Reminder of some Markov Chain properties:

1. a transition from one state to another occurs probabilistically

2. only state that matters is where you currently are (i.e. given present, future is independent of the past)

Reminder of some Markov Chain properties:

1. a transition from one state to another occurs probabilistically

+

2. only state that matters is where you currently are (i.e. given present, future is independent of the past)

=

the next value x_1 depends **only on the last value** x_0 according to some **transition probability** $p(x_0 \rightarrow x_1)$

Reminder of some Markov Chain properties:

the next value x_1 depends **only on the last value** x_0 according to some **transition probability** $p(x_0 \rightarrow x_1)$

if you can get to any state from any state in a finite number of steps, the distribution of the population in each state remains the same after a while. This is referred as the **stationary distribution**.

probability distribution!

$\pi(x)$ Probability density for the state x for an infinitely long chain.

$$\pi * p = \pi$$

Markov Chain Monte Carlo

1) Start from some initial parameter value
2) Evaluate the unnormalized posterior
3) Propose a new parameter value*
4) Evaluate the new unnormalized posterior
5) Decide whether or not to accept the new value*
6) Repeat 3-5

The idea of MCMC

* Could we find a transition rule *p* such that the *stationary distribution*:

  - Exists

  - Equals to the posterior (target distribution)

Markov Chain Monte Carlo

1) Start from some initial parameter value
2) Evaluate the unnormalized posterior
3) Propose a new parameter value*
4) Evaluate the new unnormalized posterior
5) Decide whether or not to accept the new value*
6) Repeat 3-5

Markov Chain Monte Carlo

- Looks remarkably similar to optimization
  - Evaluating posterior rather than just likelihood
  - “Repeat” does not have a stopping condition
  - **Criteria for accepting a proposed step**
    - Optimization – diverse variety of options but no “rule”
    - MCMC – stricter criteria for accepting
- Converges “in distribution” rather than to a single point

Markov Chain Monte Carlo

(vs rejection sampling)

1) Start from some initial parameter value

2) Evaluate the unnormalized posterior

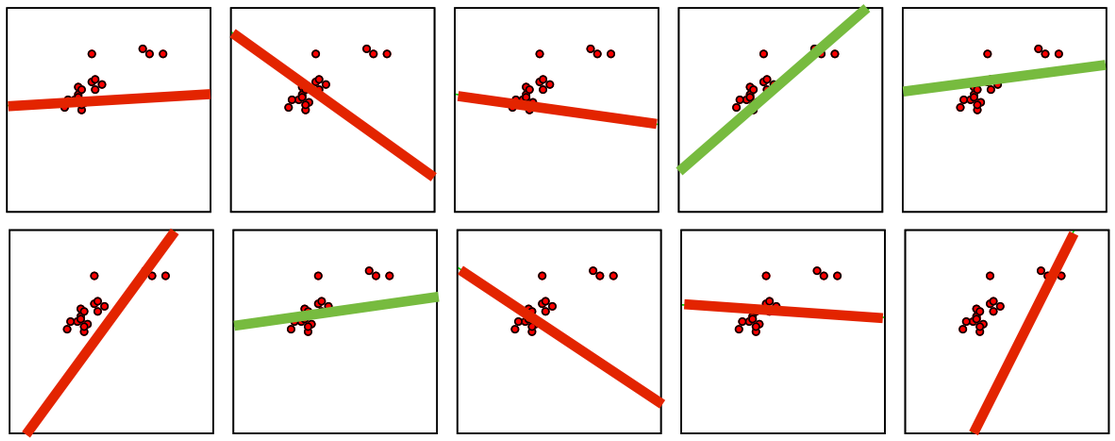
3) Propose a new parameter value depending on the previous value, according to a transition probability (jump function needed!)

4) Evaluate the new unnormalized posterior

5) Decide whether or not to accept the new value in comparison to the value from the previous step (no envelope function needed!)

6) Repeat

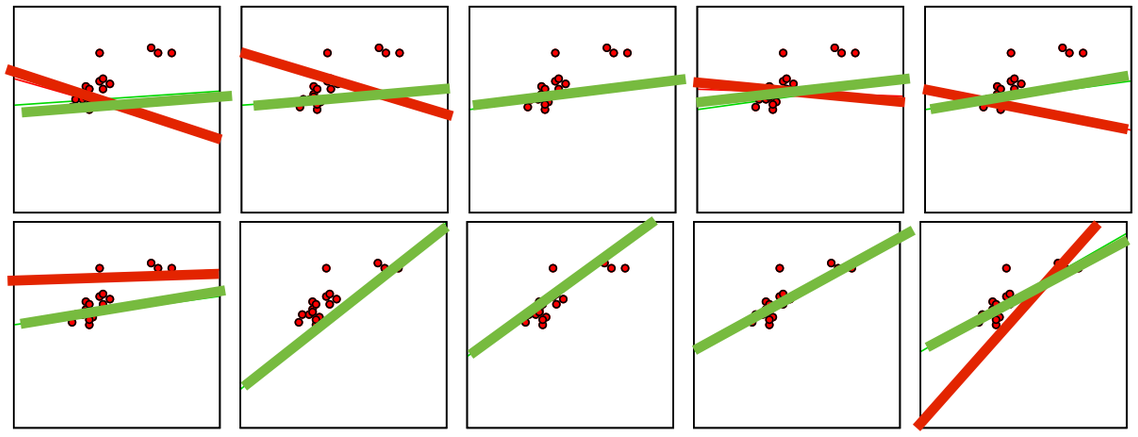
Rejection Sampling



Explore everywhere!

- Samples are independent
- Parallelizable (computationally effective)
- Easy to implement, easy to design
- Need A LOT of draws in higher dimensions (computationally costly)
- Sample everywhere, approximate answer

MCMC



Avoid empty space!

- Each sample dependent on previous
- Sequential (computationally costly)
- Easy to implement, harder to design
- Need slightly more draws than usual in higher dimensions (computationally effective)
- Did you miss anything?
  need to assess convergence

Metropolis Algorithm

- Most popular form of MCMC
- Can be applied to most any problem
- Implementation requires little additional thought beyond writing the model
- Evaluation/Tuning does require the most skill & experience
- Indirect Method
  - Requires a second distribution to propose steps

Metropolis Algorithm

1) Start from some initial parameter value θ^c

2) Evaluate the unnormalized posterior $p(\theta^c|X)$

3) Propose a new parameter value θ^*

Random draw from a "jump" distribution centered on the current parameter value

4) Evaluate new unnormalized posterior $p(\theta^*|X)$

5) Decide whether or not to accept the new value

Accept new value with probability

$a = p(\theta^*) / p(\theta^c)$

```
if
   runif(1,0,1) < a
   accept
else
   reject
```

6) Repeat 3-5

$$p(\theta^c \rightarrow \theta^*) = P(\theta^*) / P(\theta^c)$$

Accept new value with probability

$a = P(\theta^*) / P(\theta^c)$

means

$P(\theta^*) > P(\theta^c)$: $a > 1$ accept always

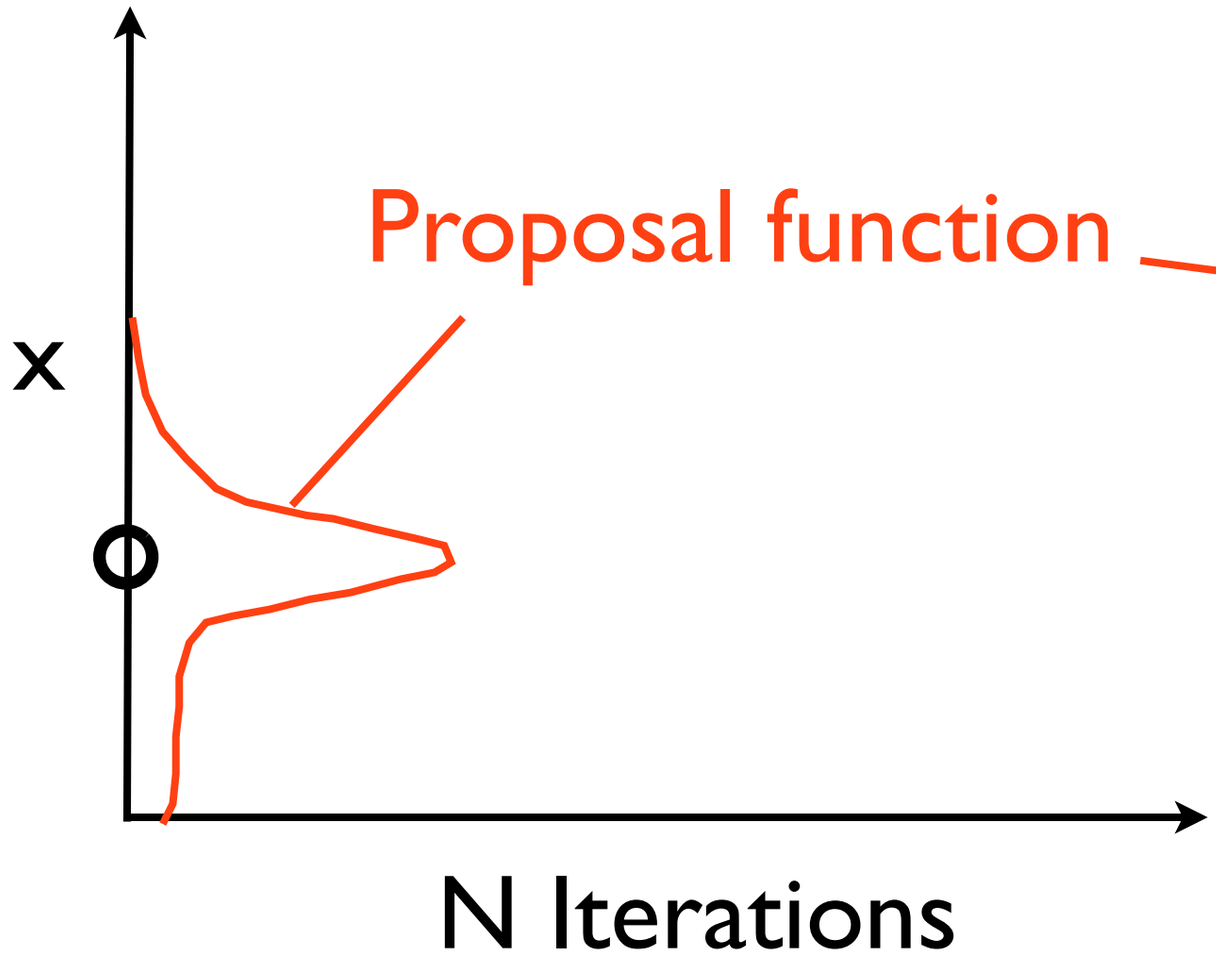
$P(\theta^*) < P(\theta^c)$: $0<a<1$ accept sometimes

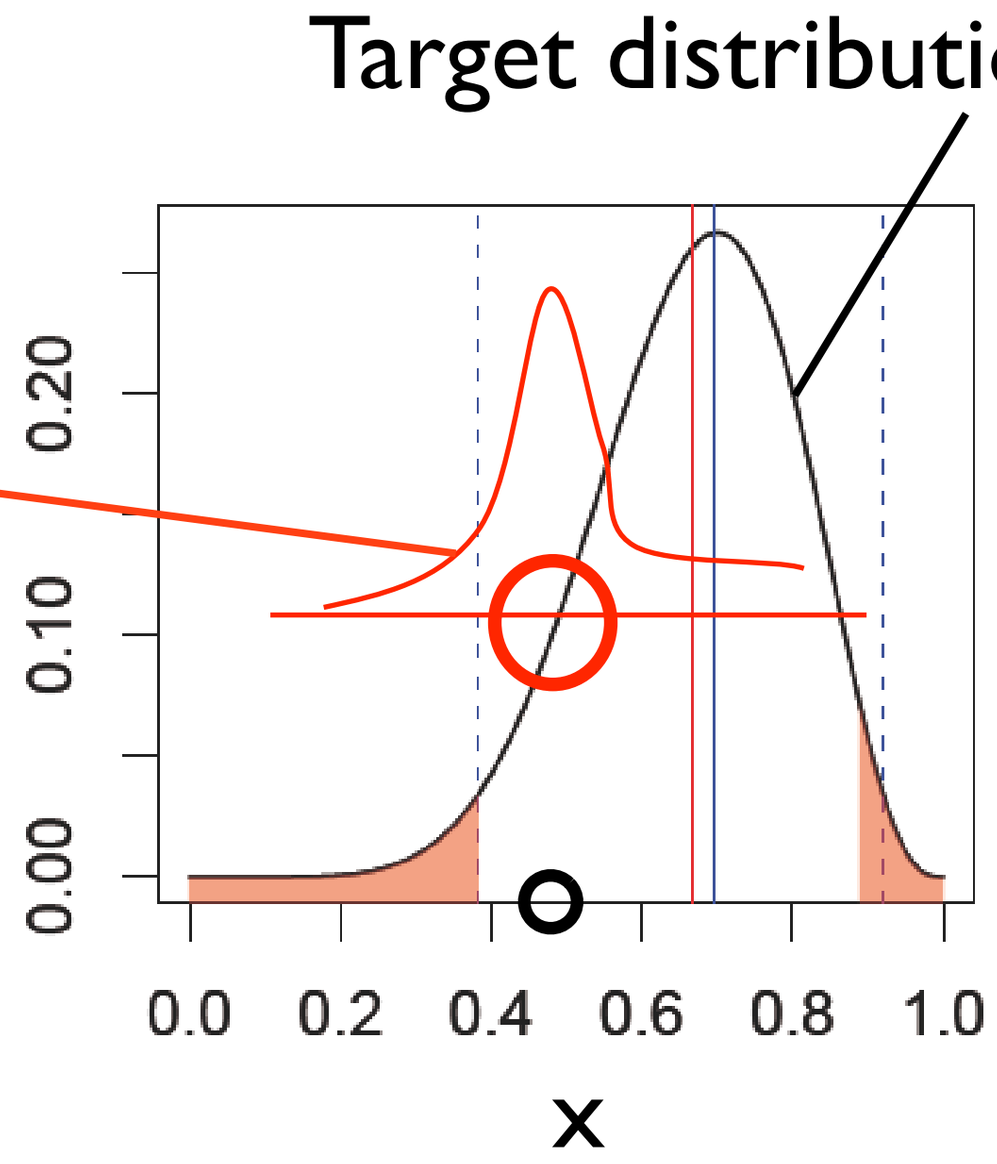
```
if
  runif(1,0,1) < a
  accept
else
  reject
```

Proposal function

Idea: concentrate the sampling in the good areas

N = 0 A = 0 AR = 0

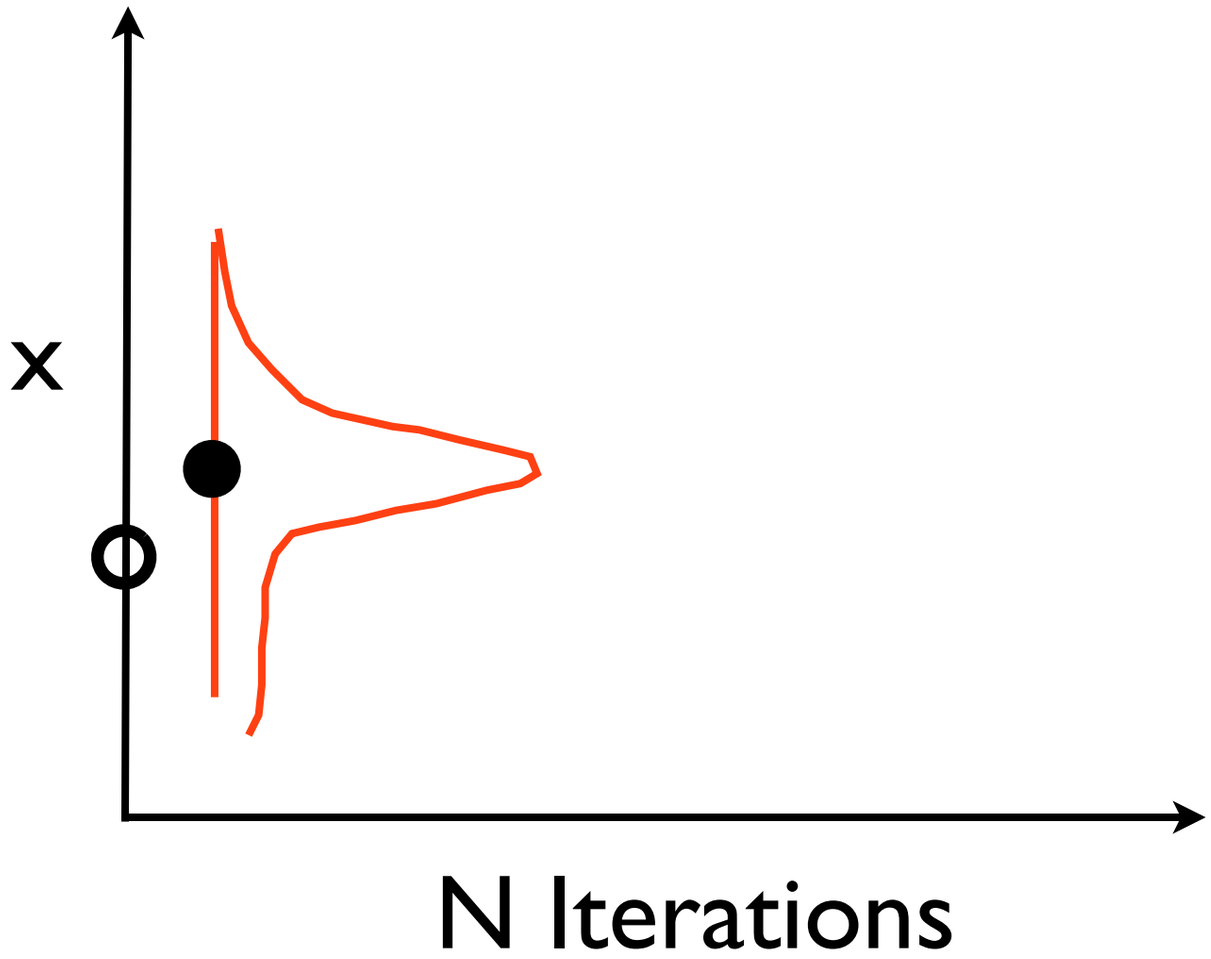




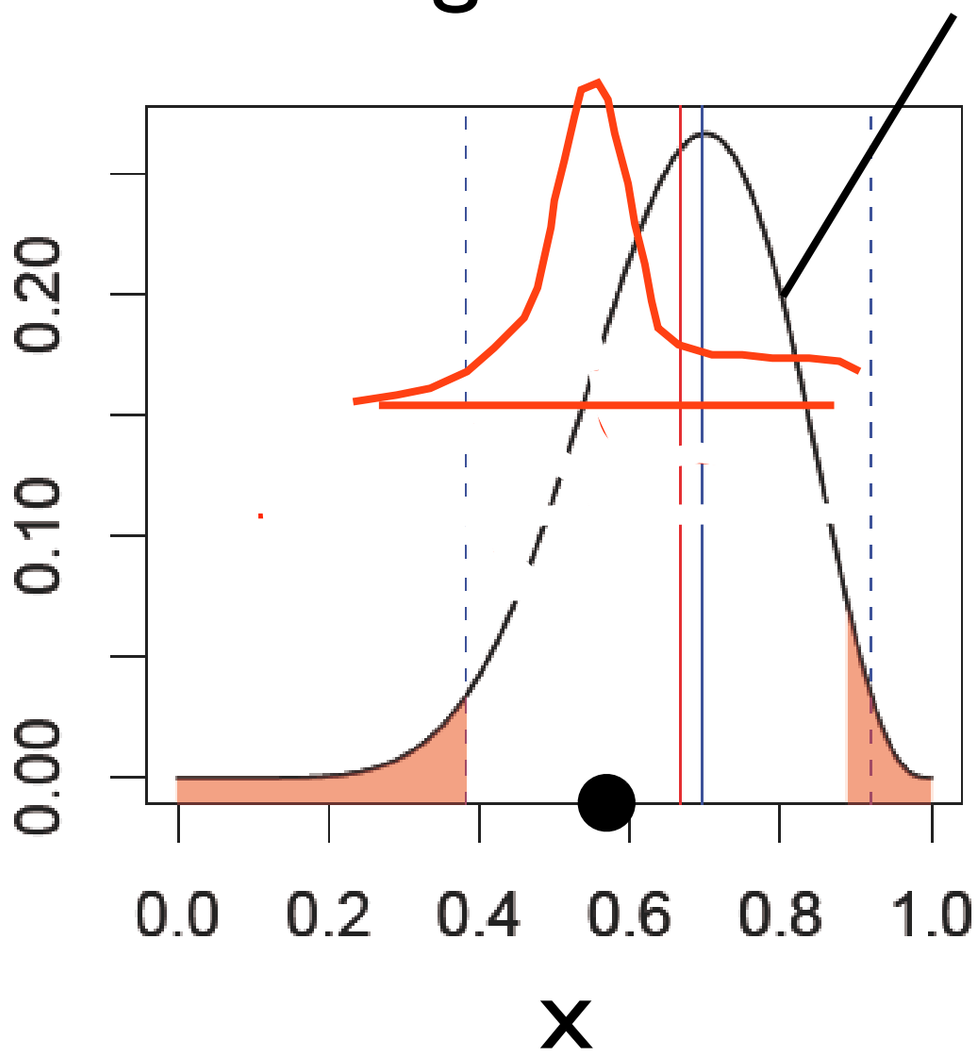
Proposal function

Idea: concentrate the sampling in the good areas

N = 1 A = 1 AR = 1



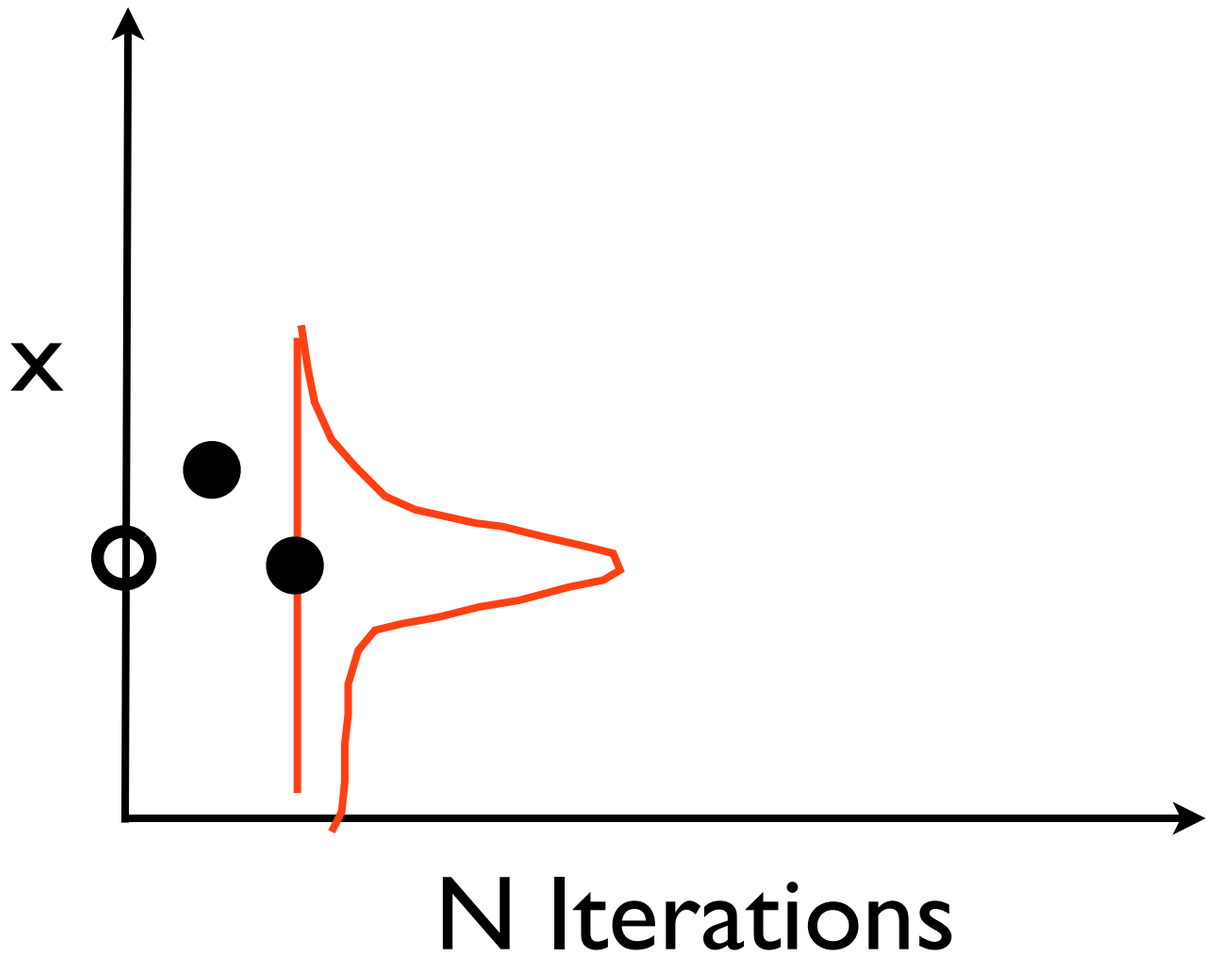
Target distribution



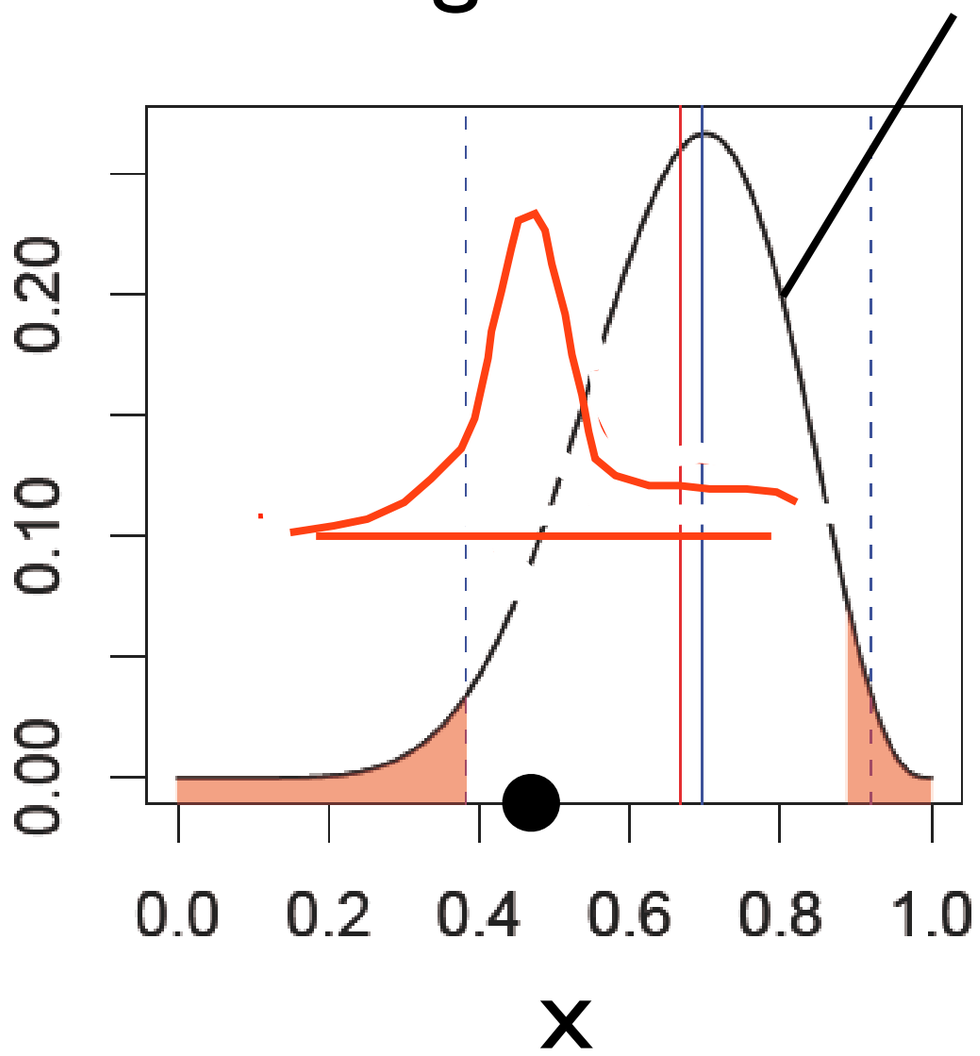
Proposal function

Idea: concentrate the sampling in the good areas

N = 2 A = 2 AR = 1



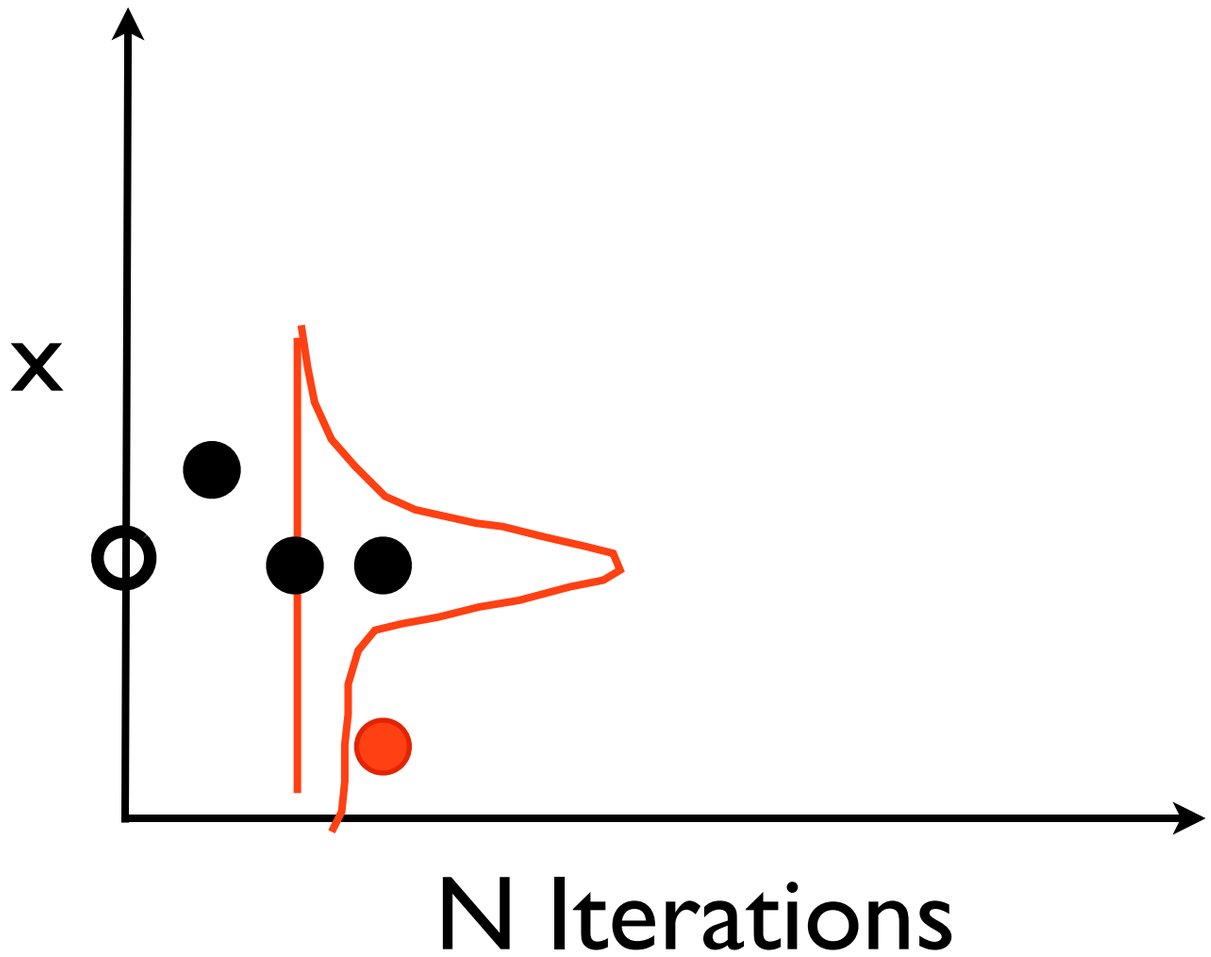
Target distribution



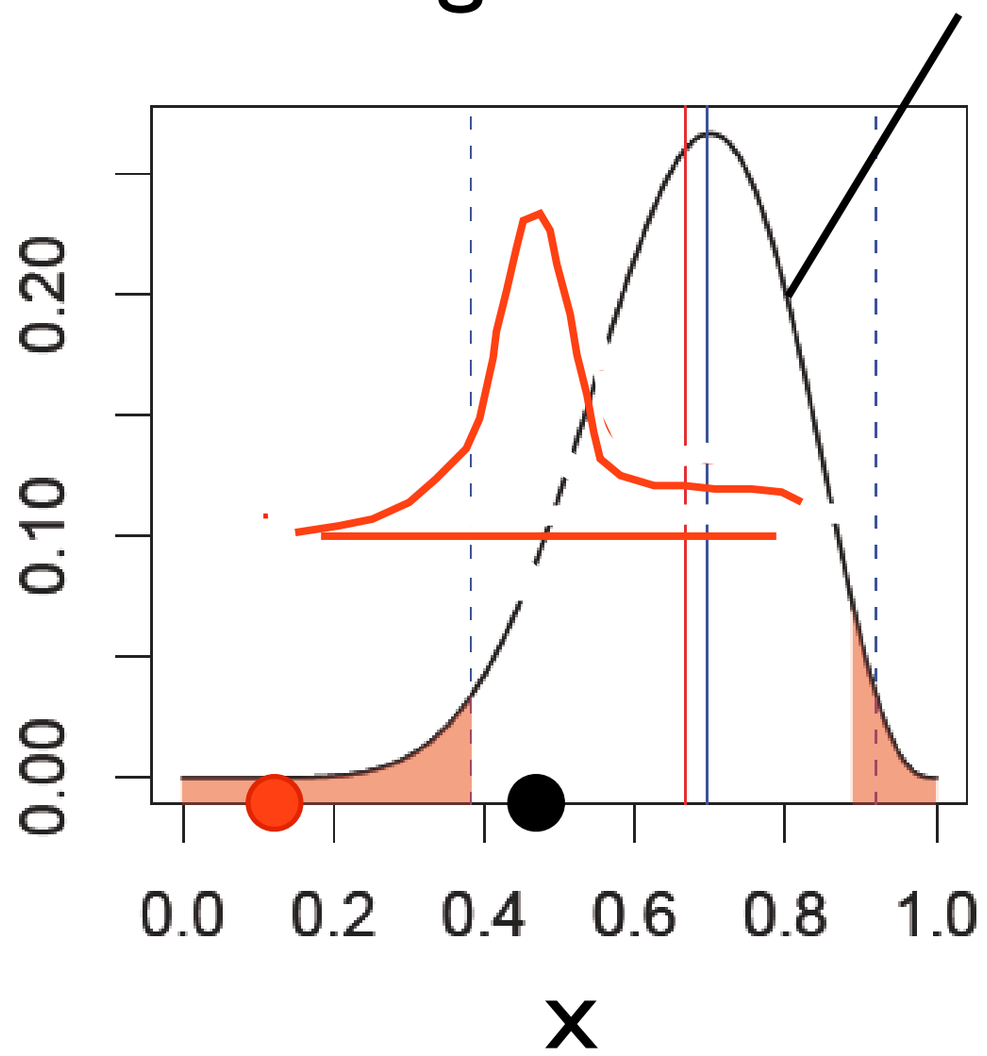
Proposal function

Idea: concentrate the sampling in the good areas

N = 3 A = 2 AR = 2/3



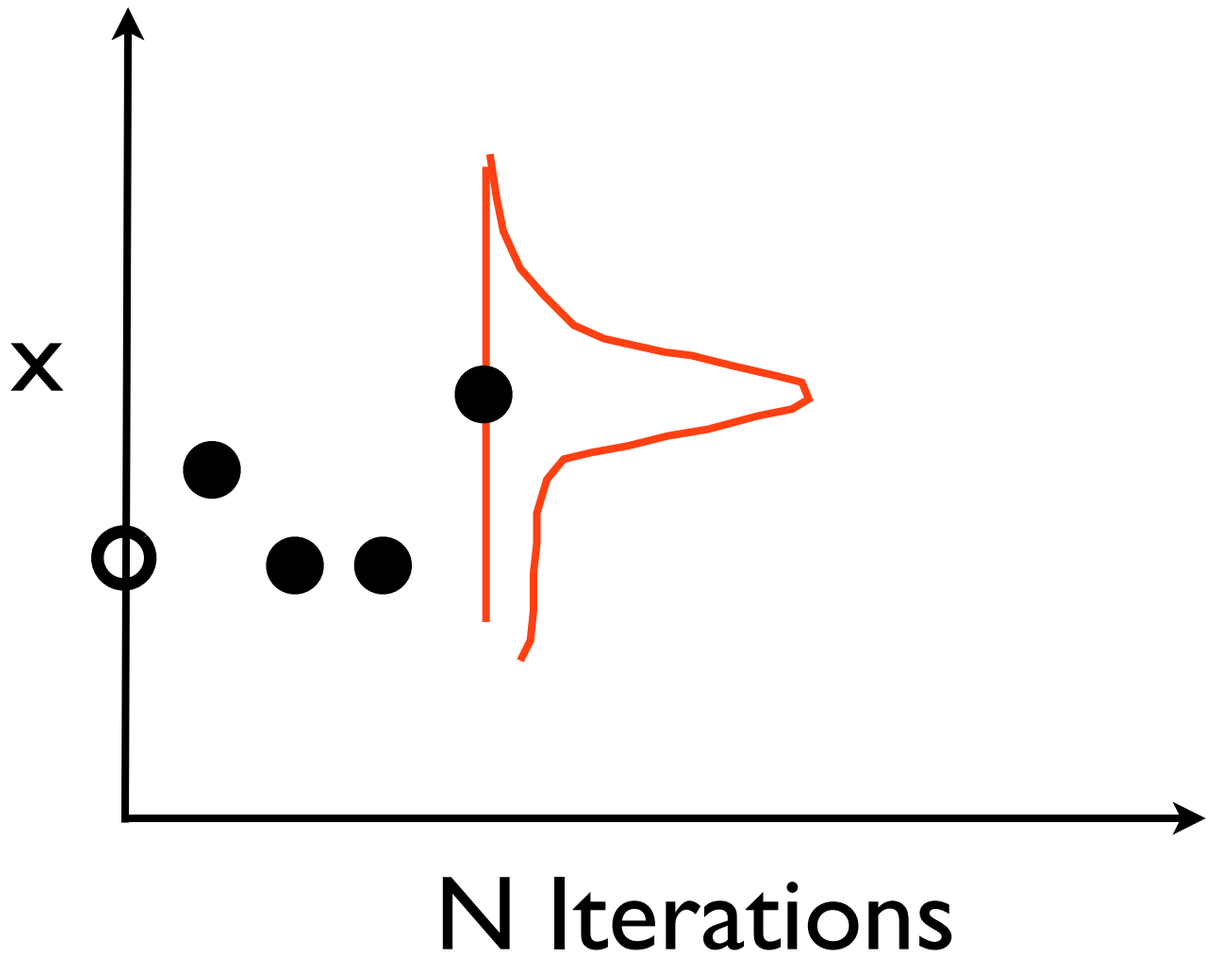
Target distribution

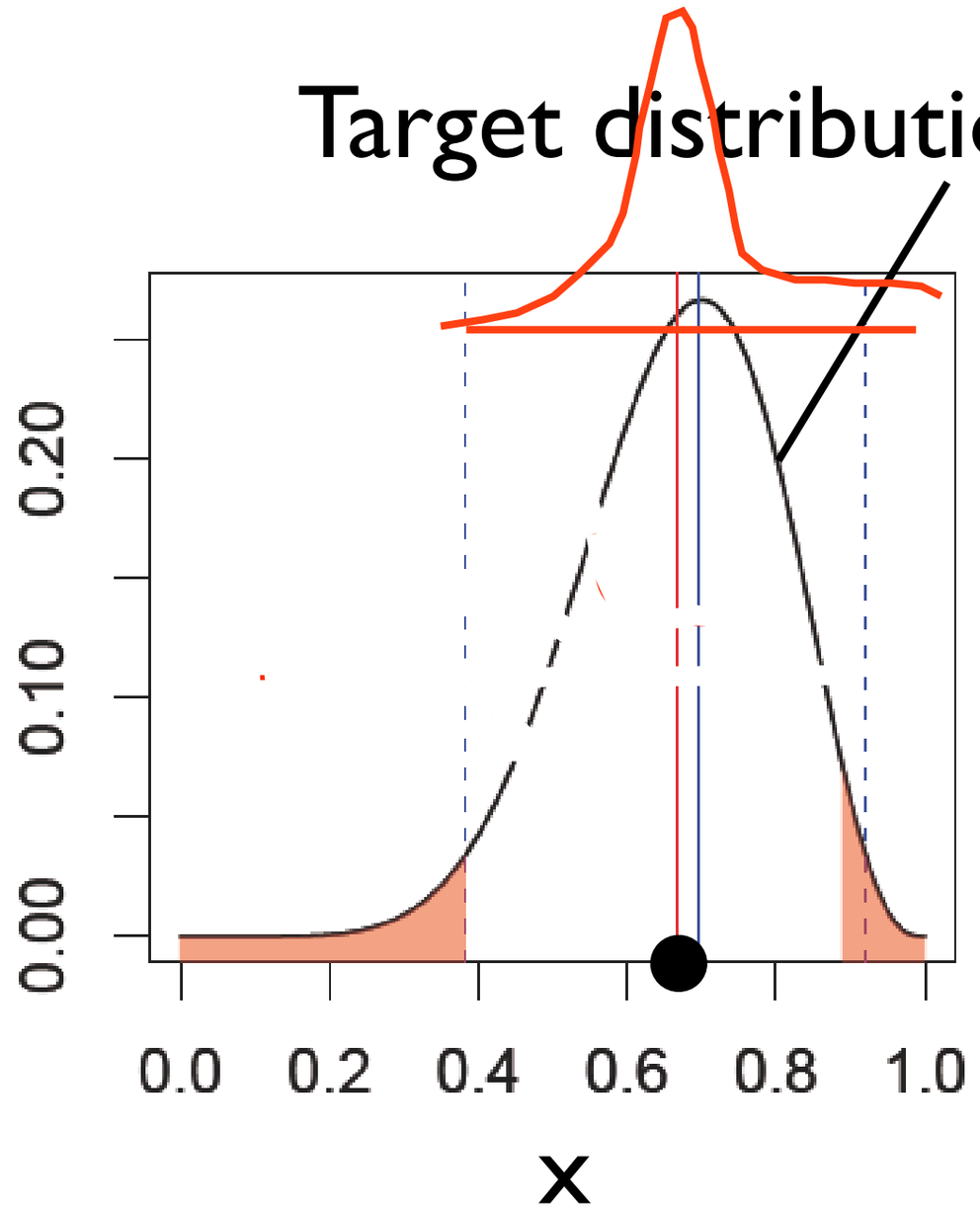


Proposal function

Idea: concentrate the sampling in the good areas

N = 4 A = 3 AR = 3/4

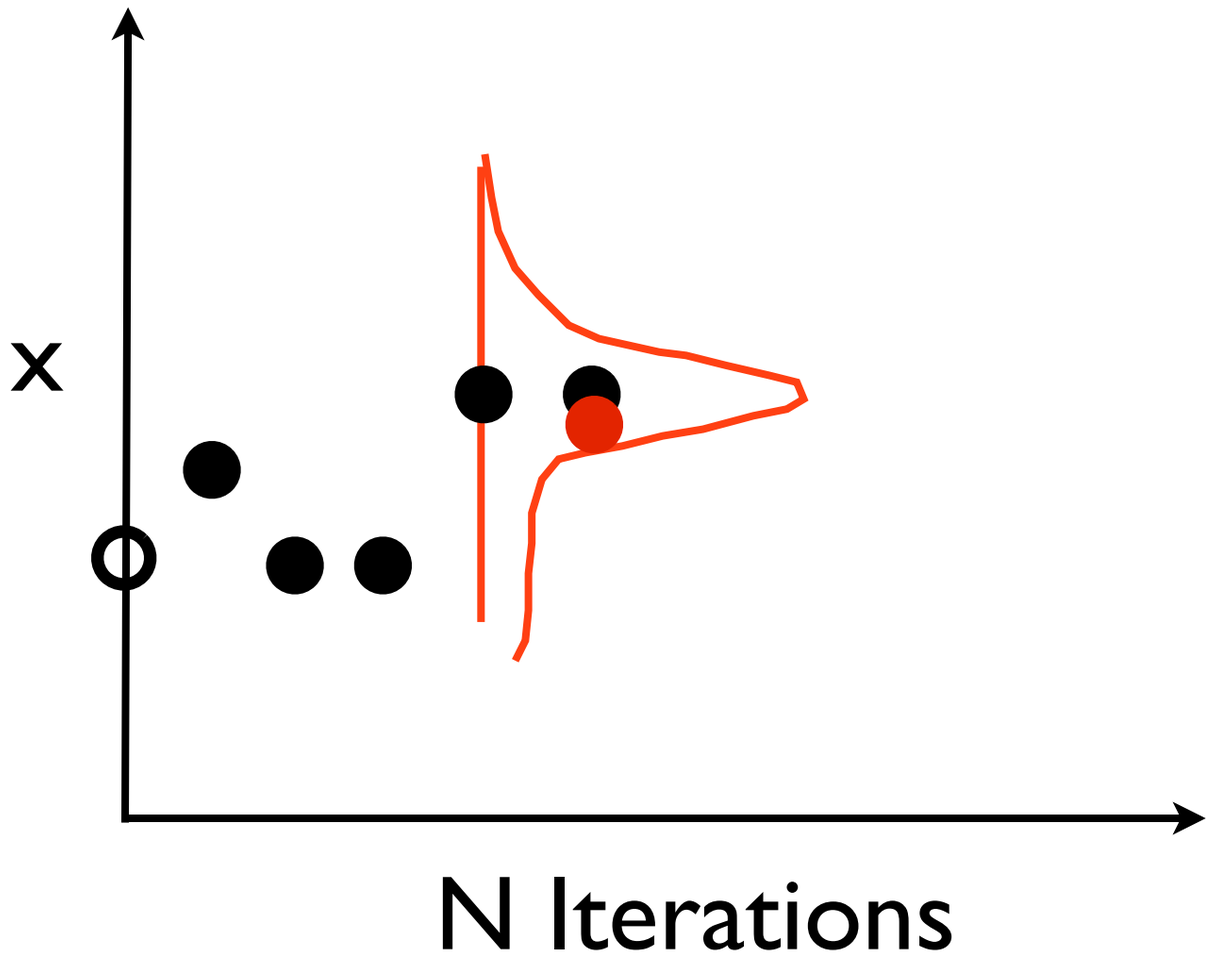


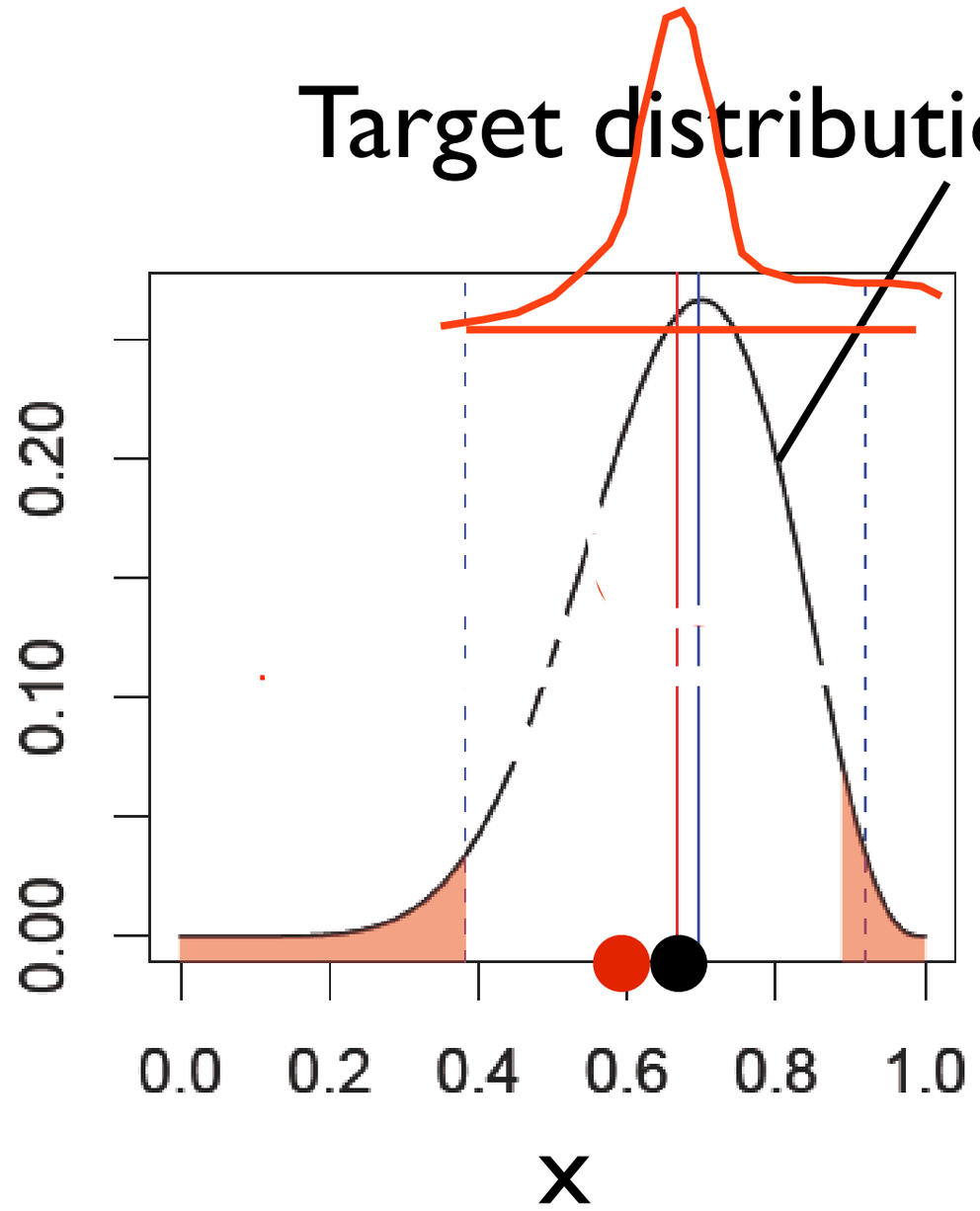


Proposal function

Idea: concentrate the sampling in the good areas

N = 5 A = 3 AR = 3/5

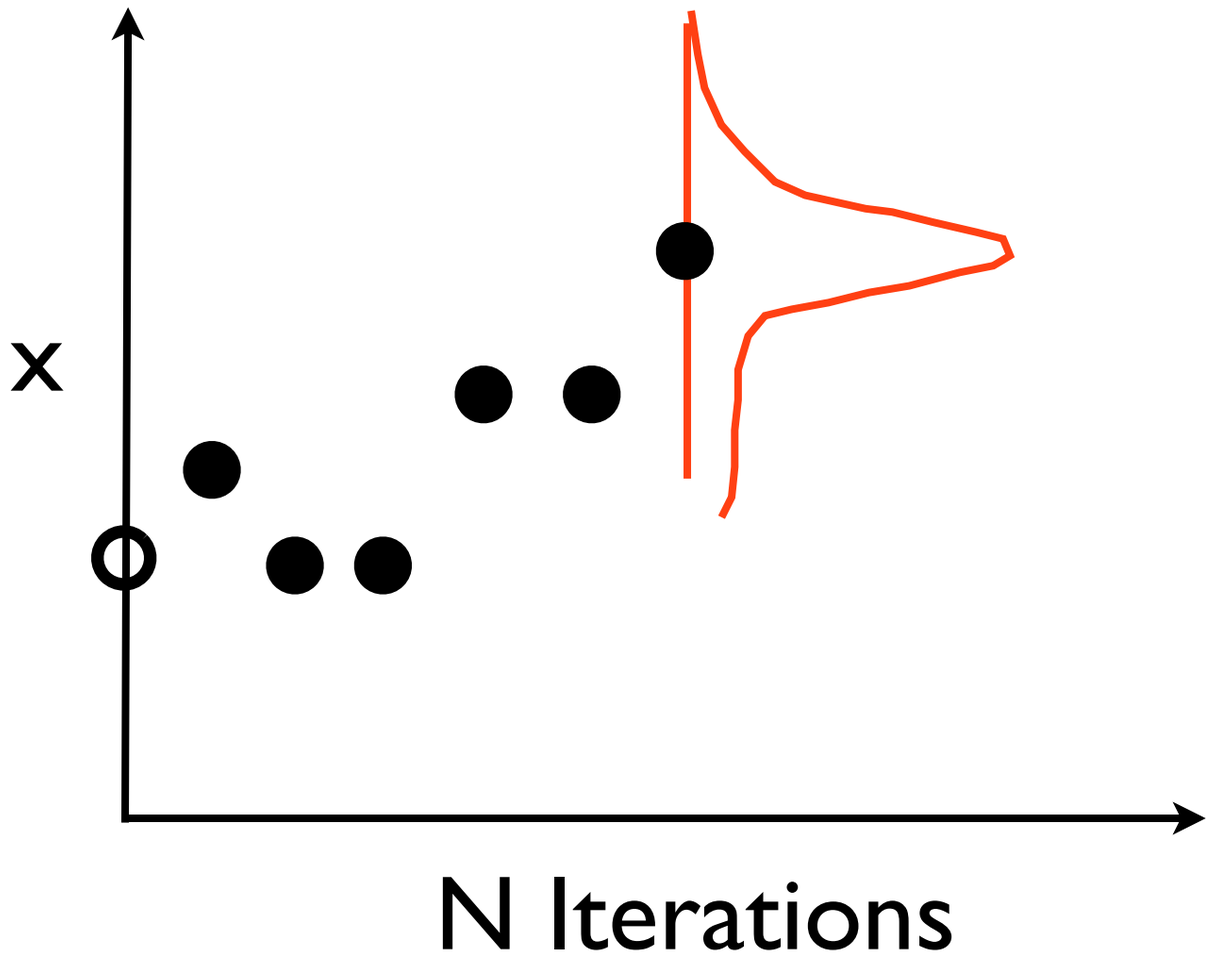


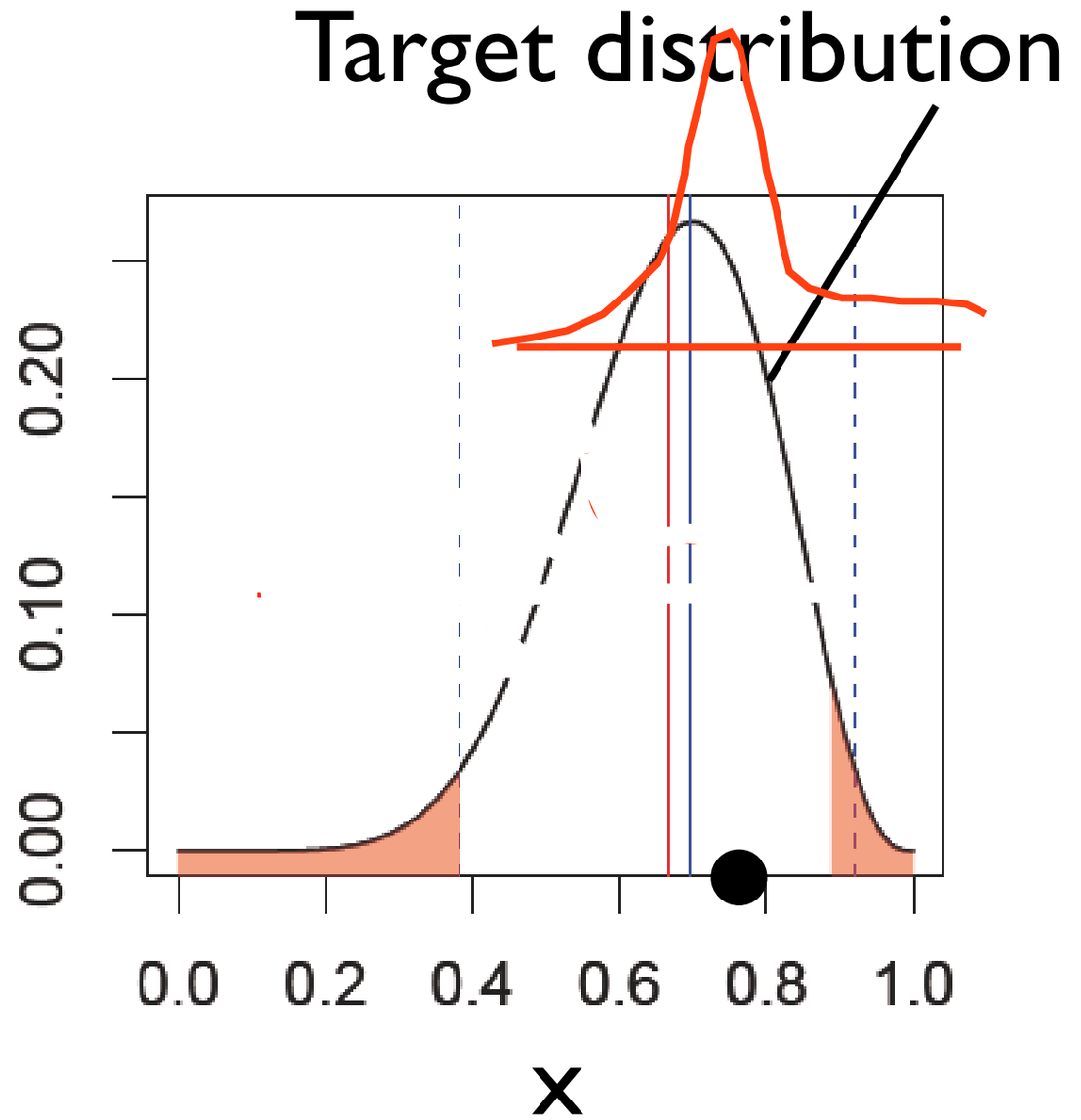


Proposal function

Idea: concentrate the sampling in the good areas

N = 6 A = 4 AR = 4/6

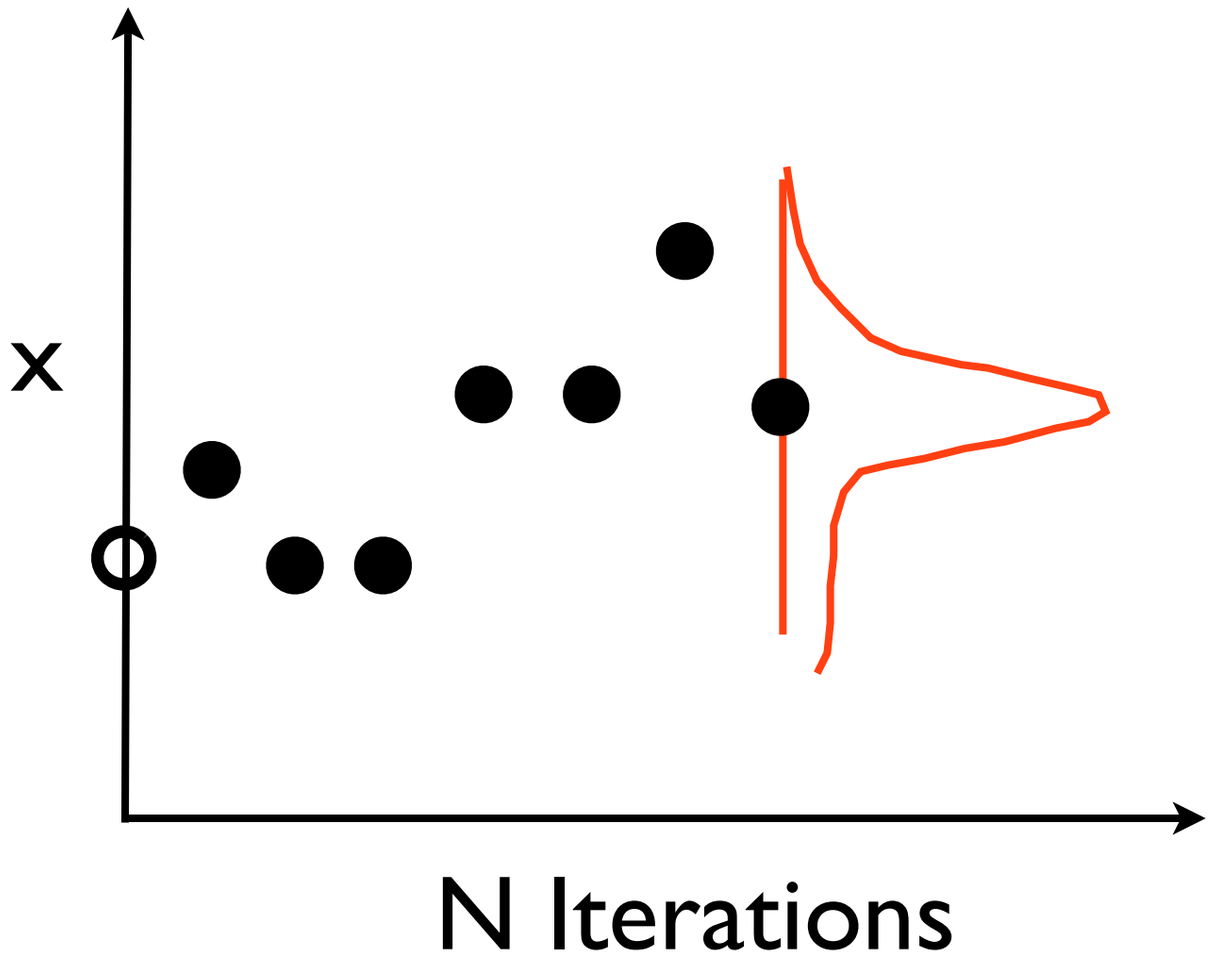


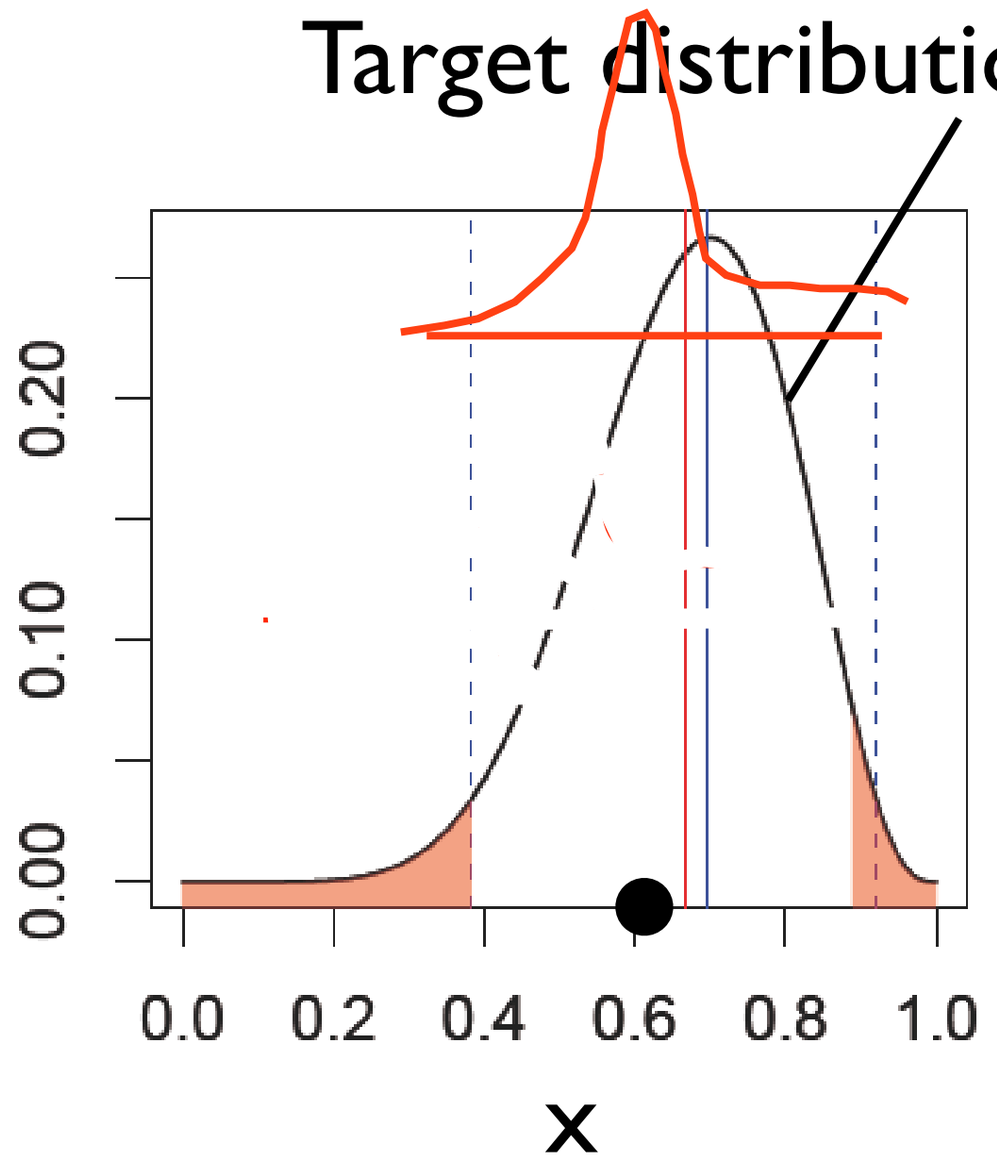


Proposal function

Idea: concentrate the sampling in the good areas

N = 7 A = 5 AR = 5/7

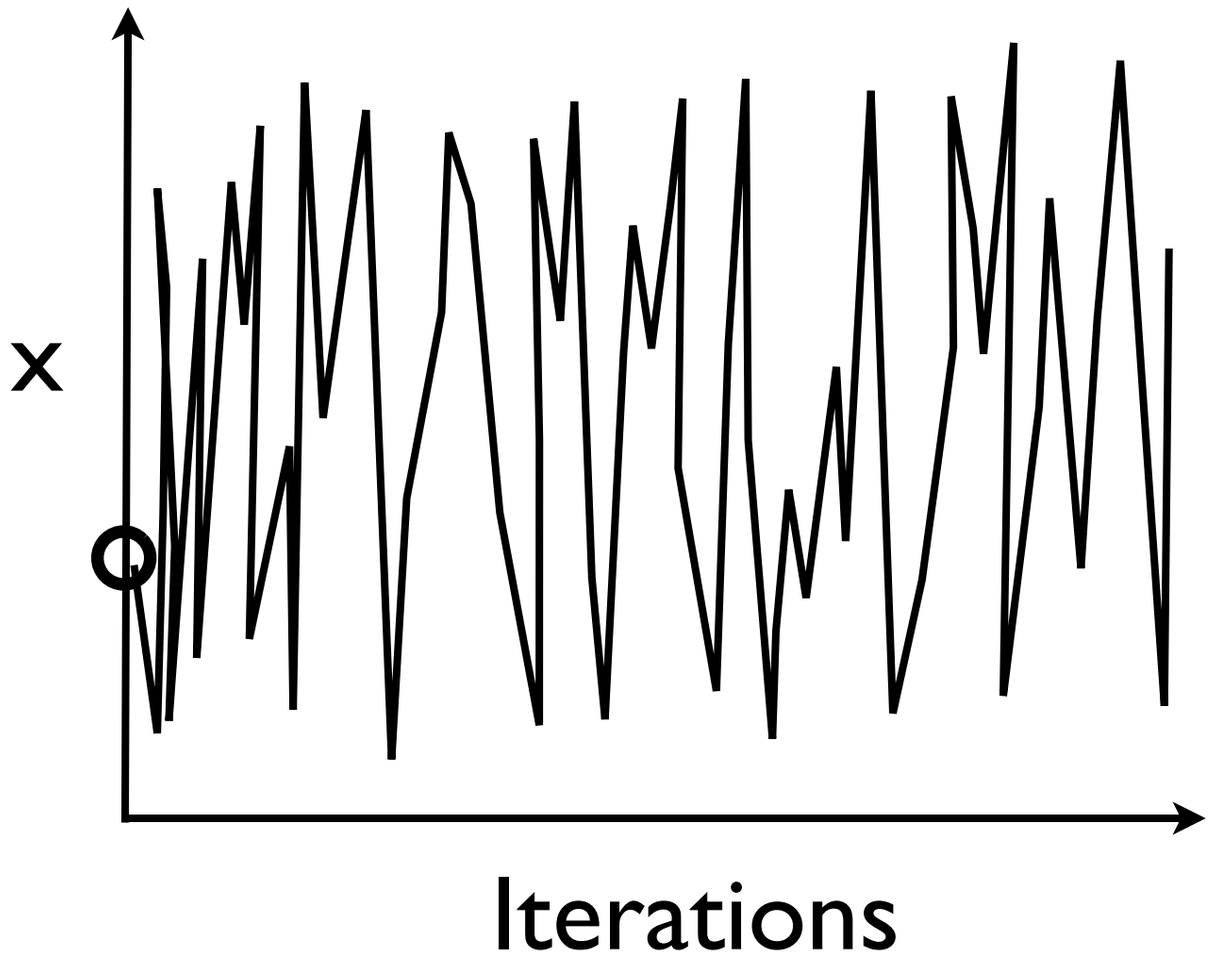




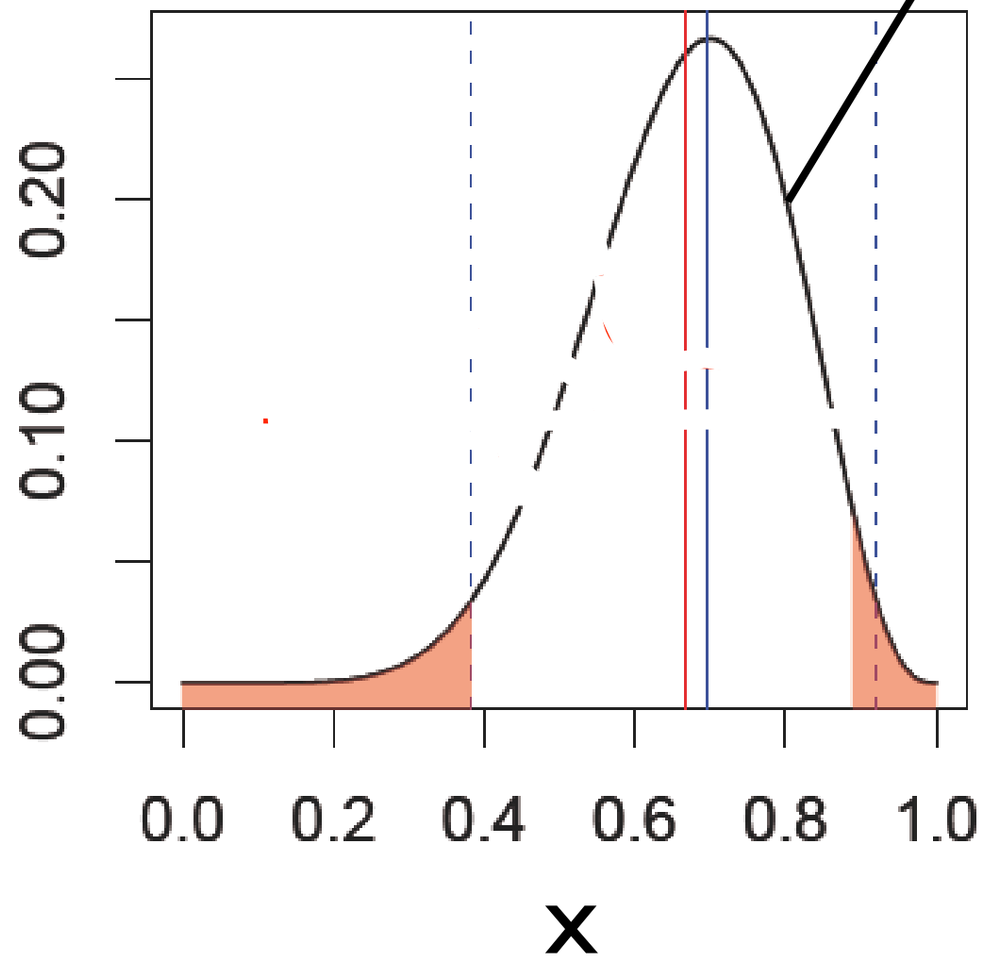
Proposal function

Idea: concentrate the sampling in the good areas

AR = 30-70%

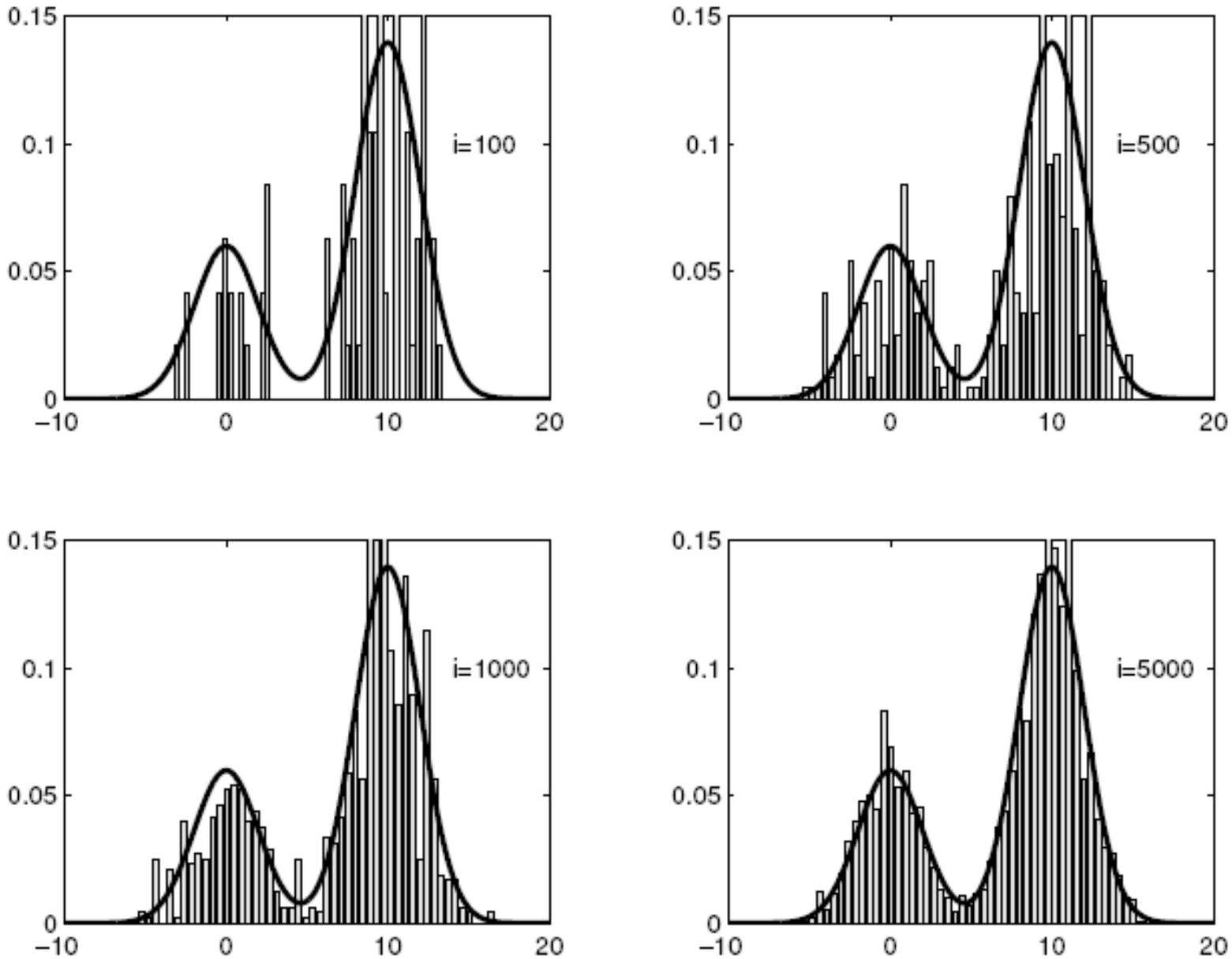


Target distribution



Convergence to the target distribution

do you have a representative sample of the target distribution (posterior) yet?



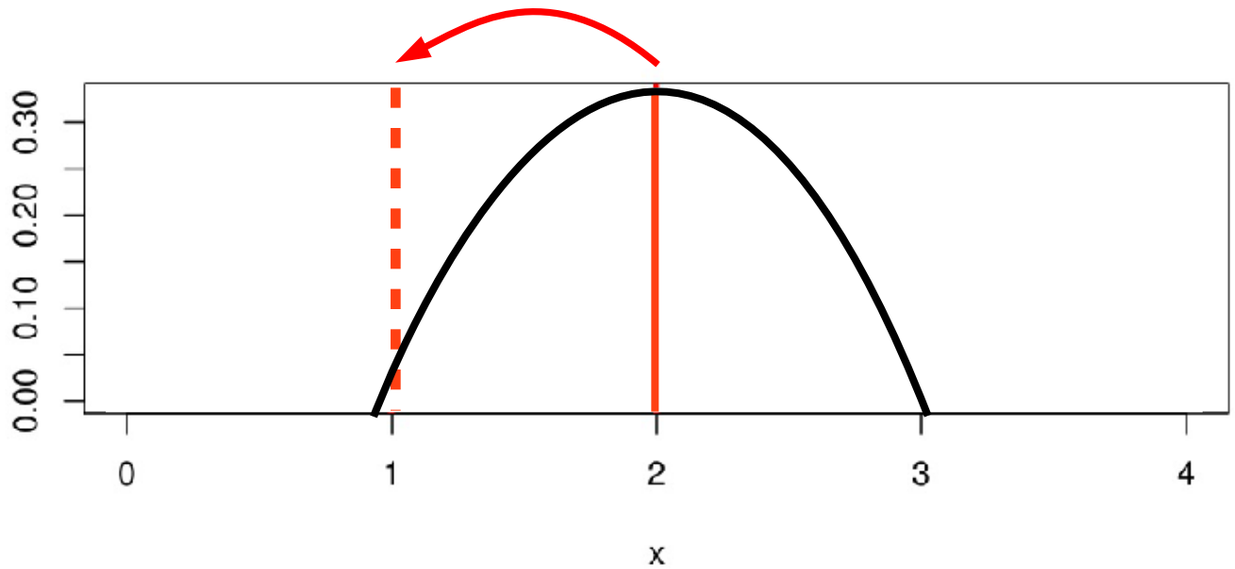
youtube: youtu.be/zL2lg_Nfi80

Jump distribution

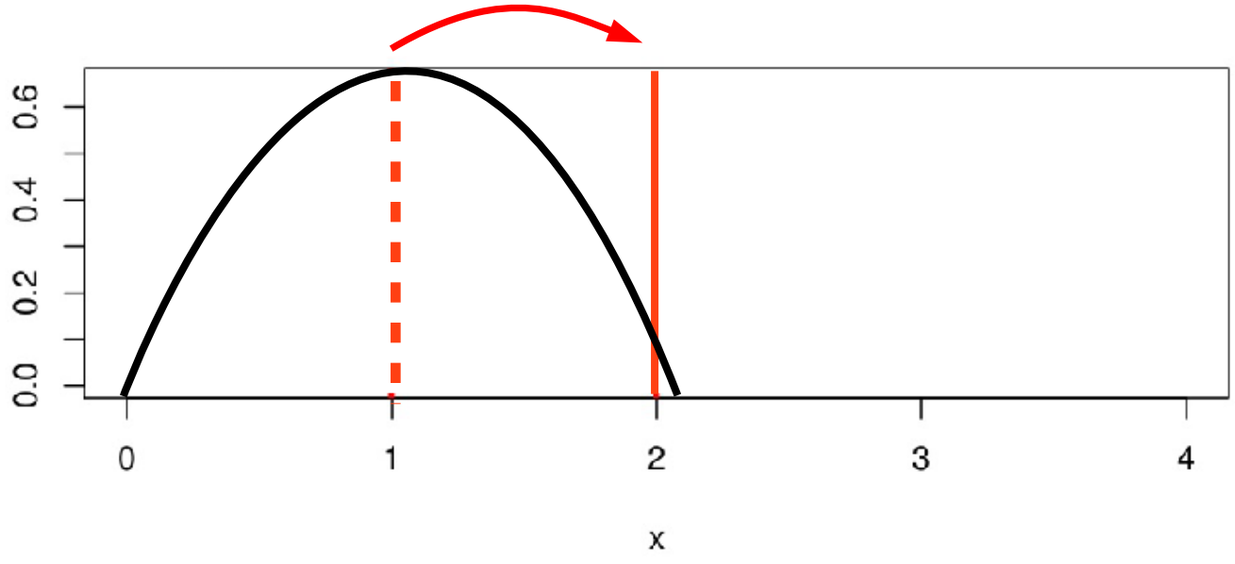
- For Metropolis, Jump distribution J must be SYMMETRIC

  $J(\theta^* \mid \theta^c) = J(\theta^c \mid \theta^*)$

$$J(\theta^* | \theta^c)$$



$$J(\theta^c | \theta^*)$$



$$J(\theta^* | \theta^c) = J(\theta^c | \theta^*)$$

Jump distribution

- For Metropolis, Jump distribution J must be SYMMETRIC

  $J(\theta^* \mid \theta^c) = J(\theta^c \mid \theta^*)$

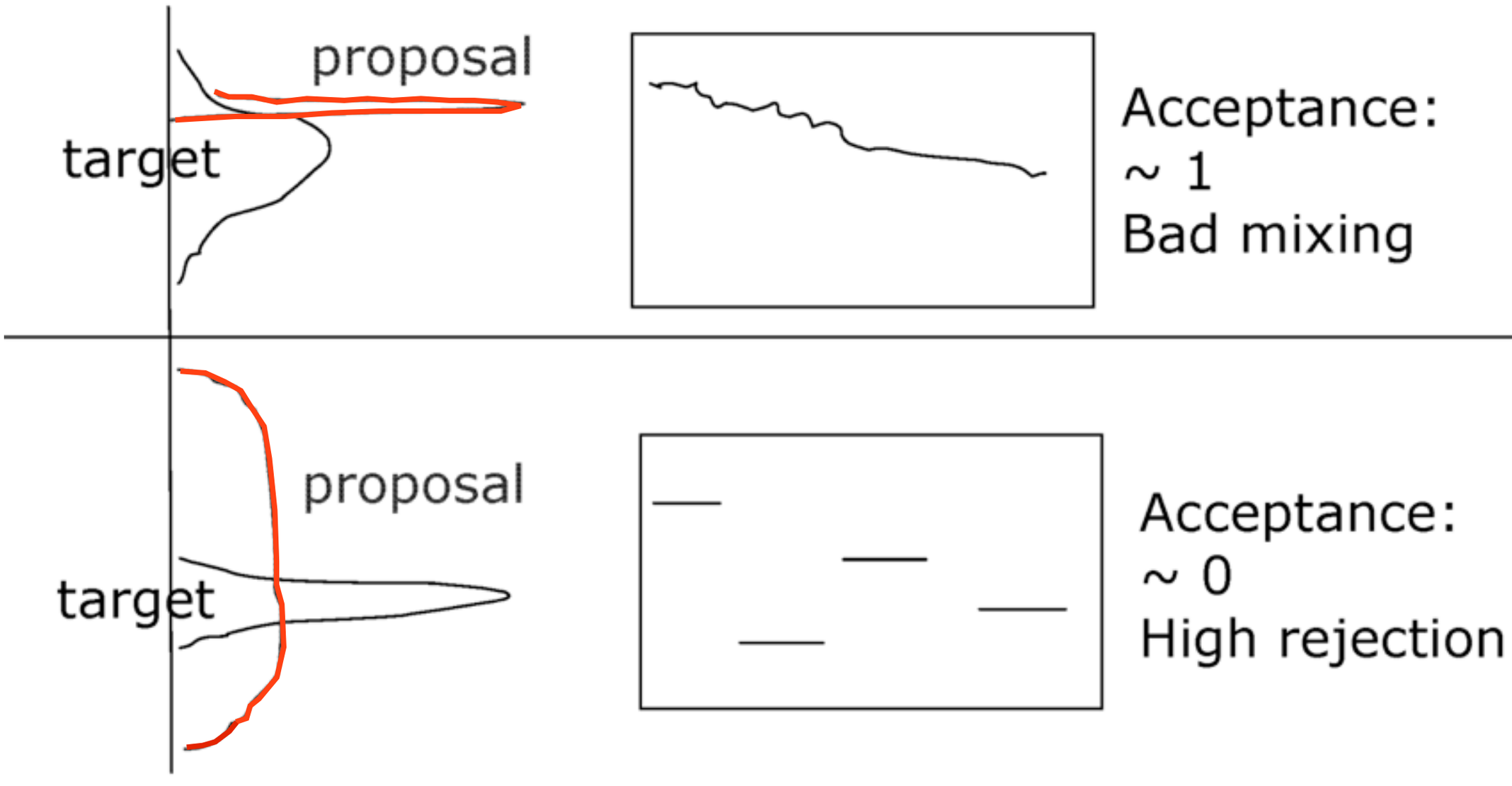
- Most common Jump distribution is the Normal

  $J(\theta^* \mid \theta^c) = N(\theta^* \mid \theta^c, \upsilon)$

- User must set the variance of the jump
  - Trial-and-error
  - Tune to get acceptance rate 30-70%
  - Low acceptance = decrease variance (smaller step)
  - Hi acceptance = increase variance (bigger step)

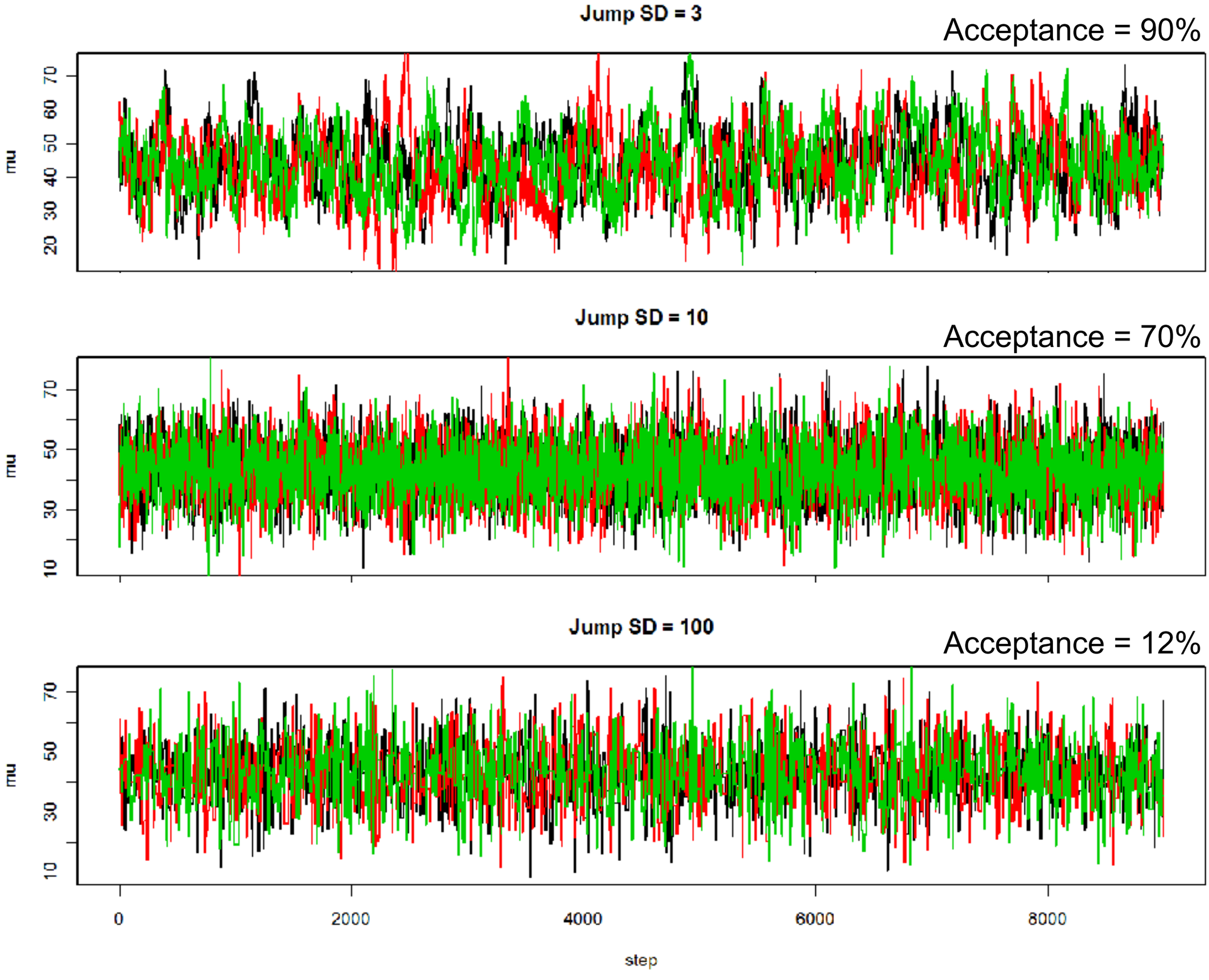
Speed of convergence

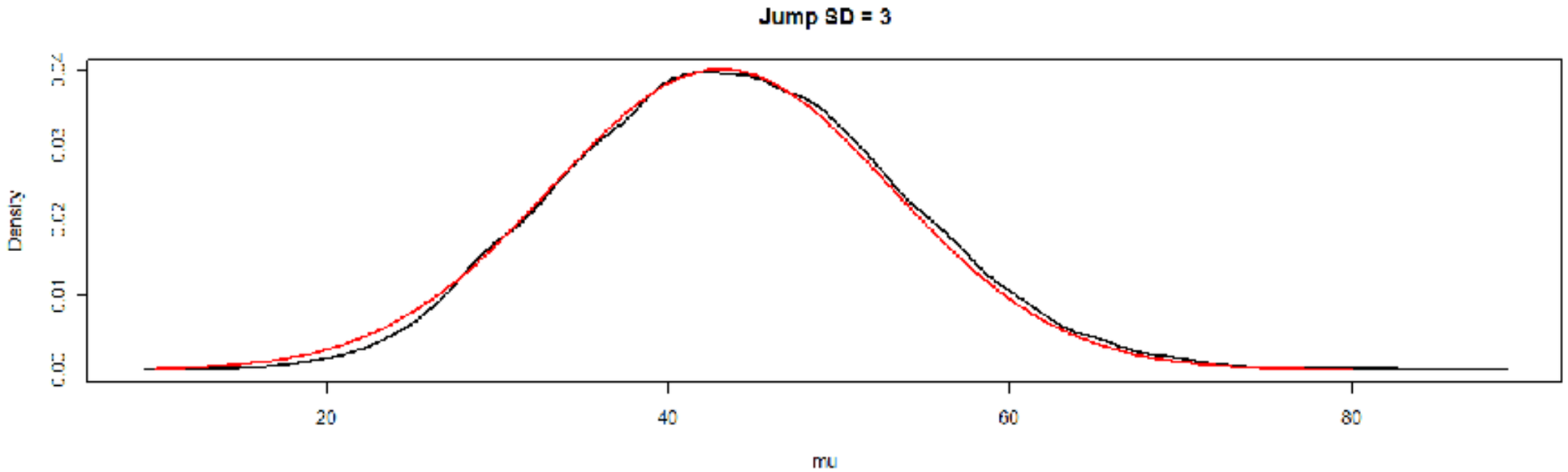
Speed of convergence is determined by the choice of good proposal distributions

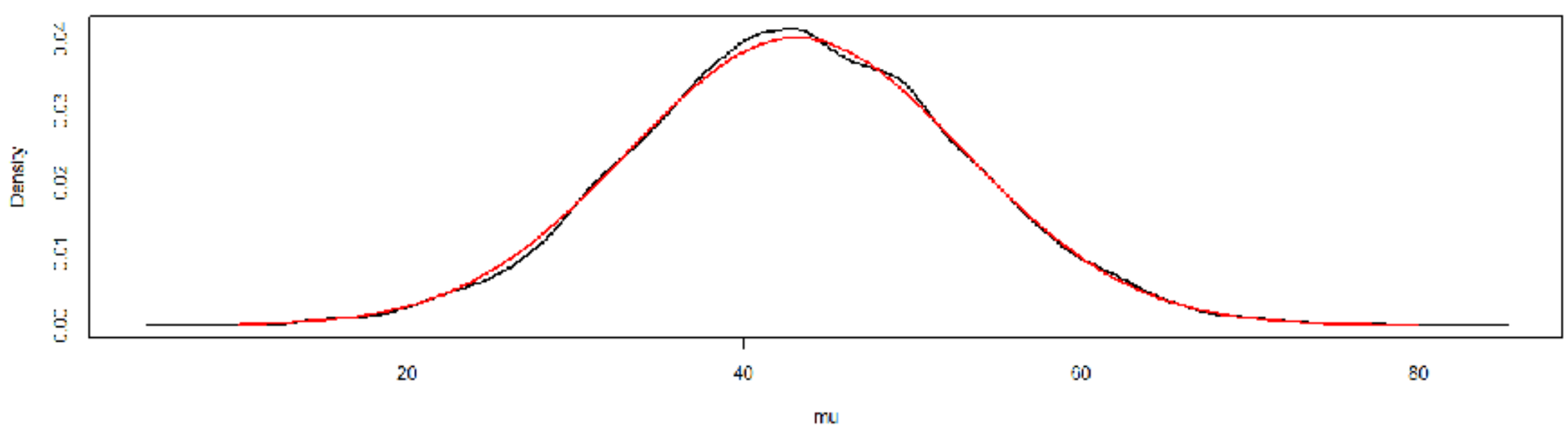


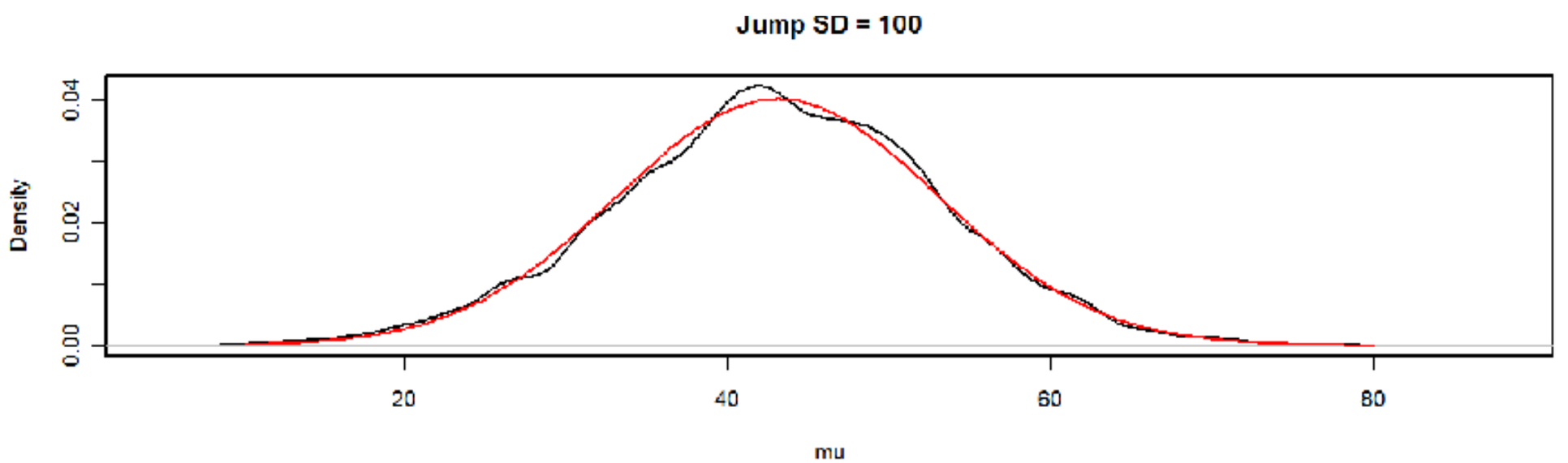
Example

- Normal with known variance, unknown mean
  - Prior: N(53,10000)
  - Data: y = 43
  - Known variance: 100
  - Initial conditions, 3 chains starting at -100, 0, 100
  - Jump distribution = Normal
  - Jump variance = 3,10,30

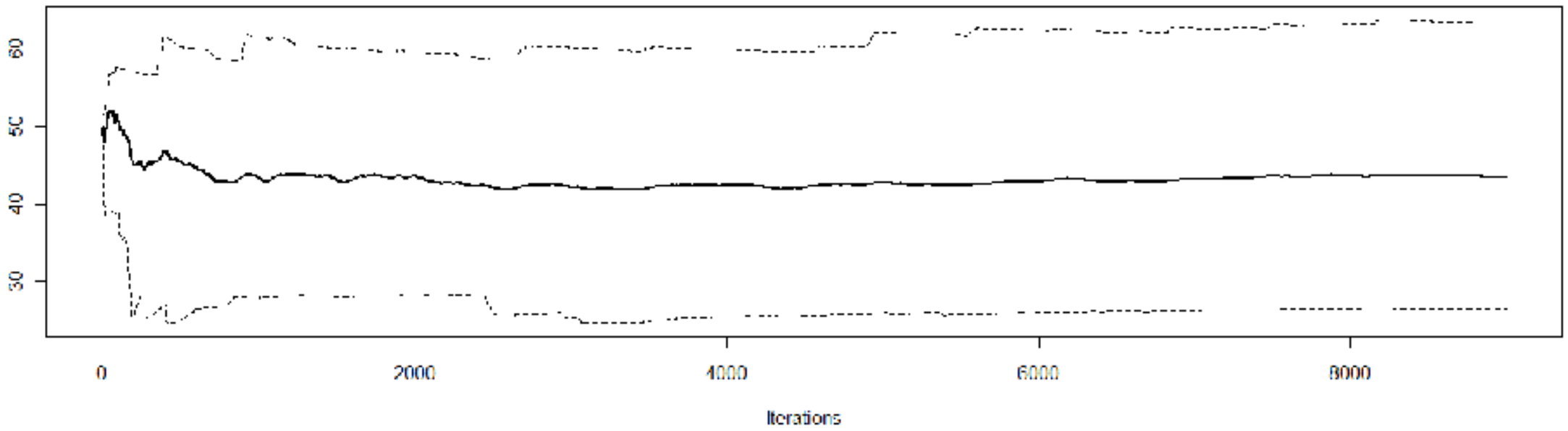

Jump SD = 3
Acceptance = 90%
Jump SD = 10
Acceptance = 70%
Jump SD = 100
Acceptance = 12%
mu
step
0
2000
4000
6000
8000
10
30
50
70

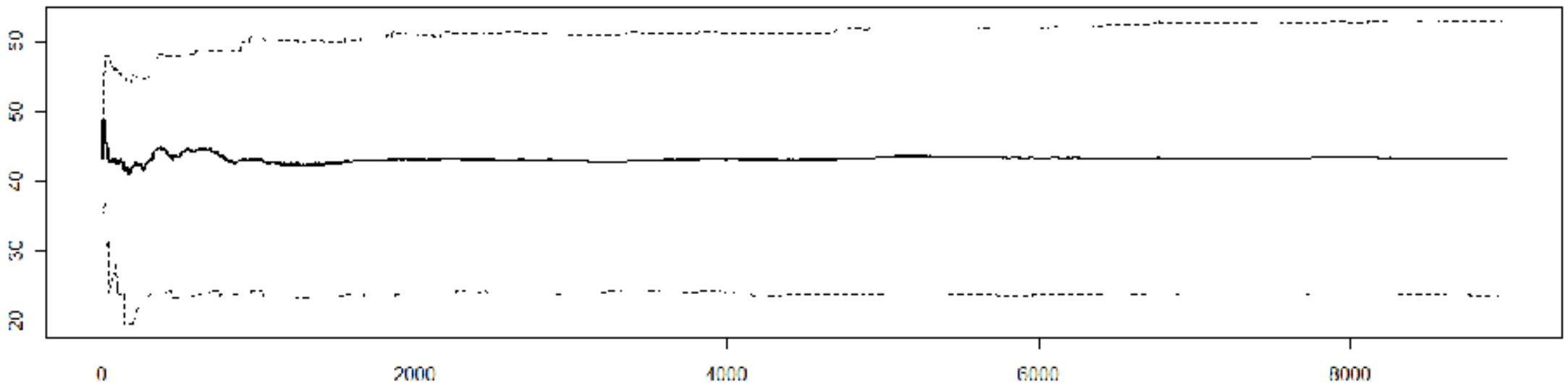

Jump SD = 3
Density
0.01
0.02
0.03
20
40
60
80
mu

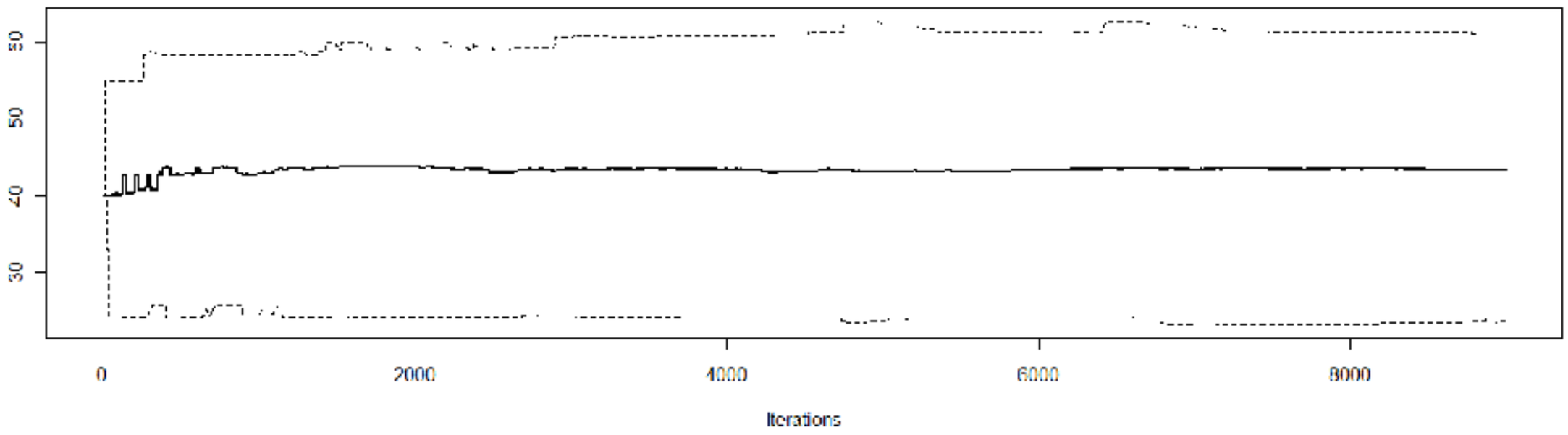

Jump SD = 10
Density
0.01
0.02
0.03
20
40
60
80
mu


Jump SD = 100
Density
0.00
0.02
0.04
20
40
60
80
mu

higher autocorrelation,
lower effective sample size


60
50
40
30
0
2000
4000
6000
8000
Iterations


60
50
40
30
0
2000
4000
6000
8000
Iterations


60
50
40
30
0
2000
4000
6000
8000
Iterations

Metropolis-Hastings

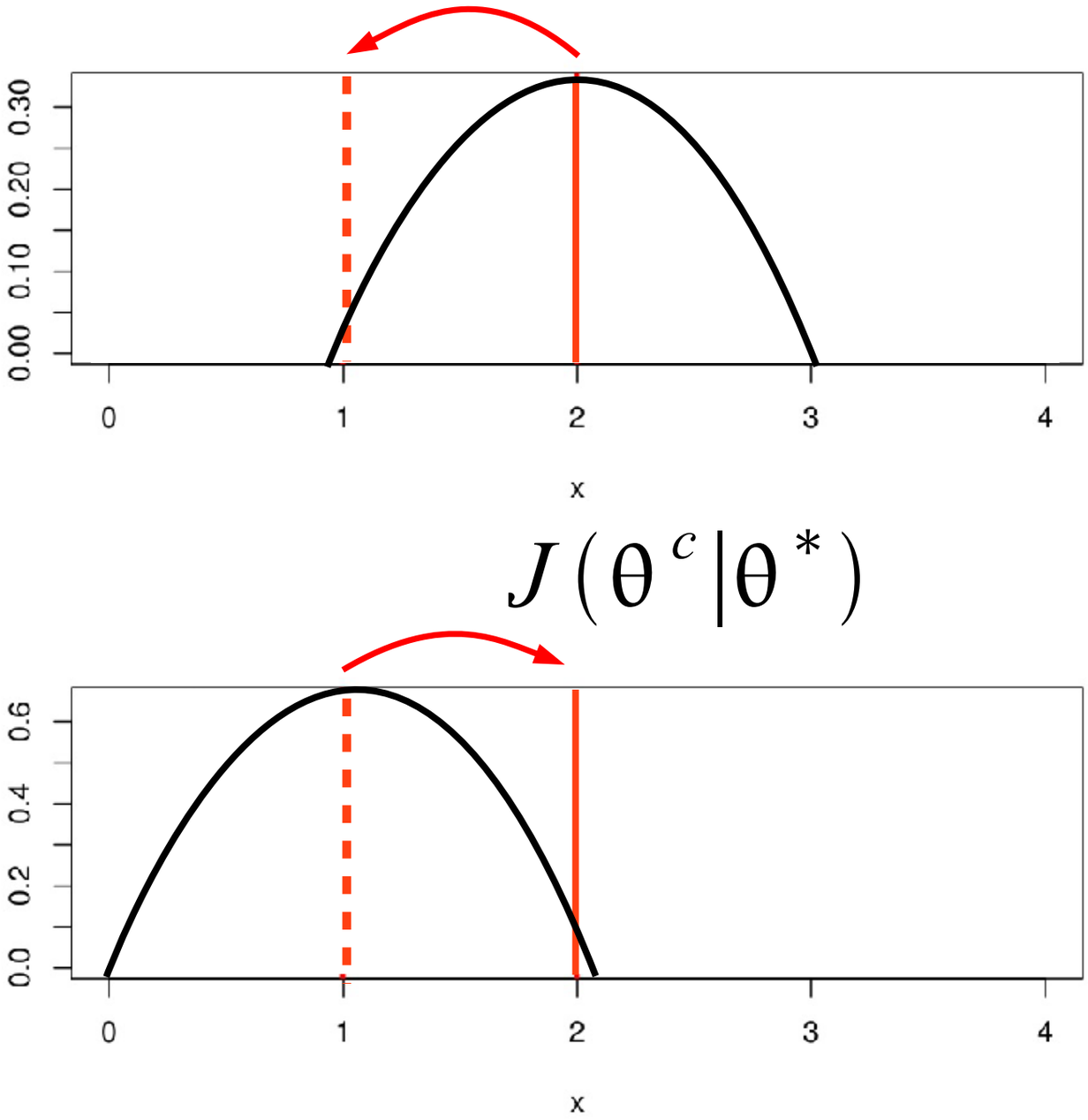
- Generalization of Metropolis
- Allows for asymmetric Jump distribution
- Acceptance criteria

$$a = \frac{p(\theta^*)/J(\theta^*|\theta^c)}{p(\theta^c)/J(\theta^c|\theta^*)}$$

- Most commonly arise due to bounds on parameter values / non-normal Jump distributions

Symmetric proposal function

$$J(\theta^* | \theta^c)$$

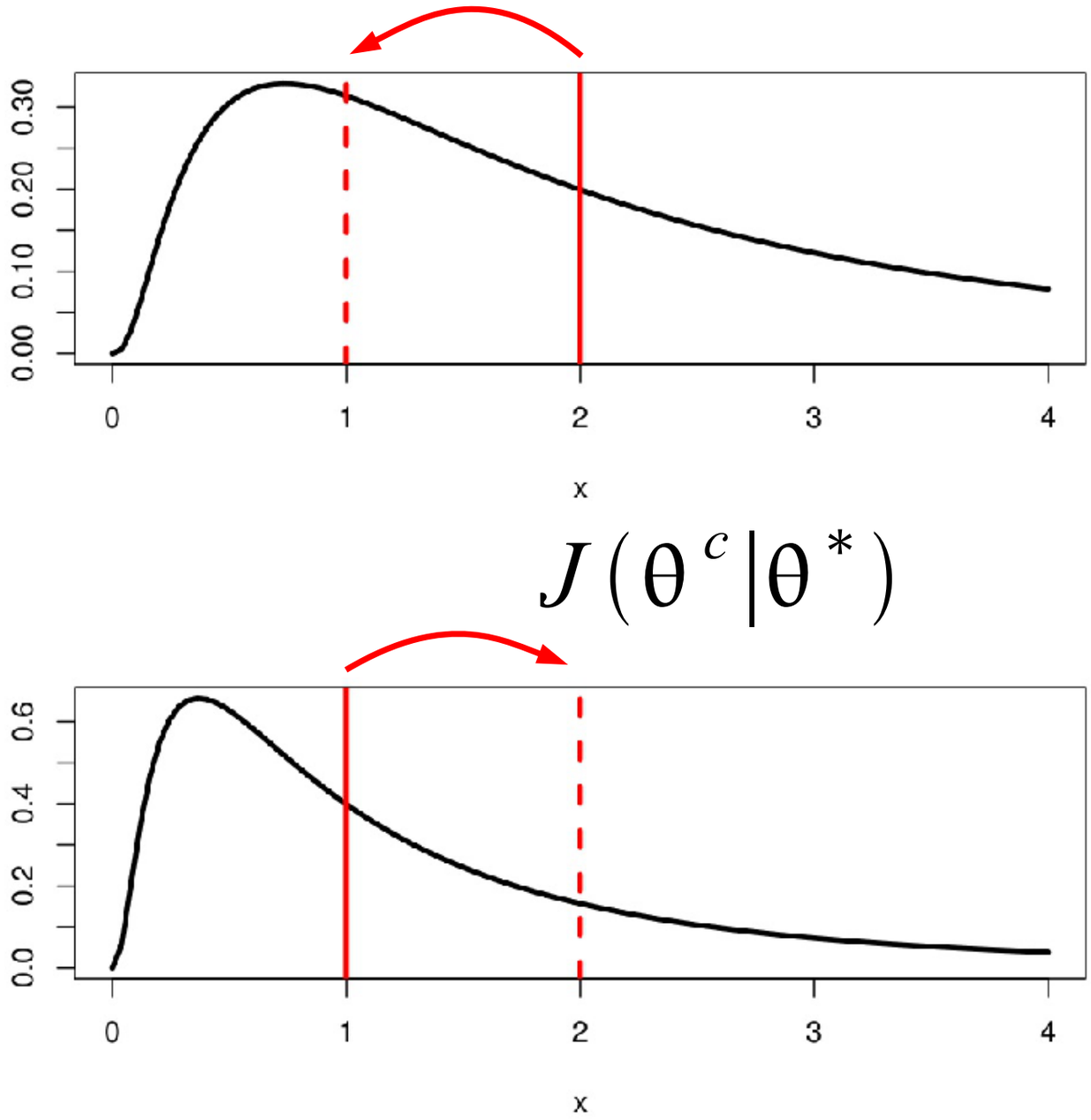


$$J(\theta^c | \theta^*)$$

$$J(\theta^* | \theta^c) = J(\theta^c | \theta^*)$$

Asymmetric proposal function

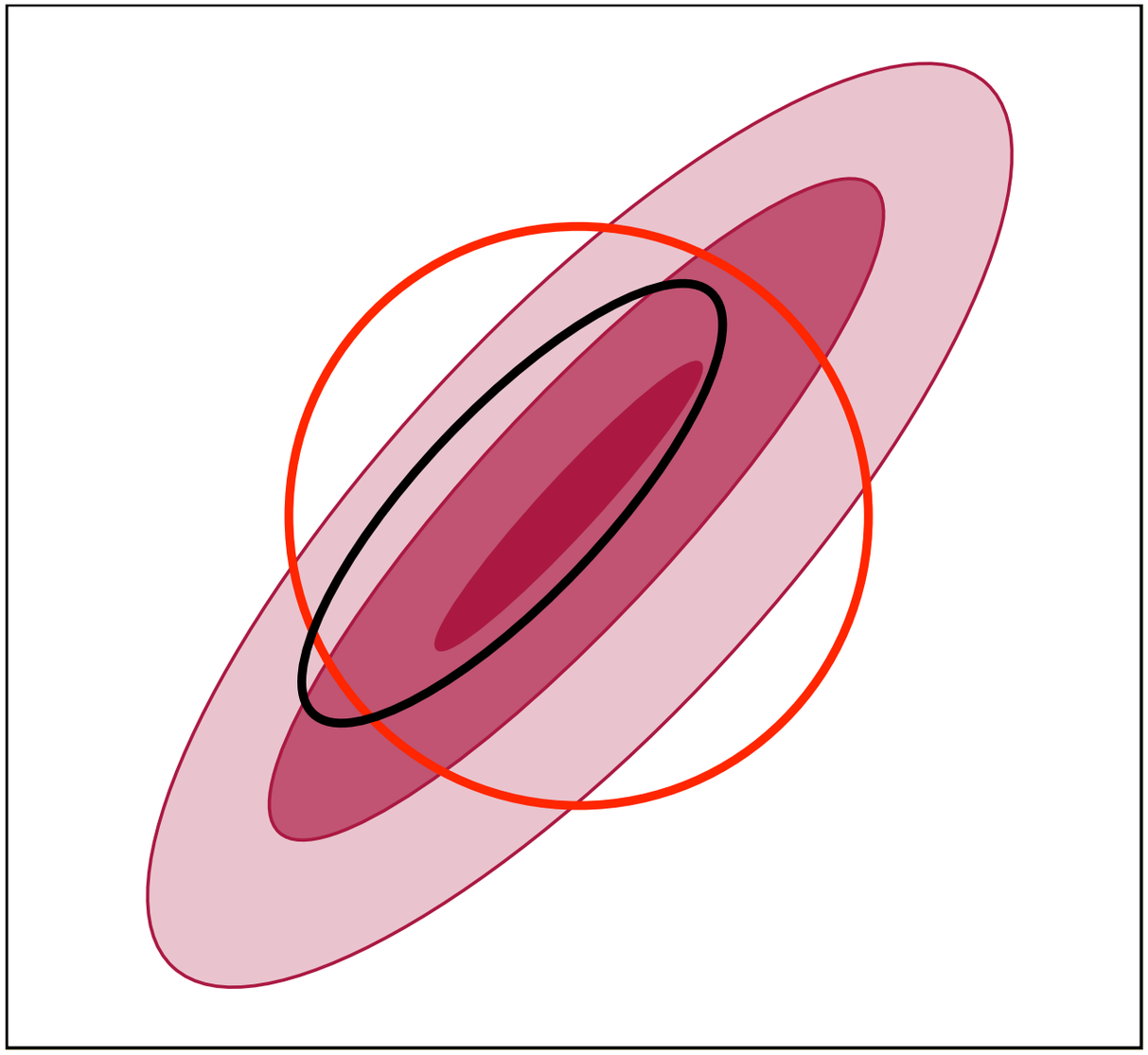
$$J(\theta^* | \theta^c)$$



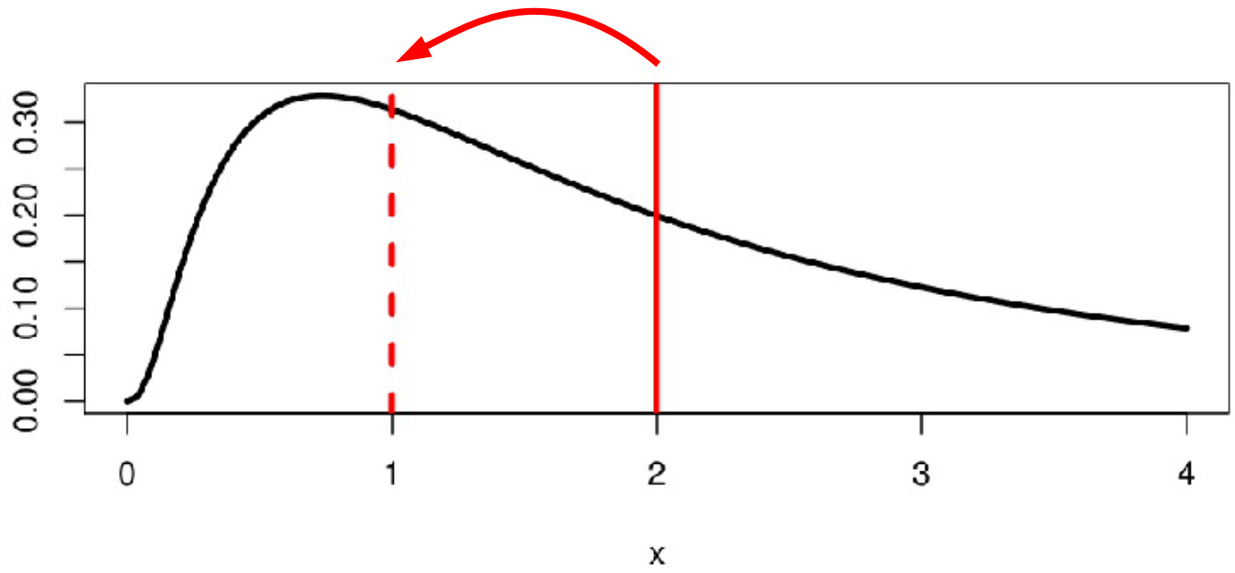
$$J(\theta^c | \theta^*)$$

$$J(\theta^* | \theta^c) \neq J(\theta^c | \theta^*)$$

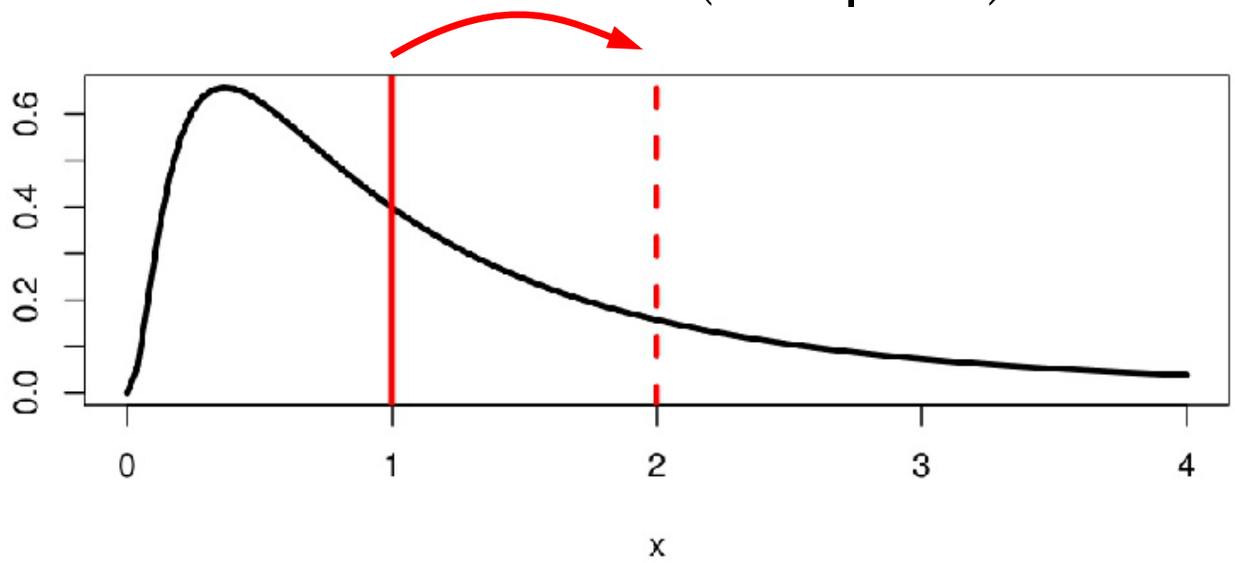
Adjust proposal to posterior shape for effective sampling



$$J(\theta^* | \theta^c)$$



$$J(\theta^c | \theta^*)$$



Multivariate example

- Bivariate Normal $N_2\left(\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}1 & 0.5\\0.5 & 1\end{bmatrix}\right)$
- Option 1: Draw from joint distribution

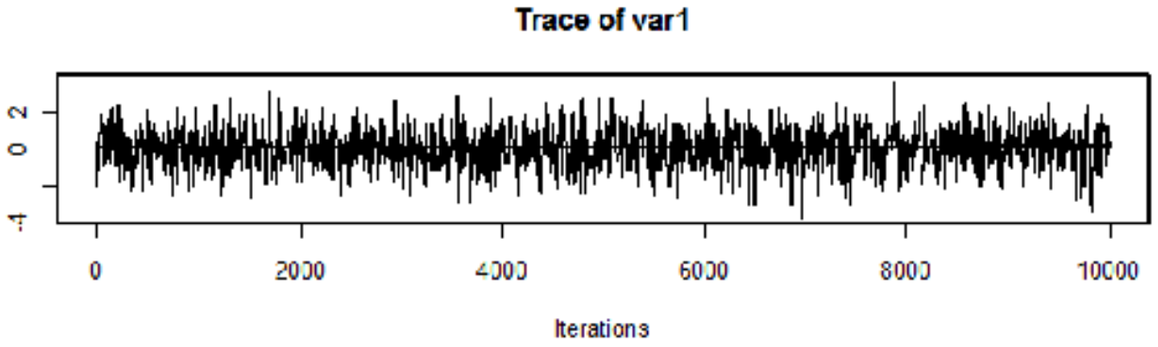
$$J = N_2\left(\begin{bmatrix}\theta_1^*\\\theta_2^*\end{bmatrix} \middle\| \begin{bmatrix}\theta_1^c\\\theta_2^c\end{bmatrix}, V\right)$$

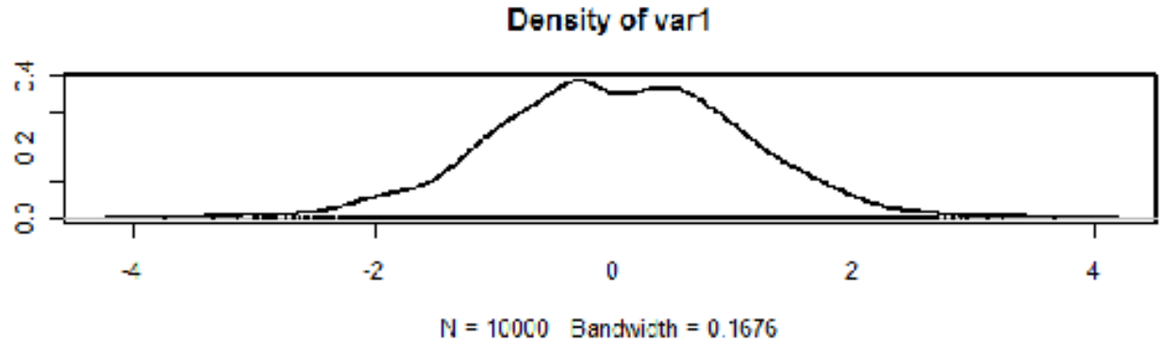
- Option 2: Draw from each parameter iteratively

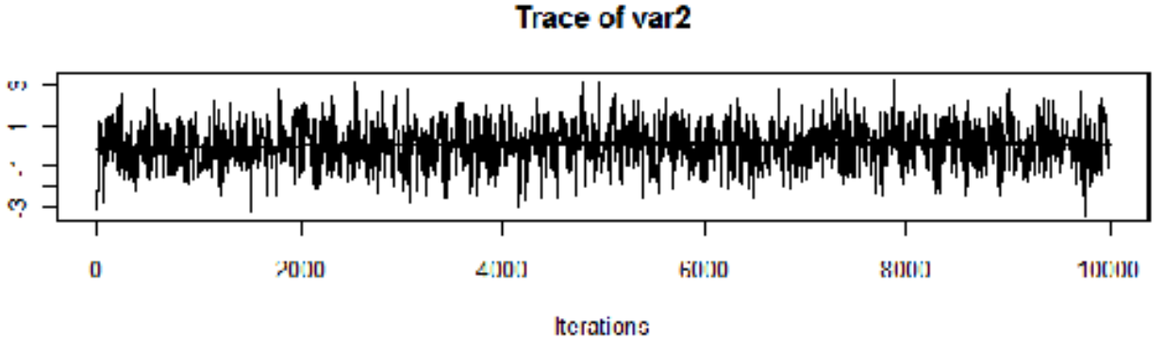
$$J_1 = N(\theta_1^* \mid \theta_1^c, V_1)$$

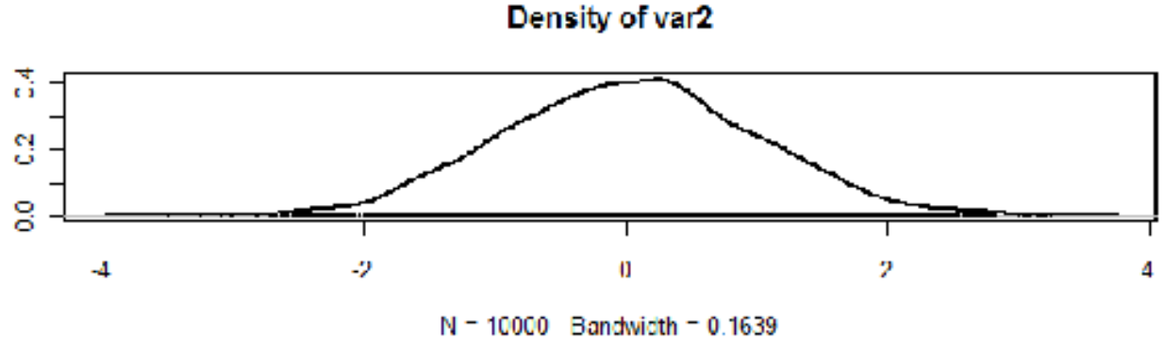
$$J_2 = N(\theta_2^* \mid \theta_2^c, V_2)$$

Joint

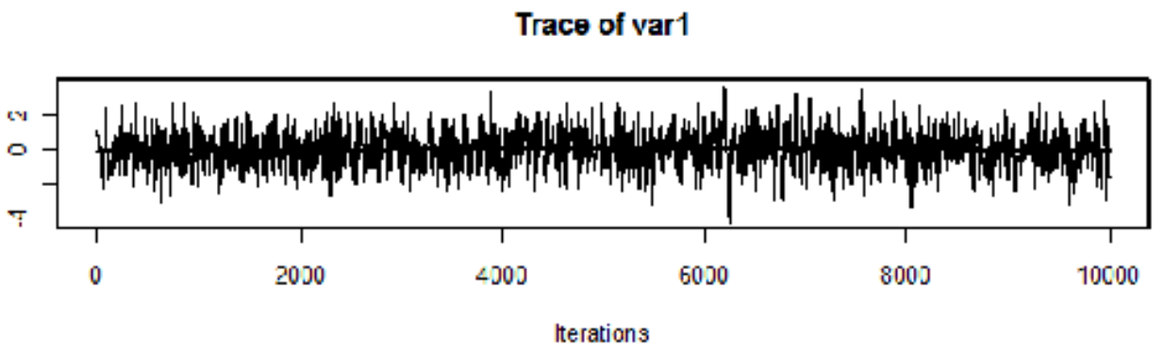


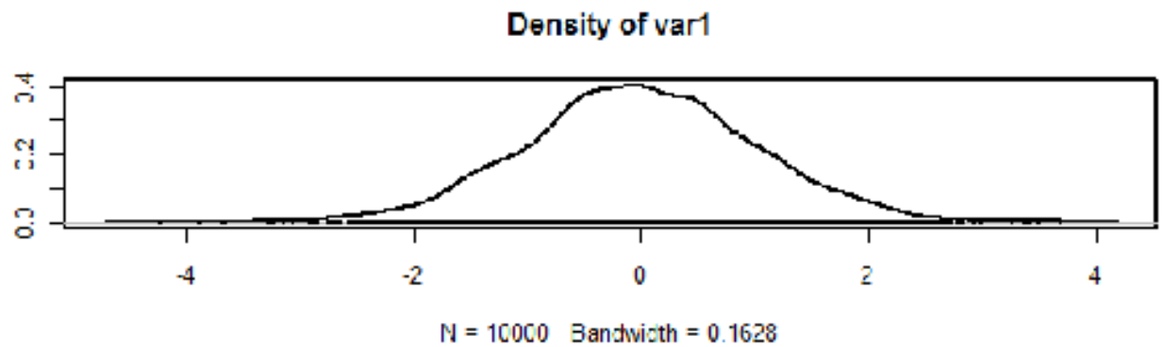


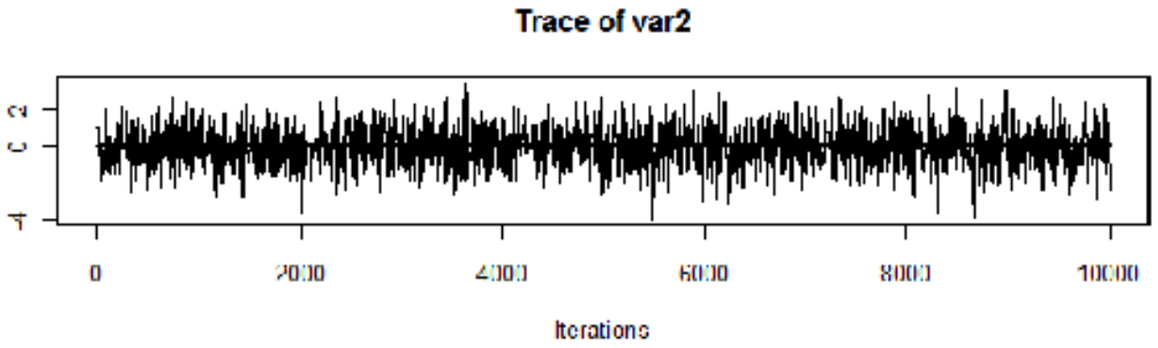


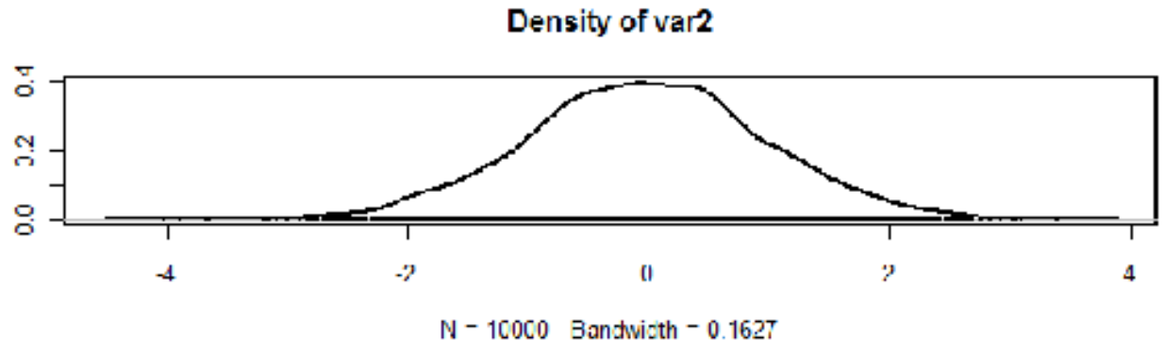


Iterative

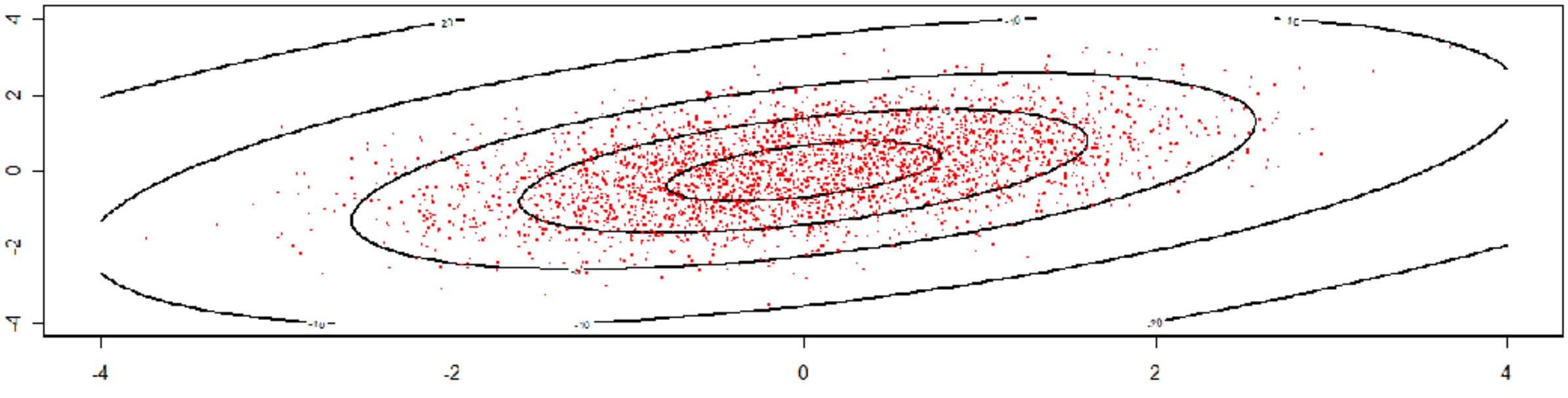




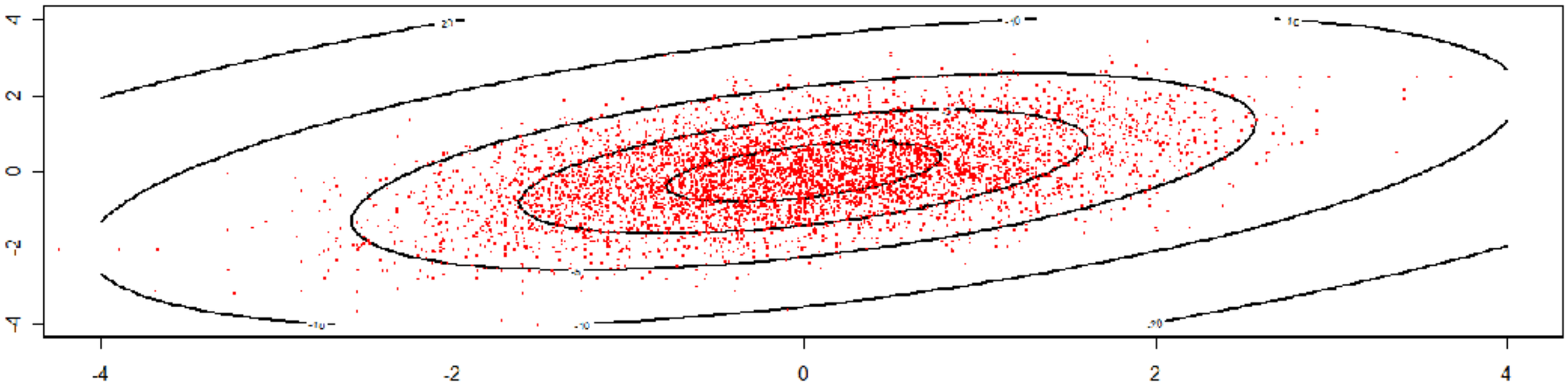




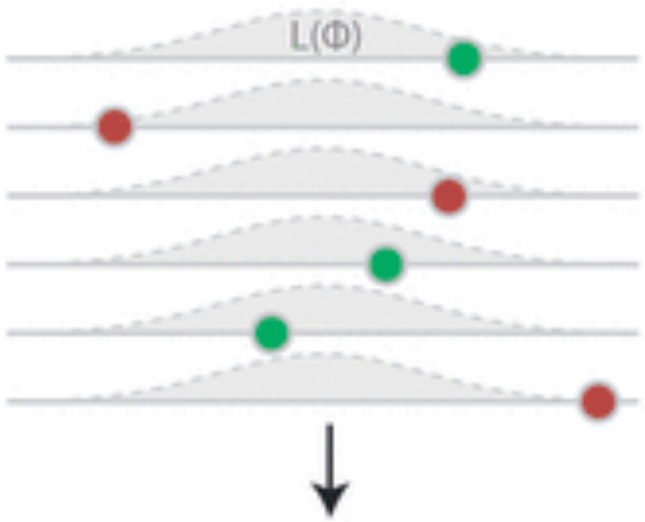
Joint

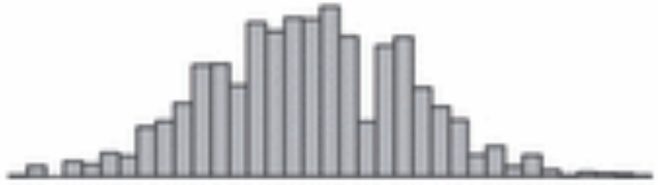


Iterative



Recjection Sampling (REJ)

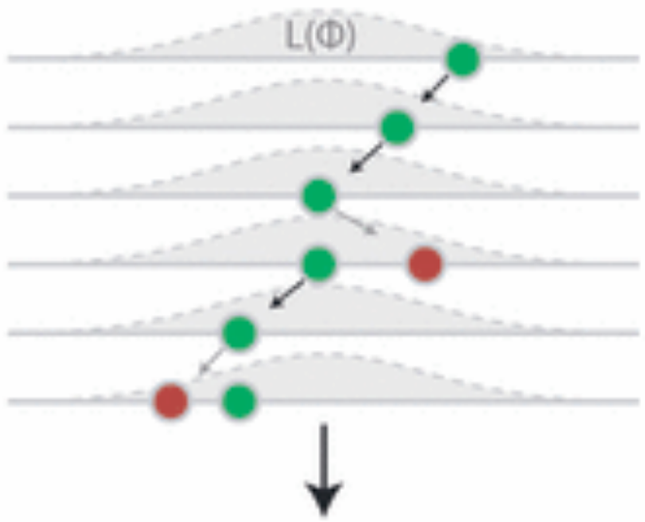


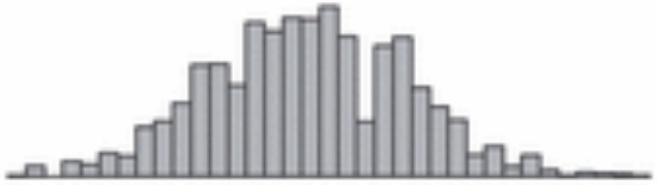


Approximated L(Φ)

1) Draw a parameter Φ
2) Calculate L(Φ)
3) Accept proportional to L(Φ)

MCMC Algorithm

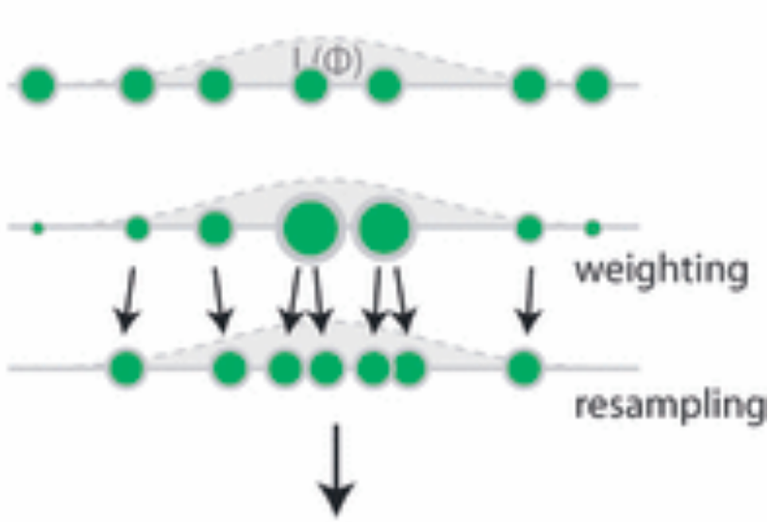


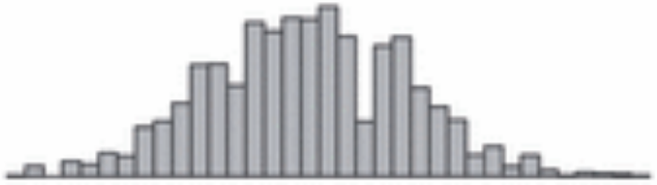


Approximated L(Φ)

1) Draw new parameter Φ' close to the old Φ
2) Calculate L(Φ')
3) Jump proportional to L(Φ')/L(Φ)

SMC Algorithm





Approximated L(Φ)

1) Last set of parameters $\{\Phi_i\}$
2) Assign weight ω_i proportional to $L(\Phi_i)$
3) Draw new $\{\Phi_i\}$ based on the ω_i

Hartig et al. 2011, Ecology Letters

theoreticalecology.wordpress.com