Math 598

Markov chain Monte Carlo

Dr David A. Stephens

Department of Mathematics & Statistics
Room 1225, Burnside Hall

Multiple chain MCMC

The principal problems Metropolis-Hastings algorithm encounters in applications relate to

- **Choice of starting values:** Poorly chosen starting values can result in slow convergence;
- **Posterior correlation:** High posterior correlation leads to inefficiency of Gibbs sampler moves, and slow convergence; this can sometimes be overcome by reparameterization.
- **Failure to explore the state space:** Algorithm can get trapped in localized regions of high probability density; this can sometimes be overcome by using *multiple chain* MCMC.

Multi-Mode Distributions

Example (Mixture of Bivariate Normals)

Suppose

$$\pi(\mathbf{x}) \equiv \omega \mathcal{N}_2(\boldsymbol{\mu}_1, \Sigma_1) + (1-\omega)\mathcal{N}_2(\boldsymbol{\mu}_2, \Sigma_2)$$

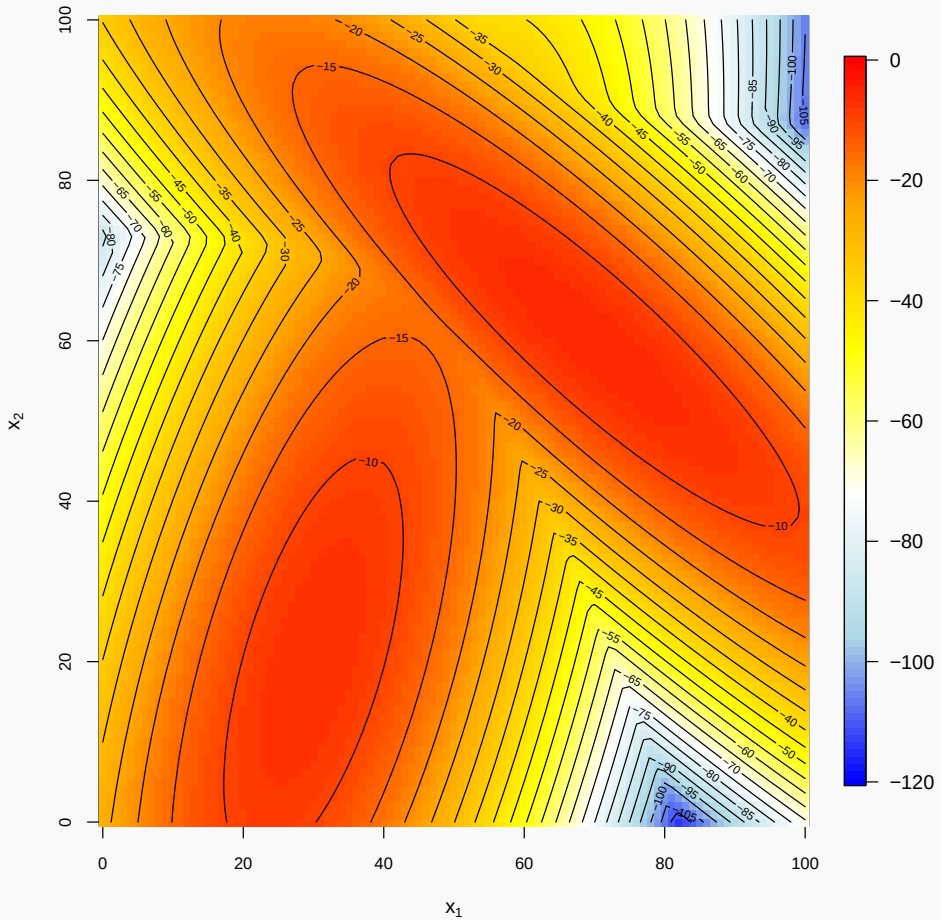
where $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_2$ are relatively well separated.

MCMC algorithms to sample from this bivariate density often encounter problems because they get stuck in the modes at $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_2$, and cannot jump between them.

Multi-Mode Distributions

Example (Mixture of Bivariate Normals)

Mixture of Bivariate Normal densities.



Dominant-Mode Distribution

Example (The Witch's Hat)

In d dimensions, suppose

$$\pi(\mathbf{x}) = \omega \left(\frac{1}{\sigma}\right)^d \prod_{j=1}^{d} \phi\left(\frac{x_j - \theta_j}{\sigma}\right) + (1-\omega) \prod_{j=1}^{d} \mathbb{1}_{(0,1)}(x_j)$$

This density has a single mode at $\boldsymbol{\theta}$, but also a diffuse probability on the d-dimensional hypercube given by the second term.

MCMC algorithms can find it difficult to explore the "diffuse" region, because they get stuck in the mode, or *vice versa*.

Multiple chain MCMC

It is possible to use multiple Markov chain algorithms to produce variates from a target distribution π.

- **Replicate chains:** we run m chains with the **same** transition kernel P from **different starting values**, and then combine the chain as independent samples.
- **Parallel chains:** we run M chains each with its **own** transition kernel $P_1, \ldots, P_M$, such that the chain is reversible with respect to π: for $m = 1, \ldots, M$

$$\pi(x)P_m(x, y) = \pi(y)P_m(y, x)$$

and then the variates from the different chains are combined as independent samples.

Multiple chain MCMC

- **Different chains:** we run M chains each with its **own** transition kernel $P_1, \ldots, P_M$, such that each chain is reversible with respect to **its own target** π_m where, say $\pi_1 \equiv \pi$. We then allow **exchange of information** between chains to facilitate sampling from π.

In this case it is not so straightforward to guarantee that the exchange of information does produce samples from π. This does need verification, once the exchange mechanism has been defined.

Such methods are called *multiple chain* or *population* MCMC.

Multiple chain MCMC

The simplest way to think about multiple chain methods in the case of different chains is via *auxiliary variables*, and an augmentation of the state space. For $m = 1, \ldots, M$, consider

- variable X_m
- (posterior) distribution $\pi_m(x)$
- irreducible transition kernel P_m, defined so that P_m and π_m exhibit detailed balance, and P_m defines a recurrent Markov chain

where, without loss of generality, π_1 is the distribution of interest, so that $X_2, \ldots, X_m$ are auxiliary variables.

Multiple chain MCMC

For $\mathbf{X} = (X_1, \ldots, X_M)$, define

$$\pi^M(\mathbf{x}) \propto \prod_{m=1}^{M} \pi_m(x_m).$$

By construction, the marginal distributions are $\pi_1, \ldots, \pi_M$.

Consider constructing a Markov chain on the extended state space, with π^M as the target distribution. The usual rules of Metropolis-Hastings apply - we propose a candidate new value, and accept it with some acceptance probability, otherwise we remain at the current value.

The values of x_1 that we collect across iterations are (dependent) samples from the marginal distribution, π_1.

▸ Move Types

Parallel Tempering

Parallel Tempering is a multiple chain method that gives a specific construction of the distributions $\pi_1, \ldots, \pi_M$. Write density π as

$$\pi(x) \propto \exp\{-H(x)\}$$

and define

$$\pi_m(x) \propto \exp\left\{-\frac{H(x)}{T_m}\right\} = \{\pi(x)\}^{1/T_m}$$

where

$$1 = T_1 < T_2 < \cdots < T_M$$

represent a series of "temperatures". The densities $\pi_2, \ldots, \pi_M$ are *tempered* versions of π_1.

Parallel Tempering

Note that

$$\pi_m(x) = \frac{\exp\left\{-\frac{H(x)}{T_m}\right\}}{Z(T_m)}$$

where

$$Z(T_m) = \int \exp\left\{-\frac{H(x)}{T_m}\right\}\, dx$$

may not be available analytically.

This integral needs to be finite for π_m to be a proper distribution.

Parallel Tempering

Example (Two component Normal mixture)

For the two component mixture

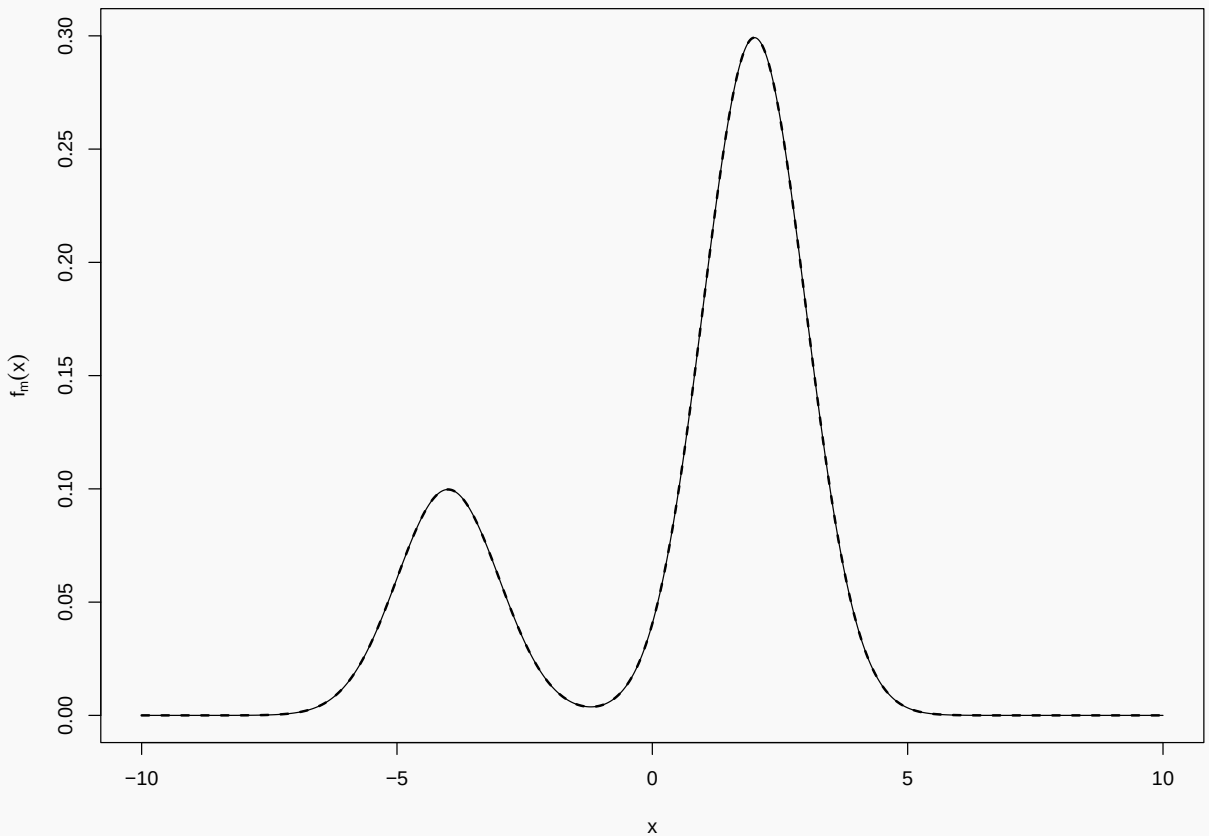
$$\pi(x) = \frac{1}{4}\phi(x+4) + \frac{3}{4}\phi(x-2)$$

the tempered distributions have modes that become increasingly flat.

Parallel Tempering

Example (Two component Normal mixture)

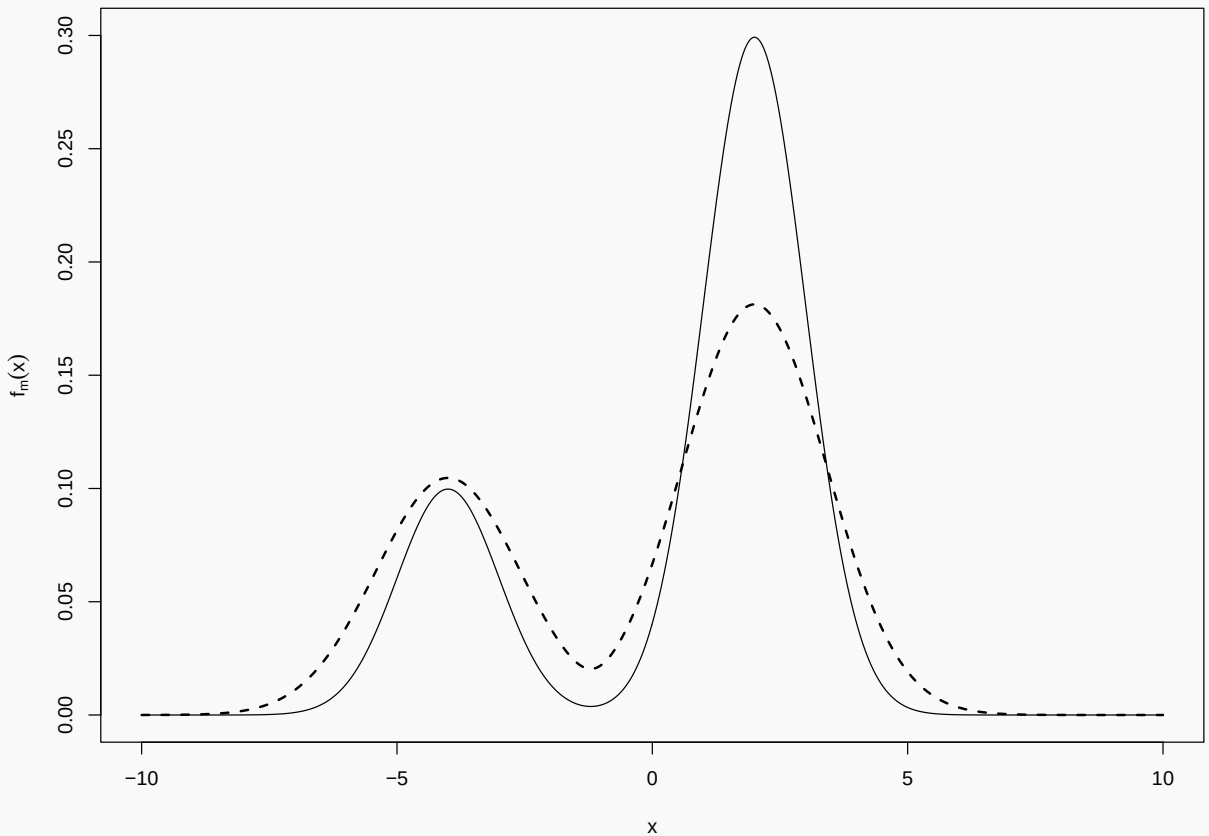
$T_1 = 1$:



Parallel Tempering

Example (Two component Normal mixture)

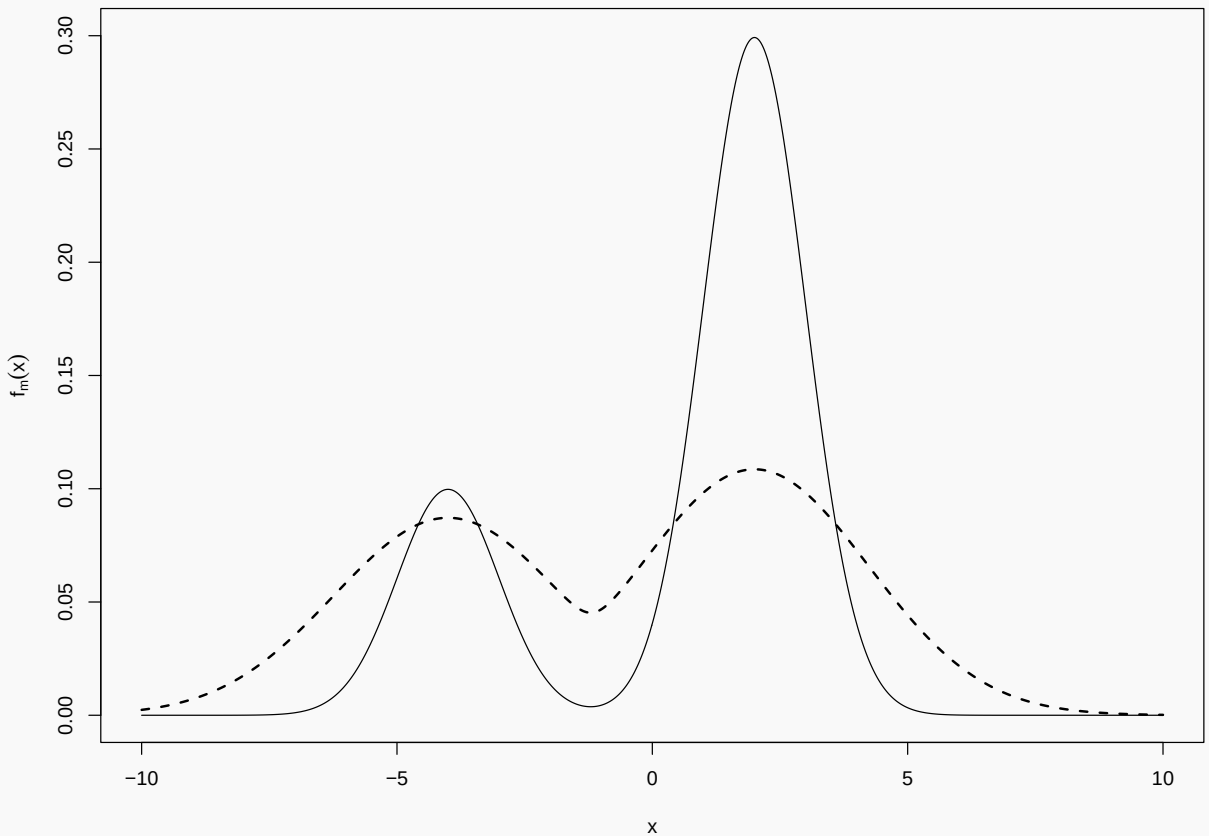
$T_2 = 2$:



Parallel Tempering

Example (Two component Normal mixture)

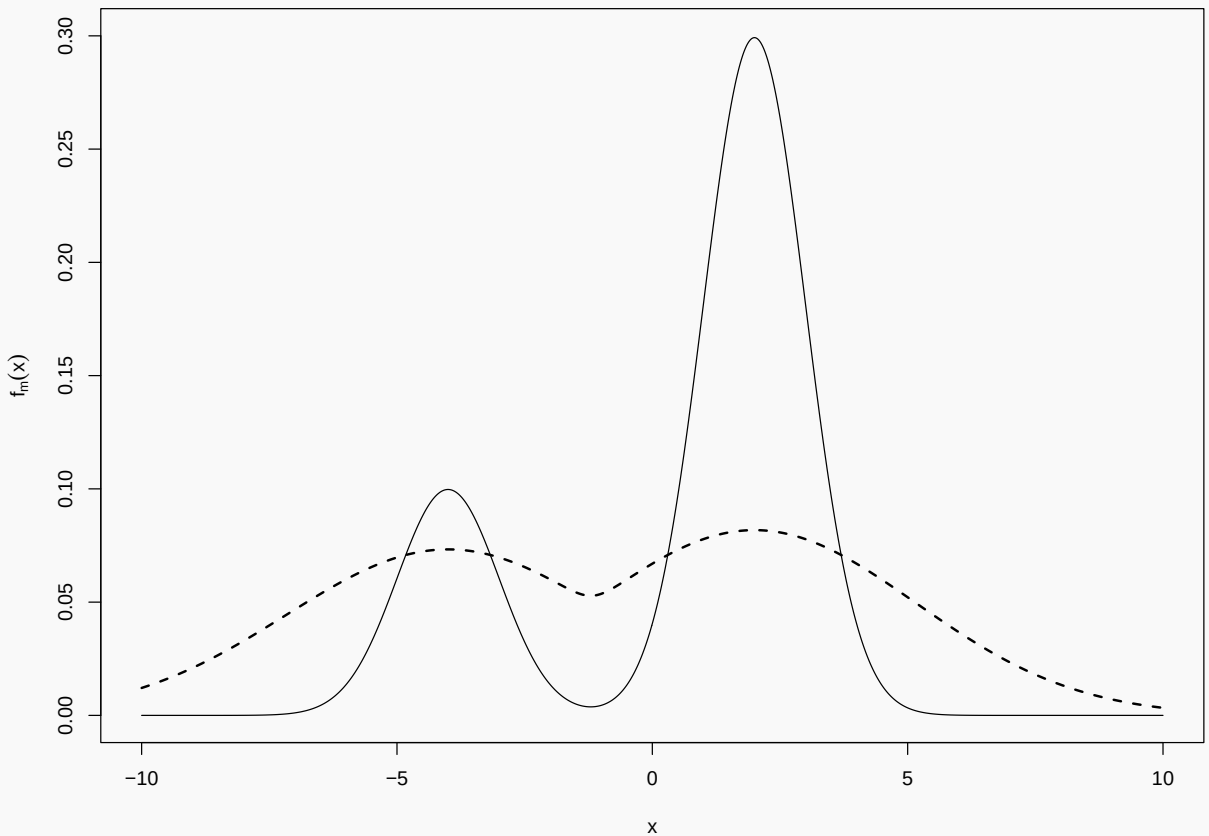
$T_3 = 5$:



Parallel Tempering

Example (Two component Normal mixture)

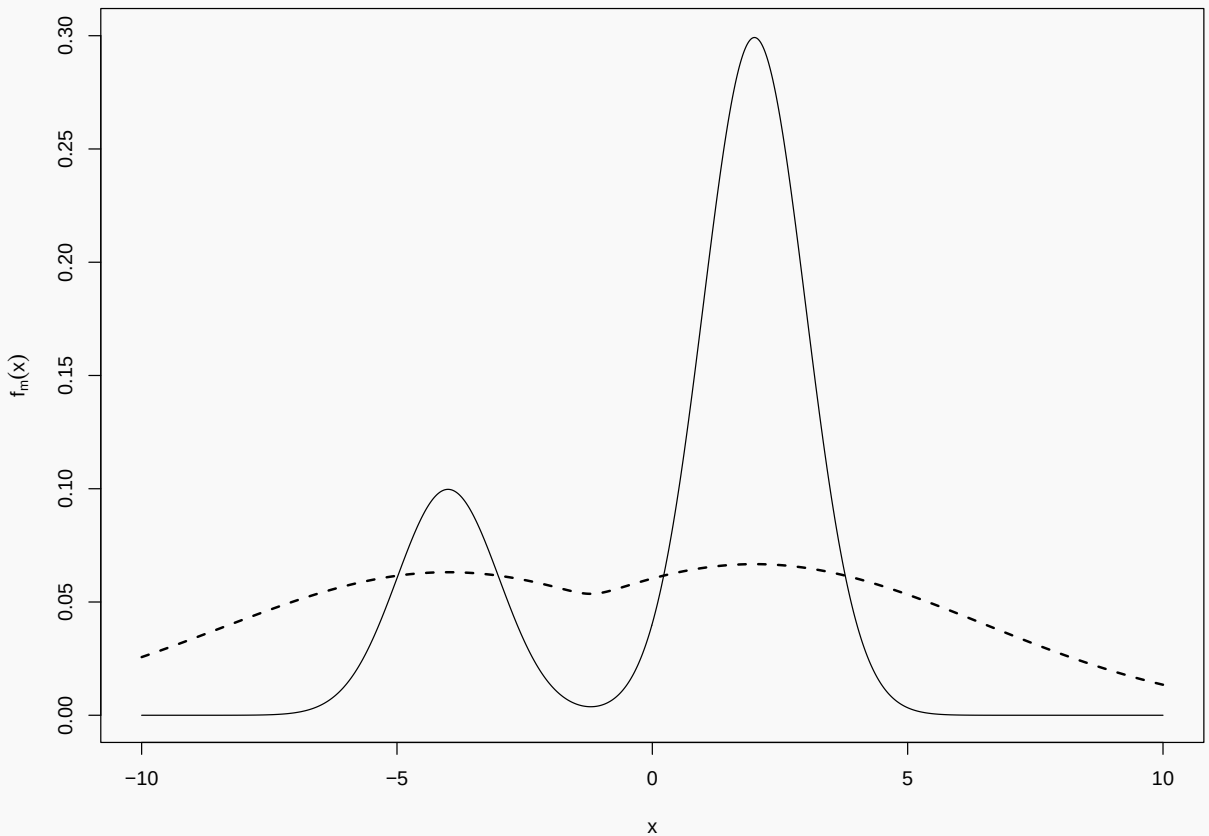
$T_4 = 10$:



Parallel Tempering

Example (Two component Normal mixture)

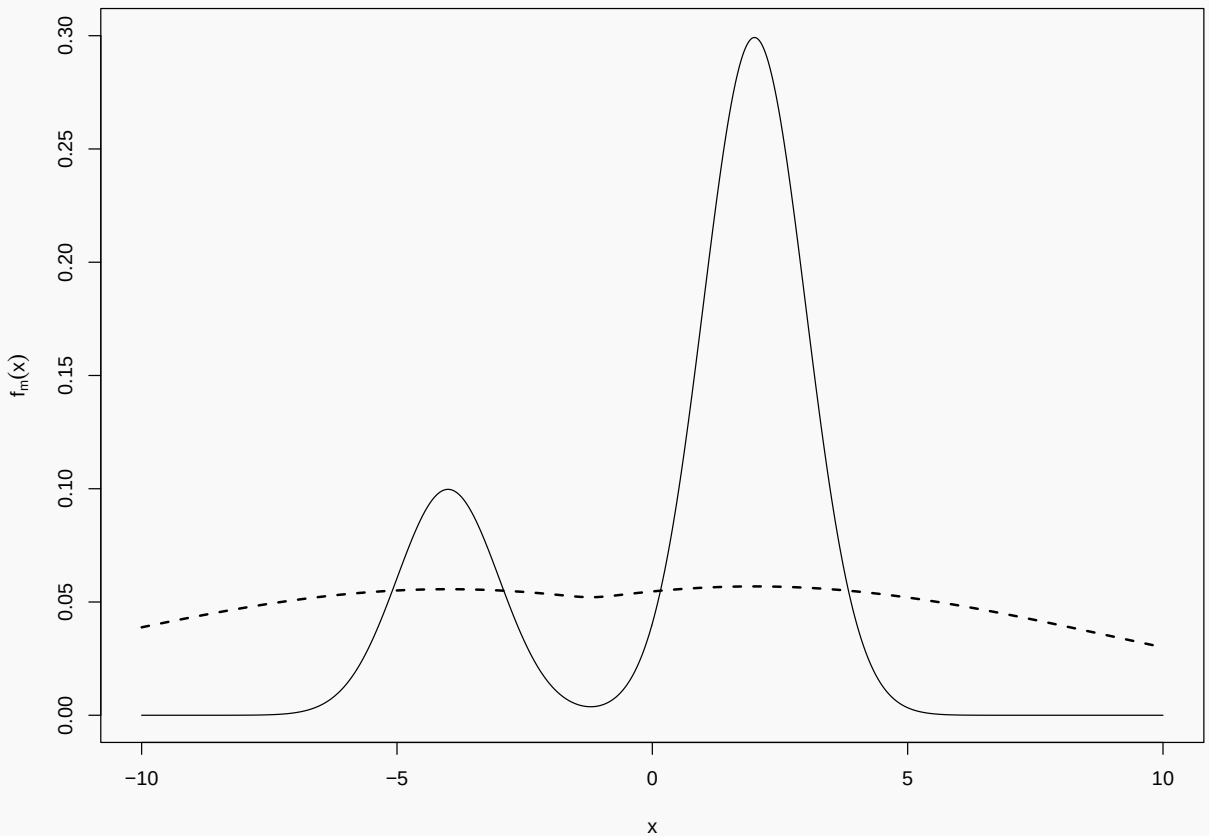
$T_5 = 20$:



Parallel Tempering

Example (Two component Normal mixture)

$T_6 = 50$:



Parallel Tempering

When T_m is large

$$\pi_m(x) \propto \exp\left\{-\frac{H(x)}{T_m}\right\}$$

is relatively "flat", that is, changes in x do not lead to large changes in $\pi_m(x)$.

Parallel tempering utilizes *within* chain and *between* chain updates. In the classical version, each iteration consists of

1. **Within:** an update for *each* chain using the MH kernels $P_1, \ldots, P_M$, then
2. **Between:** an update that attempts to swap x_m (from the chain for π_m at temperature T_m) with one of its temperature neighbours.

Parallel Tempering

Define the between-chain exchange probabilities

$$
\begin{aligned}
q_B(l,m) &= \Pr[\text{Swap chain } l \text{ with chain } m] \\
&= \begin{cases} \frac{1}{2} & 2 \leqslant l \leqslant M, m = l-1, l+1 \\ 1 & l = 1 \text{ and } m = 2, \text{ or } l = 1 \text{ and } m = 2 \\ 0 & \text{otherwise} \end{cases}
\end{aligned}
$$

Then the between-chain exchange that proposes to swap x_l with x_m is accepted with probability

$$
\alpha = \min\left\{1, \frac{\pi_l(x_m)\pi_m(x_l)q_B(m,l)}{\pi_l(x_l)\pi_m(x_m)q_B(l,m)}\right\}
$$

Parallel Tempering

For the tempered distributions

$$\frac{\pi_l(x_m)\pi_m(x_l)}{\pi_l(x_l)\pi_m(x_m)} = \exp\left\{-\frac{H(x_m)}{T_l} - \frac{H(x_l)}{T_m} + \frac{H(x_l)}{T_l} + \frac{H(x_m)}{T_m}\right\}$$

or equivalently

$$\exp\left\{(H(x_l) - H(x_m))\left(\frac{1}{T_l} - \frac{1}{T_m}\right)\right\}$$

The temperatures $T_2, \ldots, T_M$ should be chosen such that moves are accepted at neither too high nor too low a rate.

The samples collected for x_1 across iterations represent a sample from the marginal $\pi_1 \equiv \pi$, the target distribution.

Parallel Tempering

Tempered moves can be proposed in sequence in an "up-down" fashion. An update for x_1 (in the target chain for $\pi_1 \equiv \pi$) can be proposed as follows.

For the "up phase":
Propose y_1 from MH kernel $q_1(x_1, y)$ in chain 1
Propose y_2 from MH kernel $q_2(y_1, y)$ in chain 2
Propose y_3 from MH kernel $q_3(y_2, y)$ in chain 3
...
Propose y_{M-1} from MH kernel $q_{M-1}(y_{M-2}, y)$ in chain $M-1$
Propose y_M from MH kernel $q_M(y_{M-1}, y)$ in chain M

Parallel Tempering

For the "down phase":

Propose z_M from MH kernel $q_M(y_M, z)$ in chain M
Propose z_{M-1} from MH kernel $q_{M-1}(z_M, z)$ in chain $M-1$
Propose z_{M-2} from MH kernel $q_{M-2}(z_{M-1}, z)$ in chain $M-2$
...
Propose z_2 from MH kernel $q_2(z_3, z)$ in chain 2
Propose z_1 from MH kernel $q_1(z_2, z)$ in chain 1

This up and down sequence has proposed a move in the π_1 chain $x_1 \longrightarrow z_1$ via the intermediate steps through the other tempered chains.

Parallel Tempering

The acceptance probability for the move is the minimum of 1 and

$$\frac{\pi_1(z_1)}{\pi_1(x_1)} \frac{q_{\overline{U}}(z_1, \ldots, z_M, y_M) q_{\overline{D}}(y_M, \ldots, y_1, x_1)}{q_{\underline{U}}(x_1, y_1, \ldots, y_M) q_{\underline{D}}(y_M, z_M, \ldots, z_1)}$$

where

$$\begin{aligned}
q_{\overline{U}}(z_1, \ldots, z_M, y_M) &= q_1(z_1, z_2) q_2(z_2, z_3) \cdots q_M(z_M, y_M) \\
q_{\overline{D}}(y_M, \ldots, y_1, x_1) &= q_M(y_M, y_{M-1}) q_{M-1}(y_{M-1}, y_{M-2}) \cdots q_1(y_1, x_1) \\
q_{\underline{U}}(x_1, y_1, \ldots, y_M) &= q_1(x_1, y_1) q_2(y_1, y_2) \cdots q_M(y_{M-1}, y_M) \\
q_{\underline{D}}(y_M, z_M, \ldots, z_1) &= q_M(y_M, z_M) q_{M-1}(z_M, z_{M-1}) \cdots q_1(z_2, z_1)
\end{aligned}$$

If all of the proposals are symmetric $q_m(x, y) = q_m(y, x)$, then the ratio simplifies to

$$\frac{\pi_1(z_1)}{\pi_1(x_1)}$$

Evolutionary Monte Carlo

Evolutionary Monte Carlo combines multi-chain moves with tempered distributions in order to make exploration of the state space more effective.

For a "crossover" move, two chains exchange their values. For two "parent" chains that participate in the crossover select the first parent to be chain l with probability p_l proportional to

$$\exp\{-\beta H(\mathbf{x}_l)\}$$

where $\beta > 0$, and the second parent uniformly from the remaining chains.

Evolutionary Monte Carlo

The acceptance probability is computed by considering the various probabilities and densities in the proposal steps. For the crossover move

- Pick parent 1, $\mathbf{x}_l$, with probability p_l
- Pick parent 2, $\mathbf{x}_m$, uniformly, with probability $1/(M-1)$
- Pick the location of the crossover j, uniformly, with probability $1/(d-1)$
- Perform the crossover to obtain $\mathbf{x}_l^{\text{new}}$ and $\mathbf{x}_m^{\text{new}}$.

This yields the **numerator** in the Hastings ratio

$$p_m \times \frac{1}{M-1} \times \frac{1}{d-1} \times \pi_l(\mathbf{x}_l^{\text{new}}) \times \pi_m(\mathbf{x}_m^{\text{new}})$$

Considering the reverse move yields the denominator.

Simulated Tempering

In *simulated tempering*, the temperatures are also treated as quantities that are updated during the MCMC updates. Suppose

$$\pi_m(x, \tau_m) \propto \frac{\exp\{-H(x)/\tau_m\}}{Z(\tau_m)}$$

where $Z(\tau_m)$ is the normalizing constant for the mth tempered (conditional) distribution. A Metropolis-Hastings algorithm is constructed on the augmented state space

$$(x_1, \tau_1), (x_2, \tau_2), \ldots, (x_M, \tau_M)$$

and the sampled collected from chain 1 are samples from the target $\pi_1 \equiv \pi$.

Simulated Tempering

For a **mutation** move, chain m is selected uniformly from collection of chains, and then the pair (x_m, τ_m) is updated using a transition proposal

$$q_m((x_m, \tau_m), (x_m^{\text{new}}, \tau_m^{\text{new}}))$$

with the usual acceptance probability.

For the **exchange** move between l and m, the proposed exchange is accepted with probability

$$\alpha = \min\left\{1, \frac{\pi_l(x_m)\pi_m(x_l)}{\pi_l(x_l)\pi_m(x_m)}\right\}$$

where the probabilities of choosing pairs of chains l and m to attempt the exchange are equal.

Simulated Tempering

A Gibbs sampler update, involving proposals for x, and then for τ_l, can also be carried out. Specifically, we may propose an exchange for τ_l whilst holding x_l constant.

If l is selected uniformly from $\{1, \ldots, M\}$, then another index m is selected with probability q_{lm}, and a temperature swap $\tau_l \longleftrightarrow \tau_m$ is proposed. In this case, the acceptance probability is

$$\alpha = \min \left\{ 1, \frac{\pi_m(x_l)}{\pi_l(x_l)} \frac{q_{ml}}{q_{lm}} \right\}$$

Here

$$\frac{\pi_m(x_l)}{\pi_l(x_l)} = \frac{Z(\tau_m)}{Z(\tau_l)} \exp \left\{ -H(x_l) \left(\frac{1}{\tau_m} - \frac{1}{\tau_l} \right) \right\}$$

Simulated Tempering

In this formula, the normalizing constant ratio

$$\frac{Z(\tau_m)}{Z(\tau_l)}$$

must be computed. Here

$$Z(\tau) = \int \exp\{-H(x)/\tau\}\, dx$$

which, in the multivariate case is a d-dimensional integral that needs to be finite for each possible τ.

In some cases, analytical expressions can be obtained, but more generally an estimate $\widehat{Z}(\tau)$ must be used.

Simulated Annealing

Simulated annealing is a related optimization technique that utilizes *MCMC* and a temperature ladder. Suppose that the function $H(x)$ is to be minimized, let

$$\pi_T(x) \propto \exp\{-H(x)/T\}$$

and consider the sequence of **decreasing** temperatures

$$T_1 > T_2 > \cdots$$

Simulated annealing proceeds by running MCMC at the highest temperature T_1 for a fixed number N_1 of iterations, then changing temperatures to T_2 and running MCMC for N_2 iterations, and so on.

Simulated Annealing

As $i \longrightarrow \infty$ with $T_i \longrightarrow 0$, the MCMC samples become focussed at the mode(s) of $\pi(x)$, that is, the minima of $H(x)$. Theoretically, the logarithmic rate, where

$$T_i = \frac{K}{\log(M_i)} \qquad \text{with} \qquad M_i = \sum_{j=1}^{i} N_i$$

ensures convergence to the global maximum of π.

This rate is extremely slow, and typically geometric or linear cooling schedules are used; they generally have reasonable performance.

Simulated annealing is an option in the `optim` function in `R`.

Sequential Monte Carlo

Consider a sequence of probability distributions

$$\pi_1, \pi_2, \ldots,$$

defined on a sequence of state spaces

$$\mathbb{X}_1, \mathbb{X}_2,$$

where, for each n, $\mathbb{X}_n = \mathbb{X} \times \mathbb{X} \cdots \times \mathbb{X} \equiv \mathbb{X}^n$ for some $\mathbb{X}$.

We may write

$$\pi_n(x_{1:n}) = \frac{g_n(x_{1:n})}{Z_n}$$

where $x_{1:n} = (x_1, \ldots, x_n)$, g_n is an integrable non-negative function, and Z_n is a normalizing constant.

Sequential Monte Carlo: Two cases

1. **Dynamic problems:** where data are collected sequentially though time, as in *state-space models*

$$\begin{aligned} y_t &= h(x_t; \theta) + \epsilon_t \\ x_t &= \mu + \phi(x_{t-1} - \mu) + \varepsilon_t \end{aligned}$$

   where the observed data $\{y_1, \ldots, y_n\}$ are related to the unobserved states $\{x_1, \ldots, x_n\}$.

   Here the parameters $\theta, \mu, \phi, \sigma_\epsilon^2, \sigma_\varepsilon^2$ are also of interest.

   Here π_t is the posterior distribution derived from the data $\{y_1, \ldots, y_t\}$.

Sequential Monte Carlo: Two cases

2. **Static problems:** where data are collected together, but used sequentially.

   For example, in a Bayesian inference problem, we may have

   - π_1 being the posterior computed from y_1
   - π_2 being the posterior computed from y_1, y_2

   and so on.

In both cases, we may wish to perform a Monte Carlo calculation with respect to π_n, such as an expectation of the form

$$\mathbb{E}_{\pi_n}[h(x_{1:n})] = \int h(x_{1:n})\pi_n(x_{1:n})\, dx_{1:n}$$

Sequential Monte Carlo

Recall: **importance sampling** uses the identity

$$\int h(x)p(x)\,dx = \int h(x)p(x)\frac{p_0(x)}{p_0(x)}\,dx = \int \left\{ \frac{h(x)p(x)}{p_0(x)} \right\} p_0(x)\,dx$$

which demonstrates that

$$\mathbb{E}_p[h(X)] = \mathbb{E}_{p_0}\left[\frac{h(X)p(X)}{p_0(X)} \right]$$

so that an estimator of the LHS is

$$\widehat{I}_N^{(p_0)}(h) = \frac{1}{N}\sum_{i=1}^{N} \frac{h(X_i)p(X_i)}{p_0(X_i)}$$

where $X_1, \ldots, X_N \sim p_0(.)$.

Sequential Monte Carlo

Let

$$\pi_n(x_{1:n}) = \frac{g_n(x_{1:n})}{Z_n} \equiv \frac{g_n(x_{1:n})}{\int g(x_{1:n})\, dx_{1:n}} = \frac{w_n(x_{1:n}) p_n(x_{1:n})}{\int w_n(x_{1:n}) p_n(x_{1:n})\, dx_{1:n}}$$

where p_n is an importance distribution, and

$$w_n(x_{1:n}) = \frac{g_n(x_{1:n})}{p_n(x_{1:n})}$$

is the importance weight.

Sequential Monte Carlo

If an i.i.d. sample of size N $x_{1:n}^{(1)}, \ldots, x_{1:n}^{(N)}$ can be obtained from p_n, then the empirical pmf

$$\widehat{p}_n(\mathbf{x}) = \frac{1}{N} \sum_{i=1}^{N} \delta_{x_{1:n}^{(i)}}(\mathbf{x})$$

can be used to estimate π_n

$$\widehat{\pi}_n(\mathbf{x}) = \sum_{i=1}^{N} W_n^{(i)} \delta_{x_{1:n}^{(i)}}(\mathbf{x})$$

where

$$W_n^{(i)} = \frac{w_n(x_{1:n}^{(i)})}{N\widehat{Z}_n} \qquad \widehat{Z}_n = \frac{1}{N} \sum_{i=1}^{N} w_n(x_{1:n}^{(i)})$$

Sequential Monte Carlo

In a static Bayesian calculation, the full posterior can be written

$$\pi_n(\theta) = \frac{\mathcal{L}_n(\theta)\pi_0(\theta)}{\int \mathcal{L}_n(t)\pi_0(t)\,dt}$$

so could consider setting

$$p_n(\theta) = \pi_0(\theta)$$

for all n. The problem with this is that $\mathcal{L}_n(\theta)$ is likely to be very peaked compared to $\pi_0(\theta)$ once n gets moderately large.

Sequential Importance Sampling

Using the chain-rule factorization

$$\pi_n(x_{1:n}) = \pi_1(x_1) \prod_{j=2}^{n} \widetilde{\pi}_j(x_j | x_{1:(j-1)})$$

where

$$\widetilde{\pi}_j(x_j | x_{1:(j-1)}) = \frac{\pi_j(x_{1:j})}{\pi_{j-1}(x_{1:(j-1)})}$$

is the conditional distribution of element x_j, given

$$x_{1:(j-1)} = (x_1, \ldots, x_{j-1}).$$

Sequential Importance Sampling

It is then possible to construct a *Sequential Importance Sampling* (SIS) density

$$p_n(x_{1:n}) = p_1(x_1) \prod_{j=2}^{n} \widetilde{p}_j(x_j | x_{1:(j-1)})$$

where $\widetilde{p}_j(x_j | x_{1:(j-1)})$ is again a conditional distribution.

Sequential Importance Sampling

The importance sampling weight then takes the form

$$w_n(x_{1:n}) = \frac{\pi_n(x_{1:n})}{p_n(x_{1:n})} = \frac{\pi_1(x_1)}{p_1(x_1)} \prod_{j=2}^{n} \frac{\widetilde{\pi}_j(x_j|x_{1:(j-1)})}{\widetilde{p}_j(x_j|x_{1:(j-1)})}$$

so therefore

$$\begin{aligned} w_n(x_{1:n}) &= w_{n-1}(x_{1:(n-1)}) \frac{\widetilde{\pi}_n(x_n|x_{1:(n-1)})}{\widetilde{p}_n(x_n|x_{1:(n-1)})} \\ w_{n-1}(x_{1:(n-1)}) &= w_{n-2}(x_{1:(n-2)}) \frac{\widetilde{\pi}_{n-1}(x_{n-1}|x_{1:(n-2)})}{\widetilde{p}_{n-1}(x_{n-1}|x_{1:(n-2)})} \end{aligned}$$

and so on, so there is a *recursive* weight calculation,

$$w_j(x_{1:j}) = w_{j-1}(x_{1:(j-1)}) \frac{\widetilde{\pi}_j(x_j|x_{1:(j-1)})}{\widetilde{p}_j(x_j|x_{1:(j-1)})}.$$

Sequential Importance Sampling

In the numerator of the weights, we have

$$\widetilde{\pi}_j(x_j|x_{1:(j-1)}) = \frac{\pi_j(x_{1:j})}{\pi_{j-1}(x_{1:(j-1)})}$$

but to compute this we need the marginal distributions π_j and π_{j-1}, which may not be available analytically.

Sequential Importance Sampling

A strategy for the **static** Monte Carlo case involves, for each j, defining the *auxiliary distributions*

$$p_{0j}(x_{1:j}) \qquad j = 1, \ldots, n$$

such that $p_{0n}(x_{1:n}) \equiv \pi_n(x_{1:n})$, and where p_{0j} resembles π_j, but is **directly available**.

Combining Importance Sampling and MCMC

Recall that

$$\int h(x)\pi(x)\,dx = \int h(x)\frac{\pi(x)}{p_0(x)}p_0(x)\,dx = \int h(x)w(x)p_0(x)\,dx$$

If P is a Markov transition kernel with stationary (invariant) distribution π, that is, for all x

$$\int \pi(z)P(z,x)\,dz = \pi(x).$$

Combining Importance Sampling and MCMC

Then

$$\int \frac{\pi(z)}{p_0(z)} P(z,x) p_0(z)\, dz = \pi(x)$$

or

$$\int w(z) P(z,x) p_0(z)\, dz = \pi(x).$$

and so substituting in for $\pi(x)$

$$\begin{aligned}\int h(x)\pi(x)\, dx &= \int h(x) \left\{ \int w(z) P(z,x) p_0(z)\, dz \right\} dx \\ &= \iint h(x) w(z) P(z,x) p_0(z)\, dz\, dx.\end{aligned}$$

Combining Importance Sampling and MCMC

Hence, from the importance sampling identity

$$\mathbb{E}_\pi[h(X)] = \mathbb{E}_{p_0}[w(X)h(X)] = \int\int h(x)w(z)P(z,x)p_0(z)\,dzdx$$

and the estimator

$$\frac{1}{N}\sum_{i=1}^{N} h(X_i)w(Z_i)$$

can be constructed, where

$$Z_1,\ldots,Z_N \sim p_0 \qquad\qquad X_i \sim f_{X|Z}(x|z_i),\ i=1,\ldots,N$$

where $f_{X|Z}(x|z)$ is the conditional density for X given Z.

Annealed Importance Sampling

Annealed Importance Sampling (AIS) uses a SIS approach, but with auxiliary densities **of the same dimension** defined by

$$p_{0j}(x) = c_j g_{0j}(x) = c_j \{g_0(x)\}^{1-\xi_j} \{g_n(x)\}^{\xi_j}$$

where p_0 is a "diffuse" distribution, and

$$0 = \xi_0 < \xi_1 < \cdots < \xi_n = 1.$$

Annealed Importance Sampling

We have a "path" from $p_{00}(x) \equiv p_0(x)$ to the target distribution

$$p_{0n}(x) \equiv \pi_n(x) = c_n g_n(x).$$

For $j = 1, \ldots, n-1$, let P_j be a Markov kernel with invariant distribution p_{0j}.

Annealed Importance Sampling

The AIS algorithm proceeds as follows:

1. Sample x_1 from p_{00}, set $w_0 = 1/g_0(x_1)$.

2. For $j = 1, 2, \ldots, n-1$,

   (i) Sample x_{j+1} from $P_j(x_j, \cdot)$;

   (ii) Set

$$w_j = w_{j-1} \frac{p_{0j}(x_j)}{p_{0j}(x_{j+1})} = w_{j-1} \frac{g_{0j}(x_j)}{g_{0j}(x_{j+1})}.$$

3. Return to 1. and repeat to produce N samples and weights

$$x_n^{(1)}, \ldots, x_n^{(N)} \qquad\qquad w_n^{(1)}, \ldots, w_n^{(N)}$$

   where

$$w_n = \frac{1}{g_0(x_1)} \times \frac{g_{01}(x_1)}{g_{01}(x_2)} \times \frac{g_{02}(x_2)}{g_{02}(x_3)} \times \cdots \times \frac{g_{0\,n-1}(x_{n-1})}{g_{0\,n-1}(x_n)} \times g_n(x_n).$$

Annealed Importance Sampling

Consider the **reverse kernel** P_j^R defined using Bayes Theorem as

$$P_j^R(x_{j+1}, x_j) = \frac{P_j(x_j, x_{j+1}) p_{0j}(x_j)}{p_{0j}(x_{j+1})} = \frac{P_j(x_j, x_{j+1}) g_{0j}(x_j)}{g_{0j}(x_{j+1})},$$

and let

$$\begin{aligned} g^\star(x_{1:n}) &= g_n(x_n) \prod_{j=1}^{n-1} P_j^R(x_{j+1}, x_j) \\ g_0^\star(x_{1:n}) &= g_0(x_1) \prod_{j=1}^{n-1} P_j(x_j, x_{j+1}) \end{aligned}$$

where $g_0^\star(x)$ is proportional to the AIS proposal joint density $p_0^\star(x)$.

Annealed Importance Sampling

Define

$$w^\star(x_{1:n}) = \frac{g^\star(x_{1:n})}{g_0^\star(x_{1:n})}$$

as the importance sampling weight for the augmented sample

$$x_{1:n} = (x_1, \ldots, x_n)$$

generated from $p_0^\star$.

Annealed Importance Sampling

We have that

$$w^\star(x_{1:n}) = \frac{g_n(x_n)\prod_{j=1}^{n-1} P_j^R(x_{j+1}, x_j)}{g_0(x_1)\prod_{j=1}^{n-1} P_j(x_j, x_{j+1})}$$

$$= \frac{g_n(x_n)\prod_{j=1}^{n-1} \frac{P_j(x_j, x_{j+1})g_{0j}(x_j)}{g_{0j}(x_{j+1})}}{g_0(x_1)\prod_{j=1}^{n-1} P_j(x_j, x_{j+1})}$$

Annealed Importance Sampling

That is, all the terms in $P_j(\cdot, \cdot)$ cancel, and

$$w^\star(x) = \frac{1}{g_0(x)} \times \frac{g_{01}(x_1)}{g_{01}(x_2)} \times \frac{g_{02}(x_2)}{g_{02}(x_3)} \times \cdots \times \frac{g_{0\,n-1}(x_{n-1})}{g_{0\,n-1}(x_n)} \times g_n(x_n)$$

This is the identical weight to the one computed recursively above.

Annealed Importance Sampling

Note that for each $j = 1, \ldots, n-1$,

$$\int P_j^R(x_{j+1}, x_j)\, dx_j = 1$$

so the marginal distribution of x_n from

$$\pi^\star(x_{1:n}) = c^\star g^\star(x_{1:n})$$

obtained by integrating $g^\star(x_{1:n})$ with respect to

$$x_1, x_2, \ldots, x_{n-1}$$

is precisely the true target $\pi_n(x_n)$.

Sequential Monte Carlo for State-Space Models

Recall the state-space model: for $t = 1, \ldots, T$

$$\begin{aligned} y_t &= g(x_t; \theta) + \epsilon_t \\ x_{t+1} &= h(x_t; \theta) + \varepsilon_t \end{aligned}$$

where only the series $y_1, \ldots, y_T$, is observed. Sequences $\{\epsilon_t\}$ and $\{\varepsilon_t\}$ are independent zero mean random variables.

If $y_{1:t} = (y_1, \ldots, y_t)$, $x_{1:t} = (x_1, \ldots, x_t)$. we must carry out

- **Filtering:** Compute $p(x_t|y_{1:t})$
- **Prediction:** Compute $p(x_{t+1}|y_{1:t})$
- **Smoothing:** Compute $p(x_t|y_{1:T})$

Filtering: By Bayes theorem

$$p(x_t|y_{1:t}) = \frac{p(y_t|x_t)p(x_t|y_{1:(t-1)})}{p(y_t|y_{1:(t-1)})} \propto p(y_t|x_t)p(x_t|y_{1:(t-1)})$$

where the function $p(y_t|x_t)$ is usually a straightforward conditional density, but the function $p(x_t|y_{1:(t-1)})$ is the density computed at the previous time point **prediction** step.

Prediction:

$$p(x_{t+1}|y_{1:t}) \quad = \quad \int p(x_{t+1}|x_t)p(x_t|y_{1:t})\, dx_t$$

where $p(x_{t+1}|x_t)$ is again a conditional density derived from the state equation, and $p(x_t|y_{1:t})$ is the posterior density computed at the time point **filtering** step.

Sequential Monte Carlo for State-Space Models

We consider simulation-based versions of these calculations.

For example, in the **filtering** step, suppose we have a sample

$$z_t^{(1)}, \ldots, z_t^{(N)}$$

from $p(x_t|y_{1:(t-1)})$ obtained at the previous prediction step. Then we can obtain a *particle approximation* to $p(x_t|y_{1:t})$ as

$$\widehat{p}(x_t|y_{1:t}) = \frac{1}{N}\sum_{i=1}^{N} p(y_t|z_t^{(i)})\delta_{z_t^{(i)}}$$

Note also that the denominator $p(y_t|y_{1:(t-1)})$ can be approximated by the sum of the weights

$$\widehat{p}(y_t|y_{1:(t-1)}) = \frac{1}{N}\sum_{i=1}^{N} p(y_t|z_t^{(i)})$$

which is useful in approximating the likelihood by

$$\widehat{\mathcal{L}}(y_{1:t}) = \widehat{p}(y_1)\prod_{t=2}^{T}\widehat{p}(y_t|y_{1:(t-1)})$$

We can also produce a sample from the particle approximation

$$\widehat{p}(x_t|y_{1:t})$$

by performing a *weighted resampling* from $(z_t^{(1)}, \ldots, z_t^{(N)})$ with weights proportional to $p(y_t|z_t^{(i)})$.

Denote this resampled vector

$$v_t^{(1)}, \ldots, v_t^{(N)}$$

Sequential Monte Carlo for State-Space Models

For the **prediction** step, the particle approximation to $p(x_{t+1}|y_{1:t})$ is obtained using a similar Monte Carlo strategy

$$\widehat{p}(x_{t+1}|y_{1:t}) = \frac{1}{N}\sum_{i=1}^{N} p(x_{t+1}|v_t^{(i)})$$

Also, a sample from this approximation can be obtained by propagating the samples $v_t^{(1)}, \ldots, v_t^{(N)}$ forward by sampling the transition densities $p(x_{t+1}|v_t^{(i)})$ for each i.

This yields the sample

$$z_{t+1}^{(1)}, \ldots, z_{t+1}^{(N)}$$

and the recursion continues.

To perform SIS for $\pi_n(x_{1:n}) = g_n(x_{1:n})/Z_n$, use the IS density

$$p_n(x_{1:n}) = p_1(x_1) \prod_{j=2}^{n} \widetilde{p}_j(x_j|x_{1:(j-1)})$$

with weights given by $w_1(bx_1) = \pi_1(x_1)/p_1(x_1)$ and

$$\begin{aligned} w_j(x_{1:j}) &= w_{j-1}(x_{1:(j-1)}) \frac{\widetilde{\pi}_j(x_j|x_{1:(j-1)})}{\widetilde{p}_j(x_j|x_{1:(j-1)})} \\ &= w_{j-1}(x_{1:(j-1)}) \frac{\pi_j(x_{1:j})}{\pi_{j-1}(x_{1:(j-1)})\widetilde{p}_j(x_j|x_{1:(j-1)})} \end{aligned}$$

If the normalizing constants Z_j for $j = 1, \ldots, n$ are unknown, the weights can be replaced by $w_1(x_1) = g_1(x_1)/p_1(x_1)$ and

$$w_j(x_{1:j}) = w_{j-1}(x_{1:(j-1)}) \frac{g_j(x_{1:j})}{g_{j-1}(x_{1:(j-1)})\widetilde{p}_j(x_j|x_{1:(j-1)})}$$

$$= w_{j-1}(x_{1:(j-1)})\alpha_j(x_{1:j})$$

say.

Sequential Monte Carlo for State-Space Models

The Monte Carlo computation is carried out by sampling

$$x_1^{(1)}, \ldots, x_1^{(N)}$$

from p_1, then at step $j \geqslant 2$ sampling for $i = 1, \ldots, N$

$$x_j^{(1)}, \ldots, x_j^{(N)} \quad \text{from} \quad \widetilde{p}_j(.|x_{1:(j-1)}^{(i)})$$

The importance sampling weights are $w_1(x_1^{(i)})$ and, for $j \geqslant 2$,

$$w_j(x_{1:j}^{(i)}) = w_{j-1}(x_{1:(j-1)}^{(i)})\alpha_j(x_{1:j}^{(i)})$$

We then have

$$\widehat{\pi}_n(\mathbf{x}) = \sum_{i=1}^{N} W_n^{(i)} \delta_{x_{1:n}^{(i)}}(\mathbf{x})$$

where

$$W_n^{(i)} = \frac{w_n(x_{1:n}^{(i)})}{N\widehat{Z}_n} \qquad \widehat{Z}_n = \frac{1}{N}\sum_{i=1}^{N} w_n(x_{1:n}^{(i)})$$

in the usual way.

Sequential Monte Carlo for State-Space Models

In a simple state-space model, write

$$\begin{aligned} \text{Observation equation} &: p_{Y|X}(y_j|x_j) \\ \text{State equation} &: p_X(x_j|x_{j-1}) \end{aligned}$$

with $X_1 \sim p_{X_1}$.

In light of the observed data $y_{1:n} = (y_1, \ldots, y_n)$, the posterior of interest is

$$\pi_n(x_{1:n}) = p(x_{1:n}|y_{1:n}) = \frac{p(x_{1:n}, y_{1:n})}{p(y_{1:n})} = \frac{g_n(x_{1:n}, y_{1:n})}{Z_n(y_{1:n})}$$

We have

$$g_n(x_{1:n}, y_{1:n}) = p_{X_1}(x_1) \left\{ \prod_{j=2}^{n} p_X(x_j|x_{j-1}) \right\} \left\{ \prod_{j=1}^{n} p_{Y|X}(y_j|x_j) \right\}$$

and

$$Z_n(y_{1:n}) = \int g_n(x_{1:n}, y_{1:n}) \, dx_{1:n}$$

Now, for SMC, the optimal choice of IS density $\widetilde{p}_j(x_j|x_{1:(j-1)})$ is

$$\widetilde{\pi}_j(x_j|x_{1:(j-1)})$$

which in the simple state-space model is given by

$$\begin{aligned}\widetilde{\pi}_j(x_j|x_{1:(j-1)}) \equiv \widetilde{\pi}_j(x_j|y_{1:j}, x_{1:(j-1)}) &= \frac{p_{Y|X}(y_j|x_j)p_X(x_j|x_{j-1})}{p(y_j|x_{j-1})} \\ &= \frac{p_{Y|X}(y_j|x_j)p_X(x_j|x_{j-1})}{\int p_{Y|X}(y_j|x)p_X(x|x_{j-1})\,dx}\end{aligned}$$

as, conditional on $x_{1:(j-1)}$, x_j is independent of $y_{1:(j-1)}$.

In most cases (specifically, outside of the Gaussian case), the optimal choice is not feasible: recall that we need successive samples from the IS densities

$$\widetilde{p}_j(x_j|x_{1:(j-1)}) = \widetilde{\pi}_j(x_j|y_{1:j}, x_{1:(j-1)}) = \widetilde{\pi}_j(x_j|y_j, x_{(j-1)}).$$

We might choose instead

$$\widetilde{p}_j(x_j|x_{1:(j-1)}) = p_X(x_j|x_{1:(j-1)})$$

which typically is straightforward to sample from, and yields the recursive weight calculation

$$w_j(x_{1:j}) = w_{j-1}(x_{1:(j-1)})p_{Y|X}(y_j|x_j)$$

Example (Stochastic volatility model)

Suppose

$$\text{Observation equation} \quad : \quad p_{Y|X}(y_j|x_j) \equiv \mathcal{N}(0, \exp\{2x_j\})$$

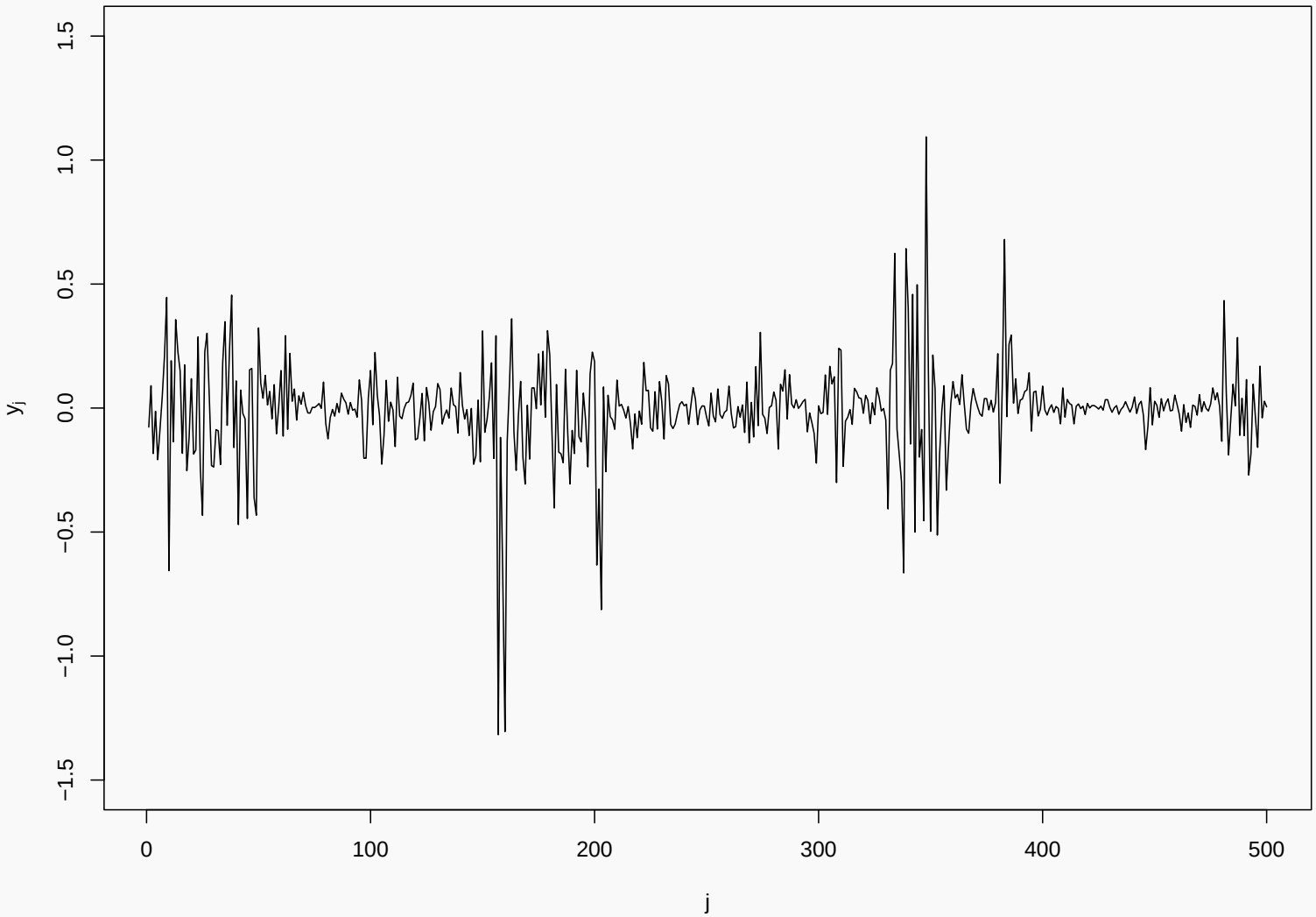
$$\text{State equation} \quad : \quad p_X(x_j|x_{j-1}) \equiv \mathcal{N}(\mu + \phi(x_{j-1} - \mu), \sigma^2)$$

with $p_{X_1}(x) \equiv \mathcal{N}(\mu, \sigma^2/(1-\phi^2))$. We regard μ, ϕ and σ as known constants.

This model is used in the analysis of financial time series: it is a non-linear state-space model, due to the non-linear dependence of the distribution of y_j on x_j.

Example (Stochastic volatility model)

$n = 500$ realizations from the stochastic volatility model.



Example (Stochastic volatility model)

Here, for $j \geqslant 2$,

$$p_{Y|X}(y_j|x_j) = \left(\frac{e^{-2x_j}}{2\pi}\right)^{1/2} \exp\{-e^{-2x_j}y_j^2/2\}$$

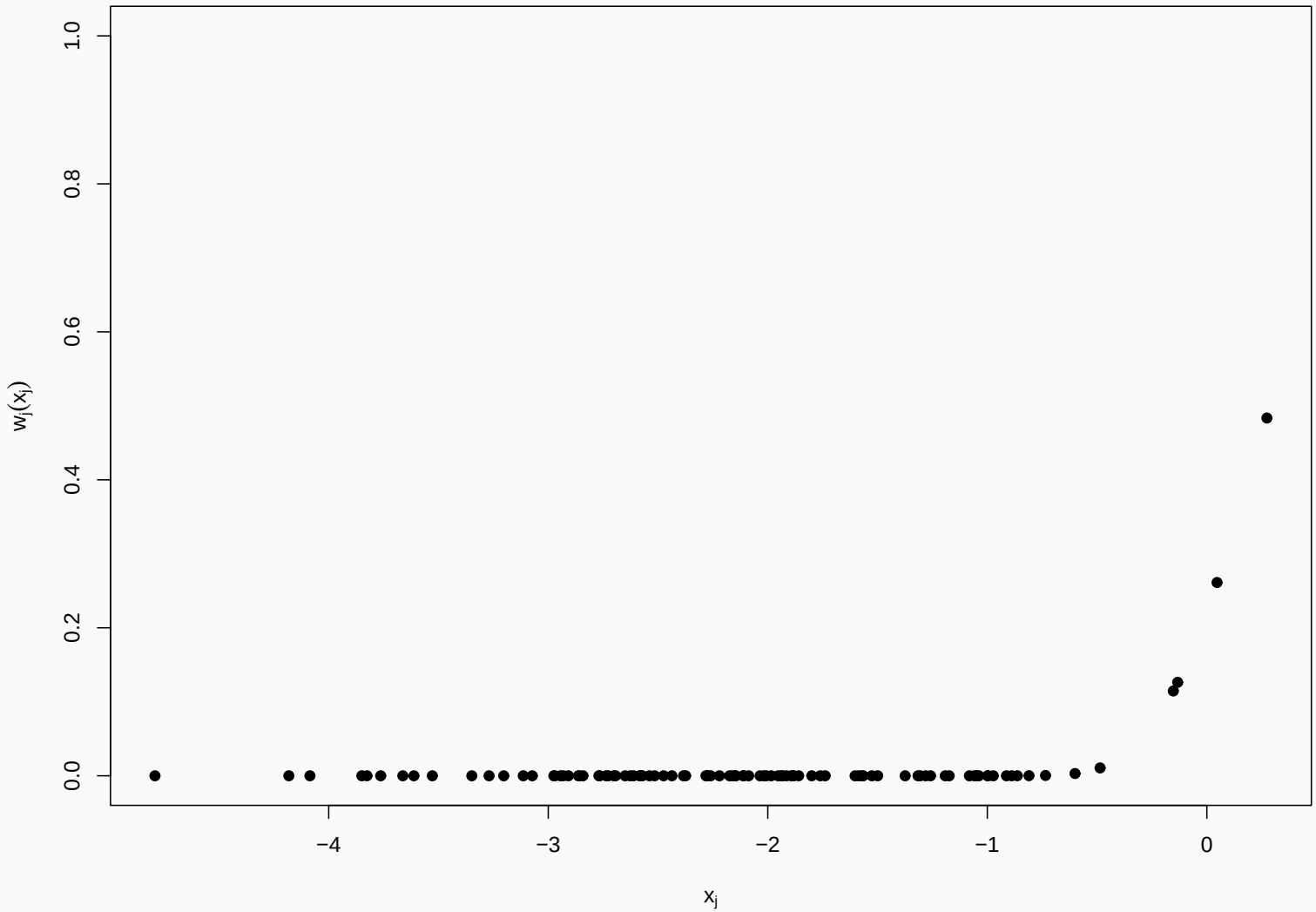
and the recursive weight calculation is based on

$$w_j(x_{1:j}) = \prod_{k=1}^{j} \frac{\widetilde{\pi}_k(x_k|x_{1:(k-1)})}{\widetilde{p}_k(x_k|x_{1:(k-1)})} = \prod_{k=1}^{j} p_{Y|X}(y_k|x_k)$$

that is, the likelihood up to observation j.

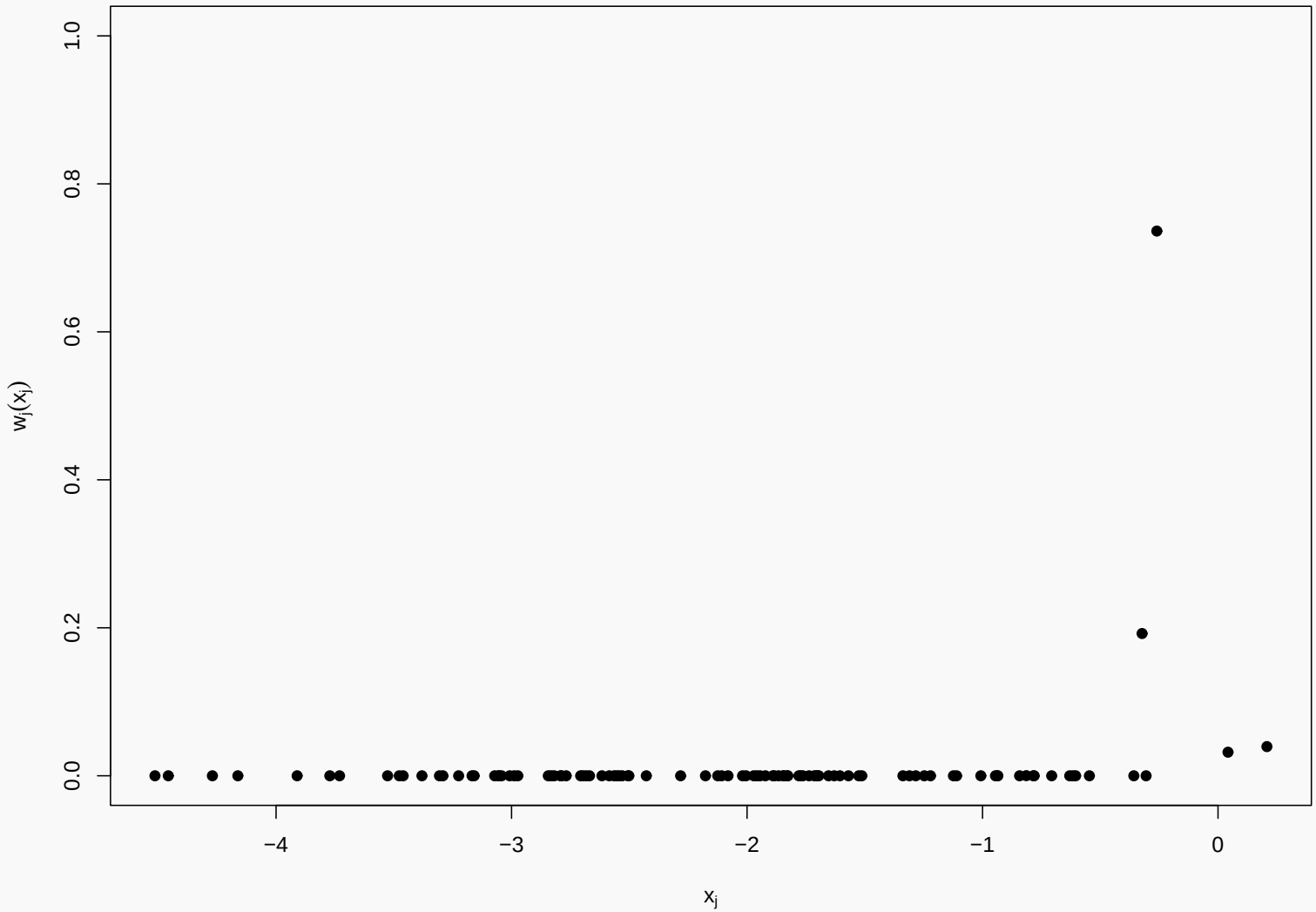
Example (Stochastic volatility model)

$N = 100$ particles and weights at $j = 1$



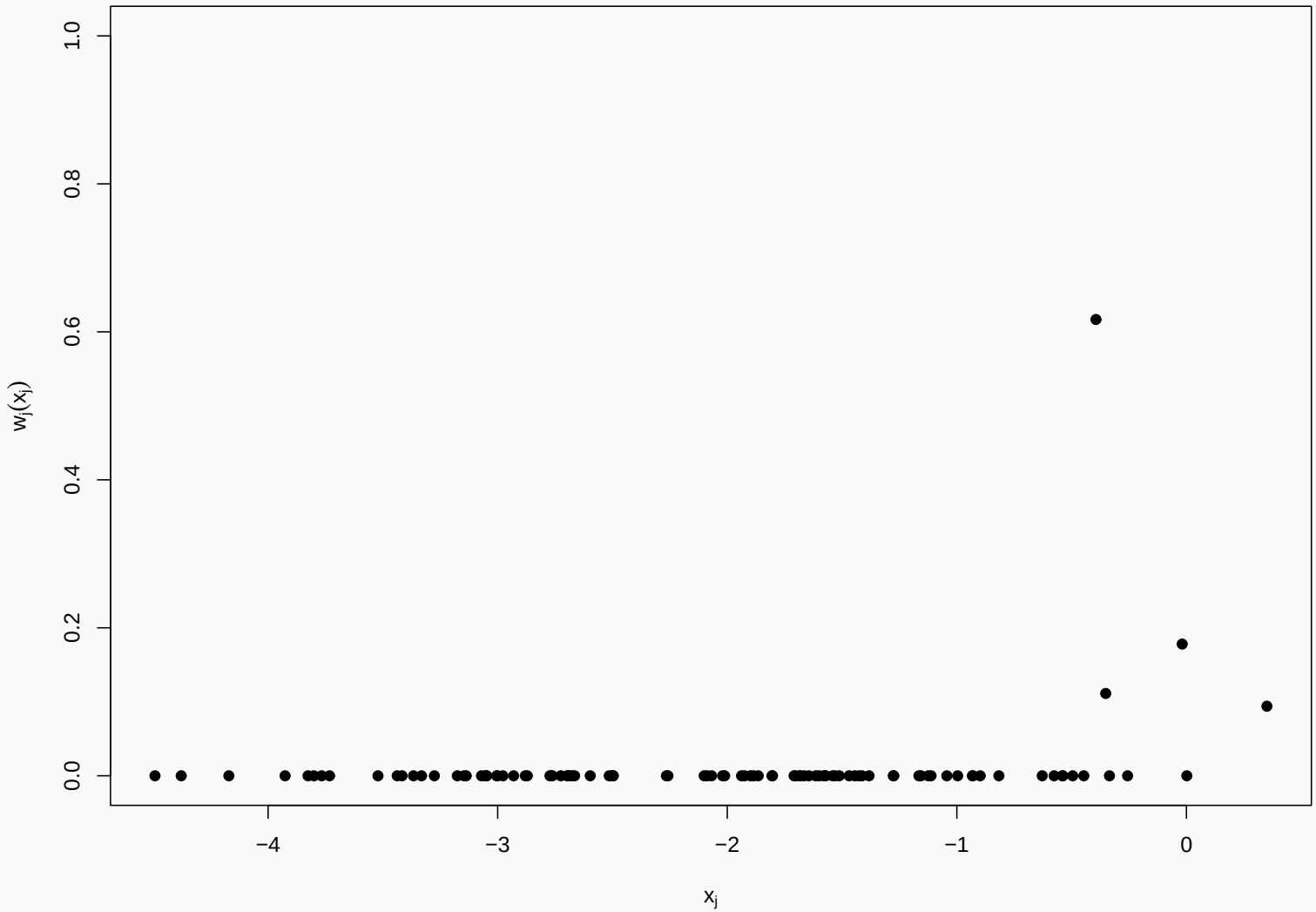
Example (Stochastic volatility model)

$N = 100$ particles and weights at $j = 2$



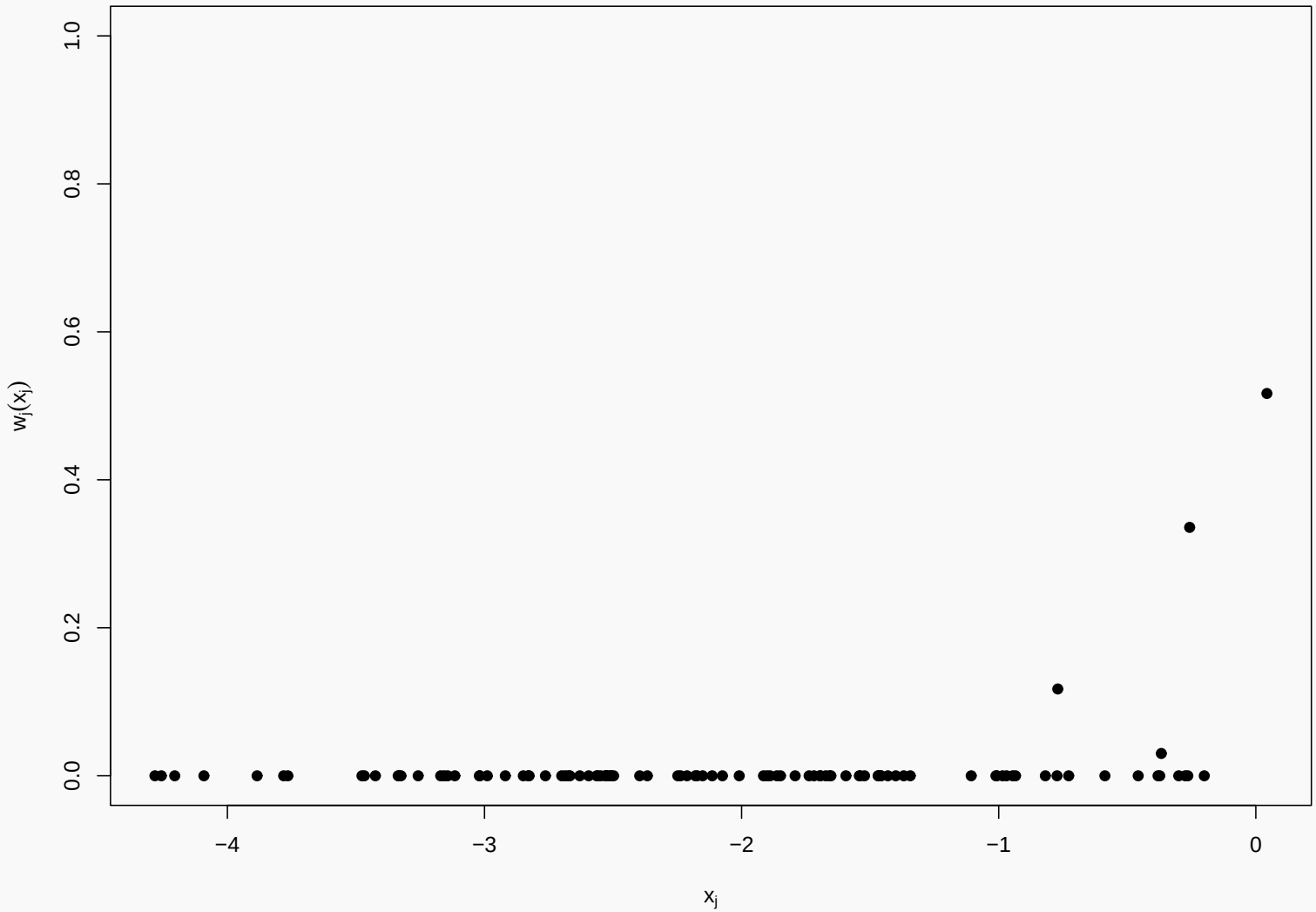
Example (Stochastic volatility model)

$N = 100$ particles and weights at $j = 3$



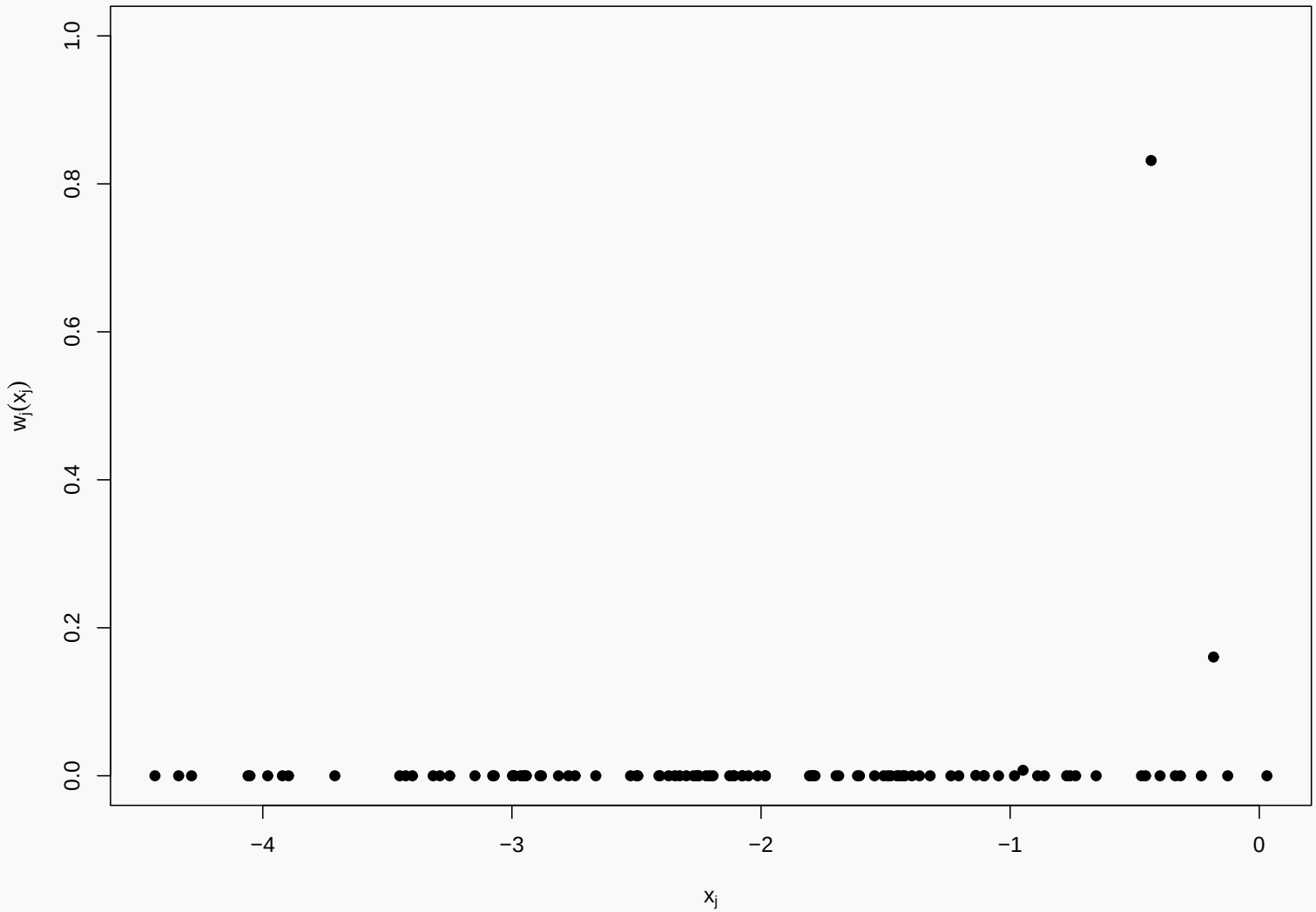
Example (Stochastic volatility model)

$N = 100$ particles and weights at $j = 4$



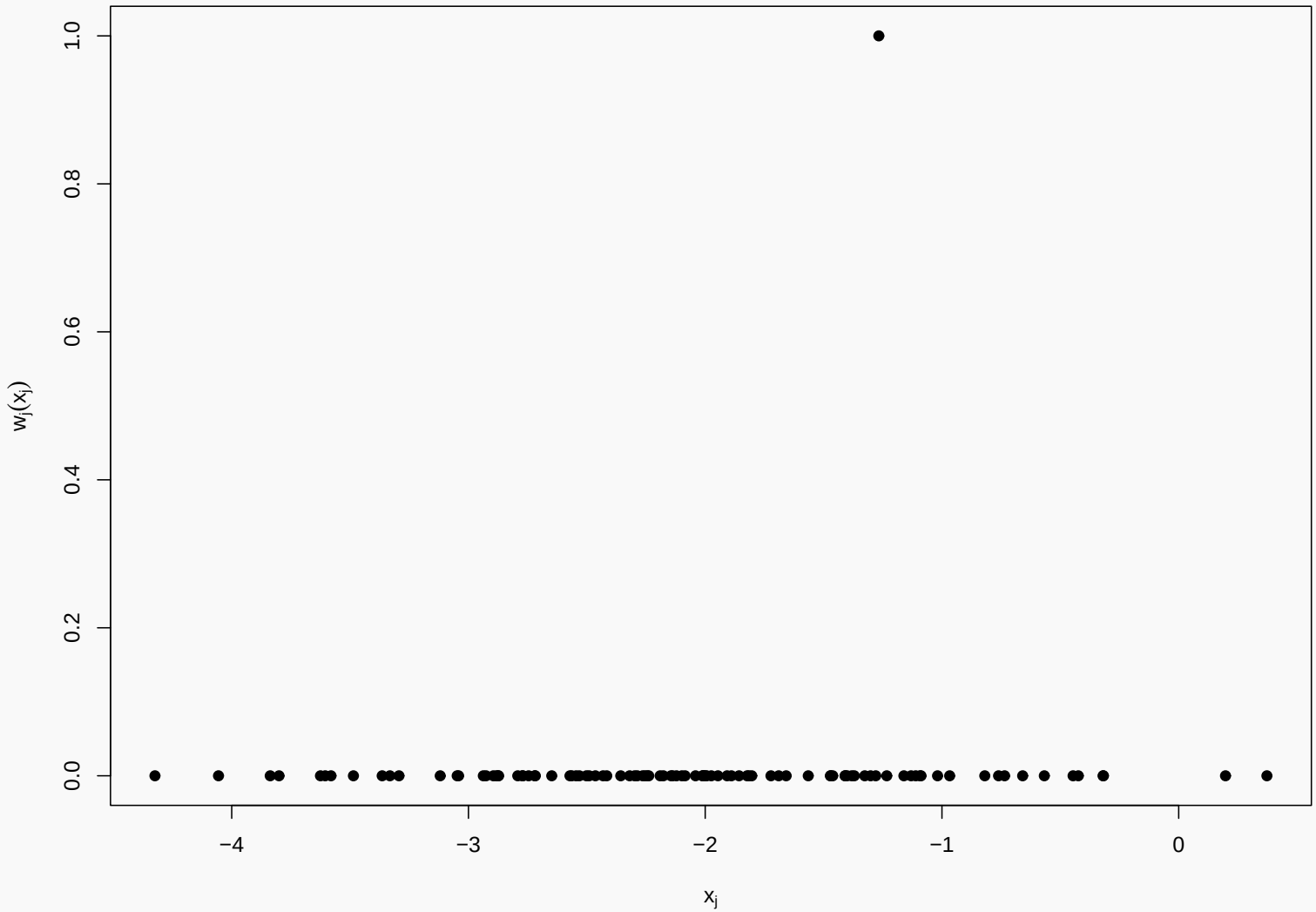
Example (Stochastic volatility model)

$N = 100$ particles and weights at $j = 5$



Example (Stochastic volatility model)

$N = 100$ particles and weights at $j = 20$



Particle Degeneracy

A problem with this strategy is that the weights start to **degenerate** as j increases;

- the weights that define the Monte Carlo estimate $\widehat{\pi}_n(\mathbf{x})$ are very small for most i, with only a few large weights.
- the empirical variance of the collection of unnormalized weights $\{w_j(x_j^{(i)})\}$ for $i = 1, \ldots, N$ increases
- this is mitigated, but not resolved by a better choice of $\widetilde{p}_j(x_j|x_{1:(j-1)})$

Particle Degeneracy

- could try a normal approximation of

$$\begin{aligned} p(x_j|y_{1:j}, x_{1:(j-1)}) &\propto p_{Y|X}(y_j|x_j)p_X(x_j|x_{j-1}) \\ &= \exp\left\{-x_j - \frac{y_j^2}{2e^{2x_j}} - \frac{1}{2\sigma^2}(x_j - \mu_j)^2\right\} \end{aligned}$$

where $\mu_j = \mu + \phi(x_{j-1} - \mu)$.

Particle Degeneracy

Example (Stochastic volatility model)

Here the mode of

$$p(x|y_j, x_{j-1}) = \exp\left\{-x - \frac{y_j^2}{2e^{2x}} - \frac{1}{2\sigma^2}(x-\mu_j)^2\right\}$$

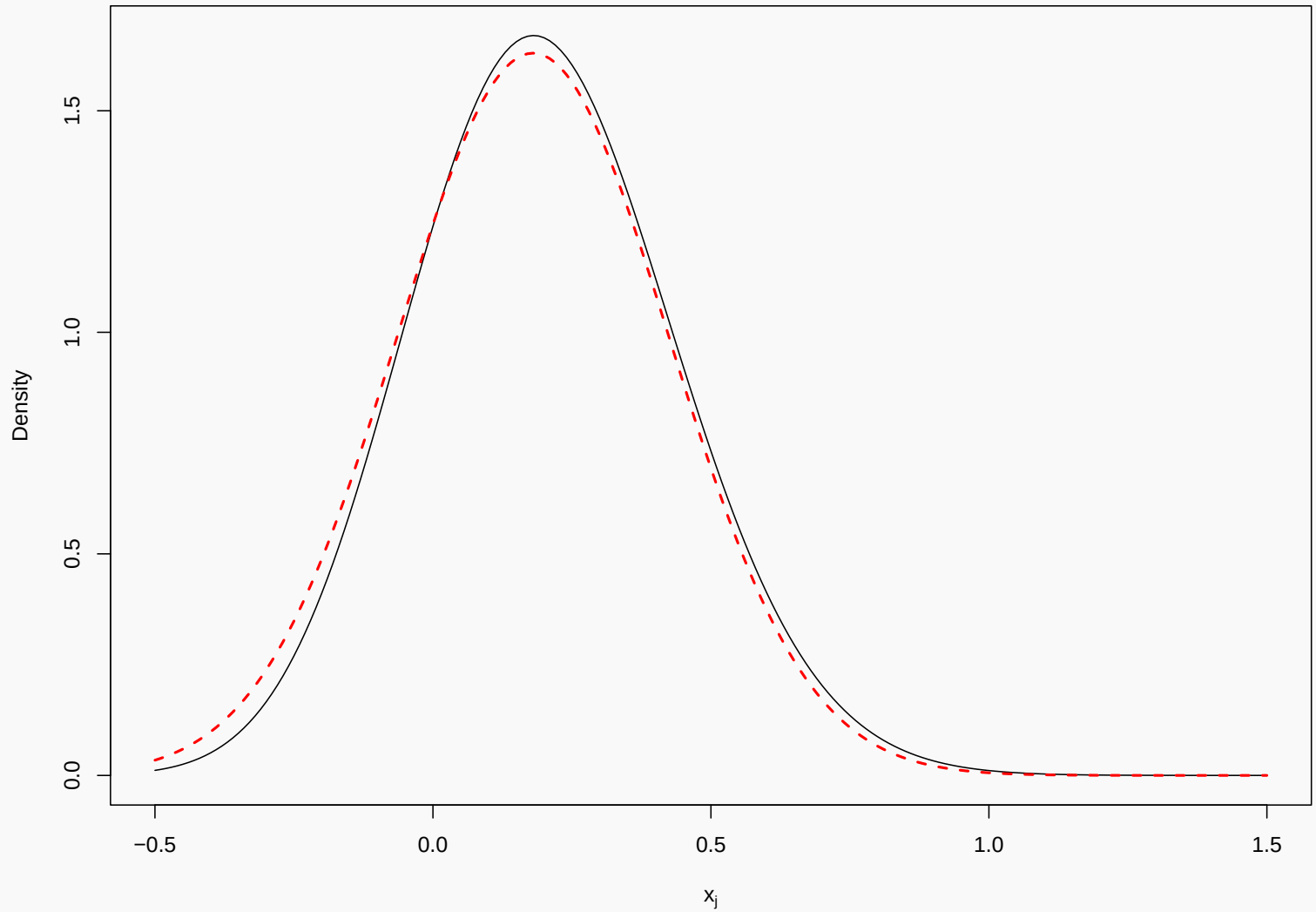
can be computed numerically, and the variance of the normal approximation can be computed by inspecting the curvature at the mode, given by

$$\left[-\frac{\partial^2 \log p(x|y_j, x_{j-1})}{\partial x^2}\right]^{-1} = \left[2y_j^2 e^{-2x} + 1/\sigma^2\right]^{-1}$$

Particle Degeneracy

Example (Stochastic volatility model)

Normal approximation to $p(x_j|y_j, x_{j-1})$ for $x_{j-1} = 0, y_j = 2$:



Particle Degeneracy

The main reason for particle degeneracy is that particles with low weight at step $j-1$ tend to have low weight at step j.

Particle degeneracy can be overcome by *resampling* that removes the particles with low weights. The commonest form of resampling is to use a non-parametric bootstrap, with multinomial sampling.

Particle Degeneracy

At step j, the Monte Carlo estimate of π_j is

$$\widehat{\pi}_j(\mathbf{x}) = \sum_{i=1}^{N} W_j(x_{1:j}^{(i)}) \delta_{x_{1:j}^{(i)}}(\mathbf{x})$$

This discrete distribution on

$$x_{1:j}^{(1)}, \ldots, x_{1:j}^{(N)}$$

is resampled N times according to the weights

$$W_j(x_{1:j}^{(1)}), \ldots, W_j(x_{1:j}^{(N)}).$$

Particle Degeneracy

After resampling, the new resampled values replace the

$$x_{1:j}^{(1)}, \ldots, x_{1:j}^{(N)}$$

collection, and each resampled point carries with it its normalized resampling weight $W_j(x_{1:j}^{(i)})$.

Each original value $x_{1:j}^{(i)}$ may appear in the resampled collection more than once.

Particle Degeneracy

Let $n_{j1}, \ldots, n_{jN}$ denote the numbers of times that each of the N particles are resampled during multinomial sampling. Let

$$\widetilde{\pi}_j(\mathbf{x}) = \sum_{i=1}^{N} \frac{n_{ji}}{N} \delta_{x_{1:j}^{(i)}}(\mathbf{x})$$

Resampling does not disrupt the particle approximation in expectation (that is, it does not introduce bias)

$$\mathbb{E}[\widetilde{\pi}_j(\mathbf{X}) | \widehat{\pi}_j(x_{1:j})] = \widehat{\pi}_j(x_{1:j})$$

but it does introduce "noise" - for any suitable h,

$$\mathrm{Var}_{\widetilde{\pi}_j}[h(x_{1:j})] \geqslant \mathrm{Var}_{\widehat{\pi}_j}[h(x_{1:j})]$$

This follows by the law of iterated variance.

Particle Degeneracy

The message from this result is that although you should perform resampling to preserve particle diversity, you should not resample too often, as that introduces variability.

A common strategy is to resample only when the observed variance of the weights exceeds some threshold. A statistic to track is the *effective sample size* (ESS) where

$$\text{ESS} = \left(\sum_{i=1}^{N} \left\{ W_j^{(i)} \right\}^2 \right)^{-1}$$

We have $1 \leqslant \text{ESS} \leqslant N$:

$$\text{ESS} = 1 \quad \text{if} \quad W_j^{(i)} = 1 \quad \text{for some } i$$

$$\text{ESS} = N \quad \text{if} \quad W_j^{(i)} = \frac{1}{N} \quad \text{for each } i$$

Particle Degeneracy

Can also use *entropy* as a measure

$$\text{Entropy} = -\sum_{i=1}^{N} W_j^{(i)} \log W_j^{(i)}$$

where $0 \leqslant \text{Entropy} \leqslant \log_2 N$

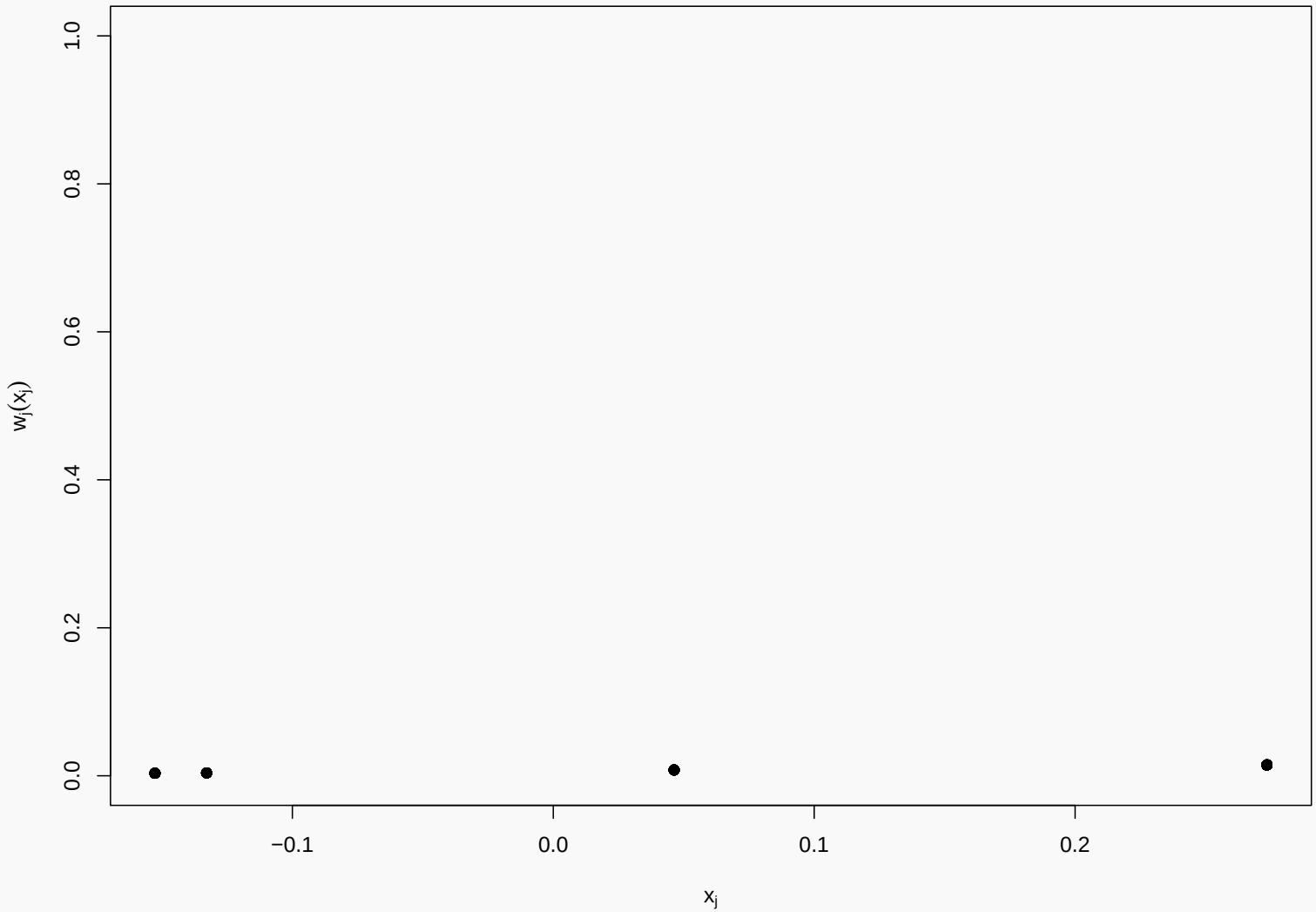
$$\text{Entropy} = 0 \quad \text{if} \quad W_j^{(i)} = 1 \quad \text{for some } i$$

$$\text{Entropy} = \log_2 N \quad \text{if} \quad W_j^{(i)} = \frac{1}{N} \quad \text{for each } i$$

Particle Degeneracy

Example (Stochastic volatility model)

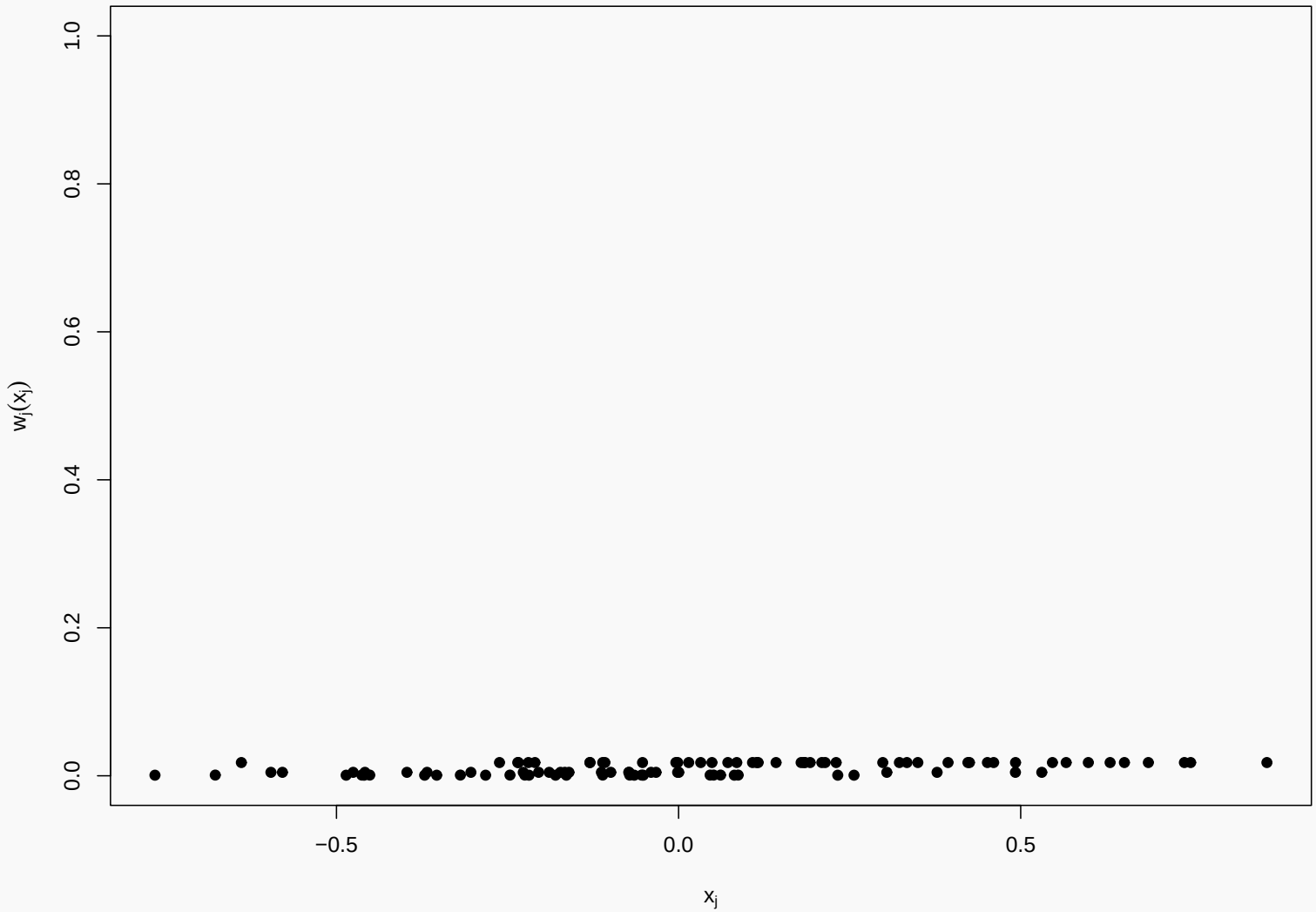
$N = 100$ resampled particles and weights at $j = 1$



Particle Degeneracy

Example (Stochastic volatility model)

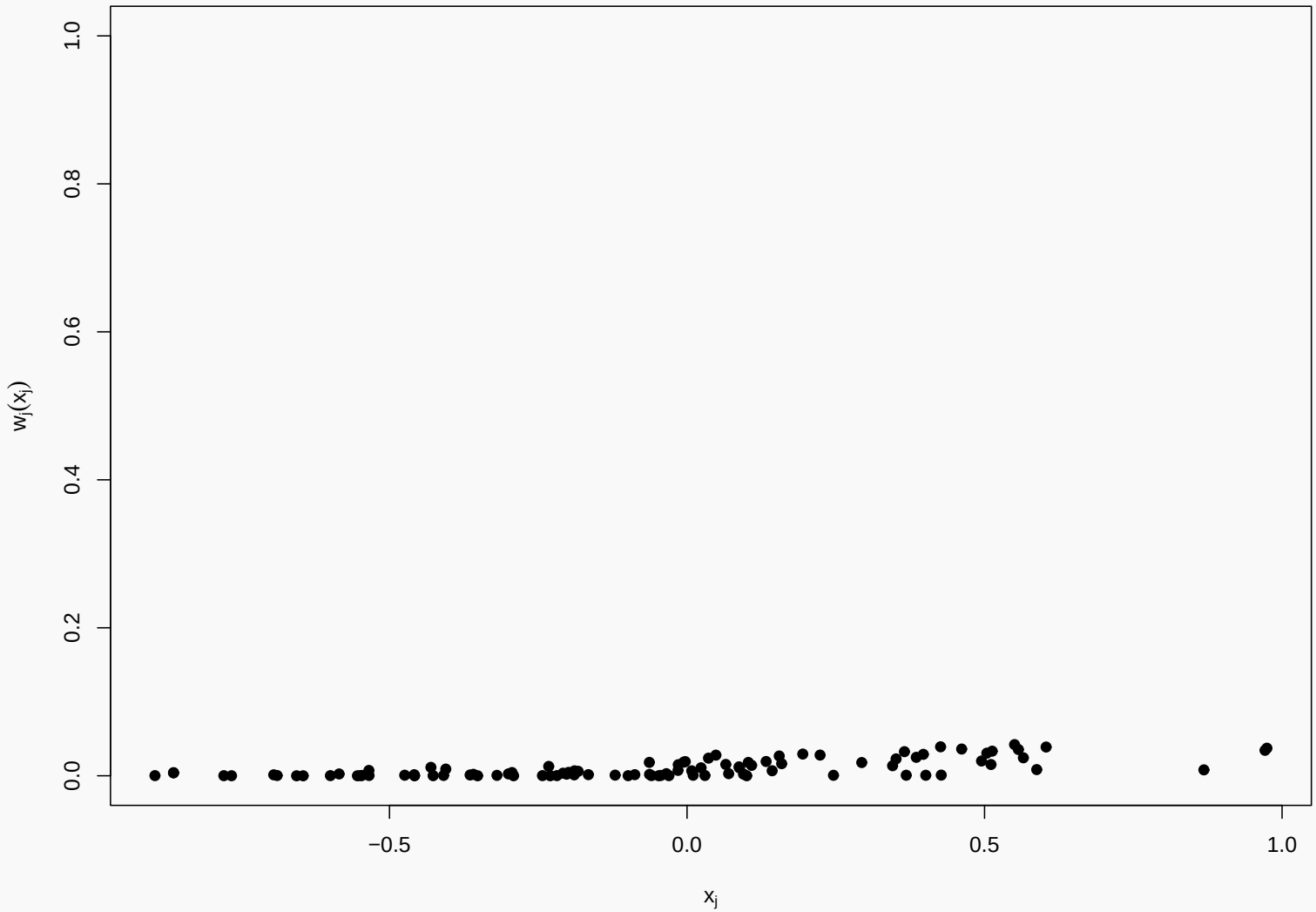
$N = 100$ resampled particles and weights at $j = 2$



Particle Degeneracy

Example (Stochastic volatility model)

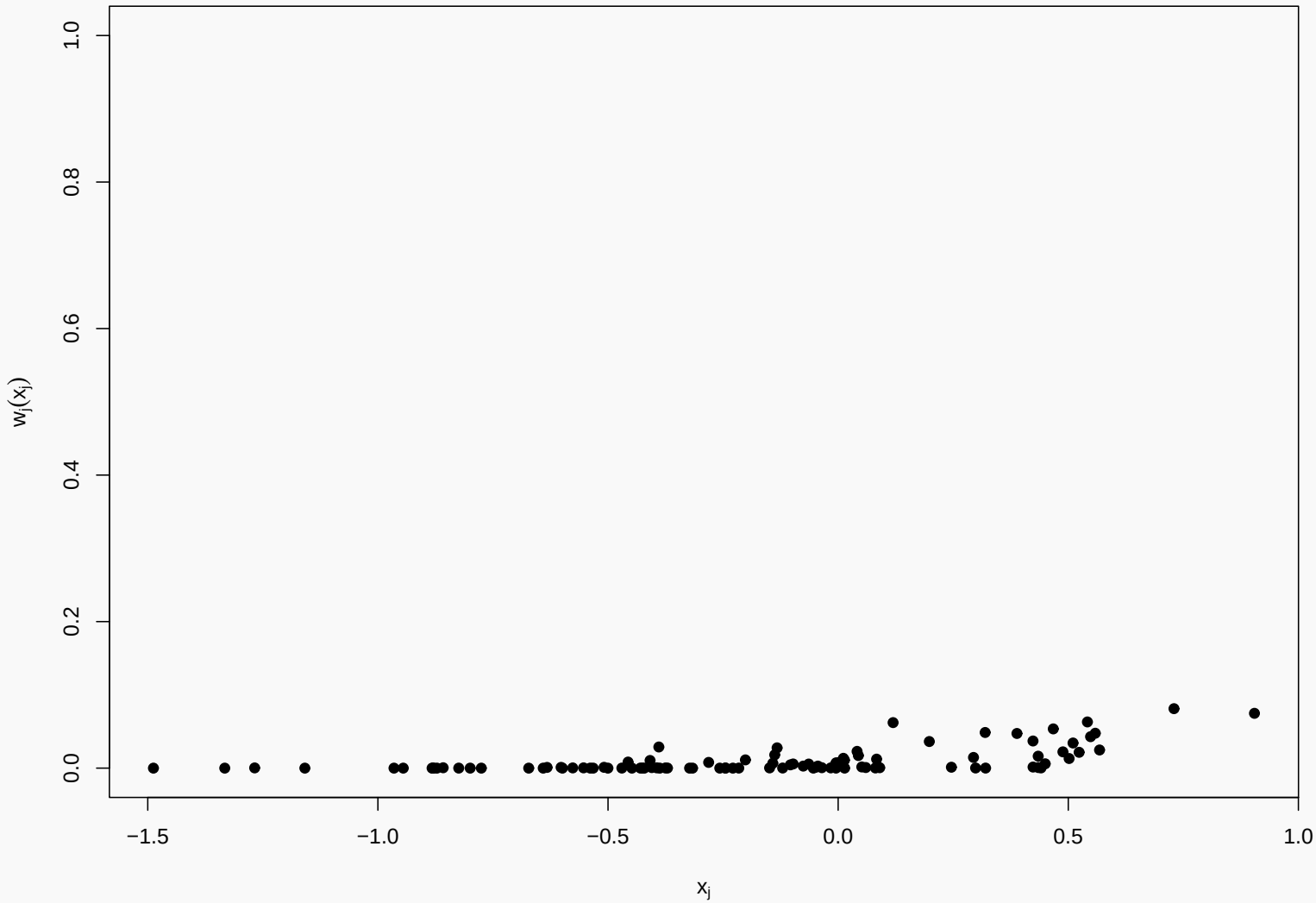
$N = 100$ resampled particles and weights at $j = 3$



Particle Degeneracy

Example (Stochastic volatility model)

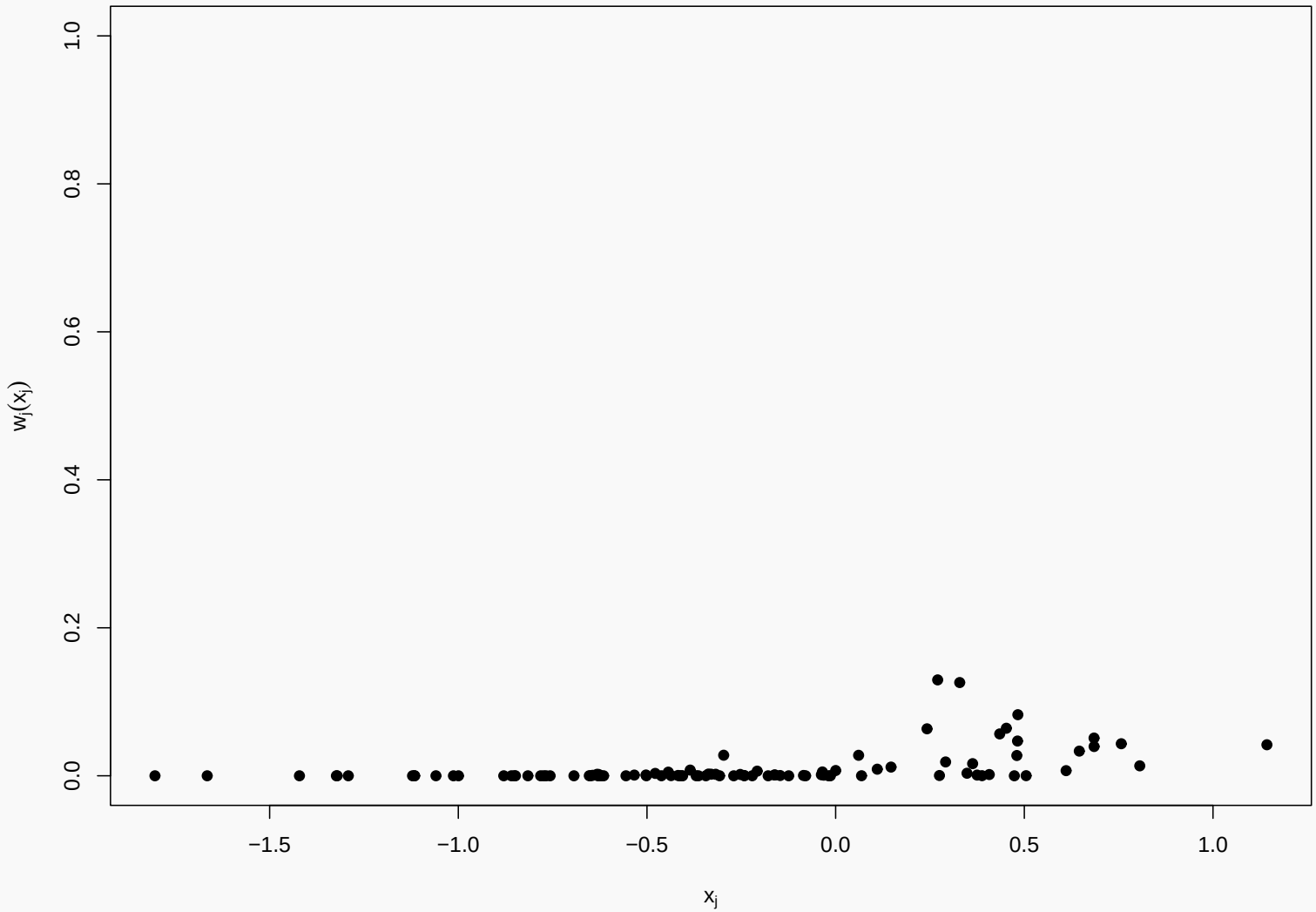
$N = 100$ resampled particles and weights at $j = 4$



Particle Degeneracy

Example (Stochastic volatility model)

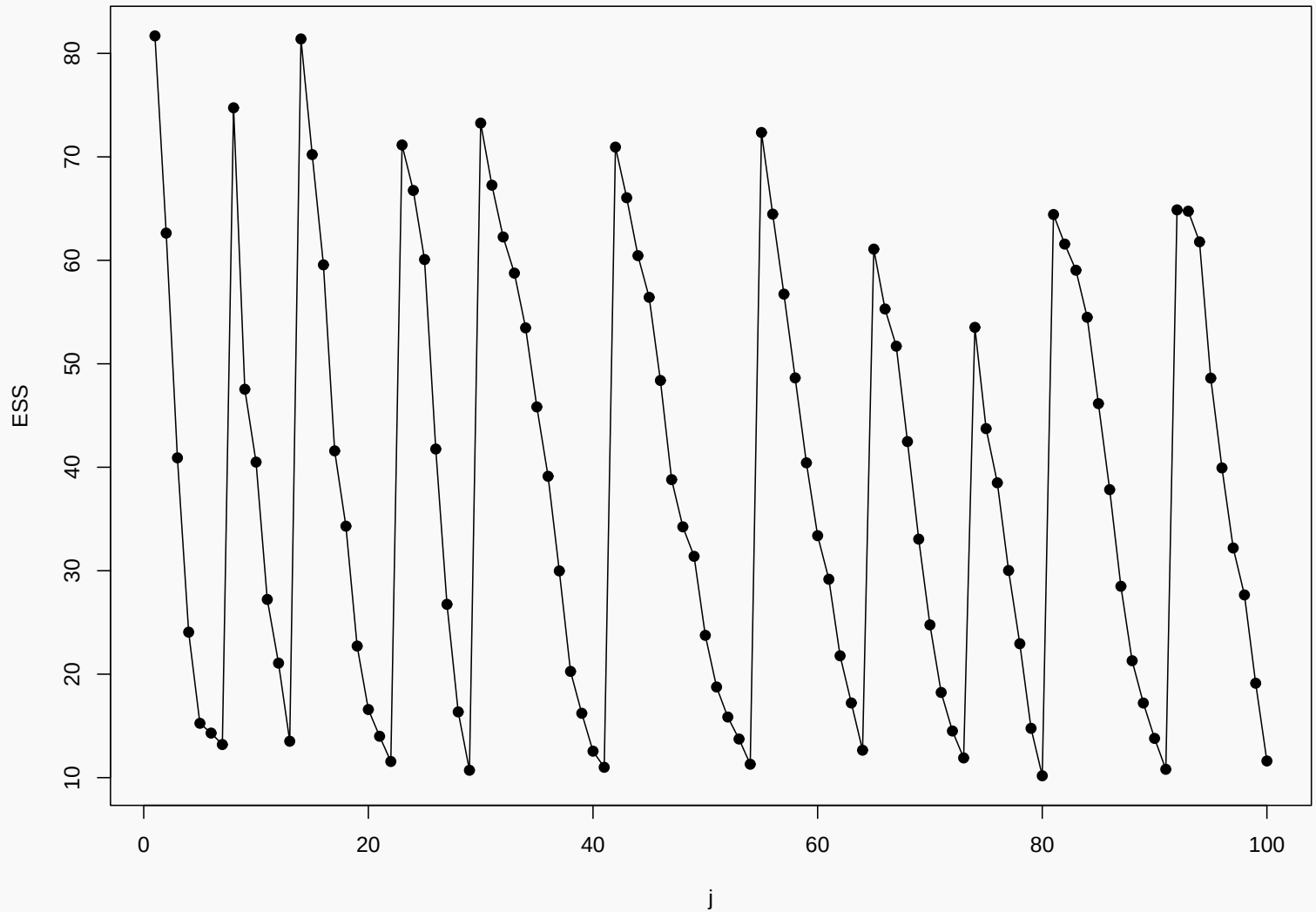
$N = 100$ resampled particles and weights at $j = 5$



Particle Degeneracy

Example (Stochastic volatility model)

Effective sample size (ESS) with resampling when ESS < 10



Appendix: Multiple chain Moves

Back to slides

Multiple chain MCMC: Types of move

1. **Mutation:** Select $m \in \{1, \ldots, M\}$, and perform a Metropolis-Hastings move using proposal density $q_m(x, z)$. If the current value for x_m is x, and the proposed value is z, accept the proposed point with probability

$$\alpha = \min \left\{ 1, \frac{\pi_m(z) q_m(z, x)}{\pi_m(x) q_m(x, z)} \right\}$$

   In the context of the vector $\boldsymbol{x}$, this move is essentially a Gibbs sampler move, as the full conditional for X_m is π_m.

▸ Back to slides

Multiple chain MCMC: Types of move

2. **Exchange:** Select $l, m \in \{1, \ldots, M\}$, and propose the swap

$$x_l \longleftrightarrow x_m$$

Accept this proposal with probability

$$\alpha = \min\left\{1, \frac{\pi_l(x_m)\pi_m(x_l)}{\pi_l(x_l)\pi_m(x_m)}\right\}.$$

Back to slides

Multiple chain MCMC: Types of move

In the case where all the π_m are equal to the target π, we can utilize the following move:

3. **Snooker:** Select $l, m \in \{1, \ldots, M\}$, and, in an attempt to update x_l, make the proposal z

$$z = x_m + u(x_l - x_m)$$

where u is a random variate sampled from the density

$$f(u) \propto \pi(x_m + u(x_l - x_m))$$

and then update x_l to z.

▸ Back to slides

Multiple chain MCMC: Types of move

This move can be utilized in the case of **vector** state spaces. If each x is d-dimensional, then the move becomes

3*. **Snooker:** Select $l, m \in \{1, \ldots, M\}$, and, in an attempt to update x_l, make the proposal z

$$z = x_m + u(x_l - x_m)$$

where u is a scalar random variate sampled from the density

$$f(u) \propto |u|^{d-1} \pi(x_m + u(x_l - x_m))$$

and then update x_l to z.

▸ Back to slides

Multiple chain MCMC: Types of move

A related move returns to basic ideas from optimization:

4. **Local optimum:** Select $l, m \in \{1, \ldots, M\}$, and, in an attempt to update x_l, proceed as follows: starting at x_m, use conjugate gradient or steepest descent to find the local mode of π, and denote this $\widehat{x}_m$.

   Then make the proposal z

   $$z = \widehat{x}_m + u(x_l - \widehat{x}_m)$$

   where u is a scalar random variate sampled from the density

   $$f(u) \propto |u|^{d-1} \pi(\widehat{x}_m + u(x_l - \widehat{x}_m))$$

   and then update x_l to z.

Back to slides

Multiple chain MCMC: Types of move

Again in the d-vector case, we have the final move type, inspired by ideas from genetic algorithms:

5. **Crossover:** Select $l, m \in \{1, \ldots, M\}$, and then select j uniformly from $\{1, \ldots, d-1\}$. Perform a crossover at position j, that is, attempt to replace x_l and x_m by

$$x_l^{\text{new}} = (x_{l1}, \ldots, x_{lj}, x_{mj+1}, \ldots, x_{md})$$

$$x_m^{\text{new}} = (x_{m1}, \ldots, x_{mj}, x_{lj+1}, \ldots, x_{ld})$$

that is, exchange the $(j+1, \ldots, d)$ portions of each vector.

Accept this proposal with probability

$$\alpha = \min\left\{1, \frac{\pi_l(x_l^{\text{new}})\pi_m(x_m^{\text{new}})}{\pi_l(x_l)\pi_m(x_m)}\right\}.$$

Back to slides