Advanced Simulation - Lecture 9

Patrick Rebeschini

February 12th, 2018

Outline

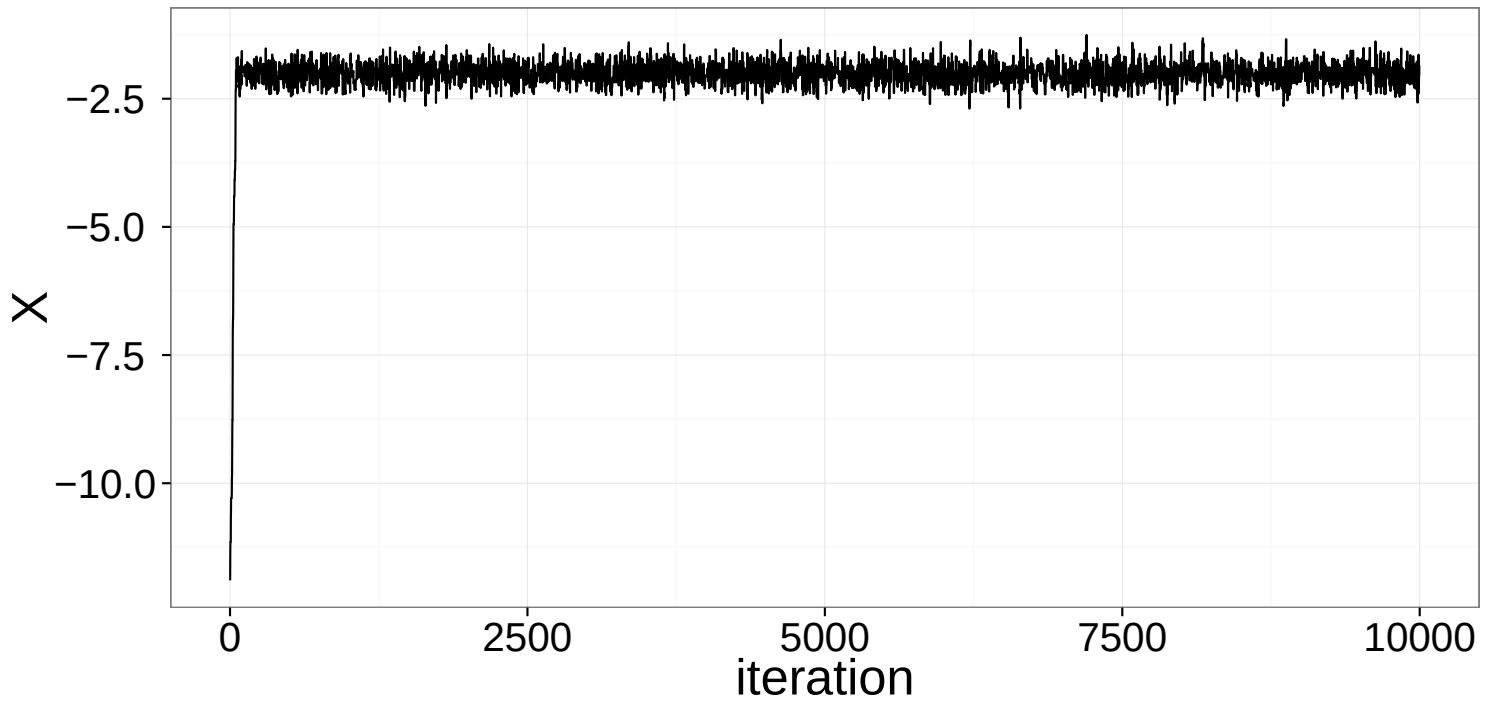
- Convergence diagnostics for MCMC.
- Parallel MCMC.
- Adaptive multiple importance sampling.

Convergence diagnostics

- Goal: assess whether MCMC chains have converged.
- In general, impossible to know for sure that there is no problem.
- But we can sometimes know for sure that there *is* a problem.

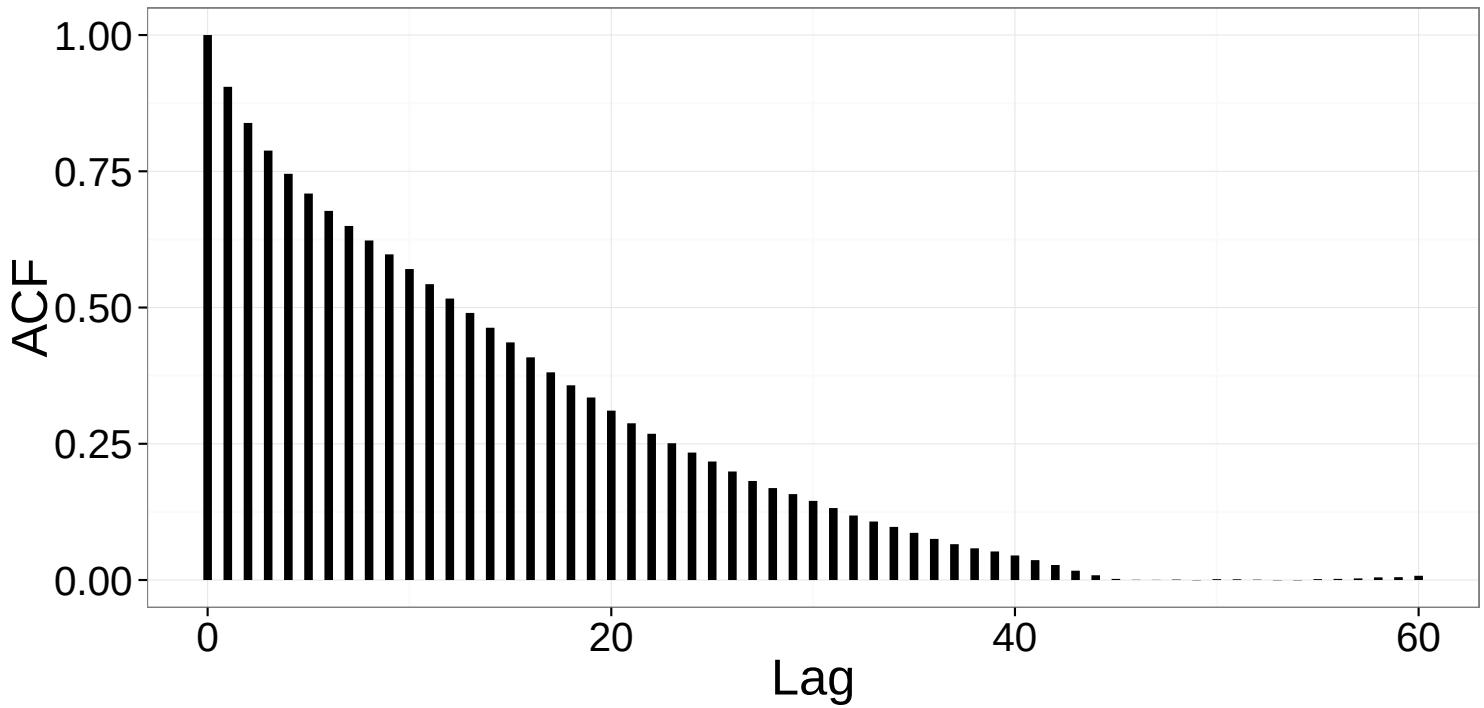
Visual diagnostics: traceplot

Target: $\pi = \mathcal{N}(-2, 0.2^2)$, proposal $q(y \mid x) = \mathcal{N}(y; x, 0.5^2)$.



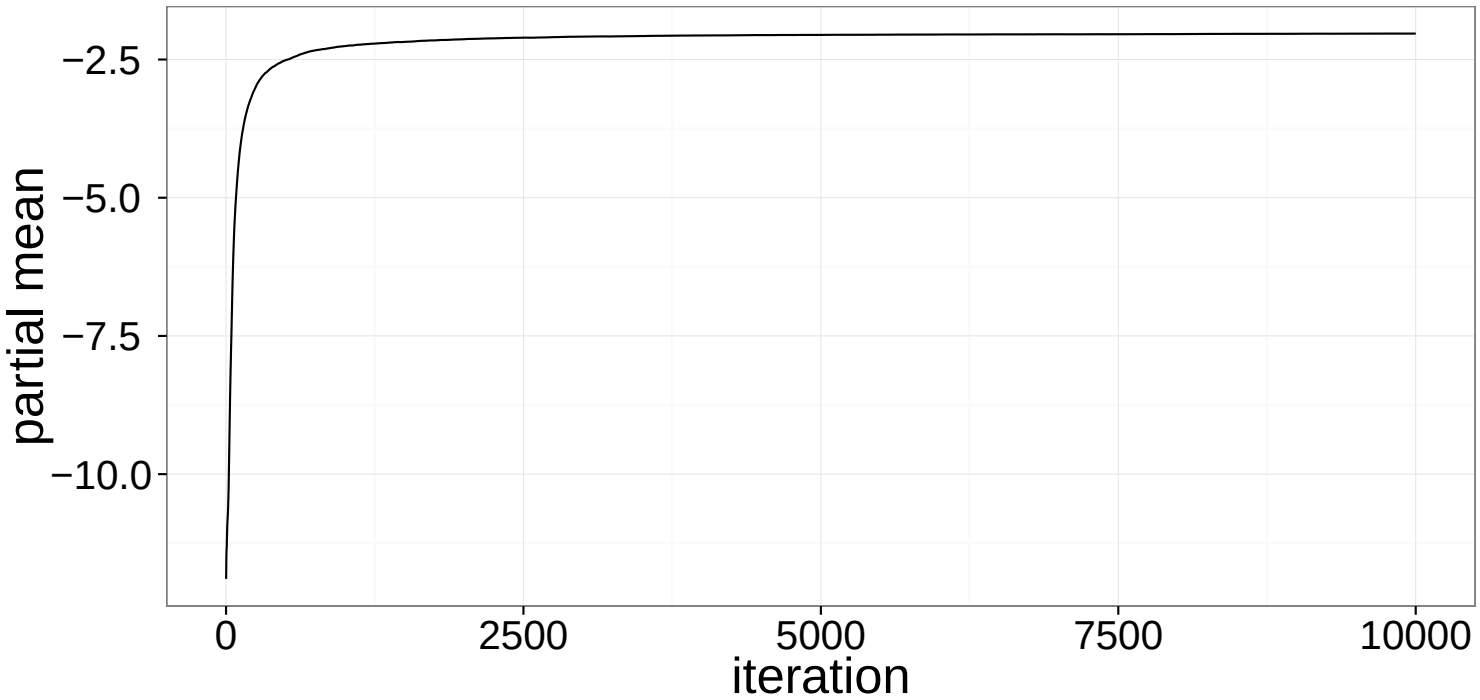
Visual diagnostics: autocorrelogram

Target: $\pi = \mathcal{N}(-2, 0.2^2)$, proposal $q(y \mid x) = \mathcal{N}(y; x, 0.5^2)$.



Visual diagnostics: convergence of estimators

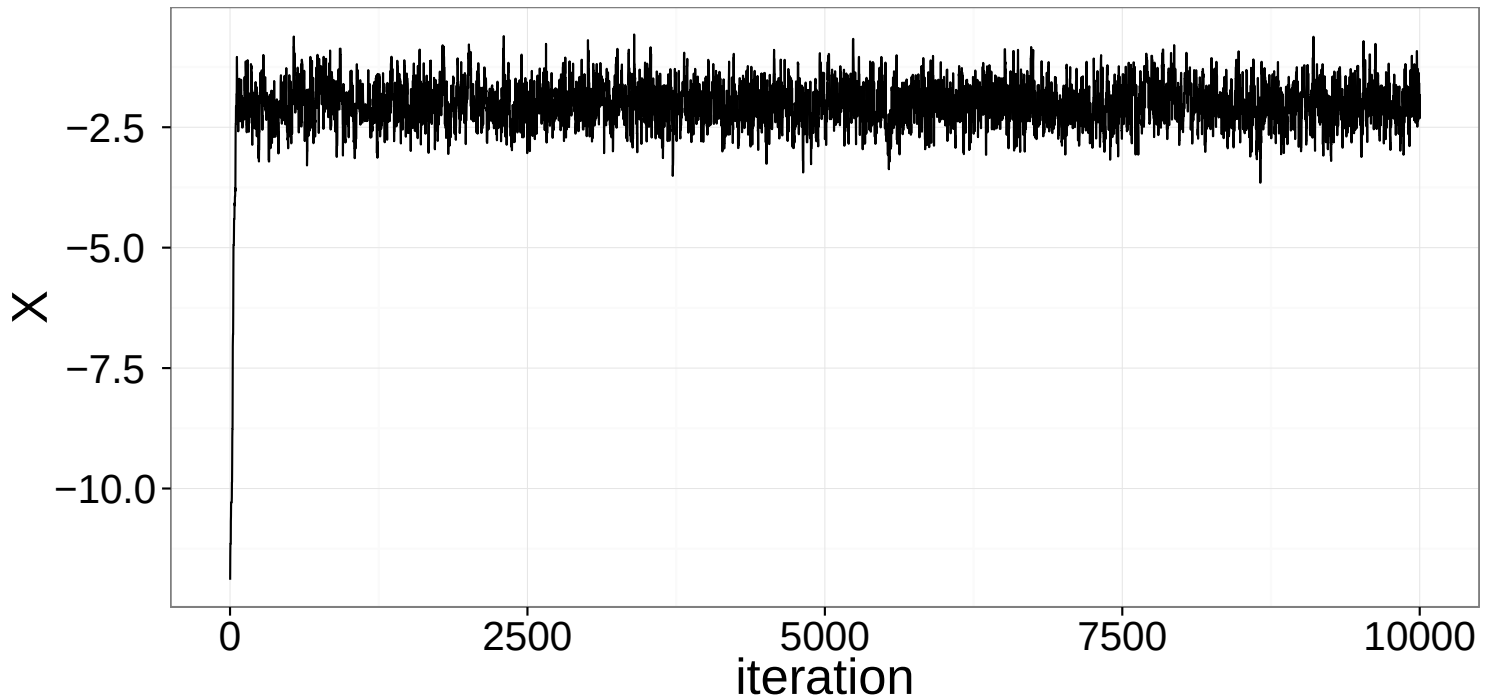
Target: $\pi = \mathcal{N}(-2, 0.2^2)$, proposal $q(y \mid x) = \mathcal{N}(y; x, 0.5^2)$.



Could be also computed on different non-overlapping subsequences, leading to Geweke's diagnostics.

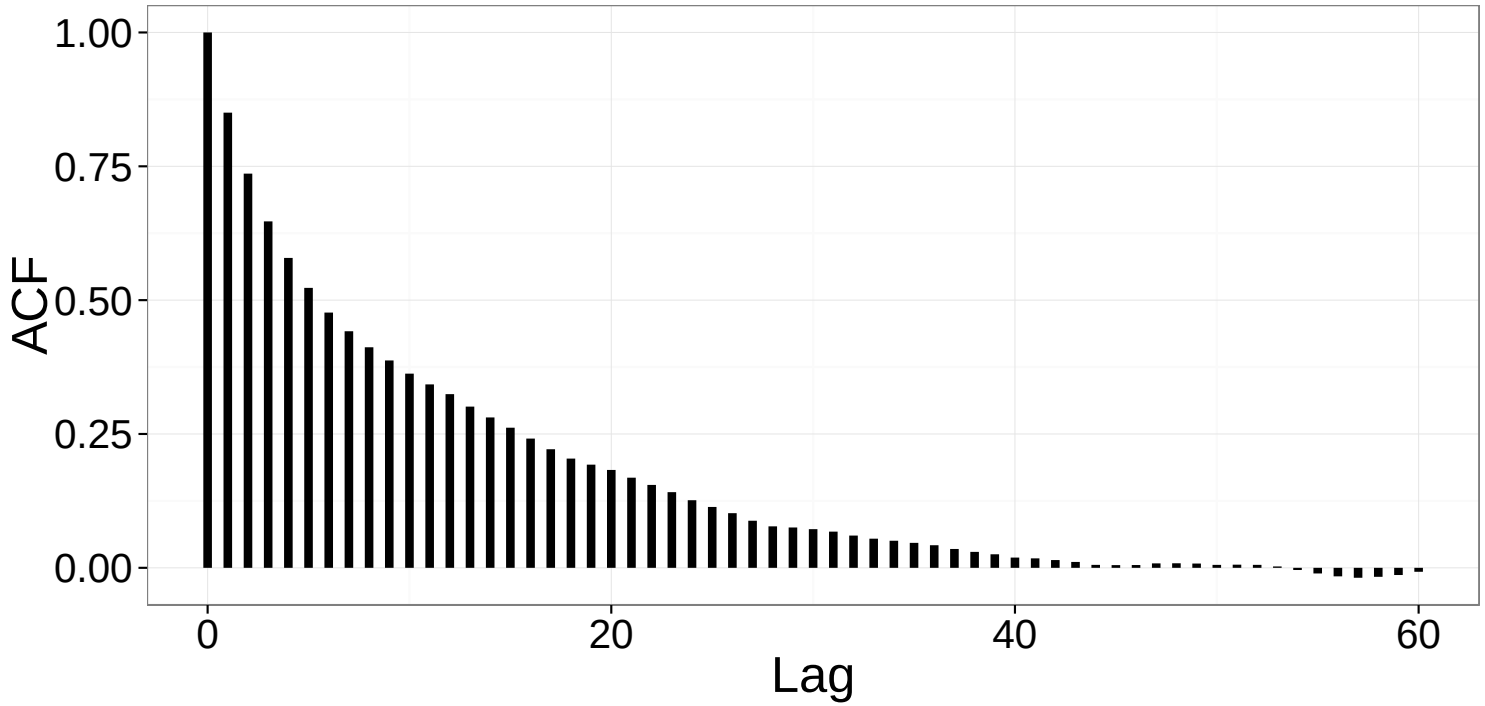
Visual diagnostics: traceplot

Target: $\pi = \frac{1}{2}\mathcal{N}(-2, 0.2^2) + \frac{1}{2}\mathcal{N}(+2, 0.2^2)$, same proposal.



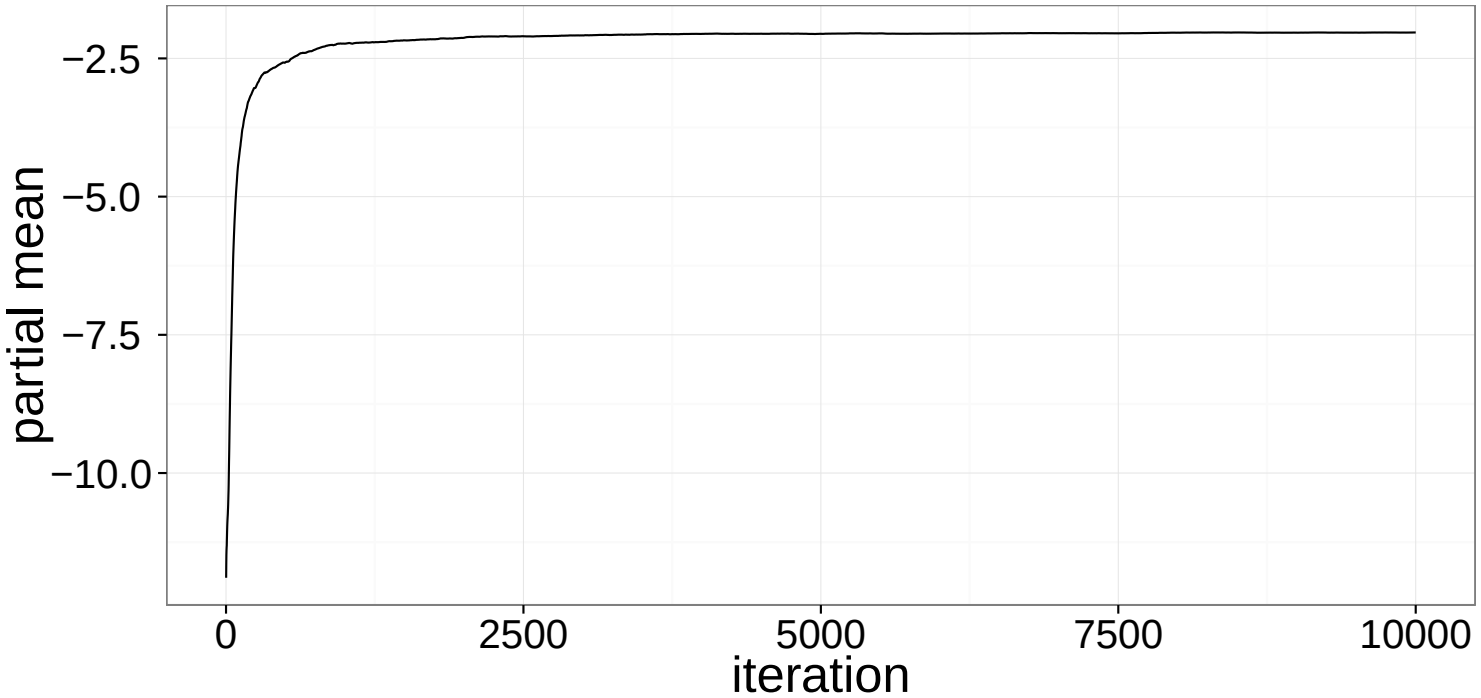
Visual diagnostics: autocorrelogram

Target: $\pi = \frac{1}{2}\mathcal{N}(-2, 0.2^2) + \frac{1}{2}\mathcal{N}(+2, 0.2^2)$, same proposal.



Visual diagnostics: convergence of estimators

Target: $\pi = \frac{1}{2}\mathcal{N}(-2, 0.2^2) + \frac{1}{2}\mathcal{N}(+2, 0.2^2)$, same proposal.



Multiple starting point

- We start M chains from various starting points.
- After enough iterations the starting point should not matter and hence we should obtain the *same* results based on each chain.
- We have the classical "sum of squares" decomposition in "intra group" and "inter group" terms:

$$\sum_{m=1}^{M}\sum_{t=1}^{T}(X_{m,t}-\bar{X}_{.,.})^2 = \sum_{m=1}^{M}\sum_{t=1}^{T}(\bar{X}_{m,.}-\bar{X}_{.,.})^2 + \sum_{m=1}^{M}\sum_{t=1}^{T}(X_{m,t}-\bar{X}_{m,.})^2$$

Multiple starting point

- This leads to considering

$$W = \frac{1}{M}\sum_{m=1}^{M}\frac{1}{T-1}\sum_{t=1}^{T}(X_{m,t} - \bar{X}_{m,\cdot})^2$$

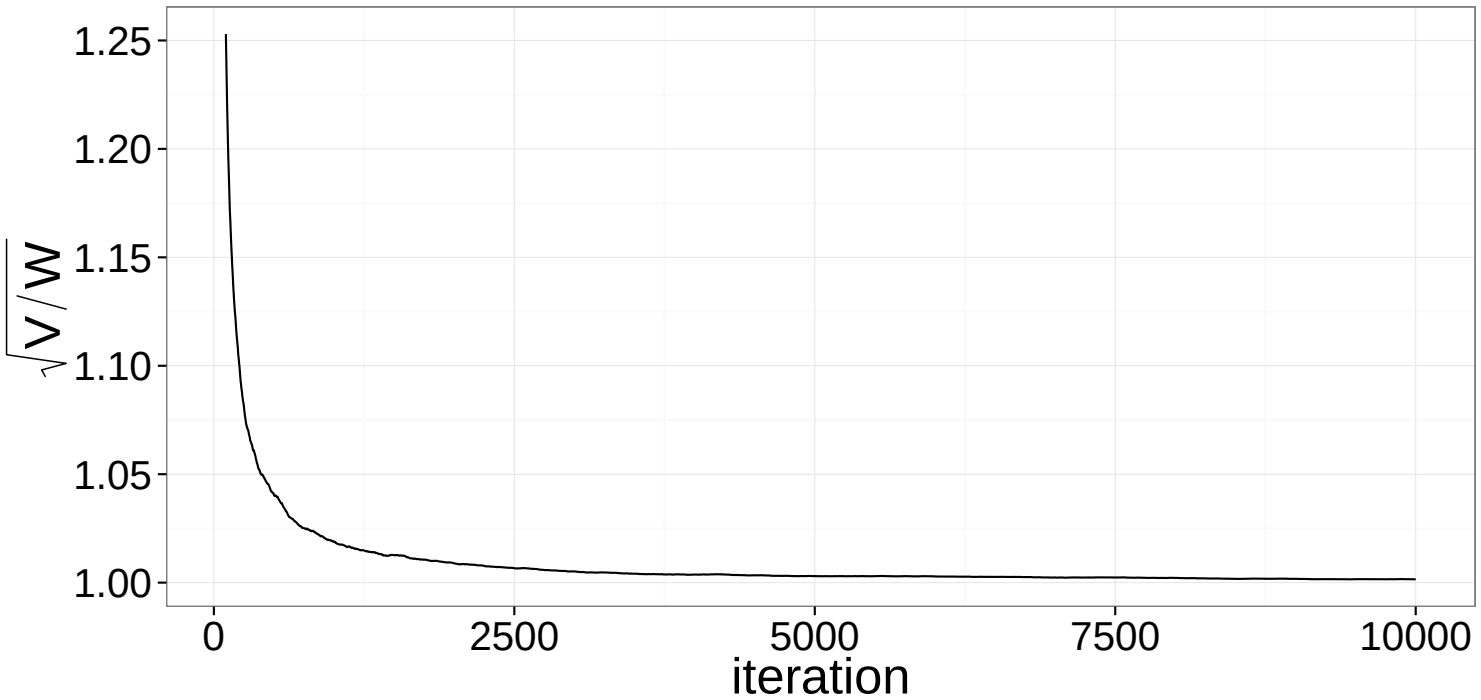
$$B = \frac{1}{M-1}\sum_{m=1}^{M}(\bar{X}_{m,\cdot} - \bar{X}_{\cdot,\cdot})^2$$

$$V = \left(1 - \frac{1}{T}\right)W + B$$

- In principle W and V should both converge to the true variance of the target distribution.
- V would be unbiased if starting points were drawn from the target, whereas W under-estimates the variance.
- We can thus plot $\sqrt{V/W}$ and compare to 1. This is the idea behind Gelman-Rubin diagnostics.

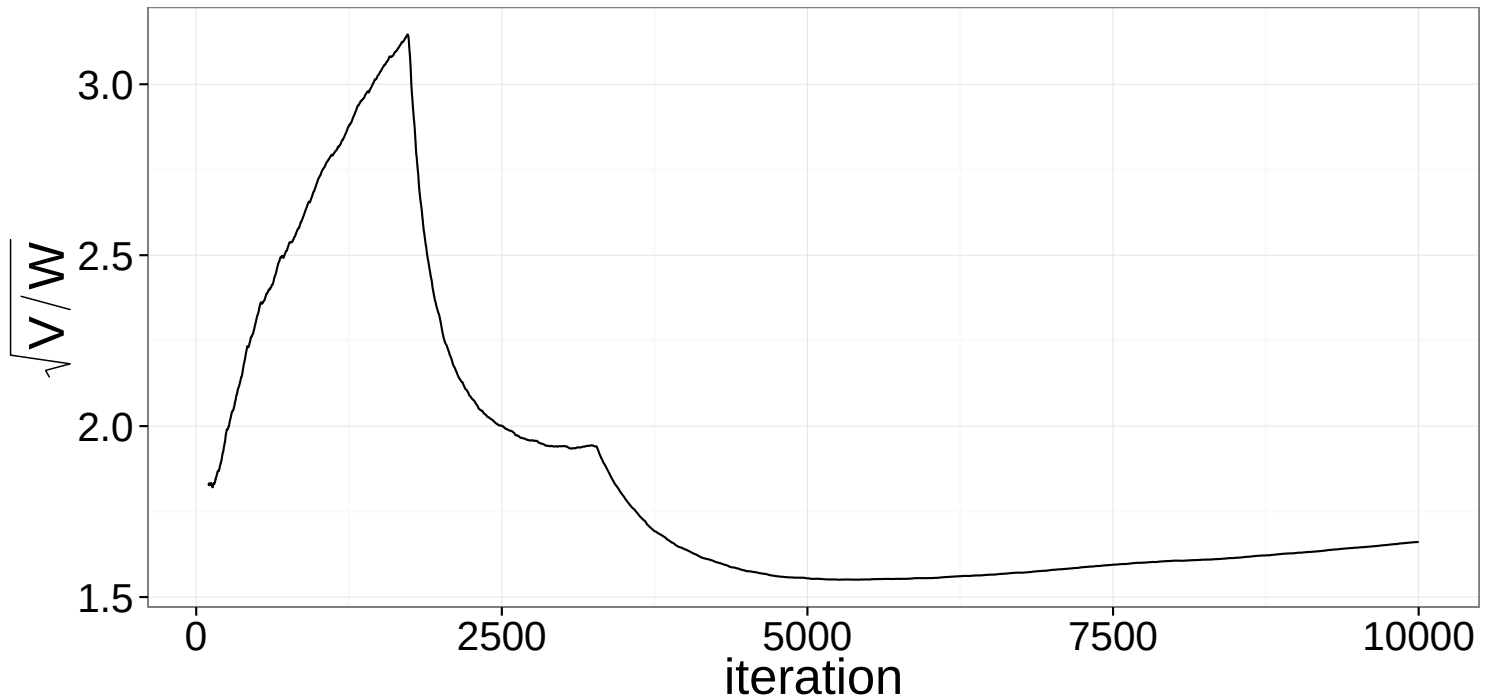
Visual diagnostics: Gelman-Rubin diagnostics

Target: $\pi = \mathcal{N}(-2, 0.2^2)$, $M = 4$ chains.



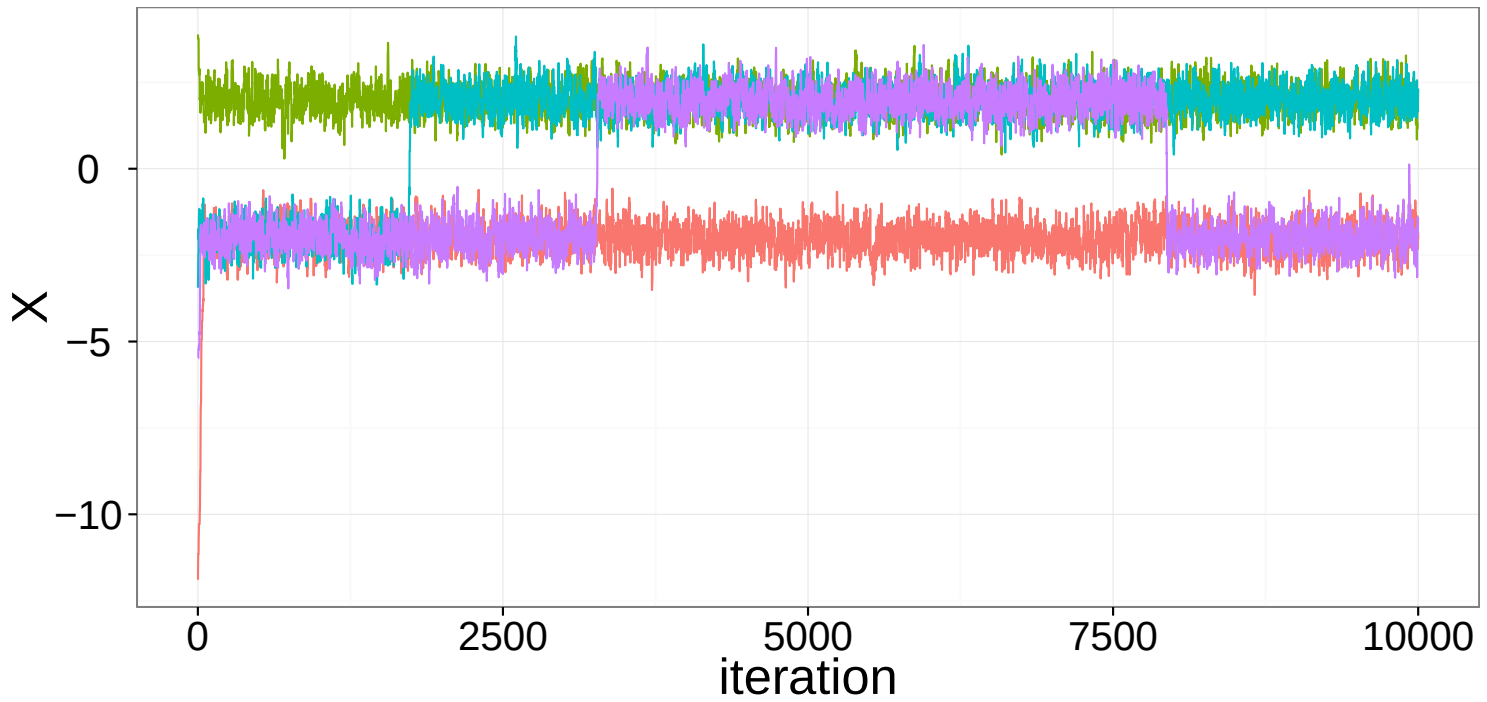
Visual diagnostics: Gelman-Rubin diagnostics

Target: $\pi = \frac{1}{2}\mathcal{N}(-2, 0.2^2) + \frac{1}{2}\mathcal{N}(+2, 0.2^2)$, $M = 4$ chains.



Visual diagnostics: traceplot with M chains

Target: $\pi = \frac{1}{2}\mathcal{N}(-2, 0.2^2) + \frac{1}{2}\mathcal{N}(+2, 0.2^2)$, $M = 4$ chains.



Parallelization

In the past (and in the next?) years, many more parallel cores, but not much more clockspeed.

- Among the methods seen so far, which are parallelizable?
- MCMC methods are by definition iterative methods. Sometimes the likelihood evaluation itself can be parallelized.
- We can run independent MCMC in parallel, as in the Gelman-Rubin diagnostics.
- Should we make the chains interact?

Parallelization of the likelihood evaluation

Consider the evaluation of the likelihood in the normal mixture case: the observations $Y_1, \ldots, Y_n$ come from

$$\forall i \in \{1, \ldots, n\} \quad Y_i \sim \sum_{k=1}^{K} p_k \mathcal{N}(\mu_k, \sigma_k^2).$$

The likelihood can be written

$$\mathcal{L}(\theta; y_1, \ldots, y_n) = \prod_{i=1}^{n} \left(\sum_{k=1}^{K} p_k \varphi(y_i; \mu_k, \sigma_k^2) \right)$$

which can be done by evaluating the n terms in the product in parallel and then taking the product.
Or $n \times K$ terms in parallel, and then partial sums and a product.

Parallelization of the likelihood evaluation

- For i.i.d. data the likelihood evaluation can be parallelized.
- In cases where
  - the likelihood is not so expensive,
  - or the likelihood evaluation cannot be efficiently parallelized.

  then a single-chain Metropolis-Hastings algorithm cannot benefit from multiple processors.
- However we can run multiple chains!

Parallel MCMC on subtargets

Suppose the target π is the posterior distribution of the parameter in a model where the observations are assumed i.i.d.

$$\pi(\theta) = p(\theta \mid y_1, \ldots, y_n) \propto \prod_{i=1}^{n} f(y_i; \theta) p(\theta).$$

We can decompose the posterior distribution as follows

$$\pi(\theta) = \prod_{g=1}^{G} \left(\prod_{i_g=1}^{n_g} f(y_{\sigma_g(i_g)}; \theta) p^{\gamma_g}(\theta) \right).$$

Here $\sum_{g=1}^{G} \gamma_g = 1$ and σ_g is such that each $i \in \{1, \ldots, n\}$ corresponds to a single $\sigma_g(i_g)$ with $i_g \in \{1, \ldots, n_g\}$.

Parallel MCMC on subtargets

Example: $n = 1,000,000$ data points, $G = 1000$ groups.

$$\pi(\theta) = \prod_{g=1}^{1000} \left(\prod_{i=1}^{1000} f(y_{(g-1)\times 1000+i}; \theta) p^{1/1000}(\theta) \right).$$

We can run a MCMC for each group g, targeting

$$\pi_g(\theta) = \prod_{i=1}^{1000} f(y_{(g-1)\times 1000+i}; \theta) p^{1/1000}(\theta).$$

Then we need to combine the G different estimates to recover the posterior of interest, $\pi(\theta)$.

Parallel MCMC on subtargets

One naive way to do is to use kernel density estimates of each subtarget π_g:

$$\widehat{\pi}_g(\theta) = \frac{1}{T}\sum_{t=1}^{T} \frac{1}{h^d} K\left(\frac{|\theta_t - \theta|}{h}\right),$$

for some kernel K and bandwidth h. Then we can approximate the target density $\pi(\theta)$ by $\widehat{\pi}(\theta)$, the product of $\widehat{\pi}_g(\theta)$ over $g = 1, \ldots, G$:

$$\widehat{\pi}(\theta) = \prod_{g=1}^{G} \widehat{\pi}_g(\theta)$$

Unfortunately product of independent estimators... the variance of $\widehat{\pi}(\theta)$ grows exponentially fast with the number of groups G.

Parallel Tempering

- The idea of parallel tempering is to run N chains targeting different versions of π, of "increasing difficulty".
- Introduce "inverse temperatures":

$$0 < \gamma_1 < \gamma_2 < \ldots < \gamma_N = 1.$$

- Introduce "tempered" distributions π^{γ_n} for $n = 1, \ldots, N$.
- For $\gamma \approx 0$, π^{γ} is considered easier to sample because the variations of π are smaller.

Parallel Tempering

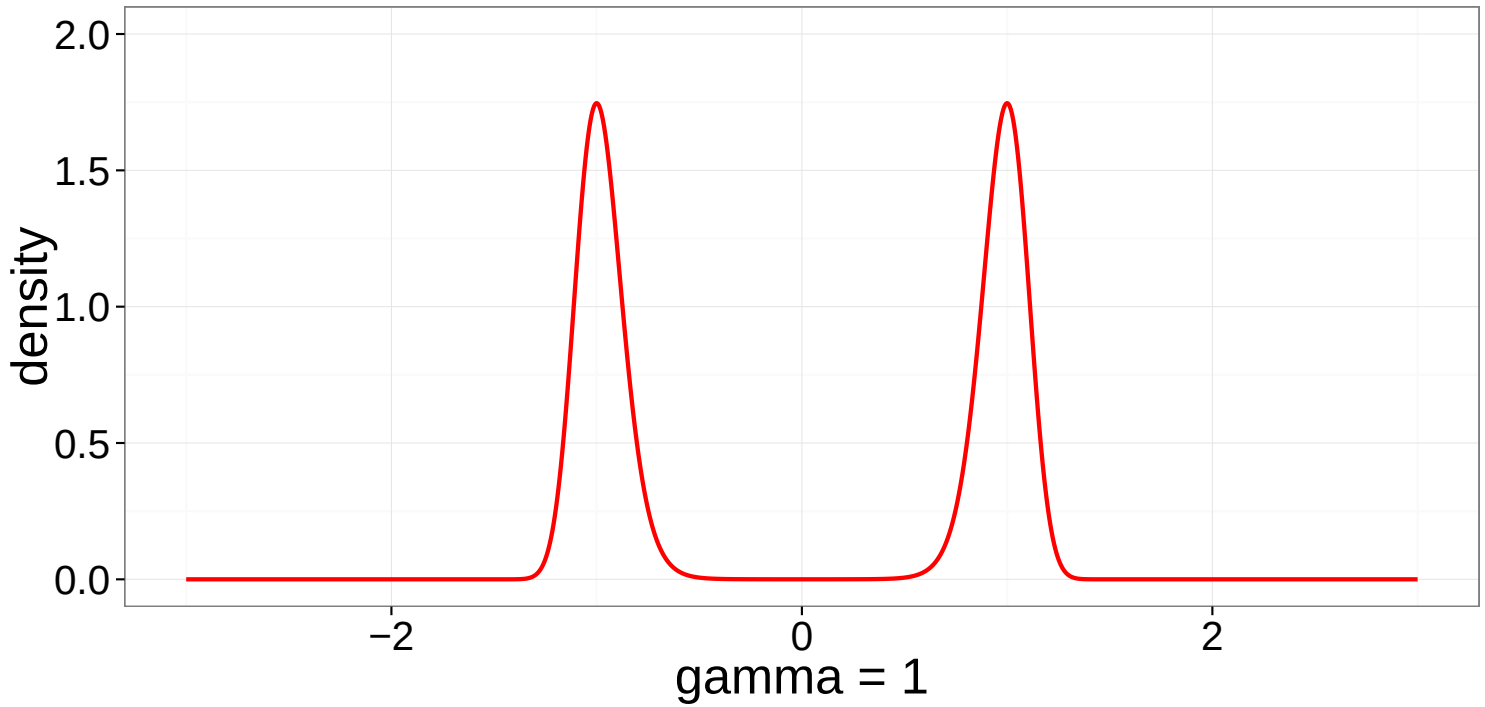


Figure: Target density function.

Parallel Tempering

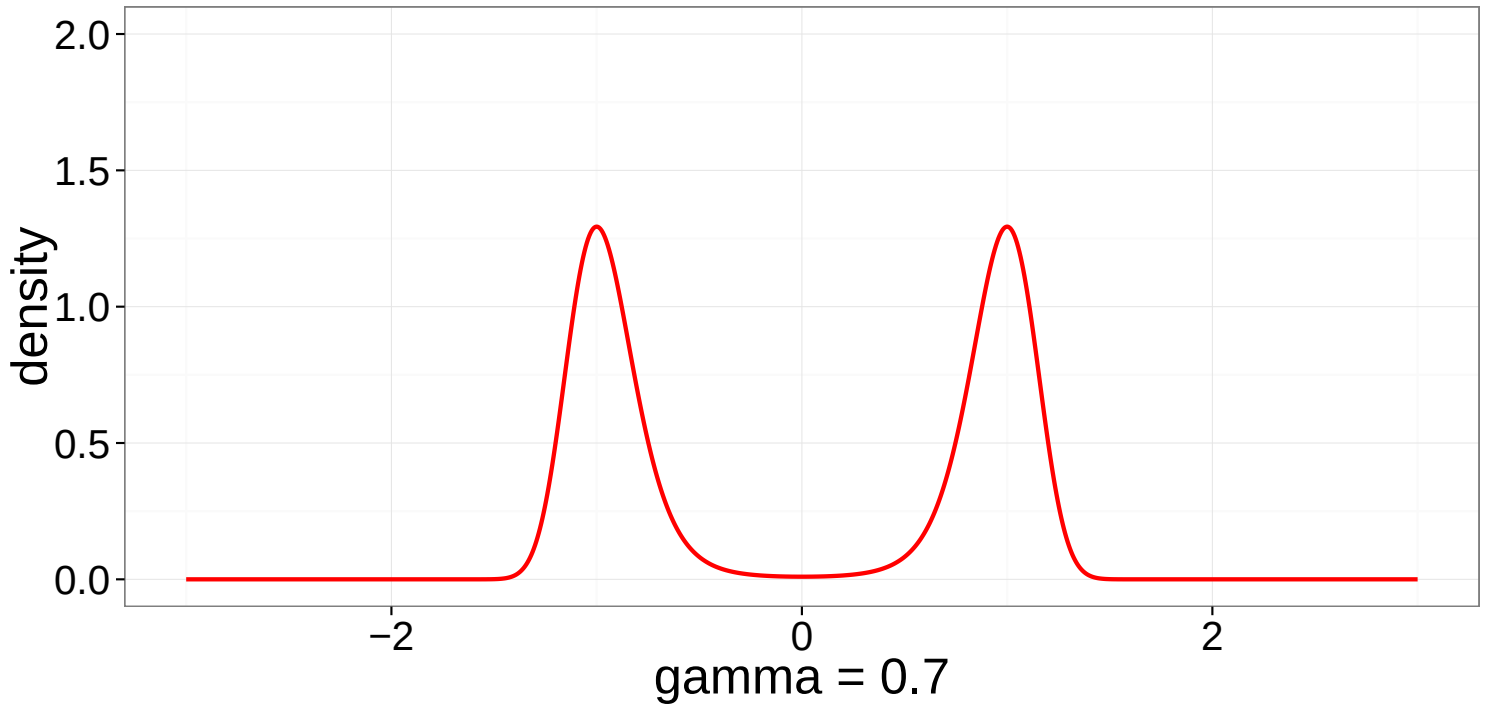


Figure: Target density function.

Parallel Tempering

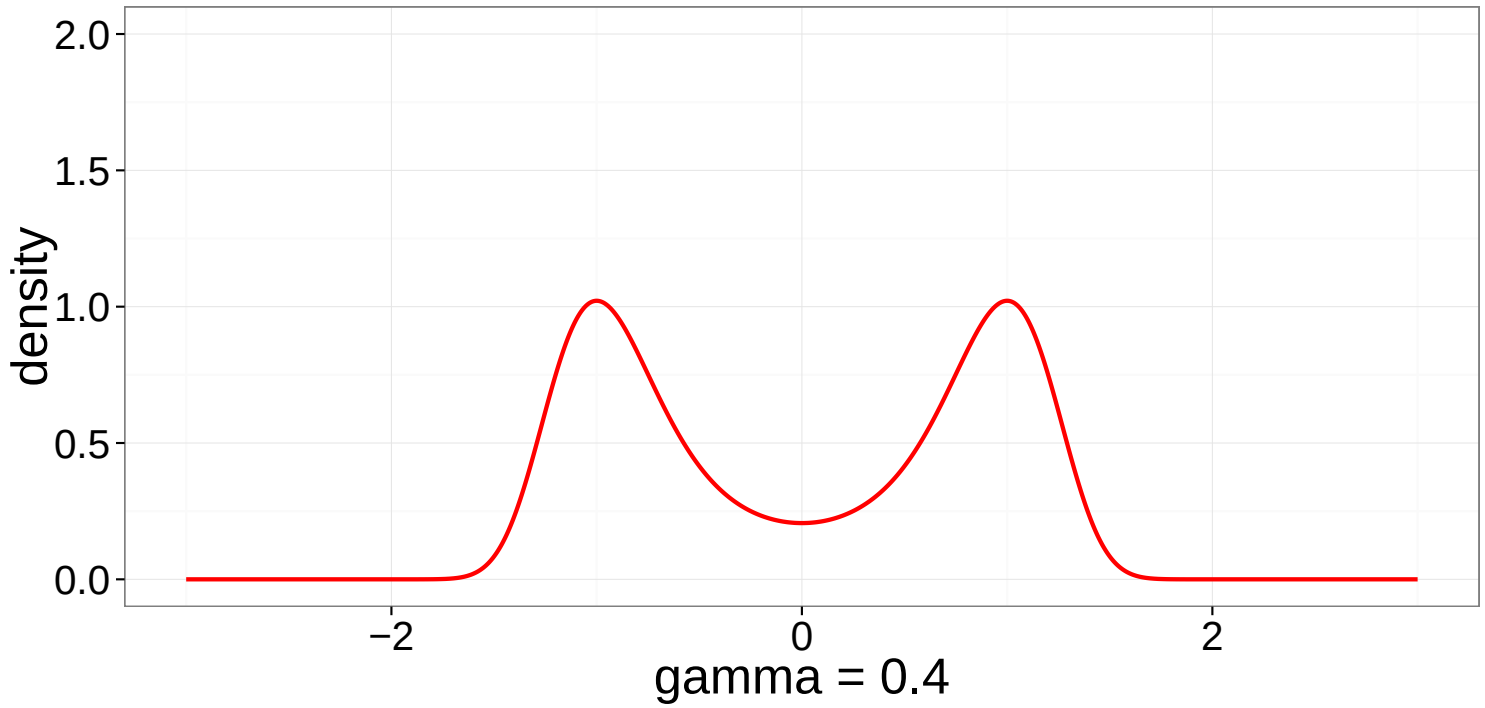


Figure: Target density function.

Parallel Tempering

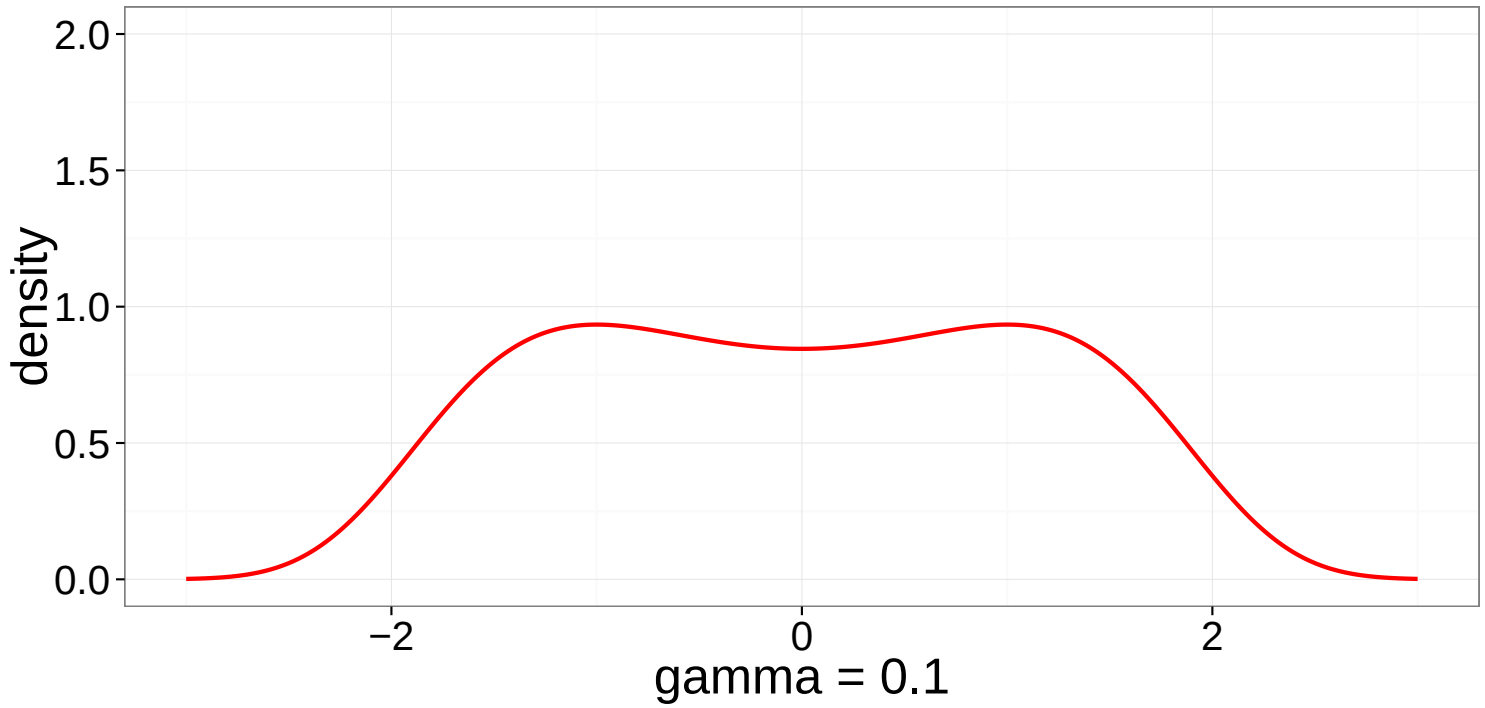


Figure: Target density function.

Parallel Tempering

Let's use $N = 10$ chains and $\gamma_1 = 0.1, \gamma_2 = 0.2, \ldots, \gamma_{10} = 1$.

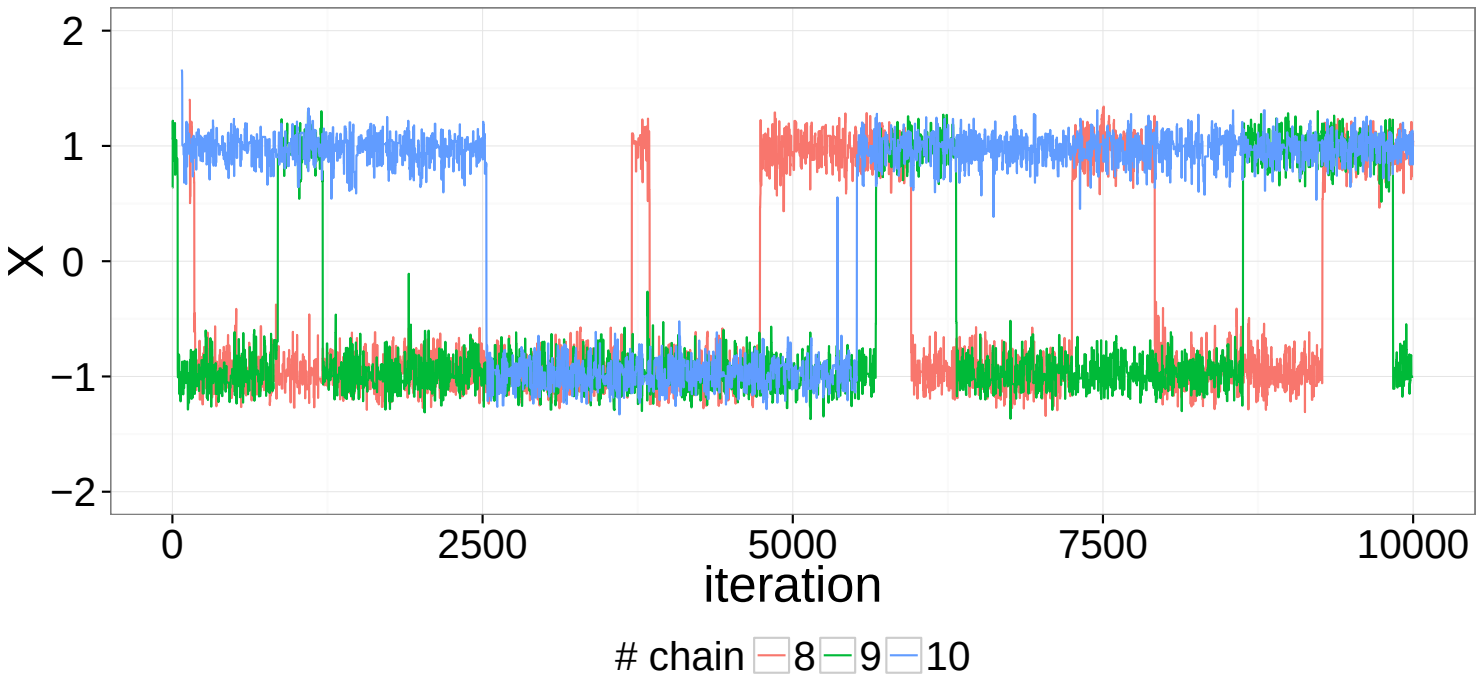


Figure: Trace plot of the "low temperature chains".

Parallel Tempering

Let's use $N = 10$ chains and $\gamma_1 = 0.1, \gamma_2 = 0.2, \ldots, \gamma_{10} = 1$.

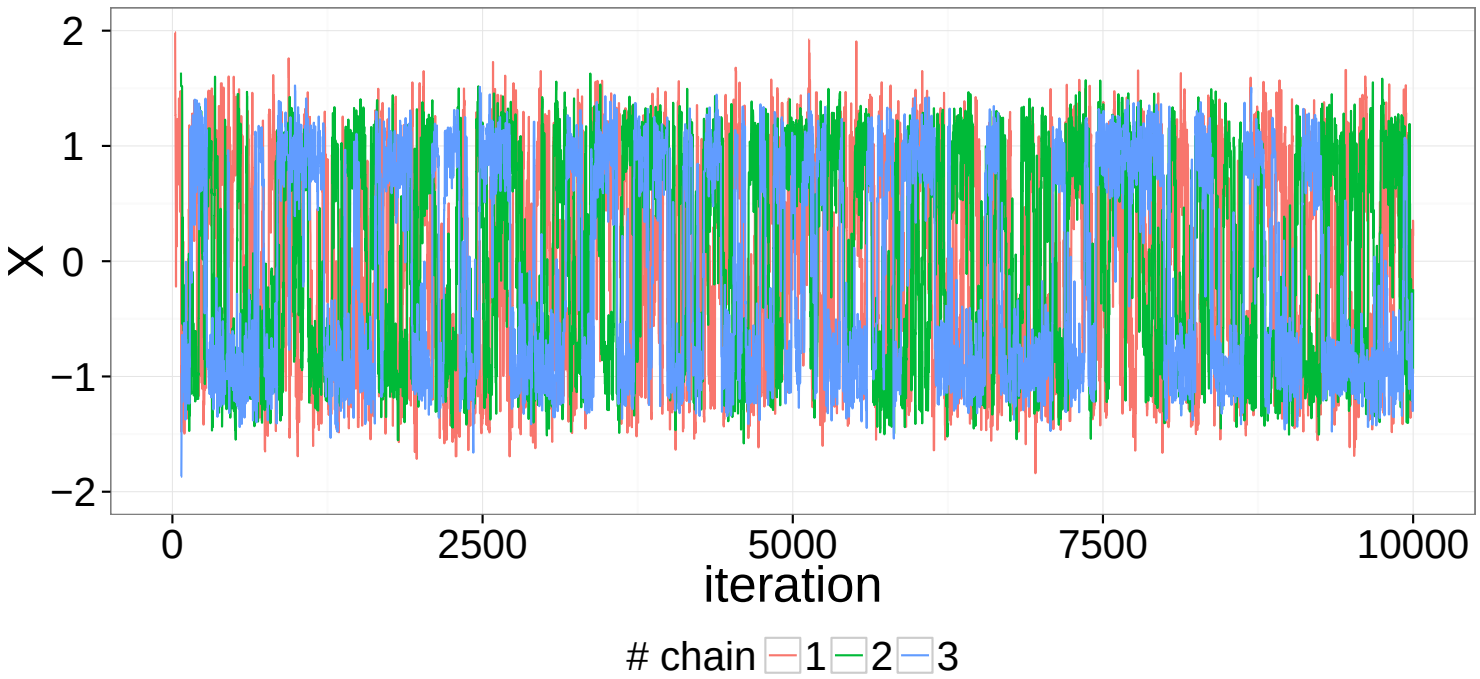


Figure: Trace plot of the "high temperature chains".

Parallel Tempering

- If we want to find the modes of π, we might just use the high temperature chains and forget about sampling directly from π.

- If we want to sample from π, can we use the "high temperature" chains to improve the mixing of the chain targeting π?

- Parallel tempering works by proposing moves where chains of different temperatures are swapped.

Parallel Tempering

When a "swap move" is to be performed, do the following.

- Sample indices k_1, k_2 uniformly in $\{1, \ldots, N\}$.
- With acceptance probability

$$\min\left(1, \frac{\pi^{\gamma_{k_1}}(x_{k_2})\pi^{\gamma_{k_2}}(x_{k_1})}{\pi^{\gamma_{k_1}}(x_{k_1})\pi^{\gamma_{k_2}}(x_{k_2})}\right).$$

  exchange the value of x_{k_1} and x_{k_2}.
- This doesn't change the joint target distribution $\pi^{\gamma_1} \otimes \pi^{\gamma_2} \otimes \ldots \otimes \pi^{\gamma_N}$.
- In particular the N-th chain still targets $\pi^{\gamma_N} = \pi$.

Parallel Tempering

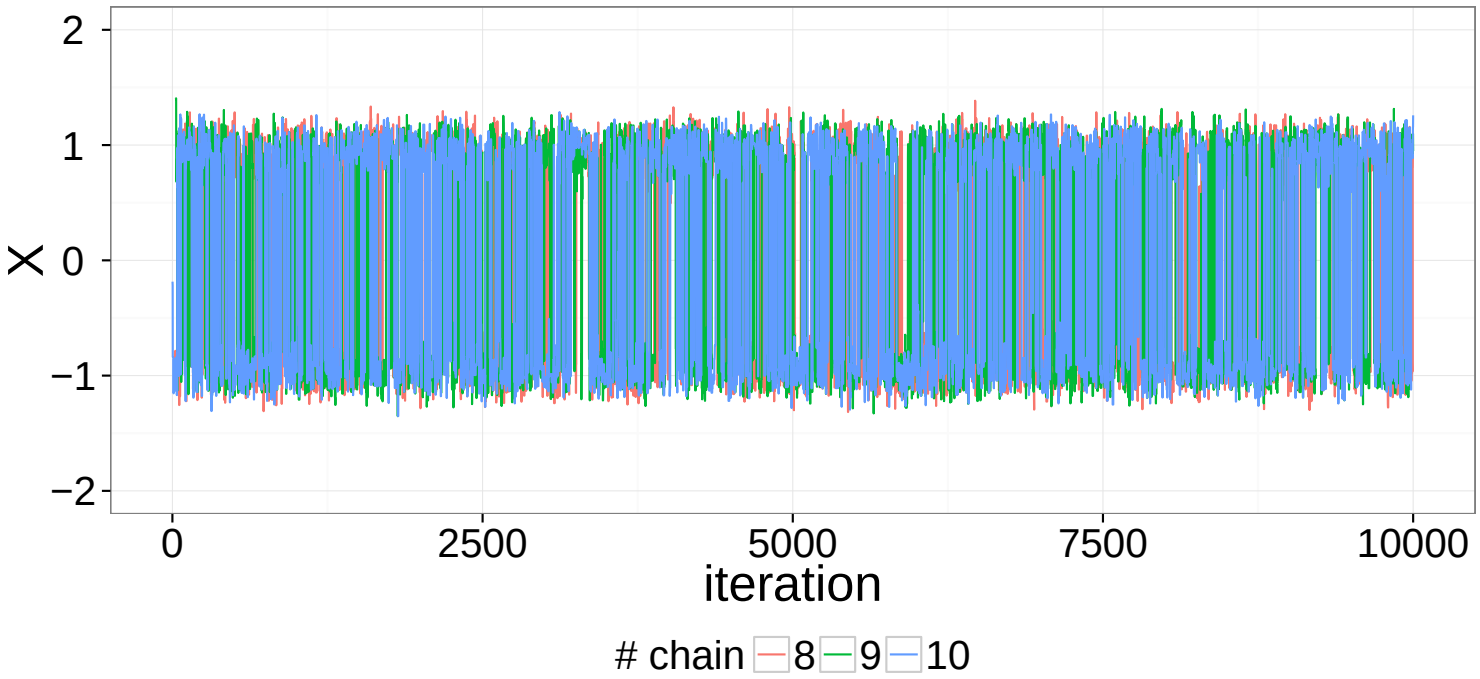


Figure: Trace plot of the "low temperature chains" using swap moves.

Parallel Tempering

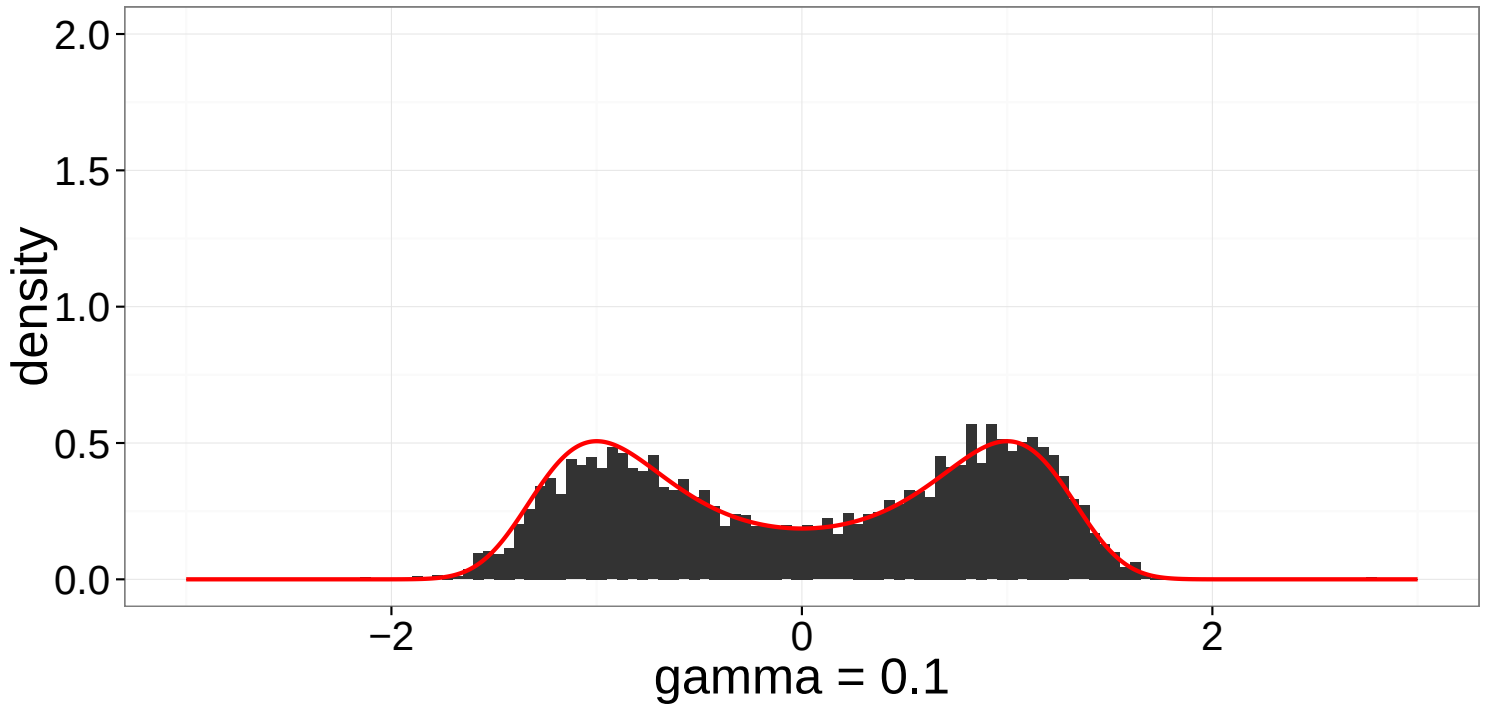


Figure: Histogram of the chain targeting π^{γ_1}.

Parallel Tempering

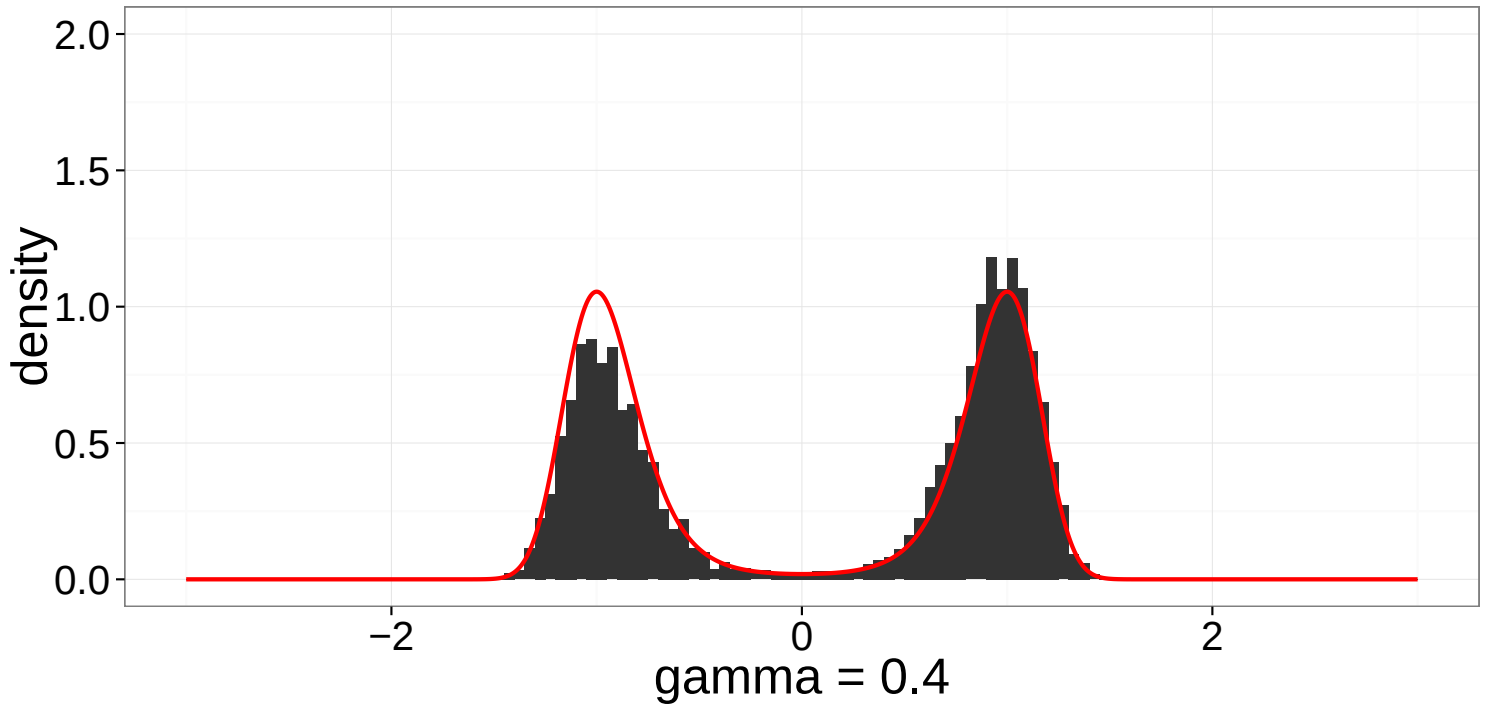


Figure: Histogram of the chain targeting π^{γ_4}.

Parallel Tempering

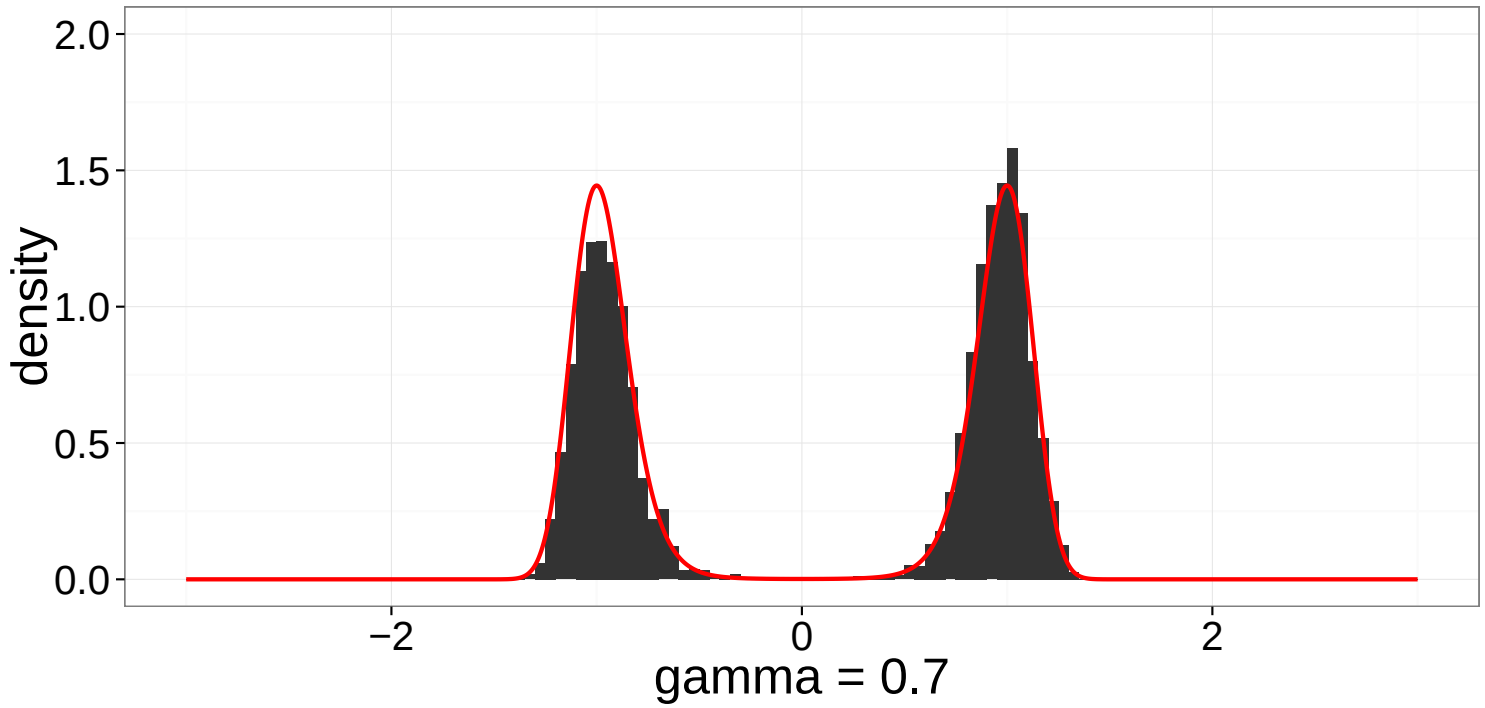


Figure: Histogram of the chain targeting π^{γ_7}.

Parallel Tempering

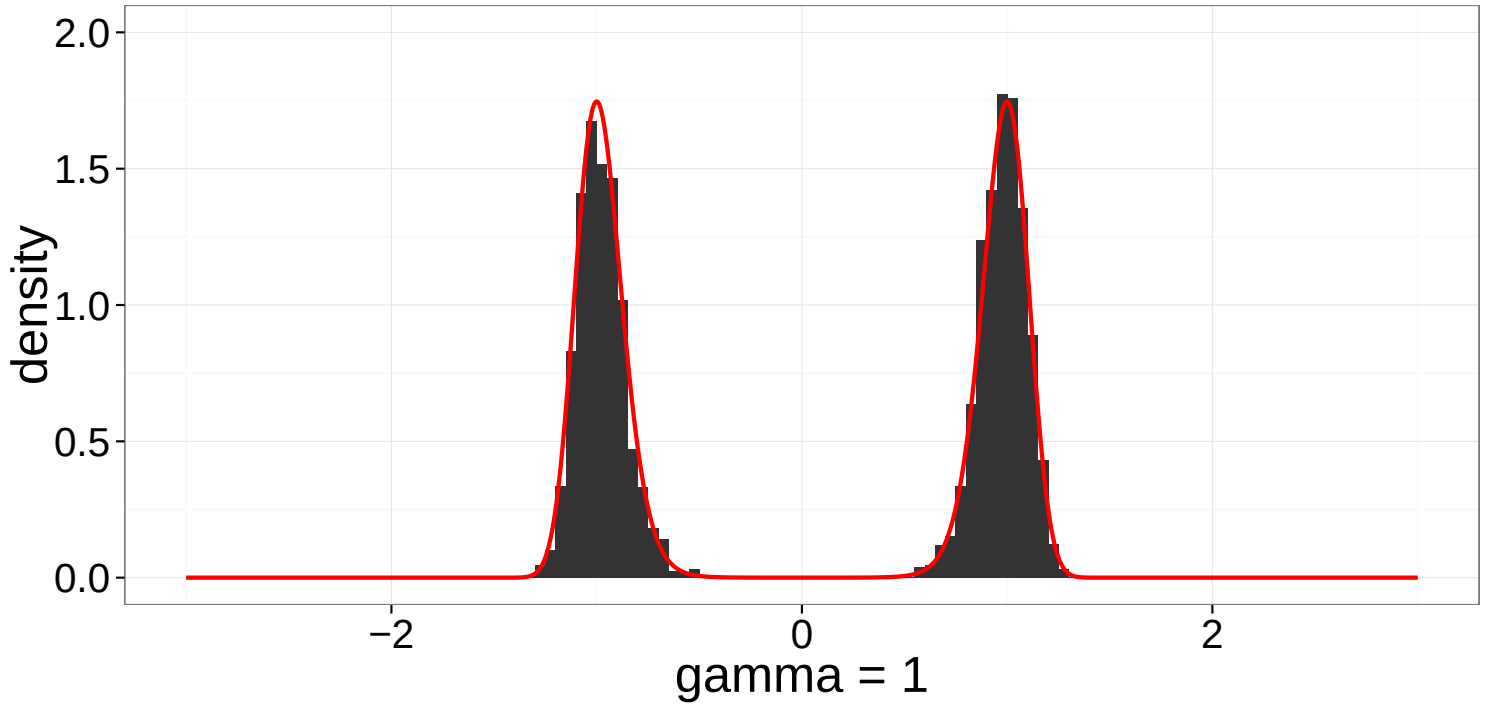


Figure: Histogram of the chain targeting $\pi^{\gamma_{10}}$.

Swap moves improve the mixing of chains with high values of γ.

Adaptive Multiple Importance Sampling

- Idea: importance sampling is very easy to parallelize.
- However it requires a good proposal distribution q, especially in high dimension.
- Suppose we start by drawing $X_1, \dots, X_n \sim q_1$ and evaluating

$$\forall k \in \{1, \dots, N\} \quad w_k \propto \frac{\pi(X_k)}{q_1(X_k)}.$$

- It provides an approximation of π in the sense

$$\int \varphi(x)\pi(x)dx \approx \frac{1}{\sum_{k=1}^{N} w_k} \sum_{k=1}^{N} w_k \varphi(X_k).$$

Adaptive Multiple Importance Sampling

- Given that first try, we can now design a better proposal q_2.
- For instance, fit a Cauchy mixture on $(w_k, X_k)_{k=1}^N$. We call the estimated mixture q_2 and sample $Y_1, \ldots, Y_N \sim q_2$.
- We then evaluate

$$\forall k \in \{1, \ldots, N\} \quad \omega_k \propto \frac{\pi(Y_k)}{q_2(Y_k)}$$

- It provides an approximation of π in the sense

$$\int \varphi(x)\pi(x)dx \approx \frac{1}{\sum_{k=1}^N \omega_k} \sum_{k=1}^N \omega_k \varphi(Y_k)$$

- We can even use all the generated sample:

$$(w_k, X_k)_{k=1}^N \text{ and } (\omega_k, Y_k)_{k=1}^N.$$

Adaptive Multiple Importance Sampling

In general AMIS works by iterating at time t:

- Importance sampling from q_t to π, generating a weighted sample $(w_k^{(t)}, X_k^{(t)})_{k=1}^N$.
- Using the generated sample to design a better proposal distribution q_{t+1} for the next step.

At any time we can stop the algorithm and use the generated sample to approximate integrals with respect to π.

Sequential Importance Sampling

- Arguably the most advanced methods use a mix of tempering, Markov chain Monte Carlo and Importance Sampling elements.

- For instance Sequential Monte Carlo Samplers sample iteratively from

$$\pi_0 = \text{prior}, \pi_1 = p(\theta \mid y_1), \ldots, \pi_N = p(\theta \mid y_1, y_2, \ldots, y_N).$$

- To do so, alternate between importance sampling a whole population from π_{n-1} to π_n,

- and MCMC moves leaving π_n invariant at step n.