A New Population-based MCMC Method

Di Zhang Advisor: Dr. Yaming Yu

March 5, 2019

Department of Statistics, University of California, Irvine

Outline

Motivation

Markov Chain Monte Carlo Method

- MCMC is a computational tool that is used to generate samples from a probability distribution.
- Construct a Markov chain that has the desired distribution as its stationary distribution.
- Usage: estimate unknown parameters, construct error bars for these estimates, compute the expectation, etc.
- Basic methods: Metropolis-Hastings, Gibbs sampler

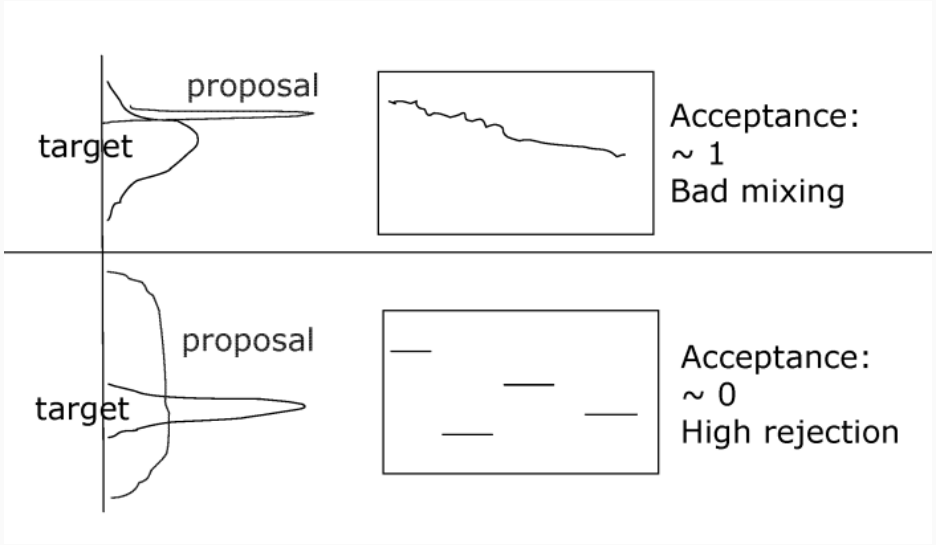
Primary Concern

- A properly derived and implemented MCMC method will produce draws from the joint posterior density once it has converged to stationarity. As a result, a primary concern is to ensure that the chain has converged.
- Poor convergence can happen when chains get stuck near local maxima of the parameter space, which is called the local trap problem.
- Example: Metropolis-Hastings, get stuck in one mode because of its inability to step over valleys of low probability.

Potential solutions

What can we do? (Kass et al., 1997)

- Run the chain longer
- Tune algorithm and make it converge faster
  - In M-H, this depends on the choice of proposal distribution



Problems in the proposal distribution and the corresponding traceplots (Hartig, 2011)

- Alter algorithm in a serious way, e.g. by putting in jumps between modes

Alternative Approach: Population-based MCMC

Population-based MCMC is "a population of Markov chains that run in parallel, each equipped with possibly different but related invariant distributions" (F Liang et al., 2010).

- Adaptive direction sampling (Gilks et al., 1994)
- Parallel tempering (Geyer et al., 1991)
- Evolutionary Monte Carlo (Liang and Wong, 2000)

Interactions between chains allow the information exchange, and this helps the target chains to learn from past samples and in turn improves the convergence.

Methods

Proposed Method: Multiple Chain Method

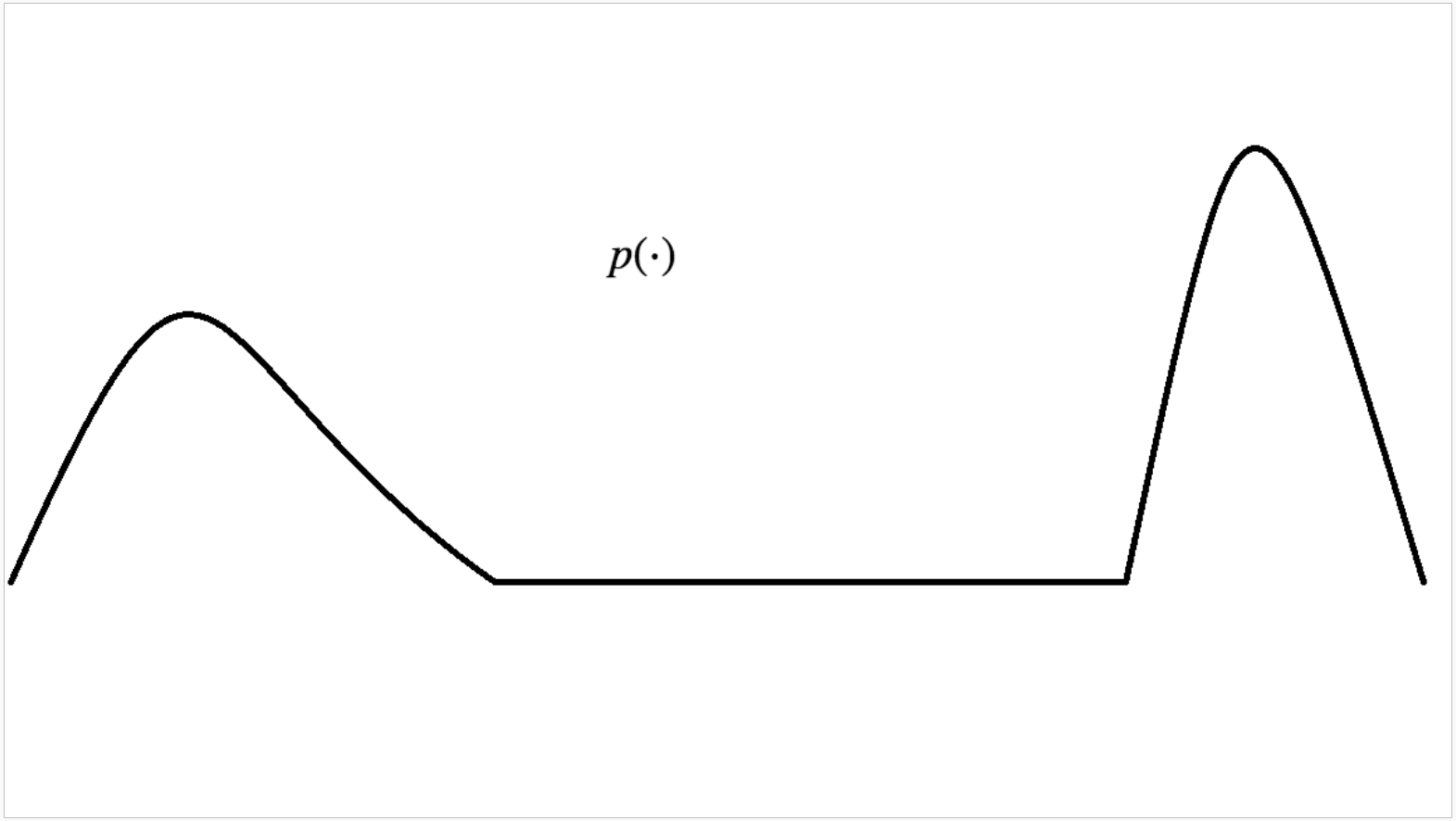
Features:

- Run multiple chains initiated at different modes
- Propose a two-step jump at each iteration

Intuition:

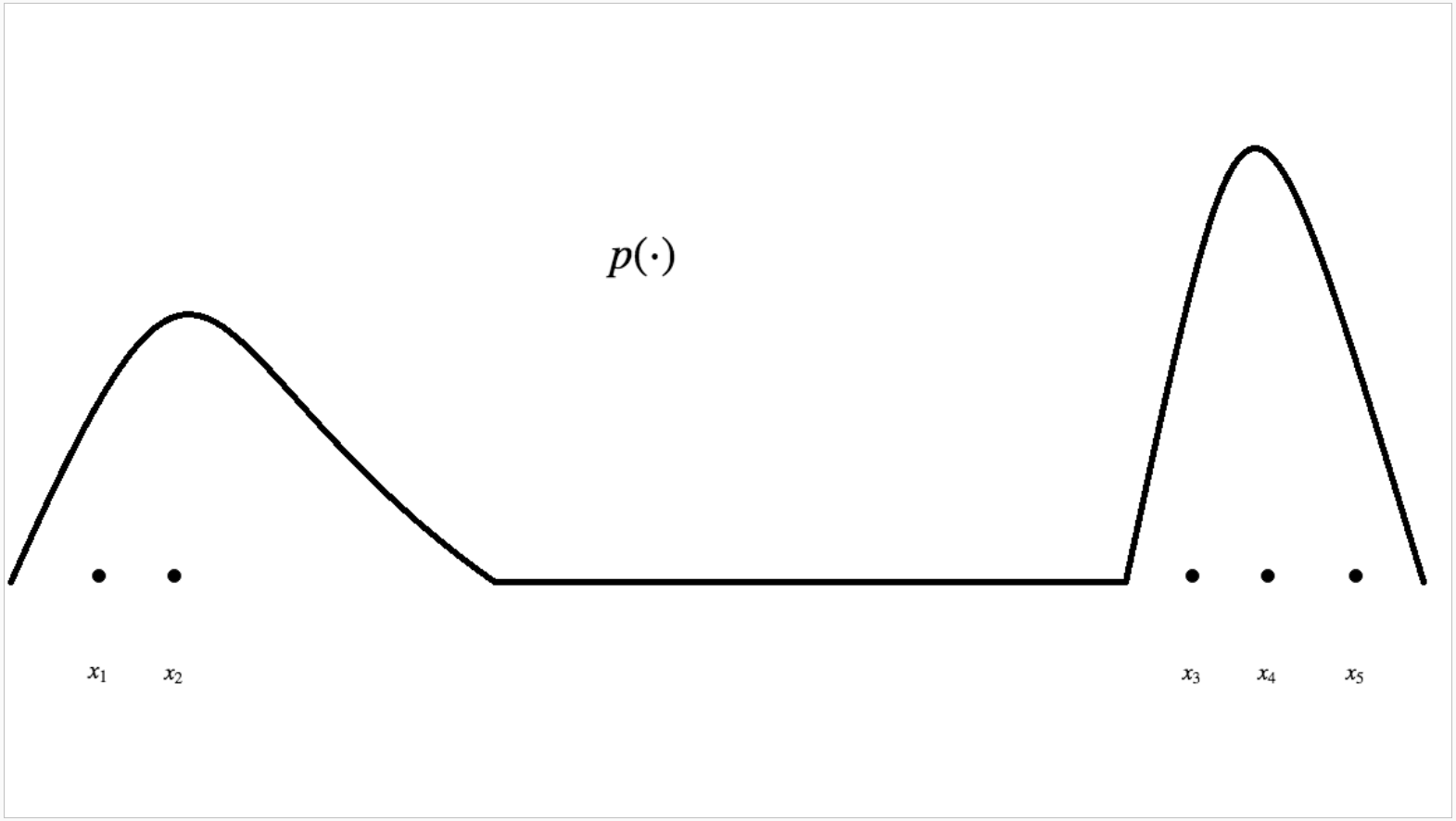
- By running multiple chains starting at different modes and enabling between chain jumps, we propose a population based MCMC method that puts in jumps between modes. Thus we are able to sample from a multi-modal distribution successfully.

Step 0: Target Distribution $p(\cdot)$



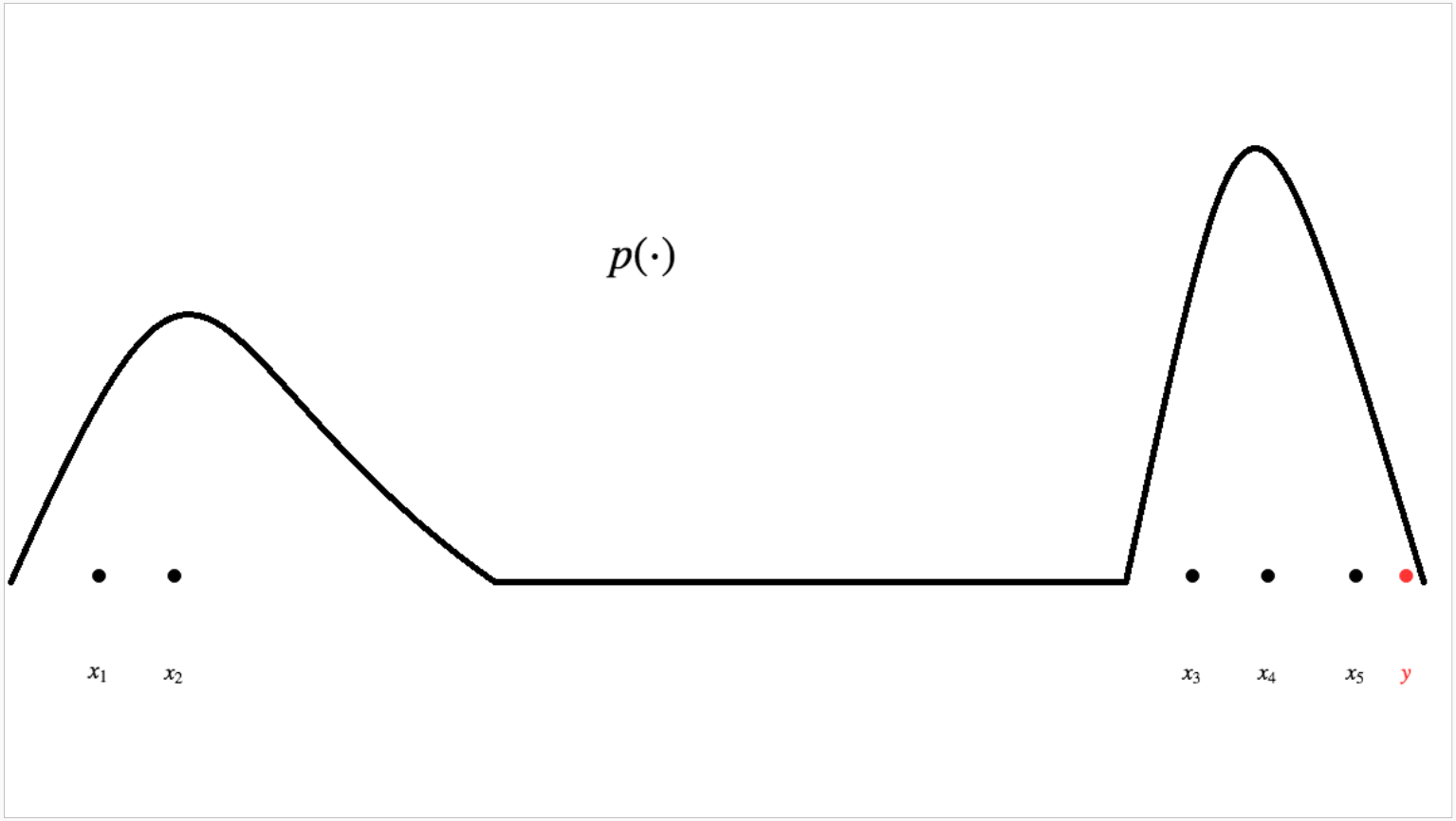
Univariate distribution with two completely separate modes

Step 1: Initialization



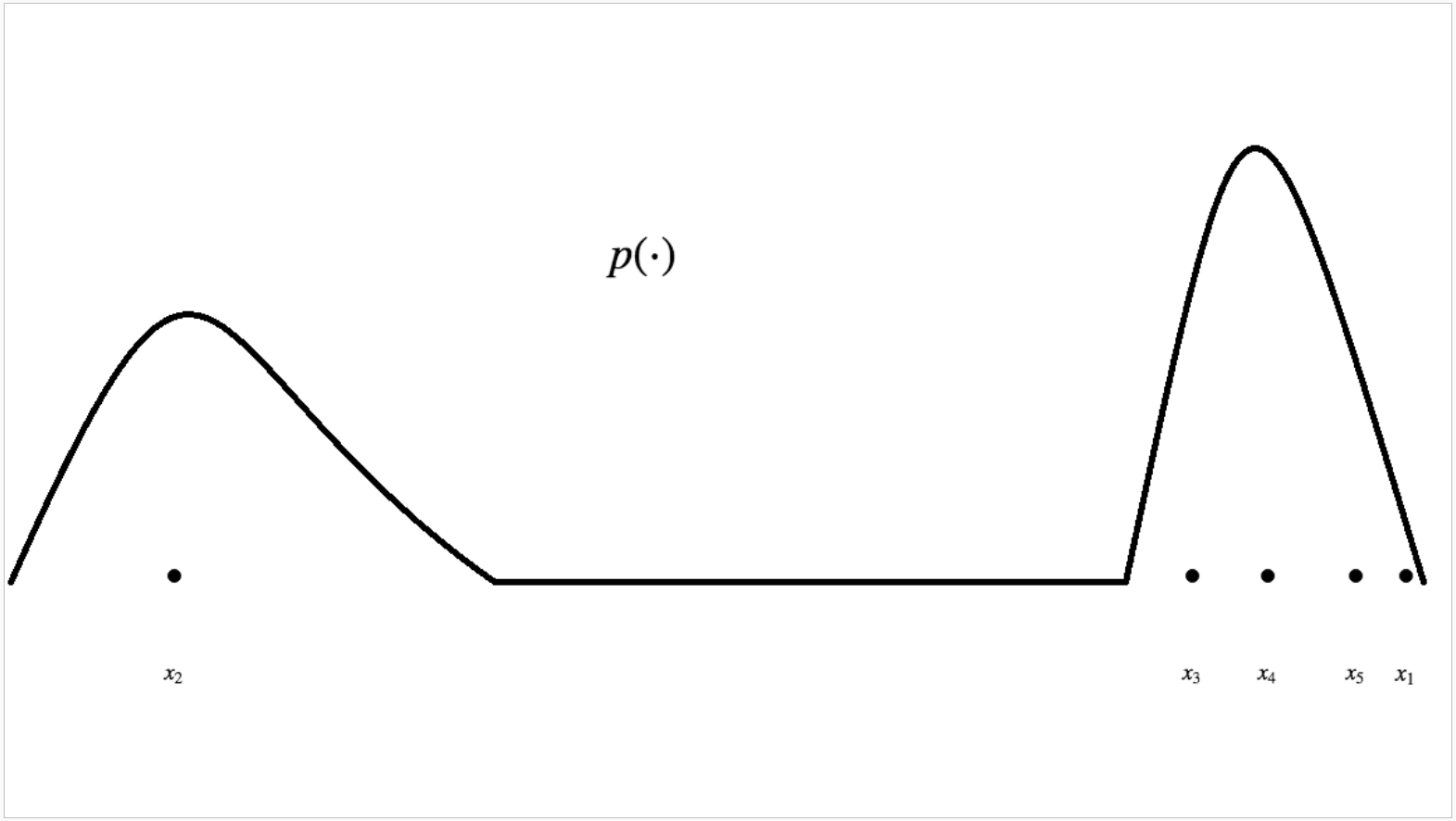
Initialize five chains from dispersed starting values.

Step 2: Update x_1



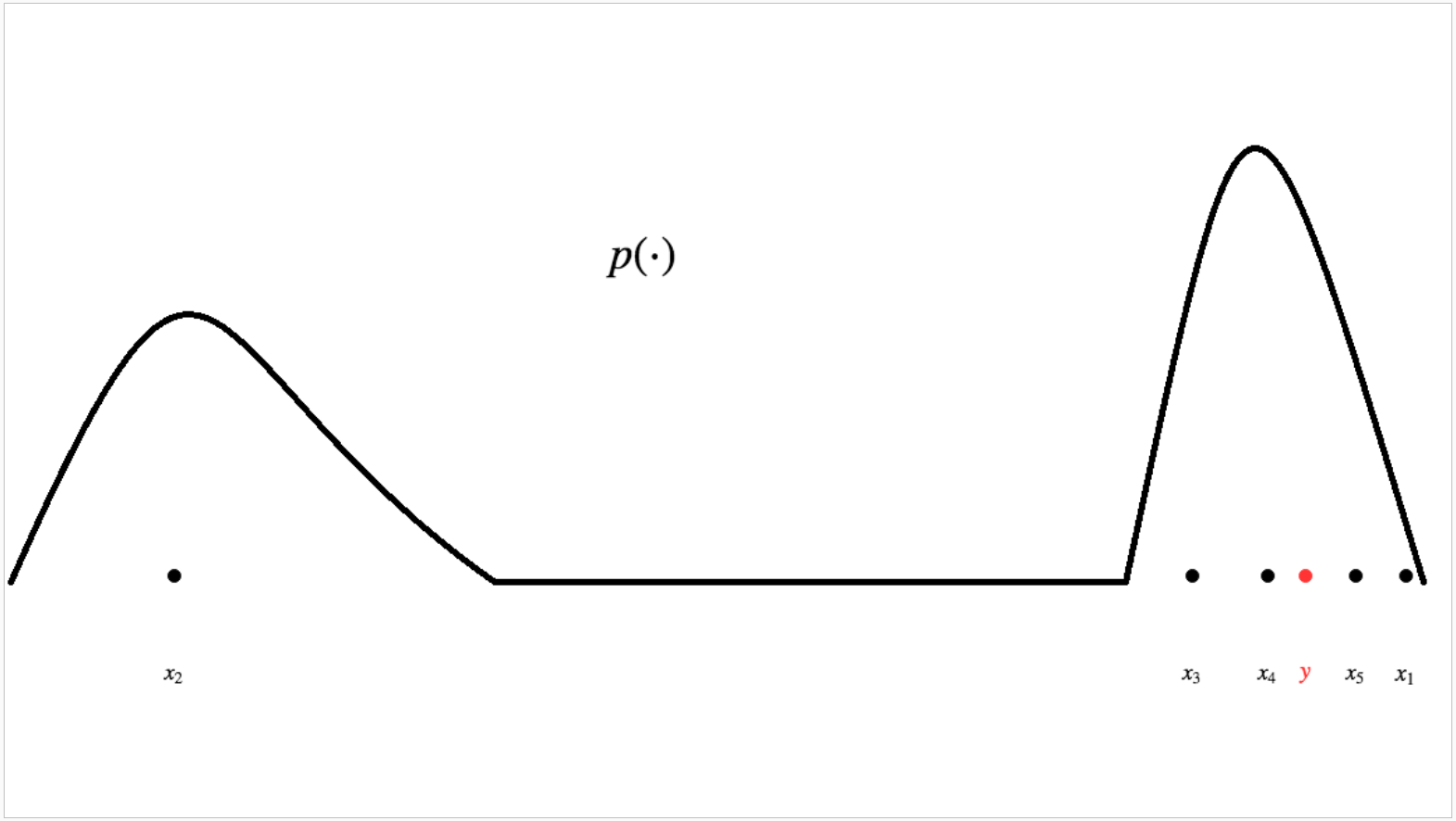
Propose y from the neighborhood of x_4 (randomly selected) and accept/reject according to the Metropolis-Hastings rule.

Step 2: Update x_1



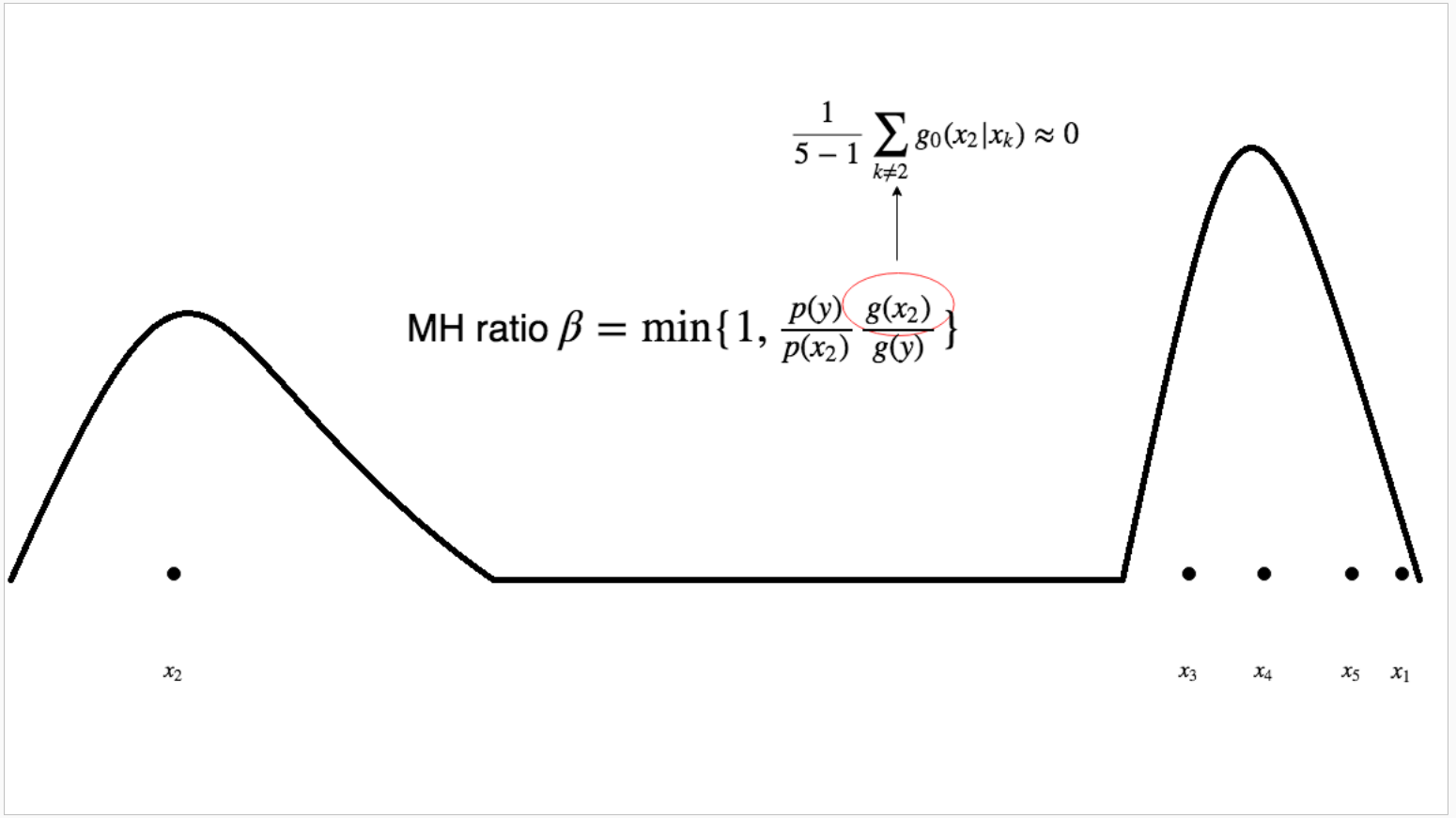
A successful between chain jump!

Step 3: Update x_2



Propose y from the neighborhood of x_1 (randomly selected) and reject according to the Metropolis-Hastings rule.

Step 3: Update x_2



$g_0(\cdot|x_k)$ proposes points from the neighborhood of x_k thus $g_0(x_2|x_k)$ is almost 0 as they are from completely separate modes.

Algorithm

Initialize $x_1^0, \ldots x_m^0 \sim \pi(\cdot)$ // $\pi(\cdot)$ is the initial distribution
for $t = 0$ to $N - 1$ **do**
  for $i = 1$ to m **do**
    $x \leftarrow X \sim q(\cdot|x_i^t)$
    $\alpha \leftarrow \min\left(1, \frac{p(x)}{p(x_i^t)}\right)$
    $u \leftarrow U \sim U[0, 1]$
    if $u \leq \alpha$ **then**
      $x_i^t \leftarrow x$
    end if // within chain jump
    $j \leftarrow S \sim \{1, 2, \ldots, m\} \backslash \{i\}$
    $y \leftarrow Y \sim g_0(y|x_j^t)$
    $\beta \leftarrow \min\left(1, \frac{p(y)}{p(x_i^t)} \frac{g_i(x_i^t)}{g_i(y)}\right)$
    $v \leftarrow U \sim U[0, 1]$
    if $v \leq \beta$ **then**
      $x_i^{t+1} \leftarrow y$
    else
      $x_i^{t+1} \leftarrow x_i^t$
    end if // between chain jump
  end for
end for

Algorithm

STEP 1 Initialize m chains from dispersed starting values.

STEP 2 For each chain i, propose a two-step jump at each iteation t.

1. Propose a candidate point x by, e.g., usual random walk and accept by Metropolis rule. (within chain jump)
2. Randomly select another chain j, propose a candidate point y from the neighborhood of the current state x_j^t of chain j, and accept by MH jumping rule. The MH ratio would be $\beta = \min\left(1, \frac{p(y)}{p(x_i^t)} \frac{g_i(x_i^t)}{g_i(y)}\right)$, where $p(\cdot)$ is the target distribution and $g_i(y) = \frac{1}{m-1} \sum_{k \neq i} g_0(y|x_k^t)$. (between chain jump)

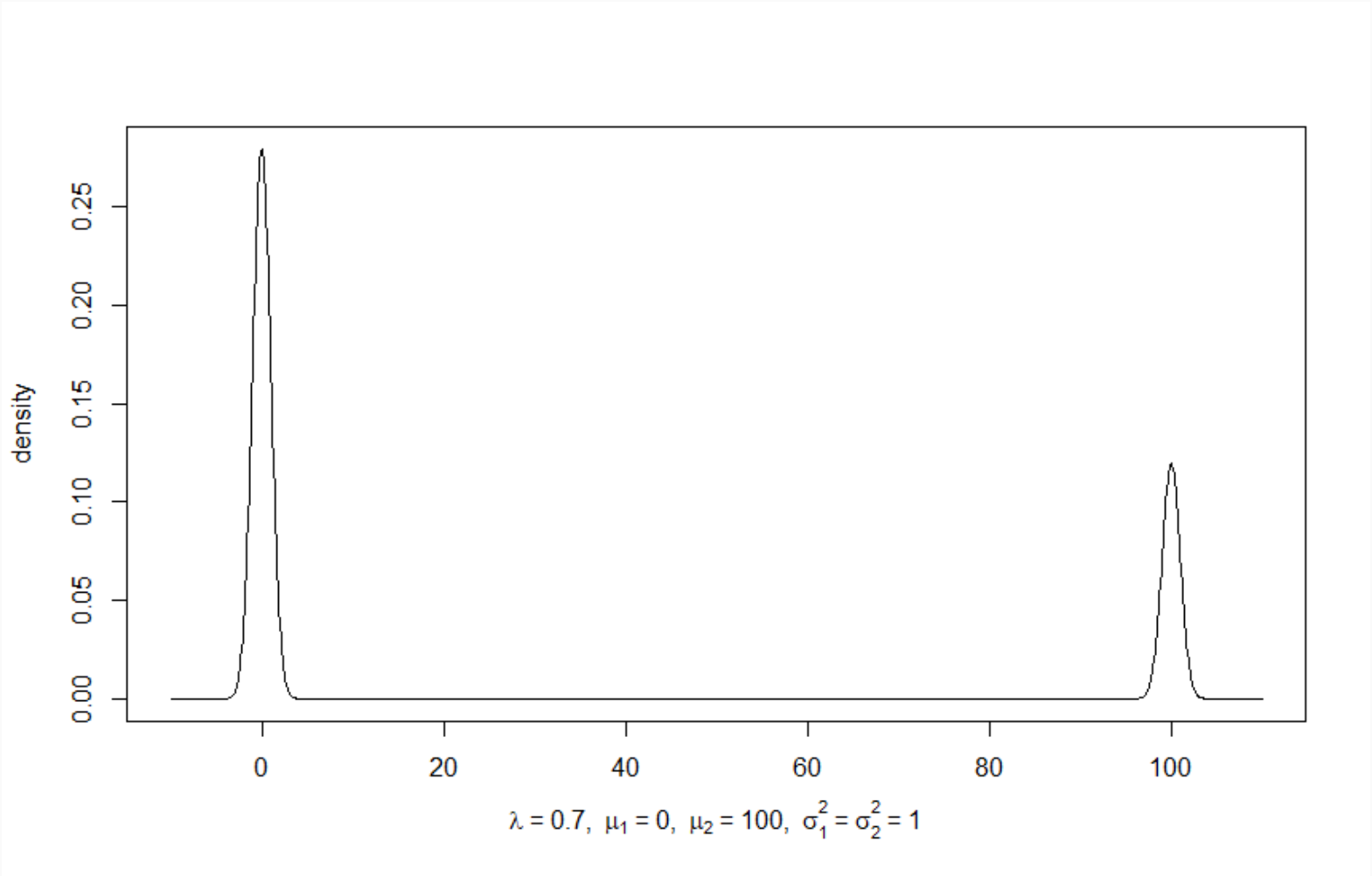
Remark

- We actually propose a new proposal function $g(\cdot|\cdot)$ in between chain jump step, which is the average of densities of the proposed point evaluated at the current iterates of the remaining chains.
- Multiple chain method allows trans-dimensional moves. (see example 2)

Toy Example

A mixture of two normals:

$$\lambda N(\mu_1, \sigma_1^2) + (1 - \lambda) N(\mu_2, \sigma_2^2)$$



Results

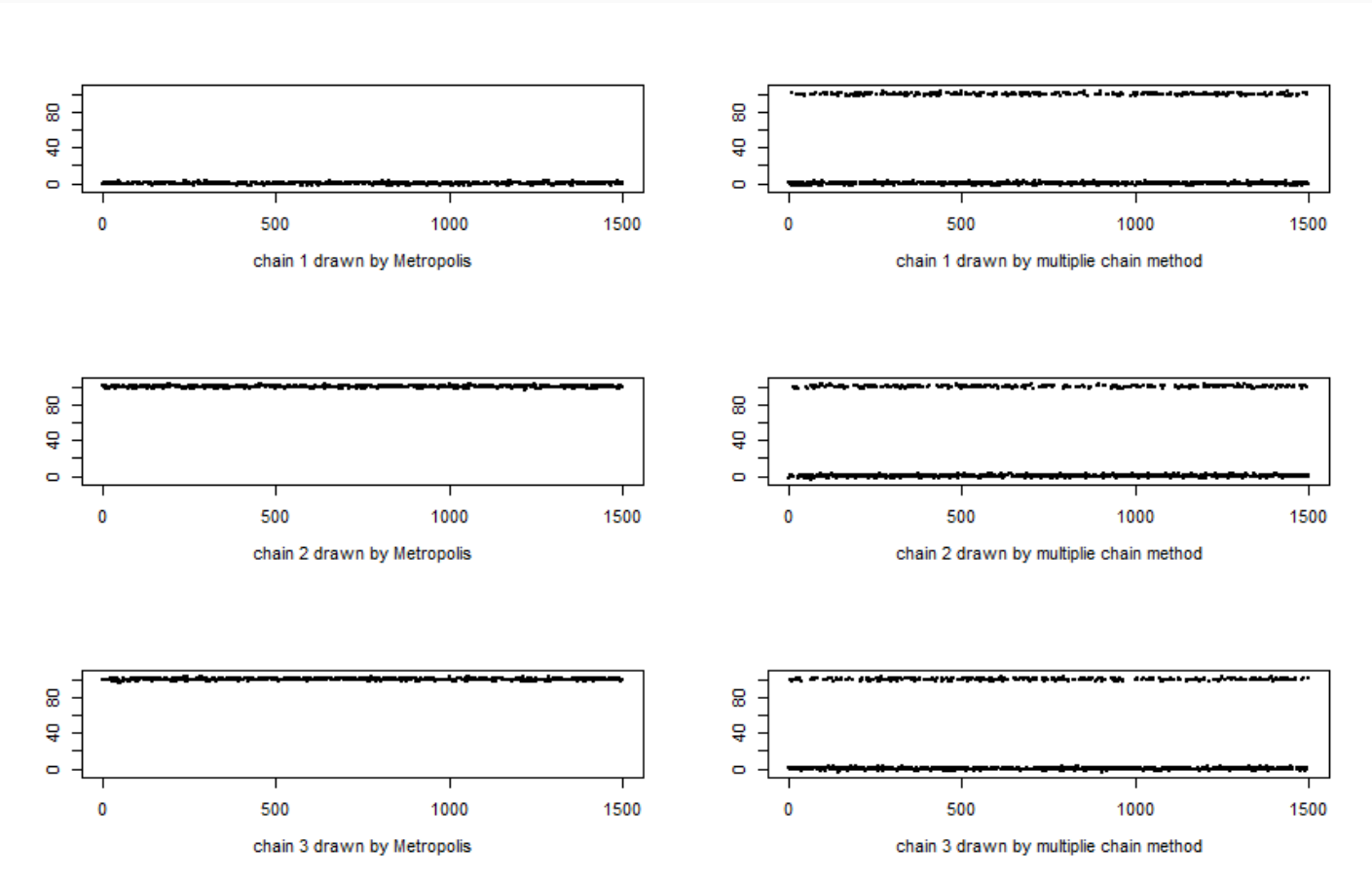


Figure 1: Dotplots of draws comparing Metropolis (left) and multiple chain method (right).

Metropolis is slow to converge in this situation.

Example: Bayesian Model Selection

- Given data Y, choose between two models M_1 and M_2.
- Make decisions by the Bayes factor (Kass, 1995):
  - Notation: $B_{21} = \frac{Pr(Y|M_2)}{Pr(Y|M_1)}$
  - Interpretation: Evidence provided by the data Y in favor of M_2 against M_1
  - How do we compute B_{21}?
- Example:
  - Model 1 (M_1):

$$Y_i|\mu \sim N(\mu, \sigma^2), \quad i = 1, 2, ...n$$

    where $\mu|\mu_0, \sigma_0^2 \sim N(\mu_0, \sigma_0^2)$
  - Model 2 (M_2):

$$Y_i|\alpha, \beta, X_i \sim N(\alpha + \beta X_i, \sigma^2), \quad i = 1, 2, ...n$$

    where $(\alpha, \beta)^T|\alpha_0, \beta_0, \Sigma \sim N_2((\alpha_0, \beta_0)^T, \Sigma)$ and Σ=diag(σ_0^2, τ_0^2).
  - $X_i's$ are known covariates and σ^2 is known.

Standard Approach

- Target: $B_{21} = \frac{Pr(Y|M_2)}{Pr(Y|M_1)}$
- Compute the densities $Pr(Y|M_i)(i = 1, 2)$ by integrating out the model parameters, i.e.

$$Pr(Y|M_i) = \int Pr(Y|\theta_i, M_i)\pi(\theta_i|M_i)d\theta_i$$

  - θ_i is the collection of parameters under M_i
  - $\pi(\theta_i|M_i)$ is the prior density of θ_i
  - $Pr(Y|\theta_i, M_i)$ is the probability density of Y given θ_i
- In this example, the integral can be computed analytically and it turns out that:
  - M_1: $Y|M_1 \sim N(\vec{1}\alpha_0, \sigma^2 I + \sigma_0^2 \vec{1}\vec{1}^T)$.
  - M_2: $Y|M_2 \sim N((\vec{1}, \vec{X_0})(\alpha_0, \beta_0)^T, \sigma^2 I + (\vec{1}, \vec{X_0})\Sigma(\vec{1}, \vec{X_0})^T)$.

  And $B_{21} = \frac{Pr(Y|M_2)}{Pr(Y|M_1)}$ is then computed.

Multiple Chain Method

- Write the joint posterior density by introducing parameter I that indexes the model collection, i.e.

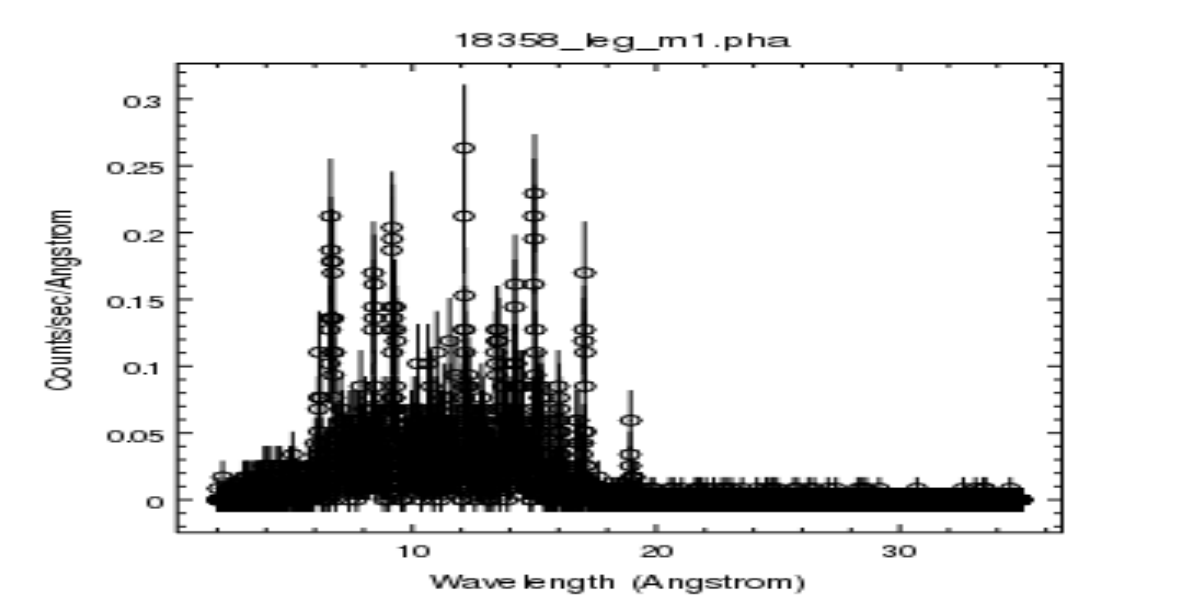
$$Pr(I, \theta_1, \theta_2|Y) \propto \prod_{i=1}^{2} [Pr(Y|\theta_i, M_i)\pi(\theta_i|M_i)Pr(M_i)]^{\mathbb{1}_{\{I=i\}}}$$

  - $Pr(M_1)$ is the prior probability that data Y comes from M_1 and $Pr(M_2) = 1 - Pr(M_1)$.
  - I is the model indicator that specifies data Y comes from M_1 if $I = 1$, or comes from M_2 if $I = 2$.
- Sample from $Pr(I, \theta_1, \theta_2|Y)$ by applying multiple chain method. Note that trans-dimensional jumps are made between M_1 and M_2, since $\dim(\theta_2) = \dim(\theta_1) + 1$.
- Compute the posterior probability $Pr(M_i|Y)$ by simply counting the frequency of samples that come from M_i, $\frac{\#of\{I=i\}}{N_{rep}}$.
- Recall that $\frac{Pr(M_2|Y)}{Pr(M_1|Y)} = B_{21} \times \frac{Pr(M_2)}{Pr(M_1)}$, we can get B_{21}.
- Our method agrees with the theoretical derivation.

Applications

Example: Chandra Data

- Source: Capella
  - Strongest non-solar coronal source accessible to X-ray telescopes
  - Very stable and the overall luminosity has been steady for many years with no discernible flaring activity
- Instrument: Chandra ACIS-S
  - Obtained a set of contiguous observations of Capella during July 2016 (ObsID: 18358-18364)



Temperature Models

Multiplicative model: xsphabs * xsvapec

- xsphabs: photo-electric absorption

Number	Name	Description
1	nh	equivalent hydrogen column (in units of 10^22 atoms/cm^2)

xsphabs Parameters (Source: Sherpa help page, CXC/SHERPA/AHELP)

- xsvapec: thermal plasma model with variable abundances

Number	Name	Description
1	kT	plasma temperature (keV)
2-14	(element)	Abundances for He, C, N, O, Ne, Mg, Al, Si, S, Ar, Ca, Fe, Ni with respect to Solar (defined by the set_xsabund command). The trace element abundances are from the set_xsxset APEC_TRACE_ABUND value, the default is 1.0.
15	redshift	redshift, z
16	norm	10^-14 / (4 pi (D_A*(1+z))^2) Int n_e n_H dV, where D_A is the angular diameter distance to the source (cm), n_e is the electron density (cm^-3), and n_H is the Hydrogen density (cm^-3)

xsvapec Parameters (Source: Sherpa help page, CXC/SHERPA/AHELP)

- The parameter of interest is the temperature (xsvapec.kT)

Temperature Models

- xsphabs: power law absorption model
- xsvapec: thermal plasma model, describing the optically thin[*] line[**] and continuum[***] emission from collisionally excited[****] plasma[*****].
  - * The probability that a photon is intercepted after emission by another ion nearby is negligible
  - ** Transitions between quantized energy levels in an ion
  - *** Transitions that occur when free electrons are absorbed by an ion or when electrons scatter from each other
  - **** Upper energy levels of the ions are populated through inelastic collisions]
  - ***** Ionized gas

Temperature Models

- We expect there to be a continuum of temperature components, with the most prominent one probably at 0.54 keV
- Sherpa fit() failed to recover this, so we try MCMC method
- First think of just one temperature component model: xsphabs.abs1*xsvapec.kT1
- Then two temperature components: xsphabs.abs1*(xsvapec.kT1+xsvapec.kT2)

xsphabs.abs1*xsvapec.kT1

- kT1.He is set to 1 and other element abundances are fixed at 0.6. Thaw Fe and we have four parameters: abs1.nH, kT1.kT, kT1.Fe and kT1.norm.
- For single chain method (usual Metropolis), run 20 parallel chains for 5,000 iterations and some chains are stuck.

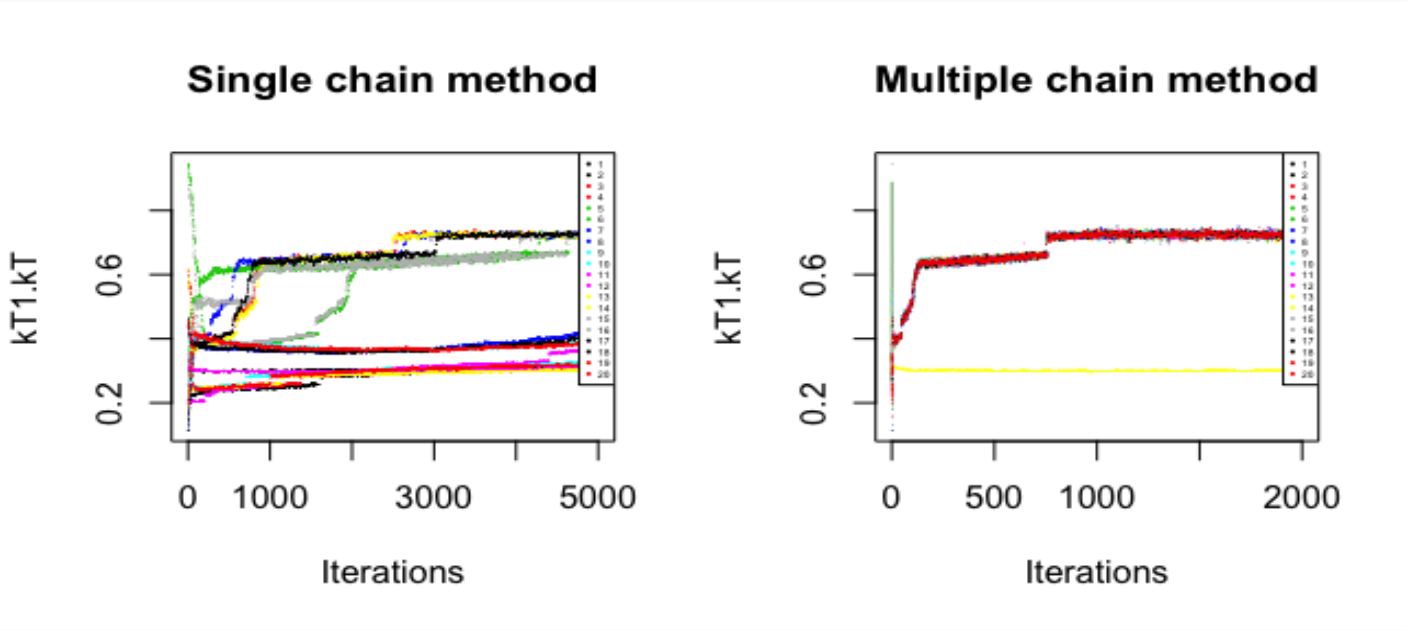


Figure 2: Traceplots of kT1.kT comparing single chain (left) and multiple chain method (right)

xsphabs.abs1*xsvapec.kT1

- For multiple chain method, 19 chains go to the same place whereas 1 chain stays somewhere else.
- No jumping between them as the likelihood of the mode is much higher.
- Find only one mode where kT1.kT $\approx$ 0.72 keV.
- How about the more realistic model with two temperature components?

xsphabs.abs1*(xsvapec.kT1+xsvapec.kT2)

- kT1.He is set to 1 and other element abundances are fixed at 0.6. Link norm, thaw Fe and we have five parameters: abs1.nH, kT1.kT, kT1.Fe, kT1.norm and kT2.kT.
- Run 20 parallel chains from two potential modes for 2,000 iterations, where both modes are discovered by the standard fitting.
- Single chain method stays at the mode where it starts, not helpful to tell how much proportion one mode will contribute to the whole distribution.
- Multiple chain method tells that the minor mode seems not important at all compared with the major mode.

xsphabs.abs1*(xsvapec.kT1+xsvapec.kT2)

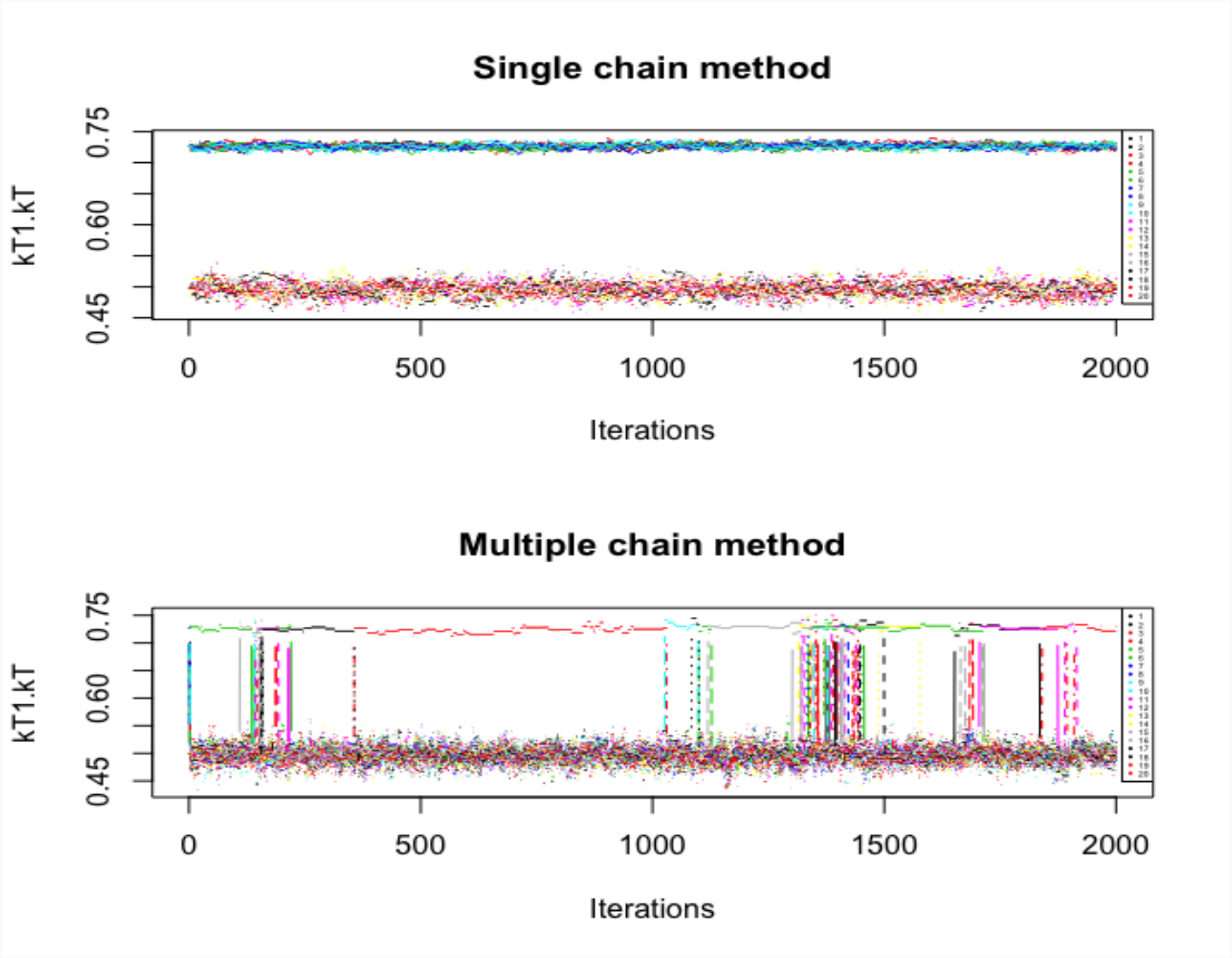


Figure 3: Traceplots of kT1.kT comparing single chain (top) and multiple chain method (bottom)

Label Switching

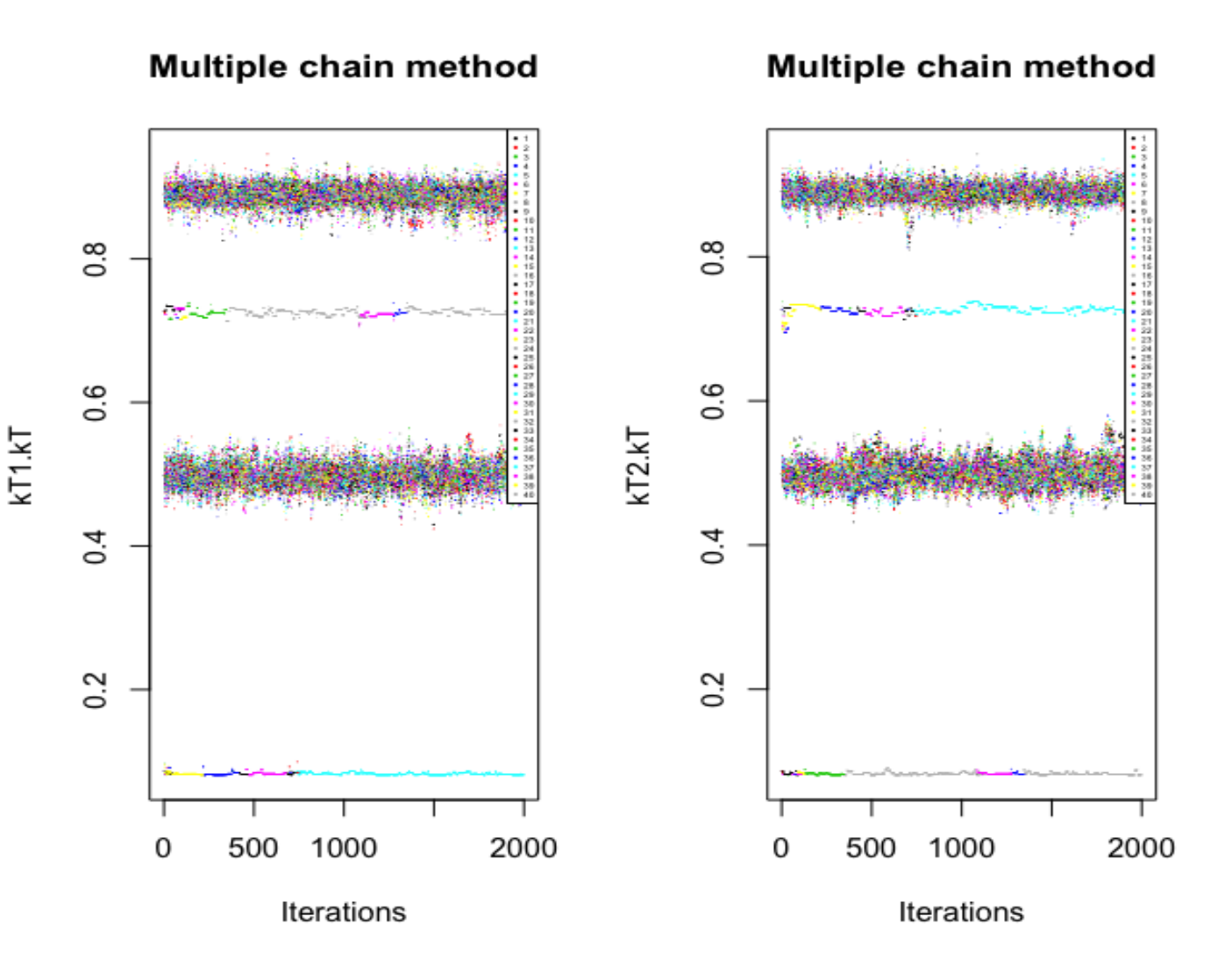


Figure 4: Dotplots of kT1.kT (left) and kT2.kT (right) by running 40 chains from four potential modes (swapping kT1.kT and kT2.kT).

Label Switching

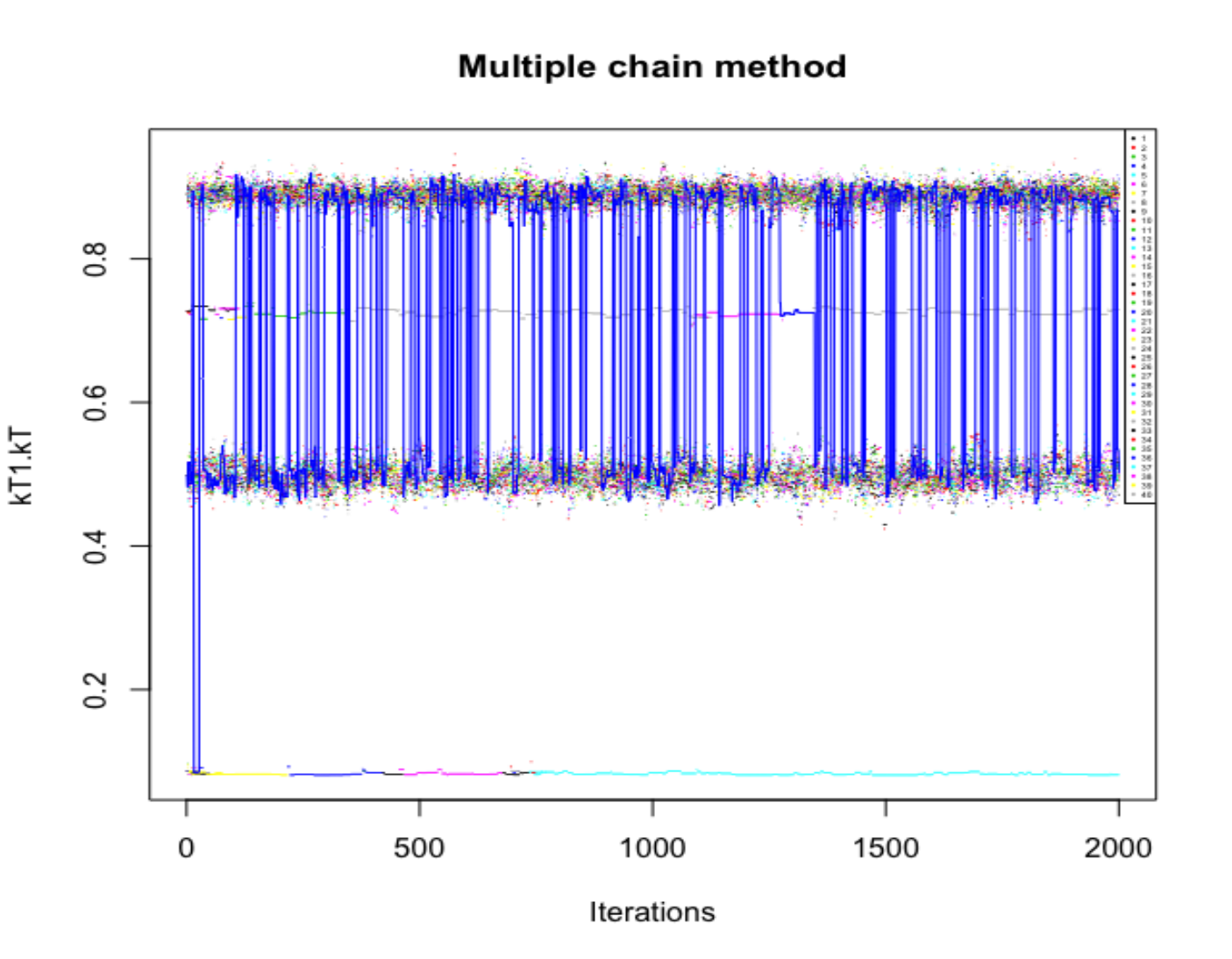


Figure 5: Dotplots of kT1.kT in multiple chain method overlaid by the traceplot of the 20th chain.

Label Switching

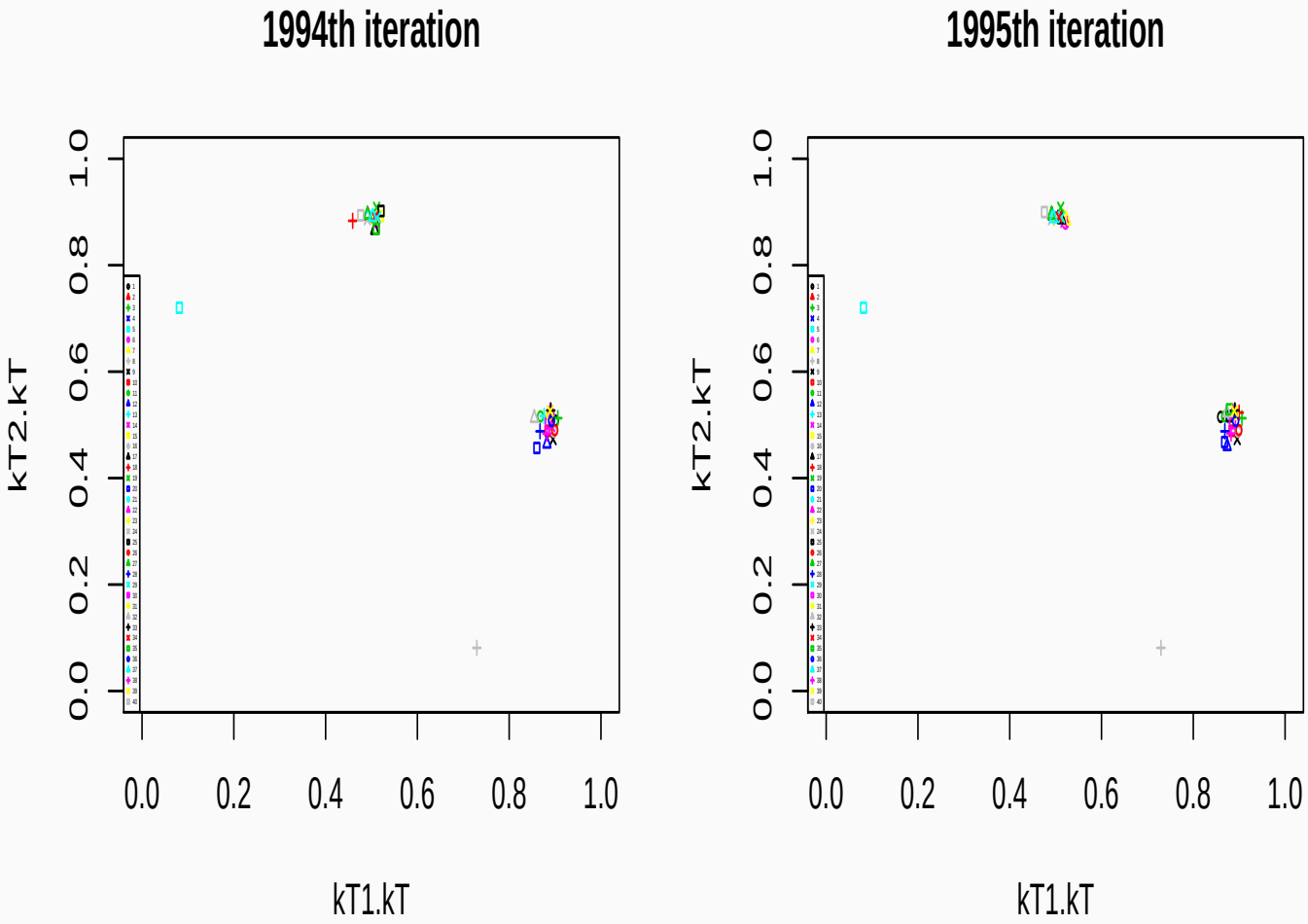


Figure 6: Scatterplots of kT2.kT vs kT1.kT at 1994th iteration (left) and 1995th iterations (right)

xsphabs.abs1*(xsvapec.kT1+xsvapec.kT2)

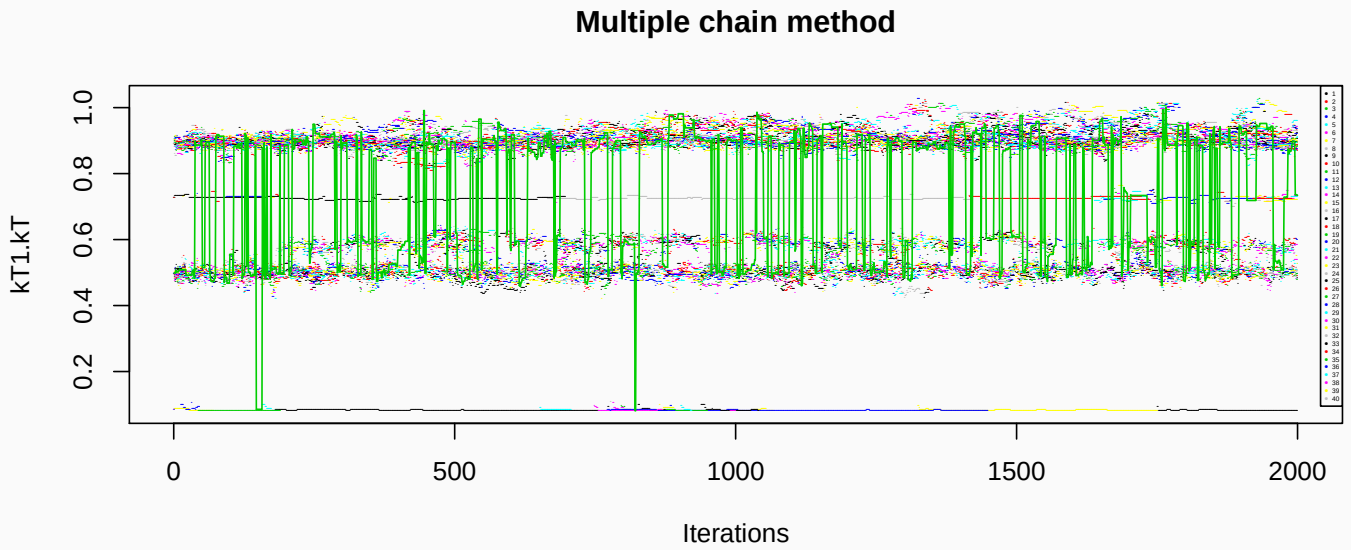
- While the multiple chain method may not be able to discover new modes if not nearby, it can certainly tell the rough proportion of each mode and which mode is more important.
- kT1.kT = 0.497 with 95% CI (0.468,0.526) and kT2.kT = 0.889 with 95% CI (0.868,0.910).
- Sherpa fit() and covariance() outputs:

Param	Best-Fit	SD
kT1.kT	0.498	0.011
kT2.kT	0.890	0.008

- Sherpa can also get to the minor mode, while the multiple chain method tells us it may not be worth looking into.

xsphabs.abs1*(xsvapec.kT1+xsvapec.kT2)

- Do not link kT.norm!



- Found more modes:
  - kT1.kT = 0.50 and kT2.kT = 0.89 where kT1.norm = 0.035 and kT2.norm = 0.034
  - kT1.kT = 0.60 and kT2.kT = 0.95 where kT1.norm = 0.045 and kT2.norm = 0.025
- Rough proportion is about 2:1