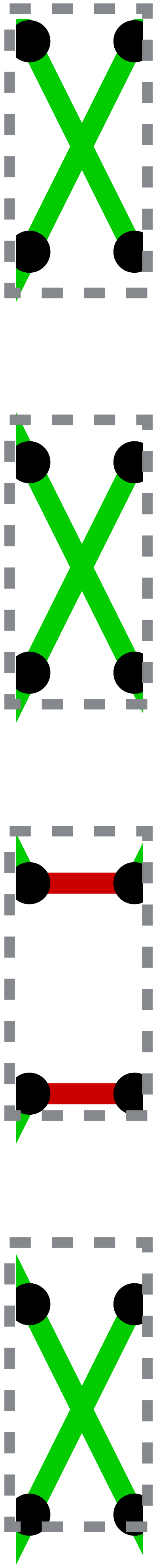
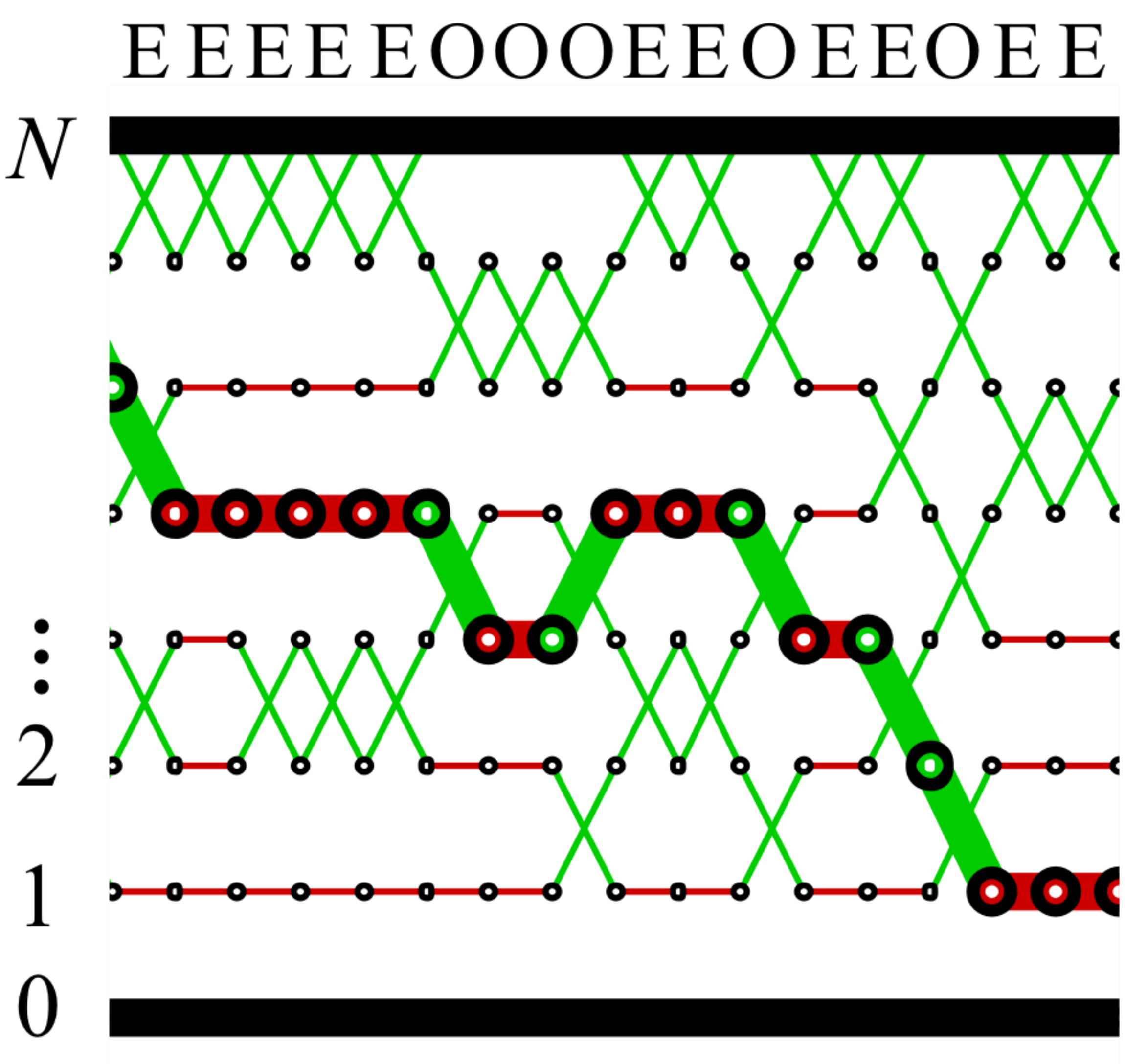


Reversible vs. non-reversible parallel tempering

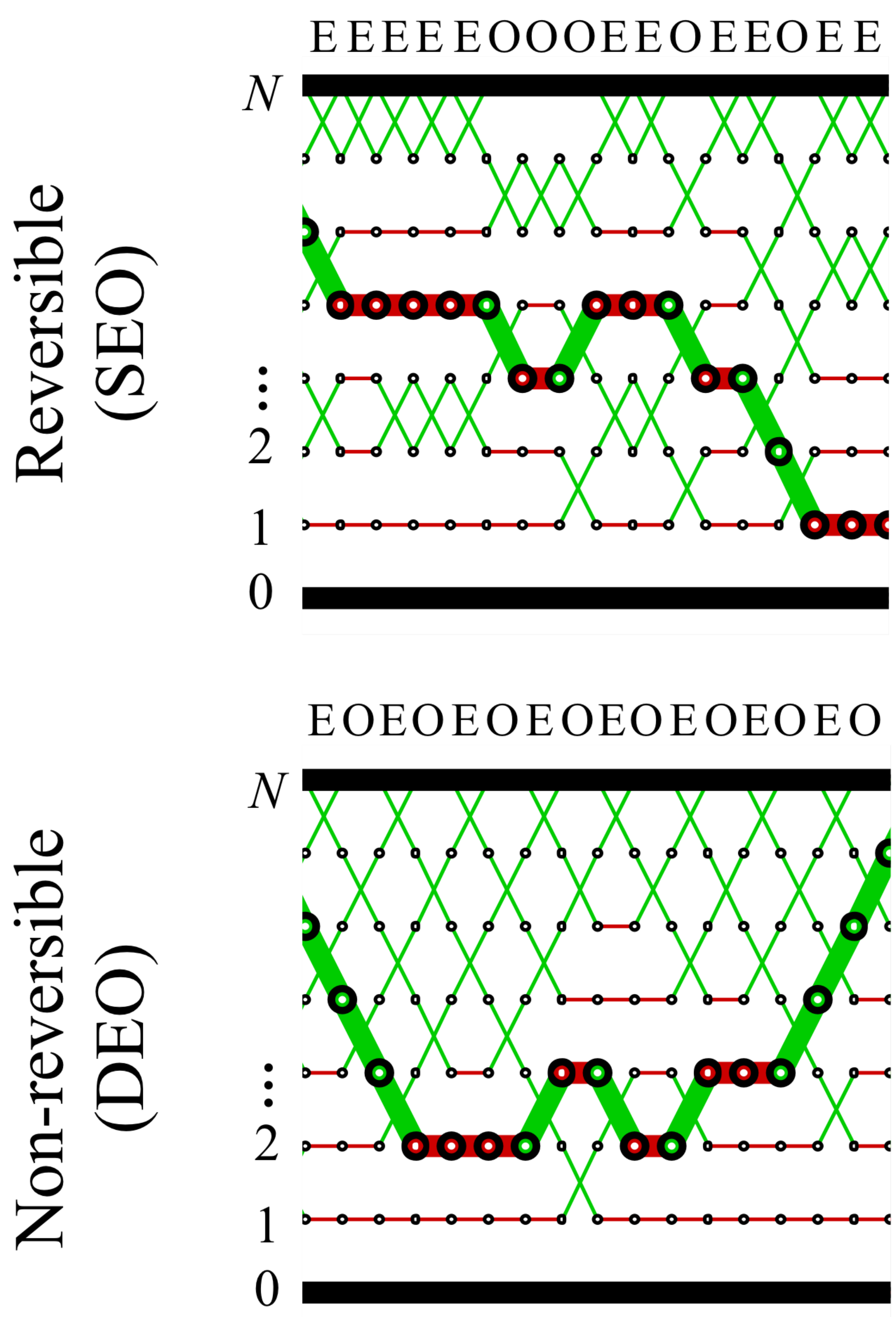
- In a given communication round, attempt to swap as many consecutive disjoint pairs of chains as possible, in parallel
- Denote swap indices by $(i, i+1)$

Reversible vs. non-reversible parallel tempering



- Denote swap indices by $(i, i+1)$
- Even swaps (E)
  - pick swap pairs such that i is <u>even</u>
- Odd swaps (O)
  - pick swap pairs such that i is <u>odd</u>

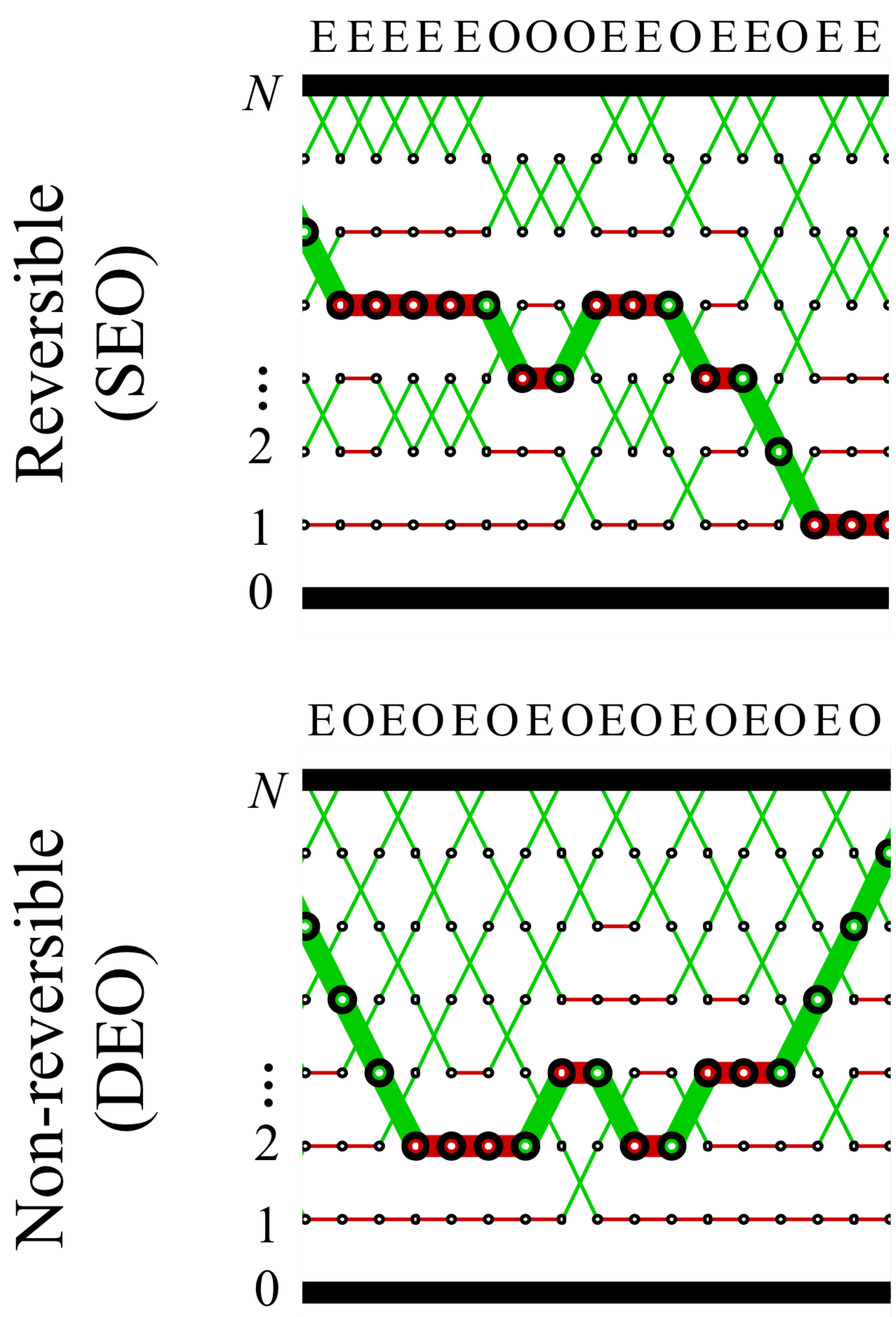
Reversible vs. non-reversible parallel tempering (PT)



- At each communication iteration, with:
- ... reversible PT:
  - Stochastically pick **E** or **O** using coin flip
- ... non-reversible PT
  - Deterministically alternates between **E** and **O**

DEO: Okabe et al. 2001

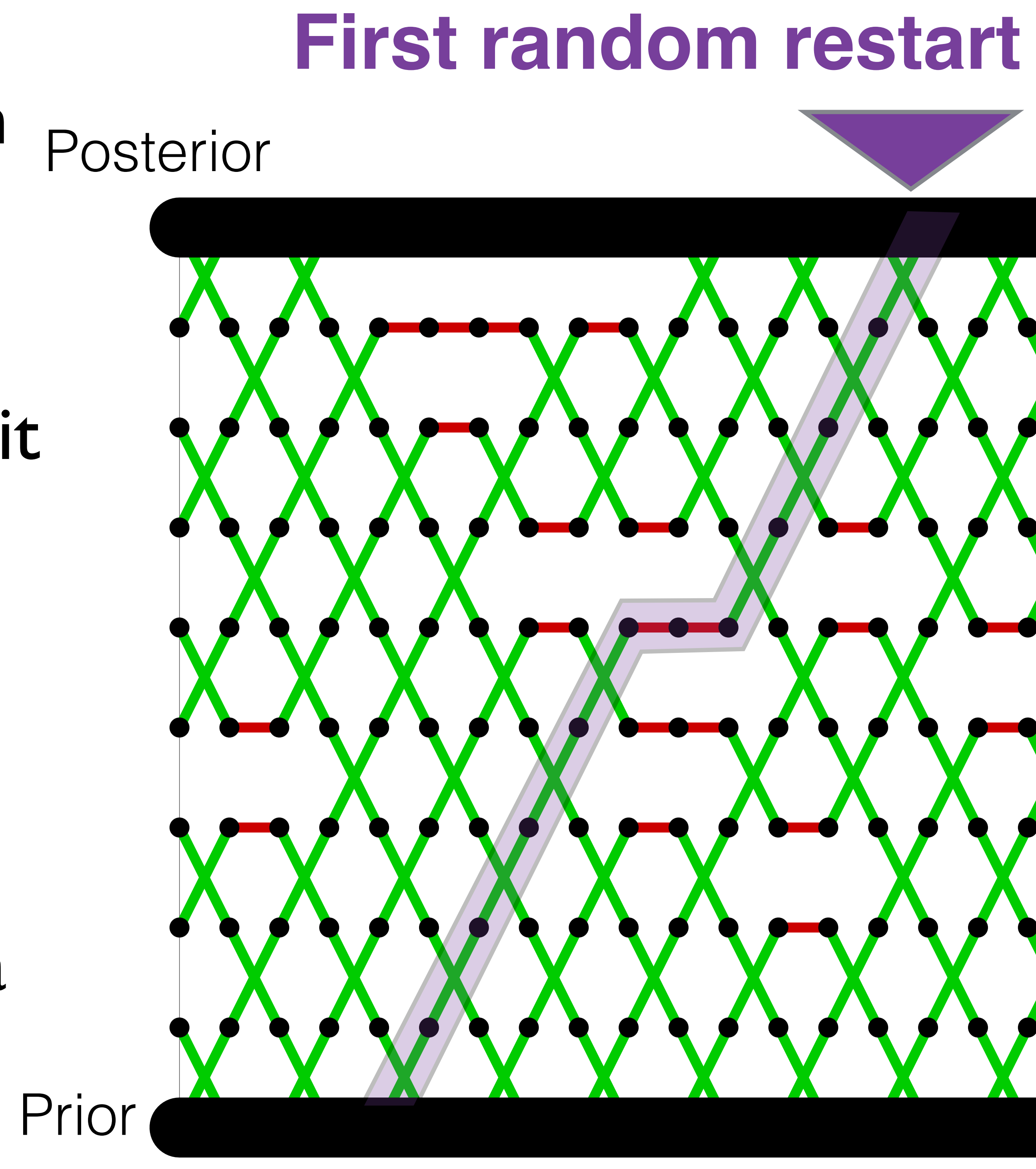
Reversible vs. non-reversible parallel tempering (PT)



Surprise: this seemingly minor detail has a profound impact on the behaviour of the PT algorithm

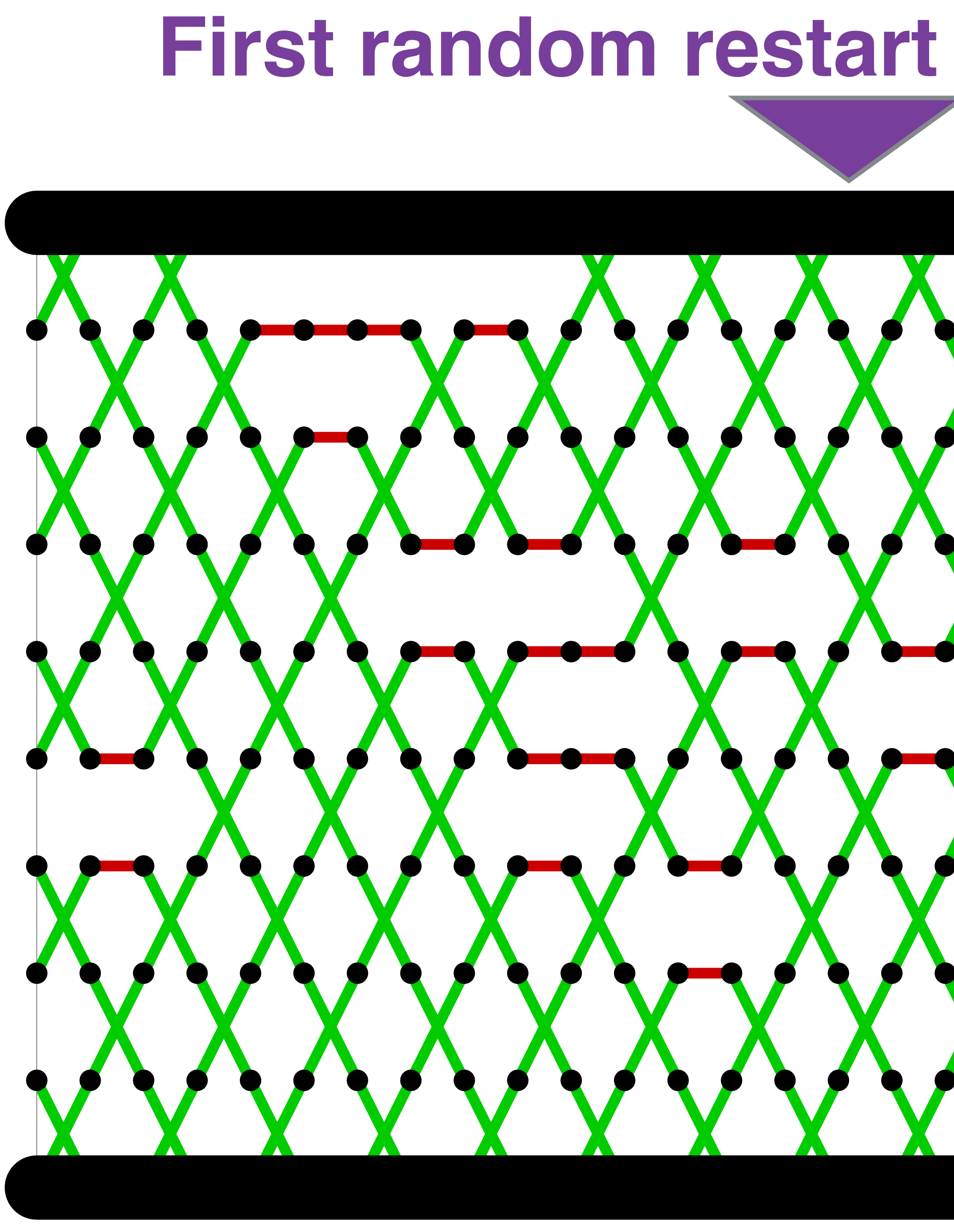
Notion of performance for Parallel tempering

- Setup we are interested in:
  - Distributed computing: we can summon as many machines as we want
  - Prior ($\beta = 0$) is special in that it gives us iid samples
- Imagine the prior sample a new colour at each iteration
- *Random restart*: an iteration where the posterior chain sees a new colour



Restart rate

- *Random restart*: an iteration where the posterior chain sees a new colour
- *Restart rate τ:* fraction of MCMC iterations that are random restarts



Non-asymptotic result

- Non-reversible PT's restart rate dominates reversible PT's

$$\tau_{N,\text{non-reversible}} > \tau_{N,\text{reversible}}, \quad \text{for all } N > 1$$

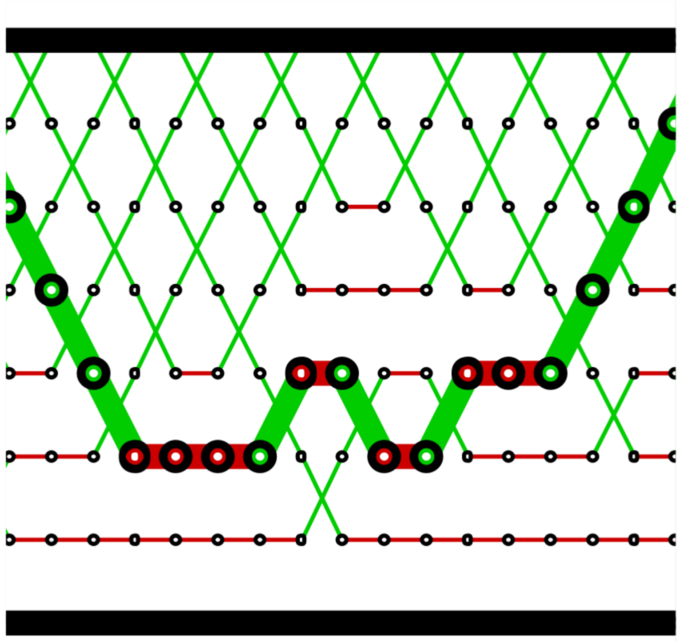
- For a number of chains *N* large enough...
  - $\tau_{N,\text{non-reversible}}$ in an increasing function of *N*
  - $\tau_{N,\text{reversible}} \to 0$

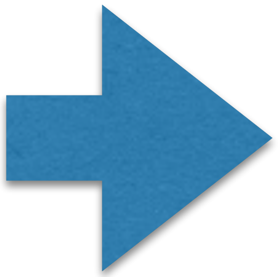
Asymptotic results

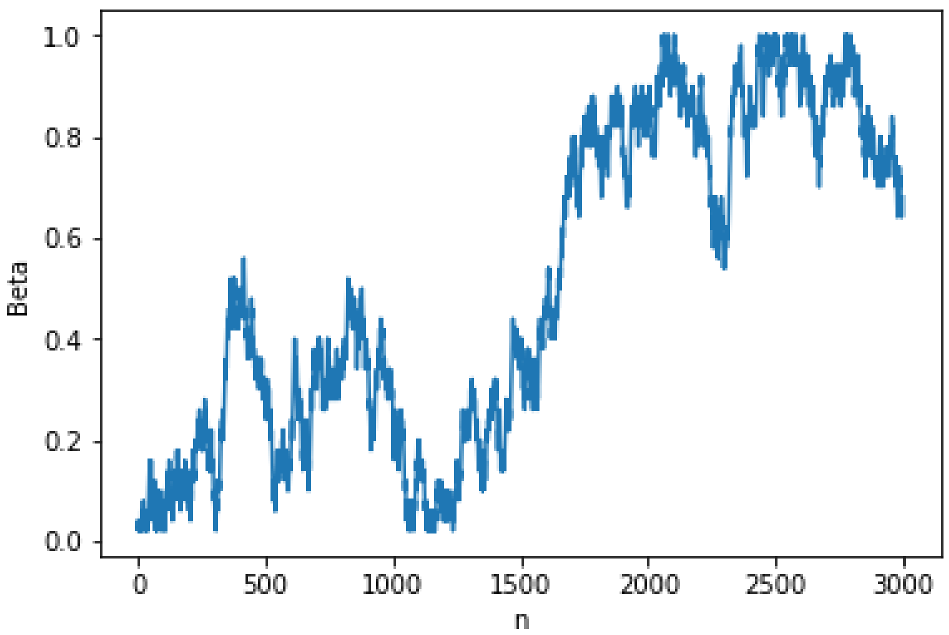
- Typical asymptotic regimes:
  - let the number of data points go to infinity (*n*), or..
  - the number of parameters (*d*)
  - the running time (e.g. # Monte Carlo iterations)
- A road less travelled (in MCMC at least):
  - let the number of cores available go to infinity!
    - Arguably more relevant nowadays than letting time go to infinity...

Asymptotics: setup

- When a swap is accepted, the 2 machines swap annealing parameter, **not** states
- Focus on marginal behaviour of the sequence of annealing parameters assigned to one machine (bold green line on the left)

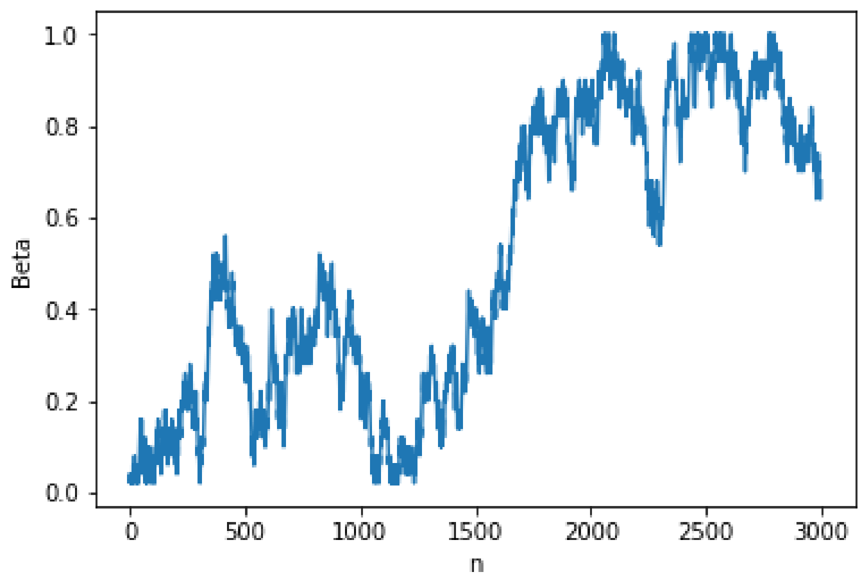






As the number of parallel chains go to infinity...

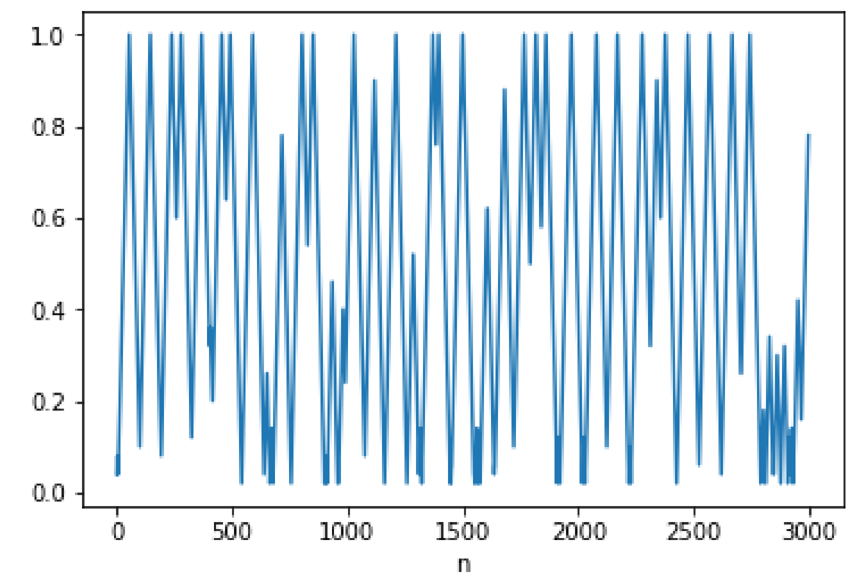
Reversible PT



Weak limit: diffusion

Time rescaled as O(N^2)

Non-reversible PT

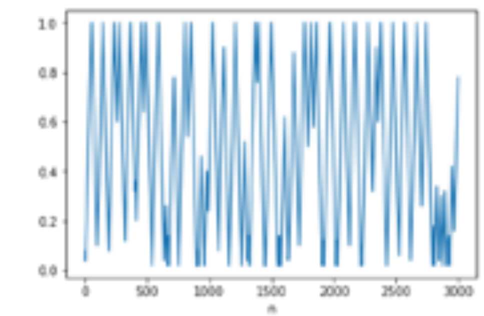


Weak limit: *Piecewise Deterministic Markov Process (**PDMP**)*

Time rescaled as O(N)

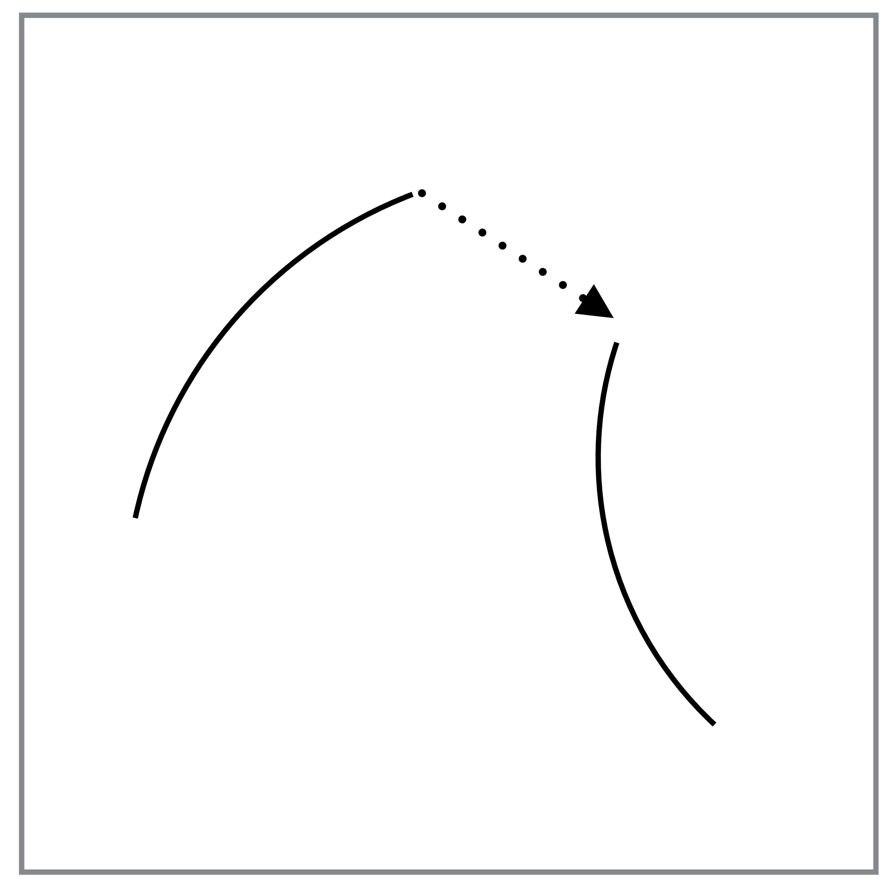
Piecewise Deterministic Markov Processes (PDMPs)

Non-reversible PT



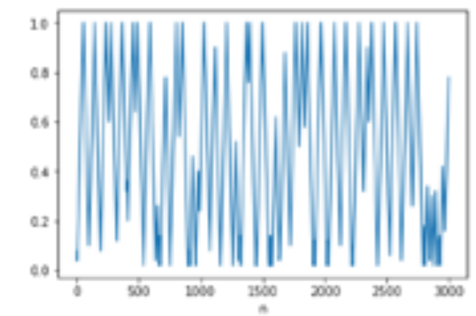
Weak limit: *Piecewise Deterministic Markov Process* (***PDMP***)

- Deterministic flow $\Phi_t(z)$
- Random jumps:
    - rate $\lambda(z)$
    - transition $Q(dz'|z)$



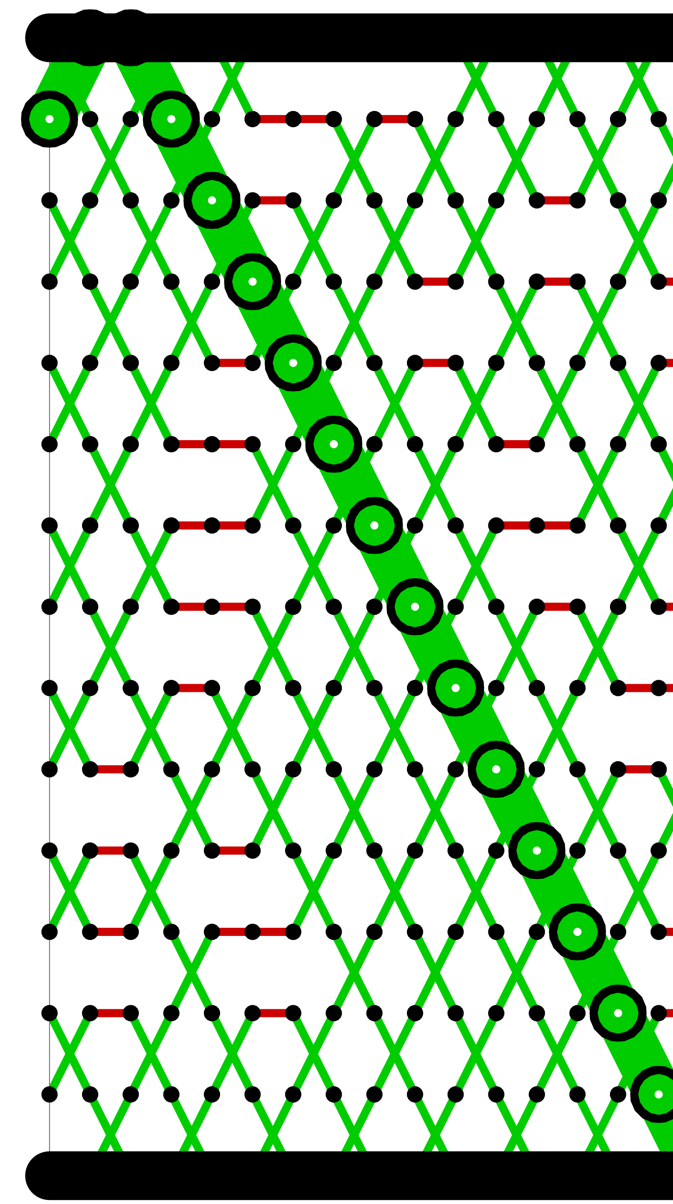
Intuition

Non-reversible PT

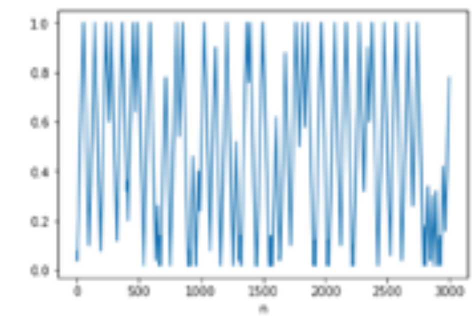


Weak limit: *Piecewise Deterministic Markov Process* (***PDMP***)

- From the deterministic alternation emerges a **persistence of motion** or inertia
  - this is true as long as a proposed swap is not rejected
  - rejection probability for one swap goes to 1 as the number of chains goes to infinity

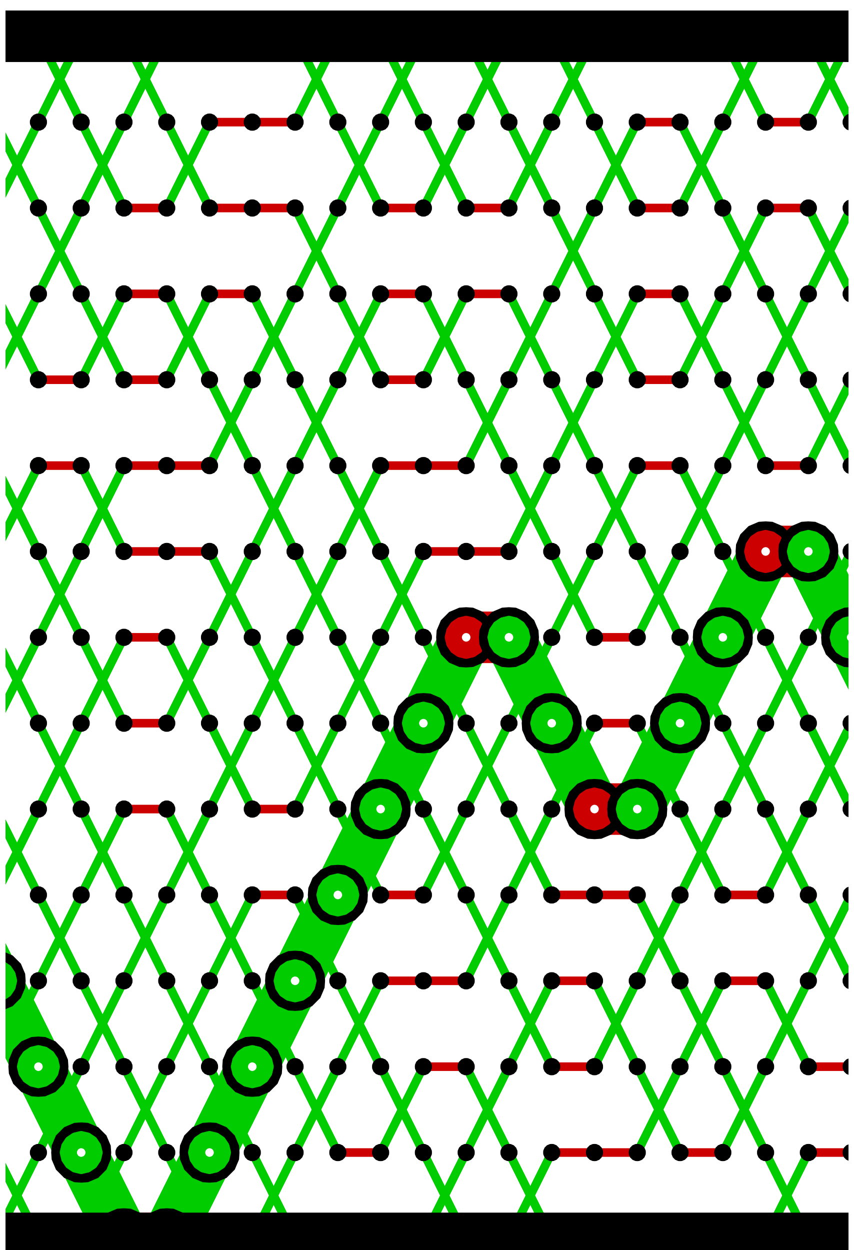


Non-reversible PT



Weak limit: *Piecewise Deterministic Markov Process* (***PDMP***)

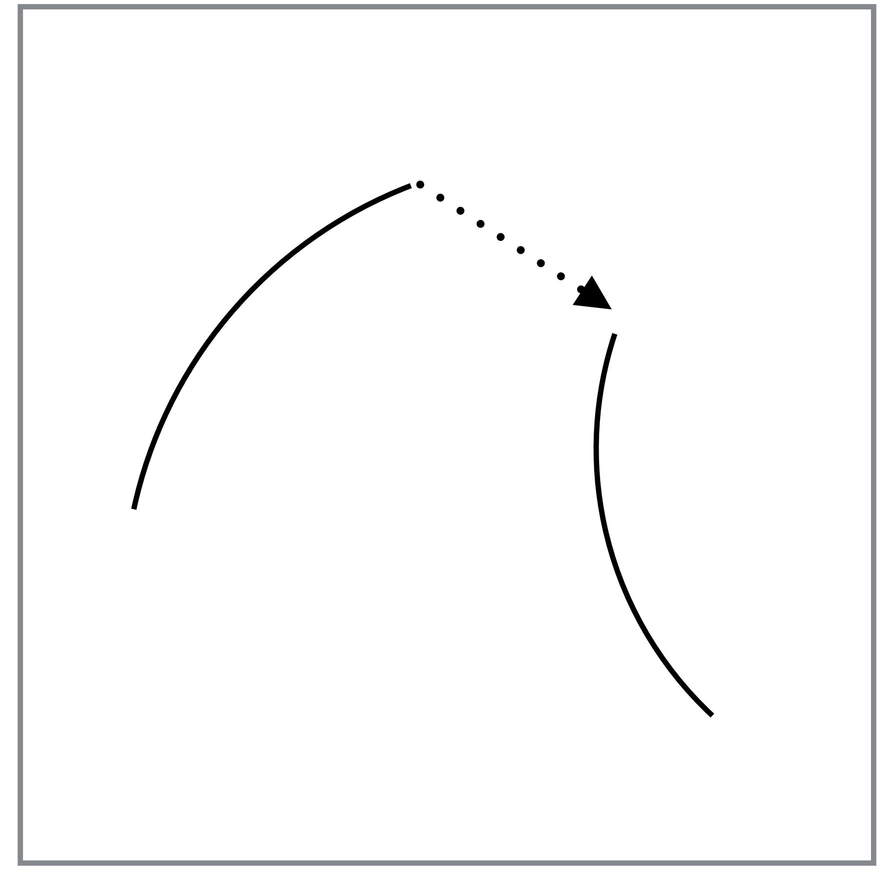
Intuition

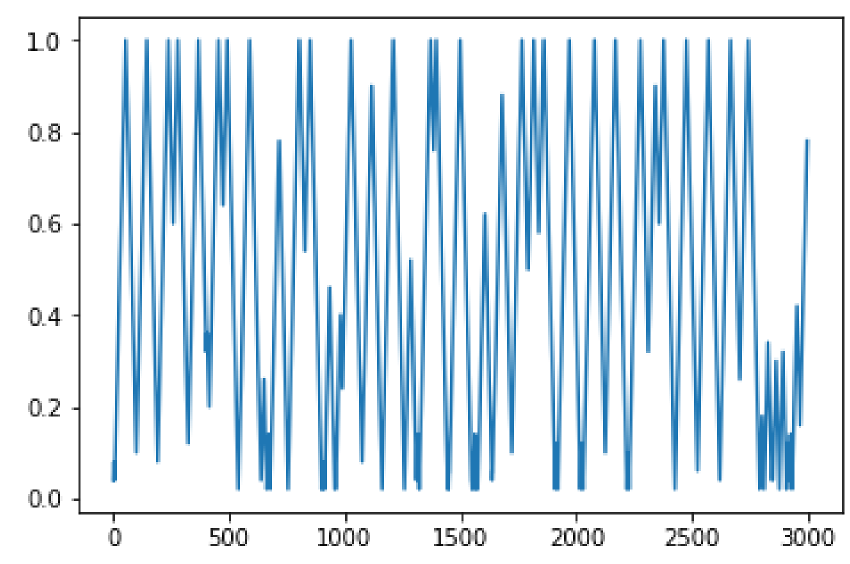


- But as soon as there is a rejected swap, direction changes
  - 1-dimensional “**bounce**”

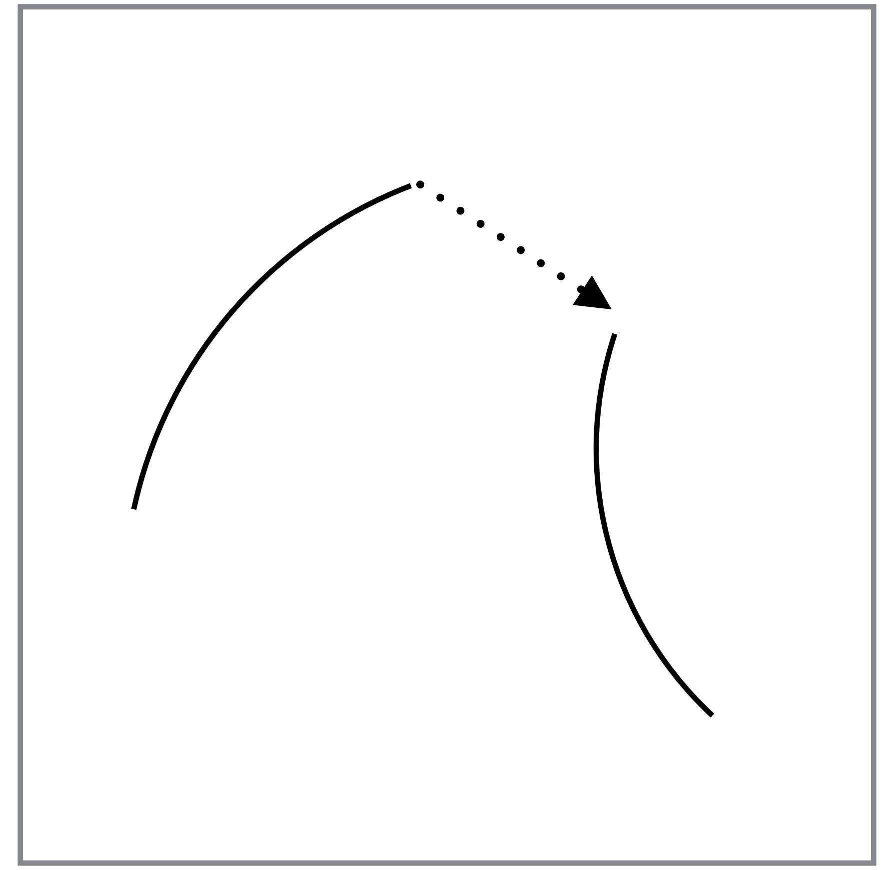
Intuition: limiting PDMPs

- State space:
  - z = (β,ε)
  - β: annealing parameter (in [0, 1])
  - ε: velocity (in {-1,+1})

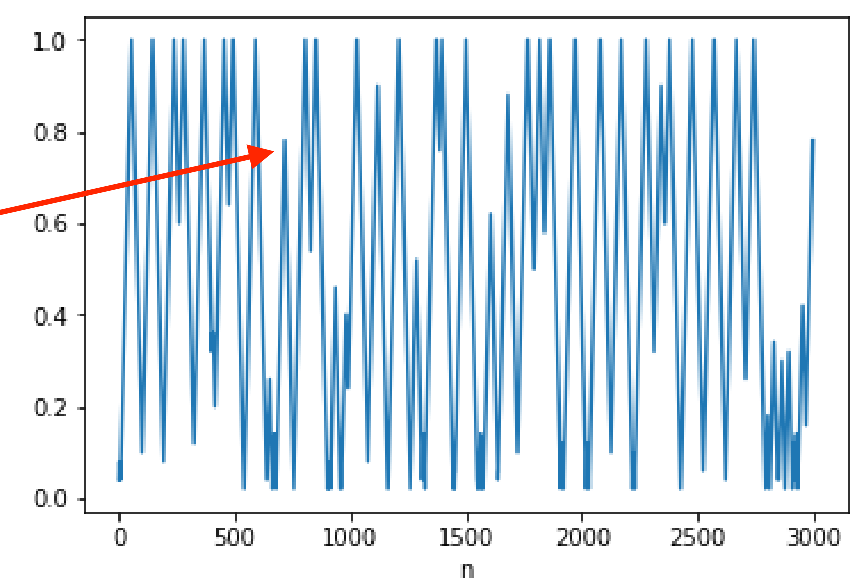




Intuition: limiting PDMPs

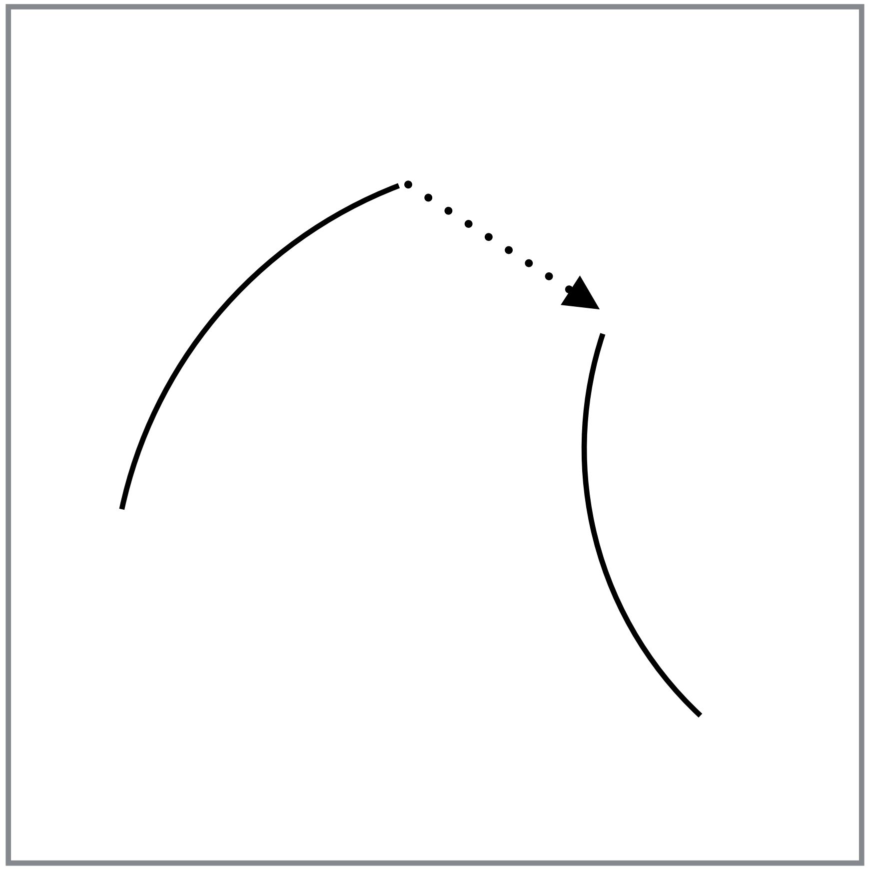


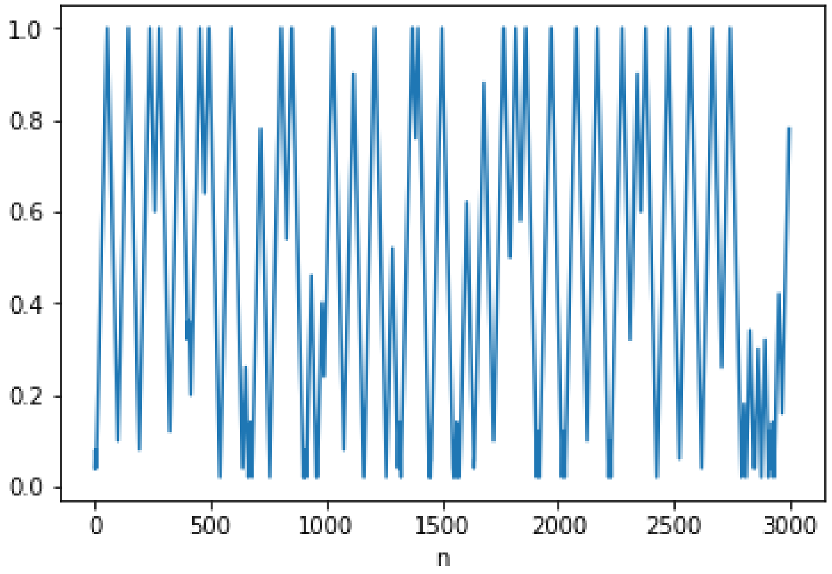
- Deterministic flow $\Phi_t(z)$
- Random jumps:
  - rate $\lambda(z)$
  - transition $Q(dz'|z)$
    - in our PT context, Q is a 1-dimensional "bounce" $(\beta, \varepsilon) \rightarrow (\beta, -\varepsilon)$



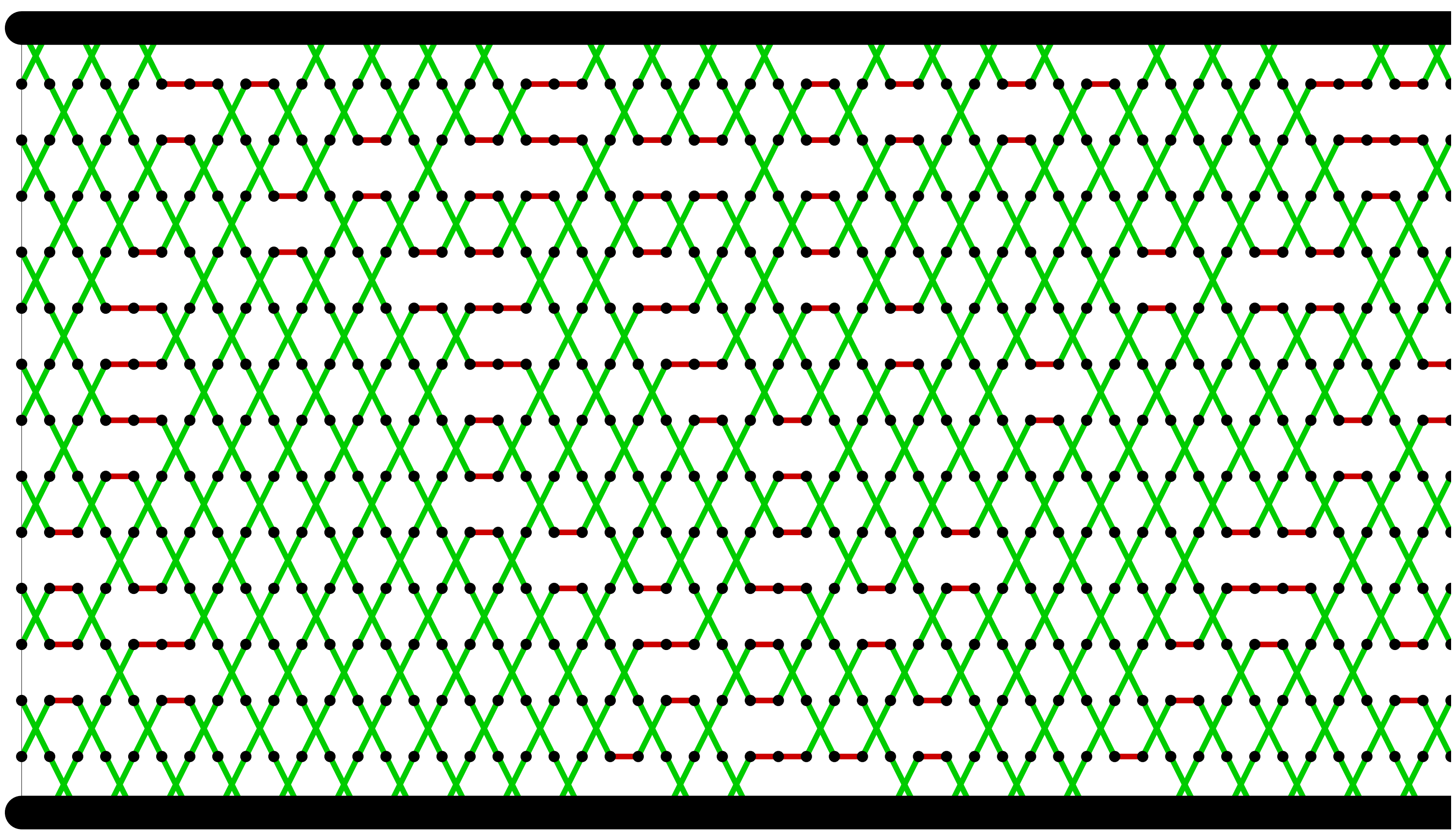
PDMPs

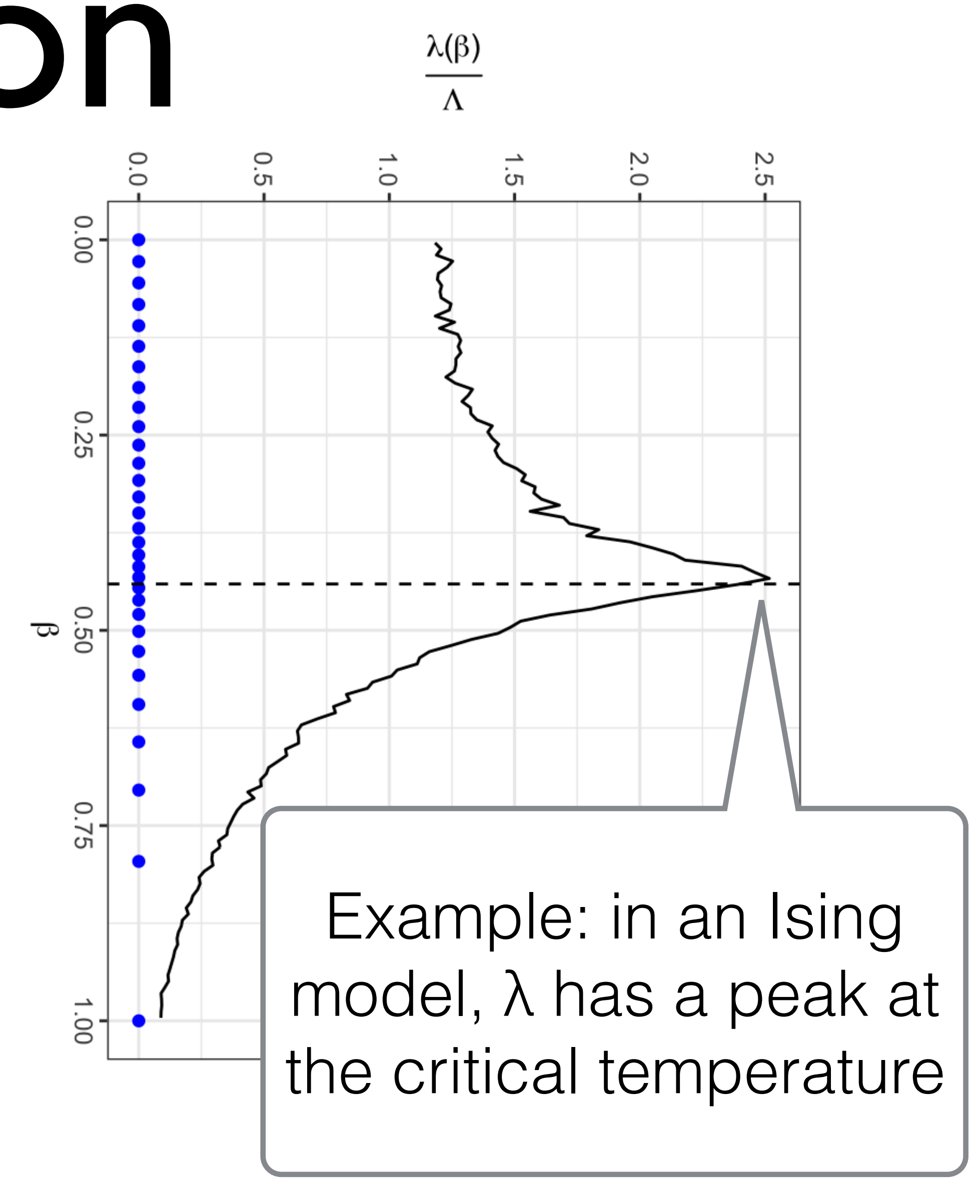
- Deterministic flow $\Phi_t(z)$
- Random jumps:
  - rate $\lambda(z)$
    - what is the bounce rate λ?
  - transition $Q(dz'|z)$





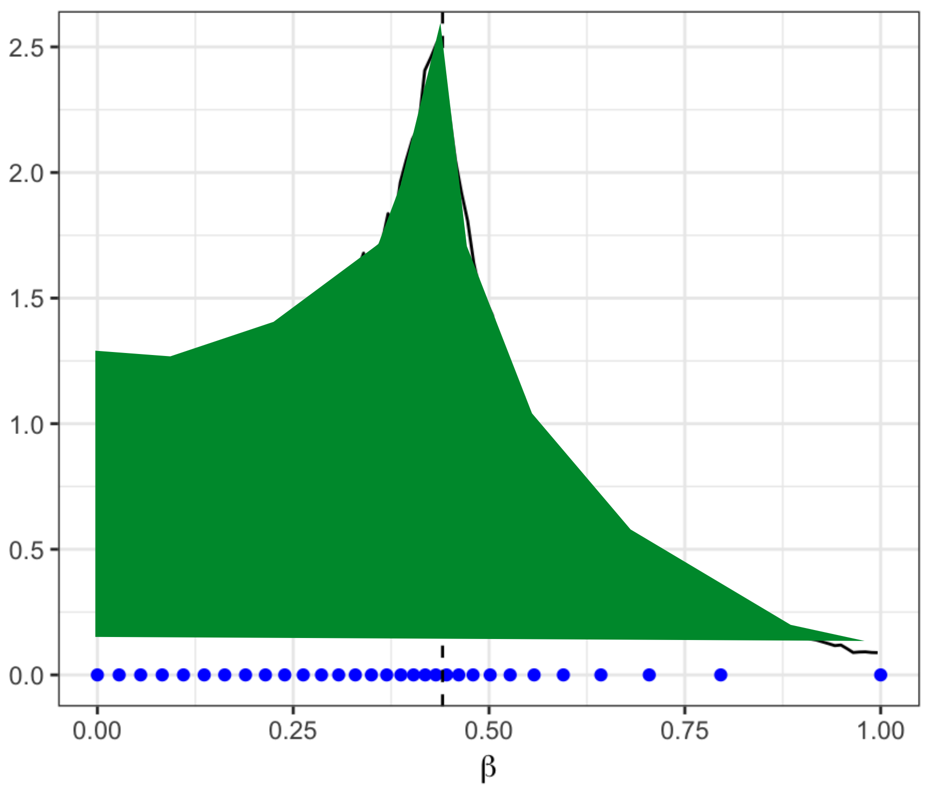
Bounce rate: interpretation





- If we normalize λ by $\Lambda = \int_0^1 \lambda(\beta) \mathrm{d}\beta$
- we obtain a distribution over the annealing parameters where swaps are rejected

Bounce rate: interpretation



- Interpretation of the normalization:

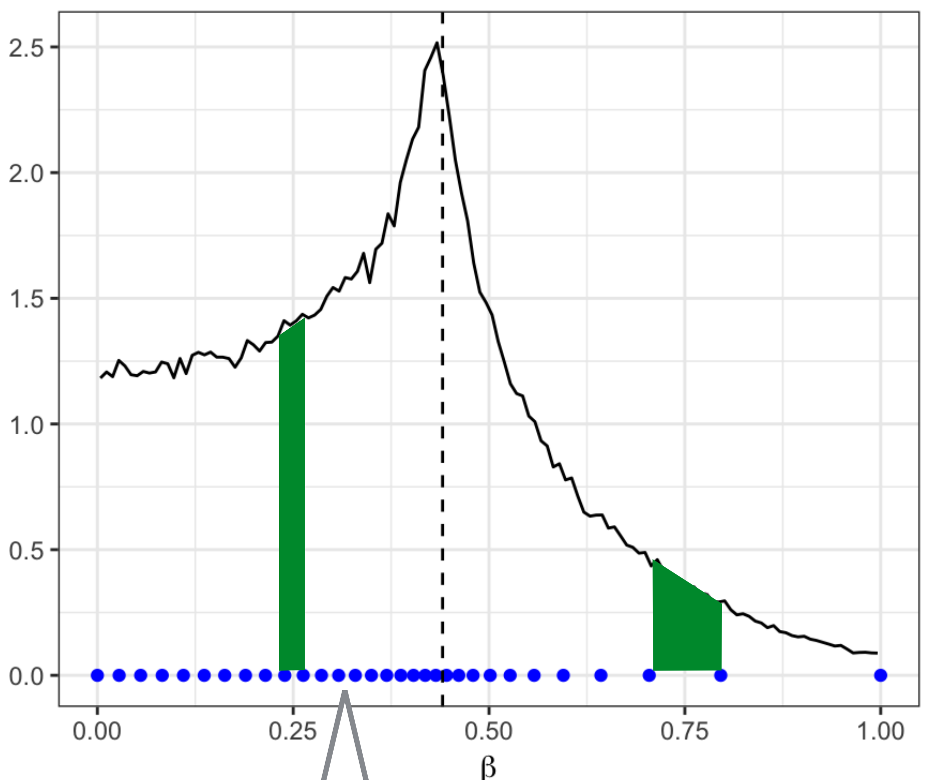
$$\tau_{N,\text{non-reversible}} \to \bar{\tau} = \frac{1}{2 + 2\Lambda}$$

$$\Lambda = \int_0^1 \lambda(\beta) \mathrm{d}\beta$$

Bounce rate: practical use

- The rate λ can be estimated from samples and used to adaptively select the *annealing schedule* $0 \leq \beta_1 \leq \beta_2 \ldots \leq \beta_N = 1$
  - Without such adaptation parallel tempering is not practical
  - previous adaptation scheme are based on stochastic optimization and empirically slower to converge and less robust

Bounce rate: practical use



Blue dots: optimal schedule

Green: area under the curve λ

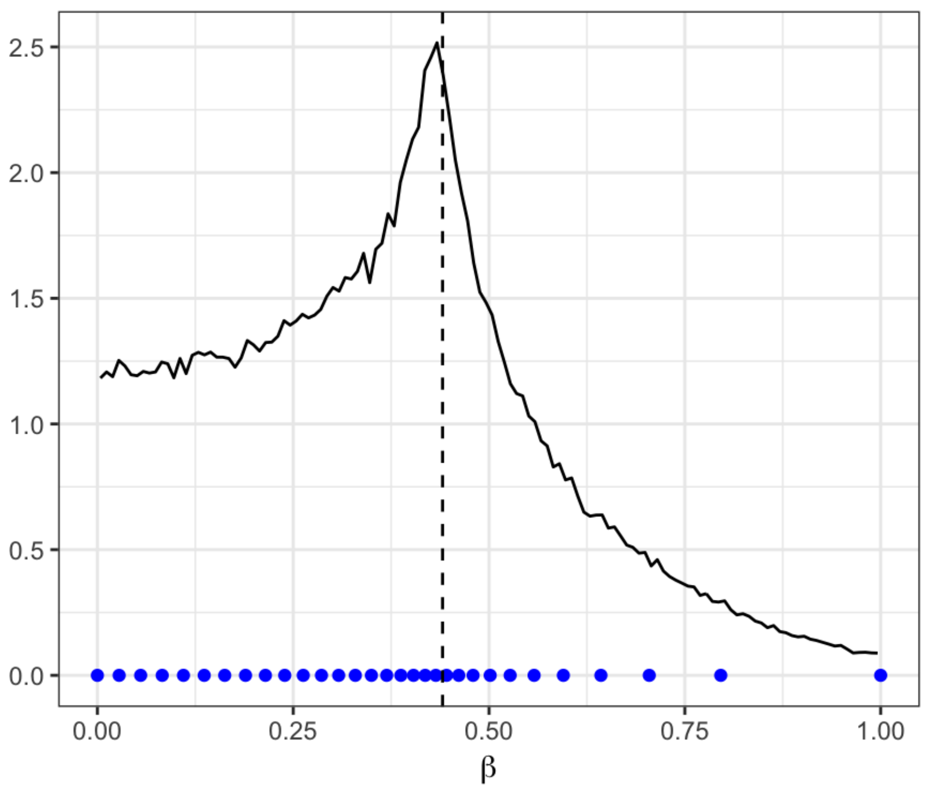
- New proof of classical result: it is optimal to use a *schedule* $0 \leq \beta_1 \leq \beta_2 \ldots \leq \beta_N = 1$ such that the acceptance rate is the same for all chains

- New method to achieve this:

  Loop:

  1. run PT and estimate λ
  2. pick schedule so that area under the curve λ between chains is constant

Bounce rate: estimation

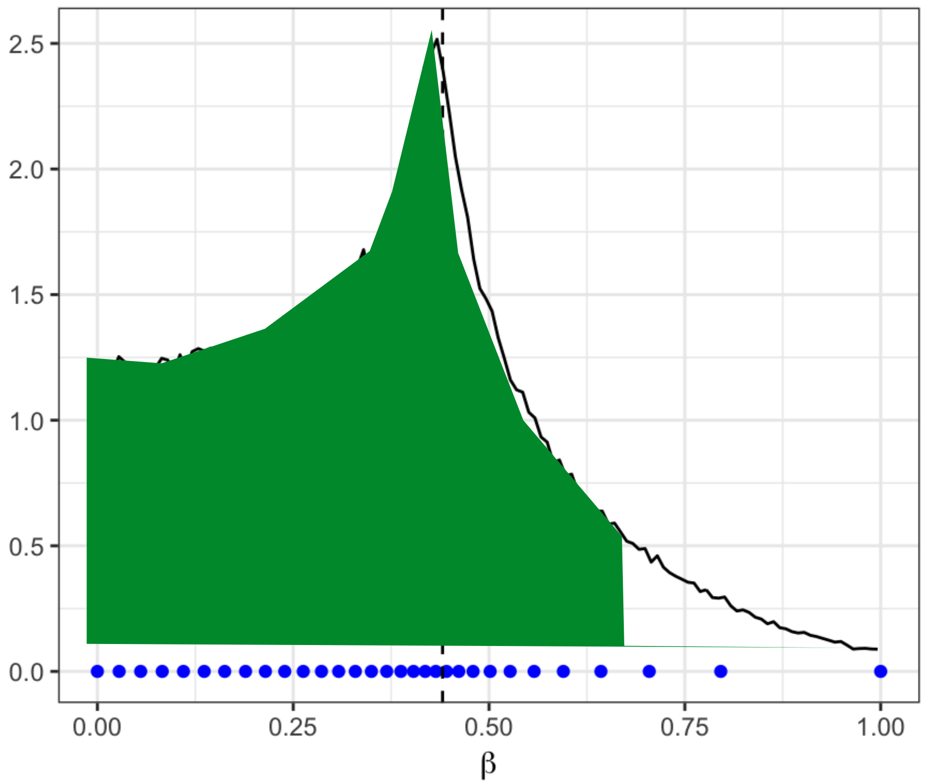


- How to estimate bounce rate λ ?
- Typically, quantities used in the study of MCMC are difficult to estimate from MCMC output (e.g., ESS, spectral gaps, mixing time, etc)
- In contrast, λ admits several nice estimators:
  - Method 1:

$$\lambda(\beta) = \mathbb{E}|\ell(X_\beta) - \ell(Y_\beta)|$$

$$X_\beta, Y_\beta \overset{\text{iid}}{\sim} \pi_\beta \qquad \ell(x) = \log(\text{likelihood}(x))$$

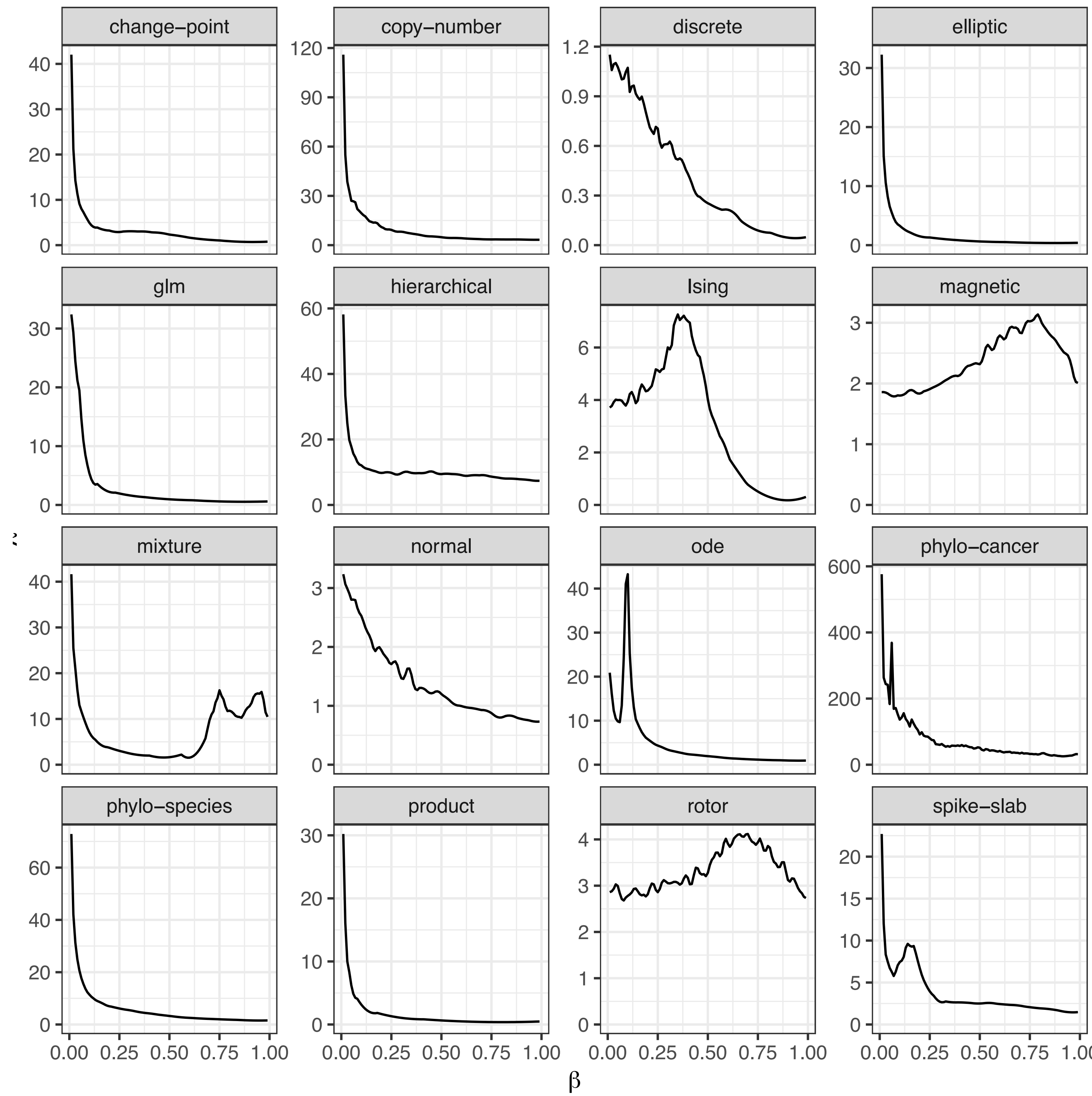
Bounce rate: estimation



- Estimating λ, method 2:
  - use equivalent cumulative barrier $\Lambda(\beta) \;=\; \int_0^\beta \lambda(\beta')\mathrm{d}\beta'$
  - which admit a simple estimator from empirical rejection rates

$$\hat{\Lambda}(\beta_i) = \sum_{j=1}^{i} \hat{r}^{(j-1,j)},$$

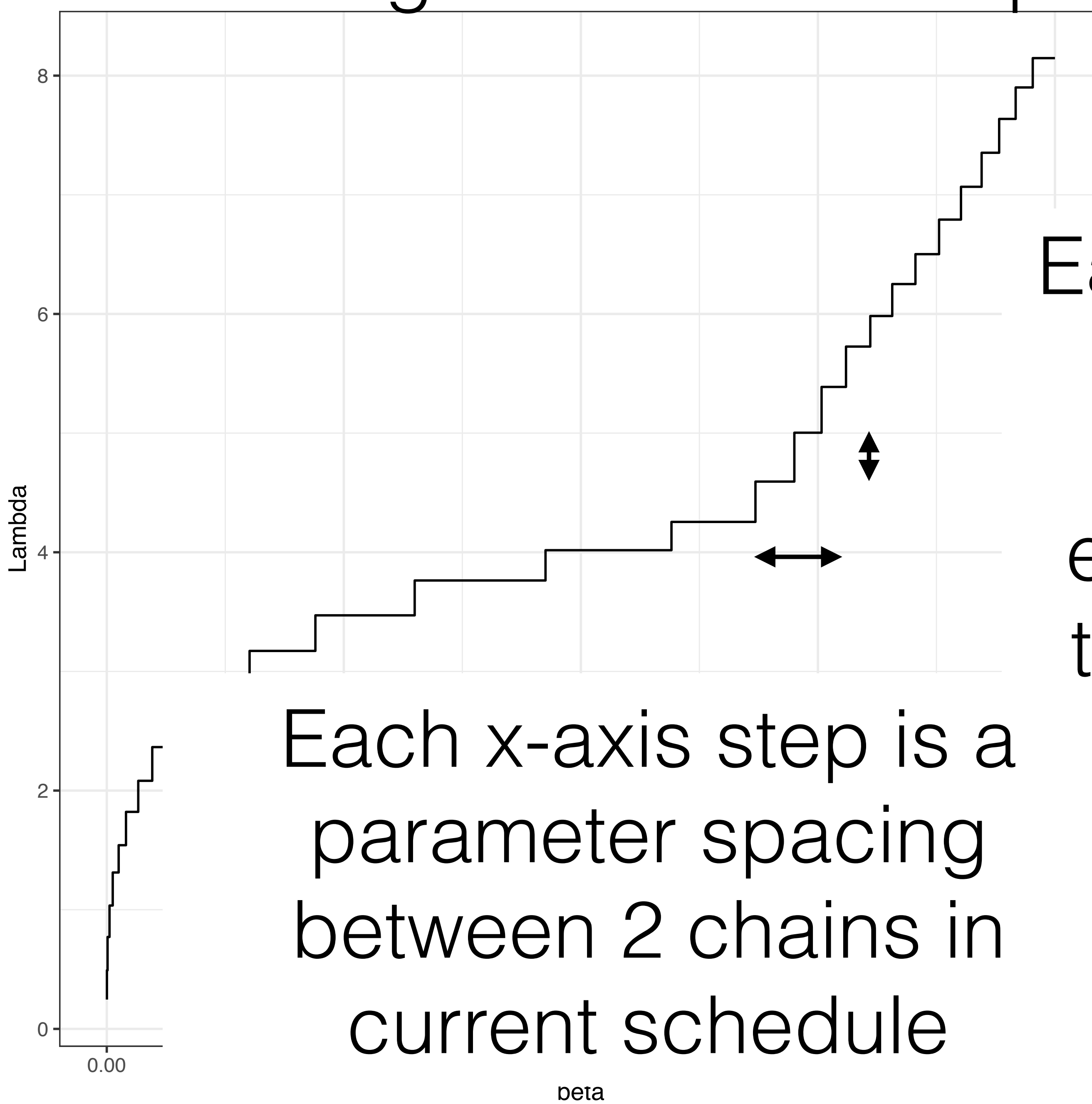
Some examples of local barriers in various models



Outline of schedule optimization algorithm derived from method 2

1: Start with an initial schedule and run PT for *n* iteration

2: compute the following cumulative swap rejection statistics:



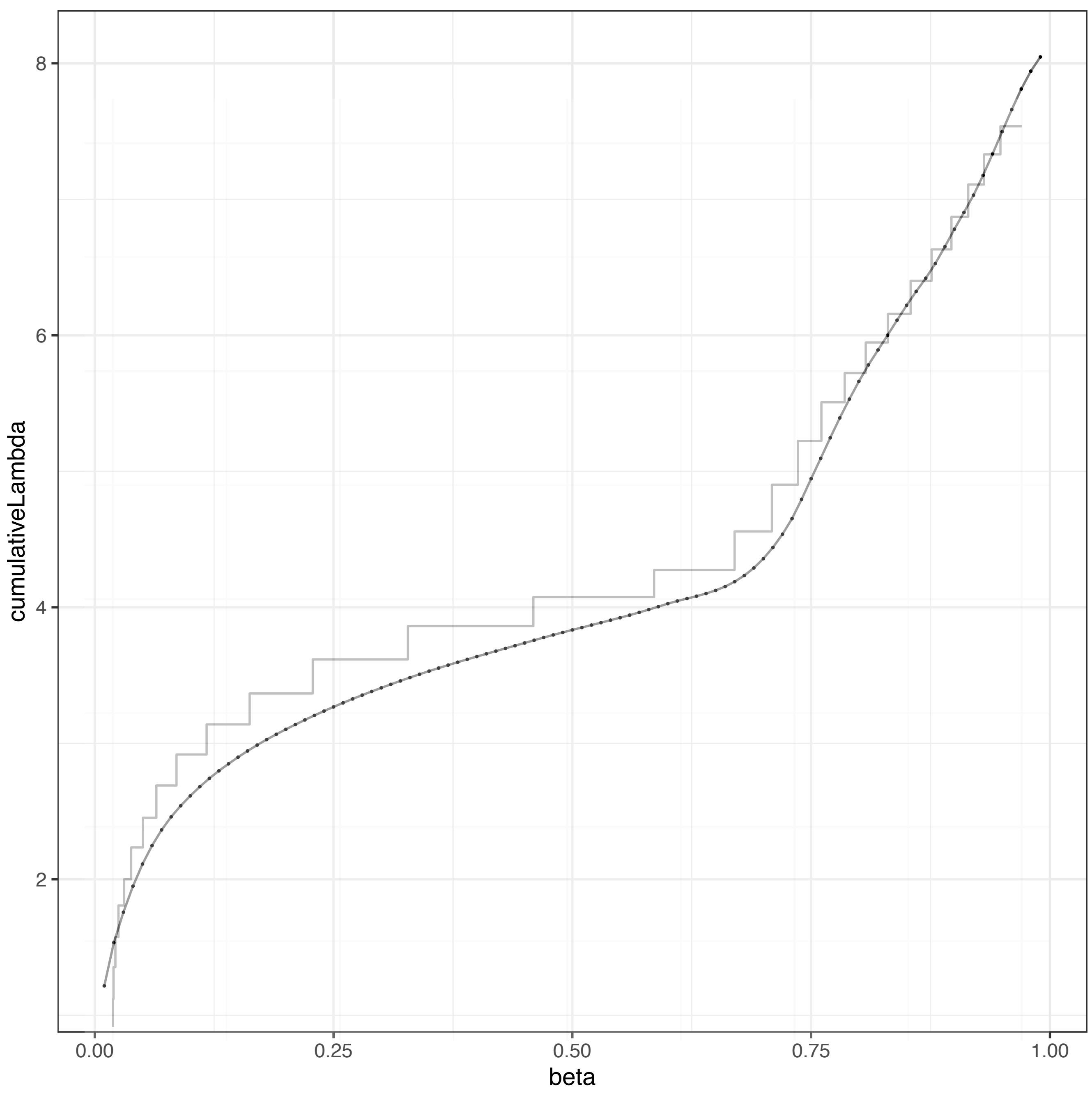
Each y-axis step is a rejection probability estimated from the *n* iterations

Each x-axis step is a parameter spacing between 2 chains in current schedule

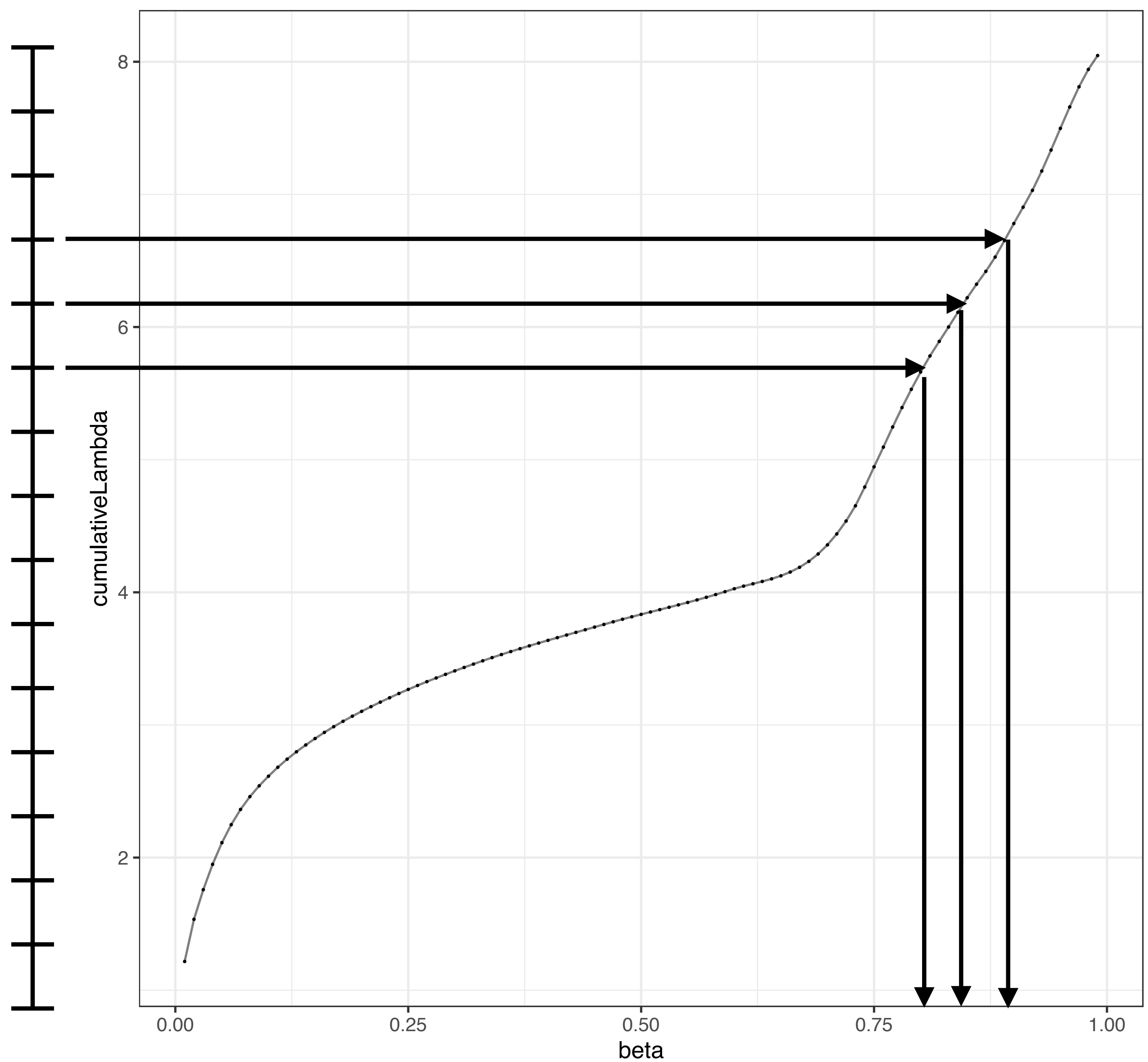
1: Start with an initial schedule and run PT for *n* iteration

2: compute the following cumulative swap rejection statistic

3: Fit a monotonic increasing smooth function



4: Project the uniform grid through the inverse of the smooth function - this yields an updated schedule



Outline of schedule optimization algorithm derived from method 2

1: Start with an initial schedule and run PT for *n* iteration

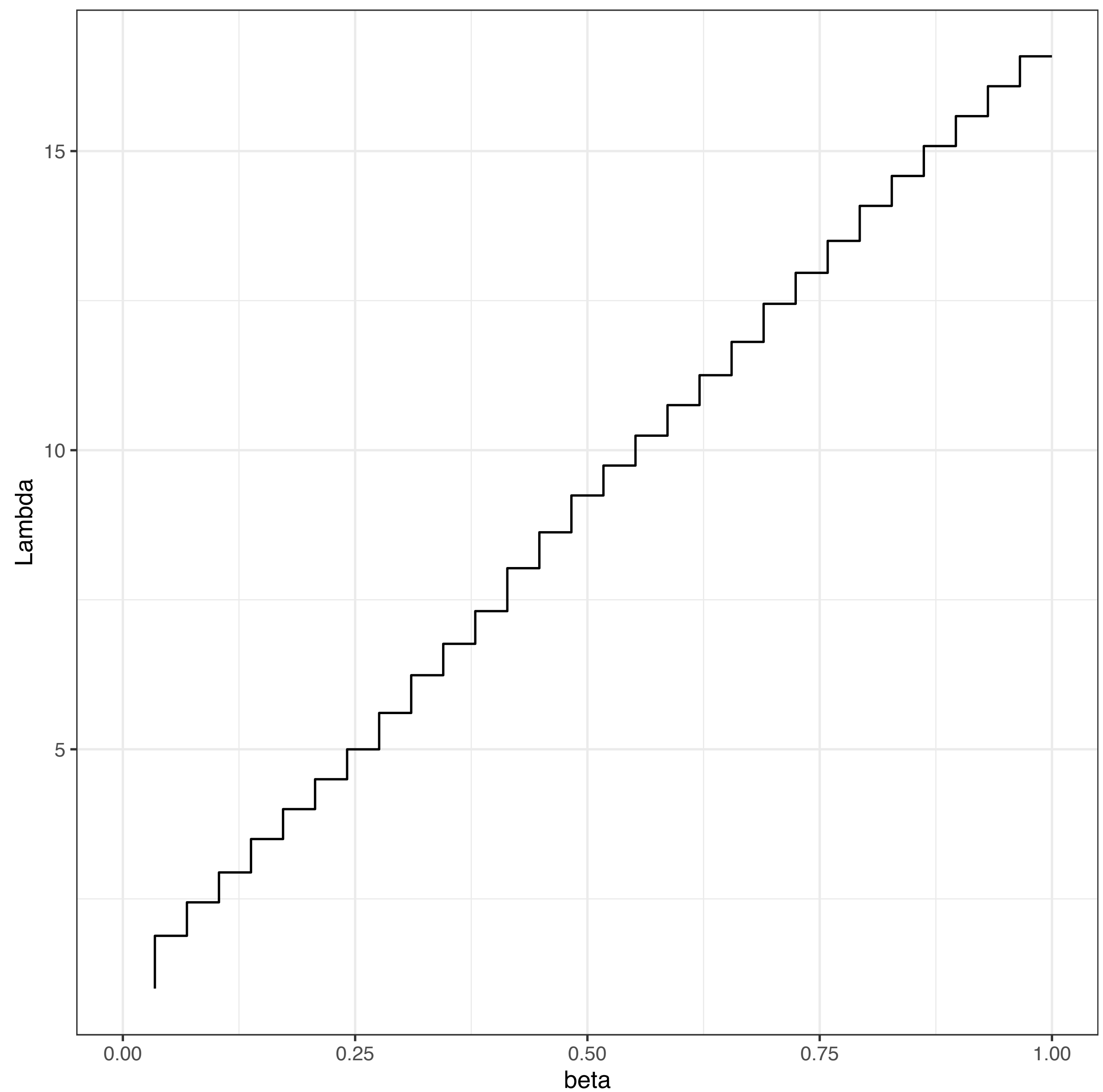
2: compute the following cumulative swap rejection statistics:

3: Fit a monotonic increasing smooth function

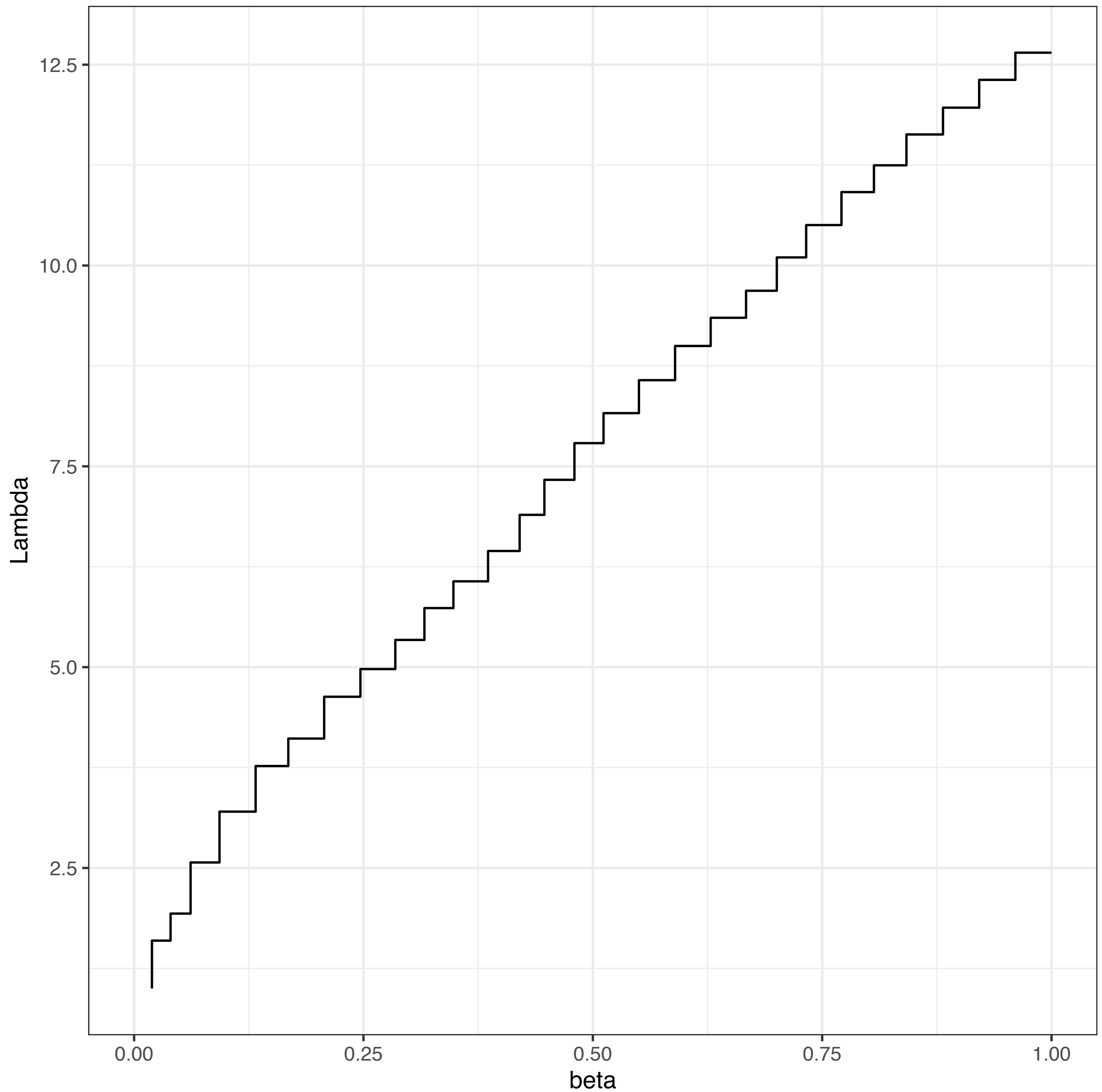
4: Project the uniform grid through the inverse of the smooth function - this yields an updated schedule

5: go to 1 using the new updated schedule and *n* := 2*n*

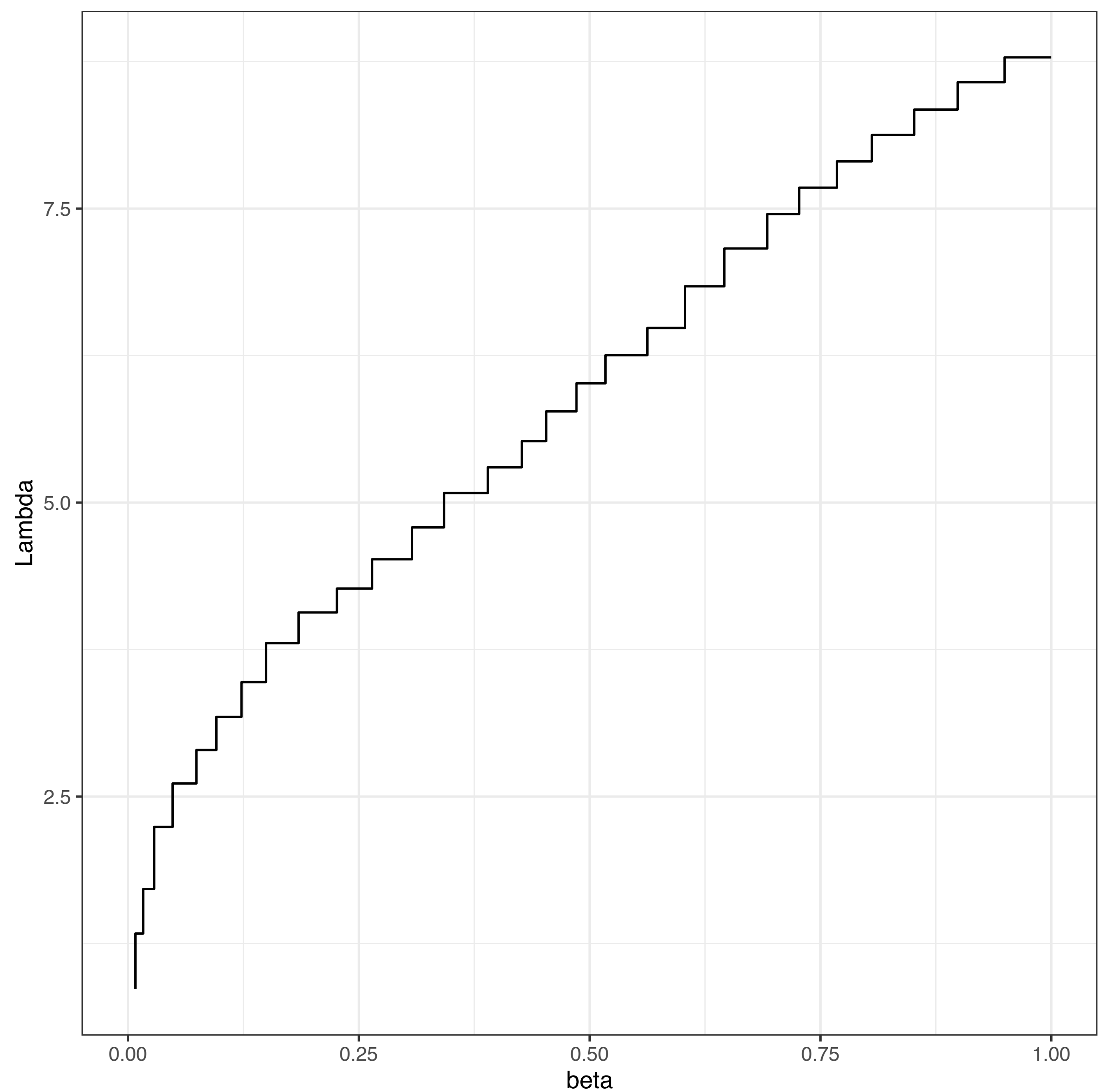
Example: Bayesian mixture model



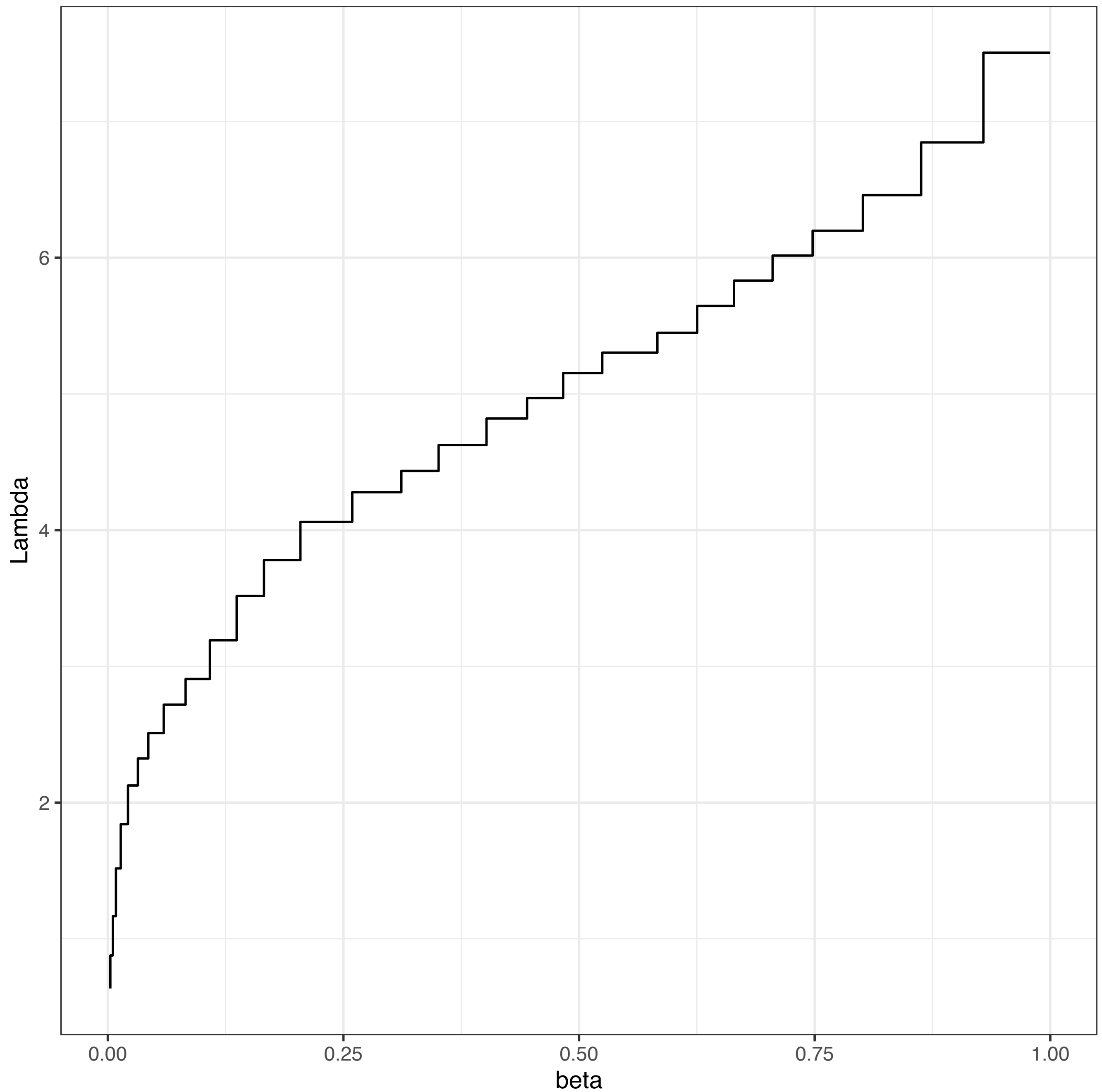
Example: Bayesian mixture model



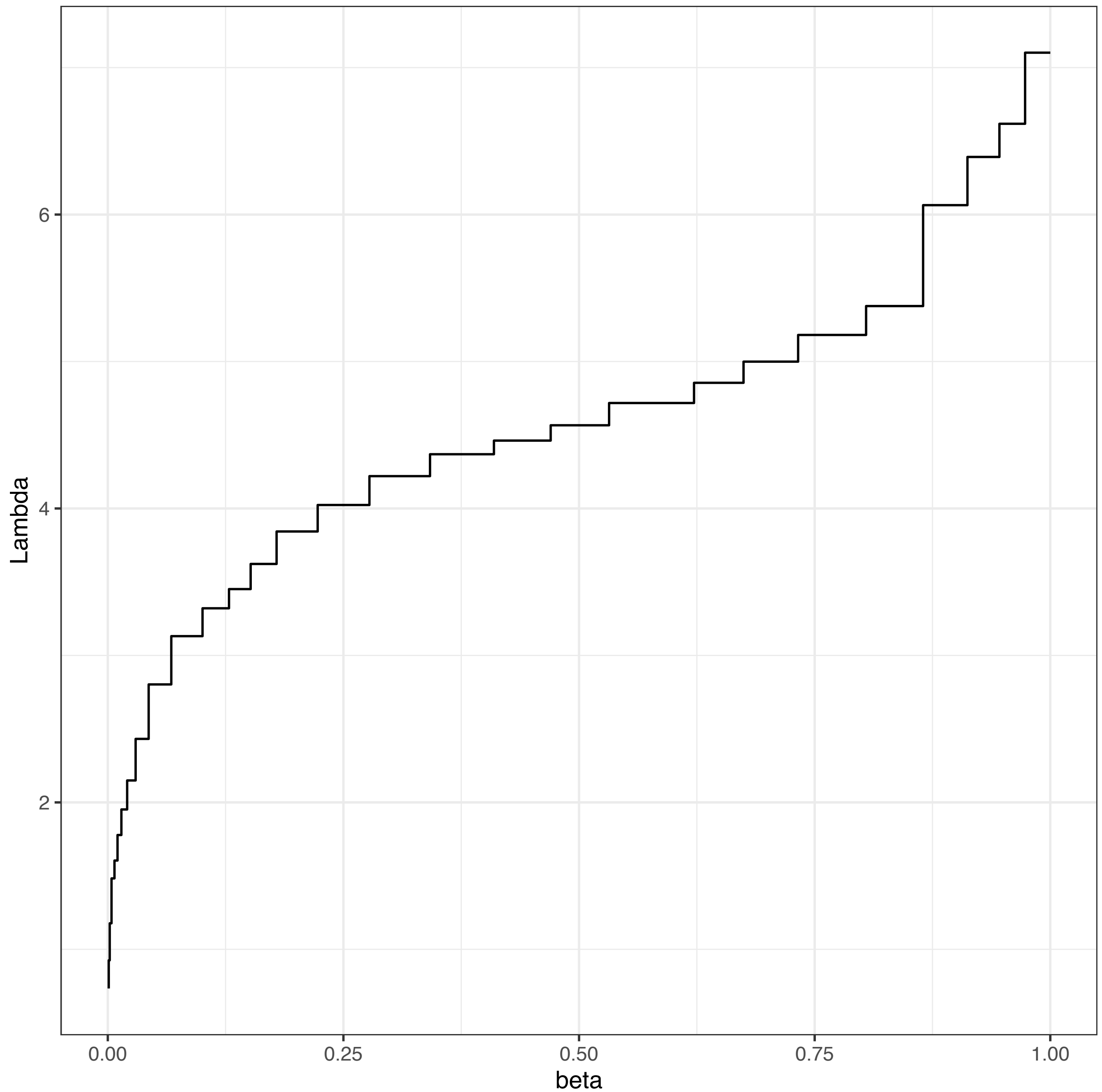
Example: Bayesian mixture model



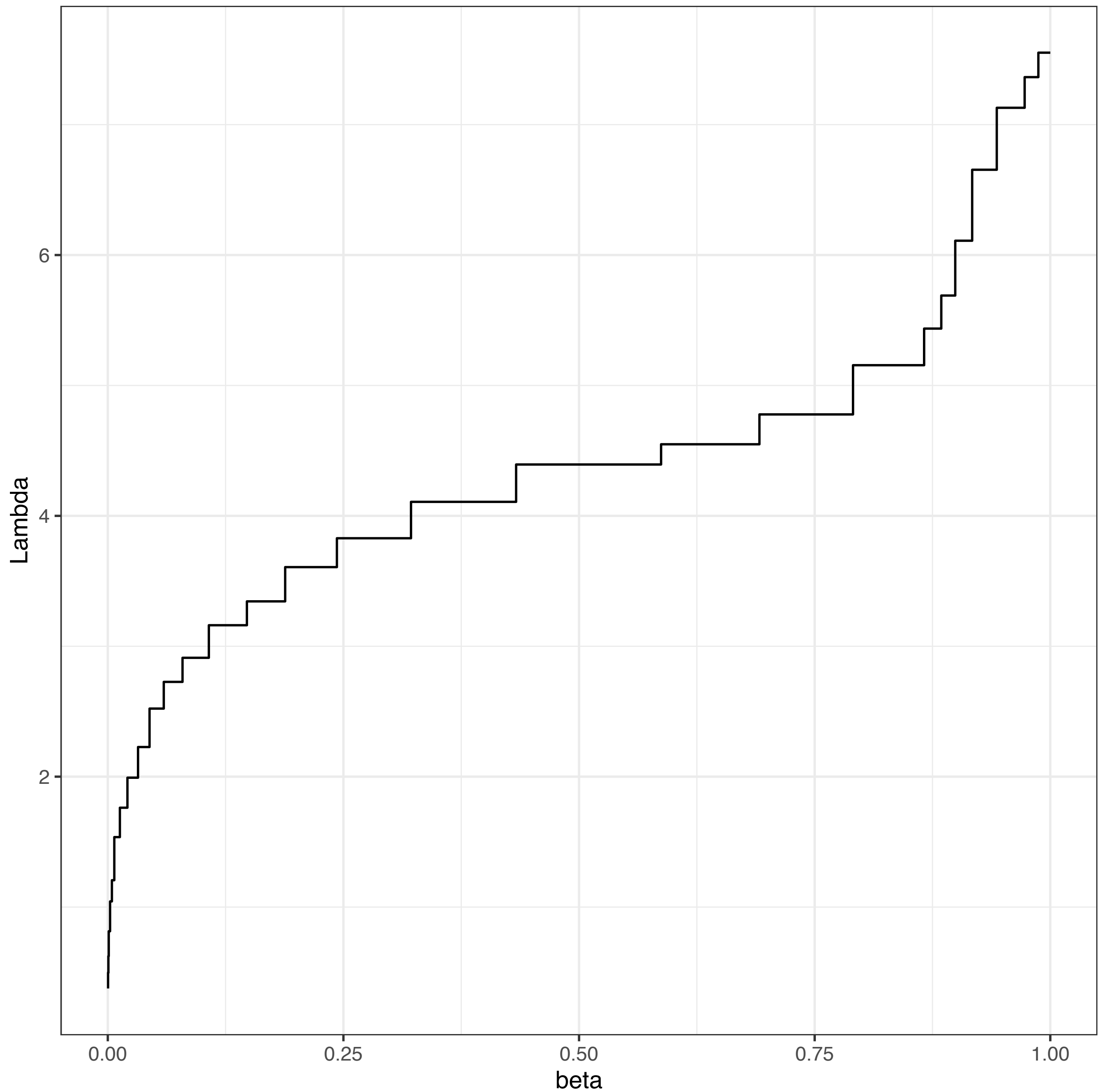
Example: Bayesian mixture model



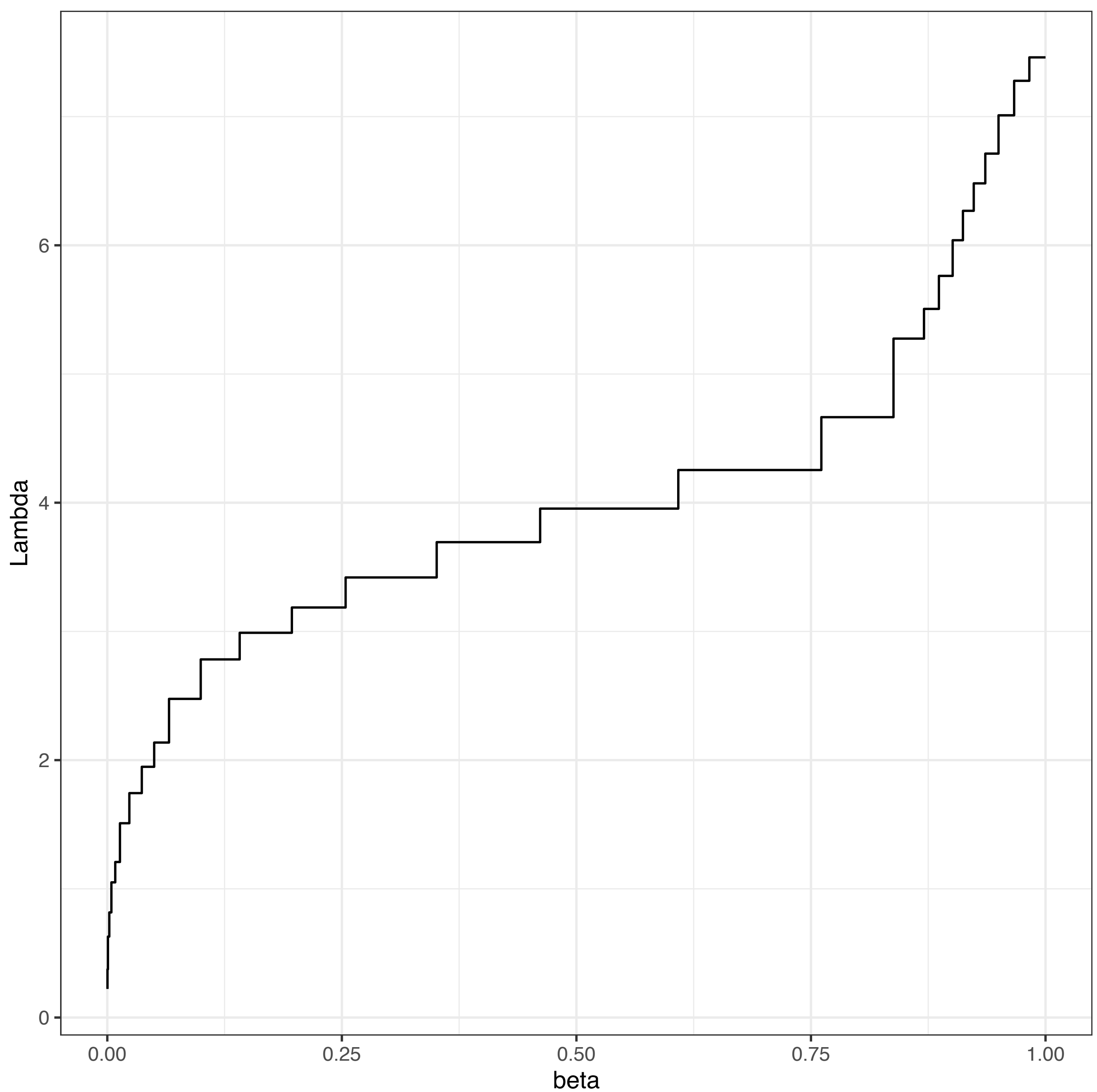
Example: Bayesian mixture model



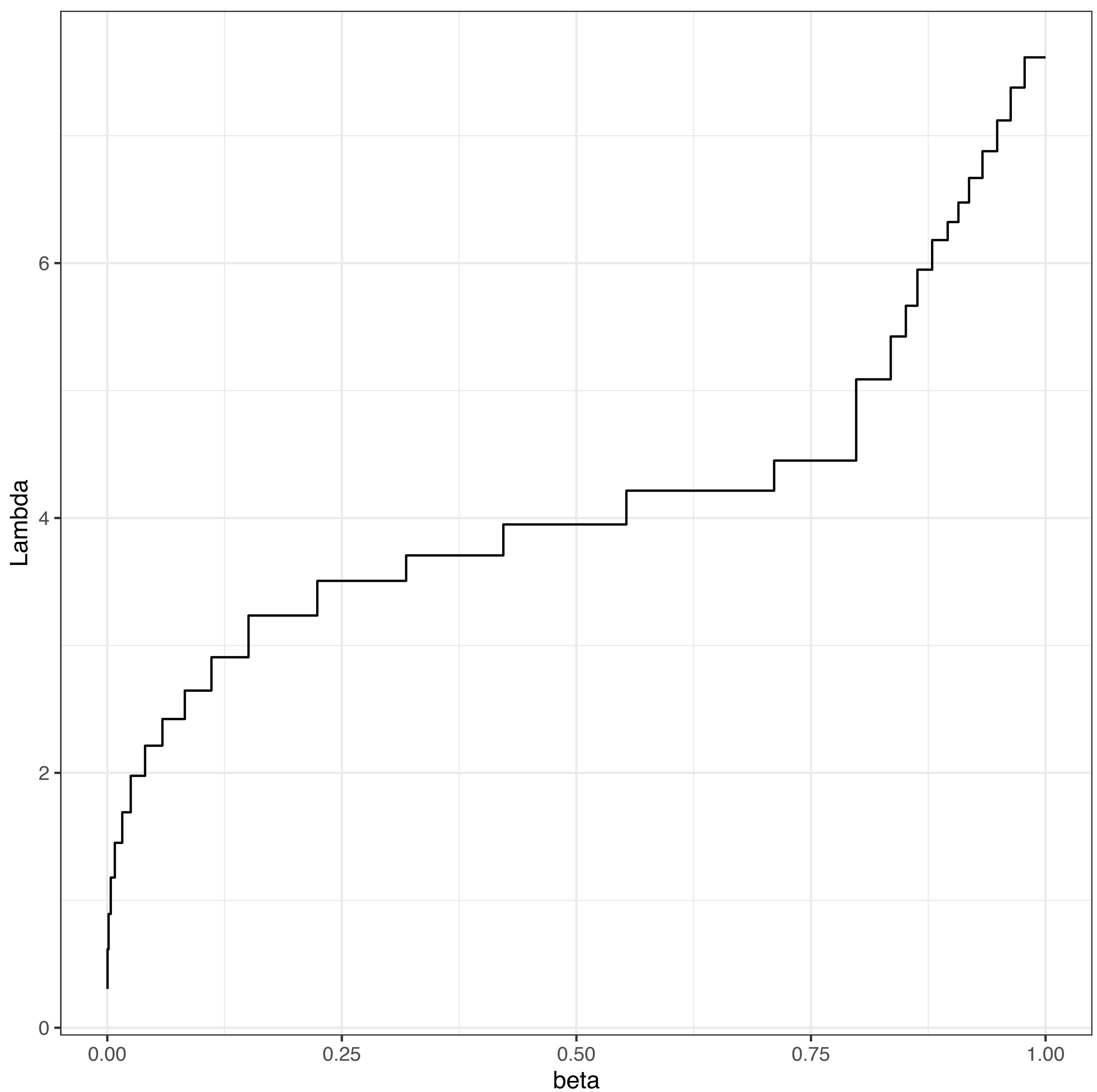
Example: Bayesian mixture model



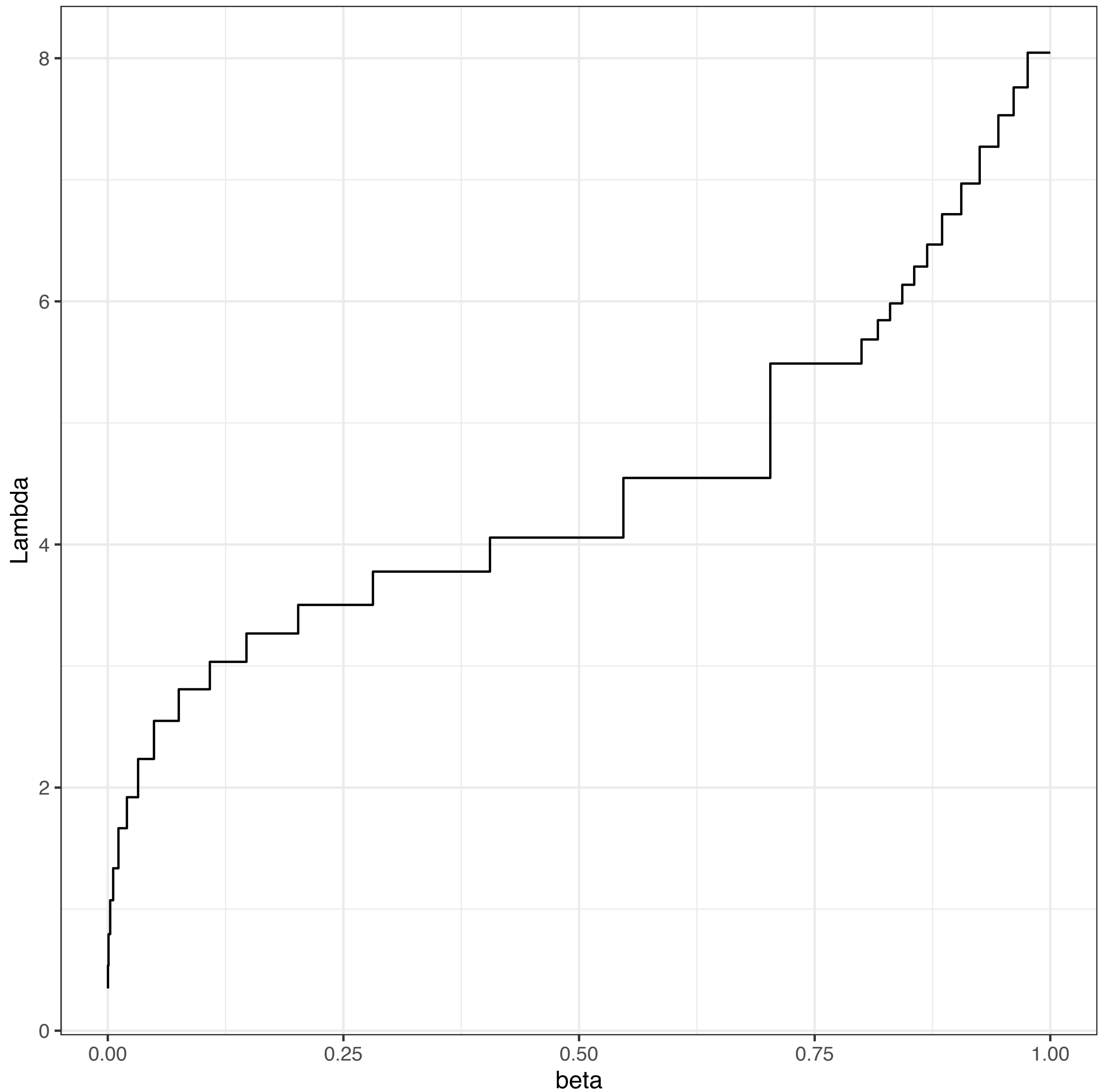
Example: Bayesian mixture model



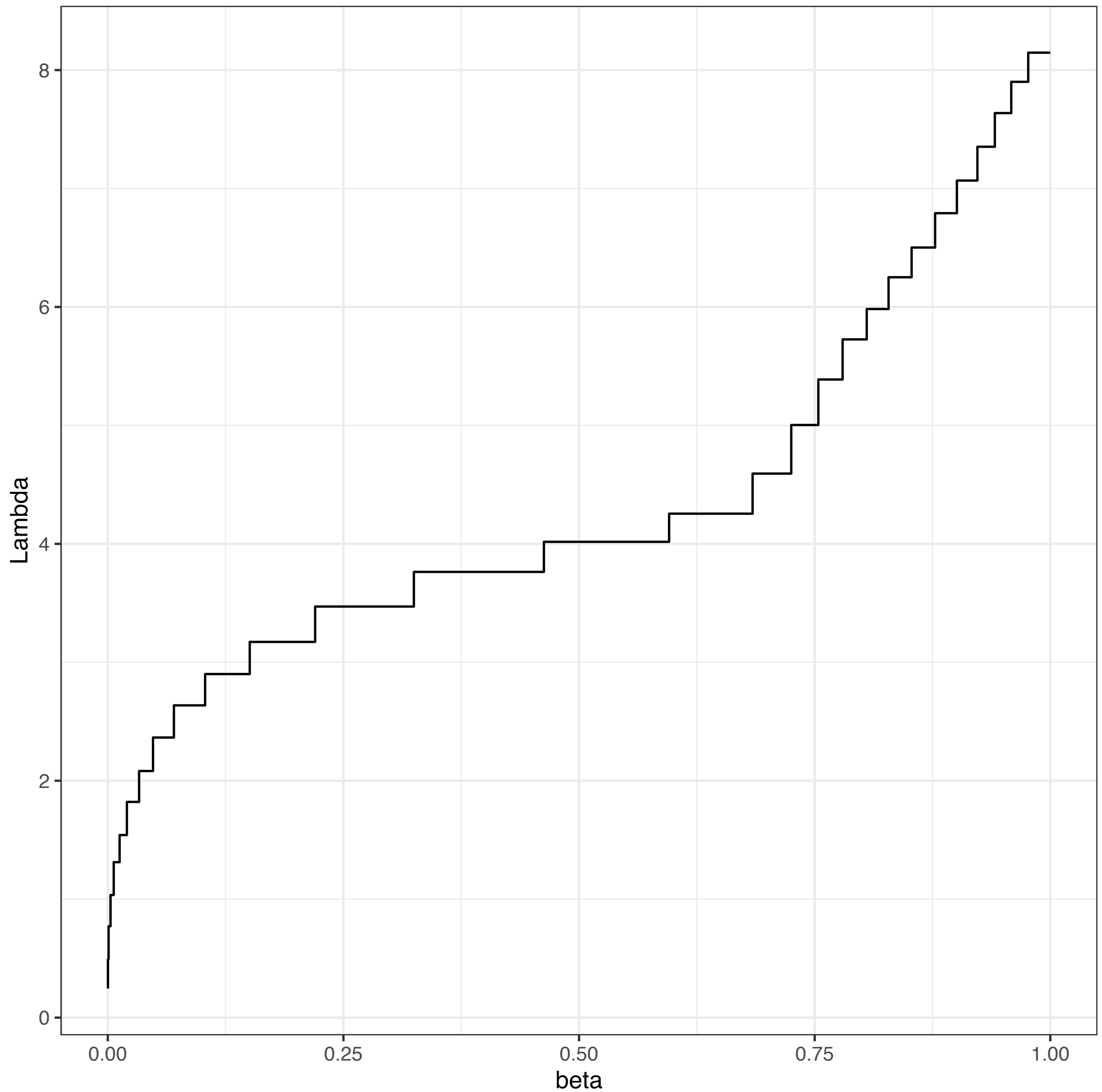
Example: Bayesian mixture model



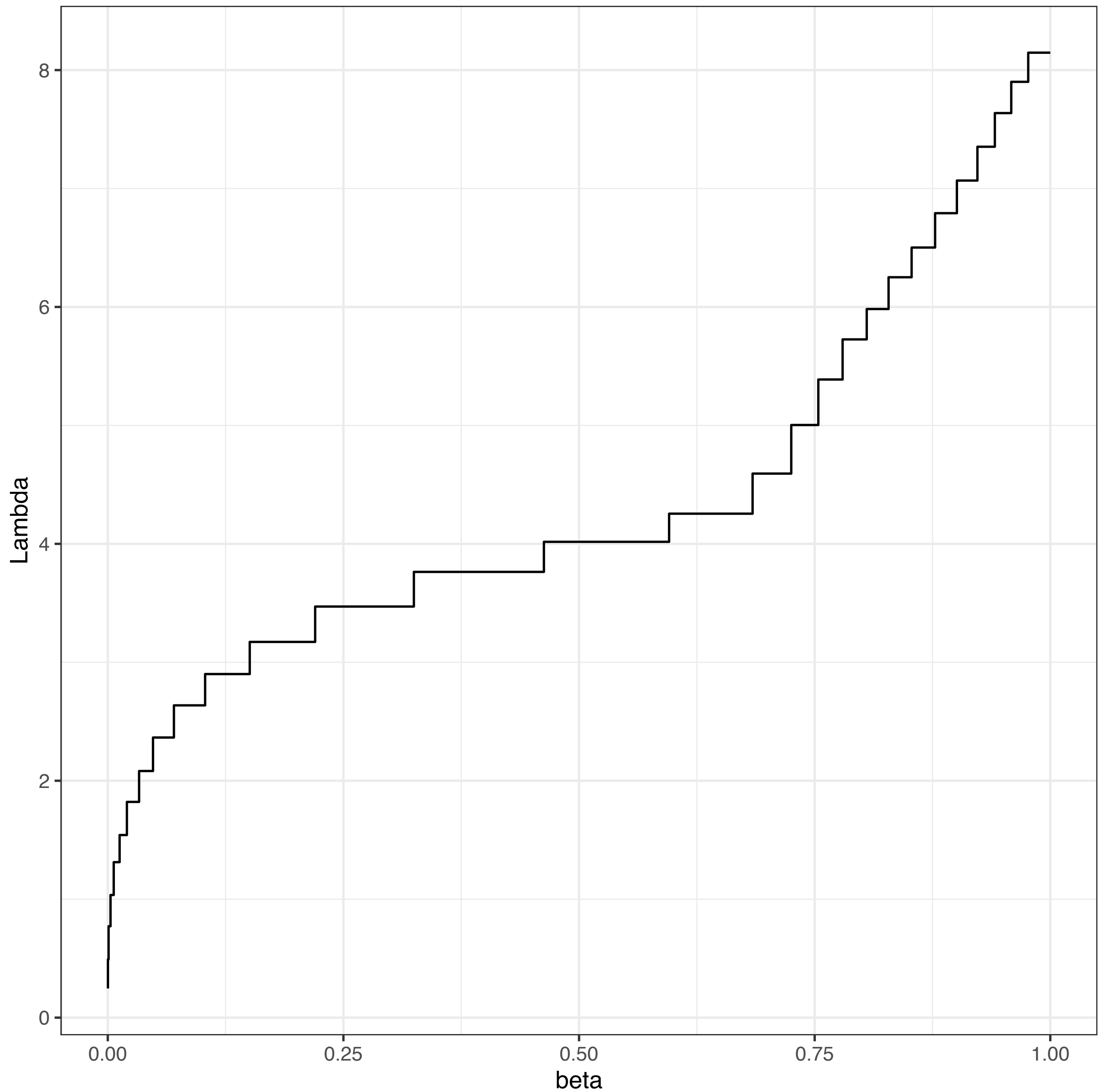
Example: Bayesian mixture model



Example: Bayesian mixture model



Example: Bayesian mixture model



The adaptive scheme performs well in a wide range of models