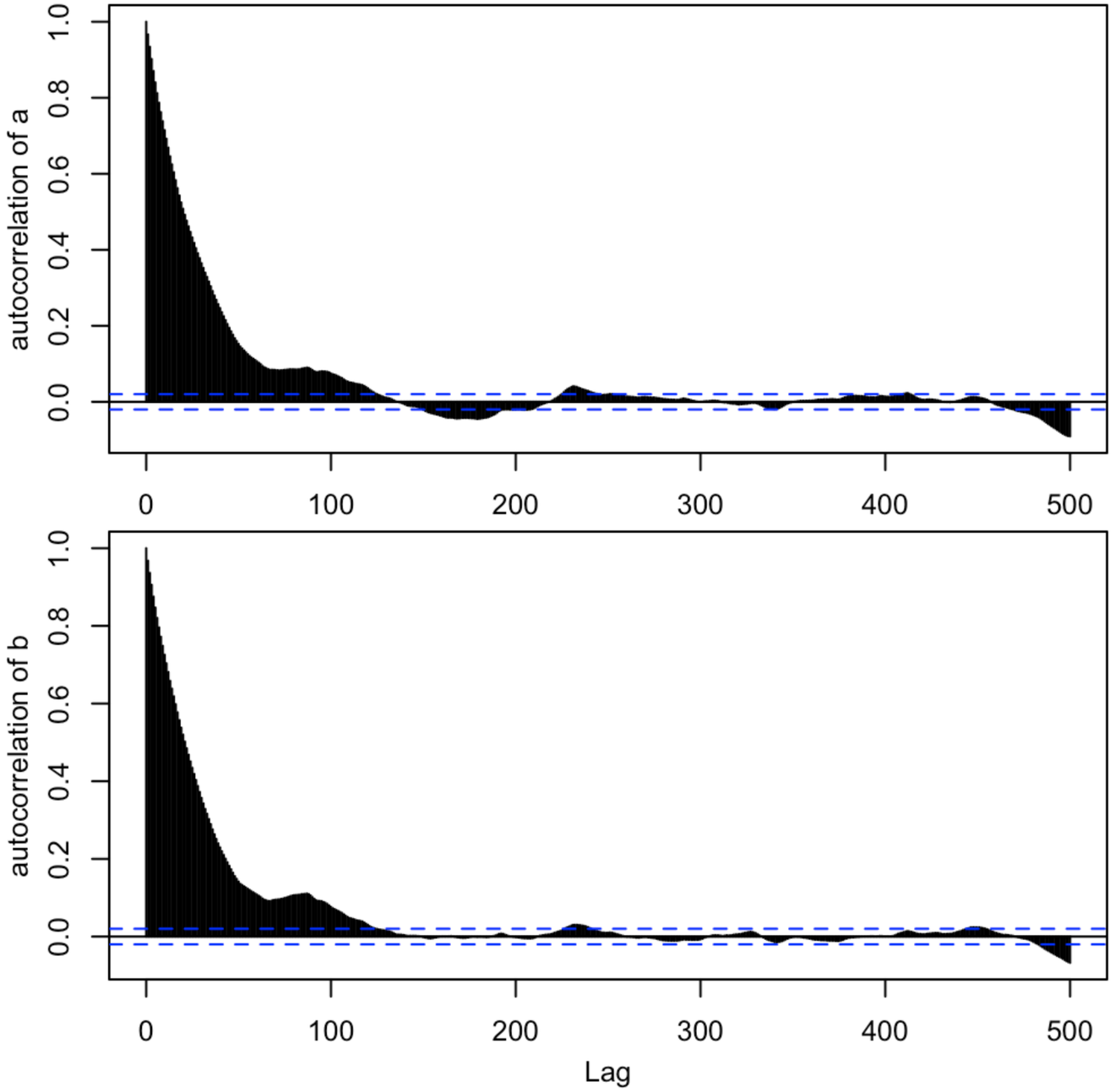
CORRELATION TESTS

- The autocorrelation of a sequence (after removing burn-in), as a function of lag, k, is

$$\rho_k = \frac{\sum_{i=1}^{n-k}\left(\theta_i - \bar{\theta}\right)\left(\theta_{i+k} - \bar{\theta}\right)}{\sum_{i=1}^{n-k}\left(\theta_i - \bar{\theta}\right)^2} = \frac{\mathrm{Cov}_i(\theta_i, \theta_{i+k})}{\mathrm{Var}(\theta)}$$

- The larger lag one needs to get a small autocorrelation, the less informative individual samples are.
- The pandas function autocorrelation_plot() may be useful for this.

CORRELATION TESTS



Note that the positive/negative oscillations basically tell us when the lag is so large compared with the chain length that the autocorrelation is too noisy to be meaningful.

We would be justified in **thining** the chains by a factor of ~150, apparently! (i.e. take every 150th sample)

EFFECTIVE NUMBER OF SAMPLES

- From m chains of length n, we can estimate the effective number of independent samples as: (though we never actually sum to infinity - it's cut off as $\hat{\rho}_t$ gets numerically unstable to calculate

$$n_{eff} = \frac{mn}{1 + 2\sum_0^\infty \hat{\rho}_t}$$

- with (*V - no subscript t -* is the same as the Gelman-Rubin calculation on slide 27)

$$\hat{\rho}_t = 1 - \frac{V_t}{2V}$$

- and

$$V_t = \frac{1}{m(n-t)} \sum_{j=0}^{m} \sum_{i=t+1}^{n} (\theta_{i,j} - \theta_{i-t,j})^2$$